UNIVERSIDADE DE CAXIAS DO SUL
ÁREA DO CONHECIMENTO DE CIÊNCIAS EXATAS E ENGENHARIAS

ESTER BACCEGA

EVOLUÇÃO DO INTERGENICDB: BANCO DE DADOS DE REGIÕES INTERGÊNICAS DE BACTÉRIAS GRAM-NEGATIVAS

BENTO GONÇALVES
2020

ESTER BACCEGA

EVOLUÇÃO DO INTERGENICDB: BANCO DE DADOS DE REGIÕES INTERGÊNICAS DE BACTÉRIAS GRAM-NEGATIVAS

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Bacharel em Ciência Da Computação na Área do Conhecimento de Ciências Exatas e Engenharias, Campus Universitário da Região dos Vinhedos, da Universidade de Caxias do Sul.

Orientador: Prof. Dr. Daniel Luis Notari
Co-orientador: Me. Jovani Dalzochio

BENTO GONÇALVES
2020

ESTER BACCEGA

EVOLUÇÃO DO INTERGENICDB: BANCO DE DADOS DE REGIÕES INTERGÊNICAS DE BACTÉRIAS GRAM-NEGATIVAS

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Bacharel em Ciência Da Computação na Área do Conhecimento de Ciências Exatas e Engenharias, Campus Universitário da Região dos Vinhedos, da Universidade de Caxias do Sul.

Aprovado em 03/12/2020

Banca examinadora

_____
Prof. Dr. Daniel Luis Notari
Universidade de Caxias do Sul – UCS

_____
Prof. Dra. Helena Graziottin Ribeiro
Universidade de Caxias do Sul – UCS

_____
Prof. Dra. Scheila de Ávila e Silva
Universidade de Caxias do Sul – UCS

AGRADECIMENTOS

Agradeço aos meus pais, Marlene e Samoel *in memorian,* que sempre me ajudaram sem medir esforços e me apoiaram em todos os momentos da minha vida acadêmica. Sem vocês nada disso seria possível.

Agradeço as minhas irmãs, Luana *in memorian* e Juliana, que sempre estiveram perto quando precisei, me ouvindo, incentivando e sendo companheiras.

Agradeço ao meu orientador, Daniel, responsável por me ajudar com esta monografia, com toda a paciência e dedicação nas reuniões.

Agradeço aos amigos que a universidade me deu, Guilherme e Thaís, por estarem comigo em muitas aulas, ajudando, compartilhando experiências e risadas. Essa trajetória foi mais leve por causa com vocês.

Agradeço ao meu namorado, Carlos Eduardo, por toda a ajuda revisando o texto dessa monografia.

Agradeço a todos os professores, que me ajudaram em todas as aulas. Não teria chegado até aqui se não fosse por vocês.

“*Tente uma, duas, três vezes e se possível tente a quarta, a quinta e quantas vezes for necessário. Só não desista nas primeiras tentativas, a persistência é amiga da conquista. Se você quer chegar aonde a maioria não chega, faça o que a maioria não faz.* ”

Bill Gates

RESUMO

O banco de dados IntergenicDB contém informações sobre regiões intergênicas de bactérias gram negativas. Foi desenvolvido pelo grupo de pesquisa em Bioinformática da Universidade de Caxias do Sul, a ferramenta MMDBImportTool foi criada para popular os bancos de dados a partir dos dados extraídos dos bancos de dados públicos Genbank e Kegg. Nos últimos anos ocorreram mudanças no acesso e nos arquivos de bactérias disponibilizados pelo Genbank. Com isso, é preciso verificar se os dados armazenados no IntergenicDB estão coerentes com o GenBank. A partir disso, essa monografia tem como objetivo evoluir a ferramenta de importação MMDBImportTool. Foi verificada a necessidade de remodelagem de alguns processos da ferramenta de importação para que fosse possível a realização de uma nova carga de dados. Após o desenvolvimento e a documentação da MMDBImportTool, foi realizada uma nova carga no banco de dados. Essa nova carga conta com 118 organismos, 18 famílias e 9209 genes.

Palavras-chave: IntergenicDB. Bancos de dados biológicos. ETL. Genbank. Evolução de software.

ABSTRACT

The IntergenicDB database has information about intergenic sequences of gram negative bactérias. It was developed by the research team of Bioinformatics from University of Caxias do Sul, The tool called MMDBImportTool, was created to populate the database with data extracted from the public databases Genbank and Kegg. In the last years, a few changes had occurred on the bacteria's archives available by Genbank. Thereby, there is a need to verify if the data stored in IntergenicDB are consistent. Based on that, this monograph had the objective to develop an evolution of the importation tool MMDBImportTool. It was realized the need to remodel a few procedures of the tool, to be possible to do the new database load. After the development and documentation of the MMDBImportTool, a new data load was made in the database. The new data load has 118 organisms, 18 families and 9209 genes.

Palavras-chave: IntergenicDB. Biological database. ETL. Genbank. Software evolution.

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
BLL	*Business Logic Layoer*
BP	Pares de base (*Base Pairs*)
Bio-TIM	*Bioinformatics - Transparent Information Management*
Cardigan	*Cancer Relation Database for Integration and Genomic Analisys*
CDS	*CoDing Segment*
DAL	*Data Acess Layover*
DTO	*Data Transfer Layover*
DNA	Ácido Desoxirribonucleico
ETL	Extrair, Transformar, Carregar (*Extract, Transform, Load*)
FIOCRUZ	Fundação Oswaldo Cruz
FTP	Protocolo de Transferências de Arquivos (*File Transfer Protocol*)
GI	*GeInfo Identifier*
GMOD	*Generic Model Organism Database*
HGP	*The Human Genome Project*
IBM	*International Business Machines*
INSDC	*International Nucleotide Sequence Database Collaboration*
IOC	Instituto Oswaldo Cruz
KEGG	*Kyoto Of Genes and Genomes*
LGECG	Laboratório de Genômica e Expressão Gênica em Câncer
NCBI	*National Center for Biotchnology Information*
NIH	*Nacional Institutes of Health*
mRNA	RNA Mensageiro
RCSB	*Research Collaboratory for Structural Bioinformatics*
RefSeq	*Reference Sequence*
RNA	Ácido Ribonucleico
rRNA	RNA Ribossomal
SGBD	Sistema Gerenciador de Banco de Dados
SQL	*Structured Query Language*
TDD	Desenvolvimento Guiado por Testes (*Test Driven Development*)
URL	Localizador Padrão de Recursos (*Uniform Resource Locator*)
tRNA	RNA Transportador

LISTA DE FIGURAS

Figura 1 – Processo espiral da evolução de software.....17
Figura 2 – Demonstração de crescimento do Genbank.....19
Figura 3 – Conjuntos de dados do KEGG.....20
Figura 4 – Representação da macro arquitetura do Bio-TIM.....21
Figura 5 – Esquema Cardigan.....23
Figura 6 – Processo de ETL.....28
Figura 7 – Fluxo TDD.....30
Figura 8 – Fluxo do processo do importador de dados.....32
Figura 9 – Modelo ER.....33
Figura 10 – Modelo lógica.....34
Figura 11 – Tela principal da MMDBImportTool.....35
Figura 12 – Tela de configuração da MMDBImportTool.....35
Figura 13 – Sentidos *forward* e *reverse*.....35
Figura 14 – Exemplo de arquivo do Genbank.....37
Figura 15 – Estrutura da tag *Origin*.....38
Figura 16 – Raiz do diretório FTP.....39
Figura 17 – Estrutura de arquivos dentro da pasta específica.....40
Figura 18 – Raiz do diretório FTP atual.....40
Figura 19 – Estrutura de arquivos dentro da pasta específica atual.....41
Figura 20 – Estrutura de arquivos.....41
Figura 21 – Estrutura contendo os arquivos finais.....42
Figura 22 – Comparação dos arquivos GBK e GBFF usando o *Notepad* ++.....43
Figura 23 – Fluxo de importação.....45
Figura 24 – Especificação da opção "*full records*".....46
Figura 25 – *Download* versão anterior.....47
Figura 26 – *Download* versão atual.....47
Figura 27 – Inicialização das *Threads*.....48
Figura 28 – Criação da *Task*.....49
Figura 29 – Camadas do importador.....50
Figura 30 – Busca do arquivo *Prok Reference Genomes*.....51
Figura 31 – Procedimento de início da importação.....52
Figura 32 – Disponibilização dos arquivos.....53

Figura 33 – Procedimento de *download*.....53
Figura 34 – Procedimento de importação.....54
Figura 35 – Procedimento de formação arquivo FASTA.....54
Figura 36 – Alteração do script em *Python*.....55
Figura 37 – Procedimento de criação IntergenicDBFile.....56
Figura 38 – Procedimento de criação do arquivo sentido *forward*.....55
Figura 39 – Procedimento de separação dos dados.....57
Figura 40 – Procedimento para salvar.....58
Figura 41 – Procedimento de gravação das informações.....59
Figura 42 – Procedimento de chamada da gravação.....59
Figura 43 – *Insert* dos dados.....60
Figura 44 – Instalação do *Python*.....61
Figura 45 – Erro na cadeia de caracteres de entrada.....64
Figura 46 – Comparativo dos dados.....64
Figura 47 – Organismos incluídos.....65

LISTA DE QUADROS

Quadro 1 – Comparativo dos trabalhos relacionados.....27

SUMÁRIO

1 INTRODUÇÃO.....14
1.1 CONCEITOS.....14
1.2 PROBLEMA.....15
1.3 OBJETIVO.....15
1.4 ESTRUTURA DO TEXTO.....16

2 REVISÃO BIBLIOGRÁFICA.....17
2.1 EVOLUÇÃO DE SOFTWARE.....17
2.2 BANCOS DE DADOS BIOLÓGICOS.....19
2.2.1 Genbank.....19
2.2.2 Kegg.....20
2.2.3 Bio-Tim.....21
2.2.4 World Protein Database.....22
2.2.5 Cardigan.....23
2.2.6 Chado.....24
2.3 TRABALHOS RELACIONADOS.....25
2.4 ETL.....27
2.5 DESENVOLVIMENTO GUIADO POR TESTES.....29

3 INTERGENICDB.....31
3.1 CARGA DE DADOS.....31
3.2 MODELAGEM DE DADOS.....32
3.3 ARQUITETURA DE SOFTWARE.....34

4 PROPOSTA DE SOLUÇÃO.....36
4.1 ARQUIVO GENBANK.....36
4.2 PROBLEMA.....38
4.3 REQUISITOS FUNCIONAIS.....39
4.3.1 Diretório FTP.....39
4.3.2 Arquivo Genbank.....42

4.3.3 Threads....44
4.3.4 Importador MMDBImportTool....44
4.3.5 Testes....45
5 DESENVOLVIMENTO DA PROPOSTA DE SOLUÇÃO....46
5.1 DIRETÓRIO FTP....46
5.2 ARQUIVO GENBANK....47
5.3 THREADS....48
5.4 IMPORTADOR MMDBIMPORTTOOL....49
5.5 INSTALAÇÃO....60
5.5.1 Python e BioPython....60
5.5.2 Importador MMDBImportTool....61
5.6 TESTES....62
5.6.1 Testes realizados....62
5.6.2 Resultado dos testes....64
6 CONCLUSÃO....67
6.1 TRABALHOS FUTUROS....68

REFERÊNCIAS....69

1 INTRODUÇÃO

Este trabalho apresentará conceitos básicos de biologia molecular, bem como de bancos de dados biológicos. Além disso, serão abordados os conceitos de ETL (*Extract, Transform, Load*) e evolução de software, que serão empregados na proposta de solução para que seja desenvolvida uma nova versão do importador de dados utilizado no banco de dados IntergenicDB.

1.1 CONCEITOS

A biologia molecular computacional é responsável pelo desenvolvimento de algoritmos e programas que resolvem grandes problemas nessa área. Juntamente com a bioinformática, responsável pela aquisição, análise e armazenamento de informações biológicas - especialmente sob forma de ácidos nucleicos e proteínas, tem crescido muito nas últimas três décadas. Isso ocorre em razão do crescente número de projetos de sequenciamento de genomas (ZAHA, 2014).

As áreas da biologia molecular computacional e da bioinformática tem como objetivo converter as informações obtidas em conhecimento bioquímico e biofísico. Isso significa que, as informações são decodificadas de uma linguagem desconhecida, para uma linguagem que seja conhecida extraindo o significado biológico. A linguagem conhecida, pode ser composta por letras, que podem representar os nucleotídeos e aminoácidos, por palavras, que podem representar os motivos e por frases, que podem representar as proteínas (ZAHA, 2014).

As bactérias são seres unicelulares. Também são classificadas como organismos procarióticos. Ou seja, são seres de apenas uma única célula e não possuem membrana nuclear envolvendo o seu material genético. Por ser um ser procariontes, as bactérias são cerca de dez vezes menores que uma única célula eucarionte (ZAHA, 2014).

Um gene, considerando aspectos moleculares e funcionais, pode ser descrito simplificadamente como um segmento da molécula de DNA, que carrega as informações necessárias para codificar o produto de RNA (moléculas rRNA ou tRNA) ou proteico (mRNAs). O gene não é formado apenas pelas sequências codificadoras,

mas também pelas sequências reguladoras, que são diretamente responsáveis pela sua expressão (ZAHA, 2014).

O genoma é a informação genética de um organismo. Em formas de vidas celulares, como os seres procariotos e eucariotos, a informação genética está codificada no DNA de cada célula, tendo um cromossomo – molécula de DNA que constitui o genoma – para seres procariotos ou mais para seres eucariotos. É composto por duas regiões, a gênica e a intergênica. A região gênica, são codificadoras de proteínas enquanto as intergênicas são as regiões não codificadoras (ZAHA, 2014).

1.2 PROBLEMA

O banco de dados IntergenicDB foi criado para auxiliar o grupo de pesquisa em bioinformática da Universidade de Caxias do Sul, contendo informações sobre sequências intergênicas de bactérias gram-negativas. O IntergenicDB é populado pelos dados vindos dos bancos públicos Genbank e Kegg. Em 2014, o aluno Jovani Dalzochio realizou a primeira carga de dados de forma automatizada, utilizando a ferramenta de sua autoria, MMDBImportTool.

Para que o processo de importação ocorra, é necessário apontar um diretório FTP onde se encontram os arquivos. Após isso, são iniciados os *downloads* e a leitura, rodados scripts e gerados os arquivos finais para a importação dos dados no InternigenicDB. Porém, em 2019 o Genbank disponibilizou novos dados, juntamente com alterações no diretório FTP e nos arquivos.

1.3 OBJETIVO

Este trabalho tem como objetivo, a análise e o desenvolvimento das alterações necessárias para que a ferramenta de importação esteja apta a realizar uma nova importação de dados no IntergenicDB, além de melhorias no processamento para maior eficiência.

1.4 ESTRUTURA DO TEXTO

O Capítulo 2 apresenta a revisão bibliográfica, conceituando bancos de dados biológicos. Além disso, também são definidos os conceitos do método ETL, da evolução de software e do desenvolvimento guiado por testes.

O Capítulo 3 apresenta o IntergenicDB, com o procedimento de carga de dados, modelagem de dados e a arquitetura de software.

O Capítulo 4 apresenta a proposta de solução, com o problema e os requisitos para evolução do IntergenicDB.

O Capítulo 5 apresenta o desenvolvimento da proposta de solução, com as alterações referentes ao diretório FTP, arquivo Genbank, *threads* e a documentação da MMDBImportTool. Apresenta também as instruções para instalação da ferramenta.

O Capítulo 6 apresenta as conclusões finais do trabalho.

2 REVISÃO BIBLIOGRÁFICA

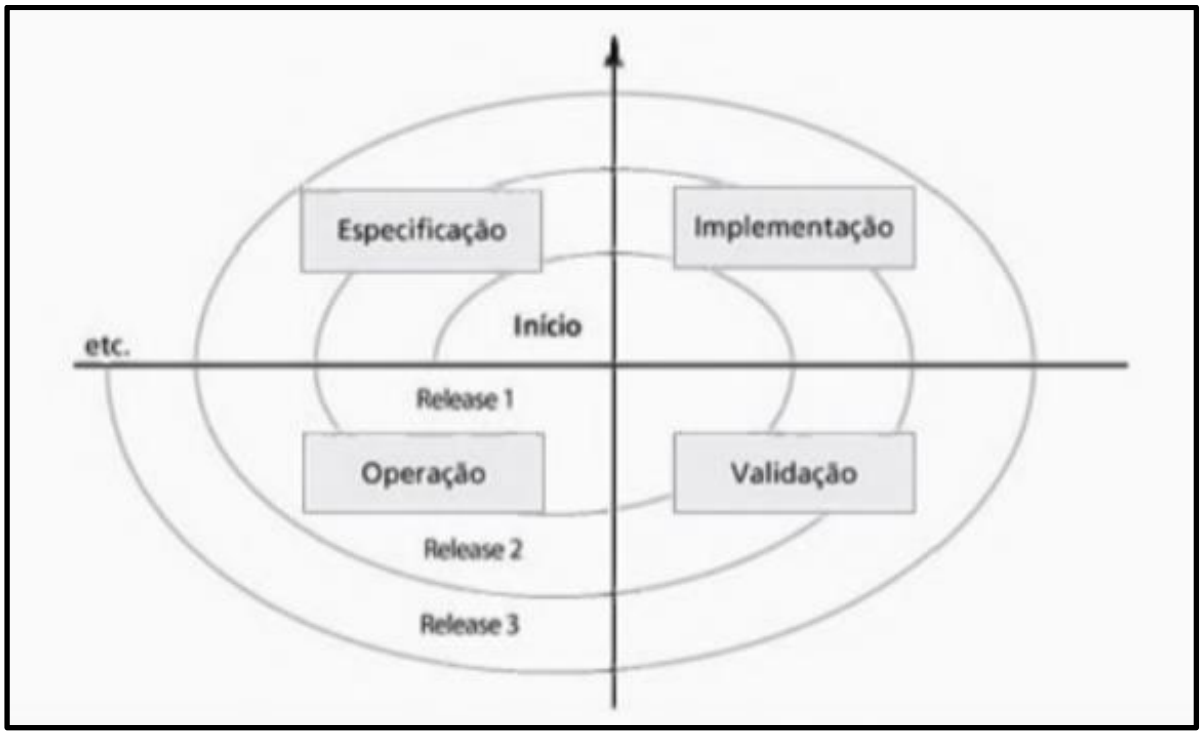
Este capítulo apresentará os conceitos de evolução de software, método ETL e desenvolvimento guiado por testes. Também apresentará os bancos de dados biológicos primários e secundários, qual tipo de informação carregam e seu funcionamento.

2.1 EVOLUÇÃO DE SOFTWARE

Segundo Sommervile (2011), o desenvolvimento do software não é interrompido quando é entregue, pois precisa continuar evoluindo durante toda sua vida útil. Quando um software é desenvolvido, é inevitável a necessidade de mudanças para que continue sendo utilizado, pois novas necessidades surgem conforme há mudanças nas regras de negócio, expectativas dos usuários, novos requisitos e funcionalidades, alterações para melhor processamento no hardware e dentro do próprio software, e também correção de erros que possam ter sido encontrados. Desta forma, uma evolução de software pode ocorrer por vários motivos e raramente é considerada de forma isolada, já que uma mudança ambiental pode levar a uma mudança no sistema, o que pode então, provocar novamente uma mudança ambiental.

A vida útil de um software costuma ser longa, cerca de 10 anos para softwares de negócios, mas isso só é possível com as alterações necessárias, que são os chamados *releases*. Para Sommervile (2011), a engenharia de software deve ser pensada como um espiral que contém requisitos, projetos, implementação e testes, os quais irão acontecer durante toda a vida útil do sistema, pois quando o *release* 1 é entregue o desenvolvimento do *release* 2 começa a ser feito imediatamente, conforme Figura 1. Com isso, a necessidade da evolução de software pode se tornar óbvia antes de um *release* ser entregue já que em muitos casos os *releases* posteriores começam a ser desenvolvidos antes dos *releases* atuais.

Figura 1 – Processo espiral da evolução de software



Fonte: Sommervile (2011).

Se for realizada uma comparação entre softwares e hardwares, segundo Pressman (2011), quando um hardware se desgasta, ele é simplesmente substituído por uma nova peça, mas para o software, não existe essa substituição. Quando um software está desgastado, é necessário que passe por um processo de análise para que seja desenvolvida ou corrigida uma funcionalidade, fazendo com que essa manutenção do software tenha uma complexidade muito maior do que a evolução do hardware.

Há ainda uma visão alternativa sobre a evolução de software, que de acordo com Rajlich e Bennett[1] (2000 apud SOMMERVILE 2011), o qual seria distinguido “evolução” e “em serviço”. Neste caso, a evolução seria a fase com mudanças significativas, como por exemplo alterações na arquitetura do software e funcionalidades que podem ser incluídas. Enquanto na fase em serviço, seriam apenas mudanças pequenas e essenciais para o funcionamento. Desta forma, durante a primeira fase, a evolução, o software ainda é usado com sucesso e sempre com um grande fluxo de alterações a serem realizadas, até chegar em um certo ponto

[1] RAJLICH, Vaclav T.; BENNETT, Keith. H. **A Staged Model for the Software Life Cycle**. 7. ed. 2000.

que já não é mais rentável continuar a implementar novas funcionalidades neste software. Então, é iniciada a fase em serviço, na qual o software ainda é utilizado, mas não são mais realizadas grandes alterações, com exceção das pequenas regras de negócio. Durante essa fase, é iniciada a substituição total deste software.

2.2 BANCOS DE DADOS BIOLÓGICOS

As informações biológicas são difíceis de trabalhar pela quantidade e complexidade dos dados. Como exemplo pode-se utilizar o genoma humano o qual, de acordo com o *The Human Genome Project* (HGP)[2], é composto por aproximadamente 3 bilhões de bases de DNA (HOOD; ROWEN, 2013). Desta forma, para guardar e consultar as sequências, foram organizados como bancos de dados biológicos. Apesar de parecer uma solução simples, o banco biológico é complexo, heterogêneo, dinâmico, mas, mesmo assim, inconsistente devido à falta de padrões em nível ontológico (BERTOLDI, 2018).

Os bancos de dados biológicos podem ser considerados primários ou secundários. Primários são aqueles que contêm apenas a informação ou a estrutura. Já os secundários são aqueles derivados dos primários (ZAHA, 2014).

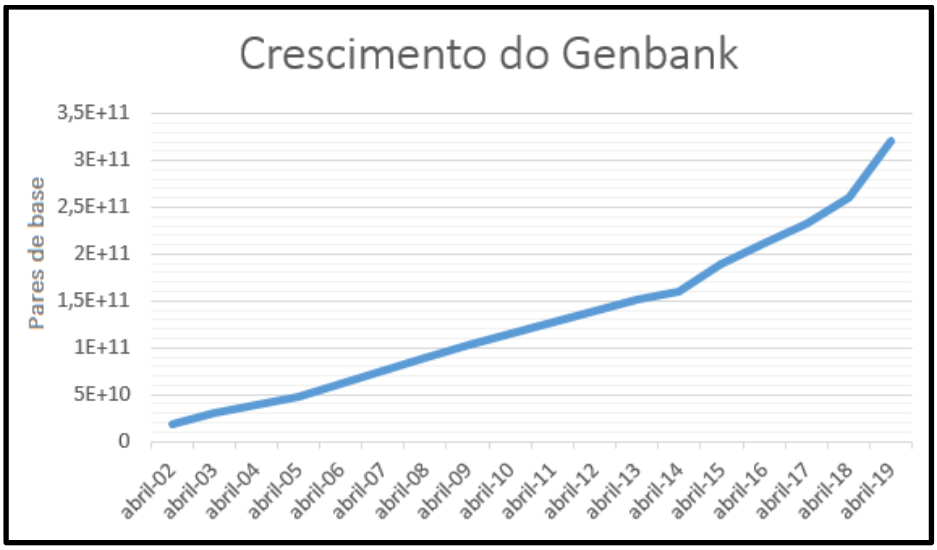
Nas seções a seguir, serão apresentados bancos biológicos primários, como o Genbank e Kegg. Também serão apresentados bancos biológicos secundários, como o Bio-TIM, World Protein Database, Cardigan e Chado.

2.2.1 Genbank

Um dos principais bancos de dados biológicos primário é o Genbank[3], um banco de dados públicos mantido pelo *National Center for Biotechnology Information* (NCBI), que vem crescendo continuamente ano a ano, como mostra a Figura 2 (SAYERS et al, 2018), e sendo utilizado por muitos outros bancos, como o Bio-TIM (OLIVEIRA, 2016) e o Protein World Database (TRISTÃO e LIFSCHITZ, 2009).

[2] https://www.genome.gov/human-genome-project
[3] https://www.ncbi.nlm.nih.gov/genbank/

Figura 2 - Demonstração de crescimento do Genbank



Fonte: Adaptado de site Genbank[4] (2020).

O Genbank, sendo um banco de dados públicos, possui informações variadas de sequências de DNA que são obtidas principalmente através de submissões individuais realizadas por laboratórios ou por lotes em larga escala. Porém muitas vezes essas informações são armazenadas de maneiras primitivas, com pouca ou nenhuma preocupação em relação ao desempenho, já que os dados não são organizados de forma eficiente (OLIVEIRA, 2016). Sendo o Genbank um banco de dados primário, é utilizado na criação de outros bancos, como é o caso do Bio-TIM, pois a criação de um novo de banco de dados a partir das consultas otimiza o tempo de processamento já que os dados podem ser organizados e armazenados de formas que facilitem a extração da informação para a pesquisa.

2.2.2 Kegg

O *Kyoto Encyclopedia of Genes and Genomes* (KEGG)[5], segundo o seu próprio *site* define, é um banco de dados biológico para entender as funções e utilidades de um sistema biológico de alto nível a partir de informações genômicas e moleculares.

[4] https://www.ncbi.nlm.nih.gov/genbank/statistics/
[5] https://www.genome.jp/kegg/

O KEGG possui conjuntos de dados, conforme Figura 3, que podem ser de diferentes tipos: estruturais de genes e proteínas (KEGG *Genes*), genomas (KEGG *Genome*), diagramas de ligação molecular e interação (KEGG *Pathway*), hierarquias, relações e papéis biológicos (KEGG *Brite*), entre outros, conforme Figura 3. O KEGG fornece uma base de conhecimento que pode ser usada para ligar genomas em sistemas biológicos e em ambientes (KANEHISA et al, 2018).

Figura 3 – Conjuntos de dados do KEGG

Categoria	Banco de dados	Conteúdo
Informações de sistemas	KEGG Pathway	Mapas pathway
	KEGG Brite	Hierarquias e tabelas
	KEGG Módulos	Módulos
Informação genômica	KEGG Ortologias (KO)	Órtologos Funcionais
	KEGG Genoma	KEGG organismos (genomas completos)
	KEGG Genes	Genes e proteínas
	KEGG SSDB	Semelhança da sequência genética
Informação química	KEGG Composto	Pequenas molécuals
	KEGG Glicano	Glicanos
	KEGG Reação	Reações bioquímicas
	KEGG Rclass	Classe de reação
	KEGG Enzimas	Nomeclatura enzimática
Informação de saúde	KEGG Rede	Doenças relacionadas aos elementos da rede
	KEGG Variável	Variáveis genéticas humanas
	KEGG Doença	Doenças humanas
	KEGG Medicamentos	Medicamentos
	KEGG Dgroup	Grupo de medicamentos
	KEGG Environ	Substâncias relacionadas a saúde

Fonte: Adaptado de site Kegg[6] (2020).

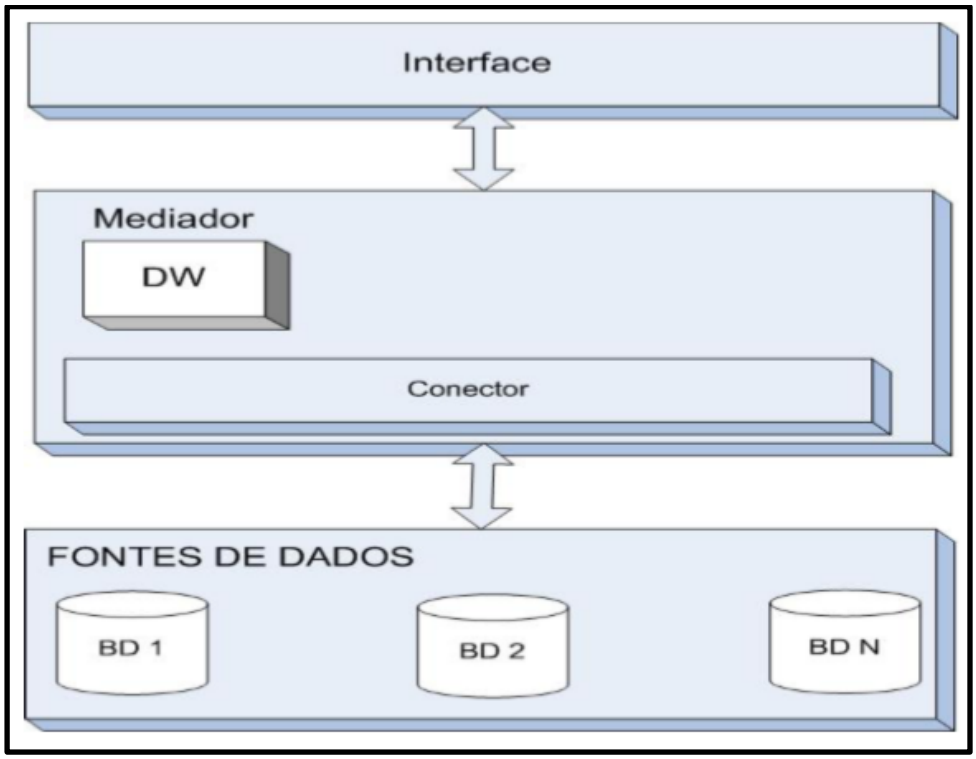
2.2.3 Bio-TIM

O banco de dados Bio-TIM (*Bioinformatics - Transparent Information Management*), é um projeto que consiste em especificar os requisitos no contexto genômico, modelar e implantar um *data warehouse* para armazenar de forma estruturada os dados que foram obtidos a partir do Genbank. As consultas ao banco poderão ser feitas tanto por biólogos e pesquisadores, quanto por outras ferramentas computacionais existentes.

[6] https://www.genome.jp/kegg/kegg1a.html

O Bio-TIM é desenvolvido em três macro camadas, conforme Figura 4. A primeira denominada fonte de dados, a qual contém os dados de diversas fontes organizados através de SGDB (Sistema Gerenciador de Bancos de Dados). A segunda, “mediador”, que são os softwares que farão a comunicação entre os dados e a aplicação. E, por fim, a última camada que é a interface (OLIVEIRA, 2006).

Figura 4 – Representação da macro arquitetura do Bio-TIM



Fonte: Oliveira (2016).

2.2.4 World Protein Database

A equipe de bioinformática do Laboratório de Genômica Funcional e Bioinformática do Instituto Oswaldo Cruz (IOC), FIOCRUZ possui o Projeto Comparação de Genomas, criado para a comparação do conteúdo de proteínas de genomas. O objetivo é calcular o grau de similaridades entre as sequências de organismos. Esse projeto resultou em parceria com a PUC-Rio, no banco de dados biológico World Protein Database. Para a criação e comparação, foi utilizado também o *World Community Grid*, da IBM, que permitiu a distribuição das tarefas e otimização do processamento (TRISTÃO e LIFSCHITZ, 2009).

Como citam Tristão e Lifschitz (2009), a informação genômica obtida pode ser usada para melhorar a interpretação e qualidade de dados biológicos e a compreensão das interações ambientais, que pode ter um papel crítico no desenvolvimento de medicamentos e vacinas.

As sequências utilizadas para as comparações foram obtidas através das bases *Reference Sequence* (*RefSeq*)[7] do NCBI, com informações sobre as sequências, e do SwissProt[8], trazendo informações sobre as proteínas (TRISTÃO e LIFSCHITZ, 2009).

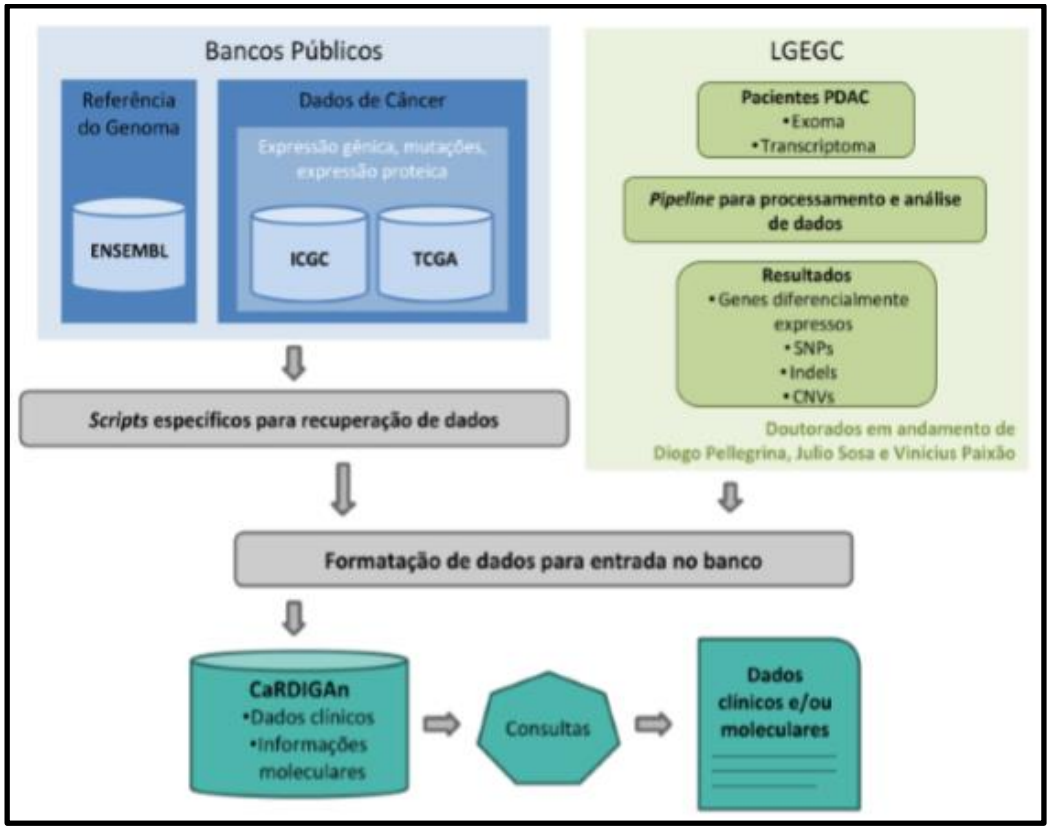
2.2.5 Cardigan

O banco de dados Cardigan, sigla para *Cancer Relation Database for Integration and Genomic Analisys*, integra dados públicos disponíveis que são resultados de amostras de diferentes tipos histopatológicos de câncer no pâncreas. De acordo com Bertoldi (2017), a funcionalidade foi demonstrada através da recuperação de dados clínicos e de expressão gênica, utilizados para uma análise de curva que resultou na identificação de genes com potencial prognóstico para o câncer de pâncreas.

O Cardigan tem como suas fontes de entradas, conforme Figura 5, os bancos públicos Ensembl[9] para obter informações sobre as referências do genoma, juntamente com o ICGC[10] e o TCGA[11], que contém dados específicos sobre mutações e expressões gênicas do câncer.

[7] https://www.ncbi.nlm.nih.gov/refseq/
[8] https://www.uniprot.org/
[9] https://www.ensembl.org/index.html
[10] https://icgc.org/
[11] https://www.cancer.gov/about-nci/organization/ccg/research/structural-genomics/tcga

Figura 5 – Esquema Cardigan



Fonte: Bertoldi (2017).

Além dos bancos de dados públicos, também há dados de três projetos de doutorado do Laboratório de Genômica e Expressão Genica em Câncer do Instituto de Química da Universidade de São Paulo (LGECG). Para a modelagem deste banco de dados, foi utilizado o método de ETL em scripts de SQL para que os dados carregados no Cardigan fossem homogêneos e padronizados (BERTOLDI, 2017).

2.2.6 Chado

O Chado é um modelo de bancos de dados relacional que pode ser usado como base para qualquer grupo de pesquisa genômica por ser flexível e genérico (MIYOSHI, 2013) e faz parte do GMOD (*Generic Model Organism DataBase*)[12]. Uma importante característica desse modelo é o uso de grupos chamados módulos, sendo 18 módulos

[12] http://gmod.org/wiki/Main_Page

no total e cada um responsável por um tipo de informações diferente. Pode-se citar como exemplo, sequências moleculares (*Sequence Module*) e resultados de análises de expressão (*Expression Module*) (JUNG, 2017). Cada módulo é um conjunto de tabelas, funções e gatilhos (*triggers*) que são responsáveis por gerenciar essas informações.

Além disso, outro ponto importante é o fato do Chado ser extensível, ou seja, permite que sejam implementados novos módulos ou ainda, se necessário, a alteração dos módulos existentes (MIYOSHI, 2013).

Um diferencial em relação aos outros bancos de dados biológicos é o Chado fazer uso intensivo de ontologias, que tem um papel central nesse banco de dados já que todas as informações armazenadas devem estar vinculadas a alguma ontologia.

2.3 TRABALHOS RELACIONADOS

Esta seção apresenta as principais características dos bancos de dados biológicos para comparação dos trabalhos relacionados mostrados na tabela 1.

Os itens comparados são:

1. Informações armazenadas: quais tipos de informações são armazenadas nesses bancos de dados, conforme a lista:
   - 1.1. Sequências de DNA
   - 1.2. Genomas
   - 1.3. Proteínas
2. Bancos de dados primários: quais bancos de dados primários foram utilizados para a geração dos dados, conforme a lista:
   - 2.1. Genbank: banco de dados que pertence ao *Nacional Institutes of Health* (NIH) que contém sequências genéticas.
   - 2.2. Kegg: banco de dados que possui informações de funções de alto nível do sistema biológico.
   - 2.3. Unigene[13]: banco de dados pertencente ao NIH com informações sobre marcadores de sequências expressas e RNA mensageiro. Este banco de dados foi retirado do ar em 2019.

[13] https://ncbiinsights.ncbi.nlm.nih.gov/2019/02/01/ncbi-to-retire-the-unigene-database/

2.4. Protein Databank[14]: banco de dados com informações sobre formatos em 3D de proteínas e nucleotídeos, pertencente ao *Research Collaboratory for Structural Bioinformatics* (RCSB).

2.5. Enzyme[15]: repositório de informação sobre nomenclaturas de enzimas.

2.6. RefSeq: banco de dados de nucleotídeos e proteínas, pertencente ao NIH.

2.7. SwissProt: base de dados completa com informações sobre sequências de proteínas e suas funções.

2.8. Gene Oncology[16]: maior banco de dados sobre as funções dos genes. Essas informações podem ser legíveis a humanos e máquinas.

2.9. Taxonomy[17]: é um banco de dados com informações de nomenclaturas e classificação do *International Nucleotide Sequence Database Collaboration* (INSDC), que também compreende o Genbank.

2.10. Ensembl: banco de dados de genomas, criado em conjunto entre *European Bioinformatics Institute* e o *Wellcome Trust Sanger Institute*.

2.11. ICGC: banco de dados que pertence ao ICGC com informações genômicas específicas sobre o câncer.

2.12. TCGA: banco de dados que contém informações genômicas que caracterizou mais de 20 mil moléculas de câncer.

3. Prática de carregamento: qual prática de carregamento foi utilizada.

[14] https://www.rcsb.org/
[15] https://enzyme.expasy.org/
[16] http://geneontology.org/
[17] https://www.ncbi.nlm.nih.gov/taxonomy

Quadro 1 – Comparativo dos trabalhos relacionados

	Item	*IntergenicDB*	*Bio-TIM*	*Protein World Database*	*Cardigan*
1.1.	Sequências de DNA	√			
1.2.	Genomas		√		√
1.3.	Proteínas			√	
2.1.	GenBank	√	√		
2.2.	Kegg	√			
2.3.	Unigeme		√		
2.4.	Protein Databank		√		
2.5.	Enzyme		√		
2.6.	RefSeq			√	
2.7.	SwissProt			√	
2.8.	Gene Oncology			√	
2.9.	Taxonomy			√	
2.10.	Ensembl				√
2.11.	ICGC				√
2.12.	TCGA				√
3.	Prática de carregamento	ETL	ETL	ETL	ETL

Fonte: Própria (2020).

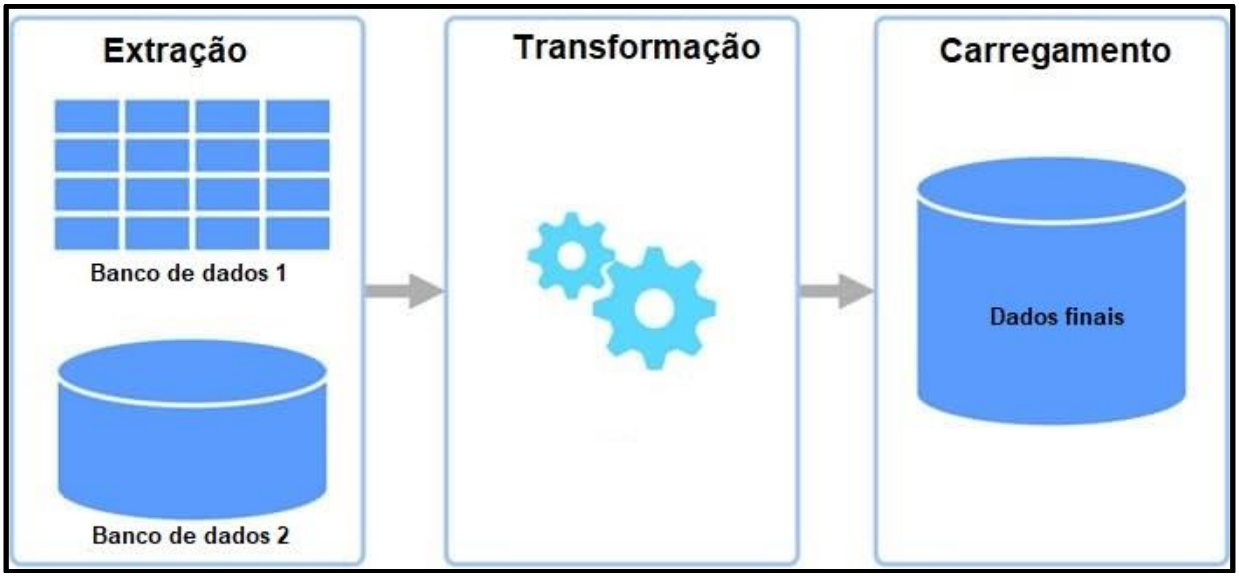
Os trabalhos relacionados foram divididos pelos seus tipos de dados - sequências de DNA, genomas e proteínas - e por seu banco de dados primário utilizado, sendo eles *Genbank, Kegg, Unigene, Protein Databank, Enzyme, RefSeq, SwissProt, Gene Oncology, Taxonomy, Ensembl*, *ICGC* e *TCGA*. Os bancos de dados primários costumam ser específicos em relação ao tipo de informação armazenada e por esse motivo, os bancos de dados secundários comparados, apresentam diferentes bancos de dados primários. Apesar disso, o Genbank é utilizado por dois dos bancos secundários comparados, sendo o IntergenicDB o único banco com informações sobre sequências de DNA.

Em relação às práticas de carregamento, todos os bancos pesquisados utilizam o método de ETL.

2.4 ETL

De acordo com Kimball (2003), o método de ETL é uma integração de dados dividido em três partes, conforme Figura 6: extrair a informação de seu local original, transformar a informação e carregar em outros bancos de dados para que o usuário final tenha acesso. É um método que pode ser usado para combinar dados de diversas fontes, como, por exemplo, no caso do *Protein World Database*, combina os dados dos bancos *RefSeq* e *SwissProt*.

Figura 6 - Processo de ETL



Fonte: Adaptado de site Eng[18] (2020).

Os processos de extração e carregamento, apesar de parecerem relativamente simples, podem ser os mais complicados, com estimativa de que demore cerca de 60% do tempo de processamento do método (KIMBALL, 2003). Apesar disso, o processo de transformação pode ter algumas etapas trabalhosas, como a seleção de quais colunas serão utilizadas, a limpeza de alguns dados – visto como parte importante para Kimball (2003) – como por exemplo a tradução de valores codificados, transposição ou rotação de colunas, codificação de valores, junção de dados provenientes de diversas fontes, quebra de linhas e colunas e demais alterações.

Além dos processos padrões de extração, transformação e carregamento, o método ETL pode ser dividido em sub processos, como cita Becker (2007). Na extração, podem haver 3 processos: criação do perfil dos dados, captura dos dados de alteração e a extração. Durante a criação do perfil dos dados, as informações são exploradas para determinar o que deve ser feito em relação as transformações e limpeza desses dados, enquanto na captura dos dados são isoladas as alterações do banco de dados original para reduzir o processamento. E por fim, a extração que remove os dados. O processo de transformação por sua vez, é divido em 5 sub processos:

- Limpeza dos dados: verifica se há violações de qualidade.

[18] http://www.flashfor.com.br/marketing/eng-blog-business-intelligence-conceito-etl-data-source.jpg

- Rastreamento de erros: captura todos os erros que podem ser vitais para o processamento.
- Criação de dimensão de auditoria: separação de dados em tabelas diferentes, como se fossem dimensões.
- Deduplicação: eliminação de dados redundantes.
- Conformidade dos dados: aplicação dos atributos para junções das dimensões.

Por último, a etapa de carregamento que pode ser dívida em até 13 etapas, entre elas as mais importantes são gerenciar as dimensões e propagação de dados - que prepara e apresenta os dados para entrega (BECKER, 2003).

2.5 DESENVOLVIMENTO GUIADO POR TESTES

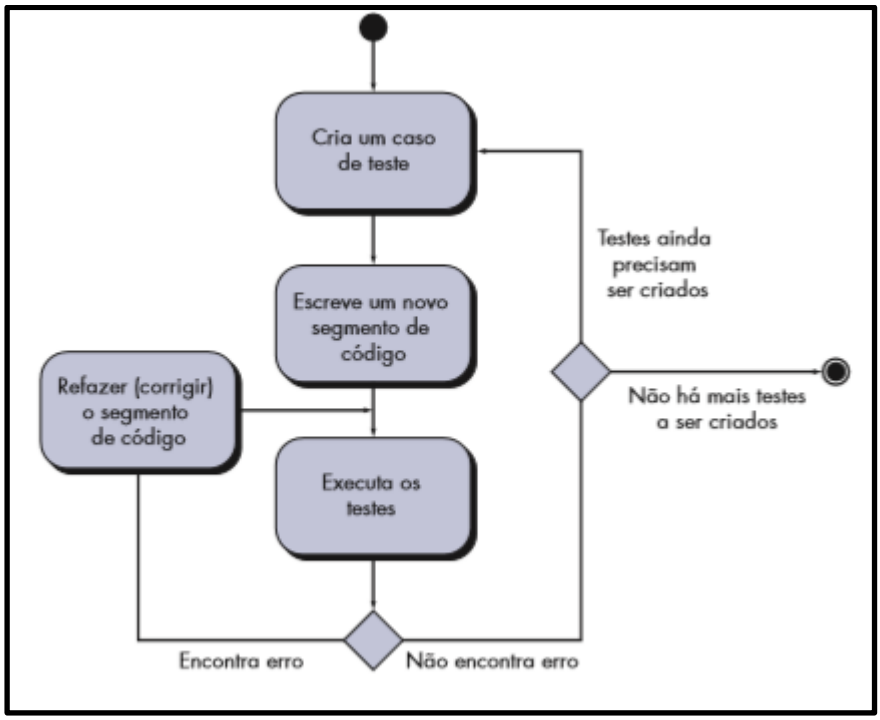
O Desenvolvimento Guiado por Testes (TDD) ou em inglês, *Test Driven Development*, é um método de desenvolvimento de software em que é feito de forma intercalada o desenvolvimento e os testes (BECK, 2010). É um método de programação ágil, pois o desenvolvedor consegue identificar as falhas de forma mais rápida, já que são feitos testes constantemente.

Primeiramente, é desenvolvido uma parte do código de forma incremental e um teste para esse incremento. Em seguida, é iniciado o teste e enquanto a parte do código não for validada pelos testes, não é possível seguir para o próximo incremento (SOMMERVILE, 2011). Para Pressman (2011), todo o código é desenvolvido em pequenos incrementos, uma subfunção de cada vez, e nada mais é escrito enquanto a parte anterior não estiver sem falhas.

Segundo Beck (2010), as etapas do TDD (Figura 7) devem ser seguidas da forma:

1. Adicione um teste.
2. Execute os testes e veja as falhas no mais recente.
3. Altere o código fonte.
4. Execute os testes e veja que não há mais falhas.
5. Refatorar o código fonte.

Figura 7 – Fluxo TDD



Fonte: Sommervile, 2011.

Este fluxo é chamado por Becker (2010) de “*Red, Green, Refactor*”. Onde o vermelho (*red*), significa que há falhas no código fonte e não passou no teste; o verde (*green*), significa que houve sucesso no teste executado; e refatorar (*refactor*), significa que o código fonte está pronto para passar pelo processo de refatoração para que o código seja sempre limpo e sem redundâncias.

3 INTERGENICDB

O IntergenicDB é um banco de dados público desenvolvido para o estudo de sequências intergênicas e foi modelado para armazenar informações relevantes sobre a estrutura das sequências de bactérias gram-negativas (NOTARI et al, 2014). Sua criação ocorreu por necessidade do grupo de bioinformática da UCS para armazenamento e pesquisa, com informações sobre as sequências vindas do banco de dados primário Genbank e os papéis biológicos (*roles*) do Kegg.

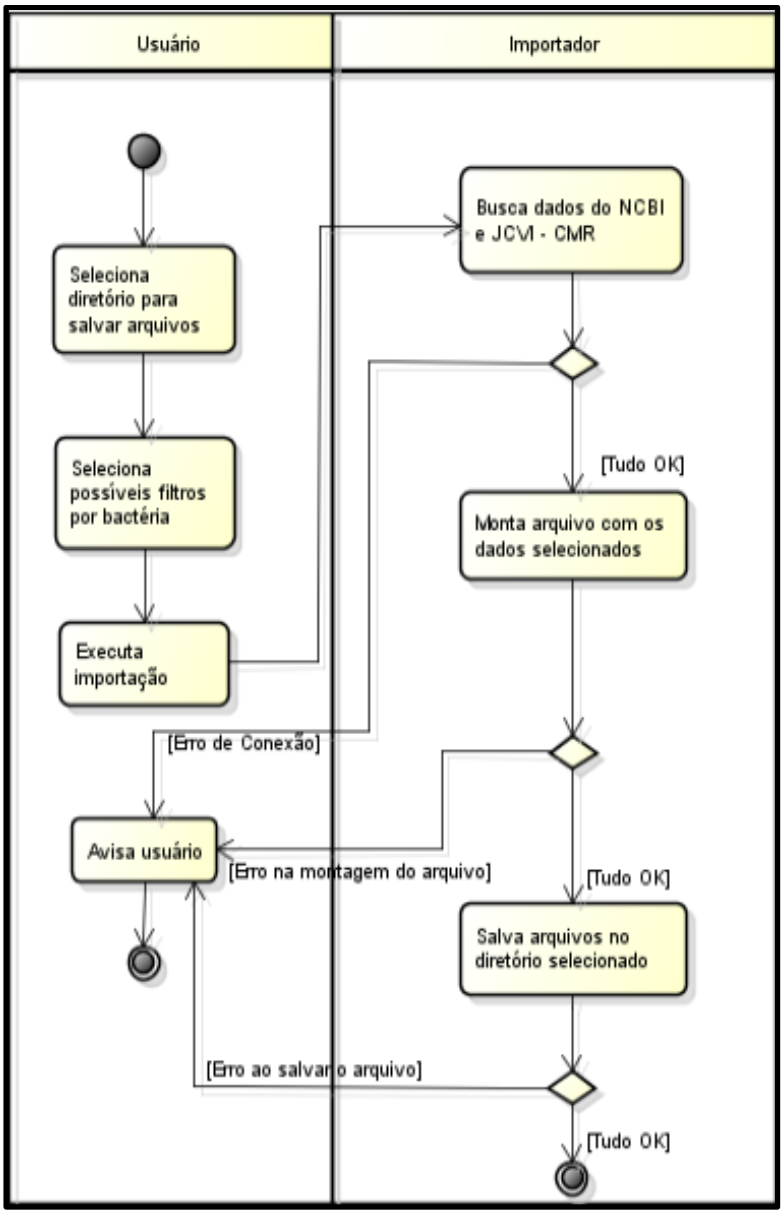
3.1 CARGA DE DADOS

A atual ferramenta de importação, a MMDBImportTool foi desenvolvida na linguagem de programação C# utilizando alguns scripts em *Python*, assim como sua evolução também será. O processo do importador, conforme Figura 8, consiste em conectar ao banco de dados, realizar o *download* dos arquivos contidos no diretório FTP, executar dois scripts em *Python* para que sejam criados os arquivos contendo as informações no sentido *forward* e *reverse*, monta os arquivos com as informações selecionadas e salva os mesmos em um diretório selecionado na tela do importador.

Apesar da MMDBImportTool ter um resultado satisfatório, o tempo de processamento é bastante elevado, pois os arquivos disponibilizados pelo GenBank são grandes e não possuem índice. Com isso, a evolução da ferramenta será desenvolvida com a utilização de *threads* – será detalhado na Seção 4.3.3 – para que o processamento seja dividido e as ações ocorram de forma simultânea, desta forma o tempo de processamento será otimizado.

Além disso, a utilização do método ETL, ajudará na combinação dos dados do GenBank e do Kegg, facilitando a formatação dos dados do arquivo que será gerado e consequentemente sua importação no IntergenicDB.

Figura 8 – Fluxo do processo do importador de dados



Fonte: Dalzochio (2014).

3.2 MODELAGEM DE DADOS

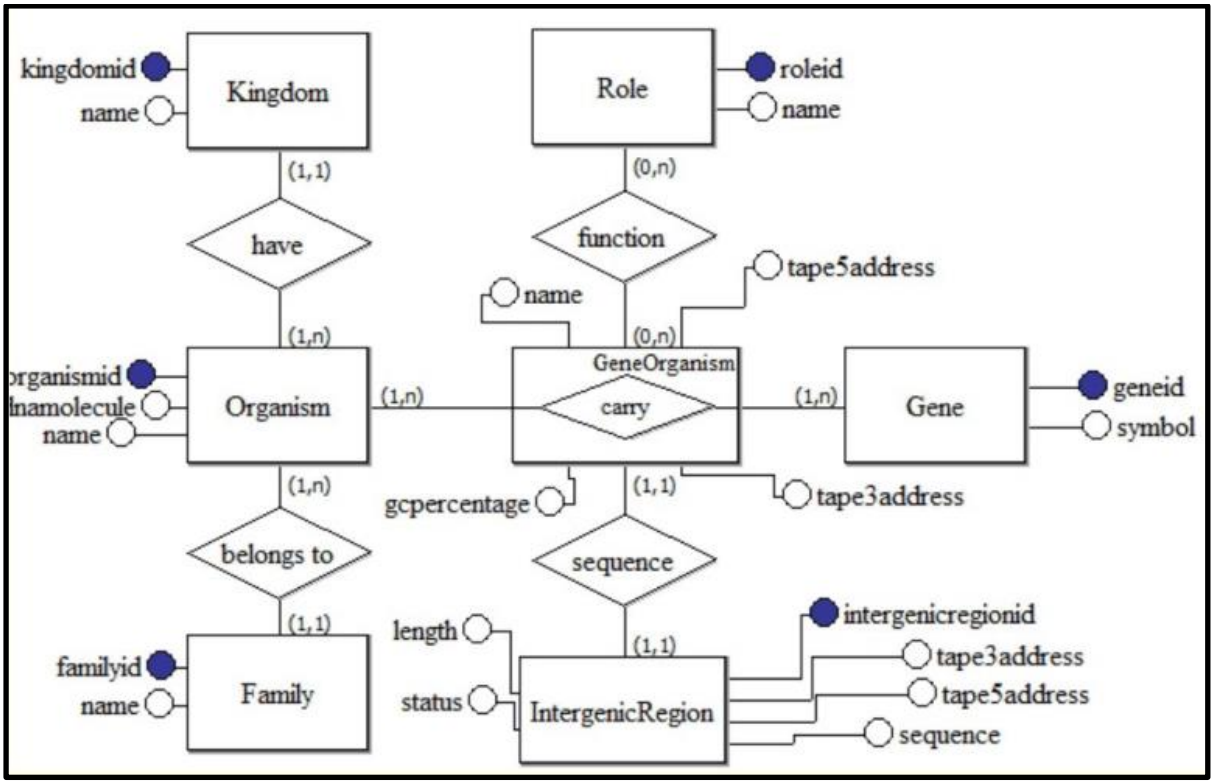
O modelo de ER, conforme Figura 9, apresentado pela última alteração do IntergenicDB não será alterado, já que as últimas mudanças foram eficientes na estruturação do banco de dados. Apesar da estrutura do banco não ser alterada, as informações presentes serão geradas novamente, no método de substituição, sendo importante ressaltar que não será utilizado o método de acréscimo, pois podem haver

informações que foram reclassificadas ou excluídas e desta forma, os dados do IntergenicDB ficariam incorretos.

A estrutura do banco de dados é feita da seguinte forma, a partir da região promotora:

- Organismo (*Organism*): possui nome, reino (*Kingdom*), família (*Family*), tipo de molécula e papel biológico (*Role*).
- Gene (*Gene*): possui nome, símbolo, número de identificação de ínicio e fim, função e quantidade de percentual de CG (guanina-citosina).
- Região Intergênica (*Intergenic Region*): possui número para posição inicial e final na sequência, tamanho, sequência de nucleotídeos e o tipo de fita que pertence.

Figura 9 – Modelo ER



Fonte: Notari et al (2017).

Na Figura 10 pode-se observar, o modelo lógico do banco de dados, onde as colunas sublinhadas indicam a chave primária da tabela e as colunas com *hashtag* indicam a chave estrangeira, para fazer a conexão com as outras tabelas.

Figura 10 – Modelo lógico

Family (familyid, name), *Kingdom* (kingdomid, name), *Gene* (geneid, symbol), *Role* (roleid, name)

Organism (organismid, name, dnamolecule, #familyid, #kingdomid)

GeneOrganism (geneorganismid, name, gcpercentage, tape5address, tape3address, #geneid, #organismid)

GeneOrganismRole (geneorganismroleid, #geneorganismid, #geneid)

IntergenicRegion (intergenicregionid, tape5address, tape3address, sequence, length, status, #geneorgnismid)

Fonte: Notari et al (2017).

3.3 ARQUITETURA DE SOFTWARE

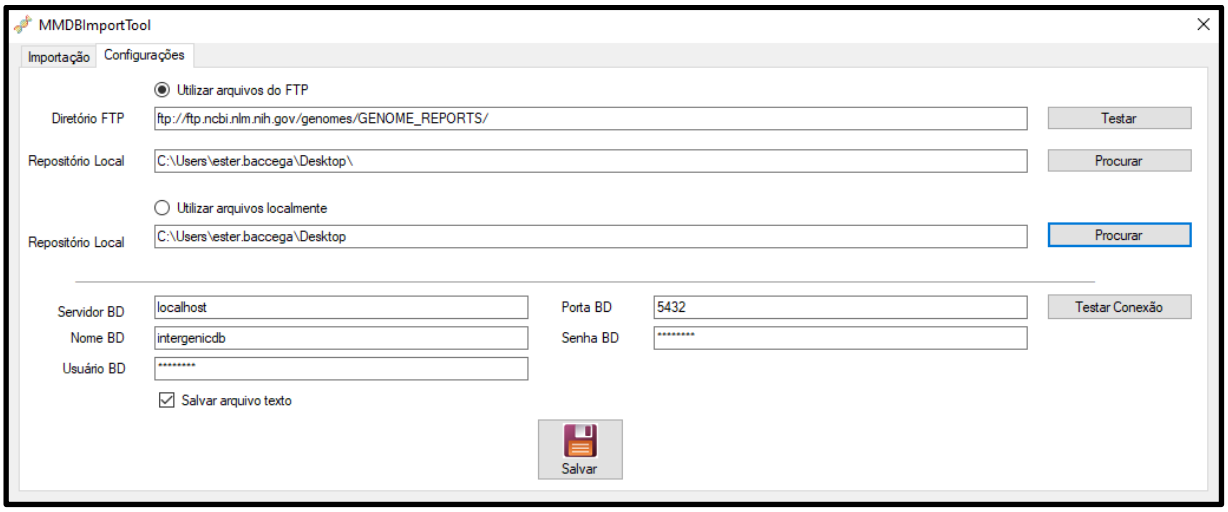
A ferramenta de importação desenvolvida em 2014, chamada de MMDBImportTool, realiza o *download* automático dos arquivos do Genbank e inicia o processo de importação para o IntergenicDB. Na tela principal do importador, conforme Figura 11, é possível escolher qual organismo será importado. Caso nenhum organismo seja selecionado, será realizada a importação de todos.

Figura 11 – Tela principal da MMDBImportTool

Fonte: Própria (2020).

Além da aba principal de importação, a MMDBImportTool conta com uma aba para configurações, incluindo o caminho do diretório FTP, conforme Figura 12.

Figura 12 – Tela de configuração da MMDBImportTool

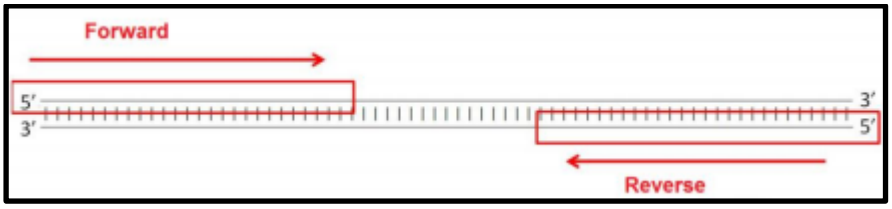


Fonte: Própria (2020).

Após a configuração do diretório FTP e da escolha dos arquivos, é realizado o *download* dos arquivos do Genbank, os quais são salvos em uma pasta local – também escolhida na aba de configurações. Em seguida, ocorre a execução dos scripts em *Python*:

- *ExecutePytonFileMuchosArquivos*: responsável por gerar um arquivo do tipo FASTA que é utilizado pelos scripts rodados a seguir.
- *ExecutePytonFileSelecionaFoward*: responsável por separar os dados da fita dupla de DNA no sentindo *forward*, conforme Figura 13.
- *ExecutePytonFileSelecionaReverse*: responsável por separar os dados da fita dupla de DNA no sentindo *reverse.*
- *ExecutaPythonFileCorrigeSequenciaReversa*: responsável por corrigir as sequências reversas, invertendo e substituindo os nucleotídeos.

Figura 13 – Sentidos *forward* e *reverse*



Fonte: Dalzochio (2014).

4 PROPOSTA DE SOLUÇÃO

Este capítulo apresentará as alterações que serão feitas no software para a realização da nova carga de dados. Será apresentado o problema, bem como os arquivos GBK do Genbank, os novos requisitos e os testes a serem realizados.

4.1 ARQUIVOS GENBANK

Alguns arquivos do Genbank sofreram alterações em relação a sua extensão. Em versões anteriores, dentro do diretório do FTP, existiam os arquivos com extensão GBK (*Genbank Data File*) e os arquivos com extensão GBS. Na versão atual, essas duas extensões foram substituídas pela extensão GBFF.GZ (*Genbank Flat File*), uma versão compactada.

Porém, a estrutura em si do arquivo não sofreu nenhuma alteração em relação as *tags,* marcadas na Figura 14, sendo elas:

- *LOCUS* (1): contém o nome do locus (ou nome arbitrário), comprimento em pares de base (BP), natureza da molécula, topologia da molécula (linear ou circular) e a data de alteração do arquivo.
- *DEFINITION* (2): definição sobre o gene que corresponde a sequência da entrada.
- *ACESSION* (3): número de acesso para essa sequência da entrada e a ligação com outros bancos de dados.
- *VERSION* (4): versão que a sequência se encontra e outros ID sinônimos, como o GI (*GenInfo Identifier*).
- *DBLINK* (5): identificação em qual projeto essa sequência pode estar associada.
- *KEYWORDS* (6): palavras chaves que caracterizam essa sequência.
- *SOURCE* (7): nome comum do organismo que esta sequência representa.
- *ORGANISM* (8): nome científico do organismo e classificação das espécies a qual essa sequência pertence.
- *REFERENCE* (9): referências das sequencias com as sub *tags AUTHORS, TITLE, JOURNAL, PUBMED*. Cada arquivo pode conter mais de uma seção *REFERENCE*, sendo elas numeradas.

- *COMMENT* (10): comentário livre, onde podem haver informações que não se adequam a nenhum dos outros campos acima.
- *FEATURES* (11): segunda seção do arquivo do Genbank, onde se encontram as sub *tags*, *source, gene, CDS*.
- *Source*: origem de regiões específicas da sequência.
- *Gene*: identificação do gene.
- *CDS* (*CoDing Segment*): determina o intervalo de leitura do gene.

Figura 14 – Exemplo de arquivo do Genbank

```
1  LOCUS       CP000828              6503724 bp    DNA     circular BCT 31-JAN-2014
2  DEFINITION  Acaryochloris marina MBIC11017, complete genome.
3  ACCESSION   CP000828
4  VERSION     CP000828.1
5  DBLINK      BioProject: PRJNA12997
               BioSample: SAMN02604308
6  KEYWORDS    .
7  SOURCE      Acaryochloris marina MBIC11017
8    ORGANISM  Acaryochloris marina MBIC11017
               Bacteria; Cyanobacteria; Synechococcales; Acaryochloridaceae;
               Acaryochloris.
9  REFERENCE   1  (bases 1 to 6503724)
     AUTHORS   Swingley,W.D., Chen,M., Cheung,P.C., Conrad,A.L., Dejesa,L.C.,
               Hao,J., Honchak,B.M., Karbach,L.E., Kurdoglu,A., Lahiri,S.,
               Mastrian,S.D., Miyashita,H., Page,L., Ramakrishna,P., Satoh,S.,
               Sattley,W.M., Shimada,Y., Taylor,H.L., Tomo,T., Tsuchiya,T.,
               Wang,Z.T., Raymond,J., Mimuro,M., Blankenship,R.E. and
               Touchman,J.W.
     TITLE     Niche adaptation and genome expansion in the chlorophyll
               d-producing cyanobacterium Acaryochloris marina
     JOURNAL   Proc. Natl. Acad. Sci. U.S.A. 105 (6), 2005-2010 (2008)
      PUBMED   18252824
   REFERENCE   2  (bases 1 to 6503724)
     AUTHORS   Touchman,J.W.
     TITLE     Direct Submission
     JOURNAL   Submitted (27-AUG-2007) Pharmaceutical Genomics Division,
               Translational Genomics Research Institute, 13208 E Shea Blvd,
               Scottsdale, AZ 85004, USA
10 COMMENT     Source bacteria from Marine Biotechnology Institute Culture
               Collection, Marine Biotechnology Institute, 3-75-1 Heita, Kamaishi,
               Iwate 026-0001, Japan.
11 FEATURES             Location/Qualifiers
        source          1..6503724
                        /organism="Acaryochloris marina MBIC11017"
                        /mol_type="genomic DNA"
                        /strain="MBIC11017"
                        /type_material="type strain of Acaryochloris marina"
                        /db_xref="taxon:329726"
```

Fonte: Própria (2020).

- *ORIGIN*: região dos nucleotídeos, onde cada linha contém 60 nucleotídeos. O número inicial de cada linha, indica a posição da primeira base daquela linha, conforme Figura 15.

Figura 15 – Estrutura da tag *Origin*

```
ORIGIN
        1 ttattgtcca caggcatcgc ggaaaaatct gtcccagaag attttgaaga acggtttacg
       61 gccttttacc ttgctgaatc aacctgtctt ggtcaacttg ctaaatctaa aaacggtgag
      121 gcttgctgcg gaactgctga gtgtaccgtt gatgacctgg gttgctgtgc ataggatttt
      181 agaaccagag atgcaagctt tagtcaaagg tagttccgat catctcggtg aagtaagcaa
      241 gtgctcatca acccagaaga caagcaggtg gagatttata gaccagggca agatgttgaa
      301 ttgctgcaat ctcctagtac gatttctgga gcggatgtgc tgccagagtt tagcctcaat
      361 ctggaatgga tttggaggta gcttgtattg aaggaggcag gattttcgga aaattcgcga
      421 tcgcgtgaag gccaacccaa actggcgaaa gatgctgcga tcgcgaagcc gtaccttgtc
      481 gcaatgacag cagaactcca aataatggct acggagactt tgtagcctgt gctcagatca
      541 agtcttccca acggtctacg gcttcttcct ccatggcagc aataaaatca gccacgttcg
      601 atccataaat cccccagctg ttgaccccgc caatttccac aagcttaggt tcgccagcat
      661 gtatcccaat gtcaacagtc agcatgttca taccagttac gtccaaggct tgtgcaacac
      721 tctctgcaac ggatcttatc tcctgaggca cttcagagta tggctcgtct ggatacgagc
      781 tgccgcagaa aaatattccg ttccggcaaa agcaacggta ttcagcctca aatgaaacca
      841 cttcagagac cacagctagc tcattttgat gatgtgagaa agtctttacc caatgtgaaa
      901 gctcttgggt ttttaagacc tccgctggga aaagcttgta attggtatcg gaacggacaa
      961 aaatattatc tccgaacatc ctttgaaggg gtaaattggg tagtgctcca aatggtgcaa
     1021 ataggaaagc acgtcctaaa aaaggataga tgtaagaata gtatttagag caccttagac
     1081 tcgaataatc gtcaaaaata gctgtactga gtggtggatg cctgagcaag cgggtcatag
     1141 caaacatcgt cccgtagccc acaatccaat cttcagtatc tggccatcta cctaacatta
     1201 catcgtcccg atcgatgacg atgggctcta ttttcttatg ttgtcttaaa tatcctgcaa
     1261 ctatctccga tgctgggaac tctgggtgca agtctatcag ccagcgcaaa gtcttcatga
     1321 cgcaaaattc caagagaagc caagttgatt tctatgcaaa gatgactttt gatcatcgag
     1381 cagcaaaaat ttgattaatc cacatgaggc tgtcacatcc atacggcata gcttaactac
     1441 gggattttag agaggcaaag cctaacgcct cgcgtcaaca gtgcggcttt aacacgtcca
     1501 gctgcgtgcg catgttaggt ggcgtcaact ctttgcttaa tggtctggta tgtgtccaca
     1561 ttggtgttaa tcttcgaccc ttgaaaaaca ataacctctc tggcgccagc acccaatgct
     1621 agctctctta acactctctc ctcaatttcc gtaagcccac cttcattctt ctccagttga
     1681 tgaattatta ctctaggaga agttatgagc cgatttctca gaagtttccg gatcaaatat
     1741 tgcagtattt tttcagcata tgtgaagttt gcgacgaaag agcgggaatg atcaaaggga
     1801 tttatgagtt gcagcccagg cttactcatg ttttgacatt cagcgccaac accctcaacc
     1861 gtcaatttat ttttttctcc gcttacggcg attaatggtt tgtcttttac agtggcccca
```

Fonte: Própria (2020).

4.2 PROBLEMA

O IntergenicDB atualmente conta com uma carga de dados realizada em 2017 pela MMDBImportTool (DALZOCHIO, 2014)**,** utilizando os bancos de dados GenBank e Kegg. Recentemente, no ano de 2019, uma nova carga de dados foi disponibilizada pelo GenBank e com isso veio a necessidade de atualizar os dados do IntergenicDB

para que o trabalho de pesquisa de regiões intergênicas do Instituto de Bioinformática da Universidade de Caxias (UCS) consiga continuar em evolução.

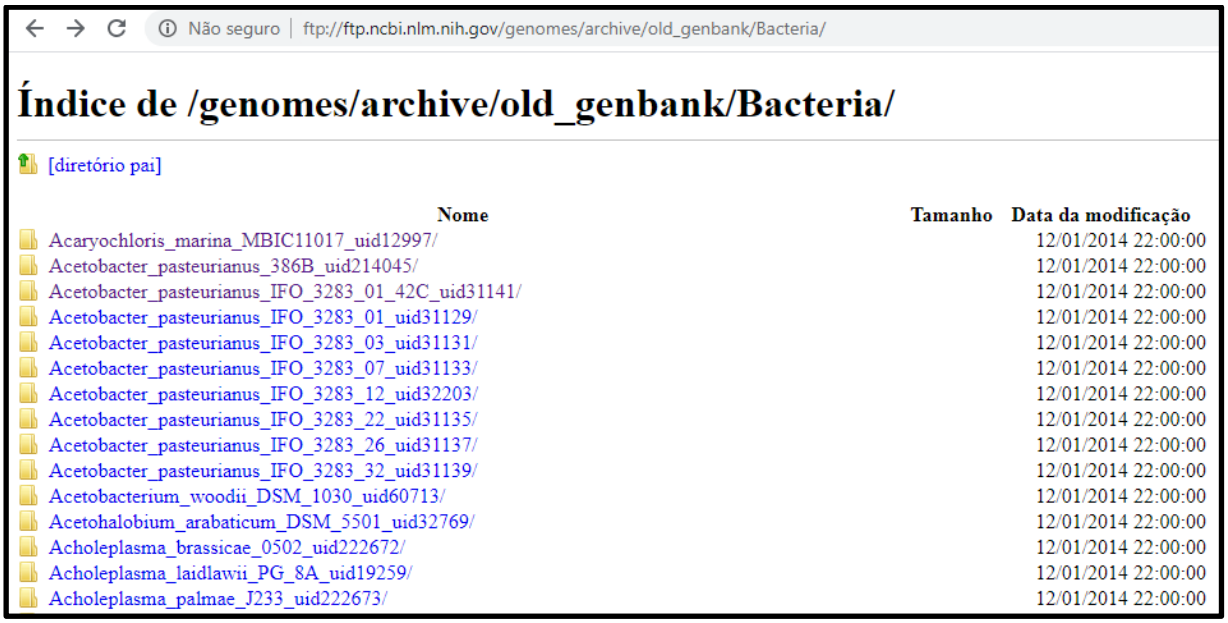
4.3 REQUISITOS FUNCIONAIS

A estrutura de pastas do diretório FTP, bem como a extensão e o próprio arquivo do Genbank sofreram alterações nos últimos anos, com isso serão necessárias algumas alterações funcionais e não funcionais para que o processo de carga de dados seja completo e com melhor processamento.

4.3.1 Diretório FTP

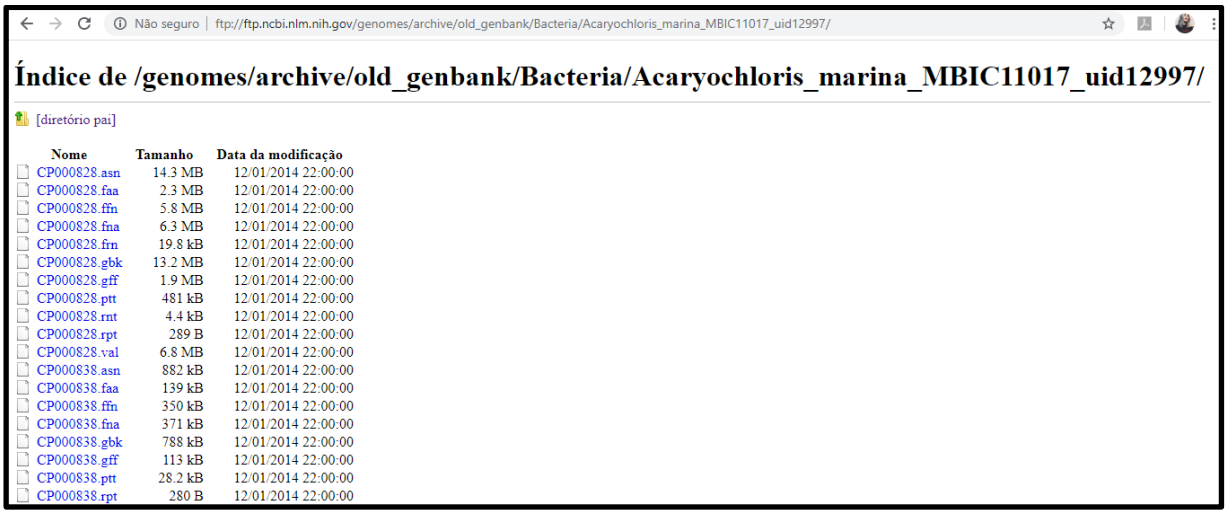
A estrutura de pastas do diretório FTP do Genbank foi totalmente alterada. Inicialmente, ao acessar a raiz FTP para bactérias, existiam as pastas com as nomenclaturas de cada uma (Figura 16) e ao acessar estas pastas, já se encontravam todos os arquivos referentes, inclusive os arquivos GBK, podendo conter um ou mais (Figura 17).

Figura 16 – Raiz do diretório FTP



Fonte: Própria (2020).

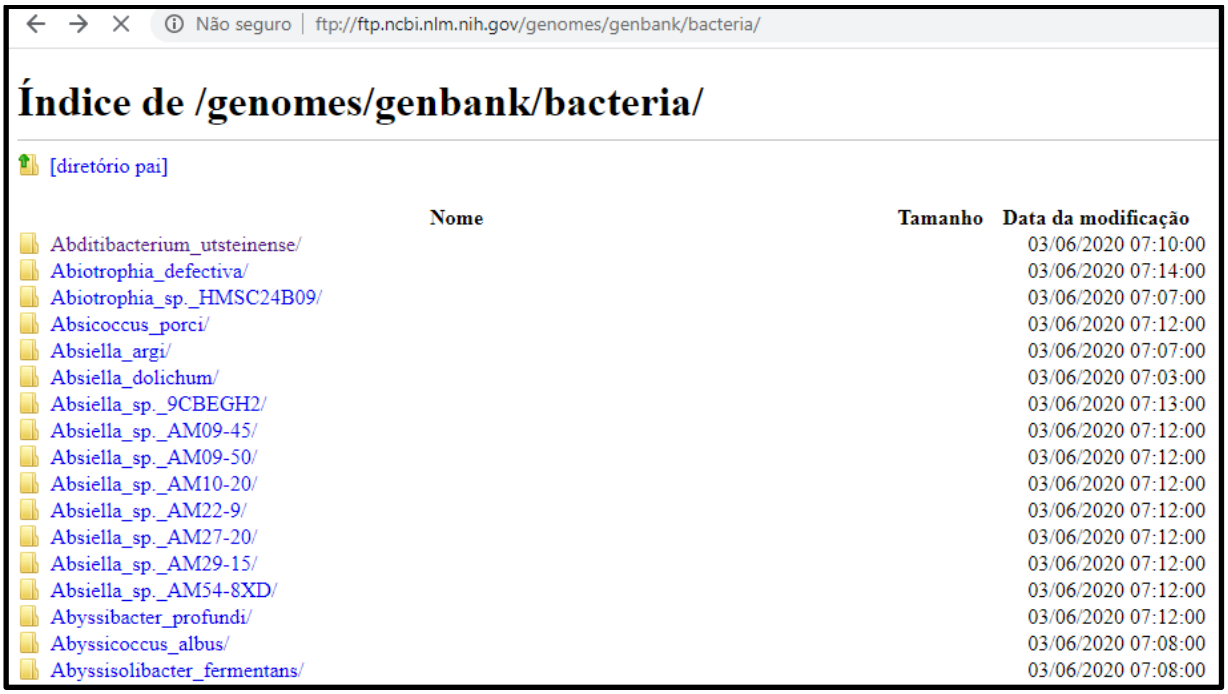
Figura 17 – Estrutura de arquivos dentro da pasta específica



Fonte: Própria (2020).

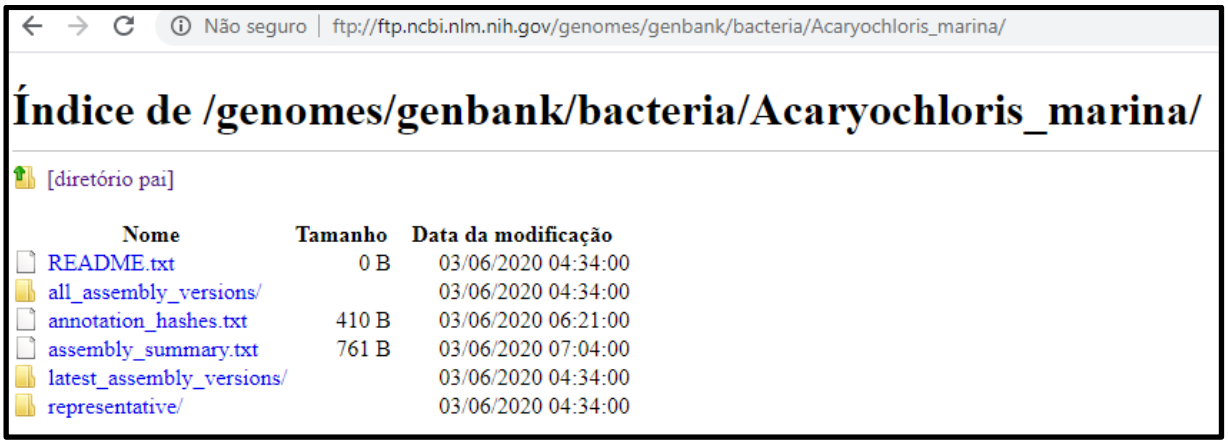
Com as alterações realizadas pelo próprio Genbank, o caminho do diretório FTP ficou mais complexo. Ao acessar a raiz do FTP para bactérias, ainda existem as pastas com as nomenclaturas (Figura 18), porém ao acessar essas pastas, a estrutura já está diferente (Figura 19).

Figura 18 – Raiz do diretório FTP atual



Fonte: Própria (2020).

Figura 19 – Estrutura de arquivos dentro da pasta específica atual



Fonte: Própria (2020).

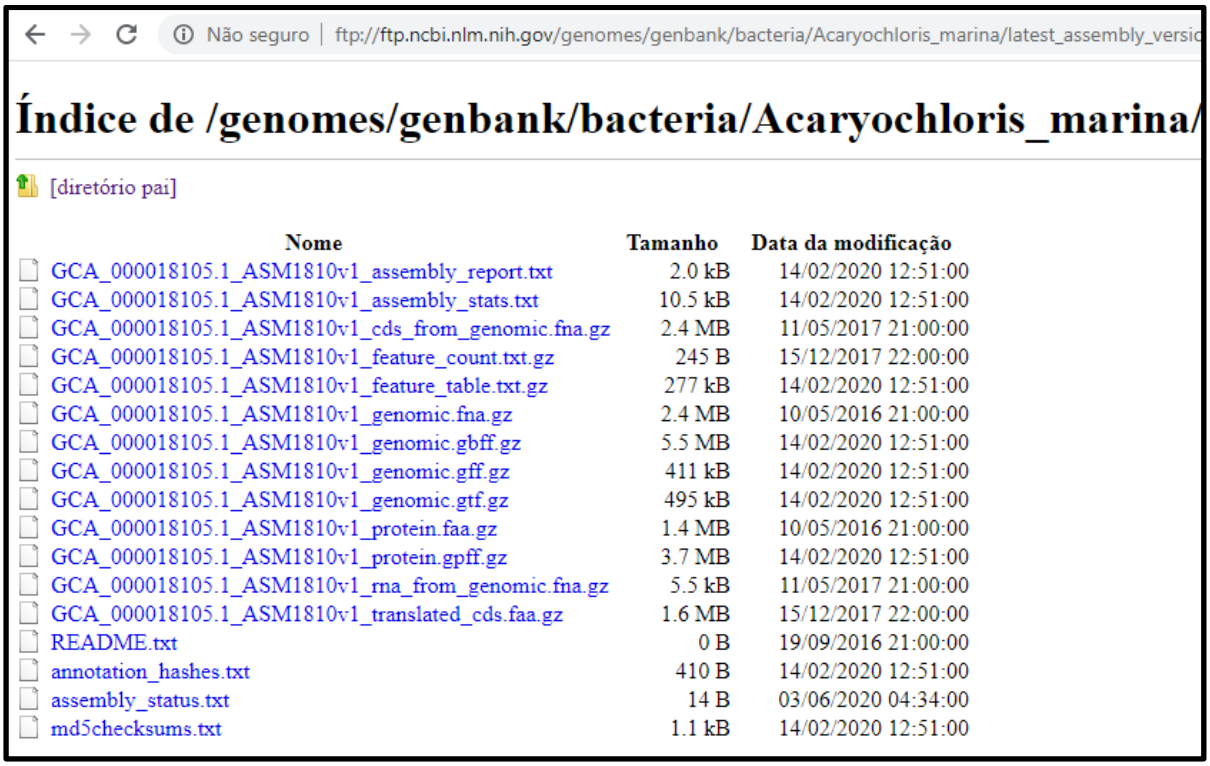
Dentro deste novo nível do FTP, será necessário acessar a pasta "*lastest_assembly_versions*", e então acessar a única pasta existente dentro dela (Figura 20), para que seja acessado o nível final do FTP.

Figura 20 – Estrutura de arquivos

Fonte: Própria (2020).

Após o acesso da pasta mencionada na Figura 20, tem-se a estrutura final de pastas com os arquivos necessários para a importação, conforme Figura 21. Nesse nível, existem vários arquivos, porém o único necessário será o arquivo GBFF.GZ.

Figura 21 – Estrutura contendo os arquivos finais



Fonte: Própria (2020).

Na ferramenta de importação dos dados, atualmente é possível escolher qual o caminho do diretório FTP será utilizado para a realização do *download*. Porém, serão necessárias alterações para comportar a nova estrutura das pastas. Após estar na raiz do FTP, conforme a Figura 18, será necessário alterar via código fonte o diretório para acessar a pasta "*lastest_assembly_versions"*, e então, o único arquivo existente dentro da mesma. Ao chegar na estrutura final, conforme a Figura 21 mostra, será realizado o *download* dos arquivos com a extensão GBFF.GZ.

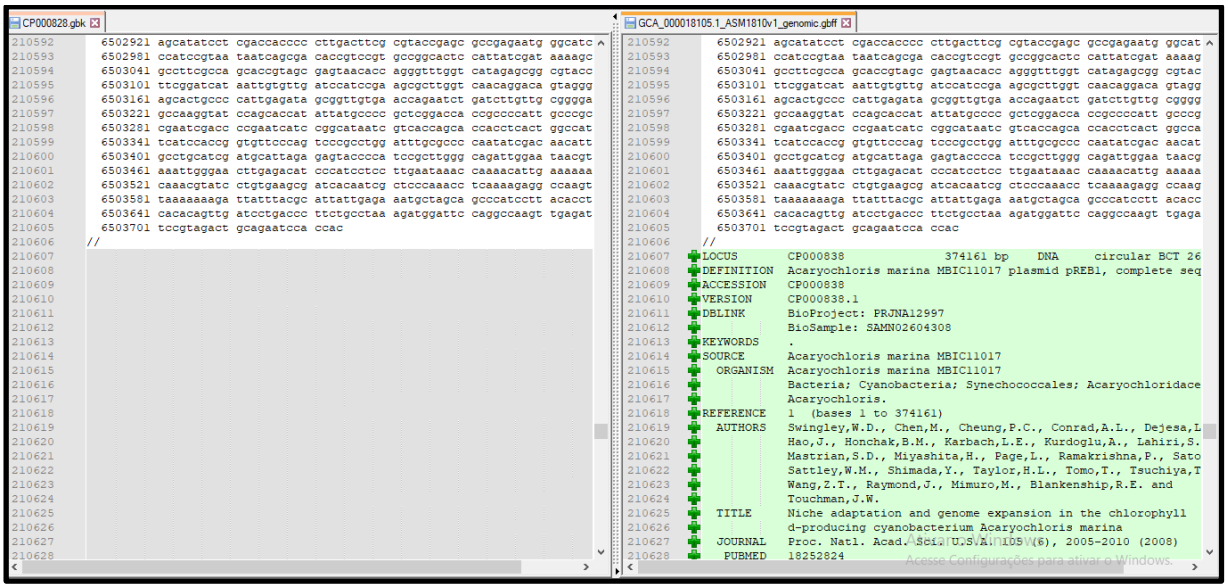
4.3.2 Arquivo Genbank

O arquivo do Genbank sofreu alterações em relação ao seu formato, desta forma será necessário adicionar mais uma etapa na ferramenta de importação da carga. A extensão utilizada pelo Genbank era a GBK, porém atualmente essa extensão foi alterada para GBFF. Além disso, os arquivos passaram a ficar compactados dentro do diretório FTP.

Com isso, além de alteração para busca e *download* desses arquivos, será necessário alterar para que seja descompactado o arquivo GBFF.GZ, realizando a chamada do *7-Zip* antes que seja usado para a importação.

Em relação a estrutura do arquivo GBFF em si, não houveram grandes alterações, porém na antiga versão do FTP poderiam existir um ou mais arquivos GBK para cada bactéria. Atualmente existe apenas um arquivo GBFF. O arquivo GBFF engloba todos os arquivos GBK em um único arquivo. Sendo assim, ao final do arquivo, após o delimitador final "//", inicia-se novamente a estrutura de tags – caso ela exista, já que cada organismo podia conter um ou mais arquivos GBK -, conforme destacado em verde na Figura 22.

Figura 22 – Comparação dos arquivos GBK e GBFF usando o *Notepad* ++



Fonte: Própria (2020).

A ferramenta de importação dos dados realiza a leitura dos arquivos pelas *tags* existentes e não por linhas, desta forma existem duas opções de alterações que podem ser realizadas no importador:

1. Ler a mesma *tag*, exemplo "*LOCUS*", mais de uma vez para cada arquivo GBFF;
2. Quebrar o arquivo GBFF em várias partes, para ficar com a estrutura do antigo arquivo GBK.

4.3.3 Threads

A principal razão para que sejam utilizados *threads*, é quando uma aplicação tem múltiplas atividades que podem ocorrer ao mesmo tempo. Quando um processo grande é decomposto em vários processos pequenos, pode-se ter a aplicação de *threads* sequenciais que são executadas quase em paralelo, o que torna o processamento mais ágil e a programação mais simples (TANENBAUM, 2016).

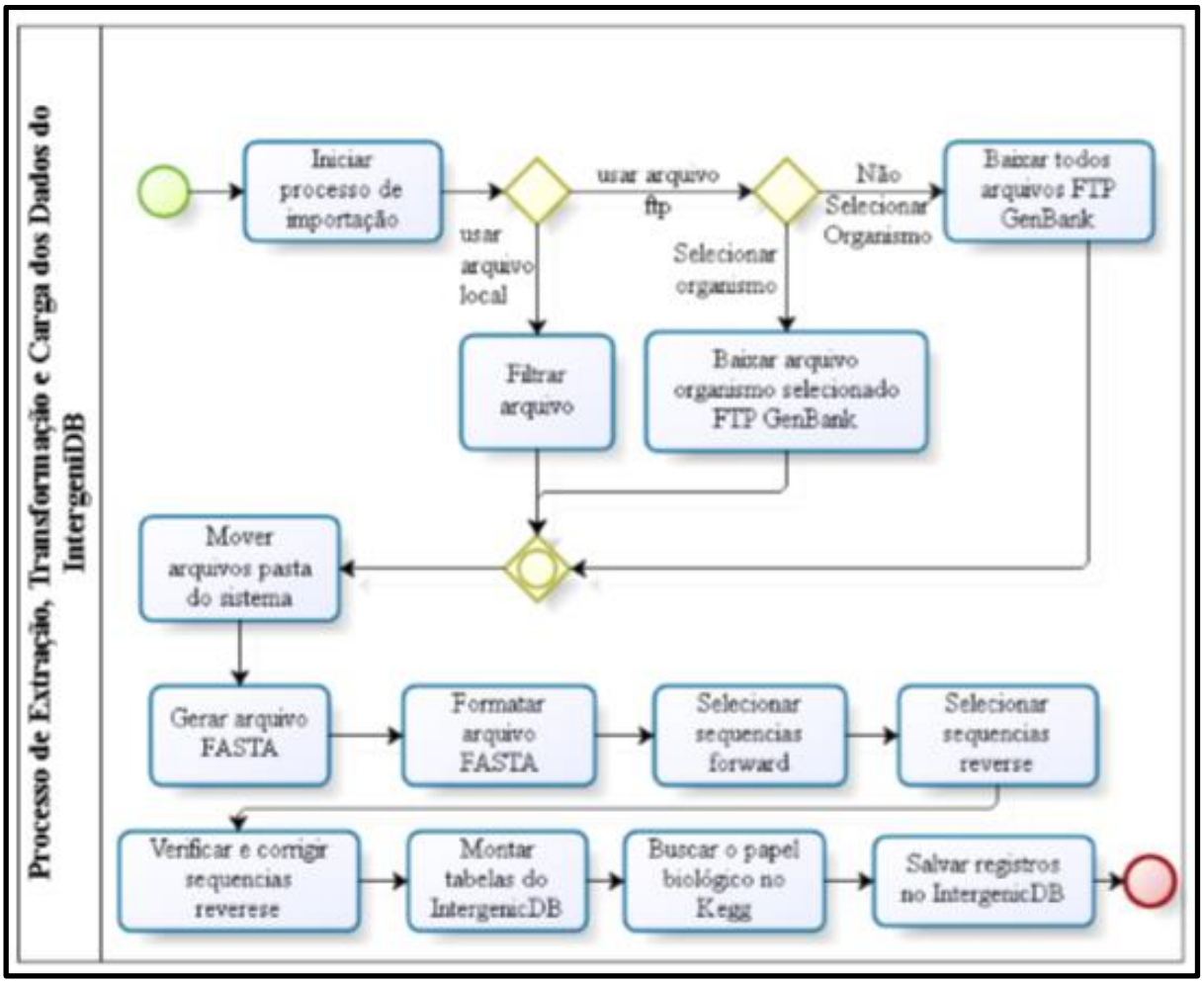
Porém, é importante ressaltar que o uso de *threads* não irá gerar aumento de desempenho quando o processamento é limitado pela CPU. Apenas quando há uma computação e E/S substancial, já que desta forma é permitido que as atividades dos *threads* se sobreponham, gerando uma aceleração no processo como um todo (TANENBAUM, 2016).

A aplicação de threads dentro da MMDBImportTool pode ser implementada durante o *download* dos arquivos. Enquanto um *thread* continua com o processo de *download*, um outro inicia o processo seguinte – a troca dos arquivos de lugar e formatação do arquivo FASTA. Além disso, é possível que seja programado um terceiro *thread*, que irá iniciar a execução dos scripts em *Python*.

4.3.4 Importador MMDBImportTool

O funcionamento básico da MMDBImportTool consiste em *download* dos dados, alterações e carregamento no IntergenicDB, conforme Figura 23. Esses três passos principais serão mantidos. Serão realizadas apenas as alterações necessárias em relação ao processamento dos arquivos, as mudanças de estrutura do diretório FTP e a estrutura dos arquivos disponibilizados pelo Genbank.

Figura 23 – Fluxo de importação



Fonte: Notari et al (2017).

4.3.5 Testes

Para os testes, serão realizadas cargas em bancos de dados de testes, de diferentes formas. Inicialmente, serão realizados testes de importação unitários, onde apenas alguns organismos escolhidos serão importados. Após os primeiros testes, uma carga de dados completa será realizada. Além disso serão realizados testes comparativos, entre a carga do IntergenicDB realizada em 2017 e a nova carga que será realizada na conclusão do desenvolvimento. Com isso, haverá a verificação se os dados foram importados pela MMDBImportTool.

5 DESENVOLVIMENTO DA PROPOSTA DE SOLUÇÃO

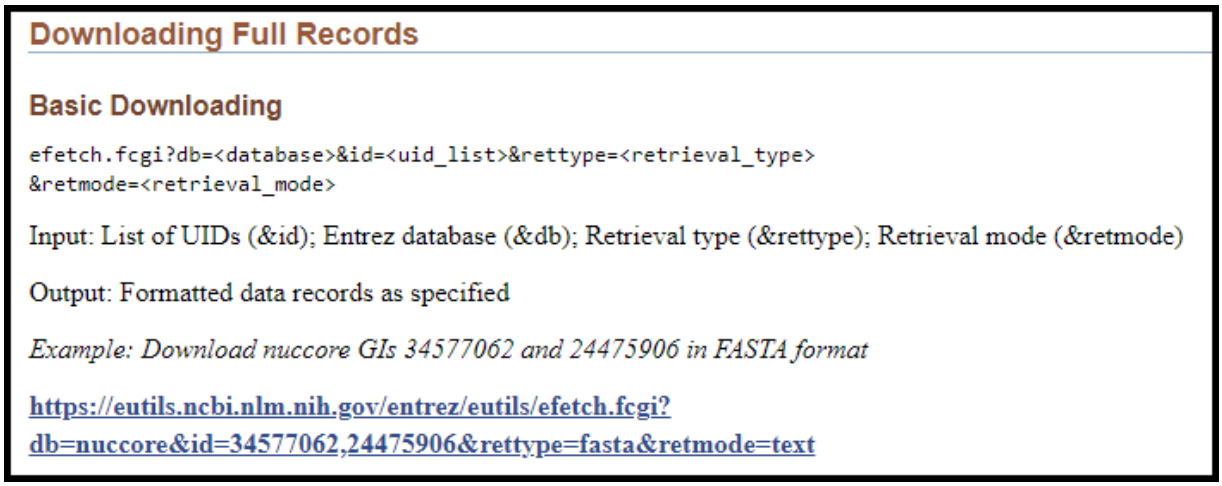
Este capítulo apresentará as alterações desenvolvidas no *software* para a realização da nova carga de dados. Serão apresentadas as alterações em relação ao diretório FTP, o arquivo Genbank, *threads* e a documentação da MMDBImportTool. Além disso, será apresentado o processo de instalação da ferramenta.

5.1 DIRETÓRIO FTP

Além do Genbank disponibilizar um diretório FTP para o *download* de seus arquivos, também existe a opção do uso da API *E-utilities*[19]. A API conta com várias opções para *download* de arquivos, entre elas a opção "*full records*", conforme Figura 24, que como o próprio nome sugere, faz o *download* do arquivo completo. Com a utilização da API, é possível definir os padrões desejados para que seja realizado o *download* do arquivo Genbank, como o formato do arquivo e a sua extensão.

Para a utilização da API, é necessário montar uma *URL* com o caminho correto, onde será informado o tipo de *download*, organismo, formato do arquivo e extensão.

Figura 24 – Especificação da opção "*full records*"



Fonte: Própria (2020).

A versão antiga do importador, utilizava o *download* pelo diretório FTP, conforme Figura 25, sendo necessários vários testes para que estivesse no caminho

[19] https://www.ncbi.nlm.nih.gov/books/NBK25501/

correto e fosse apenas o arquivo GBK (A) e a versão atual, utiliza o *download* pela API, conforme Figura 26. Para que seja realizado o *download* do arquivo na versão atual, é realizada a chamada do procedimento *ReturnaURLDownloadArquivo*, onde é informado o código do arquivo que foi escolhido em tela (B).

Figura 25 – *Download* versão anterior

```
public void GetGBKFromFTPNCBI(ConfiguratioDTO configuratioDTO, string folder)
{
    string file ="";
    List<string> files = ListFileFromFTP(configuratioDTO.FTPFolder + folder);
    foreach (string item in files)
    {
A       if (item.ToLower().Contains(".gbk"))
        {
            file = item;

            FtpWebRequest request = (FtpWebRequest)WebRequest.Create("ftp://ftp.ncbi.nlm.nih.gov/genomes/Bacteria/" + file);
            request.Method = WebRequestMethods.Ftp.DownloadFile;
            request.Credentials = new NetworkCredential("anonymous", "bioinfoucs@gmail.com");
            request.UsePassive = true;
            request.UseBinary = true;
            request.KeepAlive = true;

            FtpWebResponse response = (FtpWebResponse)request.GetResponse();
            Stream responseStream = response.GetResponseStream();

            byte[] buffer = new byte[2048];

            string _diretorio = configuratioDTO.LocalFolder + "\\Arquivos\\" + file.Replace(file.Substring(item.IndexOf("/")), "");
            if (!Directory.Exists(_diretorio))
            {
                Directory.CreateDirectory(_diretorio);
            }

            FileStream newFile = new FileStream(_diretorio + "\\" + file.Substring(item.IndexOf("/") +1), FileMode.Create);
```

Fonte: Própria (2020).

Figura 26 – *Download* versão atual

```
namespace MMDBImportToolUtil
{
    public static class Recursos
    {
        public static string RetornaNomeArquivoOrganismo()
        {
            return "prok_reference_genomes.txt";
        }
        public static string RetornaURLDeDownloadArquivo(string id)
        {
B           return string.Concat("http://eutils.ncbi.nlm.nih.gov/entrez/eutils/efetch.fcgi?db=nuccore&id=", id, "&rettype=gbwithparts&retmode=text");
        }
    }
}
```

Fonte: Própria (2020).

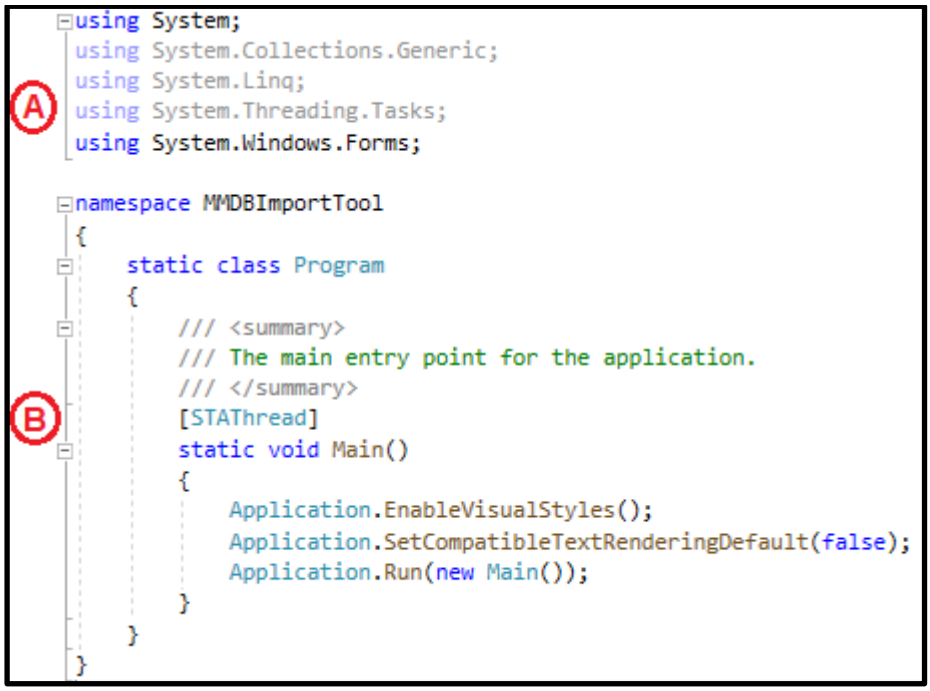
5.2 ARQUIVO GENBANK

Nenhuma alteração em relação ao arquivo Genbank foi necessária, pois com a utilização da API, os arquivos não sofreram as alterações citadas na Seção 4.3.2.

5.3 THREADS

A MMDBImportTool utiliza atualmente juntamente com as *threads*, o método *tasks* da própria linguagem *C#*. Na inicialização do *software*, além de incluir o pacote "*System.Threading.Tasks*" (A), também é chamado o procedimento padrão *STAThread* (B), conforme Figura 27. A chamada desse procedimento faz com que toda a execução do *software* já seja otimizada, pois automaticamente ocorrerão a utilização das *threads*.

Figura 27 – Inicialização das *Threads*

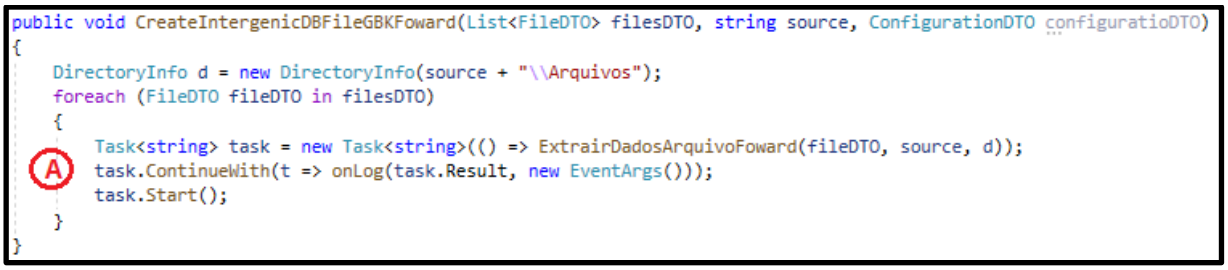


Fonte: Própria (2020).

A *namespace* chamada *System.Threading.Tasks* contém classes que permitem obter a funcionalidade de *threads*, onde internamente são desenvolvidas com o uso das *ThreadsPools*. Quando uma *task* – tarefa – é chamada, o fluxo do código fica mais flexível, já que essa tarefa é iniciada, mas o processamento não é exclusivo até que seja finalizada, sendo também possível definir se essa tarefa deve continuar ou aguardar até que outro processo seja finalizado dentro do código fonte do software. Dentro da MMDBImportTool, as *tasks* são utilizadas, por exemplo, na chamada dos procedimentos que irão extrair os dados do organismo do arquivo FASTA, tanto para sentindo *forward* (A) - conforme Figura 28 - quanto para sentido

reverse. Dessa forma, todo o procedimento de leitura dos arquivos, será uma tarefa separada, permitindo que haja chamada dos procedimentos seguintes enquanto a leitura ocorre.

Figura 28 – Criação da *Task*

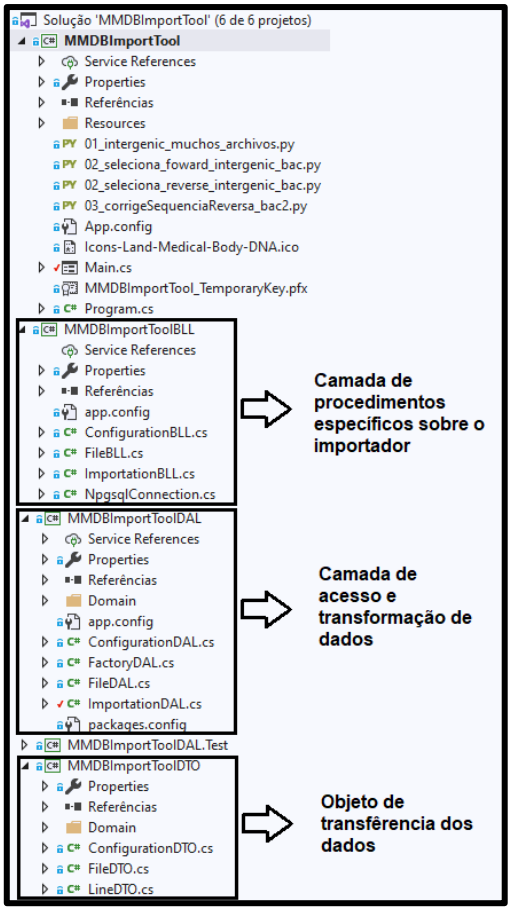


Fonte: Própria (2020).

5.4 IMPORTADOR MMDBIMPORTTOOL

O desenvolvimento da ferramenta MMDBImportTool é dividido em duas camadas e um objeto: *ImportationBLL, ImportationDAL* e *ImportationDTO*, conforme Figura 29. A primeira camada é a chamada de *Bussiness Logic Layover* (BLL) a qual contém os procedimentos específicos sobre o importador, como criação dos arquivos do IntergenicDB e a execução dos *scripts* em *Python*. A segunda camada é chamada de *Data Acess Layover* (DAL) e é responsável por acessar os dados e fazer todas as transformações necessárias nessas informações. Já o objeto, chamado *Data Transfer Object* (DTO), é o responsável pelas *structs* onde as informações são armazenadas até sua gravação no banco de dados.

Figura 29 – Camadas do importador



Fonte: Própria (2020).

Na ferramenta de importação, é possível selecionar qual organismo será realizada a importação. Essa lista de organismos disponíveis, é disponibilizada através do *download* do arquivo *Prok Reference Genomes*, diretamente do diretório FTP do Genbank, conforme o procedimento *BuscaArquivoProk_Reference_Genome* – Figura 30 -, onde é feita uma requisição ao FTP (A), inserido os dados de acesso (B) e criado o arquivo (C).

Figura 30 – Busca do arquivo *Prok Reference Genomes*

```
public void BuscaArquivoProk_Reference_Genomes(ConfigurationDTO configurationDTO)
{
    //Cria comunicação com o servidor
    //definindo o arquivo para download
A   FtpWebRequest request = (FtpWebRequest)WebRequest.Create(configurationDTO.FTPFolder + Recursos.RetornaNomeArquivoOrganismo());
    //Define que a ação vai ser de download
    request.Method = WebRequestMethods.Ftp.DownloadFile;
    //Credenciais para o login (usuario, senha)
B   request.Credentials = new NetworkCredential("anonymous", "bioinfoucs@gmail.com");
    //modo passivo
    request.UsePassive = true;
    //dados binarios
    request.UseBinary = true;
    //setar o KeepAlive para true
    request.KeepAlive = true;

    //criando o objeto FtpWebResponse
    FtpWebResponse response = (FtpWebResponse)request.GetResponse();
    //Criando a Stream para ler o arquivo
    Stream responseStream = response.GetResponseStream();

    byte[] buffer = new byte[2048];

    //Definir o local onde o arquivo será criado.
    string _diretorio = configurationDTO.LocalFolder;
    if (!Directory.Exists(_diretorio))
        Directory.CreateDirectory(_diretorio);

    FileStream newFile = new FileStream(_diretorio + Recursos.RetornaNomeArquivoOrganismo(), FileMode.Create);
C   //Ler o arquivo de origem
    int readCount = responseStream.Read(buffer, 0, buffer.Length);
    while (readCount > 0)
    {
        //Escrever o arquivo
        newFile.Write(buffer, 0, readCount);
        readCount = responseStream.Read(buffer, 0, buffer.Length);
    }
    newFile.Close();
    responseStream.Close();
    response.Close();
```

Fonte: Própria (2020).

Uma vez que o organismo foi escolhido, ao clicar no botão "*Importar*", é inicializado o processo de importação, conforme Figura 31. O processo ocorre dentro do procedimento *btnImport_Click* (A). Além disso, como existe a opção de importar apenas um ou mais organismos, existem duas opções para a chamada do procedimento *ImportationBLL.DisponibilizaArquivos* que irá fazer o *download* do arquivo. Na primeira opção (B), dentro de um laço com a lista de todos os organismos disponíveis no arquivo *Prok Reference Genomes*, para que seja feita a importação completa e na segunda opção (C), apenas para o organismo selecionado em tela.

Figura 31 – Procedimento de início da importação

Fonte: Própria (2020).

A partir disso é chamado o procedimento *DisponibilizaArquivos*, conforme Figura 32, em que o código do organismo do arquivo *Prok Reference Genomes* é ajustado para a chamada do procedimento *ImportationDAL.FazDownloadArquivo*, conforme Figura 33. Nesse momento, é solicitado o *download* do arquivo pela API, descrito na Seção 5.1.

Figura 32 – Disponibilização dos arquivos

```
public void DisponibilizaArquivos(string bacteria)
{


    ImportationDAL importationDAL = new ImportationDAL(configurationDTO);
    if (configurationDTO.UseFTP)
    {
        var item = bacteria.Split('|');
        Log("Realizando Download do organismo => " + item[0].Trim(), new EventArgs());
        importationDAL.FazDownloadArquivo(configurationDTO, item[0].Trim(), item[1].Trim());
        MoverArquivosParaPastaAplicacao(configurationDTO.LocalFolder + "Arquivos\\" + item[0].Trim(), item[0].Trim());

    }
    else
    {
        MoveLocalFiles(bacteria);

    }
}
```

Fonte: Própria (2020).

Figura 33 – Procedimento de download

```
public void FazDownloadArquivo(ConfigurationDTO configuratioDTO, string organismo, string id)
{

    //Definir o local onde o arquivo será criado.
    string _diretorio = configuratioDTO.LocalFolder + "\\Arquivos\\" + organismo;
    if (!Directory.Exists(_diretorio))
    {
        Directory.CreateDirectory(_diretorio);
    }
    //definindo o arquivo para download
    using (WebClient request = new WebClient())
    {
        System.Net.ServicePointManager.SecurityProtocol = System.Net.SecurityProtocolType.Tls12;
        request.DownloadFile(Recursos.RetornaURLDeDownloadArquivo(id), configuratioDTO.LocalFolder + "Arquivos\\" + organismo + "\\" + id + ".txt");
    }


}
```

Fonte: Própria (2020).

Após a finalização do *download*, é iniciado o processo de tratamento das informações dos arquivos e para isso, são rodados alguns *scripts* em *Python,* conforme Figura 34*.* Esses *scripts* são responsáveis gerar um arquivo novo, que será do tipo FASTA (A), além de formatar, fazendo a troca de alguns símbolos (B) – conforme Figura 35 -, selecionar as sequências *reverse* e *forward*, além de corrigir a sequência *reverse*.

Figura 34 – Procedimento de importação

```
public void ExecutarImportacao()
{
    onLog("Executando Script PytonFileMuchosArquivos", new EventArgs());
    ExecutePytonFileMuchosArquivos();
    FormatFasta();
    onLog("Executando Script PytonFileSelecionaFoward", new EventArgs());
    ExecutePytonFileSelecionaFoward();
    onLog("Executando Script PytonFileSelecionaReverse", new EventArgs());
    ExecutePytonFileSelecionaReverse();
    onLog("Executando Script PytonFileCorrigeSequenciaReversa", new EventArgs());
    ExecutePytonFileCorrigeSequenciaReversa();
    CreateIntergenicDBFiles();

}
```

Fonte: Própria (2020).

Figura 35 – Procedimento de formação arquivo FASTA

```
private void FormatFasta()
{

    DirectoryInfo d = new DirectoryInfo(Directory.GetCurrentDirectory());
    FileInfo[] Files = d.GetFiles("*.fasta"); //Getting Text files
    foreach (FileInfo file in Files)
    {

        bool bFisrt = true;

        using (FileStream fs = File.Open(Directory.GetCurrentDirectory() + "\\" + file.Name, System.IO.FileMode.Open))
        using (BufferedStream bs = new BufferedStream(fs))
        using (StreamReader sr = new StreamReader(bs))
        {
            string _diretorio = @Directory.GetCurrentDirectory() +  "\\ArquivosFasta";
            if (!Directory.Exists(_diretorio))
            {
                Directory.CreateDirectory(_diretorio);
            }

A       using (var outputFile = File.CreateText(_diretorio + "\\" + file.Name))
            {
                string s;
                while ((s = sr.ReadLine()) != null)
                {
                    if (s.Contains(">"))
                    {
                        if (!bFisrt)
                        {
                            outputFile.WriteLine("");
                        }
                        bFisrt = false;
B                       s = s.Replace(" +", " +" + "\t").Replace(" -", " -" + "\t").Replace(" ", "\t");
                        outputFile.Write(s);
                    }
                    else
                    {
                        outputFile.Write(s);
                    }
```

Fonte: Própria (2020).

A versão do *Python* pode influenciar os *paths* (caminhos) dos repositórios do computador e por conta disso, foi necessária fazer uma pequena alteração no *script 01_INTERGENIC_MUCHOS_ARQUIVOS.py*, conforme Figura 36, onde a versão atual está na marcação A, enquanto a versão anterior na marcação B.

Figura 36 – Alteração do script em *Python*

```
for line3 in todas_carpetas: # este eh o for maior - tudo deve estar dentro dele
    z = str(line3)
#   print line3
A   todos_arquivos=os.listdir(camino_carpeta+"\\"+complemento_carpeta+"\\"+line3)
    for arquivo in todos_arquivos:
        if __name__ == '__main__':
            if len(sys.argv) == 2:
                get_interregions(camino_carpeta+"\\"+complemento_carpeta+"\\"+line3+"\\"+arquivo)
            elif len(sys.argv) == 3:
                get_interregions(line3,int(sys.argv[2]))
        else:
            #print "Usage: get_intergenic.py gb_file [intergenic_length]"
            sys.exit(0)
```

```
for line3 in todas_carpetas: # este eh o for maior - tudo deve estar dentro dele
    z = str(line3)
#   print line3
B   todos_arquivos=os.listdir(camino_carpeta+"/"+complemento_carpeta+"/"+line3)
    for arquivo in todos_arquivos:
        if __name__ == '__main__':
            if len(sys.argv) == 2:
                get_interregions(camino_carpeta+"/"+complemento_carpeta+"/"+line3+"/"+arquivo)
            elif len(sys.argv) == 3:
                get_interregions(line3,int(sys.argv[2]))
        else:
            #print "Usage: get_intergenic.py gb_file [intergenic_length]"
            sys.exit(0)
```

Fonte: Própria (2020).

Quando todos os *scripts* terminam sua execução, o repositório terá três arquivos FASTA distintos para cada organismo, sendo eles um arquivo *reverse*, um arquivo *reverse* corrigido e um arquivo *forward*. Com isso, é iniciado o procedimento chamado *CreateIntergenicDBFile* – Figura 37 -, para a criação do arquivo que será importado para o banco de dados. Durante a primeira parte deste procedimento, são buscados os arquivos no diretório “ArquivosFasta” (A), para que sejam criados os arquivos do IntergenicDB sentido *forward* (B). Na segunda parte, são buscados os arquivos no diretório “ArquivosFastaReverse” (C), onde ficam os arquivos já corrigidos, para que sejam criados os arquivos do IntergenicDB no sentido *reverse* (D).

Figura 37 – Procedimento de criação *IntergenicDBFile*


```
private void CreateIntergenicDBFile()
{

    ImportationDAL importationDAL = new ImportationDAL(configurationDTO);
    importationDAL.Log += new ImportationDAL.LogEventHandler(onLog);
    List<FileDTO> filesDTO = new List<FileDTO>();

    DirectoryInfo d = new DirectoryInfo(diretorioLocal + "\\ArquivosFasta");
    FileInfo[] Files = d.GetFiles("*.txt");
    foreach (FileInfo file in Files)
    {

        List<LineDTO> linesDTO = importationDAL.CreateIntergenicDBFileFoward(file);
        FileDTO fileDTO = new FileDTO();
        fileDTO.Lines = linesDTO;
        fileDTO.Name = file.Name;
        filesDTO.Add(fileDTO);
    }

    importationDAL.CreateIntergenicDBFileGBKFoward(filesDTO, diretorioLocal, configurationDTO);

    d = new DirectoryInfo(diretorioLocal + "\\ArquivosFasta\\Reverse");
    Files = d.GetFiles("*.txt");
    filesDTO.Clear();
    foreach (FileInfo file in Files)
    {
        if (file.Name.IndexOf("CONVERTIDO_") >= 0)
        {
            List<LineDTO> linesDTO = importationDAL.CreateIntergenicDBFileReverse(file);
            FileDTO fileDTO = new FileDTO();
            fileDTO.Lines = linesDTO;
            fileDTO.Name = file.Name;
            filesDTO.Add(fileDTO);
        }
    }
    importationDAL.CreateIntergenicDBFileGBKReverse(filesDTO, diretorioLocal, configurationDTO);
}
```

Fonte: Própria (2020).

Durante esse processo, são chamados os procedimentos *importationDAL.CreateIntergenicDBFileGBKForward* (conforme Figura 38) e *importationDAL.CreateIntergenicDBFileGBKReverse*. Os dois procedimentos funcionam da mesma forma, com diferença apenas no tipo de dado que cada arquivo contém.

Figura 38 – Procedimento de criação do arquivo sentido *forward*

```
public void CreateIntergenicDBFileGBKFoward(List<FileDTO> filesDTO, string source, ConfigurationDTO configuratioDTO)
{
    DirectoryInfo d = new DirectoryInfo(source + "\\Arquivos");
    foreach (FileDTO fileDTO in filesDTO)
    {
        Task<string> task = new Task<string>(() => ExtrairDadosArquivoFoward(fileDTO, source, d));
        task.ContinueWith(t => onLog(task.Result, new EventArgs()));
        task.Start();
    }
}
```

Fonte: Própria (2020).

Dentro de cada um desses processos, é realizada a chamada do procedimento *ExtrairDadosArquivosForward* (conforme Figura 39) ou *ExtrairDadosArquivosReverse*, os quais são responsáveis pela leitura dos arquivos FASTA (A) e separação dos dados contidos nele. Durante esse processo, é realizada a separação de todas as informações contidas nos arquivos para cada organismo (B), como o nome do organismo, reino (*kingdom*) (C), família (*family*) (D), gene e as sequências de DNA. Essas informações ficam armazenadas variáveis do tipo *struct* para posteriormente serem gravadas no banco de dados.

Figura 39 – Procedimento de separação dos dados

```
private string ExtrairDadosArquivoFoward(FileDTO fileDTO, string source, DirectoryInfo d)
{
    StringBuilder log = new StringBuilder();
A   string organismName, kingdom, dnaMolecule, family;
    string fileName = fileDTO.Name.Substring(0, fileDTO.Name.IndexOf("_ign")) + ".txt";
    foreach (DirectoryInfo dSub in d.GetDirectories())
    {
        foreach (FileInfo f in dSub.GetFiles("*.txt"))
        {
            organismName = "";
            kingdom = "";
            dnaMolecule = "";
            family = "";

            if (fileName.CompareTo(f.Name) == 0)
            {
                List<String> listLines = new List<String>();
                listLines = GetLines(f.FullName);
                int i = 0;
                foreach (LineDTO lineDTO in fileDTO.Lines)
                {
                    string s;
                    for (int k = i; i < listLines.Count; i++)
                    {

                        s = listLines.ElementAt(i);
                        // Console.Write(s);
B                       if (s.IndexOf("  ORGANISM  ") >= 0)
                        {
                            organismName = s.Substring(s.IndexOf("  ORGANISM  ") + "  ORGANISM  ".Length).Trim();
                            log.AppendLine("Extraindo informações (Foward) Organismo => " + organismName);
                            i++;
                            s = listLines.ElementAt(i);
C                           kingdom = s.Split(';')[0].Trim();
                            try
                            {
D                               family = s.Split(';')[1].Trim();
                            }
                            catch
                            {
                                family = "No Data";
                            }
```

Fonte: Própria (2020).

Quando o arquivo já foi inteiramente lido e todas suas informações carregadas nas variáveis *structs*, são iniciados os procedimentos para a gravação no banco de dados. Nesse momento, é realizada a chamada do procedimento

SaveIntergenicDBFile, conforme Figura 40, marcação A. Na tela inicial de configurações do importador, existe uma opção chamada “Salvar arquivo texto”, quando marcada, após salvar as informações no banco de dados, é realizada a criação de um arquivo do tipo texto no repositório (B) que contém as informações que foram inseridas no banco de dados.

Figura 40 – Procedimento para salvar

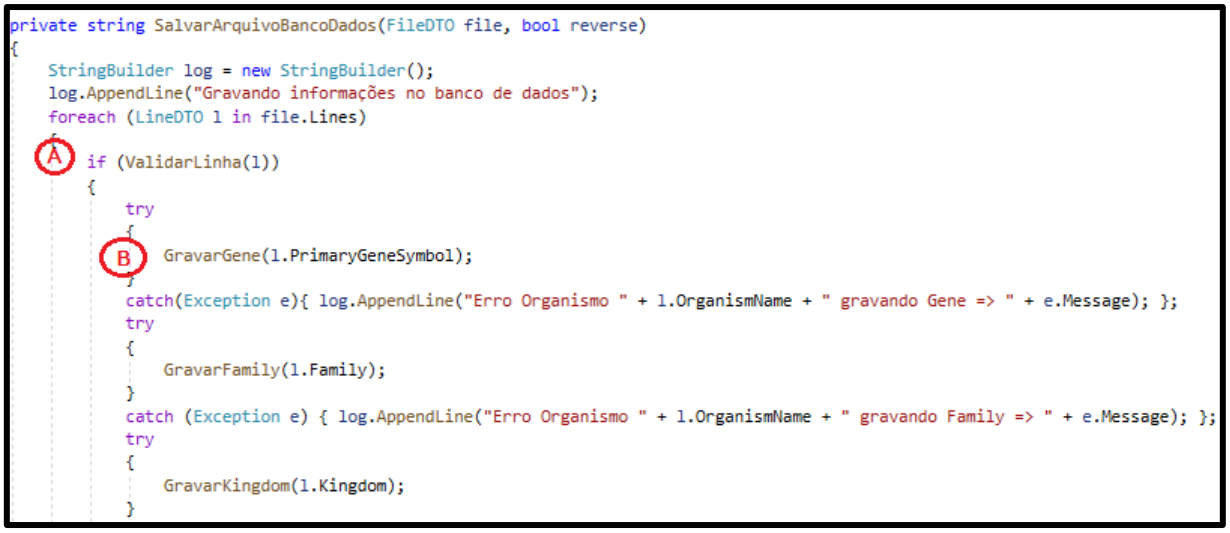

```
private string SaveIntergenicDBFile(FileDTO file, string source, bool reverse)
{
    DirectoryInfo dIntergenicDB = new DirectoryInfo(source + "\\ArquivosIntergenicDB");
    if (!dIntergenicDB.Exists)
    {
        dIntergenicDB.Create();
    }
    string log = SalvarArquivoBancoDados(file, reverse);

    if (configurationDTO.SalvarArquivoTexto)
    {
        using (var outputFile = File.CreateText(dIntergenicDB.FullName + "\\" + file.Name))
        {
            outputFile.Write("Primary Common Name" + '\t' +
                             "Primary Gene Symbol" + '\t' +
                             "Primary 5'" + '\t' +
                             "End   Primary 3'" + '\t' +
                             "End   Primary GC" + '\t' +
                             "Mainrole" + '\t' +
                             "Kingdom" + '\t' +
                             "Family" + '\t' +
                             "Organism Name" + '\t' +
                             "DNA Molecule" + '\t' +
                             "Start_SequenciaIntergenic" + '\t' +
                             "End_SequenciaIntergenic" + '\t' +
                             "Sequencia_Intergenic" + '\t' +
                             "Length_SequenciaIntergenic" + '\n');
```

Fonte: Própria (2020).

No procedimento *SalvarArquivoBancoDados* – Figura 41 -, as informações são validadas (A) para garantir que não sejam gravados registros inconsistentes. Após validação, é iniciada a gravação de cada tabela de forma individual (B).

Figura 41 – Procedimento de gravação das informações




```
private string SalvarArquivoBancoDados(FileDTO file, bool reverse)
{
    StringBuilder log = new StringBuilder();
    log.AppendLine("Gravando informações no banco de dados");
    foreach (LineDTO l in file.Lines)
    {
        if (ValidarLinha(l))
        {
            try
            {
                GravarGene(l.PrimaryGeneSymbol);
            }
            catch(Exception e){ log.AppendLine("Erro Organismo " + l.OrganismName + " gravando Gene => " + e.Message); };
            try
            {
                GravarFamily(l.Family);
            }
            catch (Exception e) { log.AppendLine("Erro Organismo " + l.OrganismName + " gravando Family => " + e.Message); };
            try
            {
                GravarKingdom(l.Kingdom);
            }
```

Fonte: Própria (2020).

Cada procedimento chamado dentro da *SalvarArquivoBancoDados,* é individual para chamada da camada de acesso ao banco de dados, conforme Figura 42.

Figura 42 – Procedimento de chamada da gravação

```
private void GravarGene(string symbol)
{
    Gene gene = new Gene() { Symbol = symbol };
    GeneDAL geneDAL = new GeneDAL(configurationDTO);
    geneDAL.Salvar(gene);
}
```

Fonte: Própria (2020).

A partir da chamada do procedimento *geneDAL.Salvar* – Figura 43 -, são realizados testes para verificar se essa informação já está registrada no banco de dados (A), a fim de evitar registros duplicados. Caso não esteja, é realizado o *insert* (B) desses dados em sua determinada tabela.

Figura 43 – *Insert* dos dados

Fonte: Própria (2020).

5.5 INSTALAÇÃO

Essa seção irá mostrar os passos necessários de instalação para funcionamento correto da ferramanta MMDBImportTool.

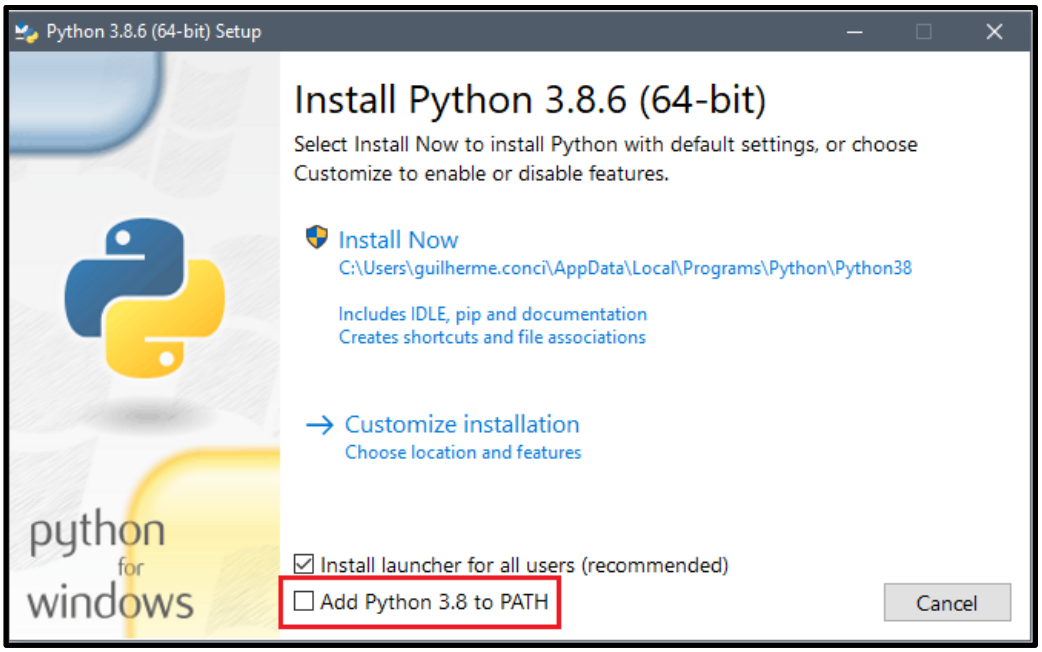
5.5.1 Python e BioPython

A MMDBImportTool utiliza *scripts* em *Python*[20] para alguns processos e por isso, é necessário a instalação para *Python* para a interpretação do código. A versão que deverá ser instalada é a 3.8.6 para manter a compatibilidade com o importador.

Após *download*, é necessário que seja executado o instalador, marcando a opção "*Add Python 3.8 to PATH*", conforme Figura 44.

[20] https://www.python.org/downloads/release/python-386/

Figura 44 – Instalação do *Python*



Fonte: Própria (2020).

Quando a instalação do *Python* for finalizada, é necessário realizar a instalação do *BioPython*. Para isso, é necessário abrir o *prompt* de comandos do computador e executar o comando "*python –V*". Esse comando irá retornar a versão instalada do Python. Caso a versão mostrada no *prompt* de comandos seja 3.8.6, então poderá ser feita a execução do comando "*pip install biopython".*

5.5.2 Importador MMDBImportTool

A instalação da ferramenta em si, é bastante simples. Basta que a lista de arquivos a seguir estejam todos dentro do mesmo diretório.

- MMDBImportTool.exe
- MMDBImportTool.exe.manifest
- MMDBImportToolBLL.dll
- MMDBImportToolDAL.dll
- MMDBImportToolDTO.dll
- MMDBImportToolUtil.dll
- Npgsql.dll

- 01_intergenic_muchos_archivos.py
- 02_seleciona_foward_intergenic_bac.py
- 02_seleciona_reverse_intergenic_bac.py
- 03_corrigeSequenciaReversa_bac2.py

5.6 TESTES

Essa seção irá descrever todos os testes que foram realizados e os resultados encontrados na execução da MMDBImportTool para a geração da nova carga de dados do IntergenicDB.

5.6.1 Testes realizados

Para a fase de testes da MMDBImportTool, inicialmente foram realizados testes de organismos individuais, para verificar se os dados gravados nas tabelas estavam consistentes com o arquivo importado. Para essa fase, cerca de 15 organismos foram escolhidos de forma aleatória para a importação. Durante a primeira fase de testes, nenhum dos organismos apresentou erros, tendo todas suas informações importadas com sucesso.

Após a primeira fase de testes concluída, foi criado um novo banco de dados local, para garantir que não haveria nenhum dado anterior e então iniciada uma nova fase de testes.

Na segunda fase de testes, a ferramenta de importação foi executada de forma completa e automatizada, com seleção de todos os organismos disponibilizados pelo Genbank, tendo duração total de 4 dias. Quando a importação foi concluída, foram observadas inconsistências nos dados. Durante a execução da ferramenta, em determinados momentos ocorreram erros de *timeout* do banco de dados. Além disso, por alguma razão não conhecida – podendo ter como causas o próprio ambientes de testes -, a MMDBImportTool parava sua execução, porém não apresentava nenhum tipo de erro em tela.

Desta forma, alguns arquivos de dados não eram processados até o final, e com isso, não eram gravadas todas as informações no banco de dados. Como os

erros descritos não apresentavam nenhum padrão, foi realizada uma terceira fase de testes.

A terceira fase de testes foi iniciada em um banco de dados novo e durante essa fase, os organismos foram importados de forma manual, até que todos fossem importados, com duração total de 3 dias. Com a importação de forma controlada, os organismos eram importados um a um, realizando anotações sobre cada um deles. Então, foi possível identificar um padrão para os erros que estavam acontecendo. Para alguns organismos, a importação ocorria apenas no sentido *forward,* ocorrendo erro ao tentar executar a extração de dados no sentido *reverse*. Cerca de 20 organismos se encontravam nessa situação e foram marcados para novos testes após o término da importação.

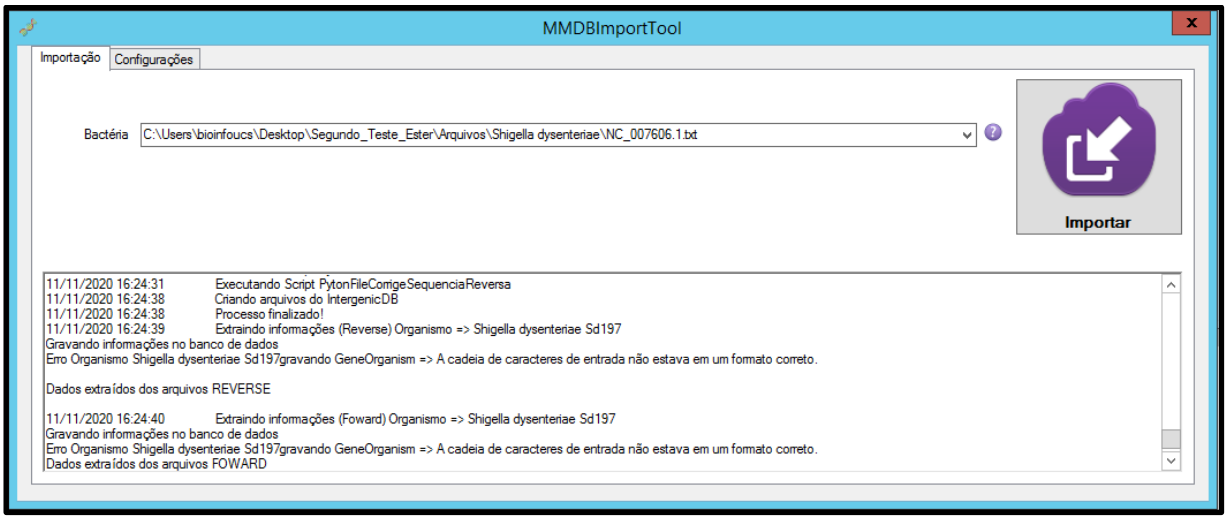
Após a finalização da importação, foi realizada uma nova tentativa para esses organismos, e então, após algumas tentativas, a maior parte foi importada com sucesso nos sentidos *forward* e *reverse*. Porém, o erro ainda persistiu para 10 organismos. Então, foi executada uma importação em um banco de dados paralelo, em outro ambiente de testes, para garantir que o problema não fosse relacionado ao banco, mas o erro ocorreu novamente.

Por esse motivo foi realizada uma verificação nos arquivos desses organismos para buscar possíveis causas do erro – como algum caractere diferente - porém nada anormal foi encontrado. Entre esses organismos estão *Aliivibrio fischeri, Bacillus subtilis, Burkholderia mallei, Burkholderia pseudomallei, Escherichia coli UMN026, Leptospira interrogans serovar, Rhodobacter sphaeroides, Streptomyces coelicolor, Vibrio cholerae* e *Vibrio parahaemolyticus.*

Ainda durante a terceira fase de testes, juntamente com os erros no sentido *reverse*, foram identificados arquivos do Genbank de organismos que não tinham a cadeia de caracteres de entrada correta, resultando no erro conforme Figura 45. Foi realizada uma verificação em todos os arquivos desses organismos, mas novamente, nada anormal foi encontrado.

Para esses organismos, foi realizada uma nova solicitação de *download,* para buscar por possíveis correções fornecidas pelo Genbank e realizar uma nova tentativa para a importação, mas o erro permaneceu. Entre esses organismos com cadeira incorreta então *Bordetella pertussis Tohama, Buchnera aphidicola, Escherichia coli IAI39, Shigella dysenteriae* e *Streptococcus agalactiae.*

Figura 45 – Erro na cadeia de caracteres de entrada



Fonte: Própria (2020)

5.6.2 Resultados dos testes

A carga do IntergenicDB realizada em 2017, conforme Figura 46, contém 15095 genes, 17 famílias e 88 organismos. A primeira carga de testes completa contém 7042 genes, 17 famílias e 110 organismos, enquanto a segunda - e final -, carga de testes, contém 9209 genes, 18 famílias e 118 organismos.

Figura 46 – Comparativo de dados

	Carga de 2017	Primeira carga completa de testes	Segunda carga completa de testes
Genes	15095	7043	9209
Famílias	17	17	18
Organismos	88	110	118

Fonte: Própria (2020)

Apesar da carga final de testes conter mais organismos, ainda assim, possui menos genes registrados, sendo uma diferença significativa de 8114 genes – quase metade de carga. Mas ainda assim, ouve aumento em relação a primeira carga completa de testes, onde o número de genes era apenas 7043.

É possível que essa diferença de registros seja uma consequência dos problemas de importação da carga citados na seção anterior, como os organismos com problemas na cadeia de caracteres de entrada ou em relação a extração das informações no sentido *reverse*.

Em relação a quantidade de organismos, na primeira carga de testes, o número havia aumentado em 22 organismos na comparação com a carga atual do IntergenicDB, realizada em 2017, porém, na carga final, esse número subiu significativamente em 30 novos organismos importados, totalizando 118 registros. Na Figura 47, consta os 30 organismos que foram incluídos na carga final de testes em relação a carga de 2017.

Figura 47 – Organismos incluídos

Acinetobacter pittii PHEA-2
Aeromonas hydrophila subsp. hydrophila ATCC 7966
Aliivibrio fischeri ES114
Bacteroides fragilis YCH46
Bordetella bronchiseptica 253
Bordetella parapertussis Bpp5
Brachybacterium faecium DSM 4810
Burkholderia mallei ATCC 23344
Caulobacter vibrioides CB15
Caulobacter vibrioides NA1000
Clostridium botulinum A str. ATCC 3502
Enterococcus faecium DO
Gardnerella vaginalis ATCC 14019
Klebsiella aerogenes KCTC 2190
Lactobacillus paracasei ATCC 334
Legionella pneumophila subsp. pneumophila str. Philadelphia 1
Mesorhizobium ciceri biovar biserrulae WSM1271
Mycobacteroides abscessus
Pseudomonas putida KT2440
Pseudomonas syringae pv. syringae B728a
Sinorhizobium fredii NGR234
Sinorhizobium meliloti 1021
Staphylococcus epidermidis ATCC 12228
Streptococcus agalactiae 2603V/R
Streptococcus mitis B6
Streptococcus mutans UA159
Streptococcus pyogenes M1 GAS
Streptococcus sanguinis SK36
Thermanaerovibrio acidaminovorans DSM 6589
Treponema denticola ATCC 35405
Vibrio parahaemolyticus RIMD 2210633

Fonte: Própria (2020)

O número de registros de famílias, se manteve igual na comparação da carga de dados de 2017 com a primeira carga completa de testes, porém em comparação com a carga final, aumentou em 1 registro, passando de 17 para 18 famílias. A família que foi incluída na carga final de testes é a *Planctomycetes.*

6 CONCLUSÃO

Para o desenvolvimento dessa monografia, foi realizada uma pesquisa sobre bancos de dados biológicos, primários e secundários, fazendo uma relação com as suas características e os tipos de dados que guardavam, e consequentemente com o objetivo principal desse trabalho, o banco de dados IntergenicDB. Além disso, também foram pesquisados métodos que poderiam ser usados para esse fim, como o ETL. A ferramenta usada para a importação dos dados do IntergenicDB, a MMDBImportTool, foi criada em 2014, e por esse motivo, também foi realizada uma pesquisa sobre evolução de *software*.

Durante seu andamento, foram apresentadas as necessidades de alteração na ferramenta MMDBImportTool, envolvendo o diretório FTP do Genbank, arquivo GBK e uso de *threads* para melhoramento do processamento. Porém, algumas alterações haviam sido feitas anteriormente, para realização da última carga de dados em 2017. Por esse motivo, não foi possível fazer nenhuma comparação em relação ao melhoramento do processamento pelas *threads*, já que a versão anterior, não pode mais ser executada pelas mudanças que ocorreram do diretório FTP do Genbank. As alterações realizadas anteriormente, não haviam sido documentas, então, foi realizada uma documentação completa para futuras alterações da MMDBImportTool, contendo informações sobre a estrutura e os principais procedimentos da ferramenta, como o *download* dos arquivos, a execução dos *scripts* em *Python*, a separação dos dados e a gravação no banco de dados.

Como as alterações que já haviam sido feitas, o processo não pode ser considerado uma evolução de *software*, porém, é considerado uma evolução na base de dados, visto que a carga atual possui inúmeras diferenças da carga anterior em relação a genes, famílias e organismos. Em relação ao processo de ETL, continua sendo utilizado pela MMDBImportTool para fazer a extração e a transformação de todos os dados dos arquivos do Genbank que são importados no IntergenicDB.

Na fase final, onde ocorreram três etapas de testes, foram verificadas inconsistências em relação a alguns arquivos de entrada do Genbank e também erros na execução da ferramenta. Porém, os erros na ferramenta, não foram determinados, visto que não ocorriam dentro de nenhum padrão, podendo ter como causa o servidor

ou o banco de dados em que a importação estava sendo executada e não a ferramenta em si.

Na carga final de testes, foram identificados um número maior organismos que na carga anterior, porém, o número de genes está consideravelmente menor, sendo assim, será necessária uma análise mais complexa em relação a esses dados para determinar a consistência das informações dessa carga.

Por se tratar de informações biológicas, os dados são de alta complexidade, tornando difícil a importação - pelo fato dos arquivos do Genbank serem muito extensos, com muitas informações e pelo próprio tipo de informação, que dificulta a criação de tabelas e índices no banco de dados -, e também a verificação das informações importadas, visto que exige um vasto conhecimento em biologia para que seja possível a realização de uma análise sobre esses dados.

A nova carga de dados do IntergenicDB será de grande valor para o grupo de pesquisa em bioinformática da UCS, pois com ela será possível dar continuidade aos trabalhos e pesquisas sobre bactérias gram negativas que já vem sendo realizados.

6.1 TRABALHOS FUTUROS

Para futuros trabalhos, há a possibilidade de revisão dos *scripts* em *Python*, buscando uma alternativa para a biblioteca *BioPython* em linguagem *C#*, para que seja possível a eliminação do uso de *scripts* externos, desta forma, todo processo desenvolvido ficará dentro da própria MMDBImportTool, tornando mais fácil a manutenção e instalação.

REFERÊNCIAS

OLIVEIRA, Gustavo Borges de. **Bio-TIM - Ambiente para convergência de informações em Bioinformática**. 2005. 100 f. Dissertação (Mestrado em Ciências Exatas e da Terra) - Universidade Federal de São Carlos, São Carlos, 2005. Disponível em: https://repositorio.ufscar.br/handle/ufscar/627. Acesso em: 6 abr. 2020.

ZAHA, Arnaldo; FERREIRA, Henrique Bunselmeyer; PASSAGLIA, Luciane M.P. **Biologia Molecular Básica**. 5. ed. Porto Alegre: Artmed, 2014.

NOTARI, Daniel. et al. **Biological Databases Integration: A Data Warehouse Perspective Applied to Intergenic Regions.** 2019. Universidade de Caxias do Sul.

JUNK, Soon et al. **Chado use case: storing genomic, genetic and breeding data of Rosaceae and Gossypium crops in Chado**. PudMed. 2016. Disponível em: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4795932/. Acesso em: 25 abr. 2020.

SOMMERVILE, Ian. **Engenharia de Software**. 9. ed. São Paulo: Pearson Prentice Hall, 2011.

PRESSMAN, Roger S. **Engenharia de Software Uma Abordagem Profissional**. 7. ed. Porto Alegre: AMGH, 2011.

SAYERS, E. W. et al. **Genbank**. PudMed. 2018. Disponível em: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6323954/. Acesso em: 9 mai. 2020.

GENEBIO. **Genbank**. Disponível em: http://www.genebio.ufba.br/genbank/. Acesso em: 1 jun. 2020.

MIYOSHI, Newton Shydeo Brandão. **Genômica translacional: integrando dados clínicos e biomoleculares.** 2013. 84f. Dissertação (Mestrado em Ciências) – Universidade de São Paulo, São Paulo, 2013. Disponível em: https://teses.usp.br/teses/disponiveis/95/95131/tde-18072013-100518/en.php. Acesso em: 6 abr. 2020.

DALZOCHIO, Jovani. **Implementação de novas funcionalidades para o portal IntergenicDB e povoamento do seu banco de dados.** 2014. 74f. Dissertação (Bacharelado em Ciência da Computação) – Universidade de Caxias do Sul, Caxias do Sul, 2014.

NOTARI, Daniel. et al. **IntergenicDB: Banco de dados de regiões intergênicas de Bactérias Gram-Negativas.** 2018. Universidade de Caxias do Sul.

BECKER, Bob. **Subsystems of ETL Revisited**. Kimballl Group**.** 2007. Disponível em https://www.kimballgroup.com/2007/10/subsystems-of-etl-revisited/. Acesso em: 25 abr. 2020.

KEGG. **Kyoto Encyclopedia of Genes and Genomes**. Página inicial. Disponível em: http://www.genome.jp/kegg/. Acesso em: 7 abr. 2020.

BERTOLDI, Ester Risério Matos. **Modelagem e implementação de banco de dados clínicos e moleculares de pacientes com câncer e seu uso para identificação de marcadores em câncer de pâncreas**. 2017. 90f. Dissertação (Mestrado em Bioinformática) – Universidade de São Paulo, São Paulo, 2017. Disponível em: https://teses.usp.br/teses/disponiveis/95/95131/tde-14032018-150144/en.php. Acesso em: 6 abr. 2020.

GENBANK. **National Center for biotechnology Information**. Página inicial. Disponível em: https://www.ncbi.nlm.nih.gov/genbank/. Acesso em: 7 abr. 2020.

KANEHISA, Minoru. et al. **New approach for understanding genome variations in KEGG**. Oxford Academy. 2018. Disponível em: https://academic.oup.com/nar/article/47/D1/D590/5128935. Acesso em: 9 mai. 2020.

TRISTÃO, Cristian; LIFSCHITZ, Sergio. **Protein world database: geração do esquema lógico e processo de ETL.** 2009. 25f. Dissertação (Bacharelado em Ciência da Computação) – Pontífica Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2009.

TANENBAUM, Andrew. S; BOS, Herbert. **Sistemas operacionais modernos**. 4 ed. São Paulo: Pearson Education do Brasil, 2016.

BECK, Kent. **TDD: Desenvolvimento guiado por testes**. 1 ed. Porto Alegre: Bookman, 2010.

KIMBALL Ralph; CASERTA Joe. **The data warehouse ETL toolkit: Practical techniques for extracting, cleaning, conforming and delivering data.** Indianopolis, USA: Wiley Publishing Inc. 2004.

HOOD, Leroy; ROWEN, Lee. **The Human Genome Project: Big science transforms biology and medicine**. PudMed. 2013. Disponível em: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4066586/. Acesso em: 25 abr. 2020.