FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

A Toolbox for Genomic Studies

Carlos Miguel da Silva Pereira

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Rui Camacho (FEUP)

Co-Supervisor: Pedro Ferreira (I3S/IPATIMUP)

March 24, 2019

A Toolbox for Genomic Studies

Carlos Miguel da Silva Pereira

Mestrado Integrado em Engenharia Informática e Computação

March 24, 2019

Abstract

Understanding how entire genomes, which represent the whole biological sequence of an organism, rather than individual genes, influence the development of a species biology is a powerful process into making public health more responsive and widely available. This often requires processing extremely large amounts of loosely organized data and apply a large number of software tools. The scarcity of high performance and scalable computational resources to do so, directly influences the flexibility of the currently used tools. Analysis at the genome level is crucial to understand the origin and treatment of cancer. It is also quite relevant for medical care evolution towards personalized medicine.

Presently used platforms do not provide a dynamic way to adapt existing solutions to version updates and computational advances. That particular aspect turns into a problem when an entire platform has to be re-arranged, when a specific tool needs to be updated or when the studies need to scale-up. We propose the development of a platform that will scale easily with the amount of data and complexity of genome studies. It will also offer scientists the use of a diverse amount of tools for genome studies all integrated in via the use of RESTful Web Services and usable though a single Web portal. These specifications will have a direct impact on how genome-related research is done mainly due to how flexible the solution aims to be. When a small subset of the aforementioned tools have an update or can make use of a faster machine for its own service, the change can be applied only to the particular tool. Allowing researchers to have modern tools will benefit public health as genome research progresses.

We will also propose a uniformed representation language that will allow to export any type of data for Data Mining analysis.

Resumo

Ter a possibildade de entender como o genoma, que consiste em toda a sequência de informação genética de um organismo ao invés de genes específicos, influencia o desenvolvimento de uma espécie, é um processo preponderante para o melhoramento da saúde pública. Um procedimento de estudo do genoma implica, por norma, o processamento de grandes quantidades de dados referentes ao mesmo que carecem frequentemente de coesão estrutural, organizacional e sintática. A falta de recursos e plataformas computacionais escaláveis e eficientes para este tipo de investigação tem um impacto directo na evolução e desenvolvimento da área de pesquisa, bem como da flexibilidade das ferramentas usadas atualmente. A análise feita ao nível do genoma é crucial para o estudo e prevenção de doenças como o cancro mas também para a evolução dos cuidados médicos generalizados para a medicina personalizada.

As plataformas e ferramentas usadas atualmente não permitem a adaptação de soluções existentes a atualizações de versões e avanços computacionais, sem comprometer outros constituintes das mesmas. Neste aspeto surge um problema quando é necessário reformular toda a plataforma para atualizar uma ferramenta específica ou para escalar os recursos computacionais para estudos específicios. Este projeto propõe o desenvolvimento de um conjunto de ferramentas integrado que seja fácil de escalar de acordo com a complexidade e exigência do estudo genómico em causa. Pretende-se também proporcionar aos investigadores um acesso único e centralizado, através do uso de *RESTful Web Services* e acessível através de um portal *Web* único. Todas estas características terão influência direta no modo como a pesquisa sobre o genoma é levada a cabo devido à flexibilidade da solução a implementar. Quando surgir a necessidade de atualizar uma ferramenta em particular ou proporcionar melhores recursos computacionais à mesma, apenas essa ferramenta será alterada sem comprometer o funcionamento ou disponibilidade das restantes funcionalidades da plataforma. Devido à quantidade de dados não estruturados com que os investigadores têm que lidar, pretende-se também desenvolver um linguagem de representação uniformizada que permita a exportação de todos os dados associados ao estudo do genoma. Esta linguagem vai permitir a aplicação de estudos na área do *Data Mining* sobre a informação exportada.

Acknowledgements

Given the time invested into making this dissertation work possible, there were people and other factors that vitally helped towards making the process with less bumps than it could have had for me.

First, I would like to thank my parents and a very scarce group of close friends for supporting all my tribulations during this important period of my life.

I want to thank the Macro-to-Nano Human Sensing: Towards Integrated Multimodal Health Monitoring(NanoSTIMA) and Analytics/NORTE-01-0145-FEDER-000016 projects. They are funded by the North Regional Operational Program (NORTE 2020), under the partnership agreement PORTUGAL 2020 and the European Regional Development Fund - ERDF for the availability of data used to carry out this project.

Secondly, none of the developed work would be possible without the crucial help from my supervisor Rui Camacho. Besides providing me with the scientific and technological knowledge needed for the project, he also showed a tremendous amount of support and understanding all throughout this journey. Even with his personal needs he managed to always help and find time to look at my progress.

Lastly I would like to thank FEUP for such a fantastic support over the past six years by providing me with the best superior education I could ever ask for.

Carlos Pereira - MIEIC

"What is not started
will never get finished"

Johann Wolfgang von Goethe

Contents

List of Figures

LIST OF FIGURES

Abbreviations

API	Application Program Interface
BAM	Binary Alignment Map
cDNA	Complementary Deoxyribonucleic Acid
CDS	Coding Sequence
CRISP-DM	Cross-industry Process for Data Mining
DBMS	Database Management System
DNA	Deoxyribonucleic Acid
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IUPAC	International Union of Pure and Applied Chemistry
JSON	JavaScript Object Notation
KEGG	Kyoto Encyclopedia of Genes and Genomes
mRNA	Messenger RNA
NCBI	National Center for Biotechnology Information
RCSB PDB	Research Collaboratory for Structural Bioinformatics Protein Data Bank
RDBMS	Relational Database Management System
REST	Representational State Transfer
RNA	Ribonucleic Acid
SAM	Sequence Alignment Map
SQL	Structured Query Language
SVM	Support Vector Machine
tRNA	Transport RNA
WWW	*World Wide Web*
WEKA	Waikato Environment for Knowledge Analysis

Chapter 1

Introduction

This particular chapter presents an overview of this dissertation's objectives, motivations and work performed. It will also serve as a structural summary for the project development and for this document.

The suitable and possible technologies will also be described in order to demonstrate how the dissertation will be supported on a technological standpoint.

1.1 Context

While genetics focus on studying individual and specific genes, genome oriented studies are done taking into consideration the organism's complete set of genetic information, its sequencing and behavior. This body of genetic information comprised of all the DNA present in a cell is referred to as genome.

Study of the genome has derived from early genetic research focused on analyzing either individual genes or small gene fragments.

Development in this field occurred due to the evolution of DNA sequencing methods, which allowed to properly document and map gene data for several organisms.

Being able to properly sequence entire genomes enables researchers to provide answers to several medical problems. In this scenario a genetic trait may have a key role in the problem development. Even though disease identification and prevention are the primary applications of genome studies, being capable of sequencing entire genomes from healthy individuals allows to predict and possibly find definitive remedies for health conditions that are virtually incurable so far.

Furthermore, this thesis work has as its main goal the development of a platform for the unification of a group of ever growing and diverse resources of information and data for the conducting of genomic studies. The context behind such platform arises from the lack of centralised options for conducting, monitoring and storing such studies, in a user friendly and scalable way.

1.2 Problem description and motivation

Even with all the potential applications mentioned above, the lack of scalable resources and platforms for genome related investigation, study and research still represents a limitation to building such systems. Given the resource scarcity associated with the mentioned field, it will also be the main motivation behind this dissertation project. Another goal concerns the definition of a standardized format to store genomic related data for data analytic studies, in order to help researchers deal with the extremely large amounts of data that result from such studies.

There are several research tools and sources of information spread through the web that represent a vast range of useful resources for genome related research. Each one these information providers require a different way of interacting with its interfaces in order to extract the desired data from them.

The lack of connection between said resources arises the need to gather an ever growing number of resources into a single platform. Together with a well structured group of web services to abstract those data gathering processes, this thesis work aims to effortlessly provide researchers with one tool only for genome research purposes.

The absence of a centralised, yet internally independent, source of information for this types of studies also contributes to the motivation behind this work. By building the system based on a *RESTful* approach, we aim to provide the system with the flexibility to update, maintain or rework any of its services without compromising the whole system integrity and functionality.

The development of such platform implies several advantages for any genetic related researcher investigation process and study data management. Such advantages are:

- **Centralised interaction for various repositories -** Giving researchers a standardized way of accessing independent databases and repositories strongly simplifies the process of conducting studies that require information from different sources.

- **Reduced specialized knowledge to interact with different API -** By having a single way of interacting with any data source, researchers would not need to spend time learning how to conduct studies in different platforms for different data gathering.

- **Reduced down time for the Toolbox -** With the use of Docker containers the entirety of the platform will likely spend less time without any service running.

- **Easy scalability -** Due to the technological and structural choices made for the development of the aforementioned platform, adding new services and continuous service development are significantly easier due to how scalable the Toolbox structure is.

Providing solutions for this problems directly impacts how efficiently and effectively genome research is done, which is translated into a better disease and health conditions assessment from medical professionals.

1.3 Project development

The work related to this dissertation comprises the development of a demonstration platform dedicated to genome related research as well as a structured data format for genome data handling and parsing.

The previously mentioned platform consists in a set of *RESTful* web services, distributed through different hosting machines. In each one of these machines each web service is deployed through a docker container, providing the system with a high level of scalability and flexibility.

Such characteristics allow the platform to have independent services that do not compromise the entirety of the system every time an update, machine fix or any kind of operation is required for a specific service. All of the remaining services will be able to be kept active while the particular docker container is down.

Researchers have a basic user profile and where they can search, conduct, perform and store genome related studies.

In order to centralise different kinds of data analysis services, several genome related web sites are taken into consideration. The ability to provide varied types of data from the developed web services, that retrieve information from different sites gives the application the opportunity of being constantly improved and added new services.

The ability to perform such studies is supported through the implementation of web services with Node.js. Having the ability to perform specific studies demands heavy computational capacity which requires FEUP's servers to be hosted. Having this web services developed allows for the platform to retrieve the resulting data after retaining the researchers input.

1.4 Document structure

This dissertation document is comprised of five chapters structured to introduce the state of the art regarding distributed genome analysis platforms and to describe the thesis work.

Chapter 2, "Molecular Biology Concepts and Technological Assessment", focus on introducing the main biological concepts associated with the intended work as well as an evaluation of the available technologies and resources to implement the necessary components. Available web service technologies and frameworks are listed to allow for proper tool selection and technological choices.

In Chapter 3 we detail the tools provided through the aforementioned *RESTful Web Services*. It is described what the services provide, how they work and how they can be traced and stored by the researchers on the developed Application. Each Web Service developed is detailed in this chapter in order to describe the range and capacity of the application. It presents the approach to the technological development of the Toolbox for Genomic Studies previously described. From its architecture, to the technological decisions made for the development.

Chapter 4, presents a set of case studies that illustrate the developed Toolbox.

Lastly, Chapter 5 provides a final summary of the developed work for this dissertation. It also illustrates the possible future work to be developed into the application. From the possible data mining implementations and usability, to new services to be added as the platform grows into a more dense and full source of genome related information for researchers.

Chapter 2

Molecular Biology Concepts and Technological Background

This chapter introduces the main concepts and processes related to Molecular Biology required to fully comprehend and follow the work developed for this dissertation.

It is also presented and detailed the gene expression process and its components, the standard formats used for genome data representation and how the RNA is processed.

It also presents the technological context for the Toolbox development regarding data mining, its importance to the thesis, the use and implementation of Web Services and the Database technologies available.

2.1 Molecular Biology

2.1.1 Relevant concepts

To fully understand and contextualize a group of concepts that will be continuously used during this dissertation development a list of definitions is provided below:

1. **Chromosome -** thread-like structure located in the nucleus of human and animal cells. Composed by protein and one molecule of deoxyribonucleic acid, also known as DNA, a chromosome is responsible for keeping the DNA structure during cell division and carrying the genetic information of the organisms through genes.

2. **Genome -** complete collection of an organism's hereditary information. Comprised of all of its genes. The nucleus of every human cell contains the copy of the individual genome.

3. **Gene -** specific DNA sequence that composes a section of the chromosome.

4. **Protein -** molecules consisting of one or more amino-acid chains, in a specific order according to the DNA that modeled the said protein.

5. **DNA -** molecule containing the hereditary information of every living being and most viruses. It is responsible for the proper development, growth and individuality of the aforementioned organisms.
6. **RNA -** although it can act as DNA on certain viruses, on humans and most living beings the RNA molecule is responsible for transporting information from the DNA to guide protein synthesis.
7. **mRNA -** temporary copy of the DNA's information after synthesis.
8. **tRNA -** specific RNA molecules responsible for transporting the amino-acids for polymerization.
9. **RNA polymerase -** essential protein to the gene transcription process, found in most living organisms.
10. **Exon -** represents any DNA or RNA sequence that will be a part of the final mature RNA sequence after the splicing process of RNA.
11. **Intron -** represents any DNA or RNA sequence that will be removed by RNA splicing and that does not code for protein formation.
12. **Alternative Splicing -** RNA splicing method that enables a single mRNA sequence to synthesize several protein variants with the ability of having distinct properties or functions.
13. **Fusion Gene -** resulting gene formed by two different genes.
14. **Genetic Code -** consists of the ruling by which the information present in the DNA and mRNA molecules is translated to proteins. A process handled by the ribosome.
15. **Biological Pathway -** is a set of operations that can occur in the inside of a cell and that can lead to different changes and processes for the mentioned cell.

2.1.2 Genome transcription process

Gene expression consists in the process of transforming a particular gene sequence into a fully functional genome based product. The entire process can be divided into two main parts: Gene transcription and gene translation.

Both of these separate processes are represented in a simplified fashion in Figure 2.1 and 2.2.

In this chapter, we focused on describing the gene transcription process as it relates to how the genome is transcribed.

The RNA polymerase enzyme goes through a particular gene sequence. It then goes on to create a complementary RNA sequence by destroying the hydrogen connections amongst the DNA nucleotide. This process originates the mRNA molecule that, in turn, exits the cell nucleus, heading into its cytoplasm in order to properly synthesize its associated protein.

[1]Image taken from: https://simple.wikipedia.org/wiki/Alternative_splicing

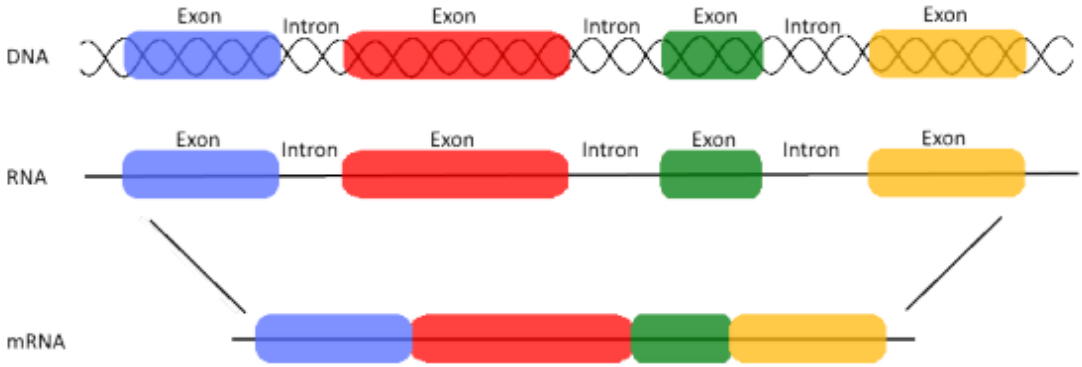

Figure 2.1: Simplified representation of the transcription part of the Gene Expression process [1]

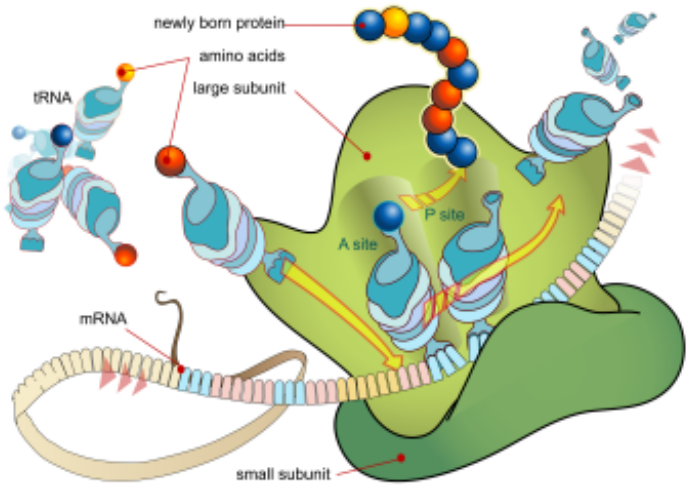

Figure 2.2: Representation of the translation part of the Gene Expression process [2]

2.1.3 Genetic Code

Consisting of a group of rules by which the genetic information (DNA and mRNA) is translated into proteins by living cells, the genetic code is also responsible for transcribing the genetic material into non-coding RNA. This particular type of RNA is able to properly regulate the gene expression process by acting as a compound of regulatory tools.

The process of decoding the biological information is carried out by the ribosome through the connection of amino-acids according to a mRNA defined order. The amino-acid transportation is done by the tRNA molecules which also allow for mRNA reading.

The rules associated with the genetic code define how nucleotide sequences can specify which particular amino-acid is added at a particular time during protein synthesis. Even though most genes are assembled using the same standardized genetic code there are exceptions where specific rules are applied, generating alternative, non-standard codes.

[2]Image taken from: https://en.wikipedia.org/wiki/Translationmedia/File:Ribosome_mRNA_translation_en.svg

2.2 Sequence Coding Formats

Currently there are a wide range of tools used in genome related research. Such tools can also use different types of sequencing coding formats for data presentation and processing. Some of this formats will be mentioned in this section in order to expose how data parsing can be affected.

The data presented represents a genetic sequence formed by a string of the letters A,C,G,T and U for RNA which represent respectively, adenine, cytosine, guanine, thymine and uracil, the nucleobases from nucleic acids. Uracil being RNA specific while thymine is found on DNA.

Plain Sequence Format

Fairly simple format as it can only take one sequence for file and can only use IUPAC characters. Listing 2.1 shows an example representation of sequence in plain sequence format.

Listing 2.1: Plain format

```
ACAAGATGCCATTGTCCCCCGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCCACCGCTGCCCTGCC
CCTGGAGGGTGGCCCCACCGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAATAAGGAAAAGCAGC
CTCCTGACTTTCCTCGCTTGGTGGTTTGAGTGGACCTCCCAGGCCAGTGCCGGGCCCCTCATAGGAGAGG
AAGCTCGGGAGGTGGCCAGGCGGCAGGAAGGCGCACCCCCCCAGCAATCCGCGCGCCGGGACAGAATGCC
CTGCAGGAACTTCTTCTGGAAGACCTTCTCCTCCTGCAAATAAAACCTCACCCATGAATGCTCACGCAAG
TTTAATTACAGACCTGAA
```

FASTA

It is a widely used format in Bioinformatics due to its simple structure and consequent simple parsing. Unlike the plain sequence format it is able to take in several sequences in the same file (see Listing 2.2). For the representation of each sequence two sections are annotated. The first line always starts with the '>' character followed by the sequence description (textual or numeric identification, any possible comments and its database).

After sequence identification on the first line the subsequent lines represent the sequence data.

Listing 2.2: FASTA format

```
>U03518 Aspergillus awamori internal transcribed spacer 1 (ITS1)
AACCTGCGGAAGGATCATTACCGAGTGCGGGTCCTTTGGGCCCAACCTCCCATCCGTGTCTATTGTACCC
TGTTGCTTCGGCGGGCCCGCCGCTTGTCGGCCGCCGGGGGGGCGCCTCTGCCCCCCGGGCCCGTGCCCGC
CGGAGACCCCAACACGAACACTGTCTGAAAGCGTGCAGTCTGAGTTGATTGAATGCAATCAGTTAAAACT
TTCAACAATGGATCTCTTGGTTCCGGC
```

FASTQ

Initially developed in order to incorporate sequence scoring in the FASTA format. Now it is widely used as a standalone format for sequence storing with its quality scores.

A FASTQ sequence storing example can be found in Listing 2.3.

Listing 2.3: FASTAQ format

```
@SEQ_ID
GTGGAAGTTCTTAGGGCATGGCAAAGAGTCAGAATTTGAC
+
FAFFADEDGDBGEGGBCGGHE>EEBA@@=
```

The first line represents the sequence identification started with '@'. As for the second line it contains the sequence characters, while the third line has a '+' followed by the sequence ID or any other comments to be made.

Lastly the fourth line represents the scoring data for the sequence in line number 2.

SAM and BAM

SAM (Sequence Alignment Map) is a text based format designed to store alignment data against reference sequences. It also has a binary version called BAM (Binary Alignment Map). It is widely accepted as the standard file format for aligned sequence information. The format structure consists in a optional header section, started with '@', placed before an ending alignment section. Sequences are supported in all sizes up untill 128Mbp. As it consists in the binary version of SAM, the BAM format allows for faster searches. SAM to BAM conversion can be done using SAMtools.

The information present in a SAM representation of a sequence consists in the following lines. Any line that is not started with '@' represents an alignment and are obligated to contain eleven data fields.

A small example can be found in Listing 2.4.

Listing 2.4: SAM format

```
@HD VN:1.5 SO:coordinate
@SQ SN:ref LN:45
r001 99 ref 7 30 8M2I4M1D3M = 37 39 TTAGATAAAGGATACTG *
r002 0 ref 9 30 3S6M1P1I4M * 0 0 AAAAGATAAGGATA *
r003 0 ref 9 30 5S6M * 0 0 GCCTAAGCTAA * SA:Z:ref,29,-,6H5M,17,0;
r004 0 ref 16 30 6M14N5M * 0 0 ATAGCTTCAGC *
r003 2064 ref 29 17 6H5M * 0 0 TAGGC * SA:Z:ref,9,+,5S6M,30,1;
r001 147 ref 37 30 9M = 7 -39 CAGCGGCAT * NM:i:1
```

2.3 Genome Related Repositories

As manual genome annotation is not feasible, several platforms to access, view, compare, store or analyze its data are available. Given the need to focus on data analysis and standardization, it is important to be able to have a wide range of genome related data accessible. Several repositories are available for information retrieval, each one comprising data related to particular areas of interest.

The Ensembl Project started being developed in 1999 and can be used to retrieve genome data for any vertebrate species. Acting as genome browser it allows for FTP based data retrieval. Its primary goal is to automatically handle genome data annotation in order to provide intuitive and user-friendly access to the information. Ensembl also provides a *RESTful* API for data sharing which is used latter in this dissertation as one of the services gathered and available at the web platform developed[3].

GenBank consists of another collection for DNA sequences and its corresponding protein translations. It is a free access platform that gets its data submitted by a wide range of different laboratories across the world[4].

KEGG is another database with genome related data mainly dedicated to comprehending high-level concepts of the molecular system. It provides a set of different data entry points for topic specific information. This allows users to narrow down the desired elements and areas of research[5].

For protein specific data retrieval both Uniprot and RCSB PDB represent viable sources.

Uniprot is comprised of four different components, each one providing a dedicated environment for different protein related information.

As for the RCSB PDB web api, its complete and accessible platform allowed for crucial protein related research services to be abstracted into the platform developed in this dissertation. It is also susceptible to be even more deeply embedded into the platform as it allows for a widely representation of protein and amino acid related structured information. In addition to that it also enables researchers to display 3D representation of proteins and acids which allows for a clearer understanding of molecular biology by scholars and researchers[6].

Reactome is an online resource for retrieving biological pathway data. It was used as a source of biological pathways for a service regarding the Toolbox. Being created by several specialist biologists it focus on human biological pathways. As an open source repository it is useful for several genome related studies[7].

Lastly, QuickGO was also used as a gene annotation repository as it supplies a extendable API for the Web Service development[8].

[3] https://ensembl.org
[4] https://ncbi.nlm.nih.gov/genbank/
[5] https://kegg.jp
[6] https://uniprot.org/
[7] https://reactome.org/
[8] https://ebi.ac.uk/QuickGO/api/index.html

2.4 Data mining in genome sequencing

Mapping and annotating genome data requires performing of a vast set of operations in large amounts of information often lacking any kind of organization. Data mining fulfills this need as it consists in the analysis and locating of useful data patterns in large amounts of information. Data mining methods and processes will be evaluated in order to satisfy this dissertation needs.

2.4.1 Data mining methods

As our ability to generate and collect data increases over time, the need for better and more efficient tools for handling those massive amounts of information also grows. Such methods allow for a broad range of tasks to be performed against data storage in order to extract knowledge from those sources.

CRISP-DM

The cross-industry process for data mining consists in a set of defined steps to approach and structure any data mining related project or research. As a widely used and proven methodology CRISP-DM divides the data mining process into six different and, at times, interchangeable phases. The order in which these different stages are executed can change according to project needs. This confers a more dynamic approach to data mining.

- **Business Understanding -** Initial project conceptualization. Business oriented tasks with a initial data mining problem definition.
- **Data Understanding -** The dataset in its original format. Data familiarization procedures in order to identify interesting data characteristic or groups.
- **Data Preparation -** Consists of all the activities associated with refining the final dataset for data mining purposes. Structure and organization are well defined at this point.
- **Modeling -** Given the dataset originated from the Data Preparation phase, several data modeling techniques are applied, and the process can go back to the previous phase due to data characteristics for a specific modeling approach.
- **Evaluation -** After a number of models have been implemented, all of them need to be verified against odd data to observe possible behavioral issues. When this phase ends the champion models have been selected.
- **Deployment -** Delivery of a user-friendly way of getting new data and knowledge from the original raw dataset.

2.4.2 Data mining tasks

The growing amount of unstructured data generated by diverse sources leads to the need of a way to extract viable knowledge from such information. Having the ability to discover and infer connections with such large and unorganized datasets allows researchers from the most various fields to automate and digitalize the process of Data Mining through the use of several techniques.

Data mining tasks aim to successfully find and register patterns against several kinds of data storage structures. Such tasks are used either descriptive, used to label and properly describe the data characteristics, or predictive, meaning they will be used to actually mine knowledge and find patterns in the required data.

Such tasks are described in this section in order to fully understand how data mining processes organize data.

- **Classification:** Consists in generating models to properly group data objects according to their characteristics. This particular task uses already classified objects and aims to correctly predict the class type of a object yet to be labeled. It is a descriptive task.
- **Regression:** Data mining tasks used to measure how object relationships affect the behavior of future values.
- **Clustering:** Aims to generate class labels for unlabeled objects. By not having the objects already labeled it groups the data in sets of similar objects. Data objects grouped in the same label class are very similar, while being as unrelated as possible to data objects in other classes.

2.4.3 Data mining tools

In oder to help on data mining tasks there are a variety of tools and facilitating platforms designed to ease development and knowledge extraction.

RapidMiner

RapidMiner is a platform that allows for integrated data mining project development and management. It abstracts machine learning and data mining processes for its users as it comprises a complete and integrated data mining development tool. With a XML-based scripting language it allows for clear data handling based on a modular operator concept [].

Orange

Orange is an open-source component based software tool for data analysis and display. It provides a diverse and wide toolbox for any type of user as it can be used for interactive data mining project visualization or directly with python script integration. It also provides the ability to extend its functionalities through the use of the available add-ons in order to amplify what the platform is able to provide to its users [Ora].

R programming language

R is a programming language specifically aimed at data analysis and mining as well as statistical studies and programming. R has a wide set of graphical properties to allow developers to visually explore the data. With all of the graphical attributes for data processing R helps shift developers focus to data analysis rather than coding related configuration tasks [Pro].

2.5 Web Services

Web services are software components accessible to other programs through the web using a well established protocol like HTTP. These functions allow other software applications to consume and integrate already developed and tested functionalities over a network.

Other programs communicate with the web service through a standardized Web format, ready to be consumed response.

Being able to implement web services that query and handle information from any web resource that possesses a web API able to be accessed through an established protocol, concedes the platform to be implemented to make available a wide number of services from diverse sources.

2.5.1 REST

Consisting of a group of architecture related principles to implement Web services and make them available, the REST architecture aims to develop such services in a scalable and HTTP based fashion.

The REST architecture has a set of principles making it the preferential way of developing scalable and efficient web services. This principles are listed below.

Client-Server

Both the client and the server in a RESTful service operate independently from each other. While the server side deals with data storage, handling and its availability to be requested, the client is responsible for displaying the requested information to the user as well as requesting more complex sets of data to the server according to previous requests.

Stateless server side

Not having to store state related information on the server side helps with scalability and flexibility for the applications. Every request should contain all the information necessary to be fulfilled, resulting in a totally independent server with no need to store any sort of information across requests.

Resource caching

In REST applications it is expected for every one of its components, being the client, the server or any other used elements to be able to resort to cache storing methods in order to improve performance and flexibility.

Layer oriented system

Any component interacting with a REST system will only be able to interact with the layer it is currently in. Not being able to see any other layers or how they interact enforce independence between components which fuels scalability and improves performance.

Code on demand

Even not being obligatory, being able to request entire code snippets or compiled libraries is also a feature of REST. This allows clients to perform certain heavy operations using code done beforehand.

2.5.2 Technologies

Implementing RESTful web services can be done with a variety of tools in different programming languages, as most of them offer functionality through libraries or default behaviour for such services implementations.

2.5.2.1 Node.js

Node consists in an asynchronous event driven JavaScript runtime [Cap]. This enables for several concurring connections to be established with a Node application. This particular characteristic allows Node.js applications to be easily scalable, trait that makes the development of web API able to be more efficient. By firing a callback for each connections established it guarantees the non-blocking functioning of Node. On the other hand if there is a connection established but no instructions to do, Node will not do anything.

2.5.2.2 Express

Express can be described as a minimalist and non-intrusive Node.js framework for web and mobile application development. Given the previous experience on working with the framework and its variety of HTTP related methods this was the chosen framework to implement the platform associated with this dissertation.

2.6 Database technologies

2.6.1 Relational databases

Databases are used to store and organize information in a certain form. Relational databases use tables to represent different types of objects abstracted from the data. Such objects have primary keys to serve as identifiers and can commonly refer to other objects with the means of a foreign key acting as a reference to another database table. This type of databases use SQL as a mean of assessing data and perform transactions with it, allowing for systems with needs for data reviewing [Hom].

Oracle database

Being one of the most widely used, Oracle Database is a Relational Database Management System and its used in many applications regarding data storage and warehousing.

MySQL

Another commonly used RDBMS is MySQL, an open source alternative that is highly scalable and suited for web site database handling. Praised as a secure database management system it can offer serious advantages for small Internet businesses in need for online shopping [Bra].

2.6.2 Non Relational databases

Non relational databases are able to store data without the integrity concerns added to the relational ones. When the data structure to be dealt with is not strongly defined, non relational databases represent a way to represent the information without having a strict schema. Such characteristics also increase the amount of resources needed to go through the data.

MongoDB

MongoDB represents the most used NoSQL database across multiple enterprises. With the flexibility of a document oriented approach, it has become a suitable alternative for handling massive amounts of loosely structured data.

The use of a non relational database when it comes to scalability eliminates structuring data tasks, also dealing with potential performance issues. The use of JSON also facilitates data retrieval and parsing as there is not a need for object type definition [Wod].

Chosen approach

Given the amount of data in need of storage to ensure the proper web service implementation, the approach chosen to handle such task was through a set of MongoDB databases. Each web service has an associated database in order to preserve independence between services.

With MongoDB being a non-relational database system, it provides the needed flexibility for any kind of study formats to be stored without the relational model implications of traditional RDBMS.

The volume of information to be stored also represents a factor that illustrates the need to efficiently escalate the platform web services in a RESTful way.

2.7 Docker, Containers and Virtual Machines

The environment, in which the developed web services run, represents a technological choice that can have an impact in how well the platform scales and behaves.

2.7.1 Virtual Machines

A virtual machine is a piece of software that reproduces a computer system, allowing for an operative system to run in its entirety through a virtual server. By emulating the hardware on which it runs, a virtual machine represents a way to have different operative systems running through the same physical machine.

2.7.2 Dockers and Containers

Docker is a tool used to improve how easily an application can be deployed and set up to run in an environment with a standardized set of dependencies and libraries strictly needed for the task. Such package of standard dependencies and libraries it is called a Container.

Docker acts as an engine able to turn container images into containers at run time in the Docker engine.

Containers enable for the application's software to be separated from the infrastructure on which it is running. Such level of independence allows for an application to be deployed with a set of advantages when through a Container.

The before mentioned advantages are listed below[BBR17]:

- **Speed** - Given the size of each Container needed to be built, the process of creating and deploying an application through one becomes fast and easy to accomplish.
- **Scalability** - Being able to be deployed in variety of environment and in any Linux machine allows for Containers to be easily moved from local to cloud environments. This enables for fixes to be made rapidly even with a growing application thus improving scalability. With this particular feature, the application developed in this thesis can have independent services hosted, which allows for any upgrade or machine fix not to compromise the entire service.
- **Portability** - With the ability to deployed as a single software package, a Container can be moved from hosting infrastructures with ease.

2.7.3 Docker Containers vs Virtual Machines

By allowing to initialize a Docker Container from a working virtual machine, both approaches can be combined for managing how applications are deployed.

Still, when used separately from Docker Containers, virtual machines have an additional layer in their architecture, that represents the operative system associated with each one of them. Having to meet the requirements for the whole OS to be run is not as lightweight as the Docker approach.

On the other hand, Docker allows for an application to be launched with just the needed software for that effect. This represents a more lightweight approach to deploying applications. When having a single machine or server hosting several applications, Docker Containers are independent from each other, which allows for a single application to be changed or updated without having to shutdown the entire server.

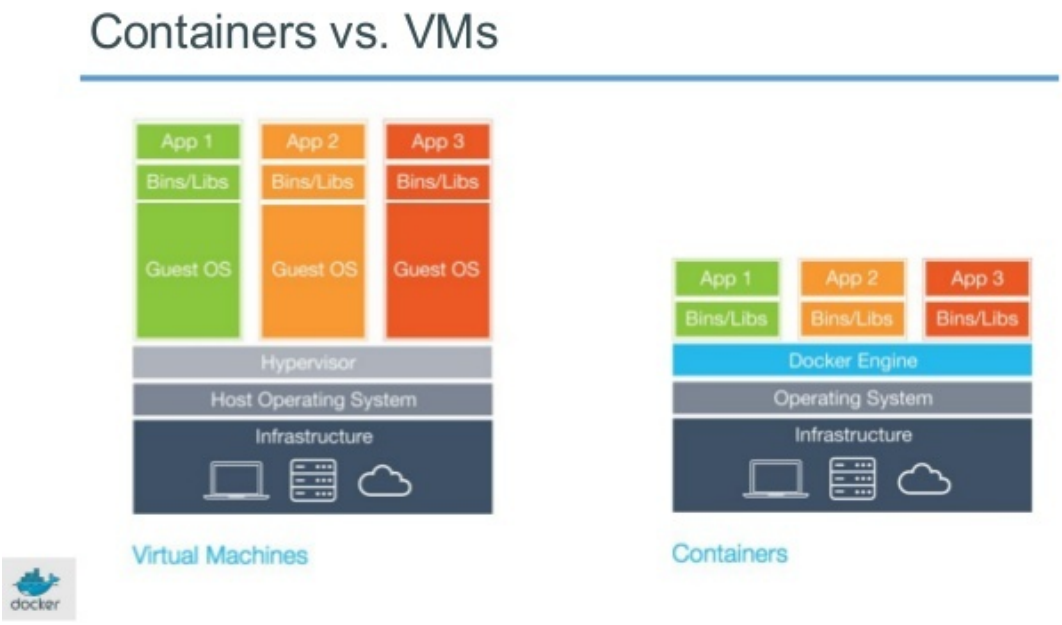

Figure 2.3: Overview and architectural structure for Docker Containers and Virtual Machines

Chosen approach

As for this dissertation work, Docker Containers were chosen as the deploy method. Using FEUP's servers, several containers were launched, each dedicated to a particular web service regarding a specific repository.

The ability to maintain, make code changes or update a service without compromising the remaining functioning system's stability is one of the main reasons why this approach was chosen.

2.8 Conclusions

On its initial sections this chapter introduced and explained a group of core molecular biology concepts and processes to contextualize this dissertation topic, necessities and work.

On latter sections the technological needs were displayed and developed in order to provide an insight on the wide range of approaches a project like this can take. Several technologies for the same goal were compared in order to present the reasons behind the chosen approach for each task.

It was also presented some repositories available for data retrieval that allow this thesis goal to be attained. All of this contributed to the definition of the state of the art regarding genome related research platforms and the idea behind a centralised Toolbox for such resources.

Chapter 3

A Toolbox for Genomic Studies

In this chapter we present and describe the structure associated with the Toolbox, the services to be available and the database architecture for the Toolbox for Genome research.

Initially this chapter illustrates the Toolbox architecture and organization, how the Docker Containers will be run and how the whole application will be hosted.

It also presents the services structure, organization and how they were developed and presented. Lastly we will describe how such services are implemented according to the REST architecture in order to ensure the scalability of the centralized Web application.

Lastly, a description of the MongoDB collections is presented in order to demonstrate how the services will interact through data sharing with the web portal.

3.1 System Architecture

In order to properly develop the described platform a wide set of technological decisions have to be made. Several components will be needed to come up with a structured solution. The researchers need a web portal where they can log in and keep track of the studies conducted in the platform. This web portal will be developed with HTML, CSS and bootstrap integration for styling and responsiveness purposes.

The overall structure of the developed platform has three main components responsible for providing the appropriate and desired features. The central component is the web portal in charge of researcher identification, login and study monitoring.

The remaining components compose a network of machines arranged in a flexible fashion, with several Docker Containers running in each one of them. This approach allows to have, with a smaller number of physical machines, several independent containers running the services separately helping on the flexibility of the platform. When the need for a specific web service to be updated or shut down for any reason, the remainder of the platform will keep running regardless of a particular part of it being inaccessible.

In order to be consumed by the web portal the final component is a set of web services, developed with Node.js, each of which, with a dedicated MongoDB database that aims to meet the specific service storing needs.

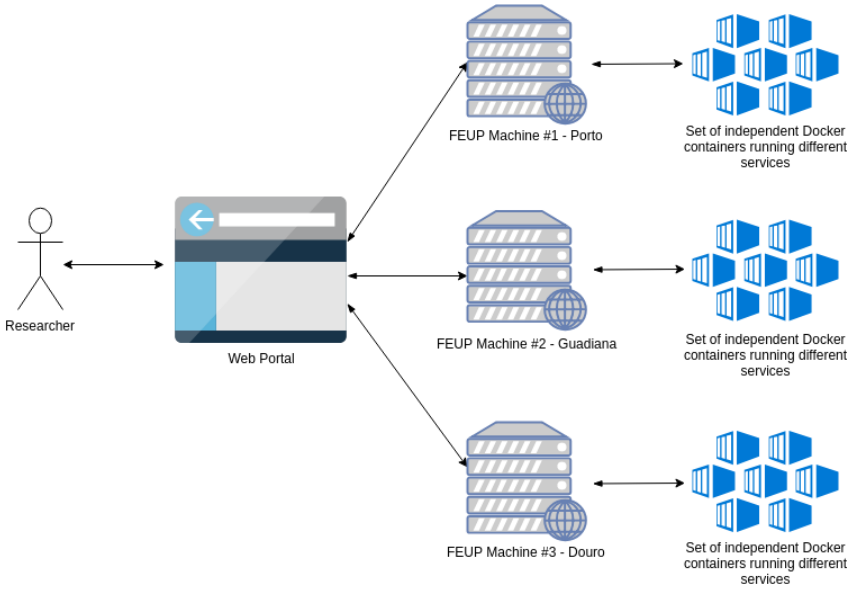

Figure 3.1: Proposed solution architecture

With scalability and flexibility being one of the main focus on the Toolbox development, having all the *RESTful* web services hosted on FEUP's machines, with Docker Containers for each service in that machine, allows to work towards those goals.

Services can be individually updated without compromising any other available web service. Services that demand a higher level of computation are hosted on adequate machines that can be independently maintained and upgraded in case of poor performance or unexpected malfunctioning.

3.1.1 Service Network

Like previously mentioned there are a multitude of technologies to implement web services to be consumed by an interface. The REST architecture will have to be enforced in order to give the platform the desired scalability and flexibility, while allowing for proper component integration.

Node.js

The web services for providing research techniques on the platform will be implemented with Node.js. Several reasons were taken into consideration regarding the web portal development. Previous development experience in the framework, easy NoSQL integration and the ability to develop scalable RESTful web services allow to meet all the platform requirements. With a non-blocking approach to web development Node.js presents a reliable way to merge back end and front end development with a scalable framework structure [Cap].

3.2 Web Services

Each one of the developed web services has its own simple MongoDB collection for storing information associated with the particular service needs.

3.2.1 Read alignment and realignment

When dealing with such huge amounts of data like in genome related research, read alignment occurs when, at the end of the sequencing process, parts of a cluster are stored.

Read alignment consists in aligning those cluster parts, being DNA, RNA or protein reads, against referenced genome sequences in order to establish possible associations between different sequences. This comparison allows for a proper identification of possible errors in sampled reads, which in turn provides researchers with the base for more complex and in depth studies and mutation identification.

3.2.2 Reference genome comparison

Reference genome represents a species entire genomic information. Comparing reads to previously annotated reference genome presents researchers with the possibility to find new healthy genome sequences, increasing the knowledge base regarding sequencing alignment.

3.2.3 Gene expression data enrichment

When processing genome sequences, the frequency in which specific patterns or even realizing certain gene classes are overly present is particularly important for disease pattern detection. Several tools are available for this process to be handled, most of them being statistical based.

3.3 Repository Access services

Given the vast and diverse amount of genome related repositories available in the web, any of their information retrieval functions can be added to the Toolbox either through the development of a new particular web service or by adding that specific functionality to an already existing service in the platform.

Having the ability to update and add functionality to any of the web services running through the Docker containers allows for a continuous addition of data retrieval operations to the already running platform.

Keeping in mind the amount of repositories available, it was chosen to develop a selected few as a way to validate the platform functionality and desired behaviour in a bigger scale, which can be seen as representation of the system in various sizes. Docker containers can be easily removed from machines and added to other environments with minimal effort given its simplicity and efficiency in software needed to run.

Below the chosen web services are described both in behaviour and in usage, given its operations and database models, showing how each container runs a structurally different web service, with fields that go to meet their specific data storage requirements.

3.3.1 Ensembl API Web service

Ensembl is a genome browser for vertebrate genomes that enables research in diverse genomic areas, from comparative research to sequence variation.

By extending its REST API as a web service, we allow for Ensembl's resources to be used for data retrieval and research in group with the remaining Web Services that extend genome related repositories[1].

Model for MongoDB Collection

The model developed for the Ensembl repository web service is displayed in Snippet 3.1.

Listing 3.1: "Ensembl Model Code Snippet"

```
// Ensembl Study Model
var ensembl_study_schema = new mongoose.Schema({
  speciesId : String,
  speciesAlias: String,
  req_type : String,
  format : String,
  feature : String,
  reg_init: String,
  reg_end: String,
  asm_1: String,
  asm_2: String,
  seq_type: String,
  dataset : Object
  }, {collection : 'requestcollection' }
);
```

In addition to the fields displayed in code snippet, it is automatically created by the MongoDB engine, an unique ID for each database entry that allows for study identification and retrieval.

Both, the *speciesID* and *speciesAlias* fields are destined to store Ensembl's identification for the species on which an operation is required. The first of which represents an Ensembl stable ID, that can have values like *ENSG00000157764*. The latter is aimed to the species name. The *req_type* field represents the type of request that was requested to the platform. As for the fields such as *format*, *feature*, *reg_init*, *reg_end*, *asm_1*, *asm_2* and *seq_type*, the set represents respectively the formats to emit from a specific endpoint, the requested feature in some operations, the

[1] https://ensembl.org

beginning and end of a region to be returned to the user, the versions of the input and output assemblies and lastly, the sequence type in a request.

The *dataset* field is aimed at storing the retrieved core data of a request. It is of type Object so it can have different structure considering the type of request it is storing information for [Ens].

Operations

The operations extended in this web service for the Ensembl repository are listed and described below:

- **Lookup** - Given any identifiers that are Ensembl compatible this service returns the database and species for the given gene, transcript or protein in the Ensembl repository. The endpoint for this operation is reached through */lookup/:id/:expand* from the web portal. With *:id* being the Ensembl id.
- **Archive** - Returns the latest version of the given identifier through the endpoint */archive/:id*.
- **Map** - This operation provides conversions from cDNA and CDS coordinates to genomic coordinates. It also allows to convert the coordinates of one assembly to another as well as from protein coordinates to genomic coordinates. Given the different types of coordinates conversions the endpoints are reached from the web portal through calling the endpoint, */map/:cdna_cds/:id/:reg_init/:reg_end* for cDNA and CDS coordinate conversion. With *:reg_init* and *:reg_end* being the delimiters for the region to be translated. As for assembly conversion a species ID is provided as well as two identifiers for the two assemblies in the operation. For the protein coordinates a species ID is needed as well as a region to be translated.
- **Overlap** - This operation is destined to return features such as genes, transcripts or variants from a provided id that overlap its region. Its endpoint can be reached through the */overlap/:id/:feature*. The *:id* is for the species identification and the *:feature* as for the desired feature to be returned.
- **Sequence** - Lastly, this operation returns several types of sequences by the Ensembl identifier provided and it can return such sequences in diverse formats such as FASTA, JSON and plain text.

Example Output

Snippet 3.2 illustrates a possible outcome of calling the developed Web Service for a lookup operation:

Listing 3.2: Ensembl Web Service Response


```
[
   {
      "_id": "5c3b69d1299f15642b6cce1b",
      "speciesId": "29",
      "req_type": "lookup",
      "format": "n/a for lookup",
      "feature": "n/a for lookup",
      "reg_init": "n/a for lookup",
      "reg_end": "n/a for lookup",
      "dataset": {
         "version": 1,
         "species": "homo_sapiens",
         "db_type": "otherfeatures",
         "strand": -1,
         "assembly_name": "GRCh38",
         "seq_region_name": "17",
         "source": "refseq",
         "object_type": "Gene",
         "end": 1187322,
         "logic_name": "refseq_human_import",
         "id": "29",
         "biotype": "protein_coding",
         "start": 1003518
      },
      "__v": 0
   }
]
```


3.3.2 Kegg API Web service

KEGG is an online resource and database that provides a way to access, analyse and understand a diversity of information related to gens like pathways, metabolic networks, genome sequences and large molecular data obtained from high resource computational services. It also provides a REST API for more customized searches and operations to be performed. This allows for our web service to use the mentioned API to retrieve information and return it to the web portal.

KEGG also provides several different entry points for specific biological entities such as KEGG BRITE, KEGG GENOME, KEGG GENES alongside many more, which enables a wide

and personalised range of services to be continuously added to the Toolbox[2].

Model for MongoDB Collection

The model developed for the KEGG API consuming web service in Snippet 3.3.

Listing 3.3: Kegg Model Code Snippet

```
// KEGG Study Model
var kegg_study_schema = new mongoose.Schema({
  req_type : String,
  kegg_id : String,
  dbentry: String,
  name : String,
  definition : String,
  orthology : String,
  organism: String,
  pathway: String,
  module: String,
  brite: String,
  position: String,
  dblinks : String,
  structure : String,
  aaseq : String,
  ntseq : String,
  dataset : Object
  }, {collection : 'requestcollection' }
);
```

Operations

The operations extended in this web service for the KEGG REST API are listed and described below:

- **Find** - This operation allows to search for specific data provided in a request like keywords or chemical formulas. It can be accessed through */find/:db/:data* with *:db* being the KEGG database targeted for the request and *:data* being a underscore separated set of keywords or chemical formulas.
- **Get** - Returns the information for any KEGG database entries, underscore separated, in a variety of formats available through */get/:dbentry/:data/:format*.
- **IdConvert** - This is a useful operation, that converts KEGG identifiers to other web resources for the same molecular entities. This particular operation can be reached through

[2] https://kegg.jp

/idconvert/:out_db/:dbentries being the two parameters respective to the outside database name and the KEGG database entries that are in need to have translated identifiers.

- **Link** - Operation implemented to cross examine through KEGG databases for genes or pathways.

3.3.3 GenBank API Web service

GenBank consists in the in a publicly accessible and annotated database resource used for DNA sequencing related research.

Providing information for more than one hundred thousand organisms it has become a widely used resource for biological research aid by a wide range of researchers from different molecular biology related studies [3]. This service also uses the PUG REST API for accessing PubChem data.

Model for MongoDB Collection

The model developed for the GenBank API consuming web service is displayed in Snippet 3.4.

Listing 3.4: GenBank Model Code Snippet

```
// GenBank Study Model
var genbank_study_schema = new mongoose.Schema({

  name: String,
  description: String,
  chromosome: String,
  aliases: String,
  gen_origin: String,
  map_local: String,
  symbol: String,
  info: String,
  sci_name: String,
  tax_id: String,
  dataset : Object
  }, {collection : 'requestcollection' }

);
```

Operations

The operations extended in this web service for the GenBank repository are listed and described below:

[3] https://ncbi.nlm.nih.gov/genbank/

- **Gene Search** - This operation allows for searching for all the information associated with a GenBank gene identifier. The required input is comprised of a *:db* parameter that is referent to the database in which the search will act. This is set to *"genes"* and a *:id* parameter for the aforementioned parameter.

3.3.4 Reactome API Web service

Reactome is an online resource that provides biological pathway information to researchers and users. With a wide set of data visualization and interpretation tools and an API for programmatic access its database allowing for personalized research and analysis [4]

Model for MongoDB Collection

The model developed for the Reactome API consuming web service is displayed in Snippet 3.5.

Listing 3.5: Reactome Model Code Snippet

```
// Reactome Study Model
var reactome_study_schema = new mongoose.Schema({
    identifier: String,
   pathwayID: String,
   feature: String,
   orthology: String,
   attribute: String,
   dataset : Object
   }, {collection : 'requestcollection' }

);
```

Operations

The operations extended in this web service for the Reactome API are listed and described below:

- **Pathway Feature** - This operation allows to search for a specific feature associated with a provided pathway. It can be accessed through */reactome_feature/:pathway/:feature* with *:pathway* being the provided pathway identifier and the feature parameter being the feature required for this request.

- **Orthology** - Returns the orthology information for any identifier provided through */reactome_orthology/:identifier*

[4] https://reactome.org/

- **Data Query** - This operation allows for the retrieval of all the information associated with a provided identifier. It returns all the links and database related information for that particular identifier.

3.3.5 QuickGO API Web service

QuickGO is yet another repository that provides a REST API in order to perform operation on the GO and GOA databases[5].

Model for MongoDB Collection

The model developed for the QuickGO API consuming web service is displayed in Snippet 3.6.

Listing 3.6: QuickGO Model Code Snippet

```
// QuickGO Study Model
var quickgo_study_schema = new mongoose.Schema({
   type_search: String,
  db: String,
  quickgo_id: String,
  quickgo_name: String,
  task: String,
  dataset : Object
  }, {collection : 'requestcollection' }

);
```

Operations

The operations extended in this web service for the QuickGO API are listed and described below:

- **GeneProducts** - This operation allows to search for a specific feature associated with a provided pathway. It can be accessed through */reactome_feature/:pathway/:feature* with *:pathway* being the provided pathway identifier and the feature parameter being the feature required for this request.

3.3.6 Repository Access Web Services Overview

Each one of these services provides part of the operations supported from the listed repositories. Given the ability to easily and quickly update the running Docker Containers, any operation can be developed, worked on and tested locally which turns such operation addition to a reduced time of the service being down.

[5] https://ebi.ac.uk/QuickGO/api/index.html

Also the responses from such services can be found in a latter section of this document, giving an insight on how the service network responds to the web portal calls.

3.4 Computational services

Unlike the repository access web services mentioned in the previous section, computational services do not interact with any web based repositories.

By being hosted in a specific machine for these kind of services, maintenance and scalability will be independent from the repository related services as it only communicates directly with the web portal.

3.4.1 WEKA

Consisting on a set of machine learning algorithms specifically dedicated to data mining requirements and tasks, Weka allows for data preparation, clustering, visualization and other operations[6].

Allowing for easily apply learning algorithms to a variety of datasets, users can performs either data management, machine learning or even dataset modifying operations through the WEKA service [EF16]. It will be used to allow researchers using the Toolbox to perform data mining related operations over already stored data from the repository access web services.

3.4.1.1 WEKA as a Web Service

Weka data analysis will be available using a Docker Container with Weka software and a wrapper script.

The wrapper script will take as arguments the *dataset* as arff file as well as the algorithm to run from the WEKA available ones. Using Docker technology we may add and remove any data analysis tool we want or move it to a more powerful machine if required.

Also, being able to use the functionalities present in WEKA set of tools allows researchers to perform more computationally demanding operations on the desired datasets.

Following the same fashion, other algorithm providing services can be inserted into the platform resulting from the ensured scalability and flexibility of the Toolbox.

The usability of such service is of the highest importance as it not only allows researchers to collect and request raw data it also provides machine learning algorithms to use on the collected dataset.

3.5 MongoDB Collections for the Web Portal

Given the existing need to deal with often unstructured data from genome related research, a NoSQL approach was chosen in order to properly deal with data to be processed. Not having a

[6] https://cs.waikato.ac.nz/ml/weka/

strictly defined structure for each object to be inserted helps to parse heterogeneous data that may be related to the same studies [Wod].

Using MongoDB, a NoSQL database, allows for storing extremely large data files without the strict normalization rules associated with relational databases. MongoDB stores information in collections regardless of its format and associated files.

The Web Portal for the Toolbox acts as a centralised repository that handles the already performed studies for future consultation as well as user authentication and role.

3.5.1 User Collection

Each user on the Toolbox has the necessary fields to characterize its profile and origin. A server side generated unique ID, the user's full name, username and password. Which are the login needed fields.

Also for each user there is a type field associated to it which indicates if the user has administrator rights or not. Both the institution and the field are identifying fields for the user's reasoning behind using the Toolbox.

This specific collection will be used for the web portal user management and session handling. Through storing important user information it allows to have access for each of the services provided by the Toolbox, taking into consideration user access, restrictions and previously done study requests.

3.5.2 Studies Collection

In order to store the studies performed by each user, a Studies collection was created, linking a unique identifier to each study allowing to retrieve said study to the user without having to compute the request again. Each user will be associated to the studies it has done in order to have a organized and useful profile in the Toolbox.

This collection also aids in controlling how the researcher requested tasks are handled and stored which will allow for data collecting and mining for several factors like what kind of studies are requested more often and so on.

Being able to keep track of the studies helps researchers to be more efficient while performing tasks over data sets for several databases and web repositories.

3.5.3 Block Diagram

In Figure 3.2 we can see the three main application blocks and how they interact with each other. Being the main difference between the two possible types of Docker containers being the kind of service they host.

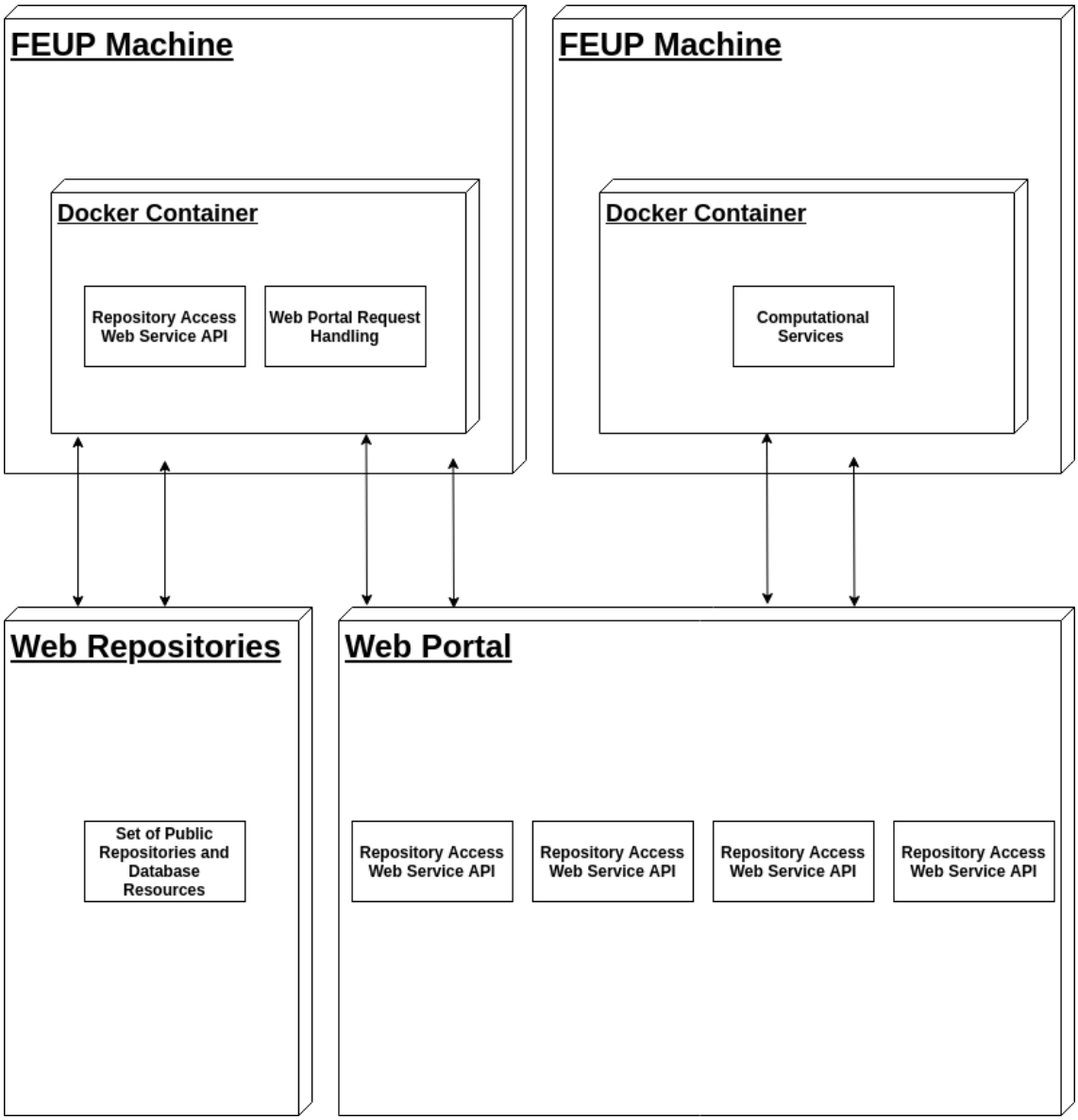

Figure 3.2: Proposed block model

3.6 Chapter Summary

Given the services described in the previous sections, the Toolbox for Genome Services will expand and complement work done in previous MIEIC thesis. Using Data Mining methods and the centralized Web Portal allows to continue developing a scalable and flexible platform for monitoring and conducting genome related studies.

Given the amount and diversity of possible operations in genome related data analysis, the Web Services implemented can have a future need to expand its functionalities. Such task can be developed and tested separately from the Toolbox, in a local environment, being then added to the web service network running on Docker containers without causing a major impact for the functioning of the rest of the system.

Chapter 4

Case Studies

In this chapter we used predefined datasets to perform data retrieval operations on the implemented Web Services. Each of the Web Services used in this case study is described in this part of the document, specifying input and output obtained.

One last case study illustrates the use of Weka data analysis tool.

4.1 Input Datasets

As input data for the case study to be performed on a set of developed or incorporated Web Services, a group of files. Such files were either generated from diverse gene and protein identifying codes or collected from a specific dataset for the effect.

One of the sites used was Pisces [1] and the specific list file used was *Proteins.list*, which represents a subset from a wider protein dataset.

The *Proteins.list* file is comprised of 3 277 entries.

As for the other datasets a file named *mitocondrialGenes.list* was used as input either as a whole or from generating subsets to be used to perform requests on the Web Services. This file contains 1 135 entries.

Lastly, the *protCodinggenesNoIsomorphs.list* file is a set of protein encoding genes that is made of 19 147 entries. Smaller subsets were used in some data retrieval operations.

4.2 Ensembl Information Retrieval

For this particular service a pre obtained list of protein encoding genes was used. A set of that list was treated as input for the web service.

4.2.1 Input

The input list file was the *protCodinggenesNoIsomorphs.list* file in *plain text* format. The aforementioned list file is exemplified by a small subset of its entries in the following file snippet:

[1] http://dunbrack.fccc.edu/PISCES.php

```
...
ENSG00000184389
ENSG00000128274
ENSG00000118017
ENSG00000094914
ENSG00000081760
ENSG00000114771
ENSG00000197953
ENSG00000188984
ENSG00000204518
ENSG00000109576
...
```

4.2.2 Output Information

The Ensembl Web Service ran and parsed the file line by line executing information retrieval operations and returning the results as JSON objects corresponding to the previously described model in section 3.3.1. The *dataset* parameter will store JSON formatted information as in the Ensembl API.

The researcher is allowed to store the returned dataset as a JSON or plain text file.

4.3 Kegg Information Retrieval

Similarly to the Ensembl API part of the study, an input list file was used in order to provide the starting data for the information retrieval from the Web Service.

Like the last case tested, the file used was the *protCodinggenesNoIsomorphs.list* file.

4.3.1 Input

The aforementioned list file is exemplified by a small subset of its entries in the following file snippet:

```
...
ENSG00000144827
ENSG00000106077
ENSG00000100997
ENSG00000131969
ENSG00000139826
ENSG00000248487
ENSG00000114779
ENSG00000168792
...
```

4.3.2 Output Information

The Kegg Web Service read the input and called gene information gathering operations to be performed.

The results were exported as JSON objects corresponding to the previously described model in Section 3.3.2. The *dataset* parameter will store JSON formatted information as in the Ensembl API.

The researcher is allowed to store the returned dataset as a JSON or plain text file.

4.4 Reactome Information Retrieval

As for the pathway retrieval service a list of specific Reactome identifiers was used as an input dataset with 1568 entries.

Such entries are then translated into Reactome compatible identifiers and processed.

4.4.1 Input

The aforementioned list file is exemplified by a small subset of its entries in the following file snippet:

```
...
ENSG00000144827
ENSG00000106077
ENSG00000100997
ENSG00000131969
ENSG00000139826
ENSG00000248487
ENSG00000114779
ENSG00000168792
...
```

4.4.2 Output Information

The output of the pathway related information was again stored and retrieved as JSON objects for the researchers to use as desired.

This JSON output objects followed the MongoDB model illustrated in Section 3.3.4.

4.5 QuickGO Information Retrieval

Unlike the previously mentioned cases, the QuickGO API will use the identifiers in the *mitocondrialGenes.list* file as it has gene identifiers directly accepted in the QuickGO API.

4.5.1 Input

The aforementioned list file is exemplified by a small subset of its entries in the following file snippet:

```
...
PDK4
MDH2
MRPS27
CS
GRPEL1
DLAT
LRPPRC
DLST
PDHX
GFM1
MPC2
NDUFS1
MRPL46
ATP5E
...
```

4.5.2 Output Information

The desired information was outputted through a JSON or plain text file following the MongoDB model illustrated in Section 3.3.5.

4.6 GenBank Information Retrieval

The GenBank case was made with entries from the *mitocondrialGenes.list* file.

4.6.1 Input

The aforementioned list file is exemplified by a small subset of its entries in the following file snippet:

```
...
BCS1L
ERAL1
CMC1
MRPL28
TSFM
FXN
NFU1
...
```

4.6.2 Output Information

The desired information was outputted through a JSON or plain text file according to the MongoDB model illustrated in Section 3.3.3.

4.7 WEKA Computational Service

As for the WEKA service, a bash script developed called *runweka* that takes an input *arff* file, either *thyroidTest.arff* or *thyroidTrain.arff*, containing a list of identifiers and the data relative to such dataset.

The aforementioned script was developed for the case study conducted during the development of the this dissertation. It is easily adaptable to take any algorithm available on the WEKA tools as an argument as well as the entry dataset. It also can be seen below:

```
#!/bin/bash

OUTFILE="output"
MEMOPTION=" -Xmx8192M "
CLASSPATH="./"
Classifier="j48"
CLASS_POS="last"
CLASSPATH="weka.jar"

if [[ $# -ne 2 || $# != "2" ]]
  then
  echo "Syntax; runweka dataset algorithm"
  exit
   fi

case $2 in
"j48")
   Classifier="weka.classifiers.trees.J48"
 ;;
"svm")
   Classifier="weka.classifiers.functions.SVMreg"
 ;;
esac

\rm -f $OUTFILE
echo "[Running WEKA ...]"
java $MEMOPTION -cp $CLASSPATH $Classifier $parset -c $CLASS_POS -t
    ${1}"Train.arff" -T ${1}"Test.arff" > $OUTFILE

exit
```

4.7.1 Output Information

As it was mentioned before the output of the WEKA script will be passed to a *output.txt* file containing the results for the algorithm ran on the *arff* file provided as parameter.

The result of calling the WEKA computational service for the J48 decision tree algorithm on the *thyroidTrain.arff* will produce the output file in 4.7.1.

```
=== Classifier model (full training set) ===

J48 pruned tree
------------------

2 <= -0.00014
|   5 <= 0.000041
|   |   1 <= 0.000201
|   |   |   6 <= -0.000024
|   |   |   |   125 <= -0.000018: Healthy (19.0/1.0)
|   |   |   |   125 > -0.000018: Cancer (8.0)
|   |   |   6 > -0.000024
|   |   |   |   7 <= 0.000049: Healthy (362.0/4.0)
|   |   |   |   7 > 0.000049
|   |   |   |   |   97 <= -0.000004: Cancer (4.0)
|   |   |   |   |   97 > -0.000004: Healthy (16.0)
|   |   1 > 0.000201
|   |   |   72 <= -0.000107: Healthy (5.0)
|   |   |   72 > -0.000107: Cancer (28.0)
|   5 > 0.000041: Cancer (69.0)
2 > -0.00014
|   5 <= -0.000448
|   |   161 <= -0.000009: Healthy (11.0)
|   |   161 > -0.000009: Cancer (6.0)
|   5 > -0.000448: Cancer (288.0)

Number of Leaves :  11

Size of the tree :  21


Time taken to build model: 0.47 seconds

Time taken to test model on training data: 0.06 seconds

=== Error on training data ===

Correctly Classified Instances 811         99.3873 %
```

```
Incorrectly Classified Instances 5          0.6127 %
Kappa statistic                  0.9877
Mean absolute error              0.012
Root mean squared error          0.0775
Relative absolute error          2.4035 %
Root relative squared error     15.5033 %
Total Number of Instances      816


=== Detailed Accuracy By Class ===

               TP Rate FP Rate Precision Recall F-Measure MCC ROC Area PRC
                   Area Class
               1.000  0.012  0.988    1.000  0.994    0.988  0.995   0.991
                   Healthy
               0.988  0.000  1.000    0.988  0.994    0.988  0.995   0.995
                   Cancer
Weighted Avg. 0.994 0.006  0.994    0.994  0.994    0.988  0.995   0.993


=== Confusion Matrix ===

   a   b   <-- classified as
 408   0 |  a = Healthy
   5 403 |  b = Cancer

Time taken to test model on test data: 0.01 seconds

=== Error on test data ===

Correctly Classified Instances 110         96.4912 %
Incorrectly Classified Instances 4          3.5088 %
Kappa statistic                  0.7989
Mean absolute error              0.0363
Root mean squared error          0.1874
Relative absolute error          7.265 %
Root relative squared error     37.481 %
Total Number of Instances      114


=== Detailed Accuracy By Class ===

               TP Rate FP Rate Precision Recall F-Measure MCC ROC Area PRC
                   Area Class
```

```
               0.750  0.010  0.900    0.750  0.818    0.803  0.866   0.694
                  Healthy
               0.990  0.250  0.971    0.990  0.981    0.803  0.866   0.970
                  Cancer
Weighted Avg. 0.965 0.225 0.964    0.965  0.963    0.803  0.866   0.941


=== Confusion Matrix ===

  a  b  <-- classified as
  9  3 |  a = Healthy
  1 101 |  b = Cancer
```

4.8 Conclusions

After performing the described tests on the developed Web Services the application responded as expected. JSON objects were returned following the structures of the aforementioned models.

Given the high scalability of the solution proposed the addition of new services is eased. Routine maintenance tasks, service update, addition of new service features or machine repair operations will only affect the specific Docker Container on which said service is running. This approach will ensure the system will continue working even with a certain part of it being down for any reason.

Computational services are also easily added to the platform as they only need a new running Container and a set of parameter to run.

Given the returned results the platform behaved as expected by retrieving the desired JSON formatted information.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

Genome research represents a dense area for computer systems to help in data processing. After consulting and researching several molecular biology concepts one can comprehend the basic functioning of genome sequencing and annotation.

The technological aspect of implementing a toolbox dedicated to genome related studies demanded a careful investigation on what frameworks, languages and approaches are going to be used. The programming language does not represent a barrier of any sort as most modern ones allow for RESTful web service development. The choice of Node.js was purely based on the previous knowledge regarding the technology, as far as syntax goes.

When making use of pre loaded libraries is necessary, programming languages are not so irrelevant anymore. Python has a set of seasoned libraries used in Bioinformatics, but previous development experience with Node.js its consequent ease of development and the non-blocking functioning led to such decision.

The RESTful architecture used throughout the project reflects an important part of how scalable the final solution became. As a single threaded technology, not waiting for requests to finish before handling concurrent ones represents a non blocking approach that allows for multiple users to use the platform simultaneously with a seamless and efficient behaviour.

Having the Web Portal centralising all the services provides an easy to use interface for study consultation and monitoring.

5.2 Future work

5.2.1 Data Mining functionalities

The toolbox aims to provide to its users a set of data mining algorithm implementations in order to allow for statistical analysis and processing of study related data.

5.2.2 Classification algorithms

Decision trees

Decision trees are important in data mining as a mean of properly identifying all the possible outcomes of a problem with associated probability, costs and utility. Given this structure ease to be interpreted, ability to intuitively lay down a situation or problem and the little need for data in order to have value, it is a useful tool for genome related studies.

Artificial Neural networks

Artificial neural networks model entire computer systems around the structure of a human brain. This systems are able to progressively perform better over time by learning from experiences, input type and other variables associated with the computing process. As for its data mining importance such structures are especially suited to pattern recognition, do not make broad assumptions concerning the data generation process and model the data according to its use and experiences.

SVM

Support Vector Machines use the notion of decision planes to properly define different and appropriate classes for the dataset objects. Such planes are delimited by boundaries separating different types of objects. This method will also be available for researchers to use in order to provide several methods for object classification.

Random forest

Grouping a set of decision trees in order to generate knowledge and properly classify objects can be done with the random forest algorithm. This method builds a group of decision trees and displays the most common and recurring object class.

Naive Bayes

Another useful classification method is the Naive Bayes algorithm. This technique applies the probability related Bayes theorem to assign classifying values to objects in a data mining project. This probabilistic approach lowers the assumptions made regarding new data aggregation and influence.

5.2.3 Clustering algorithms

K-Means

K-Means consists in a unsupervised learning algorithm that is used to solve a clustering problem which makes it crucial for data mining on genome related research. The K-Means algorithm assumes a number of clusters where the objects should be inserted into. It then defines the same number of centroids, widely separate from each other. It then associates each point of the dataset to a certain cluster and recursively refine this steps in order to properly define clusters for the given datasets.

Density based clustering

While the K-Means algorithm has to necessarily find a cluster to place every object the density based method does not, which is particularly important for mutation detection. Without having abnormal objects in normal object clusters, anomalous points will no longer drag clusters with them. This allows for a very important mutation detection and proper clustering [Nan].

References

[BBR17] Mohammad Ahmadi Babak Bashari Rad, Harrison John Bhatti. An introduction to docker and analysis of its performance, 2017.

[Bra] Tony Branson. 8 major advantages of using mysql. Available at: http://www.datamation.com/storage/8-major-advantages-of-using-mysql.html. Accessed: 2018-02-06.

[Cap] Tomislav Capan. Why the hell would you use node.js. Available at: https://medium.com/the-node-js-collection/why-the-hell-would-you-use-node-js-4b053b94ab8e. Accessed: 2018-03-06.

[EF16] Mark A. Hall Eibe Frank and Ian H. Witten (2016). The weka workbench. online appendix for data mining: Practical machine learning tools and techniques, 2016.

[Ens] Ensembl. Ensembl rest api. Available at: https://rest.ensembl.org/.

[Hom] Jacqueline Homan. Relational vs. non-relational databases: Which one is right for you? Available at: http://www.pluralsight.com/blog/software-development/relational-non-relational-databases. Accessed: 2018-02-06.

[Nan] Manojit Nandi. Density-based clustering. Available at: http:///blog.dominodatalab.com/topology-and-density-based-clustering/. Accessed: 2018-02-08.

[Ora] Orange. Orange software. Available at: http://orange.biolab.si. Accessed: 2018-02-08.

[Pro] GNU Project. What is r? Available at: http://www.r-project.org/about.html. Accessed: 2018-02-08.

[Wod] Carey Wodehouse. Should you use mongodb? a look at the leading nosql database. Available at: http://www.upwork.com/hiring/data/should-you-use-mongodb-a-look-at-the-leading-nosql-database/. Accessed: 2018-02-06.

REFERENCES

Appendix A

Appendixes

A.1 Creating a Docker Container for the Toolbox

In order to set up a new Docker Container, some simple steps are needed. The simplicity of this process shows how scalable and flexible the developed platform is.

A.1.1 Installing Docker

Docker can be installed via command line with:

```
$ sudo apt-get install docker-ce docker-ce-cli containerd.io
```

A.1.2 Dockerizing a Node.js app

When a new Web Service is developed in Node.js it needs do be Dockerized in order to follow the project's structure to ensure scalability.

A Dockerfile needs to be created within the Node project directory.

```
FROM node:8
# Creates app directory
WORKDIR /usr/src/app
# Installs application dependencies
COPY package*.json ./

RUN npm install
# Bundle app source
COPY . .

EXPOSE 8080
CMD [ "npm", "start" ]
```

A.1.3 Adding Web Service Code to the API

If a new Web Service is developed and needs to be ran on a specific Docker Container, previously installed, all it needs to be done is to copy the Node.js Project code onto a directory in the destination environment, including the newly created Dockerfile.

A Container Image needs now to be built in the host machine. To do so, run the following command:

```
$ docker build -t <username>/<web_service_name> .
```

Finally, the image just needs to be ran using Docker through the next command:

```
$ docker run -p 49160:8080 -d <username>/<web_service_name>
```

The new Web Service is now running on a Docker Container and the endpoint is reachable for data collecting from the central Web Portal.

A.1.4 Calling the new Web Service

Depending on the structure used for the Web Service development the calls to its URL would depend on the developer. Following the example, the new endpoint would be requested as:

```
GET https://<host_machine_ip_address>/<web_service_name>/<params>
```