Universidade Federal do Paraná

Andressa Eloisa Valengo

miRSNaPi: integrating access to data on polymorphisms in microRNA target sites

Curitiba
2016

Andressa Eloisa Valengo

miRSNaPi: integrating access to data on polymorphisms in microRNA target sites

Trabalho apresentado como requisito parcial à conclusão do Curso de Bacharelado em Informática Biomédica, setor de Ciências Exatas, da Universidade Federal do Paraná.

Área de concentração: *Bioinformática*.

Orientador: Daniel Weingaertner.

Coorientador: Rodrigo Coutinho de Almeida.

Curitiba
2016

Acknowledgments

It has being amazing to study and work as a scientific trainee at Universidade Federal do Paraná. I'd like to thank **every good person** who works/studies at this great university, and without even knowing it, made this project possible. Thank you very much! Specially, many thanks to my co-workers of the **Laboratório de Genética Molecular Humana** and **Simepar**, who are just amazing people and incredible good at doing science. You're really great! For the **PET guys and girls**, thank you for being so nice! I even ate *pastel* because of you! By now, I can remember many people, many things, many little things, but I have being told that I can write only one page about acknowledgments [Somebody, 2016], so I'll try to keep it short, meaning that I can't name everyone I'd like to :(

You! Thank you for reading this simple work. Hope it can help you somehow. Share your thoughts about it with me! :)

Professor Maria Luiza, thank you for introducing me to the world of science. You're an amazing advisor and taught me how to approach the worlds' problems as a scientist, this is one of the best things that happened to me. Thank you very much.

Professor Rodrigo, thank you for actively helping me in this project and all your recommendations and thoughtful advice. You're a great person.

Professor Daniel, thank you for giving me the opportunity of going on one year exchange in Germany, in the Friedrich-Alexander-Universität, where I could learn so many things that helped me in my course, projects and life. Thanks for helping out with this project.

Professor Spinosa, thank you for your help, advice, and motivational words.

Marco Aurélio, thank you for helping me with the server stuff! You're a great person!

Biel, thanks for sharing your great ideas with me. Even this project, you're the one who first thought about it. Thank you for having confidence in me to get this job done!

Lê and Stael! Thank you for being so nice, and helping me every time I needed. You're amazing people and we´re all lucky to have you on our back, handling all the paper work like a charm.

Nicolas Pansardi, thank you for being such a good friend. Please let's have some hamburgers!

Vinícius, thank you for helping me with my projects and for being such good friend. Let's keep coding! :D

Aldayr, **Aninha**, **Apu**, **Bovs**, **Fran**, **Helena**, **Nicholas**, **Rafaela**, and **Raphael** thank you for attending classes with me, and also for being good friends.

Mom and dad, I can't think of being here, finishing this course, without your help. Thank you for my education, your love, and everything you gave to me. I love you! I'm sorry for the moments of absence (and stress! hehe). Thanks for being so understanding.

Brother (Toasty!), thank you for being so funny and for helping me in the moments of need. I'm sorry for being so absent in the last years and I thank you for being this awesome brother and friend. I love you! Be prepared for many mid X1! No Pudge allowed!

Thelma and Carlos, thank you for letting me being part of your life and helping me this much. I'm really thankful for everything you've done for me. Carlos, I just love your jokes :p

Li! Wow, how do I begin thanking you? Are there enough words? Thank you for sharing everything with me. Thank you for your patience, love, comprehension, and advice. Thanks for helping me in my studies (even this project!), English and everything. I really love you! Thank you!!! <3

Resumo

O aumento da quantidade de dados biológicos, em conjunto com as novas possibilidades de disponibilizá-los, permite o desenvolvimento de ferramentas e bases de dados nas mais diferentes áreas científicas. Esse é também o caso dos microRNAs (miRNAs) e sua variação genética. Considerando o contexto dos Polimorfismos de Nucleotídeo Único (SNPs) em sítios alvos de microRNAs (miRSNPs), existem muitas bases de dados que reportam predições sobre os efeitos dos miRSNPs. Pesquisadores que trabalham com miRNAs/miRSNPs comumente realizam buscas nas bases de dados para encontrar efeitos de miRSNPs que são reportados nas diferentes bases. Essa tarefa, quando realizada manualmente, pode ser muito cansativa, além de gerar resultados imprecisos. Até então, não existe, de acordo com o nosso conhecimento, nenhuma aplicação capaz de integrar o acesso aos dados de miRSNPs que estão espalhados em diferentes bases de dados *online*, o que justifica o desenvolvimento de uma ferramenta que faça esta integração. O objetivo deste trabalho foi o desenvolvimento de uma RESTful *application programming interface* (API) que integre o acesso às mais recentes bases de dados de miRSNPs. Cinco bases de dados foram selecionadas para integração, as quais foram encontradas através de buscas de artigos realizadas no *Pubmed*. Nossa API, chamada miRSNaPi (http://lgmh.c3sl.ufpr.br/miRSNaPi), possui uma interface uniforme com métodos HTTP para receber requisições de dados de miRSNPs (por gene, miRNA ou SNP), acessa as bases de dados selecionadas e envia os dados combinados como resposta. A API disponibiliza os dados no formato JSON, o que significa que aplicações externas, que solicitam os dados, podem utilizá-los de diferentes maneiras. Além da API miRSNaPi, também foi desenvolvido o *web client* miRdiver (http://lgmh.c3sl.ufpr.br/miRdiver), que faz uso da API (miRSNaPi), permite que o usuário busque por gene, miRNA ou SNP e ainda exibe em quais bases de dados os miRSNPs foram reportados, com uma interface responsiva e amigável.

Palavras-chave: microRNA, variação genética, interface de programação de aplicação, integração de dados.

Abstract

The growth of biological data, together with the new possibilities of making data available, allow the creation of different tools and databases that cover various scientific areas. That is also the case of microRNAs and their genetic variation information. Considering the context of the single nucleotide polymorphism (SNP) in microRNA target sites (miRSNPs), there are plenty of online databases full of predictions about the effect of miRSNPs. It is a common task for a researcher who works with microRNAs/miRSNPs to search over miRSNPs databases to look for strongly reported effects, and such task can be tiring and imprecise when done manually. To the best of our knowledge, there is no application that integrates the access to miRSNPs data that is spread out in different databases and this may explain the urge for a web service that fulfills this task. This project aimed to create a programmer-friendly RESTful Application Programming Interface (API) that integrates the access of the most recent miRSNPs databases. In order to select the miRSNPs databases, a *Pubmed* search was performed, resulting in five online databases to work with. Our API, named miRSNaPi (http://lgmh.c3sl.ufpr.br/miRSNaPi), has a uniform interface that uses HTTP methods to receive miRSNPs data requests (by gene name, miRNA name, or SNP identifier), accesses the selected online databases, and sends the combined miRSNPs data as a response. As the API provides the data in JSON format, the external applications are able to easily work with them in many different ways. Besides our main result, the miRSNaPi API itself, the icing on the cake is our miRdiver (http://lgmh.c3sl.ufpr.br/miRdiver) web client that makes use of miRSNaPi, allows the user to search by gene, microRNA, or SNP, and shows how frequently reported are the miRSNPs found, with a responsive and user-friendly interface.

Keywords: microRNA, genetic variation, application program interface, data integration.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

MicroRNAs (miRNAs) are small non-coding ribonucleic acids (RNAs) of approximately 23 nucleotides (nt) that play a role in post-transciptional gene regulation. After being processed, the miRNA will be incorporated into the RNA-induced silencing complex (RISC) that will target the 3' untranslated region (3'UTR) of a messenger RNA (mRNA). This binding depends on the sequence complementarity between the miRNA and the target and may lead to mRNA cleavage or translation inhibition.

Genetic polymorphisms, as single nucleotide polymorphisms (SNPs), can be located in a miRNA target sequence (miRSNPs), affecting the targeting relationship by changing the binding site. Their presence can cause the gain or loss of a miRNA binding site.

There are approximately two thousand miRNAs described in humans and there are many applications that predict the effect of SNPs on their target sites. These tools apply different computational techniques and can have therefore different results among them. To the best of our knowledge, there is no application that integrates the access to the spread out data on miRSNPs, meaning that every time one needs information about miRSNPs, one has to manually check every database web site.

1.1 Aims

This project aimed to create an Application Programming Interface (API) to integrate the data access of the most recent versions of miRSNPs prediction tools. In addition, an user interface that makes use of the API was also created to show the integrated data by using tables and graphical elements. Specifically the projected aimed to:

- Select the most recent/applied web-available databases of information on human miRSNPs.
- Develop a RESTful Application Programming Interface that integrates the data access of the selected databases and allows the user to search by gene, miRNA or SNP.
- Illustrate the use of the proposed RESTful API by developing a web application to generate a table showing gene-miRNA-SNP relationships along with the respective databases in which they appear.

Chapter 2

Background

This chapter covers the basic fundamentals about microRNAs, their function, and also how genetic variation can affect it. In addition, basic concepts about web services, Application Programming Interface (API), REST and RESTful are presented.

2.1 MicroRNAs

Even though microRNAs (miRNAs) probably exist for a long time, being present even in the genome of unicellular organisms [Molnar et al., 2007], they were not discovered until 1993, and only by the year of 2001 were recognized as a class of regulatory RNAs. MiRNAs play a role in post-transcriptional regulation of genes and it is thought that they can work coordinately in order to control pathways or biological functions. Additionally, the appearance and expansion of miRNAs seem to have contributed to the development of multicellular complex organisms, because they could add an extra level of regulation of gene expression [Heimberg et al., 2008].

MiRNAs can be classified, considering their genomic location, in: **1) intergenic miRNAs**; **2) intronic miRNAs**; and **3) exonic miRNAs** [Pinhal et al., 2015]. As shown in figure 2.1, each category may have more than one subclass, regarding the miRNAs having their own promoters. In addition, miRNAs in humans can commonly be arranged in *tandem*, what is known as miRNA clusters (Figure 2.1-A, Figure 2.1-B).

MiRNAs are well known to lack Open Reading Frames (ORFs); they are part of the non-coding RNAs (ncRNAs), meaning that miRNAs are not translated into proteins. Furthermore, mature miRNAs are single stranded and really small, having approximately 23 nucleotides (nt). Considering both conserved and non-conserved miRNA target sites, the majority of coding genes are regulated by at least one miRNA [Friedman et al., 2009], what could explain why miRNA biogenesis and function are strictly regulated. MiRNAs dysregulation can affect gene expression and in consequence biological paths and functions leading to diseases, such as cancer [Lujambio and Lowe, 2012].

A canonical view of the miRNA biogenesis is shown in the figure 2.2. MiRNAs are processed from a primary RNA, known as **pri-miRNA** (primary miRNA), which is firstly generated by the transcription process undertaken by RNA polymerase II (RNA Pol II). The pri-miRNA is made up of one or many hairpins (Figure 2.2) that are characterized by a region of double stranded RNA (dsRNA). Hairpins are also known as stem-loop and their size can vary, having more than one thousand nt. The nuclear enzymes Drosha and DGCR8 (DiGeorge syndrome critical region 8) recognize the pri-miRNAs and process them into the precursor miRNAs (**pre-miRNAs**) that are 60-80 nt long. The pre-miRNA is then exported to the cytosol by the protein Exportin-5, where the Dicer (Trans-activation-responsive RNA-binding) enzyme

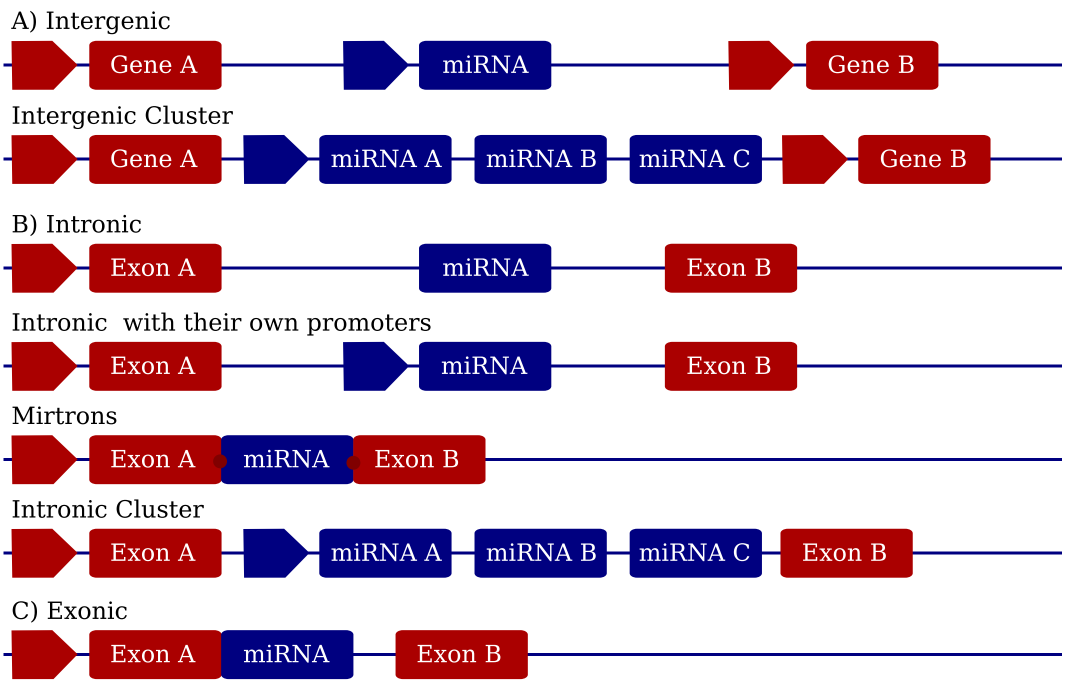


Figure 2.1: MiRNA classification considering their genomic location. Adapted from [Pinhal et al., 2015].

removes the loop from the hairpin/stem-loop. Finally, one of the RNA strands (the guide strand) is incorporated by the RNA-induced silencing complex (RISC) that contains the Argonaute protein 2 (AGO 2).

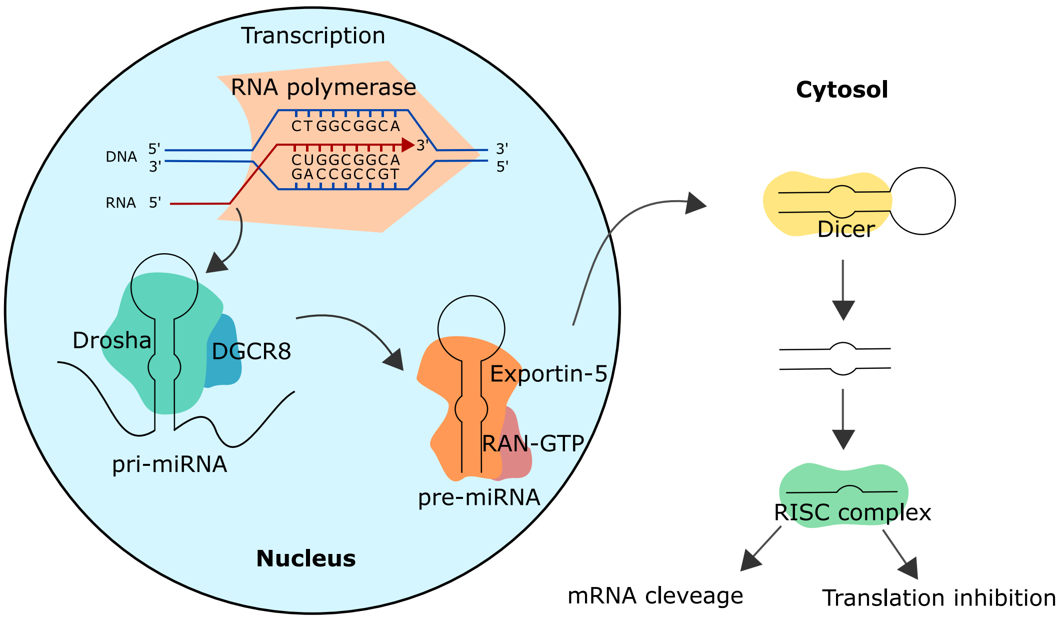


Figure 2.2: Canonical miRNA biogenesis. Source: the author.

The **miRISC** (RISC + miRNA) guided by the miRNA will then target an mRNA transcript promoting regulation of gene expression. MiRNA/mRNA biding depends on complementarity between the two sequences (Table 2.1). Furthermore, miRNAs have a region termed as *seed* (from the 2^{nd} to 8^{th} nt), which is determinant in the biding effectiveness and affects, in consequence, the way a given miRNA regulates gene expression. In a simplified way, a complete match between the *seed* sequence and the target may lead to **mRNA cleavage**, otherwise, the process may lead to **translation inhibition** (Figure 2.2). In addition, miRNAs can also act in: **1) transcriptional silencing**; **2) promoting transcription** and **3) translation enhancement** [Pereira et al., 2015].

Table 2.1: Complementarity among RNA nucleotides

Nucleotide (miRNA)	**Complementary nucleotide (RNA)**
C	G
G	C
A	U
U	A

2.2 MiRNA nomenclature

As shown in table 2.2, miRNA genomic locations (*loci*) are named using the prefix **mir**, while the mature miRNA (after being processed by Dicer) receives the prefix **miR** (capital letter R). These prefixes are followed by a hyphen (-) and an identification number (mir-#; miR-#). It is recommended to write *loci* name itself using italic font (*mir-#*) whilst the pre-miRNA using regular font (mir-#) [Lagos-Quintana et al., 2001]. When the miRNAs are clustered (Figure 2.1) they are normally processed together and named according to their location in the 5' → 3' sense. MiRNAs can also have paralogues that will result in different *loci* generating miRNAs with identical sequences, these miRNAs are named with an extra hyphen and number (mir-#-1, mir-#-2). On the other hand, similar miRNAs (but not identical) are named using lowercase letter after the identification number (mir-#a, mir-#b).

Table 2.2: MiRNA naming conventions

	loci	**pre-miRNA**	**mature miRNA**
Regular/Cluster	*mir-#*	mir-#	miR-#
Paralogue	*mir-#-1, mir-#-2*	mir-#-1, mir-#-2	miR-#-1, miR-#-2
Similar	*mir-#a, mir-#b*	mir-#a, mir-#b	miR-#a, miR-#b
Homo sapiens	*hsa-mir-#*	hsa-mir-#	hsa-miR-#

Considering miRNAs from different species, the miRBase [Griffiths-Jones, 2006], which is the most used miRNA database, recommends an additional prefix, for example: hsa-mir for *Homo sapiens* and dme-mir for the fly *Drosophila melanogaster*. Furthermore, according to the miRBase the miRNA identification numbers are given sequentially.

2.3 Genetic variation and miRNA function

The binding efficiency between the miRNA and its target site depends on sequence complementarity between the 3'UTR of an mRNA and the mature miRNA. Therefore, genetic polymorphisms may affect the miRNA function by changing the interaction between miRNA and mRNA sequences. In this context, the most described polymorphisms are the single nucleotide polymorphisms (SNPs). When SNPs are located in the miRNA target sequence (miRSNPs) they can cause: **1) miRNA target loss**; **2) miRNA target gain**; **3) binding enhancement**; and **4) binding weakening** when the SNPs alters the complementarity with the *seed* sequence (Figure 2.3). In the dbSNP database [Sherry et al., 1999], SNPs are named using the prefix *rs* (reference SNP) followed by a number (rs#).

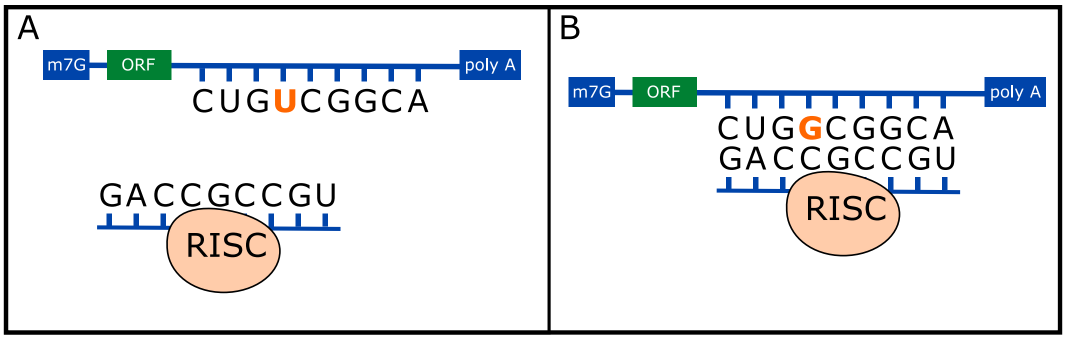


Figure 2.3: Representation of SNP effects on miRNA/mRNA binding.
A) represents miRNA target loss, and B) represents miRNA target gain. Source: the author.

SNPs with effects on miRNA/mRNA binding can be represented by using the Entity–relationship model (Figure 2.4). In simple terms, miRNA binds to an mRNA and this relationship can be affected by SNPs. Both miRNA and mRNA have their own names, as well as the SNP has its own rs. Finally, the SNP interference has its effect on the miRNA/mRNA binding.

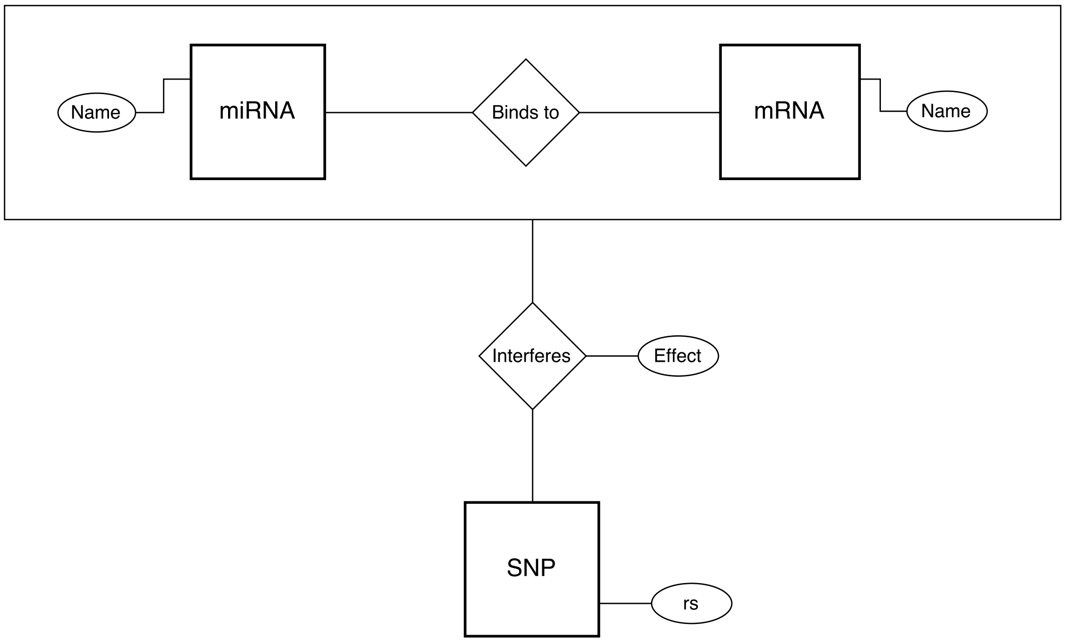


Figure 2.4: MiRSNPs - Entity–relationship model.

2.4 MiRNA: zooming in

If one searches for a miRNA in the miRBase, its sequence can be retrieved in the FASTA format:

```
>hsa-miR-9-5p MIMAT0000441
UCUUUGGUUAUCUAGCUGUAUGA
```

The above miRNA can be retrieved through the following address: `www.mirbase.org/cgi-bin/get_seq.pl?acc=MIMAT0000441`.

2.5 Application Programming Interface

An Application Programming Interface (API) is a set of rules defined by a program so other applications can communicate with it. There is a variety of APIs for many purposes, such as Java libraries, Ruby classes and methods, Python Packages, OpenGL, Apache CouchDB, and others. In addition, APIs are used to provide access to data; for example, Twitter has its own API that allows developers to retrieve *tweets* and complete *timelines*. Such API, known as Web API, are web services and generally communicate over the Hypertext Transfer Protocol (HTTP).

2.6 REST

Web based services are most of the time related to the Client-Server (CS) concept, where a server provides a service and the client makes requests to the server, which may perform a service in order to respond the requests [Fielding, 2000]. Web APIs are currently strongly related with the REST (Representational State Transfer) concept that was defined by Fielding in his doctoral thesis by the year of 2000. REST is a design criteria for distributed hypermedia systems that reflects the modern Web architecture created by the examination of its own constraints [Fielding, 2000].

When creating the REST style, Fielding started from a null style defined by the **World Wide Web** itself and added elements/constraints in the model. The first constraint added was the **Client-Server** architectural style; the idea of using this communication model is that it is simple and also implies that the server and client evolve in a separated way. The communication between the server and the client must be **stateless**, what means that every request the client makes to the server must be understood by the server without any previous information. Among other constraints, the **uniform interface** is the feature that most distinguishes REST from other network styles, because it applies the software engineering principle of generality to the component interface, the information/request is then transferred in a standardized manner [Fielding, 2000]. This information exchange standardization obligates different clients to adapt to the server interface. An API that applies REST design is called **RESTful** API.

2.7 Resource-Oriented Architecture

REST concepts described in the Fielding thesis are sparse and can be understood in different ways. For example, when specifying the concept of uniform interface, Fielding did not describe interface on the matter of which/how methods should be used to achieve interface standardization [Richardson and Ruby, 2007]. Considering this, Richardson and Ruby described the Resource-Oriented Architecture (ROA) in their book. ROA is RESTful because it is based in the following four features: **addressability**, **statelessness**, **connectedness** and **uniform interface** [Richardson and Ruby, 2007].

2.7.1 Resources

Web RESTful services expose their functionality as HTTP objects named resources [Richardson and Ruby, 2007]. That being said, RESTful services instead of using methods/functions applying the most different names for getting, for example, a list of objects, they all use the HTTP methods (GET, HEAD, POST, PUT, DELETE and OPTIONS). Going deeper in the getting object list example, one could create a function named *getBooksList()*, or one could create another function named *getSongsList()*. These two functions return different data, but they are basically the same, so instead of using different functions, one could use the HTTP GET method, making both functions uniform. This goes even more beautiful if one thinks that these lists are hosted in different places and both support the same basic operations.

A resource is a piece of an interesting information, a data set of something, a result of a script/algorithm, a picture, or something similar. Every resource needs a URI (Uniform Resource Identifier) to be addressed on the web [Richardson and Ruby, 2007]. In ROA, resources are just as important as their name (URI), their representation, and the links that connect them.

2.7.2 Addressability

The fact that a resource needs a URI to be something in the web agrees with the addressability concept. The URI itself is a resource.

2.7.3 Statelessness

As defined by Fielding, statelessness means the isolation of each request the client makes to the web service. Isolation makes it easier to cache the data.

2.7.4 Connectedness

Connectedness means that each resource is connected to the others that are part of the same web service. This is normally achieved by adding links in the resources themselves that make reference to the other resource URIs. The connectedness feature is mostly not applied by web services. However, web services that do not fulfill the connectedness concept are still considered RESTful [Richardson and Ruby, 2007].

2.7.5 Uniform interface

Uniform interface is achieved by using the HTTP methods, making all method names standardized.

2.8 More on HTTP

2.8.1 Safety and idempotence

HTTP methods have two properties: *safety* and *idempotence*. A method is safe when it does not require any change to the server state. This means that making a safe request (GET or HEAD) is, to the server, the same thing as not making it, because safe requests do not change anything. The concept of idempotence means that making many times an idempotent (PUT or DELETE) request to the same URI is the same thing of doing only one idempotent request.

2.8.2 HTTP status code

The HTTP status codes are based on five series (Table 2.3). The most used status codes are: 200, 400, 404, and 500, meaning, respectively, **OK** (success), **Bad Request** (a problem on the client side), **Not Found** (the URI does not exist), and **Internal Server Error**.

Table 2.3: The five HTTP status code series

Serie	**Description**
1xx	Transition
2xx	Success
3xx	Redirection
4xx	Client side error
5xx	Server side error

These codes are even more important in the context of web services/APIs because it is easier for a computer program to handle codes instead of reading a page full of messages about a given error. On the other hand, text messages are preferred over codes when showing an error to humans, because they do not like codes as much as programs do.

2.9 API: zooming in

The miRNA hsa-miR-9-5p found in the section 2.4 was retrieved by making a GET HTTP request to the miRBase API through a web browser (such as Chrome or Firefox). The regular user needs only to paste the link given in section 2.4 in the address bar of a web browser and it redirects to the miRNA Fasta file, which will be displayed on the browser's window. Of course, a person more used to code can access miRBase API by creating a script.

2.10 Conclusion

When using HTTP to create a RESTful API, HTTP methods are available, among them: GET, PUT, POST and DELETE. The API itself can be developed using any programming language that allows responding HTTP requests, for example: JavaScript, PHP and Python. The clients must be able to perform HTTP requests (GET, PUT and the others). Each request may have its own parameters (rules) to be interpreted correctly by the server (API).

Chapter 3

Related Works

The achievements of big data biology require integration of skills across several fields, including mathematics, statistics, and computer sciences. The organization of data, as well as their integration and analysis, demand sophisticated computational methods. These tasks are essential on the way towards comprehension of the molecular, cellular and physiological processes underlying phenotypes such as complex (multifactorial) diseases. That is also the case of the microRNAs and their related information. This chapter will cover some miRNA tools, specially tools4miRs [Lukasik et al., 2016] and the two programmable interfaces: targetHub [Manyam et al., 2013] and miRMaid [Jacobsen et al., 2010].

Lukasik et al. developed and made available a very interesting tool, called tools4miRs, that helps researchers to find out which miRNA tools fit the best to their problems. The authors have made a survey of the scientific literature and gathered over 160 tools that are related to miRNA analysis. Besides gathering the tools, they also manually classified them regarding the methods that each tool applies, creating four categories: **1) sequencing data analysis** (N = 36); **2) all-in-one-tools** (N = 2); **3) Databases** (N = 62); and **4) other tools** (N = 16), and seven more detailed sections: **1) known miRNA identification**; **2) isomiRs identification**; **3) novel miRNA/precursor analysis**; **4) differential expression analysis**; **5) target prediction**; **6) target functional analysis**; and **7) miRSNPs analysis** [Lukasik et al., 2016]. Method classification results are displayed using tables on the tools4miRs web site, which has an user-friendly interface. In addition, tools4miRs allows for the prediction of miRNA targets. However, so far, there is no API or related reported by tools4miRs.

We identified only two miRNA related APIs available: targetHub [Manyam et al., 2013] and miRMaid [Jacobsen et al., 2010]. Even though none of them integrate information about SNPs in miRNA target sites, they will be briefly described.

miRMaid was created by the year of 2009 and it is completely RESTful. It was developed using Ruby and Rails and it uses the miRBase database as its core [Jacobsen et al., 2010]. When one uses the API on the web browser, it is almost the same of navigating the miRBase; the difference comes when considering the plugins that can be used together with the miRMaid API. However, there are only two plugins currently available, one that tracks miRNAs and diseases and another that is related to miRNA cloning. Besides being well designed, the API has not been updated since the year of 2010.

The targetHub web service integrates data of different miRNA-target binding predictions tools, such as: TargetScan [Grimson et al., 2007], PicTar [Krek et al., 2005], and miRanda [Enright et al., 2003]. It was developed using the Apache CouchDB database (*couchdb.apache.org*) which provides storage and a RESTful access programmable interface [Manyam et al., 2013]. The user can then access targetHub data using HTTP requests or using the web interface that is available [Manyam et al., 2013]. Nonetheless, the data integration is made by downloading all the available data or running algorithms over the available miRNAs (miRBase), what means that every time the data is changed on its original source, targetHub programmers have to update it.

To the best of our knowledge, there is no application that addresses miRSNPs, integrating the access of the different online databases that apply different computational techniques and can have therefore different results among them. That being said, the creation of a web service that fulfills this task is needed.

The problem of combining data residing at different sources, and providing the user with a unified view of these data, is known as data integration [Lenzerini, 2002]. Data integration is hard considering that there are differences among the sources, for example, sources may differ on schema, language, implementation, and most importantly, how one can perform queries over their data. There are various data integration architectures, however, the majority of the integration systems uses a composition of warehousing and/or virtual integration architectures [Doan et al., 2012]. The concept of data warehousing is that the data from every source is loaded and materialized into one physical database, which can be used to answer queries over the data. On the other hand, in virtual integration, the data are kept in their original source and accessed at query time [Doan et al., 2012]. The main advantage of using virtual integration over warehousing is that one can query the most recent data from different sources/databases.

Chapter 4

Databases Selection

4.1 Selection

In order to select which databases were going to be integrated, a *Pubmed* search was carefully done using the key words showed in table 4.1. The main idea was to find the manuscripts that describe the databases, as well as to avoid missing relevant ones. The searches were performed on April 6th, 2016, considering the last five years and only title and abstract (TIAB).

Table 4.1: *Pubmed* search results

Search Terms	**Results**
mirna[TIAB] AND polymorphism[TIAB] AND database[TIAB]	14
mirna[TIAB] AND snp[TIAB] AND database[TIAB]	16
microRNA[TIAB] AND snp[TIAB] AND database[TIAB]	12
microRNA[TIAB] AND polymorphism[TIAB] AND database[TIAB]	15
Total	57

As shown in table 4.1, 57 manuscripts were found and, as expected, some of them appeared more than once. The replicates were merged summing 37 manuscripts. The abstracts were read to extract the following information: 1) objective; 2) type of study; 3) whether an actual application was developed; 4) species considered in the study; 5) whether the study actually considered SNPs in miRNA target sites (miRSNPs); 6) whether there is an online available database, and if so 7) how the database is available (search, browse and download were taken into account). Firstly, considering criteria 1 to 3, 23 additional manuscripts were discarded, because they did not report database creation or were not written in English (Figure 4.1). Finally, considering the criteria 4, 5, 6, and 7, the final manuscript count was 5 (Table 4.2).

Table 4.2: Final filtering from *Pubmed* search

Name	**URL**	**Year**	**Reference**
miRNASNP2	http://bioinfo.life.hust.edu.cn/miRNASNP2/	2015	[Gong et al., 2015]
miRNASNP	http://www.bioguo.org/miRNASNP/	2012	[Gong et al., 2012]
mirSNP	http://cmbi.bjmu.edu.cn/mirsnp	2012	[Liu et al., 2012]
mirsnpscore	http://www.bigr.medisin.ntnu.no/mirsnpscore/	2011	[Thomas et al., 2011]
PolymiRTS 2.0	http://compbio.uthsc.edu/miRSNP/	2012	[Ziebarth et al., 2012]

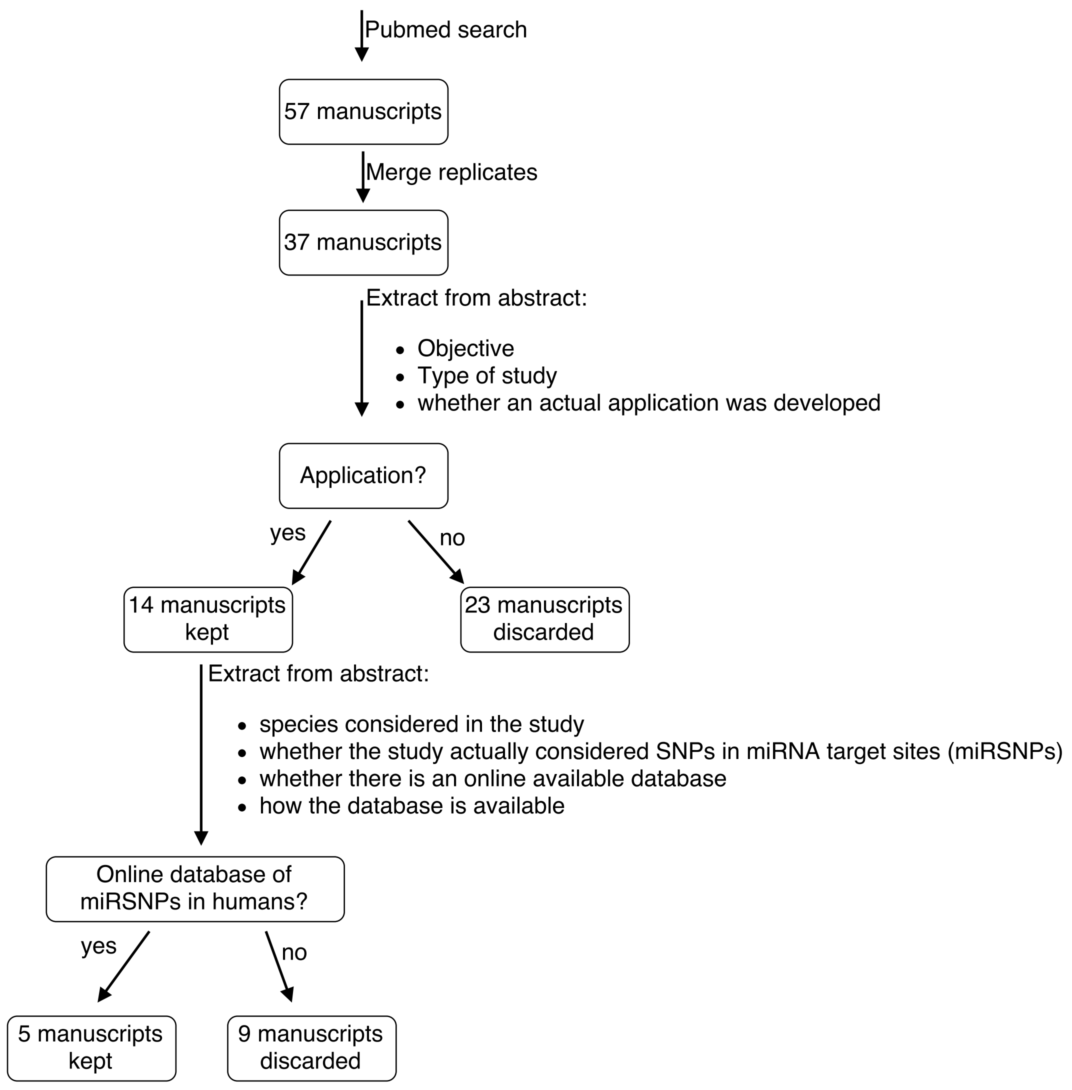

Pubmed search
57 manuscripts
Merge replicates
37 manuscripts
Extract from abstract:
Objective
Type of study
whether an actual application was developed
Application?
yes
no
14 manuscripts kept
23 manuscripts discarded
Extract from abstract:
species considered in the study
whether the study actually considered SNPs in miRNA target sites (miRSNPs)
whether there is an online available database
how the database is available
Online database of miRSNPs in humans?
yes
no
5 manuscripts kept
9 manuscripts discarded

Figure 4.1: MiRSNPs database selection workflow.

One of the manuscripts (miRNASNP2) described an update to a database that was previously reported in another manuscript (miRNASNP). In addition, the PolymiRTS website reported a new version of the database (PolymiRTS 3.0) which was used instead of the previous one (PolymiRTS 2.0).

In addition to the databases found through the *Pubmed* search, miRSNP databases enumerated in a recent review [Hrdlickova et al., 2014] were considered. One different database [Bruno et al., 2012] was included, summing five databases to work with (Table 4.3).

Table 4.3: Selected miRSNPs databases

Name	URL	Year	Reference
miRNASNP2	http://bioinfo.life.hust.edu.cn/miRNASNP2/	2015	[Gong et al., 2015]
mirSNP	http://cmbi.bjmu.edu.cn/mirsnp	2012	[Liu et al., 2012]
mirsnpscore	http://www.bigr.medisin.ntnu.no/mirsnpscore/	2011	[Thomas et al., 2011]
PolymiRTS 3.0	http://compbio.uthsc.edu/miRSNP/	2014	[Bhattacharya et al., 2014]
mirdSNP	http://mirdsnp.ccr.buffalo.edu/index.php	2012	[Bruno et al., 2012]

Chapter 5

Online Search

In order to integrate the most recent available data of SNPs in miRNA target sites (miRSNPs), it is a better approach to query the web applications of interest each time the data is needed, instead of downloading and storing them. Therefore, this online up-to-date approach was used in this project. All data access and data parsing scripts were written in *Python* language using multiple libraries (Table 5.1). For each selected database, at least one HTTP request is performed. Considering the similarity between all requests, one of the ways to build a GET and POST HTTP request in Python is shown in the listing 5.1.

Listing 5.1: Example of GET and POST HTTP requests in Python.

```python
import requests

endpoint = 'server/url'
payload = {'gene': 'a gene id'}

# Here the GET request is done
response = requests.get(endpoint, params=payload, timeout=3)

# Do something with response...

# And now a POST request
response = requests.post(endpoint, data=payload, timeout=3)

# Note that when making POST requets, the parameters are assigned to data
# (data=payload) instead of params (params=payload)
```

Even though the way all the requests are build is similar, it is important to highlight that each server responds a different endpoint and expects specific parameters (payload) to understand what it should do to respond the client.

The source code of each selected online database (miRNASNP2, mirSNP, mirsnpscore, PolymiRTS, and mirdSNP) was read/analyzed to figure out which and how server URLs (Uniform Resource Locator) are called from each application in order to retrieve its own miRSNPs data (Table 5.2). Chrome Developer Tools were also applied to check network requests and HTML (Hypertext Markup Language) elements. Table 5.2 shows the server endpoints that are called when making HTTP requests, and also which HTTP method is expected for each endpoint/URL. In the following sections, the steps to retrieve data from each of the web applications and how the data was parsed are described.

Table 5.1: Python Packages Used in the Scripts

Name	Brief Description
argparse	Command-line parser
sys	Provides information about functions and variables of the Python interpreter
beautifulsoup4	HTML and XML parser
csv	CSV file reading and writing
urllib	Package for openig URLs
urllib2	Package for openig URLs
requests	Package to make HTTP requests
re	Regex handling
html.parser	HTML Parser
html5lib	Handles HTML5 (parsing and so on)

Table 5.2: MiRSNPs database endpoints

Name	Endpoints	HTTP Request
miRNASNP2	http://bioinfo.life.hust.edu.cn/miRNASNP2 http://bioinfo.life.hust.edu.cn/miRNASNP2/geneTargets.php	GET
mirSNP	http://bioinfo.bjmu.edu.cn/mirsnp/search/SingleSearch	GET
mirsnpscore	http://www.bigr.medisin.ntnu.no/mirsnpscore/search2.php	POST
PolymiRTS 3.0	http://compbio.uthsc.edu/miRSNP/search.php http://compbio.uthsc.edu/miRSNP/miRSNP_detail_all.php	POST
mirdSNP	http://mirdsnp.ccr.buffalo.edu/data/search/csv.php	GET

5.1 miRNASNP2

The miRNASNP2 miRSNPs data is split in two categories: gain and loss. The user can search by gene, miRNA or SNP. For each category, one must firstly make a GET HTTP request (Table 5.3) in order to retrieve and parse an HTML page that contains at its source code a link, identified by the word "downloading", to download the complete searched data as text file by making another GET request.

A single report from a downloaded text file when searching for *miRNA = hsa-miR-155-5p* is shown in listing 5.2. Text parsing is done by simply removing the blank lines (2nd and 4th lines in the snippet) and combining each six (1st, 3rd, 5th, 6th, 7th and 8th lines in the snippet) of the remaining ones. The resulting combined lines are then translated into JSON (JavaScript Object Notation).

Listing 5.2: Example of miRNASNP2 report.

```
1 104976 KIAA1751   NM_001080484 rs3795288  chr1:1886888 hsa-miR-155-5p
     -16.40 -16.80
2
3            rs3795288: C --> U
4
5   miRNA:  3' uggggauaGUGCUAAUCGUAAUu 5'
6                   |||   ||||||Y
7   UTR:    5' cttgaaggCACCTCCAGCATTSg 3'
8       7.45  8994  567.46 138  NULL  NULL
```

Table 5.3: miRNASNP2 parameters for *GET* request

Param	Value
gene	any gene id
mirna	any miRNA id
snp	any SNP id
gainorlost	gain or loss

5.2 mirSNP

The mirSNP miRSNPs data can be retrieved by making a single GET request (Table 5.4) that returns an HTML page. The parsing is done by firstly getting a table containing a CSS (Cascading Style Sheets) class named "results" using the Beautiful Soup library. The second step is done by translating the table from HTML into JSON.

Table 5.4: mirSNP parameters for *GET* request

Param	Value	Description
gene	any gene id	-
mirna	any miRNA id	-
snp	any SNP id	-
what	rgene, rmir or rsnp	what to search for (gene, mirna or snp)

5.3 mirsnpscore

The mirsnpscore miRSNPs data can be retrieved by making a single POST request (Table 5.5) that returns an HTML page. The parsing is done by firstly getting a table containing a CSS class named "mybox" using the Beautiful Soup library. The second step is done by translating the table from HTML into JSON.

Table 5.5: mirsnpscore parameters for *POST* request

Param	Value	Description
gene	any gene id	-
mirna	any miRNA id	-
snp	any SNP id	-
limited	yes or no	whether to consider or not the ’limit’ key
limit	any number	number of results to show after searching
what	rgene, rmir or rsnp	what to search for (gene, mirna or snp)
searchMult	Search	-

5.4 PolymiRTS

The PolymiRTS miRSNPs data can be retrieved by firstly making a POST request (Table 5.6) that returns an HTML page that contains a list of gene transcripts which are related to the search key (gene id, SNP id, or miRNA id). Using Beautiful Soup it is possible to extract from the HTML page the transcript id, gene name, gene description, and a link to get the detailed data related to the transcript itself. Finally, making a new POST request (Table 5.7) for each transcript link found in the list, a new HTML page that contains all the detailed data about the referred transcript can be retrieved. Each HTML page is parsed using Beautiful Soup, the first step is to select all tables from the page, because they have no unique attributes (like class or id). Then, from the tables with the captions 1) "SNPs and INDELs in miRNA target sites from CLASH data", and 2) "SNPs and INDELs in miRNA target sites", translate data into JSON.

Table 5.6: PolymiRTS parameters for 1st *POST* request

Param	Value	Description
my_run	1	-
my_page	1	-
phenoid	”	-
my_org	Human	-
Fclass[]	[’C’, ’D’, ’N’, ’O’]	-
Support	0	-
Evicode[]	[’CLASH’, ’LT’, ’LTL’, ’HT’, ’HTL’, ’N’]	-
my_snpid	any snp id	-
search_method	”	-
my_mirid	any miRNA id	-
my_geneid	any gene id	-
my_genedesc	”	-
my_phenodesc	”	-
my_GOAccession	”	-
Search	1, 2 or 3	what to search for (snp, mirna or gene)

Table 5.7: PolymiRTS parameters for 2nd *POST* request

Param	Value
Transcipt_id	any transcript id
species	Human

5.5 mirdSNP

The mirdSNP miRSNPs data can be retrieved by making a single GET request (Table 5.8) that returns a CSV file. Parsing was done by using the Python CSV reader and translating it into JSON.

Table 5.8: mirdSNP Parameters for *GET* Request

Param	Value
sSearch_0	”
sSearch_1	any gene id
sSearch_3	any miRNA id
sSearch_4	any SNP id
sSearch_5	”
filter_flank	”
iSortCol_0	6
sSortDir_0	asc

5.6 Conclusion

Considering the need of analyzing the source code of the web pages of each database in order to access their data, it is even more clear that a web API is needed to integrate the miRSNPs database data access.

Chapter 6

miRSNaPi: an Application Programming Interface

Named **miRSNaPi** (http://lgmh.c3sl.ufpr.br/miRSNaPi), our API is capable of integrating the access to the online data of five different miRSNPs databases and it can be used to create new applications based on the integrated data. An overview of miRSNaPi is shown in the figure 6.1. Details about the API implementation, how it works, and how it should be used will be presented in this chapter.

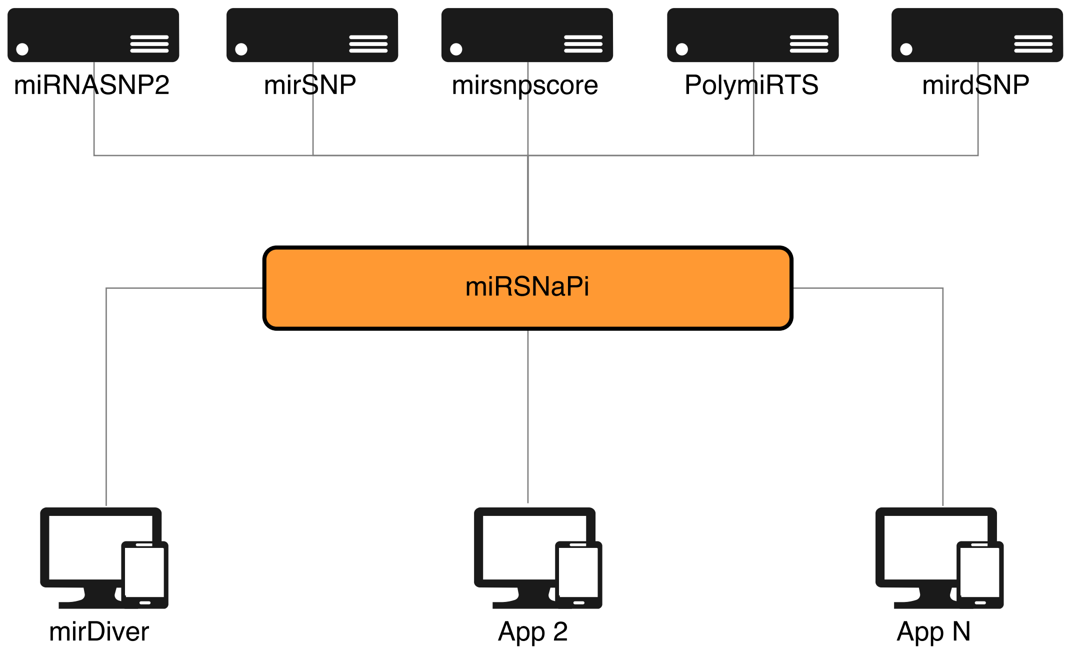


Figure 6.1: miRSNaPi overview.

6.1 Introduction

APIs are web services that are just like any computer system, the main difference being that web services are run over the Internet. Like any other system, an API must be planed and developed in accord to the software engineering processes and concepts. In addition, as miRSNaPi was intended to be RESTful, a RESTful architecture was followed: the Resource-

Oriented Architecture (ROA) that was described in chapter 2. The next sections contain information about the problem, its solution, requirements, use cases, design, and more.

6.2 A real world problem

The explosion of biological data together with the computational advances, data storage capacity growth, and new ways of making data available, allow for the creation of different tools and databases that cover various scientific areas. That is also the case of miRNAs and their related information, specifically their genetic variation information. Considering the context of SNPs in miRNA target sites (miRSNPs), there are plenty of databases full of predictions about the effect of miRSNPs. It is a really common task for a researcher who works with miRNAs/miRSNPs to search over miRSNPs databases looking for frequently reported effects. This task can be really tiring because of the different ways in which each database shows its results, meaning that the researcher has to combine the data manually using spreadsheets, for example, under the risk of missing something or manually processing something wrong. To the best of our knowledge, there is no application that integrates the access to miRSNPs data spread out in different databases and that is why the development of a web service that fulfills this task is needed.

6.3 Requirements

6.3.1 User requirements

The user requirements were gathered by watching and talking to co-workers who actually use miRSNPs databases. The requirements agree with the aims of the project. That being said, the user requirements are:

- Simultaneously search data of different online databases that contain information about the effect of SNPs in miRNA target sites.
- Search data by gene, miRNA or SNP
- Be able to use the integrated data in many different ways.

6.3.2 Product requirements

Considering the user requirements, the solution was designed to be a web service, a RESTful API, that provides the simultaneous search of data from different databases given a gene name, miRNA name or SNP identifier. In order to provide the possibility of using the integrated data access in different ways, the data is served in JSON format.

6.4 Use cases

6.4.1 Actors

As an API can be seen as a user interface for a developer, the actor could be the developer herself and her applications. As the applications themselves are the ones that will be communicating with the API, the only actor will be named "External Application" (EA).

6.4.2 List of use cases

Considering that the external application will be able to perform the requests: **1) request data by gene**; **2) request data by miRNA** and **3) request data by SNP**, it is possible to write these requests as a single use case: **make a request** (Figure 6.2).

6.4.3 Use case diagrams

Make a request

Make a request
External Application

Figure 6.2: Use case: make a request.
The use case description can be found in the table 6.1.

6.4.4 Use cases

The use case **make a request** is detailed in the table 6.1.

6.5 Other diagrams

6.5.1 System sequence diagram

The system sequence diagram (SSD) helps to understand the order of the events and the interactions of the system that is being developed. In figure 6.3, the SSD for **make a request** use case is shown.

6.6 Design overview

6.6.1 Data set definition

The data served by the API comes from different web applications, this means that miRSNaPi resources are the combination of the results of the scripts (see chapter 5) that access the data from the other web applications.

6.6.2 Data set split and URI

The data was split in three different URIs, depending on what the external applications is looking for: by gene, by miRNA or by SNP (Table 6.2). This division is strongly related to the user's requirements. Following ROA concepts, the URI names are very well related with their resources. POST method was chosen rather than GET because the user must send objects to the server (an array of identifiers for each database that miRSNaPi has to look for data).

Table 6.1: Use case **make a request** detailed

Use case name: Make a request	**ID:** UC-0	**Prioroty:** Very high
Primary actor: External Application	**Level:** User	
Goal: Request data by a gene name, miRNa name or SNP identifier		
Precondition: External application is being/was developed by someone		
Trigger: External application receives the searched data		
Typical flow of events: 1. User requests the data passing expected parameters 2. System receives the request and reads the parameters 3. System searches for the data 4. System builds the and sends the response 5. User receives the response that contains the searched data		
Alternative flow of events: 1. User passes wrong parameters 3. Parameters are incorrect, system does not perform the search 4. Error while parsing parameters, system sends an error code and message 4. Error while searching data, system sends an error code and message 4. Error while parsing data, system sends an error code and message 5. User receives the response that contains the error code and message		

Table 6.2: Data set split, URI and HTTP method

Split	URI	HTTP method	Parameters
By gene	miRSNaPi/query/gene	POST	databases, gene name
By miRNA	miRSNaPi/query/mirna	POST	databases, miRNA name
By SNP	miRSNaPi/query/snp	POST	databases, SNP identifier

6.6.3 Resource representation

The resources are represented by the data itself and some additional information (Table 6.3). In addition, JSON is used as the representation format because the data is structured. When smart programmers create HTTP requests, they normally use wrappers/libraries, so they do not have to write many times the same code. In addition, libraries offer an abstraction in the sense of HTTP headers (language, cache-control, content-encoding and so on) that are also part of the resource.

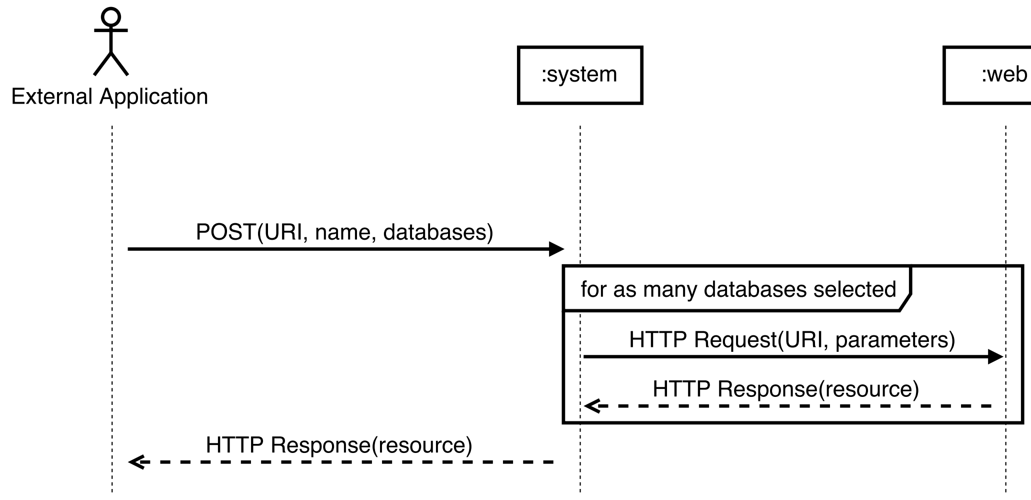

External Application
:system
:web
POST(URI, name, databases)
for as many databases selected
HTTP Request(URI, parameters)
HTTP Response(resource)
HTTP Response(resource)

Figure 6.3: System sequence diagram for **make a request** use case.
When one (external application) makes a request targeting an API URI, one sends within the request two parameters: *1) name* for a gene name, a miRNA name, or a SNP identifier and *2) databases* representing the databases of interest. For as many databases selected, the API will request data and receive a response. After combining the answers, the API will respond the external application sending the data (resource).

Table 6.3: Additional information about the resource

Key	**Value**
runtime	Time to fetch and parse all the data
gene name/mirna name/snp rs	Gene name/miRNA name/SNP identifier
databases	The 'databases' array sent by the client

6.6.4 How it should work

The API responds for three URI (Table 6.2) and the *databases* (Table 6.4) parameter is an array that must be sent in the body HTTP POST request together with the search name that agrees with the URI being requested (gene, miRNA, or SNP). These database identifiers kind of add a complexity to the requests, but that is a way of providing the external application the possibility of requesting only the data from databases of their own interest.

Table 6.4: Database identifiers

Database	**Identifier**
miRNASNP2	miRNASNP2
mirSNP	mirSNP
mirsnpscore	mirsnpscore
PolymiRTS	PolymiRTS
mirdSNP	mirdSNP

Once the external application calls the URI, using the right method and passing the correct parameters, the API tries to collect the data from the desired databases. If data fetching and parsing are successful, the API finally sends the data over HTTP (status code 200).

External application (our user) could send the requests using one of the many libraries available that handle HTTP requests. These libraries are really useful considering that they hide the complexity of creating an actual HTTP request, they are wrappers. Listing 6.1 shows how to create an AJAX HTTP request that can be used to fetch data from our API (by gene).

Listing 6.1: Example of AJAX HTTP request.

```
var gene = 'TNFSF13B',
  selected_bds = ['miRNASNP2', 'mirsnpscore', 'mirSNP', 'PolymiRTS', '
      mirdSNP'];

ajax({
      url : 'miRSNaPi/query/gene',
      type : 'POST',
      cache : 'true',
      data: {'gene' : gene,
      'selected_bds' : selected_bds},
      success: function (data, textStatus, jqXHR) {
            // do something
      },
      error: function (jqXHR, textStatus, errorThrown) {
            // handle error
      }
});
```

The response sent by the API is shown in listing 6.2. The data for each database is hidden because it is too large to be pasted here (Listing 6.2).

Listing 6.2: Example of API response

```
{
  "runtime": "13.303421 s",
  "gene id": "TNFSF13B",
  "databases": [
    "miRNASNP2",
    "mirSNP",
    "mirsnpscore",
    "PolymiRTS",
    "mirdSNP"
  ],
  "miRNASNP2": [...],
  "mirSNP": [...],
  "mirsnpscore": [...],
  "PolymiRTS": [...],
  "mirdSNP": [...]
}
```

6.6.5 What if something went wrong

In case an error occurs, the API responds the client with an HTTP error status code and a message, such as: 500 (internal error), 404 (URI not found) or 400 (bad request) . As web services are dedicated to programs/applications, sending different status codes is a good approach because it is easier for programs (that are not humans!) to understand a status code rather than a complete detailed error message sent using HTTP status code = 200.

6.7 Implementation details

miRSNaPi was implemented using the Node.js (Node) that is an asynchronous event driven based on JavaScript language. Node is meant to build scalable network applications and it is normally used together with npm to better control JavaScript packages that are used in the project. Our API uses many npm packages (Table 6.5) and one of the most important is the *express* package that is used to route and handle HTTP requests. To create an express application, one can install (Table 6.5) and call the express-generator:

```
express miRSNaPi
```

Table 6.5: npm packages and their installation commands

npm package	Installation
express-generator	npm install express-generator -g
body-parser	npm install body-parser –save
cookie-parser	npm install cookie-parser –save
debug	npm install debug –save
download-file	npm install download-file–save
express	npm install express –save
file-exists	npm install file-exists –save
performance-now	npm install performance-now –save
pug	npm install pug –save
python-shell	npm install python-shell –save
shelljs	npm install shelljs –save

The express-generator will create files (Figure 6.4) that are helpful in the sense of avoiding the need of writing every time the same code when creating a new application. In figure 6.4 the express-generator directory structure is shown. To this date, files in the view directory were created by the generator in the *jade* format, but the jade package was renamed *pug*, so the files need to stick to the *pug* format to be handled by the *pug* package.

In the package.json file (Figure 6.4) the list of packages needed by the application (JSON format) is placed. One installs all packages by calling:

```
npm install
```

Once all dependencies are installed, the server can be started (for development) calling:

```
npm start
```

After this, one can start giving resources their URIs and programming request handling. In the following snippet, there is an example of how to define a name (URI) and route a resource using express.router(). A highlight on the use of post, our very own HTTP POST.

```
router.post('/query', function (req, res) {
      var resource = 'something interesting';
      res.status(200).send(resource);
});
```

When an external application makes a request for the miRSNaPi API, it will search for the requested data in every database in the *databases* array (Table 6.4) by using the Python

```
miRSNaPi
├── app.js
├── bin
│   └── www
├── package.json
├── public
│   ├── images
│   ├── javascripts
│   └── stylesheets
│       └── style.css
├── routes
│   ├── index.js
│   └── users.js
└── views
    ├── error.pug
    ├── index.pug
    └── layout.pug
```

Figure 6.4: Express-generator directory structure.

scripts described in chapter 5. That being said, and considering the API description presented in this chapter, a final overview of our solution is shown in figure 6.5.

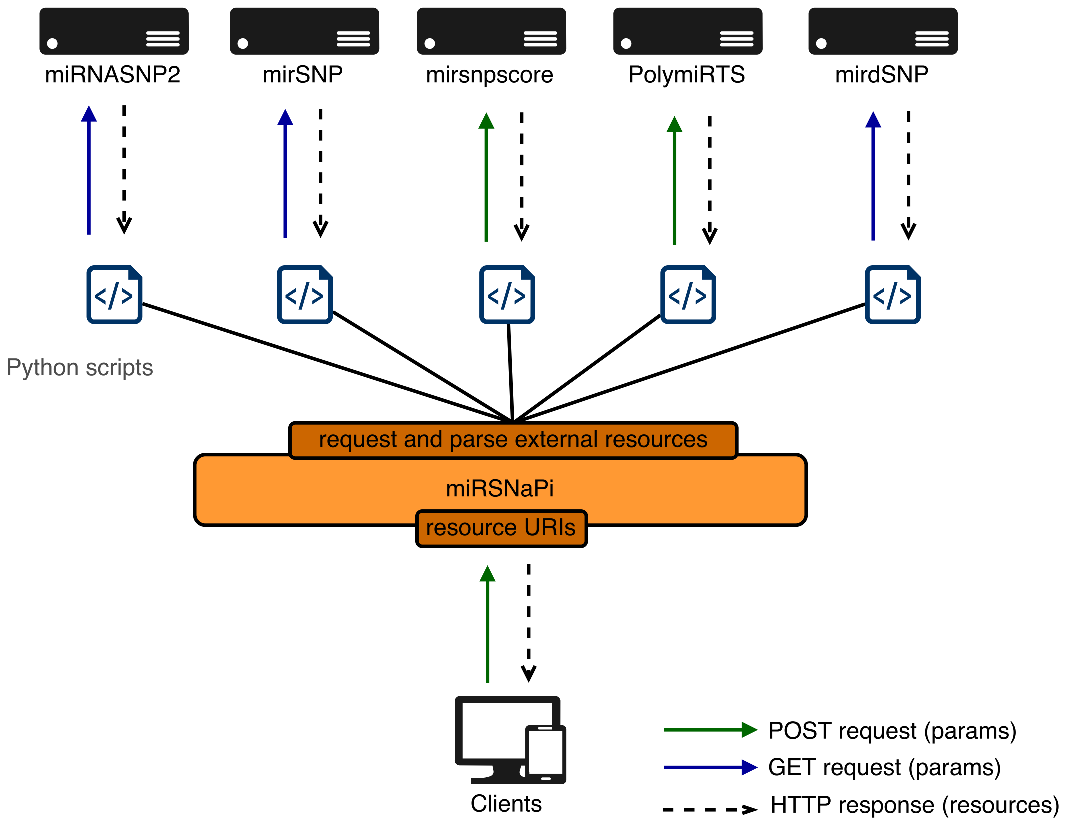


Figure 6.5: miRSNaPi final overview.

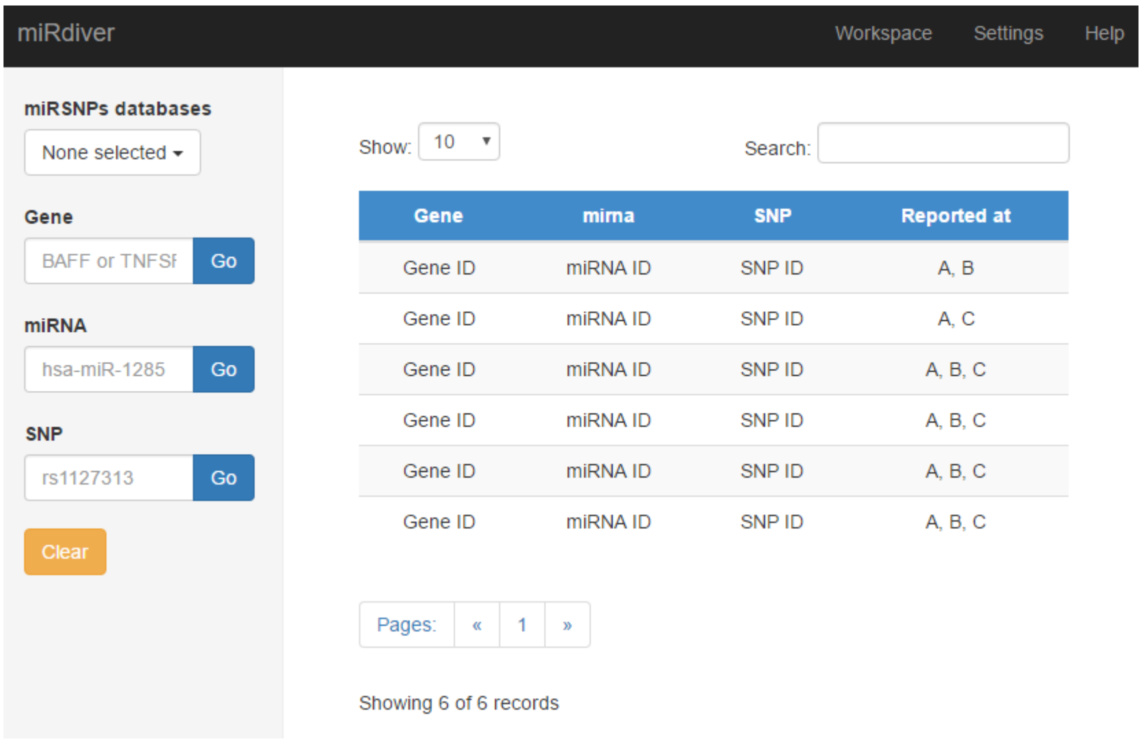
6.8 Conclusion

The miRSNaPi API is a RESTful API for providing addressability, statelessness, and a uniform interface by using HTTP methods. In addition, the API agrees with the user requirements by serving the miRSNPs data spread out in different online databases in a structured format allowing for the development of various external applications based on this data.

Chapter 7

miRdiver: consistent genetic diversity at miRNA binding sites

Our very own miRSNaPi API can effectively be used to address the real world problem described in the chapter 6. Here we present the miRdiver (Figure 7.1) web interface that requests the miRSNaPi data and graphically shows it to the user, who is finally a human being.



Gene	mirna	SNP	Reported at
Gene ID	miRNA ID	SNP ID	A, B
Gene ID	miRNA ID	SNP ID	A, C
Gene ID	miRNA ID	SNP ID	A, B, C
Gene ID	miRNA ID	SNP ID	A, B, C
Gene ID	miRNA ID	SNP ID	A, B, C
Gene ID	miRNA ID	SNP ID	A, B, C

Showing 6 of 6 records

Figure 7.1: Screenshot of the miRdiver homepage.
(http://lgmh.c3sl.ufpr.br/miRdiver)

7.1 Requirements

7.1.1 User requirements

As well as miRSNaPi, the user requirements were gathered by watching and talking to co-workers, but this time requirements are a little bit different, because they are related to an user interface, which is quite different from an API on the matter of usage. That being said, the user requirements are:

- To find frequently reported miRSNPs searching by gene, miRNA or SNP on different miRSNPs databases.
- To see each 3-tuple (gene, miRNA, SNP) being combined with the databases where the referred 3-tuple is reported.

These two requirements are very similar, because by frequently reported miRSNPs, the user means miRSNPs that appear in different databases.

7.1.2 Product requirements

Considering the user requirements, the solution was designed to be a **web user interface** that allows the **search by gene**, **miRNA**, or **SNP** and shows the miRSNPs databases in which each 3-tuple (gene, miRNA, SNP) was reported. A good way to show this combination (gene, miRNA, SNP, databases) is a **table**, because it is simple and fits well to the problem. Furthermore, the user interface must be **responsive**, what means that the interface layout must be adapted for different screen resolutions (mobiles, personal computers, tablets and so on).

7.2 Use cases

7.2.1 Actor

Once again, there is only one actor (a human one!) named **researcher**.

7.2.2 List of use cases

As the researcher wants to simultaneously access the different miRSNPs databases, by gene, miRNA, or SNP, there are three use cases:

- Search by gene
- Search by miRNA
- Search by SNP

7.2.3 Use case diagrams

Use case diagram for the researcher actor is shown in figure 7.2.

7.2.4 Use cases

The use cases **search by gene**, **search by miRNA**, and **search by SNP** are described in tables 7.1, 7.2 , and 7.3, respectively.

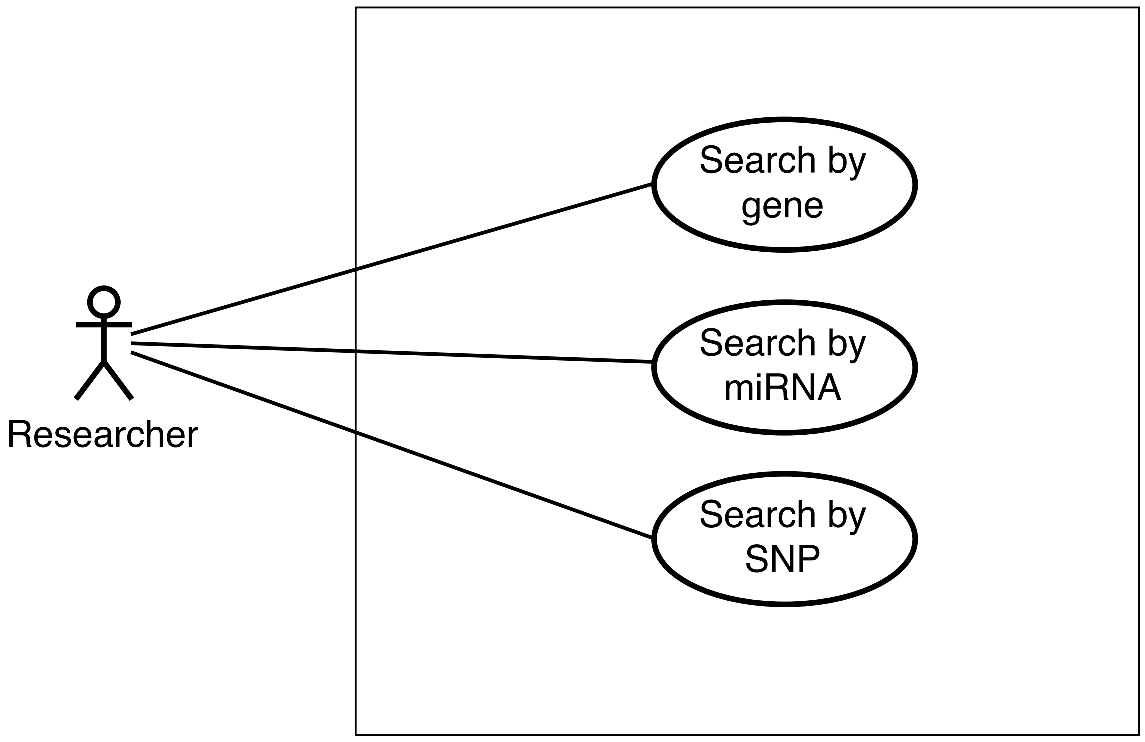


Figure 7.2: Use cases for the **researcher** actor.
The use cases **search by gene**, **search by miRNA**, and **search by SNP** descriptions can be found in the tables 7.1, 7.2, and 7.3, respectively.

7.3 Other diagrams

7.3.1 System sequence diagram

System sequence diagrams (SSD) for each use case are shown in the figures 7.3, 7.4, and 7.5. The SSD diagram shows that the miRSNPs data is fetched through the miRSNaPi API.

7.4 Implementation details

As miRdiver is a web application, it was also developed using Node and npm (see chapter 6). Nevertheless, considering miRdiver also is an user web interface, bower package manager was used in order to control the client (user interface) packages. Bower is also part of the Node environment and can be installed as follows:

```
npm install -g bower
```

Bower packages are also placed in a JSON file and can be installed using:

```
bower install
```

The list of npm packages is shown in table 7.4 and the list of bower packages can be found in the table 7.5. In addition, HTML (HyperText Markup Language), CSS (Cascading Style Sheets), and JavaScript were used to develop the user interface. Bootstrap related packages were used to make the interface responsive.

Table 7.1: Use case **search by gene**

Use case name: Search by gene	ID: UC-0	Prioriota: Very high
Primary actor: Researcher	Level: User	
Goal: To search miRSNPs data from different online databases by a gene name		
Precondition: None		
Trigger: User can see a table that contains the reported data		
Typical flow of events: 1. User selects the miRSNPs databases of interest at the multiselect element 2. User types a gene name in the gene name field and clicks over the GO button 3. System checks if the user selected any database and typed something at the gene name field 4. System requests the data to an external application and parses it 5. System shows the table of results		
Alternative flow of events: 1. User does not select any database 2. User does not type anything in the gene name field 3. The system understands that the user did not type/select valid information 4. The system does not request any data 5. The system shows an error message		

Table 7.2: Use case **search by miRNA**

Use case name: Search by miRNA	ID: UC-1	Prioriоty: Very high
Primary actor: Researcher	Level: User	
Goal: To search miRSNPs data from different online databases by a miRNA name		
Precondition: None		
Trigger: User can see a table that contains the reported data		
Typical flow of events: 1. User selects the miRSNPs databases of interest at the multiselect element 2. User types a miRNA name in the miRNA name field and clicks over the GO button 3. System checks if the user selected any database and typed something at the miRNA name field 4. System requests the data to an external application and parses it 5. System shows the table of results		
Alternative flow of events: 1. User does not select any database 2. User does not type anything in the miRNA name field 3. The system understands that the user did not type/select valid information 4. The system does not request any data 5. The system shows an error message		

Table 7.3: Use case **search by SNP**

Use case name: Search by SNP	ID: UC-2	Prioroty: Very high
Primary actor: Researcher	Level: User	
Goal: To search miRSNPs data from different online databases by a SNP name		
Precondition: None		
Trigger: User can see a table that contains the reported data		
Typical flow of events: 1. User selects the miRSNPs databases of interest at the multiselect element 2. User types a SNP identifier in the SNP identifier field and clicks over the GO button 3. System checks if the user selected any database and typed something at the SNP identifier field 4. System requests the data to an external application and parses it 5. System shows the table of results		
Alternative flow of events: 1. User does not select any database 2. User does not type anything in the SNP identifier field 3. The system understands that the user did not type/select valid information 4. The system does not request any data 5. The system shows an error message		

Table 7.4: npm packages used in miRdiver application

npm package	**Installation**
express-generator	npm install express-generator -g
body-parser	npm install body-parser –save
cookie-parser	npm install cookie-parser –save
debug	npm install debug –save
express	npm install express –save
pug	npm install pug –save
cli-color	npm install cli-color –save

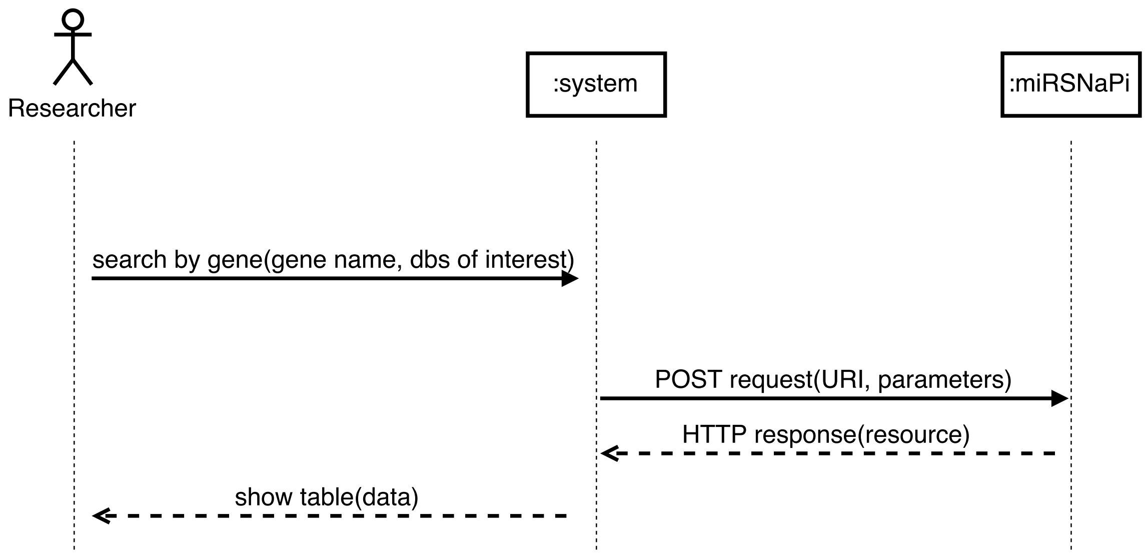

Researcher
:system
:miRSNaPi
search by gene(gene name, dbs of interest)
POST request(URI, parameters)
HTTP response(resource)
show table(data)

Figure 7.3: System sequence diagram for **search by gene** use case.
The user provides the system with gene name and the databases (dbs) of interest. miRdiver will request data to the miRSNaPi API, which will respond with the combined data from the selected databases. miRdiver parses the response and shows a table that contains gene-miRNA-SNP relationships and where (original databases) they were reported.

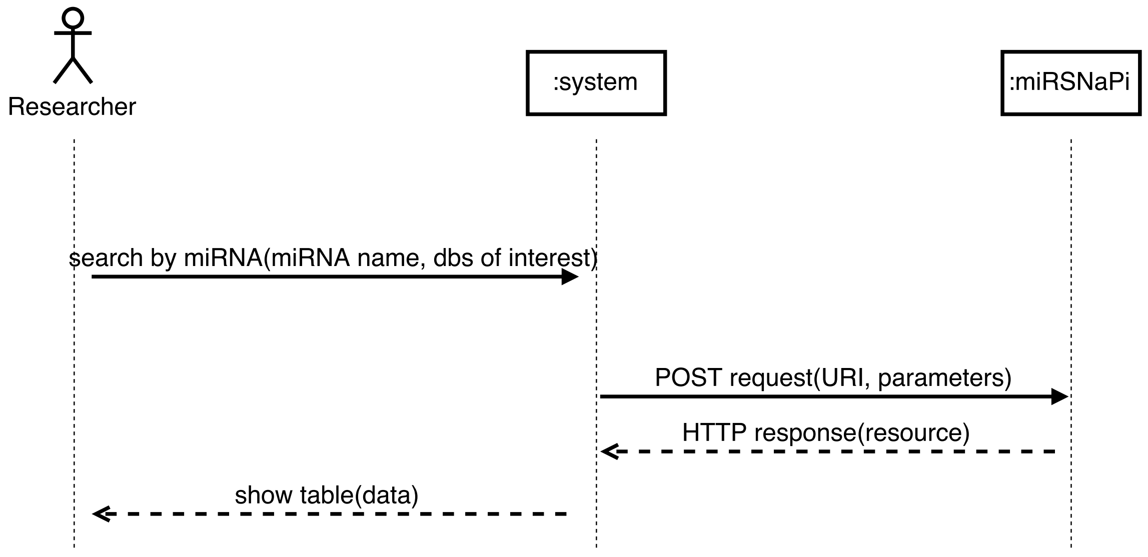

Researcher
:system
:miRSNaPi
search by miRNA(miRNA name, dbs of interest)
POST request(URI, parameters)
HTTP response(resource)
show table(data)

Figure 7.4: System sequence diagram for **search by miRNA** use case.
The user provides the system with miRNA name and the databases (dbs) of interest. miRdiver will request data to the miRSNaPi API, which will respond with the combined data from the selected databases. miRdiver parses the response and shows a table that contains gene-miRNA-SNP relationships and where (original databases) they were reported.

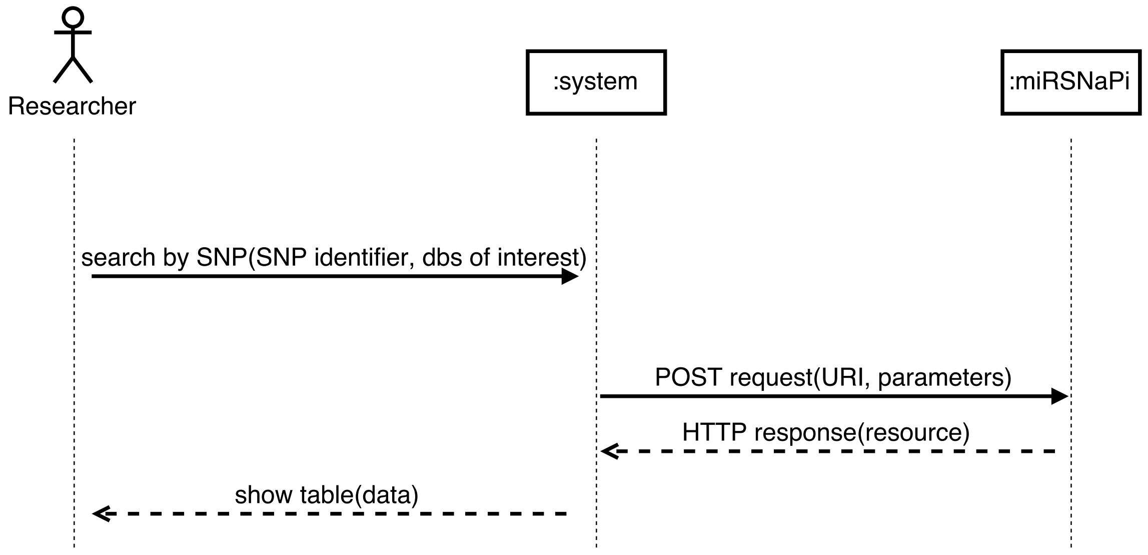

Researcher
:system
:miRSNaPi
search by SNP(SNP identifier, dbs of interest)
POST request(URI, parameters)
HTTP response(resource)
show table(data)

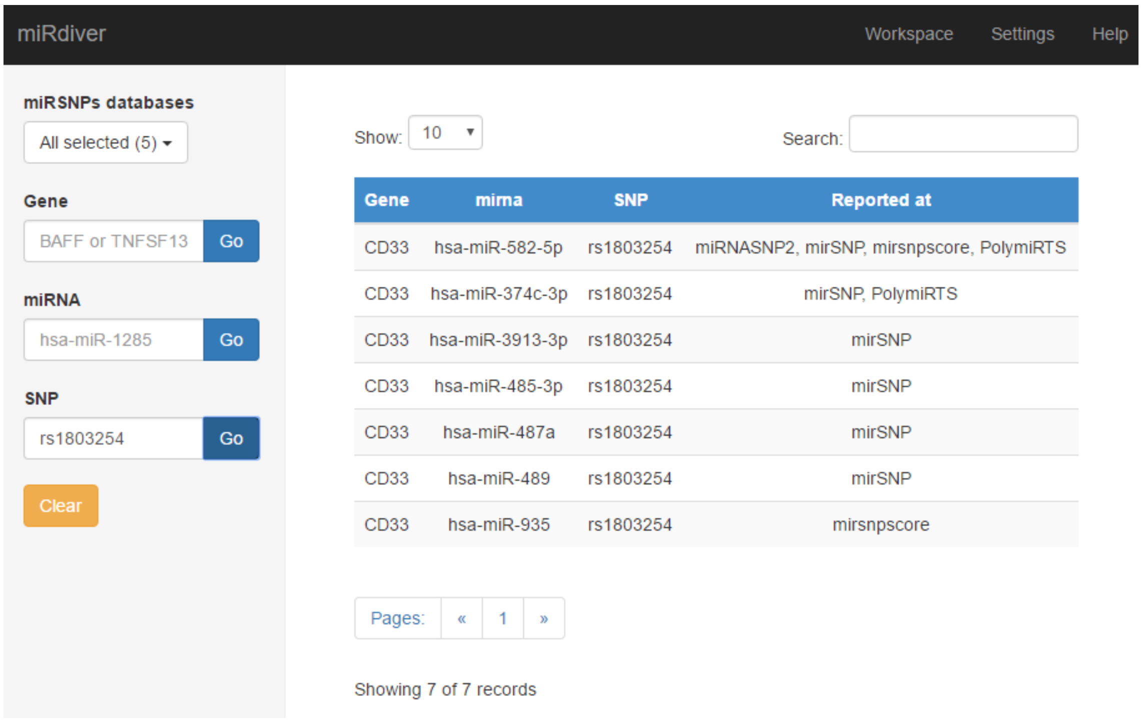
Figure 7.5: System sequence diagram for **search by SNP** use case.
The user provides the system with SNP identifier and the databases (dbs) of interest. miRdiver will request data to the miRSNaPi API, which will respond with the combined data from the selected databases. miRdiver parses the response and shows a table that contains gene-miRNA-SNP relationships and where (original databases) they were reported.

Table 7.5: Packages used on the client side

Package	**Installation**
bootstrap	bower install bootstrap –save
jquery	bower install jquery–save
font-awesome	bower install font-awesome –save
bootstrap-multiselect	bower install bootstrap-multiselect –save
dynatable	./vendor/jquery-dynatable-0.3.1
spin.js	./vendor/spin.js

7.5 rs1803254: a frequently reported miRSNP

In figure 7.6, a miRdiver screenshot, generated when a search by the SNP rs1803254 was performed, is shown. It is possible to see that this SNP considering the miRNA hsa-miR-582-5p is reported by four of the five miRSNPs databases, making rs1803254/hsa-miR-582-5p a frequently reported miRSNP.



Gene	mirna	SNP	Reported at
CD33	hsa-miR-582-5p	rs1803254	miRNASNP2, mirSNP, mirsnpscore, PolymiRTS
CD33	hsa-miR-374c-3p	rs1803254	mirSNP, PolymiRTS
CD33	hsa-miR-3913-3p	rs1803254	mirSNP
CD33	hsa-miR-485-3p	rs1803254	mirSNP
CD33	hsa-miR-487a	rs1803254	mirSNP
CD33	hsa-miR-489	rs1803254	mirSNP
CD33	hsa-miR-935	rs1803254	mirsnpscore

Showing 7 of 7 records

Figure 7.6: Screenshot of the search by the rs1803254 SNP.

7.6 Conclusion

The miRdiver application is responsive and shows miRSNPs data very well. Furthermore, it makes use of the miRSNaPi API like a charm.

Chapter 8

Concluding remarks

miRSNaPi (http://lgmh.c3sl.ufpr.br/miRSNaPi) is a RESTful API that is capable of integrating the access to data of five different online miRSNPs databases (miRNASNP2, mirSNP, mirsnpscore, PolymiRTS, and mirdSNP). It is RESTful because it provides addressability, statelessness, and a uniform interface that uses HTTP methods to receive and respond requests. Furthermore, the API uses JSON as its data format, allowing its clients to use the integrated data in various forms. An example is the miRdiver (http://lgmh.c3sl.ufpr.br/miRdiver) web client that generates a table showing gene-miRNA-SNP relationships along with the respective databases in which they appear. Therefore, miRSNaPi is a very relevant data access tool that allows testing hypothesis more easily, performing investigations, and developing new applications.

8.1 Further work

8.1.1 miRSNaPi

miRSNaPi works very well and can, additionally, be improved as follows:

- **Increasing the number of databases it provides access to**. For example, adding access to databases of non-human miRSNPs, as well as keeping track of new ones.
- **Creating a local cache**. The result of an user request could be stored on the server for a short period of time, in order to avoid making requests (to the original databases) that were recently made.
- **Applying data warehousing together with the virtual integration architecture**. This can be done in order to solve issues related to the availability of the different databases. All original data could be warehoused, creating a database that can answer data queries when a given miRSNP online database is offline.
- **Creating an easy-to-use way for adding new databases for data access**. miRSNaPi could provide an interface that allows miRSNP database creators to submit their online data to access integration.

8.1.2 miRdiver

miRdiver web client could also be improved, for example by:

- **Highlighting experimentally supported reports**. Not all relationships (gene-miRNA-SNP) are experimentally validated and it would be interesting to show which ones are.
- **Reporting strongly predicted miRSNP effects**. This can be achieved by using algorithms to work with the different database scores of predicted effects, giving the user a probabilistic approach to the results.

8.1.3 New tools

Apart from miRdiver, new and different web clients could be created, for instance, one could use machine learning algorithms in order to provide a miRSNP classification tool.

Bibliography

Bhattacharya, A., Ziebarth, J. D., and Cui, Y. (2014). PolymiRTS Database 3.0: linking polymorphisms in microRNAs and their target sites with human diseases and biological pathways. *Nucleic Acids Res.*, 42(Database issue):86–91.

Bruno, A. E., Li, L., Kalabus, J. L., Pan, Y., Yu, A., and Hu, Z. (2012). miRdSNP: a database of disease-associated SNPs and microRNA target sites on 3'UTRs of human genes. *BMC Genomics*, 13:44.

Doan, A., Halevy, A., and Ives, Z. (2012). *Principles of Data Integration*. Morgan Kaufmann.

Enright, A. J., John, B., Gaul, U., Tuschl, T., Sander, C., and Marks, D. S. (2003). MicroRNA targets in Drosophila. *Genome Biol.*, 5(1):R1.

Fielding, R. T. (2000). *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California, Irvine - United States of America.

Friedman, R. C., Farh, K. K., Burge, C. B., and Bartel, D. P. (2009). Most mammalian mRNAs are conserved targets of microRNAs. *Genome Res.*, 19(1):92–105.

Gong, J., Liu, C., Liu, W., Wu, Y., Ma, Z., Chen, H., and Guo, A. Y. (2015). An update of miRNASNP database for better SNP selection by GWAS data, miRNA expression and online tools. *Database (Oxford)*, 2015:bav029.

Gong, J., Tong, Y., Zhang, H. M., Wang, K., Hu, T., Shan, G., Sun, J., and Guo, A. Y. (2012). Genome-wide identification of SNPs in microRNA genes and the SNP effects on microRNA target binding and biogenesis. *Hum. Mutat.*, 33(1):254–263.

Griffiths-Jones, S. (2006). miRBase: the microRNA sequence database. *Methods Mol. Biol.*, 342:129–138.

Grimson, A., Farh, K. K., Johnston, W. K., Garrett-Engele, P., Lim, L. P., and Bartel, D. P. (2007). MicroRNA targeting specificity in mammals: determinants beyond seed pairing. *Mol. Cell*, 27(1):91–105.

Heimberg, A. M., Sempere, L. F., Moy, V. N., Donoghue, P. C., and Peterson, K. J. (2008). MicroRNAs and the advent of vertebrate morphological complexity. *Proc. Natl. Acad. Sci. U.S.A.*, 105(8):2946–2950.

Hrdlickova, B., de Almeida, R. C., Borek, Z., and Withoff, S. (2014). Genetic variation in the non-coding genome: Involvement of micro-RNAs and long non-coding RNAs in disease. *Biochim. Biophys. Acta*, 1842(10):1910–1922.

Jacobsen, A., Krogh, A., Kauppinen, S., and Lindow, M. (2010). miRMaid: a unified programming interface for microRNA data resources. *BMC Bioinformatics*, 11:29.

Krek, A., Grun, D., Poy, M. N., Wolf, R., Rosenberg, L., Epstein, E. J., MacMenamin, P., da Piedade, I., Gunsalus, K. C., Stoffel, M., and Rajewsky, N. (2005). Combinatorial microRNA target predictions. *Nat. Genet.*, 37(5):495–500.

Lagos-Quintana, M., Rauhut, R., Lendeckel, W., and Tuschl, T. (2001). Identification of novel genes coding for small expressed RNAs. *Science*, 294(5543):853–858.

Lenzerini, M. (2002). Data integration: A theoretical perspective. In *Proceedings of the Twenty-first ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS ’02, pages 233–246, New York, NY, USA. ACM.

Liu, C., Zhang, F., Li, T., Lu, M., Wang, L., Yue, W., and Zhang, D. (2012). MirSNP, a database of polymorphisms altering miRNA target sites, identifies miRNA-related SNPs in GWAS SNPs and eQTLs. *BMC Genomics*, 13:661.

Lujambio, A. and Lowe, S. W. (2012). The microcosmos of cancer. *Nature*, 482(7385):347–355.

Lukasik, A., Wojcikowski, M., and Zielenkiewicz, P. (2016). Tools4miRs - one place to gather all the tools for miRNA analysis. *Bioinformatics*.

Manyam, G., Ivan, C., Calin, G. A., and Coombes, K. R. (2013). targetHub: a programmable interface for miRNA-gene interactions. *Bioinformatics*, 29(20):2657–2658.

Molnar, A., Schwach, F., Studholme, D. J., Thuenemann, E. C., and Baulcombe, D. C. (2007). miRNAs control gene expression in the single-cell alga Chlamydomonas reinhardtii. *Nature*, 447(7148):1126–1129.

Pereira, T. C., de Santis Alves, C., e Silva, G. F. F., Ortiz-Morea, F. A., and Nogueira, F. T. S. (2015). *Introdução ao mundo dos microRNAs*, volume 1, chapter 5, pages 95–105. SBG, Rua Cap. Adelmio Norberto da Silva, 736.

Pinhal, D., Nachtigall, P. G., de Oliveira, A. C., Bovolenta, L. A., and Herkenhoff, M. E. (2015). *Introdução ao mundo dos microRNAs*, volume 1, chapter 5, pages 95–105. SBG, Rua Cap. Adelmio Norberto da Silva, 736.

Richardson, L. and Ruby, S. (2007). *Restful Web Services*. O’Reilly, first edition.

Sherry, S. T., Ward, M., and Sirotkin, K. (1999). dbSNP-database for single nucleotide polymorphisms and other classes of minor genetic variation. *Genome Res.*, 9(8):677–679.

Thomas, L. F., Saito, T., and Sætrom, P. (2011). Inferring causative variants in microRNA target sites. *Nucleic Acids Res.*, 39(16):e109.

Ziebarth, J. D., Bhattacharya, A., Chen, A., and Cui, Y. (2012). PolymiRTS Database 2.0: linking polymorphisms in microRNA target sites with human diseases and complex traits. *Nucleic Acids Res.*, 40(Database issue):D216–221.