UNIVERSIDAD DE LA LAGUNA

TRABAJO DE FIN DE GRADO

SYSTEMATIC SEARCH ON ASTRONOMICAL DATABASES

Instituto de Astrofísica de Canarias

AUTOR: ADRIÁN HERNÁNDEZ PADRÓN

TUTOR: IGNACIO TRUJILLO CABRERA

CURSO 2020/2021

Summary

Big Data is present in practically any area of our life and astronomy is no exception. We have at our disposal a massive amount of data thanks to the satellites/telescopes that we have to collect information day after day. This massive information comes raw, they are a lot of numbers that without the correct interpretation are useless, so it is necessary to have a series of tools to select and process the information with which we wish to work. This is where Gnuastro comes in, GNU Astronomy Utilities (Gnuastro) [1] is an official GNU package that contains a set of libraries and programs for manipulating and analyzing astronomical data through the Linux command line.

The GNU Astronomy Utilities (Gnuastro) is an official GNU package consisting of various programs and library functions for the manipulation and analysis of astronomical data. All the programs share the same basic command-line user interface for the comfort of both the users and developers. Gnuastro is written to comply fully with the GNU coding standards so it integrates finely with the GNU/Linux operating system. This also enables astronomers to expect a fully familiar experience in the source code, building, installing, and command-line user interaction that they have seen in all the other GNU software that they use.

The main goal for this project is to develop software that is going to come in Astquery, Astquery is the part of Gnuastro in charge of obtaining the information of the databases. Actually, Astquery works with four different databases (ned, Gaia, VizieR, Astron) and the program is built in C. The work done can be separated in three parts.

The first part is to change the way Astquery requests information from databases. Right now, when the program wants to requests information to a database, the program stops itself, makes a system call to execute a curl by the command line, and then returns to the program. This is not very optimal and can make the program fails, The solution for this is to do the cURL directly from the program and this can be done with a library called libcurl that works in the same way as the traditional cURL but in C.

In the second part, we want to add some of the APIs (Application Programming Interfaces) that SDSS Data Release 16 presents to access its information by writing a series of programs that work in the same way as Astquery, with the curl already corrected. Those programs are build in C language and present the same structure, the structure of this code is as follows:

- The data entry by console: This data entry had to be unique for each API and present a clear structure.
- Construction of the URL: To access the information we had to modify the base URL that the API gave with the input data from the console.
- Curl and output: The curl is executed with the URL already built and the output of the response table is prepared so that it is saved in memory as a variable. Doing that if the tables did not present the desired structure they could be modified in the same program and thus create a standard response.

The programs developed in C were the following: Radial query, rectangular query, SQL query, imaging, and spectrum.

Two programs written in Bash were also created, annexed to the image and spectrum program that using Bash syntax and some Gnuastro programs get all the images or spectra from a table that contains the coordinates and the specobjID of the objects.

The third part is about the creation of a program that is going to obtain all the possible information from an image or a space region. This works through VizieR because VizieR provides the most complete library of published astronomical catalogs –tables and associated data– with verified and enriched data, accessible via multiple interfaces. This program creates a file .txt with the information of all the tables that match the query and has the capacity to download all the tables and then save them in a separate folder. This program accepts three different inputs: Radial search, rectangular search and a fits image. For the fits image input the program uses skycoverage(a Gnuastro program) and gets the center and the limit region for the fits image, with that the program executes a simple search similar to the radial and rectangle.

This program is made in Bash and accesses one of the URLs that VizieR provides us and then manipulates the response to eliminate the empty tables. It has different entries in which we can search for all the catalogs of a region of the space, we can filter these catalogs so that it only shows those that contain keywords (such as a star) and we can also search for a specific column thanks to the fact that VizieR provides the Unified Content Descriptor (UCD).

To show all the tools, a real exercise was proposed as an example, the exercise was to obtain all the possible stars and quasars around the galaxy NGC1042 in a radius of 10 arcminutes. Once we had these stars and quasars, the PSF magnitude had to be obtained from the SDSS and then made some color diagrams [g-r] versus [r-i] for the stars and the quasars. And to finish we would use the SDSS tools to obtain the images and spectra of different parts of the graph. Throughout the exercise, we will be working through the Linux console making use of the tools created, some Gnuastro tools, and bash commands to streamline the process. As for the graphics part, Python3 is going to be used, specifically, the Pandas libraries to import the tables and Matplotlib to make the diagrams.

The document ends with a conclusion where the main problems that the databases present are discussed, these problems are mainly the lack of universality when requesting information. Astquery intends to solve this problem, to generate an interface where we can request the information from any database and that the response always presents the same structure. Also complementing it with all the other tools present in Gnuastro makes it a very versatile and powerful software when it comes to working with astronomical data.

Índice

1. Introducción

Big Data is present in practically any area of our life and astronomy is no exception. We have at our disposal a massive amount of data thanks to the satellites/telescopes that we have to collect information day after day. This massive information comes raw, they are a lot of numbers that without the correct interpretation are useless, so it is necessary to have a series of tools to select and process the information with which we wish to work. This is where Gnuastro comes in, GNU Astronomy Utilities (Gnuastro) is an official GNU package that contains a set of libraries and programs for manipulating and analyzing astronomical data through the Linux command line.

The GNU Astronomy Utilities (Gnuastro) is an official GNU package consisting of various programs and library functions for the manipulation and analysis of astronomical data. All the programs share the same basic command-line user interface for the comfort of both the users and developers. Gnuastro is written to comply fully with the GNU coding standards so it integrates finely with the GNU/Linux operating system. This also enables astronomers to expect a fully familiar experience in the source code, building, installing, and command-line user interaction that they have seen in all the other GNU software that they use.

The main goal for this project is to develop software that is going to come in Astquery, Astquery is the part of Gnuastro in charge of obtaining the information of the databases. Actually, Astquery works with four different databases (ned, Gaia, VizieR, Astron) and the program is built in C.

El Big Data está presente prácticamente en cualquier ámbito de nuestra vida y la astronomía no es una excepción. Tenemos a nuestra disposición una cantidad masiva de datos gracias a los satélites y telescopios que tenemos recopilando información día tras día. Esta masiva información viene en crudo, dado que son un montón de números que sin la interpretación correcta no serían de gran utilidad, por eso es necesario tener una serie de herramientas para seleccionar y tratar la información con la que deseamos trabajar. Aquí es donde entra Gnuastro, GNU Astronomy Utilities (Gnuastro)[1] es un paquete oficial de GNU que contiene un conjunto de librerías y programas para la manipulación y análisis de los datos astronómicos a través de la línea de comandos de linux.

1.1. Gnuastro

Gnuastro está escrito siguiendo los estándares de GNU por lo que se integra perfectamente con los sistemas operativos GNU/Linux y el código está escrito principalmente en C. La razón de esto es porque necesitamos que esta herramienta perdure en el tiempo y para eso el lenguaje C es el indicado, como sabemos este lenguaje es unos de los pilares de la programación y no va a cambiar en un futuro. Otra de las razones es que es un software libre y abierto, esto quiere decir que cualquiera puede interactuar con el código y puede mejorar o adaptarlo a su uso, esta es otra de las razones por la cual C es el lenguaje indicado. Aunque no sepamos nada de C el libro The C Programming Language[2] escrito por Brian Kernighan y Dennis Ritchie es considerado la

biblia de la programación en C y lo mejor es que, en tan solo 272 páginas (en la segunda edición), podemos adquirir los conocimientos necesarios para generar, modificar o entender cualquier código dentro de Gnuastro.

Dentro de Gnuastro se encuentran diferentes herramientas astronómicas, el proyecto estaba enfocado en las bases de datos astronómicas así que únicamente se trabajó en mejorar Astquery.

1.2. Astquery

Astquery es una serie de programas presentes en Gnuastro que permiten solicitar información de las bases de datos mediante la línea de comandos, inicialmente tenía en su módulo cuatro bases de datos añadidas: Gaia, Ned, Vizier y Astron.

Estas bases de datos fueron añadidas por Mohammad Akhlaghi y tenían en común que presentaban una interfaz tipo TAP (Traffic Access Point) para poder acceder a ellas. La manera de funcionar de Astquery era como acceder a cualquier base de datos (GAIA, SDSS, ...) por lo que para cada búsqueda deseada necesitábamos introducir una serie de parámetros.

1.2.1. Ejemplos de Astquery

Vamos a presentar una serie de ejemplos de manera que podamos entender como funciona esta herramienta para así comprender sus problemas y fortalezas.

Estos ejemplos están sacados de la web de Gnuastro donde se explican todas las posibilidades a la hora de usar Astquery. Por ejemplo, si queremos obtener el ID y las coordenadas RAJ2000, DEJ2000 de la base de datos GAIA edr3 de todos los objetos que se encuentren en nuestra imagen tan solo tenemos que correr el siguiente comando en la consola de Linux.

```
astquery gaia --dataset=edr3 --overlapwith=image.fits -csource_id,ra,
    dec
```

Código 1: Primer ejemplo de uso de Astquery.

Otro ejemplo podría ser todos los objetos en una determinada región que posean un redshift dentro de un rango determinado, con el siguiente comando estamos accediendo al SDSS data release 12 a través de VizieR buscando en torno a un centro y un radio aquellos objetos que se encuentren con un redshift entre [0.5, 5].

```
astquery vizier --dataset=sdss12 --center=113.8729761,31.9027152 --
    radius=1/60 -cRA_ICRS,DE_ICRS,zsp --range=zsp,0.5,5
```

Código 2: Segundo ejemplo de uso de Astquery.

Ambos ejemplos generan una tabla con una extensión .fits puesto que esta extensión es con la que Gnuastro principalmente trabaja.

Podemos ver como en Astquery podemos solicitar toda la información existente en estas bases de datos y tenemos la particularidad de que en un mismo software podemos acceder a diferentes bases de datos. Esto elimina el tener que acceder a la información mediante la página web de cada base de datos por lo que la convierte en una herramienta muy poderosa.

1.2.2. Problemas de Astquery

Sin embargo aunque el código funcionaba perfectamente y se podía realizar las búsquedas a estas bases de datos sin ningún tipo de problema, la manera en la que se realizaban las peticiones a cada una de las bases de datos no era óptima. Para acceder a las páginas web de las correspondientes bases de datos Astquery usaba curl, este es un software libre que se usa por línea de comandos principalmente para acceder a páginas web, descargar enlaces o como en nuestro caso, hacer peticiones.

```
❯ curl https://www.gnu.org/software/gnuastro/ | html2text | head -300
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 33352    0 33352    0     0  48689      0 --:--:-- --:--:-- --:--:-- 48618









Skip_to_main_text
     Free_Software_Supporter:  [email address     ] [Sign up]
JOIN THE FSF
[ [A_GNU_head] ]GNU_Operating_System
Supported by the Free_Software_Foundation
[ [Search_www.gnu.org] ] [ [Languages] ]

Site_navigation Skip
    * ABOUT GNU
    * PHILOSOPHY
    * LICENSES
    * EDUCATION
    * = SOFTWARE  =
    * DOCS
    * MALWARE
    * HELP GNU
    * GNU ART
    * GNU'S WHO?
    * SITEMAP
    * SOFTWARE DIRECTORY
    * HARDWARE
***** GNU Astronomy Utilities *****
The GNU Astronomy Utilities (Gnuastro) is an official GNU package consisting of
various programs and library functions for the manipulation and analysis of
astronomical data. All the programs share the same basic command-line user
interface for the comfort of both the users and developers. Gnuastro is written
to comply fully with the GNU_coding_standards so it integrates finely with the
GNU/Linux operating system. This also enables astronomers to expect a fully
familiar experience in the source code, building, installing and command-line
user interaction that they have seen in all the other GNU software that they
use.
In case you are new to Gnuastro, you might find these links useful:
    * Quick_start: To install Gnuastro.
    * List_of_Gnuastro_programs: For a complete list of the programs.
```

Figura 1: El comando que se muestra es un simple curl a la web de Gnuastro, el cual obtiene como respuesta todo el texto escrito en dicha web.

El problema que presentaba este sistema era que para poder hacer el curl se tenía que parar el programa, ejecutar el curl por la línea de comando y luego devolver el control al programa, por lo que no era óptimo.

Para solucionar esto y buscar una manera de acceder estas bases de datos sin parar el

programa se propuso añadir una nueva base de datos a Astquery escribiendo el código en C desde cero. Además de esto, se pretendía desarrollar un método para añadir bases de datos las cuales no tuvieran una interfaz tipo TAP para acceder a la información.

2. Libcurl

In this section the Astquery problem is solved, a C library called libcurl is presented as a solution to the problem. This library works like the traditional curl and got some features that simplify the C script build a lot. As an example this presents how can make an existence Astquery query using libcurl instead cURL.

El problema del curl en Astquery se solucionó usando una librería llamada libcurl[3] que presentaba una serie de utilidades que facilitaba enormemente el trabajo. Esta librería era una adaptación para C del cURL que Astquery usaba hasta entonces, por lo que garantizaba que, si la conseguíamos incluir en el código que ya se encontraba escrito el programa funcionaría sin ningún tipo de problema.

Una de las utilidades que esta librería presentaba fue la que se utilizó para dar los primeros pasos en cada script. Si hacemos un curl por consola a cualquier página web y añadimos el comando "–libcurl program_name.c" al principio de este, se generaba automáticamente un script en C el cual cuando se ejecutaba iba a acceder a esta URL como si de la página web se tratase.

Entonces, ejecutando el siguiente comando en la URL de la API del SDSS ya se lograba obtener las primeras líneas de código:

```
curl --libcurl nombre_script.c "http://skyserver.sdss.org/dr16/
    SkyServerWS/SearchTools/RadialSearch?ra=258.2&dec=64&radius=4.1&
    whichway=equatorial&limit=10&format=txt&fp=none&uband=0,17&check_u=
    u&gband=0,15&check_g=g&whichquery=imaging"
```

Código 3: Ejemplo de uso del comando con la API de la búsqueda radial.

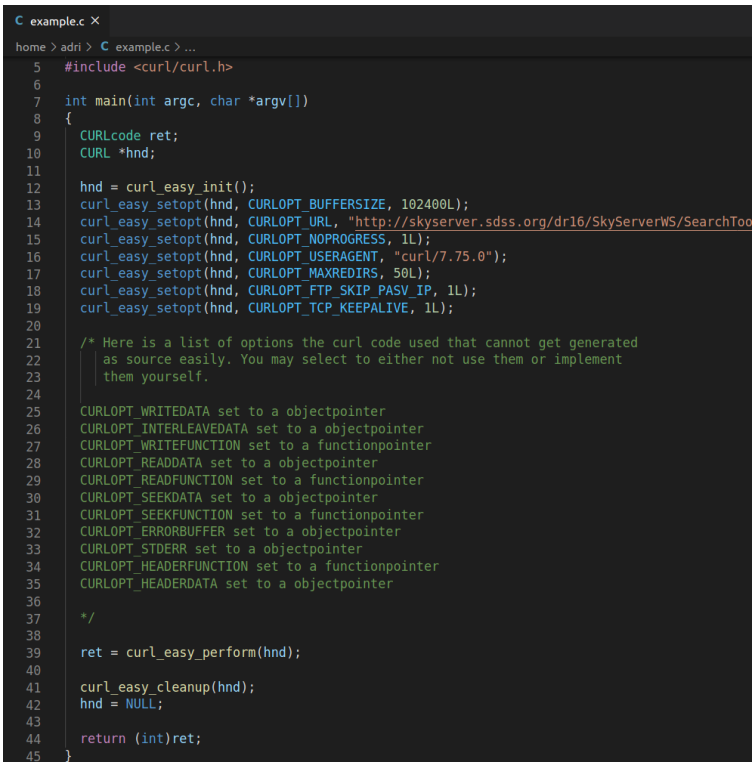

Figura 2: Script generado.

Como se ve en la imagen, el código que se genera tiene lo necesario para realizar el curl al enlace de la API. Para poder ejecutar este programa por línea de comandos, era necesario compilarlo primero usando el siguiente comando:

```
gcc nombre_script.c -lcurl nombre_script
```

Código 4: Comando necesario para compilar el programa y generar un ejecutable.

Una vez teníamos el ejecutable y lo lanzábamos por línea de comandos obteníamos el siguiente resultado:

Figura 3: En esta imagen se muestra como se compila el programa, se le dan permisos de ejecución y, por último, se ejecuta.

Esta tabla que se obtiene como el resultado del programa es el mismo que se obtiene en el ejemplo que se muestra en la API del SDSS.

2.1. Solución del problema de Astquery

De esta manera si se conseguía introducir este método a los programas presentes en Astquery se solucionaría el problema. Para ello se ejecutó uno de los ejemplos de la página de Astquery, puesto que este mostraba la petición TAP que hacía. Con esta petición ejecutábamos el comando anterior cambiando únicamente la extensión de fits a csv para poder visualizarlo por consola.

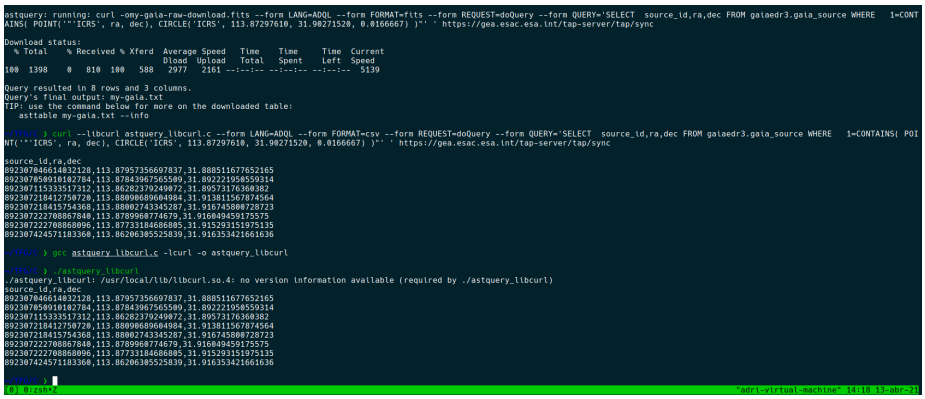

Figura 4: En esta imagen se ejecuta el comando anterior con la petición TAP generada por Astquery. Además, se compila y ejecuta el programa generado(astquery_libcurl) mostrando su funcionamiento.

Como se puede ver en la Figura 4, el programa generado otorga la misma respuesta que el propio Astquery, esto soluciona el problema y demuestra que esta librería es la indicada para introducir en el código de Astquery debido a que es nativa en C y muy completa.

3. SDSS

This section begins by commenting on how to access Data Release 16 of the SDSS, then it shows how the programs that access the SDSS API are structured, in addition to presenting the input and characteristics of each one of them. The programs written in Bash that complement the previous ones.

La base de datos indicada para añadir a Astquery era el Sloan Digital Sky Server[4] ya que, aunque a esta base de datos se podía acceder únicamente a través de VizieR, nos posibilita la entrada al Data Release 13.

El SDSS presentaba una serie de APIs para así poder acceder directamente a la información del Data Release 16. Otro punto a favor es que esta información no se limitaba únicamente a tablas de datos, sino que también se podía acceder a imágenes y espectros.

3.1. Estructura del código

La estructura de los programas para acceder a las APIs se podía dividir en tres partes:

- La entrada de datos por consola: esta entrada de datos tenía que ser única para cada API y presentar una estructura clara.
- Construcción de la URL: para acceder a la información teníamos que modificar la URL base que daba la API con los datos de entrada por consola.
- curl y salida: se ejecuta el curl con la URL ya construida y se prepara la salida de la tabla respuesta de manera que se guarda en la memoria en forma de variable, de esta manera si las tablas no presentaban la estructura deseada se podrían modificar en el mismo programa y así crear una respuesta estándar.

3.2. Entrada de datos por consola

La entrada de datos por consola debía contener los parámetros necesarios para solicitar información en cada una de las APIs: bandas, límites, formato de salida, id del objeto... Teniendo en cuenta esto, el código se tenía que escribir de manera que recogiera los datos introducidos por consola y los preparara para poder añadirlos luego a la la URL.

Para esta entrada de datos por consola se usó getops, este comando se incluía simplemente añadiendo la librería getopts en C. La función usada fue concretamente getops_long la cual permite que usemos tanto palabras de texto corto como de texto largo para la entrada de datos: para el caso de radio podemos usar –r o —radio, ambos formatos son completamente válidos y el programa lo acepta por igual.

Como estábamos trabajando en C es necesario añadir esto en la función main junto con argc y argv que son dos variables predefinidas dentro del lenguaje que reciben los argumentos que se pasan al ejecutar el programa.

```
int main(int argc, char **argv)
{
    static struct option long_options[] =
    {
    {"radius", required_argument,0, 'r'},
    {"centre", required_argument,0, 'c'},
    {"limit", optional_argument,0, 'l'},
    {"format", required_argument,0, 'f'},
    {"uband", required_argument,0, 'u'},
    {"gband", required_argument,0, 'g'},
```

```
    {"rband", required_argument,0, 'j'},
    {"iband", required_argument,0, 'i'},
    {"zband", required_argument,0, 'z'},
    {"output", required_argument,0, 'o'},
    {"help", no_argument,0, 'h'},
    {0 , 0, 0, 0}
};
```

Código 5: Código perteneciente al programa de búsqueda radial. Todas estas opciones a exepción de "*help*" eran las el programa aceptaba.

El paso siguiente era guardar la información de entrada en variables, esto se hacía con un switch de manera que el programa entraba en cada 'case' si al ejecutar dicho programa se indicaba su argumento en la línea de comandos.

```
while ((opt = getopt_long(argc, argv, "r:c:l:f:o:u:g:j:i:z:h",
    long_options, &option_index)) != -1) {
    switch (opt) {
  case 'r':
    radio = optarg;
    ++parameter_counter;
    break;
  case 'c':
    centro = optarg;
    ++parameter_counter;
    break;
  case 'l':
    if (optarg) {
      limite = atoi(optarg);
    }
    break;
  case 'f':
    if (optarg) {
        formato = optarg;
    }
    break;
  case 'o':
    salida = optarg;
    break;
  case 'u':
    u = optarg;
    snprintf(u_band, 40, "&uband=%s&check_u=u",u);
    break;
```

Código 6: Código perteneciente al programa de búsqueda radial. La estructura de cada case depende de se manipula la información de entrada.

Sin embargo no todos estos argumentos eran necesarios para solicitar información, cada programa necesitaba un mínimo de argumentos para funcionar que dependía de la API, además estos programas están configurados para introducir valores por defecto en algunos casos y para manipular la información de entrada de manera que se pudiera usar posteriormente.

Con esto ya podremos introducir datos en nuestro programa por la línea de comandos, es decir esto lo que permite es que cuando se ejecute el programa de la siguiente manera:

```
      ./radial_query --radius=1 --centre=255.990851710,40.250541
```

Código 7: Ejemplo de uso sencillo del programa de la busqueda radial.

El programa tiene la información guardada en variables de cuanto vale el radio y cual es el centro.

3.3. Construccion de la URL

Esta URL se puede formar haciendo uso de una función de C llamada snprintf(), en la cual se tenía que escribir primero la base de la URL de cada API correspondiente.

```
"http://skyserver.sdss.org/dr16/SkyServerWS/SearchTools/RadialSearch?ra
    =&dec=&radius=&whichway=equatorial&limit=&format=&whichquery=
    imaging"
```

Código 8: URL base de la API de busqueda radial.

```
snprintf(url, 400, "http://skyserver.sdss.org/dr16/SkyServerWS/
    SearchTools/RadialSearch?ra=%s&dec=%s&radius=%s&whichway=equatorial
    &limit=%d&format=%s&whichquery=imaging%s%s%s%s%s",ra, dec, radio,
    limite ,formato,u_band, g_band, r_band, i_band, z_band);
```

Código 9: Código perteneciente al programa radialquery. La función almacena la respuesta en la vartiable url.

Se conseguía sustituir cada uno de los %s y %d por la información recogida de la entrada y manipulada correctamente para que no de ningún tipo de error. De esta manera, la URL necesaria para solicitar la información a la API ya está correctamente construida y almacenada en la variable url.

3.4. curl y salida

Ya con la URL construida simplemente bastaba con realizar el curl a la API correspondiente para obtener la respuesta de esta. De primeras esta respuesta se podía visualizar en la consola directamente, si bien este es el propósito del programa, no es la manera correcta de ejecutarlo, ya que esta respuesta por consola sería distinta para cada una de las bases de datos de Astquery.

La solución a este problema era buscar la manera de guardar la respuesta en la memoria, es decir, en una variable para que esta respuesta se pueda modificar y así la salida presentara la misma estructura en cada base de datos.

Para ello era necesario asignar espacio en la memoria para la respuesta y escribir la función WriteMemoryCallback:

```
struct MemoryStruct {
    char *memory;
    size_t size;
};

      static size_t
WriteMemoryCallback(void *contents, size_t size, size_t nmemb, void *
    userp)
{
```

```
  size_t realsize = size * nmemb;
  struct MemoryStruct *mem = (struct MemoryStruct *)userp;

  char *ptr = realloc(mem->memory, mem->size + realsize + 1);
  if(ptr == NULL) {
    /* out of memory! */
    printf("not enough memory (realloc returned NULL)\n");
    return 0;
  }

  mem->memory = ptr;
  memcpy(&(mem->memory[mem->size]), contents, realsize);
  mem->size += realsize;
  mem->memory[mem->size] = 0;

  return realsize;
}
```

Código 10: Código perteneciente al programa de búsqueda radial. Esta parte del código se encarga de asignar un espacio en la memoria para después guardar la respuesta en una variable.

Y, con esto, se asignaba la respuesta que se obtenía del SDSS en este espacio de memoria que hemos asignado previamente. Esto lo hacíamos desde la propia función que se encargaba de acceder a la API con libcurl.

```
int Radial_query(char *url){

  CURLcode ret;
  CURL *hnd;
  struct MemoryStruct query;

  query.memory = malloc(1);
  query.size = 0;
  hnd = curl_easy_init();

  curl_easy_setopt(hnd, CURLOPT_BUFFERSIZE, 102400L);
  curl_easy_setopt(hnd, CURLOPT_URL, url);
  curl_easy_setopt(hnd, CURLOPT_WRITEFUNCTION, WriteMemoryCallback);
  curl_easy_setopt(hnd, CURLOPT_WRITEDATA, (void *)&query);
  curl_easy_setopt(hnd, CURLOPT_NOPROGRESS, 1L);
  curl_easy_setopt(hnd, CURLOPT_USERAGENT, "curl/7.75.0");


  ret = curl_easy_perform(hnd);

  printf("%s", query.memory);

  curl_easy_cleanup(hnd);
  hnd = NULL;

  return 0;
}
```

Código 11: Función generada por libcurl encargada de hacer el curl y modificada para guardar la información en una variable.

3.5. Programas desarrollados usando esta metodología

Todos estos programas se escribieron en C usando la metodología anterior por lo que su estructura principal es la misma aunque algunos presenten diferencias a la hora de tratar con los datos de entrada por consola.

3.5.1. Búsqueda radial

Este programa realiza una búsqueda radial en torno a un centro y cierto rango obteniendo las magnitudes fotométricas de los objetos dentro de esta área. El programa contiene todas sus características y requiere de dos argumentos obligatorios: el centro y el radio. Con estos dos argumentos ya se puede realizar cualquier búsqueda y obtendremos como respuesta una tabla con 19 columnas.

Los parámetros que acepta el programa son los siguientes:

- Radio[-r][–radius]: radio en arcominutos.
- Centro[-c][–centre]: centro(ra,dec) en grados.
- Nombre de la tabla [-o][–output]: nombre de la tabla que obtenemos por respuesta,
- Formato [-f][–format]: formato de la tabla (csv html xml votable json fits mydb), si no se solicita ninguno la tabla presentará un formato txt.
- Límite [-l][–limit]: límite de filas que tendrá la tabla, por defecto se mostrarán todas las filas.
- Bandas [-*][–*band]: límite superior para las diferentes bandas fotométricas(* se substituye por las bandas u, g, r, i o z en cada caso)
- Panel de ayuda [-h][–help]

3.5.2. Búsqueda rectangular

Esta API funciona igual que la anterior salvo que la búsqueda es rectangular, requiere de dos argumentos como mínimo: rango de la coordenada RA y rango de la coordenada DEC. El resultado de respuesta es el mismo, una tabla con 19 columnas.

Los parámetros que acepta el programa son los siguientes:

- Ascención recta [-r][–ra]: valor mínimo y máximo de la ascención recta en grados.
- Declinación [-d][–de]: valor mínimo y máximo de la declinación en grados.

- Nombre de la tabla [-o][–output]: nombre de la tabla que obtenemos por respuesta,
- Formato [-f][–format]: formato de la tabla (csv html xml votable json fits mydb), si no se solicita ninguno la tabla presentará un formato txt.
- Límite [-l][–limit]: límite de filas que tendrá la tabla, por defecto se mostrarán todas las filas.
- Bandas [-*][–*band]: límite superior para las diferentes bandas fotométricas(* se substituye por las bandas u, g, r, i o z en cada caso)
- Panel de ayuda [-g][–help]

3.5.3. Imagen

Con este programa se puede obtener una imagen tomadas por el telescopio del SDSS, la respuesta se guarda como una imagen.

Los parámetros que acepta el programa son los siguientes:

- Escala [-s][–scale]: escala de la imagen en arcosegundo por pixel, el tamaño predeterminado es 0.4
- Centro [-c][–centro]: centro(ra, dec) en grados.
- Altura [-h][–height]: en píxeles.
- Anchura [-w][–wide]: en píxeles.
- Nombre de la imagen [-o][–output]: nombre de la imagen respuesta.
- Panel de ayuda [-h][–help]
- OPT [-l][–overlay]: Una cadena de caracteres que actúan como efectos para resaltar la imagen.

Code	Effect on image
G	Grid
L	Label
P	PhotoObjs
S	SpecObjs
T	TargetObjs
O	Outline
B	BoundingBox
F	Fields
M	Masks
Q	Plates
I	InvertImage

Figura 5: Lista de los diferentes efectos que podemos usar en la imagen.

3.5.4. Espectro

De una manera similar al programa de imagen, este programa obtiene el espectro del objeto solicitado. Funciona únicamente con la id del SDSS del objeto. Este programa acepta solo dos parametros:

- ID [-i][–objid]: esta id tiene que ser la specobjID del objeto.
- Panel de ayuda [-h][–help]

3.5.5. SQL

Este programa es el más completo de todos los anteriores y es con el que podemos acceder a toda la información que se encuentra en el SDSS. De hecho, los perfiles radiales y rectangular se pueden considerar una petición SQL. La entrada de datos por consola requiere de conocimientos en SQL y la respuesta de salida es idéntica al perfil radial y rectangular aunque el número de columnas depende de la petición.

- Petición SQL [-s][–sqlquery]: en este comando se tiene que escribir entre comillas la petición SQL que se quiera hacer.
- Nombre de la tabla [-o][–output]: nombre de la tabla que obtenemos por respuesta.
- Formato [-f][–format]: formato de la tabla (csv html xml votable json fits mydb), si no se solicita ninguno la tabla presentará un formato txt.
- Panel de ayuda [-h][–help]

3.6. Programas anexos

Estos programas forman parte del paquete de programas del SDSS aunque están escritos en Bash y su función es ejecutar algunos de los programas anteriores para múltiples objetos.

3.6.1. Macro imagen

Este programa tiene relación con el de imagen y funciona tal que dada una tabla de objetos en la que se tenga sus coordenadas, el programa va a generar una carpeta con la imagen de cada uno de estos objetos proveniente del SDSS y un archivo con su enlace a la documentación en el SkyServer. Usando una de las herramientas de Gnuastro conocida como Asttable y sintaxis de Bash obtenemos las coordenadas, las cuales se usan en un bucle sobre el programa de imagen del SDSS presentado anteriormente generando como resultado una carpeta con las imágenes.

```
coordinates=$(asttable $file -c $column | sed -e 's/ /,/' | sed -s 's/
    //g')
img_counter=0
for i in $coordinates; do
    let img_counter+=1
    ./sdss16imaging --centre=$i --output="($img_counter)$i" --overlay=
        GOL 2>/dev/null
    mv "($img_counter)$i" $destination
    ra=$(echo $i | sed -e 's/,/ /' | awk '{print $1}')
    dec=$(echo $i | sed -e 's/,/ /' | awk '{print $2}')
    echo "($img_counter)$i" >> $urlfile
    echo "http://skyserver.sdss.org/dr16/en/tools/explore/Summary.aspx?
        ra=$ra&dec=$dec" >> $urlfile
done
mv $urlfile $destination
```

Código 12: El programa funciona simplemente con este código. Usando asttable guardamos en una variable las coordenadas RA y DEC que posteriormente introducimos en el programa de imagen.

El programa necesita cuatro comandos para funcionar que son los siguientes:

- Tabla [-f]: Tabla que contiene las coordenadas de los objetos.
- Columnas [-c]: Posición que ocupan las columnas o nombre de esta.
- Nombre de la carpeta [-d]
- Nombre de la URL [-u]
- Panel de ayuda [-h]

3.6.2. Macro espectro

Este programa trabaja de igual manera que el macro imagen con la excepción que funciona con las columnas en las que se encuentran las coordenadas sino la columna en donde esta el specobjid del SDSS.

De igual manera que el anterior, este programa necesita cuatro comandos para funcionar:

- Tabla [-f]: Tabla que contiene las coordenadas de los objetos.
- Columnas [-c]: Posición que ocupan la columna o nombre de esta (para indicar la posición se tiene que usar $1,$2,..)
- Nombre de la carpeta [-d]
- Nombre de la URL [-u]
- Panel de ayuda [-h]

4. VizieR tool

This is about the creation of a program that is going to obtain all the possible information from an image or a space region. This works through VizieR because VizieR provides the most complete library of published astronomical catalogs –tables and associated data– with verified and enriched data, accessible via multiple interfaces. This program creates a file .txt with the information of all the tables that match the query and has the capacity to download all the tables and then save them in a separate folder. This program accepts three different inputs: Radial search, rectangular search and a fits image. For the fits image input the program uses skycoverage(a Gnuastro program) and gets the center and the limit region for the fits image, with that the program executes a simple search similar to the radial and rectangle.

Además del SDSS y mejorar Astquery el proyecto presentaba otro objetivo, dada una imagen o una región del espacio se quería obtener todos los objetos conocidos dentro de esta. Es decir, de una manera similar funciona Aladin[6] se pretendía desarrollar una herramienta con la cual pudieramos obtener toda la información posible de la región seleccionada.

La clave de esta herramienta recaía en trabajar a través de VizieR[5]. Esta proporciona la biblioteca más completa de catálogos astronómicos publicados (tablas, datos asociados, bases de datos) con datos verificados y accesibles a través de multiples interfaces.

Mohammad Akhlaghi junto con François Ochsenbein (creador de vizquery, un código de python para acceder a esta base de datos) habían dado los primeros pasos para la creación de esta herramienta y era el siguiente comando:

```
curl -oout.tsv --form -c=23.4621+30.66 --form -c.rm=5 http://vizier.
    unistra.fr/viz-bin/asu-tsv?-meta
```

Código 13: El fichero respuesta out.tsv solo iba a mostrar información de las tablas que tuvieran objetos entorno al centro(23.4621, 30.66) y un radio de 5 arcominutos.

Esta comando realizaba un curl a la web de VizieR solicitando información entorno a un centro y un radio en arcmin. La respuesta de este comando era archivo tsv(tab-separated-values) que indicaba todas las tablas de VizieR que tenían algún tipo de información de la zona solicitada.

Aunque esto resultaba útil, la respuesta era algo incompleta puesto que si se querían ver las tablas era necesario acceder a la página web de VizieR, por lo que lo primero que había que investigar era la URL a la que solicitábamos información. Tras la búsqueda de información en la propia documentación de VizieR y probando diferentes alternativas se cambió la parte final de la URL ?-meta por ?-out, con esto el comando anterior además de generar el archivo tsv con información de las tablas que se encuentran en la región solicitada iba a mostrar estas tablas.

El número de columnas que iban a presentar estas tablas dependía de que indicaramos en la propia URL, para empezar se añadió que esta tablas presentaran las coordenadas. El comando por tanto pasó a ser:

```
curl -oout.tsv --form -c=23.4621+30.66 --form -c.rm=5 "http://vizier.
    unistra.fr/viz-bin/asu-tsv?-out=*pos.eq.ra;meta.main,*pos.eq.dec;
    meta.main"
```

Código 14: Las tablas van a mostrar unicamente dos columnas RAJ2000 y DECJ2000.

4.1. Burpsuite

Una vez se tenía la URL indicada, teníamos que mejorar la información de entrada para poder buscar tablas más detalladas o buscar por ciertas columnas, para ello se usó Burpsuite[7].

Burpsuite es una herramienta la cual permite interceptar peticiones en las páginas web, de esta manera, probando en la web de VizieR diferentes búsquedas, se pudo identificar cómo se podían añadir diferentes características. Estas características fueron:

- Búsqueda por palabras claves, funciona de igual manera que la web de VizieR
- Búsqueda por Unified Content Descriptor (UCD), los UCD representan una clasificación de diferentes tipos de parámetros que tiene como objetivo la asignación de nombres genéricos para todos los parámetros contenidos en los catálogos.

Con estas opciones teníamos un comando por consola que permitía una búsqueda igual de completa que en la web de VizieR.

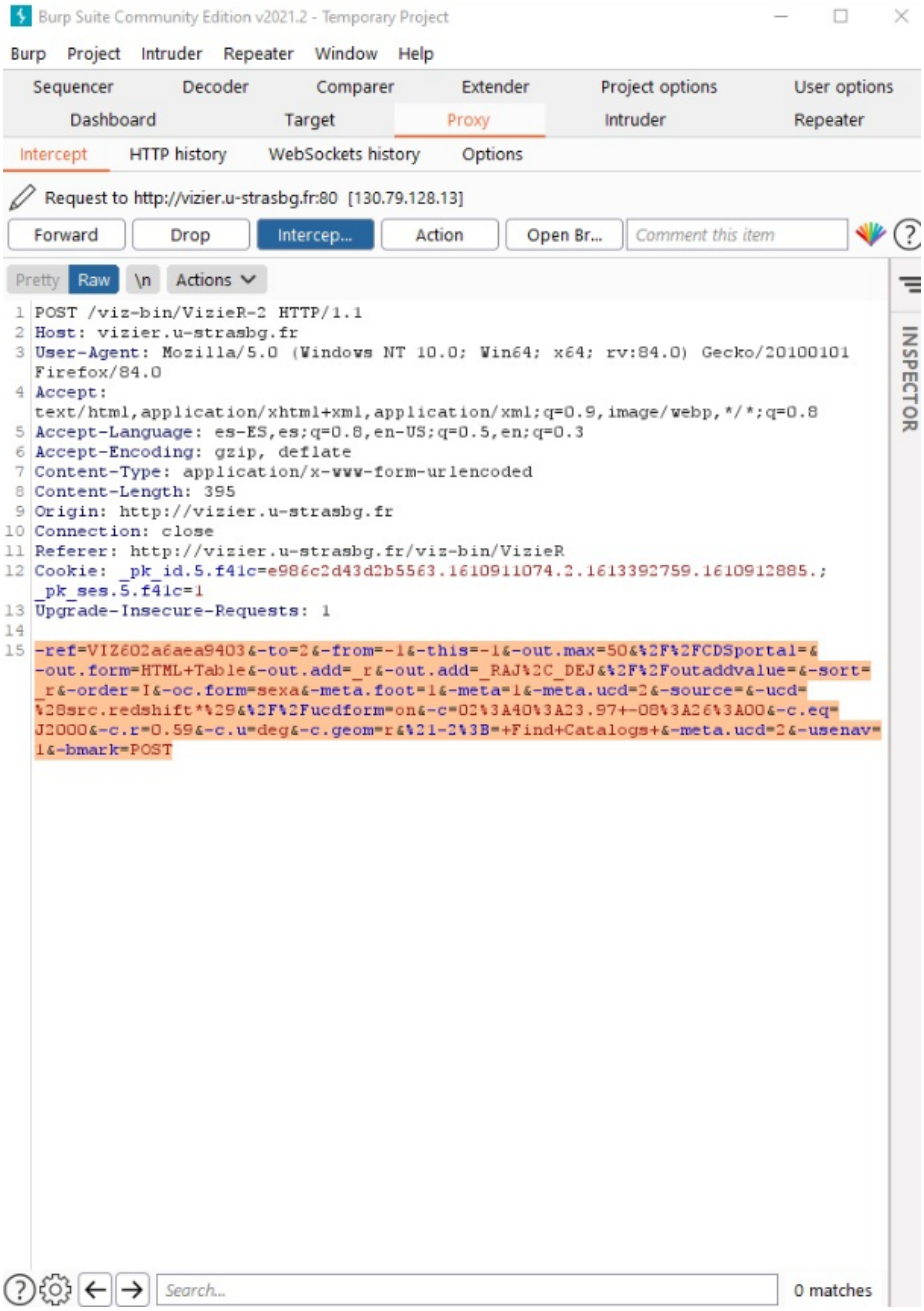

Figura 6: En esta imagen se puede ver como burpsuite está interceptando una petición. De esta manera se pudo identificar los parámetros necesarios para después usarlos por consola.

4.2. Características del script

Para la escritura del programa se optó por usar Bash, la razón de esto era que iba a ser necesario una manipulación del fichero respuesta del curl y Bash es el indicado para manipular cadenas de caracteres.

El programa acepta tres entradas distintas:

- Perfil radial[-m][-r]: como en el SDSS el programa necesita una entrada del centro[-m] en grados y el radio[-r] en arcominutos.

```
curl -o rawdata.txt --form -c=$center --form -c.rm="$radio" --form
     -ucd=$columns_input --form -source=$key_word "http://vizier.
    unistra.fr/viz-bin/asu-txt?-out=*pos.eq.ra;meta.main,*pos.eq.
    dec;meta.main,$columns_url"
```

Código 15: Código encargado de realizar el curl si se solicita el perfil radial con la información obtenida por consola.

- Perfil rectangular[-m][-w]: de igual manera necesita un centro[-m] en grados y unos límites[-w] en arcominutos.

```
curl -o rawdata.txt --form -c=$center --form -c.bm="$width" --form
     -ucd=$columns_input --form -source=$key_word "http://vizier.
    unistra.fr/viz-bin/asu-txt?-out=*pos.eq.ra;meta.main,*pos.eq.
    dec;meta.main,$columns_url"
```

Código 16: Código encargado de realizar el curl si se solicita el perfil rectangular con la información obtenida por consola.

- Imagen[-o]: esta entrada es la más prometedora, dada una imagen fits el programa obtiene el centro y el radio usando una función de Gnuastro conocida como Skycoverage.

```
centerra=$(astfits -h0 $overlapwith --skycoverage | grep "Center:"
     | awk '{print $2}')
centerde=$(astfits -h0 $overlapwith --skycoverage | grep "Center:"
     | awk '{print $3}')
rad1=$(astfits -h0 $overlapwith --skycoverage | grep "Width:" |
    awk '{print $2}')
rad2=$(astfits -h0 $overlapwith --skycoverage | grep "Width:" |
    awk '{print $3}')
curl -o rawdata.txt --form -c="$centerra $centerde" --form -c.bd="
    $rad1/2,$rad2/2" --form -ucd=$columns_input --form -source=
    $key_word "http://vizier.unistra.fr/viz-bin/asu-txt?-out=*pos.
    eq.ra;meta.main,*pos.eq.dec;meta.main,$columns_url"
```

Código 17: Código encargado de realizar el curl si se solicita el perfil imagen con la información obtenida por consola.

Además de esto el programa presenta las siguientes opciones:

- Columna [-c]: columna deseada, esta debe estar escrita siguiendo los UCD de VizieR.

- Palabra clave [-d]: de esta manera solo busca tablas que contengan las palabras introducidas.
- Opción de descarga [-s]: este comando descarga todas las tablas que se encuentren en VizieR en la región seleccionada usando Astquery.
- Lista de UCD [-u]: otorga una URL con la lista completa de todos los UCD.
- Panel de ayuda [-h]

Cada uno de los perfiles genera un fichero txt llamado rawdata.txt en el cual está toda la información que devuelve VizieR. Esta presenta un problema con esto, pues en la propia web cuando realizamos una búsqueda y queremos que nos muestre las tablas tal y como hace el programa, puede llegar a mostrar un gran número de tablas vacías. Este hecho se ve reflejado tanto en la web como en el fichero rawdata.txt por lo que dificulta enormemente la lectura de la información que solicitamos. Por esto es necesario modificar este fichero respuesta de manera que solo muestre estas tablas que sí contengan información.

Esto se soluciona de una manera relativamente sencilla introduciendo el siguiente código:

```
resource=$(cat rawdata.txt | grep -B 7 'pos.eq.ra' | grep '#RESOURCE')
for NAME in $resource; do
  sed -n "/$NAME/,/^$/p" rawdata.txt >> "$key_word$columns_input"
      _catalogs.txt
done
rm rawdata.txt
```

Código 18: Código usado en el script de VizieR para remover todas las tablas vacías

Como el #RESOURCE es distinto en cada tabla, este código selecciona solo aquellos #RESOURCE que no presenten una tabla vacia y los envía a un fichero nuevo.

```
#RESOURCE=yCat_7273
#Name: VII/273
#Title: The Half Million Quasars (HMQ) catalogue (Flesch, 2015)
#Table VII_273_hmq:
#Name: VII/273/hmq
#Title: The Half Million Quasars catalogue
#---Details of Columns:
    RAJ2000 (deg)     (F10.6) Right ascension (J2000) (1) [ucd=pos.eq.ra;meta.main]
    DEJ2000 (deg)     (F10.6) Declination (J2000) (1) [ucd=pos.eq.dec;meta.main]
---------- ----------
RAJ        DEJ
2000 (deg) 2000 (deg)
---------- ----------
040.020253 -08.560225
040.090891 -08.361691
040.128862 -08.514984
040.148615 -08.547080
040.153823 -08.589969
040.199426 -08.463399
040.232548 -08.331474

#RESOURCE=yCat_7279
#Name: VII/279
#Title: SDSS quasar catalog: twelfth data release (Paris+, 2017)
#Table VII_279_dr12q:
#Name: VII/279/dr12q
#Title: SDSS-DR12 Quasar Catalog
#---Details of Columns:
    RAJ2000 (deg)     (F10.6) Right Ascension (J2000) (RA) [ucd=pos.eq.ra;meta.main]
    DEJ2000 (deg)     (F10.6) Declination (J2000) (DEC) [ucd=pos.eq.dec;meta.main]
---------- ----------
RAJ        DEJ
2000 (deg) 2000 (deg)
---------- ----------
040.020253 -08.560225
040.128863 -08.514984
```

Figura 7: Estructura del fichero eliminando las tablas vacias.

5. Ejemplo práctico

To show all the tools, a real exercise was proposed as an example, the exercise was to obtain all the possible stars and quasars around the galaxy NGC1042 in a radius of 10 arcminutes. Once we had these stars and quasars, the PSF magnitude had to be obtained from the SDSS and then made some color diagrams [g-r] versus [r-i] for the stars and the quasars. And to finish we would use the SDSS tools to obtain the images and spectra of different parts of the graph. Throughout the exercise, we will be working through the Linux console making use of the tools created, some Gnuastro tools, and bash commands to streamline the process. As for the graphics part, Python3 is going to be used, specifically, the Pandas libraries to import the tables and Matplotlib to make the diagrams. It is a very detailed example in which we show all the steps taken.

Para mostrar todas las herramientas se planteó un ejercicio real a modo de ejemplo. Este ejercicio consistía en obtener todas las estrellas y quásares recopilados en las bases de datos públicos entorno a la galaxia NGC1042 en un radio de 10 arcominutos. Una vez se tenían estas estrellas y quásares, se tenía que obtener la magnitud PSF por parte del SDSS y con ello realizar unos diagramas de color [g-r] frente a [r-i] tanto para las estrellas como para los quásares. Finalmente haríamos uso de la herramientas del SDSS para obtener las imágenes y espectros de diferentes partes de la gráfica.

Durante todo el ejercicio vamos a estar trabajando por la consola de Linux haciendo uso de las herramientas creadas, de algunas herramientas de Gnuastro y de comandos de Bash para agilizar el proceso. En cuanto a la parte de las gráficas se va a usar Python 3, concretamente las librerías Pandas[10] para importar las tablas y Matplotlib[11] para realizar los diagramas.

5.1. Obtención de las tablas

Entonces comenzamos haciendo uso de la herramienta de VizieR, escribiendo el siguiente comando para buscar en todo VizieR las tablas que contengan en su nombre la palabra "$star$" o la palabra "$quasar$" y se encuentren en un rango de 10 arcominutos de la galaxia seleccionada.

```
./vizierquery.sh -m40.0998875,-8.435208333 -sstar -r10 -dyes
```

Código 19: Comando usado para ejecutar el programa vizierquery para la búsqueda de estrellas.

```
./vizierquery.sh -m40.0998875,-8.435208333 -squasar -r10 -dyes
```

Código 20: Comando usado para ejecutar el programa vizierquery para la búsqueda de quásares.

Este programa, además de generar el fichero txt con las tablas, va a generar una carpeta en cada caso con todas las tablas correspondientes. Estas carpetas hay que limpiarlas, debido a que Astquery a veces falla a la hora de solicitar ciertas tablas en VizieR.

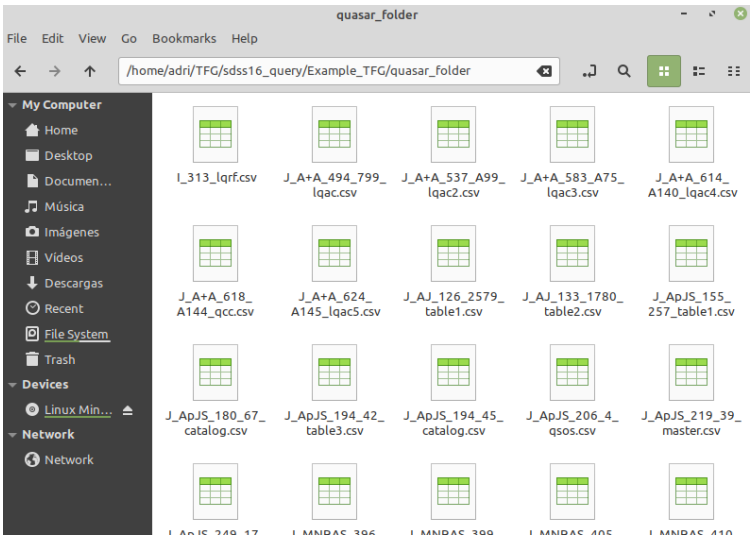

Figura 8: Carpeta con las tablas que contienen información sobre los quásares.

Estos archivos presenta una extension fits, el siguiente paso es transformar estas tablas a formato csv y juntarlas en una única sola para poder compararla con los objetos del SDSS.

```
for i in $(ls -C1); do
asttable $i >> "$i.csv"
done

for i in *.fits.csv; do
    mv -- "$i" "${i%.fits.csv}.csv"
done

rm *.fits
for i in  $(ls -C1); do
    cat $i | tr -s ' ' ',' | sed 's/.$//g' >> NGC1042_stars_merged.csv
done
```

Código 21: En este fragmento de código se cambian las tablas de fits a csv y se borran las tablas fits. Posteriormente se juntan todas las tablas csv para tener una tabla con toda la información. La razón de trabajar en csv en lugar de fits es porque a la hora de realizar las graficas en python es mucho mas sencillo.

Y con esto, se obtienen todas las tablas juntas tanto para estrellas como para quásares. Ahora con el programa que realiza la petición SQL al SDSS obtenemos una tabla con las magnitudes PSF y las coordenadas.

```
./sqlsdss --sqlquery="SELECT TOP 500000 p.ra, p.dec, psfMag_u, psfMag_i
    , psfMag_z, psfMag_r, psfMag_g FROM fGetNearbyObjEq
    (40.0998875,-8.435208333,10) n, PhotoPrimary p WHERE n.objID=p.
    objID" --format=csv
```

Código 22: Se ejecuta el programa sqlsdss con la petición SQL correspondiente.

Una vez obtenidas esta tabla del SDSS ya tenemos lo necesario para compararlas, esto se realiza usando un software de Gnuastro conocido como Astmatch, el cual permite comparar tablas por las columnas indicadas en torno a un cierto rango. Este hecho es muy importante porque cuando trabajamos con coordenadas en diferentes bases de datos puede darse que para un objeto los decimales de las coordenadas cambien ligeramente.

```
astmatch --ccol1=1,2 --ccol2=1,2 --aperture=2 NGC1042_star_merged.txt
    sloanpsf.txt --outcols=a1,a2,b3,b4,b5,b6,b7,b8
astmatch --ccol1=1,2 --ccol2=1,2 --aperture=2 NGC1042_quasar_merged.txt
     sloanpsf.txt --outcols=a1,a2,b3,b4,b5,b6,b7,b8
```

Código 23: En estos comandos comparamos las tablas obtenidas previamente con la tabla del SDSS. De esta manera comprobamos cuáles objetos de las tablas iniciales se encuentran en esta base de datos. Con –aperture seleccionamos el rango y con –outcols seleccionamos qué columnas de cada tabla queremos obtener en la tabla de respuesta.

El resultado de Astmatch genera una tabla con los objetos que se encuentran en ambas tablas, es decir, la tabla respuesta en cada caso serán las estrellas y los quásares que se encuentran en el SDSS. Estas tablas respuestas contienen como columnas las coordenadas, el id y las magnitudes PSF.

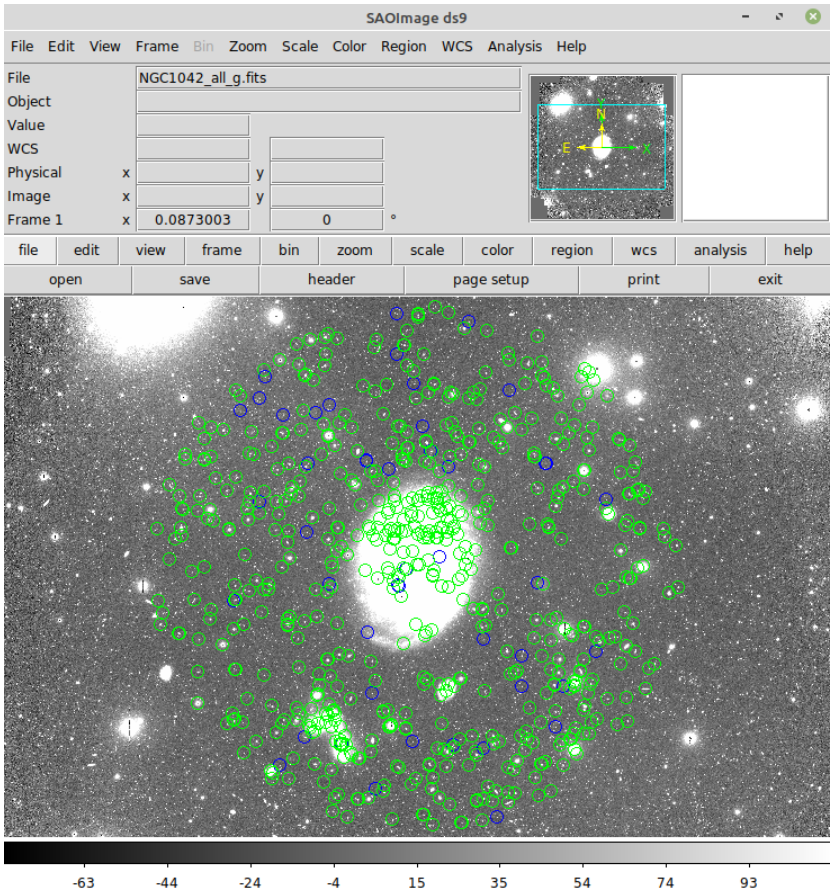

Figura 9: Con la ayuda del programa ds9[12] se pueden vizualizar las tablas de objetos en la imagen. En verde se tienen las estrellas y en azul lo quásares.

5.2. Diagramas

El siguiente paso era realizar el diagrama color-color [g-r] vs [r-i] , se esperaba obtener un resultado similar al obtenido en la figura 8 del artículo The IAC Stripe 82 Legacy Project: a wide-area survey for faint surface brightness astronomy(Jürgen Fliri & Ignacio Trujillo, 2017)[8] Sin embargo, en este caso, se estaba trabajando con muchos menos datos por lo que estos diagramas no iban a estar tan bien definidos.

Lo primero que se tenía que hacer era importar las tablas

```
import pandas as pd
quasar=pd.read_csv(r"C:\Users\Usuario\Documents\data\TFG\
    quasar_definitivo.csv")
star=pd.read_csv(r"C:\Users\Usuario\Documents\data\TFG\stars_definitivo
    .csv")
```

Código 24: Las tablas se leen y se guardan en python como DataFrames usando la librería Pandas.

y después generar dos nuevas columnas (tanto en la tabla de estrellas como en la de quásares) en la que guardemos los valores [g-r] y [r-i]

```
star["g-r"]=(star["psfMag_g"]-star["psfMag_r"])
star["r-i"]=(star["psfMag_r"]-star["psfMag_i"])
quasar["g-r"]=(quasar["psfMag_g"]-quasar["psfMag_r"])
quasar["r-i"]=(quasar["psfMag_r"]-quasar["psfMag_i"])
```

Código 25: De esta manera se generan las dos nuevas columnas para cada tabla.

Para la creación de estos diagramas se usó Python presentando las estrellas y los quásares en una misma gráfica representándolos con una figuras distintas.

```
import matplotlib.pyplot as plt

plt.scatter(star["g-r"], star["r-i"], marker='^', color='teal', label='
    Stars')
plt.scatter(quasar["g-r"], quasar["r-i"], marker='.', color='navy',
    label='Quasars')
plt.legend()
plt.xlim(-1,2)
plt.xlabel('[g-r]')
plt.ylabel('[r-i]')
```

Código 26: Ejemplo de uso de Matplotlib usado para la realización de los diagramas.

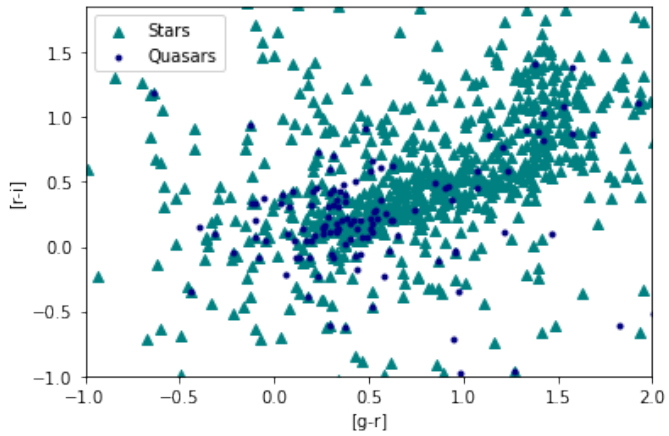

Figura 10: Diagrama de color [g-r] frente a [r-i] para el total de estrellas y quásares.

Este diagrama muestra mucha dispersión pues no se han establecido cortes en las magnitudes de los objetos. Para ver claramente la estructura del diagrama color-color es necesario hacer los siguientes cortes en la banda r de la magnitud PSF.

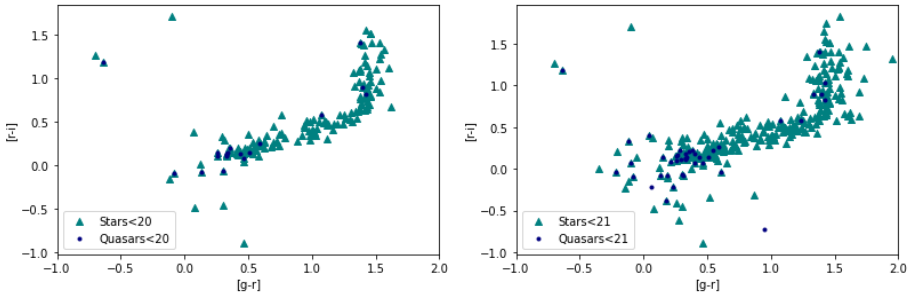

Figura 11: Diagramas de color [g-r] frente a [r-i] para los cortes en magnitud [r] 20 y 21. Esta es la estructura que se esperaba obtener.

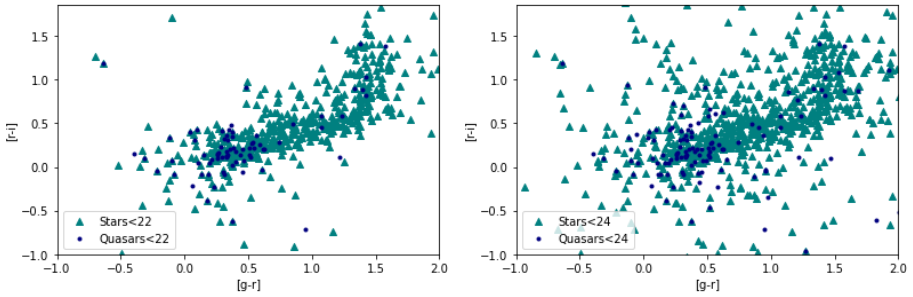

Figura 12: Diagramas de color [g-r] frente a [r-i] para los cortes en magnitud [r] 22 y 24.

Ahora tenemos que seleccionar una serie de estrellas de los diagramas para obtener sus imágenes y espectros. Vamos a seleccionar estrellas fuera del brazo, estrellas de la parte inicial del brazo y de la parte final de este de la figura en corte de magnitud r< 20.

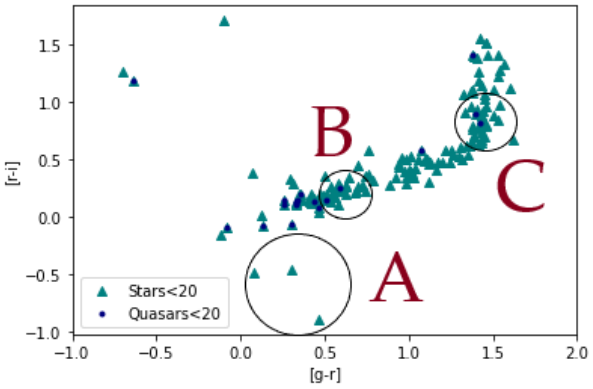

Figura 13: Regiones de donde se van a seleccionar estrellas al azar para mostrar sus imágenes y espectros.

Este diagrama color-color es característico de las estrellas, debido a esto se espera que los quásares caigan fuera del brazo, aún así se puede observar la presencia de quásares dentro de este. Por tanto, se tuvo que realizar un análisis manual de los quásares de la imagen r<20.

Gracias a este análisis se detectó que la mayoría de estos aparecen en el fichero generado anteriormente en tablas de candidatos a quásares. Debido a esto y a que el propio SDSS no da una respuesta clara en su documentación para estos objetos no se puede confirmar que sean quásares. El único objeto que si entra dentro de una clasificación de quásares (QSOs selection from SDSS and WISE (Richards+, 2015) [15]) es el siguiente:

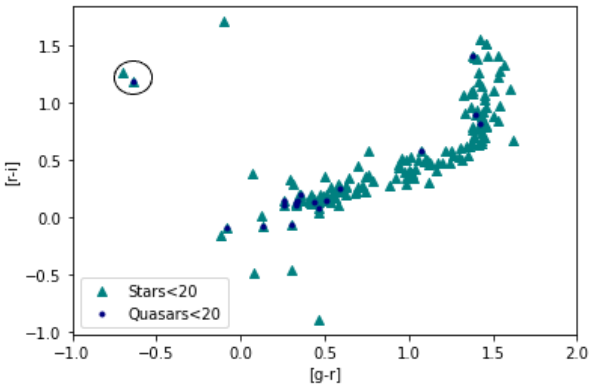

Figura 14: Único quasar encontrado en el diagrama.

5.3. Obtención de imágenes y espectros

Exportamos las tablas con los objetos seleccionados y ahora se aplica tanto el programa de macro imagen como el programa de macro espectro.

	ra	dec	psfMag_u	psfMag_i	psfMag_z	psfMag_r	psfMag_g	g-r	r-i
357	40.079852	-8.448030	19.64676	20.55348	20.14961	19.66443	20.13108	0.46665	-0.88905
389	40.087242	-8.416749	19.47431	19.79556	20.17868	19.31354	19.39295	0.07941	-0.48202
541	40.125076	-8.442512	15.22170	15.71768	13.26105	14.95422	13.91306	-1.04116	-0.76346
449	40.100298	-8.471429	18.32147	15.69675	15.56294	15.92857	16.61780	0.68923	0.23182
497	40.112515	-8.561424	21.68876	19.69850	19.50054	19.92327	20.52071	0.59744	0.22477
818	40.220330	-8.411297	18.99020	16.00131	15.93263	16.22154	16.96341	0.74187	0.22023
84	39.989475	-8.311841	21.79384	16.40574	15.88153	17.36499	18.82296	1.45797	0.95925
333	40.073279	-8.443306	20.26935	15.00992	14.47891	15.97951	17.52005	1.54054	0.96959
580	40.141551	-8.380667	22.87507	18.07345	17.68317	19.03724	20.40476	1.36752	0.96379

Figura 15: Tabla de estrellas seleccionadas.

Aplicando el siguiende comando a la tabla de las estrellas:

```
./sdss16imaging_macro -fstars.txt -c2,3 -dstars_imaging -uurlstars
```

Código 27: Uso del programa de imagen macro para obtener las imágenes de los quásares seleccionadas.

Se genera la carpeta con las imágenes de los objetos que se encuentran en la tabla:

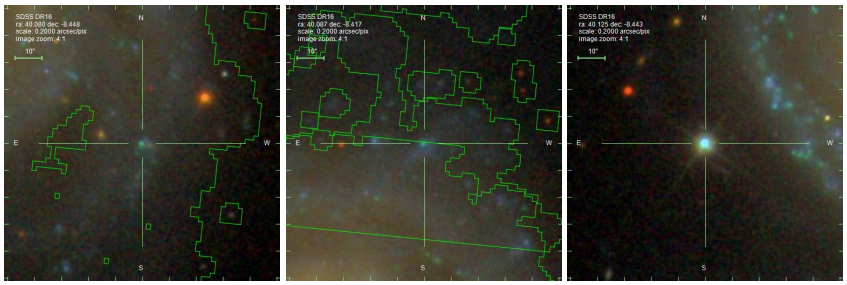

Figura 16: Imágenes de la estrellas obtenidas con el programa macro imagen la región A. Las dos primeras imágenes corresponden a regiones de formación estelar mientras que la tercera parece una estrella sobresaturada.

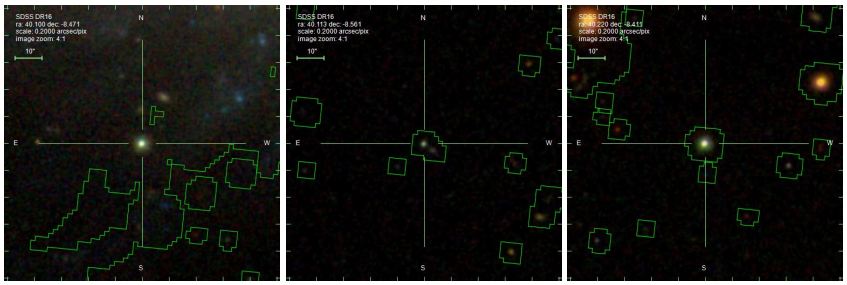

Figura 17: Imágenes de las estrellas obtenidas con el programa macro imagen en la región B.

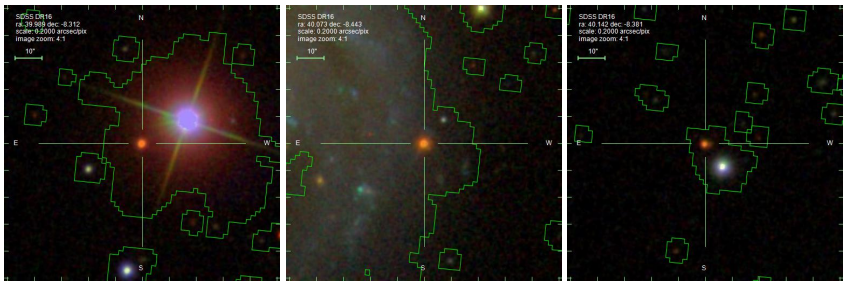

Figura 18: Imágenes de las estrellas obtenidas con el programa macro imagen la región C. En esta región se esperaba obtener estrellas del tipo enanas rojas.

Con respecto al quasar, simplemente se tiene que aplicar el programa de imagen:

```
./sdss16imaging -c"40.217485051,-8.474281511" -oquasar -lGOL
```

Código 28: Comando usado para lanzar el programa imagen.

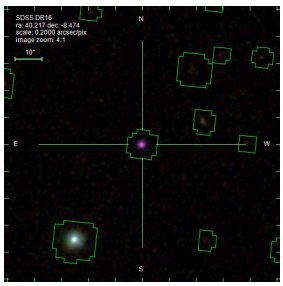

Figura 19: Imagen del quasar obtenido con el programa del SDSS.

En cuanto a los espectros, solo uno de los objetos tiene asignado un espectro. Esto se debe a que los objetos alrededor de NGC1042 son tan débiles que el SDSS tiene problemas a la hora de medir su espectro. Aquí se puede ver la importancia del fichero

que se genera con el enlace a las URL del SkyServer, gracias a este se pudo analizar los objetos uno a uno y observar si presentaban espectros o no.

```
(1)40.079852,-8.44803
http://skyserver.sdss.org/dr16/en/tools/explore/Summary.aspx?ra=40.079852&dec=-8.44803
(2)40.087242,-8.416749
http://skyserver.sdss.org/dr16/en/tools/explore/Summary.aspx?ra=40.087242&dec=-8.416749
(3)40.12507559245,-8.44251248684
http://skyserver.sdss.org/dr16/en/tools/explore/Summary.aspx?ra=40.12507559245&dec=-8.44251248684
(4)40.100298,-8.471429
http://skyserver.sdss.org/dr16/en/tools/explore/Summary.aspx?ra=40.100298&dec=-8.471429
(5)39.989475,-8.311841
http://skyserver.sdss.org/dr16/en/tools/explore/Summary.aspx?ra=39.989475&dec=-8.311841
(6)40.073279,-8.443306
http://skyserver.sdss.org/dr16/en/tools/explore/Summary.aspx?ra=40.073279&dec=-8.443306
(7)40.141551,-8.380667
http://skyserver.sdss.org/dr16/en/tools/explore/Summary.aspx?ra=40.141551&dec=-8.380667
```

Figura 20: Estructura del fichero con las URL al Skyserver

Para obtener el único espectro en los objetos lo conseguimos lanzando el programa de la siguiente manera

```
./sdssspectrum --objid=513494502204270592
```

Código 29: Uso del programa sdssspectrum para obtener el espectro.

que genera la siguiente imagen con el nombre de su correspondiente specobjID:

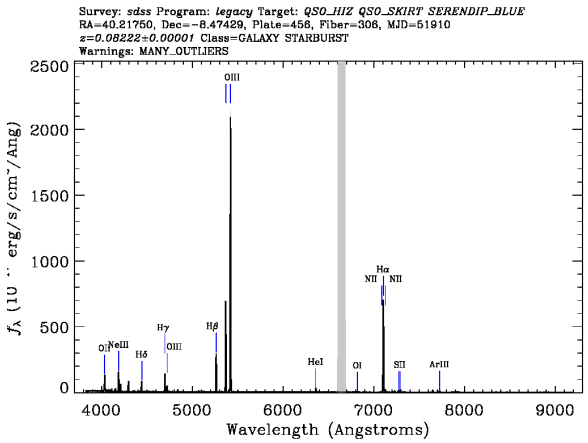

Figura 21: Espectro obtenido con el programa del SDSS.

Como se ha visto durante todo el ejemplo práctico si se acompaña estas herramientas con comandos de Bash por consola se consigue agilizar mucho el proceso puesto que desde esta podemos tanto visualizar como manejar los documentos y no hace falta acceder a las carpetas.

6. Conclusión

In the conclusion the main problems that the databases present are discussed, these problems are mainly the lack of universality when requesting information. Astquery intends to solve this problem, to generate an interface where we can request the information from any database and that the response always presents the same structure. Also complementing it with all the other tools present in gnuastro makes it a very versatile and powerful software when it comes to working with astronomical data.

A lo largo de este año he estado en contacto con bastantes bases de datos y con la información que estas aportan. Me he percatado de que el problema que estas bases de datos es el mismo, se han preocupado de obtener muchísima información pero no de como la presentan. Tienen que mejorar la manera en la que presentan esta información para automatizar las búsquedas y que no sea tan manual.

Es un proceso difícil para una base de datos puesto que no pueden dedicarle el tiempo necesario a cada diminuta región del espacio para obtener la calidad adecuada. Por eso es importante el tratado posterior a la información recibida, algunas bases de datos optan por no tratar la información y simplemente dan los datos tomados por sus telescopio y otras si deciden darle un sentido a los datos obtenidos. En este último aspecto sí que se puede afirmar que el tratado de las bases de datos deja mucho que desear, pues estas bases de datos que tratan la información (como el propio SDSS que intenta una clasificación de los objetos identificados) lo hacen de manera no supervisada que actualmente y basado en mi experiencia no es muy precisa. Hay que tener en cuenta que todo proceso no supervisado va evolucionando y siendo más completo a medida que pasa el tiempo y van corrigiendo estos datos.

Esto plantea un dilema a la hora de querer acceder a esta información: es mejor tener la mayor cantidad de datos posibles o lo mejor es asegurarnos que la información que solicitemos sea de calidad. Aquí es donde entra la comunidad científica puesto, que es la principal encargada de tratar la información y darle un sentido. Sin embargo, todo esto presenta el mismo problema: ni se puede acceder a las bases de datos de una manera universal ni el trato de los datos se hace bajo unos estándares. Esto se puede ver reflejado en VizieR, la mayoría de las tablas presentan diferentes nombres para sus diferentes columnas por lo que dificulta la obtención de los datos.

Sin embargo, VizieR era hasta ahora el único proyecto que intentaba poner una solución a ambos problemas dado que, desde su página web, podemos acceder a una gran cantidad de tablas y bases de datos. Asimismo, gracias a los UCD que VizieR asigna podemos buscar las columnas sin importar el nombre que el autor haya indicado. Aun así, esto está aún incompleto puesto que no están todas las bases de datos y los UCD presentan fallos en algunas tablas.

Por eso creo que, tanto las herramientas desarrolladas durante este TFG como Astquery, son fundamentales para la obtención de datos. Hemos generado las bases de un software por el cual podremos acceder en un futuro a todas las bases de datos y tablas existentes además de que, tanto la entrada como la salida, presentarán la misma estructura independientemente de a cuál base de datos solicitemos la información. También con la herramienta de VizieR podemos solicitar la información de un región por lo que

lo hace aún mas completo. No solo eso sino que con las demás herramientas dentro de Gnuastro se ha desarrollado un entorno de trabajo astronómico ideal para trabajar desde un único sitio.

Estos programas desarrollados serán incluidos en Gnuastro una vez se le otorgue la entrada y salida estándar de Astquery, al ser un software libre y funcionar de la manera que funciona, estoy seguro de que esta herramienta va a estar en constante evolución y se va a convertir en un referente en el campo del Big Data dentro de la astronomía.

Referencias

[1] https://www.gnu.org/software/gnuastro/

[2] Brian Kernighan & Dennis Ritchie, *The C Programming Language*, 1978.

[3] https://curl.se/libcurl/

[4] https://www.sdss.org/

[5] https://vizier.u-strasbg.fr/viz-bin/VizieR-2

[6] https://aladin.u-strasbg.fr/

[7] https://portswigger.net/burp

[8] Fliri, Jürgen & Trujillo, Ignacio, *The IAC Stripe 82 Legacy Project: a wide-area survey for faint surface brightness astronomy*, 2016.

[9] https://jupyter.org/

[10] https://pandas.pydata.org/

[11] https://matplotlib.org/

[12] https://sites.google.com/cfa.harvard.edu/saoimageds9

[13] https://gea.esac.esa.int/archive/

[14] https://stackoverflow.com/

[15] https://ui.adsabs.harvard.edu/abs/2015yCat..22190039R/abstract


```
./sdss16 sql --center=40.0998875,-8.435208333 --radius=10 --columns=p.
    ra, p.dec, psfMag_u, psfMag_i, psfMag_z, psfMag_r, psfMag_g
```