Integración de variantes genómicas en estructuras de proteínas

Diego Alberto Arenivar Díaz
Máster universitario de Bioinformática y Bioestadística
Bioinformática Clínica

Guerau Fernández Isern

5 de junio del 2018

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Integración de variantes genómicas en estructuras de proteínas*
Nombre del autor:	*Diego Arenivar Díaz*
Nombre del consultor/a:	*Guerau Fernández Isern*
Nombre del PRA:	*Maria Jesús Marco Galindo*
Fecha de entrega (mm/aaaa):	06/2018
Titulación:	*Máster universitario de Bioinformática y Bioestadística*
Área del Trabajo Final:	*Bioinformática Clínica*
Idioma del trabajo:	*Castellano*
Palabras clave	*Proteína, VCF y PDB*

Resumen del Trabajo:

En las proteínas, pueden presentarse variaciones genómicas que producen cambios aminoacídicos, y este fenómeno suele asociarse con problemas de salud graves, como trastornos en el desarrollo neuronal, enfermedades en el sistema digestivo, enfermedades cardiovasculares, cáncer, entre otros muchos padecimientos. Con los archivos en formato VCF (por sus siglas en inglés *Variant Call Format*) o Formato de Llamada Variante es sencillo interpretar cuáles han sido las variantes genómicas por posición para una secuencia determinada, es por eso, que en este trabajo se pretenden localizar tales variantes a nivel tridimensional directamente utilizando proteínas disponibles en el Banco de Datos de Proteínas o PDB (Del inglés *Protein Data Bank*). Es por eso que se creó una aplicación web montada en un servidor HTTP Apache con acceso público controlado, en la cual es posible ingresar mediante HTML + PHP el archivo VCF con los cambios aminoacídicos para la proteína PDB, y determinar la desviación media cuadrática (RMSD) entre la proteína original y la proteína mutada utilizando PyMOL como lenguaje base. Los resultados es posible visualizarlos en 3D y en imágenes PNG con acercamientos a las mutaciones realizadas, además de la generación de un archivo de alineamiento CLUSTALW. La herramienta realizada es de uso amigable sencilla de interpretar. Definitivamente este pudiese ser el arranque de un grupo de herramientas que puede ser ofrecida a los facultativos, quienes serán los usuarios principales de este esfuerzo.

Abstract:

In the big world of proteins, it is possible to present genomic variations which cause aminoacidic changes, this phenomenon can be associated with health problems, such as neuronal disorders, digestive system diseases, cardiovascular diseases, cancer, and a lot more health conditions. With the VCF (Variant Call Format) files, it's kind of easy to interpret what have been the genomic variations by position for a particular sequence. So, for this reason, in this project it is intended to locate the genomic variants within the 3D recreated structures directly using the PDB (Protein Data Bank) proteins available. Due this information, we created a web application mounted on Apache HTTP server with controlled public access, where it is possible to inject using HTML + PHP the VCF file with the amino acid changes for the PDB protein, and then be able to get the root mean square deviation (RMSD) between the original protein and the mutated protein using PyMOL as the main language. The mutated protein can be visualized in 3D and all the mutations can be analyzed with the zoomed images for the mutated residues. In addition, the system generates an alignment file in CLUSTALW file format. The tool is user friendly and easy to interpret. Definitely, this can be the beginning of a bigger project with a set of tools that can be offered to the physicians, which are going to be the main users.

Índice

Lista de figuras

No se encuentran elementos de tabla de ilustraciones.

1. Introducción

1.1 Contexto y justificación del Trabajo

Hay una gran importancia en esta investigación, comenzando por tener a las proteínas (Macromoléculas biológicas) como protagonistas. Éstas realizan ni más ni menos que las funciones de un organismo, pueden actuar como transportadoras, reguladoras, de defensa, funciones estructurales, enzimáticas, etc.

Ahora bien, teniendo en cuenta diversas investigaciones, se suelen asociar las variantes genómicas (Variaciones en la secuencia del ADN) en centros activos o de interacción de proteínas (De igual forma en centros catalíticos) con problemas de salud graves, como trastornos en el desarrollo neuronal, enfermedades en el sistema digestivo, enfermedades cardiovasculares, cáncer, entre muchos otros padecimientos.

Es de suma importancia continuar colaborando con la ciencia, desarrollando herramientas para facilitar diagnósticos rápidos (Como este trabajo final), para así tomar acción en tratamientos que logren tratar, y por qué no curar enfermedades relacionadas con este fenómeno.

Se estudiaron diversos métodos para culminar este trabajo. Se buscaron los medios para formular una solución de una manera funcional y amigable al usuario (Parte vital, ya que esta herramienta se pretende sea atractiva y funcional para los facultativos.). Esto incluye la búsqueda de los recursos adecuados para su elaboración (Entorno UNIX, lenguaje de programación, instalación de servidores, interfaces de bases de datos como SQL, y PHP y demás lenguajes para la interfaz de usuario) yendo de la mano con los conceptos de Biología Molecular como se menciona al inicio de este apartado introductorio.

1.2 Objetivos del Trabajo

El objetivo general de este TFM es el desarrollar una herramienta computacional para uso de facultativos la cual permita mapear la variabilidad genómica a nivel de estructura terciaria pudiendo determinarse su distancia respecto a los distintos dominios descritos para la proteína.

Los objetivos específicos para la culminación de este TFM, se enlistan a continuación.

- Definir los recursos que serán necesarios para la culminación del TFM, de acuerdo a la minuciosa investigación de las bases teóricas. Incluyendo sistemas operativos, lenguajes de programación para el funcionamiento principal de la solución, instalación de servidores, interfaces de bases de datos, lenguajes de programación la interfaz para el usuario.

- Creación del diseño para la integración de cada uno de los módulos recopilados en el proceso de investigación de fundamentos teóricos.

- Llevar a cabo la construcción del prototipo.

- Elaborar casos de prueba para detección de defectos de la solución integral.

- Redacción de la memoria.

- Presentación y defensa pública.

1.3 Enfoque y método seguido

Dado a que fue un proyecto meramente individual, se fueron repasando cada uno de los objetivos con sus respectivas tareas, verificando el mayor número de fuentes de información posibles, prueba y error, y por supuesto, comunicar al director de la TFM sobre los avances que se fueron llevando a lo largo del semestre.

1.4 Planificación del Trabajo

En resumen, la planificación de este TFM se muestra en la figura 1.4.1.

Mes	Marzo		Abril				Mayo				Junio			
Semana	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Tarea 1: Investigación de recursos técnicos necesarios verificando su funcionamiento.	X	X	X	X	X									
Tarea 2: Elaborar diagramas de bloques y/o diagramas de flujo que reflejen el objetivo principal.			X	X	X									
Tarea 3: Trabajar con el código fuente con todos los componentes siguiendo el diseño definido.				X	X	X	X	X	X					
Tarea 4: Utilizar diversos tipos de proteínas al inyectar las variantes de aminoácidos							X	X	X					
Tarea 5: Redactar la memoria.								X	X	X	X			
Tarea 6: Elaboración de Presentación.											X			

Desarrollo del trabajo (Fase 1)

Desarrollo del trabajo (Fase 2)

Cierre de la memoria

Elaboración de la presentación

Defensa pública

Figura 1.4.1. Calendario de planificación del TFM

Como es posible analizar en el calendario, se trabajaron básicamente en seis grandes tareas, las cuales sus recursos, se desglosan a continuación, (La descripción detallada de los recursos técnicos se analizará más a fondo en capítulos posteriores).

- Equipo de cómputo que cumpla con las expectativas de poder instalar una máquina virtual y funcionar sin ningún problema. En específico fue usada un ordenador Lenovo ThinkPad (Intel Core i5-5200U con 16 GB de memoria RAM con 64-bit de Sistema Operativo).
- Acceso a Internet.
- Máquina Virtual.
- Imagen nueva de sistema operativo Linux para ser instalado en la máquina virtual.
- Documentación y acceso a ejemplos para lenguajes de programación candidatos.
- Paquetería de Microsoft Office.
- Acceso a bibliotecas.
- Correo electrónico.

1.5 Breve sumario de productos obtenidos

El producto obtenido principal es una página web con acceso público controlado. En éste, será posible ingresar el nombre de una proteína PDB y un archivo VCF que contendrá el detalle de las mutaciones ocurridas. Al llamar la aplicación principal, después de hacer el respectivo procesamiento, se obtendrá la visualización en 3D de la proteína mutada, así como algunos datos técnicos de la mutación (Incluyendo imágenes).

1.6 Breve descripción de los otros capítulos de la memoria

El resumen de cada capítulo principal que se detallarán más a fondo en la sección 2 es el siguiente:

- Sistema operativo.

Se explicarán las razones de haber seleccionado el sistema operativo *Linux Mint* 17.3 (versión 3.19 de 64 bits). Incluyendo, además como subtemas las máquinas virtuales y la descripción de la que fue utilizada.

- Instalación de servidor para brindar servicios de acceso remoto a por HTTP.

Se explicará brevemente sobre HTTP, para ir de lleno a la selección de un servidor Apache (Versión 1.8.13), después se estudiará la forma de comunicación entre el usuario y el servidor será a través de HTML y PHP. Posteriormente se identificó un servicio de renta de URL público (*Ngrok Tunnel*) para los servicios de acceso remoto, de vital importancia, dado que esta herramienta cuenta con acceso público controlado a través de la Internet.

- Selección de lenguaje de programación.

Al establecer la comunicación entre usuario y servidor, se hizo un análisis detallado de selección de lenguaje principal para lograr el funcionamiento deseado. Se hablará de R y PyMOL y cómo paulatinamente se fue utilizando PyMOL como lenguaje principal de la solución.

- Aplicación principal: Mutador de Aminoácidos para Proteínas a través archivos VCF.

Por supuesto que el corazón de esta presente memoria es la aplicación final, que cuenta con la integración de los recursos de software investigados y seleccionados. Se detallarán cada uno de sus puntos ya contando con una base sólida de los capítulos anteriores. En este capítulo se incluyen los resultados finales.

- Creación de nuevas herramientas.

Se hablará de algunas nuevas herramientas que fueron creadas y que tienen un rol vital en la aplicación final.

2. Resto de capítulos

2.1 Archivos VCF y PDB.

2.1.1 Archivos en formato VCF.

Abreviación de Formato de Llamada Variante (Del inglés *Variant Call Format*). Éstos son archivos en formato de texto que registran la información de posiciones en el genoma o en otras palabras, la variabilidad [1].

Un ejemplo de un archivo VCF tomado de *1000Genomes* sería el siguiente:

```
##fileformat=VCFv4.0
##fileDate=20090805
##source=myImputationProgramV3.1
##reference=1000GenomesPilot-NCBI36
##phasing=partial
##INFO=<ID=NS,Number=1,Type=Integer,Description="Number of Samples With Data">
##INFO=<ID=DP,Number=1,Type=Integer,Description="Total Depth">
##INFO=<ID=AF,Number=.,Type=Float,Description="Allele Frequency">
##INFO=<ID=AA,Number=1,Type=String,Description="Ancestral Allele">
##INFO=<ID=DB,Number=0,Type=Flag,Description="dbSNP membership, build 129">
##INFO=<ID=H2,Number=0,Type=Flag,Description="HapMap2 membership">
##FILTER=<ID=q10,Description="Quality below 10">
##FILTER=<ID=s50,Description="Less than 50% of samples have data">
##FORMAT=<ID=GT,Number=1,Type=String,Description="Genotype">
##FORMAT=<ID=GQ,Number=1,Type=Integer,Description="Genotype Quality">
##FORMAT=<ID=DP,Number=1,Type=Integer,Description="Read Depth">
##FORMAT=<ID=HQ,Number=2,Type=Integer,Description="Haplotype Quality">
#CHROM POS     ID        REF ALT    QUAL FILTER INFO                              FORMAT      NA00001        NA00002        NA00003
20     14370   rs6054257 G   A       29   PASS   NS=3;DP=14;AF=0.5;DB;H2           GT:GQ:DP:HQ 0|0:48:1:51,51 1|0:48:8:51,51 1/1:43:5:.,.
20     17330   .         T   A       3    q10    NS=3;DP=11;AF=0.017               GT:GQ:DP:HQ 0|0:49:3:58,50 0|1:3:5:65,3   0/0:41:3
20     1110696 rs6040355 A   G,T     67   PASS   NS=2;DP=10;AF=0.333,0.667;AA=T;DB GT:GQ:DP:HQ 1|2:21:6:23,27 2|1:2:0:18,2   2/2:35:4
20     1230237 .         T   .       47   PASS   NS=3;DP=13;AA=T                   GT:GQ:DP:HQ 0|0:54:7:56,60 0|0:48:4:51,51 0/0:61:2
20     1234567 microsat1 GTCT G,GTACT 50   PASS   NS=3;DP=9;AA=G                    GT:GQ:DP    0/1:35:4       0/2:17:2       1/1:40:3
```

Es posible observar en el primer renglón que fue remarcado de forma sencilla que para el cromosoma 20 (Columna CHROM) en la posición 14370 (Columna POS) habrá un cambio de codón de ADN G (Columna REF) a un codón ADN A (Columna ALT).

2.2.1 Archivos en formato PDB.

Antes de entrar a los detalles de esta clase de archivos, se hará un breve repaso sobre conceptos básicos de estructuras de proteínas [2].

Las proteínas se pueden presentar en cuatro niveles de estructura:

La estructura primaria, donde es posible encontrar el orden de unión de los aminoácidos. A continuación, se muestra un ejemplo de tal estructura:

```
SKGEELFTGVVPILVELDGDVNGHKFSVSGEGEGDATYGKLTLKFICTTGKLPVPWPTLVTTLVQCFARY
PDHMKQHDFFKSAMPEGYVQERTIFFKDDGNYKTRAEVKFEGDTLVNRIELKGIDFKEDGNILGHKLEYN
YNHHKVYITADKQKNGIKVNFKTRHNIEDGSVQLADHYQQNTPIGDGPVLLPDNHYLHTHSKLSKDPNEK
RDHMVLLEFVTAAGITL
```

La estructura secundaria, en la cual se forman los hélices alfa y hojas beta de las cadenas primarias. En la figura 2.2.1.1 se muestra un ejemplo de tales conformaciones.

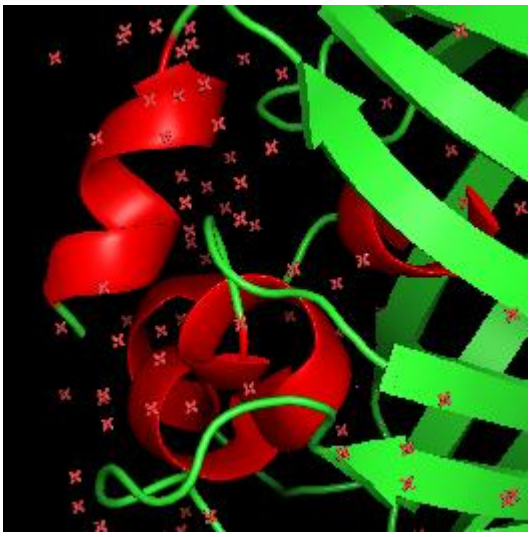


Figura 2.2.1.1. Ejemplo de estructura secundaria (Hélices Alfa en rojo y Hojas Beta en verde).

La estructura terciara (figura 2.2.1.2), en la cual las estructuras secundarias toman una forma y se mueven de alguna manera que definirá el funcionamiento de la proteína.

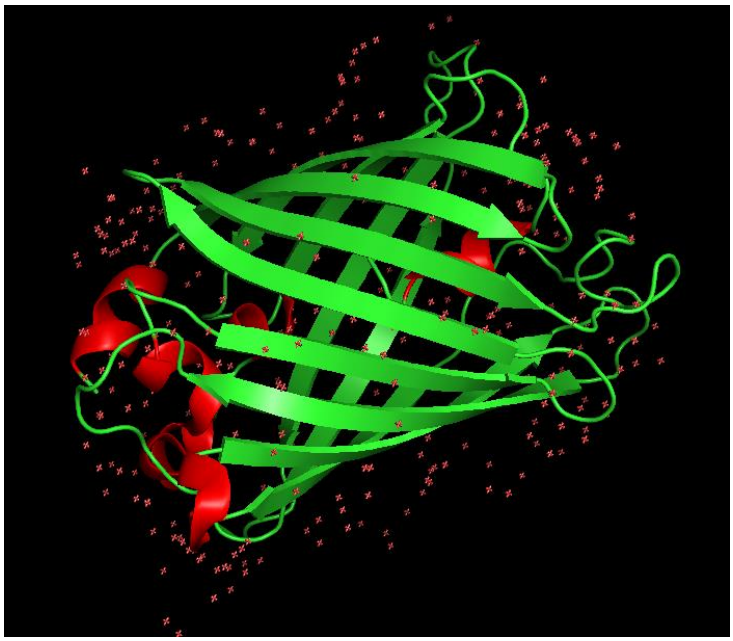


Figura 2.2.1.2. Ejemplo de estructura terciaria

La estructura cuaternaria, que son agrupaciones que algunas veces se forman de estructuras terciarias de proteínas. En la figura 2.2.1.3 es posible observar como ejemplo la hemoglobina.

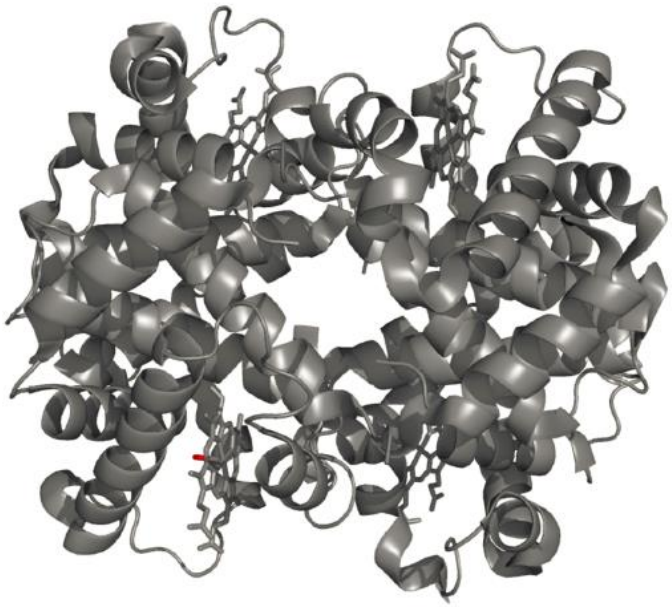


Figura 2.2.1.3. Ejemplo de estructura terciaria

Es importante mencionar, que las proteínas dependen de sus estructuras 3D, ya que sus diversas conformaciones espaciales específicas definen su función biológica.

Los archivos PDB, que es una abreviación de Banco de Datos Proteicos (Del inglés *Protein Data Bank*). Esta clase de archivos se utiliza para indicar coordenadas atómicas. Destacando el tipo de *record* ATOM, el cual cuenta con toda la información referente a la ubicación de una proteína en un plano cartesiano de tres ejes (En otras palabras, en la estructura terciaria). En la figura 2.2.1.4 se muestra la descripción del tipo de *record* ATOM [3].

Protein Data Bank Format: Coordinate Section				
Record Type	**Columns**	**Data**	**Justification**	**Data Type**
ATOM	1-4	"ATOM"		character
	7-11[#]	Atom serial number	right	integer
	13-16	Atom name	left[*]	character
	17	Alternate location indicator		character
	18-20[§]	Residue name	right	character
	22	Chain identifier		character
	23-26	Residue sequence number	right	integer
	27	Code for insertions of residues		character
	31-38	X orthogonal Å coordinate	right	real (8.3)
	39-46	Y orthogonal Å coordinate	right	real (8.3)
	47-54	Z orthogonal Å coordinate	right	real (8.3)
	55-60	Occupancy	right	real (6.2)
	61-66	Temperature factor	right	real (6.2)
	73-76	Segment identifier[¶]	left	character
	77-78	Element symbol	right	character

Figura 2.2.1.4. Descripción de tipo de *record* ATOM para un archivo PDB.

Es posible comprobar tal descripción con un ejemplo real (1L02) de un archivo extraído directamente del sitio oficial de PDB (https://www.rcsb.org/).

```
ATOM      1  N   MET A   1      36.698 -24.858   8.976  1.00 23.62           N
ATOM      2  CA  MET A   1      36.907 -23.464   9.049  1.00 17.66           C
ATOM      3  C   MET A   1      35.642 -22.815   9.564  1.00 21.43           C
ATOM      4  O   MET A   1      34.601 -23.320   9.263  1.00 19.13           O
ATOM      5  CB  MET A   1      37.255 -22.976   7.652  1.00 21.23           C
ATOM      6  CG  MET A   1      37.612 -21.494   7.684  1.00 44.42           C
ATOM      7  SD  MET A   1      39.347 -21.148   7.356  1.00 38.63           S
ATOM      8  CE  MET A   1      40.139 -22.333   8.439  1.00 44.56           C
ATOM      9  N   ASN A   2      35.757 -21.740  10.358  1.00 20.14           N
ATOM     10  CA  ASN A   2      34.657 -20.980  10.923  1.00  7.62           C
ATOM     11  C   ASN A   2      35.049 -19.496  10.928  1.00 20.59           C
ATOM     12  O   ASN A   2      36.199 -19.202  10.583  1.00  9.57           O
ATOM     13  CB  ASN A   2      34.302 -21.541  12.339  1.00  7.93           C
ATOM     14  CG  ASN A   2      35.425 -21.490  13.303  1.00 14.55           C
ATOM     15  OD1 ASN A   2      35.905 -20.379  13.534  1.00 13.98           O
ATOM     16  ND2 ASN A   2      35.757 -22.628  13.948  1.00 10.19           N
ATOM     17  N   ILE A   3      34.131 -18.625  11.327  1.00 10.38           N
ATOM     18  CA  ILE A   3      34.344 -17.185  11.415  1.00 14.39           C
ATOM     19  C   ILE A   3      35.605 -16.799  12.226  1.00 17.39           C
ATOM     20  O   ILE A   3      36.375 -15.917  11.787  1.00 14.21           O
ATOM     21  CB  ILE A   3      33.073 -16.378  11.795  1.00 13.39           C
ATOM     22  CG1 ILE A   3      33.271 -14.850  11.610  1.00  8.74           C
ATOM     23  CG2 ILE A   3      32.644 -16.706  13.221  1.00 10.54           C
ATOM     24  CD1 ILE A   3      33.660 -14.471  10.140  1.00  6.97           C
```

Como un antecedente, al ser revelada la estructura terciaria de la Mioglobina (1MWC para PDB) a través de una técnica de cristalografía de rayos X, los científicos Max Preutz y John Cowdery Kendrew ganaron el premio nobel de química de 1962 [2]. En la figura 2.2.1.5 se muestra la estructura terciaria de la Mioglobina.

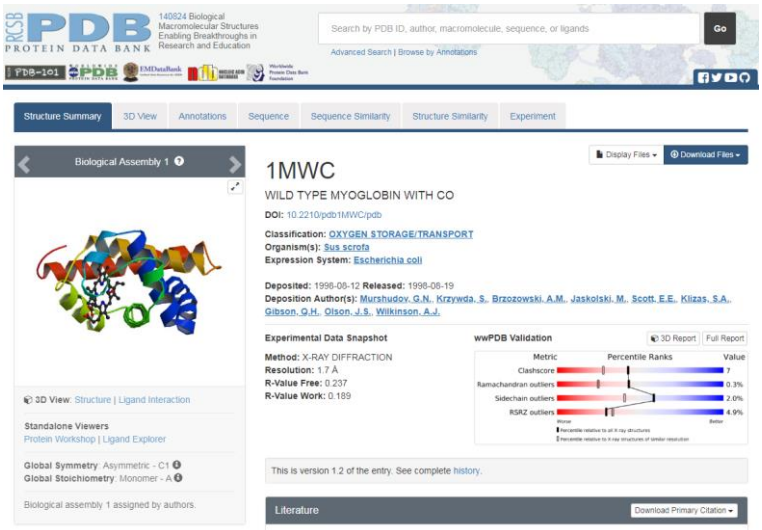


Figura 2.2.1.5. Estructura terciaria de la Mioglobina desde el Banco de Datos de Proteínas.

2.3 Sistema operativo.

2.3.1 *Linux Mint*.

La intención del sistema operativo *Linux Mint* [4] es brindar una solución completa que incluye una interfaz de usuario sencilla, como se muestra en la figura 2.3.1.1, y cuente con las fortalezas de la colaboración global del Software Libre (Que además cuide la economía) y aplicaciones poderosas y seguras. En este proyecto, se pretende lograr una herramienta web para una solución bioinformática con acceso público controlado. Por lo tanto, este sistema operativo indicado para lograrlo.

Figura 2.3.1.1. Escritorio de Sistema Operativo Linux Mint 17.3 (Versión 3.19).

2.3.2 *VMware Workstation Pro*.

Dado que se pretende instalar el sistema operativo de Linux, y se cuenta con una computadora con el sistema operativo Windows como base, se determinó que la solución *VMware Workstation Pro* [5] sería la elegida para brindar los servicios de virtualización. En la figura 2.3.1.2 se muestra la máquina virtual instalada para el proyecto, remarcando el encabezado que indica que es se utiliza una solución *VMware*. A su vez, en la figura 2.3.1.3 es posible observar la pantalla principal indicando la máquina virtual instalada llamada *Ubuntu 64-bit* (Puede ser llamada de la forma que se deseé).

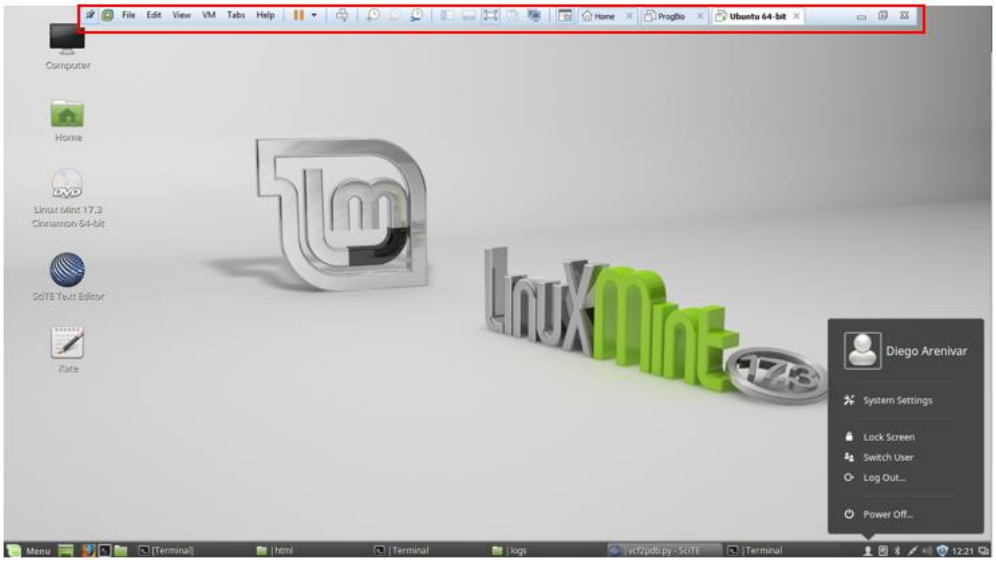


Figura 2.3.1.2. Escritorio de Sistema Operativo *Linux Mint* 17.3 (Versión 3.19) indicando el encabezado de *VMware*.

Figura 2.3.1.3. Pantalla principal de *VMware,* indicando la máquina virtual que se encuentra corriendo (*Ubuntu 64-bit*).

2.4 Instalación de servidor para brindar servicios de acceso remoto por HTTP.

2.4.1 Protocolo HTTP.

HTTP, por sus siglas en inglés (*Hypertext Transfer Protocol*) significa Protocolo de Transferencia de Hipertexto, éste se encarga brindar pauta de cómo es que la información es transmitida a través del sistema WWW (*World Wide Web*) que accede la conocida Red de Redes, la Internet [6].

2.4.2 Servidor Apache.

Después de analizar que existe un protocolo para transferir información, es necesario contar con una herramienta que se encargue de cumplir con esa función. El proyecto del servidor HTTP Apache [7] es la solución para establecer el acceso externo a través del sistema operativo Linux.

Al ser instalado el servidor Apache (Versión 2.4.7) en Linux, es posible verificar que se encuentra funcionando utilizando el comando *ps* (Comando para verificar procesos ejecutándose al momento).

```
darenivar@darenivar-virtual-machine /var/www/html $ ps -ef | grep 'apache2'
root       1910      1  0 May25 ?        00:00:47 /usr/sbin/apache2 -k start
www-data  16257   1910  0 May28 ?        00:00:02 /usr/sbin/apache2 -k start
www-data  16258   1910  0 May28 ?        00:00:01 /usr/sbin/apache2 -k start
www-data  16259   1910  0 May28 ?        00:00:01 /usr/sbin/apache2 -k start
www-data  16260   1910  0 May28 ?        00:00:01 /usr/sbin/apache2 -k start
www-data  16261   1910  0 May28 ?        00:00:01 /usr/sbin/apache2 -k start
www-data  17130   1910  0 May28 ?        00:00:01 /usr/sbin/apache2 -k start
www-data  18861   1910  0 May29 ?        00:00:01 /usr/sbin/apache2 -k start
www-data  20179   1910  0 May29 ?        00:00:01 /usr/sbin/apache2 -k start
www-data  20180   1910  0 May29 ?        00:00:01 /usr/sbin/apache2 -k start
www-data  20181   1910  0 May29 ?        00:00:01 /usr/sbin/apache2 -k start
dareniv+  41430  33545  0 13:11 pts/5    00:00:00 grep --colour=auto apache2
```

Verificando la versión instalada:

```
darenivar@darenivar-virtual-machine /var/www/html $ apache2 -v
Server version: Apache/2.4.7 (Ubuntu)
Server built:   Apr 18 2018 15:36:26
```

Otra forma de comprobar que el servidor HTTP esté funcionando correctamente, definitivamente sería con una interfaz de usuario, por lo cual, en la siguiente sección, se concentrará en el análisis de esa verificación.

2.4.3 Lenguaje HTML y la verificación del servidor HTTP Apache.

HTML, por sus siglas en inglés (*HyperText Markup Language*) significa Lenguaje de Marcado de Hipertexto y es la base para la creación de páginas web[8]. El servidor HTTP Apache, puede ser accedido mediante la dirección IP definida en el servidor para la conexión de red. Por ejemplo, la dirección IP que fue definida estáticamente en la máquina virtual Linux, es 192.168.126.130 que se muestra a continuación:

```
darenivar@darenivar-virtual-machine /var/www/html $ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:39:fe:ee
          inet addr:192.168.126.130  Bcast:192.168.126.255
Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe39:feee/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1296605 errors:0 dropped:0 overruns:0 frame:0
          TX packets:883861 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:729611053 (729.6 MB)  TX bytes:169015354 (169.0 MB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:7902 errors:0 dropped:0 overruns:0 frame:0
          TX packets:7902 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:10486828 (10.4 MB)  TX bytes:10486828 (10.4 MB)
```

Al abrir un explorador externo (En este caso el sistema operativo de Windows fuera de la máquina virtual), y escribir el IP en la barra de búsqueda, un programa HTML llamado index.html es buscado desde el directorio raíz de Apache. A continuación, se muestra que existe un archivo de tales características en el directorio raíz (*/var/www/html*).

```
darenivar@darenivar-virtual-machine /var/www/html $ ls -ltr index.html
-rwxrwxrwx 1 darenivar darenivar 430 Jun  3 14:10 index.html
darenivar@darenivar-virtual-machine /var/www/html $
```

Al ejecutar el ejercicio mencionado en un navegador, el resultado se muestra en la figura 2.4.3.1.

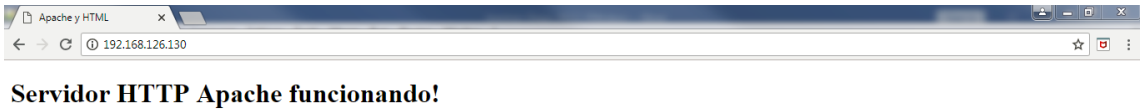


Figura 2.4.3.1. Demostración de funcionamiento de Apache más HTML.

El archivo *index.html*, contiene lo siguiente:

```
<!DOCTYPE html>

<html>
<head>
 <meta http-equiv="content-type" content="text/html;
charset=UTF-8">
 <meta charset="utf-8">

 <title>Apache y HTML</title>
 <style type="text/css">
 </style>
</head>

<body>
```

```
<h1>Servidor HTTP Apache funcionando!</h1>

</body></html>
```

2.4.4 Lenguaje PHP.

Bien definido como un lenguaje de scripting de servidores [9], PHP brinda la posibilidad de ejecutar funciones avanzadas de programación y hacer visible el trabajo a través de HTML.

En la figura 2.4.4.1, es posible observar que se agregó un sitio web *php* utilizando el símbolo "/" como separador, que indica que, a través de Apache, es posible también llamar a una página en lenguaje PHP.

Figura 2.4.4.1. Demostración de funcionamiento de Apache más PHP.

El archivo *php_test.php* (Instalado en el directorio raíz del servidor Apache), contiene lo siguiente:

```
<!DOCTYPE html>

<html>
<head>
 <meta http-equiv="content-type" content="text/html; charset=UTF-8">
 <meta charset="utf-8">

 <title>Apache y PHP</title>
 <style type="text/css">
 </style>
```

```
</head>

<body>
<h1>Servidor HTTP Apache funcionando! ... Y PHP.</h1>

<?php
echo 'Esto es PHP y trabajo en conjunto con HTML';
echo '<br><br>';

for ($i = 1; $i <= 10; $i++) {
    echo $i;
    echo '<br><br>';

}
?>

</body></html>
```

2.4.5 Túnel Seguro para publicar servidor HTTP Apache.

Teniendo ya establecida comunicación entre usuario y servidor dentro de una red local, se investigó si hay alguna forma de publicarla a la Internet. Se encontró una herramienta llamada *ngrok* [10], la cual proporciona la posibilidad de abrir el servidor local a Internet, con el único requisito de hacer la IP estática.

Existe una licencia de prueba, en la cual la dirección de Internet cambia dinámicamente cada vez que se levanta el servicio *ngrok*. Debido a que se pretendió dar acceso al director del TFM desde la etapa de pruebas, para una mayor comunicación, se decidió contratar un servicio básico en el cual es posible seleccionar un dominio siendo estático, para cuantas veces se decida reiniciar el servidor.

La sesión ngrok se muestra a continuación, donde es posible observar que se estará utilizando el dominio *diegobioinformatics.ngrok.io*:

```
ngrok by @inconshreveable                                       (Ctrl+C to quit)

Session Status                online
Account                       Diego Arenivar (Plan: Basic)
Version                       2.2.8
Region                        United States (us)
Web Interface                 http://127.0.0.1:4040
Forwarding                    http://diegobioinformatics.ngrok.io -> localhost:80
Forwarding                    https://diegobioinformatics.ngrok.io -> localhost:80

Connections                   ttl     opn     rt1     rt5     p50     p90
                              0       0       0.00    0.00    0.00    0.00
```

Utilizando el mismo ejemplo de PHP para la sección 2.4.4, ahora es posible visualizar en la figuras 2.4.5.1 y 2.4.5.2, el sitio web de prueba de forma pública controlada (Con credenciales de acceso, que es posible habilitar por *ngrok*).

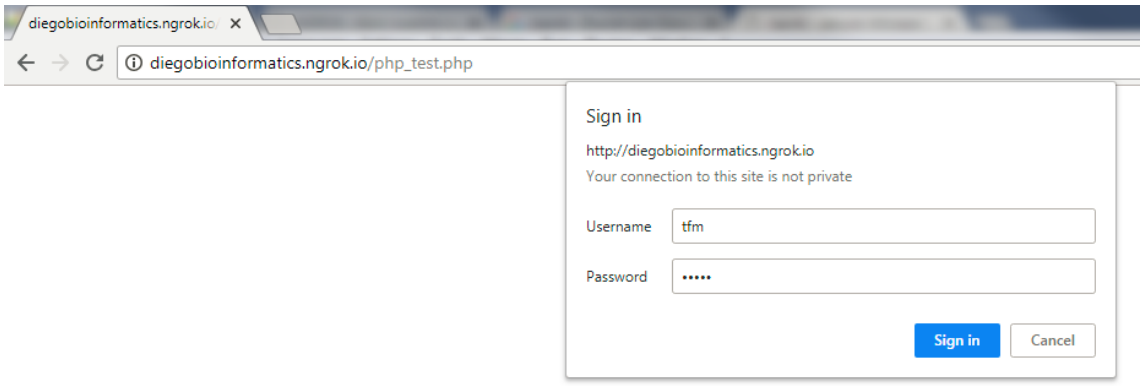


Figura 2.4.5.1. Demostración de control de acceso al acceder a la nueva página web publicada en la Internet a través de *ngrok*.

Figura 2.4.5.2. Demostración de funcionamiento de Apache más PHP en página web publicada en la Internet.

La sesión *ngrok* se muestra a continuación, muestra ya la nueva visita:

```
ngrok by @inconshreveable                                       (Ctrl+C to quit)

Session Status                online
Account                       Diego Arenivar (Plan: Basic)
Version                       2.2.8
Region                        United States (us)
Web Interface                 http://127.0.0.1:4040
Forwarding                    http://diegobioinformatics.ngrok.io -> localhost:80
Forwarding                    https://diegobioinformatics.ngrok.io -> localhost:80

Connections                   ttl     opn     rt1     rt5     p50     p90
                              1       0       0.00    0.00    5.28    5.28
```

2.5 Selección de lenguaje de programación.

2.5.1 Lenguaje R y PyMOL.

La idea inicial, fue que se estaría desarrollando la aplicación en lenguaje R [11], por el gran poder de esta solución en aplicaciones bioinformáticas y bioestadísticas, incluyendo actividades como elaborar la migración de Windows (Sistema operativo base a lo largo del Máster) a Linux completamente funcional (Incluyendo las actualizaciones de sistema correspondientes e instalaciones manuales).

Como muestra lo siguiente:

```
darenivar@darenivar-virtual-machine ~ $ R

R version 3.3.3 RC (2017-02-27 r72279) -- "Another Canoe"
Copyright (C) 2017 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Previously saved workspace restored]
```

Incluyendo la instalación de *RStudio* (Figura 2.5.1.1)

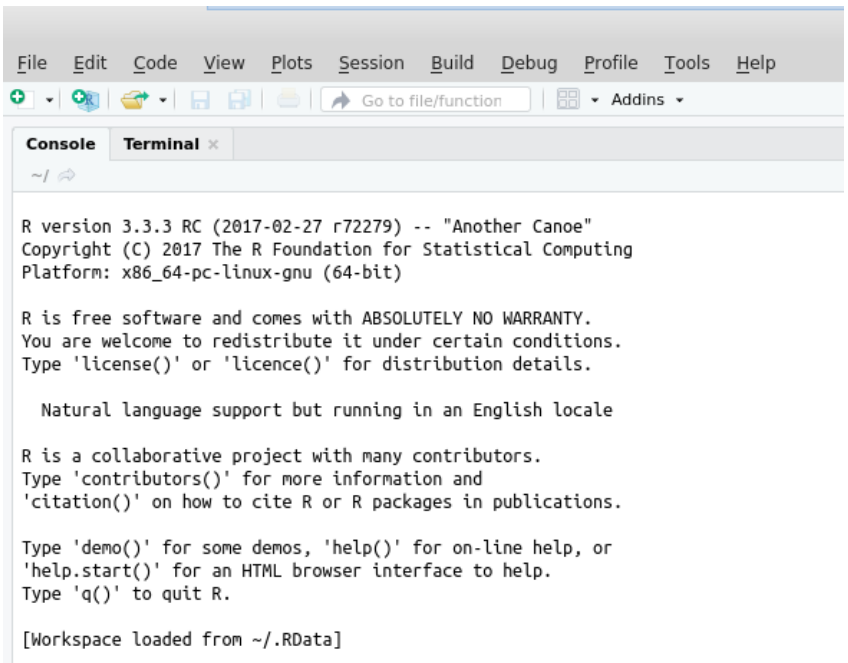


Figura 2.5.1.1. Sección de pantalla principal de *RStudio*.

Se instalaron los paquetes *VariantAnnotation* [12] para la lectura/escritura de los archivos *VCF* y *ggplot2* para graficación (Figura 2.5.1.2), así como *rpdb* [13] y bio3d [14] para lectura/escritura y visualización de archivos *PDB* (Figuras 2.5.1.3 y 2.5.1.4).

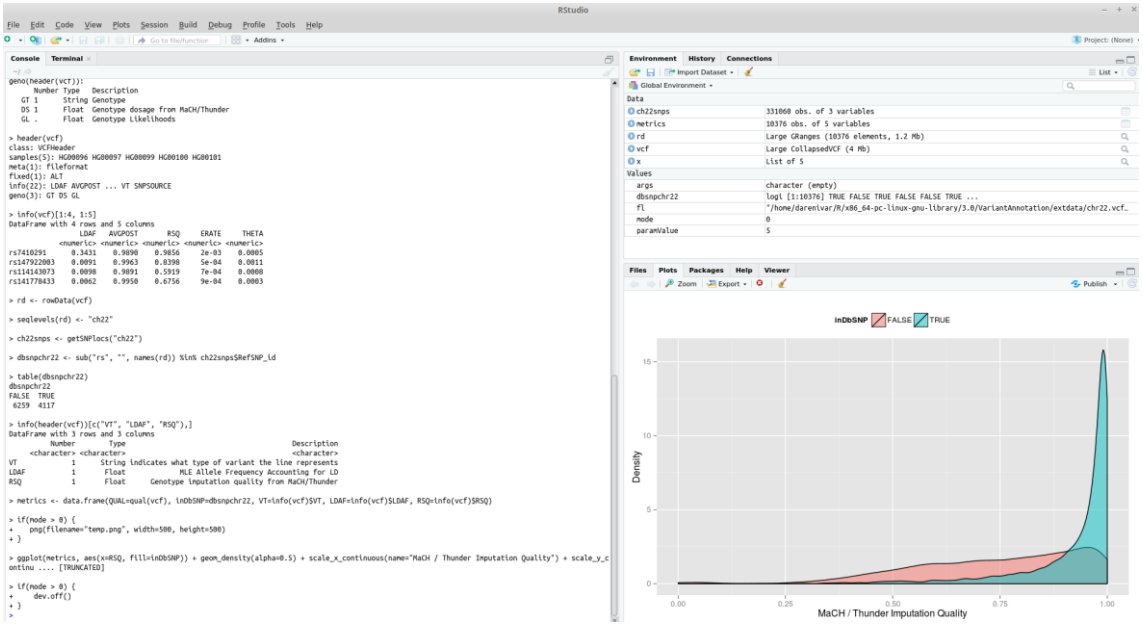


Figura 2.5.1.2. Ejemplo de aplicación desarrollada con el paquete *VariantAnnotation* y *ggplot*.

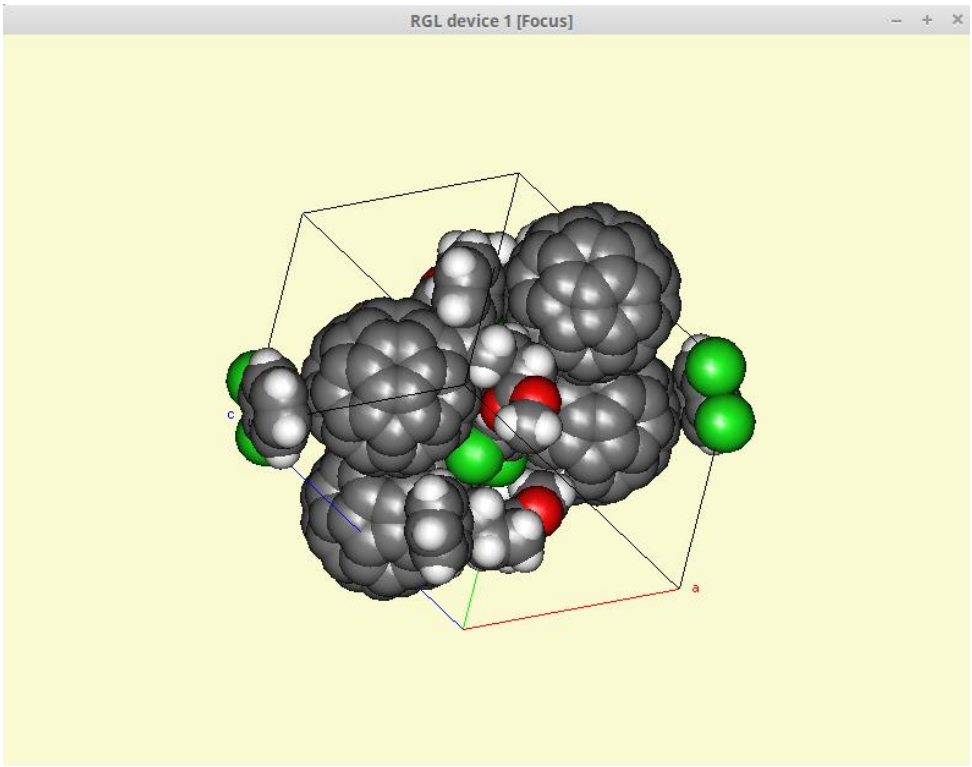


Figura 2.5.1.3. Ejemplo de aplicación desarrollada con el paquete *rpdb*.

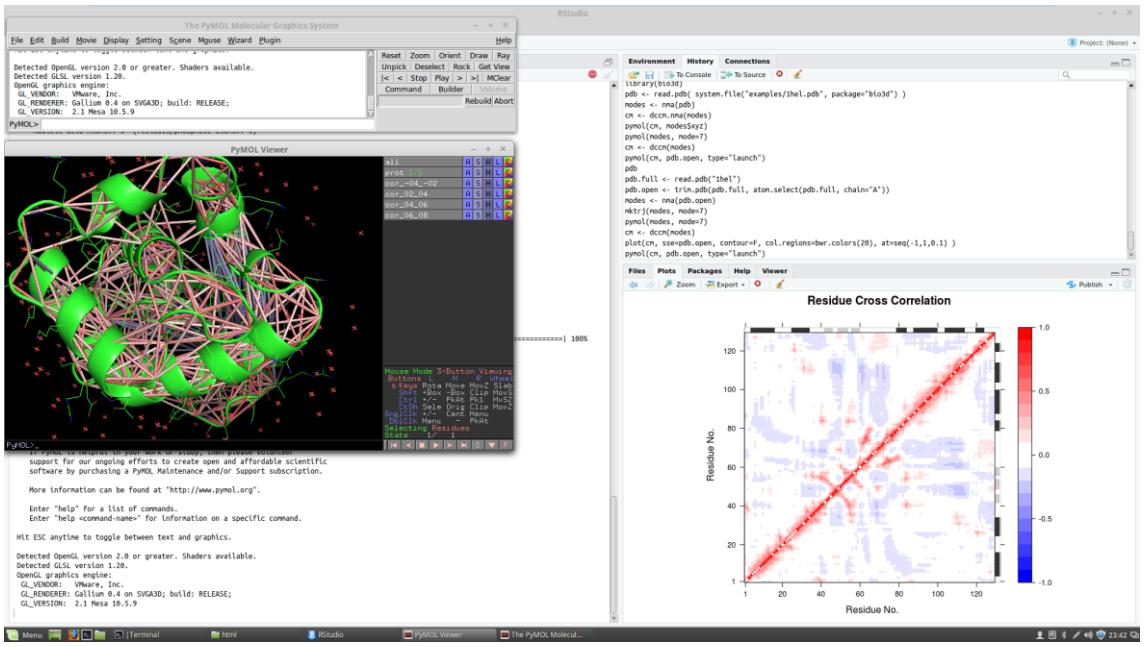


Figura 2.5.1.4. Ejemplo de aplicación desarrollada con el paquete *bio3d*.

Como es posible observar en la figura 2.5.1.4, la estructura 3D generada, es a cargo de *PyMOL* [15], que es un software de visualización molecular basado en el lenguaje Python[16], que tuvo que ser instalado adicional al paquete bio3d. Debido a que al estar desarrollando la correcta visualización de proteínas PDB en programas HTML, se fue conociendo y familiarizando más en el mundo de PyMOL, se determinó que al poder inyectar de forma libre código en lenguaje Python, era posible tanto hacer la visualización de la proteína en 3D, como la lectura y traducción de datos. Por lo tanto, se decidió simplificar los recursos utilizando solamente PyMOL y dejando de utilizar R en esta versión (Aunque quedará disponible de ser necesario para mejoras futuras). En capítulos posteriores se analizarán los detalles técnicos correspondientes.

2.5 Aplicación principal: Mutador de Aminoácidos para Proteínas a través archivos VCF.

2.5.1 Diagrama de bloques de la aplicación principal.

La aplicación principal, consolidando todos los recursos investigados a lo largo del TFM, y mencionados en apartados anteriores, se muestran en el diagrama de bloques de la figura 2.5.1.1.

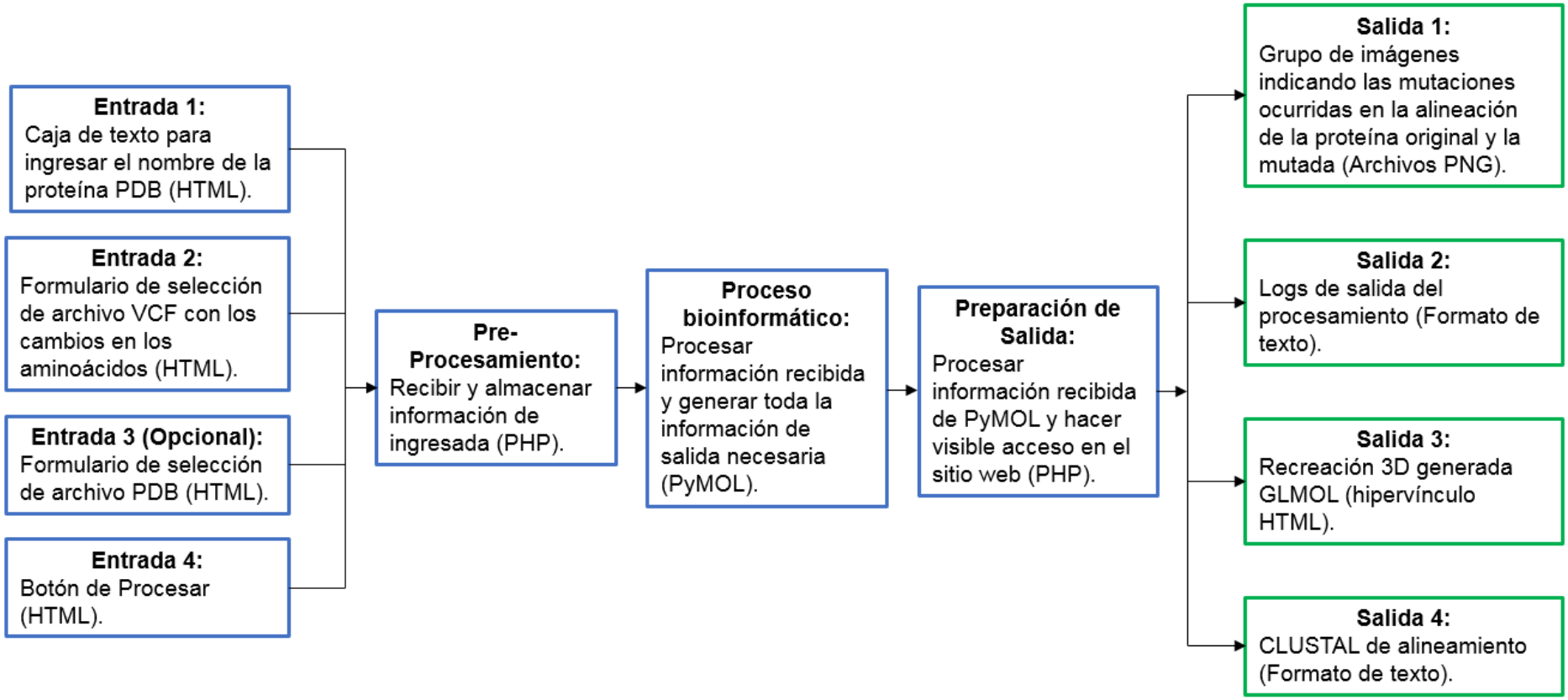


Figura 2.5.1.1. Diagrama de bloques de la aplicación principal.

2.5.2 Página de bienvenida y página principal de la aplicación.

En la figura 2.5.2.1, se muestra la página de bienvenida al sitio (index.html page

Figura 2.5.2.1. Página de bienvenida al sitio web.

Cabe mencionar, que el encabezado de la universidad es un enlace a la página principal de la misma.

El código del index.html se muestra a continuación.

```
<!DOCTYPE html>
<p><a href="https://www.uoc.edu"><img src="UOC.png" alt="UOC"></a></p>
<html>
<head>
 <meta http-equiv="content-type" content="text/html; charset=UTF-8">
 <meta charset="utf-8">

 <title>Diego UOC Bioinformatics</title>
 <style type="text/css">
 </style>
</head>

<body>
<h1>Bienvenidos al sitio de Diego para el TFM</h1>

<p>En este sitio se pretende publicar las aplicaciones, por ejemplo <a
href="vcf2pdb.php">Mutador de Aminoacidos para Proteinas a traves archivos VCF</a>
</p>

</body></html>
```

La página principal de la aplicación se muestra en la figura 2.5.2.2.

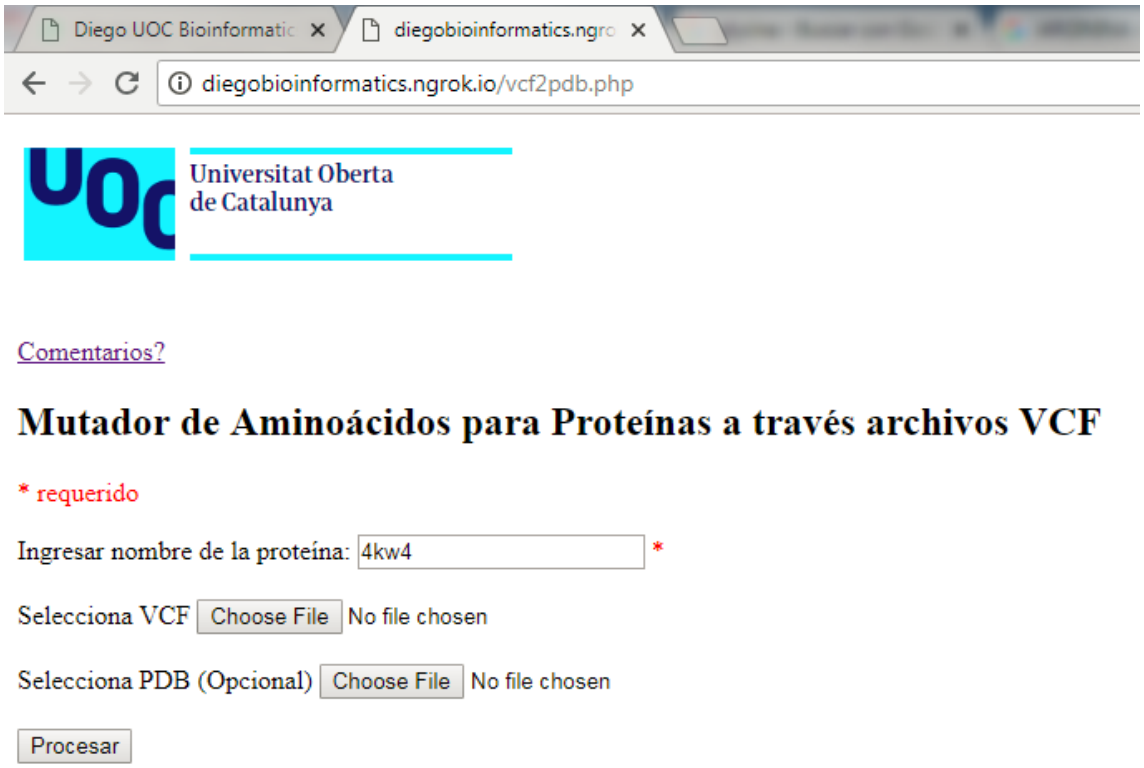


Figura 2.5.2.2. Página principal de la aplicación.

2.5.3 Entrada de datos.

2.5.3.1 Caja de texto.

En la aplicación se utilizará una caja de texto para ingresar el nombre de la proteína PDB. En la figura 2.5.3.1.1, se muestra la caja de texto.

Figura 2.5.3.1.1. Caja de texto de entrada.

La sección de código en HTML para adquirir los datos (Con una proteína por defecto) se muestra a continuación:

```
Ingresar nombre de la proteina: <input type="text" name="name" value="4kw4">
<span class="error">* <?php echo $genErr;?></span>
<br><br>
```

2.5.3.2 Formularios de selección de archivos.

En la aplicación se utilizarán dos formularios de selección de archivos, para subir el archivo VCF con los cambios en los aminoácidos y para subir el archivo PDB en caso que éste no esté en el repositorio FTP de la PDB RSCB.

En la figura 2.5.3.2.1 se muestran los dos formularios de selección de archivos.

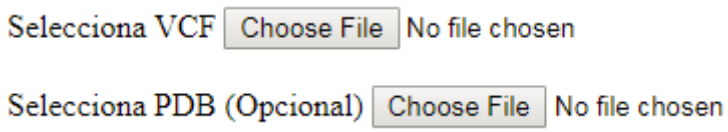


Figura 2.5.3.2.1 Formularios utilizados.

La sección de código en HTML para los archivos de los formularios se muestra a continuación:

```
<label for="file"> Selecciona VCF </label>
  <input type="file" name="file" id="file" />
  <br><br>
<label for="file"> Selecciona PDB (Opcional) </label>
  <input type="file" name="pdbfile" id="pdbfile" />
  <br><br>
```

Hay que tomar en cuenta que el atributo `enctype="multipart/form-data"` deberá estar siempre presente en la forma HTML para que el formulario funcione.

2.5.3.3 Botón de Procesar.

En la aplicación se utilizará un botón de Procesar que mandará la señal que todo está listo para la etapa de Pre-Procesamiento. En la figura 2.5.3.3.1, se muestra el botón.

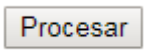


Figura 2.5.3.3.1. Botón de Procesar.

La sección de código en HTML para el botón se muestra a continuación:

```
<input type="submit" name="submit" value="Procesar">
```

2.5.4 Pre-Procesamiento.

La sección de Pre-Procesamiento consiste en recibir toda la información de entrada, aplicar algunas reglas como que el nombre de la proteína siempre deberá estar presente, y algunas preparaciones para cuando la aplicación bioinformática entre en acción. A continuación, se muestran las secciones de código PHP.

Nombre de la proteína es requerida:

```
if ($_SERVER["REQUEST_METHOD"] == "POST")
{
  if (empty($_POST["name"]))
  {
    $genErr = "Nombre de PDB es Requerido";
```

```
  }
  else
  {
    $prot      = test_input($_POST["name"]);
  }
}
```

Pasar a minúsculas el nombre del PDB:

```
    $prot = strtolower($prot);
```

Mover los archivos del formulario a un directorio en específico.

```
    $target_dir = "uploads/";
    $filename=$_FILES['file']['name'];
    $file_tmp = $_FILES['file']['tmp_name'];
    $pdbfilename=$_FILES['pdbfile']['name'];
    $pdbfile_tmp = $_FILES['pdbfile']['tmp_name'];
    move_uploaded_file($file_tmp, $target_dir . $filename);
    move_uploaded_file($pdbfile_tmp, $pdbfilename);
```

2.5.5 Proceso bioinformático.

2.5.5.1 Funcionamiento principal.

Al asignado el nombre de la proteína, y ambos archivos del formulario son almacenados, PHP mediante la función *shell_execute()* manda llamar al programa principal PyMol de la siguiente manera:

```
$salida_pymol = shell_exec('pymol -cq vcf2pdb.py -- '. $prot . '
' . $target_dir . $filename .' 2>&1');
```

El diagrama a bloques del funcionamiento completo de la aplicación se muestra en la figura 2.5.5.1.1.

En los siguientes apartados de este capítulo se mostrarán algunos ejemplos del funcionamiento de la aplicación, donde se irá detallando paso a paso lo que está sucediendo a través del proceso.

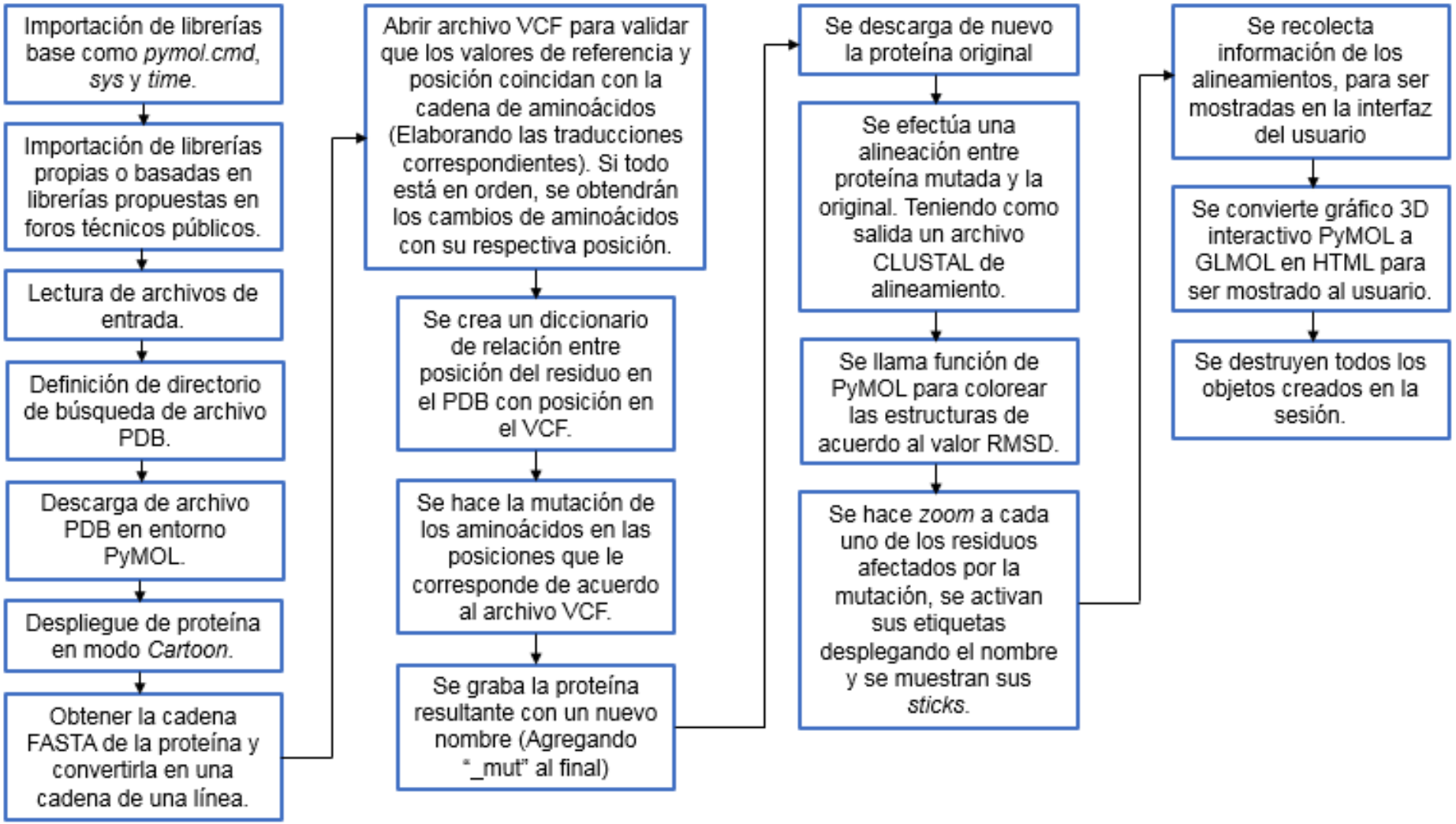


Figura 2.5.5.1.1. Diagrama de bloques del procesamiento bioinformático.

2.5.5.2 Primer Ejemplo: Mutación en cinco aminoácidos diferentes para la proteína PDB 4KW4.

El archivo VCF que indica los cambios en los aminoácidos se muestra a continuación.

```
##fileformat=VCFv4.1
##fileDate=20180519
##source=TFM
##phasing=partial
##FILTER=<ID=q10,Description="Quality below 10">
##FILTER=<ID=s50,Description="Less than 50% of samples have
data">
#CHROM  POS ID  REF ALT QUAL    FILTER  INFO    FORMAT  NA00001
NA00002 NA00003
1   3   rs001   GGT CTG 75  PASS
1   10  rs002   GTC ACC 50  PASS
1   137 rs004   CTG AAA 50  PASS
1   151 rs003   GAT CGG 50  PASS
1   210 rs005   AAA GTT 50  PASS
```

El archivo VCF viene expresado en términos de ADN, por lo tanto, es posible deducir lo siguiente:

- En la posición 3 de la secuencia, se encuentran los codones de ADN GGT, que se tratan del aminoácido Glicina (G) los cuales se mutarán en los codones de ADN CTG, que se tratan del aminoácido Leucina (L).

- En la posición 10 de la secuencia, se encuentran los codones de ADN GTC, que se tratan del aminoácido Valina (V) los cuales se mutarán en los codones de ADN ACC, que se tratan del aminoácido Treonina (T).

- En la posición 137 de la secuencia, se encuentran los codones de ADN CTG, que se tratan del aminoácido Leucina (L) los cuales se mutarán en los codones de ADN AAA, que se tratan del aminoácido Lisina (K).
- En la posición 151 de la secuencia, se encuentran los codones de ADN GAT, que se tratan del aminoácido Aspartato (D) los cuales se mutarán en los codones de ADN CGG, que se tratan del aminoácido Arginina (R).
- En la posición 210 de la secuencia, se encuentran los codones de ADN AAA, que se tratan del aminoácido Lisina (K) los cuales se mutarán en los codones de ADN GTT, que se tratan del aminoácido Valina (V).

Como bien se ha venido mencionando, el objetivo de la aplicación es determinar el efecto de tales mutaciones en un alineamiento verificando

los valores RMSD. En otras palabras, determinar de una forma sencilla y visual que tan diferente resulta ser la proteína mutada.
En la figura 2.5.5.2.1, se muestra el ingreso del archivo VCF mencionado.

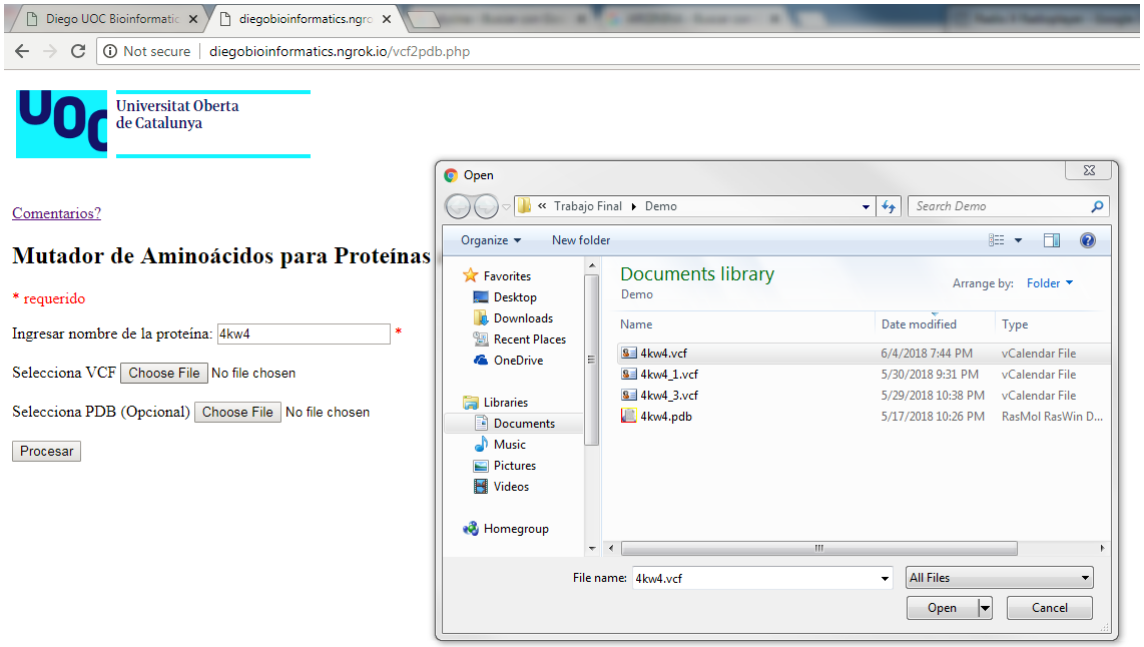


Figura 2.5.5.2.1. Ingreso de archivo VCF.

Como bien se ha venido mencionando, el objetivo de la aplicación es determinar el efecto de tales mutaciones en un alineamiento verificando los valores RMSD. En otras palabras, determinar de una forma sencilla y visual que tan diferente resulta ser la proteína mutada.

En este ejemplo, se dejará el formulario vacío para descargar el archivo PDB directamente del repositorio de PDB RCSB.

Al presionar el botón Procesar la aplicación comienza preparar los datos de salida.

Al desplegar los resultados, se muestra una breve ayuda de interpretación de los mismos (Figura 2.5.5.2.2).

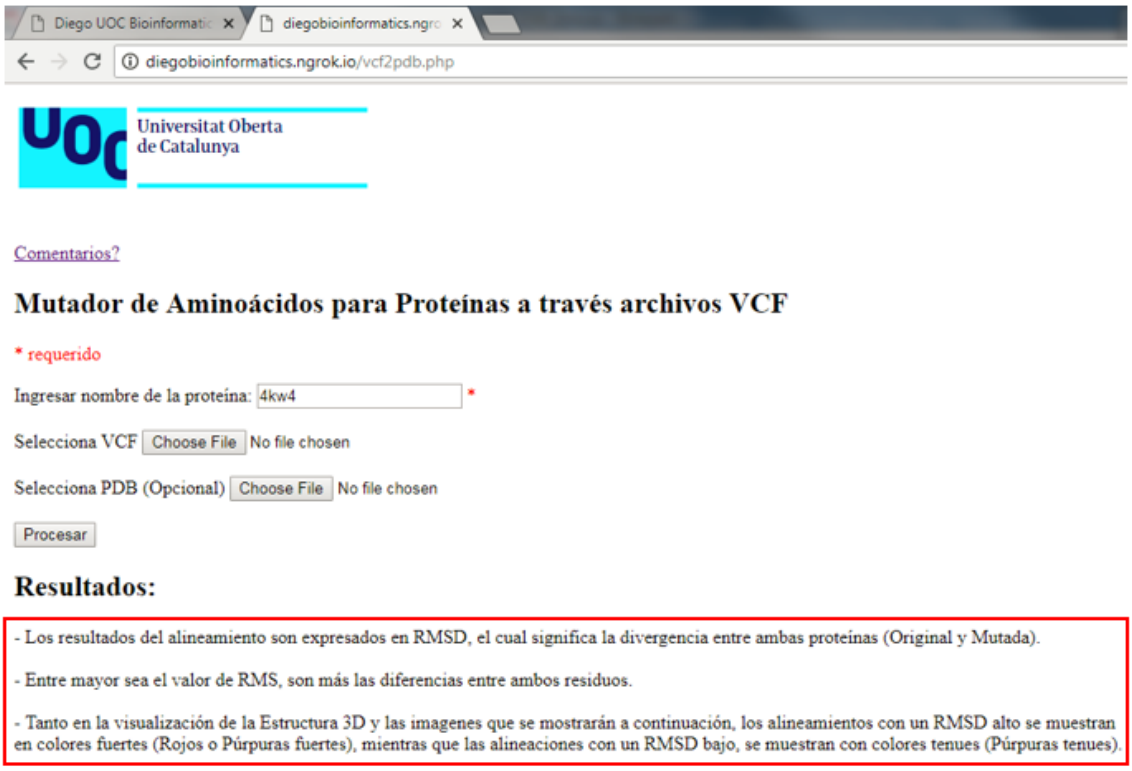


Figura 2.5.5.2.2. Ayuda en la interpretación de resultados.

La aplicación genera un hipervínculo (Figura 2.5.5.2.3) que es la visualización de la estructura 3D resultante coloreada de acuerdo al valor RMSD en HTML (Figura 2.5.5.2.4).

Estructura 3D de 4kw4 generada (Secciones diferentes a azul, indican variaciones de RMSD de proteínas)

Figura 2.5.5.2.3. Hipervínculo a estructura 3D de la proteína 4KW4 mutada definida desde el archivo VCF.

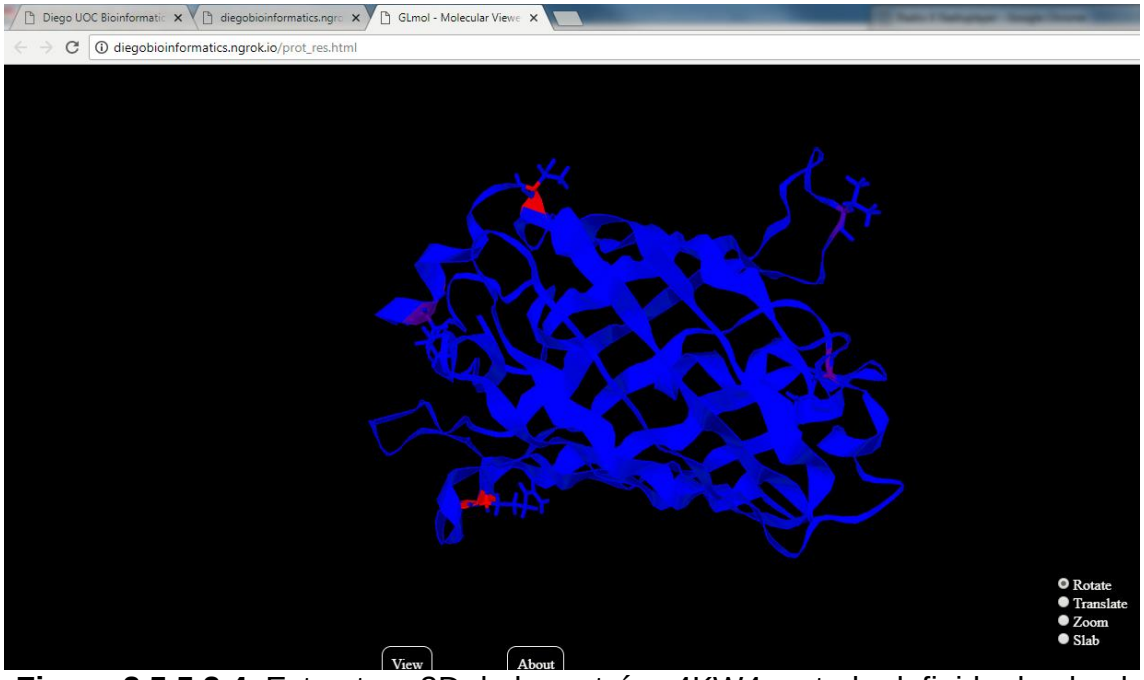


Figura 2.5.5.2.4. Estructura 3D de la proteína 4KW4 mutada definida desde el archivo VCF.

La visualización en 3D permite rotar la proteína resultante, hacer zoom, entre otras funcionalidades disponibles en GLmol (Estructura PyMOL 3D convertida para ser visualizada en HTML). En la figura 2.5.5.2.5 es posible visualizar la estructura más de cerca en diferente posición, mostrando residuo mutado con un RMSD alto en rojo.

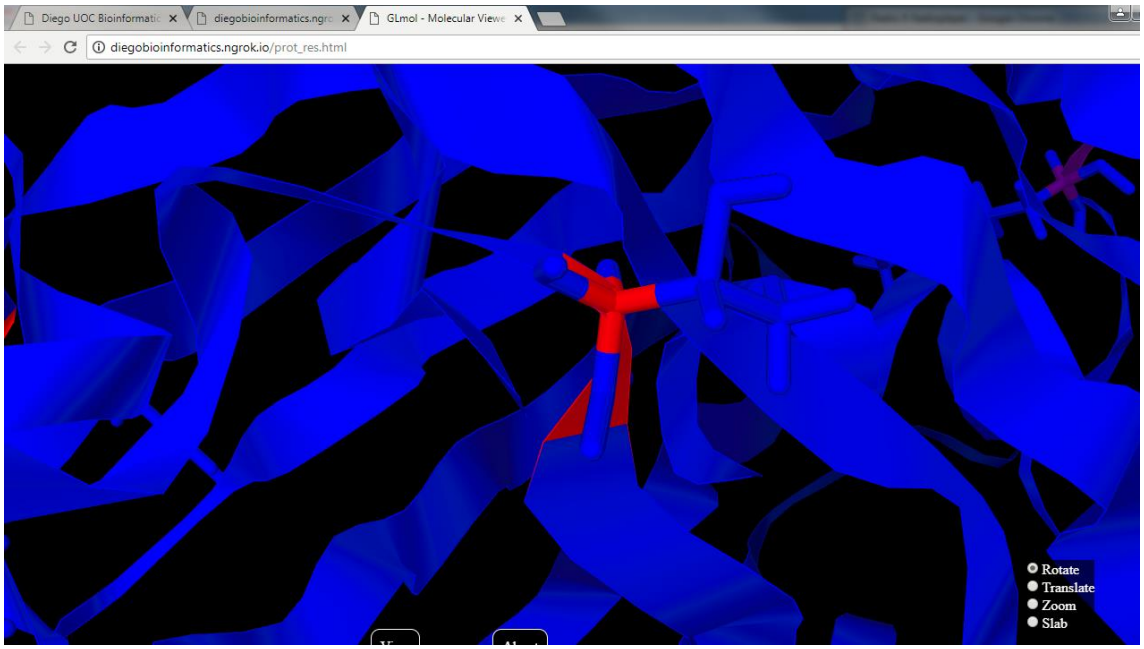


Figura 2.5.5.2.5. Estructura 3D de la proteína 4KW4 mutada desde archivo VCF mostrando de cerca residuo mutado con un RMSD alto.

La aplicación genera otro hipervínculo (Figura 2.5.5.2.6) que son los resultados del alineamiento entre proteína original y mutada en formato CLUSTAL(Figura 2.5.5.2.7).

CLUSTAL de Alineamiento de 4kw4 generado

Figura 2.5.5.2.6. Hipervínculo a resultados del alineamiento en formato CLUSTAL.

Figura 2.5.5.2.7. CLUSTAL de alineamiento.

El análisis de cada mutación efectuada a través del archivo VCF es mostrado desplegando el número de residuo para PDB y su posición el archivo VCF, los nombres del aminoácido original y mutado, y el valor del RMSD resultante. En las figuras 2.5.5.2.8 a 2.5.5.2.12 es posible visualizar el análisis de cada mutación.

Mutación resultante para residuo PDB 4 definida en posición VCF 3 (GLY->LEU).
RMSD resultante: 0.029584

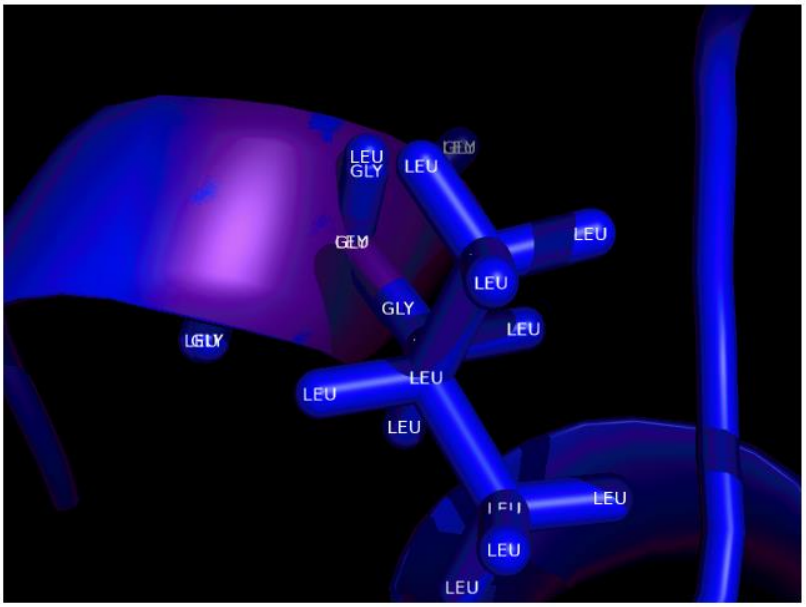


Figura 2.5.5.2.8. Posición 3 del VCF en púrpura bajo (RMSD bajo).

Mutación resultante para residuo PDB 11 definida en posición VCF 10 (VAL->THR).
RMSD resultante: 0.655897

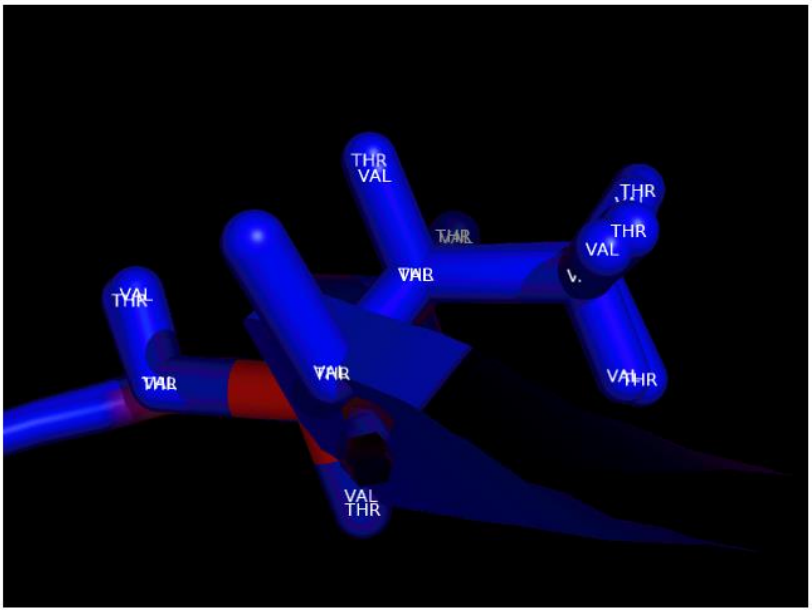


Figura 2.5.5.2.9. Posición 11 del VCF en rojo (RMSD alto).

Mutación resultante para residuo PDB 141 definida en posición VCF 137 (LEU->LYS).
RMSD resultante: 0.081539

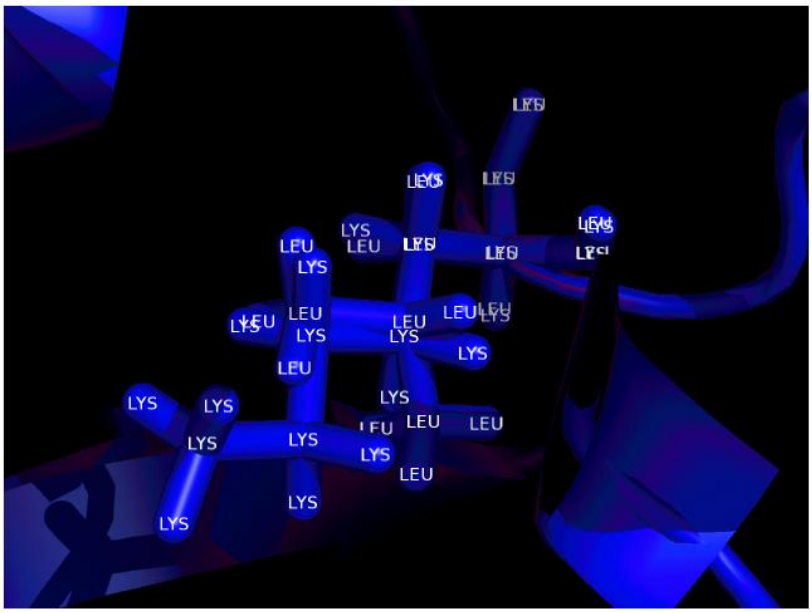


Figura 2.5.5.2.10. Posición 141 del VCF en púrpura bajo (RMSD bajo).

Mutación resultante para residuo PDB 155 definida en posición VCF 151 (ASP->ARG).
RMSD resultante: 0.560946

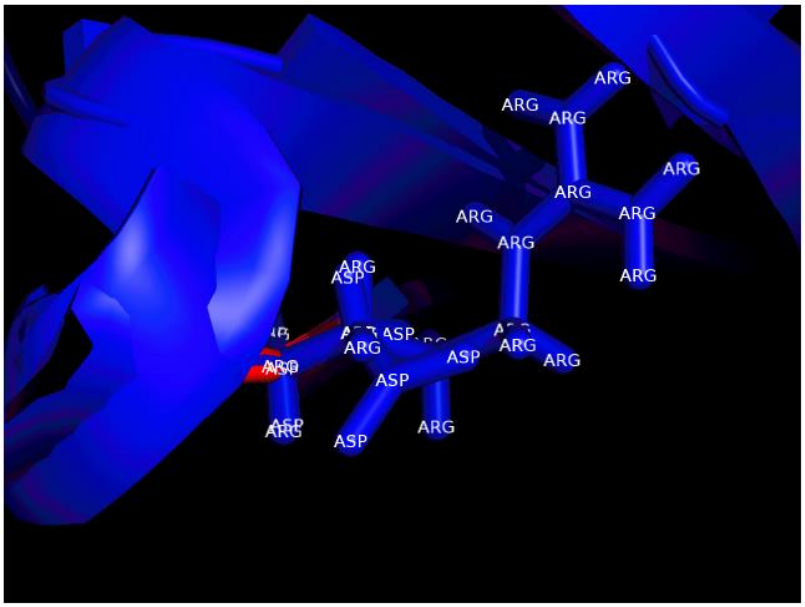


Figura 2.5.5.2.11. Posición 151 del VCF en rojo (RMSD alto).

Mutación resultante para residuo PDB 214 definida en posición VCF 210 (LYS->VAL).
RMSD resultante: 0.062309

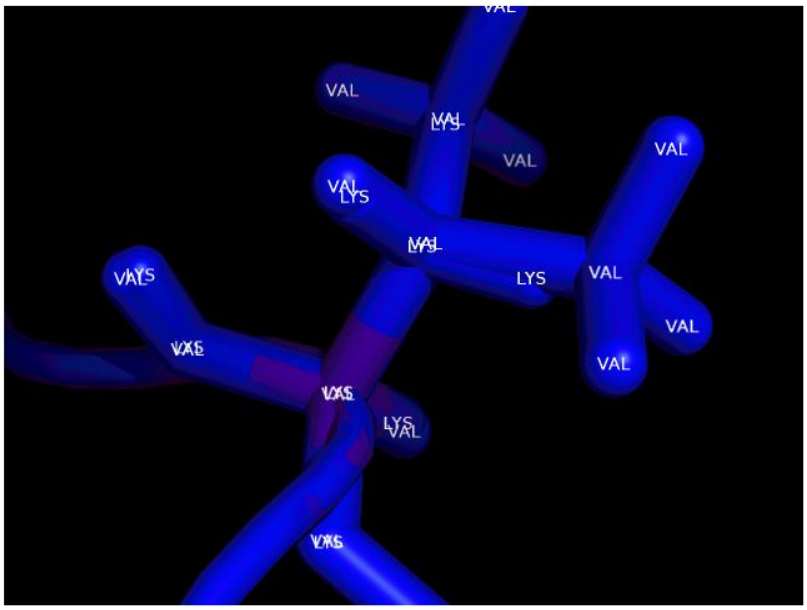
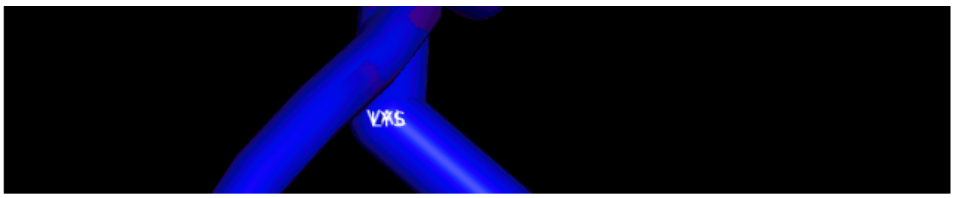


Figura 2.5.5.2.12. Posición 214 del VCF en púrpura bajo (RMSD bajo).

Al final de la aplicación, se muestra el contenido de todo el log de ejecución para referencia del usuario (Figura 2.5.5.2.13).



```
PyMOL>delete all
PyMOL>set fetch_path, /var/www/html
 Setting: fetch_path set to /var/www/html.
PyMOL>fetch 4kw4, async=0; as cartoon
HEADER    FLUORESCENT PROTEIN                     23-MAY-13   4KW4
TITLE     CRYSTAL STRUCTURE OF GREEN FLUORESCENT PROTEIN
COMPND    MOL_ID: 1;
COMPND   2 MOLECULE: GREEN FLUORESCENT PROTEIN;
COMPND   3 CHAIN: A;
COMPND   4 ENGINEERED: YES;
COMPND   5 MUTATION: YES;
COMPND   6 OTHER_DETAILS: THE PROTEIN IN THIS STRUCTURE IS A VERSION OF GFP
COMPND   7 CALLED EMERALD GFP
 ObjectMolecule: Read secondary structure assignments.
 ObjectMolecule: Read crystal symmetry information.
 Symmetry: Found 8 symmetry operators.
 CmdLoad: "/var/www/html/4kw4.pdb" loaded as "4kw4".
PyMOL>show cartoon
PyMOL>hide lines
['4kw4']
La estructura primaria de 4kw4 es:
SKGEELFTGVVPILVELDGDVNGHKFSVSGEGEGDATYGKLTLKFICTTGKLPVPWPTLVTTLVQCFARYPDHMKQHDFFKSAMPEGYVQERTIFFKDDGNYKTRA
Residues in '4kw4, without HOH': 227
[['GLY', 'LEU', 3], ['VAL', 'THR', 10], ['LEU', 'LYS', 137], ['ASP', 'ARG', 151], ['LYS', 'VAL', 210]]
The length of the VCF is:  5
Pymol resi: 4, Pymol ref resn: GLY, Pymol mut resn: LEU
4kw4///A/4/ Mutate to: LEU
Selected!
 Mutagenesis: no rotamers found in library.
1) Select done
```

Figura 2.5.5.2.13. Log de ejecución.

El log de ejecución completo de la aplicación también es respaldado en el servidor, y se muestra a continuación (Resaltando partes destacables de la ejecución):

```
darenivar@darenivar-virtual-machine /var/www/html/logs $ ls -ltr
total 168
-rw-r--r-- 1 www-data www-data 6298 Jun  1 12:55 vcf2pdb.20180601125530.log
-rw-r--r-- 1 www-data www-data 6298 Jun  1 22:10 vcf2pdb.20180601221026.log
-rw-r--r-- 1 www-data www-data 6589 Jun  1 22:29 vcf2pdb.20180601222943.log
-rw-r--r-- 1 www-data www-data 5301 Jun  2 03:58 vcf2pdb.20180602035851.log
-rw-r--r-- 1 www-data www-data 5301 Jun  2 04:00 vcf2pdb.20180602040036.log
-rw-r--r-- 1 www-data www-data 3935 Jun  2 04:00 vcf2pdb.20180602040054.log
-rw-r--r-- 1 www-data www-data 5301 Jun  2 04:04 vcf2pdb.20180602040408.log
-rw-r--r-- 1 www-data www-data 6589 Jun  2 04:19 vcf2pdb.20180602041937.log
-rw-r--r-- 1 www-data www-data 6660 Jun  2 21:15 vcf2pdb.20180602211536.log
-rw-r--r-- 1 www-data www-data 6211 Jun  4 09:30 vcf2pdb.20180604093039.log
-rw-r--r-- 1 www-data www-data 6211 Jun  4 10:00 vcf2pdb.20180604100035.log
-rw-r--r-- 1 www-data www-data 6211 Jun  4 20:22 vcf2pdb.20180604202234.log
-rw-r--r-- 1 www-data www-data 6211 Jun  4 20:25 vcf2pdb.20180604202531.log
-rw-r--r-- 1 www-data www-data 6211 Jun  4 21:28 vcf2pdb.20180604212831.log
-rw-r--r-- 1 www-data www-data 6211 Jun  4 21:31 vcf2pdb.20180604213130.log
-rw-r--r-- 1 www-data www-data 1694 Jun  4 21:35 vcf2pdb.20180604213539.log
-rw-r--r-- 1 www-data www-data 6211 Jun  4 21:36 vcf2pdb.20180604213622.log
-rw-r--r-- 1 www-data www-data 6211 Jun  4 21:45 vcf2pdb.20180604214531.log
-rw-r--r-- 1 www-data www-data 6211 Jun  4 21:46 vcf2pdb.20180604214651.log
-rw-r--r-- 1 www-data www-data 6211 Jun  4 21:47 vcf2pdb.20180604214745.log
-rw-r--r-- 1 www-data www-data 6113 Jun  4 22:18 vcf2pdb.20180604221824.log
-rw-r--r-- 1 www-data www-data 6068 Jun  4 22:33 vcf2pdb.20180604223318.log
darenivar@darenivar-virtual-machine /var/www/html/logs $ cat vcf2pdb.20180604223318.log
PyMOL>delete all
PyMOL>set fetch_path, /var/www/html
 Setting: fetch_path set to /var/www/html.
PyMOL>fetch 4kw4, async=0; as cartoon
HEADER    FLUORESCENT PROTEIN                     23-MAY-13   4KW4
TITLE     CRYSTAL STRUCTURE OF GREEN FLUORESCENT PROTEIN
COMPND    MOL_ID: 1;
COMPND   2 MOLECULE: GREEN FLUORESCENT PROTEIN;
COMPND   3 CHAIN: A;
COMPND   4 ENGINEERED: YES;
COMPND   5 MUTATION: YES;
COMPND   6 OTHER_DETAILS: THE PROTEIN IN THIS STRUCTURE IS A VERSION OF GFP
COMPND   7 CALLED EMERALD GFP
 ObjectMolecule: Read secondary structure assignments.
 ObjectMolecule: Read crystal symmetry information.
 Symmetry: Found 8 symmetry operators.
 CmdLoad: "/var/www/html/4kw4.pdb" loaded as "4kw4".
PyMOL>show cartoon
PyMOL>hide lines
['4kw4']
La estructura primaria de 4kw4 es:
SKGEELFTGVVPILVELDGDVNGHKFSVSGEGEGDATYGKLTLKFICTTGKLPVPWPTLVTTLVQCFARYPDHMKQHDFFKSAMPEGYVQERTIFFKDDGN
YKTRAEVKFEGDTLVNRIELKGIDFKEDGNILGHKLEYNYNHHKVYITADKQKNGIKVNFKTRHNIEDGSVQLADHYQQNTPIGDGPVLLPDNHYLHTHSK
LSKDPNEKRDHMVLLEFVTAAGITL
Residues in '4kw4, without HOH': 227
[['GLY', 'LEU', 3], ['VAL', 'THR', 10], ['LEU', 'LYS', 137], ['ASP', 'ARG', 151], ['LYS', 'VAL',
210]]
The length of the VCF is:  5
Pymol resi: 4, Pymol ref resn: GLY, Pymol mut resn: LEU
4kw4///A/4/ Mutate to: LEU
Selected!
 Mutagenesis: no rotamers found in library.
1) Select done
 ExecutiveRMS: RMS =    0.020 (3 to 3 atoms)
 Mutagenesis: 4 rotamers loaded.
2) Set mutation done
3) Apply done
PyMOL>show sticks, resi 4
PyMOL>color firebrick, resi 4
 Executive: Colored 19 atoms.
Pymol resi: 11, Pymol ref resn: VAL, Pymol mut resn: THR
4kw4///A/11/ Mutate to: THR
Selected!
 ExecutiveRMS: RMS =    0.026 (4 to 4 atoms)
 Mutagenesis: 4 rotamers loaded.
1) Select done
 ExecutiveRMS: RMS =    0.027 (4 to 4 atoms)
 Mutagenesis: 3 rotamers loaded.
2) Set mutation done
3) Apply done
PyMOL>show sticks, resi 11
PyMOL>color firebrick, resi 11
 Executive: Colored 14 atoms.
Pymol resi: 141, Pymol ref resn: LEU, Pymol mut resn: LYS
4kw4///A/141/ Mutate to: LYS
Selected!
```

```
 ExecutiveRMS: RMS =    0.017 (4 to 4 atoms)
 Mutagenesis: 3 rotamers loaded.
1) Select done
 ExecutiveRMS: RMS =    0.016 (4 to 4 atoms)
 Mutagenesis: 20 rotamers loaded.
2) Set mutation done
3) Apply done
PyMOL>show sticks, resi 141
PyMOL>color firebrick, resi 141
 Executive: Colored 22 atoms.
Pymol resi: 155, Pymol ref resn: ASP, Pymol mut resn: ARG
4kw4///A/155/ Mutate to: ARG
Selected!
 ExecutiveRMS: RMS =    0.037 (4 to 4 atoms)
 Mutagenesis: 13 rotamers loaded.
1) Select done
 ExecutiveRMS: RMS =    0.037 (4 to 4 atoms)
 Mutagenesis: 17 rotamers loaded.
2) Set mutation done
3) Apply done
PyMOL>show sticks, resi 155
PyMOL>color firebrick, resi 155
 Executive: Colored 24 atoms.
Pymol resi: 214, Pymol ref resn: LYS, Pymol mut resn: VAL
4kw4///A/214/ Mutate to: VAL
Selected!
 ExecutiveRMS: RMS =    0.016 (4 to 4 atoms)
 Mutagenesis: 18 rotamers loaded.
1) Select done
 ExecutiveRMS: RMS =    0.026 (4 to 4 atoms)
 Mutagenesis: 3 rotamers loaded.
2) Set mutation done
3) Apply done
PyMOL>show sticks, resi 214
PyMOL>color firebrick, resi 214
 Executive: Colored 16 atoms.
PyMOL>save 4kw4 mut.pdb, 4kw4
 Save: wrote "4kw4_mut.pdb".
PyMOL>delete 4kw4
PyMOL>fetch 4kw4_mut, async=0; as cartoon
 CmdLoad: "/var/www/html/4kw4_mut.pdb" loaded as "4kw4_mut".
PyMOL>alter (chain A), chain="B"
 Alter: modified 3982 atoms.
PyMOL>fetch 4kw4, async=0; as cartoon
HEADER    FLUORESCENT PROTEIN                     23-MAY-13   4KW4
TITLE     CRYSTAL STRUCTURE OF GREEN FLUORESCENT PROTEIN
COMPND    MOL_ID: 1;
COMPND   2 MOLECULE: GREEN FLUORESCENT PROTEIN;
COMPND   3 CHAIN: A;
COMPND   4 ENGINEERED: YES;
COMPND   5 MUTATION: YES;
COMPND   6 OTHER_DETAILS: THE PROTEIN IN THIS STRUCTURE IS A VERSION OF GFP
COMPND   7 CALLED EMERALD GFP
 ObjectMolecule: Read secondary structure assignments.
 ObjectMolecule: Read crystal symmetry information.
 Symmetry: Found 8 symmetry operators.
 CmdLoad: "/var/www/html/4kw4.pdb" loaded as "4kw4".
PyMOL>align 4kw4_org, 4kw4_mut, object=aln4kw4
 Match-Warning: unknown residue type 'CRO' (using X).
 Match-Warning: unknown residue type 'CRO' (using X).
 Match: read scoring matrix.
 Match: assigning 531 x 531 pairwise scores.
 MatchAlign: aligning residues (531 vs 531)...
 ExecutiveAlign: 3621 atoms aligned.
 ExecutiveRMS: 10 atoms rejected during cycle 1 (RMS=0.05).
 ExecutiveRMS: 22 atoms rejected during cycle 2 (RMS=0.00).
 Executive: RMS =    0.000 (3589 to 3589 atoms)
 Executive: object "aln4kw4" created.
PyMOL>save 4kw4.aln, aln4kw4
 Save: wrote "4kw4.aln".
PyMOL>save vcf_align.pdb, (4kw4_org, 4kw4_mut)
 Save: wrote "vcf_align.pdb".
PyMOL>fetch vcf_align, async=0; as cartoon
 CmdLoad: "/var/www/html/vcf_align.pdb" loaded as "vcf_align".
PyMOL>color blue
 Executive: Colored 15866 atoms.
PyMOL>colorbyrmsd 4kw4 mut, 4kw4 org, doAlign=1, doPretty=1
 ColorByRMSD: Minimum Distance: 0.00
 ColorByRMSD: Maximum Distance: 0.05
 ColorByRMSD: Average Distance: 0.00
PyMOL>set label_color, white, 4kw4_org
 Setting: label_color set to white in object "4kw4_org".
PyMOL>show sticks, resi 4
PyMOL>label i. 4, resn
 Label: labelled 52 atoms.
PyMOL>zoom resi 4
 Movie: frame    1 of    1, 3.84 sec. (0:00:03 - 0:00:03 to go).
PyMOL>show sticks, resi 11
```

```
PyMOL>label i. 11, resn
 Label: labelled 60 atoms.
PyMOL>zoom resi 11
 Movie: frame    1 of    1, 2.01 sec. (0:00:02 - 0:00:02 to go).
PyMOL>show sticks, resi 141
PyMOL>label i. 141, resn
 Label: labelled 82 atoms.
PyMOL>zoom resi 141
 Movie: frame    1 of    1, 3.33 sec. (0:00:03 - 0:00:03 to go).
PyMOL>show sticks, resi 155
PyMOL>label i. 155, resn
 Label: labelled 72 atoms.
PyMOL>zoom resi 155
 Movie: frame    1 of    1, 3.28 sec. (0:00:03 - 0:00:03 to go).
PyMOL>show sticks, resi 214
PyMOL>label i. 214, resn
 Label: labelled 52 atoms.
PyMOL>zoom resi 214
 Movie: frame    1 of    1, 1.98 sec. (0:00:01 - 0:00:01 to go).
PyMOL>orient
PyMOL>show sticks, resi 4
PyMOL>show sticks, resi 11
PyMOL>show sticks, resi 141
PyMOL>show sticks, resi 155
PyMOL>show sticks, resi 214
PyMOL>run pymol2glmol.py
PyMOL>pymol2glmol prot_res
PyMOL>delete all
darenivar@darenivar-virtual-machine /var/www/html/logs $
```

2.5.5.3 Segundo Ejemplo: Mutación en cuatro aminoácidos diferentes para la proteína PDB 1L02.

El archivo VCF que indica los cambios en los aminoácidos se muestra a continuación.

```
##fileformat=VCFv4.1
##fileDate=20180531
##source=TFM
##phasing=partial
##FILTER=<ID=q10,Description="Quality below 10">
##FILTER=<ID=s50,Description="Less than 50% of samples have data">
#CHROM      POS    ID    REF    ALT    QUAL  FILTER      INFO
      FORMAT      NA00001     NA00002     NA00003
1      30     rs001 GGA    AAA    75     PASS
1      74     rs002 GCC    TGG    50     PASS
1      135    rs003 AAA    AAT    50     PASS
1      158    rs005 TGG    ACA    50     PASS
```

El archivo VCF viene expresado en términos de ADN, por lo tanto, es posible deducir lo siguiente:

- En la posición 30 de la secuencia, se encuentran los codones de ADN GGA, que se tratan del aminoácido Glicina (G) los cuales se mutarán en los codones de ADN AAA, que se tratan del aminoácido Lisina (K).

- En la posición 74 de la secuencia, se encuentran los codones de ADN GCC, que se tratan del aminoácido Alanina (A) los cuales se mutarán en los codones de ADN TGG, que se tratan del aminoácido Triptófano (W).

- En la posición 135 de la secuencia, se encuentran los codones de ADN AAA, que se tratan del aminoácido Lisina (K) los cuales se mutarán en los codones de ADN AAT, que se tratan del aminoácido Asparagina (N).
- En la posición 158 de la secuencia, se encuentran los codones de ADN TGG, que se tratan del aminoácido Triptófano (W) los cuales se mutarán en los codones de ADN ACA, que se tratan del aminoácido Treonina (T).

En la figura 2.5.5.3.1, se muesta el ingreso del archivo VCF mencionado.

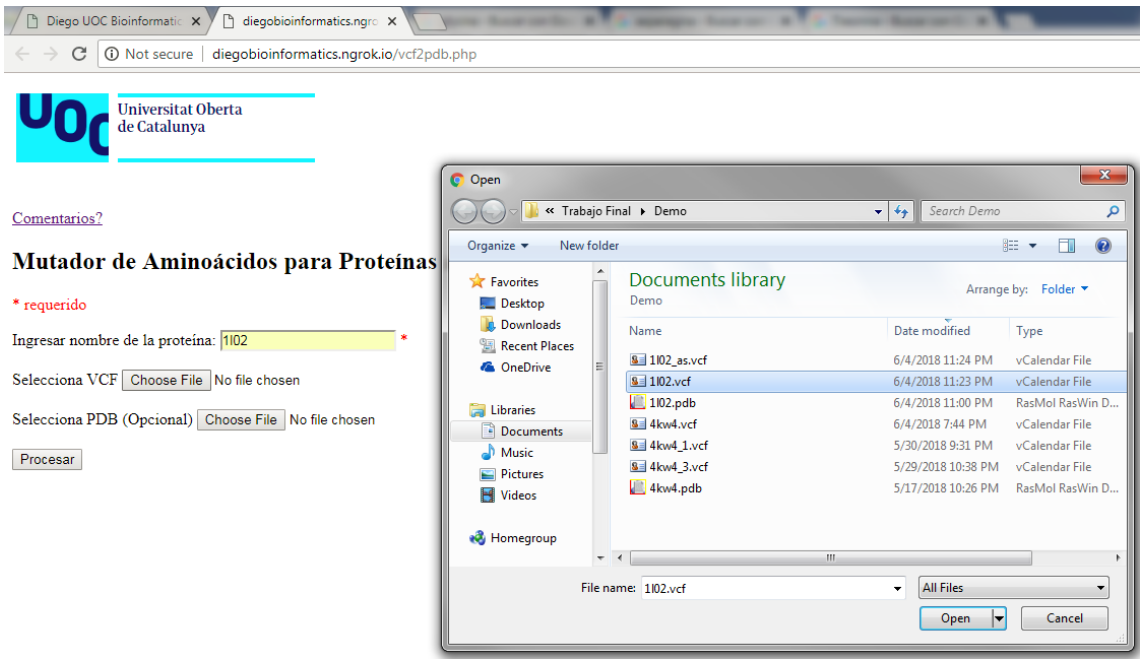


Figura 2.5.5.3.1. Ingreso de archivo VCF.

En este ejemplo, se ingresará el archivo PDB también (Figura 2.5.5.3.2)

Al presionar el botón Procesar la aplicación comienza preparar los datos de salida.

Al desplegar los resultados, se muestra una breve ayuda de interpretación de los mismos (Figura 2.5.5.3.3).

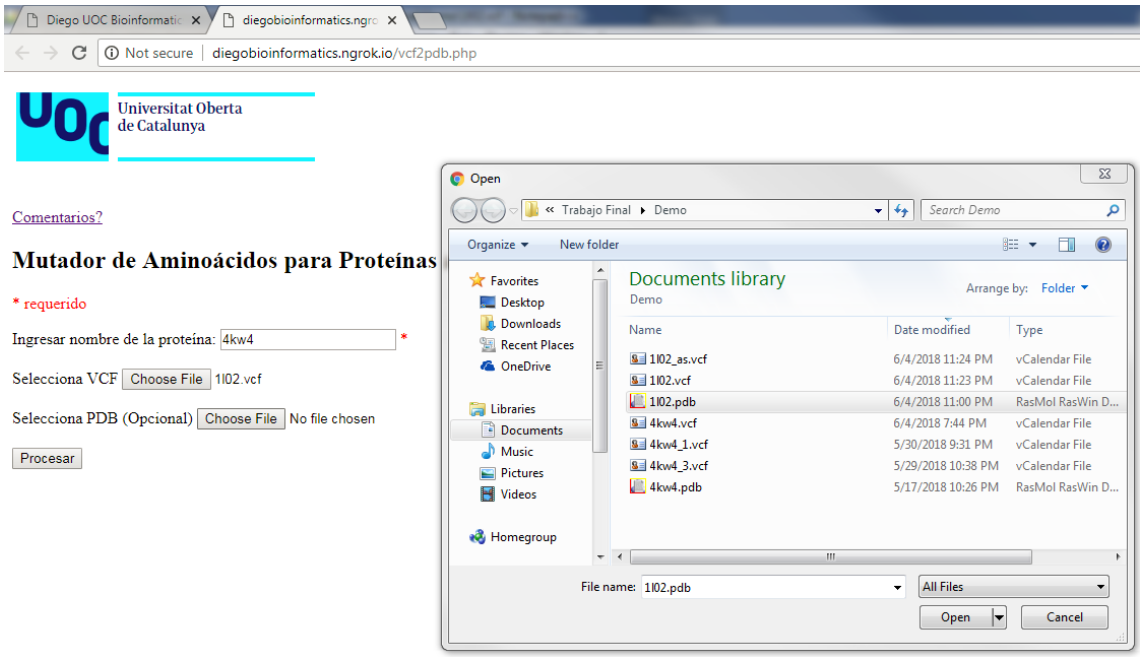


Figura 2.5.5.3.2. Ingreso de archivo VCF.

Figura 2.5.5.3.3. Ayuda en la interpretación de resultados e hipervínculos de visualización 3D y CLUSTAL de alineamiento.

La visualización de la estructura 3D resultante coloreada de acuerdo al valor RMSD en HTML se muestra en la figura 2.5.5.3.4.

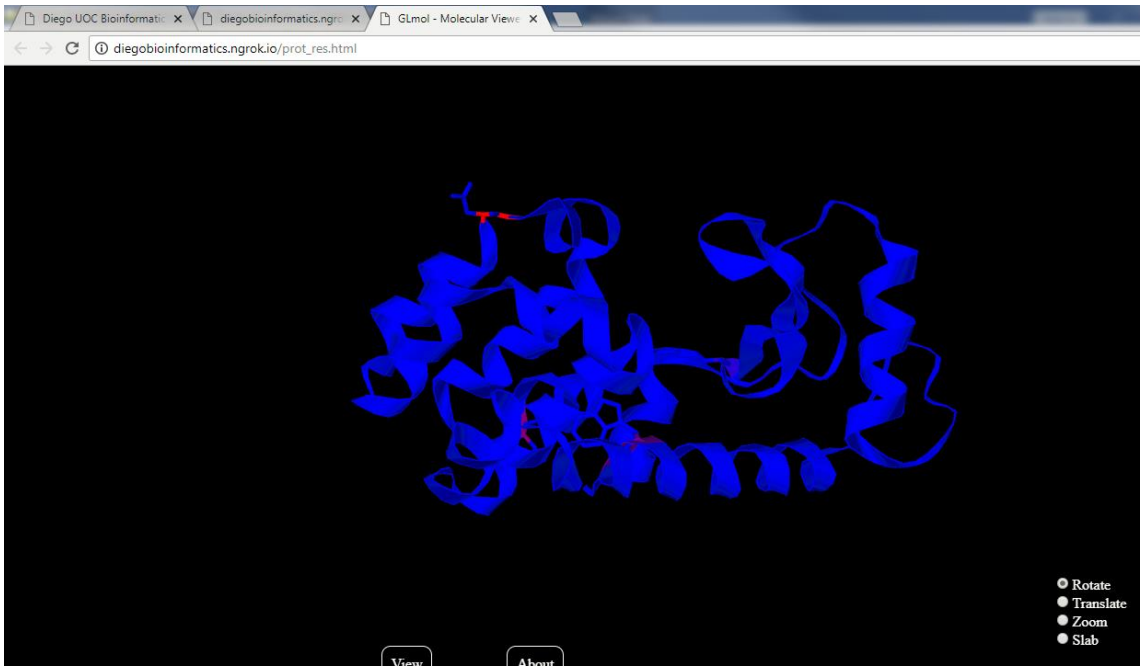


Figura 2.5.5.3.4. Estructura 3D de la proteína 1L02 mutada definida desde el archivo VCF.

En la figura 2.5.5.3.5 es posible visualizar la estructura más de cerca en diferente posición, mostrando residuo mutado con un RMSD alto en rojo.

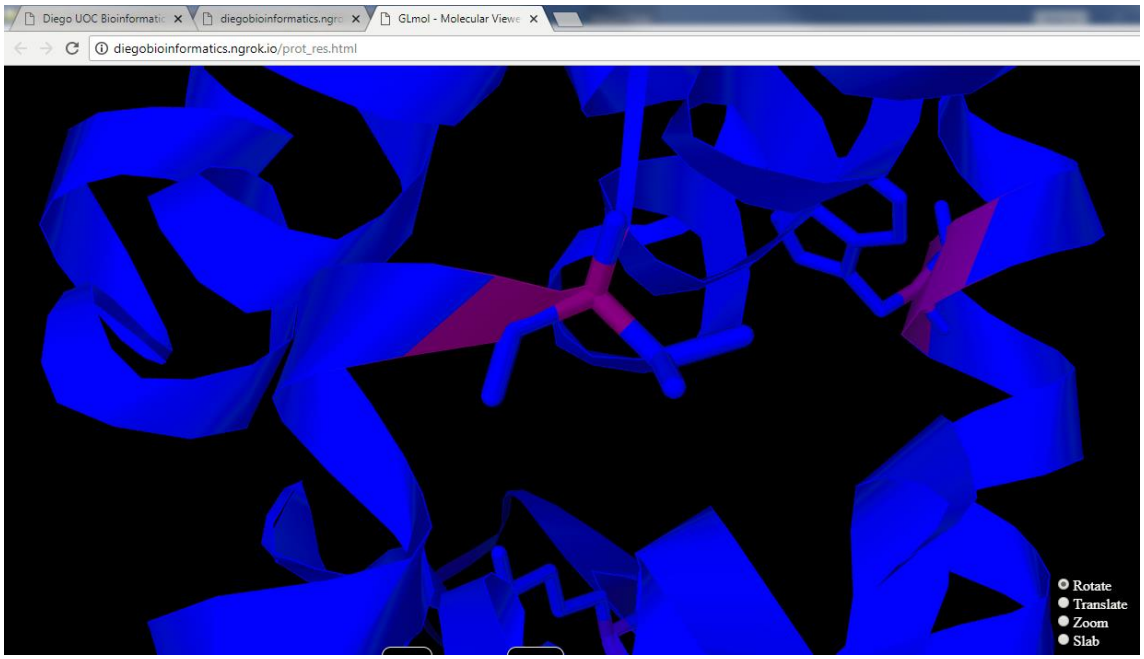


Figura 2.5.5.3.5. Estructura 3D de la proteína 4KW4 mutada desde archivo VCF mostrando de cerca residuo mutado con un RMSD alto.

Los resultados del alineamiento entre proteína original y mutada en formato CLUSTAL se muestran en la figura 2.5.5.3.6.

Figura 2.5.5.3.6. CLUSTAL de alineamiento.

El análisis de cada mutación efectuada a través del archivo VCF es mostrado desplegando el número de residuo para PDB y su posición el archivo VCF, los nombres del aminoácido original y mutado, y el valor del RMSD resultante. En las figuras 2.5.5.3.7 a 2.5.5.2.10 es posible visualizar el análisis de cada mutación.

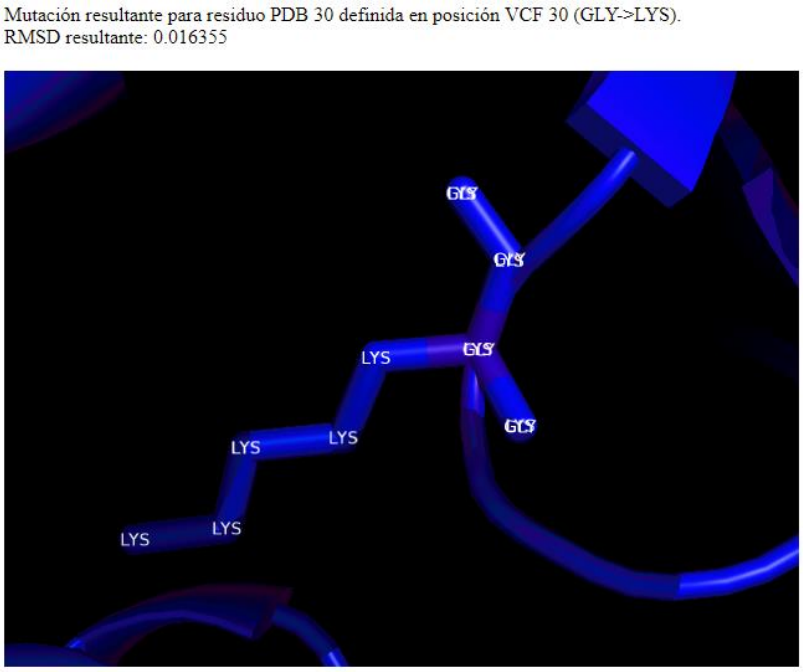


Figura 2.5.5.3.7. Posición 30 del VCF en púrpura bajo (RMSD bajo).

Mutación resultante para residuo PDB 74 definida en posición VCF 74 (ALA->TRP).
RMSD resultante: 0.023737

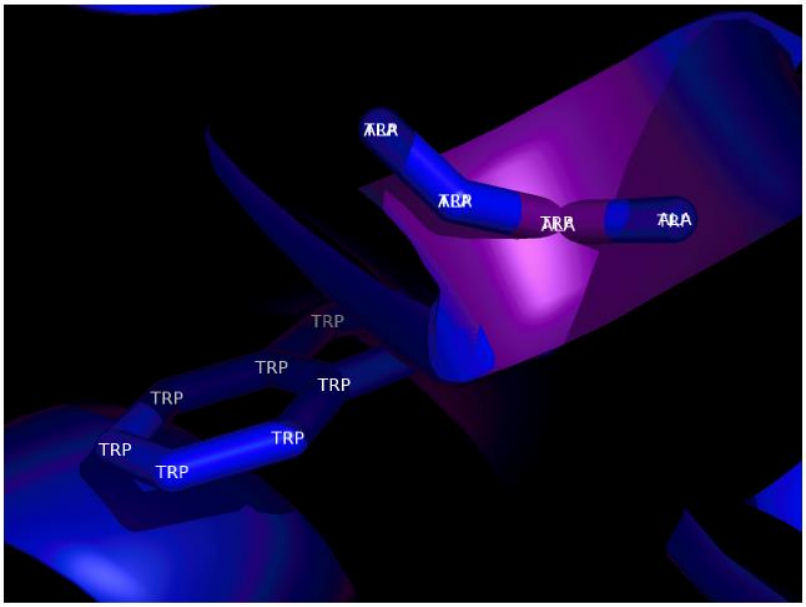


Figura 2.5.5.3.8. Posición 74 del VCF en púrpura bajo (RMSD bajo).

Mutación resultante para residuo PDB 135 definida en posición VCF 135 (LYS->ASN).
RMSD resultante: 0.389325

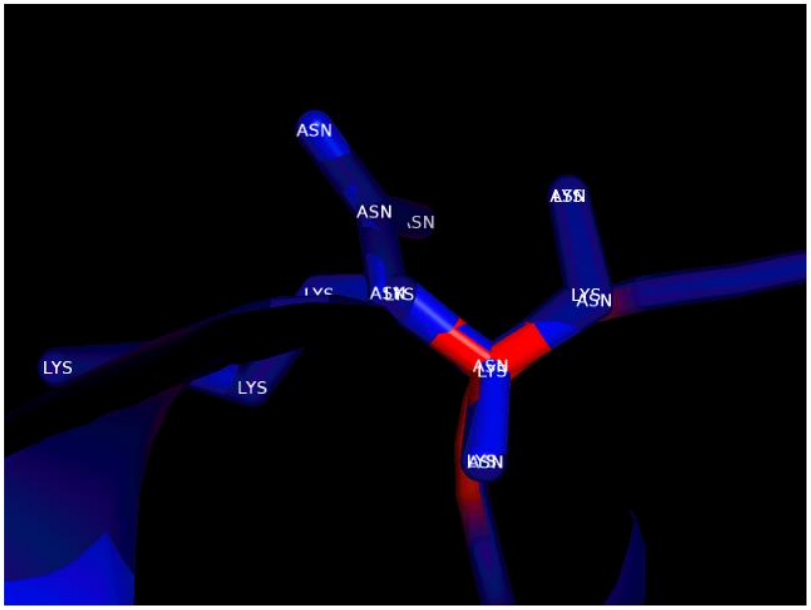


Figura 2.5.5.3.9. Posición 135 del VCF en rojo (RMSD alto).

Mutación resultante para residuo PDB 158 definida en posición VCF 158 (TRP->THR).
RMSD resultante: 0.041281

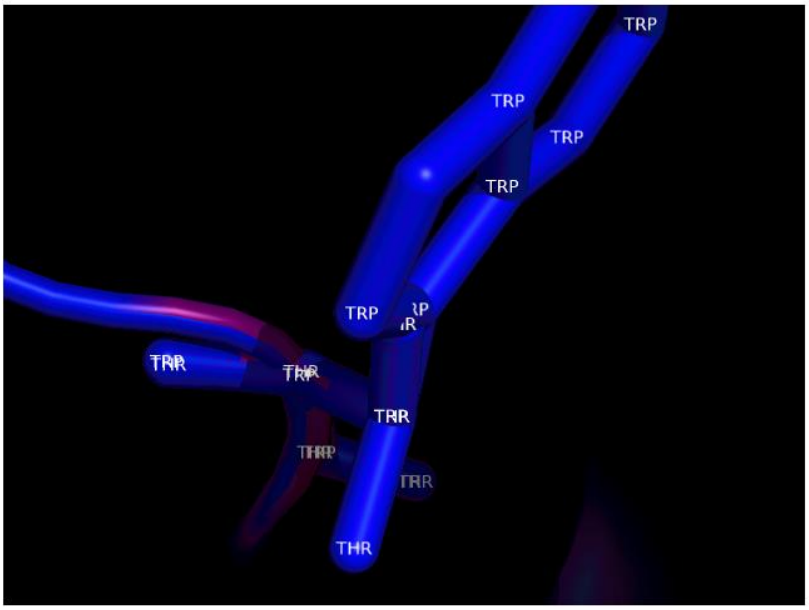
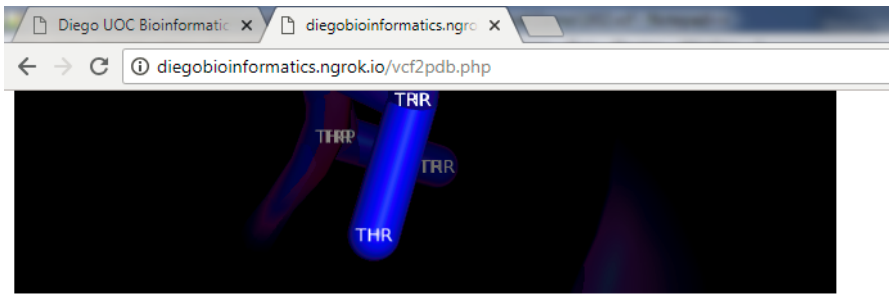


Figura 2.5.5.3.10. Posición 158 del VCF púrpura bajo (RMSD bajo).

Al final de la aplicación, se muestra el contenido de todo el log de ejecución para referencia del usuario (Figura 2.5.5.3.11).




```
PyMOL>delete all
PyMOL>set fetch_path, /var/www/html
 Setting: fetch_path set to /var/www/html.
PyMOL>fetch 1l02, async=0; as cartoon
HEADER    HYDROLASE (O-GLYCOSYL)                  05-FEB-88   1L02
TITLE     CONTRIBUTIONS OF HYDROGEN BONDS OF THR 157 TO THE THERMODYNAMIC
TITLE    2 STABILITY OF PHAGE T4 LYSOZYME
COMPND    MOL_ID: 1;
COMPND   2 MOLECULE: T4 LYSOZYME;
COMPND   3 CHAIN: A;
COMPND   4 EC: 3.2.1.17;
COMPND   5 ENGINEERED: YES
 ObjectMolecule: Read secondary structure assignments.
 ObjectMolecule: Read crystal symmetry information.
 Symmetry: Found 6 symmetry operators.
 ObjectMolReadPDBStr: using SCALEn to compute orthogonal coordinates.
 CmdLoad: "/var/www/html/1l02.pdb" loaded as "1l02".
PyMOL>show cartoon
PyMOL>hide lines
['1l02']
La estructura primaria de 1l02 es:
MNIFEMLRIDEGLRLKIYKDTEGYYTIGIGHLLTKSPSLNAAKSELDKAIGRNCNGVITKDEAEKLFNQDVDAAVRGILRNAKLKPVYDSLDAVRI
Residues in '1l02, without HOH': 164
[['GLY', 'LYS', 30], ['ALA', 'TRP', 74], ['LYS', 'ASN', 135], ['TRP', 'THR', 158]]
The length of the VCF is:  4
Pymol resi: 30, Pymol ref resn: GLY, Pymol mut resn: LYS
1l02///A/30/ Mutate to: LYS
Selected!
 Mutagenesis: no rotamers found in library.
1) Select done
```

Figura 2.5.5.3.11. Log de ejecución.

El log de ejecución completo de la aplicación también es respaldado en el servidor, y se muestra a continuación (Resaltando partes destacables de la ejecución):

```
darenivar@darenivar-virtual-machine /var/www/html/logs $ cat vcf2pdb.20180605005450.log
PyMOL>delete all
PyMOL>set fetch path, /var/www/html
 Setting: fetch path set to /var/www/html.
PyMOL>fetch 1l02, async=0; as cartoon
HEADER    HYDROLASE (O-GLYCOSYL)                  05-FEB-88   1L02
TITLE     CONTRIBUTIONS OF HYDROGEN BONDS OF THR 157 TO THE THERMODYNAMIC
TITLE    2 STABILITY OF PHAGE T4 LYSOZYME
COMPND    MOL_ID: 1;
COMPND   2 MOLECULE: T4 LYSOZYME;
COMPND   3 CHAIN: A;
COMPND   4 EC: 3.2.1.17;
COMPND   5 ENGINEERED: YES
 ObjectMolecule: Read secondary structure assignments.
 ObjectMolecule: Read crystal symmetry information.
 Symmetry: Found 6 symmetry operators.
 ObjectMolReadPDBStr: using SCALEn to compute orthogonal coordinates.
 CmdLoad: "/var/www/html/1l02.pdb" loaded as "1l02".
PyMOL>show cartoon
PyMOL>hide lines
['1l02']
La estructura primaria de 1l02 es:
MNIFEMLRIDEGLRLKIYKDTEGYYTIGIGHLLTKSPSLNAAKSELDKAIGRNCNGVITKDEAEKLFNQDVDAAVRGILRNAKLKPVYDSLDAVRRCALIN
MVFQMGETGVAGFTNSLRMLQQKRWDEAAVNLAKSRWYNQTPNRAKRVITTFRTGAWDAYKNL
Residues in '1l02, without HOH': 164
[['GLY', 'LYS', 30], ['ALA', 'TRP', 74], ['LYS', 'ASN', 135], ['TRP', 'THR', 158]]
The length of the VCF is:  4
Pymol resi: 30, Pymol ref resn: GLY, Pymol mut resn: LYS
1l02///A/30/ Mutate to: LYS
Selected!
 Mutagenesis: no rotamers found in library.
1) Select done
 ExecutiveRMS: RMS =    0.019 (3 to 3 atoms)
 Mutagenesis: 14 rotamers loaded.
2) Set mutation done
3) Apply done
PyMOL>show sticks, resi 30
PyMOL>color firebrick, resi 30
 Executive: Colored 9 atoms.
Pymol resi: 74, Pymol ref resn: ALA, Pymol mut resn: TRP
1l02///A/74/ Mutate to: TRP
Selected!
 ExecutiveRMS: RMS =    0.029 (4 to 4 atoms)
 Mutagenesis: 16 rotamers loaded.
1) Select done
 ExecutiveRMS: RMS =    0.026 (4 to 4 atoms)
 Mutagenesis: 7 rotamers loaded.
2) Set mutation done
3) Apply done
PyMOL>show sticks, resi 74
PyMOL>color firebrick, resi 74
 Executive: Colored 14 atoms.
Pymol resi: 135, Pymol ref resn: LYS, Pymol mut resn: ASN
1l02///A/135/ Mutate to: ASN
Selected!
 ExecutiveRMS: RMS =    0.100 (4 to 4 atoms)
 Mutagenesis: 6 rotamers loaded.
1) Select done
 ExecutiveRMS: RMS =    0.108 (4 to 4 atoms)
 Mutagenesis: 13 rotamers loaded.
2) Set mutation done
3) Apply done
PyMOL>show sticks, resi 135
PyMOL>color firebrick, resi 135
 Executive: Colored 8 atoms.
Pymol resi: 158, Pymol ref resn: TRP, Pymol mut resn: THR
1l02///A/158/ Mutate to: THR
Selected!
 ExecutiveRMS: RMS =    0.047 (4 to 4 atoms)
 Mutagenesis: 9 rotamers loaded.
1) Select done
 ExecutiveRMS: RMS =    0.046 (4 to 4 atoms)
 Mutagenesis: 1 rotamers loaded.
2) Set mutation done
3) Apply done
PyMOL>show sticks, resi 158
PyMOL>color firebrick, resi 158
 Executive: Colored 7 atoms.
PyMOL>save 1l02_mut.pdb, 1l02
 Save: wrote "1l02_mut.pdb".
PyMOL>delete 1l02
```

```
PyMOL>fetch 1l02_mut, async=0; as cartoon
 CmdLoad: "/var/www/html/1l02_mut.pdb" loaded as "1l02_mut".
PyMOL>alter (chain A), chain="B"
 Alter: modified 1432 atoms.
PyMOL>fetch 1l02, async=0; as cartoon
HEADER    HYDROLASE (O-GLYCOSYL)                  05-FEB-88   1L02
TITLE     CONTRIBUTIONS OF HYDROGEN BONDS OF THR 157 TO THE THERMODYNAMIC
TITLE    2 STABILITY OF PHAGE T4 LYSOZYME
COMPND    MOL_ID: 1;
COMPND   2 MOLECULE: T4 LYSOZYME;
COMPND   3 CHAIN: A;
COMPND   4 EC: 3.2.1.17;
COMPND   5 ENGINEERED: YES
 ObjectMolecule: Read secondary structure assignments.
 ObjectMolecule: Read crystal symmetry information.
 Symmetry: Found 6 symmetry operators.
 ObjectMolReadPDBStr: using SCALEn to compute orthogonal coordinates.
 CmdLoad: "/var/www/html/1l02.pdb" loaded as "1l02".
PyMOL>align 1l02_org, 1l02_mut, object=aln1l02
 Match: read scoring matrix.
 Match: assigning 283 x 283 pairwise scores.
 MatchAlign: aligning residues (283 vs 283)...
 ExecutiveAlign: 1295 atoms aligned.
 ExecutiveRMS: 4 atoms rejected during cycle 1 (RMS=0.04).
 ExecutiveRMS: 12 atoms rejected during cycle 2 (RMS=0.00).
 Executive: RMS =    0.000 (1279 to 1279 atoms)
 Executive: object "aln1l02" created.
PyMOL>save 1l02.aln, aln1l02
 Save: wrote "1l02.aln".
PyMOL>save vcf_align.pdb, (1l02_org, 1l02_mut)
 Save: wrote "vcf align.pdb".
PyMOL>fetch vcf align, async=0; as cartoon
 CmdLoad: "/var/www/html/vcf_align.pdb" loaded as "vcf_align".
PyMOL>color blue
 Executive: Colored 5716 atoms.
PyMOL>colorbyrmsd 1l02_mut, 1l02_org, doAlign=1, doPretty=1
 ColorByRMSD: Minimum Distance: 0.00
 ColorByRMSD: Maximum Distance: 0.09
 ColorByRMSD: Average Distance: 0.00
PyMOL>set label_color, white, 1l02_org
 Setting: label_color set to white in object "1l02_org".
PyMOL>show sticks, resi 30
PyMOL>label i. 30, resn
 Label: labelled 26 atoms.
PyMOL>zoom resi 30
 Movie: frame    1 of    1, 2.40 sec. (0:00:02 - 0:00:02 to go).
PyMOL>show sticks, resi 74
PyMOL>label i. 74, resn
 Label: labelled 38 atoms.
PyMOL>zoom resi 74
 Movie: frame    1 of    1, 2.90 sec. (0:00:02 - 0:00:02 to go).
PyMOL>show sticks, resi 135
PyMOL>label i. 135, resn
 Label: labelled 34 atoms.
PyMOL>zoom resi 135
 Movie: frame    1 of    1, 2.11 sec. (0:00:02 - 0:00:02 to go).
PyMOL>show sticks, resi 158
PyMOL>label i. 158, resn
 Label: labelled 42 atoms.
PyMOL>zoom resi 158
 Movie: frame    1 of    1, 1.44 sec. (0:00:01 - 0:00:01 to go).
PyMOL>orient
PyMOL>show sticks, resi 30
PyMOL>show sticks, resi 74
PyMOL>show sticks, resi 135
PyMOL>show sticks, resi 158
PyMOL>run pymol2glmol.py
PyMOL>pymol2glmol prot_res
PyMOL>delete all
```

2.6 Creación de nuevas herramientas.

Se decidió agregar este apartado para detallar que a lo largo del proyecto fue necesario crear nuevas herramientas, las cuales se mencionan en los siguientes apartados de este capítulo.

2.6.1 Traducción de cadenas de aminoácidos a codones de ADN o viceversa.

Debido a la experiencia que se cuenta haciendo herramientas con Python, se decidió crear los traductores necesarios para establecer la comunicación entre archivos VCF (Expresado con codones de ADN) y archivos PDB (Expresado con formato de tres letras de aminoácidos).

Las secciones de código se muestran a continuación:

```
# Diccionario para convertir de codigo de aminoacido de una letra a codones de ADN
(Pudiendo ser consultado en forma reversible de ser necesario).
Aa2Cod ={
             'I':['ATT','ATC','ATA']
            ,'L':['CTT','CTC','CTA','CTG','TTA','TTG']
            ,'V':['GTT','GTC','GTA','GTG']
            ,'F':['TTT','TTC']
            ,'M':['ATG']
            ,'C':['TGT','TGC']
            ,'A':['GCT','GCC','GCA','GCG']
            ,'G':['GGT','GGC','GGA','GGG']
            ,'P':['CCT','CCC','CCA','CCG']
            ,'T':['ACT','ACC','ACA','ACG']
            ,'S':['TCT','TCC','TCA','TCG','AGT','AGC']
            ,'Y':['TAT','TAC']
            ,'W':['TGG']
            ,'Q':['CAA','CAG']
            ,'N':['AAT','AAC']
            ,'H':['CAT','CAC']
            ,'E':['GAA','GAG']
            ,'D':['GAT','GAC']
            ,'K':['AAA','AAG']
            ,'R':['CGT','CGC','CGA','CGG','AGA','AGG']
            ,'Stop':['TAA','TAG','TGA']
       }
# Diccionario para convertir de codigo de aminoacido de una letra a codigo de aminoacido
de tres letras (Pudiendo ser consultado en forma reversible de ser necesario).
Aa12Aa3 ={
             'A':'ALA'
            ,'R':'ARG'
            ,'N':'ASN'
            ,'D':'ASP'
            ,'B':'ASX'
            ,'C':'CYS'
            ,'E':'GLU'
            ,'Q':'GLN'
            ,'Z':'GLX'
            ,'G':'GLY'
            ,'H':'HIS'
            ,'I':'ILE'
            ,'L':'LEU'
            ,'K':'LYS'
            ,'M':'MET'
            ,'F':'PHE'
            ,'P':'PRO'
            ,'S':'SER'
            ,'T':'THR'
            ,'W':'TRP'
            ,'Y':'TYR'
            ,'V':'VAL'
         }
```

```
# Funcion para invertir diccionario (Tomado como base ejemplo de ActiveState Community
[17])
def Inv_dict(d):
    return dict((v, k) for k in d for v in d[k])

# Llamada a diccionario de forma invertida para obtener codones de ADN dado codigo de
aminoacido de una letra.
Cod2Aa = Inv_dict(Aa2Cod)

# Funcion para convertir cadena FASTA de aminoacidos a cadena FASTA de codones de ADN
def aa2dna(fasta_in):
    rev_name = ''
    fasta_str = fasta_in.split('\n')
    seqnum = 0
    dna_seq = ''
    dna_seq_out = ''
    for line in fasta_str:
        if '>' in line:
            seqnum += 1
            if ':' in line: name = line.split(':')[0][1:]
            else: name = line[1:]
            dna_seq = ''
            rev_name = '>reversed_%s\n'%name
            continue
        else:
            for aa in line:
                if aa == '\n': continue
                dna_seq += random.choice(Aa2Cod[aa.upper()]).lower()

        raw_dna_seq = dna_seq.replace('\n', '')
        test_dna_seq = '\n'.join([raw_dna_seq[i:i+78] for i in range(0,
len(raw_dna_seq), 78)])
    dna_seq_out = rev_name + test_dna_seq
    return dna_seq_out

# Funcion para convertir cadena FASTA de aminoacidos de una letra a cadena de tres
letras (Solo para validaciones)
def aa123(fasta_in):
    rev_name = ''
    fasta_str = fasta_in.split('\n')
    seqnum = 0
    dna_seq = ''
    dna_seq_out = ''
    for line in fasta_str:
        if '>' in line:
            seqnum += 1
            if ':' in line: name = line.split(':')[0][1:]
            else: name = line[1:]
            dna_seq = ''
            rev_name = '>aa3_%s\n'%name
            continue
        else:
            for aa in line:
                if aa == '\n': continue
                dna_seq += Aa12Aa3[aa.upper()] + ' '
        raw_dna_seq = dna_seq.replace('\n', '')
        test_dna_seq = '\n'.join([raw_dna_seq[i:i+80] for i in range(0,
len(raw_dna_seq), 80)])
    dna_seq_out = rev_name + test_dna_seq
    return dna_seq_out

# Funcion para convertir cadena FASTA de codones de ADN a cadena FASTA de aminoacidos de
una letra
def dna2aa(fasta_in):
    fasta_str = fasta_in.split('\n')
    seqnum = 0
    aa_seq = ''
    aa_seq_out = ''
    rev_name = ''
    for line in fasta_str:
        if '>' in line:
            seqnum += 1
            if ':' in line: name = line.split(':')[0][1:]
            else: name = line[1:]
            aa_seq = ''
            rev_name = '>aa_%s'%name
            continue
```

```
        else:
            rline = line.replace('\n', '')
            aa_seq += ''.join( [Cod2Aa[rline[i:i+3].upper()] for i in range(0,
len(rline), 3)] )
    test_aa_seq = '\n'.join([aa_seq[i:i+80] for i in range(0, len(aa_seq), 80)])
    return rev_name + test_aa_seq

# Funcion para convertir cadena FASTA cadena de una sola linea.
def GetOneLineSeq(fasta_in):
    fasta_str = fasta_in.split('\n')
    seq_out = ''
    for line in fasta_str:
        if '>' in line:
            if ':' in line: name = line.split(':')[0][1:]
            else: name = line[1:]
            aa_seq = ''
            rev_name = '>aa_%s'%name
            continue
        else:
            seq_out += line.replace('\n', '')
    return seq_out
```

2.6.2 Lectura de VCF.

Se elaboró la función principal de lectura del archivo VCF de entrada, agregando validaciones a los codones de ADN puestos en la columna POS respecto a cadena de aminoácidos principal.

La función se muestra a continuación:

```
def GetMutFASTA(vcf_in, fasta_in):
    fasta_out = list(fasta_in)
    pymol_resi_lst = []
    with open( vcf_in, 'rb') as vcf_file: vcf = vcf_file.readlines()
    for line in vcf:
        if '#' in line:
            continue
        else:
            vcf_data = line.split('\t')
            [
               chrom
              ,pos
              ,pymol_resi
              ,id
              ,ref
              ,alt
              ,qual
              ,filter
              ,info
              ,format
              ,na00001
              ,na00002
              ,na00003
            ] = [
                   vcf_data[0]
                  ,int(vcf_data[1]) - 1
                  ,int(vcf_data[1]) + 1
                  ,vcf_data[2]
                  ,vcf_data[3]
                  ,vcf_data[4]
                  ,vcf_data[5]
                  ,vcf_data[6]
                  ,vcf_data[7]
                  ,vcf_data[8]
                  ,vcf_data[9]
                  ,vcf_data[10]
                  ,vcf_data[11]
                ]
            aaref = dna2aa(ref)
            aaalt = dna2aa(alt)
            if aaref == fasta_in[pos]:
                fasta_out[pos] = aaalt
            else:
```

```
                print 'Diferent aminoacid!! Please check'
                print 'Input Sequence value: %s'%fasta_in[pos]
                print 'VCF Reference value: %s'%aaref
            pymol_resi_lst.append([Aa12Aa3[aaref], Aa12Aa3[aaalt], pymol_resi])
    return ''.join(fasta_out), pymol_resi_lst
```

2.6.3 Página para recibir comentarios.

Con el fin de contar con un medio adicional en el cual se puedan mandar comentarios al instante sobre la herramienta, se creó un hipervínculo a una pequeña página web la cual se muestra en la figura 2.6.3.1.

Figura 2.6.3.1. Página principal de la aplicación remarcando hipervínculo a página de comentarios.

La página de comentarios se muestra en la figura 2.6.3.2.

Figura 2.6.3.2. Página de comentarios.

Todos los mensajes son escritos a una base de datos en el servidor. Los logs de verificación se muestran a continuación:

```
darenivar@darenivar-virtual-machine /var/www/html $ mysql -u root -p -h localhost
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 69
Server version: 5.5.59-0ubuntu0.14.04.1 (Ubuntu)

Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use tfm;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> desc comments;
+------------+---------------+------+-----+---------+-------+
| Field      | Type          | Null | Key | Default | Extra |
+------------+---------------+------+-----+---------+-------+
| dte submit | datetime      | YES  |     | NULL    |       |
| name       | varchar(255)  | YES  |     | NULL    |       |
| site       | varchar(255)  | YES  |     | NULL    |       |
| comments   | varchar(1000) | YES  |     | NULL    |       |
+------------+---------------+------+-----+---------+-------+
4 rows in set (0.17 sec)

mysql> select * from comments order by dte_submit
```

```
    -> ;
+---------------------+----------------+----------------+-----------------------------------------
-------------------------------------------------------------------+
| dte submit          | name           | site           | comments
|
+---------------------+----------------+----------------+-----------------------------------------
-------------------------------------------------------------------+
| 2018-05-13 13:36:34 | Diego          | pdb_reader.php | Comentarios de la página son.
|
| 2018-05-14 07:17:32 | Diego          | pdb reader.php | Comentario de prueba
|
| 2018-05-14 08:28:05 | Miguel         | pdb_reader.php | Si jala al 100% =P
|
| 2018-05-14 11:47:50 | Victor         | pdb_reader.php | Test
|
| 2018-05-21 21:30:54 | Diego          | pdb_reader.php | Comentario para PEC3.
|
| 2018-06-05 05:39:06 | Usuario de TFM | vcf2pdb.php    | Considero que pueden ser aplicadas algunas
mejoras a esta página. Porqué no agregar un correo electrónico.    |
+---------------------+----------------+----------------+-----------------------------------------
-------------------------------------------------------------------+
6 rows in set (0.00 sec)

mysql>
```

3. Conclusiones

Principales lecciones aprendidas en este TFM:

1) Organizar mi tiempo de la mejor manera.

   Verificando en mis registros de entrega de las PEC, me doy cuenta que la PEC0 (Definición de los contenidos del trabajo) se entregó el día 3 de marzo de este año, en otras palabras, de estar definiendo los contenidos al cierre de la memoria y la aplicación completa ¡solo han sido un poco más de tres meses al día de hoy! Definitivamente fechas muy agresivas por cumplir.

   En el caso particular de mi persona, tuve que poner mucha atención en la organización de tiempos de trabajo del proyecto para haber podido lograr llegar hasta estas instancias. Entre un inicio de año con entregas de proyectos importantes en el ámbito laboral y actividades personales, sin organización no podría haber sido posible nada de esto.

   Siempre está presente la sensación de haberlo podido hacer mejor, pero considero que no fue del todo mal.

2) Buscar el camino más fácil.

   Al considerarme una persona curiosa e inquieta, en ocasiones por tratar de lograr un objetivo como lo imaginé en inicio, quizás en ocasiones pensé demás soluciones que se podían haber realizado de una forma más rápida. Como ejemplo puedo citar el consumir algo de tiempo buscando traductores de aminoácidos, y lectores de archivos VCF cuando en realidad yo era capaz de elaborar las herramientas.

3) Nunca perder el entusiasmo por tomar nuevos retos.

   Esto forma parte de mi naturaleza como persona, la bioinformática es un fue un terreno completamente nuevo al iniciar este Máster, lo disfruté y lo sigo disfrutando, a pesar de lo complicado de este semestre.

4) Seguir esforzándome hasta el final.
   Hubo momentos algo complicados en este trabajo final, pero nada me detuvo el intentarlo y tratar de perfeccionar lo más posible, hasta lograr un producto que cumpla con las expectativas. Todo esto con el apoyo incondicional de mis seres queridos.

Crítica sobre los objetivos logrados:

Es de vital importancia tener autocrítica sobre el logro de los objetivos planeados, por lo tanto, a expongo mis conclusiones:

El objetivo principal fue la creación de una herramienta computacional que trate de facilitar las actividades de los facultativos, en particular que la herramienta pueda integrar variantes genómicas en estructuras de proteínas utilizando archivos en formato VCF. Este objetivo fue cumplido, ya que la herramienta computacional creada es una aplicación web de acceso público controlado. Sin embargo, considero que la mejor manera de saber si el objetivo se está cumpliendo por completo, es sometiendo a prueba la herramienta web con diferentes facultativos y tomar en cuenta sus comentarios, ya que son ellos los que en realidad estarían utilizando nuestro trabajo.

Análisis crítico del seguimiento de la planificación y metodología:

Esta conclusión va directamente de la mano con las lecciones aprendidas, ya que se mencionó sobre el organizar el tiempo y buscar el camino más fácil. Algunas veces la planificación sufría de ligeras desviaciones por eventos inesperados, esto se trató de notificar en los reportes de actividades durante el semestre. Pero considero que a pesar de sí sufrir un incremento drástico en las actividades de culminación de herramienta y cierre de memoria, la planificación y un seguimiento de la metodología fueron buenas.

Líneas de trabajo futuro pendientes:

Uno de mis grandes deseos, que espero poder cumplir, es formar o incorporarme a algún grupo de estudio o investigación que pueda darle continuidad a trabajos como este que fue presentado. Me gustaría seguir haciendo mejoras a la aplicación, y someterla a evaluación con los facultativos principalmente.

Las actualizaciones y mejoras que se pudiesen hacerse en un futuro podrían ser las siguientes.

- Migrar el servidor de la máquina virtual en una simple PC a un servidor físico dedicado a las soluciones bioinformáticas.

- Continuar buscando nuevas alternativas más poderosas que podrían sustituir los recursos definidos en este trabajo. Tanto en la interfaz del usuario como en el procesamiento interno.

- A pesar de que esta herramienta puede ser llamada desde cualquier dispositivo móvil, ya sea teléfono o tableta (A través de Chrome), sería interesante crear aplicaciones dedicadas que sean instaladas en los móviles.

- Sería por demás grato, poder hacer un artículo de investigación con este trabajo.

Como se menciona en estas conclusiones, quizás haya algunas cosas que se tengan que ajustar, pero ni las más grandes aplicaciones de clase mundial se salvan de estar haciendo actualizaciones y mejoras continuamente.

4. Glosario

- ADN: Ácido Desoxirribonucleico.
- CLUSTALW file: General purpose DNA or Protein Alignment file.
- GLmol: *Molecular Viewer on WebGL*.
- HTML: *HyperText Markup Language*.
- HTTP: *HyperText Transfer Protocol.*
- PDB: *Protein Data Bank.*
- PEC: Prueba de Evaluación Continua.
- PHP: *HyperText Preprocessor*.
- PyMOL: *Python-Enhanced Molecular graphic tool*.
- RAM: *Random Access Memory.*
- RMSD: *Root Mean Square Deviation*.
- SQL: *Structured Query Language*.
- TFM: Trabajo Final de Máster.
- UNIX: Familia de sistemas operativos multitareas.
- URL: *Uniform Resource Identifier.*
- VCF: *Variant Call Format*.

5. Bibliografía

1) http://www.internationalgenome.org/wiki/Analysis/vcf4.0/: "*VCF (Variant Call Format) version 4.0 | 1000 Genomes"*, consultada por última vez el día 3 de junio del 2018.
2) Barbany Puig, Montserrat: "Apuntes de Biología Estructural (PID_00192773) para la Universidad Oberta de Catalunya (UOC)".
3) https://www.cgl.ucsf.edu/chimera/docs/UsersGuide/tutorials/pdbintro.html: "*Introduction to Protein Data Bank Format"*, consultada por última vez el día 3 de junio del 2018.
4) https://www.linuxmint.com/rel_rosa_cinnamon.php: "*Linux Mint 17.3 Cinnamon Release Notes"*, consultada por última vez el día 3 de junio del 2018.
5) https://www.vmware.com/products/workstation-pro.html: "*Windows VM | VMware Workstation Pro"*, consultada por última vez el día 3 de junio del 2018.

6) https://www.w3.org/Protocols/rfc2616/rfc2616.html: "*Hypertext Transfer Protocol HTTP/1.1"*, consultada por última vez el día 3 de junio del 2018.

7) https://httpd.apache.org/ : "*Apache HTTP Server Project"*, consultada por última vez el día 3 de junio del 2018.

8) https://www.w3schools.com/Html/ : "*HTML Tutorial"*, consultada por última vez el día 3 de junio del 2018.

9) https://www.w3schools.com/php/: "*PHP 5 Tutorial"*, consultada por última vez el día 3 de junio del 2018.

10) https://ngrok.com/: "*ngrok - secure introspectable tunnels to localhost*", consultada el día 3 de junio del 2018.

11) https://cran.r-project.org : "*The Comprehensive R Archive Network"*, consultada por última vez el día 3 de junio del 2018.

12) https://bioconductor.riken.jp/packages/3.3/bioc/vignettes/VariantAnnotation/inst/doc/VariantAnnotation.pdf: "*Introduction to VariantAnnotation"*, consultada el día 3 de junio del 2018.

13) https://cran.r-project.org/web/packages/Rpdb/Rpdb.pdf: "*Package 'Rpdb'"*, consultada el día 3 de junio del 2018.

14) http://thegrantlab.org/bio3d/index.php: "*Bio3D"*, consultada el día 3 de junio del 2018.

15) https://pymol.org/2/: "*PyMOL"*, consultada el día 3 de junio del 2018.

16) https://www.python.org/: "*Welcome to Python.org*", consultada el día 3 de junio del 2018.

17) http://code.activestate.com/recipes/415100-invert-a-dictionary-where-values-are-lists-one-lin/: "*Invert a dictionary where values are lists (one-liner) « Python recipes « ActiveState Code*", consultada el día 3 de junio del 2018.

6. Anexos

Se determinó no agregar ningún anexo.