Universidad del Bío-Bío, Chile

Facultad de Ciencias Empresariales

Departamento de Sistemas de Información

Creación de una Base de Datos para Estructuras Artificiales de proteínas

Proyecto de título presentado por Nicol Angélica Bascuñán Faúndez

para obtener el grado de Ingeniero de Ejecución en Computación e Informática

Dirigida por Tatiana Gutiérrez-Bunster

Co-dirigida por Natalia Jaña

2017

Agradecimientos

Mi tesis la dedico con todo mi amor y cariño a mi compañero de vida Fabian Castillo, quien me ha impulsado de todas las maneras posibles ha seguir adelante, entregando su plena confianza en mis capacidades y siendo mi fuente de inspiración y motivación para superarme cada día.

A mi amada familia, por darme la oportunidad de estudiar esta hermosa carrera, cultivar mi perseverancia y brindar su apoyo incondicional que no me permitieron decaer en los peores momentos, por sobre todas las cosas gracias por creer en mi.

A mis amigos, en especial a Christian Rodriguez, Christopher Gutierrez y Cristian Venegas por su apoyo y ayuda constante durante la carrera y el desarrollo de esta investigación, por estar siempre presente en los momentos de tristeza y alegría, y por sobre todo por su gran cariño y confianza en mi.

A mi profesora Tatiana Gutierrez por su paciencia, comprensión, disponibilidad y motivación constante durante el desarrollo del proyecto que me guiaron de la forma correcta.

Resumen

Actualmente, no existe registro sobre los intentos realizados por los investigadores para obtener las estructuras tridimensionales de las proteínas, lo que significa una perdida de información, que podría ser utilizada en futuras predicciones de estructuras tridimensionales.

La presente investigación se basa en el desarrollo de una base de datos junto con una interfaz web que registra estos intentos. Esto con la finalidad de aportar a la creación de posibles nuevas conformaciones de proteínas.

El sistema web permite realizar ingresos de modelos y diseños al sistema, también se puede editar, eliminar y añadir más diseños al modelo ingresado. Además permite la visualización y descarga de todos los modelos y diseños para cualquier usuario.

Para concretar el desarrollo del sistema se realiza una revisión de los métodos de predicción de estructuras de proteínas, además de las herramientas que se utilizarían y el tipo de base de datos que se desarrollaría. Se determinó trabajar con una base de datos NoSQL en MongoDB, ésto por su libre estructura, adaptación y su capacidad de manipular grandes cantidades de información. Además de utilizar Flask como microframework por sus propiedades no estructuradas y su extensibilidad.

Palabras Clave — DPE, Base de datos, proteínas

Índice General

Lista de Figuras

Índice de Tablas

Índice de JSONs

Capítulo 1

Introducción

La información estructural de proteínas naturales es esencial para comprender aspectos de la biomedicina y la agricultura, donde estudiantes e investigadores se dedican al análisis de estas áreas, abarcando desde la salud y enfermedad hasta la síntesis de proteínas.

Biólogos de todo el mundo, utilizan diversos métodos para detectar complejos proteicos y así determinar las estructura tridimensional de una proteína, estos métodos son los llamados de detección *Interacción Proteína-Proteína (IPP)*. Sin embargo existen otros métodos, los de predicción de estructuras o métodos computacionales de predicción, que a partir de la secuencia de aminoácidos de las proteínas pueden intentar construir un modelo tridimensional, entre ellos se encuentran in vivo, in vitro e in silico los cuales contienen diversas técnicas de detección, como la cristalografía. Si el resultado es exitoso, es almacenado en el único repositorio mundial de información sobre las estructuras tridimensionales, Protein Data Bank (PDB).

Esta investigación tiene como propósito abordar los intentos fallidos de estas estructuras, ya que actualmente no existe trabajo relacionado a la creación de una base de

datos que contenga dicha información, la cual serviría para creación de posibles nuevas conformaciones de proteínas.

La idea de ésta investigación nace de la publicación "The unexpected structure of the designed protein Octarellin V.1 forms a challenge for protein structure prediction tools" en la cual se detectó la importancia de crear una base de datos que permita almacenar toda la información asociada a diseños experimentales que actualmente los científicos no publican (no se conocen), y que es significativa para mejorar futuros diseños o predicciones de estructuras [17].

1.1. Definición del problema

En la actualidad existe una base de datos con información estructural de proteínas naturales, llamada Protein Data Bank (PDB). La cual almacena todas las estructuras de proteínas resueltas hasta ahora. Para que un investigador pueda depositar su proteína en esta base de datos, es necesario contar con un modelo que cumpla con los requerimientos solicitados por PDB. Este modelo se puede obtener a través de variados intentos, realizados mediante *métodos de predicción de estructuras* tales como in vivo, in vitro o in silico (Detallados en la sección 2.3). Los intentos antes mencionados son lo que se llama "diseño de proteínas experimentales (DPE)", de los que se obtienen diversos diseños de la proteína en estudio, que no necesariamente son depositados en PDB. Actualmente sólo se conocen los éxitos de proteínas finales, y no existe ningún registro de los intentos realizados por los investigadores para obtener las estructuras tridimensionales de las proteínas.

Esto representa una carencia de información y datos, que podrían ser tomados en cuenta en futuros algoritmos para el diseño de proteínas o predicciones de estructuras de proteínas [17]. Según el criterio de investigadores, las publicaciones o registros de estos

diseños son aquellos que obtienen mejores resultados.

1.2. Hipótesis

El registro de los intentos realizados por investigadores, para obtener las estructuras tridimensionales de las proteínas, incrementaría el número de plegamiento de proteínas registradas, dado a que habría un registro almacenado de los intentos de predicción de estructura de proteínas. Esto permitiría la creación de posibles nuevas conformaciones de proteínas (diferentes tipos de plegamientos).

Los datos de los "diseños de proteínas" serán una herramienta adicional para investigadores, ya que al momento de aparecer una nueva estructura en PDB se puede realizar una comparación con sus diseños, buscando el más asertivo. Al tener depositados previamente todos los diseños, y no sólo el que se creía era el mejor, serán capaces de observar sus parametrizaciones para aprender de ellas y así mejorar futuros diseños de otras proteínas, predecir estructuras y mejorar los softwares de modelaje y análisis computacional de estructura, como por ejemplo el software Rosetta [40].

1.3. Objetivos

Los objetivos de esta tesis se han dividido de la siguiente forma.

1.3.1. Objetivo General

El objetivo principal de esta investigación es diseñar e implementar una base de datos que permita almacenar la estructura tridimensional experimental de proteínas, incluyendo los parámetros usados en los diseños de creación de la proteína.

1.3.2. Objetivos Específicos

- Estudiar los parámetros utilizados en el proceso de "diseño de proteínas" experimentales.
- Investigar sobre el manejo y almacenamiento de proteínas en su estructura tridimensional.
- Obtener de un set de datos para trabajo inicial.
- Modelado y diseño de los parámetros del "diseño de las proteínas" experimentales y sus diseños resultantes (la base de datos).
- Implementar la base de datos
- Realizar pruebas de la base de datos.
- Crear interfaz web para la base de datos que permita ver e ingresar un nuevo "diseño de proteína" a la base de datos.

1.4. Organización

Este informe desarrolla los siguientes capítulos:

Capítulo 1: Se introduce el planteamiento de la investigación para poder entender la importancia de ésta, junto con la definición del problema, la hipótesis, además de los objetivos generales y específicos.

Capítulo 2: Se introduce la bioinformática para luego definir qué es una proteína, junto a los aminoácidos que la componen y la estructura de ésta, además de dar a conocer la Interacción Proteína-Proteína (IPP). También se habla sobre el *Protein Data Bank* y el formato estándar *.pdb*, junto con las secciones y columnas que lo componen. Para luego finalizar el capítulo con los métodos de predicción de estructura existentes

como in vitro, in vivo e in silico.

Capítulo 3: Se introduce y se presentan los Requerimientos Funcionales y No Funcionales del sistema, para luego presentar los Casos de Uso y sus respectivas descripciones, ésto da paso al Diagrama de clases que entrega más detalle sobre el funcionamiento del sistema mediante la explicación de cada atributo. Para luego finalizar el capítulo con los diagramas de secuencia de ingresar modelo e ingresar diseño.

Capítulo 4: Se introduce y presenta el sistema a crear, partiendo por las herramientas que se utilizan para la realización del software, la descripción de cada una de ellas y sus ventajas. Luego se da paso a iniciar los componentes con los que se trabaja de forma local, crear la base de datos con datos de prueba y la creación de la página web sólo de manera visual, para después dar funcionalidad al sistema, pasar al servidor y poblar la base de datos.

Capítulo 5: Se realizan las pruebas del software, desarrolladas en 3 tipos de interfaces, una pública, una de usuario autor y otra de usuario administrador.

Capítulo 6: Finalmente se expone la conclusión del proyecto y se mencionan trabajos futuros sobre la misma línea de la investigación.

Capítulo 2

Bioinformática

Con el comienzo de nuevas ideas y datos biológicos en los años 60, surgió la necesidad de implementar nuevas disciplinas que pudieran complementar la gestión de la información biológica, donde investigadores pudieran manipular y analizar los datos existentes de una forma mucho más eficiente. Debido a esta necesidad, de gestionar y extraer esta información de forma general en biología, es como surgió el campo de la bioinformática donde se combinan: Biología, Informática y Tecnología, una disciplina indispensable para la enorme cantidad de datos que se generan anualmente [18].

Existen variadas definiciones de bioinformática, una bastante completa es la entregada por el autor Coltell, quien la describe como:
"Una disciplina científica y tecnológica en la que interaccionan en armonía los planteamientos investigadores de la Biología Genética y Molecular, con los enfoques metodológicos y teconológicos de la Ciencia de la Computación y la Ingeniería Informática, para la obtención y gestión del conocimiento biológico genómico y proteómico" [8].

Entonces la bioinformática se divide en 3 subdisciplinas, primero el desarrollo de al-

goritmos nuevos que relacionen grandes conjuntos de datos, segundo el análisis e interpretación de diversos tipos de datos, entre ellos nucleótidos y aminoácidos, estructuras proteicas, entre otros, y tercero la creación de herramientas que permitan facilitar el acceso y manipulación de esta información.

Las proteínas y la gran cantidad de información recabada sobre ellas, ha vuelto indispensable la utilización de la informática para el procesamiento de sus datos, gracias a ésto su estructura tridimensional u otras características se pueden representar de forma digital. Un ejemplo sería la reconocida Base de Datos PDB, de la que se hablará en la sección 2.2, donde se encuentran almacenadas alrededor de 135.000 estructuras proteicas (Ver Figura 2.1).

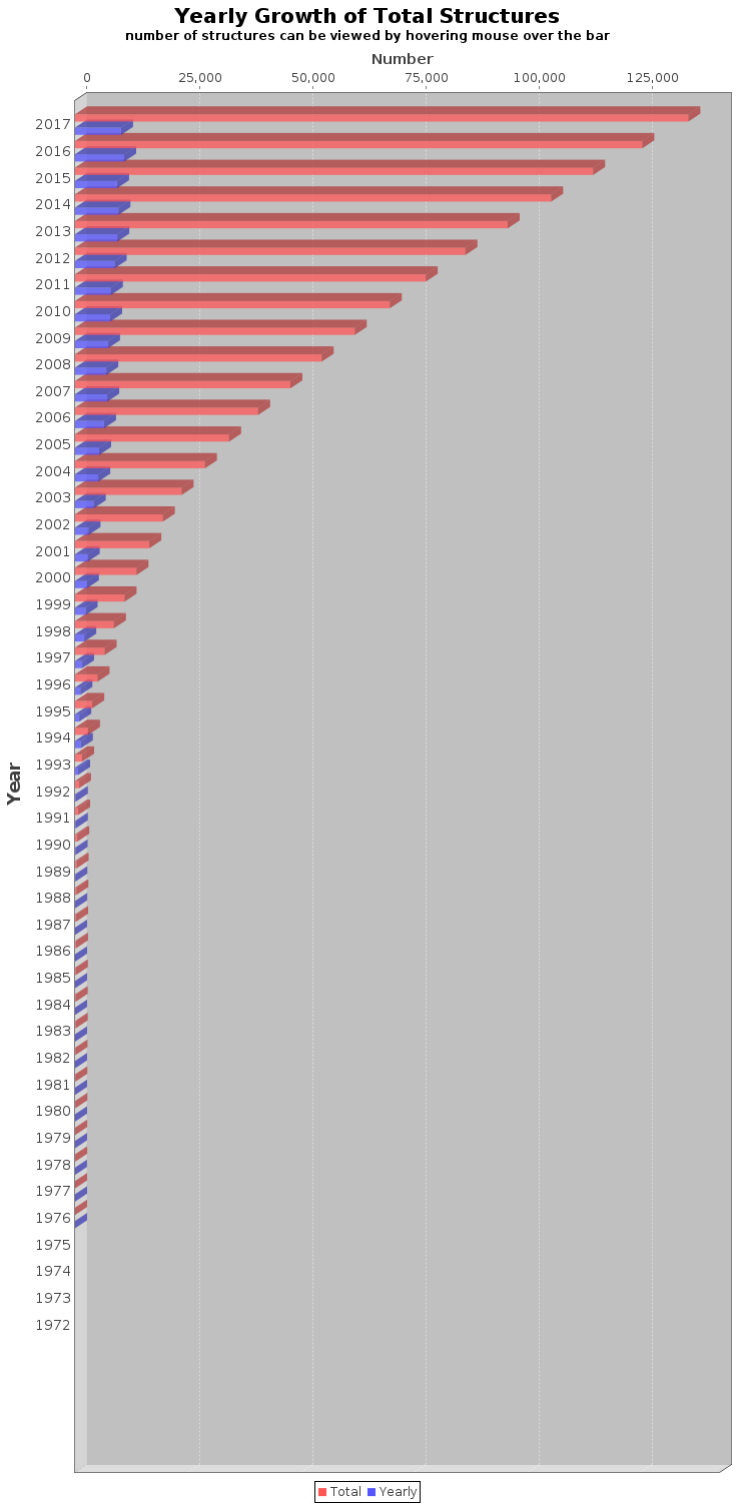

Figura 2.1: Crecimiento anual de las estructuras totales [23].

En este capítulo se definirá qué es una proteína y su estructura principal, para introducir su importancia a nivel biológico.

2.1. Proteínas

La palabra Proteína viene del griego *proteios* que significa *primordial* o *primer lugar* y fue ideado por el químico sueco Berzelius para nombrar la revelación de Gerardus Mulder, quien en 1838 descubrió que ciertas sustancias derivadas de los aminoácidos, conformaban la materia básica del organismo de plantas y animales. La composición de esta *sustancia compleja* incluía nitrógeno (N), y parecía ser la sustancia más importante dentro del *universo orgánico* pues constituyen uno de los nutrimentos de mayor trascendencia en los seres vivos [20].

Las proteínas son macromoléculas de gran importancia a nivel biológico, esto debido a que son responsables de cumplir con la mayoría de las funciones en las células de los seres vivos, además de formar parte de la estructura básica de tejidos como músculos, uñas, tendones, entre otros, son las encargadas de crear, reparar y mantener estos tejidos corporales durante todos los procesos de crecimiento y desarrollo. Realizan funciones metabólicas actuando como enzimas, hormonas o anticuerpos y regularizan los nutrientes, vitaminas, minerales, el transporte de oxígeno y grasas en la sangre y se encargan de eliminar los materiales tóxicos del organismo [20].

Estas moléculas de gran tamaño están constituidas por cadenas lineales de aminoácidos, formadas por enlaces covalentes llamados peptídicos, y tienen una composición química formada mayoritariamente por átomos de hidrógeno, nitrógeno, carbono y oxígeno, además de otros componentes que varían de acuerdo a la proteína en cuestión.

2.1.1. Aminoácidos

Los aminoácidos son moléculas orgánicas que se combinan mediante un *enlace peptídico* (Ver Apéndice A.1) para formar proteínas. Existen más de 300 aminoácidos diferentes en nuestro planeta pero sólo 20 de ellos, llamados *proteinogénicos* o *L-alfa-aminoácidos*, se consideran esenciales para el correcto funcionamiento del organismo (Ver Tabla 2.1). Éstos pueden combinarse de cualquier forma, incluso repetirse para formar una proteína que típicamente se forma por 100 o 200 aminoácidos, es decir, la cantidad de combinaciones diferentes que existen es considerable [10].

Su estructura esta formada por la presencia de un *carbono central* (alfa) que se encuentra unido a un *grupo carboxilo* (Ver Apéndice A.2) (rojo), un *grupo amino* (Ver Apéndice A.3) (verde), un *hidrógeno* (negro) y la *cadena lateral* (azul) (Ver Figura 2.2).[10]

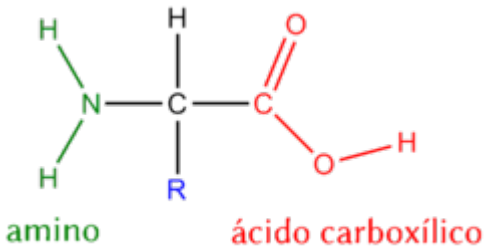

Figura 2.2: Estructura general de un aminoácido [10].

La unión a través de enlaces peptídicos da como resultado una cadena de aminoácidos, si esta cadena contiene [10]:

- Dos ó más aminoácidos, recibe el nombre de péptido.
- Menor a 10 aminoácidos, denomina oligopéptido.
- Mayor a 10 aminoácidos, se llama polipéptido.
- Superior a 50 aminoácidos es proteína.

Aminoácido	Tipo	Abreviatura	Letra
Glicina	NAp	GLI	G
Alanina	NAp	ALA	A
Valina	NAp	VAL	V
Leucina	NAp	LEU	L
Isoleucina	NAp	ILE	I
Metionina	NAp	MET	M
Prolina	NAp	PRO	P
Fenilalanina	NAr	PHE	F
Tirosina	NAr	TRY	Y
Triptófano	NAr	TRP	W
Serina	NP	SER	S
Cisteína	NP	CYS	C
Treonina	NP	TRE	T
Asparagina	NP	ASG	N
Glutamina	NP	GLN	Q
Ácido aspártico	Ácidos	ASP	D
Ácido glutámico	Ácidos	GLU	E
Lisina	Básicos	LYS	K
Arginina	Básicos	ARG	R
Histidina	Básicos	HIS	H

NP: Neutros Polares, NAp: Neutros Apolares, NAr: Neutros Aromatizados

Tabla 2.1: Proteinogénicos [10].

2.1.2. Estructura de las Proteínas

De acorde a su complejidad, las proteínas se organizan en cuatro niveles estructurales:

2.1.2.1. Estructura primaria

Siendo el más básico de los niveles, es el que permite diferenciar a una proteína de otra, debido a que corresponde a la secuencia de aminoácidos que la conforman, es decir, el número de aminoácidos presentes y el orden en que éstos se enlazan para formar la

cadena polipeptídica. Ésta secuencia de aminoácidos es quien determina la estructura tridimensional de la proteína y la funcionalidad que cumplirá (Ver Figura 2.3) [10].

Figura 2.3: Estructura primaria de una proteína [30]

2.1.2.2. Estructura secundaria

Es el modo en que los átomos que forman el enlace peptídico, producen un plegamiento de aminoácidos ordenado y repetitivo, mediante puentes de hidrógeno (Ver Apéndice A.4) [10].

Las dos estructuras secundarias más comunes son:

Hélice alfa: Se produce cuando, debido a los giros en torno al carbono alfa, la cadena de aminoácidos se enrolla en espiral sobre sí misma (Ver Figura 2.4) [10].

Hoja beta: Se produce cuando los aminoácidos se alinean uno al lado del otro formando láminas u hojas unidas por puentes de hidrógeno (Ver Figura 2.4) [10].

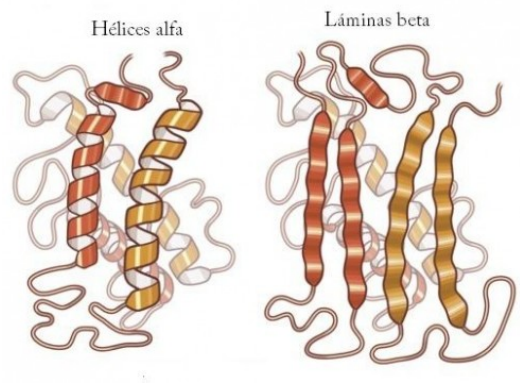

Figura 2.4: Estructura secundaria de una proteína [3]

2.1.2.3. Estructura terciaria

Es la estructura tridimensional plegada y completa de una proteína, que se produce cuando hay ciertas atracciones entre hélices alfa y hojas beta. Es única para cada proteína y también es responsable directa de las propiedades biológicas de ésta [10]. Existen 2 tipos de estructura terciaria:

Fibroso: Este tipo se caracteriza por ser insoluble y dar a la proteína un aspecto de *filamento*. Una de las dimensiones es mucho mayor que las otras dos y se forman a partir de la repetición de estructuras secundarias simples. Un ejemplo sería el *colágeno* que se puede ver en la Figura 2.5 [10].

Globular: Este tipo se caracteriza por ser soluble, dar a la proteína un aspecto de *ovillo* y no tener una dimensión que predomine sobre las demás. Es una estructura más compleja, que se forman a partir de varias estructuras secundarias diferentes. Un ejemplo sería la *mioglobina* que se puede ver en la Figura 2.5 [10].

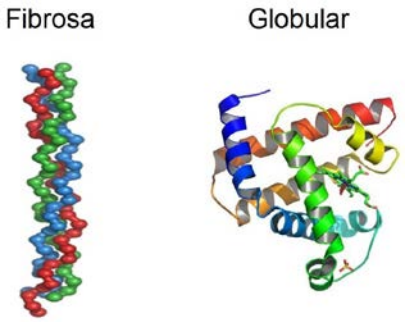

Figura 2.5: Proteínas globulares y fibrosas [28]

2.1.2.4. Estructura cuaternaria

Decimos que una proteína tiene estructura cuaternaria cuando es *oligomérica*, es decir, cuando está formada por varias cadenas polipeptídicas, que reciben el nombre de protómeros. Un ejemplo de proteína cuaternaria es la *hemoglobina*, la cual se compone de 4 cadenas, 2 alfas y dos betas, cada una con una molécula de hierro llamada *grupo hemo* que se puede ver en la Figura 2.6 [10].

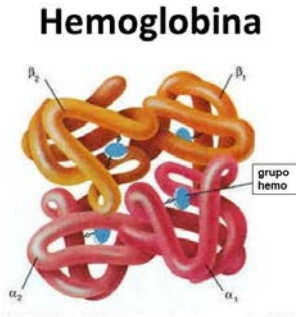

Figura 2.6: Estructura cuaternaria de una proteína [1]

Para resumir los diferentes niveles de la estructura de una proteína, ver la Figura 2.7

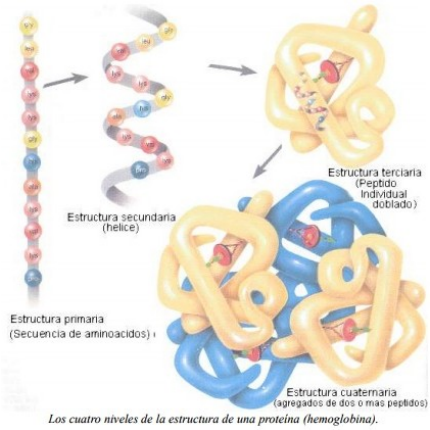

Figura 2.7: Cuatro niveles de la estructura de una proteína [35]

2.1.3. Interacción Proteína-Proteína(IPP)

Las interacciones Proteína-Proteína (IPP) se refiere a la interacción entre dos o más proteínas como resultado de un acoplamiento molecular, que se lleva a cabo en una célula o en un organismo vivo *in vivo*, a esta interacción se le denomina complejos proteicos o multiproteicos y se puede apreciar en la Figura 2.8 que fue visualizada utilizando JMOL en la página "Protein Data Bank"[31].

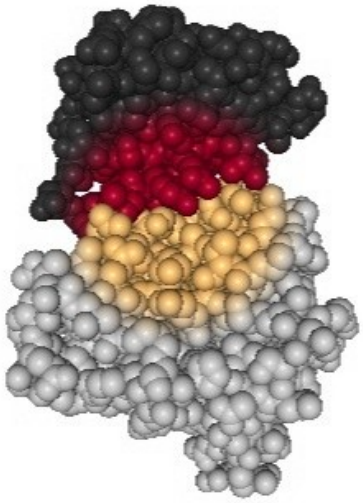

Figura 2.8: Representación zona de interacción (PDB:1LFD)

En la Figura 2.8 se pueden apreciar dos proteínas involucradas en una IPP, donde cada color representa a una proteína y en el centro se destaca la interacción entre cadenas, esta pequeña zona es a lo que se le llama *"Zona de interacción"*, y es la que produce el enlace entre proteínas. Esta zona de interacción cuenta con propiedades diferentes al resto de la estructura del complejo, ya que se forma por fragmentos de ambas cadenas polipétidas, es decir, consiste en el enlace de residuos de aminoácidos mediante puentes de hidrógeno, que pertenecen a dos cadenas diferentes [26].

La gran mayoría de las proteínas no actúan por sí solas, es por esto que el estudio de sus interacciones es tan importante, además de ser esenciales para cualquier proceso celular, se encargan de biosíntesis y degradación de vías, inhibición de enzimas, la regulación celular, entre otros.

2.2. Protein Data Bank PDB

Protein Data Bank (Banco de Datos de Proteínas) es el único repositorio mundial de información sobre las estructuras tridimensionales de las moléculas biológicas grandes, incluidas las proteínas y los ácidos nucleicos. Hoy en día la base de datos contiene más de cien mil entradas de datos de proteínas disponibles de manera gratuita para todo el público. La gran mayoría de los datos son conseguidos mediante a los métodos de difracción de rayos X y de resonancia magnética nuclear (detallada en la Sección 2.3.1.4) [2].

Debido al crecimiento exponencial de estructuras descubiertas por biólogos y científicos, en el año 1976, se creó un formato estándar por los miembros de *Bookhaven Nacional Laboratory*, llamado *.pdb*, que sería capaz de facilitar el almacenamiento y la lectura de las entradas de datos al "Protein Data Bank". Este formato definiría la forma para describir estructuras tridimensionales de macromoléculas y facilitaría la creación de algoritmos para el análisis de grandes datos.

El nombre de un archivo .pdb representa el nombre de la proteína que será ingresada a la base de datos, y esta compuesto por 4 caracteres alfanuméricos seguidos por la extensión *.pdb*, éste nombre es único para cada entrada de datos y sirve para identificar a la proteína.

Secciones de un archivo con formato .pdb

Dentro del archivo .pdb se encuentran las siguientes secciones [4]:

(a) Título: Esta es la primera sección dentro de un archivo .pdb y se encuentra el

resumen de operaciones descriptivas como el nombre de la molécula, autores que participaron en el estudio, tipo de método utilizado en su análisis, entre otros.

(b) Observación: En esta sección van comentarios acerca del ingreso.

(c) Estructura primaria: En esta sección se encuentran identificadores y números de secuencia, que permiten a otros registros vincularse, además incluye la información de cada cadena de la proteína.

(d) Heterógeno: Contiene la descripción no estándar de la macromolécula ingresada.

(e) Estructura Secundaria: Se describe la estructura secundaria, ya sea hélice alfa y/u hoja beta que están presentes en la proteína.

(f) Anotaciones de conectividad: Permite introducir información sobre la existencia y ubicación de puentes disulfuro (ver Apéndice A.5) y otros tipos de enlace.

(g) Características Diversas: Esta sección describe características como el entorno que rodea a un residuo no estándar o el ensamblado de un centro activo.

(h) Cristalografía: Contiene la descripción de la geometría del experimento cristalográfico.

(i) Transformación de coordenadas: Se describen las transformaciones realizadas en el sistema de coordenadas.

(j) Coordenadas: En esta sección se incluyen las coordenadas de cada molécula de aminoácido a nivel atómico, es decir, el registro de coordenadas en el espacio tridimensional (ver Figura 2.9). Estas coordenadas son las que se utilizan en esta investigación y se almacenan en nuestra base de datos para el registro diseños de proteínas experimentales.

	1	2	3	4	5	6	7	8	9	10	11
1	ATOM	1	N	MET	A	1	39.848	54.842	58.312	1.00	6.29
2	ATOM	2	CA	MET	A	1	38.554	54.197	58.536	1.00	6.29
3	ATOM	3	HA	MET	A	1	38.013	54.747	59.304	1.00	6.29
4	ATOM	4	CB	MET	A	1	37.723	54.211	57.240	1.00	6.29
5	ATOM	5	HB1	MET	A	1	36.786	53.683	57.413	1.00	6.29
6	ATOM	6	HB2	MET	A	1	38.273	53.694	56.455	1.00	6.29
7	ATOM	7	CG	MET	A	1	37.404	55.630	56.756	1.00	6.29
8	ATOM	8	HG1	MET	A	1	38.337	56.161	56.567	1.00	6.29
9	ATOM	9	HG2	MET	A	1	36.872	56.155	57.549	1.00	6.29
10	ATOM	10	SD	MET	A	1	36.394	55.709	55.249	1.00	6.29
11	ATOM	11	CE	MET	A	1	37.540	54.978	54.043	1.00	6.29
12	ATOM	12	HE1	MET	A	1	37.115	55.056	53.043	1.00	6.29
13	ATOM	13	HE2	MET	A	1	38.494	55.504	54.069	1.00	6.29
14	ATOM	14	HE3	MET	A	1	37.709	53.926	54.273	1.00	6.29
15	ATOM	15	C	MET	A	1	38.807	52.775	59.065	1.00	6.29
16	ATOM	16	O	MET	A	1	38.811	52.543	60.278	1.00	6.29
17	ATOM	17	N	GLY	A	2	39.207	51.866	58.173	1.00	4.81
18	ATOM	18	H	GLY	A	2	39.117	52.124	57.196	1.00	4.81
19	ATOM	19	CA	GLY	A	2	40.165	50.793	58.474	1.00	4.81
20	ATOM	20	HA1	GLY	A	2	39.778	49.856	58.074	1.00	4.81
21	ATOM	21	HA2	GLY	A	2	40.301	50.685	59.547	1.00	4.81
22	ATOM	22	C	GLY	A	2	41.516	51.062	57.815	1.00	4.81
23	ATOM	23	O	GLY	A	2	42.562	50.673	58.320	1.00	4.81
24	ATOM	24	N	ASP	A	3	41.477	51.819	56.730	1.00	3.47
25	ATOM	25	H	ASP	A	3	40.568	52.012	56.320	1.00	3.47
26	ATOM	26	CA	ASP	A	3	42.586	52.312	55.937	1.00	3.47
27	ATOM	27	HA	ASP	A	3	43.096	51.455	55.497	1.00	3.47
28	ATOM	28	CB	ASP	A	3	41.986	53.151	54.788	1.00	3.47
29	ATOM	29	HB1	ASP	A	3	42.778	53.415	54.085	1.00	3.47
30	ATOM	30	HB2	ASP	A	3	41.588	54.080	55.200	1.00	3.47
31	ATOM	31	CG	ASP	A	3	40.861	52.424	54.041	1.00	3.47
32	ATOM	32	OD1	ASP	A	3	41.113	51.976	52.902	1.00	3.47
33	ATOM	33	OD2	ASP	A	3	39.761	52.305	54.637	1.00	3.47

Figura 2.9: Coordenadas de un archivo .pdb

En la Figura 2.9 podemos apreciar las diferentes columnas que contiene la sección de coordenadas de un archivo .pdb, entre ellas tenemos [4]:

- Columna 1: Esta columna contiene los ATOM que describen el nombre del registro y TER que señala el termino de una cadena de registros.
- Columna 2: Contiene el número serial del átomo.
- Columna 3: Contiene el símbolo químico del átomo.
- Columna 4: Son tres caracteres que describen la abreviatura de los residuos de los

aminoácidos en el registro.

- Columna 5: Representa el identificador de la cadena, cada cadena posee un identificador diferente.
- Columna 6: Contiene el identificador de cada registro de aminoácidos.
- Columna 7-8-9: Estas tres columnas describen las coordenadas x, y, z de todos los átomos de la proteína, que representan la posición de cada uno de ellos en un sistema de coordenadas ortogonal. Estas coordenadas son las que hacen posible la figuración tridimensional de la macromolécula.
- Columna 10: La columna de ocupancia muestra la probabilidad de que el átomo en el registro ocupe la posición determinada. Por lo general es 1.00, que significa que el átomo ocupa esa posición.
- Columna 11: Muestra el factor de temperatura del átomo.

2.3. Métodos de Predicción de Estructuras

Los métodos de detección de IPP proporcionan estructuras de alta resolución, sin embargo la cantidad de proteínas que se pueden caracterizar de esta manera es mínima. Con los métodos computacionales de predicción de estructuras se puede intentar construir un modelo tridimensional a partir de la secuencia de aminoácidos, esto cuando la estructura terciaria de una proteína no se ha determinado experimentalmente.

La predicción de la estructura terciaria de las proteínas tiene como objetivo estimar la posición espacial de todos los átomos de la molécula proteica, utilizando métodos computacionales. Estos ofrecen información para explicar los aspectos funcionales que se pueden derivar del conocimiento estructural [34].

Los métodos de predicción de estructuras se pueden categorizar en tres tipos *in vitro*,

in vivo e *in silico* los cuales son descritos a continuación.

2.3.1. In vitro

Estos métodos de detección se realizan en un ambiente controlado fuera de un organismo vivo y están basado en el uso de extractos celulares más o menos complejos.

2.3.1.1. Cristalografía de rayos X

Esta técnica utiliza *"difracción de rayos X"* (Ver Apéndice A.6) en las macromoléculas previamente cristalizadas para así determinar los complejos proteicos a nivel atómico e inequívocamente determinar la estructura tridimensional [31].

2.3.1.2. Coinmunoprecipitación

Esta técnica utiliza *inmunoprecipitación* (Ver Apéndice A.7) lo que permite identificar las interacciones directas o indirectas entre proteínas [31].

2.3.1.3. Espectrometría de masas

Esta técnica, además de contar con una eficaz capacidad para la identificación de proteínas, permite determinar la masa de moléculas. Esto se realiza mediante la medida de la relacion masa/carga que resulta de dividir la masa del objeto entre su carga eléctrica [31].

2.3.1.4. Espectroscopía de resonancia magnética nuclear

Esta técnica explota las propiedades magnéticas de ciertos núcleos. Permite extraer las distancias entre los átomos midiendo transmisiones en un campo magnético los dife-

rentes estados de spin nuclear. Luego las distancias son usadas como restricciones para así construir la estructura tridimensional [31].

2.3.2. In vivo

Este método de detección se realiza en el propio organismo vivo y se basa en mantener un ambiente celular semejante al de la proteína nativa.

2.3.2.1. Sistema de dos híbridos

Esta técnica es usada para detectar la interacción entre dos proteínas. Requiere la coexpresión de dos proteínas de fusión, una definida como cebo y la otra como presa, las cuales al interactuar, son capaz de reconstruir una proteína por sí misma, también mediante la activación de un gen reportero o más, confieren al organismo una propiedad que puede ser detectada [31].

2.3.2.2. FRET (Transferencia de energía de resonancia de Förster)

Esta técnica es un mecanismo de transferencia de energía entre cromóforos, uno etiquetado como donador y el otro como receptor.

Cuando interactúan se produce una transferencia energética desde el cromóforo donador al receptor y la energía producida genera un cambio en la fluorescencia del cromóforo receptor que puede ser detectada por medio de microscopía confocal de fluorescencia [31].

2.3.3. In silico

Este método se realiza a través de un computador o a través de una simulación por computador. Existen dos estrategias para ello:

2.3.3.1. Utilización de un molde

También llamado "Modelado Comparativo", esta estrategia utiliza estructuras previas como plantillas o puntos de inicio, por este motivo es más exacta [24]. Se pueden dividir en los siguientes métodos:

Modelado por homología (*homology modeling, comparative modeling*): Este método esta basado en que la estructura tridimensional de una proteína (obtenida mediante métodos de detección IPP) sirve como base para la creación de otros modelos tridimensionales, relacionados con su misma familia. Esto enfocado a la situación en que dos secuencias de proteínas sean similares entre sí, es decir homólogas. Es uno de los métodos de predicción más utilizados debido a que genera modelos de gran calidad. El proceso de construcción de un modelo mediante homología, consta de diversas etapas, entre ellas [24]:

- Identificar las *estructuras base* relacionadas con la secuencia problema, es decir, buscar proteínas con una secuencia similar a la secuencia de la proteína que se quiere modelar, para esto se emplean métodos de comparaciones de secuencias como FASTA, BLAST o PSI-BLAST [24].
- Alinear lo mejor posible ambas secuencias, para establecer la correspondencia entre sus aminoácidos. Conforme a este alineamiento se realizará la construcción del modelo, es por esto que es la etapa más importante. Se utilizan programas de alineamiento como CLUSTAL para esta etapa [24].
- Construir el modelo tridimensional desde el alineamiento realizado, utilizando por ejemplo ProModII en el servidor SWISS-MODEL [24].
- Evaluación del modelo, su finalidad es la detección de errores mediante la comprobación correcta de diversos enfoques [24].

Reconocimiento del plegamiento (*fold recognition* o Enhebrado de proteínas: Si no es posible dar con una proteína homologa a la proteína que se quiere modelar, se puede intentar encontrar una proteína que cuente con un plegamiento similar a la proteína problema, sin importar la similitud entre secuencias. Esto ocurre cuando dos proteínas contienen los mismos tipos principales de estructura secundaria, en el mismo orden y conectados por la misma topología. El reconocimiento del plegamiento se basa en que la estructura se conserva mejor que la secuencia, por ende se busca un plegamiento que sea compatible con la proteína a modelar para que pueda servir de base en la construcción del modelo tridimensional. Existen 2 tipos de métodos de predicción que utilizan reconocimiento del plegamiento [24]:

- Método basado en el perfil físico-químico: Basado en propiedades físico-químicas de los aminoácidos de la proteína a modelar, éstas deben adaptarse al entorno que ocupan en la estructura del modelo [24].
- Método de enhebrado (*threading*): Basado en enhebrar la secuencia de la proteína a modelar en una estructura conocida, para evaluar si se adaptan bien o no. Esto se realiza mediante la creación de modelos estructurales de la proteína problema, empleando todos los plegamientos conocidos como posibles bases o puntos de partida, así se puede determinar cual el mejor modelo estructural. Se debe evaluar la calidad de los modelos, ésto calculando la energía de la molécula, el que contenga menor energía será el adecuado [24].

2.3.3.2. Sin utilización de un molde

También llamado "Métodos ab initio (*de novo*)", esta estrategia sólo utiliza la secuencia de la proteína, lo que es una ventaja debido a que es posible modelar proteínas que

correspondan a plegamientos nuevos, sin embargo disminuye su eficiencia con el tamaño de la proteína. Se pueden dividir en los siguientes métodos [34]:

Métodos basados en el conocimiento: Los métodos basado en el conocimiento utiliza fragmentos cortos de proteínas con estructura conocida, para construir el modelo tridimensional de la proteína. Esto debido a que a pesar que la proteína a modelar corresponda a un plegamiento nuevo, puede compartir estructura con plegamientos ya conocidos. El programa ROSETTA utiliza este método de predicción de estructuras [34].

Métodos de simulación: Los métodos basados en simulación se basan en principios físico-químicos, que mediante intentos de simulación del proceso natural de plegamiento proteico, buscan alcanzar la conformación nativa [34].

Las proteínas son de gran importancia en el mundo de la biología, su descubrimiento ha llevado a biólogos y científicos a seguir investigando sobre sus estructuras, así han sido capaces de analizar esta información, utilizando la informática como una herramienta para facilitar el manejo y almacenamiento de esta gran cantidad de datos.

Como ya mencionamos en este capítulo, estos datos finales son almacenados en el gran Banco de Datos de Proteínas mediante el formato *.pdb*. Lo que no se considera es la cantidad de información que se utiliza en los intentos para llegar a ese resultado, y que son desechados. Es por esto que se cree necesario un lugar donde se puedan reunir todos aquellos intentos fallidos de proteínas. Para esto se propone crear una base de datos que sea capaz de reunir estos diseños de proteínas experimentales para incrementar la creación de posibles nuevas conformaciones de proteínas además de mejorar la parametrización de los softwares de modelaje y análisis computacional.

Ya comprendiendo qué es una proteína, la estructura e importancia de ésta, un aminoácido, la interacción Proteína-Proteína, los archivos con formato *.pdb* y por supuesto la predicción de estructuras. Podemos pasar al siguiente capítulo, donde analizaremos los requerimientos funcionales y no funcionales, además de los tipos de usuarios y la estructura del sistema.

Capítulo 3

Base de Datos

En este capítulo se presenta la base de datos y todo lo necesario para llevarla a cabo, esto permite conocer los requerimientos y así analizar la funcionalidad que se incorpora en nuestro sistema.

3.1. Requerimientos

La base de datos se crea para almacenar la estructura tridimensional experimental de proteínas, incluyendo los parámetros usados en los diseños de creación de la proteína. Esta base de datos es visible para diferentes tipos usuarios mediante una página web, con la que se podrá interactuar de diversas maneras con el sistema dependiendo de los permisos éste.

El primer paso para la creación del sistema, es la obtención de los modelos que serán ingresados a la base de datos, esto se realiza a través de *web server* que predicen las estructuras tridimensionales de proteínas. Los web server a utilizar son Bcl Fold, I-Tasser, Psipred y Robetta (Ver Tabla 3.3).

Luego de la obtención de datos se cuenta con la información para poblar la base de datos, que se solicita contenga los requerimientos mencionados en las Tablas (3.1, 3.2).

Los usuarios mencionados en las tablas de requerimientos son detallados en la sección 3.4.

3.1.1. Requerimientos Funcionales

ID	Nombre	Descripción	Usuarios
RF 01	Ingresar al Sistema	El sistema debe permitir iniciar sesión sólo a usuarios registrados, mostrando mensajes de inicio exitoso, contraseña incorrecta o de usuario no registrado.	Admin, Author
RF 02	Gestión de roles	El sistema maneja dos tipos roles para los usuarios, de este rol dependen las opciones que tendrán éstos para poder ingresar, editar o eliminar modelos/diseños.	
RF 03	Ver Modelos	El sistema debe permitir al usuario visualizar los modelos y sus respectivos diseños. El usuario logueado puede ver sus propios modelos, junto con sus diseños ingresados.	Admin, Author, Web
RF 04	Ver Diseños	El sistema debe permitir al usuario visualizar los diseños de cada uno de los modelos.	Admin, Author, Web
RF 05	Descargar Modelos	El sistema debe permitir al usuario descargar los modelos.	Admin, Author, Web
RF 06	Descargar Diseños	El sistema debe permitir al usuario descargar los diseños.	Admin, Author, Web

Tabla 3.1: Requerimientos Funcionales RF01 - RF07

ID	Nombre	Descripción	Usuarios
RF 07	Ingresar Modelos	El sistema debe permitir el ingreso de modelos al sistema.	Admin, Author
RF 08	Ingresar Diseños	El sistema debe permitir el ingreso de diseños al sistema.	Admin, Author
RF 09	Editar Modelo	El sistema debe permitir la edición de los modelos. En el caso del author, podrá editar sólo los modelos ingresados por él, a diferencia del administrador quien podrá editar cualquier modelo ingresado.	Admin, Author
RF 10	Editar Diseño	El sistema debe permitir la edición de los diseños. En el caso del author, podrá editar sólo los diseños ingresados por él, a diferencia del administrador quien podrá editar cualquier diseño ingresado.	Admin, Author
RF 11	Eliminar Modelo	El sistema debe permitir la eliminación de los modelos. En el caso del author, podrá eliminar sólo los modelos ingresados por él, a diferencia del administrador quien podrá eliminar cualquier modelo ingresado.	Admin, Author
RF 12	Eliminar Diseño	El sistema debe permitir la eliminación de los diseños. En el caso del author, podrá eliminar sólo los diseños ingresados por él, a diferencia del administrador quien podrá eliminar cualquier diseño ingresado.	Admin, Author
RF 13	Registrar Autor	El sistema debe permitir el registro de autores.	Admin
RF 14	Ver Autores	El sistema debe permitir la visualización de los autores registrados.	Admin
RF 15	Editar Autor	El sistema debe permitir la edición de los autores registrados.	Admin
RF 16	Eliminar Autor	El sistema debe permitir la eliminación de los autores registrados.	Admin

Tabla 3.2: Requerimientos Funcionales RF08 - RF16

3.1.2. Requerimientos No Funcionales

ID	Nombre	Descripción
RNF 01	Obtención de datos	El sistema debe estar poblado con datos recopilados desde los web server Bcl Fold, I-Tasser, Psipred y Robetta.
RNF 02	Accesibilidad	La interfaz debe ser algo simple, con un diseño de fácil uso y accesible para todo tipo de usuario.
RNF 03	Usabilidad	El sistema debe ser intuitivo, presentar funciones y menús sencillos.

Tabla 3.3: Requerimientos No Funcionales RNF01 - RNF03

Para cumplir con los requerimientos funcionales y no funcionales solicitados, se deben utilizar un conjunto de herramientas que se describen a en la sección 4 de Implementación. Con la herramienta Umbrello mencionada en la sección 4.1.9 de Implementación, se realizan diversos diagramas para dar mayor explicación al sistema. La siguiente sección explica la creación del software y la interacción del usuario con el sistema mediante estos diagramas.

3.2. Casos de uso

En esta sección se mostrará la interacción del usuario con el sistema y las acciones que realizarán cada uno de ellos.

3.2.1. Actores

Los actores son quienes interactuan con el sistema, y se representan en la Figura 3.1.

Figura 3.1: Actores

Como se puede apreciar en la Figura 3.1 hay tres tipos de usuarios para nuestro software, los que se describen a continuación en la tabla 3.4:

Actor	Descripción
Administrador	Persona encargada de la gestión del sistema y de administrar la totalidad de la información que se almacene, puede acceder a todas las opciones, entre ellas el registrar Autores, modificar cualquier información almacenada o eliminarla.
Autor	Persona que realiza las inserciones de modelos o diseños, puede modificar sus propios ingresos o eliminarlos, además de poder visualizar toda la información y descargarla.
Web	Persona externa que puede visualizar y descargar la información almacenada en el sistema.

Tabla 3.4: Diccionario de Actores.

3.2.2. Casos de Uso

En los casos de uso presentados a continuación se visualizan los usuarios que pueden ingresar al sistema, el Administrador y el Autor (Ver Figura 3.2). Éstos cuentan con un registro dentro de la base de datos y pueden acceder a diversas opciones dentro del sistema, a diferencia del Usuario Web, quien sólo será un observador dentro de la página, podrá revisar la información ingresada e incluso descargarla, pero no podrá ingresar con un login, ni insertar modelos o diseños, editar o eliminar datos (Ver Figura 3.3).

3.2.2.1. Caso de uso Ingresar al sistema

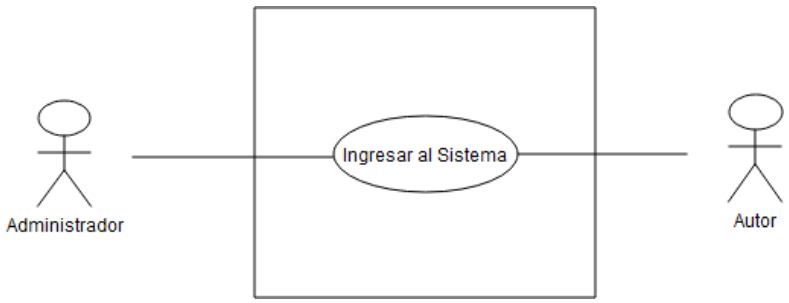

Figura 3.2: Ingresar al Sistema

1- Ingresar al Sistema	
Campo Caso de uso	***Descripción***
Actores	Administrador, Autor.
Descripción del caso de uso	El Administrador o Autor, realizan el inicio de sesión al sistema para poder tener acceso a sus respectivos privilegios.
Pre-Condición	Administrador y Autor deben estar registrados en la base de datos.
Post-Condición	Administrador y Autor realizan el inicio de sesión de manera exitosa.
Asociación de caso de uso	No presenta.
Resumen de entradas	Rut, contraseña.
Resumen de salidas	Ingreso Exitoso.
Indice de Dificultad	Bajo.
Curso normal de eventos	
Acción del Actor	***Respuesta del sistema***
El caso de uso comienza cuando el usuario ingresa su rut, contraseña y presiona Ingresar.	El sistema valida la información, verifica se encuentre en la base de datos y que tipo de usuario esta ingresando.
El usuario accede al sistema.	El sistema entrega el mensaje de haber ingresado con éxito y muestra las opciones del usuario.
Curso alternativo de eventos	
El usuario no esta registrado e intenta ingresar al sistema.	El sistema arroja mensaje de error diciendo que no se encuentra registrado.
El usuario ingresa una contraseña incorrecta.	El sistema arroja mensaje de error diciendo que la contraseña es incorrecta.
Nota de Caso de uso	

Tabla 3.5: Descripción Caso de uso - Ingresar al Sistema

3.2.2.2. Caso de uso Visibilidad del Sistema

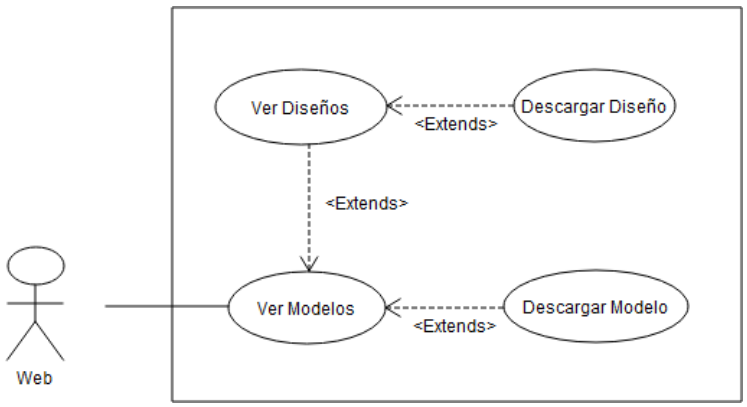

Figura 3.3: Casos de Uso Visibilidad del Sistema

2- Ver Modelos	
Campo Caso de uso	***Descripción***
Actores	Administrador, Autor, Web.
Descripción del caso de uso	El administrador, el autor y el usuario web pueden visualizar los modelos ingresados a la base de datos.
Pre-Condición	Sin pre-condición.
Post-Condición	Visualizar los modelos de forma exitosa.
Asociación de caso de uso	Para los tres usuarios se asocian los casos de uso: Descargar Modelo, Ver Diseños. Para autor se asocian los casos de uso: Ingresar diseño, Editar Modelo y Eliminar Modelo, sólo a los ingresados por él. Para administrador se asocian los casos de uso: Ingresar diseño, Editar Modelo, Eliminar Modelo, a todos.
Resumen de entradas	Presionar la opción “Ver Modelos”.
Resumen de salidas	Visualización Exitosa de los modelos.
Indice de Dificultad	Bajo.
Curso normal de eventos	
Acción del Actor	***Respuesta del sistema***
El caso de uso comienza cuando el administrador, el autor o el usuario web seleccionan la opción *Ver Modelos*.	El sistema redireccionará a Ver Modelos, mostrando todos los modelos que se encuentren en el sistema.
El administrador, el autor o el usuario web podrán buscar un modelo en particular, para ello ingresan el nombre en la barra de búsqueda y presionan *Buscar*.	El sistema buscará el nombre ingresado en la base de datos y lo mostrará.
El administrador, el autor o el usuario web podrán volver a ver todos los modelos, para ellos seleccionan la opción *Ver Todos*.	El sistema buscará todos los modelos en la base de datos y los mostrará.
Curso alternativo de eventos	
El administrador, el autor o el usuario web ingresan un nombre a buscar que no esté ingresado.	El sistema no mostrará modelos.
Nota de Caso de uso	

Tabla 3.6: Descripción Caso de uso - Ver Modelos

3- Ver Diseños	
Campo Caso de uso	***Descripción***
Actores	Administrador, Autor, Web.
Descripción del caso de uso	El administrador, el autor y el usuario web pueden visualizar los diseños de cada modelo ingresado a la base de datos.
Pre-Condición	Ingresar a la página web, en la sección Ver modelos.
Post-Condición	Visualizar los diseños de cada modelo de forma exitosa.
Asociación de caso de uso	Para los tres usuarios, se asocian los casos de uso: Descargar Diseños, volver a Ver Modelos. Para autor se asocian los casos de uso: Editar Diseño y Eliminar Diseño, sólo a los ingresados por él. Para administrador se asocian los casos de uso: Editar Diseño, Eliminar Diseño, a todos.
Resumen de entradas	Presionar la opción "Ver Diseños" del modelo que se quiera.
Resumen de salidas	Visualización Exitosa de los diseños del modelo seleccionado.
Indice de Dificultad	Bajo.
Curso normal de eventos	
Acción del Actor	***Respuesta del sistema***
El caso de uso comienza cuando el administrador, el autor o el usuario web seleccionan la opción *Ver Diseños*.	El sistema redireccionará a *Ver Diseños*, mostrando todos los diseños del modelo seleccionado.
El administrador, el autor o el usuario web podrán volver a ver los modelos, para ellos seleccionan la opción *Volver*.	El sistema volverá a todos los modelos y los mostrará.
Curso alternativo de eventos	
Nota de Caso de uso	

Tabla 3.7: Descripción Caso de uso - Ver Diseños

4- Descargar Modelo	
Campo Caso de uso	***Descripción***
Actores	Administrador, Autor, Web.
Descripción del caso de uso	El administrador, el autor y el usuario web pueden descargar los modelos de la base de datos.
Pre-Condición	Ingresar a la página web, en la sección "Ver Modelos".
Post-Condición	Descargar modelos de forma exitosa.
Asociación de caso de uso	Extiende de Ver Modelos.
Resumen de entradas	Presionar la opción *Descargar Modelo*.
Resumen de salidas	Descarga de modelo exitosa.
Indice de Dificultad	Bajo.
Curso normal de eventos	
Acción del Actor	***Respuesta del sistema***
El caso de uso comienza cuando el administrador, el autor o el usuario web seleccionan la opción *Descargar Modelo*.	El sistema descargará el modelo seleccionado.
Curso alternativo de eventos	
Nota de Caso de uso	

Tabla 3.8: Descripción Caso de uso - Descargar Modelo

5- Descargar Diseño	
Campo Caso de uso	***Descripción***
Actores	Administrador, Autor, Web.
Descripción del caso de uso	El administrador, el autor y el usuario web pueden descargar los diseños de cada modelo de la base de datos.
Pre-Condición	Ingresar a la página web, en la sección *Ver Diseños* del modelo que se quiera.
Post-Condición	Descargar diseños de forma exitosa.
Asociación de caso de uso	Extiende de Ver Diseños.
Resumen de entradas	Presionar la opción *Descargar Diseño*.
Resumen de salidas	Descarga de diseño exitosa.
Indice de Dificultad	Bajo.
Curso normal de eventos	
Acción del Actor	***Respuesta del sistema***
El caso de uso comienza cuando el administrador, el autor o el usuario web seleccionan la opción *Descargar Diseño*.	El sistema descargará el diseño seleccionado.
Curso alternativo de eventos	
Nota de Caso de uso	

Tabla 3.9: Descripción Caso de uso - Descargar Diseño

En el caso de uso de la Figura 3.4 se representan las funciones que el Autor tiene en el sistema, dentro de las cuales se incluyen ingresar modelos y diseños a la base de datos, la opción de editar lo ingresado, eliminar estos modelos o diseños que pertenecen al usuario y visualizar tanto los propios como los ingresados por los demás.

3.2.2.3. Caso de uso Usuario Autor

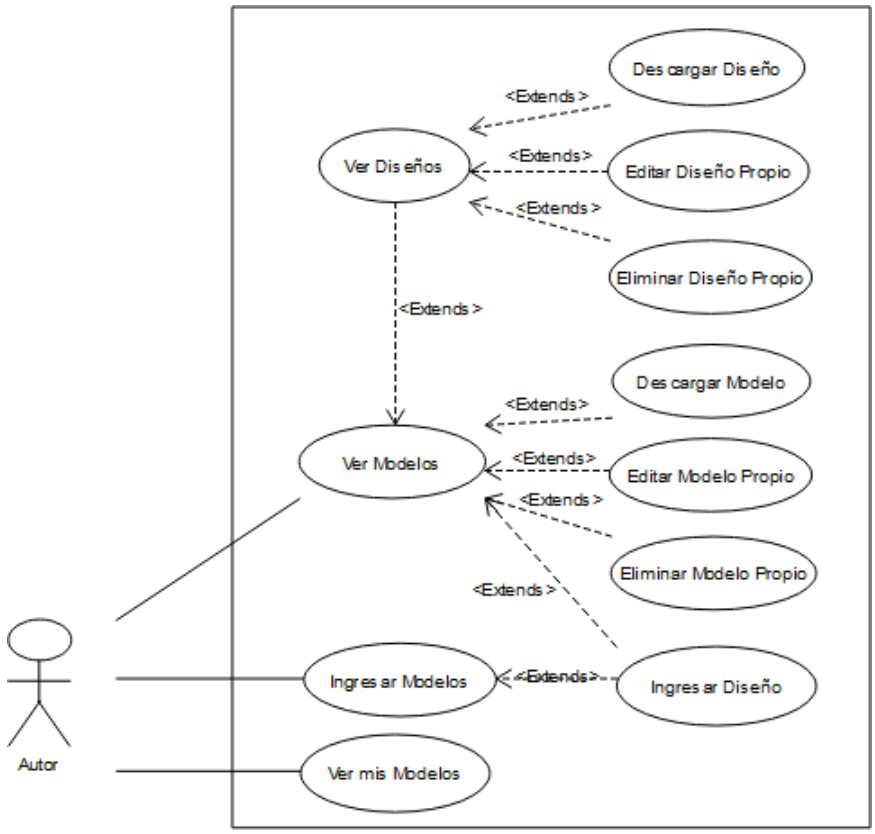

Figura 3.4: Casos de Uso Usuario Autor

6- Ver mis Modelos	
Campo Caso de uso	***Descripción***
Actores	Administrador, Autor.
Descripción del caso de uso	El administrador y el autor pueden visualizar sus modelos ingresados a la base de datos.
Pre-Condición	Sin pre-condición.
Post-Condición	Visualizar los modelos propios de forma exitosa.
Asociación de caso de uso	No presenta.
Resumen de entradas	Presionar la opción *Ver Mis Modelos*.
Resumen de salidas	Visualización Exitosa de los modelos propios.
Indice de Dificultad	Bajo.
Curso normal de eventos	
Acción del Actor	***Respuesta del sistema***
El caso de uso comienza cuando el administrador o el autor seleccionan la opción *Ver Mis Modelos*.	El sistema redireccionará a Ver Modelos, mostrando todos los modelos ingresados por el usuario logueado.
El administrador o el autor podrán buscar un modelo en particular, para ello ingresan el nombre en la barra de búsqueda y presionan *Buscar*.	El sistema buscará el nombre ingresado en la base de datos y lo mostrará.
El administrador o el autor podrán volver a ver todos los modelos, para ellos seleccionan la opción *Ver Todos*.	El sistema buscará todos los modelos en la base de datos y los mostrará.
Curso alternativo de eventos	
El administrador, el autor o el usuario web ingresan un nombre a buscar que no esté ingresado.	El sistema no mostrará modelos.
Nota de Caso de uso	

Tabla 3.10: Descripción Caso de uso - Ver mis Modelos

7- Ingresar Modelo	
Campo Caso de uso	***Descripción***
Actores	Administrador, Autor.
Descripción del caso de uso	El administrador o el autor pueden insertar modelos en la base de datos.
Pre-Condición	El administrador o autor deben estar logueados en el sistema.
Post-Condición	Insertar modelo de forma exitosa.
Asociación de caso de uso	No presenta.
Resumen de entradas	Pdb link, autor, nombre proteína, archivo string atom.
Resumen de salidas	Inserción de modelo exitosa.
Indice de Dificultad	Bajo.
Curso normal de eventos	
Acción del Actor	***Respuesta del sistema***
El caso de uso comienza cuando el administrador o el autor seleccionan la opción *Ingresar Modelo*.	El sistema despliega el formulario para ingresar modelo.
El administrador o el autor debe llenar el formulario con los datos solicitados y presionar el botón *Siguiente*.	El sistema valida los datos ingresados, los registra en la base de datos y pasa a la siguiente ventana para ingresar un diseño al modelo.
Curso alternativo de eventos	
El administrador o el autor no ingresa todos los datos al formulario.	El sistema arroja mensaje de error pidiendo los campos obligatorios.
Nota de Caso de uso	

Tabla 3.11: Descripción Caso de uso - Ingresar Modelo

8- Ingresar Diseño	
Campo Caso de uso	***Descripción***
Actores	Administrador, Autor.
Descripción del caso de uso	El administrador o el autor pueden insertar diseños en la base de datos.
Pre-Condición	El administrador o el autor deben estar logueados en el sistema. Ingresar a la página web, en la sección siguiente a *Insertar Modelos* o bien en la sección *Ver Modelos*.
Post-Condición	Insertar diseño de forma exitosa.
Asociación de caso de uso	Extiende de Ingresar Modelos o de Ver Modelos.
Resumen de entradas	Archivo pdb, string atom design, topología, categoría del diseño, funciona, server, parametros (ranking, parametros por defecto), rmsd.
Resumen de salidas	Inserción de diseño exitosa.
Indice de Dificultad	Bajo.
Curso normal de eventos	
Acción del Actor	***Respuesta del sistema***
El caso de uso comienza cuando el administrador o el autor seleccionan la opción *Ingresar Diseño* en la sección de *Ver Modelos* o cuando luego de haber ingresado un modelo se llega a la ventana de *Ingresar Diseño*.	El sistema despliega el formulario para ingresar diseño.
El administrador o el autor debe llenar el formulario con los datos solicitados y presionar el botón *Guardar*.	El sistema valida los datos ingresados, los registra en la base de datos y da la opción de ingresar otro diseño al modelo.
Curso alternativo de eventos	
El administrador o el autor no ingresa todos los datos al formulario.	El sistema arroja mensaje de error pidiendo los campos obligatorios.
Nota de Caso de uso	

Tabla 3.12: Descripción Caso de uso - Ingresar Diseño

9- Editar Modelo	
Campo Caso de uso	***Descripción***
Actores	Administrador, Autor.
Descripción del caso de uso	El administrador puede modificar cualquiera de los modelos ya ingresados a la base de datos. El autor puede modificar sólo los modelos ingresados por el mismo en la base de datos.
Pre-Condición	El administrador o el autor deben estar logueados en el sistema. Ingresar a la página web, en la sección *Ver Modelos*.
Post-Condición	Editar modelo de forma exitosa.
Asociación de caso de uso	Extiende de Ver Modelos.
Resumen de entradas	Presionar el botón *Editar Modelo* e ingresar los datos que se requieran.
Resumen de salidas	Edición de diseño exitosa.
Indice de Dificultad	Bajo.
Curso normal de eventos	
Acción del Actor	***Respuesta del sistema***
El caso de uso comienza cuando el administrador o el autor seleccionan la opción *Editar Modelo* en la sección de *Ver Modelos*.	El sistema despliega el formulario para ingresar modelo, con los datos ingresados originalmente.
El administrador o el autor debe modificar lo deseado en el formulario y presionar el botón *Siguiente*.	El sistema valida los datos ingresados, los modifica en la base de datos y da la opción de ingresar un diseño al modelo.
Curso alternativo de eventos	
El administrador o el autor no ingresa todos los datos al formulario.	El sistema arroja mensaje de error pidiendo los campos obligatorios.
Nota de Caso de uso	

Tabla 3.13: Descripción Caso de uso - Editar Modelo

10- Editar Diseño	
Campo Caso de uso	***Descripción***
Actores	Administrador, Autor.
Descripción del caso de uso	El administrador puede modificar cualquiera de los diseños ya ingresados a la base de datos. El autor puede modificar sólo los diseños ingresados por el mismo en la base de datos.
Pre-Condición	El administrador o el autor deben estar logueados en el sistema. Ingresar a la página web, en la sección *Ver Diseños*.
Post-Condición	Editar diseño de forma exitosa.
Asociación de caso de uso	Extiende de Ver Diseños.
Resumen de entradas	Presionar el botón "Editar Diseño" e ingresar los datos que se requieran.
Resumen de salidas	Edición de diseño exitosa.
Indice de Dificultad	Bajo.
Curso normal de eventos	
Acción del Actor	***Respuesta del sistema***
El caso de uso comienza cuando el administrador o el autor seleccionan la opción *Editar Diseño* en la sección de *Ver Diseños*.	El sistema despliega el formulario para ingresar diseño, con los datos ingresados originalmente.
El administrador o el autor debe modificar lo deseado en el formulario y presionar el botón *Guardar*.	El sistema valida los datos ingresados, los modifica en la base de datos y da la opción de ingresar otro diseño al modelo.
Curso alternativo de eventos	
El administrador o el autor no ingresa todos los datos al formulario.	El sistema arroja mensaje de error pidiendo los campos obligatorios.
Nota de Caso de uso	

Tabla 3.14: Descripción Caso de uso - Editar Diseño

11- Eliminar Modelo	
Campo Caso de uso	***Descripción***
Actores	Administrador, Autor.
Descripción del caso de uso	El administrador puede eliminar cualquiera de los modelos ya ingresados a la base de datos. El autor puede eliminar sólo los modelos ingresados por el mismo en la base de datos.
Pre-Condición	El administrador o el autor deben estar logueados en el sistema. Ingresar a la página web, en la sección *Ver Modelos*.
Post-Condición	Eliminar modelo de forma exitosa.
Asociación de caso de uso	Extiende de Ver Modelos.
Resumen de entradas	Presionar el botón “Eliminar Modelo”.
Resumen de salidas	Eliminación de modelo exitosa.
Indice de Dificultad	Bajo.
Curso normal de eventos	
Acción del Actor	***Respuesta del sistema***
El caso de uso comienza cuando el administrador o el autor seleccionan la opción *Eliminar Modelo* en la sección de *Ver Modelos*.	El sistema eliminara de la base de datos el modelo seleccionado y arrojara un mensaje de eliminación exitosa.
Curso alternativo de eventos	
Nota de Caso de uso	

Tabla 3.15: Descripción Caso de uso - Eliminar Modelo

12- Eliminar Diseño	
Campo Caso de uso	***Descripción***
Actores	Administrador, Autor.
Descripción del caso de uso	El administrador puede eliminar cualquiera de los diseños ya ingresados a la base de datos. El autor puede eliminar sólo los diseños ingresados por el mismo en la base de datos.
Pre-Condición	El administrador o el autor deben estar logueados en el sistema. Ingresar a la página web, en la sección *Ver Diseños*.
Post-Condición	Eliminar diseño de forma exitosa.
Asociación de caso de uso	Extiende de Ver Diseños.
Resumen de entradas	Presionar el botón *Eliminar Diseño*.
Resumen de salidas	Eliminación de diseño exitosa.
Indice de Dificultad	Bajo.
Curso normal de eventos	
Acción del Actor	***Respuesta del sistema***
El caso de uso comienza cuando el administrador o el autor seleccionan la opción *Eliminar Diseño* en la sección de *Ver Diseños*.	El sistema eliminara de la base de datos el diseño seleccionado y arrojara un mensaje de eliminación exitosa.
Curso alternativo de eventos	
Nota de Caso de uso	

Tabla 3.16: Descripción Caso de uso - Eliminar Diseño

En la Figura 3.5 se representan las funciones que el Administrador tiene en el sistema, este actor tiene la mayor responsabilidad dentro del software, ya que cuenta con la responsabilidad de gestionar toda la información dentro de éste. El nivel de conocimientos técnicos requeridos para este rol, es un nivel medio. No es necesario tener mucho conocimiento en cuanto a computación, pero si debe conocer el sistema en su totalidad ya que posee acceso a funcionalidades de alto riesgo, como la eliminación.

3.2.2.4. Caso de uso Usuario Administrador

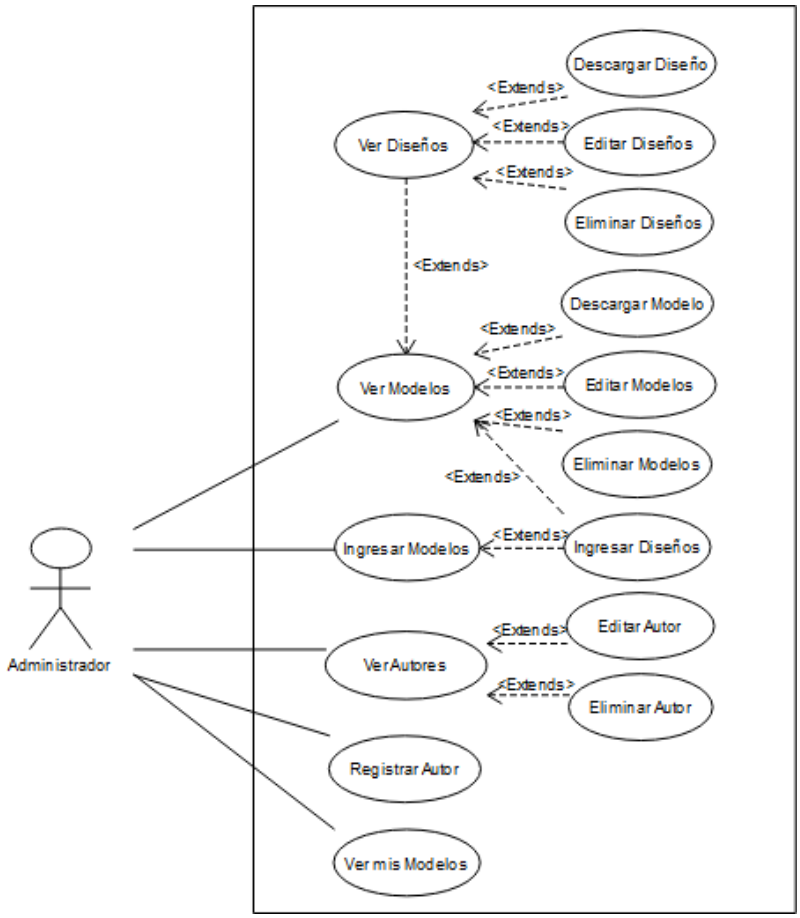

Figura 3.5: Casos de Uso Usuario Administrador

13- Registrar Autor	
Campo Caso de uso	***Descripción***
Actores	Administrador.
Descripción del caso de uso	El administrador realiza el registro del autor, para que pueda acceder a los contenidos que sólo son visibles para usuarios registrados.
Pre-Condición	El administrador debe estar logueado en el sistema. El actor a registrar no debe estar registrado con anterioridad.
Post-Condición	El administrador realiza el registro de forma exitosa.
Asociación de caso de uso	No presenta.
Resumen de entradas	Rut, nombre, e-mail, contraseña, verificación de contraseña.
Resumen de salidas	Registro Exitoso.
Indice de Dificultad	Bajo.
Curso normal de eventos	
Acción del Actor	***Respuesta del sistema***
El caso de uso comienza cuando el administrador selecciona la opción *Registrar Autor*.	El sistema despliega el formulario de registro.
El administrador debe llenar el formulario con los datos del autor y presionar *Registrar*.	El sistema valida los datos ingresados, los registra en la base de datos y envía un mensaje indicando que el registro se ha realizado con éxito.
Curso alternativo de eventos	
El administrador no ingresa todos los datos al formulario.	El sistema arroja mensaje de error pidiendo los campos obligatorios.
El administrador intenta registrar un autor que ya se encuentra en el sistema.	El sistema arroja mensaje de error diciendo que el autor ya se encuentra registrado.
Nota de Caso de uso	

Tabla 3.17: Descripción Caso de uso - Registrar Actor

14- Ver Autores	
Campo Caso de uso	***Descripción***
Actores	Administrador.
Descripción del caso de uso	El administrador puede visualizar todos los autores registrados hasta el momento.
Pre-Condición	El administrador debe estar logueado en el sistema.
Post-Condición	Visualizar autores de forma exitosa.
Asociación de caso de uso	No presenta.
Resumen de entradas	Presionar el botón “Ver Autores”.
Resumen de salidas	Visualización de autores exitosa.
Indice de Dificultad	Bajo.
Curso normal de eventos	
Acción del Actor	***Respuesta del sistema***
El caso de uso comienza cuando el administrador selecciona la opción *Ver Autores*.	El sistema mostrará un listado con todos los autores registrados en la base de datos.
Curso alternativo de eventos	
Nota de Caso de uso	

Tabla 3.18: Descripción Caso de uso - Ver Autores

15- Editar Autor	
Campo Caso de uso	***Descripción***
Actores	Administrador.
Descripción del caso de uso	El administrador puede editar cualquiera de los autores registrados en la base de datos.
Pre-Condición	El administrador debe estar logueado en el sistema.
Post-Condición	Edición de autor de forma exitosa.
Asociación de caso de uso	Extiende de Ver Autores.
Resumen de entradas	Presionar el botón *Editar Autor*.
Resumen de salidas	Edición de autor exitosa.
Indice de Dificultad	Bajo.
Curso normal de eventos	
Acción del Actor	***Respuesta del sistema***
El caso de uso comienza cuando el administrador selecciona la opción *Editar Autor*.	El sistema despliega el formulario para ingresar autor, con los datos ingresados originalmente.
El administrador debe modificar lo deseado en el formulario y presionar el botón *Guardar*.	El sistema valida los datos ingresados y los modifica en la base de datos.
Curso alternativo de eventos	
El administrador no ingresa todos los datos al formulario.	El sistema arroja mensaje de error pidiendo los campos obligatorios.
Nota de Caso de uso	

Tabla 3.19: Descripción Caso de uso - Editar Autor

16- Eliminar Autor	
Campo Caso de uso	***Descripción***
Actores	Administrador.
Descripción del caso de uso	El administrador puede eliminar cualquiera de los autores registrados en la base de datos.
Pre-Condición	El administrador debe estar logueado en el sistema.
Post-Condición	Eliminación de autor de forma exitosa.
Asociación de caso de uso	Extiende de Ver Autores.
Resumen de entradas	Presionar el botón *Editar Autor*.
Resumen de salidas	Eliminación de autor exitosa.
Indice de Dificultad	Bajo.
Curso normal de eventos	
Acción del Actor	***Respuesta del sistema***
El caso de uso comienza cuando el administrador selecciona la opción *Eliminar Autor*.	El sistema eliminara de la base de datos el autor seleccionado y arrojara un mensaje de eliminación exitosa.
Curso alternativo de eventos	
Nota de Caso de uso	

Tabla 3.20: Descripción Caso de uso - Eliminar Autor

3.3. Diagrama de clases

Ya explicada la interacción de los usuarios con el sistema y las diferentes funcionalidades que éstos contemplan, se da paso a describir la estructura del software. Esto mediante un diagrama de clases que muestra las clases, atributos y operaciones (o métodos) del sistema.

Como se esta trabajando con una base de datos NoSQL, no se cuenta con tablas como en las bases de datos relacionales, sino con colecciones. En la Figura 3.6 se representa la colección Usuarios, donde se almacenarán los usuarios administradores y autores, cada usuario registrado será un documento guardado con una id única y sus respectivos datos que son el rut, nombre completo, correo electrónico, contraseña y un rol. Este rol será para diferenciar a los administradores de los autores, como se explicó en el caso de uso Registrar Autor, el administrador será quien registre a los autores pero el administrador no podrá registrar administradores. Estos están establecidos y no pueden agregarse de la misma forma que un autor.

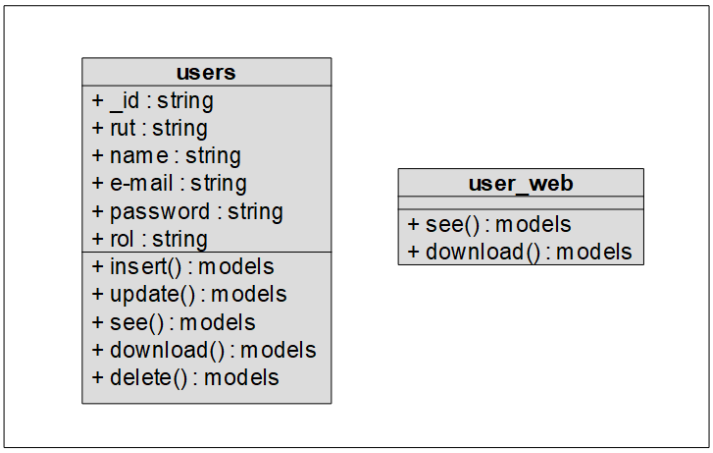

Figura 3.6: Diagrama de Clases Usuarios

La colección Usuarios de la Figura 3.6 muestra las operaciones que están asociadas a los usuarios administrador y autor, estas son insert(), update(), see(), download() y delete(). Estos métodos fueron mencionados como casos de uso en la sección anterior, junto con el papel que juega el atributo *rol* del usuario en estas operaciones.

En la Figura 3.6 también se muestra una tabla llamada *User web* la cual no representa una colección en la base de datos, sino que representa a todos esos usuarios que podrán visitar la página web, sin la necesidad de estar registrados podrán acceder a la información almacenada, tanto para visualizarla como para descargarla.

Dentro de la base de datos se encuentra una segunda colección llamada Modelos, donde se almacenan todos los modelos que han sido o serán ingresados al sistema. Cada modelo ingresado es un documento guardado con una id única.

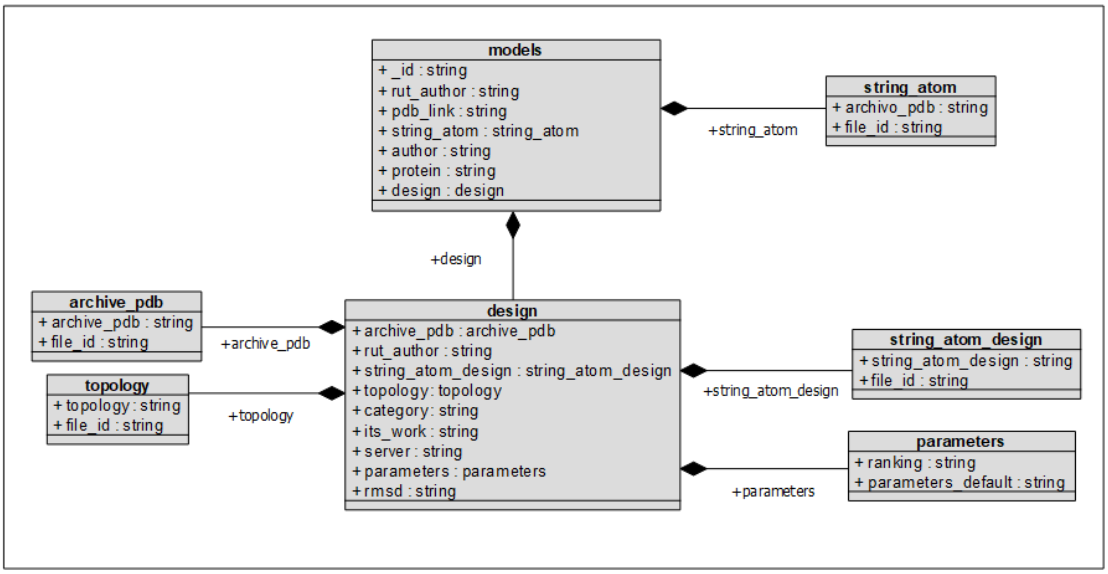

Figura 3.7: Diagrama de Clases Modelos

En la Figura 3.7 se pueden apreciar los atributos de cada documento, estos son:

- rut_author: Este atributo corresponde al rut de autor que ingreso el modelo, este dato es importante dentro del sistema pero no es visualizado por el usuario, se utiliza para dar el permiso de edición y eliminación sólo al autor que lo ingresó.
- pdb_link: Este atributo corresponde al link de la proteína alojada en el Protein Data Bank, para así poder acceder a mayor contenido o detalle de ésta.
- author/s: Este atributo corresponde a el o los nombres de los autores que detectaron el complejo proteico y determinaron su estructura tridimensional.
- protein: Corresponde al nombre de la proteína que se modeló.
- string_atom: Corresponde a un archivo ingresado a la base de datos con extensión .pdb que contiene la información de la proteína.

Éste último atributo es un documento dentro del documento que contiene 2 atributos más: *archive_pdb* es el nombre del archivo y *file_id* corresponde a la id del archivo guardado. Lo mismo ocurre con el atributo design, es un documento dentro de un documento que contiene sus propios atributos como:

- rut_author: Este atributo corresponde al rut de autor que ingreso el diseño, este dato es importante dentro del sistema pero no es visualizado por el usuario, se utiliza para dar el permiso de edición y eliminación sólo al autor que lo ingresó.
- category: Corresponde a la categoría del diseño ingresado.
- its_work: Este atributo corresponde a seleccionar la opción de si funcionó o no el intento.
- server: Corresponde al nombre del server que se utilizó para la predicción de la estructura.
- rmsd: Corresponde al cálculo de igualdad de proteínas.
- archive pdb: Corresponde a un archivo con extensión .pdb que se almacenará en la base de datos, este atributo es un documento dentro del documento que almacena

los atributos archive_pdb y file_id que son el nombre del archivo y la id respectivamente.

- string_atom_design: Corresponde a un archivo con extensión .pdb que se almacenará en la base de datos, contiene los ATOM del diseño, este atributo es un documento dentro del documento que almacena los atributos *string_atom_design* y *file_id* que son el nombre del archivo y la id respectivamente.
- topology: Corresponde a un archivo con extensión .pdf que se almacenará en la base de datos, este atributo es un documento dentro del documento que almacena los atributos *topology* y *file_id* que son el nombre del archivo y la id respectivamente.
- parameters: Este atributo es un documento dentro del documento que almacena los atributos *ranking* y *parameters_default* que corresponden al número de ranking que pertenece el diseño dentro del modelo, es decir, si fue el intento 1, 2, 3, etc. y a parámetros por defecto que será un cuadro de texto donde podrá ingresar notas o comentarios.

3.4. Diagramas de secuencia

Para dar mayor explicación al Diagrama de clases, precisamente al ingreso de modelos y diseños, se utilizan dos diagramas de secuencia. Éstos diagramas son escogidos por la interacción del usuario con el ingreso de información, ya que ayudan a modelar como el usuario administrador o autor, se relacionan con el sistema. Los diagramas de las Figuras 3.8 y 3.9 nos muestra tres objetos en la parte superior, *Ventana ingreso modelo, un ingreso de modelo y base de datos*, unidos por líneas horizontales que representan los eventos de forma cronológica, es decir de la parte superior a la inferior.

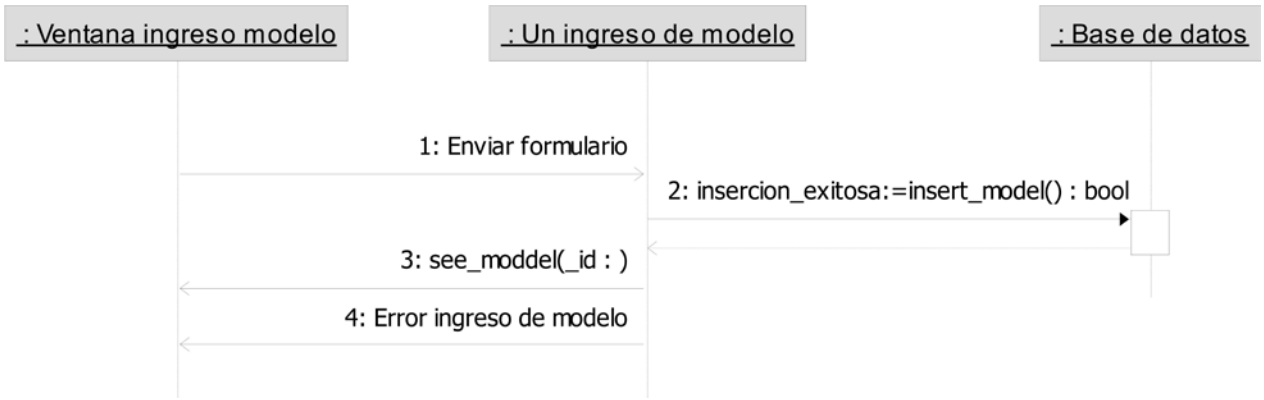

Figura 3.8: Ingresar Modelo

La Figura 3.8 muestra el Ingreso del modelo, inicia cuando el administrador o el autor tiene acceso a la ventana de ingreso de modelo y llena el formulario respectivo, al presionar el botón *siguiente* la información se envía a la base de datos para almacenarla. Si la inserción es exitosa entonces redirecciona a *Ver Modelos*, en caso contrario envía mensaje de error.

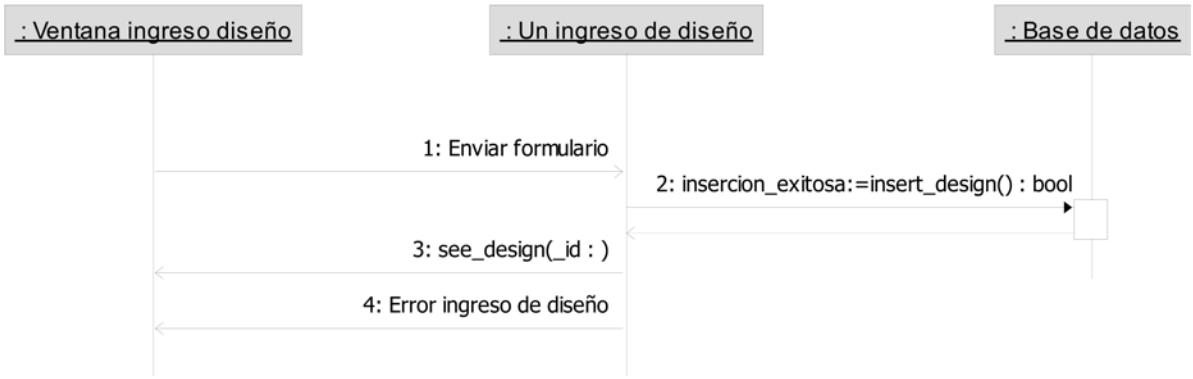

Figura 3.9: Ingresar Diseño

La Figura 3.9 muestra el Ingreso del diseño, inicia cuando el administrador o el autor tiene acceso a la ventana de ingreso de diseño y llena el formulario respectivo, al presio-

nar el botón *guardar* la información se envía a la base de datos para almacenarla. Si la inserción es exitosa entonces redirecciona a *Ver Diseños*, en caso contrario envía mensaje de error.

Ya comprendiendo los requerimientos funcionales y no funcionales del sistema, la interacción de los diferentes tipos de usuario, los atributos a insertar en la base de datos y la forma en que se guardarán, la secuencia o interacción al momento de ingresar los datos. Podemos continuar con la implementación donde se da paso a la creación del sistema.

Capítulo 4

Implementación

El primer paso para lograr el objetivo propuesto, es identificar que tipo de base de datos se requiere. De acuerdo a la información de proteínas que se almacena en la base de datos y a la variedad de datos, como direcciones web o archivos que pudiera tener un diseño, se opta por una base de datos no relacional y orientada al almacenamiento de documentos, esto debido a ser *no estructurada* o con esquema libre, además de ser bastante rápido ya que sus datos se encuentran en binario (detallado en la Sección 4.1.4). Ya identificado el tipo de base de datos a utilizar, se debe ver qué sistema operativo es el más apropiado para ésta, lo que depende de la compatibilidad con el sistema de base de datos. De acuerdo a las características necesarias mencionadas, se decide utilizar el gestor de base de datos MongoDB (detallado en la Sección 4.1.4, que sólo proporciona paquetes para versiones de Ubuntu LTS (soporte a largo plazo) de 64 bits, es por este motivo que se opta por Ubuntu 16.04.3 (xenial) (detallado en la Sección 4.1.2 siendo la versión mas actualizada y estable dentro de lo que es compatible con los paquetes proporcionados.

Las herramientas son instaladas en dos ordenadores, siendo uno el servidor y el otro

un laptop personal, desde el cual se realizan pruebas de cada paso antes de hacerlo en el servidor, evitando errores críticos para el sistema final. Para esto se utiliza VirtualBox (detallado en la Sección 4.1.1) que permite la instalación de un sistema operativo virtual, éste puede clonarse de forma completa, ayudando a guardar cambios y tener un punto de partida siempre que sea necesario.

Una vez instalado Ubuntu en ambos equipos, se instala un entorno de escritorio, GNOME Shell (detallado en la Sección 4.1.3), para una mejor interacción con el sistema y así evitar trabajar únicamente desde consola. Posterior a esto se realiza la instalación de MongoDB (detallado en la Sección 4.1.4) y el administrador gráfico Robomongo (detallado en la Sección 4.1.5), éste último cumple con la función de administrar de forma gráfica nuestra base de datos. Luego de verificar que el lenguaje python (detallado en la Sección 4.1.6) esta instalado como se menciona en la instalación B.6, se instala el microframework Flask (detallado en la Sección 4.1.7), la herramienta clave para la creación de la sistema web.

4.1. Herramientas a utilizar

Para la creación de nuestro sistema se requiere el sistema operativo Ubuntu 16.04.3, ya sea en el laptop personal así como en el servidor, pero debido a que el equipo personal cuenta con *Windows 10* también se utiliza como complemento de la investigación, entonces se cuenta con dos sistemas operativos *Windows 10 (64 bits)*/*Ubuntu 16.04.3*. El equipo personal cuenta con un procesador Intel Core i5 1.7GHz, un disco duro de 500 GB y una RAM de 4 GB DDR3. Éstas características cumplen con lo mínimo requerido para poder realizar el software sin inconvenientes.

4.1.1. Oracle VM VirtualBox 5.1.30

VirtualBox es un software de virtualización x86 y amd64/intel64, desarrollado por Oracle Corporation para uso empresarial y domestico. Esta aplicación permite la instalación de sistemas operativos adicionales conocidos como *invitados* dentro de otro sistema operativo *anfitrión*, cada uno con su propio ambiente virtual. Actualmente existe la versión privativa Oracle VM VirtualBox, que es gratuita únicamente bajo uso personal o de evaluación y la versión Open Source, VirtualBox OSE, que es software libre, sujeta a la licencia GPL [9].

VirtualBox se ejecuta en servidores Linux, Macintosh, Microsoft Windows y Solaris y es compatible con variados sistemas invitados, entre ellos Windows (NT 4.0, 2000, XP, Server 2003, Vista, Windows 7, Windows 8, Windows 10), DOS/Windows 3.x, Linux (2.4, 2.6, 3.xy 4.x), Solaris y OpenSolaris, OS/2 y OpenBSD [9].

Se utiliza VirtualBox con la finalidad de poder crear el sistema en un ambiente virtual de un ordenador personal, esto para trabajar y hacer pruebas antes de pasarlo al servidor. Con esto se evita, en caso de un error, el reinstalar todo nuevamente en el servidor, también el particionar el disco en el ordenador personal o la instalación de otro sistema operativo en éste. Esto gracias a que VirtualBox permite clonar un sistema invitado completamente, lo que es bastante útil para guardar avances del sistema sin temor a tener que reinstalar un sistema operativo o reinstalar todo nuevamente en el servidor.

Para lograr el objetivo propuesto, se utiliza como anfitrión el sistema operativo Windows 10 (64 bits) y como sistema invitado Ubuntu 16.04.3 (Ver Instalación 4.1.1).

4.1.2. Ubuntu 16.04.3

Ubuntu es una distribución de GNU/Linux de código abierto, es un sistema operativo enfocado a computadoras personales, pero también puede utilizarse en servidores. Su nombre proviene de la lengua africana zulú, y se traduce como *humanidad a otros* [27].

La versión primaria de Ubuntu utiliza *GNOME*, una interfaz gráfica de usuario(GUI) que pretende hacer que linux sea fácil de utilizar para los no programadores simulando un entorno similar a Windows [27].

Ubuntu es un software libre y a la vez gratuito, es decir, permite usarlo sin ningún tipo de restricción y además no tiene costo, lo que facilita el desarrollo de cualquier tipo de sistema o instalación de cualquier programa.

Se utiliza Ubuntu 16.04.3 por la compatibilidad con los paquetes de MongoDB, ya que MongoDB sólo proporciona paquetes para versiones de Ubuntu LTS (soporte a largo plazo) de 64 bits (Ver Instalación 4.1.2).

4.1.3. GNOME Shell

Una vez instalado el sistema operativo Ubuntu 16.04.3 en el servidor y en la máquina virtual, podemos comenzar a trabajar e instalar las herramientas necesarias para creación de nuestro sistema. Una de las herramientas que utilizaremos es *GNOME Shell*, con ella se dará una interfaz gráfica al sistema operativo y se evitará trabajar únicamente desde consola, esto ayuda a que linux sea más sencillo de utilizar brindando un ambiente más agradable de trabajo.

GNOME Shell es un tipo de *entorno de escritorio* compuesto completamente de software libre, fue desarrollado por los mexicanos Federico Mena y Miguel de Icaza, para sistemas operativos como GNU/Linux, Unix y derivados. Este entorno nació como una

alternativa al famoso KDE y su objetivo es crear un sistema de escritorio para el usuario final que sea completo, libre y fácil de usar mediante las bibliotecas gráficas GTK [19] (Ver Instalación B.3).

4.1.4. MongoDB

MongoDB es un sistema de base de datos NoSQL de código abierto, escrito en C++ y orientado al almacenamiento de documentos. Esto quiere decir que en lugar de almacenar los datos en registros, los guarda en documentos. Estos documentos son almacenados en BSON, que es una representación binaria de JSON, lo que significa que los campos pueden variar de un documento a otro y la estructura de datos se puede cambiar con el tiempo [29].

4.1.4.1. NoSQL (Not Only SQL)

El término NoSQL se refiere a un conjunto de bases de datos que se diferencian de las bases de datos convencionales, estos tipos de bases de datos no cumplen con el esquema entidad-relación y tampoco usan SQL como lenguaje predeterminado para la realización de consultas. Las bases de datos NoSQL no imponen una estructura de datos en forma de tablas ni relaciones entre ellas, sino que suelen permitir almacenar la información en forma de clave-valor, Mapeo de columnas, Documentos o Grafos [21].

Características de NoSQL [6]:

- Capacidad para *escalar horizontalmente* A.8 a lo largo de muchos servidores. La escalabilidad se refiere a la capacidad de adaptación y rendimiento del sistema a medida que aumentan los usuarios.
- Capacidad para replicar y distribuir datos a través de múltiples servidores.

- Una sencilla interfaz de nivel de llamada o protocolo (en contraste con una unión SQL).
- Un modelo de concurrencia *glosario:Concurrencia* débil.
- Uso eficiente de índices distribuidos y memoria RAM para almacenamiento de datos.
- La posibilidad de añadir nuevos atributos de forma dinámica.

Las bases de datos no relacionales son capaces de manipular grandes cantidades de información, esto debido a la escalabilidad horizontal, el crecimiento es prácticamente infinito. También reducen el tiempo de desarrollo evitando sentencias SQL complejas.

4.1.4.2. JSON - JavaScript Object Notation

JSON es un formato de texto ligero para el intercambio de datos, nació como alternativa a XML y por su simplicidad ha generado un enorme número de seguidores. Su gran ventaja es que soporta una gran cantidad de tipos de datos, y puede ser leído por cualquier lenguaje de programación [21].

MongoDB almacena documentos en colecciones, los documentos de una misma colección simbolizan tablas dentro de una base de datos, si nos refiriéramos al esquema tradicional relacional. Estos documentos pueden tener esquemas diferentes entre ellos como se muestra a continuación:

```
{
  Nombre: "Alejandro",
  Apellido: "Quezada",
  Edad : 50,
  Casado:true,
  Hijos : [
   {
     Nombre:"Camila",
     Edad:27
   }
  ]
}
```

JSON 4.1: Ejemplo de documento 1

En el primer ejemplo se ha creado un objeto, dentro de él un array llamado *Hijos* con un solo elemento, que contiene un nombre y una edad (Ver Algoritmo 4.1).

En el segundo ejemplo se ha creado un objeto, dentro de él un array llamado *Padres* con dos solo elementos, que contiene un nombre y una edad (Ver Algoritmo 4.2).

El algoritmo 4.1 contiene un esquema diferente al algoritmo 4.2 y están dentro de la misma colección. En una base de datos relacional esto no sería posible, mientras que en MongoDB es totalmente válido.

```
{
  Nombre: "Camila",
  Apellido: "Quezada",
  Edad : 27,
  Casada: false,
  Padres : [
   {
     Nombre:"Alejandro",
     Edad:50
   },
   {
     Nombre:"Marisa",
     Edad:47
   }
  ]
}
```

JSON 4.2: Ejemplo de documento 2

Los tipos de datos que se pueden apreciar en ambos ejemplos son:

- String, están entre comillas simples.
- Array, están entre corchetes.
- Objeto, están entre llaves.

- Número, enteros o float.
- Booleano, true o false.

Luego de analizar las ventajas que tiene una base de datos NoSQL, orientada a documentos como MongoDB, frente a una base de datos relacional podemos pasar la instalación del sistema de base de datos (Ver Instalación B.4).

4.1.5. Robomongo

Robomongo es una herramienta multiplataforma que permite la administración gráfica de nuestras bases de datos, sustituyendo o complementando a la terminal. Se integra con el shell de MongoDB para brindar una gestión moderna, robusta y orientada a la comunidad [36]. Se utilizará para facilitar el uso de la base de datos y la realización de consultas (Ver Instalación B.5).

4.1.6. Python

Python es un lenguaje de programación de propósito general, orientado a objetos e interactivo. Contiene una gran cantidad de librerías, además de módulos, clases, excepciones, tipos de datos dinámicos de muy alto nivel y tipado dinámico. También se puede utilizar como un lenguaje de extensión para aplicaciones escritas en otros lenguajes que necesitan interfaces de automatización o scripting fáciles de usar. Python esta preparado para realizar desde aplicaciones windows a servidores de red o incluso, páginas web. Además de ser un lenguaje gratuito, es un lenguaje interpretado, esto significa que el código de fuente no requiere compilación para su ejecución, lo que brinda rapidez de desarrollo. Cuenta con funciones incorporadas en el propio lenguaje, esto ayuda a realizar una gran cantidad de tareas habituales sin la necesidad de programarlas desde cero. Se puede

desarrollar en diversas plataformas, como Unix, Windows, OS/2, Mac, entre otros [33].

4.1.7. Flask

Es un microframework para Python basado en la biblioteca Werkzeug A.11 y el motor de templates Jinja 2 A.12, y cuenta con licencia BSD de tres cláusulas, esto quiere decir que se puede hacer lo que se desee con Flask, siempre y cuando los derechos de autor se mantengan, las condiciones no se modifiquen y la exención de responsabilidad este presente.

El *micro* en microframework no quiere decir que Flask carezca de funcionalidad, muy por el contrario significa que Flask pretende mantener el núcleo simple pero extensible. Esto quiere decir que Flask solo se conecta con Werkzeug para implementar una aplicación WSGI adecuada y con Jinja2 para manejar las plantillas. También se une a algunos paquetes comunes de bibliotecas estándar, como el registro. Todo lo demás está disponible para extensiones que agregan funcionalidad a su aplicación como si fuera implementada en el Flask mismo, como integración de bases de datos, validación de formularios, manejo de cargas, entre otras [37] (Ver Instalación B.6).

4.1.8. Sublime Text 3

Sublime Text es un editor de código multiplataforma, que permite tener varios documentos abiertos mediante pestañas, e incluso emplear varios paneles en caso de usar más de un monitor. Es una herramienta concebida para programar sin distracciones, debido a su interfaz de color oscuro y la diversidad de colores para la sintaxis. También cuenta con un Minimap, un panel que permite movilizarse rápidamente por el código, dispone de auto-guardado, opciones de personalización, soporta macros, auto completado, entre

muchas funcionalidades más que incluso pueden ser ampliables mediante plugins. Sublime Text soporta un gran número de lenguajes, entre ellos: C, C++, C, CSS, D, Erlang, HTML, Groovy, Haskell, Java, JavaScript, LaTeX, Lisp, Lua, Markdown, Matlab, OCaml, Perl, PHP, Python, R, Ruby, SQL, TCL, Textile and XML [38]. En esta investigación se utiliza para desarrollar y editar código en Python, HTML y CSS.

4.1.9. Umbrello 2.23

Umbrello UML Modeller es un programa de diagramas de lenguaje unificado de modelado (UML) basado en la tecnología de KDE. Permite crear diagramas de software y otros sistemas en un formato estándar, para documentar o diseñar la estructura y comportamiento de sus programas, y así destacar sus principales características. Esta herramienta ayuda al proceso de desarrollo, en especial en fases de análisis y de diseño de software [39]. Umbrello permite usar los siguientes tipos de diagramas [39]:

- Diagrama de clases
- Diagrama de secuencias
- Diagrama de colaboraciones
- Diagrama de caso de uso
- Diagrama de estados
- Diagrama de actividades
- Diagrama de componentes
- Diagrama de despliegue
- Diagrama de relación de entidades

4.1.10. Dropbox

Dropbox es una herramienta que permite alojar diferentes tipos de archivos y sincronizarlos a través de un disco duro virtual en la red, esto permite disponer de un disco duro virtual de forma remota y accesible desde cualquier parte del mundo. Independiente del dispositivo que se utilice o el lugar, se puede acceder a los archivos alojados en la carpeta virtual, incluso es posible, si se desea, añadir colaboradores que puedan participar en la visibilidad o edición, de otra manera quedarían de forma privada. Esta herramienta ofrece seguridad y protección, mediante un cifrado para almacenar los archivos, además cuenta con capas adicionales de protección, como la verificación en dos pasos [15].

4.2. Inicio de componentes y Creación de Página Web

Ya instaladas las herramientas mencionadas en la Sección 4.1, se inicia la construcción de la base de datos en el laptop personal. El primer paso para ello es iniciar MongoDB, se debe ingresar a una terminal y escribir el siguiente comando:

```
$sudo service mongod start
```

Una vez corriendo MongoDB se ingresa a la ubicación de Robomongo mediante consola, con el comando:

```
$cd /usr/local/bin/robomongo/bin
```

Dentro de la ubicación se inicia la herramienta con

```
$./robomongo
```

Este comando abrirá una ventana para iniciar la conexión con MongoDB, una vez iniciada la conexión se puede crear una base de datos nueva de forma gráfica, y agregar coleccio-

nes y documentos. Por el momento se probará con dos colecciones con sólo un documento por cada una, esto evita perder información debido a las pruebas realizadas durante el desarrollo del software, y el tener que reingresar nuevamente los datos al terminarlo. Las colecciones de prueba son *Usuarios* y *Models*, con sólo un documento por cada una como se muestra a continuación (Ver Algoritmo 4.3 y 4.4)

```
{
  "_id": "ObjectId("5a9a2f5170a4fe17d87e70da")",
  "user": "183883062",
  "name": "Nicol Bascunan Faundez",
  "email": "nbascuna@alumnos.ubiobio.cl",
  "password": "1234",
  "rol": "Admin"
}
```

JSON 4.3: Documento Usuario Prueba

```
{
  "_id": "ObjectId("5a9b2e5770a4fe1452718a4b")",
  "rut_author": "183883062",
  "author": "Kuhlman, B., Dantas, G., Ireton, G.C.,
     Varani, G., Stoddard, B.L., Baker, D.",
  "pdb_link": "https://www.rcsb.org/structure/1QYS",
  "protein": "1QYS",
  "string_atom": {
    "archivo_pdb": "1QYS.pdb",
    "file_id": "ObjectId("5a9b340270a4fe16a2d82e5b")"
  },
  "design":[
  {
    "string_atom_design": {
          "string_atom_design": "Model 1.pdb",
          "file_id": "ObjectId("5a9b2e7a70a4fe1452718a50
            ")"
    },
    "archive_pdb": {
          "archive_pdb": "Model 1.pdb",
          "file_id": "ObjectId("5a9b2e7a70a4fe1452718a4e
            ")"
        },
        "topology": {
          "topology": "archivo.pdf",
          "file_id": "ObjectId("5a9b2e7b70a4fe1452718a52
            ")"
        },
    "category": "DE NOVO",
        "rut_author": 183883062,
        "its_work": "YES",
        "server": "I-TASSER",
        "parameters": {
          "ranking": "1",
          "parameters_default": "MODEL 1 - PROTEIN 1QYS"
        }
  }]
}
```

JSON 4.4: Documento Modelo Prueba

Ya creada la base de datos con datos de prueba, comienza la creación del sistema web, una vez instalado Flask se crea una carpeta en *Documentos* la cual se llama *Proyecto*.

Esta carpeta contiene un entorno virtual llamado *flask*. Este entorno virtual se crea para utilizar python, flask y pymongo (Ver Instalación de pymongo en la Sección B.7) de forma local, para el proyecto. Esto resulta bastante útil ya que no debe instalarse en todo el sistema y las modificaciones realizadas quedan dentro del proyecto, y no en todo el sistema operativo, permitiendo puedan utilizarse para otros propósitos.

Dentro de la carpeta *Proyecto*, se encuentra también un archivo llamado *run.py*, es el primero en ejecutarse y realiza la llamada a los demás archivos almacenados en una carpeta llamada *app*, ésta contiene el view, los templates y los formularios del sistema descritos a continuación.

4.2.1. View.py

El archivo *view.py* que se encuentra dentro de la carpeta app, es donde se realiza la conexión a la base de datos, mediante pymongo (detallado en la Sección B.7), además contiene la funcionalidad del sistema, es decir, se realizan los requerimientos mencionados en la Sección 3.1. También realiza la llamada a los templates mediante un decorador *@app.route('/')*, esto para vincular una función a un URL, permitiendo visualizar lo realizado.

4.2.2. Templates

En la carpeta templates se guardan todas las vistas, archivos *.html* creados como el Home, Index, Login, Ver Modelos, Ver Diseños, Ver Mis Modelos, Registro, Insertar Modelos e Insertar Diseños. El primer archivo creado fue el Index.html, el que contenía sólo un mensaje para mostrar en pantalla. Las vistas son lo primero a crear dentro de la página web, Flask busca plantillas en la carpeta templates para renderizarlas, esto

mediante el método *return render_template()*, al que se le proporciona el nombre de la plantilla y las variables que desee pasar al motor de la plantilla. Una vez creadas las vistas se da paso a la implementación de botones, accesos y permisos. Con la utilización de Boostrap4 se diseñó la página de forma más atractiva para el usuario.

4.2.3. Formularios

El archivo *forms.py* es el archivo de formularios del sistema, que se encuentra dentro de la carpeta app al igual que *view.py*. Este archivo utiliza un paquete de Flask llamado WTForms, que cumple con la función de facilitar la administración de formularios enviados al sistema por una vista de navegador, como son los casos de ingreso de datos como el login, registro, edición o el ingreso de modelos y diseños al sistema. La instalación de éste paquete es bastante simple, se realiza con el comando:

```
$pip install Flask-WTF
```

Usado este comando ya se encuentra habilitado el paquete, sólo queda crear un archivo llamado *forms.py* e importarlo dentro de éste. Dentro de *forms.py* se definen los formularios como clases y se pueden añadir validaciones, que devolverán True si los datos se validan o False si ocurre lo contrario. Realizado esto se pueden ingresar datos dentro de los formularios agregados, para llamarlos dentro de *views.py* se necesita utilizar los metodos *GET*, donde el navegador le dice al servidor que solo obtenga la información almacenada en esa página y la envíe, y *POST*, donde el navegador le dice al servidor que quiere publicar información nueva en esa URL y que el servidor debe asegurarse de que los datos se almacenen y sólo se almacenen una vez.

Ya contando con las carpetas, vistas e instalaciones mencionadas se ejecuta Flask, para esto se accede a la ubicación del proyecto con:

```
$cd Documentos/Proyecto
```

Y se utilizan los siguientes comandos:

```
$cd chmod a+ x run . py
$./run.py
```

El comando *chmod* es el encargado de gestionar los accesos y permisos de archivos, carpetas, particiones de disco o red. El *a* se refiere a todos los usuarios (all), el símbolo + otorga los permisos y el *x* da permiso de ejecución. Ambos comandos dejan corriendo el proyecto y se puede visualizar su ejecución en la dirección *http://127.0.0.1:5000/*. Sin embargo esta vista no tendrá funcionalidad alguna hasta que sea programada, es decir, los botones o ingresos no realizan ninguna función, solo se visualizan.

El código se desarrolla en Sublime Text (detallado en la Sección 4.1.8), esto facilita la creación y edición de código, la indentación, el poder acceder a varios archivos en el mismo lugar y colores para diferenciar rutas, variables, importaciones, entre otros.

4.3. Funcionalidad del sistema

Una vez creadas las vistas se comienza la funcionalidad del sistema, el login es el primero en la lista, éste requiere dos datos el rut y una contraseña. Para el ingreso de estos datos y del resto de formularios, se utiliza el *forms.py* mencionado en la Sección 4.2.3.

Entonces, el usuario intenta ingresar al sistema ingresando su rut y contraseña. El navegador toma los datos ingresados y se los envía al sistema, donde revisa si éste se encuentra registrado en la base de datos a través de una consulta realizada con Pymongo.

usuario = db.Usuarios.find_one("user":login_form.user.data)

En esta consulta *usuario* almacena el usuario encontrado o en caso de no encontrarlo guarda *None*. Si la respuesta es *None*, entonces devuelve un mensaje diciendo *Usuario no registrado*. En caso contrario el sistema compara el rut y contraseña ingresadas, con la contraseña del rut encontrado en la base de datos, de no coincidir arroja el mensaje *La contraseña es incorrecta!*. En caso de que la contraseña sea ingresada de forma correcta se envía un mensaje de *Ha ingresado con éxito!* y se utiliza la importación *session* para almacenar el logged_in con valor True, que servirá para saber que el usuario se encuentra activo, también para guardar el rol, rut y nombre del usuario ingresado, estos para ser utilizado en los permisos de usuario en el ingreso, edición y eliminación de modelos o diseños.

El logout es la continuación del login, si un usuario puede estar activo también puede cerrar su sesión. Para esto se utiliza el *session['logged_in']* almacenado en el login con valor True, si el botón de cerrar sesión es presionado, entonces *session['logged_in']* cambia su valor a False, es decir ya no se encuentra activo el usuario por ende finaliza su sesión.

La visualización de modelos también se realiza mediante una consulta a la base de datos, esta a diferencia del login es general, usando:

modelos = db.Models.find()

Esta consulta devuelve todos los modelos almacenados en el sistema. En caso de querer buscar algún modelo en específico, se cuenta con una barra de búsqueda en la que se puede ingresar el nombre de la proteína y visualizarla de forma individual. También se cuenta con un botón de *Buscar Todos* para volver a recargar todos los modelos almacenados. Esto se aplica también para visualizar los autores, solo que la consulta es:

usuarios = db.Usuarios.find()

Lo que devuelve todos los autores registrados, de la misma manera se puede buscar un autor en específico, donde se cuenta con una barra de búsqueda en la que se puede ingresar el rut del usuario y visualizarlo de forma individual.

Para ver los diseños debe estar visible la lista de modelos, ya que cada uno de éstos contará con un botón *Ver Diseños* que mostrará los diseños respectivos. Esto se hace mediante el traspaso del nombre de la proteína por url desde la vista *Ver Modelos* a *Ver Diseños*, así mostrará sólo los diseños del modelo seleccionado.

El ingreso de modelos o diseños al sistema, se hace utilizando formularios al igual que el login, donde el navegador toma los datos ingresados y se los envía al sistema, el sistema los almacena utilizando un *insert_one* y de forma automática agrega el rut del usuario que lo ingresa. Esto para tener un registro de qué autor ingresó cada modelo o diseño, ya que sólo él podrá editar o eliminar ésta información, a excepción del administrador. Para ambos ingresos, ya sea de modelo o diseño, se deben subir archivos, los que deben quedar almacenados en la base de datos. Para ello se utiliza GridFS, una especificación para almacenar y recuperar archivos. Cuando esto ocurre, la base de datos crea dos colecciones adicionales *fs.chunks* y *fs.files*, donde fs.chunks almacena los fragmentos binarios y fs.files almacena los metadatos del archivo [29].

El ingreso de autores a la base de datos, se realiza de la misma manera que de modelos o diseños, y sólo puede ser registrado por un administrador.

La opción de edición de modelos, diseños o autores se redirecciona al formulario de ingresar modelo, ingresar diseño o registrar respectivamente, pero esta vez con los campos visibles dentro de las casillas. El usuario ingresa los datos nuevos y guarda los cambios. El sistema utiliza un *update_one* para realizar esta edición, además de la eliminación de archivos subidos en la base de datos en el ingreso, dando paso a reingresar los *.pdb o .pdf*.

En la eliminación de un modelo, diseño o autor el sistema utiliza *delete_one*, no sólo

deben eliminarse los campos ingresados, sino los archivos subidos y almacenados. Para eliminar un diseño, se traspasa el ranking del diseño seleccionado por url, así el sistema sabe que modelo buscar en el sistema y cual de todos sus diseños eliminar. La eliminación se realiza consultando a la base de datos la *id* del archivo a eliminar.

Para la administración de los permisos de cada usuario, se utiliza la importación *session*. En el login se guardó el rol del usuario activo en *session['rol']*, esto es utilizado para ver si el usuario es autor o administrador, a través de condiciones *if* en los sectores correspondientes.

Finalizada la funcionalidad del sistema, éste se encuentra listo para implementarlo en el servidor, para ello se utiliza Dropbox donde se almacena el código programado y se traspasa. Se verifica que todo funcione de forma correcta, para dar paso a poblar la base de datos. Esto se realiza mediante los formularios de las páginas de ingreso, donde se cargan los archivos y datos obtenidos mencionados en la Sección 3.1.

Ya poblada nuestra base de datos, es posible dar inicio al siguiente capítulo, donde se realizan las pruebas correspondientes para confirmar el cumplimiento de los requerimientos propuestos en la Sección 3.1.

Capítulo 5

Pruebas

Una vez poblada la base de datos, se da paso a realizar las pruebas del software, con esto se verifica que cumple con los requerimientos solicitados en la Sección 3.1.

Las pruebas a realizar se dividen en 3 tipos de interfaces, una pública, una para un autor y otra para el administrador. Estas pruebas comienzan en la página de Inicio y buscan que la funcionalidad del sistema este realizada correctamente. Para esto el usuario debe poder descargar los archivos, ingresar información al sistema, editar, eliminar y registrar usuarios, todo esto con las restricciones de permisos que dependen del rol de cada uno de ellos.

5.1. Prueba 1: Interfaz pública

La interfaz pública consta de un menú superior que da acceso a diferentes páginas de la aplicación web, este menú contiene un inicio, ver modelos, ver diseños y un ingresar, además consta de opción de descarga para los archivos almacenados. Las pruebas realizadas para esta interfaz constan de verificar que los accesos funcionen de forma correcta, es de-

cir, que los botones de inicio, ver modelos y ver diseños, lleven al destino correspondiente y que las visualizaciones y descargas de archivos se realicen sin inconvenientes.

5.1.1. Inicio

La página de inicio es la presentación del sistema, da la bienvenida junto a una imagen en movimiento de la proteína *2KIQ*, además de dar opción de ingresar al sistema, visualizar modelos y ver sus respectivos diseños (Ver Figura 5.1).

Figura 5.1: Inicio

5.1.2. Ver Modelos

La página de Ver Modelos permite visualizar los modelos ingresados a la base de datos, dando la opción de descargar el archivo *.pdb* almacenado (Ver Figura 5.2). También contiene una barra de búsqueda donde se puede ingresar un nombre de proteína y arroja la indicada (Ver Figura 5.3).

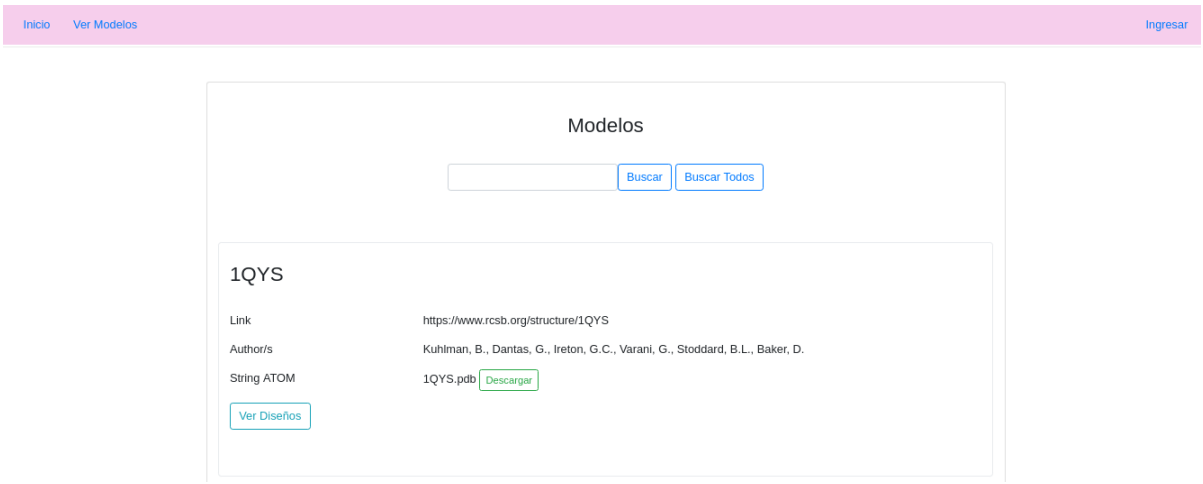

Figura 5.2: Ver Modelos

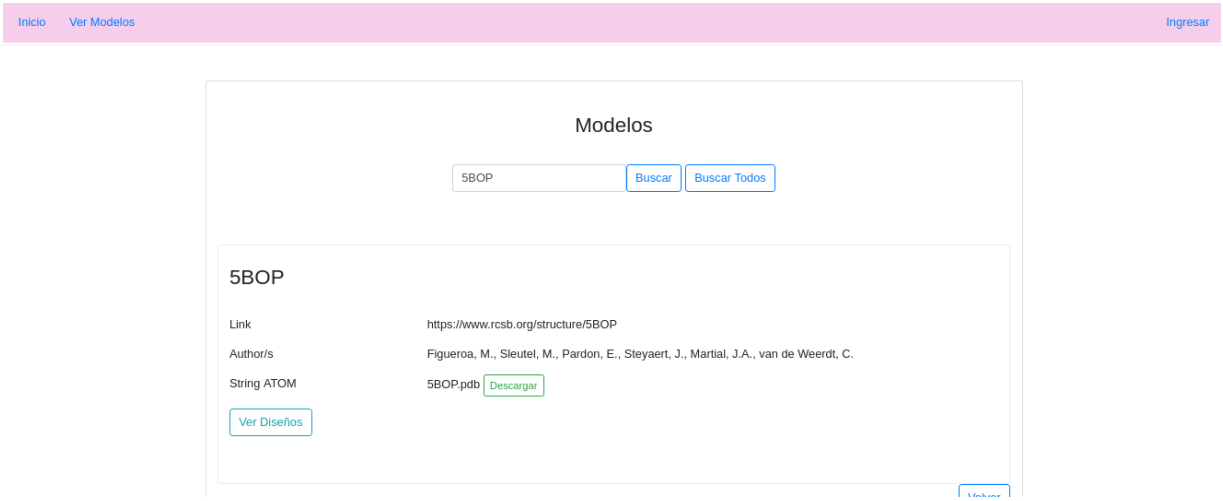

Figura 5.3: Búsqueda

Al presionar el botón de descarga se ve que enseguida aparece en la barra inferior de la pantalla, la descarga del archivo (Ver Figura 5.4). Esto funciona para todos los usuarios.

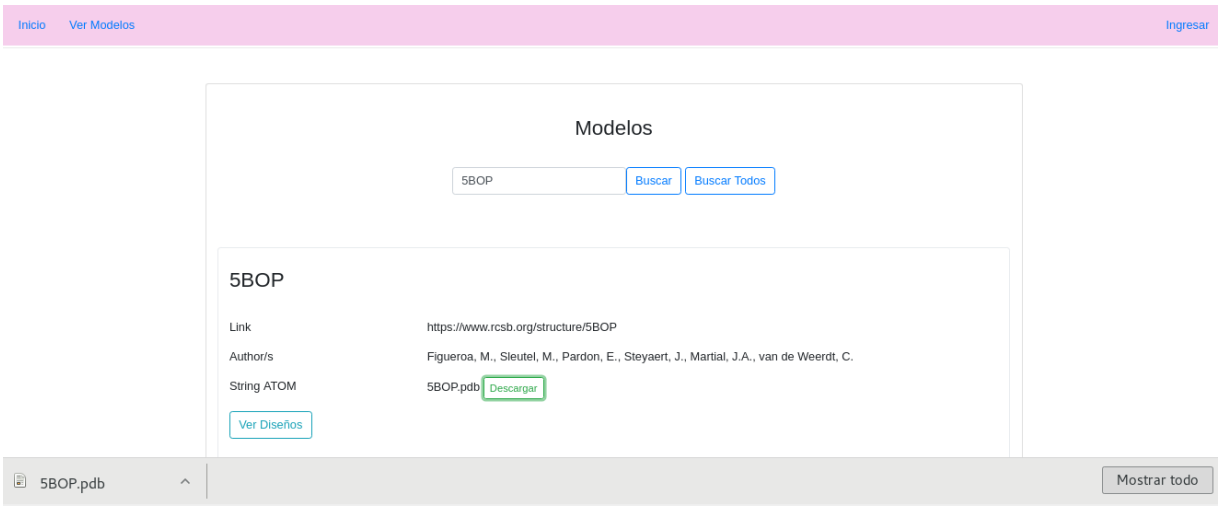

Figura 5.4: Descarga de archivo

5.1.3. Ver diseños

En la página de ver modelos se aprecia un botón más, llamado ver diseños, este botón permite ver los diseños de cada modelo, redireccionando a la página de *Ver Diseños* como se aprecia en la Figura 5.5. Esto es efectivo para todos los usuarios.

Figura 5.5: Ver diseños

En esta página también esta la opción de descargar los archivos, y como se ve en la Figura 5.6, descarga sin inconvenientes. Esto es efectivo para todos los usuarios.

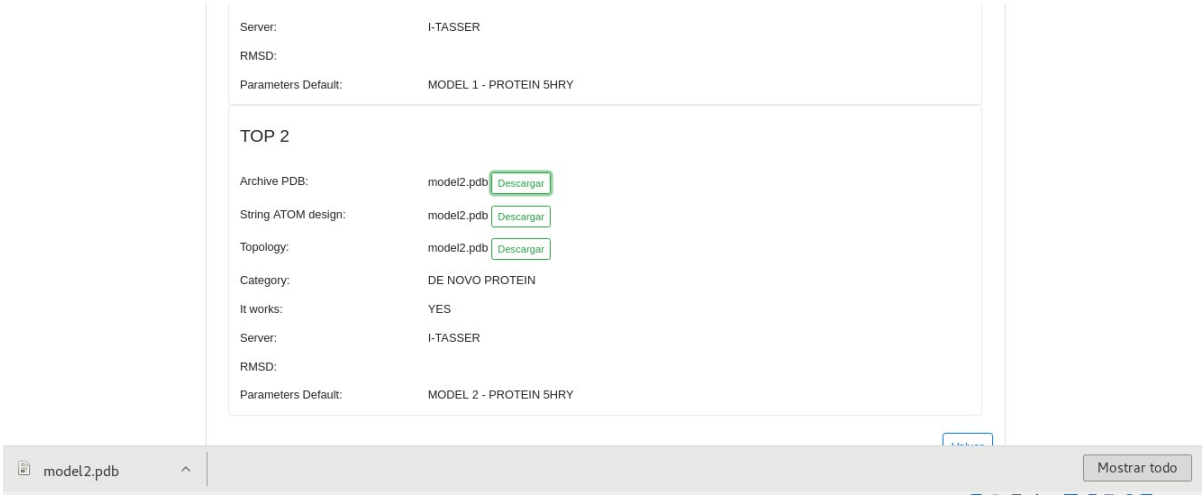

Figura 5.6: Descarga de archivo

5.1.4. Ingresar

La página de iniciar sesión contiene la autenticación de los Administradores y Autores, con un formulario para ingresar el rut y contraseña como se muestra en la Figura 5.7.

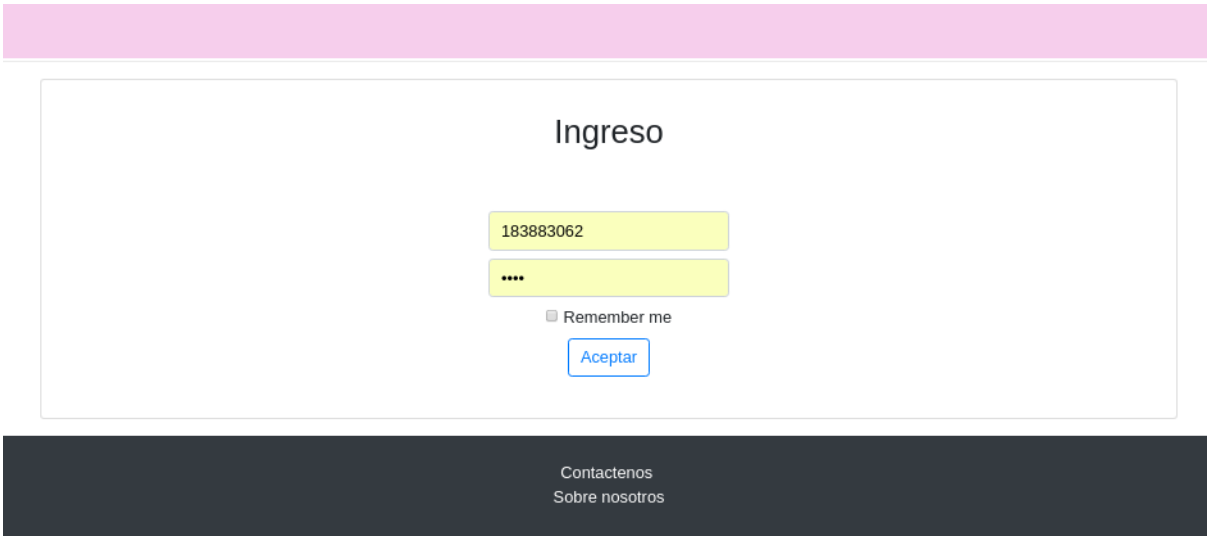

Figura 5.7: Ingresar al Sistema

Si es el administrador quien ingresa, y lo hace de forma correcta, cambia totalmente el menú de de la página de inicio, arroja un mensaje de ingreso exitoso y da más opciones de acceso como se aprecia en la Figura 5.8.

Figura 5.8: Ingreso Administrador

Si es el autor quien ingresa, y lo hace de forma correcta, también cambia totalmente el menú de la página de inicio, arroja un mensaje de ingreso exitoso y da más opciones de acceso, pero son menores a los del Administrador (Ver Figura 5.9).

Figura 5.9: Ingreso Autor

5.2. Prueba 2: Interfaz Autor

La interfaz de autor consta de un menú superior que da acceso a diferentes páginas de la aplicación web, este menú contiene al igual que la interfaz pública el ver modelos y el inicio con más accesos, pero contiene además la opción de *Ver Mis Modelos y Diseños*, *Ingresar Modelos y Diseños*, *Editar* cada uno de ellos, *Eliminar* cada uno de ellos y *Salir o Cerrar Sesión*. Las pruebas realizadas para esta interfaz constan de verificar que los accesos funcionen de forma correcta, es decir, que los botones de inicio, ver modelos y ver diseños, ver mis modelos y mis diseños, ingresar modelo y Salir lleven al destino correspondiente y que los ingresos, visualizaciones, ediciones y eliminaciones se realicen sin inconvenientes.

5.2.1. Inicio

El inicio es el que se muestra en imagen 5.9 donde se aprecia que se agregaron más opciones como *Ver Mis Modelos, Ingresar Modelos* y el *Salir*. La opción Ver Modelos se ve de la misma manera que un usuario público, donde puede acceder a descargar o ver los diseños de los modelos ingresados.

5.2.2. Ver Mis Modelos y Diseños

Cuando accede a Ver Mis Modelos, puede ver los modelos ingresados por el mismo y le da la opción de agregar más diseños al modelo, editar y eliminar como se ve en la Figura 5.10.

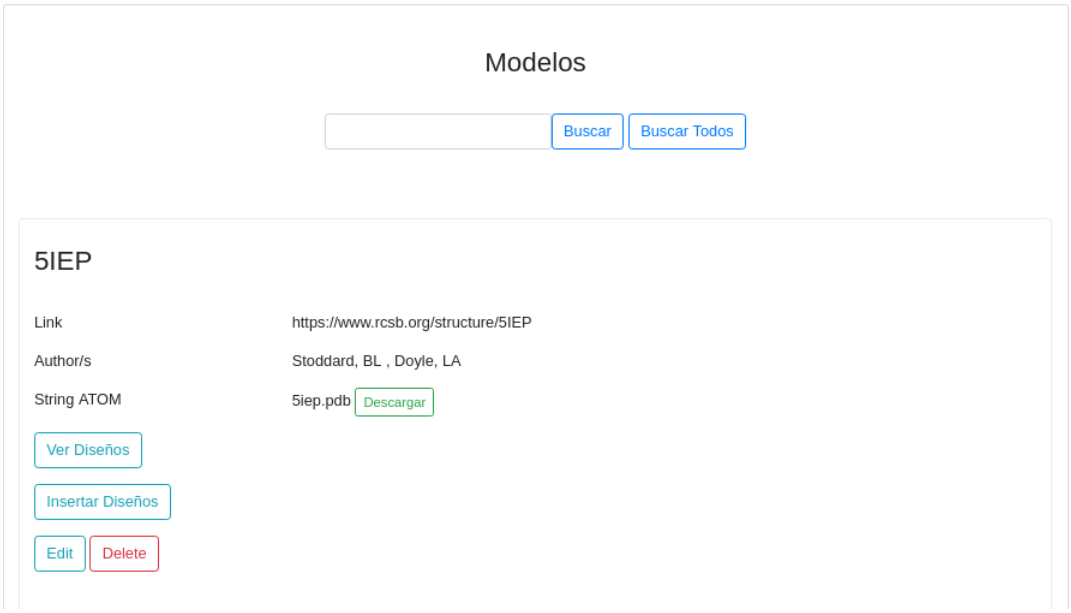

Figura 5.10: Ver Mis Modelos

Si se dirige a la opción de ver diseños del modelo propio, arrojará los diseños pero esta vez con la opción de editar y eliminar (Ver Figura 5.11).

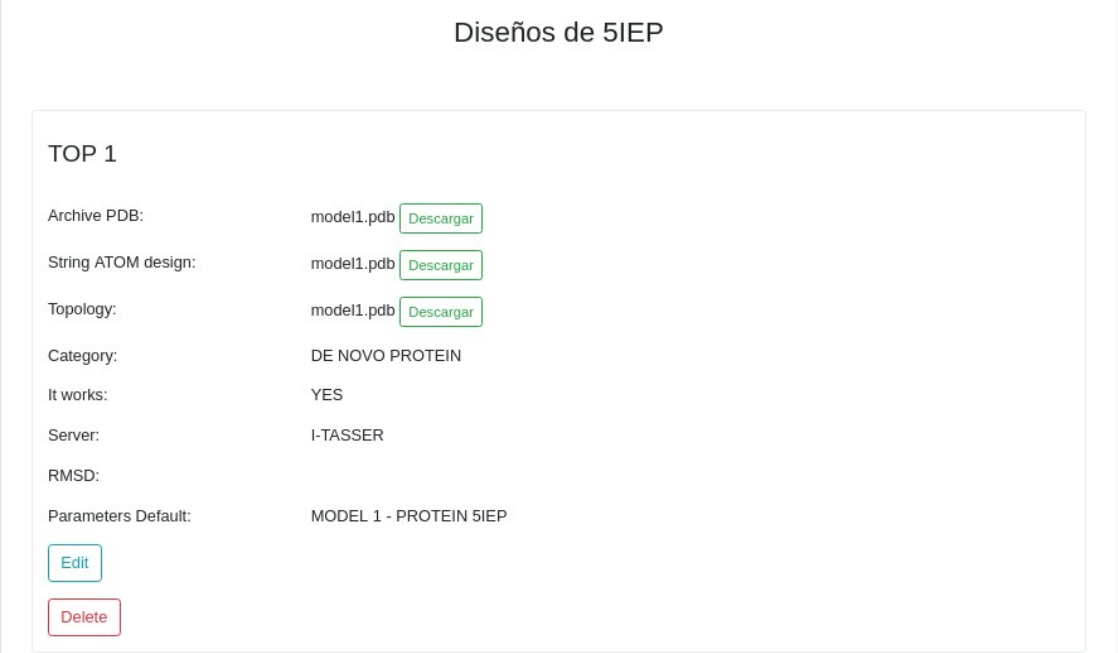

Figura 5.11: Ver Mis Diseños

Si se desea editar el diseño guardado, basta con presionar *Editar* y redirecciona al ingresar diseño pero con los datos anteriores visibles en los campos de ingreso (Ver Figura 5.12), esto no ocurre cuando se quiere ingresar un diseño nuevo.

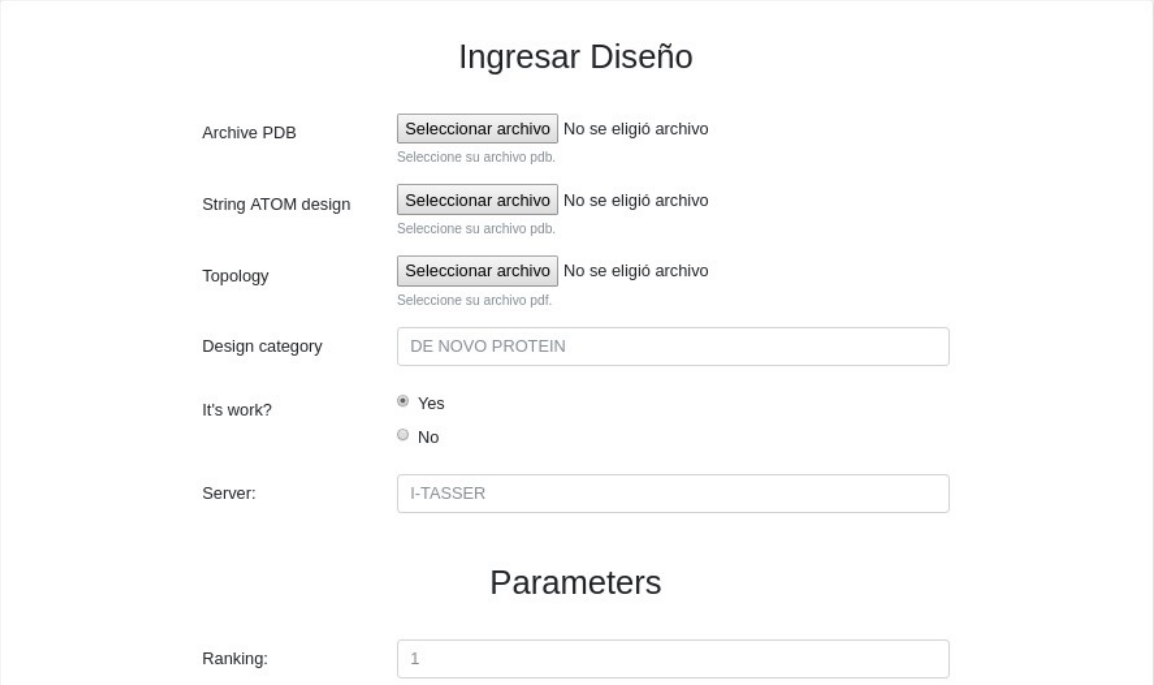

Figura 5.12: Editar Diseño

Al presionar *Eliminar* el diseño se remueve de forma exitosa y arroja un mensaje de éxito junto al diseño eliminado como se ve en la Figura 5.13. En este caso sólo tenía un diseño ingresado así que quedó vacío.

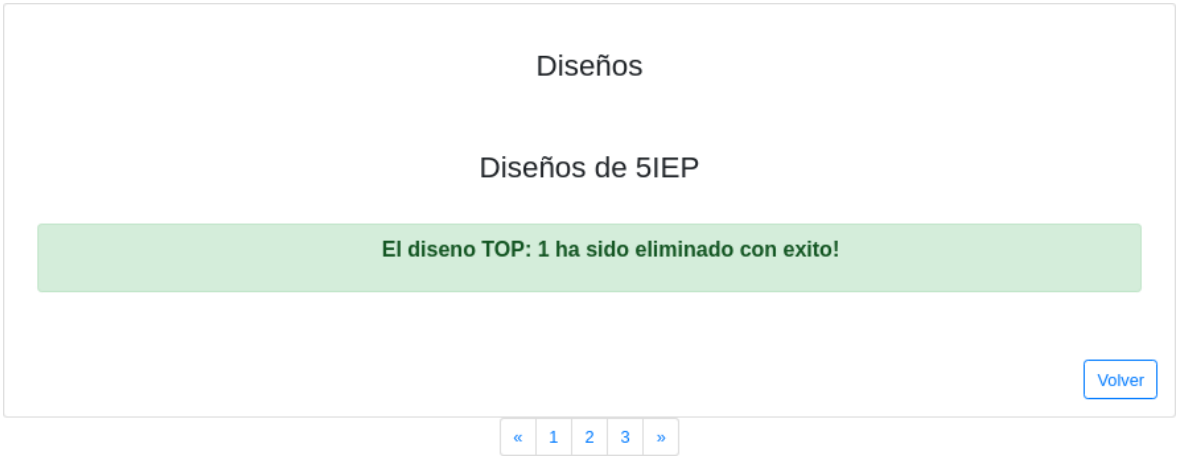

Figura 5.13: Eliminar Diseño

En la página de Ver Mis Modelos, se puede agregar un diseño nuevo al Modelo ingresado, para esto se accede mediante *Ingresar Diseño*. Como se muestra en la Figura 5.14, redirecciona al formulario de Ingresar Diseño con los campos vacíos. Aquí se llena el formulario, se presiona Guardar y quedará visible junto con los que se encuentren o en este caso quedará solo un diseño.

Figura 5.14: Ingresar Diseño

Se agrega un diseño de prueba para confirmar fue ingresado, este contiene todos los campos posibles con el nombre *PRUEBA*, es ingresado sin ningún problema (Ver Figura 5.15).

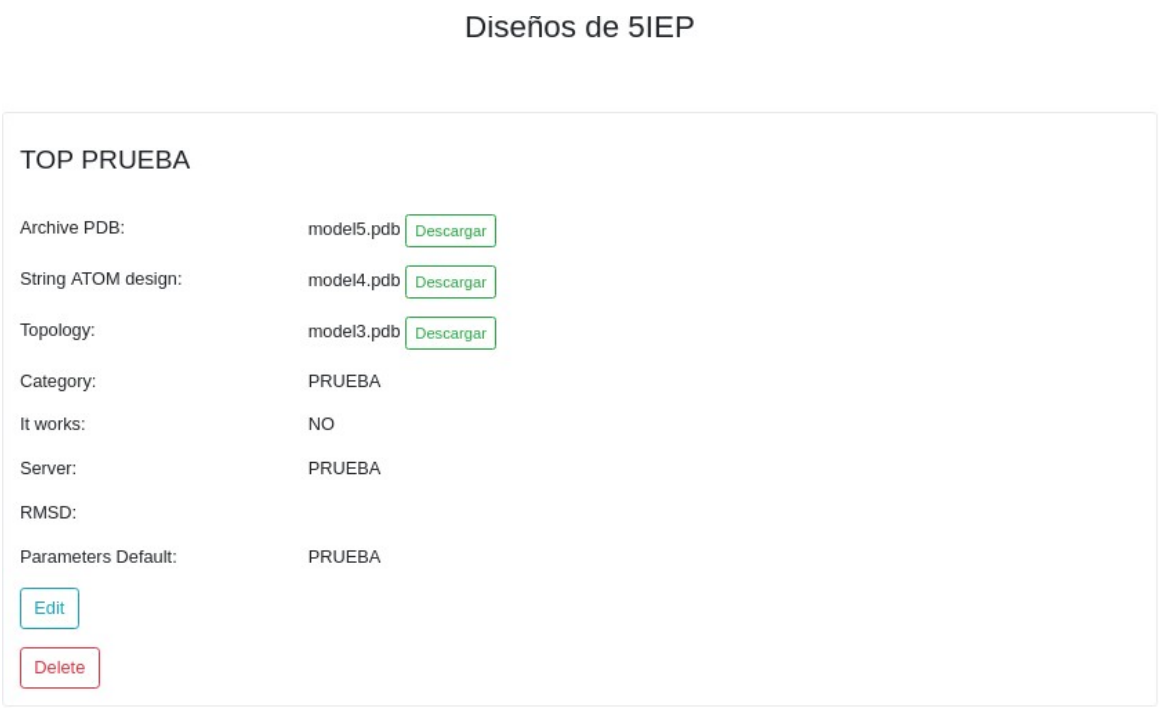

Figura 5.15: Ingresar Diseño Prueba

Para la edición del Modelo, funciona de la misma manera que de diseño, se redirecciona al ingreso de modelo pero con los datos anteriores visibles en los campos de ingreso como se muestra en la Figura 5.16. Esto no ocurre al ingresar un modelo nuevo.

Figura 5.16: Editar Modelo

Para validar la funcionalidad se dejan los mismos datos anteriores y solo se cambian: el archivo a subir y el nombre del ranking que será *PRUEBA*, como se ve en la Figura 5.17.

Figura 5.17: Editar Modelo Prueba

Se guardan los cambios y se dirige a Ver Mis Modelos, aparecerá el modelo editado

como se aprecia en la Figura 5.18.

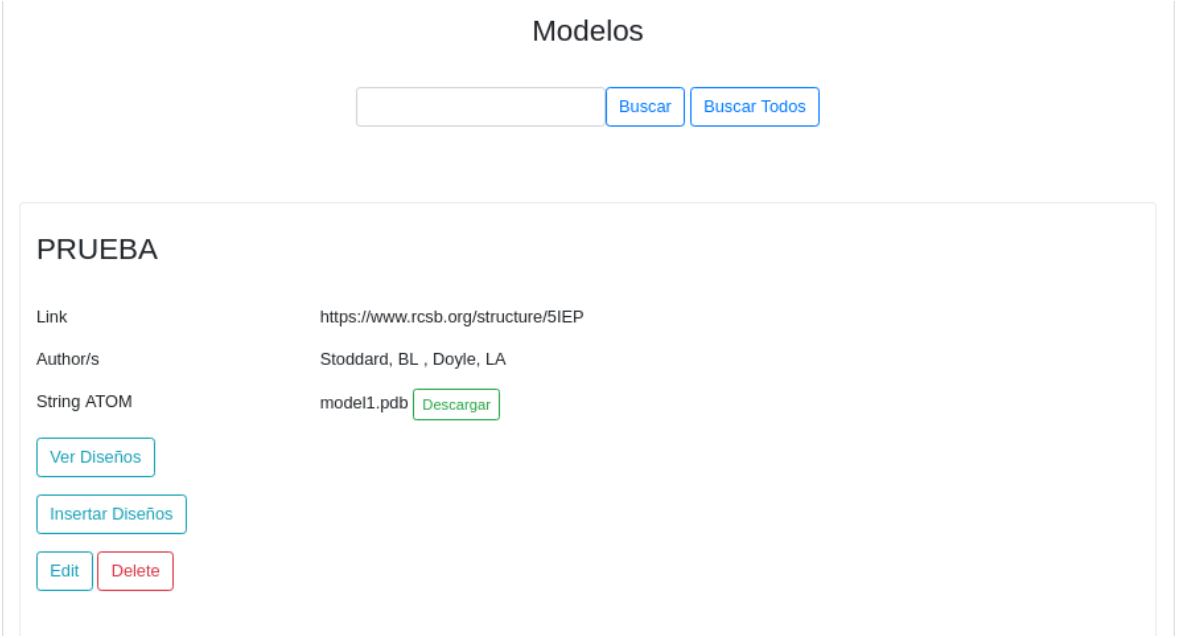

Figura 5.18: Modelo Prueba Editado

Se presiona *Eliminar* y se remueve con éxito el modelo PRUEBA editado anteriormente, envía mensaje exitoso y muestra todos los modelos que están ingresados (Ver Figura 5.19).

Figura 5.19: Modelo Prueba Eliminado

5.2.3. Ingresar Modelos y Diseños

El usuario autor también puede ingresar modelos y diseños al sistema, para esto puede ir a la pantalla de Inicio y seleccionar *Ingresar Modelo* esto lo redireccionará al formulario de ingreso que se muestra en la Figura 5.20, una vez ingresado los datos se presiona *Siguiente* y el modelo es guardado en la base de datos. La siguiente página que muestra es la de ingresar diseño (Ver Figura 5.14), facilitando el ingreso de diseños de forma continua.

Figura 5.20: Ingresar Modelo

5.2.4. Salir

El autor o administrador pueden cerrar su sesión cuando lo deseen, se presiona el botón *Salir*, que se encuentra al lado de su nombre de usuario (Ver Figura 5.8 y 5.9) y lo envía ala página de ingreso nuevamente como se mostró en la Figura 5.7.

5.3. Prueba 3: Interfaz Administrador

El administrador cuenta con todos los accesos mencionados anteriormente en ambas interfaces, la diferencia es que éste puede editar o eliminar cualquier modelo o diseño ingresado, no necesariamente los propios. Su pantalla de inicio es exactamente la misma que de autor, pero con dos accesos más disponibles que son *Ver Autores* y *Registrar Autor*. Las pruebas realizadas para esta interfaz constan de verificar que los accesos funcionen de forma correcta, es decir, que los botones de inicio, ver modelos y ver diseños, ver mis modelos y mis diseños, ingresar modelo, Ver Autores, Registrar Autor y Salir lleven al

destino correspondiente y que los registros, visualizaciones, ediciones y eliminaciones se realicen sin inconvenientes.

5.3.1. Ver Autores

El Ver Autores muestra a los usuarios registrados en la base de datos y da la opción de editar o eliminar cualquiera de ellos, como aparece en la Figura 5.21.

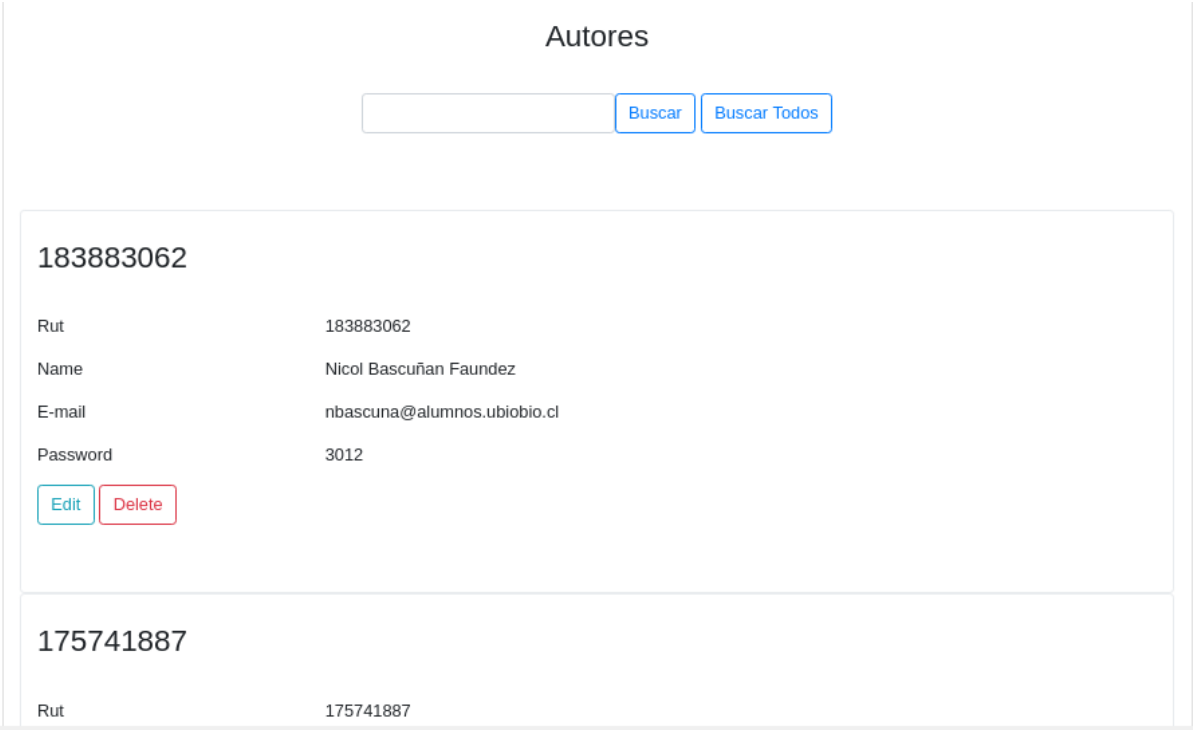

Figura 5.21: Ver Autores

Si se elije la opción editar, redirecciona a la página de *Registrar Autor* pero con los datos anteriores visibles en los campos de ingreso, como se muestra en la Figura 5.22. Se realizan los cambios, se presiona *Guardar* y el usuario es modificado.

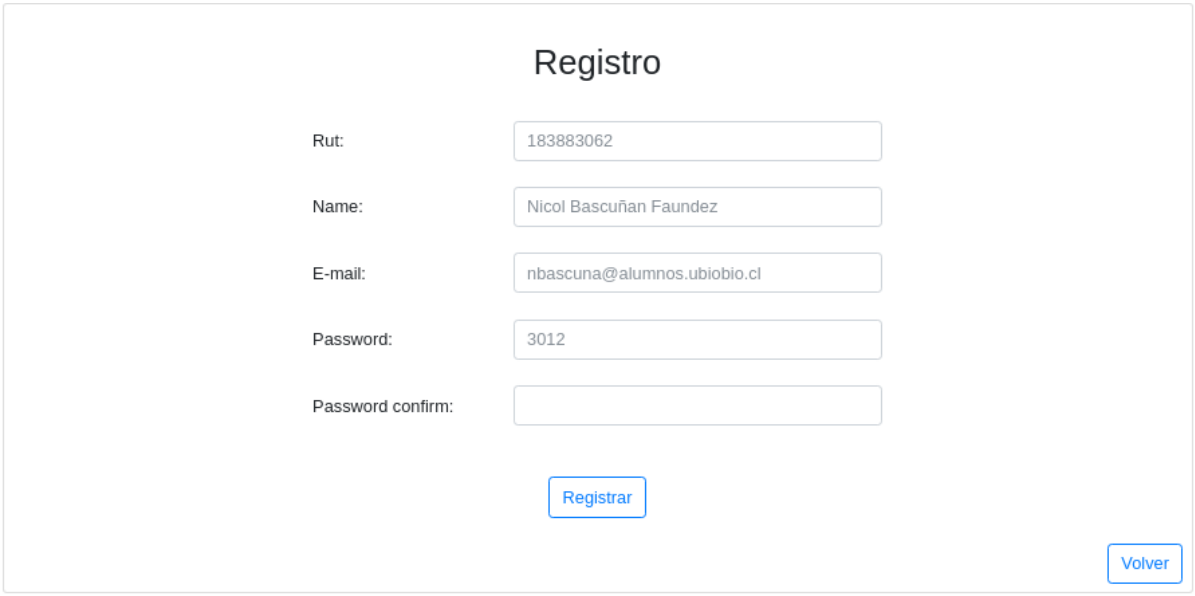

Figura 5.22: Editar Autor

Si se desea buscar un usuario en particular, hay una barra de búsqueda donde puede ingresar el rut y aparecerá sin problemas como se ve en la Figura 5.23.

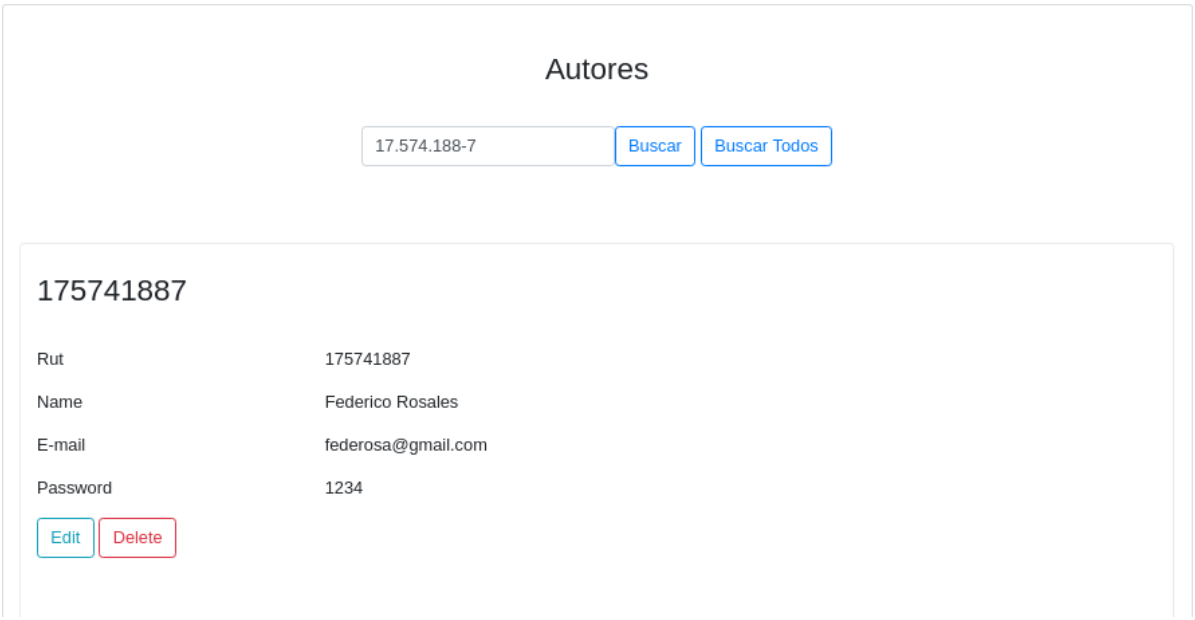

Figura 5.23: Buscar Autor

Para eliminar se presiona el botón *Eliminar* y se remueve con éxito el usuario que se desea (Ver Figura 5.24).

Figura 5.24: Eliminar Autor

5.3.2. Registrar Autor

El administrador puede registrar un autor o varios, en la Figura 5.25 se muestra el formulario de registro llenado por éste con los datos del usuario a registrar. Este formulario se presenta con los campos vacíos a diferencia de una edición.

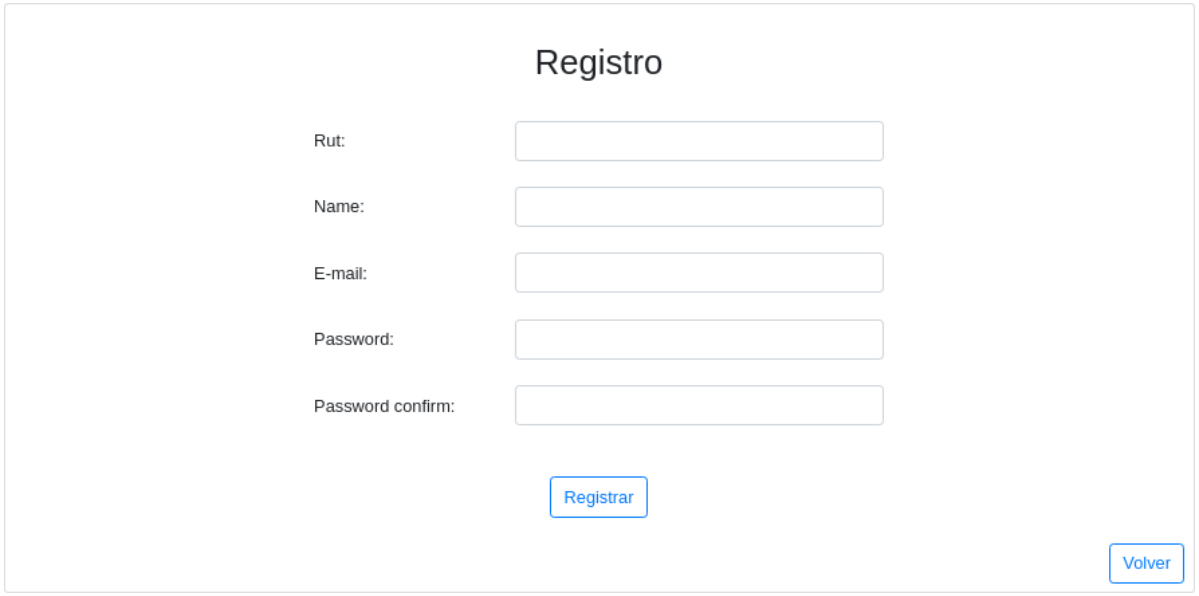

Figura 5.25: Registrar Autor

Al terminar el sistema se realizaron inserciones directas en la base de datos, y se realizaron las consultas correspondientes. Se verificó que los modelos, diseños y autores ingresados se realizaron de forma correcta, almacenando los atributos y archivos sin problemas. Que las eliminaciones borraban los archivos y atributos correctos de modelos, diseños y autores. Y que las ediciones reemplazaban el modelo, diseño y autor correspondiente, incluyendo los archivos de éstos. Al tratarse de una página web las pruebas se realizaron de acuerdo a la navegación. Se verificó que los permisos de cada usuario estaban correctamente.

Capítulo 6

Conclusión y Trabajos Futuros

Los intentos de predicción de estructuras, como vimos, son de vital importancia en una gran cantidad de procesos biológicos. Estos intentos de predicciones, almacenados en nuestra base de datos, podrían ayudar a reducir el tiempo, costos y recursos humanos significativamente al momento de determinar estructuras tridimensionales de proteínas futuras. Cuando una proteína es almacenada en PDB, puede utilizarse para realizar comparaciones con el más asertivo de los diversos diseños de la proteína en estudio, almacenados en nuestra base de datos. La comparación de sus parametrizaciones, serviría para realizar un análisis para aprender de ellas, lo que sería de gran ayuda para futuros diseños de otras proteínas, también para la predicción de estructuras y la mejora de diferentes softwares de modelaje y análisis computacional de estructuras.

En esta investigación se trabajó con diversas herramientas que ayudaron al desarrollo del sistema, entre ellas y la más importante, el sistema gestor de base de datos MongoDB, que permitió la creación de la base de datos y el almacenamiento de los diseños de las proteínas de una forma no estructurada. Este gestor permite flexibilidad, que era necesaria

para los tipos de datos que se necesitaban almacenar y para el crecimiento, dado que ahora la cantidad de diseños almacenados es baja y se pretende llegar a una cantidad aproximada de 1.755.000 datos en un futuro, esto calculado de los diseños almacenados, que son 127 diseños en un total de 10 proteínas diferentes. Si multiplicamos la cantidad aproximada de diseños por proteína que esta almacenada por la cantidad de estructuras proteicas que contiene el PDB nos queda [13 (diseños) * 135.000 (modelo)] = 1.775.000 una cantidad que sobrepasa lo almacenado por PDB. MongoDB permite también realizar modificaciones futuras, dado que no genera complicaciones modificar el código programado para agregar datos o quitarlos siempre que se estime necesario. Otra herramienta utilizada fue Flask, que permitió la simple creación del sistema web, al ser flexible permite que se puedan seguir añadiendo extensiones sin tener que modificar lo programado. Flask se adapta a lo requerido y se puede seguir potenciando el sistema a futuro.

Como trabajos futuros se puede considerar poblar la base de datos con una gran cantidad de información, esto para analizar el funcionamiento del sistema y quizás realizar modificaciones de ser necesario para incrementar su población. También se pueden agregar datos que sean de importancia para hacerla más completa. El sistema web se puede mejorar de acuerdo a las necesidades de los investigadores, ésto puede significar realizar cálculos o ingresar otro tipo de archivos. Esto incentiva a la investigación dado que contendría más herramientas e información para realizar futuros diseños o predicciones.

Referencias

[1] Araya, E. P. Hemoglobina. http://slideplayer.es/slide/9049408/, 2016. [Online; accessed 17-Enero-2018].

[2] Berman, H., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T., Weissig, H., Shindyalov, I., and Bourne, P. The protein data bank nucleic acids research. *View Article PubMed/NCBI Google Scholar 28* (2000), 235–242.

[3] Biologia. Estructura Secundaria. https://biologia.laguia2000.com/bioquimica/prionlas-proteinas-que-infectan, 2018. [Online; accessed 17-Enero-2018].

[4] Biology, N. S. Nature Structural Biology 10 , 980 (2003) doi: 10.1038 7 nsb1203-980. http://www.wwpdb.org/documentation/file-format, 2003. [Online; accessed 17-Enero-2018].

[5] Blanca, O. Escalabilidad Horizontal. https://www.oscarblancarteblog.com/2017/03/07/escalabilidad-horizontal-y-vertical/, 2018. [Online; accessed 24-Febrero-2018].

[6] Cattell, R. Scalable sql and nosql data stores. *Acm Sigmod Record 39*, 4 (2011), 12–27.

[7] codigo y algo mas, D. Tutorial Flask. https://decodigoyalgomas.com/tutorial-flask-basico/, 2018. [Online; accessed 25-Febrero-2018].

[8] Coltell, Ó. La disciplina de bioinformática: definición y caracterización. *Informática y Salud*, 43 (2003), 29–34.

[9] Corporation, O. VirtualBox2018. https://www.virtualbox.org/, 2018. [Online; accessed 10-Enero-2018].

[10] Damodaran, S. *Amino acids, peptides, and proteins*, vol. 4. CRC Press: Boca Raton, FL, 2008.

[11] de Lourdes Cornejo Arteaga, P. M. Los ácidos carboxílicos. https://www.uaeh.edu.mx/scige/boletin/prepa3/n8/m9.html, 2018. [Online; accessed 23-Febrero-2018].

[12] Definiciones. Puente de Hidrogeno. https://definicion.de/puente-de-hidrogeno/, 2018. [Online; accessed 23-Febrero-2018].

[13] DeQuimica. Glosario de Quimica. http://dequimica.com/glosario, 2018. [Online; accessed 23-Febrero-2018].

[14] Diccionario. Inmunoprecipitación. http://diccionario.raing.es/es/lema/inmunoprecipitaci%C3%B3n, 2018. [Online; accessed 23-Febrero-2018].

[15] Dropbox. Dropbox. https://www.dropbox.com, 2018. [Online; accessed 25-Febrero-2018].

[16] ehueus. Difraccion de rayos X. http://www.ehu.eus/imacris/PIE06/web/DRXP.htm, 2018. [Online; accessed 23-Febrero-2018].

[17] Figueroa, M., Oliveira, N., Lejeune, A., Kaufmann, K. W., Dorr, B. M., Matagne, A., Martial, J. A., Meiler, J., and Van de Weerdt, C. Octarellin vi: Using rosetta to design a putative artificial (β/α) 8 protein. *PloSone8*, 8 (2013), e71858.

[18] Franco, M. L., Cediel, J. F., and Payán, C. Brief history of bioinformatics. *Colombia Médica 39*, 1 (2008), 117–120.

[19] GNOME. GNOME. https://www.gnome.org/, 2018. [Online; accessed 15-Enero-2018].

[20] González-Torres, L., Téllez-Valencia, A., Sampedro, J., and Nájera, H. Las proteínas en la nutrición. *Rev Salud Pública y Nutrición 8*, 2 (2007).

[21] Graterol, Y. *Mongo DB en español Tomo 1:El Principio*. Yohan Graterol (26 May 2014), 2014.

[22] guia, L. Quimica. https://quimica.laguia2000.com/conceptos-basicos/enlace-disulfuro, 2018. [Online; accessed 23-Febrero-2018].

[23] HM Berman, J. Westbrook, Z. F. G. G. T. B. H. W. I. S. P. B. The Protein Data Bank Nucleic Acids Research , 28: 235-242. https://www.rcsb.org/, 2000. [Online; accessed 17-Enero-2018].

[24] Hochreiter, S. Bioinformatics iii.

[25] Infobiologia. Biologia. http://www.infobiologia.net/2015/07/, 2018. [Online; accessed 23-Febrero-2018].

[26] Keskin, O., Gursoy, A., Ma, B., and Nussinov, R. Principles of protein-protein interactions: What are the preferred ways for proteins to interact? *Chemical reviews 108*, 4 (2008), 1225–1244.

[27] Linux. Ubuntu. https://www.ubuntu.com/, 2018. [Online; accessed 14-Enero-2018].

[28] Mezo, O. Globulares y Fibrosas. http://slideplayer.es/slide/3450021/, 2016. [Online; accessed 17-Enero-2018].

[29] MongoDB. MongoDB. https://www.mongodb.com/, 2018. [Online; accessed 15-Enero-2018].

[30] Monografias. Estructura Primaria. http://www.monografias.com/trabajos-pdf5/rey-eternidad-g080/rey-eternidad-g0804.shtml, 2018. [Online; accessed 17-Enero-2018].

[31] Phizicky, E. M., and Fields, S. Protein-protein interactions: methods for detection and analysis. *Microbiological reviews 59*, 1 (1995), 94–123.

[32] Programacion, W. Concurrencia. https://webprogramacion.com/43/sistemas-operativos/concurrencia-de-procesos.aspx, 2018. [Online; accessed 24-Febrero-2018].

[33] Python. Python. https://www.python.org/, 2018. [Online; accessed 25-Febrero-2018].

[34] R Shenoy, S., and Jayaram, B. Proteins: sequence to structure and function-current status. *Current Protein and Peptide Science 11*, 7 (2010), 498–514.

[35] Raisman, J. S. Cuatro Niveles de una Proteina. http://www.biologia.edu.ar/macromoleculas/structup.htm, 2018. [Online; accessed 17-Enero-2018].

[36] Robomongo. Robomongo. https://robomongo.org/, 2018. [Online; accessed 25-Febrero-2018].

[37] Ronacher, A. Flask. http://flask.pocoo.org/, 2018. [Online; accessed 25-Febrero-2018].

[38] SublimeText. SublimeText. https://www.sublimetext.com/, 2018. [Online; accessed 25-Febrero-2018].

[39] Umbrello. Umbrello. https://docs.kde.org/trunk4/es/kdesdk/umbrello/introduction.html, 2018. [Online; accessed 25-Febrero-2018].

[40] Rosetta. Rosetta. https://www.rosettacommons.org/software, 2018. [Online; accessed 07-Marzo-2018].

Apéndice A

Glosario, Siglas y Abreviaciones

A.1. Glosario

Definición A.1 ***Enlace peptídico*** *Es un enlace entre el grupo amino y el grupo carboxilo pertenecientes diferentes aminoácidos [25].* Q

Definición A.2 ***Grupo carboxilo*** *Grupo funcional orgánico que consiste en un átomo de carbono unido de forma doble a un átomo de oxígeno y unido por unión simple a un grupo hidroxilo [11].* Q

Definición A.3 ***Grupo amino*** *Grupo funcional derivado del amoniaco (NH3) por pérdida de un hidrógeno [13].* Q

Definición A.4 ***Puente de Hidrógeno*** *Atracción existente en un átomo de hidrógeno y un átomo de oxígeno, flúor o nitrógeno con carga negativa [12].* Q

Definición A.5 ***Puentes disulfuro*** *Es un enlace covalente fuerte entre grupos de azufre [22].* Q

Definición A.6 ***Difracción de rayos X*** *Determina la estructura detallada de un material, como la posición que ocupan los átomos, iones o moléculas que lo forman [16].* Q

Definición A.7 ***Inmunoprecipitación*** *Técnica en la cuál un antígeno proteico es precipitado de una solución utilizando un anticuerpo específico de esa proteína [14].* Q

Definición A.8 ***Escalar Horizontalmente*** *Consiste en potenciar el rendimiento del sistema desde un aspecto de mejora global [5].* Q

Definición A.9 ***Concurrencia*** *Propiedad de los sistemas donde los procesos se hacen simultáneamente o procesados al mismo tiempo [32].* Q

Definición A.10 ***Cristalografía*** *Ciencia que estudia los cristales, sus formas y sus propiedades. Establece la relación entre la forma geométrica y la estructura interna de los átomos o moléculas constituyentes de la sustancia [13].* Q

Definición A.11 ***Werkzeug*** *Es una libreria WSGI que maneja básicamente los requests entre el cliente y el servidor [7].* Q

Definición A.12 ***Jinja2*** *Es un lenguaje de templates para python inspirado en el utilizado por Django [7].* Q

A.2. Siglas y Abreviaciones

- ***RAM:*** Random Access Memory (Memoria de Acceso Aleatorio).

- ***UML:*** Unified Modeling Language(Lenguaje de modelado Unificado).
- ***IEEE:*** Institute of Electrical and Electronics Engineers(Instituto de Ingenieros Eléctricos y Electrónicos).
- ***APA:*** American Psychological Association(Asociacion Americana de Psicologia).
- ***DPE:*** Diseño de Proteínas Experimentales.
- ***GPL:*** General Public License.
- ***GUI:*** Graphical User Interface.
- ***LVM:*** Logical Volume Manager (Administrador de Volúmenes Lógicos).
- ***APT:*** Advanced Packaging Tool (Herramienta Avanzada de Empaquetado).
- ***KDE:*** K Desktop Environment (Entorno de escritorio).
- ***XML:*** Extensible Markup Language (Lenguaje de Marcado Extensible, formato universal).
- ***GTK:*** The GIMP Toolkit (Kit de herramientas GIMP).
- ***GRUB:*** GRand Unified Bootloader (Gestor de arranque múltiple).

- ***BSD:*** Berkeley Software Distribution.

- ***WSGI:*** Web Server Gateway Interface (Interfaz de gateway de servidor web).

Apéndice B

Instalación de componentes

B.1. Instalación y configuración de VirtualBox

Para la instalación de VirtualBox se dirige a la página oficial, donde podrá realizar la descarga del software como aparece en la Figura B.1. Una vez descargado, es ejecutado y basta con seguir los pasos de instalación que son relativamente sencillos.

Figura B.1: Página de descarga VirtualBox [9].

Una vez instalado de forma correcta, aparecerá la pantalla de inicio "¡Bienvenido a VirtualBox!" que se muestra en la Figura B.2. Aquí procedemos a presionar el botón que

aparece en la esquina superior izquierda de la pantalla *Nueva*, con el que daremos paso a la creación de una nueva máquina virtual.

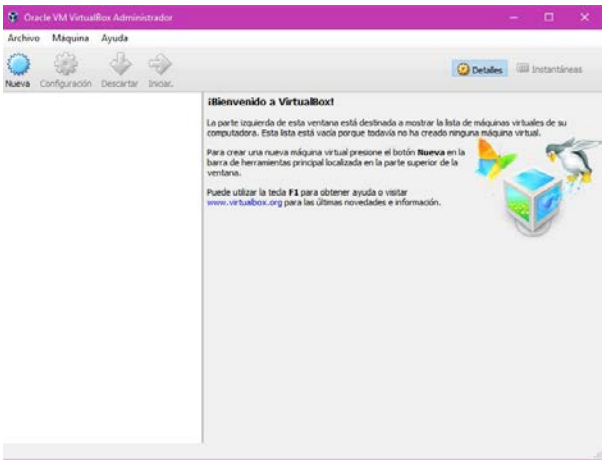

Figura B.2: ¡Bienvenido a VirtualBox!.

Le asignaremos un nombre a elección a la máquina, el tipo de sistema operativo que le instalaremos y la versión (ver Figura B.3).

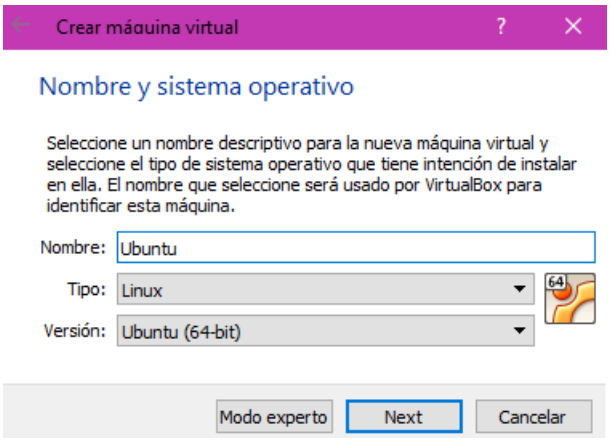

Figura B.3: Creando máquina virtual.

Una vez que son llenados los campos se presiona *Next*, que dará paso a seleccionar

cuánta memoria RAM queremos asignarle a nuestra máquina virtual, ver Figura B.4. Lo importante es no dejar sin memoria RAM al sistema *anfitrión*, si esto ocurriese no funcionaría ni el sistema real ni el sistema invitado, esto debido a que a pesar de ser una memoria RAM virtual, se asocia de la real.

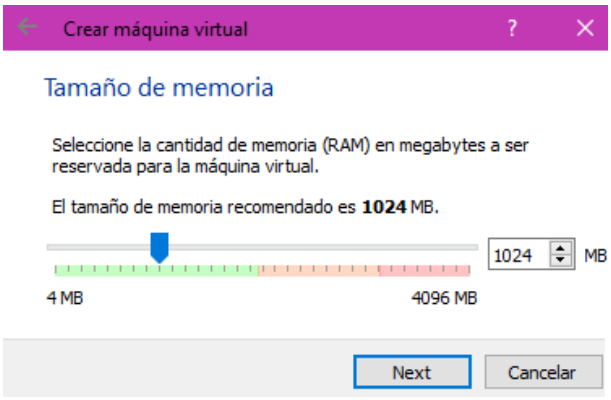

Figura B.4: Seleccionar la memoria RAM.

Ya seleccionada la memoria RAM, se presiona el botón *Next*, éste llevará a la configuración del disco duro, ya sea para agregar uno, crear un archivo de disco duro o seleccionar uno de la lista. Al igual que con la memoria RAM es importante no dejar sin disco duro al sistema *anfitrión* (ver Figura B.5).

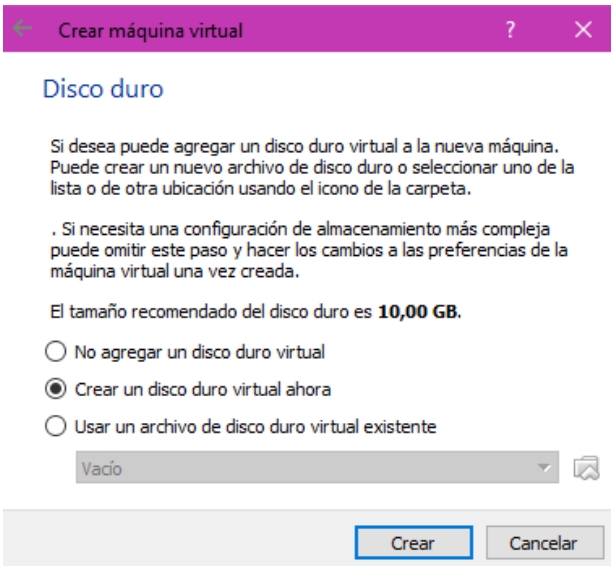

Figura B.5: Disco duro.

En el caso de esta máquina virtual utilizaremos la opción de *Crear un disco duro virtual ahora* ya que no se cuenta con uno existente y necesitamos guardar archivos. Esto guiará a seleccionar que tipo de archivo de disco duro queremos crear, como se muestra en la Figura B.6. VDI es el formato nativo de VirtualBox por ende será utilizado.

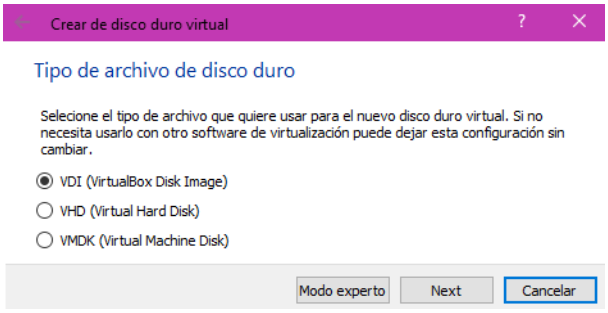

Figura B.6: Tipo de archivo de disco duro.

El siguiente paso es seleccionar el tipo de almacenamiento del disco duro virtual, como

explica la Figura B.7 existe la opción de un *Tamaño fijo* o de un *Reservado dinámica-mente*, que según el tamaño que vaya necesitando el sistema operativo lo va tomando del disco duro real.

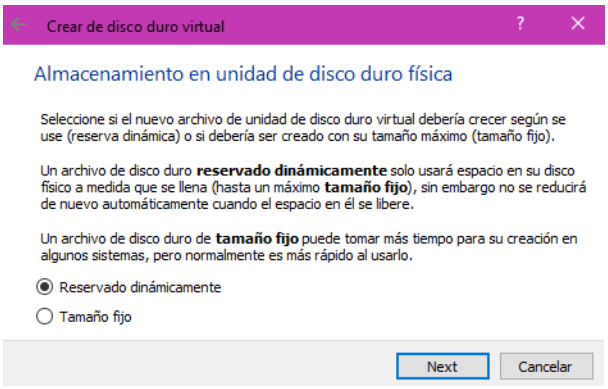

Figura B.7: Almacenamiento en unidad de disco duro física.

Como se ha seleccionado el *Reservado dinámica-mente*, se debe indicar un tope máxi-mo, con esto se controlará el crecimiento del disco duro virtual, así no dejará al anfitrión sin espacio. También debemos asignar la ubicación del archivo y le damos a crear (ver Figura B.8).

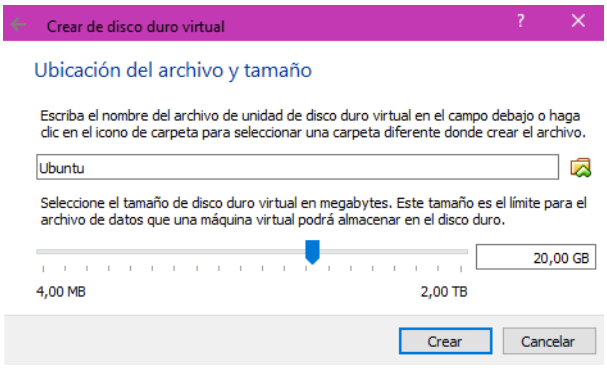

Figura B.8: Ubicación del archivo y tamaño.

Una vez creado el disco duro virtual, arrojará un resumen general de la máquina virtual creada donde se puede ver el nombre asignado, el sistema operativo que utilizaremos, la memoria base, entre otros. Con esto se finaliza la configuración previa de la máquina virtual, por lo que se encuentra lista para la instalación del sistema operativo que se desee, en este caso se utilizará Ubuntu 16.04.3 (ver Figura B.9)

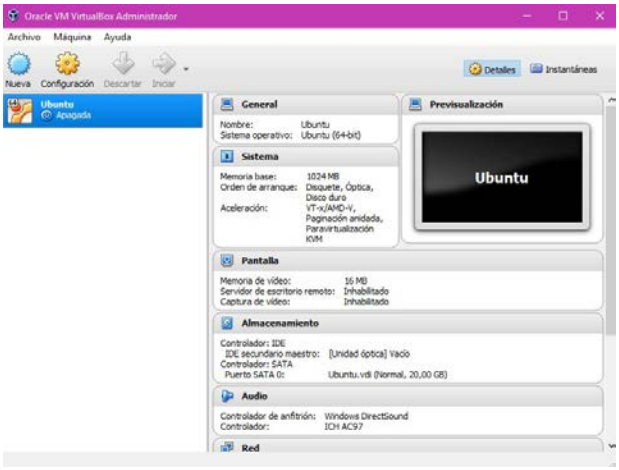

Figura B.9: Máquina virtual creada.

B.2. Instalación de Ubuntu

Para la instalación de Ubuntu se dirige a la página oficial, donde podrá realizar la descarga del sistema operativo de forma gratuita como aparece en la Figura B.10.

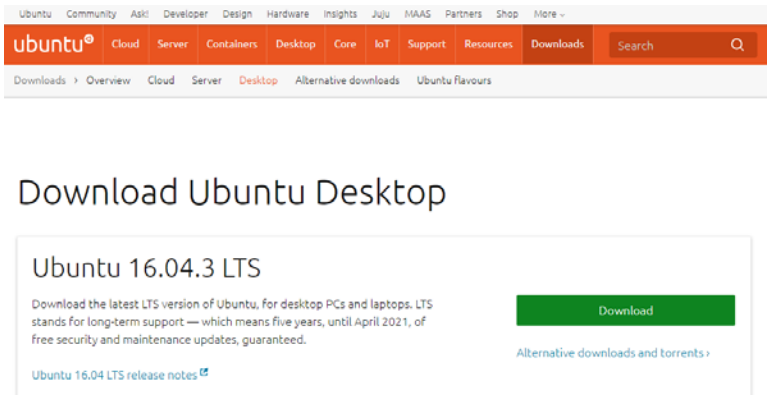

Figura B.10: Página de descarga Ubuntu [27].

Una vez descargado, se procede a realizar la instalación dentro del computador personal, servidor o de la máquina virtual, en este caso se empleará en un servidor y en una máquina virtual, ésto para poder realizar pruebas desde una máquina sin afectar al software principal del servidor.

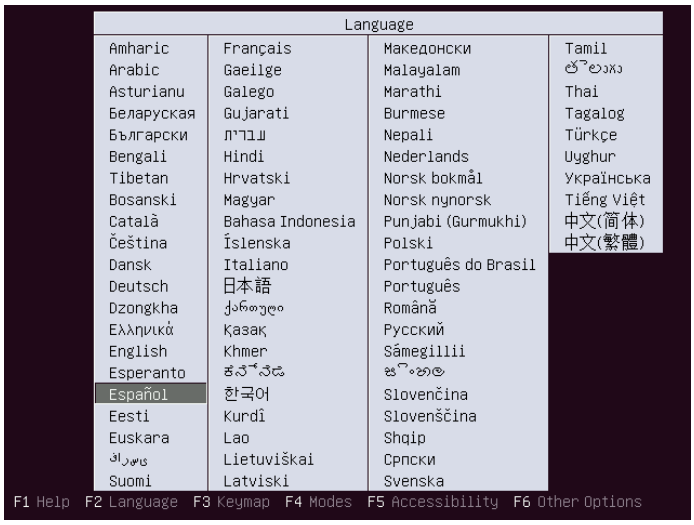

Figura B.11: Selección del Lenguaje.

Al iniciar la instalación se mostrará una ventana que solicitará seleccione el lenguaje en que desea instalar el sistema operativo, ver Figura B.11. Luego de seleccionarlo la pantalla de inicio de Ubuntu se hará visible para que escoja como desea la instalación o si busca hacer alguna reparación del sistema. Para este sistema se utilizó *Instalar Ubuntu Server* (ver Figura B.12).

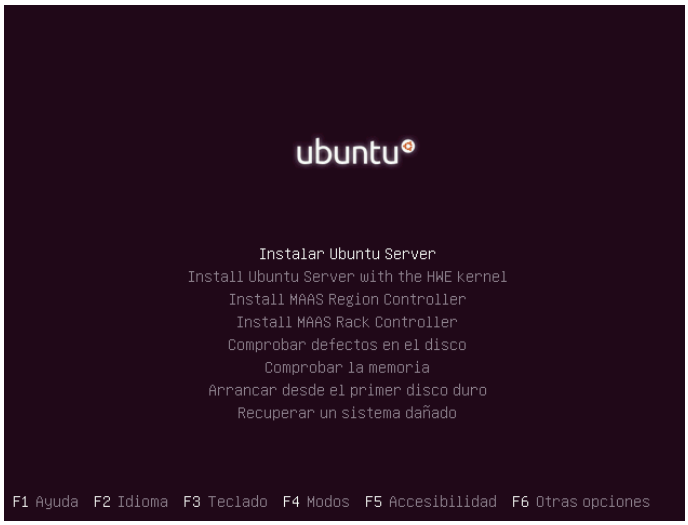

Figura B.12: Inicio de Ubuntu para instalación.

Luego se debe seleccionar la ubicación para fijar la zona horaria, ésta lista se reduce a lugares donde se emplea el idioma que se escogió anteriormente. (ver Figura B.13).

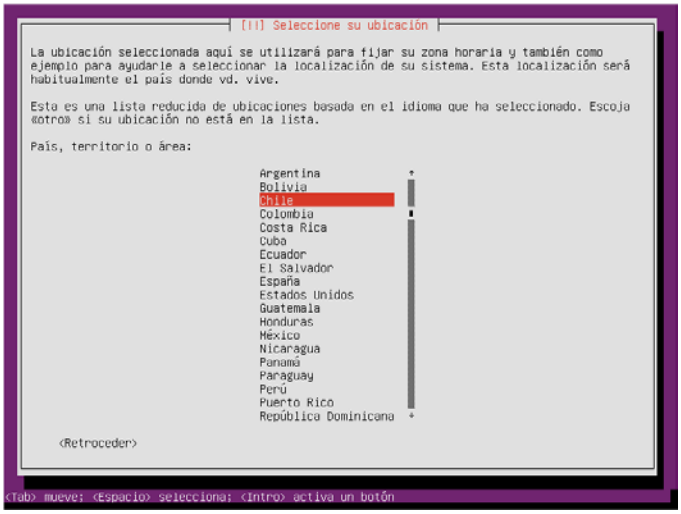

Figura B.13: Selección del Ubicación.

Como las distribuciones de teclado varían por país se debe seleccionar el país de origen

del teclado, como se muestra en la Figura B.14.

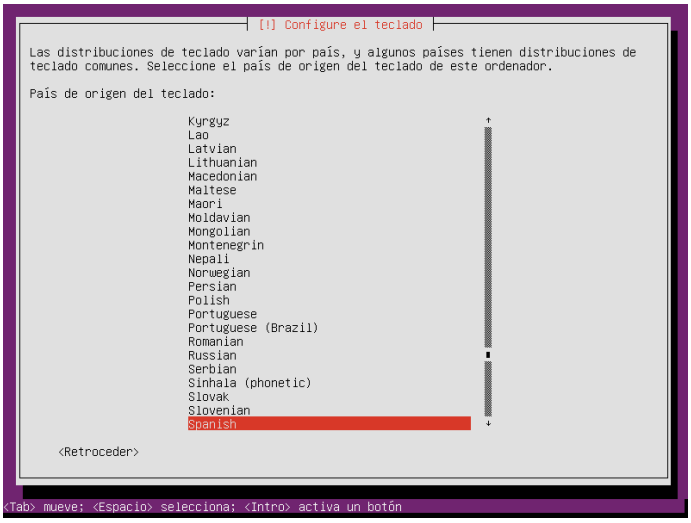

Figura B.14: Configuración de teclado.

Luego se debe seleccionar la distribución de teclado como se muestra en la siguiente Figura:

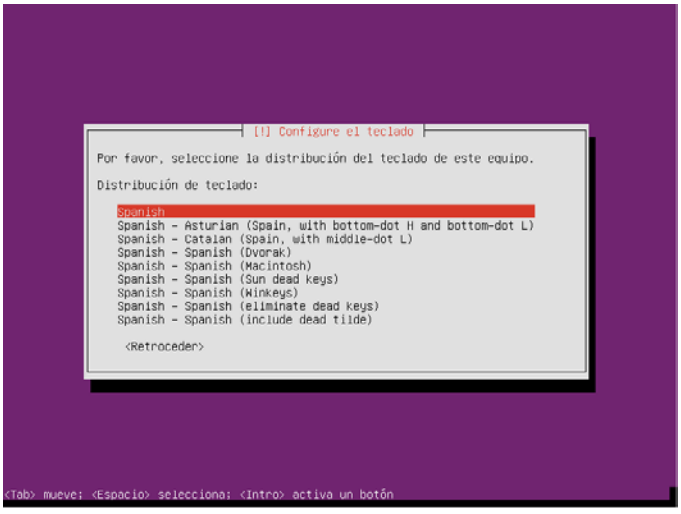

Figura B.15: Distribución de teclado.

Al cargar los datos ingresados arroja la *configuración de red* donde solicita ingresar el nombre de la máquina que identifica el sistema de red (ver Figura B.16).

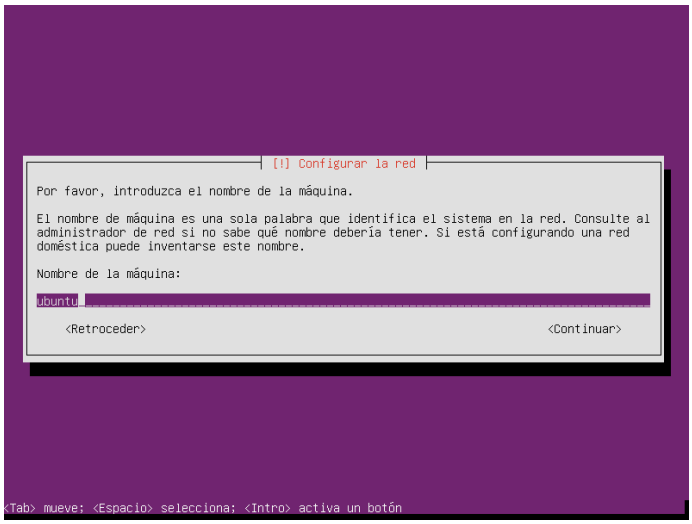

Figura B.16: Configurar la Red.

Ingresado el nombre de máquina, llega el momento de *configurar usuarios y contraseñas* donde se creará una cuenta de usuario para el uso de tareas no administrativas, se ingresa el nombre que idealmente es el nombre completo de la persona que la utilizará como se muestra en la Figura B.17.

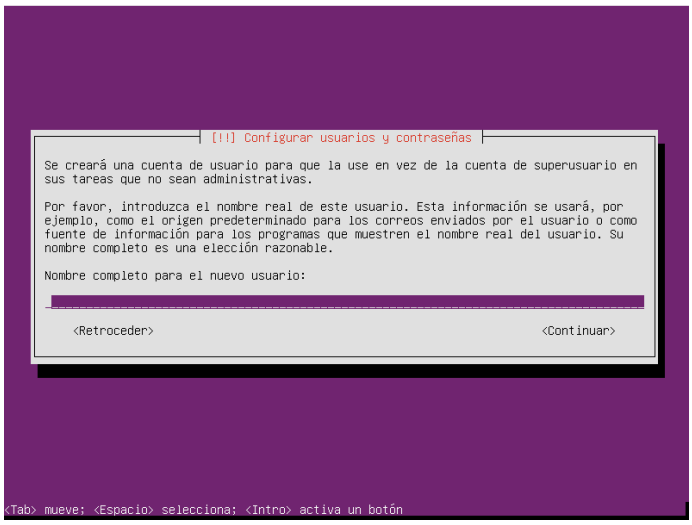

Figura B.17: Nombre completo para el nuevo usuario.

Luego se solicita un nombre de usuario, se sugiere el nombre de pila de éste (ver Figura B.18).

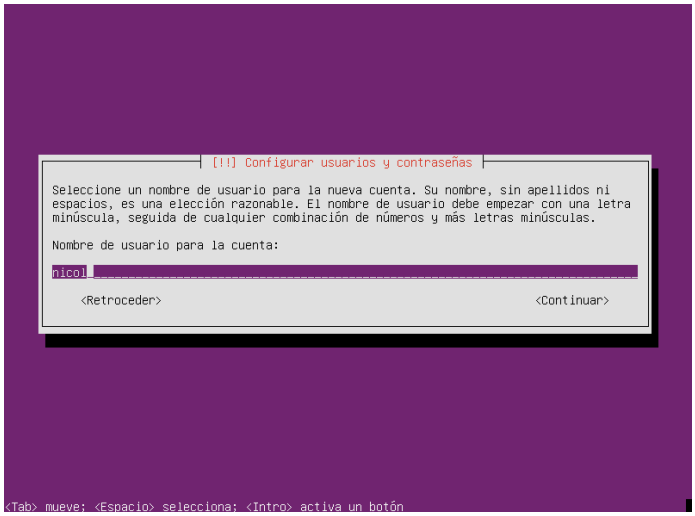

Figura B.18: Nombre de usuario para la cuenta.

Una vez ingresado el nombre de usuario se solicita una contraseña para éste que puede

tener letras, números y signos de puntuación (ver Figura B.19).

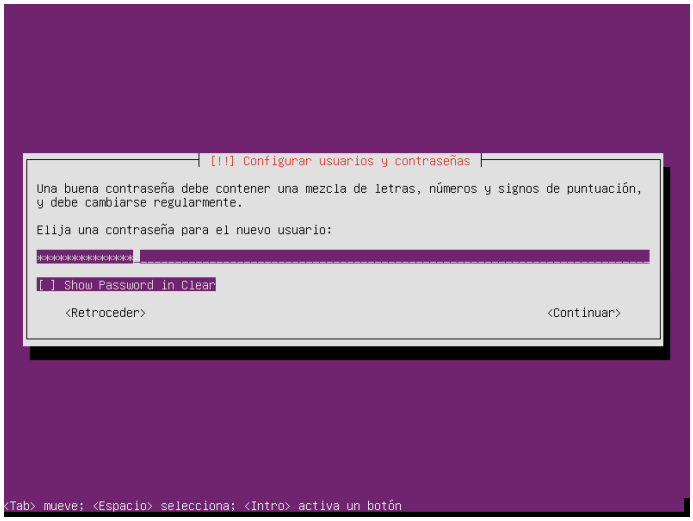

Figura B.19: Contraseña para el nuevo usuario.

Se debe repetir el ingreso de contraseña para verificar que se encuentra bien ingresada (ver Figura B.20).

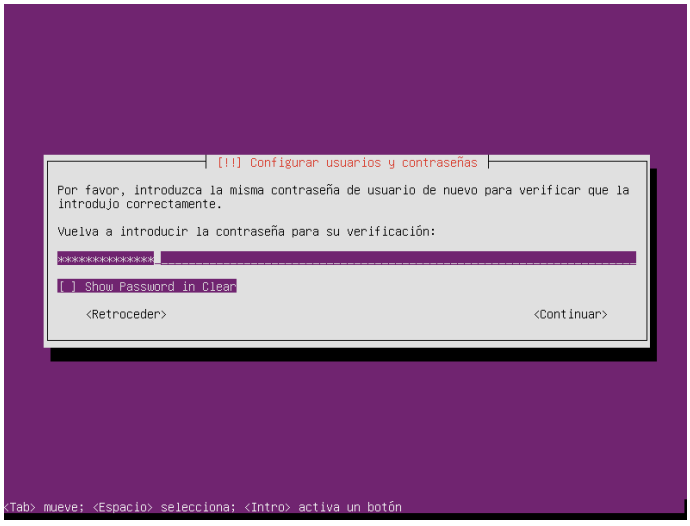

Figura B.20: Repetir contraseña.

Se cargan los datos y comienza la *configuración del reloj* donde se busca la hora en un servidor de hora de red (ver Figura B.21).

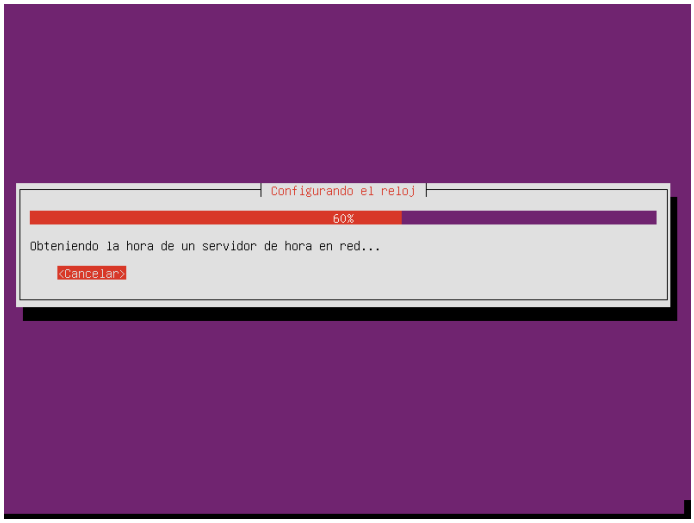

Figura B.21: Configuración del reloj.

Una vez encontrada, pregunta si la zona horaria encontrada corresponde a la zona donde se esta instalando el sistema operativo, a lo que puede responder que si es la zona correcta y continuar, o responder que no y volver a buscar la zona horaria correspondiente (ver Figura B.22).

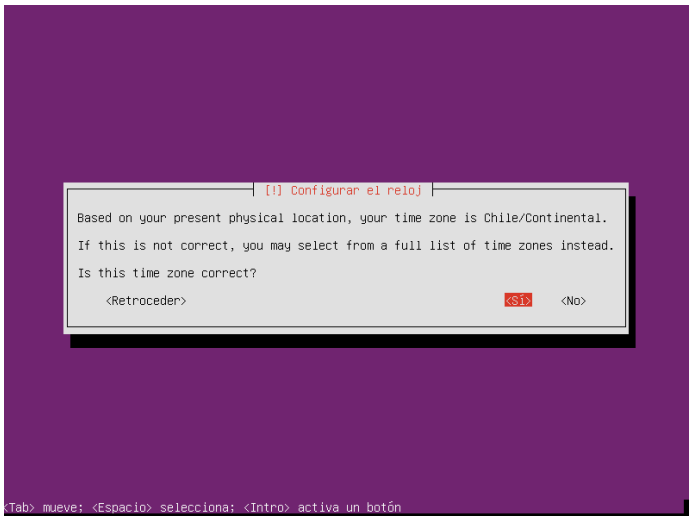

Figura B.22: Verificando configuración del reloj.

La siguiente etapa consiste en el *particionado de discos* donde se puede seleccionar la forma en la que se particionará, ya sea de forma guiada o manual como se muestra en la Figura B.23.

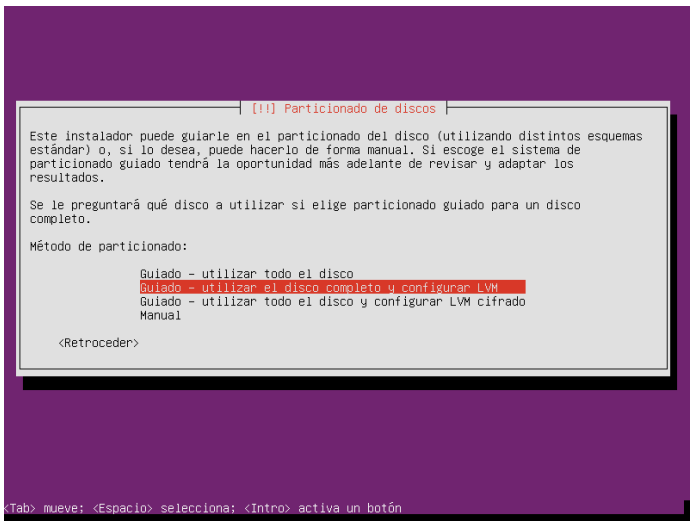

Figura B.23: Particionado de discos.

Para esta instalación se seleccionó la opción *Guiado-utilizar el disco completo y configurar LVM*. Posterior a ello se elije el disco a particionar, que fue el asignado cuando se creo la máquina virtual (ver Figura B.24).

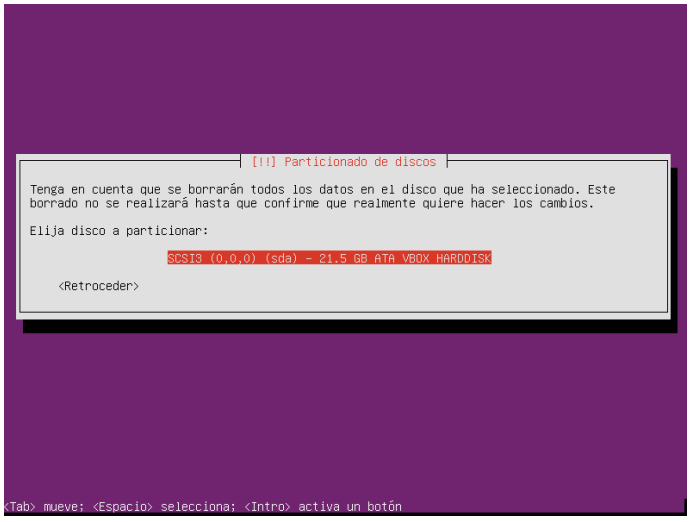

Figura B.24: Selección de disco a particionar.

Una vez seleccionado el disco, se solicita la confirmación de los cambios y configuración de LVM (ver Figura B.25).

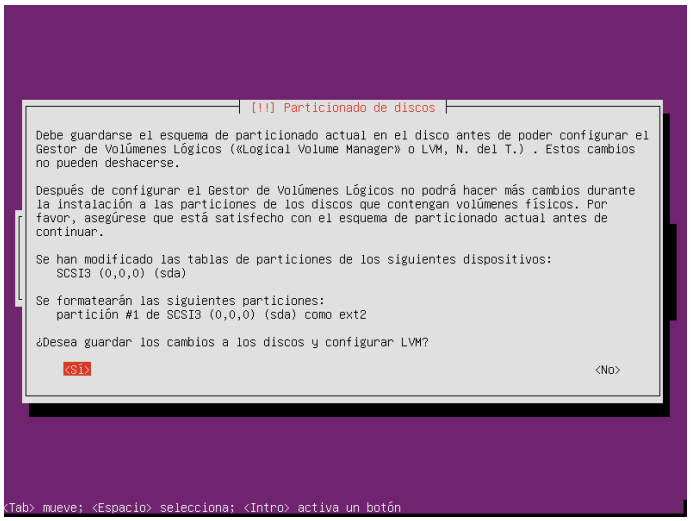

Figura B.25: Confirmación de disco a particionar.

La siguiente ventana, apreciada en la Figura B.26, explica como se puede utilizar el volumen del disco y da consejos de como ingresar la cantidad, además solicita especificar el tamaño máximo a ocupar.

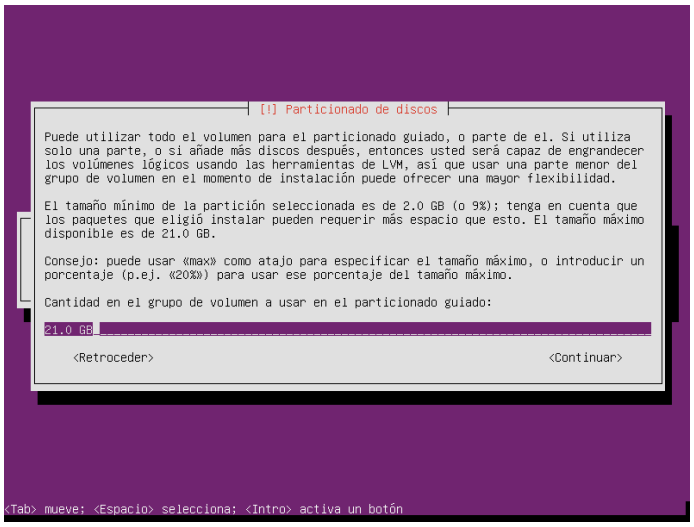

Figura B.26: Cantidad de particionado.

Ya ingresado se presenta un resumen de las particiones y configuraciones realizadas y da la opción de finalizar el proceso u de deshacer los cambios e intentarlo nuevamente (ver Figura B.27).

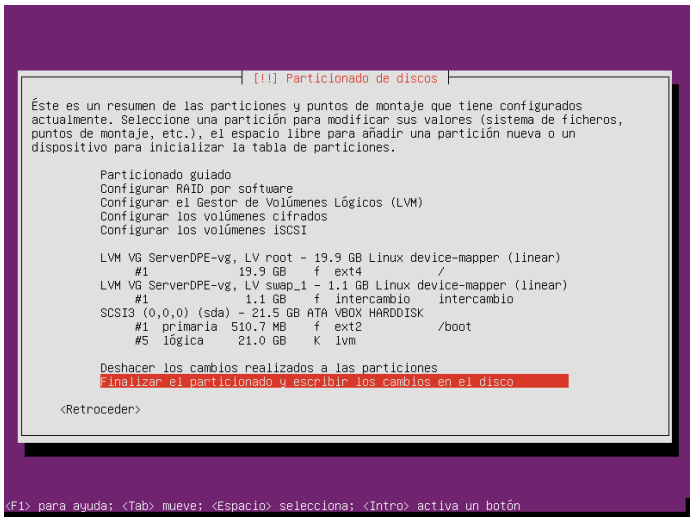

Figura B.27: Finalización del particionado.

Finalizando el particionado se descargan los ficheros y se configura de forma automática el apt, dando paso a la administración de actualizaciones que dependerá de lo que desee el usuario. En este caso optaremos por que no hayan actualizaciones automáticas (Ver Figura B.28).

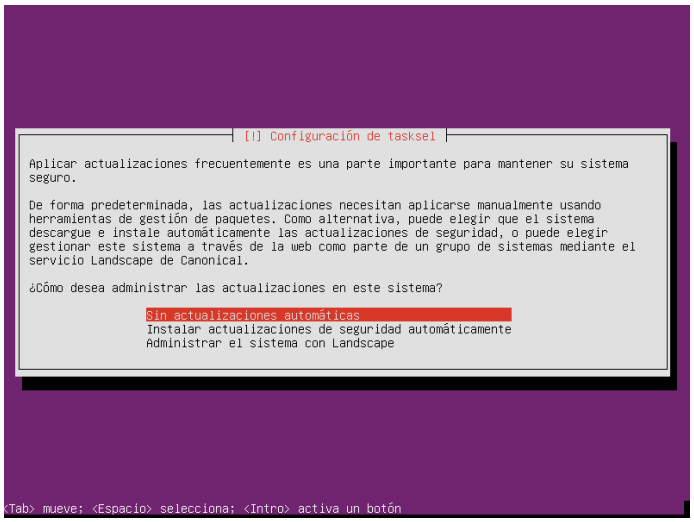

Figura B.28: Configuración de taskel.

El siguiente paso es decidir si se desea la instalación de *GRUB* en el registro principal de arranque, en este caso si se instala como muestra la Figura B.29.

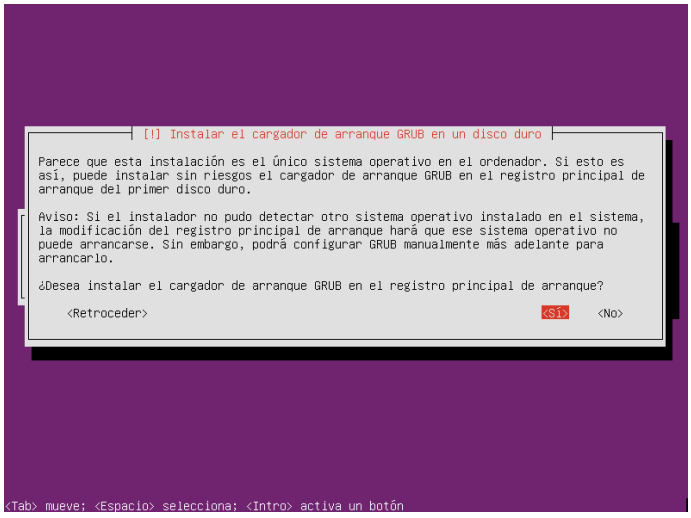

Figura B.29: Instalación del cargador de arranque GRUB.

Una vez terminada la instalación se podrá arrancar el nuevo sistema operativo desde

el disco (ver Figura B.30).

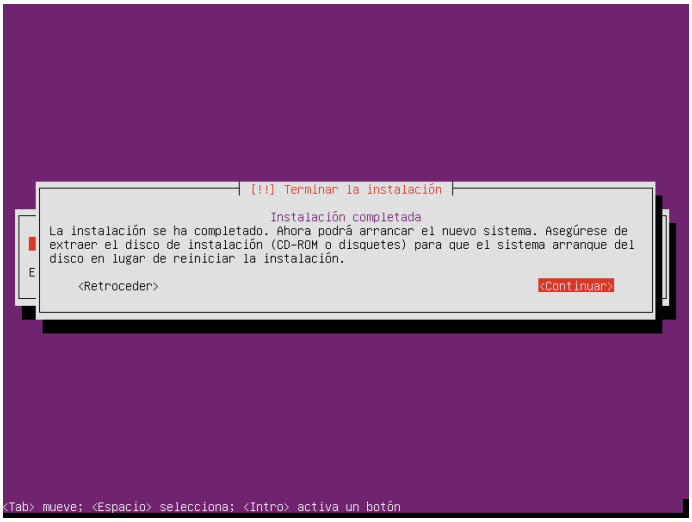

Figura B.30: Finalización de la instalación.

Al iniciar se mostrará la pantalla de la Figura B.31 donde se puede seleccionar el sistema operativo al que ingresaremos, en este caso tenemos sólo uno instalado en la máquina virtual, así que le damos enter.

Figura B.31: Iniciando Ubuntu.

Como se muestra en la Figura B.32 ya ha sido iniciado el sistema operativo, donde se puede acceder con el nombre de usuario y la contraseña que se ingresó en la instalación. Una vez identificado el usuario, se puede comenzar a utilizar el sistema.

Figura B.32: Ubuntu instalado e iniciado.

B.3. Instalación de GNOME Shell

Para la instalación de esta interfaz gráfica se debe ingresar mediante el nombre de usuario y contraseña al sistema operativo como se mostró en la Figura B.32, posterior a ello se ingresan ciertos comandos a través de la consola que darán paso a la instalación. El primer comando que utilizaremos es el que se muestra en la Figura B.33 y lo que hace es añadir el repositorio GNOME 3 al sistema.

```
~$ sudo add-apt-repository ppa:gnome3-team/gnome3-staging
```

Figura B.33: Comando para agregar repositorio GNOME 3.

Luego de añadido el repositorio GNOME 3 debemos actualizar la lista de paquetes disponibles y sus versiones, para esto se utiliza el comando de la Figura B.34 *apt update*, no obstante se debe aclarar que sólo actualiza la lista de los servidores con repositorios definidos en el *sources.list* y no es encargado de instalar ni actualizar ningún paquete.

```
sudo apt update_
```

Figura B.34: Comando para actualizar lista de paquetes disponibles.

Una vez actualizada la lista de software disponible y la versión en la que se encuentra, podemos actualizar los paquetes con el comando de la Figura B.35. Este comando utiliza un algoritmo para intentar actualizar los paquetes de mayor importancia por sobre los de menor importancia.

```
sudo apt dist-upgrade
```

Figura B.35: Comando para actualizar los paquetes.

Ya actualizados los paquetes procedemos a instalar GNOME Shell con el comando de la Figura B.36, esto tardará varios minutos.

```
sudo apt install gnome gnome-shell
```

Figura B.36: Comando de instalación.

Ya instalado de forma correcta reiniciamos nuestro sistema operativo para poder acceder con la nueva interfaz gráfica de GNOME (ver Figura B.37), esto puede realizarse de forma manual o con el comando *sudo reboot*.

Figura B.37: Interfaz gráfica GNOME Shell instalada.

B.4. Instalación MongoDB

Para la instalación de MongoDB es necesario ingresar a la página oficial de éste (https://docs.mongodb.com/), aquí se encuentra un tutorial de instalación, que incluye los comandos de instalación como se detallará a continuación: En la página oficial se especifica que MongoDB sólo es compatible con versiones de Ubuntu LTS (soporte a largo plazo) de 64 bits. Como el que fue instalado para creación de nuestro sistema, Ubuntu 16.04 LTS (xenial).

El primer paso de instalación es la importación de la clave pública como se muestra en la Figura B.38, esta garantiza la coherencia y autenticidad del paquete.

```
nicol@ubuntu:~$ sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80
--recv 2930ADAE8CAF5059EE73BB4B58712A2291FA4AD5
```

Figura B.38: Importar clave pública.

Dependiendo de la versión del sistema operativo, se ingresará un comando(Ver Figura B.39) para la creación del archivo de lista de MongoDB, (/etc/apt/sources.list.d/mongodb-org-3.6.list).

```
nicol@ubuntu:~$ echo "deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt
/ubuntu xenial/mongodb-org/3.6 multiverse" | sudo tee /etc/apt/sources.list
.d/mongodb-org-3.6.list
```

Figura B.39: Creación del archivo de lista mongodb-org-3.6.list.

Una vez creado el archivo de lista de MongoDB, se actualizará la base de datos del paquete local con el comando de la Figura B.40.

```
nicol@ubuntu:~$ sudo apt-get update
```

Figura B.40: Recargar la base de datos del paquete local.

Ya sólo queda instalar la última versión estable de MongoDB, como muestra la Figura B.41.

```
nicol@ubuntu:~$ sudo apt-get install -y mongodb-org
```

Figura B.41: Instalación de paquetes MongoDB.

Finalizada la instalación se puede iniciar sin ningún inconveniente, para esto se utiliza el comando que se muestra en la Figura B.42, si ha iniciado correctamente no emitirá ningún mensaje.

```
nicol@ubuntu:~$ sudo service mongod start
```

Figura B.42: Iniciar MongoDB.

Para verificar que el proceso de ha iniciado de forma correcta, se puede ingresar a al dirección */var/log/mongodb/* y abrir el archivo *mongod.log* que se muestra en la Figura B.43.

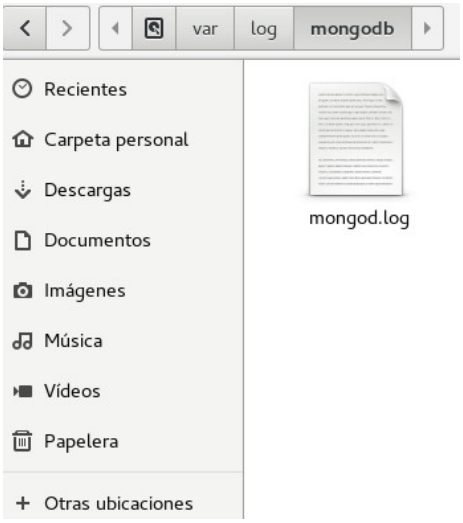

Figura B.43: Verificar que MongoDB ha iniciado correctamente.

Dentro del archivo *mongod.log*, encontrará la línea que muestra la Figura B.44, esta linea verifica que MongoDB ha iniciado correctamente, dice que se están esperando conexiones en el puerto *27017*, *port* es el puerto que mongod escucha.

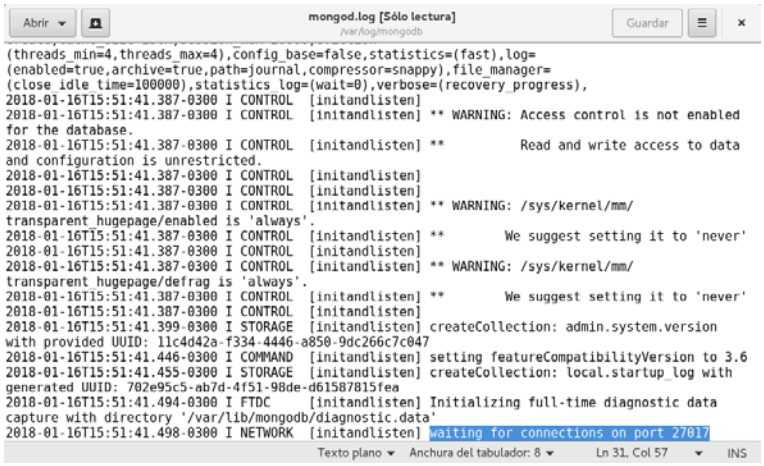

Figura B.44: Línea *waiting for connections on port 27017*.

Mientras MongoDB este corriendo puede utilizarse sin ningún problema, ya sea para crear bases de datos, editarlas o manipularlas, ahora bien si desea detener el proceso mongod puede emitir el comando de la Figura B.45.

```
nicol@ubuntu:~$ sudo service mongod stop
```

Figura B.45: Detener MongoDB.

B.5. Instalación y configuración de Robomongo

Para la instalación de Robomongo se dirige a la página oficial, donde podrá realizar la descarga del software como aparece en la Figura B.46. Una vez descargado, se descomprime mediante el comando de la Figura B.47.

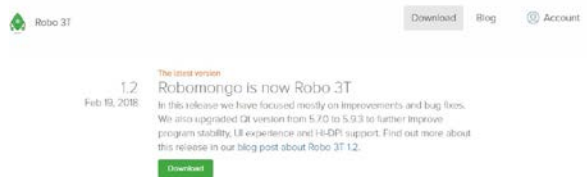

Figura B.46: Página de descarga Robomongo [36].

```
tar -xvzf robo3t-1.2.1-linux-x86_64-3e50a65.tar.gz
```

Figura B.47: Comando para descomprimir el archivo

Posterior a la descompresión, se crea el directorio de instalación (Ver Figura B.48).

```
sudo mkdir /usr/local/bin/robomongo
```

Figura B.48: Comando para creación del Directorio

Luego se copian los archivos del directorio origen al creado en el punto anterior, esto con el comando de la Figura B.49

```
sudo mv  robo3t-1.2.1-linux-x86_64-3e50a65/* /usr/local/bin/robomongo
```

Figura B.49: Comando para copiar los archivos

Entramos al directorio de instalación con el comando de la Figura B.50.

```
cd /usr/local/bin/robomongo/bin
```

Figura B.50: Comando para entrar al directorio

Ahora sólo queda correr robomongo esto con el comando *./robomongo*, lo que arrojará una ventana que solicita aceptar los términos y condiciones. (Ver Figura B.51).

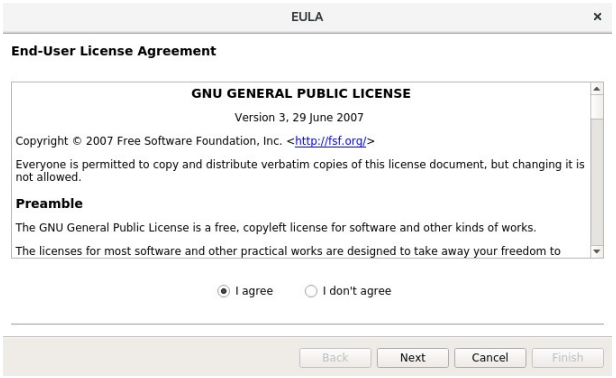

Figura B.51: Aceptar Términos y condiciones

Aceptando los términos podemos pasar a la ventana de conexiones, donde podemos generar una nueva conexión o editar alguna existente (Ver Figura B.52).

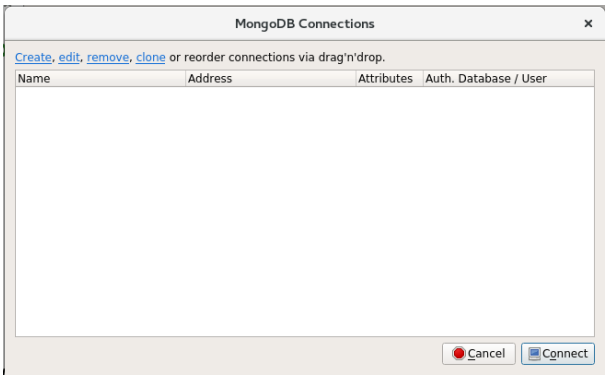

Figura B.52: Ventana de Conexión

Creamos una nueva y dejamos los datos que aparecen por defecto, la guardamos y presionamos conectar. Esto debe realizarse cuando MongoDB este corriendo sino arrojará error de conexión.

B.6. Instalación y configuración de Flask

Para la instalación de Flask es necesario tener instalado Python, con el comando *python3* se puede revisar si éste se encuentra instalado y funcional, si no funciona puede escribir sólo *pyhton*. El intérprete python esta esperando ahora una sentencia a ingresar como muestra la Figura B.53, para salir use comando *exit()*.

```
$ python3
Python 3.5.2 (default, Nov 17 2016, 17:05:23)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> _
```

Figura B.53: Interprete de Python

Ya sabiendo que python se encuentra funcional, se procede a la instalación de Flask.

Python cuenta con un repositorio oficial llamado PyPI por sus siglas en ingles *Python Package Index* donde se encuentran paquetes como Flask. Además cuenta con una herramienta llamada *pip* que cumple la función de instalar un paquete en el sistema. Entonces se utiliza el comando de la Figura B.54 para la instalación de Flask.

```
$ pip install flask
```

Figura B.54: Instalación de Flask

Ya realizado el comando, Flask se encuentra instalado en el sistema listo para utilizar.

B.7. Instalación de PyMongo

PyMongo es una librería de Python que sirve para conectarse desde Python a MongoDB. Para la instalación se utiliza el comando *pip install pymongo*.