UNIVERSIDAD DE LA REPÚBLICA URUGUAY

FACULTAD DE INGENIERÍA

Integración y automatización de procesos para la búsqueda de proteínas homólogas con alto acoplamiento molecular

Informe de Proyecto de Grado presentado por

Stephanie Casas, Sabrina Sellanes y Ana Laura Rodriguez

en cumplimiento parcial de los requerimientos para la graduación de la carrera de Ingeniería en Computación de Facultad de Ingeniería de la Universidad de la República

Usuarios

Paola Panizza, César Iglesias y Gonzalo López

Facultad de Química, UdelaR

Supervisor

Carlos Testuri

Montevideo, 22 de diciembre de 2023

Agradecimientos

Queremos agradecer a todas las personas que contribuyeron de manera significativa a la realización de esta tesis. En particular a nuestro tutor Carlos Testuri por su guía y apoyo en el desarollo de este proyecto. También queremos agradecer a todo el equipo de usuarios, Paola, César y Gonzalo, por su colaboración necesaria para llevar a cabo este trabajo. Su apoyo ha sido fundamental para el éxito de este proyecto.

También agradecemos a nuestros familiares y amigos por su aliento y comprensión durante todo este tiempo.

¡Gracias a todos!

Resumen

El cuerpo humano está formado por miles de proteínas que desempeñan un papel fundamental en los seres vivos, cada una con su estructura y funciones específicas. A pesar de que actualmente se conoce el propósito de muchas de ellas, una gran cantidad aún se desconocen en el ámbito de la biomedicina. Los investigadores trabajan arduamente en la exploración de las proteínas analizando su estructura espacial formada por un plegamiento que determina las funciones principales y su propósito; Así como las reacciones químicas que pueden ocurrir en el sitio activo de las proteínas, resultado de la interacción con ciertas moléculas denominadas sustratos. Es de especial interés para los investigadores encontrar proteínas capaces de acelerar las reacciones químicas, lo cual es crucial en el área de la medicina para la creación de fármacos específicos para el tratamiento o detección de diferentes enfermedades como la diabetes y el Alzheimer, entre otras. El presente proyecto realiza un aporte desde el ámbito informático enmarcado en la búsqueda de proteínas que se comportan y reaccionan de cierta forma frente a la exposición con un sustrato conocido. Dado que existen en bases de datos miles de proteínas que no han sido estudiadas, en este trabajo se plantea la creación de un sistema que ofrezca la posibilidad de realizar una búsqueda de forma automatizada, basada en un conjunto de proteínas y un sustrato de los cuales ya se conoce su reacción química pero se desea expandir la exploración apuntando a analizar grandes bases de datos de proteínas existentes. De la mano de la bioinformática y del uso de enfoques de investigación en el campo de la biología computacional se intenta encontrar un conjunto de proteínas que puedan reaccionar frente a un sustrato de forma similar, o mejor aún, respecto a las proteínas ya conocidas, bajo las condiciones iniciales indicadas por el usuario. A partir de esta motivación, se implementa un sistema de flujo de información proteica automatizado, denominado *workflow*, en base al trabajo manual realizado por el usuario para analizar el modelado tridimensional de proteínas por homología y acoplamiento molecular, denominado *docking* molecular. Enfocado en los resultados obtenidos luego de la búsqueda, se lista un conjunto de proteínas candidatas que presentan características que sugieren una potencial capacidad para realizar una determinada reacción química.

Palabras clave: Proteínas, Bioinformática, *Workflow*, Modelado por homología, *Docking* molecular

Tabla de Contenido

Lista de Figuras

Lista de Tablas

Lista de Abreviaturas

API: *Application programming interface*
ARN: Ácido Ribonucleico
ADN: Ácido Desoxirribonucleico
BLAST: *Basic Local Alignment Search Tool*
CEIBOS: Centro de Estudios Interdisciplinarios de Biodiversidad Orientado a aplicaciones en Salud - Udelar
CUDIM: Centro Uruguayo de Imagenología Molecular
CSS: *Cascading Style Sheets*
MER: Modelo Entidad Relación
EMBL-EBI: *European Molecular Biology Laboratory - European Bioinformatics Institute*
ETL: *Extract, Transform and Load*
FAMSA: *Fast Multiple Sequence Alignment*
FASTA: *Format for Sequence Alignment and Search Tool*
GUI: *Graphical User Interface*
HTML5: *HyperText Markup Language* versión 5
JSON: *JavaScript Object Notation*
KNIME: *Konstanz Information Miner*
NADPH: *Nicotinamida adenina dinucleótido fosfato reducido*
NCBI: *National Center for Biotechnology Information*
MB: *Megabyte*
PDB: *Protein Data Bank*
PBIS: *Pentaho BI Suite*
PIR: *Protein Information Resource*
RCSB: *Research Collaboratory for Structural Bioinformatics*
REDAM: *Reductive Aminases*
REST: *Representational State Transfer*
SOAP: *Simple Object Access Protocol*
SQL: *Structured Query Language*
XML: *Extensible Markup Language*
YASARA: *Yet Another Scientific Artificial Reality Application*

Glosario

Alineación de secuencias: Comparación de las secuencias de aminoácidos o nucleótidos de diferentes proteínas para identificar similitudes y relaciones evolutivas.

Alzheimer: Enfermedad neurodegenerativa del cerebro que afecta la memoria, el pensamiento y el comportamiento.

Aminoácido: Es la unidad básica que conforma a las proteínas. Existen 20 aminoácidos distintos.

API: Interfaz de programación de aplicaciones que permite la interacción entre diferentes programas o sistemas.

Átomo: Unidad más pequeña de materia que conserva todas las propiedades químicas de un elemento.

Bioinformática: Aplicación de técnicas computacionales y estadísticas para el análisis de datos de origen biológico, incluyendo su organización y almacenamiento así como el desarrollo de herramientas para manejarlos.

Catálisis: Proceso que aumenta la velocidad de una reacción química, a partir de una sustancia llamada catalizador.

Cofactor: Compuesto no proteico que se une a una enzima y es esencial para su actividad catalítica.

Criterios de aceptación: Conjunto de condiciones o estándares que deben cumplirse para considerar que un sistema o programa es aceptable o funcional.

Deep Learning: Rama del *machine learning* que utiliza algoritmos inspirados en el cerebro humano para crear modelos de inteligencia artificial capaces de procesar datos de manera jerárquica y automática, extrayendo características y patrones complejos.

Diabetes: Enfermedad crónica que se caracteriza por niveles altos de azúcar en la sangre debido a la incapacidad del cuerpo para producir o utilizar adecuadamente la insulina.

Docking molecular: Procedimiento computacional que intenta predecir la unión no covalente entre dos moléculas, denominadas receptor y ligando de manera eficiente.

EMBL-EBI: Centro de investigación y servicios en bioinformática con sede en el Reino Unido, que ofrece herramientas y recursos en línea para el análisis de datos biológicos.

Energía de activación: Energía necesaria para que una reacción química ocurra. Las enzimas disminuyen la energía de activación, lo cual acelera las reacciones químicas.

Enzimas: Proteínas o ciertas moléculas de ácido ribonucleico (ARN) que actúan como catalizadores biológicos, acelerando las reacciones químicas en el organismo.

Estructuras de proteínas: Configuraciones tridimensionales de las proteínas, que pueden ser visualizadas y manipuladas computacionalmente con programas especializados.

Estructuras espaciales: Configuraciones tridimensionales que adoptan las proteínas al plegarse, lo cual define su funcionalidad.

Expasy: Base de datos y plataforma en línea utilizada para el análisis y anotación de proteínas.

GUI: Interfaz gráfica de usuario, es decir, la forma en que los usuarios interactúan con un *software* a través de elementos visuales.

Heteroátomo: Es cualquier átomo que no sea carbono e hidrógeno.

Homodímero: Estructura formada por dos subunidades idénticas de una misma molécula o proteína que se unen entre sí.

Imino reductasas: Familia de enzimas que son capaces de catalizar la reducción de grupos iminas a aminas.

In silico: Expresión que refiere a modelados, experimentos o análisis que se realizan por computadora mediante algoritmos de simulación y predicción.

Monómero: Molécula simple, generalmente de peso molecular bajo, que forma cadenas lineales o ramificadas de dos, tres o más unidades.

Nucleótido: Es la unidad básica que conforma los ácidos nucleicos ADN y ARN.

Proteínas: Macromoléculas formadas por aminoácidos que tienen una función importante en la estructura y función celular, así como en la regulación de las reacciones químicas en el organismo.

Proteínas homólogas: Proteínas que comparten una secuencia de aminoácidos similar debido a un ancestro común, lo cual puede indicar una estructura y función relacionadas.

Pruebas unitarias: Evaluación de componentes individuales de un sistema o programa para asegurar su funcionamiento correcto de manera aislada.

Química fina: Rama especializada que se centra en la síntesis de compuestos químicos de alta pureza y en cantidades más reducidas, especialmente diseñados para aplicaciones específicas como la industria farmacéutica.

Reacción química: Proceso termodinámico en el cual dos o más sustancias experimentan cambios en su estructura molecular, formando nuevos productos.

Servicios REST: Estilo de arquitectura de *software* utilizado para diseñar servicios web.

Síntesis de fármacos: es el proceso químico mediante el cual se desarrollan compuestos químicos con propiedades farmacológicas específicas, con el objetivo de ser utilizados como medicamentos para el tratamiento de enfermedades o trastornos en seres humanos u otros organismos.

Sitio activo: Región de la enzima donde se une el sustrato y donde ocurre la acción catalítica.

Sustrato: Molécula sobre la cual actúa una enzima, uniéndose a su sitio activo y participando en la reacción química.

Virtual screening: Búsqueda de compuestos bioactivos a través de métodos computacionales.

Workflow: Flujo de trabajo. Conjunto de tareas secuenciales o procesos interconectados que se realizan para completar una actividad o proyecto.

Capítulo 1

Introducción

Las proteínas son macromoléculas que cumplen funciones cruciales en prácticamente todos los procesos biológicos (Berg, Tymoczko, y Stryer, 2002). Están compuestas por cadenas de un conjunto de veinte moléculas básicas denominadas aminoácidos, que se pliegan formando estructuras espaciales que determinan su funcionalidad. Algunas proteínas denominadas enzimas son capaces de acelerar reacciones químicas al unirse con alta especificidad a ciertas moléculas, llamadas sustratos, para facilitar su conversión en una nueva molécula denominada producto. La interacción se da en una parte específica de la proteína que se conoce como sitio activo. Las enzimas son capaces de acelerar las reacciones químicas en más de 10^8 veces, realizando reacciones con alta especificidad y trabajando en condiciones suaves de reacción. Por estos motivos, resultan de gran interés a nivel de laboratorio, especialmente en la producción de fármacos y otras moléculas bioactivas (Zvelebil, 2008c).

1.1. Motivación

Los métodos biocatalíticos utilizan enzimas para llevar a cabo reacciones químicas con un alto grado de eficacia y selectividad. Esta aproximación presenta ventajas significativas desde una perspectiva tanto ambiental como económica, dado que las enzimas son catalizadores altamente selectivos y eficientes, que actúan bajo condiciones suaves de reacción. Existe un creciente interés en los procesos de biocatálisis, especialmente para aplicaciones en química fina y en la síntesis de productos farmacéuticos. Una lista de nuevos tratamientos en constante crecimiento logra minimizar los efectos graves de diferentes formas de cáncer a través de la utilización de proteínas específicas en la síntesis de fármacos, abordar las complicaciones vinculadas a la diabetes mediante el desarrollo de insulinas recombinantes, y proporcionar opciones de tratamiento para accidentes cerebrovasculares y ataques cardíacos mediante la creación de fármacos anticoagulantes. Esta lista altamente selectiva es el resultado de comprender la estructura y función de las proteínas y contribuye a una mejora notable en el manejo de enfermedades (Whitford, 2005). Estos avances pueden extenderse a otras enfermedades y dependen de un conocimiento amplio y profundo de proteínas de tamaño y complejidad cada vez mayores.

Para llevar a cabo determinada reacción específica, una enzima debe tener un sitio activo con una forma y propiedades químicas precisas. Actualmente se encuentran disponibles bases de datos abiertas que proporcionan una enorme cantidad de información sobre las estructuras de las enzimas. Existen una gran cantidad de estructuras, pero aún mayor es la cantidad de secuencias. Los avances en las técnicas de modelado permiten la generación de nuevas estructuras con un nivel de certeza razonable. Es aquí donde la bioinformática puede hacer un gran aporte al minimizar costos y tiempo en la búsqueda, modelado y análisis computacional de proteínas, y brindar una aproximación al resultado del experimento efectuado en el laboratorio (Schneider, Volkamer, Nittinger, y Rarey, 2016).

Conocer la relación entre la estructura de una proteína y su función proporciona una mayor comprensión de cómo funciona la proteína y, por lo tanto, a menudo permite al investigador proponer experimentos para explorar cómo la estructura tridimensional afecta la función. Los estudios que investigan la relación entre estructura y función de las proteínas son esenciales en el desarrollo de nuevos medicamentos. La bioinformática desempeña un papel fundamental para acelerar este proceso al permitir la simulación computarizada de estas interacciones.

1.2. Objetivos

El proyecto tiene como objetivo el desarrollo de un *workflow* que automatice el proceso de modelado tridimensional de proteínas por homología, *docking* molecular y clasificación, que actualmente el equipo de química -de aquí en adelante, usuario- realiza ejecutando una serie de tareas de forma manual.

El propósito del *workflow* es buscar enzimas candidatas similares a las que se emplean en el Centro Uruguayo de Imagenología Molecular (CUDIM), para la aplicación en la síntesis de un fármaco empleado en la detección del Alzheimer. Dado que la enzima utilizada actualmente no toma el sustrato, la meta es encontrar otras enzimas que sí lo hagan y logren acelerar la reacción química, esta velocidad de reacción se determina en el laboratorio.

Se pretende ofrecer una interfaz que facilite al usuario la operativa, con la finalidad de permitir la ejecución de todos los sistemas participantes en una sola plataforma que brinde un acceso uniforme, simplificado y rápido.

1.3. Desafíos

Uno de los retos principales consiste en analizar y comprender el contexto químico en el que se encuentra inmerso el proyecto, puesto que se trata de una temática en la que el grupo de estudiantes no está especializado. Es entonces

que resulta esencial unificar conceptos entre el usuario y el grupo de estudiantes para lograr una comunicación efectiva durante la ejecución del proyecto.

Otro desafío encontrado implica efectuar la integración de los diferentes programas involucrados en el proyecto, los cuales deben procesar diversos tipos de archivos que retroalimenten los pasos del *workflow*. Esta tarea requiere una planificación y ejecución minuciosa para asegurar su éxito.

1.4. Organización del documento

En esta sección se presenta una visión general de la estructura y organización del documento, resaltando los capítulos que lo conforman. Esto permite brindar una idea clara y concisa de cómo está estructurado el informe y qué temas se abordarán en cada capítulo, lo que sirve como guía para la comprensión del contenido del informe.

El presente informe se estructura en capítulos, los cuales se resumen a continuación: En el Capítulo 1 se introduce el tema de la tesis, se presenta la motivación del proyecto y se establecen los objetivos de la investigación. Se explican las características del problema a resolver, su contexto, motivación, objetivos del proyecto y los resultados esperados. En el Capítulo 2 se proporciona el contexto teórico necesario para comprender el proyecto, incluyendo los fundamentos teóricos y conceptuales relevantes. En el Capítulo 3 se revisan los estudios e investigaciones previas relacionados con el tema del proyecto, incluyendo aquellos que hayan abordado aspectos similares o relacionados. Se realiza un análisis de la bibliografía existente en el campo de estudio, con el fin de identificar los conocimientos y avances previos que han sido relevantes para el desarrollo del proyecto. En el Capítulo 4 se detallan los requerimientos del proyecto, las herramientas analizadas y se realiza un análisis de fuentes y tipos de datos utilizados en la investigación. En el Capítulo 5 se presenta en detalle la solución propuesta. Se describe la arquitectura y el diseño del sistema propuesto, detallando los componentes, módulos y subsistemas que lo conforman. Se explican las decisiones de diseño tomadas. Además, se proporciona información sobre la implementación realizada, describiendo los pasos realizados para la construcción del sistema. Se incluyen detalles técnicos relevantes, como lenguajes de programación utilizados, plataformas, bases de datos, algoritmos o metodologías empleadas en la implementación. En el Capítulo 6 se describen los problemas detectados durante el desarrollo del proyecto y cómo se abordaron. Se presentan los resultados de las pruebas unitarias, pruebas de integración, y analisis de los resultados. El Capítulo 7 presenta una visión sobre el proceso de ejecución del proyecto y los obstáculos superados. Además, se plantean propuestas de trabajo futuro para continuar la investigación, identificando posibles áreas de mejora o líneas de investigación en el tema abordado.

Capítulo 2

Marco Teórico

En el presente capítulo, se exponen los principales conceptos y fundamentos que resultan necesarios para brindar al lector un marco de referencia integral en relación a la temática tratada en este proyecto.

2.1. Estructura y función proteica

Las proteínas, macromoléculas fundamentales en la biología, desempeñan una amplia variedad de funciones esenciales en los seres vivos. Las unidades básicas conocidas como aminoácidos, están compuestas por pequeñas moléculas formadas por átomos de carbono, oxígeno, nitrógeno, azufre e hidrógeno. La estructura única de cada aminoácido se caracteriza por una cadena lateral de átomos que se extiende desde su cadena principal, otorgándoles propiedades y funciones específicas. En la naturaleza se pueden encontrar veinte aminoácidos diferentes, y son precisamente los átomos que componen sus cadenas laterales lo que los distingue entre sí. A continuación, se presenta la Tabla 2.1 que detalla estos veinte aminoácidos, acompañada de sus abreviaturas y la codificación de nucleótidos que los define en el ADN y ARN, revelando así la conexión íntima entre la información genética y la síntesis de proteínas.

Tabla 2.1: Tabla de Aminoácidos y Tripletes de Nucleótidos

Aminoácido	**Abreviatura**	**Triplete de Nucleótidos (ARN)**	**Triplete de Nucleótidos (ADN)**
Alanina	Ala (A)	GCU, GCC, GCA, GCG	GCT, GCC, GCA, GCG
Arginina	Arg (R)	CGU, CGC, CGA, CGG, AGA, AGG	CGT, CGC, CGA, CGG, AGA, AGG
Asparagina	Asn (N)	AAU, AAC	AAT, AAC
Aspartato	Asp (D)	GAU, GAC	GAT, GAC
Cisteína	Cys (C)	UGU, UGC	TGT, TGC
Fenilalanina	Phe (F)	UUU, UUC	TTT, TTC
Glicina	Gly (G)	GGU, GGC, GGA, GGG	GGT, GGC, GGA, GGG
Glutamato	Glu (E)	GAA, GAG	GAA, GAG
Glutamina	Gln (Q)	CAA, CAG	CAA, CAG
Histidina	His (H)	CAU, CAC	CAT, CAC
Isoleucina	Ile (I)	AUU, AUC, AUA	ATT, ATC, ATA
Leucina	Leu (L)	UUA, UUG, CUU, CUC, CUA, CUG	TTA, TTG, CTT, CTC, CTA, CTG
Lisina	Lys (K)	AAA, AAG	AAA, AAG
Metionina	Met (M)	AUG	ATG
Prolina	Pro (P)	CCU, CCC, CCA, CCG	CCT, CCC, CCA, CCG
Serina	Ser (S)	UCU, UCC, UCA, UCG, AGU, AGC	TCT, TCC, TCA, TCG, AGT, AGC
Tirosina	Tyr (Y)	UAU, UAC	TAT, TAC
Treonina	Thr (T)	ACU, ACC, ACA, ACG	ACT, ACC, ACA, ACG
Triptófano	Trp (W)	UGG	TGG
Valina	Val (V)	GUU, GUC, GUA, GUG	GTT, GTC, GTA, GTG

Para constituir una proteína los aminoácidos forman una cadena no ramificada y se pliegan generando estructuras espaciales, cuya geometría define su funcionalidad. Una proteína se pliega para formar una estructura compacta. Cada tipo de proteína se pliega en una forma específica, pero de la misma forma cada vez, y éste es el estado más estable que puede adoptar. Esta estructura específica determina la función de la proteína.

Una reacción química es un proceso termodinámico, en el que dos o más sustancias transforman su estructura molecular, los enlaces químicos entre átomos se rompen y se forman nuevos enlaces, convirtiéndose en nuevas sustancias llamadas productos.

Un catalizador es una sustancia que acelera una reacción química. Las enzimas son catalizadores de reacciones químicas y, por lo general, son proteínas. Las enzimas son las encargadas de disminuir la energía de activación necesaria para que una reacción comience. Al unirse a las moléculas del sustrato, facili-

tan que las reacciones químicas sucedan de forma más rápida. La parte de la enzima donde se une el sustrato se denomina sitio activo, ya que ahí es donde sucede la acción catalítica.

En las proteínas que son enzimas, las propiedades del sitio activo se obtienen a partir de los aminoácidos que lo conforman. El grupo de aminoácidos que se encuentra presente en el sitio activo y su posición en el espacio tridimensional, le proporcionan al sitio activo un tamaño, forma y comportamiento químico específicos. Debido a estos aminoácidos, el sitio activo de una enzima es apto para unirse con una molécula objetivo en particular (el sustrato) y ayudarla a experimentar (catalizar) una reacción química. Esto ocasiona una ligera alteración en la forma de la enzima, lo que da como resultado un ajuste aún más preciso (Zvelebil, 2008c).

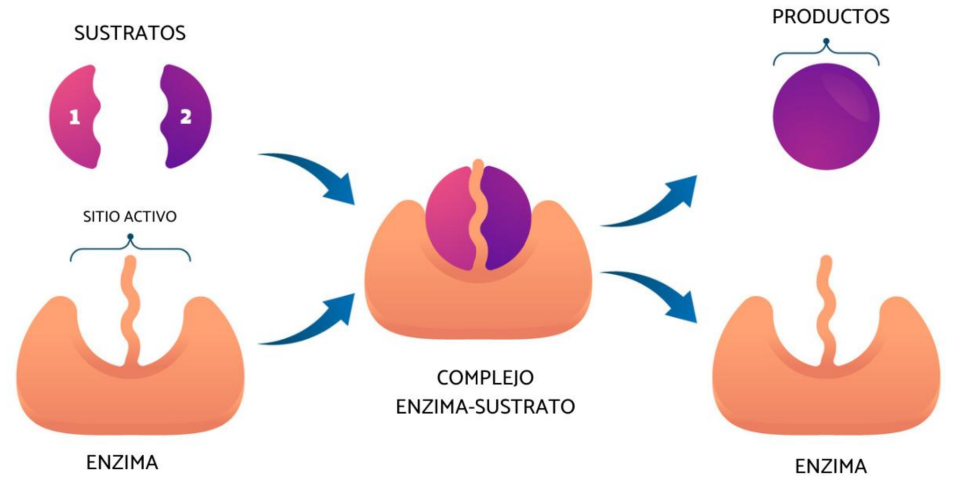

Figura 2.1: Representación del Proceso de Reacción Catalítica

En la Figura 2.1, se puede ver el proceso de reacción catalítica. La imagen comienza con la interacción entre una enzima y el sustrato. En esta etapa inicial, se pone de manifiesto la unión específica de la enzima a las moléculas del sustrato, creando un ambiente propicio para la reacción química. A medida que se avanza en la figura, se observa cómo las moléculas del sustrato se acomodan en el sitio activo de la enzima, disminuyendo la energía de activación requerida para la reacción química. Finalmente, la imagen culmina con la formación de los productos de la reacción. Después de que la enzima ha catalizado la transformación del sustrato, los productos son liberados. La enzima permanece inalterada y se encuentra lista para ser utilizada en futuros ciclos de reacción.

Esta representación visual encapsula de manera efectiva cómo las reacciones químicas son facilitadas y aceleradas por las enzimas, mostrando la esencia de la catálisis en la naturaleza.

2.2. Homología proteica y *docking* molecular

Las proteínas son fundamentales en las células porque tienen un papel esencial en la mayoría de las reacciones químicas y en la regulación de los genes. También contribuyen a darle forma y estructura a las células. En la década de 1980, hubo un avance importante en biología, se descubrió cómo leer el código de ADN para determinar la secuencia de aminoácidos de una proteína. Hoy en día, la bioinformática utiliza esta información para identificar proteínas similares y acumular conocimientos que nos ayudan a comprender las propiedades y funciones de proteínas desconocidas. La bioinformática desempeña un papel esencial al acelerar este proceso mediante modelos computacionales, brinda eficiencia en términos de costos y tiempo, permite el modelado y realización de experimentos *in silico*, con buenas aproximaciones a los resultados obtenidos de forma experimental.

La similitud de la secuencia de proteínas es una herramienta poderosa para caracterizar la función y la estructura de las proteínas debido a la gran cantidad de información conservada a lo largo de la evolución. Las proteínas homólogas con un ancestro común a menudo tienen estructuras tridimensionales similares y sitios activos relacionados. Además, las proteínas homólogas a menudo también tienen funciones relacionadas, aunque no siempre es así.

La modelización por homología, también conocida como modelización comparativa o modelización basada en conocimiento, es un método para obtener información estructural de una proteína cuando no se dispone de una estructura obtenida por un método experimental (cristalografía), lo que a su vez contribuye a la comprensión de procesos biológicos fundamentales. Se basa en el hecho de que las estructuras de proteínas homólogas se conservan mejor durante la evolución que las secuencias de aminoácidos. Esto significa que la mayoría de las proteínas con secuencias muy similares tendrán estructuras casi idénticas, e incluso las proteínas con secuencias muy diferentes pueden tener conformaciones similares.

Las proteínas que son parecidas en su estructura tridimensional suelen conservar esta similitud a lo largo de la evolución mucho más que en las secuencias de aminoácidos. Esto hace que la modelización por homología sea una técnica confiable para predecir como se ve la estructura tridimensional de la proteína que se está investigando. Para hacer esto se usa la estructura de una proteína conocida y similar (denominada proteína homóloga), como plantilla. La calidad y precisión del modelo dependen en gran medida de cuán parecidas sean las secuencias de la proteína que se estén estudiando, y la proteína que se utilice como plantilla. Cuanto más parecidas sean, mejor será la predicción y más sencillo será el proceso. Si la similitud en las secuencias es muy alta, más del 90 %, el modelo puede ser tan preciso como si se tuviera la estructura real de la proteína. Sin embargo, cuando la similitud en las secuencias es muy baja, por debajo del 25 %, es probable que se cometan errores importantes en el modelo. Por lo general, para proteínas grandes (con más de 100 aminoácidos), se pueden lograr modelos bastante precisos incluso si la similitud de secuencia

es del 25 %. Pero para proteínas más pequeñas, es necesario una similitud de secuencia mayor, alrededor del 30 % (Zvelebil, 2008a).

La modelización por homología se ha automatizado para manejar la gran cantidad de secuencias de proteínas disponibles. Esto es muy importante para entender la información genética y saber qué hacen los genes. Herramientas automatizadas como el *software* MODELLER (Webb y Sali, 2016) permiten crear modelos de la estructura de proteínas usando la información de las secuencias de manera eficiente.

La alineación de secuencias y las técnicas de búsqueda de bases de datos facilitan la identificación de proteínas homólogas. Con el creciente número de secuencias de nucleótidos y proteínas en la investigación biomédica, se ha vuelto imprescindible el uso de bases de datos para su almacenamiento y análisis. Hay muchas formas diferentes en las que se pueden diseñar las bases de datos, tanto en términos de las formas en que se almacena la información como en las formas en que se puede recuperar y analizar. Las bases de datos tienen una estructura específica que consta de archivos o tablas, cada una de las cuales contiene numerosos registros y campos, lo que permite la organización de la información en secciones distintas para su fácil recuperación y análisis.

Para poder cuantificar el grado de similitud de dos secuencias se utiliza la alineación. El alineamiento de secuencias es una herramienta fundamental en bioinformática utilizada para comparar secuencias de aminoácidos en proteínas y encontrar similitudes o patrones que puedan proporcionar información sobre la evolución, función o estructura molecular.

El proceso de alineación consiste en alinear las secuencias y encontrar regiones donde las secuencias sean similares, es decir, consiste en comparar las secuencias y alinearlas para que las posiciones correspondientes estén alineadas entre sí.

Se presenta en la Figura 2.2 un alineamiento múltiple de dos secuencias, identificadas como ‘IR044’ e ‘IR047’. Las letras presentes en las posiciones alineadas reflejan similitudes o diferencias entre las secuencias en esas ubicaciones específicas. Si se observa la misma letra en todas las secuencias alineadas en una posición dada, indica que todas las secuencias comparten el mismo aminoácido en esa posición, lo que sugiere similitud. Por otro lado, letras diferentes en las secuencias alineadas en una posición indican que las secuencias tienen aminoácidos distintos en esa ubicación, señalando diferencias. Además, los guiones (“-”) utilizados en el alineamiento representan espacios en blanco o *gaps*, indicando que en esa posición no se encuentra una correspondencia adecuada entre las secuencias. Esta información es esencial para el análisis de secuencias, ayuda a identificar regiones conservadas y variables, y proporciona información relevante sobre la función y evolución de las secuencias.

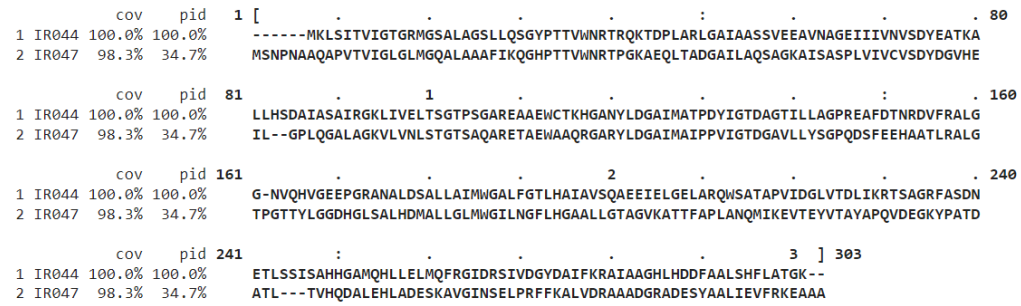

Figura 2.2: Ejemplo de alineación de aminoácidos

La técnica se utiliza para identificar similitudes entre especies, para identificar patrones comunes en una familia de proteínas y para predecir la estructura de una proteína desconocida. Se asignan puntajes según las coincidencias encontradas. El alineamiento con mejor puntuación debería ser el más razonable desde un punto de vista químico, ya que es el que alinea más posiciones homólogas. El objetivo es minimizar la cantidad de cambios (deleciones, inserciones, sustituciones) necesarios para lograr la alineación óptima, lo que se conoce como la "puntuación"del alineamiento.

El alineamiento global se utiliza para comparar secuencias completas, mientras que el alineamiento local se utiliza para encontrar regiones específicas de similitud en secuencias más largas. El alineamiento global es útil cuando las secuencias son similares en toda su longitud, mientras que el alineamiento local es útil cuando las secuencias solo tienen regiones similares. El conocimiento de esta técnica es esencial para la comprensión de muchos aspectos de la bioinformática y puede ser utilizado en una variedad de aplicaciones, desde la identificación de genes hasta la comparación de genomas completos.

Una manera de descubrir enzimas que muestren una actividad específica es a través de la estrategia basada en motivos conservados. Los motivos conservados en un alineamiento son patrones cortos de aminoácidos, y esto se emplea en el alineamiento múltiple de secuencias de proteínas cuya función ya se conoce. Las secuencias que están relacionadas en términos de función comparan patrones de distribución similares de aminoácidos críticos, los cuales no necesariamente se encuentran uno al lado del otro en la secuencia. Por ejemplo, los aminoácidos conservados que forman parte del sitio activo de una enzima pueden estar separados en la secuencia de la proteína, pero continúan siendo reconocibles en un patrón debido a que deben unirse en una configuración específica para formar el sitio activo en la estructura tridimensional.

La modelización por homología se ha convertido en una etapa esencial en la interpretación de datos genómicos y en la asignación de funciones a genes. La conservación de estructuras entre proteínas homólogas proporciona una base sólida para la modelización, aunque la similitud de secuencia y la longitud de la proteína influyen en la precisión del modelo. La automatización ha facilitado el proceso, pero el análisis cercano del modelo resultante sigue siendo crucial,

especialmente para proteínas con menor similitud de secuencia. La modelización por homología es una herramienta esencial para comprender la estructura y función de las proteínas en diversos contextos biológicos y médicos. Permite aprovechar la información de proteínas homólogas para obtener conocimientos valiosos sobre proteínas desconocidas, lo que tiene un impacto significativo en la investigación científica y la aplicación práctica en campos como la medicina y la biotecnología. Puede ser utilizada en el diseño de fármacos al predecir cómo una proteína objetivo interactuará con un compuesto específico.

La modelización por homología puede proporcionar información detallada sobre los sitios de unión a ligandos y los centros catalíticos en una proteína. A través de la comparación con proteínas homólogas de estructura conocida, es posible identificar regiones estructuralmente conservadas que desempeñan un papel crucial en la función biológica. Estas regiones pueden estar involucradas en interacciones con sustratos, cofactores o moléculas reguladoras, lo que permite una comprensión más profunda de los mecanismos moleculares subyacentes.

El acoplamiento molecular, también conocido como *docking* molecular, es una herramienta esencial en la investigación farmacéutica y la bioinformática en la que se pueden estudiar las interacciones entre proteínas y pequeñas moléculas, como sustratos enzimáticos, inhibidores y cofactores. Estos métodos de acoplamiento molecular se han convertido en una manera efectiva de modelar y analizar estas interacciones. Es relevante señalar cómo las técnicas y programas de acoplamiento molecular permiten la exploración de las interacciones entre ligandos y proteínas, así cómo la estructura de la proteína y sus características fisicoquímicas influyen en las interacciones.

El primer paso en el proceso del *docking* es la identificación del sitio de unión en la macromolécula donde el ligando se une preferentemente. Este sitio de unión suele ser una región que desempeña un papel crítico en la función biológica de la macromolécula, como el sitio activo en una enzima. Una vez que se ha identificado un sitio de unión activo o probable en una proteína, es posible modelar la interacción entre la proteína y las pequeñas moléculas, como los ligandos. Estos ligandos pueden ser sustratos enzimáticos, inhibidores u otras moléculas que interactúan específicamente con la proteína. El análisis de la unión de los ligandos a sitios específicos en las proteínas y la modelización de estas interacciones, permiten comprender cómo se relacionan la estructura y la función de la proteína. Además, al examinar las interacciones entre el ligando y la proteína, es posible identificar los determinantes estructurales y químicos que contribuyen a una unión fuerte. Este conocimiento es fundamental para el diseño de fármacos y terapias, ya que puede proporcionar información sobre cómo modificar los ligandos para lograr una interacción más efectiva y específica. Los programas de *software* de acoplamiento molecular permiten detectar sitios de unión y analizar cómo diferentes ligandos se ajustan en estos sitios (Zvelebil, 2008b).

Las estructuras de proteínas se pueden visualizar y manipular computacionalmente utilizando una variedad de programas disponibles que leen archivos de coordenadas y los convierten en una representación tridimensional visi-

ble de la proteína. Algunos de estos programas son muy potentes y útiles para analizar propiedades estructurales y funciones moleculares, además de permitir la modificación manual de moléculas. Algunos programas son gratuitos como VMD (Humphrey, Dalke, y Schulten, 1996) o de bajo costo, como YASARA (Land y Humble, 2018). Otros programas como Glide de *software* MAESTRO (*Schrödinger Release 2023-4: Glide*, 2023), *Molecular Operating Environment* (MOE) (*Molecular Operating Environment (MOE)*, 2022), ICM-PRO (*ICM-Pro*, s.f.) y *Protein–Ligand Docking Software* GOLD (Jones, Willett, Glen, Leach, y Taylor, 1997) son muy poderosos que permiten a los usuarios realizar modificaciones computacionalmente intensivas a las moléculas, pero son costosos.

2.3. *Workflow* bioinformático

Un *workflow* o flujo de trabajo se refiere a los pasos ordenados que se siguen para hacer una tarea o varias tareas relacionadas. El objetivo de un *workflow* en informática es optimizar y automatizar la realización de actividades, lo que ayuda a gestionar mejor los recursos y tener mayor productividad.

Los *workflows* se usan en diferentes áreas de la informática, como el desarrollo de *software*, la administración de sistemas, el procesamiento de datos y la gestión de proyectos. Cada *workflow* se diseña según los requisitos y objetivos específicos de cada actividad, para que las acciones se realicen en el orden adecuado y se eviten demoras o repeticiones innecesarias.

Para implementar un *workflow* en informática, se utilizan herramientas y tecnologías que permiten definir, ejecutar y controlar los pasos del flujo de trabajo. Estas herramientas pueden ser sistemas de gestión de *workflow* o plataformas de automatización de procesos. Con ellas, se pueden configurar y ajustar los pasos, reglas y condiciones del *workflow*, ya sea a través de una interfaz gráfica o de programación.

Algunas características importantes de los *workflows* en informática son la capacidad de manejar tareas en secuencia o en paralelo, asignar roles y responsabilidades a los participantes, manejar errores o excepciones, generar informes y métricas de rendimiento, e integrarse con otros sistemas y aplicaciones. También suelen contar con mecanismos de control y seguimiento, para monitorear el progreso de las tareas y tomar decisiones en base a los resultados obtenidos.

Los *workflows* ayudan a ser más eficientes y consistentes en la realización de tareas, reduciendo la posibilidad de cometer errores y mejorando la calidad del trabajo. Los *workflows* también son adaptables y escalables, lo que significa que se pueden ajustar y modificar fácilmente según los cambios en los requisitos o en el entorno de trabajo.

Un *workflow* en el ámbito de la bioinformática desempeña un papel fundamental en la gestión y automatización del procesamiento y análisis de datos. Se trata de una secuencia estructurada y ordenada de pasos que permiten abordar de manera sistemática y eficiente la manipulación y exploración de la información biológica. Al diseñar un *workflow* bioinformático, se debe realizar

la identificación de los pasos necesarios para llevar a cabo el análisis de los datos específicos que se investigarán. Esto implica comprender en profundidad el objetivo del análisis. Al definir de manera precisa y exhaustiva los pasos requeridos, se establece un marco sólido que guiará el desarrollo del *workflow*, asegurando que se cubran todos los aspectos relevantes y se alcancen los resultados deseados. La selección de las herramientas y algoritmos adecuados constituye otro elemento importante en el diseño de un *workflow*. En cada etapa del flujo de trabajo, es necesario evaluar las opciones disponibles y elegir aquellas que se ajusten mejor a las necesidades y características de los datos. Esta elección acertada implica considerar factores como la precisión, eficiencia, escalabilidad y compatibilidad de las herramientas y algoritmos con los datos y objetivos específicos del análisis. Además hay que establecer protocolos para el almacenamiento y organización de los datos, así como abordar aspectos relacionados con la privacidad y la seguridad. Es importante considerar la implementación de estándares y formatos adecuados para facilitar la interoperabilidad y el intercambio de datos entre diferentes herramientas y plataformas. Los flujos de trabajo deben estar diseñados de manera que puedan manejar eficientemente grandes volúmenes de datos y ejecutar análisis en paralelo o distribuidos. Esto es especialmente relevante en proyectos bioinformáticos a gran escala, donde la capacidad de procesamiento eficiente de los datos se vuelve fundamental para cumplir con los plazos y objetivos del análisis. Considerar la escalabilidad en el diseño del *workflow* garantiza su adaptabilidad y capacidad de respuesta ante el crecimiento de los volúmenes de datos y las demandas computacionales. Los resultados obtenidos en un análisis deben ser estructurados de manera modular y bien documentados, de modo que puedan ser fácilmente reutilizados. La integración de diferentes fuentes de datos es un desafío común en los análisis. Los *workflows* deben ser capaces de integrar datos provenientes de diversas fuentes y tipos. Esto implica considerar las diferencias en la estructura y formato de los datos, así como desarrollar métodos para combinarlos de manera coherente y estructurada. La integración de diferentes fuentes de datos en el *workflow* permite amplíar el análisis al ofrecer una perspectiva más integral y detallada, que abarca todos los aspectos interconectados y proporciona una visión más precisa y completa.

Capítulo 3

Revisión de Antecedentes

En este capítulo se realiza una revisión de los trabajos más relevantes relacionados con la automatización del modelado tridimensional y *docking* de las proteínas. Se identifican las principales características de los *pipelines* presentados y se intenta resaltar las fortalezas de los más robustos.

3.1. Diseño de *workflow* bioinformático

En el artículo de Leipzig (2016) se realiza un repaso de los principales *frameworks* de *pipelines* bioinformáticos actuales. Un *framework* de *pipelines* es un conjunto de herramientas y recursos diseñados para facilitar y agilizar la implementación y gestión de *pipelines* de procesamiento de datos, siendo en este contexto un *pipeline*, una serie secuencial de pasos utilizados para procesar datos.

El autor compara sus filosofías de diseño, y brinda recomendaciones prácticas para su construcción basadas en los requisitos de análisis y las necesidades de los usuarios. Expresa que un *framework* bioinformático debe ser capaz de mantener *pipelines* consistentes en pasos tanto seriales como paralelos, con dependencias complejas, *software* variado y múltiples tipos de archivos de datos, parámetros fijos y definidos por el usuario, y entregables. Afirma que un *pipeline* debe poder recuperarse desde el punto de control más cercano en lugar de sobrescribir archivos intermedios que puedan ser utilizados.

Con respecto a las implementaciones de *software* existentes aplicadas a la bioinformática, comenta que muchas de ellas funcionan con grandes archivos monolíticos ubicados en disco, lo que hace que la distribución de tareas de trabajo resulte poco eficiente.

3.2. Enfoque tecnológico tradicional

"@Tome" es un *pipeline* web dedicado al modelado de estructuras de proteínas y *docking* de ligandos pequeños Pons y Labesse (2009). El *pipeline* comienza por la identificación de los motivos de plegamiento, para ello utiliza Psi-BLAST (Altschul, 1997) y otros programas similares. Esto permite mediante

cálculos de conformidad, identificar los *templates* compatibles para el modelado tridimensional. Luego, las alineaciones estructurales se editan para mejorar la calidad del modelo tridimensional y se evalúan utilizando 3D-Jury (Ginalski, Elofsson, Fischer, y Rychlewski, 2003). Una función de puntuación basada en el puntaje obtenido determina las mejores veinte alineaciones. Se vuelven a editar las alineaciones manteniendo intactos los residuos conservados de la estructura y se evalúan los modelos obtenidos con el *software* Verify3D (Eisenberg, Lüthy, y Bowie, 1997). Se aplica una nueva función de puntuación y se presenta en la pantalla de resultados una alineación múltiple donde se comparan estructura y secuencia. Los resultados indican si se lograron estructuras significativamente similares para el modelado y el *docking* molecular. El usuario tiene la opción de elaborar modelos tridimensionales utilizando MODELLER. Se permiten construir hasta cuatro modelos por ejecución, mostrándose el de mejor energía en la página de resultados. El modelo resultante se evalúa con Verify3D para determinar su calidad y precisión. Finalmente, se realiza el *docking* molecular de los modelos que hayan alcanzado excelentes estructuras. Se utiliza MaxCluster (Siew, Elofsson, Rychlewski, y Fischer, 2000) para la superposición del modelo en complejo con el ligando. Para las interacciones proteína-ligando, se utilizan AutoDock y MedusaScore (Yin, Biedermannova, Vondrasek, y Dokholyan, 2008). Los resultados del *docking* molecular se clasifican y se presentan al usuario en una página, donde se incluye la mejor puntuación de interacción entre proteína y ligando, y una clasificación por puntaje de superposición o puntaje de interacción.

En la misma línea, el manuscrito (Samdani y Vetrivel, 2018) presenta un *workflow* denominado POAP que integra la ejecución de los programas Open Babel y AutoDock *suite* para mejorar el proceso de *virtual screening* de alto rendimiento en la identificación de nuevos fármacos. El sistema ofrece módulos para la preparación de ligandos y el *virtual screening* de un solo receptor o de múltiples receptores. Todos estos módulos están diseñados para ejecutarse en GNU Parallel, una herramienta de línea de comandos para ejecutar comandos en paralelo en múltiples núcleos de CPU. GNU Parallel ejecuta tareas en paralelo según la cantidad de subprocesos de CPU asignados por el usuario. POAP se distribuye libremente bajo la licencia GNU GPL e integra las herramientas Open Babel-2.4.0 junto a AutoDock-4.2.6, AutoDock Vina-1.1.2 para optimización de ligandos, AutoDockZn para *virtual screening*, y *scripts* de MGLTOOLS-1.5.7.

El módulo de preparación del ligando emplea Open Babel, una herramienta de libre acceso que se utiliza para la optimización de ligandos, la conversión de formatos, la generación de coordenadas 3D y la filtración basada en características, etc (O'Boyle, Banck, y cols., 2011) (O'Boyle, Vandermeersch, Flynn, Maguire, y Hutchison, 2011). Sin embargo, esta herramienta solo puede manejar un ligando a la vez en una sola CPU. En el módulo desarrollado, Open Babel se paraleliza y proporciona opciones para corregir los errores estereoquímicos, donde la estructura química de una molécula no coincide con la información de su disposición espacial, y la minimización de energía de los ligandos. De forma predeterminada, Open Babel también procesa los ligandos con datos erróneos,

lo que da como resultado el archivo de archivos erróneos incompletos. Estos tipos de datos de ligandos se identifican y ponen en cuarentena durante el proceso de preparación de ligandos, una función esencial para mantener una ejecución sin perturbaciones.

El módulo de *virtual screening* utiliza AutoDock, un *software* popular y ampliamente utilizado para el acoplamiento de proteínas y ligandos. El uso de múltiples CPU de AutoDock Vina depende del nivel de exhaustividad establecido durante la búsqueda. POAP ofrece la flexibilidad de elegir el número máximo de CPUs al nivel deseado de exhaustividad, a través de GNU Parallel. Esta función permite maximizar y optimizar la planificación de recursos durante la ejecución de AutoDock Vina. Mientras que AutoDockZN involucra un modelo direccional e independiente de la carga, que se puede utilizar para acoplar ligandos a metaloproteínas de zinc. Al momento de ejecutar AutoDock para el acoplamiento de proteína-ligando, se requiere preparar una cuadrícula de acoplamiento con el mapeo de asignación de tipos de átomos a los ligandos. En ejecuciones paralelas, esto puede causar problemas. Esto se soluciona usando un *script* para verificar los tipos de átomos compatibles con AutoDock en los ligandos. De esta forma se evita problemas con tipos de átomos no identificados. Además, implementa un manejo dinámico y bien optimizado de archivos, lo que permite un uso óptimo de la RAM. Tiene además accesibilidad estructurada de los archivos de entrada, salida e intermediarios. Estudios recientes de evaluación comparativa sobre herramientas comerciales de acoplamiento gratuitas demuestran que AutoDock Vina tiene un óptimo desempeño en la identificación de la mejor posición de unión de ligandos (Wang y cols., 2016).

El trabajo demuestra la aplicabilidad del módulo de acoplamiento multirreceptor de POAP basado en AutoDock Vina utilizando medicamentos aprobados por la FDA de DrugBank en cuatro objetivos: ROCK1, EthR, Pks13 y PqsA. Durante la evaluación se acoplan las proteínas a los ligandos cocristalizados correspondientes mediante AutoDock Vina, y se anotan las energías de unión. Luego se realiza el acoplamiento multirreceptor con POAP generando un archivo con las energías de unión de los conjuntos de datos de la FDA frente a las proteínas estudiadas. Los ligandos se clasifican por energía de unión máxima. Finalmente se concluyen los probables ligandos reutilizables basados en puntuaciones e interacciones intermoleculares.

Por su parte, la interfaz web gratuita IREDFisher(Yu y cols., 2022) aplica el flujo de trabajo computacional para modelar complejos enzima-sustrato utilizando imino reductasas (IREDs). Este *workflow* permite priorizar secuencias definidas por el usuario, de paneles IRED establecidos, o de bases de datos públicas curadas por IREDFisher. El usuario ingresa las secuencias iniciales y la estructura tridimensional de un sustrato como entradas y el *workflow* comienza con un preprocesamiento de las secuencias en el que se recopilan las proteínas homólogas en el *Protein Data Bank* (wwPDB consortium y cols., 2019) mediante MODELLER. Se toma como *template* de la secuencia de consulta, la proteína con la identidad de secuencia más alta. El trabajo expone que las IRED forman homodímeros con el sitio activo ubicado en la interfaz

entre los monómeros, lo que dificulta la construcción de un modelo de forma automatizada (Mangas-Sanchez y cols., 2017). Para subsanarlo, realiza la alineación de secuencias entre la proteína *template* y la secuencia de consulta en base a una sola cadena, luego la modifica mediante un *script* interno para generar un archivo de alineamiento de secuencias de doble cadena. Se generan y optimizan los modelos usando MODELLER en base a la alineación de secuencia de cadena doble modificada usando como *template* para el modelado el dímero 5OCM (Lenz y cols., 2018) dada su alta resolución y la clara presencia del sitio de unión del sustrato y el cofactor NADPH. Los modelos se evalúan en función del diagrama de Ramachandran (Hollingsworth y Karplus, 2010). Posteriormente el sustrato se acopla al sitio activo de cada modelo y se selecciona la pose que se adapta a la geometría conocida. Se utiliza la biblioteca PyMol de Python para alinear los modelos con la estructura del *template* 5OCM y así ubicar el sitio activo. Luego se generan las coordenadas tridimensionales para el sustrato con Babel (O'Boyle, Banck, y cols., 2011). Se preparan las estructuras alineadas con NADPH y la del sustrato con AutoDockTools (Morris y cols., 2009).

Para el *docking* molecular usa como centro el centro de masa del ligando unido a 5OCM y el tamaño de la caja se calcula utilizando el *script* eBoxSize.pl (Feinstein y Brylinski, 2015) teniendo en cuenta el radio de giro del sustrato. Se realiza el *docking* molecular con Autodock Vina (Trott y Olson, 2009) con exhaustividad diez. Los complejos proteína-sustrato son calificados en el paso final mediante una función de puntuación. Se toma como criterio para la clasificación la distancia entre el átomo de carbono en posición 4, C4 en el NADPH y el átomo de nitrógeno N, en la imina del sustrato. Las posiciones de acoplamiento con una distancia C4–N inferior a 3,5 Å se eliminan, y se seleccionan como mejor el modelo las poses restantes que tienen la distancia C4–N más cercana.

La validación experimental de las predicciones hechas por IREDFisher respalda su efectividad en la selección de secuencias para el descubrimiento de enzimas. En este trabajo se realizaron pruebas de validación experimental en las que el *workflow* seleccionó veinte enzimas sobre una muestra mil cuatro cientas secuencias para efectuar cinco reacciones de aminación reductiva. Para las cinco reacciones objetivo, la tasa de aciertos disminuye a medida que aumenta la dificultad de las reacciones IRED. Los resultados mostraron que IREDFisher fue capaz de seleccionar las mejores enzimas de un panel pequeño de viente secuencias, además de aumentar la tasa de aciertos en comparación con la selección aleatoria. Para la validación se utilizaron tres paneles IRED públicos, y se seleccionaron aleatoriamente veinte secuencias de cada uno de los paneles. IREDFisher recuperó el mejor *hit* entre las veinte secuencias principales con una probabilidad del 100 %. El artículo concluye que para ampliar el alcance de la enzima es necesario explorar una función de puntuación más genérica ya que la función de puntuación empleada hasta el momento en este *workflow* está diseñada específicamente para las IRED.

Este proyecto de grado se encuentra en una fase avanzada al momento de la publicación de IREDFisher y trabajos asociados.

3.3. Enfoque guiado por Inteligencia Artificial

Actualmente la tendencia en el desarrollo de *software*, apunta cada vez más a la utilización de técnicas de inteligencia artificial, tanto para el modelado de proteínas como para la clasificación de los resultados obtenidos en el *docking* molecular. Un ejemplo es Alphafold (*AlphaFold*, 2018), que emplea técnicas de inteligencia artificial y *deep learning* para predecir la estructura tridimensional de una proteína de una sola cadena a partir de su secuencia de aminoácidos.

El trabajo de Evans y cols. describe la predicción de estructuras de proteínas complejas utilizando el modelo AlphaFold-Multimer. Este modelo fue entrenado de forma similar a AlphaFold, pero extiende la predicción de la estructura a múltiples cadenas. Por otra parte, el estudio realizado por Wong y cols. combina una versión avanzada de AlphaFold, AlphaFold2, con simulaciones de *docking* molecular para predecir interacciones entre proteínas y ligandos. Los resultados de esta investigación indican que se necesitan avances en el modelado de estas interacciones, en particular mediante el uso de enfoques basados en el aprendizaje automático, para maximizar el potencial de AlphaFold2 en el descubrimiento de fármacos.

La plataforma Schrödinger (*Schrödinger Platform*, s.f.) utiliza métodos basados en la física predictiva y técnicas de aprendizaje automático para acelerar el descubrimiento de fármacos. Su proceso iterativo está diseñado para acelerar la evaluación y optimización de sustancias *in silico*, antes de la síntesis y el ensayo. Los compuestos más prometedores descubiertos en cada proyecto químico experimental se optimizan aún más a través de ciclos adicionales de análisis computacional.

Gupta y Zhou proponen un enfoque basado en aprendizaje automático para afrontar dos de los grandes desafíos que presenta la búsqueda de compuestos bioactivos a través de métodos computacionales, *virtual screening*, incluyendo el manejo de bibliotecas de compuestos cada vez más grandes y la distinción entre los verdaderos positivos y los falsos positivos. En este trabajo los autores presentan un *pipeline* basado en técnicas de agrupamiento y *deep learning* para reducir la cantidad de compuestos a examinar, y mejorar la precisión del *virtual screening*.

La revisión de antecedentes desempeña un papel esencial en el proyecto al proporcionar un panorama actualizado de las tecnologías y herramientas disponibles. Este proceso es de gran utilidad para tomar decisiones más fundamentadas, evitar duplicaciones de esfuerzos y recursos, y comprender los desafíos existentes. De esta forma, contribuye a mantenerse al tanto de las últimas investigaciones y a optimizar el enfoque del proyecto.

Capítulo 4

Análisis

En este capítulo se presenta la especificación de requerimientos acordados con el usuario, el análisis del problema y las decisiones tomadas para realizar su desarrollo.

4.1. Requerimientos

En base a la propuesta del proyecto y al relevamiento realizado con el usuario se derivan los requerimientos:

1. Conformar un *workflow* que interconecte los programas necesarios para el proceso de selección de proteínas homólogas, modelado y *docking* molecular.
2. Proveer una interfaz al usuario que permita el ingreso de los parámetros utilizados en los diferentes programas involucrados en el *workflow*.
3. Implementar un sistema de almacenamiento de resultados intermedios con el fin de ponerlos a disposición del usuario para su posterior análisis.
4. Presentar una clasificación de los resultados del *docking* molecular basada en la distancia establecida por parámetro.

4.2. Herramientas analizadas

En esta sección se presenta una descripción de las herramientas analizadas y se justifica su elección como parte de la solución propuesta.

4.2.1. Plataforma para desarrollo del *workflow*

En cuanto a la plataforma para el desarrollo del *workflow*, se sugiere utilizar KNIME Analytics Platform (Berthold y cols., 2009) en la propuesta del proyecto. Sin embargo, se plantea la posibilidad de utilizar PENTAHO BI Suite(PBIS) (*PENTAHO BI Suite*, 2004) en su lugar, debido a la experiencia previa por parte del grupo de estudiantes, en el uso de esta plataforma. Se realiza entonces un análisis comparativo entre ambas opciones, teniendo

en cuenta los aspectos más relevantes para la elección de la plataforma. Los resultados se resumen en la Tabla 4.1.

Tabla 4.1: Análisis comparativo de plataformas

	KNIME Analytics Platform	PENTAHO BI Suite
Plataforma de *software* libre	✓	✓
Ofrece procesos ETL	✓	✓
Cuenta con extensiones específicas aplicables en bioinformática	✓	×
Previo conocimiento de la herramienta por parte del grupo	×	✓

Si bien ambas plataformas son de propósito general e incluyen nodos para ejecución de *scripts* de Python, lo que amplía las opciones de desarrollo, KNIME cuenta con extensiones de dominio especializadas en el área de la bioinformática, y esto puede llegar a ser de gran utilidad para la incorporación de nuevos desarrollos a futuro.

Otro punto a favor de KNIME es que se pueden utilizar extensiones, y que con la evolución de las versiones no se eliminan, solo se vuelven obsoletas, lo que garantiza que los flujos desarrollados con versiones anteriores sigan funcionando. Tras unas pruebas de concepto de la plataforma se valora como un *software* muy intuitivo al uso y que está bien documentado, determinando la elección de esta plataforma para la implementación del *workflow*.

Posteriormente, se divide el análisis en dos etapas: la primera, enfocada en las herramientas para el modelado por homología y la segunda en las herramientas para el *docking* molecular. Para llevar a cabo el modelado por homología es necesario realizar el alineamiento de secuencias, buscar los motivos conservados, identificar secuencias homólogas que cumplan con los motivos y, por último, explorar secuencias similares a las obtenidas en el paso anterior que ya tengan definido su modelo tridimensional, con el fin de obtener el modelado de la secuencia en cuestión. Por su parte, el *docking* molecular implica el análisis de la interacción entre dos moléculas para predecir su estructura y función, y se puede realizar mediante diferentes programas y técnicas. Es fundamental escoger las herramientas adecuadas para cada etapa del análisis con el fin de obtener resultados precisos y confiables.

En la siguiente subsección se analizan las herramientas para cada paso del modelado por homología.

4.2.2. Modelado por homología

En primer lugar, se realiza un análisis de las herramientas sugeridas por el usuario para realizar la tarea de alineamiento. Entre las opciones propuestas se incluyeron MEGA X (Tamura, Stecher, y Kumar, 2021), CLUSTAL-OMEGA (*Clustal Omega*, 2011), MUSCLE (*MUSCLE*, 2004), y FAMSA (*FAMSA*, 2016).

MEGA X es caracterizada por el usuario por ofrecer resultados confiables. Utiliza los algoritmos Muscle y ClustalW (*MEGAX-Help*, s.f.) (ver Apéndice 2), y cuenta con una interfaz de línea de comandos. Sin embargo se descarta debido a su poca flexibilidad, pues requiere especificar el análisis a ser ejecutado y generarlo exclusivamente a partir de su propia interfaz GUI junto a los parámetros de entrada (*MEGAX-Help - Running in Command-Line Mode*, s.f.). Además, al tratarse de *software* propietario, no pone a disposición su código fuente.

CLUSTAL-OMEGA es otra herramienta conocida por el usuario. Tiene una limitante de procesamiento de cuatro mil secuencias o un máximo de 4MB por archivo de entrada. Ofrece servicios bajo protocolos REST y SOAP (*Clustal Omega Help and Documentation*, s.f.).

MUSCLE utiliza el algoritmo Muscle y ofrece servicios a través de una API REST. Tiene la limitante de procesamiento de un máximo de quinientas secuencias por archivo.

FAMSA, por su parte, es *software* libre bajo uso no comercial y ofrece una interfaz por línea de comandos, siendo especialmente adecuado para grandes conjuntos de secuencias. Utiliza el algoritmo FAMSA. Aunque es desconocido por el usuario, se considera como una opción.

Finalmente, se acuerda con el usuario emplear MUSCLE y ejecutarlo en diferentes hilos de ejecución para optimizar el procesamiento, agrupando de a cuatrocientos noventa y nueve secuencias y agregando en cada grupo la secuencia de referencia.

En segundo lugar, se procede a analizar las herramientas disponibles para la búsqueda de motivos en un conjunto de secuencias. Entre las opciones consideradas se encuentran PRATT Expasy (*PRATT*, 2003) y Pratt EMBL-EBI (*Pratt*, 2002). El usuario sugiere PRATT Expasy, sin embargo, se determina que esta herramienta no ofrece servicios para su ejecución y la alternativa de extraer los datos directamente desde la web implica un riesgo para el correcto funcionamiento del sistema. Por otro lado, Pratt EMBL-EBI ofrece tanto una interfaz web como servicios para su consumo, lo que lo hace una opción más viable y segura. Por lo tanto, se acuerda con el usuario utilizar Pratt EMBL-EBI para evitar posibles dependencias futuras frente a actualizaciones de la web de PRATT Expasy.

En tercer lugar, se procede a analizar las herramientas disponibles para la búsqueda de secuencias a partir de motivos conservados. Entre las opciones

evaluadas se encuentra SCAN PROSITE (*ScanProsite tool*, 1996), una herramienta sugerida por el usuario debido a su conocimiento previo. Esta aplicación ofrece la posibilidad de utilizar un *script* para ejecutar la búsqueda de forma local, lo cual requiere previamente la descarga de las bases de datos necesarias. No obstante, SCAN PROSITE también ofrece la opción de consumir un servicio REST (*ScanProsite - user manual*, s.f.) para llevar a cabo la tarea. Tras analizar las ventajas y desventajas de ambas alternativas, se decide optar por la segunda opción con el objetivo de evitar la necesidad de mantener actualizadas las bases de datos de proteínas locales, dado su crecimiento exponencial.

En el cuarto lugar del análisis, se evalúan las herramientas disponibles para la búsqueda de templates para el modelado. Entre ellas se encuentra BLAST (Altschul, 1997), sugerido por el usuario debido a su experiencia y conocimiento del dominio. Este *software* proporciona tres variantes para su uso: una *suite* de herramientas de línea de comandos para su ejecución en un servidor local, la posibilidad de ejecutar búsquedas en un servidor en la nube, y una API REST para realizar búsquedas.

De las tres variantes de BLAST, se descarta la interfaz por línea de comandos debido a que se requiere mantener de forma local las bases de datos, las cuales se actualizan diariamente (*BLAST-Help. Download*, s.f.). Esto implica descargarlas de forma periódica. La ejecución de BLAST en la nube, aunque es la opción más indicada según su descripción, se excluye debido al uso de los servicios de Google Cloud (*Google Cloud*, 2008), que ofrecen cierta actividad gratuita, pero exigen pago por su uso general.

Por último, se evalúa la API de BLAST y se detectan las siguientes restricciones de uso: los servidores NCBI BLAST son un recurso compartido en los que los usuarios interactivos tienen prioridad, por lo que las solicitudes de grandes volúmenes pueden ser derivadas a una cola más lenta o incluso bloquearse. Además, no se permite contactar a un servidor más de una vez cada diez segundos, ni hacer *pooling* de ningún *Request ID* más de una vez por minuto. Es necesario proporcionar un correo electrónico para que el NCBI pueda contactar al usuario en caso de que surja un problema. En caso de necesitar enviar más de cincuenta búsquedas, se recomienda ejecutar los *scripts* los fines de semana o de lunes a viernes entre las 9:00 PM y las 5:00 AM (UTC-5). BLAST es más eficiente si se envían múltiples consultas como una búsqueda en lugar de enviar cada consulta como una búsqueda individual (*BLAST-Help. Developer info*, s.f.). Se analiza que estas restricciones no representan un problema para el procesamiento esperado. Por otra parte, para la búsqueda de la estructura tridimensional de las proteínas, se identifica que *Protein Data Bank*, una base de datos ampliamente conocida y utilizada, está disponible y proporciona servicios para la descarga de archivos. Por lo tanto, se decide utilizar la API de BLAST para la búsqueda de modelos tridimensionales similares y *Protein Data Bank* para la obtención de los *templates* para el modelado.

Para la última etapa del modelado por homología se evalúa la herramienta más adecuada para llevar a cabo el modelado tridimensional. La herramienta seleccionada fue MODELLER, recomendada por el usuario debido a su conoci-

miento previo en su uso. Este *software* es de libre acceso para fines académicos y proporciona instaladores para los sistemas operativos Linux, Mac y Windows. Además, ofrece diversas opciones de modelado, siendo la opción Avanzado (Sali, 1989) la elegida para realizar esta tarea.

4.2.3. *Docking* molecular

Se evalúan distintas herramientas para llevar a cabo la etapa de *docking* molecular. Entre las opciones se encuentran YASARA, que ofrece una combinación de visualización, modelado y *docking* molecular, aunque solo su herramienta YASARA View es de acceso gratuito, y requiere licencia para sus herramientas de modelado y *docking* molecular (*YASARA-Help*, s.f.). Otros ejemplo de *software* que pueden ser utilizados para este propósito son GOLD y MOE que también requieren licencia.

Otra herramienta evaluada es *LigandScout Extensions for the KNIME Workbench* (*LigandScout Extensions for the KNIME Workbench*, 2005), una extensión para KNIME que incluye herramientas de diseño y modelado molecular, como el nodo visor tridimensional para visualización de moléculas y el nodo AutoDock Vina para experimentación de *docking* molecular. Sin embargo, esta herramienta es propietaria y está sujeta a una licencia de *software* comercial.

Por su parte, AutoDock (*AutoDock*, 1989) es una *suite* de código abierto que ofrece herramientas automatizadas para llevar a cabo el *docking* molecular, especialmente para predecir cómo se unen las moléculas pequeñas, como los sustratos, a un receptor de estructura tridimensional conocida. Dentro de esta *suite*, se encuentra AutoDockTools, una interfaz gráfica que simplifica la preparación de los archivos de entrada para el *docking* molecular y la ejecución de los programas correspondientes. Asimismo, AutoDock Vina presenta mejoras significativa en la precisión de las predicciones de resultados, y permite la entrada de archivos en formato `pdbqt` para el ligando y el receptor, generando un archivo de salida en formato `pdbqt` que contiene la información energética y las mejores posiciones del ligando.

Finalmente, se opta por utilizar la *suite* de Autodock Vina debido a que es un *software* libre.

4.3. Análisis de fuentes y tipos de datos

En la investigación de la biología molecular y la genética, se utilizan diversas bases de datos y formatos para almacenar y manipular información sobre proteínas. *Protein Data Bank* (PDB) es una base de datos que contiene información sobre estructuras tridimensionales de macromoléculas determinadas experimentalmente como se mencionó en la Sección 4.2, mientras que UniProtKB (*UniProt*, 2003) es una base de datos que contiene información sobre proteínas, incluyendo su secuencia, función y artículos de investigación inde-

xados. UniProtKB se compone de dos sub-bibliotecas: UniProtKB/Swiss-Prot y UniProtKB/TrEMBL.

UniProtKB/Swiss-Prot es una base de datos no redundante y de alta calidad obtenida mediante anotación manual, y se enfoca en proporcionar secuencias de proteínas fiables y relacionadas con anotaciones de alto nivel, como descripciones de la función de la proteína y su estructura de dominio. Además, cuenta con una redundancia mínima y un alto nivel de integración con otras bases de datos. Por otro lado, UniProtKB/TrEMBL realiza traducción automática de secuencias de proteínas y contiene secuencias de proteínas no anotadas que se actualizan automáticamente a medida que se generan nuevas secuencias de proteínas.

En cuanto a los formatos utilizados, se utiliza el formato `fasta` para representar secuencias de aminoácidos. Se trata de un archivo de texto plano en el que cada secuencia comienza con un encabezado de una sola línea que comienza con el carácter ">"seguido de un identificador de secuencia y una breve descripción. La línea siguiente contiene la secuencia en sí misma. La secuencia puede estar en una sola línea o dividida en varias líneas. Por otra parte, el formato `pdb` describe las estructuras tridimensionales de las macromoléculas determinadas experimentalmente. Los archivos contienen información sobre la estructura tridimensional de la macromolécula, incluyendo las coordenadas espaciales de los átomos que la conforman, así como información adicional como ser el autor de la estructura, la resolución de la estructura, la técnica utilizada para resolverla y los descriptores de la proteína, como el nombre y la función. También se utiliza el formato `pdbqt`. El formato `pdbqt` (no estándar) es una extensión del formato `pdb` utilizado por el programa AutoDock para realizar simulaciones de acoplamiento molecular. Incluye información sobre la posición y tipo de átomos, información de conectividad, cargas y radios atómicos de los átomos, así como información sobre los puntos de acoplamiento y los tipos de átomos que pueden interactuar con el ligando. Por último, el formato PIR, legible por los programas BLAST y MODELLER para la comparación de secuencias y el modelado de proteínas, utiliza una estructura de registro para almacenar información de secuencia y anotación. Cada registro comienza con una línea de encabezado que comienza con un símbolo de asterisco, seguida de información de identificación y datos de secuencia. A continuación, se proporciona información de anotación, como la estructura tridimensional y la función.

Para la manipulación de los datos, se utilizan expresiones regulares para extraer datos de interés en los archivos de texto plano y el parseo en el caso de los documentos XML. Una descripción detallada de los formatos empleados se encuentra en el Apéndice 1.

Estas bases de datos y formatos son fundamentales en la investigación de la biología molecular, ya que permiten el almacenamiento, manipulación y análisis de información esencial para comprender la estructura y función de las proteínas.

Capítulo 5

Solución Propuesta

En el presente capítulo se describe la arquitectura de la solución implementada para el cumplimiento del objetivo principal del proyecto, en el cual se plantea la realización de un *workflow* enfocado en el análisis y procesamiento de proteínas utilizando las herramientas definidas en la Sección 4.2

5.1. Arquitectura y Diseño

La solución técnica propuesta se diseña con el objetivo de abordar los desafíos que presenta el usuario al momento de realizar la ejecución de sus tareas. La propuesta de la solución se basa en una arquitectura de cuatro capas interconectadas: *frontend*, *backend*, plataforma de creación del *workflow*, y programas/servicios externos. Esta estructura modularizada permite gestionar cada componente de manera independiente, lo que facilita el mantenimiento y la evolución constante de la aplicación. A su vez, esta solución también se crea con el objetivo de garantizar una experiencia de usuario fluida, una comunicación eficiente entre los componentes del sistema y una integración exitosa con recursos externos.

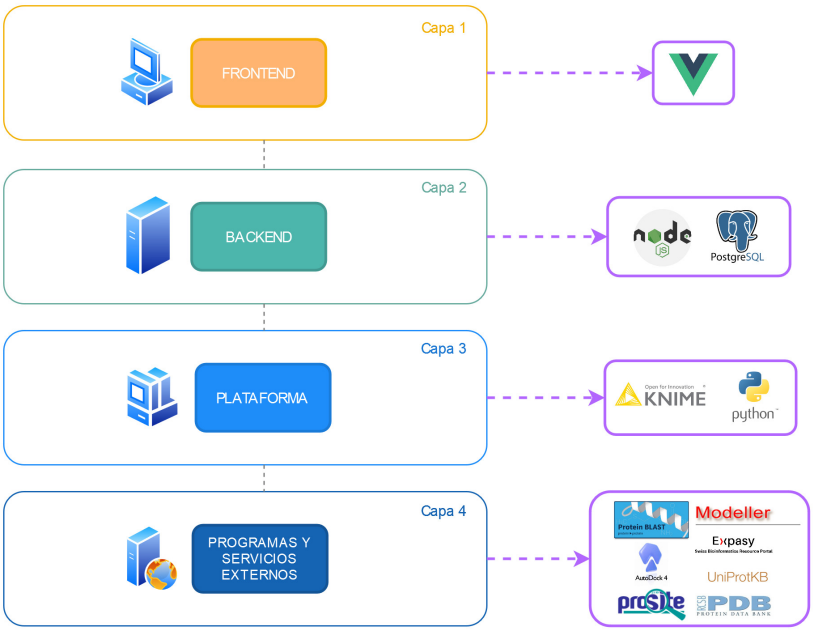

Figura 5.1: Diseño de arquitectura

En la Figura 5.1 se presenta un diagrama donde se puede visualizar el diseño de la arquitectura para la solución propuesta con sus cuatro capas. En primer lugar, el *frontend* es el responsable de proporcionar la interfaz con la que los usuarios interactúan.

La capa de *backend* juega un papel fundamental en la solución técnica, está compuesta por un servidor y una base de datos relacional. Se implementa un servidor que gestiona las solicitudes del usuario, procesa los datos y coordina la comunicación entre el *frontend* y la plataforma KNIME. Para lograr una comunicación óptima, se opta por el uso de API REST, esto permite una interacción estructurada entre los diferentes componentes y facilita futuras expansiones y mejoras, más adelante se presenta la API definida. Por otra parte, la base de datos relacional desempeña un papel esencial en la persistencia de datos. Esta base de datos almacena las configuraciones y preferencias ingresadas por el usuario a través del *frontend*. Las interacciones entre el servidor y la base de datos se gestionan mediante consultas SQL, asegurando un almacenamiento y recuperación eficientes de la información.

La capa tres contiene la integración con la plataforma KNIME, otro componente esencial de la solución, podría decirse que es uno de los pilares destacados de la arquitectura. Se encarga de la transferencia, tratamiento y procesamiento de los datos de todo el flujo de trabajo, interactuando con el *backend* y servicios/programas externos que se encuentran en la capa superior. En la plataforma se definen los nodos necesarios para la ejecución de todas las tareas ejecutadas por el usuario, permitiendo la automatización de procesos complejos, ampliando la capacidad del sistema para realizar análisis y cálculos

sofisticados.

Por último, en esta capa se incorporan un conjunto de *scripts* para procesar tareas específicas que requieren lógica y cálculos complejos. Estos *scripts* se ejecutan en esta capa para optimizar el rendimiento y la eficiencia en el procesamiento de datos.

La capa superior de administración de programas/servicios externos se diseña con un propósito esencial: separar y gestionar de manera eficiente los programas y servicios externos que son necesarios para completar el procesamiento de los datos dentro del sistema. Esta capa despliega un enfoque estratégico para optimizar la funcionalidad del sistema y orquestar con mayor precisión la interacción con componentes externos.

Para la solución técnica se combinaron buenas prácticas de diseño de *software*, la comprensión de las necesidades de los usuarios y los objetivos del proyecto. Cada decisión de diseño, desde la elección de las tecnologías hasta la implementación de la arquitectura en capas, se enfoca en la usabilidad, la funcionalidad y la eficiencia en los tiempos de ejecución del sistema.

En las siguientes subsecciones se profundiza su explicación, donde se muestra con mayor detalle los aspectos específicos de cada componente y cómo interactúan entre sí.

5.1.1. Descripción funcional

La descripción funcional se expone con un enfoque global, centrado en el funcionamiento del sistema a nivel de componentes y responsabilidades.

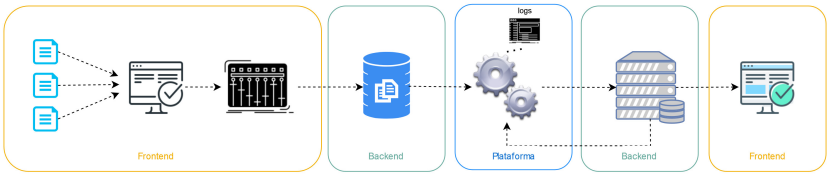

Figura 5.2: Descripción funcional

Analizando la Figura 5.2 de izquierda a derecha, en primer lugar se hace referencia a la interacción del usuario con el *frontend*, indicando los datos de entrada y la calibración de parámetros. Una vez calibrado ocurre la etapa de almacenamiento de las configuraciones e información de entrada en el *backend*, esto es persistido en una base de datos relacional, (más adelante en el documento, se presenta el modelo entidad relación) y en el *file system* definido dentro del servidor. En este momento el sistema se encuentra en condiciones de realizar la ejecución del *workflow* implementado en la plataforma KNIME generando los logs correspondientes en cada paso del flujo. En cada etapa se almacenan en el *backend* los resultados intermedios obtenidos y a su vez, estos se utilizan para procesamientos posteriores. Una vez culminada la ejecución,

se almacenan los resultados y el usuario visualiza el análisis en el *frontend*.

5.1.2. Diagramas de secuencia

En esta fase se describe a alto nivel las interacciones del usuario con el *frontend* y sus correspondientes hilos de ejecución interactuando con el servidor y KNIME, esto puede visualizarse en la Figura 5.3 a continuación.

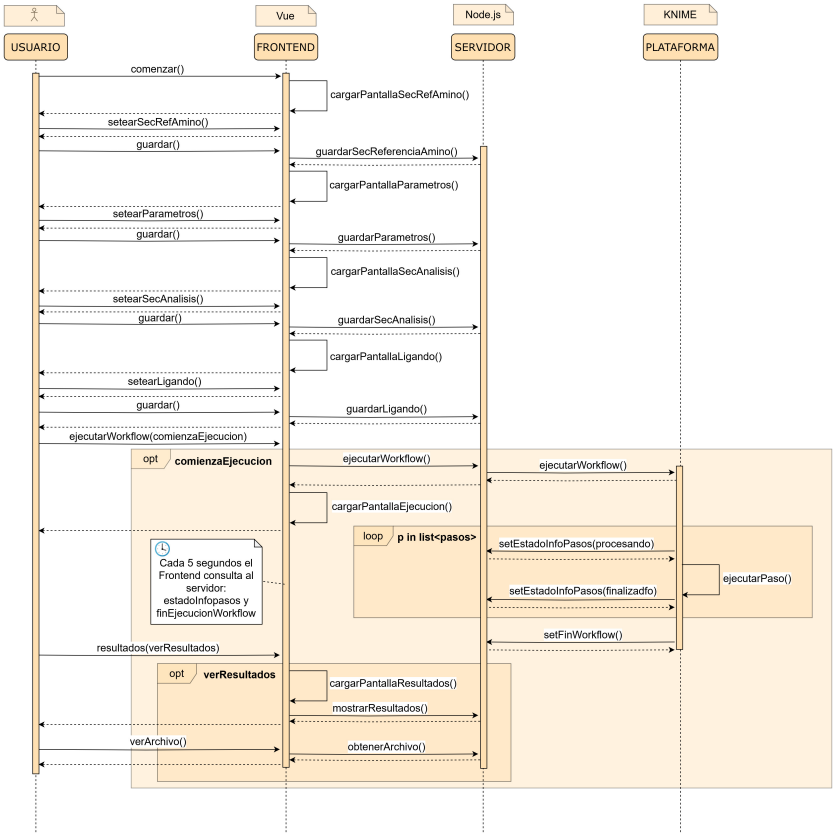

Figura 5.3: Diagrama de secuencia - Vista general

Además, en la Figura 5.4 se presenta un diagrama de secuencia que resalta las principales etapas de ejecución del *workflow* implementado en la plataforma KNIME. Este diagrama se centra en la interacción con servicios y programas externos que forman parte integral del proceso.

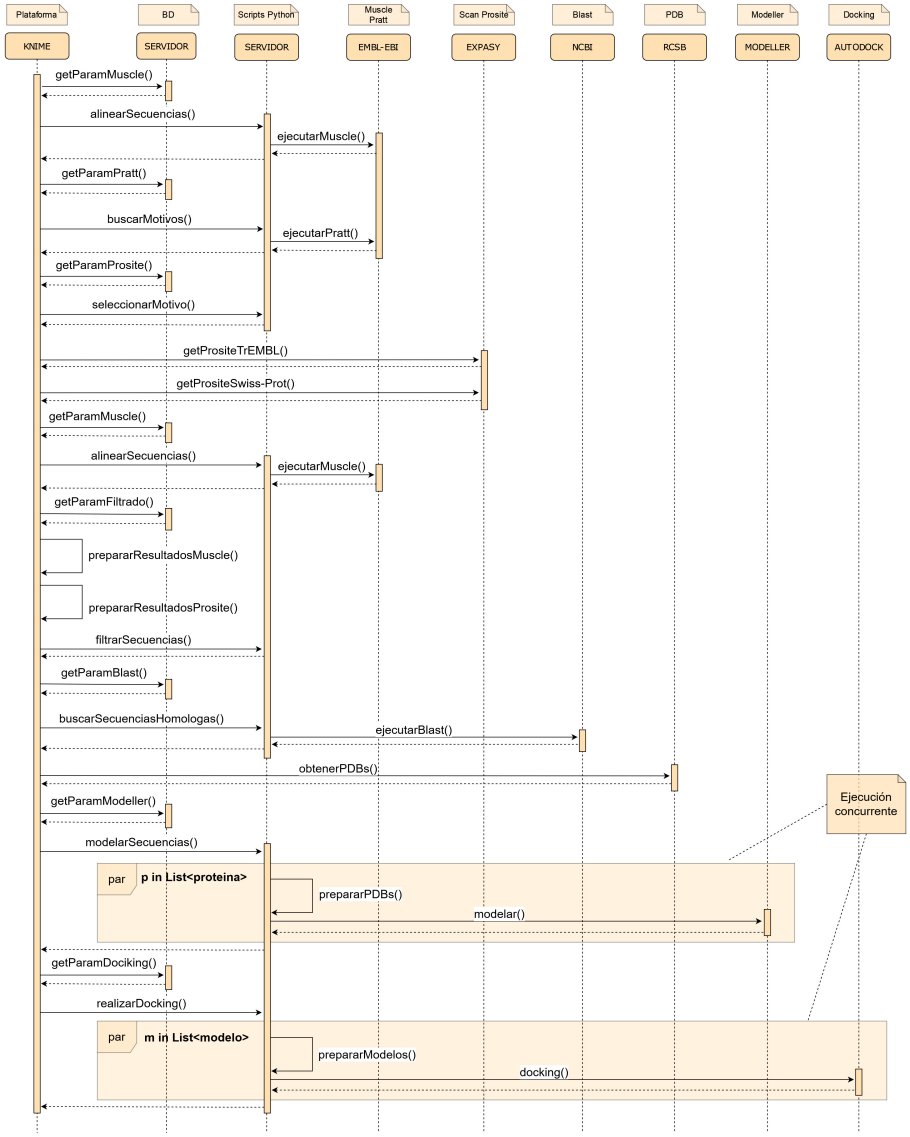

Figura 5.4: Diagrama de secuencia - Interacción *workflow* y programas/servicios externos

5.1.3. *Frontend*

El diseño del *frontend* se realiza siguiendo una estrategia donde cada faceta funcional y visual de la interfaz de usuario se descompone en componentes individuales. Este enfoque permite un desarrollo eficiente y una adaptación fluida conforme el sistema evoluciona. La Figura 5.5 ilustra con claridad el árbol de componentes, cada uno de los cuales lleva sus propias responsabilidades y funcionalidades.

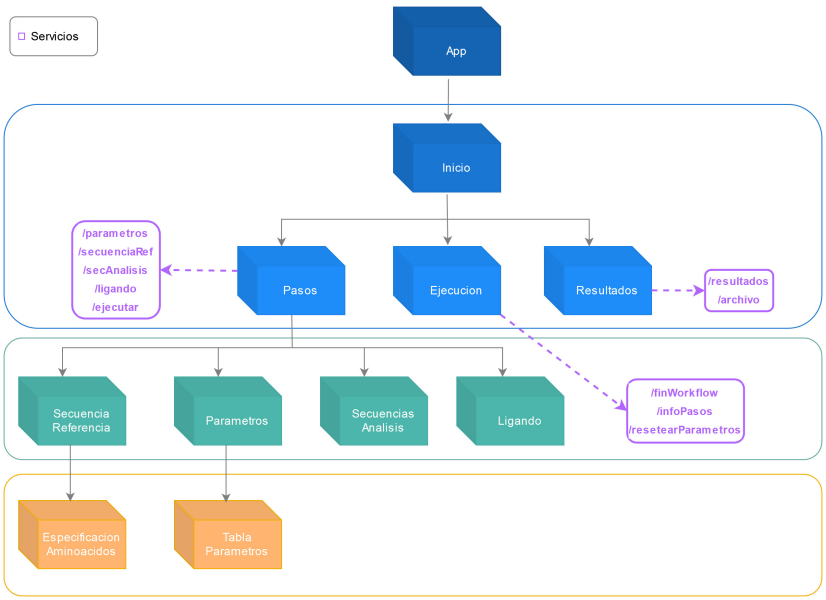

Figura 5.5: Diagrama de componentes

El punto de entrada de la aplicación se sitúa en el componente “App”, que desempeña un papel primordial como contenedor principal. Aquí, se encuentra la barra de navegación principal, el pie de página y el enrutador, que proporciona una navegación coherente entre las diversas secciones de la aplicación.

A un nivel subsiguiente, se encuentra el componente “Inicio”, que aloja una subdivisión esencial denominada “Pasos”. Este componente encapsula las diferentes etapas del proceso de carga de datos y parámetros y se fragmenta en componentes más específicos: “SecuenciaReferencia”, “Parámetros”, “SecuenciaAnalisis” y “Ligando”.

En particular, “SecuenciaReferencia” se encarga de establecer el entorno propicio para la entrada de la secuencia de referencia, ya sea mediante un archivo o introducción en texto plano. Dentro de este subcomponente, se encuentra la “Especificación de Aminoácidos”, detallando las características de los aminoácidos que se tendrán en cuenta en el análisis. Por su parte, “Parámetros” presenta una estructura de pestañas, donde cada pestaña contiene el subcomponente “TablaParametros”. Aquí, se presentan valores numéricos, umbrales y otros elementos relevantes para el análisis. Este enfoque segmentado se implementa con la finalidad de simplificar el manejo de los parámetros teniendo presente la interacción con el usuario.

“SecuenciaAnalisis”, por otro lado, es similar a “SecuenciaReferencia”, pero se enfoca en brindar la posibilidad de ingresar las secuencias específicas para el análisis, ya sea mediante archivo o texto plano. Finalmente, “Ligando” se encarga de permitir el ingreso del ligando en formato `pdb` y, posteriormente, inicia la ejecución del *workflow*.

Dos componentes adicionales, “Ejecucion” y “Resultados”, coexisten en el mismo nivel que “Pasos”. “Ejecucion” tiene el propósito de comunicar al usuario el estado de la ejecución del proceso hasta su conclusión, brindando la oportunidad de visualizar los resultados obtenidos. Precisamente, esta responsabilidad se transfiere al componente “Resultados”. Como se mencionó inicialmente, el componente App contiene el enrutador que en la estructura de la aplicación desempeña un papel fundamental para la creación de una experiencia de usuario fluida y coherente. En la Figura 5.6 se presenta un diagrama de flujo que ilustra de manera visual cómo el enrutador gestiona la navegación y las transiciones entre las diversas vistas y componentes de la aplicación.

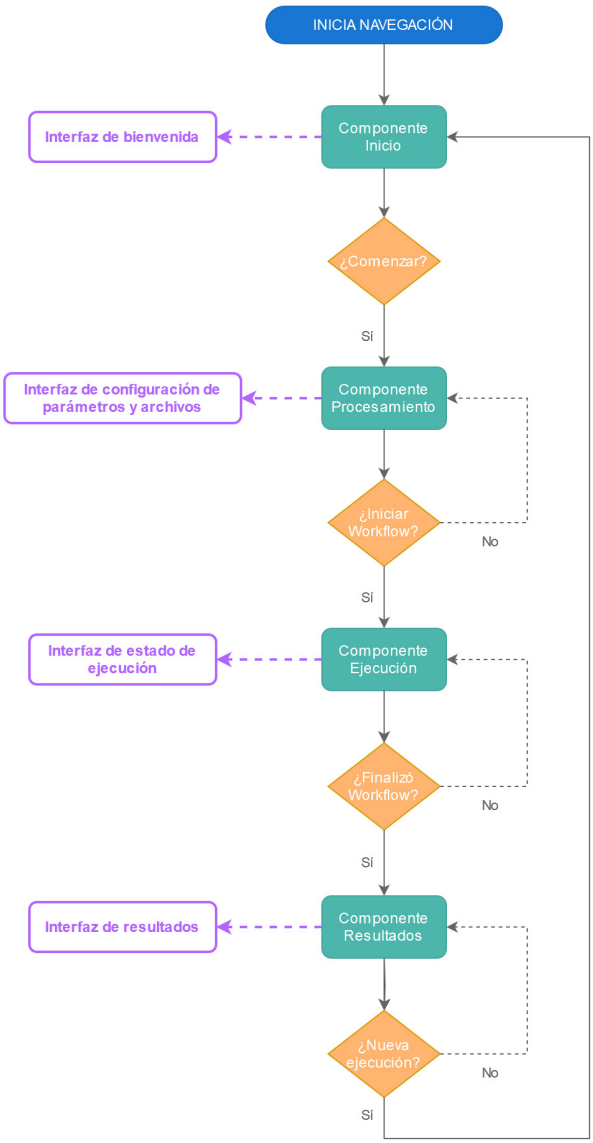

Figura 5.6: Diagrama de ruteo

Concluyendo con la descripción del diseño del *frontend*, se puede destacar que el enfoque utilizado facilita un desarrollo eficiente y una adaptación fluida a medida que el sistema evoluciona, ya que la estructura segmentada no solo simplifica la interacción con el usuario, sino que también promueve la reutilización y la mantenibilidad del código.

5.1.4. *Backend*

A continuación se detalla la API REST definida para la comunicación del *backend* con el *frontend* y la plataforma. Para ello, se utiliza la arquitectura cliente-servidor, donde el servidor se encarga de la implementación de la API. Esta API se encarga de recibir las peticiones del cliente (en este caso, el *frontend* o plataforma), procesarlas y devolver la respuesta correspondiente en formato JSON.

POST /api/parametros

Descripción: Almacena y controla los parámetros de entrada proporcionados en la interfaz web. Los datos se persisten en una base de datos para su recuperación por el *workflow* y el servidor. Además, se archivan para su uso como historial de configuración, permitiendo su consulta en la visualización de resultados.

```
REQUEST:
 {
     parametros: [
         { name: 'Muscle', value: 0, parameters: [Array] },
         { name: 'Pratt', value: 1, parameters: [Array] },
         { name: 'Prosite', value: 2, parameters: [Array] },
         { name: 'Blast', value: 3, parameters: [Array] },
         { name: 'Modeller', value: 4, parameters: [Array] },
         { name: 'Autodock', value: 5, parameters: [Array] }
       ]
 }

 RESPONSE:
 Código 200: 'Los datos se cargaron correctamente'
 Código 500: 'No se pudieron guardar los parámetros'
 Código 400: 'Se requiere un valor para el parámetro X en el
 paso Y'
```

Nota: X es un nombre de parámetro de entrada de la web, Y es un paso en el flujo de trabajo.

POST /api/resetearParametros

Descripción: Persiste los parámetros de entrada por defecto para el *workflow*, dejando valores vacíos según necesidad del usuario para el caso de prueba que desee ejecutar.

```
REQUEST:
 {
     parametros: [
         { name: 'Muscle', value: 0, parameters: [Array] },
         { name: 'Pratt', value: 1, parameters: [Array] },
         { name: 'Prosite', value: 2, parameters: [Array] },
         { name: 'Blast', value: 3, parameters: [Array] },
         { name: 'Modeller', value: 4, parameters: [Array] },
         { name: 'Autodock', value: 5, parameters: [Array] }
       ]
 }

 RESPONSE:
 Código 200: 'Los datos se cargaron correctamente'
 Código 500: 'No se pudieron guardar los parámetros'
```

POST /api/secAnalisis

Descripción: Almacena las secuencias de análisis en el sistema de archivos para ser utilizadas posteriormente por el *workflow*.

```
 REQUEST
 {
     secAnalisis: 'secuencias de análisis'
 }

 RESPONSE:
 Código 200: 'La/s secuencia/s se cargaron correctamente.'
 Código 500: 'No se pudo cargar la/s secuencia/s'
 Código 400: 'Se deben ingresar la/s secuencia/s de análisis'
```

POST /api/secuenciaRef

Descripción: Almacenamiento de la secuencia de referencia y aminoácidos de interés para el análisis de proteínas.

```
REQUEST:
 {
     secuenciaRef: 'secuencia de referencia',
     aminoaciodosRef: [
         { posicion: '1', aminoacidos: [Array], obligatorio: false },
         { posicion: '2', aminoacidos: [Array], obligatorio: false }
     ],
     cantAminosObligatorios: 0
 }

 RESPONSE:
 Código 200: 'Los datos se cargaron correctamente'
```

```
Código 500: 'No se pudo cargar la/s secuencia/s'
Código 500: 'No se pudieron guardar los aminoácidos de referencia'
```

POST /api/ligando

Descripción: Almacena el ligando a utilizar para el *docking* en un archivo con formato pdb.

```
REQUEST:
{
    ligando: 'ligando'
}

RESPONSE:
Código 200: 'Los datos se cargaron correctamente. ¿Desea
comenzar la ejecución del flujo de trabajo?'
Código 500: 'No se pudo cargar el ligando'
Código 500: 'Se debe ingresar el ligando'
```

POST /api/ejecutar

Descripción: Inicia la ejecución del *workflow* implementado con KNIME.

```
REQUEST:
{}

RESPONSE:
Código 200: 'Comenzando la ejecución del flujo de trabajo'
```

POST /api/infoPasos

Descripción: Informa el estado de cada paso a medida que avanza la ejecución del *workflow*, almacenando en la base de datos los datos recibidos.

```
REQUEST:
{
    "paso":"nombre del paso","valor":"estado"
}

RESPONSE:
Código 200
Código 500
```

GET /api/infoPasos

Descripción: Retorna un JSON con el estado de los pasos para mostrar el progreso del *workflow* en la interfaz web.

```
REQUEST:
```

```
{}

RESPONSE:
Código 200:
    {
        "paso":"Muscle","valor":"procesando",
        "paso":"Muscle","valor":"finalizado",
        // ...
    }
```

POST /api/finWorkflow

Descripción: Informa la finalización de la ejecución desde el *workflow* y almacena en la base de datos el resultado de las proteínas que clasificaron ok.

```
REQUEST:
{
    "proteinasOk": List["nombre proteina"]
}

RESPONSE:
Código 200
Código 500
```

GET /api/finWorkflow

Descripción: Retorna un valor booleano que indica si el *workflow* ha finalizado.

```
REQUEST:
{}

RESPONSE:
Código 200: Booleano
```

POST /api/guardarHistorialEjecucion

Descripción: Bajo el directorio Historial, se genera una carpeta con el nombre enviado en el *request* y se copian los resultados obtenidos recientemente al finalizar la ejecución del *workflow*. Dejando esta información disponible para futuras consultas de los resultados. Además se almacena en la base de datos la ruta generada para la ejecución guardada.

```
REQUEST:
{   nombre: 'Nombre con el que se guarda la ejecución'}

RESPONSE:
Código 200: 'Ejecución guardada exitosamente.'
Código 500: 'Error al guardar la ejecución. La ejecución
```

```
req.nombre ya existe.'
Código 500: 'Error al guardar la ejecución.'
```

GET /api/resultados

Descripción: Obtiene la estructura de archivos del sistema de archivos definido para guardar los resultados de cada paso de la ejecución del *workflow* clasificando cada nodo como archivo o carpeta. Esto se utiliza con el fin de visualizar los resultados en la web.

```
REQUEST:
{}

RESPONSE:
Código 200: Árbol de archivos
Código 500: 'No se pudieron cargar los resultados'
```

GET /api/historial

Descripción: Obtiene los resultados de ejecuciones anteriores que han sido guardados por el usuario.

```
REQUEST:
{}

RESPONSE:
Código 200: Árbol de archivos
Código 500: 'No se pudo cargar el historial de resultados'
```

GET /api/archivo

Descripción: Se obtiene el archivo correspondiente a la ruta indicada como parámetro del *request* y lo envía al *frontend*.

```
REQUEST:
{
    path: 'C:\\Servidor_Proyecto\\Archivos_workflow
    \\aminoacidosRef.txt',
    name: 'aminoacidosRef.txt'
}

RESPONSE:
Código 200: Archivo
Código 500: 'No se pudo recuperar el archivo'
```

Por otra parte, el *backend* está conformado por una base de datos que está diseñada centrándose en los pasos del *workflow* y la representación de sus ejecuciones.

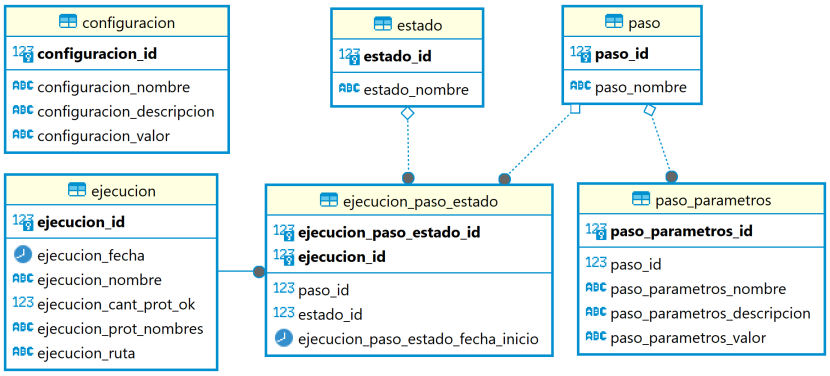

Figura 5.7: Modelo entidad relación de la base de datos

La Figura 5.7 presenta el modelo Entidad-Relación (E-R) donde se observan las entidades utilizadas en el diseño de la base de datos para representar tanto los pasos del *workflow* como sus ejecuciones. En relación a los pasos del *workflow*, se utiliza la entidad "paso", que contiene un identificador y el nombre del paso. La entidad "paso parametros" se encarga de especificar los parámetros de cada paso, incluyendo su identificación, nombre, descripción y valor.

Por otro lado, para representar las ejecuciones del *workflow*, se crea la entidad "ejecucion" con carácterísticas como identificador, fecha, nombre, cantidad de proteínas que superaron el umbral de clasificación después del *docking* molecular, y sus nombres. Además, la entidad "ejecucion paso estado" hace referencia a la ejecución de cada paso en una ejecución global, incluyendo la fecha y hora y el estado actual del proceso en el paso, que se especifica mediante la entidad "estado" cuyos valores pueden ser "PENDIENTE", "PROCESANDO", "FINALIZADO" o "ERROR".

Por último, se incluye la entidad "configuracion" con el objetivo de incorporar todas las configuraciones globales del sistema, como las rutas del *file system* y la cantidad de procesadores que se utilizan para la ejecución.

5.2. Implementación

En esta sección se describe el proceso de implementación del sistema de *software* diseñado en el marco de este proyecto. Con el fin de explicar en detalle el proceso de implementación del sistema, se describen dos etapas fundamentales: Por un lado, el *frontend*, que corresponde a la interfaz web del sistema, y por otro lado, el *backend*, que involucra tanto el servidor como el *workflow*.

5.2.1. Interfaz

La aplicación web realizada se desarrolla utilizando Vue.js (*Vue.js*, 2014), sus características fueron de gran utilidad para la implementación de un frontend funcional, responsive y estético, permitiendo interactuar de manera amigable con el usuario y los demás componentes del sistema.

Además, la incorporación de Vue.js resulta de gran ayuda para el desarrollo del *frontend* ya que cuenta con una amplia documentación y una comunidad activa que proporciona soporte y recursos para los desarrolladores.

La elección de colores para el diseño de las pantallas se basa en la paleta representada en la Figura 5.8. Se busca lograr una armonización con los colores presentes en los logotipos institucionales de la Facultad de Ingeniería y la Facultad de Química.

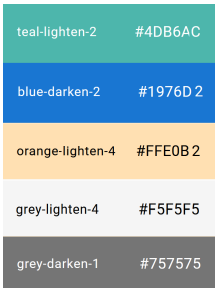

Figura 5.8: Paleta de colores

En relación al *frontend*, se presenta en la Figura 5.9 una captura de pantalla de la página principal de la aplicación, en la que se brinda una breve introducción sobre su objetivo y se ofrece la posibilidad de iniciar el procesamiento de las proteínas.

Figura 5.9: Pantalla principal

En cuanto a la segunda pantalla que se muestra luego de dar comienzo al

proceso, en la Figura 5.10 se pueden ver cuatro apartados principales: “Secuencia de referencia” , “Parámetros”, “Secuencias de análisis” y “Ligando”. En el primer apartado se requiere la carga de la secuencia de referencia ya sea mediante la subida de un archivo o ingresando texto plano. También se debe especificar los aminoácidos de interés en las posiciones correspondientes de la secuencia cargada.

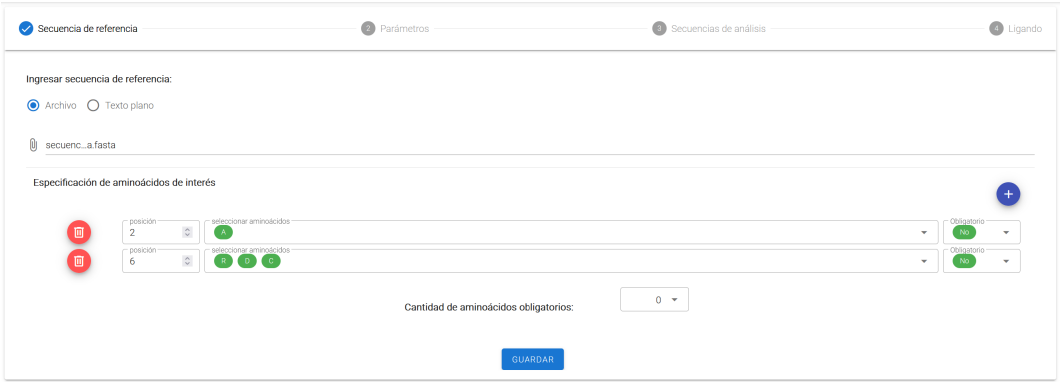

Figura 5.10: Pantalla Secuencia de referencia

En la Figura 5.11 se visualiza la solicitud de los parámetros necesarios para cada paso del proceso (MUSCLE, Pratt, PROSITE, BLAST, MODELLER y AutoDock Vina). Algunos de estos parámetros son precargados automáticamente con valores por omisión, permitiendo modificar en caso de ser necesario según la realidad que se quiera estudiar.

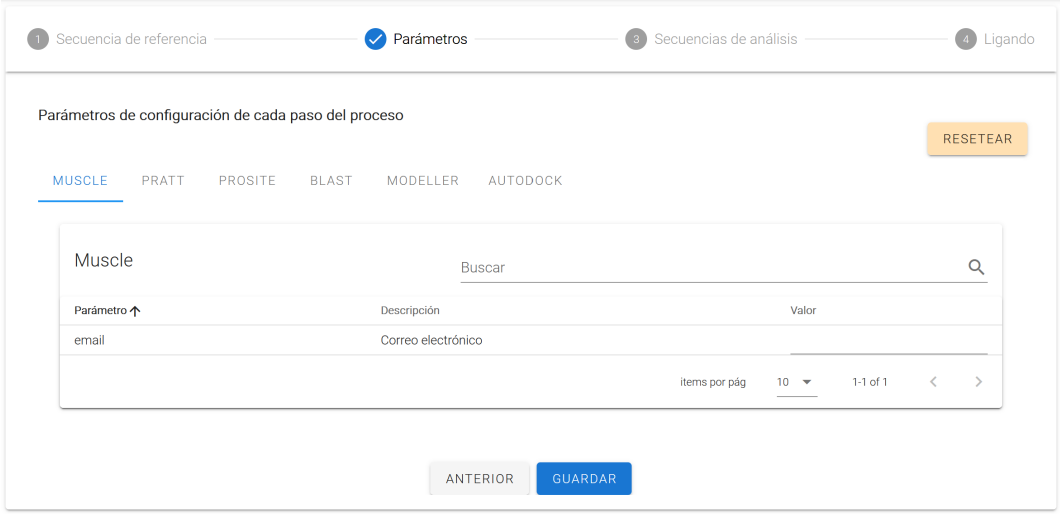

Figura 5.11: Pantalla de Especificación de parámetros

En el apartado “Secuencias de análisis” y “Ligando”, se cargan las secuencias analizadas por el usuario y el ligando para poder dar comienzo al

procesamiento. Es importante recordar que todos los archivos cargados en la web y los generados en la ejecución se almacenan en el *file system* definido: ..\Servidor_Proyecto\Archivos_workflow, en el cual la ruta raíz es configurable en la base de datos. Por otra parte, los parámetros ingresados en la web también se almacenan en la base de datos, para poder recordarlos posteriormente o reutilizarlos.

Finalmente, se muestra por un lado el estado de ejecución del *workflow* en la Figura 5.12, donde se informa el estado de cada paso al iniciar y finalizar su procesamiento, estos datos también son almacenados en la base de datos. Para presentar esta información al usuario de forma actualizada y en tiempo real, el frontend realiza un *get* del servicio /api/infopasos al servidor cada 5 segundos. Luego de finalizado el proceso, la ejecución realizada puede ser almacenada por el usuario en el *file system* del sistema, indicando el nombre con el que desea identificarla. Por otro lado, los resultados obtenidos se exponen en la pantalla "Resultados" como se aprecia en la Figura 5.13. A la izquierda se presenta la estructura de *file system*, y luego a la derecha se visualiza el archivo seleccionado, teniendo la posibilidad de descargarlo si se desea.

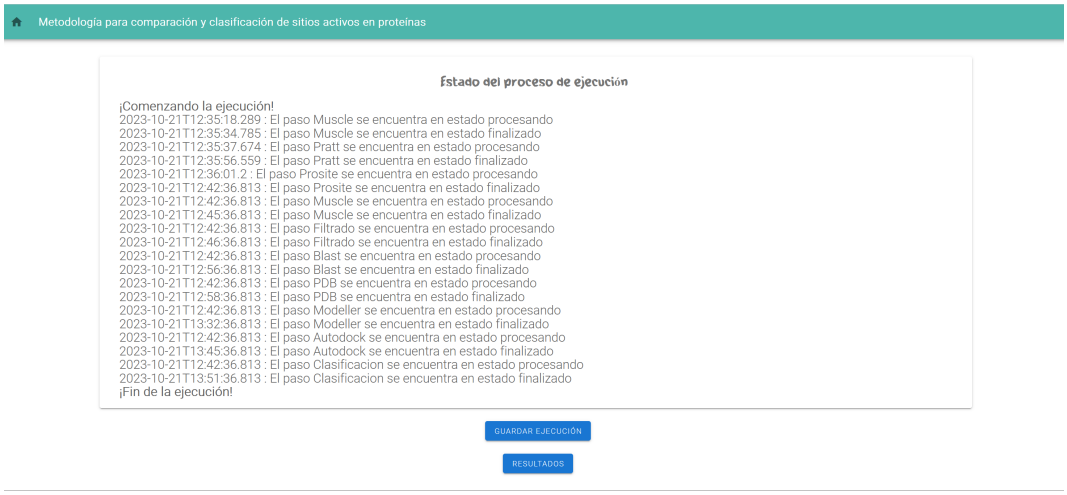

Figura 5.12: Pantalla de estado de ejecución

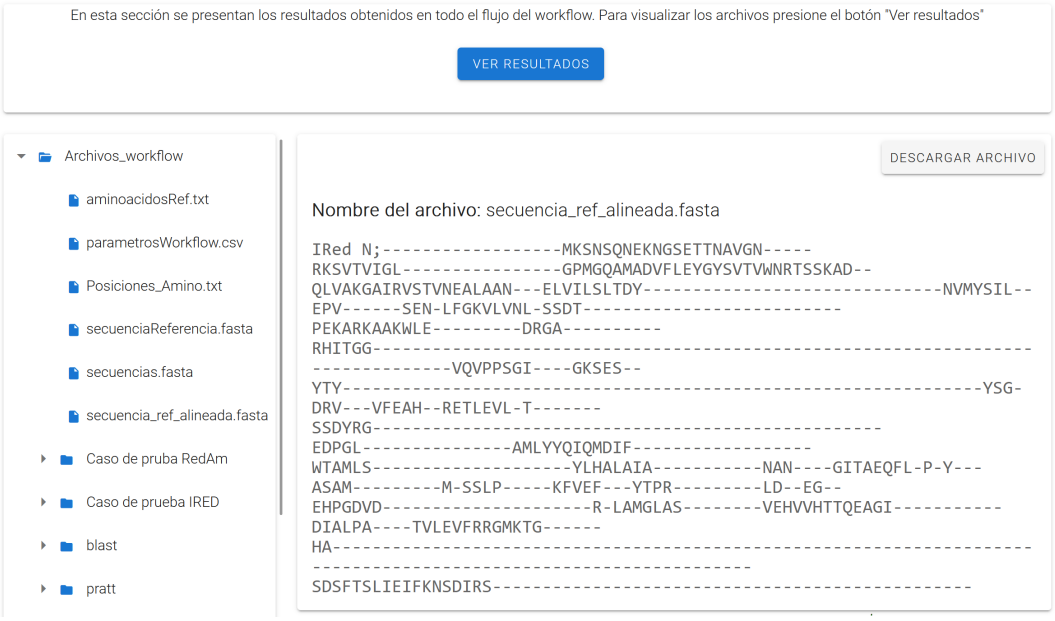

Figura 5.13: Pantalla de resultados

5.2.2. Servidor y Procesos

El servidor es implementado con Node.js (*Node.js*, 2009), ya que utiliza un modelo de E/S sin bloqueo y orientado a eventos, lo que lo hace altamente eficiente para aplicaciones en tiempo real. Además, el lenguaje utilizado es *JavaScript*, lo que facilita la creación rápida de prototipos y la iteración ágil en el desarrollo. A su vez cuenta con npm (*Node Package Manager*), que es un repositorio de paquetes de código abierto que facilita la instalación y gestión de bibliotecas, lo cual proporciona una gran cantidad de bibliotecas listas para usar. Los *scripts* creados para el procesamiento de algunos pasos del proceso fueron desarrollados con *python*.

Por otra parte, la metodología en la que está basado el *workflow* implementado se presenta en la Figura 5.14. Allí se encuentran plasmados los pasos que comprende este proceso, una vez realizada la lectura de archivos y la especificación de parámetros, comienza su ejecución. Este inicia cuando se consume el servicio "api/ejecutar" del servidor, y se procede a ejecutar en segundo plano un proceso externo invocando a KNIME con sus correspondientes parámetros, entre estos el directorio donde se encuentra almacenado el *workflow*.

Figura 5.14: Flujo del proceso

En la Figura 5.15 se muestra una representación visual del *workflow* implementado en la plataforma KNIME, que resalta la modularización de acuerdo a cada paso en la implementación de este proceso. Los pasos se presentan en las siguientes subsecciones.

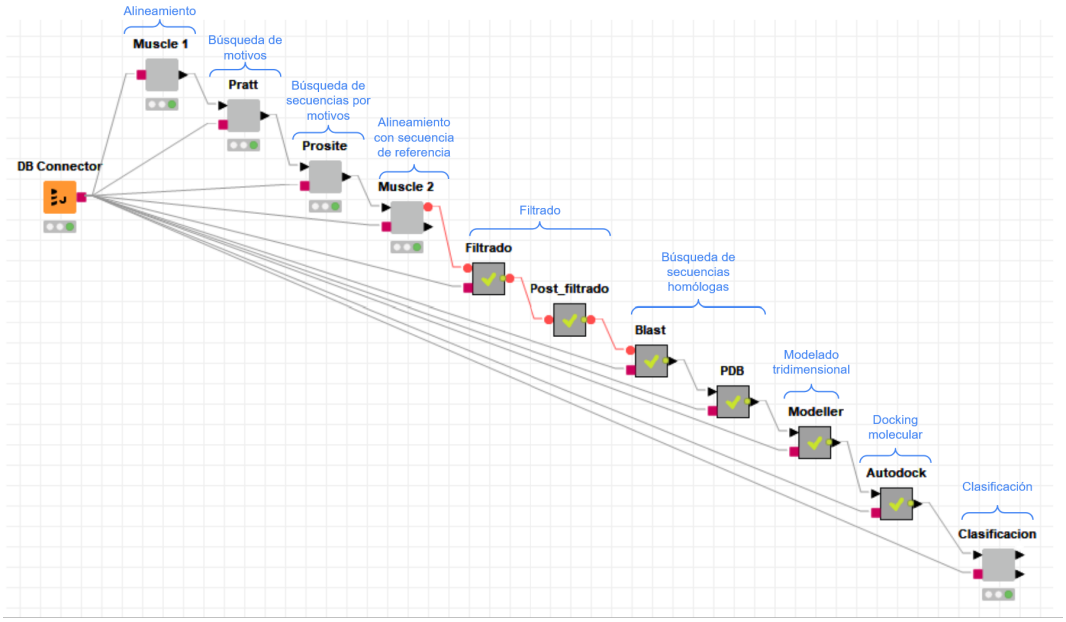

Figura 5.15: Implementación general del *workflow*

Alineamiento

En primer lugar se ejecuta MUSCLE para realizar el alineamiento de las secuencias. Como parámetros de entrada recibe las secuencias ingresadas en formato multifasta y una dirección de correo electrónico.

Búsqueda de motivos

A partir de las secuencias de entrada alineadas, se procede a buscar los motivos. Se utilizan Modelos Ocultos de Markov para buscar los parámetros ocultos a partir de parámetros observables. Para esto se utiliza el algoritmo de búsqueda que ofrece Pratt, que a partir del alineamiento múltiple retorna una lista de motivos conservados en orden, respetando una puntuación establecida. La puntuación se refiere a la asignación de un valor numérico a los motivos identificados en el proceso de búsqueda de secuencias alineadas, con el fin de evaluar su importancia y relevancia biológica. Los motivos con puntuaciones más altas se consideran más significativos y es más probable que representen patrones biológicamente relevantes. Al aplicar puntuaciones a los motivos encontrados, se establece una medida cuantitativa que permite discernir entre aquellos patrones que son más propensos a tener funciones biológicas significativas y los que podrían ser el resultado de coincidencias aleatorias. Los motivos que obtienen puntuaciones más altas son considerados como candidatos más destacados, ya que se considera que tienen más probabilidades de representar regiones conservadas con una función biológica específica.

Para la implementación el paso Pratt utiliza un *script* de Python en el que se invoca la API REST, ambos recursos ofrecidos por EMBL-EBI (Madeira y cols., 2019). Se ingresan como parámetros las secuencias alineadas en formato fasta, un correo electrónico, y el *stype* cargado con valor "*protein*". La salida del programa se presenta en formato de texto plano, y contiene los motivos encontrados.

Búsqueda de sequencias por motivos

El flujo implementado en KNIME prosigue a buscar las secuencias de proteínas que contengan el motivo obtenido con Pratt. Para ello, de la lista resultante se selecciona el primer motivo que alcance el largo mínimo introducido como parámetro de entrada, respetando el orden de aparición en el listado para obtener así el motivo con mejor puntuación. El cálculo del largo mínimo se puede ver en Apéndices 3.2. Luego se usa PROSITE con los parámetros definidos en la web para este paso. En el afán de optimizar los recursos, se consultan de forma simultánea las bases de datos UniprotKB (SwissProt y TrEMBL). Una vez obtenidas las secuencias de proteínas con PROSITE, se eliminan las que son idénticas a las secuencias de entrada, y las duplicadas (en caso de que existan). El contenido central de este paso de la implementación se puede apreciar en la Figura 5.16.

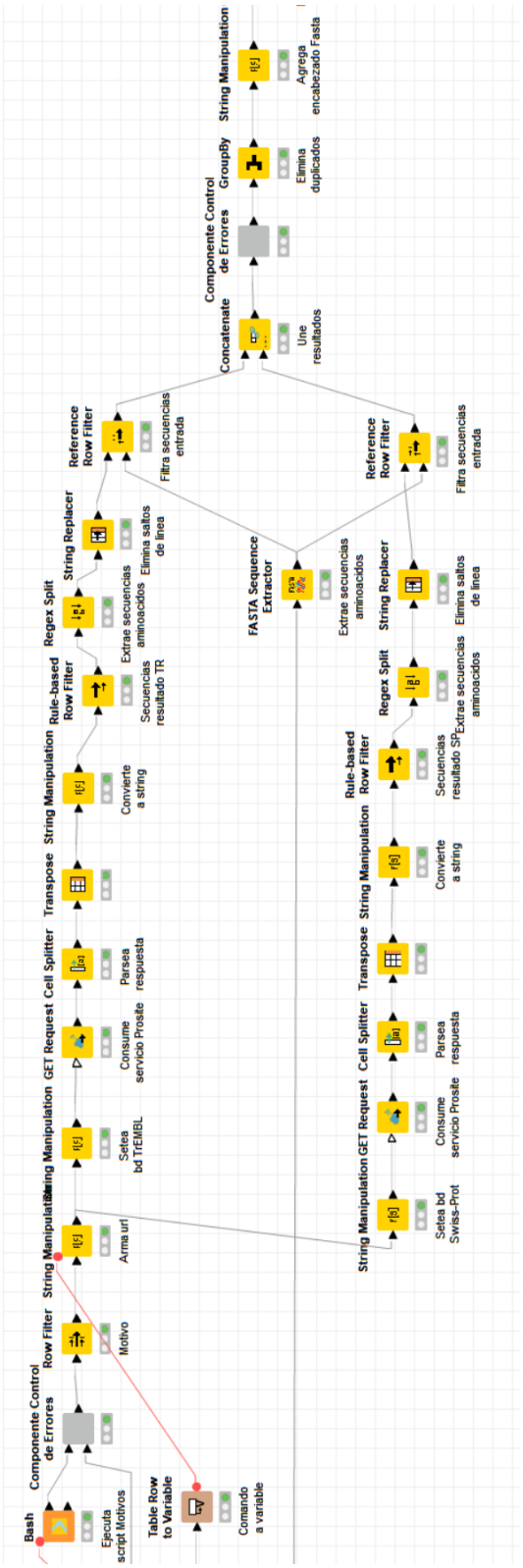

Figura 5.16: Implementación para la ejecución de PROSITE

Alineamiento con secuencia de referencia

El siguiente paso consiste en realizar nuevamente la ejecución de MUSCLE para la alineación de las secuencias obtenidas en PROSITE. Como MUSCLE admite el procesamiento de una cantidad máxima de quinientas secuencias en cada ejecución, se agrupan los registros obtenidos en archivos de cuatrocientos noventa y nueve registros, y se adiciona la secuencia de referencia.

Con el objetivo de facilitar la ejecución concurrente se desarrolla el módulo denominado "llamadaConcurrente" que aprovecha la capacidad de gestionar múltiples hilos a través de la librería ThreadPoolExecutor de Python. Este módulo implementa una estrategia que considera un array de procesos, con lo que se asegura la ejecución una única vez por proceso. Para su invocación, en este paso cada proceso se corresponde a un archivo del Prosite. De esta forma se permite establecer una configuración dinámica a nivel de base de datos de la cantidad de núcleos a ser utilizados por el módulo, y así optimizar el rendimiento de la ejecución.

La implementación realizada en KNIME para este paso, consume el servicio REST (*MUSCLE REST service*, s.f.) invocado desde un *script* de Python (*MUSCLE service Python client*, s.f.), recursos ofrecidos por EMBL-EBI, y obteniendo como salida archivos en formato fasta.

Filtrado

Una vez que se tienen los archivos con las secuencias alineadas agrupadas de a quinientas, se concatenan y se eliminan las secuencias de referencia que quedan duplicadas. A partir del archivo resultante se realiza un filtrado por aminoácidos del sitio activo. Para esto se implementa un código Python donde se toman los datos del archivo concatenado, la alineación de sus secuencias (sin la de referencia), la secuencia de referencia alineada, y las secuencias sin alinear provenientes de PROSITE.

Se cuenta además con los aminoácidos de entrada, sus posiciones en la secuencia de referencia sin alinear, cuáles de estos son obligatorios, y un umbral que indica la cantidad mínima de aminoácidos que deben estar presentes en su posición correspondiente en las secuencias. Se define como pre-condición que el umbral no puede ser menor a la cantidad de aminoácidos obligatorios. Con esta información, se calculan las posiciones correspondientes en la secuencia de referencia alineada, a partir de los aminoácidos de la secuencia de referencia sin alinear.

El algoritmo realiza una búsqueda de los aminoácidos que deben encontrarse en cada posición de las secuencias alineadas, teniendo en cuenta el resultado de la correspondencia entre las secuencias alineadas y sin alinear (salidas de MUSCLE y de PROSITE respectivamente). Se itera en el total de secuencias, y a su vez, para cada iteración se realiza la búsqueda de los aminoácidos. Mientras que no se llegue a la cantidad de obligatorios o no se alcance el umbral,

se continúa recorriendo la secuencia.

Búsqueda de secuencias homólogas

En este paso se utiliza BLAST (Protein Blast) para realizar una búsqueda preliminar por homología de un archivo multifasta con las secuencias resultado del filtrado en las bases de datos disponibles, de forma de poder obtener nuevas proteínas utilizando algoritmos de búsqueda por similitud.

Luego de ejecutar BLAST, se extrae el nombre de las proteínas, el nombre de las proteínas adicionado a cada cadena, el valor de *e-value* y la identidad. El nombre se utiliza para obtener el archivo `pdb` de la base de datos *Protein Data Bank* y utilizarlo como *template* en el modelado. Por otra parte, el nombre de la proteína adicionado con cada cadena se utiliza para contar la cantidad de cadenas presentes. Esta información es utilizada para la selección de *templates*.

El *e-value* es el resultado de un cálculo estadístico basado en la calidad de la alineación. A esto se le denomina puntuación, es la medida de similitud entre dos secuencias y se calcula a partir de la matriz de alineación. Por lo tanto, cuanto menor sea el *e-value*, mayor será la adecuación de la secuencia de consulta y la secuencia recuperada. El valor cero significa que hay una coincidencia exacta para la secuencia, no existe probabilidad de que la alineación ocurra por casualidad. Por ejemplo, un valor de 1e-3 indica que hay una probabilidad de 0.001 de que esa alineación exista en la base de datos por casualidad. Por último, el valor de la identidad hace referencia a la cantidad de aminoácidos que coinciden entre dos secuencias.

En la implementación del *workflow* se toman las dos secuencias con el *e-value* más próximo a cero, cuyo valor de identidad sea mayor o igual a treinta (valor definido por el usuario), y además, la cantidad de cadenas sea menor o igual a la cantidad de cadenas ingresada como parámetro de entrada. Posteriormente se procede a recuperar el archivo `pdb` correspondiente a cada secuencia obtenida, para lo cual se consume de un servicio web ofrecido por RCSB (*RCSB web service*, s.f.) utilizando como identificador de búsqueda el nombre de la proteína *template*. Este paso se invoca a través del *script* "llamadaConcurrente", donde cada proceso se corresponde al archivo de Blast de una proteína de la que que desea recuperar las dos proteínas homólogas, de modo de garantizar la ejecución concurrente para la cantidad de núcleos predefinidos.

Modelado tridimensional

El proceso continúa con el modelado de cada secuencia del filtrado para la que se haya obtenido al menos dos templates (BLAST + RCSB), para este propósito se utiliza como base MODELLER.

En la Figura 5.17 se presenta la parte central de la implementación en KNIME para este paso.

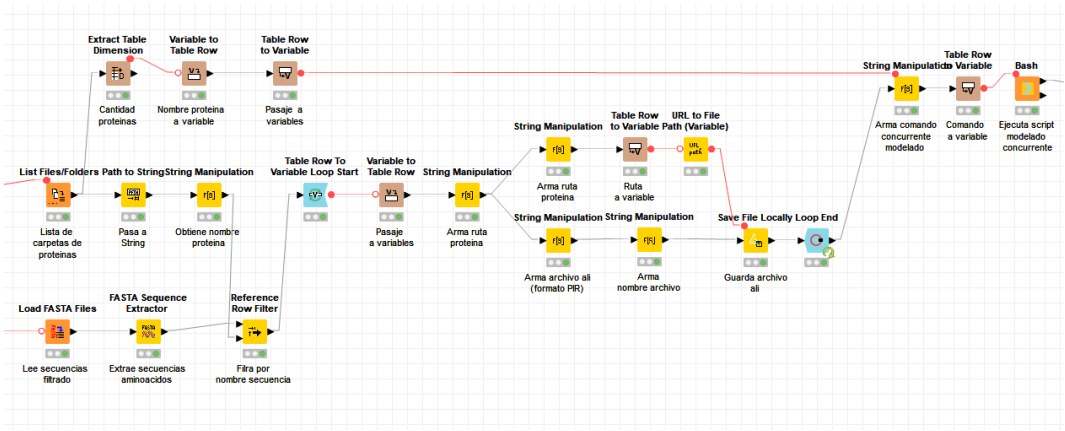

Figura 5.17: Implementación para la ejecución de MODELLER

Tal como se menciona al inicio de la sección, en la interfaz web se definen los parámetros necesarios para la ejecución de cada paso del *workflow*. Para el caso del modelado se especifica la cantidad de cadenas, parámetro que será utilizado para la creación del modelo. En este punto se implementa un *script* Python principal que contiene los pasos para generar cada modelo en base a los dos templates obtenidos con BLAST + RCSB. A continuación se presenta el pseudocódigo para facilitar una explicación más detallada.

```
ok=control_previo_largo_secuencia()
Si ok:
    ejecucion_formatear_pdb()
    ejecucion_agregar_cadenas_faltantes()
    ejecucion_separacion_por_cadenas()
    ejecucion_salign()
    ejecucion_align2d_mult()
    ejecucion_depuracion_proteinas()
    ejecucion_union_ali()
    okModelo=ejecucion_model_mult()
    Si okModelo:
            mejorModelo=ejecucion_model_energies()
            seleccion_modelo_docking(mejorModelo)
Sino:
    descartar_proteina()
```

En el pseudocódigo se puede apreciar que en primera instancia se realiza un control en relación al largo de las secuencias obtenidas en el paso del filtrado (candidatas para generar el modelo). Este control es establecido por el usuario para evitar trabajar con *templates* que no estén bien estructurados. El control realizado debe cumplir con las siguientes dos condiciones:

- $largo(secuencia) + tolerancia \geq cantResiduos(template)$
- $largo(secuencia) - tolerancia < cantResiduos(template)$

Siendo,

- largo(secuencia): función que devuelve la cantidad total de aminoácidos.
- tolerancia: calculada con respecto al parámetro de entrada que indica el porcentaje de tolerancia que se tendrá para la diferencia entre el largo de la secuencia y los residuos del *template*.
- cantResiduos(*template*): función que devuelve la cantidad de residuos diferentes que tiene el *template* sin tener en cuenta los heteroátomos.

Una vez realizado el control se tienen dos resultados posibles, cumple/no cumple. Para el caso de los *templates* y secuencias que no cumplen, se descarta la proteína y no se continúa con la ejecución del modelo. Por el contrario, cuando se cumplen las condiciones, se procede con las líneas detalladas en el pseudocódigo.

Normalización de los templates (archivos pdb), línea "ejecucion_formatear_pdb()":
Se realiza una normalización de los archivos pdb correspondientes a los *templates*. Esto es necesario para poder manipularlos de manera uniforme en los pasos posteriores.

Control de la cantidad de cadenas, línea "ejecucion_agregar_cadenas_faltantes()":
Controla la cantidad de cadenas que tiene cada *template*. Si un *template* contiene menos cadenas de las requeridas como parámetro de entrada, se procede a repetir las cadenas que ya están presentes en el pdb del *template* hasta alcanzar la cantidad deseada. En caso contrario, se pasa al siguiente punto de ejecución.

Separación por cadenas, línea "ejecucion_separacion_por_cadenas()":
Se encarga de dividir el archivo pdb de cada *template* en múltiples archivos pdb, siendo el número de archivos igual a la cantidad de cadenas indicadas como parámetro de entrada. Esta separación es necesaria para preparar los archivos que serán utilizados por el salign (Sali, 1989), siguiente línea del pseudocódigo.

Alineación con "salign", línea "ejecucion_salign()":
Es una ejecución específica de MODELLER con ajustes pertinentes para automatizar su invocación desde el *workflow*. En esta etapa, se realiza una alineación para cada par de cadenas que aparecen en los archivos pdb generados en el punto anterior correspondientes a cada *template*, teniendo en cuenta la cantidad de cadenas indicadas en el parámetro de entrada. El resultado de este paso son archivos con extensión ".ali".

Segunda alineación con .ªlign2d mult()", línea "ejecucion_align2d_mult()":

También corresponde a MODELLER y se encarga de realizar una segunda alineación, comparando múltiples estructuras. El objetivo es refinar la alineación realizada anteriormente.

Depuración de proteínas, línea "ejecucion_depuracion_proteinas()":
Después de la alineación, es necesario realizar una depuración de los archivos pdb que se normalizaron inicialmente. El propósito es eliminar las aguas y los heteroátomos que no son NADPH, identificados con nombres como "NAP", "NAD", "NDP", "TXP", "FVH" y "HOH".

Creación de un archivo de alineación múltiple, línea "ejecucion_union_ali()":
Antes de comenzar con el modelado, se genera un archivo adicional que se utilizará como entrada. Esta etapa refiere a la fusión de los archivos ali generados previamente y los pdb depurados, teniendo en cuenta ciertas características de estos últimos.

Generación de modelos, línea "ejecucion_model_mult()":
En esta etapa, se generan cinco modelos a partir de los archivos de entrada mencionados anteriormente. Se adapta la función proporcionada por MODELLER para su ejecución automática y se agrega una condición para que el modelo generado tenga en cuenta también los heteroátomos, "env.io.hetatm = True" en el *script*.

Evaluación de modelos, línea "ejecucion_model_energies()":
Se realiza un control para verificar si se generaron todos los modelos, en caso de que el resultado haya sido negativo se descartan las proteínas correspondientes a esos *templates*, de lo contrario, se procede a realizar una evaluación de los cinco modelos, seleccionando como modelo final el que tenga mejor calidad en su estructura. Se utiliza la métrica DOPE (energía de la estructura) para determinar la calidad de la estructura. Esta función es ofrecida por MODELLER y devuelve como resultado un listado con los nombres de los modelos y su valor de DOPE. Esta es modificada para obtener sólo el nombre del archivo del modelo de mejor calidad.

Selección del mejor modelo, línea "seleccion_modelo_docking(mejorModelo)":
Último paso del modelado, se realiza la selección y almacenamiento del mejor modelo, es decir, se deja en la carpeta correspondiente el mejor pdb obtenido y se descartan los demás. Este mejor modelo se guarda en la carpeta correspondiente y se utiliza como uno de los archivos de entrada necesario para el paso de *docking* molecular.

Con el propósito de paralelizar las tareas de modelado de las proteínas se utiliza el *script* "llamadaConcurrente" para la ejecución del modelado. En este caso cada proceso se mapea con una proteína a modelar. Es de interés mencionar que para la manipulación en general de los archivos en formato pdb, se

utiliza la librería Biopython (*Biopython*, 1999), la que ofrece una amplia gama de herramientas y módulos para trabajar con estructura de proteínas. Esto permite acceder a información detallada sobre átomos, residuos y cadenas en las estructuras proteicas, información de gran relevancia para el procesamiento principalmente en la etapa de modelado.

Docking molecular

Cada modelo generado en el paso anterior junto al ligando en formato pdb ingresado como parámetro de entrada, se somete al *docking* molecular. Aplicar este proceso a un modelo y ligando implica varias fases como ser:

- Preparación del ligando y el receptor mediante MGLTools donde se produce la transformación de formato pdb a pdbqt.
- Obtención de la posición resultado del paso Filtrado para el cálculo del centro del *docking* molecular.
- Estructuración de un archivo de configuración que especifique las coordenadas cartesianas del centro, la distancia en ángstroms(Å) en cada eje, las rutas del ligando y del receptor utilizados, y la ruta del archivo de salida generado por el procesamiento.
- Ejecución de AutoDock Vina, con lo que se realiza el *docking* molecular.

Para implementar este paso del *workflow* en el caso general donde de tener varios modelos disponibles, se analiza cómo maximizar la utilización de los recursos dado que no todas las etapas son paralelizables. Del análisis se desprende que la preparación del ligando solo es necesaria hacerla una vez durante todo el proceso, al igual que la obtención de las posiciones devueltas por el Filtrado, ya que vienen todas juntas en la misma respuesta. La preparación del receptor, la elaboración del archivo de configuración y la ejecución de Vina son necesarios para cada modelo, por lo que se opta por ejecución concurrente de las primeras dos fases, para lo que se utiliza el módulo "llamadaConcurrente" presentado anteriormente. Esta vez, cada proceso se corresponde al nombre de una proteína a la que se le desea aplicar el proceso de *docking* molecular. Adicionalmente, para la ejecución de Vina se utiliza el parámetro --cpu establecido en la configuracion inicial, con lo que se determina cuántos núcleos serán dedicados a cada ejecución.

A partir del análisis se elaboran dos *scripts* en Python. Uno para la obtención de la posición para el cálculo del centro, y otro para la preparación del receptor a través de MGLTools, el cálculo del centro para el *docking* y la generación de un archivo de texto plano de configuración. El paso del filtrado calcula para cada proteína la posición donde se encuentra el sitio activo en la secuencia alineada, a partir de la posición indicada en la secuencia de referencia. Para calcular el centro del *docking* se toma el átomo ingresado como parámetro de entrada y se extraen con Biopython las coordenadas desde el archivo pdb del modelo para ese átomo, dada la posición indicada por el filtrado.

Ver más detalles en Apéndice 3.

La Figura 5.18 muestra como ambos *scripts* se integran al *workflow* implementado en KNIME junto al *scripts* proporcionado por MGLTools para la preparación del ligando y al *script* de Vina.

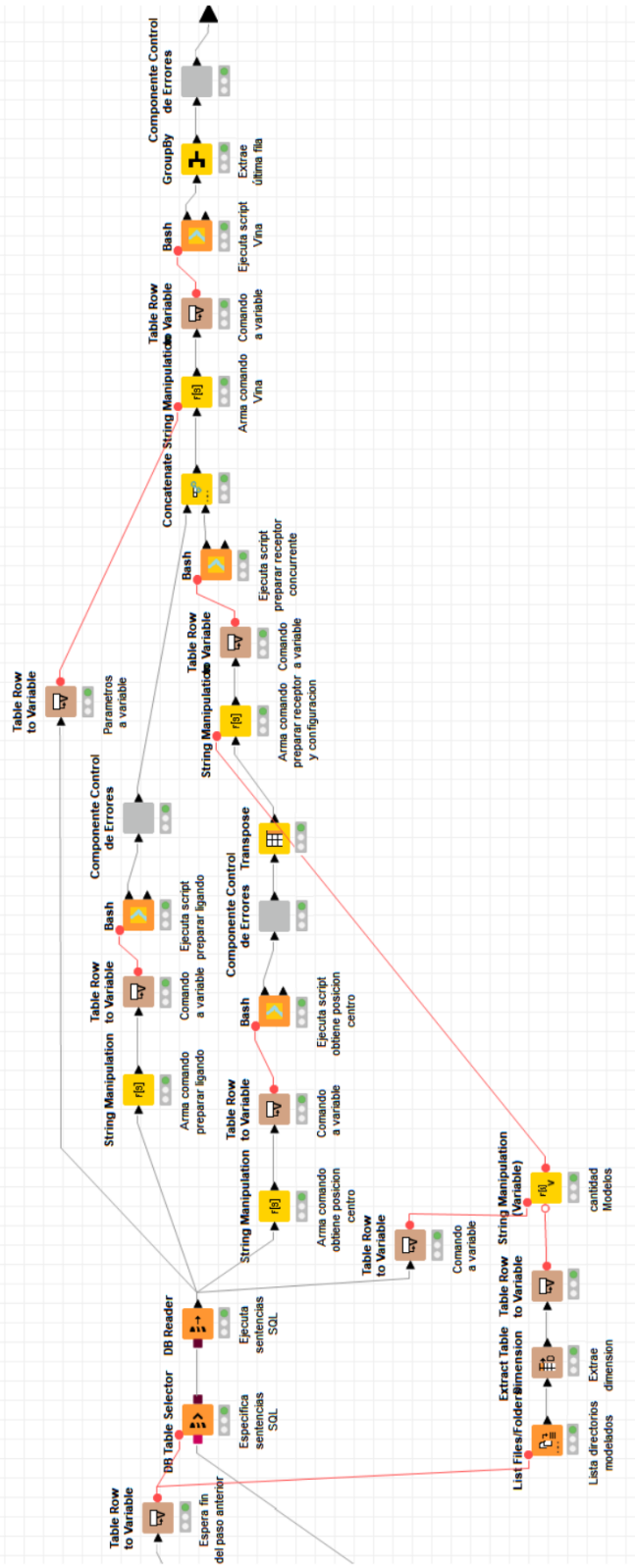

Figura 5.18: Implementación para la ejecución de AutoDock

Clasificación

Finalmente, los resultados del *docking* molecular se clasifican de acuerdo a la distancia entre el centro de la proteína modelada (ver detalles de obtención en Apéndice 3.1) y las coordenadas del átomo de nitrógeno N correspondiente a las diferentes poses del resultado del *docking* molecular . Para cada una se obtienen las coordenadas del átomo N mediante Biopython y se calcula la distancia euclideana. Si se encuentra alguna pose que cumpla que la distancia entre el átomo N y el centro de la proteína sea menor a cierto umbral predefinido como parámetro de entrada, el resultado clasifica como positivo; En caso contrario se considera no clasificado.

Para gestionar la ejecución concurrente de esta parte de la implementación se utiliza el *script* "llamadaConcurrente" donde cada proceso es el nombre de una proteína, cuyo resultado de *docking* molecular se desea clasificar. Luego, a nivel de *workflow* se crean los archivos cumplen.txt y no_cumplen.txt que contienen los nombres de las proteínas que cumplen y no cumplen con los criterios de la clasificación respectivamente.

5.3. Control y Recuperación de Errores

Se incorporan controles de errores en cada paso para detener automáticamente la ejecución. Se diseña un enfoque proactivo para manejar este tipo de situaciones generando un archivo con el error correspondiente ante la eventual ocurrencia de una falla. De esta manera se pretende identificar y comprender el error de forma oportuna, en lugar de permitir que el *workflow* continúe su ejecución y la falla sea detectada en pasos posteriores sin saber cómo se produjo. Para implementar los controles de errores se tienen en cuenta factores como la validación de los parámetros en cada paso, que el *script* de cada paso se encuentre disponible para su ejecución, y conexión lenta o pérdida total de la conexión a Internet durante la ejecución del paso del *workflow*. También se evalúa la correctitud de la respuesta generada por los servicios externos.

Se diseña un mecanismo de recuperación de errores que monitorea el estado de la respuesta a la invocación de los servicios de Muscle, Pratt y Blast. Esto lleva a la adaptación de los *scripts* de cada uno de estos pasos, de modo que sean capaces de reconocer cuando cada servicio devuelve estado "queued" para que cada uno reintente automáticamente la consulta al servicio correspondiente en un ciclo iterativo controlado. Para ello se establece un protocolo de espera de treinta segundos entre cada intento, y se define un límite de reintento de hasta tres veces antes de interrumpir el *workflow*, de esta froma se previenen posibles bucles infinitos. Este enfoque garantiza que el sistema intente nuevamente la consulta al servicio en lugar de detenerse por completo ante la detección del estado "queued".

5.4. Limpieza del Sistema de Archivos

Un aspecto crucial en la implementación de un *workflow* es garantizar la integridad y validez de los resultados, lo que requiere abordar eficazmente la gestión de archivos generados en cada ejecución. En el contexto de un *workflow* bioinformático, es común que se generen archivos intermedios y resultados parciales en cada ejecución. A medida que el flujo de trabajo se ejecuta múltiples veces, la acumulación de estos archivos puede dar lugar a una interferencia no deseada en las siguientes ejecuciones. Se desarrolla entonces un *script* en Python de limpieza de archivos y se integra al inicio del *workflow*. Este proceso de limpieza consiste en identificar y eliminar todos los archivos generados en la ejecución previa que no sean necesarios para la ejecución actual. Una estructura de carpetas organizada y jerarquizada de acuerdo a los diferentes pasos del *workflow*, resulta ser altamente efectiva para lograr tener bien identificado el directorio que almacena todos los archivos relacionados con cada paso específico.

Capítulo 6

Experimentación

En este Capítulo se exponen las dificultades encontradas durante la experimentación, así como las estrategias utlizadas para superarlas. También se analizan las pruebas realizadas y los resultados obtenidos.

6.1. Problemas Detectados y Estrategias de Abordaje

En ocasiones el servicio de PROSITE responde con *timeout*, o responde con *no hit* tanto para UniProtKB como para TrEMBL y no devuelve secuencias. Para contemplar este escenario, se decide agregar un nodo de control de errores en el *workflow* para interrumpir el flujo en ese caso, ya que sin secuencias es imposible continuar. Aunque sí permite seguir si obtuvo secuencias de al menos una de las bases de datos.

En la etapa de modelado, se detectan proteínas que generan advertencias de memoria insuficiente. Si estas advertencias son ignoradas, causan error en archivos internos de la librería de MODELLER, lo que resulta en un bucle infinito. Como primera medida se intenta aumentar el espacio de memoria virtual, pero aún así deja colgado el proceso. Se contacta al MODELLER *Caretaker* y se plantea el problema específico. La respuesta recibida indica que las dimensiones del cuadro delimitador del modelo en el espacio cartesiano son muy grandes. Se intenta atacar el problema intentando liberar el hilo de ejecución que diera el error, pero se llega a un callejón sin salida puesto que la biblioteca que administra los hilos establece que no es posible interrumpir una tarea en ejecución, siendo responsabilidad del programador asegurarse de que el programa que está siendo llamado termine de ejecutarse correctamente antes de finalizar el hilo (Brownlee, 2022). Como no es posible predecir de antemano cuanta memoria es necesaria para modelar cada proteína del conjunto, para evitar un bucle infinito, se modifica la librería de MODELLER, de modo que si no puede modelar una, continúe con la siguiente.

Si bien la utilización de la librería Biopython facilita la extracción de datos sobre la estructura de los archivos `pdb`, la librería tiene ciertas limitaciones

como que no diferencia átomos de heteroátomos al obtener los residuos. Ese parseo se realiza de forma manual.

En el contexto de este trabajo, la implementación del *workflow* bioinformático radica en la integración y coordinación de múltiples programas y servicios externos. Durante la integración de los *scripts* que invocan servicios externos se identifica un desafío en el manejo de estados de respuesta. En la implementación provista por EMBL-EBI solo se contemplaron los estados "*running*" o "*pending*" para tomar decisiones sobre el flujo de trabajo. Sin embargo, se descubrió que los estados de respuesta también incluían el estado "*queued*", lo que podría provocar errores en la ejecución. Para abordar esta variante, se realizó un ajuste en los *scripts* de invocación de servicios externos (Muscle, Pratt, y Blast), desarrollando un mecanismo de recuperación de errores cuya implementación se detalla en la Sección 5.3.

Uno de los mayores desafíos encontrados durante la implementación tuvo que ver con el no determinismo en los resultados de algunos de los programas involucrados en el *workflow*, debido a su dependencia de bases de datos externas y servicios en línea. En el contexto de este trabajo, se refleja de forma en que ciertos componentes del *workflow* implementado pueden arrojar resultados diferentes para las mismas entradas en momentos diferentes. Este fenómeno se debe a la disponibilidad y la actualización en tiempo real de las bases de datos. Este no determinismo plantea un desafío significativo, ya que puede influir en la reproducibilidad y la consistencia de los resultados obtenidos. Particularmente en el paso de obtención de homólogos se logran estructuras muy diversas, en algunos casos con etiquetas no especificadas en el formato `pdb`. Para evitar fallas en el modelado, se realiza un pre-pocesamiento de los *templates*.

Además este trabajo atraviesa un extenso proceso de experimentación en colaboración con el usuario, quien realiza sus aportes activamente mientras continúa investigando sobre el complejo proceso de modelado molecular.

6.2. Validación y Casos de Estudio

La etapa de validación tiene como objetivo principal detectar posibles defectos y asegurar que sean corregidos antes de la liberación de la versión final del sistema. Además, se busca verificar que el sistema cumpla con los objetivos y requerimientos iniciales del proyecto, así como garantizar una integración adecuada de los componentes que conforman el sistema. Para ello, se llevan a cabo pruebas tanto a nivel de *frontend* como de *backend*, evaluando la funcionalidad del diseño, la interacción de los distintos paneles y la integración con el *backend*.

En la etapa de pruebas de *backend* se realiza una división en dos partes. Por un lado, se llevan a cabo pruebas unitarias en relación a los servicios del servidor implementado, mientras que por otro lado, se realizan pruebas unitarias a nivel de pasos del *workflow*, lo que permite la detección temprana de

errores y facilita su corrección. Asimismo, se realizan pruebas de integración para analizar los resultados obtenidos de forma global.

En esta etapa resulta fundamental la participación del usuario, ya que a partir de sus conocimientos se puede contar con un conjunto de datos de entrada para la ejecución de los pasos.

Adicionalmente, se establecen instancias de validaciones con el usuario con el fin de evaluar si el sistema desarrollado cumple con los objetivos para su puesta en producción. En esencia, el objetivo es asegurar que el sistema cumpla con los requerimientos establecidos y que todos los componentes requeridos estén comprendidos en la versión final del producto.

6.2.1. Pruebas unitarias

En la fase de pruebas unitarias, se llevan a cabo evaluaciones detalladas del servidor y del *workflow*. En lo que respecta al servidor, se procede a identificar las distintas funciones que desempeñan los servicios y se ejecutan pruebas exhaustivas para evaluar sus resultados. En cuanto al *workflow*, se realizan pruebas detalladas en cada uno de los pasos identificados, definiendo con claridad los resultados esperados en cada caso.

Durante la implementación del *workflow*, se analizaron los programas utilizados por el usuario y se realizaron pruebas independientes en cada uno de ellos con el fin de generar conocimiento sobre sus características y ejecución, ya que la mayoría de ellos eran utilizados a través de una interfaz gráfica y no por línea de comandos o servicios que era lo necesario para la implementación. En esta etapa se detectaron características y configuraciones de cada uno de los programas en los que se tuvo que hacer especial foco y dedicar esfuerzos para obtener un correcto funcionamiento al momento de la integración en el *workflow*.

Algunas de las pruebas ejecutadas se detallan a continuación. Se dividen en dos etapas con el objetivo de lograr una explicación más precisa y completa.

Modelado por homología

En esta etapa se utilizan los programas MUSCLE, Pratt, PROSITE, BLAST, la base de proteínas RCSB PDB y MODELLER.

Para los pasos de obtención de motivos a partir de secuencias y obtención de secuencias a partir de motivos se probaron Pratt y PROSITE. Se realizaron pruebas teniendo un conjunto de secuencias conocidas, a partir de las cuales se obtuvieron sus motivos (patrones) y luego se realizó la búsqueda de secuencias correspondientes a estos motivos. Durante esta prueba se verificó que el resultado final contuviera las secuencias iniciales.

Se realizó la validación de MUSCLE mediante la ejecución de los *scripts* correspondientes al programa. Posteriormente, se hizo una verificación manual observando uno a uno los aminoácidos de las secuencias. Es importante destacar que se utilizó un conjunto reducido de secuencias para quitarle complejidad a la verificación manual.

En el caso de BLAST se probaron los servicios ofrecidos y se analizó el resultado presentado en un archivo XML que contenía cada uno de los *hits* encontrados. En el transcurso de esta prueba se validaron los resultados con el usuario ya que podían ahondar más en sus características.

Por último, se realizaron pruebas con MODELLER, en las que se probaron las diferentes versiones que ofrece este programa como son la versión básica, iterativa, avanzada, entre otras. Una vez tomada la decisión de la versión que sería utilizada en el *workflow* en base a los resultados obtenidos y el conocimiento aportado por el usuario, se realizaron diversas pruebas para lograr generar los modelos, alcanzando a conocer cada punto de ejecución del programa junto a sus respectivos parámetros.

Docking **molecular**

En esta etapa se utilizó el programa especializado Vina y AutoDockTools. Se realizaron pruebas en cuanto a la transformación de archivos de `pdb` a `pdbqt` utilizando MGLtools, ya que este último es el formato recibido por el programa Vina. Para esto se realizaron pruebas con archivos `pdb` descargados de la base de datos RCSB PDB con el fin de tener archivos confiables al momento de realizar la transformación. Luego se ejecutó Vina para comprobar si el formato `pdbqt` obtenido era aceptado por el programa.

Una vez finalizadas las pruebas unitarias de estas dos etapas, se posee el conocimiento y la validación de cada punto necesarios para realizar la integración en el *workflow*, y con esto proceder a realizar las verificaciones posteriores.

6.2.2. Pruebas de integración

El propósito de las pruebas de integración consiste en asegurar que las diferentes partes del sistema funcionen juntas sin problemas y cumplan con los requisitos funcionales y no funcionales especificados.

Pruebas de integración del *workflow* implementado

Como se menciona en la Sección 5.2 de implementación, el *workflow* se enfoca en la búsqueda de proteínas catalizadoras de una reacción química con un sustrato conocido. Para evaluar la eficacia del *worflow* implementado, se han establecido varios parámetros de entrada para las pruebas de integración. Estos incluyen las siguientes variables:

1. Secuencias de análisis: se trata de secuencias conocidas por el equipo técnico que han demostrado su capacidad para reaccionar con el sustrato en cuestión. Se utilizan como base para buscar otras proteínas que presenten motivos conservados en sus secuencias.

2. Secuencia de referencia: se trata de una secuencia conocida por el usuario que se utiliza como modelo para filtrar las proteínas encontradas. Las proteínas descubiertas deben contener ciertos aminoácidos esenciales para la catálisis en el sitio activo para aumentar la probabilidad de encontrar proteínas con la actividad deseada.

3. Especificación de aminoácidos de interés para el filtrado: se indica la posición y opciones de los aminoácidos que deben aparecer en cada posición de la secuencia, basándose en la secuencia de referencia mencionada anteriormente.

4. Cantidad de cadenas de proteínas a modelar: se filtran las proteínas que contiene menor o igual cantidad de cadenas que las indicadas por el usuario.

5. Identificación del centro para el *docking* molecular: se debe especificar la posición y átomo de la secuencia de referencia. Este paso es necesario para la ejecución del *docking* molecular, ya que se debe indicar el centro del sitio activo donde ocurrirá la reacción química.

6. Distancia en ángstroms para la esfera de interés donde se realizará el *docking* molecular.

7. PDB del ligando: se indica el ligando (sustrato) utilizado para la reacción química.

8. Umbral de clasificación: se especifica el umbral máximo que puede ser alcanzado por la distancia entre en centro del sitio activo y el átomo especificado. Con esta información se determina si la proteína clasifica o no.

Además, se establecen dos casos de prueba para validar el funcionamiento del *workflow*:

1. Caso de prueba IRED: este caso de prueba utiliza sustratos genéricos conocidos por el usuario y de interés específico. El objetivo es probar el modelado y el *docking* molecular.

2. Caso de prueba RedAm: este caso de prueba utiliza el sustrato necesario para formar el producto que se utilizará en el CUDIM para la detección de Alzheimer. Después de que se realiza la reacción química, el producto obtenido se somete a un tratamiento por parte de CUDIM y se utiliza como radiofármaco. Previamente a la elaboración de fármacos, se realizan estudios y pruebas clínicas para conocer qué radiofármacos se unen a ciertas enzimas del cuerpo humano para detectar diferentes enfermedades.

En conclusión, el *workflow* implementado para la búsqueda de proteínas catalizadoras se somete a pruebas de integración, en las cuales se establecen distintos parámetros de entrada en cada caso de prueba para validar su eficacia. Los datos correspondientes a estos casos de prueba se encuentran adjuntos en la sección de Apéndices 4.

En el proceso de modelado por homología, se llevan a cabo múltiples iteraciones de validación en colaboración con el usuario. Esto se debe a que, en las primeras ejecuciones del flujo completo hasta esta etapa del proceso, se detectan diversas cuestiones y desafíos en relación con el logro del resultado deseado. Estas problemáticas pueden abordar aspectos de precisión, coherencia o adecuación del modelo generado. Para perfeccionar el modelo, se realizan ajustes en los parámetros utilizados en el modelado por homología, así como la inclusión de controles adicionales. La validación continua y la retroalimentación del usuario son esenciales para mejorar la precisión del modelo.

Una vez que se ha alcanzado un consenso de los modelos obtenidos, se procede a avanzar en la ejecución de pruebas adicionales. Estas pruebas comprenden los pasos de *docking* molecular y clasificación, que son cruciales para comprender las interacciones entre moléculas a nivel tridimensional.

En el contexto del *docking* molecular, el foco principal es en la obtención del centro del sitio activo donde ocurrirá la reacción química con el sustrato.

En cuanto a la clasificación, se realizan pruebas que abarcan la variación de átomos y distancias de manera arbitraria. Este proceso garantiza que el sistema sea capaz de identificar y categorizar adecuadamente las interacciones moleculares.

6.2.3. Análisis de Resultados

En la Tabla 6.1 se presenta a modo de ejemplo el detalle de los resultados obtenidos en los pasos del *workflow*, para la ejecución de los dos casos de prueba. Es importante destacar que en ambos casos se configura una tolerancia del 25 %, largo mínimo 10 para la elección del motivo obtenido en Pratt y una distancia de 10 ángstroms para la clasificación.

Tabla 6.1: Resultados

Pasos	IRED	RedAm
Pratt	T-x(2)-G-L-G-x-M-G-x-A-x(2,3)-A-x(1,2)-L	G-x(3)-G-x(5)-D-x(3,5)-L-x(2,4)-M-x-G
PROSITE	759 secuencias encontradas	93 secuencias encontradas
Filtrado	757 secuencias filtradas	92 secuencias filtradas
BLAST+PDB	205 secuencias con templates	92 secuencias con templates
MODELLER	203 secuencias modeladas	27 secuencias modeladas
AutoDock	7 proteínas con docking	12 proteínas con docking
Clasificación	7 proteínas clasificadas	8 proteínas clasificadas

Se puede observar que en el proceso de IRED, el primer paso involucra la búsqueda de secuencias mediante PROSITE, utilizando el motivo obtenido en la etapa de Pratt, lo que da un total de 759 secuencias encontradas. El motivo seleccionado es el que Pratt devuelve con mayor *fitness*, 36.5305 y tiene largo mínimo 16. Después de aplicar el proceso de filtrado, la cantidad de secuencias se reduce a 757 secuencias filtradas. En la etapa de BLAST+PDB, se identifican 205 secuencias que cumplen las condiciones necesarias. Luego, en el paso de modelado, se generan 203 secuencias modeladas. A medida que avanza el proceso, se observa una disminución gradual en la cantidad de resultados obtenidos. En el paso del AutoDock, se realiza el *docking* molecular de 7 proteínas ya que en la posición buscada no se encuentra el átomo ingresado por parámetro de entrada en el sistema. Finalmente, en la fase de clasificación, IRED logra clasificar con éxito la totalidad de las proteínas.

Por otro lado, en el caso de RedAm, PROSITE brinda 93 secuencias encontradas con el motivo seleccionado en la etapa de Pratt que cumple con la condición de largo mínimo, el cual ocupa el segundo lugar de la lista con mayor *fitness*, con un valor de 23.0203 y largo mínimo 20. Después del proceso de filtrado, la cantidad se reduce a 92 secuencias filtradas. En la etapa de BLAST+PDB, se identifican 92 secuencias que cumplen con las condiciones necesarias. Sin embargo, en la fase de modelado, solo se generan 27 secuencias modeladas. En el proceso del AutoDock, se realiza el *docking* molecular de 12 proteínas ya que en la posición buscada no se encuentra el átomo ingresado por parámetro de entrada en el sistema. A pesar de esta disminución en la cantidad de secuencias, RedAm logra clasificar con éxito 8 proteínas en la fase de clasificación.

Es muy importante destacar que en ambos casos, el proceso inicia con una cantidad considerable de secuencias y culmina con una cantidad notablemente menor. Esto sugiere un proceso de refinamiento y selección durante las etapas intermedias del proceso de análisis y modelado. Además, se puede notar la diferencia entre la cantidad de secuencias con las que parte cada caso, siendo un número mayor el del caso IRED. Esto sucede por las características del motivo con el que se generó la búsqueda en PROSITE, es decir, que tan específico es

el motivo, dado que en RedAm posee más nivel de detalle que en IRED.

Adicionalmente, en el transcurso de la implementación del *workflow*, se llevan a cabo diversas pruebas de rendimiento con el propósito de evaluar el tiempo de ejecución de los distintos pasos y optimizarlo en la medida de lo posible.

En la tabla 6.2 se presentan los tiempos de ejecución correspondientes a dos casos distintos. En el primer caso, se ejecuta el *workflow* para los casos de IRED y RedAm con la utilización de un único núcleo, lo que simula un procesamiento secuencial. En el segundo caso, se ejecuta el flujo completo, también para IRED y RedAm, pero con la configuración de ocho núcleos, lo que implica la apertura de ocho hilos en paralelo para el procesamiento. Es importante señalar que la configuración de la cantidad de núcleos es un parámetro que puede ajustarse en la base de datos.

El propósito de esta comparación es evaluar los tiempos de ejecución y determinar si la implementación de la concurrencia ha tenido un impacto significativo en el rendimiento del *workflow*.

Tabla 6.2: Tiempos de ejecución

Tipo de ejecución	**Tiempo de ejecución IRED**	**Tiempo de ejecución RedAm**
Secuencial	17 horas, 9 minutos y 6 segundos	4 horas, 20 minutos y 21 segundos
Concurrente	8 horas, 6 minutos y 49 segundos	1 hora, 35 minutos y 40 segundos

Al analizar la tabla, se puede observar que el tiempo de ejecución en el modo concurrente representa aproximadamente una tercera parte del tiempo requerido en el modo secuencial. Este resultado demuestra de manera concluyente que la adopción de la concurrencia ha mejorado sustancialmente el rendimiento del proceso, lo que podría tener implicaciones significativas en cuanto a la eficiencia y la velocidad de ejecución de las tareas relacionadas con IRED y RedAm.

Se presenta una gráfica, por cada caso de prueba, Gráfica 6.1 y Gráfica 6.2, en las cuales se utilizó una escala logarítmica, lo que permite una visualización más efectiva de relaciones y patrones que podrían pasar desapercibidos en una representación lineal convencional. En lugar de representar los valores de manera proporcional, la escala logarítmica amplía la percepción de detalles en la región de los valores más pequeños.

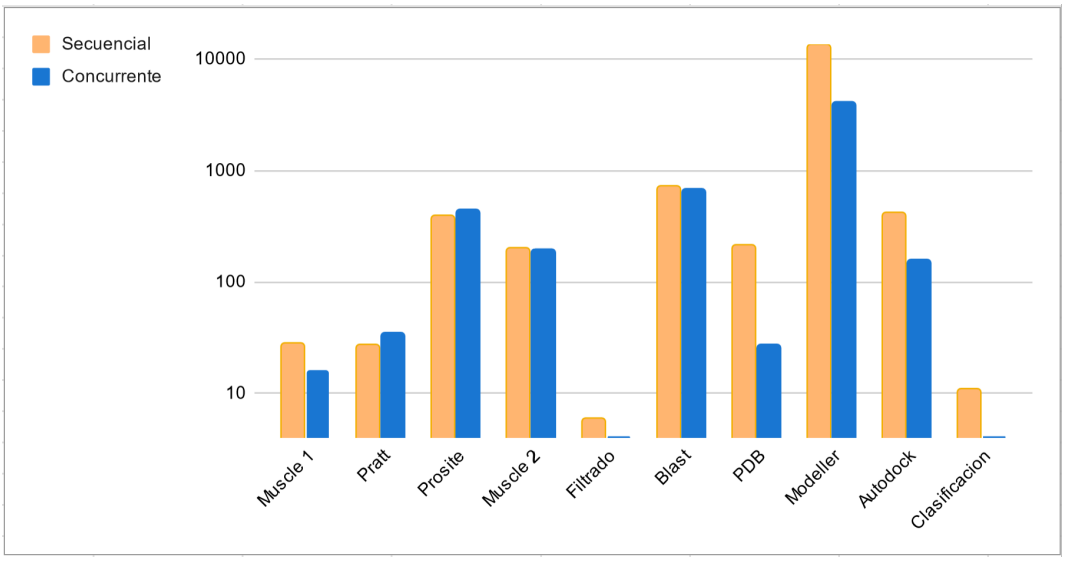

Figura 6.1: Tiempo secuencial vs tiempo concurrente - RedAm

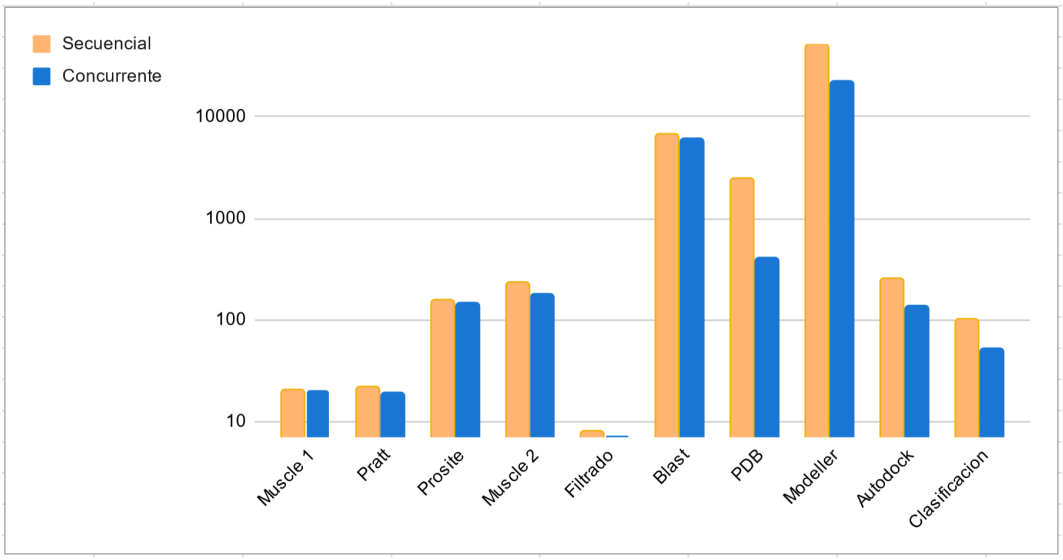

Figura 6.2: Tiempo secuencial vs tiempo concurrente - IRED

Es importante destacar que existe una diferencia significativa entre el tiempo de ejecución del paso del Modeller en su versión secuencial, que para el caso de prueba RedAm tarda 13.585 segundos, y su versión concurrente, que se reduce a 4.082 segundos. Al igual en el caso de IRED, su versión secuencial tarda 51.737 segundos, y su versión concurrente se reduce a 22.183 segundos. Sin embargo, en la gráfica con escala logarítmica, esta diferencia no es tan evidente.

Capítulo 7

Conclusiones y Trabajo Futuro

El trabajo realizado requirió la exploración de un dominio complejo antes de poder abordar el análisis de la solución, lo que requirió adquirir conocimientos sobre ADN, genes y proteínas, la forma en que los genes codifican proteínas y sus estructuras, modelado por secuencias y tridimensional por homología, y *docking* molecular de las proteínas. En cuanto al modelado por homología, se analizaron los procesos claves necesarios para la búsqueda de secuencias candidatas. Algunos de estos procesos son el alineamiento de secuencias, búsqueda de motivos a partir de secuencias, búsqueda de secuencias a partir de motivos y modelado propiamente dicho. En lo que refiere al *docking* molecular, luego de su ejecución, fue necesario realizar una clasificación evaluando el acoplamiento del sustrato y la proteína en su sitio activo. Posteriormente, con la selección de las secuencias candidatas obtenidas como resultado, el Centro de Estudios Interdisciplinarios de Biodiversidad Orientado a aplicaciones en Salud - Udelar (CEIBOS) puede verificar si las enzimas clasificadas son aptas para utilizarse en la síntesis del fármaco utilizado por el CUDIM, en la elaboración del fármaco para la detección del Alzheimer. Se trabaja en el laboratorio para realizar las pruebas experimentales que confirmen las características deseadas en cuanto a la velocidad y actividad de la reacción química.

Entonces, una vez superada la curva de aprendizaje del dominio, se diseñó e implementó el *workflow*, obteniendo como resultado final un producto que cuenta con parámetros de entrada que lo hacen flexible, que involucra la ejecución de *scripts*, el consumo de servicios y maneja diferentes tipos de formatos de archivos que son transformados a lo largo de su ejecución y son transparentes para el usuario. A su vez, guarda los resultados intermedios obtenidos, para poner a disposición del usuario en caso de que se desee analizar o consultar algún punto en particular del proceso.

Contemplando las etapas de soporte, mantenimiento y evolución del producto, se realizó una modularización del flujo para su mejor comprensión, conservando en cada etapa solo las entradas necesarias.

El tiempo de procesamiento y obtención de resultados se consideró relevante para el equipo de estudiantes, es por esto que durante la etapa de diseño

y desarrollo se buscó paralelizar la ejecución siempre que fuera posible, considerando en todo momento que los datos de cada etapa estuvieran disponibles para poder ejecutar la siguiente, evitando así la introducción de errores.

Haciendo foco en los beneficios del *workflow*, es importante destacar que logra minimizar los tiempos de dedicación del usuario, automatizando los procesos y disminuyendo en gran medida la cantidad de tareas operativas. Es decir, canalizando su dedicación a las pruebas experimentales en el laboratorio, a partir de los resultados tentativos obtenidos, y así tener a corto plazo resultados más certeros y comprobados.

También reduce en gran medida la curva de aprendizaje e investigación del usuario que no conoce las herramientas que deberían utilizarse para la ejecución de cada paso, ya que con el *workflow* implementado solo es necesario indicar un conjunto de archivos, y parámetros de entrada para que internamente se realice el proceso, y se obtenga el resultado final. A su vez, desde el punto de vista de infraestructura la instalación de cada uno de los programas utilizados se realiza una única vez en la máquina destinada para esta tarea, se ofrece una solución integrada que funciona como servidor para poder conectarse desde cualquier equipo que se encuentre en la misma red. Lo único que deberá realizarse, es levantar el servidor para dejar accesible este servicio, con el *backend* y *frontend* desarrollado.

En la comunidad científica hay muy poca información sobre cuáles son las enzimas que pueden servir para fabricar algún fármaco para la detección o el tratamiento de enfermedades, es decir, no se conoce cuales son las secuencias que realizan la reacción química deseada. Entonces, el equipo de estudiantes entiende que la búsqueda de enzimas candidatas realizada en el presente proyecto, sería un aporte a la industria farmacéutica, ya que realiza una depuración de un gran volúmen de información, con los desafíos que esto implica. Pues en los datos no se especifican de forma exhaustiva las propiedades y características de cada secuencia, y mucho menos existen modelos tridimensionales para todas ellas.

En relación a la fase de clasificación de los resultados del *docking* molecular, en el presente proyecto se llevó a cabo una evaluación que tuvo en cuenta la distancia existente entre el sustrato y el sitio activo de la proteína después de la reacción química. No obstante, se sugiere considerar la posibilidad de incluir en futuros trabajos una dimensión adicional para la clasificación, tal como la energía de unión, entre otros aspectos relevantes que puedan complementar la valoración de los resultados obtenidos. Se podrían hacer además dinámicas moleculares cortas, que logren mejorar la calidad del resultado del *docking* molecular y realizar cálculos de energía de unión más complejos que los empleados por AutoDock Vina.

En cuanto a la implementación del *frontend* proporcionado, sería posible considerar la adición de algunas funcionalidades que puedan mejorar la expe-

riencia del usuario. Por ejemplo, se podría incluir una barra de progreso que muestre el avance del *workflow*, así como también un visualizador para poder presentar el modelo tridimensional de la proteína, antes y después del *docking* molecular.

Analizando el trabajo realizado se visualiza que este puede ser separado en dos partes, por un lado la búsqueda de secuencias a partir de motivos, alineamiento y filtrado, y por otro el modelado por homología y *docking* molecular dirigido a las reacciones químicas con ciertos sustratos. Como la primera parte lo que se realiza es análisis general, este podría incorporarse en otros procedimientos de interés que considere el usuario.

Referencias

AlphaFold. (2018). DeepMind de Alphabet/Google. Descargado de https://www.deepmind.com/research/highlighted-research/alphafold

Altschul, S. (1997, septiembre). Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, *25*(17), 3389–3402. Descargado 2023-04-15, de https://academic.oup.com/nar/article-lookup/doi/10.1093/nar/25.17.3389 doi: 10.1093/nar/25.17.3389

AutoDock. (1989). Scripps Research. Descargado de https://autodock.scripps.edu/

Berg, J. M., Tymoczko, J. L., y Stryer, L. (2002). *Biochemistry* (5th ed.).

Berthold, M. R., Cebron, N., Dill, F., Gabriel, T. R., Meinl, T., Ohl, P., ... De, U.-K. (2009, junio). KNIME – The Konstanz Information Miner. , *11*(1).

Biopython. (1999). Community project. Descargado de https://biopython.org/

BLAST-Help. Developer info. (s.f.). Descargado de https://blast.ncbi.nlm.nih.gov/doc/blast-help/developerinfo.html#developerinfo

BLAST-Help. Download. (s.f.). Descargado de https://blast.ncbi.nlm.nih.gov/doc/blast-help/downloadblastdata.html#downloadblastdata

Brownlee, J. (2022, enero). *ThreadPoolExecutor in Python: The Complete Guide.* Descargado de https://superfastpython.com/threadpoolexecutor-in-python/#How_to_Cancel_Futures

Clustal Omega. (2011). EMBL-EBI. Descargado de https://www.ebi.ac.uk/Tools/msa/clustalo/

Clustal Omega Help and Documentation. (s.f.). Descargado de https://www.ebi.ac.uk/seqdb/confluence/display/JDSAT/Clustal+Omega+Help+and+Documentation

Edgar, R. C. (2004a). MUSCLE: a multiple sequence alignment method with reduced time and space complexity. *BMC Bioinformatics*, *5*(1), 113. Descargado 2023-10-17, de http://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-5-113 doi: 10.1186/1471-2105-5-113

Edgar, R. C. (2004b, marzo). MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Research*, *32*(5), 1792–1797. Descargado 2023-10-17, de https://academic.oup.com/nar/article-lookup/doi/10.1093/nar/gkh340 doi: 10.1093/nar/gkh340

Eisenberg, D., Lüthy, R., y Bowie, J. U. (1997). VERIFY3D: Assessment

of protein models with three-dimensional profiles. En *Macromolecular Crystallography Part B* (Vol. 277, pp. 396–404). Academic Press. Descargado de https://www.sciencedirect.com/science/article/pii/S0076687997770228 doi: https://doi.org/10.1016/S0076-6879(97)77022-8

Evans, R., O'Neill, M., Pritzel, A., Antropova, N., Senior, A., Green, T., ... Hassabis, D. (2021, octubre). *Protein complex prediction with AlphaFold-Multimer* (preprint). Bioinformatics. Descargado 2023-04-15, de http://biorxiv.org/lookup/doi/10.1101/2021.10.04.463034 doi: 10.1101/2021.10.04.463034

FAMSA. (2016). Utrecht University. Descargado de https://bioinformaticshome.com/tools/msa/descriptions/FAMSA.html

Feinstein, W. P., y Brylinski, M. (2015, diciembre). Calculating an optimal box size for ligand docking and virtual screening against experimental and predicted binding pockets. *Journal of Cheminformatics*, *7*(1), 18. Descargado 2023-07-27, de https://jcheminf.biomedcentral.com/articles/10.1186/s13321-015-0067-5 doi: 10.1186/s13321-015-0067-5

Ginalski, K., Elofsson, A., Fischer, D., y Rychlewski, L. (2003, mayo). 3D-Jury: a simple approach to improve protein structure predictions. *Bioinformatics*, *19*(8), 1015–1018. Descargado 2023-06-24, de https://academic.oup.com/bioinformatics/article/19/8/1015/235399 doi: 10.1093/bioinformatics/btg124

Google Cloud. (2008, abril). Google. Descargado de https://cloud.google.com

Gupta, A., y Zhou, H.-X. (2021, septiembre). Machine Learning-Enabled Pipeline for Large-Scale Virtual Drug Screening. *Journal of Chemical Information and Modeling*, *61*(9), 4236–4244. Descargado 2023-04-15, de https://pubs.acs.org/doi/10.1021/acs.jcim.1c00710 doi: 10.1021/acs.jcim.1c00710

Hollingsworth, S. A., y Karplus, P. A. (2010, octubre). A fresh look at the Ramachandran plot and the occurrence of standard structures in proteins. *BioMolecular Concepts*, *1*(3-4), 271–283. Descargado 2023-07-27, de https://www.degruyter.com/document/doi/10.1515/bmc.2010.022/html doi: 10.1515/bmc.2010.022

Humphrey, W., Dalke, A., y Schulten, K. (1996). Vmd - visual molecular dynamics. *J. Molec. Graphics*, *14*, 33-38.

ICM-Pro. (s.f.). Molsoft, L.L.C. (http://www.molsoft.com/)

Jones, G., Willett, P., Glen, R. C., Leach, A. R., y Taylor, R. (1997). Development and validation of a genetic algorithm for flexible docking11edited by f. e. cohen. *Journal of Molecular Biology*, *267*(3), 727-748. Descargado de https://www.sciencedirect.com/science/article/pii/S0022283696908979 doi: https://doi.org/10.1006/jmbi.1996.0897

Land, H., y Humble, M. S. (2018). YASARA: A Tool to Obtain Structural Guidance in Biocatalytic Investigations. En U. T. Bornscheuer y M. Höhne (Eds.), *Protein Engineering* (Vol. 1685, pp. 43–67). New York, NY: Springer New York. Descargado 2023-08-13, de http://link.springer

.com/10.1007/978-1-4939-7366-8_4 (Series Title: Methods in Molecular Biology) doi: 10.1007/978-1-4939-7366-8_4

Leipzig, J. (2016, marzo). A review of bioinformatic pipeline frameworks. *Briefings in Bioinformatics*. Descargado 2023-04-14, de https://academic.oup.com/bib/article-lookup/doi/10.1093/bib/bbw020 doi: 10.1093/bib/bbw020

Lenz, M., Fademrecht, S., Sharma, M., Pleiss, J., Grogan, G., y Nestl, B. M. (2018, abril). New imine-reducing enzymes from *B* -hydroxyacid dehydrogenases by single amino acid substitutions. *Protein Engineering, Design and Selection*, *31*(4), 109–120. Descargado 2023-06-29, de https://academic.oup.com/peds/article/31/4/109/4992468 doi: 10.1093/protein/gzy006

LigandScout Extensions for the KNIME Workbench. (2005). Inte:Ligand GmbH. Descargado de https://hub.knime.com/inteligand/extensions/com.inteligand.knime/latest

Madeira, F., Park, Y. M., Lee, J., Buso, N., Gur, T., Madhusoodanan, N., ... Lopez, R. (2019, julio). The EMBL-EBI search and sequence analysis tools APIs in 2019. *Nucleic acids research*, *47*(W1), W636–W641. Descargado de https://europepmc.org/articles/PMC6602479 doi: 10.1093/nar/gkz268

Mangas-Sanchez, J., France, S. P., Montgomery, S. L., Aleku, G. A., Man, H., Sharma, M., ... Turner, N. J. (2017). Imine reductases (IREDs). *Current Opinion in Chemical Biology*, *37*, 19–25. Descargado de https://www.sciencedirect.com/science/article/pii/S1367593116301880 doi: https://doi.org/10.1016/j.cbpa.2016.11.022

MEGAX-Help. (s.f.). Descargado de https://www.megasoftware.net/web_help_11/index.htm#t=Part_I_Getting_Started%2FA_Walk_Through_MEGA%2FAligning_Sequences.htm

MEGAX-Help - Running in Command-Line Mode. (s.f.). Descargado de https://www.megasoftware.net/web_help_10/Running_in_Command-Line_Mode.htm

Molecular operating environment (moe). (2022). Chemical Computing Group ULC. 910-1010 Sherbrooke St. W., Montreal, QC H3A 2R7, Canada. (Version 2022.02)

Morris, G. M., Huey, R., Lindstrom, W., Sanner, M. F., Belew, R. K., Goodsell, D. S., y Olson, A. J. (2009, diciembre). AutoDock4 and AutoDockTools4: Automated docking with selective receptor flexibility. *Journal of Computational Chemistry*, *30*(16), 2785–2791. Descargado 2023-07-26, de https://onlinelibrary.wiley.com/doi/10.1002/jcc.21256 doi: 10.1002/jcc.21256

MUSCLE. (2004). EMBL-EBI. Descargado de https://www.ebi.ac.uk/Tools/msa/muscle/

MUSCLE REST service. (s.f.). EMBL-EBI. Descargado de https://www.ebi.ac.uk/Tools/services/rest/muscle

MUSCLE service Python client. (s.f.). Descargado de https://raw.githubusercontent.com/ebi-wp/webservice-clients/master/python/muscle.py

Node.js. (2009, mayo). Node.js Developers Joyent. Descargado de https://nodejs.org/es/

O'Boyle, N. M., Banck, M., James, C. A., Morley, C., Vandermeersch, T., y Hutchison, G. R. (2011, diciembre). Open Babel: An open chemical toolbox. *Journal of Cheminformatics*, *3*(1), 33. Descargado 2023-08-12, de https://jcheminf.biomedcentral.com/articles/10.1186/1758-2946-3-33 doi: 10.1186/1758-2946-3-33

O'Boyle, N. M., Vandermeersch, T., Flynn, C. J., Maguire, A. R., y Hutchison, G. R. (2011, diciembre). Confab - Systematic generation of diverse low-energy conformers. *Journal of Cheminformatics*, *3*(1), 8. Descargado 2023-08-20, de https://jcheminf.biomedcentral.com/articles/10.1186/1758-2946-3-8 doi: 10.1186/1758-2946-3-8

Pearson, W. R. (2003, diciembre). *FASTA documentation.* Descargado de http://nebc.nerc.ac.uk/bioinformatics/documentation/fasta/fasta3x.doc

Pearson, W. R., y Lipman, D. J. (s.f.). *FASTA.* Descargado de https://www.ebi.ac.uk/Tools/sss/fasta/

PENTAHO BI Suite. (2004). Hitachi Vantara. Descargado de https://www.hitachivantara.com

Pons, J.-L., y Labesse, G. (2009, julio). @TOME-2: a new pipeline for comparative modeling of protein-ligand complexes. *Nucleic Acids Research*, *37*(Web Server), W485–W491. Descargado 2023-04-15, de https://academic.oup.com/nar/article-lookup/doi/10.1093/nar/gkp368 doi: 10.1093/nar/gkp368

Pratt. (2002). EMBL-EBI. Descargado de https://www.ebi.ac.uk/Tools/pfa/pratt/

PRATT. (2003). University of Geneva, Switzerland. Descargado de https://web.expasy.org/pratt/

RCSB web service. (s.f.). Descargado de https://www.rcsb.org/docs/programmatic-access/file-download-services#other-downloads-offered-by-rcsb-pdb

Sali, A. (1989). *MODELLER Tutorial Advanced.* University of California, San Francisco. Descargado de https://salilab.org/modeller/tutorial/advanced.html

Samdani, A., y Vetrivel, U. (2018, junio). *POAP: A GNU parallel based multithreaded pipeline of open babel and AutoDock suite for boosted high throughput virtual screening.* Descargado 2023-08-20, de https://linkinghub.elsevier.com/retrieve/pii/S1476927117305753

ScanProsite tool. (1996). Swiss Institute of Bioinformatics. Descargado de https://prosite.expasy.org/scanprosite/

ScanProsite - user manual. (s.f.). Descargado de https://prosite.expasy.org/scanprosite/scanprosite_doc.html

Schneider, N., Volkamer, A., Nittinger, E., y Rarey, M. (2016). Supporting biocatalysis research with structural bioinformatics. En L. Hilterhaus, A. Liese, U. Kettling, y G. Antranikian (Eds.), *Biocatalysis.* Wiley-VCH Verlag GmbH & Co. KGaA. Descargado de https://doi.org/10.1002/9783527677122.ch5 doi: 10.1002/9783527677122.ch5

Schrödinger Platform. (s.f.). Schrödinger, Inc. Descargado de https://www.schrodinger.com/platform/drug-discovery

Schrödinger Release 2023-4: Glide. (2023). Schrödinger, LLC. New York, NY.

Siew, N., Elofsson, A., Rychlewski, L., y Fischer, D. (2000, septiembre). MaxSub: an automated measure for the assessment of protein structure prediction quality. *Bioinformatics*, *16*(9), 776–785. Descargado 2023-06-24, de https://academic.oup.com/bioinformatics/article/16/9/776/307645 doi: 10.1093/bioinformatics/16.9.776

Tamura, K., Stecher, G., y Kumar, S. (2021, junio). MEGA11: Molecular Evolutionary Genetics Analysis Version 11. *Molecular Biology and Evolution*, *38*(7), 3022–3027. Descargado 2023-08-20, de https://academic.oup.com/mbe/article/38/7/3022/6248099 doi: 10.1093/molbev/msab120

Trott, O., y Olson, A. J. (2009). AutoDock Vina: Improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading. *Journal of Computational Chemistry*, NA–NA. Descargado 2023-07-27, de https://onlinelibrary.wiley.com/doi/10.1002/jcc.21334 doi: 10.1002/jcc.21334

UniProt. (2003). European Bioinformatics Institute. Descargado de https://www.uniprot.org/

Vue.js. (2014, febrero). Netlify, Netguru. Descargado de https://vuejs.org/

Wang, Z., Sun, H., Yao, X., Li, D., Xu, L., Li, Y., ... Hou, T. (2016). Comprehensive evaluation of ten docking programs on a diverse set of protein–ligand complexes: the prediction accuracy of sampling power and scoring power. *Physical Chemistry Chemical Physics*, *18*(18), 12964–12975. Descargado 2023-08-20, de http://xlink.rsc.org/?DOI=C6CP01555G doi: 10.1039/C6CP01555G

Webb, B., y Sali, A. (2016, junio). Comparative Protein Structure Modeling Using MODELLER. *Current Protocols in Bioinformatics*, *54*(1). Descargado 2023-07-27, de https://onlinelibrary.wiley.com/doi/10.1002/cpbi.3 doi: 10.1002/cpbi.3

Whitford, D. (2005). *Proteins: structure and function*. Hoboken, NJ: J. Wiley & Sons.

Wong, F., Krishnan, A., Zheng, E. J., Stärk, H., Manson, A. L., Earl, A. M., ... Collins, J. J. (2022, septiembre). Benchmarking \textbackslashtextlessspan style="font-variant:small-caps;"\textbackslashtextgreaterAlphaFold\textbackslashtextless/span\ textbackslashtextgreater -enabled molecular docking predictions for antibiotic discovery. *Molecular Systems Biology*, *18*(9). Descargado 2023-04-15, de https://onlinelibrary.wiley.com/doi/10.15252/msb.202211081 doi: 10.15252/msb.202211081

wwPDB consortium, Burley, S. K., Berman, H. M., Bhikadiya, C., Bi, C., Chen, L., ... Ioannidis, Y. E. (2019, enero). Protein Data Bank: the single global archive for 3D macromolecular structure data. *Nucleic Acids Research*, *47*(D1), D520–D528. Descargado 2023-07-27, de https://academic.oup.com/nar/article/47/D1/D520/5144142 doi: 10.1093/

nar/gky949

YASARA-Help. (s.f.). Descargado de http://www.yasara.org/index.html

Yin, S., Biedermannova, L., Vondrasek, J., y Dokholyan, N. V. (2008, agosto). MedusaScore: An Accurate Force Field-Based Scoring Function for Virtual Drug Screening. *Journal of Chemical Information and Modeling*, *48*(8), 1656–1662. Descargado 2023-06-24, de https://pubs.acs.org/doi/10.1021/ci8001167 doi: 10.1021/ci8001167

Yu, Y., Casamajo, A. R., Finnigan, W., Schnepel, C., Barker, R., Morrill, C., ... Turner, N. J. (2022, noviembre). Structure-based design of biocatalytic panels for pharmaceuticals synthesis. Descargado de https://chemrxiv.org/engage/chemrxiv/article-details/637159b9bef5d41b8a54e6a6 doi: 10.26434/chemrxiv-2022-vz6gd

Zvelebil, M. (2008a). Chapter 13: Comparative genomics. En *Understanding bioinformatics* (1st ed., p. 521-566). New York, NY: Garland Science.

Zvelebil, M. (2008b). Chapter 14: Functional genomics and systems biology. En *Understanding bioinformatics* (1st ed., p. 587-593). New York, NY: Garland Science. (Section 14.4)

Zvelebil, M. (2008c). Chapter 2: Basics of molecular biology. En *Understanding bioinformatics* (1st ed., p. 25-44). New York, NY: Garland Science.

Anexo 1

En este apartado se presentan los anexos que complementan el trabajo realizado en esta tesis de grado.

Estos anexos amplían la comprensión de los conceptos tratados y proporcionan material de referencia para aquellos interesados en profundizar en el tema, brindando una mayor contextualización.

1. Formatos

En esta Sección se presentan los diferentes formatos tratados en el análisis y desarrollo de este trabajo.

1.1. Formato fasta

El formato FASTA surge del paquete de *software* FASTA (Pearson y Lipman, s.f.). Una secuencia en este formato se representa por una serie de líneas de hasta 120 caracteres. La primera línea comienza con el caracter ">"seguido del nombre e identificador de la secuencia. En las líneas posteriores se expresa la secuencia. (Pearson, 2003)

```
>Sequence name and identifier
A F A S Y T  .... actual sequence.
F S S        .... second line of sequence.
> Next sequence name and identifier
```

1.2. Formato PIR

El formato PIR es el formato utilizado para el modelado comparativo. Ejemplos de formato PIR:

```
>P1;5fd1
structureX:5fd1:1    :A:106  :A:ferredoxin:Azotobacter vinelandii: 1.90: 0.19
AFVVTDNCIKCKYTDCVEVCPVDCFYEGPNFLVIHPDECIDCALCEPECPAQAIFSEDEVPEDMQEFIQLNAELA
EVWPNITEKKDPLPDAEDWDGVKGKLQHLER*

>P1;1fdx
sequence:1fdx:1    : :54   : :ferredoxin:Peptococcus aerogenes: 2.00:-1.00
AYVINDSC--IACGACKPECPVNIIQGS--IYAIDADSCIDCGSCASVCPVGAPNPED----------------
--------------------------------*
```

En la primera línea de cada secuencia se especifica “>P1;”, luego el código de la proteína. Debe ser único para todas las proteínas en el archivo. La segunda línea de cada entrada contiene la información necesaria para extraer las coordenadas atómicas en el PDB correspondiente. Los campos en esta línea están separados por “:” y son los siguientes:

Campo 1: Una especificación de si se dispone o no de estructura 3D y del tipo de método utilizado para obtener la estructura (structureX, X-ray; structureN, NMR; structureM, model; sequence, sequence). Sólo ‘structure’ también es un valor válido.

Campo 2: El nombre o código del archivo PDB. Se usa para obtener datos estructurales. No tiene por qué ser único.

Campo 3-6: Los identificadores de residuos y cadenas para el primer (campos 3-4) y el último residuo (campos 5-6) de la secuencia en las líneas posteriores.

Campo7: Nombre de la proteína. Opcional.

Campo8: Fuente de la proteína. Opcional.

Campo9: Resolución del análisis cristalográfico. Opcional.

Campo10: Factor R del análisis cristalográfico. Opcional.

Cuando se utiliza un archivo de alineación junto con información estructural, se deben completar los dos primeros campos; el resto puede estar vacío. Si la alineación no se utiliza junto con datos estructurales, todos los campos excepto el primero pueden estar vacíos. Esto significa que en el modelado comparativo, las estructuras de *templates* deben tener al menos los dos primeros campos especificados, mientras que la secuencia objetivo solo debe tener el primer campo completo.

Cada secuencia debe terminar con el carácter ‘*’.

Cuando el primer carácter de la línea de secuencia es el carácter final, ‘*’, la secuencia se obtiene del archivo de coordenadas PDB especificado.

El archivo de alineación puede contener cualquier número de líneas en blanco entre las entradas de proteínas. También puede contener líneas de comentarios por fuera de las entradas de proteínas, debiendo comenzar con los identificadores ‘C;’ o ‘R;’ como los dos primeros caracteres de la línea.

También se utiliza un archivo de alineación para ingresar secuencias no alineadas.

1.3. Formato pdb

El formato pdb describe las estructuras de proteínas y ácidos nucleicos, sus coordenadas atómicas, las asignaciones de estructuras secundarias y la conectividad atómica. A continuación se realiza una breve descripción de sus componentes claves.

HEADER, TITLE, AUTHOR proporcionan información general del archivo; como por ejemplo su fecha de creación, autor, y una breve descripción de

su estructura.

REMARK puede contener anotaciones de forma libre, pero también acomoda información estandarizada.

ATOM y HETATM contienen información sobre la ubicación tridimensional de los átomos y heteroátomos en la estructura de la biomolécula. Cada línea contiene detalles sobre un átomo, incluyendo su nombre, número de serie, coordenadas x, y, z, y otros atributos como el tipo de elemento y el nombre del residuo al que pertenece.

CONECT describe las conexiones covalentes entre átomos en la estructura, como enlaces sencillos, dobles y triples.

El formato PDB permite compartir, analizar y visualizar estructuras de macromoléculas,lo que lo convierte en un estándar importante en la bioinformática y la investigación estructural.

1.4. Formato xml

Es un lenguaje de marcado que se utiliza para estructurar y almacenar datos. Se basa en etiquetas que definen la jerarquía de los datos. Es legible tanto por humanos como por máquinas, y es ampliamente utilizado para intercambiar información entre sistemas, aplicaciones y servicios web. XML es extensible, lo que permite definir etiquetas y estructura de datos específicas, y es compatible con una variedad de plataformas y lenguajes de programación. También admite la validación de datos mediante esquemas.

2. Algoritmos

En esta Sección se presentan algunos de los algoritmos empleados en el desarrollo de este proyecto de grado.

2.1. Algoritmo CLUSTALW

Alinea cada par de secuencias de entrada y a partir del alineamiento calcula la identidad por pares. Transforma las identidades en una medida de distancia, generando una matriz de distancias. Por último construye un árbol mediante el método de agrupación-unión de vecinos.

El siguiente paso es utilizar el árbol para construir una alineación progresiva. En cada nodo del árbol binario se construye una alineación por pares, partiendo desde las hojas hacia la raíz. La primera alineación se realiza a partir de dos secuencias, y las siguientes alineaciones son de uno de los tipos:

secuencia-secuencia, perfil-secuencia o perfil-perfil, donde "perfil"significa la alineación múltiple de las secuencias en un determinado nodo interno del árbol.

2.2. Algoritmo MUSCLE

Calcula las distancias a partir de subsecuencias denominadas kmers, k-tuplas o palabras que dos secuencias tienen en común, sin construir un alineamiento. Transforma las identidades en una medida de distancia, generando una matriz de distancias. Finalmente construye un árbol del que obtiene un alineamiento múltiple que puede mejorar su calidad si se calcula las identidades por pares de cada par de secuencias. Esto proporciona una nueva matriz de distancia, a partir de la cual se estima un nuevo árbol. Tras comparar el primer árbol obtenido con el último, se re-alinean los subgrupos donde sea necesario para producir una alineación múltiple progresiva a partir del nuevo árbol. Si los dos árboles son idénticos no hay nada para hacer, pero en caso de que no haya subárboles que coincidan, debe repetirse todo el procedimiento de alineación progresiva. En general, el árbol es estable cerca de las hojas, con necesidad de algunos realineamientos cerca de la raíz.

El procedimiento de calcular identidades por pares, estimar un nuevo árbol, comparar árboles, y realinear se denomina refinamiento del árbolz debe repetirse hasta que el árbol se estabilice o hasta alcanzar un número máximo de iteraciones.

Una vez el árbol sea estable, el algoritmo pasa a un nuevo procedimiento diseñado para mejorar la alineación múltiple. El conjunto de secuencias se divide en dos subconjuntos y se construye un perfil para cada uno de los dos subconjuntos en función de la alineación múltiple actual. Luego, estos dos perfiles se vuelven a alinear entre sí usando el mismo algoritmo de alineación por pares que se usa en la etapa progresiva. Si esto mejora una "puntuación objetiva" que mide la calidad de la alineación, entonces se mantiene la nueva alineación múltiple, de lo contrario, se descarta. De forma predeterminada, la puntuación objetiva es la puntuación clásica de suma de pares que toma el promedio (ponderado en secuencia) de la puntuación de alineación por pares de cada par de secuencias en la alineación (Edgar, 2004b), (Edgar, 2004a).

3. Implementación

Esta sección proporciona más detalle al lector sobre aspectos de la implementación.

3.1. Obtención del centro para el *docking* molecular

Para obtener el centro para el *docking* molecular se realiza una búsqueda de las coordenadas cartesianas del átomo ingresado como parámetro de entrada, sujeto a la posición devuelta por el paso del filtrado para la posición especificada en la secuencia de referencia, dentro del archivo `pdb` del modelado al que se le va a efectuar el procesamiento. A modo de ejemplo, en el caso de que el filtrado devuelva la posición 311, para el siguiente fragmento del modelo:

```
ATOM   2256  O   PHE A 310    51.346  66.048  57.493  1.00999.99     O
ATOM   2257  N   ASP A 311    53.415  66.156  56.664  1.00900.49     N
ATOM   2258  CA  ASP A 311    53.417  65.332  55.556  1.00900.49     C
ATOM   2259  CB  ASP A 311    54.838  65.264  54.763  1.00900.49     C
ATOM   2260  CG  ASP A 311    56.323  65.368  55.020  1.00900.49     C
ATOM   2261  OD1 ASP A 311    56.697  66.508  55.343  1.00900.49     O
ATOM   2262  OD2 ASP A 311    57.110  64.399  54.883  1.00900.49     O
ATOM   2263  C   ASP A 311    52.453  64.141  55.809  1.00900.49     C
ATOM   2264  O   ASP A 311    52.214  63.879  56.849  1.00900.49     O
ATOM   2265  N   ASP A 312    51.955  63.276  54.939  1.00738.94     N
```

Y átomo OD1, se obtienen las coordenadas x=56.697, y=66.508, z=55.343.

3.2. Cálculo de largo del motivo

El motivo encontrado por el Pratt está compuesto por una secuencia de caracteres alfanuméricos y otros caracteres especiales, que se representa con una expresión regular.

se proporcionan algunas pautas generales para calcular el largo mínimo de la expresión regular:

- Caracteres Literales: Los caracteres literales en una expresión regular (por ejemplo, 'abc') contribuyen al largo mínimo. En este caso, el largo mínimo sería la longitud de la cadena literal, que es 3 en el ejemplo.
- Cuantificadores: Los cuantificadores en una expresión regular indican la cantidad de repeticiones necesarias para que la expresión sea válida. Los cuantificadores comunes incluyen * (cero o más repeticiones), + (una o más repeticiones) y ? (cero o una repetición). El largo mínimo depende del cuantificador y del contenido que cuantifica. Por ejemplo, en la expresión regular 'a+b+', el largo mínimo sería 2, ya que ambos caracteres 'a' y 'b' deben estar presentes al menos una vez.
- Grupos y Opciones: Los grupos '( )' y las opciones — permiten definir estructuras alternativas en una expresión regular. El largo mínimo de un grupo u opción depende de las partes dentro de ellos. Por ejemplo, en la expresión regular '(ab—cd)', el largo mínimo sería 2, ya que se debe coincidir con al menos una de las opciones 'ab' o 'cd'.
- Clases de Caracteres: Las clases de caracteres como '[ ]' especifican un conjunto de caracteres posibles que pueden coincidir. El largo mínimo depende del número mínimo de caracteres especificados en la clase. Por ejemplo, en la expresión regular '[a-zA-Z]', el largo mínimo sería 1, ya que al menos un carácter alfabético debe coincidir.

A modo de ejemplo, el largo mínimo para el siguiente motivo es: C-x(2,4)-[DE] es 1+4+1=6.

4. Casos de prueba

En esta Sección se presentan casos de prueba complementarios.

4.1. Caso de prueba: Actividad IRED

- Nombres de las secuencias de análisis: IRED C, IRED D, IRED K, IRED M, IRED A, IRED B, IRED N, gi—460838084—dbj—BAM99302.1— (R)imine reductase [Streptomyces sp. GF3587], gi—460838082—dbj—BAM99301.1— (S)imine reductase [Streptomyces sp. GF3546], Translation\of\S\83\IRAriel y Translation\of\Imino\reductasa\R\04420.
- Secuencia de referencia: IRed N
- Aminoácidos de interés:

Residuo en posición respecto a secuencia de referencia	187	140	194
Aminoácidos para filtrar	Y,D,E	P, V, T, I ,W, F, Q	F,M,L,I

- Cadenas a modelar: A y B
- *Docking* centro:
  - Posición: posición 194 de la secuencia de referencia alineada
  - Átomo: OD1
  - Radio de la esfera: 10A o 12A
- Ligando: Proporcionado por el usuario

4.2. Caso de prueba: Actividad RedAm

- Nombres de las secuencias de análisis: IR047, IR048, IR050, IR044, IR077, IR061, IR056, IR066, IR091, IR058, IR070 y IR081.
- Secuencia de referencia: 5G6R_1—Chains A, B—IMINE REDUCTASE—ASPERGILLUS ORYZAE (5062)
- Aminoácidos de interés:

Residuo en posición respecto a secuencia de referencia	93	169	177
Aminoácidos para filtrar	S, T o N	D o Y	W,L,Y o A

- Cadenas a modelar: A y B
- *Docking* centro:
  - Posición: posición 173 de la secuencia de referencia alineada
  - Átomo: OD1
  - Radio de la esfera: 10A o 12A:
- Ligando: Proporcionado por el usuario

5. Repositorio

En el siguiente link se encuentra el repositorio con el código fuente del proyecto de grado realizado: https://gitlab.fing.edu.uy/ana.rodriguez/bioprot.git

6. Manual de Usuario

UNIVERSIDAD DE LA REPÚBLICA URUGUAY

FACULTAD DE INGENIERÍA

Manual de usuario

presentado por

Stephanie Casas, Sabrina Sellanes y Ana Laura Rodriguez

Supervisor

Carlos Testuri

Montevideo, 22 de diciembre de 2023

Índice general

Interfaz de Usuario

Pantalla Principal

La interfaz de usuario de Bioprot es intuitiva y fácil de usar. Esta sección del manual guía a través de los elementos clave de la interfaz y explica cómo navegar por ella.

En la Figura 1 se presenta la pantalla principal de la aplicación.

Figura 1: Pantalla principal

Ingreso de la Secuencia de Referencia

Al acceder a la aplicación a través del botón “Comenzar”, se presenta una pantalla como la que se muestra en la Figura 2. En esta etapa, se solicita la secuencia de referencia.

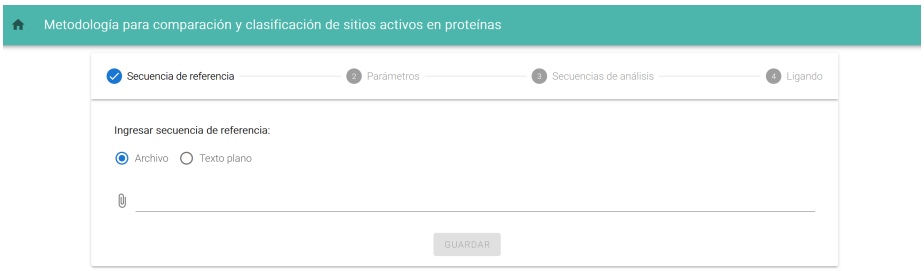

Figura 2: Ingreso secuencia de referencia

Se brindan dos alternativas para llevar a cabo este proceso.

Opción de Archivo:

- Esta opción permite la carga de un archivo que contiene las secuencias de referencia en formato fasta.
- El botón con forma de "clip" debe ser seleccionado para ubicar y cargar el archivo deseado desde el dispositivo.
- Se debe garantizar que el archivo esté en formato fasta antes de proceder con la carga.
- Una vez que el archivo haya sido seleccionado, la acción debe ser confirmada y la aplicación procesará las secuencias contenidas en el mismo.

Opción de Texto Plano:

- Si se prefiere ingresar manualmente la secuencia de referencia en formato fasta, se debe seleccionar esta opción.
- En el campo de texto proporcionado, la secuencia de referencia deseada debe ser escrita o pegada desde el portapapeles.
- La secuencia ingresada debe cumplir con el formato fasta.
- Después de introducir la secuencia, se requiere confirmar la acción, y la aplicación procesará la información proporcionada.

Es importante recordar que el cumplimiento del formato fasta es esencial para que la aplicación pueda interpretar y emplear la secuencia de referencia de manera efectiva.

Especificación de Aminoácidos

Habilitación de la Opción de Especificación de Aminoácidos de Interés después de Ingresar la Secuencia de Referencia.

Tras la introducción de la secuencia de referencia, la opción de "Especificación de aminoácidos de interés" será habilitada, como se muestra en la Figura 3.

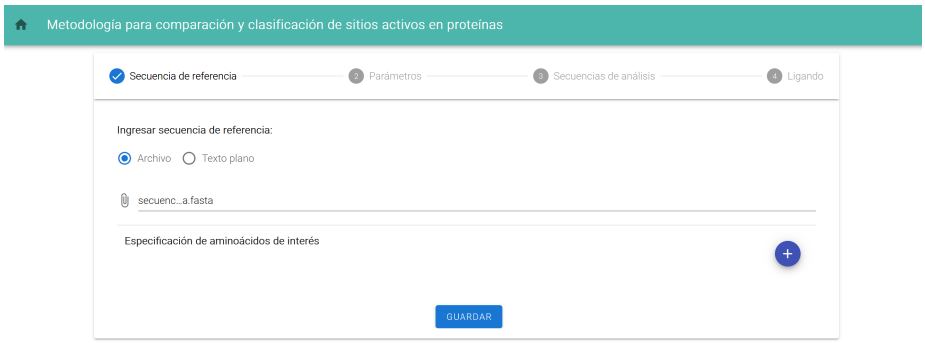

Figura 3: Especificación de Aminoácidos de interés

Al presionar el botón "+", se permite la selección de la posición deseada dentro de la secuencia. En esta sección, se encuentran las siguientes opciones:

- Posición: La posición en la que se desean especificar los aminoácidos de interés puede ser seleccionada a través de un combobox que muestra las posiciones disponibles dentro de la secuencia.
- Seleccionar Aminoácidos: Los aminoácidos que deben estar en la posición seleccionada pueden ser elegidos en un combobox que ofrece una variedad de opciones de aminoácidos de interés.
- Obligatorio: Se especifica la decisión de si es necesario que los aminoácidos seleccionados estén presentes en la posición. Las opciones disponibles son "Sí" o "No", en función de los requisitos específicos.

Cada vez que se presiona el botón "+", se añade una nueva línea en la sección, lo que posibilita la especificación de múltiples posiciones y sus respectivos aminoácidos de interés. Esta funcionalidad proporciona flexibilidad para la definición precisa de criterios relacionados con la secuencia de referencia y los aminoácidos de interés, también presionando el ícono rojo al comienzo de cada línea, se brinda la posibilidad de eliminar las líneas si así se desea.
Asimismo, se brinda la posibilidad de elegir la cantidad de aminoácidos obligatorios en función de los valores "Sí" seleccionados en cada sección de "Obligatorio." Esta cantidad solo puede ser menor o igual a la cantidad de valores "Sí"

seleccionados.

En caso de dejar incompletas algunas de las especificaciones de aminoácidos, al presionar el botón “Guardar” la aplicación despliega una alerta que indica cuáles son los campos que faltan completar, sin permitir avanzar a la siguiente pantalla, como puede observarse en la Figura 4.

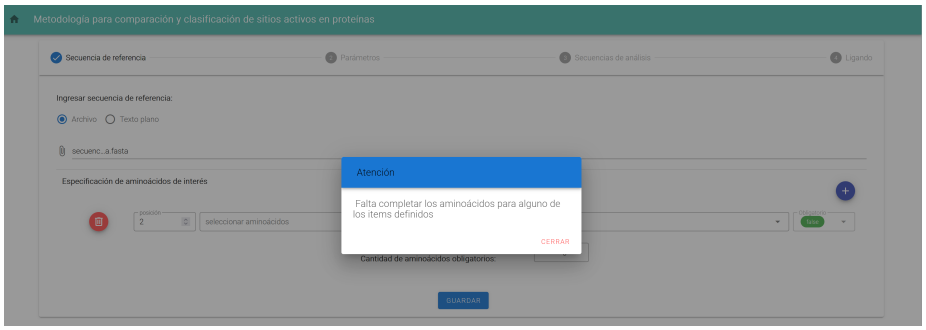

Figura 4: Alerta de datos incompletos

Además, la opción de “Guardar” estará deshabilitada hasta que la secuencia de referencia sea ingresada. Una vez ingresada toda la información mencionada, al hacer clic en el botón “Guardar”, se procede a la siguiente pantalla que se refiere a los “Parámetros de Configuración de cada paso del proceso.” Simultáneamente, se muestra un mensaje emergente (*pop-up*), como se ve en la Figura 5, que indica que los datos se han cargado correctamente.

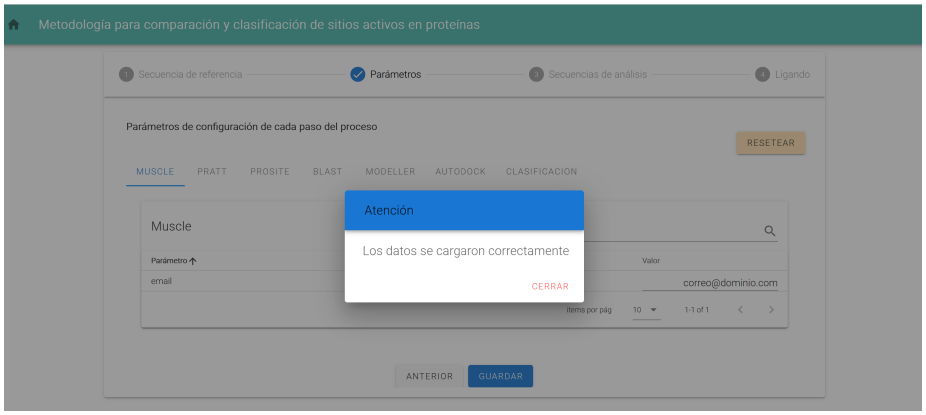

Figura 5: Datos ingresados correctamente

Ingreso de Parámetros

En la pantalla de Parámetros, presentada en la Figura 6, se ingresan los parámetros necesarios para la ejecución de cada paso del flujo. Muchos parámetros están predefinidos y pueden sobreescribirse si así se desea.

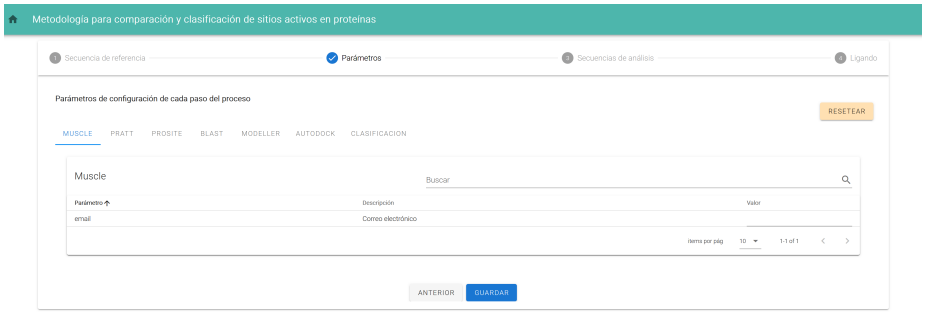

Figura 6: Ingreso de parámetros de configuración

En primer lugar, en la pestaña correspondiente al paso MUSCLE, se encuentra un único parámetro que debe ser configurado, que es el "Correo electrónico", como se puede ver en la Tabla 1. El valor del correo electrónico debe ser ingresado en el campo designado bajo la etiqueta "Valor".

Tabla 1: Muscle

Parámetro	Descripción	Valor
email	Correo electrónico	correo@dominio.com

En cada pestaña que corresponde a un paso del proceso, se dispone de una barra de búsqueda que permite la búsqueda y el hallazgo de un parámetro específico. El nombre del parámetro o su descripción pueden ser ingresados en esta barra de búsqueda, y los parámetros que coincidan con la entrada serán resaltados por la aplicación, lo que facilitará la configuración eficiente de los parámetros requeridos en cada paso del proceso.

También se cuenta con un botón denominado "Resetear" que permite que todos los parámetros sean restablecidos a sus valores por defecto, tal como estaban inicialmente. Al hacer clic en este botón, cualquier configuración o cambios que hayan sido realizados en los parámetros serán revertidos, restaurando así los valores originales de manera inmediata. Esta función es útil en caso de que se desee regresar a la configuración inicial o eliminar cualquier configuración personalizada.

En caso de que un parámetro no sea ingresado, no se permite continuar avanzando en el proceso y se genera un mensaje de error, como se muestra en

la Figura 7. Es necesario asegurarse de que todos los parámetros sean proporcionados con la información necesaria antes de avanzar a la siguiente etapa.

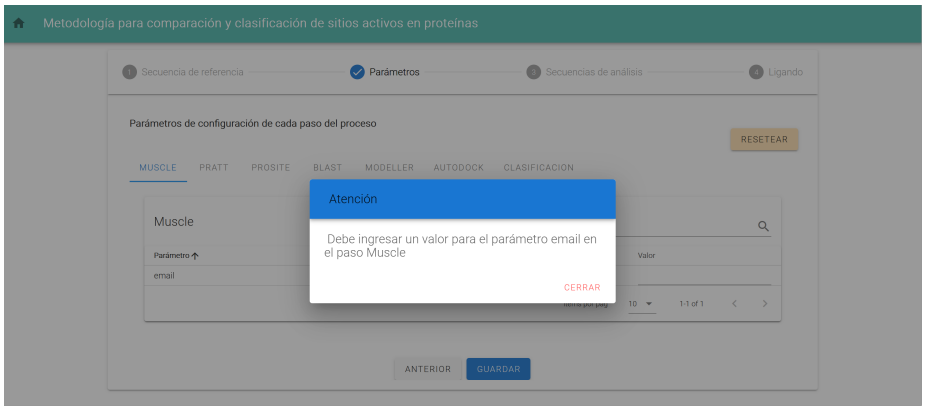

Figura 7: Falta ingresar un parámetro

En segundo lugar, en la pestaña correspondiente al paso PRATT, se encuentran parámetros predefinidos, como se puede ver en la Tabla 2. El valor de los parámetros puede cambiarse en caso de ser necesario en la ejecución que se esté realizando.

Tabla 2: Pratt

Parámetro	**Descripción**	**Valor**
minPerc	Porcentaje mínimo de coincidencia entre las secuencias y el patrón	100
patternPosition	Posición del patrón en la secuencia (off / complete / start)	off
maxPatternLength	Largo máximo de un patrón	50
maxNumPatternSymbols	Número máximo de símbolos en el patrón	50
maxNumWildcard	Largo máximo de wildcards (x)	5
maxNumFlexSpaces	Largo máximo de espacios flexibles	2
maxFlexibility	Flexibilidad máxima de un wildcard	2
maxFlexProduct	Límite superior en el producto de flexibilidades	10
patternSymbolFile	Archivo de símbolos del patrón	off
numPatternSymbols	Número de símbolos de patrones utilizados en la búsqueda inicial	20
patternScoring	Pattern scoring (info, mdl, tree, dist, ppvl)	info
patternGraph	Permite el uso de una alineación o una secuencia de consulta para restringir la búsqueda de patrones	seq
searchGreediness	"Greediness" de la búsqueda (min 0)	3
patternRefinement	Cuando este parámetro está en on, los patrones encontrados durante la búsqueda inicial de patrones se ingresan a un algoritmo de refinamiento donde se pueden agregar símbolos de patrones más ambiguos	on
genAmbigSymbols	Si el parámetro está en on, se utilizan los símbolos ambiguos enumerados en el archivo de símbolos. Si está en off, solo las letras necesarias para coincidir con las secuencias de entrada se incluyen en las posiciones de patrón ambiguas	on
patternFormat	Si el parámetro está en on, los patrones saldrán en estilo PROSITE. Cuando está en off, los patrones salen en un estilo de patrón de consenso más simple donde x coincide exactamente con un símbolo de secuencia arbitraria y - coincide con cero o un símbolo de secuencia arbitraria	on
maxNumPatterns	Número máximo de patrones (entre 1 y 100)	50
maxNumAlignments	Número máximo de alineaciones (entre 1 y 100)	50

Tabla 2: Pratt (continuación)

Parámetro	**Descripción**	**Valor**
printPatterns	Si la opción está en on, Pratt imprimirá la ubicación de los segmentos de secuencia que coincidan con cada uno de los mejores patrones (máximo 52)	on
printingRatio	Establece el valor utilizado para imprimir la información de resumen sobre dónde se encuentran las coincidencias de patrones en cada secuencia	10

En tercer lugar, en la pestaña correspondiente al paso Prosite, se encuentran parámetros predefinidos, como se puede ver en la Tabla 3. El valor de los parámetros puede cambiarse en caso de ser necesario en la ejecución que se esté realizando.

Tabla 3: Prosite

Parámetro	**Descripción**	**Valor**
greed	Define el modo de coincidencia de patrones. Extiende a lo sumo elementos de patrón de longitud variable	1
include	Define el modo de coincidencia de patrones. Permite coincidencias incluidas entre sí (implica superposición)	0
lowscore	Escaneo de alta sensibilidad (muestra coincidencias débiles)	off
minhits	Número mínimo de hits por secuencia coincidente	1
noprofile	Excluye perfiles. Escanea solo contra patrones	off
overlap	Define el modo de coincidencia de patrones. Permite coincidencias parcialmente superpuestas	1
skip	Excluye motivos con alta probabilidad de ocurrencia	on

En cuarto lugar, en la pestaña correspondiente al paso BLAST, se encuentran parámetros predefinidos, como se puede ver en la Tabla 4. El valor de los parámetros puede cambiarse en caso de ser necesario en la ejecución que se esté realizando.

Tabla 4: Blast

Parámetro	**Descripción**	**Valor**
align	Formateo para las alineaciones	0
alignments	Número máximo de alineaciones coincidentes informadas en la salida	50
compstats	Usar estadísticas basadas en composición	F
dropoff	Valor que indica cuánto puede caer la puntuación	0
exp	Limita el número de puntuaciones y alineaciones informadas según el e-value	10
filter	Filtrar regiones de baja complejidad de secuencia	F
gapalign	Realizar alineación utilizando gaps	TRUE
gapext	Penalización sobre el puntaje por prolongar un gap en la secuencia	-1
gapopen	Penalización sobre el puntaje por crearse un gap en la secuencia	-1
matrix	Matriz de puntuación	BLOSUM62
scores	Número máximo de resúmenes de puntuación coincidentes informados en la salida	50
task	Indica tipo de búsqueda	blastp
transltable	Consultar código genético para usar en la traducción	1

Para el caso del parámetro “align”, en la Tabla 5 se especifican los posibles valores numéricos con su correspondiente significado.

Tabla 5: Valores del parámetro align en Blast

Valor	**Nombre**	**Descripción**
0	pairwise	La query y match se generan como una alineación por pares con una línea de consenso entre las dos secuencias. En el consenso, los estados de coincidencia se representan como: coincidencia idéntica a la base/residuo, similitud como ‘+’, y falta de coincidencia como un espacio.
1	Query-anchored identities	Las coincidencias encontradas se muestran en relación con la secuencia de consulta sin espacios como diferencias con la consulta. Las identidades aparecen como puntos (.), las similitudes en mayúsculas, las coincidencias en minúsculas, y los espacios en blanco como guiones (-). Las inserciones se indican con una línea que apunta al sitio de inserción con la secuencia insertada en otra línea.

Tabla 5: Valores del parámetro *align* en Blast (continuación)

Valor	**Nombre**	**Descripción**
2	Query-anchored non-identities	Las coincidencias encontradas se muestran en relación con la secuencia de consulta sin espacios como diferencias con la consulta. Las identidades y similitudes aparecen en mayúsculas, las faltas de coincidencia en minúsculas, y los espacios en blanco como guiones (-). Las inserciones se indican con una línea que apunta al sitio de inserción con la secuencia insertada en otra línea.
3	Flat query-anchored identities	Las coincidencias encontradas se muestran en relación con la secuencia de consulta con espacios como diferencias con la consulta. Las identidades aparecen como puntos (.), las similitudes en mayúsculas, las coincidencias en minúsculas, y los espacios en blanco como guiones (-).
4	Flat query-anchored non-identities	Las coincidencias encontradas se muestran en relación con la secuencia de consulta con espacios como diferencias con la consulta. Las identidades y similitudes aparecen en mayúsculas, las faltas de coincidencia en minúsculas, y los espacios en blanco como guiones (-).
5	BLASTXML	Salida NCBI BLAST XML en lugar de un informe de texto sin formato.
6	tabular	Salida resumida en formato tabular.
7	tabular with comment lines	Salida resumida en formato tabular.
8	Text ASN.1	Salida en formato ASN.1.
9	Binary ASN.1	Salida en formato ASN.1.
10	Comma-separated values	Salida de resumen como valores separados por comas.
11	BLAST archive format (ASN.1)	Salida en formato de archivo BLAST (ASN.1).

En quinto lugar, en la pestaña correspondiente al paso Modeller, se encuentran dos parámetros predefinidos, como se muestra en la Tabla 6. El valor de los parámetros puede cambiarse en caso de ser necesario para la ejecución que se está realizando.

Tabla 6: Modeller

Parámetro	**Descripción**	**Valor**
cadenas	Cantidad de cadenas a modelar	2
tolerancia	Porcentaje de tolerancia entre el largo del template y la secuencia a modelar	25

En sexto lugar, en la pestaña correspondiente al paso de Autodock, se encuentran cuatro parámetros predefinidos y dos parámetros que deben ser ingresados, tal como se muestra en la Tabla 7. Los valores de los parámetros predefinidos pueden ser modificados si es necesario para la ejecución que se va a llevar a cabo. Es importante destacar que los parámetros no predefinidos, como "átomo" y "posición", deben ser ingresados de manera obligatoria.

Tabla 7: Autodock

Parámetro	**Descripción**	**Valor**
atomo	Centro del docking: átomo	OD1
distancia	Distancia en Ångströms para la esfera	10
energy_range	Diferencia de energía máxima entre el mejor y el peor modo de unión mostrado (kcal/mol)	3
exhaustiveness	Exhaustividad de la búsqueda global (aproximadamente proporcional al tiempo)	8
num_modes	Número máximo de modos de unión para generar	9
posicion	Centro del docking: posición	194

Por último, en la pestaña correspondiente al paso de la clasificación se encuentra el parámetro predefinido maxDistancia como muestra la Tabla 8, que puede ser modificado de ser necesario al momento de la ejecución.

Tabla 8: Clasificación

Parámetro	**Descripción**	**Valor**
maxDistancia	Distancia en Ångströms para la clasificación	5

Ingreso de la Secuencia de Análisis

Al ingresar todos los parámetros mencionados y posteriormente presionar el botón "Guardar" se efectúa la transición a la pantalla "Secuencia de análisis" como se presenta en la Figura 8.

Figura 8: Secuencia de Análisis

De manera similar a lo mencionado en la pantalla Secuencia de Referencia, la secuencia de análisis puede ser ingresada utilizando las opciones "Archivo" o "Texto plano" en formato `fasta`. Es importante destacar que el ingreso de la secuencia de análisis es un requisito obligatorio.
En caso de no ingresar y presionar el botón "Guardar" se muestra un mensaje emergente (*pop-up*), como se muestra en la Figura 9, indicando el error.

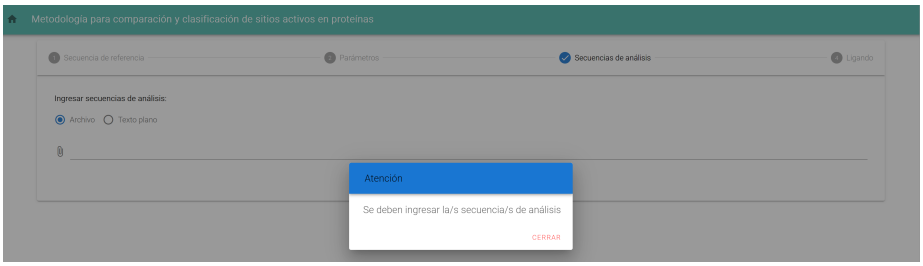

Figura 9: Alerta de secuencia de análisis no ingresada

Ingreso del Ligando

Una vez que la secuencia de análisis ha sido ingresada y el botón "Guardar" ha sido presionado, se procede a la pantalla "Ligando".

Aquí se puede ingresar el ligando con las opciones "Archivo" o "Texto plano" en formato `pdb`. Es obligatorio ingresar el ligando, pues si no es ingresado, al presionar el botón "Procesar" se muestra un mensaje emergente con el error.

En todos los pasos, "Secuencia de referencia", "Parámetros", "Secuencias de análisis" y "Ligando" es posible ir avanzando y retrocediendo para modificar lo ingresado en pantallas anteriores.

Una vez completados todos los pasos, se presiona el botón “Procesar” que se puede ver en la Figura 10.

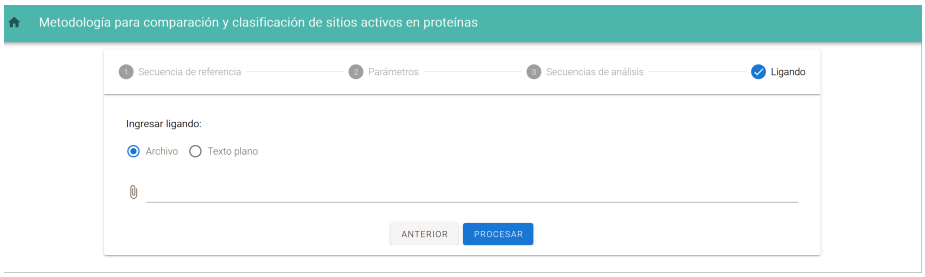

Figura 10: Pantalla ligando

Al seleccionar la opción de procesar, se despliega un *pop-up*, como se muestra en la Figura 11, donde se debe hacer clic en “ACEPTAR” para iniciar el procesamiento o en “CERRAR” en caso de que no se desee ejecutar aún.

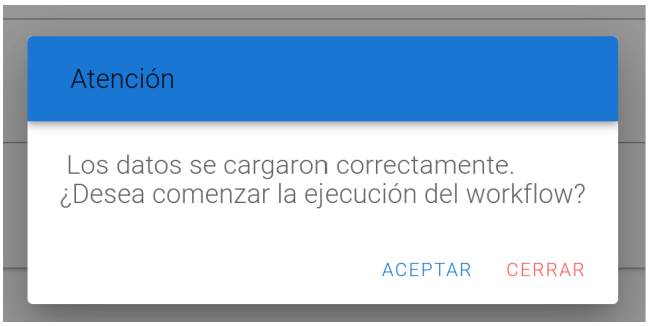

Figura 11: Confirmar ejecución del *workflow*

Estado de Ejecución

En caso de comenzar la ejecución del *workflow*, se visualiza el progreso tal como se aprecia en la Figura 12.

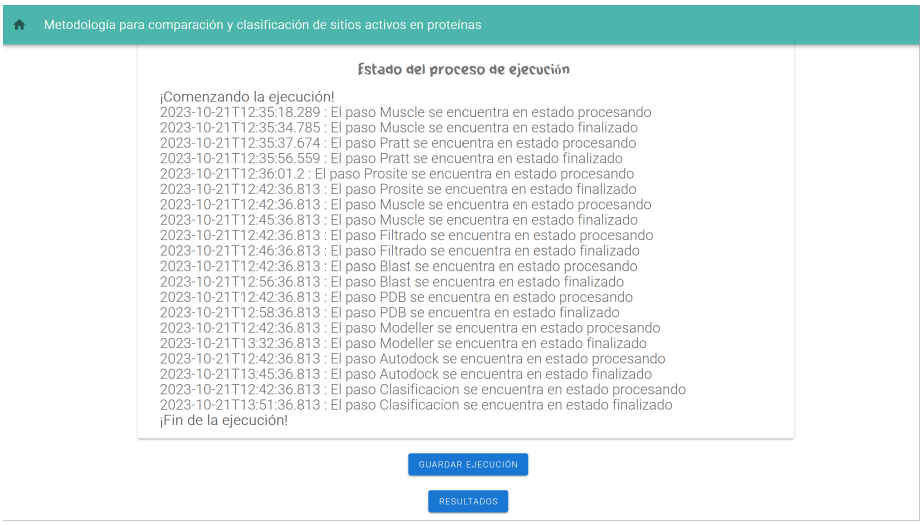

Figura 12: Estado de Ejecución

Al finalizar la ejecución se habilitan los botones “Guardar ejecución” y “Resultados”. Al presionar “Guardar ejecución” se despliega un *pop-up* como se muestra en la Figura 13, donde el usuario debe indicar el nombre con el que desea almacenar los resultados. La opción “Guardar” evalúa si ya existe una ejecución con dicho nombre, en ese caso se indica el error, de lo contrario, aparece un *pop-up* indicando que se realiza el guardado con éxito.

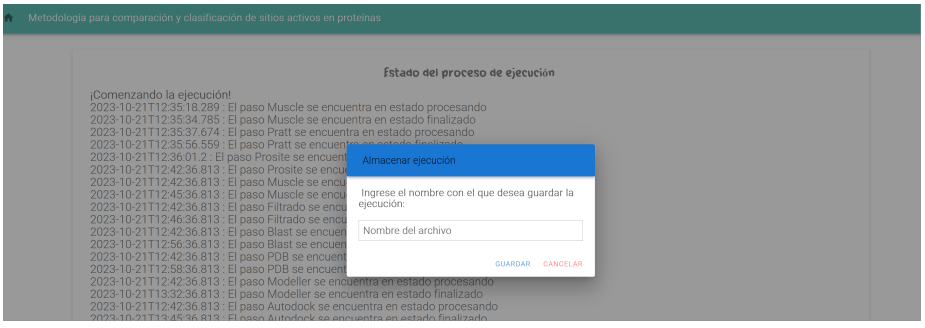

Figura 13: Guardar ejecución

Una vez almacenada la ejecución se vuelve a deshabilitar el botón “Guardar ejecución” quedando únicamente habilitado el botón “Resultados”.

Visualización de Resultados

Si se presiona el botón “Resultados” la aplicación da la posibilidad de ver los resultados obtenidos en una ejecución reciente con el botón “Ver resultados”. La aplicación muestra los archivos almacenados en estructura de árbol y ofrece además la posibilidad de seleccionarlos y visualizarlos, como se presenta en la Figura 14.

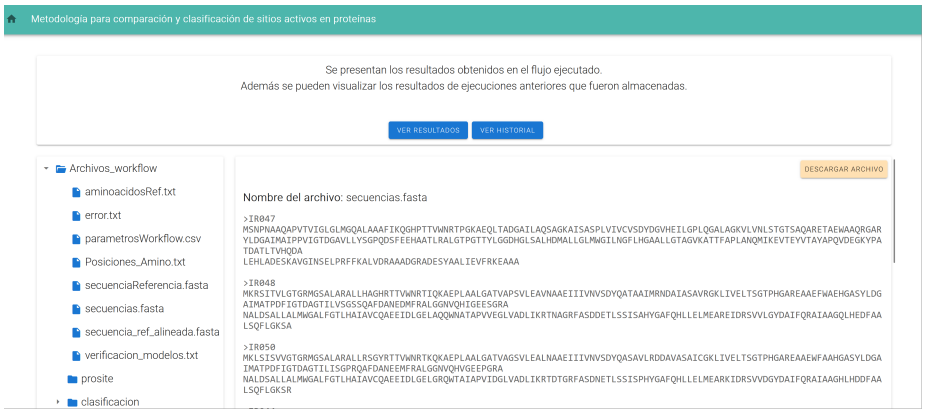

Figura 14: Pantalla de resultados

El botón “Descargar Archivo” permite la descarga del archivo que se está visualizando.

Al presionar el botón “Ver historial” como muestra la Figura 15, se presentan los resultados de ejecuciones anteriores previamente guardadas. A la izquierda se listan los nombres de las diferentes ejecuciones, y desplegando cada una se puede ver la estructura de árbol de la misma forma que se visualiza en la opción “Ver resultados” teniendo también las funcionalidades de visualizar y descargar cada archivo.

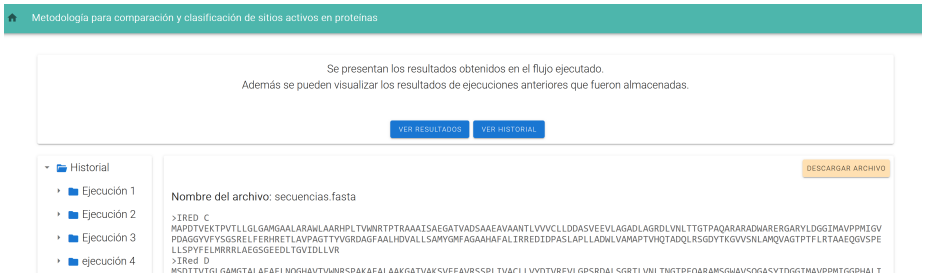

Figura 15: Pantalla historial de resultados

Esperamos que este manual haya sido de ayuda para que puedas utilizar con facilidad la aplicación, de forma eficiente y logres una buena experiencia como usuario, proporcionándo una guía completa y clara que te permita aprovechar al máximo todas las funciones y características que nuestra aplicación tiene para ofrecer.