Universidad de las Ciencias Informáticas

Facultad 6

Título: "*Herramienta informática para la generación de candidatos a péptidos antimicrobianos mediante el empleo de Algoritmos Genéticos*"

Trabajo de Diploma para optar por el título de

Ingeniero Informático

Autor: Leandro Marcelo Torres Silva

Tutor: MSc. Longendri Aguilera Mendoza.

La Habana, junio de 2015

"Año 56 de la Revolución"

¿Por qué esta magnífica tecnología científica, que ahorra trabajo y nos hace la vida más fácil, nos aporta tan poca felicidad? La repuesta es está, simplemente: porque aún no hemos aprendido a usarla con tino.

Albert Einstein

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis que tiene por título: Herramienta informática para la generación de candidatos a péptidos antimicrobianos mediante el empleo de Algoritmos Genéticos y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Leandro Marcelo Torres Silva

_____

Firma del Autor

Longendri Aguilera Mendoza

_____

Firma del Tutor

DATOS DE CONTACTO

Tutor: MSc. Longendri Aguilera Mendoza
Institución: Universidad de las Ciencias Informáticas
E-mail: loge@uci.cu

AGRADECIMIENTOS

Para toda mi gente.

DEDICATORIA

…a mi abuela que ya no está y a mi madre que si está y es lo mejor y único que tengo.

RESUMEN

En la actualidad existen algunas bacterias que han desarrollado resistencia a los fármacos convencionales y se hacen cada vez más difícil combatirlas. Los péptidos antimicrobianos han surgido como una estrategia de control ante esta situación ya que los mismos se caracterizan por atacar las membranas de los agentes patógenos y son considerados por la industria farmacéutica como un arma potencial para crear antibióticos de nuevo tipo. No obstante, su uso presenta limitaciones como la citotoxicidad, longitud de la secuencia y estabilidad, por lo que aún continúan siendo estudiados. Gracias a los adelantos tecnológicos, los péptidos pueden ser sintetizados en laboratorios, pero se hace muy costoso dicho proceso y por tal motivo se utilizan los métodos computacionales que generen candidatos iniciales que luego deben ser evaluados de forma experimental. La presente investigación tuvo como resultado principal una herramienta informática capaz de identificar candidatos a péptidos antimicrobianos. Implementándose un algoritmo genético para explorar el espacio combinatorio de posibles secuencias de aminoácidos y obtener así las secuencias candidatas. Durante la exploración y evaluación en el espacio de búsqueda se utilizó un árbol de decisión, reportado en la literatura, para predecir la actividad biológica y guiar la búsqueda hacia regiones factibles. La herramienta informática obtenida puede ser extendida en forma de módulos para incorporar otras estrategias de exploración y evaluación, por lo que puede convertirse en un marco de trabajo para la búsqueda informada de candidatos a péptidos antimicrobianos.

PALABRAS CLAVE

Péptidos antimicrobianos, algoritmos genéticos, arboles de decisión.

ABSTRACT

At present there are some bacteria that have developed resistance to conventional drugs and increasingly become difficult to combat. Antimicrobial peptides have emerged as a control strategy in this situation since they are characterized by attacking the membranes of pathogens and are considered by the pharmaceutical industry as a potential weapon to create new type antibiotics. However, its use has limitations as cytotoxicity, sequence length and stability, which are still being studied. Thanks to technological advances, the peptides can be synthesized in laboratories, but becomes very expensive this process and for that reason computational methods to generate initial candidates which must then be evaluated experimentally used. This research had as main result a software tool capable of identifying candidates for antimicrobial peptides. A genetic algorithm implemented combinatorial space to explore possible amino acid sequences and obtain the candidate sequences. A decision tree, reported in the literature to predict the biological activity and guide the search to feasible regions used during the exploration and evaluation in the search space. The software tool can be obtained as modules extended to incorporate other strategies of exploration and evaluation, so that it can become a framework for the search for candidate's informed antimicrobial peptides.

KEYWORDS:

Antimicrobial peptides, genetic algorithms, decision trees.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Los péptidos antimicrobianos (AMP por sus siglas en inglés) son secuencias de aminoácidos de origen natural que son producidos por diversos organismos, ya sean bacterias, plantas, invertebrados, peces, anfibios, aves y mamíferos, incluidos los humanos, para actuar como un mecanismo de defensa ante la presencia de microbios. En este contexto, los AMPs han surgido como una alternativa posible para controlar las enfermedades infecciosas provocadas por microorganismos resistentes a los antibióticos convencionales. Sin embargo, el uso de los AMPs como agente terapéutico tiene algunas limitaciones que han restringido las aplicaciones exitosas de este tipo de medicamento. Entre estas limitaciones se encuentran la estabilidad, la citotoxicidad, y muy importante la longitud de las cadenas de aminoácidos debido a que es muy caro sintetizar bloques de los mismos. A pesar de estas limitaciones tienen propiedades compensatorias, incluyendo actividades secundarias antitumorales e inmunomoduladoras.

En la industria farmacéutica algunos AMPs han sido utilizados con fines clínicos y comerciales, por ejemplo el AMBICIN (nisina) (Adams 1996). Por lo tanto, el diseño racional de AMPs emerge como una importante herramienta que tiene como objetivo desarrollar fármacos con un mayor rendimiento contra las bacterias resistentes (Porto, Franco et al. 2012).

Para el diseño e identificación de nuevos AMPs varias técnicas asistidas por computadoras han sido desarrolladas (Fjell, Hiss et al. 2012). Un enfoque computacional que ha sido ampliamente utilizado en ese sentido, consiste en generar o modificar la secuencia de aminoácidos de los AMPs conocidos, seleccionados como plantilla, mientras se predice y evalúa la actividad biológica de los péptidos generados hasta encontrar los compuestos líderes deseados. El principal reto a enfrentar bajo este enfoque es el alto costo computacional que se tiene al explorar el espacio combinatorio de todos los ordenamientos posibles de aminoácidos. Por ejemplo, al ser 20 los tipos de aminoácidos más comunes a encontrar en un péptido, el espacio de secuencias de aminoácidos de longitud *n* contiene 20^n péptidos y la probabilidad de encontrar el péptido más idóneo es muy pequeña. Este problema de encontrar el péptido idóneo tiene un elevado costo computacional y puede ser atacado utilizando técnicas de la Inteligencia Artificial. En particular con la búsqueda informada (Aguilar and Rivas 2001) en un espacio de estado se puede llegar a una solución favorable, o sea un conjunto de AMPs que cumplan con ciertos criterios de actividad.

Por la situación antes expuesta se plantea como **problema de la investigación:** ¿Cómo explorar el espacio de los ordenamientos posibles de aminoácidos para identificar candidatos a péptidos antimicrobianos?

Se define como **objeto de estudio** de la investigación: Identificación de candidatos a péptidos antimicrobianos. El **campo de acción** ha sido enmarcado en: Búsqueda informada en un espacio de estado para la identificación y clasificación de candidatos a péptidos antimicrobianos.

Se define como **objetivo general** de la investigación Desarrollar una herramienta informática que permita identificar candidatos a péptidos antimicrobianos, mediante una búsqueda informada en el espacio de estado del problema.

Para dar cumplimiento al objetivo planteado se proponen las siguientes **tareas de investigación:**

- Representación del problema mediante una búsqueda en un espacio de estados.
- Estudio y selección de la(s) técnica(s) de búsqueda informada en un espacio de estado.
- Análisis y diseño de la herramienta informática que va a permitir la identificación de nuevos candidatos a péptidos antimicrobianos mediante la técnica seleccionada de búsqueda informada.
- Implementación de la herramienta informática previamente diseñada.
- Validación y puesta a punto de la herramienta informática implementada.

Para el cumplimiento del objetivo general planteado se utilizarán los siguientes **métodos de investigación:**

Métodos teóricos:

- **Análisis y síntesis:** Para la elaboración del marco teórico referencial, con la finalidad de seleccionar los elementos base para el desarrollo de la plataforma informática, así como en el diagnóstico inicial y final del problema y la formulación de las conclusiones.
- **Modelación:** Para realizar los diagramas correspondientes a los modelos de análisis, diseño e implementación para la identificación de nuevos candidatos a péptidos antimicrobianos.
- **Histórico y lógico:** Permitió el estudio del desarrollo histórico sobre las plataformas informáticas y para los referentes teóricos y metodológicos.

Métodos empíricos:

- **Análisis documental:** Es utilizado para realizar una revisión detallada de la literatura y documentos relacionados con las plataformas informáticas, los métodos heurísticos basados en algoritmos genéticos y los descriptores moleculares.

El presente trabajo está compuesto por introducción, tres capítulos, conclusiones, recomendaciones y referencia bibliográfica. Los capítulos están estructurados de la siguiente manera:

Capítulo 1. Fundamentos Teóricos: En este capítulo se presentan los elementos teóricos que sirven de base a la investigación para darle solución al problema planteado. Se presentan las herramientas y metodología a utilizar para el desarrollo del sistema propuesto.

Capítulo 2. Análisis y diseño de la aplicación: El presente capítulo muestra el modelo de dominio, así como la especificación de los requisitos funcionales y no funcionales de la plataforma informática, se presenta el diagrama de casos de usos del sistema para identificar la relación entre los actores y casos de usos. Además se brinda una descripción de la arquitectura de la plataforma informática, los patrones de diseños más utilizados y la aplicación de los mismos para la confección de los diagramas del modelo del diseño.

Capítulo 3. Implementación y pruebas: En este capítulo, se expondrá el diagrama de componentes, donde se describen los elementos físicos de la plataforma informática y sus relaciones. Por último, se describen detalladamente las pruebas que se le realizan a la plataforma informática, para corroborar el correcto funcionamiento de la aplicación cumpliendo con los requisitos implementados.

CAPÍTULO 1: FUNDAMENTOS DE LA INVESTIGACIÓN, METODOLOGÍA Y TECNOLOGÍAS EMPLEADAS.

En este capítulo se presentan los elementos teóricos que sirven de base a la investigación para darle solución al problema planteado. Se presentan las herramientas y metodología a utilizar para el desarrollo del sistema propuesto.

1.1 ETAPAS DE LA BÚSQUEDA INFORMADA PARA LA IDENTIFICACIÓN DE CANDIDATOS A PÉPTIDOS ANTIMICROBIANOS

La búsqueda informada (Millán 2011) es aquella que utiliza el conocimiento específico del problema más allá de la definición del problema en sí mismo. Se utiliza para resolver problemas que generalmente no tienen solución algorítmica conocida o es tan compleja que no tiene una implementación práctica computacional, como es el caso en la presente investigación. A continuación se describen las etapas involucradas en la búsqueda informada para la identificación de candidatos a péptidos antimicrobianos.

Como se muestra en la Figura 1, a partir de un conjunto de secuencias iniciales se obtienen un conjunto de secuencias finales aplicando operadores de modificación a dichas secuencias (Exploración). La etapa de evaluación guía el proceso de exploración brindando un conocimiento para saber cuan exitoso está siendo dicho proceso.

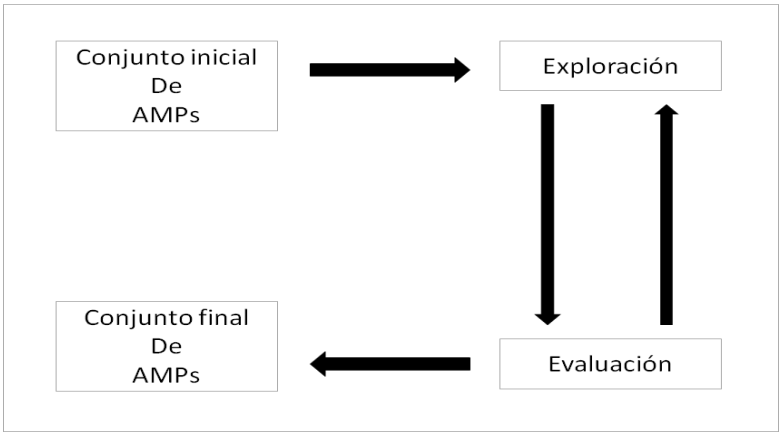

Figura 1: Etapas de la búsqueda informada de candidatos a péptidos antimicrobianos.

1.2 REVISIÓN DE LAS TÉCNICAS EMPLEADAS EN LA BÚSQUEDA INFORMADA DE CANDIDATOS A PÉPTIDOS ANTIMICROBIANOS

A continuación se hace una revisión de técnicas empleadas en la búsqueda informada de nuevos candidatos a péptidos antimicrobianos basadas en métodos computacionales.

1.2.1 ESTRATEGIAS DE DISEÑO

Al igual que en (Fjell, Hiss et al. 2012) se decide definir como estrategia de diseño la técnica responsable de implementar la etapa de exploración. Es decir, aquella técnica que a partir de un conjunto de estados iniciales obtiene nuevos estados mediante operadores aplicados a dichos estados.

Algoritmos Genéticos (AGs)

"Los Algoritmos Genéticos son algoritmos de búsqueda basados en la mecánica de selección natural y de la genética natural. Combinan la supervivencia del más apto entre estructuras de secuencias con un intercambio de información estructurado, aunque aleatorizado, para constituir así un algoritmo de búsqueda que tenga algo de las genialidades de las búsquedas humanas" (Golberg 1989).

En (Schneider and Wrede 1994) se realiza el diseño de las nuevas secuencias de péptidos mediante un Algoritmo Genético. Dicho algoritmo se basa en la calificación dada por una Red Neuronal Artificial como heurística para la selección de los mejores candidatos de la población Se hace una mutación de 12 secuencias n veces, lo cual conduce a una descendencia de n secuencias, la mejor secuencia de la descendencia de acuerdo con el filtro neural se selecciona como secuencia de los padres para la próxima generación. Se utiliza una distribución gaussiana para calcular las probabilidades de mutación, en la cual la varianza puede ser interpretada como una mutabilidad especifica de la posición. Para la mutación definen varios conceptos: un nuevo residuo, o aminoácido *Rnew*, un viejo residuo, *Rold* que es el que va a ser sustituido por el nuevo, la posición específica de la mutación P, un número aleatorio de la distribución gaussiana G, y la distancia métrica del aminoácido empleado D. Luego D = G * P y *Rnew* = F (D, *Rold*). F es la función de selección del aminoácido. En (Fjell, Jenssen et al. 2011) también se realiza el diseño de AMPs mediante un algoritmo genético. Al tamaño de la población se le permitió variar para asegurar que todos los péptidos de alta puntuación permanecieran en la población. Se utilizó una tasa de mutación de cambio de 15 aminoácidos. En la Figura 2 (A) se muestra como realizaban la mutación intercambiando

aminoácidos seleccionados y en la Figura 2 (B) se muestra como realizaban el cruzamiento mediante el operador de cruce de un punto.

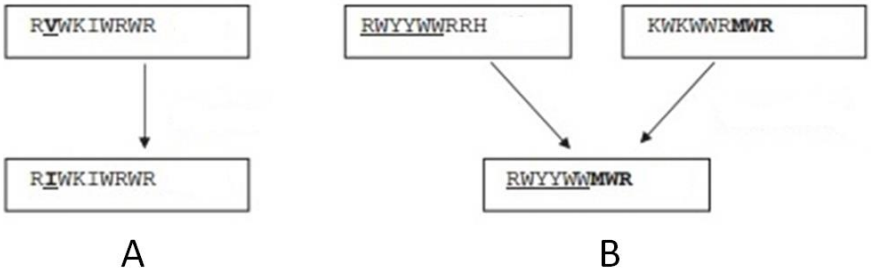

Figura 2: Ejemplo de mutación y cruzamiento.

Colonia de hormigas

En (Hiss, Bredenbeck et al. 2007) se realiza el diseño de los AMPs mediante la optimización colonia de hormigas. Se parte de un espacio de decisión mediante el cual las hormigas transitan de un aminoácido hacia otro trazando una trayectoria que formara posteriormente un nuevo candidato.

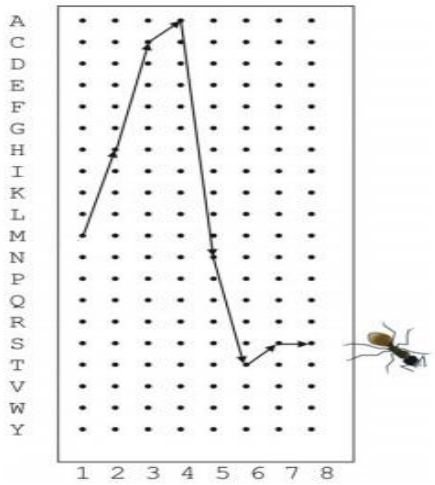

Figura 3: Optimización colonia de hormigas.

La Figura 3 muestra un camino de una hormiga artificial en un espacio de decisión. Los puntos representan probabilidades de cambio (concentraciones de feromonas) para pasar de una posición del residuo a la siguiente. En este ejemplo la hormiga artificial se movio a lo largo de un camino que representa la secuencia del péptido MHCANTSS.

1.2.2 FUNCIONES DE EVALUACIÓN

Algunas de las funciones de evaluación empleadas en la búsqueda informada de péptidos antimicrobianos utilizan descriptores moleculares (Todeschini and Consonni 2008) que constituyen valores numéricos para representar información estructural, electrónica, estérica, así como una determinada propiedad químico-física de la molécula en estudio. Por ejemplo, dado un vector de descriptores moleculares X = [x1, x2,…, xn] para cada AMP, entonces se pudiera construir y utilizar un modelo predictivo f, que relacione estos parámetros con la actividad biológica *A*. De tal manera que A = f (X), donde f es un modelo tipo Quantitative Structure Activity Relationships (QSAR) (Todeschini and Consonni 2009).

Redes Neuronales Artificiales (RNAs)

Las RNAs están motivadas en modelar la forma de procesamiento de la información en sistemas nerviosos biológicos. Especialmente, por la forma de funcionamiento del cerebro humano, que es completamente distinta de un computador digital convencional (Izaurieta and Saavedra 2000).

En (Schneider and Wrede 1994) las propiedades fisicoquímicas utilizadas fueron: hidrofobicidad, hidrofilicidad, polaridad y volumen de la cadena lateral para obtener la evaluación de las secuencias. Esta evaluación era la combinación del resultado de tres redes neuronales. La calidad de la secuencia se daba de forma numérica. En (Fjell, Jenssen et al. 2011) también se realiza la evaluación de las secuencias de péptidos mediante una red neuronal artificial. Construyen un software para clasificar péptidos como altamente activa o inactiva basándose en un conjunto de 44 descriptores QSAR calculados para cada péptido combinado con RNAs entrenadas en la actividad medida de 1433 péptidos.

Arboles de Decisión

Los arboles de decisión son herramientas para la toma de decisiones (Moskowitz, Wright et al. 1982).

En (Lira, Perez et al. 2013) se obtiene un árbol de decisión para evaluar las secuencias de péptidos. En la Figura 4, los nodos de decisión representan índices obtenidos a partir de descriptores moleculares. Los nodos hojas representan finalmente la actividad de la secuencia. La actividad de la secuencia fue dividida en cuatro niveles: alta, media, baja y ninguna. Los descriptores finales de la investigación con los que se construye el árbol de decisión fueron: *(A) isoelectric point, (C) charge, (B) hydrogen, (F) oxigen, (D) log p, (E) dreiding energy, (G) asa p y (H) balaban index.*

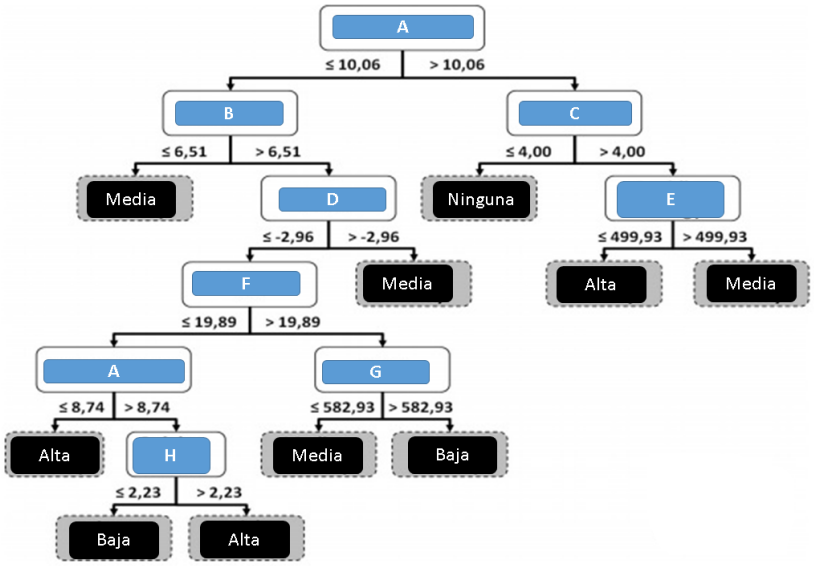

Figura 4: El árbol de decisión obtenido.

1.2 ALGORITMOS GENÉTICOS

Se selecciona como estrategia de diseño un algoritmo genético debido a que han sido utilizados en la optimización de péptidos antimicrobianos, arrojando resultados favorables en la evaluación de las secuencias; convergen más rápido a una solución que la generación aleatoria de secuencias. Además, la codificación en una letra de los aminoácidos facilita la utilización de una estrategia de este tipo, debido a que es una codificación sencilla para la cual se han definido operadores genéticos.

A continuación se muestran las etapas fundamentales de un algoritmo genético:

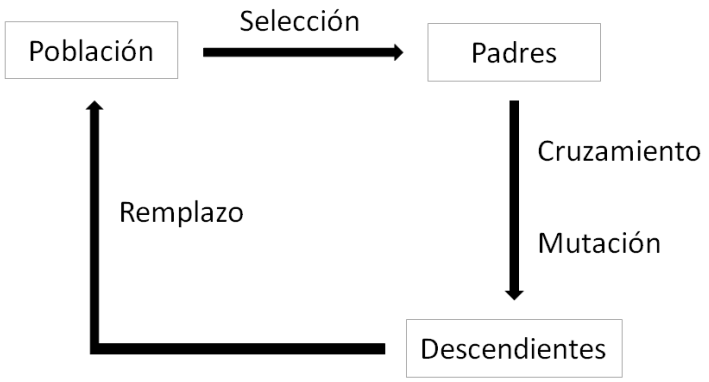

Figura 5: Etapas evolutivas.

1.2.1.1 CODIFICACIÓN

La codificación responde a la descripción de los **estados** de la búsqueda informada. Entre las codificaciones que podemos encontrar están las literales, las alfanuméricas, las numéricas y las binarias (Golberg 1989). La codificación utilizada para los individuos de la población está determinada por la representación en una letra de los 20 aminoácidos más comunes a encontrar a continuación se muestra una tabla con los aminoácidos y su representación.

Aminoácido	**Representación en una letra**
Glicina	G
Alanina	A
Valina	V
Leucina	L
Isoleucina	I
Fenilalanina	F
Tirosina	Y
Triptófano	W
Serina	S
Treonina	T
Cisteína	C

Metionina	M
Ácido aspártico	D
Ácido glutámico	E
Histidina	H
Lisina	K
Arginina	R
Asparginina	N
Glutamina	Q
Prolina	P

1.2.1.2 OPERADOR DE SELECCIÓN

Debido a las características del problema la selección de los nuevos padres serán secuencias con alta evaluación para tratar de que los hijos obtenidos hereden una evaluación de este tipo, similar a lo que ocurre en la naturaleza, se le debe dar mayor oportunidad de reproducirse a los individuos más aptos, es decir, aquellos que tienen mejor valor de aptitud. Entre los operadores de selección pueden agruparse en dos grupos:

- Selección proporcional: Los individuos son elegidos en función de su contribución de aptitud respecto al total de la población.
- Selección mediante torneo: Los individuos son elegidos en base a comparaciones directas entre ellos.

Entre los operadores proporcionales podemos encontrar la ruleta, el cual selecciona los individuos a reproducir teniendo en cuenta su contribución a la población. Entre los operadores basados en torneo podemos encontrar los deterministas que seleccionan al azar dos individuos y gana el de mayor aptitud y los probabilísticos, los cuales generan un número aleatorio y si es mayor que un parámetro p fijado para todo el proceso, gana el de mayor aptitud y en caso contrario el de menor aptitud. Variando el número de individuos que participan en cada torneo se puede modificar la presión de selección. Cuando participan muchos individuos en cada torneo, la presión de selección es elevada y los peores individuos apenas tienen oportunidades de reproducción. Un caso particular es el elitismo global. Se trata de un torneo en el que participan todos los individuos de la población con lo cual la selección se vuelve totalmente determinística (Golberg 1989).

1.2.1.3 CRUZAMIENTO Y MUTACIÓN

Los operadores de cruzamiento y mutación tendrán la responsabilidad de no perder información genética de los padres hacia los hijos. Los operadores que cumplen con esta condición son descritos a continuación:

- Cruce de un punto: Una vez seleccionados dos individuos se cortan sus cromosomas por un punto de cruce seleccionado aleatoriamente para generar dos segmentos diferenciados en cada uno de ellos: la cabeza y la cola. Se intercambian las colas entre los dos individuos para generar los nuevos descendientes.
- Cruce de dos puntos: Para generar la descendencia se escoge el segmento central de uno de los padres y los segmentos laterales del otro padre.
- Mutación de intercambio: Se seleccionan dos genes al azar y se intercambian de posición.
- Intercambio conjunto: Se selecciona un gen al azar y se intercambia de posición con el siguiente.
- Mutación de inversión: Se seleccionan dos puntos al azar en la secuencia y se intercambia la información genética contenida entre ambos.
- Mutación al borde: se selecciona un punto al azar de la secuencia y se elige la dirección de cambio hacia el mínimo o el máximo y se intercambia por el borde superior o inferior según la dirección de cambio (Golberg 1989).

1.2.1.4 TÉCNICA DE REMPLAZO

Se decide implementar un remplazo que sea capaz de mantener las secuencias con mayor evaluación dentro de la población. Dentro de las técnicas de remplazo podemos encontrar:

- En el Modelo Generacional durante cada iteración se crea una población completa con nuevos individuos, la nueva población reemplaza directamente a la antigua (se puede incluir la mejor solución de la vieja población para no perderla).
- En el Modelo Estacionario durante cada iteración se escogen dos padres de la población (diferentes mecanismos de muestreo) y se les aplican los operadores genéticos. El/los descendiente/s reemplaza/n a uno/dos cromosoma/s de la población inicial.

Dentro de los operadores de remplazo podemos encontrar:

- Aleatorio: el nuevo individuo se inserta en un ligar cualquiera de la población.

- Remplazo de los padres: se obtiene espacio para la nueva descendencia liberando el espacio ocupado por los padres.
- Remplazo de similares: una vez obtenida la evaluación de la descendencia se seleccionan un grupo de individuos con aptitud similar y se elige aleatoriamente a quienes remplazar.
- Remplazo de los peores: se eligen los peores individuos de la descendencia y se eligen al azar quienes serán remplazados para dejar lugar a la descendencia (Golberg 1989).

1.3 METODOLOGÍA, HERRAMIENTAS Y TECNOLOGÍAS

La solución propuesta requiere la selección de una metodología de desarrollo, que permita laborar un marco de trabajo para la estructuración, planeación y control del proceso de desarrollo. Se deben seleccionar además las tecnologías y herramientas a utilizar para su construcción, teniendo en cuenta las características que debe tener el sistema y las restricciones de uso de las herramientas.

1.3.1 METODOLOGÍAS DE DESARROLLO

El desarrollo de software no es una tarea fácil. Prueba de ello es que existen numerosas propuestas metodológicas que inciden en distintas dimensiones del proceso de desarrollo. Por una parte tenemos aquellas propuestas más tradicionales que se centran especialmente en el control del proceso, estableciendo rigurosamente las actividades involucradas, los artefactos que se deben producir, y las herramientas y notaciones que se usarán. Estas propuestas han demostrado ser efectivas y necesarias en un gran número de proyectos, pero también han presentado problemas en otros muchos. Una posible mejora es incluir en los procesos de desarrollo más actividades, más artefactos y más restricciones, basándose en los puntos débiles detectados. Sin embargo, el resultado final sería un proceso de desarrollo más complejo que puede incluso limitar la propia habilidad del equipo para llevar a cabo el proyecto. Otra aproximación es centrarse en otras dimensiones, como por ejemplo el factor humano o el producto software.

Rational Unified Process (RUP)

RUP es un proceso de ingeniería del software. Proporciona un acercamiento disciplinado a la asignación de tareas y responsabilidades en una organización de desarrollo. Su propósito es asegurar la producción

de software de alta calidad que se ajuste a las necesidades de sus usuarios finales con unos costos y calendario predecibles.

En definitiva el RUP es una metodología de desarrollo de software que intenta integrar todos los aspectos a tener en cuenta durante todo el ciclo de vida del software, con el objetivo de hacer abarcables tanto pequeños como grandes proyectos software. Además proporciona herramientas para todos los pasos del desarrollo así como documentación en línea para sus clientes (Martínez and Martínez 2002).

Xtreme Programing (XP)

Es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico.

Los principios y prácticas son de sentido común pero llevadas al extremo, de ahí proviene su nombre. Kent Beck, el padre de XP, describe la filosofía de XP sin cubrir los detalles técnicos y de implantación de las prácticas. Posteriormente, otras publicaciones de experiencias se han encargado de dicha tarea. A continuación presentaremos las características esenciales de XP organizadas en los tres apartados siguientes: historias de usuario, roles, proceso y prácticas (Canós, Letelier et al. 2003).

OpenUP

OpenUP es un proceso unificado que aplica enfoques iterativos e incrementales dentro de un ciclo de vida estructurado. OpenUP abarca una filosofía pragmática, ágil, que se centra en la naturaleza colaborativa de desarrollo de software. Es un proceso que puede ampliarse para hacer frente a una amplia variedad de tipos de proyectos.

Es un proceso de desarrollo del software. Es completo en el sentido que puede ser manifestado como todo el proceso para construir un sistema.

- Es extensible ya que en el proceso se pueda agregar o adaptar según lo vayan requiriendo los sistemas. OpenUP es un proceso ágil.
- Es ligero y proporciona una comprensión detallada del proyecto, beneficiando a clientes y desarrolladores sobre productos a entregar y su formalidad.
- Se centra en una arquitectura temprana para reducir al mínimo los riesgos y organizar el desarrollo.
- OpenUP es la metodología utilizada por desarrolladores de alto nivel en casi todo el mundo por sus altas cualidades administrativas (Flores and Salinas).

1.3.2 HERRAMIENTAS Y TECNOLOGÍAS

Lenguaje de modelado: UML

El lenguaje de modelado UML es el estándar más utilizado para especificar y documentar cualquier sistema de forma precisa. Sin embargo, el hecho de que UML sea una notación de propósito muy general obliga a que muchas veces sea deseable poder contar con algún lenguaje más específico para modelar y representar los conceptos de ciertos dominios particulares. Los Perfiles UML constituyen el mecanismo que proporciona el propio UML para extender su sintaxis y su semántica para expresar los conceptos específicos de un determinado dominio de aplicación.

Entre los lenguajes de modelado que define OMG (Object Management Group) el más conocido y usado es sin duda UML (Unified Modelling Language). UML es un lenguaje gráfico para especificar, construir y documentar los artefactos que modelan un sistema. UML fue diseñado para ser un lenguaje de modelado de propósito general, por lo que puede utilizarse para especificar la mayoría de los sistemas basados en objetos o en componentes, y para modelar aplicaciones de muy diversos dominios de aplicación (telecomunicaciones, comercio, sanidad, etc.) y plataformas de objetos distribuidos (como por ejemplo J2EE, .NET o CORBA) (Fuentes and Vallecillo 2004).

JGAP

Para la implementación del algoritmo genético se utiliza el paquete de algoritmos genéticos para Java (JGAP (Meffert, Meseguer et al. 2011) por sus siglas en inglés); el mismo proporciona mecanismos genéticos básicos que se pueden utilizar fácilmente para aplicar los principios evolutivos a soluciones de

problemas. Fue diseñado para que fuera muy fácil de usar, siendo altamente modular y configurable, llegando a ser realmente fácil crear incluso nuevos y personalizados operadores genéticos. JGAP es un entorno de programación que tiene implementados operadores de selección, cruce, mutación y remplazo, por lo tanto facilita las tareas de implementación (Giraldo and Perdomo 2013).

Biojava

La aplicación utiliza ficheros formato FASTA para la exploración. Los mismos contienen la información de las secuencias con un identificador definido. Los ficheros se cargan en la aplicación mediante la biblioteca Biojava (Holland, Down et al. 2008), la cual brinda estas facilidades.

Marvin Beans

Para la implementación de los descriptores moleculares, los cuales brindan el índice para los nodos de decisión, se utilizan los paquetes del programa Marvin Beans (Weber 2008). Dichos paquetes fueron utilizados para la obtención de los descriptores moleculares del árbol de decisión utilizado (ver Figura 4 en la página 18).

Herramienta de modelado: Visual Paradigm

Visual Paradigm para UML (del inglés, Unified Modeling Language) es una herramienta CASE (del inglés, Computer Aided Software Engineering) aplicable en todo el ciclo de vida del desarrollo de software. Soporta UML, SysML (del inglés, Systems Modeling Language), BPMN (del inglés, Business Process Modeling Notation), entre otras tecnologías. Permite dibujar todos los tipos de diagramas de clases, generar código desde diagramas y generar documentación. También proporciona abundantes tutoriales UML, demostraciones interactivas de UML y proyectos UML. Presenta licencia gratuita y comercial. Es fácil de instalar y actualizar y compatible entre ediciones. Visual Paradigm 6.4 ha sido la herramienta seleccionada para soportar el ciclo de desarrollo del componente web para el módulo de Toma de Decisiones del SIAPS. Los cuales lo han seleccionado por su reputación, experiencia, facilidad de los productos y servicios brindados (Reyes and García 2012).

Lenguaje de programación: Java

- Lenguaje de sintaxis sencilla, orientada a objetos e interpretada, que permite optimizar el tiempo y el ciclo de desarrollo (compilación y ejecución).
- Las aplicaciones son portables sin modificación en numerosas plataformas físicas y sistemas operativos.
- Las aplicaciones son resistentes porque el motor de ejecución de Java se encarga de la gestión de la memoria (Java Runtime Environment), y es más fácil escribir programas sin error en comparación a C++, debido a un mecanismo de gestión de errores más evolucionado y estricto.
- Las aplicaciones y en particular las aplicaciones gráficas son eficaces debido a la puesta en marcha y la asunción del funcionamiento de varios procesos ligeros (Thread y multithreading).
- Un programa Java es portable y se puede utilizar sin modificaciones en cualquier plataforma (Windows, Macintosh, Unix, Linux…).
- Java es a la vez un lenguaje y una plataforma de desarrollo.
- Java es un lenguaje interpretado, robusto, securizado, independiente de las arquitecturas, portable, eficaz, multitarea y dinámico (Groussard 2012).

1.3.3 ARQUITECTURA BASADA EN COMPONENTES

Una arquitectura basada en componentes describe una aproximación de ingeniería de software al diseño y desarrollo de un sistema. Esta arquitectura se enfoca en la descomposición del diseño en componentes funcionales o lógicos que expongan interfaces de comunicación bien definidas. Esto provee un nivel de abstracción mayor que los principios de orientación por objetos y no se enfoca en asuntos específicos de los objetos como los protocolos de comunicación y la forma como se comparte el estado (Fuentes and Vallecillo 2004). El estilo de arquitectura basado en componentes tiene las siguientes características:

• Es un estilo de diseño para aplicaciones compuestas de componentes individuales.

• Pone énfasis en la descomposición del sistema en componentes lógicos o funcionales que tienen interfaces bien definidas.

• Define una aproximación de diseño que usa componentes discretos, los que se comunican a través de interfaces que contienen métodos, eventos y propiedades.

Dentro de las implementaciones de arquitecturas basadas en componentes se encuentra Rich Client Platform.

Rich Client Platform (RCP)

Una implementación de arquitectura basada en componentes es RCP. Una RCP (Hoffmann 2008) es un entorno para un ciclo de vida de una aplicación, una base para aplicaciones de escritorio. La mayoría de las aplicaciones de escritorio tienen funcionalidades o características similares, como menús, barras de herramientas, barras de estado, visualización de progreso, personalización de configuración, persistencia y recuperación de datos de usuarios y configuración, mecanismos de internacionalización y localización, sistemas de ayuda, etc. Para todas estas y otras funcionalidades típicas de las aplicaciones de escritorio, una RCP proporciona un marco de referencia mediante el cual se pueden integrar estas funcionalidades y hacer que trabajen de manera conjunta de un modo sencillo.

En la arquitectura cliente-servidor el término rich client (cliente rico) se emplea para clientes donde el procesamiento de la información tiene lugar principalmente del lado del cliente. Una de las ventajas de las Rich Client Applications (RCAs) es que son fácilmente distribuibles y actualizables: ejecutables a través de Internet (por ejemplo mediante Java Web Start (JWS)) o actualizables de forma online mediante mecanismos proporcionados por la aplicación.

Algunas de las principales características de los rich client son:

- Arquitectura modular y flexible
- Independencia de plataforma
- Adaptabilidad al usuario final
- Trabajo online y offline
- Fácil distribución al usuario final
- Fácil actualización del cliente
- Reducción del tiempo de desarrollo

- Consistencia de la interfaz de usuario
- Reusabilidad y fiabilidad

NetBeans Platform

NetBeans Platfom es una implementación de Rich Client Platform. El tamaño y la complejidad de las aplicaciones modernas están en constante aumento. Al mismo tiempo, las aplicaciones profesionales tienen que ser flexibles, sobre todo, de modo que puedan ser rápida y fácilmente extendidas. Esto hace necesario dividir una aplicación en partes distintas. Como resultado, cada parte distinta es un bloque de construcción que conforma una arquitectura modular. Las partes distintas deben ser independientes, por lo que las interfaces bien definidas disponibles que son utilizados por otras partes de la misma aplicación, con las características que tienen otras partes de la aplicación, se pueden utilizar y extender (Böck 2011).

Sistema de módulos del NetBeans IDE

El sistema de módulos de NetBeans se encarga de gestionar todos los módulos. Esto significa que es responsable de tareas como crear el cargador de clases, la carga de módulos, o activarlos o desactivarlos. El Sistema de módulos NetBeans fue diseñado utilizando tecnologías estándar de Java, tanto como sea posible. La idea básica para el formato de módulo tiene su origen en el mecanismo de extensión de Java. Las ideas fundamentales de la especificación del paquete de control de versiones se utilizan para describir y gestionar las dependencias entre aplicaciones de
módulos y aplicaciones de módulos del sistema (Böck 2011).

1.4 CONCLUSIONES

A partir de la revisión bibliográfica realizada se propone utilizar un enfoque heurístico para explorar el espacio de los posibles ordenamientos de aminoácidos y un modelo predictivo para determinar la evaluación de las secuencias candidatas. Se selecciona como estrategia de diseño un algoritmo genético (Golberg 1989) y como modelo para evaluar las secuencias candidatas el árbol de decisión obtenido en (Lira, Perez et al. 2013) el cual se muestra en la Figura 4.

El estudio de las estrategias que se han llevado a cabo para la identificación y clasificación de nuevos candidatos a péptidos antimicrobianos permitió sentar las bases para seleccionar una estrategia de diseño basada en algoritmos genéticos y una función de evaluación basada en un árbol de decisión cuyos nodos de decisión son indicadores brindados por el cálculo de descriptores moleculares.

En este capítulo se decide la realización de la plataforma informática utilizando la plataforma del NetBeans debido a su extensibilidad y flexibilidad mediante el trabajo con componentes y módulos.

El estudio de las diferentes bibliotecas bioinformáticas permitió seleccionar las más adecuadas para el cálculo de los descriptores moleculares necesarios y la aplicación de un algoritmo genético, estas fueron: Marvin Beans para los descriptores y JGAP para el algoritmo genético.

El desarrollo del software se guiará bajo la metodología OpenUP debido a que es una metodología que se adapta al contexto y a las necesidades del sistema, y destaca con claridad las metas propuestas. Para el modelado de los diagramas necesarios se utilizará el lenguaje UML y el Visual Paradigm para la construcción de dichos diagramas.

Para el desarrollo de la plataforma informática se utilizará el lenguaje de programación Java, como entorno integrado de desarrollo el NetBeans IDE debido a su integración con Rich Client Platform y por su entorno agradable que permitirá una interacción con la aplicación de manera sencilla.

CAPÍTULO 2: DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA.

El presente capítulo muestra el modelo de dominio, así como la especificación de los requisitos funcionales y no funcionales de la aplicación, se presenta el diagrama de casos de usos del sistema para identificar la relación entre los actores y casos de usos. Además se brinda una descripción de la arquitectura del sistema, los patrones de diseños más utilizados y la aplicación de los mismos para la confección de los diagramas del modelo del diseño.

2.1 MODELO DE DOMINO

Un modelo del dominio se utiliza con frecuencia como fuente de inspiración para el diseño de los objetos software; es el artefacto más importante que se crea durante el análisis orientado a objetos (Larman 1999).

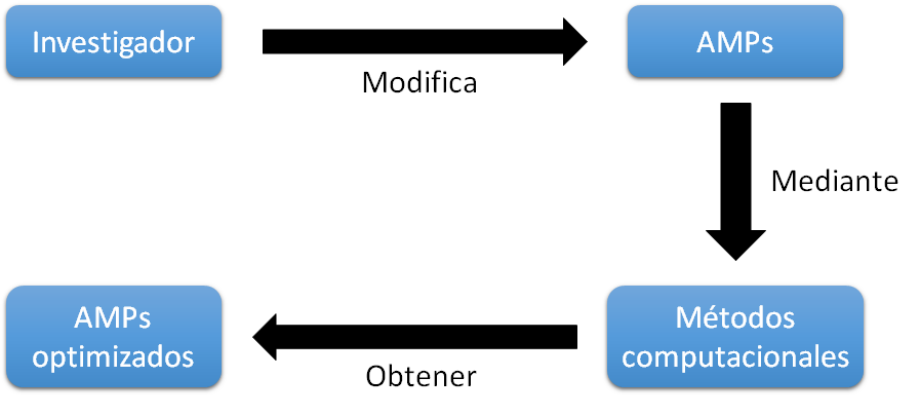

Figura 6: Diagrama de clases del dominio de la aplicación.

En este modelo se evidencian los elementos asociados a la aplicación. El investigador modifica los AMPs naturales mediante métodos computacionales, como es el caso de la presente investigación, para obtener AMPs artificiales que cumplan con un criterio objetivo definido por el mismo investigador.

Tabla 1: Descripción de las clases del dominio de la aplicación.

Investigador	Usuario final encargado de realizar la búsqueda de los nuevos AMPs
AMPs	Conjunto de secuencias iniciales. Péptidos naturales.
Métodos	Técnicas para la modificación de AMPs basadas en algoritmos

computacionales	computables.
AMPs optimizados	AMPs que cumplen con cierto criterio objetivo definido por el investigador. AMPs artificiales.

2.2 REQUISITOS DEL SISTEMA

Los requisitos se han convertido en un punto clave en el desarrollo de las aplicaciones informáticas. Un gran número de proyectos de software naufragan debido a una mala definición, especificación o administración de requisitos. Factores tales como requisitos incompletos o mal manejo de los cambios de los requisitos llevan a proyectos completos al fracaso total.

Requisitos funcionales.

A continuación se describen los requisitos funcionales de la aplicación.

1. Adicionar fichero de datos.
2. Seleccionar población inicial.
3. Seleccionar longitud de cadenas.
4. Seleccionar cantidad de iteraciones.
5. Generar nuevos péptidos antimicrobianos mediante el uso de técnicas heurísticas basadas en algoritmos genéticos.
6. Generar descriptores moleculares.
7. Exportar resultados.

Requisitos no funcionales.

A continuación se describen los requisitos no funcionales de la aplicación.

RnF_1 Usabilidad.

- La aplicación informática debe garantizar un acceso fácil y rápido, contando con un menú que satisfaga las necesidades de los usuarios. La herramienta podrá ser usada sólo por usuarios que posean conocimientos en el dominio de la especialidad.

RnF_2 Interfaz.

- La interfaz debe tener colores y tipo de letra similar a los del NetBeans.

RnF_3 Licencia.

- Al finalizar el desarrollo de la Aplicación Informática y se haya identificado el proyecto que pueda utilizar la solución, la dirección del grupo definirá la licencia por la que se regirá el mismo.

RnF_4 Software.

- Se requiere para el funcionamiento de la aplicación disponer de Windows o Linux como sistema operativo y Máquina Virtual de Java versión 1.7 o superior.

RnF_5 Hardware.

- 2 GB de memoria RAM como mínimo.
- 20 GB de capacidad en el disco duro.
- Procesador de más de dos núcleos.

2.3 CASOS DE USO DEL SISTEMA

En Ingeniería del software, es una técnica para la captura de requisitos potenciales de un nuevo Sistema o una actualización de Software. Cada caso de uso (Buhr, Casselman et al. 1996) proporciona uno o más escenarios que indican cómo debería interactuar el sistema con el usuario o con otro sistema para conseguir un objetivo específico. Normalmente, en los casos de usos se evita el empleo de jergas técnicas, prefiriendo en su lugar un lenguaje más cercano al usuario final.

A continuación se mencionan los casos de uso en que fueron agrupados los requisitos funcionales identificados para el desarrollo de la aplicación:

- Adicionar fichero de datos.(1)
- Configurar búsqueda.(2,3,4)
- Buscar nuevos péptidos antimicrobianos.(5,6)
- Exportar resultados.(7)

Diagrama de casos de uso del sistema.

En la siguiente figura se muestra el diagrama de casos de uso del sistema en el cual se representan todos los casos de uso necesarios para satisfacer los requisitos funcionales del sistema.

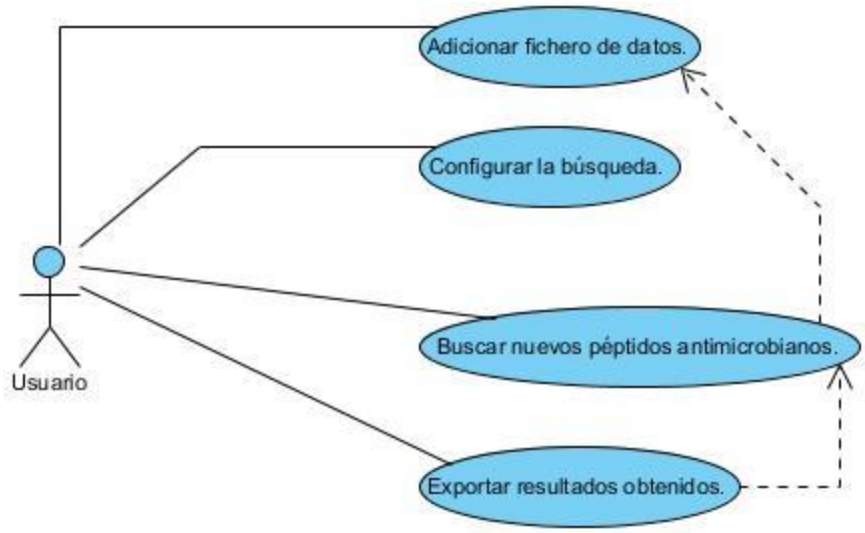

Figura 7: Diagrama de casos de uso del sistema.

Descripción de los casos de uso del sistema.

Tabla 2: Descripción del caso de uso Adicionar fichero de datos.

Objetivo	Adicionar un nuevo fichero de datos.
Actores	Investigador.
Resumen	El caso de uso inicia cuando el investigador selecciona la opción de adicionar un nuevo fichero de datos. El programa le muestra una ventana para la selección del mismo. El caso de uso termina cuando el investigador confirma su selección y el fichero de datos es adicionado al programa.
Complejidad	Media.
Prioridad	Crítico.
Precondiciones	-
Postcondiciones	Se adiciona un nuevo fichero de datos.
Referencias	RF_1
Flujo de eventos	
Flujo básico <Adición satisfactoria>	

	Actor	Sistema
1.	Selecciona la opción ¨Adicionar fichero de datos¨.	
2.		Muestra una interfaz ¨Adicionar fichero de datos¨, con un botón ¨Aceptar¨.
3.	Selecciona el fichero de datos.	
4.		Verifica que el fichero de entrada es correcto.
5.		Adiciona el fichero de datos al programa.
6.		Muestra una notificación de éxito.
Flujos alternos		
Nº Evento. 1a: Los datos son incorrectos.		
	Actor	**Sistema**
1.		Muestra un mensaje de error.

Relaciones	**CU Incluidos**	No procede
	CU Extendidos	No procede
Requisitos no funcionales		
Asuntos pendientes		

Tabla 3: Descripción del caso de uso Configurar búsqueda.

Objetivo	Configurar los parámetros de la búsqueda.
Actores	Investigador.
Resumen	El caso de uso comienza cuando el investigador define los parámetros de la búsqueda que aparecen en la ventana de propiedades del programa. El programa inicia con una configuración ya establecida con valore aleatorios en los parámetros. El usuario decide si proceder en la búsqueda con dichos parámetros o si desea establecer los propios.
Complejidad	Baja.
Prioridad	Media.

Precondiciones	-		
Postcondiciones	Los parámetros de búsqueda quedan especificados.		
Referencias	RF_2, RF_3, RF_4		
Flujo de eventos			
Flujo básico <Configuración satisfactoria>			
	Actor		**Sistema**
7.	Configura los parámetros de búsqueda y confirma.		
8.			Verifica que los parámetros de búsqueda sean correctos.
9.			Redefine los parámetros de búsqueda.
Flujos alternos			
Nº Evento. 1a: Los datos son incorrectos.			
	Actor		**Sistema**
2.			Muestra un mensaje de error.
Relaciones	**CU Incluidos**	No procede	
	CU Extendidos	No procede	
Requisitos no funcionales			
Asuntos pendientes			

Tabla 4: Descripción del caso de uso Buscar nuevos péptidos antimicrobianos.

Objetivo	Identificar y clasificar nuevos candidatos a péptidos antimicrobianos.
Actores	Investigador.
Resumen	El caso de uso comienza cuando el investigador selecciona la opción de ejecutar búsqueda. El sistema le muestra la población seleccionada y los parámetros especificados al usuario, el mismo acepta y comienza la búsqueda. Al finalizar el sistema le muestra los resultados obtenidos en la ventana de salida.

Complejidad	Alta.		
Prioridad	Crítico.		
Precondiciones	Archivo de datos adicionado al sistema y los parámetros de búsqueda correctamente especificados.		
Postcondiciones	La ventana de salida muestra los resultados obtenidos.		
Referencias	RF_5, RF_6		
Flujo de eventos			
Flujo básico <Búsqueda satisfactoria>			
	Actor	**Sistema**	
10.	Selecciona la opción ¨Ejecutar búsqueda¨.		
11.		El sistema muestra una notificación con la población inicial seleccionada y los parámetros de búsqueda especificados.	
12.	Selecciona la opción ¨Aceptar¨.		
13.		Realiza la búsqueda y muestra los resultados obtenidos en la ventana de salida.	
Relaciones	**CU Incluidos**	No procede	
	CU Extendidos	No procede	
Requisitos no funcionales			
Asuntos pendientes			

Tabla 5: Descripción del caso de uso Exportar resultados.

Objetivo	Exportar resultados obtenidos en la búsqueda.
Actores	Investigador.
Resumen	El caso de uso comienza cuando el investigador, luego de realizar la búsqueda, selecciona la opción de exportar los resultados obtenidos. El sistema le da la opción de seleccionar el archivo de salida y el

	investigador confirma y acepta. El sistema guarda los datos obtenidos en el archivo seleccionado por el usuario.		
Complejidad	Alta.		
Prioridad	Media.		
Precondiciones	Búsqueda realizada.		
Postcondiciones	Resultados guardados en fichero seleccionado por el usuario.		
Referencias	RF_7		
Flujo de eventos			
Flujo básico <Búsqueda satisfactoria>			
	Actor	**Sistema**	
14.	Selecciona la opción ¨Exportar resultados¨.		
15.		El sistema muestra una ventana para la selección del archivo de salida.	
16.	Selecciona el archivo de salida y confirma.		
17.		Guarda los resultados de la búsqueda en el archivo seleccionado.	
Relaciones	**CU Incluidos**	No procede	
	CU Extendidos	No procede	
Requisitos no funcionales			
Asuntos pendientes			

2.4 PROPUESTA GENERAL DE LA SOLUCIÓN

A continuación se muestran los operadores y elementos fundamentales utilizados en la implementación del algoritmo genético obtenido.

2.4.1 TAMAÑO DE LA POBLACIÓN

El tamaño de la población va a estar en dependencia de las necesidades del usuario, pero para un correcto funcionamiento y para que se evidencie más explícitamente el trabajo del algoritmo genético se define como población inicial mínima deseada: 10 individuos.

2.4.2 OPERADOR DE SELECCIÓN

El operador de selección utilizado en la implementación del algoritmo genético fue el operador Ruleta (VALENCIA 1997). El mismo fue seleccionado para dar oportunidad a que cualquier individuo de la población sea seleccionado como padre, debido a que se basa en probabilidades aleatorias, aunque las secuencias con mayores evaluaciones van a tener mayores probabilidades de ser seleccionadas. Es un método de selección proporcional debido a que los individuos son elegidos en función de su contribución de aptitud respecto al total de la población.

2.4.3 OPERADOR DE CRUZAMIENTO

Los operadores genéticos del algoritmo responden a las **acciones** para llegar de un **estado** a otro.
El operador de cruzamiento utilizado fue el Cruce en un punto (Golberg 1989). Una vez seleccionados dos individuos se cortan sus cromosomas por un punto de cruce seleccionado aleatoriamente para generar dos segmentos diferenciados en cada uno de ellos: la cabeza y la cola. Se intercambian las colas entre los dos individuos para generar los nuevos descendientes. De esta manera ambos descendientes heredan información genética de los padres.

2.4.4 OPERADOR DE MUTACIÓN

El operador de mutación utilizado fue la Mutación de intercambio (Golberg 1989). Se seleccionan dos genes del individuo, basados en una probabilidad aleatoria, y son intercambiados de posición. Este operador es aplicable a codificaciones basadas en el orden y en las secuencias de aminoácidos un ordenamiento define un único péptido.

2.4.5 TÉCNICA DE REMPLAZO

Modelo estacionario: durante cada iteración se escogen dos padres de la población y se les aplican los operadores genéticos definidos. Los descendientes remplazan dos individuos de la población inicial (Golberg 1989).

Remplazo de los peores: de entre un porcentaje de los peores individuos de la población se seleccionan aleatoriamente los dos necesarios para dejar sitio a la descendencia (Golberg 1989).

2.4.6 FUNCIÓN DE EVALUACIÓN

La función de evaluación de cada secuencia candidata **(x)** se define de la siguiente manera:

F(x) = Fa(x) * Fp(x)

Donde **Fa(x)** es la función de actividad de la secuencia candidata y **Fp(x)** es la función de penalización de la secuencia candidata. La función de actividad de las secuencias candidatas se describe a continuación:

Fa(x) = 4 si actividad(x) = Alta, 3 si actividad(x) = Media, 2 si actividad(x) = Baja o 1 si actividad(x) = Ninguna

Donde **actividad(x)** es brindada por el modelo predictivo a utilizar. Árbol de decisión descrito en (Lira, Perez et al. 2013).

La función de penalización se define de la siguiente manera:

- **Fp(x) = (longitud(x) – A) / (C – A) si A < longitud(x) < C**
- **Fp(x) = 1 si C ≤ longitud(x) ≤ D**
- **Fp(x) = (B – longitud(x)) / (B – D) si D < longitud(x) < B**
- **Fp(x) = 0 si longitud(x)=A o longitud(x)=B**

Donde longitud(x) es la cantidad de aminoácidos que componen a la secuencia candidata (x); A y B son los límites mínimos y máximos de longitud de secuencia permitidos respectivamente; C y D los límites mínimos y máximos deseados de longitud de secuencia respectivamente. Los límites son definidos por el usuario antes de comenzar la exploración.

Por ejemplo:

Teniendo dos secuencias (x) y (y):

x: FVPYNPPRPGQSKPFPSFPGHGPFNPKIQWPYPLPNPGH

longitud(x) = 39

y: MKVVIFIFALLATICAAFAYVPLPNVPQPGRRPFPTFPGQGPFNPKIKWPQGY

longitud(y)=53

Con los parámetros:

A = 30, B = 70, C = 50, D = 60

Y actividad:

actividad(x) = actividad(y) = Alta

Entonces:

Fa(x) = Fa(y) = 4

Fp(x) = (39-30)/(50-30) = 0.45

Fp(y) = 1

La función de evaluación total queda definida:

F(x) = 4 * 0.45 = 1.8

F(y) = 4 * 1 = 4

A continuación se muestra la gráfica de la función de penalización para el ejemplo dado:

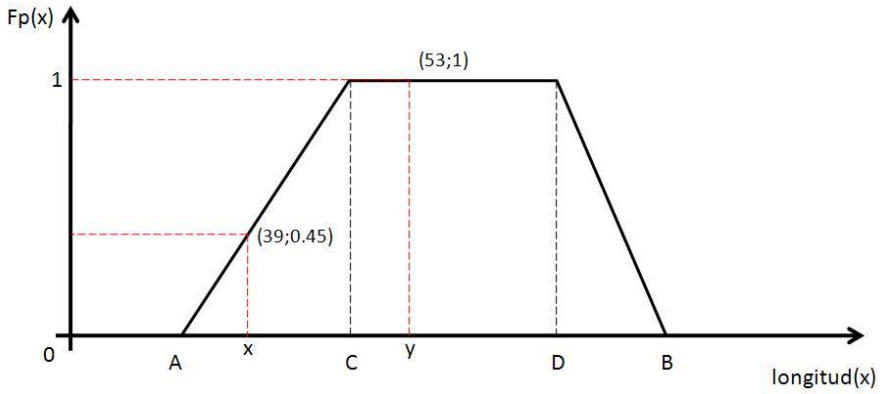

Figura 8: Función de penalización.

2.4.7 SALIDAS DE LA APLICACIÓN

La salida de la aplicación será el conjunto de los individuos de la población final. Las secuencias estarán ordenadas descendentemente atendiendo a la función de evaluación. De allí el investigador decide cuales utilizar en experimentos posteriores, si desea la mejor secuencia solo tiene que trabajar con la primera secuencia del conjunto y así sucesivamente atendiendo a la cantidad de secuencias con las cuales desee experimentar.

2.5 ARQUITECTURA DEL SISTEMA

El sistema se desarrollará completamente sobre la Plataforma del NetBeans, esta plataforma se basa principalmente en la construcción del software mediante la arquitectura basada en componentes, donde cada uno de los componentes tiene definido su propio grupo de funciones; además la plataforma está

conformada por otros componentes ya definidos que son imprescindibles para el correcto funcionamiento. Este estilo proporcionará una alta flexibilidad y escalabilidad al sistema. A continuación se presenta una vista de la arquitectura del sistema.

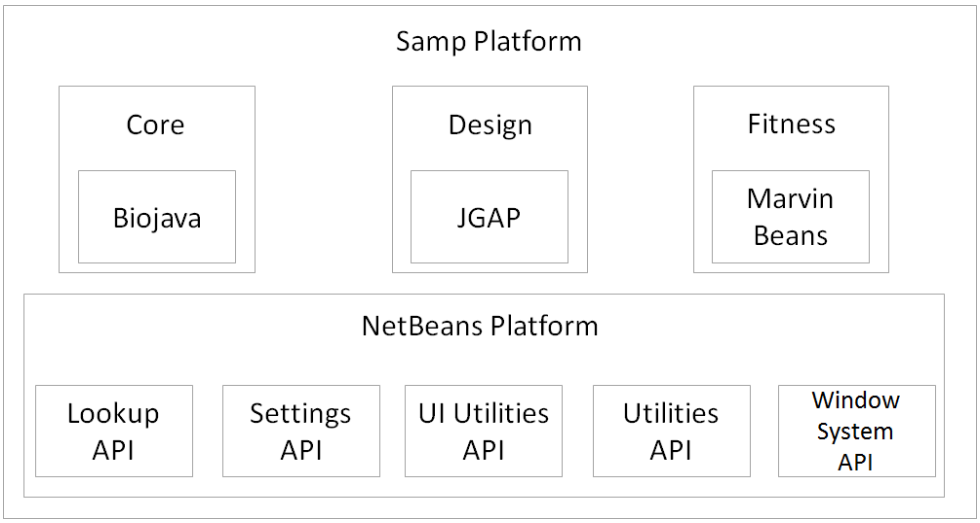

Figura 9: Arquitectura de la plataforma informática Samp.

- **Core:** módulo principal de la plataforma Samp, contiene toda la información necesaria para configurar y ejecutar la exploración de nuevos candidatos a péptidos antimicrobianos. Utiliza la biblioteca Biojava para cargar el fichero de datos con las secuencias de péptidos.
- **Design:** módulo encargado de implementar la estrategia de diseño necesaria en la exploración. La estrategia propuesta para el diseño de las nuevas secuencias es un algoritmo genético el cual estará implementado mediante la biblioteca JGAP.
- **Fitness:** módulo encargado de evaluar los resultados obtenidos en la exploración, nos muestra el indicador de éxito de la estrategia de diseño. La evaluación propuesta se basa en un árbol de decisión en el cual los nodos de decisión son índices dados por descriptores moleculares, los cuales son calculados por las bibliotecas de la suite Marvin Beans.

2.6 ESTRUCTURA DEL MODELO DE DISEÑO

Para el desarrollo del Modelo de Diseño se hace necesario organizar al mismo en: subsistemas y paquetes con sus interfaces y las dependencias entre éstos. Esta representación es muy significativa para la arquitectura en general, debido a que los subsistemas y sus interfaces constituyen la estructura fundamental del producto de software. A continuación se muestran las relaciones y dependencias entre los distintos subsistemas y paquetes.

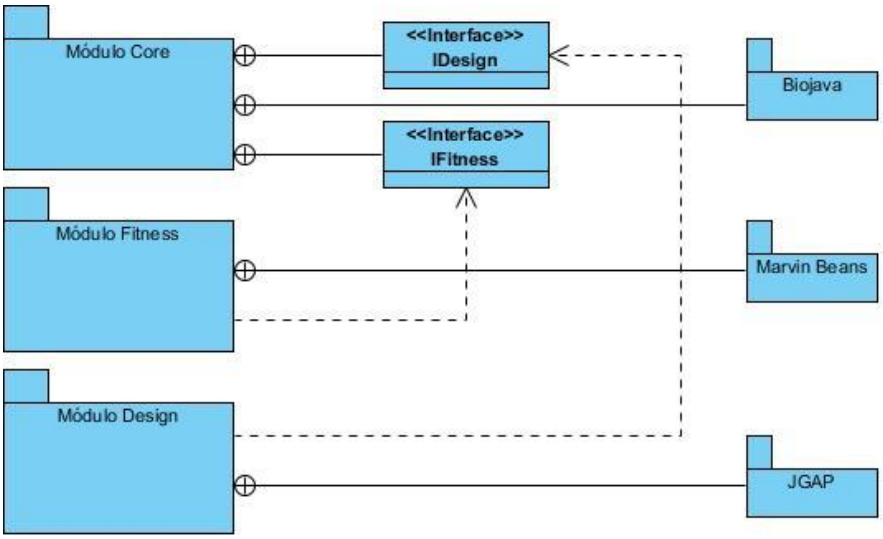

Figura 10: Diagrama de paquetes de la aplicación.

Teniendo en cuenta la importancia de las interfaces para la estructura del software se hace a continuación una descripción de las mismas.

Tabla 6: Descripción de la interfaz IDesign.

Nombre del método	**Descripción**
+**search**(String[] sequences): String[]	Se define para realizar la exploración. A partir de un conjunto inicial de secuencias (sequences) ejecuta su estrategia de diseño y retorna un conjunto final de secuencias.
+**getName**(): String	Se define para obtener el nombre de la técnica empleada para el diseño de las secuencias en la exploración. Puede contener dicho nombre la biblioteca utilizada para la implementación de

	dicha técnica.

Tabla 7: Descripción de la interfaz IFitness.

Nombre del método	Descripción
+**evaluateSequence**(String sequence): double	Se define para evaluar una secuencia y retorna dicha evaluación.
+**sequenceActivity**(String sequence): String	Se define para obtener la actividad de una secuencia determinada y retorna dicha actividad.
+**getName**()	Se define para obtener el nombre de la función de evaluación utilizada. Puede contener más información en dependencia de las necesidades del usuario.

2.7 PATRONES DE DISEÑO

Patrón de diseño de software (Larman 1999). A la hora de desarrollar un Software es importante el uso de Patrones de diseño, existen varios patrones y con distintos propósito entre los que se encuentran los patrones de creación, patrones estructurales, patrones de comportamiento, entre otros.

Proporcionan una estructura conocida por todos los programadores, de manera que la forma de trabajar no resulte distinta entre los mismos. Así la incorporación de un nuevo programador, no requerirá conocimiento de lo realizado anteriormente por otro. Permiten tener una estructura de código común a todos los proyectos que implemente una funcionalidad genérica. La utilización de patrones de diseño, permite ahorrar grandes cantidades de tiempo en la construcción de software.

2.7.1 PATRONES GRASP UTILIZADOS

Los patrones GRASP (Larman 1999) representan los principios básicos de la asignación de responsabilidades a objetos, expresados en forma de patrones. GRASP es el acrónimo para General Responsibility Assignment Software Patterns (Patrones Generales de Software para Asignar Responsabilidades).

Experto: Se encarga de asignar una responsabilidad al experto en información, o sea, aquella clase que cuenta con la información necesaria para cumplir la responsabilidad. En el paquete cu.uci.core.interfaces del módulo Core se encuentran las interfaces IDesign y IFitness las cuales son las responsables de ejecutar la estrategia de diseño y la función de evaluación respectivamente de la exploración.

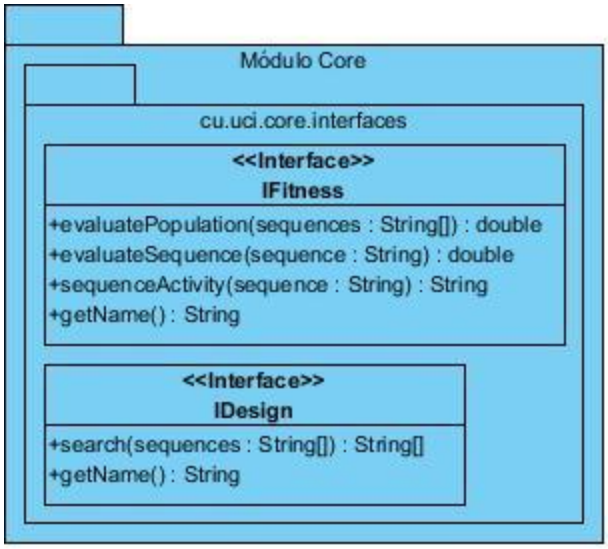

Figura 11: Interfaces del módulo Core.

Controlador: Asigna la responsabilidad de recibir o manejar un mensaje de evento del sistema a una clase. En la aplicación se hace uso de este patrón usando la clase controladora SearchController que tiene la responsabilidad de controlar todos los eventos que se suceden en la exploración.

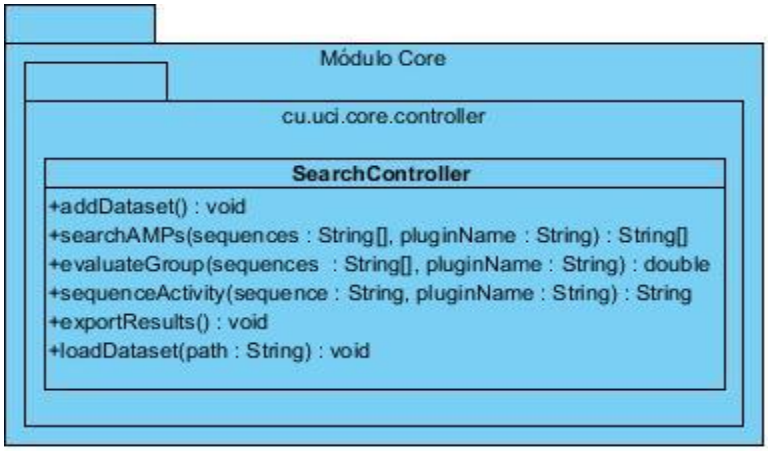

Figura 12: Clase SearchController del paquete cu.uci.core.controller del módulo Core.

Creador: Identifica quién debe ser el responsable de crear o instanciar nuevos objetos de clases. En la aplicación se hace uso de este patrón en las acciones de la plataforma, las cuales crean una instancia de la clase controladora para poder ejecutarse.

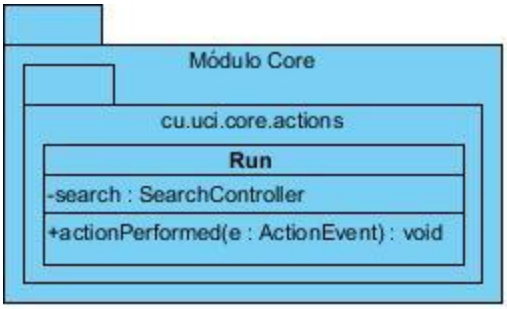

Figura 13: Clase final Run del paquete cu.uci.core.actions del módulo Core.

Alta cohesión: Propone asignar responsabilidades a las clases evitando que estas realicen un trabajo excesivo y que las tareas no sean afines. En la solución se observa el uso de este patrón en la clase SearchController.java ya que no realiza un trabajo excesivo y sus responsabilidades son afines ya que es la encargada de realizar las acciones asociadas a la gestión de la búsqueda.

Bajo Acoplamiento: Este patrón es el encargado de asignar una responsabilidad para conservar bajo acoplamiento. En la aplicación se evidencia dicho patrón en la implementación de la exploración ya que las responsabilidades de diseñar y evaluar las nuevas secuencias son delegadas a módulos independientes del módulo principal. Propone tener las clases con la menor dependencia que se pueda

entre ellas. De tal forma que en caso de producirse una modificación en alguna, se tenga una menor repercusión en el resto de clases, potenciando así la reutilización. En la Figura 15 se muestra la comunicación entre los módulos de la aplicación. Gracias a la biblioteca Lookup que brinda el Netbeans Platform, el modulo principal trabaja con aquellas clases que hallan implementado las interfaces IDesign y IFitness, posibilitando de esta manera que ambos módulos puedan ser remplazados. También pueden ser añadidos nuevos módulos y el usuario solo tendría que seleccionar cuales utilizar en la exploración.

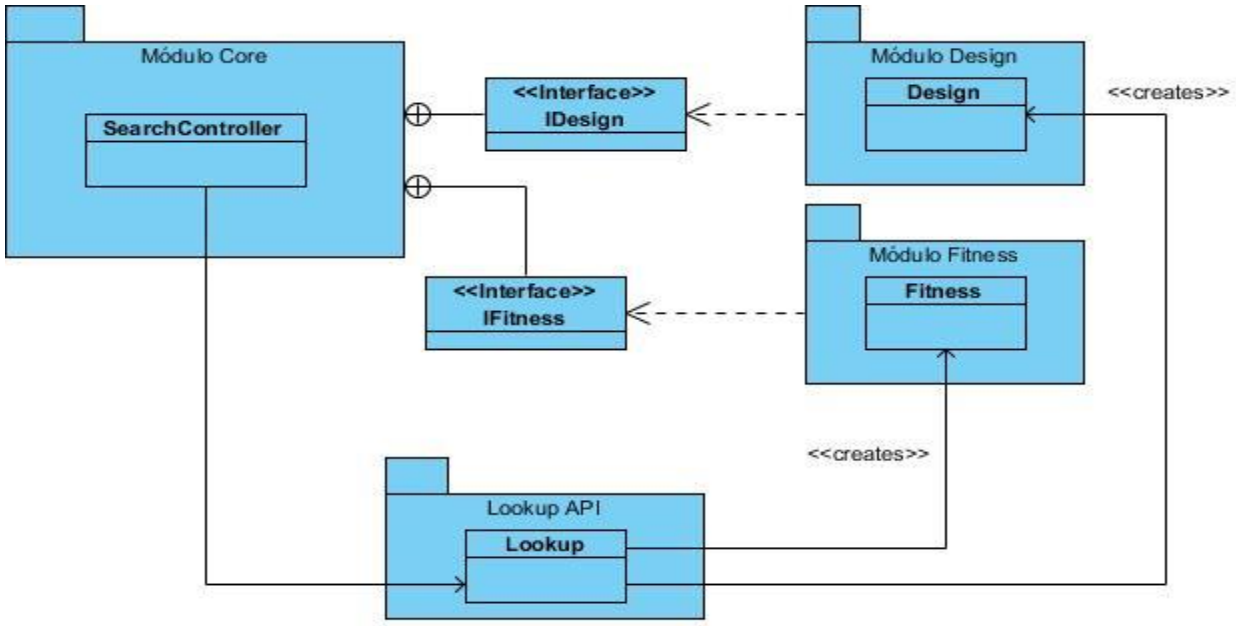

Figura 14: Bajo acoplamiento entre los módulos.

2.7.2 PATRONES GOF UTILIZADOS

Los patrones GOF (Larman 1999) utilizados en la aplicación se describen a continuación.

Inyección de dependencias: comunicación entre componentes de la aplicación mediante interfaces. Se puede ver en la Figura 15. La biblioteca Lookup del Netbeans Platform posibilita la utilización de este patrón pues permite la comunicación entre los módulos de la aplicación mediante interfaces, convirtiendo las mismas en servicios a consumir.

Observer (Observador): Define una dependencia de uno-a-muchos entre objetos, de forma que cuando un objeto cambie de estado se notifique y actualicen automáticamente todos los objetos que dependen de

él. Se muestra en la vista de datos, pues la exploración va a actuar sobre el set de datos que este activo en dicha vista, si cambia el mismo, también cambia la población inicial de la exploración.

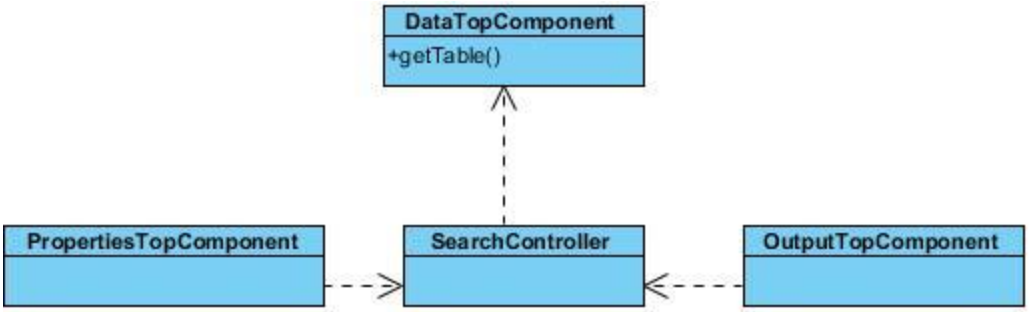

Figura 15: Evidencia del patrón Observer.

2.8 CONCLUSIONES

En este capítulo la confección del diagrama de clases del dominio permitió visualizar la relación existente entre las clases u objetos significativos en el dominio del problema. Con el levantamiento de requisitos se obtuvieron 7 requisitos funcionales agrupados en 4 casos de uso los que fueron descritos textualmente para obtener una visión más detalla del flujo de actividades a implementar en la aplicación. El diseño de los diagramas de casos de uso del sistema brindó la posibilidad de mostrar gráficamente los procesos y su interacción con los actores. La utilización de la arquitectura basada en componentes, permitió separar la implementación de la aplicación en varios módulos, posibilitando que los cambios realizados en uno de ellas no tengan una gran repercusión en el resto. Una vez concluido el diseño de la solución se obtuvo como resultado una serie de diagramas del diseño los cuales permitieron comprender detalladamente las características del sistema.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBAS

En este capítulo, se expondrá el diagrama de componentes, donde se describen los elementos físicos del sistema y sus relaciones. Además se explicarán elementos necesarios para la extensibilidad e implementación de los componentes que formaran parte del proceso de búsqueda. Por último, se describen detalladamente las pruebas que se le realizan al sistema, para corroborar el correcto funcionamiento de la aplicación cumpliendo con los requisitos implementados.

3.1 MODELO DE IMPLEMENTACIÓN

El modelo de implementación (Jacobson, Booch et al. 2000) describe cómo los elementos del modelo de diseño se implementan en términos de componentes como ficheros de código fuente, ejecutables y similares. El modelo de implementación describe también cómo se organizan los componentes de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación y en el lenguaje o los lenguajes de programación utilizados y cómo dependen los componentes unos de otros.

3.2.1 DIAGRAMA DE COMPONENTES

En los diagramas de componentes (Booch, Rumbaugh et al. 1999) se muestran los elementos de diseño de un sistema de software. Un diagrama de componentes permite visualizar con más facilidad la estructura general del sistema y el comportamiento del servicio que estos componentes proporcionan y utilizan a través de las interfaces.

A continuación se muestra el diagrama de componentes el cual muestra la relación entre los componentes físicos de la aplicación. Se puede ver como la clase controladora se encarga de gestionar todo el ciclo de vida de la aplicación, y contiene dependencias hacia los módulos que necesita para cumplir con todas sus responsabilidades. Se evidencia el uso de los patrones descritos en el Capítulo 1 los cuales son de vital importancia para desarrollar buenas prácticas de programación.

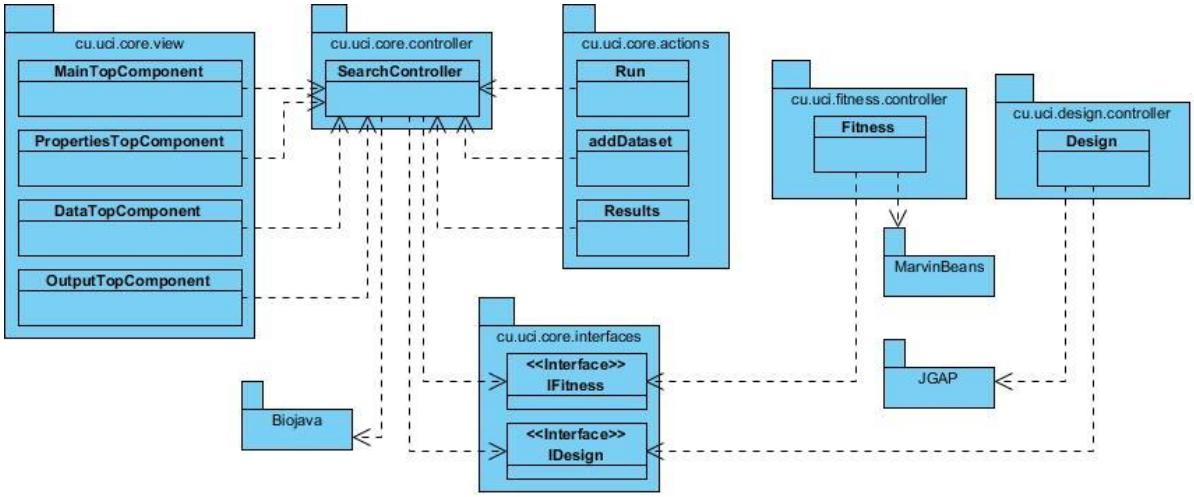

Figura 16: Diagrama de componentes de la aplicación.

3.1.1 ESTÁNDAR DE CODIFICACIÓN

Un estándar de codificación (Microsoft 2015) completo comprende todos los aspectos de la generación de código. Las clases mostradas en el diagrama de componentes tienen nombres afines con sus responsabilidades y comienzan con mayúscula. Por ejemplo la clase Run.java que es la encargada de ejecutar la exploración; la clase SearchController.java que es la encargada de controlar todo el proceso de búsqueda de la aplicación. Las variables también tienen nombres afines con su contenido, por ejemplo las variables *population* de tipo *int* que es la encargada de definir el tamaño de la población; la variable *iterations* de tipo *int* la cual define la cantidad de iteraciones del proceso de búsqueda. Los métodos también tienen nombres afines con sus responsabilidades y comienzan todos con minúscula, por ejemplo el método *searchAMP* el cual es el encargado de ejecutar la búsqueda sobre un conjunto de secuencias y devolver las secuencias resultantes.

3.2 EXTENSIÓN DE LA APLICACIÓN

La arquitectura de la aplicación permite que puedan ser modificados tanto la función de evaluación como la estrategia de diseño para la identificación y clasificación de candidatos a péptidos antimicrobianos. Para la sustitución de estos componentes de la aplicación es necesario tener en cuenta aspectos importantes de guía y apoyo para el futuro enriquecimiento de la misma. La extensión de la aplicación puede verse en

la Figura 14 donde existe muy bajo acoplamiento entre los módulos del sistema. A continuación se describe como añadir nuevos módulos al sistema.

Para añadir nuevos módulos a la aplicación se deben implementar como dependencia del módulo principal: Core. Luego con la ayuda de la librería Lookup API, brindada por la plataforma del NetBeans, se pueden implementar las interfaces para el diseño y evaluación de las secuencias candidatas. La biblioteca Lookup define mediante anotaciones los servicios a implementar por los nuevos módulos. Dichos servicios pueden ser interfaces o clases. En este caso son las interfaces IDesign e IFitness.

3.3 VISTA GENERAL DE LA APLICACIÓN

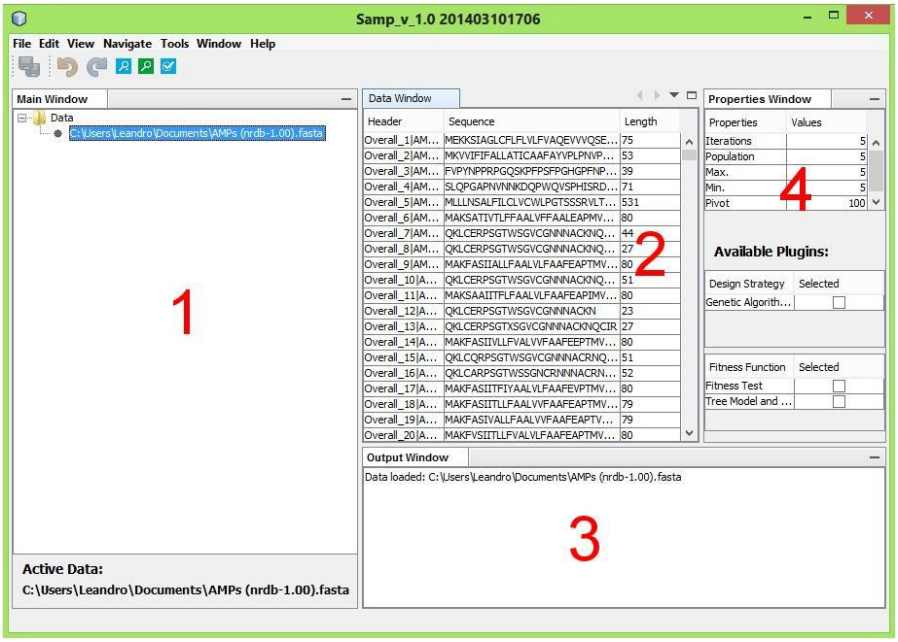

Figura 17: Vista general del sistema.

A continuación se describen los componentes fundamentales de la vista general del sistema.

1. Main Window: muestra la dirección de los datos existentes en el sistema. Cada vez que un nuevo set de datos es agregado al sistema el árbol de direcciones adiciona la nueva dirección al siatema pero no carga el set de datos. El set de datos es cargado una vez seleccionada la dirección del set de datos.

2. Data Window: ventana de datos del sistema, la misma muestra los datos activos con los cuales cuenta el sistema para la ejecución de la búsqueda.
3. Output Window: ventana de salida del sistema, la misma muestra los resultados de la búsqueda.
4. Properties Window: La ventana de propiedades del sistema muestra los parámetros necesarios en la exploración. La cantidad de iteraciones de la estrategia de diseño, la cantidad de secuencias en el grupo inicial de péptidos antimicrobianos, la longitud máxima de las secuencias, la longitud mínima de las secuencias y un elemento pivote, el cual define a partir de cual secuencia del set de datos de entrada se comenzarán a elegir los individuos del grupo inicial. También muestra los módulos disponibles en el sistema para la exploración.

3.4 PRUEBAS DE CALIDAD DEL ALGORITMO GENÉTICO

Eficacia

La eficacia es la capacidad de alcanzar el efecto que se desea tras la realización de una acción.

Se realizaron pruebas de calidad al algoritmo genético teniendo en cuenta los resultados obtenidos por la aplicación. Se determina que el algoritmo genético es eficaz pues logra ir aumentando la evaluación de los individuos por iteraciones. En la siguiente gráfica se muestra el porcentaje de los peores individuos existentes en la población por iteraciones. Debido a que la técnica de remplazo implementada es la sustitución de los peores, dicho por ciento no puede aumentar.

Se puede ver como las secuencias de actividad baja van disminuyendo a medida que va iterando el algoritmo genético y van aumentando las secuencias con mayores actividades. Esto muestra que el algoritmo implementado va maximizando la actividad de las secuencias, por lo que se puede decir que es eficaz por cumplir con el objetivo planteado. El ejemplo mostrado se realizó para una población de 10 secuencias de longitud entre 10 y 20 aminoácidos.

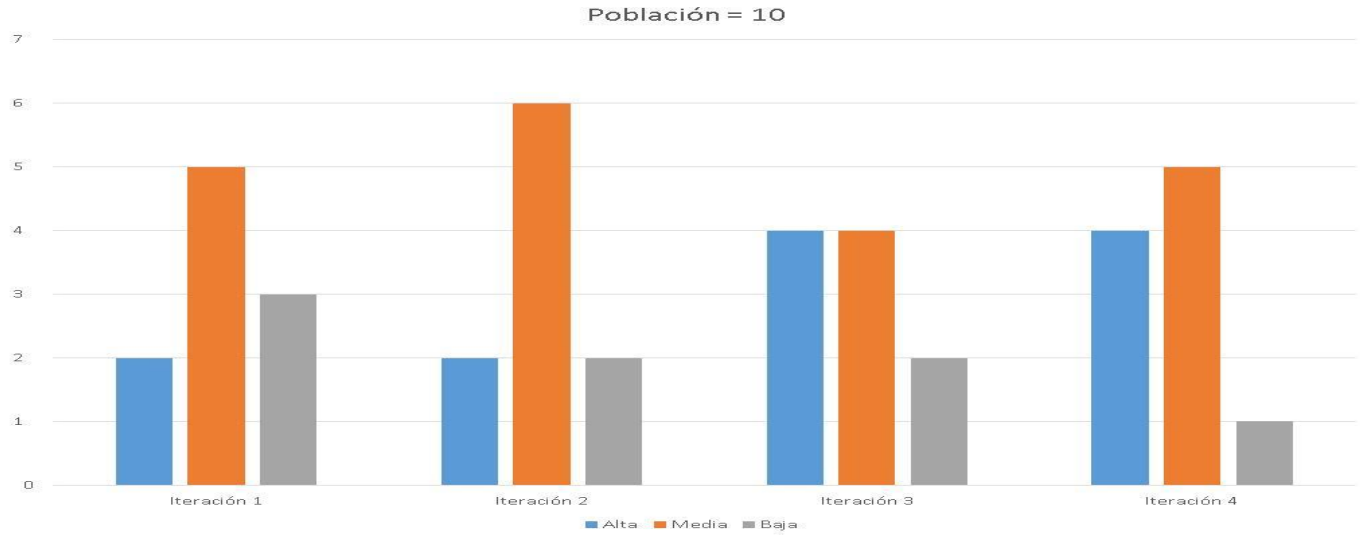

Figura 18: Resultados de la aplicación.

3.5 PRUEBAS DE CALIDAD DEL SOFTWARE

Son los procesos que permiten verificar y revelar la calidad de un producto software (EcuRed 2015). Son utilizadas para identificar posibles fallos de implementación, calidad, o usabilidad de un programa de ordenador o videojuego. Básicamente es una fase en el desarrollo de software consistente en probar las aplicaciones construidas.

Las pruebas de software se integran dentro de las diferentes fases del ciclo del software dentro de la Ingeniería de software. Así se ejecuta un programa y mediante técnicas experimentales se trata de descubrir que errores tiene.

3.5.1 DISEÑO DE PRUEBA DE CAJA NEGRA

Estas pruebas (EcuRed 2015) permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. En ellas se ignora la estructura de control, concentrándose en los requisitos funcionales del sistema y ejercitándolos.

Para preparar los casos de pruebas hacen falta un número de datos que ayuden a la ejecución de los estos casos y que permitan que el sistema se ejecute en todas sus variantes, pueden ser datos válidos o inválidos para el programa según si lo que se desea es hallar un error o probar una funcionalidad. Los

datos se escogen atendiendo a las especificaciones del problema, sin importar los detalles internos del programa, a fin de verificar que el programa corra bien.

Tabla 8: Caso de prueba para el caso de uso Adicionar fichero de datos

Nombre de la sección	Escenario	Descripción de la funcionalidad	Flujo central
SC1 Adicionar fichero de datos.	EC1.1 Se adiciona el fichero al sistema.	El investigador debe seleccionar el fichero de datos en su ordenador.	1. En la barra de menú seleccionar File/Add Dataset… 2. Buscar el fichero de datos en el ordenador. 3. Confirmar su selección.

Tabla 9: Caso de prueba para el caso de uso Exportar resultados obtenidos.

Nombre de la sección	Escenario	Descripción de la funcionalidad	Flujo central
SC4 Exportar resultados obtenidos.	EC4.1 Se adiciona el fichero al sistema.	El investigador debe seleccionar el fichero de datos en su ordenador.	1. En la barra de menú seleccionar File/Export Results… 2. Buscar el fichero de datos en el ordenador. 3. Confirmar su destino.

Tabla 10: Caso de prueba para el caso de uso Configurar la búsqueda.

Nombre de la sección	Escenario	Descripción de la funcionalidad	Flujo central
SC2 Configurar la búsqueda.	EC2.1 La búsqueda queda configurada.	El investigador debe seleccionar la cantidad de iteraciones, la población inicial, y los límites de longitud de las secuencias. Así como seleccionar los módulos de exploración.	1. Ir a la ventana de propiedades del sistema. 2. Establecer los parámetros de configuración.

Tabla 11: Caso de prueba para el caso de uso Buscar nuevos péptidos antimicrobianos.

Nombre de la sección	Escenario	Descripción de la funcionalidad	Flujo central
SC3 Buscar nuevos candidatos a péptidos antimicrobianos.	EC3.1 Se obtienen nuevos candidatos a péptidos antimicrobianos.	El investigador ejecuta la búsqueda confirmando los parámetros de configuración establecidos.	1. En la barra de menú seleccionar File/Run Search… 2. Confirmar parámetros de configuración.

Resultados de las pruebas de caja negra

Después de realizar las pruebas de caja negra mediante los casos de prueba asociados a cada caso de uso, se comprobó el correcto funcionamiento de la aplicación, verificando que cada uno de sus requisitos se ejecutara correctamente.

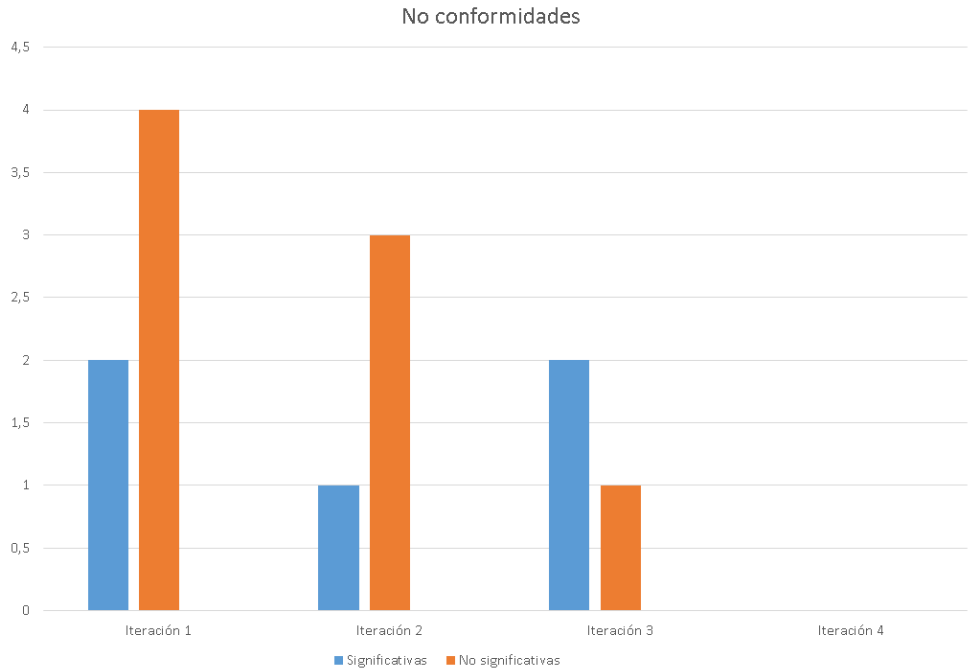

Figura 19: Gráfico de la aplicación de los casos de prueba por iteraciones.

3.5.2 PRUEBAS DE INTEGRACIÓN

Las Pruebas de Integración son aquellas que permiten probar en conjunto distintos subsistemas funcionales o componentes del proyecto para verificar que interactúan de manera correcta y que se ajustan a los requisitos especificados. Este tipo de pruebas deberán ejecutarse una vez se haya asegurado el funcionamiento correcto de cada componente implicado por separado, es decir, una vez se hayan ejecutado sin errores las pruebas unitarias de estos componentes.

Ejecución de las pruebas de integración

Se realizaron cuatro iteraciones de pruebas funcionales aplicándose los 4 casos de prueba diseñados y se detectaron no conformidades que fueron mitigadas durante las iteraciones. Esto demuestra que el sistema está libre de errores.

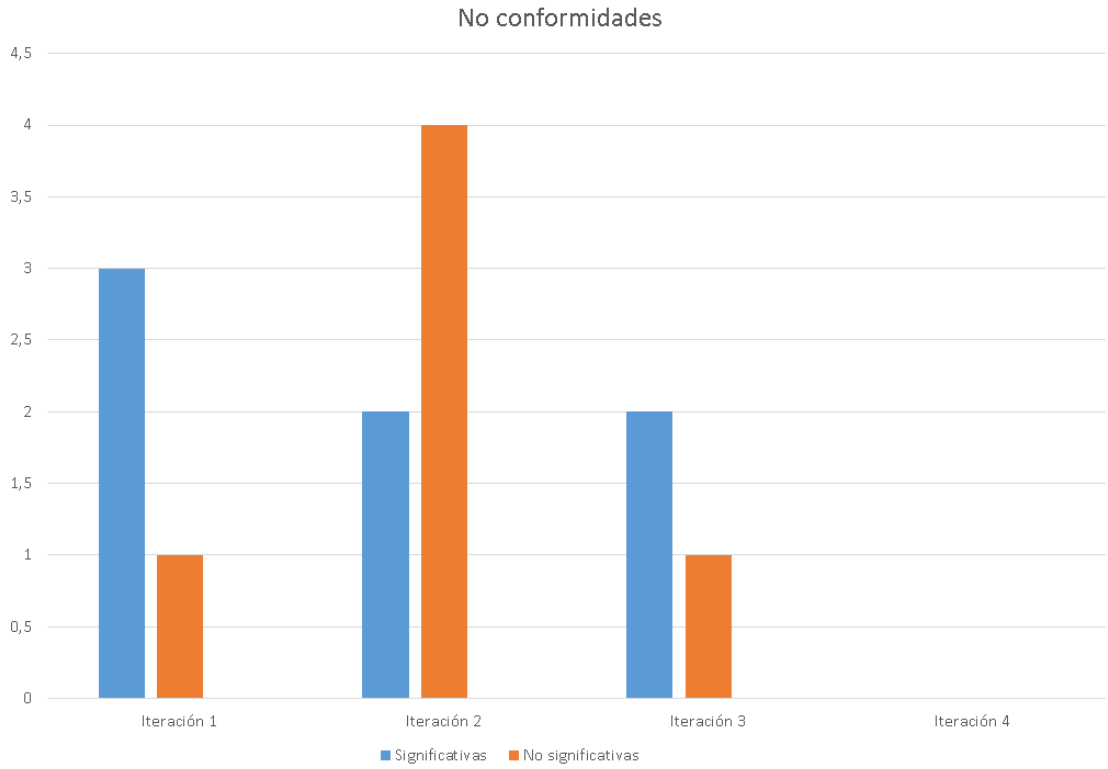

Figura 20: Gráfico de la aplicación de las pruebas de integración por iteraciones.

3.6 CONCLUSIONES

La aplicación obtenida cumple con un correcto estándar de codificación y con buenas prácticas de programación. Los componentes físicos del sistema cumplen con una correcta arquitectura que permite la extensibilidad de la aplicación mediante la obtención de un bajo acoplamiento entre sus módulos. Las pruebas realizadas a la aplicación lograron corregir los errores presentes en la misma y demostraron la correcta integración de los componentes del sistema. La aplicación cuenta con una interfaz agradable al usuario y cuenta con un fácil acceso a las funcionalidades de la misma.

CONCLUSIONES GENERALES

El desarrollo del presente trabajo se transitó por una serie de etapas que permitieron dar cumplimiento al objetivo planteado, arribando a las siguientes conclusiones:

- Se obtiene una herramienta informática basada en la plataforma del NetBeans capaz de utilizar un enfoque heurístico para explorar el espacio de los posibles ordenamientos de aminoácidos y un modelo predictivo para determinar la evaluación de las secuencias candidatas. Se implementa un algoritmo genético para explorar el espacio combinatorio de posibles secuencias de aminoácidos y obtener así las secuencias candidatas. Durante la exploración y evaluación en el espacio de búsqueda se utilizó un árbol de decisión, reportado en la literatura, para predecir la actividad biológica y guiar la búsqueda hacia regiones factibles. La herramienta informática obtenida puede ser extendida en forma de módulos para incorporar otras estrategias de exploración y evaluación, por lo que puede convertirse en un marco de trabajo para la búsqueda informada de candidatos a péptidos antimicrobianos.
- El algoritmo genético implementado cuenta con las siguientes características: codificación basada en la representación en una letra de los 20 aminoácidos más comunes a encontrar en un péptido; cruce en un punto como operador de cruzamiento; mutación de intercambio como operador de cruce; modelo estacionario como técnica de remplazo basada en la sustitución de los peores; como función de evaluación se diseña una fórmula que depende de la longitud de las secuencias candidatas y la actividad de las mismas obtenidas a partir de un árbol de decisión que responde a las características de un modelo QSAR mediante el cálculo de descriptores moleculares de las secuencias candidatas.
- Con el diseño y ejecución de las pruebas a nivel de desarrollador y de integración, se logró validar el correcto funcionamiento del sistema para la identificación y clasificación de nuevos candidatos a péptidos antimicrobianos mediante métodos computacionales basados en algoritmos genéticos.

RECOMENDACIONES

1. Probar experimentalmente los resultados de la aplicación.
2. Continuar con el desarrollo de módulos para la identificación y clasificación de péptidos antimicrobianos. Como nuevas estrategias de diseño se pueden agregar a la aplicación módulos que implementen nuevas metaheurísticas tales como la optimización colonia de hormigas. Como nuevas funciones de evaluación se pueden agregar a la aplicación módulos que implementen modelos de predicción QSAR tales como redes neuronales artificiales.

REFERENCIAS BIBLIOGRÁFICAS

Adams, K. (1996). "Ambicin (purified nisin) metaphase chromosome analysis of human lymphocytes cultured in vitro. Final report (APM 2/952601) from Huntingdon Life Sciences Ltd, Huntingdon, Cambridgeshire, England, United Kingdom." Unpublished study submitted to WHO by DuPont Nutrition and Health, USA, and the Ministry of Health, Labour and Welfare, Japan.

Aguilar, J. and F. Rivas (2001). "Computación Inteligente." MERITEC, June.

Böck, H. (2011). The Definitive Guide to NetBeans™ Platform 7, Apress.

Booch, G., J. Rumbaugh, et al. (1999). El lenguaje unificado de modelado, Addison-Wesley.

Buhr, R. J., R. S. Casselman, et al. (1996). "Use case maps for object-oriented systems."

Canós, J., P. Letelier, et al. (2003). "Metodologías Ágiles en el desarrollo de Software." Universidad Politécnica de Valencia, Valencia.

EcuRed (2015). "Pruebas de caja negra." http://www.ecured.cu/index.php/Pruebas_de_caja_negra.

EcuRed (2015). "Pruebas de Calidad de Software." http://www.ecured.cu/index.php/Pruebas_de_Calidad_de_Software.

Fjell, C. D., J. A. Hiss, et al. (2012). "Designing antimicrobial peptides: form follows function." Nature reviews Drug discovery **11**(1): 37-51.

Fjell, C. D., H. Jenssen, et al. (2011). "Optimization of antibacterial peptides by genetic algorithms and cheminformatics." Chemical biology & drug design **77**(1): 48-56.

Flores, C. L. T. and G. H. A. Salinas "Establecimiento de una Metodología de Desarrollo de Software para la Universidad de Navojoa Usando OpenUP."

Fuentes, L. and A. Vallecillo (2004). "Una introducción a los perfiles UML." Revista Novatica–Asociación de Técnicos de Informática-España.

Giraldo, F. A. G. and J. G. Perdomo (2013). "Aprendizaje de estrategias de decisión en juegos repetitivos no cooperativos." Tecnura **17**(35): 63-76.

Golberg, D. E. (1989). "Genetic algorithms in search, optimization, and machine learning." Addion wesley **1989**.

Groussard, T. (2012). JAVA 7: Los fundamentos del lenguaje Java, Ediciones ENI.

Hiss, J. A., A. Bredenbeck, et al. (2007). "Design of MHC I stabilizing peptides by agent-based exploration of sequence space." Protein Engineering Design and Selection **20**(3): 99-108.

Hoffmann, J. M. (2008). "Rich Client Platform."

Holland, R. C., T. A. Down, et al. (2008). "BioJava: an open-source framework for bioinformatics." Bioinformatics **24**(18): 2096-2097.

Izaurieta, F. and C. Saavedra (2000). "Redes neuronales artificiales." Departamento de Física, Universidad de Concepción Chile.

Jacobson, I., G. Booch, et al. (2000). El proceso unificado de desarrollo de software, Addison Wesley Reading.

Larman, C. (1999). UML y Patrones, Pearson.

Lira, F., P. S. Perez, et al. (2013). "Prediction of antimicrobial activity of synthetic peptides by a decision tree model." Applied and environmental microbiology **79**(10): 3156-3159.

Martínez, A. and R. Martínez (2002). "Guía a Rational Unified Process." Escuela Politécnica Superior de Albacete–Universidad de Castilla la Mancha.

Meffert, K., J. Meseguer, et al. (2011). "JGAP—Java Genetic Algorithms Package." URL http://jgap.sourceforge.net/.—abgerufen am **21**(08).
Microsoft (2015). "Revisiones de código y estándares de codificación." https://msdn.microsoft.com/es-es/library/aa291591%28v=vs.71%29.aspx.
Millán, M. E. (2011). "Comparación de métodos de búsqueda."
Moskowitz, H., G. P. Wright, et al. (1982). Investigación de operaciones, Prentice Hall.
Porto, W. F., O. L. Franco, et al. (2012). Prediction and rational design of antimicrobial peptides, INTECH Open Access Publisher.
Reyes, A. J. O. and A. O. García (2012). "Vista de análisis usando técnicas de agrupamiento para el sistema integral para la atención primaria de salud." Serie Científica **5**(10).
Schneider, G. and P. Wrede (1994). "The rational design of amino acid sequences by artificial neural networks and simulated molecular evolution: de novo design of an idealized leader peptidase cleavage site." Biophysical Journal **66**(2 Pt 1): 335.
Todeschini, R. and V. Consonni (2008). Handbook of molecular descriptors, John Wiley & Sons.
Todeschini, R. and V. Consonni (2009). Molecular Descriptors for Chemoinformatics, Volume 41 (2 Volume Set), John Wiley & Sons.
VALENCIA, E. (1997). Optimización mediante algoritmos genéticos. Anales del Instituto de Ingenieros de Chile.
Weber, L. (2008). JChem Base-ChemAxon, ROYAL SOC CHEMISTRY THOMAS GRAHAM HOUSE, SCIENCE PARK, MILTON RD, CAMBRIDGE CB4 0WF, CAMBS, ENGLAND. **5:** 65-66.