UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 6

INSTITUTO DE CIENCIA ANIMAL

APLICACIÓN PARA LA CLASIFICACIÓN TAXONÓMICA POR HOMOLOGÍA DE SECUENCIAS BASADA EN DISTANCIA DE EDICIÓN ESTOCÁSTICA.

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

Autores:

Raúl Dilnier Gamboa López.

Manuel Alejandro Cora González.

Tutores:

MSc. Abiel Roche Lima.

MSc. Mario Pupo Meriño.

Cuidad de La Habana Junio 2009. “Año del 50 aniversario del triunfo de la Revolución”

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis 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 _____.

Raúl Dilnier Gamboa López

_____

Firma del Autor

MSc. Abiel Roche Lima.

_____

Firma del Tutor

Manuel Alejandro Cora González.

_____

Firma del Autor

MSc. Abiel Roche Lima.

_____

Firma del Tutor

DATOS DE CONTACTO

MSc. Mario Pupo Meriño.
Licenciado en Bioquímica, UH, Máster en Matemática Aplicada, UCLV, Instructor, 5 años de experiencia. Departamento de Bioinformática.

Teléfono:
Departamento: 837 25 53
Particular: 202 74 12

Email: mpupom@uci.cu
mpupo@uclv.edu.cu

"La verdadera dificultad, la que ha decepcionado a los sabios de todos los tiempos es esta: cómo hacer de la educación algo lo suficientemente poderoso en la vida para que su influencia resista la presión de las fuerzas psíquicas elementales del individuo."

Albert Einstein.

AGRADECIMIENTOS

A mis padres y hermanos, por el amor, la confianza y su apoyo constante.

A mis abuelos, por todos los consejos y el cariño brindado.

A mis amigos y compañeros de estudio, por todos los momentos compartidos, que ahora forman parte de los mejores recuerdos de esta etapa de la vida.

A Yunet, por su comprensión, amor y cariño en todos estos años.

A mis tutores Abiel Roche Lima y Mario Pupo por su dedicación y esfuerzo.

A todos los profesores de la Universidad de las Ciencias Informáticas, por todo su empeño en nuestra formación como profesionales.

A la Revolución, por esta oportunidad maravillosa, que sin duda siempre es y será el sueño de muchos.

Raúl Dilnier Gamboa López.

A mis tutores Abiel por la oportunidad de compartir con él este proyecto y Mario por el interés mostrado en cuanto supo de su existencia, a ambos por la ayuda y el apoyo constante.

A mis amigos que siempre estuvieron junto a mí y me ayudaron tanto como les fue posible.

A Daryanis por todos los momentos felices, por quererme tanto y por brindarme su apoyo constante.

A mi familia, mis tías Osi, Nory, Lola, Rosa, a Elizabeth (las dos) y Abelardo por ser también parte de ella.

A mis padres Chely y Raúl...

Alejandro González

DEDICATORIA

A mi padre, Raúl Gamboa González, por ser la persona que más admiro, por ser ejemplo de padre y sacrificio, por su apoyo, cariño y confianza. A ti padre, por estar siempre.

A mi madre, Eyda López Moures, por su cariño incondicional, por toda su dedicación y esmero, por su preocupación. Por ser mi mejor amiga, mi conciencia, mi ángel de la guarda, porque me has llenado de amor y valentía, a ti por ser la mejor madre.

A mi hermano menor, Enrique Gamboa López, por motivar mis esfuerzos, por ayudarme a ser su ejemplo a seguir, por su compañía, por ser además de un hermano un ejemplo de amigo.

A ustedes por ser mis personajes principales, por ser mi fuente de inspiración.

Raúl Dilnier Gamboa López.

A mi mamá y mi papá por todo el cariño, su eterna preocupación, su apoyo constante, por todo el esfuerzo realizado, por quererme y ayudarme tanto, por confiar en mí y permitirme seguir mis sueños con independencia. Mami, te agradezco por tu ejemplo. Papi, un padre mejor no lo hubiera podido pedir.

Los quiero.

Alejandro González.

RESUMEN

Actualmente existen herramientas informáticas que permiten el alineamiento de secuencias biológicas (ácidos nucleídos, genes y proteínas), representadas como cadenas de caracteres. La base de estas aplicaciones es la utilización de algoritmos de distancia de edición. El Instituto de Ciencia Animal (ICA) carece de soluciones informáticas para la clasificación de microorganismos del rumen, empleando la información obtenida a partir de las tecnologías de secuenciación genética incorporadas en el mismo.

El presente trabajo de diploma propone una aplicación de escritorio que permite realizar clasificación taxonómica por homología de secuencias usando distancia de edición estocástica, de los microorganismos presentes en las comunidades microbianas; brindando funcionalidades que permiten la clasificación de microorganismos no cultivados. Dentro de estas, se destaca la realización de un aprendizaje de los costos de las operaciones de edición a partir de una Base de datos de pares de cadenas que se obtiene de una base de datos de microorganismos ya clasificados, de forma tal que la obtención de los posibles candidatos sea más exacta que usando matrices de costos unitarios o prefijadas por el usuario. Para ello se realizó este proceso utilizando distribuciones de probabilidad y transductores de estado finito estocástico. La aplicación se realizó siguiendo la metodología de desarrollo de software OpenUP, abarcando cada una de las fases e iteraciones que la misma propone.

Palabras claves: clasificación taxonómica, distancia de edición estocástica, transductores.

TABLA DE CONTENIDOS

INTRODUCCIÓN

A medida que la sociedad evoluciona va siendo dominada con una mayor fuerza por la tecnología, y en particular por las Tecnologías de la Información y las Comunicaciones. Hoy la información es el recurso clave en las diferentes ramas o sectores sociales como la economía, la cultura y la política, el tratamiento adecuado de esta a través de los sistemas computarizados o de software, permite el enriquecimiento y desarrollo de los mismos y mejora la calidad del pensamiento humano.

Desde la aparición del ordenador en la década del 50 su aplicación en el estudio de los seres vivos ha revolucionado la Biología. La fusión entre Biología e Informática ha dado como resultado una nueva disciplina, la Bioinformática, donde la aplicación de los sistemas informáticos en el análisis, modelado y simulación de las estructuras y fenómenos observados en los seres vivos ha conducido a resultados que por su impacto científico, médico y social determinarán en gran medida los avances tecnológicos que sin duda estarán entre los más espectaculares del siglo XXI.

Uno de los logros tecnológicos más significativos en los últimos tiempos ha sido el desarrollo de métodos de secuenciación de genes y proteínas. Esto ha permitido la creación de grandes bancos de datos de ácidos nucléicos (ADN, ARN), aminoácidos, genes y proteínas que pueden ser expresados como cadenas de caracteres. En este contexto la *International Nucleotide Sequence Database Collaboration* que está integrada por la base de datos de ADN de Japón (*DNA DataBank of Japan (DDBJ)*), el Laboratorio Europeo de Biología Molecular (*European Molecular Biology Laboratory (EMBL)*), y el GenBank en el *National Center for Biotechnology Information (NCBI)*, son las grandes bases de datos del mundo que almacenan secuencias genéticas. GenBank y sus colaboradores reciben secuencias genéticas producidas en laboratorios de todo el mundo, procedentes de más de 100.000 organismos distintos.

Con la existencia de estas grandes bases de datos de secuencia genéticas y su rápida expansión a ritmo exponencial se hizo inminente el desarrollo de herramientas

informáticas que permitieran brindar un sólido apoyo para manejar y procesar de manera eficiente la masiva cantidad de información biológica que es generada y almacenada en la actualidad. Con este fin, han sido desarrollados diferentes sistemas informáticos que permiten la búsqueda de similitud, clasificación, alineamientos y en general el análisis de secuencias y bases de datos biológicas, siendo el NCBI uno de los centros que ofrece a través de la Web, herramientas con este propósito.

El Instituto de Ciencia Animal (ICA) de la Habana, es el centro del país que se dedica al estudio de alternativas alimentarias, manejo y genética animal para Cuba y el área tropical. Con esta meta, el grupo de investigadores de este centro necesitan estudiar e inferir las funcionalidades de las comunidades microbianas presentes en el Rumen.

Aún con todas las tecnologías y herramientas que existen, muchas de las cuales están publicadas y son accesibles desde internet, no es posible este análisis, debido a que no hay completa seguridad que los microorganismos presentes en las base de datos existentes tienen las características de los que se encuentran en la zona tropical y específicamente en nuestro país y por tanto cualquier clasificación e inferencia de funcionalidad sería probablemente errónea.

Otro de los inconvenientes de los sitios que ofrecen estas herramientas es que no se permiten hacer búsquedas masivas, no se pueden personalizar las bases de datos, ni las matrices de costos, y las secuencias son enviadas al servidor sin ningún tipo de cifrado, lo que puede representar un problema para quienes quieran mantener sus datos privados.

Es por lo antes mencionado que en conjunto con el máster Abiel Roche Lima jefe del grupo de informatización del ICA y como parte de su tesis de doctorado “Algoritmos de Distancia de Edición Estocásticos aplicados a la Bioinformática”, se propuso la creación de una aplicación que contribuya a la clasificación taxonómica e inferencia de funcionalidades de las comunidades microbianas, con el objetivo de facilitar el trabajo y las investigaciones a los especialistas de este instituto. Esta aplicación estará basada

en algoritmos de aprendizaje y Distancia de Edición Estocástica, de los que se ha demostrado su eficiencia en el cálculo de similitud entre dos secuencias.

Tomando como motivación los aspectos tratados anteriormente y siguiendo la metodología de desarrollo OpenUP, el presente trabajo de diploma propone una aplicación de escritorio para la clasificación taxonómica por homología de secuencias, planteándose el siguiente ***problema científico***: ¿Cómo realizar clasificación taxonómica por homología de secuencias usando distancia de edición estocástica?

El problema planteado se enmarca en el ***objeto de estudio***: Algoritmos de distancia de edición estocástica.

El objeto de estudio delimita el ***campo de acción***: Algoritmos de distancia de edición estocástica aplicados a la clasificación taxonómica por homología de secuencias.

Para dar solución al problema se define como ***objetivo general***: Desarrollar una aplicación que permita la clasificación taxonómica por homología de secuencias, usando distancia de edición estocástica.

A partir del análisis del objetivo general se derivaron los siguientes ***objetivos específicos***:

- ✓ Analizar el dominio del problema.
- ✓ Definir las funcionalidades de la aplicación.
- ✓ Diseñar la aplicación.
- ✓ Implementar la aplicación.
- ✓ Realizar pruebas.

Para dar cumplimiento con los objetivos planteados, se propone la realización de las siguientes ***tareas***:

- ✓ Estudio de las tendencias y tecnologías actuales a considerar para realizar la aplicación.
- ✓ Realización del modelo de dominio.
- ✓ Realización de las actividades del flujo de trabajo de Requerimientos.
- ✓ Realización de las actividades del flujo de trabajo de Diseño.
- ✓ Realización de las actividades del flujo de trabajo de Implementación.
- ✓ Realización de pruebas de funcionalidad.

La estructura del presente trabajo está conformada por cuatro capítulos, los cuales se mencionan a continuación:

El Capítulo I. **Fundamentación Teórica.**

Este capítulo ofrece una reseña de las diferentes aplicaciones informáticas existentes en el mundo aplicado al área de investigación. Los principales conceptos y términos empleados en el estudio de los algoritmos propuestos, así como las tecnologías, las técnicas y los demás aspectos a considerar para lograr la realización de un sistema informático exitoso y con la calidad requerida.

El Capítulo II. **Características del Sistema.**

En este capítulo se muestran los principales requisitos del sistema propuesto, tanto funcionales, como no funcionales y el modelado del sistema, con las descripciones de los casos de usos presentes en el mismo.

El capítulo III. **Diseño del Sistema.**

Este capítulo describe el diseño de la aplicación, mostrándose el estilo arquitectónico utilizado, los principales patrones de diseño, los diagramas de clases e interacción y el diseño de la BD, además de ejemplificar el tratamiento de errores.

El capítulo IV. Implementación y Prueba.

Describe los aspectos relacionados con la implementación y prueba de la solución propuesta. Teniendo en cuenta diagrama de componentes, Algoritmos principales para la solución y los casos de prueba de funcionalidad, específicamente pruebas de caja negra con la técnica de partición de equivalencia.

CAPÍTULO I. FUNDAMENTACIÓN TEÓRICA.

En este capítulo se ofrece una explicación de los conceptos y fundamentos teóricos necesarios para lo total comprensión del software y el funcionamiento de sus principales algoritmos. Se analizan las aplicaciones y algoritmos que existen para la clasificación y alineamiento de secuencias. Se fundamentan las tendencias, técnicas, tecnologías, metodologías y software usados para la solución del problema.

1.1 CLASIFICACIÓN TAXONÓMICA.

La taxonomía es, en su sentido más general, la ciencia de la clasificación. Habitualmente, se emplea el término para designar a la taxonomía biológica, la ciencia de ordenar a los organismos en un sistema de clasificación compuesto por una jerarquía de taxones anidados.

Es una subdisciplina de la Biología Sistemática que estudia las relaciones de parentesco entre los organismos y su historia evolutiva. Actualmente, la Taxonomía actúa después de haberse resuelto el árbol filogenético de los organismos estudiados, esto es, una vez que están resueltos los clados, o ramas evolutivas, en función de las relaciones de parentesco entre ellos. Una de las formas de realizar la clasificación, es a partir del establecimiento de la homología de secuencias protéicas ó de ácidos nucléicos a partir del cálculo de similitud entre ellas.

1.2 ANÁLISIS DE SECUENCIAS.

Cuando se analizan secuencias es común utilizar los términos similitud y homología de forma indiscriminada, pero estos dos términos hacen referencia a conceptos distintos.

SIMILITUD: Es el resultado del análisis (observación cuantitativa) de la estructura primaria de dos o más secuencias (ácidos nucléicos o proteínas) y cuantifica los rasgos compartidos entre ellas.

HOMOLOGÍA: La homología es una medida cualitativa entre las secuencias, se presenta cuando la similitud que estas tienen, es atribuible a razones evolutivas y no al azar, es decir, la homología establece regiones entre las secuencias que se han conservado con el tiempo.

La similitud es el resultado de una medida cuantitativa, la homología es una hipótesis postulada por el investigador, basándose en la similitud de las secuencias y en otros datos biológicos que previamente conozca, sobre el origen de las mismas. Es permitido establecer el porcentaje de similitud de dos o más secuencias, pero esto no es posible para la homología: las secuencias son o no son homólogas.

1.3 CÁLCULO DE SIMILITUD.

El problema de computar la similitud entre dos secuencias se levanta en muchas áreas como la biología computacional, el proceso del idioma natural y los correctores ortográficos. Las estrategias existentes en la actualidad para calcular esta similitud son mediante el alineamiento de secuencias ó mediante el cálculo de su distancia de edición. Dichas estrategias se diferencian en que los alineamientos tienen como objetivo maximizar la cantidad de coincidencias entre una cadena y otra, mientras que, los algoritmos de distancia de edición tratan de minimizar la cantidad de operaciones de edición.

1.3.1 DISTANCIA DE EDICIÓN.

La distancia de edición entre dos cadenas finitas *x*, *y,* es el valor de menor costo al convertir la cadena x en *y*, a partir de la inserción, eliminación o substitución de símbolos simples, según la definición de Levenshtein en 1966 [1]. Este cálculo, da una medida de la similitud entre dichas cadenas.

La distancia de edición está caracterizada por una tripla (**X, Y, C_c**) consistente de alfabetos finitos **(X**, **Y)** y la función de costo primitiva $\mathbf{C_c: E \rightarrow R^+}$ donde $\mathbf{E = E_s \cup E_d \cup E_i}$ es

el alfabeto de operaciones de edición primitivas, $\mathbf{E_s = X \times Y}$ es el conjunto de substituciones, $\mathbf{E_d = X \times \{\lambda\}}$ es el conjunto de eliminaciones, $\mathbf{E_i = \{\lambda\} \times Y}$ es el conjunto de inserciones. Cada tripla **(X, Y, $\mathbf{C_c}$)** implica una función de distancia $\boldsymbol{D: X^* \times Y^* \rightarrow R^+}$, que asigna a un par de cadenas un valor real no negativo. La distancia de edición d*(*x, y*)* es definida recursivamente como:

$$d(x,y) = \min \begin{cases} 0 & \text{if } x = \lambda \text{ and } y = \lambda \\ c(a:b) + d(x',y') & \text{if } x = ax' \text{ and } y = by' \\ c(a:\lambda) + d(x',y) & \text{if } x = ax' \\ c(\lambda:b) + d(x,y') & \text{if } y = by'. \end{cases}$$

Originalmente todas las operaciones de edición tenían costos fijos, pero en general estos costos pueden ser cambiados arbitrariamente por el usuario, definidos haciendo uso de matrices de sustitución o derivados de un corpus, usando técnicas de aprendizaje [1].

1.3.2 ALINEAMIENTOS DE SECUENCIAS.

El alineamiento es un algoritmo muy utilizado para determinar la similitud entre dos secuencias. El problema consiste en, dadas dos cadenas, encontrar todos los elementos de la primera que también están presentes en la segunda respetando su posición con respecto a los otros elementos que también pertenecen a ambas cadenas.

Sean dos secuencias $x = x_1, x_2, \ldots, x_m$ e $y = y_1, y_2, \ldots, y_n$, con $m \leq n$ sobre un alfabeto X de tamaño *s*, el problema consiste en encontrar la secuencia de longitud más grande que pueda ser obtenida de *x* e *y*, maximizando el número de caracteres que coinciden en ambas secuencias.

Un alineamiento de secuencias en Bioinformática, es una forma de representar y comparar dos o más secuencias o cadenas de ADN, ARN, o estructuras primarias proteicas, estableciendo los segmentos donde el número de coincidencias (una

coincidencia se presenta cuando el nucleótido de la secuencia A sea igual al nucleótido en la secuencia B) sea máximo, para resaltar sus zonas de similitud, que podrían indicar relaciones funcionales o evolutivas entre los genes o proteínas consultados. Las secuencias alineadas se escriben con las letras (representando aminoácidos o nucleótidos) en filas de una matriz en las que, si es necesario, se insertan espacios para que las zonas con idéntica o similar estructura se alineen. El alineamiento de secuencias puede utilizarse con secuencias no biológicas, como en la identificación de similitudes en series de letras y palabras del lenguaje humano o en análisis de datos financieros.

Secuencias muy cortas o muy similares pueden alinearse manualmente. Aún así, los problemas más interesantes necesitan alinear secuencias largas, muy variables y extremadamente numerosas que no pueden ser alineadas por humanos. El conocimiento humano se ha aplicado principalmente en la construcción de algoritmos que produzcan alineamientos de alta calidad. Las aproximaciones computacionales al alineamiento de secuencias se dividen en dos categorías: *alineamiento global* y *alineamiento local*.

1.3.2.1 ALINEAMIENTO LOCAL Y GLOBAL.

Existen dos formas de alinear dos secuencias: intentando encontrar los dos fragmentos de ambas secuencias que tienen un alineamiento con una puntuación máxima (**local**) o aquél alineamiento de las secuencias completas también con una puntuación máxima (**global**). Los alineamientos locales son adecuados cuando se sospecha que en las secuencias existen regiones similares, mientras que los alineamientos globales son útiles cuando las secuencias problema son similares y de tamaños aproximados. Con secuencias suficientemente similares, no existe diferencia entre alineamientos globales y locales [3].

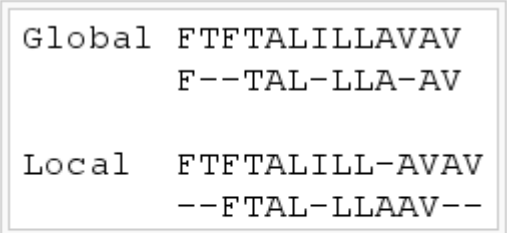


Imagen de un alineamiento local y uno global demostrando la tendencia a poner huecos de los alineamientos globales si las secuencias no son muy similares.

Los métodos híbridos, conocidos como semiglobales o métodos "glocales" intentan encontrar el mejor alineamiento posible que incluya el inicio y el final de una u otra secuencia. Puede ser especialmente útil cuando la parte "corriente arriba" de una secuencia se solapa con la parte "corriente abajo" de la otra. En este caso, ni el alineamiento global ni el local son completamente adecuados: un alineamiento global intentará forzar a la alineación a extenderse más allá de la región de solapamiento, mientras que el alineamiento local no cubrirá totalmente la región solapada [4] [5].

Existen dos algoritmos reconocidos y ampliamente utilizados en la Bioinformática para el alineamiento de secuencias. El algoritmo de SMITH & WATERMAN para alineamientos locales y el algoritmo NEEDLEMAN & WUNSCH para alineamientos globales. Ambos garantizan la búsqueda del alineamiento óptimo para las secuencias problema.

1.3.2.2 ALGORTIMO NEEDLEMAN-WUNSCH.

El algoritmo de Needleman-Wunsch es un algoritmo utilizado para realizar alineamientos globales de dos secuencias. Se suele utilizar en el ámbito de la Bioinformática para alinear secuencias de proteínas o de nucleótidos. Fue propuesto por primera vez en 1970, por Saul Needleman y Christian Wunsch. Se trata de un ejemplo típico de programación dinámica. Este algoritmo siempre termina y garantiza que la solución devuelta es la óptima [6] [7].

1.3.2.3 ALGORITMO SMITH-WATERMAN.

El algoritmo de Smith-Waterman es una reconocida estrategia para realizar alineamiento local de secuencias biológicas (ADN, ARN o proteínas); es decir que determina regiones similares entre un par de secuencias. Está basado en el uso de algoritmos de programación dinámica, de tal forma que tiene la deseable propiedad de garantizar que el alineamiento local encontrado es óptimo con respecto a un determinado sistema de puntajes que se use [6] [7].

1.4 PROGRAMACIÓN DINÁMICA Y HEURÍSTICA.

Los algoritmos de alineamiento y distancia de edición tratados anteriormente están basados en programación dinámica y aunque garantizan encontrar un alineamiento óptimo dada una función de puntuación en particular, conlleva a que el número de operaciones a realizar crece según el tamaño de las secuencias a comparar por lo que resulta costoso en tiempo y memoria. En el caso de la búsqueda en bases de datos, esto conduce a tiempos de cálculo importantes, por esta razón fueron desarrollados los algoritmos de k-tuplas que utilizan técnicas heurísticas. Estos algoritmos son mucho más rápidos, pero acarrean la pérdida de la garantía para obtener el óptimo resultado en el alineamiento. El objetivo de la heurística es la búsqueda de la fracción más pequeña posible, de las células de la matriz tratando de evitar perder los mejores alineamientos.

1.4.1 ALGORITMOS DE K-TUPLAS.

Los métodos de palabra corta, también conocidos como métodos de k-tuplas, son métodos heurísticos que no garantizan encontrar una solución de alineamiento óptima, pero son significativamente más eficientes que la programación dinámica. Estos métodos son especialmente útiles en búsquedas sobre bases de datos a gran escala, donde se asume que una larga proporción de las secuencias candidatas no tendrán coincidencias significativas con la secuencia problema. Los métodos de palabra corta son más conocidos por su implementación en las herramientas de búsqueda en bases de datos FASTA y BLAST [17]. Estos métodos identifican en la secuencia problema una serie de subsecuencias cortas que no se solapan (“palabras”), y que se contrastan contra las secuencias de la base de datos. Las posiciones relativas de la palabra en las dos secuencias a comparar se restan para obtener un valor de desplazamiento; se manifestará así una región de alineamiento si varias palabras diferentes producen el mismo desplazamiento. Sólo si esta región es detectada, estos métodos aplicarán criterios de alineamiento más sensibles. De esta forma se eliminan muchas comparaciones innecesarias entre secuencias de similitud inapreciable.

1.4.1.1 FASTA.

Es un conjunto o familia de programas cada uno pensado para una combinación específica de un tipo de secuencia y base de datos. FASTA fue el primer algoritmo ampliamente utilizado para la búsqueda de similitud en bases de datos. Desarrollado por David Lipman y William Pearson en el año 1985, es empleado principalmente por el EMBL-EBI (European Molecular Biology Laboratories - European Bioinformatics Institute), si se compara su velocidad con BLAST se notará que es mucho más lento, incluso llega a emplear varias horas para obtener los resultados, es por esta razón que el EMBL envía los cálculos al usuario por correo electrónico.

El algoritmo se basa en la búsqueda de alineamientos locales óptimos buscando coincidencias de pequeñas subsecuencias denominadas palabras o k-tuplas. La sensibilidad y velocidad del algoritmo es inversamente proporcional a la longitud de la palabra utilizada en la búsqueda, dicha longitud (k) es definido por el usuario. El método es más lento, pero más sensible, para valores bajos de (k), que también son preferibles para búsquedas que impliquen una secuencia problema muy corta.

El paquete actual de FASTA incluye programas para búsquedas del tipo proteína-proteína, ADN/ADN, proteína-ADN traducido (con cambios del marco de lectura), y búsqueda ordenada y desordenadas de péptidos. Además de los métodos de búsqueda heurísticos rápidos, el paquete FASTA provee SSEARCH, una implementación del algoritmo Smith-Waterman. Uno de los principales objetivos del paquete es el cálculo estadístico de las similitudes para que los biólogos puedan juzgar si un alineamiento ha ocurrido por azar o si se puede usar para inferir homología.

El paquete FASTA está disponible en http://fasta.bioch.virginia.edu e implementaciones de este se pueden encontrar en el portal web del EMBL: http://www.ebi.ac.uk/fasta33/.

FASTA trabaja sobre premisas distintas a las de BLAST, por lo que puede producir resultados diferentes. Habrá ocasiones en que FASTA funcione mejor o halle semejanzas que pasaron desapercibidas para BLAST y situaciones en que ocurrirá al revés.

1.4.1.2 BLAST.

Es en realidad una familia de programas que realizan varios tipos de búsqueda de secuencias. Proporcionando varios algoritmos optimizados para tipos particulares de problemas.

Es un sistema informático de alineamiento de secuencias ya sea de ADN o de proteínas. El programa es capaz de comparar una secuencia problema, contra una gran cantidad de secuencias que se encuentren en una base de datos. El algoritmo encuentra las secuencias de la base de datos que tienen mayor parecido a la secuencia problema. Es importante mencionar que BLAST usa un algoritmo heurístico por lo que no nos puede garantizar que ha encontrado la solución correcta.

Es desarrollado por Los Institutos Nacionales de Salud del gobierno de EE.UU., por lo que es de dominio público y puede usarse gratuitamente desde el servidor del Centro Nacional para la Información Biotecnológica (NCBI). También está disponible para ser instalado localmente. Algunas ventajas de usar el servidor del NCBI son que el usuario no tiene que mantener ni actualizar las bases de datos y que la búsqueda se hace en un clúster de computadoras, lo que otorga rapidez. Las desventajas son: no se permiten hacer búsquedas masivas dado que es un recurso compartido, no se puede personalizar las bases de datos contra la que busca el programa, y las secuencias son enviadas al servidor del NCBI sin ningún tipo de cifrado, lo que puede ser un problema para quienes quieran mantener sus secuencias privadas. La aplicación local de BLAST tiene la ventaja de permitir manejar varios parámetros que en las búsquedas de NCBI están estandarizados, por lo que provee una mayor flexibilidad para los usuarios avanzados. BLAST usa el algoritmo Smith-Waterman para realizar sus alineamientos.

Fue desarrollado para proporcionar una alternativa más rápida a FASTA sin sacrificar demasiada precisión. Como FASTA, BLAST utiliza una palabra de búsqueda de longitud (k), pero sólo evalúa las coincidencias más significativas de las palabras, en lugar de cada una de ellas como hace FASTA. La mayoría de las implementaciones de BLAST usan una longitud de palabra fijada por defecto que se optimiza según el problema y el

tipo de base de datos, y que se cambia sólo bajo circunstancias específicas tales como búsquedas con secuencias problema repetitivas o muy cortas. Pueden encontrarse implementaciones de este programa a través del portal web del NCBI disponible en: http://www.ncbi.nlm.nih.gov/BLAST/.

1.5 SISTEMA DE PUNTAJES.

Los algoritmos de cálculo de similitud descritos con anterioridad: distancia de edición y alineamientos tienen en común que se necesita determinado sistema de puntajes para poder calcular la similitud entre dos secuencias a partir de los costos de inserción, eliminación o sustitución de un símbolo por otro. Las operaciones de edición pueden tener costos fijos, pueden ser cambiados arbitrariamente, usar matrices de sustitución o derivarlos usando técnicas de aprendizaje. Se debe tener en cuenta que estos costos determinarán en gran medida los resultados obtenidos en la comparación de secuencias.

En biología computacional los costos más usados para el cálculo de similitud entre secuencias biológicas son los proporcionados por las matrices de sustitución PAM y BLOSUM.

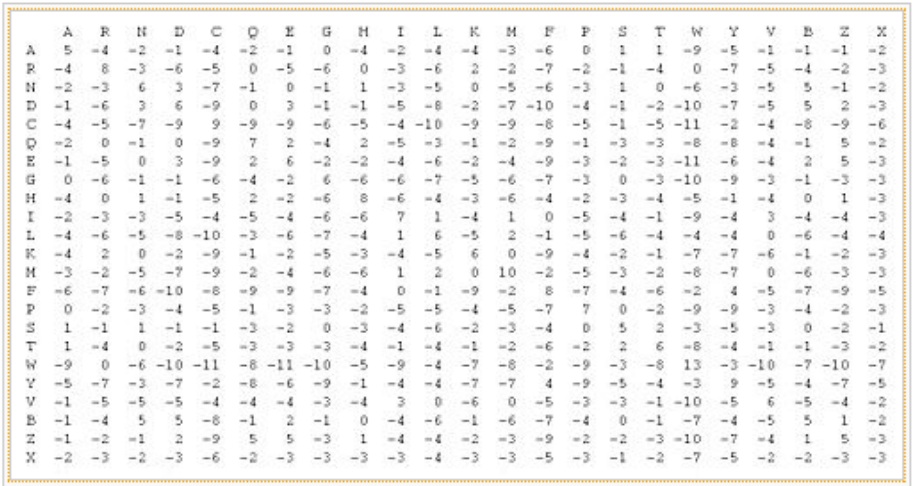
1.6 MATRICES DE SUSTITUCIÓN.

En biología evolutiva una matriz de sustitución, o de puntuación, describe el ritmo al que un carácter en una secuencia cambia a otro carácter con el tiempo. Las matrices de sustitución se ven usualmente en el contexto de alineamiento de secuencias de aminoácidos o ADN, donde la similitud entre secuencias depende del tiempo desde su divergencia y de los ritmos de sustitución según se representan en la matriz. Estas matrices se utilizan como parámetros de los algoritmos de alineamiento (Needleman-Wunsch o Smith-Waterman), en los cuales cumplen el papel de asignar una determinada puntuación a cada emparejamiento entre los aminoácidos de las secuencias a alinear, contribuyendo así a la puntuación global del alineamiento.

Este tipo de matrices son más usuales en los alineamientos de secuencias de aminoácidos (proteínas) que en los de nucleótidos (ADN), ya que en este último caso suele utilizarse un sistema de puntuación mucho más simple para los emparejamientos entre los cuatro diferentes nucleótidos y que asigna, normalmente, una puntuación positiva para la coincidencia en el emparejamiento, una puntuación nula o negativa para la no coincidencia, y una puntuación negativa para los huecos o gaps.

1.6.1 MATRIZ PAM.

Las matrices PAM (Percent Accepted Mutation) fueron descritas por Margaret Dayhoff en 1978 [6], están basadas en el alineamiento global de secuencias de proteínas estrechamente relacionadas y asumen que una modificación en algún sitio depende solamente del aminoácido presente en ese sitio. Esta matriz se calcula observando las diferencias en proteínas cercanamente relacionadas (con un mínimo del 85% de similitud). La matriz PAM1 estima qué ritmo de sustitución debería esperarse si el 1% de los aminoácidos han cambiado, y se usa como base para el cálculo de otras matrices asumiendo que mutaciones repetidas seguirían el mismo patrón que las reflejadas en la matriz PAM1, así como que múltiples sustituciones pueden darse en el mismo sitio. Usando esta lógica, Dayhoff derivó matrices tan altas como PAM250, aunque normalmente se utilizan PAM30 y PAM70.



	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V	B	Z	X
A	5	-4	-2	-1	-4	-2	-1	0	-4	-2	-4	-4	-3	-6	0	1	1	-9	-5	-1	-1	-1	-2
R	-4	8	-3	-6	-5	0	-5	-6	0	-3	-6	2	-2	-7	-2	-1	-4	0	-7	-5	-4	-2	-3
N	-2	-3	6	3	-7	-1	0	-1	1	-3	-5	0	-5	-6	-3	1	0	-6	-3	-5	5	-1	-2
D	-1	-6	3	6	-9	0	3	-1	-1	-5	-8	-2	-7	-10	-4	-1	-2	-10	-7	-5	5	2	-3
C	-4	-5	-7	-9	9	-9	-9	-6	-5	-4	-10	-9	-9	-8	-5	-1	-5	-11	-2	-4	-8	-9	-6
Q	-2	0	-1	0	-9	7	2	-4	2	-5	-3	-1	-2	-9	-1	-3	-3	-8	-8	-4	-1	5	-2
E	-1	-5	0	3	-9	2	6	-2	-2	-4	-6	-2	-4	-9	-3	-2	-3	-11	-6	-4	2	5	-3
G	0	-6	-1	-1	-6	-4	-2	6	-6	-6	-7	-5	-6	-7	-3	0	-3	-10	-9	-3	-1	-3	-3
H	-4	0	1	-1	-5	2	-2	-6	8	-6	-4	-3	-6	-4	-2	-3	-4	-5	-1	-4	0	1	-3
I	-2	-3	-3	-5	-4	-5	-4	-6	-6	7	1	-4	1	0	-5	-4	-1	-9	-4	3	-4	-4	-3
L	-4	-6	-5	-8	-10	-3	-6	-7	-4	1	6	-5	2	-1	-5	-6	-4	-4	-4	0	-6	-4	-4
K	-4	2	0	-2	-9	-1	-2	-5	-3	-4	-5	6	0	-9	-4	-2	-1	-7	-7	-6	-1	-2	-3
M	-3	-2	-5	-7	-9	-2	-4	-6	-6	1	2	0	10	-2	-5	-3	-2	-8	-7	0	-6	-3	-3
F	-6	-7	-6	-10	-8	-9	-9	-7	-4	0	-1	-9	-2	8	-7	-4	-6	-2	4	-5	-7	-9	-5
P	0	-2	-3	-4	-5	-1	-3	-3	-2	-5	-5	-4	-5	-7	7	0	-2	-9	-9	-3	-4	-2	-3
S	1	-1	1	-1	-1	-3	-2	0	-3	-4	-6	-2	-3	-4	0	5	2	-3	-5	-3	0	-2	-1
T	1	-4	0	-2	-5	-3	-3	-3	-4	-1	-4	-1	-2	-6	-2	2	6	-8	-4	-1	-1	-3	-2
W	-9	0	-6	-10	-11	-8	-11	-10	-5	-9	-4	-7	-8	-2	-9	-3	-8	13	-3	-10	-7	-10	-7
Y	-5	-7	-3	-7	-2	-8	-6	-9	-1	-4	-4	-7	-7	4	-9	-5	-4	-3	9	-5	-4	-7	-5
V	-1	-5	-5	-5	-4	-4	-4	-3	-4	3	0	-6	0	-5	-3	-3	-1	-10	-5	6	-5	-4	-2
B	-1	-4	5	5	-8	-1	2	-1	0	-4	-6	-1	-6	-7	-4	0	-1	-7	-4	-5	5	1	-2
Z	-1	-2	-1	2	-9	5	5	-3	1	-4	-4	-2	-3	-9	-2	-2	-3	-10	-7	-4	1	5	-3
X	-2	-3	-2	-3	-6	-2	-3	-3	-3	-3	-4	-3	-3	-5	-3	-1	-2	-7	-5	-2	-2	-3	-3

Figura1: Matriz PAM-70 para 23 aminoácidos.

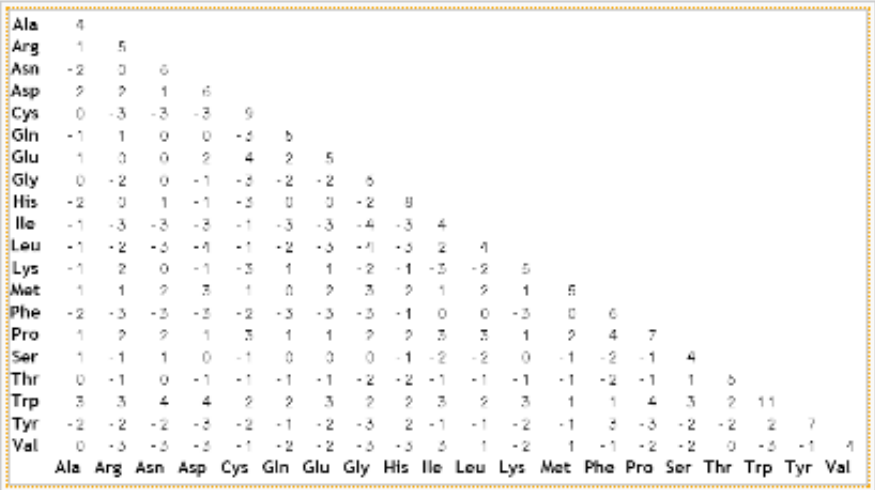
1.6.2 MATRIZ BLOSUM.

La matriz "BLOcks SUbstitution" fue propuesta por Steven Henikoff y Jorja G. Henikoff en el año de 1992 [9], fue creada a partir de un estudio sobre bloques conservados.

La metodología de Dayhoff al comparar especies cercanamente relacionadas resultó no trabajar muy bien al alinear secuencias evolutivamente divergentes. Los cambios en las secuencias a lo largo de largas escalas de tiempo evolutivo no son bien aproximados combinando pequeños cambios que ocurren en cortas escalas de tiempo. La serie de matrices BLOSUM (BLOck SUbstitution Matrix, o matriz de sustitución de bloques) corrige este problema. Steven Henikoff y Jorja G. Henikoff construyeron estas matrices usando múltiples alineamientos de proteínas evolutivamente divergentes.

Las probabilidades usadas en los cálculos de la matriz se computan observando los "bloques" de secuencias conservadas encontrados en múltiples alineamientos de proteínas. Se asume que estas secuencias conservadas son de importancia funcional dentro de las proteínas relacionadas. Para reducir el sesgo por secuencias cercanamente relacionadas, los segmentos de un bloque con una identidad secuencial por encima de un determinado umbral fueron agrupados, ponderando con un factor de 1 a cada uno de tales grupos

Para la matriz BLOSUM 62, este umbral se fijó en el 62%. Se consideraron, entonces, pares de frecuencias entre los grupos, por lo que estos pares fueron sólo tomados en consideración entre segmentos con menos de un 62% de identidad. Se usarán matrices BLOSUM de numeración alta para alinear dos secuencias cercanamente relacionadas, mientras que se utilizarán números más bajos para secuencias más divergentes. Se ha comprobado que la matriz BLOSUM 62 hace un excelente trabajo detectando similitudes en secuencias distantes, y esta es la matriz usada por defecto en las más recientes aplicaciones de alineamiento, como BLAST.



	Ala	Arg	Asn	Asp	Cys	Gln	Glu	Gly	His	Ile	Leu	Lys	Met	Phe	Pro	Ser	Thr	Trp	Tyr	Val
Ala	4																			
Arg	-1	5																		
Asn	-2	0	6																	
Asp	-2	-2	1	6																
Cys	0	-3	-3	-3	9															
Gln	-1	1	0	0	-3	5														
Glu	-1	0	0	2	-4	2	5													
Gly	0	-2	0	-1	-3	-2	-2	6												
His	-2	0	1	-1	-3	0	0	-2	8											
Ile	-1	-3	-3	-3	-1	-3	-3	-4	-3	4										
Leu	-1	-2	-3	-4	-1	-2	-3	-4	-3	2	4									
Lys	-1	2	0	-1	-3	1	1	-2	-1	-3	-2	5								
Met	-1	-1	-2	-3	-1	0	-2	-3	-2	1	-2	-1	5							
Phe	-2	-3	-3	-3	-2	-3	-3	-3	-1	0	0	-3	0	6						
Pro	-1	-2	-2	-1	-3	-1	-1	-2	-2	-3	-3	-1	-2	-4	7					
Ser	1	-1	1	0	-1	0	0	0	-1	-2	-2	0	-1	-2	-1	4				
Thr	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	1	5			
Trp	-3	-3	-4	-4	-2	-2	-3	-2	-2	-3	-2	-3	-1	1	-4	-3	-2	11		
Tyr	-2	-2	-2	-3	-2	-1	-2	-3	2	-1	-1	-2	-1	3	-3	-2	-2	2	7	
Val	0	-3	-3	-3	-1	-2	-2	-3	-3	3	1	-2	1	-1	-2	-2	0	-3	-1	4

Figura 2: Matriz BLOSUM-62.

1.7 DEFICIENCIA DE LOS SISTEMAS DE PUNTAJES CLÁSICOS.

La característica común de los métodos basados en distancia de edición y alineamientos es que son estáticos, en el sentido de que usan a priori costos fijos de las operaciones de edición (inserción, eliminación, sustitución) o usan un sistema de puntaje como matrices de sustitución lo que deja poco margen de adaptación al contexto de la cadena [11].

En muchos dominios reales, el nivel de un costo de edición debería depender no solo de los símbolos manipulados, sino también del contexto en donde se produce la operación. En biología computacional, una operación de edición que involucre a un mismo par de símbolos puede depender en gran medida en su ubicación en la secuencia de ADN. Una solución consistiría en asignar manualmente los costos para las operaciones de edición que reflejen la probabilidad de las correspondientes transformaciones. Sin embargo, el establecimiento de esta estrategia es difícil y no parece ser realista para aplicaciones generales con un bajo nivel de conocimientos especializados [11].

Entonces, se puede llegar a la conclusión de cuáles serían las consecuencias de: usar costos arbitrarios o usar un sistema de puntaje que no represente en realidad la

información contenida en las bases de datos. Por supuesto, los alineamientos y el cálculo de distancia de edición no serán correctos y por tanto realizar una inferencia errónea de familia y funcionalidad.

Investigaciones recientes tratan de superar estos inconvenientes y sustituir el ajuste a mano de los costos de edición para cada dominio con el aprendizaje de estos, mediantes técnicas informáticas. Varios modelos probabilísticos han sido propuestos para aprender la distancia de edición estocástica en forma de transductores estocásticos.

Estos modelos proporcionan una distribución de probabilidades sobre las operaciones de edición y por la tanto sobre los pares de cadena. La distancia de edición estocástica entre un par de cadenas puede ser computada como el logaritmo negativo de la probabilidad del par de cadenas [1] [14] [11] [12].

José Oncina y Marc Sebban en [12] exponen el desarrollo de una forma de aprender directamente un transductor condicional. Analizan el método de aprendizaje expuesto por Ristad y Yianilos [5] [16] lo implementan y realizan una serie de pruebas que avalan la relevancia del nuevo modelo en comparación con la distancia de edición estándar. Aquí se encuentra la base de este trabajo de diploma.

1.8 DISTANCIA DE EDICIÓN ESTOCÁSTICA.

Cuando el cálculo de las operaciones de edición (inserción, eliminación o sustitución) de un símbolo por otro para convertir una cadena (entrada) en otra (salida) y poder calcular la similitud entre este par (entrada, salida) está basado en fenómenos aleatorios y subyacen bajo una distribución de probabilidades, las operaciones de edición se convierten en variables aleatorias, esta medida de similitud pasa a ser llamada Distancia de Edición Estocástica y se denota: $d_s(x, y)$.

Una manera eficaz de modelar esta distancia consiste en verlo como una transducción estocástica entre los alfabetos de entrada y salida. Esto significa que la relación constituida por el conjunto de cadenas (entrada, salida) puede ser compilada en forma

de un *Transductor Estocástico* de estado finito. Este modelo es capaz de asignar una probabilidad a cada nuevo par de cadenas lo cual sería muy útil para hacer frente a muchos problemas sobre la base de operaciones de edición [12].

Definición 1: Un transductor de estado finito (FSET) es una 5-tuplas A = {Q, Z, i, F, T} tal que: Q es un conjunto finito de estados, Z = (X ∪ {λ}) × (Y ∪ {λ}) \ {(λ, λ)} alfabeto de las operaciones de edición, i € Q siendo el estado inicial, F : Q → [0, 1], T : Q × Z × Q → [0, 1] una función que asigna un peso a cada estado (transición). También se asumirá la siguiente condición determinista: $\forall p \in Q, \forall (a: b) \in Z, (\{q: T(p, (a: b), q) > 0\}) \leq 1$.

La estimación de los parámetros del transductor en sí mismo es el verdadero problema en este algoritmo. Por primera vez un algoritmo de entrenamiento para calcular la distribución de probabilidades y aprender automáticamente los parámetros de un transductor estocástico fue propuesto por Ristad y Yianilos. Ellos proveen un modelo estocástico que permite aprender los costos de las operaciones de edición para lograr el cálculo de la distancia de edición estocástica en forma de un transductor sin memoria es decir, con un sólo estado [12]. Este algoritmo de aprendizaje es llamado Expectation-Maximization.

1.8.1 TRANSDUCTORES. ALGORITMO DE APRENDIZAJE.

1.8.1.1 TRANSDUCTOR CONJUNTO.

Un Transductor sin memoria conjunto define una distribución de probabilidades conjuntas sobre los pares de cadenas. Se denota por una tupla (X, Y, c, γ), donde X es el alfabeto de entrada, Y es el alfabeto de salida, c es la primitiva de la función de probabilidad conjunta, c: E → [0, 1], y γ es la probabilidad del símbolo terminal de una cadena. Como (λ, λ) ∈ E, con el fin de simplificar la notación, vamos a utilizar c (λ, λ) y γ como sinónimos.

Suponga por el momento que se conoce la función de probabilidad c. Entonces se está en condiciones de calcular la probabilidad conjunta p(x, y) de un par de cadenas de

entrada-salida (x, y), definida por p: $X^* \times Y^* \rightarrow [0, 1]$ puede ser calculado recursivamente por medio de una función auxiliar (Forward) α: $X^* \times Y^* \rightarrow R^+$ como:

$$\begin{aligned}\alpha(x,y) = \quad & [1]_{x=\lambda \wedge y=\lambda} \\ & + [c(a,b) \cdot \alpha(x',y')]_{x=x'a \wedge y=y'b} \\ & + [c(a,\lambda) \cdot \alpha(x',y)]_{x=x'a} \\ & + [c(\lambda,b) \cdot \alpha(x,y')]_{y=y'b}.\end{aligned}$$

Por tanto: p(x, y) = α(x, y) γ.

En un modo simétrico, p(x, y) puede ser calculado recursivamente por medio de una función auxiliar (Backward) β: $X^* \times Y^* \rightarrow R^+$ como:

$$\begin{aligned}\beta(x,y) = \quad & [1]_{x=\lambda \wedge y=\lambda} \\ & + [c(a,b) \cdot \beta(x',y')]_{x=ax' \wedge y=by'} \\ & + [c(a,\lambda) \cdot \beta(x',y)]_{x=ax'} \\ & + [c(\lambda,b) \cdot \beta(x,y')]_{y=by'}.\end{aligned}$$

De igual manera: p (y|x) = β (y|x) γ.

Ambas funciones (Forward, Backward) puede ser calculada en tiempo O (|x| * |y|) utilizando técnicas de programación dinámica. Este modelo define una distribución de probabilidad sobre los pares de cadenas (x, y). De forma más precisa representado:

$$\sum_{x \in X^*} \sum_{y \in Y^*} p(x, y) = 1,$$

Para este modelo se deben satisfacer las siguientes condiciones:

$$\gamma > 0, c(a, b), c(\lambda, b), c(a, \lambda) \geq 0 \quad \forall a \in X, b \in Y$$

$$\sum_{\substack{a \in X \cup \{\lambda\} \\ b \in Y \cup \{\lambda\}}} c(a, b) = 1$$

Dado p(x, y), entonces es posible calcular, como se menciona en [1], la distancia de edición estocástica entre x e y. En realidad, la distancia de edición estocástica $d_S(x, y)$ se define como se mencionaba con anterioridad, como el logaritmo negativo de la probabilidad del par de cadenas p(x, y), de acuerdo al transductor sin memoria estocástico.

$$d_s(x, y) = -\log p(x, y), \forall x \in X^*, \forall y \in Y^*$$

Con el fin de calcular $d_S(x, y)$, el paso restante consiste en el aprendizaje de los parámetros c(a, b), del transductor sin memoria, o sea, los costos de edición primitivos.

Optimización de los parámetros del transductor.

Tomando S como el conjunto finito de pares de cadenas similares (x, y). Ristad y Yianilos en [1] proponen utilizar el algoritmo (Expectation-Maximization) para encontrar el transductor estocástico óptimo. El algoritmo EM consiste en dos pasos fundamentales (expectación y maximización) los que se repiten hasta que se logre arribar a un criterio de convergencia.

Usando una matriz (δ) auxiliar de tamaño (|X| + 1) × (|Y| + 1) el paso de expectación para el cálculo de los costos de edición puede ser descrito de la siguiente manera: ∀a ∈ X, b ∈ Y,

$$\delta(a,b) = \sum_{(xax',yby')\in S} \frac{\alpha(x,y)c(a,b)\beta(x',y')\gamma}{p(xax',yby')}$$
$$\delta(\lambda,b) = \sum_{(xx',yby')\in S} \frac{\alpha(x,y)c(\lambda,b)\beta(x',y')\gamma}{p(xx',yby')}$$
$$\delta(a,\lambda) = \sum_{(xax',yy')\in S} \frac{\alpha(x,y)c(a,\lambda)\beta(x',y')\gamma}{p(xax',yy')}$$
$$\delta(\lambda,\lambda) = \sum_{(x,y)\in S} \frac{\alpha(x,y)\gamma}{p(x,y)} = |S|,$$

Y el paso de Maximización:

$$c(a,b) = \frac{\delta(a,b)}{N} \quad \forall a \in X \cup \{\lambda\}, \forall$$

Donde N es:

$$N = \sum_{\substack{a\in X\cup\{\lambda\}\\ b\in Y\cup\{\lambda\}}} \delta(a,b).$$

$c^*(a,b)$	λ	a	b	c	d	$c^*(a)$
λ	0.00	0.05	0.08	0.02	0.02	0.17
a	0.01	0.04	0.01	0.01	0.01	0.08
b	0.02	0.01	0.16	0.04	0.01	0.24
c	0.01	0.02	0.01	0.15	0.00	0.19
d	0.01	0.01	0.01	0.01	0.28	0.32

Objetivo de la distribución c* (a, b) y su correspondiente distribución marginal c * (a).

1.8.1.1 TRANSDUCTOR CONDICIONAL.

Un Transductor sin memoria condicional se denota por una tupla (X, Y, c, γ), donde X es el alfabeto de entrada, Y es el alfabeto de salida, c es la primitiva de la función de probabilidad condicional, c: E → [0, 1], y γ es la probabilidad del símbolo terminal de una cadena. Como (λ, λ) ∈ E, con el fin de simplificar la notación, vamos a utilizar c (λ, λ) y γ como sinónimos.

Suponiendo por el momento que se conoce la función de probabilidad c. Entonces es posible calcular la probabilidad conjunta p(x, y) de un par de cadenas de entrada-salida (x, y), definida por p: $X^* \times Y^* \rightarrow [0, 1]$ puede ser calculado recursivamente por medio de una función auxiliar (Forward) α: $X^* \times Y^* \rightarrow R^+$ como:

La probabilidad p: $X^* \times Y^* \rightarrow [0, 1]$ de la cadena y asumiendo que la entrada era x, (note que p (y|x)) puede ser calculada recursivamente por medio de las funciones auxiliares (Forward) α: $X^* \times Y^* \rightarrow R^+$ y (Backward) β: $X^* \times Y^* \rightarrow R^+$ de la forma:

$$\begin{aligned}\alpha(y|x) = [1]_{x=\lambda \wedge y=\lambda} \\ + [c(b|a) \cdot \alpha(y'|x')]_{x=x'a \wedge y=y'b} \\ + [c(\lambda|a) \cdot \alpha(y|x')]_{x=x'a} \\ + [c(b|\lambda) \cdot \alpha(y'|x)]_{y=y'b}.\end{aligned}$$

$$\begin{aligned}\beta(y|x) = \quad [1]_{x=\lambda \wedge y=\lambda} \\ + [c(b|a) \cdot \beta(y'|x')]_{x=ax' \wedge y=by'} \\ + [c(\lambda|a) \cdot \beta(y|x')]_{x=ax'} \\ + [c(b|\lambda) \cdot \beta(y'|x)]_{y=by'}.\end{aligned}$$

Por tanto de forma respectiva: p (x|y) = α(x, y) γ y p (y|x) = β (y|x) γ.

Como en el caso de la distribución conjunta ambas funciones pueden ser computadas en un tiempo O (|x| * |y|) usando técnicas de programación dinámica. En este modelo la distribución de probabilidades es asignada condicionalmente para cada cadena de entrada. Entonces se obtiene:

$$\sum_{y \in Y^*} p(y|x) \in \{1, 0\} \quad \forall x \in X^*.$$

El 0 es en el caso en que la cadena de entrada x no esté presente en el dominio de la función (Si (p(x) = 0) entonces p(x, y) = 0 y como p (y|x) = p(x, y)/p(x) se tiene 0/0 o sea indeterminado. Esto se resuelve tomando 0/0 = 0 con el objetivo de mantener ∑y € Y * p (y|x) finito). La normalización para cada distribución condicional es lograda si son cumplidas las siguientes condiciones sobre la función c y el parámetro γ:

$$\gamma > 0, c(b|a), c(b|\lambda), c(\lambda|a) \geq 0 \quad \forall a \in X, b \in Y$$
$$\sum_{b \in Y} c(b|\lambda) + \sum_{b \in Y} c(b|a) + c(\lambda|a) = 1 \quad \forall a \in X$$
$$\sum_{b \in Y} c(b|\lambda) + \gamma = 1$$

Al igual que en el caso anterior (probabilidad conjunta) el algoritmo Expectation-Maximization es usado con el fin de encontrar los parámetros óptimos, de la misma forma que el paso de Expectación se refiere al cálculo de la matriz ð y el paso de Maximización permite deducir los costos de edición.

$$\delta(b|a) = \sum_{(xax', yby') \in S} \frac{\alpha(y|x)c(b|a)\beta(y'|x')\gamma}{p(yby'|xax')}$$
$$\delta(b|\lambda) = \sum_{(xx', yby') \in S} \frac{\alpha(y|x)c(b|\lambda)\beta(y'|x')\gamma}{p(yby'|xx')}$$
$$\delta(\lambda|a) = \sum_{(xax', yy') \in S} \frac{\alpha(y|x)c(\lambda|a)\beta(y'|x')\gamma}{p(yy'|xax')}$$
$$\delta(\lambda|\lambda) = \sum_{(x,y) \in S} \frac{\alpha(y|x)\gamma}{p(y|x)} = |S|.$$

$$c(b|\lambda) = \frac{\delta(b|\lambda)}{N} \qquad \gamma = \frac{N - N(\lambda)}{N}$$
$$c(b|a) = \frac{\delta(b|a)}{N(a)} \frac{N - N(\lambda)}{N} \qquad c(\lambda|a) = \frac{\delta(\lambda|a)}{N(a)} \frac{N - N(\lambda)}{N}$$

Donde N es:

$$N = \sum_{\substack{a \in X \cup \{\lambda\} \\ b \in Y \cup \{\lambda\}}} \delta(b|a) \qquad N(\lambda) = \sum_{b \in Y} \delta(b|\lambda) \qquad N(a) = \sum_{b \in Y \cup \{\lambda\}} \delta(b|a)$$

1.9 HERRAMIENTAS EXISTENTES.

1.9.1 BIOJAVA.

BioJava es un proyecto de código abierto dedicado a proporcionar herramientas para procesar datos biológicos. Incluye objetos para manipular secuencias biológicas, parseadores, soporte para cliente y servidor, acceso a bases de datos BioSQL y Ensembl, herramientas para hacer análisis de secuencias y potentes rutinas de análisis y estadística, incluyendo un toolkit de programación dinámica. Crea gráficos, permite comunicarse con el NCBI vía utilizando servicios web (SOAP-HTTP) y otras funcionalidades. Incluye el algoritmo BLAST para clasificación de secuencias en bases de datos.

1.9.2 CENTRO NACIONAL PARA LA INFORMACIÓN BIOTECNOLÓGICA.

El Centro Nacional para la Información Biotecnológica (National Center for Biotechnology Information (NCBI)) es la entidad responsable de hacer accesible las bases de datos del GenBank, además proporciona Herencia Mendeliana en el Hombre en línea, la Molecular Modeling Database (estructuras 3D de proteínas), dbSNP (una base de datos de polimorfismos de nucleótidos simples), la única colección de secuencias del genoma humano y un mapa de genes de dicho genoma. El NCBI ofrece además algunas herramientas bioinformáticas para el análisis de secuencias de ADN, ARN y proteínas, siendo BLAST una de las más usadas.

1.9.3 LABORATORIO EUROPEO DE BIOLOGÍA MOLECULAR.

El Laboratorio Europeo de Biología Molecular (European Molecular Biology Laboratory) (EMBL) es una institución de investigación en el campo de la biología molecular que ofrece también bases de datos de secuencias genéticas y herramientas para su análisis basadas en BLAST y FASTA.

1.10 TENDENCIAS, METODOLOGÍAS Y TECNOLOGÍAS EXISTENTES.

Es importante destacar que la selección de las tecnologías a utilizar en proceso de desarrollo de software se hace en base a las necesidades específicas de cada situación. De forma general está incorrecto decir que una tecnología es mejor que otra, sin antes hacer un análisis para escoger la que cumpla de forma satisfactoria con los requisitos y propicie un ambiente adecuado para el equipo de desarrollo.

Para el desarrollo de este sistema se hizo un estudio de las principales herramientas, tecnologías y metodologías existentes que estuvieran acorde a las especificaciones de los clientes. Luego de un análisis riguroso, entre los desarrolladores y el cliente se decidió cuales elegir para el desarrollo de la aplicación. A continuación se presentan los principales aspectos considerados que fundamentan la elección.

1.10.1 METODOLOGÍA PARA EL DESARROLLO DE SOFTWARE.

Las metodologías de desarrollo del software describen los pasos para desarrollar un producto de software. Definen detalladamente qué es lo que se debe hacer, cómo hacerse y quién debe hacerlo. Precisan cuáles son las tareas a desarrollarse durante el ciclo de vida del proyecto, los artefactos que han de ser construidos, cómo deben hacerse y el orden en que serán realizados, además de designar responsables por cada tarea. Especifican la estructura que ha de tener el proyecto para un mejor desenvolvimiento de todos los procesos llevados a cabo durante su desarrollo. De esta forma se obtiene como producto final, una solución eficaz, factible y de calidad al problema que le dio origen.

1.10.1.1 PROGRAMACIÓN EXTREMA (XP).

Es una de las metodologías de desarrollo de software ágil más reconocida en la actualidad, utilizada para proyectos de corto plazo y pequeño equipo de desarrollo, fue creada a mediados de la década de los 80 por Kent Beck. Esta metodología consiste en una programación ágil o extrema, cuya peculiaridad es tener como parte del equipo de desarrollo al usuario final, este es uno de los requerimientos para llegar al triunfo del proyecto. Está organizada en 4 cuatro fases: Planificación, Diseño, Desarrollo y Pruebas [21]. Posee cuatro variables principales que son:

- ✓ **Coste:** Equipo de desarrollo, computadoras y locales.
- ✓ **Calidad:** En el desarrollo del proyecto y en los entregables.
- ✓ **Tiempo:** Tiempo de entrega parcial y total del proyecto.
- ✓ **Ámbito:** Definición de problemas a resolver y cuáles se dejan para futuras versiones.

Esta metodología considera como aspecto fundamental, la comunicación entre los desarrolladores y el usuario final, todos forman parte del equipo de desarrollo, tienen como objetivo primordial la simplicidad al crear y codificar los módulos del sistema, la retroalimentación constante de ideas entre el equipo de desarrollo, el cliente y los usuarios finales y la refactorización.

Algunos de los beneficios de utilizar las prácticas de la programación extrema son: programación en pares, refactorización, integración continua, pruebas de aceptación, unidad de pruebas y otras que favorecen el aprendizaje de la programación e incitan a que se realicen investigaciones que apoyen la integración de estas prácticas [19].

1.10.1.2 PROCESO DE DESARROLLO UNIFICADO (RUP).

El proceso unificado de desarrollo de software (RUP - Rational Unified Process) es una de las metodologías más generales que existen, RUP es un proceso pensado en dos

dimensiones y basado en UML, el cual puede ser aplicado a cualquier proyecto de gestión. RUP es un proceso dirigido por casos de uso, lo que permite describir cada una de las funcionalidades que se espera del software, está centrado en la arquitectura del sistema y es iterativo e incremental con lo que se alcanza un desarrollo en iteraciones, en cada iteración se reproduce el ciclo de vida del software en cascada a menor escala. Los objetivos de una iteración se elaboran en función de lo que se cumplió en las anteriores. Permite la documentación de cada uno de los pasos a seguir en el proceso de desarrollo mediante planillas, que genera por cada una de las disciplinas; se basa en UML como herramienta principal [20] [22].

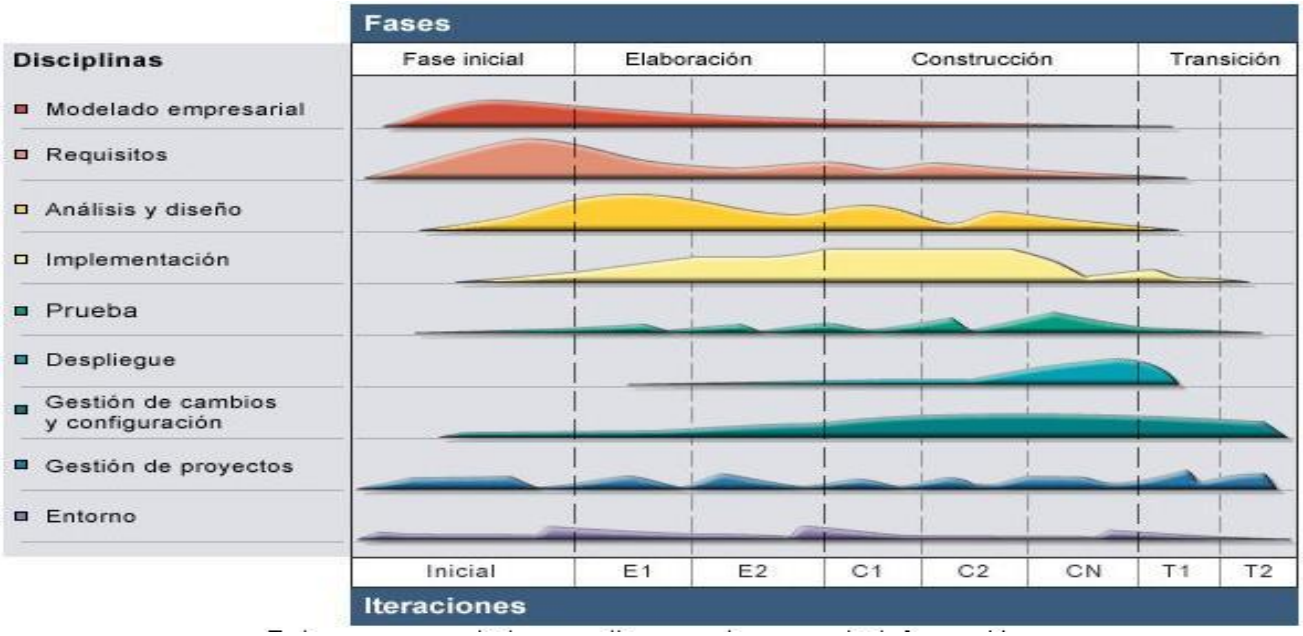


Figura 3: En esta gráfica de RUP en dos dimensiones se muestra la organización mediante flujos de trabajo.

1.10.1.3 PROCESO DE DESARROLLO DE SOFTWARE (OpenUP).

Para el desarrollo correcto de la aplicación se utiliza la metodología OpenUP. Esta preserva las características fundamentales de RUP, que incluye un desarrollo iterativo, casos de uso, escenarios, administración de riesgos, entre otros elementos. Excluye la

mayoría de las partes opcionales de RUP e incluye varios elementos nuevos. Es un proceso ágil y ligero, que promueve al desarrollo del software sobre las buenas prácticas, haciéndolo un proceso pequeño y extensible si es necesario (híbrido, incluye partes de otros modelos), por lo que presenta gran flexibilidad. Debido a estas ventajas que presenta OpenUP el polo de Bioinformática la adoptó para aplicarla en todos sus proyectos, y por consiguiente en el presente trabajo [8].

La metodología cuenta con los siguientes roles:

Stakeholder: Representa al grupo de interés cuyas necesidades deben quedar satisfechas por el proyecto. Este es un rol que podría ser desempeñado por cualquiera que esté (o potencialmente estará) materialmente afectado por el resultado del proyecto.

Líder de proyecto: Encargado de planear el proyecto. Provee un mapa a todo el equipo para que conozca la dirección del proyecto. Se adapta en base a feedback y cambio de entorno. Evalúa los riesgos. Determina el tamaño y alcance del proyecto. Define el largo, cantidad y objetivos de las iteraciones en un plan detallado, y asigna los roles a los miembros del equipo.

Analista: Describe el problema y las características del sistema basadas en las solicitudes de los stakeholders. Identifica y Refina los requerimientos, que consiste en entender los requerimientos de los stakeholders y comunicarlos al equipo de desarrollo, detalla los casos de usos y escenarios, y realiza un completo modelado del sistema.

Probador: Desarrolla los casos de prueba y los datos con que se probará para validar los requerimientos a ser testeados. Testea y evalúa los requerimientos de desarrollo desde la perspectiva del sistema, para determinar la calidad del producto.

Arquitecto: Define en alto nivel la arquitectura, desarrolla la visión de la arquitectura a través del análisis de los requerimientos más significativos, refina la arquitectura resolviendo los requerimientos con impacto en la arquitectura.

Desarrollador: Diseña la Solución, Implementa las pruebas de la solución, implementa la solución, ejecuta las pruebas de desarrollador, integra y crea el compilado.

Durante el desarrollo de la herramienta se desempeñaron los roles de analista, arquitecto, desarrollador y probador, realizando las actividades que la metodología propone.

1.10.2 LENGUAJE DE MODELADO.

El lenguaje de modelado para el desarrollo de productos de software más popular y utilizado en la actualidad es UML (Unified Modeling Language). UML es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software. Ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como: procesos de negocio y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables.

Es importante resaltar que UML es un "lenguaje" para especificar y no para describir métodos o procesos. Se utiliza para definir un sistema de software, para detallar los artefactos en el sistema y para documentar y construir. En otras palabras, es el lenguaje en el que está descrito el modelo. Se puede aplicar en una gran variedad de formas para dar soporte a una metodología de desarrollo de software, pero no especifica en sí mismo qué metodología o proceso usar.

1.10.3 LENGUAJES DE PROGRAMACIÓN.

Un lenguaje de programación es un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Es utilizado para controlar el comportamiento físico y lógico de una máquina. Un lenguaje de programación permite a uno o más programadores especificar de manera precisa sobre qué datos debe operar una computadora, cómo estos datos deben ser almacenados o transmitidos y qué acciones debe tomar bajo una variada gama de circunstancias.

1.10.3.1 C++.

C++ es un lenguaje de programación diseñado a mediados de los años 1980, la intención de su creación fue el extender al exitoso lenguaje de programación C con mecanismos que permitan la manipulación de objetos. En ese sentido, desde el punto de vista de los lenguajes orientados a objetos, el C++ es un lenguaje híbrido.

Posteriormente se añadieron facilidades, programación que se sumó a los otros dos paradigmas que ya estaban admitidos (programación estructurada y la programación orientada a objetos). Por esto se suele decir que el C++ es un lenguaje multiparadigma. Una particularidad del C++ es la posibilidad de redefinir los operadores (sobrecarga de operadores), y de poder crear nuevos tipos que se comporten como tipos fundamentales. C++ permite trabajar tanto a alto como a bajo nivel.

1.10.3.2 JAVA.

Java ofrece todas las ventajas de un lenguaje potente y robusto, pues fue diseñado para crear software altamente fiable. Es un lenguaje de programación basado en clases y orientado a objetos. Sus características de memoria liberan a los programadores de responsabilidades y errores. Es compilado en un código intermedio más abstracto que el código de máquina que es ejecutado por la máquina virtual de Java. Hereda su sintaxis de los lenguajes C y C++, pero cuenta con un modelo de objetos mucho más sencillo y elimina elementos de trabajo a bajo nivel como los punteros.

Una de las características más significativas de Java es que posee una arquitectura neutral, es decir, el compilador Java compila su código a un fichero objeto de formato independiente de la arquitectura de la máquina en que se ejecutará. La independencia de la arquitectura representa solo una parte de su portabilidad. Java especifica los tamaños de sus tipos de datos básicos y el comportamiento de sus operadores aritméticos, de manera que los programas son iguales en todas las plataformas.

Por las razones anteriormente expuestas y por solicitud del cliente, fue que se escogió Java como lenguaje para el desarrollo de esta aplicación. Además resaltar que se piensa

en un futuro, crear una aplicación Web y con la tecnología JSP se podría hacer un uso intensivo de la reutilización de código.

1.10.4 HERRAMIENTAS DE DESARROLLO (IDE).

Un entorno de desarrollo integrado (IDE), es un programa compuesto por un conjunto de herramientas para un programador. Puede dedicarse en exclusiva a un sólo lenguaje de programación o bien, poder utilizarse para varios.

Es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica GUI. Los IDEs pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes. Proveen un marco de trabajo amigable para los lenguajes de programación.

1.10.4.1 NETBEANS.

NetBeans es un producto libre y gratuito sin restricciones de uso. Es una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. Está escrito en Java pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender el IDE NetBeans. Entre sus características se encuentra sistema de proyectos basado en Ant, control de versiones y refactorización.

Todas las funciones del IDE son provistas por módulos. Cada módulo provee una función bien definida, tales como el soporte de Java, edición, o soporte para el sistema de control de versiones. NetBeans contiene todos los módulos necesarios para el desarrollo de aplicaciones Java en una sola descarga, permitiéndole al usuario comenzar a trabajar inmediatamente.

1.10.4.2 ECLIPSE.

Eclipse es una potente herramienta universal de entorno de desarrollo de software desarrollado en Java, usa java como lenguaje de programación aunque permite plugins para varios lenguajes más. La plataforma está construida en base a plugins, mecanismo que permite desarrollar, integrar y correr nuevos plugins. Otros beneficios que aporta el uso de Eclipse son:

- ✓ Es una herramienta de código abierto.
- ✓ Es neutral y adaptable a cualquier tipo de lenguaje, por ejemplo C/C++, Cobol, C#, XML, etc. La característica clave de Eclipse es la extensibilidad.
- ✓ Soporta la programación orientada a objetos (POO).
- ✓ La depuración e implementación de aplicaciones resultan mucho más sencillas.
- ✓ Corre en una gran cantidad de sistemas operativos incluyendo Windows y Linux.
- ✓ Provee a los desarrolladores, herramientas (ej.- PDE) que facilitan la creación de plugins.

¿Por qué utilizar Eclipse?

Después de analizar las características de NetBeans y las de Eclipse se decidió que este último fuera el IDE seleccionado para desarrollar la herramienta propuesta por sus grandes potencialidades y sus posibilidades de extensión mediante el uso de plugins. También una razón de gran peso es el conocimiento y experiencia del equipo de desarrollo de software, que ha usado este IDE en múltiples aplicaciones, por lo que tiene cierta experiencia en su uso.

1.10.5 SISTEMA GESTOR DE BASES DE DATOS.

Base de Datos: Una Base de datos es un conjunto de datos persistentes, pertenecientes a un mismo contexto e interrelacionados entre sí de forma lógica, que representa una situación del mundo real.

En la actualidad, y debido al desarrollo tecnológico de campos como la informática y la electrónica, la mayoría de las bases de datos están en formato digital (electrónico), que ofrece un amplio rango de soluciones al problema de almacenar datos.

Existen programas denominados **SGBD** (Sistema Gestor de Bases de Datos), que permiten almacenar y posteriormente acceder a los datos de forma rápida y estructurada. Las propiedades de estos SGBD, así como su utilización y administración, se estudian dentro del ámbito de la informática.

SGBD: Es un conjunto de programas que permite a los usuarios crear y mantener una BD, asegurando su integridad, confidencialidad y seguridad, por lo tanto, el SGBD es un software de propósito general que facilita el proceso de definir, construir y manipular la BD para diversas aplicaciones. Pueden ser de propósito general o específico.

Actualmente se hace difícil pensar en una aplicación o sistema informático que no esté conectada a una base de datos debido a las ventajas y funcionalidades que brindan los sistemas de bases de datos en la manipulación de las mismas. Existen diferentes gestores como Oracle, PostgreSQL, SQL Server, MySQL, entre otros.

1.10.5.1 POSTGRESQL.

Es un servidor de base de datos relacional orientada a objetos de software libre. Está considerado como la base de datos de código abierto más avanzada del mundo. PostgreSQL aproxima los datos a un modelo objeto-relacional, y es capaz de manejar complejas rutinas y reglas. Ejemplos de su avanzada funcionalidad son las consultas SQL declarativas, el control de concurrencia multi-versión, el soporte de multi-usuario, transacciones, optimización de consultas, herencia, y arreglos. En este proyecto se usa

PostgreSQL, además de las ventajas que posee, porque la estructura de la base de la tecnología del cliente presenta dicho gestor de base de datos y se debe de cumplir con sus peticiones.

Por otro lado, la velocidad de respuesta que ofrece este gestor con bases de datos relativamente pequeñas puede parecer un poco deficiente, aunque esta misma velocidad la mantiene al gestionar bases de datos realmente grandes, cosa que resulta loable. Tiene prácticamente todo lo que tienen los gestores comerciales, haciendo de él una muy buena alternativa GPL. A pesar de ello, el primer encuentro con este gestor es un poco "duro", ya que la sintaxis de algunos de sus comandos no es nada intuitiva. También resultan engorrosas las pequeñas variaciones que presenta este gestor en algunos de los tipos de datos que maneja. Posee una gran escalabilidad, haciéndolo idóneo para su uso en sitios web que posean alrededor de 500.000 peticiones por día.

Ventajas: Las características positivas que posee este gestor según las opiniones más comunes en Internet, son:

- ✓ Posee una gran escalabilidad. Es capaz de ajustarse al número de CPUs y a la cantidad de memoria que posee el sistema de forma óptima, haciéndole capaz de soportar una mayor cantidad de peticiones simultáneas de manera correcta (en algunos benchmarks se dice que ha llegado a soportar el triple de carga de lo que soporta MySQL).

- ✓ Implementa el uso de rollback's, subconsultas y transacciones, haciendo su funcionamiento mucho más eficaz, y ofreciendo soluciones en campos en las que MySQL no podría.

- ✓ Tiene la capacidad de comprobar la integridad referencial, así como también la de almacenar procedimientos en la propia base de datos, equiparándolo con los gestores de bases de datos de alto nivel, como puede ser Oracle.

Desventajas: Por contra, los mayores inconvenientes que se pueden encontrar a este gestor son:

- ✓ Consume gran cantidad de recursos.
- ✓ Tiene un límite de 8K por fila, aunque se puede aumentar a 32K, con una disminución considerable del rendimiento.
- ✓ Es de dos a tres veces más lento que MySQL.

1.10.5.2 MYSQL.

Es un sistema de gestión de bases de datos relacional. Su diseño multihilo le permite soportar una gran carga de forma muy eficiente. Actualmente MySQL se desarrolla como software libre en un esquema de licenciamiento dual. Por un lado se ofrece bajo la GNU GPL para cualquier uso compatible con esta licencia, pero para aquellas empresas que quieran incorporarlo en productos privados deben comprar a la empresa una licencia específica que les permita este uso [15].

Al contrario de proyectos como Apache, donde el software es desarrollado por una comunidad pública y el copyright del código está en poder del autor individual, MySQL es propietario y está patrocinado por una empresa privada, que posee el copyright de la mayor parte del código.

Este gestor de bases de datos es, probablemente, el gestor más usado en el mundo del software libre, debido a su gran rapidez y facilidad de uso. Esta gran aceptación es debida, en parte, a que existen infinidad de librerías y otras herramientas que permiten su uso a través de gran cantidad de lenguajes de programación, además de su fácil instalación y configuración.

Este gestor de bases de datos aprovecha la potencia de sistemas multiprocesador, gracias a su implementación multihilo, soporta gran cantidad de tipos de datos para las columnas. Dispone de APIs en gran cantidad de lenguajes (C, C++, Java, PHP, etc.), posee gran portabilidad entre sistemas. Soporta hasta treinta y dos índices por tabla y tiene una gestión de usuarios y contraseñas, manteniendo un muy buen nivel de seguridad en los datos.

Ventajas: Es evidente que la gran mayoría de gente usa este gestor en Internet, por lo que encontrar opiniones favorables no es en absoluto complicado:

- ✓ Sin lugar a duda, lo mejor de MySQL es su velocidad a la hora de realizar las operaciones, lo que le hace uno de los gestores que ofrecen mayor rendimiento.

- ✓ Su bajo consumo lo hacen apto para ser ejecutado en una máquina con escasos recursos sin ningún problema.

- ✓ Las utilidades de administración de este gestor son envidiables para muchos de los gestores comerciales existentes, debido a su gran facilidad de configuración e instalación.

- ✓ Tiene una probabilidad muy reducida de corromper los datos, incluso en los casos en los que los errores no se produzcan en el propio gestor, sino en el sistema en el que está.

- ✓ El conjunto de aplicaciones Apache-PHP-MySQL es uno de los más utilizados en Internet en servicios de foro (Barrapunto.com) y de buscadores de aplicaciones (Freshmeat.net).

Desventajas: Debido a esta mayor aceptación en Internet, gran parte de los inconvenientes se exponen a continuación, han sido extraídos de comparativas con otras bases de datos:

- ✓ Carece de soporte para transacciones, rollback's y subconsultas.

- ✓ El hecho de que no maneje la integridad referencial, hace de este gestor una solución pobre para muchos campos de aplicación, sobre todo para aquellos programadores que provienen de otros gestores que sí que poseen esta característica.

- ✓ No es viable para su uso con grandes bases de datos, a las que se acceda continuamente, ya que no implementa una buena escalabilidad.

Después de analizar las características de estos dos sistemas gestores de Bases de Datos y haber definido las prioridades y necesidades, se ha optado por el uso de PostGreSQL.

1.10.6 HERRAMIENTAS ORM.

Con el paradigma de desarrollo de software Orientado a Objetos (OO) ha surgido un problema con respecto al almacenamiento y gestión de datos. El paradigma relacional se basa en principios matemáticos con tablas y relaciones entre estas y el paradigma de OO se basa en principios de desarrollo de software como clases, objetos y herencia, estos dos paradigmas no conviven fácilmente. Esto se ve claramente en la manera de acceder a los datos. Mientras en el mundo OO los objetos se relacionan a través de propiedades, en el mundo relacional las tablas se unen a través de llaves primarias. Este problema es conocido como "desajuste de relaciones" y es conocido hace ya un buen tiempo. Las herramientas ORM tratan de solucionar este problema.

1.10.6.1 FRAMEWORK HIBERNATE.

En los entornos de desarrollo actuales, trabajar con software orientado a objetos y bases de datos relacionales puede hacerle invertir mucho tiempo a los desarrolladores. Hibernate realiza el mapeo entre el mundo orientado a objetos de las aplicaciones y el mundo entidad-relación de las bases de datos. El centro de la arquitectura de Hibernate lo constituyen una serie de interfaces que realizan el grueso de las funcionalidades del framework, dentro de ellas están:

Session: Es la más usada en las aplicaciones, es la manejadora de la persistencia: a través de ella se podrá guardar y cargar objetos de la base de datos.

SessionFactory: Mediante ella se obtiene la Session.

Configuration: Es usada para configurar Hibernate y para especificar la ruta de los documentos de mapeo.

Transaction: Se utiliza para el control de las transacciones.

Query y Criteria: Permiten la ejecución de consultas a la Base de Datos. Además de estas interfaces Hibernate brinda otros aspectos muy importantes y que serán de gran utilidad en las aplicaciones, por ejemplo: La interface Type, es un elemento fundamental y muy poderoso, permite el mapeo de tipos “java” a columnas en bases de datos incluso a varias columnas, incluye tipos como Calendar, byte [] y permite además; definir los propios tipo de datos (Persona, Dirección, Nombre). Las interfaces Callback gestionan el ciclo de vida de los objetos (Lifecycle, Validatable).

Un dialecto orientado a objetos (HQL) fácil de manejar y familiar a SQL que aunque no es un lenguaje de manipulación de datos como este, puesto que es usado solamente para extraer datos no para borrar, insertar o actualizar, permite realizar complejas consultas. Hibernate puede ser configurado y ejecutado en cualquier aplicación java y entorno desarrollo. Define dos tipos de configuración: Entorno manejado, entornos que proveen reservas de recursos como conexiones a base de datos (connections pool), transacciones y seguridad declarativa, en este caso se encuentran los servidores de aplicación como JBoss, BEA, WebLogic, entre otros.

Entorno no manejado, es el caso de los contenedores de servlet como Tomcat, las aplicaciones de escritorio también son incluidas aquí; este tipo de entorno no provee transacciones automáticas ni manejos de recursos, la propia aplicación realiza el manejo de las conexiones a base de datos. En los entornos no manejados Hibernate se ocupa de las transacciones y las conexiones JDBC o lo deja para que el desarrollador se ocupe de esto, en el caso de los entornos manejados Hibernate se integra con el contenedor para ocuparse de los DataSources y de las transacciones. Salvo estas diferencias lo demás es común para cualquier entorno.

En los entornos no manejados Hibernate se ocupa de las transacciones y las conexiones JDBC o lo deja para que el desarrollador se ocupe de esto, en el caso de los entornos manejados Hibernate se integra con el contenedor para ocuparse de los DataSources y de las transacciones. Salvo estas diferencias lo demás es común para cualquier entorno.

Otros aspectos que son esenciales en el desarrollo de aplicaciones con Hibernate lo constituyen los ficheros de mapeo y configuración.

Para cada clase persistente se le hace corresponder un fichero de mapeo que es el lugar donde se le especifica al framework como va a persistir dicha clase en la base de datos pero además se trata todo lo referente a las relaciones de esta clase con otras incluyendo el tratamiento de herencia y polimorfismo. El archivo de configuración es el que le dice a Hibernate todo lo referente a la conexión que se puede alcanzar vía JNDI, en el caso de los entornos manejados o cargando la conexión con el driver correspondiente al gestor, haciendo uso si se quiere de alguna herramienta que maneje la reserva de conexiones todo esto en el mismo archivo.

Ventajas:

- ✓ Abstrae del manejo del código SQL, permitiendo ganar en tiempo de desarrollo.
- ✓ No necesita ningún entorno especial para correr sea servidor aplicaciones, contenedores.
- ✓ Las clases que persisten no tienen que implementar ninguna interfaz especial ni extender ninguna clase, por lo que son completamente trasparentes haciendo posible su reutilización en distintas partes de la aplicación con el objetivo de transportar los datos, y haciéndola desacoplada.
- ✓ Permite manejar las transacciones de una manera eficaz.
- ✓ Toda la configuración de la conexión y el mapeo se realiza en ficheros externos, haciendo transparente el uso de distintos gestores.
- ✓ Permite manejo de almacén de conexiones (Connections Pool).
- ✓ Es un framework de código abierto y gratis.

Desventajas:

- ✓ No es un estándar J2EE.
- ✓ Es un framework complejo. Puede costar un poco de esfuerzo dominar cada una de las propiedades que caracterizan al mismo.

¿Por qué usar Hibernate?

Hibernate crea un puente entre el mundo orientado a objetos de las aplicaciones y en el entorno relacional de las base de datos., abstrayendo al desarrollador del código JDBC y de las consultas SQL. Es un framework ORM (Object Relational Mapping), que se basa en ficheros XML que mapean los objetos con tablas en la base de datos. Hibernate constituye un mecanismo persistente transparente pues las clases persistentes desconocen que tiene esa propiedad porque no tienen que implementar ninguna interfaz especial, ni extender de ninguna clase, y pueden ser utilizadas en cualquier capa de la aplicación. Hibernate provee un lenguaje de consultas orientado a objetos que facilita la ejecución de estas; crea una capa de abstracción que permite migrar el gestor de base de datos sin cambiar una sola línea de código. Por lo antes mencionado y la experiencia del equipo de desarrollo con este framework se decidió la utilización del mismo.

1.10.7 HERRAMIENTAS PARA EL MODELADO.

Las herramientas CASE (Computer Aided Software Engineering - Ingeniería de Software Asistida por Ordenador) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero. Estas herramientas nos pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software, permitiendo mejorar la productividad en el desarrollo y mantenimiento del software en diferentes aspectos como:

- ✓ Aumentar la calidad del software.
- ✓ Mejorar el tiempo y coste de desarrollo y mantenimiento de los sistemas informáticos.

- ✓ Mejorar la planificación de un proyecto.
- ✓ Aumentar la biblioteca de conocimiento informático de una empresa ayudando a la búsqueda de soluciones para los requisitos.
- ✓ Automatizar, desarrollo del software, documentación, generación de código, pruebas de errores y gestión del proyecto.
- ✓ Ayuda a la reutilización del software, portabilidad y estandarización de la documentación
- ✓ Gestión global en todas las fases de desarrollo de software con una misma herramienta.
- ✓ Facilitar el uso de las distintas metodologías propias de la ingeniería del software.

1.10.7.1 RATIONAL ROSE.

Rational Rose es una herramienta CASE, importante para el Modelado Visual mediante UML de sistemas de software. Permite especificar, analizar y diseñar el sistema antes de codificarlo. Mantiene la consistencia de los modelos del sistema de software realiza el chequeo de la sintaxis UML, genera la documentación automáticamente, genera código a partir de los modelos, realiza Ingeniería Inversa (crear modelo a partir código); entre otras funcionalidades. Facilita el desarrollo de un proceso cooperativo en el que todos los agentes tienen sus propias vistas de información (vista de Casos de Uso, vista Lógica, vista de Componentes y vista de Despliegue). Permite que los arquitectos y diseñadores practiquen el desarrollo orientado al modelado, permitiéndoles producir modelos independientes a la arquitectura de la plataforma del software y necesidades del negocio.

1.10.7.2 VISUAL PARADIGM.

Es una herramienta UML CASE considerada muy completa y fácil de usar, con soporte multiplataforma y que proporciona excelentes facilidades de interoperabilidad con otras aplicaciones. Permite la generación de código para Java y exportación como HTML, es fácil de instalar y actualizar y posee compatibilidad entre ediciones. Además ofrece:

- ✓ Diseño centrado en casos de uso y enfocado al negocio que genera un software de mayor calidad.
- ✓ Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- ✓ Capacidades de ingeniería directa e inversa.
- ✓ Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- ✓ Disponibilidad de múltiples versiones para cada necesidad.
- ✓ Disponibilidad de integrarse en los principales IDEs.

Después del estudio de ambas herramientas, además de que es la propuesta por nuestro polo de Bioinformática se decidió utilizar como herramienta para el modelado, el Visual Paradigm.

1.11 RESUMEN DE TA TECNOLOGÍA A UTILIZAR.

Finalmente se ha llegado a la conclusión de que la aplicación se desarrollará usando como lenguaje de programación Java, como gestor de bases de datos PostgreSQL y framework ORM Hibernate. Se determinó además hacer uso del Lenguaje Unificado de Modelado (UML) y OpenUP como metodología de desarrollo. Como herramienta de modelado se utiliza Visual Paradigm y como Entorno Integrado de Desarrollo, Eclipse.

1.12 CONCLUSIONES.

En este capítulo se definió una visión general de los principales conceptos necesarios para la comprensión y entendimiento del sistema y el funcionamiento básico de los algoritmos presentes en la aplicación. Se analizaron las metodologías, herramientas, técnicas y el lenguaje de programación a utilizar teniendo en cuenta las necesidades y funcionalidades que posee la herramienta.

CAPÍTULO II. CARACTERÍSTICAS DEL SISTEMA.

En el presente capítulo se hace una descripción del objeto de estudio, se hace alusión a los problemas existentes que dieron pie al desarrollo de la herramienta, se definen los requerimientos, tanto funcionales como no funcionales y se brinda una propuesta de solución para la construcción de la misma, haciendo uso de los artefactos y procesos que ofrece la metodología OpenUP, obteniendo como resultados: los actores que intervienen en el sistema, casos de uso del sistema y sus correspondientes descripciones, así como el diagrama de casos de uso del sistema.

2.1 OBJETO DE ESTUDIO.

Para el desarrollo de la aplicación se hizo necesario el estudio de los algoritmos de distancia de edición, métodos de k-vecinos más cercanos y los algoritmos de aprendizaje en forma de transductores sin memoria o de estado finito.

2.2 PROBLEMA Y SITUACIÓN PROBLÉMICA.

2.2.1 OBJETIVOS ESTRATÉGICOS.

El Instituto de Ciencia Animal actualmente lleva a cabo diferentes investigaciones relacionadas con el desarrollo integral de la ganadería cubana, especial atención han prestado a los recientes progresos biotecnológicos en la producción de nuevos alimentos para el ganado, como es el caso de los que han conducido a la obtención de nuevos clones de pastos y forrajes. También han diseñado programas computadorizados para su aplicación en el manejo y alimentación animal. Su misión es generar, desarrollar y transferir las mejores y más rentables tecnologías en el campo de la producción agropecuaria tropical mediante la investigación, el extensionismo, la capacitación, la divulgación y la colaboración eficiente.

2.2.2 FLUJO ACTUAL DE LOS PROCESOS.

Actualmente los procesos fluyen de la siguiente manera: En el Instituto de Ciencia Animal (ICA), el proceso de clasificación de microorganismos no cultivados se lleva a cabo en los laboratorios con instrumentos biológicos, no poseen ni utilizan ninguna herramienta o aplicación informática para automatizar el proceso.

2.2.3 ANÁLISIS CRÍTICO DE LA EJECUCIÓN DE LOS PROCESOS.

Los procesos de estudio e investigaciones, sobre el análisis y clasificación taxonómica de secuencias, para el estudio de los microorganismos de rumen que son llevados a cabo actualmente, poseen una estructura que no resuelve todos los problemas y necesidades que se originan sobre este marco de trabajo, debido a la rapidez y la eficiencia que exigen y necesitan.

Estos procesos se realizan de forma manual en los laboratorios, implicando que entre mayor sea la cantidad de secuencias a clasificar, el proceso abarca una mayor cantidad de tiempo para los nuevos estudios e investigaciones, donde entran en juego el futuro de las grandes aspiraciones en la producción de los diferentes productos que aportan los rumiantes, pues los índices de producción pueden aumentar con las soluciones alimenticias que permitan regular el comportamiento de los microorganismos presentes en el rumen.

2.3 PROCESOS OBJETO DE AUTOMATIZACIÓN.

En el proceso de estudio de las comunidades microbianas y su posterior clasificación, se debe automatizar el proceso de clasificación de un microorganismo del rumen o una base de datos de microorganismos. Para ello, es necesario además automatizar el proceso de aprendizaje de los transductores sin memoria estocásticos, y el proceso de cálculo de tasa de errores, que permitirá evaluar qué tan acertada es la clasificación que se está llevando a cabo.

Visión general del proceso: Para el proceso de clasificación de microorganismos es recomendado hacer primeramente un aprendizaje a partir de la base de datos sobre la cual se clasificará, para posteriormente utilizar este transductor como base de la clasificación de secuencias o bases de datos de secuencias. Aunque se puede llevar a cabo este proceso con la distancia de edición clásica con valores unitarios o prefijados.

En cualquier caso, es posible realizar pruebas con el módulo de Tasa de Error para tener una idea de la exactitud con la que se realizará la clasificación. Después de estas acciones recomendadas, ya el investigador está en condiciones de realizar la clasificación de uno o un conjunto de microorganismos para lo cual seleccionará el tipo de clasificación y el método por la cual se llevará a cabo.

El proceso de **aprendizaje del transductor** será realizado mediante el algoritmo Expectation-Maximization explicado en el Capítulo 1. Primeramente es necesario crear una Base de Datos de Pares de Cadena (Pair-Database), sobre la cual se realizará el aprendizaje o entrenamiento del Transductor. El paso de creación del Pair-Database sigue las siguientes etapas:

- ✓ Para todas las secuencias de la Base de Datos se busca su vecino más cercano (nearest-neighbor) usando distancia de edición con costos unitarios.
- ✓ El cálculo del valor de similitud se realiza haciendo uso de un algoritmo de programación dinámica, el cual asegura que se encontrará el mínimo número de operaciones de edición entre las cadenas a comparar, según el sistema de costos unitarios.
- ✓ Para hallar la secuencia con más similitud solo se compara con las secuencias que posean la misma clasificación y que no sea el propio elemento.
- ✓ De esta forma se asegura que el aprendizaje sea lo más real posible. Al terminar de construir esta base de datos, es cuando comienza el algoritmo de Expectation-Maximization sobre dicha base de datos.

El algoritmo Expectation-Maximization toma como parámetros de entrada el Pair-Database construido en el paso anterior y la forma de cálculo de las probabilidades del transductor (Conjunta ó Condicional). Este paso está compuesto por las siguientes etapas:

1. Primeramente se crea un transductor aleatorio y se normaliza según el tipo de distribución seleccionada.

2. Para cada par de cadenas del Pair-Database se realiza un paso de Expectación-Maximización haciendo uso del transductor obtenido en el paso anterior (el primer paso usa el transductor aleatorio, obtenido en la primera etapa). Esta etapa da como resultado un transductor temporal, que será usado en el siguiente paso de EM.

3. Al finalizar las iteraciones para cada par de cadenas del Pair-Database se verifica si se ha llegado al umbral de precisión seleccionado por el usuario, de no ser así, se comienza por la etapa dos, utilizando el transductor temporal obtenido.

4. Cuando es alcanzado el umbral de precisión se llega al final del aprendizaje, que brinda como tiene como resultado un transductor óptimo para poder realizar clasificaciones sobre la base de datos con que se realizó el entrenamiento.

El proceso de clasificación es realizado mediante el cálculo de similitud entre secuencias haciendo uso de un algoritmo de programación dinámica que permite calcular la distancia de edición entre cadenas de caracteres. Se hace uso de costos unitarios, prefijados ó especificados por transductores, siendo esta última estrategia la más recomendada. El proceso en forma general transita por las siguientes etapas:

- ✓ Dada una secuencia problema, se compara contra cada uno de los elementos de la base de datos especificada.

- ✓ Haciendo uso de la distancia de edición, se calcula la similitud entre cadenas dependiendo de la estrategia trazada, haciendo uso de los costos unitarios, prefijados ó especificados por un transductor, para cada una de las operaciones de edición.

- ✓ Al finalizar el algoritmo, son encontrados los elementos con más similitud en la base de datos, siendo estos los candidatos potenciales a homólogos de la secuencia problema, y por tanto a compartir su misma clasificación.

El módulo de tasa de error permite verificar el nivel de precisión de clasificación sobre una base de datos, dependiendo de los costos para las operaciones de edición, además de validar los algoritmos de clasificación.

El cálculo de la tasa de error interna se realiza para comprobar en una base de datos si, no teniendo en cuenta el propio elemento es posible llegar a encontrar un elemento de la misma clasificación, haciendo uso del método de clasificación propuesto. En general se siguen los siguientes pasos:

- ✓ Se compara cada elemento de la base de datos con los restantes (nunca es tenido en cuenta el propio elemento), haciendo uso de la distancia de edición y los costos de edición especificados.

- ✓ En caso que el elemento encontrado con más similitud no comparta la misma clasificación, se contabiliza como un error.

- ✓ Al finalizar el algoritmo se tiene el número de errores encontrados y la tasa de error para esta base de datos con los costos utilizados. La tasa de error se calcula como: (cantidad de errores/cantidad de secuencias de la base de datos).

El cálculo de tasa de error mediante KNN sigue una estrategia diferente a la tasa de error interna. Haciendo uso de dos bases de datos, una base de datos (A) cuyos elementos poseen una clasificación acertada y la otra base de datos (B) a cuyos elementos se les asigna una clasificación, proceso al que se le calcula la tasa de error, siguiendo los siguientes pasos:

- ✓ Para cada elemento de B, se calcula similitud mediante distancia de edición y costos especificados (unitarios, prefijados, aprendidos) con todos los elementos A.

- ✓ Si, la(s) secuencia encontrada en A, posee diferente clasificación con el elemento de B, se contabiliza como error.

- ✓ El resultado es la cantidad de errores encontrados y su tasa de error (cantidad de errores/cantidad elementos de la base de datos).

2.4 PROPUESTA DEL SISTEMA.

2.4.1 DESCRIPCIÓN GENERAL.

Con el objetivo de mejorar la calidad de las investigaciones llevadas a cabo en el Instituto de Ciencia Animal se pretende desarrollar una aplicación que de manera eficaz, ayude al estudio de las comunidades microbianas mediante la clasificación taxonómica a partir de secuencias de ADN.

De forma general se pretende desarrollar una herramienta de clasificación basada en el k-vecino más cercano, empleando la distancia Levenshtein para diferentes costos de las operaciones de edición. El sistema debe poseer las siguientes funcionalidades principales:

- ✓ Aprendizaje de un Transductor sobre distribuciones conjuntas o condicionales, según las propuestas de Ristad-Yianilos y José Oncina respectivamente.

- ✓ Clasificación de un microorganismo o una base de datos de microorganismos según las técnicas de k-vecino más cercano con costos prefijados o aprendidos según las distribuciones condicional o conjunta.

- ✓ Cálculo de tasa de error de una base de datos según las definiciones: tasa de error interna y tasa de error basada en k-vecino más cercano.

2.4.2 ANÁLISIS COMPARATIVO.

Considerando otras alternativas existentes para lograr la clasificación la mayoría de las cuales son accesibles en internet por medios de sitios que ofrecen herramientas bioinformáticas de este tipo y en algunos casos es posible descargar como es el caso del paquete FASTA y los algoritmos BLAST, se ha llegado a la conclusión de que la propuesta de este sistema es la más acertada por las siguientes razones:

- ✓ No permiten hacer búsquedas masivas ni personalizar las bases de datos sobre las cuales se clasifica. Entiéndase el término masivas a más de 500 secuencias.

- ✓ Las secuencias son enviadas al servidor sin ningún tipo de cifrado, lo que representa un problema si se quiere mantener los datos privados.

- ✓ Utilizan métodos heurísticos con lo que se sacrifica precisión por velocidad. Estos métodos no garantizan llegar a un resultado óptimo.

- ✓ Las matrices de sustitución que ellos incorporan no reflejan la realidad de los datos contenidos en bases de datos personalizadas.

- ✓ Se ha comprobado que la clasificación por medio de distancia de edición estocástica a través de transductores sin memoria aprendidos, sobre distribuciones condicionales, es más exacta y se logran resultados más óptimos.

2.4.3 MODELO DE DOMINIO.

Un modelo de dominio captura los tipos más importantes de objetos en el contexto del sistema. Los objetos del dominio representan las cosas que existen o eventos que ocurren en el entorno en el que trabaja el sistema [10]. El modelo de dominio se describe mediante diagramas UML (específicamente mediante diagramas de clases). Ver anexo 1 para observar el modelo de dominio correspondiente a la herramienta propuesta.

2.4.3.1 DESCRIPCIÓN TEXTUAL DEL MODELO DE DOMINIO.

El proceso de clasificación de secuencias puede ser representado utilizando algoritmos de homología basada en distancia de edición, donde los costos de las operaciones de edición pueden ser predefinidos o aprendidos a través de métodos de aprendizaje en la forma de un transductor sin memoria (Transductor de estado finito estocástico).

El cálculo de la distancia de edición entre un par de cadenas entrada – salida, puede ser calculada mediante distancia de edición clásica o distancia de edición estocástica, representada como una matriz de probabilidades según las operaciones de edición del alfabeto del par de cadenas.

El algoritmo de entrenamiento, da como resultado un transductor y se realiza mediante el algoritmo de Expectation-Maximization, partiendo de una Base de Datos de pares de cadena, donde para cada par de esta base de datos se obtiene un transductor base para el próximo paso. Una base de datos de pares se conforma mediante cada elemento de la base de datos y su homólogo en dicha base de datos.

Como sustento o prueba de consistencia para ver los resultados de la clasificación se determina una tasa de error, que puede ser Interna, sustentada en la determinación del vecino más cercano de cada elemento, comparando su etiqueta de clasificación con este, y contabilizar la diferencia como un error, o la tasa de error basada en la obtención de vecinos, que se utilizan dos BD, una de Prueba y la de entrenamiento.

2.5 ESPECIFICACIÓN DE LOS REQUISITOS DE SOFTWARE.

2.5.1 REQUERIMIENTOS FUNCIONALES.

Los requerimientos funcionales son capacidades o condiciones que el sistema debe cumplir. Permiten expresar una especificación más detallada de las responsabilidades del sistema en cuestión. Se mantienen invariables, sin importarle con que propiedades o cualidades se relacionen. Los requerimientos funcionales del software propuesto son los siguientes:

1. Realizar Aprendizaje de un Transductor sin memoria.

1.1 Definir entrada de datos de aprendizaje.

1.1.1 Importar BDA.

1.1.2 Realizar conexión con BDA.

1.2 Mostrar avance de la operación.

1.3 -Mostrar resultado de la operación.

1.4- Salvar resultado.

2. Realizar Clasificación.

2.1 Clasificar una BD de secuencias.

2.1.1 Definir entrada de datos.

2.1.1.1 Importar BDA.

2.1.1.2 Realizar conexión con BDA.

2.1.1.3 Importar BDC.

2.1.1.4 Realizar conexión con BDC.

2.1.2 Mostrar avance de la operación.

2.1.3 Mostrar resultado de la operación.

2.1.4 Salvar resultado.

2.2 Clasificar secuencia.

2.2.1 Definir entrada de datos.

2.2.1.1 Importar BDA.

2.2.1.2 Realizar conexión con BDA.

2.2.2 Mostrar avance de la operación.

2.2.3 -Mostrar resultado de la operación.

3. Determinar TE de una BD.

3.1. Determinar TEI de la BD.

3.1.1 Definir entrada de datos.

3.1.1.1 Importar BDP.

3.1.1.2 Realizar conexión con BDP.

3.1.2 Mostrar avance de la operación.

3.1.3 Mostrar resultado de la operación.

3.1.4 Salvar resultado.

3.2. Determinar TE por el k vecino más próximo (KNN).

3.2.1 Definir entrada de datos.

3.2.1.1 Importar BDA.

3.2.1.2 Realizar conexión con BDA.

3.2.1.3 Importar BDP.

3.2.1.4 Realizar conexión con BDP.

3.2.2 Mostrar avance de la operación.

3.2.3 Mostrar resultado de la operación.

3.2.4 Salvar resultado.

4. Realizar Alineamiento de secuencias.

4.1 Mostrar resultado de la operación.

5. Filtrar Base de Datos.

5.1 Importar BD.

5.2 Realizar conexión con BD.

5.3 Salvar BD obtenida.

6. Seccionar Base de Datos.

6.1 Importar BD.

6.2 Realizar conexión con BD.

6.3 Salvar BD obtenida.

7. Eliminar elementos duplicados.

7.1 Importar BD.

7.2 Realizar conexión con BD.

7.3 Salvar BD obtenida.

8. Mostrar Transductor Almacenado.

8.1 Importar Transductor.

8.2 Visualizar Transductor.

2.5.2 REQUERIMIENTOS NO FUNCIONALES.

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable.

Apariencia: La herramienta deberá tener un diseño de interfaz amigable y agradable, de forma tal que el usuario haga uso de la misma sin dificultad alguna, ajustándose a los estándares establecidos para el desarrollo de un buen diseño.

Usabilidad: El sistema brindará la posibilidad de ser usado por cualquier persona cuyos conocimientos en cuanto al trabajo con los ordenadores sean básicos, prácticamente no son necesarios tampoco conocimientos especializados en biología para poder entender los resultados brindados por la aplicación.

Rendimiento: Un aspecto importante a tener en cuenta a la hora de utilizar la aplicación es la cantidad de memoria RAM de la PC, necesaria para el correcto funcionamiento del

programa. El tamaño de las cadenas con que se podrá trabajar, estará limitado como se mencionaba, por la disponibilidad de memoria RAM en el computador. La siguiente fórmula expresa la relación RAM-Tamaño de cadena.

RAM = **8** * **(LMC^2) / 1024^2**

Donde **LMC** es la Longitud Máxima de la Cadena con mayor tamaño en la BD.

Portabilidad: El sistema está concebido para que sea multiplataforma.

Software: Se debe disponer de cualquier sistema operativo que tenga instalado la máquina virtual de java (JVM). La aplicación se realizará en un ambiente Desktop, la base de datos es independiente de la aplicación. Se debe disponer de un gestor de BD de PostgreSQL.

Hardware: Para el desarrollo y puesta en práctica de la aplicación se requieren PC con los siguientes requisitos:

Para las PC clientes se necesita:

- ✓ Un procesador de 2.0 GHz o superior.
- ✓ La memoria está limitada por la fórmula **RAM = 8 * (LMC^2) / 1024^2**, explicada anteriormente.

2.6 MODELADO DEL SISTEMA.

El Modelado del sistema se centra fundamentalmente en estructurar los requisitos funcionales y no funcionales mediante casos de uso. Y definir los actores que participan en el sistema.

2.6.1 ACTORES DEL SISTEMA.

Un actor es una entidad externa del sistema que de alguna manera participa en la historia del caso de uso. Siempre se beneficia de la realización de un caso de uso. Estimula el sistema con eventos de entradas o recibe algo de él. O sea, es un rol de un

usuario, que puede intercambiar información o puede ser un recipiente pasivo de información y representa a un ser humano, a un software o a una máquina que interactúa con el sistema.

Actor	Descripción
Especialista	Es aquella persona que va a interactuar con la aplicación. Procesa sus datos y obtiene los resultados para su estudio.

Tabla 1: Definición de actores del sistema a automatizar.

2.6.2 CASOS DE USO DEL SISTEMA.

Los casos de uso son “fragmentos” de funcionalidad que el sistema ofrece para aportar un resultado de valor para sus actores. Son una secuencia de acciones que el sistema debe llevar a cabo. Se utilizan para obtener información de cómo debe trabajar el sistema, describen bajo la forma de acciones y reacciones el comportamiento de un sistema desde el punto de vista del usuario. Los casos de usos definidos son los siguientes:

Referencia	Caso de uso	Prioridad
CUS 1	Realizar aprendizaje.	Crítico
CUS 2	Realizar clasificación.	Crítico
CUS 3	Clasificar BDS	Crítico

CUS 4	Clasificar Secuencia.	Crítico
CUS 5	Realizar Alineamiento de secuencias	Opcional
CUS 6	Determinar TE.	Crítico
CUS 7	Determinar TEI.	Crítico
CUS 8	Determinar TE por KNN.	Crítico
CUS 9	Filtrar BD	Auxiliar
CUS 10	Seccionar BD	Auxiliar
CUS 11	Eliminar Elementos Duplicados	Auxiliar
CUS12	Mostrar Transductor Almacenado	Opcional

Tabla 2: Definición de los casos de uso del sistema.

2.6.3 DIAGRAMA DE CASOS DE USO DEL SISTEMA.

El modelo de Casos de Uso del sistema representa las funcionalidades deseadas y el entorno del sistema a través de actores y casos de uso, además sirve como un contrato entre los clientes y los desarrolladores. Este sienta las bases necesarias para el desarrollo del análisis y el diseño del sistema.

DIAGRAMA DE CASOS DE USO

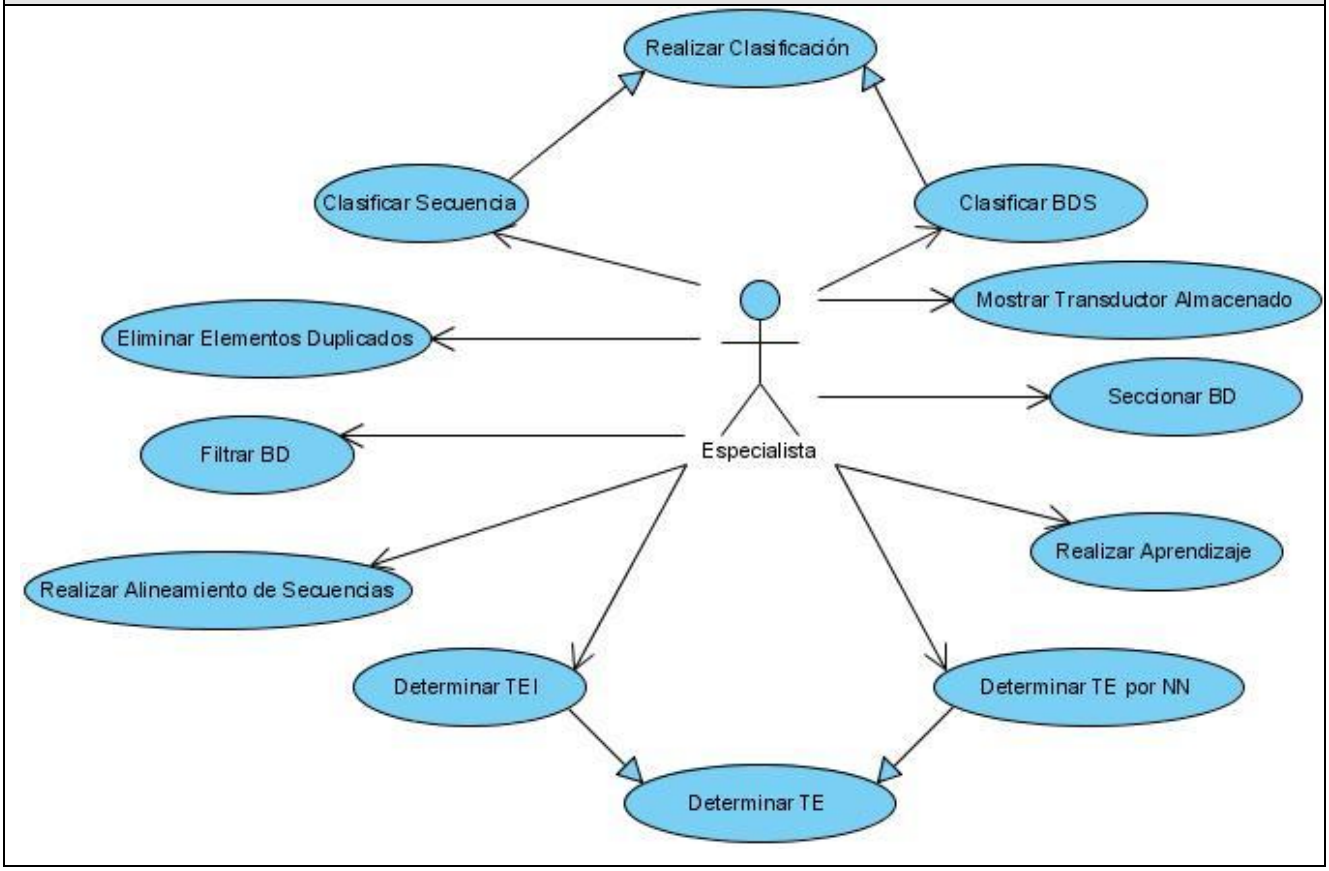


Figura 4: Diagrama de casos de uso del sistema.

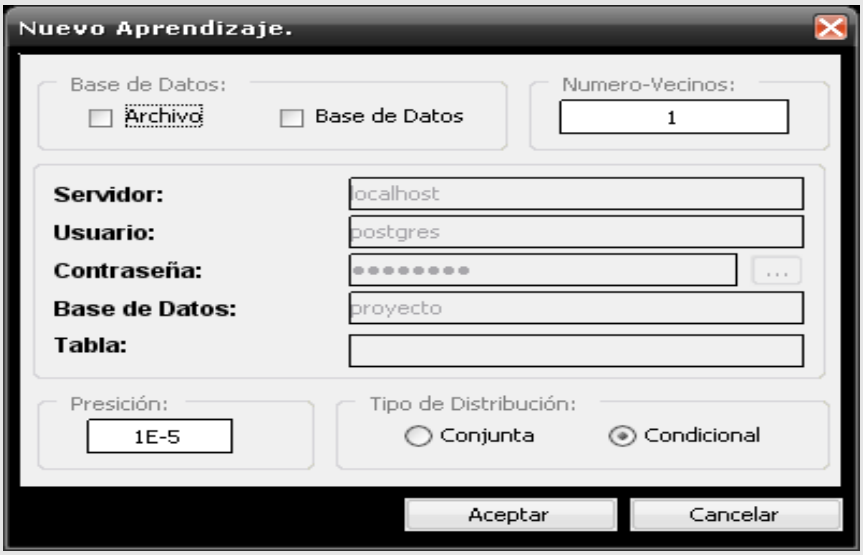
2.6.4 DESCRIPCIÓN DETALLADA DE LOS CASOS DE USO.

Con el propósito de lograr una mejor comprensión de los procesos a automatizar, se especifican los casos de uso del sistema mediante la descripción detallada de los mismos.

CU 1 :	Realizar Aprendizaje
Actores:	Especialista (Inicia)

Propósito:	Permitir al especialista realizar un aprendizaje para la posterior clasificación utilizando la Distancia de Edición Estocástica.
Resumen:	El caso de uso se inicia cuando el Especialista selecciona en el menú Aprendizaje la opción nuevo Aprendizaje y aparece la interfaz correspondiente. Luego el especialista escoge los criterios correspondientes y el sistema comienza la realización del aprendizaje.
Referencia:	R1 , R1.1, R1.2, R1.3, R1.4
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1-Selecciona en el menú Aprendizaje la opción Nuevo Aprendizaje.	1.1 – Muestra la interfaz correspondiente.
2-Carga la BD de la que desea realizar el aprendizaje o marca la opción para conectarse a ella e introduce los datos correspondientes, entra el número de vecinos para crear la BD de pares, la precisión para el transductor, y selecciona la distribución de probabilidad.	2.1 – Verifica la entrada de datos.

	2.2 – Crea la BD de pares para el aprendizaje.
	2.3 – Comienza el aprendizaje a partir de la BD de pares.
	2.4 - Muestra el avance de la operación.
	2.5 – Muestra un cuadro de diálogo preguntando si desea mostrar el transductor.
3 – Confirma la operación.	3.1 - Muestra el resultado.
	3.2 - Muestra un cuadro de diálogo preguntando si desea salvar.
4 Confirma la operación.	4.1 – Muestra el cuadro de diálogo para darle la posibilidad al Especialista de escoger el lugar donde desea Salvar.
5 El especialista escoge el camino deseado.	5.1 – El sistema salva los datos correspondientes a la operación.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
Acción 2.1	
	2.1- Muestra un cartel de error.
Acción 3	

3– Aborta la operación.	
Acción 4	
4 – Aborta la operación.	
Prototipo Interfaz 	

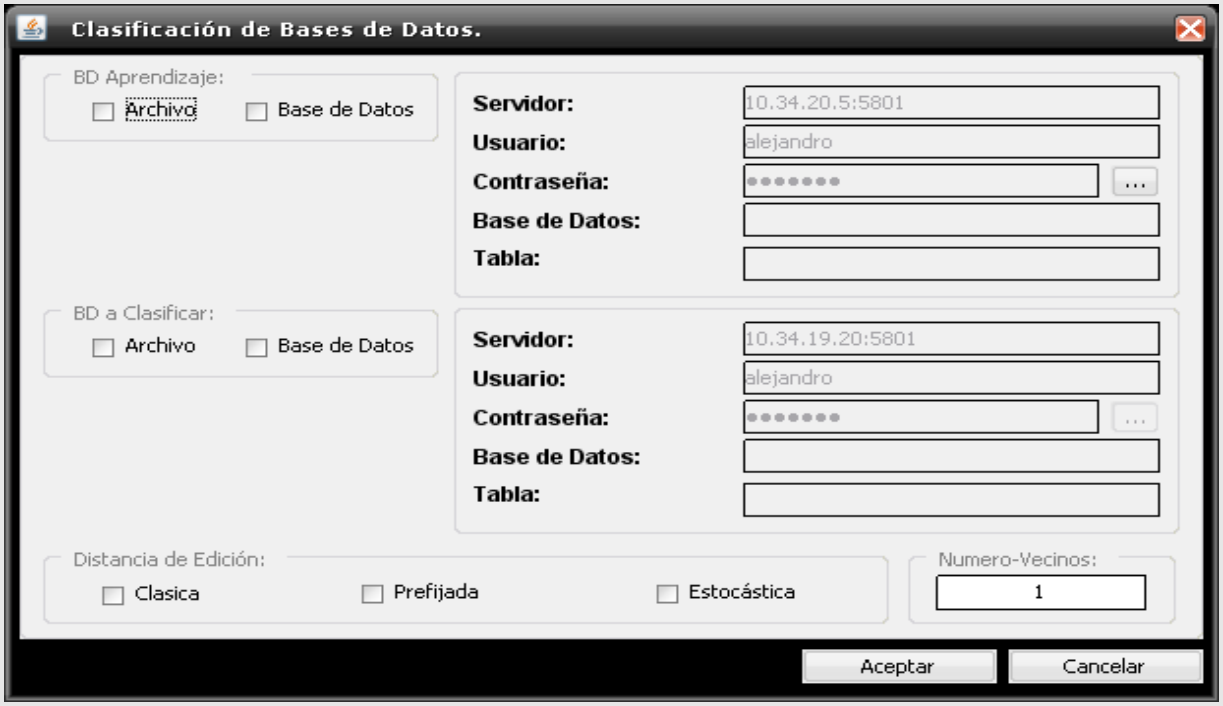
CU 2 :	Realizar Clasificación
Actores:	Especialista (Inicia)
Propósito:	Permitir al especialista realizar la clasificación Taxonómica.
Resumen:	El caso de uso se inicia cuando el Especialista selecciona en el menú Clasificación la opción Clasificar Secuencias o Clasificar Base de datos y aparece la interfaz correspondiente. Luego el

	especialista realiza la entrada de datos y el sistema comienza la clasificación.
Referencia:	R2 , R2.1, R2.2
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1 - Selecciona en el menú el tipo de clasificación (Clasificar Secuencia o Clasificación de Base de Datos).	1.1 – Muestra la interfaz correspondiente.
2 - Importa la BD de aprendizaje o realiza una conexión con ella.	
2 - Define la distancia de edición a utilizar (Clásica, Estocástica) y el método de búsqueda.	3.1- Valida la entrada de datos.
	3.2 - Muestra el avance de la operación.
	3.3 - Muestra el resultado de las operaciones.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
Acción 3.1	
	3.1 Muestra un cartel de error.

CU 3 :	Clasificar BDS
Actores:	Especialista (Inicia)
Propósito:	Permitir al especialista realizar la clasificación Taxonómica de una BDS de ADN.
Resumen:	El caso de uso se inicia cuando el Especialista selecciona la opción clasificación de Base de datos y aparece la interfaz correspondiente. Luego el especialista realiza la entrada de datos y el sistema comienza la clasificación.
Referencia:	R2.1 , R2.1.1, R2.1.2, R2.1.3
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
	1- Ver caso de uso Realizar Clasificación.
	2- Muestra un cuadro de diálogo preguntando si desea salvar la BD clasificada.
3 – Confirma la operación.	3.1 – Muestra el cuadro de diálogo para darle la posibilidad al Especialista de escoger el lugar donde desea Salvar.
4 – El especialista escoge el camino deseado.	4.1 - El sistema salva los datos correspondientes a la operación.

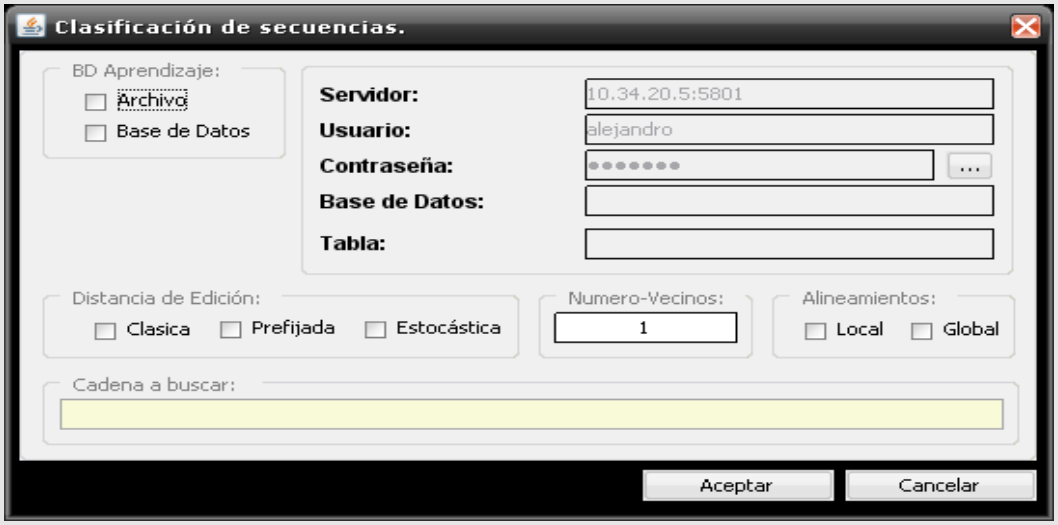
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
3 – Aborta la operación.	

Prototipo Interfaz



CU 4 :	Clasificar Secuencia.
Actores:	Especialista (Inicia)
Propósito:	Permitir al especialista realizar la clasificación Taxonómica de una Secuencia de ADN.
Resumen:	El caso de uso se inicia cuando el Especialista selecciona la opción clasificación de Secuencias

	y aparece la interfaz correspondiente. Luego el especialista realiza la entrada de datos y el sistema comienza la clasificación.
Referencia:	R2.2 , R2.2.1, R2.2.2, R2.2.3
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1- Inserta la Secuencia a clasificar y escoge el tipo de alineamiento (Global, local o ambos), opcionalmente.	1.1 – Ver caso de uso Realizar clasificación.
Prototipo Interfaz	



CU 5 :	Realizar Alineamientos de secuencias
Actores:	Especialista (Inicia)
Propósito:	Permitir al especialista realizar un alineamiento entre

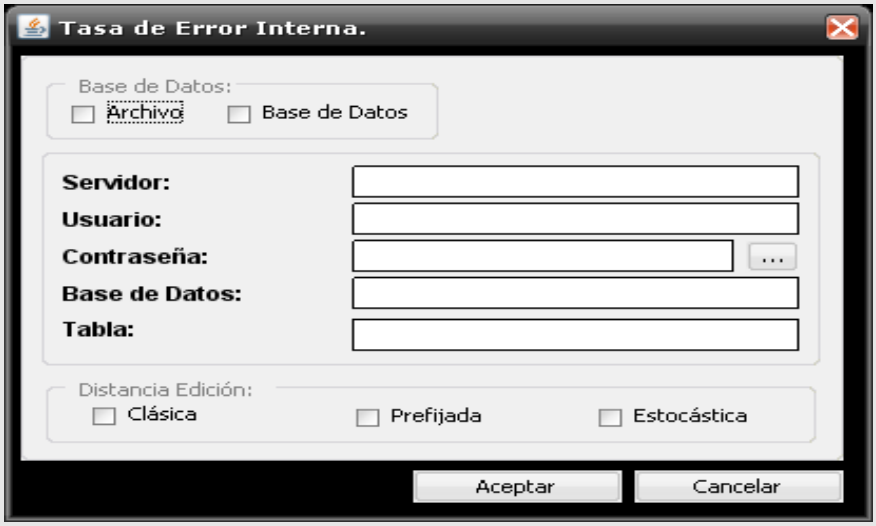
	dos secuencias de ADN. (Global o local).
Resumen:	El caso de uso se inicia cuando el Especialista selecciona en el menú Alineamientos la opción Alineamientos locales y globales y aparece la interfaz correspondiente. Luego el especialista inserta las secuencias a alinear y escoge el tipo de alineamiento para dar comienzo a la realización de la operación.
Referencia:	R4 , R4.1
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1 - Selecciona en el menú Alineamientos la opción Alineamientos locales y globales.	1.1 Muestra la interfaz correspondiente.
2 - Inserta las secuencias a alinear y selecciona el tipo de alineamiento.	2.1 Muestra el resultado del alineamiento.
Prototipo Interfaz	

CU 6 :	Determinar TE.
Actores:	Especialista (Inicia)
Propósito:	Permitir al especialista Determinar la Tasa de Error de una BD de secuencias cultivadas.
Resumen:	El caso de uso se inicia cuando el Especialista selecciona en el menú Tasa de Error la opción Tasa de Error Interna o Tasa de error basada en K-NN y aparece la interfaz correspondiente. Luego el especialista realiza las entradas correspondientes y el sistema comienza la operación.
Referencia:	R3 , R3.1, R3.2
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1- Selecciona en el menú Tasa de Error el tipo de TE que desea determinar (tasa de Error Interna o Tasa de Error basada en K-NN).	1.1 - Muestra la interfaz correspondiente.
2- Importa la BDP o realiza una conexión con ella y Selecciona el tipo de distancia a utilizar (Clásica, Prefijada, Estocástica).	2.1 - Si el usuario selecciona Clásica o Estocástica, el sistema muestra el avance de la operación.

	2.2 - Muestra el resultado de las operaciones.
	2.3 – Muestra un cuadro de diálogo preguntando si desea salvar.
3 - Confirma la operación.	3-1 – Muestra el cuadro de diálogo para darle la posibilidad al Especialista de escoger el lugar donde desea Salvar.
4 - El especialista escoge el camino deseado.	4.1 – El sistema salva los datos correspondientes a la operación.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
Acción 2.1	
	2.1 - Muestra la interfaz de Distancia prefijada para la entrada de las distancias.
2.1.1 - Inserta los valores de distancia deseados.	
Acción 3	
3- Aborta la operación.	

CU 7 :	Determinar TEI
Actores:	Especialista (Inicia)
Propósito:	Permitir al especialista Determinar la Tasa de Error Interna de una BD de secuencias.
Resumen:	El caso de uso se inicia cuando el Especialista selecciona en el

	menú Tasa de Error la opción Tasa de Error Interna y aparece la interfaz correspondiente. Luego el especialista realiza las entradas correspondientes y el sistema comienza la operación.
Referencia:	R3.1 , R3.1.1, R3.1.2, R3.1.3, R3.1.4
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
	1 – Ver caso de uso Determinar TE.
Prototipo Interfaz	

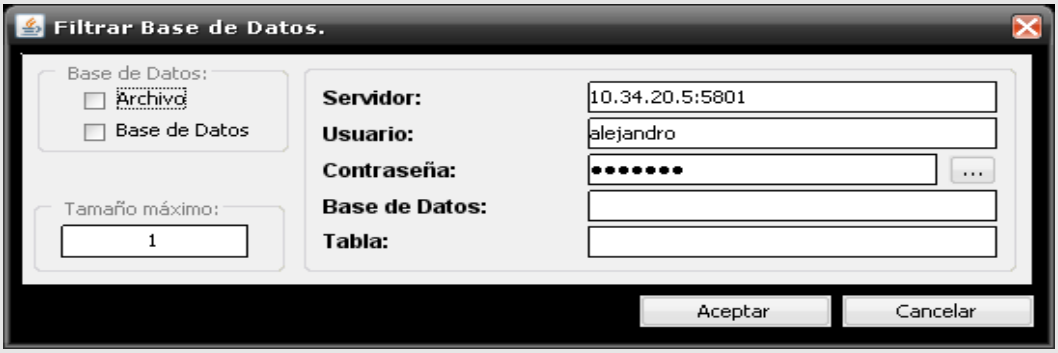


CU 8 :	Determinar TE por KNN
Actores:	Especialista (Inicia)
Propósito:	Permitir al especialista Determinar la Tasa de Error basada en K-NN de una BD de secuencias.

Resumen:	El caso de uso se inicia cuando el Especialista selecciona en el menú Tasa de Error la opción Tasa de Error basada en K-NN y aparece la interfaz correspondiente. Luego el especialista realiza las entradas correspondientes y el sistema comienza la operación.
Referencia:	R3.2 , R3.2.1, R3.2.2, R3.2.3, R3.2.4
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1- Importa la BD de Aprendizaje o realiza una conexión con ella.	
2- Define el criterio de búsqueda con el número de vecinos.	3 – Verifica que los datos sean correctos.
	1- Ver caso de uso Determinar TE.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	3- Muestra un cartel de error especificando que la entrada de datos es incorrecta.
Prototipo Interfaz	

CU 9 :	Filtrar BD
Actores:	Especialista (Inicia)
Propósito:	Permitir al especialista Filtrar una BD teniendo en cuenta el tamaño máximo de la secuencia de ADN.
Resumen:	El caso de uso se inicia cuando el Especialista selecciona en el menú Base de Datos la opción Filtrar Base de Datos y aparece la interfaz correspondiente. Luego el especialista realiza las entradas correspondientes y el sistema comienza la operación.
Referencia:	R5 , R5.1, R5.2
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**

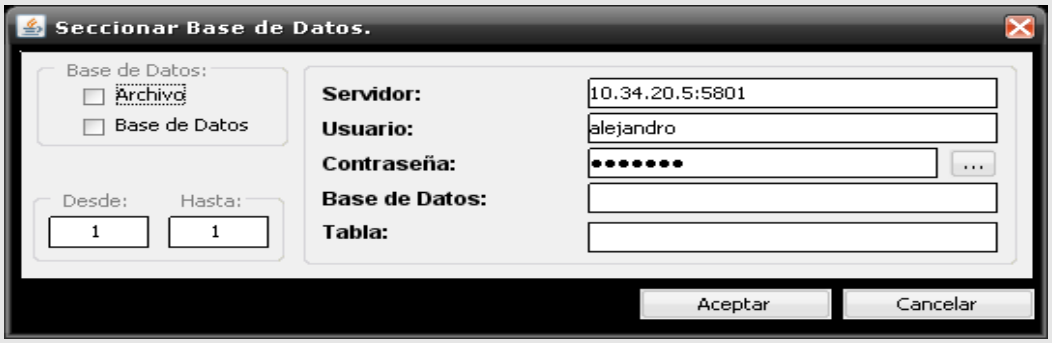
1- Selecciona en el menú Base de Datos la opción Filtrar Base de datos.	1.1 – Muestra la interfaz correspondiente.
2 - Introduce el valor máximo de caracteres para las secuencia de ADN.	2.1 – Verifica que los datos sean correctos.
	2.2 – Muestra un cuadro de diálogo preguntando si desea salvar.
3- Confirma la operación.	3.1 – Muestra el cuadro de diálogo para darle la posibilidad al Especialista de escoger el lugar donde desea Salvar.
4 - El especialista escoge el camino deseado.	4.1– El sistema salva los datos correspondientes a la operación.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
Acción 2.1	
	2.1- Muestra un cartel de error especificando que la entrada de datos es incorrecta.
Acción 3	
3- Aborta la operación.	
Prototipo Interfaz	



CU 10:	Seccionar BD.
Actores:	Especialista (Inicia)
Propósito:	Permitir al especialista Seccionar una BD teniendo en cuenta inicio y fin del intervalo de la sección.
Resumen:	El caso de uso se inicia cuando el Especialista selecciona en el menú Base de Datos la opción Seccionar Base de datos y aparece la interfaz correspondiente. Luego el especialista realiza las entradas correspondientes y el sistema comienza la operación.
Referencia:	R6 , R6.1, R6.2, R6.3
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1- Selecciona en el menú Base de Datos la opción Seccionar Base de datos.	1.1– Muestra la interfaz correspondiente.

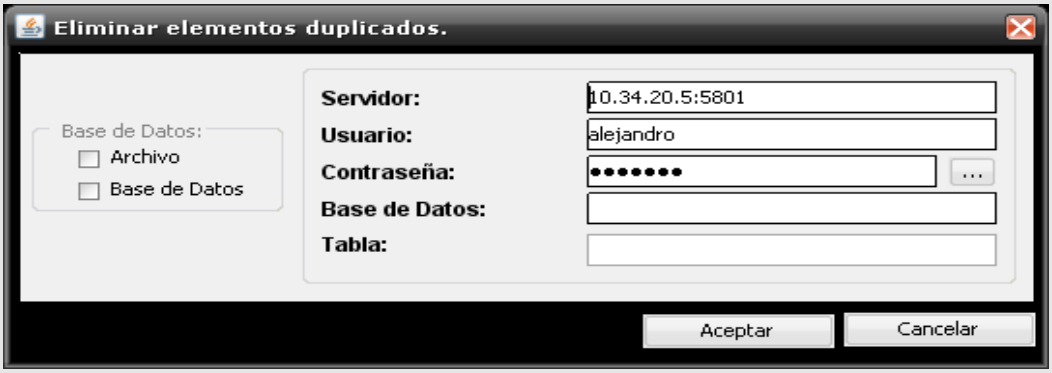
2 - Introduce el inicio y fin del intervalo para seccionar la BD.	2.1 - Verifica que los datos sean correctos.
	2.2 – Realiza la operación y Muestra un cuadro de diálogo preguntando si desea salvar.
3- Confirma la operación.	3.1 – Muestra el cuadro de diálogo para darle la posibilidad al Especialista de escoger el lugar donde desea Salvar.
4 - El especialista escoge el camino deseado.	4.1 – El sistema salva los datos correspondientes a la operación.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
Acción 2.1	
	2.1- Muestra un cartel de error especificando que la entrada de datos es incorrecta.
Acción 3	
3- Aborta la operación.	

Prototipo Interfaz



CU 11:	Eliminar Elementos Duplicados.
Actores:	Especialista (Inicia)
Propósito:	Permitir al especialista Eliminar elementos duplicados en una BD.
Resumen:	El caso de uso se inicia cuando el Especialista selecciona en el menú Base de Datos la opción Eliminar Elementos Duplicados y aparece la interfaz correspondiente. Luego el especialista realiza las entradas correspondientes y el sistema comienza la operación.
Referencia:	R7 , R7.1, R7.2, R7.3
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1- Selecciona en el menú Base de Datos la opción Eliminar elementos Duplicados.	1.1– Muestra la interfaz correspondiente.
2 – Importa la BD o realiza una conexión con ella.	2.1- Verifica que los datos sean correctos.
	2.2 – Realiza la operación y muestra un cuadro de diálogo preguntando si desea salvar.
3- Confirma la operación.	3.1 – Muestra el cuadro de diálogo para darle la posibilidad al especialista de escoger el lugar donde desea Salvar.
4 - El especialista escoge el camino deseado.	4.1 – El sistema salva los datos correspondientes a la operación.

Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
Acción 2.1	
	2.1- Muestra un cartel de error especificando que la entrada de datos es incorrecta.
Acción 3	
3- Aborta la operación.	
Prototipo Interfaz	



CU 12:	Mostrar Transductor Almacenado.
Actores:	Especialista (Inicia)
Propósito:	Permitir al especialista visualizar un transductor aprendido con anterioridad.
Resumen:	El caso de uso se inicia cuando el especialista selecciona en el menú Transductores la opción Mostrar Transductor Almacenado. Luego el especialista selecciona el camino donde se

	encuentra el fichero del transductor en el disco y el sistema realiza la operación.
Referencia:	R8.1 , R8.2
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1- Selecciona en el menú Transductores la opción Mostrar Transductor almacenado.	1.1– Muestra el cuadro de diálogo para seleccionar el camino del fichero perteneciente al transductor.
2 – Importa el transductor correspondiente.	2.1- Verifica que los datos sean correctos.
	2.2 – Realiza la operación y Muestra en la vista principal el transductor correspondiente.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
Acción 2.1	
	2.1- Muestra en pantalla un mensaje de error especificando que la entrada de datos es incorrecta.

2.7 CONCLUSIONES.

En el presente capítulo se realizaron las principales actividades del flujo de trabajo de requerimientos, propuesto por la metodología OpenUP para el desarrollo de software, tales como definición de los requisitos funcionales y no funcionales, especificación de los casos de uso de sistema y su descripción detallada, realización del diagrama de casos de usos del sistema y a modo de extensión se agregó la realización de un modelo de dominio con el objetivo de comprender mejor el entorno del problema.

CÁPITULO III. DISEÑO DEL SISTEMA.

Durante el diseño se modela el sistema y se le da la forma mediante la arquitectura, de forma tal que soporte todos los requisitos, tanto los requisitos funcionales como los no funcionales, creando una entrada apropiada y un punto de partida para las actividades de implementación, dando la posibilidad de obtener un producto con calidad y que satisfaga las necesidades del cliente. En el presente capítulo se dará una descripción del estilo arquitectónico utilizado para el desarrollo de la aplicación, se describirán los patrones de diseño empleados y los diagramas de clases de diseño e interacción de los casos de usos principales. Se realizará el diseño de la BD, el diagrama de despliegue y el tratamiento de errores.

3.1 ESTILO ARQUITECTÓNICO UTILIZADO.

Para el desarrollo de la aplicación se utilizó el estilo arquitectónico de llamada y retorno, específicamente Arquitectura en tres capas. El estilo en capas específica una organización jerárquica donde cada capa proporciona servicios a la capa inmediatamente superior y se sirve y apoya de las prestaciones que le brinda la capa inmediata inferior. En la práctica, las capas suelen ser entidades complejas, compuestas de varios paquetes o subsistemas.

En los sistemas del proyecto que se utiliza el estilo en capas se hace referencia en tres puntos de la definición de este patrón:

- ✓ **Contexto:** sirve para tratar sistemas grandes y complejos y se maneja la complejidad vía la descomposición.
- ✓ **Problema:** el problema es como estructurar la información para soportar los requerimientos de mantenimiento, reusabilidad, escalabilidad y robustez.
- ✓ **Solución:** componer la solución en una serie de capas, donde cada una debe ocuparse de un nivel del problema y tiene poca cohesión con las demás.

En la arquitectura general de capas queda claro que el hecho de dividir las tareas permite organizar mejor el desarrollo. La poca cohesión entre capas presentada por este patrón de diseño, nos permite tener libertades a la hora de modificar o mejorar en nuevas versiones el producto, inclusive permitiéndonos cambiar la capa de presentación ya sea en un programa desktop o una página Web. Generalmente este patrón posee tres capas principales, aunque dependiendo del contexto pueden agregarse.

A continuación se muestra la estructura de capas de la aplicación.

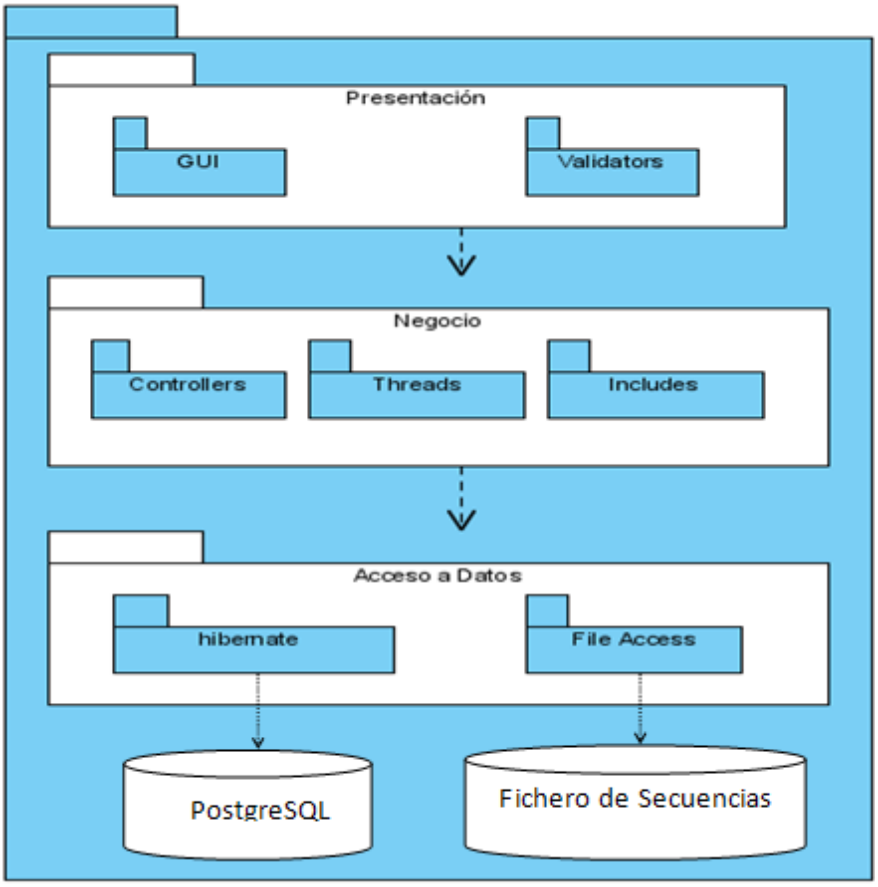


Figura 5: Estructura del sistema en capas.

Presentación: Es la capa que ve el usuario (hay quien la denomina "capa de usuario"), es la encargada de presentar el sistema al usuario, le comunica la información y captura la información del usuario en un mínimo de proceso (realiza un filtrado previo para comprobar que no hay errores de formato). Esta capa se comunica únicamente con la capa de negocio. También es conocida como interfaz gráfica y debe tener las características de ser amigable, entendible y fácil de usar para el usuario. La capa presentación, fue separada en dos paquetes. El primero, contiene las Interfaces Gráficas de Usuarios, por sus siglas en inglés (GUI) encargadas de de la interacción entre el usuario y la aplicación. El segundo paquete Validadores, en inglés Validators, contiene las clases empleadas para las validaciones de algunos de los formularios.

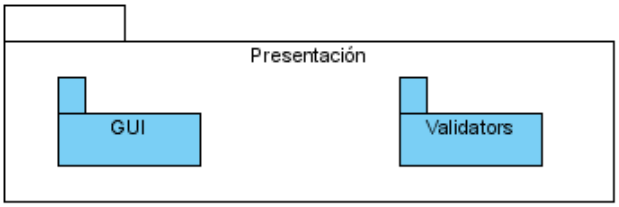


Figura 6: Paquetes de la capa Presentación.

Entre las clases de estos paquetes se establecen relaciones de tipo asociación que no fueron mostradas en el diagrama para no cargarlo de información irrelevante para este nivel de abstracción.

Negocio: Capa que contiene todas las clases encargadas de garantizar la lógica de la aplicación. Según el patrón Facade, se ha trabajado de tal forma que esta capa controle las peticiones desde la capa de presentación a través de un solo punto de entrada para el control del flujo en el sistema.

La capa Negocio (o lógica de aplicación), fue separada en tres paquetes, **Controladores, Hilos e Incluidos**, aunque el paquete incluido contiene algunos más, con el objetivo de detallar un poco la agrupación de las clases. Cada uno de estos paquetes contiene clases que son las encargadas de llevar a cabo el flujo de la

aplicación, para procesar las peticiones de los usuarios y preparar la información que va a ser enviada a la capa Presentación, para ser mostrada a través de las GUI.

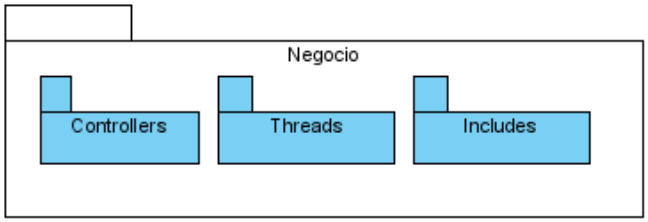


Figura 7: Paquetes de la capa Negocio.

Acceso a Datos: En esta capa es donde se accede a los datos y se manejan los mismos. Reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio. Esta capa está agrupada en dos paquetes, aunque estos contienen otros con el objetivo de detallar más la capa, que no se muestran para no cargar el diagrama de información para este nivel de abstracción. Se encuentra el paquete **Hibernate**, que contiene todas las acciones relacionadas con el acceso a los datos a través de la BD, utilizando el Framework Hibernate. El paquete **Acceso con Fichero**, que contiene las acciones para el trabajo con la persistencia de los datos a través de ficheros.

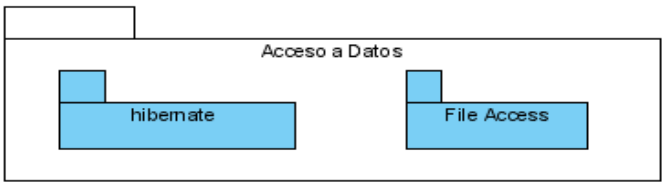


Figura 8: Paquetes de la capa Acceso a Datos.

Los patrones de uso común al aplicar este estilo son: Facade, Adapter, Bridge y Strategy. Las ventajas del estilo en capas son obvias. Primero que nada, el estilo soporta un diseño basado en niveles de abstracción crecientes, lo cual a su vez permite a los desarrolladores la partición de un problema complejo en una secuencia de pasos incrementales. En segundo lugar, el estilo admite muy naturalmente optimizaciones y

refinamientos. En tercer lugar, proporciona amplia reutilización. Al igual que los tipos de datos abstractos, se pueden utilizar diferentes implementaciones o versiones de una misma capa en la medida que soporten las mismas interfaces de cara a las capas adyacentes, brindando la posibilidad de reemplazar de cuajo una capa sin afectar a las restantes. Esto conduce a la posibilidad de definir interfaces de capa estándar, a partir de las cuales se pueden construir extensiones o prestaciones específicas.

3.2 PRINCIPALES PATRONES DE DISEÑO UTILIZADOS.

Los Patrones de Diseño son la base para la búsqueda de soluciones a problemas en el desarrollo de software y expresan esquemas para definir las estructuras en su diseño. Ofrecen soluciones y aseguran que la misma ha sido efectiva y reusable (se puede aplicar a diferentes problemas de diseño en distintas circunstancias) en la solución de problemas anteriores de forma satisfactoria, facilitando la búsqueda de soluciones para resolver los problemas que en ellos se presenta.

3.2.1 PATRÓN DELEGATION.

Es un patrón de diseño estructural. Muestra cuando no existe la necesidad de emplear la herencia en el diseño de clases. La delegación es una forma de extender y reutilizar la funcionalidad de una determinada clase, escribiendo una clase adicional con funcionalidad extra que usa instancias de la clase original para proveer y satisfacer su propia funcionalidad. La delegación es más apropiada que la herencia en muchas situaciones. La herencia es útil para modelar relaciones de tipo es-un o es-una, pues estos tipos de relaciones son de naturaleza estática. Sin embargo, relaciones de tipo es-un-rol-ejecutado-por son mal modeladas con herencia. La solución general propuesta en este patrón es: incorporar la funcionalidad de la clase original usando una instancia de esta y llamando sus métodos. Un ejemplo del uso de este en la aplicación puede apreciarse a continuación.

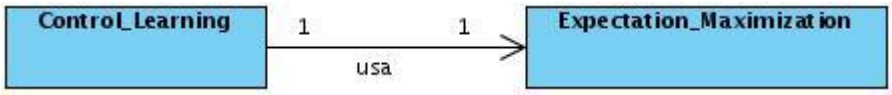


Figura 9: Patrón Delegation.

En la figura 9, se muestra la clase Control_Learning con rol Delegador, que usa la clase Expectation_Maximization, con el rol Delegado para reutilizar y extender el comportamiento de la clase. Este patrón puede complementarse o ser usado también unido al patrón Interfaz, el cual se muestra a continuación.

3.2.2 PATRÓN INTERFAZ.

Es un patrón de diseño estructural. Mantiene una clase (la interfaz) que usa datos y servicios provistos por otras clases independientes, para proveer un acceso uniforme. Esta clase interfaz provee a sus clases herederas acceso uniforme a métodos y atributos específicos, sin que deban saber a qué clase específica pertenecen. Un ejemplo de su uso en la herramienta puede apreciarse a continuación.

Figura 10: Patrón Interfaz

En la Figura anterior, la clase Control_Classification usa otra clase que implementa la interfaz xInterface, encargada de proveer la indirección que mantiene a la clase Control_Classification independiente de la clase proveedora de los servicios (clase Principal_form). Cumpliendo con el rol delegador, la clase Control_Classification, delegando la visualización de las diferentes operaciones en la interfaz xInterface que hace las veces de delegado. Por tanto la clase Control_Classification extenderá sus funcionalidades de mostrado de operaciones de conectores a través de xInterface. De esta forma se pueden crear varias instancias de la clase Control_Classification que usen

una interfaz xInterface diferente, representando operaciones distintas. Por lo general, los patrones Interfaz y Delegación son usados juntos.

3.2.3 PATRÓN FACADE.

El patrón de diseño Facade (Fachada) es de tipo estructural, sirve para proveer de una interfaz unificada sencilla que haga de intermediaria entre un cliente y una interfaz o grupo de interfaces más complejas. Conoce que clases son responsables de que peticiones y así, delega las peticiones a los objetos apropiados. El objetivo principal de uso de este patrón es ocultar todo lo posible la complejidad de un sistema o el conjunto de clases o componentes que lo forman, de forma que sólo se ofrezca un (o unos pocos) punto de entrada al sistema tapado por la fachada.

Se utilizó dicho patrón en el diseño con el objetivo de facilitar las ventajas de la arquitectura en capas, teniendo en cuenta que es uno de los más comunes en el mismo. En la siguiente figura se muestra el diagrama de clases correspondiente.

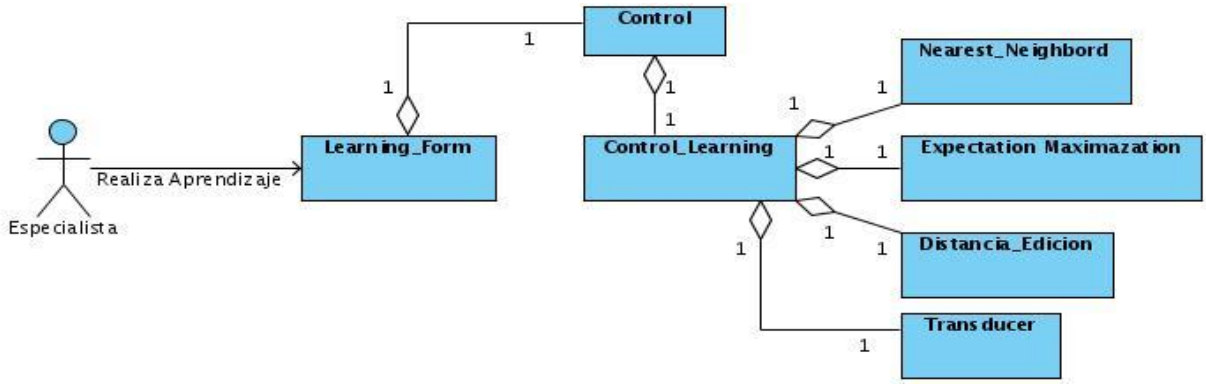


Figura 11: Patrón Facade.

3.2.4 PATRÓN STRATEGY.

Strategy (Estrategia) es uno de los patrones de diseño de comportamiento. Este patrón permite mantener un conjunto de algoritmos de los que el objeto cliente puede elegir aquel que le conviene e intercambiarlo según sus necesidades. Los distintos algoritmos

se encapsulan y el cliente trabaja contra un objeto Contexto. Como se ha dicho, el cliente puede elegir el algoritmo que prefiera de entre los disponibles o puede ser el mismo objeto contexto el que elija el más apropiado para cada situación, normalmente mediante algún parámetro que le envía el cliente.

Cualquier programa que ofrezca un servicio o función determinada, que pueda ser realizada de varias maneras, es candidato a utilizar el patrón Strategy. Puede haber cualquier número de estrategias y cualquiera de ellas podrá ser intercambiada por otra en cualquier momento, incluso en tiempo de ejecución.

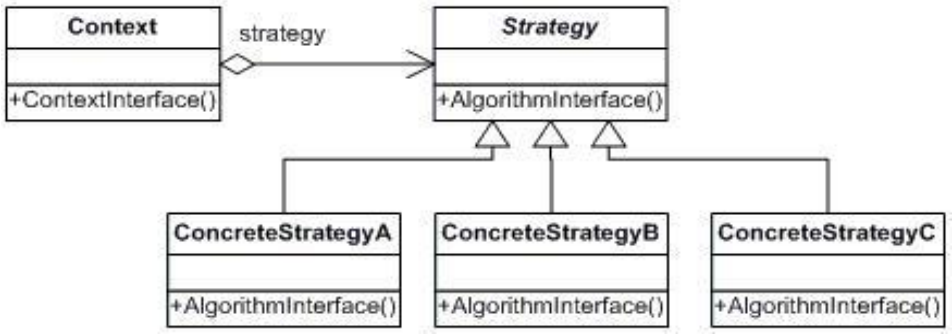


Figura 12: Patrón Strategy.

Strategy, declara una interfaz común para dar soporte a todos los algoritmos, Contexto utiliza esta interfaz para llamar al algoritmo definido por una Estrategia Concreta que implementa el algoritmo usando la interfaz Estrategia, y Contexto se configura con un objeto Estrategia Concreta sobre el que mantiene una referencia, además puede definir una interfaz que permita a Estrategia acceder a sus datos.

Lo importante de esta clase es que cada una de las estrategias que se diseñen tendrá que sobrescribir el método que identifica dicha estrategia y proveer un algoritmo concreto para la misma. Como se puede comprobar el funcionamiento de este patrón es muy simple y el añadir nuevas estrategias al programa es muy sencillo y apenas implica modificación de código alguna.

Con el objetivo de ilustrar el funcionamiento de este patrón en la aplicación se mostrará la estrategia para los algoritmos de distancia de edición.

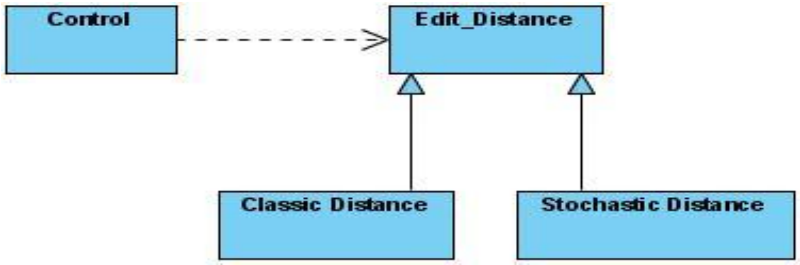


Figura 13: Patrón Strategy.

Además se usaron los patrones Generales de Asignación de Responsabilidades a Objetos (GRASP), Experto, Creador, Controlador, Alta Cohesión y Bajo Acoplamiento. Estos constituyen la base de la programación orientada a objetos (POO).

3.3 DIAGRAMAS DE CLASES DE DISEÑO.

Un diagrama de clases de diseño es un diagrama de estructura estática que describe gráficamente las especificaciones para las clases e interfaces de una aplicación, y además contiene información como clases, asociaciones, atributos, interfaces con sus operaciones y constantes, métodos o funciones, navegabilidad, dependencias, y multiplicidad en algunas de sus relaciones.

Los diagramas de clases del diseño se realizaron por cada caso de uso, aunque se muestran en el presente documento solo los correspondientes a los casos de uso críticos. Estos permiten que se describa detalladamente y de forma gráfica las especificaciones de las clases del software.

Diagramas de clases por CU. [Ver ANEXO 2**:** DIAGRAMAS DE CLASES POR CU.].

3.4 DESCRIPCIONES DE LAS CLASES PRINCIPALES.

Nombre: Expectation_Maximization

Atributo	**Tipo**
Iteraciones	int
Precision	double
Probabilidad	double
ObjtPresent;	XInterface
Para cada responsabilidad:	
Nombre:	GetIteraciones()
Descripción:	Muestra las iteraciones del algoritmo de aprendizaje.
Nombre:	GetPrecision()
Descripción:	Determina la convergencia del algoritmo.
Nombre:	GetProbabilidad()
Descripción:	Devuelve la probabilidad del transductor obtenido.
Nombre:	SetPrecision(double xPrecision)
Descripción:	Permite cambiar el valor de la precisión.
Nombre:	SetProbabilidad(double xProbabilidad)
Descripción:	Permite cambiar el valor de la probabilidad.
Nombre:	Expectation(Transducer Count, Transducer T, Trellis Alpha, Trellis Beta, MyString in, MyString out)
Descripción:	Permite modifica un transductor para cada par de cadena de la BD de pares.
Nombre:	LearnTransducer(Pair_Database db, String Tipo)
Descripción:	Permite obtener un transductor con valores aprendidos a partir

	de la BD de pares.

Nombre: Nearest_Neighbor	
Atributo	**Tipo**
Probability	double
ObjtPresent	xInterface
Para cada responsabilidad:	
Nombre:	Get_Probability()
Descripción:	Valor de distancia de edición para la secuencia que más se parece en la BD.
Nombre:	Pair_Database Create_PairDB(Distancia_Edicion Dist, Database db, int Cant)
Descripción:	Permite la creación de la BD de pares para el aprendizaje.
Nombre:	NearestNeighbor_SameClass(Database Db, Distancia_Edicion Dist, MyString in, int xId, String xClass)
Descripción:	Obtiene el vecino más cercano con la misma clase.
Nombre:	All_NearestNeighBour_SameClass(Database Db, Distancia_Edicion Dist, MyString in, int xId, String xClass, int Cant)
Descripción:	Permite obtener los vecinos de la misma clase ordenados de acuerdo a la distancia de edición.
Nombre:	All_NearestNeighBour(Database Db, Distancia_Edicion Dist, MyString in)
Descripción:	Permite obtener los vecinos de diferente clase ordenados de acuerdo a la distancia de edición.

Nombre:	NearestNeighbor(Database Db, Distancia_Edicion Dist, MyString in)
Descripción:	Permite obtener el vecino más cercano analizando la BD de pares completamente.
Nombre:	NearestNeighbor_Excluded(Database Db, Distancia_Edicion Dist, MyString in, int xId)
Descripción:	Permite obtener el vecino más cercano excluyendo el propio elemento.

Nombre: Trellis	
Para cada responsabilidad:	
Nombre:	Forward(Transducer t, MyString in, MyString out)
Descripción:	Determina las probabilidades para un par de cadena (entrada, salida), recorriendo la matriz hacia delante.
Nombre:	Backward(Transducer t, MyString in, MyString out)
Descripción:	Determina las probabilidades para un par de cadena (entrada, salida), recorriendo la matriz hacia atrás.

Nombre: Edit_Distance	
Para cada responsabilidad:	
Nombre:	abstract double Distance(MyString in , MyString out)
Descripción:	Se implementa en las clases hijas para el cálculo de la Distancia de edición en dependencia del tipo que sea.
Nombre: Classic_Distance	

Para cada responsabilidad:	
Nombre:	double Distance(MyString in, MyString out)
Descripción:	Permite realizar el cálculo de la Distancia clásica.

Nombre: Stochastic_Distance	
Para cada responsabilidad:	
Nombre:	Distance(MyString in, MyString out)
Descripción:	Permite realizar el cálculo de la Distancia de edición estocástica.

Nombre: ErrorRate	
Atributo	**Tipo**
CantErrors	int
xErrorRate	double
ObjtPresent	xInterface
xClasificator	Clasificator
Average_List	LinkedList<Double>
Neighbours_List	LinkedList<Data>
Distance_List	LinkedList<Double>
Clasification_List	LinkedList<String>
Para cada responsabilidad:	
Nombre:	Generate_InternalErrorRate(Database Db, Distancia_Edicion Dist)

Descripción:	Permite Determinar la Tasa de Error Interna.
Nombre:	Generate_ErrorRate_Based_on_NN(Database Train, Database Test, Distancia_Edicion Dist)
Descripción:	Permite Determinar la Tasa de Error teniendo en cuenta un solo vecino.
Nombre:	Generate_ErrorRate_Based_on_KNN(Database Train, Database Test, Distancia_Edicion Dist, int Cant)
Descripción:	Permite determinar la Tasa de error teniendo en cuenta los K vecinos. (K > 1).
Nombre:	Get_CantErrors()
Descripción:	Permite obtener la cantidad de errores en la BD.
Nombre:	Get_ErrorRate()
Descripción:	Determina la Tasa de error, independientemente del tipo que sea.
Nombre:	Get_Average_List()
Descripción:	Determina una lista de promedios, para complementar una de las funciones de TE.
Nombre:	Get_Average_List()
Descripción:	Determina una lista de vecinos, para complementar una de las funciones de TE.
Nombre:	Get_Distance_List()
Descripción:	Determina una lista de distancias, para complementar una de las funciones de TE.
Nombre:	Get_Clasification_List()
Descripción:	Determina una lista de clasificaciones, para complementar una de

	las funciones de TE.

3.5 DIAGRAMAS DE ITERACCIÓN.

Los diagramas de interacción describen secuencias de intercambios de mensajes entre los roles que implementan el comportamiento de un sistema y son utilizados para la modelación de los aspectos dinámicos de estos, proporcionan una vista integral de su comportamiento. Esto conlleva a modelar instancias concretas, componentes y nodos junto con los mensajes enviados entre ellos, que representan su interacción.

Los diagramas de interacción tienen dos formas de manifestarse:

- ✓ Diagramas de secuencia.
- ✓ Diagramas de colaboración.

Un diagrama de secuencia destaca la ordenación temporal de los mensajes; el diagrama de colaboración a su vez destaca la organización estructural de los objetos que envían y reciben mensajes.

Los diagramas de secuencia para el diseño de la aplicación propuesta muestran la secuencia de mensajes entre objetos durante un escenario concreto de algunos de los CU.

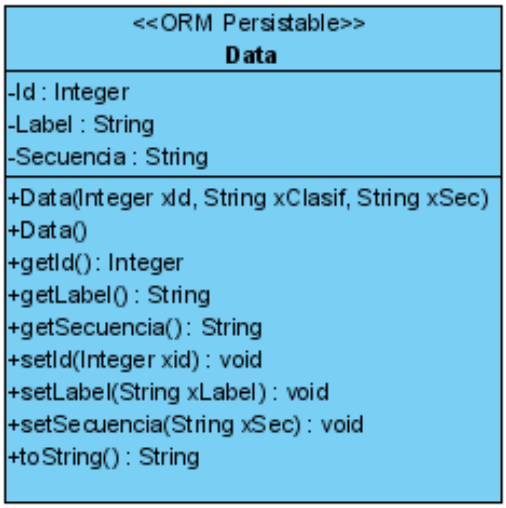
Diagramas de Secuencia. [Ver ANEXO 3: DIAGRAMAS DE SECUENCIA POR CU.

3.6 DISEÑO DE LA BASE DE DATOS.

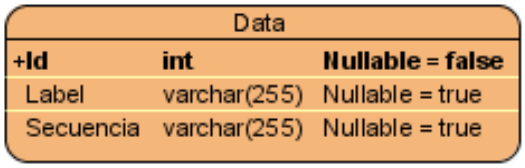
Para realizar el diseño de la base de datos de la aplicación se confeccionaron el diagrama de clases persistentes y el diagrama entidad relación (DER). Las clases persistentes son clases entidades que pueden mantener su valor en el espacio y el tiempo. En contrapartida las clases temporales son aquellas de las cuales el sistema se encarga de manejar y almacenar en tiempo de ejecución, razón por la que dejan de existir cuando termina la ejecución del programa.

A continuación se presenta el diagrama de clases persistentes que está compuesto por dichas clases y las correspondientes relaciones entre ellas y el diagrama entidad relación que es una representación de las tablas de la base de datos y sus relaciones. Aunque en el caso que se presenta es una BD muy pequeña, pues solo define una sola tabla, sobre la cual se implementan los diferentes algoritmos de la herramienta. Esto se debe a que la BD para las operaciones de la Herramienta simplemente tiene que contener una Tabla con el formato especificado (id, label, secuencia), sin importar el diseño estructural que tenga.

3.6.1 DIAGRAMA DE CLASES PERSISTENTES.



3.6.2 DIAGRAMA ENTIDAD RELACIÓN.



3.7 DESCRIPCIÓN DE LAS TABLAS DE LA BASE DE DATOS.

Nombre: Data		
Descripción: En esta tabla se almacenan las informaciones referentes a los microorganismos.		
Atributo	**Tipo**	**Descripción**
Id	integer	Identificador autogenerado de la secuencia.
Label	varchar	Clasificación del microorganismo identificado por la secuencia de ADN.
Secuencia	varchar	Secuencia de ADN.

3.8 DIAGRAMA DE DESPLIEGUE.

El modelo de despliegue describe la distribución física del sistema, en términos de funcionalidad entre los nodos y dispositivos que lo integran. A partir de este se puede observar lo siguiente:

- ✓ Los nodos están relacionados de modo que estas representan la forma en que se comunican en términos de protocolos.
- ✓ Cada nodo representa un equipo de cómputo, normalmente un procesador o dispositivo. Pueden estar incluidas impresoras, lectores de código y otros dispositivos.
- ✓ Puede describir diferentes configuraciones de red, incluidas las configuraciones para pruebas y para simulación.

A continuación se presenta el modelo de despliegue de la aplicación.

Figura 14: Diagrama de Despliegue

3.9 TRATAMIENTO DE ERRORES.

Con el objetivo de prevenir los errores que pueda cometer el especialista al interactuar con la aplicación, siempre se verifican los datos de entrada antes de ser utilizados, evitándose incidentes y procesamiento de datos erróneos.

La aplicación hace uso de mensajes para señalarle al especialista que debe rectificar los datos introducidos antes de llevar a cabo las operaciones solicitadas por el mismo. Es de gran importancia verificar la integridad de la información introducida, para de esta forma obtener un resultado de calidad. Un ejemplo de las validaciones que se realizan se evidencia cuando se importa la Base de Datos desde fichero con formato incorrecto.

Figura 15: Mensaje de información si los datos de entrada son incorrectos.

3.10 CONCLUSIONES.

En el presente capítulo se definió el diseño de la aplicación, descrito a través del estilo arquitectónico, así como los patrones de diseño utilizados y la fundamentación de su uso. El desarrollo de los diagramas de clases del diseño e interacción de los principales casos de uso, sentando las bases para una implementación detallada y organizada, de los procesos que debe realizar la aplicación. Se definió el diseño de la BD, especificando el diagrama de clases persistentes, el diagrama entidad relación y la descripción de las tablas contenidas en la misma. Se realizó además el modelo de despliegue, describiendo así la distribución física de la aplicación y se mostró la forma en que será realizado el tratamiento de errores, dejando de forma clara y concisa a través de todos los artefactos generados, el punto de partida para la implementación de la herramienta.

CAPÍTULO IV. IMPLEMENTACIÓN Y PRUEBA.

En el presente capítulo se realizará el modelo de implementación a partir de los resultados del flujo de trabajo de diseño anteriormente descrito, especificando la descripción de la aplicación en términos de componentes, así como el desarrollo de pruebas de funcionalidad, teniendo en cuenta principalmente, los casos de pruebas que describen el modelo de prueba de la herramienta.

4.1 IMPLEMENTACIÓN.

En la implementación se inicia con el resultado del diseño y se implementa el sistema en términos de componentes, es decir, ficheros de código fuente, scripts, ficheros de código binario, ejecutables y similares. El flujo de trabajo de implementación describe cómo los elementos del modelo del diseño se implementan en términos de componentes y cómo estos se organizan de acuerdo a los nodos específicos en el modelo de despliegue [10].

4.2 DIAGRAMA DE COMPONENTES.

Un diagrama de componentes es la representación y descripción de la forma en que los componentes físicos de un sistema serán separados. Son utilizados para modelar la vista estática de un sistema. Muestra la organización y las dependencias que existen entre un conjunto de componentes. Los componentes representan todos los tipos de elementos software que entran en la fabricación de una aplicación. UML define cinco **estereotipos** estándar que se aplican a los componentes:

Ejecutable: Especifica un componente que se puede ejecutar en un nodo.

Librería: Especifica una biblioteca de objetos estática o dinámica.

Tabla: Especifica un componente que representa una tabla de una base de datos.

Archivo: Especifica un componente que representa un documento que contiene código fuente o datos.

Documento: Especifica un componente que representa un documento.

A continuación se muestra el diagrama de componentes general de la aplicación y de cada una de las capas, de acuerdo a la arquitectura utilizada.

4.2.1 DIAGRAMA DE COMPONENTES DE LA ALPICACIÓN.

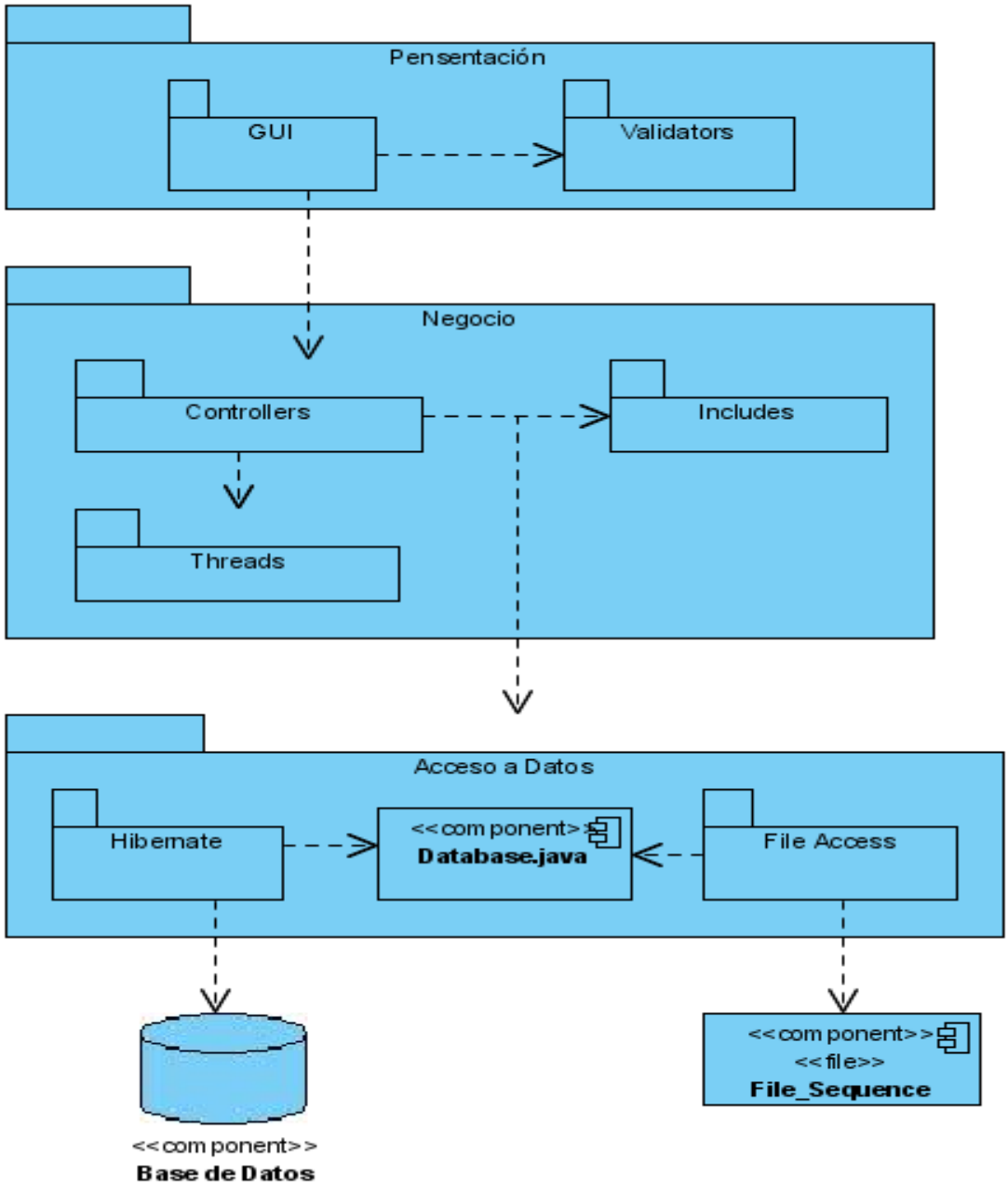


Figura 16: Diagrama de Componentes.

En el diagrama de componentes de la figura anterior se muestran los componentes por paquetes, de acuerdo a la tarea que desempeñan dentro de la aplicación. A continuación se describe cada uno de estos y los componentes relacionados.

Presentación: Este paquete contiene dos paquetes, GUI, Validadores que agrupan los ficheros fuentes que representan las Interfaces Gráficas de Usuarios y las Clases de Validaciones, respectivamente.

Negocio: En este paquete están agrupados tres paquetes que contienen los fuentes que se encargan de recibir las peticiones del usuario y envían las respuestas tras el proceso, un paquete de Hilos que facilita el trabajo brindando la posibilidad de realizar operaciones concurrentes, un paquete de controladores que permite repartir adecuadamente las funcionalidades y mantener un solo punto de entrada a la aplicación, como lo ejemplifica el patrón facade, además se encuentra el paquete Incluido que contiene los ficheros fuentes que agrupan todos los algoritmos y clases auxiliares para la solución del problema.

Acceso a Datos: Este paquete agrupa dos paquetes, Hibernate y File Access, que agrupan los ficheros fuentes que contienen las operaciones de manipulación de persistencia de los datos mediante el framework hibernate y la lógica de acceso a través de fichero respectivamente.

Database: Este componente representa al fichero fuente que permite almacenar las secuencias a partir de la BD o desde fichero, constituye la estructura de almacenamiento de los datos en memoria.

Base Datos: Este componente hace referencia la BD de secuencias que manipulará la aplicación.

File_Persist: Este componente es el fichero que contiene las secuencias de ADN que manipulará la aplicación, ejemplificando la persistencia a través de ficheros. Debe tener la siguiente estructura: (id, label, secuencia), separado por espacio.

4.2.1.1 DIAGRAMA DE COMPONENTES DE LA CAPA DE PRESENTACIÓN.

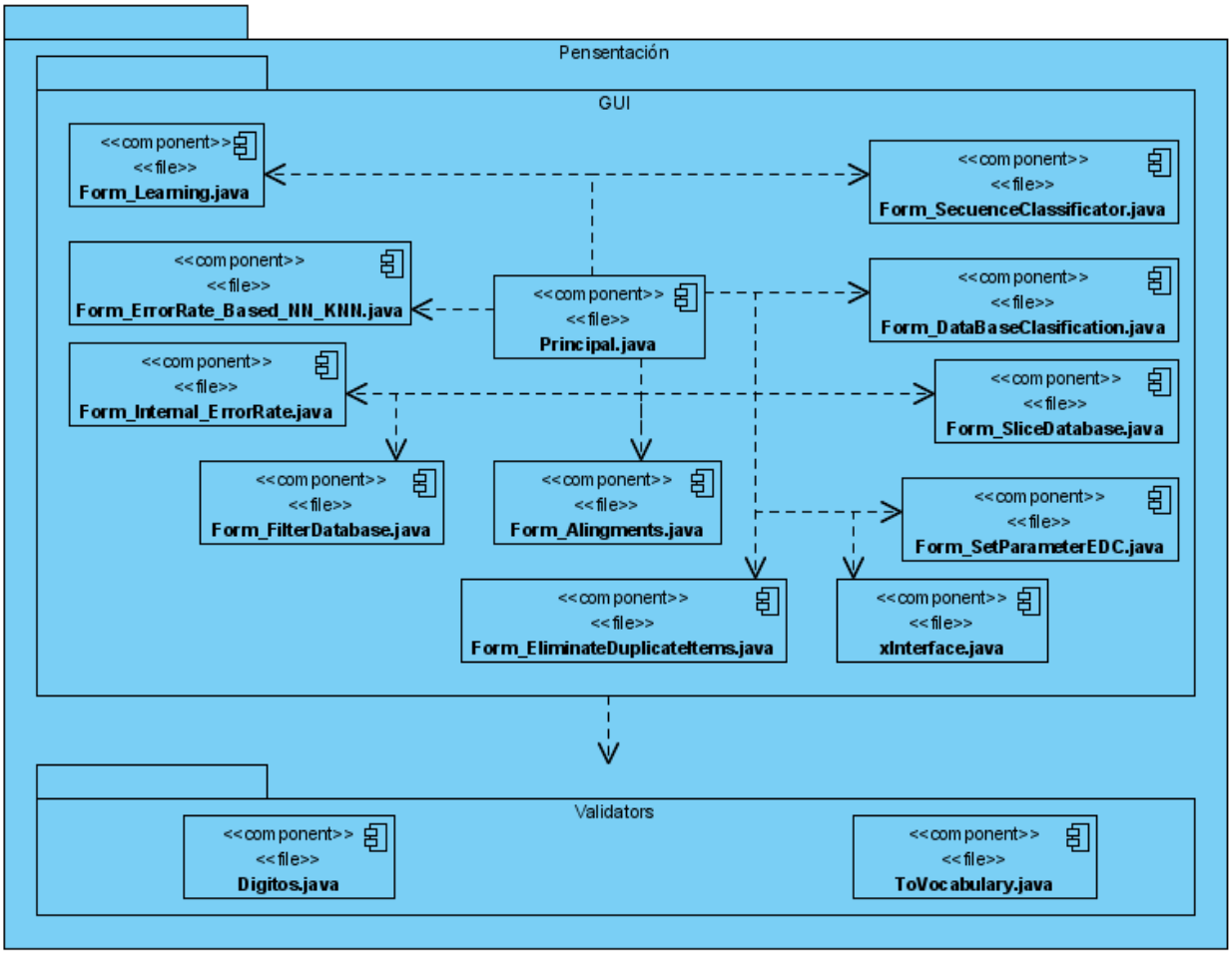


Figura 17: Diagrama de componentes de la capa de presentación.

4.2.1.2 DIAGRAMA DE COMPONENTES DE LA CAPA DE NEGOCIO.

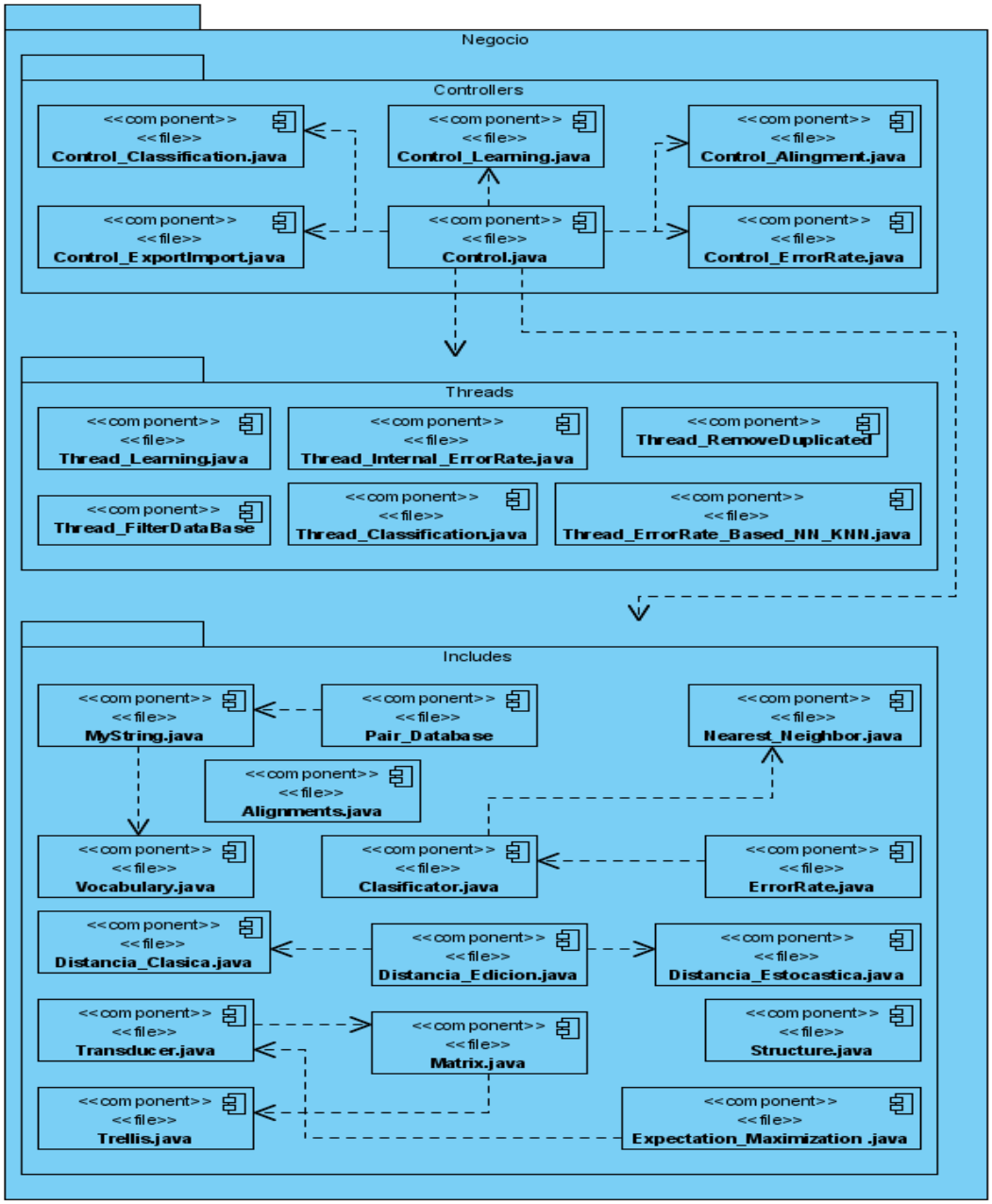


Figura 18: Diagrama de componentes de la capa de negocio.

4.2.1.3 DIAGRAMA DE COMPONENTES DE LA CAPA DE ACCESO A DATOS.

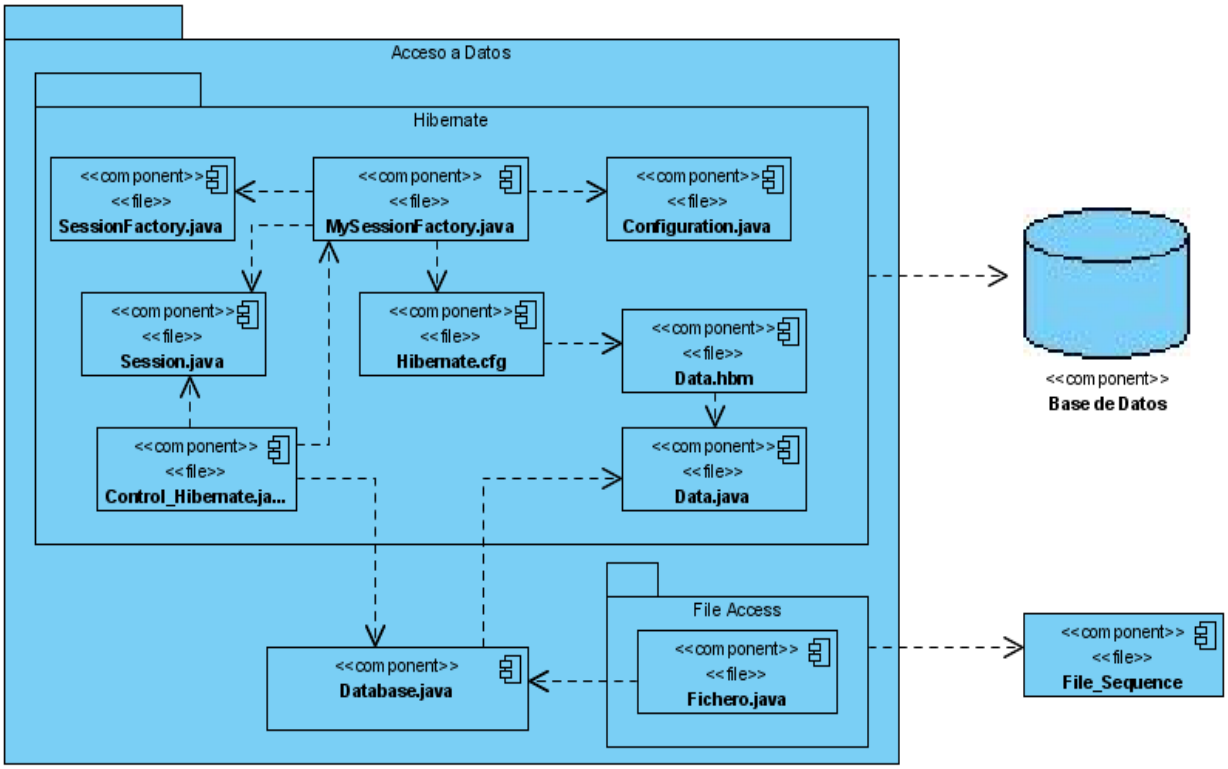


Figura 19: Diagrama de componentes de la capa de acceso a datos.

4.2.2 ALGORITMOS PRINCIPALES.

A continuación se muestra el código fuente de los algoritmos que sustentan la base para la clasificación mediante el aprendizaje de los costos de edición estocásticos a través de transductores sin memoria o transductores de estado finito estocástico.

4.2.2.1 APRENDIZAJE DEL TRANSDUCTOR.

```
public Transducer LearnTransducer(Pair_Database db, String Tipo) {
    double TemporalV = 0; double acc_ant = -Double.MAX_VALUE;
    Transducer t = new Transducer(MyString.Voc_Size());  t.Initialize_sw(0.4);
    if(Tipo == "joint") t.Normalize_Joint();
    if(Tipo == "condicional") t.Normalize_Maximum_Likelihood();
    int num_Iteraciones = 0; double acc = 0.0;
    while(true) {
        acc = 0.0;      num_Iteraciones++;
        Transducer count = new Transducer(MyString.Voc_Size()); count.Nullify();
        ObjtPresent.Set_Progress(0, "Expectation - Maximization");
        ObjtPresent.Set_MaximunValue(db.Size());
        for (int i = 0; i < db.Size(); i++) {
            int x = 0;
        for(int j = 1; j < MyString.Voc_Size(); j++) {
                x += j; }
            Trellis alpha = Trellis.Forward(t, db.Get_in(i), db.Get_out(i));
            Trellis beta = Trellis.Backward(t, db.Get_in(i), db.Get_out(i));
            Expectation(count, t, alpha, beta, db.Get_in(i), db.Get_out(i));
            TemporalV = Math.log(beta.OperatorGet(0, 0) * t.End());
        if(!Double.isInfinite(TemporalV) && !Double.isNaN(TemporalV)) acc += TemporalV;
            ObjtPresent.Set_Progress(i+1, "Expectation - Maximization. " + "Iteracion: " + num_Iteraciones);
        }
        if ((acc_ant - acc) / acc < GetPrecision()) {   break; }
        acc_ant = acc; t = count;
        if(Tipo == "joint")     t.Normalize_Joint();
        if(Tipo == "condicional")   t.Normalize_Maximum_Likelihood();  }
        Iteraciones = num_Iteraciones; Probabilidad = acc;
    return t;
}
```

```
public void Expectation(Transducer Count, Transducer T, Trellis Alpha, Trellis Beta,
    MyString in, MyString out) {
    double aux = 0.0;
    for (int i = 0; i <= in.Size(); i++)
     for (int j = 0; j <= out.Size(); j++) {
       if (i < in.Size()) {
        aux = Count.Del(in.OperatorGet(i));
        aux += Alpha.OperatorGet(i, j) * T.Del(in.OperatorGet(i)) * Beta.OperatorGet(i + 1, j)
        / Beta.OperatorGet(0, 0);
       if(!Double.isNaN(aux) && !Double.isInfinite(aux) && (aux != 0))
        Count.Del(in.OperatorGet(i), aux);}
       if (j < out.Size()) { aux = Count.Ins(out.OperatorGet(j));
        aux += Alpha.OperatorGet(i,j) * T.Ins(out.OperatorGet(j)) * Beta.OperatorGet(i, j + 1)
        / Beta.OperatorGet(0, 0);
       if(!Double.isNaN(aux) && !Double.isInfinite(aux) && (aux != 0))
        Count.Ins(out.OperatorGet(j),  aux); }
       if (i < in.Size() && j < out.Size()) {
        aux = Count.Sub(in.OperatorGet(i), out.OperatorGet(j));
        aux += Alpha.OperatorGet(i, j) * T.Sub(in.OperatorGet(i), out.OperatorGet(j))
        * Beta.OperatorGet(i + 1, j + 1) / Beta.OperatorGet(0, 0);
       if(!Double.isNaN(aux) && !Double.isInfinite(aux) && (aux != 0))
        Count.Sub(in.OperatorGet(i), out.OperatorGet(j), aux);  }
    }
    aux = Count.End()+1;
    if(!Double.isNaN(aux) && !Double.isInfinite(aux) && (aux != 0))
        Count.End(aux);
    }
```

4.2.2.2 DISTANCIA DE EDICIÓN ESTOCÁSTICA.

```
public double Distance(MyString in , MyString out) {
    Trellis alpha = Trellis.Forward(Transd, in, out);
    return -(Math.log(alpha.OperatorGet(in.Size(),out.Size()) * Transd.End()));
}
```

```
public static Trellis Forward(Transducer t, MyString in, MyString out) {
    double Temp = 0;
    Trellis alpha = new Trellis(in.Size()+1, out.Size()+1);
    for(int i = 0; i <= in.Size(); i++)
     for(int j = 0; j <= out.Size(); j++) {
        alpha.OperatorSet(i, j, 0.0);
        if(i == 0 && j == 0)  alpha.OperatorSet(i, j, 1.0);
        if(i > 0) {
        Temp = alpha.OperatorGet(i,j) + alpha.OperatorGet(i-1,j) * t.Del(in.OperatorGet(i - 1));
        if(!Double.isInfinite(Temp) && !Double.isNaN(Temp) && (Temp != 0))
         alpha.OperatorSet(i, j, Temp);
        else
        alpha.OperatorSet(i, j, alpha.OperatorGet(i-1,j)); }
        if(j > 0) {
        Temp = alpha.OperatorGet(i,j) + alpha.OperatorGet(i,j-1) * t.Ins(out.OperatorGet(j - 1));
        if(!Double.isInfinite(Temp) && !Double.isNaN(Temp) && (Temp != 0))
        alpha.OperatorSet(i, j, Temp);
        else alpha.OperatorSet(i, j, alpha.OperatorGet(i,j-1)); }
        if (i > 0 && j > 0) {
        Temp = alpha.OperatorGet(i,j) + alpha.OperatorGet(i-1,j-1)
        * t.Sub(in.OperatorGet(i - 1),out.OperatorGet(j - 1));
        if(!Double.isInfinite(Temp) && !Double.isNaN(Temp) && (Temp != 0))
        alpha.OperatorSet(i, j, Temp);
        else alpha.OperatorSet(i, j, alpha.OperatorGet(i-1,j-1));  }
      }
    return alpha;
}
```

4.2.2.3 DISTANCIA DE EDICIÓN CLÁSICA.

```
public class Distancia_Clasica extends Distancia_Edicion {
    public double Distance(MyString in, MyString out) {
        int CostoDelete = 1; int CostoInsert = 1; int CostoSubstitution = 1;

        int Temp[];
        int Del = 0; int Ins = 0; int Sub = 0;
        int Long1 = in.Size();
        int Long2 = out.Size();
        int[] X = new int[Long2 + 1];
        int[] Y = new int[Long2 + 1];

        X[0] = 0;
        for(int j = 1; j <= Long2; ++j)
            X[j] = X[j-1] + CostoInsert;

        for(int i = 1; i <= Long1; ++i) {
            Y[0] = X[0] + CostoDelete;

            for(int j = 1; j <= Long2; ++j) {
                Del = X[j] + CostoDelete;
                Ins = Y[j-1] + CostoInsert;
                Sub = X[j-1] + (in.OperatorGet(i-1) == out.OperatorGet(j-1) ? 0 : CostoSubstitution);
                Y[j] = Math.min(Math.min(Del, Ins), Sub);
            }

            Temp = X;
            X = Y;
            Y = Temp;
        }

       int Dist = X[Long2];
       return Dist;
    }
}
```

4.3 MODELO DE PRUEBAS.

El flujo de trabajo de pruebas le presta servicios a los demás flujos. Su principal objetivo es evaluar o valorar la calidad del producto a través de la búsqueda y documentación de errores, validando el cumplimiento de requerimientos, el desempeño y dando una indicación de calidad.

La prueba es un proceso de ejecución de un programa con la intención de descubrir errores. Una prueba tiene éxito si descubre un error no detectado hasta entonces.

A la aplicación desarrollada se le aplicaron pruebas de funcionalidad con el objetivo verificar el correcto cumplimiento de los requerimientos funcionales. Se utilizó el método de caja negra, específicamente la técnica de partición de equivalencia. Las pruebas de caja negra son las que se llevan a cabo sobre la interfaz del software. El objetivo es demostrar que las funciones del software son operativas, que las entradas se aceptan de forma adecuada y se produce un resultado correcto, y que la integridad de la información externa se mantiene (no se ve el código). A continuación se muestran los casos de prueba.

4.3.1 CASOS DE PRUEBA.

Caso de uso	Realizar Aprendizaje
Caso de prueba	Realizar Aprendizaje. (BDA desde archivo).
Entrada	El especialista selecciona en el panel BD la opción archivo e importa la BD desde archivo, introduce la cantidad de vecinos para la creación de la BD de pares de cadenas, introduce la precisión del transductor que se va a obtener y selecciona la distribución en el panel tipo de distribución.
Resultado esperado	Obtener un Transductor con los costos de las operaciones de edición aprendidos.

Resultado de la prueba	El sistema muestra un cuadro de diálogo brindando la posibilidad de mostrar el transductor aprendido en la pantalla principal de la aplicación, además muestra un cuadro de diálogo que brinda la opción de salvar en disco el transductor correspondiente al aprendizaje.
Condiciones	El formato de la BD cargada desde archivo debe ser (id, label, secuencia), separada por espacio. El número de vecinos debe ser menor o igual que la menor cantidad de microorganismos con un label determinado en la BD, y la precisión debe estar en un rango de (0 a 1). Y se debe seleccionar la distribución en el panel tipo de distribución.

Caso de uso	Realizar Aprendizaje
Caso de prueba	Realizar Aprendizaje (BDA a través de conexión).
Entrada	El especialista selecciona en el panel BD la opción Base de datos, luego introduce la dirección IP y el puerto correspondiente al servidor de BD, el usuario, la contraseña, en nombre de la BD y la tabla de la BD con la que se estableció la conexión. Introduce la cantidad de vecinos para la creación de la BD de pares de cadenas, introduce la precisión del transductor que se va a obtener y selecciona el tipo de distribución para el mismo.
Resultado esperado	Obtener un Transductor con los costos de las operaciones de edición aprendidos.
Resultado de la	El sistema muestra un cuadro de diálogo brindando la

prueba	posibilidad de mostrar el transductor aprendido en la pantalla principal de la aplicación, además muestra un cuadro de diálogo que brinda la opción de salvar en disco el transductor correspondiente al aprendizaje.
Condiciones	La dirección IP y puerto del servidor de BD, el usuario, el nombre de la BD, la contraseña y el nombre de la tabla debe ser el correcto y con formato: (id, label, secuencia). El número de vecinos debe ser menor o igual que la menor cantidad de microorganismos con un label determinado en la BD y la precisión debe estar en un rango de (0 a 1). Se debe seleccionar un tipo de distribución para el transductor.

Caso de uso	Clasificar Secuencia
Caso de prueba	Clasificar Secuencia (BD desde archivo).
Entrada	El especialista selecciona en el panel BD de aprendizaje la opción archivo. Selecciona el tipo de distancia de edición en el panel Distancia de Edición, introduce el número de vecinos y selecciona el tipo de alineamiento (global, local o ambos) en el panel Alineamientos opcionalmente. Finalmente introduce la secuencia que desea clasificar.
Resultado esperado	Obtener el resultado de la clasificación en pantalla.
Resultado de la prueba	El sistema muestra los resultados de la clasificación de la secuencia en la pantalla principal.
Condiciones	El formato de la BD cargada desde archivo debe ser (id, label, secuencia), separada por espacio.

	El número de vecinos debe ser menor o igual que la menor cantidad de microorganismos con un label determinado en la BD, se debe seleccionar la distancia de edición en el panel Distancia de Edición, puede o no seleccionar algún tipo de alineamiento y la secuencia introducida para la clasificación debe tener el vocabulario correspondiente a la BD.

Caso de uso	Clasificar Secuencia
Caso de prueba	Clasificar Secuencia (BD a través de conexión).
Entrada	El especialista selecciona en el panel BD de aprendizaje la opción Base de Datos. Luego introduce la dirección IP y el puerto correspondiente al servidor de BD, el usuario, la contraseña, el nombre de la BD y la tabla de esa BD con la que se estableció la conexión. Introduce la cantidad de vecinos, selecciona el tipo de distancia de edición en el panel Distancia de Edición, introduce el número de vecinos y selecciona el tipo de alineamiento (global, local o ambos) en el panel Alineamientos opcionalmente. Finalmente introduce la secuencia que desea clasificar.
Resultado esperado	Obtener el resultado de la clasificación en pantalla.
Resultado de la prueba	El sistema muestra los resultados de la clasificación de la secuencia en la pantalla principal.
Condiciones	La dirección IP y puerto del servidor de BD, el usuario, el nombre de la BD, la contraseña y el nombre de la tabla debe ser el correcto. La tabla con el nombre especificado debe

	estar en la BD con la que se realizó la conexión, y tener el formato: (id, label, secuencia). El número de vecinos debe ser menor o igual que la menor cantidad de microorganismos con un label determinado en la BD, se debe seleccionar la distancia de edición en el panel Distancia de Edición, puede o no seleccionar algún tipo de alineamiento y la cadena introducida para la clasificación debe tener el vocabulario correspondiente a la BD.

Caso de uso	Clasificar BDS
Caso de prueba	Clasificar BDS (BDA desde archivo BDC desde archivo).
Entrada	El especialista selecciona en el panel BD de aprendizaje la opción archivo, selecciona en el panel BD a Clasificar la opción archivo. Selecciona el tipo de distancia de edición en el panel Distancia de Edición e introduce el número de vecinos.
Resultado esperado	Obtener un cuadro de diálogo brindando la opción de salvar el resultado de la clasificación en el disco.
Resultado de la prueba	El sistema muestra un cuadro de diálogo brindando la opción de salvar el resultado de la clasificación en el disco.
Condiciones	El formato de la BD de aprendizaje cargada desde archivo debe ser (id, label, secuencia), separada por espacio. El formato de la BD a clasificar cargada dese archivo de ser (id, secuencia). El número de vecinos debe ser menor o igual que la menor cantidad de microorganismos con un label determinado en la BD Y se debe seleccionar la distancia de edición en el panel Distancia de Edición.

Caso de uso	Clasificar BDS
Caso de prueba	Clasificar BDS (BDA desde archivo BDC a través de conexión).
Entrada	El especialista selecciona en el panel BD de aprendizaje la opción archivo, selecciona en el panel BD a Clasificar la opción Base de Datos. Luego introduce la dirección IP y el puerto correspondiente al servidor de BD, el usuario, la contraseña, el nombre de la BD y la tabla de esa BD con la que se estableció la conexión. Introduce la cantidad de vecinos, selecciona el tipo de distancia de edición en el panel Distancia de Edición e introduce el número de vecinos.
Resultado esperado	Obtener un cuadro de diálogo brindando la opción de salvar el resultado de la clasificación en el disco.
Resultado de la prueba	El sistema muestra un cuadro de diálogo brindando la opción de salvar el resultado de la clasificación en el disco.
Condiciones	El formato de la BD de aprendizaje cargada desde archivo debe ser (id, label, secuencia), separada por espacio. El formato de la tabla de la BD a clasificar cargada dese archivo de ser (id, label, secuencia). El número de vecinos debe ser menor o igual que la menor cantidad de microorganismos con un label determinado en la BD. Se debe seleccionar la distancia de edición en el panel Distancia de Edición.

Caso de uso	Clasificar BDS
Caso de prueba	Clasificar BDS (BDA a través de conexión y BDC desde archivo).

Entrada	El especialista selecciona en el panel BD de aprendizaje la opción Base de Datos, selecciona en el panel BD a Clasificar la opción Archivo e importa la BD. Luego introduce la dirección IP y el puerto correspondiente al servidor de BD, el usuario, la contraseña, el nombre de la BD y la tabla de esa BD con la que se estableció la conexión. Introduce la cantidad de vecinos, selecciona el tipo de distancia de edición en el panel Distancia de Edición e introduce el número de vecinos.
Resultado esperado	Obtener un cuadro de diálogo brindando la opción de salvar el resultado de la clasificación en el disco.
Resultado de la prueba	El sistema muestra un cuadro de diálogo brindando la opción de salvar el resultado de la clasificación en el disco.
Condiciones	El formato de la tabla de la BD de aprendizaje debe ser (id, label, secuencia). El formato de la BD a clasificar cargada desde archivo debe ser (id, label, secuencia), separada por espacio. El número de vecinos debe ser menor o igual que la menor cantidad de microorganismos con un label determinado en la BD. Se debe seleccionar la distancia de edición en el panel Distancia de Edición.

Caso de uso	Clasificar BDS
Caso de prueba	Clasificar BDS (BDA a través de conexión y BDC a través de conexión).
Entrada	El especialista selecciona en el panel BD de aprendizaje la opción Base de Datos, selecciona en el panel BD a Clasificar

	la opción Base de Datos. Luego introduce la dirección IP y el puerto correspondiente al servidor de BD, el usuario, la contraseña, el nombre de la BD y la tabla de esa BD con la que se estableció la conexión para ambas BD(Aprendizaje y BD a Clasificar). Introduce la cantidad de vecinos, selecciona el tipo de distancia de edición en el panel Distancia de Edición e introduce el número de vecinos.
Resultado esperado	Obtener un cuadro de diálogo brindando la opción de salvar el resultado de la clasificación en el disco.
Resultado de la prueba	El sistema muestra un cuadro de diálogo brindando la opción de salvar el resultado de la clasificación en el disco.
Condiciones	El formato de la BD de aprendizaje debe ser (id, label, secuencia). El formato de la BD a clasificar (id, label, secuencia). El número de vecinos debe ser menor o igual que la menor cantidad de microorganismos con un label determinado en la BD. Se debe seleccionar la distancia de edición en el panel Distancia de Edición.

Caso de uso	Determinar TEI.
Caso de prueba	Determinar TEI. (Importando BD desde archivo).
Entrada	El especialista selecciona en el panel Base de Datos la opción archivo e importa la BD desde archivo y Selecciona el tipo de distancia de edición en el panel Distancia de Edición.
Resultado esperado	Obtener los resultados de la Tasa de Error Interna en la pantalla principal y obtener además un cuadro de diálogo brindando la posibilidad de salvar la información referente a

	la TEI en el disco.
Resultado de la prueba	El sistema muestra los resultados de la Tasa de Error Interna en la pantalla principal y muestra además un cuadro de diálogo brindando la posibilidad de salvar la información referente a la TEI en el disco.
Condiciones	El formato de la BD cargada desde archivo debe ser (id, label, secuencia), separada por espacio. Se debe seleccionar el tipo de distancia de edición en el panel Distancia de Edición.

Caso de uso	Determinar TEI.
Caso de prueba	Determinar TEI. (Conexión a BD).
Entrada	El especialista selecciona en el panel Base de Datos la opción Base de Datos, introduce la dirección IP del servidor de BD y el puerto, introduce el usuario, la contraseña, el nombre de la BD y el nombre de la tabla a la que desea conectarse. Selecciona el tipo de distancia de edición en el panel Distancia de Edición.
Resultado esperado	Obtener los resultados de la Tasa de Error Interna en la pantalla principal y obtener además un cuadro de diálogo brindando la posibilidad de salvar la información referente a la TEI en el disco.
Resultado de la prueba	El sistema muestra los resultados de la Tasa de Error Interna en la pantalla principal y muestra además un cuadro de diálogo brindando la posibilidad de salvar la información referente a la TEI en el disco.
Condiciones	El formato de la BD debe ser (id, label, secuencia). Se debe

	seleccionar la distancia de edición en el panel Distancia de Edición.

Caso de uso	Determinar TE por KNN.
Caso de prueba	Determinar TE por NN. (BDA desde archivo y BDP desde archivo).
Entrada	El especialista selecciona en el panel Base de Datos de Entrenamiento la opción archivo e importa la BD desde archivo, Selecciona en el panel BD de Prueba la opción archivo e importa la BD desde archivo, Selecciona el tipo de distancia de edición en el panel Distancia de Edición e introduce la cantidad de vecinos.
Resultado esperado	Obtener un cuadro de diálogo brindando la posibilidad de salvar la información referente a la TE por KNN en el disco.
Resultado de la prueba	El sistema muestra un cuadro de diálogo brindando la posibilidad de salvar la información referente a la TE por NN en el disco.
Condiciones	El formato de las base de datos cargadas desde archivo debe ser (id, label, secuencia), separado por espacio. Se debe seleccionar un tipo de distancia de edición y el número de vecinos debe ser menor o igual que la menor cantidad de microorganismos con un label determinado en la BD.

Caso de uso	Determinar TE por KNN.
Caso de prueba	Determinar TE por NN. (BDA desde archivo y BDP a través de conexión).
Entrada	El especialista selecciona en el panel Base de Datos de

	Entrenamiento la opción archivo e importa la BD desde archivo, Selecciona en el panel BD de Prueba la opción Base de Datos e introduce los datos para la conexión, Selecciona el tipo de distancia de edición en el panel Distancia de Edición e introduce la cantidad de vecinos.
Resultado esperado	Obtener un cuadro de diálogo brindando la posibilidad de salvar la información referente a la TE por KNN en el disco.
Resultado de la prueba	El sistema muestra un cuadro de diálogo brindando la posibilidad de salvar la información referente a la TE por KNN en el disco.
Condiciones	El formato de la base de datos cargada desde archivo debe ser (id, label, secuencia), separado por espacio. Se debe seleccionar un tipo de distancia de edición y el número de vecinos debe ser menor o igual que la menor cantidad de microorganismos con un label determinado en la BD. El formato de la tabla de la BD debe ser (id, label, secuencia).

Caso de uso	Determinar TE por KNN.
Caso de prueba	Determinar TE por NN. (BDA a través de conexión y BDP desde archivo).
Entrada	El especialista selecciona en el panel Base de Datos de Entrenamiento la opción Base de Datos e introduce los datos, Selecciona en el panel BD de Prueba la opción archivo e importa la BD desde archivo, Selecciona el tipo de distancia de edición en el panel Distancia de Edición e introduce la cantidad de vecinos.
Resultado esperado	Obtener un cuadro de diálogo brindando la posibilidad de

	salvar la información referente a la TE por KNN en el disco.
Resultado de la prueba	El sistema muestra un cuadro de diálogo brindando la posibilidad de salvar la información referente a la TE por KNN en el disco.
Condiciones	El formato de las base de datos cargada desde archivo debe ser (id, label, secuencia), separado por espacio. Se debe seleccionar un tipo de distancia de edición y el número de vecinos debe ser menor o igual que la menor cantidad de microorganismos con un label determinado en la BD. El formato de la tabla de la BD debe ser (id, label, secuencia).

Caso de uso	Determinar TE por KNN.
Caso de prueba	Determinar TE por NN. (BDA a través de conexión y BDP a través de conexión).
Entrada	El especialista selecciona en el panel Base de Datos de Entrenamiento la opción Base de Datos, Selecciona en el panel BD de Prueba la opción Base de Datos, introduce la dirección IP y el puerto del servidor de BD, el usuario, la contraseña, el nombre de la BD, el nombre de la tabla a la que se va a conectar, de ambas BD. Selecciona el tipo de distancia de edición en el panel Distancia de Edición e introduce la cantidad de vecinos.
Resultado esperado	Obtener un cuadro de diálogo brindando la posibilidad de salvar la información referente a la TE por KNN en el disco.
Resultado de la prueba	El sistema muestra un cuadro de diálogo brindando la posibilidad de salvar la información referente a la TE por KNN en el disco.

Condiciones	La tabla de cada una de las BD con la que se establece la conexión debe tener el formato (id, label, secuencia). Se debe seleccionar un tipo de distancia de edición y el número de vecinos debe ser menor o igual que la menor cantidad de microorganismos con un label determinado en la BD.

4.4 CONCLUSIONES.

En el presente capítulo se definió el modelo de implementación, especificando el diagrama de componentes para la aplicación. Se muestran además los algoritmos principales y los casos de prueba para los casos de uso críticos de la aplicación, con el objetivo de comprobar el correcto funcionamiento de las funcionalidades o requerimientos que se proponen en la herramienta.

CONCLUSIONES

La aplicación se desarrolló siguiendo la metodología OpenUP para el proceso de desarrollo de software, permitiendo la realización de los flujos de trabajos que la misma propone en sus respectivas fases, alcanzándose los siguientes resultados:

1. Se obtuvo una representación de los principales conceptos del entorno del problema a través del modelo de dominio.
2. Se obtuvieron todos los requerimientos funcionales para la herramienta propuesta.
3. Se realizó el diseño de la aplicación siguiendo la arquitectura en capas.
4. Se implementó la herramienta.
5. Se realizaron pruebas de funcionalidad que demostraron el correcto cumplimiento de los requerimientos funcionales.

RECOMENDACIONES

- ✓ Realizar un análisis comparativo de la de la clasificación taxonómica por homologías de secuencias, teniendo en cuenta el funcionamiento de otras herramientas existentes.
- ✓ Añadir la solución al portal del Instituto de Ciencia Animal (ICA).
- ✓ Distribuir el proceso de aprendizaje.

REFERENCIAS

[1]. *Learning string edit distance.* **Ristad, Eric Sven y Yianilos, Peter N.** 5, s.l. : IEEE Transactions, 1998, Vol. 20, págs. 522-532.

[3]. Abascal, Federico. Análisis de secuencias. [En línea] [Citado: Octubre 9, 2008.] http://darwin.uvigo.es/people/fabascal/Teaching/index.html.

[4]. Brudno, M, et al. Global Alignment: finding rearrangements during aliment . [En línea] 2003. [Citado: Octubre 12, 2008.] http://bioinformatics.oxfordjournals.org/cgi/pmidlookup?view=long&pmid=12855437.

[5]. Brudno, M, et al. Automate whole-genome multiple alignment of rat, mouse, and human. [En línea] 2004. [Citado: Octubre 12, 2008.] http://www.genome.org/cgi/reprint/14/4/685.pdf.

[6]. Universidad Nacional de Colombia. *Curso de Biología Computacional.* [En línea] [Citado: Octubre 15, 2008.] http://www.virtual.unal.edu.co/cursos/ingenieria/2001832/index.html.

[7]. González, Germán. *Algoritmos NEEDLEMAN-WUNSCH y SMITH-WATERMAN*. [En línea] [Citado: Octubre 18, 2008.] http://ww.bioinformaticos.com.ar/articulos.

[8]. Eclipse Process Framework. *Introduction to Open UP.* [En línea] [Citado: Noviembre 10, 2008.] http://epf.eclipse.org/wikis/openup/index.htm.

[9]. Henikoff, Steven y Henikoff, Jorja. Amino acid substitution matrices from protein blocks. [En línea] [Citado: Diciembre 12, 2008.] http://www.pnas.org/content/89/22/10915.abstract.

[10]. Jacobson, Ivar, Booch, Grady and Rumbaugh, James. *El Proceso Unificado de Desarrollo de Software. .* s.l. : PEARSON EDUCACIÓN S.A, 2000.

[11]. Janodet, Jean-Christophe, Bernard, Marc y Sebban, Marc. A Discriminative Model of Stochastic Edit Distance in the form of Conditional Transducer. [En línea] [Citado: Octubre 10, 2008.] http://labh-curien.univ-st-etienne.fr/~janodet/.

[12]. *Learning stochastic edit distance: Application in handwritten character recognition.* **Oncina, José y Sebban, Marc.** 39, 2006, Pattern Recognition, págs. 1575 - 1587.

[14]. Bilenko, Mikhail y Mooney, Raymond. Adaptive Duplicate Detection Using Learnable String Similarity Measures. [En línea] [Citado: Diciembre 15, 2008.] http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.14.423.

[15]. Sum Microsystems. *MySQL, Manual de Referencia.* [En línea] [Citado: Noviembre 12, 2008.] http://dev.mysql.com/doc/refman/5.0/es/index.html.

[16]. Ristad, Eric Sven y Yianilos, Peter. Finite growth models. *Princeton University Computer Science Department.* [En línea] [Citado: Diciembre 20, 2008.] http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.34.1736.

[17]. Mount, David. Bioinformatics: Sequence and Genome Analysis. [En línea] [Citado: Diciembre 10, 2008.] http://www.clinchem.org/cgi/content/extract/51/11/2219.

[19]. Calero, Manuel. Una explicación de la programación extrema (XP). [En Linea]. [Citado: Octubre 18, 2008.] http://www.willydev.net/InsiteCreation/v1.0/descargas/prev/explicaxp.pdf.

[20]. Ibarra, Armando. Rational Unified Process. [En línea] [Citado: Octubre 18, 2008.] https://www.u-cursos.cl/ingenieria/2004/2/CC62C/1/material_docente/objeto/44990.

[21]. Fernández, Gerardo. Introducción a Extreme Programming. [En línea]. [Citado: Octubre 20, 2008.] http://www.info-ab.uclm.es/asignaturas/42551/trabajosAnteriores/Presentacion-XP.pdf.

[22]. Itera, empresa centrada en la mejora de procesos en las áreas de TI y negocio. *Rational Unified Process.* [En línea] [Citado: Noviembre 6, 2008.]

http://www.iteraprocess.com/index.php?option=com_content&task=view&id=18&Itemid=42.

BIBLIOGRAFÍA

Learning string edit distance. **Ristad, Eric Sven y Yianilos, Peter N.** 5, s.l. : IEEE Transactions, 1998, Vol. 20, págs. 522-532.

Abascal, Federico. Análisis de secuencias. [En línea] [Citado: Octubre 9, 2008.] http://darwin.uvigo.es/people/fabascal/Teaching/index.html.

Brudno, M, et al. Global Alignment: finding rearrangements during aliment . [En línea]. [Citado: Octubre 12, 2008.] http://bioinformatics.oxfordjournals.org/cgi/pmidlookup?view=long&pmid=12855437.

Brudno, M, et al. Automate whole-genome multiple alignment of rat, mouse, and human. [En línea]. [Citado: Octubre 12, 2008.] http://www.genome.org/cgi/reprint/14/4/685.pdf.

Universidad Nacional de Colombia. *Curso de Biología Computacional.* [En línea] [Citado: Octubre 15, 2008.] http://www.virtual.unal.edu.co/cursos/ingenieria/2001832/index.html.

González, Germán. *Algoritmos NEEDLEMAN-WUNSCH y SMITH-WATERMAN*. [En línea] [Citado: Octubre 18, 2008.] http://ww.bioinformaticos.com.ar/articulos.

Eclipse Process Framework. *Introduction to Open UP.* [En línea] [Citado: Noviembre 10, 2008.] http://epf.eclipse.org/wikis/openup/index.htm.

Henikoff, Steven y Henikoff, Jorja. Amino acid substitution matrices from protein blocks. [En línea] [Citado: Diciembre 12, 2008.] http://www.pnas.org/content/89/22/10915.abstract.

Jacobson, Ivar, Booch, Grady and Rumbaugh, James. *El Proceso Unificado de Desarrollo de Software. .* s.l. : PEARSON EDUCACIÓN S.A, 2000.

Janodet, Jean-Christophe, Bernard, Marc y Sebban, Marc. A Discriminative Model of Stochastic Edit Distance in the form of Conditional Transducer. [En línea] [Citado: Octubre 10, 2008.] http://labh-curien.univ-st-etienne.fr/~janodet/.

Learning stochastic edit distance: Application in handwritten character recognition. **Oncina, José y Sebban, Marc.** 39, 2006, Pattern Recognition, págs. 1575 - 1587.

Bilenko, Mikhail y Mooney, Raymond. Adaptive Duplicate Detection Using Learnable String Similarity Measures. [En línea] [Citado: Diciembre 15, 2008.] http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.14.423.

Sum Microsystems. *MySQL, Manual de Referencia.* [En línea] [Citado: Noviembre 12, 2008.] http://dev.mysql.com/doc/refman/5.0/es/index.html.

Ristad, Eric Sven y Yianilos, Peter. Finite growth models. *Princeton University Computer Science Department.* [En línea] [Citado: Diciembre 20, 2008.] http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.34.1736.

Mount, David. Bioinformatics: Sequence and Genome Analysis. [En línea] [Citado: Diciembre 10, 2008.] http://www.clinchem.org/cgi/content/extract/51/11/2219.

Calero, Manuel. Una explicación de la programación extrema (XP). [En línea]. [Citado: Octubre 18, 2008.] http://www.willydev.net/InsiteCreation/v1.0/descargas/prev/explicaxp.pdf.

Ibarra, Armando. Rational Unified Process. [En línea] [Citado: Octubre 18, 2008.] https://www.u-cursos.cl/ingenieria/2004/2/CC62C/1/material_docente/objeto/44990.

Fernández, Gerardo. Introducción a Extreme Programming. [En línea]. [Citado: Octubre 20, 2008.] http://www.info-ab.uclm.es/asignaturas/42551/trabajosAnteriores/Presentacion-XP.pdf.

Itera, empresa centrada en la mejora de procesos en las áreas de TI y negocio. *Rational Unified Process.* [En línea] [Citado: Noviembre 6, 2008.]

http://www.iteraprocess.com/index.php?option=com_content&task=view&id=18&Itemid=42.

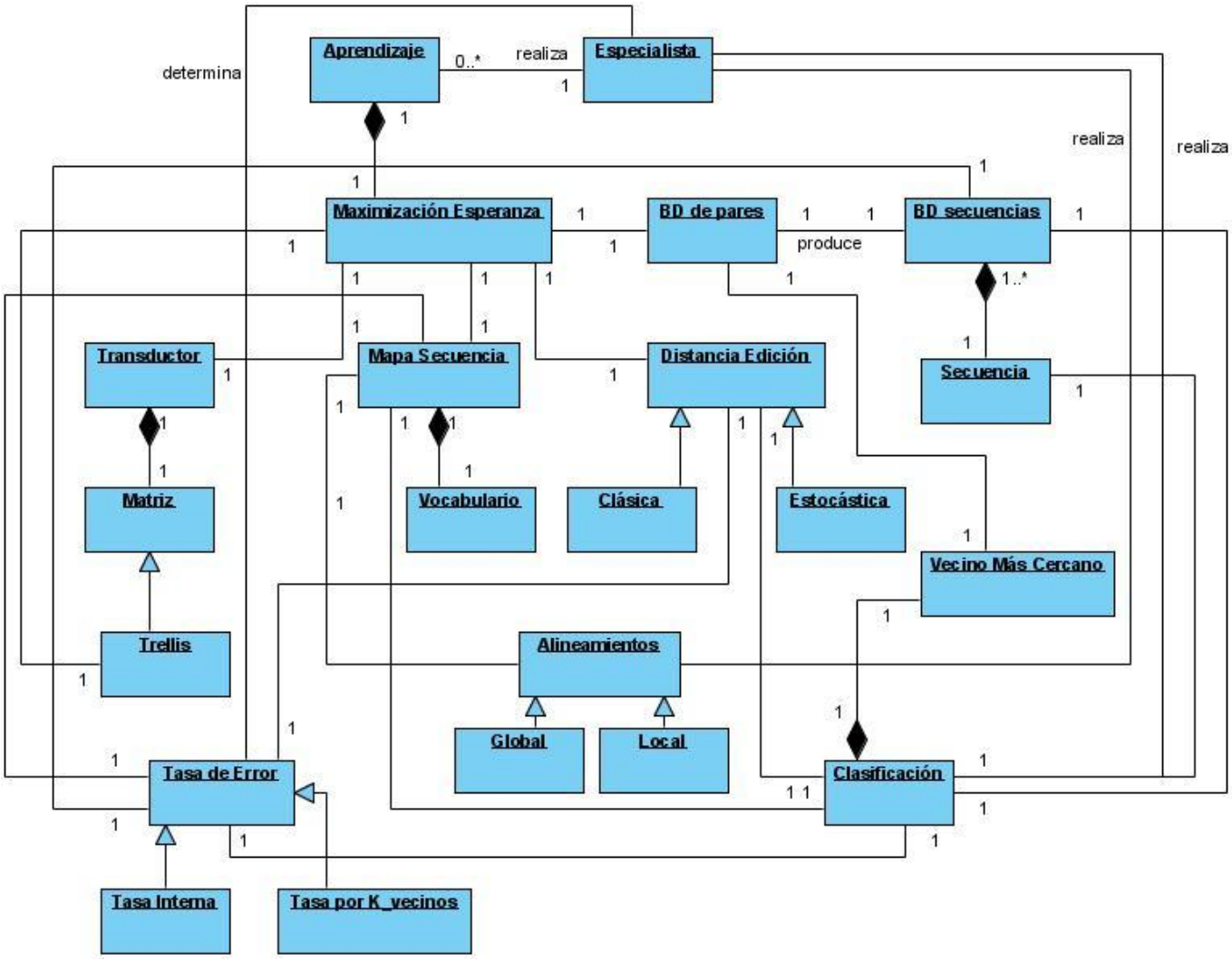
Lago, Ramiro. *Patrones de diseño software*. [En línea]. [Citado: Febrero 20, 2009]. http://www.proactiva-calidad.com/java/patrones/index.html#algunos_patrones.

Guerrero, Luis. *Análisis y Diseño Orientado a Objetos. Patrones de diseño.* [Citado: Febrero 21, 2009]. http://www.dcc.uchile.cl/~luguerre/cc40b/clase11.html

Larman, Graig. *Uml y patrones, introducción al análisis y diseño orientado a objetos.* México : Prentice Hall, 1999. 970-17-0261-1.

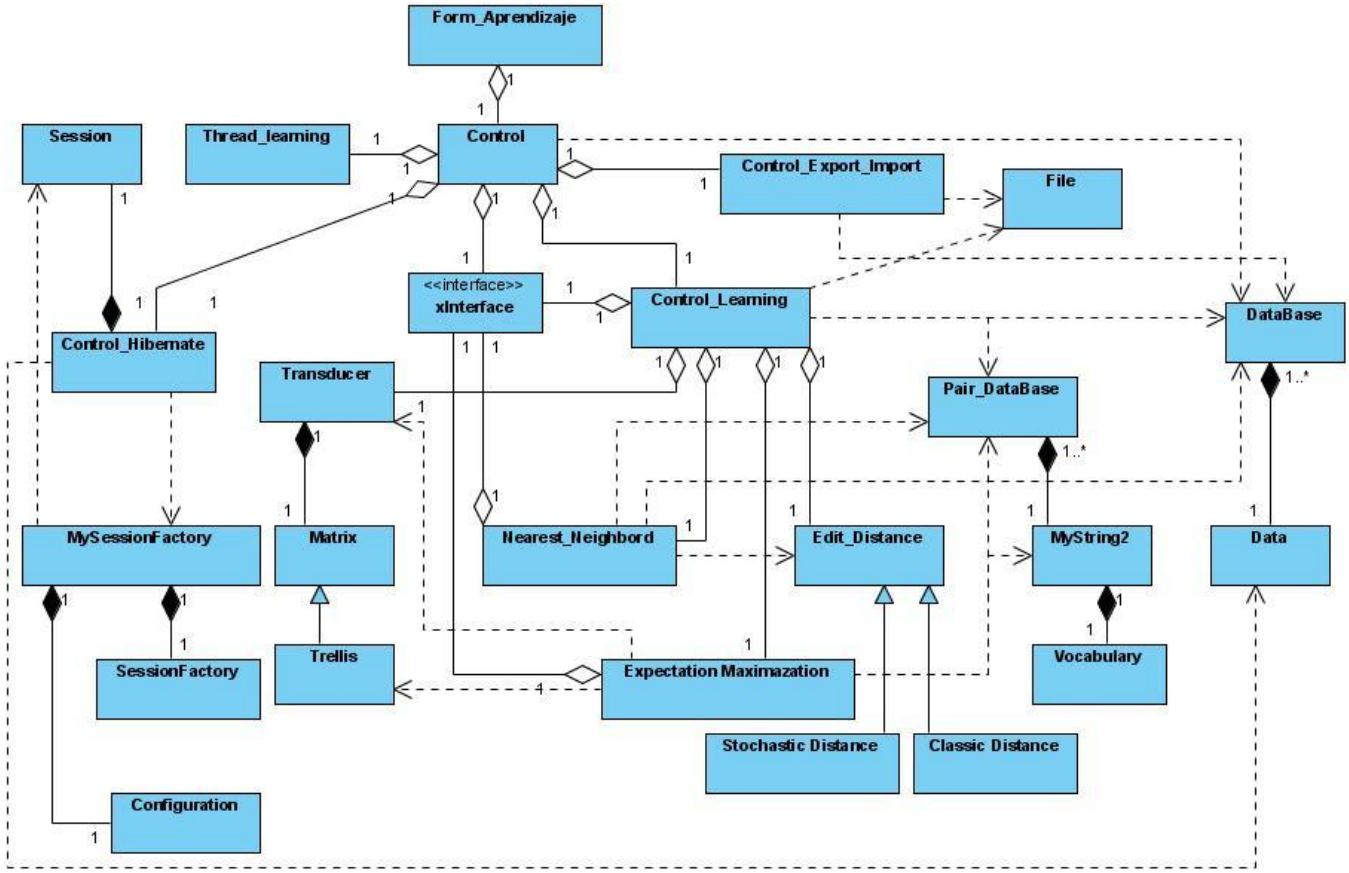
ANEXOS

1. **ANEXO 1:** MODELO DE DOMINIO.

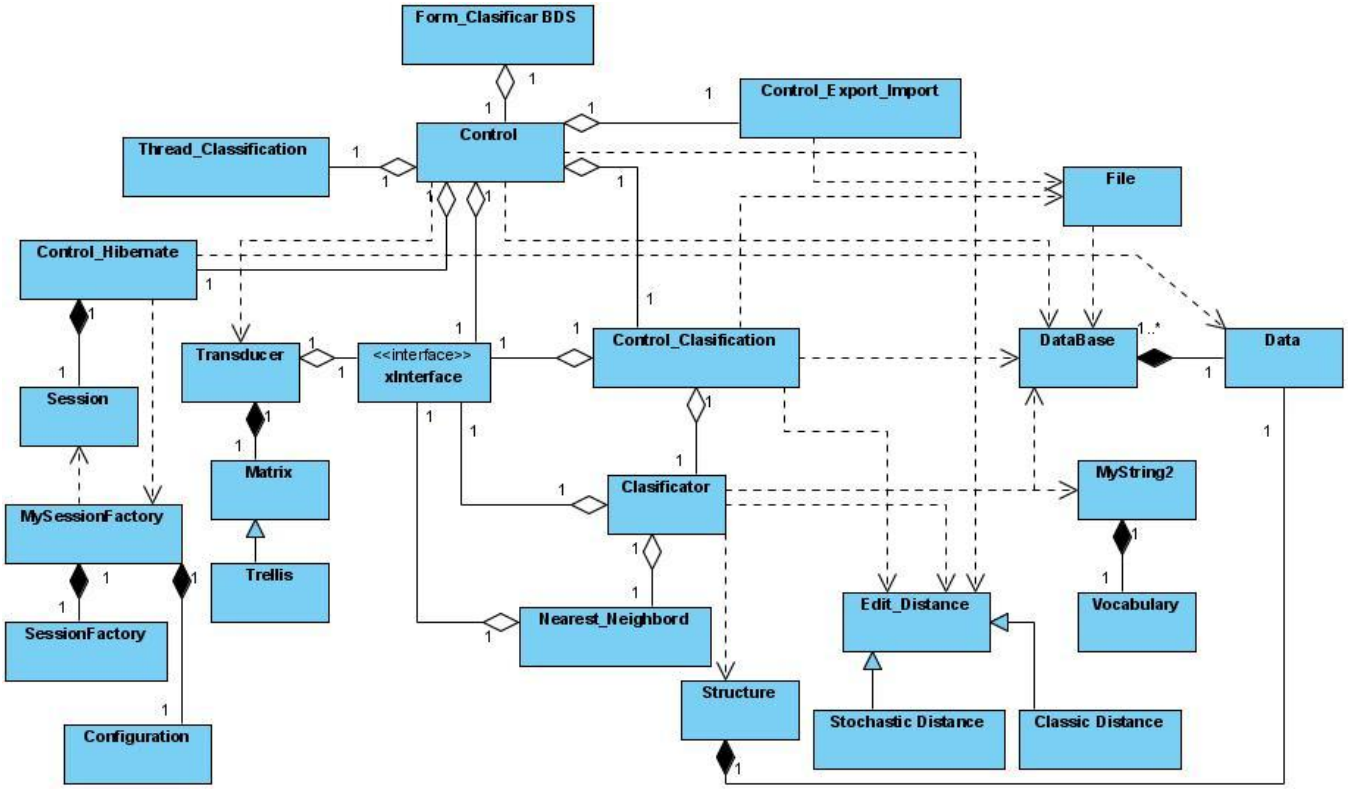


2. **ANEXO 2:** DIAGRAMAS DE CLASES POR CU.

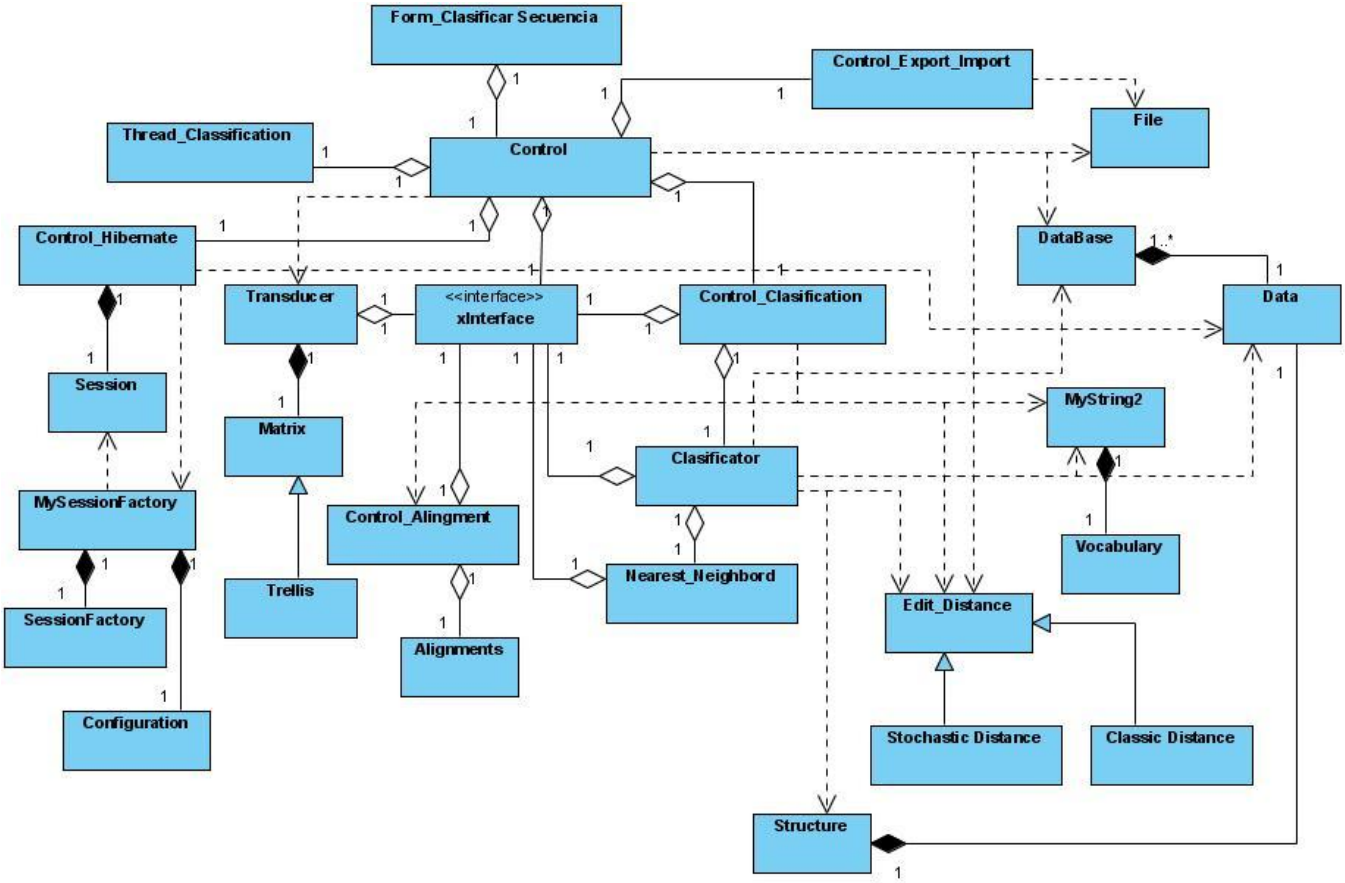
CU: **REALIZAR APRENDIZAJE.**



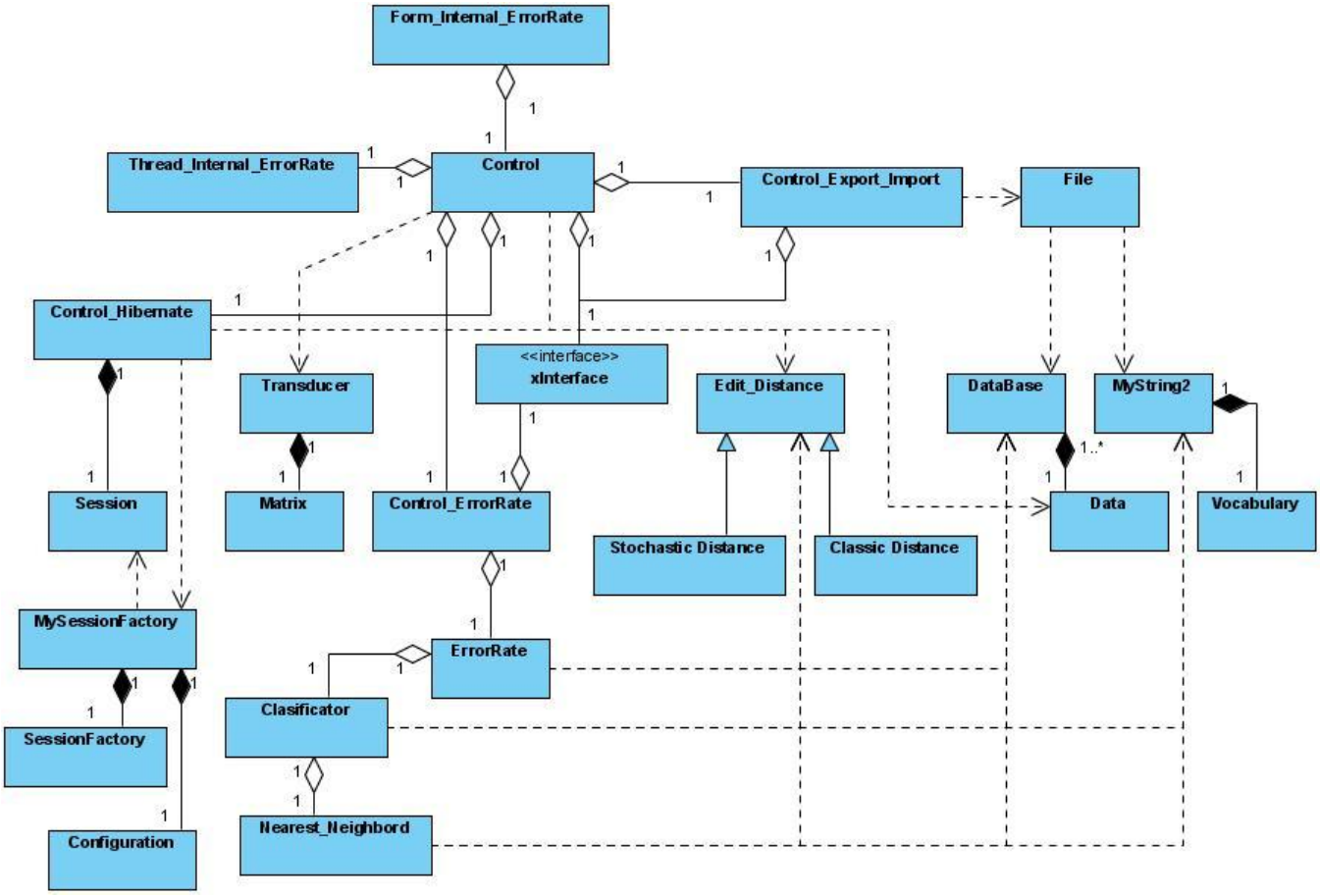
CU: CLASIFICAR BDS.



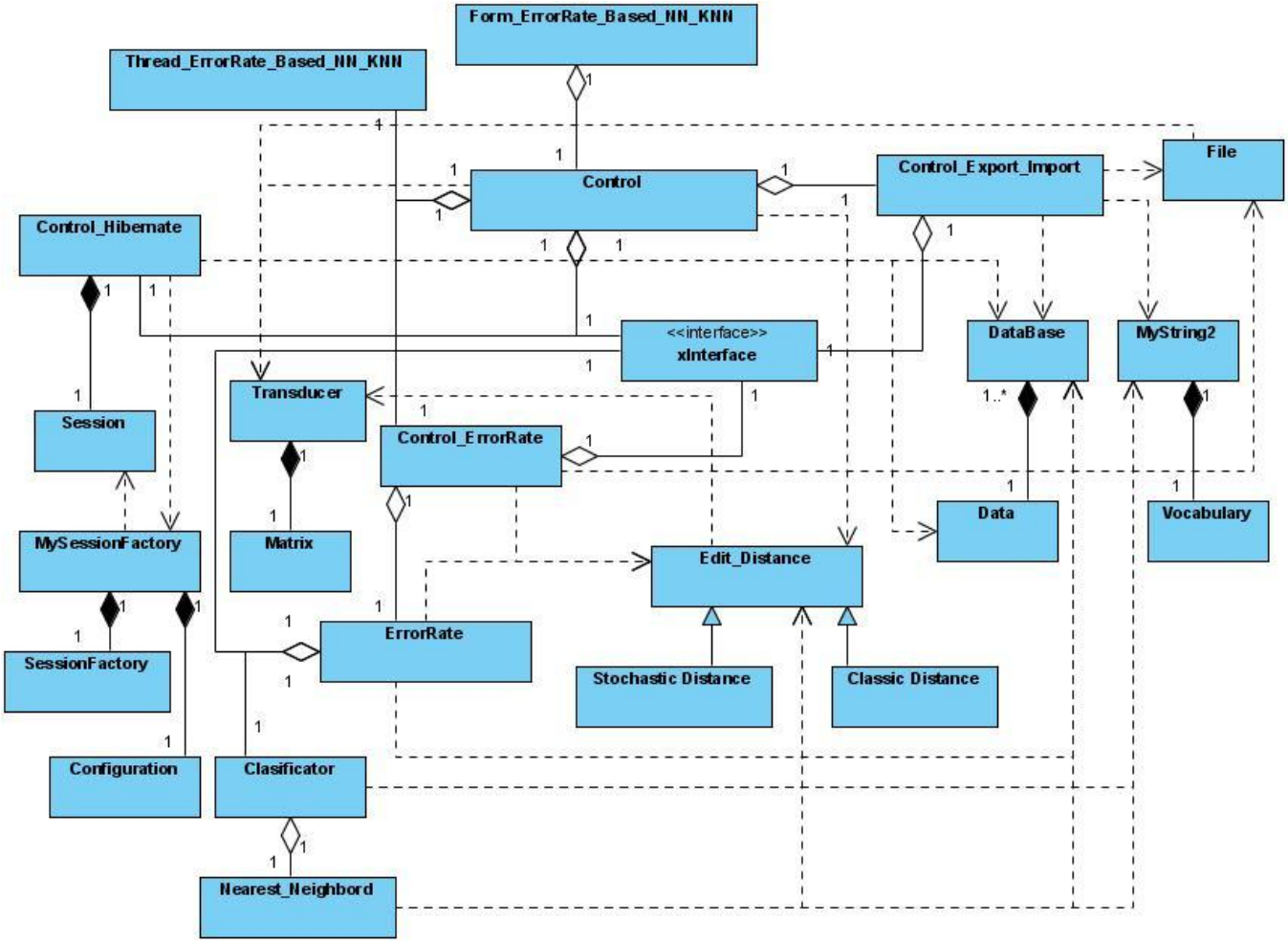
CU: CLASIFICAR SECUENCIA.



CU: DETERMINAR TEI.

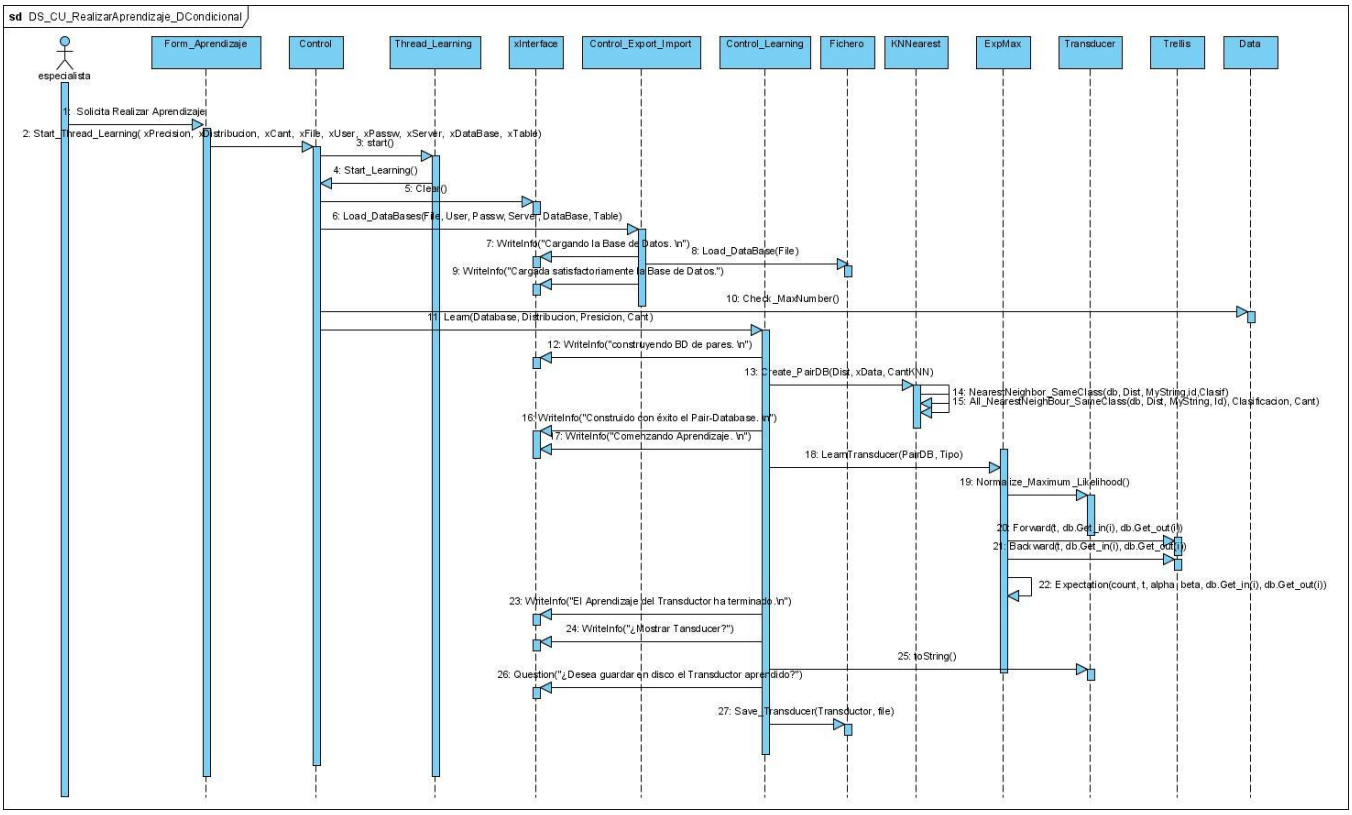


CU: DETERMINAR TE POR KNN.

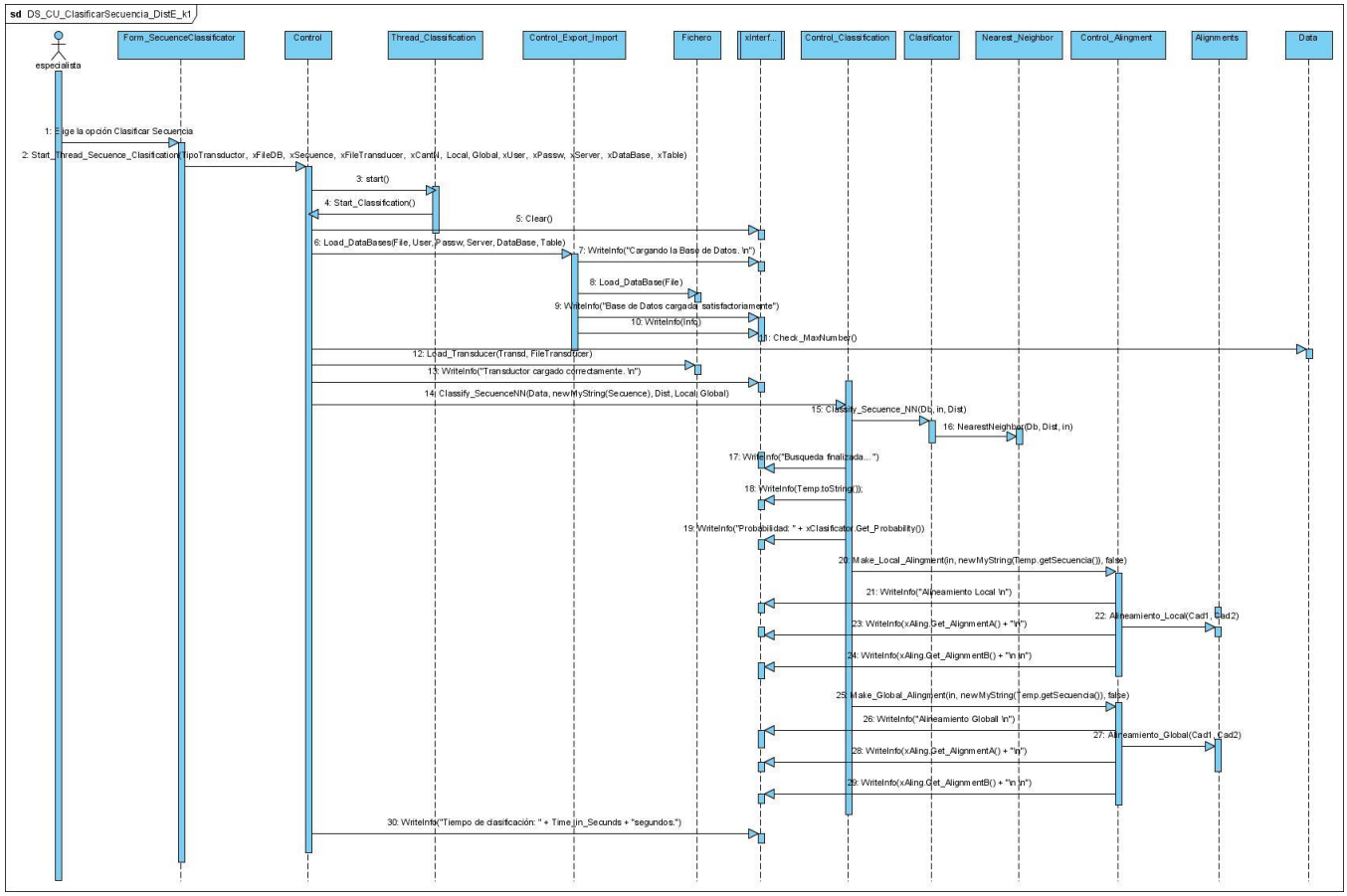


3. **ANEXO 3:** DIAGRAMAS DE SECUENCIA POR CU.

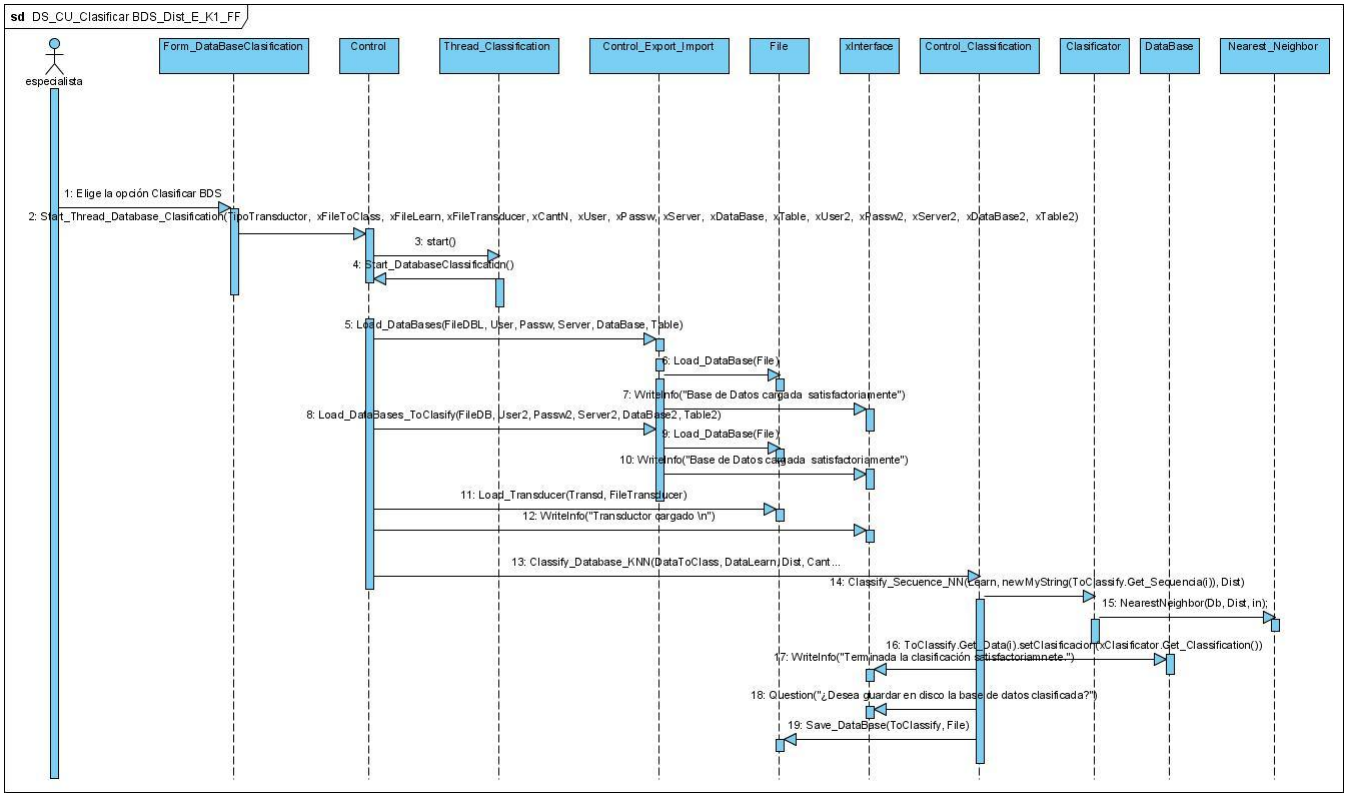
CU: REALIZAR APRENDIZAJE: **(BD importada y distribución condicional para el transductor).**



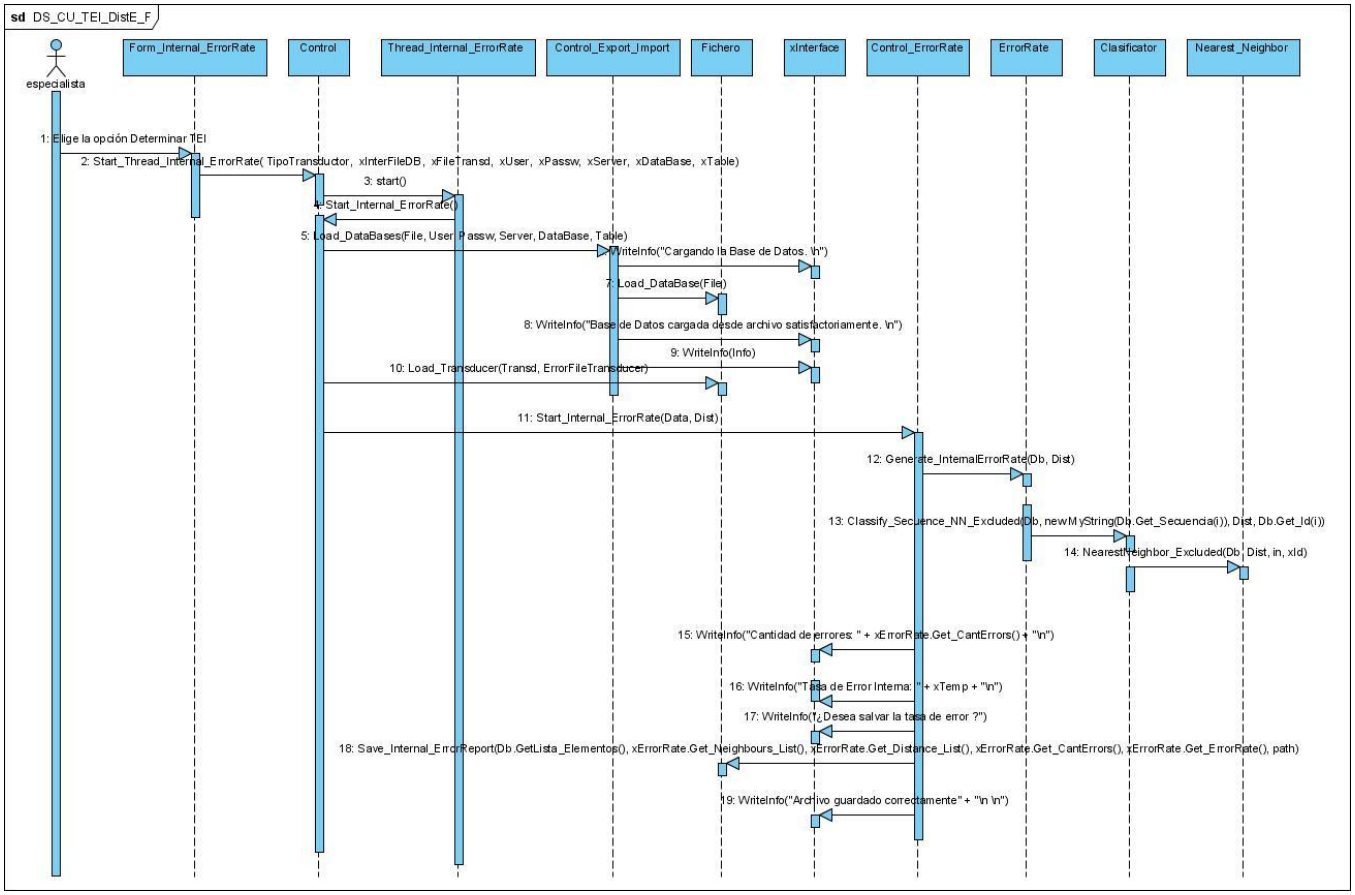
CU: CLASIFICAR SECUENCIA: **(BD importada y distancia de edición estocástica).**



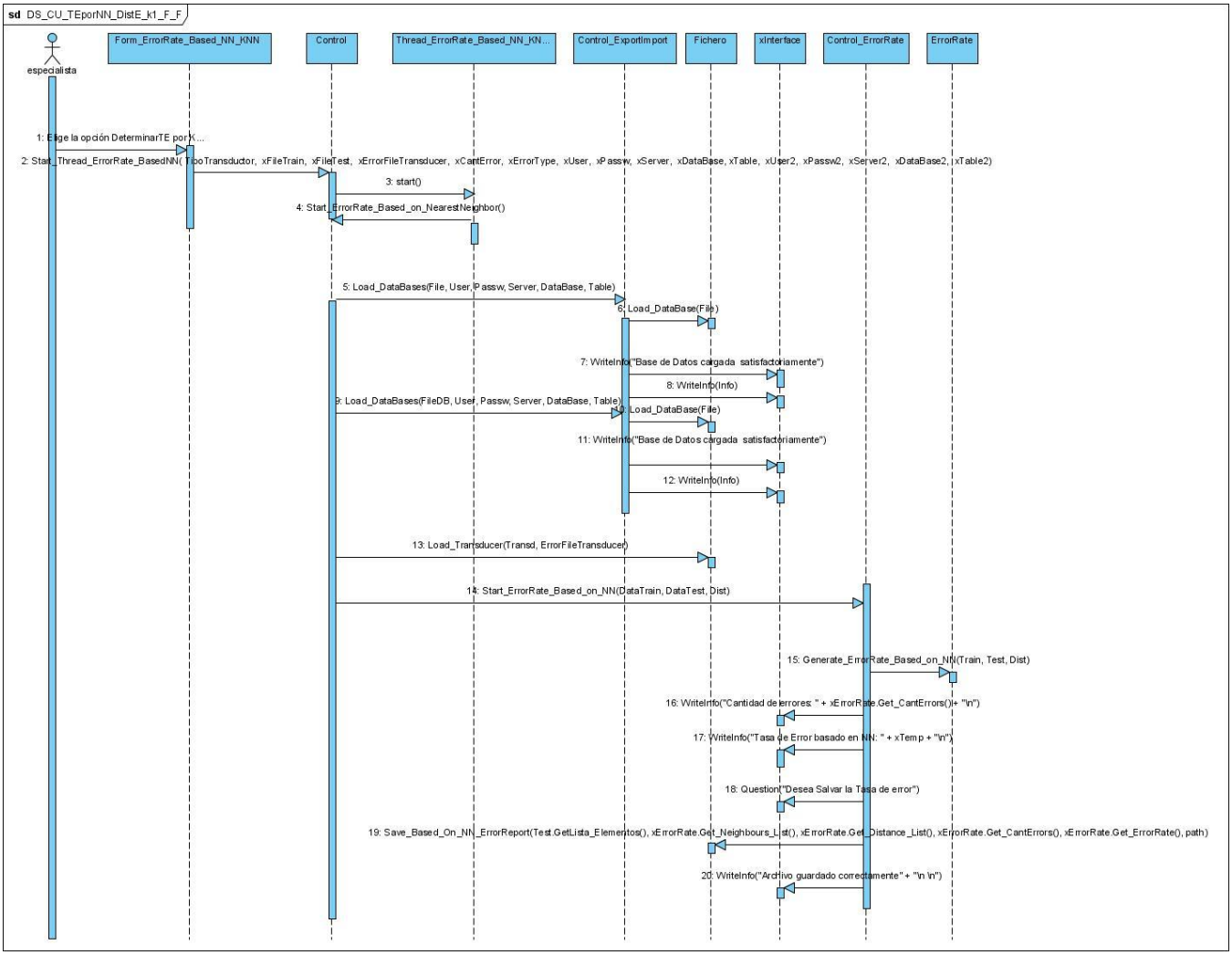
CU: CLASIFICAR BDS: **(BD importada y distancia de edición estocástica).**



CU: DETERMINAR TEI: **(BD importada y distancia de edición estocástica).**



CU: DETERMINAR TE POR KNN: **(BD importada y distancia de edición estocástica).**



GLOSARIO

BLAST: Basic Local Alignment Search Tool.

Heurística: En computación, dos objetivos fundamentales son encontrar algoritmos con buenos tiempos de ejecución y buenas soluciones, usualmente las óptimas. Regla que permite orientar un algoritmo hacia la solución de un problema. Y la heurística es una técnica de programación que permite a un sistema la creación gradual de un valor óptimo para una variable específica por medio del registro de los valores obtenidos en operaciones anteriores, es un método de ensayo y error para acercarse a la solución de un problema. No garantiza llegar a la solución pero puede acelerar el proceso para determinarla.

NN: (Nearest Neighbor). Vecino más cercano.

KNN: (K-Nearest Neighbor). K vecino más cercano.

TE: Tasa de error.

TEI: Tasa de error interna.

TE por K-NN: Tasa de error por el K (K>=1) vecino más cercano.

BD: Base de datos.

BDA: Base de datos de aprendizaje.

BDC: Base de datos a clasificar.

BDP: Base de datos de prueba.

Label: Clasificación que se corresponde con el microorganismo en la BD.

Transductor: Matriz de probabilidades resultante del aprendizaje de los costos de las operaciones de edición empleado en el cálculo de distancia de edición estocástica.