# Universidad de las Ciencias Informáticas

## Facultad 6, Bioinformática


# Título: “Aplicación web del ICA para la construcción de una base de datos de genes de microorganismos del rumen, basada en un rápido algoritmo de reconocimiento de caracteres”

**Trabajo de diploma para optar por el título de Ingeniero Informático**

**Autores:**

Rachid Alí Grave de Peralta.

Yumis Figueroa Suárez.

**Tutores:**

MSc. Abiel Roche Lima.

MSc. Alina Agramonte Delgado.

**Junio - 2009**

**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 _____.

Rachid Alí Grave de Peralta

_____

Firma del Autor

Abiel Roche Lima

_____

Firma del Tutor

Yumis Figueroa Suárez

_____

Firma del Autor

Alina Agramonte Delgado

_____

Firma del Tutor

**Datos de contacto**

MSc. Abiel Roche Lima.

Email: abiel.roche@gmail.com

MSc. Alina Agramonte Delgado.

Universidad de las Ciencias Informáticas, Ciudad de La Habana, Cuba.

Email: alinaad@uci.cu

**Agradecimientos**

*Yumis Figueroa Suárez*

*En primer lugar quiero agradecerle a Rachid, por siempre confiar en mí y saber que yo podía llegar hasta aquí, porque aún cuando existieron momentos en los que pensé que este sueño no se me haría realidad ahí estaba él para demostrarme lo contrario; por brindarme su apoyo incondicional desde primer año y por ayudarme a ser hoy lo que soy: "el orgullo de mi familia", por todo esto siempre le estaré eternamente agradecida, gracias por todo rachi.*

*Quiero agradecerle además a mi familia por estar siempre pendiente de mí, en especial a mi mamita por su sacrificio para conmigo; a mi abuela Bertha por su constante dedicación desde mis estudios en la secundaria básica, a mi abuela Estrella por su preocupación y cariño, a mi abuelo Alfonso por su compresión incondicional y todo el amor y cariño que he recibido de él desde pequeña. A mis hermanos Ernestico, Mabel y Ritica por ser tan buenos conmigo y por hacer que me sienta orgullosa de ellos. A Haydee, Karen y Ana por su inmenso cariño.*

*A mis amigos mermere y arian, por estar siempre a mi lado haciéndome saber que no estaba sola, por brindarme su mano cada vez que la necesitaba y por secar mis lágrimas cuando fue necesario. Ustedes han sido un tesoro para mí que siempre voy a conservar y me demostraron el verdadero significado de la amistad, por eso nunca los voy a olvidar.*

*A mi tutora Alina por su constante dedicación y entereza, por estar siempre pendiente de nosotros y por ser una mujer tan maravillosa. A mis amistades que de una u otra forma me brindaron su apoyo y se preocuparon por mí y a Roche por crear este proyecto. A todos los profesores que han influido en mi desarrollo profesional y a esta Revolución por haberme permitido realizar mis sueños.*

*Rachid Alí Grave de Peralta*

*Agradezco a mi familia, los que están y los que lamentablemente no. A mis padres por la excelente educación y el gran sacrificio a lo largo de todos estos años. A mis 4 abuelos por tan valiosos consejos y tanto cariño, los que ya no están… gracias. A mi hermana Karen, mi tío Ramón, mi hermano Roilan, mis tías, primos y la gente del Barrio. Eterno agradecimiento a mi tío Peralta y mi tía Sarita que han sido mis padres en estos cinco años brindándome su ayuda incondicional, indispensable en tantas ocasiones. Especial agradecimiento a Yumis por el* ***inmenso amor****, su apoyo desinteresado e incondicional, gracias a su esfuerzo y dedicación fueron posibles muchas cosas, a la familia de Yumis, por su ayuda y cariño. A nuestra tutora Alina, por tan alto apoyo profesional, total entrega, dedicación y por la gran persona que es. A Roche por enseñarnos a hacer ciencia, precursor de esta idea que logramos. A mis viejos y queridos amigos del pre, a mis amigos de Chaparra. A mis amigos del grupo y el apartamento a quienes agradezco por ser como una familia en la UCI. A mis inseparables amigos Dairon y Raúl. A amigos especiales como Norlen, el Diole, Reynaldo Rosado, San Juan. A mis alumnos por enseñarme tanto. A la revolución, a todo el que hizo posible la creación y funcionamiento de la UCI, a los profesores de la carrera a nuestro querido comandante en jefe Fidel y al pueblo trabajador de Cuba al cual me debo.*

**Dedicatoria**

*Yumis Figueroa Suárez*

*En primer lugar dedico esta tesis a mi mamá Rita, quien ha sido todo para mí, quien siempre confió en mí y supo que llegaría este momento tan especial en mi vida, la que nunca me dejó sola a pesar de haber pasado por momentos difíciles.*

*A mis abuelos Estrella, Bertha, Alfonso y Leonel por ser tan lindos conmigo, por apoyarme siempre en todo y por darme fuerzas para luchar y continuar cada vez que los miro.*

*A mi hermana Ritica por ser tan especial conmigo y por confiar en mí.*

*A mi tío Ernesto que más que un tío ha sido como un padre para mí y ha sido mi ejemplo a seguir en la vida desde pequeña, él siempre supo que yo llegaría hasta aquí cuando ni yo misma sabía lo que quería.*

*A papi, a quien quiero como mi propio padre porque siempre se ha comportado como tal y sus consejos han sido muy valiosos para mí y han servido de mucho en mi formación tanto humana como profesional.*

*A mi papá, porque a pesar de no ser de la forma que me hubiera gustado que fuera sé que me quiere y sé que puedo contar con él cuando lo necesite.*

*A mi "mermere" por ser mi amiga, por ayudarme a crecer, por regañarme cuando ha hecho falta, por ayudarme a corregir mis errores y por estar siempre ahí cuando la he necesitado.*

*A titi por ser tan especial para mí, por sus consejos, por siempre apoyarme y preocuparse por mis estudios y por ser tan maravillosa como persona.*

*Rachid Alí Grave de Peralta*

*Dedico esta tesis a mi querida familia. Especialmente a mis padres por tanto amor y sacrificio, mi hermana, mi tío Ramón y mis abuelos. A mis tíos Sara y Paco que son mis padres también. A mi tutora Alina, a Yumis por todo, a mis amigos de siempre, a la Revolución, a Fidel y al pueblo trabajador de Cuba.*

## Resumen

Uno de los primeros pasos para la caracterización de un ecosistema es la descripción de los organismos que habitan en él. En estudios microbianos, las limitaciones experimentales han impedido en algunos casos la correcta caracterización y discriminación entre diversas comunidades. Con la incorporación de la tecnología necesaria para la secuenciación de genes en el Instituto de Ciencia Animal (ICA) surge la necesidad por parte de los investigadores de crear una base de datos propia para el almacenamiento y gestión de la información. En este trabajo se propone una aplicación web capaz de gestionar la información a partir de una base de datos de genes de *ARN ribosomal 16S*[1] de microorganismos del *rumen*[2], brindando los servicios básicos para el análisis de secuencias. Se destaca entre ellas una novedosa técnica para la clasificación taxonómica basada en un algoritmo de búsqueda de patrones de homología entre genes flanqueados por *cebadores*[3] específicos extraídos de GenBank. Se consideraron las distancias de ocurrencias entre los cebadores dados, así como la permisividad de hasta tres errores y la presencia de caracteres degenerados. Nuestro sistema se validó utilizando un conjunto prueba de secuencias conocidas, las cuales fueron extraídas previamente del análisis y se lograron clasificar correctamente en un 87 % de los casos. Finalmente se logró asignar un *grupo taxonómico*[4] a 304 microorganismos que se corresponden con un 92.68% de los microorganismos imposibles de cultivar en el laboratorio.

**Palabras claves**: microorganismos del rumen, análisis de secuencias, genes ARN 16S, clasificación taxonómica.

# Tabla de contenido

## Introducción

En los últimos 10 años la genómica se ha desarrollado como consecuencia de los avances realizados en biología molecular e informática, dos áreas de la ciencia que han tenido un desarrollo tecnológico enorme, generando una revolución en el conocimiento y en la comprensión de los procesos biológicos. Las herramientas que se utilizan para el análisis individual de genes o pequeñas regiones cromosómicas, se aplican al análisis global de genomas completos, estudiando en conjunto los miles de genes, proteínas y metabolitos que constituyen un organismo, así como las complicadas redes de interacciones que operan entre ellos. La información generada es enorme y es clave para la identificación y el aislamiento de genes de interés y permite interpretar en términos moleculares los procesos biológicos. Para ayudar en este proceso han surgido poderosas herramientas bioinformáticas que permiten almacenar e interpretar esta información. En el mundo existen diversas bases de datos biológicas que almacenan una enorme cantidad de información, siendo las más representativas: GenBank, perteneciente al Centro Nacional de Información Biotecnológica (NCBI, http:\\NCBI HomePage.htm) de Estados Unidos (1), la EMBL del Instituto de Bioinformática de Europa (EBI, http:\\EMBL Nucleotide Sequence Database.htm) y la DDBJ del Instituto Nacional de Genética de Japón (NIG, http:\\DDBJ Homepage.htm).

Existen también las bases de datos dedicadas a comunidades específicas de interés para el hombre, donde se guardan registros de los organismos que se han podido secuenciar y que habitan en un nicho común. Estas bases de datos son muy útiles para hacer estudios vinculados al comportamiento y evolución de las diferentes poblaciones que coexisten dentro de una comunidad. En nuestro país existen algunas bases de datos específicas de bacterias y hongos como son: la del Centro de Bioproductos Marinos (CEBIMAR) y la del Acuario Nacional de Cuba que guardan una extensa colección de especies marinas. Sin embargo dichas bases de datos se limitan a guardar la información, y no integran herramientas informáticas para el análisis y gestión de la misma. Los investigadores del Instituto de Ciencia Animal (ICA) estudian por su parte la comunidad microbiana del rumen, la cual constituye un ejemplo de ecosistema complejo donde residen billones de bacterias, archaeas, protozoos y hongos que por su variada constitución y la gran cantidad de información que guarda ha sido por mucho tiempo el eje de las investigaciones microbianas. Estos microorganismos cuyo hábitat natural es el rumen de organismos superiores juegan un papel importante en la digestión de los alimentos y suministro de nutrientes al hospedero. La comunidad microbiana que habita en el rumen, se caracteriza por su alta densidad de población, amplia diversidad y complejidad de

interacciones. En este órgano encontramos representantes de los tres dominios: *Bacteria*, *Archaea* y *Eucarya* y se estima que existen más de $10^{11}$ células por $mL^{-1}$, que representan alrededor de 200 especies bacterianas, entre $10^{4}$-$10^{6}$ células por $mL^{-1}$ distribuidos en 25 géneros de protozoos ciliados, así como alrededor de $10^{3}$-$10^{5}$ zoosporas por $mL^{-1}$ divididos en cinco géneros de hongos anaerobios y por último entre $10^{7}$-$10^{9}$ partículas de bacteriófagos por $mL^{-1}$) (2). En los últimos años se han establecido técnicas moleculares basadas en las secuencias de ARNr 16S para el estudio de estos microorganismos, lo que facilita la identificación y el monitoreo de estas poblaciones (3). Con la reciente incorporación de la tecnología necesaria para la secuenciación de genes en el ICA surge por parte de los investigadores la necesidad de crear una base de datos propia para el almacenamiento y gestión de la información, por lo que se plantea el siguiente ***problema científico***: ¿Cómo automatizar el almacenamiento y gestión de secuencias de microorganismos del rumen?

El problema planteado se enmarca en el ***objeto de estudio***: Aplicaciones web para el estudio de comunidades microbianas.

El objeto delimita el ***campo de acción***: Aplicación web para la gestión de una base de datos de genes de microorganismos del rumen.

Para dar solución al problema se define como ***objetivo general*****:** Desarrollar una aplicación web que permita el análisis y clasificación de secuencias a partir de una base de datos de genes de microorganismos del rumen.

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

- Crear una base de datos de genes de microorganismos del rumen con todas las secuencias de ARNr 16S disponibles en GenBank.
- Diseñar una aplicación web para el análisis y gestión de secuencias de la comunidad microbiana del rumen.
- Implementar un algoritmo de reconocimiento de cadenas basado en cebadores para la asignación de un grupo taxonómico a microorganismos imposibles de cultivar en el laboratorio.

Para lograr los objetivos propuestos, se realizarán las siguientes ***tareas***:

- Obtención y revisión de todas las secuencias de ARNr 16S correspondientes a microorganismos del rumen disponibles en GenBank.
- Diseño y creación de una base de datos de genes para la comunidad microbiana del rumen.
- Investigación de las tendencias y tecnologías actuales para la construcción de aplicaciones web dinámicas, específicamente aplicadas al análisis de comunidades microbianas.
- Selección e incorporación de las herramientas bioinformáticas básicas para el análisis de secuencias.
- Estudio de los métodos para explorar las comunidades microbianas a partir de sus secuencias de genes.
- Implementación de un algoritmo de reconocimiento de cadenas basado en cebadores para la asignación de un grupo taxonómico a microorganismos imposibles de cultivar en el laboratorio.
- Realización de las actividades del flujo de trabajo requerimientos.
- Realización de las actividades del flujo de trabajo de diseño.
- Realización de las actividades del flujo de trabajo implementación.
- Validación funcional de la aplicación.

El presente trabajo está estructurado en cuatro capítulos. A continuación aparece de manera resumida el contenido de cada capítulo:

En el primer capítulo se hace un análisis del estado del arte de los recursos y herramientas bioinformáticas aplicadas al estudio de comunidades microbianas, como una fuente importante de investigación médica y biotecnológica. Se presentan los fundamentos teóricos y prácticos que cimientan el desarrollo de la aplicación, destacando la importancia de la misma. Se analizan también una serie de investigaciones previas que sirven como base para fundamentar el uso de los algoritmos

propuestos. Por último se fundamenta el uso de las tecnologías, metodologías y herramientas a utilizar para el logro de un producto con calidad.

En el segundo capítulo, se brinda una breve caracterización del sistema a desarrollar, así como la descripción del modelo de dominio perteneciente al mismo. Además se modelan los artefactos correspondientes al flujo de trabajo de requerimientos.

En el tercer capítulo, se modelan artefactos correspondientes al flujo de diseño, para asegurarnos que el sistema cumpla con los objetivos propuestos. Se incluyen algunos de los diagramas de clases del diseño así como algunos casos de uso y el diagrama de despliegue.

En el cuarto capítulo, se implementa el sistema en términos de componentes partiendo de los requerimientos identificados y de los diagramas de clases del diseño elaborados. Se muestran fragmentos relevantes del código y describe los aspectos relacionados con la prueba de la solución propuesta teniendo en cuenta los casos de prueba de funcionalidad, específicamente pruebas de caja negra con la técnica de partición de equivalencia.

# Capítulo 1: Fundamentación teórica

## 1.1 La informática y su relación con las ciencias biológicas.

El desarrollo de la biotecnología y la informática se ha incrementado vertiginosamente en los últimos años. Durante la última década del siglo XX, los avances de la ingeniería genética y las nuevas tecnologías de la información, condicionaron el surgimiento de una disciplina que creó vínculos indisolubles entre la informática y las ciencias biológicas: la bioinformática. Esta puede definirse como la aplicación de la informática, las matemáticas y la estadística para organizar, analizar y entender problemas que involucren secuencias de nucleótidos, aminoácidos o cualquier otro tipo de información biológica relacionada. De esta forma, la bioinformática es capaz de responder al reto de manejar grandes cantidades de información mediante el empleo de robustas bases de datos al tiempo que provee herramientas para facilitar el análisis de la misma; actividad que manualmente sería imposible de realizar.

## 1.2 Bases de datos de secuencias de ADN.

La colaboración de las tres bases de datos más importantes del mundo hace posible acceder a casi toda la información de secuencias de ADN desde cualquiera de sus tres sedes: EMBL-BANK del Instituto europeo de Bioinformática (EBI, http:\\EMBL Nucleotide Sequence Database.htm), la DNA Data Bank of Japan (DDBJ) del Instituto Nacional de Genética de Japón (NIG, http:\\DDBJ Homepage.htm) y GenBank del Centro Nacional de Información Biotecnológica (NCBI, http:\\NCBI HomePage.htm) de los Estados Unidos. Si bien son mantenidas por distintos organismos en distintos países, existe una coordinación entre las distintas bases. Una secuencia enviada a cualquiera de las bases se verá reflejada en las otras dos en aproximadamente una semana, ya que esa es la frecuencia de actualización promedio entre las mismas. Estas bases de datos son de acceso público por lo que son consultadas continuamente e incorporan herramientas que permiten realizar búsquedas por homología, alineamientos de secuencias con el objetivo de dilucidar la función de un nuevo gen encontrado, o refinar la búsqueda de un cebador específico para amplificar un gen característico de una patología.

Sin embargo en muchas ocasiones es necesario contextualizar el análisis a una comunidad en específico, por ello existen también las bases de datos dedicadas a comunidades de interés para el hombre, donde se guardan registros de los organismos que se han podido secuenciar y que habitan en un nicho común. Estas bases de datos son muy útiles para hacer estudios vinculados al comportamiento y evolución de las diferentes poblaciones que coexisten dentro de una comunidad. En nuestro país existen algunas bases de datos específicas como son: la del Centro de Bioproductos Marinos (CEBIMAR), elaborada en Acces y cuenta hasta el momento con una colección de 325 especies de bacterias así como la del Acuario Nacional de Cuba que guarda una extensa colección de 15 028 especies marinas correspondientes a 22 grupos taxonómicos diferentes. Sin embargo dichas bases de datos se limitan a guardar la información, y no integran herramientas informáticas para el análisis y gestión de la misma. Los investigadores del Instituto de Ciencia Animal (ICA) estudian por su parte la comunidad microbiana del rumen, la cual constituye un ejemplo de ecosistema complejo donde residen billones de bacterias, archaeas, protozoos y hongos que por su variada constitución y la gran cantidad de información que guarda ha sido por mucho tiempo el eje de las investigaciones microbianas. Estos microorganismos cuyo hábitat natural es el rumen de organismos superiores juegan un papel importante en la digestión de los alimentos y suministro de nutrientes al hospedero. Debido a que muchos de los microorganismos existentes en el rumen presentan características únicas de la región y el clima propios de su hábitat, la generación de una base de datos propia de Cuba constituye una opción más prometedora para los investigadores del ICA que la utilización de las grandes bases de datos internacionales.

## 1.3 Métodos para el estudio de comunidades microbianas.

Se sabe que el conocimiento de la composición de un ecosistema en cuanto a variedad y cantidad de organismos que lo habitan constituye un paso importante para el entendimiento de la ecología del mismo. Esta información nos permite comprender las *relaciones tróficas*[5], las rutas nutritivas, así como las interacciones competitivas existentes en diversos hábitats. Durante el estudio de plantas y animales así como de muchos protistas, los organismos pueden ser generalmente identificados y enumerados visualmente. Sin embargo en la mayoría de los procariontes no se pueden emplear estos métodos para su estudio por lo que los investigadores tienen que acudir al cultivo *in vitro* y a la caracterización de poblaciones individuales (4). En la década de los años 70, Woese y colaboradores (5) describen el uso del análisis comparativo de ARNr para estudios filogenéticos. Estos resultados no sólo

proporcionaron las bases evolutivas para la taxonomía en procariontes, sino que también dieron lugar a la organización en tres reinos del mundo viviente: Archaea, Bacteria, y Eucarya (5)

Por otra parte existe una gran variedad de microorganismos imposibles de cultivar en el laboratorio, ya sea por las características inhóspitas de su hábitat o por su lento crecimiento, por lo tanto estos métodos son solo aplicables a un grupo limitado de comunidades (4). Estos métodos proveen una medida de la abundancia relativa o de la actividad metabólica de los integrantes de la población. A pesar de que existen muchas técnicas moleculares útiles para la identificación de cepas y especies a partir de una muestra de ADN extraída de cultivos puros, solo unas pocas sirven para la clasificación de mezclas heterogéneas de ADN. En los últimos años se han empleado métodos independientes del cultivo basados en el análisis de secuencias de genes de ARN ribosomal (ARNr) 16 y 18S, lo que ha permitido la identificación de miles de microorganismos anteriormente indefinidos (5) (6) y han sido empleados ampliamente para explorar la diversidad microbiana y entender la dinámica de sus comunidades. Los genes de ARNr son marcadores taxonómicos muy útiles porque se encuentran en todos los organismos conocidos conteniendo tanto regiones variables como altamente conservadas y tienen una baja tasa de cambio o un reloj molecular relativamente constante (7).

El análisis de comunidades microbianas usando genes de ARNr se puede llevar a cabo usando diversas aproximaciones simples. Dos de los métodos más usados son: la *electroforesis*[6] en gel por gradiente desnaturalizante o de temperatura (DGGE o TGGE, de sus siglas en inglés "*Denaturing or Temparature Gradient Gel Electrophoresis*") (8) el análisis del polimorfismo de longitud de fragmentos de restricción terminales de genes de ARNr 16S y 18S (T-RFLP, de sus siglas en inglés "Terminal Restriction Fragmente Length Polymorphisms"), ambos permiten el análisis de muchas muestras en un período de tiempo relativamente corto. Cada uno de estos métodos se basa en la capacidad de amplificación de los genes y proveen al investigador un perfil (o huella digital) de la comunidad que refleja la composición de la población numéricamente dominante en la muestra (4).

### 1.3.1 Algoritmos de clasificación taxonómica a partir de secuencias de ADN.

Durante los últimos cinco años, ha sido factible la inclusión de técnicas de metagenómica a gran escala basadas en el análisis de fragmentos de genomas de microorganismos de ambientes naturales, o el uso de metodologías alternativas de secuenciación como la pirosecuenciación para el estudio de la diversidad en comunidades microbianas naturales. Científicos como O. Bejá, E. F. DeLong o J. C. Venter fueron pioneros en el uso de distintas técnicas metagenómicas para el estudio de comunidades

microbianas marinas. Craig Venter en el año 2004 mediante la técnica metagenómica *whole-genome shotgun sequencing* (WGS) basada en la secuenciación al azar de los genomas de una comunidad microbiana, quien hizo posible la secuenciación de más de un millón de genes a partir de una muestra del bacterioplancton del Mar de los Sargazos detectando más de 1800 especies o filotipos (secuencias distintas del gen ARNr 16S (9). Dicha metodología ha permitido la identificación de más de un millón de nuevos marcos abiertos de lectura (ORFs, del inglés *Open Reading Frames*), muchos de ellos con la posibilidad de codificar nuevas enzimas. Recientemente, se aplicó esta tecnología para la exploración de la estructura de una comunidad microbiana marina en profundidad (10) y se pudieron secuenciar más de 900000 blancos (pequeños fragmentos de alrededor de 100 bp de los genes ARNr 16S) de dos comunidades microbianas bentónicas cercanas a una fisura volcánica que se encuentra a más de 1500 m de profundidad (11). En la actualidad, el acceso a dichas tecnologías es restringido y por lo tanto la mayoría de los laboratorios todavía utilizan la misma estrategia para determinar la diversidad microbiana. Dicha estrategia se basa en la amplificación por PCR[7] (reacción en cadena de la polimerasa) de determinados marcadores moleculares como el gen 16S ARNr, a partir del ADN extraído directamente de los microorganismos presentes en una comunidad microbiana. Posteriormente, se genera una *genoteca*[8] y se secuencia un número determinados de clones para su análisis. Obviamente, cuantos más clones se secuencian más probabilidades se tiene de recuperar secuencias representativas de los microorganismos existentes. Seguidamente se procede a la identificación por *microarrays*[9] de los genes de ARNr a través de una serie de experimentos de hibridación usando pequeñas sondas de ADN. Borneman y colaboradores en el 2002 acuñaron el método como *Oligonucleotide fingerprinting of rRNA Genes* (OFRG) (4) y sus resultados han sido comparados con los obtenidos por electroforesis en gel por gradiente desnaturalizante demostrándose las significativas ventajas del mismo. Dicho método se ha convertido en una poderosa estrategia para el análisis a gran escala de comunidades microbianas con aplicaciones en la medicina y la biotecnología.

Al igual que muchas investigaciones en ciencia experimental, el proyecto OFRG (12) ha generado una enorme cantidad de datos que necesitan ser almacenados, procesados y analizados. Se hacía imprescindible entonces un conjunto de herramientas apropiadas para el manejo y análisis de los datos que faciliten el proceso experimental. Liu y colaboradores (2005) (7) proponen un sistema web para el manejo de datos, que puede servir como repositorio para resultados experimentales entre otras informaciones relevantes del proceso de hibridación. Además el sistema integra herramientas para el diseño de grupos de prueba, de entrenamiento, el agrupamiento y análisis estadístico (Fig.1); cuya

calidad tiene un fuerte impacto en las investigaciones de comunidades microbiana (3). A partir de genes de secuencia conocida y de secuencias blanco se genera una huella digital artificial, la cual constituye un vector binario cuyo valor en cada posición se define como 1 si la secuencia del gen contiene la sonda correspondiente y 0 si no la contiene. Dicha huella digital artificial constituye una aproximación para la estimación de la capacidad del método OFRG durante la identificación de microorganismos. Más específicamente, si el árbol filogenético generado a partir de la huella digital es muy similar al que se genera a partir de las secuencias de genes; entonces la huella es confiable y por lo tanto puede ser usada para la identificación de microorganismos indefinidos.

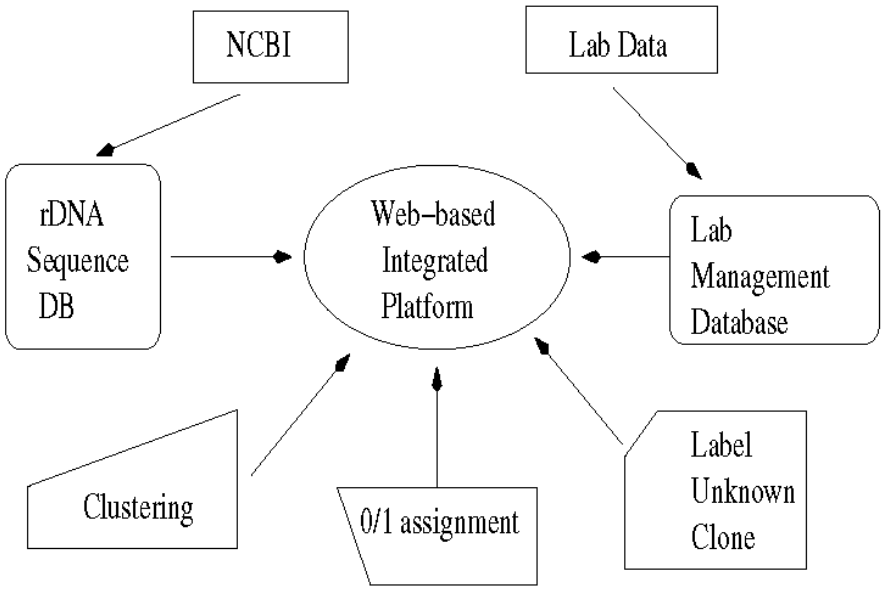



**Figura 1:** Sistema de manejo de datos para OFRG.

En la literatura existen gran cantidad de sistemas de base de datos de ADNr disponibles, entre ellos: *Ri-bosomal Database Project* (13) y *The European database of small subunit ribosomal RNA* (14) por solo mencionar algunos. En ocasiones dichos sistemas toman las secuencias directamente de GenBank sin procesar, mientras que otros realizan alineamientos múltiples basados en la estructura secundaria del ARN. La mayoría de estas bases de datos de secuencias se focalizan en brindar variadas herramientas de análisis para la reconstrucción filogenética y búsqueda de similitud entre secuencias. En el presente trabajo se implementa y aplica el algoritmo descrito por Liu y colaboradores (2005) (7) para el estudio de la comunidad microbiana del rumen. El mismo propone un sistema para

la adquisición de secuencias y construcción de una base de datos que posibilite la generación automática de una sub-base de datos de ADNr constituida por genes flanqueados por cebadores específicos. La especificación de una pareja de cebadores es imprescindible para la obtención de secuencias de un grupo taxonómico en particular. El sistema además le permite al usuario extraer aquellas secuencias que aunque no contengan los cebadores son homólogas al grupo taxonómico de interés, esto se hace empleando el algoritmo BLAST (del inglés, *Basic Local Aligment Sequence Tool*). Lo más significativo del sistema es que reduce considerablemente el tiempo de procesamiento de las secuencias de semanas (proyecto OFRG) a horas (7).

### 1.3.2 Algoritmo de reconocimiento de cadenas basado en cebadores.

El sistema para la adquisición de secuencias y construcción de la base de datos provee una interface web asequible al usuario. El uso del sistema requiere de tres pasos fundamentales para la construcción de una base de datos personalizada. Primeramente el usuario debe cargar un fichero de secuencias de genes en formato FASTA al servidor que constituirá la Base de Datos Primaria (BDPri). Dichas secuencias pueden ser adquiridas de GenBank u obtenidas de algún experimento de laboratorio. Luego es necesario especificar una pareja de cebadores que encierren al gen de interés. Seguidamente se aplica a la BDPri un algoritmo degenerado de reconocimiento de cadenas para obtener un subconjunto de secuencias que contienen ambos cebadores. A este subconjunto se le llama Base de Datos Patrón (BDP), la cuál será usada como blanco en la subsecuente búsqueda por homología. La diferencia entre la BDPri y la BDP se define como Base de Datos Restante (BDR). Finalmente se realiza una búsqueda por homología para encontrar aquellas secuencias de la BDR que pertenecen al mismo grupo taxonómico que las de la BDP a pesar de no haber macheado con los cebadores específicos.

Durante la etapa de generación de la BDP, es necesario aplicar un algoritmo degenerado eficiente y seguro para el reconocimiento de cadenas. Existen tres diferencias entre dicho algoritmo y el macheo de cadenas tradicional.

- No solo se admiten errores, sino que la secuencia de una pareja de cebadores puede contener caracteres degenerados, lo cual significa que un carácter puede representar varias alternativas.
- Se utiliza un par de secuencias en vez de una simple secuencia patrón.
- Se consideran las distancias de ocurrencia entre ambos cebadores, para evitar falsos positivos.

La última etapa de búsqueda por homología se hace necesaria porque es posible que la BDPri contenga fragmentos de genes que no contengan los cebadores o que los contengan parcialmente. Para detectar dichas secuencias se realiza la búsqueda por homología utilizando el algoritmo BLAST, que genera alineamientos locales entre las secuencias de la BDR y de la BDP. Aquellas secuencias de la BDR que alineen con una puntuación significativa (e-value ≤ 0.05) con secuencias de la BDP se considerarán homólogas. De esta manera se puede inferir el grupo taxonómico a partir de la secuencia de un microorganismo imposible de cultivar en el laboratorio, lo que constituye la herramienta principal de la aplicación, por su notable repercusión en el estudio de las comunidades microbianas y específicamente en la comunidad microbiana del rumen.

### 1.3.3 El problema de los k-desajustes degenerados

El reconocimiento de cadenas de caracteres es uno de los principales retos en la ciencia de la computación debido a que usualmente representa un importante consumo de tiempo y memoria. Es por ello que muchas aplicaciones como procesadores de texto y analizadores de secuencias de genes demandan un algoritmo eficiente para el reconocimiento de cadenas de caracteres. Existen dos importantes variantes del problema de reconocimiento aproximado de cadenas de caracteres: el problema de los k-desajustes y el problema de las k-diferencias (15). En ambos casos, dados una cadena patrón P = p1 p2…pm y un texto T = t1 t2…tn donde m<n en un alfabeto Σ, y un entero k; el problema de los k-desajustes consiste en encontrar todas las ocurrencias del patrón P en el texto T permitiendo a lo sumo k desajustes. Mientras que el problema de las k-diferencias consiste en encontrar todas las subcadenas de T con distancia de edición ≤ k.

En el presente trabajo se implementa una variante del problema de los k-desajustes, definida por Liu y colaboradores en el 2005 (7) como *el problema de los k-desajustes degenerados.* El problema se formula de la siguiente forma:

Dado un conjunto D de secuencias de ADNr de longitud *n* y una pareja de cebadores *a* y *b* con longitud *m* que pueden contener caracteres degenerados; el problema consiste en identificar todas las secuencias que macheen con *a* y *b* permitiendo a lo sumo k desajustes. Las distancias de ocurrencia entre *a* y *b* están limitadas por la mínima distancia $l_{mín}$ y la máxima distancia $l_{máx}$. El problema de los k-desajustes degenerados permite k-desajustes, en adición a los caracteres degenerados. Para secuencias de ADNr, k ≤ 3 y la longitud de un cebador es alrededor de 20, dentro de los cuales más de 4 caracteres pudieran ser degenerados.

En el trabajo publicado por Liu y colaboradores (2005) (7) se realiza una comparación entre los algoritmos de macheo aproximado de cadenas de caracteres más relevantes, entre los cuales se encuentran: (Baeza-Yates and Perleberg) (16), Baeza-Yates and Gonnet (16), y Tarhio and Ukkonen (17). Se demuestra experimentalmente la solución al problema de los k-desajustes permitiendo además caracteres degenerados mediante el uso de un algoritmo propio. La novedad del mismo consiste en la combinación de los algoritmos Baeza-Yates-Gonnet y Tarhio-Ukkonen (18), escogidos estos últimos por resultar más prácticos y acordes a la comparación de cadenas de ADN. Ambos son considerados la generalización del algoritmo de Boyer-Moore (19) para el macheo exacto de patrones, variando solamente en la vía para el cálculo de la distancia de cambio, siendo esta, la mínima cantidad de corrimientos del patrón antes de encontrar un carácter similar perteneciente a los k+1 caracteres del texto T en el alineamiento previo. El principio básico de estos algoritmos es ignorar en la búsqueda la mayor cantidad de caracteres posibles que no constituyan una ocurrencia del patrón e ir calculando las distancias de cambio. La combinación propuesta por Liu y colaboradores (7) consiste en escoger la máxima distancia de cambio calculada por cada uno separadamente e ir registrándola en una tabla teniendo en cuenta los k-caracteres degenerados. Los experimentos realizados comprobaron que la combinación de ambos algoritmos posee un tiempo de complejidad menor que cada uno por separado.

## 1.4 Tecnologías, metodologías y herramientas a utilizar.

Para un eficiente desarrollo de software es de vital importancia una correcta selección de las tecnologías, metodologías y herramientas que se usarán, poniendo en práctica esta importante premisa se hizo un estudio de las más usadas en la actualidad. Luego de un análisis riguroso, entre los desarrolladores y los clientes se tomaron los acuerdos de cuáles elegir para desarrollar la aplicación. A continuación se presentan los principales aspectos considerados que fundamentan dicha elección.

### 1.4.1 Metodología de desarrollo.

En aras de concebir una forma ordenada de trabajo se hace necesario el uso de una metodología de desarrollo. La misma es un proceso que integra y guía las múltiples facetas del desarrollo y que además, ofrece criterios para el control y la medición de los productos. “Todo desarrollo de software es riesgoso y difícil de controlar, pero si no llevamos una metodología de por medio, lo que obtenemos es clientes insatisfechos con el resultado y desarrolladores aún más insatisfechos” (20). Durante la

investigación realizada se tomaron en cuenta varias de las metodologías candidatas a usar, entre las cuales destacan:

**Extreme Programming (XP)** o programación extrema, una metodología reciente en el desarrollo de software. La filosofía de XP es satisfacer al completo las necesidades del cliente, por eso lo integra como una parte más del equipo de desarrollo. XP fue inicialmente creada para el desarrollo de aplicaciones donde el cliente no sabe muy bien lo que quiere, lo que provoca un cambio constante en los requisitos que debe cumplir la aplicación. Por este motivo es necesaria una metodología ágil como XP que se adapta a las necesidades del cliente y dónde la aplicación se va reevaluando en períodos de tiempo cortos. XP está diseñada para el desarrollo de aplicaciones que requieran un grupo de programadores pequeño, dónde la comunicación sea más factible que en grupos de desarrollo grandes. La comunicación es un punto importante y debe realizarse entre los programadores, los jefes de proyecto y los clientes. XP posee excelentes características válidas a la hora de escogerlo como metodología, estas son:

• **Comunicación:** Los programadores están en constante comunicación con los clientes para satisfacer sus requisitos y responder rápidamente a los cambios de los mismos. Muchos problemas que surgen en los proyectos se deben a que después de concretar los requisitos que debe cumplir el programa no hay una revisión de los mismos, pudiendo dejar olvidados puntos importantes.

• **Simplicidad:** Codificación y diseños simples y claros. Muchos diseños son tan complicados que cuando se quieren ampliar resulta imposible hacerlo y se tienen que desechar y partir de cero.

• **Realimentación**: Mediante la realimentación se ofrece al cliente la posibilidad de conseguir un sistema apto a sus necesidades ya que se le va mostrando el proyecto a tiempo para poder ser cambiado y poder retroceder a una fase anterior para rediseñarlo a su gusto.

**RUP**

Es un proceso de desarrollo de Software, o sea es el conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema software. Sin embargo, el Proceso Unificado de desarrollo de software es más que un simple proceso; es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas de software, para diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de aptitud y diferentes tamaños de proyecto (20).

Existen tres características claves presentes en RUP, ellas son: dirigido por casos de uso, centrado en la arquitectura, e iterativo e incremental. RUP se descartó como posible metodología de desarrollo debido a que el equipo de desarrollo es pequeño, solamente dos personas y se necesitaba una metodología ágil. No obstante sí se estudió a fondo esta metodología para aprovechar sus ventajas y adquirir una experiencia valiosa en el desarrollo de la aplicación.

**OpenUP**

OpenUP es un proceso unificado, abierto, mínimo y suficiente lo que brinda la posibilidad de solo incluir el contenido fundamental y necesario. Conserva las características principales de la metodología de desarrollo RUP por lo que se aplica de forma iterativa e incremental, provee los componentes básicos que pueden servir de base a procesos específicos a pesar de no tener lineamientos para todos los elementos que se manejan en un proyecto. Es una forma de desarrollo ágil y ligero donde la mayoría de sus elementos fomentan el intercambio entre los equipos de desarrollo, y de colaboración sincronizando intereses y compartiendo conocimientos, principio fundamental que promueve las buenas prácticas para propiciar un ambiente saludable y de colaboración. Maximiza los beneficios al equilibrar las prioridades, permitiendo a los desarrolladores desarrollar una solución que cumpla con los requerimientos del proyecto, minimiza el riesgo al centrarse en la arquitectura y proporciona la retroalimentación y mejoramiento continuo al realizar prácticas que permitan incrementos progresivos a las funcionalidades.

Teniendo en cuenta lo antes explicado la metodología que se utilizará para guiar el desarrollo de las funcionalidades es OpenUp debido a sus características, descritas, y a ser este modelo el que más se ajusta a las necesidades del desarrollo de la aplicación, además de ser la metodología escogida y utilizada en el polo de Bioinformática de la facultad.

### 1.4.1.1 Patrones.

Un patrón es una pareja de problema/solución con un nombre, que codifica (estandariza) buenos principios y sugerencias. Con el paso de los años la utilización de patrones ha evolucionado dando mejor precisión al reflejar los requisitos reales y haciendo más fácil el trabajo con los sistemas. (21)

### Patrones de Casos de Uso.

**CRUD** (Crear, Buscar, Modificar y Eliminar)

Este patrón plantea la creación de un caso de uso, llamado “Información CRUD” o “Gestionar Información”, que modele todas las operaciones que se pueden realizar sobre una parte de información de un tipo determinado, ya sea crearla, leerla, actualizarla y eliminarla. Es importante destacar que este patrón se usa cuando todos los flujos contribuyen al mismo valor de negocio y son cortos y simples. Este patrón es usado en la aplicación para la gestión de información.

**Actores múltiples:** Rol común

Si dos actores interpretan el mismo papel en determinado caso de uso, este rol es representado por otro actor, que contiene de forma hereditaria los actores que lo comparten. Este patrón debe aplicarse cuando, desde el punto de vista de un caso de uso hay solo una entidad externa interactuando con cada instancia del caso de uso.

### Patrones de diseño.

Es difícil imaginar un buen software sin el apropiado uso de los patrones de diseño, por esta fuerte razón se hizo énfasis en el estudio de los patrones de diseño más importantes. Se puede definir un patrón, además, como un proyecto o estructura de implementación que logra una finalidad determinada donde se pone de manifiesto la manera más práctica de describir ciertos aspectos de la organización de un programa. En el caso de los patrones computacionales, un software estructurado, modulado posee una mejor calidad y es más sencillo corregir errores, implementar mejoras y actualizaciones, debido a que un software que posee algún patrón de diseño es más sencillo de modificar que un software que no posee en absoluto un patrón. Los patrones de diseño proponen una forma de reutilizar la experiencia de los desarrolladores, para ello clasifica y describe formas de solucionar problemas que ocurren de forma frecuente en el desarrollo. Por tanto, están basados en la recopilación del conocimiento de los expertos en desarrollo de software, lo que lo convierte en su principal ventaja (22).

### 1.4.2 Lenguaje de Modelado (UML).

UML es el lenguaje utilizado para visualizar, especificar, construir y documentar cada una de las partes que constituye un sistema de forma general. Permite la modelación de sistemas con tecnología orientada a objetos. No es una guía para realizar el análisis y diseño orientado a objetos, es decir, no es un proceso. UML está compuesto por diversos elementos gráficos que se combinan para conformar diagramas. EL UML (Lenguaje Unificado de Modelado) es una de las herramientas más gustadas en el mundo para desarrollo de software, esta particularidad de su uso se debe a que permite a los desarrolladores de sistemas, generar diseños conforme a sus ideas de forma fácil y entendible. Por estas razones su uso se hizo vital para hacer un correcto modelado del sistema.

### 1.4.3 Herramientas CASE.

Se puede definir a las Herramientas CASE como un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos y procesos del Ciclo de Vida de desarrollo de un proyecto (23). Entre las herramientas CASE para el modelado de artefactos que existen a nivel mundial están las herramientas MetaCASE, Embarcadero ER/Studio, Erwin, Umbrello, MagicDraw, Rational Rose y Visual Paradigm. Esta última herramienta CASE, Visual Paradigm for UML en su versión 6.1 es la seleccionada para el modelamiento de los diferentes artefactos.

#### 1.4.3.1 Visual Paradigm como Herramienta CASE.

Se escoge Visual Paradigm porque es una herramienta CASE que soporta el UML como lenguaje de modelado. Además provee el Modelado del Proceso de Negocio, un generador de mapeo objeto-relación para Java, .NET y PHP. Una característica clave es su habilidad de generar no solo código del modelo de clases, sino también de la estructura de la base de datos relacional adecuados para sostener persistentemente la información contenida en las clases llamadas entidad. Genera la documentación del proyecto automáticamente en varios formatos como web, .pdf, .doc, y permite el control de versiones. Es una herramienta multiplataforma de modelado visual UML ya que aporta a los desarrolladores de software una plataforma de desarrollo para construir aplicaciones de calidad. Aporta una excelente interoperabilidad con otras herramientas CASE y muchos de los entornos IDE del mercado como NetBeans y Eclipse. Permite el uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.

### 1.4.4 Herramientas de desarrollo usadas.

El uso de una herramienta de desarrollo es vital durante el flujo de trabajo implementación, permitiendo así el completamiento de código, agilizar el trabajo del programador en todo momento y depurar el código.

#### 1.4.4.1 Eclipse como entorno integrado de desarrollo (IDE).

Un entorno de desarrollo integrado o IDE es un programa compuesto por un conjunto de herramientas para un programador. Los IDEs proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación. Eclipse es una plataforma de herramientas universal y portable que proporciona un framework para desarrollar aplicaciones y herramientas. Esto significa que distintos fabricantes pueden desarrollar e integrar sus herramientas con el workbench existente. El workbench es un conjunto de Java Frameworks y herramientas de desarrollo equipadas para constructores de herramientas. El entorno de trabajo presenta un entorno de desarrollo integrado en perspectivas personalizables. Las perspectivas son combinaciones formadas por vistas y editores que muestran los diversos aspectos de los recursos del proyecto y están organizados por el rol o la tarea del desarrollador. Se proporcionan más de una perspectiva en un momento dado, además de pasar de una a otra mientras se está trabajando. Eclipse ofrece un conjunto de valiosas ventajas entre las cuales se pueden apreciar solo por citar algunas que todo el contenido está almacenado en archivos. Los recursos como clases java, archivos HTML, archivos XML (descriptores de despliegue), están almacenados en un sistema de archivo y así se puede tener fácilmente acceso a ellos (24). Es un IDE potente y posee un amplio soporte, su gran facilidad de integración mediante el uso de plugins lo hace compatible con muchos de los frameworks más usados. A estas características se le añade la experiencia de los desarrolladores en el uso de Eclipse.

### 1.4.5 Servidor de aplicaciones.

Un servidor de aplicaciones es un software que proporciona aplicaciones a los equipos o dispositivos cliente, por lo general a través de internet y utilizando el protocolo http. Los servidores de aplicación se distinguen de los servidores web por el uso extensivo del contenido dinámico y por su frecuente integración con bases de datos.

### 1.4.5.1 Apache Tomcat.

Tomcat es un servidor web con soporte de Servlets y JSPs. Incluye el compilador Jasper, que compila páginas JSP convirtiéndolas en Servlets. El motor de servlets de Tomcat a menudo se presenta en combinación con el servidor web Apache. Tomcat puede funcionar como servidor web por sí mismo. En sus inicios existió la percepción de que el uso de Tomcat de forma autónoma era sólo recomendable para entornos de desarrollo con requisitos mínimos de velocidad y gestión de transacciones. Hoy en día ya no existe esa percepción y Tomcat es usado como servidor web autónomo en entornos con alto nivel de tráfico y alta disponibilidad. Se decide optar por el uso de Tomcat en su versión 6.0 por las ventajas mencionadas.

### 1.4.6 Tecnología J2EE

J2EE puede ser considerada la mejor plataforma disponible actualmente para el desarrollo de aplicaciones empresariales (25). Esta plataforma es producto de las lecciones en desarrollo de software empresarial del pasado. Los beneficios que brinda la plataforma son bien conocidos y pueden resumirse en:

- **Simplicidad:** el desarrollador puede concentrarse en la lógica del negocio, sin tener que invertir mucho tiempo en problemas de bajo nivel como: seguridad, transacciones, multi-threading, protocolos de seguridad, ambientes distribuidos, pool de recursos (por ejemplo acceso a base de datos), etc.
- **Separación de capas:** J2EE promueve el diseño de una aplicación en múltiples capas, permitiendo en la mayoría de los casos la separación de presentación y funcionalidad, acorde con los principios de buenas prácticas de diseño.
- **Portabilidad:** La aplicación J2EE puede ser migrada a otro servidor de aplicaciones con cambios mínimos.
- **Múltiples opciones de servidores:** Dado que J2EE es un estándar, existe la posibilidad de escoger entre múltiples servidores, tanto productos comerciales como código abierto.

### 1.4.7 Lenguajes de Programación para la Web.

Los lenguajes se clasifican en dos partes fundamentales que reconocen la propia arquitectura Cliente/Servidor de esta plataforma de desarrollo: los lenguajes del lado del servidor y los lenguajes del lado del cliente. Del lado del cliente se encuentran principalmente el lenguaje Java Script, que es el encargado de aportar dinamismo a la aplicación en los navegadores y además tienen la responsabilidad de ejercer funciones específicas como la validación y la impresión. Java Script es un lenguaje de programación muy utilizado por todos los programadores para crear pequeños programas encargados de realizar acciones dentro del ámbito de una página web. HTML es un lenguaje básico para los programadores que comienzan a incursionar en el campo del desarrollo web, es por eso que después del HTML, Java Script es considerado el nivel superior o el siguiente paso, que puede dar un programador de la web con vistas a decidirse por perfeccionar el estilo de sus páginas y la potencia de sus proyectos con el propósito de obtener un mayor alcance.

En el dominio de la red, los lenguajes del lado del servidor más ampliamente utilizados para el desarrollo de páginas dinámicas son el ASP, PHP, PERL y Java.

Java ofrece todas las ventajas de un lenguaje potente y robusto, pues fue diseñado para crear software altamente fiable (25). Es un lenguaje de programación basado en clases y orientado a objetos. Sus características de memoria liberan a los programadores de responsabilidades y errores. Java 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. Java como lenguaje se ajusta perfectamente al tipo de aplicación a realizar y cumple con las necesidades planteadas por los desarrolladores y clientes (26). Agregado a estas ventajas el uso de java se fundamenta debido al empleo de librerías escritas en este lenguaje para el análisis de secuencias y por ser el mismo definido por el polo de bioinformática de la facultad.

### 1.4.8 Framework usados.

Un framework es un término utilizado en la computación en general, para referirse a un conjunto de bibliotecas, utilizadas para implementar la estructura estándar de una aplicación. Todo esto se realiza con el propósito de promover la reutilización de código, con el fin de ahorrarle trabajo al desarrollador al no tener que rescribir ese código para cada nueva aplicación que desee crear. Existen diferentes frameworks para diferentes propósitos, algunos orientados al desarrollo de aplicaciones web, otros para desarrollar aplicaciones multiplataforma, para un sistema operativo o lenguaje de programación en específico, entre otros. “El framework captura las decisiones de diseño que son comunes a su dominio de aplicación” (27). Un framework no sólo promueve la reutilización de código sino también la reutilización de diseño. Además las aplicaciones que se construyen tienen estructuras similares, son más fáciles de mantener y consistentes para los usuarios.

#### 1.4.8.1 Framework para aplicaciones web.

Actualmente existen algunos frameworks para desarrollar aplicaciones web, que es de las ramas más importantes en las que se usan los frameworks. La mayoría de ellos utilizan el patrón de diseño MVC del cual se hablará más adelante en el capítulo 3.Todos los frameworks tienen características especiales que los hacen únicos para sobresalir y poder seguir en el mercado, además de poseer las siguientes características comunes:

• Utilizan un solo servlet que tiene la función de controlador, para toda la aplicación o gran parte de ella. Se configura un descriptor de despliegue para que todas las URL’s tengan que pasar forzosamente por dicho servlet.

• Una configuración, generalmente escrita en un archivo XML, en donde se le indicará al servlet controlador, a través de propiedades, a quien delegar la responsabilidad de atender la petición entrante. Algunas veces esas propiedades están indicadas de acuerdo a los URL’s y de acuerdo al URL entrante es como se delega la responsabilidad.

Las vistas pueden tener nombres claves, sin necesidad que exista una relación con el nombre del archivo de la vista. El framework se encarga de realizar dicha conversión para poder obtener el nombre de la vista que se tiene que cargar para que sea desplegada. La implementación de una vista con un nombre en particular puede cambiar sin afectar código del controlador. Cuando se va a desarrollar una

aplicación web, el desarrollador se debe fijar en si desea realizar una aplicación extremadamente sencilla o si quiere desarrollar una verdadera aplicación web, que tendrá actualizaciones, correcciones y mejoras a futuro. Para esto es recomendable que se elija un framework de aplicación web que utilice el patrón web MVC. Con el fin de que se pueda hacer la separación entre los 3 elementos principales y pueda aprovechar todas las ventajas que brinda este patrón de diseño. Para poder aprovechar también las ventajas adicionales que brinda el framework para la integración con otras herramientas u otros servicios. A continuación se muestran algunos de los frameworks para aplicaciones web más populares: Struts, WebWork, Maverick y Spring. Para hacer una correcta selección se analizaron sus principales características, ventajas y desventajas (28).

**Struts.**

Struts es uno de los frameworks MVC más utilizados, ya que fue uno de los pioneros en el campo, fue creado por Craig McClanahan, creador del famoso motor servlet Tomcat, ambos son distribuidos por Apache. Este framework fue lanzado a mediados del año 2000, y a partir de esta fecha comenzó a tener popularidad, y en la actualidad existen algunos componentes extra que se adaptan a Struts, lo que le da un mayor campo de acción. Struts es código abierto, por lo que no requiere licencia para su uso. Además cumple una de las funcionalidades básicas que se mencionaron acerca de los frameworks para aplicaciones web, ya que implementa un solo controlador (ActionServlet) que evalúa las peticiones del usuario mediante un archivo configurable (struts-config.xml) para poder dirigirlas a un action que procesa el request. El lenguaje de programación que utiliza es Java, así como Java Beans como modelo.

**Algunas de las ventajas y desventajas de Struts:**

• Existe un variado número de trabajos y proyectos ya hechos lo que brinda un mayor número de ejemplos para poder tomar un punto de partida y de referencia.

• Brinda librerías de etiquetas usadas para mostrar la vista bastante útiles y eficientes que permiten la eficiente reutilización de código.

• Entre las desventajas de framework se puede observar que muchas veces puede ser difícil trabajar con los ActionForms, además de que el proyecto posiblemente desaparezca en los años venideros.

Otra de las desventajas de Struts es que está muy ligado con la tecnología JSP, por lo que muchas veces se dificulta integrarlo con alguna otra tecnología para las vistas.

**Maverick.**

Este es otro framework MVC de código abierto que existe en el mercado, pero a diferencia de los demás este no cuenta con sus propias librerías de etiquetas. Sin embargo cumple con las funcionalidades típicas mencionadas, como el de tener un solo servlet controlador central como punto de entrada, el cual lleva el nombre de Dispatcher, que está definido en el Deployment Descriptor de la aplicación web (web.xml). Maverick cuenta con un archivo XML en el que se guarda toda la configuración del mismo (maverick.xml). Una característica de Maverick es que únicamente acepta un solo controlador central y un archivo de configuración por aplicación web, lo que muchas veces al desarrollar aplicaciones más grandes y complejas se puede volver confuso y difícil de configurar.

Maverick es un framework mucho más configurable que Struts, lo que brinda cierta flexibilidad, también incluye varias clases para poder extender y cambiar el flujo del trabajo o workflow en inglés. Es usualmente usado para crear nuevos controladores que sean capaces de procesar nuevas peticiones. Los controladores son Java Beans, y el framework pone de manera transparente las propiedades de los beans. Una característica interesante de Maverick es el básico y fácil uso de la funcionalidad del Dispatcher. También es multiplataforma ya que ha sido adaptado para los lenguajes .NET y PHP.

Este es un framework más reciente ya que fue lanzado a mediados del año 2002.Una de las personas que ha trabajado en este proyecto fue Rickard Oberg, quien ha participado en proyectos como JBoss, entre otros.

**WebWork**

El framework WebWork está basado en el patrón de diseño de comandos o Command Pattern. Cada acción es un comando, que es creado para manejar un request, sus propiedades son puestas de manera transparente, ya que cada acción es un Java Bean. WebWork cuenta, al igual que Struts y Spring con su propia librería de etiquetas para JSP, las cuales ayudan a realizar distintas tareas de una manera más ágil, pero no es la única tecnología para vista que soporta, también incluye soporte para Velocity. Una de las ventajas de WebWork es que cuenta con una arquitectura simple y las clases son fáciles de extender. Pero como todo, tiene sus desventajas, la creación de una acción (action) por

cada petición (request) puede llegar a ser algo confuso cuando existen muchos datos en el request se impone el patrón de diseño Command en cada interacción del usuario, ya sea bueno o malo utilizarlo en dicha interacción; es difícil saber de qué tipo son las excepciones que se lanzan. Como este framework es relativamente reciente no existe mucha documentación y ejemplos al respecto lo que puede resultar a veces frustrante cuando se requiere buscar algún ejemplo o tutorial que pueda servir.

**Spring.**

Spring propone estructurar toda una aplicación de una manera consistente y productiva, conjuntando lo mejor de cada uno de los frameworks single-tier para crear una arquitectura coherente. En sí Spring ha surgido como una solución para poder disfrutar de los beneficios clave de J2EE, mientras que minimiza la complejidad encontrada en el código de la aplicación (27).

Spring está basado en la filosofía de que un framework debe proveer una guía hacia una buena práctica. Mezclando la correcta combinación de flexibilidad y restricción, la cual es la clave en el buen diseño de un framework. Spring también cuenta con algunas de las características generales de los frameworks web, ya que cuenta con un módulo para poder desarrollar aplicaciones web de manera sencilla. También se basa en el diseño de las clases extendiendo o implementando interfaces, lo cual es un mejor diseño orientado a objetos, ya que promueve la reutilización de código. Además Spring cuenta con algunas etiquetas para las diferentes tecnologías de vista que existen, las cuales tienen funciones diferentes como vincular formularios, validación, despliegue de información, entre otras. Uno de los aspectos clave y ventajas de Spring, es que cuenta con una arquitectura modularizada, y se pueden utilizar cada uno de estos módulos de manera independiente. Cada uno está enfocado en una tarea específica, y algunos de ellos son para la integración con alguna herramienta o incluso cuenta con la posibilidad de integrase con los otros frameworks mencionados anteriormente.

En conclusión, cada framework tiene sus ventajas y desventajas, escoger uno es una decisión del desarrollador, dependiendo de sus necesidades y requerimientos, del tamaño del proyecto a desarrollar, así como de su experiencia con tecnologías existentes que se integren fácilmente con el framework que haya elegido. Esta fue una de las razones por la cual se decidió incursionar dentro de las características de Spring, y después de un análisis se llegó a la conclusión de que es un excelente framework que cumple con la mayoría de las características, de las cuales otros frameworks escasean. Una de las cuestiones que ha hecho a Spring evolucionar de manera muy rápida en tan poco tiempo, es que se han incrementando de una manera increíble el número de proyectos que utilizan esta

herramienta a través del último año. Esto conlleva a que la cantidad de personas interesadas en este framework aumente, así como el soporte, el número de foros de opinión y la publicación de un mayor número de libros y tutoriales.

### 1.4.8.2 Acegi Security System como framework de seguridad.

Para dar soporte de seguridad se decidió usar Acegi Security System, el mismo es un framework creado por Ben Alex e íntimamente ligado al proyecto Spring. Facilita la tarea de adoptar medidas de seguridad en aplicaciones Java, sean aplicaciones standalone o aplicaciones web como es el caso. Además se encuentra bajo licencia GPL, sin coste y con la seguridad añadida que proporciona el respaldo de un enorme y creciente grupo de usuarios que lo usan. Posee una excelente documentación y un amplio soporte.

Acegi provee una implementación segura y portable, con todas las características deseadas y que además es independiente de cualquier mecanismo de seguridad provisto un servidor específico. Esto permite cambiar el servidor de aplicación (por razones de costo, escalabilidad o la que fuere) sin ninguna modificación en el esquema de seguridad de la aplicación (29). Esto puede ser un problema o no dependiendo de la probabilidad de cambiar el servidor. Otra característica potente de Acegi es el aprovechamiento de las ventajas de la inversión de control de Spring, la habilidad de cambiar la inyección de dependencia en tiempo de ejecución. Esto permitirá cambiar los parámetros de autenticación y autorización en caso de un cambio en el modelo de seguridad utilizado. Un ejemplo podría ser un cambio en la estructura de roles, que se trasladaría en cambios en la configuración de roles autorizados para acceder a los recursos. Estos cambios pueden ser realizados sin ningún tipo de re-despliegue de los archivos de configuración. Sin embargo habría que analizar el impacto que tendrían los cambios de la configuración si hay que realizar cambios estructurales en la aplicación, lo que tal vez implicaría un re-despliegue de todos modos. De esta forma estas capacidades de cambio en tiempo de ejecución no serían una gran diferencia respecto a la seguridad en sí (29).

### 1.4.8.3 Hibernate como framework para la persistencia de datos.

Para la persistencia de datos en la aplicación se decidió el uso de Hibernate, una herramienta de mapeo objeto-relacional para la plataforma Java que facilita el mapeo de atributos entre una base de

datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos (XML) que permiten establecer estas relaciones (30). Hibernate es software libre, distribuido bajo los términos de la licencia GNU LGPL. Como todas las herramientas de su tipo, hibernate busca solucionar el problema de la diferencia entre los dos modelos de datos coexistentes en una aplicación: el usado en la memoria de la computadora (orientación a objetos) y el usado en las bases de datos (modelo relacional) (30). Con esta información hibernate le permite a la aplicación manipular los datos de la base operando sobre objetos, con todas las características de la programación orientada a objetos (POO). Estas características cumplen con las necesidades que se exigen para la construcción de la base de datos usada en la aplicación.

### 1.4.9 Gestores de Bases de Datos.

"El software que permite la utilización y/o la actualización de los datos almacenados en una (o varias) base(s) de datos por uno o varios usuarios desde diferentes puntos de vista y a la vez, se denomina sistema de gestión de bases de datos (SGBD)" (31). Un SGBD permite crear y mantener una base de datos, asegurando su integridad, confidencialidad y seguridad. Poseen lenguajes de definición de datos (DDL), lenguajes de manipulación de datos (DML) y lenguajes de consulta (SQL) que facilitan el proceso de diseño de aplicaciones y que los tratamientos sean más eficientes y rápidos, dando la mayor flexibilidad posible a los usuarios. Esto se le da cumplimiento a partir de los objetivos fundamentales de los sistemas de bases datos, entre los que se puede mencionar: independencia de los datos y los programas de aplicación; minimización de la redundancia; integración y sincronización de las bases de datos; integridad de los datos, seguridad y recuperación; facilidad de manipulación de la información y control centralizado.

#### 1.4.9.1 PostgreSQL.

Como resultado de un estudio acerca de los gestores de base de datos se tomaron en cuenta las principales características de este gestor. 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. Este gestor posee una gran escalabilidad, ya que 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. Además tiene la capacidad de comprobar la integridad referencial, así 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 (31).

### 1.4.10 Herramientas bioinformáticas para el análisis de secuencias de ADN.

Las herramientas de software para bioinformática van desde aplicaciones de línea de comandos hasta programas gráficos y servicios web autónomos situados en compañías privadas o instituciones públicas. Haciendo uso de las herramientas bioinformáticas básicas para el análisis de secuencias, se pone en manos de los investigadores del ICA un valioso sistema capaz de contribuir notablemente al estudio de la comunidad microbiana del rumen. Se realizó un análisis de las herramientas más usadas y acorde a las necesidades de los clientes.

#### BLAST para la búsqueda de homologías.

BLAST (de sus siglas en inglés, *Basic Local Alignment Search Tool*) es la herramienta más frecuentemente usada para calcular similitud entre secuencias, tanto de ADN como de proteínas (32). El programa es capaz de comparar una secuencia problema contra todas (o un subconjunto de) las secuencias publicadas en una o varias bases de datos públicas de secuencias. La consulta se envía a través de internet, la búsqueda se realiza en los servidores NCBI, DDBJ o EMBL y los resultados se devuelven al buscador del usuario y se muestran en el formato deseado. Sin embargo, en muchas ocasiones, se necesita utilizar el BLAST, contra una base de datos local, como es el caso del presente trabajo y para ello existe también la versión portable de la herramienta.

#### ClustalW para alineamiento múltiple de cadenas.

Un alineamiento múltiple de secuencias es un alineamiento de tres o más secuencias biológicas, generalmente proteínas, ADN o ARN. Las representaciones visuales del alineamiento ilustran mutaciones tales como mutaciones puntuales (un solo cambio de aminoácidos o nucleótidos) que aparecen como diferentes caracteres en una sola columna del alineamiento, y la inserción o supresión de mutaciones que aparecen como huecos en una o varias de las secuencias en la alineación. El alineamiento múltiple de secuencias a menudo se utiliza para evaluar la conservación de los dominios

proteicos, las estructuras terciarias y secundarias, e incluso aminoácidos o nucleótidos individuales (33).

### Análisis filogenético.

El conocimiento de las relaciones filogenéticas que presentan todos los organismos entre sí desde el inicio de la vida hasta nuestros días, siempre ha ocupado un sitio importante dentro de la comunidad científica en especial y de la humanidad en general. La construcción de árboles filogenéticos ha sido de gran utilidad para entender comportamientos similares a nivel biológico entre diferentes especies. (34). A través de los años se han propuesto varios métodos para realizar estas inferencias filogenéticas, siendo en la actualidad los desarrollados a partir de técnicas de biología molecular de los más utilizados y confiables. En biología se utilizan árboles parecidos a los genealógicos para conocer cómo se encuentran emparentados los organismos vivos, se les conoce como árboles filogenéticos. A diferencia de los árboles genealógicos, en los que se utiliza información proporcionada por los familiares, para los árboles filogenéticos se usa información proveniente de fósiles así como aquélla generada por la comparación estructural y molecular de los organismos (34).

Un árbol filogenético es un grafo que muestra las relaciones de evolución entre varias especies u otras entidades que se cree que tuvieron una descendencia común. Representa los grados de parentesco evolutivo entre diferentes taxones con indicación de antecesores, duración de líneas evolutivas o cantidad de cambio evolutivo. Para el análisis filogenético en esta aplicación se usa la librería PAL por sus siglas en inglés (Phylogenetic Analysis Library) (35), es un proyecto colaborativo y desarrollado bajo licencia GPL, provee un grupo de funciones compatibles con estructuras de datos y métodos robustos para realizar construcción, manipulación, simulación y cálculos estadísticos de árboles filogenéticos. La versión más reciente es la 1.5, la cual tiene significativas mejoras. PAL permite diferentes análisis entre los cuales se destacan: escritura y lectura de alineamientos de secuencias, matrices de distancias evolutivas, estimación de parámetros demográficos, construcción de árboles usando un alineamiento múltiple previo, usando algoritmos conocidos como: neighbour-joining UPGMA, y sUPGMA (35).

A modo de conclusión, se analizaron los principales conceptos vinculados al tema para una mejor comprensión del lector. Además se fundamentaron las tecnologías a usar por los desarrolladores, programadores y analistas para el proceso de desarrollo del sistema en cuestión. Como resultado del estudio de las tecnologías y herramientas de desarrollo a utilizar se ha llegado a la conclusión de que el sistema se desarrollará sobre la plataforma J2EE usando como lenguaje de programación Java, y como gestor de bases de datos PostgreSQL 8.2. Se determinó además hacer uso del lenguaje unificado de modelado (UML) por su estrecha integración con RUP que constituye la base de OpenUP, metodología adoptada. Como herramienta de modelado se utiliza Visual Paradigm. Como entorno de desarrollo integrado se decidió que debe ser Eclipse y como servidor de aplicaciones se adoptó Tomcat 6.0. Para el análisis de secuencias de ADN se determinó el uso de las herramientas para la bioinformática BLAST, CLUSTALW 2.0 en el caso de búsqueda por homologías y alineamientos múltiples respectivamente, así como el uso de la librería PAL para el análisis filogenético. Como frameworks se determinó usar para la web Spring, para la seguridad Acegi Security y en la persistencia hibernate.

# Capítulo 2: Características del sistema.

## 2.1 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. El modelo de dominio se describe mediante diagramas UML (específicamente mediante diagramas de clases).

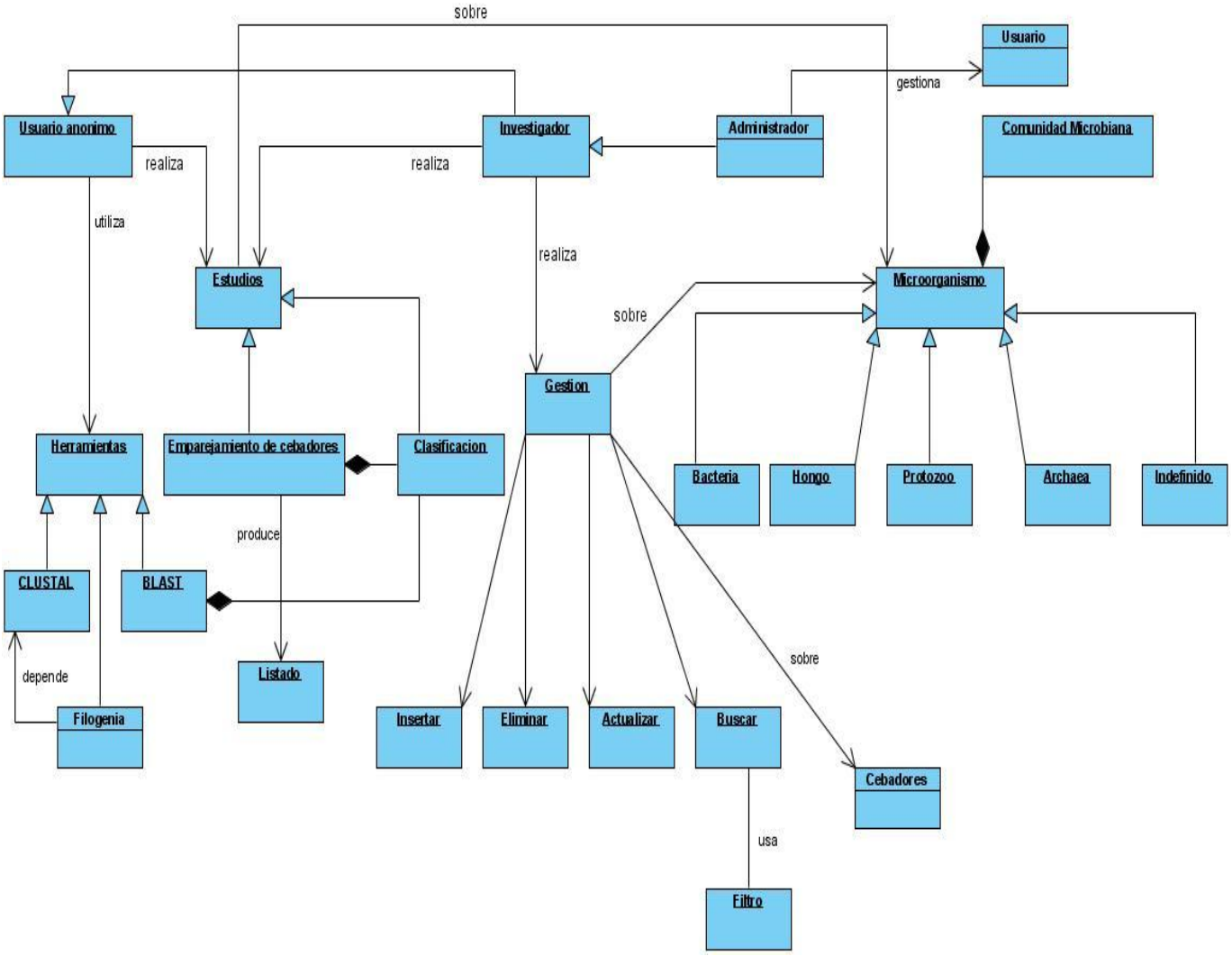



**Figura 2:** Modelo de dominio.

### 2.1.1 Descripción del modelo de dominio.

El sistema gestiona la información de la comunidad microbiana del rumen, la cual está compuesta por los datos de microorganismos divididos en cuatro grupos: Archaeas, Bacterias, Hongos, Protozoos y no identificados. Se almacena también las referencias de las parejas de cebadores y usuarios registrados en el sistema. La gestión consiste en las acciones insertar, eliminar, actualizar y buscar. Los usuarios se dividen en tres grupos: usuarios anónimos, investigadores y administradores. Los primeros no poseen los permisos para realizar ningún tipo de gestión, solamente pueden acceder a las herramientas de acceso público. El investigador puede realizar la gestión de la información relacionada con los cebadores y microorganismos, el usuario administrador además de poseer los mismos permisos del investigador puede realizar gestión sobre la información de los usuarios. Tanto las herramientas como los estudios que se pueden realizar son de pleno acceso para todos los usuarios. Entre las herramientas se encuentran: Alineamiento múltiple de secuencias usando el Clustal, búsqueda de homologías usando el BLAST y análisis filogenético. Se pueden realizar dos tipos de estudios, uno es la búsqueda de secuencias que contienen un gen flanqueado por una pareja de cebadores y el otro es la clasificación taxonómica de microorganismos, este último está compuesto por el primer estudio y la herramienta BLAST.

## 2.2 Especificación de los requisitos de software.

El propósito del flujo de trabajo de Requerimientos es guiar el desarrollo del sistema. Esto se consigue mediante una descripción detallada de los requisitos del sistema que permite llegar a un acuerdo entre el cliente y los desarrolladores sobre qué debe hacer el sistema.

### 2.2.1 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 importar con que propiedades o cualidades se relacionen. Los requerimientos funcionales que se identificaron para el desarrollo del software son:

**R1** Gestionar registro.

**R1.1** Insertar registro.

**R1.1.1** Insertar nuevo registro.

**R1.1.2** Insertar registro desde archivo.

**R1.2** Eliminar registro.

**R2** Analizar secuencias.

**R2.1** Realizar búsqueda por homología usando el algoritmo BLAST-N.

**R2.1.1** Realizar búsqueda por homología usando el algoritmo BLAST-N dada una secuencia de ADN.

**R2.1.2** Realizar búsqueda por homología usando el algoritmo BLAST-N subiendo un archivo de secuencias en formato FASTA.

**R2.2** Realizar alineamientos múltiples de secuencias usando el CLUSTALW 2.0.

**R2.2.1** Realizar alineamiento múltiple de secuencias usando el CLUSTALW 2.0 desde listado.

**R2.2.2** Realizar alineamiento múltiple de secuencias usando el CLUSTALW 2.0 subiendo un archivo de secuencias en formato FASTA.

**R2.2.3** Descargar resultado del alineamiento múltiple.

**R2.2.4** Descargar las secuencias que fueron alineadas.

**R3** Realizar reconstrucción filogenética usando la librería PAL.

**R3.1** Realizar reconstrucción filogenética usando la librería PAL desde listado.

**R3.2** Realizar reconstrucción filogenética usando la librería PAL desde archivo.

**R3.3** Descargar el árbol.

**R4** Gestionar cebadores.

**R4.1** Insertar parejas de cebadores.

**R4.2** Eliminar parejas de cebadores.

**R4.3** Editar parejas de cebadores.

**R4.4** Listar parejas de cebadores.

**R5** Clasificación taxonómica a partir de la secuencia de microorganismos imposibles de cultivar en el laboratorio.

**R5.1** Buscar registros que poseen genes flanqueados por cebadores específicos.

**R5.2** Generar Base de Datos Patrón (BDP) con los registros encontrados.

**R5.3** Ejecutar algoritmo BLAST para realizar búsqueda por homología en el Resto de la Base de Datos (RBD) partiendo de la BDP.

**R5.4** Crear una base de datos resultante que agrupe la BDP con la salida del BLAST hasta un score significativo.

**R5.5** Visualizar la clasificación.

**R6** Gestionar usuario.

**R6.1** Insertar usuario.

**R6.2** Eliminar usuario.

**R6.3** Modificar usuario.

**R6.4** Mostrar listado de usuarios.

**R7** Buscar registro.

**R8** Autenticar usuario.

**R9** Cambiar contraseña.

### 2.2.2 Requerimientos no funcionales.

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. En muchos casos son fundamentales en el éxito del producto ya que son las características que hacen a este atractivo, usable, rápido o confiable. Normalmente están vinculados a requerimientos funcionales, ya que una vez se conozca lo que el sistema debe hacer se podrá determinar cómo ha de comportarse, qué cualidades debe tener o cuán rápido o grande debe ser. Son importantes para que clientes y usuarios puedan valorar las características no funcionales del producto. Para su clasificación existen múltiples categorías. A continuación se muestran algunas de ellas y las características de acuerdo al sistema propuesto.

**Apariencia o interfaz externa.**

La interfaz será agradable, sencilla y sugerente; ya que su diseño simulará una aplicación de escritorio, presentará por defecto un color azul claro y textos que muestran la acción a realizar con un tipo de letra verdana de tamaño 12px. Debe contener pocas imágenes y gráficos para acelerar la velocidad de respuesta del sistema; por lo que se emplearán íconos sencillos que representarán la operación a realizar. El sistema será diseñado para una resolución de pantalla de 1024 X 870 píxeles.

**Seguridad.**

La información será almacenada en una base de datos, dejando registradas todas las operaciones mediante copias de seguridad. El uso y manejo del sistema estará controlado; ya que la información podrá ser consultada y modificada solamente por el personal autorizado; estableciendo para ello una serie de roles con funcionalidades específicas. La aplicación estará funcionando las 24 horas, de esta forma es posible que los usuarios tengan acceso (según sus permisos) en todo momento a la información solicitada. Es muy importante que los servidores estén sincronizados correctamente con el día y hora, ya que si ocurre un fallo el sistema no funcionaría correctamente.

**Confiabilidad.**

Todas las salidas del sistema tendrán 100% de veracidad y precisión. El sistema tendrá la capacidad de recuperarse rápidamente ante cualquier fallo mediante las copias de seguridad que serán realizadas. Se verificará constantemente la conexión con el servidor y en caso de fallo se le notificará al usuario. En caso de alguna dificultad con el funcionamiento del sistema, el tiempo medio de reparación deberá ser menor de 1 día.

**Hardware.**

El servidor de base de datos y el servidor web deben tener como mínimo 1GB y 2GB de memoria RAM respectivamente. Este último debe contar con tecnología cliente servidor para un mejor manejo de las peticiones de los usuarios. El computador del usuario debe tener como mínimo 256 MB de memoria RAM. Además es necesario contar con tarjetas de red de 10/100 MB/s para la conexión.

**Software.**

Las estaciones de trabajo clientes utilizarán como sistema operativo GNU/Linux en sus diversas distribuciones o la familia de Windows superior a Windows 2000. Las máquinas clientes contarán con un navegador, ya sea el Internet Explorer en versiones superiores a la 5.0 o Mozilla Firefox en versiones superiores a la 1.5, ambos con las opciones de JavaScript y cookies habilitadas. El servidor web funcionará sobre el sistema operativo GNU/Linux y tendrá instalado el Apache Tomcat en su versión 6. El servidor de base de datos funcionará sobre el sistema operativo GNU/Linux y tendrá instalado el gestor de bases de datos PostgreSQL 8.2.

**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 contar con conocimientos especializados en biología para poder entender los resultados brindados por la aplicación. La interfaz será fácil de usar para los diversos usuarios que interactúen con ella. Tendrá una óptima navegabilidad, brindando las opciones más demandadas por el usuario en todo momento, ejemplo de ello es el buscador de secuencias de ADN.

**Legales.**

Los módulos de desarrollo de esta aplicación web del ICA, así como la documentación de la misma, son propios de la institución y específicamente del área de informática del centro.

## 2.3 Definición de actores y casos de usos del sistema.

Un actor es una entidad externa al sistema que interactúa con él. Siempre se beneficia de la realización de un caso de uso. Estimula el sistema con eventos de entrada y salida. Es un rol de un usuario que puede intercambiar información o puede ser un recipiente pasivo de información. Representa un ser humano, una máquina o un software que interactúa con el sistema.

**Tabla 1**: Actores del sistema.

| **Nombre del actor** | **Descripción** |
|---|---|
| Administrador | Representa el usuario del sistema que tiene acceso a todas las funcionalidades del mismo. Además es el responsable de gestionar la información de los demás usuarios. |
| Investigador | Representa el usuario del sistema que tiene la capacidad profesional de entender e interactuar con todas las funcionalidades del mismo excepto con la gestión de usuarios. |
| Usuario anónimo | Representa el usuario del sistema que solo tiene acceso a las funcionalidades públicas del mismo. |

Cada forma en que los actores usan un sistema se representa con un caso de uso, estos son fragmentos de funcionalidad que el sistema ofrece para aportar un resultado de valor observable para sus actores. Se utilizan para obtener información de cómo debe trabajar el sistema. Son una secuencia de acciones a llevar a cabo por el sistema como respuesta a los requerimientos funcionales. Describen el comportamiento del sistema mediante el intercambio acción y reacción. La identificación de los casos de uso es la guía del Ingeniero de Software que lleva adelante el desarrollo de un sistema. Se muestran a continuación los Casos de Uso del Sistema que se identificaron, basados en los requerimientos expuestos con anterioridad:

**Tabla 2:** CU Gestionar registro.

| **CU 1** | Gestionar registro. |
|---|---|
| **Actor** | Administrador, Investigador. |
| **Descripción** | Brinda la posibilidad de insertar un nuevo registro en la base de datos, así como eliminar uno ya existente. |
| **Referencia** | R1, R1.1, R1.1.1, R1.1.2, R1.2, R8. |
| **Prioridad** | Crítico |

**Tabla 3:** CU Realizar BH usando BLAST-N.

| | |
|---|---|
| **CU 2** | Realizar BH usando BLAST-N. |
| **Actor** | Administrador, Investigador, Usuario anónimo. |
| **Descripción** | Se realiza un alineamiento para buscar homología entre secuencias. |
| **Referencia** | R2, R2.1, R2.1.1, R2.1.2. |
| **Prioridad** | Crítico |

**Tabla 4**: CU Realizar AS usando CLUSTALW**.**

| | |
|---|---|
| **CU 3** | Realizar AS usando CLUSTALW. |
| **Actor** | Administrador, Investigador, Usuario anónimo |
| **Descripción** | Realizar un alineamiento múltiple con las secuencias seleccionadas. |
| **Referencia** | R2,R2.2,R2.2.1, R2.2.2, R2.2.3, R2.2.4 |
| **Prioridad** | Crítico |

**Tabla 5**: CU Realizar reconstrucción filogenética

| | |
|---|---|
| **CU 4** | Realizar reconstrucción filogenética. |
| **Actor** | Administrador, Investigador, Usuario anónimo |
| **Descripción** | Brinda la posibilidad de mostrar las distancias evolutivas de las secuencias seleccionadas , así como el árbol filogenético correspondiente. |
| **Referencia** | R3,R3.1,R3.2,R3.3 |
| **Prioridad** | Crítico |

**Tabla 6**: CU Gestionar cebadores.

| | |
|---|---|
| **CU 5** | Gestionar cebadores. |
| **Actor** | Administrador, Investigador. |
| **Descripción** | Brinda la posibilidad de insertar una nueva pareja de cebadores en la base de datos, así como modificar o eliminar una ya existente. |
| **Referencia** | R4,R4.1,R4.2,R4.3,R4.4,R8 |
| **Prioridad** | Secundario |

**Tabla 7**: CU Clasificación taxonómica a partir de la secuencia de MICL.

| | |
|---|---|
| **CU 6** | Clasificación taxonómica a partir de la secuencia de MICL. |
| **Actor** | Administrador, Investigador, Usuario anónimo. |
| **Descripción** | Propone una clasificación de una secuencia blanco o un conjunto de secuencias en un archivo especificado por el usuario. |
| **Referencia** | R5, R5.1, R5.2, R5.3, R5.4, R5.5 |
| **Prioridad** | Crítico |

**Tabla 8**: CU Buscar registros que poseen GFCE**.**

| | |
|---|---|
| **CU 7** | Buscar registros que poseen GFCE. |
| **Actor** | Administrador, Investigador, Usuario anónimo. |
| **Descripción** | Buscar todas las secuencias que se complementen con una pareja de cebadores especificada. |
| **Referencia** | R5, R5.1 |
| **Prioridad** | Crítico |

**Tabla 9**: CU Gestionar usuario.

| | |
|---|---|
| **CU 8** | Gestionar usuario. |
| **Actor** | Administrador. |
| **Descripción** | Brinda la posibilidad de insertar un nuevo usuario, así como modificar, eliminar y ver uno existente en la base de datos. |
| **Referencia** | R6,R6.1,R6.2,R6.3,R6.4,R8 |
| **Prioridad** | Secundario |

**Tabla 10**: CU Buscar registro

| | |
|---|---|
| **CU 9** | Buscar registro. |
| **Actor** | Administrador, Investigador, Usuario anónimo |
| **Descripción** | El actor puede buscar cualquier registro que sea de su interés, realizando una búsqueda por filtrado haciendo uso haciendo uso de palabras claves. |
| **Referencia** | R7 |
| **Prioridad** | Secundario |

**Tabla 11**: CU Autenticar usuario.

| | |
|---|---|
| **CU 10** | Autenticar usuario. |
| **Actor** | Administrador, Investigador |
| **Descripción** | El actor se autentica en la aplicación para tener acceso a todas las funcionalidades de la misma. |
| **Referencia** | R8 |
| **Prioridad** | Crítico |

**Tabla 12**: CU Cambiar contraseña.

| | |
|---|---|
| **CU 11** | Cambiar contraseña. |
| **Actor** | Administrador, Investigador. |
| **Descripción** | Brinda la posibilidad al actor de cambiar su contraseña de acceso a la aplicación. |
| **Referencia** | R7,R9 |
| **Prioridad** | Secundario |

## 2.4 Diagrama de caso de uso del sistema.

Un diagrama de caso de uso del sistema contiene actores, casos de uso y las relaciones que existen entre ellos. Se muestra a continuación el diagrama de casos de uso correspondiente al sistema propuesto:

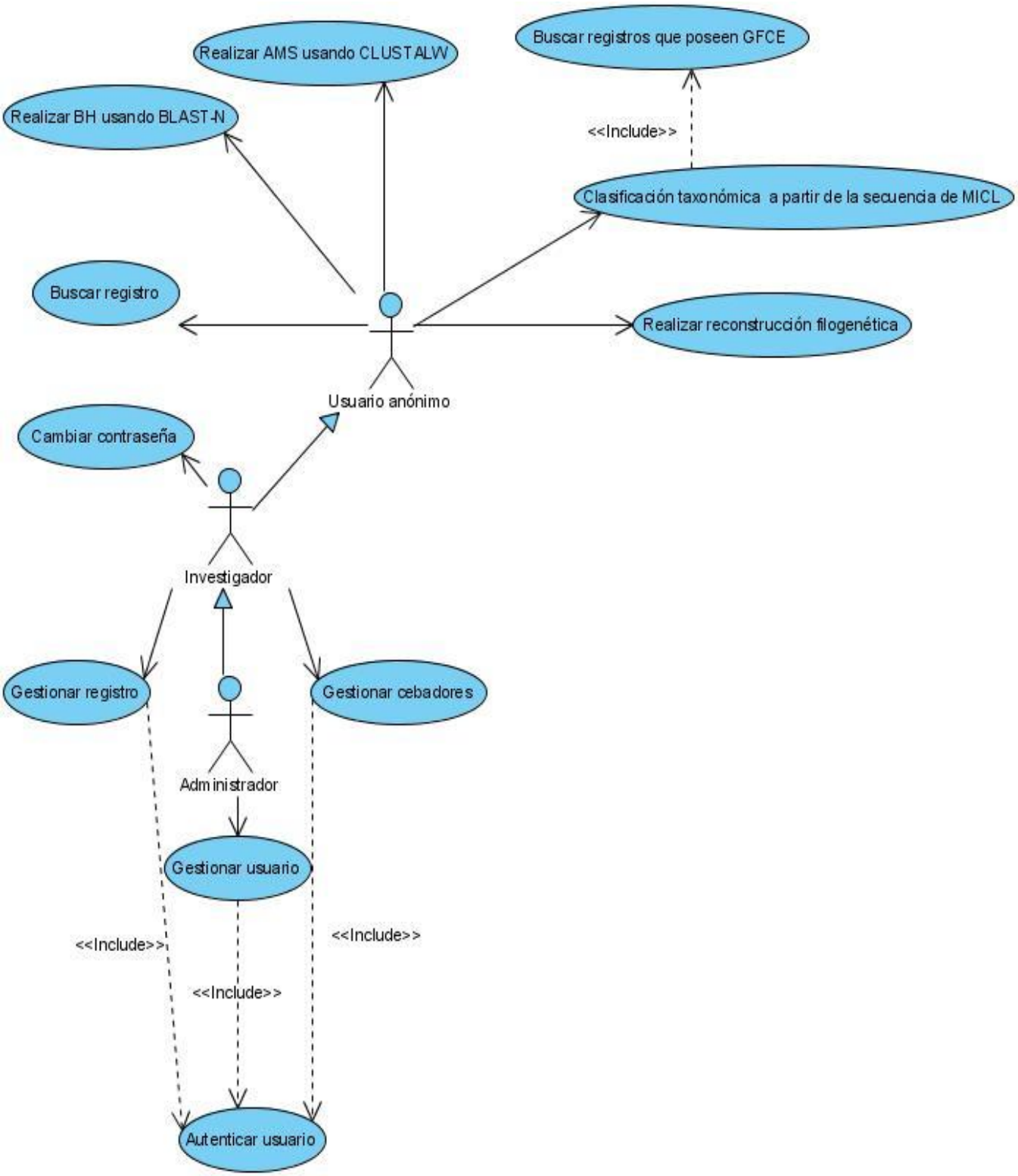



**Figura 3**: Diagrama de casos de uso del sistema.

## 2.5 Descripción detallada de los casos de uso

Para entender la funcionalidad asociada a cada caso de uso no es suficiente con la representación gráfica del diagrama de casos de uso, también se lleva a cabo la descripción textual de cada uno de ellos, las cuales se utilizan para detallar cada caso de uso, donde se describe el flujo de sucesos en una especificación precisa de la secuencia de acciones. Se incluye el comienzo, la terminación y la interacción entre los actores y lo que el sistema debe hacer. A continuación se muestran las descripciones textuales de algunos casos de uso, el Anexo 1 muestra las restantes.

### CUS Gestionar registro.

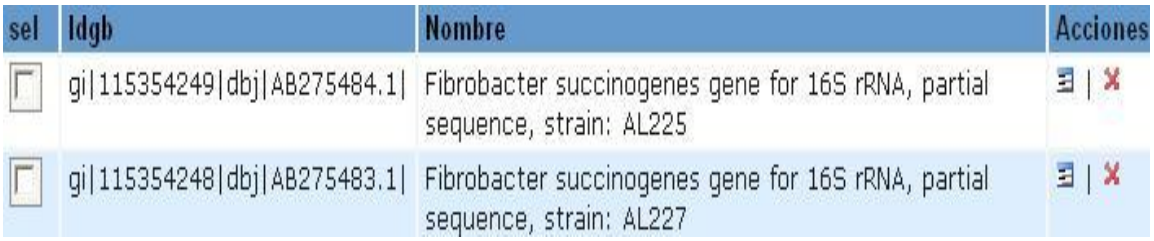
| **Nombre del Caso de Uso** | Gestionar registro |
|---|---|
| **Actores** | Administrador, Investigador. |
| **Propósito** | Insertar un registro en la BD para su estudio posterior así como eliminar los datos de uno ya existente. |
| **Resumen** | El caso de uso se inicia cuando el actor va a realizar alguna de las siguiente operaciones relacionadas con registro:<br>● Insertar un nuevo registro.<br>● Eliminar un registro.<br>El sistema le muestra la interfaz correspondiente. El caso de uso finaliza cuando se emite el resultado de la operación solicitada. |
| **Referencias** | R1, R1.1 , R1.1.1,R1.1.2,R1.2,R8 |
| **Precondiciones** | Que el actor se haya autenticado en la aplicación. |
| **Pos condiciones** | El sistema inserta un nuevo registro y elimina los datos de algún registro existente. |

| **Flujo normal de los eventos** | |
|---|---|
| **Acción del actor** | **Respuesta del Sistema** |
| 1. El actor quiere realizar una de las siguientes operaciones:<br>● Insertar un nuevo registro.<br>● Eliminar un registro. | 2.En dependencia de la operación solicitada:<br>● Si decide insertar un nuevo registro, ir a Sección **"Insertar nuevo registro"**.<br>● Si desea insertar registros desde fichero, ir a Sección **"Insertar desde fichero".**<br>● Si decide eliminar un registro, ir a la Sección **"Eliminar un registro".** |
| **Sección "Insertar nuevo registro"** | |
|  | |
| **Acción del actor** | **Respuesta del Sistema** |
| | 1.Muestra la interfaz correspondiente para la inserción de un nuevo registro con los siguientes campos:<br>- **"Reino"** es el campo que brinda la posibilidad de escoger el reino al que |

| | |
|---|---|
| | pertenece el microorganismo nuevo a insertar (Bacteria, Hongo, Protozoo, Archaea o Indefinido si no se sabe el reino al que pertenece.<br><br>- **"Familia"** es el campo que brinda la posibilidad de escoger la familia a la que pertenece el microorganismo a insertar.<br><br>- **"Descripción"** es el campo donde el actor introduce la descripción del microorganismo a insertar.<br><br>- **"Secuencia"** es el campo donde el actor introduce la secuencia de ADN del microorganismo. |
| 2. El actor proporciona los datos llenando todos los campos del formulario correspondiente. | 3. Verifica que los campos obligatorios contengan un valor.<br>• Reino.<br>• Familia<br>• Descripción<br>• Secuencia |
| | 4. Según el tipo de microorganismo lo inserta en su tabla correspondiente en la base de datos. |
| | |
| **Flujos alternos Sección "Insertar nuevo registro"** | |
| 2.1) El actor no desea proporcionar los datos para la inserción de un nuevo registro y sale de la sección. | |

| | |
|---|---|
| | 3.1) Si falta algún dato necesario para la inserción del nuevo registro, señala dicho(s) campos(s). |
| | |
| **Sección "Insertar desde fichero"** | |
|  | |
| **Acción del actor** | **Respuesta del Sistema** |
| | 1. Muestra la interfaz correspondiente para insertar registros desde fichero. |
| 2. El actor selecciona el archivo en formato FASTA y pulsa en el botón **"Enviar".** | 3. Realiza la inserción del archivo. |
| | |
| **Flujos alternos Sección "Insertar desde fichero"** | |
| 2.1. El actor no desea insertar el archivo y sale de la sección. | |
| | |
| **Sección "Eliminar registro"**. | |

| Acción del actor | Respuesta del Sistema |
|---|---|
| | 1. Se llama al caso de uso **"Buscar Registro"** |
| | 2. Muestra el resultado de la búsqueda en un listado. |
| 3. El actor da clic en la opción **"eliminar"** correspondiente al registro que desea remover de la Base de Datos. | 4. Se elimina el registro de la base de datos. |
| | |
| **Flujos alternos Sección "Eliminar registro"**. | |
| 3.1) El actor no desea eliminar el registro y sale de la sección. | |



| sel | Idgb | Nombre | Acciones |
|---|---|---|---|
| ☐ | gi\|115354249\|dbj\|AB275484.1\| | Fibrobacter succinogenes gene for 16S rRNA, partial sequence, strain: AL225 | |
| ☐ | gi\|115354248\|dbj\|AB275483.1\| | Fibrobacter succinogenes gene for 16S rRNA, partial sequence, strain: AL227 | |

**CUS Clasificación taxonómica a partir de la secuencia de MICL.**

| | |
|---|---|
| **Nombre del Caso de Uso** | Clasificación taxonómica a partir de la secuencia de MICL. |
| **Actores** | Administrador, Investigador, Usuario Anónimo. |

| | |
|---|---|
| **Propósito** | Clasificar los microorganismos que aún no se ha definido su clasificación taxonómica. |
| **Resumen** | El caso de uso se inicia cuando el actor decide realizar la clasificación de algún microorganismo.<br><br>El sistema le muestra la interfaz correspondiente y ejecuta las acciones necesarias. El caso de uso finaliza cuando se emite el resultado de la operación solicitada. |
| **Referencias** | R5, R5.1, R5.2, R5.3, R5.4, R5.5 |
| **Precondiciones** | - |
| **Pos condiciones** | El sistema realiza la clasificación y visualiza el resultado. |


**Flujo normal de los eventos**

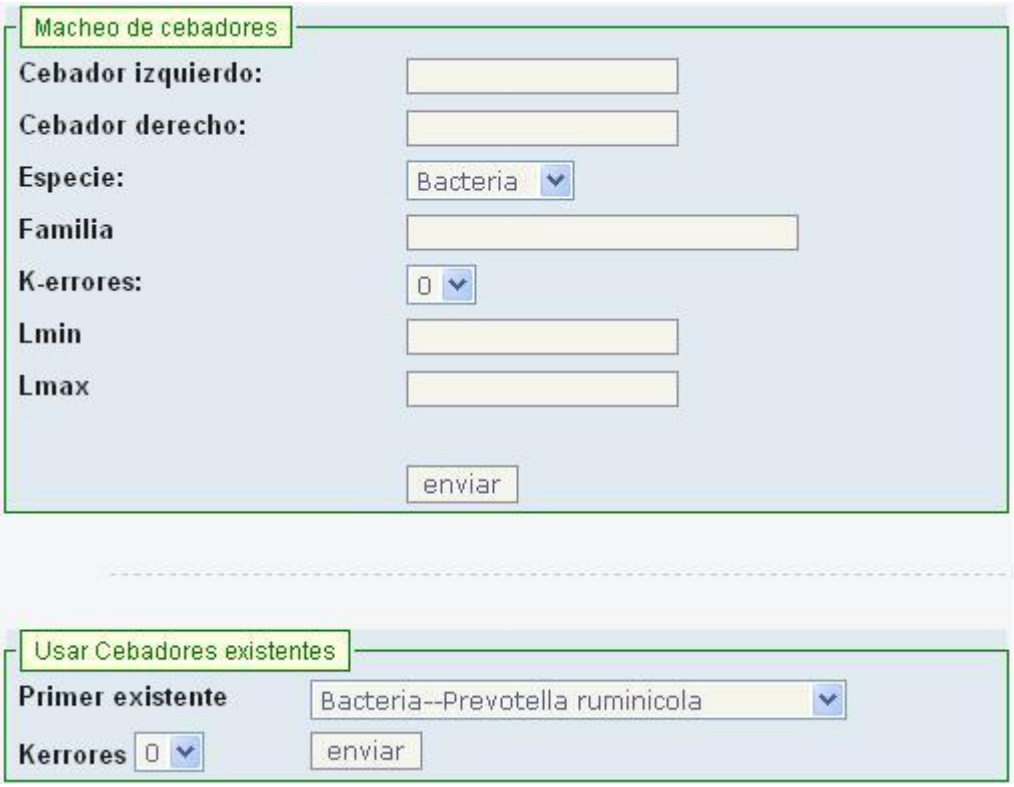
| Acción del actor | Respuesta del Sistema |
|---|---|
| 1. El actor desea clasificar taxonómicamente una o varias secuencias. | 2. Muestra la interfaz correspondiente con los siguientes campos:<br><br>-¨**Nombre del trabajo**¨ es el campo donde el actor introduce el nombre que decida ponerle a su trabajo.<br><br>-**"Secuencia":** Es el campo encargado de recibir la secuencia de ADN del microorganismo que se desea clasificar taxonómicamente en caso de ser uno solo.<br><br>-**"Desde archivo"**: Permite subir varias secuencias en formato FASTA para su futura clasificación.<br><br>Solamente se puede optar por una de estas opciones.<br><br>-**"K errores permitidos"** es el campo donde se especifica la cantidad de errores permitidos.<br><br>-¨**N Resultados significativos**¨ es el campo donde se especifica la cantidad de secuencias mas semejantes a la(s) secuencia(s) entrada(s).<br><br>-¨**Formato de salida**¨ es el campo donde se especifica el formato en que se mostrará el resultado. |

| | |
|---|---|
| | -¨**Valor de expectación (E)** ¨ es el campo donde se introduce el valor de expectación. |
| 3. El actor provee los datos y pulsa en el botón **"Clasificar".** | 4. Verifica que el campo obligatorio tenga un valor:<br>**-"Secuencia"** |
| | 5. Realiza la clasificación. |
| | 6. Visualiza el resultado de la clasificación. |
| | |
| **Flujos alternos "Clasificación taxonómica a partir de la secuencia de MICL".** | |
| 1.1) El actor no desea clasificar y sale de la sección. | |
| 3.1) El actor no provee los datos y sale de la sección. | 4.1) Si falta algún dato necesario para la realización de la clasificación, señala dicho(s) campos(s). |

**CUS Buscar registros que poseen GFCE.**

| | |
|---|---|
| **Nombre del Caso de Uso** | Buscar registros que poseen GFCE. |
| **Actores** | Administrador, Investigador, Usuario Anónimo. |
| **Propósito** | Buscar todas las secuencias que se complementen con una pareja de cebadores dados o introducidos por |

| | |
|---|---|
| | el actor. |
| **Resumen** | El caso de uso se inicia cuando el actor decide buscar todas las secuencias que se complementen con dos cebadores dados.<br><br>El sistema le muestra la interfaz correspondiente y ejecuta las acciones necesarias. El caso de uso finaliza cuando se emite el resultado de la operación solicitada. |
| **Referencias** | R5, R5.1 |
| **Precondiciones** | - |
| **Pos condiciones** | El sistema encuentra las secuencias que se complementan con la pareja de cebadores especificada. |




| **Flujo normal de los eventos** | |
|---|---|
| **Acción del actor** | **Respuesta del Sistema** |
| 1. El actor quiere realizar una de las siguientes operaciones:<br>● Realizar el macheo con una pareja de cebadores.<br>● Realizar el macheo con una pareja de cebadores existente en la base de datos. | 2. En dependencia de la operación solicitada:<br>● Si decide realizar el macheo con una pareja de cebadores introducidos por el actor, ir a Sección **"RMP introduciendo cebadores"**.<br>● Si decide realizar el macheo con una pareja de cebadores existente en la base de datos, ir a Sección **"RMP existentes"**. |

| **Sección "RMP introduciendo cebadores"** | |
|---|---|
| **Acción del actor** | **Respuesta del Sistema** |
| | 1.El sistema muestra la interfaz correspondiente con los siguientes campos:<br><br>-**"Cebador izquierdo"** es el campo donde se especifica el primer forward.<br><br>-**"Cebador derecho"** es el campo donde se especifica el primer reverse.<br><br>-**"K errores"** es el campo donde se especifica la cantidad de errores permitidos.<br><br>-**"Lmin"** y **"Lmax"** es el campo donde se especifican valores enteros positivos definidos por el usuario.<br><br>**-"Especie"** es el campo donde se especifica la especie a la que pertenece esa pareja de cebadores.<br><br>**-"Familia"** es el campo donde se especifica la familia a la que pertenece esa pareja de cebadores. |
| 2. El actor provee los datos y pulsa en el botón **"enviar".** | 3.Verifica que los campos obligatorios tengan un valor:<br><br>-**" Cebador izquierdo"**<br><br>-**" Cebador derecho"** |

| | |
|---|---|
| | -**"Lmin"** y **"Lmax"** |
| | 4. Realiza la búsqueda de las secuencias que corresponden al criterio de búsqueda |
| | 5. Muestra un listado con todas las secuencias que se emparejaron para esa pareja de cebadores Dando la opción de descargar, realizar alineamiento múltiple o análisis filogenético de las secuencias seleccionadas. |
| 6. Si el actor desea ver en detalle la secuencia pulsa en el botón **"ver"** y si desea eliminar alguna pulsa en el botón **"eliminar".** | |
| 7. Si el usuario desea realizar una de las opciones brindadas en el listado, escoge cual y hace clic en el botón **"Aceptar"**. | 8. Muestra el resultado. |
| **Flujos alternos Sección "RMP introduciendo cebadores"** | |
| 2.1) El actor no desea hacer la búsqueda y sale de la sección. | |
| | 3.1) Si falta algún dato necesario para la búsqueda de las secuencias, señala dicho(s) campos(s). |
| 6.1) Si el actor no desea ver la secuencia o eliminarla sale de la | |

| | |
|---|---|
| sección. | |
| 7.1) Si el actor no desea realizar las opciones brindadas sale de la sección. | |
| **Sección "RMP existentes"** | |
| **Acción del actor** | **Respuesta del Sistema** |
| | 1.Muestra la interfaz correspondiente con los siguientes campos:<br><br>**-"Pareja de cebadores existentes"** donde estarán todas las parejas de cebadores existentes en la base datos para realizar el macheo.<br><br>**-"K"** que será la cantidad de errores permitidos en ese emparejamiento. |
| 2. El actor llena los campos y pulsa en el botón **"enviar".** | 3. Muestra un listado con todas las secuencias que emparejaron para esa pareja de cebadores, dando la posibilidad de ver en detalle la secuencia o de eliminarla en caso que el actor desee; además de descargar, realizar alineamiento múltiple o análisis filogenético de las secuencias seleccionadas. |
| 4. Si el actor desea ver en detalle la secuencia pulsa en el botón **"ver"** y si desea eliminar alguna | |

| | |
|---|---|
| pulsa en el botón **"eliminar".** | |
| 5. Si el usuario desea realizar una de las opciones brindadas en el listado, escoge cual y hace clic en el botón **"Aceptar"**. | 6. Muestra el resultado. |
| **Flujos alternos Sección "RMP existentes"** | |
| 2.1) El actor no desea realizar el macheo y sale de la sección. | |
| 4.1) Si el actor no desea ver la secuencia o eliminarla sale de la sección. | |
| 5.1) Si el actor no desea realizar las opciones brindadas sale de la sección. | |

A modo de conclusión en este capítulo se brindó una definición de los requisitos que deberá cumplir el sistema. A través de estos se han detallado las especificaciones que han hecho los investigadores como parte del flujo de trabajo de levantamiento de requisitos. Se definió el diagrama de casos de uso del sistema lo cual permitió que se representaran detalladamente las funcionalidades requeridas, lográndose una modelación más clara de la interacción del sistema y el usuario. De esta forma se concretó el funcionamiento del sistema que se pretende desarrollar y quedaron sentadas las bases para las siguientes fases del proceso de diseño e implementación de la aplicación.

# Capítulo 3: Diseño del sistema.

## 3.1 Objetivos del diseño.

El diseño tiene como principales objetivos comprender detalladamente los requisitos funcionales y no funcionales, sistemas operativos, tecnologías de distribución, restricciones relacionadas con el lenguaje de programación, tecnologías de interfaz de usuario. Además, el diseño crea un punto de partida para las actividades de implementación que siguen. La realización física de los casos de uso del sistema se ve descrita mediante el modelo de diseño. Este se centra en cómo influyen los requisitos funcionales y no funcionales en el sistema en cuestión. Los métodos de una clase del diseño tienen correspondencia directa con el correspondiente método en la implementación de las clases.

## 3.2 Principios de diseño

"UML posee una extensión para el modelado de aplicaciones web, propuesta por Conallen, dicha extensión es usada para el diseño de las clases. Los estereotipos que usa esta extensión son" (36).


***<<Página servidor>>*** Representa la página web que tiene código que se ejecuta en el servidor. Este código interactúa con recursos en el mismo. Las operaciones representan las funciones del código y los atributos las variables visibles dentro del alcance de la página.


***<<Página cliente>>*** Una instancia de página cliente es una página web, con formato HTML; mezcla de datos, presentación y lógica. Son interpretadas por el navegador. Cada página cliente solo puede ser construida por una página servidor.


***<<Formulario>>*** Grupo de elementos de entrada que son parte de una página cliente. Se relaciona directamente con la etiqueta de igual nombre del HTML. Sus atributos son los elementos de entrada del formulario (Text Field, Text Area, Button, Label, Radio Button, Radio Group, Select, Check Box y Hidden Fields).

## 3.3 Arquitectura del Sistema.

"La arquitectura de software es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución." (37). También es denominada arquitectura lógica, consiste en un conjunto de patrones y abstracciones coherentes que proporcionan el marco de referencia necesario para guiar la construcción del software.

### 3.3.1 Patrones de arquitectura y diseño.

Para el desarrollo de la aplicación se ha decidido utilizar el framework Spring. Este introduce el patrón arquitectónico Modelo-Vista-Controlador (MVC), el cual presenta algunas similitudes con otros frameworks para web que existen en el mercado, pero son algunas características que lo vuelven único ya que hace una clara división entre controladores, modelos de JavaBeans y vistas, está basado en interfaces y es bastante flexible. Además provee interceptores al igual que controladores, se hace uso de JSP como tecnología View y los controladores son configurados de la misma manera que los demás objetos en Spring, a través de inversiones de control (IoC). Este patrón divide una aplicación interactiva en tres áreas: procesamiento, salida y entrada. Para esto, utiliza las siguientes abstracciones: (28)

**Modelo:** encapsula los datos y las funcionalidades. Este es independiente de cualquier representación de salida y/o comportamiento de entrada.

**Vista:** muestra la información al usuario. Pueden existir múltiples vistas del modelo. Cada vista tiene asociado un componente controlador.

**Controlador:** reciben las entradas, usualmente como eventos que codifican los movimientos o pulsación de botones del ratón o pulsaciones de teclas. Los eventos son traducidos a solicitudes de servicio ("service requests") para el modelo o la vista.

Se definen las clases dominio (Modelo) para que no tengan acoplamiento ni visibilidad directa respecto a las clases ventana (Vista) y para que los datos de la aplicación y de la funcionalidad se conserven en las clases de dominio, no en las de ventana. Definen las clases manejadores (Controlador) para que procesen los eventos (peticiones al sistema) y re direccionen a las clases dominio y ventana tanto el procesamiento como la visualización de resultados respectivamente.

Este patrón influye en el buen desarrollo del software ya que nos brinda un menor acoplamiento, una mayor cohesión, mayor escalabilidad, más claridad de diseño, facilita el mantenimiento y las vistas proveen mayor flexibilidad y agilidad (22).

En la implementación se hace necesario dar soporte a una mínima dependencia y a un aumento en la reutilización, para lograr esto se hace vital el uso del patrón de diseño **Bajo acoplamiento**. El acoplamiento mide qué tan fuerte está una clase conectada con otras (es decir, cuántas clases conoce y necesita). Una clase con bajo (o débil) acoplamiento no depende de muchas otras clases. Una clase con alto (o fuerte) acoplamiento recurre a muchas otras clases. Este tipo de clase no es conveniente, pues: cambios en las clases relacionadas ocasionan cambios en la clase local; son más difíciles de entender y de reutilizar. Con la finalidad de mantener la complejidad dentro de límites manejables entre las clases, muchas de las cuales se desean que manejen responsabilidades específicas, adoptamos el uso del patrón **Alta cohesión**, es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme. Una clase con baja cohesión hace muchas cosas no afines o un trabajo excesivo. No conviene este tipo de clases ya que son difíciles de comprender, de reutilizar, de conservar, son delicadas y las afectan constantemente los cambios.

Las clases con baja cohesión a menudo representan un alto grado de abstracción o han asumido responsabilidades que deberían haber delegado a otros objetos. La solución es asignar una responsabilidad de modo que la cohesión siga siendo alta. Los beneficios aportados por este patrón se hacen necesarios en la implementación, por esta razón se adopta su correcto uso. Otro patrón necesario es el **Patrón experto**, que responde a la interrogante: ¿cuál es el principio fundamental en virtud del cual se asignan las responsabilidades en el diseño orientado a objetos?, asigna responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad. Al mismo tiempo el **Patrón creador,** es la solución al problema: ¿quién debería ser el responsable de la creación de una nueva instancia de alguna clase?, guía la asignación de responsabilidades relacionadas con la creación de objetos. La intención básica del patrón es encontrar un creador que necesite conectarse al objeto creado en alguna situación. Logrando así mayor mantenimiento y reutilización (22).

Con el uso de Spring, se hace uso de los patrones definidos dentro de este framework. Estos son: instancia única, puente, objeto compuesto, proxy y cadena de responsabilidad. **Instancia única** garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso

global a dicha instancia. (Es un patrón GOF creacional utilizado para crear todos las instancias únicas de las clases controladoras, servicios, daos contenidas en el *Bean factory* de Spring).**Puente** desacopla una abstracción de su implementación. (Es un patrón GOF estructural utilizado para separar interfaces e implementación de las clases que prestan servicios y DAOs (Data access object/clases de acceso a datos). **Objeto compuesto** permite tratar objetos compuestos como si de un simple se tratase (es un patrón GOF estructural para anidar vistas, realizando abstracción pues al parecer es un solo objeto y realmente están compuestas por diferentes partes definidas en los titles). **Proxy** mantiene un representante de un objeto (es un patrón GOF estructural que se utiliza en los servicios para envolver los métodos en un concepto transaccional, crea al método como una transacción).Por último **cadena de responsabilidad** permite establecer la línea que deben llevar los mensajes para que los objetos realicen la tarea indicada (es un patrón GOF de comportamiento utilizado en la línea de mensajes reflejada en mensajes de controladores a los servicios y de los servicios a los DAO).

Otro de los patrones altamente eficientes e implementados como parte de la arquitectura de Spring es el patrón **Inyección de Dependencia.** Este es un patrón de diseño que establece que un objeto no es el responsable de setear sus dependencias, sino que las mismas deben ser seteadas por un tercero, o sea se inyectan objetos a una clase en lugar de ser la propia clase quien cree el objeto. Es una forma para mitigar la proliferación de dependencias y así fomentar el bajo acoplamiento entre los componentes, y además tiene la intención de reducir la cantidad de código de infraestructura que se debe escribir.

Para el diseño de la vista, se hace necesario reutilizar código, de tal forma que no se repita el código usado en alguna vista. Para esto se decidió usar el patrón arquitectónico **Vista de Plantilla.** La idea básica de la plantilla de vista es incrustar marcadores en una página HTML cuando esta es escrita. Cuando la página se utiliza para solicitar un servicio, los marcadores se sustituirán por los resultados obtenidos de acuerdo al servicio que brinda la página como por ejemplo, el resultado de una consulta obtenida en una base de datos.

**Fachada** es un patrón de diseño que sirve para proveer de una interfaz unificada sencilla que haga de intermediaria entre un cliente y una interfaz o grupo de interfaces más complejas. Utilizado para reducir la dependencia entre clases, ya que ofrece un punto de acceso al resto de las clases, si estas cambian o se sustituyen por otras solo hay que actualizar la clase Fachada sin que el cambio afecte a las aplicaciones cliente. Este patrón no oculta las clases sino que ofrece una forma más sencilla de acceder a ellas, en los casos en que se requiere se puede acceder directamente a ellas.

En la persistencia el uso de hibernate integrado a Spring plantea la necesidad de usar el patrón de diseño DAO. Este plantea como solución utilizar un Data Access Object (DAO) para abstraer y encapsular todos los accesos a la fuente de datos. El DAO maneja la conexión con la fuente de datos para obtener y almacenar datos.

### 3.3.2 Diagramas de clases del diseño.

Con el uso del framework Spring se introduce el patrón arquitectónico Modelo-Vista-Controlador (MVC), por lo que los diagramas de clases del diseño correspondiente reflejan cómo se realizan las interacciones entre los principales elementos y clases del sistema, cumpliendo con el patrón arquitectónico definido. Todas las peticiones realizadas por los usuarios en las páginas clientes son atendidas por el controlador frontal (Dispatcher Servlet), quien delega a otro componente de Spring el procesamiento de la solicitud; que es quien determina cuál será el controlador que recibirá la petición del usuario. El controlador definido se encarga de recuperar la información necesaria mediante comunicaciones que establece con las diferentes clases como las que prestan servicios, las clases de acceso a datos y las clases entidades contenidas en el modelo. Luego este notifica al controlador frontal para que construya las páginas clientes con los datos requeridos.

A continuación se muestran algunos diagramas de clases del diseño, para ver los restantes consultar el Anexo 2.

## CU Clasificación taxonómica  a partir de la secuencia de MICL

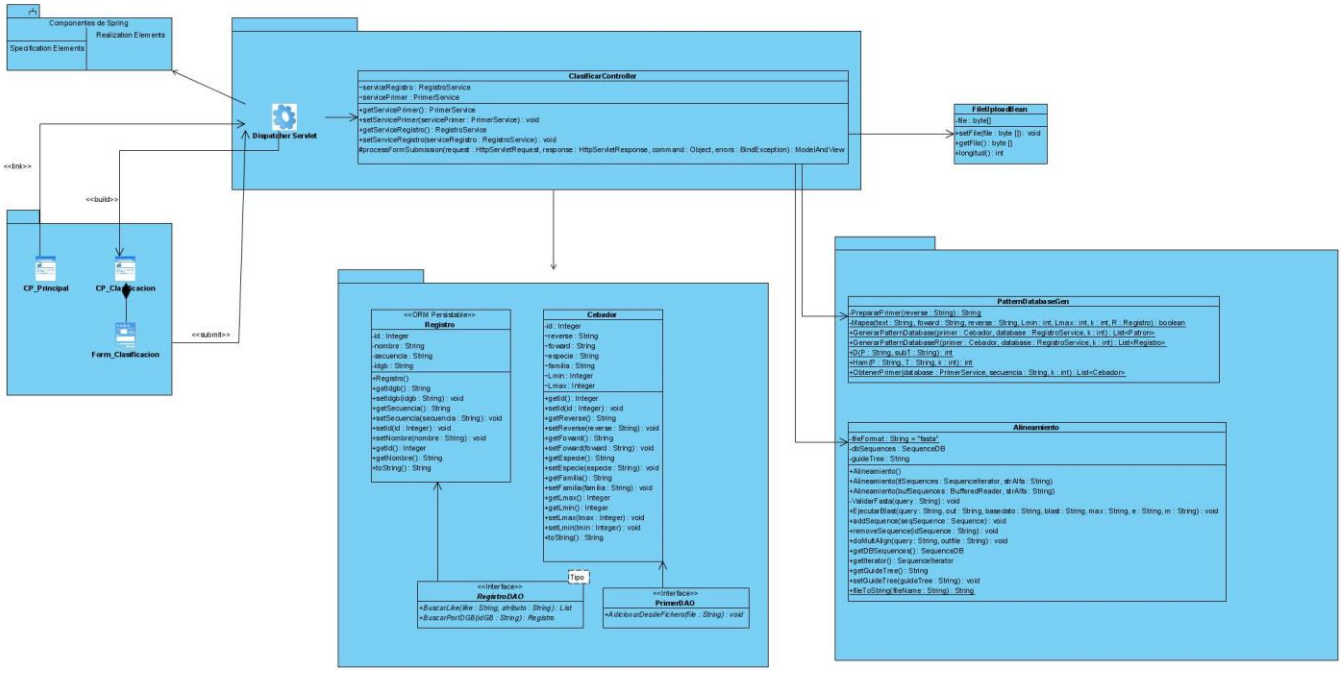



**Figura 4**: Diagrama de clases del diseño CU Clasificación taxonómica  a partir de la secuencia de MICL.

## CU  Buscar registros que poseen GFCE

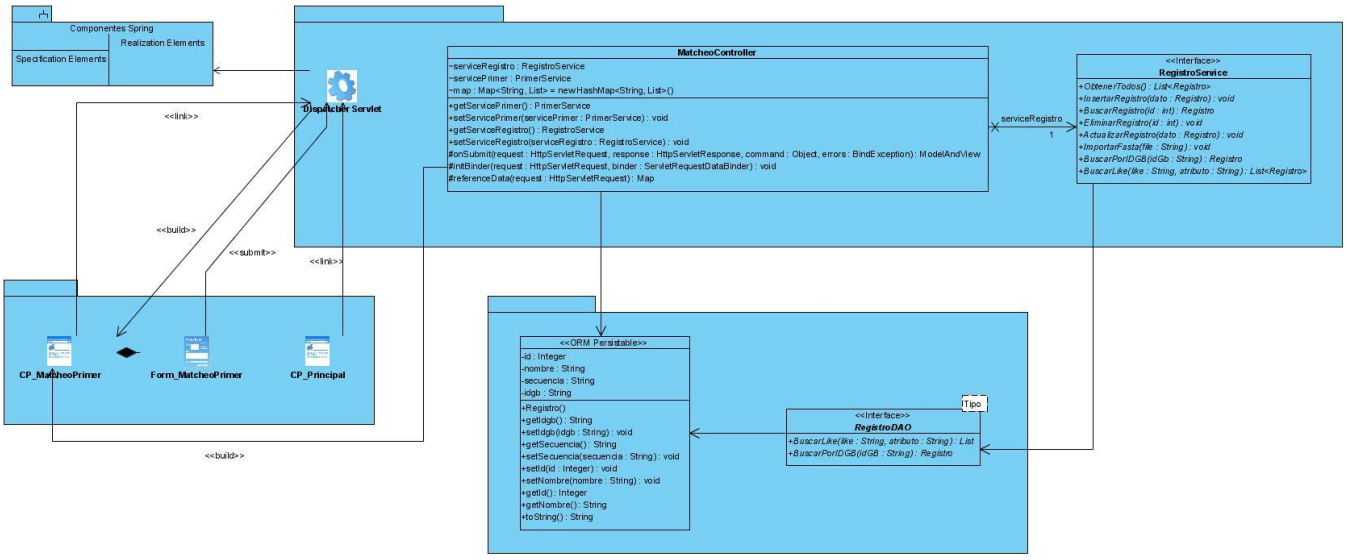



**Figura 5**: Diagrama de clases del diseño CU Buscar registros que poseen GFCE.

**CU Gestionar registro.**

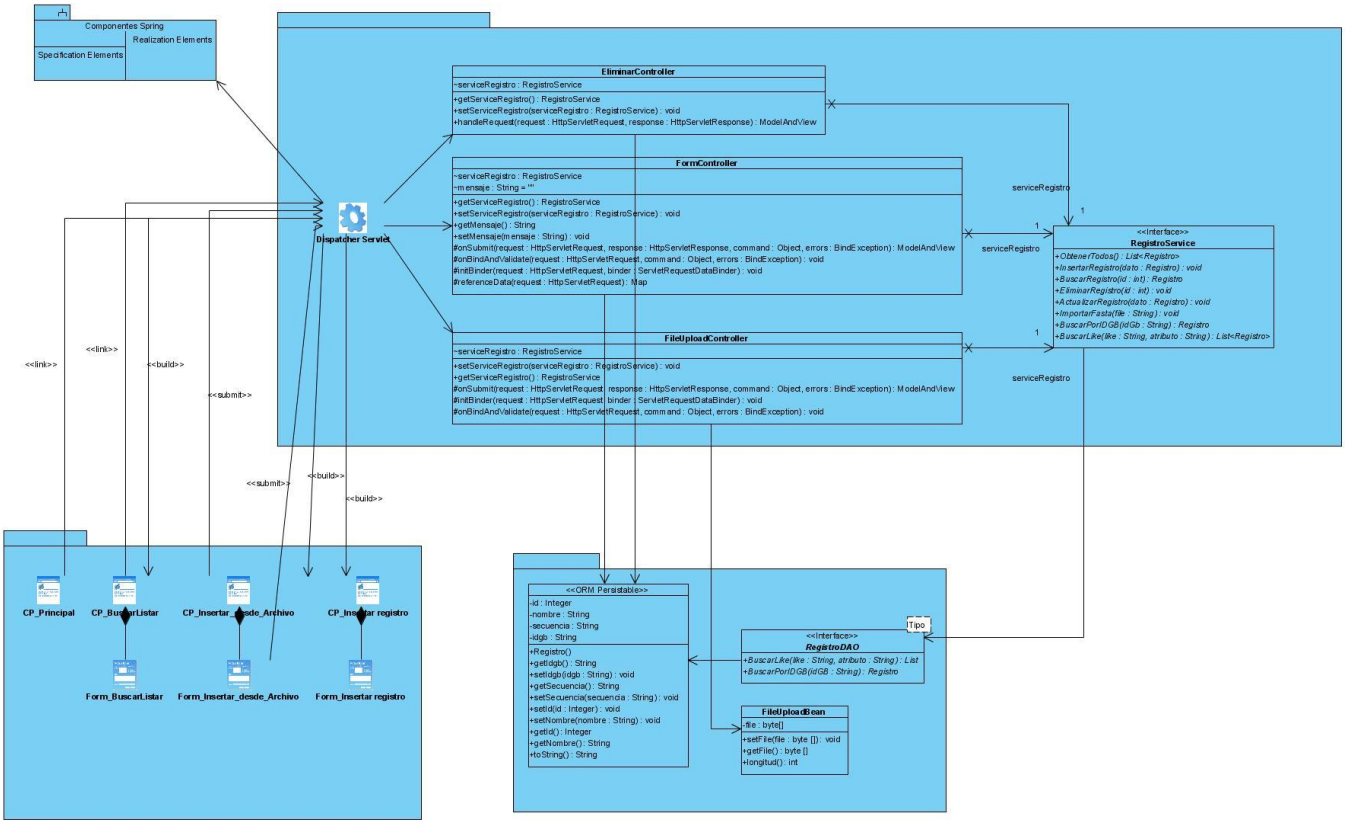



**Figura 6**: Diagrama de clases del diseño CU Gestionar registro.

### 3.3.3 Modelo de despliegue

Un diagrama de despliegue es un grafo de nodos unidos por conexiones de comunicación. Muestra las relaciones físicas entre los componentes hardware y software en el sistema final, es decir, la configuración de los elementos de procesamiento en tiempo de ejecución y los componentes software (procesos y objetos que se ejecutan en ellos), en fin, son fundamentalmente diagramas de clases que se ocupan de modelar los nodos de un sistema.

La vista de despliegue es la distribución de los componentes a través de los nodos. Dicha vista se especifica en el capítulo 4, donde se actualiza el diagrama de despliegue con dicha distribución. A continuación se muestra el diagrama de despliegue, el cual está estructurado de la siguiente manera:

Se cuenta con un servidor de aplicaciones, las estaciones clientes se conectan a esta a través del protocolo HTTP. Para el almacenamiento de los datos de la aplicación se utiliza un servidor de Base

de Datos representado como un nodo y para lograr la conexión del sistema con la base de datos se utiliza TCP / IP como protocolo de comunicación. En este caso es sumamente recomendado contar con un servidor de base de datos independiente debido a que en el servidor de aplicaciones se realizan cálculos que consumen gran cantidad de recursos.

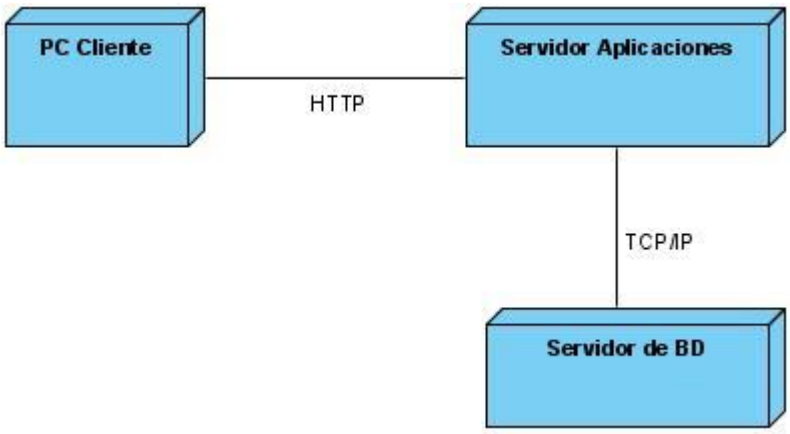



**Figura 7**: Diagrama de despliegue.

### 3.4 Prototipos de interfaz de usuario.

Los prototipos de interfaz de usuario ayudan a comprender y especificar las interacciones entre el sistema y los usuarios. Son una representación de la funcionalidad contenida en los casos de uso donde permiten que el usuario verifique que el sistema va a satisfacer sus necesidades. De esta manera se valida que se esté cumpliendo con los requerimientos. Para ver los prototipos no funcionales de los Casos de Uso del Sistema remitirse al Anexo 4.

### 3.5 Mapa de navegación.

Un mapa de navegación es una forma ordenada de distribuir tanto la información como las labores que esto implica, adecuándose a las necesidades especificas de las personas que lo utilizarán. Es una representación de la aplicación a través de una secuencia de interfaces. Para la construcción del mapa de navegación se utilizó UML, la navegabilidad se estructuró de la siguiente manera:

A partir de la página principal se podrá acceder a las demás páginas de la aplicación (Anexo 5) mediante un menú disponible y de estas a todas las restantes. Las acciones a realizar por el usuario son las siguientes:

**1. Gestión**

1.1. Insertar Registro. Permite adicionar un nuevo microorganismo a la base de datos.

1.2. Insertar desde Archivo. Permitiendo subir un archivo en formato FASTA para adicionar todas las secuencias contenidas en él a la Base de Datos.

1.3. Listado. Esta página permite listar todos los Registros y hacer una búsqueda avanzada de los mismos.

1.4. Insertar cebadores. Esta página permite que el investigador ingrese una nueva pareja de cebadores, los cuáles serán insertados en la Base de Datos.

1.5. Editar Cebadores. Página en la cual se muestra un listado de Cebadores, de la cual se pueden editar, eliminar y visualizar cada pareja de cebadores.

**2. Análisis de secuencias.**

2.1. BLAST. Esta página permite realizar alineamientos BLAST

2.2. CLUSTAL. Herramienta para realizar alineamientos múltiples.

2.3. Emparejamiento de cebadores. Página para buscar todas las secuencias que encajen con una pareja de cebadores determinada.

2.4. Clasificación. Permite realizar la clasificación taxonómica de una o más secuencias.

2.5. Análisis Filogenético. En esta página se puede visualizar el árbol filogenético de un conjunto de secuencias determinadas por el usuario.

**3. Administración**

3.1. Registrar Usuarios. Permite adicionar nuevos usuarios, asignándoles un Rol determinado.

3.2. Editar Usuarios. Se muestra un listado de los usuarios, del cual se pueden eliminar, editar y visualizar cada uno de ellos.

3.3. Cambiar la contraseña de acceso a la aplicación.

### 3.6 Diagrama de secuencia.

Un diagrama de secuencia tiene como objetivo fundamental encontrar una secuencia de interacciones entre objetos, detalladas y ordenadas en el tiempo. El diagrama de secuencia muestra en los diferentes escenarios de un caso de uso, los eventos generados por actores externos, y los eventos internos del sistema. A continuación se presentan los diagramas de secuencia de algunos casos de usos y para ver los restantes consultar el Anexo 3.

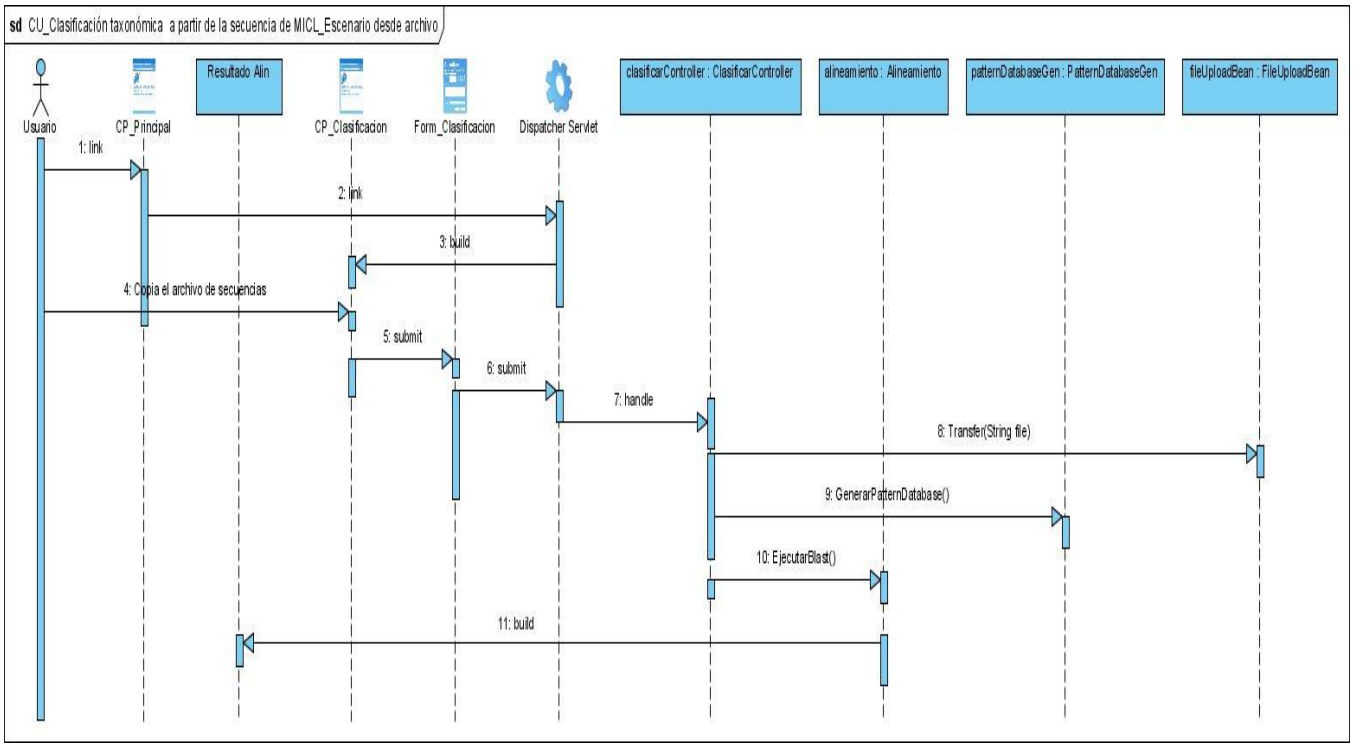



**Figura 8**: Diagrama de secuencia del CU Clasificación taxonómica a partir de la secuencia de MICL Escenario desde archivo.

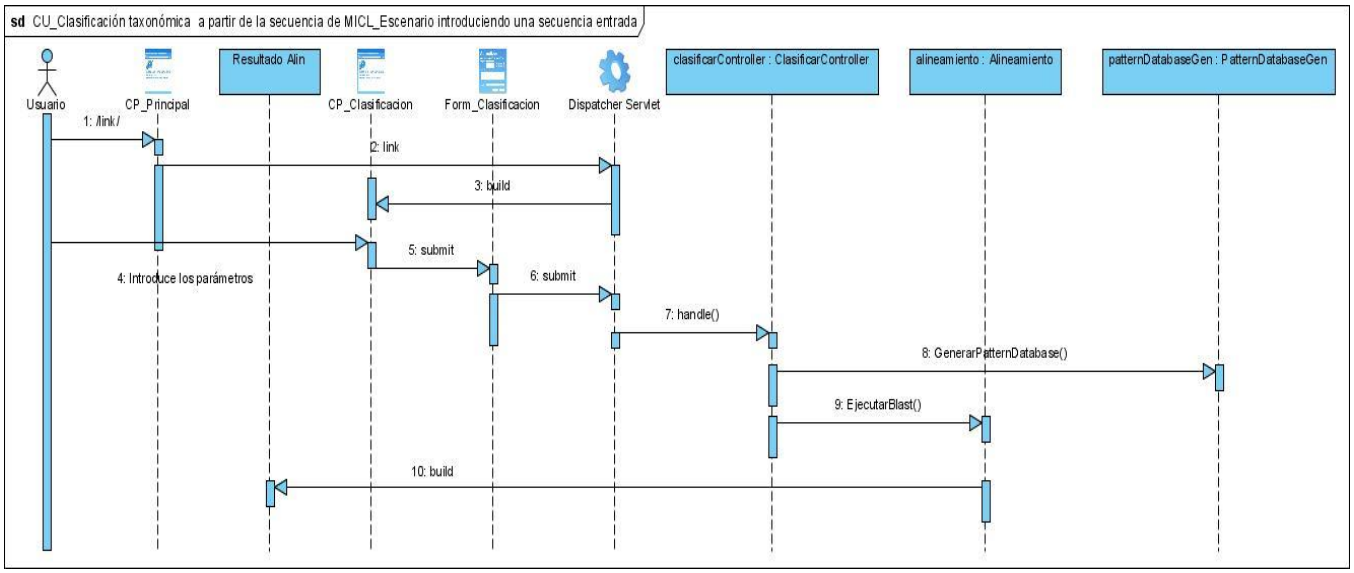



**Figura 9**: Diagrama de secuencia del CU Clasificación taxonómica a partir de la secuencia de MICL Escenario introduciendo una secuencia.

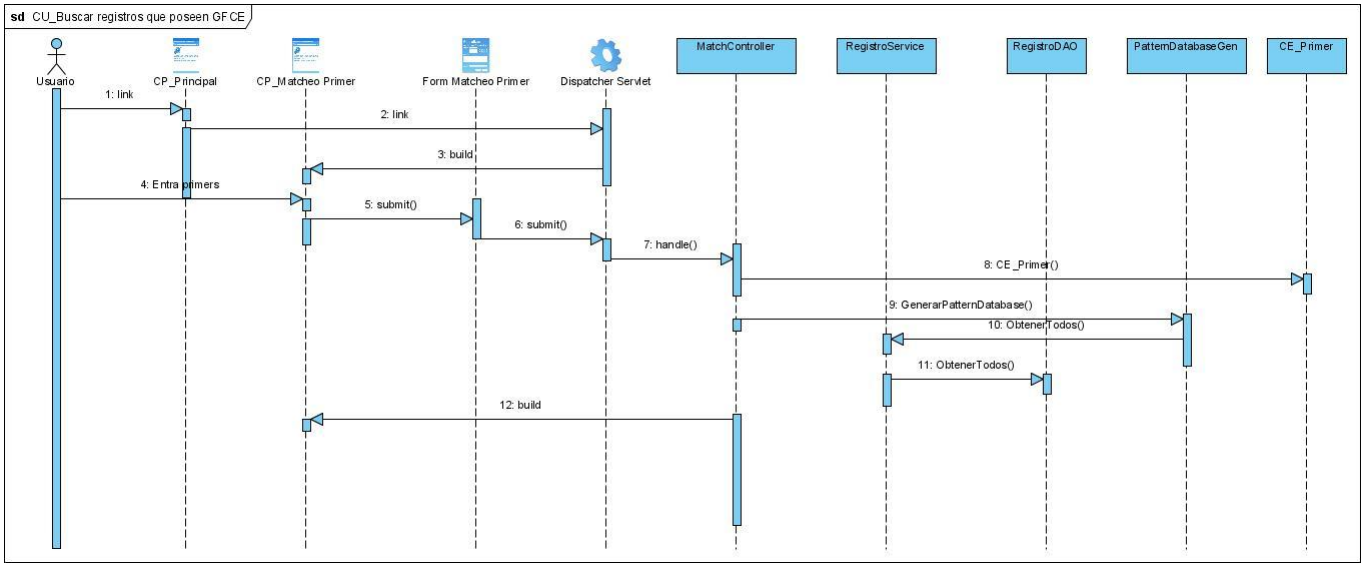



**Figura 10**: Diagrama de secuencia del CU Buscar registros que poseen GFCE.

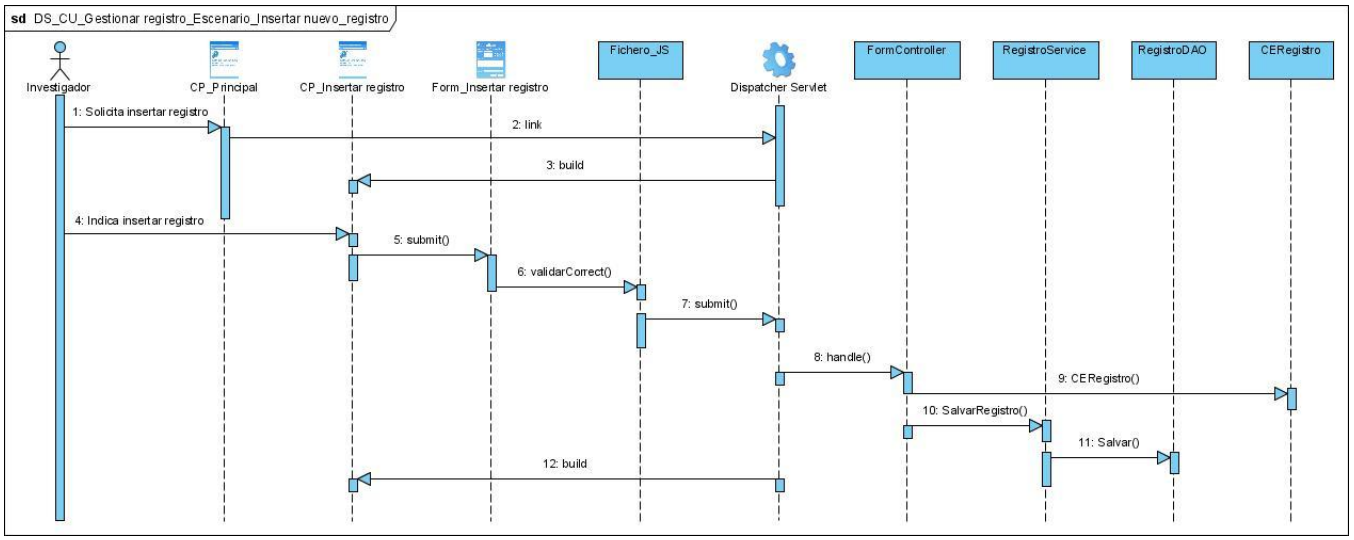



**Figura 11**: Diagrama de secuencia de CU Gestionar registro Escenario Insertar nuevo registro.

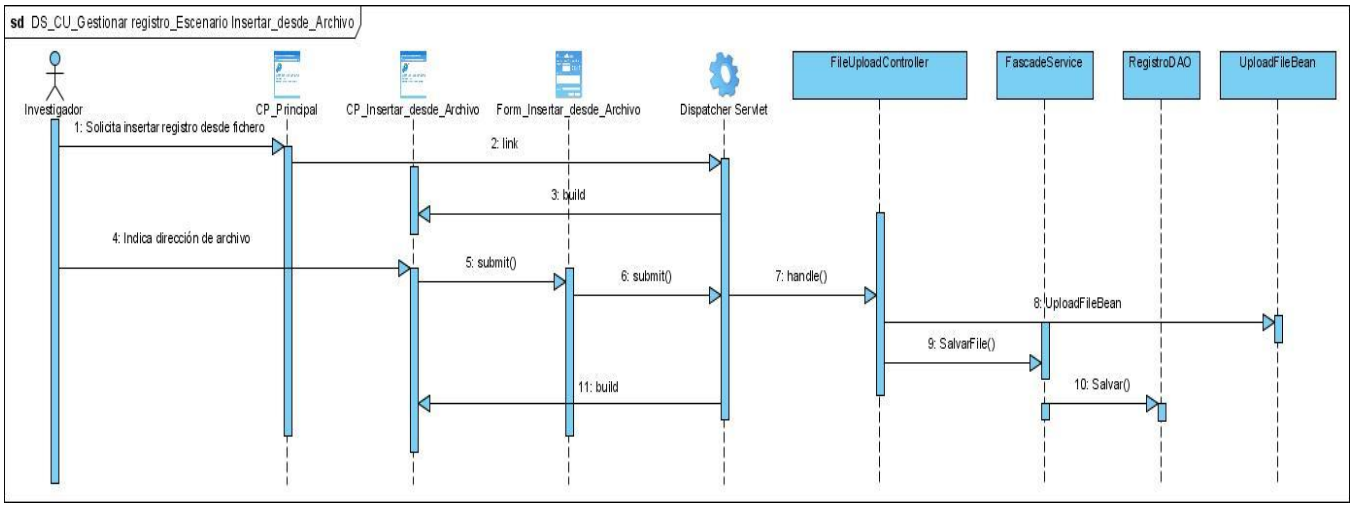



**Figura 12**: Diagrama de secuencia de CU Gestionar registro Escenario Insertar registro desde archivo.

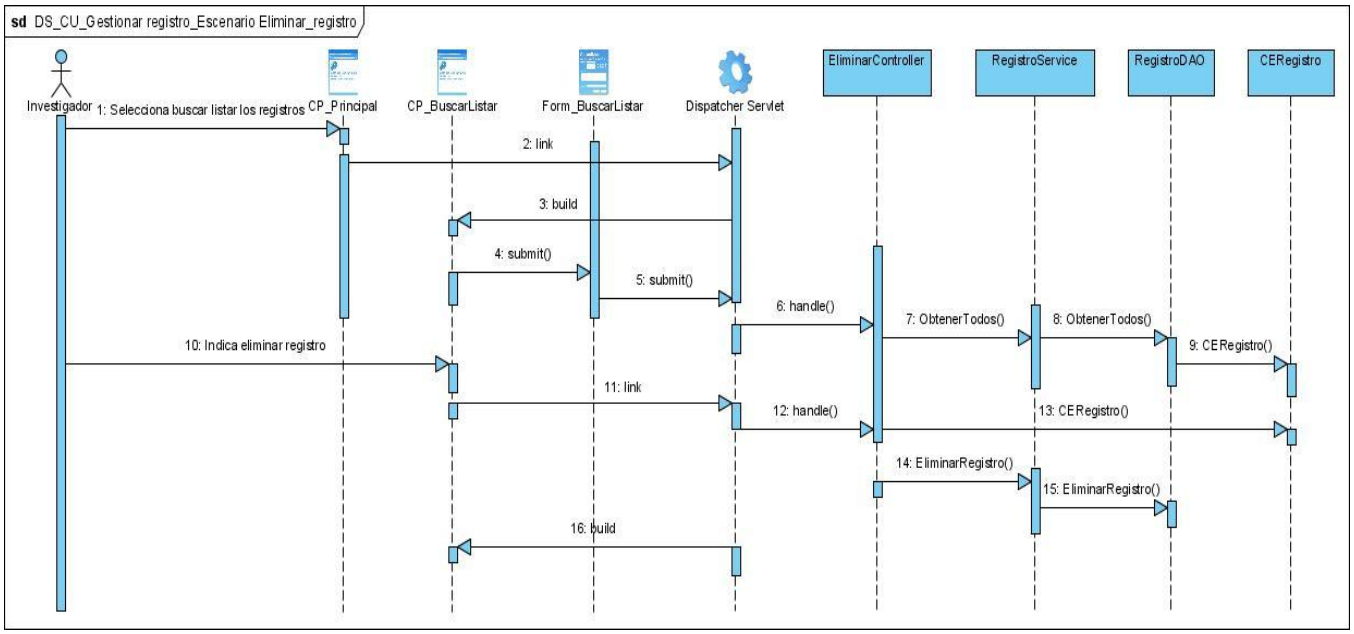



**Figura 13**: Diagrama de secuencia de CU Gestionar registro  Escenario Eliminar registro.

## 3.7 Diseño de la base de datos.

### 3.7.1 Diagrama de clases persistentes.

Para realizar el diseño de la base de datos (BD) de la aplicación se confeccionó el diagrama de clases persistentes y el modelo de datos. La persistencia es la capacidad de un objeto de mantener su valor en el espacio y en el tiempo. Las clases persistentes se definen para conocer la información real representada en las tablas de la BD. 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. El diagrama de clases se utiliza para modelar la estructura lógica de la BD, con clases representando tablas, y atributos de clases representando columnas.

A continuación se presenta el diagrama de clases persistentes que está compuesto por dichas clases y las correspondientes relaciones entre ellas y más adelante el modelo físico de datos que es una representación de las tablas de la base de datos y sus relaciones.

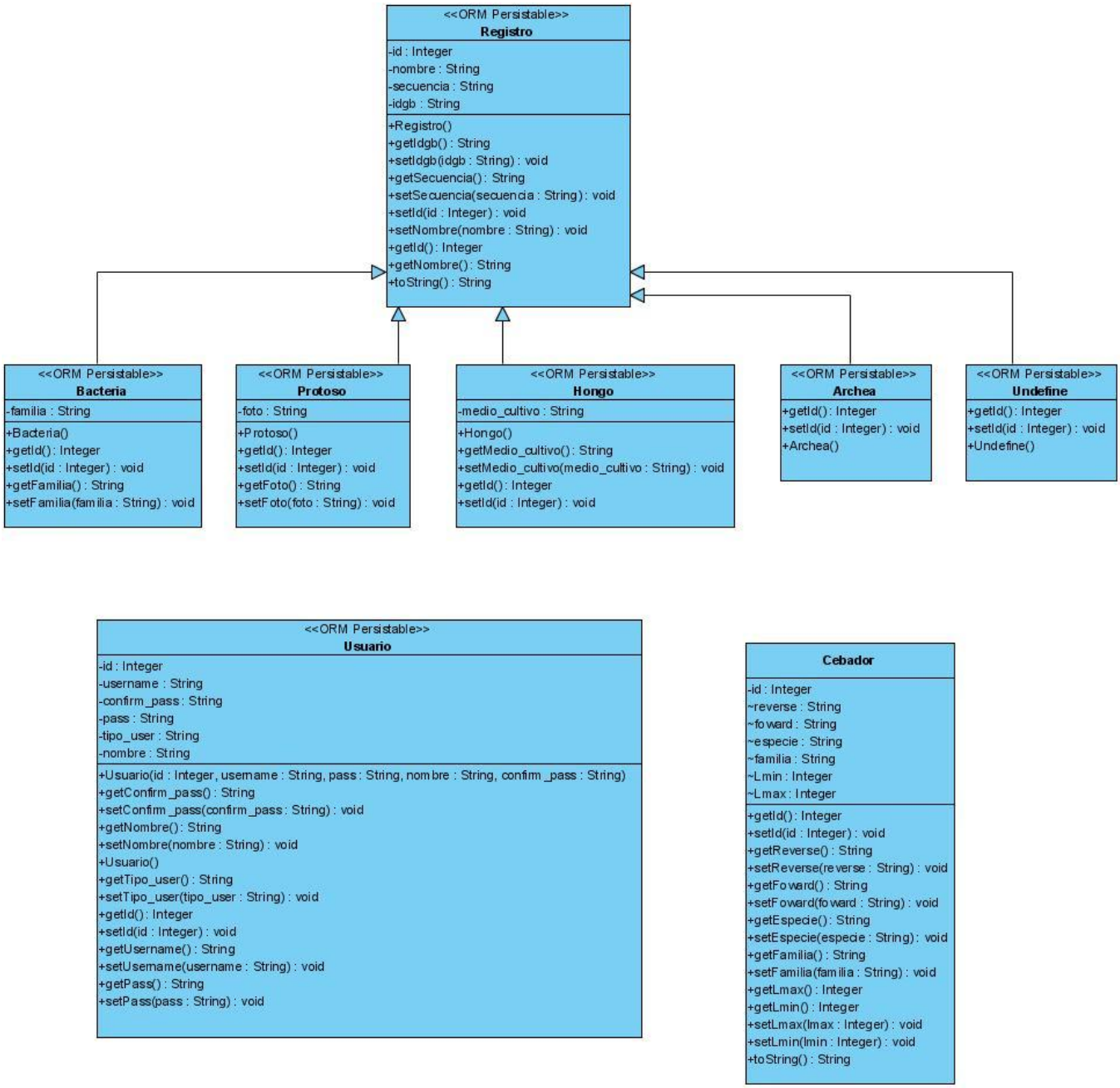



**Figura 14**: Diagrama de clases persistentes.

### 3.7.2 Modelo de datos.

Una de las características fundamentales de los sistemas de bases de datos es que proporcionan cierto nivel de abstracción de datos, al ocultar las características sobre el almacenamiento físico que la mayoría de usuarios no necesita conocer. Los modelos de datos son el instrumento principal para

ofrecer dicha abstracción. Un modelo de datos es un conjunto de conceptos que sirven para describir la estructura de una base de datos: los datos, las relaciones entre los registros y las restricciones que deben cumplirse sobre estos. A continuación se muestra el modelo de datos correspondiente a nuestra aplicación:

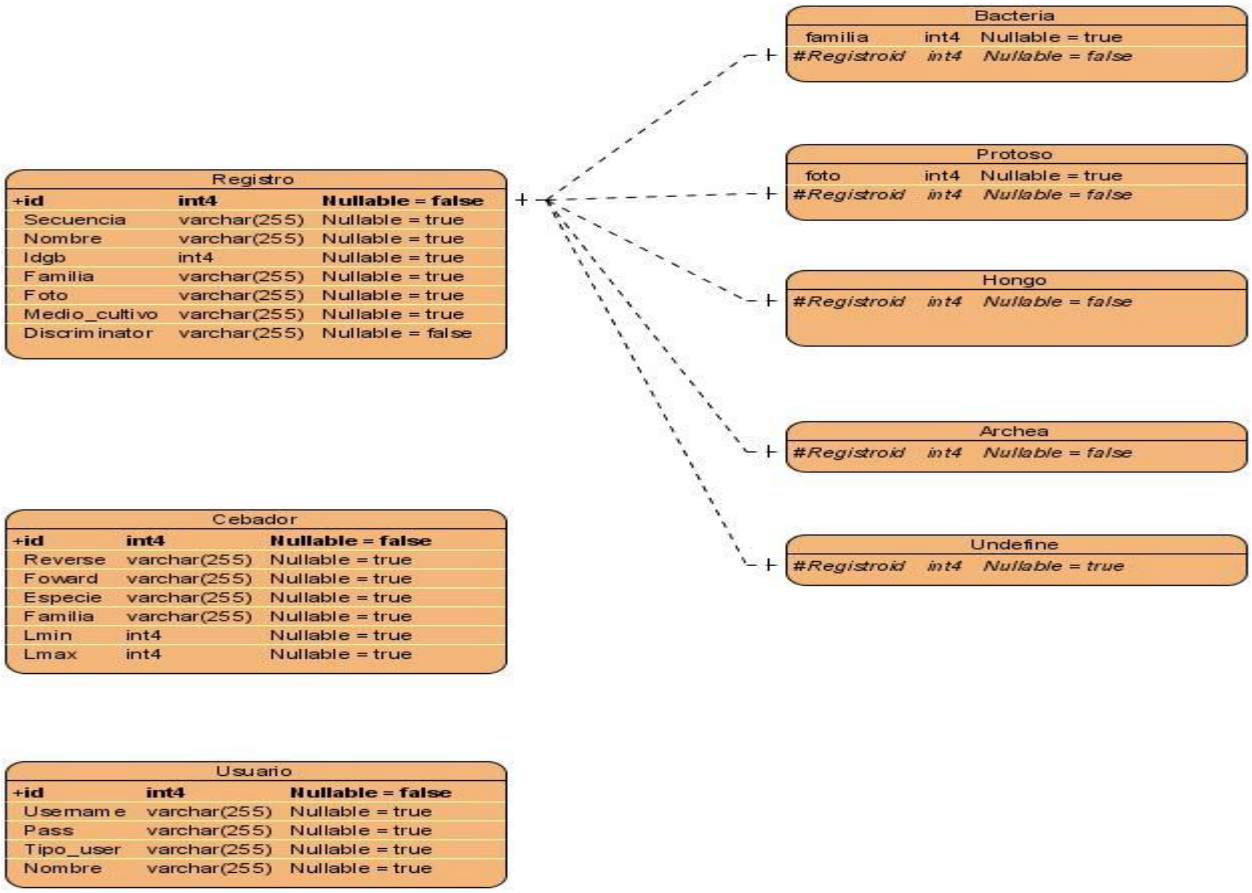



**Figura 15**: Modelo de datos.

A modo de conclusión en este capítulo se desarrolló la totalidad de los diagramas de clases del diseño para los 11 casos de uso identificados y los respectivos diagramas de secuencia, mostrándose solo algunos en el documento. Se confeccionó el mapa de navegación para garantizar una adecuada navegabilidad por parte del usuario y se obtuvo como una referencia a la arquitectura del sistema el modelo de despliegue. Se mostró el diagrama de clases persistentes y el modelo de datos.

# Capítulo 4: Implementación del sistema.

## 4.1 Propósito de la implementación.

La implementación es el centro durante las iteraciones de construcción, aunque también se lleva a cabo en la fase de elaboración para crear la línea base ejecutable de la arquitectura, y durante la transición para tratar los defectos tardíos. Se planifican las integraciones de sistema necesarias en cada iteración siguiendo un enfoque incremental, lo que da lugar a un sistema que se implementa en una sucesión de pasos pequeños y manejables. Se distribuye el sistema asignando componentes ejecutables a nodos en el diagrama de despliegue basándose en las clases activas encontradas durante el diseño. Se Implementan las clases y subsistemas encontrados durante el diseño, fundamentalmente las que se implementan como componentes de ficheros que contienen código fuente. Se prueban los componentes individualmente antes de integrarlos posteriormente enlazándolos en uno o más componentes ejecutables, antes de ser enviados para ser integrados y llevar a cabo las comprobaciones del sistema.

## 4.2 Modelo de Implementación

El modelo de implementación describe cómo los elementos de diseño se implementan en términos de componentes, describe como se organizan los componentes de acuerdo con los mecanismos de estructuración disponibles en el entorno de implementación y en el lenguaje o lenguajes de programación utilizados y como dependen los componentes unos de otros.

### 4.2.1 Diagrama de Componentes

Un Componente es la “Parte modular de un sistema, desplegable y reemplazable que encapsula implementación y un conjunto de interfaces y proporciona la realización de los mismos. Un componente típicamente contiene clases y puede ser implementado por uno o más artefactos (ficheros ejecutables, binarios).” (20)

En el diagrama de componentes se muestran las organizaciones y dependencias lógicas entre los diferentes componentes de software que interactúan en el sistema. Desde el punto de vista del diagrama de componentes se tienen en consideración los requisitos relacionados con la facilidad de

desarrollo, la gestión del software, la reutilización, y las restricciones impuestas por los lenguajes de programación y las herramientas utilizadas en el desarrollo.

En los diagramas de componentes realizados, se muestra su organización según el patrón arquitectónico Modelo-Vista-Controlador que utiliza Spring como paradigma en su organización interna, el cual está formado por tres partes fundamentales: el modelo, la vista y el controlador.

En el modelo, el diseño de los componentes cumple con el estándar propuesto por el framework Hibernate (Anexo 6, Fig. 30). Cada una de los archivos de configuración hbm.xml del paquete map tiene relación uno a uno con las clases del paquete beans y a su vez ambos paquetes se relacionan con los paquetes hibernate y la BDMicroorganismos.

En el controlador están los paquetes Controllers y Services (Anexo 6, Fig. 32). La relación entre los componentes, es de uno a muchos. En el paquete Services cada una de las clases implementa su interfaz correspondiente.

En la Vista la relación de los componentes está dada según la filosofía del framework Spring utilizando su ventaja de integración con componentes de otros frameworks (Anexo 6, Fig. 31). En este caso se integra a la vista las etiquetas de Struts, permitiendo así utilizar un solo diseño para todas las vistas y reutilizar al máximo el código empleado en la vista. La estructura del diseño es debido al uso del patrón Composite View lo que permite darle fácil mantenimiento y hacer cambios en la interfaz siempre y cuando sea necesario o solicite el cliente.

A continuación se muestran los diagramas de componentes de los casos de uso que se les viene dando seguimiento desde el diseño. En el Anexo 6 se muestran los restantes diagramas.

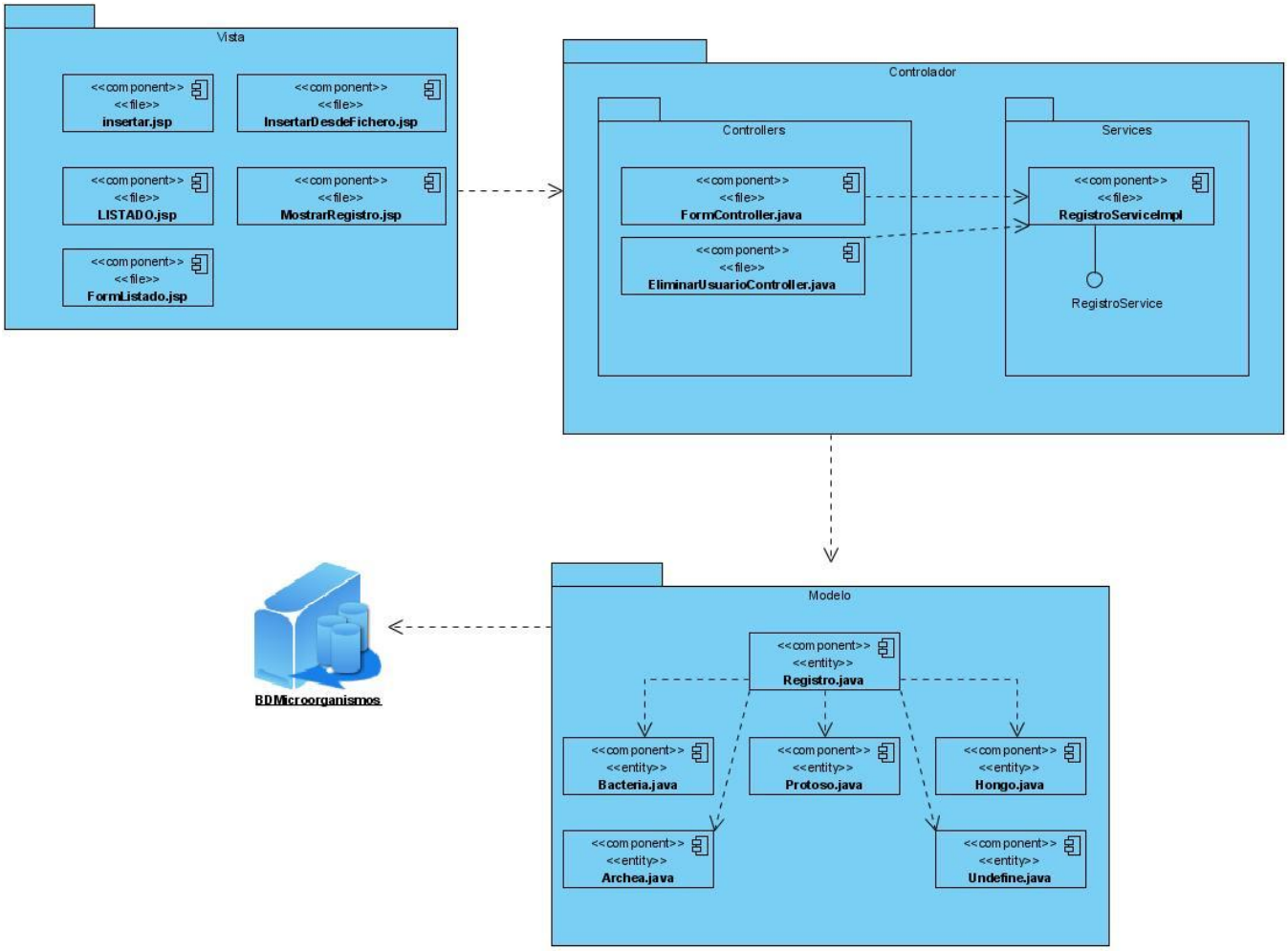



**Figura 16:** Diagrama de componentes: CU Gestionar registro.

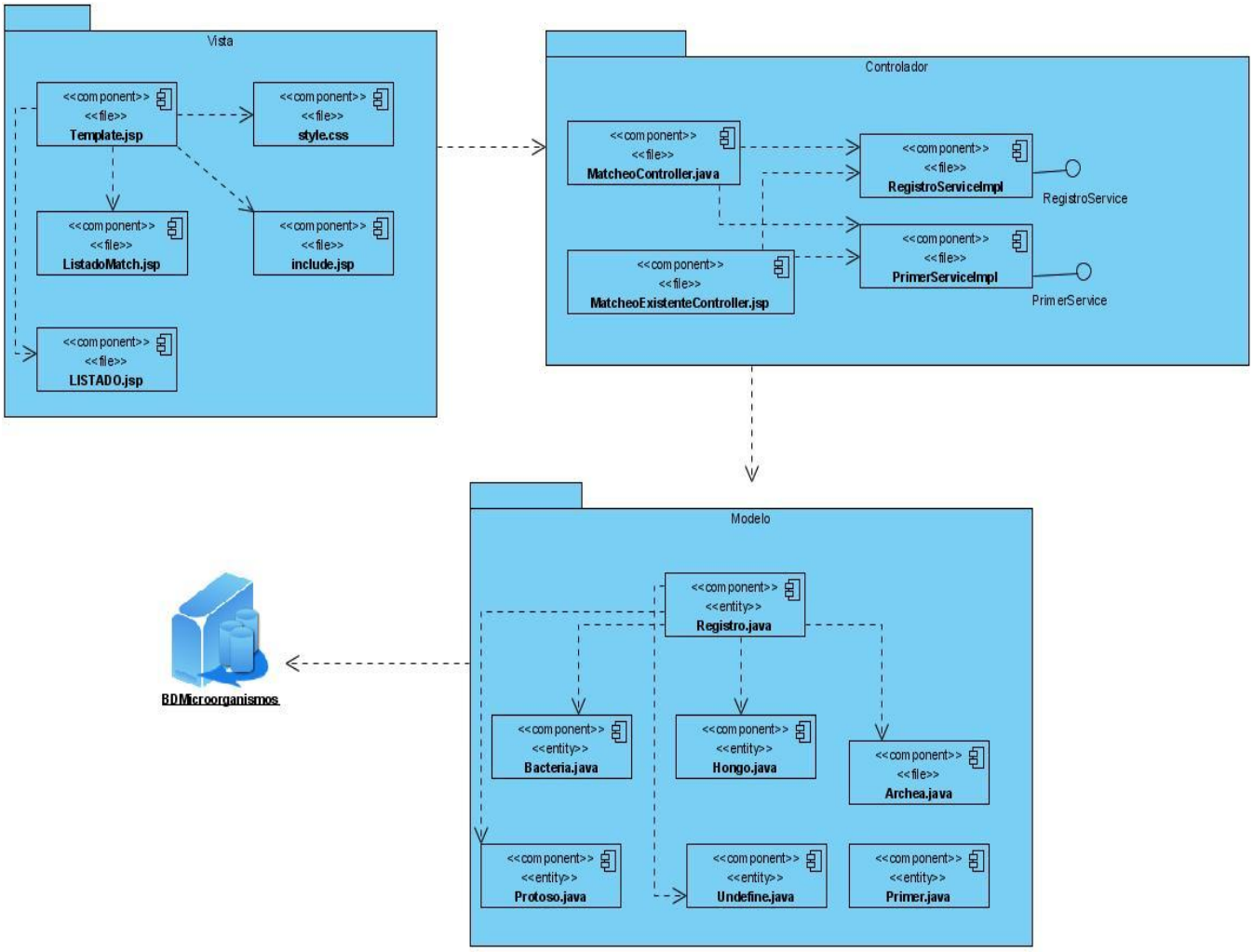



**Figura 17:** Diagrama de componentes: CU Buscar registros que poseen GFCE.

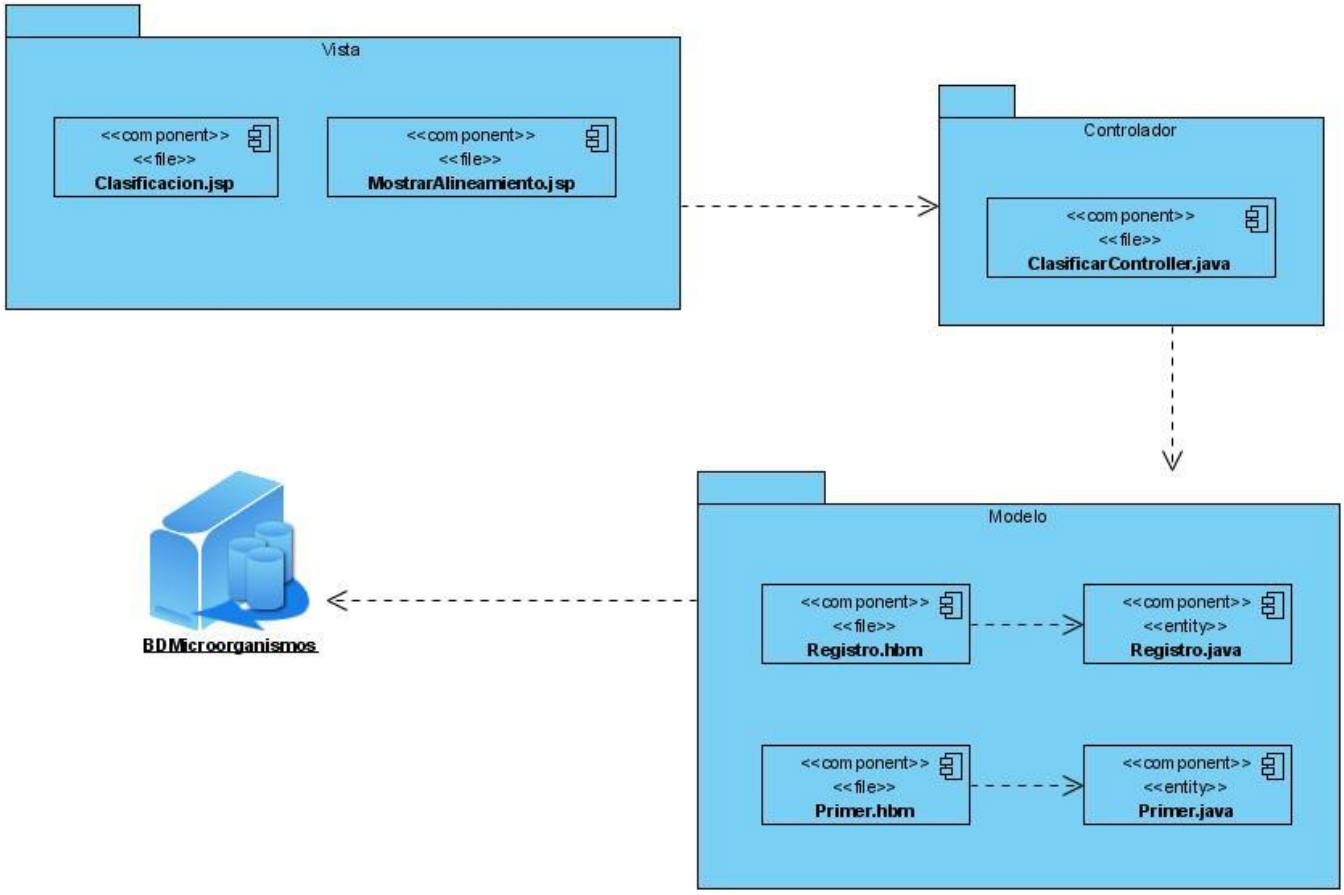



**Figura 18:** Diagrama de componentes: CU Clasificación taxonómica a partir de la secuencia de MICL.

## 4.3 Vista de despliegue

La vista de despliegue ilustra la distribución del procesamiento entre los distintos equipos que conforman la solución propuesta, incluyendo los servicios y procesos de base. Existe una traza directa del modelo de implementación, puesto que cada componente físico debe estar almacenado en un nodo. En el nodo que representa el servidor de aplicaciones se encuentran todos los componentes que son implementados en la aplicación así como los ejecutables y ficheros que se usan.

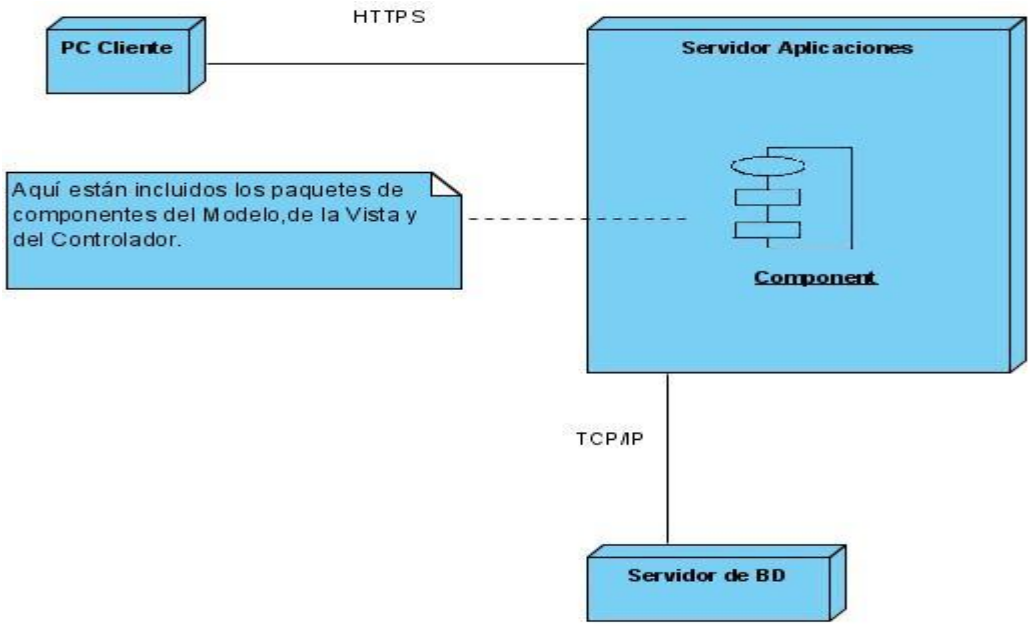



**Figura 19**: Vista de despliegue.

## 4.4 Descripción de la implementación del sistema

### 4.4.1 Procesamiento de formularios y validación

Una de las increíbles facilidades que se aprovechan de Spring es la de llenar, recibir y procesar formularios, así como desplegar errores y confirmaciones. Para el usuario todo este proceso es totalmente transparente ya que el formulario que este llenando tendrá el mismo formato y los mismos elementos que cualquier otro formulario normal.

### 4.4.2 Análisis de la seguridad del sistema implementado

Uno de los aspectos que toda aplicación debe considerar es la seguridad, entendiendo como tal la necesidad de saber que el usuario es quien dice ser (autenticación), y permitirle acceso sólo a aquellos recursos necesarios (autorización). La seguridad del sistema se logra llevando un control de cada usuario, los mismos se agrupan por roles: investigador y administrador, además de las credenciales que son privilegios de seguridad agrupados bajo un nombre y que permiten organizar la seguridad. Cuando un usuario trata de acceder a una acción restringida según sus credenciales; si el usuario está autenticado y tiene las credenciales apropiadas, entonces la acción se ejecuta; si el usuario no está

autenticado, es re direccionado a la acción de login; si el usuario está autenticado, pero no posee las credenciales apropiadas, será informado que está intentando acceder a una zona restringida.

La autenticación, sinónimo de seguridad, permite limitar el acceso a secciones de la aplicación. Los usuarios investigador y administrador tendrán que autenticarse para acceder a las áreas asignadas de acorde a su rol. De esta manera se asegura la aplicación para diferentes tipos de usuarios y se administran los privilegios. Si el usuario no se autentica el sistema lo tratará como usuario anónimo. En la aplicación mediante el uso del framework Acegi Security System se proporciona la funcionalidad necesaria para adoptar mecanismos de seguridad utilizando características de programación orientada a aspectos, de forma transparente para el desarrollador, sin necesidad de desarrollar código, utilizando para ello el soporte prestado por el framework Spring.

### 4.4.3 Código fuente de las principales clases

### Clase Alineamiento.

Esta clase se encarga de hacer los alineamientos usando el ClustalW y el BLAST, para esto hace una llamada a ficheros ejecutables externos, la dirección de los mismos es especificada en un archivo de configuración permitiendo así su portabilidad y flexibilidad. Se muestran los métodos que garantizan el alineamiento múltiple y la llamada al BLAST.

Método que ejecuta al Clustal

```
public void doMultAlign(String query, String outfile) throws IOException, InterruptedException {

    Runtime rt = Runtime.getRuntime();
    String strComando = " clustalw" + " -infile=" + query + " -outfile="
            + outfile + " -align";


    System.out.println(strComando);
    FileOutputStream out = new FileOutputStream(query);
    SeqIOTools.writeFasta(out,this.dbSequences);


    rt.exec(strComando).waitFor();//importante para que espere a escribir

}
```

Método que ejecuta al BLAST

```
public  void EjecutarBlast(String query,String out, String basedato, String blast, String max,String e,
   String m) throws IOException, InterruptedException, BioException{
    Runtime runtime = Runtime.getRuntime();

    ValidarFasta(query);//si el fichero no esta correcto lanza una exepcion

       String comando = blast+"program/"+"blastall -i "+ query +
        " -d " + basedato +
        " -m " + m+
        " -e " + e+
        " -p blastn" +
        " -v " + max+
        " -b " + max+
        " -o " + out+
        " -T t";
     System.out.println(comando);
    Process p =runtime.exec(comando);
    p.waitFor();
```

## Clase PatternDatabaseGen.

La clase PatternDatabaseGen es la encargada de realizar el emparejamiento de cebadores, los principales métodos de esta clase se muestran a continuación:

Método para la búsqueda de cebadores.

```
private static boolean Mapea(String text, String foward, String reverse,
        int Lmin, int Lmax, int k, Registro R) {
    if (text == null) {
        text = "";
    }
    int N = text.length();
    if (N > Lmin) {
        String subseq = text.substring(1, N - Lmin);
        int L1 = Ham(foward, subseq, k);
        boolean mapea = (L1 != -1);
        if (mapea) {
            int posi = L1 + Lmin;
            int posf = Math.min(L1 + Lmax, N);
            subseq = text.substring(posi, posf);
            int L2 = Ham(reverse, subseq, k) + posi;
            mapea = L2 != -1;
            if (mapea) {
                return true;
            }
        }
    }

    return false;
}
```

El método Mapea hace una búsqueda de los cebadores que flanquean el gen de interés. El método auxiliar ***Ham*** es el encargado de realizar la búsqueda de patrones en un texto permitiendo k errores, en este caso los patrones son las variables ***foward*** y ***reverse*** que representan los cebadores izquierdo y derecho respectivamente. La variable ***subsec*** representa la secuencia que se desea emparejar con dichos cebadores.

Método ***GenerarPatternDatabase***

```
public static List<Registro> GenerarPatternDatabaseR(Primer primer,
    RegistroService database, int k) throws IOException {
    List<Registro> PatternDB = new LinkedList<Registro>();
    int cont = 0;
    Iterator<Registro> iterador = database.ObtenerTodos().iterator();
    int Lmin = primer.getLmin();
    int Lmax = primer.getLmax();
    String foward = primer.getFoward();
    String reverse = primer.getReverse();
    reverse = PreparарPrimer(reverse);
    while (iterador.hasNext()) {
        Registro R = iterador.next();
        if (!(R instanceof Undefine)) {
        String text = R.getSecuencia();

        if (Mapea(text, foward, reverse, Lmin, Lmax, k, R)) {
                PatternDB.add(R);
                cont++;
            }

        }
    }
    return PatternDB;
}
```

A su vez el **GenerarPatternDatabase (Primer primer, RegistroService database, int k)**, haciendo uso del método ***Mapea*** es el encargado de seleccionar de la Base de Datos todos los registros encontrados por este método.

### 4.5 Pruebas

El principal objetivo del flujo de trabajo de prueba 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 de verificar el correcto cumplimiento de las mismas. 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 algunos casos de pruebas. Para ver los restantes consultar el Anexo 7.

### 4.5.1 Casos de prueba

| **Caso de uso** | Buscar registros que poseen GFCE. |
|---|---|
| Caso de prueba | Buscar registros que poseen GFCE.(Introducidos por el usuario) |
| Entrada | El usuario introduce el cebador izquierdo, el cebador derecho, selecciona la especie a la que pertenecen dichos cebadores e introduce la familia a la que pertenecen dentro de esa la especie, así como los valores de Lmin y Lmax. |
| Resultado esperado | Obtener las secuencias que contienen el gen flanqueado por los cebadores especificados. |
| Resultado de la prueba | El sistema muestra las secuencias que contiene el gen flanqueado por la pareja de cebadores especificada brindando las opciones de realizar Clustal, descargar FASTA, realizar el análisis filogenético así como eliminar todas las secuencias. |
| Condiciones | El valor de Lmin tiene que ser menor que el valor de Lmax. |

| **Caso de uso** | Buscar registros que poseen GFCE. |
|---|---|
| Caso de prueba | Buscar registros que poseen GFCE.(Existentes en la base de datos) |
| Entrada | El usuario selecciona una pareja de cebadores existentes en la base de datos. |
| Resultado esperado | Obtener las secuencias que contienen el gen flanqueado por los cebadores especificados. |
| Resultado de la prueba | El sistema muestra las secuencias que contiene el gen flanqueado por la pareja de cebadores especificada brindando las opciones de realizar ClustalW, descargar FASTA, realizar el análisis filogenético así como eliminar todas las secuencias. |
| Condiciones | - |

| **Caso de uso** | Clasificación taxonómica a partir de la secuencia de MICL. |
|---|---|
| Caso de prueba | Clasificación taxonómica a partir de la secuencia de MICL.(Introduciendo la secuencia). |
| Entrada | El usuario introduce el nombre del trabajo, la secuencia blanco a clasificar taxonómicamente, la cantidad de resultados y el formato que el usuario desea que el sistema muestre, así como el valor de |

| | |
|---|---|
| | expectación. |
| Resultado esperado | Un alineamiento realizado por el BLAST. Este alineamiento es analizado por los investigadores para la posterior clasificación. |
| Resultado de la prueba | El alineamiento realizado por el BLAST. |
| Condiciones | La secuencia no debe contener números ni caracteres degenerados. |

| **Caso de uso** | Clasificación taxonómica a partir de la secuencia de MICL. |
|---|---|
| Caso de prueba | Clasificación taxonómica a partir de la secuencia de MICL. (Desde archivo). |
| Entrada | El usuario introduce el nombre del trabajo, importa el archivo con las secuencias a clasificar taxonómicamente, la cantidad de resultados y el formato que el usuario desea que el sistema muestre, así como el valor de expectación. |
| Resultado esperado | Un alineamiento realizado por el BLAST. Este alineamiento es analizado por los investigadores para la posterior clasificación. |
| Resultado de la prueba | El alineamiento realizado por el BLAST. |
| Condiciones | El archivo debe estar en formato FASTA. |

| **Caso de uso** | Realizar reconstrucción filogenética. |
|---|---|

| | |
|---|---|
| Caso de prueba | Realizar reconstrucción filogenética.(Desde listado). |
| Entrada | El usuario realiza una búsqueda de las secuencias que a él le interese realizar el análisis filogenético, las selecciona y escoge la opción ¨Análisis filogenético¨. |
| Resultado esperado | Obtener el árbol filogenético de las secuencias seleccionadas. |
| Resultado de la prueba | El sistema muestra el árbol filogenético de las secuencias seleccionadas brindando la opción de descargarlo. |
| Condiciones | Debe seleccionar como mínimo dos secuencias para realizar el alineamiento múltiple. |

| | |
|---|---|
| **Caso de uso** | Realizar reconstrucción filogenética. |
| Caso de prueba | Realizar reconstrucción filogenética.(Desde archivo) |
| Entrada | El usuario introduce el nombre del trabajo e importa el archivo con el alineamiento. |
| Resultado esperado | Un árbol filogenético correspondiente al alineamiento. |
| Resultado de la prueba | El árbol filogenético correspondiente al alineamiento brindando la opción de descargarlo. |
| Condiciones | El archivo debe estar en formato FASTA. |

## 4.6 Validación funcional del algoritmo de clasificación taxonómica.

Para la validación funcional del algoritmo de clasificación taxonómica se procedió a realizar una prueba partiendo de un conjunto de secuencias de microorganismos conocidos. Para ello:

- Se definió el total de secuencias registradas por especie y se denominó TotalSP.
- Se procedió a extraer al azar un conjunto prueba de secuencias que constituyen el 10% del total. Dicho conjunto fue seleccionado de manera equitativa entre las diferentes familias de microorganismos conocidos presentes en la base de datos original y fueron apartadas del análisis.
- Seguidamente se procedió a la clasificación del conjunto prueba obteniéndose los resultados que se muestran en la Tabla 13.

**Tabla 13:** Validación funcional del algoritmo de clasificación taxonómica. En la tabla se muestra la cantidad de secuencias por especie seleccionadas como subconjunto prueba, de cada una se muestra cuantas machearon con sus cebadores específicos y cuantas fueron reconocidas por el BLAST con un e-value ≤ 0,05, así como los correspondientes porcientos generales.

| Parejas de cebadores por familias | | | BLAST | | | |
|---|---|---|---|---|---|---|
| | TotalSP | BDP | cantB | Evalue | BLAST + BDP | % del Total |
| *Fibrobacter Succinogenes* | 32 | 26 | 6 | 0.0 | 32 | 100% |
| *Prevotella Ruminicola* | 12 | 4 | 5 | 0.0 | 12 | 100% |
| | | | 1 | e-161 | | |
| | | | 1 | e-151 | | |
| | | | 1 | e-87 | | |
| *Prevotella Bryantii* | 3 | 1 | 2 | 0.0 | 3 | 100% |
| *Prevotella Albensis* | 19 | 15 | 0 | | 3 | 84% |
| *Ruminobacter Aminophilus* | 6 | 4 | 0 | | 4 | 66.% |
| *Selenomonas Ruminatum* | 27 | 22 | 5 | 0.0 | 27 | 100% |
| *Streptococus Bovis* | 12 | 10 | 2 | 0.0 | 12 | 100% |
| *Eubacterium Ruminatum* | 5 | 2 | 0 | | 2 | 40% |
| *Ruminucocus Favefaciens* | 5 | 5 | 0 | 0.0 | 5 | 100% |

El proceso de clasificación consta de dos pasos: primero se explora si las secuencias problema machean con alguna de las 11 parejas de cebadores definidos en la aplicación ( Anexo 8) de ser así ya se consideran como pertenecientes al grupo taxonómico correspondiente; y pasan a formar parte de

la BDP del mismo. Seguidamente se realiza la búsqueda por homología empleando el algoritmo BLAST entre las diferentes BDP y el resto del conjunto prueba, con el objetivo de detectar aquellas secuencias que no contienen los cebadores, pero sin embargo pertenecen a un grupo taxonómico definido. Solo se consideraron como significativas aquellas secuencias que fueron reconocidas por el BLAST con un e-value menor de 0,005.

Como se puede observar en la Tabla 1 para esta prueba se utilizaron solo 8 parejas de cebadores, pues tenemos 2 parejas para las cuales no hay o hay muy poca representación de secuencias en la base de datos original (Anexo 8).

Como se puede apreciar, en 6 de las 8 familias analizadas se logró la correcta clasificación del 100% de las secuencias del conjunto prueba, mientras que para el caso de *Eubacterium Ruminatum*, solo logramos clasificar 2 de las 5 secuencias escogidas mediante el macheo con los cebadores característicos, para un 40% de efectividad del método en este caso. Este fenómeno pudiera deberse a que las 3 secuencias que se quedan sin clasificar son fragmentos y por lo tanto no contienen a los cebadores específicos. Por otra parte la poca representatividad de esta familia en la base de datos original, pudiera ocasionar que la etapa de búsqueda por homología no sea efectiva.

En caso de *Prevotella Ruminicola*, se obtuvo el 100% de clasificación pero con diferentes valores de probabilidad reportados por BLAST, pero en todos los caso se registran menores a 0,05

En general, la efectividad del método se estimó en un 87%, para el total de secuencias probadas.

## 4.7 Clasificación taxonómica de los microorganismos indefinidos imposibles de cultivar en el laboratorio.

Finalmente se procedió a la clasificación taxonómica de 328 secuencias obtenidas de Gen Bank que se reportan como microorganismos del rumen indefinidos, porque no se han podido cultivar y/o caracterizar en el laboratorio. La herramienta de clasificación logró asignar un grupo taxonómico al 92.68% del total (304 de 328 secuencias analizadas), distribuidas en 11 grupos taxonómicos diferentes (Tabla 14). En la mayoría de los casos el macheo con los cebadores identifica una pequeña cantidad que luego es enriquecida en la etapa de búsqueda por homología. Sin embargo existen casos particulares como *Fibrobacter Succinogenes* cuyo grupo taxonómico no fue asignado a ninguna de las secuencias analizadas, lo que significa que en el conjunto de microorganismos no cultivados no parece

incluirse ningún representante. Esto pudiera explicarse por el hecho de que esta especie bacteriana se caracteriza por formar grandes colonias sobre celulosa cristalina y se aísla fácilmente de su sustrato durante la preparación de las muestras.

**Tabla14:** Clasificación taxonómica de microorganismos imposibles de cultivar a partir de sus secuencias obtenidas del GenBank.

| Parejas de cebadores por familias | No. de secuencias en la BDP | No. de secuencias que machean por pareja de cebadores | No. de secuencias reconocidas por BLAST con e-value = 0.0 |
|---|---|---|---|
| *Fibrobacter Succinogenes* | 28 | - | - |
| *Prevotella Ruminicola* | 8 | 1 | 34 |
| *Prevotella Bryantii* | 2 | 5 | 30 |
| *Prevotella Albensis* | 3 | - | 77 |
| *Ruminobacter Aminophilus* | 4 | - | 1 |
| *Selenomonas Ruminatum* | 23 | 15 | 36 |
| *Streptococus Bovis* | 10 | 1 | 18 |
| *Eubacterium Ruminatum* | 1 | 1 | 2 |
| *Ruminucocus Flavefaciens* | 5 | 17 | 77 |
| *Anaerovibrio Lypolítica* | 1 | 1 | 27 |
| *Treponema Bryantii* | 1 | 1 | 2 |

En el caso de *Prevotella Albensis* y *Ruminobacter Aminophilus* el método de macheo por cebadores no arrojó ningún resultado, probablemente porque las secuencias de este tipo presentes en la base de datos son parciales en su mayoría y por lo tanto no machean con los cebadores para un error cero. Sin embargo el BLAST, logró reconocer 77 secuencias homólogas al grupo de *Prevotella Albensis,* esto se debe a que durante la búsqueda por homología a partir de la BDP correspondiente, machean todas aquellas *Prevotellas sp* existentes en la base de datos solapándose incluso con las clasificadas

anteriormente en los grupos de *Prevotella Ruminicola* y *Prevotella Bryantii*. Lo que significa que el método es más fiable a la hora de asignar un género (en este caso *Prevotella*), pero no logra discernir entre diferentes especies dentro del mismo. Por último en el caso de *Ruminobacter Aminophilus* no obtuvimos una clasificación significativa por ninguna de las dos vías, hecho que se explica por la poca representatividad de este microorganismo en la base de datos.

A modo de conclusión en este capítulo se realizaron los diagramas de componentes de los principales casos de uso a los que se les viene dando seguimiento desde el diseño, desarrollando su implementación. Se muestran ejemplos de código de las principales clases. Se hizo la descripción del manejo de la seguridad usando el framework Acegi Security. Se realizaron pruebas de funcionalidad mediante el uso de la técnica de partición de equivalencia. La prueba al método de clasificación arrojó valores positivos, mostrados en este capítulo lo cual representa un importante resultado validando así el principal método implementado en la aplicación.

## Conclusiones

Como resultado de la investigación realizada se concluye que:

- Se construyó una base de datos de secuencias de microorganismos del rumen obtenida del GenBank filtrada por los investigadores del ICA.
- Se diseñó e implementó una aplicación web usando las tecnologías y patrones de diseño ajustados a las necesidades y justificados mediante una investigación previa, incluyendo herramientas necesarias para el análisis y gestión de la información, como son: análisis filogenético, alineamientos y clasificación taxonómica de secuencias
- Se logró asignar un grupo taxonómico al 92.68% del total (304 de 328 secuencias analizadas), de los microorganismos imposibles de cultivar en el laboratorio a partir de sus secuencias, haciendo uso del algoritmo de reconocimiento de cadenas basado en cebadores.

## Recomendaciones

- Enriquecer la base de datos de secuencias para lograr una mayor representación de las diferentes especies de microorganismos presentes en la comunidad del rumen.
- Ampliar el número de cebadores característicos de un grupo taxonómico bacteriano, para enriquecer el método de clasificación.
- Extender el método de clasificación, utilizando cebadores característicos de otras clases de microorganismos presentes en la comunidad microbiana del rumen.
- Extender la utilidad de la base de datos a otras comunidades microbianas y con el seguimiento y perfeccionamiento de la misma, en un futuro ponerla al alcance de todos los investigadores del país.
- Adicionar un módulo que permita consumir el servicio web brindado por el NCBI, para adicionar automáticamente nuevos registros a la base de datos.

## Referencias Bibliográficas

1. National Center for Biotechnology Information NCBI _ GeneBank. [En línea] [Citado el: 20 de Marzo de 2009.] http://www.ncbi.nlm.nih.gov.

2. **Klieve, A. & Swain, R.A.** *Estimation of ruminal bacteriophage numbers by pulsed-fiel gel electrophoresis and lasser densitometry.* s.l. : Appl. Environ. Microbiol, 1993. ISBN 59:2299-2303.

3. *Bacteria, fungi and protozoa of rumen.* **Hespell, R.B., Akin, D.E. Mackie, R.I., White, B.A. and Isaacson, R Dehority, B.A.** New York : International Thomson Publishing, 06 de 1997, Gastrointestinal Microbiology., Vol. 2, pág. 59.

4. *Analysis of Bacterial Community Composition by Oligonucleotide.* **Borneman, J., Valinsky, L., Scupham, A.J., Vedova, G.D., Liu, Z., Figueroa, A., Jampachaisri, K., Yin, B., Bent, E., Mancini-Jones, R., Press, J., Jiang, T.** No. 7, Dordrecht : American Society for Microbiology, 06 de 2002, Applied and Environmental Microbiology, Vol. Vol. 68.

5. *Genetic diversity in Sargasso Sea bacterioplankton.* **Giovannoni, S. J., T. B. Britschgi, C. L. Moyer, and K. G. Field.** Oregon, USA : Nature Publishing Group, 05 de 1990, Nature, Vol. 345, págs. 60–63.

6. *A molecular view of microbial diversity and the biosphere.* **Pace, N. R.** 5313, 05 de 1997, Science, Vol. 276, págs. 734 - 740.

7. *A software system for gene sequence database construction based on fast approximate string matching.* **Zheng Liu, James Borneman, Tao Jiang.** 3, s.l. : Inderscience Enterprises Ltd, 2005, Int. J. Bioinformatics Research and Applications, Vol. 1.

8. *Profiling of complex microbial populations by denaturing gradient gel electrophoresis analysis of polymerase chain reaction amplified genes coding for 16S rRNA.* **Muyzer, G., E. D. De Waal, and A. G. Uitterlinden.** 3, s.l. : Appl Environ Microbiol, 03 de 1993, Appl Environ Microbiol, Vol. 59, págs. 695-700.

9. *Environmental genome shotgun sequencing of the Sargasso Sea.* **Venter JC, Remington K, ... and Smith HO.** 5667, 2 de 04 de 2004, Science, Vol. 304, págs. 66 - 74.

10. *Microbial diversity in the deep sea and the underexplored "rare biosphere".* **Sogin ML, Morrison HG, Huber JA, Welch DM, Huse, SM, Neal PR, Arrieta JM, Herndl GJ.** 32, Cambridge, MA : s.n., 20 de 06 de 2006, Proc Natl Acad Sc, Vol. 103, págs. 12115-20.

11. *Microbial population structures in the deep marine biosphere.* **Huber JA, Welch DB, Morrison HG, Huse SM, Neal PR,Butterfield DA, Sogin ML.** 5847, 05 de 10 de 2007, Science, Vol. 318, págs. 97-100. 1146689.

12. *Oligonucleotide Fingerprinting of Ribosomal RNA Genes (OFRG).* **Valinsky, L., Scupham, A.J., Vedova, G.D., Liu, Z., Figueroa, A., Jampachaisri, K., Yin, B., Bent,E., Mancini-Jones, R., Press, J., Jiang, T. and Borneman, J.** Dordrecht, The Netherlands : Kluwer Academic Publishers, 2004, Molecular Microbial Ecology Manual, págs. pp.569–585.

13. *The Ribosomal Database Project (RDP-II): previewing a new autoaligner that allows regular updates and the new prokaryotic taxonomy.* **Cole, J.R., Chai, B., Marsh, T.L., Farris, R.J., Wang, Q., Kulam, S.A., Chandra, S.,McGarrell, D.M., Schmidt, T.M., Garrity, G.M. and Tiedje, J.M.** 1, 01 de 2003, Nucleic Acids Res, Vol. 31, págs. 442- 443.

14. *The European database on small subunit ribosomal RNA.* **Wuyts, J., van de Peer, Y., Winkelmans, T. and de Wachter, R.** 1, s.l. : Oxford University Press , 2002, Nucleic Acids Res, Vol. 30, págs. 183–185.

15. *A guided tour to approximate string matching.* **Navarro, G.** 1, New York, USA : ACM, 03 de 2001, ACM Computing Surveys , Vol. 33. ISSN:0360-0300 .

16. *Fast and practical approximate string matching.* **Baeza-Yates, R.A. and Perleberg.** Santiago, Chile : s.n., 1992, Information Processing Letters, Vol. 59, págs. 185-192.

17. *Approximate Boyer-Moore String matching.* **Ukkonen, J Tarhio anj E.** 1993, SIAM J. Comput, Vol. 22, págs. 243-260.

18. *Fast string matching with mismatches.* **Gonnet, Baeza-Yates and G.H.** Duluth, MN, USA : Academic Press, Inc, 02 de 1994, Vol. 108, págs. 187 - 199 . ISSN:0890-5401 .

19. *A fast string searching algorithm.* **Boyer, R.S and Moore.** 20, s.l. : Communications of the ACM, 10 de 1977, Vol. 10. ISSN:0001-0782 .

20. **Jacobson, Ivar.** *El proceso unificado de desarrollo de software.* s.l. : Addison-Wesley, 2000. ISBN: 8478290362.

21.**Pressman, Roger S.** *Ingeniería de Software, Un enfoque Práctico.* s.l. : McGraw-Hill, 2001. ISBN:970-10-5473-3.

22. **Gamma, Erich.** *Patrones de diseño.* s.l. : Addison-Wesley Iberoamericana España, S.A., 1995. 978-84-7829-059-8.

23. Herramientas CASE. [En línea] 4 de Diciembre de 2007. [Citado el: 13 de 2 de 2009.] http://www.cyta.com.ar/biblioteca/bddoc/bdlibros/proyectoinformatico/libro/c5/c5.htm.

24. **David Gallardo, Ed Burnette, Robert McGovern, Steven Haines.** *Eclipse in action: a guide for Java developers.* s.l. : Manning Publications Co., 2003. ISBN 1930110960.

25. **Iverson, Will.** *A J2EE™ Developer's Guide.* s.l. : Addison-Wesley Professional, 22 de 11 de 2004.

26. **Eckel, Bruce.** *Piensa en Java.* s.l. : Pearson Educación, 2002. ISBN: 8420531928.

27. **Balani, Naveen.** *The Spring series, Part 1: Introduction to the Spring framework.* [En línea] 6 de 2005. [Citado el: 3 de 12 de 2008.] http://www-128.ibm.com/developerworks/web/library/wa-spring1/.

28. The Spring series, Part 1: Introduction to the Spring framework. [En línea] junio de 2005. [Citado el: 4 de diciembre de 2008.] http://www-128.ibm.com/developerworks/web/library/wa-spring1/.

29.*acegi_spring_security.* [En línea] [Citado el: 12 de 04 de 2009.] http://www.leandroiriarte.com.ar/spanish/acegi_spring_security.php.

30. **Bauer, Christian.** *Hibernate in action.* s.l. : Manning, 2005. ISBN-13: 978-1932394153.

31.PostgreeSQL. [En línea] 25 de marzo de 2009. [Citado el: 23 de 03 de 2009.] http://www.postgresql.org/about/press/features83.

32. *Basic local alignment search tool.* **Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ.** [ed.] Research Support. 403-10, U.S. Gov't, P.H.S. : s.n., 1990, J Mol Biol, Vol. 215. 2231712.

33. *Multiple sequence alignment with the Clustal series of programs.* **Chenna R, Sugawara H, Koike T, Lopez R, Gibson TJ, Higgins DG, Thompson JD.** 13, 2003, Nucleic Acids Res, Vol. 31, págs. 3497–3500. ISSN: 0305-1048.

34. **Roderic D. M. Page, Edward C. Holmes.** *Molecular evolution: a phylogenetic approach.* s.l. : Wiley-Blackwell, 1998. ISBN 0865428891.

35. *A Brief Introduction to the Phylogenetic Analysis Library Version 1.5.* **Matthew Goode, Korbinian Strimmer, Alexei Drummond, Ed Buckler, Allen Rodrigo.** s.l. : Australian Computer Society, Inc, 2004. 2nd Asia-Pacific Bioinformatics Conference (APBC2004), Dunedin, New Zealand. Vol. 29.

36. **Consulting, Milestone.** milestone.com. [En línea] 2000-2009. [Citado el: 20 de 04 de 2009.] http://www.milestone.com.mx/articulos/uso_de_uml_en_aplicaciones_web.htm.

37. *IEEE Recommended Practice for Architectural Description of Software-Intensive Systems.* **Committee, Software Engineering Standards.** United States of America. : Published xx Month, 21 de 09 de 2000, The Institute of Electrical and Electronics Engineers, Inc.

38. **Walls, Craig, Ryan Breidenbach.** *Spring in Action.* s.l. : Greenwich: Manning Publications, 2007. ISBN:9781933988139 .

39. **Viklund, Andreas.** stackframe.net. *Patrones de Diseño.* [En línea] 31 de octubre de 2008. [Citado el: 17 de Abril de 2009.] http://www.stackframe.net/es/content/10-2008/patrones-de-diseno-abstract-factory-pattern..

40. *The ClustalX windows interface: flexible strategies for multiple sequence alignment aided by quality analysis tools.* **Thompson JD, Gibson TJ, Plewniak F, Jeanmougin F, Higgins DG.** Nucleic Acids Research, Vol. 25, págs. 4876-4882.

41. **Risberg.** Developing a Spring Framework MVC application step-by- step. [En línea] julio de 2003. [Citado el: 10 de enero de 2008.] www.springframework.org/docs/MVC-step-by-step/Spring-MVC-stepby-step.html.

42. *Fast Regular Expression Search.* **Raffinot, Gonzalo Navarro and Mathieu.** 6, s.l. : Springer-Verlag Berlin Heidelberg, 01 de 01 de 1999, LNCS, Vol. 1968, págs. 198-212. ISBN: 978-3-540-66427-7.

43. **P. N. Hobson, C. S. Stewart.** *The rumen microbial ecosystem.* 2. New York : Springer, 1997. ISBN 0751403660, 9780751403664.

44. **Orpin, C.G. & Joblin, K.N.** The rumen anaerobic fungi. [aut. libro] C. S. Stewart P. N. Hobson. *The Rumen microbial ecosystem.* New York : Chapman and Hall, 1997.

45. *Approximate regular expression searching with arbitrary integer weights.* **Navarro, G.** s.l. : Lect. Notes Comp. Sc, 2003, Vol. 2906.

46. *Ruminal Fisiology: Digestion, metabolism, growth and reproduction.* **Mackie, R.I., Aminov, R.I., White, B.A. & McSweeney, C.S.** s.l. : CAB. International. UK, Molecular ecology and diversity in gut microbial ecosystems, pág. 61.

47. **Johnson, Rob.** Introduction to the Spring Framework. [En línea] mayo de 2005. [Citado el: 4 de diciembre de 2008.] www.theserverside.com/articles/article.tss?l=SpringFramework.

48. *A new approach to text searching.* **Gonnet, Baeza-Yates G.H.** 10, New York, USA : ACM, 1992, Communications of the ACM, Vol. 35. ISSN:0001-0782.

49. *MiCA: A Web-Based Tool for the Analysis of Microbial.* **Conrad Shyu, Terry Soule, Stephen J. Bent, James A. Foster and Larry J. Forney.** University of Idaho, Moscow, USA : Springer Science + Business Media, 04 de 2007, Microbial Ecology, Vol. 53, págs. 562–570. 83843.

## Bibliografía

1. [En línea] [Citado el: 24 de Marzo de 2009.] **http://sophia.javeriana.edu.co/~lcdiaz/ADOO2006-3/grasp_cpaternostro-lvargas-jviafara.pdf**.

2. cibernetia. Conceptos básicos del servidor web. [Online] **http://www.cibernetia.com/manuales/instalacion_servidor_web/1_conceptos_basicos.php**.

3. cyta.com.ar. *Herramientas CASE.* [En línea] [Citado el: 20 de Febrero de 2009.] **http://www.cyta.com.ar/biblioteca/bddoc/bdlibros/proyectoinformatico/libro/c5/c5.htm**.

4. Entorno Virtual de Aprendizaje. Conferencia 2: Arquitectura de Patrones. Ingenieria de Software II. Curso 2007-2008. *http://teleformación.* [Online] [Cited: Febrero 25, 2009.] **http://teleformacion.uci.cu/mod/resource/view.php?id=14077**.

5. Entorno Virtual de Aprendizaje. Conferencia 4 de Ingenieria de Software I. Curso 2007-2008. *http://teleformación.uci.cu.* [En línea] [Citado el: 4 de Marzo de 2009.] **http://teleformacion.uci.cu/mod/resource/view.php?id=12103**.

6. Entorno Virtual de Aprendizaje. Conferencia 7 de Ingenieria de Software I. Curso 2007-2008. *http://teleformación.uci.cu.* [Online] [Cited: Marzo 4, 2009.] **http://teleformacion.uci.cu/course/view.php?id=102**.

7. Entorno Virtual de Aprendizaje. Conferencia de la semana 7 de Ingenieria de Software II. Curso 2007-2008. *http://teleformación.uci.cu.* [En línea] [Citado el: 5 de Marzo de 2009.] **http://teleformacion.uci.cu/mod/resource/view.php?id=14094**.

8. Patrones GRASP. [En línea] [Citado el: 16 de Febrero de 2009.] **http://eisc.univalle.edu.co/materias/Material_Desarrollo_Software/exposiciones2005B/polimorfismoFabricacionPura_G01/**.

9. **Alvarez, Sara.** desarrolloweb.com. *Modelo de Bases de Datos.* [En línea] 14 de Agosto de 2007. [Citado el: 26 de Marzo de 2009.] **http://www.desarrolloweb.com/articulos/modelos-base-datos.html**.

10. **Astudillo, Marcello Visconti y Hernán.** inf.utfsm.cl. *Fundamentos de Ingenieria de Software.* [En línea] [Citado: Marzo 24, 2009.] **http://www.inf.utfsm.cl/~visconti/ili236/Documentos/16-PatronesGRASP.pdf**.

11. **Gutierrez, Jorge A. Saavedra.** El Mundo Informático<<Patrones GRASP>>. [En línea] [Citado el: 24 de Marzo de 2009.] **http://jorgesaavedra.wordpress.com/category/patrones-grasp/**.

12. **Viklund, Andreas.** stackframe.net. *Patrones de Diseño.* [Online] Octubre 31, 2008. [Cited: Abril 17, 2009.] **http://www.stackframe.net/es/content/10-2008/patrones-de-diseno-abstract-factory-pattern**.

13. **Pastor, Javier.** The INQUIRER.es. *PostgreSQL 8.3 disponible.* [En línea] 5 de Febrero de 2008. [Citado el: 24 de Marzo de 2009.] **http://www.theinquirer.es/2008/02/05/postgresql_83_disponible.html**.

14. PostgreSql. [En línea] [Citado el: 25 de Marzo de 2009.] **http://www.postgresql.org/about/press/features83**.

15. **JACOBSON, IVAR, BOOCH, GRADY y RUMBAUGH, JAMES.** *El Proceso Unificado de Desarrollo de Software.* Madrid : PEARSON EDUCACION, S.A, 2000. ISBN:84-7829-036-2.

16. **Mendoza, Gonzalo Mena.** *Procesos de la Ingeniería de Requerimientos.* [Multimedia] Santiago de Querétaro : s.n., 2005.

17. siona.udea.edu.co. *¿Arquitectura de Software?* [En línea] [Citado el: 20 de Abril de 2009.] **http://siona.udea.edu.co/~aoviedo/Arquitectura%20de%20Software/Arquitectura%20de%20Software.htm**.

18. espanol.finance.yahoo.com. [En línea] [Citado el: 23 de Abril de 2009.] **http://espanol.finance.yahoo.com/q/hp?s=LIMS&a=04&b=24&c=2007&d=03&e=24&f=2009&g=m**.

19. *Ayuda Extendida Rational Unified Process(RUP).* 2003.

20. Entorno Virtual de Aprendizaje. Conferencia 1 Introducción al proceso de desarrollo de software. Curso 2007-2008. *http://telefomacion.uci.cu.* [Online] [Cited: Febrero 10, 2009.] **http://teleformacion.uci.cu/mod/resource/view.php?id=11402**.

## Anexos

**Anexo 1** Descripciones de casos de uso del sistema.

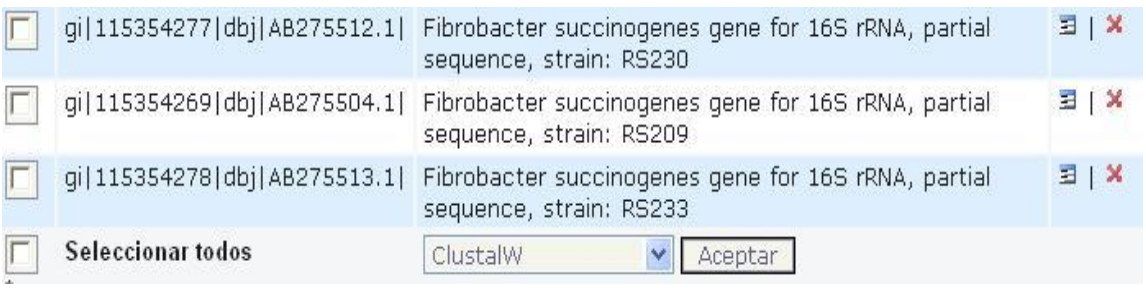
**CUS Realizar AMS usando CLUSTALW.**

| | |
|---|---|
| **Nombre del Caso de Uso** | Realizar AMS usando CLUSTALW. |
| **Actores** | Administrador, Investigador, Usuario anónimo. |
| **Propósito** | Ver el resultado del alineamiento para su posterior estudio. |
| **Resumen** | El caso de uso se inicia cuando el actor decide realizar un alineamiento múltiple.<br><br>El sistema le muestra la interfaz correspondiente y ejecuta las acciones necesarias. El caso de uso finaliza cuando se emite el resultado de la operación solicitada. |
| **Referencias** | R2,R2.2,R2.2.1, R2.2.2, R2.2.3, R2.2.4 |
| **Precondiciones** | - |
| **Pos condiciones** | El sistema realiza los alineamientos y visualiza los mismos. |

| **Flujo normal de los eventos** | |
|---|---|
| **Acción del actor** | **Respuesta del Sistema** |
| | 1. El sistema muestra la interfaz correspondiente con los siguientes campos: |

| | |
|---|---|
| | **“Nombre del trabajo”** Corresponde al nombre de la acción a realizar.<br><br>**“Desde archivo”** El usuario cargará un fichero de secuencias en formato fasta para realizarle el alineamiento múltiple |
| 2. El actor selecciona la forma en que desea realizar el alineamiento. | 3. Si desea realizar un alineamiento con secuencias existentes en la Base de Datos ir a sección: ¨**Realizar alineamiento múltiple desde listado**¨<br><br>Si desea realizar un alineamiento múltiple de secuencias en un archivo ir a la sección: ¨**Realizar alineamiento múltiple desde archivo**¨ |
| **Sección “Realizar Alineamiento múltiple desde listado”.** | |
| **Acción del actor** | **Respuesta del Sistema** |
| | 1. Muestra la interfaz correspondiente y se llama al caso de uso **“Buscar registros”.** |
| 2. Selecciona a partir del listado de secuencias las que le interese alinear y pulsa en el botón | 3. Realiza el alineamiento. |

| | |
|---|---|
| **"enviar".** | |
| | 4. Muestra el resultado en un listado dando la opción de descargarlo en formato FASTA o CLUSTAL así como la opción para graficar el árbol filogenético correspondiente a las secuencias alineadas. |
| 5.El actor selecciona una de las opciones:<br>"Descargar CLUSTAL"<br>"Descargar FASTA"<br>"Reporte Árbol filogenético" | 6. Según sea la opción seleccionada el sistema la realiza |
| | |
| **Flujos alternos Sección "Realizar alineamiento múltiple desde listado"**. | |
| 2.1) El actor no desea introducir el nombre del trabajo ni realizar el alineamiento y sale de la sección. | |
| | 4.1) Si el actor después de ver el alineamiento no decide descargar ninguna información referente al mismo sale de la sección. |



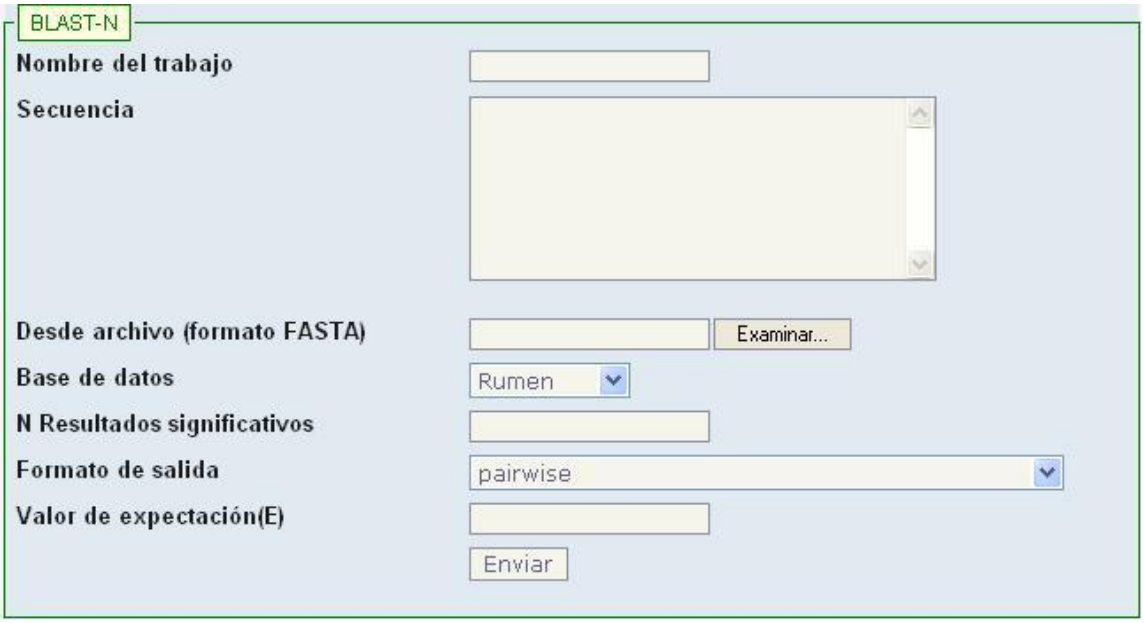

| **Sección "Realizar Alineamiento múltiple desde archivo".** | |
|---|---|
| **Acción del actor** | **Respuesta del Sistema** |
| 1. El actor selecciona el archivo en formato FASTA y pulsa en el botón **"Alinear".** | 2. Copia el archivo en el servidor y se realiza el alineamiento múltiple. |
| | 3. Muestra el resultado en un listado dando la opción de descargarlo en formato FASTA o CLUSTAL así como ver el reporte del árbol filogenético. |
| 4.El actor selecciona una de las opciones:<br>"Descargar CLUSTAL"<br>"Descargar FASTA"<br>"Reporte Árbol filogenético" | 5. Según sea la opción seleccionada el sistema la realiza. |
| | |
| **Flujos alternos Sección "Realizar Alineamiento múltiple desde archivo"** | |

| | |
|---|---|
| 1.1) El actor no desea realizar el alineamiento y sale de la sección. | |
| | 4.1) Si el actor después de ver el alineamiento no decide descargar ninguna información referente al mismo sale de la sección. |
|  | |

**CU Realizar BH usando BLAST-N.**

| | |
|---|---|
| **Nombre del Caso de Uso** | Realizar BH usando BLAST-N. |
| **Actores** | Administrador, Investigador, Usuario anónimo. |
| **Propósito** | Ver el resultado del alineamiento para su posterior estudio. |
| **Resumen** | El caso de uso se inicia cuando el actor decide saber la homología entre secuencias especificadas.<br><br>El sistema le muestra la interfaz correspondiente y ejecuta las acciones necesarias. El caso de uso finaliza cuando se emite el resultado de la operación solicitada. |
| **Referencias** | R2,R2.1,R2.1.1,R2.1.2 |

| | |
|---|---|
| **Precondiciones** | - |
| **Pos condiciones** | El sistema visualiza el resultado. |




| **Flujo normal de los eventos** | |
|---|---|
| **Acción del actor** | **Respuesta del Sistema** |
| | 1. El sistema muestra la interfaz correspondiente con los siguientes campos:<br><br>**"Nombre del trabajo"** es el campo donde el actor introduce el nombre que decida ponerle a su trabajo.<br><br>**"Secuencia"** es el campo que brinda la posibilidad de introducir la secuencia a partir de la cual se quiere ver su |

| | |
|---|---|
| | homología con las restantes en la base de datos.<br><br>**"N resultados significativos"** es el campo donde se especifica la cantidad de secuencias mas semejantes a la(s) secuencia(s) entrada(s).<br><br>**"Base de datos"** es el campo donde se especifica la base de datos con la cual se quiere hacer la homología.<br><br>**"Formato de salida"** es el campo donde se especifica el formato en que se mostrará el resultado.<br><br>**"Valor de expectación (E)"** es el campo donde se introduce el valor de expectación.<br><br>**"Desde archivo"** es el campo donde el usuario seleccionará un fichero de secuencias en formato FASTA para subirlo al servidor y realizarle el BLAST.<br><br>Si el actor introduce una secuencia entonces se deshabilita el campo **"Desde archivo"** y viceversa. |
| 2. El actor selecciona la forma en que desea realizar el BLAST-N. | 3. Si desea realizar un BLAST-N con la secuencia que le interese ir a la sección:**"Realizar BLAST-N** |

| | |
|---|---|
| | **introduciendo una secuencia blanco”**<br><br>Si desea hacer un BLAST-N a partir de un archivo en formato FASTA ir a la sección: **“Realizar BLAST-N desde archivo”** |
| **Sección “Realizar BLAST-N introduciendo una secuencia blanco”.** | |
| **Acción del actor** | **Respuesta del Sistema** |
| 1. El actor llena los campos y pulsa en el botón **“Buscar”.** | 2. Realiza el BLAST-N entre la secuencia y la Base de Datos y muestra el resultado |
| | |
| **Flujos alternos Sección “Realizar BLAST-N introduciendo una secuencia blanco”**. | |
| 2.1) El actor no desea realizar el BLAST-N y sale de la sección. | |
| | |
| **Sección “Realizar BLAST-N desde archivo”.** | |
| **Acción del actor** | **Respuesta del Sistema** |
| 1. El actor selecciona el archivo en formato | 2. Copia el archivo en el servidor |

| | |
|---|---|
| FASTA para el servidor y pulsa en el botón **"Buscar".** | |
| | 3. Realiza el BLAST-N y muestra el resultado. |
| | |
| **Flujos alternos Sección "Realizar BLAST-N desde archivo"** | |
| 2.1) El actor no desea realizar el BLAST y sale de la sección. | |
| | |

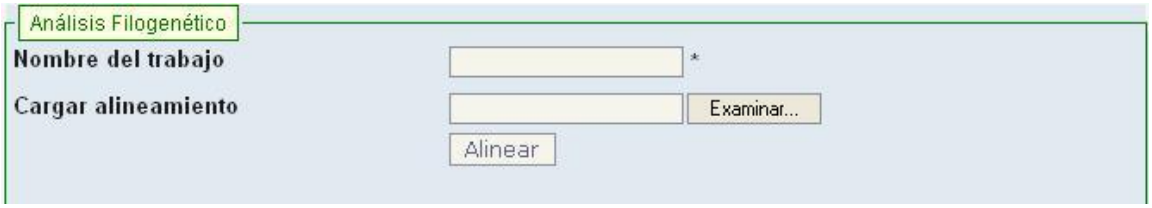
**CUS Realizar reconstrucción filogenética.**

| | |
|---|---|
| **Nombre del Caso de Uso** | Realizar reconstrucción filogenética. |
| **Actores** | Administrador, Investigador, Usuario anónimo. |
| **Propósito** | Representar mediante un grafo en forma de árbol, las relaciones filogenéticas entre las secuencias seleccionadas. |
| **Resumen** | El caso de uso se inicia cuando el actor va a realizar alguna de las siguientes operaciones:<br>● Realizar análisis filogenético desde listado<br>● Realizar análisis filogenético desde fichero.<br>El sistema le muestra la interfaz correspondiente. El caso de uso finaliza cuando se emite el resultado de la operación solicitada. |
| **Referencias** | R3,R3.1,R3.2,R3.3 |

| | |
|---|---|
| **Precondiciones** | - |
| **Pos condiciones** | El sistema muestra el resultado del análisis filogenético. |
| | |

| **Flujo normal de los eventos** | |
|---|---|
| **Acción del actor** | **Respuesta del Sistema** |
| 1. El actor desea realizar una de las siguientes operaciones:<br>● Realizar análisis filogenético desde listado.<br>● Realizar análisis filogenético desde fichero. | 2. En dependencia de la operación que solicita:<br>● Si decide realizar el análisis filogenético desde un listado para seleccionar los registros que desee, ir a Sección **"RAF desde listado"**.<br>● Si decide realizar el análisis filogenético desde fichero, ir a Sección **"RAF desde fichero"**. |
| **Sección "RAF desde listado"** | |
| **Acción del actor** | **Respuesta del Sistema** |
| | 1. Muestra la interfaz correspondiente para realizar el análisis filogenético desde un listado:<br><br>Se llama al caso de uso **"Buscar registro".** |
| 2. El actor debe seleccionar los | 3. Muestra el resultado dando la posibilidad de |

| | |
|---|---|
| registros a los cuales desea realizarle el análisis filogenético y selecciona la opción **"Análisis filogenético"** y pulsa en el botón **"Aceptar"** | descargar el árbol filogenético. |
| | |
| **Flujos alternos Sección "RAF desde listado"** | |
| 2.1) Si no selecciona los registros para realizar el análisis filogenético el sistema no responde. | |
| | 3.1) Si no decide descargar el árbol sale de la sección. |
| | |
| **Sección "RAF desde fichero"** | |
| **Acción del actor** | **Respuesta del Sistema** |
| | 1.Muestra la interfaz correspondiente para realizar el análisis filogenético con los siguientes campos:<br>**-¨Nombre del trabajo¨**<br>**-¨Seleccionar archivo¨** |

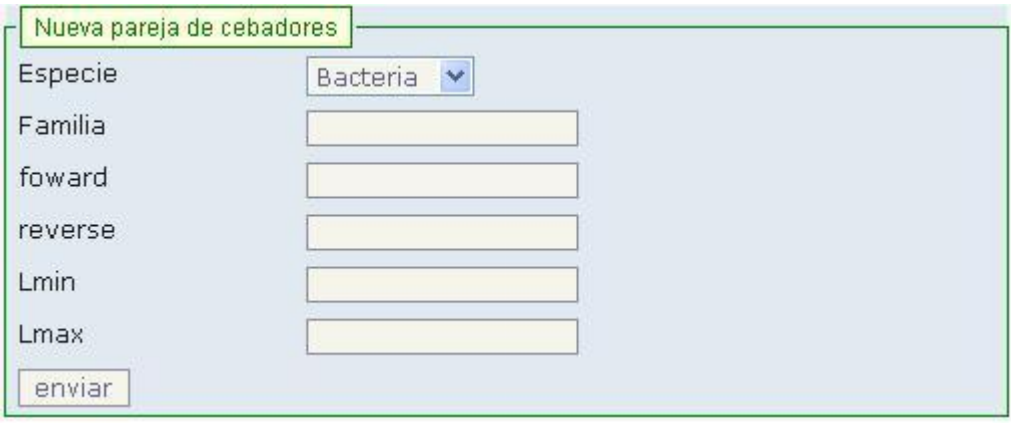
| | |
|---|---|
| 2. El actor selecciona el archivo en formato CLUSTAL y pulsa en el botón **"Aceptar".** | 3. Muestra el resultado dando la posibilidad de descargar el árbol filogenético. |
| 4.Si el actor decide descargar el árbol pulsa en el vinculo **"descargar"** | |
| **Flujos alternos Sección "RAF desde fichero"** | |
| 2.1. El actor no desea seleccionar el archivo y sale de la sección. | |
| 4.1) Si no decide descargar el árbol sale de la sección. | |
|  | |

**CUS Gestionar cebadores.**

| | |
|---|---|
| **Nombre del Caso de Uso** | Gestionar cebadores. |
| **Actores** | Administrador, Investigador. |
| **Propósito** | Brinda la posibilidad de insertar una nueva pareja de cebadores en |

| | |
|---|---|
| | la base de datos, así como modificar o eliminar una ya existente. |
| **Resumen** | El caso de uso se inicia cuando el actor va a realizar alguna de las siguiente operaciones relacionadas con registro:<br>● Insertar una nueva pareja de cebadores.<br>● Editar cebadores.<br>● Eliminar una pareja de cebadores.<br><br>El sistema le muestra la interfaz correspondiente. El caso de uso finaliza cuando se emite el resultado de la operación solicitada. |
| **Referencias** | R4,R4.1,R4.2,R4.3,R4.4,R8 |
| **Precondiciones** | Que el actor se haya autenticado en la aplicación. |
| **Pos condiciones** | El sistema inserta una nueva pareja de cebadores, así como elimina y modifica los datos de alguna pareja de cebadores existente. |

| **Flujo normal de los eventos** | |
|---|---|
| **Acción del actor** | **Respuesta del Sistema** |
| 1. El actor quiere realizar una de las siguientes operaciones:<br>● Insertar una nueva pareja de cebadores.<br>● Editar cebadores.<br>● Eliminar una pareja de cebadores. | 2.En dependencia de la operación solicitada:<br>● Si decide insertar una nueva pareja de cebadores, ir a Sección **"Insertar nueva pareja de cebadores"**.<br>● Si desea insertar editar una pareja de cebadores, ir a Sección **"Editar cebadores".**<br>● Si decide eliminar una pareja de cebadores, ir a la Sección **"Eliminar** |

| | |
|---|---|
| | **cebadores".** |
| **Sección "Insertar nueva pareja de cebadores"** | |
|  | |
| **Acción del actor** | **Respuesta del Sistema** |
| | 1.Muestra la interfaz correspondiente para la inserción de una nueva pareja de cebadores con los siguientes campos:<br><br>- **"Especie"** es el campo que brinda la posibilidad de escoger el reino al que pertenece el cebador a insertar (Bacteria, Hongo, Protozoo, Archaea o Indefinido si no se sabe el reino al que pertenece.<br><br>- **"Familia"** es el campo que brinda la posibilidad de introducir la familia a la que pertenece el cebador a insertar.<br><br>- **"foward"** es el campo donde el actor introduce el cebador foward.<br><br>- **"reverse"** es el campo donde el actor introduce el cebador reverse. |

| | |
|---|---|
| | -**"Lmin"** y **"Lmax"** es el campo donde se especifican valores enteros positivos definidos por el actor. |
| 2. El actor proporciona los datos llenando todos los campos del formulario correspondiente y pulsa en el botón ¨**enviar**¨**.** | 3. Verifica que los campos obligatorios contengan un valor.<br>• Especie.<br>• Familia<br>• foward<br>• reverse<br>• Lmin y Lmax |
| | 4. Inserta la pareja de cebadores. |
| | |
| **Flujos alternos Sección "Insertar nueva pareja de cebadores"** | |
| 2.1) El actor no desea proporcionar los datos para la inserción de una nueva pareja de cebadores y sale de la sección. | |
| | 3.1) Si falta algún dato necesario para la inserción de la nueva pareja de cebadores, señala dicho(s) campos(s). |
| | |

| **Sección "**Editar cebadores**"** | |
|---|---|
|  | |
| **Acción del actor** | **Respuesta del Sistema** |
| | 1.Muestra la interfaz correspondiente para editar una pareja de cebadores, mostrando los siguientes campos:<br><br>- **"Especie"** es el campo que brinda la posibilidad de escoger el reino al que pertenece el cebador a insertar (Bacteria, Hongo, Protozoo, Archaea o Indefinido si no se sabe el reino al que pertenece.<br><br>- **"Familia"** es el campo que brinda la posibilidad de introducir la familia a la que pertenece el cebador a insertar.<br><br>- **"foward"** es el campo donde el actor introduce el cebador foward.<br><br>- **"reverse"** es el campo donde el actor introduce el cebador reverse.<br><br>-**"Lmin"** y **"Lmax"** es el campo donde se |

| | |
|---|---|
| | especifican valores enteros positivos definidos por el actor. |
| 2. El actor llena los campos con los valores que desea modificar y pulsa en el botón **"enviar".** | 3. Realiza las modificaciones. |
| | |
| **Flujos alternos Sección "Editar cebadores"** | |
| 2.1. El actor no desea modificar la pareja de cebadores y sale de la sección. | |
| | |
| **Sección "Eliminar una pareja de cebadores"**. | |

| Especie | Familia | Lmin | Lmax | Acciones |
|---|---|---|---|---|
| Bacteria | Prevotella ruminicola | 646 | 1058 | |
| Bacteria | Prevotella bryantii | 676 | 1156 | |
| Bacteria | Prevotella albensis | 187 | 1027 | |

| Acción del actor | Respuesta del Sistema |
|---|---|
| 1. El actor decide eliminar una pareja de cebadores y pulsa en el vinculo ¨**Editar cebadores**¨ | 2. Muestra un listado con las parejas de cebadores existentes en la base de datos brindando las opciones de ver, editar o eliminar la pareja de cebadores que el actor decida. |

| | |
|---|---|
| 3. El actor pulsa en ¨**eliminar**¨ en dependencia de la pareja de cebadores que desea remover de la base de datos. | 4. Se elimina la pareja de cebadores de la base de datos. |
| **Flujos alternos Sección "Eliminar una pareja de cebadores"**. | |
| 3.1) El actor no desea eliminar una pareja de cebadores y sale de la sección. | |
| | |

**CUS Gestionar usuario.**

| | |
|---|---|
| **Nombre del Caso de Uso** | Gestionar usuario. |
| **Actores** | Administrador. |
| **Propósito** | Brinda la posibilidad de insertar un nuevo usuario en la base de datos, así como modificar, ver y eliminar uno ya existente. |
| **Resumen** | El caso de uso se inicia cuando el actor va a realizar alguna de las siguiente operaciones :<br>● Insertar usuario.<br>● Modificar usuario.<br>● Eliminar usuario.<br>El sistema le muestra la interfaz correspondiente. El caso de uso finaliza cuando se emite el resultado de la operación solicitada. |
| **Referencias** | R6,R6.1,R6.2,R6.3,R6.4,R8 |

<table>
<tr><td>Precondiciones</td><td colspan="2">Que el actor se haya autenticado en la aplicación.</td></tr>
<tr><td>Pos condiciones</td><td colspan="2">El sistema inserta un nuevo usuario, así como elimina y modifica los datos de uno existente.</td></tr>
<tr><td colspan="3">Flujo normal de los eventos</td></tr>
<tr><td>Acción del actor</td><td colspan="2">Respuesta del Sistema</td></tr>
<tr><td>1. El actor quiere realizar una de las siguientes operaciones:<ul><li>Insertar usuario.</li><li>Modificar usuario.</li><li>Eliminar usuario.</li></ul></td><td colspan="2">2.En dependencia de la operación solicitada:<ul><li>Si decide insertar un nuevo usuario, ir a Sección “Insertar usuario”.</li><li>Si desea modificar un usuario, ir a Sección “Editar usuario”.</li><li>Si decide eliminar un usuario, ir a la Sección “Eliminar usuario”.</li></ul></td></tr>
<tr><td colspan="3">Sección “Insertar usuario”</td></tr>
<tr><td colspan="3"> 
</td></tr>
<tr><td>Acción del actor</td><td colspan="2">Respuesta del Sistema</td></tr>
<tr><td></td><td colspan="2">1.Muestra la interfaz correspondiente para la inserción de un usuario con los siguientes campos:</td></tr>
</table>

| | |
|---|---|
| | - **"Tipo de Usuario"** es el campo que brinda la posibilidad de escoger el rol que desempeñará el usuario en la aplicación.<br><br>- **"Nombre de usuario"** es el campo que brinda la posibilidad de introducir el nombre del usuario a insertar.<br><br>- **"Usuario"** es el campo donde el actor introduce el usuario con el cual el Usuario podrá acceder a la aplicación.<br><br>- **"Contraseña"** es el campo donde el actor introduce la contraseña del usuario.<br><br>-**"Confirmar Contraseña"** es el campo donde se confirma la contraseña del usuario. |
| 2. El actor llena los campos y pulsa en el botón ¨**enviar**¨**.** | 3. Verifica que los campos obligatorios contengan un valor.<br>• **"Tipo de Usuario"**<br>• **"Nombre de usuario"**<br>• **"Usuario"**<br>• **"Contraseña"**<br>• **"Confirmar Contraseña"** |
| | 4. Registra el usuario en la base de datos. |
| | |
| **Flujos alternos Sección "Insertar usuario"** | |

| | |
|---|---|
| 2.1) El actor no desea llenar los campos y sale de la sección. | |
| | 3.1) Si falta algún dato necesario para la inserción de un usuario, señala dicho(s) campos(s). |
| | |

| **Sección "Editar usuario"** | |
|---|---|
|  | |
| **Acción del actor** | **Respuesta del Sistema** |
| | 1.Muestra la interfaz correspondiente para editar un usuario con los siguientes campos:<br><br>- **"Tipo de Usuario"** es el campo que brinda la posibilidad de escoger el rol que desempeñará el usuario en la aplicación.<br><br>- **"Nombre de usuario"** es el campo que brinda la posibilidad de introducir el nombre del usuario a insertar.<br><br>- **"Usuario"** es el campo donde el actor introduce |

| | |
|---|---|
| | el usuario con el cual el Usuario podrá acceder a la aplicación.<br><br>- **"Contraseña"** es el campo donde el actor introduce la contraseña del usuario.<br><br>-**"Confirmar Contraseña"** es el campo donde se confirma la contraseña del usuario. |
| 2. El actor llena los campos con los valores que desea modificar y pulsa en el botón **"enviar".** | 3. Realiza las modificaciones. |
| | |
| **Flujos alternos Sección "Editar cebadores"** | |
| 2.1. El actor no desea modificar el usuario y sale de la sección. | |
| | |
| **Sección "Eliminar usuario"**. | |

| Nombre | Usuario | Rol | Acciones |
|---|---|---|---|
| rachid | rachid | administrador | ✓ \| ✗ |
| yumis | yumis | investigador | ✓ \| ✗ |

| Acción del actor | Respuesta del Sistema |
|---|---|
| 1. El actor decide eliminar un usuario y pulsa en el vinculo ¨**Editar usuario**¨ | 2. Muestra un listado con los usuarios existentes en la base de datos brindando |

| | |
|---|---|
| | las opciones de ver, editar o eliminar el usuario que el actor decida. |
| 3. El actor pulsa en ¨**eliminar**¨ en dependencia del usuario que desea remover de la base de datos. | 4. Se elimina el usuario de la base de datos. |
| | |
| **Flujos alternos Sección "Eliminar usuario"**. | |
| 3.1) El actor no desea eliminar un usuario y sale de la sección. | |
| | |

**Anexo 2** Descripciones de clases del diseño.

## CU Realizar reconstrucción filogenética.

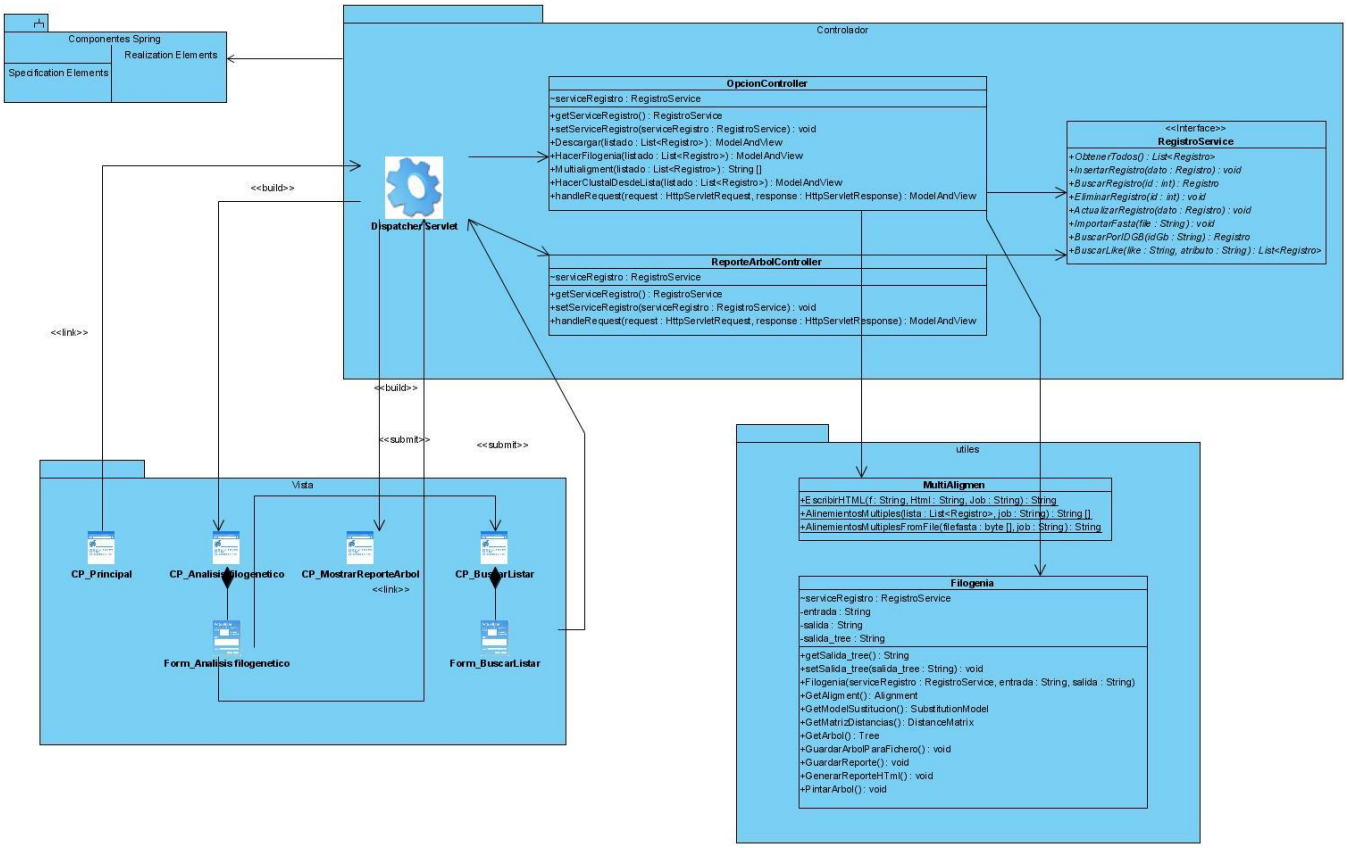



**Figura 20:** Diagrama de clases del diseño CU Realizar reconstrucción filogenética.

## CU Realizar BH usando BLAST-N.

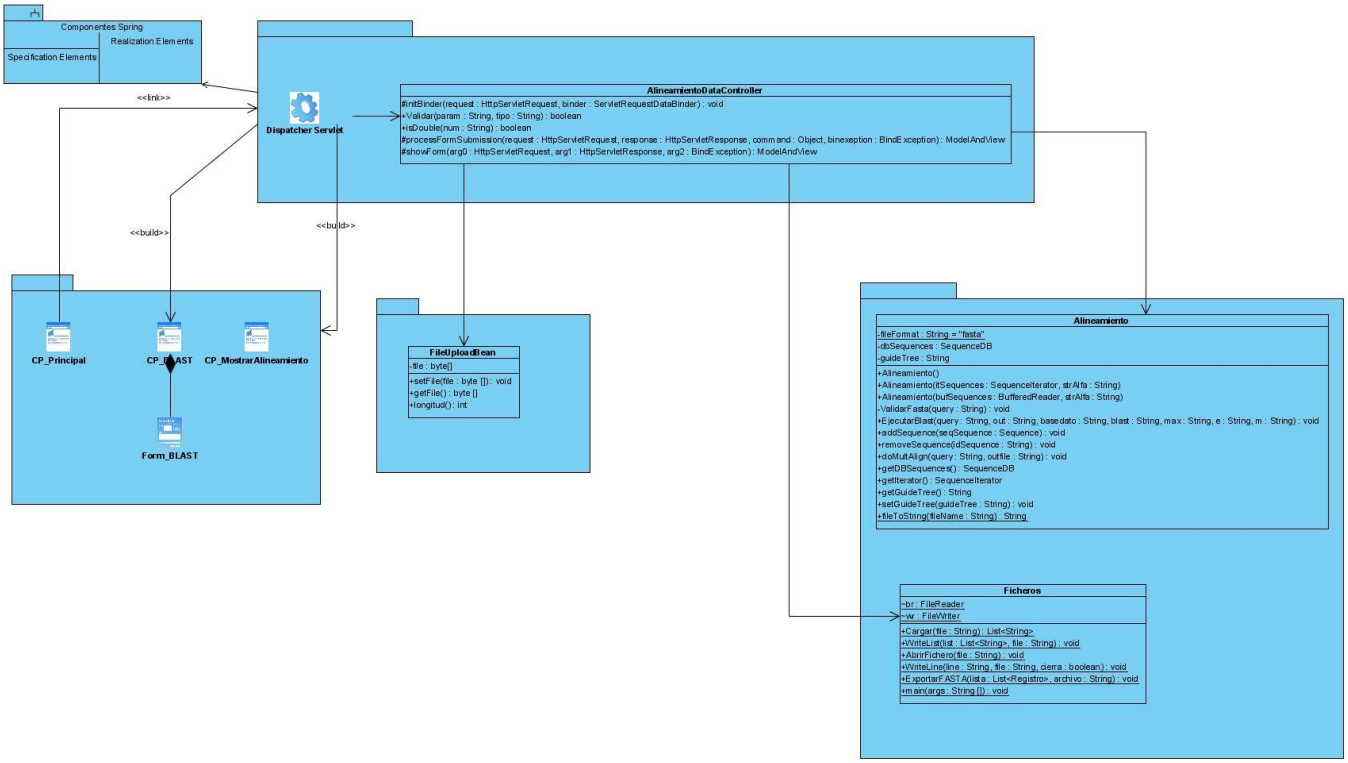



**Figura 21:** Diagrama de clases del diseño CU Realizar BH usando BLAST-N.

## CU Realizar AMS usando CLUSTALW.

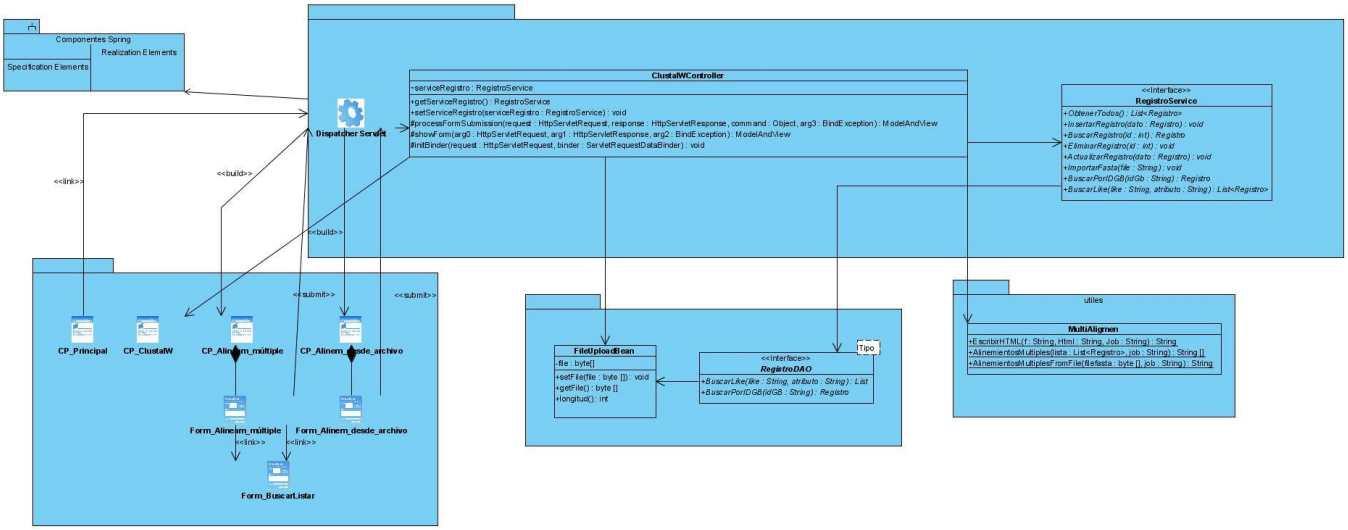



**Figura 22:** Diagrama de clases del diseño CU Realizar AMS usando CLUSTALW.

**Anexo 3** Diagramas de secuencia.

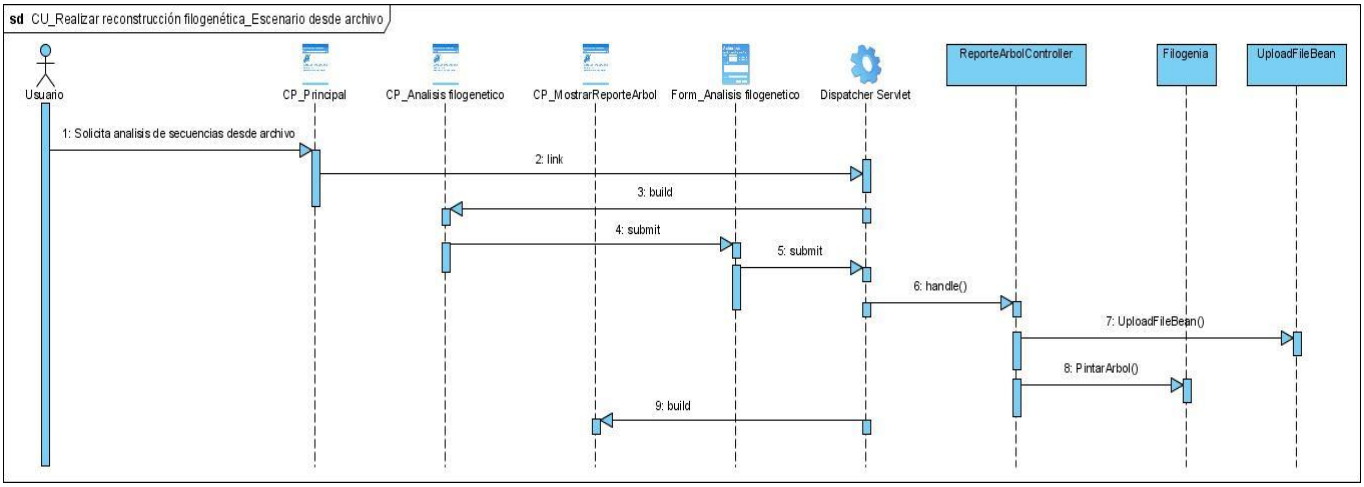



**Figura 23**: Diagrama de secuencia de CU Realizar reconstrucción filogenética escenario desde archivo.

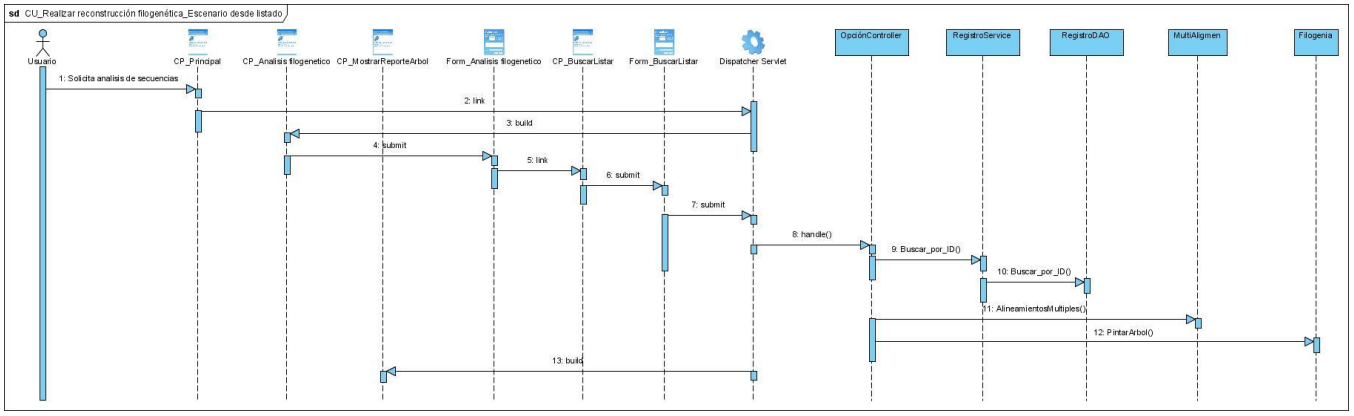



**Figura 24**: Diagrama de secuencia de CU Realizar reconstrucción filogenética escenario desde listado.

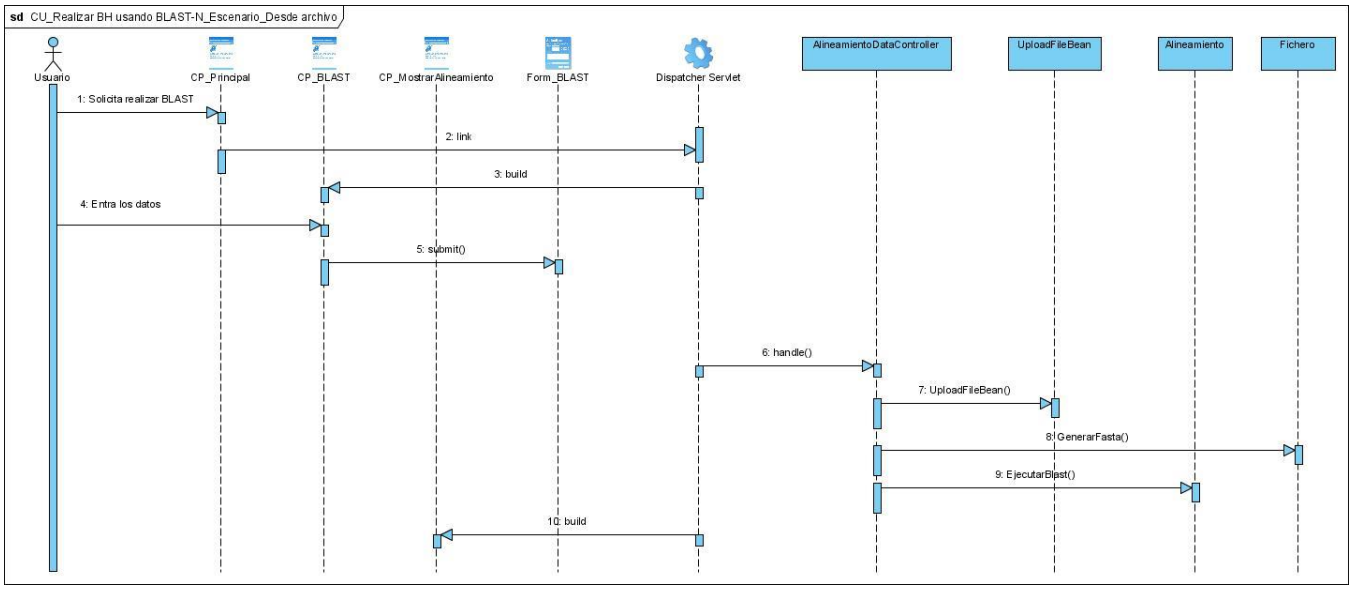



**Figura 25:** Diagrama de secuencia de CU Realizar BH usando BLAST-N Escenario desde archivo.

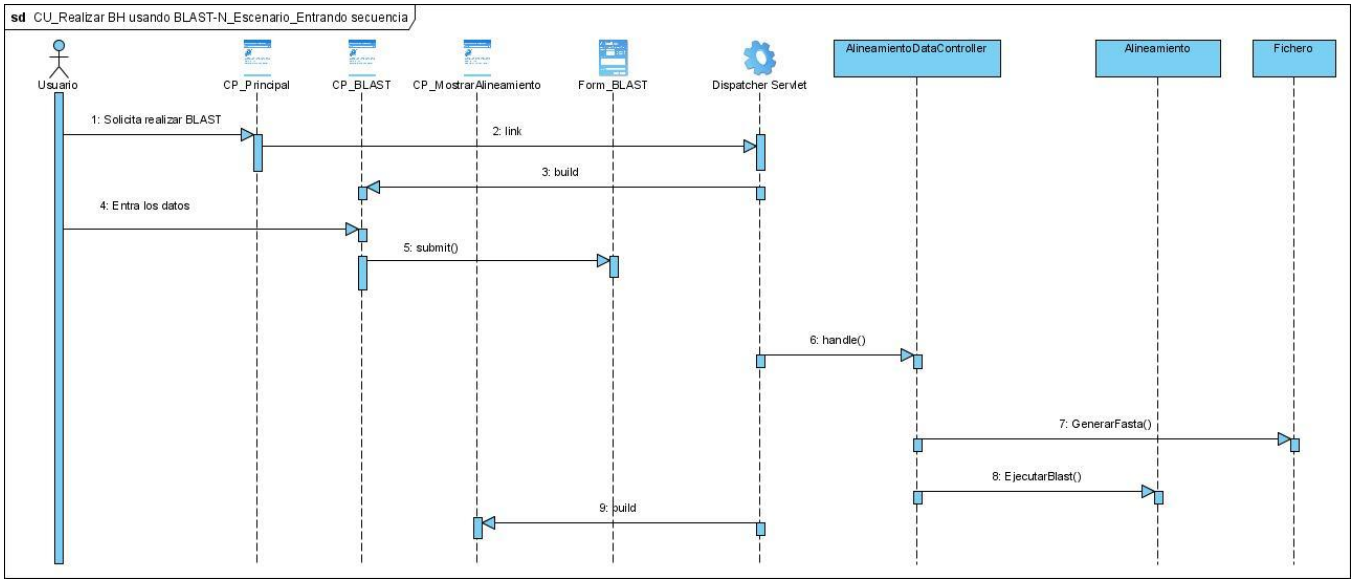



**Figura 26**: Diagrama de secuencia de CU Realizar BH usando BLAST-N Escenario introduciendo una secuencia.

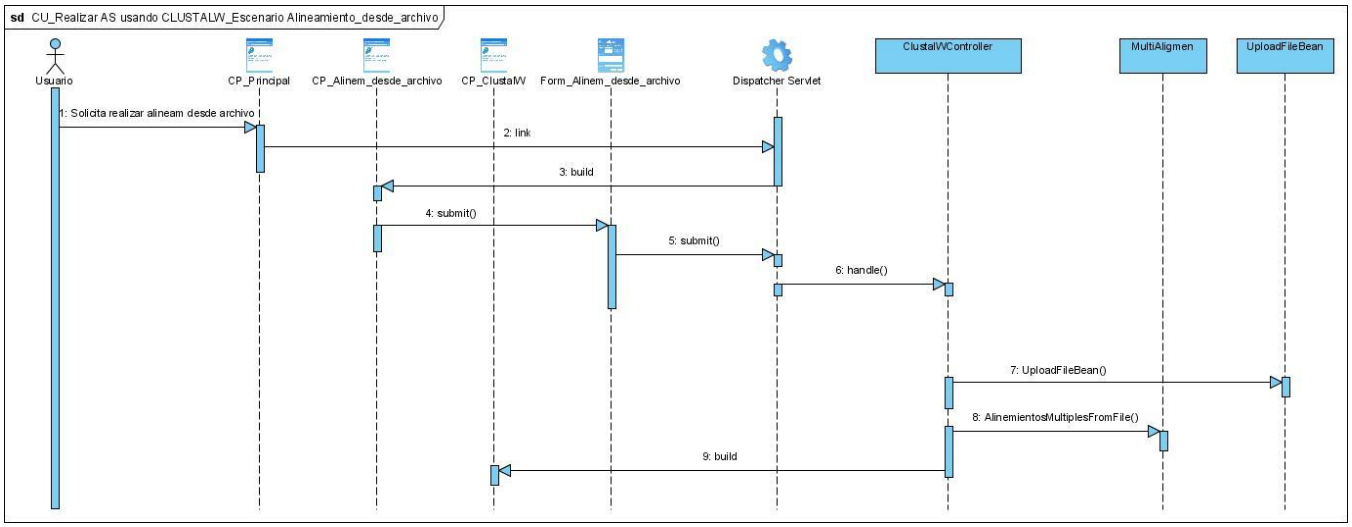



**Figura 27**: Diagrama de secuencia de CU Realizar AS usando CLUSTALW Escenario alineamiento desde archivo.

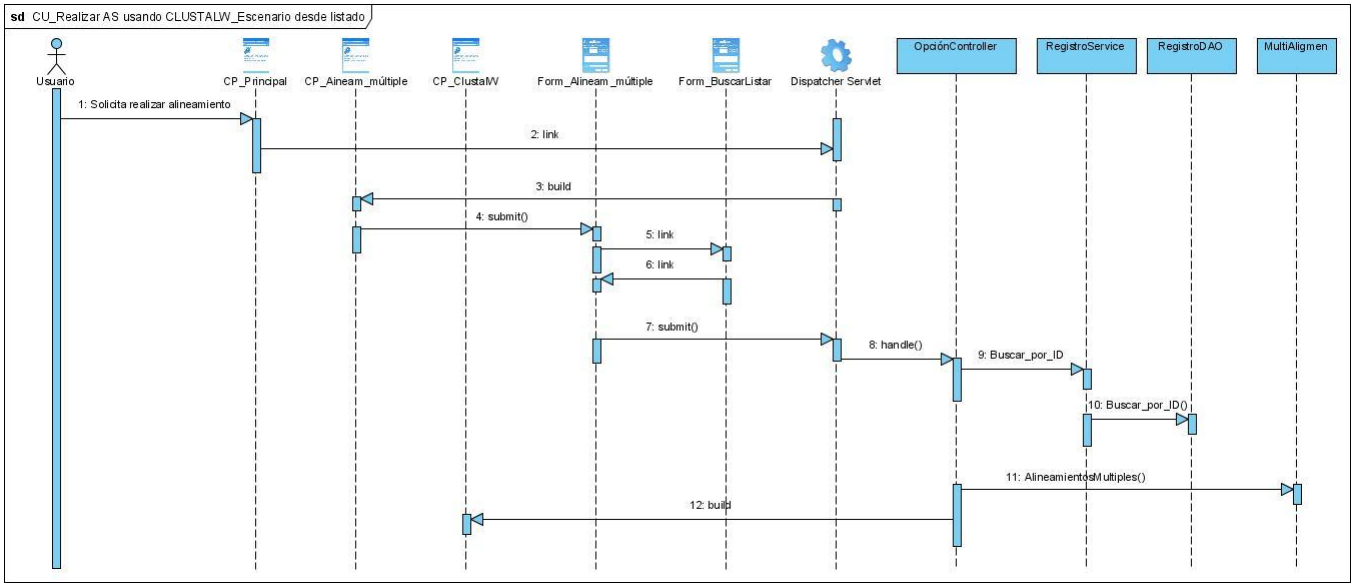
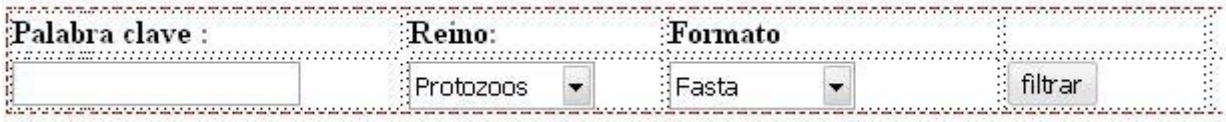



**Figura 28**: Diagrama de secuencia de CU Realizar AS usando CLUSTALW Escenario alineamiento desde listado.

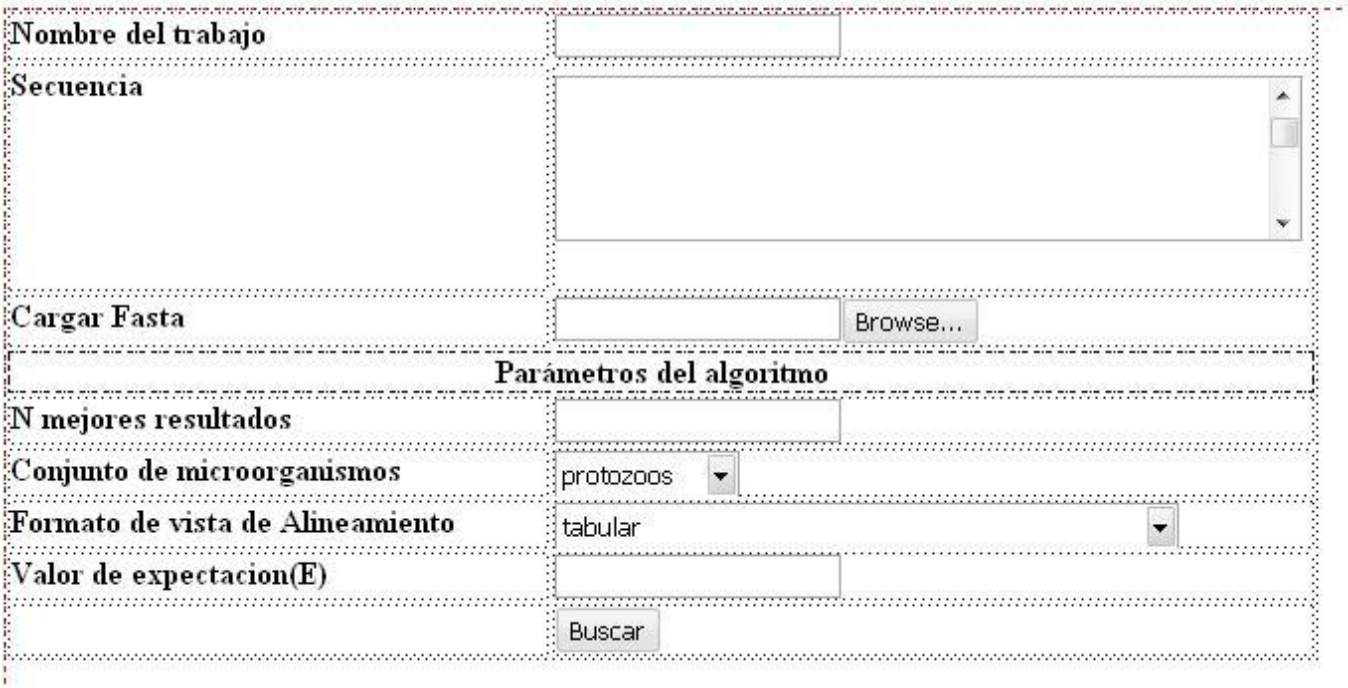
**Anexo 4** Prototipos no funcionales de interfaz de usuario.

**Prototipo de interfaz de usuario Buscar registro.**

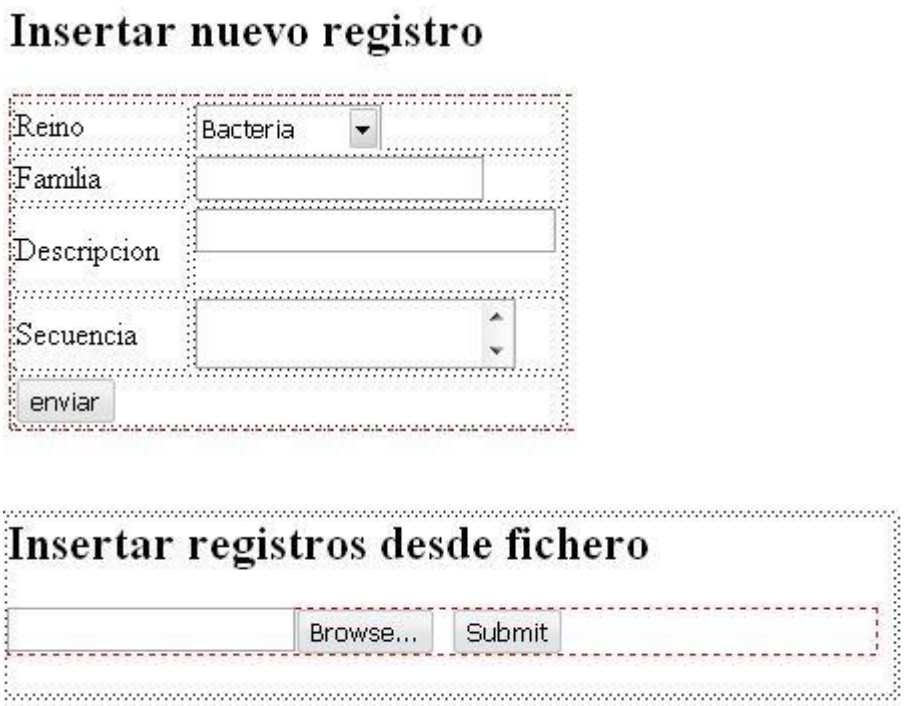



**Prototipo de interfaz de usuario Análisis filogenético.**


**Prototipo de interfaz de usuario para realizar BLAST.**

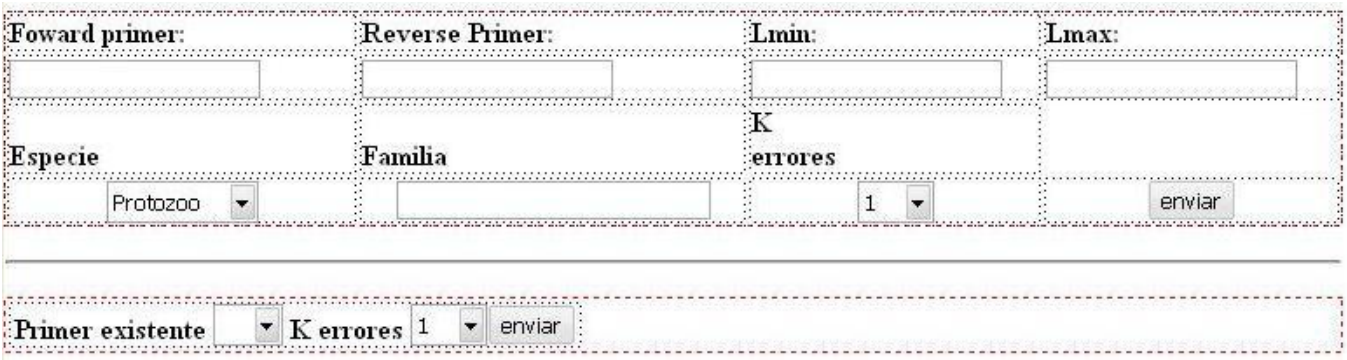


**Prototipo de interfaz de usuario para insertar un nuevo registro.**




**Prototipo de interfaz de usuario para insertar una pareja de cebadores.**

**Prototipo de interfaz de usuario para el macheo de cebadores.**



**Anexo 5**: Mapa de navegación

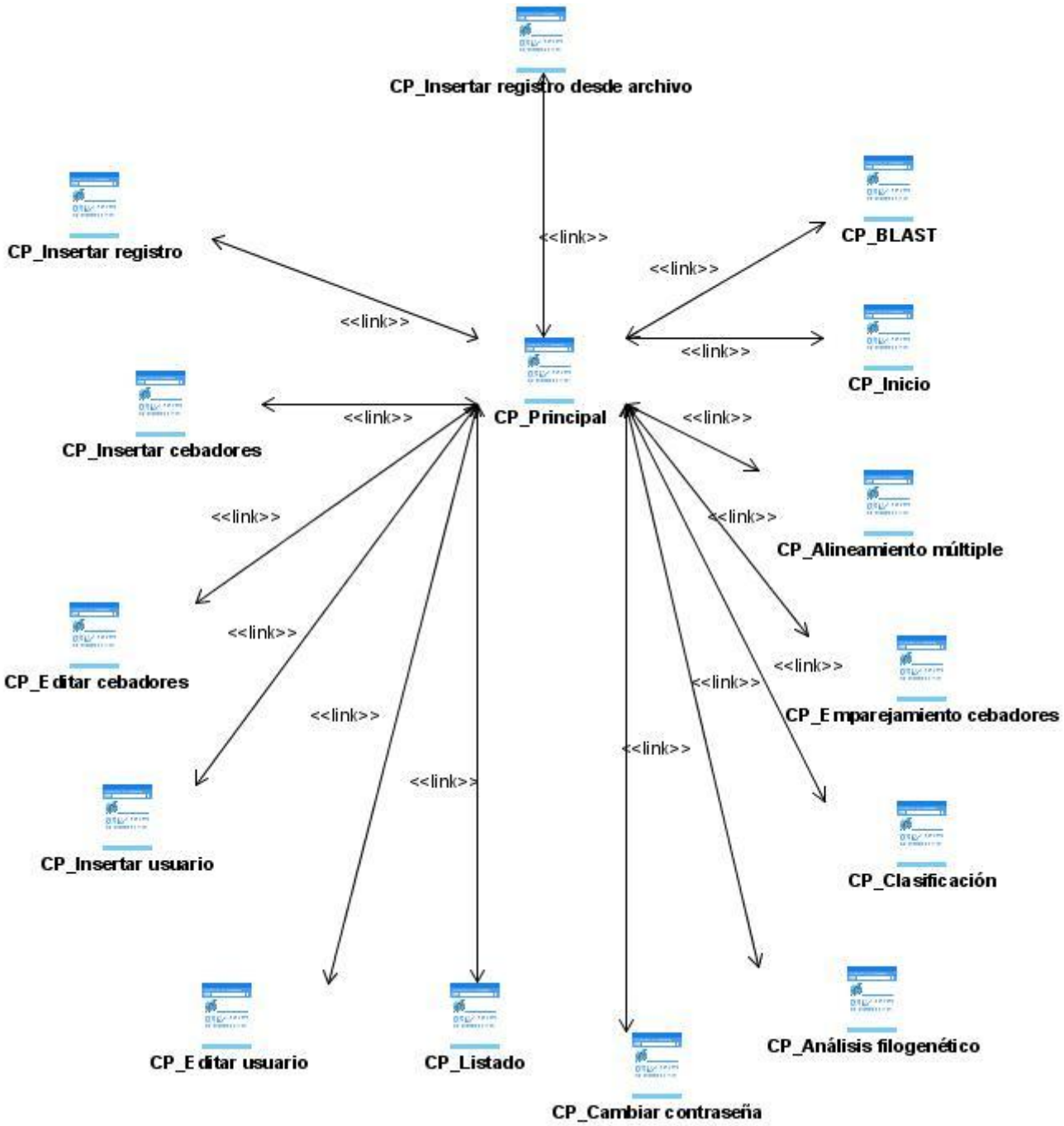



**Figura 29:** Mapa de navegación.

**Anexo 6** Diagramas de componentes

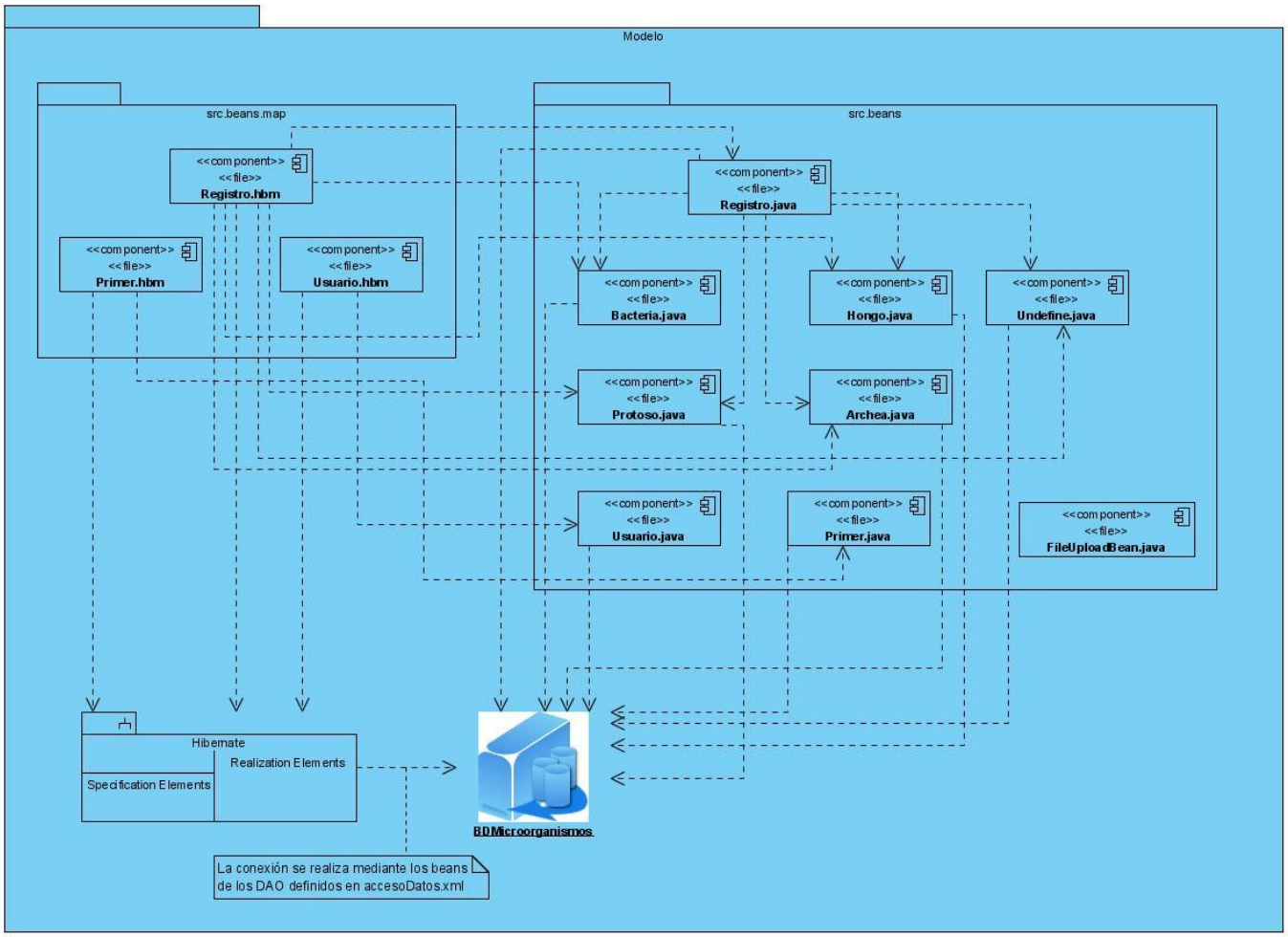



**Figura 30:** Diagrama de componentes: paquete Modelo.

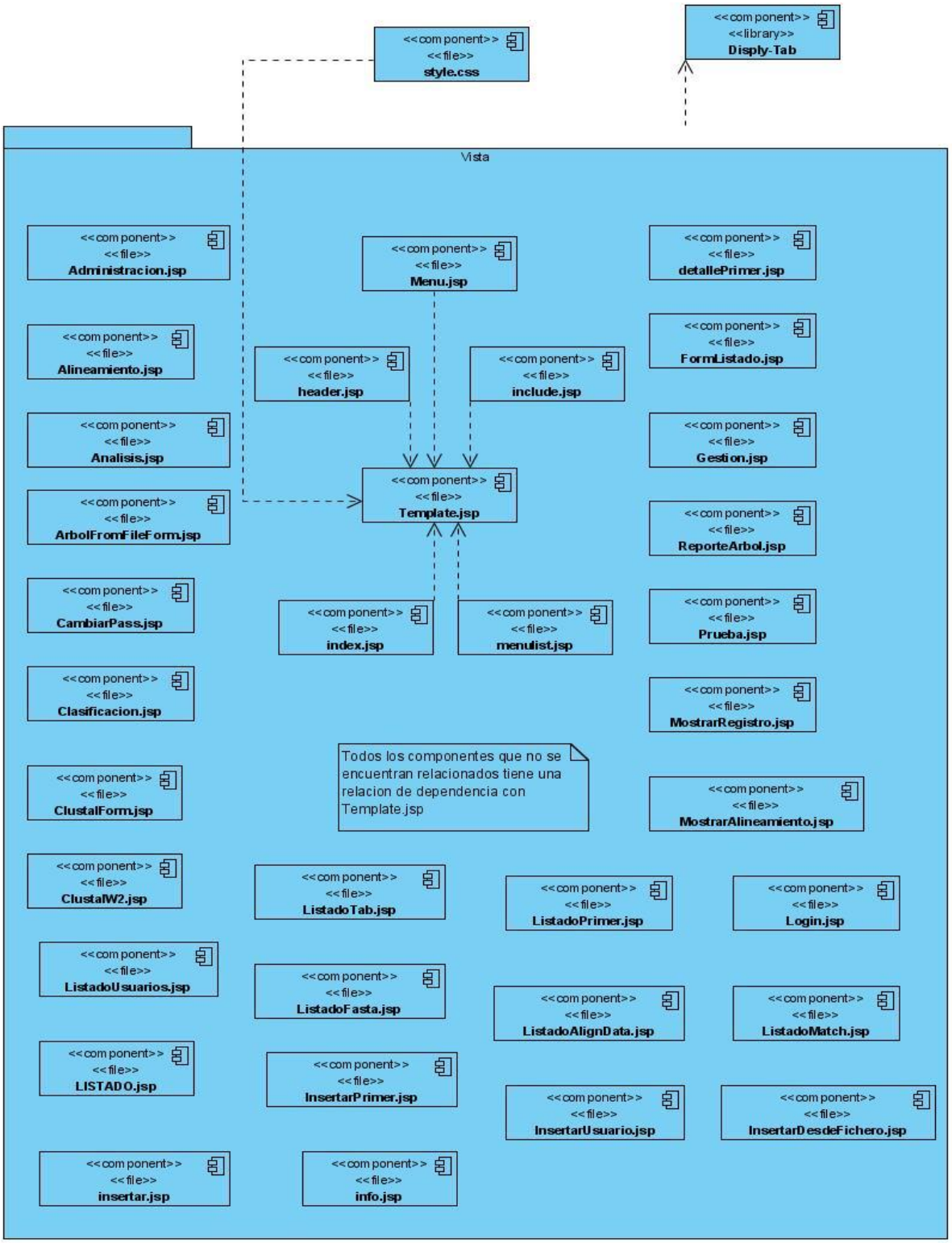



**Figura 31:** Diagrama de componentes: paquete Vista.

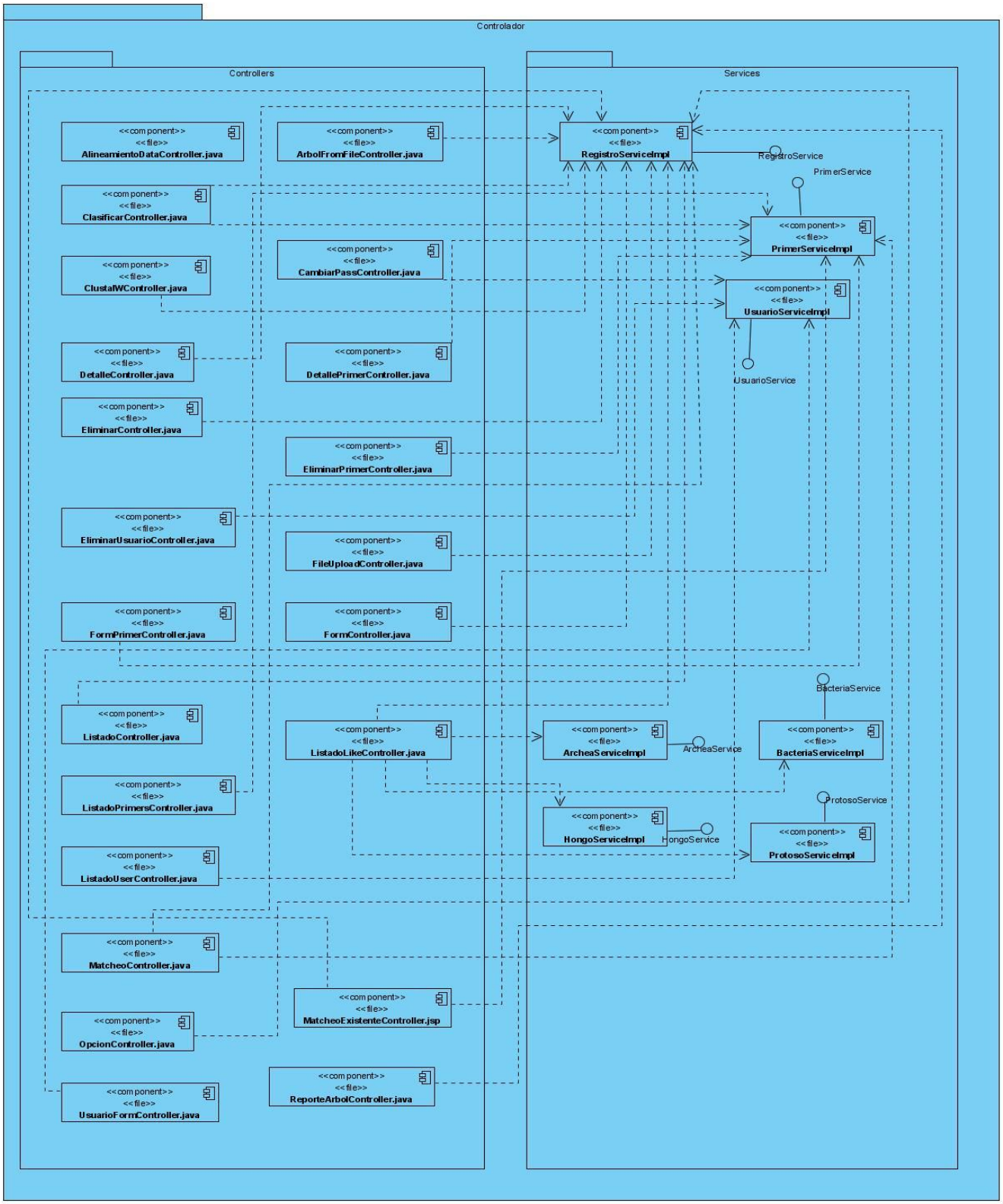



**Figura 32:** Diagrama de componentes: paquete Controlador.

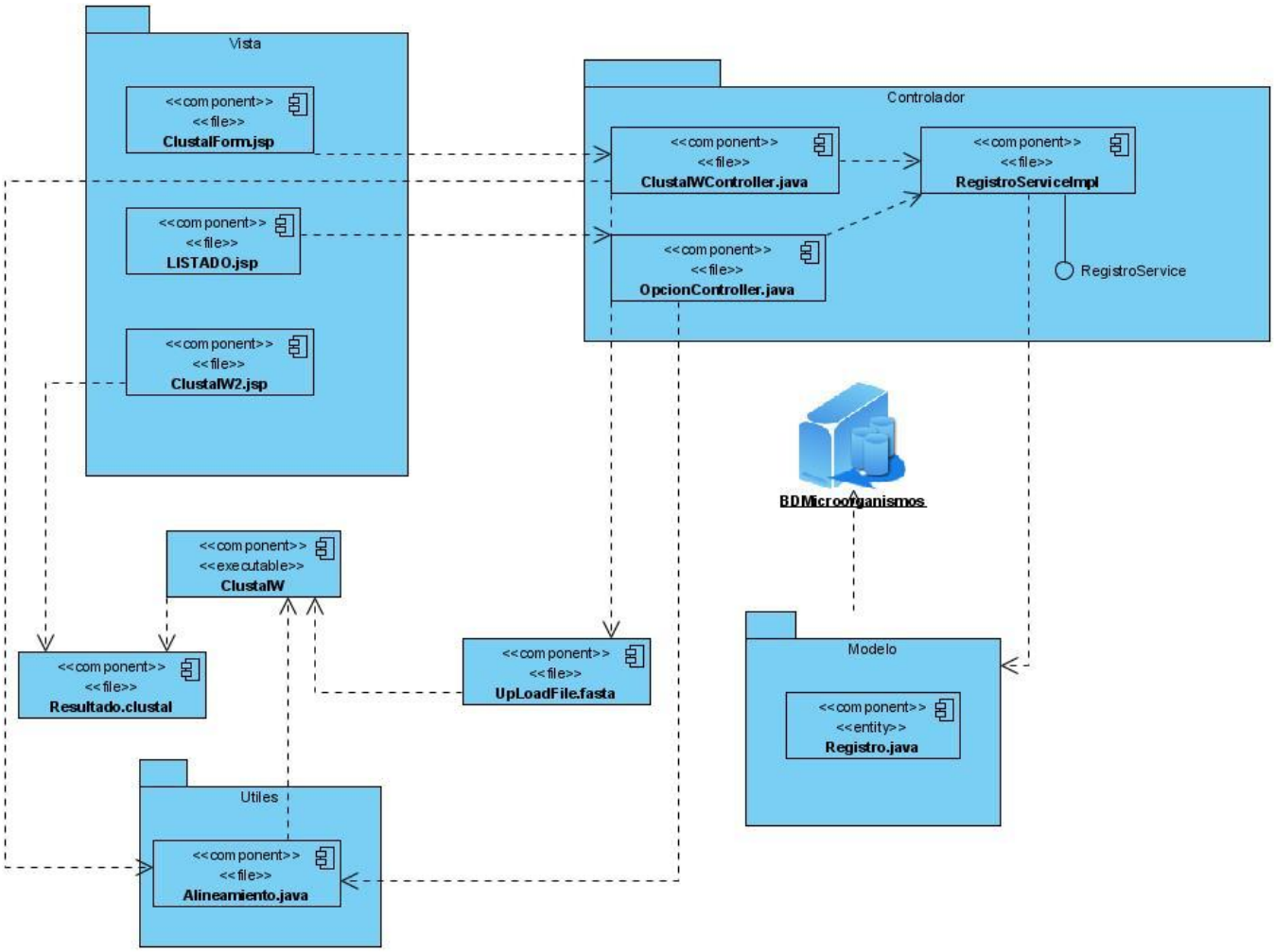



**Figura 33**: Diagrama de componentes: CU Realizar AS usando CLUSTALW.

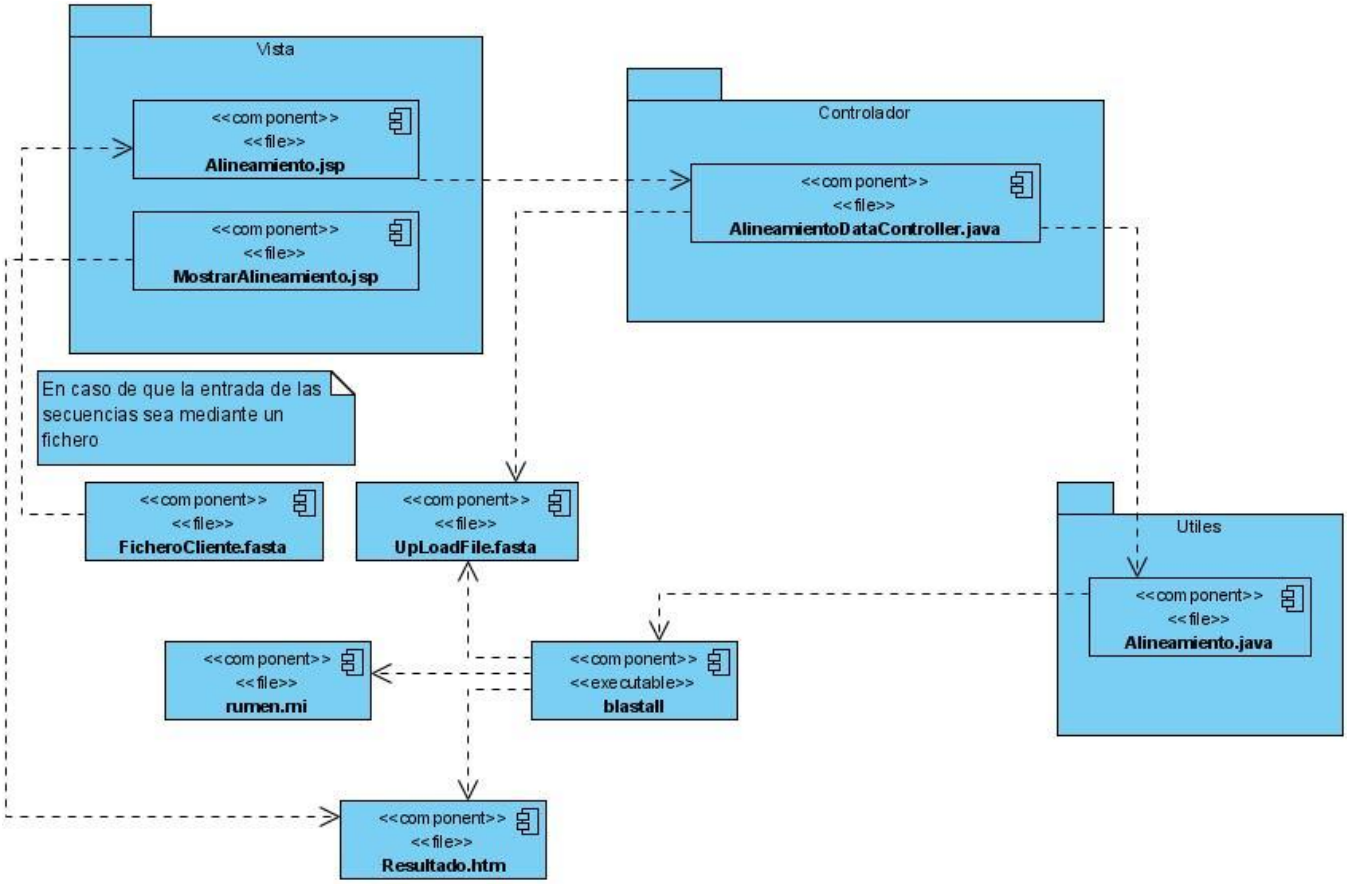



**Figura 34**: Diagrama de componentes: CU Realizar BH usando BLAST-N.

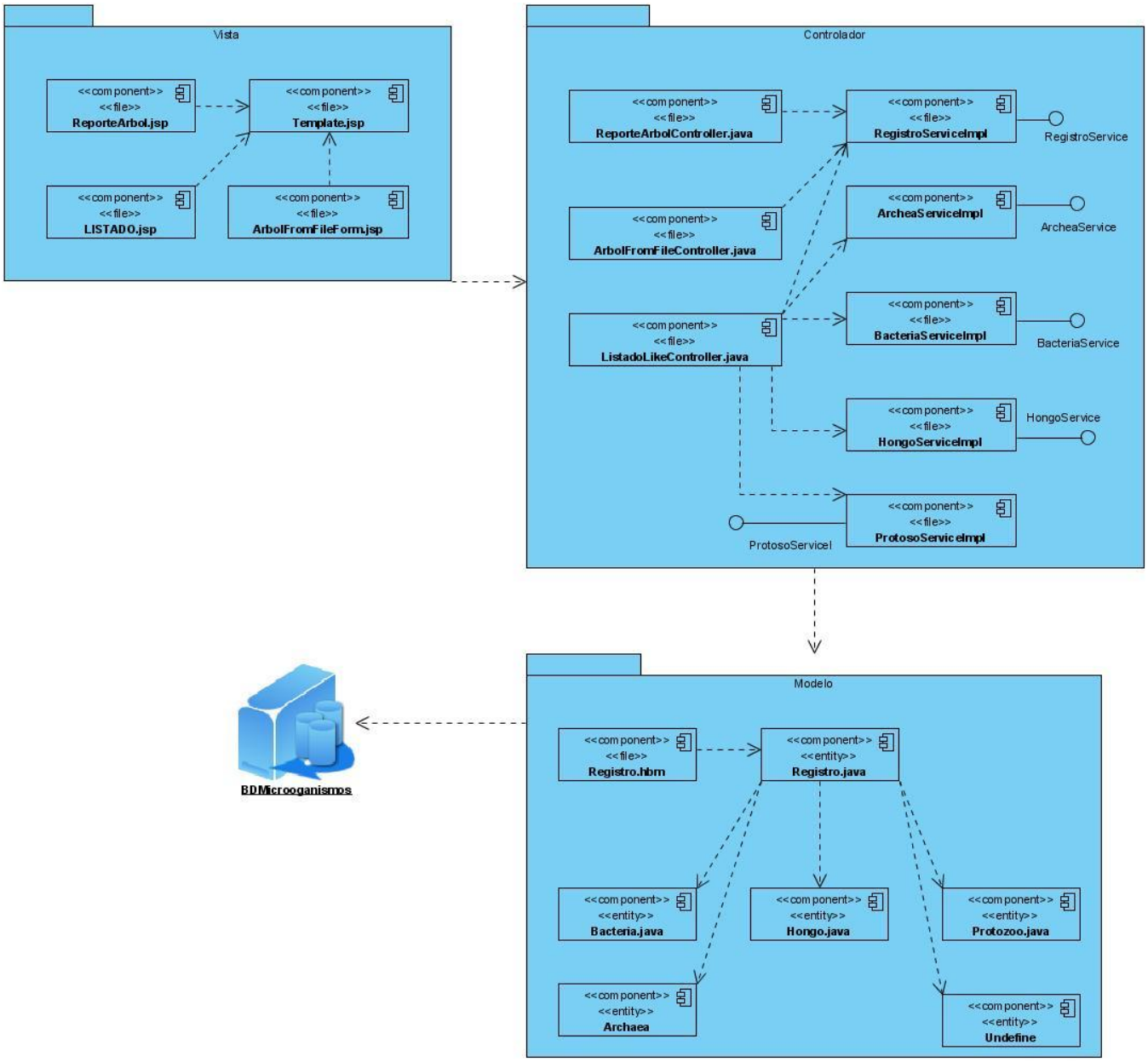



**Figura 35**: Diagrama de componentes: CU Realizar reconstrucción filogenética.

**Anexo 7:** Casos de prueba.

| **Caso de uso** | BH usando BLAST-N. |
|---|---|
| Caso de prueba | BH usando BLAST-N. (Introduciendo una secuencia blanco). |
| Entrada | El usuario introduce el nombre del trabajo, la secuencia blanco con la cual se realizará la búsqueda por homología, la base de datos contra quien se realizará esa búsqueda, la cantidad de resultados y el formato que el usuario desea que el sistema muestre, así como el valor de expectación. |
| Resultado esperado | Obtener un listado con los N resultados significativos que el usuario especificó de las secuencias homólogas a la secuencia blanco introducido. |
| Resultado de la prueba | El sistema muestra un listado con las secuencias homólogas a la secuencia blanco introducida. |
| Condiciones | La secuencia blanco entrada no puede contener números. |

| **Caso de uso** | BH usando BLAST-N. |
|---|---|
| Caso de prueba | BH usando BLAST-N. (Desde archivo). |
| Entrada | El usuario introduce el nombre del trabajo, importa el archivo de secuencias, selecciona la base de datos contra quien se realizará esa búsqueda por homología, la cantidad de resultados y el formato que el usuario desea que el sistema muestre, así como el valor de expectación. |

| | |
|---|---|
| Resultado esperado | Obtener un listado con los N resultados significativos que el usuario especificó de las secuencias homólogas a las secuencias importadas en el archivo. |
| Resultado de la prueba | El sistema muestra un listado con las secuencias homólogas a la secuencia importada en el archivo. |
| Condiciones | El archivo debe estar en FASTA o Clustal. |

| **Caso de uso** | Realizar AMS usando CLUSTALW. |
|---|---|
| Caso de prueba | Realizar AMS usando CLUSTALW (Desde listado). |
| Entrada | El usuario realiza una búsqueda de las secuencias de interés para realizar el alineamiento múltiple, las selecciona y escoge la opción ClustalW |
| Resultado esperado | Obtener el alineamiento múltiple de las secuencias seleccionadas. |
| Resultado de la prueba | El sistema muestra el alineamiento múltiple de las secuencias seleccionadas brindando las opciones de descargar ClustalW, descargar FASTA así como el reporte del árbol filogenético. |
| Condiciones | Debe seleccionar como mínimo dos secuencias para realizar el alineamiento múltiple. |

| | |
|---|---|
| **Caso de uso** | Realizar AMS usando CLUSTALW. |
| Caso de prueba | Realizar AMS usando CLUSTALW (Desde archivo). |
| Entrada | El usuario introduce el nombre del trabajo e importa el archivo para realizar el alineamiento. |
| Resultado esperado | Obtener el alineamiento múltiple de las secuencias del archivo. |
| Resultado de la prueba | El sistema muestra el alineamiento múltiple de las secuencias del archivo brindando las opciones de descargar ClustalW, descargar FASTA así como el reporte del árbol filogenético. |
| Condiciones | El formato del archivo tiene que ser FASTA. |

**Anexo 8:** La tabla muestra las 11 parejas de cebadores utilizadas, así como el número de secuencias que contienen el gen flanqueado por la pareja de cebadores correspondiente, que a su vez constituyen la base de datos patrón (BDP).

| **Pareja de cebadores por familia** | **BDP** | **Cebador "Foward"** | **Cebador "Reverse"** |
|---|---|---|---|
| *Fibrobacter Succinogenes* | 28 | GGTATGGGATGAGCTTGC | GCCTGCCCCTGAACTATC |
| *Prevotella Ruminicola* | 8 | GGTTATCTTGAGTGAGTT | CTGATGGCAACTAAAGAA |
| *Prevotella Bryantii* | 2 | ACTGCAGCGCGAACTGTCAGA | ACCTTACGGTGGCAGTGTCTC |
| *Prevotella Albensis* | 3 | CAGACGGCATCAGACGAGGAG | ATGCAGCACCTTCACAGGAGC |
| *Ruminobacter Aminophilus* | 4 | CAACCAGTCGCATTCAGA | CACTACTCATGGCAACAT |
| *Selenomonas Ruminatum* | 23 | TGCTAATACCGAATGTTG | TCCTGCACTCAAGAAAGA |
| *Streptococus Bovis* | 10 | CTAATACCGCATAACAGCAT | AGAAACTTCCTATCTCTAGG |
| *Eubacterium Ruminatum* | 1 | GCTTCTGAAGAATCATTTGAAG | TCGTGCCTCAGTGTCAGTGT |
| *Ruminucocus Flavefaciens* | 5 | GGACGATAATGACGGTACTT | GCAATCGAACTGGGACAAT |
| *Anaerovibrio Lypolítica* | 1 | TGGGTGTTAGAAATGGATTC | CTCTCCTGCACTCAAGAATT |
| *Treponema Bryantii* | 1 | AGTCGAGCGGTAAGATTG | CAAAGCGTTTCTCTCACT |

## Glosario de términos

***ARN ribosomal 16S*[1]:** gen más común que se puede usar para construir relaciones filogenéticas en procariotas.

**Rumen[2]:** Primer compartimento del estómago de los animales rumiantes, y el de mayor tamaño. Allí el alimento es fermentado por microorganismos (bacterias, protozoos y hongos) anaeróbicos que pueden utilizar la fibra (especialmente celulosa) para obtener energía.

**Cebador[3]:** es una secuencia corta de ácido nucleico que contiene un grupo 3'hidroxilo libre que forma pares de bases con una hebra molde complementaria y actúa como punto de inicio para la adición de nucleótidos con del fin de copiar la hebra molde. Se necesitan dos para la reacción de PCR. Uno en el extremo 3' y el otro complementario para la otra hebra. Son de aproximadamente 20 nucleótidos porque es la cantidad necesaria para que probabilísticamente coincida en un sitio de la cadena de DNA.

**Grupo taxonómico[4]:** Una de las categorías que se utilizan en la clasificación, como Orden, Familia, Género o Especie. (Ver clasificación, taxonomía).

**Relaciones tróficas[5]**: Son las relaciones que se establecen entre los seres vivos en función de su alimento. Están directamente relacionadas a la cadena alimentaria de los organismos involucrados.

**Electroforesis[6]:** es un método de laboratorio en el que se utiliza una corriente eléctrica controlada con la finalidad de separar biomoléculas según su tamaño y carga eléctrica a través de una matriz gelatinosa. Fue empleado por primera vez por  en el año 1937, pero su importancia vino a incrementarse cuando en los años cincuenta E. L.Durrum y Arne W.K. Tiselius , impulsaron la electroforesis de zona, nombre que se asigno a la separación de materiales en un campo eléctrico en presencia de algún tipo de soporte; aunque este término se limitó originalmente al análisis de coloides y partículas submicroscopicas , se ha convertido en estos últimos años en una metodología aplicada a sustancias de bajo peso molecular.

**PCR[7]**: es una técnica que permite duplicar un número ilimitado de veces un fragmento de ADN en un tubo de ensayo. Mediante esta técnica pueden generarse millones de moléculas idénticas, a partir de una molécula de ADN. Esto se puede conseguir en unas horas.

**Genoteca[8]:** es un banco de genes que almacena una colección de genes clasificados y preparados para su utilización en experimentación. Las genotecas permiten disponer en cualquier momento de secuencias de ADN de posible interés biomédico.

**Microarrays[9]:** es una superficie sólida a la cual se unen una serie de fragmentos de ADN. Las superficies empleadas para fijar el ADN son muy variables y pueden ser vidrio, plástico e incluso chips de silicio. Los arreglos de ADN son utilizados para averiguar la expresión de genes, monitorizándose los niveles de miles de ellos de forma simultánea.