Proyecto Fin de Carrera
Ingeniería Informática

Entorno de visualización y edición de árboles para filogenias extensas

Autor

Samuel Guajardo Esteban

Directores

Gregorio de Miguel Casado
Jorge Álvarez Jarreta

Área de Lenguajes de Sistemas Informáticos
Departamento de Informática e Ingeniería de Sistemas
ESCUELA DE INGENIERÍA Y ARQUITECTURA
Universidad de Zaragoza

Junio-2014

Resumen

Este proyecto de fin de carrera aborda el desarrollo de un sistema de visualización web con el que podamos ver y editar árboles y que además permite acceder rápidamente a la información que contienen. Como caso particular se trata la visualización de árboles con información de tipo biológico.

En la actualidad para la visualización de estos árboles no existen herramientas que satisfagan las necesidades de determinados usuarios cuando el tamaño del árbol a visualizar es muy grande o se requieren funcionalidades especiales de búsqueda. Éstas tienen como problema fundamental que al intentar mostrar el árbol completo se colapsan, debido al gran tamaño de la entrada. Además no disponen de mecanismos para encontrar información específica adicional sobre los propios nodos del árbol. En este contexto, existen páginas web estáticas que mantienen información sobre este tipo de árboles pero no se actualizan con la frecuencia deseable.

La novedad esencial que aporta el proyecto es la interfaz de visualización del árbol, que no se centra en la visualización del árbol entero, ya que solo nos mostrara padre-hijos-hermanos de una secuencia, es decir, toda la información que necesita el especialista. Como caso de aplicación se usa el árbol de ZARAMIT. En el entorno cualquier usuario registrado podrá visualizar árboles y además permitirá solicitar duplicados del árbol para trabajar con ellos, permitiendo su edición mediante la inclusión o eliminación de nodos. También podrán solicitar al administrador que añada sus árboles dentro del sistema para poder usarlos y compartirlos con otros usuarios marcándolos como arboles de dominio público.

Los resultados preliminares obtenidos por el sistema son prometedores y se espera que el alcance del trabajo sea amplio.

Índice

1. Introducción **1**
1.1. Motivación 1
1.2. Objetivos 1

2. Estado del arte **5**
2.1. Glosario de términos biológicos 5
2.2. Filogenias extensas 5
2.3. Herramientas de visualización para árboles y redes 5
2.4. Otras tecnologías de visualización 7
2.5. Problemas abiertos 7

3. Arquitectura **9**
3.1. Requisitos derivados del dominio del problema 9
3.2. Requisitos funcionales 10
3.3. Escenario para la propuesta de solución 11

4. Diseño **13**
4.1. Cliente-Servidor 13
4.2. Base de datos 15
4.2.1. Estructura 15
4.2.2. Carga de datos 17
4.2.3. Copia de seguridad 17
4.2.4. Duplicado de árboles 17
4.3. Visualización padres-hijos-hermanos 17

5. Prototipo **21**
5.1. Cliente 22
5.2. Servidor 23
5.3. Base de datos 26
5.4. Interacciones 27

6. Experimentos **29**
6.1. Validación funcional 29
6.2. Pruebas de rendimiento 29

7. Organización del proyecto **31**

8. Conclusiones **33**

9. Bibliografía **35**

ANEXOS **37**

A. Arquitectura Cliente-Servidor **38**

B. Diseño de la base de datos **42**
B.1. Diagrama entidad-relación 42
B.2. Descripción de las Entidades del entorno 43

C. Diagramas UML **48**
C.1. Diagrama de clase 48
C.1.1. Cliente 48
C.1.2. Servidor intermediario 49
C.2. Diagrama de casos de uso 56
C.2.1. Global 56
C.2.2. Parciales 57

D. Manual de usuario **62**
D.1. Requisitos Software 62
D.2. Instalación de la aplicación 62
D.2.1. Base de datos 62
D.2.2. Servidor web y cliente web 62
D.3. Autenticación 63
D.3.1. Invitado 63
D.3.2. Usuario 63
D.4. Seleccionar árbol 63
D.4.1. Selección de árbol como Invitado 64
D.4.2. Selección de árbol como Usuario 64
D.5. Visor de árboles 65
D.5.1. Visualizador de ancestros de grupos 65
D.5.2. Visualizador de secuencias 66
D.5.3. Buscador de secuencias 68
D.5.4. Historial de secuencias 68
D.5.5. Visualizador de árbol de grupos 69

E. Índice de figuras **70**

F. Índice de tablas **72**

1. Introducción

La naturaleza masiva de la información de tipo biológico que se secuencia en los laboratorios y la necesidad de herramientas bioinformáticas para el tratamiento de esta información nos ha motivado para el desarrollo de este proyecto, cuyo objetivo principal es desarrollar un entorno que facilite la visualización e interacción con árboles y redes filogenéticas extensas. Los apartados siguientes desarrollan la motivación de este proyecto y los objetivos concretos que se han abordado.

1.1. Motivación

El incremento constante de la información de tipo biológico que se secuencia en laboratorios (ADN, ARN,...) ha proporcionado la base para el desarrollo de herramientas bioinformáticas que abordan la construcción de árboles y redes que faciliten el estudio de la evolución biológica en las poblaciones de individuos secuenciados.

La naturaleza masiva de la información que se procesa impone requisitos críticos en el desarrollo de software para todas las fases del procesamiento, organización y visualización de resultados. En el caso de la reconstrucción de filogenias y redes evolutivas, el análisis de los resultados (árboles y redes) constituye un problema creciente para los biólogos a medida que la información a analizar va siendo cada vez más grande (ZARAMIT [1], SATé [2],. . .). En este contexto, la carencia de métodos y herramientas adecuados para la visualización, navegación y análisis de los resultados constituyen un campo de trabajo que es objeto de interés para este proyecto.

Este Proyecto de Fin de Carrera se ha realizado en el marco de la línea de investigación de bioinformática del GISED [3], en el que se aborda la construcción de filogenias extensas para secuencias de ADN mitocondrial humano, con el objetivo de estudiar la evolución humana y facilitar la identificación de mutaciones potencialmente malignas. Así, la visualización de los árboles evolutivos inferidos constituye el campo de aplicación concreto.

1.2. Objetivos

El objetivo principal del proyecto es el desarrollo de un entorno que facilite la interacción con árboles y redes filogenéticas extensas, de manera que los usuarios finales del entorno (biólogos, médicos y bioinformáticos) puedan concentrarse en la exploración y análisis de las características biológicas de la información contenida en los árboles o redes. Este objetivo se descompone en otros más concretos, que tienen correspondencia con la propia estructura de esta memoria.

Estado del arte

El apartado correspondiente de la memoria (pág. 5) recoge un glosario sobre términos biológicos, características de los árboles filogenéticos, herramientas para su visualización y un análisis de herramientas de visualización y de tecnologías web con las que se pueden representar árboles.

Arquitectura

El apartado correspondiente de la memoria (pág. 9) enumera los requisitos del entorno de visualización inspirados por un biólogo experto en ADN mitocondrial del Departamento de Bioquímica, Biología Molecular y Celular de la Universidad de Zaragoza y también (pág. 10) los requisitos planteados por el equipo de investigadores de GISED para el desarrollo de la aplicación.

Diseño

El apartado correspondiente de la memoria (pág. 13) muestra las funcionalidades del sistema en términos de estructura interna del entorno: Cliente-Servidor y base de datos.

Prototipo

El apartado correspondiente de la memoria (pág. 21) describe la implementación del prototipo de visualización y desarrolla la relación de las tecnologías usadas en términos comparativos con otras que se descartaron para realizar la misma función en el entorno.

Experimentos

En este apartado de la memoria (pág. 29) se describen varias pruebas de testeo del funcionamiento, pruebas de integración e integración de todas las partes del entorno.

Organización del proyecto

En este apartado de la memoria (pág. 31) se describe la gestión en la realización de proyecto.

Conclusiones

En este apartado de la memoria (pág. 33) se desarrollan las conclusiones del proyecto.

2. Estado del arte

En este apartado se muestra un glosario de términos biológicos, las características de las filogenias extensas y también un estudio sobre diferentes herramientas tanto web como aplicaciones de escritorio para su visualización. Por último, se muestra un estudio de las diferentes tecnologías web relacionadas con la visualización en el contexto del proyecto.

2.1. Glosario de términos biológicos

Se definen los términos que van a aparecer a lo largo de la memoria para una mejor comprensión por parte del lector.

Filogenias es considerada la ciencia encargada del estudio y clasificación de la relación evolutiva entre organismos.

Árbol filogenético es un árbol que muestra las relaciones evolutivas entre varias especies u otras entidades que se cree que tienen una ascendencia común.

ADN o **ácido desoxirribonucleico** es un ácido nucleico que contiene instrucciones genéticas usadas en el desarrollo y funcionamiento de todos los organismos vivos conocidos y algunos virus, y es responsable de la transmisión hereditaria.

ADN mitocondrial es el material genético de las mitocondrias, los orgánulos que generan energía para algunas células. Solo se hereda por vía materna en caso humano.

2.2. Filogenias extensas

El proyecto aborda visualizar e interactuar con árboles extensos como es el caso del árbol filogenético de ADN mitocondrial proporcionado por ZARAMIT. Este árbol está en formato Newick (ver pág. 22 para una explicación detallada de dicho formato) y ha sido creado a partir de datos obtenidos de la base de datos de GenBank [4].

Este tipo de árboles pueden ser muy extensos y los usuarios finales del entorno necesitan visualizar e interactuar con ellos.

2.3. Herramientas de visualización para árboles y redes

En primer lugar me dediqué a investigar sobre diferentes herramientas tanto del ámbito web como de escritorio para ver las distintas formas de trabajar con

árboles filogenéticos.

Esta búsqueda me proporcionó ideas para desarrollar utilidades y formas de realizar las diferentes tareas, que eran interesantes para el sistema que iba a desarrollar. Por otro lado, también me sirvió para desechar otras que vi poco interesantes o no aplicables para mi proyecto.

Entornos **Web** relacionados:

PhyloTree [5]. Página web referente en el campo de los árboles de ADN mitocondrial. El desplazamiento por el árbol se realiza mediante la navegación por grupos, ya que el árbol está formado por grupos que a su vez albergan otros subgrupos. En el caso de querer buscar una secuencia concreta se navega por los diversos grupos hasta encontrarla.

Evolgenius [6]. Este sitio web usa colores y formas para diferenciar los datos más rápidamente. El problema que tiene es que se debe proporcionar un árbol en formato Newick y en el caso de que el tamaño de dicho árbol sea demasiado grande se colapsa. Si el tamaño del árbol es de menos de mil secuencias sí que logra representarlo pero la navegación por él es muy lenta.

Otras **Web** relacionadas (TreeVector [7], bioinformatics.psb.ugent.be [8], treeapp [9], Jstree [10]). Ofrecían las mismas prestaciones y carencias que la anterior. Además, la navegación sobre el árbol que representaban era más problemática.

Aplicaciones de escritorio multiplataforma que visualizan árboles filogenéticos:

Tree Graph 2 [11]. En este entorno se puede cargar el árbol entero sin importar lo grande que sea. El formato de entrada es de tipo Newick. Tiene como inconveniente que si el tamaño del árbol es muy grande la navegación es muy lenta. Una de las características destacables de esta aplicación es que muestra a la derecha todos los nodos del árbol en una tabla y si presionas en cualquiera de ellos mueve la vista del árbol centrándola en el nodo sobre el cual has presionado. El problema de este planteamiento es que si el árbol tiene muchos nodos resulta muy difícil encontrar un nodo concreto.

FigTree [12]. Este entorno tiene las mismas características que el anterior. Carga un árbol entero sin importar el tamaño, no posee ninguna funcionalidad para buscar un nodo rápidamente, aunque sí ofrece una amplia variedad de herramientas para cambiar la estética del árbol.

Phylowidget [13]. Puede cargar un árbol de cualquier tamaño, aunque en el caso de que este sea extenso es imposible ver casi nada del árbol en la pantalla. No obstante, si el árbol es de tamaño pequeño se puede navegar y al buscar una secuencia determinada colorea la ruta desde la raíz del árbol hasta dicha secuencia.

Dendroscope [14]. Esta aplicación puede cargar cualquier árbol. En el caso que el árbol sea muy grande permite hacer zoom y adecuarlo a un tamaño en el que se pueda trabajar. Además cuando se busca una secuencia te centra la vista en la secuencia buscada. No permite la creación de árboles al vuelo, ni árboles de grupos.

2.4. Otras tecnologías de visualización

Para abordar el problema de cómo visualizar un árbol con tecnologías orientadas a la web, se investigaron un conjunto amplio de bibliotecas. Entre otras muchas, destaco las siguientes:

Graphic JavaScript Tree with Layout y [15] **Basicprimitives** [16]. Estas dos librerías permiten crear árboles fácilmente, pero hay que introducir el árbol completo, ya que no se permite crear el árbol poco a poco.

D3 [17]. Esta biblioteca JavaScript permite trabajar con gran cantidad de datos y en el caso de los árboles sin importar el tamaño permite crearlos al vuelo. También se puede personalizar la representación de los mismos.

2.5. Problemas abiertos

Tras analizar las diferentes aplicaciones citadas anteriormente y teniendo en cuenta los requisitos que se creían más necesarios para los usuarios de la aplicación, me propuse desarrollar un sistema lo más adecuado posible desde la perspectiva de la usabilidad.

Esto me llevó a la conclusión de que el entorno debería poseer un buscador de secuencias para que así fuera posible encontrar fácilmente secuencias concretas y además el sistema tendría que centrar la vista en la secuencia buscada. También debe tener dividido el árbol en grupos para que la navegación por él sea más fluida. Además encuentro que debe diferenciar secuencias visualmente para así poder comprenderlas mejor.

3. Arquitectura

Este apartado está divido en tres partes. La primera de ellas trata de los requisitos que nos propuso un biólogo experto, la segunda desarrolla una serie de requisitos de tipo informático que debe cumplir el entorno y la última parte trata de la propuesta de solución del entorno, mostrándonos dónde va a trabajar el proyecto una vez que esté en funcionamiento.

3.1. Requisitos derivados del dominio del problema

La colaboración con el investigador Eduardo Ruiz Pesini del Departamento de Biología Molecular y Celular de la Universidad de Zaragoza ha sido fundamental para definir un conjunto de requisitos que garantice la usabilidad del entorno a desarrollar.

Los requisitos que nos planteó dicho experto fueron:

Posibilidad de ir actualizando la aplicación fácilmente

El primer problema que nos expuso es que muchas de las aplicaciones o webs a las que accedía se quedaban obsoletas con el paso del tiempo, ya que estas no eran actualizadas por sus administradores o si las actualizaban pasaban periodos de tiempo bastante largos entre actualizaciones.

Facilidad para encontrar una secuencia dentro del árbol

El segundo problema planteado era el tipo de navegación, ya que si pretendía buscar una secuencia en concreto le resultaba en algunos casos bastante difícil. Algunas de las aplicaciones se basan en una navegación por grupos, para lo cual, había que ir abriendo grupo por grupo hasta encontrar la secuencia deseada. En otros casos simplemente era un árbol completo lo que mostraba la aplicación, lo que le hacía casi imposible encontrar la secuencia deseada ya que en este caso se aglutinaba demasiada información.

Para el experto lo más importante es poder acceder rápidamente a la secuencia deseada, y así poder ver las secuencias con los nodos hermanos, hijos y padre. Otra opción necesaria es poder buscar lo más rápido posible la secuencia padre de un grupo, porque según del que proceda tendrá unas características ancestrales u otras.

Posibilidad para editar el árbol manualmente

Otras de las funciones que debía tener la aplicación era poder editar el árbol o incluso poder hacerlo nuevo desde cero, ya que existen bastantes diferencias de criterio entre los biólogos en cuanto a la colocación de ciertas secuencias en una u otra rama del árbol.

Árbol de grupos

Después de analizar la web de Phylotree.org [5] vimos que el uso de grupos era interesante y el experto indicó que la mayoría de las veces interesaba poder visualizar los grupos tal como están situados entre sí y poder ir rápidamente a la secuencia padre del grupo. Por lo tanto añadimos un árbol de grupos para que así el usuario pudiera ver todos los grupos del árbol con el que estaba trabajando y además pudiera visualizar rápidamente la secuencia padre del grupo que desea solamente presionando en el grupo dentro del árbol de grupos.

3.2. Requisitos funcionales

Los requisitos que nos planteamos a la hora de crear el visualizador fueron:

Representación parcial del árbol

Crear una representación del árbol, que no muestre un árbol completo, ya que si lo hace provoca que la navegación por dicho árbol resulte casi imposible debido a su gran tamaño (más de diez mil nodos en el árbol más grande de ZARAMIT). Por este problema en la navegación se desechó la representación completa del árbol, ya que para una secuencia concreta solo es necesario visualizar su padre, hermanos e hijos.

Buscador de secuencias

Otra funcionalidad que deberá tener la aplicación es poder buscar una secuencia lo más rápidamente posible, ya que en la mayoría de casos se conoce la secuencia de la que se necesita buscar información. Para introducir esta funcionalidad añadimos un buscador de secuencias con el cual se puede buscar rápidamente la secuencia deseada.

Historial de secuencias visitadas

Otra funcionalidad necesaria, puesto que en una sesión de trabajo se necesita acceder a varias secuencias una tras otra y además la mayoría de las veces necesitamos poder volver rápidamente a una secuencia vista previamente. Como solución se propone mantener un historial de grupos con el que poder de una forma rápida ir a las secuencias que se hayan visitado anteriormente durante la sesión de trabajo.

Árbol agrupado

Como última funcionalidad importante que deberá poseer la aplicación es que cuando se trabaje con árboles heterogéneos pueda usarse un criterio para agrupar las secuencias y así facilitar la navegación del usuario por el árbol.

3.3. Escenario para la propuesta de solución

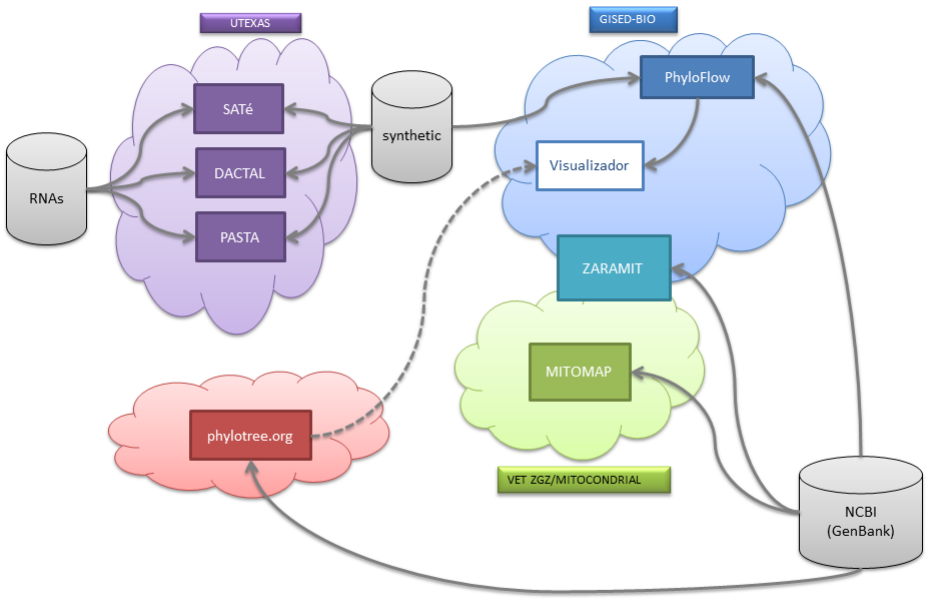

Figura 1: Escenario de grupos de investigación y herramientas para la propuesta de solución.

El entorno de visualización objeto de este trabajo tiene utilidad en sí mismo como herramienta para la visualización de árboles filogenéticos con información biológica de todo tipo. El escenario de integración previsto para el mismo es el que se retiene en la Figura 1. Dentro de la línea de investigación en Bioinformática del grupo GISED, el entorno de visualización se integra como parte fundamental para el análisis y visualización de las filogenias producidas mediante el entorno PhyloFlow desarrollado por el investigador Jorge Álvarez Jarreta. Adicionalmente, está previsto poner el entorno de visualización a otros grupos de investigación, como es el caso del grupo de investigación de Tandy Warnow de la Universidad de Texas, y el de Biogénesis y patología mitocondrial de la Universidad de Zaragoza, facilitando un entorno colaborativo para la compartición de resultados y datos.

4. Diseño

Se ha desarrollado una aplicación web basada en una arquitectura cliente-servidor compuesta por tres capas, usando el patrón MVC tanto en el cliente como en el servidor. Seguidamente se explica el servidor de datos y alguna de las operaciones importantes que se pueden realizar en dicho servidor. La descripción completa de este tipo de arquitectura se encuentra en el *Anexo A* (Arquitectura Cliente-Servidor).

4.1. Cliente-Servidor

La siguiente ilustración muestra la estructura del visualizador.

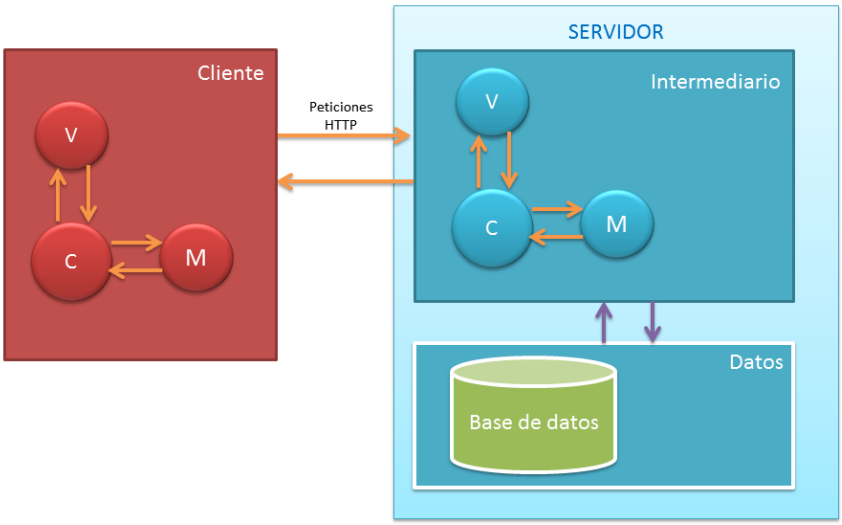

Figura 2: Arquitectura Cliente-Servidor.

Cliente

El cliente solicita las peticiones mediante la interfaz de usuario. En nuestro caso al tratarse de una aplicación web, puede ser usada desde cualquier navegador accediendo a la web donde está alojado el entorno.

Servidor

El servidor está compuesto por un servidor intermediario y un servidor de datos. El servidor intermediario tiene como función proporcionar los recursos solicitados para el cliente, los cuales estarán alojados en el servidor de datos.

La Tabla 1 recoge las solicitudes que acepta el servidor con las respuestas que se obtendrá de cada una de ellas.

Grupo	Peticiones
Árbol	Buscar (Árbol, Árboles privados y públicos, Raíz, Árbol de grupos) Insertar Editar Eliminar
Nodo	Buscar (Nodo, Padre, Hijos, Alertas, Observaciones, Grupo) Insertar Editar Eliminar
Alerta Obs Grupos	Insertar Editar Eliminar

Tabla 1: Peticiones ordenadas por grupo.

La Figura 3 ilustra una representación esquemática del funcionamiento del sistema cuando se realiza una petición. Los flujos de datos que tienen (1-4) son los siguiente.

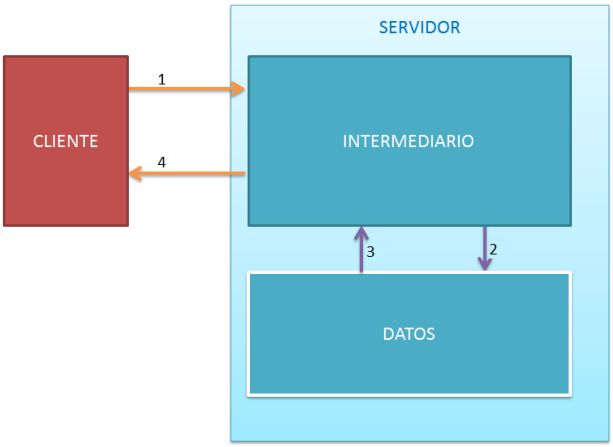

Figura 3: Comunicación entre Cliente-Servidor.

1. El cliente solicita los datos al servidor intermediario.
2. El servidor intermediario solicita los datos del servidor de datos.
3. El servidor de datos responde al servidor intermediario.
4. El servidor intermediario responde con los datos procesados al cliente.

4.2. Base de datos

Tal como está estructurada la aplicación se ha optado por el sistema de tres capas, en el que el tercer nivel tercer corresponde al servidor de datos.

4.2.1. Estructura

En nuestro caso para hacer esta función de servidor de datos se utiliza un servidor de bases de datos. Dicha base de datos esta diseñada de acuerdo al siguiente diagrama entidad-relación que tendrá las siguientes tablas:

- La Entidad Alerta sirve para almacenar las alertas que poseen las secuencias de un árbol.
- La Entidad Obs(observaciones) se usa para guardar las observaciones que tiene una secuencia.
- La Entidad Usuario contiene aquellos usuarios que pueden acceder al sistema.

- La Entidad Arbol guarda los árboles que hay en el sistema, ya que aunque inicialmente partamos de un solo árbol, más adelante se le irán añadiendo árboles a la aplicación, según las necesidades de los usuarios.
- La Entidad Grupo almacena los grupos de las secuencias que hay en el sistema.
- La Entidad Nodo guarda las secuencias de los árboles que hay en el sistema.

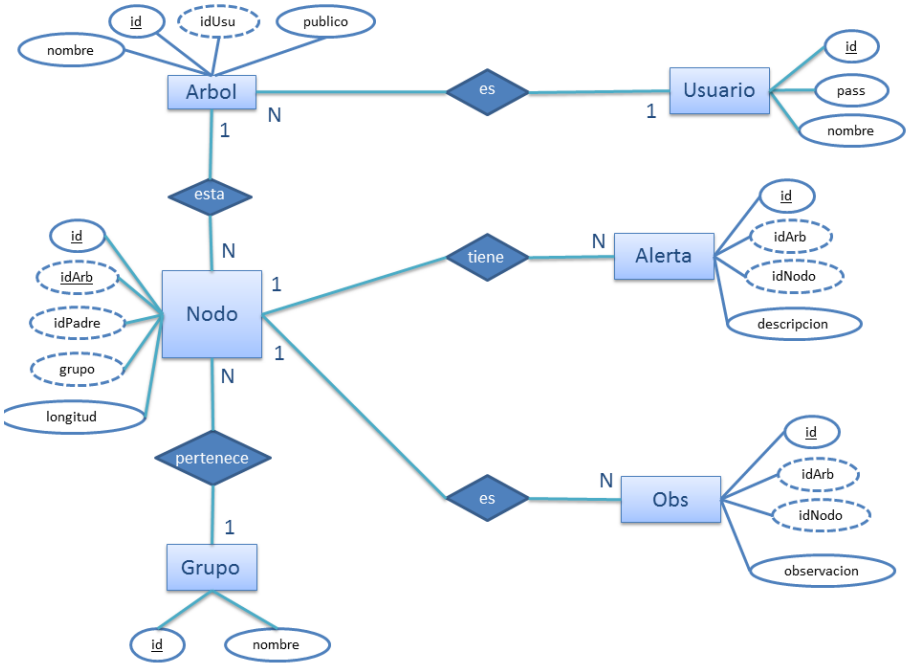

Figura 4: Diagrama entidad-relación.

En el *anexo B* (Diseño de la base de datos) se puede encontrar la información detallada de la base de datos y de sus tablas.

4.2.2. Carga de datos

La inserción de los datos iniciales del visualizador se hace mediante un fichero en formato Newick. Este es el formato predominante en las aplicaciones de en Biología que trabajan con árboles filogenéticos. El árbol incluido dentro del fichero será el que se introduce en el servidor de la base de datos de la aplicación para que los usuarios puedan trabajar con él.

4.2.3. Copia de seguridad

La forma de realizar la copia de seguridad de un árbol de un cliente se hará mediante la realización de scripts SQL que contendrán la información de ese árbol (nodos, alertas, observaciones,.....). Así el administrador después de ver las solicitudes del cliente para restaurar un árbol propio, accederá al script SQL del cliente y lo introducirá en el sistema reemplazando el árbol anterior.

4.2.4. Duplicado de árboles

En el caso de que un cliente quiera duplicar un árbol propio o tener una copia de un árbol público hará una solicitud a uno de los administradores del sistema. En el caso que la solicitud sea aceptada, el administrador genera un script SQL y lo introduce en el sistema. Se sigue este tipo de protocolo debido a que el tamaño de un árbol dentro del sistema es grande y se pretende evitar la sobrecarga del servidor por las posibles actuaciones directas que pudieran hacer los usuarios del sistema.

4.3. Visualización padres-hijos-hermanos

De acuerdo con los requisitos del dominio del problema se ha establecido que la visualización del árbol siga un planteamiento en el que se muestra un nodo seleccionado donde se puede ver además padre, hijos y hermanos de ese nodo (Figura 5). De esta forma se descarta la posibilidad de ver un árbol global.

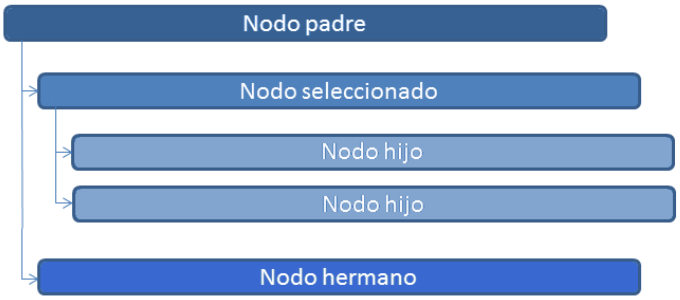

Figura 5: Ejemplo de cómo es el sistema de vista padre, hermanos e hijos de un nodos seleccionado.

1. Nodo padre del nodo seleccionado.
2. Nodo seleccionado.
3. Nodos hijos del nodo seleccionado.
4. Nodos hermanos del nodo seleccionado.

El sistema visualiza, a partir de un "nodo seleccionado", su padre, hermanos e hijos. Así, cuando se presione sobre el nodo padre, hermano o hijo, la vista se recargará de nuevo pasando este a ser el "nodo seleccionado", generando su padre, hermanos e hijos correspondientes.

Si el usuario desea añadir y eliminar nodos del árbol, tiene que estar usando un árbol propio, no uno de dominio público. Este sistema se ha desarrollado para poder añadir o eliminar el "nodo seleccionado" o los nodos hijos en el caso de que estos sean nodos finales.

En el caso de añadir nodos se dan dos situaciones:

Añadir al "nodo seleccionado". En el caso de que este posea hijos se puede elegir cualquiera de ellos o todos para que pasen a ser hijos del nodo añadido.

Añadir a un nodo denominado hijo. En este caso se recargará la vista pasando el hijo a ser el nuevo "nodo seleccionado".

Para eliminar nodos, sólo hay una situación. En el caso de que el nodo que se desea eliminar tuviera hijos, estos pasarían a ser nodos hijos del padre del

nodo eliminado.

Otras de las acciones que se pueden llevar a cabo son añadir alertas y observaciones en los nodos. Para que se pueda realizar esta acción he creado unos iconos en la "barra de nodo" que al hacer click muestran una ventana en la cual los usuarios podrán añadir alertas u observaciones.

Cuando el usuario presione sobre cualquier nodo que no sea el "nodo seleccionado" este pasa a ser el nuevo "nodo seleccionado" y se actualiza la vista de generando los nuevos padres, hermanos e hijos.

La aplicación posee un buscador de nodos que permite buscar y cargar en el sistema de la vista el nodo deseado como el "nodo seleccionado".

Otra acción que puede realizar el usuario es acceder rápidamente al nodo padre de un grupo y ver un árbol de grupos del árbol con el que está trabajando, si presiona sobre cualquier grupo el sistema carga la vista y pone al padre de ese grupo como "nodo seleccionado".

Una sugerencia del experto fue poder ver un historial de los nodos que ha ido visualizando y acceder rápidamente a ellos. Para poder realizar esta acción, he añadido un historial de nodos, que con solo presionar sobre cualquier nodo del historial este pasa a ser el nuevo "nodo seleccionado" de la vista.

5. Prototipo

Una vez planteados el diseño y la arquitectura del sistema a desarrollar, el siguiente paso fue aplicar las tecnologías adecuadas para las distintas partes de la aplicación.

En el apartado se describen qué tipo de tecnologías he usado para cada parte del sistema. Además he hecho una comparativa con otras tecnologías para justificar las decisiones de implementación frente a otras análogas.

Para la ejecución del sistema de visualización de árboles, he optado por usar tecnologías compatibles con JAVA ya que las tecnologías de esta familia son la mayoritariamente gratuitas. Otra opción que me planteé para desarrollar la aplicación era usar tecnologías de Microsoft que no son gratis. Por ejemplo, para desarrollar la web podría haber utilizado ASP en vez de JSP y también usar un servidor WebApi frente al servidor de Jersey proyect.

Las decisiones de implementación tomadas han proporcionado un sistema multiplataforma, pudiéndose instalar en equipos con distintos sistemas operativos como podrían ser Windows, Linux o Mac OSX. El prototipo de la aplicación ha sido desarrollado bajo Linux usando la distribución Ubuntu.

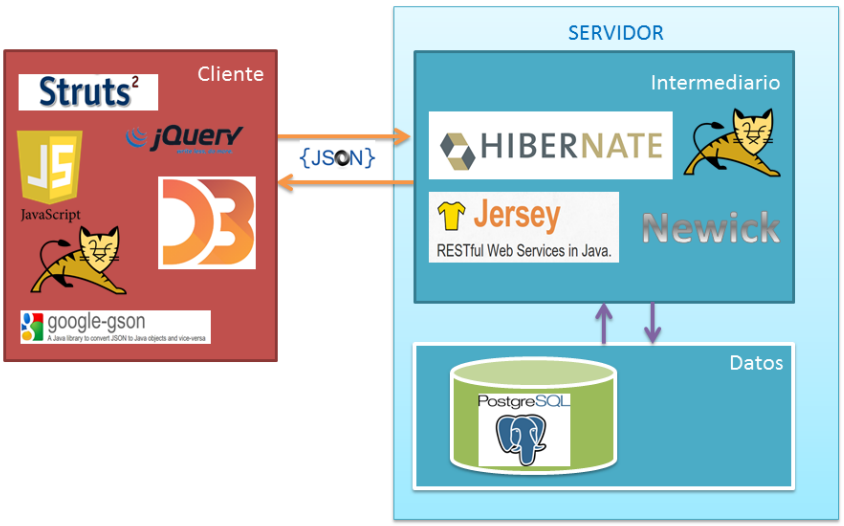

Figura 6: Implementación de la aplicación.

5.1. Cliente

En este apartado expongo las tecnologías que he usado para realizar el cliente: Struts 2, JavaScript, jQuery, D3, Gson y Apache Tomcat.

Struts 2

Struts 2 es una herramienta de soporte para el desarrollo de aplicaciones Web bajo el patrón MVC usando la plataforma Java EE (Java Enterprise Edition), además se distribuye bajo la licencia Apache License 2.0 siendo software libre.

Con esta herramienta el cliente de la aplicación implementa el patrón MVC. Este planteamiento busca reducir el acoplamiento funcional del sistema dividiendo las responsabilidades en tres capas claramente diferenciadas: el modelo, que hace referencia a los datos que maneja la aplicación, las reglas de negocio que operan sobre ellos (en Struts 2 en las "acciones") y la vista con la interfaz de usuario (en Struts 2 equivale a los resultados).

El controlador, comunica la vista y el modelo, respondiendo a eventos generados por el usuario en la vista, invocando cambios en el modelo, y devolviendo a la vista la información del modelo necesaria para que pueda generar la respuesta adecuada para el usuario.

JavaScript

JavaScript es un lenguaje de programación interpretado, siendo un dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico. Este lenguaje permite realizar mejoras en la interfaz de usuario y páginas web dinámicas. Todos los navegadores modernos interpretan el código JavaScript integrado en las páginas web.

jQuery

jQuery es una librería de JavaScript, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web.

Esta biblioteca ofrece una serie de funcionalidades basadas en JavaScript mas útiles para el desarrollador. La librería es software libre y de código abierto.

D3

D3 es una librería de JavaScript para manipular documentos basados en datos. D3 me ha ayudado a crear interesantes formas visuales para representar los datos usando HTML, SVG y CSS. D3 pone énfasis en usar estándares abiertos para así poder aprovechar todas las capacidades de los nuevos navegadores, combinando poderosos componentes de visualización y un enfoque basado en datos de la manipulación DOM.

Gson

Gson es una librería de código abierto para el lenguaje de programación Java que permite la serialización y deserialización entre objetos Java y su representación en notación JSON.

Apache Tomcat

Apache Tomcat es un servidor web con soporte de servlets y JSPs. Dado que Tomcat fue escrito en Java, funciona en cualquier sistema operativo que disponga de la máquina virtual Java.

Con este servidor he podido alojar el cliente del entorno.

5.2. Servidor

En este apartado expongo las tecnologías que he usado para realizar el servidor intermediario: Jersey, Hibernate, JSON y Apache Tomcat.

Jersey

Jersey es un patrón de diseño en código abierto ayuda a crear un servidor web RESTful en Java. Gracias a este patrón he creado el servidor intermediario, que recibe las peticiones de los clientes con los datos de la petición y les devuelve la respuesta con los datos que han solicitado.

Como alternativas de implementación valoré las posibilidades de hacerlo mediante un servidor REST o SOAP. Investigué cuál de estas dos tecnologías era mas adecuada. En la Tabla 2 y la Tabla 3 se analizan las características de cada tipo de servidor.

RESTful (Representational state transfer)

Pros
- Se trabaja con componentes y se utiliza .NET o Java es muy sencillo de consumir. - Es muy ligero, sus respuestas contienen exactamente la información que necesitamos. - Es sencillo de desarrollar y no se necesita mucho código extra. - Es flexible en cuanto al tipo de respuesta que se necesita, ya que puede ser XML o JSON(Es el que usamos en el desarrollo de la aplicación).
Contras
- No hay un estándar en sus respuestas por lo que no se definen tipos de datos.

Tabla 2: Pros y Contras de RESTFul.

SOAP (Simple Object Access Protocol)

Pros
- Se trabaja con componentes y se utiliza .NET o Java es muy sencillo de consumir. - El resultado que devuelve siempre es XML y contiene una definición específica del tipo de dato, lo que hace del protocolo algo muy estricto. - Se dice que es más seguro porque su implementación siempre o la mayoría de las veces se hacen del lado del servidor.
Contras
- Una vez implementado, si se desea cambiar algo en el servidor impacta de forma negativa en los clientes ya que estos tienen que hacer muchas modificaciones al código. - Las respuestas son demasiado complejas y difíciles de interpretar si no se tienen las herramientas correctas para hacerlo.

Tabla 3: Pros y Contras de SOAP.

Como conclusión elegí el servidor RESTful, ya que las tecnologías JavaScript, jQuery y D3 con las que trabajaba en la vista usan por defecto Json. En el caso de usar SOAP, que emplea XML, se tendría que traducir de JSON a XML cada vez que mandara o recibiera algo del servidor para así poder trabajar con los datos. Además si se tuviera que cambiar algo en el servidor, supondría

un mayor coste. Finalmente consideré que si se quisiera extender el desarrollo del visualizador de árboles a un dispositivo móvil con Android era mucho más sencillo hacerlo con REST que con SOAP.

Hibernate

Hibernate es una herramienta de Mapeo objeto-relacional (ORM) para la plataforma Java. Facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación mediante archivos declarativos.

Hibernate facilita manipular los datos de la base de datos operando como si fueran objetos, con todas las características de la Programación orientada a objetos. Hibernate convierte los datos entre los tipos utilizados por Java y los definidos por SQL. Además genera las sentencias SQL y libera del manejo manual de los datos resultantes de la ejecución de dichas sentencias.

Además en el caso de cambiar el Servidor de la Base de Datos de la aplicación por otro diferente que sea compatible con Hibernate, solamente habría que realizar cambios mínimos en un archivo de configuración para que la aplicación continúe funcionando correctamente.

Apache Tomcat

Permite alojar el servidor web del entorno.

JSON

JSON(JavaScript Object Notation) es un formato ligero para el paso de datos. Otro formato que podía haber elegido es XML, pero creo que JSON es más ligero, presenta mejor la estructura de los datos y requiere menos codificación y procesamiento por parte de cliente y servidor.

Este formato es usado por el servidor para responder una solicitud del cliente aportando los datos que se le han requerido.

```
[{"id":{"id":2,"idarbol":3},"nombre":"n1","idpadre":1,"grupo":-1,"longitud":0.0,  hijos :[
                {"id":{"id":231,"idarbol":3},"nombre":"n2","idpadre":2,"grupo":-1,"longitud":0.0},
                {"id":{"id":3,"idarbol":3},"nombre":"n3","idpadre":2,"grupo":-1,"longitud":0.0}
                ]
}
```

Figura 7: Ejemplo de un vector de nodos en JSON.

5.3. Base de datos

En este apartado expongo las tecnologías que he usado para realizar el servidor de datos: PostgreSQL y Newick.

PostgreSQL

PostgreSQL es un SGBD (sistema de gestión de bases de datos) relacional orientado a objetos y libre. Este es el sistema que he escogido para que sea el servidor de la base de datos. Ya que posee las siguientes características:

- Alta concurrencia mediante un sistema denominado *MVCC* (Acceso concurrente multiversión, por sus siglas en inglés. Permite que mientras un proceso escribe en una tabla, otros procesos pueden acceder a la misma tabla sin necesidad de bloqueos. Cada usuario obtiene una visión consistente de los últimos datos cuando se realizó el último commit.
- Tiene una *gran variedad de tipos nativos* y además los usuarios pueden crear sus propios tipos de datos.
- Implementa *claves ajenas* también denominadas Llaves ajenas o Claves Foráneas.
- Posee *disparadores (triggers)*, se definen como una acción específica que se realiza de acuerdo a un evento, cuando éste ocurra dentro de la base de datos.
- Posee *vistas*, *herencia de tablas*, *multitud de tipos de datos* y *operaciones geométricas*.

Los motivos de la elección de este sistema de base de datos frente a otros como podrían ser Oracle o MySQL fueron:

1. En el caso de Oracle fue debido al alto coste que supone añadir a la aplicación este sistema de bases de datos, además creo que no aporta grandes mejoras frente a PostgreSQL.
2. En el caso de MySQL fue por no poseer claves ajenas, las cuales te dan una integridad referencial de los datos que resulta muy útil para que estos sean coherentes.

Newick

Newick es el formato de gran aceptación en las aplicaciones típicas de la Bioinformática en Análisis Filogenético.

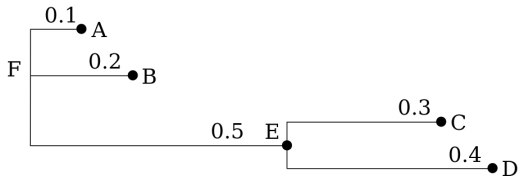

Figura 8: Ejemplo Newick para el árbol (A,B,(C,D)E)F;

5.4. Interacciones

El intercambio de datos entre el cliente y el servidor se hace mediante peticiones HTTP (GET, POST).

Las peticiones GET llevan todos los datos necesarios en la propia dirección url de la llamada. Cuando el servidor la recibe, la procesa y devuelve los datos solicitados.

Las peticiones POST los datos no vienen dentro de la dirección url. Estas tipo de peticiones indican al servidor que se prepare para recibir datos, una vez que ha recibido dichos datos, los procesa y los devuelve procesados al cliente para que trabaje con ellos.

El paso de datos entre el cliente y el servidor se realiza en formato JSON.

6. Experimentos

Este apartado recoge la validación funcional del prototipo y los resultados obtenidos para los experimentos realizados con el prototipo.

6.1. Validación funcional

La integración de las diferentes partes de que consta el prototipo de la aplicación, se hizo de la siguiente manera. El primer paso fue instalar el servidor PostgreSQL y cargar la base de datos del sistema en él. El siguiente paso fue crear el servidor intermediario y configurarlo para que aceptara las peticiones que recibe del cliente. Luego creé la vista de la aplicación y esta necesitaba los datos que poseía nuestra base de datos, a los cuales por el tipo de arquitectura usada no tenía acceso directo por mayor seguridad. Por tanto, el siguiente paso lógico fue ir añadiendo a nuestro servidor intermediario las peticiones necesarias para suministrar a la vista los datos necesarios.

6.2. Pruebas de rendimiento

El equipo usado para realizar las pruebas de rendimineto de la aplicacion fue el siguiente:

1. Procesador: AMD A-10 5800k
2. RAM: 8GB DDR3
3. Placa Base: MSI FM2-A85XA-G65

Prueba 1

Verificar el tamaño de un árbol según el numero de nodos que posea y el tiempo de carga en la base de datos. Los resultados se pueden ver en la Tabla 4.

Nombre	número nodos	Tamaño	Tiempo carga
arbol1	67	40kB	1 seg
arbol2	848	128kB	3 seg
arbol3	14512	2MB	15 seg
arbol4	276	62kB	1 seg
arbol5	43534	5MB	31 seg
arbol6	217666	21MB	130 seg

Tabla 4: Carga de arboles en la base de datos.
El número de nodos del árbol y el tamaño de las etiquetas estos, son un factor determinante en el tamaño de la base de datos.

Prueba 2

Los resultados de la Tabla 5 muestran el tiempo de respuesta del servidor intermediario al recibir una petición por parte de un cliente.

Nombre	Tiempo respuesta
dameUnNodoPorId	49 ms
dameArbolPorId	41 ms
dameHijosDeNodo	50 ms
dameRaizDeArbol	44 ms
dameBarraNodoPorId	58 ms
dameHijosEnBarraNodoPorId	106 ms
dameRaizDeArbolEnBarra	88 ms
dameAlertasNodo	60 ms
dameObsNodo	67 ms
dameArbolesDeUsuario	49 ms
dameArbolesPublicos	53 ms

Tabla 5: Tabla con el tiempo de respuesta del servidor intermediario.

El tiempo de respuesta pueden variar levemente según el tamaño de la etiquetas de los datos que se soliciten al servidor.

7. Organización del proyecto

En este apartado se exponen los detalles de la gestión en la realización del proyecto en la Figura 9.

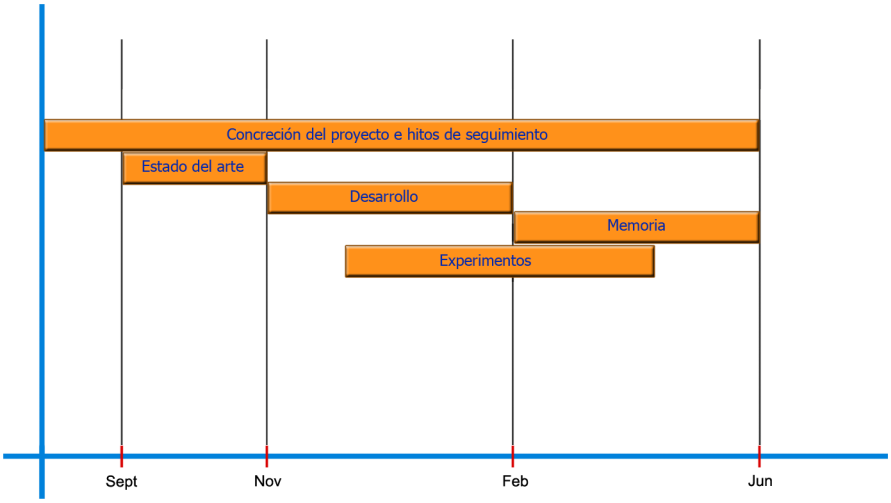

Figura 9: Gestión en la realización del proyecto

Las fases contempladas en el desarrollo del proyecto han sido las siguientes.

- *Concreción del proyecto e hitos de seguimiento*. En esta fase es donde concretamos lo que albergaría el proyecto y el seguimiento del mismo.
- *Estado del arte*. En esta fase se investigó las formas actuales para la visualización de los árboles y además se buscaron diferentes tecnologías con lo que se pudiera realizar las diferentes partes de la aplicación.
- *Desarrollo*. En esta fase se desarrolló el prototipo de la aplicación.
- *Memoria*. En esta fase se realizó la memoria.
- *Experimentos*. En esta fase se realizaron diferentes pruebas en el sistema mientras estaba en fase de desarrollo y cuando el prototipo estaba finalizado para averiguar si era estable y ofrecía un buen rendimiento.

En cuanto a las horas de trabajo de cada una de las fases, la Tabla 6 recoge los datos correspondientes.

Concreción del proyecto e hitos de seguimiento	100 horas
Estado del arte	70 horas
Desarrollo	320 horas
Experimentos	60 horas
Memoria	110 horas

Tabla 6: Resumen de horas por fase de proyecto.

8. Conclusiones

En este proyecto he desarrollado una herramienta con la cual se puede visualizar y obtener información de una secuencia molecular dentro de un árbol filogenético de ADN mitocondrial, pudiendo además conocer rápidamente sus ancestros, sus descendientes y también donde está localizada dicha secuencia dentro del árbol.

El primer problema con que me encontré fue comprender los distintos términos biológicos con los que tenía que trabajar para el desarrollo del proyecto. Para dar solución a este problema, Jorge Álvarez Jarreta me dio una charla con todos los términos biológicos que podían ser necesarios para el mejor desarrollo de mi trabajo así como la relación de dichos términos entre sí.

A la hora de desarrollar el proyecto me surgieron varias dudas sobre qué tipo de tecnologías debía usar para el desarrollo de las diferentes partes en las que este se divide. Para afrontar este problema tuve que testear varias tecnologías, para ver cuál de ellas se acoplaba mejor a los requisitos para ser aplicada en cada parte del proyecto.

Las expectativas de futuro del sistema creo que pueden ser grandes ya que he dejado la puerta abierta para que se puedan acoplar aplicaciones al sistema fácilmente. La creación de un servidor intermediario es lo que nos brinda que se puedan crear diferentes tipos de clientes que pueden acoplarse al sistema fácilmente. Por ejemplo, se puede crear un cliente de escritorio que muestre el árbol entero o solamente una secuencia, otro ejemplo podría ser crear una aplicación móvil, que muestre información de las secuencias del árbol que el usuario desee ver. En el caso de que un nuevo tipo de cliente necesite algún dato que el servidor no pueda servirle, solo tendrá que implementarse para que nuestro servidor intermediario acepte un nuevo tipo de petición HTTP que responda con el dato que este necesita. Gracias al uso de un gestor de base de datos con soporte para multitud de lenguajes de programación facilitará la tarea de integrar nuevos programas desarrollados en lenguajes de programación como Python y Perl, muy utilizados en Bioinformática.

Como conclusión, gracias a la realización del proyecto he podido conocer cómo trabajan y con qué problemas se encuentran los Biólogos especializados en el campo del ADN mitocondrial cuando trabajan con árboles filogenéticos, al usar las herramientas informáticas existentes.

Personalmente la realización del proyecto me ha aportado grandes conocimientos sobre tecnologías web:

1. jQuery y D3, tecnologías que nos ayudan a crear dinamismo y facilitan la creación de vistas en web dinámicas. Gracias a trabajar con estas librerías he adquirido experiencia para el desarrollo de este tipo de webs, ya que

nunca me había encontrado antes con este tipo de webs.

2. Struts 2 y Hibernate, ya me eran conocidas y gracias al proyecto he podido ampliar mi experiencia en el uso de estas.

3. Jersey, me ha aportado el conocimiento de cómo trabajar con servidores Restful gratuitos, puesto que solo conocía la tecnología existente de Microsoft (WebApi) para poder realizar este tipo de tarea.

Por lo tanto estoy muy contento con los conocimientos adquiridos y resultado obtenido con la realización de este PFC.

9. Bibliografía

Referencias

[1] Blacon,R. Definición y prototipo de herramienta de análisis filogenético para el estudio del ADN mitocondrial humano. Centro Politécnico Superior, Universidad de Zaragoza. 2008

Web: http://webdiis.unizar.es/ robertob/zaramit/

[2] Liu K, Warnow TJ, Holder MT, Nelesen S, Yu J, Stamatakis A, Linder RC. 2012. SATé-II: Very Fast and Accurate Simultaneous Estimation of Multiple Sequence Alignments and Phylogenetic Trees. Systematic Biology. 61(1):90-106.Google Scholar SATé: Web://phylo.bio.ku.edu/software/sate/sate.html

[3] GISED: http://webdiis.unizar.es/GISED/

[4] The NCBI handbook [Internet]. Bethesda (MD): National Library of Medicine (US), National Center for Biotechnology Information; 2002 Oct. Chapter 18, The Reference Sequence (RefSeq) Project. Available from http://www.ncbi.nlm.nih.gov/books/NBK21091/

Pruitt KD, Brown GR, Hiatt SM, Thibaud-Nissen F, Astashyn A, Ermolaeva O, Farrell CM, Hart J, Landrum MJ, McGarvey KM, Murphy MR, O'Leary NA, Pujar S, Rajput B, Rangwala SH, Riddick LD, Shkeda A, Sun H, Tamez P, Tully RE, Wallin C, Webb D, Weber J, Wu W, Dicuccio M, Kitts P, Maglott DR, Murphy TD, Ostell JM. RefSeq: an update on mammalian reference sequences. Nucleic Acids Res. 2013 [ePub] PubMed

Tatusova T, Ciufo S, Fedorov B, O'Neill K, Tolstoy I. RefSeq microbial genomes database: new representation and annotation strategy. Nucleic Acids Res. 2014 Jan 1;42(1):D553-9 PubMed

Web: https://www.ncbi.nlm.nih.gov/genbank/

[5] van Oven M, Kayser M. 2009. Updated comprehensive phylogenetic tree of global human mitochondrial DNA variation. Hum Mutat 30(2):E386-E394. doi:10.1002/humu.20921

Web: http://www.phyloTree.org

[6] Evolgenius: http://www.evolgenius.info

[7] TreeVector: http://supfam.cs.bris.ac.uk/TreeVector/

[8] bioinformatics.psb.ugent.be: http://bioinformatics.psb.ugent.be/hypergeny/quickview.php

[9] TreeApp: http://iubio.bio.indiana.edu/treeapp/treeprint-form.html

[10] Jstree: http://lh3lh3.users.sourceforge.net/jstree.shtml

[11] TreeGraph: http://treegraph.bioinfweb.info/

[12] FigTree: http://tree.bio.ed.ac.uk/software/figtree/

[13] PhyloWidget: http://www.phylowidget.org/

[14] Dendroscope: http://ab.inf.uni-tuebingen.de/software/dendroscope/welcome.html

[15] GJSTree: http://www.codeproject.com/Articles/16192/Graphic-JavaScript-Tree-with-Layout

[16] Basicprimitives: http://www.basicprimitives.com

[17] D3.js: http://d3js.org/

A. Arquitectura Cliente-Servidor

Para la realización del proyecto del visualizador necesitaba crear una aplicación web, mediante la cual los usuarios pudieran acceder vía web. Para así poder ver secuencias y además poder localizarlas dentro del árbol y obtener la información deseada.

Al tratarse de una aplicación web decidí realizarla usando la arquitectura Cliente-Servidor.

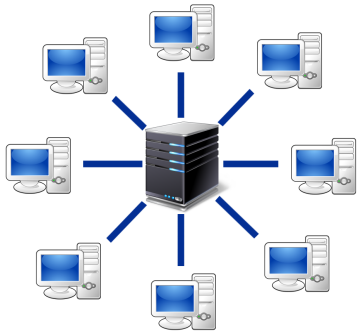

Figura 10: Ejemplo de arquitectura Cliente-Servidor.

En este tipo de arquitectura llamada Cliente-Servidor, las tareas se reparten entre el cliente y el servidor. El servidor será el proveedor que proporciona recursos o servicios y el cliente será el consumidor que contacta con él, para hacer uso de los recursos que este provee.

Este tipo de arquitectura está teniendo mucho auge gracias a lo fácil que resulta usar un navegador web como cliente ligero, además tiene otras ventajas como la independencia del sistema operativo que utilice el usuario en su PC o en cualquier dispositivo con acceso a la web. Otra razón de su popularidad es la facilidad para actualizar y mantener, ya que no se requiere la distribución, instalación y actualización de la aplicación en miles de usuarios; bastara con modificar el cliente web para cambiarle la apariencia y las acciones que pueda realizar, o actualizar el servidor para que las nuevas peticiones puedan ser servidas.

Para que las aplicaciones web funcionen de forma adecuada en cualquier navegador es necesario que se cumplan estándares, tanto por parte de la aplicación desarrollada como del navegador mismo.

Dentro de la arquitectura Cliente-Servidor podemos encontrar diferentes formas de desarrollarla. La más simple sería la arquitectura de dos niveles en donde el

cliente solicita recursos y el servidor responde directamente a la solicitud, con sus propios recursos. Esto significa que el servidor no requiere ninguna otra aplicación para proporcionar parte del servicio.

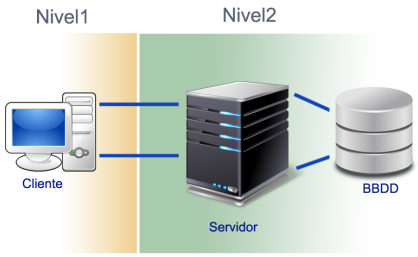

Figura 11: Ejemplo de arquitectura Cliente-Servidor de dos capas.

Otra forma de esta arquitectura Cliente-Servidor más compleja y que proporciona mucho mejor servicio que la anterior sería la compuesta por tres niveles, que es la que he decidido usar para el desarrollo de la aplicación. En este tipo de arquitectura además de los dos niveles mencionados anteriormente, lo que hago es dividir el servidor en dos niveles a su vez: el servidor de datos y el servidor intermediario.

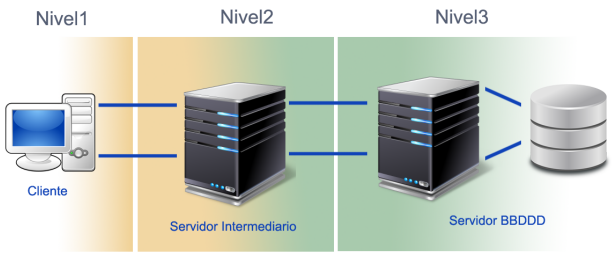

Figura 12: Ejemplo de arquitectura Cliente-Servidor de tres capas.

Con esta forma de desarrollar la aplicación la arquitectura del sistema estará formada por:

- El cliente, que solicita las peticiones, mediante una interfaz de usuario, en nuestro caso al tratarse de una aplicación web, será un interfaz web, que puede ser usada desde cualquier navegador.
- El servidor intermediario tiene como función proporcionar los recursos solicitados por cliente, los cuales están alojados en el servidor de datos.

- El servidor de datos es el que almacena los datos que componen nuestra aplicación y se los proporciona al servidor intermediario cuando este se los requiere mediante una petición previa del cliente.

Las ventajas que nos proporciona este tipo de arquitectura de tres capas son las siguientes:

- Mayor seguridad frente a los ataques contra el sistema de datos de la aplicación, ya que podemos hacer que al servidor de datos solo se pueda acceder mediante el servidor intermediario. Pueden estar conectados a los dos mediante red local, el servidor de datos sin acceso a Internet y el servidor intermediario con acceso a la web para así, poder recibir las peticiones de nuestros clientes, que estén usando la aplicación web, mediante un navegador.
- Mayor variedad de tipos de clientes que pueden acceder al sistema, unos clientes podrán acceder mediante un navegador web, otros con una aplicación móvil y otros desde aplicaciones de escritorio, todos ellos podrán acceder a los datos del servidor a través del servidor intermediario, trabajando todos ellos al mismo tiempo.
- Mayor flexibilidad frente a las actualizaciones, pudiendo mejorar las aplicaciones clientes a lo largo del tiempo, para que no se queden obsoletas. También se podrá actualizar el realizar peticiones, y en el caso de querer añadir algún tipo de mejora que necesite otros datos que provengan del sistema de datos solo habrá que añadir que el servidor intermediario pueda recibir una nueva petición para que la pueda servir.

Una vez decidida la arquitectura del sistema, desarrollé una aplicación cliente servidor con tres niveles diferenciados, por lo tanto paso a concretar la arquitectura interna de cada uno de ellos. Tanto el cliente como el servidor se van a implementar mediante el patrón de diseño MVC.

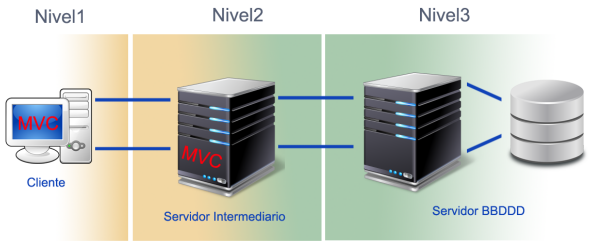

Figura 13: Ejemplo de arquitectura Cliente-Servidor tres capas con MVC.

Como he dicho anteriormente el servidor estará realizado siguiendo el patrón de diseño MVC, separando por capas las diferentes partes de la aplicación:

- La capa de la vista permite al cliente interactuar con el servidor usando peticiones HTTP, mediante estas peticiones llegarán al servidor datos por parte del cliente.
- La capa del controlador será la que se encargue de generar las respuestas para las peticiones que ha recibido la capa de la vista.
- La capa del modelo será la que proporcione los datos al controlador para que este haga los cálculos necesarios, para generar una respuesta a la capa de la vista.
- Al igual que en el servidor, en el caso del cliente del sistema también utilizo un patrón de diseño MVC para separar la aplicación cliente en diferentes capas:
- La capa de la vista que quien interactúa con ella en este caso, es el usuario o sea este podrá ver los datos e interactuar con ellos mediante eventos.
- La capa del controlador recogerá los eventos generados en la vista, según las necesidades del usuario, luego obtendrá y trabajará con los datos del modelo para después devolverlos a la vista y que esta se los represente al usuario.
- La capa del modelo es el almacén de datos de la aplicación pero en la parte del cliente del sistema, el modelo es algo peculiar ya que son los datos que pida al servidor mediante las peticiones HTTP que la capa de la vista del servidor permite.

B. Diseño de la base de datos

Este anexo está formado por una descripción detallada de la base de datos empleada en el entorno donde pueden verse la descripción de las tablas existentes y el diagrama entidad-relación.

B.1. Diagrama entidad-relación

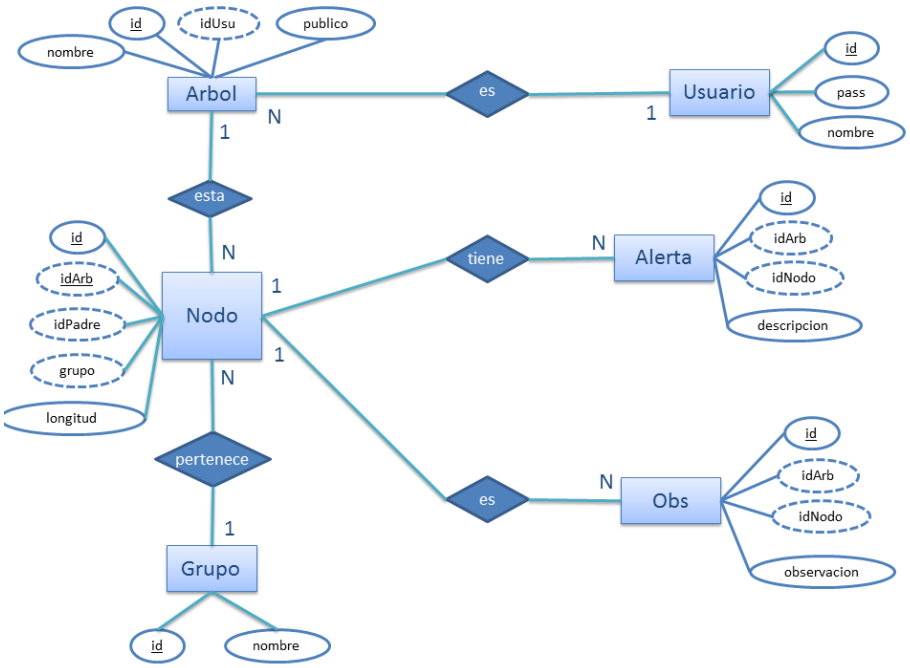

Figura 14: Diagrama entidad-relación.

En este esquema podemos visualizar comó están interrelacionadas las entidades entre si. Teniendo las siguiente relaciones:

Entidad Alerta

Una alerta solo podrá ser de un único nodo(1-1)

Entidad Obs

Una observación solo podrá ser de un único nodo(1-1)

Entidad árbol

Un árbol sera únicamente de un único usuario(1-1)

Entidad Grupo

Un Grupo puede tener muchos nodos(1-N)

Entidad Usuario

Un usuario puede poseer muchos árboles (1-N)

Entidad Nodo

Un nodo puede tener muchas alerta(1-N)

Un nodo puede tener muchas observaciones(1-N)

Un nodo pertenece a un único grupo(1-1)

Un nodo tiene que tener un único padre(1-1)

Ahora pasaré a explicar cada entidad con los tipos de datos de sus campos y con sus claves primarias y foráneas.

B.2. Descripción de las Entidades del entorno

La Entidad alerta sirve para almacenar las alertas que poseen las secuencias de un árbol.

La Entidad observaciones se usa para guardar las observaciones que posee una secuencia.

La Entidad usuario contiene aquellos usuarios que pueden acceder al sistema.

La Entidad árbol guarda los árboles que hay en el sistema, ya que aunque inicialmente partamos de un solo árbol más adelante se le irán añadiendo árboles a la aplicación, según las necesidades de los usuarios.

La Entidad grupo almacena los grupos de las secuencias que hay en el sistema.

La Entidad nodo guarda las secuencias de los árboles que hay en el sistema.

Figura 15: Entidad Alerta.

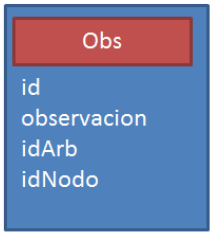

Figura 16: Entidad Obs.

Entidad Alerta

campos
id → tipo entero
descripción → tipo varchar(255)
idArb → tipo entero
idNodo → tipo entero

clave primaria
pkaler
→ id

clave foránea
fkalert0 foránea con pknodo
→ idArb
→ idNodo

Entidad Obs

campos
id → tipo entero
observación → tipo varchar(255)
idArb → tipo entero
idNodo → tipo entero

clave primaria
pkobs
→ id

clave foránea
fkobs0 foránea con pknodo
→ idArb
→ idNodo

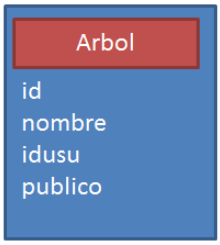

Figura 17: Entidad Arbol.

Entidad Arbol

campos
id → tipo entero
nombre → tipo varchar(255)
publico → tipo boolean
idusu → tipo entero

clave primaria
pkárbol
→ id

clave foránea
fkárbol1 foránea con pkusu
→ idusu

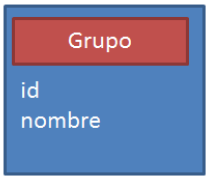

Figura 18: Entidad Grupo.

Entidad Grupo

campos
id → tipo entero
nombre → tipo varchar(255)

clave primaria
pkgrupo
→ id

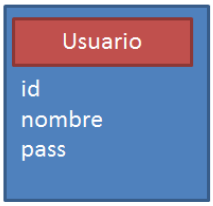

Figura 19: Entidad Usuario.

Entidad Usuario

campos

id → tipo entero
nombre → tipo varchar(255)
pass → tipo varchar(255)

clave primaria

pkusu
→ id

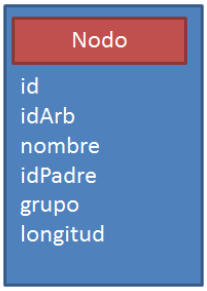

Figura 20: Entidad Nodo.

Entidad Nodo

campos

id → tipo entero
idArb → tipo entero
nombre → tipo varchar(255)
idPadre → tipo entero
grupo → tipo entero
longitud → tipo punto flotante

clave primaria

pknodo
→ id
→ idarb

clave foránea

fkgru0 foránea con pkgrupo
→ id
fknodo0 foránea con pkarb
→ id
fkgru0 foránea con pknodo
→ id
→ idarb

C. Diagramas UML

C.1. Diagrama de clase

C.1.1. Cliente

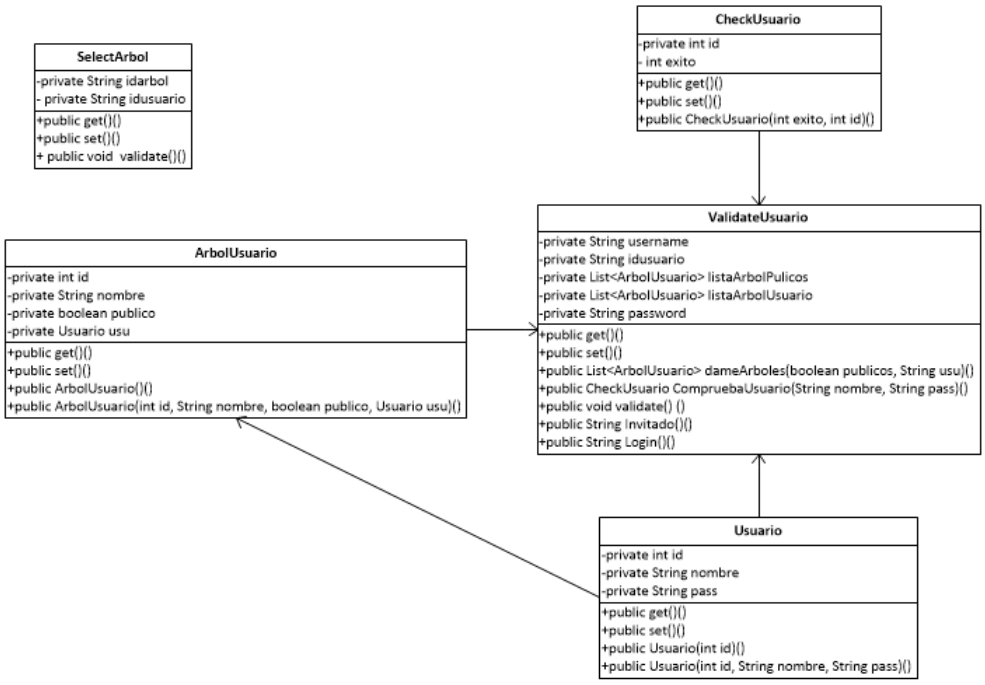

Figura 21: Diagrama de clases del cliente.

C.1.2. Servidor intermediario

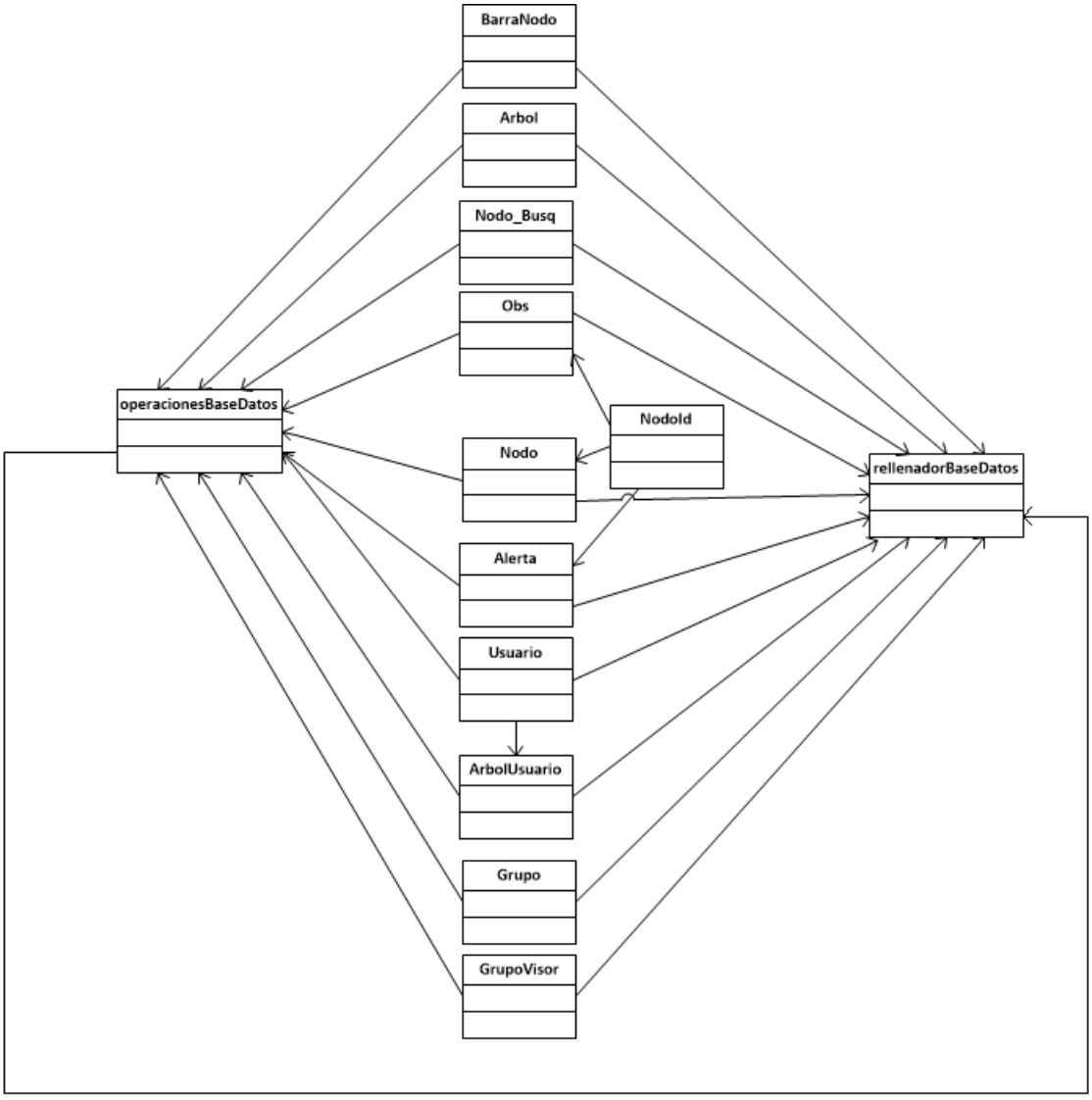

Figura 22: Diagrama de clases global del servidor intermediario.

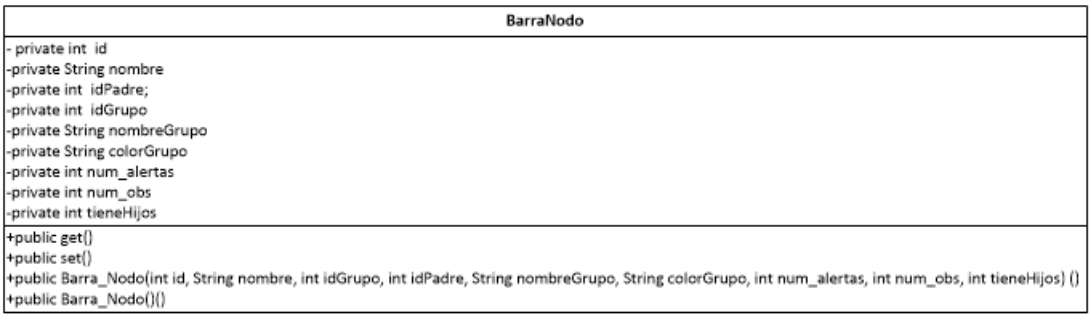

Figura 23: Clase BarraNodo.

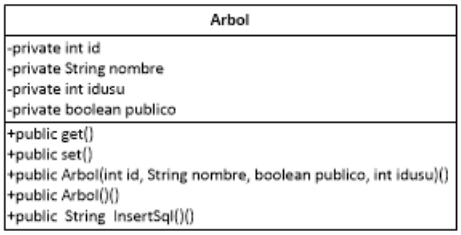

Figura 24: Clase Arbol.

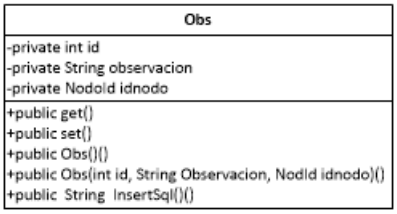

Figura 25: Clase Obs.

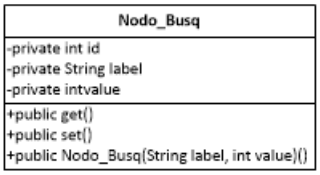

Figura 26: Clase NodoBusq.

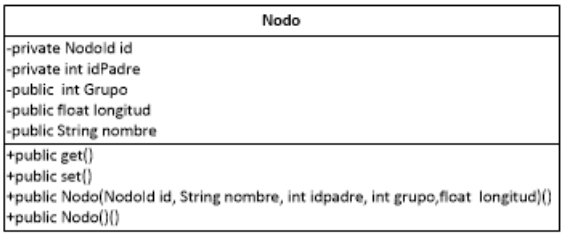

Figura 27: Clase Nodo.

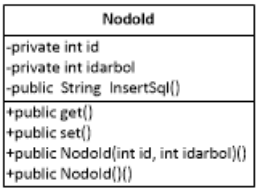

Figura 28: Clase NodoId.

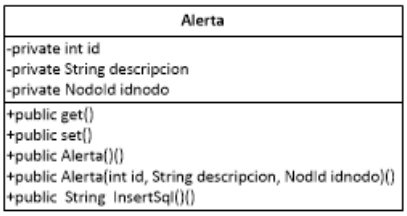

Figura 29: Clase Alerta.

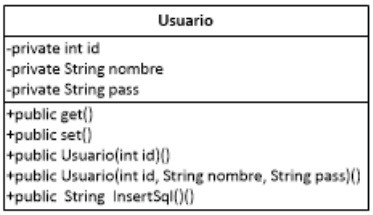

Figura 30: Clase Usuario.

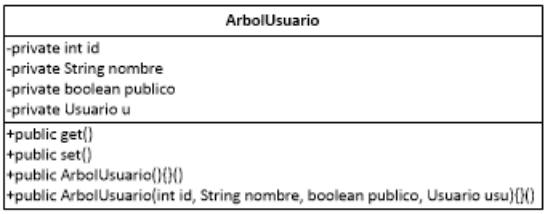

Figura 31: Clase ArbolUsuario.

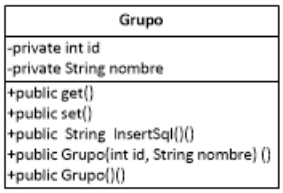

Figura 32: Clase Grupo.

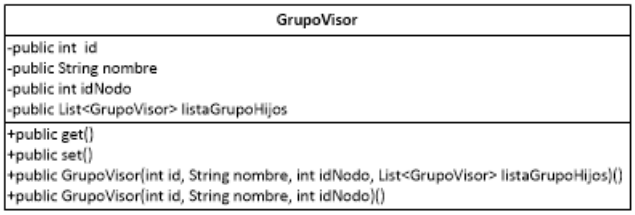

Figura 33: Clase GrupoVisor.

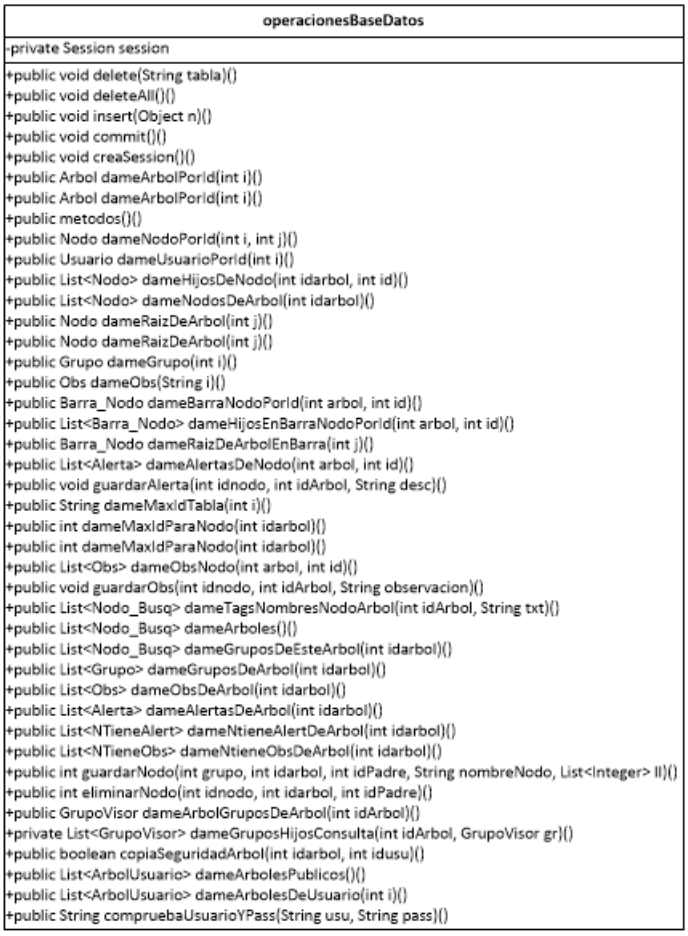

Figura 34: Clase operacionesBaseDatos.

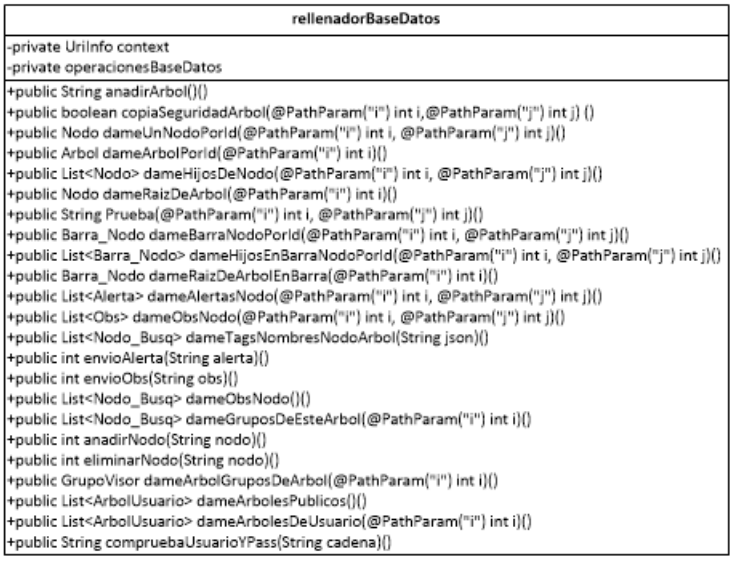

Figura 35: Clase rellenadorBaseDatos.

C.2. Diagrama de casos de uso

C.2.1. Global

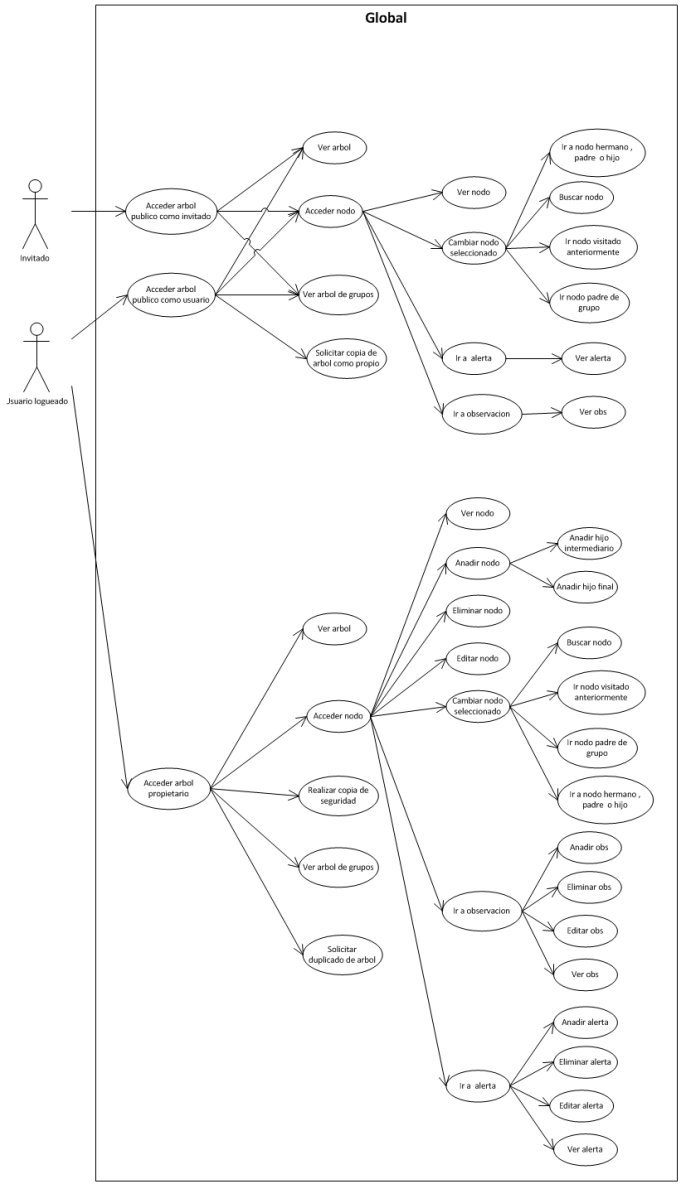

Figura 36: Diagrama entidad-relación.

C.2.2. Parciales

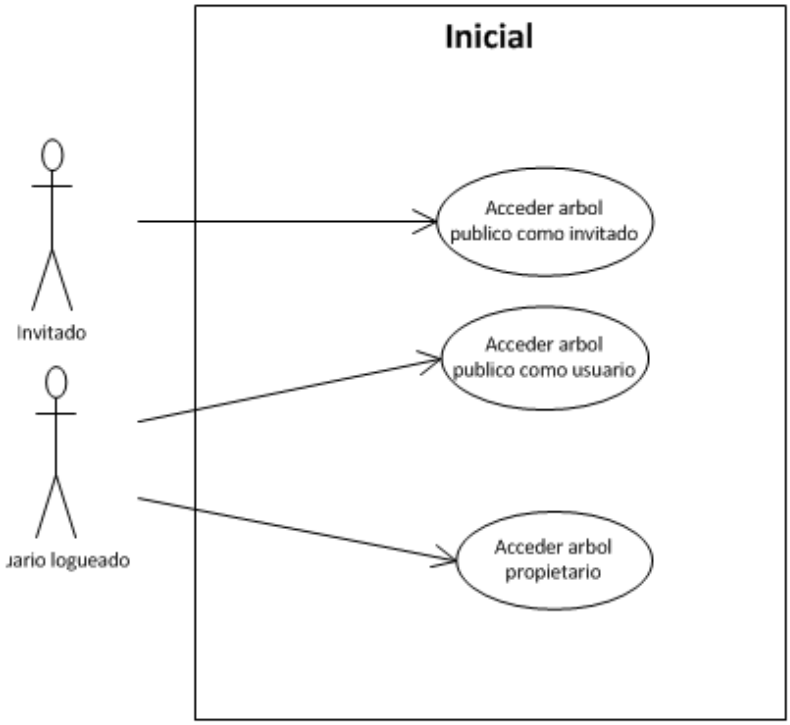

Figura 37: Diagrama Inicial.

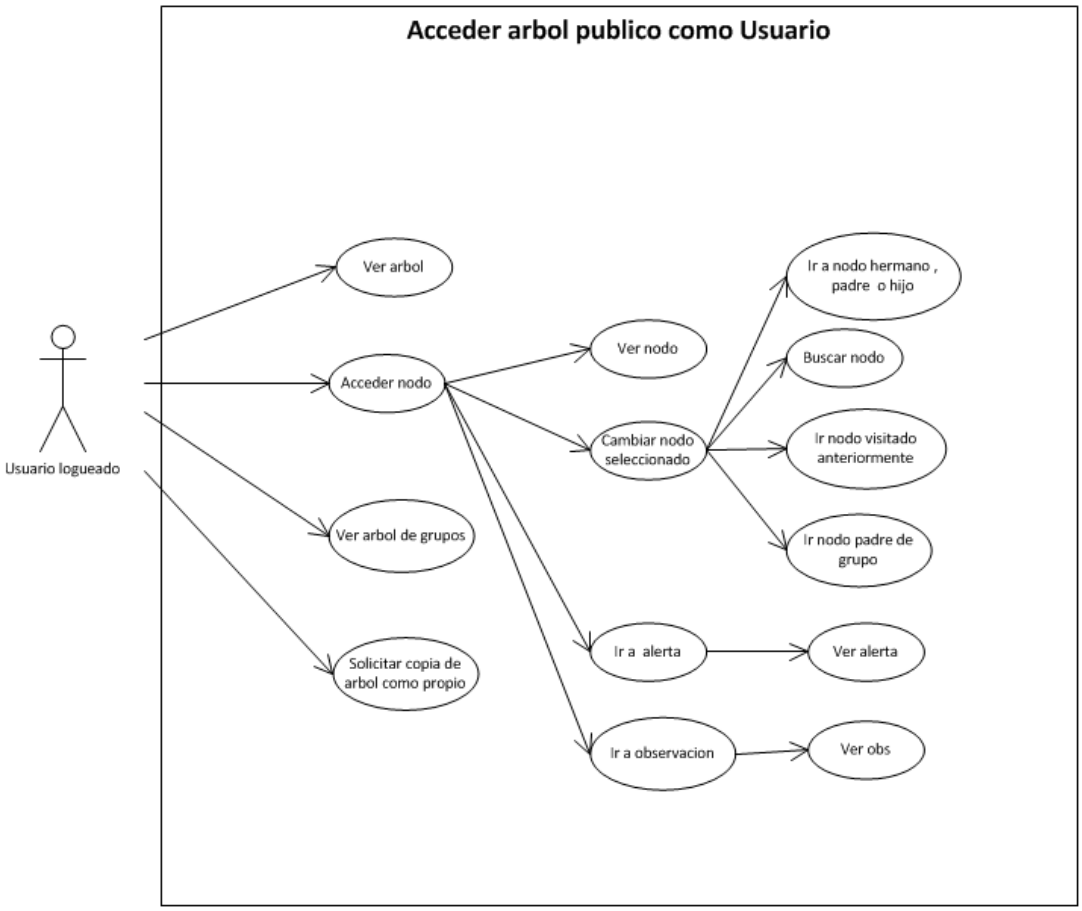

Figura 38: Diagrama Acceder árbol público como usuario.

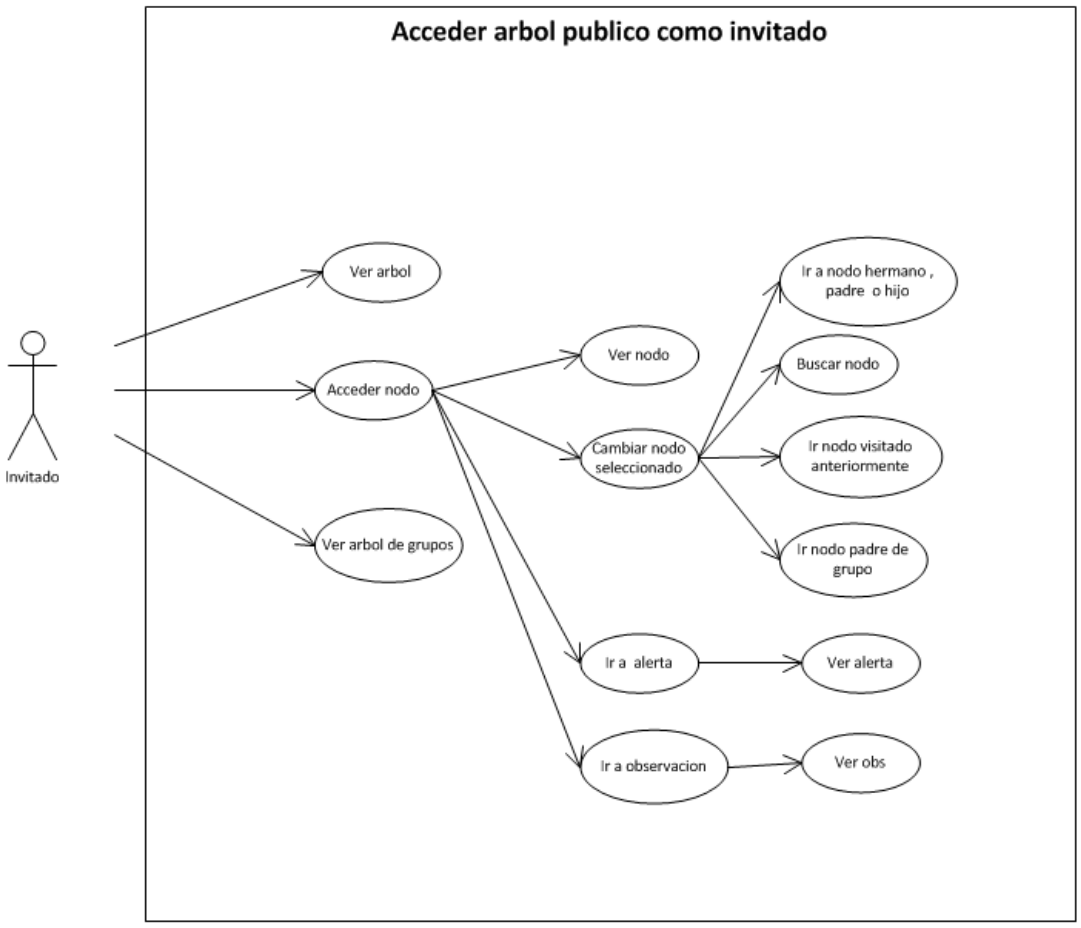

Figura 39: Diagrama Acceder árbol público como invitado.

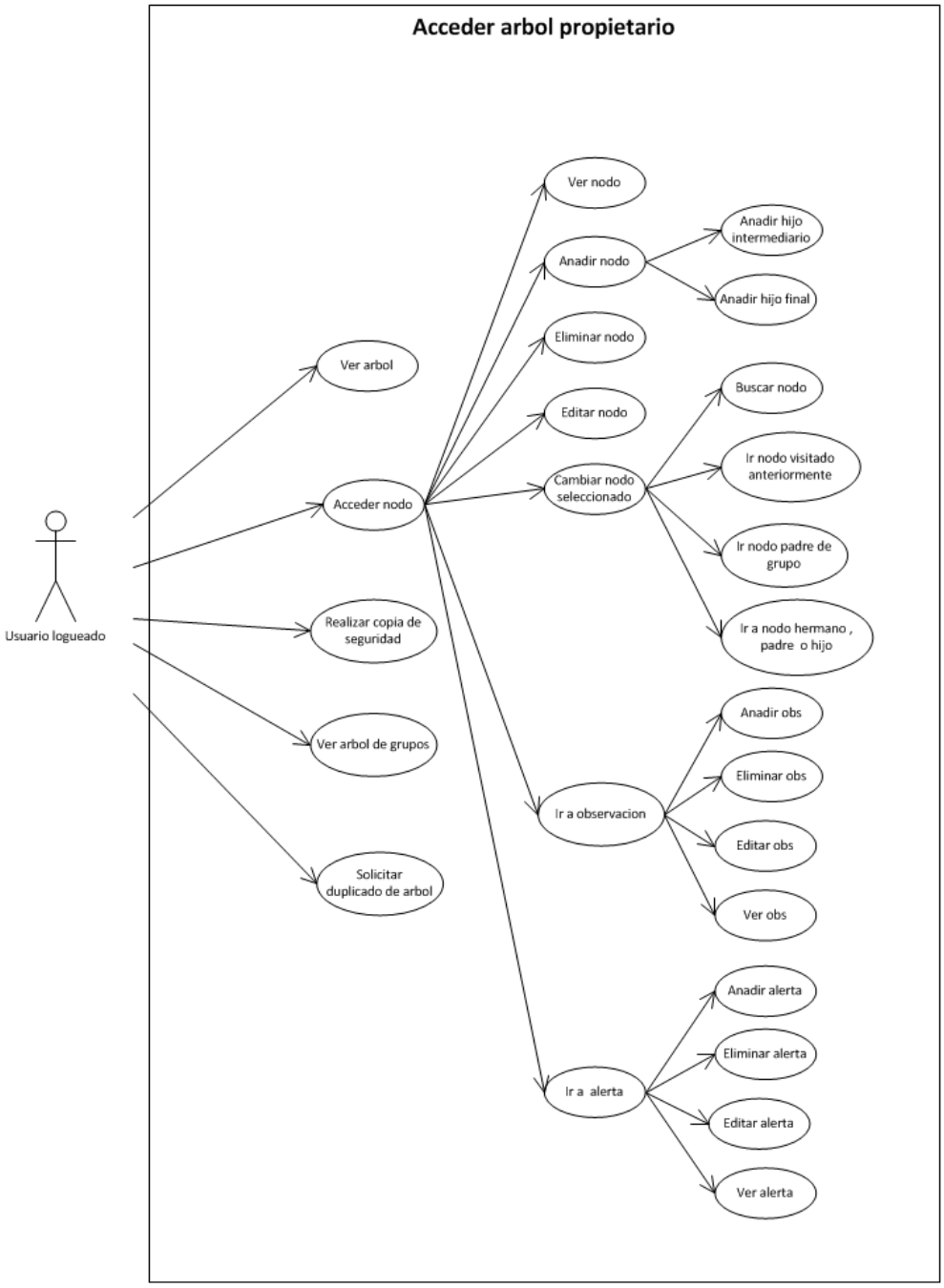

Figura 40: Diagrama Acceder árbol propietario.

D. Manual de usuario

En este documento se describirán los requisitos necesarios para que la aplicación funcione y las funcionalidades de esta.

D.1. Requisitos Software

Las aplicaciones que necesitaremos tener instaladas en nuestra sistema serán las siguientes:

1. Apache Tomcat 7 (o superior)
2. PostgreSQL 9.1.12
3. JDK 7

D.2. Instalación de la aplicación

En primer lugar se procederá a instalar los programas citados en los requisitos de software del sistema.

D.2.1. Base de datos

Ahora se procede a ejecutar el Script llamado "CreadorDeBaseDeDatos.Sql" en el PostgreSQL, el cual creará las tablas necesarias para que funcione la aplicación.

D.2.2. Servidor web y cliente web

Para que tanto el servidor web como el cliente web funcionen en nuestro sistema, se necesita montar los archivos llamados "cliente.war" y "servidor.war" en nuestro servidor web Apache Tomcat.

D.3. Autenticación

Figura 41: Pantalla de login.

1. Campo de texto para introducir el nombre de usuario.
2. Campo de text para introducir la contraseña de usuario.
3. Botón Login
4. Botón Invitado

D.3.1. Invitado

Para autenticarse como invitado se debe presionar sobre el boton Invitado(4).

D.3.2. Usuario

Para autenticarse como Usuario de la aplicación, se debe introducir su nombre de usuario(1) y su contraseña(2) . Presionar en el botón Login(3).

D.4. Seleccionar árbol

En el caso de que se acceda como invitado se muestra la imagen tal como aparece en la Figura 42 y en el caso de acceder como usuario se muestra una imagen como aparece en la Figura 56

D.4.1. Selección de árbol como Invitado

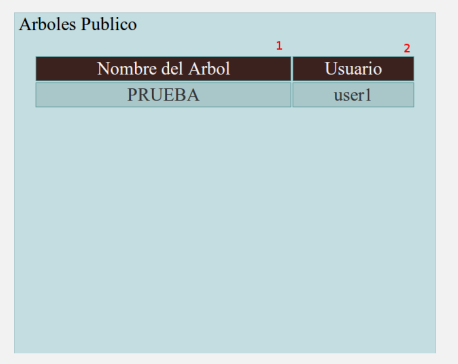

Figura 42: Selección de árbol como Invitado.

1. Columna con los nombre de los árboles.
2. Columna con los nombre de los usuarios propietarios de los árboles.

Para acceder a cualquier árbol de los públicos que posee el sistema se tendrá que presionar sobre la fila del árbol que se desee visualizar.

D.4.2. Selección de árbol como Usuario

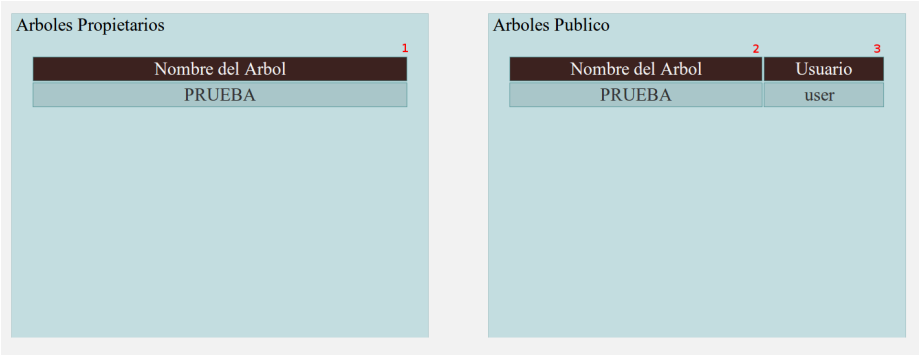

Figura 43: Selección de árbol como Usuario.

1. Columna con los nombre de los árboles que posees como usuario.

2. Columna con los nombre de los árboles de árboles publicos.

3. Columna con los nombre de los usuarios propietarios de los árboles.

Para acceder a cualquier árbol público o de los que posees como usuario se tendrá que presionar sobre la fila del árbol que se deseé visualizar.

D.5. Visor de árboles

Figura 44: Visor de árboles.

1. Visualizador de ancestros de grupos.

2. Visualizador de secuencias.

3. Buscador de secuencias.

4. Historial de secuencias.

5. Visualizador de árbol de grupos.

D.5.1. Visualizador de ancestros de grupos

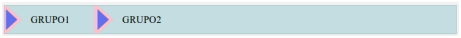

Figura 45: Visualizador de ancestros de grupos.

En esta sección se puede visualizar el grupo al que pertenece y los grupos padres de la secuencia con la que se esta trabajando.

D.5.2. Visualizador de secuencias

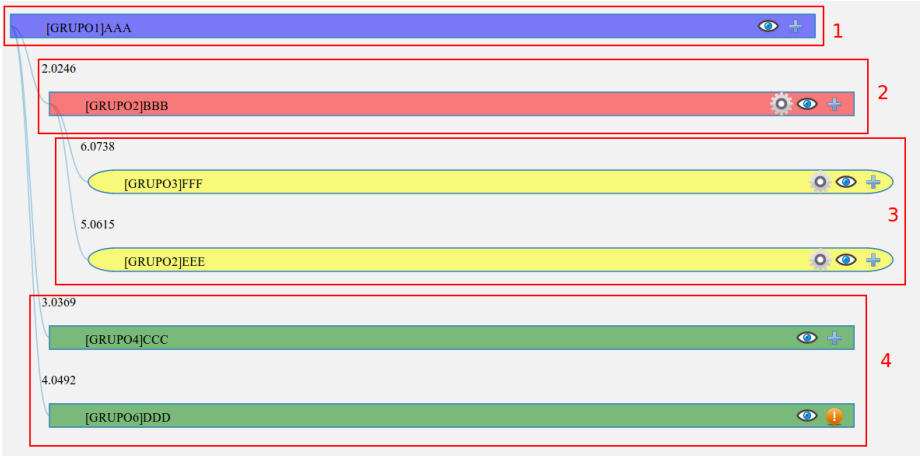

Figura 46: Visor de secuencias.

1. Secuencia padre de la secuencia seleccionada.
2. Secuencia seleccionada.
3. Secuencias hijas de la secuencia seleccionada.
4. Secuencias hermanas de la secuencia seleccionada.

El sistema se basa en ver de una secuencia selección da su padre, hermanos e hijos. Así cuando se presiona sobre la barra del padre, hermano o hijo de una secuencia, la vista se recarga de nuevo pasando esta a ser la secuencia seleccionada y así generando sus padre, hermanos e hijos.

Información de una secuencia

Figura 47: Información de una secuencia.

1. Longitud de la secuencia.
2. Grupo al que pertenece la secuencia.
3. Nombre de la secuencia.

Tipos de barra

Figura 48: Indica que esa secuencia no tiene hijos.

Figura 49: Indica que esa secuencia tiene hijos.

Iconos en la barra

Figura 50: Botón para añadir alertas.

Figura 51: Botón ver alertas.

Figura 52: Botón para añadir observaciones.

Figura 53: Botón para abrir ventana de añadir o eliminar secuencia.

D.5.3. Buscador de secuencias

Figura 54: Buscador de secuencias.

1. Campo de texto para introducir nombre de secuencia.
2. Botón Buscar.

D.5.4. Historial de secuencias

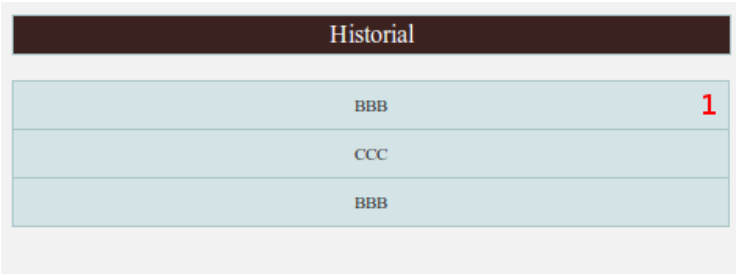

Figura 55: Historial de secuencias.

1. Botón que muestra nombre de una secuencia visitada anteriormente.

En el historial se puede verlas secuencias que se han visualizado a lo largo de la sesión de trabajo.

Si se presiona sobre una fila de una secuencia del historial se carga la secuencia seleccionada en el visualizador.

D.5.5. Visualizador de árbol de grupos

Figura 56: Visualizador de árbol de grupos.

1. Botón que muestra el nombre de un grupo del árbol.

En el visualizador de grupos se muestra un árbol completo de los grupos del árbol con el que se esta trabajando.

Si se presiona sobre un grupo cualquiera del árbol, se carga en el visualizador la secuencia padre del grupo elegido como secuencia seleccionada.

E. Índice de figuras

Índice de figuras

1. Escenario de grupos de investigación y herramientas para la propuesta de solución . . . 11
2. Arquitectura Cliente-Servidor . . . 13
3. Comunicación entre Cliente-Servidor . . . 15
4. Diagrama entidad-relación . . . 16
5. Ejemplo de vista padre-hermanos-hijos . . . 18
6. Implementación de la aplicación . . . 21
7. Ejemplo de un vector de nodos en JSON . . . 25
8. Ejemplo de formato Newick . . . 27
9. Gestión en la realización del proyecto . . . 31
10. Ejemplo de arquitectura Cliente-Servidor . . . 38
11. Ejemplo de arquitectura Cliente-Servidor de dos capas . . . 39
12. Ejemplo de arquitectura Cliente-Servidor de tres capas . . . 39
13. Ejemplo de arquitectura Cliente-Servidor tres capas con MVC . . . 40
14. Diagrama entidad-relación . . . 42
15. Entidad Alerta . . . 44
16. Entidad Obs . . . 44
17. Entidad Arbol . . . 45
18. Entidad Grupo . . . 45
19. Entidad Usuario . . . 46
20. Entidad Nodo . . . 46
21. Diagrama de clases del cliente . . . 48
22. Diagrama de clases global del servidor intermediario . . . 49
23. Clase BarraNodo . . . 50
24. Clase Arbol . . . 50
25. Clase Obs . . . 50
26. Clase NodoBusq . . . 51
27. Clase Nodo . . . 51
28. Clase NodoId . . . 51
29. Clase Alerta . . . 52
30. Clase Usuario . . . 52
31. Clase ArbolUsuario . . . 52
32. Clase Grupo . . . 53
33. Clase GrupoVisor . . . 53
34. Clase operacionesBaseDatos . . . 54
35. Clase rellenadorBaseDatos . . . 55
36. Caso de uso Global . . . 56
37. Caso de uso Inicial . . . 57
38. Caso de uso Acceder arbol publico como usuario . . . 58
39. Caso de uso Acceder árbol público como invitado . . . 59
40. Caso de uso Acceder árbol propietario . . . 60

41. Pantalla de login 63
42. Selección de árbol como Invitado 64
43. Selección de árbol como Usuario 64
44. Visor de árboles 65
45. Visualizador de ancestros de grupos 65
46. Visor de secuencias 66
47. Información de una secuencia 67
48. Indica que esa secuencia no tiene hijos 67
49. Indica que esa secuencia tiene hijos 67
50. Botón para añadir alertas 67
51. Botón para ver alertas 67
52. Botón para ver observaciones 67
53. Botón para abrir ventana de añadir o eliminar secuencia 68
54. Buscador de secuencias 68
55. Historial de secuencias 68
56. Visualizador de árbol de grupos 69

F. Índice de tablas

Índice de tablas

1. Peticiones ordenadas por grupo 14
2. Pros y Contras de RESTFul 24
3. Pros y Contras de SOAP 24
4. Carga de arboles 29
5. Tiempo de respuesta del servidor intermediario 30
6. Resumen de horas por fase de proyecto 32