Universidad Zaragoza
1542

Trabajo Fin de Grado

Formalización y desarrollo de un workflow de inferencia filogenética basado en SaaS

Autor

Álvaro Recuenco Vidosa

Directores

Francisco Javier Fabra Caro
Gregorio de Miguel Casado

Escuela de Ingeniería y Arquitectura

Junio de 2014

“Formalización y desarrollo de un workflow de inferencia filogenética basado en SaaS”
Resumen

Este trabajo aborda la formalización y desarrollo de un sistema de flujo de trabajo de inferencia filogenética empleando el paradigma de Software como Servicio (*SaaS*). Dicho sistema se ha configurado y desplegado sobre un entorno distribuido.

En primer lugar se ha llevado a cabo un estudio sobre entornos similares planteados en la literatura de divulgación científica. Este estudio contiene una descripción de la situación reciente de sistemas similares al planteado que se están desarrollando en la actualidad.

Finalizado el estudio previo, se ha llevado a cabo el análisis y posterior adaptación de un sistema de inferencia filogenética existente. El análisis previo recoge la composición y el funcionamiento del flujo de trabajo con el objetivo de conocer el tratamiento de la entrada y la salida del sistema. No obstante, no se plantea la necesidad de conocer los detalles del procesamiento que implementa cada componente en concreto.

A continuación, se ha realizado un análisis para identificar y definir los usuarios finales, además de documentar y modelar los requisitos que debía satisfacer el sistema. Este análisis se ha realizado empleando técnicas de Ingeniería del Software y en especial aquellas relacionadas con la Ingeniería de Requisitos.

Una vez documentados y modelados los requisitos funcionales y no funcionales del sistema se ha realizado el diseño de la arquitectura interna de cada uno de los componentes que constituyen el sistema. Finalizada la fase de diseño, se ha implementado cada componente individual del sistema definiendo una interfaz de entrada que permite exponer sus funcionalidades como un servicio Web. Cada componente interactúa con el sistema previo como una caja negra, ejecutando los procesos ordenadamente y capturando la salida.

Seguidamente, se ha desarrollado un sistema que implementa un flujo de trabajo completo para la inferencia filogenética de un árbol evolutivo sobre un conjunto de secuencias de ADN mitocondrial humano y se ha expuesto dicho sistema como un servicio Web independiente, integrando todos los componentes previos. Este sistema contiene la lógica de negocio relacionada con la forma de constituir el flujo de trabajo completo para distintos tipos de secuencias biológicas.

Ambos sistemas han sido desplegados en la nube como alternativa para evaluar posibles beneficios en cuanto a escalabilidad, disposición dinámica de recursos, alta disponibilidad de los servicios y reducción de costes económicos. También se ha empleado un sistema de almacenamiento de información externo en la nube para almacenar tanto las soluciones parciales como los resultados finales durante un periodo de tiempo, permitiendo al usuario el acceso a dichos resultados a través de Internet. Por último, se ha realizado un sistema de toma de decisiones mediante ficheros de log. Así se ha planteado un modelo de mejora sencillo basado en el entrenamiento de árboles de decisión.

Índice general

1. Introducción

Este capítulo contiene una introducción con el contexto y los objetivos principales del proyecto, la metodología empleada y, por último, se detalla la estructura del documento.

1.1. Contexto del proyecto

La filogenética es la ciencia encargada del estudio y clasificación de la relación evolutiva entre organismos. En la actualidad los sistemas de inferencia filogenética tienen gran interés, ya que permiten no sólo hacer estudios evolutivos del individuo sobre poblaciones sino también tratar cuestiones más especificas como la detección de enfermedades raras a partir de secuencias de ADN mitocondrial humano. Concretamente, la detección de enfermedades se basa en la ubicación de secuencias de ADN en el árbol inferido. Recientemente se está abordando este tipo de análisis tan costosos computacionales aprovechando el avance de las tecnologías en cuanto al aumento de prestaciones [1].

Como punto de partida, se dispone de un sistema de inferencia filogenética previo desarrollado por el investigador Jorge Álvarez durante la realización de sus estudios de máster. Este sistema denominado *PhyloFlow* [2] permite el estudio de modelos evolutivos en alineamientos de secuencias de ADN mitocondrial. El sistema *PhyloFlow* [2] emplea métodos de selección de modelos evolutivos y de construcción de superárboles. Este sistema se perfila en la Figura 1, y se detallará más adelante en la sección 4.1.1. del documento (pág. 8).

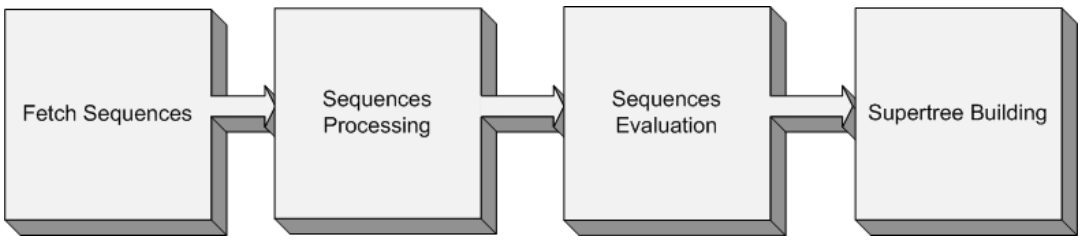


Figura 1. Sistema de inferencia filogenética mediante flujos de trabajo

El sistema *PhyloFlow* [2] está desarrollado mediante flujos de trabajo que permiten una visión conceptualmente sencilla del diseño del sistema. El flujo de trabajo está formado por cuatro componentes que permiten la descarga, procesamiento biológico y evaluación de secuencias además de la generación de superárboles. Cada componente del flujo de trabajo es independiente del resto, y la ejecución de un flujo completo de trabajo implica la ejecución ordenada de todos los componentes que forman parte del sistema.

Además este sistema emplea la herramienta *HTCondor* [3] basada también en flujos de trabajo que permite la distribución de los trabajos en distintas máquinas. Sin embargo esta solución tiene asociados una serie de problemas. En primer lugar, es un sistema acoplado en el que cada uno de los componentes depende un planificador de tareas desarrollado con la herramienta *DAGMan* [4] que establece las dependencias entre cada uno de los trabajos. En segundo lugar, el sistema no está distribuido y, por tanto, únicamente se puede ejecutar en una máquina computacionalmente potente que albergue la instalación del sistema. Además es un sistema que no es accesible remotamente y requiere la instalación del mismo para su utilización. Otro problema encontrado se debe a que la interfaz de entrada de cada uno de los componentes del flujo de trabajo no está suficientemente documentada.

Este trabajo aborda un análisis pormenorizado de las características detalladas del sistema *PhyloFlow* [2], sin profundizar en aspectos metodológicos y procedurales del procesamiento biológico realizado por el sistema para realizar el análisis filogenético. El objetivo del trabajo es desarrollar un sistema empleando el paradigma *SaaS* para obtener una solución que permite desplegar el sistema en un entorno distribuido reduciendo el acoplamiento y dando acceso al sistema de forma remota a través de un servicio Web. El entorno distribuido sobre el cual se ha desplegado el sistema es una plataforma de computación escalable en la nube que permite el aprovisionamiento de recursos dinámicamente.

1.2. Objetivos

El objetivo principal del proyecto ha sido el análisis y desarrollo de una colección de servicios basados en tecnologías Web para la inferencia de árboles filogenéticos con la información biológica bajo el paradigma de Software como Servicio (*SaaS*). Los objetivos concretos plantean la búsqueda de entornos similares en la literatura científica, el análisis del sistema previo y los requisitos de una solución *SaaS*, el modelado bajo el paradigma considerado y finalmente la implementación y evaluación de un prototipo del sistema mediante un flujo de trabajo.

Desde el primer momento se tomó la decisión de emplear un modelo basado en *SaaS* para la publicación de dichos servicios y emplear dicho modelo para conseguir una solución escalable que permite su aplicación en un entorno distribuido.

El proyecto contempla también el desarrollo y despliegue del sistema en un entorno distribuido. Concretamente, se optó por emplear una plataforma de computación elástica en la nube que permita aumentar y disminuir los recursos dinámicamente para adaptarse a las exigencias computacionales.

Además se ha llevado a cabo la validación del sistema a partir de una batería de pruebas automáticas desarrolladas durante el proyecto. Uno de los objetivos principales era validar el correcto funcionamiento del sistema en un entorno distribuido y evaluar las prestaciones del mismo para determinar el equilibrio entre el coste temporal y el coste económico.

Por último se ha desarrollado un sistema de ayuda a la toma de decisiones en base a los ficheros de log obtenidos durante la validación y evaluación del sistema. De este modo se ha planteado un modelo de mejora sencillo basado en aprendizaje que facilita la elaboración de estrategias sobre el tipo de plataformas computacionales en las cuáles desplegar el sistema.

1.3. Metodología

La metodología utilizada establece una organización en fases del proyecto y sus correspondientes hitos facilitando su seguimiento, identificación de carencias y retroalimentación del desarrollo, según un modelo de ciclo de desarrollo en cascada. Se ha seguido un flujo de trabajo basado en las fases principales que se consideran en Ingeniería del Software haciendo especial hincapié en aquellas relacionadas con la Ingeniería de Requisitos. Esto ha proporcionado como resultado un análisis profundo de las dependencias, características, necesidades y limitaciones de la solución.

1.4. Estructura de la memoria

La memoria está estructurada en nueves capítulos que corresponden esencialmente a las fases principales del proyecto además de la gestión del proyecto y la bibliografía del trabajo.

El segundo capítulo (pág. 4) contiene un glosario con la definición de aquellos conceptos más importantes relacionados con el trabajo.

El tercer capítulo (pág. 5) se corresponde al estado del arte. Este capítulo contiene la información más relevante encontrada en la literatura científica con planteamientos de entornos similares al desarrollado.

El capítulo de análisis (pág. 8) contiene la información recogida del análisis del sistema previo. Esta sección contiene una breve descripción del sistema que permite comprender suficientemente su funcionamiento y la documentación que forma parte del análisis del sistema que ha sido desarrollado. Finalmente, contiene la descripción de los usuarios finales del sistema, así como la documentación y modelado en casos de uso de los requisitos funcionales y no funcionales del sistema.

En el capítulo de diseño de la solución (pág. 15) se describe el paradigma *SaaS* y se documenta el diseño de la solución propuesta. La solución propuesta contiene la descripción, relaciones entre elementos y los diagramas de la arquitectura diseñada.

El sexto capítulo (pág.22) contiene los aspectos más interesantes de la implementación del sistema y las decisiones de implementación que se han tomado para el desarrollo del proyecto.

El capítulo de despliegue y evaluación (pág. 27) explica la configuración y despliegue del sistema en un entorno distribuido, la nube comercial de *Amazon Web Services* (*AWS*). Además se documenta el proceso de validación del sistema y la evaluación del sistema y los resultados obtenidos.

En el octavo capítulo (pág. 31) contiene la documentación relacionada con la gestión y planificación del proyecto. Se explica la metodología utilizada y se muestra la planificación del proyecto a través de un diagrama de Gantt. Finalmente se exponen las conclusiones finales del proyecto.

Por último, se encuentran las secciones de bibliografía (pág. 35) y de anexos (pág. 37).

2. Glosario

En este apartado de la memoria se encuentra la definición de aquellos conceptos de ambos que se han considerado imprescindibles para la compresión del proyecto.

- ***ADN***: ácido desoxirribonucleico. Es una macro molécula que forma parte de todas las células, y es usada para su crecimiento y funcionamiento. En ella se encuentra toda la información genética y es, por tanto, el componente responsable de transmisión hereditaria.
- ***ADN mitocondrial humano*** *(ADNmt)*: en algunas células existen unos orgánulos denominados mitocondrias. En estos orgánulos se produce la oxidación de las moléculas de glucosa sus funciones internas. Este tipo de ADN es independiente del ADN celular.
- ***API*** *(Application Programming Interface)*: especifica como un componente software debe interactuar con otro sistema. Define una serie de funciones y procedimientos que pueden ser invocados.
- ***Árbol filogenético***: estructura arborescente que refleja la relación evolutiva entre distintos organismos, almacenados en sus hojas.
- ***AWS*** *(Amazon Web Services)*: conjunto de servicios Web ofrecidos por la compañía Amazon para emplear en la nube.
- ***Cloud computing***: o computación en la nube es un paradigma de computación distribuida en red en el que un programa o aplicación en un servidor o servidores conectados. Estos servidores físicos se encuentran virtualizados, por tanto, se pueden configurar y dividir en varios servidores virtuales independientes.
- ***Filogenética***: es la ciencia encargada del estudio y clasificación de la relación evolutiva entre organismos.
- ***GenBank***: es una base de datos de secuencias de acceso abierto. Esta base de datos ha sido creada y mantenida por el *National Center of Biotechnology Information* (*NCBI*). *GenBank* recibe secuencias producidas en laboratorios de todo el mundo de más de 100.000 organismos distintos.
- ***Modelo evolutivo****:* modelo matemático que pretende ajustar la relación entre distintos organismos a la evolución real. Se sustenta en una serie de parámetros como la frecuencia de cada nucleótido o aminoácido de la secuencia y la tasa de mutaciones, entre otras.
- ***REST*** *(Representational State Transfer)*: es un patrón de diseño o estilo de arquitectura para el desarrollo de sistemas basados en servicios Web.
- ***SaaS*** *(Software as a Service)*: es un modelo de software en el que el software está alojado en la nube de los proveedores del software o proveedor de servicios de aplicaciones.
- ***SOAP*** *(Simple Object Access Protocol):* es un protocolo que define un conjunto de reglas para el intercambio y procesamiento de mensajes estructurados en servicios Web.
- ***URI*** *(Uniform Resource Identifier)*: identificador uniforme y universal de un recurso que permite localizar y acceder al recurso.

3. Estado del arte

El objetivo principal del proyecto ZARAMIT era el de desarrollar un sistema capaz de inferir filogenias a partir de secuencias de ADN mitocondrial humano [1]. Estás, a su vez serán usadas para llevar a cabo estudios evolutivos y detectar mutaciones potencialmente dañinas.

Las mitocondrias son orgánulos que se encuentran en la mayoría de las células eucariotas y son los responsables de la generación de la mayor parte de la energía química de la célula. Éstas poseen un genoma independiente heredado y en segundo lugar, existen moléculas de ADN en un entorno reactivo, donde las tasas de mutación son un orden de magnitud superior al de las de ADN nuclear. Por tanto, su resolución permite distinguir individuos separados estrechamente relacionados. Por otra parte, las mutaciones mitocondriales son una de las principales causas de enfermedades genéticas "raras". El objetivo de ZARAMIT era fusionar ambos enfoques.

El enfoque descrito anteriormente ha sido explotado en el trabajo de investigación "Análisis filogenético molecular: Diseño e implementación de algoritmos escalables y fiables y verificación automática de propiedades de una filogenia" [2]. Como resultado de dicho trabajo se desarrolló el sistema de inferencia filogenéticas mediante flujos de trabajos denominado *PhyloFlow* [2].

Un sistema de flujo de trabajo científico es una forma especial de gestión de un sistema diseñado especialmente para crear y ejecutar una serie de etapas computacionales o de manipulación de datos en aplicaciones científicas. Un ejemplo de sistema de flujo de trabajo científico son los sistemas de tipo bioinformático que están centrados en un dominio específico de la ciencia.

Anduril [5] es un entorno de flujos de trabajo de código abierto basado en componentes para el análisis de datos científicos [6]. *Anduril* [5], desarrollado en el Laboratorio de Sistemas Computacionales de Biología de la Universidad de Helsinki, fue diseñado para permitir el análisis de datos de manera sistemática, flexible y eficiente en el campo de la investigación biomédica. El sistema de flujo de trabajo actual proporciona componentes para distintos tipos de análisis como la secuenciación, expresión de genes, *SNP (Single Nucleotide Polymorphism*), análisis de imágenes de células, etc.

En *Anduril* [5], las etapas de procesamiento están implementadas utilizando componentes, los cuales son ejecutables, reutilizables y se puede escribir en cualquier lenguaje de programación. Los componentes están conectados entre sí en el flujo de trabajo o en una red de componentes. Así, el motor de *Anduril* [5] se encarga de ejecutarlos y se comunican a través de ficheros. El núcleo del motor *Anduril* [5] está implementado en *Java* [7], sin embargo los componentes están implementados en una gran variedad de lenguajes de programación como *Perl* [8], *Python* [9] o *Matlab* [10].

Galaxy [11] es una plataforma abierta, basada en la Web para datos de investigaciones biomédicas. Esta plataforma es un flujo de trabajo científico para la integración y análisis de datos que tiene como objetivo hacer accesible a los investigadores que no tiene experiencia en el campo de la programación. A pesar de que fue desarrollado inicialmente para la investigación en genómica, actualmente es utilizado como un sistema de gestión de flujos de trabajo en otras aplicaciones bioinformáticas. *Galaxy* [11] es también una plataforma para la integración de datos biomédicos. Soporta la subida de información desde el ordenador del usuario a través de una URL y otros recursos online.

Los sistemas descritos anteriormente ofrecen una gran variedad de posibilidades para realizar cálculos relacionados con el análisis del genoma. Ambos sistemas son de código abierto y están implementados con

lenguajes de programación modernos facilitando a los desarrolladores su reutilización. Sin embargo su utilización implica la instalación del sistema en una máquina con altas prestaciones debido a que los procesos de análisis del genoma son computacionalmente costosos.

Ninguno de ellos ofrece un servicio Web desde el que acceder a las funcionalidades de análisis que ofrecen dificultando su integración en sistemas externos. En este trabajo se va a abordar dicho problema empleando una aproximación bajo el paradigma *SaaS* y basada en la computación distribuida en la nube, que permita al usuario final realizar un análisis filogenético sin requerir la instalación de ningún sistema.

Debido a la abundancia de secuenciaciones del genoma humano disponibles, la necesidad de almacenar y procesar esta gran cantidad de datos y facilitar el acceso a herramientas de análisis biomédico está suponiendo un reto complicado de alcanzar. A causa de la variabilidad del volumen de datos en los resultados y las exigencias computacionales y de almacenamiento se está optando por realizar métodos más fiables y dinámicos para realizar este tipo de análisis.

En la actualidad se han propuesto plataformas de flujos de trabajo basados en la nube que permitan la realización de este tipo de análisis ofreciendo una ejecución fiable, escalable y totalmente automatizable [12]. Para afrontar los problemas comentados anteriormente, se ha propuesto una plataforma que emplea un flujo de trabajo bioinformático basado en la nube para el análisis a gran escala de datos denominados *NGS* (*Next Generation Sequencing*). Esta plataforma integra *Galaxy* [11], *Globus Provision* [13], una herramienta para desplegar *clusters* de computación en la nube, y un conjunto de herramientas y módulos para ofrecer una solución a investigadores en el campo de la biomedicina.

En *Galaxy* [11], como en la mayoría de flujo de trabajos científicos relacionados con el genoma, los recursos necesarios pueden variar drásticamente en tiempo de ejecución. A menudo es ineficiente en términos de la utilización de recursos y coste, establecer un límite máximo de recursos. La computación en la nube ofrece un modelo alternativo para adaptarse dinámicamente a la demanda de recursos de los flujos de trabajo. Desplegar la plataforma en la nube ofrece una serie de beneficios como la configuración de recursos en función de la demanda, un modelo de pago basado en la utilización de recursos y el aumento de la velocidad de procesamiento de datos.

La integración de flujos de trabajos científicos y la computación en la nube permite disponer de forma rápida de los recursos necesarios tanto a nivel computacional como a nivel de almacenamiento, dota a los sistemas de escalabilidad y emplea un modelo de pago por recursos utilizados. Para afrontar la variabilidad en el volumen de datos de los resultados y las exigencias computacionales y de almacenamiento han empleado la herramienta *HTCondor* [3].

HTCondor [3] es una herramienta de computación de alto rendimiento sobre grandes colecciones de recursos computacionales distribuidos. Mediante la integración de *Galaxy* [11] con el planificador de recursos de *HTCondor* [3], los trabajos específicos son ejecutados en paralelo usando nodos de computación distribuida. La estrategia propuesta dota de una escalabilidad automática, lo que permite incrementar la utilización de recursos y la velocidad de procesamiento significativamente, especialmente en herramientas intensivas en cálculo, como los análisis del genoma humano.

Debido a que la mayoría de los trabajos de las herramientas empleadas por *Galaxy* [11] son intensivos en CPU y memoria, la capacidad computacional necesaria puede ser superior a la inicial en el despliegue del sistema. Para incrementar las prestaciones del sistema bajo el punto de vista de la velocidad de cálculo y el coste económico, han optado por integrar *Galaxy* [11] con la herramienta *HTCondor* [3], de tal forma, que ciertos trabajos se pueden ejecutar a través de *HTCondor* [3] en *clusters* remotos con unas prestaciones superiores. La Figura 2 muestra la arquitectura del sistema.

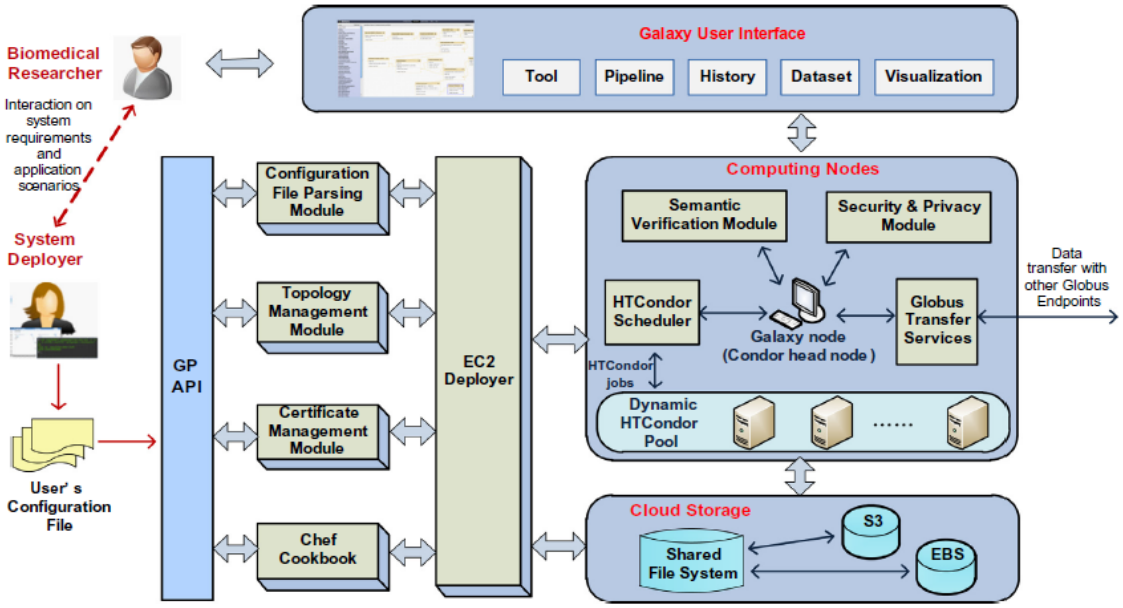


Figura 2. Arquitectura de un sistema de flujo de trabajo bioinformático basado en la nube (*Galaxy* [12])

En conclusión, ya existen sistemas de flujos de trabajos científicos desplegados en plataformas de computación en la nube similares al trabajo que se desea realizar como *Galaxy* [11]. Sin embargo, este sistema no ofrece sus funcionalidades a través de un servicio Web impidiendo su integración en sistemas externos. La utilización de una plataforma de computación en la nube permite aprovisionar recursos de forma dinámica para adaptarse a las exigencias computacionales del sistema y la variabilidad del volumen de datos en los resultados.

4. Análisis

En este capítulo se ha documentado la fase de análisis previo que se ha realizado durante el desarrollo del proyecto. Este capítulo contiene tanto en análisis del sistema existente como la captura, documentación y modelado de los requisitos funcionales y no funcionales del sistema.

4.1. Sistema previo

Se ha realizado una fase de análisis del sistema previo para capturar información del tratamiento que realiza este, tanto de la entrada como de la salida de datos. Durante esta fase, la información proporcionada por el investigador Jorge Álvarez ha resultado esencial debido a que me ha facilitado la compresión de *PhyloFlow* [2].

4.1.1. Descripción

En esta sección se describe el sistema previo y su composición. El sistema está formado por cuatro componentes principales, cuya unión constituye el flujo de trabajo completo para la inferencia filogenética de un árbol evolutivo sobre un conjunto de secuencias de ADN mitocondrial humano. La Figura 3 muestra cada uno de los cuatros componentes y las relaciones que se establece entre cada uno de ellos.

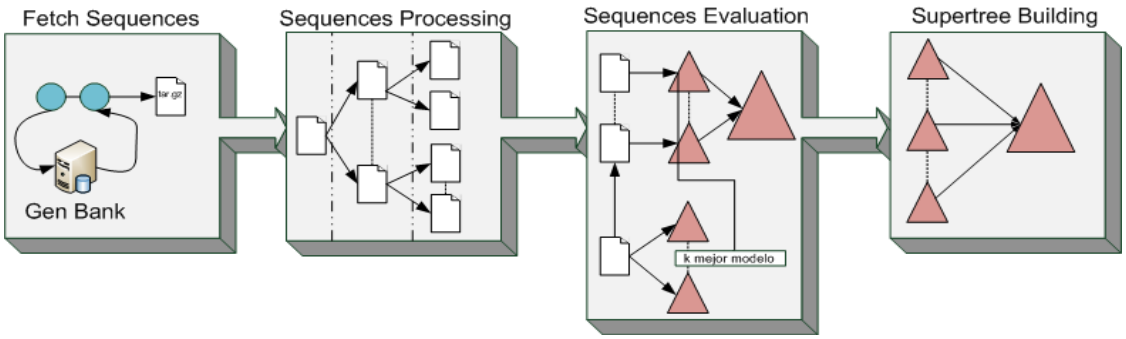


Figura 3. Sistema de inferencia filogenética mediante flujos de trabajo

En la fase *Fetch Sequences*, las secuencias de ADN se almacenan en ficheros comprimidos. A continuación, en la fase *Sequences Processing* dichos ficheros de secuencias se procesan utilizando métodos biológicos quedando divididos en subconjuntos más pequeños. En la fase *Sequences Evaluation* se realiza la evaluación de modelos evolutivos para cada fichero obtenido en la fase previa. En esta fase se generan estructuras arborescentes que al igual que las secuencias son almacenadas en ficheros. Finalmente la fase de *Supertree Building* genera un superárbol basado en las soluciones obtenidas en la fase anterior.

El sistema *PhyloFlow* [2] está compuesto por una serie de directorios donde se almacenan los resultados de cada una de los componentes, y que a su vez sirven de entrada para el siguiente componente del flujo de trabajo. Por tanto, es está estructura de directorios la que permite la comunicación de los resultados entre cada uno de los componentes empleando identificadores de ficheros para seleccionar aquellos que corresponden a los resultados parciales dentro del directorio.

El resultado producido por un componente puede ser la entrada para el siguiente componente del flujo de trabajo, pero también puede ser una nueva entrada para el mismo componente. Este último caso permite que un componente pueda realizar más de un procesamiento sobre una entrada. Por ejemplo, en el segundo componente se podría realizar una primera fase de división por genes y a continuación realizar una fase de alineamiento.

El sistema actualmente emplea secuencias de ADN mitocondrial humano para realizar el análisis filogenético, sin embargo, también se dispone de secuencias sintéticas cuyo procesamiento es menos intensivo en cálculo y permiten validar el funcionamiento del sistema.

El primer componente es el encargado de obtener los tipos de secuencias solicitados por el cliente. Si el tipo de secuencia solicitada es de ADN mitocondrial se encarga de acceder a la base de datos *GenBank,* seleccionar el tipo de secuencia que corresponde con el ADN mitocondrial humano, descargar la base de datos y almacenar la descargar en un fichero comprimido. Por el contrario, si el tipo de secuencia seleccionado es de tipo sintético se encarga de copiar dichos datos de un repositorio local del sistema y los almacena comprimidos.

El segundo componente se encarga de realizar el procesamiento biológico sobre las secuencias de entrada seleccionado por el usuario. Este procesamiento depende del tipo de secuencia de entrada. Actualmente existen 4 tipos diferentes de procesamientos:

- División en 1D consiste en dividir el fichero de secuencia por filas o columnas.
- División en 2D consiste en dividir el fichero de secuencia por filas y columnas obteniendo *ixj* ficheros, dónde *i* es el número total de divisiones por filas y *j* el número total de divisiones por columnas.
- Alineamiento.
- *PRD (Padded-Recursive-DCM3 Decomposition*): división en sets con superposición.

La división por filas implica la división del fichero de secuencias en set o conjunto de secuencias (filas). La división por columnas implica una reducción en la longitud de las secuencias. Este componente emplea técnicas de *map-reduce* para reducir los costes temporales en base a procesar conjunto de datos más pequeños.

El tercer componente se encarga de realizar una evaluación con modelos biológicos para cada fichero obtenido en la fase previa, además opcionalmente se puede realizar un análisis estadístico. El funcionamiento interno de este componente es el siguiente:

- Se evalúan 88 modelos y se selecciona el mejor de todos ellos en base a la puntuación del árbol según *Maximum Likelihood* [14].
- Análisis estadístico mediante la técnica de *bootstrapping* que consiste en alterar el orden de las columnas mediante el intercambio de columnas cercanas. Se generan un número de *r* muestras distintas entre sí. Los biólogos aconsejan usar un número *r* entre 100 y 1000, pero según algunos estudios a partir de 500 se obtiene un beneficio estadísticamente no significativo. Para cada muestra generada se evalúa el mejor modelo del paso anterior y se obtiene un árbol por muestra.
- Generación de un único árbol consensuado a partir de los ficheros de salida para el mejor modelo seleccionado aplicando conservación de ramas.

El componente número cuatro también denominado *Supertree Building* se encarga de construir un único árbol filogenético que aglutina todas las soluciones parciales obtenidas como resultados de la ejecución de los componentes anteriores en fases previas.

Por último cabe destacar que los ficheros de secuencia tanto de entrada como de salida emplean el formato FASTA. Mientras que las estructuras arborescentes generadas como soluciones parciales y resultados finales se almacenan en formato NEWICK. Ambos formatos se describen en detalle en la sección número 1 de **Anexo I. Análisis** (pág. 37).

4.2. Análisis del sistema

Una vez realizado el estudio del sistema previo, se ha realizado un análisis sobre el sistema. Este análisis contiene la descripción de los usuarios finales y los grupos de interés o *stakeholders* del sistema, la documentación de los requisitos funcionales y no funcionales que debe satisfacer el sistema y el modelado de estos en casos de uso.

4.2.1. Usuarios finales

Los usuarios finales del sistema son aquellos usuarios que van a interactuar directamente con él. Por tanto, se han descrito tres tipos de usuarios finales posibles:

- Usuario administrador: es aquel usuario que realiza de intermediario entre el sistema y el biólogo que desea obtener los resultados para proceder a realizar un análisis exhaustivo de los mismos. Este tipo de usuario está familiarizado con el dominio tecnológico, pero por el contrario puede no tener nociones del dominio de la aplicación.
- Biólogo: es aquel usuario que se encarga de proponer un experimento, determinando las herramientas a usar en la ejecución del flujo de trabajo para luego interpretar los resultados obtenidos por el sistema en base a sus conocimientos sobre el dominio de la aplicación.
- Otros sistemas externos o internos que integren componentes del flujo de trabajo.

4.2.2. Stakeholders

El grupo de interés o *stakeholders* son aquellas personas, sistemas o documentación que están estrechamente relacionados con el sistema desarrollado pero puede que no sean los propios usuarios finales del mismo. Los *stakeholders* pueden proporcionan información respecto del sistema a desarrollar o interactuar directamente o indirectamente con él. Los *stakeholders* definidos para el sistema son:

- Usuario finales
- Sistema filogenético existente
- Documentación del sistema previo

4.2.3. Requisitos

En esta sección se han seleccionado aquellos requisitos funcionales y no funcionales que debe satisfacer el sistema. La Tabla 1 contiene numerados los requisitos funcionales y la Tabla 2 contiene los requisitos no funcionales del sistema.

Requisitos funcionales

RF-1	El sistema debe ofrecer una colección de servicios al usuario que permitan el análisis del ADN mitocondrial humano.
RF-2	El sistema debe ofrecer un servicio de descarga de secuencias de ADN mitocondrial humano.
RF-3	El sistema debe ofrecer un servicio de procesamiento de secuencias de ADN mitocondrial humano.
RF-4	El sistema debe ofrecer un servicio de evaluación y construcción de soluciones parciales.
RF-5	El sistema debe ofrecer un servicio de consenso e integración de soluciones parciales.
RF-6	El sistema debe ofrecer un servicio de construcción de un único árbol filogenético a partir de soluciones parciales.
RF-7	El sistema debe ofrecer un servicio que realice un flujo de trabajo completo que integre el resto de servicios ofrecidos.
RF-8	El sistema debe ser capaz de almacenar resultados intermedios.
RF-9	El sistema debe ser capaz de notificar al usuario los resultados generados como respuesta a su petición.
RF-10	El sistema debe capturar la salida de error del sistema de inferencia filogenética existente.

Tabla 1. Requisitos funcionales del sistema

Requisitos no funcionales

RNF-1	El sistema debe integrar el sistema de inferencia filogenética existente.
RNF-2	El sistema debe ser capaz de interoperar con el sistema de inferencia filogenética existente.
RNF-3	El sistema debe almacenar en un sistema de almacenamiento externo los resultados intermedios que permita el acceso a los mismos.
RNF-4	El sistema debe emplear un protocolo de comunicación asíncrono debido al coste computacional derivado del análisis del ADN mitocondrial.
RNF-4	El sistema debe notificar a los usuarios empleando el protocolo SMTP para el envío de correos electrónicos.
RNF-5	El sistema será diseñado desde el punto de vista *SaaS*.
RNF-6	El sistema debe definir y emplear un formato de intercambio estándar.
RNF-7	El sistema debe ser robusto y realizar el tratamiento de excepciones pertinentes.
RNF-8	El sistema debe ser portable y flexible.

Tabla 2. Requisitos no funcionales del sistema

4.2.4. Casos de uso

Esta sección contiene el modelado de los requisitos descritos anteriormente en casos de uso. Los casos de uso dan una visión gráfica de los pasos e interacciones entre los actores del sistema y el propio sistema para lograr un objetivo. Los actores del sistema pueden ser personas humanos u otros sistemas externos.

Caso de uso: nivel 0

Los casos de uso de nivel cero sirven para mostrar las funcionalidades básicas del sistema y los pasos que se han de seguir para obtener un objetivo concreto. La Figura 4 muestra el modelado en casos de uso de nivel 0 del sistema.

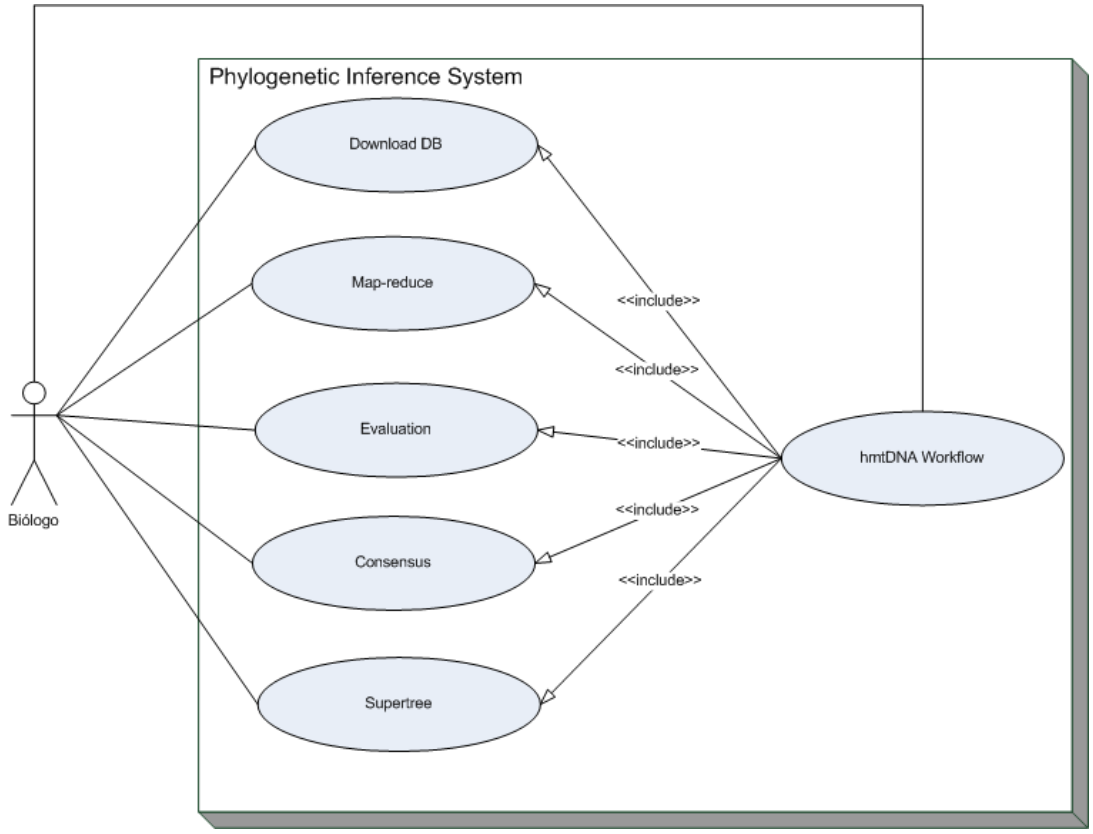


Figura 4. Diagrama de Casos de uso nivel 0

Este modelo muestra cinco casos de uso independientes entre sí que se corresponden a las principales funcionalidades del sistema *PhyloFlow* [2]. El caso de uso *Download DB* permite al actor principal ofrecer descargar secuencias de ADN mitocondrial humano de la base de datos de GenBank o recuperar bases de datos de secuencias de ADN disponibles. El caso de uso denominado *Map-reduce* permite al actor principal realizar un procesamiento biológico sobre secuencias de ADN.

El caso de uso denominado *Evaluation* permite al actor principal realizar una evaluación de modelos evolutivos sobre secuencias de ADN y construcción de soluciones parciales. El caso de uso denominado *Consensus* permite al actor principal realizar una fase de consenso e integración de soluciones parciales.

El caso de uso denominado *Supertree* permite al actor principal la construcción de un superárbol filogenético a partir de resultados parciales.

El caso de uso denominado *hmtDNAWorfklow* es el encargado de realizar un flujo de trabajo completo para datos de secuencia de ADN mitocondrial humano, por tanto, ha de incluir al resto de casos de uso para llevar a cabo el análisis pertinente. El actor principal del sistema puede interactuar con los seis casos de

usos, debido a que puede realizar un flujo de trabajo completo o realizar fases individuales e independientes. La sección número 3 de **Anexo I. Análisis** (pág. 38) contiene la descripción formal de los casos de uso.

Casos de uso: nivel 1

Los casos de uso de nivel uno sirven para mostrar las funcionalidades básicas del sistema y los funcionalidades subyacentes que emplean. La Figura 5 muestra el modelado en casos de uso de nivel 1 del sistema para los biólogos. La sección número 2 de **Anexo I. Análisis** (pág. 38) contiene los casos de uso centrados en el usuario administrador.

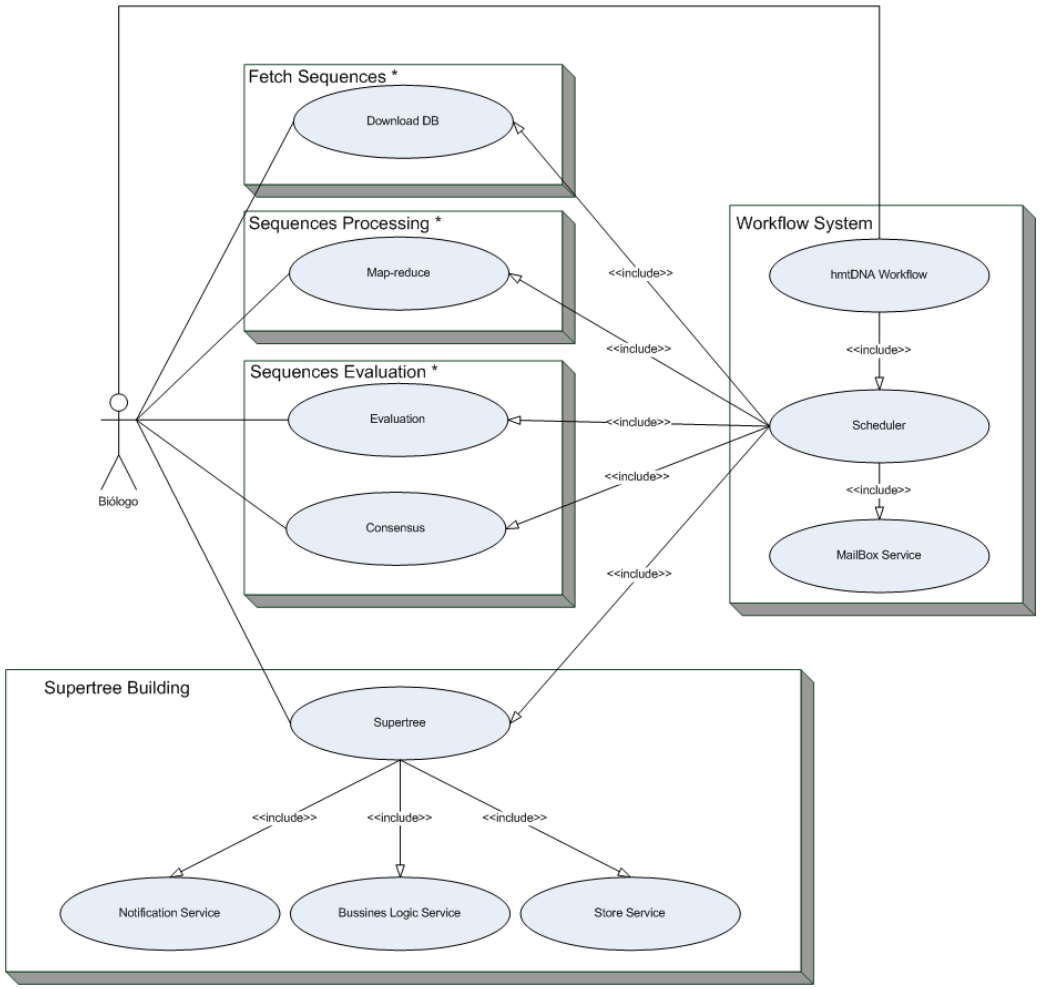


Figura 5. Diagrama de Casos de uso nivel 1 para el usuario biólogo

* Por simplificar el diagrama se han omitido el resto de casos de uso de los subsistemas *Fetch Sequences, Sequences Processing y Sequecences Evaluation*. Éstos y sus relaciones son idénticos a las mostradas en el sistema que denominado *Supertree Building*.

A diferencia del diagrama de casos de uso de nivel cero, el diagrama de casos de uso de nivel uno está formado por cinco subsistemas. El subsistema denominado *Workflow System* es el encargado de establecer el flujo de trabajo integrando el resto de subsistemas, los cuáles equivalen a cada uno de los componentes del sistema *PhyloFlow* [2]. El actor principal puede interactuar con cualquiera de ellos, por tanto, no sólo puede obtener el resultado final de un flujo de trabajo completo, sino que también puede obtener soluciones parciales interactuando con cada componente de forma individual.

Cada uno de los componentes está formado por uno o más casos de uso que representan la funcionalidad que desempeña dentro del flujo de trabajo, y tres casos de uso más que representan funcionalidades adicionales. El caso de uso denominado *Business Logic Service* representa el servicio que provee la

funcionalidad básica de la lógica de negocio. El caso de uso *Store Service* representa el servicio que provee funcionalidades relacionadas con el almacenamiento y descarga de soluciones parciales y finales. Por último, el caso de uso nombrado como *Notification Service* sería el encargado de proveer la funcionalidad para notificar al actor los resultados obtenidos.

En el subsistema *Workflow System*, el caso de uso principal es el denominado *hmtDNAWorkflow*. Este caso de uso emplea el denominado como *Scheduler*. Este último representa la funcionalidad encargada de planificar los trabajos del flujo y solicitarlos a cada componente concreto. Por tanto, incluye los casos de uso que representas las funcionalidades básicas del resto de subsistemas. Por último, el caso de uso *MailBox Service* representa la funcionalidad asociada a la recepción de los resultados notificados por el resto de subsistemas.

Cada uno de los subsistemas se ofrece como un servicio independiente bajo la visión *SaaS*. Por tanto, cada subsistema ofrece sus funcionalidades como un servicio a cada uno de los actores del diagrama. Esto permite que los subsistemas se encuentren totalmente distribuidos reduciendo el acoplamiento entre los mismos. El actor accederá a cada unos de los servicios ofrecidos a través de Internet. La sección número 3 de **Anexo I. Análisis** (pág. 38) contiene la descripción formal de los casos de uso.

4.2.5. Propuesta de solución

La solución propuesta consiste en la división del sistema *PhyloFlow* [2] en cinco subsistemas independientes y desacoplados. Cuatro de ellos representan los componentes individuales del flujo de trabajo. El último es el encargado de realizar el flujo de trabajo completo integrando los anteriores ordenadamente.

Cada una de las funcionalidades de los subsistemas será ofrecida como un servicio bajo el punto de vista *SaaS*. En concreto, cada funcionalidad será ofrecida como un servicio Web accesible a través de internet. Esta solución permite realizar un flujo completo de inferencia filogenética o realizar etapas individuales del proceso que combinadas puedan dar lugar a nuevos flujos de trabajo. Como cada una de las funcionalidades se ofrece como un servicio no sólo permite acceder al usuario final a ellas sino que cualquier otro sistema externo puedan integrar dichas funcionalidades en su dominio de aplicación. La Figura 6 muestra gráficamente la solución propuesta.

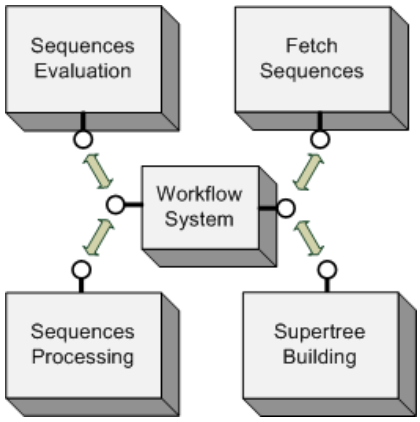


Figura 6. Distribución de los subsistemas que contienen los casos de uso

Debido a que los componentes se encuentran totalmente distribuidos se puede permitir el despliegue del sistema en un entorno distribuido. Esta solución permite que el sistema sea escalable y robusto, ya que la caída de un servicio no afectaría al resto, a excepción del que integra todos para realizar el flujo de trabajo completo. Por tanto, todos los servicios ofrecen un punto de entrada al subsistema que representan.

5. Diseño de la solución

Este capítulo contiene la documentación generada durante la fase de diseño del sistema. Esta fase del trabajo se ha realizado a partir de los objetivos que debe cubrir el sistema definidos en la fase de análisis. El diseño se ha realizado bajo el paradigma de Software como Servicio con lo que el soporte lógico y los datos se han de alojar en el servidor.

El objetivo principal ha sido realizar un diseño flexible que abstrajese mediante interfaces la implementación concreta de cada uno de los servicios, y permitiendo que cada servicio interno pueda tener una o más implementaciones concretas del mismo. Por tanto, el mayor esfuerzo invertido durante el diseño del sistema se ha destinado a realizar una solución lo más abstracta posible que dotará al sistema de flexibilidad y facilitará la reutilización de componentes.

5.1. Paradigma SaaS

El paradigma *Software as a Service* (*SaaS*) propone un modelo en el que el software es ofertado como servicio al usuario final. El software está albergado en un servidor y los usuarios acceden al software empleando un navegador Web [15]. La Figura 7 muestra la arquitectura de un sistema bajo la visión *Saas*.

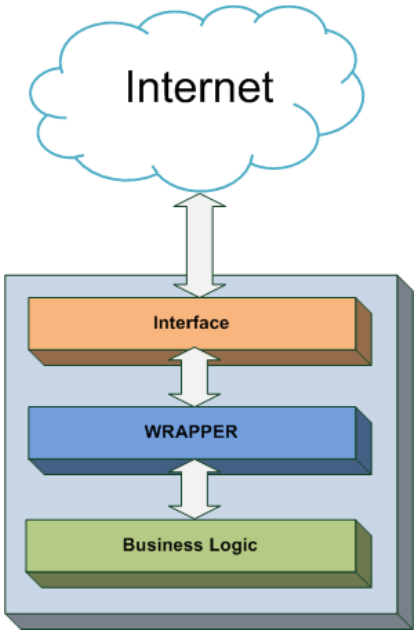


Figura 7. Vista de alto nivel de un componente del sistema bajo el paradigma SaaS

La figura anterior muestra las tres capas principales que componen el diseño una solución basada en *SaaS*. La capa denominada *Interface* es aquella capa del sistema que atiende las peticiones de los clientes que envían a través de la API ofrecida por la misma. La siguiente capa es la denominada *Wrapper*, debido a que encapsula el acceso a la lógica de negocio. Esta capa es la responsable de encapsular la comunicación entre las peticiones entrantes y las reglas de negocio del sistema. La tercera y última capa denominada *Business Logic* es la capa de negocio, es decir, aquella capa que contiene toda la lógica y reglas de negocio del sistema.

El Instituto Nacional de Estándares y Tecnologías (NIST) define la computación en la nube a través de tres modelo de servicios: Software como Servicio (*SaaS*), Plataforma como Servicio (*PaaS*) e Infraestructura como Servicio (*IaaS*) [16].

Con el modelo *SaaS*, los proveedores ofrecen aplicaciones que están desplegadas en una plataforma en la nube. El software se ejecuta en los servidores de la nube contratados por la empresa proveedora del servicio. Por tanto, si se detecta un error en el software solucionarlo en el servidor puede ser más sencillo y rápido, en lugar de distribuir y actualizarlo a todos los clientes [15]. Existen varios ejemplos de *SaaS*. Por ejemplo, *Google* ofrece varias aplicaciones Web, como *Gmail* y *Google Docs*. Ambas aplicaciones son ofrecidas como servicios online.

Los beneficios principales que aporta el modelo *SaaS* al cliente final son reducción de costes y tiempos. En el modelo *SaaS* al contrario que en modelos tradicionales, el cliente no necesita contratar hardware especifico donde albergar el software contratado ya que este reside en los servidores de la empresa proveedora, por tanto, permite desplegar las aplicaciones de forma rápida y sencilla sin la necesidad de instalar ni configurar el software contratado [17]. Además el mantenimiento como las actualizaciones del software será gestionado por parte del proveedor de servicios. En general, el software desarrollado para ser implementado como un servicio es más eficiente y provee de mejor funcionalidad y flexibilidad [16].

El modelo SaaS especifica aplicaciones que normalmente son diseñadas para operar en centros de datos distribuidos. Tienen la ventaja de escalar tanto el servidor como el ancho de banda para proveer acceso a más usuarios, aumentar el rendimiento y el tratamiento de información. No existe la necesidad de rediseñar el software o de desplegarlo de nuevo. Esto significa que se tiene el mismo servicio independientemente del número de usuarios y no se debe añadir más hardware o sistemas, ya que el proveedor de la nube proporcionará dinámicamente la escala que necesites [16].

Por tanto, los principales beneficios que aporta *SaaS* respecto a otros modelos son seguridad, escalabilidad, flexibilidad, disponibilidad y reducción de costes económicos y temporales. En el dominio de la aplicación se han tenido en cuenta los aspectos relacionados con la escalabilidad y flexibilidad para adaptarse al número de usuarios, aumentando o disminuyendo el número de recursos para ofrecer el servicio. También esta solución reduce costes temporales al usuario final, ya que no debe instalar ni configurar el sistema. El sistema ya se encuentra desplegado y accesible en una plataforma de computación en la nube.

Los centros de datos *SaaS* hoy en día garantizan que son altamente disponibles, esto implica generalmente que el proveedor tenga una cierta tolerancia a fallos o recuperación de errores para garantizar la disponibilidad en caso de producirse cualquier desastre. La elección de *Amazon EC2* nos asegura la disponibilidad del servicio aproximadamente el 99,9999% del tiempo.

5.2. Diseño de cada componente que constituye el flujo de trabajo

En primer lugar se ha llevado a cabo un diseño de alto nivel de cómo debía ser la estructura del sistema. Se ha tomado la decisión de realizar un diseño por capas, de tal forma que las capas del sistema se encargan de regir el comportamiento y dependencias del mismo. El sistema está formado por tres capas fundamentales, y cada una de ellas depende únicamente de la capa inmediatamente inferior. Mediante la arquitectura por capas se obtiene un diseño más flexible y reusable, ya que los cambios internos de una capa sólo afectan a la propia capa, y el número de dependencias se disminuye.

5.2.1. Presentación primaria

La presentación primaria de los componentes que constituyen el flujo de trabajo se corresponde a la Figura 7 mostrada anteriormente que se corresponde al diseño de arquitectura de un sistema basado en *SaaS*.

El diseño de la solución está formado por tres capas independientes. La capa denominada *Interface* es aquella capa de cada uno de los componentes encargada de atender las peticiones entrantes a través de la *API* ofrecida.

La siguiente capa es la denominada *Wrapper*, debido a que encapsula el acceso a la lógica de negocio. Esta capa es la responsable de encapsular la comunicación entre las peticiones entrantes y las reglas de negocio del sistema. En el contexto de implementación, será la capa que encapsule toda la funcionalidad asociada al lanzamiento de los scripts de cada una de los componentes del sistema inferencia filogenética y se encargue del tratamiento de las salidas generadas, así como de la monitorización necesaria.

La tercera y última capa denominada *Business Logic* es la capa de negocio, es decir, aquella capa que contiene toda la lógica y reglas de negocio del sistema. En este dominio de aplicación esta capa se corresponde con el sistema *PhyloFlow* [2], debido a que él posee todas las funcionalidades necesarias para la realización del análisis de filogenias de ADN mitocondrial humano.

Si se aumenta el nivel de detalle, se puede observar la aparición de una capa *middleware* compuesta por varios elementos que ofrecen distintos servicios a través de sus interfaces. Esta capa añade un nivel más de indirección entre la capa *Wrapper* y la capa que contiene la lógica de negocio.

La capa *Wrapper* se comunica con cada uno de ellos dotando al sistema de una funcionalidad completa en base a las funcionalidades ofrecidas por cada uno de los servicios. Por tanto, será el elemento representado como *Business Logic Service* el encargado de encapsular todas las llamas al sistema de inferencia filogenética. La Figura 8 muestra el diseño del sistema a un nivel de abstracción inferior.

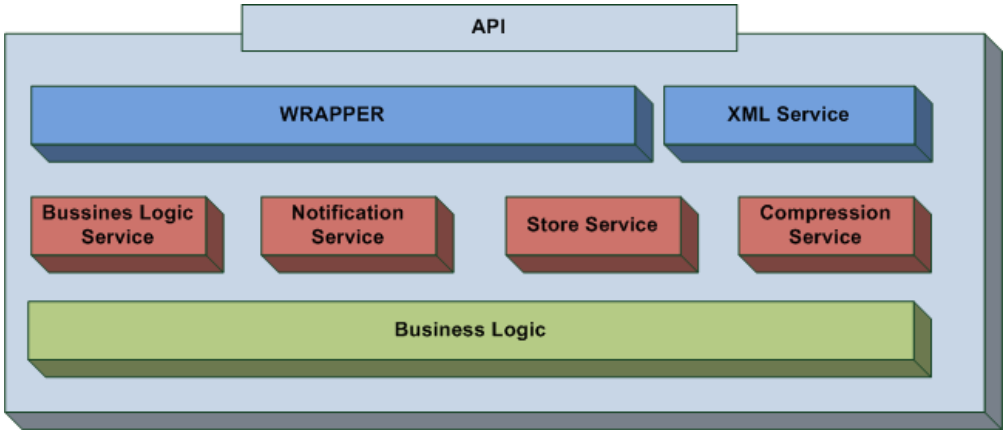


Figura 8. Vista con las capas de diseño de un componente genérico del flujo de trabajo

5.2.2. Relaciones y sus propiedades

La capa *Interface* se encarga de ofrecer los métodos del servicio que se pueden invocar remotamente. Para ello se define y publica una *API* que indica las reglas para la invocación del servicio. Cada vez que recibe una petición, la atiende y procesa. Para ello emplea el servicio denominado como *XMLService* que se encarga de validar la petición, si es válida la capa invocará a la capa inferior. Dicha *API* está explicada y detallada en la sección número 1 de **Anexo II. Diseño de la solución** (pág. 53).

La capa *Wrapper* o envoltorio es la que se encarga de encapsular el acceso a las reglas de negocio, y por tanto se delega sobre esta capa la comunicación con la capa de negocio y el acceso al resto de servicios que

dotan al sistema de la funcionalidad completa localizados en la capa middleware. La comunicación entre las capas *Interface* y *Wrapper* se rige en base a un lenguaje común previamente definido, y contendrá toda la información necesaria para realizar cualquier operación ofrecida por la lógica de negocio.

Los servicios adicionales se encuentran en una capa adicional o *middleware* ubicada en un nivel inferior respecto a la capa *Wrapper*. Esta capa de middleware contiene entre otros los servicios para acceder a la capa de negocio, el almacenamiento de resultados y la notificación de los resultados al cliente.

Un sistema diseñado por capas solo permite que existan dependencias entre los componentes de una misma capa y entre capas adyacentes. Sin embargo, todos los componentes de la capa middleware diseñada son independientes entre sí aumentado la flexibilidad del sistema en base a reducir dependencias.

La Figura 9 muestra las relaciones entre los elementos de cada una de las capas descritas anteriormente.

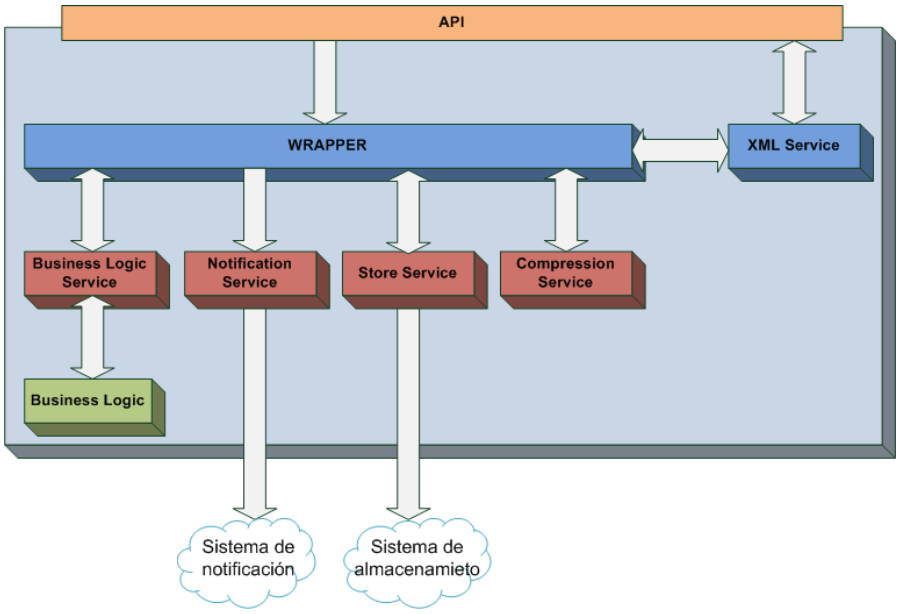


Figura 9. Vista de componentes y conectores de un componente genérico del flujo de trabajo

La capa middleware intermedia está compuesta por cuatro servicios:

- ***Business Logic Service***: este servicio ofrece todas las operaciones relacionadas con la lógica de negocio interna.
- ***Store Service***: este servicio ofrece operaciones relacionadas con la utilización de un sistema de almacenamiento externo.
- ***Notification Service***: este servicio ofrece operaciones relacionadas con la notificación de información a usuarios.
- ***Compression Service***: este servicio ofrece operaciones relacionadas con la compresión y descompresión de recursos.

El elemento *Business Logic* está formado por todos los scripts y librerías que componen cada una de los componentes del sistema de inferencia filogenética. Se encarga de realizar la descarga de secuencias de la base de datos *GenBank*, el procesamiento de las secuencias, generación de superárboles, etc. Todas las salidas se almacenan comprimidas en su correspondiente directorio. La interfaz completa está descrita en la sección número 2 de **Anexo II. Diseño de la solución** (pág. 61).

5.3. Diseño del componente que representa el flujo de trabajo

Una vez realizado el diseño de cada uno de los sistemas que representan cada una de los cuatro componentes que constituyen el sistema *PhyloFlow* [2] se ha procedido a diseñar el sistema que debía contener la lógica del funcionamiento del flujo de trabajo. El diseño del sistema de alto nivel es idéntico al del sistema para cada componente del flujo de trabajo y, por tanto, también está dividió en tres capas que dependen solamente de la capa inmediatamente inferior. La capa intermedia denominada anteriormente como *Wrapper* en este contexto es denominada como *Scheduler*, ya que informa mejor de la funcionalidad ofrecida por dicha capa.

5.3.1. Presentación primaria

La Figura 7 mostrada anteriormente revela el diseño del sistema que representa el flujo de trabajo. El nivel de interfaz se encargaría de publicar como servicio el flujo completo de un análisis filogenético.

La capa intermedia la representaría el *Scheduler* o planificador de tareas que sería aquel componente encargado de realizar las peticiones correspondientes a cada uno de los componentes del flujo de trabajo en base a la información proporcionada por la lógica de negocio.

En este caso, la capa *Workflow* representa las reglas de negocio, y por tanto, contiene la secuencia y orden de cada componente así como la operación y los parámetros que se ha de realizar.

Si aumentamos el nivel de detalle del diseño del sistema en este caso no aparece una capa de middleware que ofrezca unos servicios complementarios. Sin embargo, se puede observar cómo dentro de la capa intermedia aparece un componente denominado *MailBox*, el cual se encargará de recuperar las respuestas recibidas por parte de los componentes como respuesta a las peticiones realizadas por el componente *Scheduler*. La Figura 10 muestra el diseño del sistema a un nivel de abstracción inferior.

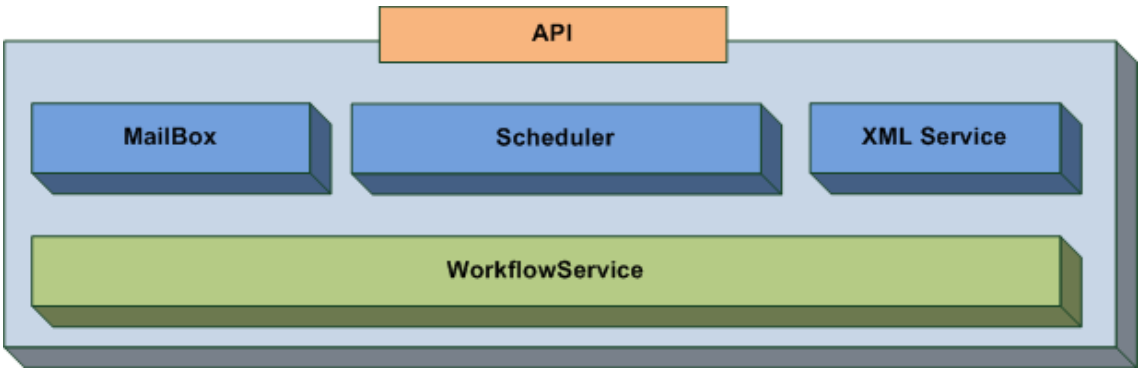


Figura 10. Vista con las capas de diseño del componente flujo de trabajo

5.3.2. Relaciones y sus propiedades

La capa Interface se encarga de ofrecer los métodos del servicio que se pueden invocar remotamente. Para ello se define y publica una *API* que indica las reglas para la invocación del servicio. Cada vez que recibe una petición, la atiende y procesa. Dicha *API* está explicada y detallada en la sección número 3 de **Anexo II. Diseño de la solución** (pág. 64).

La capa intermedia es la encargada de planificar las tareas del flujo de trabajo. Para ello, se comunica con la capa inferior para obtener la siguiente tarea a realizar dentro del flujo de trabajo actual y realizar la petición al componente concreto que debe realizarla. Para enviar las peticiones de forma correcta emplea el componente *XMLService* que se encarga de generar los mensajes en formato *XML* válidos en base al lenguaje de comunicación definido.

Por último, realiza la petición con el mensaje generado al componente concreto para que se encargue de procesarlo y realizar la tarea deseada. El componente *MailBox* será el encargado de comprobar si se ha recibido una nueva notificación que contenga los resultados solicitados. Si es así, el componente *Scheduler* recuperará los resultados y comenzará a preparar la siguiente tarea a realizar del flujo de trabajo.

La lógica de negocio está representada por el componente *Workflow Service* que ofrece una serie de servicios para interactuar con el flujo de trabajo que permite a la capa superior obtener cuál es el la siguiente tarea a realizar dentro del flujo de trabajo actual. La Figura 11 muestra las relaciones dentro del sistema.

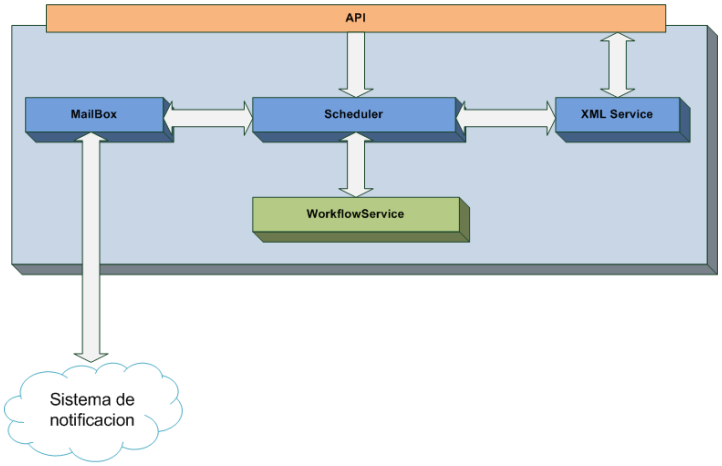


Figura 11. Vista de componentes y conectores del componente flujo de trabajo

5.4. Arquitectura global del sistema

Se ha descrito el diseño y arquitectura de cada uno de los componentes por separado. A continuación se muestra la arquitectura global del sistema combinada con la vista de despliegue. Cada componente está albergado en una máquina independiente formando una sistema totalmente distribuido. El medio de comunicación entre cada uno de los componentes es el denominado como *Message Broker*. La Figura 12 muestra el despliegue y arquitectura global del sistema.

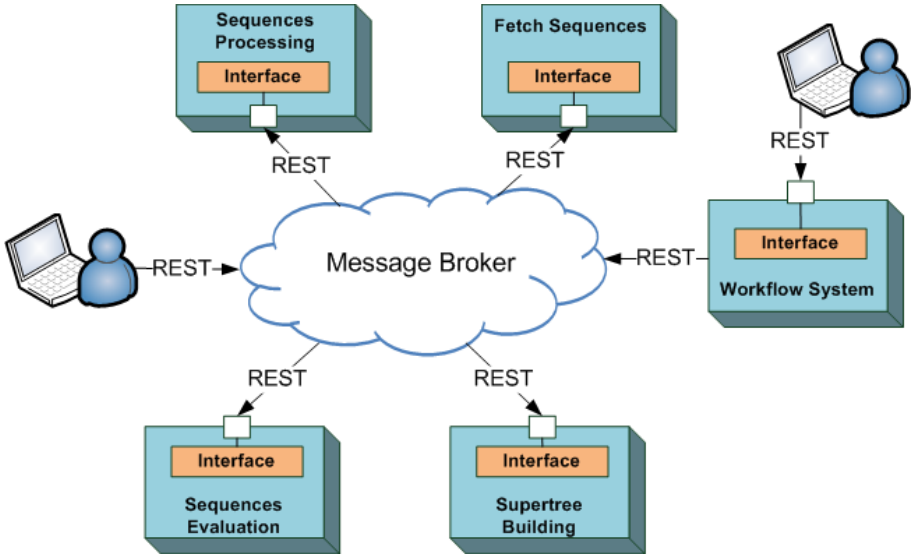


Figura 12. Vista de despliegue del sistema

5.5. Interfaz pública de cada servicio

La interfaz pública de cada servicio debe definir unas reglas de entrada y unas reglas de salida que permita a cualquier persona o sistema externo interactuar con él. Se ha definido un lenguaje común para describir tanto el estándar de entrada como de salida. De este modo, todos los componentes emplean el mismo estándar para validar las peticiones de entrada el cual está definido y descrito empleando un esquema en lenguaje XML. Tanto el lenguaje común como el esquema se muestran y explican en detalle en la sección número 4 de **Anexo II. Diseño de la solución** (pág. 66).

Esto permite que cualquier sistema externo pueda utilizar los servicios ofrecidos individualmente por cada uno de los componentes para integrarlos en el contexto en el que interopere, así como la integración completa empleando los resultados proporcionados por el flujo de trabajo completo.

6. Implementación

Una vez finalizadas las fases anteriores se ha procedido a realizar la implementación del sistema. Se han realizado sucesivas refactorizaciones del código generado durante la fase de implementación. Para la gestión y control de versiones se ha empleado un repositorio privado de tipo *Subversion*.

Como se ha mencionado anteriormente, el sistema *PhyloFlow* [2] está formado por cuatro componentes que se han implementado de forma independiente e individual. Además se ha desarrollado otro sistema externo que contiene la lógica del flujo de trabajo. Todos los componentes ofrecen sus funcionalidades a través de una interfaz pública, es decir, cada componente ofrece los servicios relacionados con su funcionalidad específica y el último ofrece como servicio la ejecución de un flujo de trabajo completo integrando los anteriores.

6.1. Software y tecnologías empleadas

Se han empleado las últimas versiones del sistema de análisis de filogenias denominado *PhyloFlow* [2] proporcionado por el investigador Jorge Álvarez, así como las librerías necesarias por dicho sistema. Este sistema está programado íntegramente con el lenguaje de programación *Python* [9] y emplea actualmente la versión 2.7 de este. Además el sistema emplea dos librerías externas: la versión 1.63 de *Biopython* [18] y la versión 3.8.1 de *DendroPy* [19].

DendroPy [19] es una librería de *Python* [9] para la computación filogenética. Provee clases y funciones para la simulación, procesamiento y manipulación de árboles filogenéticos y matrices de caracteres. *Biopython* [18] es un conjunto libre de herramientas para la computación biomédica desarrollado en *Python* [9] por un equipo internacional de desarrolladores.

El sistema implementado está programando íntegramente en *Java* 1.7 [7] y se ha empleado la herramienta *Maven* [20] para facilitar la gestión, configuración y creación del proyecto. La herramienta *Maven* [20] permite describir las dependencias del proyecto y automatizando la descarga de los módulos.

Para el desarrollo de los servicios Web se ha empleado *JAX-RS*. La tecnología *JAX-RS* o *Java API for RESTful Web Services* es una API del lenguaje de programación *Java* [7] que proporciona soporte en la creación de servicios Web de acuerdo con el estilo arquitectural *Representational State Transfer* (*REST*). *JAX-RS* emplea anotaciones para simplificar el desarrollo y despliegue de los servicios Web. Desde *Java EE 6* es una característica oficial l del lenguaje, por tanto, no es necesaria ninguna configuración para comenzar a usarla.

Para poder trabajar de forma más cómoda se ha utilizado uno de los laboratorios de investigación del Grupo de Ingeniería de Sistemas de Eventos Discretos (GISED), el L1.03B, en el que he dispuesto de un espacio físico y de una máquina (astazu) durante el periodo comprendido desde el inicio a la finalización del trabajo. La utilización de dicha máquina ha sido de gran utilidad en las fases tempranas del proyecto, ya que no disponía de ninguna máquina con un sistema operativo de tipo UNIX.

Para el desarrollo y despliegue del sistema en una plataforma en la nube se han empleado varios de los servicios ofertados por la plataforma ofrecida por *Amazon Web Services* (*AWS*) de manera gratuita durante un año con ciertas restricciones de prestaciones y recursos. En concreto el proyecto emplea el servicio de almacenamiento *S3* y el servicio *EC2* de computación elástica en la nube.

Se optó por emplear una imagen de una máquina *Ubuntu 14.04* proporcionada por defecto por el servicio de *Amazon* sobre la cual instalar y configurar las tecnologías Web necesarias, el sistema previo y las librerías de las que dependía. Una vez configurada se creó una instantánea de la misma, que permite replicarla de forma transparente al usuario y sin la necesidad de realizar de nuevo todo el proceso de instalación y configuración.

6.2. Implementación de cada componente del flujo de trabajo

La implementación completa de cada componente del flujo de trabajo se ha dividido en servicios. Una vez implementado cada componente o servicio se ha realizado la integración de los mismos. Primero se han implementado las capas inferiores del diseño y por último las capas superiores. La Figura 13 muestra la vista de componentes del sistema implementado.

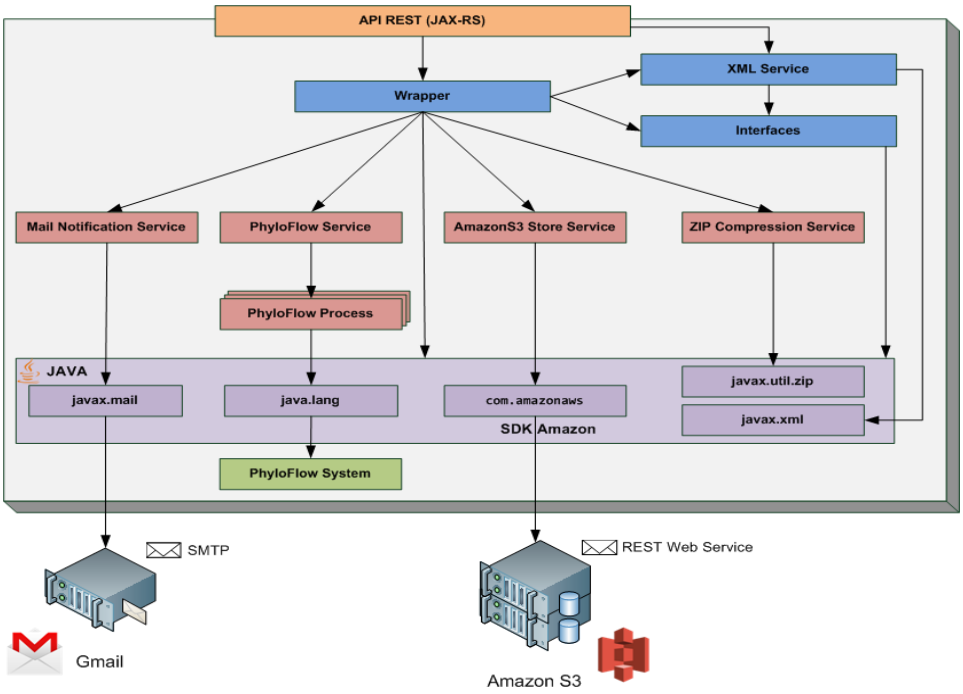


Figura 13. Vista de implementación de un componente genérico del flujo de trabajo.

6.2.1. Implementación de la capa de negocio

En primer lugar, se ha desarrollando un prototipo capaz de lanzar procesos encargados de la ejecución de los distintos scripts en *Python* [9] que representan cada uno de los componentes del sistema *PhyloFlow* [2] desde la máquina virtual de *Java* [7]. A partir de este prototipo y tras sucesivas refactorizaciones se ha obtenido el servicio que se encarga de interoperar de forma transparente con el sistema *PhyloFlow* [2].

Este servicio se encarga de generar comandos cuya ejecución en el sistema *PhyloFlow* [2] equivale a la funcionalidad solicitada y permite acceder a todas las funcionalidades ofrecidas por dicho sistema. De este modo, este servicio es reutilizado por los cuatro sistemas que representan cada uno de los componentes del flujo de trabajo.

Este servicio sigue el patrón *Façade* abstrayendo las interacciones necesarias con el sistema de análisis de filogenias a través de una interfaz simple y reveladora con los servicios de la capa de negocio. El servicio cuenta con una plantilla de comandos, cada uno de ellos representa una funcionalidad concreta del sistema *PhyloFlow* [2]. El servicio se ha de encargar de rellenar el comando para cada funcionalidad solicitada con los parámetros de entrada, creando una lista con todos los comandos que se deben ejecutar para

completar la ejecución. Una vez que se dispone de dichos comandos el sistema lanza un proceso por cada uno de ellos empleando el *shell* de la máquina.

Los procesos o trabajos se lanzan en paralelo hasta alcanzar un número máximo de procesos en ejecución. Se han tenido en cuenta todos los problemas derivados de la concurrencia de procesos y accesos en exclusión mutua para evitar que se generen errores y excepciones. Con la paralelización de los trabajos se obtienen mejores resultados. Se ha establecido un número máximo de procesos en ejecución para evitar sobrecargar la carga de trabajo de la CPU. Este parámetro es configurable, y por tanto, se puede modificar para adaptarse a la potencia y prestaciones ofrecidas por la máquina o instancia en la que se ejecute el sistema.

6.2.2. Implementación sistema de almacenamiento externo

A continuación, se ha desarrollado un sistema para el almacenamiento de resultados parciales y finales que permita acceder a ellos mediante una URI, en este caso una URL. Este servicio debe permitir además del almacenamiento de los resultados la descarga de los mismos, ya que dichos resultados forman parte de la entrada del próximo componente del flujo de trabajo.

Para el almacenamiento de estos resultados se optó por emplear un servicio de almacenamiento externo que ofreciera las funcionalidades necesarias a través de un servicio Web, para facilitar su integración de forma automática en el sistema. En concreto se ha decidido emplear un sistema de almacenamiento en la nube que proporciona una alta disponibilidad y fiabilidad de los resultados almacenados, además permite el acceso a los mismos mediante un identificador de recursos universal (URI) a través de cualquier navegador.

Se ha empleado el servicio en la nube denominado *S3* (*Simple Storage System*) que forma parte del conjunto de servicios en la nube ofrecidos por *Amazon Web Services*. La implementación del servicio de almacenamiento ha consistido en la utilización del *SDK* (*Software Development Kit*) ofrecido por *Amazon S3* que se encarga de realizar las peticiones al servicio de forma transparente. Sin embargo, se ha definido una interfaz común que posibilita emplear cualquier otro sistema de almacenamiento externo que implemente dicha interfaz de forma transparente al usuario.

También se ha añadido una funcionalidad que permite la descarga de los archivos que corresponde a la entrada de un componente debido a que algunos recursos puede que no estén ubicados en *Amazon S3*. Esta solución se ha considerado necesaria, ya que cada resultado se almacena en *Amazon S3* durante un cierto periodo de tiempo, pasado ese periodo de tiempo se elimina del sistema de almacenamiento para evitar almacenar una gran carga de datos. Los clientes puede descargar dichos resultados y dejarlos accesibles remotamente para que el sistema pueda acceder a ellos independientemente de la ubicación de los mismos.

Debido a que *Amazon S3* no permite almacenar estructuras con un formato de sistemas de ficheros a través de sus servicios Web se tomó la decisión de almacenar los resultados en único fichero comprimido.

6.2.3. Implementación de los sistemas de compresión y notificación

Se ha implementado un servicio compresor que emplea el formato *zip* [21] para comprimir y descomprimir los resultados intermedios. Este servicio implementa una interfaz definida previamente, de tal forma, que se puede extender empleando otros formatos de compresión. El sistema *PhyloFlow* [2] emplea tanto en la entrada de datos como en la salida ficheros comprimidos en formato *gzip* [22], por tanto, se ha tomado la decisión de que si el fichero pasado como entrada tiene este formato se considera una entrada válida para *PhyloFlow* [2] y, por tanto, no se realiza la descompresión previa. Si el formato del fichero de entrada no es ninguno de los anteriores se descarta la petición.

Seguidamente se ha realizado el servicio de notificación. Como el protocolo de comunicación debía ser asíncrono, se ha empleado el protocolo *SMTP* [23] para notificar los resultados. Se ha utilizado el lenguaje común definido previamente que contempla la posibilidad de que se haya producido un error, permitiendo notificar al usuario del error que se ha producido y su causa.

6.2.4. Implementación del encapsulador e interfaz

Una vez implementados estos cuatros servicios, se ha desarrollado el encapsulador que se encarga de invocar cada uno de ellos en el orden correcto para llevar a cabo el proceso completo definido. Primero se debe de encargar de descargar los datos de entrada, si es necesario, seguidamente los ha de descomprimir, invocar al sistema *PhyloFlow* [2], comprimir y almacenar los resultados. Por último, debe notificar dichos resultados al usuario.

Para acceder al sistema se ha implementado la capa Interfaz. Debido a que los análisis de genoma humano, en este caso, del ADN mitocondrial son problemas intensivos en calculo, todos los servicios debían ofrecer un protocolo de comunicación asíncrono capaz de notificar los resultados una vez finalizado el proceso solicitado. Esto obligo a realizar servicios asíncronos que se encargarían de lanzar procesos independientes encargados de generar, almacenar y notificar los resultados a los clientes. Además se implementó un servicio que debía validar y analizar la entrada en base al lenguaje común definido mediante un esquema en lenguaje XML y generar mediante dicho lenguaje un fichero XML que contenga la respuesta.

6.2.5. Aspectos interesantes de la implementación

Para aumentar la flexibilidad del sistema y facilitar la configuración del mismo se han generado distintos ficheros de propiedades que contienen propiedades del sistema así como de sus componentes. Estos ficheros almacenan las credenciales de acceso a servicios externos como *Amazon S3* o las cuentas de correo electrónico de *Gmail*, así como ciertas propiedades de la máquina, sistema operativo, variables de entorno, etc. Además se definió un espacio de trabajo que debía contener entre otros datos, los ficheros de log, las peticiones de entrada recibidas y las respuestas, así como el sistema previo.

También se ha definido un mapa de errores que contiene todas las posibles excepciones que se pueden generar durante la ejecución de un componente. El mapa de errores se encuentra descrito en la sección número 1 de **Anexo III. Implementación** (pág. 69).

6.3. Implementación del sistema flujo de trabajo

Una vez finalizado los cuatro sistemas que representan cada uno de los componentes constituyentes del flujo de trabajo se comenzó a realzar el sistema que debía planificar los trabajos del mismo para realizar un flujo de trabajo completo. La Figura 14 muestra la vista de componentes del sistema implementado:

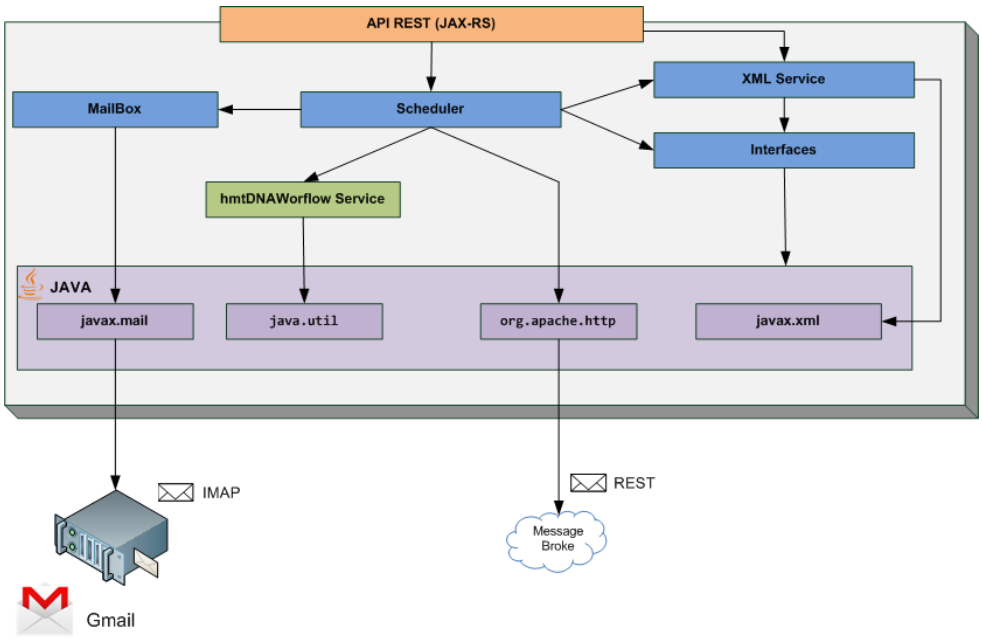


Figura 14. Vista de implementación del componente flujo de trabajo

Este servicio debía utilizar un protocolo de comunicación asíncrono debido a que el tiempo completo de procesamiento de un flujo de trabajo completo es muy elevado. Se tomó la decisión de que este sistema no empleará el protocolo *SMTP* [23], que permite el envió de correos electrónicos, a cambio de emplear el protocolo *IMAP* [24], que permite la lectura de la bandeja de entrada. Este protocolo permite leer e interpretar los mensajes de respuesta recibidos por parte de todos los componentes que integran el flujo de trabajo y delega en el último componente del flujo de trabajo la tarea de enviar el mensaje con los resultados al cliente final.

Para obtener un servicio asíncrono se genera un proceso independiente por cada petición recibida que es el encargado de gestionar las peticiones a cada uno de los componentes que integran el workflow y de recoger y analizar los resultados recibidos.

Para la implementación del componente *MailBox*, el cual es el encargado de acceder a la bandeja de entrada, leer los mensajes nuevos y notificar al resto de procesos cuando haya un mensaje nuevo, se empleo el patrón *Singleton*. Este patrón permite que haya una única instancia de la clase en ejecución, por tanto, de esta forma se evitan problemas de concurrencia debido al acceso múltiple a la bandeja de correo electrónicos.

7. Despliegue y evaluación

Una vez desarrollada la primera versión funcional del sistema se ha desplegado en un entorno distribuido. Para el despliegue del sistema en la nube se han realizado unas tareas previas. Estas tareas han consistido el lanzamiento de una instancia de una máquina *Ubuntu 14.04* a partir de una imagen proporcionada por *Amazon*. Para esta instancia se ha configurado el volumen persistente de la máquina, las reglas de seguridad para acceder a la máquina y el centro de datos donde estará disponible (Irlanda). Una vez que la instancia esta lanzada se ha accedido vía *ssh* mediante un fichero que contiene una clave privada, ya que al instanciar una máquina se añade la clave publica al fichero que almacena las claves que tienen autorización de acceso.

La configuración ha consistido en instalar todas las librerías de las que depende el sistema *PhyloFlow* [2] así como instalar un servidor *Tomcat 7.0* sobre el que desplegar el sistema. Además se han copiada el sistema utilizando el comando *scp* o la herramienta *Filezilla*, se han añadido las variables de entorno necesarias del sistema *PhyloFlow* [2].

Una vez que se ha configurado completamente el entorno se ha creado un script que se encarga de lanzar el servidor *Tomcat* en el arranque de la máquina, de tal forma, que siempre que la máquina se reinicia el servidor quedará accesible. Po último, se ha realizado una instantánea de la máquina configurada. A partir de esta instantánea se pueden crear nuevas instancias que se encuentran totalmente configuradas de forma automática.

7.1. Explicación de una traza completa de despliegue

Una vez que se encuentran las instancias que corresponden a cada uno de los componentes del sistema en ejecución y los servidores de aplicaciones se encuentran activos se puede realizar peticiones a los servicios Web. A continuación se explican las etapas principales de una traza completa de despliegue de una ejecución de un conjunto de secuencias de ADN mitocondrial.

En primer lugar se solicita al servicio Web del componente *Workflow System* la realización de una ejecución completa para el análisis del ADN mitocondrial humano. El componente *Workflow System* valida la petición recibida y en el caso de ser correcta comienza el proceso.

El planificador de tareas seleccionará el primer trabajo a realizar dentro del flujo de trabajo. A partir de dicho trabajo generará una cadena de caracteres en formato XML que cumpla el estándar de entrada definido con el método seleccionado y los parámetros de entrada necesarios. Enviará la petición al componente correspondiente y se mantendrá a la espera de la respuesta.

Cuando reciba el mensaje con la respuesta descarga el fichero que la contiene. Analizara la respuesta empleado el formato de salida definido para obtener los valores de la respuesta. Si no aparece ningún bloque *fault* indicando un error durante el procesamiento selecciona el siguiente trabajo planificado. Realiza estas etapas iterativamente hasta el último trabajo del flujo.

En la última etapa del flujo de trabajo, en lugar de emplear el correo electrónico propio introduce como parámetro el correo del cliente para que sea este el que reciba el mensaje con la solución final a su petición.

En el caso concreto del procesamiento del ADN mitocondrial humano el orden de las etapas es el siguiente:

1. Petición del cliente al componente *Workflow System*
2. Descarga de las secuencias de ADN mitocondrial humano
3. Procesamiento utilizando el método de haplogrupos
4. Procesamiento utilizando el método de genes
5. Evaluación de modelos evolutivos y generación de *bootstraps*
6. Consenso de las soluciones parciales antes de la fase de *bootstraps*
7. Consenso de las soluciones parciales tras la fase de *bootstraps*
8. Generación de un único superárbol

Las respuestas generadas por cada componente se notificarán al componente *Workflow System* a través de correos electrónicos. El componente *Workflow System* accederá a la cuenta de correo electrónico empleando el protocolo *IMAP* [24] y de esta forma recuperara las respuestas a sus peticiones. La Figura 15 muestra la traza completa de despliegue.

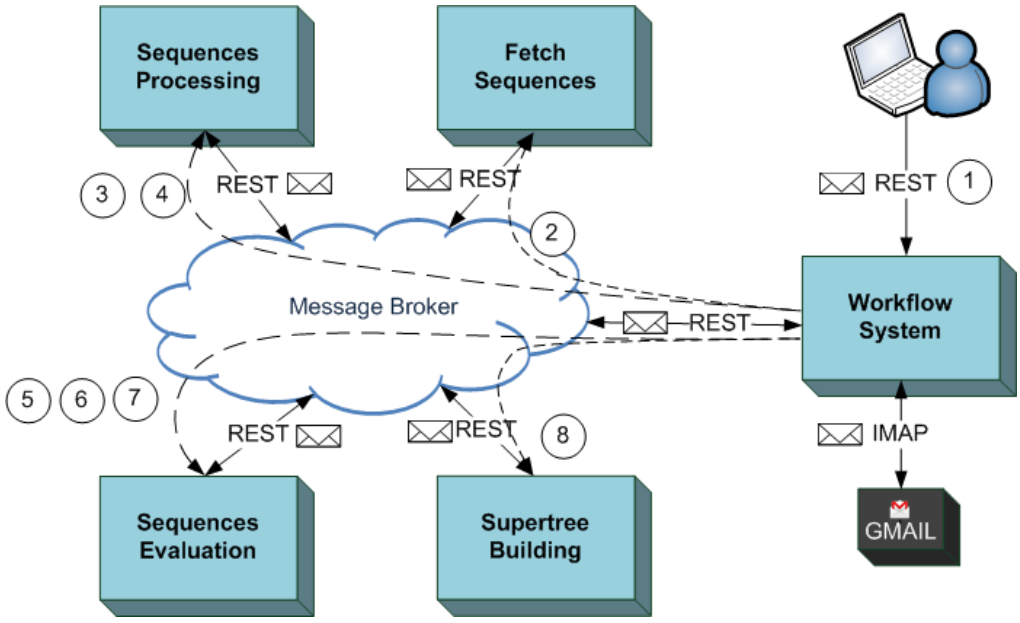


Figura 15. Traza de despliegue para un flujo completo de análisis filogenético de ADN mitocondrial

7.2. Verificación y validación del sistema

Para la validación y verificación del sistema se ha empleando la herramienta *JUnit 4* que ha permitido generar una batería de pruebas unitarias totalmente automáticas. Dichas pruebas se han realizado para validar el comportamiento del sistema ante fallos derivados de la entrada, la ejecución completa de un flujo de trabajo, el correcto funcionamiento del sistema de almacenamiento externo, etc.

Este tipo de pruebas son de gran utilidad para validar que la integración de nuevos componentes o los cambios en el código no han implicado modificaciones en el comportamiento del sistema. En caso de modificarse permite detectar el error en función de los test fallidos.

Además se han realizado pruebas del sistema en máquinas locales para verificar su correcto funcionamiento antes de desplegar el sistema en la nube proporcionada por *Amazon*. En ella también se han realizado pruebas para comprobar si permitía, entre otras coas, enviar correos electrónicos sin habilitar los puertos correspondientes en las reglas de seguridad.

Para llevar a cabo la evaluación del sistema se ha creado una batería de pruebas para la realización de flujos de trabajo completos con distintas métricas. Debido a las limitaciones computacionales de las instancias micro proporcionadas de forma gratuita por la plataforma *Amazon EC2,* en lugar de emplear secuencias de ADN mitocondrial se han empleado secuencias sintéticas o conjuntos de secuencias reducidas de las primeras.

Los objetivos principales eran validar el comportamiento del sistema y evaluar el sobrecoste que supone ofrecer como servicio las funcionalidades de cada componente individual del sistema *PhyloFlow* [2] así como el sobrecoste total de un flujo de trabajo completo.

7.3. Resultados de la evaluación

La evaluación del sistema se encuentra detallada en la sección **Anexo IV. Despliegue y evaluación** (pág. 71). Los resultados de la evaluación muestran que el sobrecoste que supone ofrecer cada componente como un servicio en el cómputo global de un flujo de trabajo es ínfimo tanto para secuencias de ADN mitocondrial como secuencias sintéticas. La Tabla 3 muestra el sobrecoste medio para cada uno de los componentes y el total en función del tipo de secuencia.

Overhead medio en porcentaje			
Sequence	*Fetch Sequences*	*Sequences Processing*	*Sequences Evaluation*
hmtDNA	94%	0%	0%
synthetic	70%	1%	2%

Tabla 3. Sobrecoste medio en función del tipo de secuencia

El componente *Supertree Building* no ha sido evaluado debido a que su implementación final no está finalizada. A pesar de que el sobrecoste sea elevado en el componente *Fetch Sequences* debido a que representa un parte ínfima del total del tiempo a penas afecta al sobrecoste total. Como el procesamiento de las secuencias sintéticas es menos costoso computacionalmente el sobrecoste es superior respecto a las secuencias de ADN mitocondrial, pero aún así es despreciable.

A continuación la Figura 16 y la Figura 17 muestran dos gráficas comparativas de la evolución del sobrecoste en función del número de conjuntos de secuencias a procesar. Estas gráficas permiten observar además la diferencia que existe en cada operación entre el tiempo total y el tiempo de ejecución del sistema *PhyloFlow* [2] y las operaciones que implican un mayor coste temporal dentro del flujo de trabajo.

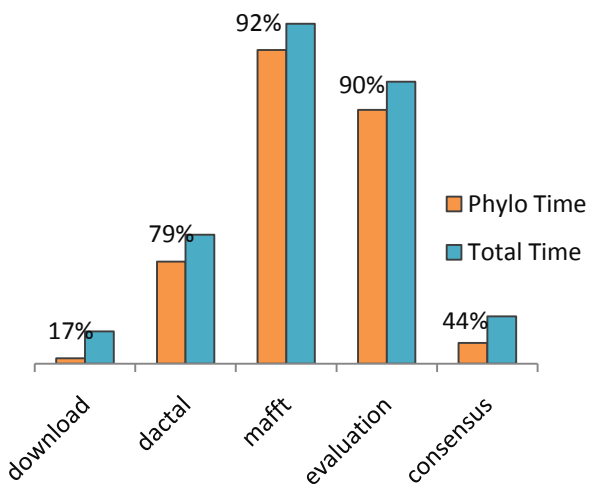


Figura 16. Gráfica con los tiempos totales para 2 sets

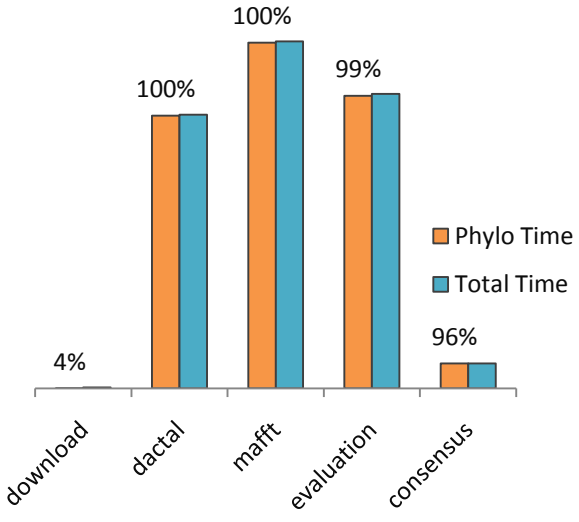


Figura 17. Gráfica con los tiempos totales para 16 sets

Cuando se aumenta el número de conjuntos de secuencias el porcentaje que supone el sobrecoste en el tiempo total se reduce. Esto se debe a que el coste temporal del procesamiento de secuencias tiene mayor complejidad, y por tanto, un coeficiente de crecimiento netamente superior.

El componente *Sequences Processing* realiza el procesamiento biológico con los métodos *dactal* y *mafft*. En este tipo de flujo de trabajo la fase de mayor coste temporal, por tanto, es la de procesamiento biológico. También se puede observar como la fase de recuperación o descarga de secuencias tiene el mayor sobrecoste de todos. Esto se debe a que para las secuencias de ADN sintético no se realiza ninguna descarga únicamente se recupera una base de datos disponible en el sistema de ficheros del sistema.

7.4. Sistema de toma de decisiones

A partir de la información recogida en los fichero de log en concreto los asociados a los servicios relacionados con el sistema *PhyloFlow* [2] se ha decidió crear perfiles de ejecución que permitan mejorar el aprovisionamiento de recursos. La información recogida de estos ficheros corresponde a tiempos de ejecución, consumo de memoria, método de procesamiento, número de reintentos y número total de sets y tipo de secuencias procesados.

Así se ha plantea un método de mejora sencillo basado en el entrenamiento de un árbol de decisión sobre un conjunto de entrenamientos basado en tablas por cada uno de los componentes que constituyen el flujo de trabajo del sistema *PhyloFlow* [2]. Por último, se ha escogido que tipo de instancia se adaptaría mejor para cada prueba realizada considerando los costes económicos y temporales que implicaría su elección de forma aproximada.

Este sistema ha permitido realizar una evaluación inicial del rendimiento del sistema estableciendo estrategias sobre el tipo de plataforma computacional en la cual desplegar cada componente del sistema. La implementación y conclusiones del sistema de toma de decisiones se encuentra detallada en la sección **Anexo IV. Despliegue y evaluación** (pág. 73).

8. Gestión del proyecto y conclusiones

Este capítulo contiene toda la información relacionada con la gestión del proyecto y los procesos de seguimiento y control del mismo. Por último se exponen las conclusiones técnicas y personales del trabajo realizado.

8.1. Gestión del proyecto

La metodología utilizada para llevar a cabo una correcta gestión del proyecto ha sido la un modelo de ciclo de desarrollo en cascada mejorado. Dicha metodología establece una organización en fases del proyecto y sus correspondientes hitos facilitando su seguimiento, identificación de carencias y retroalimentación del desarrollo.

Las fases principales del proyecto corresponden a la fase de captura de requisitos y análisis del sistema, diseño de la solución, implementación y despliegue. Al final de cada una de las fases se ha realizado una verificación de la misma proporcionando retroalimentación tanto para la fase actual como las fases previas.

Se ha seguido un flujo de trabajo basado en las fases principales que se consideran en Ingeniería del Software haciendo especial hincapié en aquellas relacionadas con la Ingeniería de Requisitos. Esto ha proporcionado como resultado un análisis profundo de las dependencias, características, necesidades y limitaciones de la solución. La Figura 18 muestra gráficamente el modelo de desarrollo en cascada mejorado.

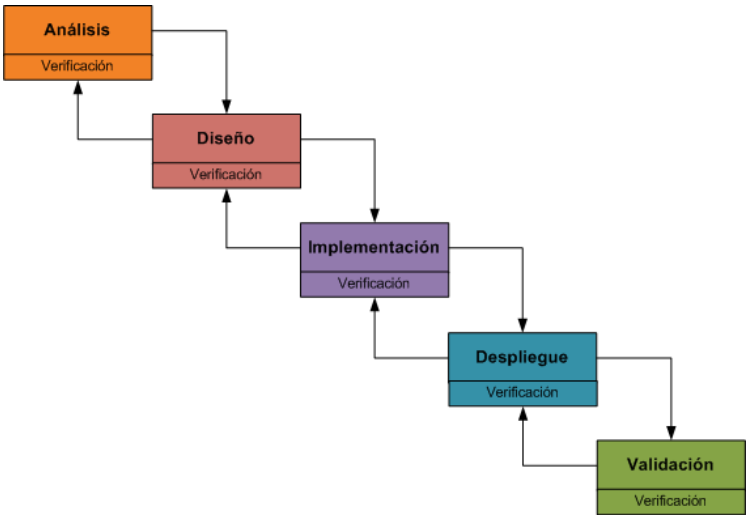


Figura 18. Modelo en cascada mejorado con retroalimentación en cada una de las fases

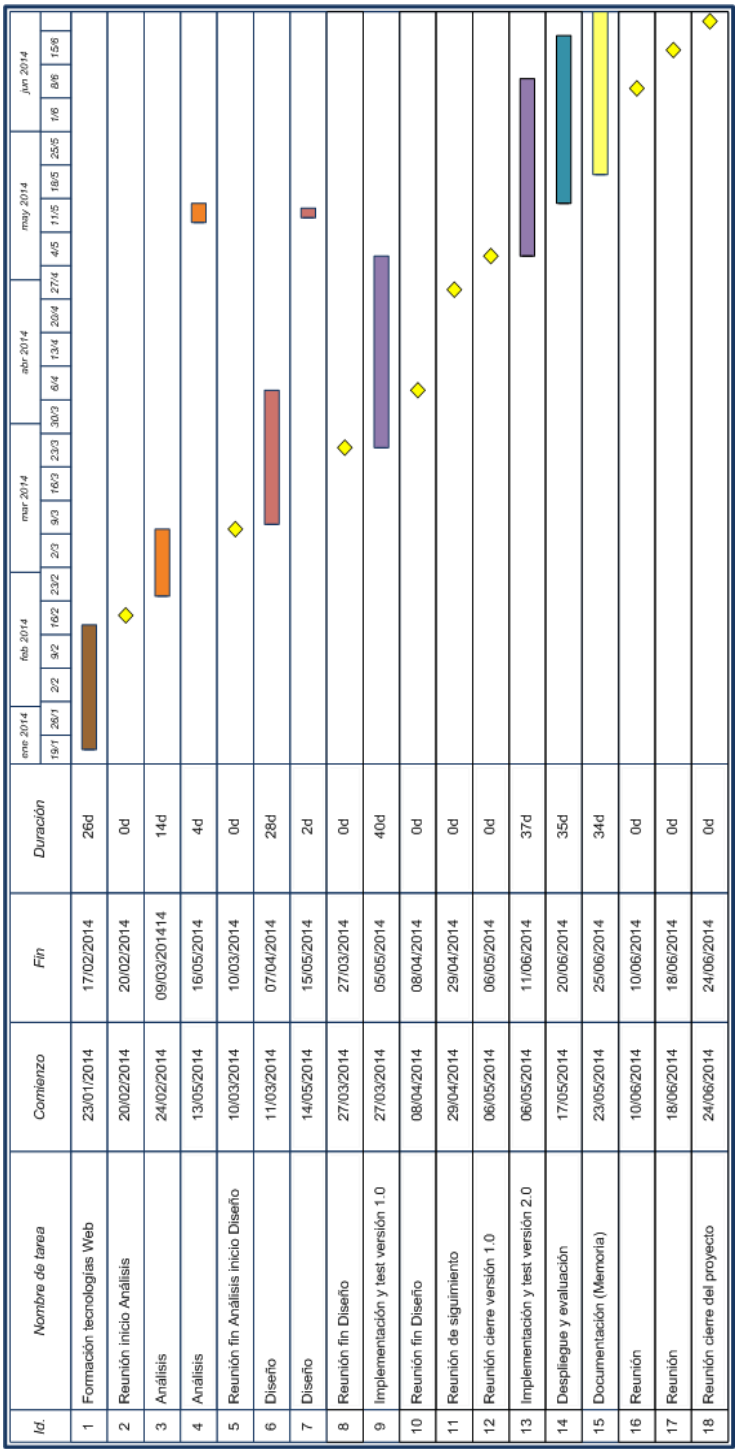
8.1.1. Gestión de configuraciones

Para facilitar la comunicación con los directores del proyecto se ha empleado una carpeta compartida en la plataforma *Dropbox*. Esta carpeta se ha empleado para almacenar la documentación generada a lo largo de cada una de las fases del proyecto.

Se ha empleado un repositorio privado *Subversion* para almacenar y realizar el control de versiones del código generado. Este repositorio ha sido de gran utilidad debido a que me ha permitido trabajar desde distintos dispositivos con una gestión totalmente automatizada del código.

8.1.2. Diagrama de Gantt

La Figura 19 muestra el diagrama de Gantt con la planificación realizada del proyecto.



Id.	Nombre de tarea	Comienzo	Fin	Duración
1	Formación tecnologías Web	23/01/2014	17/02/2014	26d
2	Reunión inicio Análisis	20/02/2014	20/02/2014	0d
3	Análisis	24/02/2014	09/03/201414	14d
4	Análisis	13/05/2014	16/05/2014	4d
5	Reunión fin Análisis inicio Diseño	10/03/2014	10/03/2014	0d
6	Diseño	11/03/2014	07/04/2014	28d
7	Diseño	14/05/2014	15/05/2014	2d
8	Reunión fin Diseño	27/03/2014	27/03/2014	0d
9	Implementación y test versión 1.0	27/03/2014	05/05/2014	40d
10	Reunión fin Diseño	08/04/2014	08/04/2014	0d
11	Reunión de siguimiento	29/04/2014	29/04/2014	0d
12	Reunión cierre versión 1.0	06/05/2014	06/05/2014	0d
13	Implementación y test versión 2.0	06/05/2014	11/06/2014	37d
14	Despliegue y evaluación	17/05/2014	20/06/2014	35d
15	Documentación (Memoria)	23/05/2014	25/06/2014	34d
16	Reunión	10/06/2014	10/06/2014	0d
17	Reunión	18/06/2014	18/06/2014	0d
18	Reunión cierre del proyecto	24/06/2014	24/06/2014	0d

Figura 19. Diagrama de Gantt con la planificación final del proyecto

Observando el diagrama se puede concluir que se ha seguido de forma estricta el modelo de desarrollo en cascada. También se puede observar como al final de cada fase se ha realizado una reunión de seguimiento que ha generado una retroalimentación del proceso y las fases.

8.1.3. Coste total del proyecto

Se ha llevado una contabilización de los esfuerzos dedicados a cada actividad del proyecto empleando una hoja de cálculo, cada punto de esfuerzo anotado sobre ella equivale a media hora de esfuerzo real. El coste total del proyecto seccionado por actividades se muestra en la Tabla 4 y la Figura 20.

Actividades	Esfuerzo total	Porcentaje relativo
Formación	54,5 h.	15%
Análisis	43,5 h.	12%
Diseño	47 h.	13%
Implementación	88 h.	24%
Despliegue y validación	47,5 h.	13%
Documentación	54 h.	15%
Gestión de proyecto	30 h.	8%
	364,5 h.	100%

Tabla 4. Esfuerzos totales y porcentaje relativo por actividad

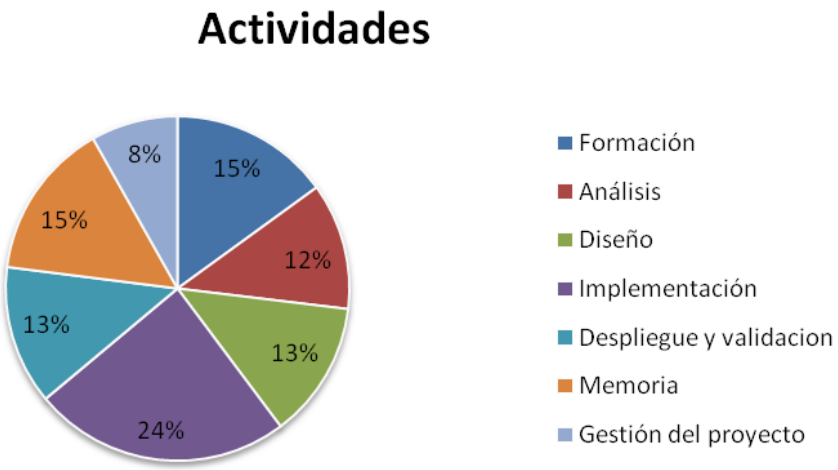


Figura 20. Gráfica con la división de los esfuerzos en actividades

Observando la gráfica cabe destacar que se ha realizado un gran esfuerzo en las fases principales de la Ingeniería del Software desde la fase temprana de análisis hasta la validación y evaluación del sistema. La actividad que más esfuerzo a requerido ha sido la de implementación del sistema

8.2. Conclusiones

Se ha desarrollado el proyecto siguiendo las fases principales de la Ingeniería del Software con el objetivo de conseguir un buen análisis y diseño de la solución. La realización de este trabajo ha supuesto afrontar, en lo personal, el trabajo de mayor envergadura hasta la fecha abordando todas las fases de desarrollo de Software. Ha sido necesario aplicar técnicas aprendidas a lo largo de la especialidad de Ingeniería del Software tanto para la gestión del proyecto como para el desarrollo del proyecto.

Las fases de análisis y diseño han requerido un gran esfuerzo debido a la importancia que tenía realizar un sistema flexible y escalable que se pudiera ejecutar sobre distintas tipos de instancias. Para ello la utilización de ficheros de propiedades configurables ha sido indispensable facilitando la reproducibilidad del sistema sin la necesidad de modificar el código fuente.

La fase de validación y evaluación del sistema ha permitido comprobar el correcto funcionamiento del sistema en una plataforma de computación en la nube. Por tanto, esta solución se podría aplicar sobre una plataforma similar con el objetivo de reducir los costes temporales y económicos empleando las instancias que mejor se adapten a cada componente.

Se han cumplido los objetivos planteados inicialmente y se ha introducido un sencillo sistema para la ayuda a la toma de decisiones basado en árboles de decisiones. La utilización de estos árboles permitiría a un componente intermedio lanzar el tipo de instancia indicada para realizar el procesamiento seleccionado.

El dominio de la aplicación ha sido novedoso, pero en general ha sido gratificante, debido a que en la actualidad se está realizando un gran esfuerzo en esta materia. Además la ayuda proporcionada por el investigador Jorge Álvarez en las fases iniciales del trabajo me permitió comprender la forma de operar con el sistema *PhyloFlow* [2].

La introducción de un sistema de ayuda a la toma de decisiones ha permitido hacer una evaluación inicial del rendimiento del sistema que, una vez en explotación permitiría generar estrategias sofisticadas de planificación y asignación dinámica de recursos que incluso tuviera en cuenta cuestiones de coste económico.

Personalmente deseaba que el trabajo de fin de grado que realizase me permitiría aprender y conocer nuevas tecnologías Web, en concreto, aquellas relacionadas con los servicios Web. La realización de este trabajo no sólo me ha permitido formarme en el diseño e implementación de servicio Web, sino que me ha brindado la oportunidad de emplear una plataforma de computación en la nube.

Por último, la ayuda que me han proporcionado tanto Javier como Gregorio para desarrollar el proyecto ha sido importantísima. Hemos realizado periódicamente reuniones de seguimiento del trabajo que me han permitido obtener una retroalimentación constante por su parte.

Bibliografía

[1] Blanco et al. Rebooting the human mitochondrial phylogeny: an automated and scalable methodology with expert knowledge. BMC Bioinformatics (12):174. 2011

[2] Jorge Álvarez. Análisis filogenético molecular: Diseño e implementación de algoritmos escalables y fiables y verificación automática de propiedades de una filogenia.

[3] HTCondor. High Throughput Computing. URL: http://research.cs.wisc.edu/htcondor/

[4] DAGMan. Directed Acyclic Graph Manager.
URL: http://research.cs.wisc.edu/htcondor/dagman/dagman.html

[5] Anduril. Component-based workflow framework. URL: http://www.anduril.org/anduril/site

[6] Ovaska et al. Large-scale data integration framework provides a comprehensive view on glioblastoma multiforme. Genome Medicine 2010 2:65.

[7] Java. Java Programming Language. URL: http://java.com/es/

[8] Perl. The Perl Programming Language. URL: http://www.perl.org/

[9] Phyton. Python Programming Language. URL: https://www.python.org/

[10] Matlab. Matlab, the Language of Technical Computing.
URL: http://www.mathworks.com/products/matlab/

[11] Galaxy. Web-based platform for biomedical research. URL: http://galaxyproject.org/

[12] Liu et al. Cloud-based bioinformatics workflow platform for large-scale next-generation sequencing analyses. J BiomedInform (2014)

[13] Globus Provision. Globus Transfer System. URL: https://www.globus.org/

[14] Maximum Likelihood. URL: http://en.wikipedia.org/wiki/Maximum_likelihood

[15] Evert Duipmans. Business Process Management in the cloud: Business Process as a Service (BPaaS). 2012

[16] Don Jones. White paper: The Business Case for Software as a Service (SaaS). 2010

[17] Fujitsu. White paper: SaaS Business Enablement Services from Fujitsu. 2011

[18] BioPython. Tool for biological computation. URL: http://biopython.org/wiki/Main_Page

[19] DendroPy. Library for phylogenetic computing. URL: https://pythonhosted.org/DendroPy/

[20] Maven. Software project management. URL: http://maven.apache.org/

[21] ZIP. ZIP archive file format. URL: http://en.wikipedia.org/wiki/Zip_(file_format)

[22] GZIP. GZIP archive file format. URL: http://en.wikipedia.org/wiki/Gzip

[23] SMTP. Simple Mail Transfer Protocol.
URL: http://en.wikipedia.org/wiki/Simple_Mail_Transfer_Protocol

[24] IMAP. Internet Message Acces Protocol.
URL: http://en.wikipedia.org/wiki/Internet_Message_Access_Protocol

[25] KNIME. Konstanz Information Milner, data analytics, reporting and integration platform.
URL: https://www.knime.org/

Anexo I. Análisis

1. Formatos de entrada

Formato FASTA

Una secuencia bajo el formato FASTA comienza con una descripción en una única línea (línea de cabecera), seguida por líneas de datos de secuencia. La línea de descripción se distingue de los datos de secuencia por un símbolo '>' en la primera columna. A continuación, le siguen el identificador de la secuencia, y el resto de la línea es la descripción (ambos son opcionales). Cada una de las líneas de la secuencia está compuesta por 60 caracteres. La secuencia termina si aparece otra línea comenzando con el símbolo '>'; esto indica el comienzo de otra secuencia. Un ejemplo simple de una secuencia sintética (*SB.fasta*) en el formato FASTA es el siguiente:

```
>SB
GACTCCCGCGCGGTGGGCCACGCGGAGCCCGGTCATATACCCTATAACCGGCCGTTGCAC
GTTAGCCATACATTCCCTTCTAATGAATCTACGAGCTCGGCATTAGGATCAATAGGTGAG
CTAAGCCAGAATAGACTCTAATAGACTTTTCGTGTGCTCTATCCTGCTAGCTTGTTTATG
TCCCCATGTGGACAGCATGTAGCTGGGATTGCATATTTTCCGCGGTTATCCGACCAGCAG
CTGTGAACACGGCGAGAGCATCTGAAACATTCTGTAGAGTCAGGTTTAATTAGTCTTTAC
CTATAAGGTCATTATCTTATGTCGTCTATGACCGTTGGAACCTATCACTTATTCTAGCAG
AAACAGATTTCGAGTAATTTGCGTTTATGTGGTCAGGAATGAGAGTGTTGATTGAGCCGC
GGCTTTCGGGCAACAATTGATACGCTTGAAGGATAGATCCCTAAATACTTGAGTACGTGG
GAGAAAAAACTCCTAGGCGAAATTGTGGAAAGTAATTACATGTTAGTCCATAAGTACCAA
AGGGAATTGGGCCTGTCCAACACTCGTTAGGCAACACGTCGTATCCGCCGAACACTATTC
TAATGCCTGTGCCGCCATAGCTAAACTTCACACGACGGAACCTGACCAGATCAAGACGGC
TCGAAATGGAAAAATAGGTGACGTATACCGTGGTAGCCACAAGCCCATAGCCGGCCTGGG
TGACTCTCTCTCAGTTCGCGACCTTCGGCTCATTGGTGTAAGACCTAATTGGGCTATTTT
ATTCCAATGTTGCAAGCTAATAAGTAAGCGTGAGTCATTAGTTAATTTCGAACCTACGGT
CGTGGCGCAATTGAAACCAAGTACCTCATATTTATAAGTAGGCCGCTGAGGTCAAGCGTC
TAAACCTTAAATCCGCCGGGTACAATCACCGAGGGGTGTCCATGTGCCTGTACCACGCAG
TATCACTGCGCCGCCGAACGGAAAAGATGCGCCTCCGCGCCCCTAACCTCATTCGTCGGA
GGCGATCCGTTA
```

Formato NEWICK

En matemáticas, el formato de árboles Newick, la notación Newick o formato de árbol New Hampshire es una manera de representar arboles de grafos teóricos con longitudes de las ramas empleando comas y paréntesis. Fue adoptado por James Archie, William H. E. Day, Joseph Felsenstein, Wayne Maddison, Christopher Meacham, F. James Rohlf, y David Swofford en dos reunión en 1986, la segunda de ellas tuvo lugar en el restaurante Newick en Dover, New Hampshire, US. El formato adoptado es una generalización del formato desarrollado por Meacham en 1984 para el primer programa de dibujo de un árbol en Felsentein *PHYLIP package*. La representación es la siguiente:

```
Tree --> Subtree ";" | Branch ";"
   Subtree --> Leaf | Internal
   Leaf --> Name
   Internal --> "(" BranchSet ")" Name
   BranchSet --> Branch | BranchSet "," Branch
   Branch --> Subtree Length
   Name --> empty | string
   Length --> empty | ":" number
```

2. Casos de uso

Casos de uso: usuario administrado

La Figura 21 muestra los casos de uso de nivel 1 para los usuarios del sistema.

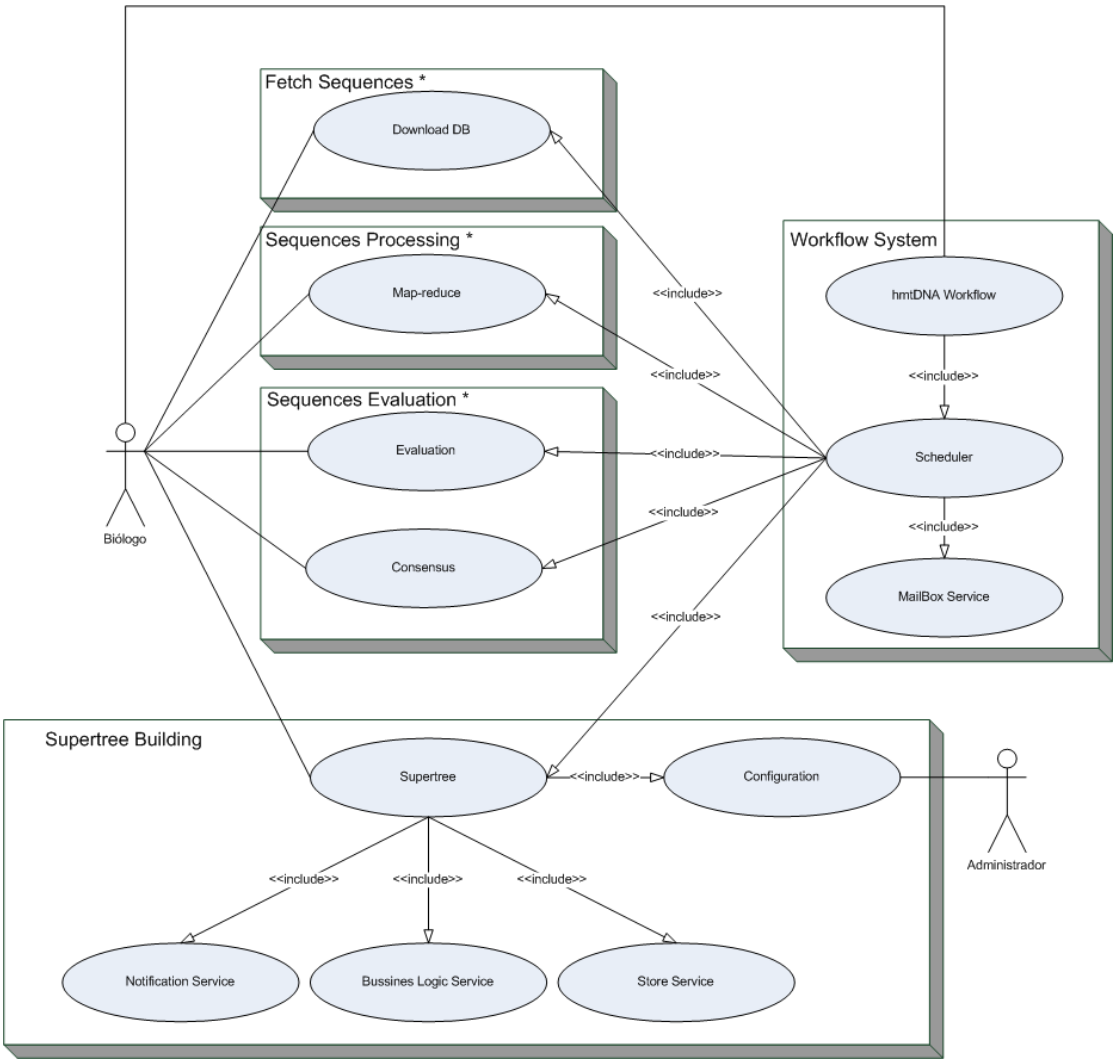


Figura 21. Diagrama de casos de uso de nivel 1 para los actores principales del sistema

* Por simplificar el diagrama se han omitido el resto de casos de uso de los subsistemas *Fetch Sequences, Sequences Processing y Sequecences Evaluation*. Éstos y sus relaciones son idénticos a las mostradas en el sistema que denominado *Supertree Building*.

El actor principal, administrador, puede interactuar con el caso de uso *Configuration* que se encuentra en los cuatro subsistemas que se corresponden a los componentes del sistema *PhyloFlow* [2]. Este caso de uso permite al administrador configurar la máquina en la que reside el subsistema para poder desplegar y lanzarlo correctamente.

3. Descripción de los casos de uso

DownloadDB

Pre: --

Post: Se descarga la última versión de la base de datos de Gen Bank y se almacena en un sistema de almacenamiento externo posibilitando su acceso al usuario.

Descripción: El caso de uso comienza cuando el usuario desea obtener la última versión de la base de datos de Gen Bank que contiene rodas las secuencias de ADN mitocondrial. El usuario envía la petición al servicio y una vez finalizada la descarga será notificado. En el caso de que la petición sea sintácticamente incorrecta el sistema le notificará inmediatamente. En caso contrario, procederá a llevar a cabo la descarga de las secuencias de ADN mitocondrial humano de la base de datos de Gen Bank. El sistema se encarga de solicitar al sistema *PhyloFlow* [2] que realice la descarga, recuperar los resultados, almacenarlos en el sistema externo y notificar al usuario la ubicación del archivo que contiene todas las secuencias de ADN mitocondrial humano albergadas en Gen Bank.

Flujo de eventos

Camino básico del caso de uso "DownloadDB"	
Usuario	**Sistema**
1. Envía la petición de descarga del ADN mitocondrial	
	2. Include Business Logic Service
	3. Include Store Service
	4. Include Notification Service
Camino alternativo	
Evento 1. El usuario ha introducido algún error en la petición. El sistema responde de forma síncrona informando del error al usuario.	
Evento 2. Se produce un error interno del sistema *PhyloFlow* que impide realizar la descargar. El sistema notifica al usuario el error producido de forma asíncrona.	
Evento 3. Se produce un error al almacenar el resultado en el sistema de almacenamiento externo. El sistema responde de forma síncrona informando del error al usuario.	

Map-reduce

Pre: El fichero de secuencias de ADN mitocondrial humano de entrada es válido.
Post: Se realiza el procedimiento biomédico solicitado sobre las secuencias de entrada de ADN mitocondrial y se almacena en un sistema de almacenamiento externo accesible al usuario.
Descripción: El caso de uso comienza cuando el usuario desea realizar un procesamiento biológico sobre las secuencias de ADN mitocondrial que posee. El usuario envía la petición al servicio y una vez finalizado el procesamiento biomédico será notificado. En el caso de que la petición sea sintácticamente incorrecta el sistema le notificará inmediatamente. En caso contrario, el sistema se encarga de solicitar al sistema *PhyloFlow* [2] que realice el procesamiento solicitado sobre el fichero de secuencias de entrada, recuperar los resultados, almacenarlos en el sistema externo y notificar al usuario la ubicación del archivo que contiene los resultados obtenidos.

Flujo de eventos

Camino básico del caso de uso "Supertree"	
Usuario	**Sistema**
1. Envía la petición de descarga del ADN mitocondrial incluye la URI que da acceso al fichero de datos de secuencia	
	2. Include Store Service
	3. Include Business Logic Service
	4. Include Store Service
	5. Include Notification Service
Camino alternativo	
Evento 1. El usuario ha introducido algún error en la petición. El sistema responde de forma síncrona informando del error al usuario.	
Evento 2. Se produce un error al descargar el fichero de entrada. El sistema responde de forma asíncrona informado del error al usuario.	
Evento 3. Se produce un error interno del sistema *PhyloFlow* que impide realizar el procedimiento biomédico solicitado. El sistema notifica al usuario el error producido de forma asíncrona.	
Evento 4. Se produce un error al almacenar el resultado en el sistema de almacenamiento externo. El sistema responde de forma asíncrona informando del error al usuario.	

Evaluation

Pre: El fichero de secuencias de ADN mitocondrial humano de entrada es válido.

Post: Se realiza la evaluación en base a modelos biológicos solicitados sobre las secuencias de entrada de ADN mitocondrial, creando de forma opcional fichero similares para realizar un análisis estadístico de la solución parcial. Finalmente, se almacenan los resultados en un sistema de almacenamiento externo accesible al usuario.

Descripción: El caso de uso comienza cuando el usuario desea realizar una evaluación mediante modelos biológicos sobre las secuencias de ADN mitocondrial que posee. El usuario envía la petición al servicio y una vez finalizado el procesamiento biomédico será notificado. En el caso de que la petición sea sintácticamente incorrecta el sistema le notificará inmediatamente. En caso contrario, el sistema se encarga de solicitar al sistema *PhyloFlow* [2] que realice la evaluación empleando los modelos biológicos disponibles sobre el fichero de secuencias de entrada. Opcionalmente, si el usuario los solicita, el sistema *PhyloFlow* [2] generará ficheros similares y evaluará el mejor modelo sobre ellos. Por último, el sistema se encargará recupera los resultados, almacenarlos en el sistema externo y notificar al usuario la ubicación del archivo que contiene los resultados obtenidos.

Flujo de eventos

Camino básico del caso de uso "Supertree"	
Usuario	**Sistema**
1. Envía la petición de descarga del ADN mitocondrial incluye la URI que da acceso al fichero de datos de secuencia	
	2. Include Store Service
	3. Include Business Logic Service
	4. Include Store Service
	5. Include Notification Service
Camino alternativo	
Evento 1. El usuario ha introducido algún error en la petición. El sistema responde de forma síncrona informando del error al usuario.	
Evento 2. Se produce un error al descargar el fichero de entrada. El sistema responde de forma asíncrona informado del error al usuario.	
Evento 3. Se produce un error interno del sistema *PhyloFlow* que impide realizar el procedimiento de evaluación con modelos biológicos. El sistema notifica al usuario el error producido de forma asíncrona.	
Evento 4. Se produce un error al almacenar el resultado en el sistema de almacenamiento externo. El sistema responde de forma asíncrona informando del error al usuario.	

Consensus

Pre: El fichero de secuencias de ADN mitocondrial humano de entrada es válido o de árboles filogenéticos.

Post: Se realiza un procedimiento de consenso sobre las secuencias de entrada de ADN mitocondrial. Finalmente, se almacenan los resultados en un sistema de almacenamiento externo accesible al usuario.

Descripción: El caso de uso comienza cuando el usuario desea realizar una fase de consenso sobre las secuencias de ADN mitocondrial que posee o sobre los árboles filogenéticos si ha realizado previamente una generación de bootstraps. El usuario envía la petición al servicio y una vez finalizado el consenso de los ficheros de secuencias o árboles filogenéticos será notificado. En el caso de que la petición sea sintácticamente incorrecta el sistema le notificará inmediatamente. En caso contrario, el sistema se encarga de solicitar al sistema *PhyloFlow* [2] que realice el procedimiento de consenso sobre el fichero de secuencias de entrada. Por último, el sistema se encargará recupera los resultados, almacenarlos en el sistema externo y notificar al usuario la ubicación del archivo que contiene los resultados obtenidos.

Flujo de eventos

Camino básico del caso de uso "Supertree"	
Usuario	**Sistema**
1. Envía la petición de descarga del ADN mitocondrial incluye la URI que da acceso al fichero de datos de secuencia	
	2. Include Store Service
	3. Include Business Logic Service
	4. Include Store Service
	5. Include Notification Service
Camino alternativo	
Evento 1. El usuario ha introducido algún error en la petición. El sistema responde de forma síncrona informando del error al usuario.	
Evento 2. Se produce un error al descargar el fichero de entrada. El sistema responde de forma asíncrona informado del error al usuario.	
Evento 3. Se produce un error interno del sistema *PhyloFlow* que impide realizar el procedimiento de consenso. El sistema notifica al usuario el error producido de forma asíncrona.	
Evento 4. Se produce un error al almacenar el resultado en el sistema de almacenamiento externo. El sistema responde de forma asíncrona informando del error al usuario.	

Supertree

Pre: El fichero de secuencias árboles filogenéticos de entrada es válido.

Post: Se realiza la construcción del superárbol a partir de los árboles filogenéticos generados previamente. Finalmente, se almacenan los resultados en un sistema de almacenamiento externo accesible al usuario.

Descripción: El caso de uso comienza cuando el usuario desea realizar la construcción de un único árbol filogenético (superárbol) a partir de los ficheros de árboles filogenéticos que posee. El usuario envía la petición al servicio y una vez finalizado el consenso de los ficheros de secuencias será notificado. En el caso de que la petición sea sintácticamente incorrecta el sistema le notificará inmediatamente. En caso contrario, el sistema se encarga de solicitar al sistema *PhyloFlow* que realice la construcción del superárbol empleando el método y perfil seleccionados por el usuario. Por último, el sistema se encargará recupera los resultados, almacenarlos en el sistema externo y notificar al usuario la ubicación del archivo que contiene los resultados obtenidos.

Flujo de eventos

Camino básico del caso de uso "Supertree"	
Usuario	**Sistema**
1. Envía la petición de descarga del ADN mitocondrial incluye la URI que da acceso al fichero de datos de secuencia	
	2. Include Store Service
	3. Include Business Logic Service
	4. Include Store Service
	5. Include Notification Service
Camino alternativo	
Evento 1. El usuario ha introducido algún error en la petición. El sistema responde de forma síncrona informando del error al usuario.	
Evento 2. Se produce un error al descargar el fichero de entrada. El sistema responde de forma asíncrona informado del error al usuario.	
Evento 3. Se produce un error interno del sistema *PhyloFlow* que impide realizar el procedimiento de construcción del superárbol. El sistema notifica al usuario el error producido de forma asíncrona.	
Evento 4. Se produce un error al almacenar el resultado en el sistema de almacenamiento externo. El sistema responde de forma asíncrona informando del error al usuario.	

Business Logic Service

Pre: -

Post: Se captura la salida estándar y la salida de error generada por el sistema *PhyloFlow* [2]. Si el proceso ha finalizado con éxito el sistema recupera los resultados generados por el sistema *PhyloFlow* [2].

Descripción: El caso de uso comienza cuando el sistema desea obtener los resultados proporcionaos por una de las funcionalidades ofrecidas por el sistema *PhyloFlow* [2]. El sistema crea un proceso que invoca por línea de comando al sistema *PhyloFlow* [2], introduciendo por la entrada estándar los parámetros seleccionados. Se genera uno o más procesos que realizan la funcionalidad solicitada. Por último, el sistema recoge los resultados si el procesamiento ha tenido éxito. En caso contrario, captura el error que se ha producido a través de la salida del sistema *PhyloFlow* [2].

Flujo de eventos

Camino básico del caso de uso "Business Logic Service"	
Sistema	**Sistema *PhyloFlow***
1. Genera un proceso, el cuál mediante línea de comandos invoca al sistema *PhyloFlow*	
	2. Realiza el procesamiento solicitado
3. Captura la salida de error	
4. Recupera los resultados	

Store Service

Pre: -

Post: Se realiza la descarga de un recurso identificado por una URI o se procede al almacenamiento de un recurso en el sistema de almacenamiento externo.

Descripción: El caso de uso comienza cuando el sistema desea realizar la descarga de un recurso en base a una URI que posee. El sistema envía la petición al servicio y sus credenciales de acceso. Si las credenciales son válidas y posee permisos para realizar las acción solicitada, lectura o escritura, el sistema de almacenamiento externo realizará la acción solicitada. Por último, el sistema recuperará los archivos descargados o la URI que identifica al recurso almacenado.

Flujo de eventos

Camino básico del caso de uso "Store Service"	
Sistema	**Sistema externo de almacenamiento**
1. Solicita la descarga de un archivo identificado por una URI	
	2. Comprueba las credenciales y las reglas de seguridad.
	3. El sistema comprueba la URI. Si es correcta, descarga el archivo
4. Recupera el archivo descargado	
Camino alternativo	
Evento 2. Las credencias de acceso son inválidas o no posee permisos de lectura. El sistema de almacenamiento generara un error y el sistema lo captura.	
Evento 3. Se produce un error interno del sistema de almacenamiento externo debido a un error interno o una URI incorrecta. El sistema capturar el error.	

Camino básico II del caso de uso "Store Service"	
Sistema	**Sistema externo de almacenamiento**
1. Solicita el almacenamiento de un archivo	
	2. Comprueba las credenciales y las reglas de seguridad. Almacena el archivo
	3. Almacena el archivo y devuelve una URI con su ubicación
4. El sistema recupera la URI	
Camino alternativo	
Evento 2. Las credencias de acceso son inválidas o no posee permisos de escritura. El sistema de almacenamiento generara un error y el sistema lo captura.	
Evento 3. Se produce un error interno del sistema de almacenamiento externo debido a un error interno o una URI incorrecta. El sistema captura el error.	

Notification Service

Pre: El sistema posee las credenciales válidas para acceder al sistema remoto.

Post: Se notifica al usuario final los resultados obtenidos a causa de la petición que realizó previamente.

Descripción: El caso de uso comienza cuando el sistema desea notificar al usuario final los resultados que ha obtenido a causa de la petición que realizó previamente el usuario. El sistema envía las credenciales de acceso al sistema remoto de notificación. Si las credenciales son válidas, el sistema tendrá acceso y podrá notificar los resultados al cliente. En caso contrario, se generará un error.

Flujo de eventos

Camino básico del caso de uso "Notification Service"	
Sistema	**Sistema externo de notificación**
1. Solicita la notificación de los resultados al cliente.	
	2. Comprueba las credenciales y las reglas de seguridad.
	3. Notifica los resultados al cliente.
Camino alternativo	
Evento 2. Las credencias de acceso son inválidas. El sistema de notificación generara un error y el sistema lo captura.	

Configuration

Pre: -

Post: Se realiza la configuración del sistema para desplegarlo y lanzarlo de forma correcta sobre la máquina en la que reside.

Descripción: El caso de uso comienza cuando el usuario desea configurar la máquina en la que se encuentra el sistema para poder utilizarlo. Para ello modifica los ficheros de configuración del subsistema estableciendo las variables de entorno, rutas y credenciales de acceso de cada servicio que compone el sistema. Por último, se realiza el despliegue del sistema. Si el sistema está bier configurado su puesta en marcha será correcta.

Flujo de eventos

Camino básico del caso de uso "hmtDNA Workflow"	
Usuario	**Sistema**
1. Modifica los ficheros de configuración del sistema	
2. Despliega el sistema sobre un servidor de aplicaciones	
	4. El sistema se actualiza y se lanza
Camino alternativo	
Evento 4. El usuario ha configurado incorrectamente el sistema, por tanto, se genera algún tipo de error durante la puesta en marcha del mismo.	

hmtDNA Workflow

Pre: -

Post: Se realiza un flujo de trabajo completo para la inferencia filogenética de un árbol evolutivo sobre un conjunto de secuencias de ADN mitocondrial humano.

Descripción: El caso de uso comienza cuando el usuario desea realiza un análisis del filogenético sobre un árbol evolutivo sobre un conjunto de secuencias de ADN mitocondrial humano. El usuario envía la petición al servicio y una vez finalizado el flujo de trabajo completo para la inferencia de un árbol filogenético será notificado. En el caso de que la petición sea sintácticamente incorrecta el sistema le notificará inmediatamente. En caso contrario, el sistema se encarga de integrar cada uno de los servicios que se corresponden con cada componente del flujo de trabajo para llevarlo a cabo.

Flujo de eventos

Camino básico del caso de uso "hmtDNA Workflow"	
Usuario	**Sistema**
1. Envía la petición de realización de un flujo de trabajo completo para secuencias de ADN mitocondrial.	
	2. Include Scheduler
Camino alternativo	
Evento 1. El usuario ha introducido algún error en la petición. El sistema responde de forma síncrona informando del error al usuario.	

Scheduler

Pre: -

Post: Se realiza un flujo de trabajo completo para la inferencia filogenética de un árbol evolutivo sobre un conjunto de secuencias de ADN mitocondrial humano integrando cada uno de los componentes del sistema *PhyloFlow* [2].

Descripción: El caso de uso comienza cuando el sistema solicita la realización de un flujo completo de ADN mitocondrial humano. Para ello solicita de forma secuencial y ordenada cada uno de los componentes que forman el flujo de trabajo la realización de uno o varios trabajos concretos. Emplea los resultados parciales de cada componente como la entrada de datos del siguiente componente.

Flujo de eventos

Camino básico del caso de uso "*Scheduler*"	
Sistema	**Componente *Fetch Sequences***
1. Envía la petición de descarga de la base de datos de ADN mitocondrial humano de *GenBank*	
	2. Include DownloadDB
3. Espera hasta obtener los resultados solicitados.	
Camino alternativo	
Evento 1. Se ha producido un error durante la realización de la petición y el mensaje recibido así lo indica.	

Flujo de eventos

Camino básico del caso de uso "*Scheduler*"	
Sistema	**Componente *Sequences Processing***
1. Envía la petición de realizar el procesamiento biológico sobre las secuencias de entrada.	
	2. Include Map-reduce
3. Espera hasta obtener los resultados solicitados.	
Camino alternativo	
Evento 1. Se ha producido un error durante la realización de la petición y el mensaje recibido así lo indica.	

Flujo de eventos

Camino básico del caso de uso "*Scheduler*"	
Sistema	**Componente *Sequences Evalutaion***
1. Envía la petición para la evaluación de modelos evolutivos sobre las secuencias de entrada.	
	2. Include Evaluation
3. Espera hasta obtener los resultados solicitados.	
Camino alternativo	
Evento 1. Se ha producido un error durante la realización de la petición y el mensaje recibido así lo indica.	

Flujo de eventos

Camino básico del caso de uso "*Scheduler*"	
Sistema	**Componente *Sequences Evalutaion***
1. Envía la petición para la generación de un único árbol filogenético consensuado a partir de las soluciones parciales de entrada.	
	2. Include Consensus
3. Espera hasta obtener los resultados solicitados.	
Camino alternativo	
Evento 1. Se ha producido un error durante la realización de la petición y el mensaje recibido así lo indica.	

Flujo de eventos

Camino básico del caso de uso "*Scheduler*"	
Sistema	**Componente *Supertree Building***
1. Envía la petición para la generación de un único superárbol a partir de los resultados parciales de entrada obtenidos previamente	
	2. Include Supertree
3. Espera hasta obtener los resultados solicitados.	
Camino alternativo	
Evento 1. Se ha producido un error durante la realización de la petición y el mensaje recibido así lo indica.	

MailBox Service

Pre: -

Post: Se recupera el mensaje solicitado por la entrada.

Descripción: El caso de uso comienza cuando el sistema *Scheduler* recuperar un mensaje que se corresponde con la notificación de unos de los servicios integrados. Este mensaje contiene la respuesta a su petición. El sistema *Scheduler* solicita la recuperación del mensaje, el sistema comprueba las credenciales de acceso. Si las credenciales son válidas y el mensaje se encuentra en la bandeja de entrada el sistema entregará el mensaje solicitado. En caso contrario, el sistema seguirá consultado la bandeja de entrada cada cierto tiempo hasta obtener el mensaje solicitado.

Flujo de eventos

Camino básico del caso de uso "MailBox Service"	
Sistema *Scheduler*	**Sistema**
1. Solicita la recuperación de un mensaje o notificación.	
	2. Comprueba las credenciales de acceso.
	3. Cada cierto tiempo recupera los mensajes y notificaciones sin leer. Si coincide con el mensaje lo pedido.
4. Recupera el mensaje solicitado	
Camino alternativo	
Evento 2. Las credencias de acceso son inválidas. El sistema de almacenamiento generara un error y el sistema lo captura.	
Evento 3. El mensaje solicitado no se ha recibido aún. Se mantiene en este estado hasta que el mensaje solicitado se encuentre en la bandeja de entrada.	
Evento 3. Se produce un error interno del sistema de lectura de notificaciones. El sistema captura el error.	

Anexo II. Diseño de la solución

1. API REST de cada componente que constituye el flujo de trabajo

Componente: Fetch Sequences

Identidad: *Fetch Sequences*, ofrece una operación relacionada con la recuperación de las distintas bases de datos de secuencias disponibles..

DownloadDB

Descripción: se encarga de recuperar la base de datos para el tipo de secuencia seleccionado. Si el tipo de secuencia es "hmtDNA" descargara la última versión de la base de datos de *GenBank* con todas las secuencias de ADN mitocondrial disponibles.

Petición:

Método	URL
POST	/phyloflow/fetchsequences/download

Parámetros	Estilo	Tipo	Descripción
input	plain	api:Input	Datos de entrada
downloadDB	plain	api:Operation	Operación a realizar
data_type	plain	xsd:string	Tipo de secuencia de datos
email	plain	xsd:string	Correo electrónico para notificar en caso de abusos
notification	plain	api:Notification	Notificación
email(choice)	plain	xsd:string	Correo electrónico para notificar los resultados
other(choice)	plain	xsd:string	Otro tipo de notificación

Respuesta:

Estado	Respuesta
200	"Ok your process ID is [PID]"
400	"Syntax Error: [error]"
500	"Internal Error, try again later"

Componente: Processing Sequences

Identidad: *Processing Sequences*, ofrece una operación relacionada con el procesamiento biológico de secuencias empleando técnicas de *map-reduce* para reducir los costes temporales en base a procesar conjunto de datos más pequeños

Map-reduce

Descripción: se encarga de realizar el procesamiento biológico de las secuencias de entrada en base al método seleccionado.

Petición:

Método	URL
POST	/phyloflow /sequencesprocessing/map-reduce

Parámetros	Estilo	Tipo	Descripción
input	plain	api:Input	Datos de entrada
map-reduce	plain	api:Operation	Operación a realizar
data_type	plain	xsd:string	Tipo de secuencia de datos
method	plain	xsd:string	Método que se desea emplear para el procesamiento de las secuencias
num_retries	plain	xsd:integer	Número total de reintentos
do_reduce	plain	xsd:boolean	*Flag* de reducción de los resultados
ref_seq (optional)	plain	xsd:string	Referencia de la secuencia, si el método de procesamiento la requiere
data	plain	api:Data	Datos de entrada
URI	plain	xsd:string	Identificador de los recursos
id_file (optional)	plain	xsd:string	Identificador de los ficheros si es necesario
directory (optional)	plain	xsd:string	Identificador de los directorios si es necesario
notification	plain	api:Notification	Notificación
email(choice)	plain	xsd:string	Correo electrónico para notificar los resultados
other(choice)	plain	xsd:string	Otro tipo de notificación

Respuesta:

Estado	Respuesta
200	“Ok your process ID is [PID]”
400	“Syntax Error: [error]”
500	“Internal Error, try again later”

Componente: Evaluation Sequences

Identidad: *Evaluation Sequences*, ofrece operaciones para la evaluación de modelos y la generación de árboles filogenéticos consensuados en base a soluciones parciales. Opcionalmente se puede realizar un análisis estadístico realizando una fase de *bootstrapping*.

Evaluation

Descripción: se encarga de evaluar los ficheros para el tipo de secuencia de entrada y de seleccionar el mejor modelo de entre todos los evaluados. Además, si se desea, se encarga de generar *bootstraps* a partir de los ficheros para el tipo de secuencia de entrada y de evaluar los *bootstraps* generados con el mejor modelo obtenido previamente.

Petición:

Método	URL
POST	/phyloflow /sequencesevaluation/evaluation

Parámetros	Estilo	Tipo	Descripción
input	plain	api:Input	Datos de entrada
evaluate	plain	api:Operation	Operación a realizar
data_type	plain	xsd:string	Tipo de secuencia de datos
method	plain	xsd:string	Método que se desea emplear para la evaluación de la secuencias
num_retries	plain	xsd:integer	Número total de reintentos
models (optional)	plain	xsd:string	Modelos para realizar la evaluación
num_bootstraps (optional)	plain	xsd:integer	Número total de ficheros *bootstraps* a generar
bs_method (optional)	plain	xsd:string	Método de *bootstrapping*
cons_method (optional)	plain	xsd:string	Método de consenso de resultados parciales
data	plain	api:Data	Datos de entrada
URI	plain	xsd:string	Identificador de los recursos
id_file (optional)	plain	xsd:string	Identificador de los ficheros si es necesario
directory (optional)	plain	xsd:string	Identificador de los directorios si es necesario

notification	plain	api:Notification	Notificación
email(choice)	plain	xsd:string	Correo electrónico para notificar los resultados
other(choice)	plain	xsd:string	Otro tipo de notificación

Respuesta:

Estado	Respuesta
200	“Ok your process ID is [PID]”
400	“Syntax Error: [error]”
500	“Internal Error, try again later”

Consensus

Descripción: se encarga de generar un único árbol filogenético consensuado a partir del mejor modelo evolutivo.

Petición:

Método	URL
POST	/phyloflow /sequencesevaluation/consense

Parámetros	Estilo	Tipo	Descripción
input	plain	api:Input	Datos de entrada
consensus	plain	api:Operation	Operación a realizar
method	plain	xsd:string	Método de consensos de soluciones parciales
after_bootstraps	plain	xsd:boolean	*Flag* que indica si la etapa de consenso es posterior a la etapa de *bootstrapping*
num_retries	plain	xsd:integer	Número total de reintentos
data	plain	api:Data	Datos de entrada
URI	plain	xsd:string	Identificador de los recursos
id_file (optional)	plain	xsd:string	Identificador de los ficheros si es necesario
directory (optional)	plain	xsd:string	Identificador de los directorios si es necesario
notification	plain	api:Notification	Notificación
email(choice)	plain	xsd:string	Correo electrónico para notificar los resultados
other(choice)	plain	xsd:string	Otro tipo de notificación

Respuesta:

Estado	Respuesta
200	"Ok your process ID is [PID]"
400	"Syntax Error: [error]"
500	"Internal Error, try again later"

Componente: Supertree Building

Identidad: *Supertree Building*, ofrece una operación para la generación de un superárbol a partir de resultados parciales.

Supertree

Descripción: se encarga de generar el proceso de construcción del superárbol para los tipos de datos de entrada en base al método seleccionado.

Petición:

Método	URL
POST	/ phyloflow /supertreebuilding/supertree

Parámetros	Estilo	Tipo	Descripción
input	plain	api:Input	Datos de entrada
supertree	plain	api:Operation	Operación a realizar
data_type	plain	xsd:string	Tipo de secuencia de datos
method	plain	xsd:string	Método de construcción de un superárbol
num_retries	plain	xsd:integer	Número total de reintentos
profile (optional)	plain	xsd:string	Perfil del superárbol a generar
data	plain	api:Data	Datos de entrada
URI	plain	xsd:string	Identificador de los recursos
id_file (optional)	plain	xsd:string	Identificador de los ficheros si es necesario
directory (optional)	plain	xsd:string	Identificador de los directorios si es necesario
notification	plain	api:Notification	Notificación
email(choice)	plain	xsd:string	Correo electrónico para notificar los resultados
other(choice)	plain	xsd:string	Otro tipo de notificación

Respuesta:

Estado	Respuesta
200	“Ok your process ID is [PID]”
400	“Syntax Error: [error]”
500	“Internal Error, try again later”

2. Interfaz de cada servicio que forma parte de la capa middleware

BussinesLogicService

Este elemento ofrece todas las operaciones de la lógica de negocio. Por tanto encapsula todas la problemática relacionadas con la generación de procesos y recuperación de las salidas. Los servicios que ofrece son los siguientes:

Operación: downloadDB (data_type, email)

Descripción: se encarga de descargar o copiar los ficheros de secuencia para el tipo de datos y de dividirlos en sets de secuencias que se almacenan en "data/sets" en formato tar.gz.

Parámetros:

Parámetro	Tipo	Representación
data_type	String	Tipo de secuencia
email	String	Email de contacto

Operación: map_reduce (data_type, method, num_retries, do_reduce, ref_seq)

Descripción: se encarga de realizar el método procesamiento seleccionado para el tipo de datos de entrada sobre los ficheros de secuencia almacenados en formato tar.gz en "*data/mapred*", almacenando en la misma carpeta los ficheros de secuencia en formato tar.gz. Si el *flag do_reduce* está activo genera un único fichero que contiene todos los resultados unidos.

Parámetros:

Parámetro	Tipo	Representación
data_type	String	Tipo de secuencia
method	String	Nombre del método que se desea emplear para el procesamiento de las secuencias
num_retries	int	Número total de reintentos
do_reduce	boolean	*Flag* de reducción
ref_seq	String	Referencia de secuencia

Operación: evaluate (data_type, eval_method, num_bootsraps, bs_method, cons_method, num_retries)

Descripción: se encarga de evaluar los ficheros almacenados en el directorio "*data/phylo*" en formato tar.gz, para el tipo de secuencia de entrada y de seleccionar el mejor modelo de entre todos los evaluados. Además si el número de bootstraps de entrada es mayor de 0, se encarga de generar el número total de bootstraps solicitados a partir de los ficheros para el tipo de secuencia de entrada y de evaluar los bootstraps generados con el mejor modelo obtenido.

Parámetros:

Parámetro	Tipo	Representación
data_type	String	Tipo de secuencia
eval_method	String	Método de evaluación
models	String	Modelos para realizar la evaluación
num_retries	int	Número total de reintentos
num_ bootsraps	int	Número total de ficheros a generar
bs_method	String	Método de *bootstrapping*
cons_method	String	Método de consenso

Operación: consensus (method, after_bootstraps, num_retries)

Descripción: Se encarga de generar un árbol filogenético en base al método de consenso seleccionado. Si el *flag after_bootstraps* está activo genera el árbol de consenso teniendo en cuenta el resultado de la evaluación de los bootstraps.

Parámetros:

Parámetro	Tipo	Representación
method	String	Método de consenso
after_bootstraps	boolean	*Flag* que índica si la etapa de consenso es posterior a la etapa de *bootstrapping*
num_retires	int	Número total de reintentos

Operación: supertree (data_type, method, num_retries)

Descripción: se encarga de generar el proceso de construcción del superárbol para los tipos de datos de entrada en base al método seleccionado.

Parámetros:

Parámetro	Tipo	Representación
data_type	String	Tipo de secuencia
method	String	Método de construcción del superárbol
profile	String	Perfil del superárbol
num_retries	int	Número de reintentos

StorageService

Este elemento ofrece operaciones relacionas con el almacenamiento, recuperación y eliminación de ficheros en un sistema de almacenamiento externo.

Operación: URI - storeFile (fichero)

Descripción: se encarga de almacenar en un sistema de almacenamiento externo el fichero solicitado. Devuelve la URI donde se encuentra el recurso.

Parámetros:

Parámetro	Tipo	Representación
fichero	File	Fichero a almacenar

Operación: File - downloadFile(URI)

Descripción: se encarga de descargar el recurso que se encuentra en la URI de entrada. Devuelve el fichero que corresponde al recurso descargado.

Parámetros:

Parámetro	Tipo	Representación
URI	URI	Identificador único de un recurso a descargar

Operación: removeFile (URI)

Descripción: se encarga de eliminar el recurso pasado como parámetro de entrada.

Parámetros:

Parámetro	Tipo	Representación
URI	URI	Identificador único de un recurso a eliminar

CompressionService

Este elemento ofrece operaciones relacionas con la compresión y descompresión de ficheros.

Operación: File – compress (fichero)

Descripción: se encarga de comprimir el fichero pasado por la entrada en un directorio temporal. Devuelve el fichero descomprimido en el directorio temporal.

Parámetros:

Parámetro	Tipo	Representación
fichero	File	Fichero a descomprimir

Operación: decompress (fichero)

Descripción: se encarga de descomprimir el fichero pasado en un directorio temporal. Devuelve el fichero descomprimido.

Parámetros:

Parámetro	Tipo	Representación
fichero	File	Fichero a descomprimir

NotificationService

Este elemento ofrece operaciones relacionas con el envió de notificaciones relacionadas con el final de un procesamiento.

Operación: notifícate (cliente, fichero)

Descripción: se encarga de notificar al cliente solicitado el fichero pasado por la entrada.

Parámetros:

Parámetro	Tipo	Representación
subject	String	Asunto de la notificación
client	String	Cliente a notificar
fichero	File	Fichero que contiene los resultados generado

3. API REST componente que representa el flujo de trabajo

Identidad: *Workflow System*, ofrece operaciones para la realización de un flujo de trabajo completo para el procesamiento y análisis filogenético de distintos tipos de secuencias.

Synthetic Workflow

Descripción: se encarga de gestionar el flujo de trabajo completo para el procesamiento y análisis filogenético de secuencias sintéticas integrando los componentes necesarios.

Petición:

Método	URL
POST	/phyloflow/workflow/synthetic

Parámetros	Estilo	Tipo	Descripción
email	plain	xsd:string	Correo electrónico para notificar los resultados

Respuesta:

Estado	Respuesta
200	"The workflow process has begun without problems. You will receive a message in your mailbox in a few hours"
400	"Syntax Error: invalid email"
500	"Internal Error, try again later"

hmtDNA Workflow

Descripción: se encarga de gestionar el flujo de trabajo completo para el procesamiento y análisis filogenético de secuencias de ADN mitocondrial humano integrando los componentes necesarios.

Petición:

Método	URL
POST	/phyloflow/workflow/hmtDNA

Parámetros	Estilo	Tipo	Descripción
email	plain	xsd:string	Correo electrónico para notificar los resultados

Respuesta:

Estado	Respuesta
200	“The workflow process has begun without problems. You will receive a message in your mailbox in a few hours”
400	“Syntax Error: invalid email”
500	“Internal Error, try again later”

4. Comunicación: estándares de intercambio

Estándar de entrada: descripción

El formato de intercambio de datos para la entrada está divido en 3 secciones. La primera sección es obligatoria y describe la operación a realizar y los parámetros de entrada necesarios para cada operación así como el tipo de los mismos. Los parámetros de entrada se pueden introducir en cualquier orden. El sistema de inferencia filogenética ofrece las siguientes operaciones:

- **downloadDB**: está operación descarga los datos deseados para el tipo de secuencia de entrada, cuyos parámetros son:
  - **data_type**: tipo de secuencia de datos
  - **email**: correo electrónico para informar al usuario en caso de abuso de descargas.
- **map-reduce**: está operación realiza el procesamiento deseado mediante el método seleccionado al tipo de secuencia de entrada, cuyos parámetros son:
  - **data_type**: tipo de secuencia de datos
  - **method**: tipo de procesamiento
  - **num_retries**: número total de reintentos
  - **do_reduce**: reduce los ficheros de salida a uno único
  - **ref_seq**: referencia a la secuencia (opcional)

- **evaluate**: está operación realiza la evaluación de modelos para el tipo de secuencia de entrada y de manera opcional realiza tanto la generación como la evaluación de bootstraps. Sus parámetros son:
  - **data_type**: tipo de secuencia de datos
  - **method**: tipo de procesamiento
  - **num_retries**: número total de reintentos
  - **models**: modelos a emplear en la evaluación (opcional)
  - **num_bootstraps**: número total de bootstraps a generar (opcional)
  - **bs_method**: método de creación de bootstraps (opcional)
  - **cons_method**: método de consenso (opcional)

- **consensus**: está operación realiza el consenso de los datos generados en la fase de evaluación de modelos:
  - **method**: método de consensos
  - **after_bootstraps**: booleano que indica si se ha de realizar antes o después de la creación de *bootstraps*, es decir, los tiene o no en cuenta para la realización del consenso
  - **num_retries**: número total de reintentos
- **supertree**: está operación genera el superárbol para los datos de entrada:
  - **data_type**: tipo de secuencia de datos
  - **method**: método de creación
  - **num_retries**: número total de reintentos
  - **profile**: perfil del superárbol (opcional)

La segunda sección describe los datos de entrada, los cuales se emplearán para realizar los procesamientos solicitados. Esta sección es obligatoria para todas las operaciones salvo para la operación *downloadDB*, ya que es la única operación que no procesa datos de entrada.

- **uri**: es una cadena de caracteres que representa el identificador único que permite el acceso al recurso que contiene o se corresponde a los datos de entrada
- **id_file**: es una cadena de caracteres que representa una expresión regular para seleccionar únicamente aquellos datos que cumplen dicha expresión
- **directory:** es una cadena de caracteres que representa el nombre de la carpeta donde se ubican los datos a procesar

La última sección está relacionada con el proceso de notificación. En la actualidad se puede notificar al usuario mediante un correo electrónico que contendrá la respuesta a su petición.

- **email**: correo electrónico al cual notificar el resultado del servicio
- **other**: otro tipo de mecanismo (sin uso actualmente)

Estándar de entrada: xsd

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:element name="input">
      <xs:complexType>
         <xs:sequence>
            <!-- Operation -->
            <xs:choice>
               <xs:element name="downloadDB">
                  <xs:complexType>
                     <xs:all>
                        <xs:element name="data_type" type="xs:string"/>
                        <xs:element name="email" type="xs:string"/>
                     </xs:all>
                  </xs:complexType>
               </xs:element>
               <xs:element name="map-reduce">
                  <xs:complexType>
                     <xs:all>
                        <xs:element name="data_type" type="xs:string"/>
                        <xs:element name="method" type="xs:string"/>
                        <xs:element name="num_retries" type="xs:integer"/>
                        <xs:element name="do_reduce" type="xs:boolean"/>
                        <xs:element minOccurs="0" maxOccurs="1" name="ref_seq" type="xs:string"/>
                     </xs:all>
                  </xs:complexType>
               </xs:element>
               <xs:element name="evaluate">
                  <xs:complexType>
                     <xs:all>
                        <xs:element name="data_type" type="xs:string"/>
                        <xs:element name="method" type="xs:string"/>
                        <xs:element name="num_retries" type="xs:integer"/>
                        <xs:element minOccurs="0" maxOccurs="1" name="models" type="xs:string"/>
                        <xs:element minOccurs="0" maxOccurs="1" name="num_bootstraps"
                        type="xs:integer"/>
                        <xs:element minOccurs="0" maxOccurs="1" name="bs_method" type="xs:string"/>
                        <xs:element minOccurs="0" maxOccurs="1" name="cons_method" type="xs:string"/>
                     </xs:all>
                  </xs:complexType>
               </xs:element>
               <xs:element name="consensus">
                  <xs:complexType>
                     <xs:all>
                        <xs:element name="method" type="xs:string"/>
                        <xs:element name="after_bootstraps" type="xs:boolean"/>
                        <xs:element name="num_retries" type="xs:integer"/>
                     </xs:all>
                  </xs:complexType>
               </xs:element>
               <xs:element name="supertree">
                  <xs:complexType>
                     <xs:all>
                        <xs:element name="data_type" type="xs:string"/>
                        <xs:element name="method" type="xs:string"/>
                        <xs:element name="num_retries" type="xs:integer"/>
                        <xs:element minOccurs="0" maxOccurs="1" name="profile" type="xs:string"/>
                     </xs:all>
                  </xs:complexType>
               </xs:element>
            </xs:choice>
            <!-- Input Data -->
            <xs:element name="data" minOccurs="0" maxOccurs="1">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element name="URI" type="xs:string"/>
                     <xs:element minOccurs="0" maxOccurs="1" name="id_file" type="xs:string"/>
                     <xs:element minOccurs="0" maxOccurs="1" name="directory" type="xs:string"/>
                  </xs:sequence>
               </xs:complexType>
            </xs:element>
            <!-- Notification -->
            <xs:element name="notification">
               <xs:complexType>
                  <xs:choice>
                     <xs:element name="email" type="xs:string"/>
                     <xs:element name="other" type="xs:string"/>
                  </xs:choice>
               </xs:complexType>
            </xs:element>
         </xs:sequence>
      </xs:complexType> </xs:element> </xs:schema>
```

Estándar de salida: descripción

La primera sección describe el resultado del servicio solicitado. El campo status indica si la petición se realizo con éxito. La sección data describe los datos de salida producidos por el servicio. Si el campo no aparece indica que se ha producido un error y la sección *fault* describirá dicho error.

- **uri**: es una cadena de caracteres que representa el identificador único que permite el acceso al recurso que contiene/corresponde a los datos de entrada
- **id_file**: es una cadena de caracteres que representa una expresión regular para seleccionar únicamente aquellos datos que cumplen dicha expresión
- **directory:** es una cadena de caracteres que representa el nombre de la carpeta donde se ubican los datos a procesar

La segunda sección, si existe, indica que se ha producido un error durante la realización del servicio y da información del error que se ha producido. Emplea un formato similar a la entidad *fault* que describe los errores en servicios Web usada por el protocolo SOAP.

- **faultcode**: es un entero que índica el código de error producido
- **faultstring**: es una cadena de caracteres que representa el error producido
- **faultdetail**: es una cadena de caracteres que detalla el error que se ha producido

Estándar de salida: xsd

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:element name="output">
      <xs:complexType>
         <xs:sequence>
            <!-- Output Result -->
            <xs:element name="result">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element name="status"/>
                     <!-- Output Data -->
                     <xs:element name="data" minOccurs="0" maxOccurs="1">
                        <xs:complexType>
                           <xs:sequence>
                              <xs:element name="URI" type="xs:string"/>
                              <xs:element minOccurs="0" maxOccurs="1" name="id_file" type="xs:string"/>
                              <xs:element minOccurs="0" maxOccurs="1" name="directory" type="xs:string"/>
                              <xs:element minOccurs="0" maxOccurs="1" name="data_type" type="xs:string"/>
                              <xs:element minOccurs="0" maxOccurs="1" name="method" type="xs:string"/>
                           </xs:sequence>
                        </xs:complexType>
                     </xs:element>
                  </xs:sequence>
               </xs:complexType>
            </xs:element>
            <!-- Fault-->
            <xs:element minOccurs="0" maxOccurs="1" name="fault">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element name="faultcode"/>
                     <xs:element name="faultstring"/>
                     <xs:element name="faultdetail"/>
                  </xs:sequence>
               </xs:complexType>
            </xs:element>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

Anexo III. Implementación

1. Mapa de errores

Se ha creado tres categorías principales de errores que corresponden con los errores internos del servicio, errores relacionados con el almacenamiento y errores relacionados con el formato de entrada y salida de datos.

InterfaceException: corresponde con las excepciones generadas tanto en la creación de ficheros que cumplan los estándares de intercambio, como en su análisis y validación (Cód. 100). Esta excepción abarca las siguientes posibles excepciones:

- **IOException** (Cód. 101)
- **ParserConfigurationException** (Cód. 102)
- **TransformeException** (Cód. 103)
- **SAXEception** (Cód. 104)
- **URISyntaxExcpetion**: excepción causada debido a que la cadena de caracteres que representa la URI no es válida. (Cód. 105)

El código de error para este tipo de categoría es el código 100 en adelante.

StoreException: corresponde con las excepciones relacionadas con el sistema de almacenamiento externo empleado, las cuales se pueden producir debido a que el servicio no se encuentre disponible, que la URI del recurso no sea válida, etc. (Cód. 200) Esta excepción abarca las siguientes posibles excepciones:

- **IOException** (Cód. 201)
- **AmazonServiceException**: corresponde con excepciones relacionadas con el servicio de Amazon producidos por la creación de buckets cuyo nombre ya existente, descarga de recursos cuya URI no se corresponde con una URI válida de Amazon, etc. (Cód. 202)
- **AmazonClientException**: corresponde con excepciones relacionadas con el cliente del servicio externo como autenticación de credenciales, etc. (Cód. 203)
- **URISyntaxExcpetion**: excepción causada debido a que la cadena de caracteres que representa la URI no es válida. (Cód. 204)

El código de error para este tipo de categoría es el código 200 en adelante.

InnerException: corresponde con las excepciones internas de la lógica de negocio del servicio, las cuales se pueden producir debido a parámetros de entrada erróneos, errores internos de los scripts en Python, número total de reintentos agotados, etc. (Cód. 300) Esta excepción abarca las siguientes posibles excepciones:

- **IOException** (Cód. 310)
- **PhyloFlowException**: corresponde con las excepciones relacionadas directamente con el sistema de inferencia filogenética existente (Cód. 320). Esta excepción abarca:
    - **IOException** (Cód. 321)
    - **InterruptedExeception**: excepción producida por una interrupción sobre el proceso generado para ejecutar el script correspondiente. (Cód. 322)

- **Salida de error**: el servicio captura la salida de error el proceso generado para ejecutar el script correspondiente y en caso de producirse uno, genera una excepción de tipo **PhyloFlowException** con la información de la salida de error. (Cód. 323)

El código de error para este tipo de categoría es el código 300 en adelante.

Anexo IV. Despliegue y evaluación

1. Evaluación del sistema

Una vez desplegado el sistema sobre una instancia micro de *Amazon EC2* se han realizado distintas pruebas para validar el sistema y evaluar la carga de trabajo que puede soportar dicha instancia. Las instancias micro de *Amazon EC2* cuentan con una única CPU, por tanto, paralelizar en diferentes procesos los trabajos a realizar no supone ningún beneficio. Además cuenta con tan sólo 0.613 GB de memoria.

Para dos o menos conjuntos de secuencias de ADN mitocondrial humano el sistema sobre la instancia micro ha sido capaz de obtener los resultados Sin embargo, tres conjuntos de secuencias supone un coste computacional excesivo para este tipo de instancias.

A continuación en la Tabla 5 se muestran en los resultados de las pruebas realizadas para secuencias de ADN mitocondrial.

Tiempo total				
Sets	*Fetch Sequences*	*Sequences Processing*	*Sequences Evaluation*	Total
2	≈7 s.	≈38 min.	≈41 min.	≈80 min.
2	≈7 s.	≈120 min.	≈51 min.	≈172 min.
3	≈7 s.	ERROR		

Tabla 5. Tiempos totales de las pruebas para ADN mitocondrial sobre una instancia micro de *Amazon EC2*

Como se puede comprobar los tiempos varían significativamente a pesar de emplear el mismo número de secuencias. En este caso, se debe al consumo de memoria que en la segunda prueba fue mayor y el coste temporal aumento significativamente en la segunda fase del procesamiento. En las Figura 22 y Figura 23 se distribuye el tiempo total en fases y se muestra la diferencia entre el tiempo de ejecución consumido por *Phyloflow* [2] y el tiempo total.

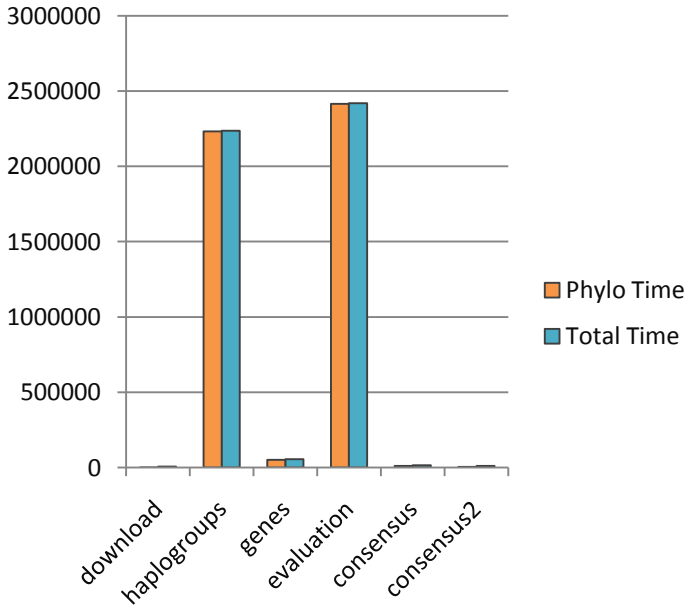


Figura 22. Gráfica de tiempos para dos conjuntos de ADN mitocondrial

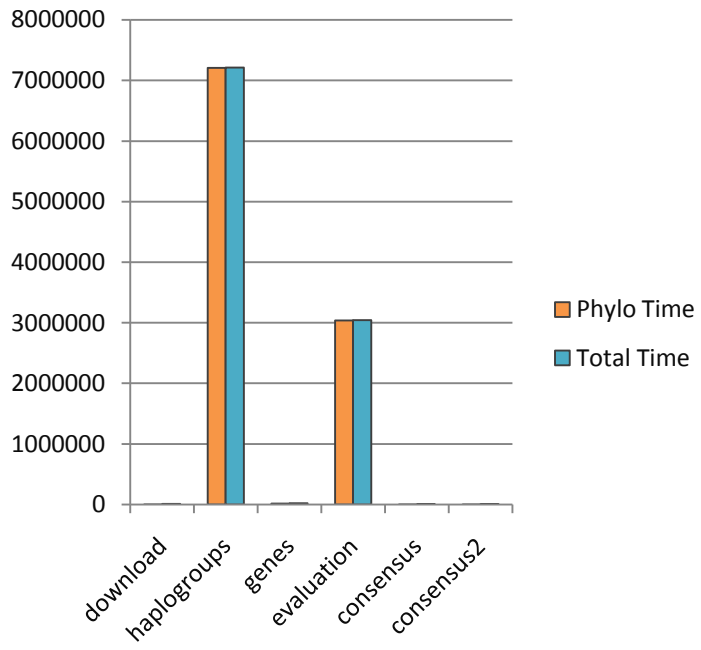


Figura 23. Gráfica de tiempos para dos conjuntos de ADN mitocondrial

Prácticamente no se puede diferenciar la sobrecarga de ofrecer el sistema como un servicio. También se puede comprobar que la fase de haplogrupos ha sido la causante de la gran diferencia que hay entre las dos pruebas realizadas. La sobrecarga en el primer caso supone un 1% y en el segundo caso aproximadamente el 0% del tiempo total.

Por tanto, se han empleado conjuntos de secuencias sintéticas cuyo coste computacional de procesamiento es netamente inferior. Para conjuntos reducidos de secuencias la sobrecarga de ofrecer el sistema como un servicio supone aproximadamente el 10% del tiempo total de ejecución. Conforme se aumenta el conjunto de secuencias el tiempo crece por encima de la complejidad lineal, reduciendo el porcentaje de sobrecarga.

A continuación se muestran en los resultados de las pruebas realizadas para secuencias de ADN mitocondrial.

Tiempo total					
Sets	Reintentos máximos	*Fetch Sequences*	*Sequences Processing*	*Sequences Evaluation*	Total
2	3	≈3 s.	≈1min.	≈0,5 min.	≈1,5 min.
2	3	≈4 s.	≈3 min.	≈3 min.	≈6 min.
2	3	≈3 s.	≈1,5 min.	≈1 min.	≈2,5 min.
4	4	≈3 s.	≈9,5 min.	≈5,5 min.	≈15 min.
4	5	≈12 s.	≈18,5 min.	≈8,5 min.	≈27 min.
8	7	≈4 s.	≈4 min.	≈2 min.	≈6 min.
16*	10	≈4 s.	≈48 min.	≈25 min.	≈73 min
26*	30	≈6 s.	≈91 min.	≈42,5 min.	≈134 min
4	3	≈3 s.	ERROR		
8	6	≈4 s.	ERROR		

Tabla 6. Tiempos totales de las pruebas para secuencia sintéticas sobre una instancia micro de *Amazon EC2*

Estas pruebas han permitido detectar un defecto en el sistema *PhyloFlow* [2] con respecto a una fase en la que realiza un *backup* de los resultados. Para la realización de las pruebas se ha aumentado el número total de reintentos para que finalice el procesamiento a pesar de dicho defecto.

Las pruebas que tienen un asterisco indican que la fase que realiza el *backup* se ha omitido para evitar que se produzca la excepción. Viendo los resultados destaca la diferencia existente entre las pruebas de 8 y 4 conjuntos de secuencia. La diferencia se debe sustancialmente al tiempo que lleva la máquina en funcionamiento. En el primer caso, la máquina está recién lanzada, mientras que en el segundo caso lleva cierto tiempo en funcionamiento y tras varias peticiones parte de la memoria está siendo desempleada por el *garbage collector* de *Java* [7].

El tiempo de sobrecarga es determinante en este contexto debido a que nunca se pondrá realizar una ejecución completa por debajo de este tiempo. Es una métrica a tener en cuenta a la hora de seleccionar tanto el tipo de instancias como el número total de hilos de ejecución. En el dominio de la aplicación el tiempo de sobrecarga no es constante debido a que depende del tamaño de los conjuntos de secuencia. Conforme el tamaño de los conjuntos de secuencia aumente también lo hará la sobrecarga a causa de que los tiempos de subida y bajada al sistema de almacenamiento remoto están ligados al tamaño de los ficheros.

2. Implementación de un sistema de toma de decisiones

Para la implementación del sistema de toma de decisiones mediante aprendizaje sobre ficheros de log se han empleado los ficheros de log generados en la fase de evaluación y validación del sistema. A partir de la información recogida en concreto la referente a tipo de secuencia, número total de conjuntos de secuencias, método de procesamiento, número total de reintentos, consumo de memoria y tiempo total de ejecución se ha decidido crear perfiles de ejecución que permiten establecer sobre qué tipo de instancia se debería utilizar.

De esta manera, se ha planteado un modelo de mejora sencillo basado en el entrenamiento de un árbol de decisión sobre un conjunto de entrenamiento basado en tablas. Se ha creado un conjunto de entrenamiento por cada uno de los componentes que constituyen el flujo de trabajo del sistema *PhyloFlow* [2]. Por último, se ha escogido que tipo de instancia se adaptaría mejor para cada prueba realizada considerando los costes económicos y temporales que implicaría su elección de forma aproximada. La Tabla 7 muestra un conjunto reducido de perfiles de entrenamiento a modo ejemplo para el componente *Sequences Processing* del formato de las tablas empleadas.

Secuencia	Sets	Método	Reintentos	T. Activa	Memoria	T. Ejecución	Decisión
sintéticos	4	dactal	5	1,5	529288	492908	t1.micro
sintéticos	4	mafft	4	4	483693	384555	t1.micro
sintéticos	8	dactal	5	0	491996	MAX INT	t1.micro
sintéticos	8	dactal	7	0	492780	107018	t1.micro
sintéticos	8	mafft	7	0	509484	133428	t1.micro
sintéticos	26	dactal	30	9,5	527192	2484217	m1.small
sintéticos	26	mafft	30	10,5	532196	2983629	m1.small

Tabla 7. Conjunto de entrenamiento de ejemplo basado en tablas

En el caso de producirse un error se ha decidió establecer el tiempo de ejecución como el máximo posible para no desvirtuar el resultado. Una vez completada las tres tablas para cada uno de los componentes se ha utilizado la herramienta KNIME [25] para generar el sistema de toma de decisiones basado en el entrenamiento de un árbol de decisión.

En la herramienta KNIME [25] se crea un flujo de trabajo como el de la Figura 24 para cada conjunto de entrenamiento. Los ficheros de entrada contienen el conjunto de entrenamiento a partir del cual se crea un árbol de decisión. Se han empleado las secuencias sintéticas para generar estos árboles y las secuencias de ADN mitocondrial para establecer árboles predictivos, por último, se genera una imagen con el árbol de decisión.

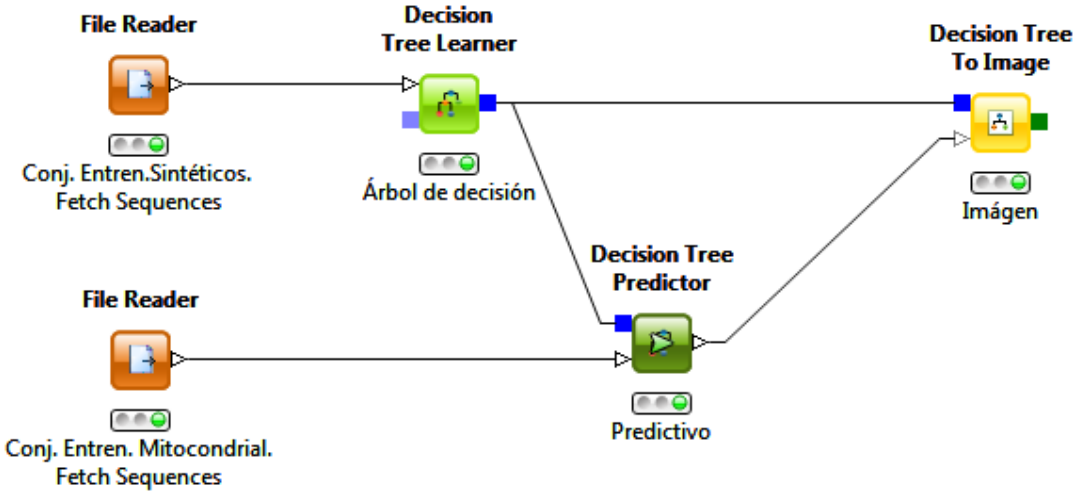


Figura 24. Flujo de trabajo para crear un árbol de decisión

A continuación, la Figura 25 muestra los resultados de los modelos derivados del entrenamiento y las pruebas realizadas con ellos para el componente *Fetch Sequences.*

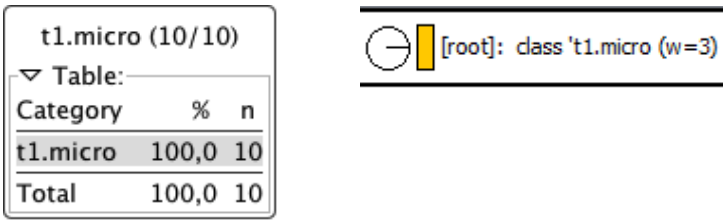


Figura 25. Árbol de decisión para el componente *Fetch Sequences*

En este caso debido a que sólo se realizaban copias de la bases de datos disponibles para todos los perfiles de entrenamiento descritos la decisión establecía emplear instancias de tipo micro.

La Figura 26 y la Figura 27 muestran los resultados de los modelos derivados del entrenamiento y las pruebas realizadas con ellos para los componentes *Sequences Processing* y *Sequences Evaluation* respectivamente. En el primer caso la variable de mayor peso es el número de reintentos debido al defecto comentado anteriormente producido en la fase de *backup* que obliga a aumentar el número de reintentos a la par que se aumenta el conjunto de secuencias.

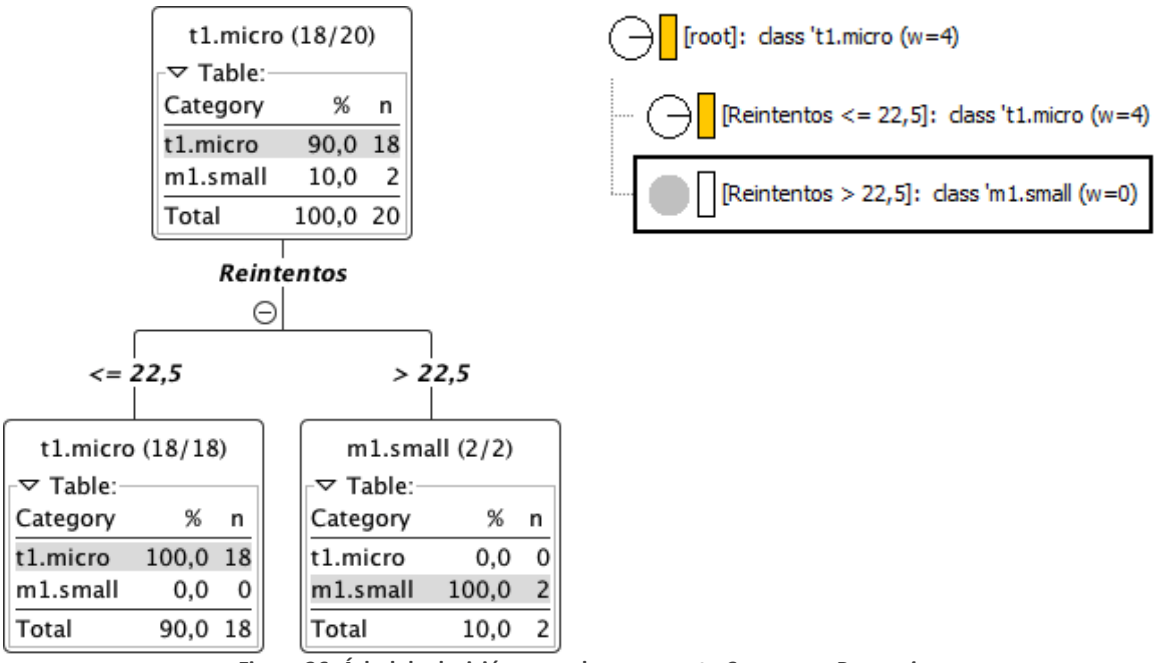


Figura 26. Árbol de decisión para el componente *Sequences Processig*

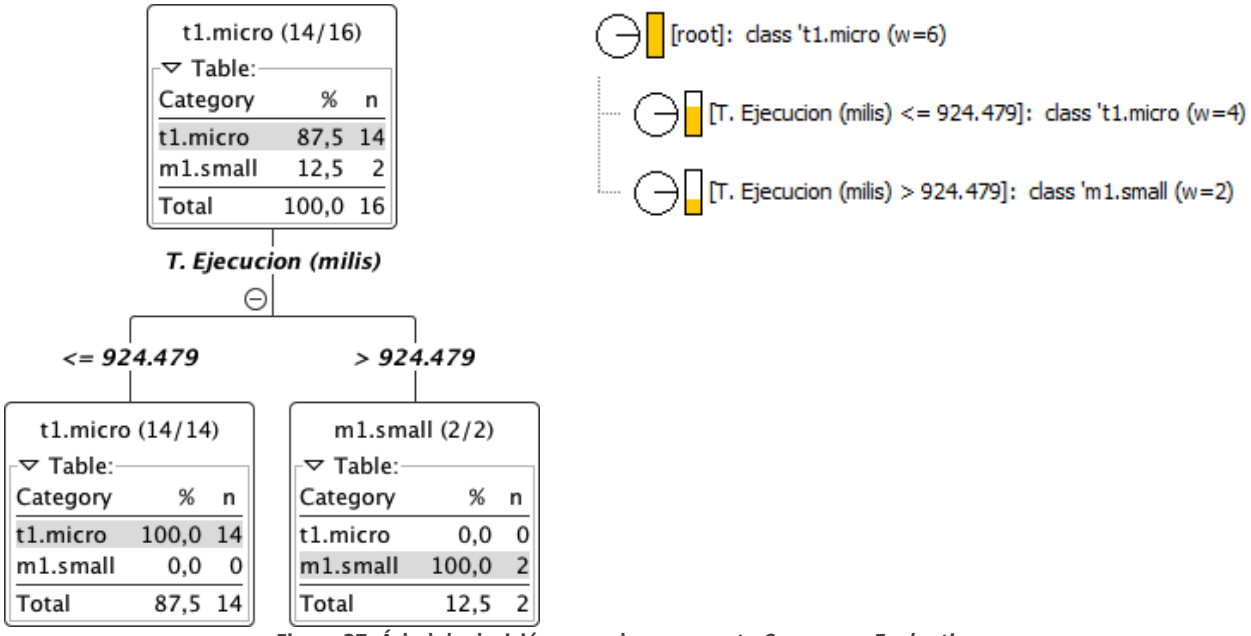


Figura 27. Árbol de decisión para el componente *Sequences Evaluation*

Anexo V. Manual de usuario

En esta sección se detallan los procedimientos que deben realizar un usuario para utilizar correctamente el sistema.

1. Identificación del servicio

Todos los servicios Web ofrecidos son de tipo *REST*, por tanto, el usuario debe emplear una herramienta que le permita realizar peticiones *REST* al servicio que desee, por ejemplo, la herramienta *REST Advanced Client* para el navegador *Google Chrome* que ofrece una interfaz gráfica para realizar las peticiones de forma sencilla. Otra alternativa sería implementar un sistema propio utilizando cualquier lenguaje de programación que soporte peticiones *REST*.

El primer paso que debe realizar el usuario es identificar el servicio que desea consumir. Una vez identificado debe consultar la *API* de dicho servicio para conocer el protocolo de comunicación establecido. Ésta informa de la ubicación exacta del servicio, el método *HTTP* y los parámetros de entrada. Además se ha de tener en cuenta el estándar de comunicación establecido.

2. Servicios individuales

Por ejemplo, para realizar una petición al servicio *Fetch Sequences* una vez consultada la *API* se conoce que la *URL* del servicio que es la dirección relativa /phyloflow/fetchsequences/download, el método *HTTP* es *POST* y los parámetros de entrada para dicha operación son el tipo de secuencia y el correo electrónico. Por tanto, la petición a enviar en el *payload* del mensaje tendrá la siguiente estructura:

```
<input>
   <downloadDB>
      <data_type>hmtDNA</data_type>
      <email>usuario@email.com</email>
   </downloadDB>
   <notification>
      <email>usuario@email.com</email>
   </notification>
</input>
```

A continuación la Figura 28 se muestra un ejemplo empleando la herramienta *REST Advanced Client* comentada anteriormente. En esta petición se solicita obtener la base de datos de secuencias sintéticas. La Figura 28 muestra el procedimiento a realizar.

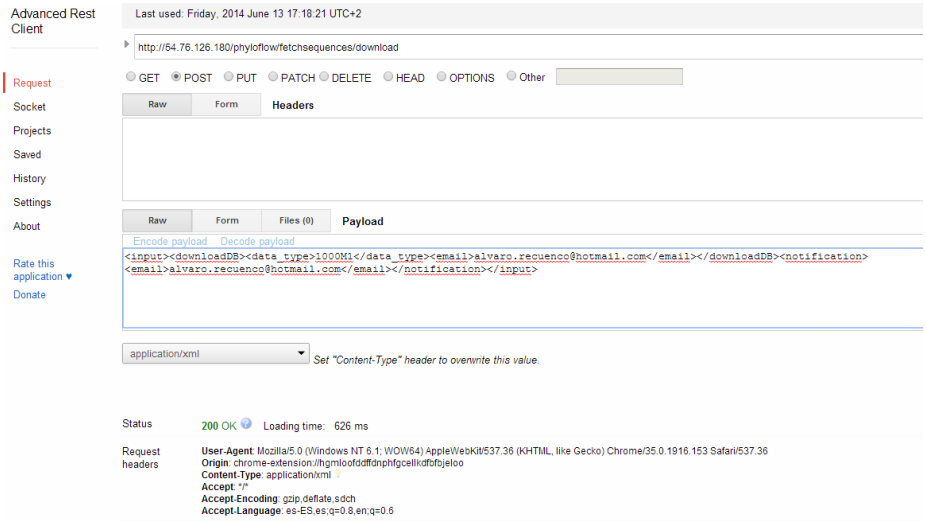


Figura 28. Petición POST al servicio *Fetch Sequences* con la herramienta *REST Advanced Client*

Si el servicio responde con un estado 200 significa que la petición ha tenido éxito. Si la petición fuera sintácticamente incorrecta el servicio responderá con el estado 400 que equivale a petición incorrecta. En cambio si se produjese un error interno durante el análisis de la petición el servicio responderá con un estado 500 que equivale a error interno del servidor.

Si la petición ha tenido éxito el servidor comenzará a realizar el procesamiento solicitad, cuando finalice el servicio notificará al correo electrónico los resultados obtenidos. La Figura 29 muestra el mensaje enviado al correo electrónico con el adjunto que contiene los resultados.

Figura 29. Mensaje de respuesta del servicio *Fetch Sequences*

El fichero adjunto con la respuesta contiene la URI que da acceso a los resultados comprimidos en el sistema de almacenamiento de *Amazon S3*. La URI contiene caracteres de escape, por tanto, se deben modificar antes de introducirla en el navegador. La respuesta a la petición anterior es la siguiente:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<output>
   <result>
      <status>Status: OK</status>
       <data>
          <URI>https://buckete3e1db59-fae0-4145-8da4-b1492f63d502.s3-eu-west-
1.amazonaws.com/sets.zip?Expires=1403726111&amp;amp;AWSAccessKeyId=AKIAI2NWUPUT5KCSJ7EQ&amp;amp;S
ignature=Asa%2BPxyxrge0s57oE2uCOkQbOkQ%3D
          </URI>
      </data>
   </result>
</output>
```

Si se hubiera producido un error durante el procesamiento la respuesta tendría un bloque *fault* que contendría el error producido y una breve explicación del mismo. De esta forma, el usuario podría detectar si ha introducido algún parámetro erróneo o si el servicio está momentáneamente fuera de servicio.

La única diferencia respecto al resto de servicios es que este servicio no requiere una entrada de datos a procesar. Las peticiones del resto de servicios como indica la *API* de cada uno de ellos debe tener un campo para los datos de entrada. Basta con utilizar el campo *data* del fichero de respuesta en la siguiente petición, siempre y cuando el recurso siga estando disponible.

A continuación, se muestra como se ha de realizar una petición al componente *Sequences Processing* para realizar un procesamiento biológico sobre las secuencias sintéticas obtenidas anteriormente. En este caso se ha seleccionado el método dactal con un número de reintentos igual a 3. La petición tiene la siguiente estructura:

```
<input>
   <map-reduce>
      <data_type>dna</data_type>
      <method>dactal</method>
      <num_retries>3</num_retries>
      <do_reduce>false</do_reduce>
      <ref_seq>backup</ref_seq>
   </map-reduce>
   <data>
      <URI>https://buckete3e1db59-fae0-4145-8da4-b1492f63d502.s3-eu-west-
1.amazonaws.com/sets.zip?Expires=1403726111&amp;amp;AWSAccessKeyId=AKIAI2NWUPUT5KCSJ7EQ&amp;amp;S
ignature=Asa%2BPxyxrge0s57oE2uCOkQbOkQ%3D
     </URI>
   </data>
   <notification>
      <email>usuario@email.com</email>
   </notification>
</input>
```

Nuevamente se ha de esperar a que finalice el procesamiento por completo para obtener los resultados. Para interactuar con el resto de componentes que constituyen el flujo de trabajo se realiza de la misma forma. La única variación se produce en el campo que corresponda a la operación y parámetros del servicio que se desee consumir.

3. Servicio para un flujo de trabajo completo

Si se desea realizar un flujo de trabajo completo sin la necesidad de integrar manualmente cada una de las fases el usuario puede consumir el servicio Web publicado por el componente *Workflow System*. Al igual que para los servicio anteriores el primer paso es consultar la *API* del servicio para conocer el protocolo de comunicación. Este servicio realiza un flujo de trabajo predeterminado y, por tanto, no es necesario que el usuario añada ningún parámetro a excepción del correo electrónico. Consecuentemente, si se desea realizar un flujo de trabajo completo simplemente se debe enviar un parámetro etiquetado con *email* que representa el correo electrónico del usuario. Por tanto, la petición sería la siguiente:

```
<email>usuario@email.com</email>
```

Exactamente igual que para los servicios anteriores si la petición tiene éxito responderá con el estado 200, si la petición es sintácticamente incorrecta lo hará con estado 400 y, por último si se produce un error interno responderá con el estado 500.

El componente *System Workflow* se encargará de realizar las peticiones ordenadamente y capturar los resultados parciales. El último nodo del flujo de trabajo completo será el encargado de notificar al usuario los resultados. El mensaje con los resultados será exactamente igual que el mostrado en la Figura 29.