Facultad de Ciencias y Tecnologías Computacionales

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas.

Título: FrontEnd de la plataforma de alto rendimiento para el análisis filogenético: PhylUCI.

Autor: Luis Miguel Puris Hernández

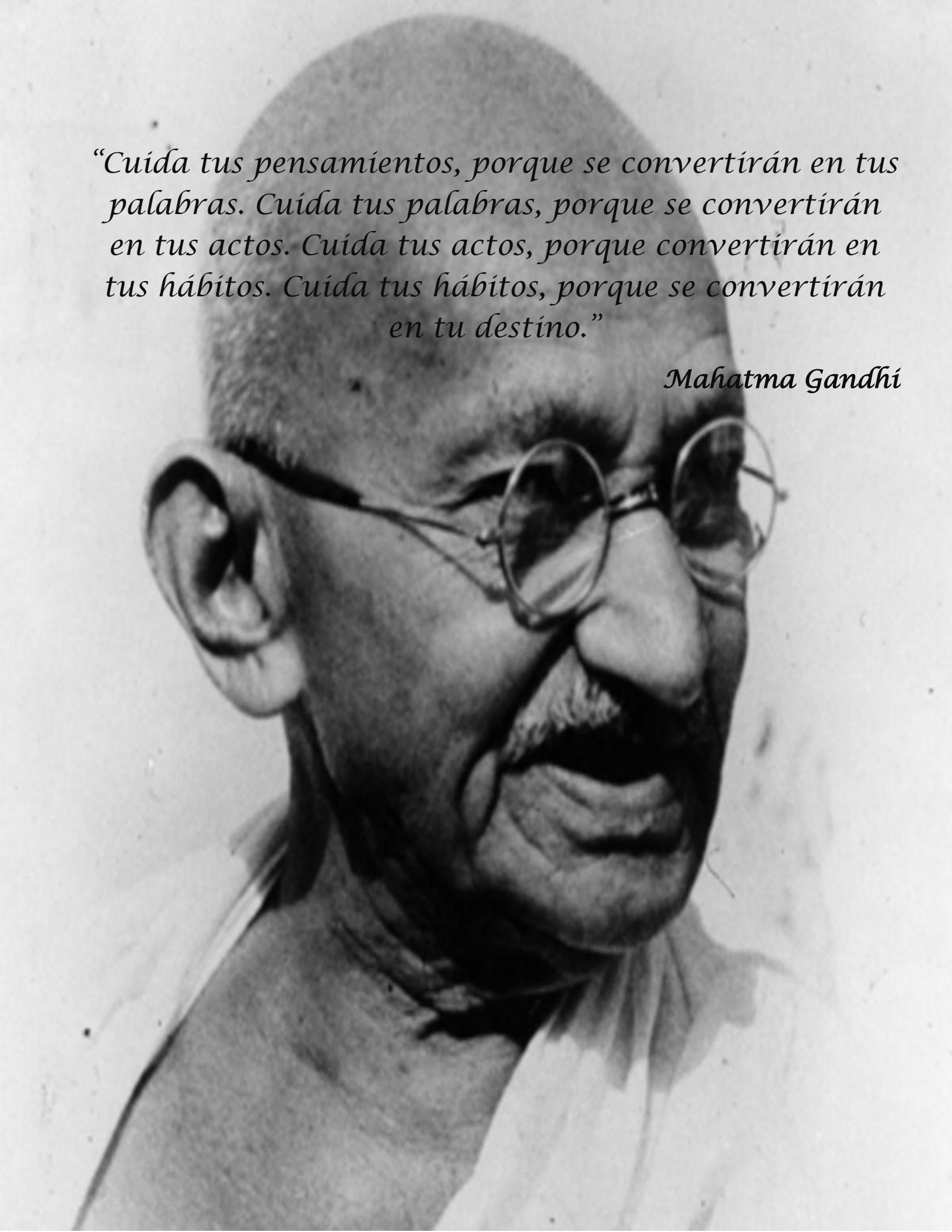
Tutor(es): MSc. Mario Pupo Meriño

Lic. Luis Antonio García González

Co-tutor: Ing. Enier Alarcón Barban

La Habana, junio de 2017

"Año 59 de la Revolución"



"Cuida tus pensamientos, porque se convertirán en tus palabras. Cuida tus palabras, porque se convertirán en tus actos. Cuida tus actos, porque convertirán en tus hábitos. Cuida tus hábitos, porque se convertirán en tu destino."

Mahatma Gandhi

FrontEnd de la plataforma de alto rendimiento para el análisis filogenético: PhylUCI.

Declaración de Autoría

Declaro ser autor del presente trabajo de diplomas y reconocemos a la Universidad de las Ciencias Informáticas (UCI) los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste se firma la presente a los ____ días del mes de _____ del año _____.

Autor:

_____
Luis Miguel Puris Hernández

Tutores:

_____ _____
Lic. Luis Antonio García González MSc. Mario Pupo Meriño

Co-tutor:

_____
Ing. Enier Alarcón Barbán

FrontEnd de la plataforma de alto rendimiento para el análisis filogenético: PhylUCI.

Datos de Autor

Tutor: Lic. Luis Antonio García

Universidad de las Ciencias Informáticas, La Habana, Cuba

Especialidad de graduación:

Años de graduado:

Área de investigación:

Correo Electrónico: luisgarcia@uci.cu

Tutor: MSc. Mario Pupo Meriño

Universidad de las Ciencias Informáticas, La Habana, Cuba

Especialidad de graduación:

Años de graduado:

Área de investigación: mpupom@uci.cu

Correo Electrónico:

Co-Tutor: Enier Alarcón Barban

Universidad de las Ciencias Informáticas, La Habana, Cuba

Especialidad de graduación: Ingeniería Informática

Años de graduado: 3

Área de investigación:

Correo Electrónico: barban@uci.cu

Autor: Luis Miguel Puris Hernández

Universidad de las Ciencias Informáticas, La Habana, Cuba

Correo Electrónico: lmpuris@estudiantes.uci.cu

Agradecimientos

Agradezco a mi madre, por ser esa mujer que entre regaños, besos y abrazos siempre desea lo mejor para mi, por estar a mi lado siempre y nunca dejarme caer y sobre todo por ser mi madre.

A mi padre, por ser el hombre que es y por siempre estar presente en las malas, por ser mi amigo y mi modelo a seguir, por requerirme y hablarme fuerte en los momentos que ha hecho falta y sobre todas las cosas por ser mi padre.

A mi hermana, por quererme tanto, por estar a mi lado y nunca darse por vencida conmigo cuando era un chiquillo malcriado y por siempre tener un momento cuando más lo he necesitado.

A mi hermano, por ser esa persona a la cual he querido parecerme tanto, por ser mi amigo, mi ídolo como siempre le he dicho.

A mi hijo por ser en estos momentos el impulso más grande para continuar y ser cada día mejor. Te amo

A mi Claudia, agradecerte por estar continuamente a mi lado, por no darte por vencida nunca conmigo, por siempre recibirme con una sonrisa, por muy grande que sea el problema, y sobre todas las cosas agradecerte por esa bendición tan grande que me regalaste; nuestro hijo.

A mi abuelo Miguel, por ser ese hombre al cual respetaba tanto, por sus consejos.

A mi abuela Blanca, aunque nunca la conocí, quiero agradecerle por haber hecho posible que yo tuviera una madre como la que tengo.

A mi abuela Nereida, por ser esa vieja tan cariñosa y atenta conmigo, por su amor y más te quiero mi "necha".

A mi abuelo Yiyo que, aunque nunca va a leer estas palabras y mucho menos entender su significado, quiero dejar plasmado mis agradecimientos hacia él, por ser ese abuelo que se desvivía por su nieto, por cuidarme tanto y aunque halla momentos que ya no me conozca, yo nunca lo voy a olvidar.

A mis sobrinas, Kamila y Carolina por ser las niñas más linda del mundo, por ser la alegría de la familia. Las quiero enormemente.

A mi tata Mariam, que, aunque es mi sobrina dice que es mi hermana, por quererme tanto siempre la voy a llevar en el corazón y nunca va a dejar de ser mi niña.

A Yaneski y Reynol por siempre desde niño quererme como uno más de su familia y acogerme en tantas vacaciones en su casa.

A mi tía nórdica Ivonne y Jesús, agradecerle por ser como son, especiales, por ser tan amables, cariñosos y atentos. Por ser unas personas intachables en cuanto amor se trata, siempre les voy a estar eternamente agradecidos.

A Diana de León, por ser esa persona que siempre se preocupó por mí, sin tener la más mínima responsabilidad de mi persona. Por ser esa mujer que siempre me tuvo presente y una de las grandes autoras de que yo me esté graduando, por todo y más mil gracias.

A mi negrita Suinny, por ser como es, nunca cambies. Por ser mi amiga, mi hermana, por estar todos estos años de carrera a mi lado.

A el negro Magdiel, por ser mi hermano y siempre aparecer cuando más falta hace, por ser mi amigo.

Al Lijo por siempre tender la mano sin importar nada ni nadie, por ser mi amigo te quiero hermanaso.

A mis tutores Mario y Luis por estar todo este tiempo cargando conmigo, por brindarme la oportunidad de unirme a ustedes, por sus consejos, correcciones en todo momento sin importar hora. Muchas gracias y eternamente agradecido.

A mi Co-tutor Enier por su ayuda en todo el proceso de tesis, por estar sus consejos, correcciones y su tiempo. Muchas gracias.

Dedicatoria

A mi madre por ser la luz y guía de mi vida.

A mi padre por ser la persona que más quiero hacer sentir orgulloso.

A mis hermanos, por ser los mejores del mundo.

A mi mujer y madre de mi hijo, por ser mi amiga y compañera.

A mi hijo que recién empieza sus pasos por esta vida.

A todas aquellas persona que durante toda mi carrera aportaron su granito para que cumpliera mi sueño.

FrontEnd de la plataforma de alto rendimiento para el análisis filogenético: PhylUCI.

Resumen

El análisis filogenético es uno de los métodos bioinformático más utilizados para el estudio de entidades virales. En la Universidad de las Ciencias Informáticas (UCI) se desarrolla la plataforma para análisis filogenético PhylUCI, la cual brinda un marco favorable a los estudios filogenéticos realizados en el país. Actualmente la plataforma cuenta con un solo módulo el BackEnd, este módulo no dispone de una interfaz que amenice el proceso a los investigadores, pues todas las herramientas se ejecutan a través de comandos. Este trabajo tiene como objetivo desarrollar una interfaz web para la plataforma, además de un visualizador y editor de árboles. Se analizó, diseñó e implementó un FrontEnd, módulo que resuelve los problemas de selección, configuración y visualización de los resultados arrojados por las herramientas que ofrece la plataforma, así como un editor de árboles filogenéticos. Estos módulos se comunicarán a través de los servicios RESTFUL. Para guiar la investigación se escogió Open Up como metodología de desarrollo, así como el patrón arquitectónico Modelo Vista Plantilla. Se trazó una estrategia de pruebas, las cuales arrojaron 24 no conformidades en 3 iteraciones, quedando todas resueltas en la última iteración, obteniendo un FrontEnd de calidad.

Palabras claves: BackEnd, FrontEnd, Módulo, Plataforma.

FrontEnd de la plataforma de alto rendimiento para el análisis filogenético: PhylUCI.

Abstract

The phylogenetic analysis is one of the study of viral entities. The Computer Science University (UCI) a platform for the phylogenetic analysis is being developed PhylUCI, which provides a favorable framework to the phylogenetic studies carried out in our country. Nowadays this platform has only one module, the BackEnd, but this module does nat have an interface that make the process enjoyable or more interesting to the researchers, because all the tools are played through commands. The objective of this work is to develop a web interface for the platform, and also a visual display unit and an editor trees diagram. To this purpose it was analyzed designed and implemented a FrontEnd, module that salves the selection of problems, configuration and visualization of the yielded results by the tools provided by the platform, as well as a phylogenetic tree editor. These modules get connected through RESTFUL service. To guide the investigation, it was chosen Open Up as methodology of development as well as Model Sight Template, architectonic standards that will lead the development of the system. The test strategy drawn yielded 24 not conformities in three iterations, but all of then iteration with a result of a FrontEnd of quality.

Keywords: BackEnd, FrontEnd, Module, Platform**.**

Índice

Índice de Figuras

Índice de Tablas

FrontEnd de la plataforma de alto rendimiento para el análisis filogenético: PhylUCI.

INTRODUCCIÓN

Las enfermedades infecciosas ocupan en la actualidad un lugar importante en el panorama de la salud pública mundial, jugando un papel fundamental las de tipo viral, los avances científico técnicos ocurridos en la medicina a finales del siglo pasado, sugerían que estos virus podrían ser controlados. Estudios recientes demuestran que estas enfermedades se incrementan y amplían de forma considerable y alarmante en todo el mundo.

Cuba por ser un país ubicado en el trópico es escenario perfecto para la propagación de estas enfermedades entre las cuales cabe resaltar el dengue, la influenza, VIH y el Zika. El país implementa un sistema de salud preventivo con el fin de minimizar los efectos a nivel social y el costo que involucraría una propagación a gran escala de alguna de estas enfermedades.

El gobierno cubano, interesado en resultados que contrarresten el desarrollo y propagación de estas enfermedades, ha creado diferentes centros encargados de la investigación y vigilancia de estas entidades, así como para el desarrollo y producción de medicamentos y vacunas. De esa iniciativa nace el Instituto de Medicina Tropical “Pedro Kouri” (IPK), la institución líder en Cuba en campos como la Microbiología, Parasitología, Medicina Tropical, Infectología Clínica y Epidemiologia. Entre los departamentos del centro, el Departamento de Virología desarrolla un número ascendente de estudios sobre el origen y evolución de entidades virales relacionados con epidemias ocurridas en Cuba. Cuenta con el equipamiento necesario para secuenciar el material genético de estas entidades virales, presentes en gran número de pacientes, por lo que se está produciendo gran cúmulo de información.

El análisis filogenético es uno de los métodos bioinformáticos más usados para el estudio del origen y evolución de entidades virales. La filogenética es la parte de la biología evolutiva que se ocupa de determinar las relaciones evolutivas entre diferentes grupos de organismos.

Los arboles filogenéticos, son la base de la clasificación filogenética. A partir de una secuencia de interés, un análisis filogenético pasa a través de etapas sucesivas que incluyen la identificación de secuencias homólogas, el alineamiento múltiple, la reconstrucción filogenética y representación gráfica de los árboles inferidos (Dereeper, y otros, 2008).Estos pasos implican un alto consumo de recursos computacionales en función del número y longitud de las secuencias a analizar. El cómputo de cada vez más grandes y más precisos árboles filogenéticos representa uno de los grandes desafíos en la Bioinformática.

Debido al alto número de secuencias a analizar y la gran longitud de estas, tomando en consideración que los avances en la Biología Molecular permiten secuenciar todo el genoma viral, en los estudios filogenéticos que se realizan en la actualidad se requiere necesariamente el uso de la Computación de Alto Rendimiento

(High Performance Computing –HPC por sus siglas en inglés) Para facilitar el acceso a los recursos filogenéticos a nivel internacional, se han desarrollado servidores web especializados que hacen uso de la tecnología HPC, a los que se puede acceder mediante Internet . Los servidores web para análisis filogenéticos van desde los que ejecutan aplicaciones individuales hasta los que logran una integración de múltiples herramientas. En general, al menos los servidores de acceso público son escasos y requieren una conexión de banda ancha, además se caracterizan por no disponer de una interfaz sencilla para los investigadores expertos y mucho menos para aquellos que comienzan. De igual modo, no todas las plataformas existentes cuentan con un visualizador de árboles que permita editar las características que desee el investigador. Otra característica negativa es que muchas plataformas acceden a herramientas externas y otras utilizan aplicaciones de terceros aumentando así los tiempos de ejecución y respuestas de dichos análisis.

Actualmente existen centros de investigación como el propio IPK y otros ubicados en universidades cubanas, que no cuentan con una infraestructura que les permita realizar una ejecución distribuida y paralela de las herramientas necesarias para los análisis filogenéticos, además de que no cuentan con un ancho de banda suficiente que les permita acceder a los recursos filogenéticos existentes en la red.

A través del proyecto “High Performance Computing Software for Bioinformatics Applications” promovido por la Universidad de las Ciencias Informáticas (UCI), la Universidad Católica de Lovaina (K.U. Leuven) y el IPK; financiado por VLIR-UOS; se realizó la compra de tecnología de alto rendimiento, lo cual crea un marco favorable para la realización de estudios filogenéticos a gran escala. La UCI en conjunto con el IPK iniciaron el desarrollo de una plataforma web para el análisis filogenético dirigida a brindar soporte a las investigaciones realizadas, con el fin de agilizar dichos procesos, aprovechando las nuevas capacidades tecnológicas. La plataforma debe ofrecer un conjunto de herramientas necesarias para los procesos asociados al análisis filogenético. La mayoría de estas herramientas están disponibles en versiones que pueden ser ejecutadas en un ambiente distribuido y todas aprovechan el paralelismo.

La plataforma estará conformada por dos módulos fundamentales. El FrontEnd, que será la interfaz donde los investigadores podrán ejecutar y ver el estado de sus tareas. Además, se encargará de la visualización y edición de árboles filogenéticos. El BackEnd, módulo que se hará cargo de la cola de tareas en el clúster y la administración de usuarios. Estos módulos se comunicarán utilizando un lenguaje de descripción de tareas (JDL).

El BackEnd por si solo es capaz de realizar un completo y detallado análisis filogenético por la variedad de herramientas de las cual dispone, pero no cuenta con una interfaz gráfica de usuario que amenice el proceso hacia un investigador, pues todas sus herramientas son ejecutadas a través de comandos por lo

que se requeriría de un avanzado conocimiento bioinformático para realizar un análisis filogenético, y tampoco sería posible llegar a modelar sobre los árboles resultantes.

Por problemas de rendimiento este módulo no puede estar de cara a internet y mucho menos contar con una propia interfaz de usuario, pues para esto se necesitarían un conjunto de aplicaciones que garantizaran la seguridad del sistema, además de un servidor web. Todo esto supondría una disminución en el rendimiento del clúster pues se estarían ejecutando procesos que no están vinculados directamente con las herramientas utilizadas para análisis filogenético.

Debido a la necesidad de dar solución a lo anteriormente planteado se identificó como **problema de investigación** cómo facilitar la realización de análisis filogenéticos, visualización y edición de los resultados; a los investigadores en la plataforma PhylUCI.

Donde el **objeto de estudio** está dirigido a la interfaz gráfica de usuario, enmarcado en el **campo de acción** interfaces de usuarios en plataformas web para análisis filogenético.

Para dar solución al problema de investigación identificado se define como **objetivo general:** desarrollar una interfaz web, además de un visualizador y editor de árboles, para la plataforma de análisis filogenético PhylUCI.

Para darle cumplimiento al objetivo general se definen los siguientes **objetivos específicos:**

1. Desarrollar una vista para cada una de las herramientas para cálculo filogenético que formen parte de los flujos de trabajo en filogenia.
2. Desarrollar una vista para cada uno de los flujos de trabajo existentes en análisis filogenético definido para la plataforma.
3. Desarrollar un visualizador y editor de árboles filogenéticos.
4. Desarrollar una vista para el estado de las tareas.
5. Desarrollar un sistema de autenticación mediante Tokens.

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

Teóricos:

- **Analítico-Sintético**: Es utilizado para descomponer los diversos procesos que se utilizan en la creación de un análisis filogenético en elementos por separado y profundizar en el estudio de cada uno de ellos de forma independiente.
- **Método de modelado**: Para representar los diagramas correspondientes a la etapa de análisis, diseño e implementación de la solución.

Empíricos:

- **Análisis documental**: Para la revisión de la literatura, que permitió consultar la información necesaria en el proceso de investigación sobre plataformas de análisis filogenético.
- **Análisis estático**: Permitió observar de otras plataformas con el objetivo de clasificar las dificultades existentes y las funcionalidades a implementar en el sistema.

Posibles Resultados: Obtención del FrontEnd para la plataforma web de análisis filogenético que contenga una vista para cada una de las herramientas disponibles y permita la edición de árboles filogenéticos.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

En el presente capítulo se exponen los conceptos asociados a filogenética, alineamiento de secuencias, filogenia, árboles filogenéticos e inferencia filogenética. Se hace una revisión de las soluciones existentes para un análisis filogenético. Se realiza un estudio para la selección de herramientas y tecnologías a utilizar para el desarrollo del trabajo de diploma.

1.1 Conceptos asociados al dominio del problema

Se abordan los conceptos principales en el área de la filogenética para una mejor comprensión de los resultados.

1.1.1 Filogenética

La filogenética es el estudio científico de la filogenia. La filogenia pertenece a la historia evolutiva de un grupo taxonómico de organismos. Así, la filogenética se ocupa principalmente de las relaciones de un organismo con otros organismos de acuerdo con similitudes y diferencias evolutivas. La filogenética, por lo tanto, es una parte de la sistemática biológica, que tiene un alcance más amplio. Esta última implica no sólo la filogenética de los organismos, sino también la identificación y clasificación de los organismos. También se relaciona con la taxonomía, que es una rama de la ciencia interesada también en encontrar, describir, clasificar y nombrar organismos, incluyendo el estudio de las relaciones entre taxa y los principios subyacentes a tal clasificación. La filogenética proporciona información a la taxonomía cuando se trata de la clasificación e identificación de organismos (Lemey, Salemi, & Vandamme, 2009).

1.1.2 Alineamiento de secuencias

Una alineación de secuencia es un conjunto de residuos correspondientes entre una colección de secuencias de nucleótidos o aminoácidos. Las secuencias pueden ser secuencias codificantes de proteínas o ARN o secuencias de nucleótidos no codificantes, tales como intrones o espaciadores. Se supone que las secuencias involucradas en una alineación son homólogas; es decir, derivado de una sola secuencia ancestral común. Los residuos alineados son usualmente interpretados por el compartiendo de su origen evolutivo. Cuando una secuencia no tiene residuos correspondientes debido a un evento de inserción o supresión, la posición se muestra como '-' u otro símbolo y se denomina 'brecha'. La mayoría de los programas de alineación no intentan filtrar secuencias no homólogas, dejando la decisión de qué secuencias incluir en el MSA (Alineamiento múltiple de secuencias) como una decisión externa para el usuario. Sin embargo, este problema es a veces importante en los análisis reales (Katoh & Toh, 2008).

1.1.3 Filogenia

Filogenia se refiere a la historia evolutiva de un grupo taxonómico de organismos. Es esencial en el estudio científico de la identificación, clasificación, ecología e historias evolutivas de los organismos. La filogenia muestra las relaciones entre grupos de organismos (taxones) en particular las diferencias y similitudes entre ellos. La filogenia está representada por un árbol llamado árbol filogenético. Una rama estrechamente relacionada de la ciencia que hace uso de los diagramas de árboles filogenético para estudiar las historias evolutivas y la relación entre varios grupos de organismos es filogenética. La relación entre taxones suele demostrarse mediante datos de secuenciación molecular y matrices de datos morfológicos. Por lo tanto, la filogenia es esencial para comprender la biodiversidad, la genética, las evoluciones y la ecología entre grupos de organismos (Lemey, Salemi, & Vandamme, 2009).

1.1.4 Árboles filogenéticos

El árbol filogenético muestra filogenia, es decir, las similitudes y diferencias en la morfología y la genética entre grupos de organismos (o taxones). Los taxones que se unen en el árbol filogenético implican relación evolutiva. También se puede suponer que han descendido de un hipotético antepasado común (nodo interno). Cuando el camino ancestral está implicado, se dice que el árbol filogenético está enraizado. Existen diferentes tipos de árboles filogenéticos. Un árbol filogenético enraizado es cuando los nodos implican el antepasado común más reciente de los taxa que son analizados. Otro tipo de árbol filogenético es el árbol no enraizado. En este tipo de árbol, no hay suposición sobre la ascendencia, sino sólo la relación evolutiva (Lemey, Salemi, & Vandamme, 2009).

1.1.5 Inferencia filogenética

La inferencia de una filogenia es un proceso de estimación donde se trata de obtener la mejor hipótesis posible de una historia evolutiva, basada en la información contenida en los datos. Un análisis filogenético permite conocer la cercanía entre las especies, así como determinar cuáles de ellas son homólogas, es decir, que han surgido de un ancestro común (Lemey, Salemi, & Vandamme, 2009).

1.1.6 Plataformas integradas

Las plataformas integradas para análisis filogenéticos son aquellas que ofrecen un conjunto de herramientas con ese fin, logrando que un investigador pueda realizar todo el contenido de su investigación sobre una misma aplicación web sin necesidad de buscar en otro sitio o aplicación web (Sanchéz, y otros, 2011).

1.1.7 Ergonomía

El término ergonomía proviene de un vocablo griego y hace referencia al estudio de los datos biológicos y tecnológicos que permiten la adaptación entre el hombre y las máquinas o los objetos. La traducción del

concepto griego está relacionada a las normas que regulan el accionar humano. La ergonomía, por lo tanto, analiza la interacción entre el ser humano y otros elementos de un sistema con el objetivo de promover el bienestar humano y el rendimiento del sistema (MONDELO, GREGORI, & BARRAU, 2005).

1.2 Estado del arte

A continuación, se describirán las herramientas para análisis filogenético con las cuales cuenta la plataforma PhylUCI, además se abordarán sobre las características, ventajas y desventajas de las plataformas web de acceso público más comunes, de dichas plataformas se extraerán sus características más frecuentes, así como las ventajas que brinden para lograr una mejor confección de la interfaz web con la que contará PhylUCI.

1.2.1 Herramientas que permiten realizar análisis filogenético

- **MAFFT**

MAFFT (Multiple Alignment using Fast Fourier Transform) es un programa de alineación de secuencias múltiples de alta velocidad que incluye dos técnicas algorítmicas. i) Utilizando la transformada rápida de Fourier (FFT) para identificar regiones homólogas rápidamente. En esta técnica, una secuencia de aminoácidos se convierte en una secuencia compuesta de valores de volumen y polaridad. ii) Uso de un sistema de puntuación simplificado para reducir el tiempo de CPU y aumentar la precisión de las alineaciones. MAFFT emplea un método progresivo (FFT-NS-2) y un método de refinamiento iterativo (FFT-NS-i) (Misawa, Miyata, & Katoh, 2002).

- **Muscle**

Programa informático para crear múltiples alineaciones de secuencias de proteínas. Los elementos del algoritmo incluyen la estimación rápida de la distancia usando la cuenta del kmer, la alineación progresiva usando una nueva función del perfil que llamamos la cuenta de la expectativa del registro, y el refinamiento usando la partición restringida dependiente del árbol (Edgar, 2004).

- **PhyML**

Es un programa cuyo principal propósito es la estimación de filogenias a partir de secuencias de nucleótidos o aminoácidos, empleando el método de máxima verosimilitud. Proporciona una amplia gama de opciones diseñadas para facilitar el análisis filogenético. La principal fortaleza de PhyML radica en el gran número de modelos de evolución que presenta, así como en las varias opciones de búsquedas en el espacio de topologías de árboles filogenéticos que posee, que comprenden desde métodos eficientes y rápidos hasta enfoques más lentos, pero más precisos. Implementa dos métodos para evaluar soportes

de ramas en un entorno estadístico sólido (bootstrap no paramétrico y test de radio de verosimilitud (Guindon S., y otros, 2010).

- **T-Coffee**

(Objetivo de coherencia basada en árboles Función para la evaluación de alineación) tiene dos características principales. En primer lugar, proporciona un medio simple y flexible de generar múltiples alineaciones, utilizando fuentes de datos heterogéneas. Los datos de estas fuentes se proporcionan a T-Coffee a través de una biblioteca de alineaciones en pares. Ahí se demuestra el poder de T-Coffee mediante el cálculo de múltiples alineaciones utilizando una biblioteca que se generó utilizando una mezcla de locales y global de pares de alineaciones (NotredameC, Higgins, & Heringa, 2000).

- **MrBayes**

Es un programa para la estimación filogenética mediante métodos bayesianos. La inferencia bayesiana de las filogenias está basada en la distribución de probabilidad posterior de los árboles, la que viene dada por la probabilidad de un árbol condicionada por las observaciones, entiéndase el alineamiento múltiple de secuencias (Pr (Árbol | Alineamiento)) y es a su vez determinada utilizando el teorema de Bayes. (Ronquist, y otros, 2012).

- **Clustalw**

ClustalW2 es una herramienta de alineamiento de secuencias múltiples para la alineación de secuencias de ADN o proteína. ClustalW2 calcula la mejor coincidencia para las secuencias de entrada basándose en los parámetros introducidos y genera un informe de fácil interpretación. Este informe de alineación de secuencias muestra la puntuación de alineación óptima, la alineación entre secuencias en una forma tal que las identidades, semejanzas y diferencias pueden ser claramente vistas y un árbol guía de las relaciones evolutivas de secuencias alineadas (Larkin, y otros, 2007).

- **ProbCons**

PROBCONS es una herramienta novedosa para generar múltiples alineaciones de secuencias de proteínas. Utilizando una combinación de modelado probabilístico y técnicas de alineación basadas en consistencia, PROBCONS ha logrado las máximas precisiones de todos los métodos de alineación hasta la fecha. En la base de datos de alineación de benchmark BAliBASE, las alineaciones producidas por PROBCONS muestran mejoras estadísticamente significativas sobre los programas actuales, que contienen un promedio de 7% más columnas correctamente alineadas que las de T-Coffee, 11% más alineadas correctamente que las de CLUSTAL W y 14 % De columnas más correctamente alineadas que las de DIALIGN (Brudno, Mahabhashyam, B. Do, & Batzoglou, 2005).

- **TNT**

TNT (Tree analysis using New Technology) El programa puede analizar conjuntos de datos con caracteres discretos (aditivos, no aditivos, paso-matriz) así como continuos (evaluados con la optimización de Farris). El análisis eficaz de grandes conjuntos de datos se puede llevar a cabo en tiempos razonables, y una serie de métodos para ayudar a identificar taxones y en el caso de conjuntos de datos ambiguos se aplican. Una variedad de métodos para diagnosticar árboles y explorar la evolución del carácter está disponible en TNT, y la calidad de la publicación de diagramas de árbol se pueden guardar como metarchivos. A través del uso de una serie de comandos nativos y un lenguaje de scripting sencillo pero potente, TNT permite al usuario una enorme flexibilidad en los análisis o simulaciones (Goloboff, Farris, & Nixon, 2008).

1.2.2 Plataformas existentes

- **Phylogeny.fr**

Es una plataforma que ofrece una serie de métodos principales para la alineación de secuencias múltiples, la reconstrucción filogenética y la representación gráfica de los árboles, esto lo logra atreves de tres modos. El modo “One Click” que está diseñado para los biólogos que no tienen experiencia en la bioinformática; dado un conjunto de secuencias no alineadas estas se ejecutan en un modo estándar usando Muscle, Gblocks, PhyML y TreeDyn el cual muestra el árbol filogenético en un formato listo para imprimir. El modo “Avanzado” permite la configuración de cada una de las herramientas utilizadas en el modo “One Click”. El modo “A la Carta” ofrece opciones flexibles en las secuencias, las herramientas y sus ajustes para adaptarlos a las necesidades más específicas de los biólogos expertos, además que este modo cuenta con una sesión que permite realizar pruebas sobre los resultados obtenidos (Dereeper, y otros, 2008).

La plataforma se ejecuta actualmente en un servidor dedicado (PowerEdge 2850-Xeon 2.8/2x2 MB Dual Core). Excepto el módulo Blast que esta paralelizado en un clúster de 25 CPU. Esto conlleva a que la ejecución de los programas se vea limitados por la baja capacidad de cómputo (ver figura 1). Phylogenic.fr es una plataforma que tuvo su última actualización en 2008 por lo que la tecnología empleada en la construcción del sitio es prácticamente obsoleta. Para la visualización de árboles hace uso de TreeDyn el cual vincula etiquetas de hojas únicas a listas de pares de variables / valores, independientemente de las topologías de árbol, permaneciendo totalmente compatible con el formato newick básico, pero por ser TreeDyn una herramienta de visualización externa hace que el sitio carezca de rapidez a la hora de la construcción, edición y visualización de árboles. Además de que es necesaria una conexión de banda ancha en todo momento a internet pues constantemente hay intercambio de información entre cliente y servidor.

- **Phylemon 2.0**

Es una plataforma que cuenta con más de 30 herramientas para realizar un análisis filogenético, siendo esta una de las primeras plataformas en desarrollar el concepto de plataformas integradas. Phylemon 2.0 ofrece un entorno integrado que permite la concatenación de análisis evolutivos, el almacenamiento de los resultados y a su vez maneja las conversiones en formato transparente. Además, una vez que se produce un archivo de salida, Phylemon sugiere otros análisis posibles, por lo que los usuarios pueden guardar secuencias completas que podrán reutilizar en investigaciones posteriores. Para la visualización de árboles usa ETE vs 2.1. (Sanchéz, y otros, 2011)

ETE vs 2.1 permite visualizar e interactuar con los árboles obtenidos brindando una serie de opciones al usuario como expandir, colapsar, intercambiar nodos, buscar distancias y otras opciones muy sencillas de usar. Phylemon 2.0 está implementado en java para la programación del lado del servidor y hace uso de la tecnología AJAX (Asynchronous JavaScript And XML) para la programación del lado del cliente, para el intercambio cliente servidor usa JSON (JavaScript Object Notation). Además, implementa un sistema de colas en el servidor. Esta versión hace uso de estándares web que le permite la compatibilidad con varios navegadores web como Chrome 7+, Firefox 3.5+, Safari 4+, Opera 10+ e Internet Explorer 8. La principal desventaja que tiene Phylemon 2.0 es que es a pesar de ser una plataforma del 2011 no cuenta con un personal de soporte.

- **ETE 3**

Es un marco computacional que simplifica la reconstrucción, análisis y visualización de árboles filogenéticos y alineamientos de secuencias múltiples, proporciona un conjunto de herramientas para realizar tareas comunes en genómica comparativa y filogenética por lo que permite construir filogenias basadas en genes y súper matrices utilizando un solo comando, además de probar y visualizar modelos evolutivos, así como calcular distancia entre árboles de diferentes tamaños (Huerta-Cepas, Serra, & Bork, 2016).

ETE se basa en una colección de múltiples herramientas externas que se instalan y ejecutan de forma transparente, un solo comando permite configurar y lanzar un análisis filogenético, el cual comienza por la alineación de secuencias, el recorte y prueba del modelo de sustitución, la inferencia de árboles y la representación gráfica del árbol obtenido. El modo de reconstrucción basado en súper matriz permite construir y concatenar múltiples alineaciones de secuencias con facilidad, simplificando la inferencia de árboles de especies basados en múltiples genes. Las opciones avanzadas permiten cambiar automáticamente de alineaciones de aminoácidos a nucleótidos basadas en la identidad de secuencia, reanudar la ejecución de flujos de trabajo o incluso probar múltiples estrategias en paralelo, todo esto se

puede hacer utilizando una sola línea de comandos para probar simultáneamente varias metodologías de alineación.

A pesar de ser ETE una moderna plataforma para análisis filogenético no cuenta con una interfaz gráfica que permita realizar un análisis filogenético cómodo para investigadores con pocos conocimientos informáticos, pues estos análisis se realizan mediante comandos. Dicha plataforma cuenta con un visualizador de árbol para formato FASTA y formato NEWICK, con solo cargar un fichero en alguno de estos formatos ETE se encarga de su visualización, pero este visualizador no cuenta con una edición para facilitar análisis a investigadores, para estas visualizaciones hace uso de librerías para Python. Además, es una plataforma que hace constante uso de herramientas externas para lograr todos sus resultados, disminuyendo así el tiempo de ejecución y respuesta de los análisis filogenéticos.

1.3 Interfaces gráficas de usuarios (GUI)

GUI es una interfaz de programa que aprovecha las capacidades gráficas de la computadora para facilitar el uso del programa. Las interfaces gráficas de usuario bien diseñadas pueden liberar al usuario del aprendizaje de complejos lenguajes de comandos. Por otro lado, muchos usuarios encuentran que trabajan más eficazmente con una interfaz controlada por comandos, especialmente si ya conocen el lenguaje de comandos (Myers, Hudson, & Pausch, 2000). Según (Shneiderman, 2010) algunos de los componentes gráficos comunes en este tipo de interfaces son:

- **Apuntador:** es un símbolo que aparece en la pantalla y que se desplaza para seleccionar objetos y comandos.
- **Dispositivo apuntador:** un dispositivo como el ratón que permite seleccionar objetos de la pantalla.
- **Iconos:** pequeñas imágenes que representan comandos, archivos o ventanas.
- **Menús:** opciones o comandos agrupados en forma de listas.
- **Ventanas**: son áreas que dividen la pantalla, y en cada una de ellas puede ejecutarse un programa o abrir un archivo diferente.
- **Escritorio:** es el área de la pantalla donde los iconos son agrupados.

La arquitectura comúnmente usada para el desarrollo de la GUI es a través del patrón modelo-vista-controlador (MVC), que permite separar los componentes gráficos, el control de los dispositivos periféricos y eventos, así como sus efectos en el modelo.

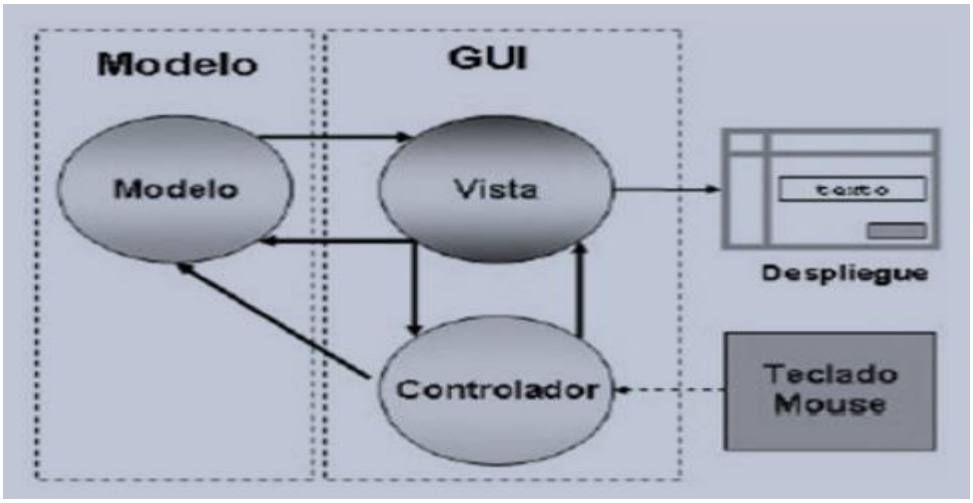


fig. 1 Patrón MVC como base de una GUI

1.3.1 Características de una interfaz de usuario

Una característica importante de la interfaz de usuario es que permite manipular los objetos e informaciones de la pantalla, no sólo presentarla. Además, los usuarios para utilizarlas deben conocer (o aprender): cómo está organizado el sistema (ficheros, directorios) los diferentes tipos de íconos y efectos de las acciones sobre ellos, los elementos básicos de una ventana, el uso de los controles y el uso del ratón (Shneiderman, 2010).

Las características que deben presentar las interfaces son:

- Poseer un monitor gráfico de alta resolución.
- Poseer un dispositivo apuntador (típicamente un ratón).
- Promover la consistencia de la interfaz entre programas.
- Seguir el paradigma de la interacción objeto-acción.
- Permitir la transferencia de información entre programas.
- Manipular en la pantalla directamente los objetos y la información.
- Proveer elementos de interfaz estándar como menús y diálogos.
- La existencia de una muestra visual de la información y los objetos (iconos y ventanas).
- Proporcionar respuesta visual a las acciones del usuario.
- Existencia de información visual de las acciones y modos del usuario/sistema (menús, paletas).
- Existencia de controles gráficos (widgets) para la selección e introducción de la información.
- Permitir a los usuarios personalizar la interfaz y las interacciones.
- Proporcionar flexibilidad en el uso de dispositivos de entrada (teclado/ratón).

1.3.2 Principios para el diseño de interfaces de usuarios

Existen principios básicos para el diseño e implementación de una Interfaz de Usuario (IU) (Galitz, 2007).

Familiaridad del usuario: significa que la interfaz debe utilizar términos e imágenes conocidos por el usuario; y los objetos que manipula el sistema deben estar relacionados con el ámbito de trabajo.

Uniformidad de la Interfaz: significa que tanto comandos como menús deben tener el mismo formato. Las Interfaces uniformes reducen el tiempo de aprendizaje.

Mínima sorpresa: el comportamiento del sistema no debe mostrar situaciones inesperadas. Ante éste tipo de situaciones el usuario puede mostrar irritabilidad, por lo tanto, perder interés en utilizar la aplicación.
Recuperación de estados: éste es uno de los principios más importantes al diseñar una Interfaz. Es inevitable cometer errores, por lo tanto, el sistema le debe proporcionar al usuario la manera de subsanarlos o volver a estados anteriores. Éste principio involucra varias acciones como pedir al usuario que confirme acciones destructivas, que el usuario pueda deshacer, etc.

Guía de usuarios: la Interfaz debe proporcionar al usuario asistencia, ayuda. No sólo cuando se cometen errores sino también cuando no se sabe qué hacer o cómo hacer alguna tarea. Esta ayuda debe estar integrada al sistema (algunas además ofrecen ayuda online) y debe ser clara cuando el usuario la requiera, sin saturar con información.
Diversidad de usuarios: se debe tener en cuenta los diferentes usuarios que pueden utilizar la aplicación. Aquellos casuales, que necesitan que los guíen, y aquellos que podrían usarla constantemente los cuales necesitarán trabajar con métodos abreviados, tan rápido como sea posible. Además, se podría incluir recursos para mostrar diferentes tamaños de texto, reemplazar sonido por texto y al revés, modificar tamaño de botones, etc.
Adoptar el punto de vista del usuario: se debe ver la interfaz desde fuera y en relación con las tareas que va a realizar el usuario. Hay que tener mucho cuidado en no centrarse en los aspectos de implementación que hagan perder la perspectiva.
Realimentación: la interfaz debe dar inmediatamente alguna respuesta a cualquier acción del usuario. Por ejemplo: movimiento del cursor, resaltar la opción elegida de un menú, comunicar el éxito o fracaso de una operación, reflejar el estado de los objetos.

Potenciar la sensación de control del usuario sobre el sistema, especialmente para los usuarios sin experiencia: que la interfaz sea intuitiva (utilizar iconos, modelos, métodos, etc. consistentes con otras aplicaciones y con el mundo real), facilitar la exploración (todas las operaciones deben ser accesibles desde el menú principal), permitir cancelar y deshacer operaciones, etc.

Minimizar la necesidad de memorización: usar controles gráficos, limitar la carga de información a corto plazo, procurar que la información necesaria en cada momento esté presente en la pantalla, utilizar nombres y símbolos auto-explicativos y fáciles de recordar, etc.

Anticipación: la aplicación debe anticiparse a las necesidades del usuario, y no esperar a que tenga que buscar información.

Percepción de color y tamaño: se debe tener en cuenta a aquellos usuarios con problema de visualización del color, pero es muy útil usar convención de colores. Además, al mostrar varios objetos en la pantalla deben estar distribuidos, debe haber distancia entre ellos para que así el usuario pueda percibirlos sin sobrecarga.

Legibilidad: no sólo se debe prestar atención a los colores y a los objetos que se ven en pantalla sino también a cómo se verá el texto. El tipo y tamaño de letra debe ser legible, y el color debe contrastar con el fondo (utilizar letras negras en fondo claro).

Valores por defecto: lo ideal es utilizar 'valores estándar'. Se debe tener en cuenta que los valores por defecto deben ser opciones inteligentes, sensatas y fáciles de modificar.

Eficiencia: se debe considerar la productividad como ideal a lograr. El usuario no debe esperar la respuesta del sistema por tiempo prolongado; los mensajes de ayuda, menús y etiquetas deben ser sencillos y deben utilizar palabras claves para poder transmitir fácilmente a qué hacen referencia.

1.4 Metodología, tecnologías y herramientas a emplear en la solución.

1.4.1 Metodología de desarrollo de software.

La metodología para el desarrollo de software es un modo sistemático de realizar, gestionar y administrar un proyecto para llevarlo a cabo con altas posibilidades de éxito. Una metodología para el desarrollo de software comprende los procesos a seguir sistemáticamente para idear, implementar y mantener un producto software desde que surge la necesidad del producto hasta que se cumple el objetivo por el cual fue creado (Letelier Torres, Penadés, Canós, & Sánchez López, 2003).

Una definición estándar de metodología puede ser el conjunto de métodos que se utilizan en una determinada actividad con el fin de formalizarla y optimizarla. Determina los pasos a seguir y cómo realizarlos para finalizar una tarea, dentro de estas metodologías se encuentra OpenUP.

- **Metodología OpenUP**

OpenUP es un proceso unificado que aplica enfoques iterativos e incrementales dentro de un ciclo de vida estructurado. OpenUP abarca una filosofía pragmática y ágil que se centra en la naturaleza colaborativa del desarrollo de software. Se trata de un proceso agnóstico que puede ampliarse para abordar una amplia variedad de tipos de proyectos (Balduino, 2007).

Se escoge esta metodología ya que:

- Es apropiada para proyectos pequeños y de bajos recursos, permite disminuir las probabilidades de fracaso en los proyectos pequeños e incrementar las probabilidades de éxito.
- Permite detectar errores tempranos a través de un ciclo iterativo e incremental.
- Se centra en la arquitectura de forma temprana para minimizar el riesgo y organizar el desarrollo, además de obtener versiones operativas desde etapas tempranas.
- Por ser una metodología ágil tiene un enfoque centrado al cliente y con iteraciones cortas.

1.4.2 Lenguaje de modelado

Lenguaje unificado de modelado (UML, por sus siglas en inglés, Unified Modeling Language) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; está respaldado por el OMG (Object Management Group).

Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un modelo del sistema, incluyendo aspectos conceptuales tales como procesos de negocio, funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y compuestos reciclados (Group, 2005).

Es importante remarcar que UML es un lenguaje de modelado para especificar o para describir métodos o procesos. Se utiliza para definir un sistema, para detallar los artefactos en el sistema y para documentar y construir.

- **UML 2.0**

El Lenguaje Unificado de Modelado (UML) es un lenguaje para especificar, visualizar, construir, y documentar los artefactos de los sistemas de software, así como para el modelado del negocio y otros sistemas no software (Larman, 2007).

1.4.3 Herramientas case

Son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero. Estas herramientas son de gran ayuda en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el proceso de realizar un diseño del proyecto, cálculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores entre otras ***(Sommerville, 2010)***

- **Visual Paradigm for UML 8.0**

Visual Paradigm (anteriormente VP-UML) es una herramienta UML. La herramienta está diseñada para una amplia gama de usuarios, incluyendo ingenieros de software, analistas de sistemas, analistas de negocios y arquitectos de sistemas, o para cualquier persona que esté interesada en la construcción fiable de los sistemas de software a gran escala con un enfoque orientado a objetos (Paradingm, 2013).

Se escoge la herramienta Visual Paradigm por soportar el ciclo de vida completo del proceso de desarrollo del software a través de la representación de todo tipo de diagramas, sirviendo como guía y facilitando el proceso de desarrollo de todas las fases de creación del módulo (Paradingm, 2013).

1.4.4 Lenguaje de programación

Es un lenguaje diseñado para describir el conjunto de acciones consecutivas que un equipo debe ejecutar. Por lo tanto, un lenguaje de programación es un modo práctico para que los seres humanos puedan dar instrucciones a un equipo.

- **Python 2.7**

Python es un lenguaje de programación interpretativo, interactivo y orientado a objetos. Incorpora módulos, excepciones, tipografía dinámica, tipos de datos dinámicos de muy alto nivel y clases. Tiene interfaces para muchas llamadas de sistema y bibliotecas, así como a varios sistemas de ventanas, y es extensible en C o C ++. También es útil como un lenguaje de extensión para aplicaciones que necesitan una interfaz programable. Por último, Python es portátil: se ejecuta en muchas variantes de Unix, en Mac, y en Windows 2000 y versiones posteriores. (Stevens & Boucher, 2015)

- **JavaScript 1.6**

JavaScript es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas. Una página web dinámica es aquella que incorpora efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario. Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios (Rauschmayer, 2014).

1.4.5 Marco de trabajo

Un marco de trabajo o framework es un diseño abstracto orientado a objetos para un determinado tipo de aplicación, es un patrón arquitectónico que proporciona una plantilla extensible para un tipo específico de aplicaciones. Es un conjunto cohesivo de interfaces y clases que colaboran para proporcionar los servicios

de la parte central e invariable de un subsistema lógico. (Anglada Martínez, Ruíz Constanten, & Torres Rubio, 2013)

- **Django 1.6.11**

Django es un framework de desarrollo web de código abierto, escrito en Python, que respeta el patrón de diseño conocido como Modelo–vista–controlador. Fue desarrollado en origen para gestionar varias páginas orientadas a noticias de la World Company de Lawrence, Kansas, y fue liberada al público bajo una licencia BSD en julio de 2005. La meta fundamental de Django es facilitar la creación de sitios web complejos. Django pone énfasis en el re-uso, la conectividad y extensibilidad de componentes, el desarrollo rápido y el principio No te repitas (DRY, del inglés *Don't Repeat Yourself*). Python es usado en todas las partes del framework, incluso en configuraciones, archivos, y en los modelos de datos (Team D. , 2013).

1.4.6 Entorno de desarrollo

Un entorno de desarrollo integrado(IDE), es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI) (IBM, 2009).

- **PyCharm 2016.3**

PyCharm es un IDE o entorno de desarrollo integrado multiplataforma utilizado para desarrollar en el lenguaje de programación Python. Proporciona análisis de código, depuración gráfica, integración con VCS / DVCS y soporte para el desarrollo web con Django, entre otras bondades. PyCharm es desarrollado por la empresa JetBrains y debido a la naturaleza de sus licencias tiene dos versiones, la Community que es gratuita y orientada a la educación y al desarrollo puro en Python y la *Professional*, que incluye más características como el soporte a desarrollo (Team J. , 2000).

Conclusiones del capítulo

En este capítulo que concluye se definieron las herramientas y tecnologías que posibilitarán el correcto desarrollo de la Plataforma de Análisis Filogenético PhylUCI, seleccionándose como metodología OpenUP, la cual guiará el correcto desarrollo del sistema propuesto. Eligiendo UML 2.1 como lenguaje de modelado y Visual Paradigm 8.0 como herramienta CASE. El lenguaje de programación a utilizar, Python y JavaScript, para facilitar el desarrollo se utilizará el framework Django, PyCharm como IDE de desarrollo. Cada una de estas herramientas y tecnologías posee características que las hacen idóneas para el trabajo con ellas durante el proceso de desarrollo del mismo.

CAPÍTULO 2: ANÁLISIS Y DISEÑO DEL FRONTEND

En este capítulo se muestran los artefactos que más se adaptan a las necesidades de la solución, generados a partir de la metodología OpenUP. Se definen los requisitos funcionales y no funcionales, así como los casos de uso que los contienen, además se define la arquitectura del sistema y los elementos que componen a esta. Se muestra el modelo conceptual y su explicación en detalle. Además de mostrar el diagrama de casos de uso del sistema y la descripción de sus actores.

2.1 Modelo del dominio

El modelo de domino es una representación visual de las clases conceptuales u objetos del mundo real en un dominio de interés. También se les denomina, modelos de objetos de dominio y modelos de objetos de análisis (Larman, 2007).

Diagrama conceptual del modelo de dominio

EL presente modelo de dominio tiene como objetivo simplificar la comprensión de la plataforma web para análisis filogenético.

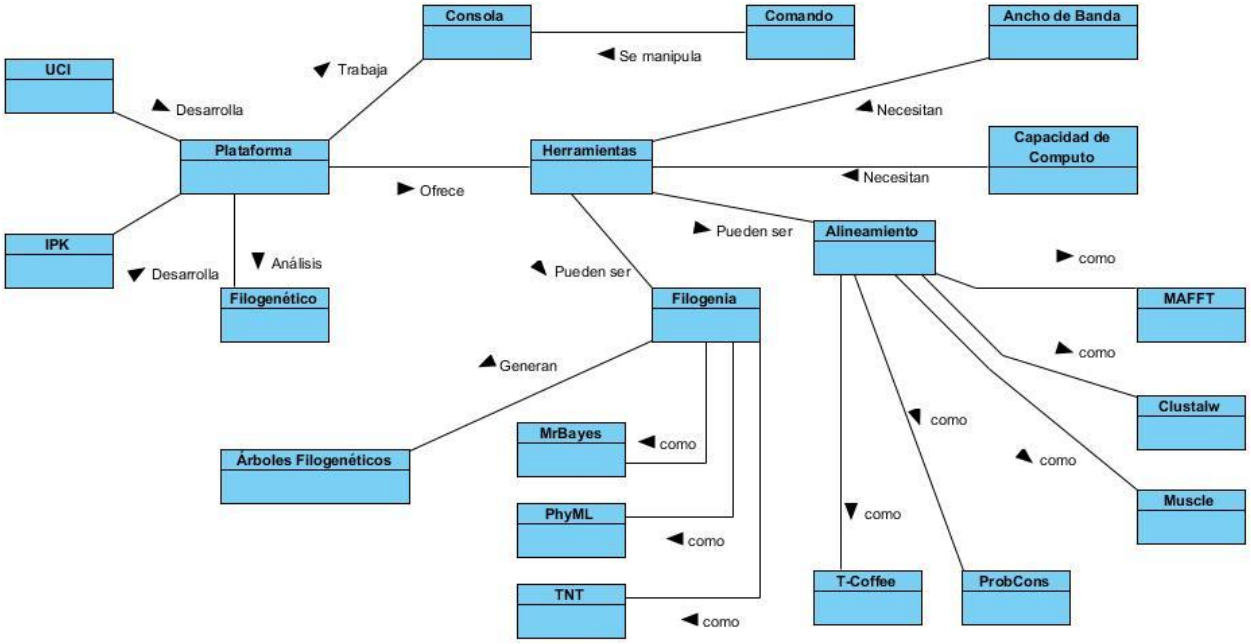


fig. 2 Diagrama del modelo de dominio

Descripción de los Objetos del Modelo del Dominio

- **UCI:** Universidad de las Ciencias Informáticas.
- **IPK:** Instituto de medicina tropical “Pedro Kouri”.
- **Plataforma:** plataforma de análisis filogenético.

- **Filogenética:** rama de la biología evolutiva.
- **Arboles Filogenéticos:** resultado obtenido después de un análisis filogenético.
- **Muscle:** programa que permite realizar análisis filogenético.
- **PhyMl:** programa que permite realizar análisis filogenético.
- **T-Coffee:** programa que permite realizar análisis filogenético.
- **TNT:** programa que permite realizar análisis filogenético.
- **MAFFT:** programa que permite realizar análisis filogenético.
- **MrBayes*:*** programa que permite realizar análisis filogenético.
- **Clustalw:** programa que permite realizar análisis filogenético.
- **ProbCons:** programa que permite realizar análisis filogenético.
- **Herramientas:** conjunto de aplicaciones existentes en la plataforma para realizar análisis filogenético.

2.2 Requisitos del sistema

La especificación de los requisitos del software es el proceso donde se define una descripción detallada de todos los aspectos del software antes de comenzar la construcción del mismo (Pressman, 2010).

2.2.1 Requisitos funcionales

Los requisitos funcionales de un sistema describen el comportamiento del sistema. Estos requisitos dependen del tipo de software que se comience a desarrollar, lo que esperan los usuarios del software y el enfoque general tomado por la organización al escribir los requisitos (Sommerville, 2010).

A continuación, se muestran los requisitos funcionales definidos:

- **RF1:** Registrar usuario.
- **RF2:** Autenticar usuario.
- **RF3:** Subir secuencia.
- **RF4:** Crear flujo de trabajo.
- **RF5:** Editar configuración del flujo de trabajo.
- **RF6:** Seleccionar herramientas.
- **RF7:** Visualizar árbol rectangular.

- **RF8:** Visualizar árbol radial.
- **RF9:** Mover árbol en área de trabajo.
- **RF10:** Ajustar escala de árbol.
- **RF11:** Marcar camino de un nodo a sus descendientes.
- **RF12:** Seleccionar aristas.
- **RF13** Rotar árbol radial.
- **RF14:** Colapsar nodos.
- **RF15:** Guardar configuración.
- **RF16:** Ajustar tamaño de fuente.
- **RF17:** Cambiar color de fuente.
- **RF18:** Ajustar tamaño de nodos.
- **RF19:** Mostrar etiquetas de nodos hoja.
- **RF20:** Mostrar etiquetas de nodos interiores.
- **RF21:** Mostrar etiquetas de distancia.
- **RF22:** Seleccionar color de marcado de los nodos a sus descendientes.
- **RF23:** Exportar imagen de árbol en formato PNG.
- **RF24:** Notificar por correo electrónico.

2.2.2 Requisitos no funcionales

Pueden estar relacionados con las propiedades del sistema como la fiabilidad, tiempo de respuesta y la capacidad de almacenamiento. Alternativamente, pueden definir restricciones en la implementación del sistema, como las capacidades de los dispositivos de Entrada/Salida (E/S) o de las representaciones de datos utilizadas en las interfaces con otros sistemas. Los requisitos no funcionales, como el rendimiento, la seguridad y la disponibilidad, suelen especificar o restringir características del sistema en su conjunto (Sommerville, 2010).

A continuación, se describen los requisitos no funcionales del sistema:

Requerimientos de hardware

Los requerimientos mínimos de hardware son los que deben existir en las computadoras para asegurar el correcto funcionamiento.

RNF 1:

- **Procesador:** Dual Core 1.7 GHZ
- **Memoria RAM:** 1 GB

Requerimientos de software

Los requerimientos mínimos de software son los que deben cumplir las computadoras para un correcto funcionamiento.

RNF 2:

- **Sistema operativo:** Windows 7 o superior, Ubuntu 12.04 o superior
- **Navegadores:** Firefox 19+, Chrome 21+, Internet Explorer 8+

Requerimientos de restricción del diseño y la implementación

RNF 3:

- **Servidor:** La implementación por parte del servidor tiene que ser en el lenguaje de programación Python usando el framework Django.
- **Visualizador de árboles:** Se tiene que usar para su implementación el lenguaje de programación JavaScript, haciendo uso de librerías como D3.

Requerimientos de seguridad

RNF 4:

- **Usabilidad:** Si ocurre una desconexión en el visualizador de árboles, el investigador podrá seguir realizando su análisis sobre ese mismo árbol resultante.
- **Integridad:** Los archivos subidos y los resultados arrojados no sufrirán cambios durante un análisis filogenético.

2.3 Modelado del sistema

2.3.1 Actores del sistema

Los actores representan un tipo de usuario del sistema. Un actor es una entidad externa que interactúa de alguna manera con el historial del caso de uso. Se entiende por actor una persona, un sistema

informatizado u organización. Generalmente son los responsables de las actividades que serán informatizadas.

Tabla 1. Descripción de actores del sistema.

Actor	Descripción
Investigador	Persona que realiza un análisis filogenético, crea flujos de trabajo y puede editar y visualizar árboles

2.3.2 Modelo de casos de uso del sistema

El modelo de casos de uso del sistema es una forma de representar gráficamente la relación existente entre los actores y casos de uso del sistema (CUS), de manera que sirva de guía para reflejar las metas y funcionalidades que persigue el negocio (Pressman, 2010).

Diagrama de casos de uso del sistema (DCU)

El diagrama de casos de uso refleja como los actores interactúan con los casos de uso.

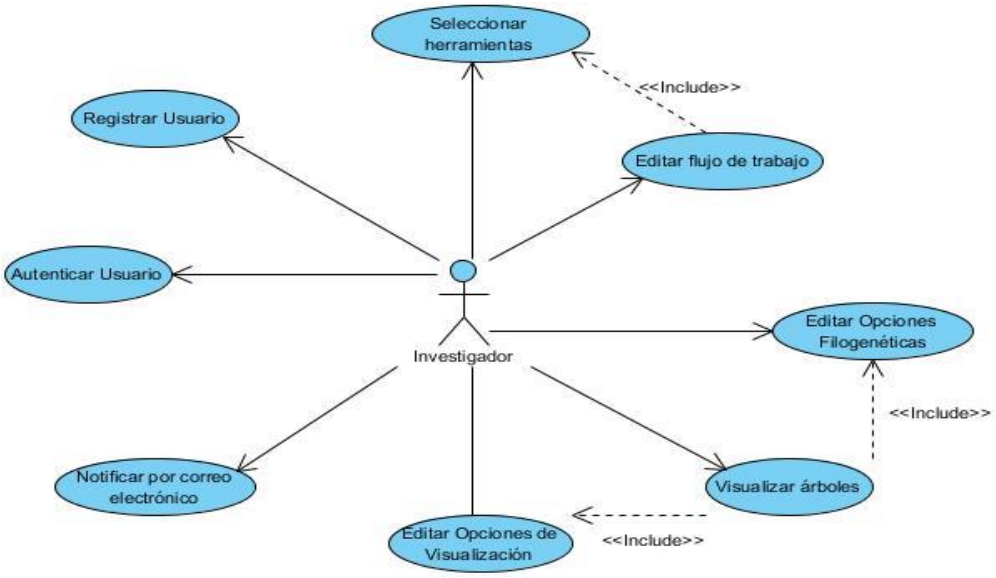


fig. 3 Diagrama de Casos de Uso del Sistema

En este diagrama se utiliza los patrones de diseño de casos de uso:

- **Concordancia de adición:**

La subsecuencia común de casos de uso, extiende los casos de uso compartiendo la subsecuencia de acciones. Los otros casos de uso modelan el flujo que será expandido con la subsecuencia. Este patrón es

preferible usarlo cuando otros casos de uso se encuentran propiamente completos, o sea, que no requieren de una subsecuencia común de acciones para modelar los usos completos del sistema.

- **Inclusión concreta:**

Se incluye una relación del caso de uso base al caso de uso de inclusión. El último puede ser instanciado en sí mismo.

2.3.3 Descripción de los casos de uso del sistema

Los casos de uso son requisitos; ante todo son requisitos funcionales que indican que hará el sistema. Definen una promesa o contrato de la manera en la que se comportará un sistema. Los casos de uso son documentos de textos, no diagramas, y el modelo de casos de uso es, sobre todo, una acción de escribir texto, no dibujar. Sin embargo, UML define un diagrama de casos de uso para ilustrar los nombres de casos de uso y actores, y sus relaciones *(Larman, 2007)*.

Tabla 2. Descripción del caso de uso "Editar opciones filogenéticas".

Objetivo	Editar árboles	
Actores	Investigador :(Inicia)	
Resumen	El caso de uso comienza cuando el investigador accede al Módulo Visor de árboles, y carga un archivo en formato Newick, este archivo se lee y se muestra un árbol filogenético con todas las opciones de edición. El caso de uso finaliza cuando el investigador abandona el módulo de visor de árboles.	
Complejidad	Alta	
Prioridad	Alta	
Referencias	RF6, RF7, RF8, RF9, RF10, RF11, RF12, RF 13	
Precondiciones	Debe de subirse un archivo en formato Newick.	
Postcondiciones	Se visualiza un árbol filogenético.	
Flujo de evento		
Flujo básico Editar opciones filogenéticas		
	Actor	**Sistema**
1.	Selecciona en el menú, la opción Visor de árboles.	
2.		Muestra la interfaz para la visualización de árboles, con el formulario para subir

		un archivo.
3.	Carga un archivo en formato Newick.	
4.		1. Lee el archivo. 2. Visualiza un árbol.
Sección 1: Visualizar árbol rectangular		
Flujo básico "Visualizar árbol rectangular"		
	Actor	**Sistema**
1.	Selecciona la opción **Visualizar árbol rectangular**.	
2.		Muestra el árbol en forma rectangular.
Sección 2: Visualizar árbol Radial		
Flujo básico "Visualizar árbol Radial"		
	Actor	**Sistema**
1.	Selecciona la opción **Visualizar árbol Radial.**	
2.		Muestra el árbol en forma radial.
Sección 3: Mover árbol en área de trabajo		
Flujo básico "Mover árbol en área de trabajo"		
	Actor	**Sistema**
1.	Hace click sobre el árbol y moverlo hacia el lugar donde se desee dentro del área de trabajo	
2.		Traslada el árbol a la posición seleccionada por el actor.
Sección 5: Ajustar escala de árbol		
Flujo básico "Ajustar escala de árbol"		
	Actor	**Sistema**
1.	Selecciona la escala que desea ajustar al árbol.	
2.		Aplica la escala seleccionada en el árbol.
Sección 6: Marcar camino de un nodo a sus descendientes		

Flujo básico "Marcar camino de un nodo a sus descendientes"		
	Actor	**Sistema**
1.	Selecciona el nodo que desee.	
2.		Muestra un menú con opciones.
3.	Selecciona la opción marcar descendientes.	
4.		Marca el camino hacia todos los descendientes del nodo seleccionado.
Sección 7: Seleccionar aristas		
Flujo básico "Seleccionar aristas"		
1.	Hace click sobre la arista que desea.	
2.		Marca la arista seleccionada.
Sección 8: Colapsar nodos		
Flujo básico " Colapsar nodos"		
1.	Selecciona el nodo que desea colapsar.	
2.		Muestra un menú con opciones.
3.	Selecciona la opción colapsar nodos.	
4.		Colapsa todos los descendientes de ese nodo mostrando en su lugar un triángulo.

Relaciones	**CU incluidos**	Visualizar árbol
	CU extendidos	N/P
Requisitos no funcionales	RNF2, RNF3, RNF4	
Asuntos pendientes	N/P	

2.4 Diseño del sistema

En el diseño se modeló el sistema y se encontró su forma (incluida la arquitectura) para que soporte todos los requisitos incluyendo los requisitos no funcionales y otras restricciones que se le suponen. Una entrada esencial en el diseño es el resultado del análisis, esto es, el modelo de análisis el cual proporciona una comprensión detallada de los requisitos. Y lo que es más importante impone una estructura del sistema que se debe conservar lo más fielmente posible cuando se construya el sistema (Jacobson, 2000).

2.4.1 Arquitectura del sistema

La arquitectura de software debe modelar la estructura de un sistema y la manera en la que los datos y los componentes colaboran unos con los otros (Pressman, 2010).

La arquitectura no es el software operativo, es una representación que permite:

- Analizar la efectividad del diseño para cumplir los requerimientos establecidos.
- Considerar alternativas arquitectónicas en una etapa en la que hacer cambios de diseño todavía es relativamente fácil.
- Reducir los riesgos asociados con la construcción del software.

Patrón arquitectónico: El sistema se desarrollará en Django lo que implica que se utilizará el patrón Modelo-Vista-Plantilla (MTV por sus siglas en inglés) que es una extensión del patrón modelo-vista-controlador.

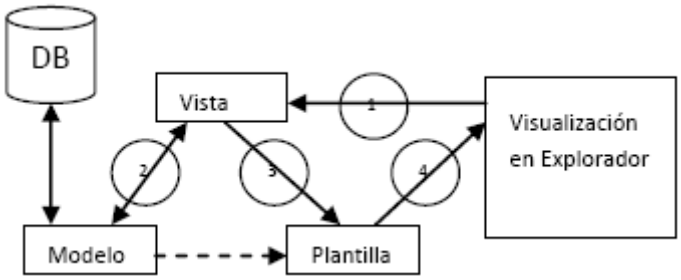


fig. 4 Patrón arquitectónico Modelo Vista Plantilla

El modelo en Django sigue siendo modelo, define los datos almacenados, se encuentra en forma de clases Python, cada tipo de dato que debe ser almacenado se encuentra en una variable con ciertos parámetros. Todo esto permite indicar y controlar el comportamiento de los datos.

La vista en Django se llama Plantilla (Template), la plantilla es básicamente una página HTML con algunas etiquetas extras propias de Django, en sí no solamente crea contenido en HTML (también XML, CSS, Javascript, CSV).

El controlador en Django se llama Vista, la vista se presenta en funciones Python, su propósito es determinar qué datos serán visualizados en las plantillas. El Mapeo objeto-relacional (ORM por sus siglas en ingles) de Django permite escribir código Python en lugar de SQL para hacer las consultas que necesita la vista. La vista no tiene nada que ver con el estilo de presentación de los datos, solo se encarga de determinar qué datos mostrará la plantilla.

El patrón modelo-vista-controlador fue diseñado para reducir el esfuerzo de programación necesario en la implementación de sistemas. Sus características principales son que el modelo, las vistas y los controladores son tratados como entidades separadas, esto hace que cualquier cambio producido en el modelo se refleje automáticamente en cada una de las vistas.

Este patrón arquitectónico tiene como ventajas:

- Hay una clara separación entre los componentes de un programa, lo cual permite implementarlos por separado.
- Hay un API muy bien definida, cualquiera que use esa API, podrá hacer cambios en el modelo, la vista o el controlador sin dificultad.
- La conexión entre el modelo y sus vistas es dinámica, se produce en tiempo de ejecución, no en tiempo de compilación.

Al incorporar el modelo de arquitectura MVC a un diseño, las piezas de un programa se pueden construir por separado y luego unirlas en tiempo de ejecución. Si uno de los Componentes, posteriormente, se observa que funciona mal, puede reemplazarse sin que las otras piezas se vean afectadas.

2.4.2 Patrones utilizados en el diseño de la solución

El patrón es una pareja de problema/solución con un nombre y que es aplicable a otros contextos, con una sugerencia sobre la manera de usarlos en situaciones nuevas. Los patrones no se proponen descubrir ni expresar nuevos principios de la ingeniería del software. Todo lo contrario: intentan codificar el conocimiento, las expresiones y los principios ya existentes (Larman, 2007).

- **Patrones GRAP**

GRASP es un acrónimo que significa General Responsibility Asignment Software Patterns (patrones generales de software para asignar responsabilidades). El nombre se eligió para indicar la importancia de captar (grasping) estos principios, si se quiere diseñar eficazmente el software orientado a objetos. Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones (Larman, 2007).

Controlador: es un objeto que no pertenece a la interfaz de usuario, responsable de recibir o manejar un evento del sistema. Un Controlador define el método para la operación del sistema. Todas las peticiones Web son manipuladas por un solo controlador frontal (URLCONF), que es el punto de entrada único de toda la aplicación en un entorno determinado. En este caso el “home.vew.py” escucha las peticiones que vienen desde una URL, luego maneja la acción requerida para satisfacer la petición realizada.

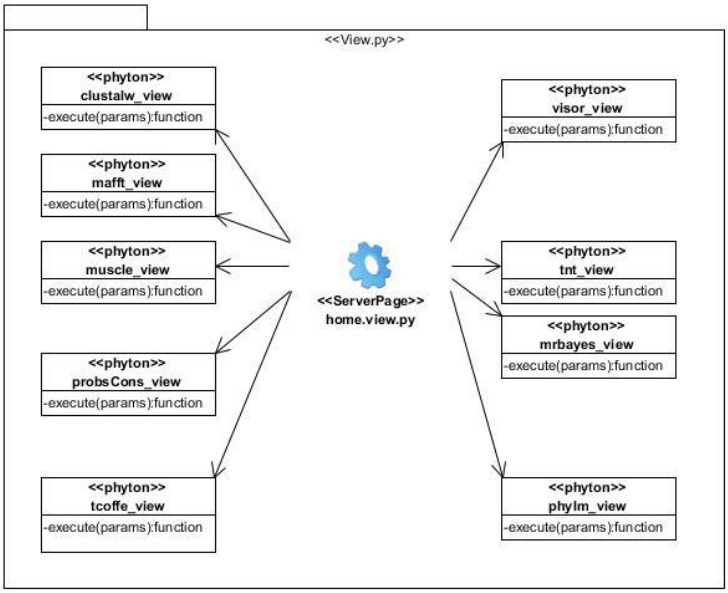


fig. 5 Patrón Controlador.

Creador: el patrón creador ayuda a identificar quién debe ser el responsable de la creación (o instanciación) de nuevos objetos o clases.

La nueva instancia deberá ser creada por la clase que:

- Tiene la información necesaria para realizar la creación del objeto.
- Usa directamente las instancias creadas del objeto.
- Almacena o maneja varias instancias de la clase.
- Contiene o agrega la clase.

fig. 6 Patrón creador

- **Patrones GOF**

Los patrones GoF (Gang of Four) se utilizan en situaciones frecuentes. Debido a que se basan en la experiencia acumulada al resolver problemas reiterativos. Ayudan a construir software basado en la reutilización, a construir clases reutilizables (Jacobson, 2000).

Constructor: el patrón builder (Constructor) es usado para permitir la creación de una variedad de objetos complejos desde un objeto fuente (Producto), el objeto fuente se compone de una variedad de partes que

contribuyen individualmente a la creación de cada objeto complejo a través de un conjunto de llamadas a interfaces comunes de la clase Abstract Builder (Osmani, 2012).

```
function menu(d) {
    d3.select('#' + menu_id).remove();

    var div = d3.select("body").append("ul")
        .attr("id", menu_id)
        .attr("class", "dropdown-menu")
        .attr("role", "menu")
        .style("display", "block")
        .style('position', 'absolute');

}
```

fig. 7 Patrón constructor

Adaptador: el patrón Adapter (Adaptador) se utiliza para transformar una interfaz en otra, de tal modo que una clase que no pudiera utilizar la primera, haga uso de ella a través de la segunda Adapter permite a las clases trabajar juntas, lo que de otra manera no podría hacerlo debido a sus interfaces incompatibles (Osmani, 2012).

```
d3.select(this)
        .style('stroke', 'SteelBlue ')
        .style('stroke-width', '8.5px');
```

fig. 8 Patrón adaptador

Fachada: el patrón Fachada (*Facade*) es un tipo de patrón de diseño estructural. Viene motivado por la necesidad de estructurar un entorno de programación y reducir su complejidad con la división en subsistemas, minimizando las comunicaciones y dependencias entre estos (Osmani, 2012).

```
var cluster = d3.layout.cluster()
    .size([360, innerRadius])
    .children(function (d) {
        return d.children;
    })
    .value(function (d) {
        return 1;
    })
    .sort(function (a, b) {
        return (a.value - b.value) || d3.ascending(a.length, b.length);
    })
    .separation(function (a, b) {
        return 1;
    });
```

fig. 9 Patrón fachada

Observador: el patrón observador consiste en un objeto, llamado *'sujeto'*, que mantiene una lista de dependencias, *'observadores'*, las cuales le notifican automáticamente de cualquier cambio (Osmani, 2012).

```
var node =vis.selectAll('g.inner.node')
    .append('svg:circle')
    .attr('r', '4')
    .attr('fill', '#000')
    .attr('stroke', '#aaa')
    .attr('stroke-width', '2px')
    .on("mouseover", function () {
        d3.select(this)
            .transition()
            .duration(100)
            .attr("r", 10)
            .attr('fill', '#4E3660')

    })
    .on("mouseout", function () {
        d3.select(this)
            .transition()
            .duration(100)
            .attr("r", 4)
            .attr('fill', 'black')
        ;
    })
    .on('click',menu);
```

fig. 10 Patrón observador

Iterador: el patrón iterador se encarga de recorrer un conjunto de elementos para interactuar de forma individual con cada uno de ellos (Osmani, 2012).

```
var childCount = function(level, n) {

    if (n.children && n.children.length > 0) {
        if (levelWidth.length <= level + 1)
            levelWidth.push(0);

        levelWidth[level + 1] += n.children.length;
        n.children.forEach(function(d) {
            childCount(level + 1, d);
        });
    }
};
```

fig. 11 Patrón iterador

2.5 Paradigma de programación orientado a eventos

En el presente epígrafe se abordará sobre la programación orientada a eventos, paradigma utilizado en la creación del visualizador de árboles, también se conceptualizará D3.js librería JavaScript utilizada para crear todas las funcionalidades del visualizador y editor de árboles filogenéticos.

La programación orientada a eventos es un paradigma de programación en el que tanto la estructura como la ejecución del programa van determinados por los sucesos que ocurran en el sistema, definidos por el usuario o provocados por ellos.

Para entender la programación dirigida por eventos, se debe oponer a lo que no es: mientras en la programación secuencial (o estructurado) es el programador el que define cuál va a ser el flujo del programa, en la programación dirigida por eventos será el propio usuario o lo que sea que esté accionando el programa el que dirija el flujo del programa. Aunque en la programación secuencial puede haber intervención de un agente externo al programa, estas intervenciones ocurrirán cuando el programador lo haya determinado, y no en cualquier momento como puede ser en el caso de la programación dirigida por eventos.

El creador de un programa dirigido por eventos debe definir los eventos que manejarán su programa y las acciones que se realizarán al producirse cada uno de ellos, lo que se conoce como el administrador de evento. Los eventos soportados estarán determinados por el lenguaje de programación utilizado, por el sistema operativo e incluso por eventos creados por el mismo programador.

En la programación dirigida por eventos, al comenzar la ejecución del programa se llevarán a cabo las inicializaciones y demás código inicial y a continuación el programa quedará bloqueado hasta que se produzca algún evento. Cuando alguno de los eventos esperados por el programa tenga lugar, el programa pasará a ejecutar el código del correspondiente administrador de evento. Por ejemplo, si el evento consiste en que el usuario ha hecho click en el botón de play de un reproductor de películas, se ejecutará el código del administrador de evento, que será el que haga que la película se muestre por pantalla (Palmer, 2008).

2.5.1 Herramientas visuales de desarrollo

Con el paso del tiempo, han ido apareciendo una nueva generación de herramientas que incluyen código que automatiza parte de las tareas más comunes en la detección y tratamiento de eventos. Destacan particularmente los entornos de programación visual que conjugan una herramienta de diseño gráfica para la GUI y un lenguaje de alto nivel.

D3.js

D3.js es una biblioteca JavaScript para manipular documentos basados en datos. D3 le ayuda a llevar datos a la vida usando HTML, SVG y CSS. El énfasis de D3 en los estándares web le brinda las capacidades completas de los navegadores modernos sin atarse a un marco propietario, combinando poderosos componentes de visualización y un enfoque basado en datos para la manipulación DOM.

D3 no es un marco monolítico que busque proporcionar todas las características imaginables. En su lugar, D3 resuelve el quid del problema: la manipulación eficiente de documentos basados en datos. Esto evita la representación propietaria y ofrece una flexibilidad extraordinaria, exponiendo las capacidades completas de estándares web como HTML, SVG y CSS. Con una sobrecarga mínima, D3 es extremadamente rápido, soportando conjuntos de datos grandes y comportamientos dinámicos para interacción y animación. El estilo funcional de D3 permite la reutilización de código a través de una diversa colección de módulos oficiales y desarrollados por la comunidad (Bostock, 2011).

Conclusiones del capitulo

En el capítulo que finaliza se obtuvo una idea más detallada del diseño y funcionamiento del sistema. Se confeccionó el modelo de dominio, donde se creó el diagrama conceptual del mismo. Se definieron los requisitos funcionales y no funcionales a tener en cuenta logrando una idea más precisa de las funcionalidades del sistema. Se identificaron 24 requisitos funcionales agrupándose en 8 casos de uso del sistema los cuales se relacionaron mediante un diagrama de casos de uso del sistema. Se determinó el

patrón arquitectónico, así como los patrones de diseños utilizados. Se caracterizó el paradigma de programación utilizado para el desarrollo del visualizador de árboles.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA DEL FRONTEND

En el capítulo se aborda sobre las actividades que se llevan a cabo durante la fase de implementación y pruebas. Se analiza el modelo de Implementación, así como una descripción de cómo los elementos del modelo de diseño se implementan en términos de componentes. Además, se diseña y aplican las pruebas para comprobar el correcto funcionamiento del sistema.

3.1 Modelo de implementación

El Modelo de Implementación es comprendido por un conjunto de componentes y subsistemas que constituyen la composición física de la implementación del sistema. Entre los componentes se puede encontrar datos, archivos, ejecutables, código fuente y los directorios. Fundamentalmente, se describe la relación que existe desde los paquetes y clases del modelo de diseño a subsistemas y componentes físicos (Pressman, 2010).

3.1.1 Diagrama de componentes

Los diagramas de componentes son usados para estructurar el modelo de implementación en términos de subsistemas de implementación y mostrar las relaciones entre los elementos de implementación. El uso más importante de estos diagramas es mostrar la estructura de alto nivel del modelo de implementación, especificando:

- Los subsistemas de implementación y sus dependencias a la hora de importar código.
- Organizar los subsistemas de implementación en capas.

También se utilizan para mostrar las dependencias de compilación de los ficheros de código, relaciones de derivación entre ficheros de código fuente y ficheros que son resultados de la compilación, dependencias entre elementos de implementación y los correspondientes elementos de diseños que son implementados (Jacobson, 2000).

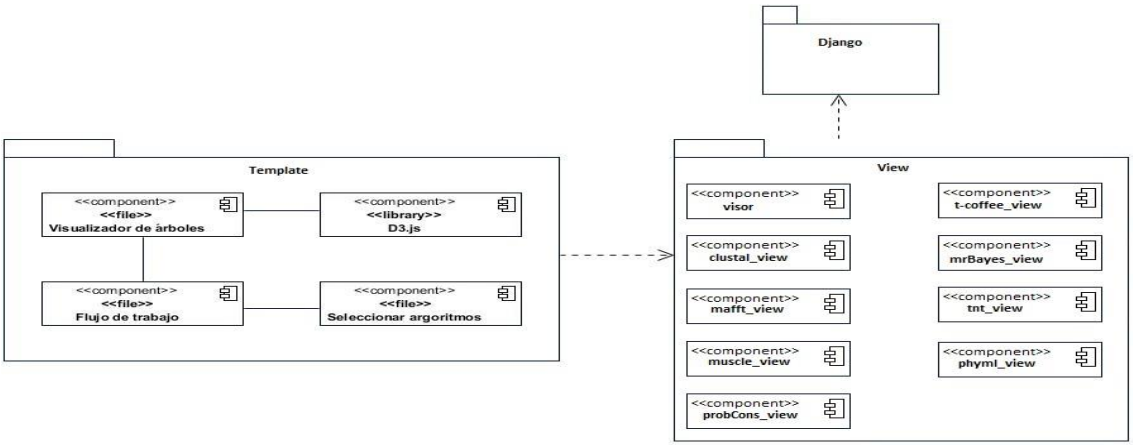


fig. 12 Diagrama de componentes

Para simplificar el desarrollo del sistema se utilizó la automatización del patrón utilizado en el framework Django: el patrón MVP. Donde los elementos del lado del cliente están alojados en el paquete Template, en este paquete se alojan 4 componentes, el seleccionar herramientas es el componente encargado de brindar la configuración al usuario de cada uno de las herramientas que conforman la plataforma, por su parte el flujo de trabajo brindará la posibilidad al usuario de escoger el orden en que ejecutará las herramientas sobre una misma cadena de secuencia. El visualizador de árboles brindará al usuario una forma fácil y sencilla de visualizar los resultados obtenidos, esto lo logrará a través de la forma radial o rectangular de los árboles, dando la posibilidad de editar ese árbol a su gusto con un variado panel de opciones dentro del cual se aloja, colapsar nodos, marcar descendientes, tamaño de nodo, escala vertical y otros tantos. El cuarto componente será el D3.js que es la librería JavaScript usada para codificar y desarrollar toda la amplia gama de opciones que brinda el propio visualizador de árboles.

3.2 Diagrama de despliegue

Un Diagrama de despliegue modela la arquitectura en tiempo de ejecución de un sistema. Esto muestra la configuración de los elementos de hardware (nodos) y muestra cómo los elementos y artefactos del software se trazan en esos nodos (Pressman, 2010).

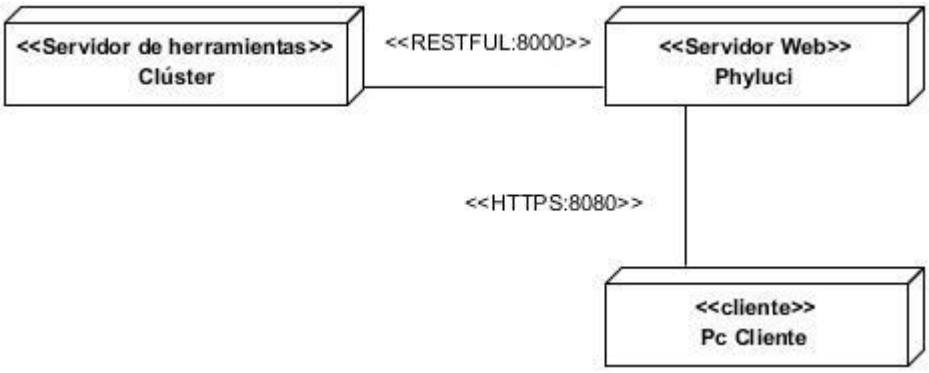


fig. 13 Diagrama de Despliegue

Pc Cliente: se refiera a las estaciones de trabajo que los investigadores utilizarán para acceder a PhylUCI y ejecutar sus tareas.

Servidor Web: servidor utilizado para establecer el FrontEnd de la plataforma de análisis filogenético, el cual logrará ejecutar dichos análisis a través de su comunicación mediante los servicios RESTFUL con el clúster de altas prestaciones. Este servidor también será el encargado del visualizador de árboles filogenéticos, el cual manejará toda la demanda que este conlleve. Para lograr la comunicación con las Pc Clientes se utilizará el protocolo de aplicación HTTPS, en si este servidor será la comunicación entre los investigadores y el BackEnd.

Servidor de herramientas: servidor de altas prestaciones utilizado para alojar todas las herramientas necesarias para realizar un completo análisis filogenético. Este servidor es el encargado de alojar el BackEnd de la plataforma web, módulo que recibirá las peticiones del FrontEnd y arrojará los resultados pertinentes.

HTTPS: protocolo seguro de transferencia de hipertexto, HTTPS por sus siglas en inglés (Hypertext Transfer Protocol Secure), es un protocolo de aplicación basado en el protocolo HTTP, destinado a la transferencia segura de Hipertexto (Chen, Wang, Wang, & Zhang, 2010).

RESTFUL: transferencia de estado representacional, REST por sus siglas en inglés, (Representacional State Transfer) es un estilo de arquitectura de software para sistemas hipermedia, se usa para describir cualquier interfaz entre sistemas que utilice directamente HTTP para obtener datos o indicar la ejecución de operaciones sobre los datos, en cualquier formato (XML, JSON) sin las abstracciones adicionales de los protocolos basados en patrones de intercambio de mensajes, como por ejemplo SOAP (Pautasso, 2014) .

3.3 Pruebas de software

Son una actividad en la cual un sistema o componente es ejecutado bajo unas condiciones o requisitos especificados, los resultados son observados y registrados, y una evaluación es hecha de algún aspecto del sistema o componente (Pressman, 2010).

Los Productos de Software, sistemas y/o aplicaciones son creadas, desarrolladas e implementadas por seres humanos y por ende en cualquiera de sus etapas de creación se puede presentar una equivocación, al generarse esa "Equivocación" se puede conllevar a un defecto en el software, por ejemplo, mala digitación, distracción al codificar, mala elaboración de un documento, entre otras. Si no se ha identificado ese defecto y el software o la aplicación se ejecuta, hay un alto riesgo de que la aplicación no haga lo que debería hacer o el objeto para lo cual fue creada, es decir se genera un fallo o desperfecto, lo que podría generar una catástrofe.

Estrategia de Pruebas

La estrategia de prueba describe el enfoque y los objetivos generales de las actividades de prueba. Incluye los niveles de prueba a ser realizados y el tipo de prueba a ser ejecutadas (Pressman, 2010).

Niveles de Pruebas

La Prueba es aplicada para diferentes tipos de objetivos, en diferentes escenarios o niveles de trabajo.

La estrategia define:

- Técnicas de pruebas y herramientas a ser usadas.
- Qué criterios de éxitos y culminación de la prueba serán usados.
- Consideraciones especiales afectadas por requisitos de recursos o que tengan implicaciones en la planificación.

Se distinguen los siguientes niveles de pruebas a realizar para evaluar en el sistema:

- Prueba de unidad.
- Prueba de sistema.
- Prueba de aceptación.
- Prueba de integración.

3.3.1 Prueba de Unidad

Al desarrollar un nuevo software o sistema de información, la primera etapa de pruebas a considerar es la etapa de pruebas unitarias o también llamada pruebas de caja blanca (White Box), estás pruebas también son llamadas pruebas modulares ya que nos permiten determinar si un sistema está listo y correctamente

terminado, estas pruebas no se deben confundir con las pruebas informales que realiza el programador mientras está desarrollando el sistema (Pressman, 2010).

Método de Caja Blanca

Se basa en el minucioso examen de los detalles procedimentales. Se comprueban los caminos lógicos del software proponiendo casos de prueba que examinen que están correctas todas las condiciones y/o bucles para determinar si el estado real coincide con el esperado o afirmado. Esto genera gran cantidad de caminos posibles por lo que hay que dedicar esfuerzos a la determinación de las condiciones de prueba que se van a verificar (Pressman, 2010).

Mediante la prueba de la caja blanca se puede obtener casos de prueba que:

- Garanticen que se ejerciten por lo menos una vez todos los caminos independientes de cada módulo, programa o método.
- Ejerciten todas las decisiones lógicas en las vertientes verdadera y falsa.
- Ejecuten todos los bucles en sus límites operacionales.
- Ejerciten las estructuras internas de datos para asegurar su validez.

Técnica de camino básico.

Es una prueba estructural que se deriva a partir del conocimiento de la estructura y la implementación del software. Tiene como objetivo principal examinar los caminos o ramificaciones posibles de flujo de la ejecución de una estructura de programa.

- A partir del diseño o del código fuente, se dibuja el grafo de flujo asociado.
- Se calcula la complejidad ciclomática del grafo.
- Se determina un conjunto básico de caminos independientes.
- Se preparan los casos de prueba que obliguen a la ejecución de cada camino del conjunto básico.

Para aplicar la técnica del camino básico se debe introducir una sencilla notación para la representación del flujo de control, el cual puede representarse por un Grafo de Flujo. Cada nodo del grafo corresponde a una o más sentencias de código fuente. Todo segmento de código de cualquier programa se puede traducir a un Grafo de Flujo (Pressman, 2010).

```
def clustalw_view(request):
    if request.method == 'POST':
        form = ClustalwForm(request.POST, request.FILES)
        if form.is_valid():

            aligment_type = request.POST.get['aligment_type']
            service.clustalw(form, aligment_type, request.FILES, request.user)
            return HttpResponseRedirect('/user_process/%s/' % request.user.username)
        else:
            return render(request, 'form/Alignements/clustalw.html', {'form': form})
    else:
        form = ClustalwForm()
        print 'Nultipart ', form.is_multipart()
        return render(request, 'form/Alignements/clustalw.html', {'form': form})
```

fig. 14 Fragmento de código a analizar

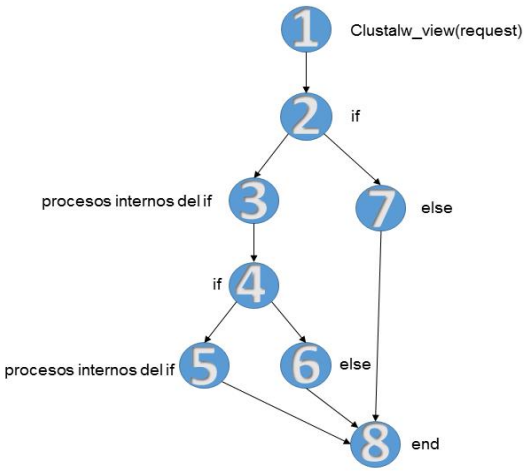


fig. 15 Representación del grafo de flujo

A partir de un grafo de control G= (N, V), donde N es un conjunto de nodos y V un conjunto de arcos entre los nodos, se calcula la complejidad ciclomática de varias maneras que resultan equivalentes. Una es más cómoda para el trabajo manual y las otras para el proceso automático (Pressman, 2010). A continuación, se definen las tres formas:

Definición 1: Sean a: número de arcos y n: número de nodos; entonces la complejidad ciclomática o número ciclomático se define como: v(G)= a – n + 2

Esta es la forma clásica de calcular la complejidad ciclomática, adecuada en programas que lo hacen en forma automática.

V(G)= a – n + 2

V(G)= 9 aristas – 8 nodos + 2

V(G)= 3

Definición 2: Sea nps el número de nodos con predicado simple (es decir, nodos de los que parten dos caminos); entonces la complejidad ciclomática o número ciclomático se define como:

v(G)= 1 + nps

Esta forma es muy útil, ya que permite omitir la construcción del grafo, centrándose únicamente en los predicados simples del programa. Note que se dice "predicados simples", es decir, supone que ya se expandieron los predicados múltiples.

V(G)= 1 + nps

V(G)= 1 + 2

V(G)= 3

Definición 3: Sea rr el número de regiones rodeadas completamente por arcos del grafo; entonces la complejidad ciclomática o número ciclomático se define como: v(G)= rr + 1.

Esta definición es de utilidad cuando se calcula manualmente la complejidad ciclomática, ya que se identifican las áreas cerradas en forma visual. En cálculo automatizado no es muy conveniente.

V(G)= 1 + rr

V(G)= 1 + 2

V(G)= 3

Derivación de casos de Prueba

Luego de tener elaborados los Grafos de Flujos y los caminos a recorrer, se preparan los casos de prueba que forzarán la ejecución de cada uno de esos caminos. Se escogen los datos de forma que las condiciones de los nodos predicados estén adecuadamente establecidas, con el fin de comprobar cada camino.

Caso de uso: Seleccionar Herramienta

Caso de prueba: Crear Clustalw

Camino: 1-2-3-4-5-8

Entrada: Para un valor valido del método POST y un valor válido de los campos del formulario se captura el tipo de alineamiento, y se crea el XML de configuración con los valores obtenidos del formulario.

Resultado: Se crea la tarea asociada a la herramienta Clustalw redireccionando a la vista de procesos.

Camino: 1-2-3-4-6-8

Entrada: Para un valor valido del método POST y un valor incorrecto en alguno de los campos del formulario.

Resultado: Se re direcciona a la vista de la herramienta Clustalw mostrando los errores en el formulario.

Camino: 1-2-7-8

Entrada: Para un valor incorrecto del método POST.

Resultado: Se crea un formulario con las opciones por defecto de la herramienta Clustalw y se re direcciona a la vista de la herramienta, enviando el formulario recién creado.

Luego de tener elaborados los Grafos de Flujos y los caminos a recorrer, se preparan los casos de prueba que forzarán la ejecución de cada uno de esos caminos. Se escogen los datos de forma que las condiciones de los nodos predicados estén adecuadamente establecidas, con el fin de comprobar cada camino.

Luego de confeccionados los casos de prueba se ejecutaron cada uno de estos y se compararon los resultados con los esperados. Una vez terminados todos los casos de prueba, se garantizó de que todas las sentencias del programa fueron ejecutadas por lo menos una vez.

3.3.2 Prueba de Sistema

Cualquier pieza de software completo, desarrollado o adquirido, puede verse como un sistema que debe probarse, ya sea para decidir acerca de su aceptación, para analizar defectos globales o para estudiar aspectos específicos de su comportamiento, tales como seguridad o rendimiento. A éste tipo de pruebas donde se estudia el producto completo se les llama Pruebas de Sistema (Jacobson, 2000).

Prueba de Caja Negra

Caja Negra este método se centra en los requisitos funcionales del software. O sea, la prueba de caja negra permite al ingeniero del software obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa (Pressman, 2010).

Casos de prueba

Los casos de pruebas se realizan con el objetivo de determinar que una funcionalidad ha sido implementada satisfaciendo las necesidades del cliente.

En la siguiente tabla se detallan las variables que se encuentran asociadas al caso de uso Registrar usuario.

Tabla 3. Variables para el caso de prueba

No	Nombre del Campo	Clasificación	Valor Nulo	Descripción
1	Usuario	Campo de texto	No	Cadena de texto con longitud mayor que 5 que inicia con letras, se pueden entrar hasta 10 letras ya sea mayúscula o minúscula y puede terminar en número, la longitud máxima es de 15.
2	Nombre	Campo de texto	No	Campo alfanumérico que permite una cadena de texto que contiene al iniciar las

				letras mayúsculas, van separados por espacio y tiene como longitud máxima 100 caracteres.
3	Apellidos	Campo de texto	No	Campo alfanumérico que permite una cadena de texto que contiene al iniciar las letras mayúsculas, van separados por espacio y tiene como longitud máxima 100 caracteres.
4	Correo	Campo de texto	No	Campo alfanumérico que permite una cadena de texto cumpliendo el estándar de correo: usuario@dominio.dominio.
5	Contraseña	Campo de texto	No	Campo alfanumérico con una combinación de mayúscula, minúsculas, números y caracteres especiales. Longitud mínima de 6 y máxima hasta 100.
6	Confirmar Contraseña	Campo de texto	No	Campo alfanumérico con una combinación de mayúscula, minúsculas, números y caracteres especiales. Longitud mínima de 8 y máxima hasta 100 caracteres. Debe coincidir con la contraseña indicada.

Esta descripción posibilito que se realizara un matriz de datos, donde se evaluó y probo la validez de los datos introducidos en el sistema. Utilizando juegos de datos válidos e inválidos se realizó la técnica de partición equivalente.

Partición equivalente

La partición equivalente es una técnica de prueba de caja negra que divide el dominio de entrada de un programa en clases de datos a partir de las cuales pueden derivarse casos de prueba. El diseño de casos de prueba para partición equivalente se basa en una evaluación de las clases de equivalencia para una condición de entrada (Pressman, 2010).

Tabla 4. Caso de prueba Registrar usuario

Escenario	Variables						Descripción	Respuesta del Sistema	Flujo Central
	1	2	3	4	5	6			
EC 1.1 Registrar Usuario	V	V	V	V	V	V	En este escenario se realiza el registro de un nuevo usuario al sistema correctamente.	Se registra correctamente el nuevo usuario y se muestra una mensaje de bienvenida.	1- Se llenan los campos correcta-mente 2- Se presiona el botón registrar.
EC 1.2 Registrar usuarios con campos incorrectos.	I	V	V	V	V	I	En este escenario se realiza el registro de un nuevo usuario al sistema con datos incorrectos.	El sistema valida que los datos sean correctos y muestra un mensaje de error indicando cuales campos son incorrectos.	1- Se llenan los campos incorrectamente. 2- Se presiona el botón registrar. 3- Se reinician los campos.
EC 1.3 Registrar usuarios con campos vacíos.		V	V	V	V	V	En este escenario se realiza el registro de un nuevo usuario al sistema con datos vacíos.	El sistema valida que los datos sean correctos y muestra un mensaje de error indicando el campo vacío.	1- Se llenan los campos incorrectamente 2- Se presiona el botón registrar 3- Se reinician los campos

Tras aplicar las pruebas funcionales mediante el método de Caja Negra, utilizando la técnica de partición equivalente se comprobó el correcto funcionamiento del sistema. Durante las tres iteraciones realizadas, se detectaron un total de catorce no conformidades, quedando todas resueltas como se evidencia en la fig. 17 y la tabla 5.

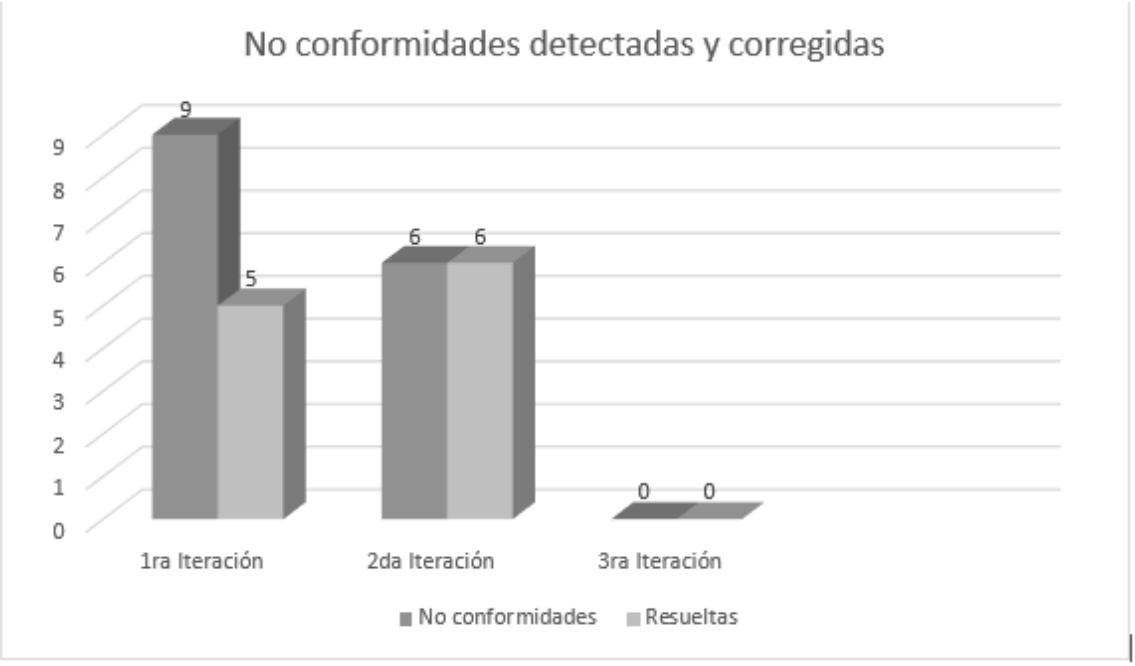


fig. 16 No Conformidades (NC) Detectadas y NC Corregidas en la técnica de partición equivalente.

Tabla 5. Descripción de las Iteraciones realizadas durante las pruebas.

No. Iteración	Cant.NC	Tipo Error	Impacto
1	9	3 Ortografía	Bajo
		1 Interfaz	Medio
		5 Validación	Alto
2	6	3 Validación 1 Interfaz 2 Ortografía	Alto Medio Bajo
3	0	-	-

3.3.3 Pruebas de Aceptación

Las pruebas de aceptación son aquellas que son diseñadas por el propio equipo de desarrollo en base a los requisitos funcionales especificados, y ejecutadas por el propio usuario de manera que este dé validez y conformidad al producto que se les está entregado en base a lo que se acordó inicialmente. De forma general las pruebas de aceptación pueden afrontarse mediante dos tipos de procedimiento para realizarlas: pruebas alfa y pruebas beta (Pressman, 2010).

Se decide aplicar las pruebas alfa ya que en ellas se le entrega a un usuario final todo el producto terminado, junto a su documentación correspondiente para que éste, en presencia del desarrollador y en entornos previamente preparados para el proceso de dichas pruebas, vaya informando de las inconsistencias y errores que detecte. El usuario final será el IPK el cual en conjunto con el líder del proyecto y los desarrolladores efectuaran las pruebas de aceptación, esta prueba se evidencia en los anexos.

3.3.4 Pruebas de Integración

La prueba de integración es una técnica sistemática para construir la estructura del programa mientras que, al mismo tiempo, se llevan a cabo pruebas para detectar errores asociados con la interacción. En este se verifica que módulos individuales son combinados y probados como un grupo. En una interfaz es posible perder datos, un módulo podría tener un efecto adverso e inadvertido sobre otro (Pressman, 2010).

Se decide aplicar como tipo de prueba las pruebas de integración ascendente, como su nombre lo indica, empieza la construcción y la prueba con módulos atómicos (es decir, componentes de los niveles más bajos de la estructura del programa). Debido a que los componentes se integran de abajo hacia arriba, siempre está disponible el procesamiento requerido para los componentes subordinados a un determinado nivel y se elimina la necesidad de resguardarlos.

Pasos para realizar la integración:

- Instalar el FrontEnd en un servidor web dedicado.
- Configurar en el settings.py la variable MEDIA_ROOT la cual tendrá la dirección donde se alojarán los archivos subidos por los investigares.
- Configurar las variables en settings.py destinadas a la comunicación con el BackEnd: IP_SERVER, PORT_SERVER, SERVICE_REGISTER, SERVICE_LOGIN, ROOT_ACCOUNT y SERCVICE_ACCOUNT_BY_NAME.
- Se debe crear un sistema de comunicación segura mediante ssh entre el BackEnd y el FrontEnd basado en llave pública y llave privada.
- Una vez realizado estos pasos el FrontEnd logra la integración al BackEnd.

Conclusiones del capítulo

En el desarrollo de este capítulo se realizó el modelo de implementación del sistema con el propósito de demostrar los componentes del sistema y sus relaciones, a través del diagrama de componentes. Se explicó a nivel de componentes en qué consistía cada uno de estos. También se realizó el diagrama de despliegue donde se detalló en qué consistía cada uno de sus nodos y conexiones. Para evaluar la calidad del sistema se validó la completitud de los requisitos con las pruebas funcionales al software, logrando un producto de calidad y listo para su explotación. Además, se comprobó que el FrontEnd para la plataforma de súper cálculo aplicada a estudios filogenéticos satisface las necesidades de los investigadores.

CONCLUSIONES

Una vez concluida la presente investigación se puede afirmar que se desarrolló satisfactoriamente el sistema PhylUCI. Durante su realización se arribaron a las siguientes conclusiones:

- Se analizaron las metodologías, herramientas y tecnologías más utilizadas a nivel internacional, con el objetivo de seleccionar las más adecuadas para el desarrollo del trabajo.
- En este trabajo se diseñó e implementó el FrontEnd para la plataforma PhylUCI, la misma cuenta con interfaces web para la edición de las configuraciones de las herramientas de cálculo filogenético alojadas en la plataforma, así como para la creación de flujos de trabajos y visualización del estado de las tareas en la plataforma. Se implementó un visualizador y editor de árboles, además de un sistema de autenticación mediante Tokens.
- Se realizaron las pruebas de Caja Blanca y Caja Negra identificando 24 no conformidades en 3 iteración, quedando todas resueltas en la última iteración.

RECOMENDACIONES

- Crear una vista para la Plataforma de Cálculo Distribuido T-arenal, que sea capaz de integrar a los flujos existentes la capacidad de ejecutar tareas sobre T-arenal.
- Crear una vista que permita realizar análisis filogenético usando la herramienta Beast.
- Permitir visualizar árboles en forma polar.
- Visualizar árboles en formato Fasta y Nexus.
- Internacionalizar el FrontEnd, al idioma inglés.

REFERENCIAS

1. Anglada Martínez, R. A., Ruíz Constanten, Y., & Torres Rubio, Y. (2013). Marco de trabajo para el desarrollo de herramientas orientadas a la gestión e integración de servicios telemáticos de infraestructura en GNU/Linux. *Revista Cubana de Ciencias Informáticas, 7*(2).

2. Balduino, R. (Agosto de 2007). Introduction to OpenUP. *OpenUp.pdf*. Recuperado el 9 de Mayo de 2017, de http://www. eclipse. org/epf/general

3. Bostock, M. (Agosto de 2011). *D3 Data Driven Documents*. Recuperado el 5 de Marzo de 2017, de d3js.org

4. Brudno, M., Mahabhashyam, M. S., B. Do, C., & Batzoglou, S. (2005). ProbCons: Probabilistic consistency-based multiple sequence alignement. *Genome Research*, 703-708.

5. Chen, S., Wang, R., Wang, X., & Zhang, K. (2010). Side-Channel Leaks in Web Applications: a Reality Today, a Challenge Tomorrow. Obtenido de https://www.microsoft.com/en-us/research/publication/side-channel-leaks-in-web-applications-a-reality-today-a-challenge-tomorrow/

6. Dereeper, A., Guignon, V., Blanc, G., Audic, S., Buffet, S., Chevenet, F., . . . Lescot, M. (2008). Phylogeny.fr: robust phylogenetic analysis for the non-specialist. (O. U. Press, Ed.) *Nucleic Acids Research, 36*, 465-469.

7. Edgar, R. C. (2004). MUSCLE: multiple sequence aligment with high accuracy and high throughput, Nucleic Acids Research. *Nucleic Acids Research, 32*(5), 1792-1797.

8. Galitz, W. O. (2007). *The Essential Guide to User Interface Design 2da Edicion.* Wiley Computer Publishing.

9. Goloboff, P. A., Farris, J. S., & Nixon, K. C. (2008). TNT, a free program for phylogenetic analysis. *Cladistics, 24*(5), 774-786.

10. Group, O. M. (2005). *Object Management Group*. Recuperado el 29 de Mayo de 2017, de UML 2.0, decurrent oficial version: http://www.uml.org/#UML2.0

11. Guindon S., J.F., D., V., L., M., A., W., H., & O., G. (2010). New algorithms and methods to estimate maximum-likelihood phylogenies: assessing the performance of PhyML 3.0. *Systematic biology, 59*(3), 307-321. Recuperado el 3 de Mayo de 2017, de www.atgc-montpellier.fr/phyml/

12. Huerta-Cepas, J., Serra, F., & Bork, P. (2016). ETE 3: Reconstruction, analysis, and visualization of phylogenomic data. *Molecular biology and evolution, 33*(6), 1635-1638.

13. IBM. (diciembre de 2009). *IBM Knowledge Center*. (IBM) Recuperado el 24 de octubre de 2016, de https://www-01.ibm.com/support/knowledgecenter/SSEPGG_8.2.0/com.ibm.db2.udb.doc/ad/c0009415.htm?lang=es

14. Jacobson, I. B. (2000). *El proceso unificado de desarrollo de software.* Addison Wesley.

15. Katoh, K., & Toh, H. (2008). Recent develoments int the MAFFT multiple sequence alignment program. *BRIEFINGS IN BIOINFORMATICS, 9*, 286-298.

16. Larkin, M. A., Blackshields, G., Brown, N., Chenna, R., McGettigan, P. A., McWilliam, H., . . . Lopez, R. (2007). Clustal W and Clustal X version 2.0. *bioinformatics, 23*(21), 2947-2948. Recuperado el 10 de Mayo de 2017, de SimGene.com/ClustalW

17. Larman, C. (2007). *UML y Patrones.* Editorial Prentice Hall.

18. Lemey, P., Salemi, M., & Vandamme, A.-M. (2009). *The Phylogenetic Handbook: A Practical Approach to Phylogenetic Analysis and Hypothesis Testing.* Cambridge University Press.

19. Letelier Torres, P., Penadés, M. C., Canós, J. H., & Sánchez López, E. A. (2003). Metodologías Ágiles en el Desarrollo de Software. *Metodologías Ágiles en el Desarrollo de Software, 1*(10), 1-8.

20. Misawa, K., Miyata, T., & Katoh, K. (2002). MAFFT: A novel method for rapid multiple sequence alignment based on fast Fourier transform. *Nucleic Acids Research*, 3059-3066.

21. MONDELO, P., GREGORI, E., & BARRAU, P. (2005). *Ergonomía 1 Fundamentos 3ra Edición.* Alfaomega, Ediciones UPC.

22. Myers, B., Hudson, S. E., & Pausch, R. (1 de Marzo de 2000). Past, present, and future of user interface software tools. (ACM, Ed.) *ACM Transactions on Computer-Human Interaction (TOCHI)*, 3-28. Recuperado el 11 de Mayo de 2017, de http://www. webopedia.com/TERM/G/Graphical_User_Interface_GUI.html

23. NotredameC, H., Higgins, D. G., & Heringa, J. (2000). T-Coffee: A novel method for fast and accuratemultiple sequence alignment. *JMB*, 205-2017.

24. Osmani, A. (2012). *Learning JavaScript Design Patterns: A JavaScript and jQuery Developer's Guide.* O'Reilly Media, Inc.

25. Palmer, G. (2008). *Java Event Handling.* Prentice Hall PTR.

26. Paradingm, V. (2013). *Visual paradigm for uml.* Visual Paradigm for UML-UML tool for software application development.

27. Pautasso, C. (2014). RESTful web services: principles, patterns, emerging technologies. En *Web Services Foundations* (págs. 31-51). Springer.

28. Pressman, R. (2010). *Ingenieria de Software un enfoque práctico.* McWraw-Hill Interamericana Editores S.A.

29. Rauschmayer, A. (2014). *Speaking JavaScript: An In-Depth Guide for Programmers.* O'Reilly Media, Inc.

30. Ronquist, F., Teslenko, M., Van Der Mark, P., Ayres, D. L., Darling, A., Larget, B., . . . Huelsenbeck, J. P. (2012). MrBayes 3.2: efficient Bayesian phylogenetic inference and model choice across a large model space. *Systematic biology, 61*(3), 539-542.

31. Sanchéz, R., Serra, F., Tarraga, J., Medina, I., Carbonell, J., Pulido, L., . . . Dopazo, H. (2011). Phylemon 2.0: a suite of web-tools for molecular evolution, phylogenetics, phylogenomics and hypotheses testing. *Nucleic Acids Research*, 470-474.

32. Shneiderman, B. (2010). *Designing the user interface: strategies for effective human-computer interaction.* Pearson Education India.

33. Sommerville, I. (2010). *Ingenieria del Software: Novena edición.* Addison-Wesley.

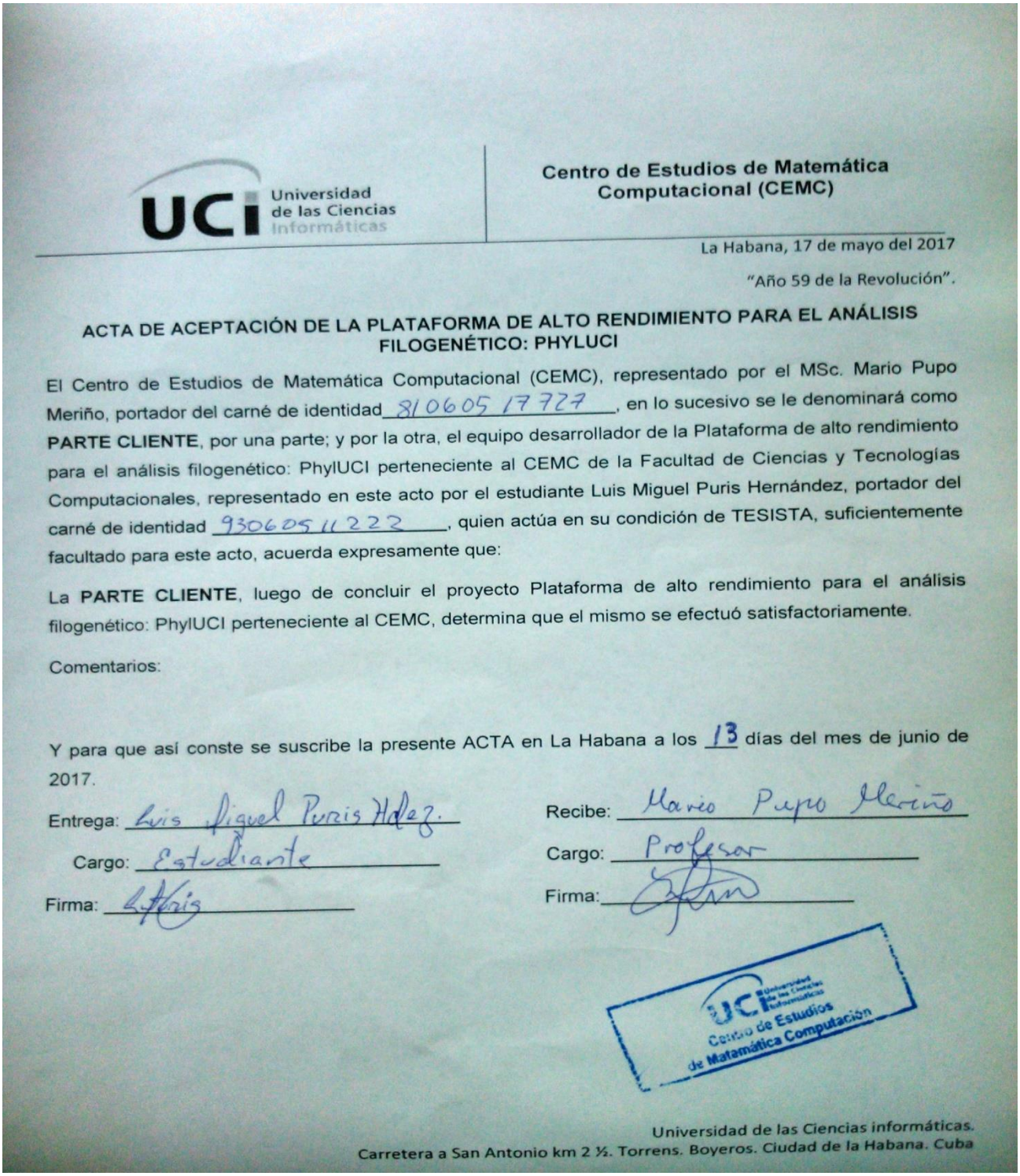
34. Stevens, T. J., & Boucher, W. (2015). *Python programming for biology.* Cambridge University Press.

35. Team, D. (2013). *Django documentation.*

36. Team, J. (2000). *JetBrains*. Recuperado el 12 de Marzo de 2017, de https://www.jetbrains.com/pycharm/

ANEXOS

Acta de aceptación.



UCI Universidad de las Ciencias Informáticas

Centro de Estudios de Matemática Computacional (CEMC)

La Habana, 17 de mayo del 2017

"Año 59 de la Revolución".

ACTA DE ACEPTACIÓN DE LA PLATAFORMA DE ALTO RENDIMIENTO PARA EL ANÁLISIS FILOGENÉTICO: PHYLUCI

El Centro de Estudios de Matemática Computacional (CEMC), representado por el MSc. Mario Pupo Meriño, portador del carné de identidad 810605 17727, en lo sucesivo se le denominará como **PARTE CLIENTE**, por una parte; y por la otra, el equipo desarrollador de la Plataforma de alto rendimiento para el análisis filogenético: PhylUCI perteneciente al CEMC de la Facultad de Ciencias y Tecnologías Computacionales, representado en este acto por el estudiante Luis Miguel Puris Hernández, portador del carné de identidad 93060511223, quien actúa en su condición de TESISTA, suficientemente facultado para este acto, acuerda expresamente que:

La **PARTE CLIENTE**, luego de concluir el proyecto Plataforma de alto rendimiento para el análisis filogenético: PhylUCI perteneciente al CEMC, determina que el mismo se efectuó satisfactoriamente.

Comentarios:

Y para que así conste se suscribe la presente ACTA en La Habana a los 13 días del mes de junio de 2017.

Entrega: Luis Miguel Puris Hdez.

Cargo: Estudiante

Firma:

Recibe: Mario Pupo Meriño

Cargo: Profesor

Firma:

UCI Universidad de las Ciencias Informáticas Centro de Estudios de Matemática Computación

Universidad de las Ciencias informáticas.
Carretera a San Antonio km 2 ½. Torrens. Boyeros. Ciudad de la Habana. Cuba