Universidad de las Ciencias Informáticas

Facultad 6

Título: "Plug-ins para la integración de los módulos de predicción de actividad biológica por Programación Genética y Árboles de Regresión a la plataforma alasGrato"

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autores: Yubisleydis Rodríguez Paredes.

Yania Uyoa Del Toro.

Tutores: Dr. Ramón Carrasco Velar.

Msc. Yania Molina Souto.

CIUDAD DE LA HABANA, CUBA

JUNIO, **2010**

“El futuro de Cuba tiene que ser necesariamente un futuro de hombres de ciencia, de hombres de pensamiento."

Fidel Castro Rúz

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Yubisleydis Rodríguez Paredes

_____

Firma del Autor

Yania Uyoa del Toro

_____

Firma del Autor

Dr. Ramón Carrasco Velar

_____

Firma del Tutor

Msc. Yania Molina Souto.

_____

Firma del Tutor

DATOS DE CONTACTO

Tutores:

Dr. Ramón Carrasco Velar

Universidad de las Ciencias Informáticas, Habana, Cuba.

E-mail: rcarrasco@uci.cu

Msc. Yania Molina Souto

Universidad de las Ciencias Informáticas, Habana, Cuba.

E-mail: ymolinas@uci.cu

AGRADECIMIENTOS

A mis padres por ayudarme siempre a seguir adelante en todo momento de mi vida y estar siempre ahí para mí cada vez que los necesité.

A mi hermana por darme siempre ánimo cuando pensaba que no terminaría nunca y por ser mi mejor amiga.

Agradecerle muy especialmente a la persona que estuvo conmigo todos estos años de mi carrera siempre a mi lado, dándome mucho apoyo y cuidando de mi cuando mis padres estaban lejos, Javi te quiero mucho.

A toda mi familia por apoyarme siempre.

A dos personas que se han portado muy bien conmigo y los considero como mis padres, Dulce y Javier.

A Claudia que es como una hermana para mí.

A nuestros tutores y especialmente a Yania por haberse comportado muy bien con nosotras y darnos mucho ánimo.

A unas personitas que sin su ayuda hubiera sido imposible terminar esta tesis: Edel, Luis Gabriel, Javier y Alejandro Cromeu.

A mi compañera Yubi, por compartir conmigo este trabajo y estar siempre a mi lado.

A Fidel y a la Revolución por darnos todo lo que tenemos y enseñarnos a ser mejores cada día.

A todos con los que he compartido estos años de la carrera y a los que de alguna forma nos ayudaron a terminar este trabajo.

Yania Uyoa Del Toro

AGRADECIMIENTOS

Agradezco a Dios por ser mi fortaleza, darme todo lo que tengo y no dejarme caer nunca.

A mis padres por ser mis guías, por haberme apoyado siempre, por su amor, dedicación y su fe incondicional en mí. Gracias a ellos he llegado hasta aquí y les estaré eternamente agradecida.

A mi hermanita por ser la personita más cariñosa del mundo conmigo, por su amor y confianza.

A mis abuelos que me educaron con amor, por su apoyo, su esfuerzo y por haber confiado siempre en mí.

A leo que en estos 5 años siempre fuiste mi ayuda, apoyo y fortaleza en cada momento. Pero sobre todo, porque viviste conmigo intensamente etapa a etapa de este trabajo y en los momentos más difíciles me aconsejaste y diste fuerzas para seguir adelante.

A todas mis tías y primos por su preocupación constante ,en especial a mi tía Iraida que es como una madre que siempre está ahí para ayudarme y aconsejarme.

A Yania por su amistad y por todos los momentos que hemos compartido juntas en estos 5 años.

A Edel, Luis, Javier y Alejandro por su permanente disposición y ayuda desinteresada.

A mis tutores, en especial a Yania que siempre estuvo ahí ayudándonos y quien en los momentos más difíciles siempre confío en nosotras y fue como una amiga que nos apoyaba y daba alientos para poder terminar este trabajo.

A todos mis compañeros de aula y de apartamento con los que he compartido en estos 5 años, mil gracias a todos por los momentos que hemos pasado juntos. De ustedes me llevo el mejor de los recuerdos.

A todos mis profes no solo los de la carrera sino a los de la vida, porque de alguna manera forman parte de lo que ahora soy.

Le agradezco a Fidel y a la Revolución por haberme dado la oportunidad de estudiar en esta universidad de excelencia y en estos momentos poder cumplir mi sueño y el de mis padres.

Yubisleydis Rodríguez Paredes.

DEDICATORIA

A mi mamita, por darme la vida.

A mi papi querido, por ser la luz de mi vida.

A mi hermanita (ia) por ser la mejor hermana del mundo.

A Javi porque lo es todo para mí.

A mi nueva familia, Dulce, Javier y Claudia.

A mis amigos.

A Fidel y a la Revolución.

Yania Uyoa Del Toro

A mis padres que son mi luz y mi mayor motivación.

A mi hermana que esa personita maravillosa que adoro y que nunca defraudaré.

A mis abuelos que son mis segundos padres.

A leo por su amor y paciencia.

Yubisleydis Rodríguez Paredes

RESUMEN

El Departamento de Bioinformática de la facultad 6 de la Universidad de las Ciencias Informáticas, desarrolla una plataforma para el modelado virtual de compuestos orgánicos. El sistema consta con un módulo para el cálculo de descriptores topológicos y topográficos, un visualizador y editor de estructuras químicas, un lenguaje descriptor y varios módulos de Inteligencia Artificial que se integrarán para el desarrollo de modelos SAR(Structure-Activity Relationship) y QSAR(Quantitative Structure-Activity Relationships),que incluyen Programación Genética, Árboles de Regresión, Lógica Difusa y Máquinas de Soporte Vectorial. Estos módulos son los encargados de predecir la actividad biológica en diferentes compuestos orgánicos.

El presente trabajo desarrolla los plug-ins para los módulos que utilizan las técnicas de Programación Genética y Árboles de Regresión, las cuales solo se reducían a un conjunto de clases agrupadas en librerías independientes de la plataforma. Dichos plugins se incorporaron al Front-End de Grato permitiendo al usuario generar modelos y predecir actividad biológica en diferentes moléculas.

PALABRAS CLAVE

- ✓ Plug-in.
- ✓ Programación Genética.
- ✓ Árboles de Regresión.
- ✓ Compuestos orgánicos.

TABLA DE CONTENIDOS

INTRODUCCIÓN

El proceso de investigación-desarrollo de un medicamento es largo y complejo, involucra grandes inversiones y se estima que las probabilidades de éxito al final del estudio son bajas. La mayoría de los compuestos que comienzan los estudios como posibles candidatos a medicamentos son desechados en cada una de las etapas del proceso, donde el tiempo promedio para encontrar un nuevo agente involucra entre 12 y 15 años con un costo de aproximadamente 600 millones de dólares. De cada 5000 moléculas que se analizan solo una termina el ciclo, supera las pruebas pre-clínicas y se comercializa en el mercado. [1]

Por esta razón la industria farmacéutica modificó sus investigaciones y a diferencia del pasado donde a partir de una dolencia se buscaba una posible medicina, en la actualidad se identifica el objetivo terapéutico y luego se hace una búsqueda de candidatos para este uso específico. El desarrollo en la informática y en la química computacional permite a los investigadores, de forma teórica, cuantificar propiedades de las moléculas y a partir de estos datos realizar tamizajes donde se evalúan decenas de miles de compuestos, almacenados en bases de datos científicas, que incluyen varias características.

Algunos de los enfoques, para realizar estos tamizajes, están basados en la interrelación estructura química-actividad biológica de las moléculas. Los primeros pasos en esa dirección se dieron con el surgimiento de las técnicas QSAR (Quantitative Structure-Activity Relationships), con el uso de este tipo de método se generan modelos que permiten predecir, en muestras que aún no se han analizado, la posibilidad de que un compuesto pueda contrarrestar determinada enfermedad. Dentro de los métodos más utilizados por los científicos para generar modelos QSAR se encuentran la Regresión Lineal Múltiple (RLM), los Mínimos Cuadrados Parciales y el Análisis de Componentes Principales. Sin embargo, estos métodos tienen grandes inconvenientes ya que presuponen que trabajan con datos que siguen una distribución normal y en otros casos se necesita de conocimientos matemáticos para seleccionar el modelo correcto. Dada esta situación una parte de las investigaciones se han centrado en establecer estas relaciones con el empleo de técnicas de ayuda a la toma de decisiones, obteniéndose resultados positivos.

En la actualidad existen un gran número de estas técnicas, el problema radica en saber distinguir cuál de ellas se adapta mejor a las condiciones del problema a resolver. En el caso de los estudios QSAR no solo se trata de predecir con exactitud el valor de los datos sino también de llegar a ecuaciones de regresión de las cuales se pueda inferir algún tipo de conocimiento. [2] Utilizando este paradigma se han construido herramientas que relacionan la estructura de los compuestos con cierta actividad en estudio. Este tipo de aplicación es comercializada a altos precios en el mercado internacional y en el peor de los casos cuando se realiza un estudio se necesita, de varias aplicaciones para completar todos los ciclos de la investigación.

Para los investigadores de Cuba, debido a la situación económica por la que se atraviesa, se les hace imposible obtener este tipo de aplicaciones, por lo que se hizo necesario desarrollar una plataforma, que haciendo uso de diferentes técnicas de Inteligencia Artificial, sea capaz de predecir cuáles serán las características físicas, químicas y farmacológicas de los posibles candidatos a fármacos, de esta forma, se reduce el campo de búsqueda y disminuyen tanto los costos como los plazos de ejecución del proceso de investigación-desarrollo de nuevos medicamentos. En este marco se crea el proyecto alasGrato, desarrollado en el Departamento de Bioinformática de la facultad 6 de la Universidad de las Ciencias Informáticas (UCI).

La plataforma tiene como principal objetivo cohesionar todas las funcionalidades que necesita un especialista para realizar un estudio completo de relación estructura-actividad. Hasta este momento cuenta con un módulo de cálculo de descriptores donde se calculan todos los datos necesarios para caracterizar la estructura molecular. Un editor y visualizador que permite observar y modificar las estructuras que están almacenadas en la base de datos y se desea incluir una nueva funcionalidad que permita generar modelos QSAR. Dentro del proyecto alasGrato se han desarrollado trabajos previos sobre relaciones estructura-actividad y se ha identificado que entre las técnicas de Inteligencia Artificial que más se destacan en estas investigaciones a nivel mundial se pueden mencionar las Redes Neuronales y las Máquinas de Soporte Vectorial, sin embargo, se demostró que otras menos utilizadas como la Programación Genética y los Árboles de Regresión generan modelos estructura-actividad con un alto grado de fiabilidad y a diferencia de las técnicas anteriores los modelos resultan más sencillos de

interpretar. Hasta el momento estos dos algoritmos se encuentran implementados de manera independiente a la plataforma reduciéndose a librerías en Java.

Debido a la necesidad de dar solución a la situación planteada se define como problema científico: *¿Cómo integrar los algoritmos de predicción de actividad biológica por Programación Genética y Árboles de Regresión a la plataforma alasGrato?*

Para enmarcar el problema que se identifica se define como **objeto de estudio**: Aplicaciones informáticas basadas en plug-ins. Para acotar este objeto de estudio se propone como **campo de acción:** Desarrollo de plug-ins para la plataforma alasGrato.

La plataforma alasGrato posee una arquitectura basada en plug-ins lo que hace del sistema una aplicación flexible, la cual puede ir aumentando sus funcionalidades gradualmente. Este tipo de diseño permite incorporar nuevos módulos que en la actualidad están en construcción, por esta razón se define como **objetivo general**: *Desarrollar los plug-ins para los módulos de Programación Genética y Árboles de Regresión de alasGrato.*

Para complementar este objetivo general se definen los siguientes **objetivos específicos**:

- ✓ Diseñar los plug-ins para predecir actividad biológica por Árboles de Regresión y Programación Genética.
- ✓ Implementar los plug-ins para predecir actividad biológica por Árboles de Regresión y Programación Genética.
- ✓ Realizar pruebas de validación.

Los objetivos específicos se desglosan para su cumplimiento en una serie de tareas que a lo largo de la investigación irán conformando los artefactos y herramientas finales. Las tareas que se identificaron se listan a continuación:

- ➢ Revisión del estado del arte sobre el desarrollo de plug-ins en Java.
- ➢ Selección de la herramienta manejadora de plug-ins a utilizar.

- Elaboración del modelo de dominio.
- Definición de los requisitos funcionales y no funcionales de la herramienta.
- Desarrollo del diagrama de casos de uso del sistema.
- Descripción de los casos de uso correspondientes al sistema.
- Desarrollo del diagrama de clases del diseño.
- Diseño del plug-in para la integración del módulo de predicción de actividad biológica por Árboles de Regresión.
- Implementación del plug-in para la integración del módulo de predicción de actividad biológica por Árboles de Regresión.
- Diseño del plug-in para la integración del módulo de predicción de actividad biológica por Programación Genética.
- Implementación del plug-in para la integración del módulo de predicción de actividad biológica por Programación Genética.
- Generación de modelos de predicción de actividad biológica.
- Realizar pruebas de validación.

La estructura del documento de investigación queda definida de la siguiente forma:

Capítulo 1: Fundamentación teórica.

En este capítulo se brinda una breve explicación sobre los métodos que se utilizan para el diseño de fármacos asistido por computadoras y se estudian algunas aplicaciones que permiten este tipo de estudios. Se analizan las principales herramientas manejadoras de plug-ins y finalmente se da una descripción de las herramientas y metodologías que se emplearán a lo largo de la investigación.

Capítulo 2: Características del sistema.

En este capítulo, mediante los componentes del modelo de dominio de la metodología OpenUP se describe la solución propuesta. Se aborda lo referente a los requisitos funcionales y no funcionales, se muestra el diagrama de casos de uso del sistema para ambos plug-ins desarrollados, así como la descripción de cada uno de sus casos de uso.

Capítulo 3: Análisis y Diseño del sistema.

En este capítulo se entra en el flujo de trabajo de Análisis y Diseño. En el Diseño se muestran los diagramas de clases del diseño por caso de uso y los diagramas de secuencia. Se describen los patrones de diseño utilizados así como el estilo arquitectónico que se utiliza.

Capítulo 4: Implementación y Prueba.

En este capítulo se describe cómo los elementos del modelo de diseño se implementan en términos de componentes, para esto se muestra el diagrama de componentes. Además, se le realizan las pruebas de caja negra a los plug-ins realizados.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

En el presente capítulo se brinda una breve explicación sobre las técnicas de Inteligencia Artificial empleadas para predecir actividad biológica. Se estudian algunas herramientas que permiten el diseño racional de fármacos asistido por computadoras. Así como se realiza un estudio de las herramientas y metodologías que se utilizarán para el desarrollo de los plug-in de Árboles de Regresión y Programación Genética.

1.1 La relación entre la estructura química y la actividad biológica de un compuesto.

Las relaciones que existen entre la actividad biológica y las características moleculares de un compuesto han sido objeto de estudio de muchas investigaciones en el campo de la Química Medicinal. A través de métodos matemáticos y de aplicaciones informáticas se tratan de establecer relaciones lineales entre parámetros físico-químicos conocidos como descriptores para explicar los valores de actividad de las sustancias.

La mayoría de los investigadores se apoyan para realizar sus análisis en la ecuación desarrollada por Crum-Brown y Frazer [3], que se observa en la Ecuación 1, donde ***A*** es la medida de actividad biológica y ***f(c)*** el conjunto de características que se utilizarán para describirla.

$$A = f(c)$$

Ecuación 1. Modelo de Crum-Brown y Frazer.

Siguiendo esta línea de investigación se ha ido transformando la Ecuación 1 y en la actualidad es muy común encontrar modelos de relación estructura actividad del tipo de la Ecuación 2.

$$AB = c_0 + \sum c_i x_j$$

Donde c_0 es el valor del intercepto de la función, x_j son los valores de descriptores o características moleculares con las que se correlacionará el valor de actividad biológica y c_j las contribuciones de cada una de estas características moleculares a la actividad.

El modelo lineal, desde el punto de vista matemático es muy sencillo y fácil de utilizar, por esta razón es el paso primario en este tipo de investigación. El modelo lineal clásico fue propuesto por Louis Hammett [4] y en este se correlacionan las constantes de velocidad de dos reacciones con parámetros electrónicos, estéricos y de resonancia del sustituyente, además de la constante de reacción. Basados en estos estudios más tarde Hansch y Fujita generan modelos lineales incluyendo parámetros hidrofóbicos.

Hansch y Fujita desarrollan durante la década de los 60 los primeros estudios QSAR, y en los años 80 se introduce el diseño computacional en el proceso de desarrollo de medicamentos debido al desarrollo teórico de técnicas de modelación molecular y a la aparición de ordenadores personales.

En los últimos años ha aumentado considerablemente la tendencia a utilizar técnicas de Inteligencia Artificial en vez de técnicas estadísticas. Se utilizan cientos de descriptores de la estructura química combinados con la aplicación de estas técnicas y se generan modelos diversos que ayudan a describir las características de generaciones de familias de compuestos químicos.

1.2 Algoritmos de optimización.

La búsqueda cuantitativa de la relación entre la estructura de un compuesto y su actividad biológica es un problema de regresión. La vía más sencilla para solucionar este problema es la Regresión Lineal Múltiple (RLM). Sin embargo, en este tipo de técnica estadística se tiene el inconveniente que se debe fijar a priori el comportamiento de los datos, y ya esto representa una dificultad cuando el tamaño de los datos es grande. Además se debe decir que las técnicas estadísticas no son capaces, por su naturaleza, de explorar todo el espacio de soluciones y encontrar la combinación óptima de variables que deben conformar el modelo. Este hecho impulsa a los investigadores a utilizar técnicas de ayuda a la toma de decisiones las cuales pueden introducir mejoras significativas en los modelos QSAR debido a que contienen variantes aleatorias que les permiten explorar una mayor área del espacio de búsqueda. [2]

En el caso de los Árboles de Regresión la principal ventaja radica en poder construir un árbol con varios puntos de ruptura y varios modelos lineales. Esta forma de representación en Química Medicinal le permite al investigador observar cómo puede comportarse el sistema para ciertos valores de una o varias variables a través de un modelo lineal y como se comporta de forma diferente para valores distintos.

En el caso de la Programación Genética es una técnica que en principio fue creada para resolver problemas de regresión. Es un algoritmo creativo que permite explorar el conjunto de funciones, desde las más simples hasta las más complejas y entre ellas elegir la mejor para ajustar los datos. No necesita que se le especifique la estructura del modelo, ni se necesitan conocimientos de estadística solo se entran tuplas de entrada-salida para construir la ecuación de regresión. Al igual que los Algoritmos Genéticos este paradigma evalúa varias soluciones a la vez y posee una componente aleatoria que le permite buscar en una gran parte del espacio de posibles soluciones.

En trabajos anteriores se investigó el desempeño de estas técnicas en estudios QSAR y se analizaron las posibilidades de ambas de construir modelos de regresión con calidad. Basados en aquellas investigaciones esta tesis construirá los plug-ins que serán integrados a la plataforma alasGrato y para ello en el siguiente epígrafe se explicará cómo funciona cada una de estas técnicas.

Programación Genética.

El hecho de que muchos problemas prácticos de diferentes dominios de aplicaciones puedan ser formulados como un problema de determinación de un "individuo solución" que produzca una salida deseada cuando se tienen presentes ciertas entradas particulares, hace a la Programación Genética una novedosa línea de investigación.

Esta técnica ha demostrado su capacidad para resolver problemas donde la relación entre las variables que participan es desconocida y una solución aproximada es aceptable; pero sobre todo en problemas donde son muy valoradas las posibles pequeñas mejoras obtenidas. [5]

El algoritmo general de la PG funciona creando una población inicial al azar de P programas compuestos por los símbolos terminales y no terminales (funciones). [6]

El algoritmo ejecuta los siguientes pasos:

1. Generar una población inicial de composiciones aleatorias de funciones y terminales del problema.

2. Ejecutar iterativamente los siguientes sub-pasos hasta que el criterio de terminación sea satisfecho:

2.1 Evaluar cada función en la población y asignarle un valor de aptitud de acuerdo a cuán bien solucione el problema.

2.2 Crear una nueva población de individuos aplicando las siguientes dos operaciones primarias. Las operaciones son aplicadas a los individuos en base a cierto criterio de selección.

2.2.1 Copiar programas de computadoras existentes en la nueva población.

2.2.2 Crear nuevos programas de computadora recombinando genéticamente partes aleatoriamente seleccionadas de los individuos ya existentes.

3. El mejor individuo que haya aparecido en cualquier generación es designado como el resultado de la Programación Genética. Este resultado puede ser una solución perfecta o aproximada al problema. [5]

Árboles de Regresión

Un árbol de regresión puede ser interpretado como una función de regresión construida a trozos, ya que las hojas de los árboles de regresión contienen valores numéricos o rectas de regresión [7].

Los árboles de regresión se diferencian de los árboles de decisión porque estos en vez de clases tienen valores o funciones. En los árboles de regresión conocidos como CART, el valor de predicción de cada hoja es el valor medio de todos los conjuntos de entrenamiento alcanzados por esa hoja. [8] Basándose en la idea de los algoritmos CART surge una nueva variante de árboles de regresión, M5.

Este último construye árboles cuyas hojas tienen funciones lineales, extendiendo la aplicación de los CART que es un caso particular de árboles donde las funciones que se consideran son constantes. Ambas estrategias construyen un primer árbol mediante un algoritmo de inducción de árboles de decisión, la diferencia radica en que en los árboles de decisión los puntos de corte se determinan maximizando en todo momento la ganancia de información, mientras que en los de regresión el corte se realiza maximizando la reducción de la varianza.

En el caso del algoritmo M5 construye un árbol dividiendo las instancias basándose en los valores de los atributos predictivos. Una vez que el árbol ha sido construido, el método genera un modelo lineal para cada nodo. Seguidamente las hojas del árbol comienzan a podarse mientras que el error disminuya, calculándose el error de cada nodo como el valor absoluto de la diferencia entre el valor predicho y el valor actual de cada ejemplo del conjunto de entrenamiento que alcanza dicho nodo. Este error es ponderado con un peso que representa el número de ejemplos que alcanza ese nodo, repitiéndose el proceso hasta que todos los ejemplos son cubiertos por una o más reglas [9].

1.3 Aplicaciones para realizar estudios QSAR.

Se han desarrollado varias aplicaciones para predecir actividad biológica en compuestos, unas con más prestaciones y exactitud que otras, pero todas, sin dudas con resultados prometedores. A continuación se relacionan varias de estas, que utilizan diferentes técnicas de Inteligencia Artificial, las cuales permiten que los especialistas predigan la posible actividad biológica de un compuesto o conjunto de compuestos.

Apex-3D

Es un sistema experto desarrollado para representar y utilizar conocimiento sobre relación estructura - actividad. Es utilizado para crear modelos 3D SAR y QSAR que luego son usados para la predicción y clasificación de la actividad biológica. Este sistema experto está insertado dentro del paquete de programas Insight II y tuvo su predecesor en el sistema experto OREX implementado sobre IBM PC 80286. Este último se basa en la descomposición topológica de las moléculas en fragmentos estructurales y su asociación a las actividades biológicas reportadas en una base de datos interna de alrededor de 15 000 compuestos. Emplea también la teoría lógico-combinatoria para el establecimiento de reglas de inferencia. [10]

ADAPT (Automated Data Analysis using Pattern Recognition Toolkit)

Es un sistema de programas que le permite al usuario el desarrollo de relaciones estructura-actividad y estructura-propiedad. Brinda la facilidad de entrada gráfica y almacenamiento de estructuras moleculares y sus datos asociados, generación de estructuras 3D, cálculo de descriptores moleculares y análisis de

estos empleando estadística multivariada, reconocimiento de patrones o redes de neuronas para construir modelos predictivos. Los enfoques estadísticos incluyen Regresión Lineal Múltiple, Análisis Clúster, Discriminante y Redes Neuronales. [11]

OncoLogic

Este programa ha sido desarrollado cooperativamente entre la EPA's Office of Pollution Prevention and Toxics (OPPT) y LogiCheminc, sobre PC tipo IBM-compatible DOS (no-Windows). Analiza las estructuras químicas para determinar la probabilidad de provocar cáncer mediante la aplicación de reglas utilizando métodos de modelación molecular y la incorporación de conocimientos sobre cómo las sustancias químicas causan cáncer en animales y humanos. Es el único sistema experto para predicción de cancinogenicidad que evalúa no sólo la estructura química sino también factores no estructurales, propiedades físicas y la ruta de la exposición. Aunque hace uso de cierto dato de la característica física dondequiera que esté disponible, no hace uso de los modelos QSAR en sus evaluaciones, y no puede calcular características fisicoquímicas para apoyar tales modelos. [12]

La mayoría de estas aplicaciones se desarrollan para solucionar problemas específicos que no cubren las necesidades de los investigadores, algunas se especializan en el cálculo de descriptores y otras utilizan una sola técnica para el desarrollo de los modelos. La intención de alasGrato es cubrir la mayor cantidad de funcionalidades para desarrollar este tipo de estudio y ofrecer varias técnicas de Inteligencia Artificial para que el usuario tenga la opción de elegir de acuerdo con sus necesidades.

1.4. Herramientas y metodologías a utilizar.

1.4.1. Herramientas Manejadoras de Plug-ins

Un plug-in es un módulo de hardware o software que añade una característica o un servicio específico a un sistema más grande, es decir, es una aplicación que puede incorporarse a otra para extender las funcionalidades de ese sistema. [13]

El uso de plug-ins se ha convertido, en la actualidad, en una tendencia para los desarrolladores de software a nivel mundial. Los grandes productores de aplicaciones informáticas desarrollan software que sean capaces de tener soporte para plug-in con el fin de ampliar su aplicación y convertirla en todo un éxito para el mundo digital.

Existen múltiples herramientas que permiten desarrollar plug-ins. Estas tienen su funcionalidad principal, pero el usuario les puede ir incorporando otras mediante plug-ins ya creados. A continuación se mencionan algunas de estas herramientas.

Eclipse

El entorno de desarrollo integrado (IDE) de Eclipse emplea plug-ins para proporcionar todas sus funcionalidades al frente de la plataforma de cliente rico, a diferencia de otros entornos monolíticos donde las funcionalidades están todas incluidas, las necesite el usuario o no. La arquitectura de plug-in le permite varias ventajas al Eclipse tales como: escribir cualquier extensión deseada en el ambiente, extenderse usando otros lenguajes de programación como son C/C++ y Python, trabajar con lenguajes para procesado de texto como LaTeX, aplicaciones en red como Telnet y Sistema de Gestión de Base de Datos. [14]

Desarrollar un plug-in para Eclipse es un poco complejo debido a que la curva de aprendizaje es muy lenta y demoraría un tiempo considerable.

NetBeans

Es un producto libre y gratuito, sin restricciones de uso. Soporta el desarrollo de todos los tipos de aplicación Java (J2SE, web, EJB y aplicaciones móviles). Entre sus características se encuentra un sistema de proyectos basado en control de versiones.
Todas las funciones del IDE son provistas por plug-ins. Cada plug-ins provee una función bien definida, tales como el soporte de Java, edición, o soporte para el sistema de control de versiones. Contiene todos los plug-ins necesarios para el desarrollo de aplicaciones Java en una sola descarga, permitiéndole al usuario comenzar a trabajar inmediatamente. [15]

Sin embargo, a pesar de ser un potente IDE de desarrollo, presenta algunas desventajas como por ejemplo, la curva de aprendizaje es un poco lenta, por lo que también, desarrollar un plug-ins para NetBeans implicaría un largo período de tiempo.

Front-End GRATO

Surgió como una necesidad del Proyecto alasGrato en el Departamento de Bioinformática de la facultad 6 de la Universidad de las Ciencias Informáticas para mejorar la interacción de los usuarios con la plataforma. Esta herramienta soluciona la principal dificultad que tenían las otras herramientas manejadoras de plug-ins antes mencionadas, o sea, la curva de aprendizaje lenta.

El Front-End GRATO es dinámico y multiplataforma, y se encarga de la portabilidad de los plug-ins garantizando:

- Unificar la carga de los plug-ins a través de descriptores XML.
- Ganar en soportes de escalabilidad para el ingreso de futuros plug-ins.
- Ganar en el manejo de memoria en ejecución controlando todos los componentes visuales desde un mismo marco.
- Mantener el principio de conservación, pues en la evolución del Front-End, los plug-ins desarrollados siempre serán compatibles. [16]

Debido a todo lo planteado se escogió al Front-End GRATO como la herramienta manejadora de plug-ins a utilizar para el desarrollo de los módulos de predicción de actividad biológica por Programación Genética y Árboles de Regresión de alasGrato. Teniendo en cuenta, además, que esta herramienta fue desarrollada específicamente para alasGrato, buscando más flexibilidad y facilitando la incorporación de funcionalidades futuras.

1.4.2 Metodologías de desarrollo de software

Las metodologías imponen un proceso disciplinado sobre el desarrollo de software con el fin de hacerlo más predecible y eficiente. Elegir la metodología adecuada es vital para lograr un sistema de alta calidad en un tiempo razonablemente corto. La metodología que se utilizará para guiar el desarrollo de las

funcionalidades de los plug-ins es OpenUP, ya que es la escogida y utilizada en el desarrollo de la plataforma alasGrato.

OpenUP

Es un proceso unificado, abierto, mínimo y suficiente, lo que brinda la posibilidad de sólo incluir el contenido fundamental y necesario. Conserva las características principales de la metodología de desarrollo RUP por lo que se aplica de forma iterativa e incremental, provee los componentes básicos que pueden servir de base a procesos específicos a pesar de no tener lineamientos para todos los elementos que se manejan en un proyecto.

Es una forma de desarrollo ágil y ligero donde la mayoría de sus elementos fomentan el intercambio entre los equipos de desarrollo, sincronizando intereses y compartiendo conocimientos. Permite a los desarrolladores obtener una solución que cumpla con los requerimientos del proyecto, enfocándose en la arquitectura y facilitando la retroalimentación y mejoramiento continuo. [17]

1.4.3 Herramientas CASE (Computer Aided Software Engineering)

Las 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 dinero. Son muy útiles, ya que permiten el desarrollo del software en diferentes tareas, tales como: el proceso de realizar un diseño del proyecto, cálculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática y documentación o detección de errores.

Como ejemplo de herramientas CASE se tiene Magic Draw, Rational Rose, Umbrello, Visual Paradigm, ArgoUML y otras. La herramienta CASE a utilizar para el desarrollo de las aplicaciones es Visual Paradigm, debido a que es multiplataforma y por las facilidades que brinda para el diseño de los diagramas necesarios para el desarrollo del software.

Visual Paradigm

Es una herramienta CASE para visualizar y diseñar elementos de software. Sirve para realizar modelado UML y tiene características gráficas cómodas que facilitan la realización de diagramas de clases, casos de

uso, comunicación, secuencia, estado, actividad, componentes, etc. Permite generar código y realizar ingeniería inversa para varios lenguajes de programación. [18]Facilita la interoperabilidad con otras herramientas CASE como el Rational Rose y se integra con las siguientes herramientas Java: Eclipse/IBM WebSphere, JBuilder, NetBeans IDE, Oracle, BEA Weblogic. Está disponible en varias ediciones: Enterprise, Professional, Community, Standard, Modeler y Personal.

1.4.4 Lenguaje de Modelado

UML (Unified Modeling Language)

Para dar solución al problema planteado se usa como lenguaje de modelado UML con el cual se puede especificar, construir, visualizar y documentar los artefactos resultantes de cada flujo de trabajo de un sistema de software orientado a objetos. Dicho lenguaje es una notación unificada con la que se permite lograr un entendimiento que propicie el intercambio entre los usuarios y los desarrolladores. El UML estándar está compuesto por tres partes: bloques de construcción (como clases, objetos, mensajes), relaciones entre los bloques (como asociación, generalización) y diagramas (por ejemplo, diagrama de actividad).

1.4.5 Lenguaje de Programación

JAVA

Es un lenguaje de programación de alto nivel desarrollado por Sun Microsystems a principios de los años 90. Es orientado a objetos, ya que soporta las tres características propias del paradigma de la orientación a objetos: encapsulación, herencia y polimorfismo. Permite realizar verificaciones en busca de problemas tanto en tiempo de compilación como en tiempo de ejecución lo que hace de este un lenguaje robusto. Cuando se utiliza Java no hay necesidad de preocuparse por la seguridad, ya que la máquina virtual, al ejecutar el código Java, realiza comprobaciones de seguridad. Es multiplataforma lo que implica que el mismo código que funciona en un sistema operativo, funcionará en cualquier otro sistema operativo que tenga instalada la máquina virtual Java. Al ser multihilos permite ejecutar muchas actividades simultáneas en un programa proporcionando de esta manera un mejor rendimiento interactivo y mejor comportamiento

en tiempo real. Proporciona las librerías y herramientas para que los programas puedan ser distribuidos, es decir, que puedan ser ejecutados en varias máquinas, interactuando. [19]

1.4.6 Herramienta de Desarrollo.

Para el desarrollo de una aplicación es necesario escoger la herramienta de desarrollo a utilizar según el lenguaje de programación que se seleccionó. La plataforma alasGrato está desarrollada sobre Eclipse, y por está razón los módulos en desarrollo y en particular los de esta tesis se desarrollan sobre el mismo IDE. A continuación se detallan las ventajas de esta herramienta, en las cuales se basó el proyecto para su elección.

Eclipse 3.4

Es un entorno de desarrollo integrado, de código abierto, multiplataforma y desarrollado originalmente por IBM, como el sucesor de su familia de herramientas para VisualAge. Esta herramienta provee al programador, de marcos de trabajo muy ricos, para el desarrollo de aplicaciones gráficas, definición y manipulación de modelos de software, aplicaciones web, etc. Por ejemplo, GEF (Graphic Editing Framework para la edición gráfica) es un plug-in de Eclipse para el desarrollo de editores visuales que pueden ir desde procesadores de texto hasta editores de diagramas UML, interfaces gráficas para el usuario (GUI), etc. Dado que los editores realizados con GEF "viven" dentro de Eclipse, además de poder ser usados conjuntamente con otros plug-ins, hacen uso de su interfaz gráfica personalizable y profesional. [14]

1.4.7 Lenguaje de Marcado

Extensible Markup Language (XML)

El lenguaje de marcas extensible (XML), es un metalenguaje extensible de etiquetas desarrollado por el W3C (World Wide Web Consortium.) Es una simplificación y adaptación del SGML (Standard Generalized Markup Language) y permite definir la gramática de lenguajes específicos. Por lo tanto, XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades. Es

una tecnología sencilla que tiene a su alrededor otras que la complementan y la hacen mucho más grande y con unas posibilidades mucho mayores. [20] Posee un papel muy importante en la actualidad ya que permite la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil. Algunos de los lenguajes que usan XML para su definición son XHTML, SVG, MathML.

1.5 Conclusiones Parciales.

En este capítulo se hizo un estudio sobre métodos que se utilizan para predecir actividad biológica y se estudiaron algunas herramientas que permiten este tipo de estudios. Además, al realizarse un análisis de las diferentes herramientas manejadoras de plug-ins, se decidió escoger para el desarrollo de los plug-ins de Programación Genética y Árboles de Regresión de alasGrato al Front-End GRATO. Además se seleccionaron las siguientes herramientas y metodologías: como metodología de desarrollo de software OpenUP, como herramienta CASE el Visual Paradigm, lenguaje de modelado UML, lenguaje de marcado XML, lenguaje de programación Java y herramienta de desarrollo Eclipse 3.4.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

En este capítulo para lograr una mayor comprensión y descripción de las clases más importantes en el sistema se realiza el modelo de dominio. Además, se describen los requisitos funcionales y no funcionales del sistema, los actores que intervienen en el mismo, así como cada uno de los casos de uso y la descripción textual de los mismos.

2.1 Definición de Modelo de Dominio.

Un modelo de dominio en la solución de problemas en la ingeniería de software, puede ser considerado como un modelo conceptual de un sistema que describe las diversas entidades que participan en este y sus relaciones. Tiene como objetivo fundamental la comprensión y descripción de las clases más importantes del sistema.

Debido a la poca estructuración de los procesos del negocio que tienen que ver con el objeto de estudio y para poder entender el contexto en que se emplaza el sistema se describe el funcionamiento de la aplicación mediante una serie de conceptos, entidades y sus relaciones, agrupándolos en un modelo de dominio.

2.1.1 Glosario de términos para el dominio

Usuario: persona que se encarga de generar los modelos matemáticos y de predecir la actividad biológica.

Modelo: es la función matemática que describe la actividad biológica.

Predicción: acción que el sistema realiza para emitir un resultado solicitado por el especialista.

Molécula: partícula formada por un conjunto de átomos ligados por enlaces covalentes.

Concentración Efectiva: actividad biológica (**AB**) asociada a un fragmento molecular, está dada por la expresión ***AB = log 1/C*** donde ***C*** es la concentración del fragmento.

Valores de Descriptores: información sobre la estructura molecular de una molécula.

2.1.2 Modelo de dominio

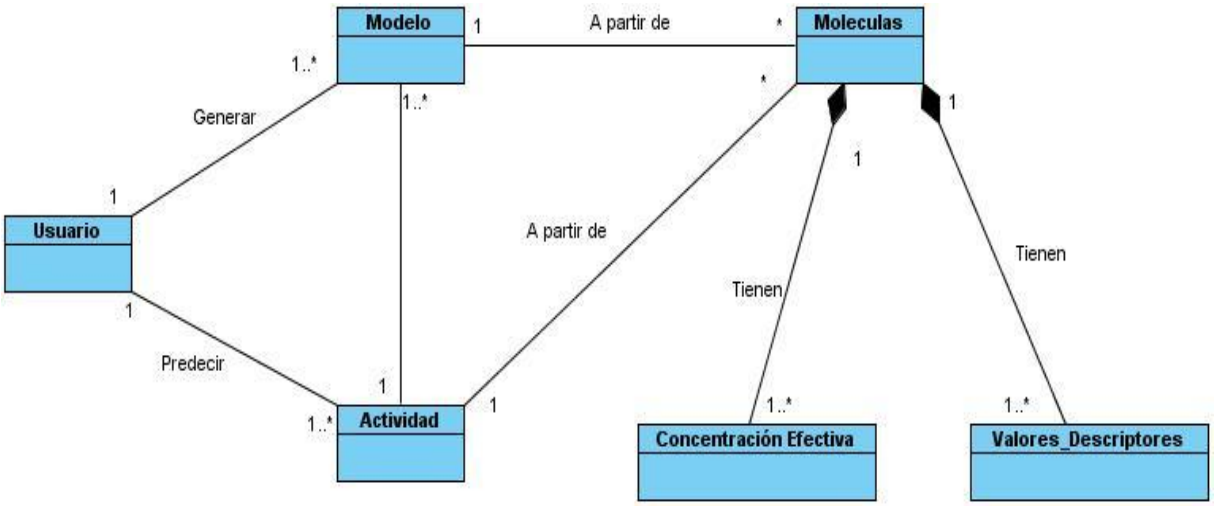


Figura 1: Modelo de dominio**.**

2.2 Requisitos no funcionales.

-Apariencia o interfaz externa

- ✓ El sistema debe contar con una interfaz amigable, fácil de comprender y usar, donde el usuario pueda orientarse fácilmente.

-Requisito de funcionalidad

- ✓ El sistema debe someterse a una etapa de adiestramiento en la que los usuarios se familiaricen con la aplicación y sean detectados los posibles errores, o puedan surgir posibles cambios en la interfaz.

-Software:

- ✓ Se debe disponer de sistemas operativos Linux, Windows 95 o superior para la instalación de la aplicación.
- ✓ Debe tenerse instalado el Java Runtime Environment (JRE) versión 1.5 o superior.

-Hardware:

Se requieren máquinas con los siguientes requisitos:

- ✓ Procesador Pentium 3 o superior.
- ✓ 256 Mb de RAM.

- ✓ 50 Mb de capacidad del disco duro.

-Usabilidad:

- ✓ El sistema le ofrecerá al usuario la posibilidad de realizar predicciones y analizar los resultados de las mismas, en este sentido se centra el diseño de la aplicación y en específico de las interfaces que harán posible el intercambio de datos de manera que le resulte al usuario de fácil entendimiento.

-Rendimiento:

- ✓ La herramienta propuesta debe ser rápida y el tiempo de respuesta debe ser el mínimo posible, adecuado a la rapidez con que el cliente requiere la respuesta a su acción.

-Soporte:

- ✓ El sistema debe permitir la interacción con los demás módulos que componen la plataforma en caso que esto sea necesario.
- ✓ Terminado el software se prestarán los servicios de instalación y configuración de la aplicación.

-Portabilidad:

- ✓ El sistema deberá funcionar en cualquier sistema operativo sobre el cual se haya instalado la máquina virtual de Java 1.5 o superior.

-Confiabilidad:

- ✓ El sistema no debe permitir que existan fallos, pero de ocurrir se debe garantizar que la pérdida de información sea mínima.

2.3 Requisitos funcionales.

Los requerimientos funcionales son capacidades o condiciones que el sistema debe cumplir.

2.3.1 Requisitos funcionales para el Plug-in de Árboles de Regresión.

R1: Convertir fichero .arff en .dat.

R2: Crear directorios (datasets, results, scripts).

R3: Cargar fichero config.

R4: Mostrar el modelo matemático.

R5: Guardar el modelo matemático.

R6: Cargar el modelo matemático.

R7: Cargar el fichero ARFF.

R8: Mostrar resultados de la predicción.

R9: Guardar los resultados de la predicción

2.3.2 Requisitos funcionales para el Plug-in de Programación Genética.

R1: Cargar fichero ARFF.

R2: Mostrar fichero ARFF

R3: Entrar Parâmetros.

R4: Mostrar el modelo matemático.

R5: Guardar el modelo matemático.

R6: Cargar el modelo matemático.

R7: Cargar el fichero ARFF.

R8: Mostrar resultados de la predicción.

R9: Guardar los resultados de la predicción

2.4 Actores del sistema.

Un actor es una entidad externa del sistema que de alguna manera participa en la historia del caso de uso. Por lo general estimula el sistema con eventos de entradas o recibe algo de él.

Nombre del actor	Descripción
Usuario	Es la persona que interactúa con el sistema ya sea con el objetivo de generar el modelo matemático o de predecir la actividad.

Tabla 1. Descripción de los actores del sistema

2.5 Diagrama de casos de uso del sistema.

Un diagrama de casos de uso del sistema representa gráficamente a los procesos y su interacción con los actores.

2.5.1. Diagrama de casos de uso del sistema para el plug-in de Árboles de Regresión.

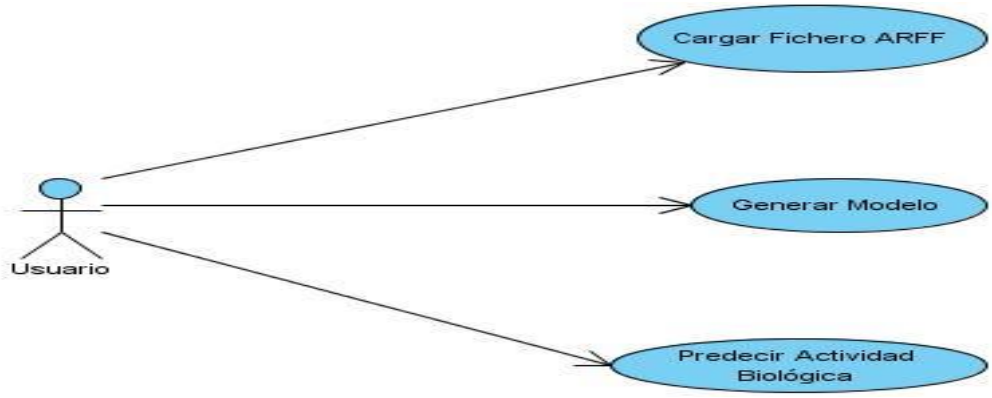


Figura 2: Diagrama de casos de uso para el plug-in de Árboles de Regresión

2.5.2. Diagrama de casos de uso del sistema para el plug-in de Programación Genética.

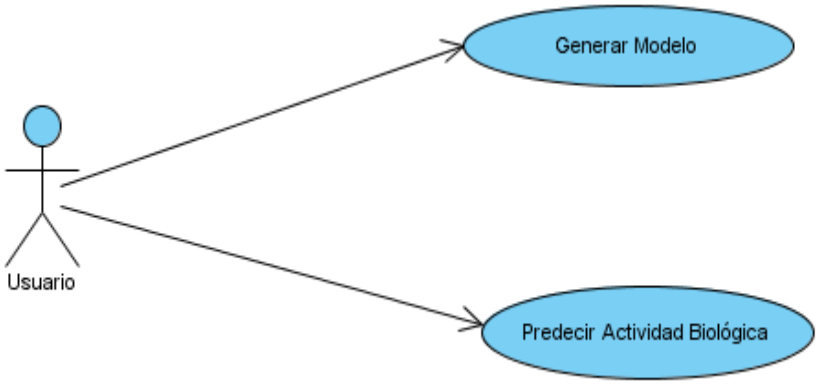


Figura 3: Diagrama de casos de uso para el plug-in de Programación Genética

2.6 Descripción de los casos de uso para el plug-in de Árboles de Regresión.

En esta sección se describe el Caso de Uso Cargar fichero ARFF del plug-in para el algoritmo de Árboles de Regresión. La descripción de los casos de usos Generar Modelo y Predecir Actividad Biológica se encuentran en el documento complementario: Descripciones de los casos de uso.

Caso de Uso	Cargar fichero ARFF
Actor:	Usuario
Resumen:	El Caso de Uso se inicia cuando el usuario selecciona la opción Cargar Fichero en el menú principal del sistema, carga los ficheros ARFF convirtiéndolos en .dat y se crea el directorio con las carpetas datasets, results, scripts.
Referencias:	R1, R2
Prioridad:	Crítico

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El usuario selecciona la opción Cargar Fichero.	2. El sistema muestra una ventana para especificar el lugar donde se cargará el fichero ARFF.
3. El usuario carga el ARFF	4. El sistema convierte el fichero ARFF en dos .dat (uno para el entrenamiento y otro para prueba) 5. Se crean las carpetas datasets, results, scripts.

	6. El sistema muestra el siguiente mensaje: "El ARFF fue convertido en .dat y ya están creadas las carpetas necesarias para poder generar el modelo".
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	4.1 El sistema muestra un mensaje de error y termina el caso de uso.
Poscondiciones:	Se convierte el fichero ARFF en dos .dat (uno para el entrenamiento y otro para prueba), se crea el directorio con las carpetas datasets, results, scripts y se guarda en datasets los .dat.

2.7 Descripción de los casos de uso para el plug-in de Programación Genética.

En esta sección se describe el Caso de Uso Generar Modelo del plug-in para el algoritmo de Programación Genética. La descripción del caso de uso Predecir Actividad Biológica se encuentra en el documento complementario: Descripciones de los casos de uso.

Caso de Uso	**Generar Modelo**
Actor:	Usuario
Resumen:	El Caso de Uso se inicia cuando el usuario selecciona la opción Cargar ARFF donde se carga el ARFF al con el cual se va a generar el modelo, seguidamente entra los parámetros necesarios y se genera el modelo matemático.

Precondiciones:	• Se debe haber cargado el fichero ARFF. • Se deben haber llenado los parámetros.
Referencias:	R1, R2,R3, R4,R5
Prioridad:	Crítico
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El usuario selecciona la opción Cargar Fichero.	2. El sistema muestra una ventana para especificar el lugar de dónde se cargará el fichero ARFF.
3. El usuario selecciona el fichero ARFF que desea cargar.	4. El sistema muestra el ARFF cargado.
5. El usuario selecciona la opción Entrar Parámetros.	6. El sistema muestra en pantalla los parámetros que el usuario debe llenar.
7. El usuario selecciona la opción Crear Modelo.	8. El sistema muestra en pantalla el modelo matemático creado.
9. El usuario selecciona la opción Guardar.	10. El sistema muestra una ventana para que el usuario especifique el lugar donde desea guardar el modelo

	matemático. 11. Se guarda el modelo y finaliza el caso de uso.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	4.1 El sistema muestra un mensaje de error y pasa a la acción 5 del flujo normal de eventos.
	8.1 El sistema muestra un mensaje de error y termina el caso de uso.
Poscondiciones:	Se creará el modelo matemático necesario para predecir actividad biológica.

2.8 Conclusiones Parciales.

En este capítulo se analiza la solución propuesta basada en el modelo del dominio brindándose una explicación de los términos utilizados en el mismo. Se definen los requisitos funcionales y no funcionales que deben cumplir los plug-ins desarrollados. También se definieron los actores y casos de uso del sistema brindándose una descripción textual de todos los casos de uso, lo cual ayuda a sentar las bases para las restantes fases del proceso de diseño e implementación.

CAPÍTULO 3: ANÁLISIS Y DISEÑO DEL SISTEMA

En este capítulo, a través del diseño del sistema que se está realizando, se profundizará en los casos de usos especificándolos de manera que permitan reflejar una vista interna del sistema descrita en el lenguaje de los desarrolladores. En esta vista interna se detallarán mejor los casos de uso y se determinarán las clases necesarias para llevar a cabo las funcionalidades en ellos contenidas. Tomándose en cuenta, a la hora de realizar el diseño, el lenguaje de programación a usar, la plataforma en la que se ejecutará la aplicación, entre otras características que afectan al sistema.

3.1 Diseño

El flujo de trabajo de Diseño tiene el propósito de formular los modelos que se centran en los requisitos no funcionales, centrando las bases para entrar en el flujo de Implementación y Prueba del sistema.

Durante este flujo se analiza si es posible dar una solución que satisfaga a los requerimientos significativos de la arquitectura describiendo los subsistemas y los componentes de un sistema informático y las relaciones entre ellos.

3.1.1 Diagramas de secuencias del diseño.

Un diagrama de secuencia muestra las interacciones entre objetos ordenadas en secuencia temporal, los objetos que se encuentran en el escenario y la secuencia de mensajes intercambiados entre los mismos para llevar a cabo la funcionalidad descrita por el escenario.

Se construyó un diagrama de secuencia para cada uno de los casos de uso de ambos plug-ins, en el caso de los que fueran muy grandes para lograr una mejor comprensión se realizó un diagrama de secuencia para cada escenario del caso de uso, donde intervienen los actores del caso de uso, los objetos que representan al sistema, y los eventos que envía cada actor al sistema.

Diagramas de secuencias del diseño para el plug-in de Árboles de Regresión.

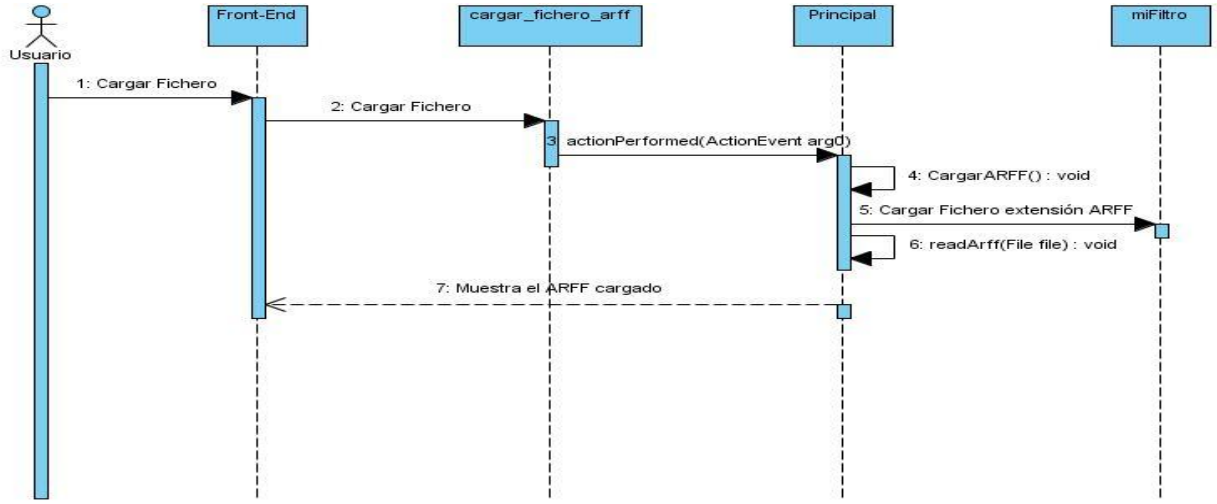


Figura 4.Diagrama de secuencia para el caso de uso Cargar fichero ARFF.

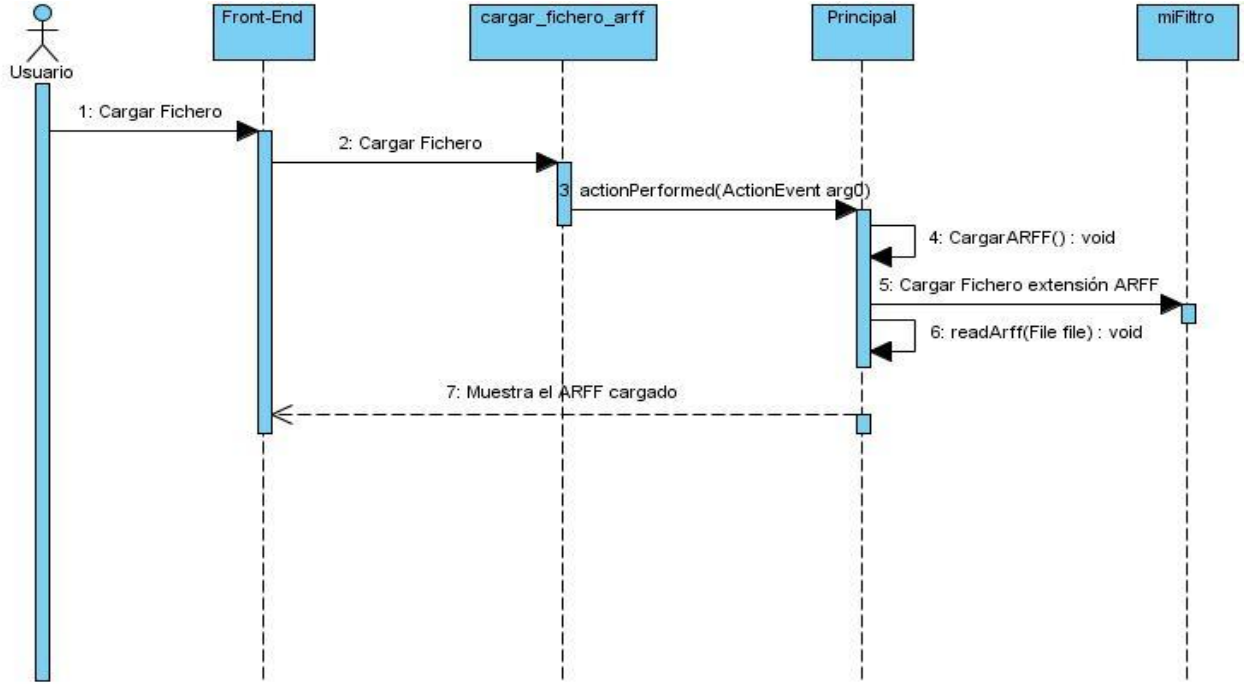


Figura 5.Diagrama de secuencia para el escenario Cargar Fichero del caso de uso Predecir Actividad Biológica.

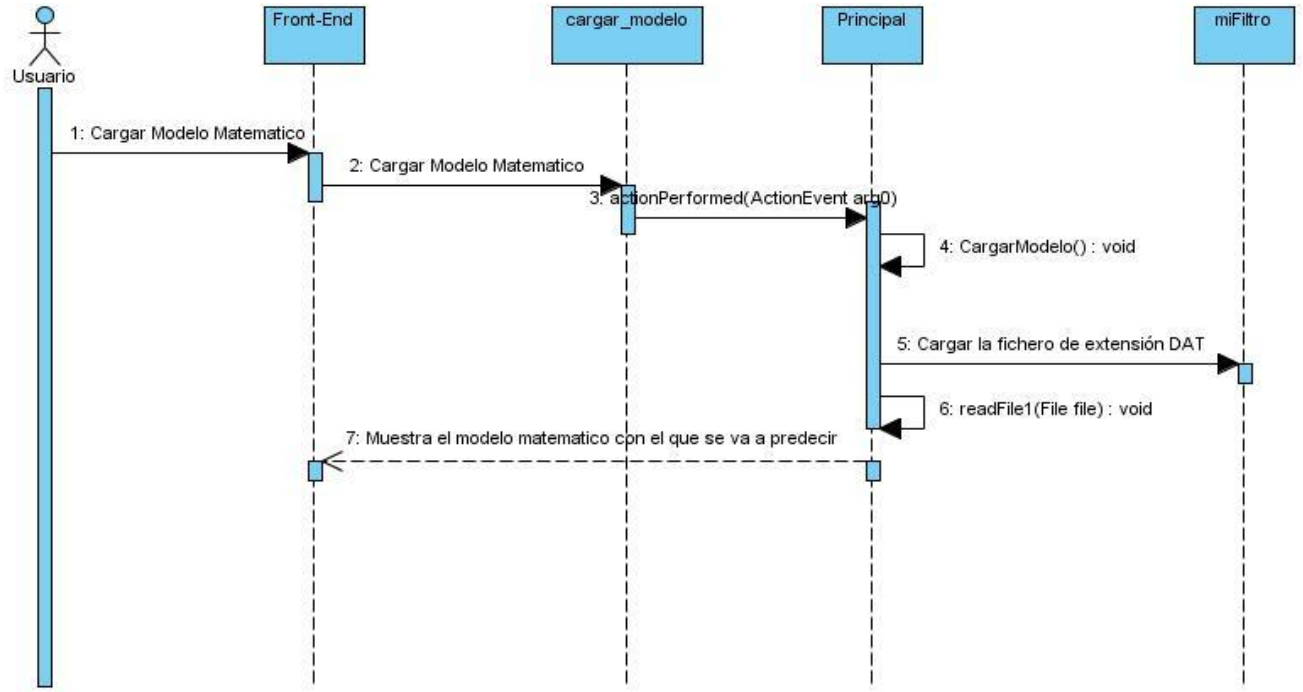


Figura 6.Diagrama de secuencia para el escenario Cargar Modelo del caso de uso Predecir Actividad Biológica.

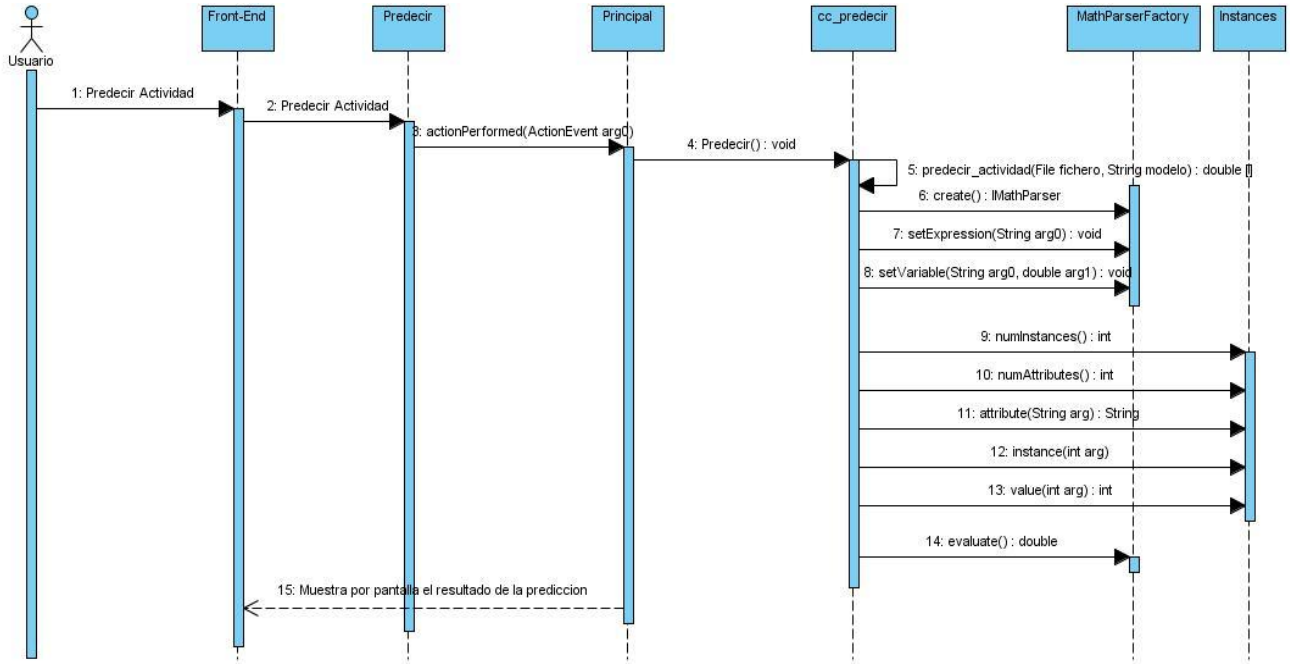


Figura 7.Diagrama de secuencia para el escenario Predecir del caso de uso Predecir Actividad Biológica.

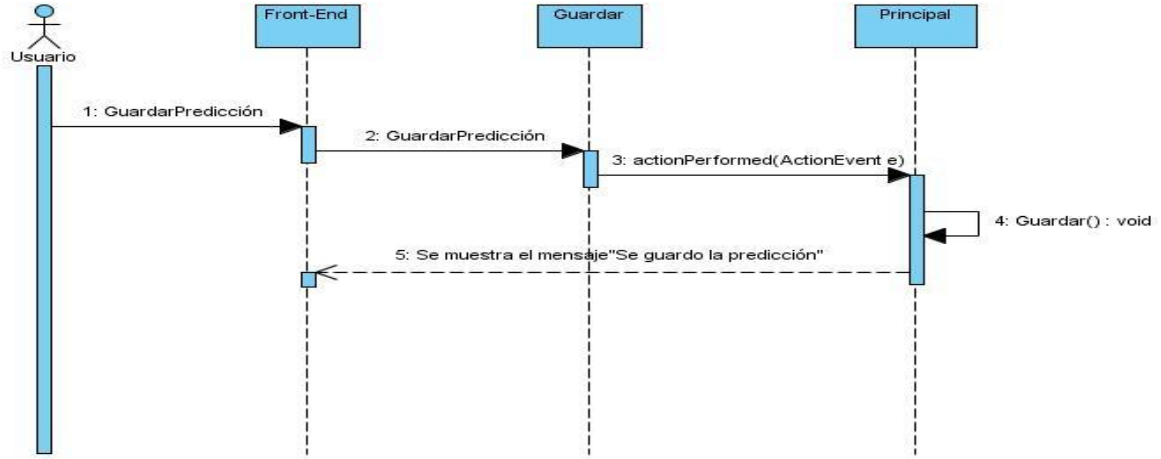


Figura 8.Diagrama de secuencia para el escenario Guardar del caso de uso Predecir Actividad Biológica.

El diagrama de secuencia para el caso de uso Generar Modelo del módulo de Árboles de Regresión se puede encontrar en el Anexo1.

Diagramas de secuencias del diseño para el plug-in de Programación Genética.

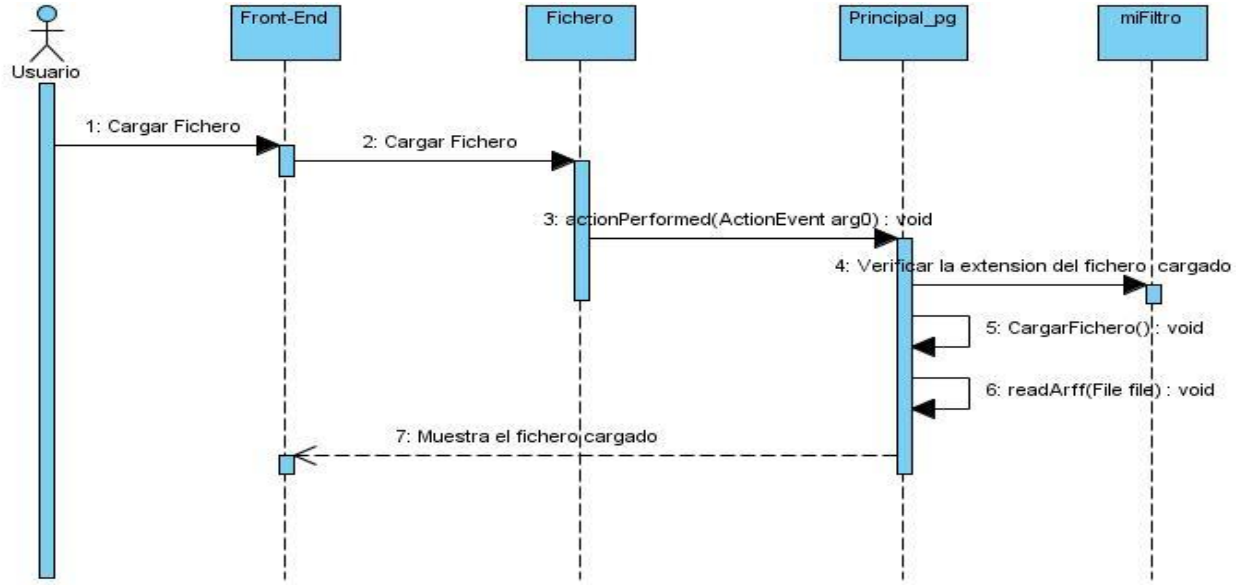


Figura 9.Diagrama de secuencia para el escenario Cargar Fichero del caso de uso Generar Modelo.

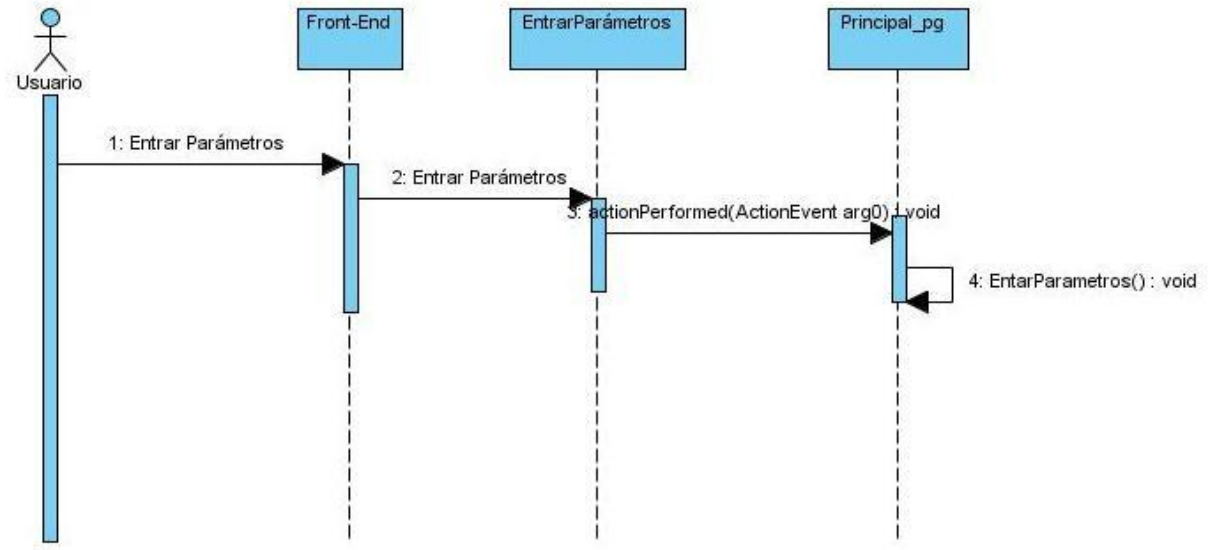


Figura 10.Diagrama de secuencia para el escenario Entrar Parámetros del caso de uso Generar Modelo.

El diagrama de secuencia para el caso de uso Generar Modelo del módulo de Programación Genética se puede encontrar en el Anexo 2.

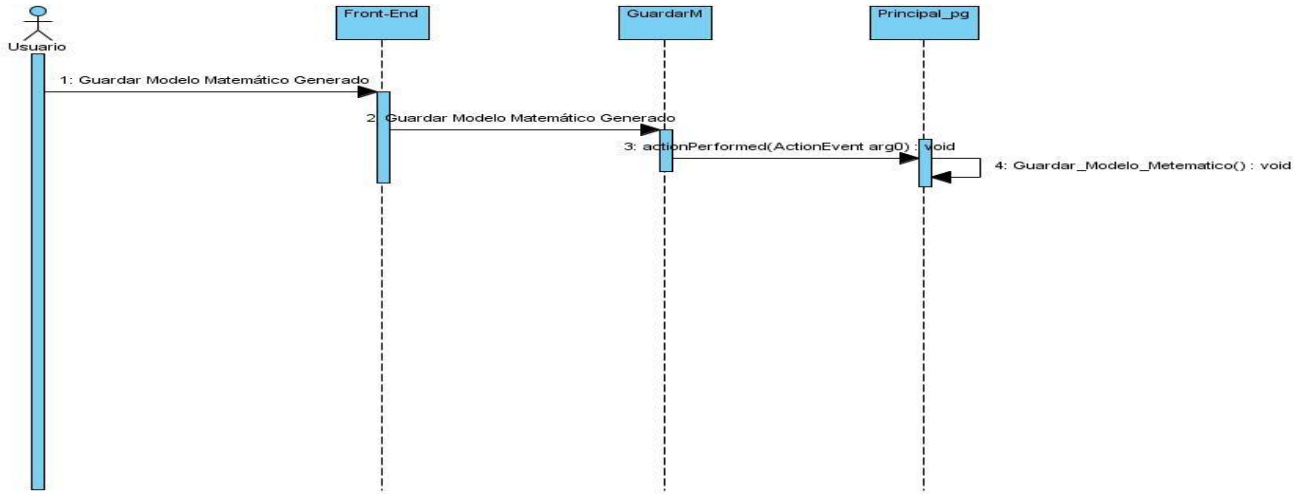


Figura 11.Diagrama de secuencia para el escenario Guardar Modelo del caso de uso Generar Modelo.

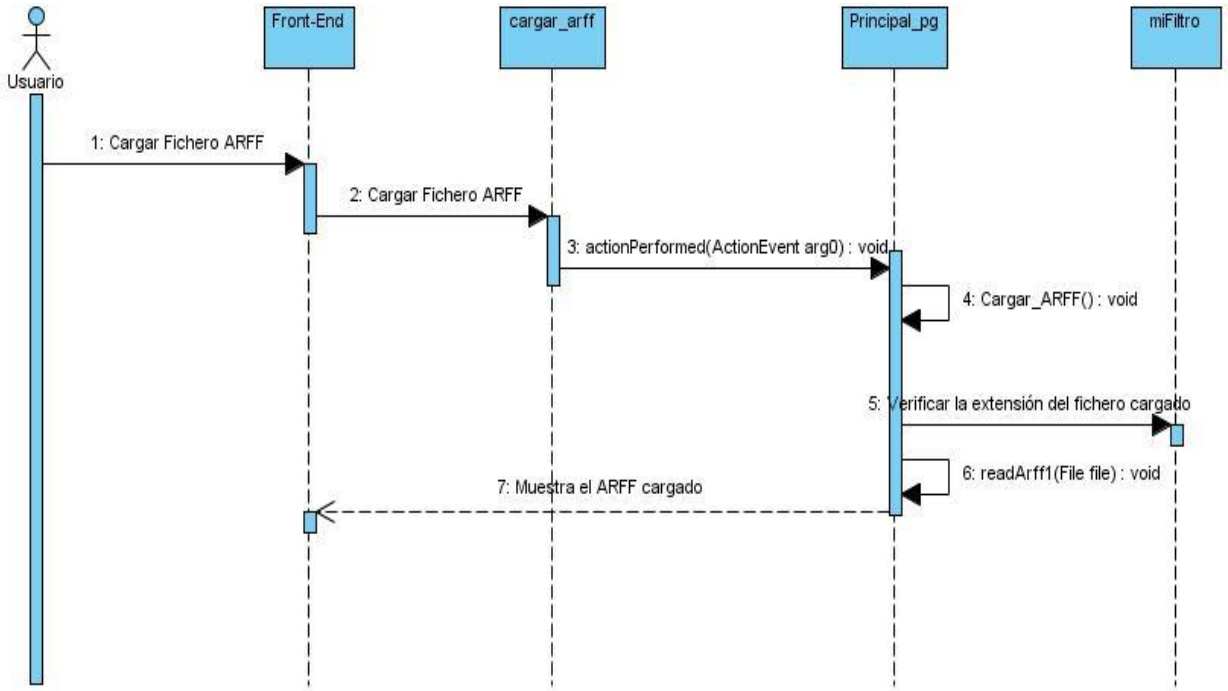


Figura 12.Diagrama de secuencia para el escenario Cargar Fichero del caso de uso Predecir Actividad Biológica.

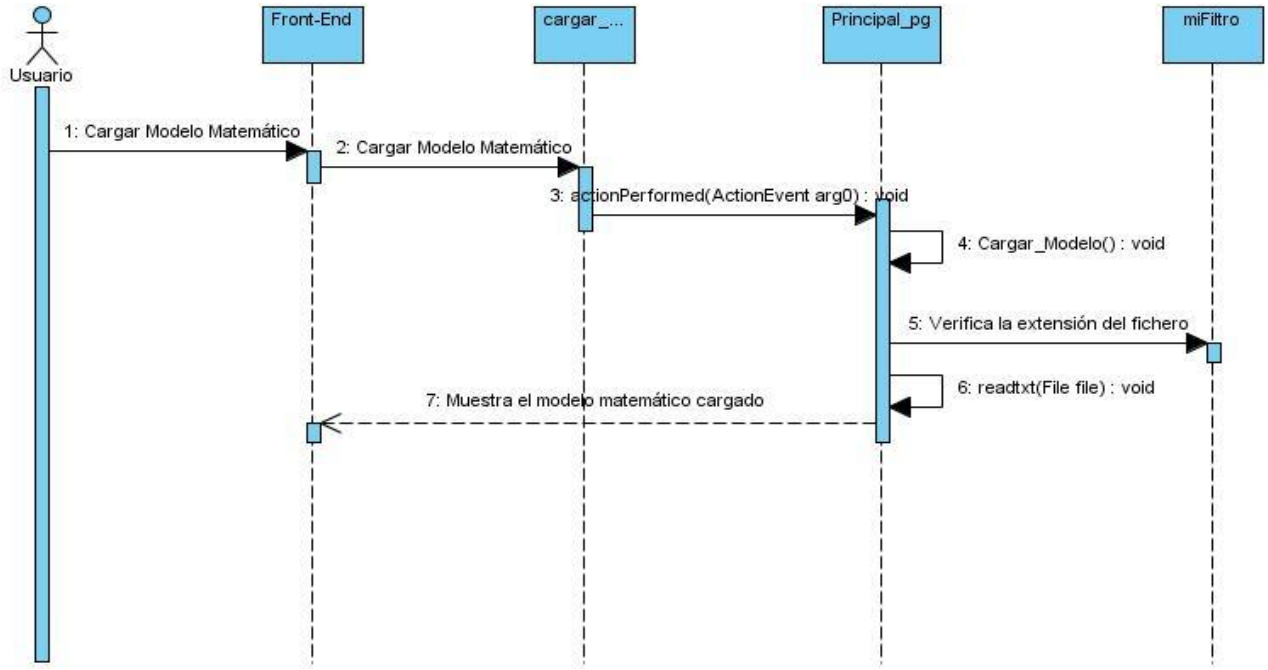


Figura 13.Diagrama de secuencia para el escenario Cargar Modelo del caso de uso Predecir Actividad Biológica.

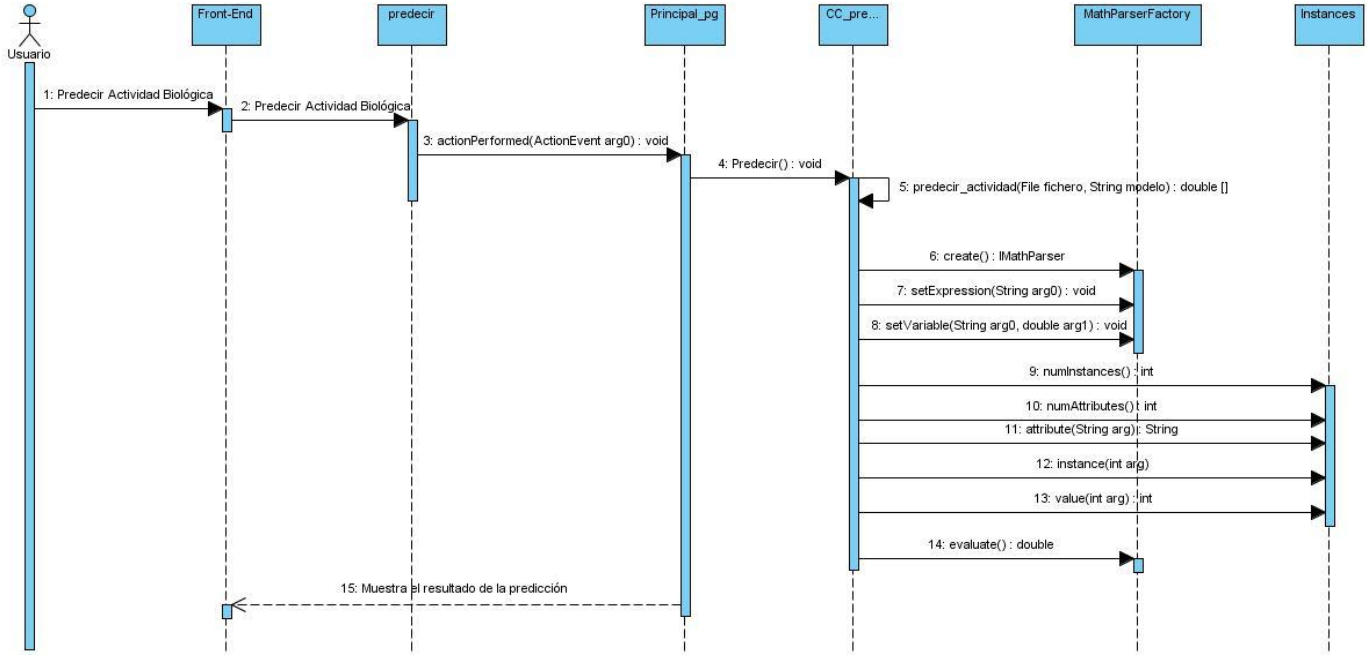


Figura 14.Diagrama de secuencia para el escenario predecir del caso de uso Predecir Actividad Biológica.

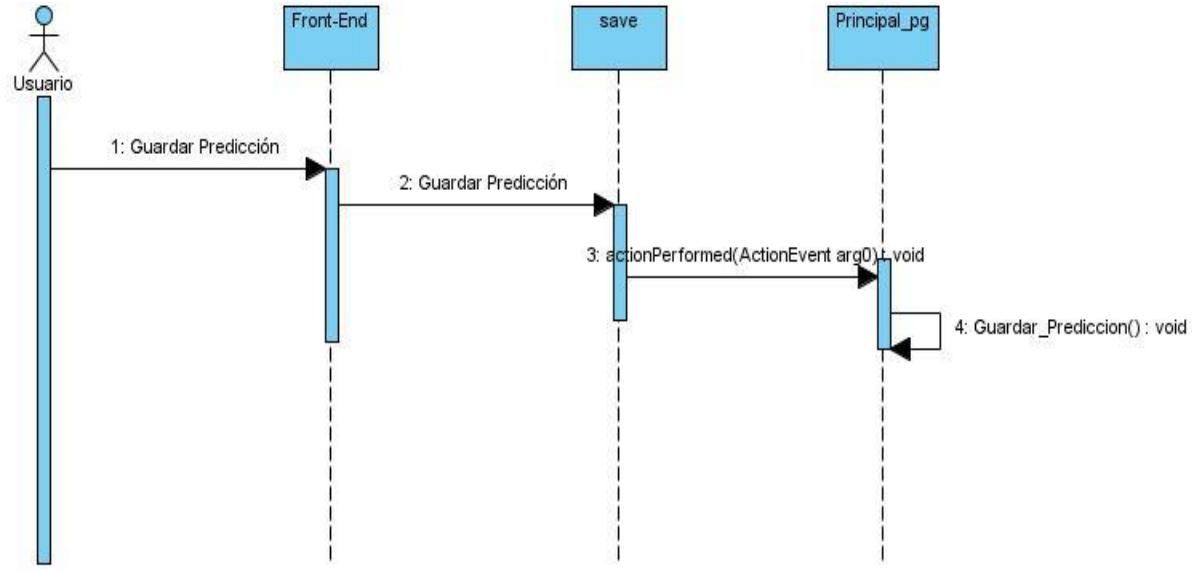


Figura 15.Diagrama de secuencia para el escenario Guardar Predicción del caso de uso Predecir Actividad Biológica.

3.1.2 Diagramas de clases del diseño.

Un diagrama de clases es un tipo de diagrama estático que describe la estructura de un sistema; muestra las clases, atributos y las relaciones entre ellos. Los diagramas de clases son utilizados durante el proceso de análisis y diseño de los sistemas, cuando se crea el diseño conceptual de la información que se manejará en el sistema. En los epígrafes siguientes se muestran los diagramas de clases tanto del plug-in de Árboles de Regresión como del plug-in de Programación Genética.

Diagramas de clases del diseño para el plug-in de Árboles de Regresión.

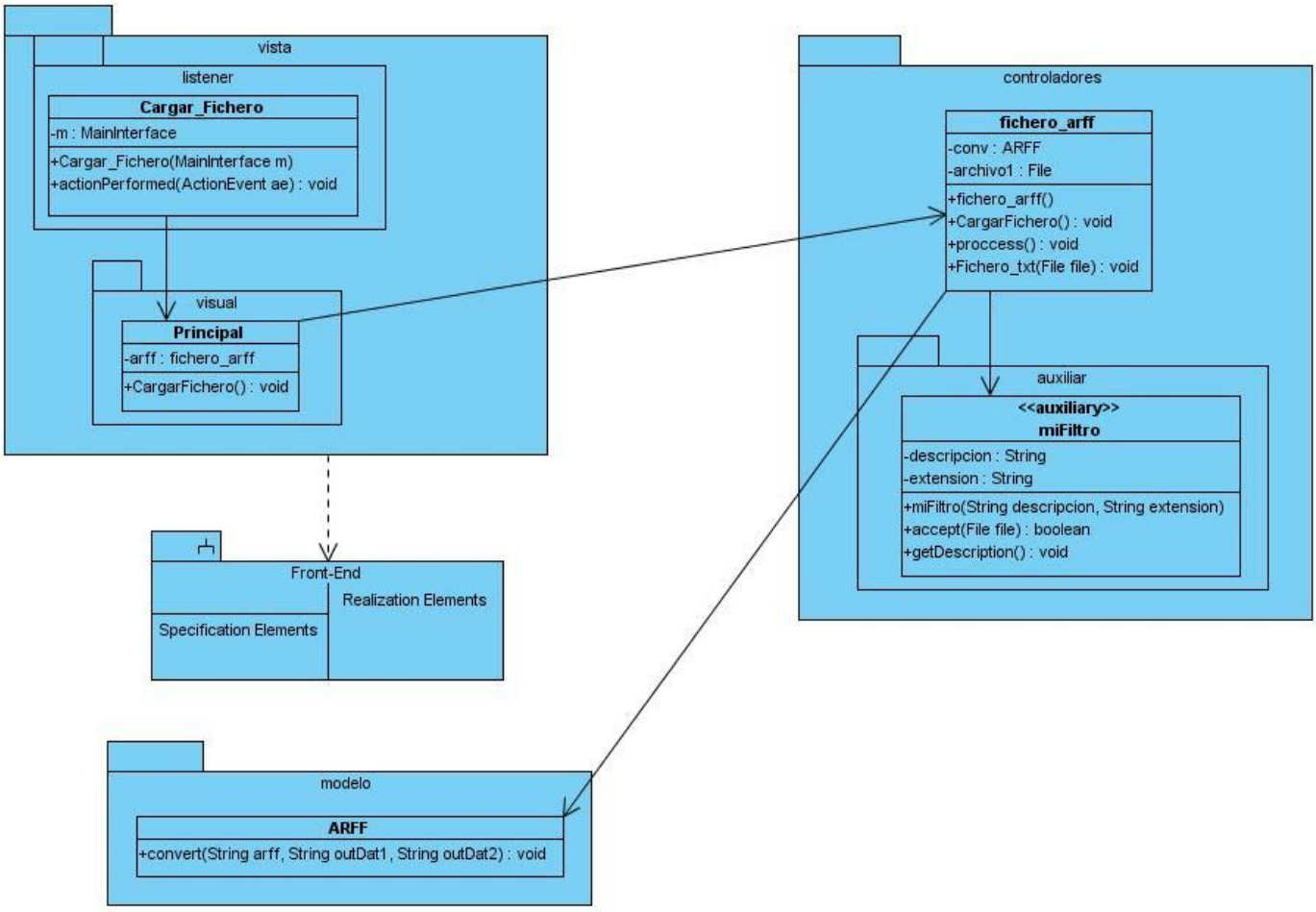


Figura 16.Diagrama de clases del diseño para el caso de uso Cargar ARFF.

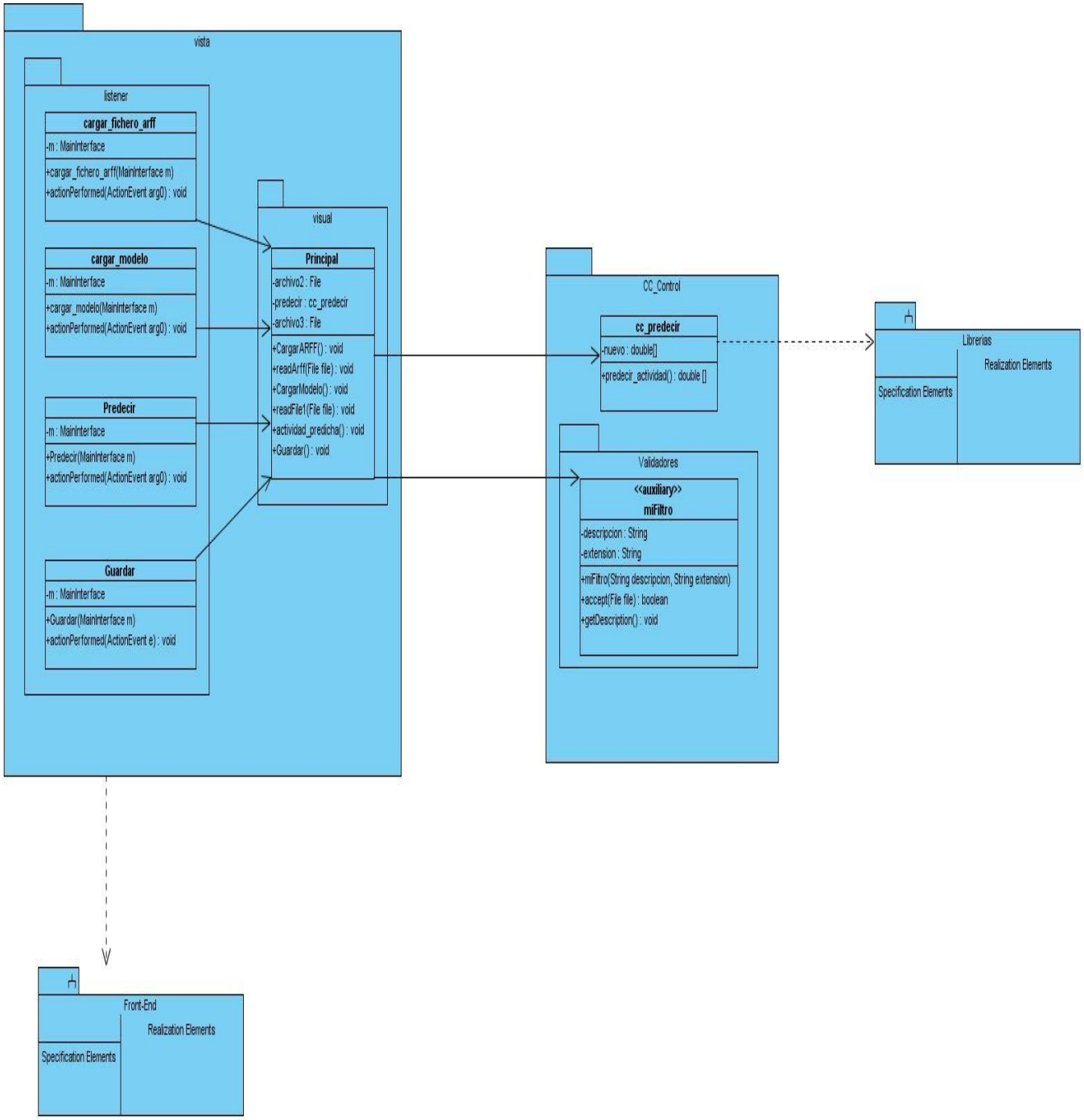


Figura 17.Diagrama de clases del diseño del caso de uso Predecir Actividad Biológica.

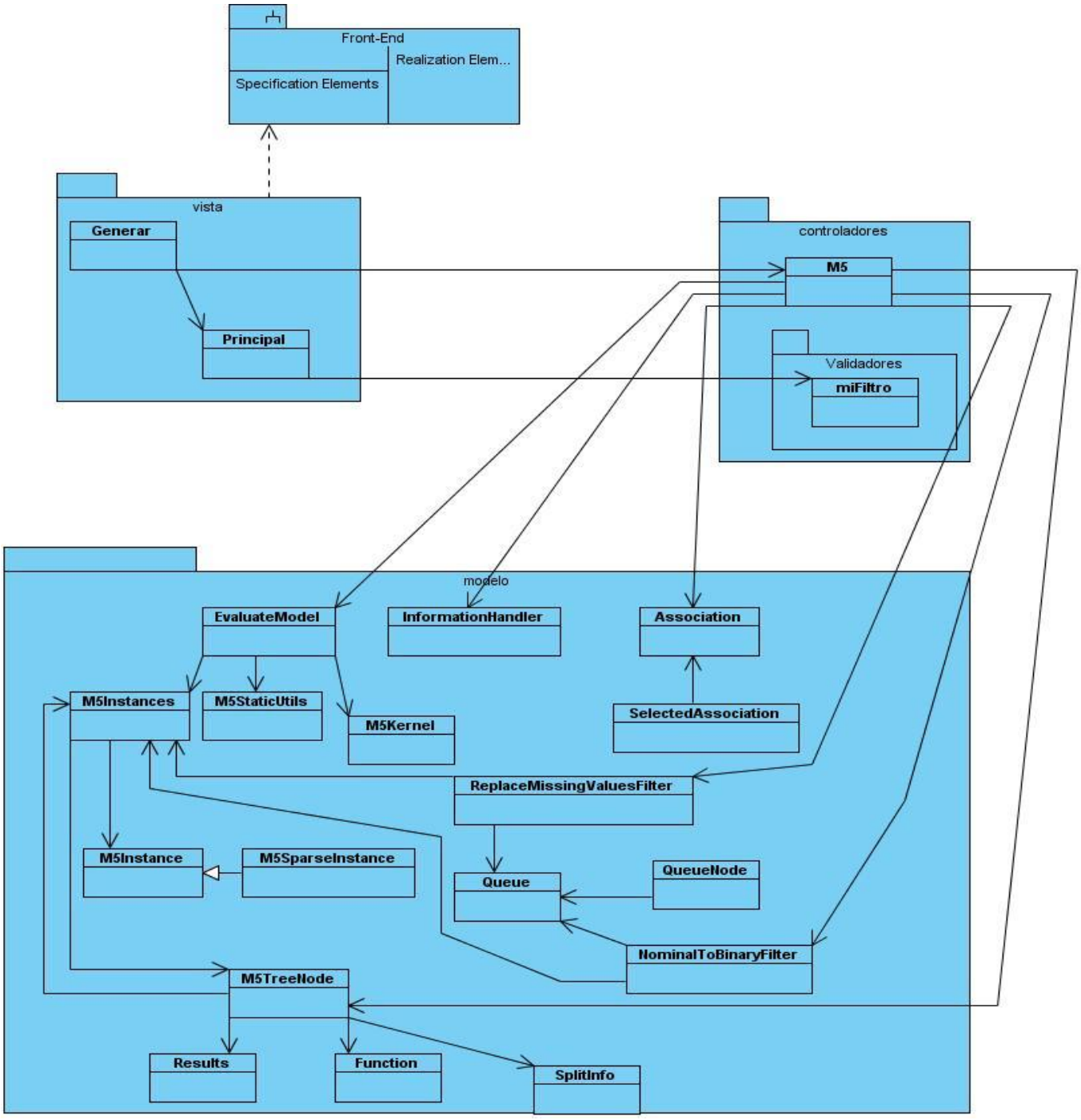


Figura 18.Diagrama de clases del diseño para el caso de uso Generar Modelo.

Diagramas de clases del diseño para el plug-in de Programación Genética.

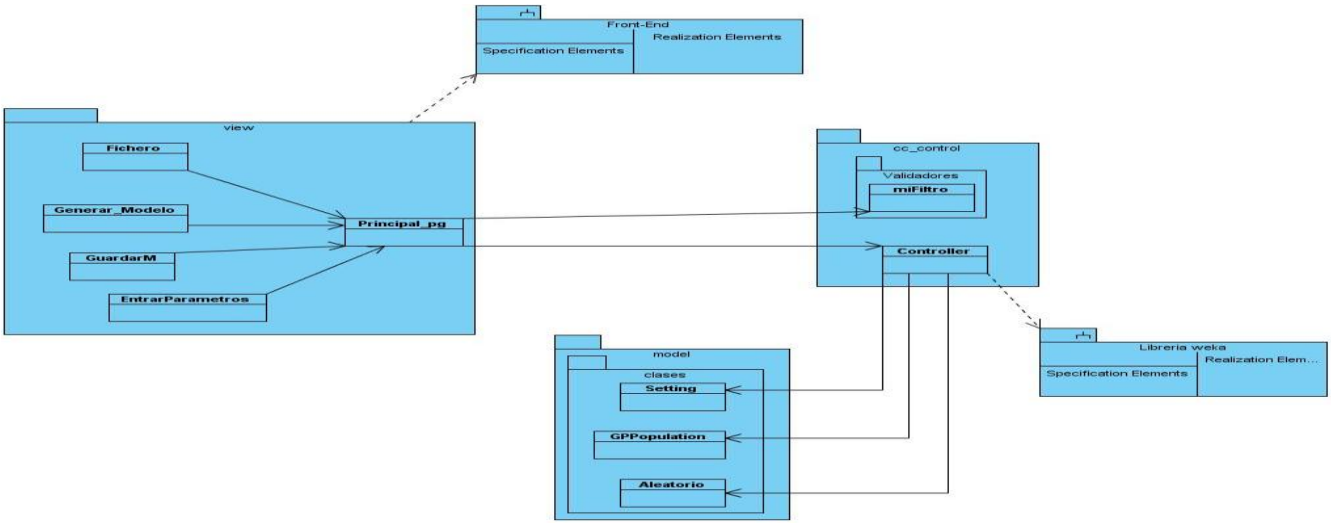


Figura 19.Diagrama de clases del diseño para el caso de uso Generar Modelo.

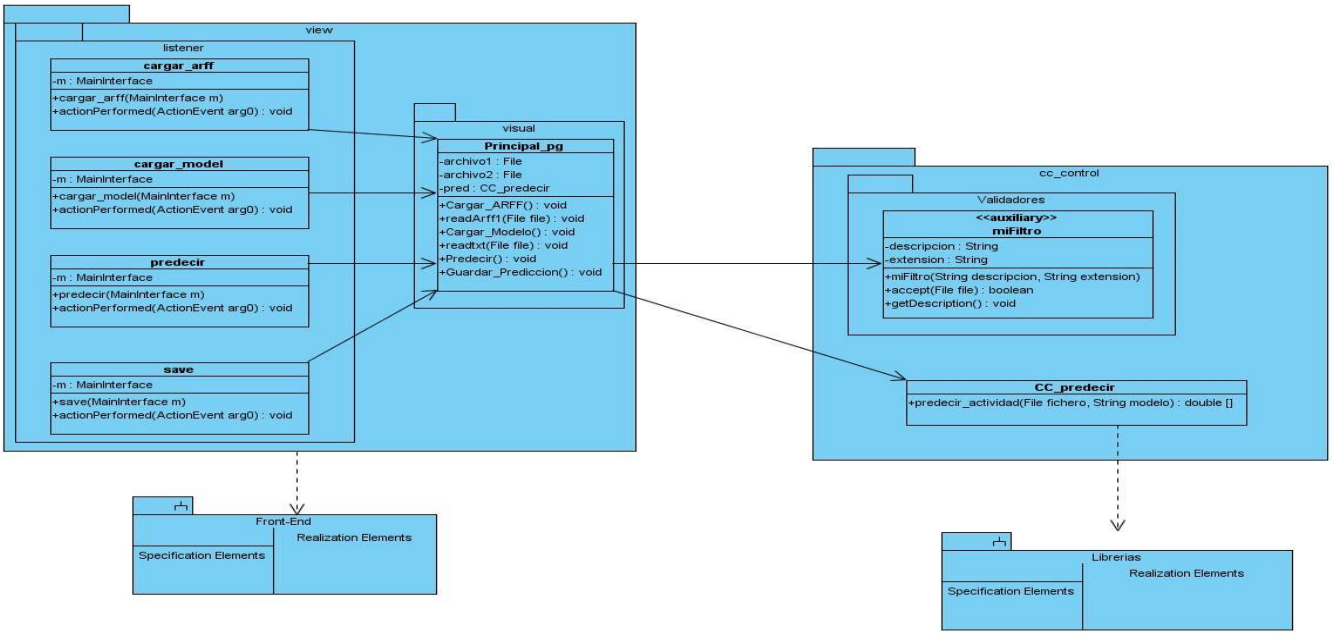


Figura 20.Diagrama de clases del diseño para el caso de uso Predecir Actividad Biológica.

3.1.3 Descripción de las clases del diseño para el plug-in de Árboles de Regresión.

Nombre: fichero_arff	
Tipo de clase: Controladora	
Atributos:	**Tipo:**
conv	ARFF
archivo1	File
Principales Responsabilidades:	
Nombre:	CargarFichero()
Descripción:	Se carga el fichero ARRF y dicho fichero es convertido en dos .dat uno con un 90% de la muestra y el otro con el 10% restante los cuales son guardados en la carpeta datasets.
Nombre:	proccess()
Descripción:	En este método se crean las carpetas results, scripts y datasets necesarias para la ejecución del algoritmo.

Las descripciones de las otras principales clases del diseño del plug-in de Árboles de Regresión se encuentran en el documento complementario: Descripciones de las clases del diseño.

3.1.4 Descripción de las clases del diseño para el plug-in de Programación Genética.

Nombre: Controller	
Tipo de clase: Controladora	
Atributos:	**Tipo:**
instClasificadas	Instances

population	GPPopulation
newPopulation	GPPopulation
setting	Setting
random	Aleatorio
Principales Responsabilidades:	
Nombre:	Controller(Instances instClasificadas, Setting setting)
Descripción:	Es el método principal para generar el modelo donde se le pasa el ARFF y los parámetros necesarios.
Nombre:	runAlgoritm()
Descripción:	Se crean las generaciones.
Nombre:	Imp()
Descripción:	En este método se van imprimiendo dentro de cada generación todos los modelos matemáticos.

Las descripciones del resto de las principales clases del diseño del plug-in de Programación Genética se encuentran en el documento complementario: Descripciones de las clases del diseño.

3.2 Patrones de diseño.

En la realización del diseño para la aplicación informática a implementar se utilizaron diferentes patrones con el objetivo de asegurar una solución mucho más confiable. A continuación se mencionan estos patrones.

Creador: el patrón creador nos ayuda a identificar quién debe ser el responsable de la creación 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, o almacena o maneja varias instancias de la clase. Este patrón brinda soporte de bajo acoplamiento, lo cual supone facilidad de mantenimiento y reutilización [21]

Alta cohesión: este patrón propone asignar la responsabilidad de manera que la complejidad se mantenga dentro de los límites manejables asumiendo solamente las responsabilidades que deben manejar, evadiendo un trabajo excesivo. Su utilización mejora la claridad y facilidad con que se entiende el diseño, simplifica el mantenimiento y las mejoras de funcionalidad, generan un bajo acoplamiento y soporta mayor capacidad de reutilización. [21]

Bajo acoplamiento: es la idea de tener las clases lo menos ligadas entre sí que se pueda. De tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de las clases, potenciando la reutilización, y disminuyendo la dependencia entre estas. [21]

Controlador: el patrón controlador sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que, la clase controladora recibe los datos del usuario y los envía a las distintas clases según el método llamado. Este patrón sugiere que la lógica de negocios debe estar separada de la capa de presentación, lo que permite aumentar la reutilización de código y a la vez tener un mayor control. [21]

3.3 Patrón Arquitectónico Empleado.

Los patrones arquitectónicos especifican un conjunto predefinido de subsistemas con sus responsabilidades y una serie de recomendaciones, para organizar los distintos componentes.

Figura 21: Representación de la arquitectura en tres capas

Para la organización del modelo de diseño se utilizó la arquitectura en capas donde cada capa está constituida por una o varias clases que colaboran en una tarea o responsabilidad específica y una capa solo puede utilizar la capa inferior a ella. La importancia de este patrón radica en simplificar la comprensión y la organización del desarrollo de sistemas complejos, reduciendo las dependencias de forma que las capas más bajas no son conscientes de ningún detalle o interfaz de las superiores. [21]En el diseño de los plug-ins se generalizó la división de las aplicaciones en 3 capas principales como se muestra en la figura 21 donde cada capa tiene las siguientes responsabilidades:

- Capa de presentación: Esta capa resuelve la presentación de datos al usuario y es la encargada de "dibujar" las pantallas de la aplicación al usuario, y tomar los eventos que el cliente genere.
- Capa de lógica de negocio: Esta capa resuelve la lógica de la aplicación. Contiene los algoritmos, validaciones y coordinación necesaria para resolver la problemática.
- Capa de acceso a datos: Esta capa resuelve el acceso a datos, abstrayendo a su capa superior de la complejidad del acceso e interacción con los diferentes orígenes de datos.

3.4 Diagrama de despliegue.

El Diagrama de Despliegue muestra la disposición física de los distintos nodos que componen un sistema y cómo se distribuyen los componentes sobre dichos nodos.

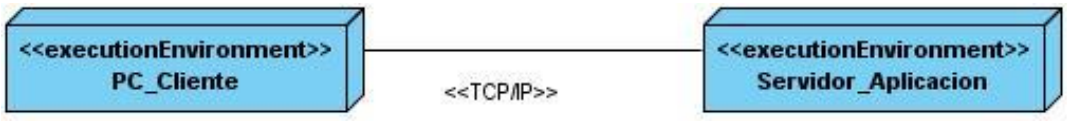


Figura 22. Diagrama de despliegue

3.5 Conclusiones Parciales.

En este capítulo se muestran los diagramas de clases y secuencia del diseño por cada caso de uso. Se describen las principales clases del diseño para ambos plug-ins desarrollados y se justifica el estilo arquitectónico usado y los patrones de diseño empleados. Finalmente, se muestra el diagrama de despliegue con la disposición física de los distintos nodos que componen el sistema.

CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBA

Como resultado del trabajo realizado en el capítulo anterior se profundizó en los casos de usos detallándolos de manera que permita reflejar una vista interna del sistema, descrita con el lenguaje de los desarrolladores lo que facilita el paso al flujo de trabajo de Implementación. En este capítulo se mostrará el diagrama de componentes por paquetes, además, se realizarán pruebas de caja negra para analizar el funcionamiento de la aplicación desarrollada.

4.1 Diagrama de componentes.

Un diagrama de componentes muestra las organizaciones y dependencias lógicas entre componentes software. Los componentes representan todos los tipos de elementos software que entran en la fabricación de aplicaciones informáticas, pueden ser de código fuente, binarios o ejecutables.
Por la complejidad del sistema, se consideró que no era factible una representación del diagrama de componentes de forma explícita de los dos plug-ins desarrollados Por lo que a continuación se muestra el diagrama de componentes con los plug-ins en forma de paquetes.

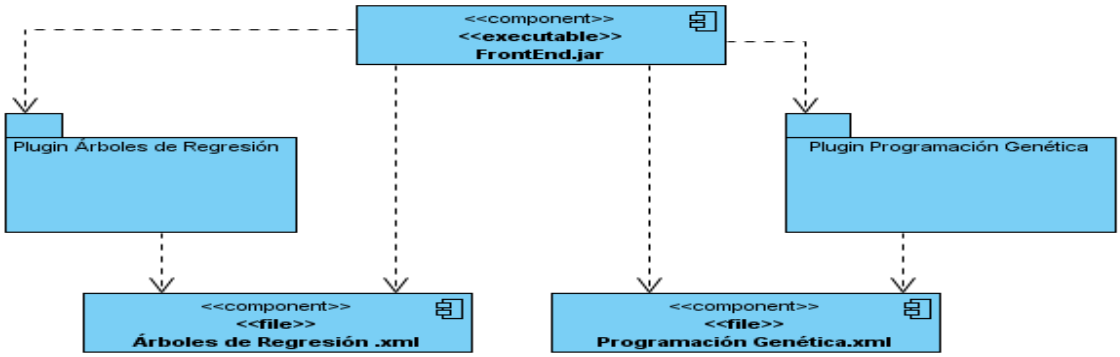


Figura 23. Diagrama de Componentes.

En los Anexos 3 y 4 se muestran los diagramas de componentes de forma explícita para los plug-ins de Árboles de Regresión y Programación Genética respectivamente.

4.2 Pruebas del Sistema

Las pruebas de software, son los procesos que permiten verificar la calidad de un producto de software. Son utilizadas para identificar posibles fallos de implementación, calidad, o usabilidad de un programa. Básicamente es una fase en el desarrollo de software que consiste en probar las aplicaciones construidas. [22]

4.2.1 Diseño de las Pruebas de Unidad (Caja Negra)

La prueba de caja negra se refiere a las pruebas que se llevan a cabo sobre la interfaz del software. O sea, los casos de prueba pretenden demostrar que las funcionalidades del software son operativas, que la entrada se acepta de forma adecuada y que se produce un resultado correcto, así como que la integridad de la información externa se mantiene. [22]

Diseño de casos de prueba para el plug-in de Árboles de Regresión.

Las celdas de las tablas que contienen los valores “V” e “I” donde “V” indica válido e “I” indica inválido.

Caso de uso: “Cargar Fichero”

Sección: “Cargar Fichero”

Variable (Var.)

1. fichero ARFF

Escenario	**Var 1**	**Respuesta del Sistema**	**Resultado de la Prueba**	**Flujo Central**
Cargar Fichero correctamente	V	Muestra el mensaje “El ARFF fue convertido en .dat y ya están creadas las carpetas	Muestra el mensaje “El ARFF fue convertido en .dat y ya están creadas las	1. Pulsar el botón Cargar Fichero. 2. Se carga el ARFF y se presiona el

		necesarias para poder generar el modelo".	carpetas necesarias para poder generar el modelo".	botón Abrir.
Cargar Fichero incorrectamente	I	Muestra el mensaje: "No se cargó el ARFF".	Muestra el mensaje: "No se cargó el ARFF".	1. Pulsar el botón Cargar Fichero. 2. Presiona el botón Cancelar.

Caso de uso: "Generar Modelo"

Sección: "Generar Modelo"

Variable (Var.)

1. fichero config.txt.

Escenario	**Var 1**	**Respuesta del Sistema**	**Resultado de la Prueba**	**Flujo Central**
Generar Modelo correctamente.	V	Se muestra en pantalla el modelo matemático que ha sido generado.	Se muestra en pantalla el modelo matemático que ha sido generado.	1. Presiona el botón Generar Modelo. 2. Se carga el fichero de texto guardado en la carpeta scripts

				y se presiona el botón Abrir.
Generar modelo incorrectamente.	I	No se generó el modelo matemático.	No se generó el modelo matemático .	1. Presiona el botón Generar Modelo. 2. Presiona el botón Cancelar.

Caso de uso: **"**Predecir Actividad"

Sección: "Cargar ARFF"

Variable (Var.)

1. fichero ARFF

Escenario	**Var 1**	**Respuesta del Sistema**	**Resultado de la Prueba**	**Flujo Central**
Cargar ARFF correctamente	V	Se visualiza en pantalla el ARFF cargado.	Se visualiza en pantalla el ARFF cargado.	1. Presiona el botón Cargar ARFF. 2. Se carga el ARFF y se presiona el botón Abrir.
Cargar ARFF incorrectamente	I	No se carga el ARFF.	No se carga el ARFF.	1. Se presiona el botón Cargar ARFF. 2. Presiona el botón Cancelar.

Sección: "Cargar Modelo"

Variable (Var.)

1. modelo matemático

Escenario	Var 1	Respuesta del Sistema	Resultado de la Prueba	Flujo Central
Cargar Modelo correctamente	V	Se visualiza en pantalla el modelo cargado.	Se visualiza en pantalla el modelo cargado.	1. Presiona el botón Cargar Modelo. 2. Se carga el modelo matemático guardado en la carpeta results y presiona el botón Abrir.
Cargar Modelo incorrectamente	I	No se carga el modelo matemático.	No se carga el modelo matemático.	1. Se presiona el botón Cargar Modelo. 2. Se presiona el botón Cancelar.

Sección: “Predecir Actividad”

Variable (Var.)

1. fichero ARFFF

2. modelo matemático

Escenario	Var 1	Var 2	Respuesta del Sistema	Resultado de la Prueba	Flujo Central
Predecir Actividad correctamente	V	V	Se muestra en pantalla el resultado de la actividad predicha para cada muestra.	Se muestra en pantalla el resultado de la actividad predicha para cada muestra.	1. Presiona el botón Predecir Actividad.

Predecir Actividad incorrectamente	I	I	Muestra el mensaje “No se realizó la predicción”.	Muestra el mensaje “No se realizó la predicción”.	1. Presiona el botón Predecir Actividad.
Predecir Actividad incorrectamente	V	I	Muestra el mensaje “No se realizó la predicción”.	Muestra el mensaje “No se realizó la predicción”	1. Presiona el botón Predecir Actividad.
Predecir Actividad incorrectamente	I	V	Muestra el mensaje “No se realizó la predicción”.	Muestra el mensaje “No se realizó la predicción”.	1. Presiona el botón Predecir Actividad.

Sección: “Guardar”

Escenario	**Respuesta del Sistema**	**Resultado de la Prueba**	**Flujo Central**
Guardar correctamente	Se guarda el resultado de la predicción realizada.	Se guarda el resultado de la predicción realizada.	1. Presiona el botón Guardar. 2. Se específica el lugar donde se desea guardar y se presiona el botón Guardar.
Guardar incorrectamente	No se guarda el resultado de la predicción realizada.	No se guarda el resultado de la predicción realizada.	1. Presiona el botón Guardar. 2. Presiona el botón Cancelar.

Diseño de casos de pruebas para el plug-in de Programación Genética.

Las celdas de las tablas que contienen los valores “V” e “I” donde “V” indica válido e “I” indica inválido.

Caso de uso: “Generar Modelo”

Sección: “Cargar ARFF”

Variable (Var.)

1. fichero ARFFF

Escenario	**Var 1**	**Respuesta del Sistema**	**Resultado de la Prueba**	**Flujo Central**
Cargar ARFF correctamente	V	Se carga el ARFF y se visualiza en pantalla.	Se carga el ARFF y se visualiza en pantalla.	1. Presiona el botón Cargar Fichero. 2. Se carga el ARFF y se presiona el botón Abrir.
Cargar ARFF incorrectamente	I	No se carga el ARFF.	No se carga el ARFF.	1. Presiona el botón Cargar Fichero. 2. Presiona el botón Cancelar.

Sección: “Entrar Parámetros”

Escenario	Respuesta del Sistema	Resultado de la Prueba	Flujo Central
Entrar Parámetros	El sistema muestra un panel con los seis parámetros que el usuario debe llenar.	El sistema muestra un panel con los seis parámetros que el usuario debe llenar.	1.Se presiona el botón Entrar Parámetros

Sección: “Crear Modelo”

Variable (Var.)

1. fichero ARFF

2. parámetros

Escenario	Var 1	Var 2	Respuesta del Sistema	Resultado de la Prueba	Flujo Central
Crear Modelo correctamente	V	V	Se muestra en pantalla el modelo matemático generado.	Se muestra en pantalla el modelo matemático generado.	1. Se presiona el botón Crear Modelo.
Crear Modelo incorrectamente	I	I	No se genera el modelo matemático.	No se genera el modelo matemático.	1. Se presiona el botón Crear

					Modelo.
Crear Modelo incorrectamente	V	I	No se genera el modelo matemático.	No se genera el modelo matemático.	1. Se presiona el botón Crear Modelo.
Crear Modelo incorrectamente	I	V	No se genera el modelo matemático.	No se genera el modelo matemático.	1. Se presiona el botón Crear Modelo

Sección: “Guardar Modelo”

Escenario	**Respuesta del Sistema**	**Resultado de la Prueba**	**Flujo Central**
Guardar correctamente	Se guarda el modelo matemático en la dirección especificada por el usuario.	Se guarda el modelo matemático en la dirección especificada por el usuario.	1. Se presiona el botón Guardar Modelo. 2. Se específica el lugar donde se desea guardar y se presiona el botón Guardar.
Guardar incorrectamente	No se guardó el modelo matemático.	No se guardó el modelo matemático.	1. Se presiona el botón Guardar Modelo. 2. Se presiona el botón Cancelar.

Caso de uso: "Predecir Actividad**"**

Sección "Cargar ARFF"

Variable (Var.)

1. fichero ARFF

Escenario	**Var 1**	**Respuesta del Sistema**	**Resultado de la Prueba**	**Flujo Central**
Cargar ARFF correctamente	V	Se muestra en pantalla el ARFF cargado.	Se muestra en pantalla el ARFF cargado.	1. Se presiona el botón Cargar ARFF. 2. Se carga el ARFF y presiona el botón Abrir.
Cargar ARFF incorrectamente	I	No se carga el ARFF.	No se carga el ARFF.	1. Se presiona el botón Cargar ARFF. 2. Se presiona el botón Cancelar.

Sección: "Cargar Modelo"

Variable (Var.)

1. modelo matemático

Escenario	**Var 1**	**Respuesta del Sistema**	**Resultado de la Prueba**	**Flujo Central**
Cargar Modelo correctamente	V	Se muestra en pantalla el modelo matemático cargado.	Se muestra en pantalla el modelo matemático	1. Se presiona el botón Cargar Modelo. 2. Se carga el modelo

			cargado.	matemático y presiona el botón Abrir.
Cargar Modelo incorrectamente	I	No se carga el modelo matemático.	No se carga el modelo matemático.	1. Se presiona el botón Cargar Modelo. 2. Se presiona el botón Cancelar.

Sección: “Predecir”

Variable (Var.)

1. fichero ARFF

2. modelo matemático

Escenario	**Var 1**	**Var 2**	**Respuesta del Sistema**	**Resultado de la Prueba**	**Flujo Central**
Predecir Actividad correctamente	V	V	Muestra en pantalla el resultado de la actividad predicha para cada muestra.	Muestra en pantalla el resultado de la actividad predicha para cada muestra.	1. Se presiona el botón Predecir.
Predecir Actividad incorrectamente	I	I	Muestra el mensaje de error “No predijo ”	Muestra el mensaje de error “No predijo ”	1. Se presiona el botón Predecir.
Predecir Actividad incorrectamente	I	V	Muestra el mensaje de error “No predijo ”	Muestra el mensaje de error “No predijo ”	1. Se presiona el botón Predecir.
Predecir Actividad incorrectamente	V	I	Muestra el mensaje de error “No predijo ”	Muestra el mensaje de error “No predijo ”	1. Se presiona el botón Predecir.

Sección: “Guardar Predicción”

Escenario	Respuesta del Sistema	Resultado de la Prueba	Flujo Central
Guardar correctamente	Se guarda el resultado de la predicción realizada.	Se guarda el resultado de la predicción realizada.	1. Se presiona el botón Guardar Predicción. 2. Se específica el lugar donde se desea guardar y se presiona el botón Guardar.
Guardar incorrectamente	No se guarda el resultado de la predicción realizada.	No se guarda el resultado de la predicción realizada.	1. Se presiona el botón Guardar Predicción. 2. Se presiona el botón Cancelar.

4.3 Validación de los modelos generados.

Para validar los modelos generados utilizando el algoritmo de Árboles de Regresión se utilizó una muestra de cefalosporina de 99 instancias de las que se conoce su actividad biológica y 81 valores de descriptores diferentes que describen cada una de las moléculas. Cada vez que se carga el fichero en la aplicación se genera una muestra aleatoria con el 90% de entrenamiento y el 10% de prueba garantizando que cada muestra cargada sea diferente y por tanto el modelo de estructura-actividad también lo sea. Para medir la calidad de los modelos se utilizó el coeficiente de regresión lineal entre el valor de actividad real y la actividad predicha por el modelo que se generó. Esta medida estadística es muy utilizada por los investigadores para evaluar la calidad en este tipo de estudio.

En el caso de la Programación Genética se fijaron los parámetros de la siguiente forma:

Tamaño de la población: 300

Máxima profundidad del árbol: 15

Probabilidad de reproducción: 0.2

Probabilidad de cruce: 0.7

Probabilidad de mutación: 0.1

Número de generaciones: 500

En el caso de los Árboles de Regresión se utilizaron los parámetros por defecto definidos en la tesis de maestría sobre modelos QSAR utilizando esta técnica, presentada dentro de la Maestría de Bioinformática. [2]

Seguidamente se muestra en la tabla 2 los valores de coeficiente de regresión que se obtienen en cada una de las 10 muestras cargadas en la aplicación.

Algoritmo	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Árboles de Regresión	0.94	0.92	0.93	0.91	0.95	0.92	0.93	0.92	0.92	0.91
Programación Genética	0.87	0.87	0.89	0.87	0.84	0.82	0.81	0.86	0.85	0.87

Tabla 2. Valores de los Coeficientes de Regresión para los plug-ins realizados.

En este tipo de estudio un coeficiente de correlación entre 0,5-0,7 se considera un resultado medio, poco significativo, sin embargo un coeficiente de correlación entre 0,8-1 es un modelo de calidad aceptable. En la tabla se puede observar que en ambos algoritmos el comportamiento es entre 0,81-0,95 por lo que la calidad de los modelos QSAR se encuentra dentro del rango correcto.

Luego se utilizó cada uno de los 10 modelos generados y se predijo en su respectiva muestra de prueba, obteniéndose un coeficiente de regresión promedio para la predicción de 0,87 para Árboles de Regresión y de 0,79 para Programación Genética. La disminución de los coeficientes de regresión de cada técnica en la predicción es completamente lógica debido a que se utiliza el modelo para predecir sobre una

muestra que no fue utilizada para su generación, estos coeficientes, 0,79 y 0,87 indican que los modelos logran predecir correctamente en nuevas muestras en estudio, siempre que las moléculas pertenezcan a la misma familia de compuestos que las moléculas con las que se generó el modelo.

Es válido mencionar además que a medida que aumenta el tamaño de la muestra y el número de generaciones a ejecutar en el algoritmo de Programación Genética el tiempo de ejecución aumenta considerablemente. También se pudo observar que para más de 500 generaciones los modelos que se generan no ganan mucho en calidad por lo que se propone al usuario de la aplicación trabajar con un número de generaciones menor de 500.

4.4 Conclusiones Parciales.

En este capítulo se mostró el diagrama de componentes del sistema por paquetes. Se le realizaron las pruebas de caja negra a la interfaz de la aplicación, además se validaron los modelos matemáticos generados realizándose 10 experimentos para ambos algoritmos. En cada uno de los modelos que se generó los coeficientes de correlación se mantuvieron sobre rangos aceptables para este tipo de estudio. Las predicciones realizadas con los modelos generados mostraron que estos son capaces de describir a las moléculas con las que fueron generados y por esta razón las predicciones fueron buenas en la mayoría de los casos.

CONCLUSIONES

- Se implementó un plug-ins para predecir actividad biológica en compuestos orgánicos por Árboles de Regresión.
- Se implementó un plug-ins para predecir actividad biológica en compuestos orgánicos por Programación Genética
- Se validaron los modelos generados, encontrándose como inconveniente en el algoritmo de Programación Genética, que para tamaños de población muy grandes el tiempo de ejecución es prolongado.

RECOMENDACIONES

- En el plug-in de Programación Genética se recomienda implementar una versión paralela del algoritmo, que permita disminuir el tiempo de ejecución.

REFERENCIAS BIBLIOGRÁFICAS

1. **MAROVAC, J.** *Investigación y desarrollo de nuevos medicamentos: de la molécula al fármaco.* Chile: Revista médica de Chile, 2001.

2. **MOLINA SOUTO, Yania.** *Desarrollo de modelos QSAR utilizando Programación Genética y Árboles de Regresión.* Ciudad de La Habana, Cuba: s.n., 2010.

3. **FATEMI, M.** *A novel quantitative structure-activity relationship model for prediction of biomagnification factor of some organochlorine pollutants.* Babolsar: MedLine, agosto de 2009.

4. **HAMMETT, Louis P.** *Physical organic chemistry; reaction rates, equilibria, and mechanisms.* New York : McGraw-Hill: International chemical series, 1940.

5. **BALABAN, A.** *Chemical Applications of Graph Theory.* s.l.: Academic Press Inc, 1976. 0120760503.

6. **MOLINA SOUTO, Yania y MEJIAS CESAR, Yuleidys.** *Módulo de predicción de actividad biológica anticancerígena de compuestos orgánicos, partiendo de fragmentos, utilizando Programación Genética.* Cuba : Bioinformatics, 2007.

7. **QUINLAN, J.** *Induction of Decision Trees. Proceedings of the 5th Australian Joint Conference on Artificial Intelligent.* Singapore: World Scientific, 1993. Vol. I.

8. **WANG, Y.** *Induction of model trees for predicting continuos classes. European Conference on Machine Learning.* Europa: s.n., 1997.

9. **BREIMAN, L.** *Classification and Regression Tree.* Belmont: s.n., 1984.

10. **GOLENDER, Valery y VESTERMAN, B.** Apex 3D San Diego. . *Apex 3D San Diego.* [En línea] [Citado el: 15 de 02 de 2010.] http://www.netsci.org/Science/Compchem/feature09.html.

11. **MAGAZINE, C. P.** Adapt or perish. *Adapt or perish.* [En línea] [Citado el: 18 de 02 de 2010.] http://www.chemicalprocessing.com/articles/2004/303.html.

12. **AGENCY, U. S. E. P.** Cancer Expert System, or OncoLogic, Fact Sheet. *Cancer Expert System, or OncoLogic, Fact Sheet.* [En línea] [Citado el: 17 de 03 de 2010.] http://www.epa.gov/oppt/cahp/pubs/can.htm.

13. **ESPAÑA, D. W.** Masadelante. *Masadelante.* [En línea] [Citado el: 22 de 02 de 2010.] http://www.masadelante.com/faqs/plug-in.

14. **LLC, Simple Machines.** ForosFilos. *ForosFilos.* [En línea] [Citado el: 18 de 02 de 2010.] http://www.forofilos.com/index.php/topic,2444.0.html.

15. **AFFILIATES, Oracle Corporation and/or its.** NetBeans. *NetBeans.* [En línea] [Citado el: 10 de 03 de 2010.] http://netbeans.org/.

16. **MARTÍNEZ VILLAVERDE, Julio.** *Un nuevo front-end para la plataforma alasgrato.* Cuba: s.n., 2009.

17. OPEN UP. *OPEN UP.* [En línea] [Citado el: 10 de 02 de 2010.] http://epf.eclipse.org/wikis/openup/index.htm.

18. VISUAL PARADIGM FOR UML 7.2 . *VISUAL PARADIGM FOR UML 7.2 .* [En línea] [Citado el: 19 de 04 de 2010.] http://www.visual-paradigm.com/product/vpuml/.

19. **FROUFE, Agustín.** Tutorial de Java. *Tutorial de Java.* [En línea] 1 de 01 de 1997. [Citado el: 20 de 03 de 2010.] http://www.ac.uma.es/localres/manuals/JavaTut-Froufe/index.html.

20. **GARCERANT, Iván.** Tecnologia y Synergix. *Tecnologia y Synergix.* [En línea] [Citado el: 15 de 03 de 2010.] http://synergix.wordpress.com/2008/07/10/modelo-de-dominio/.

21. **LARMAN, Craig.** *UML y Patrones.* s.l.: Prentice Hall, 1999.

22. *INGENIERÍA DE SOFTWARE.Un enfoque práctico.Capítulo 18 (Estrategias de Pruebas de Software) .* página 305-322.

BIBLIOGRAFÍA

1. **AFFILIATES, Oracle Corporation and/or its.** NetBeans. *NetBeans.* [En línea] [Citado el: 10 de 03 de 2010.] http://netbeans.org/.
2. **AGENCY, U. S. E. P.** Cancer Expert System, or OncoLogic, Fact Sheet. *Cancer Expert System, or OncoLogic, Fact Sheet.* [En línea] [Citado el: 17 de 03 de 2010.] http://www.epa.gov/oppt/cahp/pubs/can.htm
3. **BALABAN, A.** *Chemical Applications of Graph Theory.* s.l.: Academic Press Inc, 1976. 0120760503
4. **BREIMAN, L.** *Classification and Regression Tree.* Belmont : s.n., 1984.
5. **ESPAÑA, D. W.** Masadelante. *Masadelante.* [En línea] [Citado el: 22 de 02 de 2010.] http://www.masadelante.com/faqs/plug-in
6. **EXEC**. *Características del lenguaje Java.* [En línea] [Citado el: 8 de 04 de 2010]. Disponible en: http://www.mailxmail.com/curso/informatica/java/capitulo2.htm
7. **FATEMI, M.** *A novel quantitative structure-activity relationship model for prediction of biomagnification factor of some organochlorine pollutants.* Babolsar: MedLine, agosto de 2009.
8. **FROUFE, Agustín.** Tutorial de Java. *Tutorial de Java.* [En línea] 1 de 01 de 1997. [Citado el: 20 de 03 de 2010.] http://www.ac.uma.es/localres/manuals/JavaTut-Froufe/index.html
9. **HAMMETT, Louis P.** *Physical organic chemistry; reaction rates, equilibria, and mechanisms.* New York: McGraw-Hill: International chemical series, 1940.
10. **GARCERANT, Iván.** Tecnologia y Synergix. *Tecnologia y Synergix.* [En línea] [Citado el: 15 de 03 de 2010.] http://synergix.wordpress.com/2008/07/10/modelo-de-dominio/
11. **GOLENDER, Valery y VESTERMAN, B.** Apex 3D San Diego. *Apex 3D San Diego.* [En línea] [Citado el: 15 de 02 de 2010.] http://www.netsci.org/Science/Compchem/feature09.html.
12. **HERNÁNDEZ, S. Á**. *Metodología para el desarrollo de aplicaciones con tecnología Orientada a Objetos utilizando notación UML.* La Habana: 2000.
13. *INGENIERÍA DE SOFTWARE. Un enfoque práctico.Capítulo 18 (Estrategias de Pruebas de Software)* . página 305-322

14. **Ken, Sheriff, Paul y Forte, Stephen Spencer.** Model-View-Controller. *Model-View-Controller.* [En línea] [Citado el: 19 de 02 de 2010.] http://msdn.microsoft.com/en-us/library/ms978748.aspx.

15. **KOZA, John.** *Genetic Programming on the programming of computers by means of natural selection.* Massachusetts : The MIT Press, 1992.

16. **LARMAN, Craig.** *UML y Patrones.* s.l.: Prentice Hall , 1999.

17. **LLC, Simple Machines.** ForosFilos. *ForosFilos.* [En línea] [Citado el: 18 de 02 de 2010.] http://www.forofilos.com/index.php/topic,2444.0.html

18. **MAGAZINE, C. P.** Adapt or perish. *Adapt or perish.* [En línea] [Citado el: 18 de 02 de 2010.] http://www.chemicalprocessing.com/articles/2004/303.html

19. **MAROVAC, J.** *Investigación y desarrollo de nuevos medicamentos: de la molécula al fármaco.* Chile: Revista médica de Chile, 2001.

20. **MARTÍNEZ VILLAVERDE, Julio.** *Un nuevo front-end para la plataforma alasgrato.* Cuba : s.n., 2009. 1:4_21

21. **MOLINA SOUTO, Yania.** *Desarrollo de modelos QSAR utilizando Programación Genética y Árboles de Regresión .* Ciudad de La Habana, Cuba: s.n., 2010.

22. **MOLINA SOUTO, Yania y MEJIAS CESAR, Yuleidys.** *Módulo de predicción de actividad biológica anticancerígena de compuestos orgánicos, partiendo de fragmentos, utilizando Programación Genética.* Cuba: Bioinformática, 2007.

23. OPEN UP. *Open UP.* [En línea] [Citado el: 10 de 02 de 2010.] http://epf.eclipse.org/wikis/openup/index.htm

24. **QUINLAN, J.** *Induction of Decision Trees. Proceedings of the 5th Australian Joint Conference on Artificial Intelligent.* Singapore: World Scientific, 1993. Vol. I.

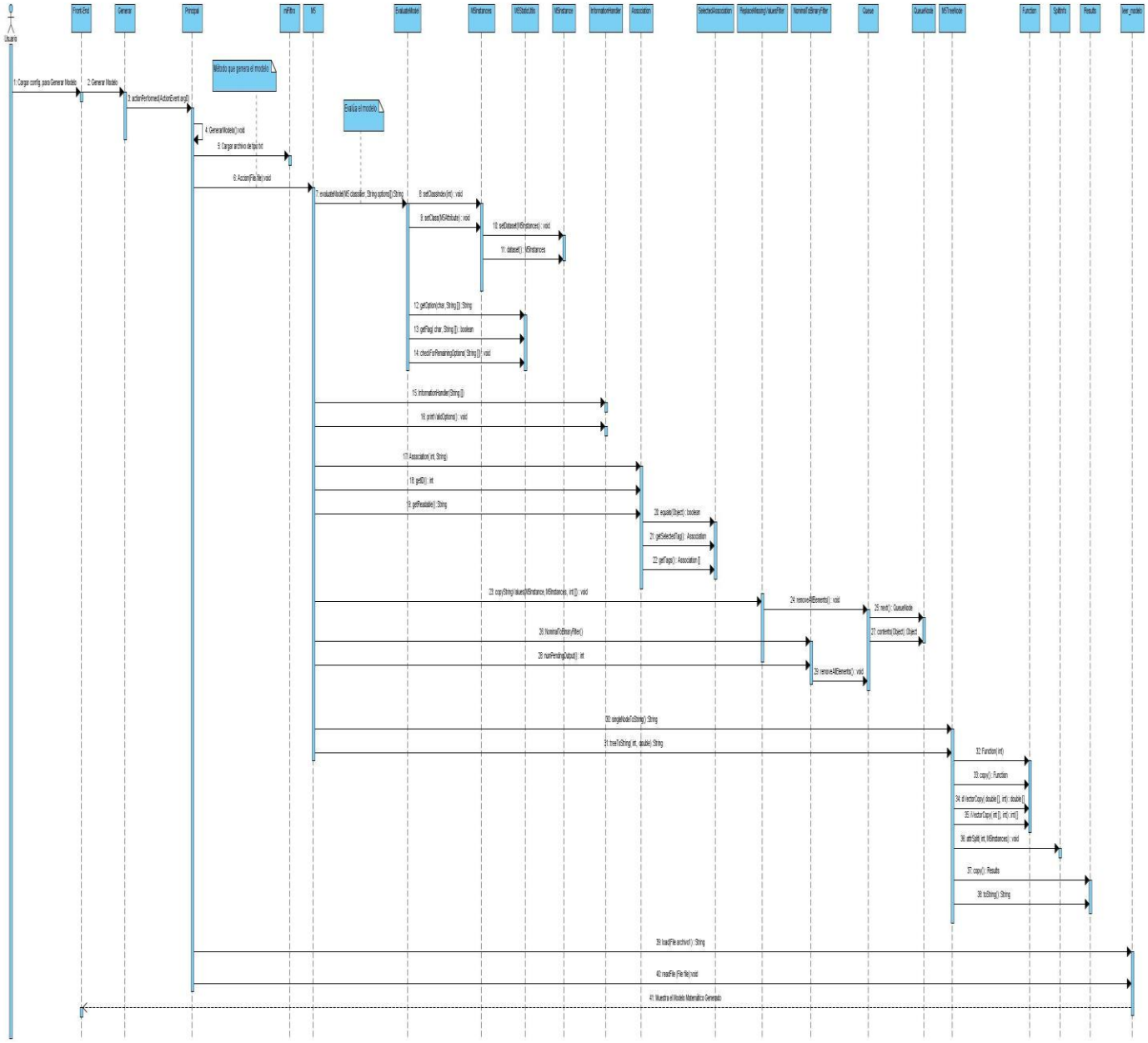
25. VISUAL PARADIGM FOR UML 7.2 . *Visual Paradigm for UML 7.2 .* [En línea] [Citado el: 19 de 04 de 2010.] http://www.visual-paradigm.com/product/vpuml/.

26. *VISUAL PARADIGM FOR UML.* Programación en castellano, Última actualización: 5 de julio de 2005. [Citado el: 5 de 03 de 2010]. Disponible en: http://www.programacion.com/noticia/1363/.

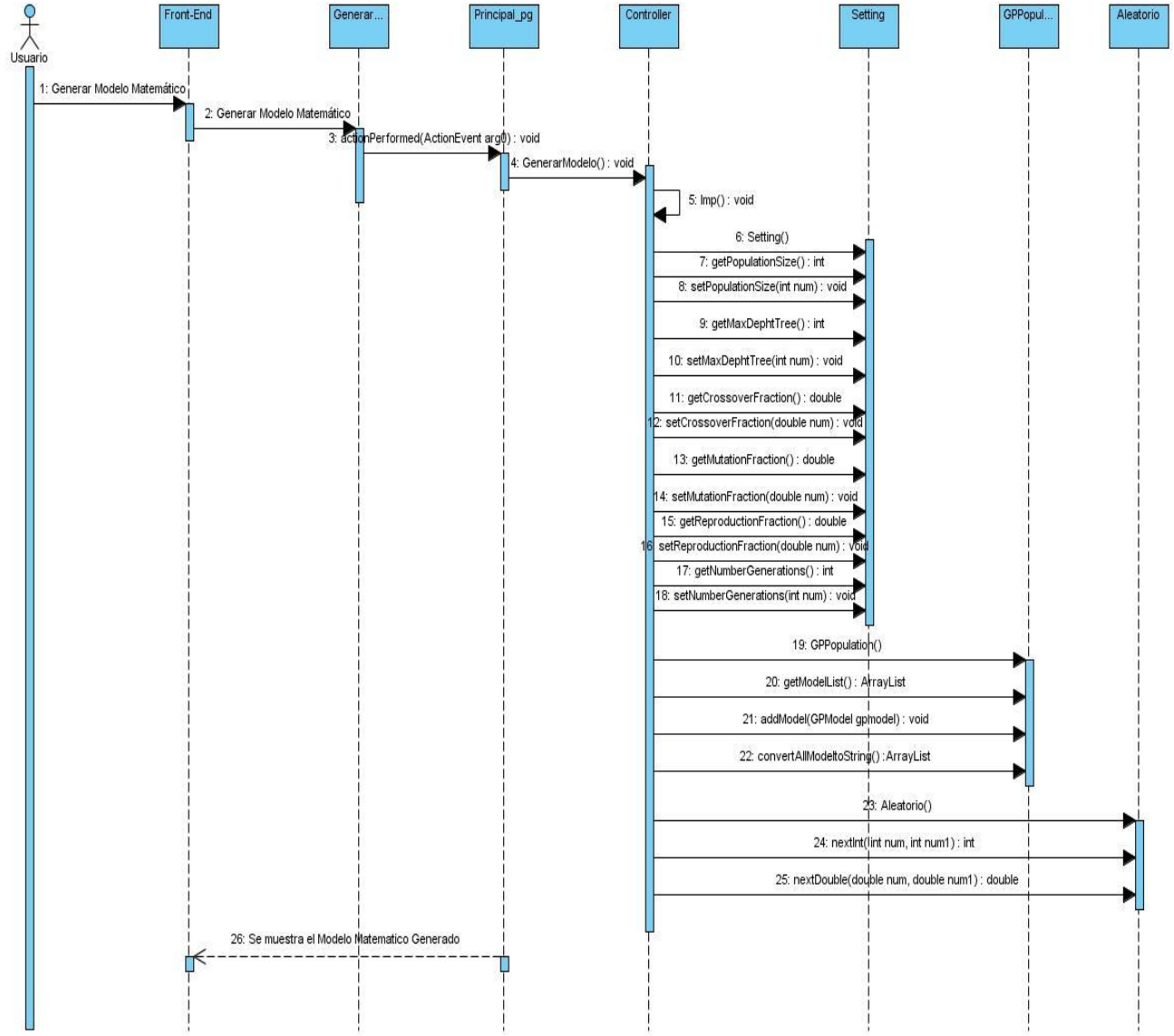
27. **WANG, Y.** Induction of model trees for predicting continuos classes. *European Conference on Machine Learning.* Europa: s.n., 1997

ANEXOS

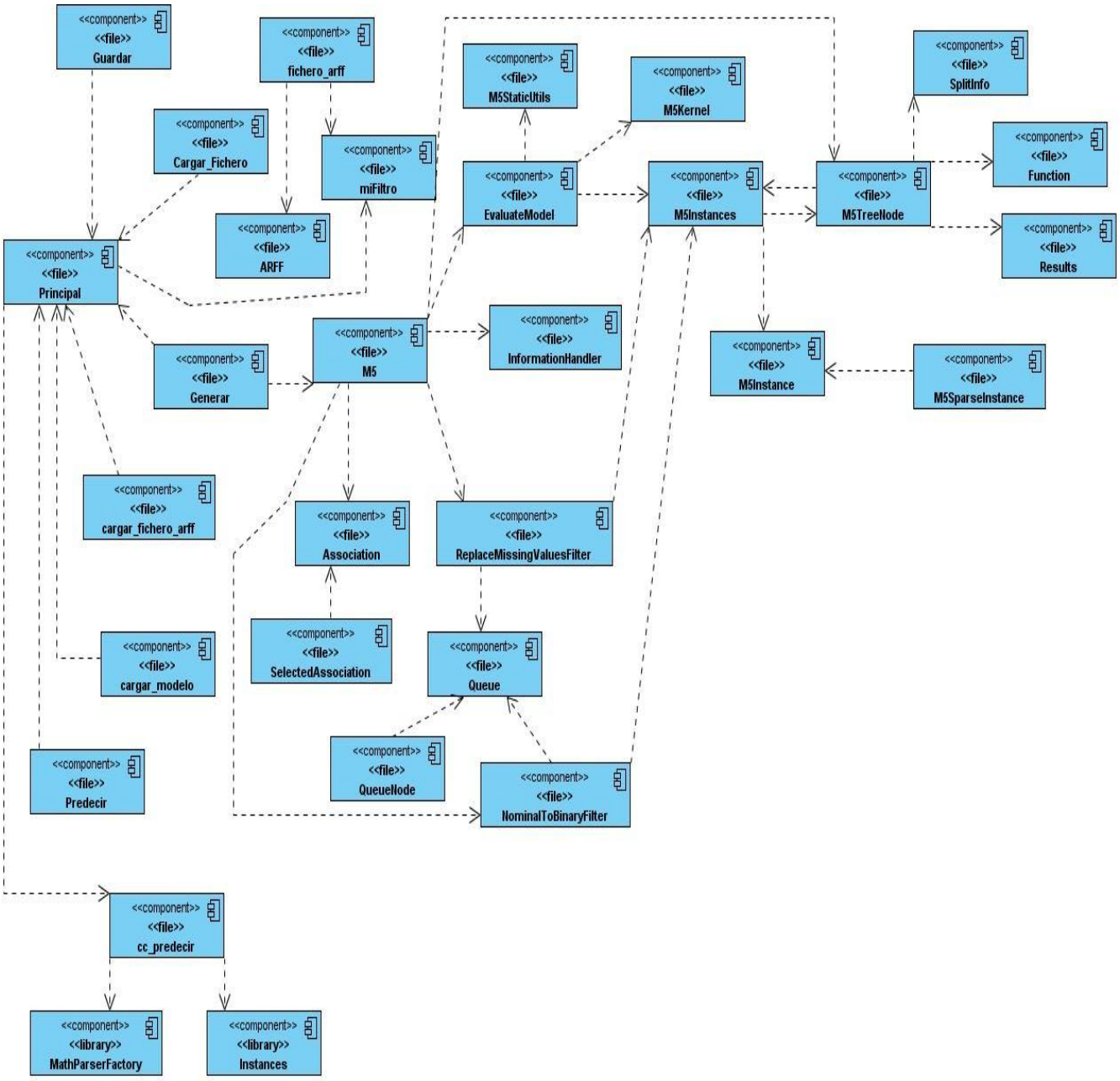
Anexo1.Diagrama de secuencia para el caso de uso Generar Modelo del plug-in de Árboles de Regresión.



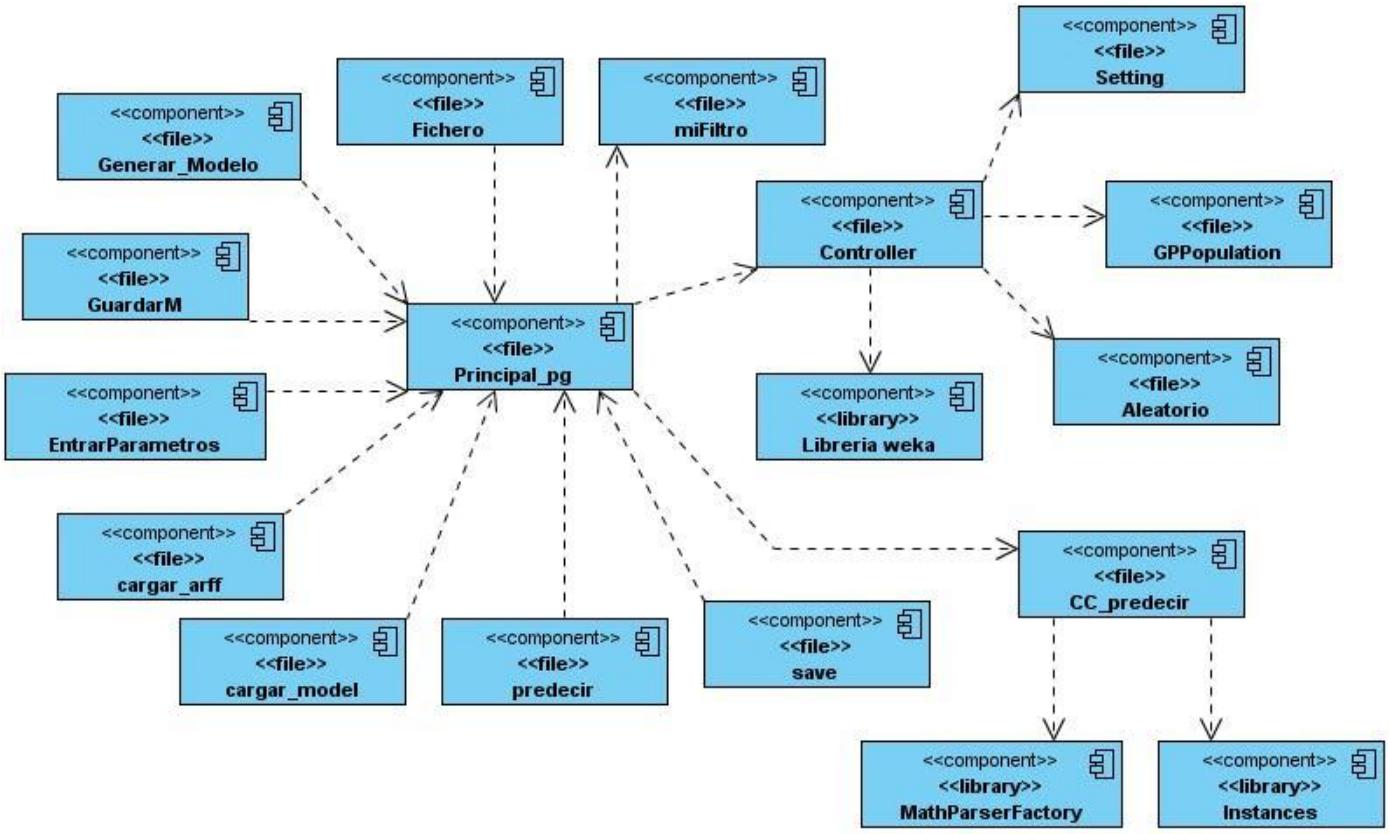
Anexo2.Diagrama de secuencia para el caso de uso Generar Modelo del plug-in de Programación Genética.



Anexo 3. Diagramas de Componentes para el plug-in de Arboles de Regresión



Anexo 4.Diagramas de Componentes para el plug-in de Programación Genética.



GLOSARIO

Moléculas: Una molécula es una partícula formada por un conjunto de átomos ligados por enlaces covalentes.

Descriptor: Número que describe la estructura química o una propiedad de la molécula o fragmento de esta.

Fragmento: Una pequeña parte de la molécula a la cual se llega a través de un método que es el encargado de fragmentarla.

Actividad biológica: Actividad que caracteriza el comportamiento biológico en compuestos químicos (Molécula o Fragmento)

Plug-in: Es una aplicación informática que interactúa con otra aplicación para aportarle una función o utilidad específica, generalmente muy específica, como por ejemplo servir como driver en una aplicación, para hacer así funcionar un dispositivo en otro programa.

CASE: Acrónimo inglés de Computer Aided Software Engineering, que significa Ingeniería de Software Asistida por Ordenador**.**

XML: Metalenguaje extensible de etiquetas, desarrollado por el World Wide Web Consortium (W3C).

Bioinformática: Es la aplicación de los ordenadores y los métodos informáticos en el análisis de datos experimentales y simulación de los sistemas biológicos.

UML: Lenguaje Unificado de Modelado.