Universidad de las Ciencias Informáticas

Facultad 6

Título: "Desarrollo de un plug-in para el Perceptrón Multicapa en la plataforma alasGRATO."

Trabajo de Diploma para optar por el título de
Ingeniero Informático

Autores: Karelia M. Del Pino Medina.
Surama Columbié Tomás.

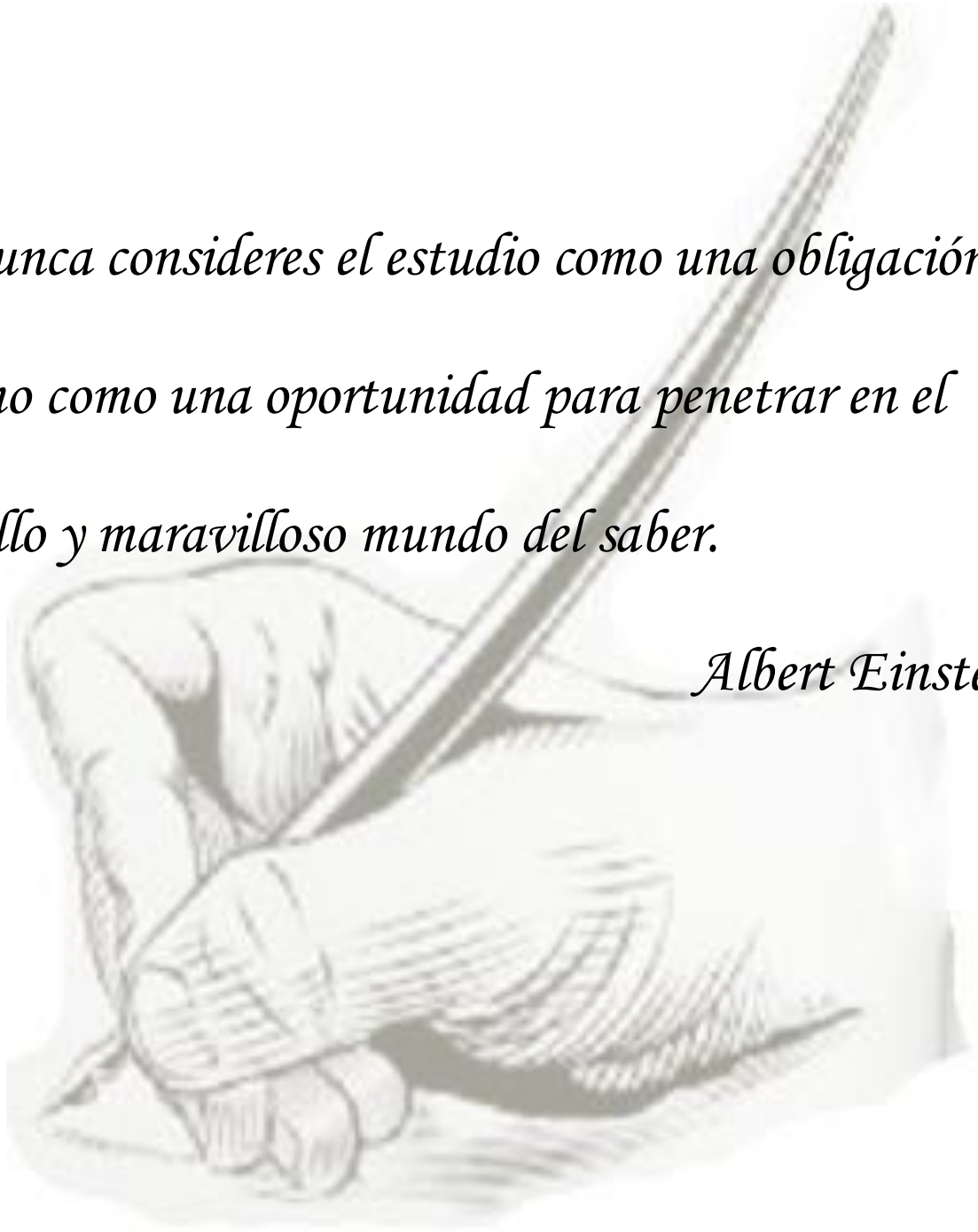
Tutores: Dr. Ramón Carrasco Velar.
Msc. Yuleydis Mejias Cesar.

Consultante: Ing. Isbel Ochoa Izquierdo.

Ciudad de La Habana, junio del 2010
"Año 52 de la Revolución"

Nunca consideres el estudio como una obligación, sino como una oportunidad para penetrar en el bello y maravilloso mundo del saber.

Albert Einstein



Declaración de Autoría

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 _____.

Karelia M. Del Pino Medina

_____
Firma del Autor

Surama Columbié Tomás

_____
Firma del Autor

Dr. Ramón Carrasco Velar

_____
Firma del Tutor

Msc. Yuleydis Mejías Cesar

_____
Firma del Tutor

DATOS DE CONTACTO

Tutores**:**

Dr. Ramón Carrasco Velar.
Universidad de las Ciencias Informáticas, Habana, Cuba.
rcarrasco@uci.cu

Msc. Yuleydis Mejias Cesar.
Universidad de las Ciencias Informáticas, Habana, Cuba.
ymejias@uci.cu

Consultante:

Ing. Isbel Ochoa Izquierdo.
Universidad de las Ciencias Informáticas, Habana, Cuba.
iochoa@uci.cu

AGRADECIMIENTOS

Queremos agradecer primero que todo a la Revolución por habernos dado la posibilidad de incorporarnos a la enseñanza escolar desde que somos pequeñitos y comenzamos a asistir al Círculo Infantil hasta esta maravillosa Universidad que es la UCI donde finalmente nos formamos como profesionales y más que todo como seres humanos. A nuestros padres por ser el motor impulsor para que llegáramos a donde nos encontramos hoy. A todos nuestros profesores por ser nuestros formadores profesionales. A Carrasco, Yuly, Isbel, Luis Gabriel, Dairon, Roberto, Edel Moreno, Lianet. A todos nuestros amigos y en general a todas esas personas que de alguna forma u otra influyeron en el éxito alcanzado con este trabajo y que en algún momento se preocuparon por saber cómo marchaba la tesis o que nos faltaba.

DEDICATORIA

De Karelia M. Del Pino Medina.

A mis padres Milagros Medina Jardines y Rolando Del Pino Calzada por ayudarme siempre, por haberme brindado apoyo, amor, cariño y porque gracias a su esfuerzo, todo lo que soy hoy se los debo a ellos.

A toda mi familia que siempre han contribuido de una forma u otra en mi formación y siempre me han brindado su apoyo incondicional con amor.

A mi novio Oscar Paredes Téllez por ayudarme, brindarme su amor y comprensión en los momentos más difíciles.

A todas las personas que de una forma u otra han contribuido a convertirme en lo que soy hoy.

Dedicatoria

De Surama Columbié Tomás.

A mi mamita que siempre ha estado a mi lado, dándome todo su apoyo y energía positiva para seguir adelante, a ella que me sostuvo en los momentos más difíciles desde pequeñita hasta ahora, sobre todo en los momentos en que pensé que ya no podía más, gracias por darme tus bendiciones cada día, quiero que sepas que te agradezco mucho el apoyo que siempre me has dado !Me has dado una lección de vida!

A mi papá que lo quiero con todo mi corazón al igual que a mi mamita, que mas que un padre ha sido un hermano, un amigo para mi, que me ha dejado valerme por mi misma para que pudiera aprender de mi experiencia, él que en cada problema me tendió la mano siempre el primero, que siempre me trato de comprender y ayudar.

Quiero que sepan que si estoy aquí en esta posición es gracias a ustedes, más que por mi por un compromiso de amor y de lealtad, sin su apoyo sin su ejemplo no estaría ahora aquí cumpliendo mis sueños, USTEDES SON MI EJEMPLO MAS GRANDE. Son el ejemplo que quiero seguir y es a ustedes a los que quiero premiar con mi esfuerzo, gracias por sacrificarse y darme todo, sobre todo el amor y cariño que siempre me han regalado.

A mi hermano, a mi sobrino, a Damaris, a mis padrinos que los amo con todo mi corazón, a mi abuelito Roque, a mis primos, Javier, Amilkar, Mercedita, Luis Davis, Syimi que han sido como mis hermanos. A todas aquellas personas que han contribuido para que cada día fuera una mejor persona.

RESUMEN

En los últimos tiempos se ha estado desarrollando un nuevo campo de las ciencias de la computación, comprendido por aquellos métodos y técnicas de resolución de problemas que no pueden ser fácilmente descritos por algoritmos tradicionales. Este campo dispone de un conjunto variado de técnicas, tales como: la lógica difusa, los algoritmos genéticos, las redes neuronales artificiales (RNA), entre otras. En el presente trabajo se abordará acerca de las RNA y específicamente el Perceptrón Multicapa (PMC). El Perceptrón Multicapa es una de las redes que mejores resultados ha aportado en los estudios de relación estructura-actividad. En el proyecto alasGRATO de la Universidad de las Ciencias informáticas no existe un plug-in visual que permita realizar el entrenamiento desde la plataforma, aunque existen los algoritmos necesarios que permiten el entrenamiento de las redes para la predicción de la actividad biológica, por lo que se hace necesario el desarrollo de una interfaz visual que sea amigable para el usuario y crear una base de datos que de la posibilidad de guardar las redes entrenadas para su posterior utilización en nuevos compuestos asociados al mismo ensayo.

PALABRAS CLAVE

Base de datos, redes neuronales artificiales, Perceptrón Multicapa, plug-in.

TABLA DE CONTENIDOS

INTRODUCCIÓN

A lo largo de los años los científicos han realizado numerosos estudios para poder comprender el funcionamiento del cerebro humano. Estos estudios han contribuido al desarrollo de nuevos sistemas relacionados con el tratamiento de la información, que permitan solucionar problemas cotidianos, tal como lo hace el cerebro humano. [1]

Actualmente se utilizan un conjunto de técnicas diferentes a las tradicionales para dar solución a las interrogantes del hombre. A este conjunto de técnicas y herramientas se les bautizó con el nombre de Inteligencia Artificial (IA) porque lo que se pretendía era que las computadoras presentaran un comportamiento inteligente, es decir, que supieran responder frente a ciertos problemas de una manera similar a como lo hacen los seres humanos.

Dentro de las técnicas de la IA podemos hacer referencia a las Redes Neuronales Artificiales (RNA), las cuales imitan en cierto modo la estructura física y el modo de operación del cerebro humano. Las RNA son capaces de aprender de la experiencia, de generalizar de casos anteriores a nuevos casos, o de abstraer características esenciales a partir de entradas que representan información aparentemente irrelevante. [2]

Las RNA permiten además darle solución a problemas de gran complejidad, un ejemplo de esto es el reconocimiento de formas o patrones, la predicción de las reacciones adversas en los medicamentos y muchas otras aplicaciones.

Una de las aplicaciones más importantes del trabajo con las RNA, es la predicción de la actividad biológica de compuestos orgánicos, la cual resulta ser un objetivo fundamental dentro de la Industria Médico Farmacéutica Mundial.

Existen diversos software para el trabajo con las RNA, un ejemplo de estos es el Joone (Java Object Oriented Neural Engine por sus siglas en inglés), una plataforma de software libre capaz de crear, entrenar y probar redes neuronales artificiales. El Javanns (Java Neuronal Network Simulator) es un eficiente simulador universal de redes neuronales artificiales, entre otros.

En la facultad 6 de la Universidad de las Ciencias Informáticas se está trabajando en el campo de la predicción de la actividad biológica en el proyecto denominado “Plataforma Inteligente para la Predicción de Actividad Biológica de Compuestos Orgánicos”, donde se hace uso de varias técnicas para predecir la actividad biológica, tales como: Máquinas de Soporte Vectorial, Lógica Difusa, Programación Genética y las RNA.

Actualmente no existe en el proyecto un plug-in visual que permita realizar el entrenamiento de las redes para la predicción de la actividad biológica desde la plataforma, aunque existen los algoritmos necesarios que permiten entrenar las redes, por lo que se hace necesario el desarrollo de una interfaz visual que sea amigable para el usuario. Esta aplicación le dará al usuario la posibilidad de entrenar, guardar las redes entrenadas, así como cargar una red de la base de datos para poder predecir la actividad biológica de nuevos compuestos asociados al mismo ensayo, pues en la bibliografía consultada no se reportó ningún software que contara con una base de datos que permita guardar las redes entrenadas para posteriormente realizar la predicción de la actividad biológica.

Por todo lo anteriormente planteado es que se define como **Problema científico:**

¿Cómo integrar una aplicación visual en forma de plug-in para Perceptrón Multicapa en la plataforma alasGRATO?

Para ello la investigación centra su **objeto de estudio** en las aplicaciones visuales para realizar el entrenamiento y predicción usando Redes Neuronales Artificiales, y el **campo de acción** plug-in para realizar el entrenamiento y predicción usando Redes Neuronales Artificiales en la plataforma alasGRATO.

Se plantea como **Objetivo de la investigación:**

Desarrollar un plug-in para el Perceptrón Multicapa que permita realizar el entrenamiento de las Redes Neuronales Artificiales y la predicción de la actividad biológica en la plataforma alasGRATO.

Teniendo como **Objetivos Específicos**:

- Implementar una aplicación visual para realizar el entrenamiento del Perceptrón Multicapa y la predicción de la actividad biológica.
- Crear una base de datos que permita guardar los entrenamientos realizados.
- Desarrollar el diseño de la aplicación.

Para lograr cumplir satisfactoriamente todos los objetivos se trazaron varias **Tareas:**

- Análisis del estado del arte de las herramientas existentes para el entrenamiento y predicción usando Redes Neuronales Artificiales.
- Estudio del algoritmo utilizado para el entrenamiento.
- Elaboración del modelo de dominio.
- Definición de los requisitos funcionales y no funcionales de la herramienta.

- Desarrollo de diagramas de casos de uso del sistema.
- Descripción de los casos de uso del sistema.
- Desarrollo del diagrama de clases del diseño.
- Creación de una base de conocimiento para guardar el entrenamiento.
- Implementación del plug-in.
- Evaluar el sistema.

Después de concluido este trabajo se podrá contar con los siguientes resultados:
Plug-in para Perceptrón Multicapa que le permita al usuario la selección de los elementos necesarios para realizar el entrenamiento del Perceptrón Multicapa y una base de datos donde se almacenen las redes entrenadas, de modo que puedan ser utilizados para la predicción de la actividad biológica de nuevos compuestos asociados al mismo ensayo.

Capítulo 1. "Fundamentación Teórica". En este capítulo se aborda el tema de las Redes Neuronales Artificiales, fundamentalmente el Perceptrón Multicapa, exponiéndose los conceptos fundamentales como el de base de datos, el de plug-in, las herramientas y metodologías necesarias para su implementación.

Capítulo 2. "Características del Sistema". Se realiza una breve descripción de la solución propuesta, los requisitos funcionales y no funcionales, así como los actores que intervienen en ella. Se presenta el modelo de dominio del sistema, el diagrama de caso de usos del sistema y la descripción de los mismos.

Capítulo 3. "Diseño del sistema". En este capítulo se describe el estilo arquitectónico utilizado para el desarrollo de la funcionalidad, se describen los patrones de diseño empleados, así como los diagramas de clases del diseño.

Capítulo 4. "Implementación y pruebas". En este capítulo se describe cómo fue implementada la herramienta en términos de componentes. Se desarrollarán varias pruebas de aceptación usando los distintos métodos para garantizar que se hayan cumplido los requisitos funcionales.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Este capítulo es el resultado de un estudio realizado sobre las Redes Neuronales Artificiales donde se presentarán sus principales características, ventajas, arquitectura, así como sus clasificaciones teniendo en cuenta diferentes aspectos. Se hará énfasis en el Perceptrón Multicapa como el tipo de RNA escogida para realizar la predicción de actividad biológica de compuestos orgánicos. Además, se aborda el tema relacionado con la base de datos, así como las herramientas y metodologías utilizadas y el concepto de plug-in y sus principales ventajas.

1.1. Redes Neuronales Artificiales

Las Redes Neuronales Artificiales son modelos de procesamiento de información basados en el funcionamiento del cerebro humano, el cual está compuesto por neuronas biológicas que son las responsables de transmitir el conocimiento a través de las conexiones que existen entre ellas.

Se puede ver una red neuronal como un grafo dirigido y ponderado donde cada uno de los nodos son neuronas artificiales y los arcos que unen los nodos son las conexiones sinápticas.

Al ser dirigido, los arcos son unidireccionales. Esto significa que la información se propaga en un único sentido, desde una neurona pre-sináptica (neurona origen) a una neurona post-sináptica (neurona destino). Al ser ponderado, las conexiones tienen asociado un número real, un peso, que indica la importancia de esa conexión con respecto al resto de las conexiones.

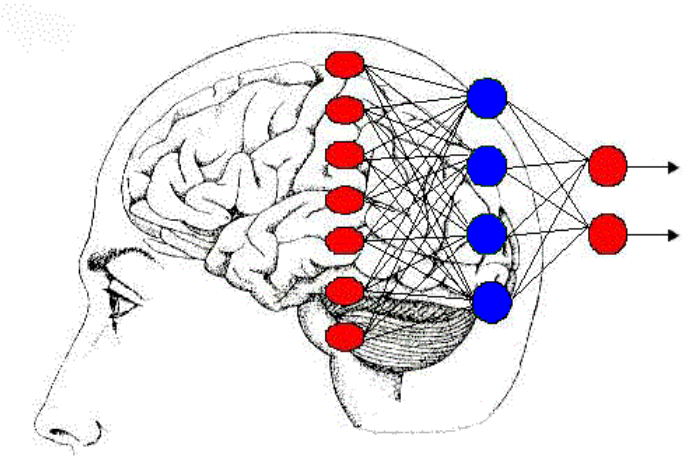


Figura 1. Las redes neuronales artificiales simulan el funcionamiento del cerebro humano.

Es importante conocer que los sistemas de cómputos tradicionales procesan la información de forma secuencial, no siendo así en el caso de las RNA las cuales no ejecutan instrucciones sino que estas responden en paralelo a las entradas que se les presentan. El resultado no se almacena en una sola posición de memoria sino que está distribuido por toda la red. [3]

1.1.1. La neurona artificial

La unidad básica de una RNA es la neurona. Uno de los primeros modelos abstractos para una neurona fue propuesto en 1943 por Warren McCulloch y Walter Pitts. Este modelo consistía en la suma de las señales de entrada, multiplicadas por unos valores de pesos escogidos aleatoriamente. La entrada es comparada con un patrón preestablecido para determinar la salida de la red. Si en la comparación, la suma de las entradas multiplicada por los pesos es mayor o igual que el patrón preestablecido la salida de la red es (1), en caso contrario la salida es (0).

Al inicio del desarrollo de los sistemas de inteligencia artificial, se creyó que este modelo podía computar cualquier función aritmética o lógica. [4]

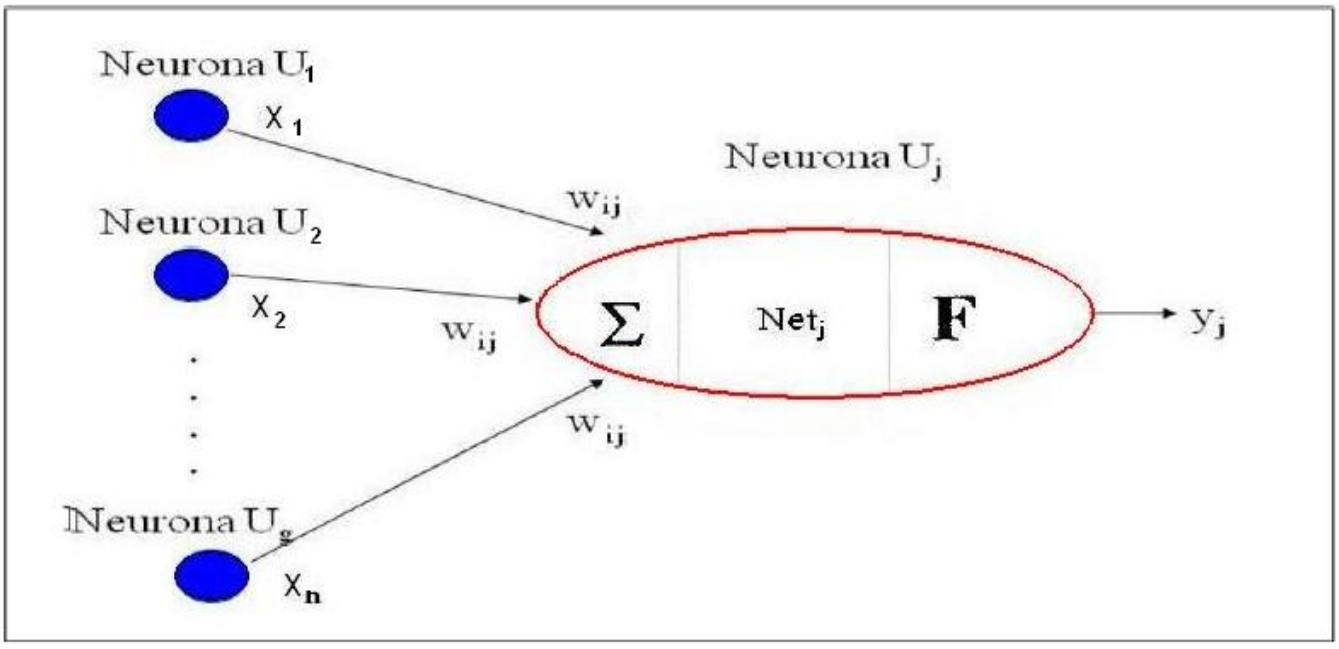


Figura 2. Estructura de una neurona artificial.

Los elementos clave de una neurona artificial son los siguientes:

- **Las entradas X_i:** reciben los datos de otras neuronas.
- **Los pesos sinápticos W_{ij}:** son las conexiones que unen a las neuronas. Es un número que se modifica durante el entrenamiento de la red neuronal, y es aquí por tanto donde se almacena la información que hará que la RNA adquiera conocimiento.
- **Una regla de propagación:** la operación que se utiliza frecuentemente para obtener el valor del potencial post sináptico es la suma ponderada, haciendo uso de las entradas y los pesos sinápticos. Esta operación consiste en sumar todas las entradas X_i (donde i toma valores desde 1 hasta la cantidad de neuronas existentes en la capa que se esté analizando) teniendo en cuenta el peso sináptico de cada una, W_{ij} (donde i y j toman valores desde 1 hasta n).

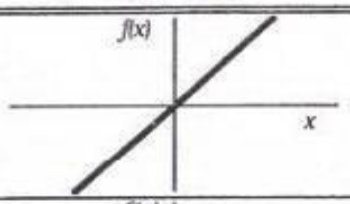
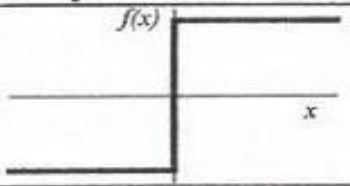
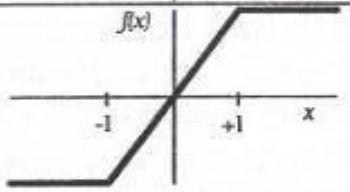
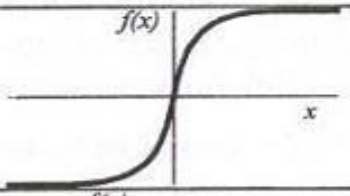
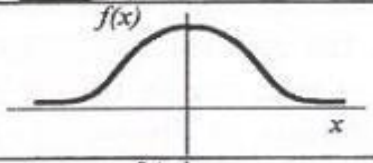
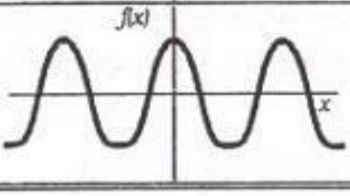
  Suma ponderada:

$$X_j = \sum_i y_i * w_{ij}$$

Ecuación 1.

- **Una función de activación F_j:** el valor obtenido con la regla de propagación, es utilizado para calcular la salida de la neurona. Esto se logra a partir de filtrar el valor obtenido con la regla de propagación usando una función conocida como función de activación o de transferencia. En dependencia del objetivo para el que se desee utilizar la red, se suele utilizar una función de activación u otra en ciertas neuronas de la red. En la tabla 1 se muestran las funciones de activación más usuales: [1]

Tabla 1. Funciones de activación más usuales

	Función	Rango	Gráfica
Identidad	$y = x$	$[-\infty, +\infty]$	
Escalón	$y = sign(x)$ $y = H(x)$	$\{-1, +1\}$ $\{0, +1\}$	
Lineal a tramos	$y = \begin{cases} -1, & si\ x < -l \\ x, & si\ +l \le x \le -l \\ +1, & si\ x > +l \end{cases}$	$[-1, +1]$	
Sigmoidal	$y = \frac{1}{1+e^{-x}}$ $y = tgh(x)$	$[0, +1]$ $[-1, +1]$	
Gaussiana	$y = Ae^{-Bx^2}$	$[0,+1]$	
Sinusoidal	$y = A\,\mathrm{sen}(\omega x + \varphi)$	$[-1,+1]$	

1.1.2. Arquitectura de una RNA

La estructura de una RNA está compuesta por varias capas ocultas, en las cuales se agrupan las neuronas. Aunque todas las capas son conjuntos de neuronas, según la función que desempeñan, suelen recibir un nombre específico. Las más comunes son las siguientes:

- **Capa de entrada:** las neuronas de la capa de entrada, reciben los datos que se proporcionan a la RNA para que los procese.
- **Capas ocultas:** estas capas introducen grados de libertad adicionales en la RNA. El número de ellas puede depender del tipo de red que se está considerando. Este tipo de capas realiza gran parte del procesamiento.
- **Capa de salida:** esta capa proporciona la respuesta de la red neuronal.

1.1.3. Características de las RNA

Dentro de las características de las RNA se pueden señalar las siguientes:

- Son un conjunto de unidades elementales, cada una de las cuales realiza un tipo de procesamiento muy simple.
- Tienen una densa estructura interconectada usando enlaces ponderados.
- Sus pesos deben ser ajustados para satisfacer los requerimientos de desempeño.[4]
- El sistema es distribuido. Esto quiere decir que la información no se almacena localmente en ciertas zonas concretas de la RNA sino que se halla presente por toda ella, en concreto, se almacena en la sinapsis entre las neuronas. Al calcular la respuesta de la red neuronal, intervienen todos y cada uno de los procesadores elementales, los cuales se hallan distribuidos por toda la arquitectura de la red.
- Presenta además un grado de adaptabilidad que se concreta en las capacidades de aprendizaje y generalización. Por aprendizaje se entiende la capacidad para recoger información de las experiencias y utilizarlas para actuar ante nuevas situaciones. Íntimamente relacionada con el aprendizaje esta la generalización, que podría definirse como la capacidad para abstraer la información útil, más allá de los casos particulares. De esta manera, la RNA es capaz de responder ante casos desconocidos. [1]

Estas características hacen que las RNA ofrezcan numerosas ventajas y que este tipo de tecnología se esté aplicando a múltiples áreas.

1.1.4. Ventajas de las RNA

Luego de conocer las características de las RNA se pueden citar dentro de sus ventajas las siguientes:

- **Aprendizaje Adaptativo:** capacidad de aprender a realizar tareas basadas en un entrenamiento o en una experiencia inicial.

- **Auto-organización:** una red neuronal puede crear su propia organización o representación de la información que recibe mediante una etapa de aprendizaje.
- **Tolerancia a fallos:** la destrucción parcial de una red, daña el funcionamiento de la misma, pero no la destruye completamente. Esto es debido a la redundancia de la información contenida.
- **Paralelismo:** las RNA ganan en tiempo ya que todas las neuronas intervienen en el proceso de análisis y procesamiento de la información, cada una de las cuales operan en paralelo.

1.1.5. Desventajas de las RNA

Existen también algunos aspectos desventajosos que hay que considerar al hacer uso de las RNA.
Entre sus desventajas podemos citar:

- Elevada duración de los tiempos de entrenamiento.
- El algoritmo de gradiente descendente puede alcanzar mínimos locales (es decir un mínimo en la función que relaciona los pesos con el error) más que un mínimo global.
- Una forma de evitar lo anterior es aumentar el número de neuronas ocultas dado que se supone que la RNA posee un escaso poder de representación interna.
- El algoritmo de gradiente descendente también puede oscilar y nunca alcanzar el resultado. Los valores de pesos que se aplican inicialmente y que suelen ser valores pequeños y aleatorios influyen en los resultados, es por ello se recomienda partir de valores iníciales aleatorios próximos a cero. [5]

1.1.6. Tipos de aprendizajes

Es importante señalar que la propiedad más importante de las redes neuronales artificiales es su capacidad de aprender a partir de un conjunto de patrones de entrenamientos, es decir, es capaz de encontrar un modelo que ajuste los datos. En el proceso de aprendizaje, también conocido como entrenamiento de la red, se dice que la red ha “aprendido” cuando los valores de los pesos se mantienen estables $dw_{ij}/dt = 0$. Existen varios tipos de aprendizaje, de los cuales se pueden mencionar los siguientes:

Aprendizaje supervisado: consiste en entrenar la red a partir de un conjunto de datos o patrones de entrenamiento compuesto por patrones de entrada y salida. El objetivo del algoritmo de aprendizaje es ajustar los pesos de la red (w_{ij}) de manera tal que la salida generada por la RNA sea lo más cercanamente posible a la verdadera salida dada una cierta entrada. Es decir, la red neuronal trata de encontrar un

modelo al proceso desconocido que generó la salida. Este aprendizaje se llama supervisado pues se conoce el patrón de salida el cual hace el papel de supervisor de la red.

Aprendizaje no supervisado: en el aprendizaje no supervisado se presenta sólo un conjunto de patrones a la RNA, y el objetivo del algoritmo de aprendizaje es ajustar los pesos de la red de manera tal que la red encuentre alguna estructura o configuración presente en los datos.

Aprendizaje reforzado: la base de este aprendizaje es muy parecida al aprendizaje supervisado, pero la información que se le proporciona a la red es mínima, se limita a indicar si la respuesta de la red es correcta o incorrecta. Este tipo de aprendizaje se basa en la noción de condicionamiento por refuerzo, estos se aprenden las conductas reforzadas positivamente y las conductas castigadas o reforzadas negativamente. [6]

1.2. Clasificación de las Redes Neuronales Artificiales

Se pueden clasificar las RNA teniendo en cuenta aspectos tales como: la transmisión de datos, número de conexiones, la topología, tipo de entrada, entre otros. [7]

1.2.1. Transmisión de datos

Redes feedforward, o con conexiones hacia adelante: contienen solo conexiones entre capas hacia delante. Esto implica que una capa no puede tener conexiones a una que reciba la señal antes que ella.

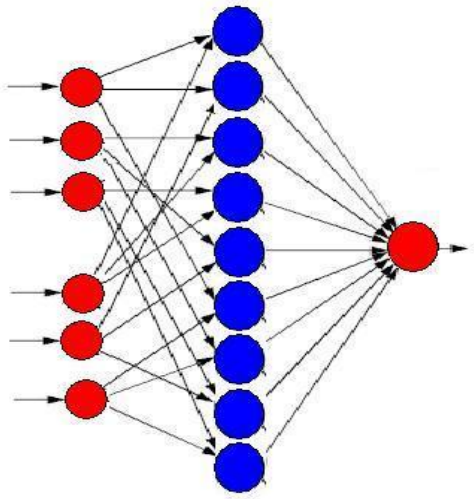


Figura 3. Redes feedforward.

Redes feedback, o con conexiones hacia atrás: aparte del orden normal algunas capas están también unidas desde la salida hasta la entrada en el orden inverso en que viajan las señales de información. Este tipo de redes se diferencia de las anteriores en que sí pueden existir conexiones de capas hacia atrás y por tanto la información puede regresar a capas anteriores en la dinámica de la red.

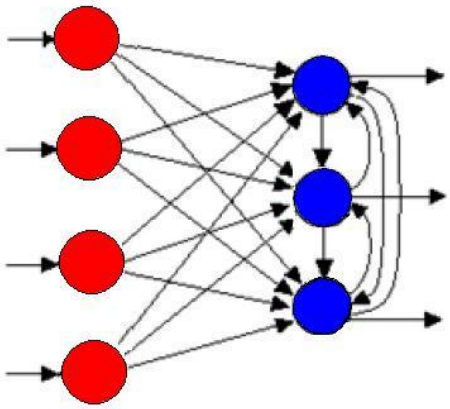


Figura 4. Redes feedback.

Redes feedlateral o con conexiones laterales: son conexiones entre neuronas de la misma capa, estos tipos de conexiones son muy comunes en las redes mono capa. [8]

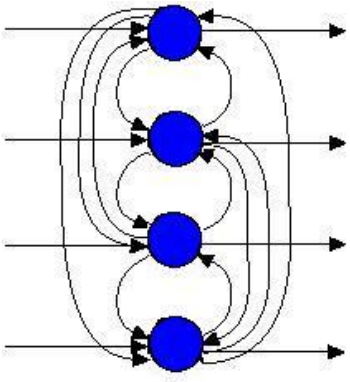


Figura 5. Redes feedlateral.

1.2.2. Número de conexiones

Redes neuronales totalmente conectadas: en este caso todas las neuronas de una capa se encuentran conectadas con las de la capa siguiente (redes no recurrentes) o con las de la anterior (redes recurrentes).

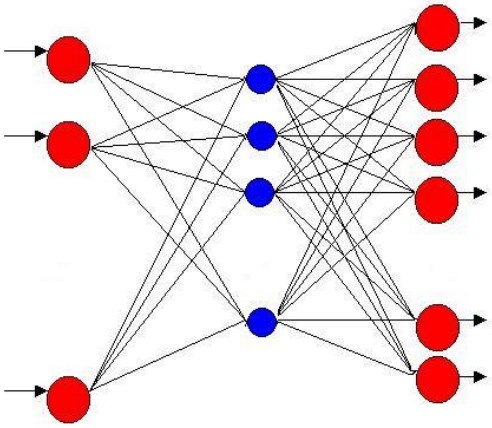


Figura 6. Redes neuronales totalmente conectadas.

Redes neuronales parcialmente conectadas: en este caso no se da la conexión total entre neuronas de diferentes capas. [7]

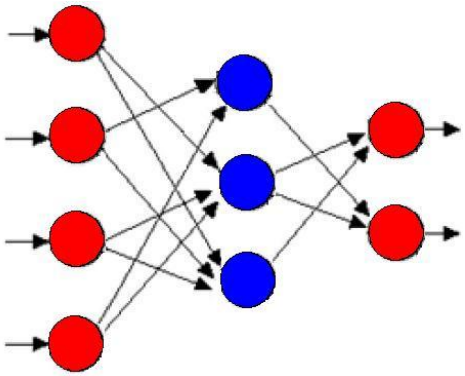


Figura 7. Redes neuronales parcialmente conectadas.

1.2.3. Topología

La topología de una red consiste en la organización y disposición de las neuronas en la red. Las neuronas se agrupan formando capas, que pueden tener muy distintas características. Además, las capas se organizan para formar la estructura de la red.

A continuación se verán las redes monocapa y multicapa:

Red monocapa: se corresponde con la red neuronal más sencilla ya que se tiene una capa de neuronas que proyectan las entradas a una capa de neuronas de salida donde se realizan diferentes cálculos. La capa de entrada, por no realizar ningún cálculo, no se cuenta, de ahí el nombre de redes neuronales con una sola capa.

En esta red monocapa Xi representa el vector de entrada i, U_j a las unidades intermedias, W_{ij} las conexiones o pesos que relacionan a la neurona i de la capa de entrada con la neurona j de la capa intermedia, donde las variables i y j, pueden tomar valores desde 1 hasta n.

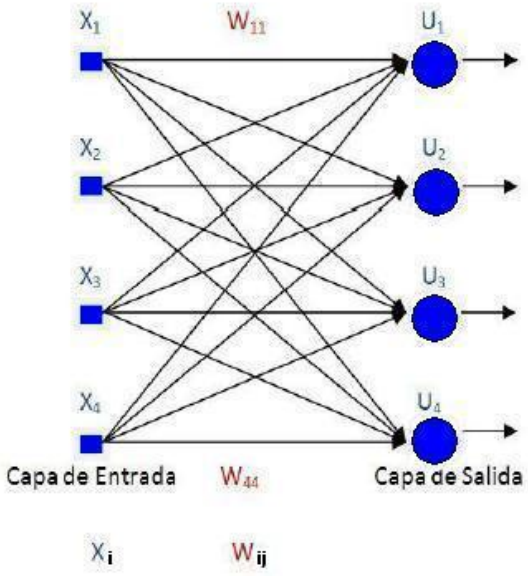


Figura 8. Red monocapa.

Redes Multicapa: una red multicapa consiste en una extensión de una red monocapa, con tantas capas ocultas como se desee.

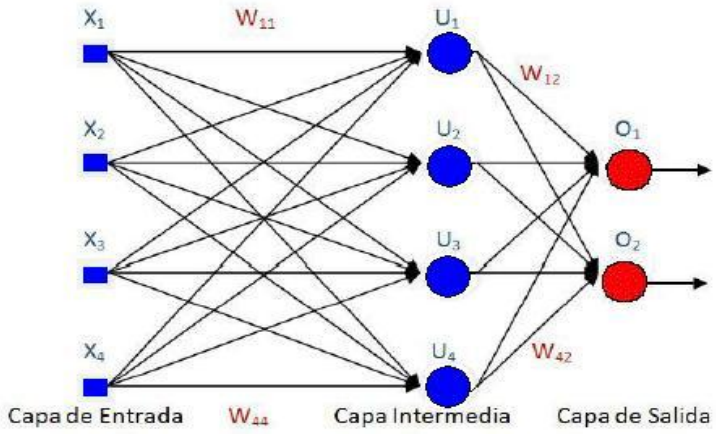


Figura 9. Red multicapa.

Donde X_i representa el vector de entrada i, U_j a las unidades intermedias, W_{ij} las conexiones o pesos que relacionan a la neurona i de la capa de entrada con la neurona j de la capa intermedia y están los pesos de la capa intermedia a las unidades de salida. Donde i y j pueden tomar valores desde 1 hasta n.

1.2.4. Tipo de Entrada

También se pueden clasificar las RNA teniendo en cuenta el tipo de información que procesan en: **Redes analógicas:** procesan datos de entrada con valores continuos y, habitualmente, acotados. Ejemplos de este tipo de redes son: Hopfield, Kohonen y las redes de aprendizaje competitivo. **Redes discretas:** procesan datos de entrada de naturaleza discreta; habitualmente valores lógicos booleanos. Ejemplos de este tipo de redes son: las máquinas de Bolzman y Cauchy, y la red discreta de Hopfield.[4]

1.3. Tipos de Redes Neuronales Artificiales

Existen diferentes tipos de RNA, entre ellas podemos citar:

- El Perceptrón Simple.
- La Red de Hopfield.
- El Perceptrón Multicapa.
- Red neuronal Competitiva Simple.
- Redes Neuronales Online ART1.

- Redes Neuronales competitivas ART2.
- Redes neuronales autoorganizadas: Mapas de Kohonen.

En la presente investigación se hace un estudio detallado del Perceptrón Multicapa, ya que existe la necesidad en el proyecto alasGRATO de tener un plug-in utilizando el algoritmo ya implementado con Perceptrón Multicapa además éste ha reportado resultados satisfactorios en problemas relacionados con la predicción de la actividad biológica.

1.3.1. Perceptrón Simple

La red tipo Perceptrón fue inventada por el psicólogo Frank Rosenblatt en el año 1957 basado en el modelo de McCulloch y Pitts. El primer modelo de Perceptrón fue desarrollado en un ambiente biológico imitando el funcionamiento del ojo humano, el fotoperceptrón como se le llamó, era un dispositivo que respondía a señales ópticas. El perceptrón es una red neuronal no lineal que consta de una combinación lineal entre las entradas y los pesos sinápticos, un valor de ajuste externo (umbral o tendencia) y cuenta además con una función de transferencia o de activación donde usualmente la que se utiliza es la función **signo:**

$$sgn(x) = \begin{cases} -1 ; x<0 \\ 1 ; x\geq 0 \end{cases}$$

Inicialmente es un dispositivo de aprendizaje, en su configuración inicial no tiene la capacidad de distinguir patrones de entrada muy complejos, sin embargo, mediante un proceso de aprendizaje puede adquirir esta capacidad.

El perceptrón está constituido por un conjunto de neuronas de entrada que reciben los patrones de entrada a reconocer o clasificar y una neurona de salida que se ocupa de clasificar a los patrones de entrada en dos clases, según que la salida de la misma sea 1 (activada) o 0 (desactivada). El perceptrón simple tiene una serie de limitaciones muy importantes. La más importante es su incapacidad para clasificar conjuntos que no son linealmente independientes. [9]

1.3.2. Perceptrón Multicapa

Este modelo es una ampliación del perceptrón a la cual añade una serie de capas que, básicamente, hacen una transformación sobre las variables de entrada, que permiten eludir el problema anterior. Las capas presentan interconexión total.

Esto acaba con el problema del perceptrón, convirtiendo las funciones linealmente no independientes en linealmente independientes gracias a la transformación de la capa oculta. Su modo de propagación de los datos es hacia adelante (feedforward), es decir, que los patrones de entrada se presentan en la capa de entrada y se propagan hasta generar la salida. Las entradas y las salidas son continuas [0,1]. La función de activación más usada en sus neuronas es la sigmoidal. Los estados de las unidades de cada capa son determinados aplicando las ecuaciones (2) y (3) a las conexiones provenientes de las capas anteriores. La entrada total X_j a la unidad U_j es una función lineal de las salidas Y_i de las unidades que están conectadas a U_j y de los pesos W_{ij} sobre estas conexiones. Para cada unidad U_j existe una entrada extra con valor 1 y peso W_{ij} que se trata como el resto.

$$X_j = \sum_i x_i * W_{ij} + b$$

Ecuación 2.

La salida de cada unidad de procesamiento, Y_j, se calcula usando una función no lineal de su entrada total teniendo en cuenta la función seleccionada. La ecuación de la función de activación sigmoidal es la siguiente:

$$Y_j = \frac{1}{1 + e^{-X_j}}$$

Ecuación 3.

Los perceptrones multicapa con aprendizaje backpropagation utilizan la regla Delta como forma de aprendizaje (Esta regla de aprendizaje, se fundamenta en la utilización del error entre la salida real y esperada de la red para modificar los pesos). Estas redes adaptan la regla Delta de tal forma, que se

facilite el entrenamiento de todas las conexiones entre los distintos niveles de la red. El Perceptrón Multicapa admite valores reales. Se puede decir que el Perceptrón Multicapa es un modelador de funciones universal.[4]

1.4. Método de aprendizaje Backpropagation

El método de aprendizaje backpropagation tiene como objetivo ajustar los pesos para reducir el error cuadrático medio de las salidas. Genera una señal de error a partir de la diferencia entre la salida actual de la red y la salida real. Los pesos (fuerzas sinápticas) se actualizan de forma proporcional al producto de la señal de error y la señal de entrada a la red, con lo cual se disminuye el error en dirección del gradiente (la dirección de cambio más rápido). El backpropagation pertenece a la categoría de supervisado, ya que requiere conocer las salidas correctas para cada ejemplo de entrada.

Este método es utilizado para el entrenamiento de una red multicapa. La red se inicializa con una serie de pesos aleatorios y tras hacer pasar los patrones de entrada a través de la estructura de la red se compara la salida con el patrón que se desea que aprenda.

Una vez que se ha aplicado un patrón a la entrada de la red como estímulo, éste se propaga desde la primera capa a través de las capas superiores de la red, hasta generar una salida. La señal de salida se compara con la salida deseada y se calcula una señal de error para cada una de las salidas.

Las salidas de error se propagan hacia atrás, partiendo de la capa de salida, hacia todas las neuronas de la capa oculta que contribuyen directamente a la salida. Sin embargo, las neuronas de la capa oculta sólo reciben una fracción de la señal total del error, basándose aproximadamente en la contribución relativa que haya aportado cada neurona a la salida original. Este proceso se repite, capa por capa, hasta que todas las neuronas de la red hayan recibido una señal de error que describa su contribución relativa al error total. Basándose en la señal de error percibida, se actualizan los pesos de conexión de cada neurona, para hacer que la red converja hacia un estado que permita clasificar correctamente todos los patrones de entrenamiento.

La importancia de este proceso consiste en que, a medida que se entrena la red, las neuronas de las capas intermedias se organizan a sí mismas de tal modo que las distintas neuronas aprenden a reconocer distintas características del espacio total de entrada.

La diferencia entre entradas-salidas, estimación de error, se utiliza para derivar los errores de los pesos de las sucesivas capas de la red.

La técnica de entrenamiento backpropagation requiere el uso de neuronas cuya función de activación sea continua, y por lo tanto, derivable.

Calcular el error E_j para todas las unidades donde D_j es el valor de la salida deseada y Y_j es el valor de salida real para la j-ésima unidad, los errores se calculan por:

Para las unidades de salida:

$$\mathbf{Ej = (Yj - Dj) * Yj * (1 - Yj)}$$

Ecuación 4.

Para las unidades en las capas ocultas:

$$Ej = Yj * (1 - Yj) * \sum_{k} Z_k * W_{kj}$$

Ecuación 5.

Donde k recorre todas las unidades conectadas a la unidad j en la capa siguiente a la que se encuentra esta y Z_k representa los errores calculados para las salidas de estas unidades (que fueron calculadas usando las ecuaciones (4) ó (5) en dependencia de si la capa en la que se encuentran las unidades indicadas por k es de salida u oculta respectivamente.

Existe una función matemática para realizar esta modificación de los pesos:

$$W_{ij}(n+1) = W_{ij}(n) + t * E_j * Y_i + h * (W_{ij}(n) - W_{ij}(n-1))$$

Ecuación 6.

Donde **(n+1)**, **(n)** y **(n-1)** indican el nuevo peso, el presente y el anterior respectivamente.

t es un número real que denota la velocidad de aprendizaje.

h es una constante entre 0 y 1 la cual determina el efecto de los cambios de pesos previos sobre la dirección actual de movimiento en el espacio de los pesos (momentum). [10]

1.4.1. Backpropagation con Momentum

Red Backpropagation con momentum: Esta modificación está basada en la observación de la última sección de la gráfica del error medio cuadrático en el proceso de convergencia típico para una red Backpropagation; este proceso puede verse en la figura 10 en la cual se nota la caída brusca del error en la iteración para la cual alcanza convergencia.

Figura 10. Comportamiento típico del proceso de convergencia para una red Backpropagation.

Este comportamiento puede causar oscilaciones no deseadas, por lo que es conveniente suavizar esta sección de la gráfica incorporando un filtro al sistema. Este algoritmo, hace que la convergencia sea estable e incluso más rápida, además permite utilizar una rata de aprendizaje alta.

La velocidad de aprendizaje es un valor que varía entre 0 y 1 dependiendo de las características del problema a solucionar, por lo general se escoge un número pequeño, para asegurar que la red encuentre una solución. Un valor pequeño de rata de aprendizaje significa que la red tendrá que hacer un gran número de iteraciones, si se toma un valor muy grande, los cambios en los pesos serán muy grandes, avanzando muy rápidamente por la superficie de error, con el riesgo de saltar el valor mínimo del error y estar oscilando alrededor de él, pero sin poder alcanzarlo. [11]

1.5. Base de datos

Una base de datos (BD) es un conjunto de información estructurada en registros y almacenada en un soporte electrónico legible desde un ordenador. Cada registro constituye una unidad autónoma de información que puede estar a su vez estructurada en diferentes campos o tipos de datos que se recogen en dicha base de datos. [12]

En el presente trabajo se crea una base de datos que permite almacenar las redes entrenadas para su posterior utilización en la predicción de la actividad biológica de nuevos compuestos orgánicos que estén asociados a un mismo ensayo. Teniendo en cuenta los gestores de base de datos que se utilizan en el proyecto se seleccionó SQLite para el desarrollo de la base de datos local.

1.5.1. Ventajas de la base de datos

A continuación se muestran algunas de las ventajas que ofrecerá la base de datos desarrollada:

- Permite un manejo simple y eficiente.
- Es propicia para seleccionar las redes entrenadas a la hora de realizar la predicción de la actividad biológica.
- Permite a los usuarios ver los resultados obtenidos en el entrenamiento para una red determinada.
- Es fácil de desarrollar y el costo de producción es bajo.
- Constituye un recurso confiable para el almacenamiento de las redes entrenadas.

1.6. Plug-in

Un plug-in es un módulo que añade una característica o servicio a un sistema ya existente; es además ejecutado por la aplicación principal. Los plug-ins le dan a una aplicación la capacidad de agregar funcionalidades nuevas en tiempo de ejecución. Los plug-ins tienen varias ventajas entre las cuales podemos mencionar las siguientes:

- Hacen posible personalizar la aplicación de maneras no pensadas por el autor, o que al menos no tenía la intención de hacer.
- Es posible compartir código entre aplicaciones sin mucho que ver entre ellas si estas comparten una infraestructura de plug-ins entre sí.
- Permite que los desarrolladores externos colaboren con la aplicación principal extendiendo sus funciones.
- Hace posible que sólo cierta funcionalidad esté disponible, si alguna librería o aplicación de terceros no está disponible en el sistema.[13]

1.6.1. Ventajas del plug-in para el Perceptrón Multicapa

Dentro de las ventajas que brindará el plug-in se encuentran las siguientes:

- Flexibilidad, ya que el plug-in se puede usar en distintos sistemas operativos.
- Permite realizar el entrenamiento de redes neuronales artificiales.
- Permite guardar las redes entrenadas en una base de datos.
- Permite realizar la predicción de la actividad biológica de nuevos compuestos orgánicos asociados a un mismo ensayo.

- Permite mostrar los resultados tanto de las redes entrenadas como de la predicción de la actividad biológica.

1.7. Herramientas y metodologías utilizadas

1.7.1. Metodología: OpenUP

Una metodología de desarrollo de software ofrece un conjunto de técnicas y procedimientos que permiten conocer la organización de los elementos necesarios para definir un proyecto de software. Teniendo en cuenta la metodología que se adapta más al medio en que se desarrolla el software y a los objetivos finales que se quieren cumplir se seleccionó el Proceso Unificado Abierto (OpenUP) para el presente trabajo.

OpenUP es un proceso unificado que aplica acercamientos iterativos e incrementales dentro de un ciclo de vida estructurado. Además, es ágil y se enfoca en la naturaleza de colaboración del desarrollo del software, es una herramienta que se puede ampliar a una gran variedad de proyectos. El ciclo de vida de un proyecto en OpenUP está estructurado en cuatro fases: inicio, elaboración, construcción y transición.

Con OpenUP se logra una mayor productividad por parte del equipo de trabajo, ya que se definen claramente actividades, roles y sus responsabilidades, el desarrollo es iterativo e incremental, es guiado por casos de uso, permite gestionar riesgos, y su enfoque es centrado en la arquitectura. [14]

1.7.2. Lenguaje de modelado: UML

Se seleccionó para el presente trabajo el Lenguaje de Modelado Unificado (UML - Unified Modeling Language) ya que es un lenguaje que permite modelar, construir y documentar los elementos que forman un producto de software que responde a un enfoque orientado a objetos.

UML está compuesto por diversos elementos gráficos que se combinan para conformar diagramas. Debido a que el UML es un lenguaje, cuenta con reglas para combinar tales elementos, además permite la modelación de sistemas con tecnología orientada a objetos. Es importante recalcar que UML no es una guía para realizar el análisis y diseño orientado a objetos, es decir, no es un proceso. En lo que corresponde al desarrollo de programas, posee elementos gráficos para soportar la captura de requisitos, el análisis, el diseño, la implementación, y las pruebas. UML describe lo que supuestamente hará un sistema, pero no dice cómo implementarlo, sus objetivos son ofrecer un lenguaje simple y legible que permitiera modelar aplicaciones en cualquier dominio, y generar de manera automática código fuente.

UML permite que diseñadores diferentes modelen sistemas diferentes y puedan ampliamente entender cada uno los diseños de los otros. [15]

1.7.3. Lenguaje de programación: Java

Java es el lenguaje seleccionado para el presente trabajo ya que se utiliza en el proyecto. Con el uso del mismo se realizará la implementación del plug-in visual para el Perceptrón Multicapa. Java es un lenguaje de propósito general, concurrente, basado en clases y orientado a objetos. Una de sus características fundamentales es que es multiplataforma (Windows, Unix y Mac) debido a que todas sus ejecuciones y compilaciones son ejecutadas sobre una máquina virtual, por lo que no depende de la arquitectura de la máquina donde se ejecute. Otra de sus características es que Java posee una arquitectura neutral, es decir, el compilador Java compila su código a un fichero objeto de formato independiente a la arquitectura de la máquina en que se ejecutará. Cualquier máquina que tenga el sistema de ejecución (run-time) puede ejecutar ese código objeto, sin importar de ningún modo la máquina en que ha sido generado. [16]

1.7.4. Entorno de desarrollo: Eclipse

Eclipse es la herramienta IDE definida para el trabajo en la plataforma alasGRATO, es por esto que será la herramienta utilizada en este trabajo. Es un entorno de desarrollo integrado, distribuido, de código abierto y multiplataforma. Al igual que el lenguaje de programación Java puede ejecutarse en sistemas operativos con características diversas. La arquitectura basada en plug-in permite integrar varios lenguajes de programación e introducir otras aplicaciones complementarias. Admite la incorporación de otros plug-ins para obtener un mayor número de funcionalidades. Tiene, además, los beneficios siguientes:

- Soporta herramientas que manipulan diferentes tipos de lenguajes, como por ejemplo Java, C, C++.
- Corre en una gran cantidad de sistemas operativos incluyendo Windows y Linux.
- Provee a los desarrolladores, herramientas que facilitan la creación de plug-in. [17]

  En la presente investigación se trabaja sobre el eclipse 3.4.

1.7.5. Herramientas CASE: Visual Paradigm

Como herramienta CASE se empleó Visual Paradigm para el trabajo con UML. La herramienta está diseñada para una amplia gama de usuarios que incluye a: ingenieros de software, analistas de sistemas, analistas de negocios y arquitectos de sistemas, interesados en la creación de grandes sistemas de

software de manera confiable, a través del paradigma Orientado a Objetos. VP-UML soporta los últimos estándares de anotaciones de Java y UML y provee soporte para la generación de código y la ingeniería inversa para Java. Además, VP-UML se integra con Eclipse, Borland® JBuilder®, NetBeans IDE/Sun™ ONE, IntelliJ IDEA™, Oracle JDeveloper y BEA WebLogic Workshop™, para soportar las fases de implementación en el desarrollo de software. Las transiciones del análisis al diseño, y de éste a la implementación, están adecuadamente integradas dentro de la herramienta CASE, de manera que reduce significativamente los esfuerzos de todas las etapas del ciclo de desarrollo de software. [18]

1.7.5.1. Gestor de base de datos: SQLite

Un gestor de base de datos o sistema de gestión de base de datos (SGBD) es un software que permite introducir, organizar, recuperar y administrar la información de la base de datos, además permiten mantener una base de datos, asegurando su integridad, confidencialidad y seguridad. A continuación se aborda el tema referente al gestor de base de datos SQLite, utilizado para guardar las redes entrenadas.

SQLite tiene la capacidad de reemplazar a grandes motores de bases de datos y acoplarse al desarrollo de nuestros proyectos informáticos, ya sea en ambientes de prototipos de sistemas como en complejos y robustos software. [19]

Una de las primeras diferencias entre los motores de bases de datos convencionales es su arquitectura cliente/servidor, pues SQLite es independiente.

Se pueden destacar las siguientes características:

- **Tamaño:** SQLite tiene una pequeña memoria y una única biblioteca es necesaria para acceder a bases de datos, lo que lo hace ideal para aplicaciones de bases de datos incorporadas.
- **Rendimiento de base de datos:** SQLite realiza operaciones de manera eficiente y es más rápido que MySQL y PostgreSQL.
- **Portabilidad**: se ejecuta en muchas plataformas y sus bases de datos pueden ser fácilmente portadas sin ninguna configuración o administración.
- **Estabilidad:** SQLite es compatible con ACID, reunión de los cuatro criterios de atomicidad, consistencia, aislamiento y durabilidad.
- **Costo:** SQLite es de dominio público, y por tanto, es libre de utilizar para cualquier propósito sin costo y se puede redistribuir libremente.

1.7.6. Front-End GRATO

El Front-End GRATO surgió como una necesidad del Proyecto alasGRATO en el Polo de Bioinformática de la Facultad 6 de la Universidad de las Ciencias Informáticas para mejorar la interacción de los usuarios con la plataforma.

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

- 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.
- Unificar la carga de los plug-ins a través de descriptores XML.
- Mantener el principio de conservación, pues en la evolución del Front-End, los plug-ins. desarrollados siempre serán compatibles.

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 plug-ins, para la integración de los módulos de predicción de la actividad biológica utilizando Redes Neuronales Artificiales. Teniendo en cuenta también que éste fue creado específicamente para la plataforma GRATO, con el objetivo de incluirle en forma de plug-in todos los módulos existentes en la misma y que su uso es mucho más sencillo ya que posee una interfaz amigable y configurable, permitiéndole al usuario cambiar la apariencia (ya sea color o idioma) de acuerdo a la necesidad del mismo.

1.8. Conclusiones parciales

En este capítulo se realizó una breve introducción sobre las Redes Neuronales Artificiales presentando sus principales características, ventajas, arquitectura, así como sus clasificaciones teniendo en cuenta diferentes aspectos. Se hizo énfasis en el Perceptrón Multicapa como el tipo de RNA escogida para realizar la predicción de actividad biológica de compuestos orgánicos, además se definieron las metodologías y herramientas que se utilizan para desarrollar el presente trabajo, se decidió utilizar como lenguaje de programación Java, el entorno de desarrollo a utilizar será el Eclipse, como gestor de Base de datos SQLite, como herramientas CASE Visual Paradigm, el lenguaje de modelado UML y la metodología OpenUP.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

En este capítulo se realiza la descripción de las principales definiciones asociadas al dominio del problema. También se especifican los requisitos funcionales y no funcionales. Se identifican los actores y casos de uso del sistema así como las relaciones entre ellos mediante el diagrama de casos de uso del sistema y las descripciones textuales de cada caso de uso.

2.1. Modelo de dominio

El modelo del dominio es una representación de los conceptos u objetos del mundo real, significativos para un problema. Tiene como objetivo fundamental la descripción de las clases más importantes en el sistema y representa conceptos del mundo real, no de los componentes de software.
El modelo del dominio define un modelo de clases común para todos los implicados en el desarrollo, sirve como interlocutor entre clientes y desarrolladores. El propósito fundamental de este modelo es generar una terminología común y sentar las bases del entendimiento del desarrollo y no para definir el sistema completo.

2.1.2. ¿Por qué Modelo de Dominio?

Se determinó definir un modelo de dominio para el proyecto en curso con el objetivo de capturar los objetos más importantes que existen o los eventos que suceden en el entorno donde estará el sistema, además existe la necesidad de modelar el problema para su mejor comprensión.

2.1.3. Representación del Modelo del Dominio

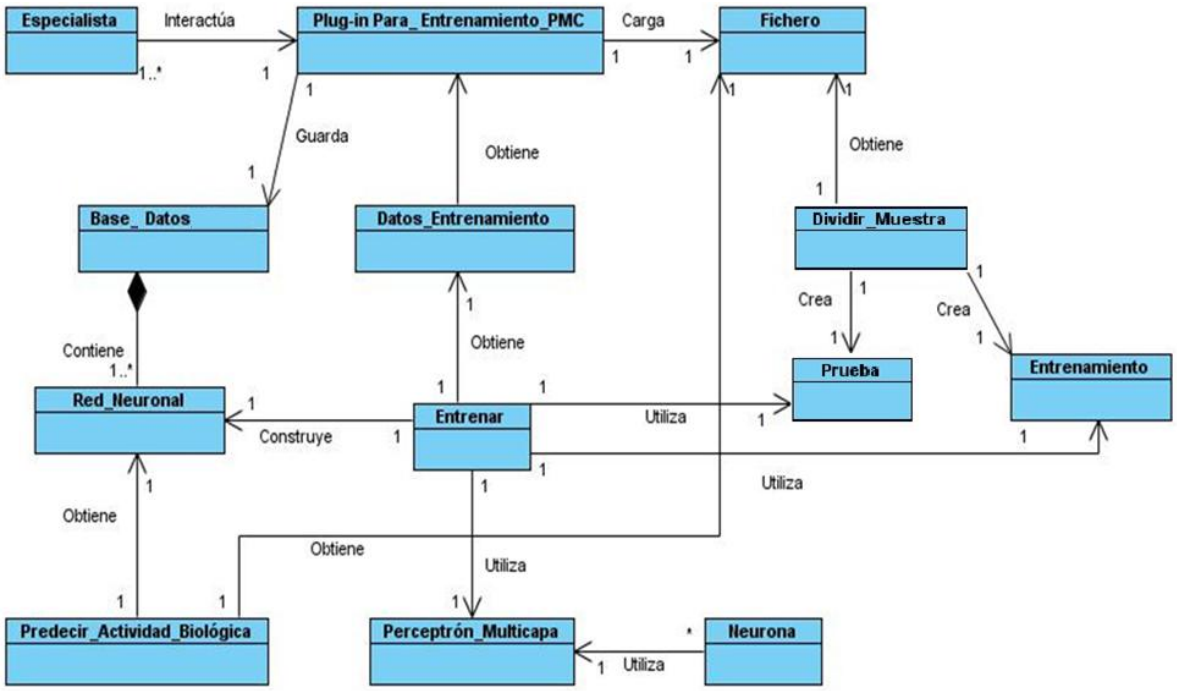


Figura 11. Modelo de Dominio.

2.2. Especificación de los Requisitos del Sistema

Los requisitos son condiciones o capacidades que necesita el usuario para resolver un problema o conseguir un objetivo determinado. Esta definición se extiende y se aplica a las condiciones que debe cumplir o poseer un sistema (o uno de sus componentes), para satisfacer un contrato, una norma o una especificación.

2.2.2. Requerimientos funcionales

Los requerimientos funcionales son capacidades o condiciones que el sistema debe cumplir. Permiten expresar una especificación más detallada de las responsabilidades del sistema en cuestión. Se mantienen invariables, sin importarle con qué propiedades o cualidades se relacionen.

RF1. Cargar Fichero.

RF2. Entrenar la red.

RF3. Gestionar Entrenamiento.

RF3.1 Guardar Entrenamiento.

RF3.2. Mostrar resultados de la red entrenada.

RF3.3. Eliminar red de la BD.

RF4. Cargar Red BD.

RF5. Predecir Actividad Biológica.

2.2.3. Requerimientos no funcionales

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable. Normalmente, están vinculados a requerimientos funcionales, es decir, una vez se conozca lo que el sistema debe hacer podemos determinar cómo ha de comportarse, qué cualidades debe tener o cuán rápido o grande debe ser:

- **Facilidad de uso:** el sistema debe ser fácil de usar de manera que tenga gran aceptación entre los usuarios.
- **Extensibilidad:** la nueva funcionalidad debe ser capaz de permitir la integración con otros módulos, además de permitir la inserción de cambios.
- **Apariencia o interfaz externa:** la interfaz que debe brindar la nueva funcionalidad debe ser sencilla, amigable y de rápida respuesta frente a una petición del usuario, de manera tal que agilice y facilite el trabajo con el software.
- **Mantenimiento y actualización:** debe dar facilidad de mantenimiento, y desarrollarse lo más sencilla y eficientemente posible para que en un futuro pueda ser atendido por grupos de trabajo no especializados.
- **Compatibilidad:** la nueva funcionalidad debe ser capaz de correr de manera independiente a la plataforma sobre la que es ejecutada. Lo cual estará asegurado por las características del lenguaje de programación utilizado, en este caso, el lenguaje de programación Java utilizado por los desarrolladores del polo garantiza la independencia de la plataforma de desarrollo.
- **Software:** para el uso del sistema es necesario tener instalado la Máquina Virtual de Java 1.5.

- **Hardware:** en el caso de las PC deberán contar con un microprocesador Intel Pentium 4 o superior. Se debe contar con 256 de RAM como mínimo. Espacio disponible en el disco duro de 20 MB como mínimo.
- **Legales:** se usarán herramientas de software libre y código abierto, o que funcionen bajo sistemas operativos representativos de código abierto.

2.3. Actores del Sistema

Se le llama Actor a toda entidad externa al sistema que guarda una relación con éste y que le demanda una funcionalidad. Esto incluye a los operadores humanos. Pero también incluye a todos los sistemas externos, así como a entidades abstractas como el tiempo.

Tabla 2. Identificación de los Actores del Sistema.

Autor	Descripción
Especialista	Representa al usuario que va a hacer uso del sistema, y quien tiene la posibilidad de interactuar con todas las funcionalidades de éste.

2.4. Diagrama de casos de uso del sistema

Un diagrama de casos de uso del sistema representa gráficamente las funcionalidades principales del sistema y su interacción con los actores. En el caso que se está tratando solo se identifica un actor, el Investigador, que es el usuario que interactúa con el sistema para realizar el proceso de importación en los formatos antes mencionados.

En la siguiente figura se muestra el Diagrama de Caso de Uso Sistema:

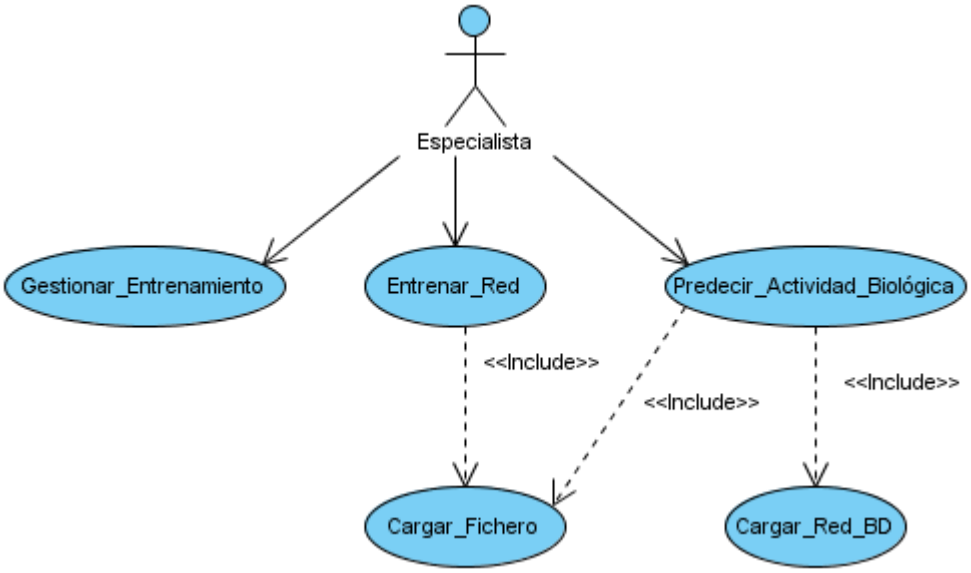


Figura 12. Diagrama de Casos de Uso del Sistema.

2.5. Descripción de los casos de uso de sistema

Para entender la funcionalidad asociada al caso de uso no es suficiente con la representación gráfica del diagrama de casos de uso, también se lleva a cabo la realización del mismo elaborando la descripción textual donde se especifican todas las acciones necesarias que realizan el actor y el sistema. Con la descripción del caso de uso del sistema, que a continuación se presenta se obtendrá claramente la idea de cómo se realizará el proceso a automatizar y quiénes intervienen directamente.

2.5.1 Descripción de los Casos de Uso del sistema

Caso de uso Cargar Fichero:

Tabla 3: Descripción del caso de uso Cargar Fichero.

Caso de Uso:	Cargar Fichero.
Actores:	Especialista.
Resumen:	El Caso de Uso se inicia cuando el especialista selecciona la opción Cargar Fichero.
Precondiciones:	Debe existir un archivo para el entrenamiento o la predicción en la ubicación seleccionada con uno de los siguientes formatos: .ARFF .DAT

Referencias	RF1.
Prioridad	Crítico
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1- El especialista selecciona la opción: • Cargar fichero	2- El sistema ejecuta la sección: • Cargar Fichero.
Sección "Cargar Fichero"	
Acción del Actor	**Respuesta del Sistema**
	1- El sistema muestra una caja de diálogo para seleccionar el archivo que desea cargar.
2- El especialista selecciona el archivo que desea cargar.	3- El sistema verifica que el fichero fue seleccionado.
	4- El sistema comprueba si el fichero es de entrenamiento o de predicción, mostrando un mensaje "El fichero es de Predicción" o "El fichero es de Entrenamiento"
	5- El sistema obtiene el fichero seleccionado.
Flujo alterno de la sección "Cargar Fichero"	
	3.1- Si no se seleccionó el fichero se muestra un mensaje "No se cargó ningún fichero".
Poscondiciones	Se cargó el fichero.

Caso de uso Entrenar la red:

Tabla 4: Descripción del caso de uso Entrenar la red.

Caso de Uso:	Entrenar la red
Actores:	Especialista
Resumen:	El Caso de Uso se inicia cuando el especialista selecciona la opción Entrenar.
Precondiciones:	Que exista un fichero cargado con la muestra para entrenar y que entre los datos necesarios en el formulario.
Referencias	RF2.
Prioridad	Crítico
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1- El especialista selecciona la siguiente opción:	2- El sistema ejecuta la sección:

• Entrenar.	• Entrenar.
Sección "Entrenar"	
Acción del Actor	**Respuesta del Sistema**
1- El especialista inserta los datos del entrenamiento.	2- El sistema verifica que los datos estén correctos, que no existan campos vacíos y que el fichero esté cargado.
	3- El sistema comienza el entrenamiento.
	4- El sistema muestra los resultados del entrenamiento.
Flujo alterno de la sección "Entrenar"	
	2.1- Si no insertó los datos correctamente se muestra un mensaje de error "Entre los datos correctamente".
	2.2- Si deja campos se muestra un mensaje de error "No debe tener campos vacíos".
	2.3- Si no se ha cargado el fichero se muestra un mensaje de error "No se ha cargado ningún fichero".
Poscondiciones	Se entrenó la red.

Caso uso Gestionar Entrenamiento:

Tabla 5: Descripción del caso de uso Gestionar Entrenamiento.

Caso de Uso:	Gestionar Entrenamiento
Actores:	Especialista
Resumen:	El Caso de Uso se inicia cuando el especialista selecciona la opción Guardar Entrenamiento, Mostrar Resultados del entrenamiento o Eliminar entrenamiento.
Precondiciones:	Que se haya realizado un entrenamiento.
Referencias	RF3.; RF3.1 RF3.2; RF3.3
Prioridad	Crítico
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1- El especialista selecciona una de las siguientes opciones: • Guardar entrenamiento. • Mostrar Resultados del entrenamiento.	2- El sistema ejecuta una de las siguientes secciones: • Guardar Entrenamiento. • Mostrar Resultados del entrenamiento. • Eliminar entrenamiento.

• Eliminar entrenamiento.	
Sección "Guardar Entrenamiento"	
Acción del Actor	**Respuesta del Sistema**
	1- El sistema muestra una caja de diálogo para insertar el nombre.
2- El especialista inserta el nombre.	3- El sistema verifica que se haya realizado un entrenamiento y que el campo nombre no esté vacío.
	4- El sistema guarda la red entrenada en le base de datos.
Flujo alterno de la sección "Guardar Entrenamiento"	
	3.1- Si no se ha realizado un entrenamiento para poder guardar se muestra un mensaje "No hay resultados para guardar".
	3.2- Si no inserta el nombre se muestra un mensaje "Debe proporcionar el nombre".
Sección "Mostrar resultados de la red."	
Acción del Actor	**Respuesta del Sistema**
1- El especialista selecciona la red entrenada de la que desea ver los resultados.	
	2- El sistema busca los resultados de la red seleccionada.
	3- El sistema muestra los resultados de la red.
Flujo alterno de la sección "Mostrar resultados de la red."	
	2.1- Si no selecciona la red entrenada muestra un mensaje "Debe seleccionar una fila".
Sección "Eliminar Entrenamiento."	
Acción del Actor	**Respuesta del Sistema**
1- El especialista selecciona la red a eliminar.	
	2- El sistema verifica que esté seleccionada la red a eliminar y elimina el entrenamiento.
Flujo alterno de la sección "Eliminar Entrenamiento."	
	2.1- Si no selecciona la red que desea eliminar, muestra un mensaje "Debe seleccionar una fila".
Poscondiciones	Se guardó el entrenamiento, se mostraron los resultados, ó se eliminó la red.

Caso uso Cargar Red BD:

Tabla 6: Descripción del caso de uso Cargar Red BD.

Caso de Uso:	Cargar Red.	
Actores:	Especialista	
Resumen:	El Caso de Uso se inicia cuando el especialista selecciona la opción Cargar Red BD.	
Precondiciones:	Que la BD tenga redes almacenadas.	
Referencias	RF4.	
Prioridad	Crítico	
Flujo Normal de Eventos		
Acción del Actor	**Respuesta del Sistema**	
1- El especialista selecciona la opción: • Cargar Red BD.	2- El sistema ejecuta la siguiente sección: • Cargar Red BD.	
Sección "Cargar Red BD"		
Acción del Actor	**Respuesta del Sistema**	
	1- El sistema muestra una caja de diálogo para Cargar Red BD.	
2- El especialista selecciona la red que quiere cargar.	3- El sistema verifica que se cargue correctamente.	
Flujo alterno de la sección "Cargar Fichero"		
	3.1- Si no carga correctamente la red se muestra un mensaje "La red no se pudo cargar".	
Poscondiciones	Se cargó la red de la base de datos.	

Caso uso Predecir Actividad biológica:

Tabla 7: Descripción del caso de uso Predecir Actividad biológica.

Caso de Uso:	Predecir Actividad biológica
Actores:	Especialista
Resumen:	El Caso de Uso se inicia cuando el especialista selecciona la opción Predecir.
Precondiciones:	Que se haya guardado un entrenamiento asociado al mismo ensayo.
Referencias	RF5.
Prioridad	Crítico

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1- El especialista selecciona la opción: • Predecir.	2- El sistema ejecuta la siguiente sección: • Predecir.
Sección "Predecir Actividad biológica"	
Acción del Actor	**Respuesta del Sistema**
1- El especialista selecciona la red para realizar la predicción de la actividad biológica.	
	2- El sistema comienza la predicción.
	3- El sistema muestra los resultados de la predicción.
Flujo alterno de la sección "Cargar Fichero"	
	3.1- Si no Realiza la predicción correctamente muestra un mensaje "No se ha realizado la predicción".
Poscondiciones	Se realizó la predicción.

2.6. Conclusiones parciales

En este capítulo se expuso la propuesta del sistema. Se trabajó en la fase de requerimientos, definiendo el modelo del dominio, así como los requerimientos funcionales y no funcionales que guiarán el proceso de desarrollo del sistema. Se elaboró el diagrama de casos de uso del sistema y se describieron los mismos.

CAPÍTULO 3: ANÁLISIS Y DISEÑO DEL SISTEMA

En este capítulo se describe la arquitectura del sistema, haciendo énfasis en los patrones de arquitectura utilizados para la realización del sistema. Además, se desarrolla el diseño del sistema, donde se elabora el diagrama de secuencia, que muestra gráficamente cómo los objetos se comunican entre ellos a fin de cumplir con los requerimientos, así como la representación del modelo de despliegue.

3.1. Representación Arquitectónica del sistema

La arquitectura del software desempeña un papel fundamental durante el desarrollo de un sistema ya que permite representar su estructura, además permite organizar sus componentes, las relaciones entre ellos, el ambiente y los principios que orientan su diseño y evolución. El presente trabajo responde a un proyecto cuya arquitectura está bien definida en el documento de arquitectura. Por ello, aquí sólo se abordará la interfaz visual del plug-in para el Perceptrón Multicapa.

A continuación se abordará el tema relacionado con los patrones utilizados para realizar el sistema. Un patrón es un par problema/solución con nombre que se puede aplicar en nuevos contextos, con consejos acerca de cómo aplicarlos en nuevas situaciones y discusiones sobre sus puntos fuertes y débiles. Existen varios tipos de patrones, ya conocidos, entre los que se encuentran los patrones de arquitectura, de diseño, de requisitos, de programación, entre otros. [21]

3.2. Patrón Arquitectónico Empleado

Los patrones arquitectónicos expresan un esquema organizativo estructural fundamental para sistemas software y definen las reglas generales de organización, las restricciones en la forma y la estructura de un grupo numeroso de sistemas de software. La selección de un patrón arquitectónico es una decisión fundamental de diseño en el desarrollo de un sistema de software. Para el desarrollo de las funcionalidades a implementar se utilizó el patrón arquitectónico Modelo Vista Controlador (MVC). El patrón MVC es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos:

- **El Modelo:** es la representación específica de la información con la cual el sistema opera. La lógica de datos asegura la integridad de estos y permite derivar nuevos datos.
- **La Vista:** presenta el modelo en un formato adecuado para interactuar con los usuarios, usualmente está conformada por las interfaces de usuario.

- **El Controlador:** responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista. [22]

Este patrón se basa principalmente en separar en tres capas el diseño de las aplicaciones, el modelo de datos, la presentación de los mismos y las acciones de los usuarios, utilizando la capa que gestiona las acciones (controlador) como administradora de los posibles eventos, ver la figura 13.

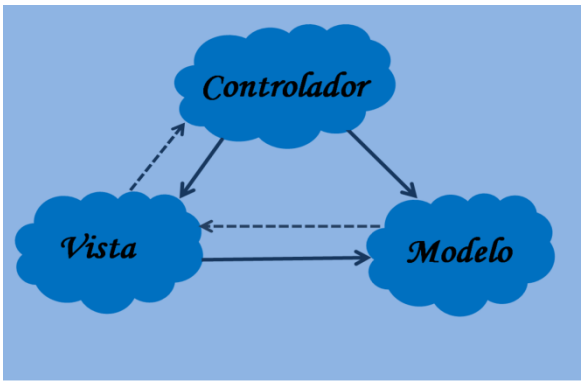


Figura 13. Representación del Patrón Modelo Vista Controlador.

Se decidió seleccionar este tipo de estilo arquitectónico para organizar los principales componentes del sistema a desarrollar y representar las relaciones que existen entre ellos, ya que el plug-in a desarrollar constituye una nueva funcionalidad que será añadida a la plataforma alasGRATO. A continuación se muestra cómo están distribuidos los componentes por las diferentes capas del patrón MVC. (Ver figura 14)

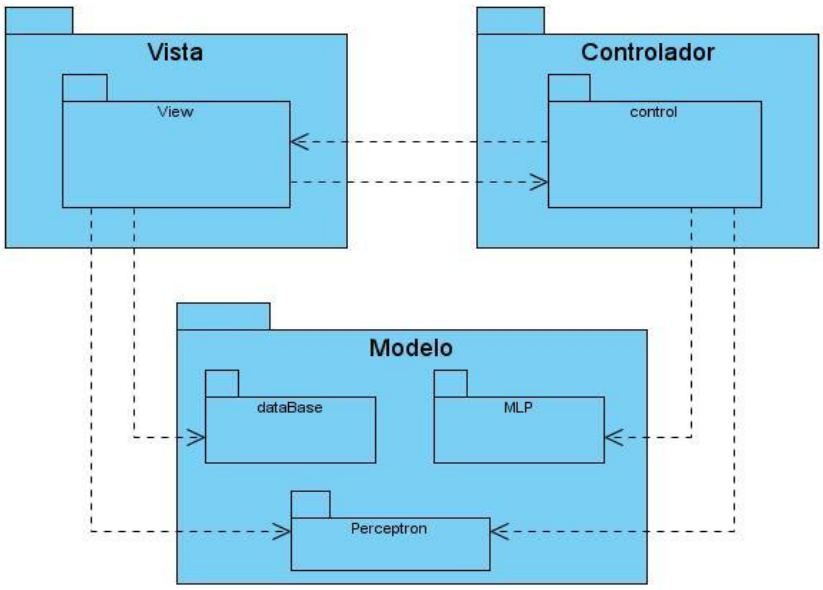


Figura 14. Vista arquitectónica.

3.2.1. Principales patrones de diseño utilizados

Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces.

Dentro de los patrones de producto de software se encuentran los de análisis, arquitectura, diseño y lenguaje de programación. Para el desarrollo de la solución se aplicaron diferentes patrones de diseño, fundamentalmente los patrones de diseño: General Responsibility Assignment Software Patterns (GRASP) y Gang of Four (GoF), con el objetivo de facilitar el mantenimiento del software. A continuación se abordarán los patrones de diseños utilizados en el sistema desarrollado para asegurar una solución mucho más confiable y contribuir a la realización de un producto reutilizable y escalable.

3.2.2. Patrones GRASP

Los patrones de asignación de responsabilidades GRASP, permiten asignar correctamente las responsabilidades a cada una de las clases que intervienen en el modelo; de este grupo de patrones fueron tomados en cuenta para una correcta asignación de las relaciones entre las clases y un correcto diseño de las mismas, los siguientes:

Creador: se refiere a asignar responsabilidades a las clases de crear instancias de otras conociendo que las primeras son las que contienen la información para ello.

En la figura 15 se puede ver el uso de este patrón en el diseño de la clase MLPControladora, pues esta clase es la encargada de crear las instancias de las diferentes clases a utilizar para poder realizar el entrenamiento de las redes así como la predicción de la actividad biológica.

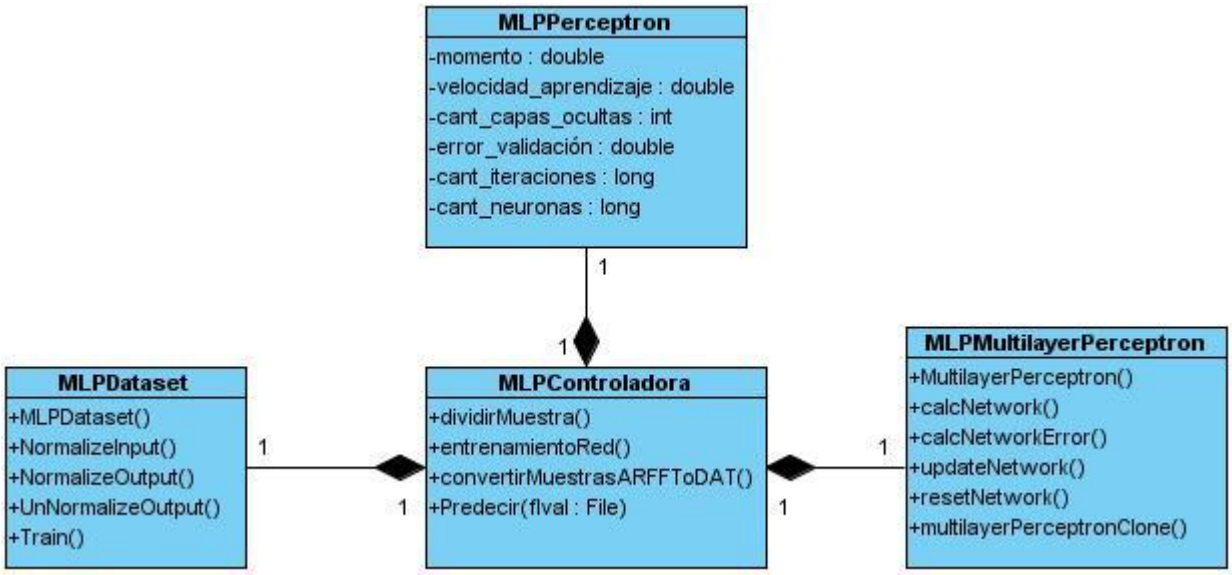


Figura 15. Diagrama donde se evidencia el uso del Patrón Creador.

Bajo acoplamiento: cada clase está relacionada a las clases estrictamente necesarias, garantizando un bajo impacto de los cambios que se producen en una clase para las demás clases que se relacionan con ella.

En la figura 15, se brinda un ejemplo de cómo se ve evidenciado dicho patrón, ya que las clases están relacionadas de manera que se establecen sólo las dependencias necesarias para cumplir con sus responsabilidades, esto favorece a la flexibilidad del diseño y a la actualización de los cambios del sistema pues las clases son menos dependientes entre sí.

Alta cohesión: asignar responsabilidades a las clases de manera que todos sus métodos tuvieran un comportamiento bien definido. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme.

Este patrón fue utilizado en el sistema desarrollado al implementar la clase controladora MLPMultilayerPerceptron, la cual se encarga de construir la red para realizar el entrenamiento, auxiliándose de la clase MLPNeuron.

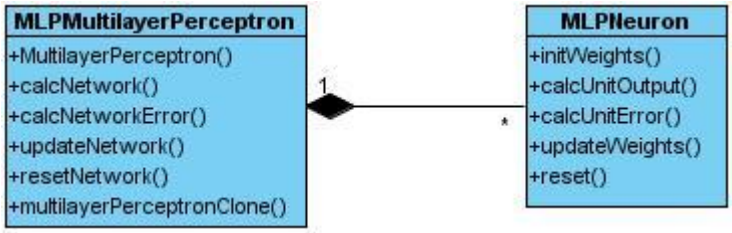


Figura 16. Diagrama donde se evidencia el uso del Patrón Alta Cohesión.

Patrón Controlador: la primera categoría de controlador es un controlador de fachada que representa al "sistema" global. Es una clase que, para el diseñador representa de alguna manera al sistema entero. En la figura 17 se puede ver el uso de este patrón en la clase MLPControladora, la cual se encarga de los eventos más significativos del sistema: entrenamiento de la red, la predicción de la actividad biológica, dividir la muestra para el entrenamiento y la prueba y convertir la muestra de ARFF a DAT.

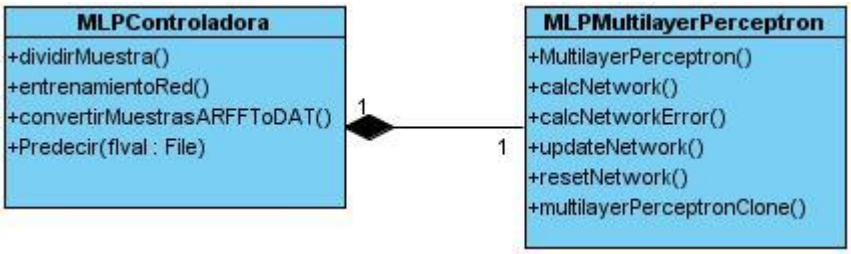


Figura 17. Diagrama donde se evidencia el uso del Patrón Controlador.

3.2.3. Patrones GoF

Los patrones de diseño GoF son conocidos como la pandilla de los cuatro, formada por Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides, los cuales clasificaron los patrones en 3 grandes categorías:

Creacionales: que abstraen el proceso de creación de instancias.

Estructurales: se ocupan de cómo clases y objetos son utilizados para componer estructuras de mayor tamaño.
Comportamiento: se centran en la comunicación entre las distintas clases y objetos.

Algunos de estos patrones fueron aplicados directamente en el diseño de este sistema, a continuación se menciona el nombre del patrón y las clases donde se aplicaron directamente.

- **Mediator:** es un patrón de diseño que coordina las relaciones entre sus asociados. Permite la interacción de varios objetos, sin generar acoples fuertes en esas relaciones.

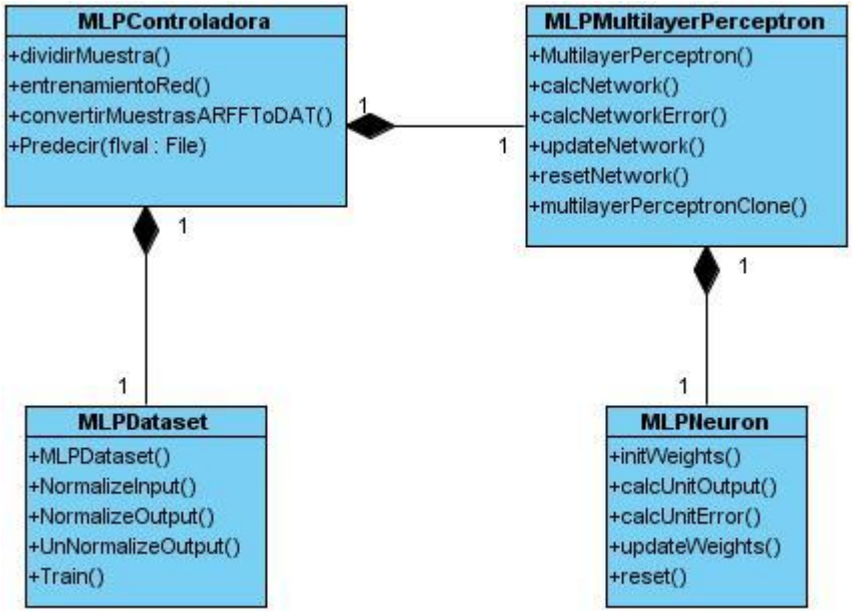


Figura 18. Diagrama donde se evidencia el uso del Patrón Mediator.

- **Template Method:** es un patrón de diseño que define una estructura algorítmica en la súper clase, delegando la implementación a las subclases. Es decir, define una serie de pasos, donde estos pasos serán redefinidos en las subclases.

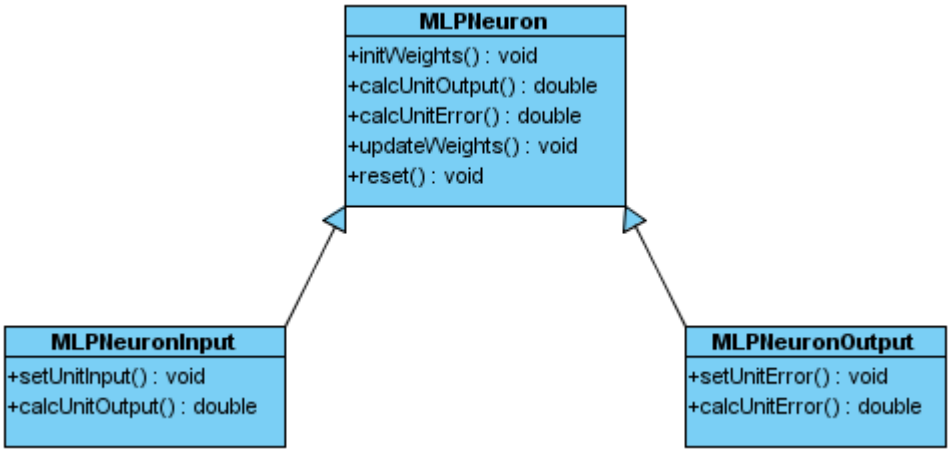


Figura 19. Diagrama donde se evidencia el uso del Patrón Template.

3.3. Modelo Del Diseño

Es una abstracción del Modelo de Implementación y su código fuente, el cual se emplea fundamentalmente para representar y documentar su diseño. Es usado como entrada esencial en las actividades relacionadas con la implementación. Representa a los casos de uso en el dominio de la solución.

A continuación se abordarán los temas relacionados con el diseño del plug-in para el Perceptrón Multicapa.

3.3.1. Diagramas de Clases

Un diagrama de clases es un tipo de diagrama estático que describe la estructura de un sistema; también muestra sus 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, conjuntamente con los componentes que se encargarán del funcionamiento y las relaciones entre unos y otros.

En la siguiente figura se muestra el diagrama de clases del diseño del sistema para el caso de uso “Cargar Fichero”.

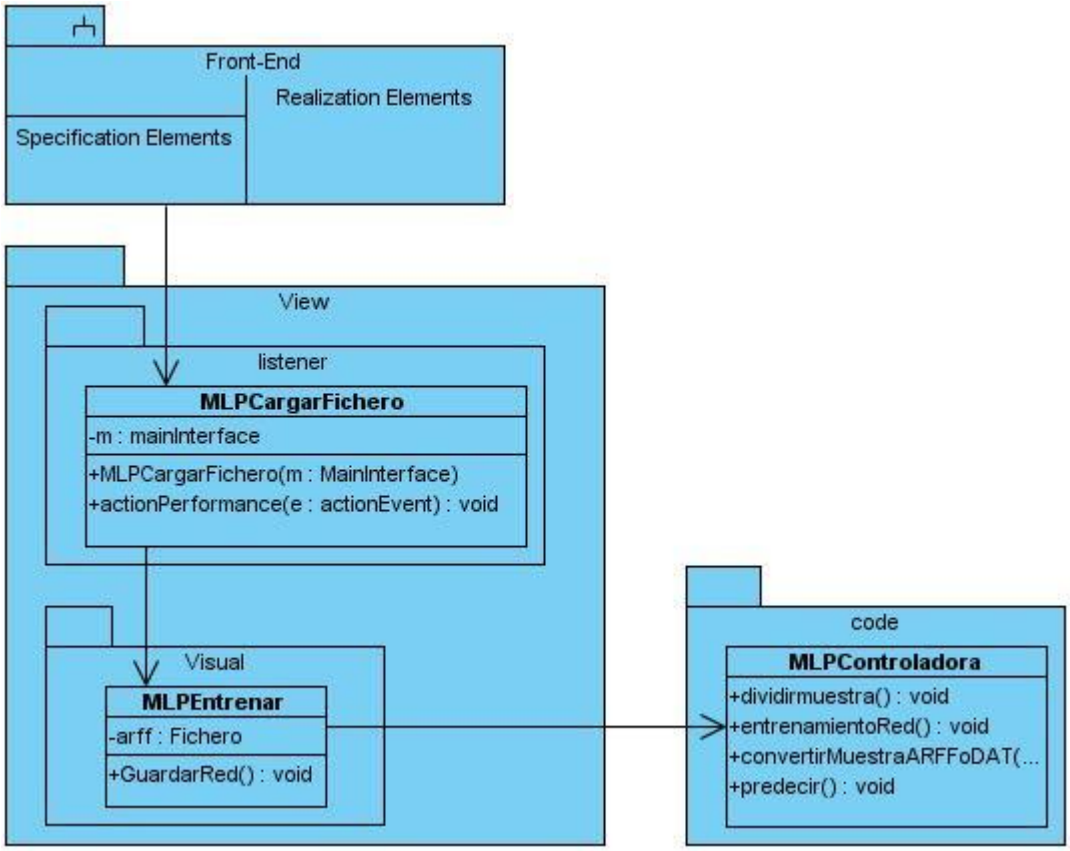


Figura 20. Diagrama de clases del diseño para el CU Cargar Fichero.

3.3.2. Diagramas de Interacción

Los diagramas de interacción se utilizan para modelar los aspectos dinámicos de un sistema, lo que conlleva modelar instancias concretas o prototípicas de clases interfaces, componentes y nodos, junto con los mensajes enviados entre ellos. Todo en el contexto de un escenario que ilustra un comportamiento. En el contexto de las clases se describe la forma en que grupos de objetos colaboran para proveer un comportamiento. Mientras que un diagrama de casos de uso presenta una visión externa del sistema, las funcionalidades de dichos casos de uso se recogen como un flujo de eventos, y se utilizan para ello, interacciones entre sociedades de objetos.

A continuación se muestra el diagrama de secuencia para el caso de uso “Cargar Fichero”.

3.3.3. Diagrama de Secuencia del Caso de Uso Cargar Fichero

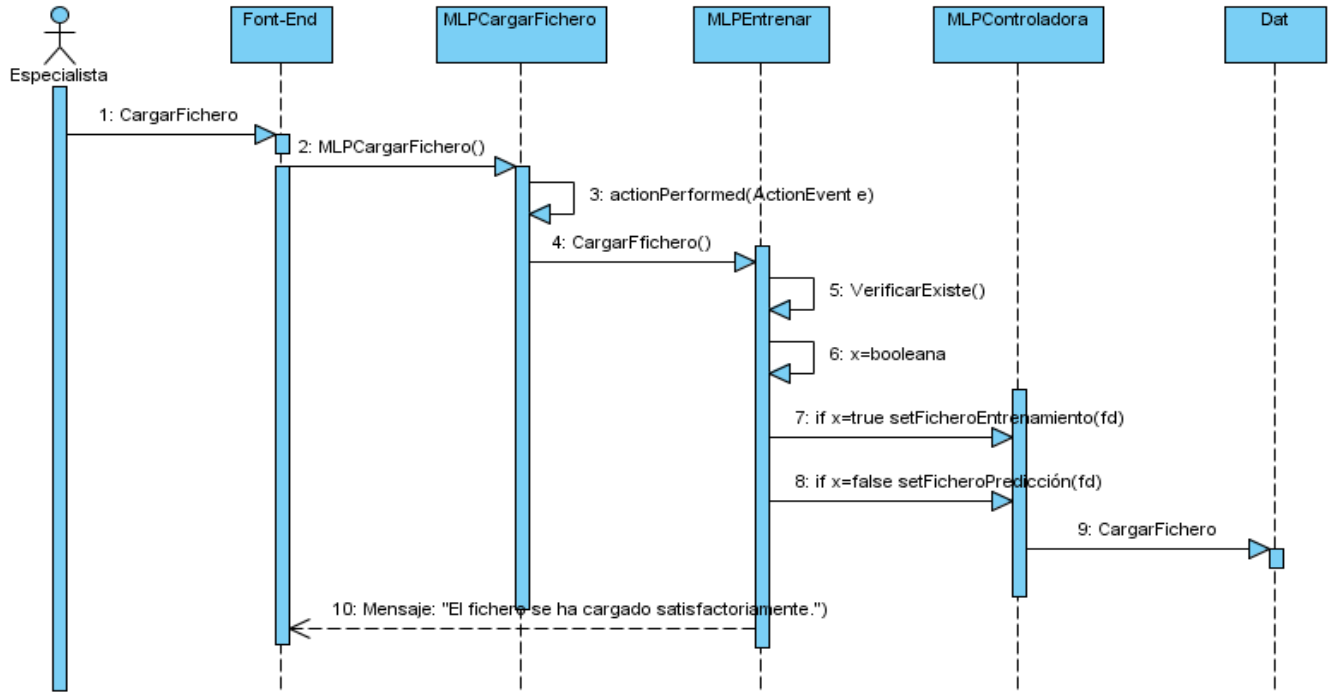


Figura 21. Diagrama de secuencia para el CU Cargar Fichero.

3.4. Modelo de Despliegue

El modelo de despliegue muestra la configuración de los nodos de procesamiento en tiempo de ejecución, los links de comunicación entre ellos, y las instancias de los componentes y objetos que residen en ellos. En la figura 22 se muestra el diagrama de despliegue que presenta el plug-in para el Perceptrón Multicapa.

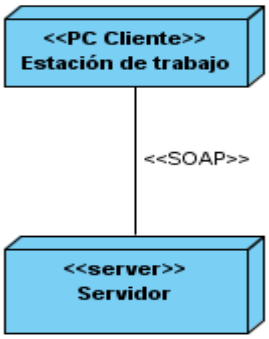


Figura 22. Diagrama de despliegue.

3.5. Diseño de la Base de Datos

Una de las actividades fundamentales para el desarrollo del plug-in es el diseño de la Base de datos, que permita guardar las redes entrenadas para su posterior utilización en nuevas predicciones. Esta base de datos permitirá almacenar datos del entrenamiento, así se podrá tener acceso y recuperar de forma eficiente la información con redundancia mínima. Para lograrlo se requiere desarrollar un conjunto de pasos bien definidos que guían el diseño de la base de datos, a partir de los cuales se obtienen dos importantes artefactos: Diagrama de clases persistentes y Modelo de datos.

3.5.1. Diagrama de clases persistentes

Todas las clases identificadas en el diseño no son persistentes. La persistencia es la capacidad de un objeto de mantener su valor en el espacio y en el tiempo, es responsabilidad del diseñador identificar estas clases. El diagrama de clases persistentes captura gráficamente las clases persistentes identificadas:

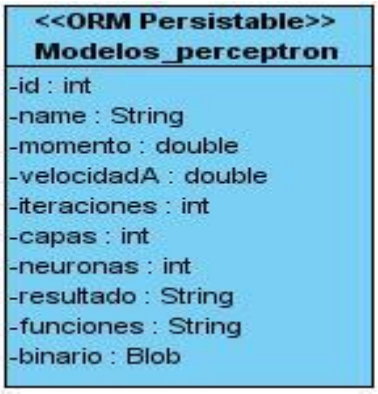


Figura 23. Diagrama de clases persistente

Descripción de la entidad Modelos_perceptron:

Tabla 8: Descripción de la entidad Modelos_perceptron

Nombre: modelos_perceptron	
Tipo de clase: entidad	
Atributo:	**Tipo:**
name	text
momento	double
VelocidadA	double
iteraciones	int
capas	int
neuronas	int
resultado	text
funciones	text
binario	longblob

3.5.2. Modelo de datos

En el Modelo de datos se describen las tablas que representan las distintas entidades que pertenecen al dominio del problema y serán almacenadas en la base de datos, en nuestro caso tenemos solamente una tabla. La siguiente figura muestra el Diagrama entidad-relación:

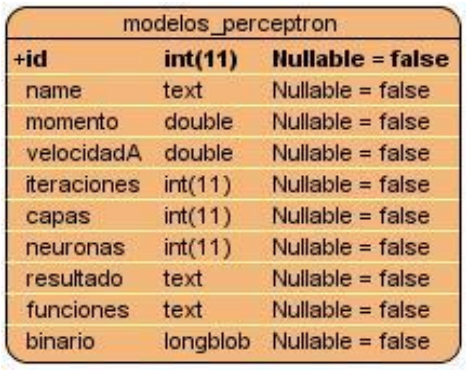


Figura 24. Diagrama entidad relación

Descripción de la tabla modelos_perceptron

La tabla modelos_perceptron almacena los datos generales de las redes entrenadas. Como el identificador de la red, que se encarga de identificar cada red almacenada, además se almacenan los datos generales de la red entrenada. Cada entrenamiento puede ser realizado de diferentes maneras: con una o dos capas ocultas y la selección de las funciones de transferencias escogidas por el usuario.

3.6. Conclusiones parciales

En este capítulo se vio la necesidad de identificar el estilo arquitectónico a utilizar para el desarrollo de las funcionalidades así como los patrones de diseño utilizados y la fundamentación del porqué y cómo se usan. También se realizó el diagrama de clases del diseño y el diagrama de secuencia para el caso de uso “Cargar Fichero”.

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. Se mostrará el código fuente de los principales métodos de las clases; se desarrollarán, además, varias pruebas para conocer la calidad del producto, utilizando los métodos de caja negra en cada caso para asegurar que los requisitos funcionales fueron cumplidos y asegurar el correcto funcionamiento de la aplicación.

4.1. Diagrama de componentes

Un diagrama de componentes muestra las organizaciones y dependencias lógicas entre componentes de software. Los componentes pueden ser de código fuente, binarios o ejecutables, archivos y bibliotecas cargadas dinámicamente. Desde el punto de vista del diagrama de componentes se tienen en consideración los requisitos relacionados con la facilidad de desarrollo, la gestión del software, la reutilización y las restricciones impuestas por los lenguajes de programación y las herramientas utilizadas en el desarrollo.

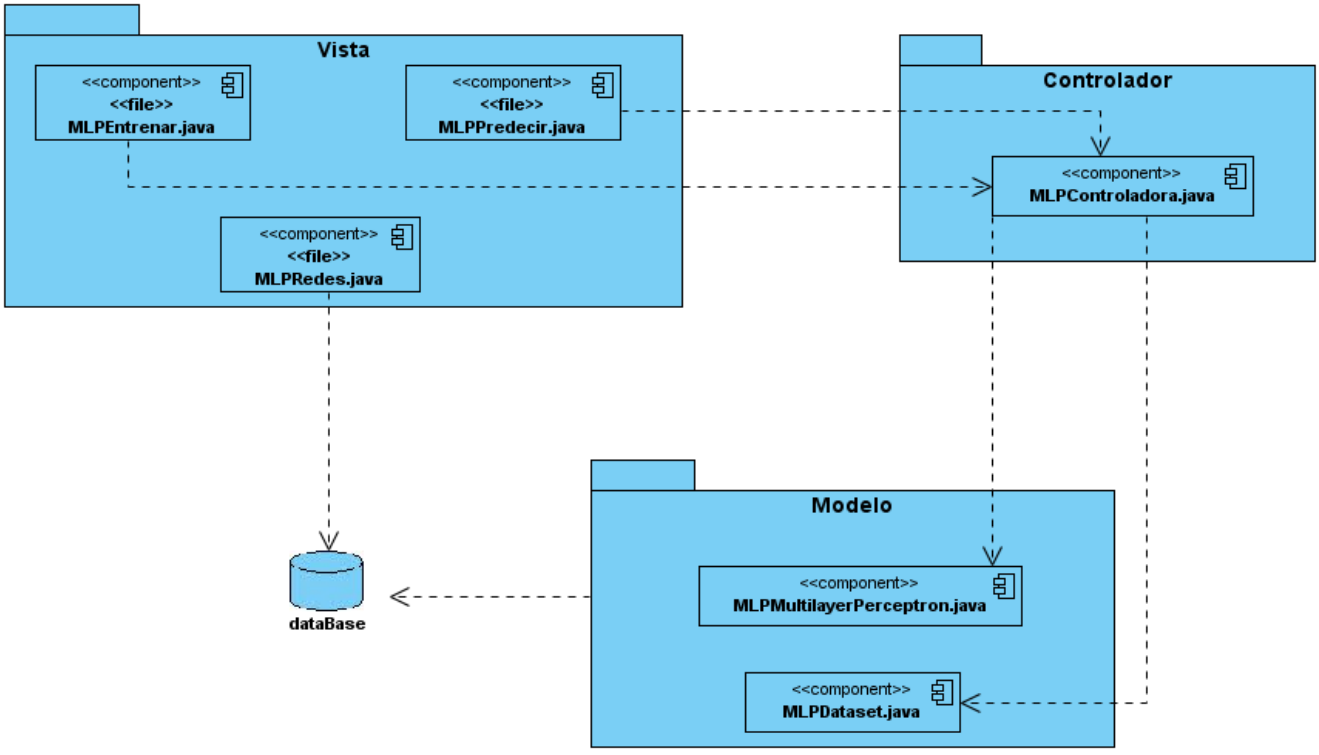


Figura 25. Diagrama de componentes.

4.2. Pasos para Implementar el Plug-in para el Perceptrón Multicapa

Para la implementación del plug-in en la plataforma alasGRATO, se realizaron diferentes pasos entre ellos podemos mencionar los siguientes:

1 - Conformar un XML con el mismo nombre del plug-in que cumpla con la arquitectura del Front-End, donde se reflejan datos como: nombre del plug-in y acciones que realiza el mismo, la dirección de las clases de las acciones a desarrollar, además de las extensiones con las que trabajará el plug-in.

Representación del XML del plug-in para el Perceptrón Multicapa:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v5 U (http://www.xmlspy.com) by neo (uci) -->
<plugin xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="D:\Personal\Tesis\workspace00\Front-End\plugins.xsd"
name="Plug-in for MLP" accion="" exts="dbs" description="" category="">
  <band name="Fichero" type="JRibbonBand" ext="all">
   <icon height="32" width="32" iconsource="svg/open.svg"/>
   <button name="Cargar Fichero" action="view.listener.MLPCargarFichero" classType="0"
mnemonic="O" popupClassType="" tooltipText="Cargar Fichero" priority="TOP"
mostImportant="True">
     <icon height="32" width="32" iconsource="svg/open.svg"/>
   </button>
  </band>
  <band name="Entrenar" type="JRibbonBand" ext="all">
   <icon height="32" width="32" iconsource="svg/open.svg"/>
   <button name="Entrenar" action="view.listener.MLP_Entrenar" classType="0" mnemonic="O"
popupClassType="" tooltipText="Entrenar" priority="TOP" mostImportant="True">
     <icon height="32" width="32" iconsource="svg/Entrenar.svg"/>
</button>
   <button name="Guardar Red" action="view.listener.MLPGuardarBD" classType="0" mnemonic="O"
popupClassType="" tooltipText="Entrenar" priority="TOP" mostImportant="True">
  <icon height="32" width="32" iconsource="svg/GuardarRed.svg"/>
   </button>
```

```
    </band>
    <band name="Predecir" type="JRibbonBand" ext="all">
      <icon height="32" width="32" iconsource="svg/open.svg"/>
  <button name="Cargar Red Entrenada" action="view.listener.MLP_CargarRedBD" classType="0" mnemonic="O" popupClassType="" tooltipText="Cargar Red" priority="TOP" mostImportant="True">
      <icon height="32" width="32" iconsource="svg/CargarRed.svg"/>
    </button>
    <button name="Predecir" action="view.listener.ViewRedes" classType="0" mnemonic="O" popupClassType="" tooltipText="Predecir" priority="TOP" mostImportant="True">
      <icon height="32" width="32" iconsource="svg/Predecir.svg"/>
    </button>
    <button name="Guardar Fichero" action="view.listener.MLPGuardarPrediccion" classType="0" mnemonic="O" popupClassType="" tooltipText="Guardar Fichero" priority="TOP" mostImportant="True">
      <icon height="32" width="32" iconsource="svg/GuardarFichero.svg"/>
  </button>
    </band>
    <band name="Redes" type="JRibbonBand" ext="all">
      <icon height="32" width="32" iconsource="svg/open.svg"/>
      <button name="Redes" action="view.listener.ViewRedes" classType="0" mnemonic="O" popupClassType="" tooltipText="Redes" priority="TOP" mostImportant="True">
      <icon height="32" width="32" iconsource="svg/TablaRedes.svg"/>
      </button>
    </band>
  </plugin>
```

2.- Implementar el plug-in cumpliendo estrictamente con la estructura de paquete definido en la arquitectura, donde las acciones estarán separadas de las vistas, debido a que se utiliza programación orientada a evento.

En la figura 26, se observa la estructura de paquete que muestra cómo están agrupadas las clases del plug-in.

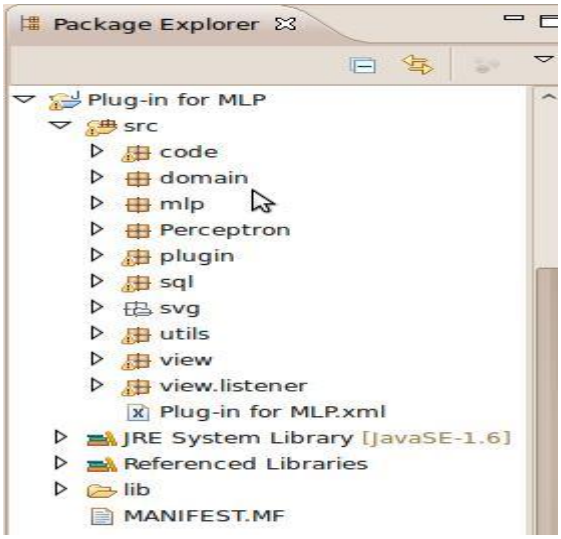


Figura 26. Estructura de paquete para el plug-in

3.- En el fichero Manifest del plug-in se especificará la dirección de la clase principal del plug-in en el campo PluginBaseClass y las librerías utilizadas.

Manifest-Version: 2.0

Name: Grato

Created-By: Karelia Marcelina Del Pino Medina, Surama Columbié Tomás

PluginName: Plug-in for MLP

Class-Path: lib/mlp.jar lib/mysql-connector-java-5.0.4-bin.jar lib/sqlitejdbc-v056.jar lib/substance.jar

Description: Molecular Viewer for alasGRATO Platform

PluginBaseClass: plugin.PluginMainMLP

4.3. Pruebas del Sistema

La prueba de funcionalidad de software es el proceso para identificar los posibles fallos de implementación, calidad, o facilidad de uso de un software. Las pruebas son ejecutadas bajo unas condiciones o requerimientos específicos, los resultados son observados y registrados, y se hace una evaluación y rectificación de algunos aspectos de sistema o componente.

Las pruebas se realizan teniendo en cuenta que todo sistema debe ser revisado con el objetivo de establecer el nivel de calidad requerido. A medida que los sistemas sean más complejos y aumente la demanda de calidad, se hacen necesarios procesos y métodos que la garanticen. La etapa de prueba es importante realizarla ya que en ella se refleja la calidad con que ha sido llevada a cabo la proyección del sistema. Aun así las pruebas no confirman la ausencia de errores en el presente software, solo brindan una medida de cómo responderá el mismo ante algunas situaciones determinadas.

El tipo de prueba realizado corresponde a la de Prueba de Unidad con el objetivo de comprobar que esta cumpla correctamente los requisitos funcionales establecidos. Se utilizó el método de caja negra, con lo cual se verificó no sólo que las funciones del software son operativas, sino también que no existieran errores en las interfaces.

4.3.1. Casos de prueba

Con el objetivo de comprobar el correcto funcionamiento del sistema se realizaron diferentes casos de prueba.

Tabla 9: Caso de Prueba Importar fichero.

Caso de uso	Cargar Fichero.
Caso de prueba	Cargar fichero.
Entrada	Inicialmente el especialista selecciona la opción de Cargar fichero, luego selecciona el archivo teniendo en cuenta su ubicación, éste contendrá la muestra para realizar el entrenamiento de la red y finalmente selecciona la opción aceptar.
Resultado esperado	Se muestra un mensaje indicando que el fichero ha sido cargado satisfactoriamente. (Ver figura 27).
Resultado de la prueba	Después de efectuar la opción Cargar fichero, se muestra un mensaje de indicando que el fichero ha sido cargado satisfactoriamente, por lo que el resultado de la prueba fue el esperado.
Condiciones	El archivo a importar debe tener una de las siguientes extensiones: ARFF ó DAT. El especialista podrá realizar la búsqueda en todos los directorios de la computadora, pero solo verá los ficheros que sean de la extensión mencionada anteriormente.

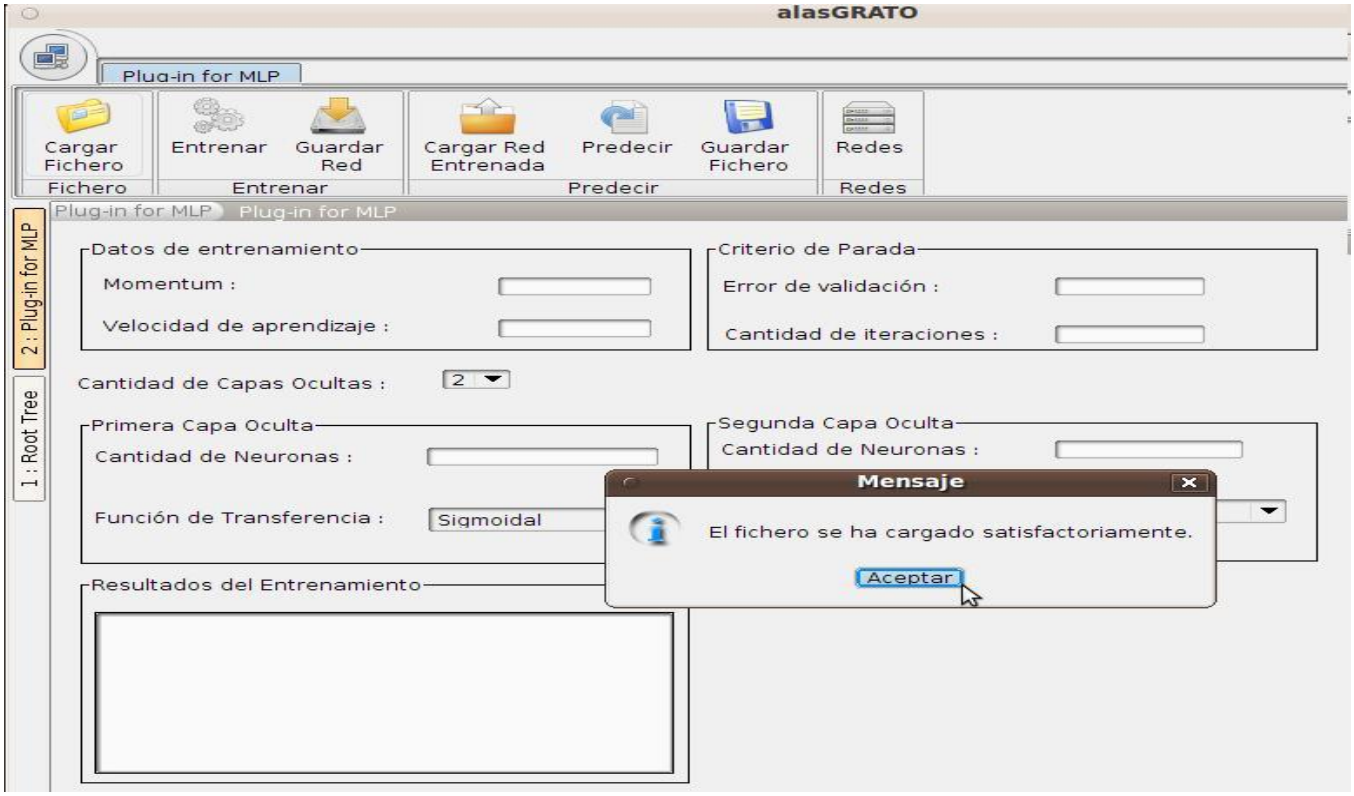


Figura 27. Resultado del caso de prueba "Cargar Fichero"

Tabla 10: Caso de Prueba Entrenar la red.

Caso de uso	Entrenar red.
Caso de prueba	Entrenar la red.
Entrada	Inicialmente el especialista proporciona los datos del entrenamiento, carga el fichero y luego selecciona la opción Entrenar.
Resultado esperado	Se muestran los resultados del entrenamiento. (Ver figura 28)
Resultado de la prueba	Después de efectuar la opción entrenar, se muestran los resultados del entrenamiento, por lo que el resultado de la prueba fue el esperado.
Condiciones	El especialista debe proporcionar todos los datos correctamente, no deben dejar campos vacíos y debe haber cargado el fichero.

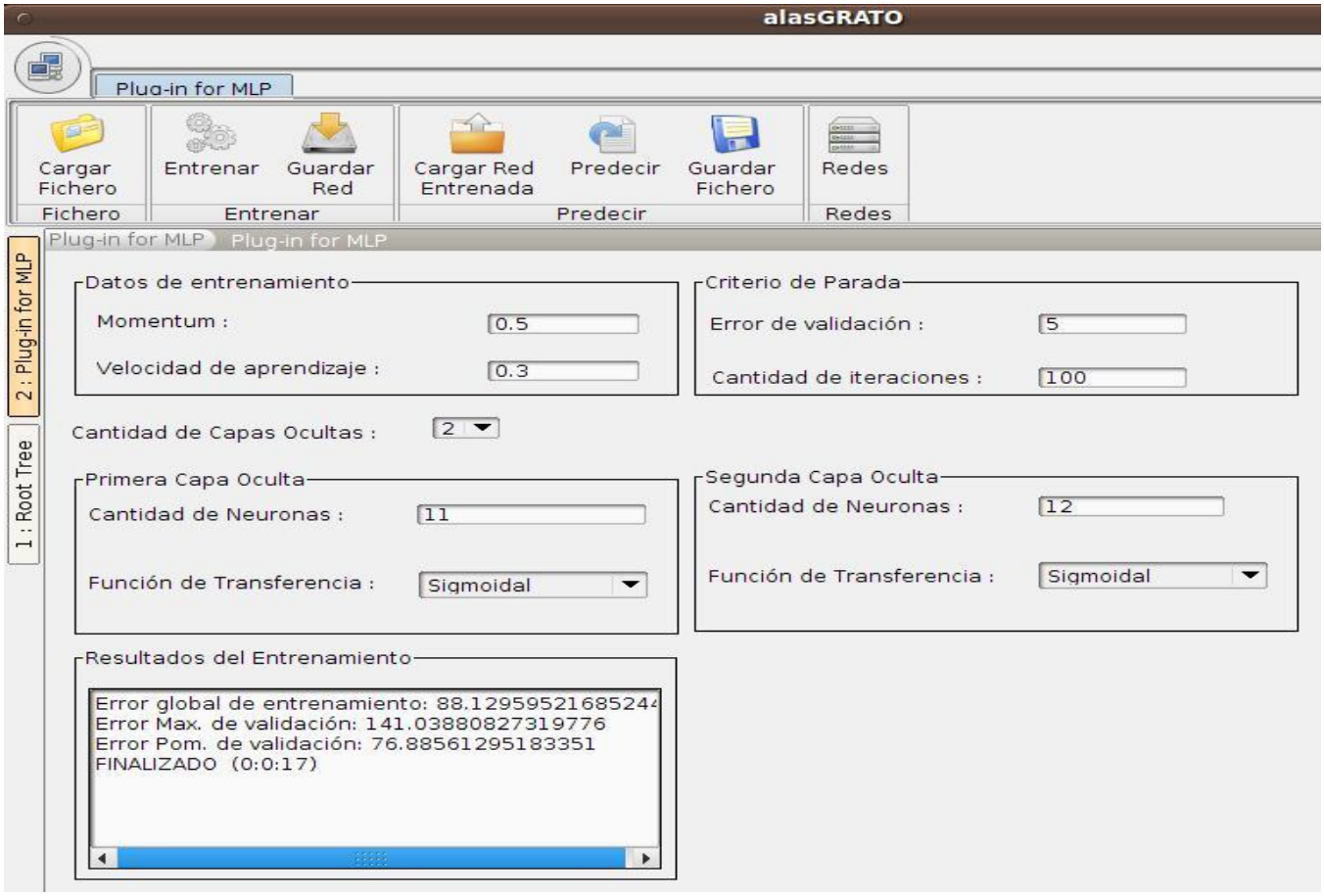


Figura 28. Resultado del caso de prueba "Entrenar la red"

Tabla 11: Caso de Prueba Entrenar la red con momentum o velocidad de aprendizaje, incorrecto.

Caso de uso	Entrenar la red.
Caso de prueba	Entrenar la red con momentum o velocidad de aprendizaje, incorrecto.
Entrada	Inicialmente el especialista proporciona los datos del entrenamiento con el valor del momentum o velocidad de aprendizaje menor que cero o mayor que 1, carga el fichero y luego selecciona la opción de Entrenar.
Resultado esperado	Se muestra un mensaje de error indicando que el valor del momentum o velocidad de aprendizaje debe estar entre 0 y 1. (Ver figura 29 y 30)

Resultado de la prueba	Después de efectuar la opción entrenar, se muestra un mensaje de error indicando que los valores están incorrectos, por lo que el resultado de la prueba fue el esperado.
Condiciones	Los valores del momentum o velocidad de aprendizaje no deben estar entre 0 y 1.

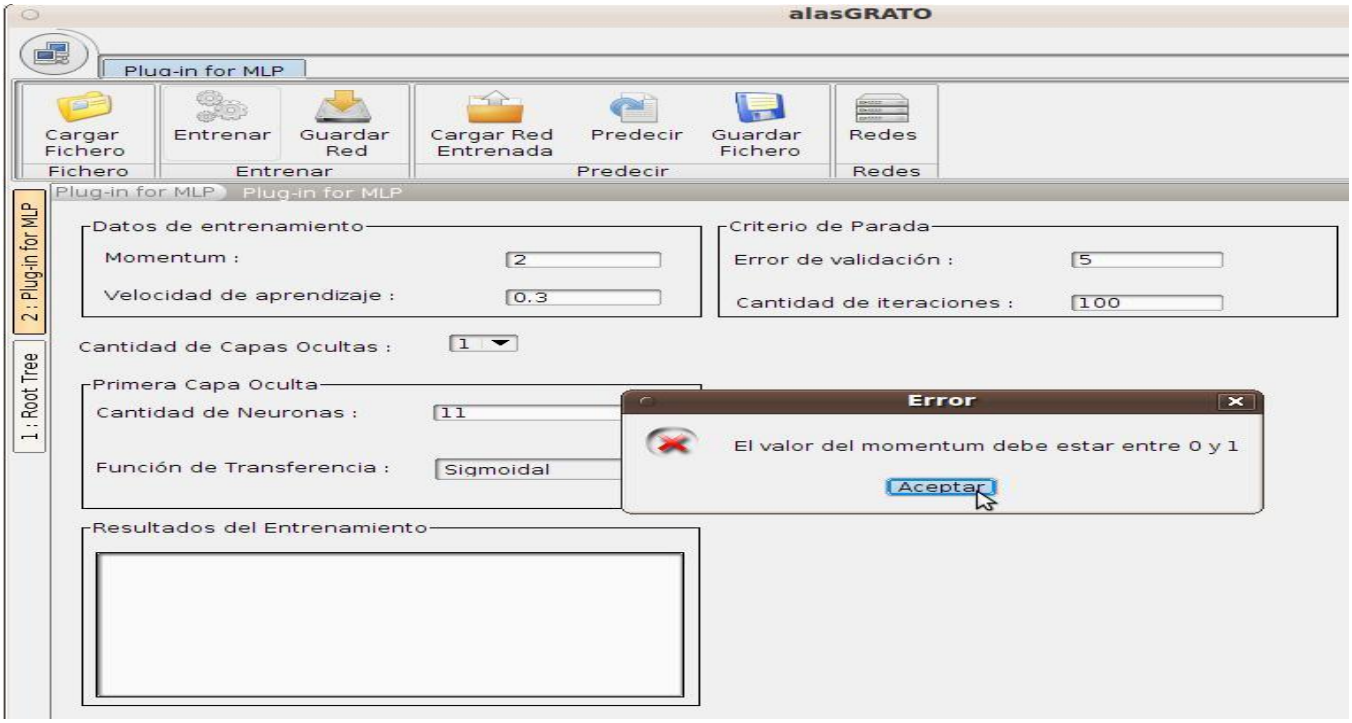


Figura 29. Resultado del caso de prueba "Entrenar Red con el parámetro momentum incorrecto"

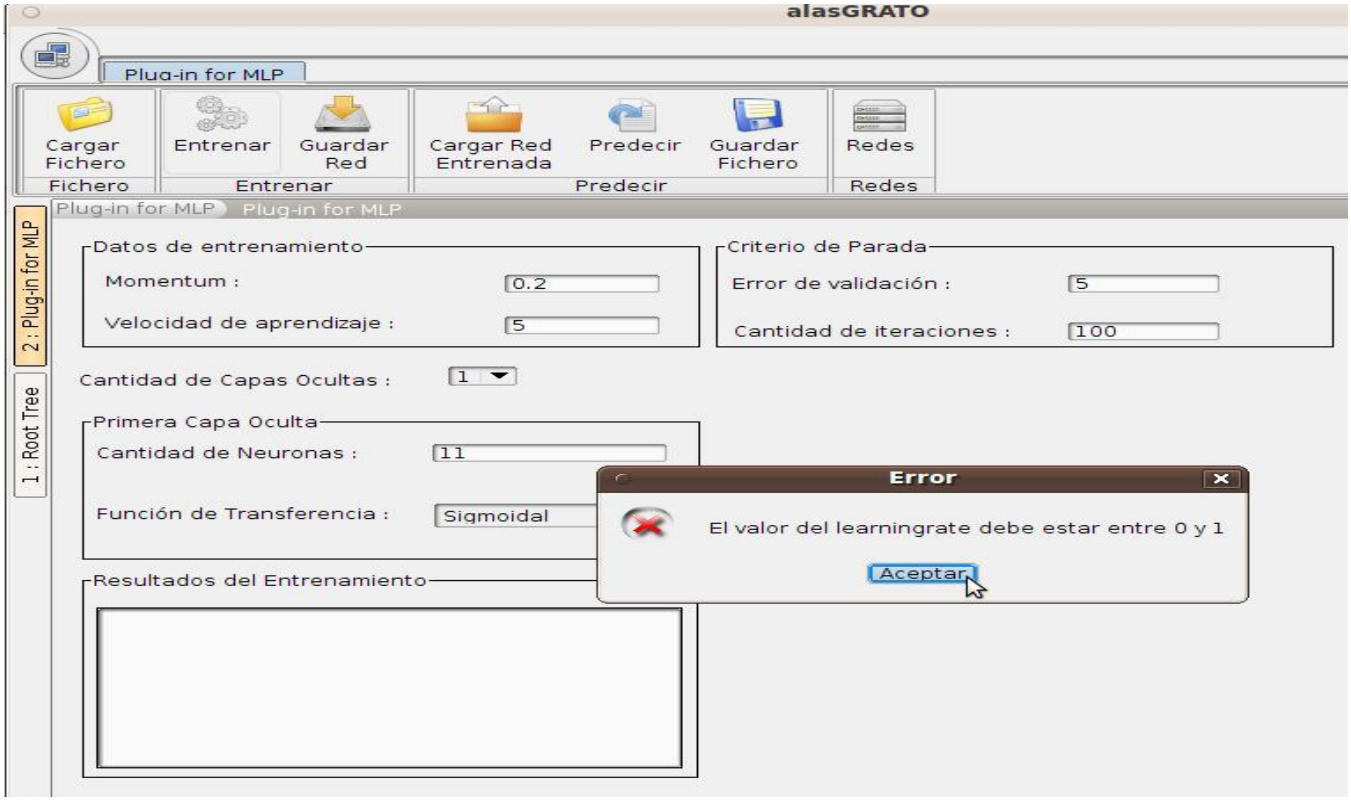


Figura 30. Resultado del caso de prueba "Entrenar Red con el parámetro velocidad de aprendizaje incorrecto"

Tabla 12: Caso de Prueba Entrenar la red con campos vacíos o datos incorrectos.

Caso de uso	Entrenar la red.
Caso de prueba	Entrenar la red con campos vacíos.
Entrada	Inicialmente el especialista deja campos vacíos, carga el fichero y luego selecciona la opción de Entrenar.
Resultado esperado	Se muestra un mensaje de error indicando que no deben existir campos vacíos. (Ver figura 31).
Resultado de la prueba	Después de efectuar la opción entrenar, se muestra un mensaje de error indicando que no deben existir campos vacíos, por lo que el resultado de la prueba fue el esperado.
Condiciones	Existen campos vacíos.

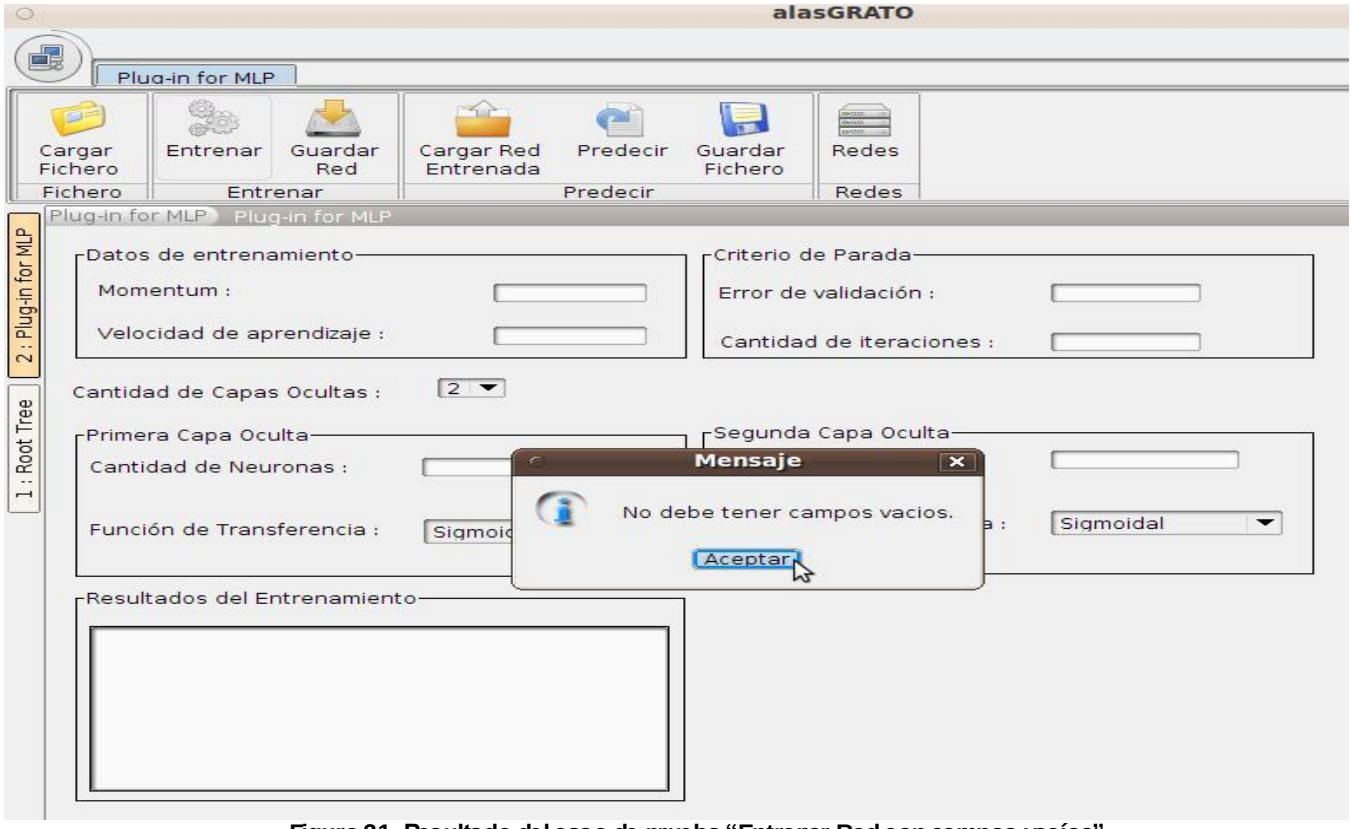


Figura 31. Resultado del caso de prueba "Entrenar Red con campos vacíos"

Tabla 13: Caso de Prueba Guardar Entrenamiento.

Caso de uso	Guardar Entrenamiento.
Caso de prueba	Guardar Entrenamiento.
Entrada	Inicialmente el especialista selecciona la opción de Guardar Entrenamiento, luego aparece una ventana de diálogo para que proporcione el nombre con el que desea guardar la red.
Resultado esperado	Aparece una ventana de diálogo para que proporcione el nombre con el que desea guardar la red. (Ver figura 32).
Resultado de la prueba	Después de efectuar la opción guardar entrenamiento aparece una ventana de diálogo para que proporcione el nombre con el que desea guardar la red, por lo que el resultado de la prueba fue el esperado.
Condiciones	Que se haya realizado un entrenamiento.

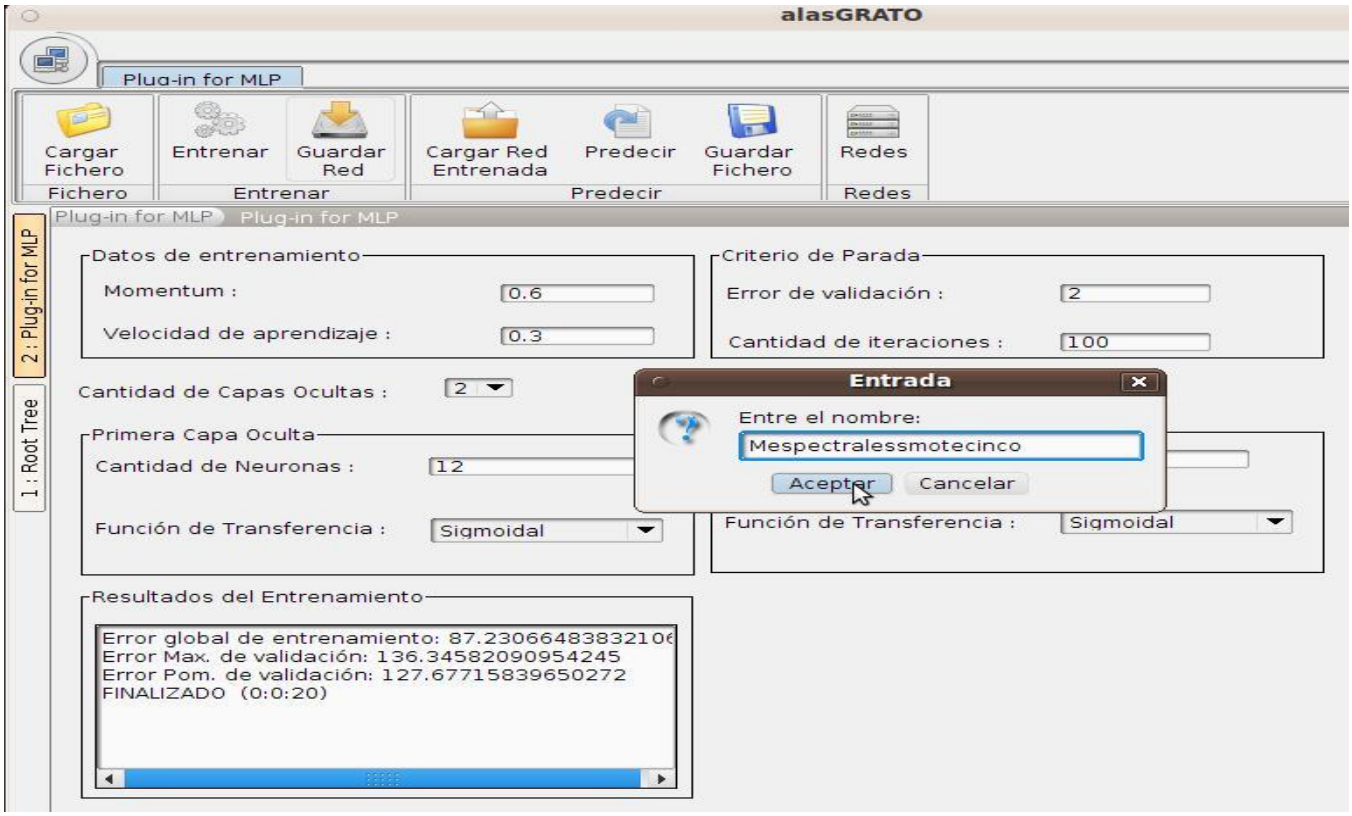


Figura 32. Resultado del caso de prueba "Guardar Entrenamiento"

Tabla 14: Caso de Prueba Cargar Red BD.

Caso de uso	Cargar Red BD.
Caso de prueba	Cargar Red BD.
Entrada	Inicialmente el especialista selecciona la red con la que desea trabajar y luego selecciona la opción de Cargar Red BD.
Resultado esperado	Se muestra un mensaje indicando que la red ha sido cargada satisfactoriamente.
Resultado de la prueba	Después de efectuar la opción Cargar Red BD se muestra un mensaje indicando que la red ha sido cargada satisfactoriamente, por lo que el resultado de la prueba fue el esperado.
Condiciones	El especialista debe seleccionar la red que desea cargar.

Tabla 15: Caso de Prueba Predecir Actividad biológica.

Caso de uso	Predecir Actividad biológica.
Caso de prueba	Predecir Actividad biológica.
Entrada	Inicialmente el especialista carga la red de la base de datos, carga el fichero de la predicción y luego selecciona la opción de Predecir.
Resultado esperado	Se muestra el resultado de la predicción.
Resultado de la prueba	Después de efectuar la opción Predecir se muestra el resultado de la predicción, por lo que el resultado de la prueba fue el esperado.
Condiciones	El especialista debe cargar la red de la base de datos y cargar el fichero de predicción.

4.4. Conclusiones parciales

En este capítulo se realizó el diagrama de componentes con el objetivo de brindar una idea de cómo se implementó el software en término de componentes. Además, se trató la validación del software mediante los casos de prueba de caja negra para el correcto funcionamiento de la aplicación, comprobándose que no existen errores en las funciones operativas del software.

CONCLUSIONES

- Se desarrolló una aplicación visual para realizar el entrenamiento del Perceptrón Multicapa y la predicción de la actividad biológica, dinámica y multiplataforma, utilizando el Front-End.
- Se creó una base de datos que permite guardar los entrenamientos realizados, para su posterior utilización en la predicción de la actividad biológica de nuevos compuestos orgánicos asociados a un mismo ensayo.
- Se desarrolló el diseño de la aplicación definiendo los principales patrones de diseño a utilizar para obtener un mejor resultado en la aplicación, además se realizaron pruebas a la misma para comprobar que cumple con los requisitos funcionales.

RECOMENDACIONES

Al concluir el presente trabajo el equipo de desarrollo recomienda:

- Agregar al plug-in otros tipos de redes neuronales para realizar el entrenamiento.
- Agregar al plug-in una nueva funcionalidad que permita graficar el error, de manera tal que el usuario pueda observar lo que está ocurriendo durante el entrenamiento.
- Agregar otras funciones de transferencias que permitan realizar el entrenamiento haciendo uso de las mismas.

REFERENCIAS BIBLIOGRÁFICAS

[1]. **Tanco, Fernando.** Grupo de Inteligencia Artificial y Robótica. [En línea] [Citado: 19 de Septiembre del 2009.] http://www.secyt.frba.utn.edu.ar/gia/RNA.pdf

[2]. **Burgos, Palacios.** Herramientas en GNU/Linux para estudiantes universitarios: Redes Neuronales con GNU/Linux. [En línea] [Citado: 17 de Septiembre del 2009.] http://es.tldp.org/Presentaciones/200304curso-glisa/redes_neuronales/curso-glisa-redes_neuronales.pdf

[3]. **Mejias Cesar, Yuleidys.** Influencia de la variación de las funciones de transferencia en el Perceptrón Multicapa, en ambiente local y distribuido. Ciudad Habana : s.n., 2010.

[4]. **Ochoa Izquierdo, Isbel; Pardo Cruz, José Carlos.** Algoritmo para la distribución de Perceptrones Multicapa. 2010.

[5]. **Gutiérrez, Elizabeth.** Redes Neuronales Ventajas y Desventajas. . [En línea] [Citado: 23 de Octubre del 2009.] http://egkafati.bligoo.com/content/view/184582/Redes-neuronales-ventajas-y-desventajas.html

[6]. **Ballesteros, Alfonso.** Neural Networks Framework. [En línea] [Citado: 13 de Noviembre del 2009.] http://www.redes-neuronales.netfirms.com/tutorial-redes-neuronales/clasificacion-de-redes-neuronales-respecto-al-aprendizaje.htm

[7]. **Soria, Emilio; Blanco, Antonio.** Redes Neuronales Artificiales. [En línea] [Citado: 15 de Noviembre del 2009.] http://www.acta.es/articulos_mf/19023.PDF

[8]. **Ballesteros, Alfonso.** Neural Networks Framework. [En línea] [Citado: 2 de Enero del 2010.] http://www.redes-neuronales.netfirms.com/tutorial-redes-neuronales/Las-redes-neuronales-multicapa.htm

[9]. **Ballesteros, Alfonso.** Neural Networks Framework. [En línea] [Citado: 23 de Enero del 2010.] http://www.redes-neuronales.netfirms.com/tutorial-redes-neuronales/el-perceptron-simple.htm

[10]. **Acosta, María Isabel; Salazar, Harold; Zuluaga, Camilo A.** Universidad Tecnológica de Pereira. [En línea] [Citado: 27 de Enero del 2010.] http://ohm.utp.edu.co/neuronales/Download/Backpropagation.pdf

[11]. **Acosta, María Isabel; Salazar, Harold; Zuluaga, Camilo A.** Universidad Tecnológica de Pereira. [En línea] [Citado: 2 de Febrero del 2010.] http://ohm.utp.edu.co:16080/neuronales/capitulo2/Backpropagation/MomentumB.htm

[12]. **Yunta Rodríguez, Luis.** Bases de datos documentales: estructura y uso. [En línea] [Citado: 12 de Febrero del 2010.] http://www.unav.es/dpp/documentacion/proteger/lryunta.pdf

[13]. **Petersen, Scot.** searchcio-midmarket.com. [En línea] [Citado: 28 de Marzo del 2010.] http://searchcio-midmarket.techtarget.com/sDefinition/0,,sid183_gci212800,00.html

[14]. **Balduino, Ricardo.** Introduction to OpenUP. [En línea] [Citado: 15 de Febrero del 2010.] http://www.eclipse.org/epf/general/OpenUP.pdf

[15]. **Braun, David.** Unified Modeling Language (UML) Tutorial. [En línea] [Citado: 16 de Febrero del 2010.] http://atlas.kennesaw.edu/~dbraun/csis4650/A&D/UML_tutorial/what_is_uml.htm

[16]. **Boutell, Tom.** Boutell.Com. [En línea] [Citado: 18 de Febrero del 2010.] http://www.boutell.com/newfaq/definitions/java.html.

[17]. **Erickson, Marc.** What is Eclipse, and how do I use it. [En línea] [Citado: 17 de Febrero del 2010.] http://www.ibm.com/developerworks/opensource/library/os-eclipse.html

[18]. **Canales Mora, Roberto.** Informática Profesional. [En línea] [Citado: 19 de Febrero del 2010.] http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=vp

[19]. **Maldonado, Daniel Martin.** Software y soluciones tecnológicas Open Source para Pymes. [En línea] [Citado: 14 de Mayo del 2010.] http://www.aplicacionesempresariales.com/sqlite-el-motor-de-base-de-datos-agil-y-robusto.html

[20]. **Pecos, Daniel**. Universidad Jaume I de Castellón, DANIELPECOS.com. [Citado: 19 de Febrero del 2010]. http://danielpecos.com/docs/mysql_postgres/x57.html

[21]. **Prieto, Félix.** Departamento de Informática Universidad de Valladolid. [En línea] [Citado: 19 de Febrero del 2010.] http://www.infor.uva.es/~felix/datos/priii/tr_patrones-2x4.pdf

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

BIBLIOGRAFÍA

Acosta, María Isabel; Salazar, Harold; Zuluaga, Camilo A. Universidad Tecnológica de Pereira. http://ohm.utp.edu.co/neuronales/Download/Backpropagation.pdf

Acosta, María Isabel; Salazar, Harold; Zuluaga, Camilo A. Universidad Tecnológica de Pereira. http://ohm.utp.edu.co:16080/neuronales/capitulo2/Backpropagation/MomentumB.htm

Balduino, Ricardo. Introduction to OpenUP. http://www.eclipse.org/epf/general/OpenUP.pdf

Ballesteros, Alfonso. Neural Networks Framework. http://www.redes-neuronales.netfirms.com/tutorial-redes-neuronales/clasificacion-de-redes-neuronales-respecto-al-aprendizaje.htm

Ballesteros, Alfonso. Neural Networks Framework. http://www.redes-neuronales.netfirms.com/tutorial-redes-neuronales/Las-redes-neuronales-multicapa.htm

Ballesteros, Alfonso. Neural Networks Framework. http://www.redes-neuronales.netfirms.com/tutorial-redes-neuronales/el-perceptron-simple.htm

Braun, David. Unified Modeling Language (UML) Tutorial. http://atlas.kennesaw.edu/~dbraun/csis4650/A&D/UML_tutorial/what_is_uml.htm

Boutell, Tom. Boutell.Com. http://www.boutell.com/newfaq/definitions/java.html.

Canales Mora, Roberto. Informática Profesional. http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=vp

Erickson, Marc. What is Eclipse, and how do I use it. http://www.ibm.com/developerworks/opensource/library/os-eclipse.html

Gutiérrez, Elizabeth. Redes Neuronales Ventajas y Desventajas. http://egkafati.bligoo.com/content/view/184582/Redes-neuronales-ventajas-y-desventajas.html

Maldonado, Daniel Martin. Software y soluciones tecnológicas Open Source para Pymes. http://www.aplicacionesempresariales.com/sqlite-el-motor-de-base-de-datos-agil-y-robusto.html

Mejias Cesar, Yuleidys. Influencia de la variación de las funciones de transferencia en el Perceptrón Multicapa, en ambiente local y distribuido. Ciudad Habana : s.n., 2010.

Ochoa Izquierdo, Isbel; Pardo Cruz, José Carlos. 2010. Algoritmo para la distribución de Perceptrones Multicapa. 2010.

Palacios, Burgos. Herramientas en GNU/Linux para estudiantes universitarios: Redes Neuronales con GNU/Linux. http://es.tldp.org/Presentaciones/200304curso-glisa/redes_neuronales/curso-glisa-redes_neuronales.pdf

Pecos, Daniel. Universidad Jaume I de Castellón, DANIELPECOS.com. http://danielpecos.com/docs/mysql_postgres/x57.html

Petersen, Scot. searchcio-midmarket.com. http://searchcio-midmarket.techtarget.com/sDefinition/0,,sid183_gci212800,00.html

Prieto, Félix. Departamento de Informática Universidad de Valladolid. http://www.infor.uva.es/~felix/datos/priii/tr_patrones-2x4.pdf

Soria, Emilio y Blanco, Antonio. Redes Neuronales Artificiales. http://www.acta.es/articulos_mf/19023.PDF

Spencer, Ken, Sheriff, Paul y Forte, Stephen. Model-View-Controller. http://msdn.microsoft.com/en-us/library/ms978748.aspx

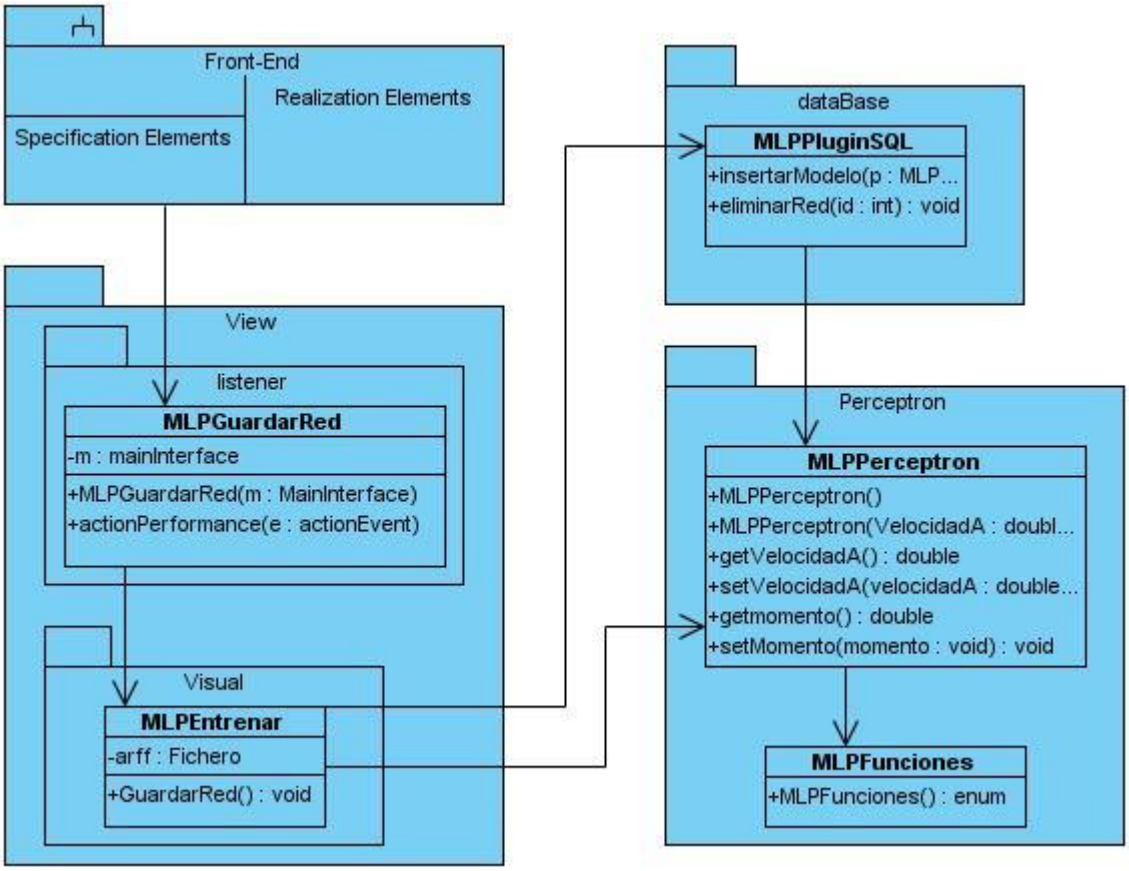
Tanco, Fernando. Grupo de Inteligencia Artificial y Robótica. http://www.secyt.frba.utn.edu.ar/gia/RNA.pdf

Yunta Rodríguez, Luis. Bases de datos documentales: estructura y uso. http://www.unav.es/dpp/documentacion/proteger/lryunta.pdf

ANEXOS

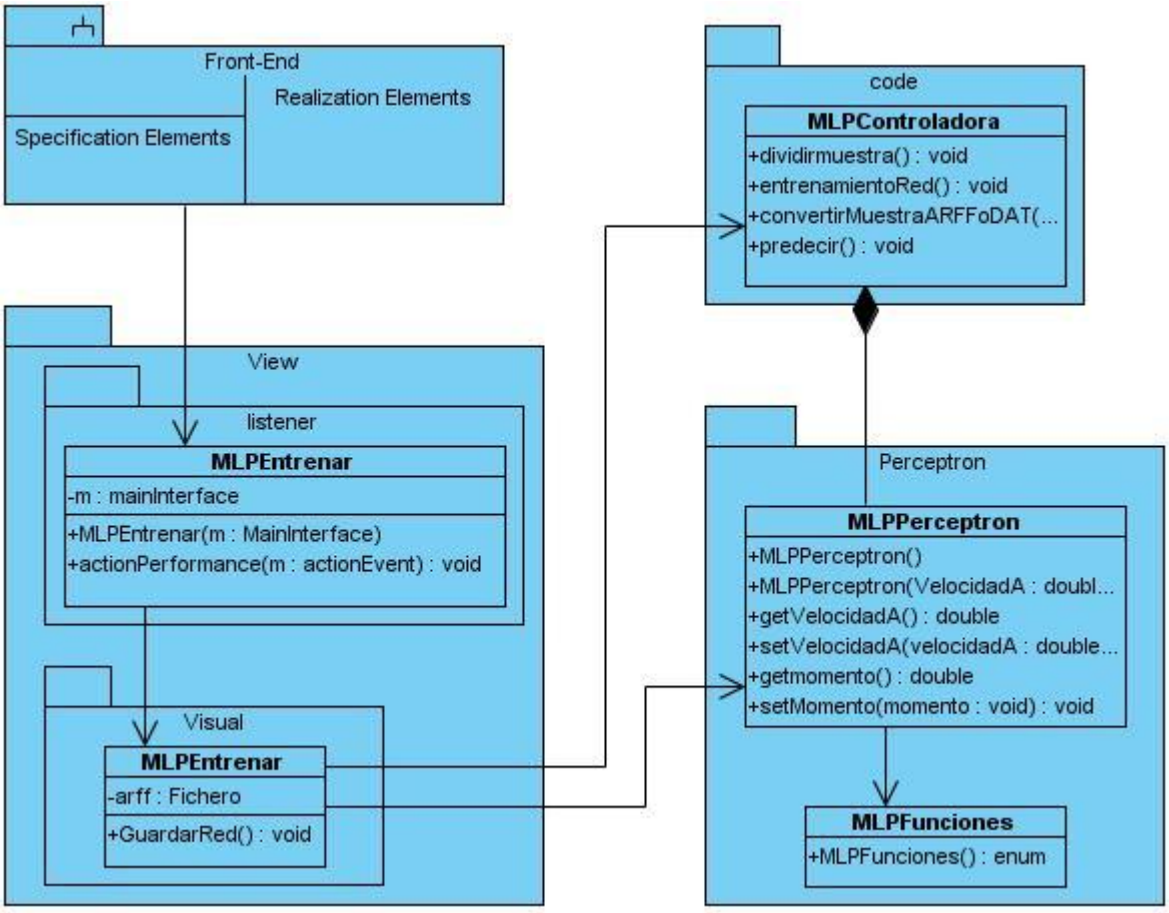
Anexo 1.

Diagrama de clases del diseño para el CU Guardar red Entrenada.



Anexo 2.

Diagrama de clases del diseño para el CU Entrenar Red



GLOSARIO

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

CASE: Computer Aided Software Engineering (Herramientas de ingeniería de software asistida por computadora).

Compuestos orgánicos: los compuestos o moléculas orgánicas son los compuestos químicos basados en Carbono, Hidrógeno y Oxígeno, y muchas veces con Nitrógeno, Azufre, Fósforo, Boro, Halógenos.

CU: caso de uso.

Front-End: es la herramienta para el manejo de los plug-ins en la plataforma alasGRATO.

IDE: Entorno de desarrollo integrado.

OpenUP: proceso unificado abierto.

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.

Ponderar: determinar el peso de algo. Atribuir un peso a un elemento de un conjunto con el fin de obtener la media ponderada.

SQL: Structure Query Language (Lenguaje de consulta estructurado).

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