**USO DE REDES NEURONALES, METODOLOGIAS AGILES, Y SOFTWARE LIBRE PARA EL DESARROLLO DE UNA HERRAMIENTA QUE RESUELVA PROBLEMAS DE PROGRAMACION NO LINEAL.**

**(GerMath.JS)**

**ESTUDIANTE:**

**WEYNDER GERMAN AGUIRRE BETANCOUR**

**UNIVERSIDAD DE LA AMAZONIA**

**FACULTAD DE INGENIERÍA**

**PROGRAMA DE INGENIERÍA DE SISTEMAS**

**FLORENCIA-CAQUETÁ**

**2019**

**USO DE REDES NEURONALES, METODOLOGIAS AGILES, Y SOFTWARE LIBRE PARA EL DESARROLLO DE UNA HERRAMIENTA QUE RESUELVA PROBLEMAS DE PROGRAMACION NO LINEAL.**

**(GerMath.JS)**

**ESTUDIANTE:**

**WEYNDER GERMAN AGUIRRE BETANCOUR**

**Estudiante, Universidad de la Amazonia – Florencia, Caquetá**

**Propuesta de trabajo de grado para optar el título de Ingeniero de sistemas de la Universidad de la Amazonia**

**UNIVERSIDAD DE LA AMAZONIA**

**FACULTAD DE INGENIERÍA**

**PROGRAMA DE INGENIERÍA DE SISTEMAS**

**FLORENCIA-CAQUETÁ**

**2019**

**Tabla de contenido**

[1. INTRODUCCIÓN 4](#_Toc24575519)

[2. FORMULACIÓN DEL PROBLEMA 5](#_Toc24575520)

[3. JUSTIFICACIÓN 6](#_Toc24575521)

[4. OBJETIVOS 7](#_Toc24575522)

[4.1 OBJETIVO GENERAL 7](#_Toc24575523)

[4.2 OBJETIVOS ESPECÍFICOS 8](#_Toc24575524)

[5. MARCO REFERENCIAL 8](#_Toc24575525)

[5.1 SOFTWARE LIBRE 8](#_Toc24575526)

[5.2 OPEN SOURCE 9](#_Toc24575527)

[5.3 METODOLOGIAS DE DESARROLLO SOFTWARE. 9](#_Toc24575528)

[5.3.1 METODOLOGIAS DE DESARROLLO TRADICIONAL 10](#_Toc24575529)

[5.3.2 METODOLOGIAS DE DESARROLLO AGIL 11](#_Toc24575530)

[5.4 LICENCIAMIENTO 12](#_Toc24575531)

[6. REDES NEURONALES ARTIFICIALES 12](#_Toc24575532)

[6.1 Origen. 12](#_Toc24575533)

[6.1.1 Definición. 13](#_Toc24575534)

[6.1.2 Perceptrón 15](#_Toc24575535)

[6.1.3 Perceptrón Multicapas (MLP) 16](#_Toc24575536)

[6.1.4 Pesos de las Neuronas Artificiales 16](#_Toc24575537)

[6.1.5 Función de entrada 16](#_Toc24575538)

[6.1.6 Función de salida (activación) 17](#_Toc24575539)

[6.1.7 Función de error de red 17](#_Toc24575540)

[6.1.8 Retropropagación De Error 17](#_Toc24575541)

[6.2 Aprendizaje Supervisado [15] 18](#_Toc24575542)

[6.3 Aprendizaje NO Supervisado 18](#_Toc24575543)

[7. MARCO METODOLOGICO 19](#_Toc24575544)

[7.1 CARACTERISTICAS DE LA METODOLOGIA XP 19](#_Toc24575545)

[7.2 ROLES DE LA METODOLOGIA EXTREME PROGRAMMING 20](#_Toc24575546)

[8. VENTAJAS Y DESVENTAJAS DE LA EXTREME PROGRAMMING 21](#_Toc24575547)

[8.1 VENTAJAS DE LA EXTREME PROGRAMMING: 21](#_Toc24575548)

[8.2 DESVENTAJAS DE LA EXTREME PROGRAMMING: 21](#_Toc24575549)

[9. FASES DE LA METODOLOGIA EXTREME PROGRAMMING. 21](#_Toc24575550)

[9.1 FASE DE EXPLORACION 21](#_Toc24575551)

[9.2 FASE DE PLANEACION DE LA ENTREGA 22](#_Toc24575552)

[9.3 FASE DE ITERACIONES 22](#_Toc24575553)

[9.4 FASE DE PRODUCCION 22](#_Toc24575554)

[9.5 FASE DE MANTENIMIENTO 22](#_Toc24575555)

[9.6 FASE DE MUERTE 23](#_Toc24575556)

[9.7 ENCAPSULAMIENTO EN FASES FUNCIONALES. 23](#_Toc24575557)

[10. CRONOGRAMA 25](#_Toc24575558)

[11. Referencias 28](#_Toc24575559)

# INTRODUCCIÓN

En el reino unido durante la segunda guerra mundial, se lograron establecer modelos eficaces para el desarrollo de las operaciones militares, siendo este el precedente histórico de la *Investigación De Operaciones* **– *IO***, que en su definición [1] abarca desde la toma de decisiones en grandes organizaciones, hasta la optimización de los procesos que se lleven a cabo en estas, también es de resaltar el desarrollo del método Simplex en el año de 1947 por George Dantzig [2], quien proporciono la posibilidad de solucionar problemas lineales multivariados, con esto, y debido a la llegada de la Revolución Industrial en 1880, los pequeños empresarios comenzaron a implementar maquinaria y nuevas tecnologías, lo cual origino su crecimiento y con ello la necesidad de aplicar la Investigación de Operaciones, para solucionar los problemas que surgen en la dirección y en la administración de sistemas de hombres, maquinas, materiales y/o dinero, en la industria, en los negocios, en el gobierno, en la defensa entre otros, notándose que por su enfoque cuantitativo, y apoyado por las matemáticas, se planteaban soluciones eficientes, para ese entonces, y con la llegada de los ordenadores se mejoró considerablemente la velocidad y capacidad de procesamiento de grandes cantidades de datos, facilitando así la solución de estos problemas en las organizaciones.

De esta forma se puede observar como mediante la utilización de modelos matemáticos se logra plasmar los acontecimientos u/o procesos de la cotidianidad, intentando de tal forma medir o cuantificar el mundo real, y optimizar eficientemente los procesos que en este se enmarcan, de tal forma la *Investigación de Operaciones* se puede aplicar a cualquier proceso cotidiano logrando modelar las variables que influyen en dicho momento y en el objetivo a analizar, determinando de forma canónica si lo que se desea es minimizar o maximizar el modelo planteado, entrados en el tema del modelamiento y lo elástico de su campo de acción, nos centraremos en la rama de la ***IO*** que trata sobre la solución de *Problemas No Lineales (****PNL****),* los cuales se caracterizan por la ausencia de linealidad, ya sea en la función objetivo, en las restricciones del problema, o en ambos casos*,* nuestro problema central se establece en, como solucionar problemas de optimización restringida con las características antes mencionadas.

En vista de lo anterior se busca dar solución a este problema utilizando *REDES* *NEURONALES* para optar al título de ingeniero de sistemas; Empleando para ello JAVASCRIPT, *EXPRESS*, *MARIADB*, *BRAIN***.***JS*, tomando como estandarte de desarrollo el software libre, que hará que las aplicaciones o herramientas sean de acceso libre y sin ánimo de lucro. De esta manera se espera tener la aplicación encapsulada en un servicio web consumible desde cualquier dispositivo, siendo esta aplicación una ayuda tanto para el estudiante, como para el docente, en cuanto a la amabilidad de los procesos de aprendizaje.

# FORMULACIÓN DEL PROBLEMA

Basados en el avance de las tecnologías de la información a lo largo del tiempo, y su auge significativo de los años 80’s y 90’s, siendo determinante la ampliación computacional en los procesos de creación e innovación de los viajes espaciales, el desarrollo de la robótica, las primeras pinceladas de la *Inteligencia**Artificial* **(*I.A*)**, la *Investigación de Operaciones* **(*I.O)***, entre otras, esta última se considera un campo bastante extenso, para este estudio haremos énfasis en la rama de la *Programación NO Lineal***. (PNL).**

Observamos que, en 1950 para fines bélicos de la segunda guerra mundial, los avances en este campo lograron optimizar los recursos existentes aumentando la eficacia y eficiencia de la tropa británica, para esto se han basado los avances principalmente en la Metaheuristica, que básicamente busca optimizar inteligentemente una situación o problema, mediante la conjunción de algoritmos, métodos y/o metodologías, haciendo uso del conocimiento específico mismo del problema, este es el punto de partida de nuestra investigación. Ahora, y en vista de la desbordante aplicabilidad de los sistemas de información, servicios web, y demás componentes tecnológicos que se ubican en la cima digital, tomamos el concepto ingenieril de las *Redes Neuronales Artificiales* **(RNA)**, las cuales emulan el comportamiento sináptico de las neuronas, enfocándose en la entrada, mantenimiento, manejo y salida de datos con alto grado de complejidad intrínseca en la toma de decisiones, aportando una salida a una o muchas entradas de datos.

Teniendo en cuenta lo anterior y los conceptos plasmados en este documento, se puede establecer una dirección respecto a la aplicabilidad de estas nuevas tecnologías, y si bien es cierto que el avance ha sido impresionante en la*Naturaleza Del Código (The Nature Of Code)*[3]**,** podemos aplicarlo a nuestro objetivo de **PNL**, y determinar cómo resolver problemas de optimización no lineal, con y sin restricciones. de esta forma lo que se espera es lograr enseñarle a la **RNA** los algoritmos Metaheurísticos necesarios para solucionar dichos problemas o en su defecto aproximar una solución óptima de gran calidad, por tanto, el objetivo de nuestra investigación es:

¿Cómo resolver problemas de programación no lineal, haciendo uso de las redes neuronales artificiales, tomando como eje fundamental la propiedad neural del autoaprendizaje, el concepto matemático del descenso del gradiente, utilizando el lenguaje de programación JavaScript?

# JUSTIFICACIÓN

Las Redes Neuronales Artificiales (RNA) son el punto de partida o por llamarlo de alguna forma el descubrimiento “precursor” de la Inteligencia Artificial (I.A), y esta, en su forma explícita emula el comportamiento biológico natural de las neuronas de tipo animal, brindando un aprendizaje neural que imita la forma como el cerebro humano, haciendo uso de su red de neuronas naturales procesa los datos y aporta una solución en función de los datos analizados, de esta forma puede enfocarse específicamente a cualquier tema que pudiera computarse; de esta manera realizando un aprendizaje supervisado que instruya a la **RNA** a responder a un problema específico, como lo es el objetivo principal de esta investigación, proporcionar solución a los Problemas de Optimización en Programación No Lineal (PNL).

Por otra parte, en los últimos años, la tecnología ha evolucionado de manera rápida dado que la humanidad se ha visto en la obligación de suplir necesidades de toda índole. Los Sistemas de información en su abrumante y acelerado cambio se han adaptado rápidamente a todos estos escalones de la modernización, en este orden de ideas, y en punta de lanza tenemos a los Servicios Web [4] que para efectos de este trabajo, nos proporcionaran el soporte estructural en el cual se desarrollara y expandirá el rango de uso de la misma, basándonos en su buen manejo de software máquina a máquina brindando una interoperabilidad asíncrona entre el cliente y el servidor, donde se alojara el Servicio Web que encapsulara la aplicación para ser consumida desde cualquier dispositivo con conexión a internet.

Como ingenieros nuestro deber es atender a las necesidades propias de nuestro entorno, haciendo énfasis en la resolución de problemáticas propias del núcleo del conocimiento de la ingeniería, para las cuales se necesite una solución directa o una conjugación de métodos o metodologías debido a la dificultad de su solución, siendo la optimización de dichas soluciones el objetivo fundamental de la estancia en el alma matter, es en este punto en donde surge la necesidad de, desarrollar una herramienta software que mediante el uso de nuevas tecnologías, metodologías, y conceptos, aporte o aproxime la solución a problemas de optimización en PNL, siendo este un excelente campo para demostrar los conocimientos adquiridos y pertinentes a la formación impartida, notándose la necesidad de una aplicación que solucione de forma automática cualquier problema de programación no lineal haciendo uso de los conceptos expuestos anteriormente.

# OBJETIVOS

## OBJETIVO GENERAL

Desarrollar una herramienta software, que, mediante el uso de Redes Neuronales Artificiales, proporcione la solución a cualquier problema de Programación No Lineal concerniente a los conceptos matemáticos de la Investigación De Operaciones.

## OBJETIVOS ESPECÍFICOS

Determinar un aprendizaje metodológico para una red neuronal que se ajuste a los conocimientos necesarios para la solución de los problemas de PNL.

Identificar una metodología de desarrollo software que se ajuste al tiempo planificado para el desarrollo de la herramienta.

Desarrollar la aplicación de tal forma que esta sea escalable, proporcional y adaptable a las necesidades de la institución.

# MARCO REFERENCIAL

## SOFTWARE LIBRE

***Free Software***, en inglés, [5] el cual se basa principalmente en 4 conceptos conocidos como libertades definidas por Richard Stallman, tales como:

* La Libertad de usar el programa con cualquier propósito.
* La Libertad de estudiar cómo funciona, y adaptar el programa a nuestras necesidades. “El acceso al código fuente es una condición estrictamente necesaria”.
* la Libertad de poder distribuir copias, con lo que se puede ayudar a los demás para obtención del material.
* La libertad de mejorar el producto y hacer públicas las mejoras realizadas al mismo para los demás.

## OPEN SOURCE

En principio se basa en las libertades de Stallman antes mencionadas, las cuales se encuentran en su mismo acuerdo de términos, siendo esta una mejora sustancial a la filosofía de Software Libre utilizada en este documento [6], de esta forma tenemos que:

1. Redistribución gratuita: La licencia no restringirá a ninguna de las partes a vender o regalar el software como un componente de una distribución agregada de software que contiene programas de varias fuentes diferentes. La licencia no requerirá una regalía u otra tarifa por dicha venta.

2. Código fuente: El programa debe incluir el código fuente y debe permitir la distribución tanto en el código fuente como en el formulario compilado.

3. Trabajos derivados: La licencia debe permitir modificaciones y trabajos derivados, y debe permitir que se distribuyan bajo los mismos términos que la licencia del software original.

4. Integridad del Código fuente del autor: La licencia debe permitir explícitamente la distribución de software creado a partir de código fuente modificado.

5. No discriminación contra personas o grupos: La licencia no debe discriminar a ninguna persona o grupo de personas.

6. No discriminación contra los campos de trabajo: La licencia no debe restringir a nadie el uso del programa en un campo específico de esfuerzo.

7. Distribución de licencia: Los derechos adjuntos al programa deben aplicarse a todos aquellos a quienes el programa se redistribuye sin la necesidad de que esas partes ejecuten una licencia adicional.

8. La licencia no debe ser específica para un producto: Los derechos adjuntos al programa no deben depender de que el programa forme parte de una distribución de software en particular.

9. La licencia no debe restringir otro software: La licencia no debe imponer restricciones a otro software que se distribuya junto con el software con licencia.

10. La licencia debe ser neutral en tecnología: Ninguna disposición de la licencia puede basarse en ninguna tecnología o estilo de interfaz individual.

## METODOLOGIAS DE DESARROLLO SOFTWARE.

Si hablamos como tal de la metodología de un proyecto, si es tradicional, se basara en planear a cabalidad los detalles rigurosos que pudieran afectar de uno u otro modo el desarrollo de las actividades o fases, sin embargo esto se convierte en un problema, debido al enfoque sistemático y pragmático de las nuevas tecnologías, las cuales causan que las metodologías tradicionales o “pesadas “ por llamarlas de algún modo, no se adapten de manera correcta a cambios específicos solicitados por el cliente, de esta forma al realizar una modificación en los requerimientos para tomar otro enfoque, se retrasarían cada una de las fases siguientes del desarrollo, lo cual es catastrófico si hablamos de presupuestos reducidos en una región que apenas empieza a requerir el desarrollo de sistemas de información o aplicaciones software consumibles desde dispositivos móviles para afrontar problemas en empresas pequeñas y medianas(pymes).

En respuesta a lo anterior, se han desarrollado metodologías, no tan robustas que brindan mayor libertad y ergonomía al permitir la integración de modificaciones en los requerimientos por parte del cliente sin afectar los diferentes procesos que se estén llevando a cabo, este el concepto de metodología ágil de desarrollo, permitiendo la adaptación rápida a cada cambio que pretenda el cliente, (*Tabla 1*.) debido intrínsecamente a que hace parte del equipo de desarrollo, participando activamente en reuniones constantes y día a día, orientando y corrigiendo cada problema tan pronto ocurra, de la mano del desarrolladores, el diseñador y cada uno de los integrantes del equipo de desarrollo.

Lo cual a la final logra que al usuario final se le entregue lo que realmente estaba buscando, ya que fue el quien superviso y aporto las ideas en cada fase del desarrollo, y puesta en marcha de la aplicación, garantizando aun después de su implantación su mejora constante, hasta que todas y cada una de las necesidades o historias de usuario del cliente sean satisfechas, llegando hasta la fase de muerte donde se le otorga total control al cliente y solo se bridaría soporte en caso de necesitarse. Siendo así, las metodologías que se pueden o no utilizar ya dependen de cada desarrollador o grupo de desarrolladores, pero en el campo de la programación practica se utilizan metodologías agiles para proyectos a corto plazo, y en proyecto bastante grandes se recomiendan metodologías tradicionales, dependiendo de cada caso.

### METODOLOGIAS DE DESARROLLO TRADICIONAL

Estas metodologías imponen una disciplina de trabajo sobre el proceso de desarrollo del software con el fin de conseguir un software más eficiente. Para ello, se hace el énfasis en la planificación total de todo el trabajo a realizar, organizando por tareas cronografiadas, dependiendo cada una de la otra, siendo así una cadena de estabones donde el más débil puede arriesgar al resto, no obstantes estas metodologías brindan mecanismos para blindar estos acontecimientos y que el sistema no se vea vulnerable o frágil, ya que después de esto es que empieza el proceso de producción del software, la puesta en marcha de la extrema y organizada planificación definida por procesos, roles, actividades, métodos y herramientas que se pueden usar para cumplir el objetivo organizativo del desarrollo documentando cada cambio rigurosamente, mediante UML llevando el registro del análisis , diseño, implementación, y documentación de sistemas orientados a objetos.

Entre las metodologías tradicionales o pesadas podemos citar:

• RUP (Rational Unified Procces)

• MSF (Microsoft Solution Framework)

• Win-Win Spiral Model

• Iconix

### METODOLOGIAS DE DESARROLLO AGIL

Desde 2001 se fundó “*The Agile Aliance”* una organización sin ánimo de lucro dedicada a promover los conceptos relacionados con el desarrollo ágil, integrada en si por 17 expertos en el área del desarrollo de software para entonces, de esta forma en EEUU se empezaron a dar pasos agigantados en vías de construir una solida metodología para el desarrollo de todo tipo de aplicaciones o proyectos, para esto se creó en ese entonces y se conserva hasta ahora, **El Manifiesto Ágil.** [7]

Enfocados claramente en encontrar la forma de modelar, organizar, y bridar la seguridad organizativa que pudiera encontrarse en la competencia, fue así como paso a paso se tomaron partes funcionales de otras metodologías y poco a poco se fue creando un consolidado de principios y reglas que se deben seguir y en los cuales se rige esta filosofía metodológica, donde priman los intereses del cliente y la adaptabilidad, la velocidad de adaptación y en sí, el manejo y la unidad que se espera transmitir haciendo uso de estas metodologías, donde la integración y factorización priman sobre temas tan arbitrarios como la documentación, donde se puede cambiar por petición o por refactoring, haciendo reingeniería aun método obsoleto, evitando así perder tiempo en banalidades que poco o nada afectan la funcionalidad del sistema véase Tabla 1.

*Tabla 1: Comparativa entre metodologías de desarrollo ágiles y metodologías de desarrollo tradicional.* [8]

|  |  |
| --- | --- |
| **Metodologías Agiles** | **Metodologías Tradicionales** |
| Basadas en heurísticas provenientes de prácticas de producción de código | Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo |
| Especialmente preparados para cambios durante el proyecto | Cierta resistencia a los cambios. |
| Impuestas internamente (por el equipo) | Impuestas externamente |
| Proceso menos controlado, con pocos principios | Proceso mucho más controlado, con numerosas políticas/normas |
| No existe contrato tradicional o al menos es bastante flexible | Existe un contrato prefijado |
| El cliente es parte del equipo de desarrollo | El cliente interactúa con el equipo de desarrollo mediante reuniones |
| Grupos pequeños (<10 integrantes) y trabajando en el mismo sitio | Grupos grandes y posiblemente distribuidos |
| Pocos artefactos | Más artefactos |
| Pocos roles | Más roles |
| Menos énfasis en la arquitectura de software | La arquitectura de software es esencial y se expresa mediante modelos |

Entre las metodologías ágiles más destacadas hasta el momento se pueden nombrar:

* XP (Extreme Programming).
* Crystal Clear.
* DSDM (Dynamic Systems Developmemt Method).
* ASD (Adaptive Software Development).
* Extreme Modeling.

## LICENCIAMIENTO

Una licencia de software es la autorización otorgada por el titular para la utilización de un producto; continuando con la linealidad del proyecto utilizaremos un licenciamiento totalmente libre, tal como la licencia GPL;

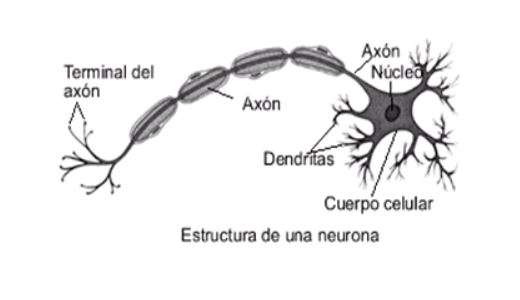
La GNU GPL (General Public License o Licencia Publica General), este tipo de licencia restringe básicamente a poner a disposición los códigos fuentes de los programas, entendiendo que las fuentes son los formatos en los que se pueden aplicar las posibles modificaciones al aplicativo (Free Software Fundación) [5].

# REDES NEURONALES ARTIFICIALES

## Origen.

Neuronas Biológicas: La neurona es la unidad funcional y estructural del sistema nervioso que produce y transmite el impulso nervioso. (*Figura 1*) Se encuentra formada por tres partes: el cuerpo neural o soma, el axón, las dendritas y zonas de conexión entre una neurona y otra llamada, sinapsis. [9]

Figura 1. Representación gráfica de una neurona biológica. [10]



### Definición.

Las Redes Neuronales Artificiales (RNA) son modelos cognoscitivos – computacionales basados en la estructura neuronal del sistema nerviosos, debido a esto, intentan replicar su comportamiento, con esto, las RNA pueden ser entrenadas para llevar a cabo tareas específicas, también se puede ejercer sobre ellas un entrenamiento, que iteradamente conlleva a un aprendizaje, y posterior adaptación del conocimiento en una mejora continua, es de notar que existen diferentes algoritmos de aprendizaje como son: el aprendizaje supervisado, el no supervisado y el aprendizaje por refuerzo, [11] cada uno con características diferentes apuntando a un mismo objetivo, que sería el correcto aprendizaje de la red neuronal en base a el conocimiento impartido.

Las RNA se componen de neuronas conectadas entre sí, estas transmiten información de forma innata, reciben, procesan y envían datos entre sí, estas neuronas interconectadas responden al aprendizaje impartido, retornando una salida con la respuesta a la tarea específica para la cual fue entrenada la red.

Uno de los primeros modelos matemáticos de una neurona fue el propuesto por McCulloch y Pitts en 1.943 [12] y en él se basan las redes neuronales actuales, en este modelo cada neurona consta de un conjunto de entradas ,y una sola salida,, cada entrada es afectada por un coeficiente denominado peso y que se representa por la letra . El subíndice refleja que el peso afecta a dicha entrada, y hace alusión a la neurona en ese determinado momento como lo muestra la *figur*a *2*.

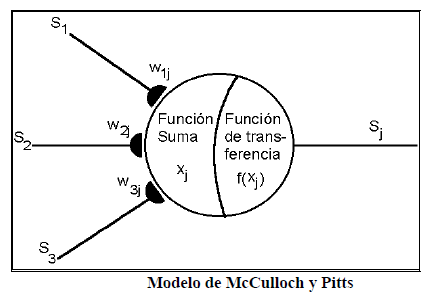
Entonces: la cantidad referente a la suma del producto de cada entrada multiplicada por su peso respectivo se denomina “***Activación****”* de la neurona  **,** la salida de la neurona es una función de activación de esta, en resumen:

**🡪** = es un valor “*umbral”* **(1)**

De esta forma:

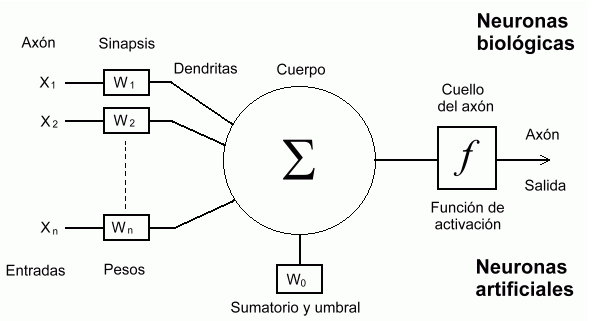
**(2)**

Figura 2. Modelo de Neurona de McCulloch y Pitts. [13]



La figura 3, expone un ejemplo de modelo neural con entradas, que consta de:

Un conjunto de entradas **, l**os pesos sinápticos , correspondientes a cada entrada, este se va ajustando de forma automática a medida que la red neuronal va aprendiendo, una función de agregación, **,** una función de activación, . mantiene el conjunto de valores de salida normalmente entre (0,1) o (-1,1), la más habitual es la función sigmoidea que se puede observar en la tabla 2, y una salida, .

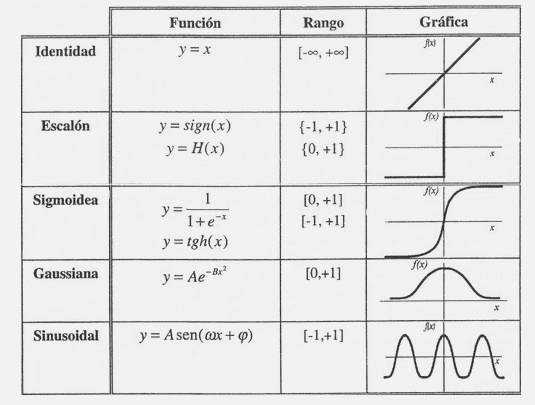
Figura 3. Modelo de comparativo de neuronas (biológica – artificial) [14]

Las entradas son el estímulo que la neurona recibido del entorno, y la salida es la respuesta a tal estímulo. La neurona puede adaptarse y aprender del medio en que se encuentre, modificando el valor de sus pesos sinápticos, por ello son conocidos como los parámetros libres del modelo, ya que pueden ser modificados y adaptados, en este modelo, la salida neuronal está dada por:

**(3)**

La función de activación se elige de acuerdo a la tarea realizada por la neurona.

Tabla 2. Diferentes tipos de función de activación. [14]



### Perceptrón

El perceptrón es la red neuronal más básica que existe, teniendo en cuenta que su aprendizaje es supervisado, se remonta a la década de los años 1.950. El funcionamiento es muy sencillo, simplemente lee las entradas, suma todas las entradas de acuerdo a unos pesos y el resultado lo introduce en una *función de activación* que genera el resultado final, el entrenamiento del perceptrón no es más que determinar los pesos sinápticos y el umbral que haga que la entrada se ajuste a la salida. Para determinar esto, se sigue un proceso adaptativo e iterativo, el proceso da inicio con valores aleatorios y se van modificando estos valores según la diferencia entre los valores deseados y los calculados por el perceptrón utilizando la medida del error para aproximar el próximo resultado o la próxima salida. [15]

1. Inicializar pesos y umbrales
2. Bucle: hasta resultado de pesos sea aceptable
   * Bucle: para todos los ejemplos
     + Leer valores de entrada
     + Calcular error
     + Actualizar pesos según el error
       - Actualizar pesos de entradas
       - Actualizar el umbral

Nota: Solo es capaz de representarfuncioneslineales debido a que no dispone de capas ocultas.

### Perceptrón Multicapas (MLP)

Es un tipo de red neuronal que tiene una capa de entrada y una capa de salida, y una o más capas ocultas en medio. Cada capa tiene un número de neuronas fijo, pero potencialmente diferente, es decir, será fijo durante todas iteraciones o épocas de la capacitación, después se podrá modificar. Cada conexión de neuronas tiene un peso, una función de entrada y una función de salida. (ver figura 4)

Figura 4. Ejemplo de Perceptrón Multicapa. [16]



### Pesos de las Neuronas Artificiales

En una red biológica, se envía una señal eléctrica por el axón hasta las dendritas de las neuronas. La fuerza de la señal determina la cantidad de influencia que la neurona encendida tiene sobre las demás neuronas, las RNA son el análogo a este tipo de red biológica, y simula la fuerza de la señal por medio del peso de la conexión de salida, que claramente podemos definir, pero a su vez la red neuronal puede ajustar según sus necesidades. De esta forma el peso se reparte entre la capacidad de ajustarse entre épocas de capacitación, y la fuerza de la señal, que denota la influencia en la red.

### Función de entrada

El tipo de función de entrada más habitual se llama suma ponderada. Una neurona de entrada está conectada a neuronas de la capa anterior de la red, la suma ponderada de las entradas para se calcula añadiendo el producto del valor de entrada de cada conexión por veces el peso de la conexión a lo largo de todas las entradas . [16]

**(4)**

### Función de salida (activación)

El tipo de función de activación que más habitualmente se usa en las redes MLP es la Función sigmoidea. Para una suma ponderada **A** para una neurona determinada, el valor sigmoideo de **, A** es calculado por:

**(5)**

**Nota:** La función sigmoidea es no lineal, lo que la hace muy adecuada para su uso en redes neuronales.

### Función de error de red

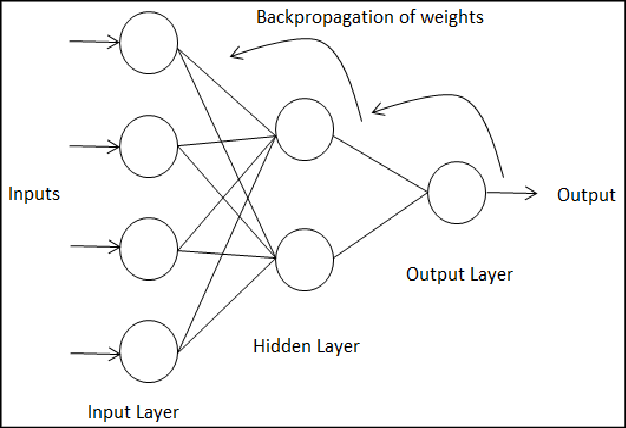
La función de error más usada con las redes MLP es la función de error cuadrático medio. La cual lo que hace es calcular la *"distancia"* media entre el valor real que el programa calcula, y el valor esperado de los datos de capacitación, dadas neuronas de salida, para cada suma ponderada de la salida , el programa de capacitación calcula la diferencia entre el valor de los datos de capacitación y el valor de la red, lo eleva al cuadrado, suma esos valores de todas las neuronas de salida y lo divide por el número de neuronas de salida para llegar al error de salida total . [17]

**(6)**

### Retropropagación De Error

Cuando una época de capacitación acaba, el programa de capacitación calcula el error de la red neuronal, y modifica los pesos de la conexión a lo largo de la red, empezando en la capa de salida y propagando el error hacia atrás, hacia la capa de entrada, ajustando el peso de cada conexión neuronal mientras recorre la red ver figura 5, a esto se le conoce como retropropagación de error y es una técnica ampliamente utilizada para capacitar redes neuronales.

Figura 5. RNA BackPropagation. [18]



## Aprendizaje Supervisado [17]

Ocurre cuando se toma un conjunto de datos muéstrales de ejemplo, y cada ejemplo está formado por una entrada y una salida deseada. Se introducen entradas hasta que la red neuronal arroje un porcentaje de fiabilidad pertinente en la salida deseada.

Entonces tenemos:

1. *alimentar la red con datos conocidos.*
2. *¿La red arroja respuestas correctas?* 
   * *Sí (dentro del porcentaje de fiabilidad): Ir al paso 3.*
   * *No:* 
     1. *Ajustar los pesos de las conexiones de la red.*
     2. *Ir al paso 1.*
3. *Fin.*

## Aprendizaje NO Supervisado

El entrenamiento no supervisado es aquel en que la red tiene que entender las entradas sin ayuda de la salida, cabe aclarar que en su gran mayoría se prefiere utilizar aprendizaje supervisado; tenemos entonces que la red es alimentada con entradas, pero no con las salidas deseadas, el sistema decidirá las características y atributos que deban tener los datos para ser agrupados.

# MARCO METODOLOGICO

Teniendo en cuenta las diferentes opciones, corrientes y filosofías de desarrollo, es necesario buscar una metodología que brinde seguridad y eficiencia, esta metodología debe cumplir a cabalidad el objetivo propuesto y bridarnos la alternativa de agilizar el proceso de desarrollo siendo un marco ideal y eficiente a la hora de la adaptación a cambios o de reveses definidos por el cliente; Siendo esas las premisas de esta búsqueda, se decidió utilizar la metodología **EXTREME PROGRAMMING (XP). (**Ver figura 6.)

## CARACTERISTICAS DE LA METODOLOGIA XP

SIMPLICIDAD: La simplicidad consiste en desarrollar solo el sistema que realmente se necesita. Implica resolver en cada momento solo las necesidades actuales.

FEEDBACK: Una metodología basada en el desarrollo iterativo de pequeñas partes, con entregas y pruebas frecuentes y continuas, proporciona un flujo de retroinformación valioso para detectar los problemas o desviaciones.

DECISIÓN:

• Implica saber tomar decisiones difíciles.

• Reparar un error cuando se detecta.

• Mejorar el código siempre tras el feedback y las sucesivas iteraciones.

COMUNICACIÓN: Algunos problemas en los proyectos tienen origen en que alguien no dijo algo importante en algún momento, XP hace casi imposible la falta de comunicación, ya que pone en comunicación directa y continua a clientes y desarrolladores.

PLANIFICACION: Se hacen las historias de usuario y se planifica en qué orden se van a hacer y las mini-versiones, la planificación se revisa continuamente.

VERSIONES PEQUEÑAS: Las mini-versiones deben ser lo suficientemente pequeñas como para poder hacer una cada pocas semanas. Deben ser versiones que ofrezcan algo útil al usuario final y no fragmentos de código que no pueda ver funcionando.

DISEÑO SIMPLE: Hacer siempre lo mínimo imprescindible de la forma más sencilla posible, mantener siempre sencillo el código.

INTEGRACION CONTINUA: Debe tenerse siempre un ejecutable del proyecto que funcione y en cuanto se tenga una nueva pequeña funcionalidad, debe recopilarse y probarse. Es un error mantener una versión congelada dos meses mientras se hacen mejoras y luego integrarlas todas de golpe.

RITMO SOSTENIBLE: Se debe trabajar a un ritmo que se pueda mantener indefinidamente. Esto quiere decir que no debe haber días muertos, en que no se sabe qué hacer y no se deben hacer en exceso horas otros días. Hay que trabajar para conseguir el objetivo cercano de terminar una historia de usuario o una mini-versión.

## ROLES DE LA METODOLOGIA EXTREME PROGRAMMING

**CLIENTE:** Responsable de definir y conducir el proyecto, de igual manera los objetivos.

**PROGRAMADORES:** Estiman el tiempo y el coste del proyecto, además de llevarlo a feliz término en cuanto a los rigores del desarrollo.

**TESTER:** Encargado de pruebas.

**TRACKER:** Encargado del seguimiento.

**COACH:** su papel es orientar y guiar el desarrollo del proyecto.

**BIG BOSS:** Gestor del proyecto.

# VENTAJAS Y DESVENTAJAS DE LA EXTREME PROGRAMMING

## VENTAJAS DE LA EXTREME PROGRAMMING:

Una de sus principales ventajas o la más resaltante a la hora de emprender los desarrollos es la gran adaptabilidad que ofrece esta metodología, siendo así escalable, adaptable y elástica, gracias a esto su utilidad y versatilidad se ven inmersas en lo amplio de su campo de acción, otra característica resaltable es la optimización del tiempo de desarrollo, al centrarse solo en los detalles funcionales de la aplicación, dejando de lado detalles o propiedades que no atacan agresivamente la funcionalidad del sistema( Diseño, Documentación, Soporte de Manual).

## DESVENTAJAS DE LA EXTREME PROGRAMMING:

Su principal desventaja es que los costes de tiempo y costo de obra de mano, o desarrollo funcional, no pueden ser definidos al inicio del proyecto, debido a su naturaleza iterativa e incremental, que soluciona fallas tan pronto son detectadas, lo cual hace que se aumente el tiempo de desarrollo y el coste, no obstante, también es de aclarar que, en felices términos sin sobresaltos generaría un ahorro sustancial en el tiempo de desarrollo funcional de la aplicación.

# FASES DE LA METODOLOGIA EXTREME PROGRAMMING.

## FASE DE EXPLORACION

En esta fase se exponen a grandes rasgos las necesidades del cliente en las historias de usuario, para la primer entrega del producto, se prueba las tecnologías que serán utilizadas, y se construye un prototipo con las mismas, evaluando el desempeño y la posible respuesta ante las necesidades venideras, de aquí es donde parte la fase de planeación, sin perder de vista la individualidad de la misma ya que el cliente puede redactar más historias de usuario si lo considera necesario. [19]

**Nota**: se recomienda que las historias de usuario no superen las 3 líneas.

## FASE DE PLANEACION DE LA ENTREGA

Ya en esta fase se estable la prioridad de cada historia de usuario, o actividad a realizar, esta fase es supremamente importante, ya que en términos ingenieriles, puede ser nuestro “Callback” ya que se ejecutaría tan pronto inicia esta fase y a su vez sigue siendo recursiva en cada iteración debido a que dependiendo el tiempo en semanas que se empleó para llevar a cabalidad la primer fase del proyecto con el prototipo funcional, sirve de medición para calcular el tiempo en semanas que llevara hacer los cambios solicitados por el cliente, de esa forma el cronometro está en marcha, corriendo en contra, lo que supone que se deba trabajar las 40 horas a la semana, siendo respetuosos con el calendario con la velocidad del proyecto que es demarcada por la velocidad del equipo de desarrollo y la cantidad de tareas pendientes en esta fase, por lo regular se debe intentar distribuir bien la carga en todos los días de la semana para evitar desiertos o colapsos.

## FASE DE ITERACIONES

Esta fase abarca la serie de iteraciones que deben hacerse antes de entregar el resultado, resolviendo la mayor cantidad de errores que se pueda en cada iteración, lo cual no nos exime que por cuestiones de tiempo o de sobre esfuerzo, se entregue un prototipo funcional con algún tipo de error, este a su vez debe ser tenido en cuenta para trabajar en “Background” o segundo plano haciendo iteraciones individuales para corregir este y los demás errores que queden en el aires, pero sin permitir que dichos erros afecten el comportamiento del prototipo funcional, sin restarle calidad siendo un proceso muy transparente de mejora continua; en iteraciones de no más de tres semanas en las cuales se deben abordar tano los problemas nuevos como los sin resolver, calculando así con la velocidad del proyecto y la cantidad en semanas que llevara realizar las tareas específicas de las historias de usuario, las iteraciones restantes para el producto final.

## FASE DE PRODUCCION

Pruebas adicionales de funcionabilidad y respuesta, revisiones de rendimiento antes de la migración al entorno del cliente, inclusión de nuevas características dependiendo de las necesidades surgidas por la organización o cambios de última hora, en este punto las entregas se harán cada semana esperando pulir el producto final lo más que se pueda, en caso de funcionalidades extra no documentadas por el cliente a su debido tiempo serán solucionadas en la fase de mantenimiento generando un valor agregado al producto.

## FASE DE MANTENIMIENTO

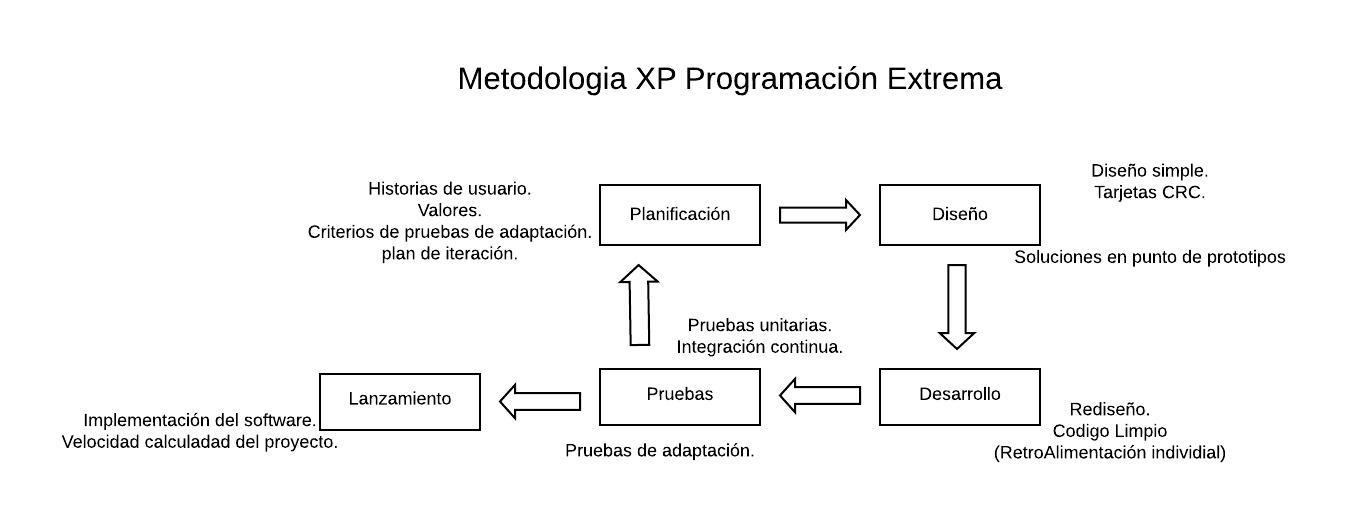
Mientras la primer versión se encuentra en producción, el sistema debe seguir iterando en la mejora continua, o en la adaptación de nuevas funcionalidades requeridas por el cliente, de esta forma la velocidad de desarrollo disminuirá debido a que se debe dividir el equipo en tanto a soporte como en la programación de las nuevas funciones o la optimización de las existentes.

## FASE DE MUERTE

En este punto, el cliente no tendrá más historias para incluir en el sistema, culminando a cabalidad con los requerimientos del cliente, esto requiere que se satisfagan las necesidades del usuario en otros aspectos como rendimiento y confiabilidad del sistema, abarcar tanto las historias funcionales como las no funcionales, brindando la seguridad, fiabilidad y estabilidad necesaria de un desarrollo software. Se generará entonces la documentación final del sistema y no se realizan más cambios en la arquitectura siendo este el reléase de esta aplicación.

## ENCAPSULAMIENTO EN FASES FUNCIONALES.

Es de aclarar que en este punto del documento y para facilidad de la abstracción de los diferentes elementos que interactúan en esta metodología, simplificaremos un poco las cosas al nivel de tomar solo 4 fases totalmente funcionales, estas son similares a las etapas de la metodología Rational United Proccess (**RUP)** [17] o cualquiera otra metodología robusta que se base en iteraciones, sin importar su tiempo de retorno.

Figura 6. (Metodología Extreme Programming).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| OBJETIVOS | TIEMPO DE EJECUCION | | | | | | | | | | | | | | | |
| AGO - 2019 | | | | SEP – 2019 | | | | OCT - 2019 | | | | NOV - 2019 | | | |
| OBJETIVO ESPECIFICO 1 | SEMANA | | | | | | | | | | | | | | | |
| 1 | **2** | **3** | **4** | **1** | **2** | **3** | **4** | **1** | **2** | **3** | **4** | **1** | **2** | **3** | **4** |
| Determinar una metodología ajustable que minimice los costes y tiempos requeridos para el desarrollo de la herramienta software |  | **X** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Identificar un lenguaje de programación que mediante la premisa de software libre logre codificar el programa. |  |  | **X** |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Establecer el algoritmo de solución más apropiado para dichos problemas. |  |  | **X** |  |  |  |  |  |  |  |  |  |  |  |  |  |
| OBJETIVO ESPECIFICO 2 | SEMANA | | | | | | | | | | | | | | | |
| **1** | **2** | **3** | **4** | **1** | **2** | **3** | **4** | **1** | **2** | **3** | **4** | **1** | **2** | **3** | **4** |
| Establecer el tipo de RNA necesaria para abarcar la totalidad del problema |  |  |  | **X** |  |  |  |  |  |  |  |  |  |  |  |  |
| Identificar el método de aprendizaje óptimo a impartir sobre la red neuronal para solucionar problemas PNL. |  |  |  | **X** |  |  |  |  |  |  |  |  |  |  |  |  |

# CRONOGRAMA

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| OBJETIVO ESPECIFICO 3 | SEMANA | | | | | | | | | | | | | | | |
| AGO - 2019 | | | | SEP – 2019 | | | | OCT - 2019 | | | | NOV - 2019 | | | |
| **1** | **2** | **3** | **4** | **1** | **2** | **3** | **4** | **1** | **2** | **3** | **4** | **1** | **2** | **3** | **4** |
| Planificación: |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Historias de Usuario |  |  |  |  | **X** |  |  |  |  |  |  |  |  |  |  |  |
| Revisión del estado de las Historias |  |  |  |  | **X** |  |  |  |  |  |  |  |  |  |  |  |
| Confirmación de las Historias |  |  |  |  |  | **X** |  |  |  |  |  |  |  |  |  |  |
| Diseño: |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Diseño de la Base de Datos. |  |  |  |  |  | **X** |  |  |  |  |  |  |  |  |  |  |
| Revisión, ajustes a la base de datos. |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Desarrollo: |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Gestión de Usuarios. |  |  |  |  |  | **X** | **X** |  |  |  |  |  |  |  |  |  |
| Gestión de Roles. |  |  |  |  |  |  | **X** | **X** |  |  |  |  |  |  |  |  |
| Gestión de Tareas |  |  |  |  |  |  |  | **X** | **X** |  |  |  |  | **X** |  |  |
| Gestor de soluciones |  |  |  |  |  |  |  |  | **X** |  |  |  |  | **X** | **X** |  |
| Analizador de Ecuaciones |  |  |  |  |  |  |  |  | **X** | **X** |  |  |  |  | **X** | **X** |
| Gestor de aprendizaje neural |  |  |  |  |  |  |  |  |  | **X** | **X** |  |  |  | **X** | **X** |
| Desarrollo de módulos independientes de PNL |  |  |  |  |  |  |  |  |  |  | **X** | **X** | **X** |  | **X** | **X** |
| Solución de PNL con restricciones |  |  |  |  |  |  |  |  |  |  |  |  | **X** |  |  |  |
| Solución de PNL sin restricciones |  |  |  |  |  |  |  |  |  |  |  |  | **X** | **X** |  |  |
| Despliegue del proyecto. |  |  |  |  |  |  |  |  |  |  |  |  |  | **X** | **X** |  |
| Pruebas: |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Test: mediante se desarrolla el sistema, cada iteración una prueba. |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **X** |
| Documentación: |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Manual de usuario. |  |  |  | **X** |  |  |  | **X** |  |  |  |  | **X** |  |  |  |
|  | | | | | | | | | | | | | | | | |
| OBJETIVO 4 | SEMANA | | | | | | | | | | | | | | | |
| **1** | **2** | **3** | **4** | **1** | **2** | **3** | **4** | **1** | **2** | **3** | **4** | **1** | **2** | **3** | **4** |
| Despliegue de la aplicación. |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **X** |

# Referencias

|  |  |
| --- | --- |
| [1] | J. C. G. Pérez, «Gestiopolis,» 21 de 06 del 2013 <https://www.gestiopolis.com/investigacion-de-operaciones-que-es-historia-y-metodologia/. |
| [2] | www.ecured.cu, «George Bernard Dantzig,» 05 del 2014. <https://www.ecured.cu/George\_Bernard\_Dantzig.> |

[3] Beck, Mike Beedle - Kent. *Manifiesto Ágil*. 2001. <https://agilemanifesto.org/iso/es/manifesto.html>.

[4] Boeree, Dr. C. George. *WebSpace.edu*. s.f. Universidad de Shippensburg . 2019. <http://webspace.ship.edu/cgboer/genesp/neuronas.html>.

[5] Calvo, Diego. *http://www.diegocalvo.es/perceptron/*. 08 de 12 de 2018. 09 de 2019. <http://www.diegocalvo.es/perceptron/>.

[6] *Metodología XP Programación Extrema (Metodología ágil)*. 12 de 2018. www.diegocalvo.com. 09 de 2019. <http://www.diegocalvo.es/metodologia-xp-programacion-extrema-metodologia-agil/>.

[7] Caparrini, Fernando Sancho. *Redes Neuronales: una visión superficial*. 26 de 12 de 2018. Dpto. de Ciencias de la Computación e Inteligencia Artificial. 09 de 2019. <http://www.cs.us.es/~fsancho/?e=72>.

[8] DeepAI. *BackPropagation*. 17 de 05 de 2019. DeepAI. 09 de 2019. <https://deepai.org/machine-learning-glossary-and-terms/backpropagation>.

[9] Free Software Foundation. *El sistema operativo GNU*. 12 de 12 de 2010. Free Software Foundation Inc. 2019. <https://www.gnu.org/licenses/licenses.es.html>.

[10] Gálvez, Rafael. *Sistema Nerviosos Humano*. s.f. https://weebly.com. 09 de 2019. <sistemanerviosohumano.weebly.com/estructura-neuronal.html>.

[11] IBM. *Rational Unified Process: Best Practices for Software*. 01 de 12 de 2001. IBM. 20 de 09 de 2019. <https://www.ibm.com/developerworks/ssa/library/cc-artificial-neural-networks-neuroph-machine-learning/index.html>.

[12] JosÈ H. CanÛs, Patricio Letelier y M™ Carmen PenadÈs. «MétodologÌas ágiles en el Desarrollo de Software.» *DSIC -Universidad PolitÈcnica de Valencia*

[13] MinTIC. *En TIC Confio*. 12 de Sep de 2015. Ministerio de las Tec. y TeleCom. 2019. <http://www.enticconfio.gov.co/internet-que-es-para-que-sirve>.

[14] Olabe, Xabier Basogain. «REDES NEURONALES ARTIFICIALES Y SUS APLICACIONES.» *Escuela Superior de Ingeniería de Bilbao.*

[15] OpenSource.Org. *OpenSource*. 04 de 08 de 2016. DigitalOcean. 2019. <https://opensource.org>.

[16] Pedro Larranaga, Inaki Inza, Abdelmalik Moujahid. «Tema 8. Redes Neuronales.» *Departamento de Ciencias de la Computacion e Inteligencia Artificial* I (2015).

[17] Penadés, Patricio Letelier & Mª Carmen. *Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP)*. 15 de 12 de 2005. Universidad Politécnica de Valencia (UPV). 20 de 09 de 2019. <http://www.cyta.com.ar/ta0502/v5n2a1.htm>.

[18] PITTS, WARREN S. MCCULLOCH AND WALTER. «A LOGICAL CALCULUS OF THE IDEAS IMMANENT IN NERVOUS ACTIVITY.» *MATHEMATICAL BIOPHYSICS FROM THE UNIVERSITY OF ILLINOIS, COLLEGE OF MEDICINE* V (1943)

[19] Shiffman, Daniel. *The Nature of Code: Simulating Natural Systems with Processing*. PapperBack, 2012.

[20] UNIVERSIDAD NACIONAL DE COLOMBIA. «REDES NEURONALES.» *disi.unal.edu.co*

[21] W3C. *https://www.w3.org/*. 1994. W3C. 08 de 2019. <https://www.w3.org/>