Facultad 2

Implementación de la herramienta SIRNA, orientada a la extracción interactiva de patrones a partir de imágenes digitales para NEUROLAB.

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

Autor: Ernesto Hita Ramos

Tutor (es): Dr.C Jose Alejandro Gutierrez Fernandez

Declaración de autoría

Declaro ser autor de la presente tesis y concedo a la empresa SOFTEL y 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 2015.

_____
Ernesto Hita Ramos

_____
Dr.C José Alejandro Gutiérrez Fernández

Dedicatoria

Agradecimientos

Agradezco:

Primeramente a Dios *por permitir que hoy este aquí.*

A mis padres Maritza y Ramón*, porque no solo durante los 5 años de mi carrera han sido la luz que me guía, sin ellos no fuese hoy quien soy.*

A mi tío Javier, *que aunque no esté aquí, siempre ha estado pendiente y preocupado por mí.*

A mi hermano Alejandro, *que me mortifica siempre, pero lo amo, (QUE RICO ESTOY).*

A mi abuela Lucia, *que no nos podemos ver, pero no podemos estar lejos uno del otro*

A mi familia, *Mi tía Marta, tío Alfredo (La Ciencia), tía Angui (La Mulatisima), a mi prima Sheily, a mi prima Emily, Ahmed, Agustín, Nubia, Miriam, Nivia y mis primos flacos, a Verónica, Augusto.*

A mis hermanos "UCI"*, Arlety (acsebastian), Roberto (rinfante) y Armando (aemesa), porque a pesar de nuestras eventuales diferencias, siempre nos hemos entendidos, siempre han estado ahí cuando los necesite (cuídense por ahí).*

A mi madrina "UCI" *Madelis Perez Gil, porque a pesar de que no tuvimos mucho vinculo, estaba pendiente de mí dándome consejos y avisándome cuando estaba ahogadillo (Matao y Salao).*

A mi tutor José Alejandro Gutiérrez Fernández, *por la paciencia que tuvo conmigo desde un principio*

A mi tribunal, *por haber sido sincero y haberme dado consejos muy útiles, durante la realización de este documento.*

A mi oponente Edisnel Carrazana, *muchas gracias por haber compartido la asignatura IA1 con usted y ahora la defensa de mi tesis, se ha convertido en mi amigo (DTB)*

A mis hermanos "Nuevo Vedado"*, Gilberto (The Pilot), Gabria (El Flaco), Jose Enrrique (Farruko) y su piquete sabroso, Ana Karla (La Ahijada).*

A Reinier Gastón Falcón, *de verdad gracias, me hubiese gustado que estuvieras aquí conmigo compartiendo este momento que tu querías venir.*

A las personas con las que compartí buenos y malos momentos dentro de la UCI

A los que confiaron en mí siempre sin dudar ni un segundo y a los que no confiaron en mí

Y a todas aquellas personas que de una forma u otra estuvieron presentes en la realización de este trabajo.

Dedicatoria

Dedico:

A mis padres

A mi Tío Javier

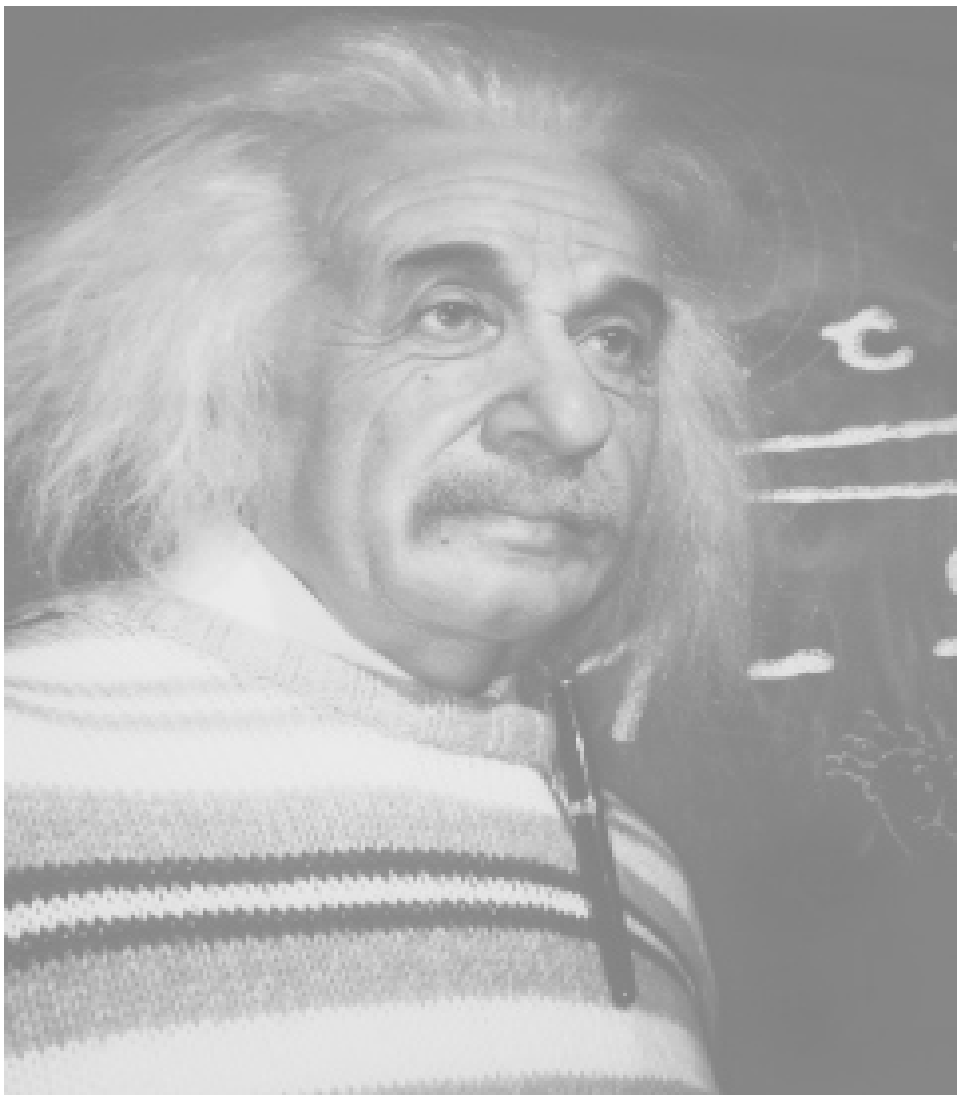
A mi familia

A mi difunto padrino Orlando

A la memoria de Reinier Gastón

A Cuba

Pensamiento



"La imaginación es más importante que el conocimiento, el conocimiento es limitado, la imaginación abraza el mundo"

Albert Einstein

(Físico alemán, 14/03/1879 – 18/04/1955)

Resumen

En la empresa “SOFTEL”, se ha desarrollado una aplicación web destinada al diseño, entrenamiento y uso de Redes Neuronales Artificiales denominada NEUROLAB. Su objetivo es brindar el servicio mencionado anteriormente, cuya red neuronal artificial entrenada podrá ser usada en la solución problemas que en la actualidad no pueden ser resueltos con procedimientos comunes.

NEUROLAB tiene opciones y funcionalidades que permiten que se le suministren los patrones a partir de datos almacenados, explícitamente en bases de datos pasados a documentos de textos con formatos específicos. Pero, no tiene la posibilidad de obtener directamente parámetros calculados a partir de otros orígenes, como son las imágenes digitales, tan altamente necesitadas del uso de este tipo de aplicaciones. Después de buscar e investigar la posibilidad de emplear rutinas y aplicaciones ya existentes y al alcance de la empresa, se decidió que ninguna se adaptaba a las exigencias expresadas por el grupo de desarrollo de SOFTEL. Por lo tanto se determinó la implementación de la herramienta que ha de permitir el suministro de patrones calculados directamente sobre imágenes digitales a NEUROLAB.

En este trabajo investigativo se exponen los detalles de la implementación de la herramienta SIRNA, el cual ha de permitir visualizar imágenes así como que el especialista pueda seleccionar regiones definidas dentro de ellas para calcular y extraer patrones de colores que son entregados a NEUROLAB para entrenar redes neuronales, y con esto garantizar el entrenamiento de las mismas. Además, posibilita hacer las pruebas de una red neuronal artificial entrenada.

Palabras claves: patrones, imágenes digitales, red neuronal artificial, entrenamiento.

ÍNDICE GENERAL

Índice general

CAPÍTULO 1. Fundamentación teórica

INTRODUCCIÓN

A la par del desarrollo de las tecnologías informáticas a nivel mundial y la incorporación a estas del conocimiento humano, haciéndolas cada vez más inteligentes, en Cuba se ha trabajado en la informatización de un gran número de sectores: económicos, sociales, académicos, empresariales, etc. empleando cada vez más herramientas y técnicas de la Inteligencia Artificial (IA). Precisamente el empleo de las mencionadas técnicas da un gran valor agregado a los sistemas de información y en muchos casos garantiza que el costo disminuya y con este, el tiempo que se va a emplear en el desarrollo de los sistemas de ayuda a la toma de decisiones.

En la empresa SOFTEL, se ha desarrollado una aplicación para posibilitar diseñar, entrenar y utilizar las Redes Neuronales Artificiales (RNA), sin que el usuario necesite tener conocimientos de programación, la aplicación se ha denominado NEUROLAB.

En la actualidad los datos se le suministran a NEUROLAB desde ficheros texto donde están bien definidas las variables (atributos) y los individuos (registros). También es necesario que a NEUROLAB se le puedan suministrar directamente los datos a partir de parámetros calculados sobre imágenes digitales, ya que el empleo y explotación de este tipo de datos es bastante frecuente y ampliamente utilizado en la solución a disimiles tareas. Así sería posible utilizar esta herramienta para la detección y estudio de eventos reales presentes en las imágenes digitales provenientes de diferentes esferas, como son:

1. La determinación de los niveles de funcionalidad renal a partir de las imágenes del riñón, en estudios de Medicina Nuclear.
2. La determinación de varios tipos de lesiones ateroescleróticas a partir de imágenes tomadas a las paredes internas de diferentes arterias.
3. La determinación del grado de quemaduras en pacientes a partir de imágenes tomadas de la piel.
4. La determinación de diferentes tipos de nubes a partir de imágenes satelitales.
5. La determinación de la presencia de elementos de interés en tejidos expuestos al microscopio.

6. En general, donde quiera que en una imagen digital puedan existir diferentes eventos, caracterizados por coloraciones propias e identificativas, y se puedan medir variables que los identifiquen.

Actualmente, NEUROLAB permite la introducción de los patrones que han de usarse en el entrenamiento, desde archivos de texto generados de forma manual. Pero para entrenar una red neuronal con el objetivo de determinar eventos presentes en una imagen digital, el proceso es más complicado y se emplea mucho tiempo para extraer los datos de las imágenes, para conformar los patrones y guardarlos para que más tarde NEUROLAB haga uso de los mismos.

SOFTEL es una empresa productora y comercializadora de productos informáticos por lo cual, el tiempo y el costo de cada producto deben ser minimizados, así como se debe garantizar su productividad y sostenibilidad. Por lo tanto, las herramientas que se empleen en los diferentes productos de la empresa han de ser desarrollados íntegramente por sus desarrolladores.

Existen aplicaciones disponibles y libres de pago que tienen como una de sus funcionalidades la extracción de parámetros a partir de imágenes digitales, con los cuales se pueden formar patrones que pueden ser utilizados por aplicaciones para entrenar redes neuronales artificiales, entre otras posibilidades, para después ser empleadas en la segmentación de imágenes, por ejemplo: los paquetes de software de código abierto como *ITK*, *ITK-SNAP*, *VXL*, *MITK*, *OpenCV*, *GRASS GIS*, *Fiji*, *Trainable Segmentation* e *IMMI*, entre otros. Estos, están destinados, entre otros objetivos, para la segmentación de imágenes. Muchos de ellos emplean métodos supervisados de reconocimiento de patrones y por lo tanto hacen uso de funcionalidades que han de permitir al especialista la selección y construcción de patrones a partir de su interacción con imágenes digitales. Los paquetes de software que aquí se mencionan, y otros más, fueron revisados buscando aquellos que permitieran manipular una imagen y extraer de ella los parámetros definidos por los desarrolladores de SOFTEL.

El análisis de las aplicaciones mencionadas en el párrafo anterior permitió detectar que:

1. Algunos productos encontrados tienen propietario y licencias de comercialización. Por lo tanto, no pueden ser usados con fines comerciales en las aplicaciones de la empresa.

2. Los que son libres, no permiten que se usen con fines comerciales o traerían complicaciones si se comercializan.
3. Ningún producto brinda la posibilidad de calcular todas las variables definidas por los especialistas del grupo de desarrollo.
4. Ningún producto permite, con pocos esfuerzos, lograr que exista una compatibilidad a la hora de la entrega de los patrones a NEUROLAB en archivos de texto.

Lo anterior exige crear una herramienta propia de SOFTEL, tal que permita calcular parámetros y construir vectores denominado patrones con los valores anteriormente calculados, seleccionados por los especialistas a partir de imágenes digitales. La imposibilidad de poder hacer uso de alguna aplicación que ya exista, conlleva al siguiente **problema a resolver**: ¿Cómo informatizar el proceso de selección, cálculo y extracción de patrones dentro de una imagen digital para entregárselos a NEUROLAB?

Para dar solución al problema, ha sido definido como **objeto de estudio** la distribución de los colores en determinadas regiones dentro de las imágenes digitales para ser utilizadas en el reconocimiento de patrones.

Actualmente, para buscar, encontrar y seleccionar los valores de los atributos en una base de datos que responden a varios individuos, se cuenta con muchas herramientas que permiten hacerlo y trabajar con ella de manera muy cómoda, llegando incluso a poder exportarla. Sin embargo, si los datos están enmascarados en la distribución de los colores de los píxeles de una imagen digital y se desea calcularlos y extraerlos, entonces es obligada la creación de una herramienta informática que automatice este proceso.

El **objetivo general** es desarrollar una herramienta que le permita al usuario construir patrones a partir de imágenes digitales para que puedan ser usados por NEUROLAB.

Se concibe como **campo de acción** los patrones dentro de una imagen digital que representan a una categoría determinada. La **idea a defender** es desarrollar un visor de imágenes tal que permita construir patrones colorimétricos de forma interactiva y depositarlos en archivos de texto para que sean utilizados por NEUROLAB.

CAPÍTULO 1. Fundamentación teórica

Los **aportes prácticos esperados** son:

1. Poder extraer patrones colorimétricos de imágenes y para ser usados por NEUROLAB con el objetivo de entrenar redes neuronales artificiales.

2. La RNA entrenada se podrá utilizar para identificar, de manera interactiva, las diferentes clases existentes dentro de las imágenes que procedan de la misma fuente original u otras con similares características.

Los **objetivos específicos** son:

1. Investigar y seleccionar las técnicas a utilizar en la extracción interactiva de patrones a partir de imágenes digitales.
2. Realizar el análisis y el diseño de una herramienta que permita extraer patrones en una imagen digital para NEUROLAB.

3. Realizar la implementación de la herramienta.

4. Validar las funcionalidades de la herramienta.

Durante el trabajo se utilizaron los siguientes **métodos científicos**:

Métodos teóricos

- Analítico-sintético: Es utilizado para el análisis de los métodos de cálculo de patrones, herramientas, tecnologías, lenguajes y la metodología de desarrollo seleccionadas.
- Modelación: Es utilizado en el diseño de la herramienta a implementar.

Métodos empíricos

- Análisis documental: Estudio de la bibliografía referente a las herramientas, metodología y tecnologías para dar solución a la problemática.

Estructura del documento

Para resolver el problema planteado, el presente documento se ha estructurado en tres capítulos:

Capítulo I: Fundamentación teórica

En este capítulo se realiza el enunciado de algunos conceptos necesarios para la correcta comprensión de los elementos que se exponen en este documento. Entre ellos, se plantean y

explican los métodos matemáticos empleados para el cálculo de las diferentes variables que pueden conformar un patrón, acordes con las exigencias de SIRNA.

Al finalizar se hace referencia a la metodología de desarrollo de *software* y herramientas utilizadas durante el desarrollo de la aplicación.

Capítulo II: Descripción de la solución

Dentro de este capítulo se hace alusión a todo lo referente a las características de la herramienta, donde se plasman los requisitos definidos por el cliente ya sean funcionales o no funcionales, se realiza la descripción de las historias de usuario con sus respectivas tareas de ingeniería, se explica cómo está estructurada la arquitectura de la aplicación que se implementó y se exponen los patrones de diseño que se utilizaron, incluyendo las iteraciones definidas con sus respectivos artefactos para la metodología en cuestión.

Capítulo III: Implementación y Pruebas

En el tercer y último capítulo se definen los aspectos relacionados con la implementación de la aplicación y las pruebas realizadas a la misma. Se expone una representación gráfica del diseño de la base de datos. Incluye ejemplos de código de las diferentes clases que explican el funcionamiento de los componentes y se concluye realizando pruebas para comprobar si la aplicación cumple sus objetivos, mostrando los resultados obtenidos.

CAPÍTULO 1. Fundamentación teórica.

1.1 Introducción

Tal y como sucede en otros ámbitos de la creación humana, en el desarrollo de *software* se tienen en cuenta las herramientas y tecnologías que han de usarse para la creación de los productos. Es por ello que, formando parte de la planificación para la elaboración del *software*, se requiere que el equipo de desarrollo defina que herramientas son necesarias y cuáles se adaptan adecuadamente a sus exigencias.

En este capítulo son expuestos los métodos utilizados en el proyecto para la manipulación y procesamiento de las imágenes digitales, así como para obtener los valores de las variables que se calculan a partir de los colores de los píxeles de una imagen digital. Además, se presentan los detalles para la creación de los patrones a partir de las variables calculadas y se hace una selección de las herramientas a utilizar durante el proceso de desarrollo de la solución que aquí se expone.

1.2 Obtención de patrones a partir de imágenes digitales.

A continuación se exponen todos los elementos que sustentan teóricamente el método de conformación de los patrones en la herramienta SIRNA, construidos a partir de parámetros calculados sobre imágenes digitales, que deben entregárseles a NEUROLAB en archivos de texto.

1.2.1 Imagen digital.

Una *imagen digital* es una representación numérica bidimensional construida a partir de una matriz binaria y de acuerdo a su resolución puede clasificarse en (1):

1. *imagen digital vectorial*, compuesta por formas geométricas simples, como pueden ser polígonos o segmentos de distinta topología (1).
2. *imagen digital matricial (o Mapa de Bits)*, compuesta por sub-matrices de la matriz binaria que reciben el nombre de *píxeles.* Cada *pixel* tiene un color asociado que responde a cualquiera de las siguientes combinaciones (1):
    2.1. RGB (Rojo, Verde y Azul)
    2.2. Matiz, Saturación y Luminosidad.

CAPÍTULO 1. Fundamentación teórica

Las *imágenes digitales matriciales* son utilizadas ampliamente en la solución de problemas prácticos como lo es en, la medicina, la teledetección, meteorologia, entre otros. Los colores en el formato RGB, asociados a los píxeles, son, también, los que más se emplean en su procesamiento.

A continuación se presentan tres figuras que ilustran la composición de una imagen digital de mapas de bits atendiendo a la combinación RGB asociada a cada pixel. Aparecen: la imagen original vista completamente (***Figura 1***), la vista ampliada de una sub-imagen extraída de la misma (***Figura 2***) y una representación matricial del conjunto de píxeles que componen dicha sub-imagen (***Figura 3***):

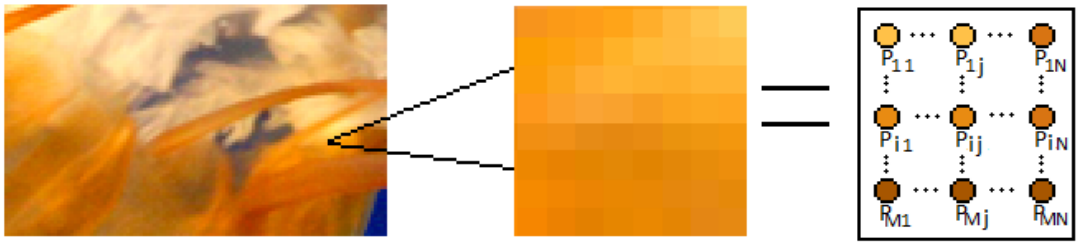


Figura 1 Figura 2 Figura 3

Figura 4. *Representacion de una submatriz de pixeles proveniente de una imagen.*

Donde:

- $\boldsymbol{N}$ es la cantidad de columnas de la sub-imagen.
- $\boldsymbol{M}$ es la cantidad de filas de la sub-imagen.
- $\boldsymbol{R_{ij}}$ es el valor del componente **rojo** del pixel $\boldsymbol{P_{ij}}$. (i = 1, 2, 3,…, $\boldsymbol{M}$ y j = 1, 2, 3,…, $\boldsymbol{N}$).
- $\boldsymbol{G_{ij}}$ es el valor del componente **verde** del pixel $\boldsymbol{P_{ij}}$. (i = 1, 2, 3,…, $\boldsymbol{M}$ y j = 1, 2, 3,…, $\boldsymbol{N}$).
- $\boldsymbol{B_{ij}}$ es el valor del componente **azul** del pixel $\boldsymbol{P_{ij}}$. (i = 1, 2, 3,…, $\boldsymbol{M}$ y j = 1, 2, 3,…, $\boldsymbol{N}$).
- $\boldsymbol{P_{ij}}$ es el pixel ubicado en la fila i columna j de la sub-imagen y está determinado por el vector ($\boldsymbol{R_{ij}}$, $\boldsymbol{G_{ij}}$, $\boldsymbol{B_{ij}}$). (i = 1, 2, 3,…, $\boldsymbol{M}$ y j = 1, 2, 3,…, $\boldsymbol{N}$)

1.2.2 Variables obtenidas a partir de los píxeles de una imagen.

Existen muchos parámetros que pueden ser obtenidos como resultado del procesamiento digital de imágenes y/o con el objetivo del análisis de las mismas. Sin embargo, para la implementación de SIRNA el grupo de desarrollo decidió, calcular solo algunos de poca complejidad, que pudieran

describir la distribución de los colores alrededor de un pixel cualquiera de una imagen digital y con ellos conformar los patrones que han de suministrarse a NEUROLAB.

Las variables en cuestión se deben calcular teniendo en cuenta un pixel cualquiera de la imagen y una vecindad alrededor del mismo. Estas son:

1. Media.
2. Desviación Estándar.
3. Asimetría.
4. Curtosis.
5. Mediana.
6. Moda.

A continuación se enumeran los métodos para calcular las variables antes mencionadas tomando como ejemplo de referencia la sub-imagen que aparece en la ***figura 3***.

Media Aritmética.

La **Media Aritmética** de un conjunto de N números X_1, X_2, X_3,…, X_N se denota por $\bar{X}$ y se define como: (2)

$$\bar{X} = \frac{1}{N}\sum_{j=1}^{N} X_j \qquad [1]$$

Desviación Estándar.

La **Desviación Estándar** de un conjunto de N números X_1, X_2, X_3,…, X_N, donde N es menor que 30, se denota por S y se define como: (2)

$$S = \sqrt{\frac{1}{N-1}\sum_{j=1}^{N}(X_j - \bar{X})^2} \qquad [2]$$

Asimetría.

La **Asimetría** es una medida del grado de asimetría de una distribución (3). Para un conjunto de N números X_1, X_2, X_3,…, X_N se denota por A y se define como:

$$A = \frac{\frac{1}{N}\sum_{j=1}^{N}(X_j - \bar{X})^3}{S^3} \quad [3]$$

Curtosis.

La **Curtosis** es el grado de esbeltez de una distribución, definida como una forma normalizada del cuarto momento central de una distribución (4). Para un conjunto de N números X_1, X_2, X_3,…, X_N se denota por C y se define como:

$$C = \frac{\frac{1}{N}\sum_{j=1}^{N}(X_j - \bar{X})^4}{S^4} \quad [4]$$

Mediana.

La **Mediana** de un conjunto de números dispuestos en orden de magnitud (es decir, en un ordenamiento) del centro o la media aritmética de los dos valores centrales (2).

Moda.

La **Moda** de un conjunto de números es el valor que aparece con mayor frecuencia. La moda puede no existir, o incluso si existe, puede no ser única (2).

1.2.3 Patrones y Clases

En la actualidad son generados grandes cantidades de datos, los cuales es necesario estudiar para extraer de ellos el conocimiento necesario. Desde la década de los años 60 se ha realizado el estudio y la investigación de las características y reglas contenidas en los datos y que responden a un mismo tipo de suceso o individuo. Esos atributos son definidos como “patrones” en los términos de la informática y el análisis de datos en general. En esos años comienza el desarrollo ascendente del “Reconocimiento de Patrones”. Esta rama de la ciencia abarca los métodos y la teoría acerca de la búsqueda e identificación de patrones que están presentes en datos provenientes de disímiles orígenes tales como bases de datos, imágenes digitales, señales digitales, etc. Los métodos pueden ser supervisados o no supervisados, atendiendo a si se usan o no patrones tomados como ejemplos, respectivamente. Una forma de empleo de estos métodos es en la clasificación para discriminar elementos dentro de un conjunto de datos y su forma más avanzada se presenta a través del uso de las Redes Neuronales Artificiales RNA (5).

CAPÍTULO 1. Fundamentación teórica

Según las exigencias de NEUROLAB, un patrón está determinado por un vector que contiene, al menos uno de los valores de alguna de las variables antes descritas para cada uno de los colores Rojo, Verde y Azul. O sea, tomando como ejemplo la sub-imagen que se ilustra en la ***figura 3***, si se desea que el patrón esté compuesto por los valores de las variables *media aritmética* y *desviación estándar*; entonces el vector correspondiente estará compuesto por seis elementos y tendrá la siguiente expresión:

$$\boldsymbol{Patrón} = (\bar{X}_R, \bar{X}_G, \bar{X}_B, S_R, S_G, S_B)$$

Considerando que,

1. $\bar{X}_k$ - Es la media aritmética correspondiente al componente k del color de todos los píxeles de la sub-imagen y se calcula atendiendo a la ecuación 1, como se muestra a continuación:

$$\bar{X}_k = \frac{1}{M * N} \sum_{i=1}^{M} \sum_{j=1}^{N} k_{ij} \qquad [5]$$

   k = {R: Rojo, G: Verde, B: Azul}

2. S_k - Es la desviación estándar correspondiente al componente k del color de todos los píxeles de la sub-imagen y se calcula atendiendo a la ecuación 2, como se muestra a continuación:

$$S_k = \sqrt{\frac{1}{M * N - 1} \sum_{i=1}^{M} \sum_{j=1}^{N} (k_{ij} - \bar{X}_k)^2} \qquad [6]$$

   k = {R: Rojo, G: Verde, B: Azul}

El patrón, de forma aislada, no expresa información alguna si este no se asocia con una cualidad determinada que represente a un conjunto de individuos. Por lo tanto, la misión final y principal del reconocimiento de patrones consiste en descubrir elementos identificativos en un conjunto de datos y asociarlos con diferentes categorías o clases, para así entender el fenómeno al cual responden. Por lo tanto, cada vez que se obtiene un patrón determinado resulta necesario darle un significado y es entonces que se impone asignarle una denominación. O sea, a cada patrón se le debe asignar una *clase*. Por todo esto, además de extraer los patrones, es imprescindible antes, definir el conjunto de clases que han de cubrir todo el universo del problema que se desea resolver.

Es así como se calculan las variables y después se construyen con ellas los patrones en la aplicación que se expone en el presente documento.

CAPÍTULO 1. Fundamentación teórica

1.3 Red Neuronal Artificial.

Una red neuronal artificial es una representación matemática inspirada en el funcionamiento de las neuronas distribuidas en el sistema nervioso de los animales y está conformada por una estructura que puede ser ajustada para hacer corresponder un conjunto de datos dados a una característica determinada o encontrar relaciones entre ellos. El modelo es ajustado, o entrenado, empleando para ello una colección de datos como entrada, que provienen de una fuente dada, normalmente conocidos como conjunto de entrenamiento. Luego de ser entrenada, la red neuronal está lista para que funcione y ser usada en la clasificación, estimación, predicción o simulación a partir de nuevos datos provenientes de fuentes similares (35).

Existe gran variedad de modelos de redes neuronales artificiales, algunos que incluyen métodos no supervisados de entrenamiento y otros que hacen uso de los métodos supervisados. A este último grupo estará orientada la aplicación que se expone en este documento, y dentro de este, específicamente, al que responde al nombre de "Perceptrón Multicapa" (también conocido como MLP o *Multilayer Perceptron* en inglés), que se define como un conjunto de capas de neuronas donde existe una capa de entrada, una capa de salida y una o varias capas intermedias u ocultas, todas conectadas entre sí. A continuación se presenta una ilustración de la arquitectura de este tipo de red neuronal (36):

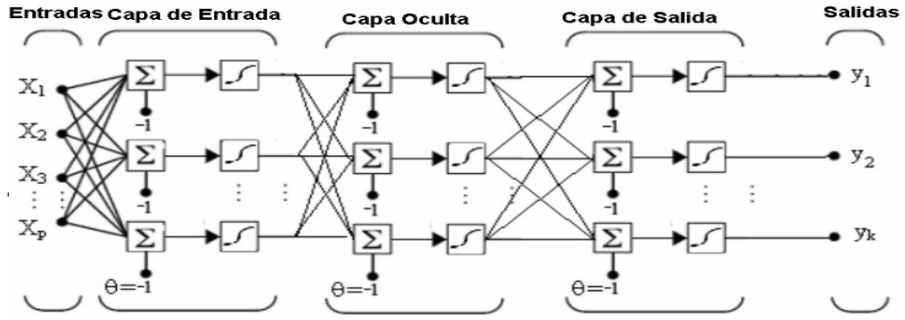


Figura 5. *Arquitectura del Perceptrón Multicapa.*

Donde los valores X_1, X_2, X_3,…, X_p son los valores de entrada (o patrón) y los valores y_1, y_2,…, y_k corresponden a los valores de salida (o clase) (37).

O sea, el funcionamiento de una red neuronal artificial de este tipo se puede interpretar como el de una función matemática a la que se le suministran, como argumentos, los valores que se muestran en la capa de entrada y ella devuelve los valores que se muestran en la capa de salida (***Figura 5***). Para SIRNA, cada vector conformado por los valores de la capa de entrada recibe el nombre de Patrón, y cada vector de la capa de salida recibe el nombre de Clase. Además, para el entrenamiento de este modelo de red neuronal es necesario asociarle a cada patrón la clase que este representa y suministrarle después esta pareja de datos (Patrón, Clase) para que la red "aprenda" a clasificar durante su uso.

1.4 Tecnologías de desarrollo de *software*

La empresa SOFTEL tiene implantado una tecnología de desarrollo de *software* la cual, le permite realizar los proyectos de forma organizada. Esta tecnología está basada en los estándares que se utilizan a nivel mundial; pero atendiendo a los años que han transcurrido produciendo productos informáticos y la experiencia acumulada, ha incorporado y modificado mecanismos y formatos que son los que regulan el desempeño de todos los trabajadores que desarrollan las aplicaciones. La metodología básica usada en SOFTEL es la denominada RUP con algunas modificaciones propias del funcionamiento de la empresa, aunque en algunos casos se emplea la metodología XP.

1.4.1 Metodologías de desarrollo de *software*

Una metodología de desarrollo de *software* se define como: *"un conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a realizar nuevas aplicaciones. La metodología indica cómo hay que obtener los distintos productos parciales y finales"* (6).

Para la selección de la metodología de desarrollo a utilizar fueron consideradas tres de las más utilizadas; estas son el Proceso Unificado de Desarrollo (RUP), la Programación Extrema (XP) y Scrum (S).

Procesamiento Unificado de Desarrollo.

Rational Unified Process (RUP por sus siglas en inglés) es un proceso formal, provee un acercamiento disciplinado para asignar tareas y responsabilidades dentro de una organización de desarrollo. Su objetivo es asegurar la producción de *software* de alta calidad que satisfaga los requerimientos de los usuarios finales (respetando el cronograma y el presupuesto). Fue desarrollado por *Rational Software*, y está integrado con toda la *suite Rational* de herramientas. Puede ser adaptado y extendido para satisfacer las necesidades de la organización que lo adopte. Es guiado por Casos de Uso y centrado en la arquitectura, utilizando UML como lenguaje de modelado. Define cuatro fases esenciales (Inicio, Elaboración, Construcción y Transición) y nueve flujos de trabajos; seis de Ingeniería (Modelado del Negocio, Requerimientos, Análisis y Diseño, Implementación, Prueba y Despliegue) y tres de apoyo. (7)

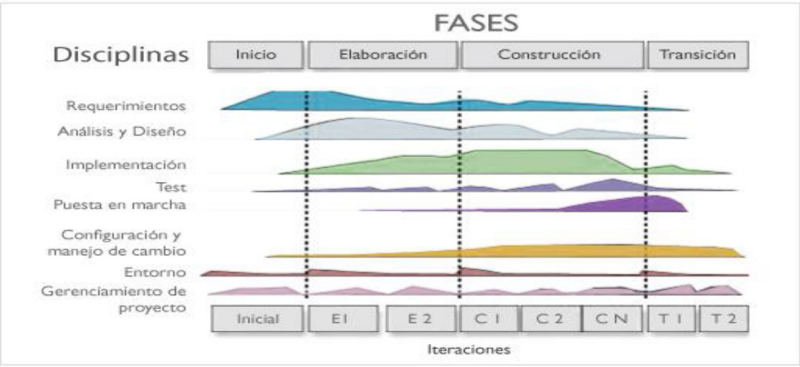


Figura 6. *Ciclo de Vida de RUP.*

Características de RUP:

1. Establece un desarrollo iterativo.
2. Permite la administración de requisitos.
3. Hace uso de una arquitectura basada en componentes.
4. Permite llevar un control de cambios.
5. Permite realizar un modelado visual del *software*.
6. Verifica la calidad del *software*.

Programación Extrema.

La Programación Extrema es una metodología ágil concebida e implementada para dirigir las necesidades específicas del desarrollo de *software* realizado por equipos pequeños. Esta le da poder a los desarrolladores para responder con confianza a los requerimientos cambiantes del consumidor final. Se caracteriza además por fomentar la comunicación desarrollador-cliente desde el primer día. Es considerada ligera, flexible, predecible, de bajo riesgo, y no por ello menos científica.

Otras ventajas que no pueden pasar por alto son los pocos requerimientos de documentación y planificación siendo las historias de usuarios los principales artefactos que se generan, así como la exigencia de tener siempre el cliente disponible para el desarrollo, implicando una mejor correspondencia entre el producto y la necesidad del negocio. (8)

Scrum.

Scrum es un marco de trabajo para la gestión y desarrollo de *software* basada en un proceso iterativo e incremental utilizado comúnmente en entornos basados en el desarrollo ágil de *software*. Está especialmente indicada para proyectos con un rápido cambio de requisitos. Sus principales características se pueden resumir en dos:

1. El desarrollo de *software* se realiza mediante iteraciones, denominadas *"sprints"*, con una duración de 1 a 4 semanas. El resultado de cada "sprint" es un incremento ejecutable que se muestra al cliente.
2. Reuniones a lo largo del proyecto. Éstas son las verdaderas protagonistas, especialmente una reunión diaria de corta duración del equipo de desarrollo para realizar la coordinación y la integración (8).

Análisis de la selección de la metodología de desarrollo.

Teniendo en cuenta el análisis de las tres metodologías más usadas, las necesidades expresas de obtener un producto en el menor tiempo posible, la simplicidad del proceso de desarrollo, la centralización de los recursos humanos como eslabón fundamental de la cadena, incluyendo al cliente como parte del proceso de desarrollo, la escasa documentación, entre otros, se selecciona XP como

metodología de desarrollo, ya que está destinada especialmente para pequeños equipos de trabajo con requisitos imprecisos o cambiantes, donde existe un alto riesgo técnico. Además, se orienta a una entrega rápida de resultados y posee alta flexibilidad. En definitiva se puede afirmar que, aunque en la empresa SOFTEL se emplea generalmente la metodología RUP y dadas las dimensiones del proyecto, este podrá implementarse exitosamente con el uso de la metodología XP.

1.4.2 Lenguajes de programación

Un lenguaje de programación es "*una técnica estándar de comunicación que permite expresar las instrucciones que han de ser ejecutadas en una computadora"* (9). Pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana (10).

Los primeros lenguajes de programación de alto nivel se diseñaron durante los años 50. En la actualidad entre ellos se encuentran Turbo Pascal, Delphi, C, C++, C#, todos con sus características ligadas a una consecuente evolución en los procedimientos, y paradigmas de programación: imperativo, declarativo, funcional, orientado a objetos, etc.

Para el desarrollo de aplicaciones web existen otros como PHP, Python y especialmente Java (11).

C++

C++ es un lenguaje de programación de propósito general basado en el lenguaje de programación C. Además de los recursos de C, C++ proporciona clases, funciones en línea, sobrecarga de nombres de funciones, tipos constantes, referencias, operadores para el manejo del almacenamiento disponible, verificación de argumentos de funciones y conversiones de tipos (12). El lenguaje C++ se comenzó a desarrollar en 1980, por Bjarne Stroustrup; en la actualidad C++ es un lenguaje versátil y potente, se puede utilizar en la construcción de todo tipo de aplicaciones aunque su mayor aplicación radica en las aplicaciones de escritorio, sistemas operativos y servicios.

PHP

PHP es un lenguaje interpretado en el lado del servidor que se caracteriza por su potencia, versatilidad, robustez y modularidad. Los programas escritos en PHP son embebidos directamente en el código HTML y ejecutados por el servidor web a través de un intérprete antes de transferir al cliente que lo ha solicitado un resultado en forma de código HTML puro. Al ser un lenguaje que sigue las corrientes *opensource*, son totalmente accesibles de forma gratuita en la red. Es un lenguaje multiplataforma, que puede funcionar sobre la mayoría de los servidores web y brinda soporte a más de 20 tipos de base de datos (13).

Java

CAPÍTULO 1. Fundamentación teórica

Java es un lenguaje de programación de alto nivel desarrollado por *Sun MicroSystem* a principios de la década del 90. Su sintaxis es muy parecida a la de C y C++ ya que su desarrollo fue inspirado en los mismos, siempre buscando eliminar errores que suelen inducirse en lo que respecta a la manipulación de memoria y punteros (14).

Los programas en Java generalmente son compilados a un lenguaje intermedio llamado *bytecode*, que luego es interpretado por una máquina virtual (VM por sus siglas en inglés). Esta última sirve como una plataforma de abstracción entre el hardware y el lenguaje. También es posible la competición a código de máquina, brindando mayor eficiencia en la ejecución del programa pero limita su característica multiplataforma (15).

Entre las características principales de este lenguaje se encuentran las siguientes:

- **Simple**: Elimina la complejidad de los lenguajes como C y C++ dando paso al contexto de los lenguajes modernos orientados a objetos.
- **Orientado a objetos**: Soporta las características esenciales del paradigma de la programación orientada a objetos: encapsulamiento, herencia y polimorfismo.
- **Robusto**: Elimina el uso de apuntadores para referenciar áreas de memoria, además, el desarrollador no necesita liberar la memoria que la aplicación ya no usa. También requiere la declaración explícita tanto de los tipos de datos como de los métodos.
- **Multiplataforma**: El mismo código Java que funciona en un sistema operativo, funciona en cualquier otro que tenga instalada la máquina virtual de Java.
- **Multitareas**: Permite la ejecución concurrente de varios procesos ligeros o hilos de ejecución.

La versatilidad y eficiencia de la tecnología Java, la portabilidad de su plataforma y la seguridad que aporta, la han convertido en la tecnología ideal para el desarrollo de aplicaciones distribuidas.

Python

Python es un lenguaje de programación creado por Guido van Rossum a principios de los años. Es un lenguaje similar a Perl, pero con una sintaxis muy limpia y que favorece un código legible. Se trata de un lenguaje interpretado o de script, multiplataforma y orientado a objetos (16).

En los últimos años el lenguaje se ha hecho muy popular, debido a:

1. La cantidad de librerías que contiene, tipos de datos y funciones incorporadas en el propio lenguaje, que ayudan a realizar muchas tareas habituales sin necesidad de tener que programarlas desde cero.
2. La sencillez y velocidad con la que se crean los programas.
3. La cantidad de plataformas para las que permite desarrollar, como Unix, Windows, OS/2, Mac.

Python es gratuito, incluso, para propósitos empresariales.

Análisis de la selección del lenguaje de programación

Para la selección del lenguaje de programación a emplear en el desarrollo de la solución se tuvo en cuenta la aplicación NEUROLAB y su comunicación con la herramienta SIRNA. NEUROLAB fue desarrollada empleando el *framework* GRAILS y resulta conocido que las herramientas desarrollados en Java pueden ser integrados a Grails de forma natural gracias a que Groovy es completamente compatible con Java, permitiendo utilizar cualquier clase, API o biblioteca de Java. Por todo ello se seleccionó el lenguaje de programación Java.

1.4.3 Entorno de Desarrollo Integrado

Un Entorno de Desarrollo Integrado (*Integrated Development Enviroment o IDE* en inglés) es un ambiente de programación que ha sido generado como una aplicación, típicamente consistente en un editor de código, un compilador, un depurador y una interfaz de usuario (20).

NetBeans

NetBeans es un entorno de desarrollo gratuito y de código abierto. Permite el uso de un amplio rango de tecnologías de desarrollo tanto para escritorio, como aplicaciones Web, o para dispositivos móviles. Da soporte a las siguientes tecnologías, entre otras: **Java**, **PHP**, **Groovy**, **C/C++**, **HTML5**,… Además puede instalarse en varios sistemas operativos como son Windows, Linux y Mac OS, tiene como características las siguientes (41):

1. **Asistentes** para la creación y configuración de distintos proyectos, incluida la elección de algunos frameworks.
2. Buen **editor de código**, **multilenguaje**, con el habitual coloreado y sugerencias de código, acceso a clases accionando una en el código, control de versiones, localización de ubicación de la clase actual, comprobaciones sintácticas y semánticas, plantillas de código, herramientas de refactorización.
3. Simplifica la **gestión de grandes proyectos** con el uso de diferentes vistas, asistentes de ayuda, y estructurando la visualización de manera ordenada, lo que ayuda en el trabajo diario.
4. Herramientas para **depurado de errores**: el debugger que incluye el IDE es bastante útil para encontrar dónde pueden fallar algunas de las instrucciones declaradas. Definición de puntos de ruptura en la línea de código que nos interese, monitorizar en tiempo real los valores de propiedades y variables.
5. **Optimización de código**: por su parte el **Profiler** nos ayuda a optimizar las aplicaciones e intentar hacer que se ejecuten más rápido y con el mínimo uso de memoria.

6. **Acceso a base de datos**: desde el propio NetBeans es posible conectarse a distintos sistemas gestores de bases de datos, como pueden ser Oracle, MySQL y demás, y ver las tablas, realizar consultas y modificaciones, todo ello integrado en el propio IDE.
7. Se integra con diversos **servidores de aplicaciones**, de tal manera que es posible gestionarlos desde el propio IDE: inicio, parada, arranque en modo debug, despliegues. Entre otros se puede usar Apache Tomcat, GlassFish, JBoss, WebLogic, Sailfin, Sun Java System Application Server,
8. Es fácilmente **extensible** a través de **plugins**. (41)

Eclipse

Eclipse es una plataforma de desarrollo, diseñada para ser extendida de forma indefinida a través de *plug-ins*. Fue concebida desde sus orígenes para convertirse en una plataforma de integración de herramientas de desarrollo. No tiene definido un lenguaje específico, sino que es un IDE genérico, aunque goza de mucha popularidad entre la comunidad de desarrolladores del lenguaje **Java** usando el *plug-in* **JDT** que viene incluido en la distribución estándar del IDE (42).

1. Proporciona herramientas para la gestión de espacios de trabajo, escribir, desplegar, ejecutar y depurar aplicaciones.
2. **Perspectivas, editores y vistas**: en Eclipse el concepto de trabajo está basado en las perspectivas, que no es otra cosa que una preconfiguración de ventanas y editores, relacionadas entre sí, y que permiten trabajar en un determinado entorno de trabajo de forma óptima.
3. **Gestión de proyectos**: el desarrollo sobre Eclipse se basa en los proyectos, que son el conjunto de recursos relacionados entre sí, como puede ser el código fuente, documentación, ficheros configuración, árbol de directorios. El IDE proporciona asistentes y ayudas para la creación de proyectos.
4. **Depurador de código**: se incluye un potente depurador, de uso fácil e intuitivo, y visualmente nos ayuda a mejorar el código. Para ello sólo es necesario ejecutar el programa en modo depuración (con un simple botón). Tiene una perspectiva específica para la depuración de código, la **perspectiva depuración**, donde se muestra de forma ordenada toda la información necesaria para realizar dicha tarea.
5. **Extensa colección de *plug-ins***: están disponibles en una gran cantidad, unos publicados por Eclipse, otros por terceros. Al haber sido un estándar de facto durante tanto tiempo (no el único estándar, pero sí uno de ellos), la colección disponible es muy grande. Los hay gratuitos, de pago, bajo distintas licencias, pero casi para cualquier necesidad existe el *plug-in* adecuado.
6. El **coloreado de código** en el editor es una característica muy interesante, realizando para ello el reconocimiento sintáctico de todas aquellas palabras que son reservadas en el lenguaje Java.
7. Permite completar el código automáticamente (***code completion***), con sugerencias dependientes del contexto, lo cual permite escribir el código más rápidamente.

8. Se puede configurar el **formateo de código**, la forma de escribir los comentarios, incluyendo **comentarios** para la posterior creación del **Javadoc**. Es posible generar los **esqueletos de clase** automáticamente, la generación de métodos getters y setters de manera automática, y una gran cantidad de funcionalidades, que al día de hoy parecen típicos, pero son muy útiles. (42)

Qt

Qt es una biblioteca de software desarrollada por Nokia para crear interfaces gráficas de usuario. Aplicaciones como: Google Earth, Skype, Adobe Photoshop Album y VirtualBox entre muchas otras, hacen uso de ésta.

Para los desarrolladores, Nokia ofrece ***Qt Creator*, un entorno de desarrollo (IDE) multiplataforma muy completo.** (43)

1. Posee un avanzado editor de código C++.
2. Además soporta los lenguajes: C#/.NET Languages (Mono), Python: PyQt y PySide, Ada, Pascal, Perl, PHP y Ruby.
3. Posee también una GUI integrada y diseñador de formularios.
4. Ayuda sensible al contexto integrado.
5. Depurador visual.
6. Resaltado y auto-completado de código.
7. Soporte para refactorización de código.
8. Distribuido bajo tres tipos de licencias: Qt Commercial Developer License, Qt GNU LGPL v. 2.1, Qt GNU GPL v. 3.0 y disponible para las plataformas: Linux, Mac OSX; Windows, Windows CE, Symbian y Maemo.

Análisis de la selección del entorno de desarrollo integrado

Para la selección del IDE a usar para el desarrollo de la herramienta en cuestión, fueron analizadas las características de cada uno de los IDE anteriormente mencionados comparándolos y verificando si cumplían con los requisitos del cliente y las políticas de la empresa, por lo que se llegó a la conclusión de usar NetBeans ya que fue definido por la empresa debido a que con él se garantiza la ejecución rápida de aplicaciones que se están desarrollando y probarlas con pocos recursos de memoria

1.4.4 Servidores de base de datos

Los servidores de base de datos, también conocidos como DBMS (acrónimo en inglés de *Database Management Systems*), son sistemas que permiten organizar datos en una o más tablas relacionadas (17). Los servidores de base de datos se utilizan en una amplia variedad de aplicaciones. Prácticamente cualquier aplicación que necesite almacenamiento, acceso y análisis de datos estructurados hace uso de algún tipo de DBMS.

MySQL

MySQL es un sistema de gestión de bases de datos relacional, multi-hilos y multi-usuario. MySQL AB, desarrolla MySQL como *software* libre en un esquema de licenciamiento dual (18).

Por un lado se ofrece bajo la GNU-GPL para cualquier uso compatible con esta licencia, pero para aquellas empresas que quieran incorporarlo en productos privativos deben comprar a la empresa una licencia específica que les permita este uso. Está desarrollado en su mayor parte sobre ANSI C.

Al contrario de proyectos como Apache, donde el *software* es desarrollado por una comunidad pública y los derechos de autor del código están en poder del autor individual, MySQL es patrocinado por una empresa privada, que posee el *copyright* de la mayor parte del código. Esto es lo que posibilita el esquema de licenciamiento anteriormente mencionado. Además de la venta de licencias privativas, la compañía ofrece soporte y servicios.

PostgresSQL

Es un sistema de gestión de bases de datos objeto-relacional (ORDBMS). Basado en el proyecto POSTGRES, de la Universidad de Berkeley. El director de este proyecto es el profesor Michael Stonebraker, y fue patrocinado por *Defense Advanced Research Projects Agency* (DARPA), el *Army Research Office* (ARO), el National Science Foundation (NSF), y ESL, Inc (19).

Fue el pionero en muchos de los conceptos existentes en el sistema de objeto-relacional actual, incluido, más tarde en otros sistemas de gestión comerciales. PostgresSQL es un sistema objeto-relacional, ya que incluye características de la orientación a objetos, como puede ser la herencia, tipos de datos, funciones, restricciones, disparadores, reglas e integridad transaccional.

Análisis de la selección del servidor de bases de datos

Dado que la aplicación NEUROLAB está implementada actualmente para interactuar con su base de datos almacenada en un servidor MySQL, SIRNA es una herramienta de NEUROLAB y han de intercambiar datos de la forma más natural posible y, además, este es el servidor de base de datos que se utiliza en la empresa SOFTEL en el desarrollo de todas sus aplicaciones.

1.5 Conclusiones

Después de exponer los objetivos del presente trabajo, los elementos teóricos que para su posterior entendimiento se deben tener en cuenta y de haber enunciado los aspectos de la tecnología de desarrollo de software de SOFTEL para sus aplicaciones y en particular el desarrollo de la herramienta SIRNA, se puede resumir el objetivo fundamental de esta herramienta de la siguiente forma:

CAPÍTULO 1. Fundamentación teórica

"SIRNA ha de permitir que el usuario visualice una imagen, escoja los parámetros que desea medir sobre ella y seleccione en ella (con ayuda del mouse) diferentes regiones para construir los patrones correspondientes y entregarlos a NEUROLAB".

Una vez definida esta etapa, las condiciones están creadas para realizar la Descripción de la solución.

CAPÍTULO 2. Descripción de la solución.

2.1 Introducción

Una vez analizadas las soluciones similares y elegidas las herramientas y la metodología a utilizar, es necesario plantear una solución al problema. Para el desarrollo de la solución se siguieron los pasos definidos por la metodología seleccionada que guían el proceso de desarrollo de *software*. Sin embargo, y utilizando su capacidad de adaptación, el proceso fue adaptado a las condiciones especiales de trabajo en las que se desarrolla la herramienta. En el presente capítulo se identifican los requisitos funcionales y no funcionales del *software*, se describen las fases por las que transcurre el desarrollo de la herramienta haciendo referencia a todo lo concerniente a ellas, así como una descripción de cada uno de los artefactos generados.

Uno de los pasos fundamentales en la fase de exploración de la metodología XP es definir una metáfora del sistema. En XP no se enfatiza la definición temprana de una arquitectura estable para el sistema. Dicha arquitectura se asume evolutiva y los posibles inconvenientes que se generarían por no contar con ella explícitamente en el comienzo del proyecto se solventan con la existencia de una metáfora. El sistema es definido mediante una metáfora o un conjunto de metáforas compartidas por el cliente y el equipo de desarrollo (26).

Una metáfora es una historia compartida que describe cómo debería funcionar el sistema. La práctica de la metáfora consiste en formar un conjunto de nombres que actúen como vocabulario para hablar sobre el dominio del problema. Este conjunto de nombres ayuda a la nomenclatura de clases y métodos del sistema. Cuando el sistema es pequeño la metáfora es extremadamente sencilla (27).

Teniendo en cuenta los parámetros anteriormente expuestos, la metáfora del sistema es:

“una herramienta donde los especialistas que la han de usarla puedan realizar la extracción interactiva de patrones a partir de imágenes digitales, y entregárselos al sistema NEUROLAB en archivos de texto como fuente de información para que este garantice el diseño y entrenamiento de redes neuronales de una forma más sencilla, rápida y eficaz”.

2.2 Descripción de la herramienta propuesta

Una vez expuestos los diferentes aspectos teóricos sobre los que ha de sostenerse SIRNA y elegidas las herramientas y metodología a utilizar, se está en condiciones de plantear los objetivos del presente trabajo.

La herramienta tiene como principal objetivo:

CAPÍTULO 2. Descripción de la Solución

Garantizar la extracción interactiva de patrones para entregárselos a NEUROLAB en archivos de texto, usando como fuente de información las imágenes digitales.

La herramienta debe permitir a los especialistas encargados de realizar la extracción de patrones:

1. Crear nuevos proyectos y poder garantizar la gestión de imágenes y clases dentro del proyecto en cuestión (Añadir, Modificar, Eliminar y Mostrar), pudiendo seleccionar alguna de las opciones habilitadas en el momento que se está añadiendo o modificando algún proyecto.
2. Para cada proyecto debe poderse crear configuraciones, definiendo las variables que se deseen calcular (Media, Moda, Mediana, Asimetría, Curtosis, Desviación Estándar) y el tamaño de la vecindad.
3. Luego de existir un proyecto creado, con sus imágenes y clases incluidas y se seleccione la configuración a usar, debe poderse seleccionar una de las imágenes y una de las clases. A continuación, se debe poder comenzar la extracción de los patrones para la imagen y clase determinada.
4. Los patrones podrán ser gestionado añadiéndolos a la base de datos, de la que más tarde estos patrones van a ser exportados en archivos de texto para NEUROLAB para usarlos en el entrenamiento de redes neuronales. También podrán ser mostrados en forma de tabla, donde podrán ser eliminados a voluntad del usuario.
5. Al finalizar la extracción de los patrones y haber entrenado una red neuronal desde NEUROLAB, la herramienta incluye una opción en la cual será posible realizar pruebas con la red neuronal previamente entrenada para verificar si los patrones que fueron extraídos representan verdaderamente a cada clase presente en las imágenes que estarán siendo procesadas.

2.2.1 Conexión con NEUROLAB.

La aplicación que aquí se expone ha sido concebida para ampliar las prestaciones de NEUROLAB y brindarles a los usuarios la posibilidad de construir los patrones de entrenamiento a partir de la información visual que contienen las imágenes digitales. Por lo tanto, SIRNA podrá garantiza la entrega de los mismos en formato de texto a NEUROLAB, pues así responderá directamente a los deseos y necesidades de los especialistas que construyan redes neuronales artificiales que exploten sus funcionalidades.

2.2.2 Descripción del proceso de extracción de patrones.

La extracción de patrones a partir de imágenes digitales es la misión fundamental de la aplicación que aquí se expone, pues permite transformar la información visual contenida en ellas en valores numéricos

que han de identificar a cada suceso categorizado previamente por el usuario. O sea, permite construir cada patrón y asignarle a él la clase correspondiente para ser almacenados después, como se muestra en la siguiente figura.

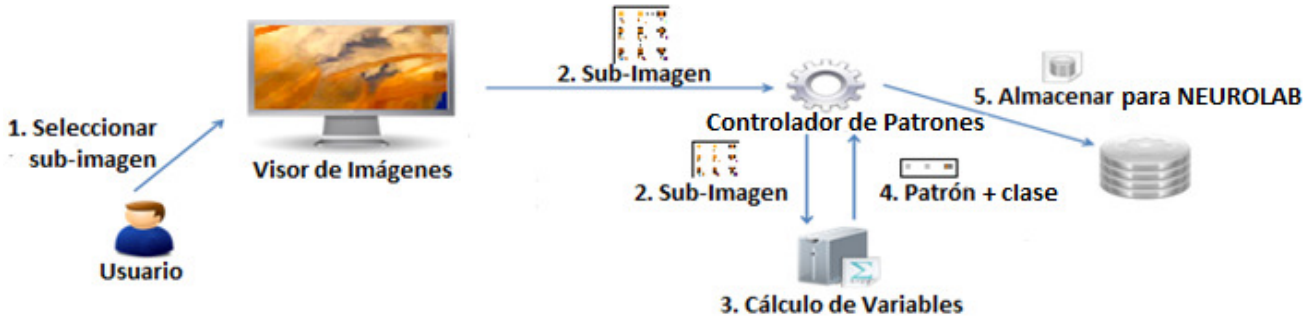


Figura 8. *Esquema general de funcionamiento de SIRNA para crear un patrón.*

De forma general, el proceso se divide en los pasos que se describen a continuación:

1. Definir las clases del proyecto: Se debe categorizar el problema asignándole una clase (o denominación) a cada uno de los sucesos posibles que pueden estar presentes en las imágenes.
2. Definir el tamaño de la vecindad: Aquí se debe especificar un valor entero positivo entre 1 y 10. Este valor define una sub-imagen cuadrada de dimensión V*2+1, donde V es el tamaño de la vecindad que se enuncia en este paso.
3. Definir el conjunto de variables: Aquí se deben especificar las variables que se desean calcular sobre la sub-imagen definida por la vecindad descrita en el paso anterior.
4. Visualizar una imagen digital: El usuario debe seleccionar y poder observar una imagen de las que estén relacionadas con el proyecto.
5. Seleccionar una clase: Debe seleccionar una de las clases del proyecto.
6. Seleccionar un punto dentro de la imagen: Con la ayuda del *mouse* se debe hacer clic sobre el punto deseado en la imagen. A continuación:
   6.1. Se extrae la sub-imagen centrada en el pixel que se corresponde con el punto seleccionado en la imagen original, cuyas dimensiones están dadas por la vecindad de tamaño V que se explicó en el paso 2.
   6.2. Se calculan las variables seleccionadas.
   6.3. Conformar un vector con los valores obtenidos. Este vector constituye así un patrón para NEUROLAB.

6.4. Se asocia el vector conformado (o patrón) con la clase seleccionada en el paso 5 y guarda estos datos en la tabla correspondiente, para que más tarde pueda ser entregado a NEUROLAB.

El paso 6 se puede ejecutar la cantidad de veces que sea necesaria sobre la misma imagen para extraer de ella todos los patrones que sean observados y sean depositados en la base de datos. Todos estarán asociados a la misma clase seleccionada en el paso 5. Si se desea especificar otra clase, es necesario que se ejecute nuevamente todo el proceso desde el paso 5; pero si se desea hacer la selección sobre otra imagen, entonces se debe regresar al paso 4 y después continuar con el paso 6.

2.2.3 Descripción del uso de la red neuronal entrenada

Una vez que la red neuronal artificial ha sido entrenada en NEUROLAB, SIRNA permite que la misma sea empleada para realizar pruebas sobre las mismas imágenes originales u otras que provengan de la misma fuente. Al hacer pruebas, el usuario puede comprobar de forma interactiva la calidad de la red neuronal entrenada y su efectividad al ser empleada para clasificar los píxeles de una imagen. Para ello, se debe cargar una imagen y sobre ella hacer clic en puntos (píxeles) ubicados en determinadas coordenadas de la misma. A continuación, el programa procede a conformar un vector, parecido al que se describe en el paso 6.3 y, después, se entrega a la red neuronal entrenada para ejecutarla y obtener de ella la salida correspondiente. La salida de la red identifica la clase a la que pertenece el pixel sobre el cual se hizo clic. O sea, el uso de la red neuronal, una vez entrenada, se reduce tomar un pixel, calcular las variables indicadas sobre la sub-imagen que define una vecindad del mismo, conformar con ellas un vector, entregárselos a la red neuronal y ejecutarla con estos datos de entrada. Luego, se toma el vector devuelto por la red y se determina con él la clase correspondiente. Esto ha de indicar la clase a la que pertenece el pixel seleccionado en la imagen.

2.3 Patrón de Arquitectura

Un patrón de arquitectura de *software* no es más que un esquema genérico probado para solucionar un problema específico recurrente que surge en un cierto contexto. (28) Este esquema se especifica describiendo los componentes, con sus responsabilidades, relaciones, y las formas en que colaboran.

CAPÍTULO 2. Descripción de la Solución

Patrón Modelo-Vista-Controlador

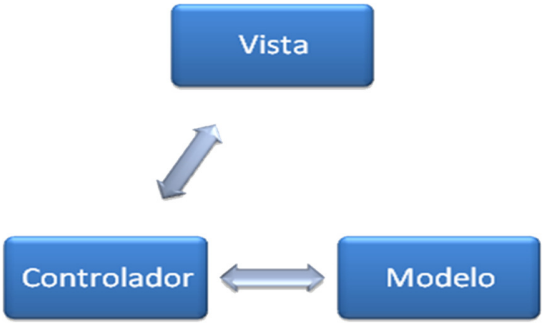


Figura 9. *Modelo-Vista-Controlador (MVC). (40)*

El patrón de arquitectura de *software* empleado para el desarrollo de la aplicación es la arquitectura MVC. Esta se basa en separar los datos de la aplicación, la interfaz de usuario, y la lógica del negocio en tres capas diferentes, es decir, que agrupa el código según su función.

1. Modelo: Esta capa define el comportamiento, los datos de la aplicación y la lógica del negocio. La base de datos pertenece a esta capa. Las clases que pertenecen a esta son:
   1.1. Clase: Almacena los datos de cada clase asociada a cada proyecto.
   1.2. Configuración: Almacena los datos de cada configuración para cada proyecto.
   1.3. Imagen: Almacena los datos de cada imagen para cada proyecto.
   1.4. Patrón: Almacena los datos de cada patrón para cada configuración.
   1.5. Proyecto: Almacena los datos de cada proyecto.
   1.6. Prueba: Almacena los datos para cada prueba de cada configuración.
2. Vista: La vista muestra la información que emplean los usuarios para interactuar con la aplicación. Las clases que pertenecen a esta capa son:
   2.1. frmMainSiRna: Es la clase principal del sistema y la que le permite al usuario interactuar con los datos del mismo.
   2.2. frmAddClass: Es la clase que le permite al usuario especificar las diferentes clases asociadas a un proyecto.
3. Controlador: La capa controlador es la que descifra las acciones realizadas por los usuarios, realiza las llamadas al modelo para obtener la información y se las envía a la vista para que las muestre a los usuarios. O sea, es la que gestiona los datos del sistema. Las clases que la componen son:
   3.1. Clases: Gestiona todos los datos del modelo Clase.
   3.2. Configuraciones: Gestiona todos los datos del modelo Configuración.
   3.3. Imágenes: Gestiona todos los datos del modelo Imagen.

3.4. Patrones: Gestiona todos los datos del modelo Patrón.

3.5. Proyectos: Gestiona todos los datos del modelo Proyecto.

3.6. Pruebas: Gestiona todos los datos del modelo Prueba.

La vista y el controlador dependen del modelo, el cual no depende de las otras clases. Esta separación permite construir y probar el modelo independientemente de la representación visual. (29)

2.4 Patrones de Diseño

Los patrones de diseño son soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos. (30) Son lenguajes de programación de alto nivel. Estos patrones hacen más fácil reutilizar con éxito los diseños y arquitecturas, ayudan a elegir entre diseños alternativos, hacen a un sistema reutilizable y evitan alternativas que comprometen a la reutilización. Además facilitan el aprendizaje al programador inexperto y ayudan a reutilizar código, facilitando la decisión "herencia o composición".

Para el diseño de la aplicación fueron analizados patrones de gran utilidad que proponen una forma de reutilizar la experiencia de los desarrolladores, para ello clasifican y describen las formas de solucionar los problemas que ocurren de forma frecuente durante el desarrollo.

2.4.1 Patrones GRASP

Los patrones GRASP, acrónimo de *General Responsibility Assignment Software Patterns* en español Patrones Generales de *Software* para Asignación de Responsabilidades, describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades. Constituyen un apoyo para la enseñanza que ayuda a entender el diseño de objeto esencial y aplica el razonamiento para el diseño de una forma sistemática, racional y explicable. (31)

Experto: La responsabilidad de realizar una labor es de la clase que tiene o puede tener los datos involucrados (atributos). Cada clase, contiene toda la información necesaria para realizar la labor que tiene encomendada. (32) Este patrón se evidencia en cada clase Modelo. Cada una contiene la información referente a la entidad que representa y es responsable de realizar la labor que tiene encomendada.

Creador: Propone que una instancia de un objeto tiene que ser creada por el objeto que tiene la información para ello. Se asigna la responsabilidad de que una clase B cree un objeto de la clase A, solamente cuando: B contiene a A, B es una agregación (o composición) de A, B almacena a A, B tiene los datos de inicialización de A (datos que requiere su constructor), B usa a A. (32)

El uso de este patrón se evidencia en la clase Controladora de los Proyectos que crea instancias de la clase Imagen y de la clase Configuración, por lo que tiene una lista de imágenes y de configuraciones.

Controlador: Asigna a clases específicas la responsabilidad de controlar el flujo de eventos del sistema. Esto facilita la centralización de actividades (validaciones, seguridad, etc.). El controlador no las realiza, sino las delega en otras clases con las que mantiene un modelo de alta cohesión. (32)

Este patrón se ve reflejado en la clase Proyecto. La arquitectura MVC brinda una capa específicamente para los controladores, que son el núcleo de este, y especifica la presencia de este patrón.

2.4.1 Patrones GoF

Los patrones GoF (*Gang of Four*, en español Pandilla de los Cuatro), se clasifican en 3 categorías basadas en su propósito: creacionales, estructurales y de comportamiento. Para el diseño de la herramienta se tuvieron en cuenta con el siguiente patrón creacionales:

***Solitario/Singlenton*:** Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. (33) Se crea una clase que gestiona una sola instancia de ella misma y cada vez que se necesite, simplemente se consulta la clase y esta devolverá siempre, la misma instancia.

Este patrón se ve reflejado en la clase Sirna, encargada de garantizar el acceso y la gestión a todas y cada una de las clases existentes dentro del proyecto.

2.5 Requisitos funcionales

A continuación son descritos los requisitos funcionales de la herramienta:

RF 1: Gestionar proyectos

Los proyectos constituyen los rectores en cuanto a la organización del trabajo en el momento de la extracción de patrones desde imágenes digitales. Para desarrollar cualquier tarea primero es necesario crea el proyecto correspondiente.

1.1 Adicionar un proyecto

La herramienta debe ser capaz de permitir adicionar un nuevo proyecto de extracción de patrones.

1.2 Modificar un proyecto

La herramienta debe ser capaz de permitir modificar un proyecto de extracción de patrones.

1.3 Eliminar un proyecto

La herramienta debe ser capaz de permitir eliminar un proyecto de extracción de patrones.

RF 2: Gestionar imágenes

Cada proyecto ha de tener la relación de las imágenes digitales desde las cuales se extraen los patrones. En este caso el sistema debe permitir gestionar las imágenes digitales para un proyecto de extracción de patrones. Aquí el dato de se procesará será el nombre del archivo (con su dirección absoluta en el disco) y no la imagen como tal.

2.1 Adicionar imágenes

La herramienta debe ser capaz de permitir adicionar una o varias imágenes especificando el proyecto a que pertenecerán.

2.2 Eliminar imágenes

La herramienta debe ser capaz de permitir eliminar una o varias imágenes especificando el proyecto al que pertenecen.

RF 3: Gestionar clases

Cada proyecto debe permitir que se definan las clases que podrán ser asociadas con los patrones extraídos. En este caso el sistema debe permitir gestionar las clases del proyecto.

3.1 Adicionar una clase

La herramienta debe ser capaz de permitir adicionar una nueva clase especificando el proyecto al que pertenecerá.

3.2 Modificar una clase

La herramienta debe ser capaz de permitir modificar una clase especificando el proyecto al que pertenece.

3.3 Eliminar una clase

La herramienta debe ser capaz de permitir eliminar una clase especificando el proyecto al que pertenece.

RF 4: Gestionar configuraciones

La herramienta debe permitir gestionar las configuraciones para un proyecto de extracción. En este caso es necesario que el usuario pueda indicar, para cada configuración, las variables con que han de conformarse los patrones y el tamaño de la vecindad en las imágenes digitales que se deben tener en cuenta durante la extracción de los mismos.

4.1 Adicionar una configuración

La herramienta debe ser capaz de permitir adicionar una nueva configuración especificando el proyecto al que pertenece.

4.2 Modificar una configuración

La herramienta debe ser capaz de permitir modificar una configuración especificando el proyecto al que pertenece.

4.3 Eliminar una configuración

La herramienta debe ser capaz de permitir eliminar una configuración especificando el proyecto al que pertenece.

RF 5: Gestionar patrones

Este es el requisito funcional más importante de la aplicación. El sistema debe permitir seleccionar un pixel en la imagen que se muestre y construir el patrón con las variables seleccionadas en la configuración correspondiente. Además, debe permitir asociarle una clase de las definidas en la gestión de los proyectos.

5.1 Adicionar patrón

La herramienta debe ser capaz de permitir adicionar un nuevo patrón especificando la clase, la imagen y la configuración a la que ha de pertenecer. Esta funcionalidad debe permitir que el usuario seleccione un punto sobre una imagen digital, calcular los valores de las variables seleccionadas en una vecindad del mismo, construir con ellas el patrón, asociarle la clase seleccionada y almacenarlo en la base de datos.

5.2 Eliminar patrones

La herramienta debe ser capaz de permitir eliminar uno o varios patrones.

RF 6: Gestionar pruebas

La herramienta debe permitir realizar las pruebas correspondientes con los datos de una red neuronal artificial entrenada en NEUROLAB, además de poder guardar, modificar y eliminar los datos asociados a

las pruebas que se deseen. Para ello debe permitir acceder a los datos de cada prueba, visualizar una imagen y sobre ella ejecutar la red neuronal artificial entrenada para obtener la clase a la que pertenece el punto seleccionado sobre dicha imagen.

6.1 Adicionar una prueba

La herramienta debe ser capaz de permitir especificar los datos de una prueba nueva y guardarlos en la base de datos. Bajo esta funcionalidad deben habilitarse, además, los controles necesarios para que le permitan al usuario visualizar una imagen y probar la efectividad de una red neuronal entrenada. Para ello, debe ser capaz de mostrar la clase a la que pertenece un punto (pixel) dentro de la imagen cada vez que el usuario lo seleccione.

6.2 Editar una prueba

La herramienta debe ser capaz de permitir modificar los datos de una prueba previamente seleccionada. Bajo esta funcionalidad deben habilitarse, además, los controles necesarios para que le permitan al usuario visualizar una imagen y probar la efectividad de una red neuronal entrenada. Para ello, debe ser capaz de mostrar la clase a la que pertenece un punto (pixel) dentro de la imagen cada vez que el usuario lo seleccione.

6.3 Eliminar una prueba

La herramienta debe ser capaz de permitir eliminar una prueba.

2.6 Requisitos no funcionales

2.6.1 Apariencia o interfaz externa

La interfaz gráfica debe ser amigable al usuario con un diseño sencillo que permita realizar cualquier tipo de interacción con el sistema de manera fácil. Esto se garantiza con el uso de una plantilla común para todas las interfaces. Todas las vistas deben tener el color azul como predominante. La tipografía definida será *"Arial"* con un tamaño de 12 *píxeles* para una mejor visualización.

2.6.2 Usabilidad

La aplicación podrá ser usada por personas que pueden o no tener conocimientos básicos de informática, por lo que el sistema ha de ser de fácil uso.

2.6.3 Portabilidad

La herramienta tiene como ventajas, que al ser desarrollada en el lenguaje Java, la convierte en multi-plataforma. Podrá ser usada bajo cualquier distribución de Linux, Windows o cualquier otro sistema operativo en el que se instale la JVM.

2.6.4 Seguridad

Esta herramienta puede ser ejecutada y utilizada por cualquier usuario que desee explorar imágenes y extraer de ellas patrones para, después, entrenar redes neuronales artificiales con el uso de NEUROLAB. El grupo de desarrollo de NEUROLAB pidió que esta herramienta pudiera ser utilizada por cualquiera y por lo tanto no necesitaba de control de usuario ni de seguridad. Por lo tanto esta característica de SIRNA no procede.

2.6.5 Hardware

La estación de trabajo debe cumplir los siguientes requisitos mínimos, al menos 2GB de RAM, un procesador con velocidad igual o mayor a 2.0 GHz, además de tener 40 GB mínimo de capacidad de disco duro y a pesar de que no es imprescindible pudiera tener una tarjeta de red.

2.7 Proceso de vida de la propuesta

El ciclo de vida de XP se enfatiza en el carácter iterativo e incremental del desarrollo. Una iteración de desarrollo es un período de tiempo en el que se realiza un conjunto de funcionalidades determinadas que en el caso de la metodología seleccionada corresponden a un conjunto de Historias de Usuario (HU). El ciclo de vida de XP consta de varias fases, a continuación se exponen algunas de ellas.

2.7.1 Fase de Exploración

Durante esta fase se enmarca el alcance de la herramienta propuesta, que tiene como fin el desarrollo y la entrega de la misma. Para ello el cliente definió sus necesidades a través de las HU, a partir de las cuales los programadores estimaron el tiempo de desarrollo.

Historias de Usuario

Mediante las Historias de Usuario (HU), escritas por el cliente en su propio lenguaje, se describe lo que la herramienta debe realizar. Las HU son las técnicas utilizadas para especificar los requisitos del *software* tanto funcionales como no funcionales. La diferencia más notable entre estas historias y los documentos de especificación de requisitos, utilizados por RUP, se encuentran en el nivel de detalle requerido. Dichas HU permiten a los programadores realizar una estimación poco riesgosa del tiempo que llevará el desarrollo (34).

CAPÍTULO 2. Descripción de la Solución

Para definir las HU se ha utilizado la siguiente tabla, la cual contiene todos los datos necesarios para desarrollar la funcionalidad descrita.

Historia de Usuario	
Numero: número de la HU, incremental en el tiempo.	**Nombre:** el nombre de la HU, sirve para identificarla fácilmente entre los desarrolladores y los clientes.
Usuario: el usuario del sistema que utiliza o protagoniza la historia.	
Prioridad en Negocio: que tan importante es para el cliente.	**Riesgo en Desarrollo:** que tan difícil es para el desarrollador.
Iteración Asignada: la iteración (liberación en nuestro proceso) a la que corresponde.	**Puntos estimados:** estimación del tiempo de duración de la HU. Cuando el valor es 1 equivale a una semana ideal de trabajo. En la metodología XP está definida una semana ideal como 5 días hábiles trabajando 40 horas, es decir, 8 horas diarias. Por lo que cuando el valor de dicho atributo es de 0.5 equivale a 2 días y medio de trabajo, lo que se traduce en 20 horas.
Descripción: la descripción de la historia, detallando las operaciones del usuario y opcionalmente las respuestas del sistema.	
Observaciones: algunas observaciones de interés, como glosario, información sobre usuarios, etc.	

Tabla 1. *Representación de una Historia de Usuario.*

Para las HU que describen las características y cualidades que debe cumplir la solución, los autores del presente trabajo decidieron excluir los siguientes aspectos: usuario, riesgo en desarrollo, iteración asignada y puntos estimados.

Durante todo el proceso se identificaron 6 Historias de Usuario, las cuales se mencionan a continuación y se describen en los anexos:

No.	Historia de Usuario
1	Gestionar Proyecto

2	Gestionar Imágenes
3	Gestionar Clases
4	Gestionar Configuraciones
5	Gestionar Patrones
6	Gestionar Prueba

Tabla 2. *Historias de Usuario.*

2.7.2 Fase de Planificación

En esta fase se estableció la prioridad de cada historia de usuario por parte del cliente, momento a partir del cual los programadores realizaron una estimación del esfuerzo necesario para desarrollar cada una de ellas.

Estimación de esfuerzo por Historias de Usuario

Para el buen desarrollo de la herramienta propuesta, se realizó una estimación de cada una de las HU identificadas que definen las funcionalidades, la cual arrojó los resultados que se muestran a continuación:

Historia de Usuario	Puntos de Estimación
Gestionar Proyecto	1
Gestionar Imágenes	1
Gestionar Clases	1
Gestionar Configuraciones	2
Gestionar Patrones	2
Gestionar Prueba	2

Tabla 3. *Estimación de esfuerzo por HU*

CAPÍTULO 2. Descripción de la Solución

Plan de iteraciones

Una vez descritas las HU que definen las funcionalidades por parte del cliente y estimado el esfuerzo por los desarrolladores para la realización de las mismas, se procede a realizar la planificación de la etapa de implementación del sistema. Este plan define las HU que serán implementadas en cada iteración y las posibles fechas de sus entregas. En relación con lo antes mencionado se decide implementar el sistema en dos iteraciones, las cuales se describen a continuación:

Iteración 1:

En esta iteración se implementan las HU que tienen prioridad alta en el negocio, de esta forma, se van creando las funcionalidades principales de la herramienta que dan soporte a la implementación de las demás funcionalidades. Estas HU (de la 1 a la 4) hacen alusión de modo general a la gestión de proyectos garantizando la inclusión al mismo de las imágenes, clases y configuraciones pertinentes. Al finalizar dicha iteración se realizará la primera entrega de la herramienta con el fin de recibir retroalimentación por parte del cliente.

Iteración 2:

Esta iteración tiene como objetivo implementar las restantes funcionalidades requeridas por la herramienta. Al finalizarla se tendrán implementadas las peticiones del cliente descritas en las HU (de la 5 a la 7) y se contará con la versión final del producto. A partir de este momento la herramienta será puesta a prueba durante un período de tiempo para evaluar el desempeño de la misma.

Iteración	Historia de Usuario	Duración total de la iteración
Iteración 1	Gestionar Proyecto Gestionar Imágenes Gestionar Clases Gestionar Configuraciones	5 Semanas
Iteración 2	Gestionar Patrones Gestionar Prueba	4 Semanas

Tabla 4. *Plan de duración de las iteraciones.*

2.8 Conclusiones

En el presente capítulo se presentó un esbozo de la aplicación con la explicación de los procesos de mayor complejidad, elementos del diseño de la misma, los diferentes requisitos y las historias de usuario que han de ejecutarse para brindar una herramienta capaz de lograr los objetivos para los que fue concebida y se desarrolla, entre otros elementos.

Ahora se está en condiciones de pasar a exponer los elementos que intervinieron y se desarrollaron durante el proceso de implementación.

CAPÍTULO 3. Implementación y Prueba.

3.1 Introducción

El momento cumbre del desarrollo de un proyecto es la implementación de cada una de las funcionalidades requeridas. La metodología XP plantea que la implementación de un producto debe realizarse de forma iterativa, esta característica trae consigo que después del desarrollo de cada iteración se obtenga un producto funcional que debe ser mostrado al cliente y previamente probado para incrementar la visión de los desarrolladores y clientes de posibles cambios. En el presente capítulo se detallan las dos iteraciones llevadas a cabo durante la construcción de la herramienta, además se exponen las tarjetas Clase, Responsabilidad y Colaboración (CRC) y las tareas generadas por cada Historia de Usuario de las que definen las funcionalidades. Por último, se le realizan las pruebas pertinentes al producto final, verificando de forma unitaria las funcionalidades de la herramienta, y validando junto al usuario final la solución desarrollada a fin de contar con todas las evidencias que garanticen el cumplimiento en alcance, funcionalidad y calidad que el cliente espera.

3.2 Diseño del sistema

La metodología XP sugiere que se deben obtener diseños simples y sencillos mediante las tarjetas CRC. El desarrollo debe ser lo menos complicado posible para conseguir un diseño agradable y de fácil interacción que permita la validez de los principios de una aplicación de escritorio.

Tarjetas Clase, Responsabilidad y Colaboración

La tarjeta CRC es una técnica de modelado orientado a objetos que permite identificar las clases y sus responsabilidades. El nombre de la clase se coloca en forma de título en la tarjeta, en la parte izquierda las funcionalidades (responsabilidades) y en la parte derecha las clases que se implican en cada funcionalidad (colaboración). A continuación se muestra la plantilla que corresponde a las tarjetas CRC, las que permitieron definir y simular los escenarios que garantizan el buen funcionamiento del diseño y las mismas se describen en los anexos:

Nombre de la clase	
Responsabilidad	**Colaboración**

Tabla 5. *Tarjeta CRC.*

Capítulo 3. Implementación y Pruebas

3.3 Modelo de datos

A pesar de que la metodología utilizada no define ningún artefacto del proceso de desarrollo para visualizar el progreso de la herramienta, algunos autores encuentran útil mantener algunos artefactos, siempre y cuando el tiempo dedicado a mantenerlos sea mucho menor que el tiempo dedicado al desarrollo. Bajo estos criterios se decide mantener un diagrama que puede ser útil para una mejor compresión del funcionamiento de la herramienta: el modelo de datos.

Una base de datos es un almacén que permite guardar una serie de datos organizados y relacionados entre sí, los cuales son recolectados y explotados, de forma organizada. La figura muestra el modelo de datos correspondiente a la base de datos de la herramienta propuesta. Para ver las tablas con más detalles.

Típicamente un modelo de datos permite:

- **Descripción de datos**: El tipo de los datos que hay en la base y la forma en que se relacionan.
- **Reglas de integridad**: Un conjunto de condiciones que deben cumplir los datos para reflejar correctamente la realidad deseada.
- **Operaciones**: Típicamente, operaciones de agregado, borrado, modificación y recuperación de los datos de la base.

Para el diseño del modelo de datos fueron utilizadas las potencialidades del Embarcadero en su versión 8.0, el cual se encarga de realizar el diseño de la base de datos.

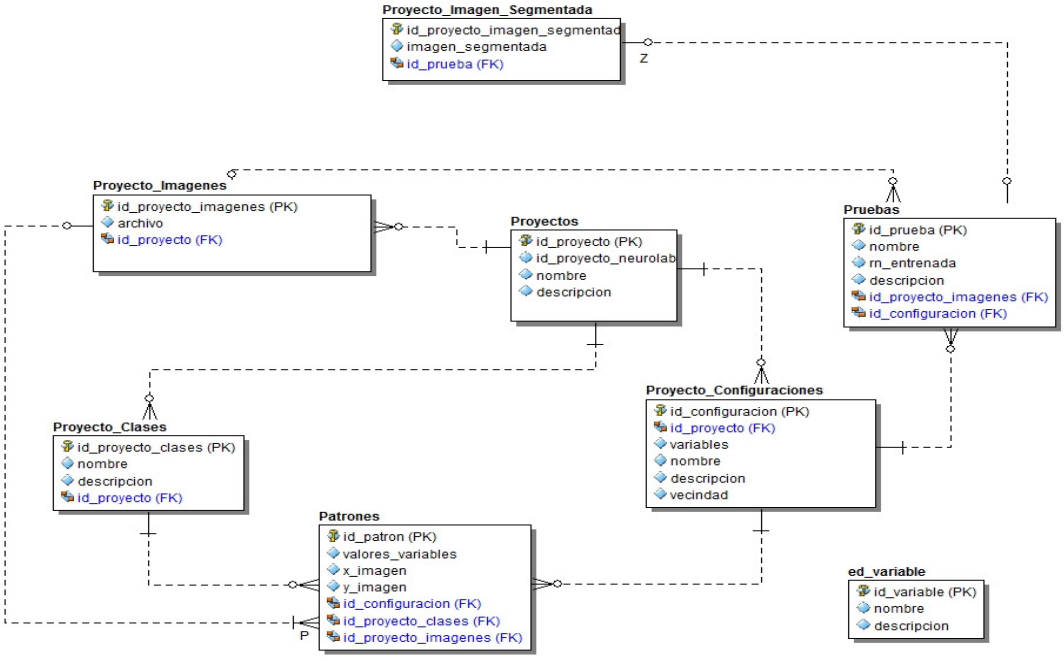


Figura 11. *Modelo de Datos.*

Capítulo 3. Implementación y Pruebas

3.4 Fase de implementación

En esta fase se realiza la implementación de la Historia de Usuario que definen las funcionalidades que fueron seleccionadas por cada iteración. Al inicio se lleva a cabo un chequeo del plan de iteraciones por si es necesario realizar modificaciones. Como parte de este plan se crean tareas de ingeniería para ayudar a organizar la implementación exitosa de la Historia de Usuario.

3.4.1 Estándares de codificación.

El uso de estándares o reglas de codificación al comenzar la implementación del sistema, trae para el producto final muchos beneficios. XP enfatiza la comunicación de los programadores a través del código, con lo cual es indispensable que se sigan ciertos estándares de programación (21).

Los estándares de codificación son pautas de programación que no están enfocadas a la lógica del programa, sino a su estructura y apariencia física para facilitar la lectura, comprensión y mantenimiento del código. Mejoran la legibilidad del código. (22) Permiten que otras personas puedan colaborar con facilidad, en consecuencia el mantenimiento es menos complejo.

Java tiene definido un conjunto de buenas prácticas para el código, a continuación se mencionan algunas de las utilizadas durante la fase de implementación de la herramienta:

Organización de ficheros: las clases se agrupan en paquetes. Estos paquetes se organizan de manera jerárquica.

Fichero fuente Java (*.java): cada fichero fuente contiene una única clase o interfaz pública. El nombre del fichero coincide con el nombre de la clase.

Sentencias de paquete: la primera línea no comentada de un fichero fuente es la sentencia de paquete, que indica el paquete al que pertenece la clase incluida en el fichero fuente.

Sentencias de importación: tras la declaración del paquete se incluyen las sentencias de importación de los paquetes necesarios.

Sangría: como norma general se establecen 4 caracteres como unidad de sangría.

Comentarios de implementación: estos comentarios se utilizan para describir el código ("el cómo"), y en ellos se incluye información relacionada con la implementación.

Una declaración por línea: se hace uso de una declaración por línea, promoviendo así el uso de comentarios.

Inicialización: toda variable local es inicializada en el momento de su declaración, salvo que su valor inicial dependa de algún valor que tenga que ser calculado previamente.

Declaración de clases / interfaces: durante la implementación de las clases o interfaces se siguen las siguientes reglas de formateo:

- No incluir ningún espacio entre el nombre del método y el paréntesis inicial del listado de parámetros.
- El carácter inicio de bloque ("{") debe aparecer al final de la línea que contiene la sentencia de declaración.
- El carácter fin de bloque ("}") se sitúa en una nueva línea tabulada al mismo nivel que su correspondiente sentencia de inicio de bloque, excepto cuando la sentencia sea nula, en tal caso se situará detrás de ("{").
- Los métodos se separarán entre sí mediante una línea en blanco.

Sentencias: cada línea contiene como máximo una sentencia.

Espacios en blanco: las líneas y espacios en blanco se usan para mejorar la legibilidad del código permitiendo identificar las secciones de código relacionadas lógicamente.

Nomenclatura de identificadores

-Paquetes: se escriben siempre en letras minúsculas para evitar que entren en conflicto con los nombres de clases o interfaces.

-Clases e interfaces: los nombres de clases son sustantivos y tienen la primera letra en mayúscula.

-Métodos: los métodos son verbos escritos en minúsculas. Cuando el método está compuesto por varias palabras cada una de ellas tiene la primera letra en mayúscula.

-Variables: las variables se escriben siempre en minúscula. Las variables compuestas tienen la primera letra de cada palabra componente en mayúscula.

3.4.2 Tareas de ingeniería por iteraciones.

Cada Historia de Usuario como funcionalidad de la aplicación está compuesta por una o varias tareas de ingeniería, éstas no son más que pasos lógicos a seguir por el programador para realizar la implementación de una Historia de Usuario. A continuación se detallan para cada una de las iteraciones las tareas a desarrollar por cada Historia de Usuario.

Capítulo 3. Implementación y Pruebas

Iteración 1:

En esta iteración se implementan las Historias de Usuario que conforman la estructura básica de la herramienta. A partir de ellas se realizan las demás funcionalidades requeridas por la aplicación.

Módulos	Historia de Usuario	Tiempo de Implementación (Semanal)	
		Estimado	Real
Proyecto	Gestionar Proyecto	1	1
Imagen	Gestionar Imagen	1	1
Clase	Gestionar Clase	1	1
Configuración	Gestionar Configuración	2	2

Tabla 6. *Tiempo real de implementación por módulo (iteración 1).*

Historia de Usuario	Tareas de Ingeniería por HU
Gestionar Proyecto	- Adicionar Proyecto - Modificar Proyecto - Eliminar Proyecto - Ver Proyecto(s)
Gestionar Imagen	- Adicionar Imagen(es) - Eliminar Imagen(es) - Ver Imagen(es)
Gestionar Clase	- Adicionar Clase - Modificar Clase - Eliminar Clase - Ver Clases
Gestionar Configuración	- Adicionar Configuración - Modificar Configuración - Eliminar Configuración - Ver Configuración(es)

Tabla 7. *Tareas de ingeniería por historias de usuario (iteración 1).*

Iteración 2:

En esta iteración se implementan las restantes Historias de Usuario requeridas por la herramienta a desarrollar.

Módulos	Historia de Usuario	Tiempo de Implementación (Semanal)	
		Estimado	Real
Patrón	Gestionar Patrón	2	2
Prueba	Gestionar Prueba	2	2

Tabla 8. *Tiempo real de implementación por módulo (iteración 2).*

Historia de Usuario	Tareas de Ingeniería por HU
Gestionar Patrón	- Adicionar Patrón - Eliminar Patrón(es) - Ver Patrón(es)
Gestionar Prueba	- Adicionar Prueba - Editar Prueba - Eliminar Prueba - Ver Prueba(s)

Tabla 9. *Tareas de ingeniería por historia de usuario (iteración 2).*

A continuación se muestra la plantilla que describe las tareas de ingeniería y las mismas se describen en los anexos:

Tarea de ingeniería	
Número Tarea: Numero de la tarea	**Número de HU:** Numero de la Historia de Usuario
Nombre Tarea: Nombre de la tarea	
Tipo de tarea: Tipología de la tarea	**Puntos Estimados:** Tiempo estimado de la duración de la tarea.

Fecha Inicio: Fecha de inicio de la tarea	**Fecha Fin:** Fecha de fin de la tarea
Programador Responsable: Nombre de los programadores responsables de realizar la tarea	
Descripción: Breve descripción de la tarea en cuestión.	

Tabla 10. *Representación de una Tarea de Ingeniería.*

3.4.3 Reseña de la implementación.

La implementación de la herramienta comenzó con el diseño de la base de datos que se encarga de guardar los datos persistentes de los proyectos que se crean. Luego de haber diseñado cada una de las entidades que se verán presente y sus atributos correspondientes, se generaron las tablas de la base de datos, a través del controlador para conectar la base de datos MySQL con Java: “mysql-connector-java-5.1.23-bin.jar”. A partir de aquí se fueron creando cada una de las clases del paquete “Modelo” que representan a cada tabla de la base de datos con sus respectivos atributos. Definiéndose de manera general la siguiente relación entre ellas: la clase principal (ControlProyecto.java) tendrá el control de cada uno de los proyectos que se creen, representados por la clase Proyecto.java; a su vez la clase ControlImagen.java y ControlClase.java mantiene el control de las imágenes y clases que van a estar asociadas al proyecto en cuestión representados por las clases “Proyecto_Imagen.java” y “Proyecto_Clase.java” respectivamente, cada proyecto debe tener estrecha relación con las configuraciones que van a estar representada por la clase “Proyecto_Configuracion.java”, esta se encarga de almacenar las configuraciones en las cuales se van seleccionar las variables que van a ser calculadas y controlada desde la clase “ControlConfiguracion.java”, , representados por la clase “Patron.java” están los patrones y las pruebas que se realizaran con el objetivo de conocer si los patrones que fueron extraídos y guardados son correctos estará representada por la clase “Prueba.java” pero controlados cada una desde sus respectivas clases controladoras nombradas ControlPatron.java y ControlPrueba.java respectivamente.

Además fueron utilizadas algunas estructuras de datos propias del lenguaje que ayudaron a modelar los datos de una manera más fácil y cómoda; entre ellas se utilizó la lista para guardar varios objetos de una clase, por ejemplo en la clase SIRNA que contiene todos los proyectos que se crean y otro ejemplo de este uso es en la clase Proyecto cuando se agregan las imágenes y las clases.

Luego de haber modelado todas las clases modelo, fue implementada la vista principal desde la cual se va a gestionar todas y cada una de las clases que componen el modelo, siguiendo los prototipos mostrados al cliente con anterioridad y fueron agrupadas en el paquete “Interfaces”.

Mediante estas clases se muestra la interfaz visual con que interactuará el cliente, en ella se implementaron las validaciones necesarias para hacer que el usuario haga entradas de datos correctos y no viole pasos en el proceso de extracción de patrones, manteniendo una comunicación mediante mensajes de alerta, error y éxito.

La generación de las clases controladoras a partir de las clases del Modelo, fue el próximo paso en la implementación. Fueron agrupadas en un paquete llamado “Control”, se obtuvo una clase controladora por cada clase entidad, mediante ellas las entidades se comunican con la base de datos, estas clases son las encargadas de crear, modificar y eliminar los datos existentes en la base de datos usando el driver para la conexión entre el servidor de base de datos MySQL y el lenguaje JAVA, para ello fue utilizada la clase ConexionMySQL.

A medida que se fueron haciendo las entregas se fueron haciendo modificaciones a la herramienta adaptándola a los nuevos requerimientos y necesidades del cliente logrando así un producto final con la funcionalidad y calidad esperada.

3.5 Fase de Pruebas

En XP uno de los pilares más importantes es el proceso de pruebas, en el cual los desarrolladores prueban tanto como sea posible. Mediante esta filosofía se reduce el número de errores no detectados, así como el tiempo de introducción de éste en el sistema y su detección. Todo esto permite una mejor calidad en los productos desarrollados y la seguridad de los programadores a la hora de introducir algún cambio o modificación en el sistema. (38)

La metodología XP divide las pruebas en dos grupos: las pruebas unitarias o de caja blanca, que son realizadas por los programadores, encargadas de verificar el código y las pruebas de aceptación o de caja negra, destinadas a verificar si al final de cada iteración se obtuvo la funcionalidad que se requiere, además de comprobar que dicha funcionalidad haga lo que el cliente quiere.

3.5.1 Pruebas unitarias

Las pruebas unitarias o también llamadas pruebas de caja blanca se basan en realizar pruebas al código del sistema. Para llevar a cabo esta tarea se comprueban los caminos lógicos de la aplicación mediante casos de prueba, que pongan a prueba los algoritmos implementados. Las pruebas unitarias no se le pueden realizar a todo el código de la aplicación, ya que el número de caminos lógicos puede llegar a crecer de manera exponencial lo cual imposibilita realizar casos de pruebas para todos estos caminos y mucho menos procesarlos todos. (39) Por este motivo las pruebas de caja blanca se realizan a los principales algoritmos o procedimientos.

Capítulo 3. Implementación y Pruebas

Uno de los métodos o técnicas de prueba unitarias, es la prueba del camino básico. Esta técnica permite obtener una medida de la complejidad de un procedimiento o algoritmo y un conjunto básico de caminos de ejecución de este, los cuales luego se utilizan para obtener los casos de prueba.

Esta técnica será la utilizada para desarrollar los casos de pruebas unitarias de la herramienta desarrollada. De igual manera existen varias métricas de *software* para realizar pruebas unitarias, entre estas se encuentra la complejidad ciclomática, la cual será utilizada junto a la técnica explicada anteriormente. Esta métrica proporciona una medición cuantitativa de la complejidad lógica de un procedimiento.

La complejidad ciclomática cuando se utiliza en el contexto del método de prueba del camino básico, el valor que se calcula como complejidad ciclomática define el número de caminos independientes dentro de un fragmento de código y determina la cota superior del número de pruebas que se deben realizar para asegurar que se ejecuta cada sentencia al menos una vez. (25)

Para realizar las pruebas unitarias existen distintas herramientas, en específico para las pruebas al código en lenguaje Java existe JUnit, un framework utilizado para automatizar las pruebas; permitiendo que el esfuerzo y el trabajo en la fase de pruebas se reduzca.

A continuación se explica, mediante un ejemplo, todo el procedimiento que se siguió para obtener los casos de prueba utilizando la técnica del camino básico junto con la métrica complejidad ciclomática.

Este algoritmo lleva como nombre “ValidarImagenDuplicada”, y tiene como finalidad buscar en la lista de alternativas de un proyecto determinado, si existe una alternativa cuyo nombre coincida con el que recibe por parámetro.

```
public int ValidarImagenDuplicada(String strImgFile) {
        int iRet = 0; // 1
        if( strImgFile != null && !strImgFile.trim().isEmpty() && tmpListImg != null ) { // 1
                strImgFile = strImgFile.replace("\\", ";"); // 2
                for (int i = 0; i < tmpListImg.size(); i++) { // 2
                        if (tmpListImg.get(i).getArchivo().equalsIgnoreCase(strImgFile)) { // 3
                                iRet = 1; // 4
                                break; // 4
                        }
```

```
            }
        } else
            iRet = -1; // 5
        return iRet; // 6
    }
```

A partir del algoritmo, se dibuja el grafo de flujo asociado.

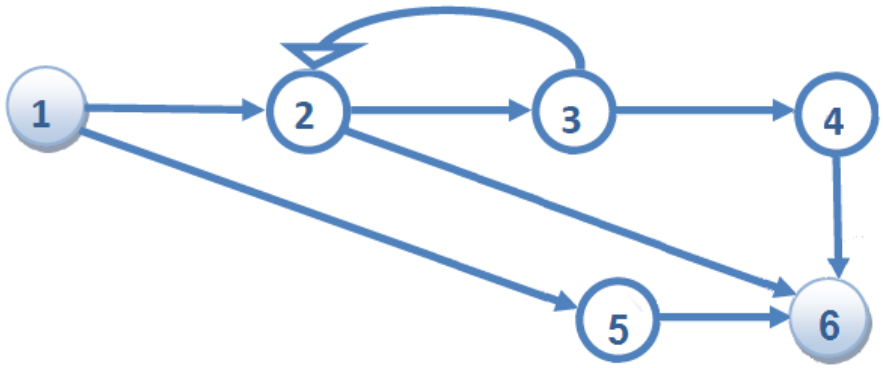


Figura 12. *Grafo de flujo de la funcionalidad "ValidarImagenDuplicada".*

Luego se determina la complejidad ciclomática V(G) del grafo resultante G. Para esto se utiliza la siguiente fórmula:

V(G) = A - N + 2

Donde: A es el número de aristas del grafo y N es el número de nodos.

V(G) = A - N + 2

V(G) = 8 – 6 + 2

V(G) = 4

El número de caminos independientes de la estructura del programa es igual a 4, y los caminos independientes son:

Camino 1: 1, 2, 3, 4, 6

Camino 2: 1, 2, 3, 2, 6

Camino 3: 1, 2, 6

Camino 4: 1, 5, 6

Luego se definen 4 casos de prueba para el código del algoritmo, uno para cada camino. Para realizar estos casos de prueba se utilizó la siguiente plantilla. Para ver los casos de prueba realizados al ejemplo.

Caso de prueba unitaria
Camino: nombre
Caso de Prueba: nombre
Entrada: descripción textual de lo que ocurre en el mundo real que hace necesario ejecutar el caso de la prueba, precisando la data de entrada y los comandos a dar por el actor. Descripción textual del estado de la información almacenada.
Resultado: descripción textual del estado en el que queda la información y las alertas que puedan generarse, una vez ejecutado el caso de uso con los valores y el estado especificado en la entrada
Condiciones: condiciones que deben cumplirse mientras se ejecuta el caso de prueba

Tabla 11. *Representación de caso de prueba unitaria*

3.5.2 Pruebas de aceptación

Las pruebas de aceptación son un tipo de prueba de caja negra orientadas a evaluar las distintas tareas en las que ha sido dividida una HU. Para asegurar el funcionamiento final de una determinada historia los test son creados y usados por los clientes para comprobar que éstas cumplen su cometido.

Un relato de usuario puede tener una o varias pruebas de aceptación y no está completo hasta no haberlas pasado todas. El cliente es el máximo responsable de verificar cada una de las pruebas y de priorizar la corrección de las pruebas fallidas.

La utilización de estas pruebas fue de vital importancia en el proceso de desarrollo ya que permitió a los programadores tener una idea más clara e inequívoca de la calidad del trabajo, a la vez se garantizó la entrega de un producto en elevada correspondencia con las necesidades del usuario final.

Seguidamente se muestra la plantilla de caso de prueba de aceptación definida por el cliente:

Caso de prueba de aceptación

Código: código que identifica la prueba.	**Historia de Usuario:** número de la historia de usuario relacionada con la prueba que se realiza.
Nombre: definición de la prueba que se realiza.	
Descripción: breve descripción del objetivo con que se realiza la prueba.	
Condiciones de Ejecución: condiciones que deben satisfacerse para que pueda realizarse la prueba.	
Entrada/Pasos de Ejecución: se describen los pasos de ejecución de la prueba en cuestión.	
Resultado Esperado: proporciona las expectativas ideales para las cuales fue pensada la prueba.	
Evaluación de la Prueba: calificación que recibe la prueba de acuerdo con los resultados obtenidos.	

Tabla 12. *Representación de caso de prueba aceptación.*

Iteración 1

Para la primera iteración se realizaron 22 casos de pruebas de aceptación, identificando 5 no conformidades, 3 no significativas y dos significativas. Debido a la complejidad mínima de éstas, fueron todas resueltas y no quedaron casos de pruebas pendientes para la siguiente iteración.

Iteración 2

Para esta iteración se realizaron 7 casos de pruebas de aceptación, identificando 3 no conformidades, de ellas 2 son significativas y 1 no significativa. Estas tres no conformidades se solucionaron dando fin al proceso de pruebas de aceptación y obteniendo resultados satisfactorios.

Las no conformidades, no significativas, se centraron en errores ortográficos como: omisiones de tildes, paréntesis, cambio de mayúscula por minúscula y aceptar letras donde se esperaban valores numéricos, y las significativas, en errores de validación y cambios en el diseño.

En la sección correspondiente a los anexos se muestran los casos de pruebas de aceptación realizados

Resultados de las pruebas de aceptación

	Iteración 1	Iteración 2
CP de Aceptación	1	1
No conformidades	3	2
Significativas	3	3
No Significativas	2	2
Resueltas	3	3
Pendientes	0	0

Tabla 13. *Resultado de las pruebas de aceptación.*

3.6 Conclusiones parciales

En el presente capítulo se abordó sobre la fase de diseño de la herramienta donde se crearon 7 tarjetas CRC (ver anexos), las cuales brindaron claridad en aspectos como las principales funcionalidades que presentan las clases así como la relación existente entre ellas. Se hace alusión a las etapas de implementación y prueba de la herramienta en desarrollo, donde se realizaron 22 tareas de ingeniería para dar solución a las HU planteadas, así como el modelo de datos, y se describe el proceso de prueba, uno de los más importantes para garantizar el éxito de la aplicación, éste se llevó a cabo con el objetivo de brindarle al cliente un producto funcional que cumpla con sus necesidades especificadas.

Con el fin del capítulo se da por terminada la propuesta de solución relacionada al desarrollo de la herramienta, inherente al trabajo de diploma: “Implementación de la herramienta SIRNA para la extracción interactiva de patrones a partir de imágenes digitales y su entrega al sistema NEUROLAB”.

CONCLUSIONES GENERALES

Luego de exponer el problema que se desea resolver, los diferentes conceptos teóricos que sirven de fundamentación para el trabajo aquí expuesto, los detalles de los elementos que conforman la propuesta de solución y la implementación y pruebas correspondientes, se está en condiciones de expresar que los objetivos han sido cumplidos con el mayor rigor. Además, se puede arribar a las siguientes conclusiones:

1. Fue construida una herramienta que permite obtener información de imágenes digitales con el objetivo de entregárselos a NEUROLAB para el entrenamiento de redes neuronales artificiales.
2. Fue construida una herramienta que permite obtener información de imágenes digitales con el objetivo de utilizarla para el reconocimiento de patrones empleando redes neuronales artificiales.
3. Se desarrolló una aplicación que cubre satisfactoriamente las necesidades iniciales expresadas por el grupo de desarrollo de NEUROLAB, pues ya cuentan con una herramienta capaz de permitir a sus usuarios construir patrones a partir de imágenes digitales, entrenar redes neuronales artificiales con ellos y hacer pruebas posteriores con la red entrenada.
4. Se realizaron varias pruebas que permitieron comprobar la funcionalidad y la calidad de la solución propuesta.
5. Se logró demostrar la usabilidad de SIRNA en muchas esferas del quehacer del hombre donde se empleen representaciones pictográficas (por ejemplo: fotos, imágenes obtenidas de equipos de resonancia magnética, ultrasonidos, etc.) para la extracción de diferentes datos que deban ser analizados. Ya sea en el campo de la teledetección, la medicina, la meteorología, etc.

El uso de la metodología XP, para guiar el proceso de desarrollo de la herramienta y los artefactos generados en cada una de sus fases, propició una mayor comunicación entre el equipo de desarrollo y el cliente, logrando obtener un mejor funcionamiento de la herramienta y el cumplimiento de las necesidades del cliente. Por lo que se considera concluida la investigación y dados por cumplido todos los objetivos establecidos al comienzo de la misma.

RECOMENDACIONES

Como resultado del desarrollo de la aplicación aquí propuesta y la necesidad de una primera versión capaz de resolver el problema inicial, surgen elementos que pudieran formar parte de SIRNA, dándole mayores prestaciones y facilidades de explotación. A continuación se enumeran algunos de los más importantes:

1. Habilitar los controles necesarios para que el sistema sea capaz de leer y cargar imágenes desde archivos con otros formatos que no sea únicamente JPG. Por ejemplo, BMP, GIF, TIF, PNG, DICOM, etc.
2. Implementar un método que permita segmentar la imagen en la sección de pruebas. De esta forma el usuario tendría la posibilidad de observar (y hacer mediciones correspondientes) el poder clasificador de la red neuronal artificial entrenada y comparar la imagen segmentada con la original.
3. Habilitar un control que permita que el usuario pueda obtener diferentes parámetros de la imagen digital segmentada que le permitan realizar estudios posteriores con dichos parámetros. Por ejemplo, pudiera pensarse en habilitar un control que se capaz de calcular el porciento de píxeles asociados a cada clase con respecto a toda la imagen.
4. Implementar esta herramienta para un ambiente web. Esto permitiría una mejor integración con NEUROLAB y otras herramientas más que están desarrolladas para ese entorno.

Utilizar este trabajo como material de estudio en la creación de alguna aplicación similar.

BIBLIOGRAFÍA

1. http://definicion.de/imagen-digital/ [En línea] [Citado el: 28 de marzo de 2015.]
2. Murray R. Spiegel. *Teoría y problemas de Estadística*.
3. http://mathworld.wolfram.com/Skewness.html [En línea] [Citado el: 29 de marzo de 2015.]
4. http://mathworld.wolfram.com/Kurtosis.html [En línea] [Citado el: 29 de marzo de 2015.]
5. Andrew R. Webb, Keith D. Copsey. *Statistical Pattern Recognition*. Third Edition. 2011.
6. Reyes González, Yunia, González Bravo, Lisbet M. y Ruiz Constanten, Yadira**.** [En línea] [Citado el: 12 de Febrero de 2015.] http://ccia.cujae.edu.cu/index.php/siia/siia2008/paper/viewFile/1240/298.
7. Kruchten, Philippe**.** *Rational Unified Process, An Introduction, 3rd Edition.* 2003.
8. http://eva.uci.cu/file.php/161/Documentos/Materiales_complementarios/UD_1_Procesos/Metodologias/METODOLOGIAS_TRADICIONALES_VS._METODOLOGIAS_AGILES.pdf. [En línea] [Citado el: 2 de Marzo de 2015.]
9. http://edumexico.net/secundaria/plan%20de%20estudios/computacion2/apuntes_sec9.htm. [En línea] [Citado el: 15 de Abril de 2015.]
10. https://www.wikispaces.com/file/view/Lenguajes formales.docx. [En línea] [Citado el: 15 de Abril de 2015.]
11. http://repositorio_institucional.uci.cu/jspui/bitstream/ident/TD_03708_10/1/TD_03708_10.pdf. [En línea] [Citado el: 15 de Abril de 2015.]
12. Ellis, Margaret A. y Stroustrup, Bjarne. *C++ Manual de referencia con anotaciones*. [En línea] [Citado el: 15 de abril de 2015.] http://www.google.com.cu/books?hl=es&lr=&id=8XB8-DMjKNMC&oi=fnd&pg=PR5&dq=historia+de+c%2B%2B&ots=0BHchtb4e&sig=YIIgXhHACtF6auf2SnU1Aj9vJOQ&redir_esc=y#v=onepage&q&f=false.
13. Lerdorf, Rasmus, Tatroe, Kevin y MacIntyre, Peter**.** *Programming PHP*. [En línea] http://www.google.com.cu/books?hl=es&lr=&id=h-E1lVkoskC&oi=fnd&pg=PT7&dq=php+reference&ots=yoH2eUMrRl&sig=cmT7lr--CwS9VmeKPrrCCMtsOE&redir_esc=y#v=onepage&q=php%20reference&f=false. [Citado el: 15 de Abril de 2015.]
14. Deitel, Paul J y Deitel, Harvey M. *Cómo programar en Java.* México: s.n., 2008. 978-970-26-1190-5.

Bibliografía

15. http://www.sc.ehu.es/sbweb/fisica/cursoJava/fundamentos/introduccion/virtual.htm. [En línea] [Citado el: 17 de Abril de 2015.]
16. González Duque, Raúl;. DSpace at Universia: [En línea] [Citado el: 17 de Abril de 2015.] http://dspace.universia.net/bitstream/2024/919/1/Python_para_todos.pdf.
17. https://sites.google.com/site/icascervantes/servidor-de-base-de-datos. [En línea] [Citado el: 17 de Abril de 2015.]
18. Cobo, Ángel. *PHP y MySQL: Tecnología para el desarrollo de aplicaciones web*. [En línea] [Citado el: 15 de marzo de 2015.] http://www.google.com.cu/books?hl=es&lr=&id=zMK3GOMOpQ4C&oi=fnd&pg=PR17&dq=php+es+un+len.
19. http://www.postgresql.org/about/. [En línea] [Citado el: 17 de Abril de 2015.]
20. http://searchsoftwarequality.techtarget.com/definition/integrated-development-environment. [En línea] [Citado el: 9 de marzo de 2015.]
21. Hommel, Scott**.** *Convenciones de Código para el lenguaje de programación.* 1999.
22. http://ingenieria.uatx.mx/marva/files/2011/02/CODIFICACION.pdf. [En línea] [Citado el: 12 de marzo de 2015.]
23. http://www.informatica-juridica.com/trabajos/Ayuda_en_linea_para_el_Sistema_Sentai_Corporacion_importadores_Cuba_Cimex.asp. [En línea] [Citado el: 12 de marzo de 2015.]
24. http://www.xprogramming.com/publications/SP99 Extreme for Web.pdf. [En línea] [Citado el: 12 de marzo de 2015.]
25. Pressman, R. S. *Ingeniería del Software. Un enfoque práctico.* 5ta. 2002.
26. Letelier, Patricio y Penadés, María Carmen. *Metodologías ágiles para el desarrollo de software: Extreme Programming.* Valencia: Universidad Politécnica, 2006.
27. Fowler, M. designDead. Is Design Dead? [En línea] 2001. [Citado el: 17 de mayo de 2013.] http://www.martinfowler.com/articles/designDead.html.
28. Scribd. [En línea] 2007. [Citado el: 25 de abril de 2013.] http://es.scribd.com/doc/34279909/Puntos-Examen-Patrones-Arquitectonicos-Secc01.
29. Bascón Pantoja, Ernesto. *El patrón de diseño Modelo Vista Controlador (MVC) y su implementación en Java Swing.* s.l.: Acta Nova, 2004. Vol. 2.
30. Scribd. [En línea] 2007. [Citado el: 24 de abril de 2013.] http://es.scribd.com/doc/27239031/PATRONES-DE-DISENO.

Bibliografía

31. Introducción a la Arquitectura de Software. [En línea] 2003 - 2008. [Citado el: 25 de abril de 2013.] http://www.willydev.net/InsiteCreation/v1.0/WillyCrawler/2008.05.01.Articulo.Introduccion%20a%20la%20arquitectura%20de%20Soft.pdf.
32. El Mundo Informático. [En línea] 2006. [Citado el: 25 de abril de 2013.] http://jorgesaavedra.wordpress.com/2006/08/17/patrones-grasp-craig-larman.
33. U-cursos. [En línea] 2011. [Citado el: 29 de abril de 2013.] https://www.u-cursos.cl/ingenieria/2011/1/CC61J/1/material_docente/previsualizar?id_material=368981.
34. Pressman, R. S. *Ingeniería del Software. Un enfoque práctico.* 5ta. 2002.
35. Jonas Sjöberg. *Mathematica Neural Networks.* Wolfram Research 2005 First edition.
36. http://numericinsight.com/uploads/A_Gentle_Introduction_to_Backpropagation.pdf [En línea] [Citado el: 29 de abril de 2015.]
37. http://catarina.udlap.mx/u_dl_a/tales/documentos/lep/mejia_s_ja/capitulo3.pdf [En línea] [Citado el: 30 de abril de 2015.]
38. Imformática Jurídica. [En línea] 2011. [Citado el: 30 de mayo de 2013.] http://www.informatica-juridica.com/trabajos/Ayuda_en_linea_para_el_Sistema_Sentai_Corporacion_importadores_exportadores_Cuba_Cimex.asp.
39. Extreme Testing. [En línea] 1999. [Citado el: 22 de marzo de 2013.] http://www.xprogramming.com/publications/SP99 Extreme for Web.pdf.
40. http://www.judavi.com/mvc-o-modelo-vista-controlador/
41. http://www.genbetadev.com/herramientas/netbeans-1
42. www.genbetadev.com/herramientas/eclipse-ide
43. http://www.taringa.net/posts/info/7538522/Qt-Creator-aun-no-lo-conoces-Entra.html

ANEXOS

Anexo 1: Historia de Usuario

Historia de Usuario	
Numero: 1	**Nombre:** Gestionar Proyecto
Usuario: Especialista	
Prioridad en Negocio: Alta	**Riesgo de Desarrollo:** Alto
Iteración Asignada: 1	**Puntos de Estimación:** 1
Descripción: El usuario podrá agregar, modificar, eliminar o ver los proyectos creados con los datos del mismo: nombre, descripción, imágenes, clases y configuración(es) asociadas al proyecto.	
Observaciones: Para modificar, eliminar o ver los proyectos estos deben haberse creado con anterioridad.	

Tabla 1. *Representación de la Historia de Usuario No. 1.*

Historia de Usuario	
Numero: 2	**Nombre:** Gestionar Clase
Usuario: Especialista	
Prioridad en Negocio: Alta	**Riesgo de Desarrollo:** Medio
Iteración Asignada: 1	**Puntos de Estimación:** 1
Descripción: El usuario podrá agregar, modificar, eliminar o ver las clases que fueron creadas dentro de un proyecto mostrándose los datos de la misma: nombre y descripción.	
Observaciones: Para modificar, eliminar o ver las clases disponibles estas deben haberse creado con anterioridad a cierto proyecto.	

Anexos

Tabla 2. *Representación de la Historia de Usuario No. 2.*

Historia de Usuario	
Numero: 3	**Nombre:** Gestionar Imagen
Usuario: Especialista	
Prioridad en Negocio: Alta	**Riesgo de Desarrollo:** Medio
Iteración Asignada: 1	**Puntos de Estimación:** 1
Descripción: El usuario podrá agregar, modificar, eliminar o ver las imágenes que fueron incluidas a un proyecto mostrándose los datos de la misma: ruta de la imagen dentro de la estación de trabajo.	
Observaciones: Para modificar, eliminar o ver las imágenes disponibles estas deben haberse asignado con anterioridad a cierto proyecto.	

Tabla 3. *Representación de la Historia de Usuario No. 3.*

Historia de Usuario	
Numero: 4	**Nombre:** Gestionar Configuración
Usuario: Especialista	
Prioridad en Negocio: Alta	**Riesgo de Desarrollo:** Alto
Iteración Asignada: 1	**Puntos de Estimación:** 2
Descripción: El usuario podrá agregar, modificar, eliminar o ver las configuraciones creadas con los datos del mismo: nombre, descripción, variables seleccionadas y vecindad.	
Observaciones: Para modificar, eliminar o ver las configuraciones estas deben haberse creado con anterioridad.	

Tabla 4. *Representación de la Historia de Usuario No. 4.*

Anexos

Historia de Usuario	
Numero: 5	**Nombre:** Gestionar Patrón
Usuario: Especialista	
Prioridad en Negocio: Alta	**Riesgo de Desarrollo:** Alto
Iteración Asignada: 2	**Puntos de Estimación:** 2
Descripción: El usuario podrá agregar, eliminar o ver los valores de las variables estadísticas calculadas a partir de las subimagen seleccionada tomando como referencia central el pixel seleccionado por el especialista, cuyos valores más tarde son los que conformaran el patrón pudiendo observar todos los valores seleccionados en la configuración.	
Observaciones: Para eliminar o ver los patrones estos deben haberse extraído antes.	

Tabla 5. *Representación de la Historia de Usuario No. 5.*

Historia de Usuario	
Numero: 6	**Nombre:** Gestionar Prueba
Usuario: Especialista	
Prioridad en Negocio: Alta	**Riesgo de Desarrollo:** Alto
Iteración Asignada: 2	**Puntos de Estimación:** 2
Descripción: El usuario podrá agregar, editar, eliminar o ver las pruebas con los datos de la misma: nombre, descripción, red neuronal asociada, e imagen.	
Observaciones: Para editar, eliminar o ver las configuraciones estas deben haberse creado con anterioridad.	

Tabla 6. *Representación de la Historia de Usuario No. 6.*

Anexo 2: Casos de pruebas unitarias

Anexos

Caso de prueba unitaria
Camino: 1
Caso de prueba: Existe imagen duplicada.
Entrada: Para un nombre de imagen no válido, recibido por parámetro, al recorrer la lista de imágenes, al menos uno de los elementos de la lista tiene que coincidir con el recibido por parámetro.
Resultado: Se devuelve la variable iRet con valor 1.
Condiciones:

Tabla 1. *Caso de prueba para el camino 1.*

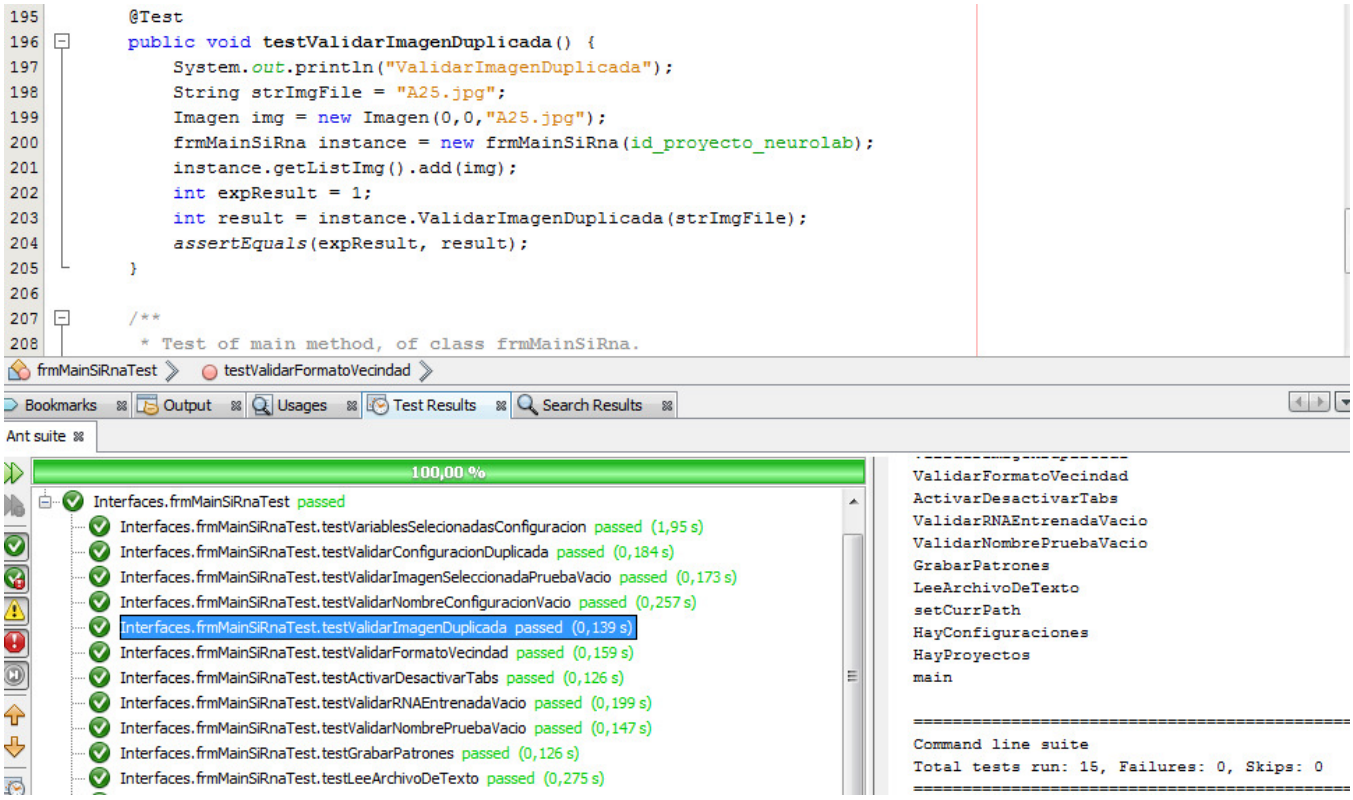


Figura 1. *Prueba unitaria realizada al camino 1.*

Caso de prueba unitaria
Camino: 2
Caso de prueba: No existe imagen duplicada
Entrada: Para un nombre de imagen válido recibido por parámetro, dentro de la lista de imágenes no puede existir ningún elemento con el nombre igual que al recibido por parámetro.
Resultado: Se devuelve la variable iRet con valor 0.
Condiciones:

Tabla 2. *Caso de prueba para el camino 2.*

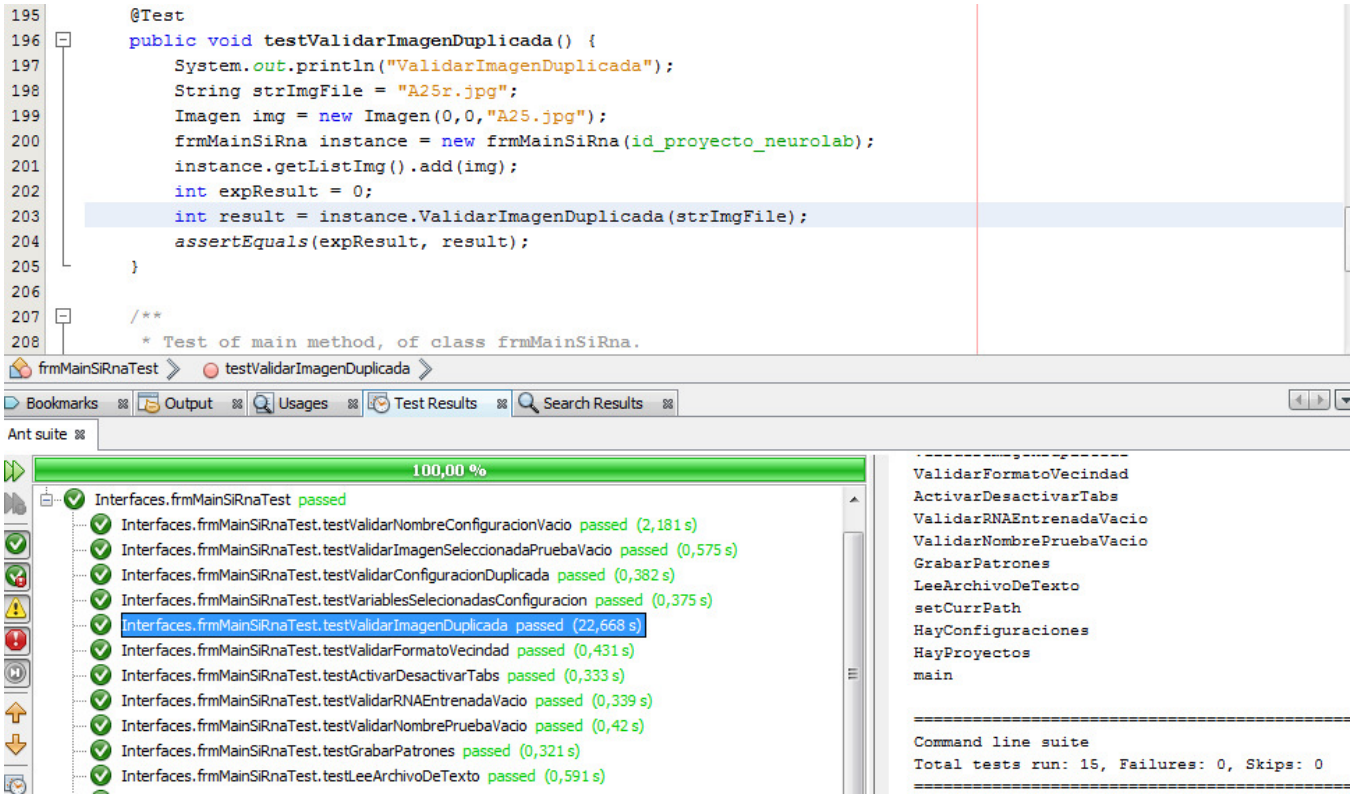


Figura 2. *Prueba unitaria realizada al camino 2.*

Caso de prueba unitaria

Camino: 3
Caso de prueba: No existen elementos en la lista a recorrer.
Entrada: Para un nombre válido de imagen recibido por parámetro, y la lista de imagen sin ningún elemento por recorrer.
Resultado: Se devuelve la variable iRet con valor 0
Condiciones:

Tabla 3. *Caso de prueba para el camino 3.*

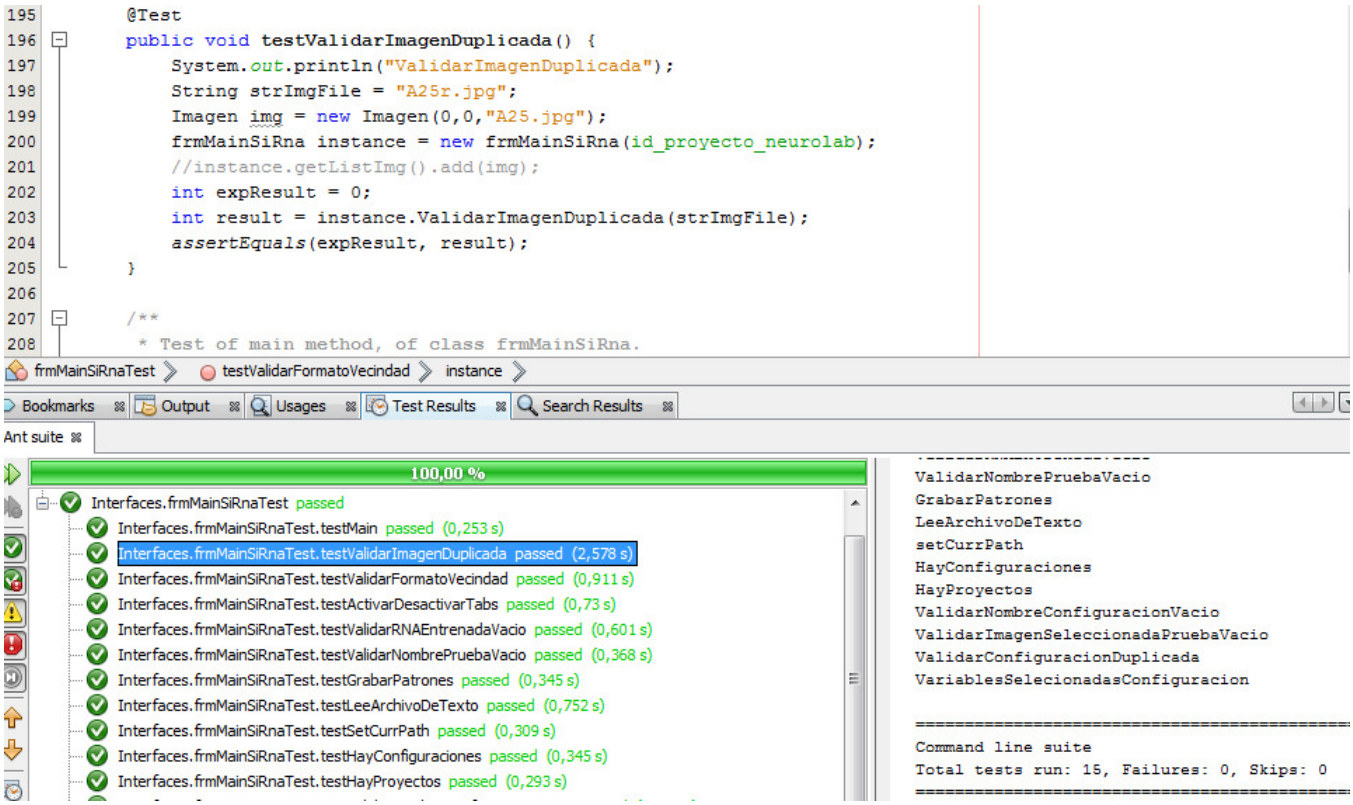


Figura 3. *Prueba unitaria realizada al camino 3.*

Anexos

Caso de prueba unitaria
Camino: 4
Caso de prueba: Nombre de archivo o lista de imágenes es NULL.
Entrada: Para un nombre de imagen o la lista de imágenes con valor igual a NULL.
Resultado: Se devuelve la variable iRet con valor -1.
Condiciones:

Tabla 4. *Caso de prueba para el camino 4.*

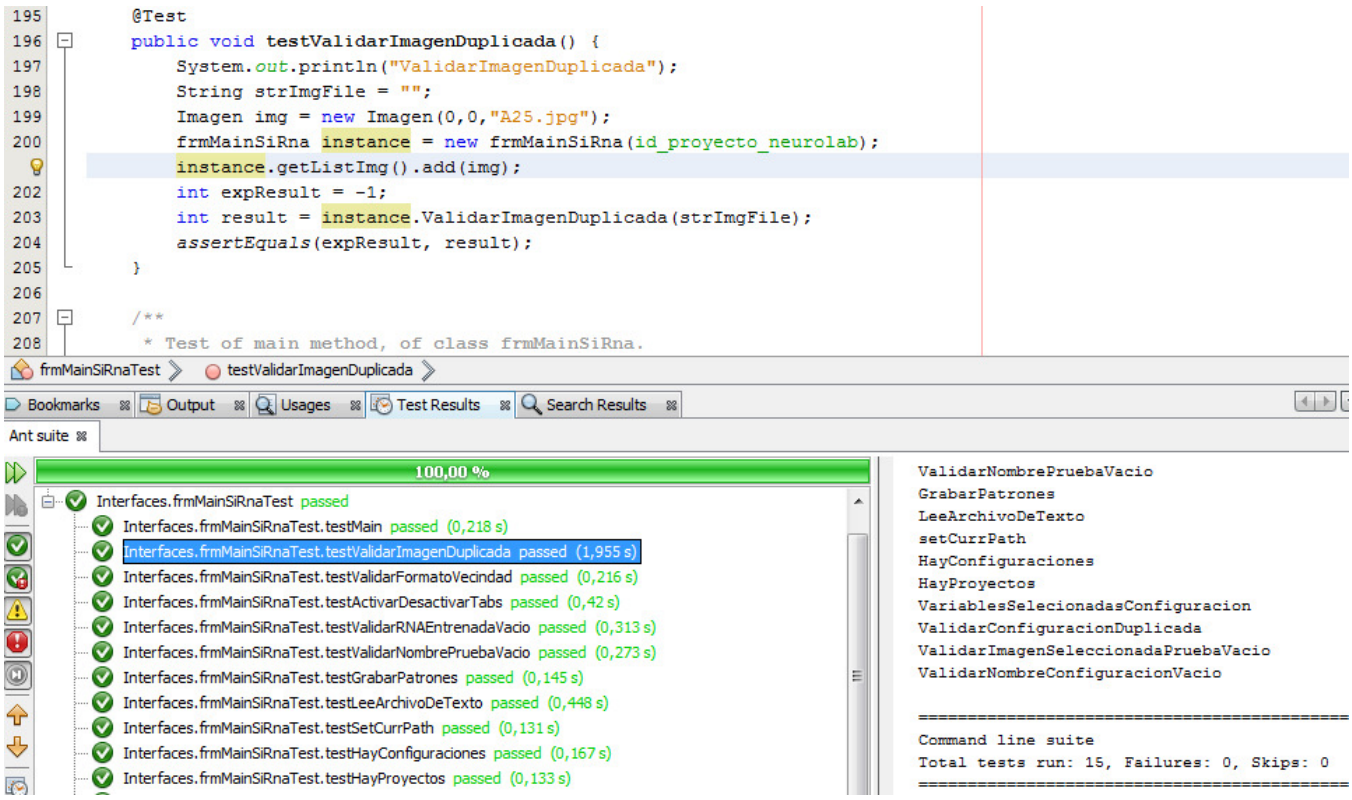


Figura 4. *Prueba unitaria realizada al camino 4.*

Anexo 3: Casos de prueba de aceptación.

Anexos

Iteración 1:

Caso de prueba de aceptación	
Código: HU_1_P1	**Historia de Usuario:** 1.
Nombre: Agregar Proyecto.	
Descripción: Prueba de funcionalidad para agregar un nuevo proyecto.	
Condiciones de Ejecución: Deben habilitarse todos los campos para introducir en ellos los datos necesarios del proyecto que se quiere agregar.	
Entrada/Pasos de Ejecución: El especialista selecciona la opción que le permite agregar un proyecto a través del botón habilitado a tales efectos. Este se encuentra ubicado en la parte inferior izquierda de la pantalla, debajo de la lista de proyectos. Al accionar en sobre este, se habilitan los componentes a llenar con los datos del proyecto.	
Resultado Esperado: Una vez introducidos los datos correctamente se agrega el nuevo proyecto satisfactoriamente.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 1. *Prueba 1 HU 1.*

Caso de prueba de aceptación	
Código: HU_1_P2	**Historia de Usuario:** 1.
Nombre: Modificar Proyecto.	
Descripción: Prueba de funcionalidad para modificar un proyecto.	
Condiciones de Ejecución: Se debe seleccionar de la lista de los proyectos, el que se desee modificar, para esto se debe haber creado con anterioridad como mínimo un proyecto.	

Entrada/Pasos de Ejecución: El especialista selecciona de la lista de proyectos disponibles, el que se desee modificar, y selecciona la opción de modificar, esta acción habilita todos los campos con los datos pertenecientes al proyecto en cuestión para ser modificados.
Resultado Esperado: Una vez seleccionado un proyecto se modifican los datos del mismo satisfactoriamente.
Evaluación de la Prueba: Prueba satisfactoria.

Tabla 2. *Prueba 2 HU 1.*

Caso de prueba de aceptación	
Código: HU_1_P3	**Historia de Usuario:** 1.
Nombre: Eliminar Proyecto.	
Descripción: Prueba de funcionalidad para eliminar un proyecto.	
Condiciones de Ejecución: Se debe seleccionar de la lista de los proyectos, el que se desee eliminar, para esto se debe haber creado con anterioridad como mínimo un proyecto.	
Entrada/Pasos de Ejecución: El especialista selecciona de la lista de proyectos disponibles, el que se desee eliminar, y selecciona la opción de eliminar.	
Resultado Esperado: Una vez seleccionado un proyecto se elimina el mismo satisfactoriamente.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 3. *Prueba 3 HU 1.*

Caso de prueba de aceptación	
Código: HU_2_P1	**Historia de Usuario:** 2.
Nombre: Adicionar Imagen(es).	

Descripción: Prueba de funcionalidad para adicionar imágenes a un proyecto.
Condiciones de Ejecución: Se debe seleccionar de la lista de los proyectos disponibles, aquel que se le desee adicionar nuevas imágenes o de lo contrario al adicionar un proyecto nuevo, se puede incluir nuevas imágenes, esto contemplado dentro de dicha acción.
Entrada/Pasos de Ejecución: El especialista al estar modificando o adicionando un nuevo proyecto tiene la posibilidad de adicionarle imágenes al mismo, por lo que al accionar sobre el botón de adicionar imagen, se visualizara una ventana que permite seleccionar las imágenes a incluir, y selecciona la opción agregar.
Resultado Esperado: Una vez seleccionadas las imágenes a incluir a un proyecto, estas son incluidas satisfactoriamente.
Evaluación de la Prueba: Prueba satisfactoria.

Tabla 4. *Prueba 1 HU 2.*

Caso de prueba de aceptación	
Código: HU_2_P2	**Historia de Usuario:** 2.
Nombre: Eliminar Imagen(es).	
Descripción: Prueba de funcionalidad para eliminar imágenes a un proyecto.	
Condiciones de Ejecución: Se debe seleccionar de la lista de los proyectos disponibles, aquel que se le desee eliminar las imágenes, para esto se debe haber creado con anterioridad como mínimo un proyecto con al menos una imagen.	
Entrada/Pasos de Ejecución: El especialista selecciona de la lista de proyectos aquel al que desee eliminar las imágenes, selecciona la opción modificar proyecto, habilitándose todos los componentes de la ventana con los datos del mismo, selecciona de la lista de imágenes las que desee eliminar y selecciona la opción eliminar imágenes.	
Resultado Esperado: Una vez seleccionadas las imágenes a eliminar de un proyecto, estas son eliminadas satisfactoriamente.	

Evaluación de la Prueba: Prueba satisfactoria.

Tabla 5. *Prueba 2 HU 2.*

Caso de prueba de aceptación	
Código: HU_3_P1	**Historia de Usuario:** 3.
Nombre: Adicionar Clase.	
Descripción: Prueba de funcionalidad para adicionar una clase a un proyecto.	
Condiciones de Ejecución: Se debe seleccionar de la lista de los proyectos disponibles, aquel que se le desee adicionar nuevas clases o de lo contrario al adicionar un proyecto nuevo, se puede incluir nuevas clases, esto contemplado dentro de dicha acción.	
Entrada/Pasos de Ejecución: El especialista al estar modificando o adicionando un nuevo proyecto tiene la posibilidad de adicionarle clases al mismo, por lo que al accionar sobre el botón de adicionar clase, se visualizara una ventana con los campos para introducir los datos de la clase que se va adicionar, y selecciona la opción aceptar.	
Resultado Esperado: Una vez mostrada la ventana e introducidos los datos de la misma, esta es adicionada satisfactoriamente.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 6. *Prueba 1 HU 3.*

Caso de prueba de aceptación	
Código: HU_3_P2	**Historia de Usuario:** 3.
Nombre: Modificar Clase.	
Descripción: Prueba de funcionalidad para modificar una clase de un proyecto.	

Condiciones de Ejecución: Se debe seleccionar de la lista de los proyectos disponibles, aquel que se le desee modificar una clase, para esto se debe haber creado con anterioridad como mínimo un proyecto con al menos una clase.
Entrada/Pasos de Ejecución: El especialista al estar modificando un proyecto tiene la posibilidad de modificar las clases del mismo, por lo que al accionar sobre el botón de modificar clase, se visualizara una ventana con los datos de la clase seleccionada para modificarlos, y selecciona la opción aceptar.
Resultado Esperado: Una vez mostrada la ventana e modificados los datos de la misma, esta es modificada satisfactoriamente.
Evaluación de la Prueba: Prueba satisfactoria.

Tabla 7. *Prueba 2 HU 3.*

Caso de prueba de aceptación	
Código: HU_3_P3	**Historia de Usuario:** 3.
Nombre: Eliminar Clase.	
Descripción: Prueba de funcionalidad para eliminar una clase de un proyecto.	
Condiciones de Ejecución: Se debe seleccionar de la lista de los proyectos disponibles, aquel que se le desee eliminar las clases, para esto se debe haber creado con anterioridad como mínimo un proyecto con al menos una clase.	
Entrada/Pasos de Ejecución: El especialista al estar modificando un proyecto tiene la posibilidad de eliminar las clases del mismo, por lo que al seleccionar una clase y accionar sobre el botón de eliminar clase, se visualizara un mensaje de confirmación para la eliminación de la misma, seleccionado la opción SI para confirmar su eliminación.	
Resultado Esperado: Una vez mostrado el mensaje de confirmación, esta es eliminada satisfactoriamente.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 8. *Prueba 3 HU 3.*

Anexos

Caso de prueba de aceptación	
Código: HU_4_P1	**Historia de Usuario:** 4.
Nombre: Adicionar Configuración.	
Descripción: Prueba de funcionalidad para adicionar una configuración a un proyecto.	
Condiciones de Ejecución: Se debe seleccionar de la lista de los proyectos disponibles, aquel que se le desee adicionar una configuración, para esto se debe haber creado con anterioridad como mínimo un proyecto.	
Entrada/Pasos de Ejecución: El especialista al tener seleccionado un proyecto, tiene la posibilidad de adicionar configuraciones, por lo que al accionar sobre el botón de adicionar configuración, se habilitaran todos los campos para la introducción de los datos, se introducen los datos y se acciona sobre el botón de aceptar.	
Resultado Esperado: Una vez mostrado el mensaje de confirmación, esta es adicionada satisfactoriamente.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 9. *Prueba 1 HU 4.*

Caso de prueba de aceptación	
Código: HU_4_P2	**Historia de Usuario:** 4.
Nombre: Modificar Configuración.	
Descripción: Prueba de funcionalidad para modificar una configuración de un proyecto.	
Condiciones de Ejecución: Se debe seleccionar de la lista de los proyectos disponibles, aquel que se le desee modificar cierta configuración, para esto se debe haber creado con anterioridad como mínimo un proyecto con al menos una configuración.	
Entrada/Pasos de Ejecución: El especialista al visualizar la lista de configuraciones, de un proyecto, tiene la posibilidad de seleccionar alguna configuración y modificar los datos de la	

misma, se visualizara un mensaje de confirmación para la modificación de la misma, seleccionado la opción SI para confirmar su modificación. Si para esa configuración existe al menos un patrón extraído, no va a poder ser modificado
Resultado Esperado: Una vez mostrado el mensaje de confirmación, esta es modificada satisfactoriamente.
Evaluación de la Prueba: Prueba satisfactoria.

Tabla 10. *Prueba 2 HU 4.*

Caso de prueba de aceptación	
Código: HU_4_P3	**Historia de Usuario:** 4.
Nombre: Eliminar Configuración.	
Descripción: Prueba de funcionalidad para eliminar una configuración de un proyecto.	
Condiciones de Ejecución: Se debe seleccionar de la lista de los proyectos disponibles, aquel que se le desee eliminar cierta configuración, para esto se debe haber creado con anterioridad como mínimo un proyecto con al menos una configuración.	
Entrada/Pasos de Ejecución: El especialista al estar en la vista donde está la lista de configuraciones tiene la posibilidad de eliminar las configuraciones, por lo que al seleccionar una configuración y accionar sobre el botón de eliminar configuración, se visualizara un mensaje de confirmación para la eliminación de la misma, seleccionado la opción SI para confirmar su eliminación.	
Resultado Esperado: Una vez mostrado el mensaje de confirmación, esta es eliminada satisfactoriamente.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 11. *Prueba 3 HU 4.*

Anexos

Caso de prueba de aceptación	
Código: HU_5_P1	**Historia de Usuario:** 5.
Nombre: Adicionar Patrón.	
Descripción: Prueba de funcionalidad para adicionar un patrón.	
Condiciones de Ejecución: Se debe haber creado con anterioridad una configuración, e incluirle al proyecto en cuestión las imágenes y las clases, seleccionar la imagen y la clase que va a ser identificadas dentro de la imagen, accionar sobre zonas determinadas de la imagen, esto desencadena el cálculo de las variables seleccionadas en la configuración en una vecindad definida.	
Entrada/Pasos de Ejecución: El especialista al extraer los patrones debe haber seleccionado una imagen y la clase a la que pertenece el patrón, que se va a extraer. Se realiza este proceso cuantas veces sea necesario y decida el usuario, hasta extraer la mayor cantidad de patrones, que puedan identificar todos los eventos contenidos en la imagen	
Resultado Esperado: El accionar sobre determinadas zonas dentro de la imagen se extrae satisfactoriamente el patrón.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 12. *Prueba 1 HU 5.*

Caso de prueba de aceptación	
Código: HU_5_P2	**Historia de Usuario:** 5.
Nombre: Eliminar Patrón.	
Descripción: Prueba de funcionalidad para eliminar un patrón.	
Condiciones de Ejecución: Debe existir con anterioridad al menos un patrón a eliminar.	
Entrada/Pasos de Ejecución: El especialista después de haber extraído al menos un patrón, este puede ser seleccionado y eliminado.	

Resultado Esperado: Al seleccionar uno o un conjunto de los patrones extraídos, y accionar sobre el botón eliminar, son eliminados satisfactoriamente.
Evaluación de la Prueba: Prueba satisfactoria.

Tabla 13. *Prueba 2 HU 5.*

Caso de prueba de aceptación	
Código: HU_6_P1	**Historia de Usuario:** 6.
Nombre: Adicionar Prueba.	
Descripción: Prueba de funcionalidad para adicionar una prueba.	
Condiciones de Ejecución: Debe haberse realizado con anterioridad la extracción de patrones, y haber entrenado una red neuronal con esos patrones para empezar con el reconocimiento de patrones dentro de una imagen original y otras con similares características a la que se extrajo los patrones.	
Entrada/Pasos de Ejecución: El especialista después de haber extraído lo patrones necesarios y haber entrenado la red con dichos patrones, entonces se adicionar una prueba, especificando la imagen a la que se le va a realizar el reconocimiento interactivo de patrones, se le asocia cierta red neuronal entrenada y se procede a accionar sobre la imagen mostrada e identificar ciertos eventos contenidos dentro de la imagen.	
Resultado Esperado: Al crear una prueba e incluirle a esta la imagen y la red neuronal entrenada, se añade satisfactoriamente.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 14. *Prueba 1 HU 6.*

Caso de prueba de aceptación	
Código: HU_6_P2	**Historia de Usuario:** 6.
Nombre: Editar Prueba.	
Descripción: Prueba de funcionalidad para editar una prueba.	
Condiciones de Ejecución: Debe haberse añadido anteriormente una prueba	
Entrada/Pasos de Ejecución: El especialista al editar una prueba tiene la posibilidad de realizar el reconocimiento de patrones y con esto si lo desea cambiar los datos de la prueba como puede ser nombre, descripción, red neuronal entrenada e imagen, al terminar este proceso, puede salvar los cambios realizados satisfactoriamente.	
Resultado Esperado: Al editar una prueba y realizar los cambios que desee, esta se edita satisfactoriamente.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 15. *Prueba 2 HU 6.*

Caso de prueba de aceptación	
Código: HU_6_P3	**Historia de Usuario:** 6.
Nombre: Eliminar Prueba.	
Descripción: Prueba de funcionalidad para eliminar una prueba.	
Condiciones de Ejecución: Debe haberse añadido anteriormente una prueba	
Entrada/Pasos de Ejecución: El especialista al eliminar una prueba, debe seleccionarla de la lista de pruebas, y accionar sobre el botón eliminar prueba.	
Resultado Esperado: Al eliminar una prueba, esta se elimina satisfactoriamente.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 16 *Prueba 3 HU 6.*

Anexos

Anexo 4: Tablas de la base de datos.

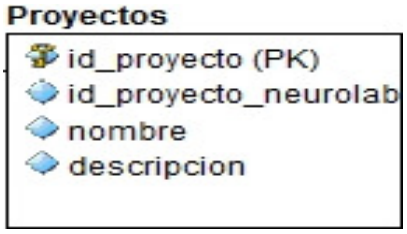


Figura 1. *Tabla Proyectos.*

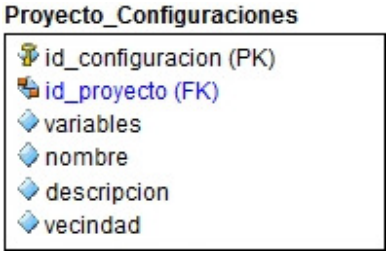


Figura 2. *Tabla Configuraciones.*

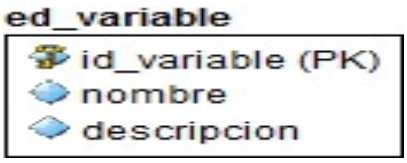


Figura 3. *Tabla ed_variables.*

Figura 4. *Tabla Patrones.*

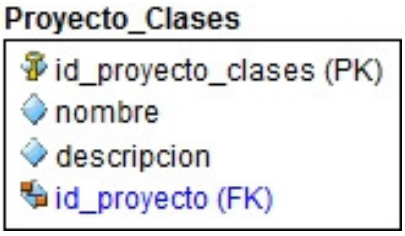


Figura 5. *Tabla Proyecto_Clases.*

Figura 6. *Tabla Proyecto_Imagenes.*

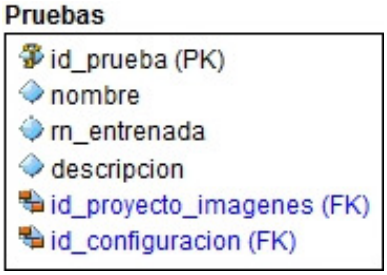


Figura 7. *Tabla Pruebas.*

Anexo 5: Carta de aceptación.

softel
soluciones informáticas

10 de junio de 2015

A quien pueda interesar:

Por medio de la presente hacemos constar que la herramienta SIRNA, la cual fue el objeto del Trabajo de Diploma del alumno Ernesto Hita Ramos cumple con todas las exigencias que le fueron definidas por la Empresa. Todas las pruebas realizadas fueron satisfactorias. El alumno, durante su trabajo demostró que puede realizar trabajo como ingeniero de forma independiente. Mantuvo buenas relaciones con todo el personal.

Sin más.

softel
Gestión del
Conocimiento

Alfredo Sánchez Rodríguez
Director de Desarrollo
SOFTEL

Anexo 6: Tarjetas CRC.

Proyecto	
Añadir Proyecto Eliminar Proyecto Modificar Proyecto Mostrar Proyecto	

Tabla 1 *CRC Proyecto.*

Imagen	
Añadir Imagen Eliminar Imagen Modificar Imagen Mostrar Imagen	Proyecto

Tabla 2 *CRC Imagen.*

Clase	
Añadir Clase Eliminar Clase Modificar Clase Mostrar Clase	Proyecto

Tabla 3 *CRC Clase.*

Configuración	
Añadir Clase Eliminar Clase Modificar Clase Mostrar Clase	Proyecto

Tabla 4 *CRC Configuración.*

Patrón	
Añadir Patrón Eliminar Patrón Mostrar Patrón Calculo de variables	Configuración Proyecto

Tabla 5 *CRC Patrón.*

Prueba

Añadir Prueba	Proyecto
Eliminar Prueba	Imagen
Editar Prueba	Clase
Mostrar Prueba	Configuración
Reconocer Patrones	

Tabla 6 *CRC Prueba*