Departamento de Computación
Facultad de Informática
UNIVERSIDADE DA CORUÑA

TRABAJO FIN DE GRADO
GRADO EN INGENIERÍA INFORMÁTICA
MENCIÓN EN COMPUTACIÓN

Herramienta para el etiquetado de tráfico de red basada en agrupamiento no supervisado

Estudiante: Mario Iglesias Otero
Directores: José Carlos Dafonte Vázquez
Manuel Fernández López-Vizcaíno

A Coruña, 6 de septiembre de 2019.

A mis abuelos Julia y Pepe y a Sara

Agradecementos

A mis abuelos dos personas que me han acompañado durante toda mi vida y me han apoyado en los buenos y en los malos momentos.

A Sara por llenar de magia y alegría todos los momentos a su lado.

A mi tutores José Carlos Dafonte Vázquez y Manuel Fernández López-Vizcaíno darme la oportunidad para realizar este proyecto y por haberme guiado, aconsejado y enseñado tanto durante todo camino. Si no fuera por vosotros esto no sería posible.

A todos vosotros y a muchos otros *muchísimas gracias.*

Mario Iglesias Otero

Resumen

El gran volumen de información que circula por las redes de comunicaciones así como la necesidad de mantener un nivel adecuado de seguridad hacen necesario el análisis del tráfico de red. El presente Trabajo de Fin de Grado (TFG) aborda la creación de una herramienta que permita identificar tráfico de red malicioso basándose en aprendizaje no supervisado e información presente en las cabeceras de los paquetes de red.

El objetivo principal de este trabajo es la obtención de una aplicación que permita la exploración y análisis de mapas auto-organizados. Así como la creación, modificación y eliminación de etiquetas asociadas a cada una de sus neuronas. Dentro de ella se incluye la visualización de la distribución de los datos en cada neurona y un análisis estadístico de los datos almacenados.

Como objetivo secundario hemos realizado un análisis de un conjunto de datos para seleccionar la información más significativa a la hora de determinar si un mensaje es peligroso para el sistema. Basándonos en los resultados hemos generado un conjunto en el formato adecuado para la aplicación de mapas auto organizados.

Abstract

The large volume of information in communication networks as well as the adequate level of security needed require to analyze network traffic. This end-of-degree project presents a tool based on unsupervised learning techniques and information present in network packets headers.

The main objective of this project is the development of a tool to explore and analyze self organized maps. And the creation, modification and the removal of labels for each neuron. Also a visualization that allow us to observe the distribution of the data inside each neuron and an statistical analysis of data is included.

Besides, we have performed an analysis to decide which is the most significant information when determining if a message is dangerous for a system. Based on this results we have generated a dataset in the format required by the self organized maps tool.

Palabras clave:

Inteligencia Artificial, Big Data, Mapas autoorganizativos, Aplicación web, Visualización

Keywords:

Artificial Intelligence, Big Data, SelfOrganized Maps, Web application, Visualization System

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

Hoy en día, a través de las redes de comunicaciones circula una inmensa cantidad de datos digitales y se espera que esta cantidad continúe aumentando [1]. Debido a esta situación, la monitorización de las redes se convierte en una parte esencial de los sistemas de seguridad para controlar su correcto funcionamiento y asegurar un nivel adecuado de seguridad [2] [3].

Como respuesta a este problema, en los últimos años, se han desarrollado sistemas de monitorización basados en SIEM (Security Information & Event Management). Estos sistemas son capaces de reunir información de diferentes fuentes pero, una vez almacenados, estos datos deben ser procesados para detectar ataques u otro tipo de anomalías.

Así mismo, este volumen incrementa de manera exponencial la complejidad del análisis y dificulta la detección de ataques u otro tipo de anomalías de manera automatizada. Una posible aproximación al análisis de esta información es el agrupamiento no supervisado. Estos métodos, que han demostrado su efectividad en investigaciones previas [4], permiten generar grupos de manera automática basándose en características similares de los datos.

Debido a esto, resulta interesante agrupar los datos y posteriormente etiquetarlos, generando conjuntos que permitan su análisis y la aplicación de técnicas supervisadas de Inteligencia Artificial (IA).

Debido al problema de análisis y visualización de la información existente en entornos *Big Data* se propone el desarrollo de un sistema que permitirá el estudio de los resultados de la aplicación de una técnica no supervisada de agrupamiento. Así mismo este sistema permitirá el etiquetado manual de las neuronas para su posterior análisis.

Antes de realizar el desarrollo de la herramienta de etiquetado es necesario disponer de un conjunto de datos apropiado que permita al sistema generar agrupaciones, sobre los cuales se puedan aplicar técnicas supervisadas de IA. El conjunto seleccionado fue *ISCX 2017* [5], generado a través de simulaciones realizadas por el *Canadian Institute for Cybersecurity*. Existen trabajos previos realizados en el grupo de investigación en los que se empleaba el conjunto *ISCX 2012*, publicado anteriormente por esta misma institución.

Uno de los mayores problemas planteados para la realización de este proyecto es agilizar todos los cálculos necesarios para la visualización de mapas auto-organizativos en un entorno de *Big Data*. De esta manera y para conseguir unos tiempos de respuesta ajustados, planteamos la implementación de mecanismos de precálculo para las funciones de visualización más habituales, facilitando además el proceso de análisis y etiquetado.

Seleccionado el conjunto de datos, comenzaremos con la implementación del proyecto, este se divide en una etapa de desarrollo para de cada uno de los objetivos planteados.

1.1 Descripción de la memoria

En este apartado realizaremos una breve explicación de cada uno de los capítulos que componen la memoria, indicando lo abordado en cada uno de ellos.

- **Capítulo 1: Introducción**. En el presente capítulo se hará una breve recapitulación de los distintos apartados del proyecto, así como de los objetivos que se quieren llevar a cabo.
- **Capítulo 2: Análisis de antecedentes.** Se presentan las diferentes alternativas que se han tenido en cuenta en la realización de este proyecto.
- **Capítulo 3: Estudio de viabilidad.** Trata sobre la motivación que lleva al desarrollo del proyecto, centrándose en las razones que llevan al desarrollo de la herramienta de etiquetado y visualización de conjuntos de datos.
- **Capítulo 4: Metodología.** Durante todo este proyecto se ha aplicado una modificación de la metodología Scrum, en este apartado se describe la misma y cuales son las modificaciones realizadas.
- **Capítulo 5: Tecnologías.** Breve mención de las herramientas de software utilizadas durante el proyecto y que facilitarán la realización del mismo.
- **Capítulo 6: Planificación y evaluación de costes.** Se indican los materiales utilizados durante el proyecto, cual es la planificación inicial del mismo y una estimación del coste total del proyecto.
- **Capítulo 7: Conceptos previos.** Se exponen los fundamentos teóricos en los que se basa el proyecto o que se utilizarán en él y daremos una visión profunda de los conceptos que deben conocerse para afrontar la realización del mismo.
- **Capítulo 8: Análisis de datos.** En este capítulo se describe el conjunto de datos con el que se trabaja, los distintos tipos de datos presentes y como se trata cada uno de ellos.

- **Capítulo 9: Desarrollo.** En este capítulo se describen las distintas funcionalidades de la aplicación y como se ha abordado su desarrollo.
- **Capítulo 10: Pruebas.** Descripción del conjunto de pruebas realizadas para validar el correcto funcionamiento de las herramientas desarrolladas.
- **Capítulo 11: Conclusiones.** Se expone el trabajo realizado durante el proyecto y se revisan los objetivos inicialmente planteados para contrastarlos con el trabajo realizado.
- **Capítulo 12: Trabajo futuro.** Se mencionan algunas posibles ampliaciones o modificaciones que se podrían aplicar sobre el proyecto desarrollado.
- **Glosario de acrónimos.** Contiene el significado de todos los acrónimos utilizados a lo largo de la memoria.

1.2 Objetivos

La finalidad de este proyecto es desarrollar una herramienta que permita el etiquetado de los grupos generados mediante la técnica de mapas auto-organizados (Self Organized Maps, SOM en sus siglas en inglés), la visualización de la distribución de los datos y un análisis estadístico para cada conjunto. Además permitirá añadir, modificar, eliminar y seleccionar el color que se desea que represente a cada etiqueta. Todo esto permitirá generar diferentes etiquetado externo para cada SOM.

El grupo de investigación ha desarrollado un sistema de visualización de mapas auto-organizados sobre el cual insertaremos nuestra herramienta. Con el fin de utilizarla sobre el conjunto seleccionado, se realizará un análisis previo del mismo con la finalidad de obtener unos resultados con el contenido y en el formato adecuado. Tras el procesado mediante la técnica SOM se podrá visualizar en la herramienta y proceder a su etiquetado.

Partiendo de las ideas indicadas, hemos desarrollado una lista de objetivos concretos:

1. Desarrollo de una herramienta de visualización de mapas auto-organizados que permitirá realizar una exploración de los mismos en la que se podrá estudiar como se distribuyen cada uno de los grupos generados.
2. Análisis del conjunto de tráfico de red para su utilización en las siguientes fases. Debido a la gran cantidad de datos con la que se va trabajar es necesario aplicar técnicas de *Big Data* para realizar un filtrado y seleccionar únicamente los datos de interés.
3. Implementar una herramienta que permita crear, actualizar y eliminar etiquetas en tiempo real, esto puede conllevar la modificación de cientos de neuronas al mismo tiempo.

4. Mejorar la usabilidad del sistema mediante un menú contextual que permita realizar acciones como el etiquetado o visualización de cada neurona.

5. Crear una nueva visualización, que muestre un análisis estadístico acerca de cada neurona.

6. Plantear la aplicación de técnicas de IA supervisadas sobre los conjuntos de datos etiquetados obtenidos mediante funcionalidades incluidas en la herramienta.

7. Desarrollar una visualización en la que se muestre la dispersión de los datos de cada neurona. Es necesario desarrollar algoritmos de simplificación de información que permitan mostrar en tiempo real grandes cantidades de datos.

Capítulo 2

Análisis de antecedentes

Durante los últimos años han surgido diferentes estudios relacionados con sistemas inteligentes, que proponen nuevas técnicas para aplicar sobre ellos y obtener distintos resultados dependiendo de las necesidades del proyecto. Mientras la gran mayoría de sistemas inteligentes crecían en gran medida, las técnicas basadas en SOM se quedaban estancadas, y como consecuencia apenas podemos encontrar estudios realizados sobre estas o herramientas que permitan su visualización y/o modificación. En este apartado hablaremos de algunas de estas aplicaciones.

En el desarrollo de este proyecto demostraremos la utilidad de la SOM para la agrupación no supervisada de datos así como para agilizar su etiquetado de manera manual obteniendo unos resultados significativamente buenos y demostrando que gracias a las nuevas herramientas de desarrollo podemos plantear una base firme con la que defender el uso de la SOM.

2.1 Viscovery SOMine

Esta herramienta[6] es la que tiene más similitudes con nuestra aplicación. Pertenece a la empresa Viscovery que está siendo apoyada por algunas de las multinacionales más importantes del mundo como son VISA, Orange o Allianz también está respaldada por distintas universidades de renombre.

Este programa consta de 4 partes, un núcleo imprescindible para su funcionamiento y 3 extensiones con distintas funcionalidades cada una. El núcleo permite agrupar los datos en función de patrones que los asocian y visualizar la distribución de los datos y sus estadísticas. Las 3 extensiones permiten: exploración avanzada del mapa y clasificación de los resultados obtenidos; realización de predicciones en función de los datos almacenados y un modulo que permite trabajar con grandes cantidades de información. Esta última extensión es la más similar a nuestro software ya que permite trabajar con grandes con juntos de datos y visualizarlos de forma interactiva, observar como cambian las regiones al introducir nuevos datos y crear,

modificar y generar en tiempo real estadísticas de los conjuntos. Se trata de un software de pago con una licencia básica de 490€ y un precio de hasta 1490€.

La gran diferencia es que nuestra aplicación tiene como base una arquitectura cliente-servidor sobre una interfaz web lo que permite un sistema distribuido más accesible sin necesidad de instalar ningún software. Además esta herramienta no permite la modificación de los datos de forma manual tras la ejecución del conjunto.

2.2 SOM Toolbox

Conjunto de librerías de MATLAB[7] que se han desarrollado para crear SOMs con diferentes topologías de red y visualizar SOMs utilizando planos de componentes, codificación de colores de agrupación y enlace de colores entre el SOM(Ver [8]).

Este software ha caído en desuso debido a sus limitaciones en las visualizaciones ya que no permite realizar una análisis en profundidad del conjunto de datos además es necesario tener una licencia de MATLAB para utilizarlo, lo que conlleva un alto coste.

2.3 SOMPY

En los últimos años Python ha tenido una gran acogida como lenguaje de programación de alto nivel para computación científica, permite acelerar código a través de diferentes métodos como el uso de Cython[9], programación paralela en la nube, procesamiento multi-core y muchas operaciones realizadas con matrices.

Este gran impulso que ha recibido ha llevado a que se desarrollara la librería SOMPY[10], para el que se han creado un conjunto de funcionalidades para mapas SOM. Algunas de estas nueva operaciones pueden resultar muy interesantes como puede ser el entrenamiento por lotes que permite acelerar el algoritmo y hacerlo escalable.Se podría decir que pretende cubrir las mismas necesidades que SOMToolbox de Matlab y solventar algunos de sus problemas como la escalabilidad aportándole un toque innovador que la convierten en una herramienta muy completa.

2.4 Databionic ESOM Tools

Conjunto de herramientas desarrollado para Java por el grupo de desarrollo Databionics Research Group[11] de la universidad de Marburg, Alemania. Databionic ESOM Tools[12] permiten realizar tareas de minería de datos, como agrupamiento, visualización y clasificación con Emergent Self-Organizing Maps (ESOM).

Durante el entrenamiento de mapas con diferentes inicializaciones permite visualizar la información en distintos formatos, tener una visualización interactiva del proceso de asignación de datos, clasificación y una descripción de los mismos entre otras funcionalidades.

Capítulo 3

Estudio de viabilidad

Este proyecto surge a raíz de una aplicación que actualmente se encuentra en desarrollo en un grupo de investigación en el cual han planteado la realización de un TFG con la intención de ampliarla. La aplicación permite visualizar como se han agrupado los datos que introducimos en una SOM y aplicarle distintos filtros para observar patrones.

Se ha decidido ampliar esta aplicación desarrollando un conjunto de herramientas que mejoren su funcionamiento. Para disponer de un entorno adecuado se ha decidido seleccionar un conjunto de datos sobre los que se realiza un análisis, se obtienen unos resultados y a partir de ellos se decide que información introduciremos en la SOM para su procesado y agrupación y con ellos poder observar el funcionamiento da las herramientas que vamos a desarrollar.

Los motivos por los que el desarrollo de este proyecto se estima viable se describen en los siguientes apartados.

3.1 Viabilidad técnica

En el apartado Tecnologías podremos encontrar todas las tecnologías con las que vamos a trabajar. La mayoría de ellas ya eran conocidas por haberlas utilizado anteriormente:

- Como sistema de control de versiones hemos seleccionado Git[13] con soporte GitHub[14]. Ya se había trabajado anteriormente con esta sistema, además es sencilla de utilizar y podemos asegurar su correcto funcionamiento ya que está respaldada por una gran empresa como Microsoft[15].

- Para realizar un plan de proyecto y hacer un seguimiento se ha utilizado Microsft Project[16] una herramienta muy completa, ya conocida y al igual que en el punto anterior, también cuenta con el soporte de Microsoft.

- En la realización del diseño se ha utilizado otra herramienta de la que ya se tenían conocimientos previos, MagicDraw[17], que esta especializada en la creación de diagramas

para la realización de aplicaciones.

- Para el desarrollo de la parte análisis de los datos se utilizará Python, un lenguaje ya conocido, junto con una librería especializada en trabajar con BigData llamada Pandas[18], el grupo de investigación ya ha lo ha utilizado anteriormente para desarrollos similares, pero es una librería completamente nueva para el desarrollador.
- La aplicación sobre la que implementaremos nuevas funcionalidades está desarrollada en Java y JS y trabaja con un servidor REST, ya se tiene conocimiento de ambos lenguaje y la arquitectura, por tanto no representan un problema. Cada uno de estos lenguajes se está utilizando con librerías que simplifican la creación de una interfaz web.
  - Vaadin es una librería de Java completamente nueva para el desarrollador pero al conocer el lenguaje facilita su aprendizaje.
  - D3JS[19] y VisJS[20] son dos librerías de JS para visualizar conjuntos de datos, ambas son nuevas para el desarrollador pero de una complejidad de aprendizaje baja.
  - También es necesario utilizar HTML y CSS con JS ya que no se pueden insertar directamente, esto es simple ya que el desarrollador ya tiene los conocimientos necesarios. La complejidad reside en la unión del código JS con Vaadin que supone un enfoque novedoso.

  Teniendo en cuenta que la mayoría de tecnologías ya son conocidas y las nuevas no conllevan demasiada dificultad, la mayor complejidad que podría existir acerca de este apartado sería el desarrollo de los algoritmos de tratamiento de datos en tiempo real que vamos a crear para implementar nuevas funcionalidades que no deben entorpecer o ralentizar la aplicación.

  Desde el punto de vista de la tecnología podríamos decir que es viable ya que los conocimientos adquiridos durante la carrera son suficientes para poder crear algoritmos que aseguren la eficacia y eficiencia de las nuevas herramientas.

3.2 Viabilidad económica

Esta herramienta surge de la necesidad de modificar manualmente el etiquetado externo de un conjunto de neuronas. Actualmente la aplicación a la que vamos a incorporar esta funcionalidad se está utilizando en distintos departamentos de la universidad para visualizar la distribución de datos en redes de neuronas artificiales. Debido a que la automatización del proceso de etiquetado externo no siempre es correcta, se desea poder modificar, insertar y eliminar etiquetas para subsanar estos errores.

En el apartado section 6.2 hacemos un análisis a fondo de los costes del proyecto. Los costes que se indican no son reales ya que el mayor de ellos es el analista programador que en el caso real es 0€ y los directores de proyecto, debido a que ya forman parte de la universidad, su coste también es de 0€.

Sabiendo que su viabilidad de cara al publico surge de una necesidad y que su coste sería mínimo podemos decir que es viable económicamente.

3.3 Viabilidad temporal

Teniendo en cuenta las posibles fechas de finalización del proyecto y la gran magnitud de trabajo que este conlleva se ha decidido tener como fecha de entrega de este el 6 de Septiembre de 2019. Pero como podemos observar en apartado section 6.1 la fecha de finalización es 6 de Julio de 2019. Esto se debe a que se quiere dejar un margen amplio por si se diera el caso de que hubiera muchos inconvenientes que ralentizaran de forma notable el proyecto, algo que no sería de extrañar en un proyecto que conlleva una parte de investigación y tantos periodos de aprendizaje.

Dado este margen tan amplio y que la intención es finalizar el provecto antes de la fecha acordada (6 de Julio de 2019) se puede decir que es viable temporalmente.

Capítulo 4

Metodología

Este proyecto se ha desarrollado bajo una metodología ágil para el desarrollo de software, que se ha llevado a cabo de manera incremental. Partiendo de una solución simplificada y añadiendo nuevas características en cada iteración hasta lograr el objetivo, implementar por completo el sistema definido.

En las metodologías clásicas[21] se comienza fijando un objetivo y desglosándolo en tareas. Son predicativas y suelen utilizarse para aquellos proyectos con un resultado conocido porque ya se ha realizado previamente. Si se trata de repetir un trabajo realizado buscando mejoras debería usarse una de estas metodologías. Para casos en los que los objetivos pueden cambiar durante el desarrollo tienen una peor aceptación. Las metodologías ágiles en cambio se ajustan correctamente a modificaciones durante el desarrollo del proyecto debido a que usan planificaciones iterativas y en espiral.

Para este proyecto es necesario realizar un periodo de investigación y análisis de datos previo al desarrollo de la herramienta, ya que en las asignaturas del grado no se ha profundizado en el conocimiento acerca de *Big Data* y el preprocesado de datos.

Además de esto, aunque los lenguajes de programación sí son conocidos, todas la tecnologías usadas son nuevas. Todo esto implica periodos de aprendizaje e investigación difíciles de estimar puesto que su duración depende de la curva de aprendizaje. Por ello cabe la posibilidad de cambios en el calendario del ciclo de desarrollo. Para poder realizar estas modificaciones, así como para asumir posibles variaciones en los requisitos, se necesita una metodología de desarrollo ágil.

De entre las metodologías ágiles se ha decidido utilizar un desarrollo basado en Scrum introduciéndole modificaciones, ya que este se ajusta en gran medida al proyecto, pero no cumple todos los requisitos necesarios para poder aplicarlo. Es una metodología adecuada para proyectos en los que pueden surgir cambios continuamente, esto puede darse por diversos motivos como no conocer completamente la tecnología o encontrar limitaciones hardware.

Scrum[21][22] significa literalmente melé de rugby. Toma esta metáfora deportiva porque

el objetivo de éste sistema de gestión de proyectos es que todo el equipo empuje en la misma dirección. Este modelo fue definido por Ikujiro Nonaka y Takeuchi a principios de los 80, al observar cómo las empresas de manufactura tecnológica desarrollaban los nuevos productos. Se ha consolidado como el método ágil más utilizado por permitir aplicarse a toda clase de proyectos.

4.1 Terminología

La metodología Scrum emplea una serie de términos propios para poder determinar sus distintos elementos, en este apartado se describirán los distintas partes que lo forman y las modificaciones que se han llevado a cabo sobre ellos. Scrum se basa en el concepto de equipo, este estará dividido en tres roles, cada componente del equipo tendrá asignado uno de estos roles. Cada rol describe la función que desempeñará cada componente del equipo. Los roles son tres: Product Owner o dueño del producto, Scrum Master y el Equipo de Trabajo. La función de estos se detalla en el apartado Roles.

Esta metodología utiliza una serie de iteraciones constantes llamadas Sprints. Estos son la clave de esta metodología; son un marco temporal en el que se realizan los trabajos y tienen una duración de entre una y cuatro semanas. Hemos modificado la duración máxima de estos, ampliándola hasta 6 semanas. Esto se debe a que hay periodos de aprendizaje y de análisis demasiado largos y resulta complicado llevarlos a cabo en un tiempo menor. Cada una de estas etapas sigue una secuencia de reuniones cuyo objetivo es garantizar el cumplimiento de los compromisos del equipo. Además se debe fijar un calendario de entregas en las que ofrecer al cliente un resultado parcial antes de ser completado.

Cada una de las tareas a realizar se introducirá en una lista de objetivos priorizada. Esta recibe el nombre de Product Backlog y nos define todo lo que vamos a realizar en el conjunto del proyecto y va sufriendo cambios a medida que evoluciona.Este se desglosa en tareas más pequeñas para realizar en el periodo establecido de cada Sprint. Debe revisarse la priorización de estas tareas antes del comienzo de cada Sprint.

Los diagramas que representan una serie temporal del trabajo pendiente se les denomina Burn down chart[23]. En el eje vertical encontramos el trabajo pendiente y en el horizontal el tiempo. Existen dos tipos de gráficas, las relacionadas con el Sprint y las relacionadas con la totalidad del proyecto.

4.2 Roles

Product Owner: crea y modifica las tareas del Product Backlog y asegura que todos sepan qué hay en él y cuáles son las prioridades. Puede ser ayudado por otros individuos pero el rol

debe ser ocupado por una única persona. Se asegura de que el equipo trabaje de forma adecuada desde la perspectiva del negocio. Representa al cliente y actúa como intermediario entre este y el usuario. Ayuda al usuario a escribir las historias de usuario[24]. Es el responsable del éxito o fracaso del producto y de la rentabilidad del proceso.

Scrum Master: denominado comúnmente ”facilitador”, su labor consiste en que todos los miembros del equipo conozcan la metodología y la gestión de trabajo adecuadamente. Su trabajo es eliminar obstáculos para poder conseguir el objetivo del Sprint. Su función principal es mejorar la productividad del equipo favoreciendo la comunicación y la colaboración entre sus miembros y debe paliar cualquier influencia externa que afecte al equipo en el desarrollo. No se debe confundir con el líder del equipo.

Equipo de Trabajo: Son los encargados de llevar a cabo las tareas. Se rigen por una organización horizontal y colaborativa. Tiene como objetivo realizar todas las tareas del Product Backlog con la intención de entregar un producto de calidad en cada Sprint. Estos se autoorganizan y por tanto ningún miembro externo tiene la autoridad de decidir como se deben realizar las tareas. Es recomendable que el número de integrantes sea de 3 a 9 personas. Se ha realizado una modificación en este rol dado que el único componente del grupo es el alumno que está realizando el proyecto.

Roles auxiliares: Son aquellos que no tiene un rol formal y no forman parte del proyecto de manera continua pero deben ser tomados en cuenta según las necesidades del proyecto. Estos pueden ser, por ejemplo, desde usuarios a la hora de realizar algún tipo de pruebas de usabilidad a expertos acerca de un tema sobre el que puede estar tratando el Sprint o proyecto.

En este caso, consideramos los clientes a las personas a las que va dirigido el proyecto. Estos serían el grupo de personas que han desarrollado la aplicación desde la que partimos al comenzar el desarrollo de la herramienta. Por tanto, el Product Owner sería al mismo tiempo el cliente. Se ha decidido que la tarea del Scrum Master se realice entre el programador-analista y los directores de proyecto, ya que ellos son los encargados de superar las dificultades de manera conjunta.

4.3 Reuniones

Trabajar en un proyecto con la metodología Scrum implica realizar tareas de muy corta duración que pueden sufrir varios cambios. A cada una de ellas se le asigna una duración máxima la cual se respeta escrupulosamente, lo que permite una mejor gestión del tiempo y conseguir los objetivos fijados en el Product Backlog.

El primer día de la iteración el grupo realiza la reunión de **Planificación de la Iteración**[22]. Tiene dos partes:

- Selección de requisitos (2 horas): El cliente describe cuales son las funcionalidades prin-

cipales del producto. El equipo pregunta distintas cuestiones que surgen y seleccionan los requisitos prioritarios que consideran que pueden completar en la iteración para poder entregárselos, si este los solicita.

- Planificación de la iteración (2 horas): El equipo elabora todas las tareas del Product Backlog necesarias para los requisitos seleccionados. La estimación del trabajo necesario para cada una de ellas se hace de manera conjunta y los miembros del equipo se auto-asignan las tareas y se auto-organizan en caso de realizar las tareas en grupos con el fin de resolver objetivos especialmente complejos.

El tiempo de cada una de estas partes se ha reducido a 45 minutos ya que el equipo de desarrollo está formado por un único programador. En las reuniones que se realizarán semanalmente se podrán añadir objetivos si el Sprint todavía no ha finalizado y se han cumplido los objetivos marcados.

Cada día el equipo realiza una **Reunión de sincronización** (15 minutos), su objetivo es analizar los avances del día anterior, inspeccionan el trabajo realizado por el resto para comprobar como progresa la iteración hacia el objetivo y planificar las siguientes 24 horas. En este caso, como el alumno es el único que está desarrollando el trabajo no se realizan revisiones diarias con el resto del grupo. Las revisiones se realizan semanalmente y con una duración entre 45 minutos y 1 hora para indicarle al resto del grupo todos los avances realizados, los problemas encontrados, las modificaciones introducidas y los objetivos completados.

El último día de la iteración se realiza la reunión **retrospectiva del sprint**. Esta dividida en dos partes:

- Demostración (1 hora 30 minutos): El equipo presenta al cliente los resultados obtenidos en la iteración de manera incremental, de modo que en cada una de ellas se puede ver los añadidos con respecto a la iteración anterior. En función de los resultados mostrados, el cliente decide adaptaciones necesarias.

- Retrospectiva (1 hora 30 minutos): El equipo analiza su manera de trabajar y cuáles son los problemas que podrían impedirle progresar adecuadamente. El Scrum Master debe encargarse de solucionar o facilitar todos los posibles problemas externos al desarrollo.

El tiempo de duración de estas dos reuniones se ha reducido a 30 minutos cada una ya que en las reuniones semanales se revisa a grandes rasgos el trabajo realizado y se buscan posibles errores en la manera de trabajar.

Capítulo 5

Tecnologías

Con la intención de facilitar el trabajo en cada una de las fases del proyecto se han utilizado distintas herramientas que lo simplifican. Cada una de ellas está especializada en una funcionalidad en concreto y solventará fácilmente la mayoría los problemas que pueden darse a lo largo del proyecto.

5.1 Control de versiones

Como sistema de control de versiones se ha utilizado Git[13], con soporte en GitHub[14] que es una plataforma de desarrollo colaborativo para alojar proyectos, se puede integrar perfectamente en los IDEs que vamos a emplear. A demás hemos utilizado una herramienta gráfica llamada SourceTree[25] para simplificar la forma de interactuar con los repositorios.

5.2 Gestión del proyecto

Realizar la gestión de proyectos de este tamaño no es una tarea trivial, sobre todo teniendo en cuenta que este proyecto requiere que se inviertan grandes cantidades de tiempo en tareas como análisis y periodos de aprendizaje. A estos dos tipos de tarea es complicado asignarles una duración. Para tener un control de todas las tareas que se desean realizar y el tiempo que se debe invertir en cada una de ellas para cumplir la fecha de finalización del proyecto se utiliza en programa Microsoft Project[16]. Una herramienta especializada en gestionar esta clase de proyectos y que nos permite visualizar la estimación del proyecto y en caso de tener algún retraso poder solucionarlo antes de que sea demasiado tarde y tener que posponer la entrega.

5.3 Diseño

Realizar un diseño previo a la implementación facilita el desarrollo de una aplicación ya que evita errores que en caso de producirse durante otra etapa serían muy costosos en tiempo. Funcionan, así mismo, como control de riesgos ya que posibles errores que pueden darse durante el desarrollo ya han podido tenerse en cuenta en la parte de diseño y se han valorado diferentes opciones para solucionarlo. Para realizar un proceso tan importante se ha utilizado MagicDraw[17], una herramienta especializada en realizar diagramas que contiene todas las funcionalidades necesarias para desarrollarlos de forma correcta.

5.4 Análisis

Durante todo este proceso se utilizará un conjunto de datos que simulan distintos tipos de tráfico de red. Este conjunto ha sido creado por *Canadian Institute for Cybersecurity* y han generado más de dos millones de mensajes de red distintos. Para almacenarlos han utilizado ficheros CSV, un tipo de documentos especializados en almacenar y modificar grandes cantidades de información. Para simplificar la obtención y modificación de los datos almacenados es conveniente utilizar librerías especializadas en tratar con estos ficheros.

Las librerías Pandas[18] de Python están especializadas en el tratamiento y análisis de datos, en particular en estructuras de datos y operaciones para manipular tablas numéricas y series temporales.

5.5 Implementación nuevas funcionalidades

La aplicación sobre la que vamos a desarrollar las nuevas herramientas utiliza una interfaz gráfica creada con Vaadin. Esta trabaja sobre un servidor REST[26] implementado en Java. Para el desarrollo de nuevas funcionalidades hemos elegido el IDE Eclipse EE.

Para llevar a cabo la implementación de la visualización de la distribución de los datos se utilizará una librería de JS llamada VisJs[20] que permite visualización dinámica de árboles y grafos. La creación de estadísticas se realizará con la librería D3JS[19] que está especializada en la representación gráfica de información en entornos web. También será necesario insertar código HTML generado mediante JS e importar código CSS proporcionado por la librería FontAwesome [27].

Hemos seleccionado el framework Vaadin que nos facilitará la generación de interfaces web. Este nos proporciona un conjunto de herramientas que facilitan el desarrollo de estas interfaces utilizando Java. En caso de que necesitemos una funcionalidad no implementada por Vaadin, tendremos que realizar un código ad-hoc en JS y CSS para que Vaadin[28] pueda ejecutarlo cuando se inicie la aplicación.

Capítulo 6

Planificación y evaluación de costes

Partiendo de la idea principal de este proyecto, el desarrollo de una herramienta de etiquetado manual, hemos analizado el ámbito en que trabajamos y seleccionado una lista de todas las posibles funcionalidades que podríamos crear, de todas ellas seleccionamos las que consideramos que pueden aportar un mayor valor a nuestro proyecto, al juntarlas hemos conseguido unificar un proyecto.

En este capítulo explicaremos la planificación previa y estimaciones realizadas para el desarrollo del proyecto. La intención es que cuando finalice este capítulo conocer, en la mayor medida de lo posible, cuanto tiempo se invertirá en el proyecto y cuál sería su coste si lo llevara a cabo una empresa. A demás se expondrán los recursos hardware y software empleados junto con los costes derivados de los mismos.

6.1 Planificación previa

Como paso previo al desarrollo del proyecto se separará cada uno de los Sprints en subtareas. Para llevar a cabo los objetivos mencionados anteriormente se dividirá el proyecto en 6 Sprints:

- **Sprint 1: Análisis del dominio:** Como fase inicial procederemos a informarnos acerca del conjunto de datos con el que vamos a trabajar y aplicar distintas técnicas de obtención y análisis de datos, finalmente se generarán un conjunto de ficheros para introducir en la SOM.
- **Sprint 2: Desarrollo de la herramienta de etiquetado:** Este es el punto principal de nuestra aplicación. Se desarrollará, para una aplicación de visualización de mapas autoorganizativos, una funcionalidad que permita etiquetar de manera manual los conjuntos asociados a una neurona. Este apartado requerirá varios periodos de aprendizaje para comprender el funcionamiento interno de la aplicación y la estructura interna de los

datos generados por la SOM.

- **Sprint 3: Desarrollo de una visualización de estadísticas:** Llegados a este punto, desea crear una visualización que permita al usuario analizar los datos almacenados en una neurona en función de sus estadísticas.

- **Sprint 4: Desarrollo de una visualización de dispersión de datos:** Al igual que el punto anterior se desea aportar más información acerca del mapa al usuario, por tanto se creará una visualización que permita al usuario observar como se distribuyen los datos asociados a una neurona.

- **Sprint 5: Desarrollo de un menú contextual:** Para dar más versatilidad a las funcionalidades desarrolladas se ha implementado un menú contextual que permitirá realizar las acciones descritas en los puntos anteriores al pulsar clic derecho sobre una neurona.

- **Sprint 6: Finalización de la memoria:** Se completarán todos los capítulos de la memoria que no se han terminado de redactar durante los Sprints anteriores. Además se realizará una revisión y corrección de posibles errores.

Con este conjunto de tareas a realizar se ha marcado el inicio del proyecto a el día 4 de Febrero de 2019 y con fin 6 de Julio de 2019. Cabe destacar que se realizó una tarea de obtención de requisitos previa que se ha utilizado como base para definir objetivos. Este proyecto que contiene periodos de aprendizaje y de análisis tiene una incertidumbre añadida a su desarrollo, además contiene una componente innovadora y por tanto no podemos prevenir los resultados de cada etapa ni la mayoría de los riesgos derivados. Aun así se ha utilizado como guía de todo el proyecto. En Figure B.1 podemos ver como se han desglosado las tareas anteriormente mencionadas y en Figure B.2 se pueden observar como se han distribuido las reuniones semanales que se mencionan en el apartado Reuniones.

Gracias al modelo de desarrollo que se ha elegido se intenta reducir el riesgo derivado de componentes desconocidas o aleatorias lo máximo posible, tener un umbral aceptable y un plan de control crisis en caso de que un riesgo convierta en un problema. Con estas decisiones se pretende realizar una valoración del progreso y brindar la oportunidad de mejorar en cada nueva tarea, es decir, aprender de errores cometidos y evitarlos en el futuro.

6.1.1 Recursos materiales

Durante el desarrollo de este proyecto serán necesarios los recursos hardware y software que se muestran a continuación:

- Hardware:

  - Ordenador Portátil
    - Modelo: Asus F556UJ
    - CPU: i7-6500U a 2.5GHz
    - Disco duro: SSD 500GB y HDD 1TB
    - Memoria-RAM: 8GB
    - SO: Windows 10 Home 64 bit
    - Tarjeta Gráfica: NVIDIA GeForce 920M de 2 GB
  - Ordenador Sobremesa
    - CPU: AMD FX-8320 8-Core a 3.5GHz
    - Disco duro: SSD 120GB y HDD 3TB
    - Memoria-RAM: 16GB
    - SO: Windows 10 Pro 64 bit
    - Tarjeta Gráfica: NVIDIA GeForce 1060 de 3 GB

- Software:
  - Eclipse EE
  - PyCharm
  - Mircrosft Project
  - MagicDraw
  - GitHub

Podemos Encontrar una descripción del software utilizado en el apartado Tecnologías

6.2 Estimación de costes

- Costes estimados del Hardware: Estos costes se consideran nulos ya que disponíamos de estos dispositivos antes del comienzo del proyecto. En cualquier caso el coste de estos dos dispositivos siempre sería nulo ya que a cualquier entidad que se encargue un proyecto software debe contar con estas herramientas.
- Costes estimados Software, al igual que en el caso anterior, estos gastos también se consideran nulos
  - Eclipse EE es una herramienta gratuita, PyCharm permite utilizar una versión de la comunidad que es gratuita y la versión gratuita de GitHub nos proporciona todas las funcionalidades necesarias.
  - En el caso de la herramienta MagicDraw la universidad nos proporciona una licencia. Si fuera necesario adquirir una licencia, la substituiríamos por alguna herramienta de software libre que permita las mismas funcionalidades.

– Al igual que en caso de MagicDraw, la universidad nos proporciona una licencia de Microsft Project. pero en este caso la herramienta no la podríamos substituir debido a que las aplicaciones de gestión de proyectos requieren un periodo de aprendizaje muy amplio, así que seria necesario obtener una licencia que tendría un coste de 25.30€/mes para cada uno de los participantes del proyecto(75.90 €/mes), ya que esta sería la licencia mínima para poder gestionarlo de la misma manera que la actual.

En los costes humanos se asumen dos categorías de trabajo: Analista programador (Mario Iglesias) y director de proyecto (José Carlos Dafonte y Manuel Fernández).

Las jornadas de trabajo pueden superar en algunas ocasiones especiales las 8h, sabiendo que la mayoría de los días se trabajará aproximadamente 7h. Además cada semana se realizarán reuniones de cerca de 1h y al finalizar cada Sprint se han programado un conjunto de reuniones que durarán 2h y 30 minutos. En total se han estimado 670 horas*hombre para el programador-analista y 34 horas*hombre para cada uno de los directores de proyecto. Teniendo en cuenta que el sueldo de un analista programador es de $15€/h$ y la de un director de $30€/h$ se estiman los costes presentes en la tabla Table 6.1.

	Sueldo	**Horas * Hombre**	**Inversión**
Director de proyecto (x2)	$30€/h$	34	2040€
Analista - programador	$15€/h$	670	10.050€
		Total:	12.090

Cuadro 6.1: Estimación de costes de los recursos humanos.

Debemos tener en cuenta que este coste solo sería real si el proyecto se llevara a cabo a través de una entidad externa, teniendo en cuenta que el coste del analista programador y los directores es 0€ ya que todo el grupo ya es parte de la universidad.

Capítulo 7

Conceptos previos

Para poder comprender las decisiones tomadas y la finalidad de este proyecto, es necesario familiarizarse con una serie de conceptos que describiremos a lo largo de este apartado, separándolos en secciones para facilitar la comprensión.

7.1 Análisis

El conjunto de datos con el que vamos a trabajar esta formado por ficheros CSV. Los CSV son un tipo de documento sencillo para representar datos en forma de tabla en los que las columnas se separan por comas o punto y coma y las filas por saltos de linea[29].

Debido a la gran cantidad de información que hay agrupada en los ficheros y con la que vamos a trabajar consideramos necesario aplicar técnicas de *Big Data*.

En estos ficheros estarán contenidos distintos tipos de tráfico de red. Cuando hablamos de este termino nos referimos al conjunto de información que utilizan varios sistemas informáticos para poder comunicarse entre ellos. Si nos referimos al análisis, estamos hablando del proceso de inferir esas información a partir de las características de los datos que circulan entre los sistemas que se están comunicando [30].

Consideramos que uno o varios sistemas informáticos están realizando un ataque de red en caso de que este realizando una maniobra ofensiva deliberada que tiene como objetivo tomar el control, desestabilizar o dañar un sistema informático [31].

Se considera que se está capturando tráfico maligno en caso de que los datos que se estén capturando en la comunicación sean de un ataque de red y benigno en caso contrario.

En cada uno de los ficheros se encuentra una parte de los datos. La información que contiene cada fichero está descrita en el nombre, en primer lugar indica el día que se obtuvieron los datos, en segundo lugar durante que parte del día se obtuvieron y por último el ataque que se ha capturado.

Cada fila del CSV representa la comunicación realizada entre dos interlocutores y cada

columna, representa para cada fila, un dato de la comunicación.

El conjunto del que vamos a obtener todos estos datos es *ISCX 2017*[5], generado a través de simulaciones que realizo *Canadian Institute for Cybersecurity*[32].

Los datos se comenzaron a generar el Lunes 3 de Julio de 2017 y finalizo el Viernes 7 de Julio de 2017, a continuación podremos encontrar un resumen del tráfico generado, para mas información ver el articulo [5].

Monday, July 3, 2017

- Benign (Normal human activities)

Tuesday, July 4, 2017

- Brute Force[33]: Trata de recuperar una clave probando todas las combinaciones posibles hasta encontrar aquella que se busca y permita el acceso al sistema.
  - FTP-Patator (9:20 – 10:20 a.m.): Utiliza la herramienta Patator[34] para realizar un ataque a un servicio que utiliza el protocolo de transferencia de datos FTP [35].
  - SSH-Patator (14:00 – 15:00 p.m.): Utiliza la herramienta Patator[34] para realizar un ataque a un servicio que utiliza el protocolo de transferencia de datos SSH [36].

Wednesday, July 5, 2017

- DoS / DDoS[36]: Ataque a una red para intentar dejarla inaccesible a sus usuarios
  - DoS slowloris[37] (9:47 – 10:10 a.m.): Con una sola máquina realizar peticiones web a un servidor intentando que sea inaccesible utilizando un mínimo de ancho de banda.
  - DoS Slowhttptest[38] (10:14 – 10:35 a.m.): Simulador de ataques lentos altamente configurable.
  - DoS Hulk[39] (10:43 – 11 a.m.): herramienta DOS cuyo principal objetivo es generar peticiones únicas para evitar motores de caché e incidir directamente en la carga del servidor.
  - DoS GoldenEye[40][41] (11:10 – 11:23 a.m.): Herramienta para realizar test de denegación de servicio.
  - Heartbleed Port 444 (15:12 - 15:32): Heartbleed[42] es un agujero de seguridad que hay en OpenSSL[43] que permite al atacante acceder a la memoria del atacado. Este ataque se realizará al puerto 444.

Thursday, July 6, 2017

- Morning
  - Web Attack – Brute Force (9:20 – 10 a.m.): Realizar un ataque web utilizando técnicas de Fuerza Bruta.
  - Web Attack – XSS (10:15 – 10:35 a.m.): Atacar webs aprovechando la vulnerabilidad XSS[44]
  - Web Attack – Sql Injection[45] (10:40 – 10:42 a.m.): Se realizan consultas SQL[46] complicadas o que carecen de sentido para una base de datos para que funcione de manera incorrecta.
- Afternoon
  - Infiltration – Dropbox download: Utilizar la sincronización de ficheros de Dropbox para descargar un fichero mediante el cual el atacante realiza un escaneo de la red.
  - Meta exploit Win Vista (14:19 and 14:20-14:21 p.m.) and (14:33 -14:35): Metasploit [47] es una plataforma de pruebas que permite encontrar, explotar y validar vulnerabilidades. En concreto se utilizará para encontrar vulnerabilidades en Windows vista y explotarlas.
  - Infiltration – Cool disk – MAC (14:53 p.m. – 15:00 p.m.): Ataque te infiltración en un sistema MAC que se realizar al insertar una memoria USB infectada.

Friday, July 7, 2017

- Morning
  - Botnet ARES (10:02 a.m. – 11:02 a.m.): Se utiliza la herramienta Ares[48] para realizar un ataque Botnet[49]
- Afternoon
  - Port Scan[50]: Permite saber que puertos están abiertos para la comunicación.
  - DDoS LOIT (15:56 – 16:16): En concreto se buscarán vulnerabilidades de DDOS utilizando Metasploit[47].

Cada día de la semana se generaron distintos tipos de tráfico, cada fila del fichero CSV contiene una etiqueta que indica si es de algún tipo de ataque de red o si se trata trafico benigno. El lunes fue el único que día en el que se puede encontrar únicamente tráfico benigno, y el resto de días se pueden encontrar diferentes ataques, junto con tráfico benigno.

Las etiquetas que se utilizan para diferenciar los ataques son las indicadas en Table 7.1.

Cuadro 7.1: Etiquetado del conjunto de datos por días

Días	**Etiquetas**
Lunes	Benign
Martes	BForce, SFTP and SSH
Miercoles.	DoS and Hearbleed Attacks slowloris, Slowhttptest, Hulk and GoldenEye
Jueves	Web and Infiltration Attacks Web BForce, XSS and Sql Inject. Infiltration Dropbox Download and Cool disk
Viernes	DDoS LOIT, Botnet ARES, PortScans (sS, sT, sF, sX, sN, sP, sV, sU, sO, sA, sW, sR, sL and B)

7.2 Implantación de la herramienta

Vamos a desarrollar un conjunto de funcionalidades para una aplicación de visualizado de datos, esta trabaja con los resultados obtenidos de una SOM(Self-Organizing Maps)[51] que los agrupa en función de los patrones que encuentra. Las redes de neuronas de este tipo agrupan información en función de rasgos similares. No existe ningún supervisor que indique si la red neuronal esta operando de forma correcta o incorrecta porque no se dispone de ninguna salida como objetivo hacia la cual la red neuronal deba tender. La red debe descubrir cuales son esos rasgos comunes, regularidades, correlaciones o categorías en los datos de entrada e incorporarlos a su estructura interna de conexiones. Por esto mismo se dice que las neuronas deben auto-organizarse en función de los estímulos(datos) que proceden del exterior.

Este tipo de redes tienen aprendizaje competitivo, esto quiere decir que varias neuronas compiten entre ellas con el fin de que se les asigne a cada una el nuevo dato introducido, pero finalmente solo puede haber una vencedora. El objetivo de este aprendizaje es agrupar datos en categorías, cada una de estas categorías se denomina neurona, cada una de ellas tiene un representante de todo el conjunto denominado prototipo.

Los resultados del agrupamiento de los datos se guardarán en un objeto serializado. La serialización[52] es la transformación de un objeto en una secuencia de bytes que pueden ser posteriormente leídos para reconstruir el objeto original.

Se utiliza una arquitectura cliente-servidor, concretamente esta se lleva a cabo mediante un servidor REST[26] que conforma un grupo de restricciones para describir cualquier interfaz entre sistemas que utilice directamente HTTP[53] para obtener datos o indicar la ejecución de operaciones sobre los mismos.

Para separar las distintas partes de la aplicación se utiliza el patrón de diseño Modelo-Vista-Controlador(MVC)[54]. Este se caracteriza por separar la interfaz de la lógica de la apli-

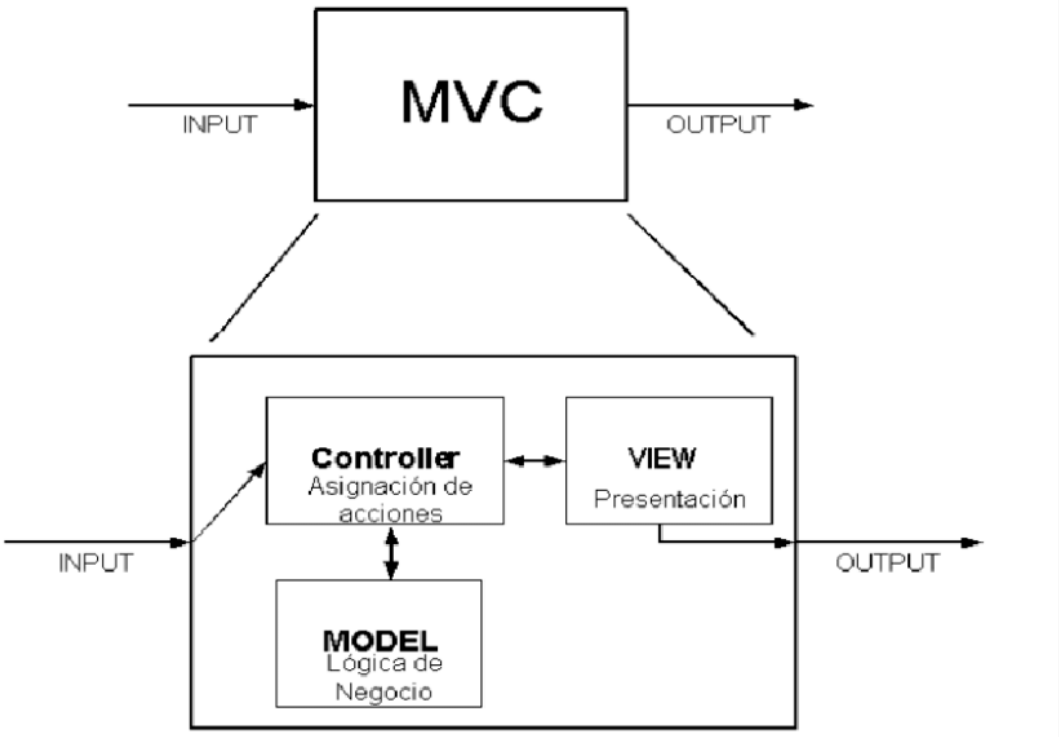


Figura 7.1: Diagrama MVC

cación. Esto nos permite realizar un cambio en las distintas partes de nuestro código sin que resulten afectadas las demás partes. Podemos observar el comportamiento de este en Figure 7.1.

Capítulo 8

Análisis de datos

Teniendo seleccionado el conjunto con el que se va a trabajar se procede a realizar un análisis de los datos que consiste en: seleccionar la información válida, un podado de datos válidos pero no necesarios, una normalización del conjunto restantes, analizar los datos normalizados y categorizados y por último la creación de un fichero con los datos de interés que contenga.

8.1 Selección de datos válidos

Los ficheros que se están utilizando tienen una gran cantidad de datos, pero no todos son correctos o contienen información. Para intentar reducirlos los máximo posible y minimizar el cómputo se realizará una búsqueda de los que no son válidos y se eliminarán.

- Al final del fichero *Thursday-WorkingHours-Morning-WebAttacks.pcap_ISCX.csv* se han encontrado 288601 líneas que no contienen ningún tipo de información.
- Todos los ficheros contienen las siguientes columnas sin ninguna información:
  - Bwd PSH Flags
  - Bwd URG Flags
  - Fwd Avg Bytes/Bulk
  - Fwd Avg Packets/Bulk
  - Fwd Avg Bulk Rate
  - Bwd Avg Bytes/Bulk
  - Bwd Avg Packets/Bulk
  - Bwd Avg Bulk Rate
- El fichero *Friday-WorkingHours-Afternoon-DDos.pcap_ISCX* es el único que contiene la columna *External IP* y no tiene contenido en ningún caso.

8.2 Tipos de datos

Los datos almacenados actualmente son demasiado dispares lo que hace que sea muy complicado operar con ellos. Por ello, es más sencillo tenerlos separados en dos tipos, categóricos y numéricos. Los valores categóricos son aquellos a los que se le puede asignar una categoría que se decide con anterioridad; los numéricos serán todos aquellos que no sean categóricos, sus valores estarán dentro de los números reales, NaN o infinito.

8.2.1 Datos categóricos

El interés principal que tenemos en los datos categóricos es su significado y no su valor, por tanto lo que se hace es cambiar su valor por su significado de manera que dos datos distintos pueden significar lo mismo. Esto permite agrupar distintos valores y simplificarlos.

Los datos seleccionados como categóricos son los siguientes:

- Flow ID: Agrupación de: Source IP, Destination IP, Source Port, Destination Port y Protocol.
- Source IP: Dirección IP que actúa como emisor en la comunicación. La información que nos proporciona este campo es conocer quien esta realizando la comunicación y si se realiza desde una dirección conocida y si esta dirección tiene asignado un PC o un Servidor. Cualquier otro valor que no sea una dirección IP conocida carece de importancia y se asignara como «Otros».
- Source Port: Puerto de origen desde el que se realiza la comunicación. De estos datos nos interesa el rango en el que se encuentran. Ya que este es el que les aporta un significado. Los rangos cubren todos los posibles puertos existentes, a mayores de esto se crea una última categoría llamada «Otros» a la que se le asignarán todos los valores erróneos.
- Destination IP: Dirección IP que actúa como receptor en la comunicación. Al igual que el Source IP, nos interesa saber si se realiza hacia una dirección IP conocida o una desconocida. Se le asignarán los valores «PC», «Servidor» u «Otros» según sea más conveniente.
- Destination Port: Puerto de destino en la comunicación. Del mismo modo que Source Port, en este caso nos interesan únicamente los rangos. La categoría «Otros» indicará si ha habido algún error.
- Protocol: Describe las normas y los criterios que indican cómo deben comunicarse dos dispositivos. Tras realizar un análisis se han encontrado únicamente 3 protocolos distintos. Pero para evitar posibles errores en caso de encontrar protocolos que no estén

categorizados se ha añadido la categoría «Otros» a la que se le asignarán los protocolos desconocidos.

- Timestamp: Indica en que momento se ha realizado la comunicación. Podría ser interesante almacenar el dato original para su posterior procesado.
- Label: Indica si durante la comunicación se ha realizado algún tipo de ataque.

8.2.2 Datos numéricos

Para analizar los datos numéricos, a diferencia de los categóricos, no nos interesa conocer el numero de repeticiones de cada valor si no como se distribuyen en un histograma. Se separan por columnas y para cada una de ellas se obtienen los siguientes valores: Máximo, Mínimo, Media Aritmética, Desviación típica, Cantidad de *NaN* encontrada y Cantidad de Infinitos encontrada.

Consideraremos numéricos todos aquellos que no se han eliminado y que no forman parte del conjunto de los categóricos.

Dado que las columnas tienen valores muy dispares, es necesario realizar una normalización a los datos para facilitar su análisis y permitir su agrupación mediante la técnica SOM.Se han realizar dos tipos de normalizaciones, MinMax y ZScore.

- MinMax: Acota todos los valores entre [0,1]. Es necesaria para generar los datos que se van a introducir en la SOM.
- ZScore: Permite conoce como de lejos se sitúa un valor de la media, siendo positivo si esta por encima y negativo si esta por debajo.

8.3 Creación ficheros de análisis

Para que los datos modificados sean más accesibles, se genera un conjunto de ficheros que facilitan su visualización. Por cada uno de los documentos analizados se generan dos nuevos, uno con los datos después de haberlos categorizado y normalizado y otro con todos los originales.

Para cada dato categórico, se guarda el número de veces que aparece repetido. En caso de conservar los valores originales no se almacenan en este fichero debido a que ya se encuentran en el fichero que contiene información acerca de los datos originales.

Los valores numéricos que se guardan tanto para los originales como para los normalizados son los indicados en el apartado Datos numéricos. Después de haberles aplicado la normalización ZScore se genera un histograma por cada columna de cada fichero analizado, ignorando todos los valores que son *NaN* e infinito ya que resulta imposible visualizarlos en

un histograma. Esto permite un filtrado de los datos más intuitivo, ya que en cada imagen se observan tres histogramas, todos con las mismas escalas en sus ejes. El más grande contiene todo los datos de la columna y a su lado dos histogramas más pequeños, uno que muestra únicamente los valores pertenecientes al tráfico maligno y otro que muestra los del tráfico benigno.

Estos ficheros que se generan no permiten observar en profundidad ni la normalización ni la categorización, por tanto también se almacenan los documentos CSV de los datos modificados. Estos no contienen el mismo número de filas y columnas que los originales debido a la selección de características realizadaEn el apartado Selección de datos válidos se indican los datos que eliminamos .Los valores numéricos que contienen *NaN* o *infinito* conservan su valor original.

De los datos que se han obtenido como resultado del análisis es necesario generar un conjunto de ficheros para poder introducirlos en la SOM. Estos deben tener el formato que se indica a continuación:

- Partimos de múltiples ficheros CSV originales y decidimos cuales son las columnas de interés. Todas las columnas que no resulten de interés son eliminadas.
- Se agrupan todos los ficheros en un único CSV. Esto se hace porque es necesario tener identificadores únicos, simplificando así la gestión de los mismos.
- Procesado de datos categóricos:
  - Los datos categóricos no pueden ser introducidos en la SOM directamente, por lo que crearemos un nuevo fichero JSON con todas las categorías y dentro de cada una de ellas asignaremos un valor numérico entre $[0, n-1]$ a cada valor dentro de una categoría(Donde n es el número de categorías en hay en cada variable categórica). Es importante mantener un orden concreto para poder identificarlas unívocamente.
  - A partir del JSON traducimos los datos a su codificación numérica y se ordenan las columnas categóricas por orden alfabético
  - Existen datos que pueden pertenecer a un rango. A cada rango se le asigna un valor y a cada dato se le asigna el valor del rango al que pertenecen.
- Los datos numéricos también es necesario modificarlos ya que debemos asegurarnos de que están en un rango entre $[0, 1]$. Se realiza una normalización utilizando MinMax, que escala los datos en este rango.
- Se modifica el orden de las columnas situando en primer lugar las columnas numéricas y en segundo las categóricas.

- Se genera un fichero CSV que contiene todas las modificaciones indicadas.
- A partir del CSV generado crearemos un último documento que tendrá una línea por cada fila del fichero. Cada línea estará compuesta por:
  - ID separado por punto y coma de los datos categóricos
  - Los datos categóricos separados por espacios entre sí y separados por punto y coma de los datos numéricos.
  - Los datos numéricos, igual que los categóricos, estarán separados por espacios.
- Este último fichero generado, junto con el JSON serán los que se utilicen en el entrenamiento y posterior visualización de la SOM.

8.4 Resultados del análisis

Actualmente existe un estudio paralelo en el grupo de investigación, en él están utilizando un dataset generado anteriormente a el utilizado en este proyecto (*ISCX 2012*). En este estudio se están obteniendo resultados prometedores. Al cambiar el dataset a *ISCX 2017* es necesario realizar un análisis que asegure que datos agrupados bajo el mismo nombre nos aportaran resultados similares o mejores.

Conjuntos de datos utilizados en el *ISCX 2012*: Source IP, Source Port, Destination IP, Destination Port, Protocol, Flow Duration, Fwd Avg Bytes/Bulk, Bwd Avg Bytes/Bulk

En el articulo[5] se realiza un análisis que demuestra que no hay diferencia entre utilizar los mismos conjuntos que en el *ISCX 2012* utilizando los datos del *ISCX 2017*. Para corroborar esta información se ha realizado un estudio a grandes rasgos de los mismos y se ha llegado a la misma conclusión.

La imagen Figure 8.1 representa como se distribuyen los datos de «Flow Duration» en un histograma. Se puede observar la distribución dispar entre el tráfico benigno y el maligno. La misma situación ocurre con el resto de características utilizadas.

Dado que el objetivo principal del proyecto es desarrollar una herramienta de etiquetado y visualización, no se ha llegado a analizar en profundidad que conjuntos de características podrían utilizarse en el ISCX2017 para mejorar los resultados del agrupamiento. A pesar de esto, se ha realizado un cómputo que permitiría en un futuro poder realizarlo de manera sencilla.

8.5 Implementación

Para obtener los resultados mencionados anteriormente, ha sido necesario dividir el trabajo en 3 fases. Cada fase consta únicamente de un módulo y el desarrollo de cada uno estos

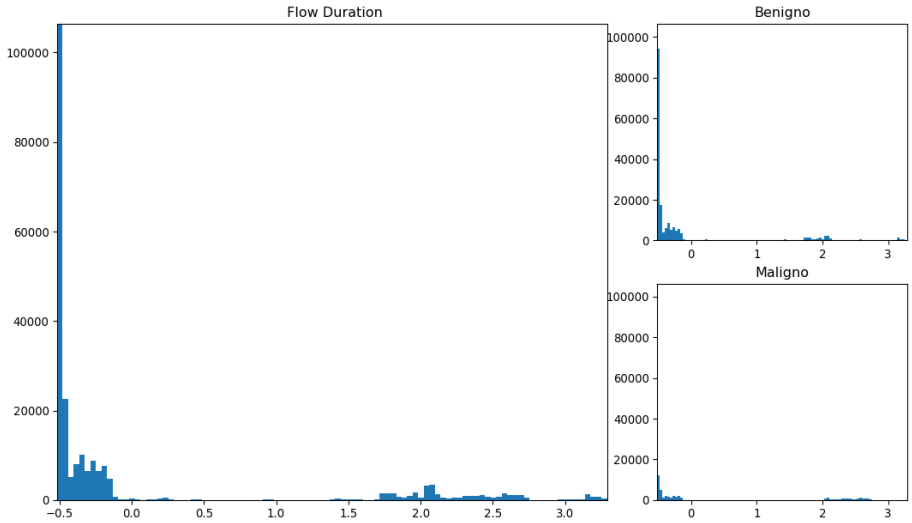


Figura 8.1: Distribución de «Flow Duration».

módulos es independiente del resto. El primer módulo corresponde a la fase preprocesado, en la que se eliminarán, categorizarán y normalizarán los datos originales. En el segundo módulo se lleva a cabo la fase de creación de ficheros de análisis, donde se exportarán los histogramas y los ficheros de texto que contienen información a analizar. El tercer y último módulo utiliza los documentos originales para crear los ficheros que se utilizarán como entrada en la SOM. La generación de todos los documentos del análisis así como los de entrada de la SOM consiste dependen de la ejecución de estos tres módulos de forma secuencial.

Todo el desarrollo, los datos generados y la información recopilada durante este proceso se almacenará en el directorio «Análisis», por tanto cada vez que nos refiramos a la ubicación de algún fichero o al directorio principal, nos estaremos refiriendo siempre a esta carpeta.

En el diagrama UML Figure 8.2 está contenida toda la implementación realizada, se ha utilizado de guía para facilitar todo este proceso. En él se observa que existen 3 documentos principales, cada uno de ellos tiene una ejecución individual ya que se ha realizado un diseño modular. Todos se sirven del fichero «utilities.py» que se utiliza para mantener la lógica en los ficheros principales y evitar código duplicado.

8.5.1 Ficheros de configuración

Antes de comenzar el desarrollo se ha tomado la decisión de diseño de crear dos documentos que simplificarán la obtención de los datos durante toda la implementación.

Estos documentos simplifican la forma de tratar los datos sin necesidad modificar el có-

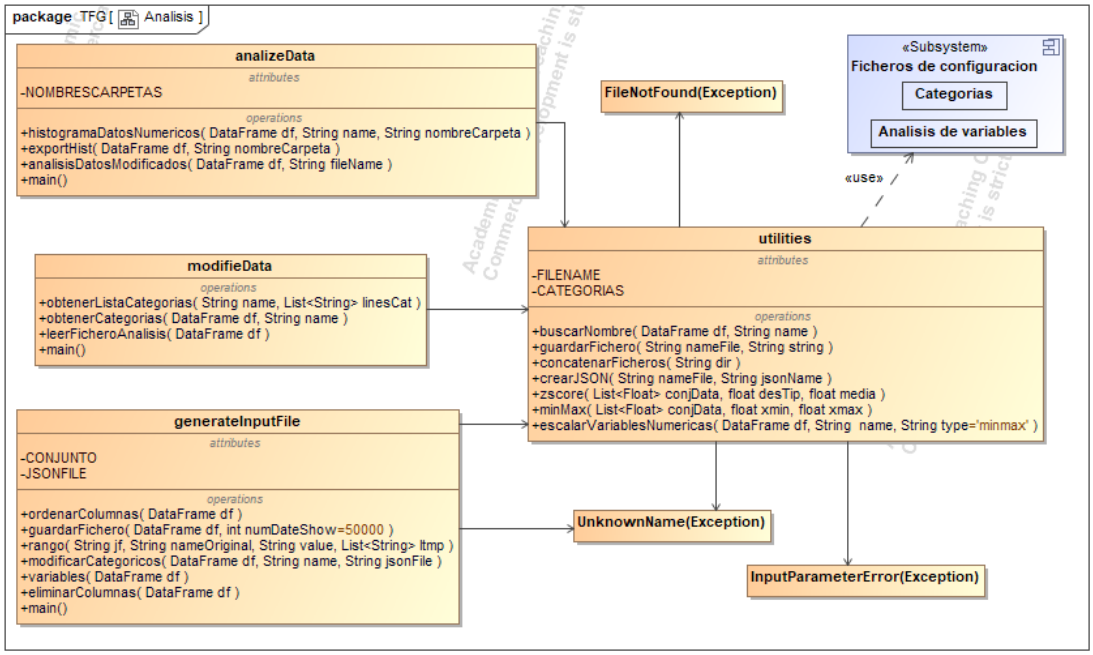


Figura 8.2: UML Análisis

digo: «Analisis de variables.txt» y «Categorias.txt» se ubican en el directorio principal. Los nombres asignados a estos ficheros son constantes creadas en el fichero «utilities.py», si se desea utilizar otro documento de configuración bastaría con cambiar los valores de esas dos constantes.

«Analisis de variables.txt»

Fichero de texto que contiene valores conocidos, que son el nombre de las columnas del CSV, y un valor asignado a cada una de esas columnas. Hay 4 posibles opciones de valores asignados.

- ”C” indica que esa columna contiene datos categóricos.
- ”N” indica que esa columna contiene datos numéricos.
- ”I” indica que los datos de esa columna pueden ser eliminados.
- ”?” indica que los datos de esa columna pueden ser ignorados.

«Categorias.txt»

Fichero de texto que proporciona información sobre qué categorías existen y cuales son las correspondencias entre los valores conocidos y los asignados dentro de cada categoría. El

formato que se sigue para cada categoría es el siguiente:

1. El nombre de la categoría, que se corresponde con el nombre de la columna
2. Opcionalmente, dentro de cada categoría, un conjunto de pares de la forma: valor conocido → valor asignado (este valor asignado podría ser un espacio en blanco o cadena vacía). Cada una de estas correspondencias estará en una nueva línea que comienza por espacio o tabulación.

Todas las categorías con alguna correspondencia deberán incluir como última entrada de la lista la correspondencia especial: Otros →Valor asignado, con el fin de asignar un significado a todos los valores desconocidos o erróneos.

Los valores conocidos pueden ser un único valor o un rango (dos valores separados por un espacio en blanco)

Finalmente la separación entre la lista de asignaciones de una categoría y el nombre de la columna de la siguiente categoría será una línea en blanco.

8.5.2 Módulo 1: Eliminación, normalización y categorización

En este apartado se explicará como se han desarrollado las secciones Selección de datos válidos y Tipos de datos. Se han utilizado los ficheros «Analisis de variables.txt» y «Categorias.txt» para simplificar el código utilizándolos para tratar los distintos tipos de datos. La implementación de este módulo se encuentra en el fichero «modifieData.py» que en algunas ocasiones utilizará funciones del fichero «utilities.py».

El directorio «CSVs» contiene los documentos CSV originales. La única modificación que se realiza sobre estos es eliminar las 288601 líneas vacias del fichero «Thursday-WorkingHours-Morning-WebAttacks.pcap_ISCX.csv» ya que carecían de información.

Para cada documento CSV original se lee «Analisis de variables.txt» línea por línea, seleccionando el nombre de la columna y comprobando el valor que se le ha sido asignado. Estas columnas se tratan en en función de su valor asignado de la siguiente manera:

- Si el valor es «C» se busca el nombre de esa columna en el fichero «Categorias.txt» y se comprueba si cada uno de los datos del CSV de esa columna se encuentran entre valores conocidos. Se le asigna el valor asociado correspondiente, en caso de no cumplir ninguna de las reglas, se le asigna el valor «Otros». Si el valor asignado es un espacio en blanco, significa que lo que se quiere guardar es el valor conocido. En el caso de que el valor conocido sea un rango, se buscará a qué rango pertenece cada uno de los datos y se almacenará el valor asignado a ese rango. Si en el fichero «Categorias.txt» el nombre de la columna no tiene ningún valor conocido, significa que se quieren guardar todos los datos originales.

- Si el valor es «N» los datos de esa columna se normalizan con la función ZScore. En caso de que se encuentren NaN o Infinitos se guardan tal y como estaban.
- Si el valor es «I» los datos se eliminarán.
- Si el valor es «?» ese campo no es de ningún tipo y se guardan los valores originales.

Al finalizar el proceso de modificación de los datos se genera, por cada fichero CSV original, uno nuevo que se almacenará en el directorio «CSVsGenerados» que contendrá todos los datos modificados y su nombre será el mismo el del fichero original.

8.5.3 Módulo 2: Creación de ficheros de análisis

En este módulo se implementará la generación de gran parte de los ficheros que se indican en el apartado Creación ficheros de análisis. En concreto, se generarán todos los histogramas y dos documentos de texto que por cada fichero CSV modificado: uno con los datos antes de modificarlos y otro con los datos modificados. Este conjunto de ficheros que se ha creado es el que se utilizará para realizar el análisis. Este módulo está implementando en el fichero «modifieData.py».

Los documentos de texto que contienen información de los CSV estarán ubicados en el directorio «Datos», la información de los datos originales se guardará con el nombre «(Datos Or)» seguido por el nombre del fichero original, mientras que el fichero de datos modificados comienza por «(Datos Mod)» seguido por el nombre del fichero original. Estos dos documentos contienen la información indicada en el apartado Tipos de datos.

Con los datos modificados se creará un fichero de texto y para los valores numéricos de cada documento se genera un conjunto de imágenes, cada una de ellas contiene histogramas de una característica. Estos documentos se crean en el directorio «Histogramas». Para las imágenes de cada fichero se crea un directorio nuevo, asociado con el día que se realiza el ataque que está siendo capturado y si hay varios ficheros para el mismo día se añade el nombre del ataque que se captura.

8.5.4 Módulo 3: Exportar ficheros SOM

En el tercer y último modulo se exportarán los datos de interés en un formato concreto para que se puedan generar los documentos de entrada de la SOM: un fichero JSON que traduce los valores categóricos a valores numéricos y un documento de texto con un formato concreto que contiene los datos numéricos y categóricos que queremos introducir en la SOM. Los documentos que se van a generar se crean a partir de los CSVs originales.

Esta etapa se desarrollará por completo en el documento «generateInputFile.py» que se encuentra en el directorio principal. El primer paso será generar un fichero JSON que contenga

la traducción de cada uno de los elementos conocidos para cada variable categórica a un valor entre $[0, n-1]$ a partir de los datos del documento «Categorias.txt». El nuevo fichero se crea en la ruta indicada por la constante «JSONFILE», por defecto: directorio «Ficheros SOM» y nombre «Categorias.json».

A continuación se abren uno por uno los ficheros CSV originales y se eliminan todas las columnas que no se encuentran en la constante «CONJUNTO» que indica cuales son las columnas de interés. Estos documentos se almacenan en el directorio «CSVsEntrada». Se obtienen todos los ficheros de esa carpeta, se concatenan para tener un único fichero con IDs únicas para cada fila. A este documento que contiene todos los datos se le realizan dos modificaciones: una normalización de los valores numéricos para acotarlos entre $[0, 1]$ y a cada dato categórico se le asigna el valor numérico que tiene asignado en el JSON. Con estas modificaciones realizadas se genera un documento CSV dentro del directorio «Ficheros SOM» con el nombre «AllCSVs.csv». Por último se generara el fichero de texto necesario para la SOM en el mismo directorio, con el nombre «inputFile.txt» (ver Creación ficheros de análisis).

8.5.5 Otros ficheros

A mayores de los ficheros que se han indicado anteriormente existen más que facilitan la reutilización del código, la comprobación de errores y ayudan a mantener la coherencia del resto de clases.

- «exceptions.py» contiene excepciones personalizadas para los posibles errores.
- «utilities.py» contienen constantes y funciones que se utilizan en diferentes clases y funciones que, aunque se utilicen solamente en un apartado podrían reutilizarse para más funcionalidades.

Capítulo 9

Desarrollo

El objetivo inicial de este trabajo es ampliar la aplicación de visualización de SOMs desarrollando un conjunto de herramientas para intentar mejorar su rendimiento y su utilidad pero sin reducir su eficiencia y eficacia. Resulta conveniente hacer una breve descripción de cuales son las principales funcionalidades de esta aplicación en el momento que llega a nuestras mas. Teniendo en cuenta que ya contiene un amplio conjunto de herramientas nos centraremos en explicar las funcionalidades de interés para el desarrollo de las que se van a desarrollar.

Tras conocer el estado del sistema a nivel de diseño y desarrollo describiremos todas las modificaciones que llevaremos a cabo. Con la intención de mantener la legibilidad del código hemos concluido respetaremos lo máximo posible todas las decisiones de diseño que se han tomado anteriormente sobre la aplicación, trabajaremos sobre ellas y añadiremos nuevas solo en caso de ser necesario.

9.1 Sistema actual

Este sistema nos permite visualizar los resultados obtenidos de una SOM y observar redes de 3 tamaños distintos: 10x10, 20x20 y 30x30. Además, nos permite realizar varias funcionalidades sobre la SOM para aportar al usuario la mejor experiencia posible en función de sus necesidades: realizar distintos filtrados y ajustar la visualización de la mejor manera posible en cada momento; descargar los datos almacenados en una neurona; almacenar varias SOM y seleccionar la que desea visualizar en cada momento y a para cada una de ellas guardar varios mapas de etiquetado externo, entre otras funcionalidades.

Los ficheros que se han generado en el capítulo Análisis de datos no se insertan directamente en la aplicación, primero se introducen en una SOM y esta los agrupa en distintas neuronas. Posteriormente se realiza un conjunto de operaciones que generan ficheros auxiliares para agilizar la visualización. Todas estas tareas son críticas ya que sin ellas una visualización en tiempo real de esta información no sería viable ya que estamos trabajando con *Big*

Data e intentar visualizar directamente su contenido implicaría tener unos tiempos de carga demasiado altos.

La base principal de la aplicación parte de una arquitectura cliente-servidor, en la que utilizamos un servidor REST[26] como base y un servidor web como cliente, que estará provisto de una interfaz de usuario. Sobre esta estructura se está utilizando el patrón MVC[54]. La Vista esta representada por el servidor web, el Controlador por el servidor REST y el Modelo, que almacena toda la información y es a este a quien se le soliciten los datos necesarios tanto desde la Vista como desde el Controlador, se situaría entre ambos.

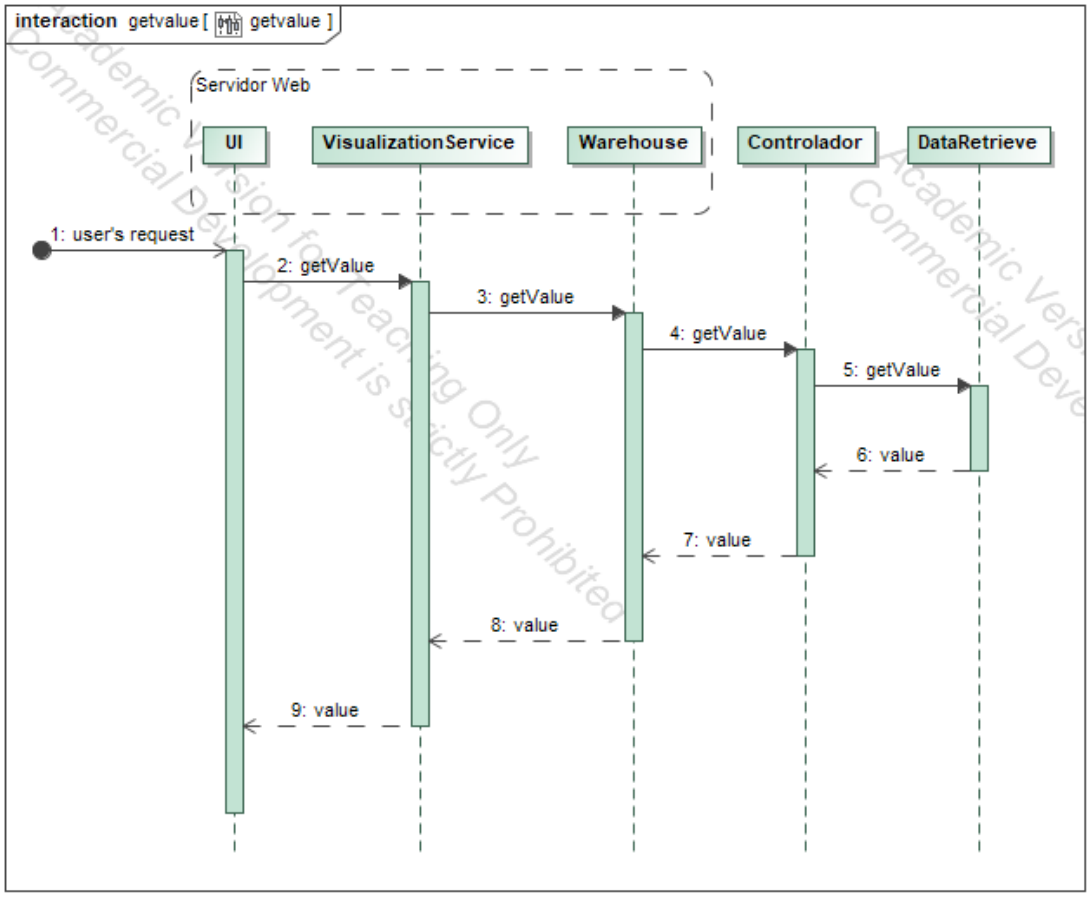


Figura 9.1: Obtener un valor del modelo

Todas las herramientas que utiliza la aplicación tienen la funcionalidad de obtener información del servidor, pero no de modificarla. En el diagrama que encontramos en Figure 9.1 se puede observar el funcionamiento cada vez que se realiza una petición desde el servidor web, esta está formada por 3 capas distintas que se utilizan para abstraer la conexión con el Controlador de la interfaz de usuario(IU), hasta que se envían los datos al Controlador, que le realiza una petición al Modelo, representado por «DateRetrieve» y como se realiza el mismo

trayecto hasta la IU donde se muestran los datos obtenidos durante esa petición.

9.1.1 Modelo

Es la representación de toda la información con la que el sistema trabaja y gestiona todos los accesos que hay a ella. Los datos almacenados de la SOM se encuentran en objetos serializados mientras que los de información para la visualización se encuentran en ficheros properties. Cada uno de ellos contiene distinta información. No todos los documentos serializados se utilizan en la visualización, algunos se utilizan para realizar un cálculo previo a la ejecución del programa principal.

Durante la ejecución del programa principal cada vez que se obtiene un dato es necesario deserializar un objeto y obtener la información necesaria. En cambio si se quiere modificar un dato es necesario modificar el valor y volver a serializarlo, para que se conserven los cambios.

Los ficheros de configuración se utilizan para mantener la estructura de ficheros correcta, almacenar documentos en su ubicación indicada o el color que deben tener las neuronas de la SOM. Estos archivos tienen la extensión «.properties», son ficheros de configuración de java. Concretamente nos interesa conocer «colors.properties» que contiene una lista clave-valor donde cada nombre es la clave y el color el valor asignado.

El fichero «colors.properties»es necesario para poder visualizar el etiquetado externo(EE). Cada etiqueta externa debe tener un par clave-valor, donde la clave será la etiqueta y el valor el color, por tanto no pueden existir etiquetas repetidas. Todas las que estén asignadas a una neurona deben estar en este fichero con su valor asignado. Cada SOM debe tener al menos un EE en el que cada neurona tenga asignada una etiqueta.

9.1.2 Vista

La Vista del sistema actual es un servidor web desde el que podremos acceder a los datos que hay almacenados en el REST. Esta Vista está dividida en dos regiones, una que contiene la SOM seleccionada y su configuración y otra que contienen la visualización de la red de neuras y una leyenda que indica que significa el color asociado a cada neurona.

En Figure 9.2 se puede observar la descripción realizada anteriormente, la parte izquierda de la imagen contiene las herramientas disponibles, cada una de ellas permite realizar una visualización diferente. En caso de realizar alguna modificación sobre esta parte provocará que se modifique la representación que hay a su derecha y se podrá ver como se comporta cada neurona al aplicarle los distintos filtrados.

La sección de SOM seleccionada y su configuración contiene:

- 4 menús desplegables (Figure 9.3)

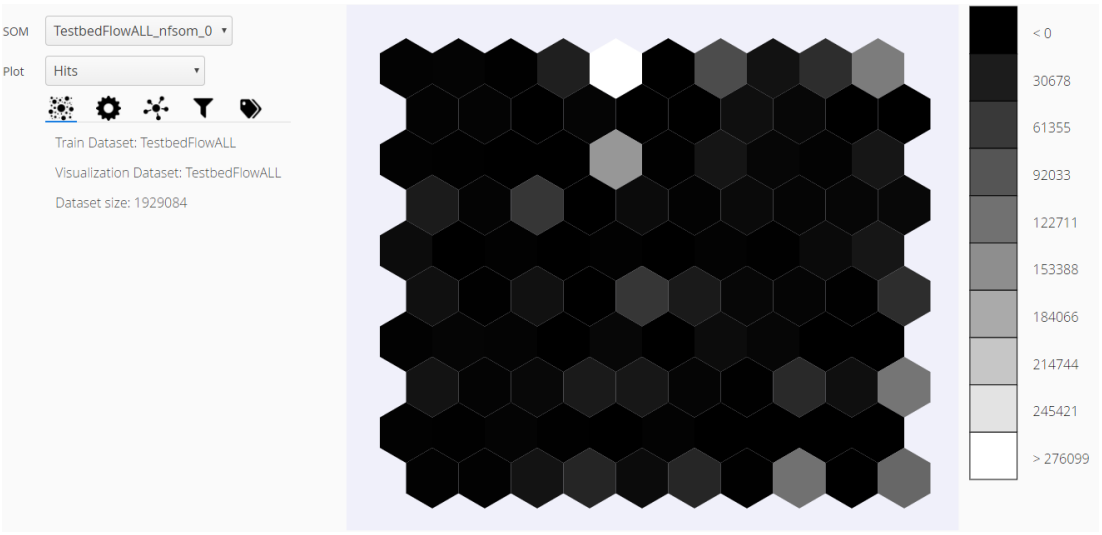


Figura 9.2: Vista inicial de la interfaz web.

(a) Desplegable Feature. (b) Desplegable Label.

Figura 9.3: Menús desplegables.

  - Som: Contiene todas las SOM que se pueden visibilizar. Siempre se encuentra visible.
  - Plot: Diferentes tipos de visualizaciones que se pueden aplicar a la SOM. Siempre se encuentra visible.
  - Feature: Muestra el etiquetado en función de los campos que contiene la SOM. Solo es visible cuando se selecciona «Labeled Plot» en Plot.
  - Label: Muestra los distintos etiquetados externos que tiene asignada una SOM. Solo es visible cuando se selecciona «External Labeled Plot» en Plot.

- 4 pestañas con una visualización cada una de ellas (Figure 9.4)

  - DatasetTab: Contiene información acerca de la SOM seleccionada
  - ControlTab: Contiene un slider que indica a partir de que porcentaje se considera que una neurona deja de tener la etiqueta «UNKNOWN»

Figura 9.4: Pestañas.

- NeuronTab: Contiene información acerca de la última neurona sobre la que se ha hecho clic, tiene un botón que permite descargarla.
- FilterTab: Contiene distintos filtros que se le pueden aplicar a la tabla junto con la opción de añadir nuevos filtros.

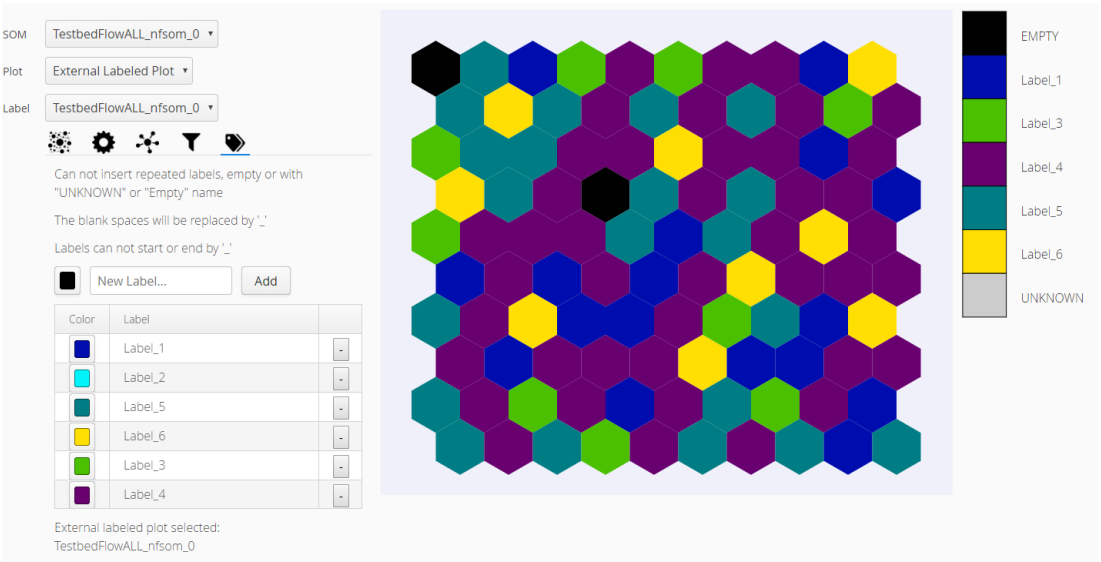


Figura 9.5: Visualización EE.

En función de los valores seleccionados en este apartado, los colores del mapa irán cambiando para que podamos visualizarlas correctamente las características seleccionadas como sucede en Figure 9.5 en la que estamos viendo como se representa un EE.

En Figure 9.6 podemos observar una simplificación del diseño que se ha realizado a la hora de crear el servidor web. La clase principal es «MyUI» que hereda de «UI» para utilizar todas las funciones que nos proporciona por defecto esta clase, y contiene todo los elementos de la visualización implementados en Vaadin[28], «SOMPlot»y «staticsReports» contienen funciones JS que son utilizadas para crear nuevas visualizaciones que no están implementadas.

«VisualizationService» contiene un grupo de funciones estáticas para poder utilizarlas sin necesidad de crear un objeto de esa clase, esto se debe a que para inicializar una instancia de esta es necesario utilizar datos almacenados en el servidor, cuando se obtiene toda la información necesaria se inicializa una variable desde la que se realizan llamas al servidor

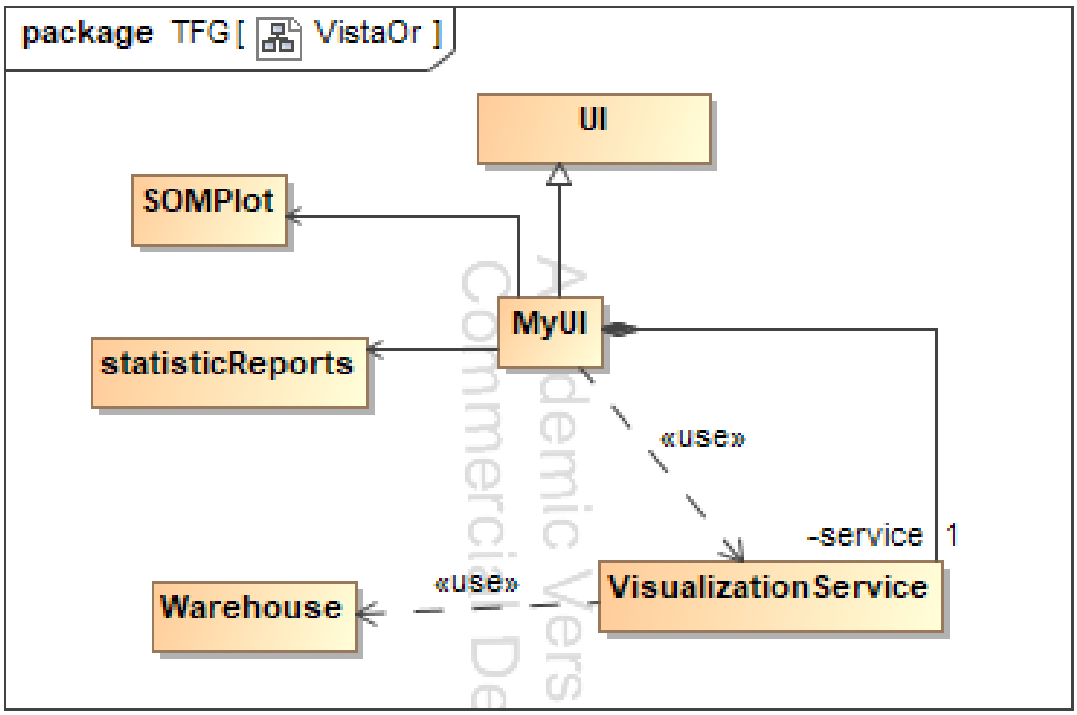


Figura 9.6: UML simplificado del servidor web.

para obtener los datos de las distintas visualizaciones. Es una capa de abstracción que se utiliza para separar las llamadas al servidor REST que están en «Warehouse» de la interfaz de usuario(IU) de «MyUI».

«Warehouse» está formada por un conjunto de funciones estáticas, cada una de ellas recupera distintos datos del servidor haciendo una petición HTTP. Aunque podrían ser utilizadas desde cualquier clase, únicamente se utilizan desde «VisualizationService» para mantener las abstracción entre la IU y las peticiones al servidor REST.

9.1.3 Controlador

Es el encargado de responder a las peticiones del usuario desde el servidor REST[26]. En concreto se encarga de recibir las peticiones de un cliente web, obtener la información necesaria en el Modelo y devolvérsela al usuario, al tratarse de peticiones web, se pueden realizar sin necesidad de utilizar la Vista aunque el sistema está desarrollado centrándose en las funcionalidades de ésta, ya que en el Controlador se crearán únicamente las peticiones que se implementen en ella.

Cada vez que se implemente una nueva funcionalidad para la Vista que implique obtener datos del modelo debe desarrollarse una petición desde la Vista al Controlador, este debe estar preparado para poder recibir esa petición y tratar los datos en caso de ser necesario, solicitarlos

al Modelo, realizar las comprobaciones pertinentes, modificarlos en caso de ser necesario y devolverle una respuesta al usuario.

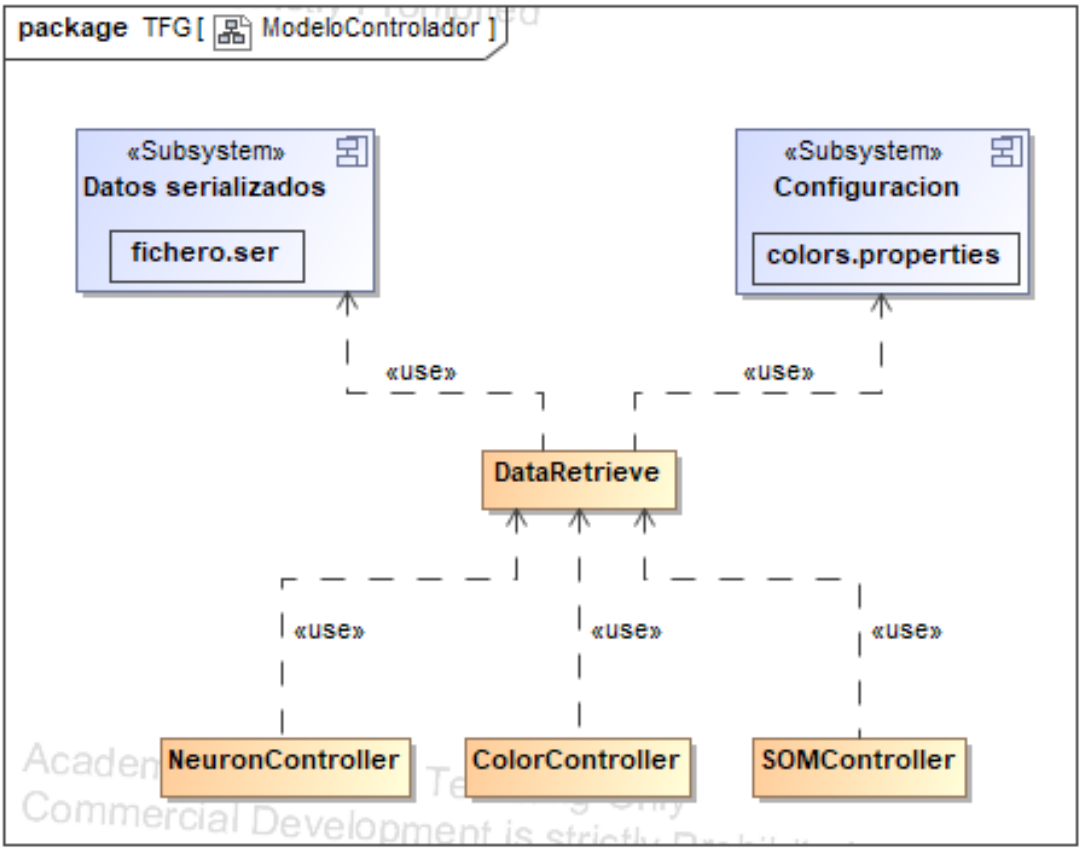


Figura 9.7: Comunicación Modelo-Controlador

En Figure 9.7 podemos observar la interacción entre el Controlador y el Modelo, donde el Modelo esta representado por «DataRetrieve» y todos los que hacen uso de este son parte del Controlador. Como las funciones que se utilizan del Modelo son estáticas se puede hacer una llamada directa a cada una de ellas sin utilizar un objeto de intermediario. El Modelo hace uso del fichero que contiene etiquetas y colores y accede a toda la información necesaria almacenada en distintos ficheros serializados.

9.2 Herramienta de etiquetado

El objetivo de este apartado es desarrollar una nueva herramienta para la interfaz web. Implementar un sistema que permita etiquetar las neuronas de una SOM en tiempo real. Este permitirá añadir nuevas etiquetas, modificar las existentes, eliminarlas en caso de no ser necesarias. Además, permite seleccionar el color que queremos que represente a cada una de ellas.

Todas estas funcionalidades estarán disponibles para los distintos EE que tenga una SOM[51], y deberá funcionar con distintas SOMs.

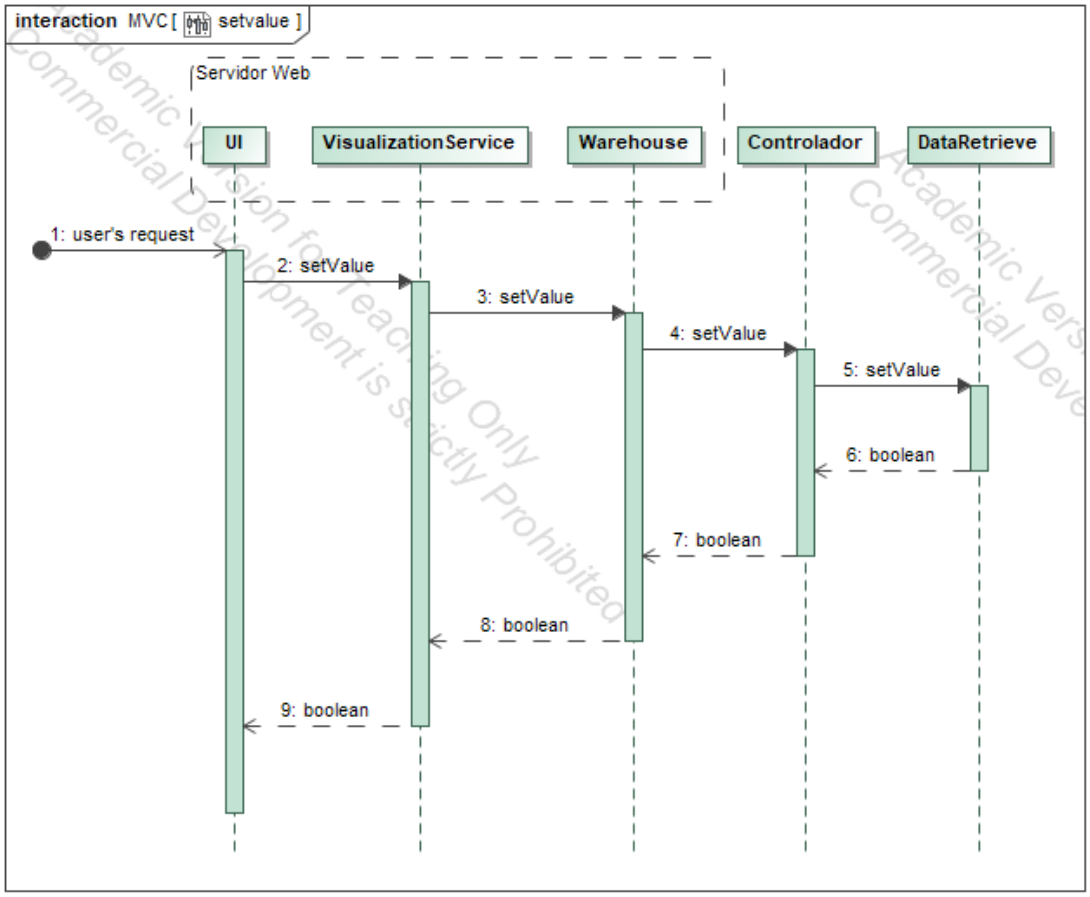


Figura 9.8: Modificar datos del servidor

Como se realizarán modificaciones en el servidor REST[26] la secuencia que se muestra en Figure 9.1 ya no cumple con los requisitos necesarios. Por tanto se ha creado un nuevo diagrama de secuencia generalista que representa como se hará cualquier modificación sobre los datos.

En Figure 9.8 podemos observar como se realizarán las modificaciones de los datos. En este diagrama el usuario indica mediante la IU los cambios que desea realizar. Esta realiza varias comprobaciones para corroborar que la modificación que se intenta realizar sigue las normas marcadas, a continuación se indica a «VisualizationService» los datos que queremos que modifique, y este se comunica con «Warehose» añadiéndole la ruta a la que debe realizar la petición. Este realiza una llamada al Controlador que comprobará que los datos introducidos son correctos y le indicara a «Dataretrieve», que representa al Modelo, el contenido que desea modificar. Este responderá con un valor boolean que indicará si se ha podido llevar realizar la

operación. En caso de que el Controlador detecte alguna anomalía en la información recibida o el Modelo no consiga llevar a cabo la operación la respuesta será «False» y en caso contrario «True».

Si la petición se esta realizando a través de la aplicación web esta informará al usuario del problema encontrado, si la comunicación es independiente del servidor web la respuesta recibida sera «False».

9.2.1 Desarrollo

El sistema actual permite visualizar distintos valores de la SOM, todos ellos son propios de la red, a excepción de uno, el EE. Los valores propios de la red aportan información de la distribución de los datos, si esta información fuese errónea, habría que modificar la configuración de la SOM o realizar de nuevo el entrenamiento para que los resultados se ajustarán más a lo esperado. Con el EE no ocurre este problema, ya que lo que indica es el nombre que le asignamos a los datos que se han asociado a una neurona, pero ese nombre no representa ningún valor real de la red.

Para permitir una visualización más intuitiva del EE podremos seleccionar uno que ya hay creados y ver como se distribuyen los datos en función del color asociado.

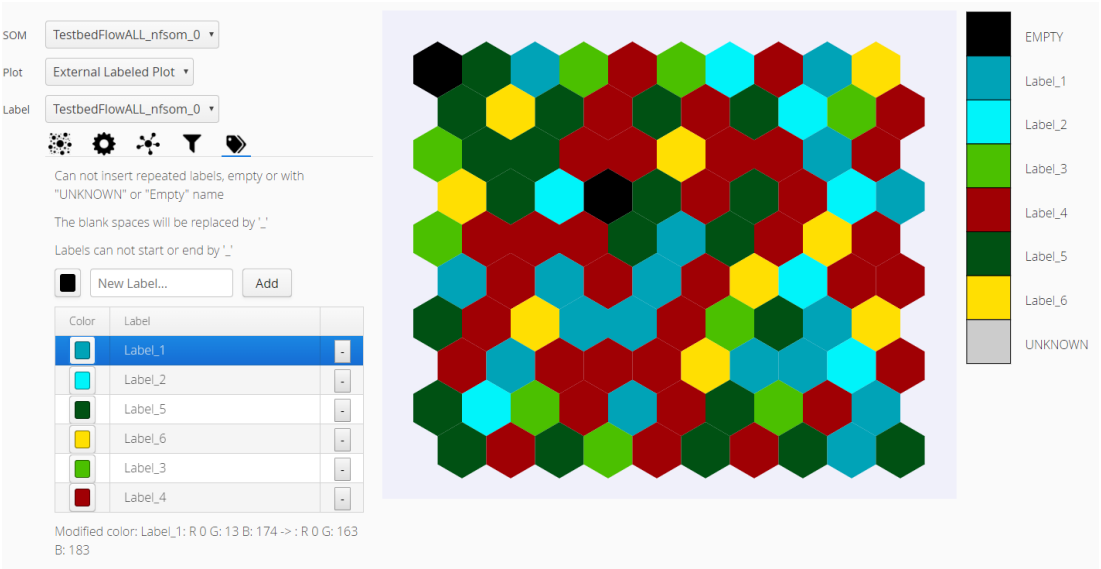


Figura 9.9: EE 1.

Como se puede observar en Figure 9.9 cada celda tiene asociado un color, y cada color una etiqueta, el color con el que se ha rellenando cada celda corresponde con el de la etiqueta que tiene asociada.

La herramienta que se va a desarrollar permitirá realizar en tiempo real las siguientes

acciones:

- Crear nuevas etiquetas.
- Modificar etiquetas.
- Eliminar etiquetas.
- Modificar color de las etiquetas.
- Seleccionar la etiqueta de una celda.

Cada vez que se realice una de estas acciones se plasmarán los cambios instantáneamente en el mapa.

9.2.2 Modificaciones de la interfaz de usuario

La aplicación sobre la que vamos a trabajar tiene 4 secciones y en cada una de ellas se realizaban un grupo de funcionalidades relacionadas, cada una de estas secciones esta identificada por una pestaña. Para mantener esta estructura crearemos una nueva pestaña que contendrá todas las funcionalidades relacionadas con el etiquetado. En Figure 9.10 podemos observar la nueva pestaña que se ha introducido.

Figura 9.10: Pestaña de etiquetado.

El contenido de la pestaña será el siguiente (ver Figure 9.11):

- **3 Labels** con información acerca del nombre de las etiquetas.
- **Layout horizontal** que se utilizará para crear etiquetas. Estará compuesto de:
  - **ColorPicker** que permitirá seleccionar el color de la etiqueta que se va crear.
  - **TextField** para escribir el nombre de la nueva etiqueta.
  - **Button** para crear una nueva etiqueta con el color del ColorPicker y el nombre introducido en el TextFiel.
- **Tabla** que contendrá la lista de etiquetas. Cada fila de la tabla contendrá 3 columnas:
  - **1ª Columna: ColorPiker** que permitirá cambiar el color de la etiqueta.
  - **2ª Columna: TextField** con el nombre de la etiqueta. Haciendo doble clic se podrá editar el nombre de la etiqueta.

  - **3ª Columna: Button** para eliminar la etiqueta de esa fila.

- **Label** para informar de la última acción que se ha realizado sobre el etiquetado.

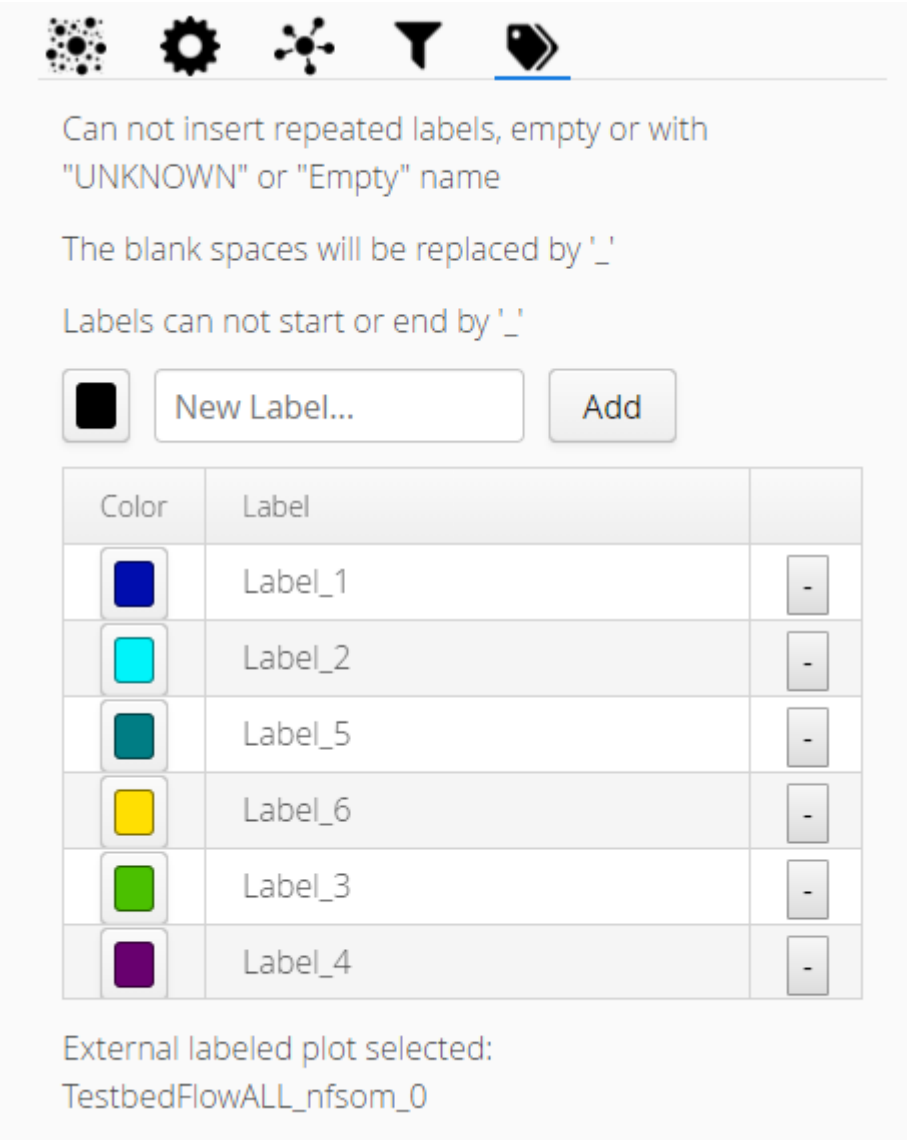


Figura 9.11: Contenido pestaña de etiquetado.

9.2.3 Funcionamiento

El comportamiento interno de toda esta vista no es un tema trivial, ya que hay que tener en cuenta multitud de casos.

Cada vez que se desarrolla una nueva funcionalidad en la Vista es necesario crear las operaciones equivalentes tanto en el Modelo como en el Controlador y realizar operaciones

de comprobación de que los datos son correctos tanto en la Vista como en el Controlador ya que, en la Vista son necesarias para informar debidamente al usuario de lo que esta ocurriendo y en el Controlador porque se pueden comunicar con él sin necesidad de utilizar la Vista.

Al crear o al modificar el nombre de las etiquetas existen ciertas restricciones:

- Se comprueba que la etiqueta no tiene el nombre vacío, no se puede llamar «UNKNOWN» y tampoco «Empty», ya que estos son nombres reservados.
- Se comprueba que la etiqueta que se va a introducir no tenga el mismo nombre que una que ya existe. Esto se debe a que en el fichero que se almacenan las etiquetas se guardan pares clave valor y no pueden existir dos claves iguales.
- Se comprueba si existen espacios o '_' al principio o al final de la cadena y se eliminan.
- Se buscan todos los espacios que hay en la cadena y se substituyen por '_', esto se debe a que en el fichero donde se almacenan las etiquetas los espacios se tratan de forma especial y supone un problema su almacenamiento y recuperación.

Cuando se intentan añadir o modificar los colores también tienen ciertas restricciones. El rango en el que se deben añadir es ([0,255], [0,255], [0,255]), no es necesario realizar una comprobación en la Vista ya que el ColorPicker gestiona automáticamente estos rangos, pero si es necesario en el Controlador. Cuando se quiera seleccionar un color, se tendrá que hacer clic sobre el ColorPiker  y se abrirá una ventana emergente como la que se muestra en Figure 9.12 en la que se permitirá elegir que color queremos que se le asigne a esa etiqueta.

Cada vez que intentemos cambiar un valor del servidor, este nos responderá con una variable boolean que nos indicará si se han realizado los cambios correctamente.

Crear etiquetas

En caso de que el usuario desee crear una nueva etiqueta deberá seleccionar un color en la selección de color, escribir un nombre en el cuadro de texto y pulsar el botón para añadir la etiqueta. Cuando se hace clic en el botón se comprueba que los datos introducidos en el cuadro de texto son correctos y se realizan las modificaciones necesarias.

Si se intenta crear una etiqueta que no respete las restricciones mencionadas anteriormente, se informará al usuario a través del cuadro de texto con un mensaje de error y a través del cuadro de información de acciones.

Por defecto el color seleccionado es el (0,0,0), si se desea cambiar el color, habrá que hacer clic en la selección de color, seleccionar un color y darle a aplicar para que se almacene en el fichero «colors.properties». Si añadimos una nueva etiqueta el cuadro de texto se vaciará pero la selección de color mantendrá el último seleccionado.

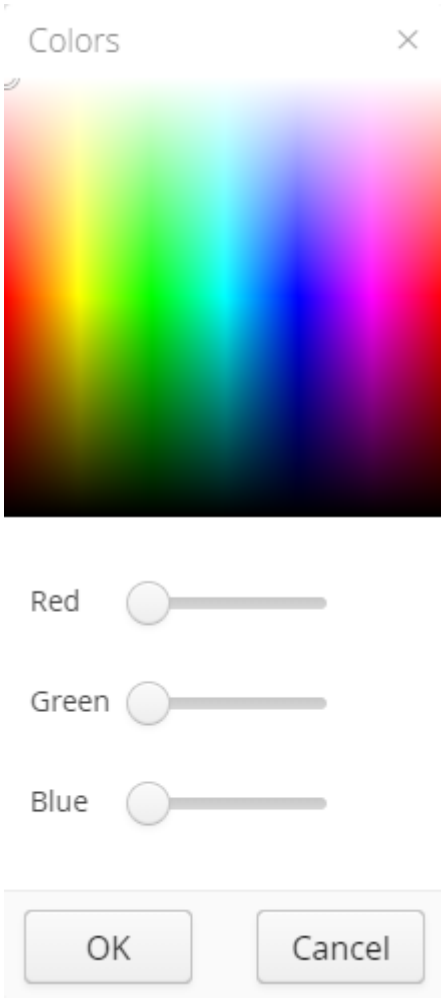


Figura 9.12: Ventana emergente ColorPiker.

Modificar y eliminar etiquetas.

En caso de desear cambiar una etiqueta, entonces debemos hacer uso de la tabla, si lo que se quiere modificar es el color habrá que hacer clic sobre la selección de color, y seleccionar en la paleta de colores el color que queremos que tenga esa etiqueta.

Para realizar esta acción es necesario modificar el fichero «colors.properties», después de haber hecho las comprobaciones pertinentes de que los datos son correctos, se debe acceder al fichero y a través del nombre de la etiqueta se elimina ese par clave valor y se crear uno nuevo con esa misma etiqueta pero con el color que se le ha indicado. Por último, tras guardar las modificaciones se actualizará el mapa y podremos observar como todas las celdas con la etiqueta modificada cambian de color.

Si deseamos cambiar el nombre de la etiqueta, se debe hacer doble clic sobre el sobre ella, se desplegará una caja de texto editable, como la que se muestra en Figure 9.13, que

permitirá modificar los datos almacenados. Al pulsar en «Save»se comprobará que los datos son correctos y se modificarán si es necesario, se enviarán al servidor que actualizará el fichero «colors.properties» y a continuación se cambiará el nombre de la etiqueta antigua. En caso de seleccionar «Cancel» volverá a su estado previo.

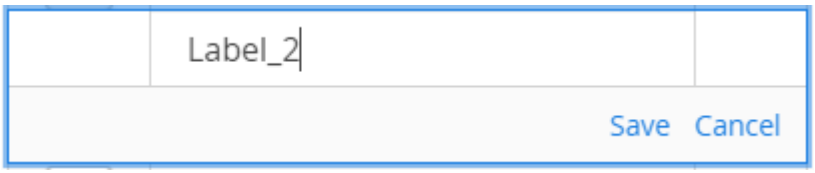


Figura 9.13: Modificar etiqueta.

Al modificar el nombre de una etiqueta también se deben cambiar las etiquetas asignadas a las neuronas. Para realizar este proceso es necesario enviar un mensaje al servidor, este desserializará el fichero que contiene la información acerca de la etiqueta que tiene asignada cada neurona, buscará todas las que la tengan asignada y las modificará, por último volverá a serializar el documento almacenando todos los cambios realizados

Cuando lo que queremos es eliminar una etiqueta, habrá que hacerlo clic en - de la fila correspondiente para intentar eliminarla. Es completamente necesario que no este asignada a ninguna neurona, en caso de que lo este se informará al usuario a través del cuadro de información de acciones que no se ha podido eliminar esa etiqueta.

Cambiar etiqueta seleccionada.

Cuando se haga clic sobre una neurona, pasará a estar seleccionada, en ese momento en la tabla se marcara la fila de la etiqueta que tiene asignada, como se muestra en Figure 9.14.

Haciendo clic sobre otra fila se cambiará la etiqueta de la neurona. Si la neurona seleccionada tiene la etiqueta "", que en la leyenda se muestra como "Empty", no se puede asignar ninguna etiqueta, al igual que a las etiquetas que tiene otro valor asignado modificable no se le puede asignar ninguno de estos dos valores.

La etiqueta "UNKNOWN" es otro caso especial, de esta etiqueta nunca se almacena su valor ya que se calcula en tiempo real. Si seleccionamos la pestaña ControlTab podremos ver un Slider, este al desplazarlo hacia la izquierda o la derecha indicará el porcentaje de pureza necesario para que una neurona tenga asignada una etiqueta. Si la pureza de esa neurona es inferior a el porcentaje indicado, se mostrará como "UNKNOWN" en el mapa, pero no se modificarán los datos almacenados.

Cuando hablamos de la pureza de una neurona nos referimos al porcentaje de datos que hay en una neurona con la misma etiqueta externa que tiene asignada.

Se permite modificar el color y el nombre de las etiquetas sin necesidad de que estas estén

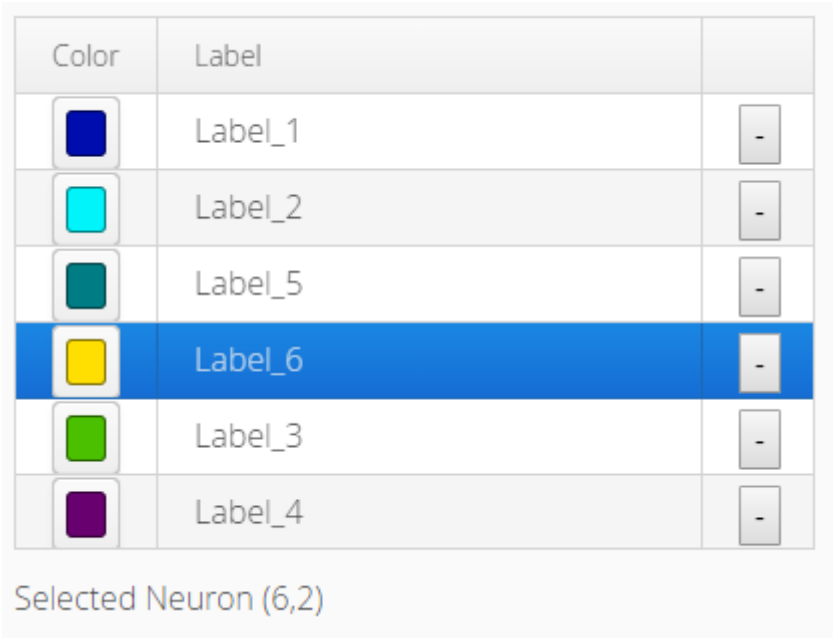


Figura 9.14: Etiqueta seleccionada.

seleccionadas.

Cuando se inicia la aplicación, la neurona seleccionada por defecto es la (0,0), y solo cambia cuando se hace clic sobre otra neurona.

Seleccionar etiquetado externo

La cantidad de etiquetados externos que se pueden tener es ilimitada, pero como mínimo siempre debe haber uno, este debe tener el mismo nombre que el de la SOM. Estará siempre seleccionado por defecto y su tabla de etiquetas creada. Cada vez que se cambie se actualizarán los valores de la tabla a los del etiquetado seleccionado, esto se guarda hasta que se recargue la UI. Se mantendrá la última neurona seleccionada en caso de seleccionar alguna otra visualización de los datos y se conservará el último etiquetado seleccionado pudiendo editarse sin necesidad de estar visualizándolo.

Si lo que se cambia es la SOM seleccionada, entonces la visualización volverá a cambiará a la por defecto, se actualizarán todos los valores de la tabla, la neurona seleccionada pasará a ser la (0,0) y se seleccionará el etiquetado externo por defecto.

Descargar neurona

Actualmente el usuario puede descargarse cada una de las neuronas de manera individual, estas contienen los datos originales que se han almacenado en ellas. La etiqueta externa que se le ha asignado no se almacena ya que esta se guarda en un fichero serializado y no se

modifican los datos originales. Para solucionar este problema se cambia el nombre del fichero que se va descargar, se obtiene la etiqueta asignada de esa neurona del último etiquetado externo seleccionada y se añade al final del nombre del fichero.

9.2.4 Implementación

Cliente

Antes de realizar el desarrollo de la herramienta se ha modificado el diagrama Figure 9.6. Se han introducido clases en las que se desarrollarán nuevas funcionalidades. Con estas clases se pretende desarrollar una herramienta modular y conseguir una herramienta lo más independiente posible de la IU ya creada.

En Figure 9.15 se pueden ver las clases que se han añadido. El Figure A.1 contiene UML completo de todas las modificaciones realizadas sobre la aplicación.

La herramienta se ha diseñado lo más independiente posible del resto de la UI, Por ello se ha creado una nueva clase que contenga todos los elementos descritos hasta el momento. En la clase principal lo único que se hará será añadir los distintos componentes de los que esta compuesta. La herramienta no puede ser completamente independiente ya que es necesario actualizar la visualización de la IU cada vez que se realiza una modificación y para ello necesita tener una instancia de la aplicación.

Es necesario que esta clase tenga un constructor vació ya que cuando se inicializa la herramienta todavía no se han cargado todos los datos necesarios para poder inicializarla. En el momento en el que se obtienen todos los datos necesarios del REST[26] se inicializa de nuevo, esta vez con todos los valores necesarios para poder crear la tabla. Cada vez que se seleccione un etiquetado externo distinto o se cambie de SOM se inicializará de nuevo la tabla con todos los datos referentes al etiquetado externo seleccionado.

Se ha creado una clase que representa cada fila de la tabla, por tanto en la clase ”TableColorsLabel”, la principal de la herramienta, será necesario tener una lista del objeto que representa cada fila.

La creación, eliminación y modificación de filas son independientes de los componentes que las realizan. En caso de querer realizar estas acciones de algún otro modo se podrían reutilizar estas funciones. Lo mismo ocurre con la función que selecciona en la tabla que fila tiene asociada una etiqueta.

La creación y eliminación se realizan sobre la lista del objeto fila, mientras que la modificación se dentro del objeto. En el momento que se modifique el nombre que se almacena en el modelo se modificará también el nombre del objeto para asegurar que el proceso realizado es atómico y no se realizan cambios en la UI hasta que se almacenan en el fichero ”colors.properties”.

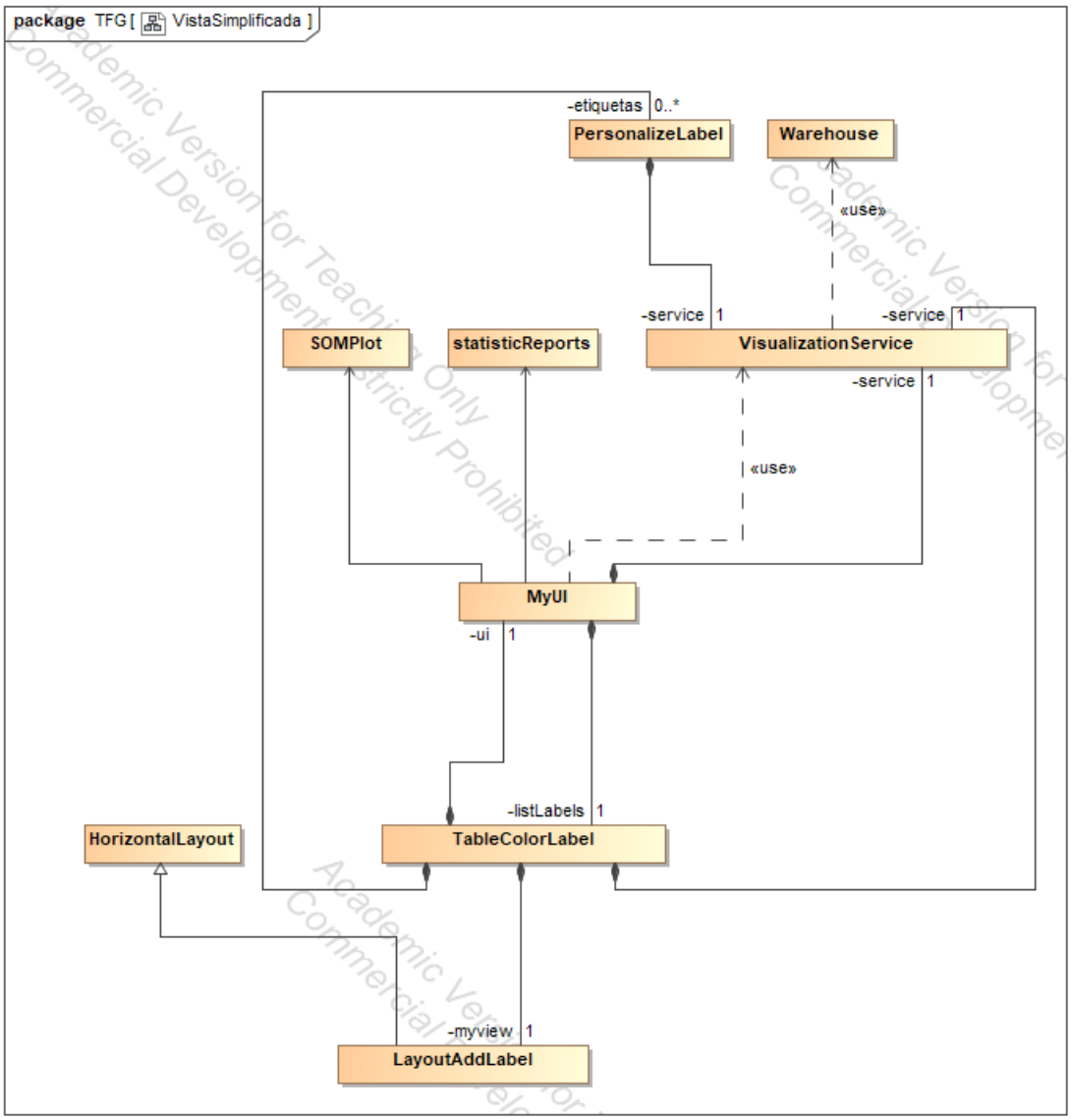


Figura 9.15: UML Modificación del servidor web.

Fue necesario crear una función que se encargará de que, en el caso de seleccionar una neurona que contenga la etiqueta con un valor reservado, la tabla cambie a modo Solo Lectura para no poder asignarle ninguna etiqueta. Si se selecciona cualquier elemento al que se le puede modificar la etiqueta, se pondrá en modo Selección Única, para que se pueda seleccionar un único elemento.

Todos los componentes de Vaadin[28] que se crean en la clase ”TableColorsLabel” deben tener una funcione ”getter” para poder añadirlos a la clase principal. De todos estos elementos se puede cambiar su contenido, ya que los ”getters” lo permiten pero no se puede modificar los elementos en si mismos ya que no se han creado ”setters”. Lo mismo ocurre para los de la

clase ”LayoutAddLabel”.

Se ha creado un Layout horizontal para añadir nuevas etiquetas. Su contenido se almacena en la clase ”LayoutAddLabel” y contiene 3 elementos. Para poder añadirlos a la clase principal es necesario que todos ellos tenga un ”getter”. El ”ColorPicker” y el ”Button” además deben tenerlo para indicarles que acciones deben realizar cuando se pulsa sobre ellos. Esta funcionalidad se controla desde la clase ”TableColorsLabel” ya que es donde esta la lista de objetos fila.

Servidor

Al igual que en el caso del cliente se ha realizado un diagrama que contiene todas las modificaciones realizadas durante todas las etapas de implementación. Podemos encontrar el diagrama completo en Figure A.2. Podemos observar una simplificación del UML completo, en este encontraremos una clase nueva que se utilizará en posteriores apartados. Para este apartado nos interesan todas las funciones que se han insertado en las clase que ya había creadas en Figure 9.7. Estas funciones se utilizan para implementar en el servidor todas las funcionalidades indicadas anteriormente.

Para insertar nuevos elementos en el fichero ”colors.properties” se utiliza la misma función del Modelo que para actualizarlos, esto se debe a que las librerías que trabajan con ficheros ”.properties” no distinguen entre actualizar y crear un nuevo par. Pero en el Controlador hay dos funciones distintas, una para insertar y otra para modificar, en las cuales se hace la comprobación de que se esta realizando el uso adecuado de estas funciones.

En el Controlador se realizan todas las comprobaciones necesarias para saber que los datos introducidos están en el formato correcto, aunque estas comprobaciones ya se estén realizando en la IU.

Todas las funciones de modificación de datos que hay en el controlador devuelven un valor ”boolean” indicando ”true” si se ha realizado correctamente la modificación o ”false” en caso de no haberse podido realizar correctamente. En la vista, a demás de realizarse las mismas comprobaciones, también se comprueba que se han introducido los datos correctamente en el modelo, en caso de que hubiera algún error no se introducirían y se informaría al usuario.

9.3 Visualización de estadísticas

Se implementara una nueva funcionalidad en la que se pretende que el usuario tenga más información de la SOM. Se genera una nueva visualización que nos permite analizar datos estadísticos de una neurona. Todos estos datos ya están precalculados y por tanto solamente es necesario realizar las operaciones pertinentes su obtención.

9.3.1 Descripción del sistema

Esta visualización se activará cuando el usuario haga doble clic sobre una neurona, el mapa desaparecerá y en su lugar se mostrará un conjunto de estadísticas referentes a los datos que hay almacenados en esa neurona. Además de esto, cambiará de pestaña a NeuronTab para que podamos observar la información de esa neurona y se creará un botón que al pulsarlo volverá a mostrar el mapa.

En la parte superior se mostrarán un conjunto de gráficos de sectores que representan las variables categóricas que se almacenaron en el apartado Resultados del análisis. En cada uno de los gráficos habrá al menos un color y como máximo tantos como categorías existen en esa variable. Al situarnos sobre uno de los colores con el ratón aparece una etiqueta con el nombre de la categoría asociada al color y la cantidad de elementos de esa categoría. El circulo completo representa el total de datos de una neurona y cada color ocupa una parte proporcional a la cantidad de elementos que hay en ella.

Justo debajo de estos gráficos aparecerá una tabla con tres columnas, la primera contiene los nombres de los valores numéricos que se han almacenado y a continuación la media y la desviación típica de todos los datos de esa neurona para cada valor de la primera columna(ver Figure 9.16).

9.3.2 implementación

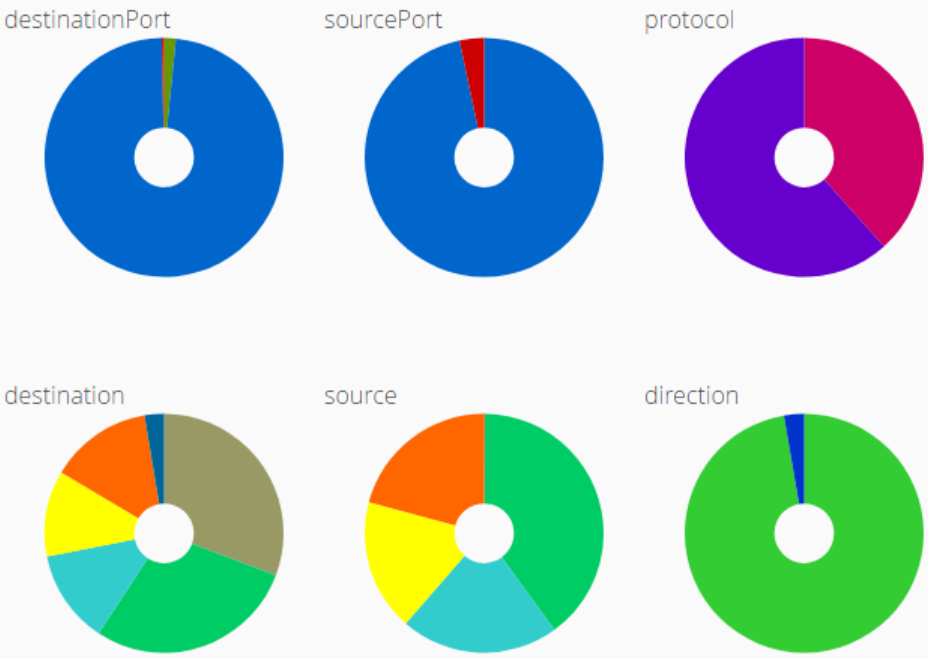
Como la funcionalidad del doble clic es necesario implementarla en JS ya que se capturan los clics realizados sobre el mapa y este se ha generado con JS. Cada vez que se captura doble clic sobre una neurona le hace una llamada desde JS a una función de Vaadin[28] que obtiene todos los datos necesarios para crear la visualización.

En caso de no tener ningún dato asociado se informará al usuario, como se puede ver en la imagen Figure 9.17.

Si esa neurona contiene datos se podrá ver la misma estructura que en Figure 9.16 pero con unos datos distintos. Para generar esta ventana es necesario crear un nuevo ”div” en el que se introducirá toda la visualización. A continuación se genera dos estructuras con los datos de obtenidos, una para crear los gráficos y otra para la tabla.

Para desarrollar la visualización de los gráficos se ha decido utilizar las librerias JS de D3JS[19], estas están especializadas en generar representaciones gráficas a partir de datos.

A la primera estructura debemos indicarle el nombre y la cantidad total de datos. A continuación el nombre de un grupo de los datos, el porcentaje que representa esa cantidad de datos y por último el color del que queremos que pinte esa región. El siguiente paso sería realizar una llamada a una función JS, de D3JS para genere los gráficos y se insertar en el ”div”.



Characteristic	Mean	Standard deviation
duration	0.0000000000	0.0000000000
destAvgBytes	0.0000000000	0.0000000000
destPortRF	-0.5748265610	0.0127960594
sourceRF	22.3280475719	0.0372188830
sourceAvgBytes	0.0000000000	0.0000000000
destRF	1.2398414272	0.0275997763

Figura 9.16: Estadísticas.

La segunda estructura se utilizara para generar la tabla. Esta contendrá para cada fila el nombre, la media y la desviación típica de la variable numérica. Por último se invocará una función JS que leerá la información y generará una tabla en código HTML y finalmente la insertará en el ”div”.

9.4 Visualización de conjuntos

La información que el usuario posee acerca de cada neurona es la cantidad de datos que contiene. Para conocer más sobre esta vamos a desarrollar una nueva funcionalidad que permita observar la distancia del prototipo de la neurona a cada uno de los datos asociados a ella.

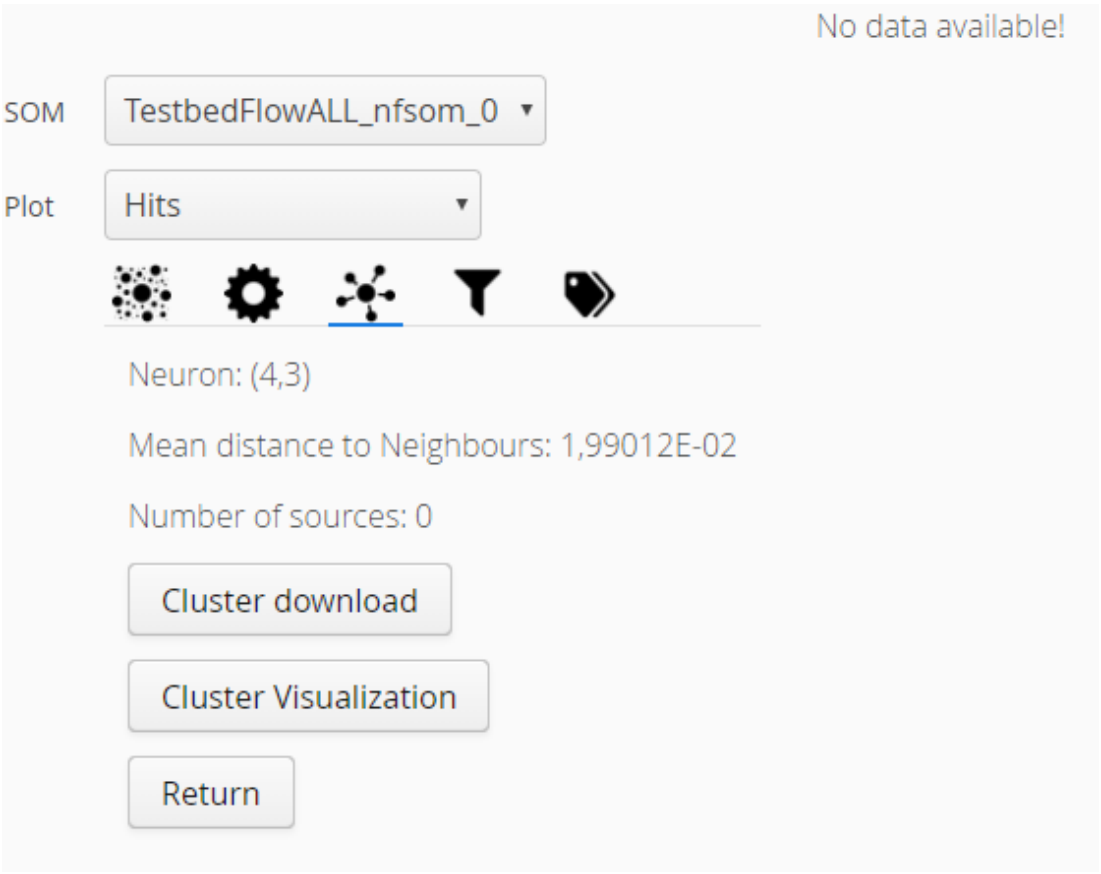


Figura 9.17: Estadística vacía.

Cabe destacar la importancia de esto, ya que hoy en día en *Big Data* existen dos grandes problemas, el más conocido es cómo tratar tal magnitud de datos, el otro problema es como podemos observarlos, ya que sin una visualización óptima de los datos resulta inviable analizarlos. En este apartado, explicaremos nuestra proposición para solucionar estos dos problemas.

9.4.1 Precálculo de los datos

Las distancias están almacenadas en en un fichero serializado, esto supone un problema debido a que ese fichero es demasiado pesado para obtener los datos en tiempo real y mostrarlos. Por este motivo se creará otro fichero serializado que contenga únicamente datos relacionados con las distancias y modificados de tal forma que se puedan obtener y mostrar en tiempo de ejecución.

Para facilitar la modificación, obtención y visualización, se han creado dos objetos: CustomTotalDistance y CustomDistance.

- **CustomDistance**: Representa la distancia de un dato al prototipo de la neurona
  - Distancia que hay desde un elemento al prototipo.
  - Rango superior para agrupar distancias parecidas.
  - Rango inferior para agrupar distancias parecidas.
  - Diferencia entre el rango superior e inferior.
- **CustomTotalDistance**: Contiene información acerca de la neurona y las distancias desde los datos al prototipo.
  - ID que representa a la neurona.
  - Lista de CustomDistance para saber cuantas distancias distintas hay en una neurona.
  - Número de elementos de la lista.
  - Máxima distancia de la lista.
  - Mínima distancia de la lista.

El fichero serializado será una lista de CustomTotalDistance.

Para generar esta lista se deben realizan varias operaciones sobre los datos, estas estarán separadas en distintas fases:

1ª Fase: Para cada neurona se obtienen las distancias de todas sus observaciones al prototipo, si ya existe esa distancia en la lista de CustomDistance se incrementa contador de elementos que tienen y si no existe, se añade un nuevo elemento a la lista. Cuando se han recorrido todas las distancias de todas las neuronas, se indica la diferencia que hay entre el rango superior e inferior para agrupar los valores. En la primera iteración el rango inferior es 0 y el rango superior es la diferencia entre el superior y el inferior, en cada iteración el rango inferior pasa a ser el superior de la iteración anterior, y el rango superior aumenta la diferencia indicada. Todas las distancias que, dentro de una neurona, pertenezcan a un rango se agruparán y se almacenará la cantidad total de datos que hay en ese rango. Se asignará un grupo a cada uno de los rangos que contengan valores, y los que no, son eliminados.

Explicación: Se realiza este agrupamiento porque hay distancias similares, pero al ser distintas, por cada una de ellas se genera un nodo en la Vista y resulta inviable mostrar tantos elementos al mismo tiempo

2ª Fase: Se asigna como distancia para cada CustomDistance el valor medio entre los dos rangos y se normaliza. Se obtienen el múltiplo de 10 mínimo para que la distancia mínima de cada neurona supere 10, a no ser que esa distancia sea 0, en ese caso se le asignará el valor 1. Todas las distancias se multiplicarán por el múltiplo de 10 calculado y se realizará una normalización logarítmica, el resultado de esta operación se multiplicará por 100.

Explicación: Es necesario realizar todo este cálculo por los siguientes motivos:

- Se busca el valor mínimo superior a 10 debido a que valores más pequeños en una normalización logarítmica no serian lo suficientemente grandes y se situarían demasiado cerca del prototipo para su visualización.

- Es necesario realizar la normalización logarítmica dado que la diferencia entre el valor máximo y mínimo puede ser muy grande y lo que nos interesa tener una buena visualización, en caso de tener una gran diferencia entre estos dos valores no se podría tener una visualización intuitiva y la normalización logarítmica, permite aproximar datos muy lejanos manteniendo el orden entre ellos.

- El motivo por el que se multiplica por 100 es para tener una correcta visualización de los nodos, para evitar que aparezcan demasiado cerca del prototipo. Esto se debe al funcionamiento de la librería, esta tiene un sistema de distancias propio, y en caso de tener valores menores que 100 implica que estén demasiado cerca del centroide.

3ª Fase: Se ordena la lista de CustomTotalDistance en función de los IDs. En caso de que haya neuronas que no tengan elementos no estarán en esta lista. El siguiente paso es añadir los elementos que no estaban el la lista con los IDs de las neuronas vacías y se indica que tienen 0 elementos.

Explicación: Esto simplifica la obtención de datos en el Modelo, ya que si se quieren recuperar los datos de una neurona, en lugar de recorrer toda la lista bastará con conocer su ID.

9.4.2 Modificar IU

En la pestaña NeuronTab podemos encontrar información sobre la última neurona seleccionada, por tanto es el lugar indicado para insertar la opción de visualizar la distribución de los datos. Se crea un botón que permitirá realizar esta acción. Al pulsar este botón se obtendrán los datos serializados en el apartado anterior y mostrará un grafo con la distribución de estos. El mapa será substituido por el grafo. Para volver a visualizar el mapa, aparecerá un botón en las pestaña que nos permitirá a mostrarlo de nuevo. El resultado final de esta pestaña es el de Figure 9.18.

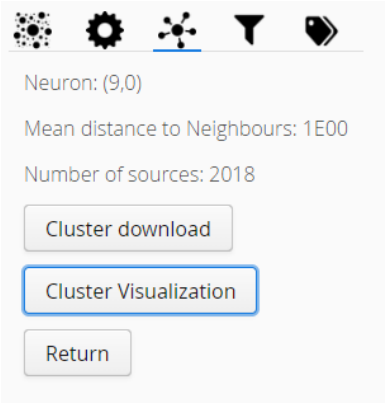


Figura 9.18: NeuronTab

Al pulsar de nuevo el botón de mostrar los datos, se actualizara la vista del grafo

Visualizar datos

Debido a la gran cantidad de información que puede haber asociados a una neurona, no es viable mostrar un nodo por cada dato. Por lo que se ha decido que cada uno represente un conjunto de datos que pertenezcan al mismo rango. El número de datos que representarán a un nodo se calcula en función del total de datos que hay en la neurona pudiendo crearse varios nodos para el mismo rango en caso de que el número de datos sea mayor que los que puede representar un nodo.

El tamaño cambiará en función de la cantidad de datos que hay almacenados en un nodo. Esta librería escala el tamaño de los nodos en función del valor máximo y mínimo asignado, por eso mismo no varía el tamaño de los nodos entre dos representaciones diferentes si la cantidad de datos asociada a un nodo es diferente.

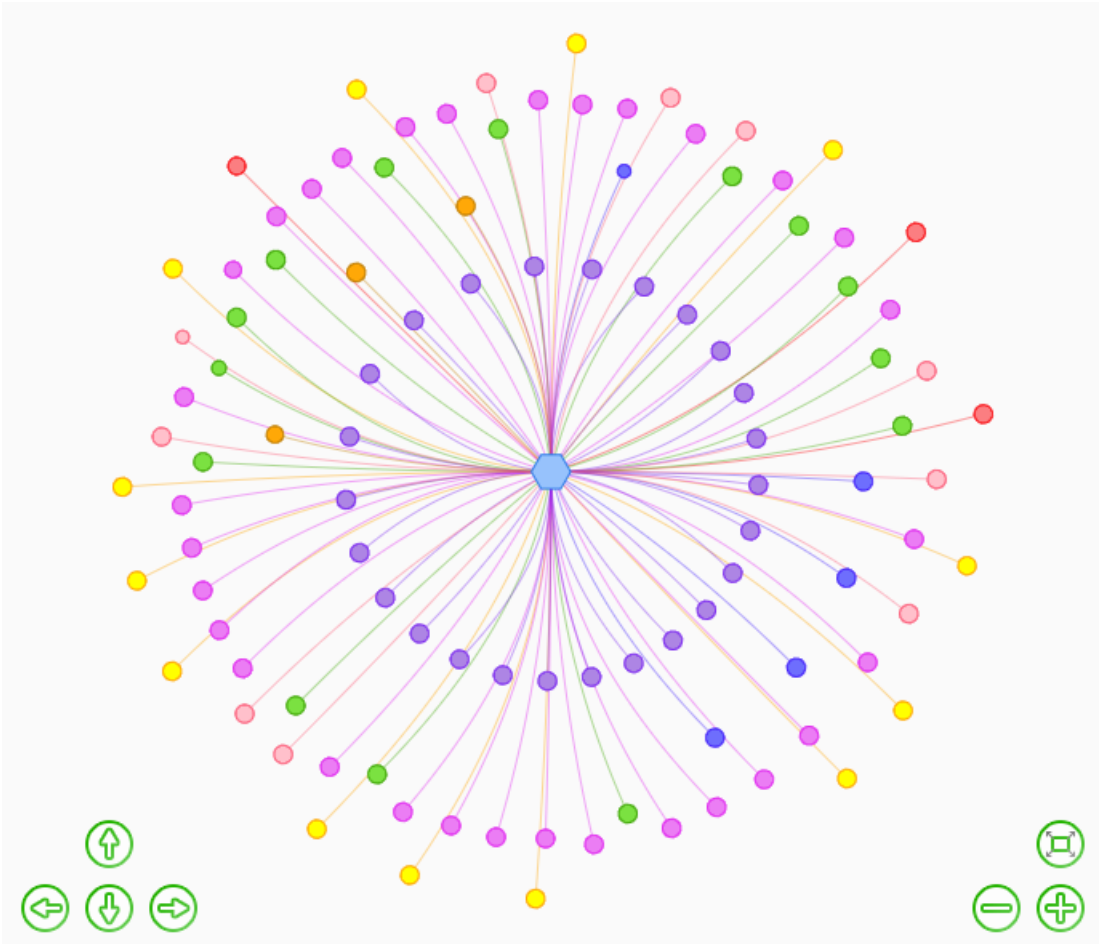


Figura 9.19: Distribución de los datos

Los que representen el mismo rango tendrán el mismo color y estarán situados a la misma distancia del prototipo ya que pertenecerán al mismo grupo. En Figure 9.19 podemos observar

un ejemplo de como se distribuirán los nodos.

Por la forma en la que funciona la librería, los nodos que forman parte de un mismo grupo tienen siempre el mismo color. La cantidad de colores asociadas a grupos es limitada así que si se supera el número máximos de grupos los colores de dos de ellos pueden ser el mismo.

En caso de querer más información acerca de algún nodo se podrá situar el ratón encima y se mostrará la siguiente información: rango al que pertenece, cuantos elementos contiene del total que puede representar y a que grupo pertenece (ver Figure 9.20). El único nodo que no contiene esta información es el que representa el prototipo de la neurona.

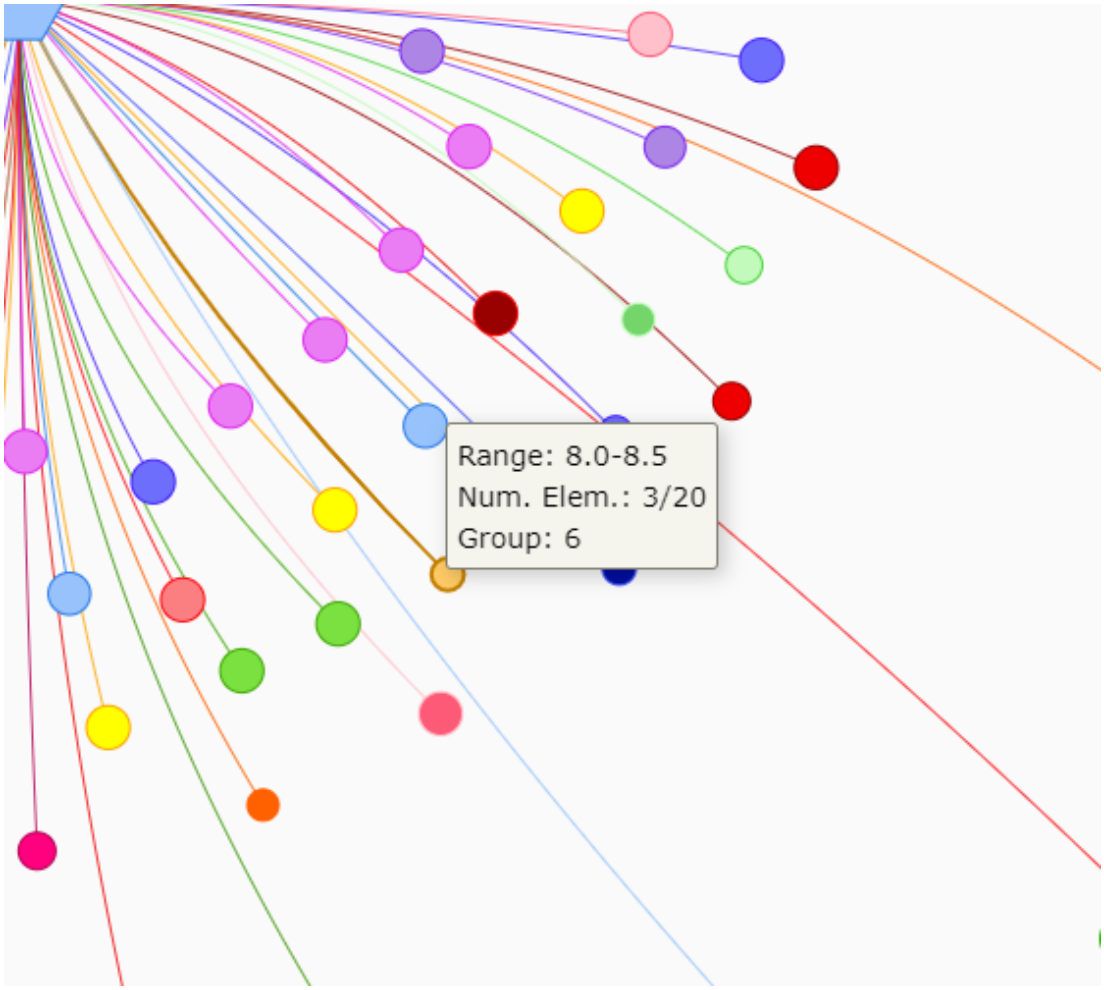


Figura 9.20: Información que contiene un nodo

Ya que el objetivo de esta visualización es permitir al usuario observar la distribución de los datos y poder obtener más información acerca de una neurona, el grafo será interactivo. Si hacemos clic sobre un nodo y lo arrastramos, este seguirá al ratón y el resto de nodos se desplazarán con él, cuando se suelte, los nodos se volverán a ajustar en función de su distancia

al prototipo. Si hacemos scroll sobre la visualización se hará zoom sobre el grafo y si hacemos clic sobre el fondo y desplazamos el ratón, podremos desplazar el grafo completo. Todas estas funcionalidades, junto con la de centrar el grafo en la vista, también se pueden realizar con los botones indicados en Figure 9.21 están situados en la parte inferior izquierda y derecha.

(a) Desplazar. (b) Zoom.

Figura 9.21: Interacción grafo.

Implementación

Antes de comenzar la implementación se han añadido al diseño las nuevas clases y funciones que se utilizarán es este apartado. El diseño completo de la Vsta esta en Figure A.1 y la del Modelo y Controlador en Figure A.2.

En la Vista únicamente se han añadido funciones, pero en el Controlador ha sido necesario crear una nueva clase para mantener la lógica que guardan las clases ya que cada una de ellas tiene un objetivo. En Figure 9.22 podemos ver una simplificación.

Como la librería de visualización se implementa en JS, es necesario indicarle a Vaadin que función debe invocar y se le pasarán por parámetro los datos que queremos que muestre.

Lo primero que se debe hacer es obtener los datos de la neurona. Para ello se hace una petición al servidor cuando se pulsa el botón de mostrar el grafo, indicándole la neurona seleccionada para obtener sus datos. Este busca en el fichero serializado y nos devuelve un objeto CustomTotalDistance que contiene el mismo ID que la neurona.

Uno de los datos contenidos en el objeto es el número total de elementos que hay en una neurona, con este valor se calcula cuantos de ellos podrá representar cada nodo. La agrupación de los datos se debe a limitaciones de la librería, si se intentan crear un número demasiado alto de nodos los tiempo de carga aumentarán demasiado y en caso de que lleguen a mostrarse podría ralentizar la aplicación o no distribuirse correctamente.

La cantidad de elementos que puede representar un nodo se cuantifica en tiempo real, esto se hace para mantener separados el precálculo y la visualización. En caso de desear realizar una visualización diferente, no será necesario modificar los datos calculados previamente.

El cálculo del número de elementos que podrá contener un nodo se realiza a partir de la cantidad total de datos que tiene la neurona, para ello se obtienen los dos dígito más representativos y la cantidad total de dígitos que tiene.

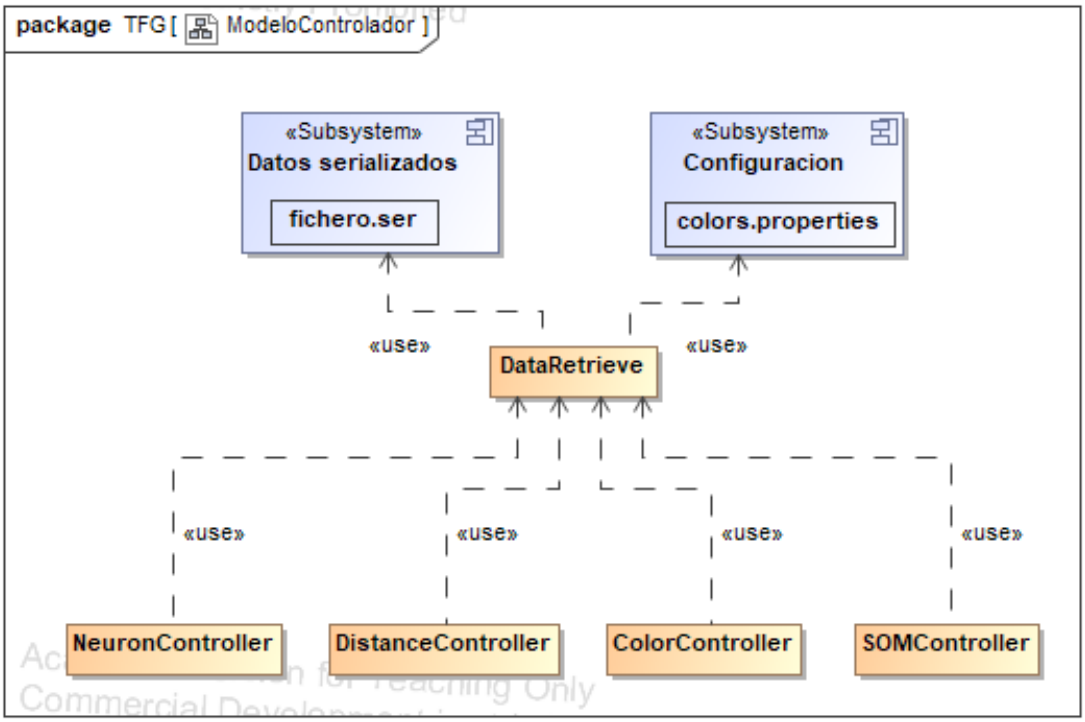


Figura 9.22: UML simplificado Modelo-Controlador

Para el caso en el que estamos trabajando, a partir de múltiples pruebas, se ha llegado a la conclusión de que el número máximo de nodos que debería representarse seria 1000, así que se ha limitado a 990, pero teniendo en cuenta que con el precalculado que se ha realizado es muy poco probable que llegue a crearse un número tan alto de nodos.

Conociendo la cantidad total de datos se realizan un conjunto de operación que limitan el número de nodos a un máximo y calculan el total de datos a los que puede representar un nodo.

A partir de los datos calculados se generan dos estructuras, una que contiene todos los nodos que se van a crear y la información de cada uno de ellos y otra con la unión que hay entre cada nodo y el prototipo.

Con estas dos estructuras se ejecuta la función JS que se encarga de crear la visualización y indicándole estas estructuras por parámetro. Esta función ya contiene toda la configuración necesaria para poder generar la nueva visualización. Para insertarla en la Vista añadirá un div al HTML en el que estará contenida.

9.5 Menú contextual

Se va a implementar una nueva funcionalidad con intención de facilitar el acceso a los datos mediante un menú contextual en el momento en que el usuario haga clic derecho sobre alguna neurona.

Como el mapa se ha creado con JS no podemos capturar la acción sobre él con Vaadin[28]. Por tanto se creará un menú contextual, como el que se muestra en Figure 9.23, con JS y después se realizará una llamada a Vaadin para ejecute la acción asociada. Existen diversas posibilidades de implementar esta herramienta así que, en primer lugar se buscarán distintas posibilidades y se seleccionará una de ellas para insertarla. Para simplificar este trabajo, se ha decidido desarrollar las distintas posibilidades de forma externa a la aplicación y en el momento que se decida cual de ellas se ajusta mejor a la IU se integrará a la aplicación.

Cuando se haga clic derecho sobre una neurona se marcará como la seleccionada aunque a continuación no se realice ninguna acción sobre el menú contextual. Si lo abrimos y a continuación hacemos clic izquierdo sobre cualquier elemento fuera del menú, se cierra, si hacemos clic derecho sobre otra neurona, se cierra en la primera neurona y se abre en la posición de la segunda. Pero si se hace clic derecho fuera del mapa no se cierra el menú abierto. Esto se debe a que consideramos que no se puede hacer clic derecho fuera del mapa.

Este menú permitirá realizar 4 acciones:

- **Descargar:** Cambia de pestaña a NeuronTab y descarga la neurona sobre la que se ha hecho clic derecho.
- **Etiquetar:** Cambia de pestaña a LabelTab, selecciona la neurona indicada y pone el foco en el TextField de crear una nueva etiqueta.
- **Estadisticas:** Cambia de pestaña a NeuronTab, muestra el Button de volver y carga la visualización de las distancias.
- **Observar datos**: Cambia de pestaña a NeuronTab, muestra el Button de volver y carga la visualización de la distribución de los datos.

9.5.1 Implementación

El menú contextual seleccionado para introducir en la aplicación es código HTML y CSS. Se ha tomado la decisión de crear una función de JS que lo generará. Se añaden las referencias al código CSS necesario y se importan las librerías JS que se van a utilizar.

El código HTML del menú siempre esta cargado en la aplicación y se oculta o muestra mediante CSS. Este se modifica con JS cada vez que se detecta un clic izquierdo o derecho.

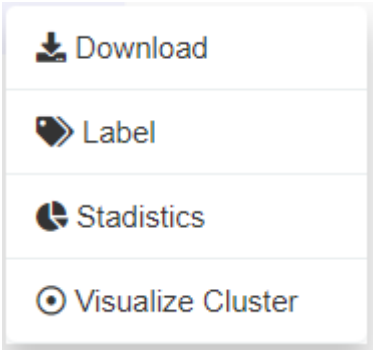


Figura 9.23: Menú contextual.

Si se detecta un clic sobre alguno de los tres elementos del menú, este se oculta y llama a la función que tiene asociada en Vaadin.

Para que el menú se oculte correctamente es necesario controlar el clic izquierdo en 2 lugares: sobre alguna neurona y sobre el resto de la interfaz. También es necesario controlar que el clic izquierdo por defecto del navegador en dos lugares distintos para que no se llegue a mostrar en ningún momento. Esto se hace, al igual que el clic izquierdo, tanto en el clic derecho sobre las neuronas y sobre el resto del documento.

Es necesario saber que visualización se está mostrando en cada momento debido a que la función que controla el clic izquierdo sobre toda la interfaz si se activa al hacer clic sobre una de las dos nuevas visualizaciones informará de un error de ejecución, para solucionar este problema, cada vez que se muestre alguna de estas visualizaciones se impedirá que el menú se active hasta que se vuelva a mostrar el mapa.

Capítulo 10

Pruebas

JUnit[55] es un framework creado para la automatización de pruebas unitarias en Java. Permite la ejecución de clases de manera controlada para el evaluar si el comportamiento de las funciones es el esperado. Estas pruebas son de caja negra[56] ya que en función de un valor de entrada se comprueba si el valor de salida es correcto, pero no si el método de obtención del valor lo es.

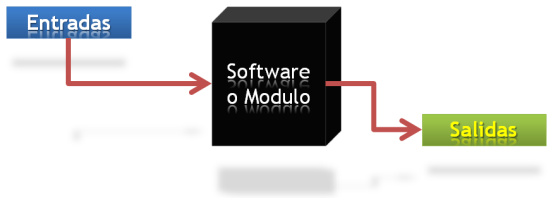


Figura 10.1: Esquema de una caja negra

Comprobar el correcto funcionamiento de un sistema software es una tarea muy costosa en tiempo y recursos. Con la intención de reducirlos lo máximo posible se han automatizado un conjunto de pruebas unitarias para asegurarnos que el software añadido a la aplicación de visualización funciona correctamente. Se comprueba la comunicación entre el cliente y servidor además del correcto funcionamiento de este último. Se han realizado pruebas manuales sobre la Vista, y no se considera necesario automatizarlas ya que se está utilizando un framework para crear la visualización y asumimos que ya se han realizado todas las pruebas pertinentes sobre cada uno de los elementos.

En los apartados siguientes podres encontrar las comprobaciones desglosadas en distintos grupos, cada vez que se ejecuten las pruebas, cada uno de ellos creará, modificará y eliminará todos los elementos necesarios de modo que al finalizar las pruebas el estado de los datos almacenados en el sistema será el mismo que al inicio.

Se han creado un conjunto de tablas que aportan una visualización más intuitiva de las pruebas que se crearán. En cada una de ellas se mostrarán un conjunto de pruebas realizadas. En cada una de ellas solo se muestran los datos que se consideran de interés para las pruebas.

En todas las tablas podremos encontrar elementos de entrada una columna que indica si esas operaciones producen un error, elementos de salida y una descripción de una o varias

pruebas, a demás puede haber una columna adicional que contiene información de un valor ya almacenado, esta columna se añade cuando queremos indicar que se va a modificar el contenido actualmente almacenado. Puede darse el caso de que en algunas celdas no tengamos ningún valor a introducir, en ese caso tendrán en su interior un "-", si la celda esta en blanco significa que el valor es la cadena vacía(""). En caso de que se de un error no podrá haber un valor de salida ya que no se ha podido realizar correctamente la operación.

10.1 Pruebas de inserción de etiquetas

En esta sección describiremos brevemente las pruebas creada sobre el apartado de inserción de nuevas etiquetas, se debe tener en cuenta que para realizarlas correctamente, es necesario utilizar funcionalidades que no son propias de esta tabla, como puede ser eliminar los elementos creados o obtener una etiqueta creada.

Las pruebas de inserción se han dividido en dos tablas: Table 10.1, las pruebas realizadas con los nombres de las etiquetas y Table 10.2, que contiene que describe las distintas formas de la que se han tratado los colores al insertar una nueva etiqueta.

Table 10.1 está dividida en 5 apartados: "Palabras Reservadas" en las que se intenta insertar nuevas etiquetas con los nombres que están reservados para el sistema; "Normal" que contiene una única prueba donde se inserta correctamente un nuevo valor; "Espacios" como ya se ha mencionado anteriormente las etiquetas con espacios se tratan de una forma especial, en estas filas se podrá comprobar el correcto funcionamiento de las distintas posibilidades de utilizar un espacio; "Barra Baja" al igual que con los espacios, las barras bajas también se tratan de una manera peculiar y por tanto se realizan las comprobaciones necesarias; "Repeticiones" corrobora que no puede haber etiquetas repetidas y que los espacios en medio de las etiquetas se convierten en barras bajas.

	Nombre de entrada	Error	Nombre de Salida	Descripción
Palabras Reservadas		Si	-	No se pueden crear etiquetas con los nombres de las palabras reservadas para el sistema.
	Empty	Si	-	
	UNKNOWN	Si	-	

Normal	InName	No	InName	La etiqueta se inserta sin realizar modificaciones sobre ella.
Espacios	InName1	No	InName1	Tiene un espacio al final del texto que se elimina cuando se inserta.
	InName2	No	InName2	Tiene un espacio al principio del texto que se elimina cuando se inserta.
	In Name3	No	Prue_ba3	-
Barra Baja	InName4_	No	InName4	Se eliminan las barras bajas al final del texto.
	_InName5	No	InName5	Se eliminan las barras bajas al principio del texto.
	In_Name6	No	In_Name6	-
Repeticiones	InName7	No	InName7	-
	InName7	Si	-	No se pueden insertar etiquetas repetidas.
	In Name8	No	In_Name8	-
	In Name8	Si	-	Debido a que los espacios en el centro se convierten en barras bajas la etiqueta que se intenta crear ya existe.

Cuadro 10.1: Pruebas realizadas sobre los nombres al añadir nuevas etiquetas.

	Color de entrada	**Error**	**Color de Salida**	**Descripción**
Dentro del Rango	0,0,0	No	0,0,0	-
	255,255,255	No	255,255,255	-
Fuera de rango Inferior	-1,0,0	Si	-	El rango de valores RGB se encuentra entre $[0-255]$ para cada uno de los colores
	0,-1,0	Si	-	
	0,0,-1	Si	-	
Fuera de rango Superior	256,255,255	Si	-	El rango de valores RGB se encuentra entre $[0-255]$ para cada uno de los colores
	255,256,255	Si	-	
	255,255,256	Si	-	

Cuadro 10.2: Pruebas realizadas sobre los colores al añadir nuevas etiquetas.

Table 10.2 contiene 3 secciones: ”Dentro del rango” comprueban el limite superior e inferior de cada uno de los colores; ”Fuera del rango inferior” para cada color comprueba el limite inferior fuera de rango; ”Fuera del rango superior” al igual que la sección anterior para cada color comprueba el limite superior fuera del rango.

10.2 Pruebas de modificación de etiquetas

Como ya hemos dicho en ese apartado se añade la columna ”Nombre a Modificar” el nombre al cual queremos que se modifique el valor del ”Nombre Original” que contiene el nombre de la etiqueta que se quiere cambiar.

Table 10.3 encontrar 6 secciones: ”Palabras Reservadas” en las que se intenta modificar el nombre de las palabras reservadas a otra etiqueta y una etiqueta normal a una palabra reservada;”Normal” contiene una única prueba donde se modifica correctamente sin necesidad de modificar;”Espacios” Partiendo de etiquetas originales se comprueba que al introducir un espacio al principio, en el medio y al final en el nombre que queremos que substituya a la etiqueta original su funcionamiento es adecuado;”Barra Baja” dado que las barras bajas también se tratan de forma peculiar se realizan las mismas pruebas que en ”Espacios” pero en este caso con barras bajas en su lugar;”Repeticiones” Se intenta modificar el nombre de una etiqueta por otro que ya existe, se prueban sus variantes con barras bajas;”Inexistente” intenta modificar una etiqueta inexistente; ”Asignada” Se modifica una etiqueta que tiene neuronas asignadas.

	Nombre Original	Nuevo Nombre	Error	Nombre de Salida	Descripción
Palabras Reservadas	UpName		Si	-	No se puede cambiar el valor de una etiqueta a una de las palabras reservadas para el sistema.
	UpName	Empty	Si	-	
	UpName	UNKNOWN	Si	-	
		Modificada	Si	-	No se puede modificar las etiquetas reservadas para el sistema.
	Empty	Modificada	Si	-	
	UNKNOWN	Modificada	Si	-	
Normal	UpName	Modificada	No	Modificada	La etiqueta se modifica sin realizar cambios sobre ella.
	UpName1	Modificada	No	UpName1	Tiene un espacio al final del texto que se elimina cuando se realiza la modificación.

Espacios	UpName1	Modificada	No	UpName1	Tiene un espacio al principio del texto que se elimina cuando se realiza la modificación.
	UpName2	Modi ficada	No	Modi_ficada	-
Barra Baja	UpName3	Modificada_	No	Modificada	Se eliminan las barras bajas al final del texto.
	UpName3	_Modificada	No	Modificada	Se eliminan las barras bajas al principio del texto.
	UpName4	Modi_ficada	No	Modi_ficada	-
Repeticiones	UpName5	UpName5	Si	-	-
	UpName5	_UpName5_	Si	-	No se pueden insertar etiquetas repetidas.
	Up_Name6	Up Name6	Si	-	Debido a que los espacios en el centro se convierten en barras bajas la etiqueta que se intenta crear ya existe.
Inexistente	-	UpName7	Si	-	Se intenta modificar una etiqueta que no existe.
Asignada	NameLabel	UpName8	No	UpName8	Se modifica el nombre de una etiqueta que tiene neuronas asignadas.

Cuadro 10.3: Pruebas realizadas al modificar los nombres de las etiquetas.

En Table 10.4 y Table 10.5 encontraremos las prueba que corroboran el correcto funcionamiento de la funcionalidad de cambiar colores a etiquetas ya existentes. En ambas tablas se introducen las columnas ”Color Original”, que contiene los colores originales y ”Nuevo Color”, donde se encuentra el color por el que se va a substituir al original.

En la primera de estas tablas(Table 10.4) encontraremos comprobaciones de los rangos, tanto internos como externos de para cada color, además de la modificación de una etiqueta que tiene asignada una neurona.

En Table 10.5 se intenta modificar el color asignado a las palabras reservadas, como es necesario saber que color deseamos asignar a cada palabra se introduce la columna ”Nombre Original” en la que podremos encontrar los nombres reservados para el sistema. Se ha tomado

	Color Original	Nuevo Color	Error	Color de Salida	Descripción
Dentro del Rango	94,144,58	0,0,0	No	0,0,0	-
	230,148,169	255,255,255	No	255,255,255	-
Fuera de rango Inferior	212,141,1	-1,0,0	Si	-	El rango de valores RGB se encuentra entre $[0-255]$ para cada uno de los colores
	87,27,128	0,-1,0	Si	-	
	221,203,251	0,0,-1	Si	-	
Fuera de rango Superior	164,247,51	256,255,255	Si	-	El rango de valores RGB se encuentra entre $[0-255]$ para cada uno de los colores
	190,16,251	255,256,255	Si	-	
	106,71,139	255,255,256	Si	-	
Asignada	180,200,29	67,248,188	No	127, 127, 127	Se modifica el color de una etiqueta que tiene neuronas asignadas.

Cuadro 10.4: Pruebas realizadas al modificar los colores de las etiquetas.

	Nombre Original	Color Original	Nuevo Color	Error	Descripción
Palabras reservadas		0,0,0	241,5,163	Si	No se puede cambiar el nombre de una etiqueta a las palabras reservadas para el sistema.
	Empty	0,0,0	74,172,236	Si	
	UNKNOWN	204,204,204	241,220,137	Si	

Cuadro 10.5: Pruebas realizadas al cambiar los colores de las palabras reservadas.

la decisión de eliminar la columna ”Color de Salida” ya que ninguna de estas modificaciones podrá llevarse a cabo.

10.3 Pruebas de eliminación de etiquetas

El borrado de etiquetas es una funcionalidad crítica, en caso de eliminar una etiqueta que no esta permitido provocará un erres tanto en el cliente como en el servidor, la vista no se mostrará adecuadamente y el servidor fallará al intentar recuperar valores que no existen. Logísticamente también sería un problema ya que podríamos tener neuronas con etiquetas que no existen.

En Table 10.6 podremos encontrar las pruebas necesarias para comprobar que la implementación es correcta. En primer lugar se intentan eliminar las palabras clave, como son

	Nombre Original	Error	Eliminado	Descripción
Palabras reservadas		Si	-	No se pueden eliminar las etiquetas reservadas para el sistema.
	Empty	Si	-	
	UNKNOWN	Si	-	
Inexistente	-	No	Si	Se intenta eliminar una etiqueta que no existe.
Asignada	NameLabel	No	No	Se intenta elimina una etiqueta que tiene neuronas asignadas.
Normal	DelLabel	No	Si	La etiqueta se elimina correctamente.

Cuadro 10.6: Pruebas realizadas sobre la función de eliminado de etiquetas.

esenciales para el funcionamiento de la aplicación, al igual que en los apartados anteriores, se producirá un error que impedirá su eliminación. En caso de intentar eliminar una etiqueta que no existe, el sistema indicará al usuario que se ha eliminado correctamente, de este modo se le informará que ese nombre puede ser utilizado. Puede darse el caso de que se intente borrar una etiqueta que esta asignada a una o varias neuronas, en este caso el sistema no fallará, indicará al usuario que la acción no se ha podido llevar a cabo. Por último se comprueba que el correcto funcionamiento de eliminar una etiqueta que no está reservada y que no esta asignada.

10.4 Pruebas restantes

Para esta sección se ha creado una tabla que describe las pruebas restantes. Dado que los datos de entrada pueden variar en función de la comprobación que deseemos hacer se ha introducido la columna ”Valor Introducido” que representa cualquier posible valor que vaya a insertar. También podremos encontrar dos columnas, que se comunes para todas las tablas, ”Error” y ”Descripción”.

Las pruebas realizadas en Table 10.7 se dividen en 3 secciones: ”Obtener Etiqueta” En la que se comprueba que la recuperación de la etiqueta de una neurona se realiza correctamente, también se comprueban los casos en los que se intentan obtener una etiqueta de una neurona

	Valor Introducido	Error	Descripción
Obtener Etiqueta	1000	Si	Se intenta recuperar una fuera del rango superior
	-1	Si	Se intenta recuperar una fuera del rango inferior
	5	No	-
Asignar	NoAsingar	Si	Se intenta asignar una etiqueta que no existe a una neurona
	Asignar	No	Se asigna una etiqueta a una neurona
Distancia Neurona	1000	Si	Se intenta obtener la distancia de los datos al centroide de una neurona fuera del rango superior
	-1	Si	Se intenta obtener la distancia de los datos al centroide de una neurona fuera del rango inferior
	3	No	Se obtiene la distancia de una neurona

Cuadro 10.7: Conjunto de pruebas restantes.

que no existe, se hace una comprobación de rango superior e inferior ya que las neuronas están enumeradas desde 0 hasta, como máximo, 999 ; ”Asignar” Se comprueba el funcionamiento de la asignación de una neurona a una etiqueta; ”Distancia Neurona” Fue necesario desarrollar la funcionalidad de obtener la distancia de cada uno de los datos al centroide, y por tanto es necesario realizar las comprobaciones pertinentes, se comprueba tanto la obtención de la distancia de una neurona y de neuronas que si existen. Al igual que en el caso de ”Obtener Etiqueta” se hace una comprobación de rango superior e inferior.

Capítulo 11

Conclusiones

El principal objetivo de este proyecto era dotar a una herramienta de visualización de mapas auto-organizados de la capacidad de etiquetado de los grupos creados. Esta nueva funcionalidad permitirá asociar etiquetas a los distintos conjuntos de datos agrupados por la herramienta. Dado que se ha modificado con éxito la aplicación para permitir dicha funcionalidad puede decirse que el objetivo principal de este proyecto ha sido alcanzado.

Para cumplir con la necesidad de obtener los conjuntos de datos con las etiquetas asignadas manualmente a las neuronas de la SOM, se ha modificado el proceso de generación de los datos con la información asociada de las etiquetas.

Con la intención de aumentar la información que se le aporta al usuario acerca del mapa, se ha planteado el desarrollo de una nueva visualización que muestre de datos estadísticos de los grupos generados. Para ello se ha incorporado una funcionalidad que permite acceder a estos mostrando las propiedades del conjunto de datos agrupados en la neurona.

Un objetivo relacionado con el anterior, es el de visualizar la dispersión de los datos a los que representa una neurona. Para alcanzarlo ha sido necesario realizar un precálculo que posibilite la creación de un grafo que representa la distribución de los datos de la neurona en tiempo real. Este desarrollo ha permitido alcanzar los resultados esperados.

Con la intención de proporcionar al usuario más versatilidad para realizar las acciones implementadas en los objetivos anteriores se ha desarrollado un menú contextual que se muestra al hacer clic derecho sobre una neurona y permite: obtener el conjunto de datos que contiene junto con la etiqueta asociada, etiquetar la neurona seleccionada, visualizar las estadísticas y la dispersión de los datos. Por tanto se ha alcanzado el resultado deseado sobre la mejora de la usabilidad.

Con la finalidad de generar un conjunto de datos en el formato adecuado de entrada de la SOM, se han aplicado técnicas de análisis y filtrado para transformar el conjunto inicial. Los resultados de este preprocesado han sido satisfactorios y se ha obtenido un conjunto de datos para su visualización, por lo que consideramos que hemos cumplido este objetivo.

La asociación de datos mediante patrones es una técnica compleja, pero gracias ha haber cumplido el objetivo de modificar el Etiquetado Externo hemos conseguido generar conjuntos de datos agrupados por una SOM que los une mediante patrones desconocidos. Gracias a las herramientas creadas de visualización de estadísticas y dispersión de los datos, podemos analizarlos y realizar un etiquetado manual. Se permite la extracción de los conjuntos etiquetados que contarán con las características idóneas para ser procesados con técnicas de IA supervisada.

Con todo esto, se cumplen los requisitos de la aplicación en cuanto al etiquetado y visualización de datos. Permitiendo la obtención del contenido de las neuronas con la etiqueta asociada. Por ello, tras desarrollar este proyecto y redactar esta memoria, dado que se han alcanzado los objetivos propuestos puede decirse que los resultados han sido satisfactorios para la realización de este proyecto.

Capítulo 12

Trabajo futuro

Durante el avance de este proyecto se han detectado posibles líneas de trabajo futuras que se encuentran fuera del alcance del este proyecto. Estos trabajos se salen de los objetivos definidos pero no obstante son relevantes y merecen ser tenidas en cuenta para su aplicación futura.

- Al trabajar con una cantidad de datos inmensa, la generación de ficheros del análisis tienen una complejidad temporal alta, sería interesante realizar una paralelización de este código con la intención de reducir los tiempos de carga de manera significativa.

- El sistema de etiquetado ha demostrado ser de gran utilidad para los usuarios de la aplicación, por tanto se han planteado posibles mejoras que podrían simplificarlo. Se plantea el etiquetado multi-neurona. Esto permitiría que el usuario seleccione varias neuronas al mismo tiempo y les pueda asignar una etiqueta.

- Uno de las funcionalidades mas interesantes de nuestra aplicación es que utiliza como base una estructura cliente-servidor, esto permite que pueda existir diversos etiquetados externos creados por distintos usuarios. Podría aprovecharse esta funcionalidad permitiendo realizar cruces entre etiquetados externos y generar uno nuevo basándose en el número de coincidencias.

- La herramienta desarrollada permite exportar conjuntos de datos etiquetados, sobre los que se podrían aplicar técnicas de IA supervisada. Además podría realizarse un estudio de estas técnicas y comparar cual de ellas ofrece mejores resultados.

- El mapa muestra una cantidad de información limitada, por este motivo se podría substituir esta visualización por un nuevo diseño similar al de la dispersión, en el cual se pueda insertar más información acerca de la red.

- Conocer la dispersión de los datos asignados a una neurona resulta interesante, a raíz de esto se ha planteado la posibilidad de agrupar los datos por las etiquetas que tiene cada uno de ellos originalmente.
- Como linea futura, consideramos ampliar la herramienta de visualización de la dispersión de los datos de una neurona, teniendo varios precálculos realizados con distintos rango de agrupación y que cada uno de ellos muestre información más detallada de como se distribuyen.

Apéndice A

Diagramas UML de la aplicación

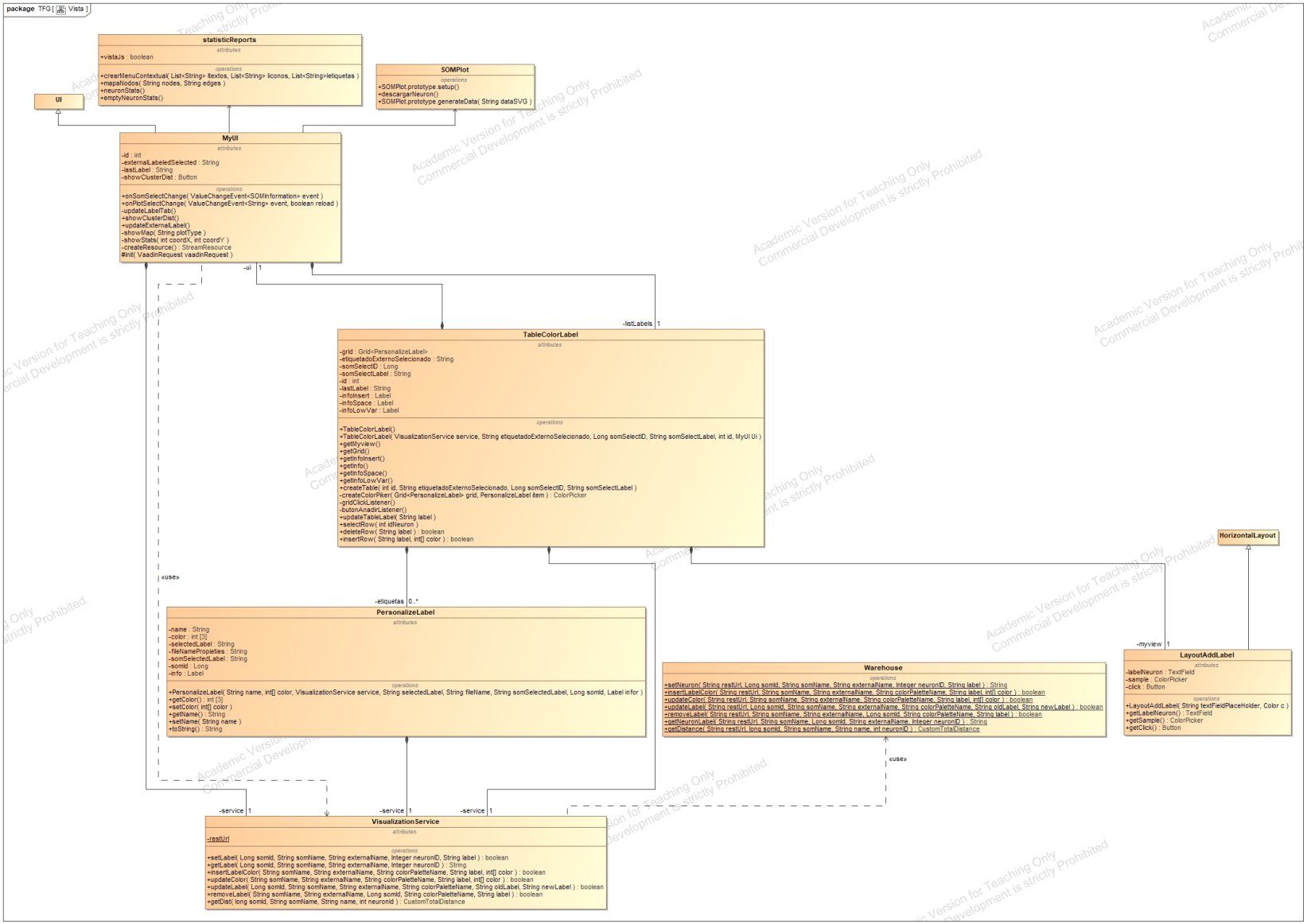


Figura A.1: UML completo de la Vista.

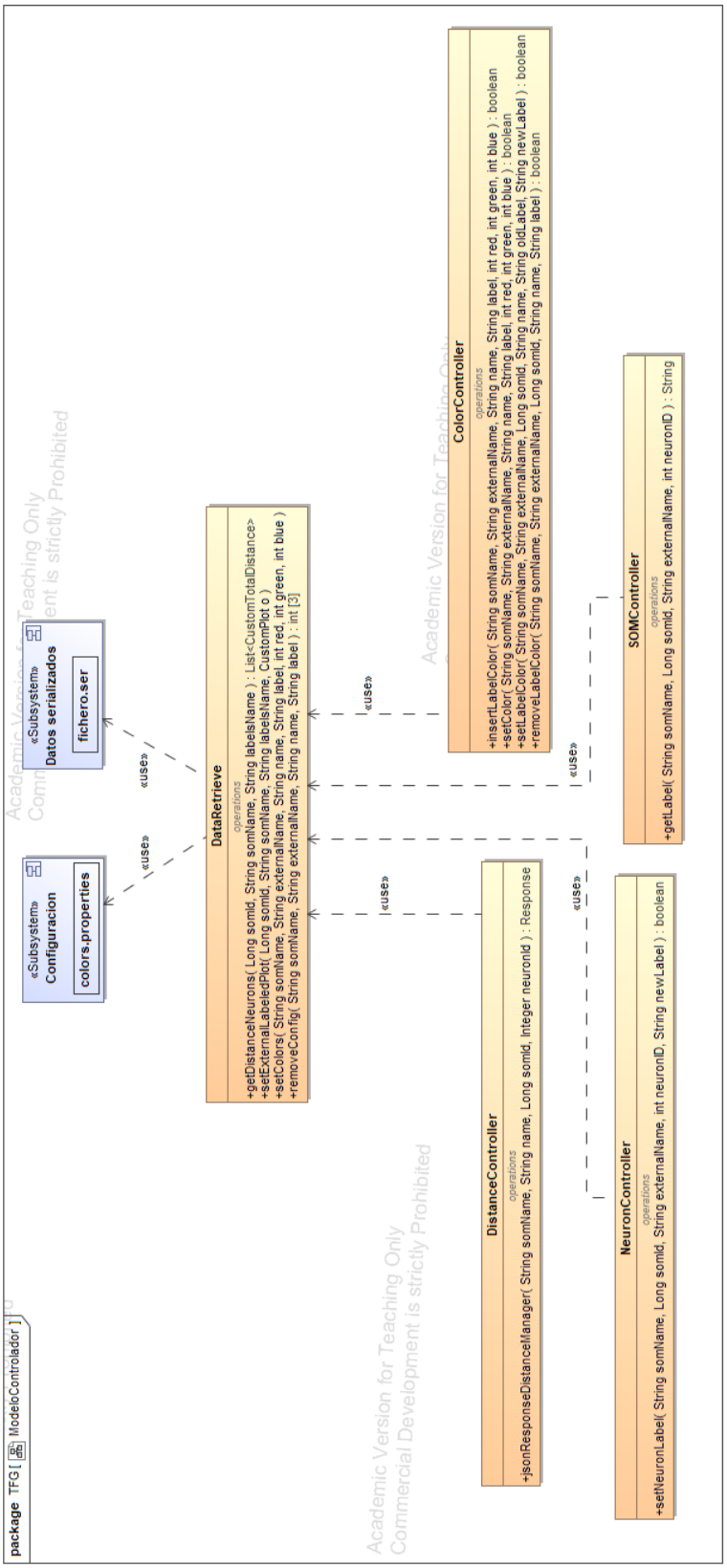


Figura A.2: UML completo de Modelo-Controlador.

Apéndice B

Planificación

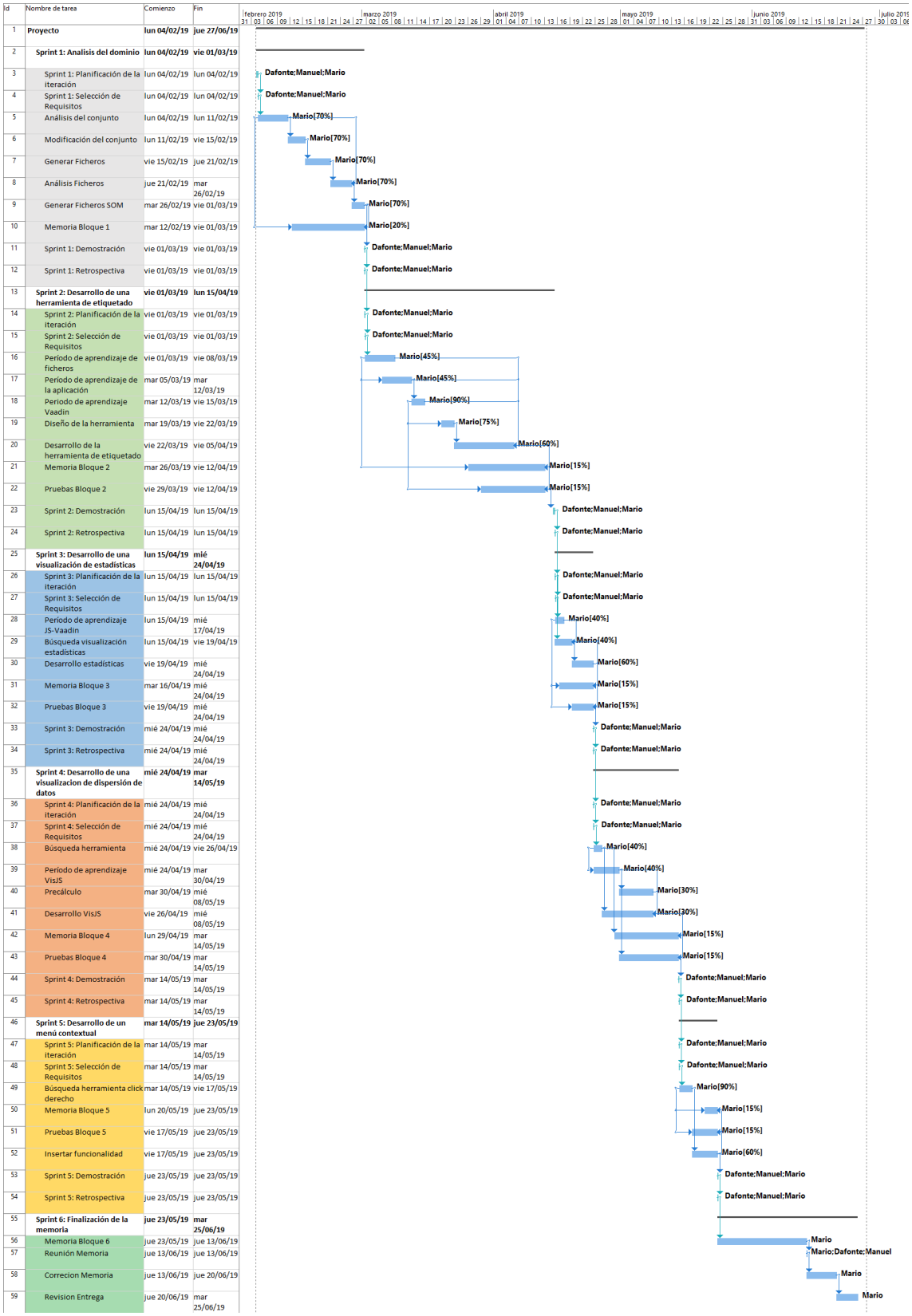


Figura B.1: Diagrama de Gantt de los Sprints del proyecto.

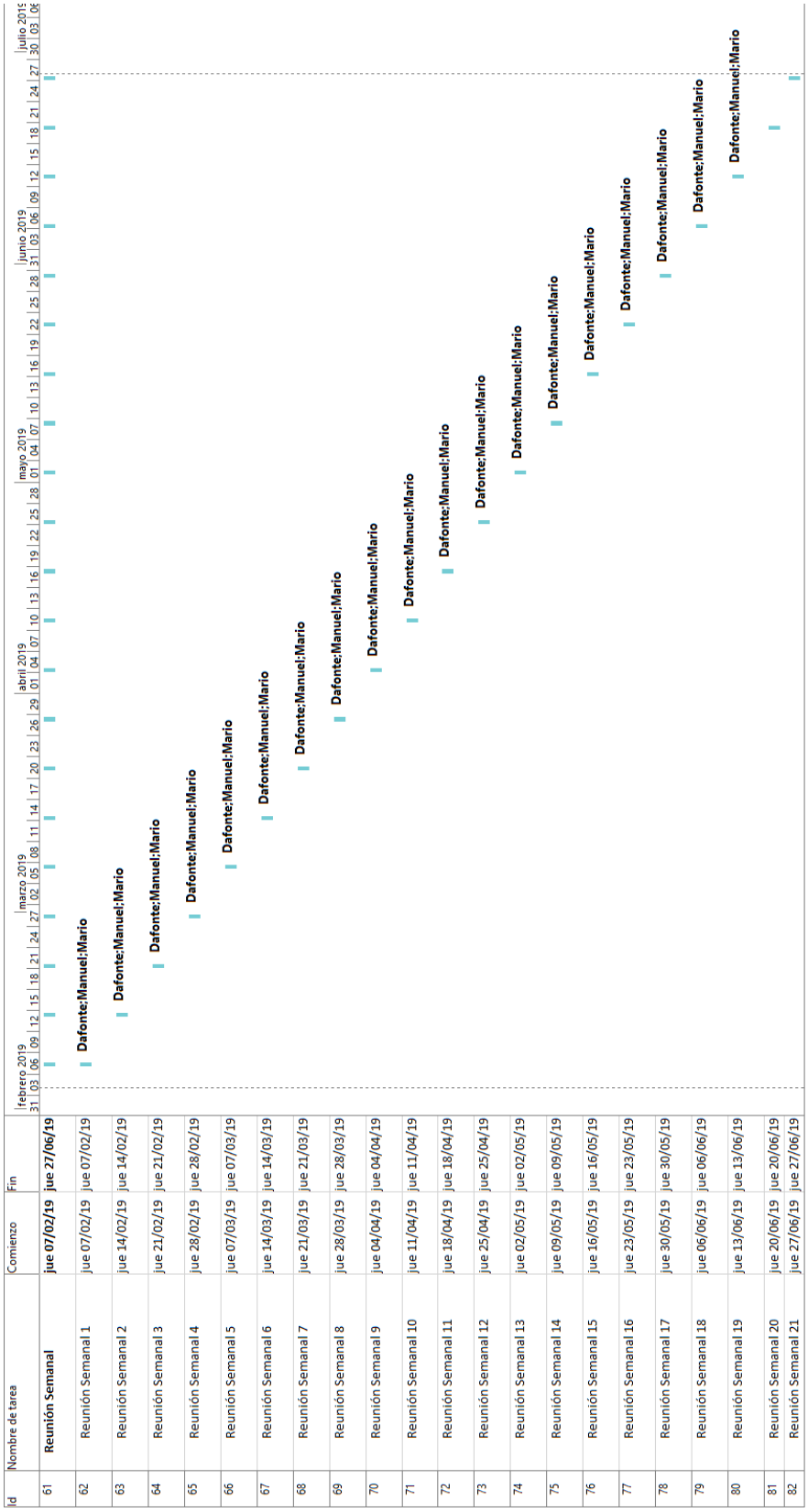


Figura B.2: Diagrama de Gantt de las reuniones.

Apéndice C

Manual de Usuario

El conjunto de herramientas que hemos creado permiten realizar distintas visualizaciones sobre los datos que ha procesado una SOM, así como modificaciones sobre los resultados de la misma. Este sistema esta orientado a trabajar con *Big Data*, por eso mismo hemos seleccionado un conjunto de datos, los hemos analizado y procesado con el fin de comprobar que el funcionamiento de la aplicación es correcto.

C.1 Ejecución

Para llevar a cabo este proyecto ha sido necesario separarlo en dos grandes apartados:

- Análisis de datos donde a partir del conjunto de datos *ISCX 2017*, creado por Canadian Institute for Cybersecurity[5], obtenemos información sobre los distintos ataques de red.
- Desarrollo de un grupo de herramientas de visualización que se insertarán en una aplicación web.

Para poder utilizar y generar todos los documentos utilizados para el análisis es necesario:

- Python 3.7 o superior.
- El conjunto de datos *ISCX 2017* o una versión similar.

Para poder ejecutar el apartado análisis debemos seguir los siguientes pasos:

1. Lanzar el fichero «modifieData.py », que generará un conjunto de ficheros en el directorio «CSVsGenerados/».
2. A continuación se ejecuta «analizeData.py », que generará un conjunto de ficheros en el directorio «Datos » y un conjunto de imágenes en «Histogramas. »

3. Para finalizar ejecutaremos «generateInputFile.py» que creará en «ficheros SOM» todos los documentos necesarios para introducir en la SOM.

Para poder utilizar la aplicación que contiene todo el conjunto de herramientas que hemos desarrollado es necesario:

- Java 1.8 o superior.
- Eclipse EE con el plugin de Maven y de Jetty.
- Un mapa generado y configurado con todos los ficheros de visualización disponibles.

Para poder ejecutar la aplicación debemos:

- Compilar el modelo haciendo click derecho sobre «SOMUP_Model» →«Run as» →«Maven install».
- A continuación, click derecho sobre «SOMUP_Rest» →«Run configuration» →en la pestaña main completamos «Base directory» seleccionan el workspace «SOMUP_Rest» y en goals escribimos «spring-boot:run», en la pestaña JRE introducimos «-Dserver.port=8090» y pulsamos «Run».
- Hacemos click derecho sobre «SOMUP_WebVis» →«Run configuration» →en la pestaña «main » completamos «Base directory » seleccionando el workspace «SOMUP_WebVis» y en goals escribimos «jetty:run». En el momento en el que el servidor haya terminado de cargar pulsamos «Run».
- Por último, abrimos un navegador y escribimos «http://localhost:8080/ ». Al acceder a esta ruta cargará una interfaz web durante unos minutos.

Una vez seguidos estos pasos, se podrá hacer uso de la aplicación. En el futuro se contará con un servidor de aplicaciones que aloje esta herramienta y permita trabajar de forma remota.

Apéndice D

Glosario de acrónimos

SOM *Self-Organizing Maps.*

MVC *Model Vista Controlador.*

JS *JavaScript.*

CSS *Cascading Style Sheets.*

IU *Intefaz de Usuario.*

HTML *HyperText Markup Language.*

IA *Inteligencia Artificial.*

EE *Etiquetado Externo.*

Bibliografía

[1] Statista, "IHS. "Internet of Things (Iot) Connected Devices Installed Base Worldwide from 2015 to 2025 (in Billions)."," 2018. [Online]. Available: https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/{%}22

[2] H. Lin, Z. Yan, Y. Chen, and L. Zhang, "A Survey on Network Security-Related Data Collection Technologies," *IEEE Access*, 2018.

[3] D. Zhou, Z. Yan, Y. Fu, and Z. Yao, "A survey on network data collection," *Journal of Network and Computer Applications*, vol. 116, pp. 9–23, aug 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1084804518301607

[4] A. L. Buczak and E. Guven, "A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1153–1176, 2016. [Online]. Available: http://ieeexplore.ieee.org/document/7307098/

[5] "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization," *Proceedings of the 4th International Conference on Information Systems Security and Privacy*, no. January, pp. 108–116, 2018. [Online]. Available: http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0006639801080116

[6] "The unique visual approach to predictive modeling." [Online]. Available: https://www.viscovery.net/

[7] "Matlab." [Online]. Available: https://www.mathworks.com/products/matlab.html

[8] J. P. E. Alhoniemi, J. Himberg and J. Vesanto. (2005) Som toolbox. [Online]. Available: http://www.cis.hut.fi/projects/somtoolbox/about.shtml

[9] "C-extensions for python." [Online]. Available: https://cython.org/

[10] V. Moosavi and S. Packmann. (2015) SOMPY: A self organizing map library in python. [Online]. Available: https://github.com/sevamoo/SOMPY

[11] Bingell, "Willkommen am fachbereich mathematik und informatik," Feb 2017. [Online]. Available: https://www.uni-marburg.de/fb12

[12] "Databionic esom tools - databionic esom tools." [Online]. Available: http://databionic-esom.sourceforge.net/

[13] [Online]. Available: https://git-scm.com/

[14] "Build software better, together." [Online]. Available: https://github.com/

[15] "Official home page." [Online]. Available: https://www.microsoft.com/es-es

[16] "Microsoft project." [Online]. Available: https://products.office.com/es-es/project/project-and-portfolio-management-software

[17] M. Inc, "Magicdraw." [Online]. Available: https://www.nomagic.com/products/magicdraw

[18] "Python data analysis library¶." [Online]. Available: https://pandas.pydata.org/

[19] M. Bostock, "Data-driven documents." [Online]. Available: https://d3js.org/

[20] "vis.js." [Online]. Available: https://visjs.org/

[21] "Metodologías ágiles y clásicas en la gestión de un proyecto," Oct 2018. [Online]. Available: http://www.itmplatform.com/es/blog/metodologias-agiles-y-clasicas-en-la-gestion-de-un-proyecto/

[22] "Qué es scrum," Oct 2018. [Online]. Available: https://proyectosagiles.org/que-es-scrum/

[23] "Release burndown." [Online]. Available: https://web.archive.org/web/20071101053001/http://www.mountaingoatsoftware.com/release_burndown

[24] "Qué son las historias de usuario y su función en agilidad," Dec 2017. [Online]. Available: https://solvingadhoc.com/las-historias-usuario-funcion-agilidad/

[25] Atlassian, "Free git gui for mac and windows." [Online]. Available: https://www.sourcetreeapp.com/

[26] J. Ordóñez, "¿qué es una api rest?" Sep 2018. [Online]. Available: https://www.idento.es/blog/desarrollo-web/que-es-una-api-rest/

[27] FortAwesome. (2019, Jun) Font awesome 5. [Online]. Available: https://fontawesome.com/

[28] Vaddin: The platform for building web apps with java and web components. [Online]. Available: https://vaadin.com/

[29] "Font awesome 5." [Online]. Available: https://fontawesome.com/

[30] "Encyclopedia of cryptography and security." [Online]. Available: https://books.google.es/books?id=UGyUUK9LUhUC&lpg=PR1&ots=4UcN489Zfz&dq=EncyclopediaofCryptographyandSecurity&lr&hl=es&pg=PA314#v=onepage&q&f=false

[31] S. User, "Control interno control interno biblioteca de controles biblioteca de riesgos autoevaluación control interno buenas prácticas guías y modelos coso indicadores informes y recomendaciones cuestionarios y evaluaciones matrices segregación de funciones políticas riesgos, matrices y alertas Últimos artículos 11 abril 2019 buenas prácticas para proteger a la organización de una amenaza cibernética 11 abril 2019 buenas prácticas para diseñar un programa de ciberseguridad 11 abril 2019 checklist para evaluar el riesgo de fraude por parte de la junta directiva y la alta gerencia 28 marzo 2019 checklist diagnóstico general del sistema de control interno para el comité de auditoría - actividades de control," Mar 2019. [Online]. Available: https://web.archive.org/web/20190418144529/https://www.auditool.org/blog/auditoria-de-ti/3423-que-es-un-ciberataque

[32] "Search unb." [Online]. Available: https://www.unb.ca/cic/

[33] A. Guedez, "Conoce los ataques de fuerza bruta y las mejores formas de evitarlos," Oct 2018. [Online]. Available: https://www.gb-advisors.com/es/conoce-los-ataques-de-fuerza-bruta/

[34] Lanjelot, "lanjelot/patator," Aug 2018. [Online]. Available: https://github.com/lanjelot/patator

[35] [Online]. Available: http://slacksite.com/other/ftp.html#passive

[36] "Ssh." [Online]. Available: https://www.ssh.com/ssh/

[37] "Slowloris http dos." [Online]. Available: https://web.archive.org/web/20150426090206/http://ha.ckers.org/slowloris

[38] Shekyan, "shekyan/slowhttptest," Aug 2017. [Online]. Available: https://github.com/shekyan/slowHttpTest

[39] V. Brujeador, "Hulk, una herramienta dos para servidores web." [Online]. Available: https://www.hackplayers.com/2012/05/hulk-una-herramienta-dos-para.html

[40] Jseidl, "jseidl/goldeneye," Jun 2018. [Online]. Available: https://github.com/jseidl/GoldenEye

[41] "Goldeneye denial of service ddos attack using kali linux." [Online]. Available: https://www.thesecurityblogger.com/goldeneye-denial-of-service-ddos-attack-using-kali-linux/

[42] D. Goodin and Utc, "Critical crypto bug in openssl opens two-thirds of the web to eavesdropping," Apr 2014. [Online]. Available: https://arstechnica.com/information-technology/2014/04/critical-crypto-bug-in-openssl-opens-two-thirds-of-the-web-to-eavesdropping/

[43] I. OpenSSL Foundation, "Openssl." [Online]. Available: https://www.openssl.org/

[44] "The cross-site scripting (xss) faq." [Online]. Available: https://www.cgisecurity.com/xss-faq.html

[45] "What is sql injection?" [Online]. Available: https://www.cgisecurity.com/questions/sql.shtml

[46] "I'd rather play golf." [Online]. Available: https://web.archive.org/web/20110628130925/http://iablog.sybase.com/paulley/2008/07/sql2008-now-an-approved-iso-international-standard/

[47] "Penetration testing software, pen testing security." [Online]. Available: https://www.metasploit.com/

[48] Sweetsoftware, "sweetsoftware/ares," Dec 2017. [Online]. Available: https://github.com/sweetsoftware/Ares

[49] "What is a botnet? - definition from techopedia." [Online]. Available: https://www.techopedia.com/definition/384/botnet

[50] [Online]. Available: https://nmap.org/man/es/man-port-scanning-techniques.html

[51] T. Kohonen, "Self-organized formation of topologically correct feature maps," *Biological Cybernetics*, vol. 43, no. 1, pp. 59–69, 1982. [Online]. Available: https://doi.org/10.1007/BF00337288

[52] "Serialización. ficheros de objetos en java. interface serializable." [Online]. Available: http://puntocomnoesunlenguaje.blogspot.com/2013/10/java-serializacion-persistencia.html

[53] "What is http?" [Online]. Available: https://www.w3schools.com/whatis/whatis_http.asp

[54] "Mvc (model, view, controller) explicado." [Online]. Available: https://codigofacilito.com/articulos/mvc-model-view-controller-explicado

[55] "The new major version of the programmer-friendly testing framework for java." [Online]. Available: https://junit.org/junit5/

[56] "Pruebas de caja negra y un enfoque práctico," Feb 2017. [Online]. Available: https://testingbaires.com/2017/02/26/pruebas-caja-negra-enfoque-practico/