



# **Sistema de análisis de datos y soporte a la toma de decisiones basado en Mapas Autoorganizados “VisualiSOM”**

*Informe final*

Proyecto final para optar por el grado de Ingeniero en Informática  
*Departamento de Informática*

Autores:

- [Lisandro D'Alú De Boni](#)
- [María Camila Ezama](#)
- [Augusto Maletta](#)

Director:

- [Dr. Ing. Gustavo Javier Meschino](#)

Co-Director:

- [Dr. Ing. Diego Sebastián Comas](#)

Mar del Plata, 4 de noviembre de 2024



RINFI es desarrollado por la Biblioteca de la Facultad de Ingeniería de la  
Universidad Nacional de Mar del Plata.

Tiene como objetivo recopilar, organizar, gestionar, difundir y preservar  
documentos digitales en Ingeniería, Ciencia y Tecnología de Materiales y  
Ciencias Afines.

A través del Acceso Abierto, se pretende aumentar la visibilidad y el impacto  
de los resultados de la investigación, asumiendo las políticas y cumpliendo  
con los protocolos y estándares internacionales para la interoperabilidad  
entre repositorios



Esta obra está bajo una [Licencia Creative Commons  
Atribución- NoComercial-CompartirIgual 4.0  
Internacional](https://creativecommons.org/licenses/by-nc-sa/4.0/).



# **Sistema de análisis de datos y soporte a la toma de decisiones basado en Mapas Autoorganizados “VisualiSOM”**

*Informe final*

Proyecto final para optar por el grado de Ingeniero en Informática  
*Departamento de Informática*

Autores:

- [Lisandro D'Alú De Boni](#)
- [María Camila Ezama](#)
- [Augusto Maletta](#)

Director:

- [Dr. Ing. Gustavo Javier Meschino](#)

Co-Director:

- [Dr. Ing. Diego Sebastián Comas](#)

Mar del Plata, 4 de noviembre de 2024



<b>1. Agradecimientos.....</b>	<b>3</b>
<b>2. Resumen.....</b>	<b>3</b>
<b>3. Introducción.....</b>	<b>4</b>
<b>4. Objetivos del proyecto.....</b>	<b>4</b>
4.1 Objetivo general.....	4
4.2 Objetivo secundario.....	5
4.3 Innovación e impacto del proyecto.....	5
<b>5. Planificación del proyecto.....</b>	<b>7</b>
5.1 Análisis del problema.....	7
5.2 Solución planteada.....	7
5.3 Equipo de trabajo.....	8
5.4 Análisis FODA.....	9
5.4.1 Fortalezas.....	9
5.4.2 Oportunidades.....	9
5.4.3 Debilidades.....	9
5.4.4 Amenazas.....	10
5.5 Análisis de riesgos.....	11
5.6 Estimación de tiempos.....	12
<b>6. Mapas autoorganizados como herramienta.....</b>	<b>13</b>
6.1 Definición.....	13
6.2 Surgimiento de la herramienta y su implementación.....	13
<b>7. Ejecución del proyecto.....</b>	<b>20</b>
7.1 Elicitación de requerimientos.....	20
7.1.1 Requerimientos funcionales.....	20
7.1.2 Requerimientos no funcionales.....	21
7.2 Diseño del sistema.....	22
7.2.1 Modelado de procesos.....	22
7.2.2 Arquitectura del sistema.....	25
7.3 Diseño de interfaces gráficas.....	26
7.3.1 Carga de datos.....	26
7.3.2 Visualización del mapa entrenado.....	27
7.3.3 Mapas de componentes.....	27
7.3.4 Hits/Etiquetas.....	28
7.3.5 Clustering.....	28
7.3.6 Segmentación de imágenes.....	29
7.4 Tecnologías.....	29
7.4.1 Backend.....	29
7.4.2 Frontend.....	30
<b>8. Módulos.....</b>	<b>31</b>
8.1 Carga de datos y parametrización.....	31



---

8.2 Entrenamiento.....	33
8.3 Visualización e interacción con grilla.....	35
8.4 Componentes.....	39
8.5 Etiquetado (hits).....	40
8.6 Agrupamiento (clustering).....	43
8.7 Nuevos datos.....	44
8.8 Segmentación y coloreado de imágenes.....	46
<b>9. Testing.....</b>	<b>54</b>
9.1 API.....	54
9.2 Aplicación Web.....	54
<b>10. Entregables.....</b>	<b>58</b>
10.1 Software.....	58
10.2 Manual de uso.....	58
<b>11. Benchmarking.....</b>	<b>59</b>
<b>12. Memoria del proyecto.....</b>	<b>62</b>
12.1 Nacimiento del proyecto.....	62
12.2 Cumplimiento de objetivos.....	62
12.3 Comparación planificación original y ejecutada.....	63
12.4 Análisis.....	63
12.5 Diseño.....	65
12.6 Desarrollo e implementación.....	66
12.7 Validación y ajustes.....	66
12.8 Pruebas de integración.....	67
12.9 Redacción del informe.....	67
12.10 Metodología de trabajo.....	69
12.11 Software auxiliar para la gestión del proyecto.....	69
<b>13. Conclusiones.....</b>	<b>73</b>
<b>14. Apéndices.....</b>	<b>76</b>
14.1 Apéndice A - Endpoints - API.....	76
14.2 Apéndice B - Glosario.....	77
<b>15. Bibliografía.....</b>	<b>79</b>
<b>16. Anexo: Manual de uso.....</b>	<b>84</b>



## 1. Agradecimientos

En primer lugar, queremos agradecer a nuestros directores de trabajo final, Gustavo y Diego, por guiarnos durante todo el proyecto y por estar siempre presentes durante el desarrollo del mismo.

Nos parece importante resaltar el apoyo de nuestros familiares y amigos, que nos acompañaron en el transcurso de toda la carrera, y su apoyo fue fundamental para no ceder ante tantas situaciones adversas que se presentaron en los últimos años.

Por último, a la Facultad de Ingeniería, especialmente a todos los docentes que aportaron en nuestra formación, tanto académica como profesional y personal. A todos ellos, gracias.

## 2. Resumen

Este proyecto fue propuesto por el Laboratorio de Bioingeniería en conjunto con el Laboratorio de Procesamiento de Imágenes, ambos pertenecientes al Instituto de Investigaciones Científicas y Tecnológicas en Electrónica (ICYTE, CONICET, UNMDP). Tiene como objetivo desarrollar una herramienta para asistir a los usuarios a tomar decisiones efectivas y eficientes, utilizando la capacidad de los Mapas Autoorganizados (SOM, de *'Self Organizing Maps'*) para visualizar y organizar datos multivariable de forma intuitiva y comprensible.

La herramienta está compuesta por una aplicación web que será el punto de ingreso de los datos y de visualización de los resultados por el usuario y un servidor que se encargue del procesamiento de los datos y luego del entrenamiento hiperparametrizado de la red neuronal.

El sistema informático propuesto utilizará datos como entrada para entrenar un SOM, que luego se visualizará en un mapa bidimensional de celdas. Este mapa



permitirá descubrir patrones y relaciones entre los datos, lo que constituirá una asistencia en la toma de decisiones basada en la exploración y análisis de los mismos.

### **3. Introducción**

Los Mapas Autoorganizados (*SOM*, de '*Self Organizing Maps*') son redes neuronales de aprendizaje no supervisado, utilizadas primordialmente en la visualización de datos con dos objetivos fundamentales: a) la compresión de la información contenida en los datos (pudiendo manejar grandes volúmenes) y b) la operabilidad con datos multivariable (no existiendo nominalmente límite en la cantidad de variables). El aprendizaje no supervisado es un tipo de enfoque en el campo del aprendizaje automático donde el algoritmo se entrena sobre un conjunto de datos que no tiene etiquetas o categorías predefinidas. Este enfoque explora la estructura inherente de los datos y busca agruparlos en grupos (*clusters*) basados en similitudes, distribuciones, densidades u otras características<sup>(1)</sup>.

### **4. Objetivos del proyecto**

#### **4.1 Objetivo general**

Desarrollar una herramienta para asistir a los usuarios a tomar decisiones de manera efectiva y eficiente, utilizando la capacidad de la técnica de aprendizaje no supervisado denominada Mapas Autoorganizados, para organizar y visualizar grandes cantidades de datos de manera que los usuarios puedan identificar patrones, tendencias y relaciones entre los datos y sus variables, de forma intuitiva y eficiente. La herramienta debe brindar al usuario una interfaz amigable y que se abstraiga de los detalles de la implementación algorítmica, centrándose en la interactividad.



## 4.2 Objetivo secundario

A partir del objetivo general, hemos planteado un objetivo secundario: dar visibilidad a los SOM debido al crecimiento de la disciplina de la Inteligencia Artificial (IA) polarizado hacia otros paradigmas. Se propone reivindicar el uso de estos mapas para destacar su potencial.

## 4.3 Innovación e impacto del proyecto

En la actualidad, la investigación y desarrollo sobre los SOM fue opacada por el avance de otras tecnologías de redes neuronales que ofrecieron soluciones atractivas y potentes para la generalidad del mercado consumidor.

Los SOM son una herramienta muy útil y versátil para la visualización e interpretación de datos voluminosos y multidimensionales, permitiendo al usuario facilitar la toma de decisiones.

Si bien existen librerías y herramientas de programación de SOM, no existe hasta el momento un sistema informático que permita que un usuario pueda cargar sus datos, filtrarlos, entrenar un mapa y aprovechar todas las capacidades visuales y de análisis de forma esencialmente interactiva, sin necesidad de programar ni una línea de código, pudiendo exportar resultados.

La inexistencia de un sistema con las características y funcionalidades enunciadas, motiva y deja clara la necesidad de llevar a cabo este proyecto.

En el Laboratorio de Bioingeniería suele ser un requerimiento el análisis multivariable de fenómenos biológicos para encontrar patrones de interés mediante agrupamiento de los datos que puedan ser explicados conceptualmente mediante el análisis de las variables. El software desarrollado permitirá implementar este proceso de forma interactiva.





En el Laboratorio de Procesamiento de Imágenes esta herramienta será de interés para evaluar sistemas de extracción de características en imágenes por medio de otras herramientas de IA, como las redes convolucionales. También se podrá utilizar una ampliación de las características del software para producir imágenes pseudo coloreadas a través de los SOM.



## 5. Planificación del proyecto

### 5.1 Análisis del problema

En la actualidad, estamos transitando la “Era de la Información”. La digitalización del conocimiento nos provee con grandes cantidades de datos que deben ser analizados para detectar patrones, tendencias o para la toma de decisiones.

El análisis de datos de gran volumen y/o con alta dimensionalidad es un problema para los científicos e investigadores debido a la ausencia de herramientas robustas que lo hagan. A su vez, no existe una herramienta dentro del mercado con las funcionalidades necesarias para poder obtener visualizaciones con capacidades gráficas interactivas.

### 5.2 Solución planteada

Como solución al problema se plantea un sistema web que facilite la toma de decisiones a partir del uso de SOM. Para poder lograrlo se desarrollaron dos aplicaciones independientes que se comunican entre sí.

- Una aplicación web, que es el punto de ingreso de datos y de visualización de resultados con las siguientes funcionalidades:
  - Ingreso de los datos en la pantalla inicial.
  - Filtrado de los datos y selección de variables de entrada de interés.
  - Configuración de hiper parámetros de entrenamiento y visualización.
  - Visualización del mapa de celdas hexagonales con información pertinente, generado después del entrenamiento.
  - Agrupamiento en *clusters*.
  - Descarga del mapa generado en formato gráfico.
  - Aplicación en segmentación y pseudocoloreado de imágenes.
  - Descarga de los resultados en formato de texto.



Esta aplicación cuenta con un menú desplegable con varias pestañas para cada una de las funcionalidades del sistema.

- Una Web API que se encarga del procesamiento de datos y del entrenamiento hiper parametrizado de la red neuronal:
  - Comunicación con la aplicación web para recibir los datos de entrenamiento.
  - Hiper parametrización del entrenamiento de los SOM.
  - Entrenamiento del mapa a partir de los datos recibidos.
  - Preprocesamiento de los datos y respuesta para la visualización de todas las pestañas.
  - Manejo de las medidas de calidad de los mapas entrenados.

### 5.3 Equipo de trabajo

El equipo de trabajo para el desarrollo del proyecto está compuesto por 5 integrantes, 3 estudiantes de la carrera de Ingeniería en Informática y dos directores de proyecto que colaboraron desde el inicio hasta el final del mismo.

Es importante resaltar que la herramienta utilizada (SOM) para afrontar el problema planteado es uno de los contenidos de la asignatura “Inteligencia Artificial” de nuestra carrera, la cual ha sido cursada por los estudiantes que participan de este proyecto, quienes aspiran al título de grado.

Por otro lado, la formación de los directores del proyecto, Gustavo Javier Meschino y Diego Sebastián Comas, ambos ingenieros y doctores en Ingeniería, con amplia trayectoria y conocimientos de IA, han sido un recurso imprescindible para consultar y apoyarnos en el desarrollo de la solución.

El excelente ambiente de trabajo y fluida comunicación entre el equipo de trabajo fue esencial para garantizar el avance del proyecto.



## 5.4 Análisis FODA

Mediante el análisis FODA se determinaron los aspectos positivos que pueden impulsar a un mejor desarrollo por parte del equipo (fortalezas y oportunidades) y también resalta los aspectos negativos que podrían obstaculizar su desempeño (debilidades y amenazas) durante el desarrollo del proyecto. Los elementos clave del análisis son:

### 5.4.1 Fortalezas

- El sistema garantiza al usuario un gran aislamiento de la implementación de una red neuronal para el uso de SOM.
- Hemos conformado grupos de trabajo en otras asignaturas, podemos afirmar que conocemos la forma de trabajo de cada uno y sus fortalezas.
- No existe una herramienta visual interactiva integral para el uso de SOM contra la que se competirá.

### 5.4.2 Oportunidades

- Existe la posibilidad de escalabilidad a futuro del proyecto, incorporando nuevas funcionalidades que puedan surgir o el uso de los mapas para otros procesos.
- El proyecto es independiente de la rama o área de trabajo, puede aplicarse al modelo de negocio de cualquier usuario.
- La toma de decisiones es un factor clave tanto para empresas como para investigación. El sistema que planteamos será una herramienta a tener en cuenta.

### 5.4.3 Debilidades

- Si bien aprendimos el tema durante la carrera, no somos expertos en esta área y, por tanto, es necesario un estudio previo sobre la base teórica de los temas a desarrollar.



- Es necesario un mínimo conocimiento de la parametrización del entrenamiento de una red neuronal.
- El desconocimiento parcial de los lenguajes de programación a utilizar es una barrera a superar, ya que nos demandará tiempo.

#### 5.4.4 Amenazas

- Se aprecia una velocidad de crecimiento exponencial en los desarrollos del área de IA, por lo que la herramienta tiene que mostrar superar los enfoques más modernos en algunos aspectos.
- La disposición de tiempos de los integrantes del proyecto puede fluctuar de forma impredecible, generando eventuales atrasos respecto a la planificación de tiempos dedicados al proyecto.

## 5.5 Análisis de riesgos

Es importante realizar un análisis de los riesgos y cuantificar de manera tal de tener en cuenta cuáles deben prevenirse y cuáles remediarse. Hemos identificado aquellos riesgos más relevantes para el proyecto actual. Se detallan en la siguiente tabla.

Riesgos técnicos				
Riesgo	Consecuencia	Probabilidad	Impacto	Riesgo
<b>R1:</b> Bibliotecas externas utilizadas dejan de funcionar	Pérdida de tiempo	2	2	4
<b>R2:</b> Fallo en la comunicación entre aplicación y servidor	Pérdida de datos y resultado erróneo	1	3	3
<b>R3:</b> Interfaz no intuitiva	Uso ineficiente del sistema	1	3	3
<b>R5:</b> Incompatibilidad con navegadores web	Sistema inaccesible	1	3	3
<b>R7:</b> Fallo en el manejo de errores	Falla del sistema	1	3	3
<b>R8:</b> Pérdida de conexión a Internet	Incapacidad de acceso al sistema	1	3	3
<b>R9:</b> Inviabilidad de la implementación de algún requerimiento proyectado	Insatisfacción del demandante.	1	3	3
Riesgos de gestión del proyecto				
<b>R10:</b> Integrante abandona el proyecto	Retrasos en el cronograma	1	3	3
<b>R11:</b> Mala estimación de plazos en la planificación.	Retrasos en el cronograma	2	2	4

Tabla 1: Análisis de riesgos



## 5.6 Estimación de tiempos

A continuación se adjunta la planificación del proyecto, indicando las tareas correspondientes a cada semana.

TAREA	TIEMPO EN HORAS	PLANIFICACIÓN POR SEMANAS																																						
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36			
<b>ANÁLISIS</b>	<b>60</b>																																							
Especificación de los objetivos del sistema	15																																							
Definición de las funcionalidades del sistema	15																																							
Análisis de tecnologías a utilizar	15																																							
Estudio de tecnologías seleccionadas	15																																							
<b>DISEÑO</b>	<b>45</b>																																							
Diseño de la interfaz de usuario	15																																							
Diagramado de la arquitectura del sistema	15																																							
Diagramado de procesos de negocio	15																																							
<b>DESARROLLO</b>	<b>450</b>																																							
Implementación inicial: factibilidad	60																																							
Servidor en Python	120																																							
Módulo de carga de datos y parametros	60																																							
Módulo de visualización e interacción con grilla	60																																							
Módulo de componentes	30																																							
Módulo de clustering	30																																							
Módulo de segmentación y coloreado de imágenes	90																																							
<b>CONCLUSIÓN</b>	<b>159</b>																																							
Validación de requerimientos con demandantes	12																																							
Ajustes correspondientes a retroalimentación	12																																							
Pruebas de integración	30																																							
Informe: parte técnica	45																																							
Informe: gestión de proyecto	45																																							
Informe: revisión y publicación	15																																							
<b>SUMA DE HORAS TOTAL</b>	<b>714</b>																																							

Figura 1: Planificación del proyecto



## 6. Mapas autoorganizados como herramienta

### 6.1 Definición

Un mapa auto organizado (SOM) es una técnica de aprendizaje automático (llamado también *machine learning*) usada para producir representaciones de datos multidimensionales en una baja dimensionalidad, usualmente dos dimensiones, preservando la estructura de los datos (sus agrupamientos, sus distancias). Son redes neuronales que se entrenan con paradigma de aprendizaje no supervisado, formadas por arreglos de celdas que mapean un espacio de entrada (espacio de datos) a un espacio de celdas (espacio de salida) generalmente 2D, conservando la topología del espacio de entrada, con notables capacidades para eliminar el ruido, detectar outliers y completar datos faltantes<sup>(2)</sup>.

Un conjunto de datos multivariable con D dimensiones, con diferentes instancias en N observaciones puede ser representado con agrupamientos o subgrupos de observaciones con valores similares para las variables. Estos agrupamientos pueden ser visualizados en un mapa de dos dimensiones de manera que aquellos datos más cercanos topológicamente presentan más similaridad en el espacio de los datos que aquellos en agrupamientos más distantes. De esta manera, se logra que datos con alta dimensionalidad sean fáciles de ver y analizar. A su vez, tienen como objetivo facilitar la extracción de conclusiones útiles en base a los agrupamientos hallados.

### 6.2 Surgimiento de la herramienta y su implementación

Los SOM fueron creados e implementados por el científico finlandés Teuvo Kohonen en la década del 80 y presentados formalmente en un artículo en 1990<sup>(1)</sup>. Estos mapas se inspiran en los órganos sensoriales que captan información del entorno. El ser humano tiene diferentes sensores para los cinco sentidos. Luego, esa información se procesa y almacena en el cerebro. El cerebro está conformado por





neuronas que se representan en capas bidimensionales, en mapas de características. El científico Kohonen comprobó que la reiterada aparición de estímulos externos fuerza la formación de mapas. Estas neuronas artificiales se ubican en grillas conformando el mapa, en donde no están conectadas entre sí pero tienen influencia unas con otras según cuán cercanas son. Si hay un efecto de cambio en una neurona, este efecto se propaga a las celdas vecinas. Estas neuronas son llamadas celdas, y así se las referirá de aquí en más. Los SOM tienen dos características principales. En primer lugar, se tiene un vector de entradas conectado a cada una de las celdas. En segundo lugar, cada neurona querrá ser la ganadora cuando aparezca un dato de entrada.

Los elementos que componen el mapa se llaman “nodos” o “celdas”, como mencionamos, que usualmente están diagramadas en una grilla hexagonal o rectangular de dos dimensiones. Generalmente se utilizan grillas hexagonales ya que cada celda tiene seis vecinos ubicados a igual distancia topológica. Por otro lado, en las grillas rectangulares se tienen ocho vecinos por celda pero a diferente distancia. La cantidad de celdas y su disposición son especificadas antes del entrenamiento basándose en los objetivos de análisis y exploración de los datos.

Cada nodo del mapa está asociado a un vector de pesos que tiene la misma dimensión que los datos de entrada. El conjunto de los vectores de pesos de todos los nodos se denomina *codebook*. A pesar de que las celdas del mapa mantienen su posición fija, la fase de entrenamiento consiste en ajustar estos vectores de forma de representar de la mejor manera los datos de entrada (puede usarse, por ejemplo una medida de similitud como la distancia euclidiana, pero sabiendo que hay otras). Luego del entrenamiento, el mapa puede ser usado para inferir la cercanía de nuevos datos de entrada adicionales con los de entrenamiento, encontrando el nodo con el vector de pesos más cercano a cada uno.

Los SOM operan de forma similar a la mayoría de las redes neuronales artificiales: entrenamiento y luego consulta (también llamada inferencia). En primer lugar, la fase de entrenamiento usa un conjunto de datos de entrada para generar una representación en una dimensión inferior a la provista. Luego de esto, en la fase de



consulta, se toman nuevos datos y se los ubica donde corresponda partiendo del mapa generado anteriormente.

Durante la fase de entrenamiento, el mapa aprende a partir de la información contenida con el objetivo de autoorganizar los datos. Lo primero que se debe hacer es inicializar el mapa. Esto quiere decir, asignar vectores prototipo al *codebook*. Existen varios métodos de inicialización. Uno de ellos es el de asignación aleatoria que se refiere a asignar valores iniciales aleatorios a los pesos. Por otro lado también se tiene la inicialización utilizando *Principal Component Analysis* (PCA). El PCA es un método estadístico utilizado para describir un conjunto de datos en términos de nuevas variables no correlacionadas. Los componentes se ordenan por la cantidad de varianza original que describen, por lo que la técnica es útil para reducir la dimensionalidad de un conjunto de datos<sup>(3)</sup>. Se basa en inicializar linealmente, considerando los autovalores y autovectores de los datos de entrenamiento. Dentro de esta inicialización, lo primero que se hace es centrar los datos en su espacio. Esto se logra calculando la media de cada componente de todos los datos de entrenamiento y restando esta a cada una de ellos. A su vez se le asigna a cada nodo un vector prototipo con la media de cada componente calculada en el paso previo. Luego se asignan coordenadas (x, y) para cada nodo del mapa y se normalizan estas coordenadas para el rango [-1,1]. Por ejemplo, para un mapa de 2x2 se tendrían 4 coordenadas.

Luego, se realiza el PCA sobre los datos de entrenamiento, quedándonos con los dos vectores y autovalores más grandes y significativos, ya que se trata de un mapa bidimensional. Estos dos son los que mejor describen a los datos de entrada ya que son las dos direcciones en la que los datos más crecen. Posteriormente, se escalan los vectores propios multiplicando por él su valor propio. Finalmente, se realiza una transformación lineal entre los valores centrados, las coordenadas y los autovectores. Este proceso se repite para las demás coordenadas y nodos. De esta manera, como los pesos iniciales ya están alineados con la estructura principal de los datos, el SOM requiere menos ajustes grandes durante el entrenamiento, lo que facilita una convergencia más rápida y estable.

Previo a la fase de entrenamiento, se requiere una normalización de los datos, un método vital de preprocesamiento, mapeo y escalamiento que ayuda a que los modelos de pronóstico y predicción sean precisos<sup>(4)</sup>. Este proceso es fundamental en cualquier algoritmo que considere distancias vectoriales. El rango de datos actual se transforma en un nuevo rango estandarizado utilizando este método. Esto es sumamente importante y se realiza debido a que se tienen variables con diferentes unidades de medida y diferente escala. Se trata de armonizar las técnicas de predicción y pronóstico dispares. Se busca estandarizar el rango de las variables del conjunto de datos para que todas pesen lo mismo en el cálculo de distancias.

Existen varias maneras de poder realizarlo. Una de ellas es la normalización min-max. Este método de normalización de datos implica transformar los datos originales de forma lineal. Se obtienen los valores mínimo y máximo de los datos y, a continuación, se modifica cada valor utilizando la fórmula que se muestra a continuación:

$$X_{normalization} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

*Ecuación 1: Normalización min-max*

En la ecuación 1, se le resta a X su valor mínimo y luego se divide esta diferencia por el rango de la variable.

Otra normalización común es la denominada **normalización por varianza**<sup>(5)</sup>. El primer paso implica restar el valor medio de cada característica de todos los puntos de datos. Esto centra los datos alrededor del cero, lo que significa que la nueva media de la característica se convierte en cero. El segundo paso implica dividir cada característica centrada por su desviación estándar. Esto escala los datos de modo que la varianza de cada característica se convierte en uno, de ahí el término "varianza unitaria". Se puede expresar de la siguiente manera:

$$X_{normalization} = \frac{x - \mu}{\sigma}$$

Ecuación 2: Normalización por varianza

Esta técnica reduce el impacto de los valores atípicos al reducir el rango de valores y hacer que la distribución sea más parecida a la gaussiana. A su vez, ayuda a que los algoritmos converjan más rápido y evita que ciertas funciones dominen a otras.

La fase de entrenamiento consiste en dos pasos. El primer paso consiste en ver cual es la neurona “ganadora” mediante alguna medida. Generalmente mediante la obtención de la menor distancia euclidiana entre el dato y todos los vectores prototipo, pero podría elegirse otros criterios de distancia. Esta neurona es denominada BMU<sup>(6)</sup>. Es la neurona o celda cuyo vector prototipo es el “más similar” al dato de entrada.

$$\|X_i - w_{BMU}\|$$

Ecuación 3: Valor mínimo de distancia

El segundo paso es adaptar los pesos. El criterio será premiar al vector prototipo que contiene la BMU, acercándose más al dato de entrenamiento. Esto se hace de a poco según una *Learning rate* (constante de aprendizaje) como las que se utilizan en otros procesos iterativos de entrenamiento. No solo se modifica la neurona ganadora, sino también las celdas vecinas según una función de vecindad.

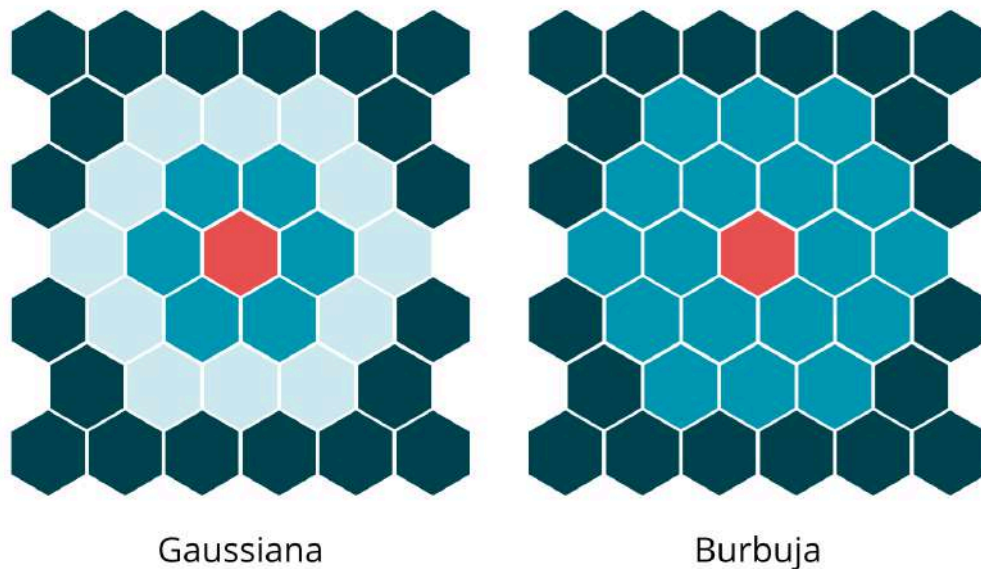
$$W_k(n + 1) = W_k(n) + \alpha(n) h_k(n) [X_i - W_k(n)]$$

$\alpha(n)$ : Tasa de aprendizaje

$h_k(n)$ : Función de vecindad

Ecuación 4: Adaptación de los pesos

Dentro de las funciones de vecindad, se tiene la forma gaussiana en donde se modifica en menor medida cuando nos alejamos de la BMU. Esto sucede ya que se tiene como centro a la BMU, y luego el valor decae a medida que nos alejamos. Por otro lado, se tiene el tipo burbuja en donde se modifica de la misma manera a todas las celdas hasta cierto umbral. Se puede usar una combinación de ambas. En la siguiente figura se puede observar una comparativa de ambos métodos.



*Figura 2: Comparación de funciones de vecindad*

Se tienen dos requerimientos principales para que el entrenamiento del mapa sea correcto. En primer lugar, dos datos cercanos en el espacio de los datos deben mapearse en celdas aledañas. Deben ser mapeados a la misma BMU o en celdas adyacentes. En segundo lugar, los vectores prototipo deben ser lo más parecidos a los datos de entrenamiento.

Para estos requerimientos se tienen medidas de calidad. No se puede utilizar un error con respecto a lo que queríamos lograr ya que es un entrenamiento no supervisado.

Por un lado, se tiene el error de cuantización. Este error se calcula haciendo el promedio de las distancias entre cada dato y el vector prototipo de su BMU. Nos



dice en qué medida hemos podido representar los datos de entrenamiento con el *codebook*. Lo que ayuda a saber que tan bien hemos podido cumplir el primer requerimiento de los SOM.

El error de cuantización tiene varias desventajas. En primer lugar, no tiene en cuenta la topología del mapa, es decir, en dónde se encuentran las BMU de cada dato. A su vez se ha demostrado que el valor del error decrece a medida que el mapa se hace más grande. Esto se debe a que se tienen más vectores prototipo para representar los datos.

Por otro lado se tiene el error topológico. El error se define contando qué porcentaje de los datos encuentran su primera y segunda BMU adyacentes en el mapa. Nos ayuda a saber cuán similares son los vectores prototipo de celdas cercanas. Lo que determina que tan bien hemos cumplido el segundo requerimiento mencionado anteriormente. Una de sus desventajas es que no es confiable cuando se utilizan mapas pequeños ya que se tienen pocas celdas, y la probabilidad de que se asignen a celdas adyacentes es más alta.



## 7. Ejecución del proyecto

### 7.1 Elicitación de requerimientos

#### 7.1.1 Requerimientos funcionales

##### *Ingreso datos*

**RF01:** El sistema debe permitir a los usuarios ingresar los datos de entrada, en formato CSV.

**RF02:** El sistema debe permitir a los usuarios filtrar los datos ingresados, y seleccionar las columnas que se van a considerar para el entrenamiento.

**RF03:** El sistema debe permitir al usuario configurar los parámetros relevantes para el entrenamiento.

**RF04:** El sistema debe permitir al usuario ingresar datos resultantes de un entrenamiento previo generado por la aplicación, en formato JSON.

##### *Server*

**RF05:** El servidor deberá recibir los datos de la aplicación web, utilizar la respectiva biblioteca para el entrenamiento y devolver el resultado.

##### *Visualización de mapas*

**RF06:** La aplicación debe permitir la visualización del mapa generado tras el entrenamiento.

**RF07:** La visualización del mapa debe ser interactiva, permitiendo al usuario seleccionar hexágonos y acceder a información del mismo.

**RF08:** El sistema debe permitir una visualización más precisa de las distintas grillas permitiendo acercar o alejar la vista de los hexágonos.

**RF09:** El sistema debe permitir la descarga de las métricas del entrenamiento y el *codebook* generado. Ambos en formato txt.

##### *Mapa de componentes*

**RF10:** El sistema debe permitir visualizar los mapas de componentes.

**RF11:** Debe permitir seleccionar qué componentes quiere mostrar el usuario.



### *Clustering*

**RF12:** El sistema debe permitir agrupar los datos en *clusters* basados en similitudes, y permitir que el usuario ingrese la cantidad de *clusters*.

### *Hits*

**RF13:** El sistema debe permitir visualizar en el mapa de hexágonos, la cantidad de hits por hexágono.

**RF14:** El sistema debe permitir visualizar las etiquetas que tienen los datos de entrada que cayeron en los hexágonos.

**RF15:** El sistema debe permitir la visualización del hit mayoritario dentro de cada hexágono.

### Descarga de mapas

**RF16:** El sistema permitirá al usuario descargar los mapas generados en formato *png*.

### *Nuevo dato*

**RF17:** El sistema debe permitir cargar nuevos datos y utilizar el codebook entrenado del SOM para identificar las BMUs correspondientes y visualizarlas en la grilla de hexágonos.

### *Segmentacion de imagenes*

**RF18:** El sistema permitirá la segmentación y coloreado de imágenes, a partir de datos que resulten de una extracción de características.

**RF19:** El sistema permitirá la descarga de la matriz de colores de la imagen generada en formato *txt*.

**RF20:** El sistema permitirá la descarga de la imagen generada en formato *png*.

## 7.1.2 Requerimientos no funcionales

**RNF01:** La interfaz de usuario debe ser intuitiva y fácil de usar.

**RNF02:** El sistema debe estar bien documentado y ser escalable, para que otras personas en el futuro puedan seguir el trabajo.

**RNF03:** El sistema debe ser compatible con variedad de dispositivos y navegadores web.

**RNF04:** La comunicación entre la aplicación web y el servidor debe ser fluida y eficiente.



## 7.2 Diseño del sistema

Dentro de esta sección se explican las decisiones de diseño tomadas a lo largo del desarrollo del proyecto. Se analiza tanto la arquitectura de la solución general, como del frontend y backend. Finalmente, se explican y detallan las decisiones acerca de las tecnologías utilizadas durante el desarrollo.

### 7.2.1 Modelado de procesos

En los siguientes diagramas de proceso de negocios se muestra como es la interacción del usuario con el sistema en todas sus funcionalidades. La idea es mostrar cómo son los procesos internos del sistema y explicar gráficamente su funcionamiento.

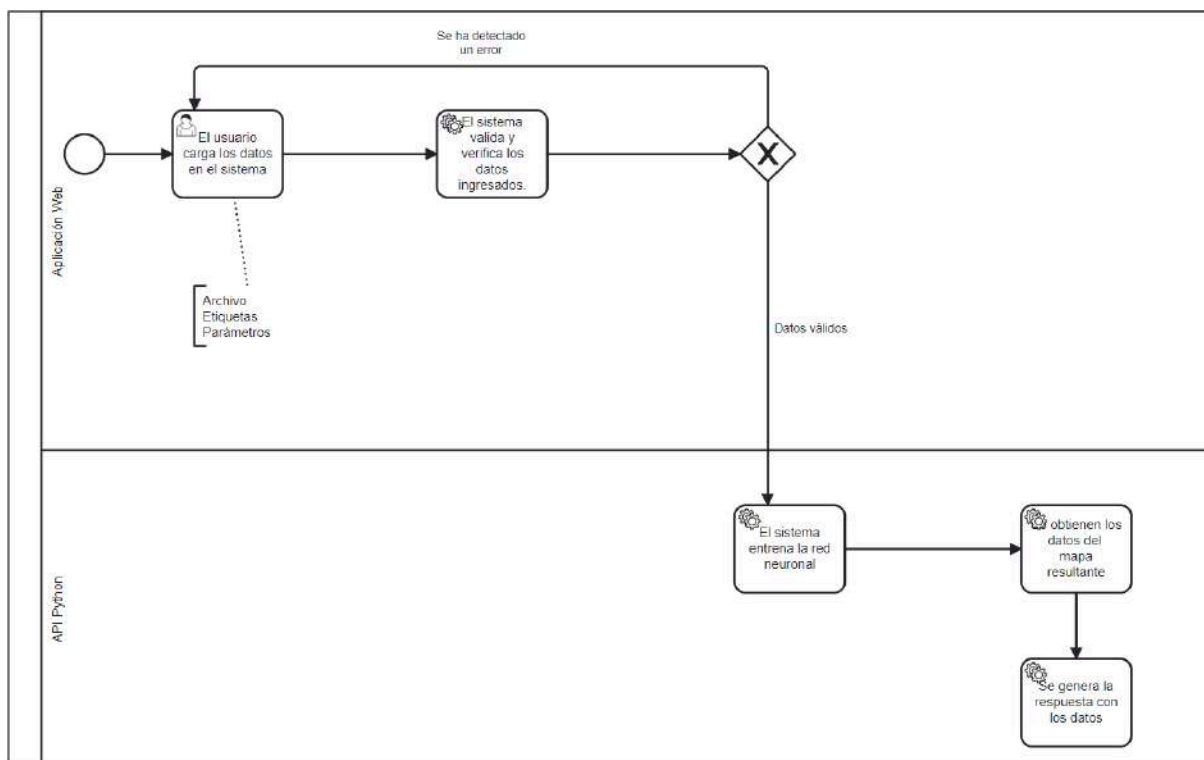


Figura 3: Entrenamiento

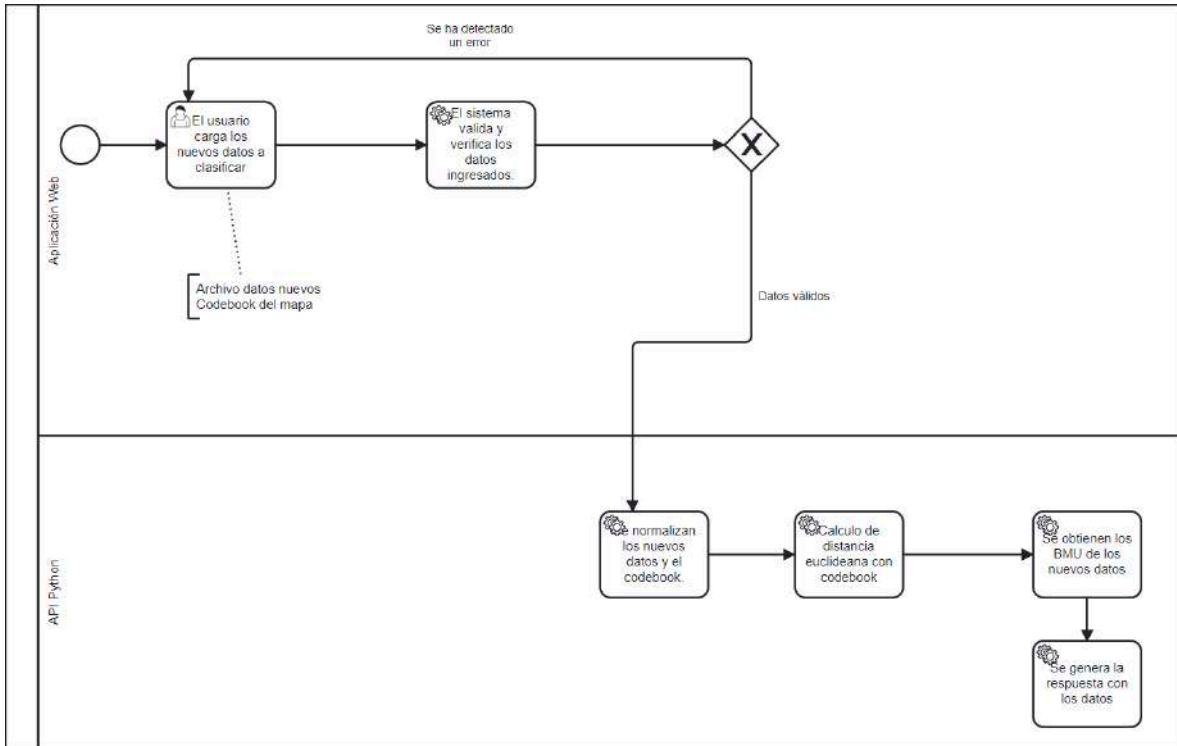


Figura 4: Nuevos datos

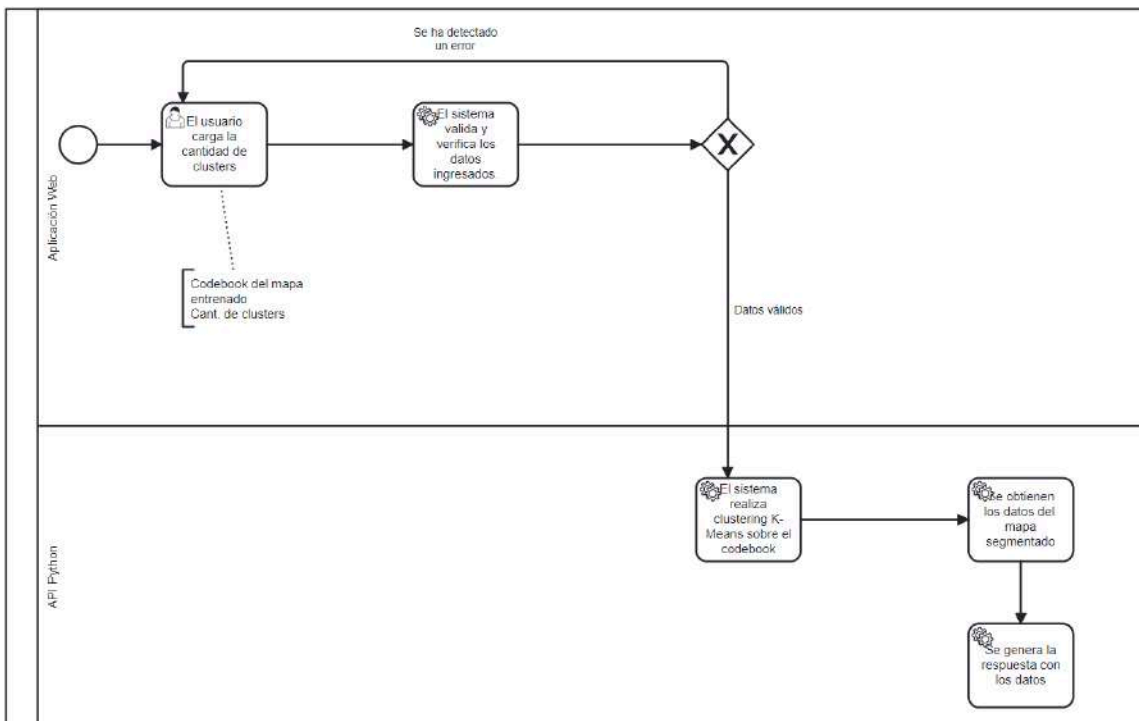


Figura 5: Clustering

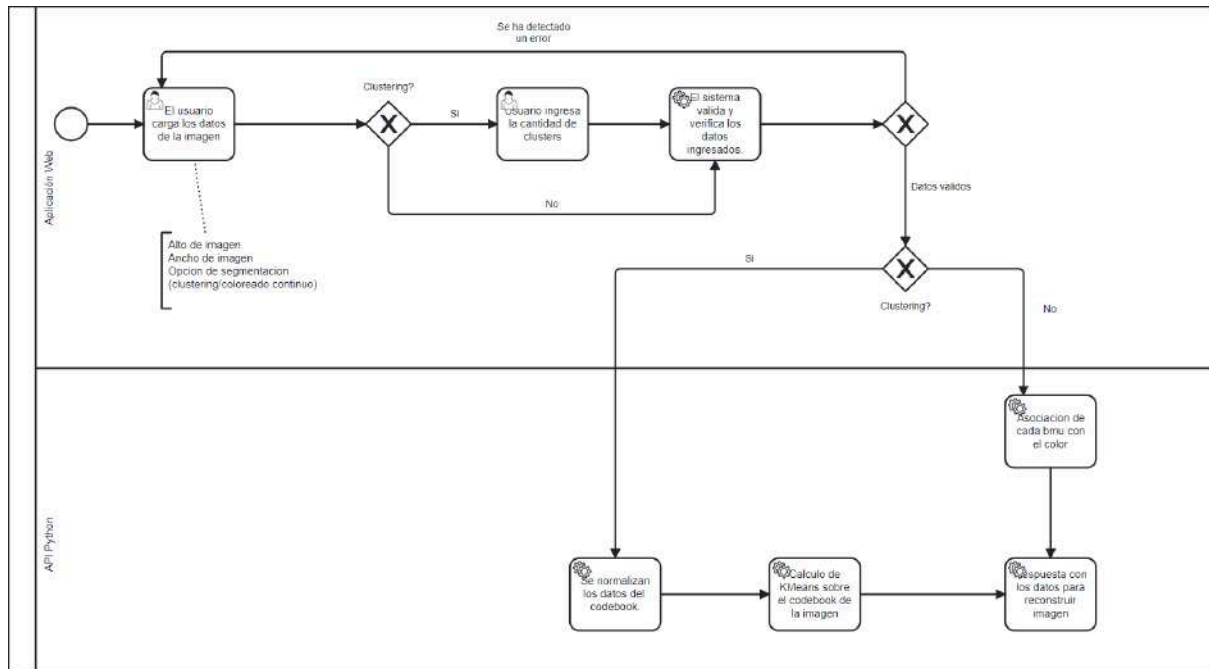


Figura 6: Segmentación de imágenes

## 7.2.2 Arquitectura del sistema

A continuación, se presenta un diagrama en el que se muestran las dos aplicaciones principales e independientes del sistema, y cómo interactúan entre sí para cada uno de los procesos. La arquitectura general sigue un modelo de cliente-servidor, en dónde el cliente es la aplicación web y el servidor es una API encargada de conectarse con las librerías de entrenamiento y realizar la respuesta con los datos a la aplicación web para su correcta visualización <sup>(7)</sup>.

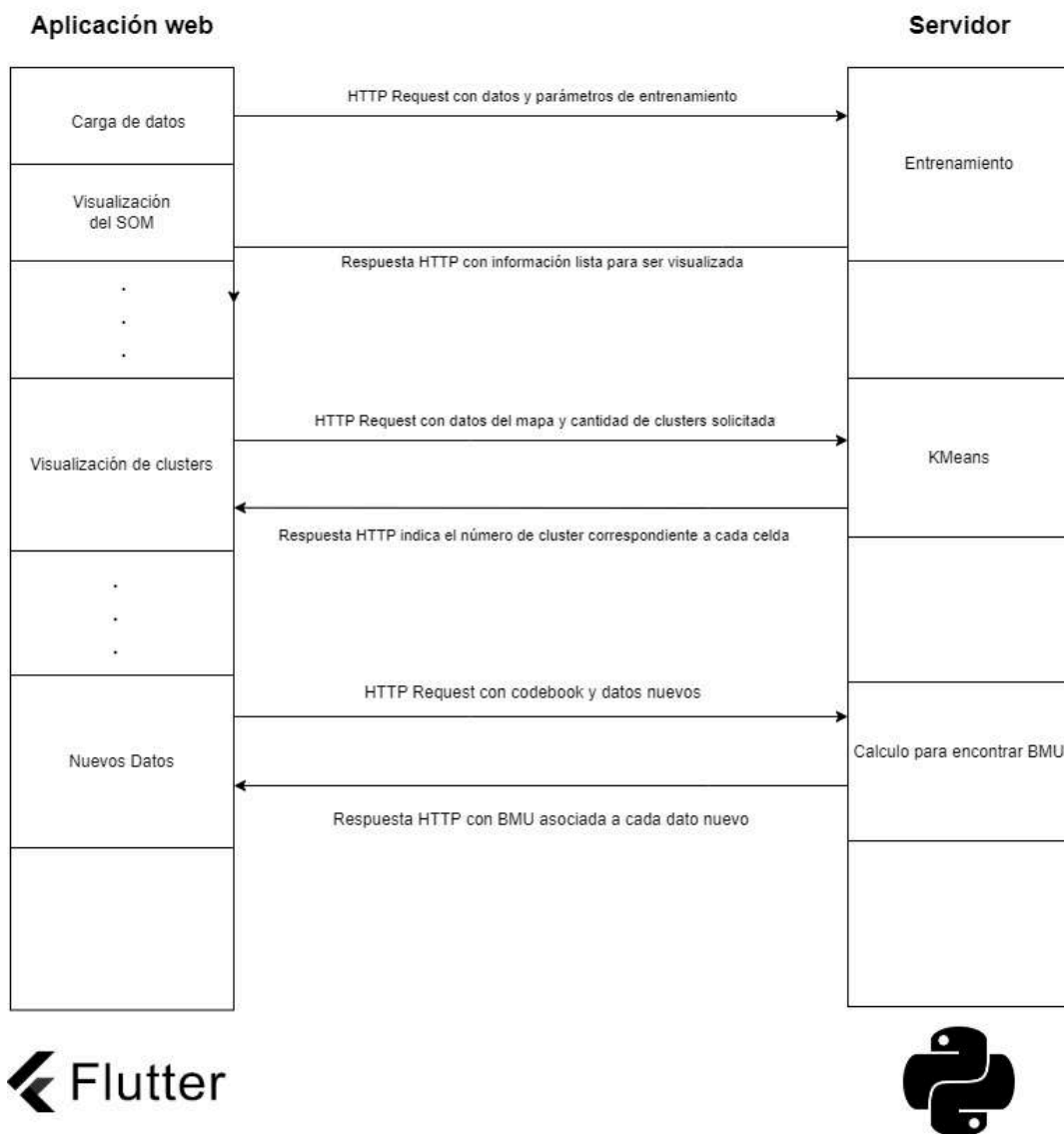


Figura 7: Arquitectura del sistema

## 7.3 Diseño de interfaces gráficas

Para el diseño de la parte gráfica de la aplicación se decidió por el uso de la aplicación *Moqups* que es una aplicación web simplificada e intuitiva que te ayuda a crear y colaborar en esquemas, maquetas, diagramas y prototipos<sup>(8)</sup>. Es una aplicación que nos permitió crear el prototipo de las interfaces de manera online, y con la posibilidad de colaborar entre nosotros en el armado. A su vez, esta aplicación ofrece un plan gratuito al registrarse que no tiene límite de tiempo, pero sí tiene algunas restricciones ya que ofrece 2 proyectos activos, un límite de 400 objetos por proyecto, 25 MB de almacenamiento de imágenes y 7 colaboradores. Algo que no nos limitó a lo largo del desarrollo. A continuación, se detallan los prototipos para cada una de las pantallas del sistema.

### 7.3.1 Carga de datos

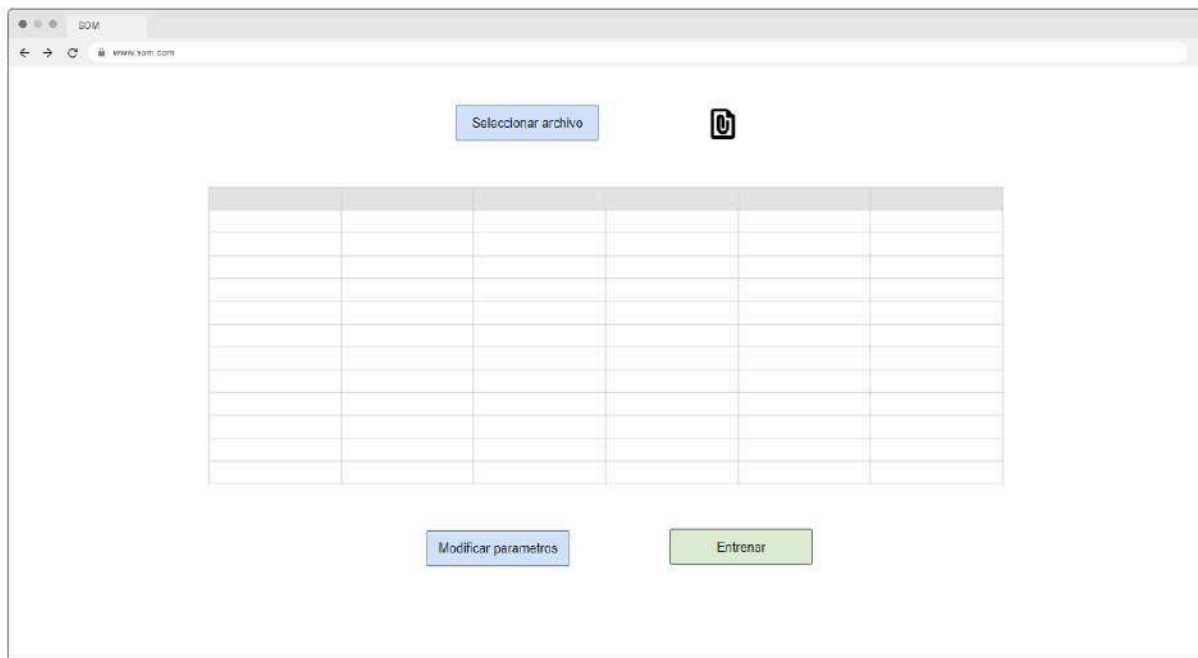


Figura 8: Prototipo de pantalla de inicio

### 7.3.2 Visualización del mapa entrenado

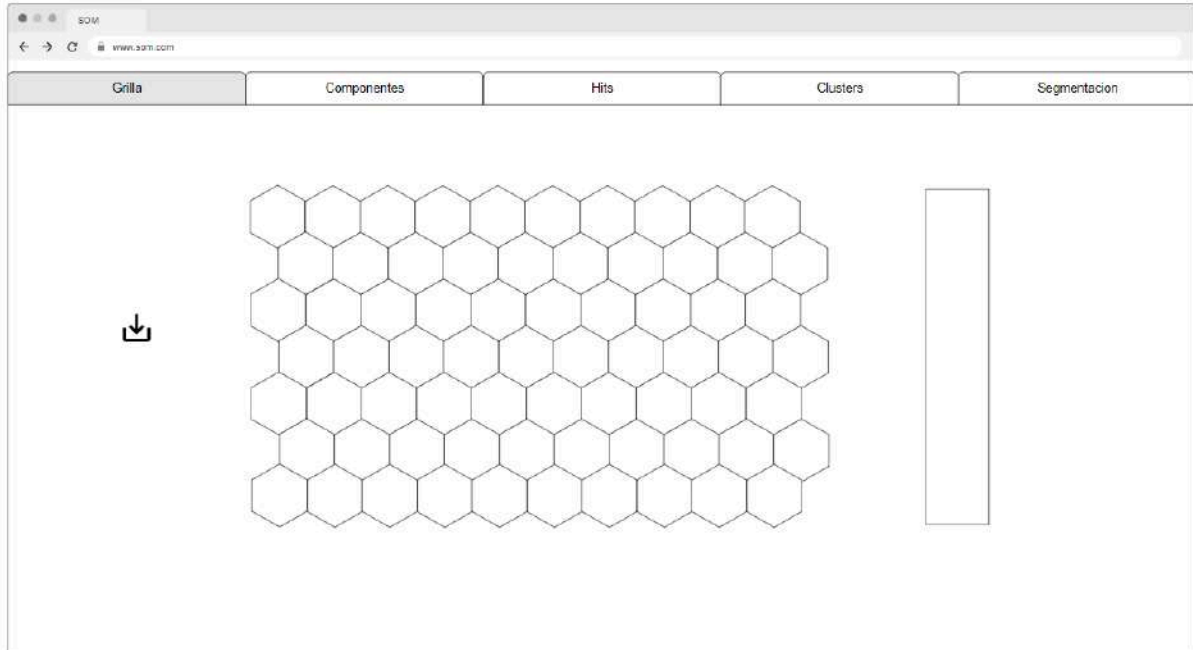


Figura 9: Prototipo de pestaña de visualización de mapa

### 7.3.3 Mapas de componentes

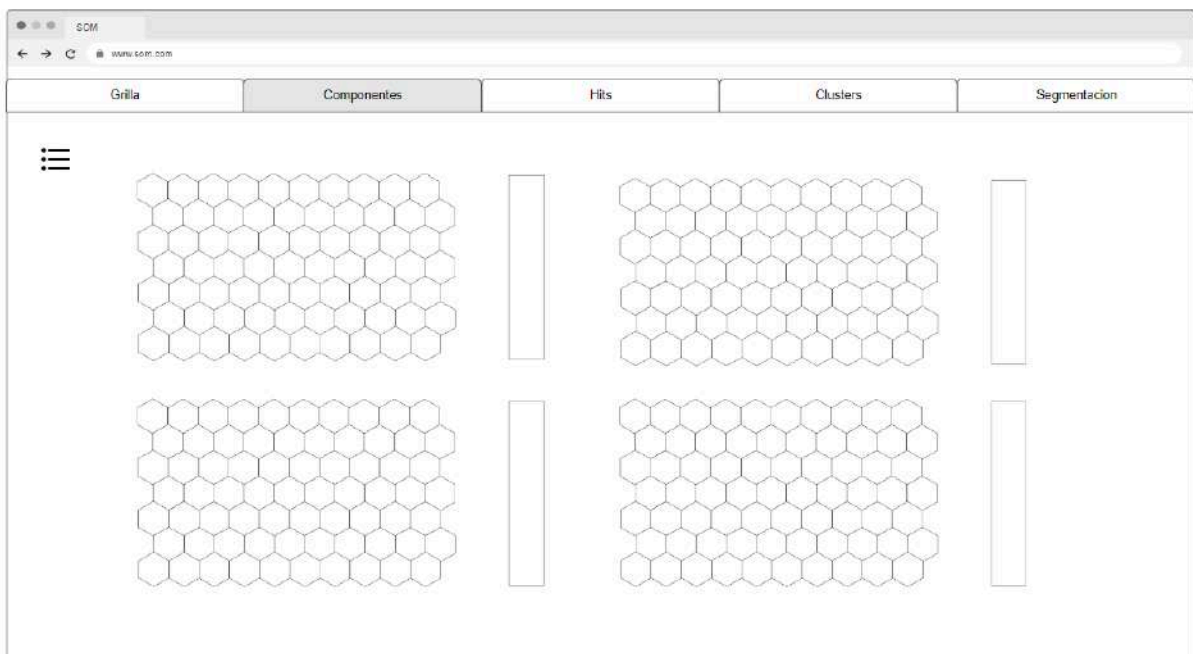


Figura 10: Prototipo de pestaña de mapa de componentes

### 7.3.4 Hits/Etiquetas

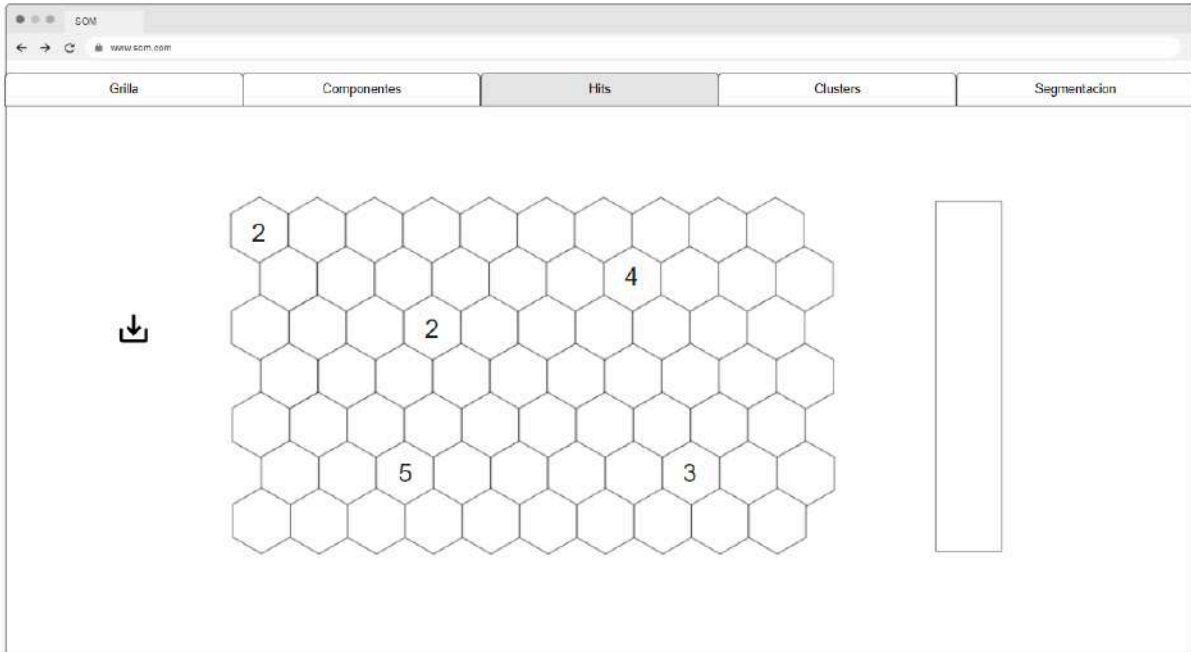


Figura 11: Prototipo de pestaña de etiquetas

### 7.3.5 Clustering

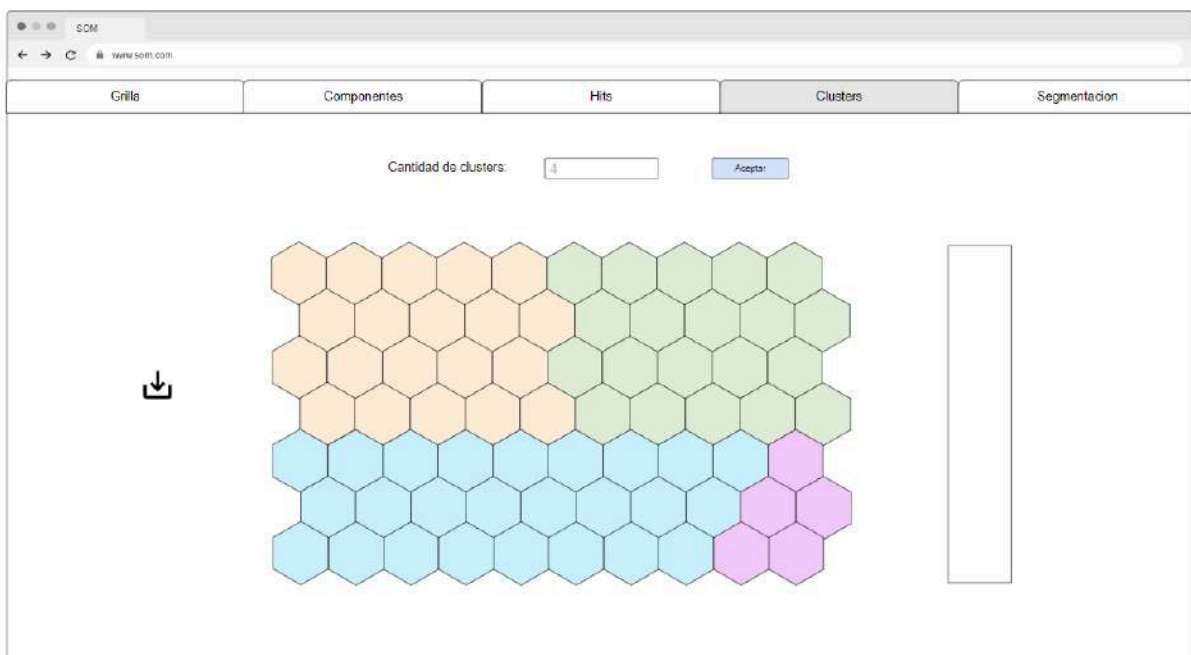


Figura 12: Prototipo de pestaña de clustering

### 7.3.6 Segmentación de imágenes

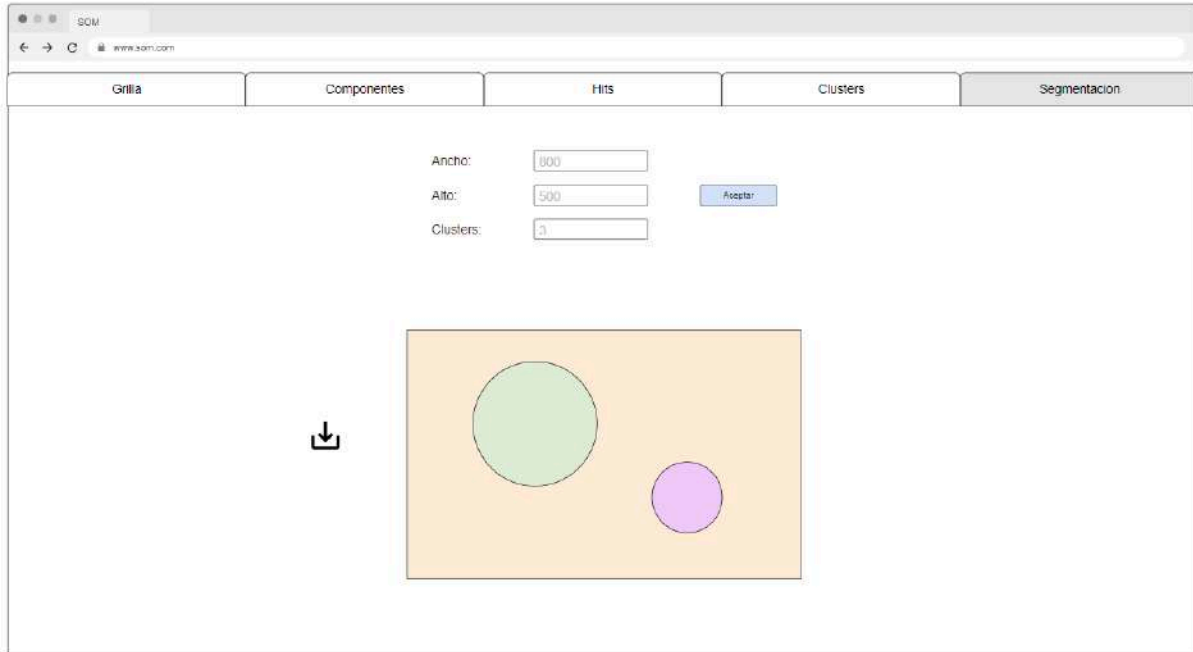


Figura 13: Prototipo de pestaña de segmentación de imágenes

## 7.4 Tecnologías

### 7.4.1 Backend

Para el desarrollo de la Web API se decidió y se optó por usar el lenguaje Python y el framework web Flask<sup>(9)</sup>. La decisión en el uso de Python se basó en que para el entrenamiento de la red neuronal y para el procesamiento de datos se utiliza como base una librería enfocada en el entrenamiento de SOM llamada IntraSOM<sup>(10)</sup>. Es una librería de código abierto basada en Python que se encuentra en constante actualización. A su vez, se utilizó Flask ya que es un framework minimalista escrito en Python que dispone de un conjunto de facilidades para el desarrollo de servidores HTTP y APIs RESTful altamente escalables. A su vez, es gratuita y es una de las más utilizadas dentro de la actualidad por lo que tiene una comunidad activa, algo muy importante ya que se encuentra en constante actualización y mantenimiento.





## 7.4.2 Frontend

El lenguaje de programación elegido para el desarrollo de la aplicación web fue Dart en conjunto con el SDK para interfaces de usuario llamado Flutter. Dart es un lenguaje de programación de código abierto, creado por Google en 2011 como una alternativa más moderna a JavaScript<sup>(11)</sup>. A su vez, Flutter fue desarrollado por Google en 2017 y es usado internamente dentro de sus aplicaciones de *Google Play* y *Google Earth*<sup>(12)</sup>. En el último año, ha sufrido un crecimiento muy grande en cuanto a su popularidad. Eso se debe a su velocidad de desarrollo, experiencia nativa y renderización de la interfaz. La decisión del uso de estos dos lenguajes se basó en que uno de los requerimientos del sistema era la visualización e interacción con las grillas hexagonales que corresponden a los mapas generados. Flutter posee un paquete llamado *Hexagon Grid* que contiene todas las funcionalidades necesarias para lograrlo.

## 8. Módulos

### 8.1 Carga de datos y parametrización

La primera pantalla de nuestro sistema está diseñada para la carga de datos y la configuración del algoritmo SOM. En esta pantalla, se encuentra un botón que permite seleccionar el archivo CSV deseado desde el sistema de archivos local del usuario. Una vez que se selecciona y carga el archivo, el contenido se muestra en un formato de tabla dentro de la interfaz, proporcionando una visualización clara y organizada de los datos.

La interfaz de carga de datos permite a los usuarios realizar configuraciones para adaptar el proceso de entrenamiento a sus necesidades. Entre estas opciones se incluye la capacidad de personalizar el gradiente de colores que se utilizará para la visualización del mapa autoorganizado. Los usuarios pueden elegir entre diferentes esquemas de colores.

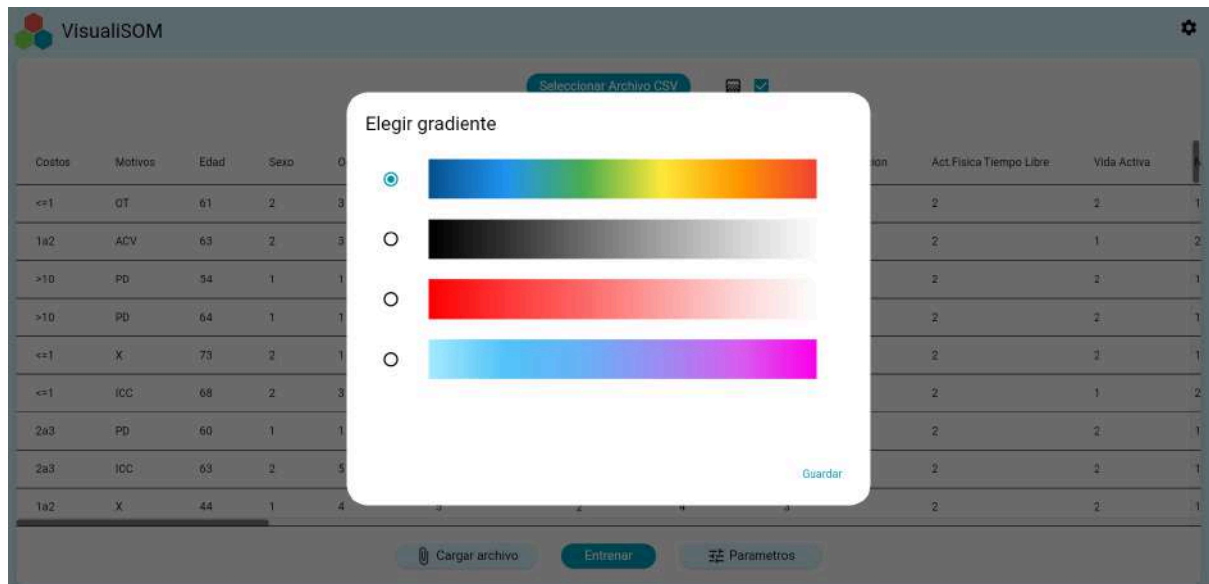


Figura 14: Personalización de gradiente

El sistema también ofrece funcionalidades para la selección de variables de los datos. Los usuarios pueden decidir qué variables serán consideradas para el entrenamiento del SOM y cuáles columnas del archivo se utilizarán como etiquetas para los datos. Para simplificar esta tarea, el sistema ofrece una opción de "Identificación automática de features" mediante un checkbox. Al activar esta opción, el sistema detecta automáticamente las columnas de tipo *string*, que suelen funcionar como etiquetas y las selecciona de forma predeterminada como tales.

The screenshot shows the VisualiSOM interface. At the top, there is a header with the logo and the text 'VisualiSOM'. Below the header, there is a button 'Seleccionar Archivo CSV' and a checkbox that is checked. Below that, there are two tabs: 'Features' and 'Etiquetas'. The main part of the interface is a table with the following columns: Costos, Motivos, Edad, Sexo, Ocupacion, Ingreso Economico, Estado Civil, Convivencia, Grado de Instruccion, Act.Fisica Tiempo Libre, and Vida Activa. The table contains 10 rows of data. At the bottom of the interface, there are three buttons: 'Cargar archivo', 'Entrenar', and 'Parametros'.

Costos	Motivos	Edad	Sexo	Ocupacion	Ingreso Economico	Estado Civil	Convivencia	Grado de Instruccion	Act.Fisica Tiempo Libre	Vida Activa
<=1	OT	61	2	3	1	3	4	3	2	2
1a2	ACV	63	2	3	2	3	4	2	2	1
>10	PD	54	1	1	1	3	4	4	2	2
>10	PD	64	1	1	1	2	4	1	2	2
<=1	X	73	2	1	1	1	2	1	2	2
<=1	ICC	68	2	3	3	2	2	3	2	1
2a3	PD	60	1	1	1	2	2	2	2	2
2a3	ICC	63	2	5	1	2	4	3	2	2
1a2	X	44	1	4	5	2	4	3	2	2

Figura 15: Pantalla de inicio

Además de la carga y configuración de datos, la pantalla tiene un botón que permite ajustar diversos hiperparámetros del entrenamiento del SOM. Entre los parámetros ajustables se incluyen:

- Cantidad de filas de celdas
- Cantidad de columnas de celdas
- Cantidad de iteraciones de entrenamiento
- Cantidad de iteraciones de refinamiento (entrenamiento fino)
- Función de vecindad (gaussiana o burbuja)
- Inicialización (aleatoria o PCA)

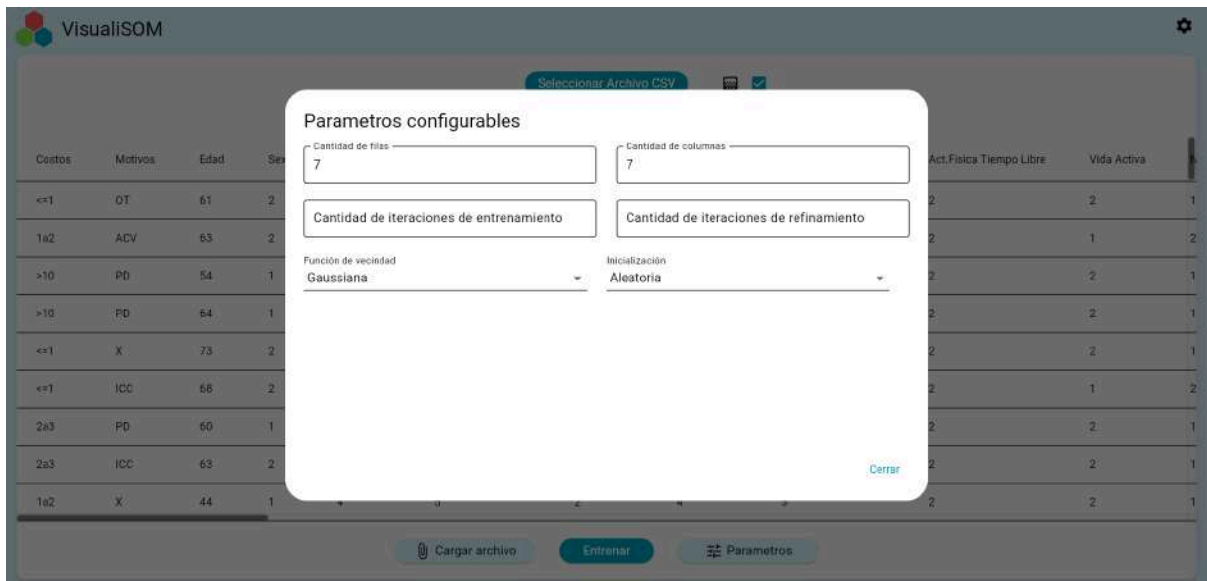


Figura 16: Configuración de parámetros

Una vez que todas las configuraciones han sido ajustadas y los datos han sido preparados, los usuarios pueden iniciar el proceso de entrenamiento mediante el botón "Entrenar". Este botón envía una solicitud a la API en Python que ejecuta el algoritmo SOM utilizando los parámetros y datos proporcionados.

A su vez, la pantalla tiene un botón que permite la carga de un archivo que contenga un entrenamiento realizado previamente en formato JSON<sup>(13)</sup>. Con la carga de este archivo se cargarán todos las pestañas sin la necesidad de realizar nuevamente el entrenamiento del mapa, para así poder continuar con el análisis y estudio previamente realizado.

## 8.2 Entrenamiento

A partir de los datos proporcionados por el usuario, es necesario realizar un entrenamiento de una red neuronal para aprender a clasificar (sin supervisión), lo que implica que no necesitamos un objetivo o target, sino que se generará una distribución a partir de las características de los datos de entrenamiento.



A diferencia de los algoritmos de aprendizaje supervisado, no existe una función objetivo para minimizar, por lo que el algoritmo de aprendizaje de los SOM se rige por dos principios:

- celdas cercanas tienen vectores de peso cercanos<sup>1</sup>
- Cada dato de entrada es representado por una única neurona, que será aquella que tenga el vector de pesos más similar.

En el caso del proyecto que hemos llevado a cabo, hemos delegado parte del entrenamiento a una librería externa, llamada IntraSOM. A pesar de que en la actualidad existen múltiples alternativas, hemos elegido esta debido a que es gratuita, libre (código abierto) y está actualmente mantenida y actualizada.

IntraSOM es una librería implementada en Python desarrollada por un centro de investigación de la Universidad de San Pablo, el *Integrated Technology for Rock and Fluid Analysis (InTRA)*. Esta librería tiene como objetivo lograr que las técnicas de SOM sean más accesibles para investigadores y profesionales proporcionando un framework para expandir e implementar otros algoritmos basados en SOMs.

Por lo tanto, entre el software que hemos generado y entregado, se encuentra una API en Python, que implementa la librería nombrada anteriormente, y tiene como objetivos la comunicación entre la interfaz gráfica y el procesamiento de los datos antes de comenzar el entrenamiento. A su vez, es la encargada de realizar el entrenamiento y formar la respuesta con los resultados del mismo. Además de procesar y realizar el entrenamiento, se encarga de detectar a qué celda del mapa pertenecen los nuevos datos (BMU) en base a similitud con el vector prototipo. Finalmente y no menos importante, realiza el algoritmo de agrupamiento para poder agrupar a las celdas del mapa en cuanto a similitud para luego poder sacar conclusiones. Todas estas funcionalidades están explicadas con mucho mayor detalle a continuación en cada uno de los módulos de la aplicación.

---

<sup>1</sup> Es importante recordar que hablar de cercanía entre dos vectores implica que su distancia euclidiana es pequeña en comparación a otro vector.



### 8.3 Visualización e interacción con grilla

Una vez finalizado el proceso de entrenamiento, el sistema redirige a la siguiente pantalla, donde se despliega un menú de opciones. La opción predeterminada es "Mapa". Las primeras dos opciones de este menú son: Mapa y U-Matrix (Umat+).

En la opción Mapa, cada hexágono de la grilla representa una neurona, que corresponde a un vector dentro del codebook. Dicho vector tiene la misma dimensionalidad que los datos de entrada, lo que permite compararlo con cada dato durante el proceso de entrenamiento.

Durante el entrenamiento, el SOM identificó la neurona cuya representación vectorial es la más cercana a un determinado dato de entrada, y la designó como su BMU. Por lo tanto, cada dato "cayó" en una neurona específica, y dicha neurona puede llegar a ser BMU de uno, varios, o ningún dato, dependiendo de la distribución de los datos y la estructura de la grilla.

En la opción Umat+, la grilla se amplía. Dentro de esta opción, se tiene la representación de la U-Matrix o Matriz de distancias. Aunque conserva la estructura hexagonal original de la opción Mapa, en esta representación se introducen nuevas subceldas visuales. Estas celdas o hexágonos intermedios representan las diferencias entre los vectores del codebook de los hexágonos circundantes. Como resultado, cada hexágono principal queda rodeado por nuevos hexágonos secundarios que representan la distancia entre vectores prototipo de sus celdas adyacentes<sup>(6)</sup>.

En estas nuevas celdas tendremos representados escalares que surgen de los cálculos de distancia. Las celdas de los bordes tendrán menos subceldas. El color de los hexágonos principales o "centrales" en la opción Mapa se obtiene promediando las distancias entre los hexágonos intermedios, y se representa de manera coherente en ambas visualizaciones.

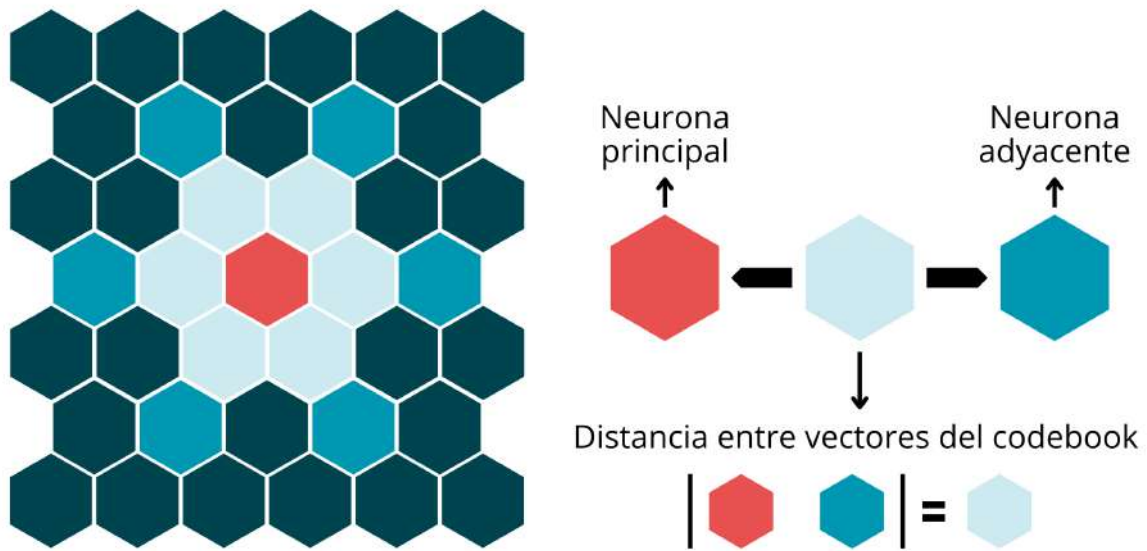


Figura 17: Explicación de hexágonos de UMat



Figura 18: Promedio de distancias

A partir de la figura 19, podemos observar que la opción Umat+ incluye los hexágonos adicionales de tonalidades claras, los cuales representan las distancias entre las celdas principales, es decir, la distancia entre los vectores del codebook asociados a esas celdas. En contraste, la opción Mapa únicamente muestra los hexágonos correspondientes a las celdas principales, y omite la visualización de las distancias entre ellas.

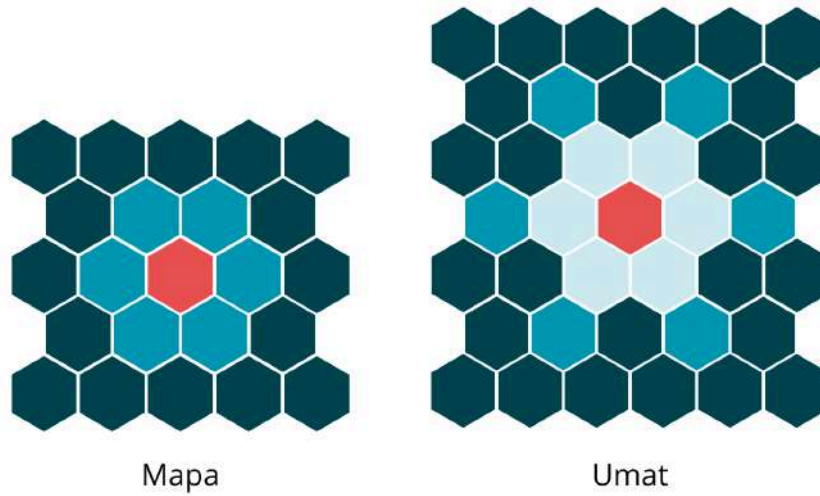


Figura 19: Comparativa entre el mapa y la UMat

El color asignado a cada hexágono se determina aplicando un gradiente de color en el que el valor máximo de las distancias se sitúa en un extremo y el mínimo en el otro. Según la posición del valor de cada hexágono dentro de este rango, se le asigna un color correspondiente. A la derecha de ambas grillas, se muestran los colores utilizados junto con los valores numéricos que representan, facilitando la interpretación visual de las distancias.

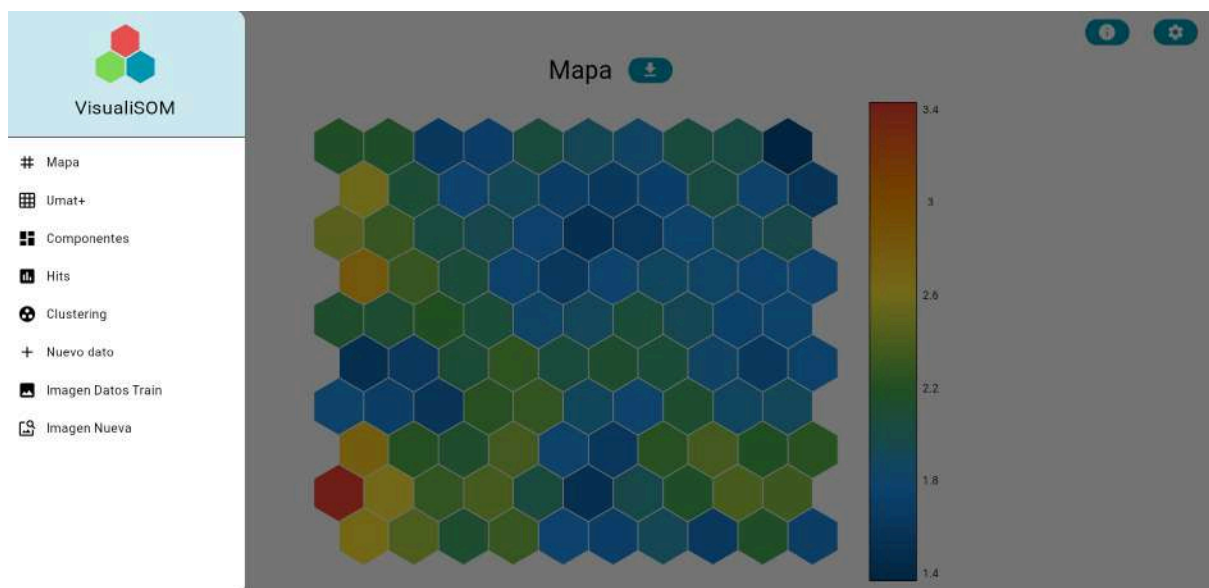


Figura 20: Pantalla de visualización de mapa y menú desplegable



En el sistema, cada hexágono principal es interactivo, permitiendo al usuario hacer clic sobre él para acceder a información detallada. Al seleccionar un hexágono, se abre una ventana emergente que presenta el vector del codebook asociado a la neurona, el valor de la Udist (que representa el promedio de las distancias con las celdas circundantes) y el número identificador de la BMU, asignado a cada neurona para facilitar su identificación y análisis dentro de la grilla.

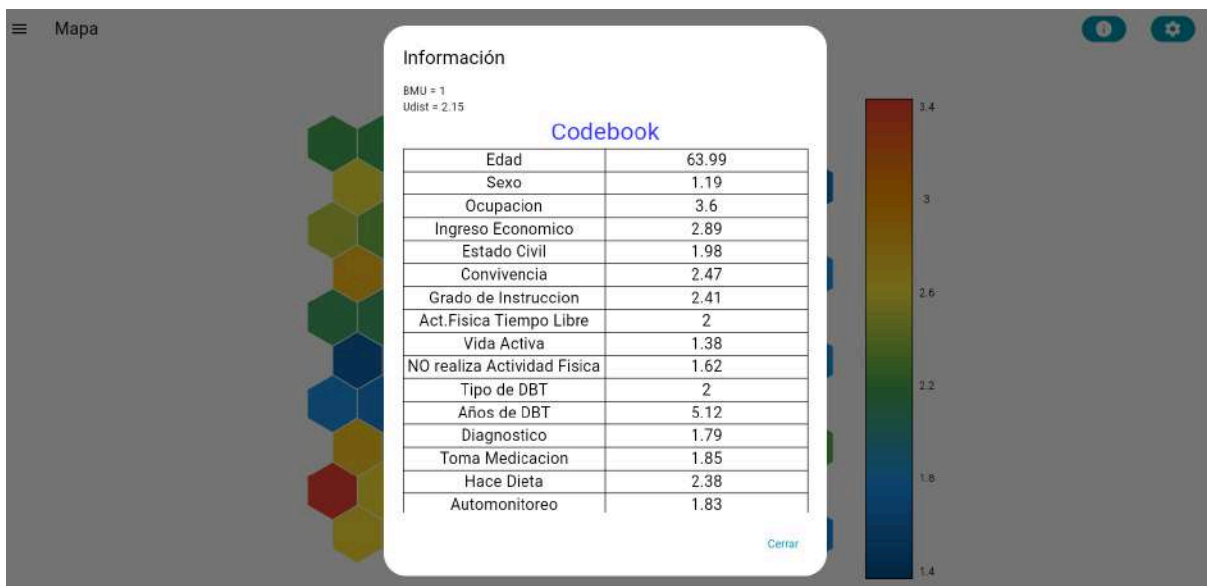


Figura 21: Información de cada neurona del mapa

En la opción Umat, los hexágonos secundarios, que representan las distancias entre las celdas principales, no son interactivos, ya que no están asociados a ningún vector del codebook ni pueden actuar como BMU de los datos.

Ambas opciones cuentan con un botón de descarga que permite guardar una imagen de la grilla. Esta imagen incluye el gradiente de colores utilizado en la visualización y el título correspondiente, facilitando así la obtención de una representación visual completa de la grilla para su análisis o documentación.

## 8.4 Componentes

Los mapas de componentes en los SOM visualizan cómo se distribuyen los valores de las variables de los datos a lo largo de la grilla. Cada mapa bidimensional muestra un componente específico, donde cada hexágono refleja el valor de una variable dentro del vector prototipo para la celda correspondiente<sup>(14)</sup>. Para generar estos mapas, se extraen los valores de las variables de cada vector del codebook, con cada componente del vector representando una variable particular de los datos.

En el sistema se permite visualizar los mapas de componentes de manera interactiva. Con un botón, el usuario puede seleccionar las variables específicas para las cuales desea generar el mapa de componentes. Esto facilita la personalización del análisis, permitiendo al usuario explorar cómo se distribuye cada variable a lo largo de la grilla del SOM mediante una representación visual.

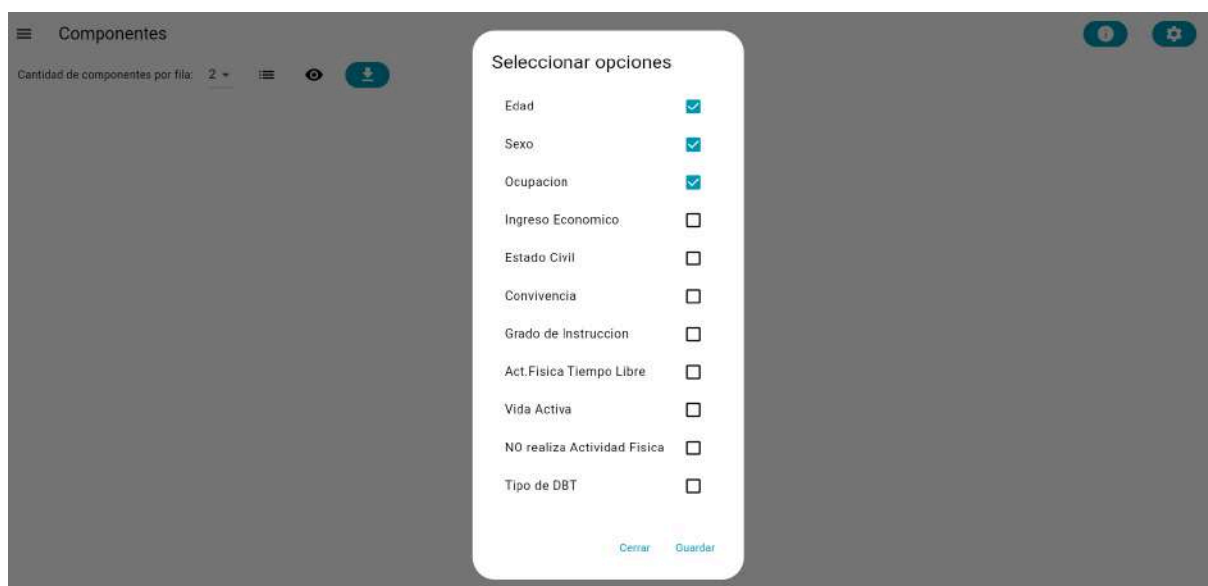


Figura 22: Selección de componentes

Además, el sistema ofrece opciones de personalización para la visualización de los mapas de componentes. Los usuarios pueden ajustar la cantidad de mapas que desean ver por fila, adaptando la disposición a sus necesidades, permitiendo, de

esta manera, un análisis visual multivariable. Asimismo, tienen la opción de elegir si desean mostrar o no el gradiente.

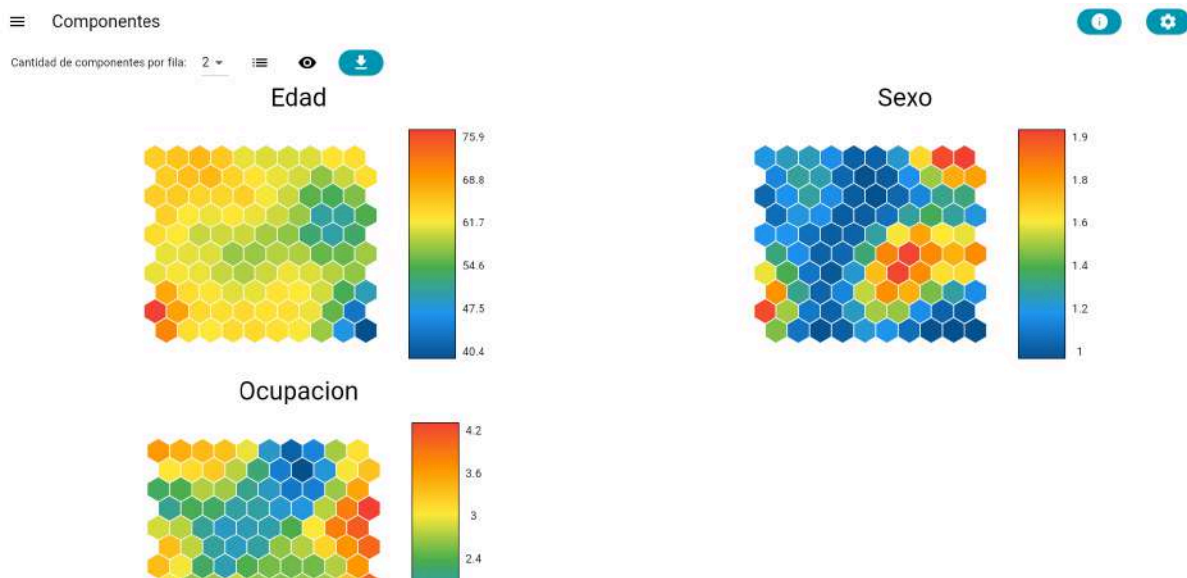


Figura 23: Pestaña de mapa de componentes

Al hacer clic en un hexágono del mapa, se despliega la misma información que en las pestañas anteriores: el vector del codebook asociado a la neurona, el valor de la Udist, y el número identificador de la BMU. Además, en el vector del codebook, se resalta la componente específica que está vinculada al mapa en el que se hizo clic.

El botón de descarga permite guardar una imagen de los mapas que se están mostrando en pantalla, conservando el formato y la disposición actual.

## 8.5 Etiquetado (*hits*)

En un SOM, cada dato se asigna a la neurona cuyo vector del codebook asociado sea el más parecido al dato. Este proceso asocia cada dato de entrada con una Best Matching Unit (BMU), que es la neurona que contiene el prototipo más cercano, en términos de similitud<sup>(6)</sup>. Así, una neurona puede contener desde ningún dato hasta múltiples datos que hayan sido asignados a ella.

Es importante señalar que los datos de entrada están etiquetados, es decir, poseen una o más etiquetas que los clasifican según distintas características. Por ejemplo, si cada dato representa a una persona, una etiqueta podría identificar su rango etario ("niño", "adolescente" o "adulto"), mientras que otra podría describir su estado de salud ("sano" o "enfermo"). Esta flexibilidad permite organizar los datos en múltiples categorías de etiquetas, lo que facilita su análisis y segmentación.

La funcionalidad "Hits" en nuestro sistema proporciona una visualización interactiva mediante una grilla de hexágonos, la cual refleja la distribución de los datos etiquetados entre las celdas. A la izquierda de la interfaz, un menú desplegable permite seleccionar la categoría de etiquetas que se desea examinar, cómo "Rango etario" o "Estado de salud". Debajo de este menú, se presentan las diferentes etiquetas dentro de la categoría elegida, cada una asociada a un color único para facilitar su identificación.

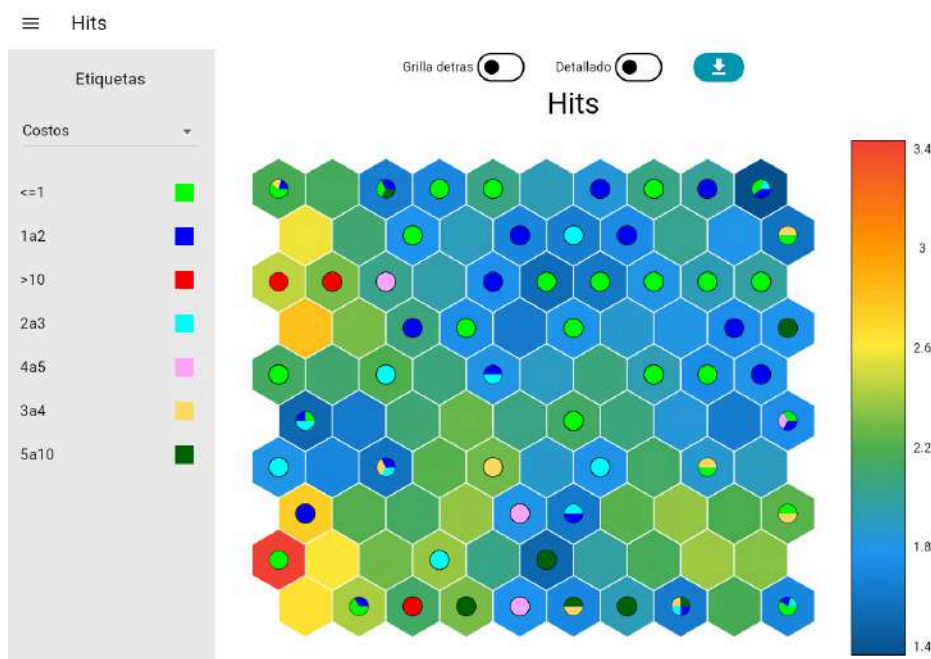


Figura 24: Pestaña de hits

Cada hexágono de la grilla puede contener un círculo o un gráfico de torta, los cuales ilustran la proporción de datos asociados a cada etiqueta dentro de esa

neurona. Por ejemplo, al seleccionar la categoría "Rango etario", podríamos observar que en un hexágono específico hay 3 datos etiquetados como "Niño" y 5 datos etiquetados como "Adulto", visualizados mediante un gráfico de torta que segmenta estas proporciones.

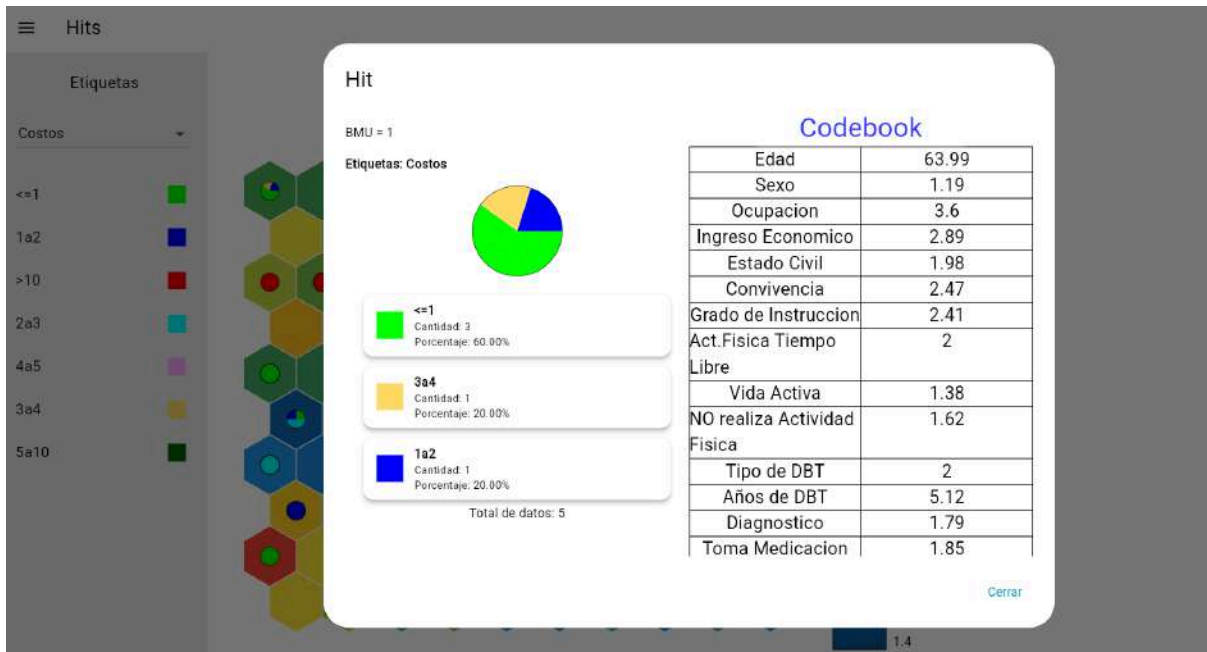


Figura 25: Información de hits

Al mover el cursor sobre un hexágono, se despliega información detallada que muestra las etiquetas de los datos asignados a esa neurona. Si se hace clic en el hexágono, se presenta un gráfico de torta que ilustra la distribución de los datos según sus etiquetas, mostrando tanto la cantidad de datos con cada etiqueta como el porcentaje que representan dentro de esa neurona. Adicionalmente, se visualiza el vector del codebook correspondiente.

El sistema también permite personalizar la visualización de la grilla. Es posible optar por mostrar u ocultar la grilla subyacente, que es la que corresponde a la opción "Mapa". Asimismo, existe la opción de mostrar los gráficos de torta en cada hexágono para visualizar las etiquetas o, alternativamente, mostrar sólo el color de la etiqueta predominante.

## 8.6 Agrupamiento (*clustering*)

Un algoritmo de *clustering* tiene como objetivo agrupar los objetos según su similitud, de forma que los objetos que hay dentro de un grupo (*cluster*) sean más similares que aquellos que caen en grupos distintos<sup>(15)</sup>.

Dentro del SOM, esto consiste en aplicar un algoritmo sobre el codebook de todas las celdas. De esta manera cada vector prototipo pertenece a un *cluster*, y por lo tanto podemos “pintar” la celda que lo contiene de un color asignado. Así veremos el mapa de las celdas agrupadas.

Dentro de la funcionalidad del sistema, se tiene una pantalla “*Clustering*” que contiene una caja de texto en donde se puede colocar la cantidad de *clusters* que se desean generar. Posteriormente, luego de oprimir el botón “Aceptar” se aplicará el algoritmo de *clustering Kmeans* sobre el codebook del mapa previamente entrenado. Este algoritmo tiene varios métodos de inicialización. Entre ellos está el *random* (aleatorio) en donde se eligen  $n$  (cantidad de *clusters*) filas al azar a partir de los datos para los centroides iniciales. Por otro lado, se tiene el *kmeans++* que selecciona los centroides iniciales del grupo mediante un muestreo basado en una distribución de probabilidad empírica de la contribución de los puntos a la inercia general. Esta técnica fue la elegida ya que acelera la convergencia<sup>(16)</sup>. El algoritmo implementado es “*greedy k-means++*”. Se diferencia del *k-means++* tradicional en que realiza varios ensayos en cada paso de muestreo y elige el mejor centroide entre ellos<sup>(17)</sup>.

Así obteniendo la grilla hexagonal interactiva con cada celda “pintada” según su *clúster* asociado. Esta grilla cuenta con las mismas funcionalidades que las demás. Al hacer clic en un hexágono del mapa, se despliega la misma información que en las pestañas anteriores: el vector del codebook asociado a la neurona, el valor de la  $U_{dist}$ , y el número identificador de la BMU. Asimismo se tiene el mismo botón para poder descargar la imagen del mapa generado.

≡ Clustering



Figura 26: Pestaña de clustering con mapa segmentado

## 8.7 Nuevos datos

Como se mencionó anteriormente, uno de los puntos más importantes de los SOM es la clasificación de nuevos datos. Esto implica introducir datos nuevos que tengan las mismas dimensiones y variables (en el mismo orden) que los utilizados en el entrenamiento para clasificarlos dentro del SOM. De este modo, es posible analizar y extraer conclusiones.

Para poder lograr esto, el sistema posee una solapa desplegable en la cual se puede cargar un archivo csv con esos datos. Similar a la pantalla principal, se permite seleccionar las columnas que serán usadas y las que funcionan como etiquetas. Luego de este procedimiento, se procede a obtener la neurona asociada a cada dato. Esto se realiza encontrando la neurona que más se acerca en cuanto a similitud al dato. Al igual que en el entrenamiento, esto se realiza al obtener la menor distancia euclidiana entre el dato y los vectores prototipo del mapa entrenado.



☰ Nuevo dato

Seleccionar Archivo CSV Features Etiquetas Datos

Dato	Costos	Motivos	Edad	Sexo	Ocupacion	Ingreso Economico	Estado Civil
1	>10	PD	48	1	1	1	2
2	5a10	PD	63	1	2	3	3
3	<=1	IPIEL	44	1	5	3	3
4	1a2	X	58	2	4	3	2

Devolver BMU datos

Figura 27: Selección de archivo en pestaña de nuevos datos

Una vez identificadas las celdas correspondientes, los nuevos datos se proyectan en la grilla para visualizarlos gráficamente. Las celdas del SOM se muestran con los datos marcados en su BMU, y a la izquierda se lista el color asignado a cada dato para facilitar su identificación.

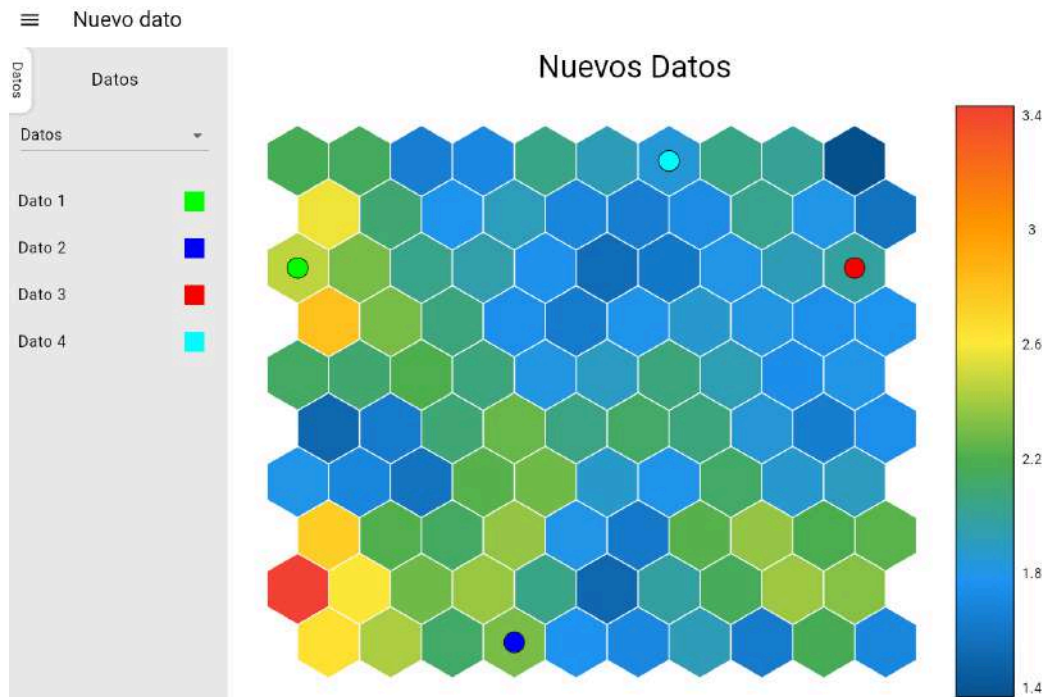


Figura 28: Mapa con nuevos datos clasificados





Al hacer clic en un hexágono del mapa, se despliega la misma información que en las pestañas anteriores: el vector del codebook asociado a la neurona, el valor de la Udist, y el número identificador de la BMU.

## 8.8 Segmentación y coloreado de imágenes

La segmentación consiste en dividir una imagen digital en varias regiones (grupos de píxeles) denominadas segmentos. Más concretamente, la segmentación es un proceso de clasificación por píxel que asigna una categoría a cada píxel de la imagen analizada.

Cada problema especializado le otorga un significado propio a las categorías que se usan en la clasificación de los píxeles. Uno de los casos más elementales de segmentación es la umbralización, un tipo particular de segmentación por color con solo dos categorías: claro y oscuro. Cada píxel se clasifica como claro u oscuro comparando su intensidad con una intensidad de referencia dada denominado umbral. Por otro lado, uno de los casos más sofisticados es la segmentación semántica que clasifica objetos diversos.

El objetivo de la segmentación es localizar regiones con significado. La segmentación se usa tanto para localizar objetos como para encontrar sus bordes dentro de una imagen. El resultado de la segmentación de una imagen es un conjunto de segmentos que cubren toda la imagen sin superponerse. Se puede representar como una imagen de etiquetas (una etiqueta para cada píxel) o como un conjunto de contornos.

Los SOM son una buena alternativa a los métodos tradicionales de agrupamiento cuando se cuenta con suficiente cantidad de información y existe una discriminación de características razonable, son una gran herramienta para entender qué tan agrupables son las clases, qué tan diferenciales son de otros grupos y a qué grupos se parecen más que otros.

A continuación, explicaremos cómo abordamos el tratamiento de las imágenes en nuestro proyecto.

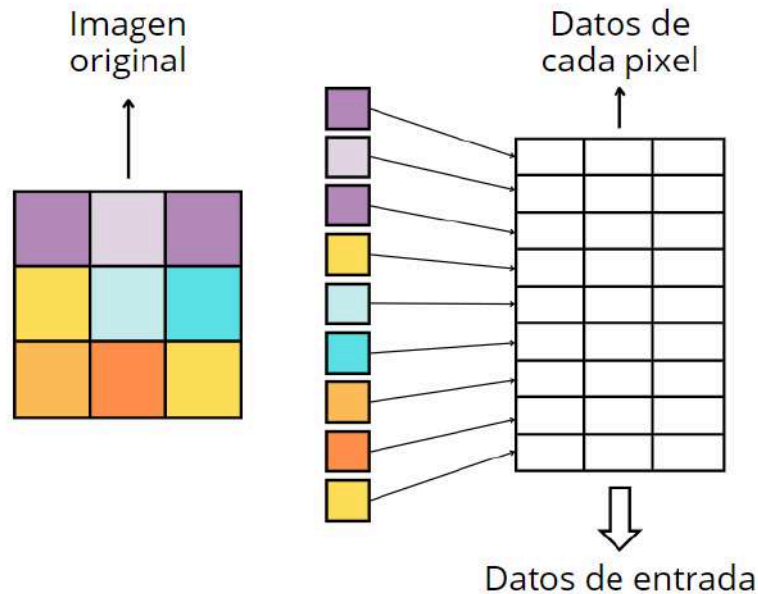


Figura 29: Tratamiento de imágenes

En nuestro sistema, los datos de entrada provienen directamente de los píxeles de una imagen, lo que significa que cada dato representa la información asociada a un único píxel de dicha imagen. Estos datos incluyen valores de intensidad, color, descriptores de textura, descriptores de rugosidad, u otras características que deseemos utilizar como entrada para el modelo.

De esta manera, tratamos a la imagen como un conjunto de datos, donde cada píxel aporta información relevante para el entrenamiento del Mapa Autoorganizado (SOM). Esta representación no solo permite el análisis detallado de cada píxel en función de sus características, sino que también facilita la posterior reconstrucción y visualización de la imagen, una vez completado el entrenamiento.

La extracción de características (descriptores) de imágenes no fue incluida en nuestro trabajo, ya que se consideró este un proceso fuera del alcance de nuestros objetivos. Por esta razón, para nuestras pruebas utilizamos datos de entrada con

características previamente extraídas de imágenes, los cuales nos fueron proporcionados por los directores.

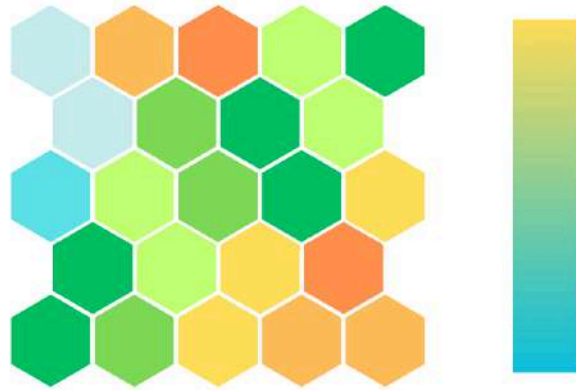


Figura 30: Entrenamiento con datos de imagen

Como explicamos en apartados anteriores, el resultado del proceso de entrenamiento es un conjunto de vectores de referencia, conocido como *codebook*. Cada neurona en el mapa tiene un vector que captura las características más representativas de los datos que se le han asignado. Este codebook es fundamental para comprender cómo los datos se distribuyen en el mapa y cómo se agrupan en torno a las celdas más representativas (BMUs).

Una vez finalizado el entrenamiento, cada dato de entrada queda asociado a una neurona específica del SOM, a su BMU.

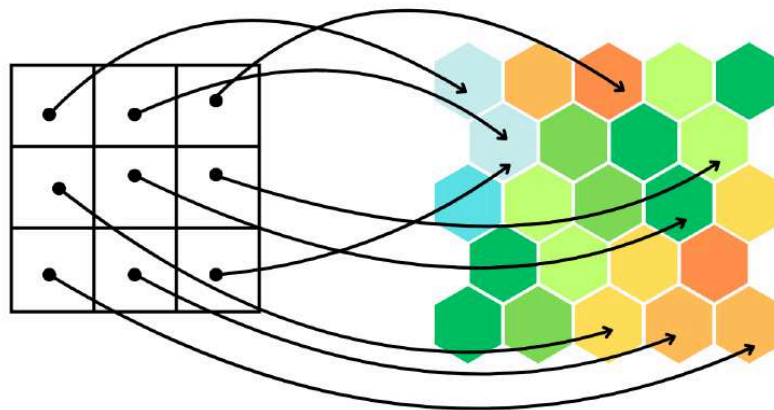


Figura 31: Asociación de dato a neurona del mapa

En el contexto de nuestros datos, cada píxel es representado por un vector de variables (características, descriptores) derivado de sus propiedades. En el entrenamiento, cada uno de estos vectores de características es asignado a la neurona cuyo vector de referencia (en el codebook) es el más cercano en términos de similitud, convirtiéndose en su BMU.

Por lo tanto, cada píxel queda asociado a una BMU específica, lo que significa que la neurona que representa esa BMU es la que mejor "captura" las características del píxel en cuestión. Este vínculo entre los píxeles y sus respectivas BMUs nos permite estructurar los datos de la imagen dentro del mapa autoorganizado, facilitando el análisis de patrones y relaciones dentro de la imagen.

A partir de esta asignación (pixel-bmu), hemos implementado dos procedimientos de coloreado de imágenes, basados en el *clustering* de los vectores del codebook y en un esquema de coloración continua.

- *Clustering*

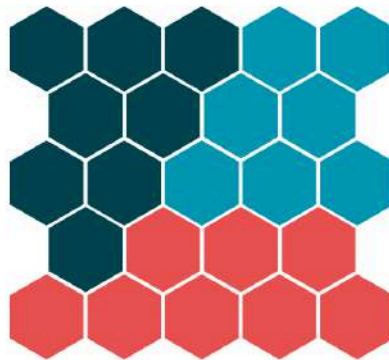


Figura 32: Mapa con clustering aplicado

Si aplicamos un algoritmo de *clustering* sobre los vectores del *codebook*, podemos agrupar las celdas del SOM en *clusters*. Este proceso fue detallado previamente en el apartado de *clustering* del sistema, donde explicamos cómo se agrupan las celdas en función de su similitud.

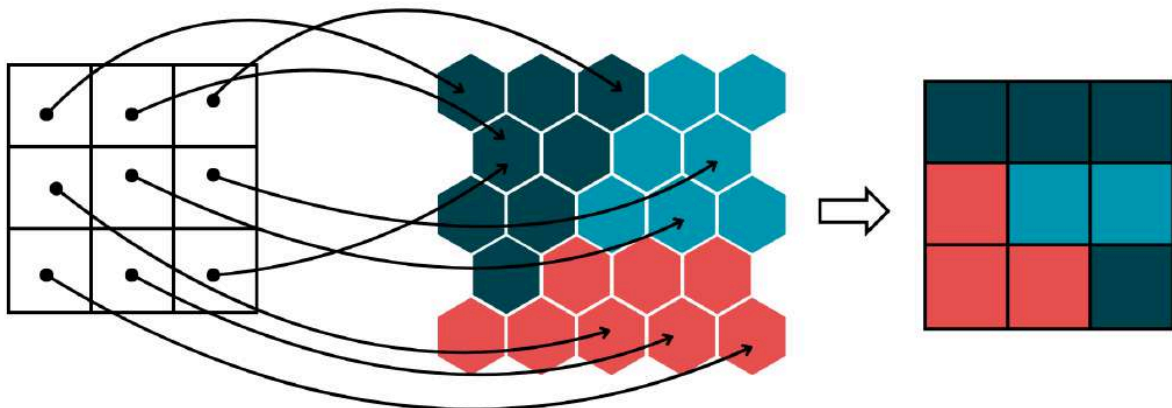


Figura 33: Asociación de píxel a neurona segmentada

Como cada neurona tiene un *cluster* asignado, y cada píxel de la imagen está vinculado a una neurona a través de su BMU, esto implica que cada píxel también pertenece a un *cluster*. Al asignar un color distintivo a cada *cluster*, podemos reconstruir la imagen original píxel por píxel, coloreando cada uno con el color asociado al *cluster* de su BMU correspondiente. De esta manera, la imagen resultante estará segmentada según los *clusters* definidos en el SOM, lo que nos permite visualizar las diferentes regiones de la imagen de acuerdo a las características compartidas por los píxeles dentro de cada *clúster*.

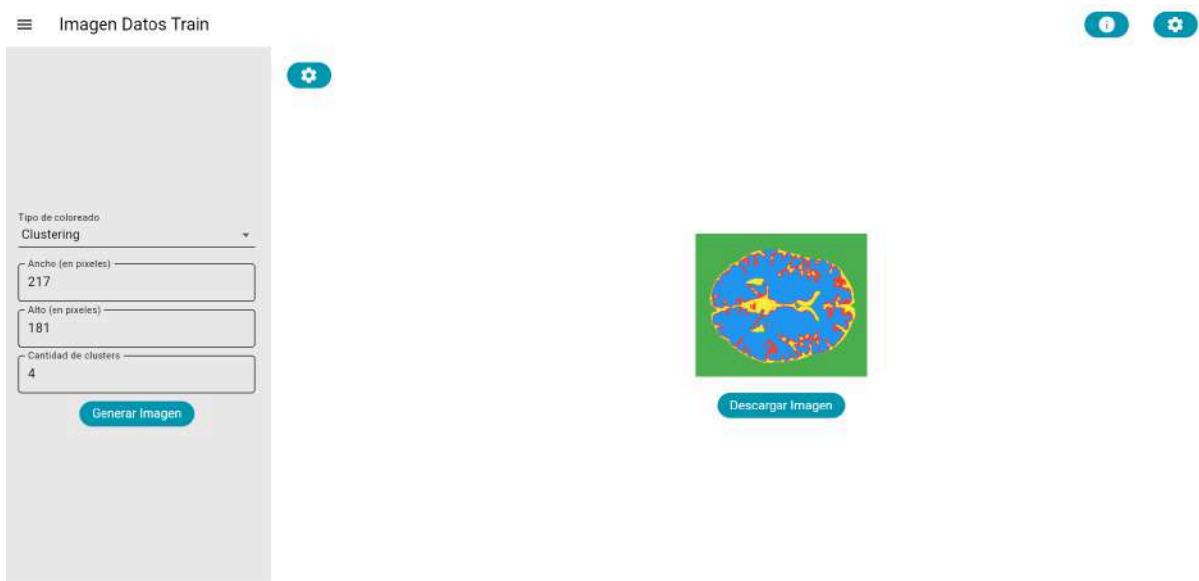


Figura 34: Pestaña con imagen segmentada

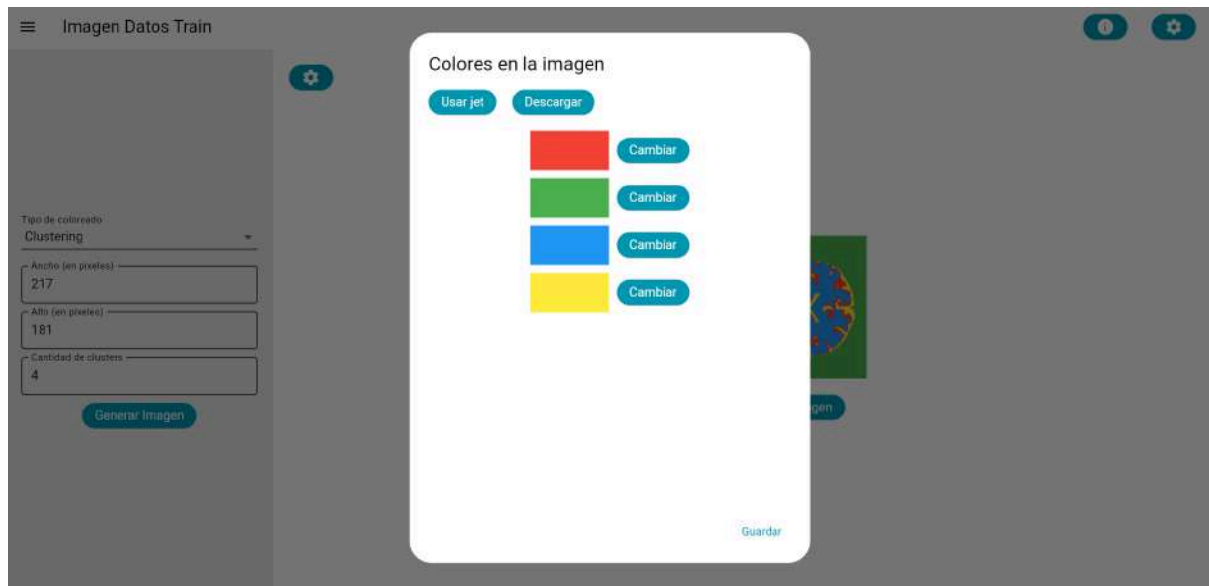


Figura 35: Configuración de colores

Dentro de esta opción, se encuentra la posibilidad de cambiar los colores de los *clusters*. A su vez, el usuario puede descargar en formato txt la matriz de colores de la imagen generada para su posterior análisis.

- Coloreado continuo

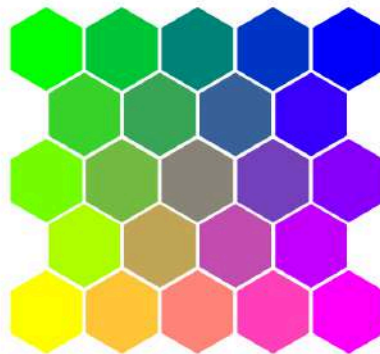


Figura 36: Diagrama de grilla con gama continua de colores

En lugar de utilizar un enfoque basado en *clústeres*, el segundo procedimiento de coloreado utiliza una gama continua de colores para representar la similitud entre las celdas del *codebook*. En este esquema, las celdas cercanas en el espacio del

SOM son asignadas a colores similares, mientras que las celdas más distantes reciben colores más diferenciados.

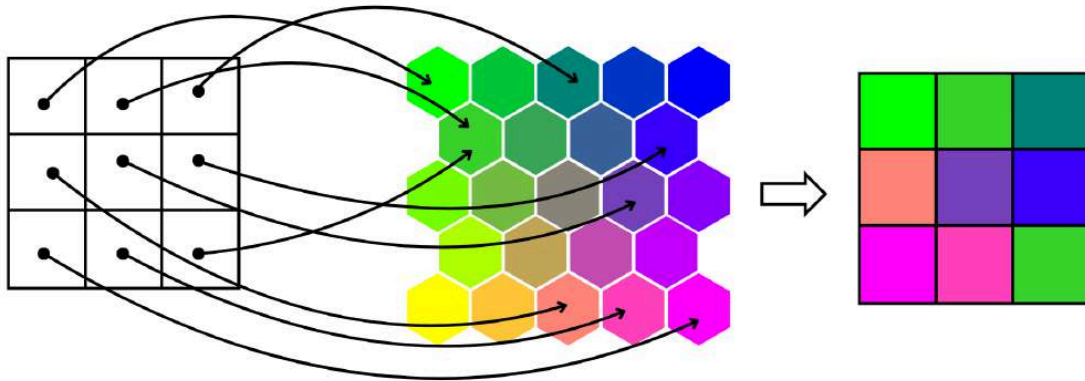


Figura 37: Asociación de pixel a grilla de gama continua

De manera similar al proceso que utiliza *clustering*, al identificar la neurona en la que se encuentra cada píxel y el color asociado a esa neurona, podemos reconstruir la imagen original utilizando el color correspondiente a cada neurona. Este proceso implica que los píxeles que comparten características similares serán representados con colores similares, mientras que los píxeles que difieren en sus propiedades se visualizarán con colores distintos.

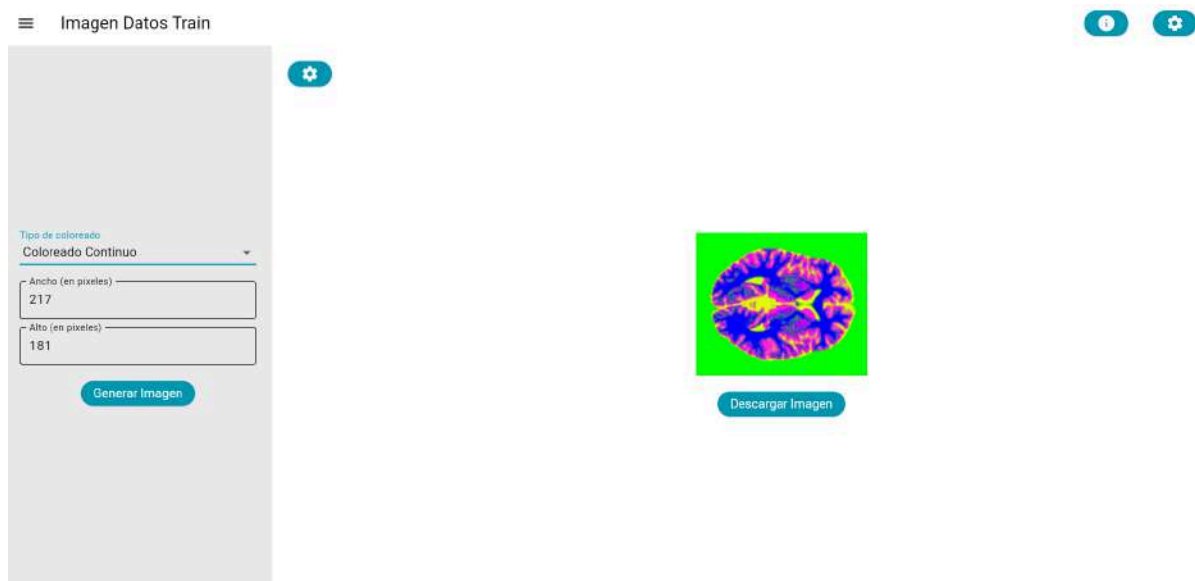


Figura 38: Pestaña con imagen segmentada

Como resultado, la imagen reconstruida no solo presenta una segmentación visual clara, sino que también destaca las áreas donde los píxeles tienen propiedades similares. Esto facilita la identificación de regiones homogéneas y de transiciones entre diferentes áreas de la imagen.

Como función adicional, está la posibilidad de cargar un archivo nuevo para la generación y segmentación de imágenes. La cantidad de datos del archivo debe coincidir con el ancho y alto de la imagen. A su vez, la cantidad de columnas del archivo debe coincidir con las del archivo utilizado durante el entrenamiento. Al igual que en la función anterior, se tiene la opción de selección de los dos procedimientos de coloreado de imágenes, clustering y coloreado continuo.

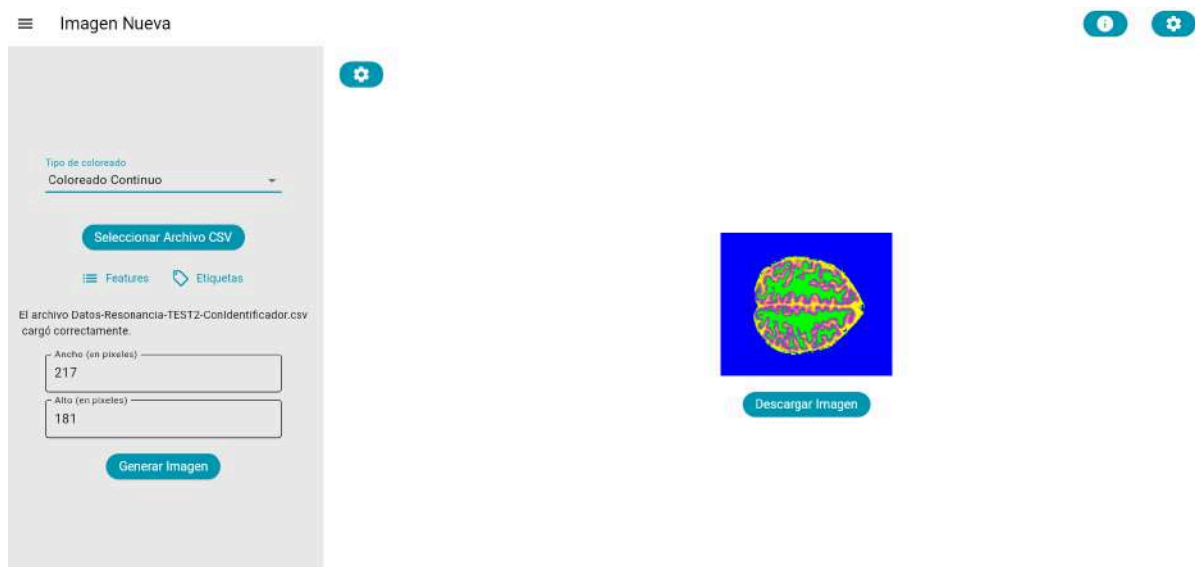


Figura 39: Pestaña de segmentación con archivo nuevo

A diferencia de la otra funcionalidad, se debe detectar a qué neurona pertenece cada dato dentro del mapa entrenado previo a asociar el color. Para poder realizar esto, se debe encontrar la neurona que más se acerca en cuanto a similitud al dato. Al igual que la funcionalidad de “Nuevos Datos”, esto se realiza al obtener la menor distancia euclidiana entre el dato y los vectores prototipo del mapa entrenado. Luego de la obtención de la neurona, se obtiene el color realizando el clustering correspondiente, o de la grilla con gama continua de colores.





## 9. Testing

### 9.1 API

Para realizar las pruebas de la API se optó por hacer uso de la aplicación Postman. Se eligió ya que es una popular herramienta utilizada para probar APIs, permitiendo enviar peticiones a servicios web y ver respuestas<sup>(18)</sup>. A su vez, es una aplicación simple y los integrantes ya contaban con experiencia en el uso de esta tecnología debido a que fue utilizada en varias materias a lo largo de la carrera.

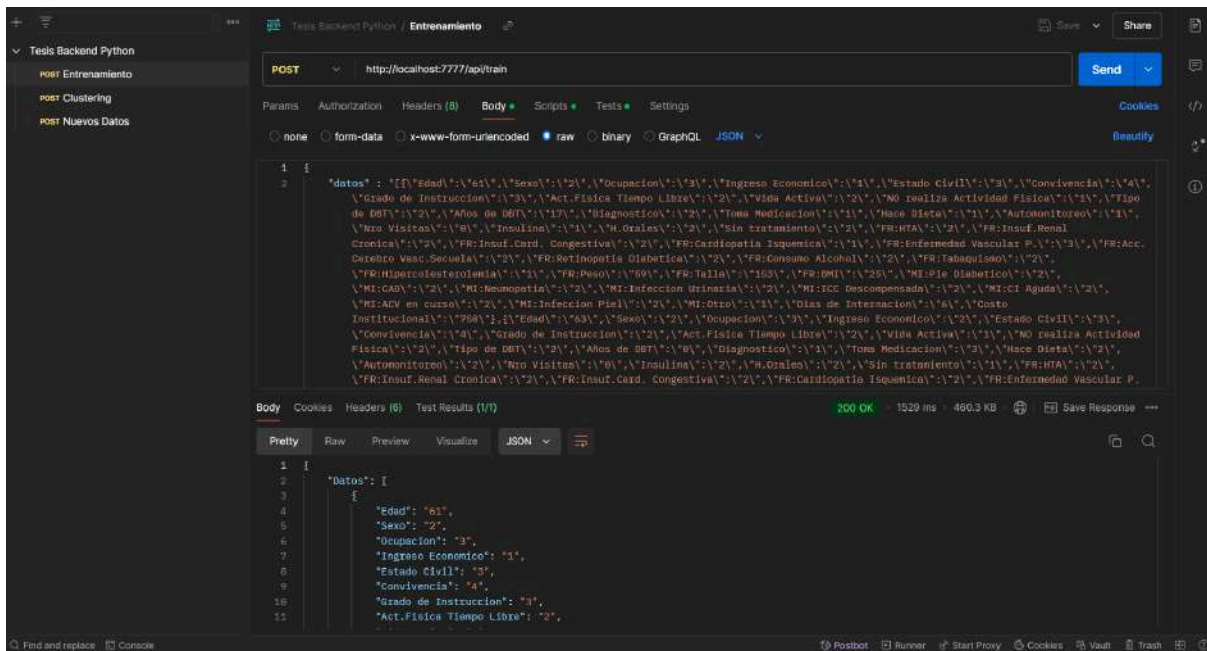


Figura 40: Testing API en Postman

### 9.2 Aplicación Web

Como primera medida, en cuanto a las pruebas realizadas dentro de la aplicación web se utilizaron las herramientas de desarrollo existentes dentro de los navegadores como puede ser Chrome. Además, se utilizaron las DevTools de Dart que te permiten analizar en detalle el estado de la aplicación en tiempo real



permitiendo analizar variables, parámetros y realizar *breakpoints* específicos para analizar partes del código.

Dentro de las pruebas más importantes, utilizamos el paquete *integration\_test* de Flutter para llevar a cabo pruebas de integración. Las pruebas de integración permiten verificar cómo interactúan entre sí los distintos módulos y componentes de la aplicación en un entorno similar al de producción. A diferencia de las pruebas unitarias, que evalúan partes aisladas del código, las pruebas de integración simulan el comportamiento del usuario final y validan la funcionalidad completa del sistema.

Además, utilizamos *ChromeDriver*, una herramienta esencial para la automatización de pruebas en entornos basados en navegadores web. ChromeDriver actúa como un puente entre nuestras pruebas automatizadas y el navegador Chrome, permitiendo simular acciones del usuario como hacer clic en botones, desplazarse a través de diferentes pantallas y manipular diversos elementos de la interfaz.

Dentro del código de testing, se simulan diversas acciones para validar el comportamiento de la aplicación. En primer lugar, se realiza la carga inicial de la aplicación mediante la ejecución del widget principal MyApp. Posteriormente, se utiliza la función *pumpAndSettle* para esperar brevemente y asegurarse de que todos los widgets se hayan renderizado correctamente antes de proceder con las verificaciones. Para validar que la página principal (HomePage) se ha cargado correctamente, se emplea *find.byType* y se verifica que el widget haya sido encontrado sin problemas.

En la siguiente fase, se simula la carga de un archivo que contiene el resultado de un entrenamiento. Dado que en esta etapa solo se está probando el lado de Flutter, no es necesario realizar una llamada a la API para entrenar los datos. Una vez que el archivo se ha cargado y los datos han sido procesados, se navega hacia la pantalla de grillas (GrillasPage), donde estos datos son esenciales para su correcto funcionamiento.



Se simulan interacciones con la grilla de hexágonos, como clics en el botón de información y el botón de configuración, mediante la función *tester.tapAt*. Estas interacciones aseguran que los *popup* puedan abrirse y cerrarse correctamente. Asimismo, se validan las interacciones con el menú lateral desplegable, y dentro de cada sección, se simulan acciones como la selección de opciones o la modificación de configuraciones utilizando *CheckboxListTile* y *Switch*.

En la sección de visualización de "Hits", se simula la modificación de las etiquetas mostradas en los hexágonos de la grilla, lo que permite evaluar cómo responde la aplicación a estos cambios.

Las pruebas aseguran que la aplicación responda correctamente a la navegación entre pantallas y menús, y que todas las acciones del usuario, como clics en hexágonos, ajustes de configuraciones y apertura de diálogos, funcionen como se espera.

```
/// TEST EMPEZADO: 28-9-2024 18:17:41 ///  
/// TEST 1: Cargo HomePage correctamente ///  
/// TEST 2: Simulo carga de archivo ///  
/// TEST 3: Navegación a la pantalla de grillas ///  
/// TEST 4: Apretar boton de un hexagono ///  
/// TEST 5: Salir del dialog ///  
/// TEST 6: Apretar boton dialog Informacion ///  
/// TEST 7: Salir del dialog ///  
/// TEST 6: Apretar boton dialog Configuracion ///  
/// TEST 7: Salir del dialog ///  
/// TEST 8: Abrir menu lateral ///  
/// TEST 9: Abrir opcion Umat+ ///  
/// TEST 10: Apretar boton de un hexagono ///  
/// TEST 11: Salir del dialog ///  
/// TEST 12: Abrir menu lateral ///  
/// TEST 13: Abrir opcion Componentes ///  
/// TEST 14: Abrir lista de opciones ///  
/// TEST 15: Seleccionar checks ///  
/// TEST 16: Apretar guardar ///  
/// TEST 17: Abrir menu lateral ///  
/// TEST 18: Abrir opcion Hits ///  
/// TEST 19: Apretar hexagono con etiquetas ///  
/// TEST 20: Salir del dialog ///  
/// TEST 21: Switch 1 ///  
/// TEST 22: Switch 2 ///  
/// TEST 23: Abrir opciones de etiquetas ///  
/// TEST 24: Cambiar a Motivos ///  
/// TEST 25: Abrir menu lateral ///  
/// TEST 25: Abrir opcion Clustering ///  
/// TEST 26: Abrir menu lateral ///  
/// TEST 27: Abrir opcion Nuevo dato ///  
/// TEST 28: Abrir menu lateral ///  
/// TEST 29: Abrir opcion Imagen Datos Train ///  
/// TEST 30: Abrir menu lateral ///  
/// TEST 31: Abrir opcion Imagen Nueva ///  
03:51 +1: (tearDownAll)  
03:51 +2: All tests passed!
```

Figura 41: Resultado test de flutter



Incorporamos diversos mensajes de impresión en el test para rastrear las acciones ejecutadas durante su transcurso. Al finalizar el test, es posible visualizar en la terminal todos los mensajes generados, siempre y cuando no haya ocurrido ningún error. Además, al final del proceso, se proporciona un resumen que confirma que todas las pruebas se han completado con éxito, indicando que "todos los tests pasaron" (all tests passed).



## 10. Entregables

### 10.1 Software

El software entregado corresponde a las dos aplicaciones, el *backend* en Python (framework Flask) y el *frontend* en Dart (framework Flutter).

Ambas aplicaciones se encuentran desplegadas en plataformas online (Vercel y Render) y además se proporciona el código fuente para su mantenimiento a futuro y posibles mejoras o nuevas funcionalidades que se quieran implementar a futuro.

### 10.2 Manual de uso

Junto con el software y el código fuente, se entrega un manual de uso detallado que explica cómo utilizar el sistema de principio a fin. A su vez, se anexa al final del informe para su visualización.

Este manual incluye una descripción completa de cada uno de los módulos, comenzando con el proceso de carga de datos, seguido por la configuración de los parámetros necesarios para la correcta ejecución del sistema. Se detallan todas las opciones de personalización disponibles, permitiendo al usuario ajustar el sistema de acuerdo con sus necesidades específicas.

Además, se explica el funcionamiento de la interfaz para la visualización de la grilla del SOM y cómo navegar entre las diferentes pantallas del menú. Cada pantalla y funcionalidad está descrita paso a paso, para que el usuario pueda utilizar todas las herramientas del sistema sin dificultad.

El manual también ofrece una guía detallada para ejecutar la API localmente en un entorno Python, brindando instrucciones claras sobre los requisitos previos, las dependencias necesarias y los pasos para configurar el entorno.



Aunque el manual de uso detalla el funcionamiento del sistema, incluyendo los diferentes módulos con sus funcionalidades y la configuración del servidor, como grupo nos comprometemos a brindar las capacitaciones necesarias en caso de que se requieran.

## 11. Benchmarking

En el mercado hay varias herramientas que permiten generar SOM. Estas herramientas varían en complejidad y características, desde bibliotecas de código abierto hasta software especializado de análisis de datos. Todas tienen algo en común, el hecho de no ser una aplicación específica que permita entrenar y visualizar los mapas. Algunas siendo solo librerías o paquetes, y otras siendo parte de una aplicación más grande que tiene otras funcionalidades presentando un gasto muy grande. A continuación presentamos las más importantes.

### *Herramientas de código abierto*

- *SOMPY*: Biblioteca Python para SOM que proporciona más configuraciones y capacidades de visualización<sup>(19)</sup>.
- *IntraSOM*
- *MiniSOM*: Una implementación ligera y fácil de usar de SOM en Python<sup>(20)</sup>.
- *R (Kohonen package)*: Una implementación popular de SOM en R. Ofrece varias funciones para entrenar SOM, visualizar resultados y evaluar modelos<sup>(21)</sup>.
- *Sklearn-SOM*: Es una implementación minimalista de mapas autoorganizados que utiliza una topología rectangular, lo que la diferencia de otras bibliotecas que usan redes hexagonales. Esta biblioteca permite entrenar el modelo, predecir grupos de datos y visualizar los resultados en una topología sencilla<sup>(22)</sup>.

### *Software de análisis de datos*

- *Orange*: Orange es una plataforma de minería de datos y aprendizaje automático que incluye un widget específico para SOM. Ofrece una interfaz gráfica intuitiva para construir, entrenar y visualizar SOM<sup>(23)</sup>.



- Knime : Es otra plataforma de análisis de datos que permite integrar algoritmos de SOM a través de extensiones y nodos personalizables<sup>(24)</sup>.
- Weka: Es un software de minería de datos que también incluye una implementación de SOM dentro de su conjunto de herramientas de clustering<sup>(25)</sup>.

### *Software Comercial*

- *MATLAB*: Ofrece una implementación robusta de SOM a través de su Toolbox de Machine Learning y Neural Network. Proporciona una variedad de herramientas para el análisis visual de SOM<sup>(26)</sup>.
- *Viscovery SOMine*: Una herramienta de análisis de datos y minería de datos que utiliza SOM para la segmentación y análisis de patrones en conjuntos de datos<sup>(27)</sup>.

En comparación con las herramientas disponibles en el mercado, nuestro sistema ofrece una solución más accesible y centrada en el usuario. A diferencia de las bibliotecas de código abierto, como SOMPY o MiniSOM, que requieren conocimientos de programación para ser utilizadas, nuestro sistema permite a cualquier usuario interactuar de manera sencilla con los datos, entrenar mapas autoorganizados y visualizar resultados sin necesidad de escribir código. Esto representa una ventaja significativa para usuarios con menos experiencia técnica que buscan una experiencia intuitiva y directa.

Plataformas amplias como Orange o Knime destacan por permitir la integración de SOM con otros algoritmos y funcionalidades avanzadas, algo que actualmente no ofrece nuestro sistema. Sin embargo, esta simplicidad puede ser una ventaja para quienes buscan una herramienta específica para SOM, sin las distracciones ni la curva de aprendizaje asociadas a plataformas multifuncionales.

Ninguna de las herramientas mencionadas incluye de forma integrada la funcionalidad de segmentación de imágenes. Este es un punto que diferencia a





nuestro sistema, ya que permite aplicar SOM directamente para segmentar y pseudo colorear imágenes de manera sencilla.

Tanto en plataformas como Orange o Knime, como en software comercial como MatLab, identificamos que nuestro sistema presenta una desventaja en términos de velocidad de procesamiento. Esto se debe a que muchas de estas herramientas provienen de grandes empresas con una amplia gama de recursos, en contraste con nuestras limitaciones. No obstante, cabe destacar que estas alternativas no son gratuitas y requieren una inversión considerable para su uso, lo que motivó en gran medida el desarrollo de este proyecto.



## 12. Memoria del proyecto

### 12.1 Nacimiento del proyecto

El problema a resolver fue presentado por los miembros del Laboratorio de Bioingeniería. La necesidad de estos surgía por la inexistencia de un entorno visual que permita el análisis de conjuntos de datos de alta linealidad.

Los 3 integrantes del proyecto hemos cursado juntos y aprobado la asignatura “Inteligencia Artificial” en el segundo cuatrimestre de 2022, la cual nos aportó las herramientas y conocimientos teóricos necesarios para afrontar un desafío como fue la ejecución de este proyecto.

Encabezan la dirección del proyecto dos docentes de nuestra unidad académica, Gustavo y Diego, que poseen amplia experiencia en IA y especialmente en SOMs, el tema central de nuestro proyecto final.

### 12.2 Cumplimiento de objetivos

A partir del desarrollo de este trabajo, se obtuvo un sistema que cumple con todos los requerimientos comprendidos en el alcance pactado. Se logró desarrollar un sistema para el análisis de datos de gran volumen y/o con alta dimensionalidad para su posterior detección de patrones, tendencias o para la toma de decisiones. A su vez, se logró desarrollar las funcionalidades necesarias para poder obtener visualizaciones con capacidades gráficas interactivas mediante una aplicación simple e intuitiva para el usuario. Se logró desarrollar todos los módulos funcionales para el cumplimiento de todos los requerimientos funcionales y no funcionales. Por lo tanto, el objetivo principal de este proyecto ha sido alcanzado con éxito.

A diferencia del objetivo principal, cuyo cumplimiento puede evaluarse fácilmente al verificar si se satisfacen los requerimientos establecidos, el objetivo secundario

resulta menos medible debido a la inexistencia de un criterio cuantificable. Con este proyecto, buscamos contribuir a la visibilidad futura del SOM como una herramienta recomendable. Aunque no podemos asegurar que hayamos alcanzado este objetivo por completo en este momento, tenemos la esperanza de que se cumpla en el futuro y estamos convencidos de haber realizado todos los esfuerzos necesarios para lograrlo.

### 12.3 Comparación planificación original y ejecutada



Figura 42: Comparativa de planificaciones

### 12.4 Análisis

La fase de análisis fue la primera que se realizó en el proyecto. Toda planificación que requiera resolver un problema implica en primera instancia aprender sobre éste para poder afrontarlo de la mejor manera.

En nuestro caso, la fase de análisis tuvo como primera medida reuniones con los demandantes del proyecto, quienes comunicaron el problema que habían detectado y la ausencia de herramientas que cumplieran con sus necesidades. Las primeras



reuniones fueron de carácter informal a modo informativo para nosotros, ya que incluso los demandantes tenían en claro que no era tarea sencilla desarrollar una solución al problema que planteaban.

Como grupo, reconocimos la difícil curva de aprendizaje necesaria para trabajar con inteligencia artificial, tanto por el complejo fundamento matemático (álgebra lineal) que está detrás como por el desconocimiento de los lenguajes de programación y librerías utilizadas para desarrollar. Es por esto, que antes de generar la documentación mostrada realizamos una prueba de factibilidad generando código que pueda resolver de forma casi manual una porción pequeña del problema. Al concluir que esta prueba fue exitosa, en mutuo acuerdo nos comprometimos a la realización de este proyecto que nos mantuvo ocupados durante gran parte del año. La fase de análisis consistió en reuniones presenciales con los demandantes del proyecto, se relevaron las necesidades del mismo, se definió su alcance y se generó la documentación presente en los primeros capítulos de este informe. Dentro del análisis se realizó la elicitación de requerimientos y evaluación de riesgos.

El proceso fue ameno y de la dificultad esperada, fuimos cautos y no subestimamos esta fase, sin embargo, a la mitad del proyecto, se realizaron dos etapas de análisis adicionales no planificadas debido a que los requerimientos del proyecto fueron dinámicos, mutando para generar un producto de mayor calidad. Más allá del alcance pactado, se decidió la incorporación de nuevas funcionalidades, como fue el caso del módulo de nuevos datos y el de segmentación de imágenes. Esto es solamente una reflexión ya que no fue algo que haya alterado las fases siguientes del proyecto, los cambios propuestos fueron siempre constructivos y no de una magnitud que implicara un consumo de tiempo excesivo, recurso bastante limitado durante todo el ciclo.

A lo largo del proyecto surgieron riesgos no previstos inicialmente, como la posibilidad de que la librería seleccionada (IntraSOM) dejará de estar mantenida. Sin embargo, esto no representaría un inconveniente significativo, ya que existen alternativas en el mercado que podrían integrarse al desarrollo actual con facilidad y



en poco tiempo. Además, emergió el riesgo de que el proyecto no se completara, algo que no consideramos en un principio debido al entusiasmo inicial. Estos desafíos representan una valiosa oportunidad de aprendizaje, permitiéndonos comprender la importancia de realizar una etapa de análisis más clara y precisa.

## 12.5 Diseño

A continuación, se detallan las decisiones relacionadas al diseño del proyecto:

Una de las decisiones más importantes luego de la fase de análisis fueron las tecnologías a usar, en el caso del servidor el uso de Python fue casi obligatorio y altamente recomendado debido a la gran cantidad de librerías y lo avanzado que se encuentran para implementaciones relacionadas a IA o proyectos de análisis de datos, incluso en el mercado actual.

Por otro lado, teniendo en cuenta las características del proyecto debimos tomar la decisión del frontend a utilizar. Se contemplaron varias alternativas, pero finalmente tomamos la decisión de usar Flutter, la decisión partió de la experiencia como grupo en esta tecnología y por el gran soporte que posee el lenguaje, es importante recordar que es de código abierto y fue creado por Google, su principal uso es para desarrollar aplicaciones cross platform desde una sola base de código para web.

La fase de diseño de las interfaces gráficas fue realizada en conjunto con los referentes funcionales y directores del proyecto, nos pareció importante la participación y su posterior satisfacción, ya que la usabilidad es producto de un desarrollo intuitivo y amigable al usuario.

Al igual que en la etapa de análisis, en la mitad del proyecto hubo que realizar decisiones de diseño para cuestiones relacionadas al módulo de segmentación de imágenes. Esto se realizó luego de tener reuniones con los referentes funcionales y directores del proyecto.



## 12.6 Desarrollo e implementación

Tal como se puede observar en la comparación entre la planificación original y la ejecutada, el tiempo demandado en la fase de desarrollo fue menor. La estimación de desarrollo en el módulo de segmentación de imágenes terminó siendo levemente menor al planificado. Esto se debe al desconocimiento que teníamos sobre el tema al momento de estimar la planificación. Por otro lado, la resolución de los módulos anteriores nos dieron más experiencia sobre el tema que hicieron que los tiempos de desarrollo sean más cortos.

A diferencia de lo planteado en el cronograma, el orden de las tareas de desarrollo no fue respetado de forma secuencial. La posibilidad de dividirnos tareas nos permitió resolver diferentes módulos del proyecto simultáneamente, en donde cada integrante del equipo de trabajo resolvía algo distinto, esta metodología de trabajo nos resultó extremadamente eficiente y consideramos que fue el motivo por el que logramos adelantarnos al cronograma.

La desventaja de esta metodología de trabajo fue que cada semana era estrictamente necesario mantener reuniones en los fines de semana en las que se revisaban los avances de todo y así lograr que todos estemos al tanto de los avances y poder hacer una revisión global sobre lo que se avanzaría la semana siguiente.

## 12.7 Validación y ajustes

Dentro de esta fase, se contemplaron las reuniones con los referentes y su posterior retroalimentación al desarrollo realizado. Las horas estimadas en un principio no fueron las correctas por muy poco. Esto se debió a que se necesitó una reunión adicional sobre el final debido a funcionalidades adicionales sobre el módulo de imágenes. A su vez, no se realizaron en el momento que se había previsto. Esto se debió a varios factores. Principalmente la disponibilidad de cada uno y también a que el desarrollo fue menor al estimado en un principio, por lo que algunas



reuniones se adelantaron para poder mostrar el desarrollo final de alguno de los módulos.

## 12.8 Pruebas de integración

Como se puede observar, existe una diferencia entre lo planificado y lo ejecutado finalmente. Primero, las fases de integración no fueron realizadas dentro de las semanas previstas. Esto se debió a que el desarrollo de algunos módulos fueron terminados antes de lo previsto inicialmente. A su vez, se necesitaron más horas que las planificadas ya que finalmente el módulo de segmentación de imágenes fue dividido en dos submódulos ya que se agregó la funcionalidad de poder cargar un archivo nuevo. Esto requería realizar pruebas separadas.

## 12.9 Redacción del informe

Esta fase final nos sirvió como enseñanza sobre el manejo de los tiempos y la necesidad de mantener la documentación constantemente actualizada a la par de los avances del proyecto. En la planificación inicial, la estimación de tiempos era mucho menor. Finalmente, nos llevó mucho más tiempo de lo esperado.

Cometimos el error de respetar la planificación secuencial planteada en el protocolo y comenzamos con la redacción del informe casi al final del desarrollo, hacer esto implicó volver varios meses atrás en nuestra bitácora y hacer memoria sobre las pequeñas decisiones tomadas que no estaban documentadas. Sin embargo, fue un aprendizaje para nosotros y consideramos que para futuros proyectos sería una buena decisión no dejar la fase de documentación al final.

Hemos aprendido que hubiera sido una mejor decisión comenzar con la redacción a la par de las implementaciones de manera que el informe también colabore en el rol de bitácora o seguimiento del proyecto.

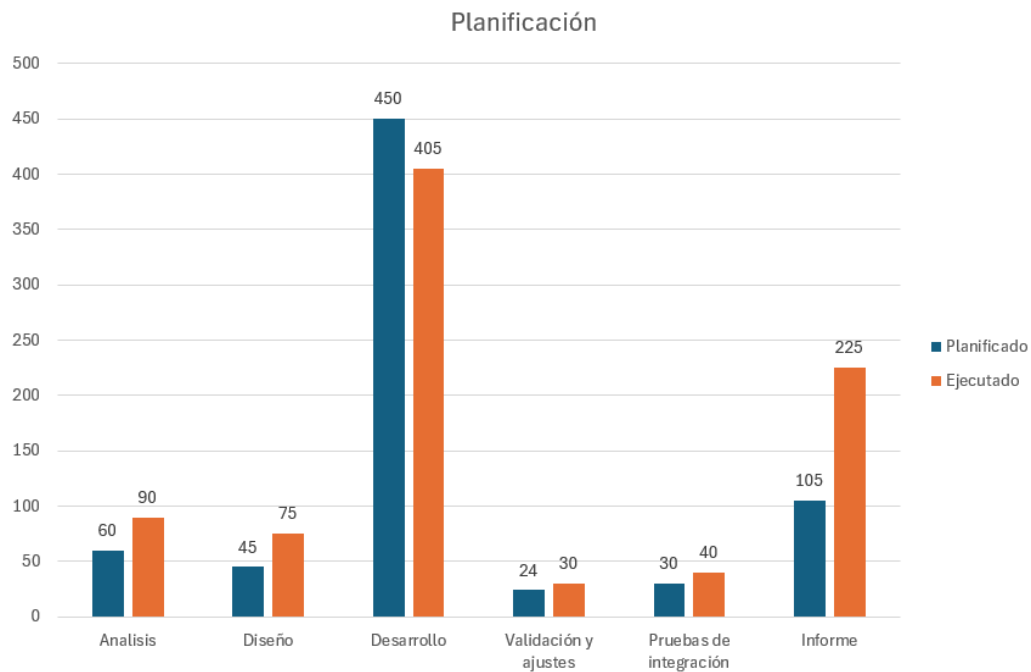


Figura 43: Especificación de horas por etapa

Como se observa en la figura anterior, la planificación de las etapas no se desarrolló como se esperaba. Hubo diferencias de costo en todas las fases del proyecto, alcanzando un total de 865 horas frente a las 714 horas estimadas inicialmente. Sin embargo, los plazos de entrega finales no se vieron afectados y el proyecto se completó en las 36 semanas estipuladas desde el inicio. Esto fue posible gracias a varios factores: primero, la fase de desarrollo finalizó antes de lo previsto, lo que permitió adelantar el inicio de algunas etapas. Además, todo el equipo mostró un alto compromiso y disciplina para cumplir con los plazos establecidos. Esto se logró dedicando más horas semanales a las distintas tareas y optando por una planificación más flexible, en lugar de seguir la estimación lineal inicial. Así, se trabajaron simultáneamente en actividades como el desarrollo, las pruebas de integración y la redacción del informe.





## 12.10 Metodología de trabajo

Para mantener el ritmo de trabajo, los estudiantes que desarrollamos este proyecto realizamos reuniones semanales, en algunos casos presenciales y otros virtuales, en los que dedicamos el tiempo en compartir los avances que había hecho cada uno, coordinar las tareas a realizar para la próxima reunión y actualizar la documentación del proyecto para mantenerla al día.

Creemos que la disciplina por no suspender estas reuniones a pesar de otros compromisos académicos o laborales fue esencial para obtener la ejecución del proyecto que se vio reflejada en la grilla mostrada anteriormente en este informe. Esta fue una de las principales razones por las que a pesar de las diferencias de estimación de horas por etapa, se logró cumplir con el plazo de entrega previsto según la planificación inicial.

Luego, más allá de las reuniones para avanzar en el desarrollo, coordinamos reuniones mensuales con nuestros directores y el Laboratorio de Bioingeniería para validar los avances sobre el proyecto y tener en claro que son positivos. Estas reuniones fueron muy fructíferas para corregir el rumbo y fueron la retroalimentación necesaria para lograr el producto esperado.

Para la organización y un desarrollo prolijo del proyecto implementamos múltiples herramientas y buenas prácticas de programación para garantizar un eficiente manejo del tiempo.

## 12.11 Software auxiliar para la gestión del proyecto

En primer lugar, para organización de las tareas y documentar los avances utilizamos un tablero Trello que constantemente actualizamos con la información pertinente a cada semana.



Trello es un software diseñado para la gestión de actividades y la administración de proyectos de manera colectiva. Está optimizado para organizar información en formatos visuales simples que facilitan la realización de tareas y el cumplimiento de objetivos<sup>(28)</sup>.

Para mantener un diálogo fluido incorporamos el uso de un servidor de Discord para coordinar las reuniones virtuales, compartir archivos e incluso pantalla, de esta manera las reuniones semanales virtuales fueron fluidas y productivas.

Discord es un servicio de mensajería instantánea y chat de voz *VoIP*. En esta plataforma, los usuarios tienen la capacidad de comunicarse por llamadas de voz, videollamadas, mensajes de texto, o con archivos y contenido multimedia en conversaciones privadas o como parte de comunidades llamadas servidores<sup>(29)</sup>.

Por otro lado, respecto al manejo de código utilizamos dos repositorios Git alojados en la plataforma Github.

Git es un software de control de versiones diseñado por Linus Torvalds, pensando en la eficiencia, la confiabilidad y compatibilidad del mantenimiento de versiones de aplicaciones cuando estas tienen un gran número de archivos de código fuente. Su propósito es llevar registro de los cambios en archivos de computadora incluyendo coordinar el trabajo que varias personas realizan sobre archivos compartidos en un repositorio de código<sup>(30)</sup>.

De esta manera, logramos manejar fácilmente las nuevas versiones del proyecto, queremos resaltar la importancia de esta herramienta ya que como balance una vez finalizada la fase de desarrollo podemos concluir que las siguientes funcionalidades de Git fueron de suma importancia para el proyecto:

- *Atomicidad de las tareas y manejo de ramas*: El uso de Git en conjunto con Trello permitió subdividir el proyecto en subtareas las cuales podían asignarse a uno de nosotros para su ejecución. La división de tareas fue un



factor clave durante todo el proyecto para lograr un ritmo de trabajo constante, con tareas atómicas y bien definidas para facilitar la tarea de desarrollo. Las ramas de git permitieron evitar que los cambios que uno de nosotros hacía repercutan sobre los cambios que otro integrante del proyecto estaba haciendo.

- *Control de versiones y backup:* Al ser Git un software de control de versiones, ante cualquier imprevisto o error que cometimos en el proyecto fue muy sencillo volver atrás a una versión anterior y corregir aquellos inconvenientes que aparecieron durante la fase de desarrollo.
- *Integración de cambios:* La integración de cambios se simplificó a partir de las “Pull Request” que propone Github, al finalizar las tareas de desarrollo asignadas a cada uno, los cambios se movían a la rama principal usando una pull request donde el resto de las integrantes podía verificar los cambios, entenderlos y tras mutuo acuerdo entre todos confirmar e incorporar los cambios en la rama principal del repositorio.

Por último, cuando el proyecto se encontró en estado avanzado y funcionando fue necesario desplegar la aplicación ya que las primeras fases de prueba se realizaron en un entorno local. Optamos por utilizar una plataforma cloud llamada Vercel para nuestro frontend, mientras que utilizamos Render para el backend.

Vercel es una aplicación que permite el *deploy* de aplicaciones independientes de su backend, y con la ventaja de proporcionar planes gratuitos que son suficientes para las necesidades de nuestra aplicación<sup>(31)</sup>. Vercel simplifica el proceso de hosteo de la aplicación ya que solamente necesita acceso al código fuente en un repositorio Git y establecer un dominio.

En el caso de backend, implementamos el sistema en Render, una plataforma cloud que permite compilar, desplegar y escalar aplicaciones sin importar el número de usuarios que consulten el backend del servicio<sup>(32)</sup>.



Sin embargo, el plan gratuito no cumple con nuestras expectativas, ya que los entrenamientos de la red neuronal requieren mucho poder de cómputo, y para lograr esto en Render debe contratarse alguno de los planes pagos. Queda a disposición de los referentes funcionales integrantes del Laboratorio de Bioingeniería financiar o no esta alternativa.

El *deploy* tanto del frontend como del backend fue diseñado inicialmente para uso interno del Departamento. Sin embargo, si en el futuro se busca que el sistema sea utilizado de forma más amplia y por un mayor número de usuarios, será necesario replantear el despliegue de las aplicaciones. Esto se debe a que un uso más masivo implicaría mayores demandas en términos de escalabilidad, rendimiento y seguridad, aspectos que no son críticos en el entorno interno actual. Además, el software deberá rediseñarse para admitir diferentes perfiles de usuario, incluyendo administradores para quienes pertenezcan al Departamento y usuarios comunes para el resto.



## 13. Conclusiones

El objetivo principal del proyecto era analizar la problemática referida al análisis de datos de gran volumen y/o con alta dimensionalidad mediante el uso de una herramienta intuitiva y simple. A su vez, el sistema debe poder ser escalable para poder continuar con el desarrollo de nuevas funcionalidades en el futuro. El sistema obtenido como resultado del desarrollo de este trabajo cumple con los requerimientos y características indicadas por los referentes funcionales, por lo que el objetivo fue cumplido con éxito. Respecto al objetivo secundario (dar visibilidad a los SOM) no podemos concluir respecto a su cumplimiento o no debido a su dificultad para ser cuantificado. Desde nuestra parte, creemos que nuestro esfuerzo es positivo y fructífero, y ojalá a futuro este trabajo sirva como base para que aquellos interesados en el tema puedan aprender más sobre el mismo y utilizar la herramienta.

Una de las tareas más difíciles de llevar a cabo fue la planificación y estimación de los tiempos. Si bien la extensión del proyecto en cuanto a cantidad de horas totales fue la esperada, no se dividieron de la manera que se estimó en un principio y la distribución de tiempos para cada una de las etapas y la dedicación semanal no fue la esperada finalmente. Una de las causas principales fue el hecho de no contar con la experiencia necesaria dado que ninguno había realizado una estimación de esta magnitud.

Durante la fase final del proyecto aprendimos que la estimación de tiempos de cada tarea a realizar y su distribución en subtareas no es tan simple como una “regla de 3 simple”. Por más que no sea simple estimar el tiempo que demandará una tarea, estamos muy conformes con el trabajo realizado y creemos que fue una experiencia muy positiva teniendo en cuenta que es nuestra primera gestión de un proyecto real. Por otro lado, en el desarrollo de las actividades es una tarea difícil pero no es imposible predecir las circunstancias ajenas al proyecto que pueden demorarlo, es una de las tareas de un buen líder de proyecto poder hacerlo.



Tanto las autoridades del Laboratorio de Bioingeniería como nosotros coincidimos en que el trabajo está en condiciones de ser continuado por otro grupo de estudiantes dispuesto a mejorar el sistema e incorporar nuevos módulos. El crecimiento de la Inteligencia Artificial como disciplina nos deja curiosos sobre los posibles avances que a futuro puedan incorporarse al software desarrollado. Estamos dispuestos como grupo a colaborar a futuro con los estudiantes que se interesen en el tema y deseen aportar en el mismo. No obstante, si por razones ajenas a nosotros esto no es posible nos comprometemos a realizar las capacitaciones correspondientes a los nuevos estudiantes.

Hay que destacar el aporte intrainstitucional que este proyecto significa para la Facultad de Ingeniería de la Universidad Nacional de Mar del Plata, debido a que es un proyecto interdepartamental que se realizó en conjunto con el Departamento de Electrónica y Computación.

A modo de cierre y para tener una reflexión final de nuestros demandantes les consultamos su opinión, a lo que dijeron: "Desde la concepción teórica hasta la ejecución práctica, el trabajo evidencia una profunda comprensión de los conceptos fundamentales y una notable capacidad para aplicarlos a problemas reales, tal como fuera planteado desde su comienzo. Presenta un alto potencial de aplicación en el ámbito de la bioingeniería y el procesamiento digital de imágenes. Felicitamos a los autores por su dedicación y les auguramos un futuro exitoso en su carrera profesional."

La realización de este trabajo final permitió poner en práctica una amplia gama de competencias adquiridas a lo largo de la carrera, reforzando nuestra formación como futuros ingenieros en informática. Entre estas competencias se destacan la capacidad de llevar adelante proyectos complejos desde la idea hasta la ejecución, organizar recursos y tiempos de manera eficiente, y resolver problemas mediante el uso de herramientas de ingeniería informática. Asimismo, el trabajo en equipo fue fundamental para alcanzar los objetivos planteados, favoreciendo la comunicación efectiva y la toma de decisiones basada en técnicas de gestión actuales. Esta



experiencia no solo fortaleció habilidades técnicas, sino que también subrayó la importancia de la capacitación continua y la capacidad de adaptarse a circunstancias imprevistas, aspectos esenciales en el ejercicio profesional de la ingeniería.

En conclusión, el resultado fue exitoso. Desde el punto de vista de desarrollo, se logró entregar un producto que cumple con los requerimientos pedidos. Esto se logró debido a la colaboración y compromiso que se tuvo a lo largo de todo el trabajo. Por otro lado, desde el punto de vista personal, el aprendizaje fue muy importante. Fue nuestra primera experiencia administrando y gestionando un proyecto en el que pudimos implementar muchos de los conocimientos aprendidos a lo largo de toda nuestra carrera como estudiantes.



## 14. Apéndices

### 14.1 Apéndice A - Endpoints - API

Endpoint	Method	Body	Status code	Response
api/train	POST	{ datos: string , params: string, etiquetas:string }	200	{ data: json [ ], neurons:json, parámetros:json errores:json }
api/train	POST	{ datos: string , params: string, etiquetas:string }	500	{ error: string }
api/bmu	POST	{ datos: string , params: string, etiquetas:string, codebook:string, nuevosDatos:string }	200	{ Resultado: string, Dato:string }
api/bmu	POST	{ datos: string , params: string, etiquetas:string, codebook:string, nuevosDatos:string }	500	{ error: string }
api/clusters	POST	{ datos: string, params: string, codebook:string }	200	int [ ]
api/clusters	POST	{ datos: string, params: string, codebook:string }	500	{ error: string }

Tabla 2 - endpoints API implementada en Python





## 14.2 Apéndice B - Glosario

**API:** conjunto de definiciones y protocolos que se utiliza para desarrollar e integrar los sistemas de software de las aplicaciones<sup>(33)</sup>.

**Backend:** se refiere a la parte del sistema informático que se encarga de almacenar y procesar los datos de la aplicación para los usuarios<sup>(34)</sup>.

**Frontend:** se refiere a la parte visible de un sistema con la que los usuarios pueden interactuar directamente<sup>(34)</sup>.

**Software:** sistema formal de un sistema informático, que comprende el conjunto de los componentes lógicos necesarios que hace posible la realización de tareas específicas<sup>(35)</sup>.

**HTTP:** (en inglés: *Hypertext Transfer Protocol*, abreviado HTTP) es el protocolo de comunicación que permite las transferencias de información a través de archivos en la World Wide Web<sup>(36)</sup>.

**REST:** REST no es un protocolo ni un estándar, sino más bien un conjunto de límites relacionados con la arquitectura. Los desarrolladores de las API pueden implementarlo de distintas maneras.<sup>(37)</sup>

**JSON:** (acrónimo de *JavaScript Object Notation*, 'notación de objeto de JavaScript') es un formato de texto sencillo para el intercambio de datos. Se trata de un subconjunto de la notación literal de objetos de JavaScript.

**PNG:** el formato de archivo PNG se usa ampliamente en sitios web para mostrar imágenes digitales de alta calidad. Creados para superar el rendimiento de los archivos GIF, los PNG no solo ofrecen una compresión sin pérdida, sino también una paleta de colores mucho más amplia y brillante <sup>(38)</sup>.



**Testing:** es un proceso para verificar y validar la funcionalidad de un programa o una aplicación de software con el objetivo de garantizar que el producto de software esté libre de defectos<sup>(39)</sup>.

**BMU:** de las siglas en inglés *Best Matching Unit*. Se define como la célula cuyo vector prototipo es el más cercano de un vector de datos de entrada<sup>(5)</sup>.

**Mapa autoorganizado:** redes neuronales que se entrenan con paradigma de aprendizaje no supervisado, formadas por cuadrículas hexagonales de células para mapear espacios de entrada (espacios de patrón) a espacios de celda (espacios topológicos), preservando la topología del espacio de entrada y tener capacidades notables para eliminar ruido, para detectar valores atípicos y para completar valores faltantes<sup>(2)</sup>.

**U-Matrix:** es una representación del mapa en el que las distancias entre los vectores prototipo de dos celdas adyacentes se representan con un código de color<sup>(5)</sup>.

**Mapa de componentes:** un mapa en el que pueden analizarse los valores que toma cada variable del vector prototipo de cada celda y representarse en mapas separados, mediante una escala de colores o de intensidades de grises. Estos mapas guardan relación topológica entre sí<sup>(14)</sup>.

**Mapa de hits:** un mapa en el que cada celda está etiquetada con el número de veces que fue BMU. Sigue el mismo criterio de color que en los mapas de componentes<sup>(5)</sup>.

**Clustering:** es la tarea de agrupar objetos por similitud, en grupos o conjuntos (*clusters*) de manera que los miembros del mismo grupo tengan características similares<sup>(15)</sup>.

**String:** secuencia de caracteres usado para representar el texto<sup>(40)</sup>.



## 15. Bibliografía

1. The Self-Organizing map.  
<https://ieeexplore.ieee.org/abstract/document/58325>
2. Diego S. Comas , Agustín Amalfitano, Luciana Simón González Gustavo J. Meschino, y Virginia L. Ballarin.  
“Pneumonia classification and analysis in Chest X Ray by means of Convolutional Neural Networks”.XXIII CONGRESO ARGENTINO DE BIOINGENIERÍA Y XII JORNADAS DE INGENIERÍA CLÍNICA – SABI 2022.
3. Analisis de componentes principales. Wikipedia.  
[https://es.wikipedia.org/wiki/An%C3%A1lisis\\_de\\_componentes\\_principales](https://es.wikipedia.org/wiki/An%C3%A1lisis_de_componentes_principales)
4. GeeksForGeeks. *¿Qué es la normalización de datos?*.  
[https://www.geeksforgeeks.org/what-is-data-normalization/?ref=oin\\_asr4](https://www.geeksforgeeks.org/what-is-data-normalization/?ref=oin_asr4)
5. GeeksForGeeks. *¿Qué es la normalización por varianza?* Recuperado de  
<https://www.geeksforgeeks.org/what-is-zero-mean-and-unit-variance-normalization/>
6. Diego S. Comas, Gustavo J. Meschino, Agustín Amalfitano y Virginia Ballarin.  
“Analysis and interpretation of Deep Convolutional Features using Self-Organizing Maps”. Capítulo del libro: Innovations in Machine and Deep Learning. Eds: L. Cruz-Reyes, B. Dorronsoro, G. Rivera, A. Rosete. Serie: Studies in Big Data, Volume: 134. Springer, Cham, Switzerland, 2023. ISBN: 978-3-031-40687-4, pp. 213-229.
7. Arquitectura Cliente-Servidor  
[https://developer.mozilla.org/es/docs/Learn/Server-side/First\\_steps/Client-Server\\_overview](https://developer.mozilla.org/es/docs/Learn/Server-side/First_steps/Client-Server_overview)



8. Web Oficial de Moqups.  
<https://moqups.com/es/>
9. Web oficial Flask [en línea].  
<https://flask.palletsprojects.com/en/3.0.x/>
10. IntraSOM. *Repositorio de IntraSOM*. GitHub. Recuperado de  
<https://github.com/InTRA-USP/IntraSOM>
11. Web oficial Dart [en línea].  
<https://dart.dev/>
12. Web oficial Flutter [en línea].  
<https://flutter.dev/>
13. Web oficial JSON [en línea].  
<https://www.json.org/json-es.html>
14. Gustavo J. Meschino, Gerardo Tusman , Lucía I. Passoni  
“Mapas autoorganizados para el descubrimiento de patrones fisiológicos de ventilación pulmonar”. 2018 - CONAIIISI
15. Agrupamiento.Wikipedia  
[https://es.wikipedia.org/wiki/An%C3%A1lisis\\_de\\_grupos](https://es.wikipedia.org/wiki/An%C3%A1lisis_de_grupos)
16. Web oficial K-Means de ScikitLearn [en línea].  
<https://scikit-learn.org/1.5/modules/generated/sklearn.cluster.KMeans.html>
17. Foro de discusión de GitHub sobre el método de clustering K-Means de ScikitLearn. GitHub.  
<https://github.com/scikit-learn/scikit-learn/discussions/24964>



18. Web oficial Postman [en línea].

<https://www.postman.com/>

19. Repositorio de GitHub de SOMPY.

<https://github.com/sevamoo/SOMPY>

20. Repositorio de GitHub de MiniSOM.

<https://github.com/JustGlowing/minisom>

21. Documentación oficial de R sobre el paquete Kohonen.

<https://cran.r-project.org/web/packages/kohonen/kohonen.pdf>

22. Repositorio de GitHub de SkLearnSOM .

<https://github.com/rileypsmith/sklearn-som>

23. Documentación de Orange [en línea] .

<https://docs.biolab.si/orange/2/reference/rst/Orange.projection.som.html>

24. Documentación de Knime [en línea] .

[https://hub.knime.com/knime/extensions/org.knime.features.ext.weka\\_3.7/la  
test/org.knime.ext.weka37.clusterer.WekaClustererNodeFactory:def80f02](https://hub.knime.com/knime/extensions/org.knime.features.ext.weka_3.7/la<br/>test/org.knime.ext.weka37.clusterer.WekaClustererNodeFactory:def80f02)

25. Documentación de Weka [en línea] .

<https://jsalatas.ictpro.gr/weka/doc/SelfOrganizingMap/>

26. Documentación de MatLab [en línea] .

<https://www.mathworks.com/help/deeplearning/ref/selforgmap.html>

27. Documentación de Viscovery Somine [en línea] .

<https://www.viscovery.net/self-organizing-maps>



28. Web oficial Trello [en línea].

<https://trello.com/es>

29. Web oficial Discord [en línea].

<https://discord.com/>

30. Web oficial GitHub [en línea]

<https://git-scm.com/>

31. Web oficial GitHub [en línea]

<https://github.com/>

32. Web oficial Vercel [en línea].

<https://vercel.com>

33. Web oficial Render [en línea].

<https://render.com>

34. Definición API. Foro de RedHat.

<https://www.redhat.com/es/topics/api/what-is-a-rest-api>

35. Platzi. *¿Qué es el Frontend y Backend?*.

<https://platzi.com/blog/que-es-frontend-y-backend/>

36. Definición de software. Wikipedia.

<https://es.wikipedia.org/wiki/Software>

37. HTTP [en línea]. Foro de Mozilla.

<https://developer.mozilla.org/es/docs/Web/HTTP>

38. Definición de PNG. Foro de Adobe.

<https://www.adobe.com/ar/creativecloud/file-types/image/raster/png-file.html>



39. Definición de Testing.

<https://profile.es/blog/que-es-el-testing-de-software/>

40. Definición de String. Foro de Mozilla.

<https://developer.mozilla.org/es/docs/Glossary/String>



## **16. Anexo: Manual de uso**





# **Sistema de análisis de datos y soporte a la toma de decisiones basado en Mapas Autoorganizados “VisualiSOM”**

*Manual de uso*

Autores:

- [Lisandro D'Alu De Boni](#)
- [María Camila Ezama](#)
- [Augusto Maletta](#)



<b>Requisitos previos.....</b>	<b>2</b>
(Opcional) Ejecución local del servidor Python.....	2
<b>Carga de datos.....</b>	<b>3</b>
Configuración API.....	4
Carga de archivo.....	5
Opción 1: Cargar archivo de datos.....	6
1. Marcar o no detección automática de features.....	6
2. Cargar un archivo de datos al sistema.....	6
3. Seleccionar columnas que se utilizaran en el entrenamiento.....	7
4. Seleccionar columnas que son etiquetas.....	7
5. Configurar parámetros.....	8
6. Elegir gradiente de colores.....	8
7. Presionar Entrenar.....	9
Opción 2: Cargar archivo con datos de un entrenamiento anterior.....	10
1. Elegir gradiente de colores.....	10
8. Presionar Cargar archivo.....	10
<b>Visualización del SOM.....</b>	<b>11</b>
Información del resultado.....	12
Menú de opciones.....	13
Opción Mapa.....	14
Opción Umat+.....	15
Opción Componentes.....	16
Opción Hits.....	18
Opción Clustering.....	20
Opción Nuevos datos.....	21
Opción Imagen Datos Train.....	23
Opción Clustering.....	24
Opción Coloreado Continuo.....	27
Opción Imagen Nueva.....	29



## Requisitos previos

- Navegador (Chrome, Firefox, Edge o similares)
- En caso de de decidir ejecutar localmente el servicio, se requiere tener instalado Python (se sugiere 3.10+)

## (Opcional) Ejecución local del servidor Python

### Requisitos:

- Una computadora con conexión a internet constante, encendida.
- Abrir el puerto TCP usado por la aplicación (7777) para permitir conexiones.

En primer lugar, instalar los archivos fuentes de la aplicación y sus prerequisites, esto puede lograrse descargando los archivos del repositorio: [Repositorio](#)

Abrir una terminal en el directorio que contiene los archivos descargados, y ejecutar el comando

```
pip install -r requirements.txt
```

Una vez hecho esto estarán instaladas las dependencias necesarias para la ejecución de la aplicación

### Pasos a seguir:

1. Averiguar la IP local del router para abrir puertos. Suele ser 192.168.0.1
2. Abrir puertos en el router.
3. En la configuración del Router abrir puerto 7777

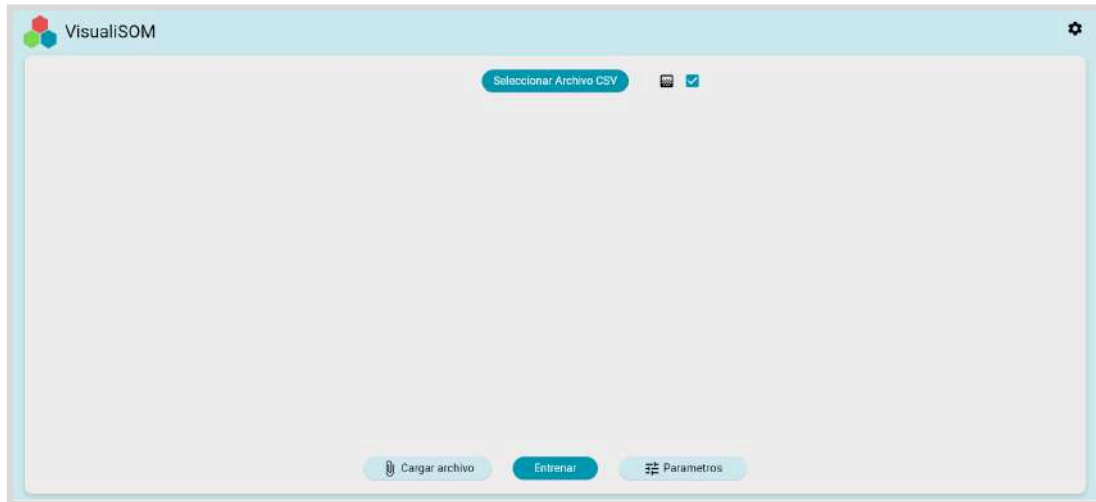
Nombre regla de puertos	SOM <span style="color: green;">Nombre para identificar</span>
Dirección IP	192.168.1.40 <span style="color: green;">La IP local de la computadora que va atender (server)</span>
Protocolo	TCP
Abrir Puerto/Rango Externo (WAN)	7777:7777
Abrir Puerto/Rango Interno (LAN)	7777:7777

Añadir



## Carga de datos

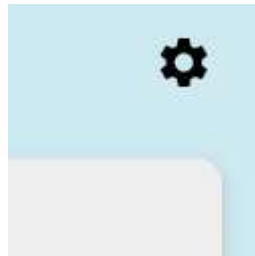
La primer pantalla de nuestro sistema se ve así:





## Configuración API

Lo primero que debemos hacer es configurar la API a la que se realizarán las consultas. Para ello, se presiona el botón de configuración ubicado en la esquina superior derecha.



Al hacerlo, se abrirá un diálogo de configuración. En este diálogo, se puede optar por la opción predeterminada (nuestra API publicada) o presionar el botón "Manual" para ingresar la dirección IP deseada, o "localhost" en caso de que la API esté ejecutándose localmente.

Configuración

Por defecto  Manual

Dirección IP: som-flask.onrender.com

Puerto: 7777

Cerrar



## **Carga de archivo**

Para utilizar la primera pantalla de nuestro sistema, hay dos opciones disponibles: cargar un archivo de datos para realizar el entrenamiento o cargar un archivo (descargado previamente de nuestro sistema) que contenga un entrenamiento anterior.

## Opción 1: Cargar archivo de datos

### 1. Marcar o no detección automática de features



Antes de cargar el archivo, puedes activar la opción Detección automática de features. Si esta opción está marcada, el sistema seleccionará automáticamente las etiquetas al cargar el archivo. Si no, deberás seleccionar las etiquetas manualmente más adelante.

### 2. Cargar un archivo de datos al sistema



Para cargar el archivo de datos, hacer clic en el botón **Seleccionar Archivo CSV**. A continuación, se abrirá un explorador de archivos en el dispositivo, donde deberás seleccionar el archivo CSV que contenga los datos.

Una vez que hayas elegido el archivo, su contenido se visualizará en pantalla:

The screenshot shows the VisualiSOM interface. At the top left is the logo and name 'VisualiSOM'. In the center, there is a button 'Seleccionar Archivo CSV' and a checked checkbox. Below this are two tabs: 'Features' and 'Etiquetas'. The main area contains a table with 11 columns and 9 rows of data. At the bottom, there are three buttons: 'Cargar archivo', 'Entrenar', and 'Parametros'.

Costos	Motivos	Edad	Sexo	Ocupacion	Ingreso Economico	Estado Civil	Convivencia	Grado de Instruccion	Act.Fisica Tiempo Libre	Vida Activa
<=1	OT	61	2	3	1	3	4	3	2	2
1a2	ACV	63	2	3	2	3	4	2	2	1
>10	PD	54	1	1	1	3	4	4	2	2
>10	PD	64	1	1	1	2	4	1	2	2
<=1	X	73	2	1	1	1	2	1	2	2
<=1	ICC	68	2	3	3	2	2	3	2	1
2a3	PD	60	1	1	1	2	2	2	2	2
2a3	ICC	63	2	5	1	2	4	3	2	2



### 3. Seleccionar columnas que se utilizaran en el entrenamiento

#### Features

Seleccionar columnas a utilizar. Las columnas que no estén seleccionadas no se tendrán en cuenta en el entrenamiento.

Columna	Seleccionada
Costos	<input type="checkbox"/>
Motivos	<input type="checkbox"/>
Edad	<input checked="" type="checkbox"/>
Sexo	<input checked="" type="checkbox"/>
Ocupacion	<input checked="" type="checkbox"/>
Ingreso Economico	<input checked="" type="checkbox"/>
Estado Civil	<input checked="" type="checkbox"/>
Convivencia	<input checked="" type="checkbox"/>
Grado de Instruccion	<input checked="" type="checkbox"/>
Act.Fisica Tiempo Libre	<input checked="" type="checkbox"/>

### 4. Seleccionar columnas que son etiquetas

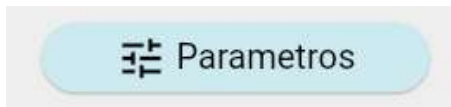
#### Etiquetas

Seleccionar también las columnas que funcionan como etiquetas, de modo que puedan ser visualizadas como tales en el sistema.

Columna	Seleccionada
Costos	<input checked="" type="checkbox"/>
Motivos	<input checked="" type="checkbox"/>
Edad	<input type="checkbox"/>
Sexo	<input type="checkbox"/>
Ocupacion	<input type="checkbox"/>
Ingreso Economico	<input type="checkbox"/>
Estado Civil	<input type="checkbox"/>
Convivencia	<input type="checkbox"/>
Grado de Instruccion	<input type="checkbox"/>
Act.Fisica Tiempo Libre	<input type="checkbox"/>



## 5. Configurar parámetros



Hacer clic en el botón Parámetros para abrir un diálogo donde podrás configurar los parámetros que se utilizarán en el entrenamiento.

Parametros configurables

Cantidad de filas: 14

Cantidad de columnas: 24

Cantidad de iteraciones de entrenamiento

Cantidad de iteraciones de refinamiento

Función de vecindad: Gaussiana

Inicialización: Aleatoria

Cerrar

## 6. Elegir gradiente de colores



Antes de presionar Entrenar, se puede configurar el gradiente de colores que se desea utilizar para visualizar el SOM.

Elegir gradiente

Guardar



## 7. Presionar Entrenar



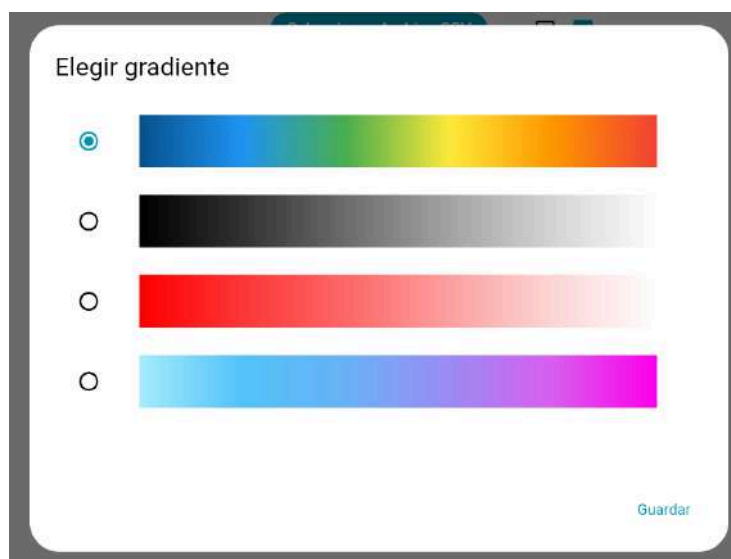
Una vez que se hayan configurado todos los aspectos anteriores, se está listo para iniciar el proceso de entrenamiento. Se debe hacer clic en el botón Entrenar para comenzar.

## Opción 2: Cargar archivo con datos de un entrenamiento anterior

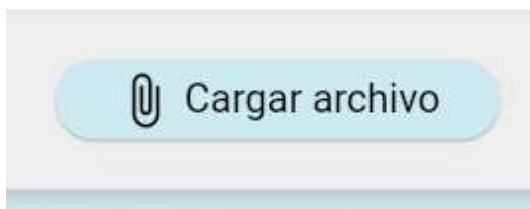
### 1. Elegir gradiente de colores



Antes de presionar el botón para cargar el archivo, se puede configurar el gradiente de colores que se desea utilizar para visualizar el SOM.



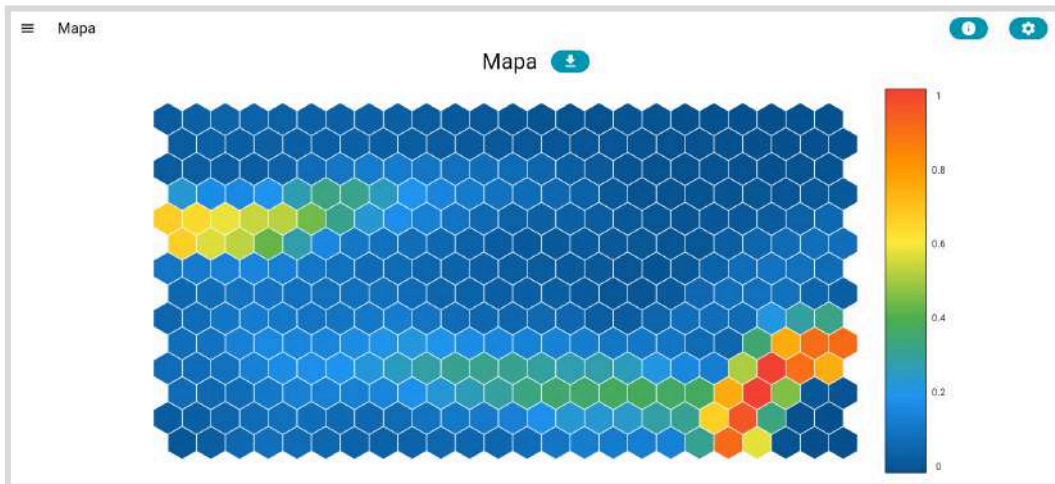
### 8. Presionar Cargar archivo



Para cargar el archivo con los datos de un entrenamiento previo, hacer clic en el botón **Cargar archivo**. A continuación, se abrirá un explorador de archivos en el dispositivo, donde deberás seleccionar el archivo CSV que contenga los datos.

## Visualización del SOM

Luego del entrenamiento, la pantalla que se observa sera algo asi:

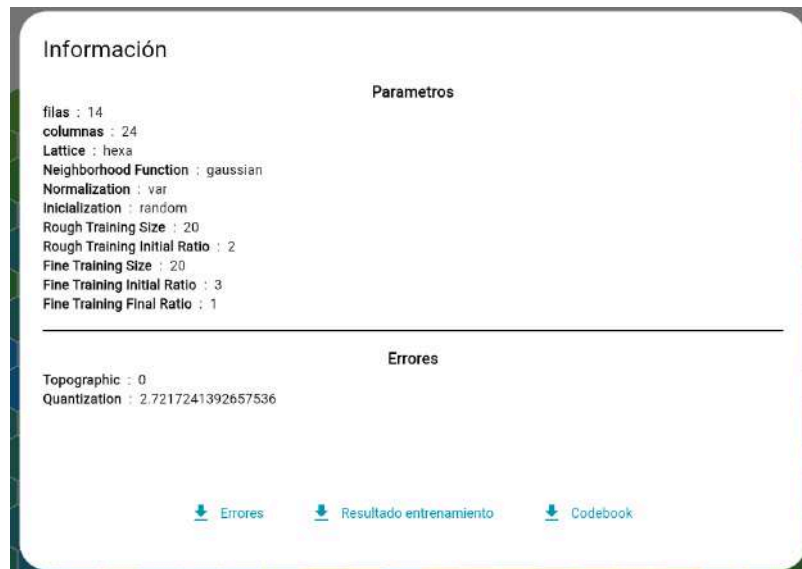




## Información del resultado



El boton de información que se encuentra en la esquina superior derecha, abre un dialog que muestra información del entrenamiento y los errores calculados.



Errores

El boton de Errores, descarga un archivo de texto con el resumen del entrenamiento (La información que se muestra en el dialog)



Resultado entrenamiento

El boton Resultado entrenamiento, descarga un archivo de texto, que contiene todos los datos que dio como resultado el entrenamiento.

**El archivo descargado es el que se puede ingresar en el sistema después, para ingresar a la pantalla que muestra el SOM sin volver a realizar el entrenamiento.**



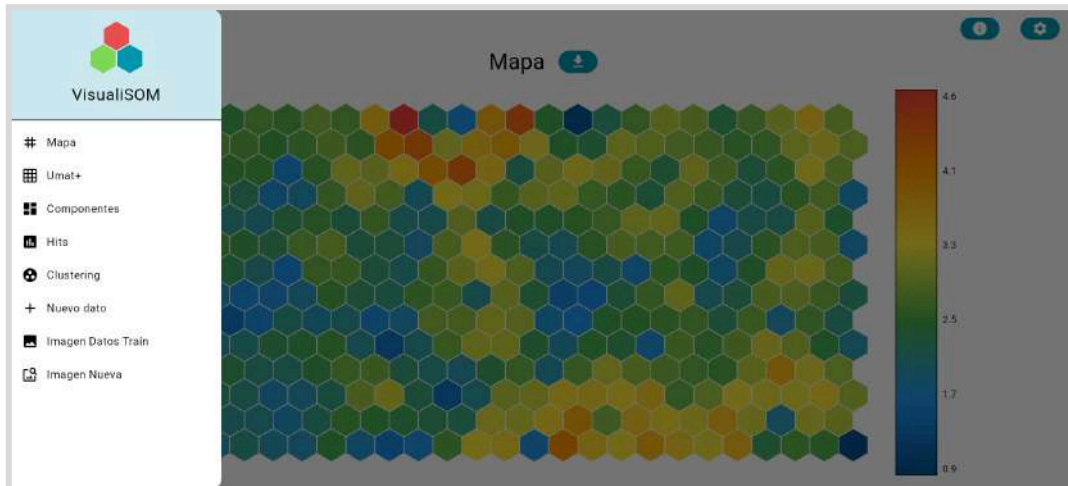
Codebook

El boton Codebook descarga un archivo que contiene todos los vectores del codebook.

## Menú de opciones



El boton que se encuentra en la esquina superior izquierda, abre el menú de opciones.



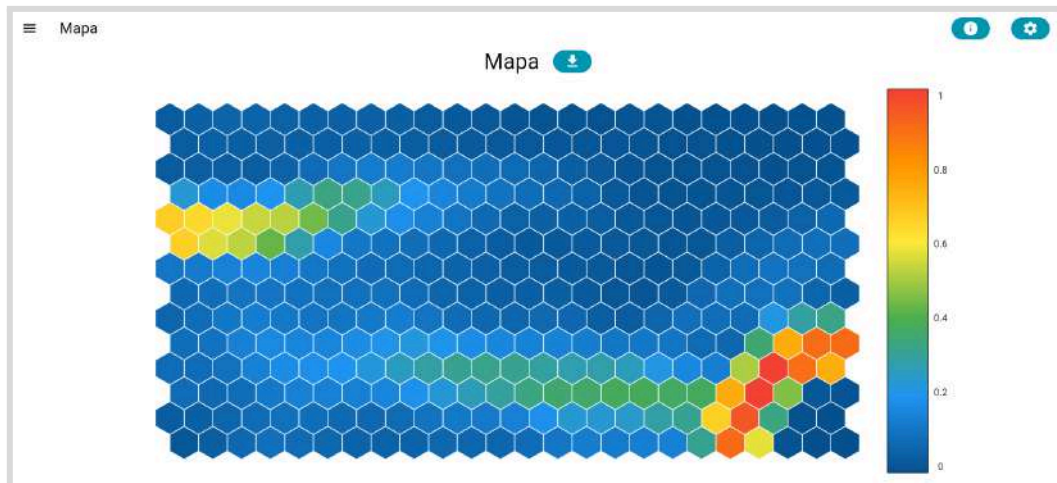
Seleccionando las opciones del menú, podemos cambiar entre las pantallas de nuestro sistema.

- # Mapa
- Umat+
- Componentes
- Hits
- Clustering
- + Nuevo dato
- Imagen Datos Train
- Imagen Nueva



## Opción Mapa

Mapa es la opción por defecto en nuestro sistema. Cada hexágono de la grilla representa una neurona, que corresponde a un vector dentro del codebook.

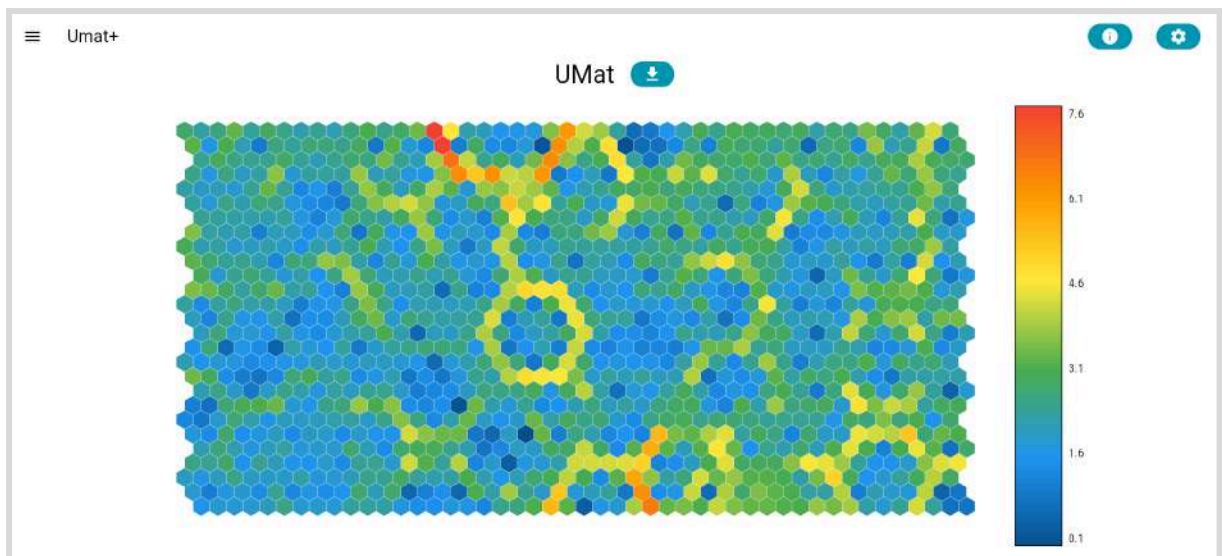


El boton descarga una imagen en formato .png de la grilla de hexágonos junto con el gradiente utilizado.



## Opción Umat+

La opción Umat+ muestra la grilla ampliada. Cada hexágono principal queda rodeado por nuevos hexágonos secundarios que representan la distancia entre vectores prototipo de sus celdas adyacentes.



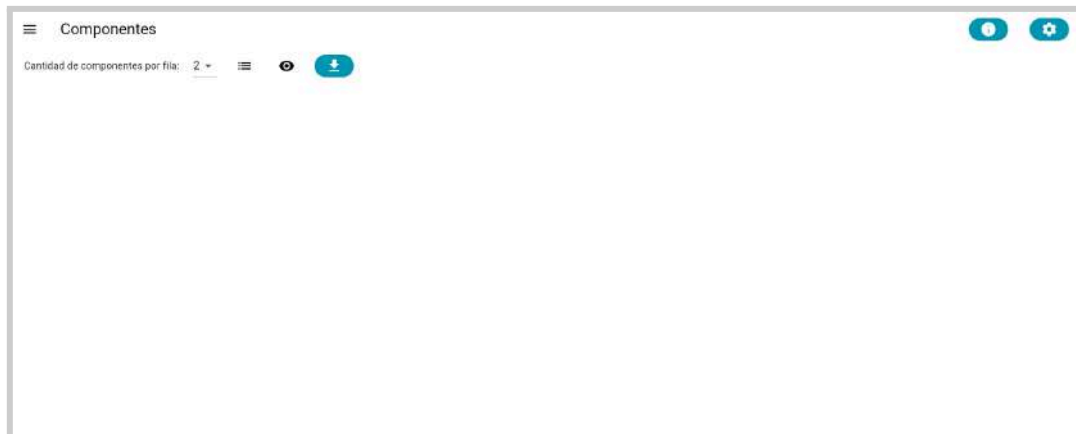
El botón descarga una imagen en formato .png de la grilla de hexágonos junto con el gradiente utilizado.





## Opción Componentes

La pantalla de Componentes, permite visualizar los mapas de componentes de manera interactiva.



Cantidad de componentes por fila: 2 ▼

Se puede seleccionar la cantidad de componentes que se quiere visualizar por fila.



Este boton, abre un dialog que permite seleccionar los features de los cuales queremos ver el mapa de componentes.

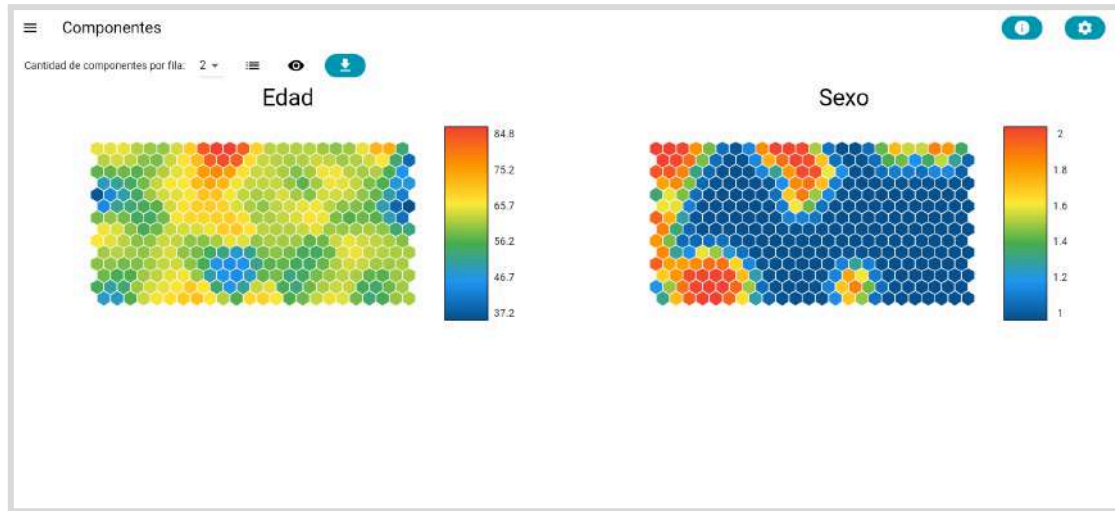
Seleccionar opciones

Edad	<input checked="" type="checkbox"/>
Sexo	<input checked="" type="checkbox"/>
Ocupacion	<input type="checkbox"/>
Ingreso Economico	<input type="checkbox"/>
Estado Civil	<input type="checkbox"/>
Convivencia	<input type="checkbox"/>
Grado de Instruccion	<input type="checkbox"/>
Act.Fisica Tiempo Libre	<input type="checkbox"/>
Vida Activa	<input type="checkbox"/>
NO realiza Actividad Fisica	<input type="checkbox"/>


Cerrar Guardar



Luego de seleccionar las Features que queremos ver en pantalla, se mostrará:

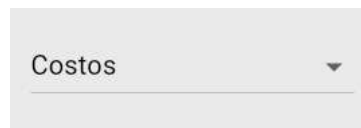
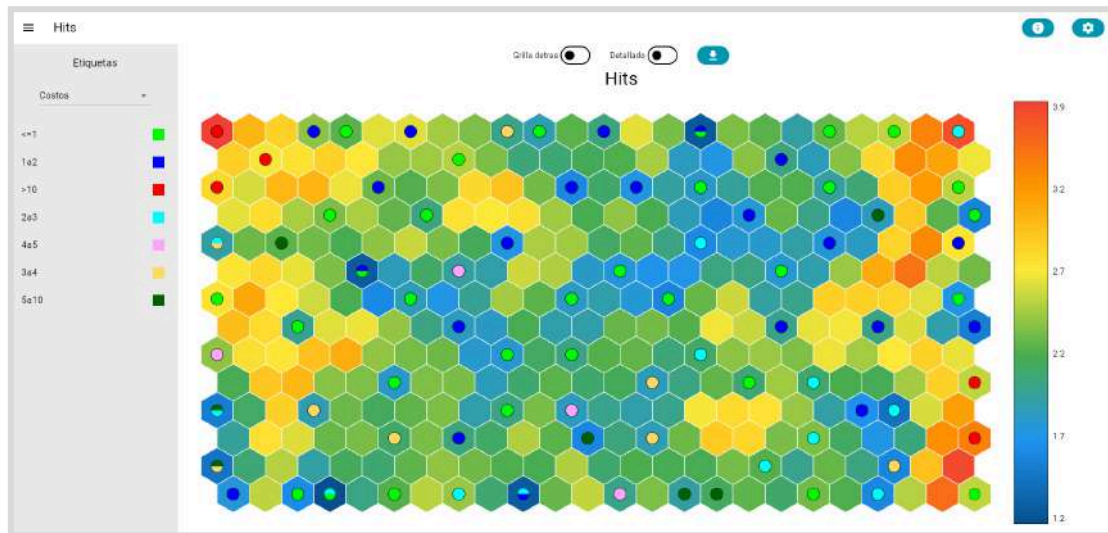


 El boton es para mostrar o no el gradiente al lado de la grilla

 El boton descarga una imagen en formato .png de los mapas que se ven en pantalla.

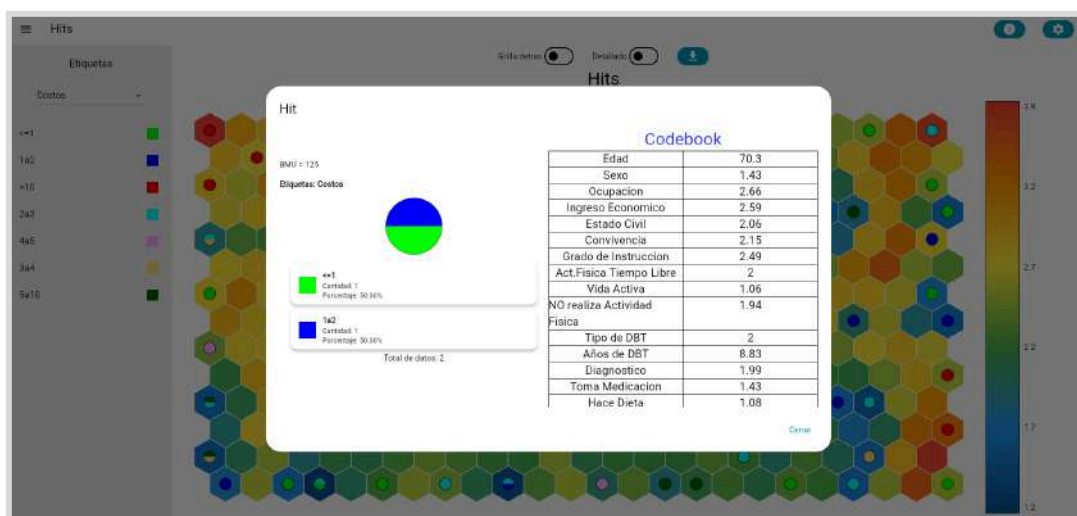
## Opción Hits

La funcionalidad "Hits" en nuestro sistema proporciona una visualización interactiva mediante una grilla de hexágonos, la cual refleja la distribución de los datos etiquetados entre las neuronas.



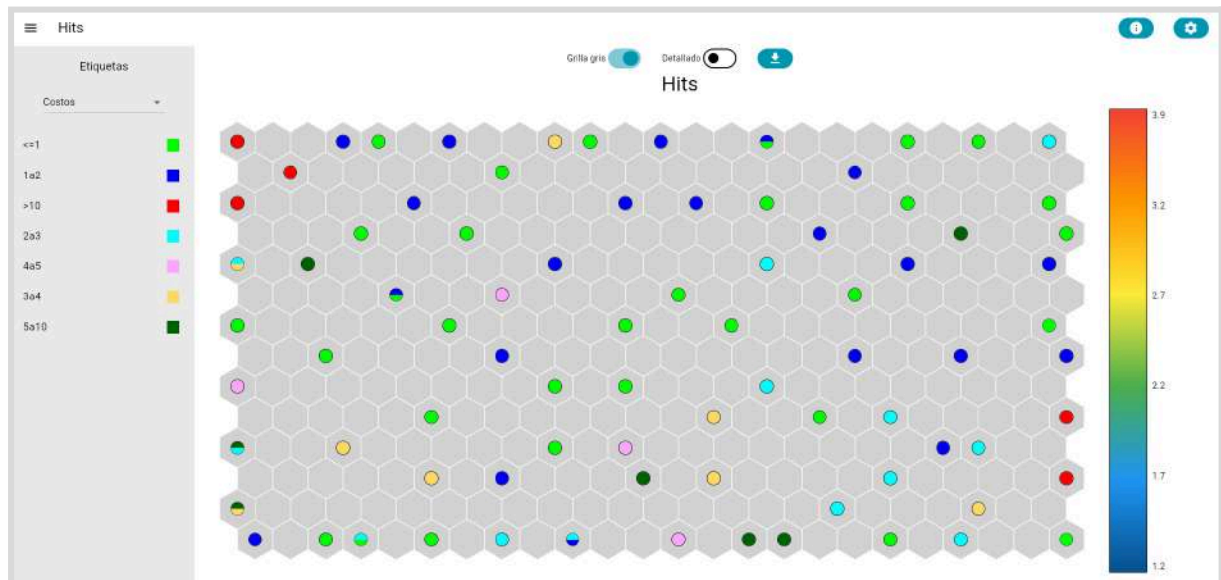
El campo de opciones permite seleccionar la categoría de etiquetas que queremos ver en el mapa.

Al hacer clic en un hexágono marcado, se abre un dialog con un gráfico de torta que ilustra la distribución de los datos según sus etiquetas y se visualiza el vector del codebook correspondiente.



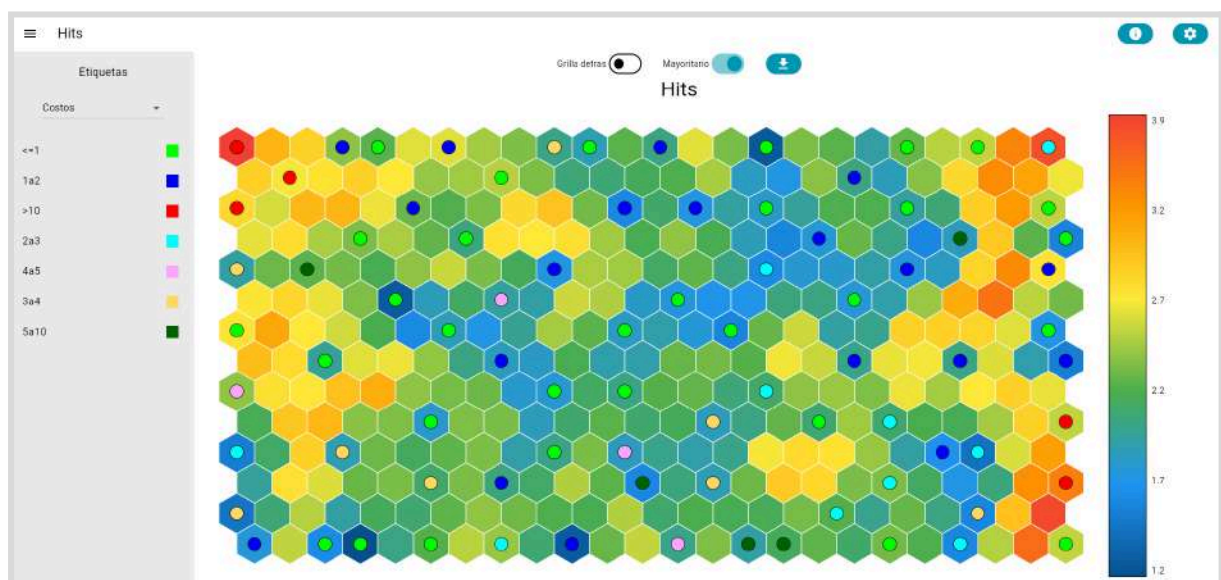
Grilla detras

Presionando el switch de Grilla detras, es posible mostrar u ocultar la grilla subyacente, que es la que corresponde a la opción "Mapa".



Detallado

Presionando el switch Detallado, se puede mostrar los gráficos de torta en cada hexágono para visualizar las etiquetas o mostrar sólo el color de la etiqueta predominante.

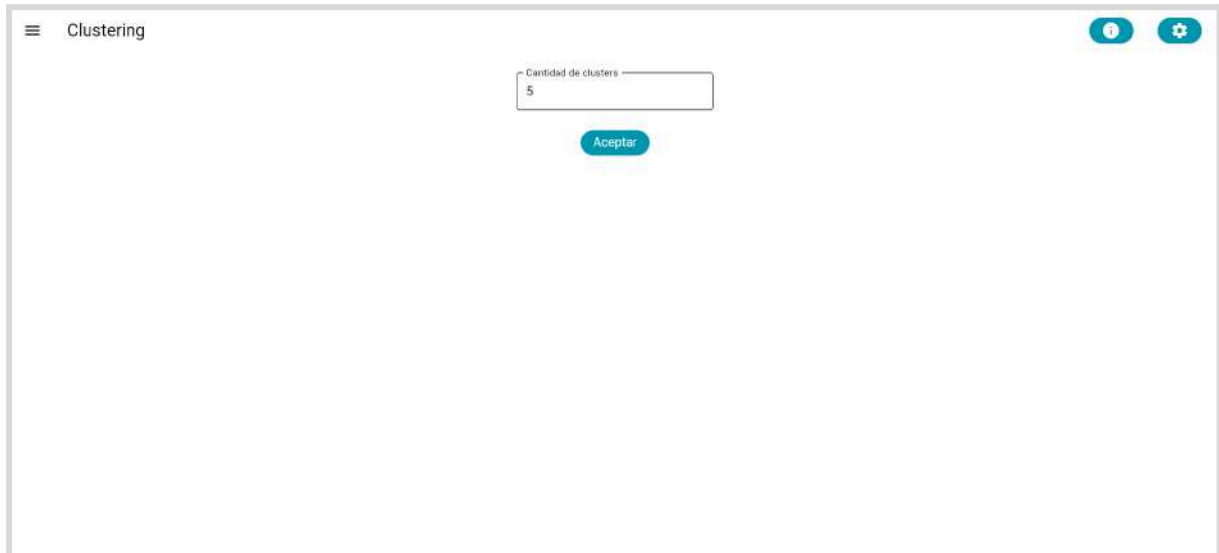


El botón descarga una imagen en formato .png de la grilla de hexágonos junto con las etiquetas que se muestran en el lateral.



## Opción Clustering

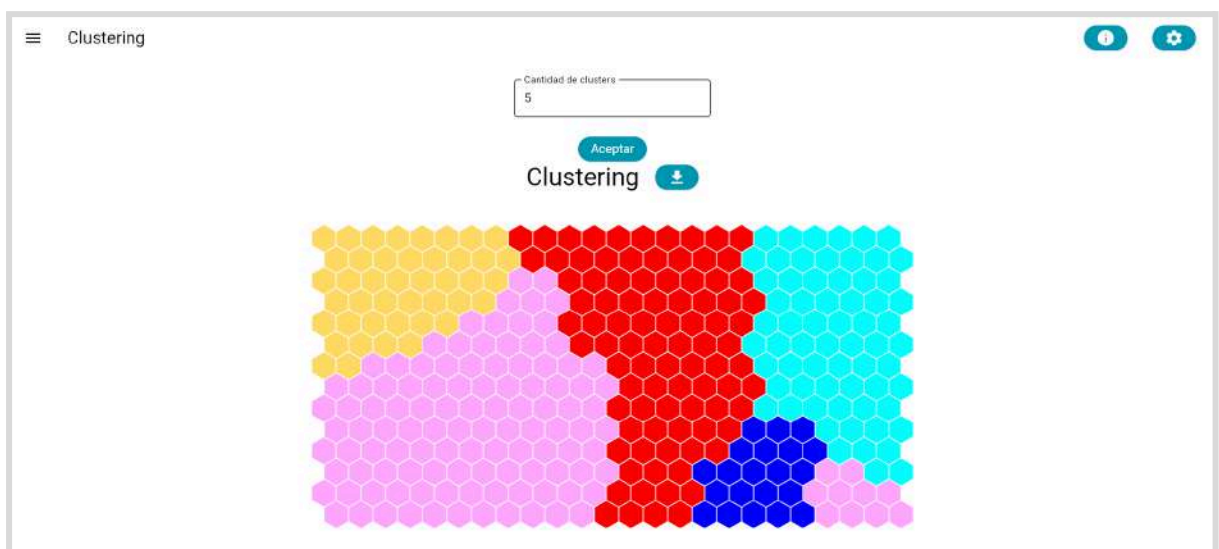
La pantalla de Clustering nos permite agrupar las neuronas del SOM en *clusters*.



Cantidad de clusters

El campo permite ingresar la cantidad de clusters

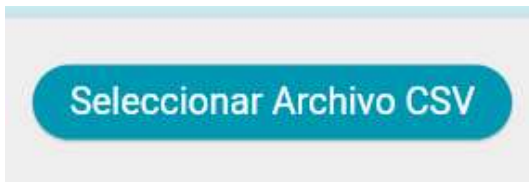
Al presionar Enviar, devuelve el SOM dividido en la cantidad de clusters indicada:





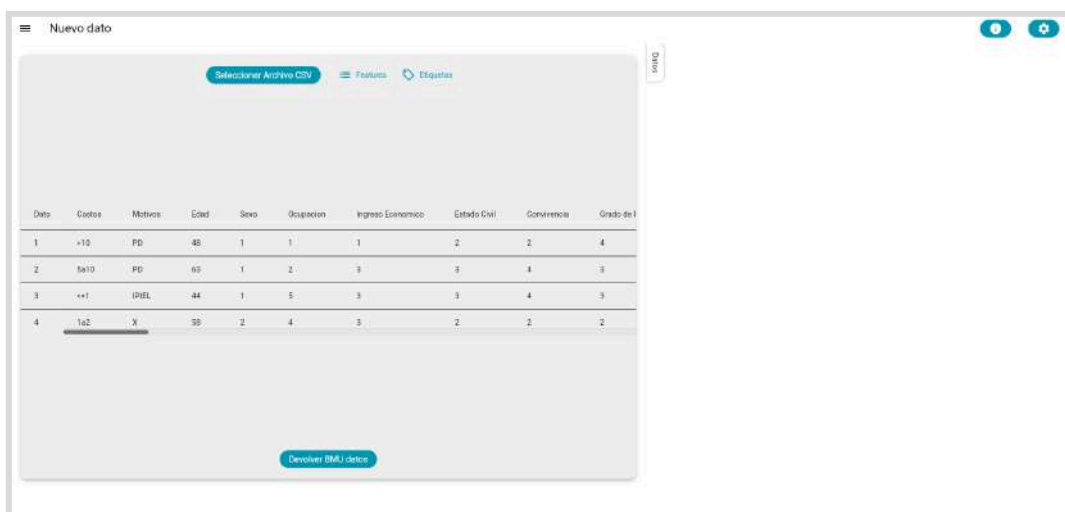
## Opción Nuevos datos

La pantalla Nuevos datos permite introducir datos nuevos que tengan las mismas dimensiones y características que los utilizados en el entrenamiento para clasificarlos dentro del SOM.



Para cargar el archivo de datos, hacer clic en el botón **Seleccionar Archivo CSV**. A continuación, se abrirá un explorador de archivos en el dispositivo, donde deberás seleccionar el archivo CSV que contenga los datos.

Al seleccionar el archivo podremos verlo en pantalla:





Al igual que en la carga de datos, se deben seguir los siguientes pasos:

 Features

Seleccionar columnas a utilizar. Las columnas que no estén seleccionadas no se tendrán en cuenta en el entrenamiento.

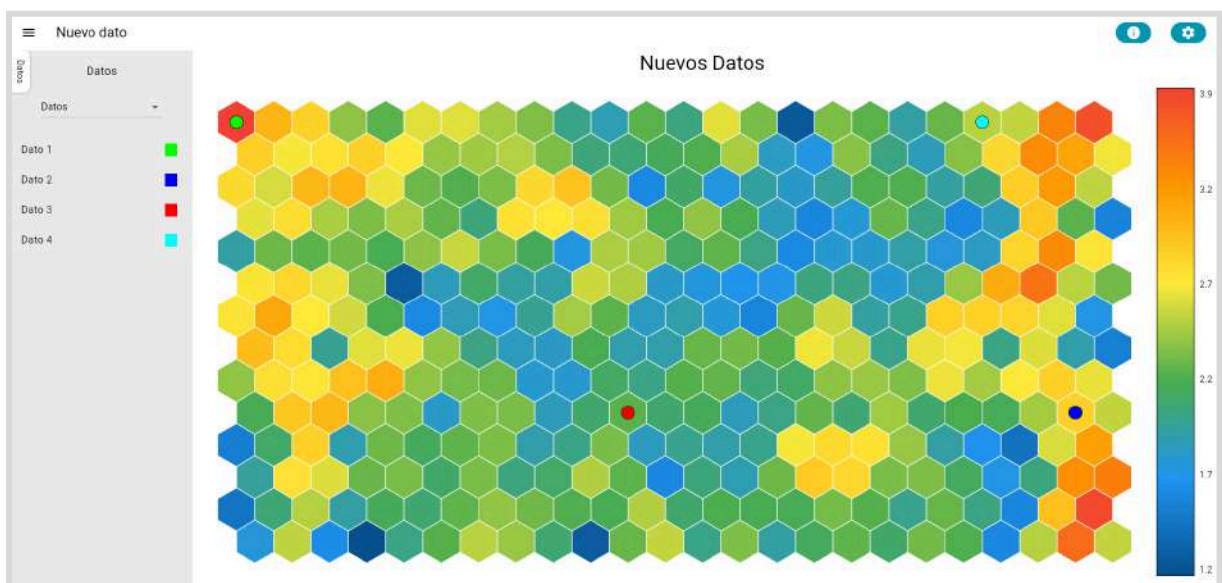
 Etiquetas

Seleccionar también las columnas que funcionan como etiquetas, de modo que puedan ser visualizadas como tales en el sistema.

 Devolver BMU datos

Presionar boton Devolver BMU datos para traer las BMU de los datos cargados.

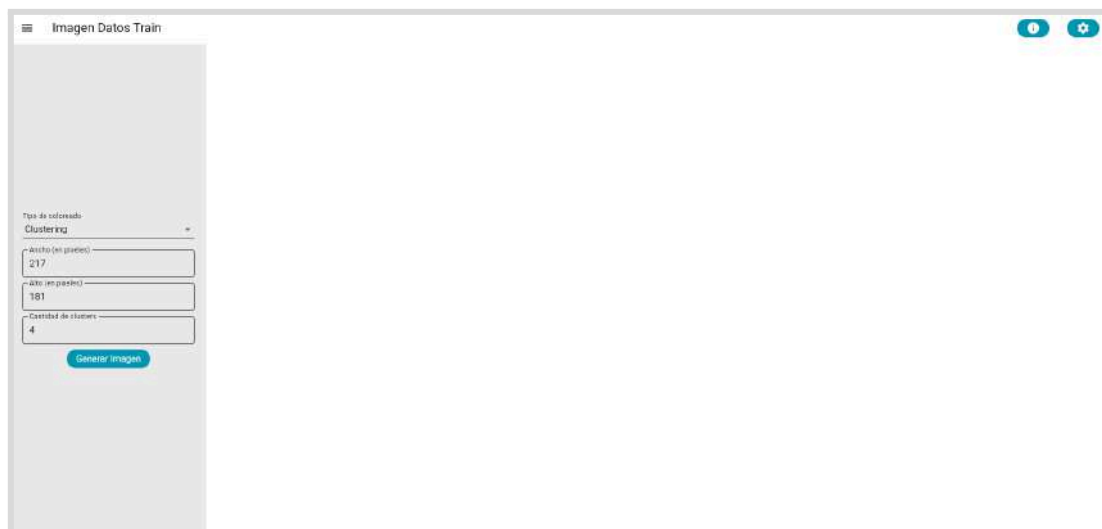
Una vez que la api responde podremos ver la respuesta. Las neuronas del SOM se muestran con los datos marcados en su BMU, y a la izquierda se lista el color asignado a cada dato para facilitar su identificación:





## Opción Imagen Datos Train

La pantalla Imagen Datos Train es la que nos permite utilizar la funcionalidad de segmentación de imágenes, generando una imagen a partir de los datos de entrada.



El campo de opciones “Tipo de Coloreado” permite seleccionar entre las opciones de segmentación: Clustering o Coloreado continuo



## Opción Clustering

Cuando la opción Clustering está seleccionada, aparecen los siguientes campos:

El campo Ancho permite ingresar la cantidad de píxeles de ancho que tiene la imagen.

El campo Alto permite ingresar la cantidad de píxeles de alto que tiene la imagen.

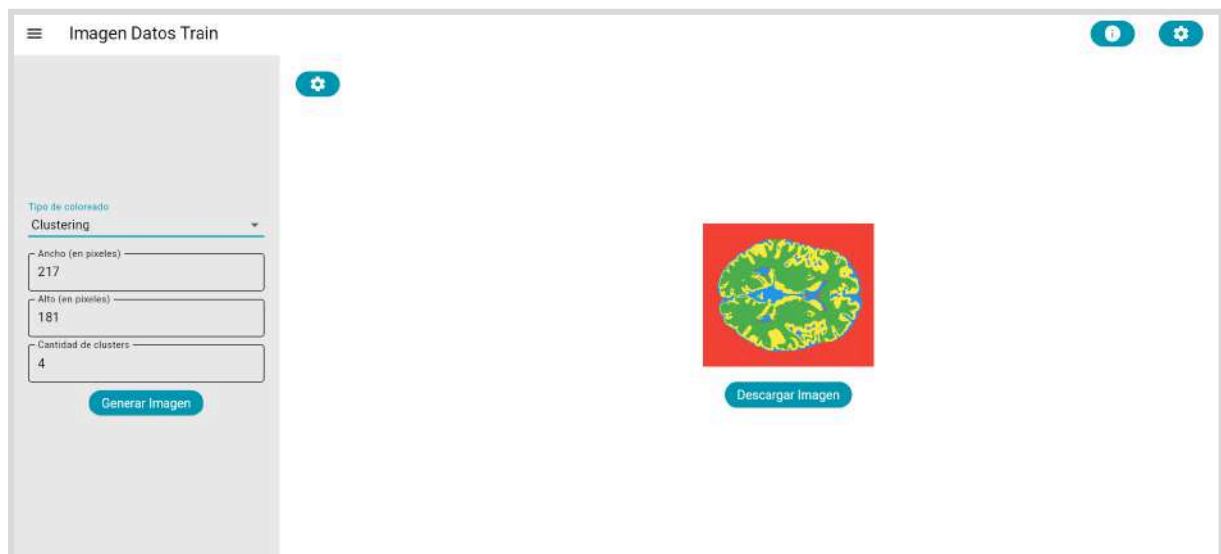
  

El campo Cantidad de clusters permite ingresar la cantidad de clusters en los que se quiere dividir la imagen

**Generar Imagen**

Presionar Generar imagen para que el sistema la genere

A continuación se mostrará la imagen generada en pantalla:

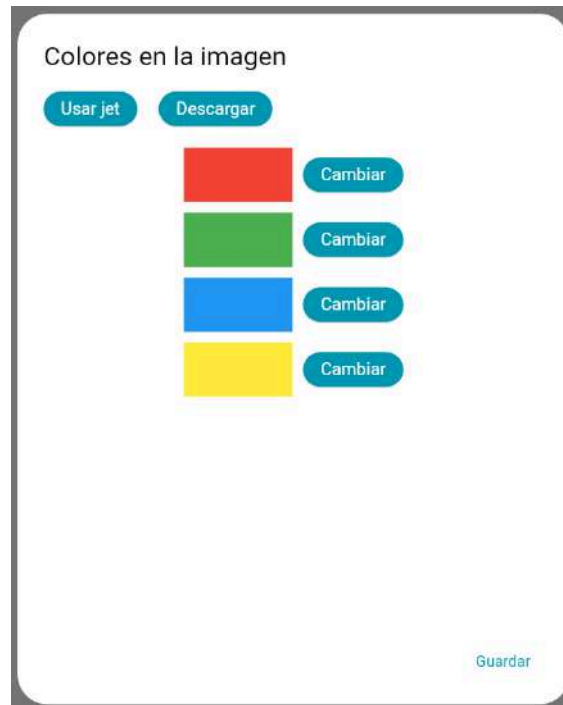


**Descargar Imagen**

El boton Descargar Imagen permite descargar la imagen generada en formato .png

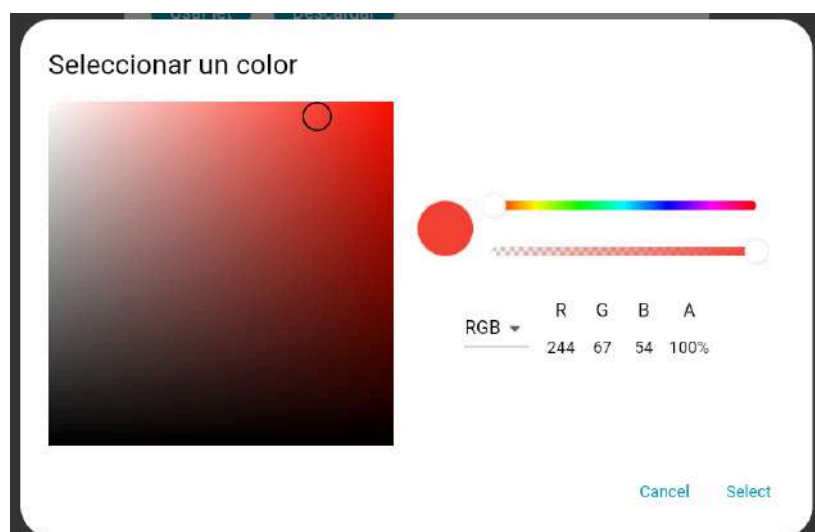


Al presionar el boton de configuración se abrirá el siguiente dialog



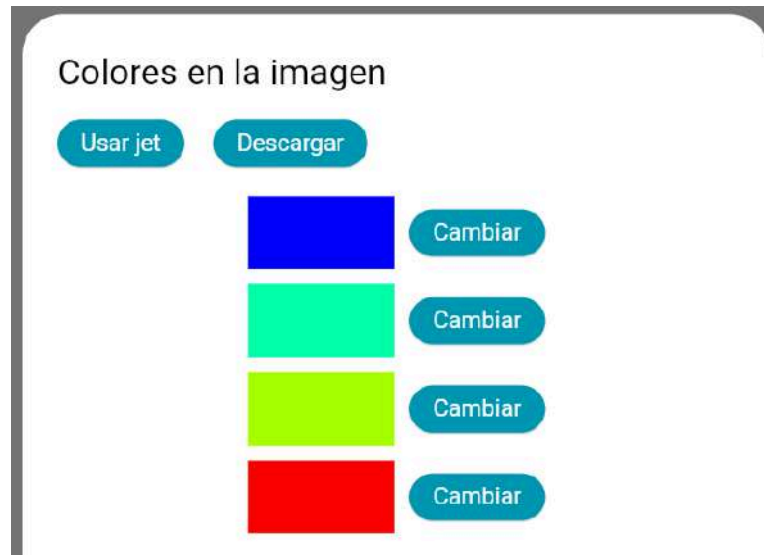
**Cambiar**

Con los botones Cambiar, es posible cambiar los colores utilizados en la imagen, mediante un selector de color:



### Usar jet

El boton usar jet permite utilizar el mapa de colores jet (una secuencia de colores que va desde el azul oscuro hasta el rojo oscuro pasando por una transición de colores que incluye azul, verde, amarillo y rojo).



### Descargar

El boton descargar permite descargar en formato txt los clusters a los que pertenece cada pixel.



## Opción Coloreado Continuo

Imagen Datos Train

Tipo de coloreado  
Coloreado Continuo

Ancho (en pixeles)  
217

Alto (en pixeles)  
181

Generar Imagen

Ancho (en pixeles)  
217

El campo Ancho permite ingresar la cantidad de píxeles de ancho que tiene la imagen.

Alto (en pixeles)  
181

El campo Alto permite ingresar la cantidad de píxeles de alto que tiene la imagen.

Generar Imagen

Presionar Generar imagen para que el sistema la genere

A continuación se mostrará la imagen generada en pantalla:

Imagen Datos Train

Tipo de coloreado  
Coloreado Continuo

Ancho (en pixeles)  
217

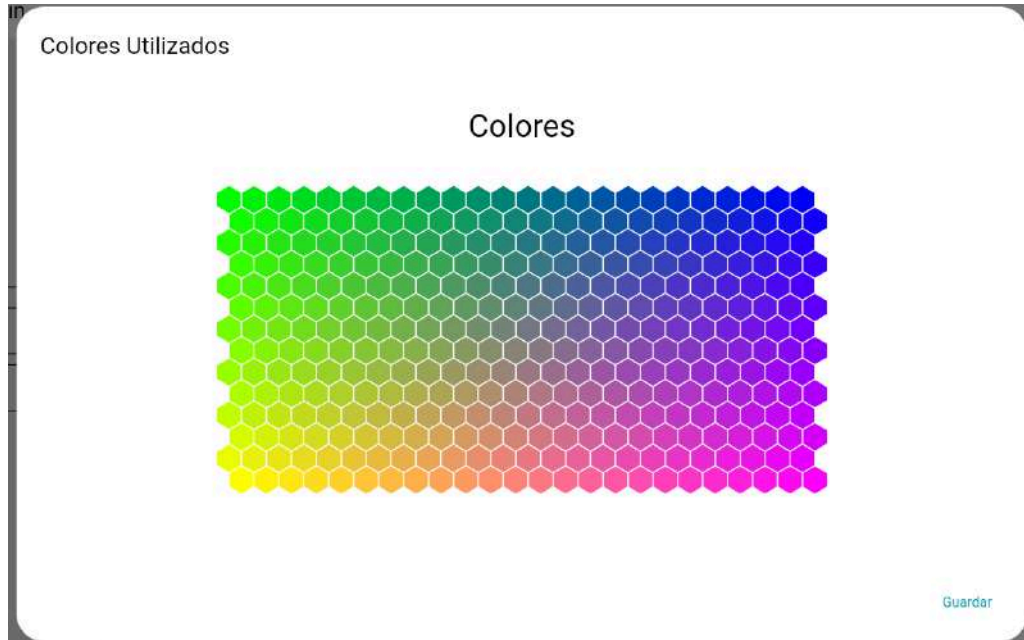
Alto (en pixeles)  
181

Generar Imagen

Descargar Imagen



El boton de configuración, permite ver la grilla de SOM con los colores que se le asignaron a cada neurona.





## Opción Imagen Nueva

La pantalla Imagen Nueva permite utilizar la funcionalidad de segmentación de imágenes con datos correspondientes a una imagen diferente a los utilizados durante el entrenamiento.

The screenshot shows the 'Imagen Nueva' interface. At the top left is a hamburger menu icon and the title 'Imagen Nueva'. At the top right are two circular icons: one with an information symbol and one with a gear symbol. Below the title is a dropdown menu labeled 'Tipo de coloreado' with 'Clustering' selected. A teal button labeled 'Seleccionar Archivo CSV' is positioned below the dropdown. Underneath are three input fields: 'Ancho (en pixeles)' with the value '217', 'Alto (en pixeles)' with the value '181', and 'Cantidad de clusters' with the value '4'. At the bottom of this section is another teal button labeled 'Generar Imagen'.

**Seleccionar Archivo CSV**

El botón **Seleccionar Archivo CSV**. abrirá un explorador de archivos en el dispositivo, que permite seleccionar el archivo CSV que contenga los datos.

This screenshot shows the same 'Imagen Nueva' interface as the previous one, but with additional elements. Below the 'Seleccionar Archivo CSV' button, there are two smaller buttons: 'Features' with a list icon and 'Etiquetas' with a tag icon. Below these is a confirmation message: 'El archivo Datos-Resonancia-TEST2-Contenedor.csv cargó correctamente.' The input fields for 'Ancho (en pixeles)', 'Alto (en pixeles)', and 'Cantidad de clusters' remain the same, with values 217, 181, and 4 respectively. The 'Generar Imagen' button is still at the bottom.



Los botones Features y Etiquetas funcionan de la misma forma que la carga principal de datos.

Las demás funcionalidades relacionadas con imágenes, como los campos de Ancho y Alto, los tipos de coloreado, entre otros, funcionan de la misma manera que en la pantalla de Imagen Datos Train.

