

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA INFORMÁTICA

**Aplicación de técnicas de
preservación de la privacidad a
anonimización de datos en registros
para el análisis de secuencias**

AUTOR: Rosso Giner, Rafael

Puerto Real, octubre de 2023

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA INFORMÁTICA

**Aplicación de técnicas de
preservación de la privacidad a
anonimización de datos en registros
para el análisis de secuencias**

DIRECTOR:

Juan Manuel Dodero Beardo
Manuel Palomo DuarteAUTOR:
Rafael Rosso Giner

Puerto Real, octubre de 2023

Declaración personal de autoría

Rafael Rosso Giner con DNI 45382412-P, estudiante del Grado en Ingeniería Informática en la Escuela Superior de Ingeniería de la Universidad de Cádiz, como autor de este documento académico titulado “Aplicación de técnicas de preservación de la privacidad a anonimización de datos en registros para el análisis de secuencias” y presentado como Trabajo Final de Grado.

DECLARO QUE

Es un trabajo original, que no copio ni utilizo parte de obra alguna sin mencionar de forma clara y precisa su origen tanto en el cuerpo del texto como en su bibliografía y que no empleo datos de terceros sin la debida autorización, de acuerdo con la legislación vigente. Asimismo, declaro que soy plenamente consciente de que no respetar esta obligación podrá implicar la aplicación de sanciones académicas, sin perjuicio de otras actuaciones que pudieran iniciarse. En Puerto Real, a 2 octubre de 2023

Fdo: Rosso Giner, Rafael

Resumen

En los últimos años, el uso de la inteligencia artificial basada en análisis de datos y deep learning ha experimentado un notable incremento, gracias al aumento en la potencia de cómputo en sistemas modernos y a la generación masiva de datos. Para impulsar la investigación y el avance en estos campos, es fundamental llevar a cabo la recopilación, etiquetación y tratamiento de grandes volúmenes de datos.

Sin embargo, nos enfrentamos a desafíos significativos, ya que muchos de estos datos son de carácter personal o su divulgación podría resultar dañina para las personas a las que involucra. Esto dificulta la capacidad de difusión de los datos en el ámbito de las ciencias abiertas y aumenta el riesgo de divulgación no autorizada.

En este contexto, este documento presenta una aplicación web innovadora que aborda esta problemática. La aplicación permite el análisis de datasets para obtener métricas acerca del riesgo de divulgación de los datos. Se utilizan tanto técnicas deterministas como técnicas basadas en mapas auto-organizado para llevar a cabo este análisis, brindando nuevas herramientas para mejorar la seguridad y privacidad de los datos.

Esto significa que el usuario puede aplicar medidas de anonimización o enmascaramiento de información sensible, lo que contribuye a la creación de fuentes de datos abiertas más seguras y protege la privacidad de las personas involucradas.

La importancia de esta aplicación web radica en su capacidad para facilitar la investigación y el trabajo con datos sensibles de manera ética y responsable. Al permitir una mayor protección de la privacidad, se fomenta la colaboración en el ámbito de las ciencias abiertas y se minimiza el riesgo asociado con la manipulación y divulgación de información delicada o reservada. Su enfoque en la seguridad y la privacidad es crucial para superar las barreras que impiden la difusión segura de datos sensibles en el ámbito científico y tecnológico.

Palabras clave: Anonimización de datos, Dataset, Aplicación Web, Self Organizing Maps

Índice general

Índice de figuras	viii
Índice de tablas	x
I Prolegómeno	1
1. Introducción	3
1.1. Motivación	3
1.2. Alcance y Objetivos	3
1.3. Glosario de términos	4
1.4. Organización del documento	4
2. Antecedentes	7
2.1. Contexto	7
2.2. Estado de la técnica	7
3. Plan de gestión de proyecto	11
3.1. Metodología de desarrollo	11
3.2. Tecnologías	11
3.3. Planificación del proyecto	11
3.4. Organización	15
3.5. Costes	15
3.6. Riesgos	16
3.7. Gestión de la configuración	17
3.8. Aseguramiento de calidad	17
II Desarrollo	18
4. Requisitos del Sistema	20
4.1. Objetivos del Sistema	20
4.2. Requisitos funcionales	20
4.3. Requisitos no funcionales	20
4.4. Requisitos de información	21
4.5. Análisis GAP	22
5. Análisis del Sistema	25
5.1. Modelo Conceptual	25
5.2. Modelo de Casos de Uso	25
5.3. Modelo de Interfaz de Usuario	27
5.4. Modelo de Comportamiento	28

6. Diseño del Sistema	33
6.1. Arquitectura del Sistema	33
6.1.1. Diseño de alto nivel	33
6.1.2. Diseño detallado	36
6.2. Parametrización del software base	42
6.3. Diseño Físico de Datos	44
6.3.1. Base de datos	44
6.3.2. Sistema de archivos	45
6.4. Diseño de la Interfaz de Usuario	45
6.4.1. Flask	45
6.4.2. Autenticación	46
6.4.3. Perfil	47
6.4.4. Página principal	48
6.4.5. Anonimización de datos	48
6.4.6. Análisis SOM	50
6.4.7. Optimización de hiper-parámetros	51
7. Construcción del Sistema	55
7.1. Entorno de Construcción	55
7.2. Código Fuente	55
7.2.1. Optimización en operaciones sobre grandes conjuntos de datos .	56
7.3. Scripts de Base de datos	57
7.3.1. Creación de la base de datos	57
7.3.2. Sincronización de la base de datos	57
8. Pruebas del Sistema	61
8.1. Estrategia	61
8.2. Pruebas Unitarias	61
8.3. Pruebas de Integración	61
8.4. Pruebas de Sistema	62
8.4.1. Pruebas Funcionales	62
8.4.2. Pruebas No Funcionales	62
8.5. Pruebas de Aceptación	62
9. Despliegue del Sistema	65
9.1. Arquitectura Física	65
9.2. Instrucciones de despliegue	65
9.2.1. Docker	65
9.2.2. Requisitos previos	66
9.2.3. Inventario de componentes	67
9.2.4. Procedimientos de instalación	67
9.2.5. Pruebas de implantación	68
9.3. Instrucciones para la operación del sistema y mantenimiento del nivel de servicio	69

III Epílogo	71
10. Conclusiones	73
10.1. Objetivos alcanzados	73
10.2. Lecciones aprendidas	73
10.3. Trabajo futuro	73
Información sobre Licencia	77
Bibliografía	79
IV Anexos	83
A. Manual de usuario	85
A.1. Introducción	85
A.2. Instalación	85
A.3. Uso del sistema	87
A.3.1. Perfil de usuario	87
A.3.2. Subir un dataset	87
A.3.3. Privatización de Datasets	88
A.3.4. Análisis SOM	88
B. Diagramas de diseño	91
B.1. Diagramas del modelo C4	91
B.2. Pantallas del sistema	98
C. Casos de Uso	103
C.1. Gestión de Usuarios	103
C.2. Datasets	109
C.3. Árboles de Taxonomía	112
C.4. Operaciones	116
D. Diagramas adicionales	121

Índice de figuras

3.1. Diagrama de Gantt	13
3.2. Comparación de Diagramas de Gantt	14
5.1. Modelo conceptual de datos	25
5.2. Caso de Uso “Subir un Dataset”	26
5.3. Caso de Uso “Análisis SOM”	27
5.4. Flujo de pantallas del sistema	28
5.5. Proceso Análisis SOM	29
5.6. Proceso subida de dataset	30
6.1. Vista principal	34
6.2. Servidor	35
6.3. Comparación de Datasets	37
6.4. Análisis de componentes principales	37
6.5. Actualización SOM	39
6.6. Coeficiente de Aprendizaje	39
6.7. Evaluación de resultados	40
6.8. Distancia entre nodos	41
6.9. Autenticación	47
6.10. Perfil de usuario	47
6.11. Página principal	48
6.12. Permutación de datos	49
6.13. Generalización de datos	49
6.15. Contraste de Hipótesis	50
6.16. Optimización de hiper-parámetros	51
6.14. Análisis SOM	53
B.1. Interfaz	91
B.2. Aplicación	92
B.3. Autenticación	92
B.4. Controlador de Procesos	93
B.5. Análisis Clásico	94
B.6. Análisis SOM	95
B.7. Computación Optimizada	96
B.8. Base de Datos	97
B.9. Pantalla de progreso	98
B.10. Selección de condiciones categóricas	98
B.11. Evaluación de contraste de Hipótesis	99
B.12. Anonimización mediante adición de Ruido	100
C.1. Caso de Uso “Acceder al servicio”	103
C.2. Caso de Uso “Crear una cuenta”	104
C.3. Caso de Uso “Editar nombre”	105

C.4. Caso de Uso “Editar email”	106
C.5. Caso de Uso “Editar contraseña”	107
C.6. Caso de Uso “Eliminar cuenta”	108
C.7. Caso de Uso “Subir un Dataset”	109
C.8. Caso de Uso “Eliminar un Dataset”	110
C.9. Caso de Uso “Descargar un Dataset”	111
C.10. Caso de Uso “Subir un Árbol de Taxonomía”	112
C.11. Caso de Uso “Descargar un Árbol de Taxonomía”	113
C.12. Caso de Uso “Editar un Árbol de Taxonomía”	114
C.13. Caso de Uso “Eliminar un Árbol de Taxonomía”	115
C.14. Caso de Uso “Análisis SOM”	116
C.15. Caso de Uso “Privatización de Datos”	117
C.16. Caso de Uso “Comparación de Datos”	118
D.1. Ejemplo de árbol de taxonomía	121

Índice de tablas

3.1. Desglose de Costes	15
3.2. Riesgos	16
4.1. Análisis GAP	22
D.1. Paquetes de Python	122

Parte I

Prolegómeno

1. Introducción

1.1. Motivación

Hoy en día, sistemas de gran volúmenes de datos sensibles como los centros sanitarios se han convertido en el principal objetivo de ataques y fugas de información ([Seh, 2020](#)). Debido al progreso de las tecnologías basadas en datos y recopilación masiva de estos, se hace necesaria una forma restringir la trazabilidad de los datos para proteger la privacidad de los usuarios.

Además, uno de los puntos más relevantes en el ámbito de la ciencia abierta es la publicación de datos ([EU, 2020](#)). Resulta cada vez más necesaria la evaluación de grandes volúmenes de datos (provenientes habitualmente de diversos orígenes) que puedan ser compartidos de forma segura y eficiente. A pesar de ello, estos datos no siempre pueden publicarse como es el caso de situaciones médicas como el COVID-19 donde los datos recogidos son altamente sensibles para el usuario. Esta problemática es más acentuada en el caso de secuencias de información sobre procesos (por ejemplo registros logs de interacción de una persona con un sistema informático).

Actualmente, los métodos de análisis manual de secuencias de información presentan limitaciones para llevar a cabo un análisis detallado de grandes conjuntos de datos. Por ejemplo, la minería de procesos (*Process Mining*) es un conjunto de técnicas de análisis de secuencias que permite extraer conocimiento de registros de eventos de manera automática ([Caballero-Hernández et al., 2019](#)). La preservación de la privacidad en datos es una serie de técnicas que permiten publicar datos sensibles de forma que no sea posible trazar e identificar a los individuos que conforman dichos datos. Mediante la anonimización de un dataset, es posible obtener un dataset transformado cuyos resultados estadísticos sean igual de válidos que el dataset original ([Rodriguez-Garcia et al., 2019](#)). Estas técnicas se han usado con éxito en datos tabulados, pero hasta la fecha no se ha estudiado su aplicación a secuencias de información.

1.2. Alcance y Objetivos

El objetivo principal del proyecto es desarrollar una aplicación y un conjunto de algoritmos para facilitar la publicación abierta de registros de datos de forma que se garantice la preservación de la privacidad. Entre las funciones que se ofrecen existen varios algoritmos clásicos y el análisis mediante redes SOM, que incluye un módulo de árboles de taxonomía para dotar de relación semántica los campos categóricos.

Se buscará no solo analizar, sino también poder impulsar algoritmos de generalización de datos basados en la redes SOM para poder inferir la forma de la nube de datos a través de un aprendizaje iterativo sobre los datos proporcionados.

El proyecto buscará estudiar las tecnologías que resulten más adecuadas para la implementación de una interfaz de usuario que permita automatizar la aplicación de estos algoritmos sobre unos datos concretos de forma sencilla e intuitiva. Además debe tener la capacidad de compartir y operar sobre datos publicados por otros usuarios, de forma que sea más sencillo comparar resultados con otros investigadores.

1.3. Glosario de términos

IND: Inclusion Dependency.

INDD: Inclusion Dependency Discovery.

MEC: Ministerio de Educación y Ciencia.

SOM: Self Organizing Map.

1.4. Organización del documento

La sección I se centrará en el apartado de gestión de proyectos, detallando el contexto tecnológico, la metodología aplicada y la planificación del proyecto. Se incluirán además apartados especificando los costes, riesgos y la organización del mismo.

La sección II desarrolla en profundidad el análisis, diseño e implementación del producto de software presentado. Dentro de esta sección se especificarán los requisitos identificados durante la fase de análisis, donde se aportarán estudios comparativos entre la solución propuesta y otras existentes en el mercado.

En el apartado de diseño se ilustrará con diagramas, acordes al modelo C4 ([Brown, 2022](#)), los diferentes componentes de software que existen en el programa y las relaciones entre ellos. Esta sección explicará los algoritmos subyacentes a las operaciones de la aplicación, junto a los detalles técnicos del código que se entrega junto a este documento. Finalmente, se incluye un apartado para especificar la estrategia de depuración que se ha seguido durante el desarrollo y la documentación necesaria para el despliegue de la aplicación.

En la sección III se exponen las conclusiones obtenidas durante el desarrollo del proyecto; incluyendo los objetivos alcanzados y el aprendizaje en el campo de estudio. Adicionalmente, se realizan propuestas de mejora y posibles trabajo futuro.

2. Antecedentes

2.1. Contexto

El problema central que aborda esta investigación se refiere a la protección de la privacidad al analizar datos provenientes de fuentes de aprendizaje distribuido. El proyecto se enfoca en dos cuestiones clave relacionadas con la información personal: en primer lugar, la combinación de datos multidimensionales; y en segundo lugar, la publicación de conjuntos de datos personales de manera accesible, interoperable y reutilizable, asegurando la privacidad de los individuos.

Por un lado, las dependencias de inclusión (IND) entre conjuntos de datos implican que cualquier combinación de valores en un conjunto de datos multidimensional esté contenida en otro conjunto. Sin embargo, este problema de descubrimiento de dependencias de inclusión (INDD) no es directamente aplicable cuando se trata de datos representados en valores numéricos, como ocurre con coordenadas y medidas de actividades humanas. Por lo tanto, se investigarán versiones de las relaciones de IND que tengan en cuenta las distribuciones de probabilidad de los datos analizados al verificar la igualdad de datos multidimensionales resultantes de la fusión de diversas fuentes, como dispositivos de interacción multimodal y sistemas de información en entornos inteligentes de aprendizaje.

Por otro lado, la publicación de datos con preservación de la privacidad (PPDP) tiene como objetivo garantizar que los datos personales en un conjunto sean indistinguibles del resto de datos, dificultando así los intentos de identificar a los individuos de origen de cada microdato. Esto se logra mediante técnicas de anonimización, diversificación y privacidad diferencial, entre otras. En este proyecto, se investigarán versiones semánticas de las técnicas de PPDP aplicadas a datos nominales y categóricos, para los cuales las técnicas tradicionales basadas en estadísticas numéricas no son apropiadas.

Recientemente, se han desarrollado nuevos algoritmos deterministas muy prometedores para el estudio de dependencias entre datos, haciendo uso de grafos para buscar relaciones entre los atributos de un dataset ([Álvarez Ayllón et al., 2022](#)).

Con el objetivo de explorar alternativas, en este documento se investigarán técnicas de aprendizaje basadas en redes SOM y algoritmos de agrupamiento que hagan uso de árboles de taxonomía para tratar con datos categóricos de una manera más efectiva. Se implementarán las técnicas resultantes mediante una aplicación web que facilite su uso sobre múltiples conjuntos de datos de manera intuitiva y accesible.

2.2. Estado de la técnica

En los últimos años se han realizado numerosas investigaciones relacionadas con las técnicas de privatización de datos. Las técnicas de publicación de datos con preser-

vación de la privacidad (PPDP) son una forma de permitir compartir datos anónimos garantizando la protección contra la divulgación de la identidad de un individuo ([Zaman y Obimbo, 2014](#)).

Hoy en día, la publicación de datos se ha convertido en rutina para múltiples individuos, empresas y organizaciones. Estos métodos tienen como objetivo mantener los datos subyacentes útiles basándose en la preservación de la privacidad, creando grandes oportunidades para modelos de aprendizaje basados en el conocimiento y la información. Los recientes avances en el campo han mejorado significativamente la toma de decisiones, especialmente en campos como la medicina donde un gran conjunto de los datos recopilados son de carácter sensible ([Rashid y Yasin, 2015](#)).

En el ámbito se pueden encontrar también estudios comparativos acerca de la eficiencia de múltiples algoritmos que tratan de obtener la k-anonimidad aplicada a datos numéricos, categóricos y la combinación de ambos ([Ji-Won Byun y Li, 2003](#)). En la actualidad, existe una gran cantidad de técnicas de privatización que siguen estrategias muy diversas en función de los recursos disponibles, tipo de datos recopilados y objetivos que se quieren alcanzar ([Arcolezi, 2022](#)).

En este documento se desarrollará la implementación de algunas de estas técnicas, con el fin de ofrecer una plataforma de código libre que permita la aplicación de estos algoritmos a datasets de cualquier tipo de forma sencilla y accesible para todo investigador o analista que quiera preservar la privacidad. Además, se incluyen varias formas de análisis para estimar el riesgo de divulgación presente en un conjunto de datos.

3. Plan de gestión de proyecto

3.1. Metodología de desarrollo

La planificación del proyecto se ha dividido en cuatro categorías:

- Gestión de proyecto: Recoge todas las tareas de análisis y documentación del proyecto desarrollado. Esta engloba también los hitos principales
- Implementación: Recoge todas las tareas necesarias para la implementación del software desarrollado.
- Desajustes temporales: Sirve para alojar las tareas emergentes que hayan surgido durante el desarrollo del proyecto. Al comienzo de la planificación se estimó una holgura temporal de cuatro semanas para que, en circunstancias sobrevenidas, fuese posible finalizar en tiempo el proyecto.
- Despliegue: Define la lista de tareas necesarias para desplegar tanto el software como el código fuente de forma pública.

Las tareas se han organizado en función de su prioridad y dependencia, siguiendo una estrategia ágil de sprints semanales.

3.2. Tecnologías

Para implementar el proyecto, se ha decidido utilizar Python como lenguaje de programación debido a la amplia disponibilidad de paquetes para el cálculo de operaciones matriciales e integración de arquitecturas web. Específicamente, hemos optado por emplear el framework Flask de Python (sección 6.4.1) para desarrollar la aplicación web. Flask es uno de los frameworks más versátiles y robustos disponibles en el mercado, lo que facilita la implementación intuitiva de sistemas externos, como la autenticación de usuarios y el manejo de peticiones web.

Para mitigar el impacto en el rendimiento de interpretadores como Python, se ha utilizado el paquete Numpy para operar con grandes volúmenes de datos. Adicionalmente, se han optimizado las funciones principales de cómputo mediante el paquete Numba, para poder precompilar el cálculo y aumentar exponencialmente el rendimiento de la aplicación.

3.3. Planificación del proyecto

En el diagrama de Gantt 3.1 se puede observar la planificación detallada de las tareas a que se han completado en la duración del proyecto. En la figura 3.2 se ilustra

una comparación visual entre el tiempo estimado y el invertido en cada una de las tareas del proyecto.

Hay que destacar que ha sido necesario añadir a la planificación tres tareas emergentes debido a varios imprevistos que han surgido durante la fase de desarrollo de la aplicación. Para empezar, debido a las limitaciones técnicas del framework Bottle, fue necesario cambiar a Flask y reimplementar toda la interfaz entre la web y las funciones del sistema. Por otro lado, el motor de base de datos seleccionado durante el prototipado inicial, SQLite, fue desecharado durante el desarrollo debido a que no está preparado para gestionar conexiones simultáneas de manera efectiva. Finalmente, durante el despliegue de la aplicación en un contenedor de Docker, se identificaron varios ajustes necesarios al código que interactúa con la base de datos.

Debido al desajuste temporal en las tareas planificadas y los imprevistos mencionados, se decidió retrasar la fecha de finalización del proyecto para poder implementar y pulir todas las funcionalidades del sistema.

Anonymization

Read-only view, generated on 04 Oct 2023

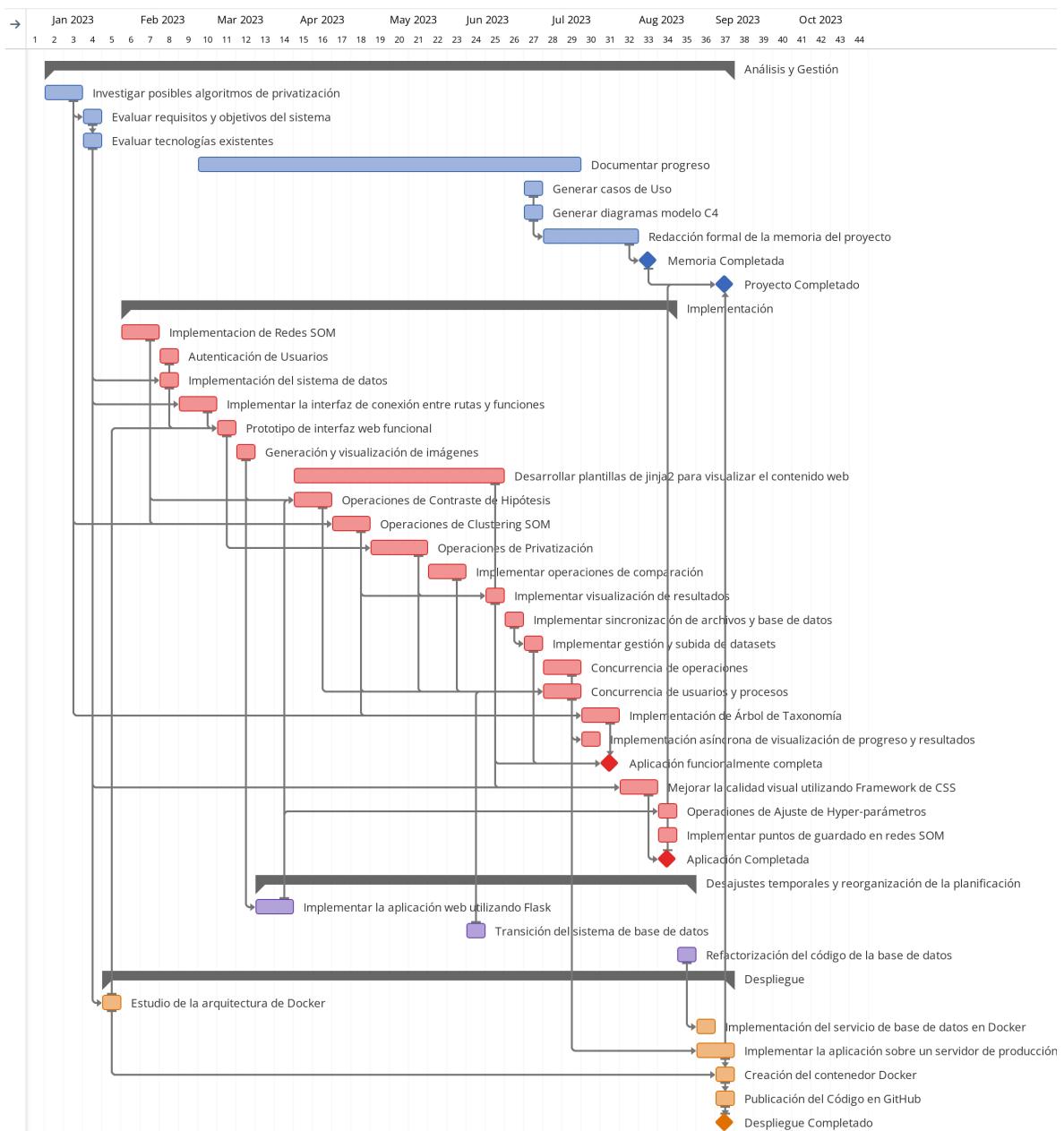


Figura 3.1: Diagrama de Gantt.

Anonymization

Read-only view, generated on 05 Oct 2023

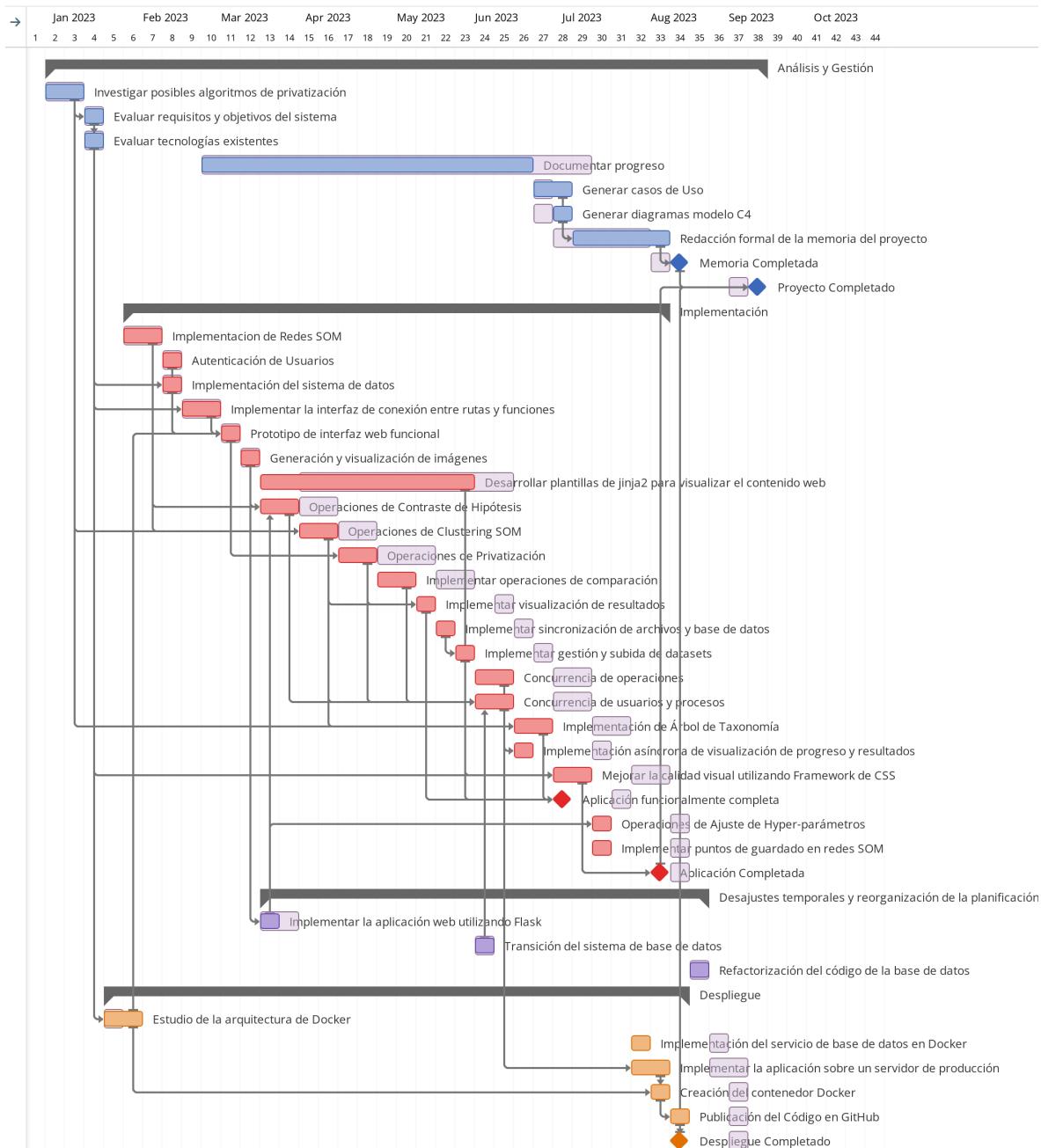


Figura 3.2: Comparación de Diagramas de Gantt.

En segundo plano se superpone a las fechas estimadas, las fechas realmente invertidas en cada tareas. Esto ilustra el desplazamiento en la planificación debido a los desajustes temporales.

3.4. Organización

El proyecto está asociado a una beca de colaboración concedida por el Ministerio de Educación y Formación Profesional para el curso 2022-2023.

El principal desarrollador del documento presente y el proyecto asociado al mismo es quién suscribe, Rafael Rosso Giner, con la supervisión y dirección de los doctores Juan Manuel Dodero Beardo y Manuel Palomo Duarte.

Para poder llevar a cabo esta tarea ha sido necesario asumir múltiples roles característicos de cualquier proyecto de ingeniería de software, comenzando por el análisis de requisitos. Este rol tiene la responsabilidad de recoger los diferentes requisitos funcionales y no funcionales del sistema, según de los objetivos propuestos para el proyecto.

Una vez formalizados estos requisitos, se inicia la fase de diseño y prototipado, para poder estimar el tiempo y los recursos necesarios para implementar las diferentes partes del proyecto. Durante este periodo, se investigan además las diferentes tecnologías disponibles en el mercado actual y comparativa del diseño propuesto respecto a otras aplicaciones similares. Finalmente se inicia la fase de implementación donde se construye el software diseñado, utilizando las herramientas óptimas para ello.

Respecto al software y herramientas empleadas, se ha utilizado GitHub como repositorio online de código, PyCharm como IDE principal para el desarrollo del proyecto y un ordenador personal para poder llevar a cabo la implementación. Adicionalmente, se ha utilizado el plugin de SonarLint¹ para verificar la integridad del código y poder realizar análisis de integración continua sobre la aplicación.

3.5. Costes

Tabla 3.1

Desglose de Costes

Desglose de Costes	
Concepto	Coste
Técnico Investigador Graduado	10288€
Costes Indirectos (10 %)	1028€
Servidor	350€
Licencias de Software	0€
Coste de Realización	11666€

El presupuesto anterior se basa en las siguientes estimaciones:

- Técnico Investigador Graduado: Siguiendo la normativa presupuestaria de la Universidad de Cádiz respecto a los costes de Capítulo VI (6 meses) ([UCA, 2023](#)).

¹Sonar Lint: <https://www.sonarsource.com/products/sonarlint/>

El periodo se ha estimado a partir del número de horas por crédito ($25h * 18$ ECTS), junto a la estimación de las horas de trabajo vinculadas a la beca MEC ($3h/dia$ durante 7.5 meses). Obteniéndose un total de 900 horas o lo equivalente, 6 meses de trabajo a tiempo completo.

- Costes Indirectos: 10 % del coste de personal.
- Servidor: Se recomienda un servidor con un hardware de características similares a las siguientes.
 - Intel Xeon E-2288G.
 - 32 GB RAM DDR4.
 - 4 TB SSD.

Se estima que la vida útil del servidor es de tres años, lo que coincidiría su amortización, incluyéndose en el presupuesto el equivalente a seis meses de la misma.

- Licencias de Software: El proyecto ha sido construido en su totalidad utilizando software gratuito y de código libre o abierto. Las herramientas software empleadas ([3.2](#)) son también gratuitas o con licencia sin coste para estudiantes e investigadores.

Al coste de realización, habría que agregarle el importe anual del mantenimiento del software más allá del periodo de realización.

3.6. Riesgos

A continuación se detalla una síntesis de los principales riesgos del proyecto, asociados a su probabilidad de ocurrencia e impacto. Asimismo se adjunta un plan de mitigación para cada uno de ellos.

Riesgos del proyecto		
Descripción	Probabilidad	Impacto
Definición de Objetivos	Baja	Alto
Plazo límite	Baja	Alto
Docker	Media	Media
Requisitos	Alta	Bajo

Tabla 3.2

Riesgos

Descripción

- Definición de Objetivos: El objetivo del proyecto no queda bien definido durante la fase de planificación.
- Plazo límite: No se cumplen con los objetivos del proyecto dentro del plazo.

- Docker: No es posible implementar un contenedor Docker ² para el proyecto
- Requisitos: El software no integra todos los puntos propuestos durante la planificación.

Plan de Mitigación.

- Definición de Objetivos: Comenzar desde el inicio a involucrar a los directores del proyecto para definir claramente cuáles son los objetivos del proyecto y qué se pretende lograr con el desarrollo del mismo.
- Plazo límite: Realizar una planificación objetiva del tiempo disponible y centrar todos los esfuerzos durante las primeras fases en obtener un prototipo funcional que pulir en el futuro.
- Docker: Realizar un estudio preventivo de los requisitos de un contenedor de Docker para diseñar la aplicación siguiendo una estructura similar.
- Requisitos: Definir una lista de tareas que relacione el tiempo necesario y el impacto que tiene sobre el producto final.

3.7. Gestión de la configuración

Como ya se ha mencionado, el proyecto en su totalidad se publicará como código abierto en la plataforma de GitHub ³. El control de versiones y lanzamientos oficiales será gestionado desde dicha plataforma para facilitar la distribución del programa.

3.8. Aseguramiento de calidad

Para mejorar la calidad del código producido, se ha utilizado una combinación de SonarLint y la corrección de sintaxis especializada de PyCharm. Gracias a esta metodología, se ha podido generar un código que sigue los estándares de estilo indicados en PEP-8 ⁴ y PEP 257 ⁵, según se indica en la web oficial de Python. Además, se evitan flujos lógicos incoherentes, redundancia de variables y comportamientos no definidos.

²Docker: <https://www.docker.com/>

³GitHub: <https://github.com/RafaelRossoGiner/Anonymization>

⁴PEP 8: <https://peps.python.org/pep-0008/>

⁵PEP 257: <https://peps.python.org/pep-0257/>

Parte II

Desarrollo

4. Requisitos del Sistema

4.1. Objetivos del Sistema

Los objetivos del sistema son los siguientes:

- Proporcionar una interfaz gráfica e intuitiva para anonimización de datos y análisis de los mismos.
- Mejorar la privacidad de los datos para facilitar procesos de investigación en ciencias abiertas, disuadir del uso malicioso de los mismos y proteger a los usuarios.
- Buscar vectores de riesgo en datos mediante el uso de mapas autorganizativos, facilitando la identificación de patrones y tendencias en nubes de datos complejas.

4.2. Requisitos funcionales

El sistema ofrece la siguiente funcionalidad a sus usuarios.

- Subida de archivos a un servidor web.
- Visualización de archivos de datos subidos al servidor web.
- Operaciones de privatización de datos.
 - Data-swapping.
 - Aplicación de ruido.
 - Reducción de características.
- Análisis de riesgo de la privacidad de los datos.
 - Cálculo de métricas de unicidad de los datos.
 - Análisis de datos para detectar características críticas de privacidad.
 - Reducción de características.

4.3. Requisitos no funcionales

Adicionalmente se plantean estos requisitos con el objetivo de garantizar la calidad del producto desarrollado.

- El software debe ser eficiente a nivel computacional, optimizando procesos, aprovechando los recursos hardware disponibles y evitando el uso de lenguajes interpretados para operaciones complejas.

- La aplicación web debe ser intuitiva y sencilla de utilizar.
- La aplicación web debe ser atractiva visualmente y adaptable a múltiples dispositivos o configuraciones de pantalla.
- El software debe ser portable y modular.
- El software debe ser de código abierto y facilitar la modificación o integración mediante un repositorio público.
- El software debe ser sencillo de instalar y configurable para múltiples sistemas o entornos.

4.4. Requisitos de información

Respecto a la gestión de los datos del sistema, se definen los siguientes requisitos:

- Gestionar usuarios. Un usuario debe poseer:
 - Identificador único.
 - Nombre.
 - Correo.
 - Contraseña. Esta debe ser encriptada para mejorar la seguridad del sistema.
 - Lista de Datasets asociados a este usuario.
- Gestionar datasets.
 - Identificador único.
 - Nombre.
 - Nombre del archivo en el sistema.
 - Identificador del usuario dueño del dataset.
 - Visibilidad. Un dataset puede ser público o privado.
 - Modo de tratamiento sobre los campos categóricos.
 - Dimensionalidad del dataset.
 - Número de registros en el dataset.
 - Lista de puntos de entrenamiento generados a partir de este dataset.
- Gestionar puntos de entrenamiento.
 - Identificador único.
 - Parametro de anonimización (K).
 - Base de la curva de aprendizaje (A).
 - Exponente de la curva de aprendizaje (B).

- Parámetro de vecindad en la red SOM (σ).
- Número de pases totales sobre el dataset durante el entrenamiento.
- Número de iteraciones completadas en el momento en el que se guardó el punto de entrenamiento.
- Atributo especial para analizar la pérdida de información.
- Fecha en la que se guardó el punto de entrenamiento.
- Identificador del dataset al que pertenecen.

4.5. Análisis GAP

Existen múltiples aplicaciones de código abierto y privativo en el mercado actual que tratan de dar solución al problema de la privatización de datos. La tabla 4.1 ilustra una comparación entre la aplicación presentada y otras existentes en el mercado actual.

Comparativa de aplicaciones

Características	Amnesia ¹	Nymiz ²	Aplicación Propuesta
<i>Licencia</i>	Código abierto	Código privado	Código abierto
<i>Privatización de Datos</i>	Si	Si	Si
<i>Estrategias de anonimización</i>	Supresión, Enmascaramiento, Sustitución	Enmascaramiento, Supresión, Generalización	Enmascaramiento, Sustitución, Generalización
<i>Soporte de diferentes tipos de datos</i>	Si	Si	Si
<i>Evaluación de Riesgos de Reidentificación</i>	Si	Si	Si
<i>Gestión de Metadatos</i>	Gratis	Tarifa por uso	Gratis

Tabla 4.1

Análisis GAP

5. Análisis del Sistema

5.1. Modelo Conceptual

En base a los requisitos de información detallados en la sección 4.4, se dibuja el siguiente diagrama conceptual para ilustrar las relaciones entre las entidades que forman el sistema.

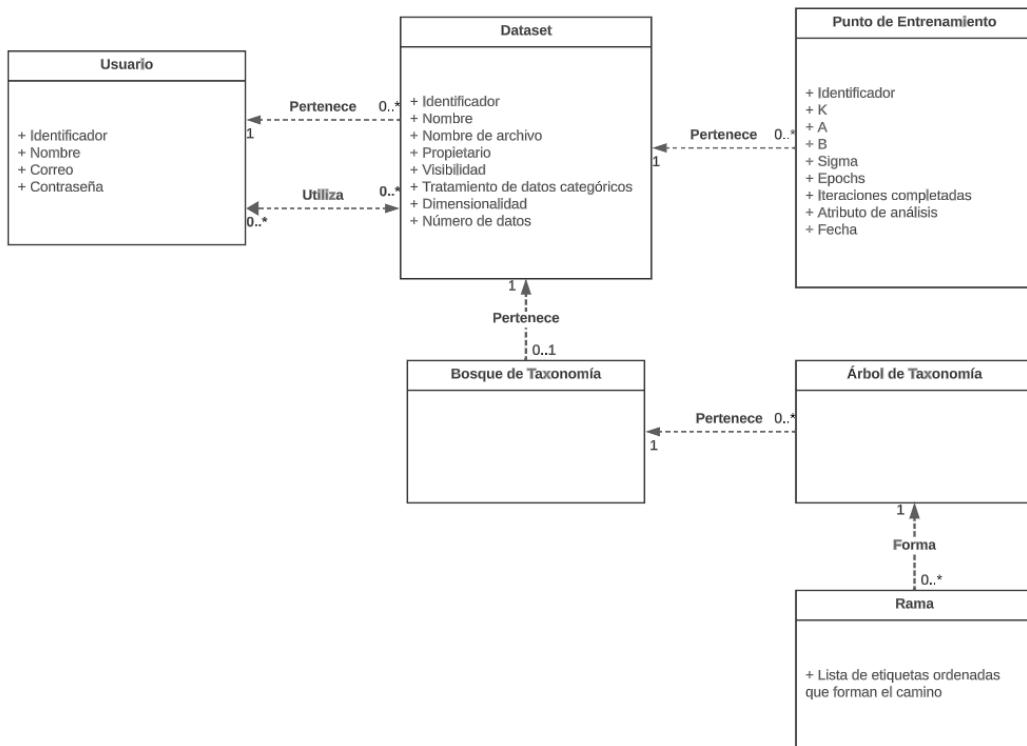


Figura 5.1: Modelo conceptual de datos.

5.2. Modelo de Casos de Uso

En esta sección se detallan los dos casos de uso principales que conforman el sistema, subir archivos y operar sobre ellos. El conjunto completo de flujos posibles y casos de uso del sistema se incluyen en el anexo C.

Nombre del CU:	Subir Dataset		
Creador por:	Rafael Rosso Giner	Última odificación por:	Rafael Rosso Giner
Fecha de creación:	22/07/2023	Última Revisión:	22/07/2023
Descripción:	El usuario sube un nuevo conjunto de datos al sistema que podrá ser usado en el futuro por el propietario y, opcionalmente, por otros usuarios de la plataforma.		
Actores:	Usuario, Sistema		
Precondiciones:	1. El usuario está conectado al servicio. 2. El usuario posee un dataset con formato CSV. 3. El Sistema es accesible, preparado para recibir los datos y con espacio suficiente para almacenarlos		
Postcondiciones:	1. El sistema guarda el documento con el dataset y genera una nueva entrada en la base de datos con los metadatos correspondientes.		
Flujo:	1. El usuario accede al sistema. 2. El usuario navega hasta su perfil. 3. El usuario selecciona la opción “subir dataset”. 4. El usuario selecciona el botón “seleccionar archivo”. 5. El usuario selecciona el documento a subir utilizando la interfaz de su SO. 6. El usuario selecciona la opción “subir archivo”. 7. El sistema indica al usuario que el documento se ha subido correctamente.		
Flujos alternativos:	5a. En el paso 5, si el usuario no ha seleccionado ningún archivo. 6. El usuario selecciona la opción “subir archivo”. 7. El Sistema le informa de que debe seleccionar un archivo. 8. El caso de uso vuelve al paso 4 del flujo. 5b. En el paso 5, si el usuario selecciona un archivo inválido. 6. El usuario selecciona la opción “subir archivo”. 7. El Sistema informa al usuario de que el dataset es inválido y lista los errores que deben solventarse para que pueda subirse. 8. El caso de uso vuelve al paso 4 del flujo.		
Excepciones:	6a. En el paso 6, la conexión es detenida de forma abrupta o el usuario cierra la aplicación. 7. El sistema no almacena el dataset ni ningún dato relacionado. 8. El caso de uso finaliza.		
Requisitos:	Las siguientes condiciones deben cumplirse antes de comenzar con el caso de uso. 1. El usuario debe estar registrado y conectado al servicio con su cuenta. 2. El Sistema debe poseer suficiente espacio para almacenar los datos.		

Figura 5.2: Caso de Uso “Subir un Dataset”.

Nombre del CU:	Análisis SOM		
Creador por:	Rafael Rosso Giner	Última odificación por:	Rafael Rosso Giner
Fecha de creación:	22/07/2023	Última Revisión:	22/07/2023
Descripción:	El usuario realiza un análisis SOM sobre un conjunto de datos para buscar agrupaciones.		
Actores:	Usuario, Sistema		
Precondiciones:	1. El usuario está conectado al servicio. 2. El usuario tiene acceso a uno o más datasets del sistema. 3. El Sistema es accesible, preparado para recibir los datos y con espacio suficiente para almacenarlo		
Postcondiciones:	1. El usuario accede a una pantalla con los análisis del dataset.		
Flujo:	1. El usuario accede al sistema. 2. El usuario navega hasta la pantalla principal. 3. El usuario selecciona la opción “Análisis SOM”. 4. El usuario selecciona la operación “Clustering”. 5. El usuario indica los parámetros que desea utilizar para el entrenamiento. 6. El usuario presiona el botón “Train SOM”. 7. El sistema muestra una pantalla de carga al usuario. 8. El usuario espera hasta que el sistema termine de procesar la operación. 9. El sistema muestra los resultados.		
Flujos alternativos:	8a. El usuario decide pulsar la opción de abortar antes de que termine de ejecutarse la operación. 8. El Sistema cancela la operación actual. 9. El caso de uso vuelve al paso 5. 7b. Hay algún error con la combinación de parámetros o dataset seleccionado para dicha combinación. 8. El Sistema informa al usuario del error y como debe solventarlo. 9. El caso de uso vuelve al paso 5.		
Excepciones:	8a. La operación se detiene de forma abrupta. 8. El sistema informa al usuario de que ha ocurrido un error y la operación no ha podido completarse. 9. El caso de uso finaliza.		
Requisitos:	Las siguientes condiciones deben cumplirse antes de comenzar con el caso de uso. 1. El usuario debe estar registrado y conectado al servicio con su cuenta. 2. El usuario debe tener acceso a uno o más datasets en el sistema.		

Figura 5.3: Caso de Uso “Análisis SOM”.

5.3. Modelo de Interfaz de Usuario

La interacción con el usuario se realiza a través de las múltiples páginas web que forman la aplicación. Aquí se muestra un diagrama del flujo de navegación de la misma y las acciones necesarias para moverse entre pantallas.

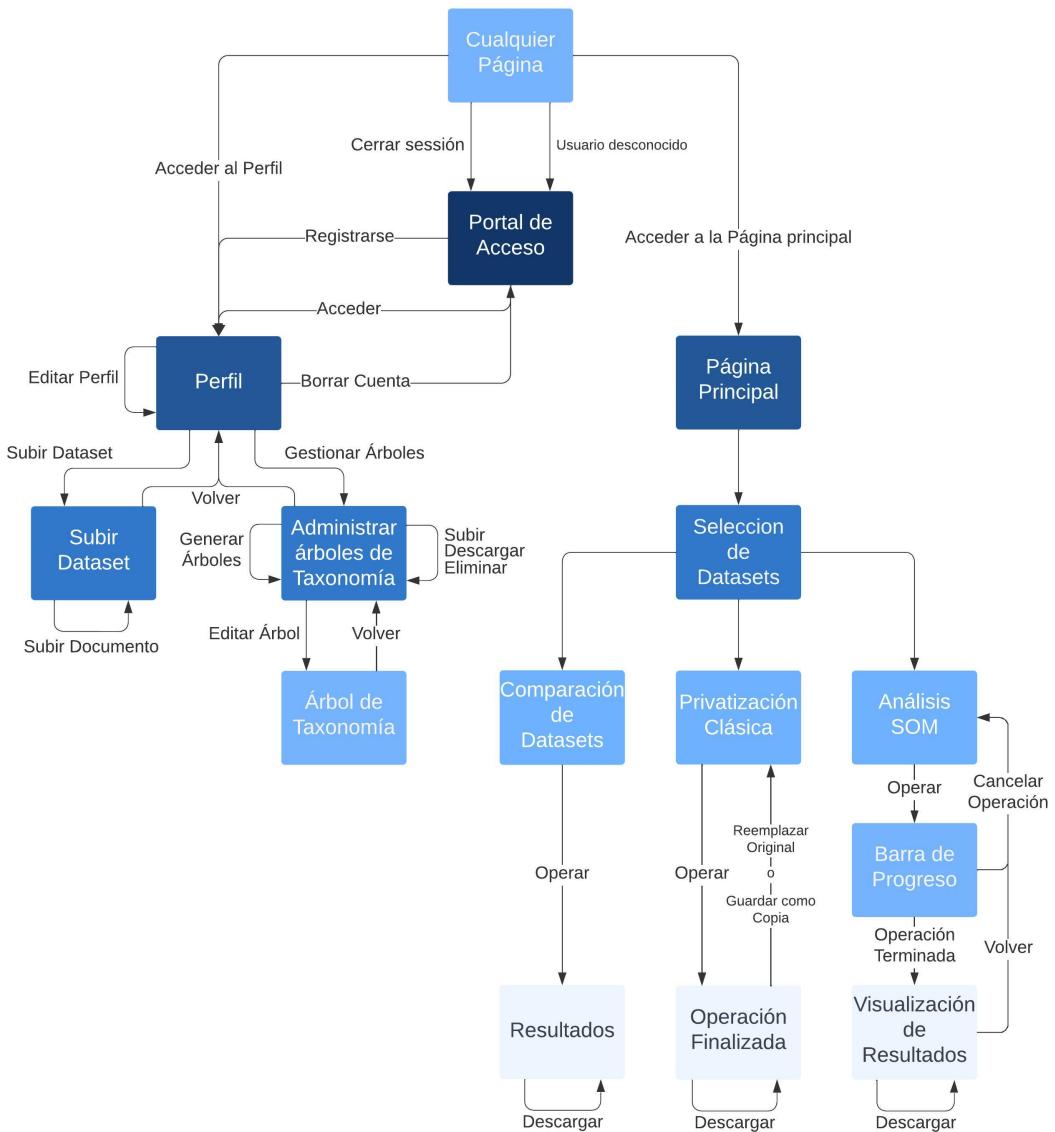


Figura 5.4: Flujo de pantallas del sistema.

5.4. Modelo de Comportamiento

La arquitectura del sistema sigue un modelo de capas donde cada capa utiliza de manera interna los servicios de la capa inferior, tal y como se puede observar en los siguientes diagramas de secuencia de operaciones (figura 5.6 y figura 5.5).

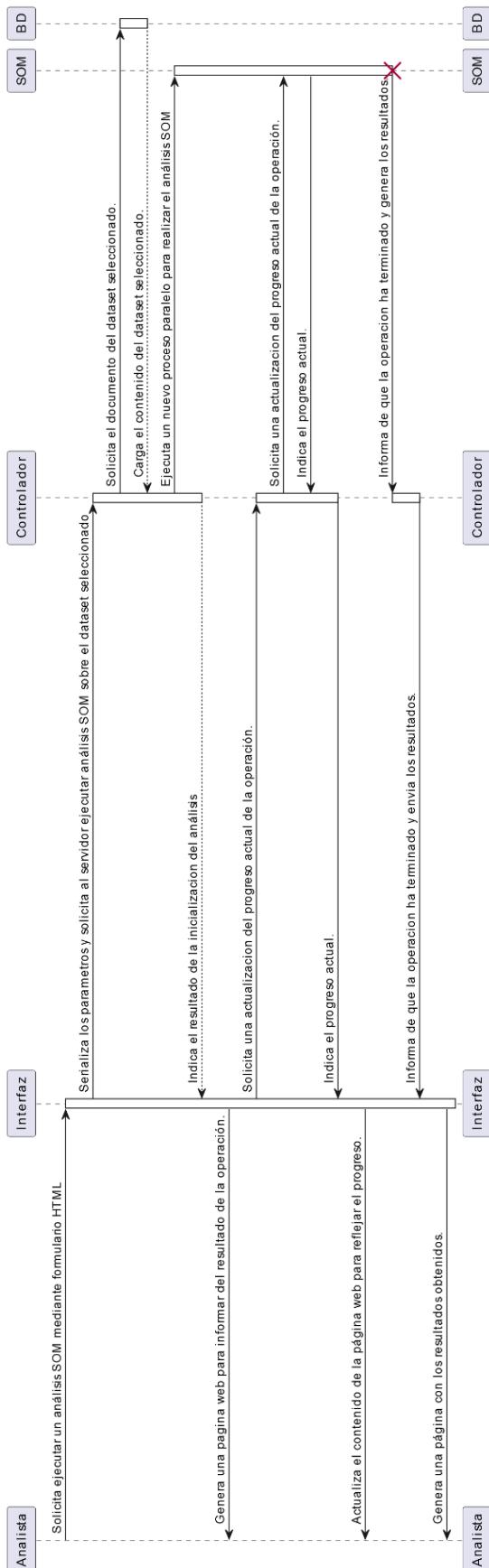


Figura 5.5: Proceso Análisis SOM

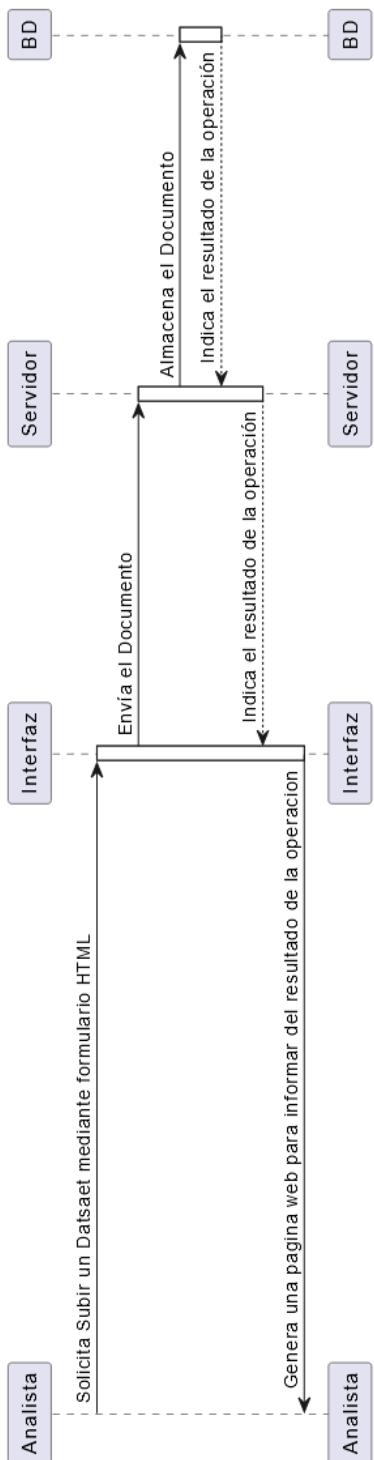


Figura 5.6: Proceso subida de dataset

En el anexo B.1 se detallan el resto de operaciones del sistema siguiendo una estructura similar.

6. Diseño del Sistema

6.1. Arquitectura del Sistema

El sistema utiliza Python como interfaz principal entre las diferentes tecnologías que permiten al usuario realizar un tratamiento de datos. Toda la interacción con el usuario se realiza a través de un servicio web diseñado mediante el paquete Flask¹.

El procesamiento de los datos se realiza a través de la interfaz que proporciona Numpy, a través de funciones compatibles con el interprete de Numba. De esta forma se acelera el procesamiento de operaciones computacionalmente costosas, maximizando el uso de los recursos hardware disponibles en la máquina.

6.1.1. Diseño de alto nivel

La arquitectura del sistema sigue un modelo de capas donde cada capa utiliza de manera interna los servicios de la capa inferior. La estructura principal (figura 6.1) de la aplicación consta de tres sistemas principales, que se ocupan de gestionar las diferentes áreas del proyecto.

¹Flask: <https://flask.palletsprojects.com/en/2.3.x/>

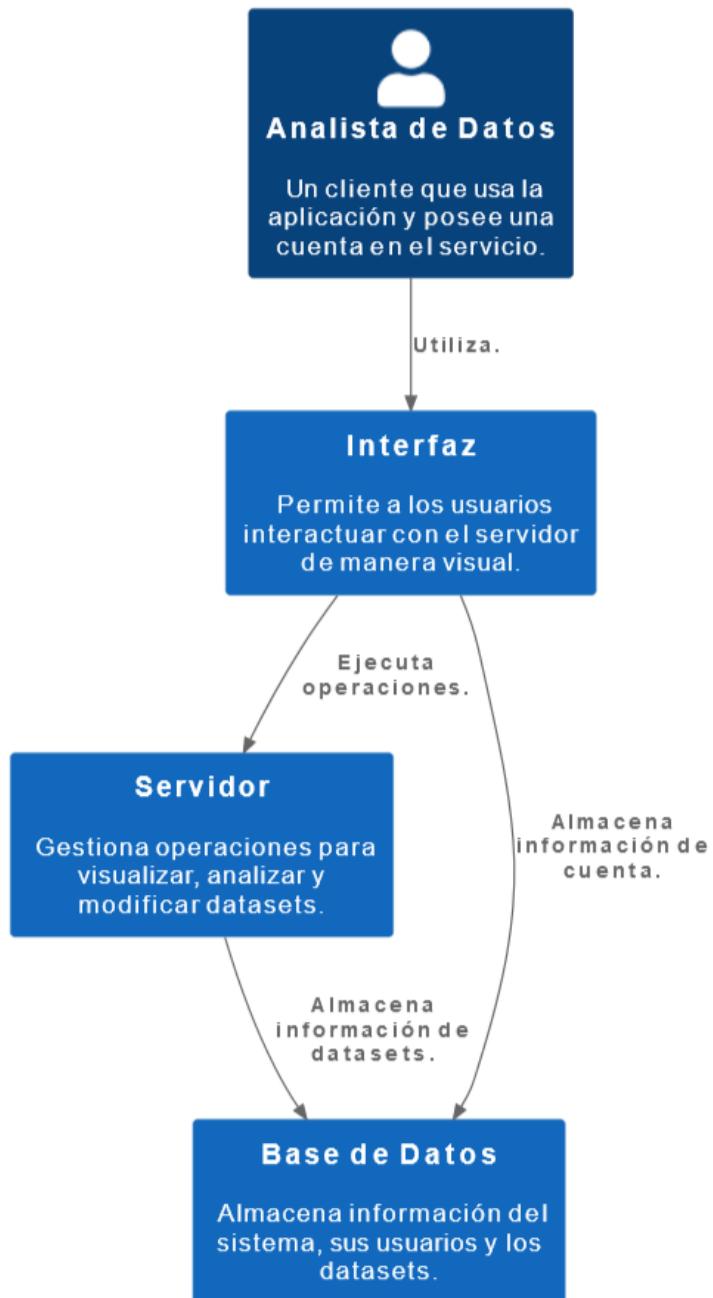


Figura 6.1: Vista principal

La primera capa es la de interfaz (figura B.1), encargada de ofrecer un entorno visual donde el usuario pueda interactuar libremente con el servicio web. El servidor web que se encarga de ello esta basado en WSGI y es generado automáticamente por Flask a partir del código definido en las secciones de Aplicación (figura B.2) y Autenticación (figura B.3). Cada petición es procesada por una de estas dos secciones según su ruta, la de Autenticación se comunica directamente con la capa de Base de Datos (figura B.8) mientras que la de Aplicación interacciona con el Servidor (figura 6.2).

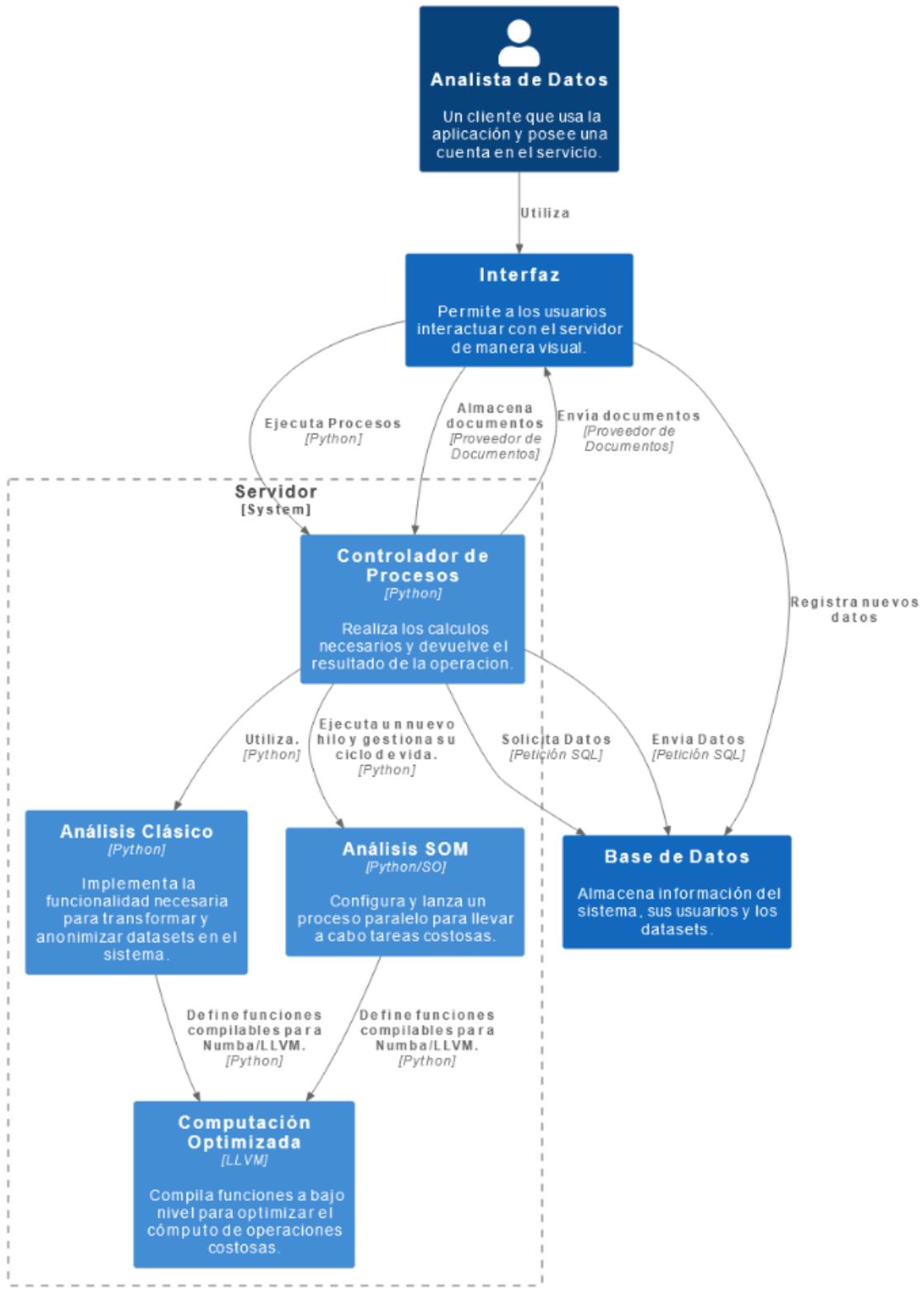


Figura 6.2: Servidor

La Base de datos tiene un sistema de almacenamiento principal centrado en un motor de base de datos relacional, MariaDB si se esta utilizando la configuración por defecto. Adicionalmente se utiliza el sistema de archivos proporcionado por el SO para respaldar los datos de cada uno de los datasets almacenados en el sistema.

La capa de Servidor involucra múltiples partes, donde para controlar el flujo de las operaciones y sus resultados se define el controlador de procesos (figura B.4). El servidor implementa el módulo de Análisis Clásico (figura B.5) para poder aplicar algoritmos sencillos de privatización y comparación de datos a los datasets que existen en el sistema. Mientras que, para realizar análisis y transformaciones más complejas, se utiliza el módulo de Análisis SOM (figura (B.6)). Adicionalmente, el servidor consta de un módulo de computo optimizado que permite acelerar el procesamiento de operaciones especialmente costosas (figura B.7).

El módulo de análisis SOM consta a su vez de dos componentes diferenciados, la red SOM y el Bosque de Taxonomía, que se combinan para ofrecer mayores opciones a la hora de trabajar con campos categóricos.

6.1.2. Diseño detallado

A continuación se explica en profundidad el diseño algorítmico de las técnicas aplicadas y el servicio web desarrollado. Se dejará para la sección 6.4 el diseño de la interfaz visual y para la sección 7.2 los apartados de optimización y gestión de la información.

Comparación de Datasets

Una de las funciones del sistema es comparar dos datasets utilizando algoritmos deterministas, para ello se evalúan ambos conjuntos de datos y se realiza un análisis de componentes principales (PCA) sobre cada uno de los datasets. Luego, se computa el coeficiente de correlación de Pearson entre los componentes principales y la nube de datos original. Una vez obtenidos estos indicadores para cada uno de los dos datasets es posible calcular el error cuadrático medio (MSE), error absoluto medio (MAE) y variación media(MV).

Para completar la comparación, se añade la media y varianza de cada uno de los atributos numéricos a los datasets originales. Aunque en el caso de los campos categóricos, la media se sustituye por el valor más frecuente, acompañado del porcentaje de los datos de dicho campo que representa. Finalmente todos los resultados se muestran al usuario para que este pueda extraer sus conclusiones de los valores generados por el sistema como se puede ver en las figuras 6.3 y 6.4.

Dataset comparison							
Dataset: adultNamed.data							
Index	age	workclass	education	education-num	marital-status	occupation	
0	39	State-gov	Bachelors	13	Never-married	Adm-clerical	
1	50	Self-emp-not-inc	Bachelors	13	Married-civ-spouse	Exec-managerial	
2	38	Private	HS-grad	9	Divorced	Handlers-cleaners	

Dataset: NoisyAdult.data							
Index	age	workclass	education	education-num	marital-status	occupation	
0	39.374	State-gov	Bachelors	12.813	Never-married	Adm-clerical	
1	50.513	Self-emp-not-inc	Bachelors	13.03	Married-civ-spouse	Exec-managerial	
2	37.371	Private	HS-grad	9.041	Divorced	Handlers-cleaners	

Figura 6.3: Comparación de Datasets

PCA Summary							
Component	Significance	Most important feature (Significance)					
PC0	99.703%	capital-gain (100.0%)					
PC1	0.297%	capital-loss (99.999%)					
PC2	0.0%	age (96.333%)					
PC3	0.0%	hours-per-week (96.295%)					
PC4	0.0%	education-num (99.957%)					

Summary of Principal Components Analysis (PCA) over the dataset							
Component	age	education-num	capital-gain	capital-loss	hours-per-week		
PC0	0.014%	0.004%	100.0%	0.173%	0.013%		
PC1	0.204%	0.054%	0.173%	99.999%	0.174%		
PC2	96.333%	1.042%	0.018%	0.244%	26.809%		
PC3	26.828%	2.738%	0.009%	0.114%	96.295%		
PC4	0.269%	99.957%	0.004%	0.048%	2.917%		

PCA Summary							
Component	Significance	Most important feature (Significance)					
PC0	99.703%	capital-gain (100.0%)					
PC1	0.297%	capital-loss (99.999%)					
PC2	0.0%	age (96.364%)					
PC3	0.0%	hours-per-week (96.326%)					
PC4	0.0%	education-num (99.957%)					

Figura 6.4: Análisis de componentes principales

Anonimización

Para transformar el dataset, la aplicación implementa cuatro operaciones clásicas de transformación de datos numéricos y categóricos. Siendo una de estas es la extracción de subconjuntos de datos, para eliminar valores indeseados.

El sistema ofrece la posibilidad de realizar una permutación sobre los datos, la cual permite intercambiar el valor de uno o más campos de forma aleatoria a lo largo de todos

los registros del dataset. Se trata de un proceso muy efectivo en la anonimización de los registros, pero también es una operación destructiva que rompe las correlaciones entre los diferentes atributos del dataset. Para mitigar este daño, se incluye la posibilidad de intercambiar los valores de los campos de forma conjunta, lo que nos permite preservar las correlaciones entre los atributos que están intercambiándose. Resulta especialmente útil cuando se conoce de forma previa alguna relación entre múltiples campos que se quiera mantener.

Otra forma de anonimizar los datos es la adición de ruido, aunque esta operación está limitada únicamente a los campos numéricos. El sistema permite añadir ruido uniforme o, en su lugar, utilizar el método de [Kim \(1986\)](#) para generar ruido de forma más precisa. En ambos casos, la generación de ruido es parametrizable y está escalada a la magnitud del atributo en cuestión.

Finalmente la aplicación es capaz de realizar un proceso de generalización del dataset indicando cuántas subsecciones tendrá cada uno de los campos. Este tipo de operaciones sirve para enmascarar los datos al mismo tiempo que se preserva la información de cada uno de los registros y relaciones entre ellos. Puede ser especialmente útil para la publicación de datos en portales públicos siempre y cuando se haya comprobado el riesgo de divulgación existente.

Mapas Auto Organizados (SOM)

Un mapa auto organizado es un algoritmo de red neuronal no supervisada que tiene como objetivo amoldarse a la forma que tiene una nube de datos multidimensional ([Hulle, 2012](#)). En la aplicación desarrollada, estas redes toman un papel importante en el análisis de los datos, puesto que al tratarse de un entrenamiento no supervisado pueden trabajar sin necesidad de aportarles ningún dato adicional o tratamiento previo del dataset.

El usuario puede aprovecharse de la red para generar grupos diferenciados de datos que sirvan como indicio del riesgo de identificación presente en los datos a analizar. Si se generan pocos grupos bien balanceados, el análisis sugiere que es difícil separar y por tanto identificar uno de los registros mediante sus atributos. Sin embargo, en el caso de encontrar grupos marginales con un número reducido de datos, podemos asumir que existe la posibilidad de identificar a través de los atributos a dichos registros. Normalmente, se denomina a estos grupos de datos mediante su término inglés *cluster*.

Una red SOM se puede definir como un conjunto de neuronas conectadas por una topología específica. La topología define la cercanía que hay entre cada par de neuronas y afecta significativamente al resultado final de la red. Este proyecto utiliza una malla cuadrada de neuronas y se emplea una función gaussiana para indicar la distancia entre cada pareja de neuronas de la red con el objetivo de suavizar la diferencia de pesos a medida que se separan las neuronas. Otra alternativa más simple sería definir una matriz de conexiones binaria, pero esto podría causar diferencias demasiado bruscas puesto que solo existen dos estados de influencia.

El proceso de entrenamiento de una red SOM consta de dos etapas diferenciadas, la competitiva y la cooperativa, que se llevan a cabo para cada iteración del dataset.

Durante la etapa competitiva se selecciona una fila aleatoria del dataset y se calcula la distancia entre el elemento y cada una de las neuronas. La más cercana se denomina neurona ganadora y sirve para actualizar la red durante la fase cooperativa, cada neurona de la red actualizará el valor de sus campos siguiendo la siguiente ecuación 6.5.

$$W(N_i, N_{bmu}) = W(N_i) + \alpha * [\beta(N_i, N_{bmu}) * \frac{1 - I}{I_{max}}] * [V - W(N_i)] \quad (6.1)$$

Figura 6.5: Actualización SOM

Actualización de los pesos W de la neurona N_i donde N_{bmu} representa a la neurona ganadora, I el número de iteraciones, I_{max} las iteraciones totales, $\beta(N_A, N_B)$ la cercanía entre las neuronas N_A y N_B . V representa el vector de atributos seleccionado en la iteración actual (I). El coeficiente de aprendizaje (α) queda definido por la ecuación 6.6.

$$\alpha = A^{-B \frac{I}{N}} \quad (6.2)$$

Figura 6.6: Coeficiente de Aprendizaje

Coeficiente de Aprendizaje donde A y B son parámetros proporcionados por el usuario; I representa el número de iteraciones completadas sobre el dataset y N el número de registro en el dataset.

Evaluación de resultados

El entrenamiento de las redes SOM da lugar a una matriz de neuronas adaptadas a la distribución de la nube de datos. Estas neuronas pueden ser representadas de múltiples formas según la necesidad e interpretadas de acorde a ello.

Cada vez que un usuario ejecuta este tipo de análisis, se generan múltiples imágenes para representar la influencia de cada atributo en el mapa neuronal, tal y como se muestra en las figuras 6.7a y 6.7b. Además, se genera un mapa de aciertos que representa la activación total que ha sufrido una neurona a lo largo del entrenamiento, lo que sirve para tener una perspectiva más objetiva de la magnitud del resto de mapas generados (ver figura 6.7d).

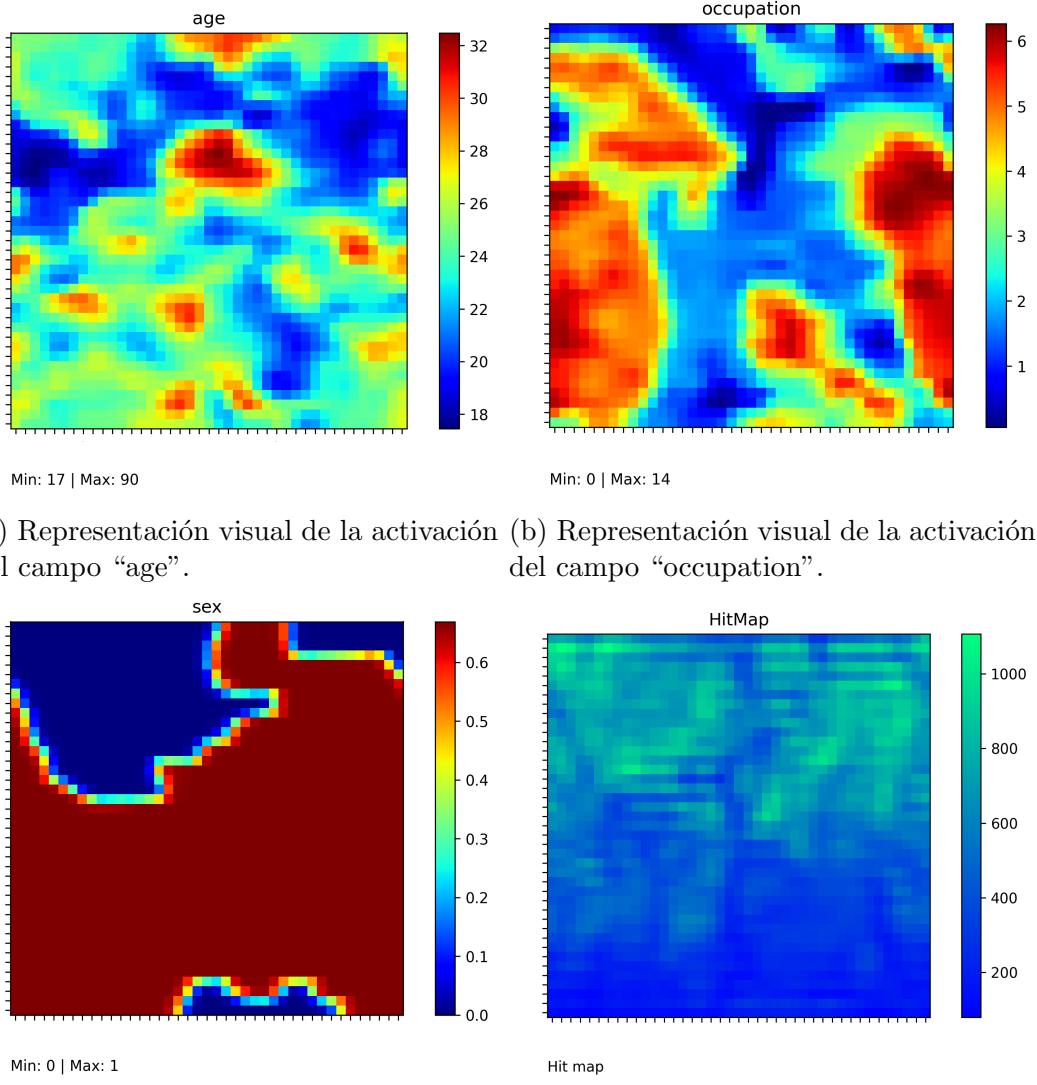


Figura 6.7: Evaluación de resultados.

Imágenes generadas automáticamente por la aplicación para representar la activación neuronal en la red para cada campo.

Adicionalmente, todos los clusters generados se acompañan de un valor que representa la información perdida al generalizar los datos. Este valor es una combinación de la distancia entre el centroide del cluster y todos los elementos del cluster. Los campos categóricos son evaluados utilizando el algoritmo descrito en el siguiente apartado, para obtener el camino más corto posible de entre todas las combinaciones existentes entre los valores del cluster para dicho atributo. De forma simultánea, cada campo numérico se procesa dividiendo el rango de los valores del cluster entre el rango numérico de ese mismo atributo en el dataset completo. Finalmente se suman los valores de cada atributo para generar el valor de información perdida dentro de cada cluster.

Árboles de Taxonomía

Para calcular la distancia entre valores categóricos, el operador de sustracción deja de ser aplicable puesto que la mayoría de dominios categóricos no pueden ser enumerados en ningún orden específico. Si bien es posible asumir que la distancia es máxima cuando los valores son diferentes y mínima cuando se trata del mismo valor, en la práctica esto resulta poco útil. Además, es común que haya atributos donde los posibles valores del dominio estén relacionados semánticamente, lo que hace particularmente efectivo el uso de árboles de taxonomía para reflejar dichas relaciones (Ji-Won Byun y Li, 2003).

Un árbol de taxonomía es un árbol donde cada nodo hoja representa un posible valor que existe en el dominio estudiado y los nodos intermedios sirven para definir relaciones jerárquicas que los agrupen. En el contexto de la aplicación, estos árboles son definidos por el usuario mediante un documento de texto. El formato de este documento se trata sencillamente de una lista de valores separados por punto y coma donde cada fila indica un único camino que va desde el nodo hoja hasta la raíz, pasando por todos los nodos intermedios. En el anexo se puede observar un ejemplo que ilustra como se formaría un árbol D.1. La distancia entre dos nodos cualquiera de un árbol de taxonomía puede calcularse a partir de la función de distancia 6.8.

$$D(a, b) = \frac{H(T_\alpha(a, b))}{H(T_d)} \quad (6.3)$$

Figura 6.8: Distancia entre nodos

Distancia $D(a, b)$ entre dos nodos a y b donde $T_\alpha(a, b)$ representa el sub-árbol más pequeño cuya raíz es ancestro de ambos nodos y T_d representa el árbol de taxonomía. $H(T_x)$ es la función que define altura del árbol T_x

Cada árbol de taxonomía representa a un único campo categórico de un dataset. Al conjunto de todos los árboles de un dataset se le denominará bosque de taxonomía. Para optimizar el cómputo, todos los árboles y sus correspondientes etiquetas son codificadas durante la inicialización del bosque de taxonomía con un identificador numérico. Debido a la naturaleza de los árboles, es posible pre-calcular todas las distancias durante la inicialización para facilitar luego el cómputo de las distancias entre registros. Para ello iteramos por todas y cada una de las posibles combinaciones de etiquetas calculando la distancia entre ellas.

Fase de Ajuste

Debido a que la red SOM no garantiza que el resultado del algoritmo siempre genere clusters de mínimo K elementos, es necesario realizar una fase de ajuste final para poder lograrlo. Esta fase es opcional y solo se realizará si el usuario lo indica expresamente en los parámetros de la operación.

Para la fase de ajuste se utiliza el siguiente algoritmo ([Lin y Wei, 2008](#)), donde nuestro objetivo es poder generar clusters de mínimo K elementos.

Luego necesitaremos calcular el centroide de cada uno de los clusters restantes. Para calcular estos centroides se procesan por separado los datos numéricos y categóricos, utilizando una media estadística para los campos numéricos y la función de distancia [6.8](#) definida en el apartado anterior para los campos categóricos. El valor que tomará el centroide en este segundo tipo de atributos será la raíz del sub-árbol con menor altura que contenga a todos los posibles valores del atributo.

Una vez obtenido el centroide de cada cluster, se itera a través de todos ellos y, para cada cluster con más de K elementos, se calcula la distancia entre el centroide correspondiente y todos los elementos del mismo. Posteriormente se ordenan todos los elementos en orden descendente, de forma que quede una lista de elementos donde el primer elemento es el más alejado de su centroide. Finalmente moveremos a la lista de distribución tantos elementos como sean necesarios para que el cluster quede en K elementos.

A continuación se itera por cada uno de los elementos de la lista de distribución y se calcula la distancia entre dicho elemento y todos los clusters con menos de K elementos. Los elementos se irán asignando al centroide más cercano hasta vaciar la lista de distribución o hasta que todos los centroides alcancen un número de K elementos.

Una vez alcanzado este punto, hay que localizar todos los clusters con menos de K elementos, para eliminarlos y guardar todos sus registros en una lista de distribución. Si solo estamos tratando con campos numéricos, este proceso se realiza al comienzo para optimizar el cómputo pues los centroides coincidirán con aquellos utilizados para la fase de clasificación.

Finalmente, si la lista de distribución aún contiene algún elemento, este será colocado en el cluster cuyo centroide se encuentre más cerca. En esta etapa no es relevante el número de elementos que tenga el centroide destino puesto que ya hemos satisfecho la condición de la K-anonimidad requerida.

6.2. Parametrización del software base

El sistema contiene múltiples archivos de configuración, que parametrizan las diferentes partes el entorno de ejecución de la aplicación web.

Config.py: Se aloja en el directorio raíz de la aplicación web y es responsable de la mayoría de parámetros que dictan el comportamiento del sistema.

- **FLASK_WORKERS:** Número máximo de hilos que utilizará la aplicación de Flask para ejecutarse.
- **WEB_MAX_PREVIEW_ENTRIES:** Número de filas a mostrar cuando se previsualiza un dataset.
- **WEB_MAX_TRAININGPOINTS_PER_DATASET:** Número máximo de puntos de entrenamiento que almacena el sistema por cada dataset.

- PROGRESS_BAR_UPDATE_SPEED: Tiempo en ms que tarda el cliente en solicitar una actualización del progreso.
- DATASET_FILENAME_PREFIX: Prefijo que se le añade a los datasets subidos al sistema.
- TRAININGPOINT_FILENAME_PREFIX: Prefijo que se le añade a los puntos de entrenamiento creados en el sistema.
- TAXONOMY_TREE_FILENAME_PREFIX: Prefijo que se le añade a los árboles de taxonomía subidos al sistema.
- TAXONOMY_TREE_DIRECTORY_PREFIX: Prefijo que se le añade al directorio de árboles de taxonomía de cada dataset.
- DATASET_CLUSTERED_SUFFIX: Sufijo que se le añade al documento descargado con los resultados de un análisis SOM.
- DATASETS_SAVE_PATH: Directorio donde se guardan los datasets.
- DATASETS_TEMPORARY_PATH: Directorio donde se guardan los documentos temporales.
- DATASETS_TRAINED_POINTS_PATH: Directorio donde se guardan los puntos de entrenamiento.
- DATASETS_RESULTS_PATH: Directorio donde se guardan los resultados.
- TAXONOMY_TREE_PATH: Directorio donde se guardan los árboles de taxonomía.
- UPLOAD_FOLDER_PATH: Directorio donde se almacenan temporalmente los datasets.
- OUTPUT_IMAGE_FORMAT: Formato de salida de las imágenes generadas.
- ALLOWED_DATASET_EXTENSIONS: Extensiones permitidas a la hora de subir un dataset.
- TAXONOMY_TREE_EXTENSION: Extensiones permitidas a la hora de subir un árbol de taxonomía.
- DataBaseURI: Cadena de caractéres que define la conexión a la base de datos del sistema.
- SecretKey: Variable de protección de Flask. Debe ser privada.
- LOGGING_ENABLED: Habilita o deshabilita la opción para generar un archivo que registre los eventos de la aplicación.
- LOGGING_LEVEL: Define el nivel de importancia que debe tener un evento para ser registrado.
- LOGGING_TRACEBACK: Define si los eventos deben contener o no la lista de llamadas que causaron el evento.
- LOGGING_FILENAME: Nombre que tendrá el archivo de eventos generado.

flask_settings.py: Se aloja en el directorio instancia de la aplicación web y sirve como base de configuración de Flask. Gracias a este documento es posible definir cualquier configuración que no se haya indicado ya en **Config.py**.

gunicorn.py: Se aloja en el directorio raíz del contenedor Docker y es exclusivo al mismo. Sirve para proporcionar argumentos adicionales al servidor WSGI proporcionado por gunicorn ².

6.3. Diseño Físico de Datos

El sistema almacena todos los datos necesarios para su funcionamiento en dos bloques interconectados, por una parte se utiliza un sistema de base de datos relacional para almacenar toda la información de los usuarios y metadatos acerca de los archivos que actúan en las operaciones de la aplicación. Por otra parte, se utiliza el sistema de archivos proporcionado por el sistema operativo de la máquina que ejecuta el servicio web para almacenar los documentos en bruto sobre los que se opera desde la aplicación.

6.3.1. Base de datos

El sistema de base de datos está construido a partir del uso de middleware de SQLAlchemy lo que permite tener cierta flexibilidad y configurar diferentes motores de base de datos según las necesidades del entorno donde vaya a desplegarse la aplicación, siempre que estos cumplan requisitos esenciales como la capacidad de gestionar múltiples conexiones simultáneas. Durante el desarrollo de este proyecto se ha empleado MariaDB como motor de base de datos debido a la flexibilidad y facilidad de uso que provee, siendo además un software de código abierto en sintonía con el objetivo de nuestra aplicación.

La estructura de la base de datos es sencilla y está compuesta por tan solo tres tablas que gestionan al completo la información necesaria.

- User. Cada fila representa a un usuario, se almacenan los datos necesarios para permitir el acceso al servicio y gestionar de control sobre el uso de cada dataset.
- Dataset. Almacena todos los datos referentes a un dataset. Esto incluye el identificador del archivo en disco que lo representa, la visibilidad del mismo y su identificador en el sistema entre otros campos. Se guardan también campos como la dimensionalidad y número de registros de cada archivo con el objetivo de agilizar las búsquedas ordenadas en el sistema.
- Trained_som. Representan un estado ya entrenado de una red SOM sobre un dataset concreto. Cada fila se relaciona con un único dataset y almacenan toda la información necesaria sobre el entrenamiento en el momento en el que se realiza el punto de guardado, de forma que pueda retomarse el entrenamiento en el futuro sin necesidad de aportar datos adicionales.

²Gunicorn: <https://gunicorn.org/>

6.3.2. Sistema de archivos

La estructura se basa en cinco directorios configurables a través del archivo Config 6.2, cada uno de ellos guarda un tipo de archivo en concreto y tiene unas reglas específicas sobre la permanencia de los datos que se guarden en el mismo.

- Datasets. En este directorio se almacenan todos los datasets que gestiona el sistema, los archivos se identifican mediante un nombre único generado en función del identificador único de la base de datos. Cuando se inicia la aplicación, se sincroniza con la base de datos de forma que todo archivo sin representación queda eliminado del directorio y toda entrada de la tabla de la base de datos sin archivo en disco queda eliminada de la misma.
- Temporary. Este directorio intermedio sirve para almacenar archivos durante las operaciones del sistema, es eliminado cada vez que se inicia la aplicación. Debido a su volatilidad, las funciones que operan sobre archivos en este directorio asumen que los documentos pueden ser eliminados en cualquier momento.
- Results. Este directorio es similar al temporal, sirven para almacenar los resultados de la última operación realizada por el usuario. También se elimina su contenido al inicio de la aplicación, por lo que es volátil y las funciones que operan sobre él tienen esto en cuenta para el manejo de posibles errores. Cada vez que el usuario realiza una operación, reemplaza los datos de su última operación con la nueva.
- Taxonomy Trees. Almacena un directorio por cada uno de los datasets del sistema. En cada uno de estos directorios se guardan todos los documentos que representan cada árbol de taxonomía para un dataset específico. Estos archivos están vinculados a su dataset de forma que estos son eliminados y creados junto a la eliminación y creación del dataset al que representan.
- Trained Points. Almacena documentos para representar el estado de la matriz de pesos de la red SOM en el momento en el que el sistema decide realizar un punto de guardado durante el proceso de entrenamiento. Permiten cargar en memoria los valores junto a los metadatos de la base de datos para retomar el proceso en el futuro.

6.4. Diseño de la Interfaz de Usuario

6.4.1. Flask

A la hora de desarrollar la interfaz visual, se plantearon varias opciones como Django y Bottle, aunque ambas fueron descartadas en etapas tempranas de la construcción de la aplicación. Django se trata de un framework muy completo y con gran cantidad de sistemas e implementaciones que se escapan del alcance del proyecto, habría requerido una gran inversión de tiempo, esfuerzo y aprendizaje para obtener un resultado similar a cualquier otro framework más accesible.

Es por ello que se decidió prototipar el servicio web con Bottle para la fase inicial de experimentación. Bottle se trata de un framework que nace de un subconjunto de la funcionalidad de Flask, es un software minimalista orientado a pequeños proyectos que facilita el prototipado de aplicaciones de forma rápida y eficiente. Sin embargo, a lo largo de dicho prototipado, fueron apareciendo numerosos inconvenientes a la hora de trabajar con bases de datos y mostrar contenido dinámico en la web. Es por ello que se terminó optando por migrar todo el sistema a un nuevo framework más avanzado como es Flask, principalmente porque al partir de una misma base, la transición sería mucho más directa y efectiva que en un entorno completamente nuevo.

Flask se sigue considerando como un sistema minimalista, ya que tiene dependencias de software externo y funciona completamente mediante paquetes de Python. Aunque esto no limita sus capacidades a la hora de desarrollar aplicaciones web completas y de carácter profesional, algunos ejemplos de sitios exitosos que han sido construidos con Flask por ejemplo son Reddit, LinkedIn y Pinterest.

Gracias a la implementación de jinja2 y werkzeug, Flask es capaz de construir un servicio web basado completamente en WSGI para ofrecer un servicio escalable que se adapte a las necesidades de este proyecto.

La interfaz de usuario se realiza mediante rutas web gestionadas por la arquitectura de Flask, cada ruta está asociada a una función de Python que se encarga de realizar todas las operaciones necesarias y enviar una respuesta al cliente. Esta respuesta se trata de una nueva página html donde el usuario puede decidir como desea proseguir con su trabajo, en algunos casos la respuesta puede tratarse de un archivo descargable o un fragmento de código para que el cliente pueda actualizar el contenido de la página de forma dinámica.

6.4.2. Autenticación

Cuando un usuario accede a cualquier ruta del servicio web, primero se comprobará si el usuario está o no autenticado. En caso de no estarlo, será enviado a la página de inicio donde podrá introducir los datos para acceder a su cuenta [6.9a](#) o crear una nueva cuenta si lo desea [6.9b](#).

Una cuenta solo precisa de nombre, email y contraseña. El nombre debe ser único puesto que se utilizará para acceder al servicio y será visible para otros usuarios en los datasets compartidos, es posible crear múltiples cuentas asociadas a un único email. Las contraseñas se encriptan antes de almacenarse para mitigar los daños que pudiesen resultar de una filtración de datos en el sistema.

Además, la página de Log-In [6.9a](#) permite marcar una casilla de forma que, si el usuario lo desea, pueda acceder al servicio automáticamente en el futuro sin necesidad de aportar sus credenciales.

(a) Log-In

(b) Sign-Up

Figura 6.9: Autenticación

6.4.3. Perfil

El perfil (sección 6.10) es accesible desde cualquier punto de la aplicación a través del panel superior, contiene información básica sobre el usuario y le permite modificar sus datos. Además contiene la lista de datasets accesibles para el usuario en cuestión, tanto propios como compartidos a través de otros clientes.

Los usuarios pueden usar esta página para subir nuevos datasets, editar su visibilidad, editar los árboles de taxonomía de cada uno de ellos, eliminarlos o incluso descargarlos a su máquina local. En la sección de datasets públicos que pertenezcan a otros usuarios, solo se permitirá descargar el documento pero no alterarlo de ninguna forma.

Figura 6.10: Perfil de usuario

6.4.4. Página principal

La página principal es siempre accesible desde cualquier punto de la aplicación a través del panel superior, sirve como punto de acceso a las diferentes secciones de la aplicación (figura 6.11a). Una vez se haya seleccionado una de las operaciones, el usuario es redirigido a la pantalla de selección de datasets (figura 6.11b).

Esta pantalla muestra todos los datasets accesibles, propios o compartidos con el usuario. Para facilitar la búsqueda de datasets es posible filtrar y ordenar la lista utilizando los botones en la parte superior de cada columna. El usuario deberá seleccionar tantos datasets como se le indique antes de poder continuar, el número de datasets dependerá de la operación seleccionada en la pantalla principal.

(a) Selección de operaciones

Available Datasets			
Select 1 dataset			
Your datasets			
Dataset	Dimensions	Entries	Selection
DotBKYProjectsDataAnalysis.csv	24	215221	<input type="radio"/>
adultSOMAdjOut.csv	15	32561	<input type="radio"/>
Public datasets			
Dataset	Dimensions	Entries	Selection
NoisyAdult.data	14	32561	<input type="radio"/>
NewAdultNamed.data	14	32561	<input type="radio"/>
GeneralizedAdult.data	14	32561	<input type="radio"/>
adultNamed.data	14	32561	<input type="radio"/>
adultNamed.data	15	32561	<input type="radio"/>

(b) Selección de dataset

Figura 6.11: Página principal

6.4.5. Anonimización de datos

Las operaciones de anonimización permiten transformar el dataset seleccionado utilizando múltiples técnicas conocidas de privatización. En la sección 6.1.2 se detalla

detenidamente cada uno de estos procesos, los parámetros se explican en la propia aplicación tal y como se puede observar en las figuras 6.12 y 6.13. El uso de cada parámetro también se detalla en la guía de usuario A.3.

Dataset adultNamed.data

age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	income_class
38.757	State-gov	69296.251	Bachelors	13.107	Never-married	Adm-clerical	Not-in-family	White	Male	2363.015	18.286	43.25	United-States	<=50K
53.105	Self-emp-not-inc	65452.982	Bachelors	13.969	Married-civ-spouse	Exec-managerial	Husband	White	Male	553.38	-77.843	15.026	United-States	<=50K
34.454	Private	208967.333	HS-grad	9.469	Divorced	Handlers-cleaners	Not-in-family	White	Male	-262.039	-6.454	41.093	United-States	<=50K

Parameter Value Description

Operation **Swap data** Allows swapping data values between different entries. The operation will break correlations between the swapped fields and the stationary fields, but it will preserve some statistical properties of the datasets.

Parameter Value Description

Swap Rate (0 to 100%) **30** Percentage of the dataset records that will be affected by the operation

SwapAttributes **hours-per-week** Fields that will be affected by the data swap operation

relationship

Add Field

Remove Field

Fields are grouped If selected, all selected fields will be swapped as a group instead of being swapped individually

Swap Data

Figura 6.12: Permutación de datos

Dataset adultNamed.data

age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	income_class
38.757	State-gov	69296.251	Bachelors	13.107	Never-married	Adm-clerical	Not-in-family	White	Male	2363.015	18.286	43.25	United-States	<=50K
53.105	Self-emp-not-inc	65452.982	Bachelors	13.969	Married-civ-spouse	Exec-managerial	Husband	White	Male	553.38	-77.843	15.026	United-States	<=50K
34.454	Private	208967.333	HS-grad	9.469	Divorced	Handlers-cleaners	Not-in-family	White	Male	-262.039	-6.454	41.093	United-States	<=50K

Parameter Value Description

Operation **Swap data** Allows swapping data values between different entries. The operation will break correlations between the swapped fields and the stationary fields, but it will preserve some statistical properties of the datasets.

Parameter Value Description

Swap Rate (0 to 100%) **30** Percentage of the dataset records that will be affected by the operation

SwapAttributes **hours-per-week** Fields that will be affected by the data swap operation

relationship

Add Field

Remove Field

Fields are grouped If selected, all selected fields will be swapped as a group instead of being swapped individually

Swap Data

Figura 6.13: Generalización de datos

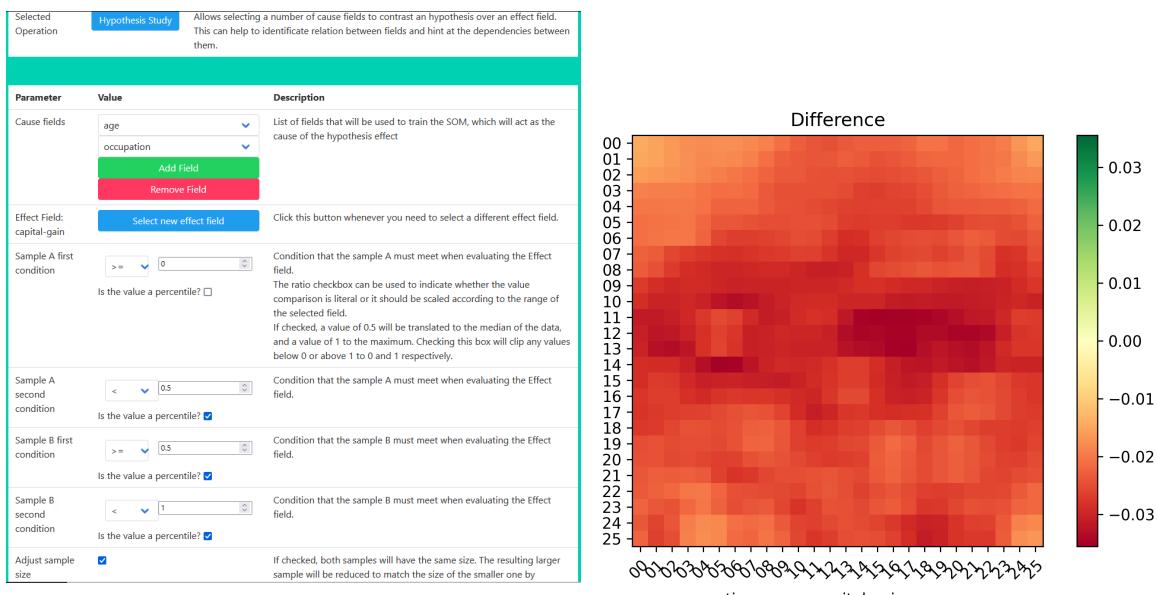
6.4.6. Análisis SOM

La sección de análisis SOM permite utilizar tanto redes SOM convencionales como algoritmos apoyados en árboles de taxonomía para poder analizar la privacidad del dataset seleccionado. En la sección 6.1.2 se detalla el funcionamiento de la red SOM.

Existen dos operaciones principales en esta sección, el agrupamiento de datos 6.14 y el análisis de hipótesis 6.15. La primera permite agrupar datos a través de una red SOM, o si el usuario lo desea, combinándolos con un algoritmo de distribución posterior que hace uso de árboles de taxonomía 6.1.2 para operar sobre los datos categóricos. Esta operación produce una página de resultados con todas las gráficas que representan el estado de la red y la opción de descargar una versión del dataset separada en grupos diferenciados.

La segunda operación requiere que se indique un campo denominado **efecto** y uno o más campos denominados **causa**. El campo **efecto** será acompañado de múltiples condiciones definidas por el usuario para delimitar dos muestras de datos separadas. En el caso de los atributos numéricicos se podrán indicar hasta cuatro operadores de comparación y límites numéricos literales o proporcionales al atributo. En el caso de los atributos categóricos, se debe proporcionar la lista de valores que formará cada muestra de datos. Adicionalmente, si el usuario lo desea, se le puede indicar al sistema que se asegure de tomar muestras del mismo tamaño descartando datos aleatorios de la muestra de mayor tamaño.

Una vez iniciado el proceso de entrenamiento, se entrenará una red SOM con el subconjunto de elementos que forman la **causa** de la hipótesis. El resultado será un mapa de activación de los campos de la causa contrastado con cada una de las particiones del dataset. Adicionalmente, se generará un mapa de calor para ilustrar la diferencia entre cada uno de las muestras definidos por el usuario.



(a) Análisis de hipótesis mediante el uso de redes SOM

(b) Resultado del análisis de hipótesis

Figura 6.15: Contraste de Hipótesis

6.4.7. Optimización de hiper-parámetros

Para facilitar el ajuste de los hiper-parámetros de la red SOM, la aplicación ofrece una funcionalidad de ajuste automático donde poder comparar el rendimiento de la combinación de múltiples parámetros de manera más intuitiva.

Este sistema toma un mínimo, máximo y valor de incremento para el valor de K, A y B (explicados en la sección 6.1.2). En este caso, será necesario aportar una clase de solución (clase Y) para poder evaluar el rendimiento de la operación mediante una correlación de los datos originales y los generados por la clasificación SOM. Una vez completado el proceso, la aplicación mostrará al usuario una representación visual del resultado del ajuste (figura 6.16) y permitirá también la opción de descargarlos como un archivo de texto.

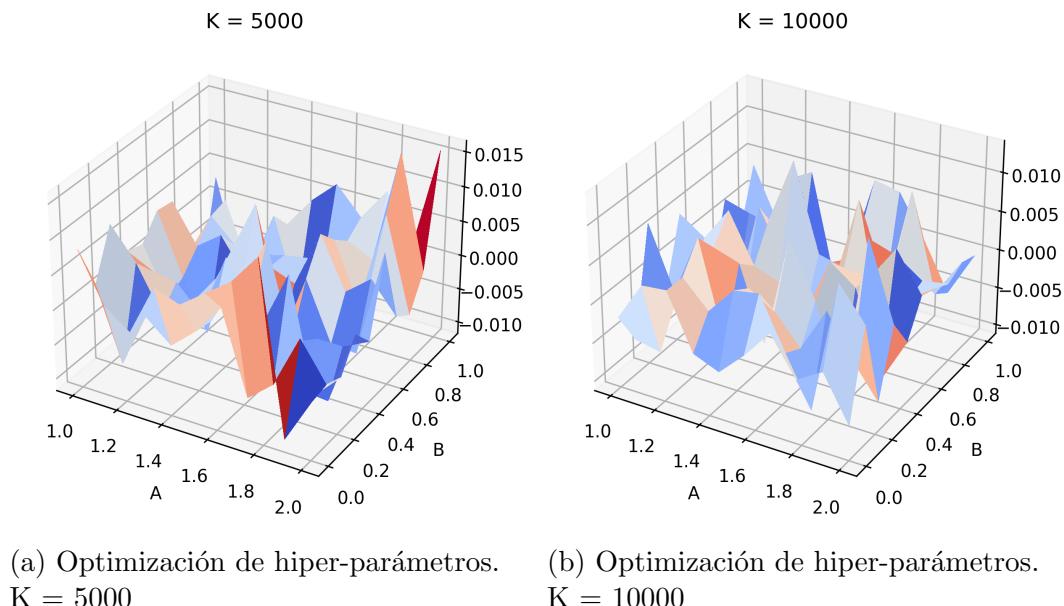


Figura 6.16: Optimización de hiper-parámetros.

Imágenes generadas por la aplicación para ilustrar el rendimiento de cada combinación de parámetros.

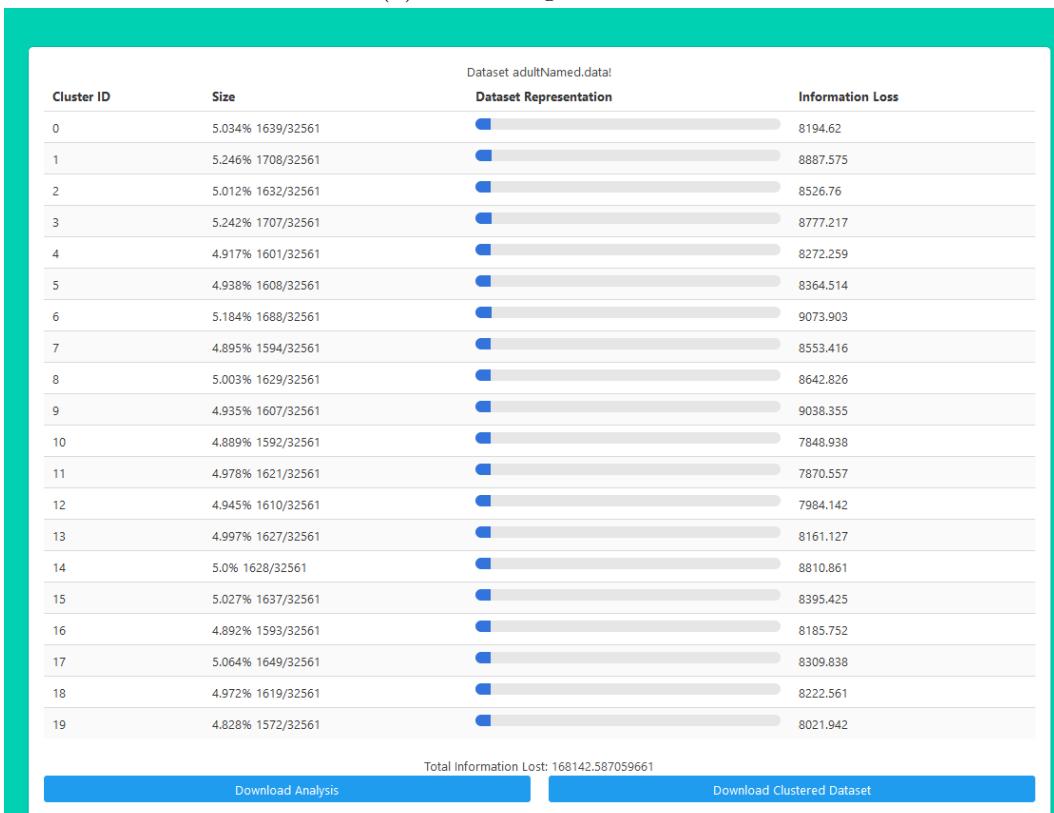
Parameter	Value	Description
Categorical Fields	Encoded	The system will automatically encode categorical fields by assigning each unique value a different numerical value, these values are equally distant and normalized.
Selected Operation	Clustering	Train the SOM network to clusterize the data and analyze the causes of the segregation obtained. Can be also used to cluster the dataset itself and ensure K-anonymity if needed.

Parameter	Value	Description
Y Class	None	This special field can be used to obtain additional error data, by comparing the resulting clustering against the values in this field
K:	100	Estimation of the desired elements per cluster, if "Ensure K" is selected, the algorithm will adjust the SOM result to ensure the K-anonymity
A:	1.2	A component of the learning rate (LR) function, which decreases along the iterations (I), following the function: $LR = A \cdot B^I / \text{entries}$
B:	0.3	B component of the learning rate (LR) function, which decreases along the iterations (I), following the function: $LR = A \cdot B^I / \text{entries}$
Epochs	0	Number of iterations over the dataset, leave at 0 to let the algorithm decide.
Sigma	1	Initial neighbourhood size.
Ensure K-anonymity	<input type="checkbox"/>	If checked, an additional adjustment stage will be performed to ensure that all clusters have at least K elements
Saving interval (0% to 100%)	10	The interval between saving points. The execution can be continued in the future from any of these saving points. Leave at 0% to disable save points completely

Train SOM

There are no training points available

(a) Clustering con SOM



(b) Resultado del Clustering

Figura 6.14: Análisis SOM

7. Construcción del Sistema

7.1. Entorno de Construcción

El sitio web se ha desarrollado desde Python por su versatilidad y capacidad de actuar como una interfaz entre múltiples tecnologías. Se ha utilizado PyCharm como IDE principal, junto a Visual Studio para definir el código del modelo C4 que acompaña a este documento. Gracias a la integración entre PyCharm y GitHub, el proyecto es público y se puede sincronizar automáticamente con el repositorio en la nube que ofrece GitHub. Los diagramas del modelo C4 se han construido utilizando la herramienta C4Builder¹.

7.2. Código Fuente

El sistema sigue la arquitectura del framework de flask para el desarrollo de aplicaciones, el directorio FlaskApp es la raíz del paquete y contiene los siete archivos de código que componen el esqueleto principal del servicio web. Para el apartado visual del sitio web se ha optado por el Bulma², que consiste de un framework de software libre cuya implementación incluye múltiples componentes de CSS de alto nivel.

El código fuente desarrollado en este proyecto se encuentra publicado en un repositorio público de GitHub³. La estructura principal del proyecto es la siguiente.

- **app.py** Se trata del punto de inicio de la aplicación, define la función para configurar, crear y lanzar el servicio web junto a la base de datos que la soporta. Hace uso del sistema de “Blueprints” para implementar todas las rutas que ofrece la aplicación desarrollada.
- **Authentication.py** Define las rutas para el registro de usuarios, acceso al servicio, desconexiones y encriptación de claves.
- **Backend.py** Implementa los algoritmos de las diferentes operaciones que existen. Gestiona el ciclo de vida y la creación de procesos en el sistema.
- **FrontEndWeb.py** Define todas las rutas que definen la funcionalidad del sistema web al completo. Esto incluye las diferentes operaciones sobre datos del sistema, subida de archivos, descarga de archivos y portal principal de la interfaz web.
- **models.py** Define todas las tablas de la base de datos que respalda el funcionamiento del sitio web.

¹C4Builder: <https://github.com/adrianvlupu/C4-Builder>

²Bulma: <https://bulma.io/>

³GitHub: <https://github.com/RafaelRossoGiner/Anonymization>

- **SelfOrganizingMap.py** Implementa la clase SOM que permite realizar todas las funciones relacionadas con el análisis de datos mediante mapas autorganizativos.
- **TaxonomyForest.py** Implementa la clase que representa un árbol de taxonomía para aplicar esta representación sobre los campos categóricos de un dataset.

A su vez, el directorio FlaskApp contiene cuatro sub-directorios donde se define el contenido que será visualizado por el navegador web cuando el usuario interactúa con el servicio ofrecido.

- **favicon.io** Contiene el icono de la aplicación en múltiples formatos, siguiendo los estándares definidos por los principales navegadores web actuales. Este icono se utiliza, por ejemplo, para representar la pestaña que apunta al servicio web en el navegador.
- **instance** En este directorio se sitúan los archivos de configuración. Como por ejemplo, *flask_settings.py*, que servirá para poder añadir parámetros adicionales a la aplicación de Flask.
- **static** Es el directorio donde se configuran los archivos generados dinámicamente para ser visualizados desde el navegador. En el contexto de la aplicación se utiliza para almacenar las imágenes que cada usuario genera como resultado de las operaciones realizadas.
- **templates** Aquí se guardan todas las plantillas html definidas para generar el contenido web en cada una de las rutas de la aplicación. Están escritas en el lenguaje de jinja2 que permite insertar pseudo-código de Python para generar código html dinámico en función de los datos del servidor.

7.2.1. Optimización en operaciones sobre grandes conjuntos de datos

Debido a la naturaleza del problema propuesto, es necesario operar con grandes volúmenes de datos lo que puede generar problemas de optimización en tiempo y memoria. Para solventar esta situación se utilizan dos paquetes de Python diferentes.

Numpy proporciona una librería para operar con matrices de forma más eficiente en tiempo y espacio. Además de implementar listas mucho más rápidas que las de Python y paralelizar el computo de operaciones con múltiples hilos, se aprovecha del sistema de archivos para permitir trabajar con volúmenes de datos mayores que la memoria del sistema. Si es necesario, **Numpy** utilizará el disco duro para particionar grandes volúmenes de datos y trabajar de forma secuencial sobre cada uno de ellos usando la memoria principal del sistema. Al implementar todas las operaciones con esta librería, el tamaño máximo de los dataset solo se ve limitado por los requisitos del sistema de archivos de la máquina y la cantidad de espacio de disco libre que se le han asignado a la aplicación.

Aún así, incluso con la implementación de **Numpy**, el intérprete de Python sigue siendo poco óptimo en comparación con lenguajes compilados como C++. Para poder

optimizar los procesos y reducir el tiempo de cómputo de las operaciones, se utiliza la librería **Numba** para mejorar la eficiencia de los algoritmos principales del sistema. La principal funcionalidad de **Numba** es hacer de interfaz entre Python y el código máquina del sistema operativo, incluye un compilador propio capaz de pre-compilar funciones de Python y producir un programa ejecutable que lleve a cabo la misma operación mediante un proceso externo. Para que esto sea posible es necesario definir correctamente la representación de todas las variables y datos de la función que van a ser pre-compiladas, intentando maximizar la optimización del programa resultante.

Finalmente en el apartado de optimización, es necesario implementar un sistema de paralelización de alto nivel que nos permita responder peticiones web mientras que se está ejecutando un cálculo complejo como el entrenamiento de una red SOM. Para ello se ha implementado un sistema de múltiples procesos con la interfaz **Futures** que proporciona la API de Python. Esta interfaz permite crear un grupo de procesos que llevarán a cabo el control de todas las operaciones delegadas del hilo principal, mientras el hilo principal se ocupa exclusivamente de responder a las peticiones webs de los usuarios. Esta arquitectura permite además utilizar múltiples núcleos del procesador durante la ejecución del programa, incrementando significativamente el rendimiento del sistema.

7.3. Scripts de Base de datos

El proyecto contiene dos algoritmos para trabajar sobre la base de datos seleccionada y el sistema de archivos que la respalda. El objetivo es minimizar el espacio ocupado por los datos almacenados, evitar posibles errores y facilitar la instalación en una nueva máquina.

7.3.1. Creación de la base de datos

Este script se ejecuta cada vez que se inicia la aplicación y lleva a cabo su función durante la inicialización de la aplicación web. El objetivo es crear todas las tablas que componen la base de datos necesarias que funcione el servicio web, configurando cada uno de los campos, claves primarias y foráneas de cada entidad definida. La estructura que tendrá la base de datos viene definida en el archivo *models.py*.

Además, este script comprueba la existencia de los cinco directorios de archivos configurados para la aplicación en *Config.py*. Si falta alguno de ellos, el sistema lo creará automáticamente antes de finalizar la creación de la aplicación de Flask.

7.3.2. Sincronización de la base de datos

Se ejecuta después del algoritmo previamente mencionado y su función es eliminar registros o documentos aislados. Se define como registro aislado cualquier entrada de la tabla dataset cuyo documento asociado no se encuentre en el directorio indicado.

De forma simétrica, se considera como un documento aislado cualquier archivo que se encuentre en el directorio pero no tenga asociado ningún elemento en la base de datos.

8. Pruebas del Sistema

8.1. Estrategia

Con el objetivo de detectar errores no solo en código nuevo sino en el flujo completo de la operación, se ha seguido una estrategia de integración continua. La metodología a seguir durante el proyecto ha sido prototipar de forma temprana para obtener sistemas funcionales lo antes posible. Una vez obtenido el prototipo de base, se siguen las listas de comprobación para depurar el programa y se itera sobre ellas cada vez que se añade una nueva funcionalidad o cambio significativo al mismo.

Las pruebas se han desarrollado principalmente en un servidor local, utilizando puertos internos para conectar la aplicación, base de datos y navegador web. Adicionalmente se ha utilizado el servidor de depuración que ofrece Flask para poder ejecutar la aplicación de forma sencilla y facilitar la depuración de la misma.

Con el objetivo de poder verificar el correcto funcionamiento del sistema al tratar tanto con datos numéricos como con datos categóricos, se ha utilizado el conocido dataset *Adult* para depurar el sistema.

8.2. Pruebas Unitarias

Se ha utilizado un marco de pruebas en Python para depurar las funciones implementadas en el sistema. Todo algoritmo incluido en el sistema pasa por un proceso de depuración antes de ser incluido en el proyecto final. Todas las funciones que aparecen en la sección de privatización deben ser correctas y eficientes.

Para verificar la corrección de estos algoritmos se emplean conjuntos de datos conocidos y se compara el resultado con la solución aportada por aplicaciones de confianza. Por otro lado, la eficiencia se mide mediante un estudio comparativo del algoritmo implementado en Python y con el uso de Numba para pre-compilar la función cuando esta es ejecutada.

8.3. Pruebas de Integración

Durante la fase de desarrollo, cada vez que se implementaba un nuevo módulo, ha sido necesario iterar sobre las listas de comprobación de todos los casos de uso relevantes para verificar la integridad del resto de módulos relacionados.

De esta forma, se han podido identificar errores de diseño o implementación de forma temprana. Esta metodología de integración ha ayudado a realizar acciones pre-

ventivas para facilitar la escalabilidad y modularidad de cada una de las partes del código.

8.4. Pruebas de Sistema

8.4.1. Pruebas Funcionales

Para comprobar que el sistema cumple con los requisitos, se han realizado iteraciones manuales siguiendo los flujos normales y alternativos mencionados en la sección 5.2 para los casos de uso.

Mediante listas de comprobación se han ido recopilando los resultados de las pruebas para poder identificar flujos problemáticos. Además, estas han servido para descubrir nuevos flujos alternativos a implementar y los flujos de error que necesitan protegerse.

8.4.2. Pruebas No Funcionales

El software ha sido probado en múltiples dispositivos y navegadores, para asegurar la adaptabilidad a diferentes sistemas y el comportamiento de las diferentes pantallas del sistema.

Se han realizado pruebas de manipulación de URL y manipulación de peticiones http, para asegurar que el sistema no permite el acceso a ningún dato privado o perteneciente a otro usuario.

El sistema ha sido además probado en un entorno con múltiples sesiones simultáneas, que verifiquen la capacidad de respuesta y multi-procesamiento de la aplicación.

8.5. Pruebas de Aceptación

Para las pruebas de aceptación, se han ejecutado de nuevo todas las pruebas mencionadas en los apartados previos. Esta vez utilizando una nueva instalación del proyecto al completo y en un entorno de producción final. Mediante el uso de gunicorn, se ha lanzado la aplicación en un entorno de producción real utilizando puertos externos en la conexión, a diferencia del servidor local de depuración de Flask que se ha utilizado durante el desarrollo.

9. Despliegue del Sistema

9.1. Arquitectura Física

La arquitectura física requiere de un servidor que ejecute el servicio web y un motor de base de datos. Estas funciones pueden ser llevadas a cabo por la misma máquina, utilizando una red privada para interconectar ambos sistemas o distribuirse en múltiples dispositivos según las necesidades del entorno. Tal y como se indica en la sección 9.2.1, también es posible virtualizar cualquiera de los dos sistemas en caso de que sea necesario. Las especificaciones del hardware necesario deberían escalar acorde al número de usuarios y el flujo de operaciones en el sistema. En la sección 3.5 se describe una estimación de las características de un servidor adecuado para esta tarea.

9.2. Instrucciones de despliegue

A continuación, se describen las instrucciones de instalación del sistema sobre la infraestructura física descrita anteriormente.

9.2.1. Docker

Con el objetivo de simplificar la instalación de la aplicación web y hacerla más accesible a múltiples dispositivos y entornos de ejecución, se ha construido un contenedor Docker¹ para virtualizar el sistema completo. Se ha decidido utilizar Docker-Compose específicamente para mantener una arquitectura modular y con mayor portabilidad, siguiendo las recomendaciones especificadas en los estándares de Docker. En el archivo de configuración se definen dos servicios, la base de datos (MariaDB) y la aplicación a desarrollar (AnonymizationAPI), los detalles del documento se pueden observar en el anexo A.1.

La aplicación web ejecuta como un servicio gracias a gunicorn², que se trata de un potente servidor WSGI de producción para plataformas unix que permite múltiples conexiones simultáneas de forma eficiente. El contenedor diseñado incluye un archivo de configuración donde poder modificar tanto las opciones del servidor de gunicorn como las opciones de la aplicación Flask A.2. La base de datos está definida como una dependencia de la aplicación, de forma que esta siempre se inicie primero. Adicionalmente, se le ha añadido una condición de estado para que solo se lance el servicio web una vez se haya podido establecer una conexión exitosa con la base de datos.

Finalmente, ambos servicios están conectados entre sí mediante una red virtual interna, lo que permite que la aplicación web se comunique de forma local con la base

¹Docker: <https://www.docker.com/>

²Gunicorn: <https://gunicorn.org/>

de datos sin necesidad de exponer ningún puerto al exterior. Esto mejora la eficiencia del sistema y enmascara puertos innecesarios para evitar problemas de seguridad.

La base de datos seleccionada para el contenedor es MariaDB ³ 11.1.2, y mediante los parámetros de configuración se puede incluir tanto al usuario root como a uno adicional que tendrá todos los permisos por defecto. Este usuario será el que utilice el servicio web por lo cuál tanto el nombre como la contraseña debe coincidir.

Por otro lado, el servicio web se construye como un nuevo contenedor utilizando la arquitectura clásica de Docker, siguiendo las instrucciones definidas en Dockerfile [A.2](#). Este archivo indicará al sistema virtualizado cómo debe lanzar la aplicación y conectarla al servicio de la base de datos mencionado en el punto anterior. En términos generales, Docker realizará las siguientes operaciones.

1. Instalar la imagen base con el intérprete de Python 3.7.8.
2. Añadir a la imagen los contenidos el directorio de configuración y el directorio de la aplicación Flask.
3. Descargar e instalar la versión específica de connector/C de MariaDB necesaria para el proyecto. Este conector de C es una dependencia del conector de Python, necesario en los requisitos especificados para pip.
4. Actualizar pip e instalar todos los requisitos indicados en el archivo requirements.txt ([D.1](#)).
5. Limpiar directorios temporales innecesarios.
6. Ejecutar gunicorn, apuntando al punto de inicio de la aplicación Flask.

En el anexo, se pueden encontrar más detalles acerca de este proceso [A.2](#).

9.2.2. Requisitos previos

Se recomienda instalar el sistema mediante el servicio ofrecido por Docker [9.2.1](#), debido a la facilidad y portabilidad que aporta. En el caso de querer modificar el software, la arquitectura de Docker-compose permite modificar el motor de base de datos y los múltiples archivos de configuración permiten personalizar considerablemente la aplicación.

En caso de querer realizar una instalación manual, deben seguirse los siguientes puntos.

Cliente

- Navegador web moderno con soporte para CSS. Para los navegadores más comunes en el mercado actual se recomienda utilizar al menos la siguiente versión.
 - Chrome 87 o superior.

³MariaDB: <https://mariadb.org/>

- Firefox 89 o superior.
- Safari 12 o superior.
- Edge 87 o superior.
- Opera 77 o superior.

Servidor

- Entorno de Python (versión 3.7.8). Debe contener los paquetes mencionados en la tabla de requisitos de [D.1](#)
- Motor de base de Datos relacional compatible con SQLAlchemy. Se recomienda utilizar MariaDB (versión 11.1.2)
- Servidor capaz de alojar aplicaciones WSGI. Se recomienda el uso de Gunicorn o aplicaciones similares.

9.2.3. Inventario de componentes

En la versión presentada del proyecto se incluyen los siguientes componentes:

- Servicio web de análisis y privatización de datos, desarrollado en Python como aplicación WSGI.
- Documento detallando todos los paquetes de Python necesarios para el correcto funcionamiento de la aplicación. Este archivo puede usarse junto a pip para instalar automáticamente todos los paquetes.

Adicionalmente, si el contenedor Docker asociado al proyecto contiene.

- MariaDB 11.1.2.
- Gunicorn 21.2.0.
- Entorno de ejecución de Python ya preparado y equipado con todos los paquetes necesarios [D.1](#) para lanzar el servicio web.

9.2.4. Procedimientos de instalación

Para la instalación del servicio es necesario crear un entorno compatible con Python y tener instalados los paquetes y tecnologías indicadas en los apartados previos. Durante la primera ejecución del sistema, es necesario proveer de una base de datos y un usuario con acceso a los permisos de creación, destrucción y modificación de tablas. Una vez creado el usuario y la base de datos, deben indicarse en el archivo Config.py situado en el directorio raíz de la aplicación.

El proceso principal de la aplicación debe lanzarse desde el servidor WSGI, bien tomando el valor de la variable *application* que se puede encontrar en *app.py* o, si se prefiere usar una arquitectura de fábrica, ejecutando la función *create_app()* del

mismo módulo. La aplicación se encargará de crear todas las tablas necesarias para su funcionamiento en la base de datos y generar los directorios para almacenar los archivos del servicio web. Estos directorios son configurables desde el documento Config.py y la aplicación debe disponer de los permisos necesarios para crear, destruir y editarlos junto a su contenido.

9.2.5. Pruebas de implantación

Para confirmar que el sistema se ha instalado correctamente y es apto para el uso del cliente, se pueden llevar a cabo las siguientes acciones. Están diseñadas para probar cada parte distintiva del sistema que pueda verse afectada por la instalación o entorno en el que se este ejecutando. Si se completa con éxito este proceso, se puede considerar que la instalación ha sido correcta y el sistema está listo para comenzar a usarse.

1. Acceder al dominio donde se aloja la aplicación a través de un navegador web.
2. Acceder a una ruta del sistema sin iniciar sesión, por ejemplo /profile. La aplicación debe redirigir al navegador a la página de inicio de sesión.
3. Crear una nueva cuenta en el sistema y conectarse a ella, la casilla “Remember me” debe estar desmarcada.
4. Cerrar el navegador y volver acceder a la ruta /profile. La aplicación debe redirigir de nuevo al navegador a la página de inicio de sesión.
5. Iniciar sesión y acceder a /profile. Modificar alguno de los datos de la cuenta (como nombre o email) y subir un dataset al sistema.
6. Subir árboles de taxonomía para el dataset que se ha añadido o utilizar la opción de generación automática.
7. Tratar de descargar de nuevo el dataset.
8. Acceder a la sección de Análisis SOM y ejecutar un proceso de clustering, utilizando árboles de taxonomía para analizar los campos categóricos.
9. Descargar los resultados.
10. Acceder a la sección de Análisis SOM y ejecutar un contraste de hipótesis, omitiendo los campos categóricos.
11. Descargar los resultados.
12. Desconectarse del servicio y tratar de acceder de nuevo a la ruta /profile (o cualquier otra del sistema).

9.3. Instrucciones para la operación del sistema y mantenimiento del nivel de servicio

Logs

El archivo de configuración permite activar un sistema de Log para la aplicación, este sistema es configurable e incluye cuatro opciones:

- **LOGGING_ENABLED:** *True* o *False*, indica si esta habilitado la creación de logs.
- **LOGGING_LEVEL:** 0 a 5 indica el nivel de importancia que debe tener un evento para que se registre.
 5. 0 DISABLED. Deshabilita los logs, es similar a LOGGING_ENABLED = *False*.
 6. 1 DEBUG. No debería usarse a menos que se estén depurando programas muy específicos. Puede aumentar exponencialmente el tamaño de los archivos.
 7. 2 INFO. Incluye información del sistema, tanto de errores como de funcionamiento estable.
 8. 3 WARNING. Incluye información de eventos inesperados o que pueden causar problemas en el futuro.
 9. 4 ERROR. Indican qué ha ocurrido un problema y el software ha sido incapaz de realizar alguna operación.
 10. 5 CRITICAL. Indica un error importante que posiblemente detenga la ejecución del programa.
- **LOGGING_TRACEBACK:** *True* o *False*, indica si los logs deberían incluir la cadena de llamadas (traceback) o no.
- **LOGGING_FILENAME:** indica el nombre que tendrá el archivo donde se guarda el log. Este archivo es sobreescrito cada vez que se lanza la aplicación, por lo que sería necesario renombrarlo o moverlo si se quiere mantener una copia de logs previos.

Backups

Es recomendable realizar una copia de seguridad de los datos de los usuarios, para poder evitar ciertos tipos de ataques informáticos y recuperar el estado del servicio si ocurriese algún error inesperado.

Si se está utilizando Docker, tal y como se indica en la sección de despliegue [9.2.1](#), es suficiente con realizar una copia del contenedor. El contenedor contiene toda la información de usuarios y es fácilmente portable al poder integrarse fácilmente como sistemas automatizados o repositorios remotos.

Si se ha realizado una instalación local del sistema, será necesario copiar la base de datos, que en el caso de MariaDB es tan sencillo como ejecutar el comando `ma-riabackup -backup -target-dir [directorio]`. Además, hay que hacer una copia de los documentos almacenados en el sistema, copiando los directorios de datos permanentes [6.3.2](#) indicados en la configuración de la aplicación [6.2](#).

Parte III

Epílogo

10. Conclusiones

10.1. Objetivos alcanzados

Se ha logrado generar un servicio web robusto que permita a múltiples usuarios realizar operaciones de privatización, analizar datasets y compartirlos de forma sencilla e intuitiva. La aplicación hace más accesible el uso de estas técnicas sobre cualquier dataset de forma generalizada, mejorando la privacidad de los datos y facilitando el uso de los mismos en proyectos de ciencias abiertas.

La aplicación se ha decidido implementar mediante una arquitectura web de forma que sea accesible desde cualquier dispositivo sin necesidad de instalar software de ningún tipo. Es un medio muy común, intuitivo y que fomenta la participación e integración del sistema en entornos de trabajo.

10.2. Lecciones aprendidas

Durante el desarrollo de este proyecto, se han empleado múltiples tecnologías y sistemas completamente nuevos, empezando por Flask como framework de desarrollo. Esto ha fomentado el aprendizaje de arquitecturas web, sistemas de peticiones REST y gestión de servidores.

Este proyecto ha servido como guía para estudiar las técnicas de preservación de la privacidad que existen actualmente, principalmente aquellas relacionadas con la enmascaramiento y generalización de datos.

Asimismo se han desarrollado las aptitudes necesarias para el despliegue de aplicaciones web y creación de contenedores virtuales que proporcionen todos los servicios necesarios para su correcto funcionamiento.

10.3. Trabajo futuro

El proyecto aún tiene varios puntos a mejorar y expandir acerca de la funcionalidad del mismo. Como trabajo futuro uno de los puntos que se plantean es la implementación de una arquitectura REST, de forma que la aplicación disponga de una API que acepte y responda peticiones web sin necesidad de usar una interfaz gráfica. Esto facilitaría el uso de la aplicación mediante sistemas automatizados y servicios de terceros.

A pesar de que se da soporte a los datos categóricos de múltiples formas, uno de los puntos más interesantes sería poder operar con datos categóricos directamente sobre la red SOM. En este campo existen ya algunos algoritmos conocidos como MixSOM,

NCSOM y FMSOM ([del Coso et al., 2015](#)); aunque lo más interesante sería investigar la posibilidad de integrar los árboles de taxonomía propuestos en este documento como método para calcular la distancia entre nodos.

Otro aspecto interesante, sería la aplicación de ruido en campos categóricos basado en el sistema de árboles de taxonomía implementado para la aplicación. Estudios recientes sobre la aplicación de algoritmos a datos nominales de este tipo se desarrollan en profundidad en ([Rodriguez-Garcia et al., 2017](#)) y ([Rodriguez-Garcia et al., 2019](#)), y serían una gran contribución al conjunto de herramientas que proporciona el sistema actual.

Información sobre Licencia

Información de la Licencia

“Aplicación de técnicas de preservación de la privacidad a anonimización de datos en registros para el análisis de secuencias” es un software de código abierto distribuido bajo la Licencia Pública General de GNU versión 3.0 (GPL-3.0¹) o cualquier versión posterior, según publicada por la Free Software Foundation.

Esto significa que tienes los siguientes derechos y obligaciones:

- Tienes la libertad de usar, modificar y distribuir este software.
- Cualquier modificación o mejora que realices en el software también debe ser distribuida bajo la licencia GPL-3.0.
- Si distribuyes este software a otros, debes proporcionarles acceso al código fuente completo correspondiente.

Para obtener el texto completo de la licencia GPL-3.0, por favor consulta el archivo *LICENSE* incluido con este software.

“Aplicación de técnicas de preservación de la privacidad a anonimización de datos en registros para el análisis de secuencias” se proporciona “tal cual” sin garantías de ningún tipo. Por favor, lee atentamente la licencia GPL-3.0 para obtener más detalles sobre tus derechos y responsabilidades.

¹<http://www.gnu.org/licenses/gpl.html>

Bibliografía

- Arcolezi, H. H. (2022). Production of categorical data verifying differential privacy: Conception and applications to machine learning.
- Brown, S. (2022). The C4 model for visualising software architecture.
- Caballero-Hernández, J. A., Palomo-Duarte, M., y Dodero, J. M. (2019). Evaluación de competencias en serious games mediante analítica de aprendizaje con process mining.
- del Coso, C., Fustes, D., Dafonte, C., Nóvoa, F. J., Rodríguez-Pedreira, J. M., y Arcay, B. (2015). Mixing numerical and categorical data in a self-organizing map by means of frequency neurons. *Applied Soft Computing*, 36:246–254.
- EU (2020). European Union Open Science Portal.
- Hulle, M. M. V. (2012). Handbook of Natural Computing. Self-Organizing Maps. http://www.pspc.unige.it/~drivsco/Papers/VanHulle_Springer.pdf.
- Ji-Won Byun, Ashish Kamra, E. B. y Li, N. (2003). Efficient k-Anonymization Using Clustering Techniques. https://www.cs.purdue.edu/homes/ninghui/papers/clustering_dasfaa07.pdf.
- Kim, J. (1986). Method for limiting disclosure in microdata based on random noise and transformation. http://www.asasrms.org/Proceedings/papers/1986_069.pdf.
- Lin, J.-L. y Wei, M.-C. (2008). An efficient clustering method for k-anonymization. page 46–50.
- Rashid, A. H. y Yasin, N. B. M. (2015). Privacy preserving data publishing: Review. *International Journal of Physical Sciences*, 10:239–247.
- Rodriguez-Garcia, M., Batet, M., y Sánchez, D. (2017). A semantic framework for noise addition with nominal data. *Knowledge-Based Systems*, 122:103–118.
- Rodriguez-Garcia, M., Batet, M., y Sánchez, D. (2019). Utility-preserving privacy protection of nominal data sets via semantic rank swapping. *Information Fusion*, 45:282–295.
- Seh, A. H. (2020). Healthcare data breaches: Insights and implications.
- UCA (2023). Tabla Capítulo VI. <https://personal.uca.es/wp-content/uploads/2022/01/COSTES-APROXIMADOS-CAPITULO-VI-2022.pdf>.
- Zaman, A. N. K. y Obimbo, C. (2014). Privacy preserving data publishing: A classification perspective. *International Journal of Advanced Computer Science and Applications*, 5.

Álvarez Ayllón, A., Palomo-Duarte, M., y Dodero, J.-M. (2022). `scjpresq`/`scj`: Discovery of multidimensional equally-distributed dependencies via quasi-cliques on hypergraphs. *IEEE Transactions C4on Emerging Topics in Computing*, pages 1–16.

Parte IV

Anexos

A. Manual de usuario

Las instrucciones de uso del software se detallan a continuación. Este manual se dirige al usuario final del software objetivo de este proyecto.

A.1. Introducción

La aplicación web ofrece la posibilidad de subir múltiples conjuntos de datos (datasets) y operar sobre ellos. El sistema permite compartir datasets, operar sobre ellos mediante técnicas clásicas, compararlos y realizar análisis basados en aprendizaje computacional.

A.2. Instalación

El software no requiere de ninguna instalación por parte del usuario, simplemente tener acceso al dominio donde se aloja el servicio web y acceder al mismo mediante un navegador web moderno de acuerdo a las especificaciones indicadas en la sección 9.2.2.

En el caso de querer ejecutar el servicio de forma local, es posible hacerlo siguiendo las instrucciones de despliegue indicadas en la sección 9.2. En caso de necesitar configurar el entorno de ejecución de Docker, el contenedor se ha construido a partir de los siguientes archivos:

Extracto de código A.1: Docker-Compose

```
version: '3.9'

networks:
  privatenet:
    external: false
    driver: bridge

services:
  mariadb:
    image: mariadb:11.1.2-jammy
    restart: always
    platform: linux/amd64
    networks:
      - privatenet
    ports:
      - 3306:3306
    expose:
      - 3306
    volumes:
      - mariadb_data:/var/lib/mysql
    environment:
```

```

- MYSQL_ROOT_PASSWORD=root
- MYSQL_PASSWORD=anonymizationPswd
- MYSQL_USER=anonymizationUser
- MYSQL_DATABASE=anonymization
healthcheck:
  test: ["CMD", "mariadb", "-h", "127.0.0.1", "-u", "anonymizationUser",
    ↪ "-panonymizationPswd"]
  interval: 5s
  timeout: 5s
  retries: 5
  start_period: 10s

anonymization_docker:
  build:
    context: .
    dockerfile: docker/anonymization_docker/Dockerfile
  platform: linux/amd64
  networks:
    - privatenet
  ports:
    - 8080:8080
  depends_on:
    mariadb:
      condition: "service_healthy"
  volumes:
    - type: bind
      source: ./anonymization_docker
      target: /anonymization_docker/anonymization_docker
    - type: bind
      source: ./config
      target: /anonymization_docker/config
  command: >
    sh -c "cd ./anonymization_docker && gunicorn --bind 0.0.0.0:8080 --reload
      ↪ app:application"

volumes:
  mariadb_data:

```

Extracto de código A.2: Dockerfile

```

FROM python:3.7.8

ENV INSTALL_PATH /anonymization_docker
RUN mkdir -p $INSTALL_PATH
WORKDIR $INSTALL_PATH

ADD Pipfile .
RUN mkdir -p .venv

ADD config ${INSTALL_PATH}/config
COPY anonymization_docker ${INSTALL_PATH}/anonymization_docker

RUN apt-get update && apt-get install -y gcc wget
RUN wget https://dlm.mariadb.com/2678574/Connectors/c/connector-c-3.3.3
  ↪ /mariadb-connector-c-3.3.3-debian-bullseye-amd64.tar.gz -O - | tar -zxf -

```

```

    ↢ --strip-components=1 -C /usr

RUN pip install --upgrade pip
RUN pip3 install pipenv==2022.1.8

COPY ./requirements.txt $INSTALL_PATH/requirements.txt

RUN pip3 install --no-cache-dir -r requirements.txt
RUN rm -rf ~/.cache
RUN rm -rf /tmp/*

CMD gunicorn -b 0.0.0.0:8080 --access-logfile - "anonymization_docker.app:run_app()"

```

A.3. Uso del sistema

Para utilizar el servicio, el primer paso es acceder a la aplicación y crear una nueva cuenta, esta debe tener un nombre y correo electrónico asociado único.

Una vez hemos accedido al porta principal, es posible trabajar con los datasets públicos del sistema o subir un dataset propio sobre el que trabajar.

A.3.1. Perfil de usuario

Una vez se accede a la cuenta de usuario, aparece el perfil de usuario. Esta página muestra todos los datasets a los que se tiene acceso y la información de la cuenta asociada. Desde esta ventana es posible modificar los datos de la cuenta e incluso eliminarla si fuera necesario, eliminar una cuenta borra del sistema todos los datasets que le pertenezcan al usuario.

A.3.2. Subir un dataset

Para subir un nuevo dataset pulsamos el botón que aparece en la esquina superior derecha “Profile” y nos dirigimos a la sección de datasets, donde habrá un botón que nos ofrezca la posibilidad de subir nuevos documentos.

Todos los datasets a los que tengamos acceso se podrán visualizarse y ser descargados desde el perfil. Adicionalmente, podemos eliminar y modificar la privacidad de aquellos datasets que nos pertenezcan. Finalmente, existe una opción para gestionar los árboles de taxonomía del dataset seleccionados.

Cada dataset puede tener un árbol de taxonomía por cada atributo categórico que posea, estos árboles se definen a partir de la sintaxis especificada en [6.1.2](#). Es posible subir, editar, eliminar y descargar cada uno de los documentos que definen el conjunto de árboles del dataset. Además, el sistema informará de qué campos aún no tienen un árbol asignado y ofrecerá la posibilidad de generarlos de forma automática. Un árbol

de taxonomía generado de forma automática no es más que un árbol de altura 1 donde todos los posibles valores del campo son hijos de la raíz.

A.3.3. Privatización de Datasets

Desde el portal principal, la opción de privatización de datasets permite realizar cuatro operaciones sobre cualquier conjunto de datos.

- Intercambio de datos: Intercambio de valores entre los datos de uno o más campos del dataset.
- Adición de ruido: Añade el tipo de ruido seleccionado sobre el conjunto de datos.
- Eliminación de datos: Elimina un campo o un intervalo de datos del dataset.
- Generalización de datos: Genera intervalos para generalizar los datos, indicando a qué grupo pertenecen.

Todas las operaciones son parametrizables y el sistema incluyen una descripción de cada parámetro para facilitar su correcta utilización.

A.3.4. Análisis SOM

Desde el portal principal, la opción de análisis SOM permite realizar tres operaciones sobre cualquier conjunto de datos.

- Clustering: El sistema utiliza una red SOM para tratar de separar todos los registros del dataset en diferentes grupos. Adicionalmente se pueden utilizar árboles de taxonomía para aportar información semántica a los campos categóricos y clasificar mediante un algoritmo de k-anonimidad los registros del dataset.
- Contraste de Hipótesis: Permite seleccionar un subconjunto de los campos del dataset y contrastar su relación frente a un conjunto de valores de otro de los campos del dataset.
- Ajuste de hiperparámetros: Con el fin de facilitar la parametrización del análisis, esta opción permite definir el dominio de cada uno de los parámetros y realizar un estudio comparativo entre ellos.

Todas las operaciones son parametrizables y el sistema incluyen una descripción de cada parámetro para facilitar su correcta utilización.

B. Diagramas de diseño

B.1. Diagramas del modelo C4

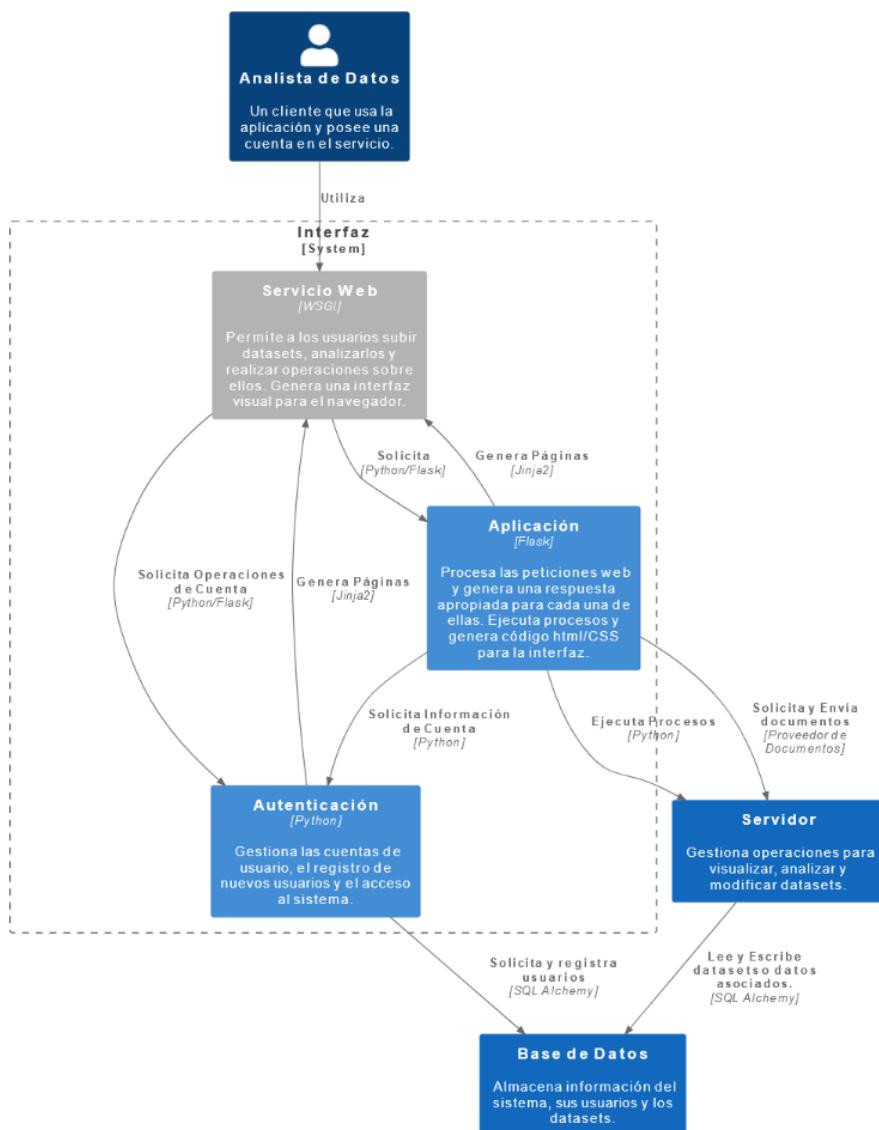


Figura B.1: Interfaz

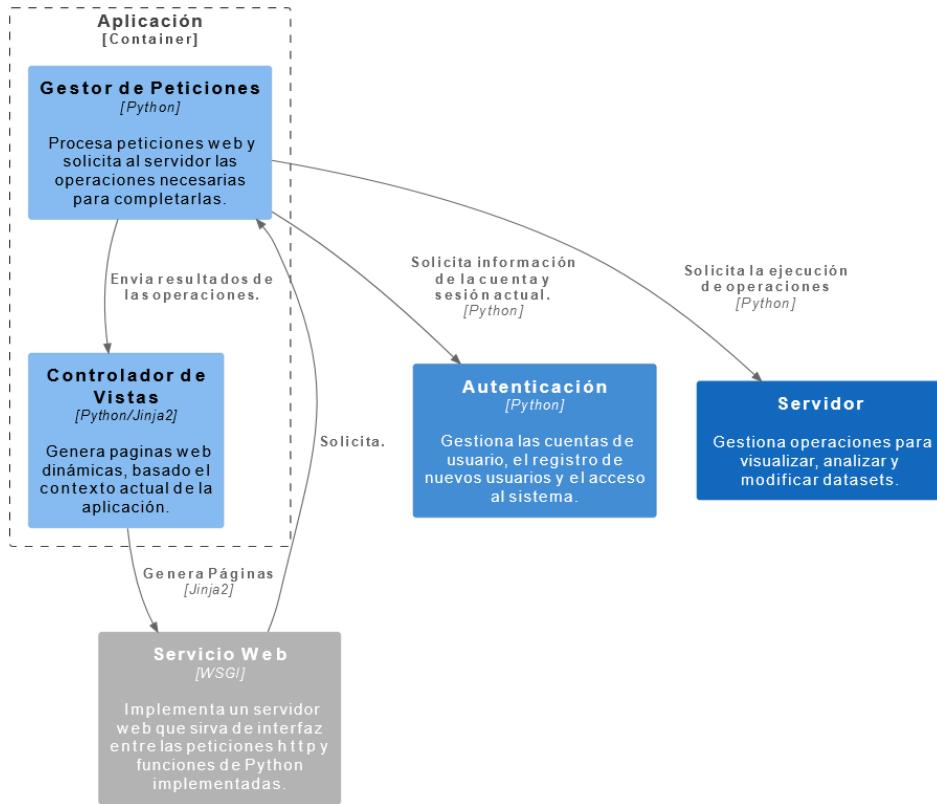


Figura B.2: Aplicación

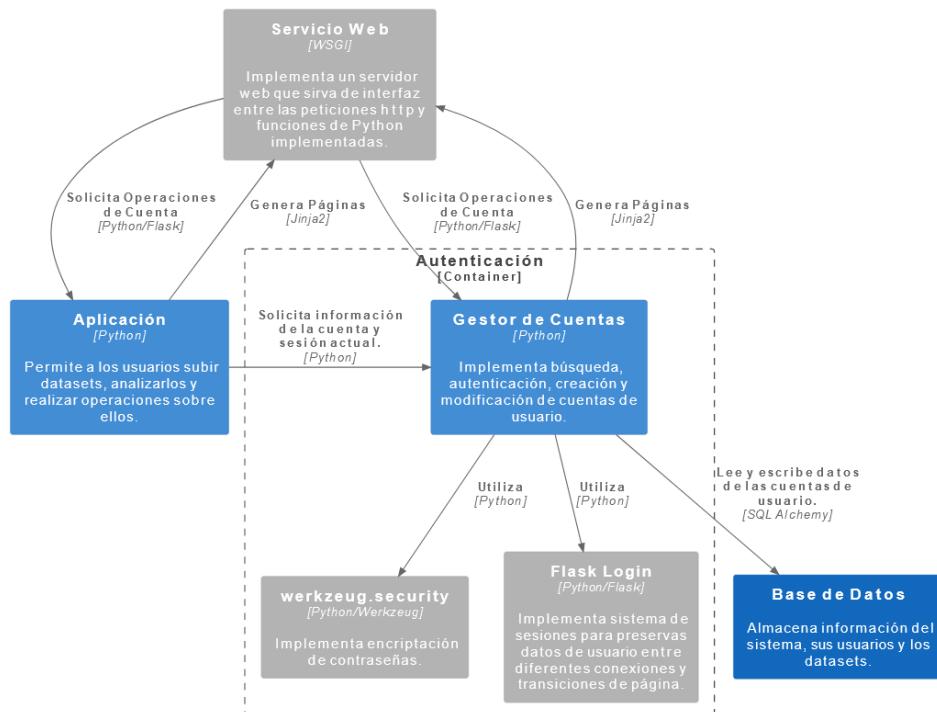


Figura B.3: Autenticación

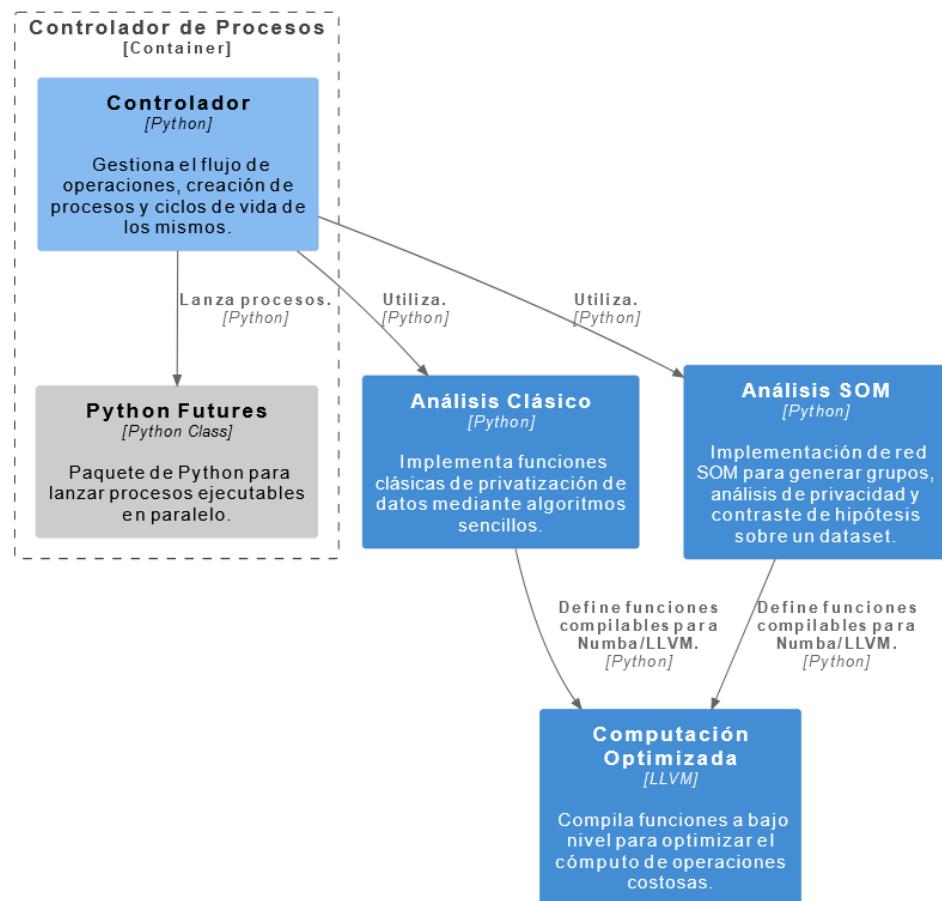


Figura B.4: Controlador de Procesos

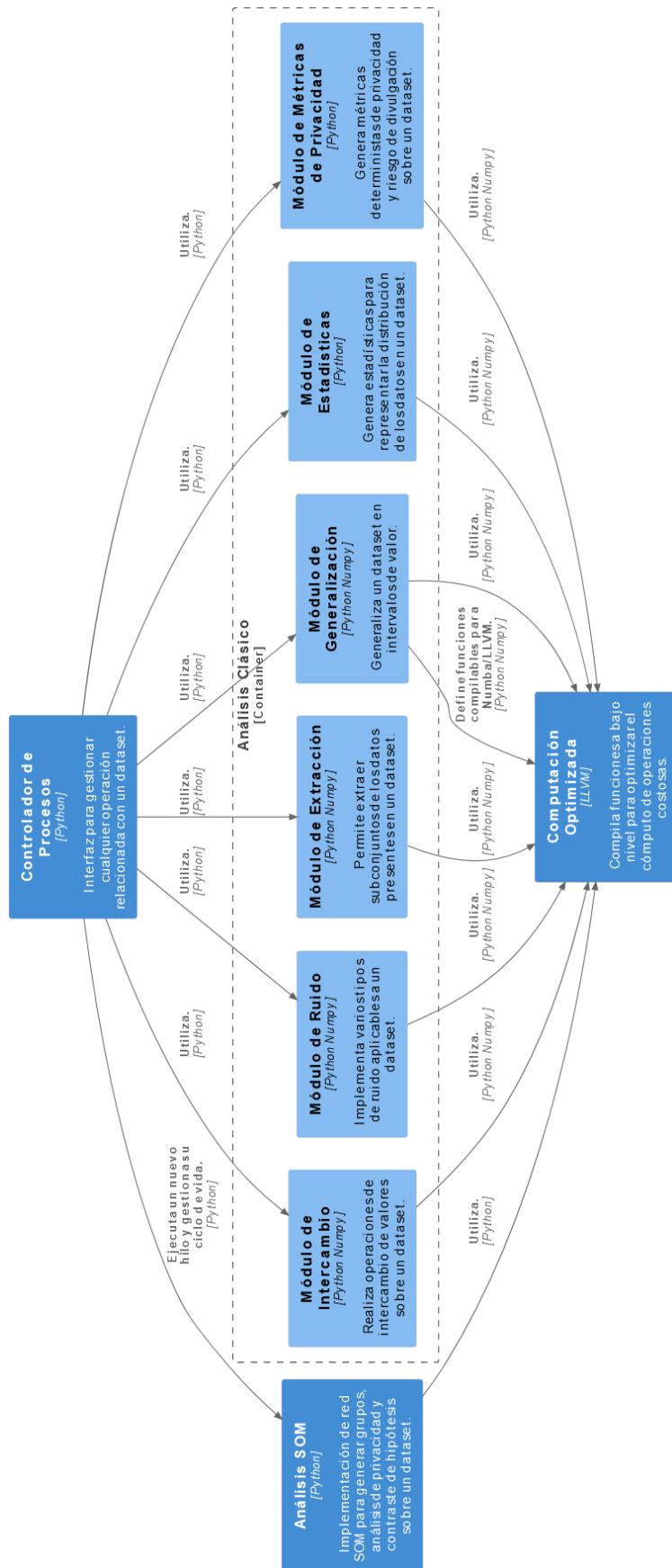


Figura B.5: Análisis Clásico

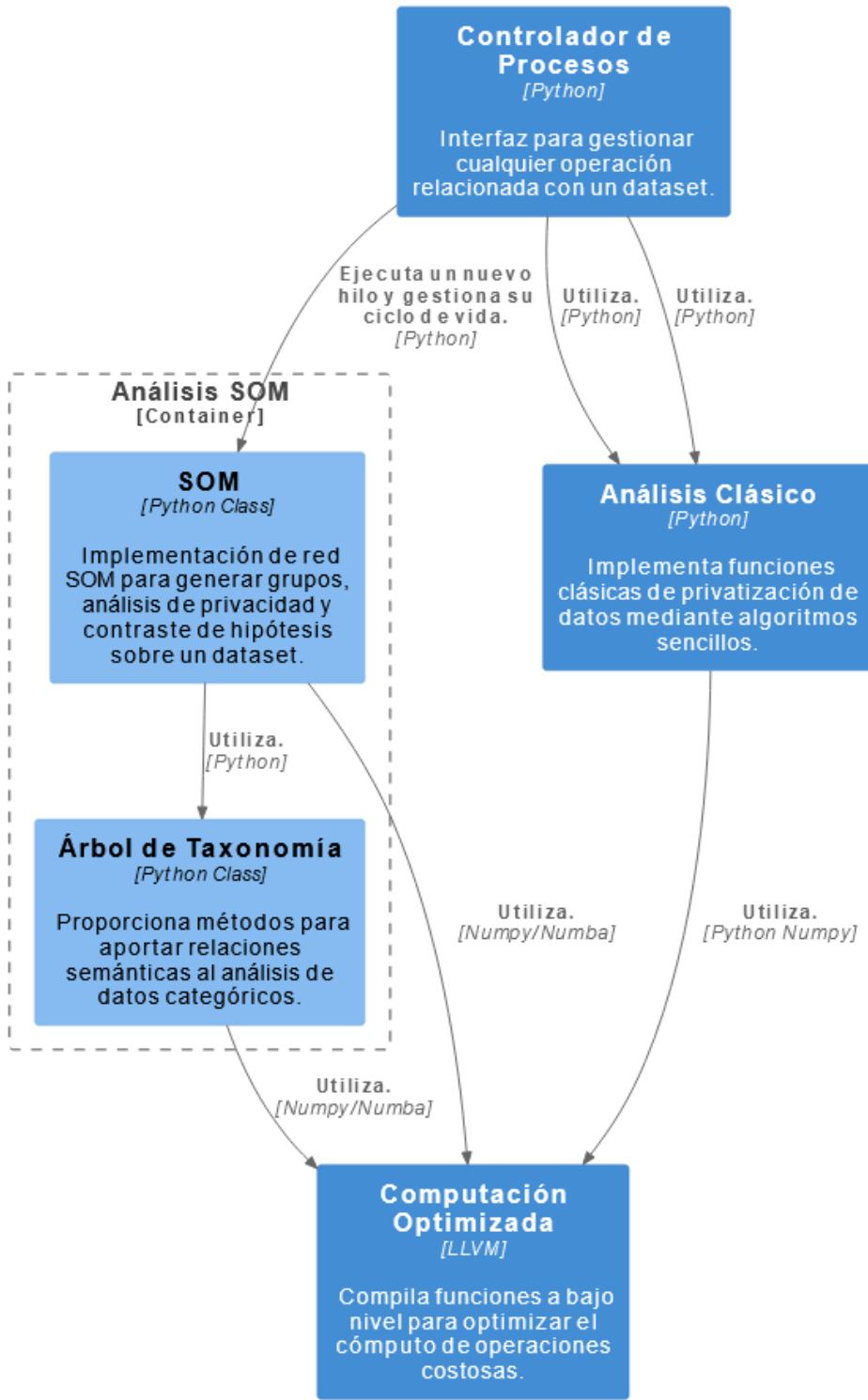


Figura B.6: Análisis SOM

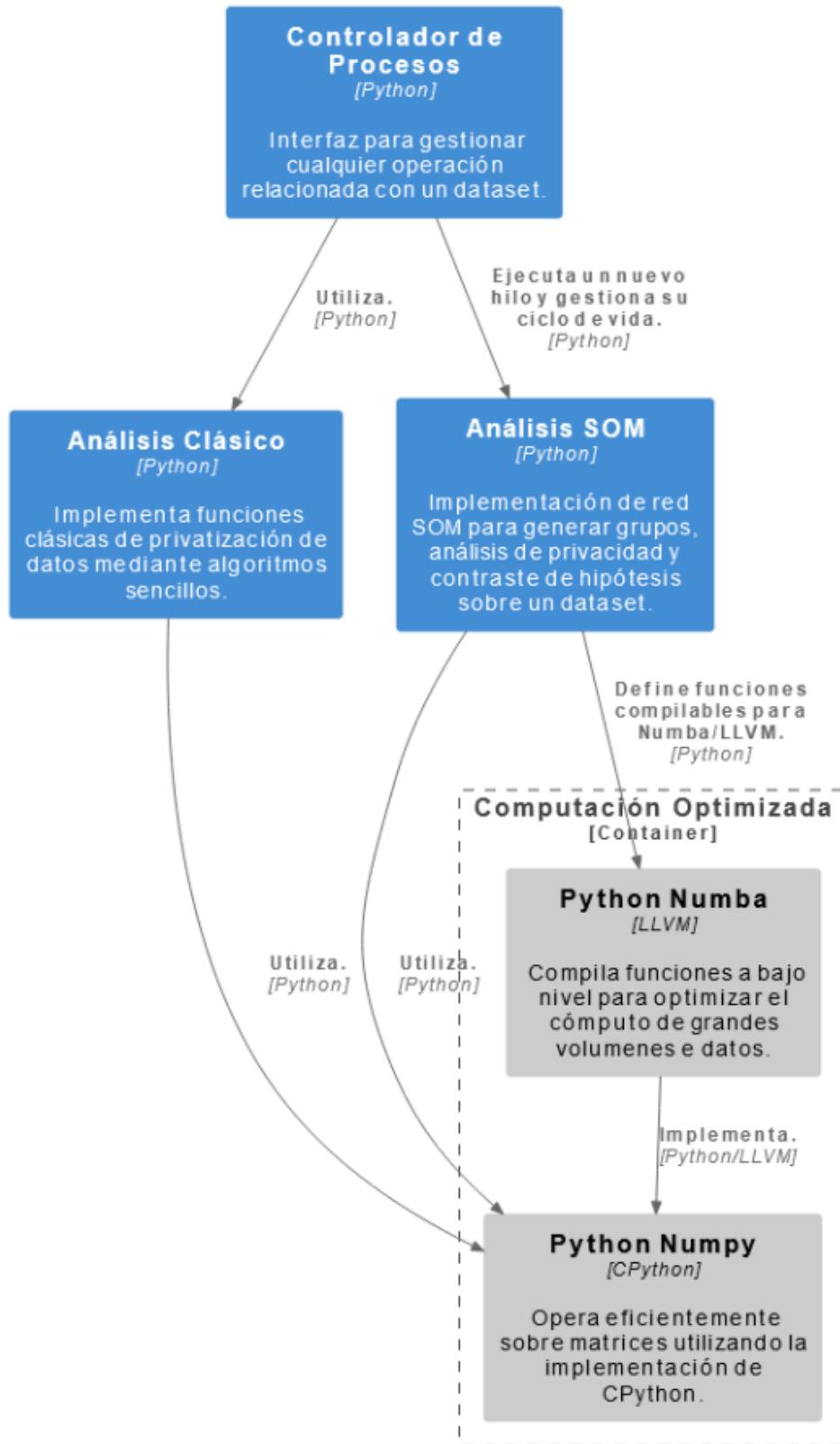


Figura B.7: Computación Optimizada

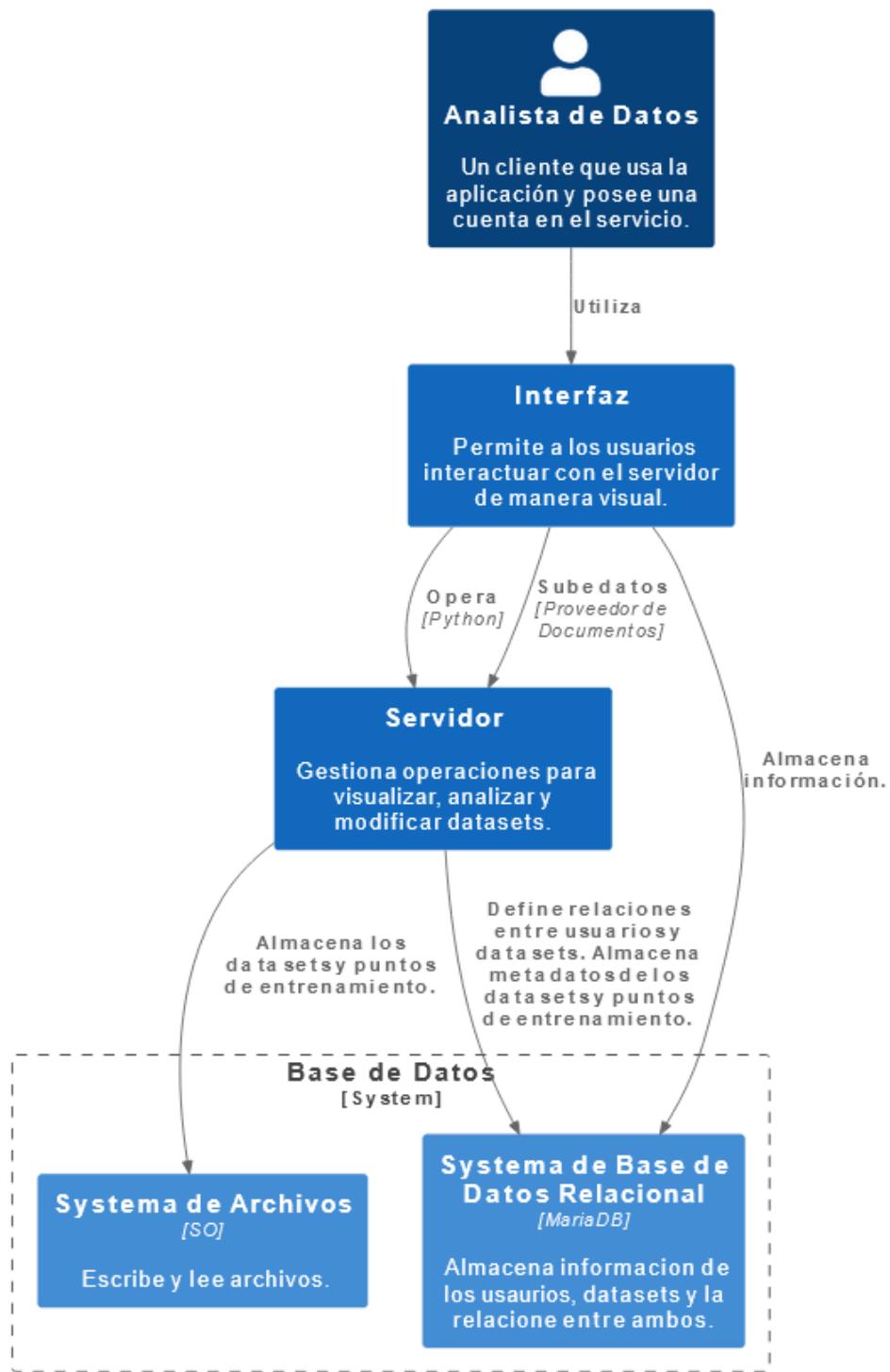


Figura B.8: Base de Datos

B.2. Pantallas del sistema

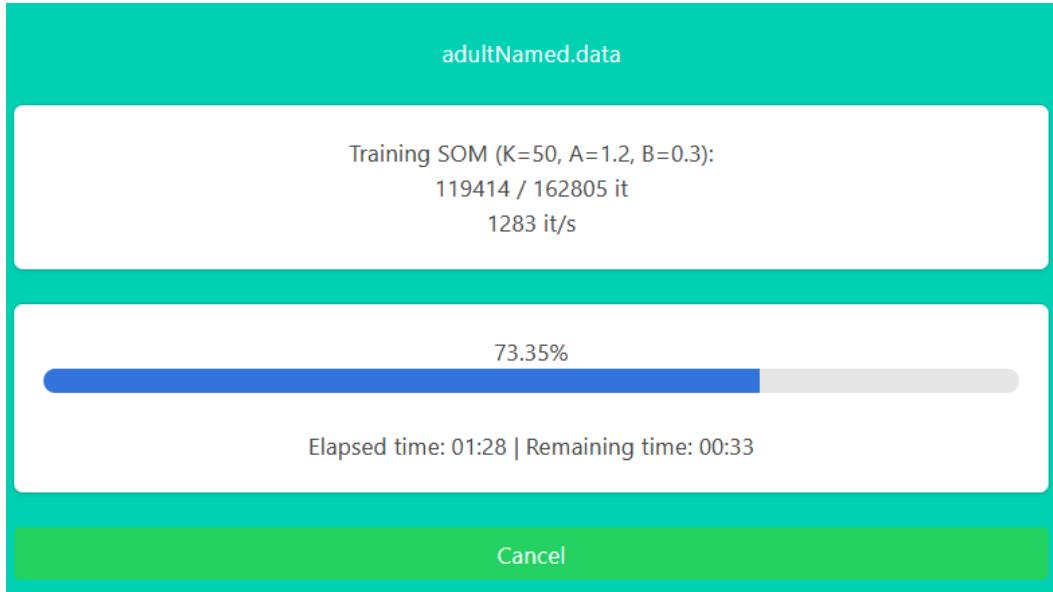


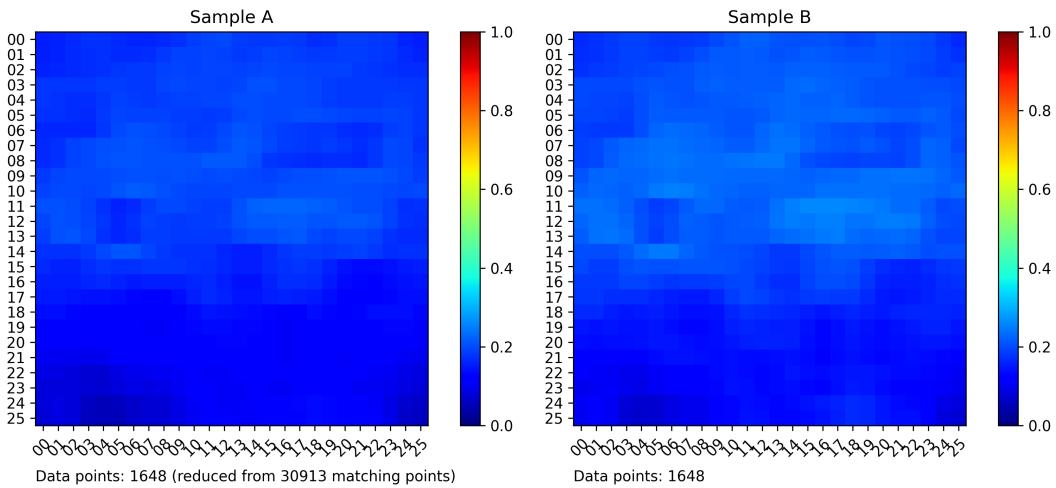
Figura B.9: Pantalla de progreso.

Parameter	Value	Description
Categorical Fields	Encoded	The system will automatically encode categorical fields by assigning each unique value a different numerical value, these values are equally distant and normalized.
Selected Operation	Hypothesis Study	Allows selecting a number of cause fields to contrast an hypothesis over an effect field. This can help to identify relation between fields and hint at the dependencies between them.

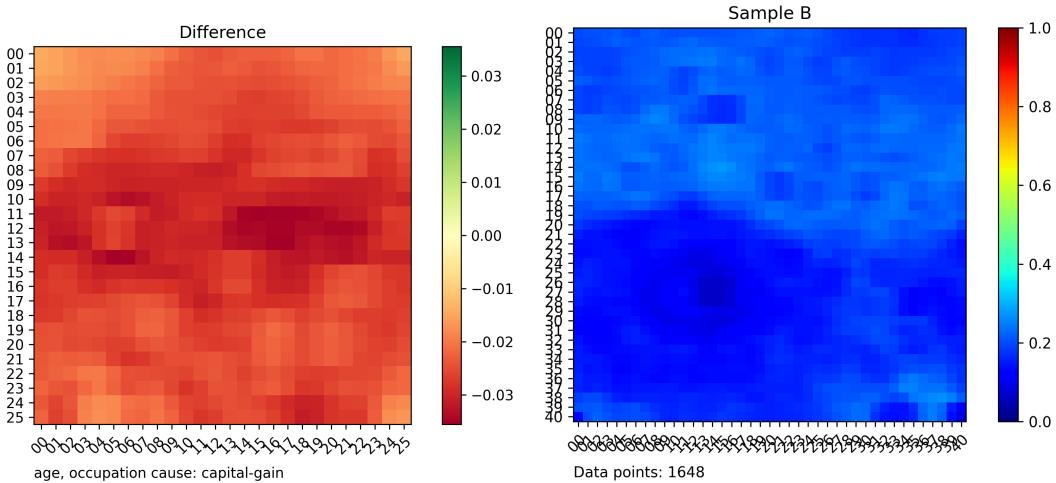
Parameter	Value	Description
Cause fields	age occupation Add Field Remove Field	List of fields that will be used to train the SOM, which will act as the cause of the hypothesis effect.
Effect Field: income_class	Select new effect field	Click this button whenever you need to select a different effect field.
Sample A first condition	A <=50K B >50K	Define which values must be present in each of the two samples.
Adjust sample size	<input checked="" type="checkbox"/>	If checked, both samples will have the same size. The resulting larger sample will be reduced to match the size of the smaller one by discarding random data points.
K:	100	Estimation of the desired elements per cluster, if "Ensure K" is selected, the algorithm will adjust the SOM result to ensure the K-anonymity.
Epochs	0	Number of iterations over the dataset, leave at 0 to let the algorithm decide.
Sigma	1	Neighbourhood size.

Hypothesis Study

Figura B.10: Selección de condiciones categóricas.



(a) Activación neuronal para la muestra A. (b) Activación neuronal para la muestra B.



(c) Diferencia de activación. (d) Mapa de aciertos.

Figura B.11: Evaluación de contraste de Hipótesis.

Imágenes generadas automáticamente por la aplicación para contrastar la hipótesis entre las muestras A *Capital-gain* |

Dataset adultNamed.data

age	workclass	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	income-class
39	State-gov	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
50	Self-emp-not-inc	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
38	Private	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K

Parameter	Value	Description
Operation	Add noise to data	Adds noise to the data.

Parameter	Value	Description
Affected Rows (0 to 100%)	<input type="text" value="100"/>	The percentage of the dataset affected by this operation.
Operation	Kim Method	Allows swapping data values between different entries. The operation will break correlations between the swapped fields and the stationary fields, but it will preserve some statistical properties of the datasets.
Noise Magnitude (0 to 100%)	<input type="text" value="10"/>	Variance modifier of the generated normal noise. It is proportional to the maximum range of each field in the dataset.

Add Noise

Figura B.12: Anonimización mediante adición de Ruido.

C. Casos de Uso

C.1. Gestión de Usuarios

Nombre del CU:	Acceder al servicio		
Creador por:	Rafael Rosso Giner	Última modificación por:	Rafael Rosso Giner
Fecha de creación:	22/07/2023	Última Revisión:	22/07/2023
Descripción:	El usuario accede al servicio autenticándose en el sistema.		
Actores:	Usuario, Sistema		
Precondiciones:	1. El servicio está operativo y listo para recibir nuevos usuarios.		
Postcondiciones:	1. El sistema otorga acceso al usuario.		
Flujo:	1. Un usuario sin autenticar accede al servicio. 2. El sistema redirecciona al usuario al portal de acceso. 3. El usuario introduce un email. 4. El usuario introduce una contraseña. 5. El usuario pulsa el botón “Entrar”. 6. El sistema le otorga acceso al usuario. 7. El sistema redirige al usuario a la página principal.		
Flujos alternativos:	6a. El sistema detecta que el email introducido no corresponde a ningún usuario. 8. El sistema informa al usuario del error detectado. 9. El caso de uso vuelve al paso 3. 6a. El sistema detecta que la contraseña introducida no se corresponde con la contraseña asociada al email introducido. 7. El sistema informa al usuario del error detectado. 8. El caso de uso vuelve al paso 3.		
Excepciones:	6a. El servicio con la base de datos se desconecta de forma abrupta. 7. El sistema informa al usuario del error. 8. El caso de uso finaliza.		
Requisitos:	Las siguientes condiciones deben cumplirse antes de comenzar con el caso de uso. 1. El servicio está operativo y listo para recibir nuevos usuarios.		

Figura C.1: Caso de Uso “Acceder al servicio”.

Nombre del CU:	Crear una nueva cuenta		
Creador por:	Rafael Rosso Giner	Última modificación por:	Rafael Rosso Giner
Fecha de creación:	22/07/2023	Última Revisión:	22/07/2023
Descripción:	El sistema crea una nueva cuenta para el usuario con los datos proporcionados.		
Actores:	Usuario, Sistema		
Precondiciones:	1. El servicio esta operativo y listo para recibir nuevos usuarios.		
Postcondiciones:	1. El sistema almacena una nueva cuenta de usuario con los datos proporcionados por el mismo.		
Flujo:	1. Un usuario sin autenticar accede al servicio. 2. El sistema redirecciona al usuario al portal de acceso. 3. El usuario selecciona la opción “Registrarse”. 4. El usuario introduce un nombre de usuario. 5. El usuario introduce un email. 6. El usuario introduce una contraseña. 7. El usuario introduce de nuevo su contraseña a modo de confirmación. 8. El usuario pulsa el botón “Registrarse” 9. El sistema crea una nueva cuenta con los datos proporcionados. 10. El sistema redirige al usuario al portal de acceso.		
Flujos alternativos:	9a. El sistema detecta que ya existe un usuario con el mismo nombre o email. 10. El sistema informa al usuario del error detectado. 11. El caso de uso vuelve al paso 4. 9a. El sistema detecta que el campo de contraseña y el de confirmación de la contraseña no coinciden. 11. El sistema informa al usuario del error detectado. 12. El caso de uso vuelve al paso 4.		
Excepciones:	9a. El servicio con la base de datos se desconecta de forma abrupta. 10. El sistema informa al usuario del error. 11. El caso de uso finaliza.		
Requisitos:	Las siguientes condiciones deben cumplirse antes de comenzar con el caso de uso. 1. El servicio esta operativo y listo para recibir nuevos usuarios.		

Figura C.2: Caso de Uso “Crear una cuenta”.

Nombre del CU:	Editar Nombre		
Creador por:	Rafael Rosso Giner	Última modificación por:	Rafael Rosso Giner
Fecha de creación:	22/07/2023	Última Revisión:	22/07/2023
Descripción:	El sistema edita el nombre asociado a la cuenta del usuario.		
Actores:	Usuario, Sistema		
Precondiciones:	1. El usuario esta conectado al servicio.		
Postcondiciones:	1. El sistema actualiza el nombre del usuario con uno nuevo introducido por el mismo.		
Flujo:	1. El usuario accede al sistema. 2. El usuario navega hasta la pantalla principal. 3. El usuario selecciona la opción “Perfil”. 4. El usuario introduce un nuevo nombre. 5. El usuario pulsa el botón “Actualizar nombre”. 6. El sistema modifica el nombre del usuario.		
Flujos alternativos:	6a. El nombre introducido coincide con el nombre de otro usuario existente. 7. El sistema informa del error al usuario. 8. El caso de uso vuelve al paso 4.		
Excepciones:	6a. El servicio con la base de datos se desconecta de forma abrupta. 7. El sistema informa al usuario del error. 8. El caso de uso finaliza.		
Requisitos:	Las siguientes condiciones deben cumplirse antes de comenzar con el caso de uso. 1. El usuario debe estar registrado y conectado al servicio con su cuenta.		

Figura C.3: Caso de Uso “Editar nombre”.

Nombre del CU:	Editar Email		
Creador por:	Rafael Rosso Giner	Última modificación por:	Rafael Rosso Giner
Fecha de creación:	22/07/2023	Última Revisión:	22/07/2023
Descripción:	El sistema edita el email asociado a la cuenta del usuario.		
Actores:	Usuario, Sistema		
Precondiciones:	1. El usuario esta conectado al servicio.		
Postcondiciones:	1. El sistema actualiza el email del usuario con uno nuevo introducido por el mismo.		
Flujo:	1. El usuario accede al sistema. 2. El usuario navega hasta la pantalla principal. 3. El usuario selecciona la opción "Perfil". 4. El usuario introduce un nuevo email. 5. El usuario pulsa el botón "Actualizar email". 6. El sistema modifica el email del usuario.		
Flujos alternativos:	6a. El email introducido coincide con el email de otro usuario existente. 7. El sistema informa del error al usuario. 8. El caso de uso vuelve al paso 4.		
Excepciones:	6a. El servicio con la base de datos se desconecta de forma abrupta. 7. El sistema informa al usuario del error. 8. El caso de uso finaliza.		
Requisitos:	Las siguientes condiciones deben cumplirse antes de comenzar con el caso de uso. 1. El usuario debe estar registrado y conectado al servicio con su cuenta.		

Figura C.4: Caso de Uso “Editar email”.

Nombre del CU:	Editar Contraseña		
Creador por:	Rafael Rosso Giner	Última modificación por:	Rafael Rosso Giner
Fecha de creación:	22/07/2023	Última Revisión:	22/07/2023
Descripción:	El sistema edita la contraseña asociada a la cuenta del usuario.		
Actores:	Usuario, Sistema		
Precondiciones:	1. El usuario esta conectado al servicio.		
Postcondiciones:	1. El sistema actualiza la contraseña del usuario con una nueva introducida por el mismo.		
Flujo:	1. El usuario accede al sistema. 2. El usuario navega hasta la pantalla principal. 3. El usuario selecciona la opción “Perfil”. 4. El usuario introduce una nueva contraseña. 5. El usuario introduce de nuevo la contraseña en el campo de confirmación. 6. El usuario pulsa el botón “Actualizar contraseña”. 7. El sistema modifica la contraseña del usuario.		
Flujos alternativos:	7a. El campo de contraseña no coincide con el campo de confirmación. 8. El sistema informa del error al usuario. 9. El caso de uso vuelve al paso 4.		
Excepciones:	7a. El servicio con la base de datos se desconecta de forma abrupta. 8. El sistema informa al usuario del error. 9. El caso de uso finaliza.		
Requisitos:	Las siguientes condiciones deben cumplirse antes de comenzar con el caso de uso. 1. El usuario debe estar registrado y conectado al servicio con su cuenta.		

Figura C.5: Caso de Uso “Editar contraseña”.

Nombre del CU:	Borrar Cuenta		
Creador por:	Rafael Rosso Giner	Última modificación por:	Rafael Rosso Giner
Fecha de creación:	22/07/2023	Última Revisión:	22/07/2023
Descripción:	El sistema elimina la cuenta del usuario y todos los datos asociados.		
Actores:	Usuario, Sistema		
Precondiciones:	1. El usuario está conectado al servicio.		
Postcondiciones:	1. La cuenta asociada al usuario es eliminada del sistema. Todos los datasets y datos asociados al usuario son también eliminados.		
Flujo:	1. El usuario accede al sistema. 2. El usuario navega hasta la pantalla principal. 3. El usuario selecciona la opción "Perfil". 4. El usuario marca la casilla de confirmación para eliminar su cuenta. 5. El usuario pulsa el botón "Eliminar cuenta". 6. El sistema elimina la cuenta del usuario junto a todos los datasets y datos asociados a ella. 7. El sistema desconecta al usuario y muestra la página de acceso.		
Flujos alternativos:	5a. El usuario no marca la casilla de confirmación para eliminar su cuenta. 6. El usuario pulsa el botón "Eliminar cuenta". 7. El sistema informa al usuario que debe marcar la casilla. 8. El caso de uso vuelve al paso 4.		
Excepciones:	6a. El servicio con la base de datos se desconecta de forma abrupta. 7. El sistema informa al usuario del error. 8. El caso de uso finaliza.		
Requisitos:	Las siguientes condiciones deben cumplirse antes de comenzar con el caso de uso. 1. El usuario debe estar registrado y conectado al servicio con su cuenta.		

Figura C.6: Caso de Uso “Eliminar cuenta”.

C.2. Datasets

Nombre del CU:	Subir Dataset
Creador por:	Rafael Rosso Giner
Fecha de creación:	22/07/2023
Descripción:	El usuario sube un nuevo conjunto de datos al sistema que podrá ser usado en el futuro por el propietario y, opcionalmente, por otros usuarios de la plataforma.
Actores:	Usuario, Sistema
Precondiciones:	<ul style="list-style-type: none"> 1. El usuario está conectado al servicio. 2. El usuario posee un dataset con formato CSV. 3. El Sistema es accesible, preparado para recibir los datos y con espacio suficiente para almacenarlos
Postcondiciones:	<ul style="list-style-type: none"> 1. El sistema guarda el documento con el dataset y genera una nueva entrada en la base de datos con los metadatos correspondientes.
Flujo:	<ul style="list-style-type: none"> 1. El usuario accede al sistema. 2. El usuario navega hasta su perfil. 3. El usuario selecciona la opción “subir dataset”. 4. El usuario selecciona el botón “seleccionar archivo”. 5. El usuario selecciona el documento a subir utilizando la interfaz de su SO. 6. El usuario selecciona la opción “subir archivo”. 7. El sistema indica al usuario que el documento se ha subido correctamente.
Flujos alternativos:	<ul style="list-style-type: none"> 5a. En el paso 5, si el usuario no ha seleccionado ningún archivo. 6. El usuario selecciona la opción “subir archivo”. 7. El Sistema le informa de que debe seleccionar un archivo. 8. El caso de uso vuelve al paso 4 del flujo. 5b. En el paso 5, si el usuario selecciona un archivo inválido. 6. El usuario selecciona la opción “subir archivo”. 7. El Sistema informa al usuario de que el dataset es inválido y lista los errores que deben solventarse para que pueda subirse. 8. El caso de uso vuelve al paso 4 del flujo.
Excepciones:	<ul style="list-style-type: none"> 6a. En el paso 6, la conexión es detenida de forma abrupta o el usuario cierra la aplicación. 7. El sistema no almacena el dataset ni ningún dato relacionado. 8. El caso de uso finaliza.
Requisitos:	<p>Las siguientes condiciones deben cumplirse antes de comenzar con el caso de uso.</p> <ul style="list-style-type: none"> 1. El usuario debe estar registrado y conectado al servicio con su cuenta. 2. El Sistema debe poseer suficiente espacio para almacenar los datos.

Figura C.7: Caso de Uso “Subir un Dataset”.

Nombre del CU:	Eliminar dataset				
Creador por:	Rafael Rosso Giner	Última modificación por:	Rafael Rosso Giner		
Fecha de creación:	22/07/2023	Última Revisión: 22/07/2023			
Descripción:	El usuario elimina un dataset del sistema.				
Actores:	Usuario, Sistema				
Precondiciones:	1. El usuario está conectado al servicio. 2. El usuario es propietario de uno o más datasets del sistema.				
Postcondiciones:	1. El sistema elimina el documento que representa un dataset, los metadatos asociados y las entidades relacionadas, árboles de taxonomía y puntos de guardado.				
Flujo:	1. El usuario accede al sistema. 2. El usuario navega hasta la pantalla principal. 3. El usuario selecciona la opción “Perfil”. 4. El usuario pulsa el botón “Eliminar dataset” de uno de los datasets disponibles. 5. El sistema elimina el documento que representa el atributo seleccionado.				
Flujos alternativos:					
Excepciones:					
Requisitos:	Las siguientes condiciones deben cumplirse antes de comenzar con el caso de uso. 1. El usuario debe estar registrado y conectado al servicio con su cuenta. 2. El usuario debe ser propietario de uno o más datasets en el sistema.				

Figura C.8: Caso de Uso “Eliminar un Dataset”.

Nombre del CU:	Descargar dataset		
Creador por:	Rafael Rosso Giner	Última modificación por:	Rafael Rosso Giner
Fecha de creación:	22/07/2023	Última Revisión:	22/07/2023
Descripción:	El usuario descarga un dataset del sistema.		
Actores:	Usuario, Sistema		
Precondiciones:	1. El usuario está conectado al servicio. 2. El usuario debe tener acceso a uno o más datasets del sistema.		
Postcondiciones:	1. El usuario obtiene un documento con los datos del dataset seleccionado.		
Flujo:	1. El usuario accede al sistema. 2. El usuario navega hasta la pantalla principal. 3. El usuario selecciona la opción “Perfil”. 4. El usuario pulsa el botón “Descargar dataset” de uno de los datasets disponibles. 5. El sistema envía al usuario un documento con los datos almacenados sobre el dataset seleccionado.		
Flujos alternativos:			
Excepciones:			
Requisitos:	Las siguientes condiciones deben cumplirse antes de comenzar con el caso de uso. 1. El usuario debe estar registrado y conectado al servicio con su cuenta. 2. El usuario debe tener acceso a uno o más datasets del sistema.		

Figura C.9: Caso de Uso “Descargar un Dataset”.

C.3. Árboles de Taxonomía

Nombre del CU:	Subir árbol de taxonomía		
Creador por:	Rafael Rosso Giner	Última modificación por:	Rafael Rosso Giner
Fecha de creación:	22/07/2023	Última Revisión:	22/07/2023
Descripción:	El usuario sube un nuevo árbol de taxonomía al sistema.		
Actores:	Usuario, Sistema		
Precondiciones:	1. El usuario está conectado al servicio. 2. El usuario tiene acceso a uno o más datasets del sistema. 3. El Sistema es accesible, preparado para recibir los datos y con espacio suficiente para almacenarlo		
Postcondiciones:	1. El sistema almacena un nuevo documento para representar un atributo categórico de un dataset concreto.		
Flujo:	1. El usuario accede al sistema. 2. El usuario navega hasta la pantalla principal. 3. El usuario selecciona la opción “Perfil”. 4. El usuario pulsa el botón “Gestionar Árboles” de uno de los datasets disponibles. 5. El usuario pulsa el botón “Subir Árbol” de uno de los atributos disponibles. 6. El usuario selecciona el documento a subir utilizando la interfaz de su SO. 7. El sistema almacena el nuevo documento.		
Flujos alternativos:	6a. En el paso 6, si el usuario no ha seleccionado ningún archivo. 6. El usuario selecciona la opción “subir archivo”. 7. El Sistema le informa de que debe seleccionar un archivo. 8. El caso de uso vuelve al paso 5 del flujo. 6b. En el paso 6, si el usuario selecciona un archivo inválido. 7. El usuario selecciona la opción “subir archivo”. 8. El Sistema informa al usuario de que el documento es inválido y lista los errores que deben solventarse para que pueda subirse. 9. El caso de uso vuelve al paso 5 del flujo.		
Excepciones:	6a. En el paso 6, la conexión es detenida de forma abrupta o el usuario cierra la aplicación. 7. El sistema no almacena el documento ni ningún dato relacionado. 8. El caso de uso finaliza.		
Requisitos:	Las siguientes condiciones deben cumplirse antes de comenzar con el caso de uso. 1. El usuario debe estar registrado y conectado al servicio con su cuenta. 2. El usuario debe tener acceso a uno o más datasets en el sistema. 3. El usuario debe poseer un documento de texto válido para representar un árbol de taxonomía.		

Figura C.10: Caso de Uso “Subir un Árbol de Taxonomía”.

Nombre del CU:	Descargar árbol de taxonomía		
Creador por:	Rafael Rosso Giner	Última modificación por:	Rafael Rosso Giner
Fecha de creación:	22/07/2023	Última Revisión:	22/07/2023
Descripción:	El usuario descarga un árbol de taxonomía del sistema.		
Actores:	Usuario, Sistema		
Precondiciones:	1. El usuario está conectado al servicio. 2. El usuario tiene acceso a uno o más datasets del sistema y debe existir un documento en el sistema que represente el árbol de taxonomía a editar.		
Postcondiciones:	1. El sistema descarga el documento que representa un atributo categórico de un dataset concreto.		
Flujo:	1. El usuario accede al sistema. 2. El usuario navega hasta la pantalla principal. 3. El usuario selecciona la opción “Perfil”. 4. El usuario pulsa el botón “Gestionar Árboles” de uno de los datasets disponibles. 5. El usuario pulsa el botón “Descargar árbol” de uno de los atributos disponibles. 6. El sistema envía al usuario el documento que representa el atributo seleccionado.		
Flujos alternativos:			
Excepciones:			
Requisitos:	Las siguientes condiciones deben cumplirse antes de comenzar con el caso de uso. <ol style="list-style-type: none"> 1. El usuario debe estar registrado y conectado al servicio con su cuenta. 2. El usuario debe tener acceso a uno o más datasets en el sistema. 3. El dataset sobre el que se trabaja debe tener un árbol de taxonomía almacenado en el sistema para el atributo seleccionado. 		

Figura C.11: Caso de Uso “Descargar un Árbol de Taxonomía”.

Nombre del CU:	Editar árbol de taxonomía		
Creador por:	Rafael Rosso Giner	Última modificación por:	Rafael Rosso Giner
Fecha de creación:	22/07/2023	Última Revisión:	22/07/2023
Descripción:	El usuario edita un árbol de taxonomía del sistema.		
Actores:	Usuario, Sistema		
Precondiciones:	1. El usuario está conectado al servicio. 2. El usuario tiene acceso a uno o más datasets del sistema y debe existir un documento en el sistema que represente el árbol de taxonomía a editar.		
Postcondiciones:	1. El sistema modifica el documento que representa un atributo categórico de un dataset concreto.		
Flujo:	1. El usuario accede al sistema. 2. El usuario navega hasta la pantalla principal. 3. El usuario selecciona la opción “Perfil”. 4. El usuario pulsa el botón “Gestionar Árboles” de uno de los datasets disponibles. 5. El usuario pulsa el botón “Editar árbol” de uno de los atributos disponibles. 6. El usuario redacta la nueva sintaxis del árbol de taxonomía. 7. El usuario pulsa el botón “Guardar cambios”. 8. El sistema modifica el documento correspondiente con los nuevos datos.		
Flujos alternativos:	7a. En el paso 7, el usuario escribe una sintaxis inválida. 8. El sistema informa al usuario del error. 9. El caso de uso vuelve al paso 6 del flujo.		
Excepciones:			
Requisitos:	Las siguientes condiciones deben cumplirse antes de comenzar con el caso de uso. 1. El usuario debe estar registrado y conectado al servicio con su cuenta. 2. El usuario debe tener acceso a uno o más datasets en el sistema. 3. El dataset sobre el que se trabaja debe tener un árbol de taxonomía almacenado en el sistema para el atributo seleccionado.		

Figura C.12: Caso de Uso “Editar un Árbol de Taxonomía”.

Nombre del CU:	Eliminar árbol de taxonomía		
Creador por:	Rafael Rosso Giner	Última modificación por:	Rafael Rosso Giner
Fecha de creación:	22/07/2023	Última Revisión:	22/07/2023
Descripción:	El usuario elimina un árbol de taxonomía del sistema.		
Actores:	Usuario, Sistema		
Precondiciones:	1. El usuario está conectado al servicio. 2. El usuario tiene acceso a uno o más datasets del sistema y debe existir un documento en el sistema que represente el árbol de taxonomía a editar.		
Postcondiciones:	1. El sistema elimina el documento que representa un atributo categórico de un dataset concreto.		
Flujo:	1. El usuario accede al sistema. 2. El usuario navega hasta la pantalla principal. 3. El usuario selecciona la opción "Perfil". 4. El usuario pulsa el botón "Gestionar Árboles" de uno de los datasets disponibles. 5. El usuario pulsa el botón "Eliminar árbol" de uno de los atributos disponibles. 6. El sistema elimina el documento que representa el atributo seleccionado.		
Flujos alternativos:			
Excepciones:			
Requisitos:	Las siguientes condiciones deben cumplirse antes de comenzar con el caso de uso. <ol style="list-style-type: none"> 1. El usuario debe estar registrado y conectado al servicio con su cuenta. 2. El usuario debe tener acceso a uno o más datasets en el sistema. 3. El dataset sobre el que se trabaja debe tener un árbol de taxonomía almacenado en el sistema para el atributo seleccionado. 		

Figura C.13: Caso de Uso “Eliminar un Árbol de Taxonomía” .

C.4. Operaciones

Nombre del CU:	Análisis SOM		
Creador por:	Rafael Rosso Giner	Última modificación por:	Rafael Rosso Giner
Fecha de creación:	22/07/2023	Última Revisión:	22/07/2023
Descripción:	El usuario realiza un análisis SOM sobre un conjunto de datos para buscar agrupaciones.		
Actores:	Usuario, Sistema		
Precondiciones:	1. El usuario está conectado al servicio. 2. El usuario tiene acceso a uno o más datasets del sistema. 3. El Sistema es accesible, preparado para recibir los datos y con espacio suficiente para almacenarlo		
Postcondiciones:	1. El usuario accede a una pantalla con los análisis del dataset.		
Flujo:	1. El usuario accede al sistema. 2. El usuario navega hasta la pantalla principal. 3. El usuario selecciona la opción “Análisis SOM”. 4. El usuario selecciona la operación “Clustering”. 5. El usuario indica los parámetros que desea utilizar para el entrenamiento. 6. El usuario presiona el botón “Train SOM”. 7. El sistema muestra una pantalla de carga al usuario. 8. El usuario espera hasta que el sistema termine de procesar la operación. 9. El sistema muestra los resultados.		
Flujos alternativos:	8a. El usuario decide pulsar la opción de abortar antes de que termine de ejecutarse la operación. 8. El Sistema cancela la operación actual. 9. El caso de uso vuelve al paso 5. 7b. Hay algún error con la combinación de parámetros o dataset seleccionado para dicha combinación. 8. El Sistema informa al usuario del error y como debe solventarlo. 9. El caso de uso vuelve al paso 5.		
Excepciones:	8a. La operación se detiene de forma abrupta. 8. El sistema informa al usuario de que ha ocurrido un error y la operación no ha podido completarse. 9. El caso de uso finaliza.		
Requisitos:	Las siguientes condiciones deben cumplirse antes de comenzar con el caso de uso. 1. El usuario debe estar registrado y conectado al servicio con su cuenta. 2. El usuario debe tener acceso a uno o más datasets en el sistema.		

Figura C.14: Caso de Uso “Análisis SOM”.

Nombre del CU:	Privatización de Datos		
Creador por:	Rafael Rosso Giner	Última modificación por:	Rafael Rosso Giner
Fecha de creación:	22/07/2023	Última Revisión:	22/07/2023
Descripción:	El usuario realiza una operación de privatización sobre uno de los datasets disponibles en el sistema.		
Actores:	Usuario, Sistema		
Precondiciones:	1. El usuario está conectado al servicio. 2. El usuario tiene acceso a uno o más datasets del sistema. 3. El Sistema es accesible, preparado para generar los datos y con espacio suficiente para almacenar el nuevo dataset generado.		
Postcondiciones:	1. El usuario obtiene un dataset transformado en función de la operación de privatización seleccionada.		
Flujo:	1. El usuario accede al sistema. 2. El usuario navega hasta la pantalla principal. 3. El usuario selecciona la opción “Privatización”. 4. El usuario selecciona uno de los datasets disponibles. 5. El usuario selecciona la operación de privatización deseada. 6. El usuario parametriza la operación seleccionada. 7. El usuario presiona el botón “Privatizar”. 8. El sistema genera un nuevo dataset privatizado. 9. El usuario introduce el nombre del nuevo dataset y pulsa el botón “Guardar como nuevo dataset”. 10. El sistema almacena el nuevo dataset transformado, con el nombre indicado por el usuario.		
Flujos alternativos:	8a. El sistema detecta un error con los datos o los parámetros seleccionados. 9. El sistema indica al usuario donde se encuentra el error. 10. El caso de uso vuelve al paso 6. 9a. El usuario pulsa el botón “Reemplazar original”. 10. El sistema reemplaza los datos, y metadatos, del dataset original con los del nuevo dataset generado. 9b. El usuario pulsa el botón “Descargar”. 10. El Sistema genera un documento de texto y lo envía al navegador web del cliente para que pueda descargarlo.		
Excepciones:	8a. La operación se detiene de forma abrupta. 8. El sistema informa al usuario de que ha ocurrido un error y la operación no ha podido completarse. 9. El caso de uso finaliza.		
Requisitos:	Las siguientes condiciones deben cumplirse antes de comenzar con el caso de uso. 1. El usuario debe estar registrado y conectado al servicio con su cuenta. 2. El usuario debe tener acceso a uno o más datasets en el sistema.		

Figura C.15: Caso de Uso “Privatización de Datos”.

Nombre del CU:	Comparación de Datos		
Creador por:	Rafael Rosso Giner	Última modificación por:	Rafael Rosso Giner
Fecha de creación:	22/07/2023	Última Revisión:	22/07/2023
Descripción:	El usuario realiza una operación de comparación entre dos de los datasets disponibles en el sistema.		
Actores:	Usuario, Sistema		
Precondiciones:	1. El usuario está conectado al servicio. 2. El usuario tiene acceso a dos o más datasets del sistema.		
Postcondiciones:	1. El sistema muestra al usuario un estudio comparativo entre los dos datasets seleccionados.		
Flujo:	1. El usuario accede al sistema. 2. El usuario navega hasta la pantalla principal. 3. El usuario selecciona la opción “Comparación”. 4. El usuario selecciona dos de los datasets disponibles. 5. El sistema muestra el estudio comparativo entre los datasets seleccionados. 6. Finaliza el caso de uso.		
Flujos alternativos:	4a. Si se detecta un error en los datos almacenados. 5. El sistema informa al usuario del error detectado. 6. El caso de uso vuelve al paso 4. 6a. El usuario pulsa el botón “Descargar resultados”. 7. El sistema envía al usuario un documento con los resultados del estudio. 8. El caso de uso finaliza.		
Excepciones:	4a. Si se detiene el servicio de forma abrupta. 11. El sistema informa al usuario de que ha ocurrido un error y la operación no ha podido completarse. 12. El caso de uso finaliza.		
Requisitos:	Las siguientes condiciones deben cumplirse antes de comenzar con el caso de uso. 1. El usuario debe estar registrado y conectado al servicio con su cuenta. 2. El usuario debe tener acceso a dos o más datasets en el sistema.		

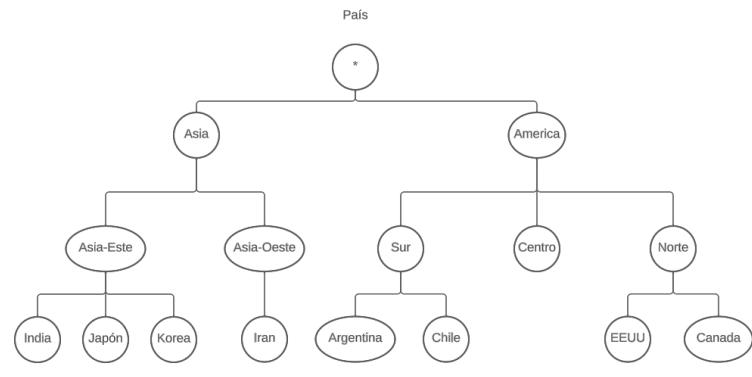
Figura C.16: Caso de Uso “Comparación de Datos”.

D. Diagramas adicionales

```

1  India;Asia-Este;Asia;*
2  Japón;Asia-Este;Asia;*
3  Korea;Asia-Este;Asia;*
4  Iran;Asia-Oeste;Asia;*
5  Argentina;Sur;America;*
6  Chile;Sur;America;*
7  Centro;America;*
8  EEUU;Norte;America;*
9  Canada;Norte;America;*

```



(a) Sintaxis del árbol

(b) Representación del árbol

En este árbol, India y Corea están a una distancia de $\frac{1}{3} = 0,33$, India y Canadá se encuentran a $\frac{3}{3} = 1$. India y Asia-Oeste estarían a una distancia de $\frac{2}{3} = 0,66$

Figura D.1: Ejemplo de árbol de taxonomía

Tabla D.1

Paquetes de Python

Paquetes de Python		
Nombre	Compatibilidad	Version
adjustText	exacta	0.7.3
bottle	exacta	0.12.25
certifi	exacta	2022.12.7
charset-normalizer	exacta	3.0.1
click	exacta	8.1.3
colorama	exacta	0.4.6
colourmap	exacta	1.1.10
cycler	exacta	0.11.0
Flask	compatible	2.2.5
fonttools	exacta	4.38.0
idna	exacta	3.4
importlib-metadata	exacta	6.3.0
itsdangerous	exacta	2.1.2
Jinja2	exacta	3.1.2
joblib	exacta	1.2.0
kiwisolver	exacta	1.4.4
llvmlite	exacta	0.39.1
MarkupSafe	exacta	2.1.2
matplotlib	exacta	3.5.3
numba	exacta	0.56.4
numpy	exacta	1.21.6
packaging	exacta	23.0
pandas	exacta	1.3.5
pca	exacta	1.8.6
Pillow	exacta	9.4.0
pyparsing	exacta	3.0.9
python-dateutil	exacta	2.8.2
pytz	exacta	2022.7.1
requests	exacta	2.28.2
scatterd	exacta	1.2.5
scikit-learn	exacta	1.0.2
scipy	exacta	1.7.3
seaborn	exacta	0.12.2
six	exacta	1.16.0
threadpoolctl	exacta	3.1.0
tqdm	exacta	4.64.1
typing-extensions	exacta	4.4.0
urllib3	exacta	1.26.14
Werkzeug	exacta	2.2.3
wget	exacta	3.2
zipp	exacta	3.15.0
SQLAlchemy	compatible	2.0.18