FACULTAD DE CIENCIAS Y TECNOLOGÍAS COMPUTACIONALES

Sistema web de apoyo a la enseñanza de los algoritmos de clasificación no supervisada del Reconocimiento Lógico Combinatorio de Patrones

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

Autor: Susana Cremé Zerquera
Tutores: Dr. C. Natalia Martínez Sánchez
Dr. C. Yunia Reyes González

La Habana, noviembre de 2022

Año 64 de la Revolución

DECLARACIÓN DE AUTORÍA

La autora del trabajo de diploma con título “Sistema web para la clasificación no supervisada utilizando el Reconocimiento Lógico Combinatorio de Patrones”, concede a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la investigación, con carácter exclusivo. De forma similar se declara como único autor de su contenido. Para que así conste firma la presente a los 2 días del mes de noviembre del año 2022.

Susana Cremé Zerquera

_____

Firma del Autor

<nombre del tutor>

_____

Firma del Tutor

“La actividad ha de ser el símbolo de la juventud”.

José Martí

DATOS DE CONTACTO

Dr.C. Natalia Martínez Sánchez (natalia@uci.cu)

Se desempeña como Vicerrectora de Formación. Licenciada en Cibernética-Matemática. Obtuvo el grado de Máster en Computación Aplicada, Profesora Titular, 36 años en educación superior. Trabaja en la Línea de Reconocimiento de Patrones e Inteligencia Artificial.

Dr.C. Yunia Reyes González (yrglez@uci.cu)

Se desempeña como Vicedecana de Investigación y Postgrado de la Facultad 2. Ingeniera en Ciencias Informáticas. Obtuvo el grado de Máster en Informática Aplicada. Trabaja en la Línea de Reconocimiento de Patrones e Inteligencia Artificial.

AGRADECIMIENTOS

Agradezco a la Universidad de las Ciencias Informáticas por la inmensa oportunidad de estudiar y graduarme aquí. A mi tutora Natalia por confiar en mí y darme su apoyo en la realización de este trabajo de diploma. A Cristhian por su amor y su apoyo incondicional cuando más lo necesité. A todos mis amigos, en especial a Michel, Reynier, Javier, Brian, Rixon, Ernesto, Dayana y Arianna por ayudarme profesional y personalmente durante toda la carrera y en particular en este último período de la tesis. A mi familia, en especial a mis padres y a mi hermana mayor por quererme, apoyarme y ayudarme siempre en todos los aspectos de mi vida, en mis años de estudio y a cumplir mis sueños.

DEDICATORIA

Este trabajo y todos sus resultados se los dedico a mis padres y mis hermanas; las personas más importantes en mi vida. Gracias por el amor, la entrega, la paciencia y la confianza que siempre dedicaron. Nunca podré llegar a agradecerles lo suficiente por todo lo que hicieron por mí. Gracias por creer en mí. Los amo.

RESUMEN

En la Universidad de las Ciencias Informáticas existe una Línea de investigación de Reconocimiento de Patrones e Inteligencia Artificial donde se desarrollan módulos que aportan un valor agregado a las aplicaciones informáticas desarrolladas en los centros de desarrollo de software de la institución, por lo que interesa la formación profesional de los estudiantes en esta rama a través del currículo optativo de la disciplina de Inteligencia Computacional. La asignatura Reconocimiento de Patrones se fundamenta en el Reconocimiento Lógico Combinatorio de Patrones y dentro de esta juega un papel esencial la clasificación no supervisada, por lo que el objetivo del presente trabajo de diploma es desarrollar un sistema web de apoyo a la docencia. El sistema web, el cual que lleva como nombre “USGest”, será capaz de ofrecer a los estudiantes una visualización del funcionamiento del algoritmo

Class mediante su interfaz sencilla y de fácil utilización. Se empleó Python como lenguaje de programación, Django como marco de trabajo y se utilizó el editor de código Visual Estudio Code. Se obtuvo como resultado un sistema web que permite la clasificación no supervisada de objetos utilizando el Reconocimiento Lógico Combinatorio de Patrones como apoyo para la docencia. Las pruebas realizadas determinaron la correcta funcionalidad y la aceptación del sistema web por parte del cliente, las cuales fueron de gran valor para la calidad de la propuesta solución.

PALABRAS CLAVE

algoritmo, clasificación no supervisada, patrones, sistema web

ABSTRACT

At the Informatics Sciences University (UCI) there is a research line of Pattern Recognition and Artificial Intelligence where modules are developed that provide an added value to the computer applications developed in the software development centers of the institution, so the professional training of students in this branch is of interest through the optional curriculum of the Computational Intelligence line. The Pattern Recognition subject is based on the Combinatorial Logical Pattern Recognition and within it plays an essential role the unsupervised classification, so the objective of this diploma work is to develop a web system to support teaching. The web system, named "USGest", will be able to provide students with a visualization of the Class algorithm operation through its simple and user-friendly interface. Python was used as the programming language, Django as the framework and the Visual Studio Code editor was used. The result was a web system that allows the unsupervised classification of objects using Combinatorial Logical Pattern Recognition as a support

for teaching. The tests performed determined the correct functionality and the acceptance of the web system by the client, which were of great value for the quality of the proposed solution.

KEYWORDS
algorithm, unsupervised sorting, patterns, web system

TABLA DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

OPINIÓN DEL(OS) TUTOR(ES)

<Contenido de la opinión de los tutores>

AVAL DEL CLIENTE

Universidad de las Ciencias Informáticas

La Habana, 21 de noviembre de 2022
"Año 63 de la Revolución"

A: Dr. C. Jorge Gulín González
Presidente del Tribunal.

En la carrera de Ingeniería en Ciencias Informáticas como parte del currículo optativo de la Disciplina de Inteligencia Computacional se imparte la asignatura Reconocimiento de Patrones.

Esta asignatura está dirigida al reconocimiento lógico combinatorio de patrones, que es la zona del conocimiento que se ocupa del desarrollo de teorías, métodos, técnicas y dispositivos computacionales para la realización de procesos ingenieriles, computacionales y/o matemáticos, relacionados con objetos físicos o abstractos, que tienen el propósito de extraer información que permita establecer propiedades y/o vínculos de o entre conjuntos de dichos objetos sobre la base de los cuales se realiza una tarea de identificación o clasificación. Por problemas de reconocimiento de patrones en esta asignatura se entienden todos aquellos relacionados con la clasificación de objetos y fenómenos y con la determinación de los factores que inciden en los mismos.

La asignatura se imparte a partir del tercer año de la carrera indistintamente y en los últimos cursos académicos se comenzó a impartir en la carrera de Ingeniería en Bioinformática.

En los análisis realizados se ha podido contactar que en esta etapa de la carrera aún persisten insuficiencias en la capacidad de razonamiento y abstracción, mediante el análisis conceptual y algorítmico en los estudiantes, lo que limita la comprensión de los algoritmos que se estudian en la asignatura. Por tal razón es de interés de las tutoras que la estudiante Susana Cremé Zerquera desarrollara un sistema web de apoyo a la enseñanza – aprendizaje comenzando por los algoritmos de clasificación no supervisada del reconocimiento lógico combinatorio de patrones, como complemento a la impartición de la asignatura. Esta necesidad, se evidenció aún más en la etapa de pandemia que fue necesario impartir la asignatura a distancia.

La estudiante, fue capaz de implementar la herramienta computacional con las funcionalidades que demandaron las tutoras y superar las expectativas, a partir de su creatividad y conocimientos adquiridos durante la carrera.

Dr. C. Natalia Martínez Sánchez.
Tutora

+53 7 8358200
www.uci.cu
rector@uci.cu
@universidad_uci

INTRODUCCIÓN

En los últimos años, la aplicación automatizada de la minería de datos (etapa de análisis de "*Knowledge Discovery in Databases*" o KDD) ha permitido la fácil detección de patrones en los datos, mostrando una eficiencia en su aplicación y resultados a la hora su utilización en programas pensados y creados para todo tipo de usuarios.

La **minería de datos** es el proceso de analizar grandes cantidades de datos para encontrar tendencias y patrones. Permite extraer información de un conjunto de datos sin procesar y estructurarlos en información comprensible sobre diversas áreas(1). Utiliza los métodos de la inteligencia artificial, aprendizaje automático, estadística y sistemas de bases de datos. Involucra aspectos de bases de datos, gestión y procesamiento de datos, el modelo y las consideraciones de inferencia, métricas de intereses, consideraciones de la teoría de la complejidad computacional, entre otras.

Un proceso típico de minería de datos consta de los siguientes pasos generales(2):

1. **Selección del conjunto de datos**, tanto en lo que se refiere a las variables objetivo (aquellas que se quiere predecir, calcular o inferir), como a las variables independientes (las que sirven para hacer el cálculo o proceso), también hace referencia al muestreo de los registros disponibles.
2. **Análisis de las propiedades de los datos**, en especial los histogramas, diagramas de dispersión, presencia de valores atípicos y ausencia de datos (valores nulos).
3. **Transformación del conjunto de datos de entrada**, con el objetivo de prepararlo para aplicar la técnica de minería de datos que mejor se adapte a los datos y al problema, conocido como **pre procesamiento** de los datos.
4. **Seleccionar y aplicar la técnica de minería de datos**, se construye el modelo predictivo, de clasificación o segmentación.
5. **Extracción de conocimiento**, mediante una o varias técnicas de minería de datos, se obtiene un modelo de conocimiento, que representa patrones de comportamiento observados en los valores de las variables del problema o relaciones de asociación entre dichas variables.
6. **Interpretación y evaluación de datos**, una vez obtenido el modelo, se debe proceder a su validación comprobando que las conclusiones que arroja son válidas y suficientemente satisfactorias y establecer una comparación de los modelos en busca de aquel que se ajuste mejor al problema.

Entre las técnicas más representativas de la minería de datos se encuentran(3):

- **Redes neuronales**.- Son un paradigma de aprendizaje y procesamiento automático, inspirado en la forma en que funciona el sistema nervioso de los animales. Se trata de un sistema de inter conexión de neuronas en una red que colabora para producir un estímulo de salida. Algunos ejemplos de red neuronal son:
    - El perceptrón.
    - El perceptrón multicapa.
    - Los mapas autoorganizados, también conocidos como redes de Kohonen.
- **Regresión lineal**.- Es la más utilizada para formar relaciones entre datos. Rápida y eficaz pero insuficiente en espacios multidimensionales donde puedan relacionarse más de 2 variables.
- **Árboles de decisión**.- Un árbol de decisión es un modelo de predicción utilizado en el ámbito de la inteligencia artificial, dada una base de datos se construyen estos diagramas de construcciones lógicas, muy similares a los sistemas de predicción basados en reglas, que sirven para representar y categorizar una serie de condiciones que transcurren de forma sucesiva, para la resolución de un problema. Ejemplos:
    - Algoritmo ID3.
    - Algoritmo C4.5.
- **Modelos estadísticos**.- Es una expresión simbólica en forma de igualdad o ecuación que se emplea en todos los diseños experimentales y en la regresión para indicar los diferentes factores que modifican la variable de respuesta.
- **Agrupamiento o *Clustering***.- Es un procedimiento de agrupación de una serie de vectores según criterios habitualmente de distancia; se tratará de disponer los vectores de entrada de forma que estén más cercanos aquellos que tengan características comunes. Ejemplos:
    - Algoritmo K-means.
    - Algoritmo K-medoids.
    - Anomaly Detection
    - Class
    - Holotipo
- **Reglas de asociación**.- Se utilizan para descubrir hechos que ocurren en común dentro de un determinado conjunto de datos.

El **Reconocimiento de Patrones** es una de las ciencias de la minería de datos que se ocupa de los procesos sobre ingeniería, computación y matemáticas, relacionados con objetos físicos y/o abstractos, con el propósito de extraer información que permita establecer propiedades de o entre conjuntos de dichos objetos(2).

Entre los **enfoques** del Reconocimiento de Patrones se encuentra el **Lógico Combinatorio** que se basa en la idea de que la modelación del problema debe ser lo más cercana posible a la realidad del mismo, sin hacer suposiciones que no estén fundamentadas. Uno de los aspectos esenciales del enfoque es que las características utilizadas para describir a los objetos de estudio deben ser tratadas cuidadosamente(2).

Entre los **problemas** del Reconocimiento de Patrones se encuentra la **clasificación no supervisada**, la cual nos permite, dado un conjunto de objetos, agruparlos teniendo en cuenta varias características o rasgos en común, desconociéndose qué cantidad de agrupaciones resultarán formadas, qué cantidad de objetos pertenecerán a cada una o cuál es su finalidad. Estos grupos van a variar de dimensión y composición de acuerdo a los distintos valores de las variables de clasificación. En el aprendizaje no supervisado los datos de entrenamiento no incluyen etiquetas y el algoritmo intentará clasificar o descifrar la información por sí solo(2) (4).

El ***Agrupamiento*** o ***Clustering*** tiene como objetivo clasificar un conjunto de objetos en grupos, de forma tal que los objetos dentro de ese grupo posean un alto grado de semejanza, mientras que los pertenecientes a grupos diferentes sean poco semejantes entre sí. Es una herramienta muy utilizada en distintos contextos como (5) (6):

- La recuperación de información y la minería de textos.
- El procesamiento de secuencias descriptoras de genes y proteínas.
- El seguimiento y detección de sucesos en un flujo continuo de noticias.
- La segmentación de imágenes.
- La compresión de datos y el procesamiento de bases de datos especiales.
- La clasificación de zonas geográficas, la visualización de datos y la prospección geológica.
- La organización de documentos en bibliotecas.

Los algoritmos de clustering se utilizan para resolver problemas de aprendizaje automático no supervisado, en los datos no tienen ninguna etiqueta, no se puede saber si hay patrones ocultos en

los datos, así que dejamos que el algoritmo encuentre todas las conexiones que pueda(7) (8).

El algoritmo **Class** de agrupamiento no supervisado, utiliza como criterio de agrupamiento β0-compacto y realiza varias iteraciones tomando como objetos los grupos que se van formando. Su objetivo es lograr el agrupamiento de los objetos de una base de datos dada para saber la cantidad de grupos y sus características, los cuales serían las clases que se formarían.

Una vez que se tenga la cantidad de grupos y sus características (clases), se desea introducir un nuevo objeto y clasificarlo, entonces se necesitaría un representante de la clase, que es el objeto que más se parece a los restantes, que es lo que el algoritmo **Holotipo** nos plantea, utilizando como criterio de agrupamiento β0-conexo.

En nuestro país se creó el Centro de Estudios de Reconocimiento de Patrones y Minería de Datos, en abril de 2005 a partir de un grupo de profesores del Departamento de Computación de la Facultad de Matemática y Computación de la Universidad de Oriente. Está orientado a la investigación básica y aplicada en el área del Reconocimiento de Patrones y su aplicación a la Minería de Datos y Textos, cuya misión es desarrollar algoritmos para el procesamiento y análisis de grandes volúmenes de información estructurada o textual. Cuenta con una serie de proyectos terminados como: Construcción de herramientas para el Procesamiento del Lenguaje Natural: lematizador, desambiguador, reconocedor de nombres de entidades y analizador sintáctico; Sistema Automatizado para el diseño y procesamiento de encuestas; y Procesamiento automático de noticias(9).

En la Universidad de las Ciencias Informáticas (UCI) existe una Línea de investigación de Reconocimiento de Patrones e Inteligencia Artificial que se dedica a la búsqueda y desarrollo de módulos que aporten un valor agregado a las aplicaciones informáticas desarrolladas en los centros de desarrollo de software de la institución, por lo que interesa la formación profesional de los estudiantes en esta rama. En la Facultad de Ciencias y Tecnologías Computacionales (CITEC) se imparte en los años terminales de la carrera de Ingeniería en Ciencias Informáticas e Ingeniería en Bioinformática, la asignatura Reconocimiento de Patrones, que forma parte del currículo optativo y tributa a la línea de investigación, en donde el aprendizaje de los algoritmos de clasificación no supervisada, ocupan un papel esencial.

Durante la reciente pandemia de la COVID-19 provocada por el nuevo coronavirus SARS-CoV-2, se evidenció el necesario papel que cumplen las Tecnologías de la Información y las Comunicaciones en

el proceso de enseñanza y aprendizaje. Muchos estudiantes y docentes cubanos estuvieron forzados a pasar sin preparación previa de un entorno de aprendizaje presencial a uno virtual. Se presentaron diversos problemas en cuanto a la continuidad de estudios y, por tanto, el aprendizaje de estas asignaturas consideradas de difícil comprensión en condiciones de enseñanza normales se volvió aún más complejo.

Después de revisar el informe semestral de la asignatura de los últimos cinco años académicos se constataron las siguientes problemáticas:

- Insuficiencia en la capacidad de razonamiento y abstracción, mediante el análisis conceptual y algorítmico en los estudiantes, lo que limita la comprensión de los algoritmos que se estudian en la asignatura.
- Falta de una herramienta computacional que permita una visualización práctica del procedimiento y los resultados de los algoritmos.

De la situación problemática descrita anteriormente se deriva el siguiente **problema de investigación:** ¿Cómo implementar un sistema web de apoyo a la enseñanza de los algoritmos de clasificación no supervisada del Reconocimiento Lógico Combinatorio de Patrones?

Se define como **objeto de estudio:** Algoritmos de clasificación no supervisada del Reconocimiento Lógico Combinatorio de Patrones, por tanto, el **campo de acción** quedaría definido de la siguiente forma**:** Sistemas web que utilizan los algoritmos de clasificación no supervisada del Reconocimiento Lógico Combinatorio de Patrones para su enseñanza.

El **objetivo general** del presente trabajo se centra en**:** Implementar un sistema web que sirva de apoyo a la enseñanza de los algoritmos de clasificación no supervisada del Reconocimiento Lógico Combinatorio de Patrones en la Universidad de las Ciencias Informáticas.

Se establecen los siguientes **objetivos específicos:**

- Elaborar un marco teórico referencial sobre Reconocimiento de Patrones con un enfoque Lógico Combinatorio.
- Analizar los sistemas que apoyen la enseñanza de las técnicas del Reconocimiento Lógico Combinatorio de Patrones a nivel nacional e internacional.
- Seleccionar el algoritmo de clasificación no supervisada del Reconocimiento Lógico Combinatorio de Patrones para su implementación.

- Seleccionar la metodología, las tecnologías, los frameworks, y herramientas informáticas más acordes para la implamentación del sistema.
- Generar los artefactos ingenieriles correspondientes a cada una de las etapas de desarrollo del software.
- Implementar un sistema web que permita la clasificación no supervisada mediante el algoritmo seleccionado.
- Validar los resultados del sistema.

Métodos científicos de investigación:

Métodos teóricos: Permiten revelar las relaciones esenciales del objeto de investigación, no observables directamente. Participan en la etapa de asimilación de hechos, fenómenos y procesos(10).

- **Histórico-lógico:** Este método se empleó para identificar las tendencias actuales de los sistemas que utilizan algoritmos de clasificación no supervisada del Reconocimiento Lógico Combinatorio de Patrones y la evolución de estos, además en la compilación de información sobre aplicaciones análogas que los utilizan.
- **Inductivo-deductivo:** Este método se empleó para realizar los objetivos específicos y así obtener la información necesaria para cumplir con el objetivo general del trabajo.

Métodos empíricos: Revelan y explican las características del objeto. Se emplean en la primera etapa de colecta de información empírica y en la de comprobación experimental de la hipótesis del trabajo(10).

- **Observación**: Este método se utilizó mediante la observación directa de los algoritmos de clasificación no supervisada en proyectos en tiempo real, para registrar su comportamiento, describirlo y analizarlo.
- **Análisis de documentos**: Este método se utilizó para consultar y analizar información referente a los procesos de los algoritmos de clasificación no supervisada, cómo deben ser almacenados y tratados los datos, así como las pautas que se deben seguir.
- **Análisis de antecedentes:** Consiste en estudiar los sistemas web ya existentes que estén relacionados con la clasificación no supervisada del reconocimiento de patrones.

Técnicas de investigación científica: son procedimientos típicos, validados a obtener y transformar información útil para la solución de problemas de conocimiento en las disciplinas científicas(10).

- **Entrevista:** Esta técnica se utilizó para compilar toda la información necesaria sobre el algoritmo para establecer las necesidades del cliente y que el software cumpla con todos sus requerimientos.

El presente trabajo está compuesto por los tres capítulos siguientes:

- **Capítulo 1:** Marco teórico referencial sobre la automatización de la clasificación no supervisada: en este capítulo se exponen todos los elementos teóricos asociados a nuestro objeto de estudio, además de realizar una investigación de los distintos sistemas que utilizan los algoritmos de clasificación no supervisada, se hace una selección de la metodología a utilizar y se describen las técnicas y herramientas computacionales que van a aplicarse a la hora de la implementación.
- **Capítulo 2:** Características del sistema web USGest para la clasificación no supervisada del reconocimiento lógico combinatorio de patrones: en este capítulo se analiza la solución ofrecida mediante el modelado de la misma, la definición de los requisitos funcionales y no funcionales y las historias de usuario. Se realiza el diseño de la arquitectura con el uso del patrón arquitectónico Modelo-Vista-Controlador. Se establecen los patrones de diseño a utilizar y las tarjetas CRC. Se realiza el diseño del modelado de los datos, el mapa de navegación y las pantallas del sistema.
- **Capítulo 3: Validación de la propuesta de solución:** en este capítulo se muestra todo lo relacionado con la implementación, verificación y validación del sistema web de clasificación no supervisada, lo cual incluye las tareas de ingeniería, el diagrama de despliegue, las técnicas de validación, la estrategia de pruebas realizada y los resultados obtenidos.

CAPÍTULO I: MARCO TEÓRICO REFERENCIAL SOBRE LOS ALGORITMOS DE CLASIFICACIÓN NO SUPERVISADA DEL RECONOCIMIENTO LÓGICO COMBINATORIO DE PATRONES

En este capítulo, que consta de tres subepígrafes: se abordarán conceptos tales como: inteligencia artificial, inteligencia computacional, reconocimiento de patrones, sus enfoques y problemas, el problema de la clasificación no supervisada, los algoritmos de clasificación no supervisada, entre otros conceptos asociados al reconocimiento de patrones (objeto, patrón, rasgo, clase). Se definen las diferentes soluciones existentes en la actualidad que apoyen el aprendizaje de los algoritmos de clasificación no supervisada del Reconocimiento Lógico Combinatorio de Patrones. Por último, se establecerán las herramientas y tecnologías que serán empleadas en la ejecución del sistema en cuestión.

1.1 La clasificación no supervisada en el reconocimiento lógico combinatorio de patrones.

1.1.1 Inteligencia Artificial

La **Inteligencia Artificial (IA)** es una de las ramas de las ciencias de la computación. Es la habilidad de los ordenadores para hacer actividades que normalmente requieren inteligencia humana. Es la capacidad de las máquinas para usar algoritmos, aprender de los datos y utilizar lo aprendido en la toma de decisiones tal y como lo haría un ser humano(11) (12).

La IA se puede aplicar en casi todas las situaciones. Éstas son sólo algunas de las aplicaciones técnicas de la IA que están creciendo rápidamente en la actualidad(11):

- Reconocimiento de imágenes estáticas, clasificación y etiquetado: estas herramientas son útiles para una amplia gama de industrias.
- Mejoras del desempeño de la estrategia algorítmica comercial: ya ha sido implementada de diversas maneras en el sector financiero.
- Procesamiento eficiente y escalable de datos de pacientes: esto ayudará a que la atención médica sea más efectiva y eficiente.
- Detección y clasificación de objetos: puede verse en la industria de vehículos autónomos, aunque también tiene potencial para muchos otros campos.
- Distribución de contenido en las redes sociales: se trata principalmente de una herramienta de marketing utilizada en las redes sociales.

1.1.2 Reconocimiento de Patrones

El **Reconocimiento de Patrones** es la rama interdisciplinaria de las ciencias que se ocupa de los procesos sobre ingeniería, computación y matemáticas relacionados con objetos físicos y/o abstractos, con el propósito de extraer información que permita establecer propiedades de o entre conjuntos de dichos objetos y además se encarga de resolver problemas como clasificación de patrones(13).

Enfoques del Reconocimiento de Patrones(13):

- **Reconocimiento Estadístico de Patrones:** Se basa en la teoría de probabilidad y estadística y supone que se tiene un conjunto de medidas numéricas con distribuciones de probabilidad conocidas y a partir de ellas se hace el reconocimiento.
- **Reconocimiento Sintáctico de Patrones:** Se basa en encontrar las relaciones estructurales que guardan los objetos de estudio, utilizando la teoría de lenguajes formales. El objetivo es construir una gramática que describa la estructura del universo de objetos.
- **Redes Neuronales:** Supone que tiene una estructura de neuronas interconectadas que se estimulan unas a otras, las cuales pueden ser “entrenadas” para dar una cierta respuesta cuando se le presentan determinados valores.
- **Reconocimiento Lógico Combinatorio de Patrones:** Es la zona del conocimiento que se ocupa del desarrollo de teorías, métodos, técnicas y dispositivos computacionales para la realización de procesos ingenieriles, computacionales y/o matemáticos relacionados con objetos físicos y abstractos, que tienen el propósito de extraer información, que permite establecer propiedades y vínculos de o entre conjuntos de dichos objetos sobre la base de los cuales se realiza una tarea de identificación o clasificación.

Los **Problemas del Reconocimiento de Patrones** son todos aquellos relacionados con la clasificación de objetos y fenómenos y con la determinación de los factores que inciden en los mismos.

- **Selección de variables:** - Consiste en determinar cuál es el conjunto de características más adecuado para describir a los objetos(13). Tienen dos manifestaciones fundamentales:

  a) Para reducir el número de rasgos en términos de los cuales se deben describir los objetos en modo eficiente

  b) Para encontrar los rasgos que inciden en el problema de manera determinante.
- **Clasificación Supervisada**: Consiste en determinar cuál es el conjunto de características más adecuado para describir a los objetos(13). En el aprendizaje supervisado, los algoritmos usan datos que ya han sido etiquetados u organizados previamente para indicar cómo tendría que ser

categorizada la nueva información. Con este método, se requiere la intervención humana para proporcionar retroalimentación(11).

- **Clasificación Parcialmente Supervisada:** Dado un universo de objetos y el conocimiento acerca de la existencia de ciertas clases con características (propiedades) de especial interés y una muestra de objetos que pertenecen a cada una de ellas, el problema consiste en determinar para cada uno de los objetos no clasificados las relaciones de pertenencia de los mismos con cada una de las clases(13).
- **Clasificación no supervisada:** consiste en dada una muestra no clasificada encontrar la clasificación de la misma(13). En el aprendizaje no supervisado, los algoritmos no usan ningún dato etiquetado u organizado previamente para indicar cómo tendría que ser categorizada la nueva información, sino que tienen que encontrar la manera de clasificarlas ellos mismos. Por tanto, este método no requiere la intervención humana(11).

1.1.3 Clasificación No Supervisada

El problema de la **Clasificación No Supervisada** es una de las etapas más importantes en la mayoría de las ciencias. Resolver un problema de este tipo consiste en hallar la estructura interna de un conjunto de descripciones de objetos en el espacio de representación y agruparlos según su semejanza o cercanía, encontrando las relaciones entre ellos de un universo de rasgos las cuales se establecen sobre una similitud. Se desconocen tanto el número de agrupaciones resultantes una vez aplicado, como el uso o propósito que tendrán los objetos ya agrupados(2) (14).

Los algoritmos de clasificación no supervisada para su ejecución necesitan como entradas: un conjunto de objetos o matriz inicial (MI), además de una función de semejanza, un umbral de semejanza y un criterio de agrupamiento. Como salida se obtiene una matriz de aprendizaje (MA) que se compone por las clases o grupos resultantes.
Los pasos a seguir son:

- Calcular la Matriz de Semejanza (MS)
- Calcular el Umbral de Semejanza (β_0)
- Agrupar siguiendo el criterio de agrupamiento (π)

Matriz inicial o Muestra Inicial: es una matriz donde las filas son las descripciones de los objetos y las columnas los rasgos.

Función de Semejanza: nos permite la comparación o el grado de semejanza entre dos descripciones de objetos. A continuación, se presenta la fórmula para el cálculo de la función de semejanza que se implementará en el sistema, la cual nos indica que dado el rasgo X_i de los objetos O_t (objeto t) y O_o (objeto o) si son iguales suman 1 y 0 en otro caso.

$$\delta_i(X_i(O_o), X_i(O_t)) = \begin{cases} 1 \, si \, x_i(O_o) = x_i(O_t) \\ o e . o . c \end{cases}$$

Matriz de Semejanza (MS): a partir de MI y β (la medida de similaridad o semejanza) se puede construir una matriz que refleje las relaciones de semejanza entre todos los objetos sujetos a estudio. Es una matriz simétrica y la diagonal principal es 1 o 100 dependiendo de la escala en la que se trabaje(13).

Umbral de Semejanza (β_0): es la magnitud de semejanza que se exigirá que tengan los objetos de un mismo grupo o clase. Su valor puede ser dado por el experto o puede calcularse mediante cualquiera de las tres formas siguientes:(13).

A continuación, se presentan las distintas formas para el cálculo del umbral de semejanza, considerando ***m*** como la cantidad de objetos.

- Se calcula hallando la media del valor de semejanza entre los objetos.

$$\beta_0 = \frac{2}{m(m-1)} \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} \beta(O_i, O_j)$$

- Se calcula buscando el mínimo valor de semejanza por filas y seleccionando el mayor de esos mínimos.

$$\beta_0 = \frac{1}{m} \sum_{i=1}^{m} \underset{\substack{j=1\ldots m \\ i \neq j}}{max} \left\{ \beta\left(I(O_i), I(O_j)\right) \right\}$$

- Se calcula buscando el máximo valor de semejanza por filas y se divide entre la cantidad de objetos.

$$\beta_0 = \underset{\substack{i=1\ldots m-1 \\ i \neq j}}{max} \left\{ \underset{j=i+1\ldots m}{min} \left\{ \beta\left(I(O_i), I(O_j)\right) \right\} \right\}$$

Criterios de Agrupamiento (π): La selección de un criterio de agrupamiento puede realizarse de maneras diferentes(13):

- Se llega al criterio de agrupamiento mediante la modelación matemática del problema y por la misma vía se llega al paradigma de realización de la estructuración del conjunto de objetos.
- Se supone el paradigma, es decir, se impone, y se condiciona el criterio de agrupamiento de modo tal que resulte una estructuración acorde con el paradigma seleccionado.

De los 5 criterios de agrupamiento posibles: β_0 – compacto, β_0 – conexo, β_0 – fuertemente conexo, β_0 – fuertemente compacto y β_0 – completo maximal, se analizarán los dos primeros, debido a que son objetivos del contenido de la optativa Reconocimiento de Patrones.

- **Componente β_0 – compacto:** Se basa en tres condiciones: a) Todo elemento de B tiene en B al elemento que más se le parece que es **β_0** semejante con él; b) No existe fuera de B un elemento cuyo elemento más parecido que sea **β_0** este en B; c) y d) Dicen que B debe ser el conjunto más pequeño de cardinalidad mayor o igual que 1. Esto se traduce a lo siguiente: Dada una matriz de semejanza y un umbral de semejanza, se busca en la primera fila el mayor valor de semejanza de los mayores o iguales al umbral y se añaden los objetos relacionados a ese mayor a un grupo. Luego se analiza en la fila del objeto al cual se seleccionó su semejanza con el primero, así sucesivamente.
- **Componente β_0 – conexo:** Se basa en dos condiciones: a) Para cualquier par de elementos C existe una sucesión de elementos en C, que empieza en O_i y termina en O_j tales que uno es β_0 semejante al siguiente; b) No existe fuera de C un elemento β_0 semejante a un elemento de C. Esto se traduce a lo siguiente: Dada una matriz de semejanza y un umbral de semejanza, se buscan en la primera fila los mayores valores de semejanza de los mayores o iguales al umbral y se añaden los objetos relacionados a esos valores a un grupo. Luego se analizan las filas de los objetos a los cuales se seleccionaron su semejanza con el primero, así sucesivamente.

El nombre de los algoritmos está condicionado por la selección del criterio de agrupamiento, debido a que el procedimiento hasta el punto de seleccionar el criterio de agrupamiento es común para todos. En el caso de escoger el criterio Componente β_0 – compacto, se denomina Class y en el caso de escoger el criterio Componente β_0 – conexo, se denomina Holotipo.

1.1.4 Conceptos asociados al Reconocimiento de Patrones

Los **Datos** son un elemento importante para el desarrollo de la IA: sin ellos, sería casi imposible crear productos y aplicaciones con esta tecnología. El análisis de datos se basa generalmente en dos tipos de información: datos estructurados y no estructurados(11).

- Datos estructurados: incluyen la introducción de información, como valores numéricos, fechas, monedas o direcciones. pueden utilizarse para hacer recomendaciones y predicciones.

- Datos no estructurados: son más complicados de analizar, como textos, imágenes, audios y vídeos.

Un **Objeto** es una unidad dentro de un programa de computadora que consta de un estado y de un comportamiento, que a su vez constan respectivamente de datos almacenados y de tareas realizables durante el tiempo de ejecución. Un objeto puede ser creado instanciando una clase, como ocurre en la programación orientada a objetos, o mediante escritura directa de código y la replicación de otros objetos, como ocurre en la programación basada en prototipos(15). Los objetos pueden ser concretos o abstractos. Ejemplos de ellos pueden ser: un paciente, un conjunto de personas, un texto, entre otros.

Un **Patrón** es una propiedad, una regularidad, una cualidad invariante que expresa una relación estructural entre los elementos de una determinada configuración, disposición, composición. Es una sucesión de signos (orales, gestuales, gráficos, geométricos, numéricos, etc.) que se construye siguiendo una regla o algoritmo, ya sea de repetición o de recurrencia(16). Un patrón es un tipo de tema de sucesos u objetos recurrentes, como por ejemplo grecas, a veces referidos como ornamentos de un conjunto de objetos. Más abstractamente, podría definirse "patrón" como aquella serie de variables constantes, identificables dentro de un conjunto mayor de datos(17).

Un **Rasgo** es cualquier medida o cualidad extraíble. Conocido además como atributo, variable, característica, parámetro, propiedad o factor. Un rasgo nos aporta la información por la cual se va a guiar para la solución de los problemas del Reconocimiento de Patrones. Primeramente, se establece el dominio de dicho rasgo, que son los valores que puede tomar esa variable y por el tipo de datos que se establezca, se utilizará el criterio de comparación de rasgo correspondiente. Estos criterios pueden ser: por datos nominales o booleanos, por datos cuantitativos y por datos establecidos por intervalos o por conjuntos.
Ejemplos:
Rasgo (sexo) = dominio {femenino, masculino} -> criterio nominal o booleano
Rasgo (edad) = dominio {[10-19], [20-29], [30-39, [40-49]]} -> criterio por intervalos
Rasgo (fiebre) = dominio {36 a 42} -> criterio cuantitativo
Rasgo (alumnos) = dominio {(2, 3.4), (3, 5.1), (6, 8)} -> criterio por intervalos

Una **Clase** es una descripción abstracta de un grupo de objetos con propiedades similares (atributos), comportamiento común (operaciones), relaciones comunes con otros objetos y semántica común. Una clase es una construcción que se utiliza como un modelo (o plantilla) para

crear objetos de ese tipo. Un objeto creado a partir de una determinada clase se denomina una instancia de esa clase(15).

Un **Algoritmo** es un conjunto ordenado y finito de operaciones que permite hallar la solución de un problema. Se componen de tres partes principales(18):

- Entrada: Conjunto de datos que el algoritmo necesita como insumo para procesar.
- Proceso: Pasos necesarios aplicados por el algoritmo a la entrada recibida para llegar a la resolución del problema.
- Salida: Es el resultado producido por el algoritmo a partir del procesamiento de la entrada una vez terminada la ejecución del proceso.

1.1.5 Comparación del funcionamiento del algoritmo Class y Holotipo

En el siguiente ejemplo, se muestran los pasos a seguir para la clasificación no supervisada de los algoritmos Class y Holotipo para una mejor comprensión de su funcionamiento y se establece una comparación de sus resultados.

Dada una matriz inicial que posee un listado de objetos (pacientes) con sus rasgos por dominios ya identificados, un umbral de semejanza 0.7 dado por el experto, con una función de semejanza de igualdad se obtuvo la siguiente matriz de semejanza:

Tabla 1: Matriz de semejanza ejemplo

Objeto / Objeto	**Bruno**	**Sandra**	**Mónica**	**Edgard**	**Raúl**
Bruno	1	0.8	0.7	0.6	0.6
Sandra	0.8	1	0.3	0.4	0.5
Mónica	0.7	0.3	1	0.1	0.9
Edgard	0.6	0.4	0.1	1	0.1
Raúl	0.6	0.5	0.9	0.1	1

Se desea conocer cuántos grupos se formarán después de aplicar los algoritmos y cuáles serán los objetos pertenecientes a cada uno.

Para la agrupación de los objetos utilizando el criterio de agrupamiento β_0 – compacto del algoritmo Class, se busca en la matriz de semejanza, comenzando por la primera fila, aquellos valores iguales o mayores al umbral de semejanza (0.58) y se selecciona el mayor. Se adicionan a la lista los objetos a los cuales el valor tomado es su semejanza y luego se repite la operación buscando en la fila del segundo objeto.
Al finalizar se obtuvieron los siguientes grupos:

- G1 = {Bruno, Sandra}
- G2 = {Tamara, Raúl}
- G3 = {Edgard}

Para la agrupación de los objetos utilizando el criterio de agrupamiento β_0 – conexo del algoritmo Holotipo, se busca en la matriz de semejanza, comenzando por la primera fila, todos aquellos valores iguales o mayores al umbral de semejanza (0.58). Se adicionan a la lista los objetos a los cuales el valor tomado es su semejanza y luego se repite la operación buscando en las filas del resto de los objetos.

Al finalizar se obtuvieron los siguientes grupos:

- G1 = {Bruno, Sandra, Tamara, Raúl}
- G2 = {Edgard}

Al comparar las agrupaciones resultantes de la aplicación de los algoritmos Class y Holotipo en el ejemplo anteriormente propuesto, se puede apreciar que el primero solamente va a agrupar en un mismo conjunto a los objetos cuya semejanza sea mayor que el umbral establecido que más se asemejen entre sí, en cambio en el segundo algoritmo, los grupos quedarán formados por todos los objetos cuya semejanza sea mayor que el umbral establecido. Esto nos da la idea de que el algoritmo Class, ofrece un mayor nivel de especificidad debido a que están los elementos más β_0–semejantes, por tanto se selecciona para la implementación.

1.2 Soluciones computacionales para la clasificación no supervisada en el Reconocimiento Lógico Combinatorio de Patrones.

El desarrollo de soluciones computacionales enfocadas en la resolución de los problemas de Reconocimiento Lógico Combinatorio de Patrones, en especial los de clasificación no supervisada se encuentra restringido debido a que deben ser confeccionados de una manera abarcadora para que puedan modelarse problemas de diversa índole sin afectar los resultados de los mismos.

En la actualidad, existen múltiples herramientas o sistemas disponibles que utilizan módulos o componentes para los algoritmos de la clasificación no supervisada para la resolución de problemas específicos de una entidad o de un tema en particular que cubren los procesos de recolección y procesamiento de datos y la muestra de los resultados.

Ejemplo de ellas se analizan a continuación:

ArcGIS(19):

Es un conjunto de herramientas que permiten la visualización y manejo de información geográfica, y

que cuenta con una arquitectura extensible mediante la que pueden añadirse nuevas funcionalidades, entre las cuales se encuentran:

- Crear, compartir y utilizar mapas inteligentes.
- Compilar información geográfica.
- Crear y administrar bases de datos geográficas.
- Resolver problemas con el análisis espacial.
- Dar a conocer y compartir información mediante la geografía y la visualización.

QGIS 3(20)

Es un Sistema de Información Geográfica de código abierto que permite manejar formatos ráster, así como bases de datos. Nos brinda una serie de herramientas tales como:

- Cálculo de estadísticas e histogramas.
- Filtrajes: análisis digital para filtrar componentes de interés.
- Corrección y clasificación de imágenes de satélite.
- Detección de cambios.
- Cálculo de índices de vegetación y monitorización de incendios.

RapidMiner(1)

Basado en el machine learning y la minería de datos, se compone de una gran variedad de elementos de diferentes operadores. Cuenta con más de 500 operadores para toda clase de procedimientos y también combina esquemas de una herramienta stand-alone para el análisis de datos. Además, puede integrarse a los propios productos de los clientes.

KNIME

KNIME, o Konstanz Information Miner, es una plataforma gratuita de análisis, relaciones, integración, procesamiento y exploración de datos(1). Es usado por integrar diversos componentes para el aprendizaje de la máquina y la redacción de datos de su diseño modular de datos(21). Esta herramienta contiene módulos de clustering (clasificación no supervisada), los cuales predicen una clase para cada fila de la tabla o dataset de entrada.

Estos sistemas mundialmente reconocidos, comprenden una serie de módulos de clustering y funcionalidades como fue constatado; pero no existe un sistema que apoye a los estudiantes en la comprensión del funcionamiento y comportamiento de los diferentes algoritmos de clustering que son objetivos del contenido, por lo que surge la necesidad de implementar un sistema que utilice

algoritmos de clasificación no supervisada del Reconocimiento Lógico Combinatorio de Patrones como apoyo a su aprendizaje.

1.3 Técnicas y herramientas computacionales para la informatización de la clasificación no supervisada en el Reconocimiento Lógico Combinatorio de Patrones.

Para el desarrollo de aplicaciones y softwares se utilizan diversas técnicas, herramientas y tecnologías guiadas por la metodología de desarrollo de software seleccionada a las cuales se les hace referencia en el presente capítulo.

1.3.1 Metodología de desarrollo

Conceptos generales:

Metodología: Conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a realizar un nuevo software. Indica cómo hay que obtener los distintos productos parciales y finales(22).

Tarea: Actividades elementales en que se dividen los procesos(22).

Procedimiento: Definición de la forma de ejecutar la tarea(22).

Técnica: Herramienta utilizada para aplicar un procedimiento. Se pueden utilizar una o varias(22).

Herramienta: Para realizar una técnica, podemos apoyarnos en las herramientas de software que automatizan su aplicación(22).

Producto: Resultado de cada etapa de desarrollo(22).

Para la selección de la metodología se utilizó la estrella de Boehm y Turner, un modelo creado por Barry W. Boehm y Richard Turner el cual, a partir de una serie de criterios y consideraciones, establece bajo qué enfoque de desarrollo (ágil o prescriptivo) debería seguir el software. Los criterios son: tamaño del equipo, personal, cultura del equipo, criticidad y dinamismo.

El **enfoque prescriptivo**, denominado en algunas bibliografías como tradicional o pesado, busca la estructura, orden y consistencia del proyecto de desarrollo de software en cuestión(23).

El **enfoque ágil**, llamado también como enfoque ligero se centra en los miembros del equipo y su interacción, en la entrega rápida de versiones de software funcional, en la colaboración constante del cliente y la facilidad para manejar los cambios, dándole menor importancia a las herramientas, documentación, formalidad y planificación exhaustiva del proceso(23).

Ahora se analizarán los cinco criterios por sus definiciones, evaluados en el proyecto para decidir qué enfoque utilizar.

Tamaño: indica el número de personas involucradas en la ejecución del proyecto.
El proyecto está formado por un estudiante de ingeniería informática y dos doctoras en ciencias.

Personal: representa la experiencia de los integrantes del equipo con experiencia alta, media y baja.
El equipo del proyecto cuenta con dos doctoras en ciencias con una amplia experiencia por su categoría científica y tutoría a varias tesis relacionadas, además de una estudiante de ingeniería con un nivel Junior, pero con conocimientos ingenieriles y de desarrollo de softwares.

Cultura: se refiere a que las organizaciones y las personas que relaciona el proyecto pueden depender de la confianza o de la relación contractual.
El equipo de proyecto cuenta con una estructura de mando bien definida, donde cada miembro del equipo conoce sus responsabilidades y actividades. Existe una buena comunicación y confianza entre sus miembros. Las decisiones tomadas son previamente analizadas y consultadas con todos los involucrados. Las actividades son planificadas en función de los hitos del proyecto y asignadas a cada miembro, teniendo en cuenta la carga de trabajo y el rol que desempeñan. El trabajo es supervisado por la dirección del proyecto para identificar posibles problemas que provoquen el atraso del mismo.

Criticidad: se utiliza para conocer el impacto que provocaría cualquier falla en nuestro proyecto dependiendo de la importancia o finalidad del mismo, que pueden ser: pérdidas de vidas humanas, pérdidas de bienes esenciales y utilidades.
El proyecto que se realizará no constituye ningún problema respecto a pérdida de vidas humanas o de bienes ya que solo trabaja con información almacenada. El único inconveniente sería la confidencialidad y el uso de esos datos.

Dinamismo: se utiliza para representar la rapidez con la que el equipo reaccionará a los diferentes cambios de requerimientos del proyecto.
El proyecto puede estar sujeto a distintos cambios de requerimientos durante todo su ciclo de vida. Por tanto, es un riesgo que, para mitigarlo, se necesitan adoptar una serie de medidas en aras de que no se vea afectada la fecha de entrega final del producto.

La estrella quedaría constituida de la siguiente forma:

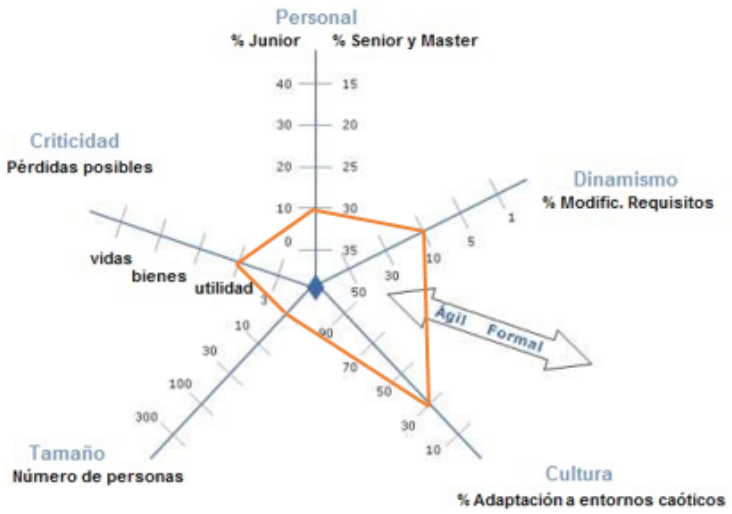


Ilustración 1: Estrella de Boehm y Turner. Fuente: Elaboración propia

Metodología ágil:

Una metodología ágil se basa en dos aspectos puntuales: retrasar las decisiones y la planificación adaptativa, potenciando el desarrollo de software a gran escala. Retrasar las decisiones y una planificación adaptativa reduce el número de decisiones de alta inversión que se toman, reducen el número de cambios necesarios en el proyecto y reducen el costo del cambio, por lo que suponen ventajas cuando se trata de la realización de un proyecto(24).

Sus principales ideas son(24):

- Los individuos y las interacciones entre ellos son más importantes que las herramientas y los procesos empleados.
- Es más importante crear un producto software que funcione antes que escribir una documentación exhaustiva.
- La colaboración con el cliente debe prevalecer sobre la negociación de contratos.
- La capacidad de respuesta ante un cambio es más importante que el seguimiento estricto de un plan.

Entre las metodologías ágiles más utilizadas tenemos a XP y a SCRUM(25):

Tabla 2: Comparación entre las metodologías ágiles SCRUM y XP.

Características	SCRUM	XP
Más enfocado en los procesos		
Más enfocado en las personas	X	X
Resultados rápidos	X	X

Cliente activo	X	X
Manejo del tiempo	X	X
Refactorización del código		X
Iterativo	X	X
Respuesta a los cambios	X	X

Por las características de XP reflejadas en la tabla anterior, se selecciona como metodología para continuar el proceso de desarrollo de nuestro sistema web.

Extreme Programming (XP)(24):

Es la más destacada de los procesos ágiles de desarrollo de software formulada por Kent Beck. La programación extrema se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad.

Las características fundamentales del método son(24):

- Desarrollo iterativo e incremental: pequeñas mejoras, unas tras otras.
- Pruebas unitarias continuas, frecuentemente repetidas y automatizadas, incluyendo pruebas de regresión. Se aconseja escribir el código de la prueba antes de la codificación.
- Frecuente interacción del equipo de programación con el cliente o usuario. Se recomienda que un representante del cliente trabaje junto al equipo de desarrollo.
- Corrección de todos los errores antes de añadir nueva funcionalidad. Hacer entregas frecuentes.
- Refactorización del código, es decir, reescribir ciertas partes del código para aumentar su legibilidad y mantenibilidad, pero sin modificar su comportamiento. Las pruebas han de garantizar que en la refactorización no se ha introducido ningún fallo.
- Propiedad del código compartida: en vez de dividir la responsabilidad en el desarrollo de cada módulo en grupos de trabajo distintos, este método promueve el que todo el personal pueda corregir y extender cualquier parte del proyecto. Las frecuentes pruebas de regresión garantizan que los posibles errores serán detectados.
- Simplicidad en el código: es la mejor manera de que las cosas funcionen. Cuando todo funcione se podrá añadir funcionalidad si es necesario. La programación extrema apuesta que es más sencillo hacer algo simple y tener un poco de trabajo extra para cambiarlo si se requiere, que realizar algo complicado y quizás nunca utilizarlo.

Ventajas de XP(24):

- Apropiado para entornos volátiles

- Estar preparados para el cambio, significa reducir su costo.
- Planificación más transparente para nuestros clientes, conocen las fechas de entrega de funcionalidades. Vital para su negocio
- Permitirá definir en cada iteración cuales son los objetivos de la siguiente
- Permite tener realimentación de los usuarios muy útil.

Fases del ciclo de desarrollo de XP(26):

- **Fase de planeación:** Esta fase inicia con las historias de usuario que describen las características y funcionalidades del software. El cliente asigna un valor o prioridad a la historia, los desarrolladores evalúan cada historia y le asignan un costo el cual se mide en semanas de desarrollo. En XP se realizan: las historias de usuario (HU), los requisitos no funcionales, el plan de iteraciones y el plan de entregas.
- **Fase de diseño**: El proceso de diseño debe procurar diseños simples y sencillos para facilitar el desarrollo. Este proceso se apoya en el uso de tarjetas CRC (Colaborador-Responsabilidad-Clase) la cual identifica las clases orientadas a objetos que son relevantes para el incremento del software. En XP se realiza: descripción de la arquitectura (*patrón arquitectónico*), diagrama de paquetes (*organización funcional*), especificación de cada paquete funcional, descripción de tarjetas CRC y patrones de diseño GRASP y GOF. Además, se adicionó el mapa de navegación para definir la jerarquía del sistema y un diagrama de clases entre el usuario y el sistema para una comprensión de los pasos a seguir desde la creación de las bases de datos hasta la visualización de los grupos.
- **Fase de codificación o desarrollo**: En esta fase los desarrolladores deben diseñar las pruebas de unidad que ejercitarán cada historia de usuario. En XP se realiza un desglose de las operaciones en las que se pueden implementar cada requisito funcional (HU) mediante las tareas de ingeniería o programación.
- **Fase de pruebas:** Las pruebas de unidad deben implementarse con un marco de trabajo que permita automatizarlas, con la finalidad de realizar pruebas de integración y validación diarias, esto proporcionará al equipo un indicador del progreso y revelarán a tiempo si existe alguna falla en el sistema. En XP realizan las pruebas unitarias y de aceptación.

1.3.2 Ambiente de desarrollo

Entorno de Desarrollo Integrado (IDE):

Los IDEs comenzaron integrando componentes básicos como un compilador, un editor y un depurador, sin embargo, con el pasar de los años los IDEs se han convertido en programas especializados

para el desarrollo de softwares, convirtiéndose en herramientas que en la actualidad permiten maximizar el potencial de los programadores, facilitando todas las herramientas y procesos repetitivos que permitan mejoran la productividad en el desarrollo de sistemas, permitiendo que los desarrolladores sólo se enfoquen en la lógica(27).

PyCharm:

Este IDE cuenta con un sistema de control local de versiones, que permite la vinculación con GitHub, el servicio basado en la nube que se encuentra alojado en https://github.com e implementa a Git, un VCS[1] distribuido de código abierto y gratuito, diseñado para manejar todo, desde proyectos pequeños hasta muy grandes, con velocidad y eficiencia(28).

Visual Estudio Code 1.72.2:

El editor de código de Visual Estudio o Visual Estudio Code, es una potente herramienta que permite introducir el código de la aplicación. Las palabras reservadas y los tipos de datos se colorean. Esto permite facilitar la lectura y la comprensión del código(29).

Al ser PyCharm un IDE muy completo, consume muchos recursos para poder usarse, por lo tanto, Visual Estudio Code es una buena opción para ordenadores de bajo rendimiento.

Lenguajes de programación del lado del cliente:

Actualmente existen diferentes lenguajes de programación, cada uno enfocado a un proceso en particular, siendo algunos más fácil de aprender que otros. Los lenguajes como HTML (Lenguaje de Modelado de Hipertexto), CSS (Hoja de Estilos en Cascada) y el JavaScript, de ahora en adelante JS, presentan gran demanda, ya que son fáciles de aprender y son recursos predominantes para el diseño de sitos web, logrando generar páginas ricas en contenidos, manejando una presentación estructurada y mejorando su comportamiento gracias a la utilización de Scripts como JS, el cual es un programa insertado o embebido en un documento HTML(30).

Este conjunto de tecnologías (HTML, CSS y JS) son las utilizadas para programar sitios web del lado del cliente, es decir, que se ejecutan del lado del navegador del usuario, no del servidor donde se almacena la página web diseñada, logrando así, “ofrecer al usuario interfaces gráficas mucho más ricas y a la vez complejas, controlar formularios de forma más eficiente, brindar un número de facilidades al usuario y proporcionar un intercambio más interactivo”(30).

1 Los **VCS** (version control system) son aplicaciones que permiten realizar el control de modificaciones de código fuente de manera automática y eficiente.

HTML 5:

HTML es un lenguaje de marcas de hipertexto o de etiquetado con el que se escriben las páginas web. […]Es el lenguaje usado por los navegadores para mostrar las páginas web al usuario, siendo hoy la interfaz más extendida en la red. Nos permite aglutinar textos, sonidos e imágenes y combinarlos a nuestro gusto. Además, nos permite la introducción de referencias a otras páginas por medio de enlaces de hipertexto(31).

Es el lenguaje de marcas más utilizado para hacer páginas web. Se utiliza para describir la estructura y el contenido de las páginas Web en forma de texto, y para complementarlo con otros objetos tales como imágenes, sonidos y videos. Se escribe en forma de etiquetas rodeadas por corchetes angulares (<>). Los archivos escritos en este lenguaje tienen normalmente extensión HTM o HTML(15).

CCS 3:

CSS es un lenguaje de hojas de estilo creado para controlar la presentación de los documentos electrónicos definidos con HTML y XHTML. Es la mejor forma de separar los contenidos y su presentación y es imprescindible para la creación de páginas web complejas(30).

Es un lenguaje de hojas de estilos creado para controlar el aspecto o presentación de los documentos electrónicos definidos con HTML y XHTML. CSS es la mejor forma de separar los contenidos y su presentación y es imprescindible para crear páginas Web complejas(15).

Bootstrap 4:

Bootstrap es un framework desarrollado y liberado por Twitter, que tiene como gran objetivo facilitar el diseño web sirviendo de guía para emplear buenas prácticas y estándares. Permite crear de manera fácil sitios web que son adaptables o responsivas para ajustarse a cualquier dispositivo y tamaños de pantalla. Consiguiendo un sitio web muy organizado visualmente. Posee un sistema de grillas en la cual maquetamos el sitio mediante columnas que lo hace bastante fácil. Se integra perfectamente con librerías de JavaScript(32).

JavaScript:

JavaScript es un lenguaje interpretado usado para múltiples propósitos, pero principalmente para crear páginas web dinámicas. Una página web dinámica es aquella que incorpora efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario. Por ser un lenguaje de programación interpretado, no es necesario

compilar los programas para ejecutarlos, sea que los programas hechos en JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios(31).

JavaScript es un lenguaje de programación que se utiliza principalmente para crear páginas Web dinámicas. JavaScript es un lenguaje de programación interpretado. JavaScript no guarda ninguna relación directa con el lenguaje de programación Java. Se utilizó en las validaciones de campos y datos del sistema de las páginas WEB en las distintas librerías libres disponibles de la WEB(15).

Lenguaje de desarrollo del lado del servidor:

Los lenguajes de programación del lado del servidor son especialmente útiles en trabajos en que se tiene que acceder a información centralizada, situada en una base de datos en el servidor, y cuando por razones de seguridad no se pueden realizar en la computadora del usuario. Son necesarios porque para hacer la mayoría de las aplicaciones web se debe tener acceso a muchos recursos externos a la computadora del cliente, principalmente bases de datos alojadas en servidores de internet(33).

Java:

Java es un lenguaje de programación de alto nivel orientado a objetos. Esta tecnología permite el desarrollo de programas capaces de ejecutarse en entornos distribuidos y heterogéneos, programas ejecutables en muchas plataformas hardware y software, principalmente dispositivos electrónicos embebidos(34).

Python 3.9:

Python es un lenguaje de programación interpretado, orientado a objetos de alto nivel y con semántica dinámica. Su sintaxis se hace énfasis en la legibilidad del código, lo que facilita su depuración y favorece la productividad. Ofrece la potencia y la flexibilidad de los lenguajes compilados con una curva de aprendizaje suave. Aunque fue creado como lenguaje de programación de uso general, cuenta con una serie de librerías y entornos de desarrollo para cada una de las fases del proceso de Data Science(35).

Por ser un lenguaje fácil de aprender y por ser además de alto nivel ya que contiene implícitas algunas estructuras de datos como listas, diccionarios, conjuntos y tuplas, que permiten realizar algunas tareas complejas en pocas líneas de código y de manera legible(36), se selecciona Python para la programación en el servidor(37).

Gestor de Bases de Datos:

Un Sistema Gestor de Bases de Datos (SGBD) es un conjunto de aplicaciones informáticas que nos permite manejar bases de datos. Este tipo de programas sirven para que los usuarios y las bases de datos se puedan comunicar de forma sencilla. Estos sistemas no solo comprenden los programas para gestión de datos, sino que también incluyen los propios datos almacenados, que normalmente se encuentran relacionados(38).

Entre sus características principales tenemos(39):

− Independencia: Los datos se gestionan independientemente de los programas que los vayan a utilizar teniendo así una independencia lógica y de los ficheros en los que se almacenen obteniendo una independencia física.
− Gestión de concurrencia de acceso.
− Integridad de datos: Mediante transacciones ACID y BASE.
− Persistencia.
− Recuperación frente a fallos
– Seguridad
– Rendimiento
− Lenguajes de Definición y Manipulación de Datos.

Las bases de datos relacionales han evolucionado de forma que hoy en día se puede hablar de los siguientes ejemplos: MariaDB, PostgreSQL, Oracle, Access, MySQL, SQL Server entre otras. Algunas de las mencionadas son propietarias y otras de código abierto(39).

MariaDB:

MariaDB es un servidor de base de datos de código abierto. La implementación que ofrece, posee una mayor gama de opciones de configuración, administración y rendimiento(40).

PostgreSQL 13.2:

PostgreSQL es un gestor de bases de datos orientadas a objetos (SGBDOO o ORDBMS en sus siglas en inglés) muy conocido y usado en entornos de software libre porque cumple los estándares SQL92 y SQL99, y también por el conjunto de funcionalidades avanzadas que soporta, lo que lo sitúa al mismo o a un mejor nivel que muchos SGBD comerciales(41).

Se escogió PostgreSQL debido a que está muy bien catalogado por su estabilidad, potencia, robustez y la facilidad de administración e implementación. Adicionalmente, utiliza un sistema cliente servidor con el uso de hilos para un procesamiento correcto de las consultas hacia la base de datos(39).

Herramienta CASE:

Existen una gran variedad de herramientas CASE (Computer Aided Software Engineering o Ingeniería de Software Asistida por Computadora) para el proceso de desarrollo de software. Las herramientas CASE, están tomando cada vez más relevancia en la planeación y ejecución de proyectos que involucren sistemas de información, pues suelen inducir a sus usuarios a la correcta utilización de metodologías que le ayudan a llegar con facilidad a los productos de software construidos. Todas las herramientas CASE prestan soporte a un lenguaje de modelado para acompañar la metodología y es lógico suponer, que un alto porcentaje de ellas soportan el Lenguaje de Modelado Unificado (UML). Esto se debe a la amplia aceptación de este lenguaje, su valor conceptual y visual, así como su facilidad para ser extendido para representar elementos particulares a determinados tipos de aplicaciones(42).

Visual Paradigm for UML 8.0:

Es una herramienta profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Agiliza la construcción de aplicaciones con calidad y a un menor coste. Posibilita la generación de bases de datos, transformación de diagramas de Entidad-Relación en tablas de base de datos, así como ingeniería inversa de bases de datos(43).

Lenguaje de Modelado UML 2.0

“El modelado constituye una simplificación de la realidad donde se define lo esencial para la construcción del software con los objetivos de comunicar la estructura de un sistema complejo, especificar el comportamiento deseado del sistema, comprender mejor lo que se está desarrollando y descubrir oportunidades de simplificación y reutilización”(44).

UML es el acrónimo de Lenguaje Unificado de Modelado, es el lenguaje estándar para visualizar, especificar, construir y documentar los artefactos de un sistema, utilizándose para el modelado del negocio y sistemas de software. También ofrece un estándar para describir los modelos, incluyendo aspectos conceptuales como procesos de negocio, funciones del sistema, expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables(44).

Framework

Un framework es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. En programación, es un set de funciones o código genérico que realiza tareas comunes y frecuentes en todo tipo de aplicaciones (creación de objetos, conexión a base de da-

tos, etc.). Esto brinda una base sólida sobre la cual desarrollar aplicaciones concretas y permite obviar los componentes más triviales y genéricos del desarrollo y abre camino a que diseñadores y programadores puedan pasar más tiempo identificando requerimientos de software(45).

Flask:

Flask es un framework minimalista escrito en Python que hace uso de rutas para redirigir a una estructura de páginas HTML y de extensiones(46).

Django 3.2.15:

Django representa un marco de trabajo para el desarrollo rápido de sistemas de información web con Python. Considera algunas funcionalidades listas para usar para facilitar el desarrollo de aplicaciones web. Como resultado, no es necesario escribir todo el código ni usar tiempo para buscar errores de código en el framework. El desarrollo de sistemas de información web se hace rápido, seguro, escalable y también fáciles de mantener(47).

Los dos frameworks ofrecen un backend limpio en manejo de usuarios, pudiendo limitar memoria y velocidad para evitar caídas en el servidor. Django cuenta además con un sistema de autenticación de usuarios, un ORM completo y un sistema de plantillas, mientras que Flask no. El desempeño en velocidad de Flask es superior gracias a su diseño minimalista en su estructura. Se seleccionó Django como framework por ser muy completo, robusto, y contener todos los elementos necesarios para el desarrollo y validación de aplicaciones de alto nivel.

Conclusiones del capítulo

Tras el análisis y conceptualización realizados durante la investigación y con la ayuda de la entrevista realizada, se posibilitó la extracción de las características esenciales sobre los algoritmos de clasificación no supervisada del Reconocimiento Lógico Combinatorio de Patrones.

Se examinó sobre sistemas que utilizan estos algoritmos y no se encontró ninguna herramienta de clasificación no supervisada en el Reconocimiento Lógico Combinatorio de Patrones con fines académicos debido a la alta complejidad en los procesos de las mismas.

Se identificó como metodología ágil a utilizar la metodología XP, además de todas las herramientas a manejar como son: lenguaje Python 3.9 para el servidor con Django 3.2.15 como framework, HTML 5, CSS 3 y JavaScript como lenguajes del lado del cliente, Visual Paradigm for UML 8.0 para el modelado de los artefactos, Visual Studio Code 1.72.2 como IDE y PostgreSQL 13.2 para gestionar

las bases de datos.

CAPÍTULO II: CARACTERÍSTICAS DEL SISTEMA WEB USGEST PARA LA CLASIFICACIÓN NO SUPERVISADA DEL RECONOCIMIENTO LÓGICO COMBINATORIO DE PATRONES

En este capítulo se muestra una breve descripción del sistema propuesto, el modelado de los procesos del negocio, lo cual incluye el establecimiento de los requisitos funcionales y no funcionales. También se presenta el diseño e implementación del almacenamiento y los procesos y flujos de información.

2.1 Propuesta del sistema

Con este trabajo se propone la implementación de un sistema web con el nombre “USGest” (Unsupervised Gest), de fácil comprensión, que utilizará la arquitectura Modelo-Vista-Controlador (MVC), para ser utilizado por el Departamento de Inteligencia Computacional, como herramienta de apoyo en el proceso de enseñanza de los algoritmos de clasificación no supervisada.

Este sistema contará con una autenticación de usuario, por tanto, solo las personas registradas en él podrán acceder.
Contará con dos roles establecidos: el administrador de usuario y el usuario del sistema. El administrador de usuario solo podrá manejar lo referente a los usuarios y sus permisos, y el usuario, podrá realizar todo lo referente a la clasificación.

Para la clasificación, los usuarios deberán primeramente crear los rasgos que desean tener en su base de datos y ajustar sus características. Después deberán introducir los objetos que desean clasificar y asignarle un valor a cada uno de los rasgos ya definidos anteriormente. Por último, deberá seleccionar la forma en que quieren clasificar esos objetos ya caracterizados al seleccionar el umbral y el criterio de agrupamiento, dando como resultado la confección de los grupos.

2.2 Definición de requisitos, análisis y diseño del sistema web USGest

En el siguiente epígrafe se exponen los artefactos resultantes de la ingeniería de requisitos, el modelado del análisis, la descripción de la arquitectura y el diseño de la propuesta de solución realizados.
Se realizó una entrevista (**ver Anexo 1**) a los jefes de la línea de Reconocimiento de Patrones e Inteligencia Artificial, la cual tuvo como objetivo conocer en detalles el funcionamiento de los algoritmos para una correcta ejecución de los procesos a implementar.

2.2.1 Requisitos funcionales

Un **requisito funcional** define una función del sistema de software o sus componentes. Una función

es descrita como un conjunto de entradas, comportamientos y salidas. Los requerimientos funcionales pueden ser: cálculos, detalles técnicos, manipulación de datos y otras funcionalidades específicas que se supone, un sistema debe cumplir(48). Los requerimientos de comportamiento para cada requerimiento funcional se muestran en las historias de usuarios. A continuación, se listan los requisitos que contiene la propuesta de solución con una breve descripción de cada uno:

Tabla 3: Listado de requisitos funcionales.

No.	Requisitos	Descripción
RF-1	Crear rasgo	El sistema debe permitir crear un rasgo. Se deberán ingresar los datos siguientes: • Nombre del rasgo • Dominios También se deberá seleccionar el criterio de comparación entre: • Nominal • Booleano • Cuantitativo • Por intervalos
RF-2	Listar rasgo	El sistema debe permitir listar todos los rasgos y sus dominios.
RF-3	Modificar rasgo	El sistema debe permitir modificar los rasgos. Se deberán modificar los datos siguientes: • Nombre del rasgo • Dominios También se deberá seleccionar el criterio de comparación entre: • Nominal • Booleano • Cuantitativo • Por intervalos
RF-4	Eliminar rasgo	El sistema debe permitir eliminar los rasgos.
RF-5	Crear base de datos	El sistema debe permitir crear una base de datos. Se deberán ingresar los datos siguientes: • Nombre de la base de datos
RF-6	Listar base de datos	El sistema debe permitir listar todas las bases de datos.
RF-7	Modificar base de datos	El sistema debe permitir modificar las bases de datos. Se deberán modificar los datos siguientes: • Nombre de la base de datos
RF-8	Eliminar base de datos	El sistema debe permitir eliminar las bases de datos.
RF-9	Ver base de datos	El sistema debe permitir ver las bases de datos con sus rasgos y dominios de rasgos.
RF-10	Adicionar objetos	El sistema debe permitir adicionar objetos. Se deberán ingresar los datos siguientes: • Nombre del objeto También se deberán seleccionar los distintos dominios de cada rasgo añadido.

RF-11	Modificar objetos	El sistema debe permitir modificar los objetos. Se deberán ingresar los datos siguientes: • Nombre del objeto También se deberán modificar los distintos dominios de cada rasgo añadido.
RF-12	Eliminar objetos	El sistema debe permitir eliminar los objetos.
RF-13	Listar objetos	El sistema debe permitir listar todos los objetos.
RF-14	Ver objetos	El sistema debe permitir ver los objetos.
RF-15	Crear grupos	El sistema debe permitir crear grupos. Se deberán seleccionar: • Umbral de semejanza: con las opciones de: o Criterio de expertos (escribir un número entre 0 y 1) o Calculado por el sistema (Mínimo, Máximo) • Criterio de agrupamiento: con las opciones de: o Conexa o Compacta
RF-16	Ver grupo	El sistema debe permitir ver el grupo seleccionado.
RF-17	Cerrar sesión de usuario	El sistema debe permitir cerrar la sesión del usuario ya autenticado en el sistema
RF-18	Autenticar usuario	El sistema debe permitir autenticar un usuario. Se deberán ingresar los datos siguientes: • Usuario • Contraseña
RF-19	Adicionar usuario	El sistema debe permitir adicionar usuarios. Se deberán ingresar los datos siguientes: • Nombre • Usuario • Contraseña
RF-20	Listar usuario	El sistema debe permitir listar todos los usuarios.
RF-21	Modificar usuario	El sistema debe permitir modificar los usuarios. Se deberán ingresar los datos siguientes: • Nombre • Usuario • Contraseña
RF-22	Eliminar usuario	El sistema debe permitir eliminar los usuarios.

2.2.2 Personas relacionadas con el sistema

Se les llama a las **personas relacionadas con el sistema** a todas aquellas que interactúan con el mismo, las cuales obtienen resultados de sus procesos. A continuación se detallan los roles que van a cumplir estas personas dentro de la aplicación.

Tabla 4: Personas relacionadas con el sistema

Personas relacionadas con el sistema	**Funciones**
Usuario del sistema	Realiza las actividades de: Gestión de Rasgos, Gestión de Bases de datos, Gestionar objetos, Administrar grupos y además autenticarse en el sistema.

Administrador del sistema	Realiza la gestión de los usuarios del sistema.

2.2.3 Historias de usuario

Las **historias de usuario** se usan, en el contexto de la ingeniería de requisitos ágil, como una herramienta de comunicación que combina las fortalezas de ambos medios: escrito y verbal. Describen, en una o dos frases, una funcionalidad de software desde el punto de vista del usuario, con el lenguaje que éste emplearía. El foco está puesto en qué necesidades o problemas soluciona lo que se va a construir(49).

Ventajas de las historias de usuario:

- Proporcionan la documentación necesaria fomentando a la vez el debate.
- Fomentan la colaboración entre todos interesados y el equipo ágil.
- Se escriben en el lenguaje del usuario, manteniendo así una relación cercana con el cliente.
- Involucran y captan al cliente para el proceso y para el producto.
- Por su naturaleza son independientes.
- Facilitan la planificación e implementación.

Las historias de usuario tienen la siguiente estructura:

- Número: Número de la historia.
- Usuario: Rol que lleva a cabo la funcionalidad.
- Nombre: Nombre de la historia, debe estar relacionado con el objetivo de la funcionalidad a implementar. Representa un requisito funcional del sistema.
- Prioridad en el negocio: Importancia y urgencia para el negocio. Puede ser: Alta, Media o Baja.
- Riesgo en desarrollo: Probabilidad de que existan problemas técnicos en su desarrollo que influyan en su terminación en tiempo. Puede ser: Alto, Medio o Bajo.
- Tiempo estimado: Entre 1 y 3 puntos, equivalentes a semanas.
- Iteración asignada: Numero de la iteración en que se implementara la historia de usuario.
- Programador responsable: Programador del equipo de desarrollo que la implementara.
- Descripción: Descripción de la funcionalidad a realizar por la historia de usuario.
- Observaciones: Restricciones asociadas a la historia.
- Prototipo de interfaz gráfica elemental: Aquí se coloca una ilustración con la interfaz gráfica de esta historia de usuario.

Tabla 5: Historia de usuario Gestionar rasgos

Historia de Usuario	
Número: HU 1	**Usuario:** Usuario del sistema
Nombre: Gestionar rasgos.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto
Tiempo estimado: 1	**Iteración asignada:** 1
Programador responsable: Susana Cremé	
Descripción: Permite insertar, modificar, eliminar y listar los rasgos.	
Observaciones: Este recoge los siguientes datos de cada rasgo: nombre, criterio, dominio.	
Prototipo de interfaz gráfica elemental: 	

Tabla 6: Historia de usuario Gestionar bases de datos

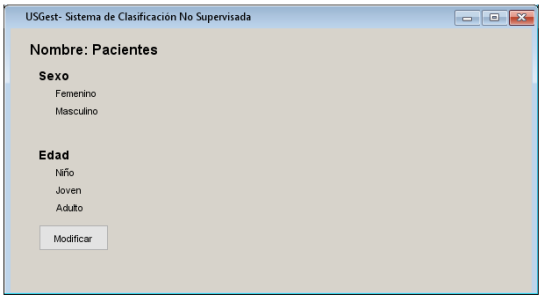
Historia de Usuario	
Número: HU 2	**Usuario:** Usuario del sistema
Nombre: Gestionar bases de datos.	
Prioridad en negocio: Alto	**Riesgo en desarrollo:** Alto
Tiempo estimado: 1	**Iteración asignada:** 1
Programador responsable: Susana Cremé	
Descripción: Permite insertar, modificar, eliminar, listar y ver las bases de datos.	
Observaciones: Este recoge los siguientes datos de cada base de datos: nombre.	
Prototipo de interfaz gráfica elemental:  	

Tabla 7: Historia de usuario Gestionar objetos.

Historia de Usuario	
Número: HU 3	**Usuario:** Usuario del sistema

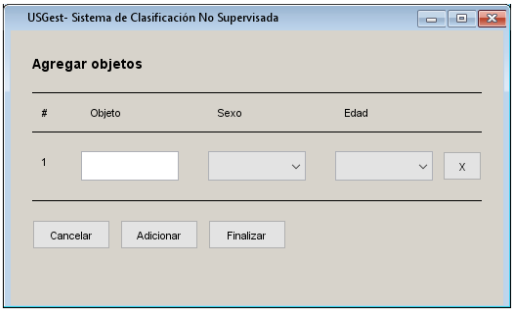
Nombre: Gestionar objetos.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto
Tiempo estimado: 2	**Iteración asignada:** 1
Programador responsable: Susana Cremé	
Descripción: Permite insertar, modificar, eliminar, listar y ver los objetos.	
Observaciones: Este recoge los siguientes datos de cada objeto: nombre, rasgos.	
Prototipo de interfaz gráfica elemental:  	

Tabla 8: Historia de usuario Administrar grupos.

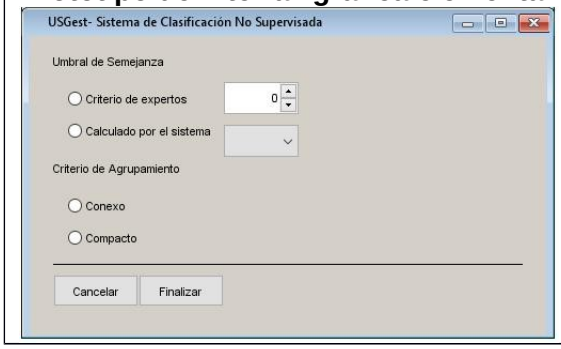
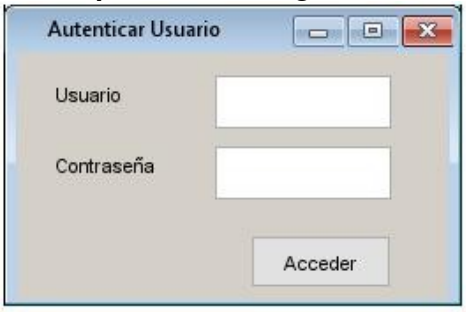
Historia de Usuario	
Número: HU 4	**Usuario:** Usuario del sistema
Nombre: Administrar grupos.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto
Tiempo estimado: 3	**Iteración asignada: 2**
Programador responsable: Susana Cremé	
Descripción: Permite crear y ver los grupos.	
Observaciones: Este recoge los siguientes datos de cada grupo: umbral de semejanza y criterio de agrupamiento.	
Prototipo de interfaz gráfica elemental: 	

Tabla 9: Historia de usuario Gestionar usuario.

Historia de Usuario	
Número: HU 5	**Usuario:** Administrador
Nombre: Gestionar usuario.	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Bajo
Tiempo estimado: 1	**Iteración asignada:** 3
Programador responsable: Susana Cremé	
Descripción: Permite insertar, modificar, eliminar y listar los usuarios.	
Observaciones: Este recoge los siguientes datos de cada usuario: nombre, usuario, contraseña.	

Prototipo de interfaz gráfica elemental:

Tabla 10: Historia de usuario: Autenticar usuario.

Historia de Usuario	
Número: HU 6	**Usuario:** Administrador, Usuario del sistema
Nombre: Autenticar usuario.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Media
Tiempo estimado: 1	**Iteración asignada:** 3
Programador responsable: Susana Cremé	
Descripción: Permite acceder a la aplicación una vez registrado correctamente y poder asignárs-ele los permisos según su rol	
Observaciones: Este recoge los siguientes datos de cada usuario para su autenticación: usuario y contraseña.	
Prototipo de interfaz gráfica elemental: 	

2.2.4 Requisitos no funcionales

Los requisitos no funcionales son requisitos que especifican criterios que pueden usarse para juzgar la operación de un sistema en lugar de sus comportamientos específicos, ya que éstos corresponden a los requisitos funcionales. Por tanto, se refieren a todos los requisitos que ni describen información a guardar, ni funciones a realizar(50).

A continuación, se muestra un listado con los requisitos no funcionales definidos para la propuesta de solución que se presenta.

RNF1 Hardware

RNF 1.1- El servidor web y el servidor de base de datos requieren 2 GB de RAM, un CPU Intel I3 2.4 GHz y al menos 40 Gb de almacenamiento.

RNF 1.2- El cliente requiere una máquina con al menos 128 MB de RAM.

RNF 1.3- Todas las terminales implicadas deben encontrarse conectadas a la red con una velocidad de al menos 100 Mbps.

RNF2 Software

RNF 2.1- El programa puede ser ejecutado en cualquier navegador web y bajo cualquier sistema operativo.

RNF3 Seguridad

RNF 3.1- Confiabilidad: La información guardada por los profesionales debe de tener un permiso de autenticación de usuario.

RNF 3.2- Se deberá verificar antes de eliminar datos (acciones irreversibles).

RNF 3.2- CSRF[2] protection

RNF 3.4- XSS[3] protection

RNF 3.5- HTTPS[4]

RNF4 Apariencia o interfaz externa

RNF 4.1- Los colores de las interfaces graficas serán: azul y blanco.

RNF 4.2- Será libre de contenido disociable.

RNF5 Restricciones del diseño y la implementación

RNF 5.1- Se utilizará como herramienta CASE de modelado el Visual Paradigm for UML 8.0

RNF 5.2- Se empleará como lenguaje de programación del lado del servidor Python 3.9.

RNF 5.3- Se empleará como lenguajes del lado del cliente HTML5, CSS3 y JavaScript.

RNF 5.4- Se utilizará como IDE Visual Estudio Code 1.72.2.

RNF 5.5- Se empleará como gestor de bases de datos Postgre SQL 13.2.

RNF6 Usabilidad

RNF 6.1- El sistema podrá ser utilizado por los estudiantes y profesionales autorizados que tengan un conocimiento esencial del reconocimiento de patrones y además deberán recibir un adiestramiento básico para su uso.

2.2.5 Plan de iteraciones

Las historias de usuarios seleccionadas para cada entrega son desarrolladas y probadas en un ciclo

2 El CSRF (del inglés Cross-site request forgery o falsificación de petición en sitios cruzados) es un tipo de exploit malicioso de un sitio web en el que comandos no autorizados son transmitidos por un usuario en el cual el sitio web confía.

3 XSS, del inglés Cross-site scripting es un tipo de inseguridad informática o agujero de seguridad típico de las aplicaciones Web, que permite a una tercera parte inyectar en páginas web vistas por el usuario código JavaScript o en otro lenguaje script similar, evitando medidas de control como la Política del mismo origen.

4 Hypertext Transfer Protocol Secure (Protocolo seguro de transferencia de hipertexto), es un protocolo de aplicación basado en el protocolo HTTP, destinado a la transferencia segura de datos de Hipertexto, es decir, es la versión segura de HTTP.

de iteración, de acuerdo al orden preestablecido. Al comienzo de cada ciclo, se realiza una reunión de planificación de la iteración. Cada historia de usuario se traduce en tareas específicas de programación.

Asimismo, para cada historia de usuario se establecen las pruebas de aceptación. Estas pruebas se realizan al final del ciclo en el que se desarrollan, pero también al final de cada uno de los ciclos siguientes, para verificar que subsiguientes iteraciones no han afectado a las anteriores(51). Este plan se encarga de mostrar las historias de usuario que deberán implementarse en cada iteración, así como la duración estimada en puntos (semanas) y el orden de implementación.

Tabla 11: Plan de Iteraciones.

Iteración	Historias de usuario		Duración (puntos)
1	HU 1	Gestionar rasgos	1
	HU 2	Gestionar bases de datos	1
	HU 3	Gestionar objetos	2
2	HU 4	Administrar grupos	3
3	HU 5	Gestionar usuario	1
	HU 6	Autenticar usuario	1
Total			**9**

La fase de la realización de las iteraciones es primordial en el ciclo de desarrollo de XP, la cual incluye varias iteraciones en el sistema previos a su entrega final. En esta fase se desarrollan las funcionalidades entregables y junto con la retroalimentación conjunta con el cliente, se da la terminación exitosa del proyecto(52).

1ra Iteración:

Tiene como objetivo darle cumplimiento a las historias de usuario con mayor prioridad para el cliente en cuanto al negocio, las cuales serían las historias de usuario 1, 2 y 3.

2da Iteración:

Tiene como objetivo darle cumplimiento a las historias de usuario con prioridad media que terminan las funcionalidades de la iteración anterior, y es la historia de usuario 4.

3ra Iteración:

Las historias de usuario 5 y 6 tienen como objetivo el conceder los permisos para el acceso a nuestro sistema y establecer los usuarios para ello. Al concluir Esta iteración, se concluirá el proyecto.

2.2.6 Plan de entregas

El cronograma de entregas (“Release Plan”) establece qué historias de usuario serán agrupadas para conformar una entrega, y el orden de las mismas. Este cronograma será el resultado de una reunión entre todos los actores del proyecto (cliente, desarrolladores, gerentes, etc.). XP denomina a esta

reunión “Juego de planeamiento” (“Planning game”), pero puede denominarse de la manera que sea más apropiada al tipo de empresa y el cliente(51).

Tabla 12: Plan de entregas.

Iteraciones (Entregable)	Fecha de inicio	Fecha de fin
Iteración 1	06/06/2022	01/07/2022
Iteración 2	05/09/2022	26/09/2022
Iteración 3	10/10/2022	01/11/2022

2.2.7 Vista de presentación

En el presente epígrafe se hará un análisis sobre la vista de presentación de nuestro sistema informático, que incluye el mapa de navegación, que nos indica la estructura interna de las vistas, y la descripción textual y visual de los contenidos de las pantallas general y por roles.

Mapa de navegación

El Mapa de navegación contiene la jerarquía del sistema web, con las pantallas y los requisitos funcionales principales.

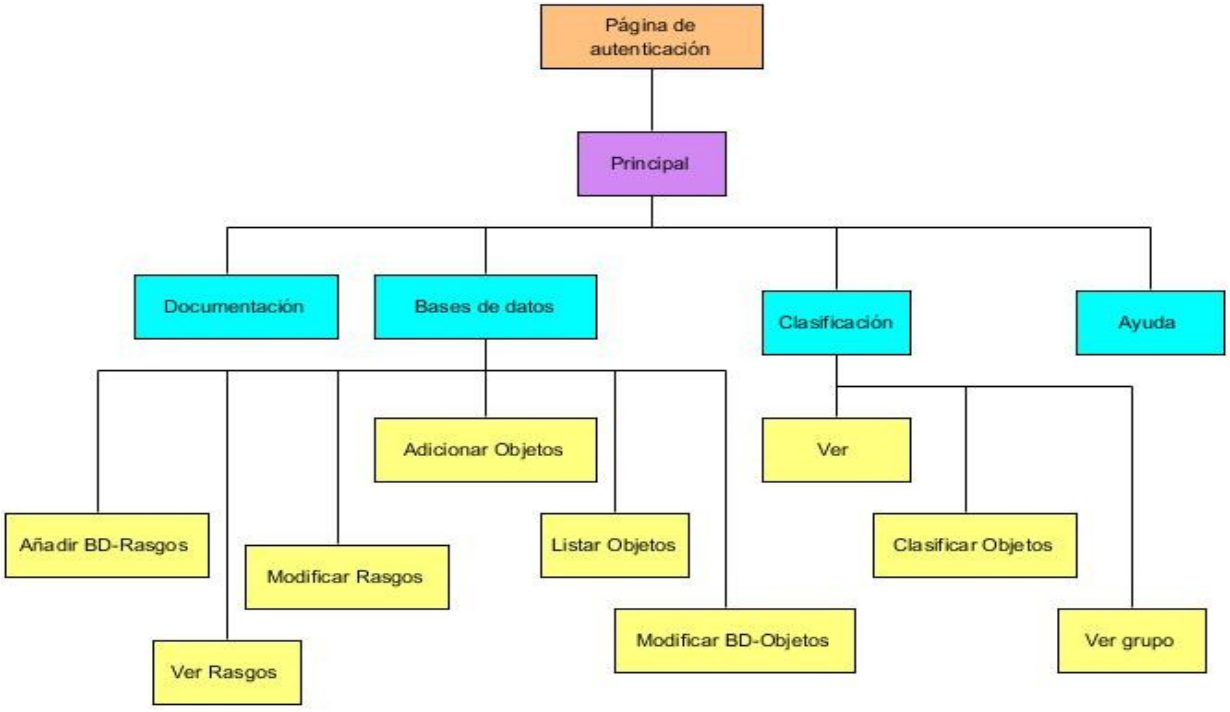


Ilustración 2: Mapa de navegación USGest. Fuente: Elaboración propia

Definición de contenidos de la página principal

Ilustración 3: Pantalla de inicio Fuente: Elaboración propia

Descripción de los contenidos a mostrar en página principal

En la parte superior de la pantalla de inicio se mostrará en la barra de navegación (común para todas las vistas) el logotipo, el usuario activo y el nombre del sistema en la izquierda y en la derecha, los elementos de navegación dentro del propio programa, como son los enlaces a: la página Inicio, la página Documentación, la página relacionada con las Bases de datos, la página Clasificación relacionada con la formación de grupos y la página Ayuda que nos brinda información sobre cómo trabajar con el sistema.

Debajo de la barra de navegación se muestra la ruta en donde se encuentra el usuario a medida que navega por el sistema (común para todas las vistas).

En la parte posterior de la pantalla de inicio se mostrará en el pie de página (común para todas las vistas) el nombre del autor del sistema, la facultad y el grupo al que pertenece y el centro de estudios.

Definición y descripción de los contenidos a mostrar por las pantallas

Pantalla Documentación:

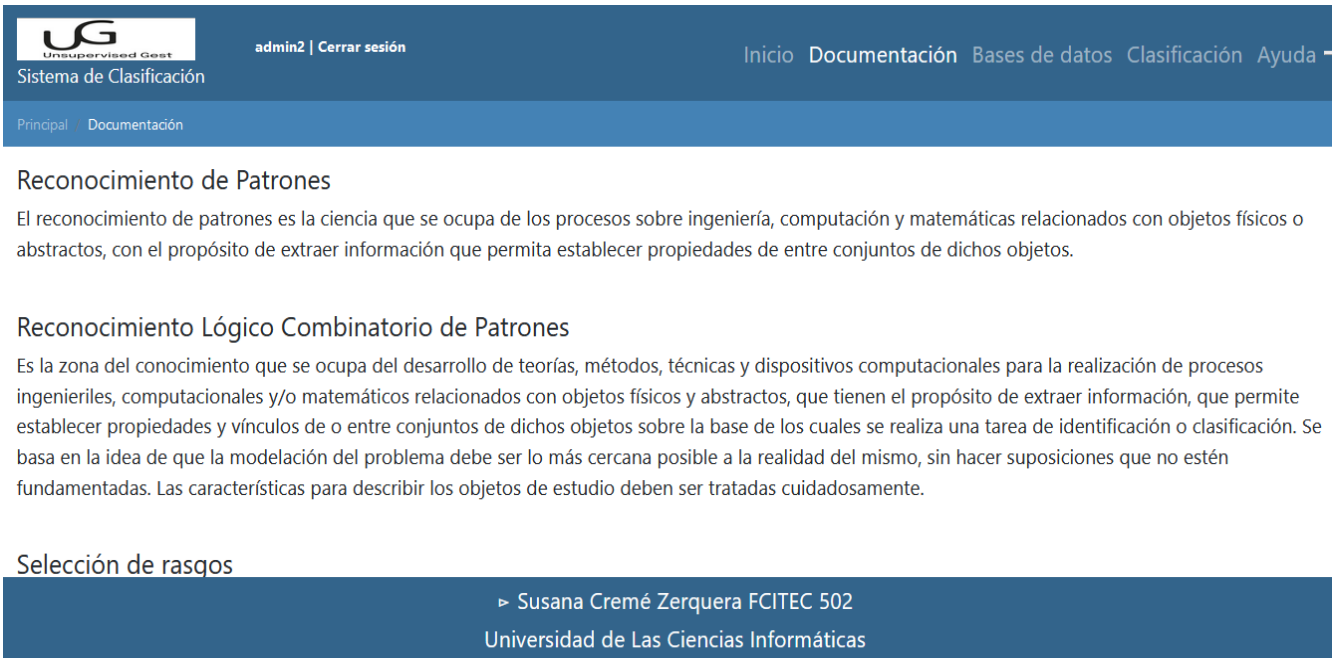


Ilustración 4: Pantalla Documentación Fuente: Elaboración propia

Descripción de los contenidos a mostrar en la pantalla Documentación

Esta vista muestra los conceptos fundamentales y los procedimientos relacionados con los contenidos sobre reconocimiento de patrones, clasificación no supervisada y los distintos algoritmos.

Pantalla Bases de datos:

Ilustración 5: Pantalla Bases de datos Fuente: Elaboración propia

Descripción de los contenidos a mostrar en la pantalla Bases de datos

En esta vista se realiza la gestión y se muestran los listados de las bases de datos de objetos y rasgos.

Pantalla Clasificación:

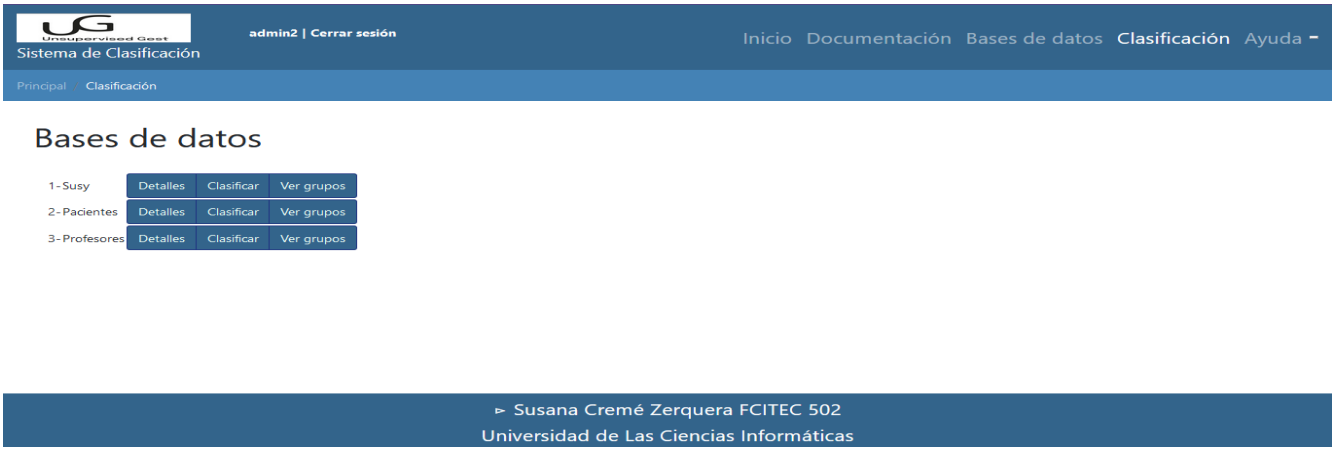


Ilustración 6: Pantalla Clasificación Fuente: Elaboración propia

Descripción de los contenidos a mostrar en la pantalla Clasificación

En la Vista Clasificación se muestran las bases de datos existentes, se selecciona la deseada para la clasificación y se pasa a ver los grupos.

Pantalla Ayuda:

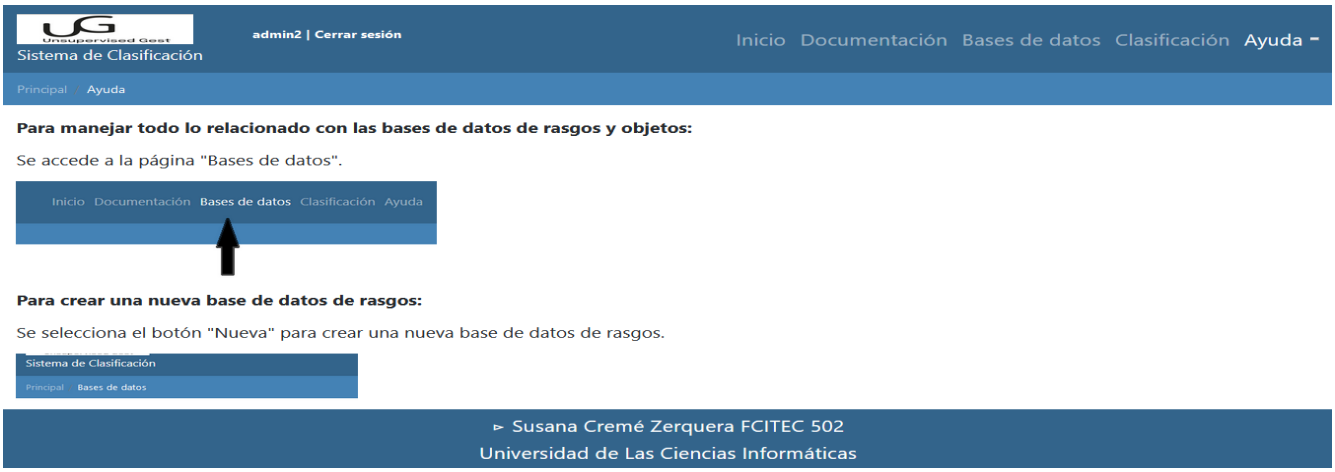


Ilustración 7: Pantalla Ayuda Fuente: Elaboración propia

Descripción de los contenidos a mostrar en la pantalla Ayuda

En la vista Ayuda, se ofrece un manual de usuario donde se explica de forma sencilla y detallada el funcionamiento de la aplicación.

2.2.8 Diagrama de actividades

Los diagramas de actividades permiten establecer la secuencia de las actividades involucradas en un proceso o sistema e identificar sus responsables(53). Para la comprensión del funcionamiento del sistema, se presenta el diagrama de actividades donde se aprecian las acciones que realizará el usuario y la respectiva respuesta del sistema.

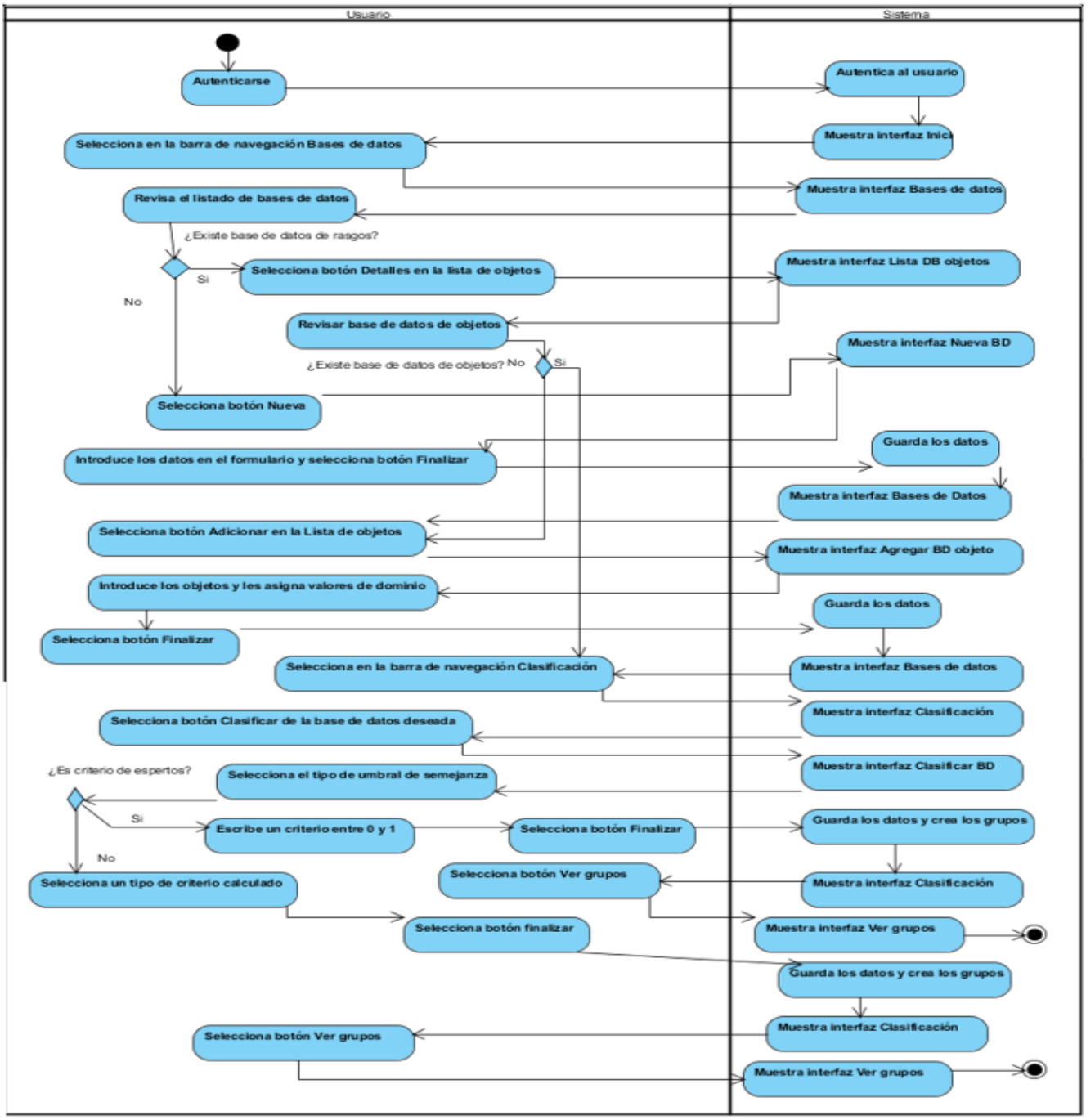


Ilustración 8: Diagrama de actividades. Fuente: Elaboración propia

2.3 Diseño e implementación del almacenamiento, procesamiento y flujo de información en el sistema web USGest

Este sistema se diseñó con el objetivo de facilitar el trabajo con algoritmos de clasificación no supervisada utilizando el Reconocimiento Lógico Combinatorio de Patrones en apoyo a la docencia.

En el presente epígrafe se presentará el diseño de los mecanismos para el almacenamiento, procesamiento y transmisión de los datos en el sistema propuesto USGest, también se brindarán ejemplos de estos mismos mecanismos implementados, así como algunas de las interfaces graficas de usuario correspondientes.

2.3.1 Arquitectura de software

Una arquitectura de software de un programa o sistema computacional es la estructura del sistema, la cual comprende elementos de software, las propiedades externamente visibles de esos elementos, y las relaciones entre ellos(54).

Django sigue el patrón de arquitectura de software Modelo-Vista-Controlador (MVC), donde el “Modelo” es la porción de acceso a la base de datos, es manejada por la capa de la base de datos de Django, la “Vista” es la porción que selecciona qué datos mostrar y cómo mostrarlos, es manejada por la vista y las plantillas, y el “Controlador” es la porción que delega a la vista dependiendo de la entrada del usuario, es manejada por el framework mismo siguiendo la URLconf y llamando a la función apropiada de Python para la URL obtenida(55).

Debido a que el “Controlador” es manejado por el mismo framework y la parte más emocionante se produce en los modelos, las plantillas y las vistas, Django es conocido como un Framework MTV.

En el patrón de diseño MVT:

- M significa *Model* (Modelo): la capa de acceso a la base de datos. Esta capa contiene toda la información sobre los datos: cómo acceder a estos, cómo validarlos, cuál es el comportamiento que tiene, y las relaciones entre los datos(55). El modelo se encarga de guardar los datos de manera persistente mediante una base de datos. Contiene la lógica del negocio y los datos que se utilizarán(56). Son objetos Python que definen la estructura de los datos de una aplicación y proporcionan mecanismos para gestionar y consultar registros de la base de datos(57).
- T significa *Templates* (Plantillas): la capa de presentación. Esta capa contiene las decisiones relacionadas a la presentación: cómo algunas cosas son mostradas sobre una página web u otro tipo de documento(55). Las plantillas son ficheros de texto que definen la estructura o diagrama de otro fichero. Se puede crear dinámicamente una página usando una plantilla, rellenándola con datos de un modelo(57).
- V significa *View* (Vista): la capa de la lógica de negocios. Esta capa contiene la lógica que accede al modelo y la delega a la plantilla apropiada, como un puente entre modelos y las plantillas(55). La vista se encarga de mostrar a los usuarios finales los datos mediante la interfaz ya predefinida

por los programadores del software(56). Es una función de gestión de peticiones que recibe peticiones HTTP y devuelve respuestas HTTP, acceden a los datos que necesitan y delegan el formateo de la respuesta a las plantillas(57).

Diagrama de paquetes

Un diagrama de clases es usado para ilustrar modelos de clases. Un diagrama de clases puede decir rápidamente que tipos de abstracciones se hallan presentes en el sistema y en donde se encuentran las partes cuestionables que necesitan más trabajo(58).

Un diagrama de paquetes es un diagrama que muestra los paquetes de clases y las dependencias entre ellos. Los paquetes y las dependencias son elementos de un diagrama de clases, por lo cual un diagrama de paquetes un una forma de diagrama de clases(58).

A continuación, se representa mediante el diagrama de paquetes agrupando elementos relacionados semánticamente y a su vez mostrando las dependencias entre esas agrupaciones la arquitectura de software MVC del sistema web USGest:

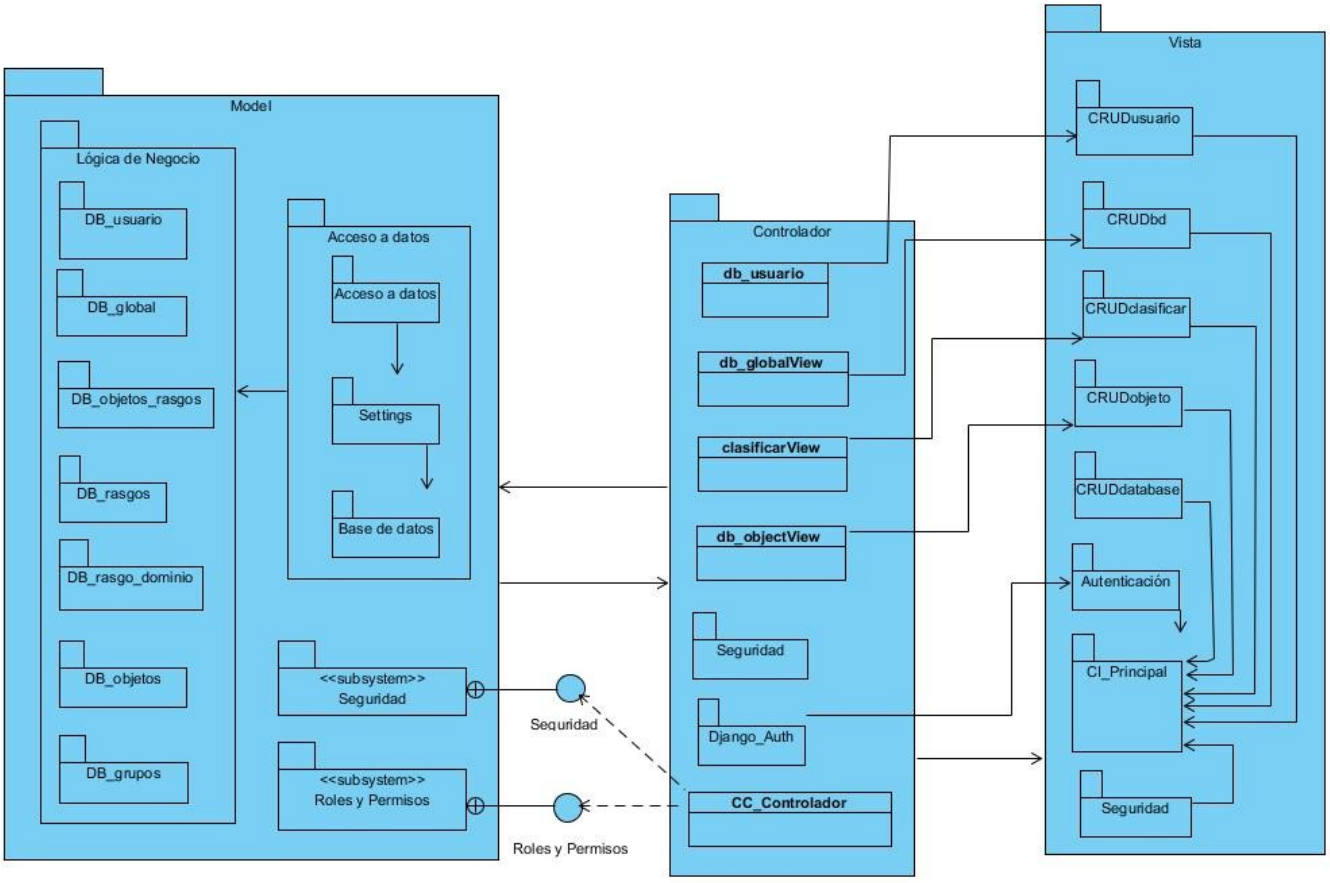


Ilustración 9: Diagrama de paquetes Fuente: Elaboración prueba

2.3.2 Descripción de tarjetas CRC

Las tarjetas CRC (Clase-Responsabilidad-Colaborador) fueron presentadas para la Programación Orientada a Objetos. Sus características más sobresalientes son su simpleza y ductilidad ya que representan a una entidad del sistema a las cuales asigna responsabilidades y colaboraciones(59).

Tabla 13: Tarjeta CRC de la clase db_global

Nombre de la clase: db_global	
Responsabilidades	**Colaboradores**
detallesBDrasgos() listarDB() crearBD() editarBD() eliminarBD()	DB_global DB_globalForm DB_globalService

Tabla 14: Tarjeta CRC de la clase db_objects

Nombre de la clase: db_objects	
Responsabilidades	**Colaboradores**
listarBDobjetos() agregarBDobjetos() modificarBDobjetos()	DB_global DB_ObjetosService

Tabla 15: Tarjeta CRC de la clase clasificar

Nombre de la clase: clasificar	
Responsabilidades	**Colaboradores**
clasificacion() clasificarBD() vergrupo()	DB_GlobalService DB_ObjetosService

2.3.3 Patrones de Diseño

Patrones GRASP:

Los patrones GRASP (General Responsibility Assignment Software Patterns) o patrones de asignación de responsabilidades, son una serie de buenas prácticas enfocadas a la calidad estructural del software. GRASP es considerado como una serie de buenas prácticas o consejos que hay que seguir para hacer las cosas bien(60).

Se basan en la determinación de las clases adecuadas y decidir cómo estas clases deben interactuar. Incluso cuando se utilizan metodologías rápidas como XP y el proceso se centra en el desarrollo continuo, es necesario elegir cuidadosamente las responsabilidades de cada clase desde la primera codificación y, fundamentalmente, en la refactorización del programa(60). Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones:

GRASP son 7 patrones o consejos, de los cuales se emplearon 4 en la realización del software.

- **Alta Cohesión:** El grado de cohesión mide la coherencia de la clase, esto es, lo coherente que es la información que almacena una clase con las responsabilidades y relaciones que ésta tiene(61). Se utilizó en db_global.py
- **Bajo acoplamiento:** El grado de acoplamiento indica lo vinculadas que están unas clases con otras, lo que afecta un cambio en una clase a las demás y por tanto lo dependientes que son unas clases de otras(61). Se utilizó en db_global.py

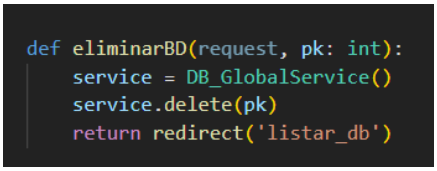


Ilustración 10: Código Patrón Alta cohesión - Bajo acoplamiento. Fuente: Elaboración propia

- **Controlador:** El patrón controlador es un patrón que sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es el controlador quien recibe los datos del usuario y quien los envía a las distintas clases según el método llamado. Este patrón sugiere que la lógica de negocio debe estar separada de la capa de presentación, lo que aumenta la reutilización de código y permite a la vez tener un mayor control. Hay muchas arquitecturas de aplicación que se basan en ésto, desde el famoso MVC que ya vi estudiando la carrera hasta la arquitectura MVVM tan utilizada últimamente en aplicaciones de escritorio y la última que he practicado profesionalmente, MVVMP, que no es más que un refinamiento sutil de la anterior(62). Se utilizó en db_global.py

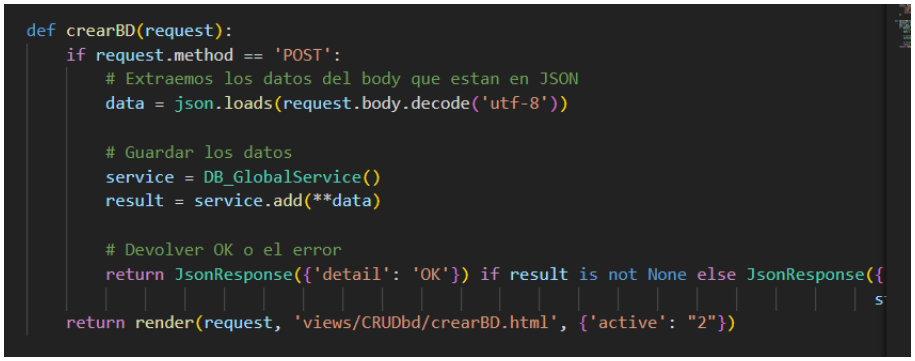


Ilustración 11: Código Patrón Controlador. Fuente: Elaboración propia

- **Creador:** El patrón creador nos ayuda a identificar quién debe ser el responsable de la creación o instanciación de nuevos objetos o clases. Éste patrón nos dice que la nueva instancia podrá ser creada por una clase si: Contiene o agrega la clase. Almacena la instancia en algún sitio (por

ejemplo una base de datos) Tiene la información necesaria para realizar la creación del objeto (es 'Experta') Usa directamente las instancias creadas del objeto(63). Se utilizó en service.py

```python
class DB_GlobalService:
    def add(self, nombre: str, rasgos: List[dict]) -> Optional[DB_global]:
        # Validate data
        if not Validator.db_is_valid({'nombre': nombre, 'rasgos': rasgos}):
            return None

        # Agregar modelo DB Global
        model = DB_global()
        model.nombre = nombre
        model.save()

        # Agregar modelo rasgos
        for rasgo in rasgos:
            rasgo_model = DB_rasgos()
            rasgo_model.nombre = rasgo.get('nombre')
            rasgo_model.criterio = rasgo.get('criterio')
            rasgo_model.db_global_id = model
            rasgo_model.save()
```

Ilustración 12: Código Patrón Creador. Fuente: Elaboración propia

2.3.4 Modelo de Datos

Una base de datos una compilación de datos persistentes en el tiempo, los cuales se pueden compartir e interrelacionar(64). Sus principales características son la persistencia e interrelación. Las bases de datos relacionales son unos de los distintos tipos en que se pueden dividir, las cuales son un repositorio donde se comparten datos, creando relaciones entre tablas y asignando un nombre único a cada tabla(40).

A continuación se presenta el modelo de datos relacional de la propuesta del sistema USGest.

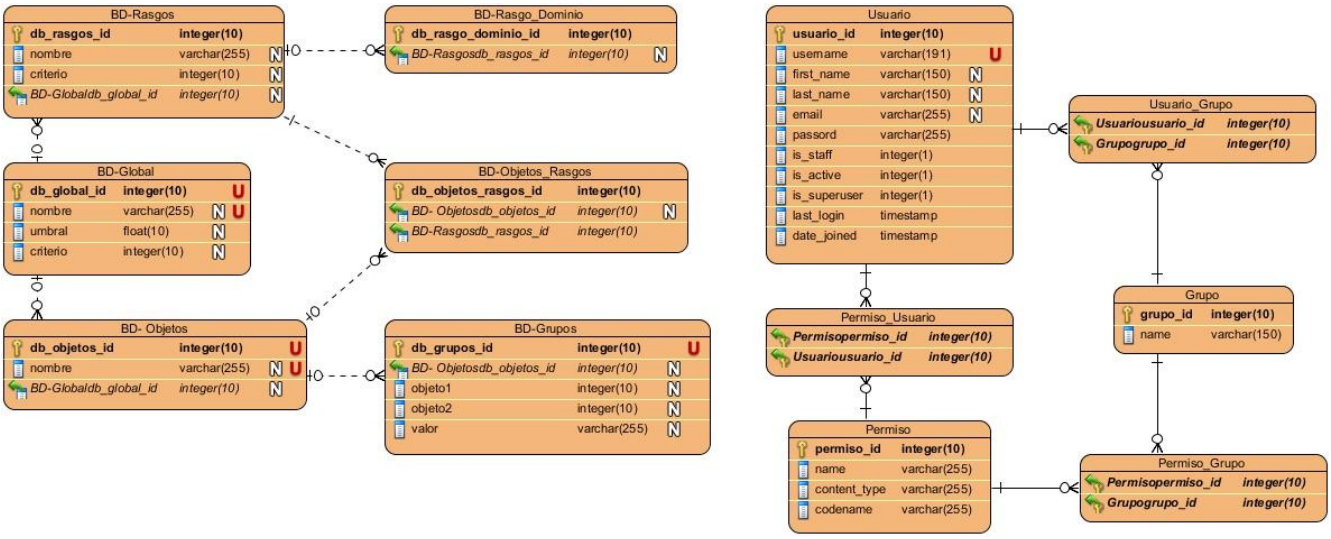


Ilustración 13: Diagrama Entidad-Relación Fuente: Elaboración propia

Conclusiones del capítulo

En este capítulo se analizó la propuesta del sistema con algunas de sus especificaciones y funcionalidades. Se detallaron todos los requisitos tanto funcionales como no funcionales necesarios para el correcto funcionamiento del sistema, se describieron las historias de usuarios y algunos de los prototipos a utilizar, se estableció un cronograma de trabajo a partir del plan de iteraciones y el plan de entregas.

Se estableció la jerarquía del sistema mediante el mapa de navegación y se mostraron los contenidos de la pantalla de inicio comunes en todas las vistas.

Se construyó la arquitectura del software mediante el patrón arquitectónico MVC y las tarjetas CRC para conocer la estructura interna de las clases y los métodos involucrados.

Se definieron los patrones de diseño GRASP utilizados, y se representó el modelo de datos relacional para conocer la estructura de la base de datos.

CAPÍTULO III: IMPLEMENTACIÓN Y PRUEBAS DEL SISTEMA WEB USGEST PARA LA CLASIFICACIÓN NO SUPERVISADA UTILIZANDO EL RECONOCIMIENTO LÓGICO COMBINATORIO DE PATRONES

En el presente capítulo se presentan las determinaciones sobre la implementación del sistema. Se desglosan las tareas de ingeniería y se analiza el diagrama de despliegue. Se exponen además las pruebas a las que es sometido el sistema web propuesto para la verificación del cumplimiento de los requisitos previamente descritos.

3.1 Fase de desarrollo

En la fase de desarrollo del software se ejecutan las tareas de la ingeniería relacionadas con la implementación de cada una de las historias de usuario de cada iteración antes desglosadas.

3.1.1 Tareas de Ingeniería

Las **tareas de ingeniería** del software son las pautas a seguir descritas por los analistas, las cuales serán la guía para los programadores en el momento de la implementación del software(65).
A continuación, se muestran las distintas tareas asociadas con cada historia de usuario en cada una de las iteraciones y una breve descripción de las mismas.

Tabla 16: Tareas de Ingeniería de la iteración 1

Historias de Usuario	**Tareas de Ingeniería**
1. Gestionar rasgos	1. Insertar rasgos 2. Modificar rasgos 3. Eliminar rasgos 4. Listar rasgos
2. Gestionar bases de datos	1. Insertar bases de datos 2. Modificar bases de datos 3. Eliminar bases de datos 4. Listar bases de datos 5. Ver bases de datos
3. Gestionar objetos	1. Insertar objetos 2. Modificar objetos 3. Eliminar objetos 4. Listar objetos 5. Ver objetos

Tabla 17: Tarea de Ingeniería 1 Insertar rasgos

Tarea	
Número de tarea: 1	**Número de la historia de usuario:** 1
Nombre de la tarea: Insertar rasgos	
Tipo de tarea: Desarrollo	**Puntos estimados:**

Fecha de inicio: 06/06/2022	**Fecha de fin:** 10/06/2022
Descripción: El usuario puede insertar un rasgo	

Tabla 18: Tarea de ingeniería 2 Modificar rasgo

Tarea	
Número de tarea: 2	**Número de la historia de usuario:** 1
Nombre de la tarea: Modificar rasgo	
Tipo de tarea: Desarrollo	**Puntos estimados:**
Fecha de inicio: 06/06/2022	**Fecha de fin:** 10/06/2022
Descripción: El usuario puede modificar un rasgo en caso de que exista	

Tareas de ingeniería para la primera iteración ***(Ver Anexo 2)***

Tabla 19: Tareas de Ingeniería de la iteración 2

Historias de Usuario	**Tareas de Ingeniería**
4. Administrar grupos	1. Crear grupos 2. Ver grupos

Tabla 20: Tarea de ingeniería 15 Crear grupos

Tarea	
Número de tarea: 15	**Número de la historia de usuario:** 4
Nombre de la tarea: Crear grupos	
Tipo de tarea: Desarrollo	**Puntos estimados:**
Fecha de inicio: 05/09/2022	**Fecha de fin:** 23/09/2022
Descripción: El usuario puede crear grupos	

Tareas de ingeniería para la segunda iteración ***(Ver Anexo 2)***

Tabla 21: Tareas de Ingeniería de la iteración 3

Historias de Usuario	**Tareas de Ingeniería**
5. Gestionar usuario	1. Insertar usuario 2. Modificar usuario 3. Eliminar usuario 4. Listar usuario
6. Autenticar usuario	1. Autenticar usuario 2. Cerrar sesión

Tabla 22: Tarea de ingeniería 18 Insertar usuario

Tarea	
Número de tarea: 18	**Número de la historia de usuario:** 5
Nombre de la tarea: Insertar usuario	
Tipo de tarea: Desarrollo	**Puntos estimados:**
Fecha de inicio: 10/10/2022	**Fecha de fin:** 01/11/2022
Descripción: El administrador puede insertar los usuarios del sistema	

Tareas de ingeniería para la tercera iteración ***(Ver Anexo 2)***

Tabla 23: Tarea de ingeniería 22 Autenticar usuario

Tarea	
Número de tarea: 22	**Número de la historia de usuario:** 6
Nombre de la tarea: Autenticar usuario	
Tipo de tarea:	**Puntos estimados:**
Fecha de inicio: 24/10/2022	**Fecha de fin: 01**/11/2022
Descripción: El usuario puede autenticarse en el sistema	

Tabla 24: Tarea de ingeniería 23 Cerrar sesión

Tarea	
Número de tarea: 23	**Número de la historia de usuario:** 6
Nombre de la tarea: Cerrar sesión	
Tipo de tarea: Desarrollo	**Puntos estimados:**
Fecha de inicio: 24/10/2022	**Fecha de fin:** 01/11/2022
Descripción: El usuario puede cerrar su sesión en el sistema	

3.1.2 Despliegue del sistema web USGest

El **diagrama de despliegue** es un diagrama estructurado que muestra la arquitectura del sistema desde el punto de vista del despliegue (distribución) de los artefactos del software en los destinos de despliegue. Los artefactos representan elementos concretos en el mundo físico que son el resultado de un proceso de desarrollo. Ejemplos de artefactos son archivos ejecutables, bibliotecas, archivos, esquemas de bases de datos, archivos de configuración entre otros (Descripción, 2014).

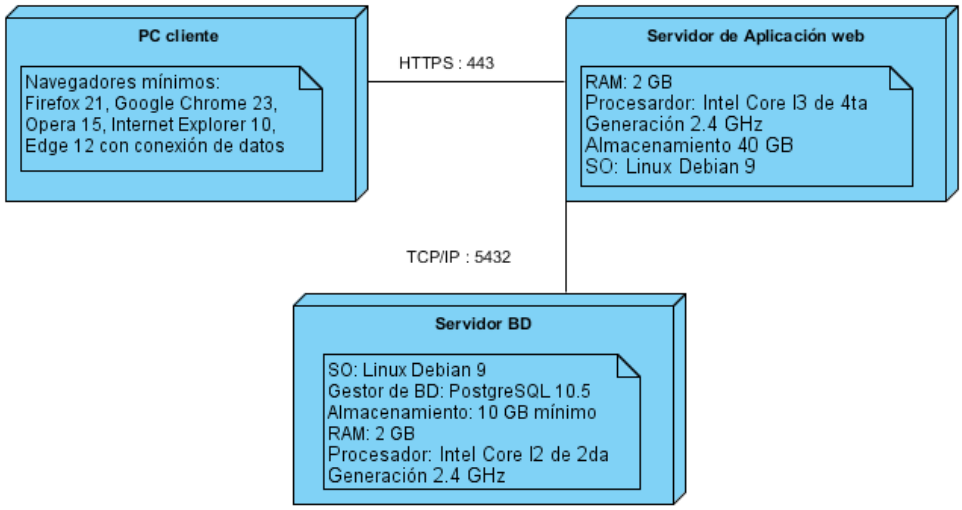


Ilustración 14: Diagrama de despliegue Fuente: Elaboración propia

3.2 Verificación y validación del sistema web USGest

La Verificación y Validación (V&V) Conjunto de procesos de comprobación y análisis que aseguran

que el software que se desarrolla está acorde a su especificación y cumple las necesidades de los clientes(26).

La verificación se refiere a si se está construyendo correctamente el software o no, por tanto, sus funciones son(66):

- Comprobar que el software está de acuerdo a su especificación.
- Cumplimiento de los requisitos funcionales y no funcionales según su especificación.

La validación se refiere a si se está construyendo el producto correcto o no, por tanto, su objetivo es demostrar que el software hace lo que el cliente espera que haga para cumplir con sus expectativas(66).

La verificación y la validación poseen muchas actividades de aseguramiento de la calidad como son:

- Revisiones técnicas.
- Auditorías de calidad y configuración.
- Monitoreo de rendimiento.
- Simulación.
- Revisión de base de datos.
- Revisión de documentos.
- Pruebas de desarrollo, usabilidad, aceptación, instalación, entre otras.

Las técnicas de V&V son las inspecciones del software y las pruebas del software, las cuales se recogen en la estrategia de prueba confeccionada.

3.2.1 Estrategia de Prueba:

Una estrategia de prueba de software integra las técnicas de diseño de casos de prueba en una serie de pasos bien planificados que dan como resultado una correcta construcción del software. La estrategia proporciona un mapa que describe los pasos que hay que llevar acabo como parte de la prueba, cuándo se deben planificar y realizar esos pasos y cuánto esfuerzo, tiempo y recursos se van a requerir. Cualquier estrategia de prueba debe incorporar la planificación de la prueba, el diseño de casos de prueba, la ejecución de las pruebas; y la agrupación y evaluación de los datos resultantes(67).

Las características generales de las estrategias de prueba de software son las siguientes(68):

1. La prueba comienza en el nivel módulo y trabaja “hacia fuera”, hacia la integración de todo el sistema basado en computadora.
2. Diferentes técnicas de prueba son apropiadas en diferentes momentos.
3. La prueba la realiza el que desarrolla el software y un grupo de prueba independiente.

4. La prueba y la depuración son actividades, pero la depuración se puede incluir en cualquier estrategia de prueba.

A continuación, se muestra un fragmento de la Estrategia de Prueba llevada a cabo para la validación y verificación del sistema web para la clasificación no supervisada utilizando el Reconocimiento Lógico Combinatorio de Patrones:

Tabla 25: Tipos y niveles de prueba

Etapa	**Tipo/Nivel de Prueba**	Objetivo
Vista de Presentación	Usabilidad	Validar la aceptación del diseño de las vistas del sistema por parte del cliente.
Vista Lógica	Inspección del software	Verificar los artefactos generados durante etapa de la Vista Lógica.
Vista de Implementación	Unidad	Verificar la organización y ejecución estática del sistema en su ambiente de desarrollo.
Vista de Despliegue	Seguridad, Partición de equivalencia Usabilidad, Aceptación	Validar el sistema en un ambiente de ejecución del cliente.

Las inspecciones del software, son las técnicas estáticas de V&V, ya que no se necesita ejecutar el software y además pueden usarse en todas las etapas del proceso de desarrollo. Para ellas se utilizan las **listas de chequeo**, que consisten en un formulario de preguntas, las cuales dependen del objetivo para el cual son usadas. Son fáciles de aplicar, resumir y comparar.

Parámetros dentro de la lista de chequeo y su significado:

Evaluación: Es la forma de evaluar el indicador en cuestión. El mismo se evalúa de 1 en caso de mal (cuando la respuesta al indicador sea “No”) y 0 en caso que elemento revisado no presente errores (cuando la respuesta al indicador sea “Sí”).

NP (No Procede): Se usa para especificar que el indicador a evaluar no se puede aplicar en ese caso.

Observación: Especifica los señalamientos o sugerencias que quiera incluir la persona que aplica la lista de chequeo.

“Las pruebas del software son un elemento crítico para la garantía de calidad del software y representa una revisión final de las especificaciones, del diseño y de la codificación”(26).

La prueba es un proceso que se enfoca sobre la lógica interna del software y las funciones externas. Es un proceso de ejecución de un programa con la intención de descubrir un error, no puede asegurar la ausencia de defectos; sólo puede demostrar que existen defectos en el software(66).

De acuerdo con la metodología previamente seleccionada XP, se realizarán las pruebas en los niveles de unidad y aceptación. Estos niveles, están compuestos por un conjunto de tipos, métodos y técnicas de pruebas, las cuales se evidenciarán a continuación en cada etapa del desarrollo del sistema web.

Etapa: Vista de presentación

Usabilidad: Prueba enfocada a factores humanos, estéticos, consistencia en la interfaz de usuario, ayuda sensitiva al contexto y en línea, asistente documentación de usuarios y materiales de entrenamiento, lo cual se centra en la habilidad del software para satisfacer al usuario(26).

A continuación, se presenta la lista de chequeo correspondiente a la técnica de Inspección del software para el sistema web en la etapa de Vista de Presentación.

Tabla 26: Lista de chequeo del sistema web USGest. Vista de Presentación

Lista de chequeo del sistema web USGest				
No.	**Indicador a evaluar**	**Evaluación**	**NP**	**Observación**
1	¿Está visible el nombre de la página?	**0**		
2	¿Se muestra la información en orden, o sea primero las más recientes?	**0**		
3	¿Tiene el contenido principal dentro de la primera página?	**1**		
4	¿Posee enlaces para acceder a otras informaciones propuestas?	**0**		
5	¿Están separados los contenidos de acuerdo a su tema?	**0**		
6	¿Es monótona?	**0**		
7	¿El nombre de la página tiene correspondencia con el tema de la misma y es intuitivo?	**0**		
8	¿Es visible el texto de la página?	**0**		
9	¿Hay una distancia aceptable entre enlaces e iconos evitando fallar al navegar?	**0**		
10	¿El contraste de colores entre fondo y contenido es aceptable?	**0**		
11	¿Es adaptable a otros dispositivos o dimensiones?	**0**		

Etapa: Vista lógica

A continuación, se presentan las listas de chequeo correspondientes a la técnica de Inspección del software para el sistema web en la etapa de Vista Lógica.

Tabla 27: Lista de chequeo para evaluar la historia de usuario: Gestionar rasgo. Vista Lógica

Lista de chequeo para evaluar la Historia de usuario: Gestionar rasgo				
No.	**Indicador a evaluar**	**Evaluación**	**NP**	**Observación**

1	¿Está en infinitivo y refleja de manera clara el objetivo del usuario sobre el sistema?	**0**		
2	¿El nombre de la historia de usuario es único?	**0**		
3	¿El nombre de la historia de usuario es intuitivo?	**0**		
4	¿El nombre del caso de uso es explicativo?	**0**		
5	¿Recoge las acciones fundamentales del CU?	**0**		

Etapa: Vista de implementación

Las pruebas **unitarias** comprueban el funcionamiento de manera aislada de piezas de software que pueden ser probadas de forma separada. Dependiendo del contexto, estas piezas de software pueden ser una función, un método, una clase, un módulo o, en general, una “unidad” individual dentro del diseño de un sistema software que puede ser probada de forma independiente. Estas pruebas son realizadas siempre por los programadores(69).

Las pruebas de tipo Función: fijan su atención en la validación de las funciones, métodos, servicios, casos de uso(26).

Las pruebas unitarias de función presentadas a continuación se realizaron al modelo y se probaron los métodos de adicionar, modificar y eliminar rasgos y objetos.

```
#Prueba unitaria
class DBglobal_TestCase(TestCase):
    def test_pruebaGlobal_is_resolved(self):
        modelo = DB_global()
        modelo.nombre = 'paciente'
        modelo.criterio = '1'
        modelo.calculado = '1'
        modelo.umbral = '0.4'
        modelo.created = True
        modelo.save()

        exists = DB_global.objects.filter(nombre='paciente').exists()
        self.assertEquals(exists, True)

        modelo.nombre = 'sexo'
        modelo.save()
        yes_exists = DB_global.objects.filter(nombre='sexo').exists()
        self.assertEquals(yes_exists, True)

        modelo.delete()

        non_exists = DB_global.objects.filter(nombre='sexo').exists()
        self.assertEquals(non_exists, False)
```

Ilustración 15: Prueba unitaria al modelo DB_global

```python
def test_pruebaRasgos_is_resolved(self):
    modelo = DB_global()
    modelo.nombre = 'paciente'
    modelo.criterio = '1'
    modelo.calculado = '1'
    modelo.umbral = '0.4'
    modelo.created = True
    modelo.save()
    rasgos = DB_rasgos()
    rasgos.nombre = 'sexo'
    rasgos.criterio = '2'
    rasgos.db_global_id = modelo
    rasgos.save()

    exists = DB_rasgos.objects.filter(nombre='sexo').exists()
    self.assertEquals(exists, True)

    rasgos.nombre = 'tiempo'
    rasgos.save()
    yes_exists = DB_rasgos.objects.filter(nombre='tiempo').exists()
    self.assertEquals(yes_exists, True)

    rasgos.delete()
    non_exists = DB_rasgos.objects.filter(nombre='tiempo').exists()
    self.assertEquals(non_exists, False)
```

Ilustración 16: Prueba unitaria al modelo DB_rasgos

Además, se realizaron pruebas unitarias a las urls del sistema, para comprobar la correcta función de la navegabilidad dentro del mismo.

```python
class TestUrls(SimpleTestCase):

    def test_home_url_is_resolved(self):
        url = reverse('home')
        self.assertEquals(resolve(url).func, index)

    def test_ayuda_url_is_resolved(self):
        url = reverse('ayuda')
        self.assertEquals(resolve(url).func, ayuda)

    def test_documentacion_url_is_resolved(self):
        url = reverse('documentacion')
        self.assertEquals(resolve(url).func, documentacion)
```

Ilustración 17: Pruebas unitarias de función a las urls

En las pruebas realizadas se obtuvieron resultados satisfactorios, por tanto, no se encontraron errores en la lógica ni el direccionamiento del sistema.

```
(venv) H:\!Tesis\clasificacion>py manage.py test system
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
.....
----------------------------------------------------------------------
Ran 5 tests in 0.159s

OK
Destroying test database for alias 'default'...

(venv) H:\!Tesis\clasificacion>
```

Ilustración 18: Resultado de pruebas unitarias

Etapa: Vista de despliegue

Seguridad:

Seguridad: Asegurar que los datos o el sistema solamente es accedido por los actores deseados(26).
El sistema cuenta con una autenticación de usuario de Django, el cual garantiza que sólo los usuarios autenticados al sistema podrán acceder a sus datos, y además posee un administrador de usuario el cual va a manejar los roles y los permisos de los mismos.

Las pruebas de **aceptación** se encargan de comprobar si el software completado cumple las necesidades por las que dicho software ha sido creado. Estas actividades de pruebas pueden o no involucrar a los desarrolladores del sistema, pero siempre involucrarán al cliente o usuarios(69).
Tipos de pruebas de aceptación:
Las pruebas con alpha o beta testers, cuyo objetivo es que un grupo representativo de usuarios utilice el sistema antes de realizar su paso a la etapa en producción, y las utilizadas fueron las Beta ya que son ejecutadas desde la infraestructura del cliente y por este(26).

Un **Casos de prueba** es un conjunto de acciones con resultados y salidas previstas basadas en los requisitos de especificación del sistema; sus componentes son(70):
1. Propósito: de la prueba o descripción del requisito que se está probando
2. Método: o forma como se probará
3. Versión: o configuración de la prueba, versión de la aplicación en prueba, el hardware, el software, el sistema operativo, los archivos de datos, entre otros
4. Resultados: acciones y resultados esperados o entradas y salidas
5. Documentación: de la prueba y sus anexos.

Los casos de pruebas nos permiten encontrar funciones incorrectas o ausentes, errores de interfaz,

errores en estructuras de datos o en accesos a las bases de datos externas, errores de rendimiento y errores de inicialización y terminación.

Las **Pruebas de caja negra** se llevan a cabo sobre la interfaz del software y se centran en los requisitos no funcionales(71). Su objetivo es demostrar que las funciones del software son operativas, que las entradas se aceptan de forma adecuada y se produce un resultado correcto, y que la integridad de la información externa se mantiene)(26).

Se utilizó la **técnica de caja negra de partición de equivalencia** la cual nos permite examinar los valores válidos e inválidos de las entradas existentes en el software y descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico.

A continuación, se muestran los casos de prueba analizados donde se utilizó la técnica de partición de equivalencia para el método de prueba de caja negra.

Tabla 28: Caso de prueba 1 de la Historia de usuario Gestionar rasgos. Escenario Adicionar rasgo

Id	Escenario	Descripción	Campos			Respuesta del sistema	Resultado de la prueba
			Rasgo	Criterio	Dominio		
1.1	Entrada de datos válidos	Permite adicionar un rasgo	V Edad	V Cuantitativo	V 23	Se añadió satisfactoriamente un rasgo. Muestra el mensaje ´´Agregado. Redirigiendo a la base de datos´´ y se actualiza el listado de rasgos.	La prueba es satisfactoria ya que el sistema posibilita al usuario de manera exitosa la adición de un rasgo.
1.2	Ausencia de campos llenos de datos válidos	No se introducen datos requeridos	V Sexo	I Nulo	I Nulo	No añade el rasgo y muestra el mensaje de error: ´´Debe seleccionar un criterio de comparación ´´	La prueba es satisfactoria ya que el sistema no posibilita dejar campos vacíos al añadir un rasgo.
			I Nulo	V Cuantitativo	V 788	No añade el rasgo y muestra el mensaje de error: ´´El campo nombre del rasgo está vacío en el rasgo X´´	
			V Estatura	V Por intervalos	I Nulo	No añade el rasgo y muestra el mensaje de error: ´´Los intervalos no están correctos en el rasgo X´´	

1.3	Entrada de datos incorrectos	Se introducen campos incorrectos	I 3no	V Nominal	V Azul	Elimina los caracteres incorrectos en tiempo real para que no puedan permanecer.	La prueba es satisfactoria ya que el sistema no posibilita escribir campos incorrectos al añadir un rasgo.
			V Color	V Nominal	I 34_azul	Elimina los caracteres incorrectos en tiempo real para que no puedan permanecer.	
1.4	Selecciona el botón Cancelar	Se cancela la operación Adicionar rasgo.	N/A	N/A	N/A	Cancela la operación Crear rasgo. Regresa a la página Listar bases de datos.	La prueba es satisfactoria ya que se procede con la cancelación de la creación del rasgo.

Tabla 29: Caso de prueba 2 de la Historia de usuario Gestionar grupos. Escenario Crear grupo

Id	**Escenario**	**Descripción**	**Campos**			**Respuesta del sistema**	**Resultado de la prueba**
			Criterio de comparación	**Tipo de Umbral**	**Umbral experto**		
2.1	Entrada de datos válidos	Permite adicionar un rasgo	V β0-compacto	V Experto	V 0.8	Se crearon satisfactoriamente los grupos. Muestra el mensaje ´´Agregado. Ver grupos´´ y se muestran los grupos y la matriz de semejanza.	La prueba es satisfactoria ya que el sistema posibilita al usuario de manera exitosa la creación de grupos.
2.2	Ausencia de campos llenos de datos válidos	No se introducen datos requeridos	V β0-compacto	I Nulo	I Nulo	No crea los grupos y muestra el mensaje de error: ´´Debe seleccionar un tipo de umbral de semejanza´´	La prueba es satisfactoria ya que el sistema no posibilita dejar campos vacíos al crear un grupo.
			I Nulo	V Experto	V 788	No crea los grupos y muestra el mensaje de error: ´´Debe seleccionar un criterio de comparación´´	
			V β0-conexo	V Experto	I Nulo	No crea los grupos y muestra el mensaje de error: ´´Debe escribir un valor para el criterio de expertos´´	

2.3	Entrada de datos incorrectos	Se introducen campos incorrectos	V β0-compacto	V Experto	V 9	No crea los grupos y muestra el mensaje de error: ´´Valores de umbral de experto fuera de rango´´	
			V β0-compacto	V Experto	I -5	No crea los grupos y muestra el mensaje de error: ´´Valores de umbral de experto fuera de rango´´	
			V β0-compacto	V Experto	I Es9	Elimina los caracteres incorrectos en tiempo real para que no puedan permanecer.	
2.4	Selecciona el botón Cancelar	Se cancela la operación Crear grupos.	N/A	N/A	N/A	Cancela la operación Crear grupos. Regresa a la página Listar bases de datos.	La prueba es satisfactoria ya que se procede con la cancelación de la creación de grupos.

A continuación, se muestran algunas de las pruebas de aceptación realizadas al sistema web USGest y sus resultados:

Tabla 30: Prueba de aceptación 1 Gestionar rasgo

Caso de prueba de aceptación	
Código: HU1_P1	**Historia de usuario:** 1
Nombre: Gestionar rasgos	
Descripción: Prueba la funcionalidad para Gestionar rasgos	
Condiciones de ejecución: Ninguna	
Pasos de ejecución: El usuario llena el campo “nombre del rasgo”, selecciona una opción de “criterio de comparación”, y en caso de ser distinta de “booleano” escribe los valores correspondientes en el(los) campo(s) “dominio” y presiona el botón “Finalizar”.	
Resultados esperados: El rasgo se ha adicionado sin problemas.	

Tabla 31: Prueba de aceptación 2 Gestionar bases de datos

Caso de prueba de aceptación	
Código: HU2_P2	**Historia de usuario:** 2
Nombre: Gestionar bases de datos	
Descripción: Prueba la funcionalidad eliminar base de datos.	
Condiciones de ejecución: Se necesitan tener creadas bases de datos de rasgos.	
Pasos de ejecución: El usuario selecciona la opción “Eliminar”, aparece el mensaje de confirmación “¿Está seguro que desea eliminar la base de datos (nombre de la base de datos)?” y el usuario presiona el botón aceptar.	
Resultados esperados: La base de datos ha sido eliminada sin problemas.	

Tabla 32: Prueba de aceptación 3 Gestionar objetos

Caso de prueba de aceptación	
Código: HU3_P3	**Historia de usuario:** 3
Nombre: Administrar objetos	
Descripción: Prueba la funcionalidad Adicionar objetos	
Condiciones de ejecución: Se necesita tener creada una base de datos de rasgos.	
Pasos de ejecución: El usuario rellena el campo “nombre del objeto” y presiona el botón “Finalizar”.	
Resultados esperados: Los objetos han sido adicionados sin problemas.	

Tabla 33: Prueba de aceptación 4 Administrar grupos

Caso de prueba de aceptación	
Código: HU4_P4	**Historia de usuario:** 4
Nombre: Administrar grupos	
Descripción: Prueba la funcionalidad Crear grupos	
Condiciones de ejecución: Se necesita tener creada una base de datos de objetos con los valores de los rasgos ya asociados.	
Pasos de ejecución: El usuario selecciona el tipo de “umbral de semejanza”, en caso de seleccionar criterio de expertos, se debe escribir un umbral entre el 0 y el 1. Luego el usuario selecciona el tipo de “criterio de agrupamiento” y presiona el botón “Ver grupos”.	
Resultados esperados: Los grupos han sido creados sin problemas.	

Se realizaron 12 casos de pruebas de aceptación, de los que en la primera iteración se encontraron cuatro no conformidades y después fueron rectificadas. Durante la segunda iteración se obtuvieron cinco casos de prueba y se detectaron dos no conformidades que fueron resueltas más adelante. En la tercera iteración se obtuvieron 2 casos de prueba y no se encontraron no conformidades.

En la siguiente figura se muestran los resultados de las pruebas de aceptación realizadas al sistema.

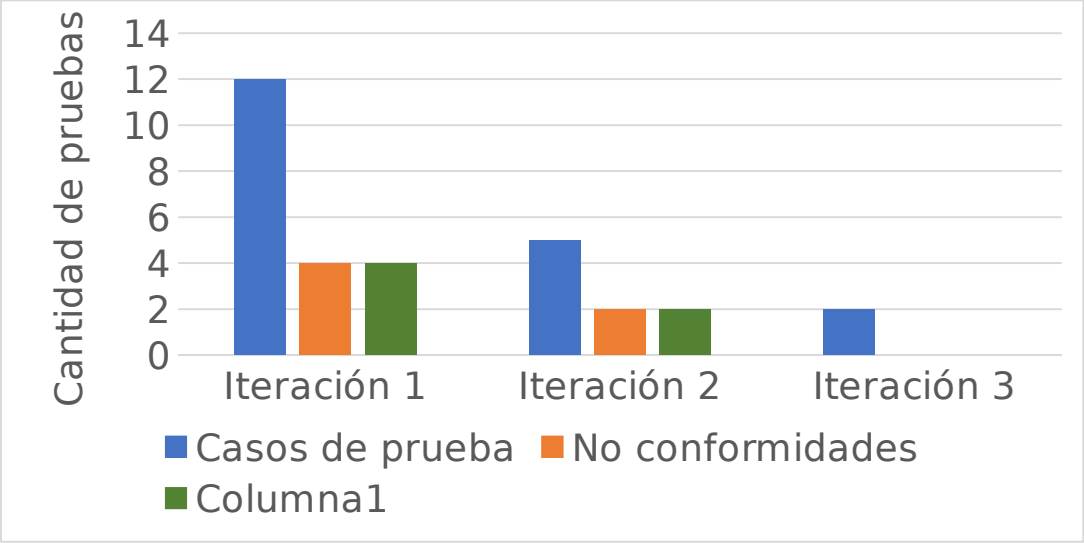


Ilustración 19: Resultados de las pruebas de aceptación al final de cada iteración

Conclusiones del capítulo

Las tareas de programación establecidas nos permitieron la separación de las funcionalidades a programar de cada historia de usuario.

Se plasmó el diagrama de despliegue para conocer la distribución y las prestaciones necesarias para ejecución y puesta en funcionamiento del sistema.

Se procedió con la validación y verificación del sistema web propuesto, aplicando la técnica estática de inspecciones del software mediante las listas de chequeo, y la técnica dinámica de pruebas del software durante todo su ciclo de vida, mediante los casos de prueba de la estrategia, los que permitieron referenciar la aceptación del cliente y unidad de los módulos y en general del sistema web propuesto.

CONCLUSIONES FINALES

Con el desarrollo del presente trabajo de diploma, se dio cumplimiento al objetivo general de la investigación y los objetivos específicos trazados, por tanto se concluye lo siguiente:

- Se elaboró un marco teórico referencial sobre Reconocimiento de Patrones con un enfoque Lógico Combinatorio que sentó las bases para el conocimiento y la profundización del tema.
- El análisis de los sistemas que utilizan las técnicas del Reconocimiento Lógico Combinatorio de Patrones a nivel nacional e internacional mostró la ausencia de los mismos para el proceso de aprendizaje de los algoritmos de clasificación no supervisada.
- Se seleccionó el algoritmo Class de clasificación no supervisada del Reconocimiento Lógico Combinatorio de Patrones para su implementación debido a que forma los grupos por un mayor nivel de semejanza entre los objetos.
- Se seleccionó la metodología ágil XP; las tecnologías HTML5, CSS3, JavaScript y Python 3.9; los frameworks Django 3.2.15 y Bootstrap 4 y las herramientas informáticas Visual Estudio Code 1.72.2, Visual Paradigm for UML 8.0 y PostgreSQL 13.2 para gestionar las bases de datos como los más acordes para una correcta implementación del sistema web.
- Se generaron los artefactos ingenieriles correspondientes a cada una de las etapas de desarrollo del software, lo cual posibilitó realizar el sistema web.
- Se implementó un sistema web que permite la clasificación no supervisada de los objetos mediante el algoritmo Class.
- Se validó la calidad del sistema web resultante mediante las pruebas de software constituidas en la estrategia de prueba, quedando como resultado la satisfacción del cliente con el producto entregado.

RECOMENDACIONES

En busca de una continuidad y mejoras del sistema se recomienda:

- Continuar la implementación aplicando distintos criterios de agrupamiento para establecer comparaciones entre los resultados de la clasificación.
- Aplicar el sistema, como una mejora importante de la problemática descrita.

REFERENCIAS BIBLIOGRÁFICAS

1. COPPOLA, Maria. Qué es la minería de datos: conceptos, técnicas y ejemplos. Online. [Accessed 12 November 2022]. Available from: https://blog.hubspot.es/marketing/mineria-datos
2. RUIZ SHULCLOPER, José. Acerca del surgimiento del Reconocimiento de Patrones en Cuba. *Revista Cubana de Ciencias Informáticas*. June 2013. Vol. 7, no. 2, p. 169–192.
3. Minería de Datos - EcuRed. Online. [Accessed 12 November 2022]. Available from: https://www.ecured.cu/Miner%C3%ADa_de_Datos
4. WEISS, Sholom M. and KAPOULEAS, Ioannis. An empirical comparison of pattern recognition, neural nets, and machine learning classification methods. In : *IJCAI*. 1989. p. 781–787.
5. MONTENEGRO-BERMÚDEZ, Andrés Felipe and PARADA-ROJAS, Cristian David. Diseño e implementación de un sistema de detección de malezas en cultivos cundiboyacenses. Online. 2015. [Accessed 12 November 2022]. Available from: https://repository.ucatolica.edu.co/handle/10983/3202Accepted: 2016-05-14T13:24:59Z
6. FLASIŃSKI, Mariusz. Pattern Recognition and Cluster Analysis. In : FLASIŃSKI, Mariusz (ed.), *Introduction to Artificial Intelligence*. Online. Cham : Springer International Publishing, 2016. p. 141–156. [Accessed 11 November 2022]. ISBN 978-3-319-40022-8.
7. CANALS, Pep and CYBERCLICK. Algoritmos de clustering: el futuro del marketing sin cookies. Online. [Accessed 12 November 2022]. Available from: https://www.cyberclick.es/numerical-blog/algoritmos-de-clustering-el-futuro-del-marketing-sin-cookies
8. PINNINGHOFF J, M. Angélica, CONTRERAS A, Ricardo and SALCEDO L, Pedro. Genetic algorithms as a tool for structuring collaborative groups. *Natural Computing*. 2017. Vol. 16, no. 2, p. 231–239.
9. Centro de Estudios de Reconocimiento de Patrones y Minería de Datos - EcuRed. Online. [Accessed 12 November 2022]. Available from: https://www.ecured.cu/Centro_de_Estudios_de_Reconocimiento_de_Patrones_y_Miner%C3%ADa_de_Datos
10. CASTILLO, Manuel Capote. *Trabajo con documentos: en Ciencias de la Educación*. . Editorial Universitaria (Cuba), 2020. ISBN 978-959-16-4268-4. Google-Books-ID: gZv6DwAAQBAJ
11. ROUHIAINEN, Lasse. Inteligencia artificial - 1. Introducción a la inteligencia artificial. Online. 2018. [Accessed 11 November 2022]. Available from: https://reader.digitalbooks.pro/book/preview/111864/id_1_Introduccion_a_la_inteligencia_artificial_0005_0000
12. ORRIOLS-PUIG, Albert, CASILLAS, Jorge and BERNADÓ-MANSILLA, Ester. Genetic-based machine learning systems are competitive for pattern recognition. *Evolutionary Intelligence*. 1 October 2008. Vol. 1, no. 3, p. 209–232. DOI 10.1007/s12065-008-0013-9.
13. RUIZ-SHULCLOPER, José, CARRASCO-OCHOA, Jesús Ariel and MARTÍNEZ-TRINIDAD, J. Reconocimiento de patrones: conceptos y metodología. *undefined*. Online. 2013. [Accessed 11 November 2022]. Available from: https://www.semanticscholar.org/paper/Reconocimiento-de-patrones%3A-conceptos-y-metodolog%C3%ADa-Ruiz-Shulcloper-Carrasco-Ochoa/e0a32b8ca044474ad2c3109c70a06551c3e12b55
14. KOZAK, Marcin and SCAMAN, Christine H. Unsupervised classification methods in food sciences: discussion and outlook. *Journal of the Science of Food and Agriculture*. 2008. Vol. 88, no. 7, p. 1115–1127. DOI https://doi.org/10.1002/jsfa.3215.
15. SEPÚLVEDA HERAZO, Luis and COAVAS ALMENTERO, Yeni. Desarrollar una plataforma web para el control de asistencia de estudiantes, docentes y administrativos de la Universidad de Córdoba sede Lorica mediante el uso de tarjetas inteligentes, dispositivos móviles y sms. Online. [Accessed 12 November 2022]. Available from: https://repositorio.unicordoba.edu.co/handle/123456789/733
16. TORRES, Ligia Amparo and SÁNCHEZ, Luisa Fernanda. Generalización de patrones numéricos en el desarrollo del pensamiento variacional en la educación primaria. *VII Congreso Iberoamericano de Educación Matemática*. Online. September 2013. [Accessed 12 November 2022]. Available from: http://cibem.org/
17. RUIZ FERNÁNDEZ, Fredesbindo. Resolvemos patrones aditivos. *Universidad Nacional de Trujillo*. Online. 2019. [Accessed 11 November 2022]. Available from: http://dspace.unitru.edu.pe/handle/UNITRU/14691Accepted: 2019-10-16T22:31:27Z
18. EMPRESA, Universidad de la. ¿Qué entendemos por algoritmo? Online. 6 July 2021. [Accessed 12 November 2022]. Available from: https://ude.edu.uy/que-son-algoritmos/

19. PUCHA-COFREP, Franz, FRIES, Andreas, CÁNOVAS-GARCÍA, Fulgencio, OÑATE-VALDIVIESO, Fernando, GONZÁLEZ-JARAMILLO, Víctor and PUCHA-COFREP, Darwin. *Fundamentos de SIG: Aplicaciones con ArcGIS*. . Franz Pucha Cofrep, 2017. ISBN 978-9942-28-901-8. Google-Books-ID: XOIsDwAAQBAJ
20. RODRÍGUEZ CRUZ, Josué and ZELAYA, Carlos. QGIS: Sistemas de Información Geográfica. Guía metodológica - Módulo IV diplomado. Online. April 2021. [Accessed 12 November 2022]. Available from: https://cgspace.cgiar.org/handle/10568/113244Accepted: 2021-04-08T14:42:52Z
21. ALVES, Fernanda B. and LIMA, Danielli A. Uso de la clasificación para el análisis y la minería de datos en la herramienta de enseñanza-aprendizaje Google Classroom. *Nuevas Ideas en Informática Educativa*. 2018. Vol. 4, p. 589–594.
22. ANDRADE, Juan Carlos. *Desarrollo de un prototipo informático de consulta y asignación de plazas libres en parqueaderos*. Online. bachelorThesis. Pontificia Universidad Católica del Ecuador, 2012. [Accessed 12 November 2022]. Available from: http://repositorio.puce.edu.ec:80/handle/22000/6354Accepted: 2015-04-07T16:17:58Z
23. Aplicando el método de Boehm y Turner. Applying the Boehm and Turner method - PDF Descargar libre. Online. [Accessed 11 November 2022]. Available from: https://docplayer.es/1648353-Aplicando-el-metodo-de-boehm-y-turner-applying-the-boehm-and-turner-method.html
24. FIGUEROA-DIAZ, Roberth, SÓLIS, Camilo and CABRERA, Armando. *METODOLOGÍAS TRADICIONALES VS. METODOLOGÍAS ÁGILES*. . 2007.
25. DUARTE, Ailin Orjuela and ROJAS, Mauricio. Las metodologías de desarrollo ágil como una oportunidad para la ingeniería del software educativo. *Revista Avances en Sistemas e Informática*. 2008. Vol. 5, no. 2, p. 159–171.
26. CHANDI, Pablo. Ingenieria.de.software.enfoque.practico.7ed.Pressman.PDF. Online. [Accessed 11 November 2022]. Available from: https://www.academia.edu/26345704/Ingenieria_de_software_enfoque_practico_7ed_Pressman_PDFIngenieria.de.software.enfoque.practico.7ed.Pressman.PDF
27. BRIONES, Ponce and KLÉBER, Darwin. *Análisis Comparativo De Los Entornos De Desarrollo Integrados (Ide): Eclipse, Netbeans Y Jdeveloper Para El Desarrollo De Aplicaciones Java Enterprise Edition*. Online. Thesis. Universidad de Guayaquil. Facultad de Ciencias Matemáticas y Físicas. Carrera de Ingeniería en Sistemas Computacionales, 2016. [Accessed 11 November 2022]. Available from: http://repositorio.ug.edu.ec/handle/redug/15862Accepted: 2017-03-20T18:01:14Z
28. BERMÚDEZ, Pablo Enmanuel Ramos, ZOPPÉ, Monica, LONI, Tiziana, MERIÑO, Mario Pupo and CONYEDO, Edisel Navas. Desarrollo de una nueva versión de BioBlender, un módulo de Blender para visualización de biomoléculas. *Serie Científica de la Universidad de las Ciencias Informáticas*. 2022. Vol. 15, no. 6, p. 35–53.
29. HUGON, Jérôme. *C# 7: desarrolle aplicaciones Windows con Visual Studio 2017*. . Ediciones ENI, 2018. ISBN 978-2-409-01344-7. Google-Books-ID: 1e9dfieV4UEC
30. CLARO, Rosendo L. Hernández and NAVARRO, Deibys Greguas. Estándares de diseño Web. *Ciencias de la Información*. 2010. Vol. 41, no. 2, p. 69–71.
31. El gran libro de HTML5, CSS3 y JavaScript - PDF Drive. Online. [Accessed 11 November 2022]. Available from: http://www.pdfdrive.com/el-gran-libro-de-html5-css3-y-javascript-e51778459.html
32. ORTEGA, Denzel Javier Ovando, ASESOR, O. R. and PÉREZ, Miguel Alonso Berrelleza. Bootstrap y Laravel, herramientas para el desarrollo de aplicaciones web. *UNIVERSIDAD POLITÉCNICA DE SINALOA, Mexico*. 2019.
33. DIEGUEZ, M. Sc Fidel Enrique Castro, TOLEDO, Carlos Luis Meriño, VILTRES, M. Sc Yamira Medel and DE LA CRUZ, Eric Guía. Sistema informático para la elaboración de los planes de trabajo en la Empresa de Acumuladores “XX Aniversario” del municipio de Manzanillo. .
34. ABENZA, P. Pablo Garrido. *Comenzando a programar con JAVA*. . Universidad Miguel Hernández, 2015. ISBN 978-84-16024-24-7. Google-Books-ID: 4v8QCgAAQBAJ
35. HERAS, Fernando Herrera de las. Python, el lenguaje de moda. *Linux Actual: la primera revista en castellano del sistema operativo Gnu/Linux*. 2000. Vol. 2, no. 12, p. 42–44.
36. CHALLENGER-PÉREZ, Ivet, DÍAZ-RICARDO, Yanet and BECERRA-GARCÍA, Roberto Antonio. El lenguaje de programación Python. *Ciencias Holguín*. 2014. Vol. 20, no. 2, p. 1–13.
37. THANAKI, Jalaj. *Python Natural Language Processing*. . Packt Publishing Ltd, 2017. ISBN 978-1-78728-552-1. Google-Books-ID: IedDDwAAQBAJ
38. SERGIO, DE LA PEÑA O’SHEA. *SGBD e instalación*. . Editorial Paraninfo, 2017. ISBN 978-84-283-9656-1. Google-Books-ID: yVPVDQAAQBAJ
39. PILICITA GARRIDO, Anabel, BORJA LÓPEZ, Yolanda and GUTIÉRREZ CONSTANTE, Gonzalo. Rendimiento de MariaDB y PostgreSQL. Online. 2021. [Accessed 11 November 2022]. Available from: https://repositorio.upse.edu.ec/handle/46000/7315Accepted: 2022-05-15T21:11:42Z

40. RAMOS, Sarai de Jesús Gil, RAMÍREZ, Gabriela Xicoténcatl, MANDUJANO, Martín Muñoz and SERRANO, Sharol Rubí González. Implementación de un modelo de datos para el almacenamiento de información climatológica en el estado de Querétaro. *Revista NTHE*. 2018. Vol. 24, p. 16–19.
41. Bases de datos en PostgreSQL - PDF Descargar libre. Online. [Accessed 11 November 2022]. Available from: https://docplayer.es/2240181-Bases-de-datos-en-postgresql.html
42. MENDOZA PEÑA, Dayana and BAQUERO HERNÁNDEZ, Lionel Rodolfo. Extensión de la herramienta visual paradigm for UML para la evaluación y corrección de diagramas de casos de uso. Online. June 2016. [Accessed 11 November 2022]. Available from: https://repositorio.uci.cu/jspui/handle/123456789/7944Accepted: 2019-05-28T15:16:31Z
43. Diseño De La Base De Datos Para Sistemas De Digitalización Y - ID:5eab364fd9c30. Online. [Accessed 11 November 2022]. Available from: https://xdoc.mx/documents/diseo-de-la-base-de-datos-para-sistemas-de-digitalizacion-y-5eab364fd9c30
44. JACOBSON, Ivar, RUMBAUGH, James E. and BOOCH, Grady. El proceso unificado de desarrollo de software. *undefined*. Online. 2000. [Accessed 11 November 2022]. Available from: https://www.semanticscholar.org/paper/El-proceso-unificado-de-desarrollo-de-software-Jacobson-Rumbaugh/a8b25bc8c8103e486a348d0b1aa8b4c1bb0b9728
45. VILLALOBOS, Gustavo Martínez, SÁNCHEZ, Germán Darío Camacho and GUTIÉRREZ, Daniel Alberto Biancha. Diseño de framework web para el desarrollo dinámico de aplicaciones. *Scientia et technica*. 2010. Vol. 16, no. 44, p. 178–183.
46. LÓPEZ PÉREZ, David. Desarrollo de una Aplicación Web para la Gestión de una Beca Deportiva en Python. Online. July 2022. [Accessed 11 November 2022]. Available from: https://oa.upm.es/71548/
47. VIDAL-SILVA, Cristian L., SÁNCHEZ-ORTIZ, Aurora, SERRANO, Jorge, RUBIO, José M., VIDAL-SILVA, Cristian L., SÁNCHEZ-ORTIZ, Aurora, SERRANO, Jorge and RUBIO, José M. Experiencia académica en desarrollo rápido de sistemas de información web con Python y Django. *Formación universitaria*. October 2021. Vol. 14, no. 5, p. 85–94. DOI 10.4067/S0718-50062021000500085.
48. LEYVA, Luis A. Hernández, TAMAYO, Sergio Cleger and SALGADO, Rosa Urquiza. Sistema informático para la elaboración y publicación del horario docente de la Universidad de Holguín. *RILCO: Revista de Investigación Latinoamericana en Competitividad Organizacional*. 2020. No. 5, p. 5.
49. Historias de Usuario Ingeniería de requisitos ágil. v PDF Free Download. Online. [Accessed 11 November 2022]. Available from: https://docplayer.es/19604718-Historias-de-usuario-ingenieria-de-requisitos-agil-v-1-0.html
50. PALACIOS, Ramos and ESTEFANÍA, Diana. Diseño de un modelo de evaluación de la calidad de productos de software, basado en métricas externas y usabilidad aplicado a un caso de estudio. Online. 30 August 2016. [Accessed 11 November 2022]. Available from: http://bibdigital.epn.edu.ec/handle/15000/16668Accepted: 2016-09-02T14:15:06Z
51. LOPEZ, Doheny Vicent Rivas. Reglas y Prácticas en eXtreme Programming. Online. [Accessed 11 November 2022]. Available from: https://www.academia.edu/39538575/Reglas_y_Pr%C3%A1cticas_en_eXtreme_ProgrammingReglas y Prácticas en eXtreme Programming
52. MARTÍNEZ, Raúl Noriega. *El Proceso de Desarrollo de Software*. . IT Campus Academy, 2015. ISBN 978-1-5146-4786-8. Google-Books-ID: BTTsCgAAQBAJ
53. HAYA, Mariela, FRANCH, Xavier and MAYOL, Enric. Uso de los Diagramas de Actividades UML Y el Lenguage i* Modelado del Proceso de Implantacion del Balanced Scorecard. In : *WER*. 2004. p. 88–99.
54. Arquitectura de Software. *SG Buzz*. Online. [Accessed 11 November 2022]. Available from: https://sg.com.mx/content/view/409
55. PDF de programación - El libro de Django. Online. [Accessed 11 November 2022]. Available from: https://www.lawebdelprogramador.com/pdf/9186-El-libro-de-Django.html
56. NUÑEZ MALDONADO, Alejandro, PERAZA SUÁREZ, Yuniel, MÉNDEZ MEDEROS, Arianne and ESCALONA PERAL, Lourdes. Sistema para la gestion administrativa de la facultad 7 : modulo para la planificacion y seguimiento de actividades. Online. 2010. [Accessed 11 November 2022]. Available from: https://repositorio.uci.cu/jspui/handle/ident/TD_03088_10Accepted: 2016-09-14T19:32:04Z
57. OLIVEIRA, Nunes de and DAVID, José. Aplicación web de apoyo al regante. Online. September 2018. [Accessed 11 November 2022]. Available from: https://repositori.udl.cat/handle/10459.1/64822Accepted: 2018-10-04T08:16:48Z
58. FOWLER, Martin and SCOTT, Kendall. *UML gota a gota: actualizado para cubrir la version 1*. . Pearson Educación, 1999. ISBN 978-968-444-364-8. Google-Books-ID: AL0YkFeaHwIC
59. CASAS, Sandra I. and REINAGA, Héctor H. ASPECTOS TEMPRANOS: UN ENFOQUE BASADO EN TARJETAS CRC. *Avances en Sistemas e Informática*. 1 January 2009. Vol. 6, no. 1, p. 85–92.

60. CAMPOS CALÁS, David Juan, LA ROSA SORDO, Vilmavis and ROCHE RODRÍGUEZ, Flavio E. Plugin para la migración de datos desde MS Acces a PostgreSQL, para la herramienta de administración de Bases de Datos HABD. Online. 15 February 2015. [Accessed 11 November 2022]. Available from: https://repositorio.uci.cu/jspui/handle/ident/8786Accepted: 2016-09-14T19:30:41Z
61. GRASP: Alta cohesión y bajo acoplamiento. *GRASP*. Online. [Accessed 11 November 2022]. Available from: https://juan-garcia-carmona.blogspot.com/2012/09/grasp-alta-cohesion-y-bajo-acoplamiento.html
62. GRASP: Controlador. *GRASP*. Online. [Accessed 11 November 2022]. Available from: https://juan-garcia-carmona.blogspot.com/2012/09/grasp-controlador.html
63. GRASP: Creador. *GRASP*. Online. [Accessed 11 November 2022]. Available from: https://juan-garcia-carmona.blogspot.com/2012/09/grasp-creador.html
64. SILBERSCHATZ, Abraham, KORTH, Henry F., SUDARSHAN, S., PÉREZ, Fernando Sáenz, SANTIAGO, Antonio Ibarra and SÁNCHEZ, Antonio Vaquero. *Fundamentos de bases de datos*. . McGraw-Hill Ciudad de México, México, 2002.
65. VARÓN, Ángel. *Ingeniería de software I*. Online. Bogotá: AREANDINA. Fundación Universitaria del Área Andina, 2018. [Accessed 12 November 2022]. ISBN 978-958-54-6299-1. Available from: https://digitk.areandina.edu.co/handle/areandina/1235Accepted: 2018-11-27T16:31:44ZArtwork Medium: application/pdfInterview Medium: application/pdf
66. SOMMERVILLE, Ian. *Ingeniería del software*. . Pearson Educación, 2005. ISBN 978-84-7829-074-1. Google-Books-ID: gQWd49zSut4C
67. SAMPER HERNÁNDEZ, Yasniel. Aplicación móvil en Android con la información estadística de la planificación en tiempo real para el Sistema de Planificación de Actividades 3. Online. June 2018. [Accessed 11 November 2022]. Available from: https://repositorio.uci.cu/jspui/handle/123456789/9959Accepted: 2022-02-18T13:14:39Z
68. LEÓN PERDOMO, Yeniset, FEBLES ESTRADA, Ailyn and ESTRADA SENTÍ, Vivian. Estrategia de Pruebas Exploratorias para mejorar el rendimiento del Laboratorio Industrial de Pruebas de Software de Calisoft. Online. 29 October 2013. [Accessed 11 November 2022]. Available from: https://repositorio.uci.cu/jspui/handle/ident/7972Accepted: 2016-09-21T15:00:14Z
69. FERNÁNDEZ, Francisco and ÁNGEL, Miguel. Aplicación de técnicas de pruebas automáticas basadas en propiedades a los diferentes niveles de prueba del software. Online. 2015. [Accessed 11 November 2022]. Available from: https://ruc.udc.es/dspace/handle/2183/14814Accepted: 2015-07-24T14:31:20Z
70. O, José Luis Aristegui. Los casos de prueba en la prueba del software. *Lámpsakos*. 2010. No. 3, p. 27–34. Autoría: José Luis Aristegui O..Localización: Lámpsakos. Nº. 3, 2010.Artículo de Revista en Dialnet.
71. RUSKIN, John. *La calidad nunca es un accidente; siempre es el resultado de un esfuerzo de la inteligencia.* . PhD Thesis. Universidad de las Ciencias Informáticas, 2010.

ANEXOS

Anexo 1: Entrevista realizada a los principales dirigentes del Grupo de Inteligencia Artificial y Reconocimiento de Patrones de la Universidad de las Ciencias Informáticas.
Entrevista:
¿Qué actividades se realizan en el centro?
¿Qué proyectos se están desarrollando?
¿Qué es el Reconocimiento de Patrones? ¿En qué se utiliza?
¿Cuáles son los algoritmos existentes? ¿Cuáles de ellos se utilizan en el Grupo?
¿Cómo funciona el algoritmo CLASS?

Anexo 2: Tareas de ingeniería

Iteración 1

Tabla 34: Tarea de ingeniería 3 Eliminar rasgos

Tarea	
Número de tarea: 3	**Número de la historia de usuario:** 1
Nombre de la tarea: Eliminar rasgos	
Tipo de tarea: Desarrollo	**Puntos estimados:**
Fecha de inicio: 06/06/2022	**Fecha de fin:** 10/06/2022
Descripción: El usuario puede eliminar un rasgo ya existente	

Tabla 35: Tarea de ingeniería 4 Listar rasgos

Tarea	
Número de tarea: 3	**Número de la historia de usuario:** 1
Nombre de la tarea: Eliminar rasgos	
Tipo de tarea: Desarrollo	**Puntos estimados:**
Fecha de inicio: 06/06/2022	**Fecha de fin:** 10/06/2022
Descripción: El usuario puede eliminar un rasgo ya existente	

Tabla 36: Tarea de ingeniería 5 Insertar bases de datos

Tarea	
Número de tarea: 5	**Número de la historia de usuario:** 2
Nombre de la tarea: Insertar bases de datos	
Tipo de tarea: Desarrollo	**Puntos estimados:**
Fecha de inicio: 13/06/2022	**Fecha de fin:** 17/06/2022
Descripción: El usuario puede insertar una nueva base de datos	

Tabla 37: Tarea de ingeniería 6 Modificar bases de datos

Tarea	
Número de tarea: 6	**Número de la historia de usuario:** 2
Nombre de la tarea: Modificar bases de datos	
Tipo de tarea: Desarrollo	**Puntos estimados:**
Fecha de inicio: 13/06/2022	**Fecha de fin:** 17/06/2022

Descripción: El usuario puede modificar una base de datos ya existente

Tabla 38: Tarea de ingeniería 7 Eliminar bases de datos

Tarea	
Número de tarea: 7	**Número de la historia de usuario:** 2
Nombre de la tarea: Eliminar bases de datos	
Tipo de tarea: Desarrollo	**Puntos estimados:**
Fecha de inicio: 13/06/2022	**Fecha de fin:** 17/06/2022
Descripción: El usuario puede eliminar una base de datos ya existente	

Tabla 39: Tarea de ingeniería 8 Listar bases de datos

Tarea	
Número de tarea: 8	**Número de la historia de usuario:** 2
Nombre de la tarea: Listar bases de datos	
Tipo de tarea: Desarrollo	**Puntos estimados:**
Fecha de inicio: 13/06/2022	**Fecha de fin:** 17/06/2022
Descripción: El usuario puede insertar una nueva base de datos	

Tabla 40: Tarea de ingeniería 9 Ver bases de datos

Tarea	
Número de tarea: 9	**Número de la historia de usuario:** 2
Nombre de la tarea: Ver bases de datos	
Tipo de tarea: Desarrollo	**Puntos estimados:**
Fecha de inicio: 13/06/2022	**Fecha de fin:** 17/06/2022
Descripción: El usuario puede insertar una nueva base de datos	

Tabla 41: Tarea de ingeniería 10 Insertar objetos

Tarea	
Número de tarea: 10	**Número de la historia de usuario:** 3
Nombre de la tarea: Insertar objetos	
Tipo de tarea: Desarrollo	**Puntos estimados:**
Fecha de inicio: 20/06/2022	**Fecha de fin:** 01/07/2022
Descripción: El usuario puede insertar una nueva base de datos	

Tabla 42: Tarea de ingeniería 11 Modificar objetos

Tarea	
Número de tarea: 11	**Número de la historia de usuario:** 3
Nombre de la tarea: Modificar objetos	
Tipo de tarea: Desarrollo	**Puntos estimados:**
Fecha de inicio: 20/06/2022	**Fecha de fin:** 01/07/2022
Descripción: El usuario puede modificar un objeto ya existente	

Tabla 43: Tarea de ingeniería 12 Eliminar objetos

Tarea	
Número de tarea: 12	**Número de la historia de usuario:** 3
Nombre de la tarea: Eliminar objetos	
Tipo de tarea: Desarrollo	**Puntos estimados:**
Fecha de inicio: 20/06/2022	**Fecha de fin:** 01/07/2022
Descripción: El usuario puede eliminar un objeto ya existente	

Tabla 44: Tarea de ingeniería 13 Listar objetos

Tarea	
Número de tarea: 13	**Número de la historia de usuario:** 3
Nombre de la tarea: Listar objetos	
Tipo de tarea: Desarrollo	**Puntos estimados:**
Fecha de inicio: 20/06/2022	**Fecha de fin:** 01/07/2022
Descripción: El usuario puede listar los objetos ya existentes	

Tabla 45: Tarea de ingeniería 14 Ver objetos

Tarea	
Número de tarea: 14	**Número de la historia de usuario:** 3
Nombre de la tarea: Ver objetos	
Tipo de tarea: Desarrollo	**Puntos estimados:**
Fecha de inicio: 20/06/2022	**Fecha de fin:** 01/07/2022
Descripción: El usuario puede ver los detalles de los objetos ya existentes	

Iteración 2

Tabla 46: Tarea de ingeniería 17 Ver grupos

Tarea	
Número de tarea: 17	**Número de la historia de usuario:** 4
Nombre de la tarea: Ver grupos	
Tipo de tarea: Desarrollo	**Puntos estimados:**
Fecha de inicio: 05/09/2022	**Fecha de fin:** 23/09/2022
Descripción: El usuario puede ver los detalles de los grupos ya existentes	

Iteración 3

Tabla 47: Tarea de ingeniería 19 Modificar usuario

Tarea	
Número de tarea: 19	**Número de la historia de usuario:** 5
Nombre de la tarea: Modificar usuario	
Tipo de tarea: Desarrollo	**Puntos estimados:**
Fecha de inicio: 26/09/2022	**Fecha de fin:** 30/09/2022
Descripción: El administrador puede modificar usuarios ya existentes	

Tabla 48: Tarea de ingeniería 20 Eliminar usuario

Tarea	
Número de tarea: 20	**Número de la historia de usuario:** 5
Nombre de la tarea: Eliminar usuario	
Tipo de tarea: Desarrollo	**Puntos estimados:**
Fecha de inicio: 26/09/2022	**Fecha de fin:** 30/09/2022
Descripción: El administrador puede eliminar usuarios ya existentes	

Tabla 49: Tarea de ingeniería 21 Listar usuarios

Tarea	
Número de tarea: 21	**Número de la historia de usuario:** 5
Nombre de la tarea: Listar usuarios	
Tipo de tarea: Desarrollo	**Puntos estimados:**
Fecha de inicio: 26/09/2022	**Fecha de fin:** 30/09/2022
Descripción: El administrador puede listar los usuarios ya existentes	