FACULTAD 4

Trabajo de diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Paquete de Interfaces para el Consumo de Servicios de Procesamientos de Datos en BrainSSys

Autor: Kevin Castillo Pérez

Tutor: Dr.C. Arturo Orellana García, P.A, Inv. A.

La Habana, noviembre de 2023

Año 65 de la Revolución

DECLARACIÓN DE AUTORÍA

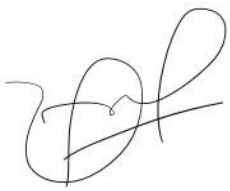
Declaro ser autor de la presente tesis “Paquete de Interfaces para el Consumo de Servicios de Procesamientos de Datos en BrainSSys”, concedo a la Universidad de las Ciencias Informáticas y en especial al Centro de Informática Médica la autorización a hacer uso del mismo en su beneficio.

Para que así conste firmamos la presente a los 27 días del mes de noviembre del año 2023.

Kevin Castillo Pérez

Firma del Autor

Dr.C Arturo Orellana García



Firma del Tutor

DATOS DE CONTACTO

Dr. C Arturo Orellana García <aorellana@uci.cu>

Graduado de Ingeniería en Ciencias Informáticas en la Universidad de las Ciencias Informáticas en el año 2012. Obtuvo el grado de Máster en Informática Aplicada en 2015 desarrollando una herramienta informática basada en técnicas de minería de procesos para identificar problemas en la ejecución de procesos de negocio. Doctor en Ciencias Técnicas desde 2016 presentando un modelo computacional para la detección de variabilidad en procesos de negocio del entorno sanitario aplicando minería de procesos. Se desempeña como líder del Grupo de Investigación de Informática Médica y Asesor de Capacitación, Desarrollo e Investigación del Centro de Informática Médica. Ha liderado proyectos I+D+i de desarrollo de componentes de software a partir de minería de procesos para el análisis de procesos de negocio del entorno hospitalario. Investiga la Ingeniería de comportamiento, la medicina de precisión y el procesamiento de imágenes médicas. Ha tutorado Trabajos de Diploma, Maestrías y Doctorados enfocados al análisis de procesos de negocio, la informática médica y otras áreas del conocimiento. Posee más de 100 publicaciones en revistas y eventos científicos.

AGRADECIMIENTOS

A mi familia por el incansable apoyo, a mi padre por la persistencia y consejos, a mi madre por la preocupación y entrega, a mi hermano por enseñarme a siempre seguir adelante y tomar todas las oportunidades que se me presenten. Su comprensión y motivación constante fueron fundamentales para alcanzar esta meta. A mis amigos, compañeros de escuela, y todas aquellas personas que, de manera directa o indirecta, me ayudaron en el transcurso de mi carrera y tesis. Asimismo, valoro en gran medida la orientación brindada por mi tutor Arturo Orellana y por confiar en mí para pertenecer a este gran proyecto.

.

DEDICATORIA

La presente tesis se la dedico a mi familia que gracias a su apoyo pude concluir mi carrera.

RESUMEN

La neurociencia es una ciencia multidisciplinaria que estudia el sistema nervioso generando gran cantidad de datos. El procesamiento de esta información es posible mediante la computación de alto rendimiento (HPC). El Centro de Neurociencias de Cuba (CNEURO) se dedica a analizar datos neurológicos para comprender el funcionamiento cerebral. Sus especialistas enfrentan la dificultad de ejecutar tareas en el clúster de forma manual, pudiendo cometer errores. Además, muchos no cuentan con los conocimientos necesarios para trabajar con servidores e ingresar instrucciones a través de línea de comandos. Esta problemática motivó al desarrollo de interfaces gráficas para facilitar a los investigadores la gestión remota de procesos en el clúster. El objetivo fue simplificar el uso del clúster para los investigadores y poder enfocarse en la interpretación de datos sin preocuparse por aspectos técnicos de computación. La solución permite a los especialistas de neurociencia ejecutar análisis de forma automática a través de una interfaz gráfica amigable, evita los posibles errores humanos al escribir comandos de forma manual, agilizando los procesamientos y analizando una mayor cantidad de información en menos tiempo. La herramienta fue validada mediante pruebas que demostraron ser satisfactoria en atributos como facilidad de uso, rendimiento y confiabilidad. Cumplió con simplificar las tareas computacionales para los expertos en neurociencia.

PALABRAS CLAVE: Clúster, Neurociencia, Procesamiento, Usabilidad

ABSTRACT

Neuroscience is a multidisciplinary science that studies the nervous system by generating large amounts of data. The processing of this information is made possible through high performance computing (HPC). The Center for Neurosciences of Cuba (CNEURO) is dedicated to analyzing neurological data to understand brain functioning. Its specialists face the difficulty of executing tasks on the cluster manually, which can lead to errors. In addition, many do not have the necessary knowledge to work with servers and enter instructions through the command line. This problem motivated the development of graphical interfaces to make it easier for researchers to remotely manage processes in the cluster. The goal was to simplify the use of the cluster for researchers and to be able to focus on data interpretation without worrying about technical computing issues. The solution allows neuroscience specialists to run analyses automatically through a user-friendly graphical interface, avoiding possible human errors when typing commands manually, speeding up processing and analyzing a greater amount of information in less time. The tool was validated through tests that proved to be satisfactory in attributes such as ease of use, performance and reliability. It simplified computational tasks for neuroscience experts.

KEYWORDS: *Cluster Computing, Data Processing, Neuroscience, Usability*

INDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

El Centro de Neurociencias de Cuba (CNEURO) es una institución cubana de investigación y desarrollo dedicado a la investigación transnacional en neurociencia, neurotecnología y otras tecnologías médicas, que abarca desde la investigación básica hasta el desarrollo, producción y comercialización de tecnologías. La entidad lleva a cabo investigaciones en una amplia gama de temas que incluyen neurociencia cognitiva, neuroinformática, neuroimagen funcional, análisis de señal bioeléctrica, modelación matemática, investigación neuroquímica, genética molecular e impresión 3D para dispositivos médicos. Entre sus áreas de especial interés se encuentran: desarrollo de nuevos métodos de neuroimagen, la búsqueda de nuevos biomarcadores, enfoques de diagnóstico y moléculas terapéuticas para la enfermedad de Alzheimer, el desarrollo de tecnologías basadas en neuroinformática para el análisis de datos de EEG y MRI en condiciones cerebrales normales y patológicas. (CNEURO, 2023)

La institución CNEURO es la coordinadora del Programa Nacional de Ciencia y Tecnología de Neurociencia y Neuro tecnología, por lo que trabaja en estrecha colaboración con distintas entidades, una de ellas la Universidad de Ciencias Informáticas, que a partir de las investigaciones conjuntas de CNEURO se aprobó y desarrolló un proyecto de investigación nacional denominado BrainSSys. El proyecto tiene entre sus objetivos desarrollar una plataforma digital para la estructuración, manejo de bases de datos multimodales de neurociencias y análisis de datos estandarizados de cerebro. (CNEURO, 2023)

La plataforma BrainSSys es una herramienta orientada al almacenamiento, gestión, visualización y procesamiento de datos provenientes de estudios del cerebro humano y, en última instancia, contribuye al avance de las neurociencias. Para el uso de procesamiento de datos, CNEURO hace uso de clúster de computadoras, mayormente conocido como HPC (*High Performance Computer*) o Computadora de Alto Rendimiento, se define como un sistema de procesamiento paralelo o distribuido que tiene como finalidad mejorar el rendimiento en la ejecución de algoritmos que requieran grandes cantidades de tiempo-máquina, y, por ende, facilitar la obtención de resultados en los procesos investigativos. Consta de un conjunto de computadoras independientes, interconectadas entre sí, de tal manera que funcionan como un solo recurso computacional. A cada uno de los elementos del clúster se le conoce como nodo que pueden tener uno o varios procesadores, memoria RAM, interfaces de red, dispositivos de entrada y salida, y sistema operativo. (IBM, 2023) Los nodos pueden estar contenidos e interconectados en un solo gabinete, o, como en muchos casos, acoplados a través de una LAN (*Local Area Network*). Otro componente básico en un clúster es la interfaz de la red, la cual es responsable de transmitir y recibir los paquetes de datos, que viajan a partir de la red entre los nodos. Finalmente, el lograr que todos

estos elementos funcionen como un solo sistema, es la meta a la que se quiere llegar para dar origen a un clúster. (UNAM, 2022)

En una entrevista realizada a los especialistas que utilizan el HPC del centro se detectaron los principales problemas que presenta la plataforma a la hora de controlar el procesamiento de los datos y gestionar los usuarios que la utilizan los cuales fueron las siguientes deficiencias:

- En CNEURO se presentan dificultades tecnológicas y financieras para acceder a los servicios que ofrecen las plataformas internacionales, por lo que el procesamiento que se desarrolla sobre sus bases de datos se realiza de forma manual en sus servidores de HPC.
- Los investigadores no cuentan con especialistas se ven obligados a programar instrucciones en consola de forma manual para ejecutar tareas de procesamiento y análisis.
- El centro no cuenta con especialistas informáticos para manipular las instrucciones de consola.
- En ocasiones se debe repetir el proceso debido a defectos en las instrucciones lo cual atrasa la planificación de la investigación.
- No se cuenta con mecanismos de seguridad para gestionar los permisos de los usuarios que acceden a los procesamientos.
- Los investigadores no están capacitados para ejecutar una tarea en el clúster por lo que necesitan el apoyo de un recurso humano que disponga de las habilidades técnicas necesarias.

A partir de la situación problemática antes descrita, se identifica el siguiente problema de la investigación: ¿Cómo facilitar el uso de los procesos de lanzamiento de tareas en el clúster para los especialistas del Centro Nacional de Neurociencias de Cuba?

El objetivo es desarrollar un paquete de interfaces de la herramienta para el clúster de CNEURO que optimice la experiencia de usuario en la gestión de las tareas de procesamiento de datos de neurociencias y poder desplegar estos servicios a la nube para mejor disposición del sistema.

Objeto de estudio: Proceso de lanzamiento de tareas en un clúster de procesamiento de datos.

El campo de acción se centra en la gestión del procesamiento de datos de neurociencias en el clúster de CNEURO.

Para dar cumplimiento al objetivo general se definen las siguientes tareas de la investigación:

1. Elaboración del marco teórico de la investigación relacionado con la gestión de tareas en los sistemas informáticos y las plataformas internacionales existentes vinculados al campo de acción.

2. Análisis de sistemas homólogos para identificar las buenas prácticas y requerimientos.

3. Desarrollo de los paquetes de interfaces de la herramienta de procesamiento de datos de neurociencias en el clúster de CNEURO.

4. Integración de las interfaces desarrolladas a la plataforma BrainSSys, para facilitar el uso del clúster.

5. Validación de los resultados obtenidos aplicando la estrategia de pruebas de la investigación.

Para el desarrollo de la presente investigación se emplearon los siguientes métodos científicos:

El método de modelación permitió la elaboración de diagramas y esquemas para modelar en detalle cada una de las fases del desarrollo de la herramienta de gestión de tareas en el clúster.

El método histórico lógico se utilizó para el análisis de otras soluciones similares como: CBRAIN, LORIS, entre otros. Además, se analizó la evolución de los sistemas de análisis de neurodatos y cómo se han modernizado para lograr la informatización de la gestión del procesamiento de datos de Neurociencias.

El método analítico sintético, en el análisis de la teoría y extracción de los principales conceptos de los elementos esenciales del objeto de estudio, así como sus componentes, características y las relaciones entre ellos. (Economipedia, 2023)

El método inductivo deductivo, en el análisis general sobre el objeto de estudio y su decantación hasta llegar al campo de acción, es decir, del conocimiento general a un conocimiento más particular. (Economipedia, 2023)

El método de análisis documental o literatura, en la identificación, recolección y análisis de los referentes sobre el tema en cuestión, así como los trabajos y aportes que servirán de sustento de los planteamientos hechos por el autor. (Economipedia, 2023)

Entre los métodos empíricos utilizados se tiene la revisión documental, observación simple y observación itinerante a partir de que el investigador fue un elemento activo en todo el proceso, la aplicación de estos métodos fue una regularidad durante todas las etapas de la investigación.

Con el desarrollo de la investigación se esperan los siguientes resultados:

Una herramienta que gestione el procesamiento de datos de neurociencias para la plataforma BrainSSys con interfaces graficas que facilite la navegación y utilización del proceso de lanzamiento de tareas en el clúster para agilizar el trabajo que desarrollan los especialistas de la institución.

CAPÍTULO I: FUNDAMENTOS Y REFERENTES TEÓRICO-METODOLÓGICOS SOBRE EL OBJETO DE ESTUDIO

El presente capítulo tiene como objetivo abordar los diferentes contenidos que brindan la base teórica y conceptual para el desarrollo de la investigación. Se valorarán de forma crítica las tendencias y tecnologías actuales, así como los antecedentes asociados al campo de acción, como lo son las neurociencias, la neuroinformática, entre otros derivados de la rama científica del estudio del cerebro. Se realizará un análisis de las características de los diversos repositorios de datos neurológicos existentes a nivel internacional y un acercamiento a los conceptos a abordar en la investigación. Esta revisión permitirá caracterizar adecuadamente el contexto y problema de estudio abordado en este trabajo.

I.1 Conceptos asociados al problema

La neurociencia es una disciplina científica que estudia el sistema nervioso y todos sus aspectos, tales como la estructura, función, desarrollo ontogenético y filogenético, bioquímica, farmacología y patología, y cómo sus diferentes elementos interactúan, dando lugar a las bases biológicas de la cognición y la conducta. La neurociencia trata de dar respuesta a una amplia gama de interrogantes acerca de cómo se organizan los sistemas nerviosos de los seres humanos y de otros animales, cómo se desarrollan y cómo funcionan para generar la conducta. Estas preguntas pueden explorarse usando las herramientas analíticas de la genética y la genómica, la biología molecular y la biología celular, la anatomía y la fisiología de los aparatos y sistemas, la filosofía, la biología conductual y la psicología. Por ejemplo, algunos neurocientíficos se centran en el sistema nervioso y su impacto en el comportamiento y las funciones cognitivas, mientras que otros investigan qué sucede con el sistema nervioso cuando las personas tienen trastornos neurológicos, psiquiátricos o del neurodesarrollo. (Soriano, 2023)

La neurociencia ha sido tradicionalmente clasificada como una subdivisión de la biología, pero en realidad, se trata de una ciencia interdisciplinaria relacionada estrechamente con otras disciplinas, como las matemáticas, la lingüística, la ingeniería, la informática, la química, la filosofía, la psicología o la medicina. Los neurocientíficos estudian los aspectos celular, funcional, evolutivo, computacional, molecular, celular y médico del sistema nervioso, por eso más que hablar de “neurociencia” en singular, cabe hablar de “neurociencias” en plural. (Rojas Porras, 2022)

El término neurociencia, brevemente definido por Mora y Sanguinetti como la disciplina que estudia el desarrollo, estructura, función, farmacología y patología del sistema nervioso, fue introducido en la lengua inglesa (*neuroscience*) entre finales de los 60 y principios de los 70. Por ello, de acuerdo con la opinión de E. G. Jones (2000), podemos decir que la neurociencia como tal es un fenómeno que se inscribe fundamentalmente en el siglo XX, a pesar de sus profundas raíces dentro del campo del conocimiento biomédico. Durante la última parte de este milenio, el estudio del cerebro se trasladó desde una posición periférica dentro de las ciencias biológicas y psicológicas hasta convertirse en este campo interdisciplinario denominado neurociencia que ahora ocupa una posición central en cada una de dichas disciplinas. Este giro tuvo lugar principalmente debido a que el estudio del cerebro se incorporó en un marco general de conocimiento que contaba, por un lado, con los avances de la biología celular y molecular y, por otro, con el surgimiento de la psicología como disciplina científica. Dentro de esta nueva línea, el alcance de la neurociencia fue capaz de abarcar desde el estudio de los genes y de las moléculas hasta la cognición y la propia mente del individuo. (Redolar-Ripoll, 2002)

El conjunto de herramientas que sirven para analizar e influir sobre el sistema nervioso del ser humano se conoce como neurotecnologías. Estas tecnologías incluyen simulaciones de modelos neuronales, computadores biológicos, aparatos para interconectar el cerebro con sistemas electrónicos y aparatos para medir y analizar la actividad cerebral. Tecnologías como el electroencefalograma, la tomografía, la resonancia magnética y algunas más especializadas, son entre las más utilizadas para el estudio. La neurotecnología ha alcanzado otros ámbitos que normalmente pasan desapercibidos y que van desde el desarrollo de fármacos para tratar alteraciones mentales**,** como la depresión, el insomnio o el déficit de atención, hasta el de tecnologías dedicadas a la rehabilitación neurológica después de accidentes cerebrovasculares o a la recuperación de la *audición* con los implantes cocleares. Y esto, como veremos más adelante, no ha hecho más que empezar. (Iberdrola, 2019)

Dentro de la neurociencia se sitúa la neuroinformática que mediante la aplicación de modelos computacionales y herramientas analíticas para el estudio y entendimiento del cerebro compartan datos y hallazgos de una manera estructurada y disciplinada. La neuroinformática viene a cubrir tres objetivos principales: el desarrollo de herramientas y bases de datos para el manejo y distribución de datos neurocientíficos a todos los niveles de análisis, el desarrollo de herramientas para análisis y modelado de datos neurocientíficos, y el desarrollo de modelos computacionales del sistema nervioso y procesos neuronales.

La neuroinformática está relacionada con la filosofía (teoría computacional de la mente), la psicología (teoría del procesamiento de la información), la informática (computación natural, computación bioinspirada), entre otras. La neuroinformática no se ocupa de la materia o la energía, por lo que puede verse como una rama de la neurobiología que estudia varios aspectos del sistema nervioso. El término neuroinformática parece usarse como sinónimo de informática cognitiva, descrito por *Journal of Biomedical Informatics* como dominio interdisciplinario que se centra en el procesamiento, los mecanismos y los procesos de la información humana dentro del contexto de la informática y las aplicaciones informáticas. (TechEdu: Definiciones de términos técnicos, 2023)

Un clúster de computadoras es una agrupación de múltiples sistemas informáticos conectados entre sí que funcionan como una sola computadora. Estos sistemas se pueden utilizar para realizar tareas complejas, como el procesamiento de grandes cantidades de datos, y se caracteriza por su alta eficiencia y escalabilidad. A diferencia de la computación en malla (*grid computing*), los clústeres de computadoras tienen a cada nodo realizando la misma tarea, controlada y planificada por software. El cómputo con clústeres surge como resultado de la convergencia de varias tendencias actuales que incluyen la disponibilidad de microprocesadores económicos de alto rendimiento y redes de alta velocidad, el desarrollo de herramientas de software para cómputo distribuido de alto rendimiento, así como la creciente necesidad de potencia computacional para aplicaciones que la requieran. (Diccionario Tecnológico, 2023)

High Performance Computing (HPC) hace referencia a la computación de alto rendimiento. El propio nombre indica que no se trata tanto de una tecnología definida con instrumentos específicos, sino más bien de procedimientos que utilizan o ponen a disposición el rendimiento y las capacidades de almacenamiento de los ordenadores. No hay criterios fijos para la esencia de la HPC, ya que cambia con el tiempo y se adapta a las nuevas tecnologías informáticas. En general, se puede decir que las soluciones HPC se utilizan para operaciones de cálculo complejas con cantidades muy grandes de datos o para el análisis, el cálculo y la simulación de sistemas y modelos. Por un lado, los procedimientos HPC pueden utilizarse en ordenadores individuales muy potentes. Sin embargo, por otro lado, es más frecuente encontrar la HPC en forma de nodos que forman superordenadores como clústeres. Estos superordenadores son capaces de realizar cálculos paralelos de alto rendimiento con varios recursos agregados. (Digital Guide IONOS, 2023)

Al contextualizar el origen y evolución histórica del campo neurológico y delimitar su enfoque multidimensional, incorporando perspectivas desde diversas ramas del conocimiento, se sientan las

premisas para la correcta comprensión del problema. La definición de conceptos afines como clústeres de computadoras y cómputo de alto rendimiento provee el sustento tecnológico requerido. De este modo, el lector cuenta con los elementos conceptuales básicos para interpretar adecuadamente los fines y alcances de la investigación.

I.2 Colaboración UCI-CNEURO

El Centro de Neurociencias de Cuba (CNEURO) es la mayor institución del Grupo BioCubaFarma dedicada al desarrollo de neurotecnologías y la investigación básica sobre la base de los principales problemas de salud mental de la población cubana. Con más de 200 investigadores y personal de apoyo que participan en la investigación básica y aplicada, y el desarrollo de nuevos productos, las investigaciones de CNEURO están orientadas a la aplicación médica y se ejecutan en coordinación con instituciones de los sistemas nacionales de salud y educación. Su principal objetivo es investigar y producir tecnologías de última generación para el diagnóstico y tratamiento de las enfermedades cerebrales. El centro es reconocido como un productor de dispositivos médicos avanzados en Cuba, enfocado en electroencefalografía, electromiografía, audiología y audífonos. (CNEURO, 2022)

Las áreas de especial interés son:

- La búsqueda de nuevos biomarcadores, enfoques de diagnóstico y moléculas terapéuticas para la enfermedad de Alzheimer.
- Desarrollo de nuevos métodos de neuroimagen.
- Desarrollo de tecnologías para el manejo del envejecimiento y los trastornos del neurodesarrollo.
- Desarrollo de dispositivos de neuroestimulación para la depresión y otros trastornos psiquiátricos.
- Desarrollo de tecnologías para el manejo de las deficiencias auditivas.
- Desarrollo de tecnologías basadas en neuroinformática para el análisis de datos de EEG y MRI en condiciones cerebrales normales y patológicas.
- Desarrollo de dispositivos médicos en respuesta a COVID-19

CNEURO es una institución de investigación que pertenece a varios consorcios de investigación internacionales como el Global Brain Consortium (GBC, 2023), la International Brain Initiative (IBI, 2023) y el Proyecto de mapeo cerebral de Cuba-China-Canadá. Además, tiene una fuerte colaboración académica con varios institutos y universidades extranjeras, como la Universidad de Ciencia y Tecnología

Electrónica de China en Chengdu (UESTC, 2023), la Universidad de Maastricht (Maastricht University, 2023), la Universidad de Aachen (RWTH-Aachen, 2023), la Universidad McGill (McGill University, 2023) en Canadá, entre otros. También ha participado en proyectos internacionales financiados por el Human Frontier Science Program (HFSP, 2023) y algunos proyectos para la Organización Internacional de Investigación del Cerebro (IBRO, 2023). A través de una entrevista al Dr. C Arturo Orellana, plantea que la variedad de datos de estudios neurofisiológicos en correspondencia a la aparición cada vez mayor de nuevas tecnologías, la diversidad de fuentes de datos existentes y la complejidad para su utilización en las investigaciones científicas son el motivo principal del desarrollo del proyecto BrainSSys.

I.3 Proyecto BrainSSys

Ecosistema de software neurocientífico orientado al almacenamiento, gestión, visualización y procesamiento de datos provenientes de estudios del cerebro humano. Desarrollada con tecnologías emergentes y escalables con múltiples ventajas para su aplicación en entornos heterogéneos.

- **DataBrain**: Repositorio para el almacenamiento y gestión de datos de neurociencias, posee funcionalidades para la información estadística y toma de decisiones. Incorpora herramientas para el manejo, seguridad de los datos y la actualización de sus bases de datos. Posee API's de servicios a otros sistemas.
- **PixelBrain**: Herramienta de visualización, análisis y procesamiento de datos de neurociencias. Compuesta por algoritmos y técnicas que propician la mejora de información diagnóstica y científica; y detección de elementos de interés investigativo en datos de neurociencias.
- **SyncBrain**: Herramienta para la conexión y transmisión bidireccional de datos y bases de datos con plataformas internacionales. Permite la regulación de la transmisión en función de la velocidad y carga de red.
- **ToolsBrain**: Conjunto de herramientas para el soporte y gestión adicional de datos de neurociencias. Propician la transformación y estandarización de datos de neurociencias, la gestión de colas de procesamiento en HPC, la modelación dinámica de Pipelines de datos, algoritmos de corregistro y fusión de datos espacio-temporales multimodales.

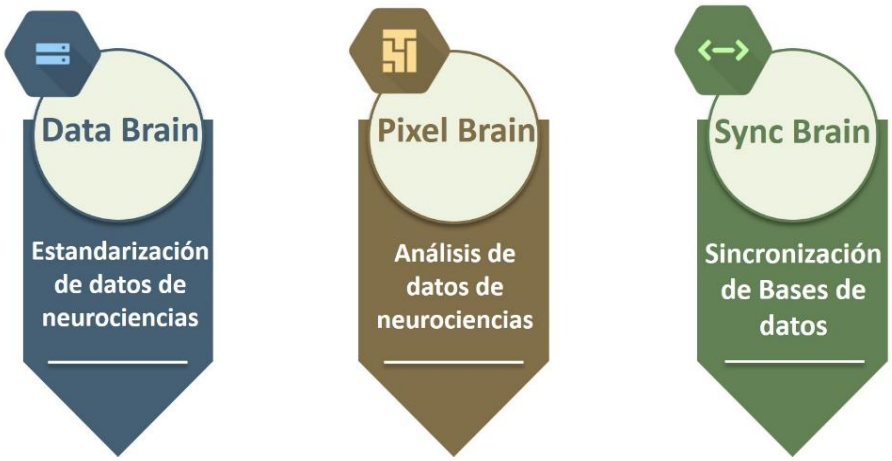


Figura 1: Sistemas internos de BrainSSys

Fuente: BrainSSys.

La información provista sobre el Centro de Neurociencias de Cuba y el sistema BrainSSys sitúa la investigación en un contexto aplicado concreto. Al describir las líneas de trabajo del CNEURO y caracterizar las funcionalidades de la herramienta objeto de estudio, se delimita claramente el escenario real al que se enfocarán los esfuerzos de diseño. Asimismo, al precisar detalles técnicos como las especificaciones del sistema, así como las necesidades de procesamiento de datos neurológicos, se sientan las bases para el correcto planteamiento de soluciones acordes a una problemática identificada en este ámbito específico.

I.4 Estado del arte de soluciones similares

Se han desarrollado plataformas para el procesamiento y análisis de datos de neurociencias producto de las alianzas entre diferentes países, patrocinados por proyectos y financiamientos internacionales; entre ellos CBRAIN, BIL (*Brain Image Library*) y LORIS (*Longitudinal Online Research and Imaging System*). Se procedió a realizar un análisis comparativo de las soluciones existentes más representativas para el procesamiento y gestión de datos neurocientíficos, considerando los siguientes criterios:

- Acceso a la información: Modalidades de acceso y tipos de datos neurocientíficos que cada plataforma permite gestionar.
- Arquitectura tecnológica: Plataformas y lenguajes de programación sobre los que se sustentan, aspectos de seguridad y manejo de recursos.

- Aplicabilidad: Población objetivo a la que van dirigidas, contexto institucional de pertenencia y posibilidades de personalización.

- Usabilidad: Facilidad de uso e integración con otros sistemas desde la perspectiva del usuario final.

- Mantenimiento y soporte: Políticas de actualización, corrección de errores y disponibilidad a largo plazo.

La Plataforma Canadiense de Investigación de Imágenes Cerebrales (CBRAIN) es una plataforma de investigación colaborativa basada en la web, desarrollada en respuesta a los desafíos planteados por la investigación de neuroimágenes con gran cantidad de datos y con uso intensivo de ordenadores. CBRAIN ofrece un acceso transparente a fuentes de datos remotas, sitios de computación distribuidos y una serie de herramientas de procesamiento y visualización dentro de un entorno controlado y seguro. (Sherif, y otros, 2014)

Longitudinal Online Research and Imaging System (LORIS) es un sistema modular y extensible de gestión de datos basado en la web que integra todos los aspectos de un estudio multicéntrico: desde la adquisición de datos heterogéneos (imagen, clínica, comportamiento y genética) hasta el almacenamiento, el procesamiento y, finalmente, la difusión. Proporciona una plataforma segura, fácil de usar y racionalizada para automatizar el flujo de los ensayos clínicos y los estudios multicéntricos complejos. Una organización interna centrada en el sujeto permite a los investigadores capturar y posteriormente extraer toda la información, longitudinal o transversal, de cualquier subconjunto del estudio. (Das, Zijdenbos, Harlap, Vins, & Evans, 2011)

La Biblioteca de imágenes cerebrales (BIL) es un recurso público nacional que permite a los investigadores depositar, analizar, extraer, compartir e interactuar con grandes conjuntos de datos de imágenes cerebrales. BIL abarca la deposición de conjuntos de datos, la integración de conjuntos de datos en un sistema de búsqueda accesible en la web. La redistribución de conjuntos de datos y un enclave computacional para permitir a los investigadores procesar conjuntos de datos en el lugar y compartir conjuntos de datos restringidos y previos al lanzamiento. (BIL, 2023)

Análisis comparativo a nivel de plataforma:

Tabla 1: Análisis comparativo de las plataformas.

Fuente: los autores.

	CBRAIN	**LORIS**	**BIL**
Acceso al repositorio	SSL, SSH	SSL	SSH, Bridges-2, Neocórtex, Sistema VM
Acceso a los datos	Inicio de sesión.	Inicio de sesión.	Inicio de sesión.
Tipos de datos	Archivos planos, imágenes y metadatos textuales	Archivos planos, imágenes, metadatos textuales y visualizaciones 3D	Imágenes (tiff, jpg-2000), Datos de neuronas(swc)
Tecnologías / Lenguajes	Ruby y Rails	Linux, Apache, MySQL, PHP (LAMP)	No especificado.
Código abierto	Si	Si	No
Organización/ Institución a la que pertenece	Canadian Open Neuroscience Platform	Canadian Open Neuroscience Platform	Iniciativa BRAIN, Universidad de Pittsburgh

Las principales funcionalidades comunes entre las plataformas CBRAIN, LORIS y BIL son: (Dokmanic, Parhizkar, Raninen, Vetterli, & Schlögel, 2019) (Poldrack, y otros, 2020)

- Almacenamiento centralizado y gestión de datos neurocientíficos como imágenes cerebrales, datos clínicos, resultados de pruebas, etc.

- Interfaz web de acceso que permite carga, visualización y búsqueda básica de información alojada.
- Procesamiento distribuido de grandes conjuntos de datos a través de recursos computacionales remotos.
- Implementación de flujos de trabajo y pipelines de procesamiento parametrizables.
- Soporte a estándares y estandarización de metadatos asociados a los datos.
- Control de acceso seguro a la información mediante perfiles y roles de usuarios.
- Administración centralizada de proyectos y estudios multicéntricos longitudinales.
- Almacenamiento a largo plazo de datos e imágenes en la nube.
- Interfaz de monitoreo de recursos y ejecución de tareas distribuidas.
- Publicación y redistribución controlada de conjuntos de datos de forma segura.

Las plataformas antes descritas son válidas en el entorno para el que fueron desarrolladas, sin embargo, no son útiles para Cuba. Esto se debe a que pertenecen o están asociados a instituciones u organizaciones extranjeras por lo que no cumplen con las políticas de soberanía del país y no todas son de código abierto. De ahí que no sea posible dar soporte, ni personalizar a las características propias de CNEURO. Por eso se decide desarrollar una herramienta para la gestión del procesamiento de tareas, tomando en cuenta un grupo de características deseadas que presentan las plataformas anteriores en aspectos como:

- Tecnologías: Los protocolos que se emplearían, el lenguaje, los diferentes mecanismos de comunicación y manejo de archivos.
- Acceso a la información: Los diferentes neurodatos que serán manejados por el sistema y la forma en el cual son compartidos.

La realización del procesamiento de datos neurocientíficos conlleva una elevada complejidad técnica, pues los investigadores se ven obligados a dominar habilidades informáticas especializadas como la manipulación de servidores Linux mediante línea de comandos para poder dictar instrucciones y llevar a cabo los procesamientos de manera distribuida en el clúster. Esto representa una traba para los científicos sin formación de ciencias de la computación, quienes deben invertir mucho tiempo en

adquirir conocimientos ajenos a su área para poder aprovechar plenamente las capacidades computacionales disponibles.

A nivel de solución:
Con respecto a las soluciones presentadas, no todas cumplen con las necesidades de la institución debido a que están especializados a una tarea genérica, no están integradas en una plataforma web con facilidad de acceso, no son intuitivas y tampoco son fáciles de utilizar incluso para los usuarios más expertos necesitando preparación técnica previa, viendo como resultado posibles fallas que se están cometiendo actualmente en el clúster. Algunas ya están obsoletas por lo que no se realizaran mejoras, arreglo de fallos ni ningún tipo de parches de seguridad. Sin embargo, algunas de las funcionalidades son factibles para la solución, tales como servicios web que permiten mejor accesibilidad del sistema, interfaces de gestión de recursos y ejecución de tareas, distribución de datos segura y control de acceso seguro mediante roles, y almacenamiento a largo plazo en la nube.

I.5 Ambiente de desarrollo

Las metodologías de desarrollo de software son un conjunto de procedimientos, técnicas y ayudas a la documentación para el desarrollo de productos de software. Van indicando paso a paso todas las actividades a realizar para lograr el producto informático deseado.

La metodología escogida para el proceso de desarrollo es SCRUM debido a que esta herramienta tributa al proyecto BrainSSys, el cual utiliza dicha metodología, la cual tiene un 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.

SCRUM se puede dividir de forma general en 3 fases, las cuales se entienden como reuniones: Planificación del Backlog donde se define el documento que refleja los requisitos del sistema organizado por prioridades, además de la planificación del Sprint 0, en la que se decide los objetivos y el trabajo para esta iteración, el objetivo más importante de esta fase es obtener el Sprint Backlog, siendo esta la lista de tareas a realizar; Seguimiento del Sprint, en esta fase se realizan reuniones diarias para evaluar el avance de las tareas; y por último Revisión del Sprint, en esta fase se realiza una revisión del incremento generado, se presentan los resultados finales y una versión. (Trigas Gallego, 2012)

Los elementos que forman Scrum son:

- ***Product Backlog*** (Pila de Productos): Lista de necesidades del cliente.
- ***Sprint Backlog*** (Pila de Sprint): Lista de tareas que se realizan en un sprint.
- **Incremento**: Parte añadida o desarrollada en un sprint, es una parte terminada y totalmente operativa.

Además de los elementos mencionados anteriormente, Scrum genera una serie de productos de trabajo que impulsan la entrega incremental del producto y facilitan la colaboración y la transparencia en el equipo. (Trigas Gallego, 2012)

Estos productos incluyen:

Historias de Usuario: Son descripciones breves y centradas en los usuarios de las funcionalidades deseadas del producto. Las historias de usuario capturan los requisitos desde la perspectiva del usuario final y sirven como base para la planificación y ejecución del trabajo dentro de los sprints.

Prototipos o Mockups: Son representaciones visuales o interactivas de las funcionalidades propuestas del producto. Los prototipos permiten a los interesados y al equipo comprender mejor cómo se verá y se comportará el producto final, facilitando la validación temprana y la retroalimentación.

Incrementos de Producto: Scrum se enfoca en la entrega iterativa e incremental del producto. En cada sprint, el equipo desarrolla y entrega incrementos funcionales del producto, que son versiones mejoradas o nuevas características que se agregan al producto principal.

Documentación de Sprint: A medida que se avanza en los sprints, se genera documentación relevante para el desarrollo y entendimiento del producto. Esto puede incluir documentación técnica, manuales de usuario, informes de pruebas, entre otros.

Reportes de Seguimiento y Métricas: Durante el seguimiento del sprint, se generan reportes y métricas que ayudan a evaluar el progreso del equipo. Estos informes pueden incluir el gráfico de Burndown, que muestra la cantidad de trabajo restante a lo largo del tiempo, y otros datos que brindan visibilidad sobre la eficiencia y la calidad del trabajo realizado.

Se considera a SCRUM como la metodología más apropiada debido a que favorece la colaboración, el trabajo en equipo, la gestión flexible de cambios y la entrega continua de valor propuesta por el enfoque ágil. Son aspectos fundamentales para la entrega rápida del proyecto, el cual CNEURO necesita en

primeras instancias para su uso y asimilar la herramienta, que puede suponer un gran avance para sus tareas.

I.6 Tecnologías

I.6.1 Lenguajes

Un lenguaje de programación es un lenguaje formal diseñado para realizar procesos que pueden ser llevados a cabo por máquinas como las computadoras. Pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana. Está formado por un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones.

Para la selección del lenguaje se tuvo en cuenta, Python, es un lenguaje de programación de alto nivel y de propósito general que se utiliza para una amplia variedad de aplicaciones, desde el desarrollo de aplicaciones web hasta la inteligencia artificial y la ciencia de datos. (Python Docs, 2023) Directamente la herramienta de procesamientos de datos del clúster a la cual estará vinculada, utiliza este lenguaje, en lo que se conoce como *backend*, es el encargado de realizar las funcionalidades de manipulación de datos y ejecución de tareas en el clúster.

I.6.2 Marco de trabajo

Un marco de trabajo o *framework* es un diseño abstracto orientado a objetos para un determinado tipo de aplicación, es un patrón arquitectónico que proporciona una plantilla extensible para un tipo específico de aplicaciones. Según un framework o "esquema" es un subsistema expandible de un conjunto de servicios, es un conjunto cohesivo de interfaces y clases que colaboran para proporcionar los servicios de la parte central e invariable de un subsistema lógico.

Para la selección del marco de trabajo, en el servidor (*backend*), se tuvo en cuenta FastAPI, es un *framework* para construir API’s de forma sencilla y rápida con Python que en los últimos tiempos se ha vuelto muy popular. Actualmente es considerado como uno de los *frameworks* basados en Python más rápidos y, además, proporciona también una gran velocidad a la hora de abordar el desarrollo de una API al incorporar, entre otras cosas, validación de datos y de documentación de forma automática, lo cual lo convierte en un candidato ideal para realizar el *backend* de cualquier aplicación web. (Ramírez, 2023)

Sus características principales son:

- **Rapidez**: Alto rendimiento, a la par con NodeJS y Go (gracias a Starlette y Pydantic). Uno de los *frameworks* de Python más rápidos.
- **Rápido de programar**: Incrementa la velocidad de desarrollo entre 200% y 300%. *
- **Menos errores**: Reduce los errores humanos (de programador) aproximadamente un 40%.
- **Intuitivo**: Gran soporte en los editores con auto completado en todas partes. Gasta menos tiempo *debugging*.
- **Fácil**: Está diseñado para ser fácil de usar y aprender. Gastando menos tiempo leyendo documentación.
- **Corto**: Minimiza la duplicación de código. Múltiples funcionalidades con cada declaración de parámetros. Menos errores.
- **Robusto**: Crea código listo para producción con documentación automática interactiva.
- **Basado en estándares**: Basado y totalmente compatible con los estándares abiertos para API's: OpenAPI (conocido previamente como Swagger) y JSON Schema.

Para la selección del marco de trabajo o librerías, en el cliente (*frontend*), se tuvo en cuenta Vue.js, un framework de JavaScript de código abierto utilizado para crear aplicaciones web de una sola página e interfaces de usuario interactivas. Se enfoca en la creación de componentes reutilizables, lo que significa que los desarrolladores pueden crear componentes que se pueden utilizar en diferentes partes de una aplicación. Esto permite una mayor modularidad y reutilización de código, lo que a su vez facilita el mantenimiento y la escalabilidad de las aplicaciones. (Vue.js, 2023)

Al igual que otros *frameworks* de JavaScript, Vue.js utiliza el enlace de datos bidireccional, lo que significa que los cambios realizados en la vista se reflejan automáticamente en el modelo y viceversa. Además, Vue.js utiliza una sintaxis clara y concisa que facilita su aprendizaje y uso.

Vue.js también cuenta con una amplia variedad de herramientas y bibliotecas, como Vue Router para la gestión de rutas y Vuex para la gestión del estado de la aplicación. Además, Vue.js es altamente personalizable y se puede integrar fácilmente con otras bibliotecas y *frameworks* de JavaScript.

I.6.3 API

El término API es una abreviatura de *Application Programming Interfaces*, que en español significa interfaz de programación de aplicaciones. Se trata de un conjunto de definiciones y protocolos que se utiliza para desarrollar e integrar el software de las aplicaciones, permitiendo la comunicación entre dos aplicaciones de software a partir de un conjunto de reglas. Un API como una especificación formal que establece cómo un módulo de un software se comunica o interactúa con otro para cumplir una o muchas funciones. Todo dependiendo de las aplicaciones que las vayan a utilizar, y de los permisos que les dé el propietario de la API a los desarrolladores de terceros. (Fernández, 2019) El elemento principal de la solución será un servicio en forma de API, que permita luego ser consumida por la Interfaz Web de la Plataforma BrainSSys.

Nube

Una nube es una infraestructura de computación distribuida que proporciona acceso a recursos informáticos, almacenamiento y software a través de Internet. En lugar de tener que mantener y ejecutar aplicaciones y servicios en equipos locales, los usuarios pueden acceder a ellos a través de una red en línea. Las plataformas en la nube proporcionan una forma rentable, flexible y escalable de acceder a recursos informáticos y almacenamiento.

Nextcloud, por otro lado, es una plataforma de nube privada y de código abierto que permite a los usuarios almacenar y compartir archivos, así como acceder a aplicaciones y servicios en línea. Nextcloud se ejecuta en servidores privados o en la nube, lo que significa que los usuarios pueden controlar dónde se almacenan sus datos y quién tiene acceso a ellos. Ofrece una amplia variedad de características, incluyendo la sincronización de archivos, la colaboración en línea, la gestión de calendarios y contactos, y la integración de aplicaciones de terceros. Además, es altamente personalizable y se puede ampliar mediante la instalación de aplicaciones adicionales. (Nextcloud, 2023)

I.6.4 Plataforma de despliegue

Una plataforma de despliegue es un conjunto de herramientas y servicios que permiten a los desarrolladores implementar aplicaciones en la nube. El despliegue *cloud* es el proceso de implementar una aplicación en uno o más modelos de alojamiento que utilizan la nube, como software como servicio (SaaS), plataforma como servicio (PaaS) y/o infraestructura como servicio (IaaS).

Docker es una plataforma de virtualización de contenedores que permite a los desarrolladores empaquetar sus aplicaciones y sus dependencias en contenedores portátiles y autónomos que se pueden ejecutar en cualquier sistema operativo. Docker se basa en la tecnología de contenedores de Linux para proporcionar una forma eficiente y confiable de empaquetar y ejecutar aplicaciones en diferentes entornos. Una de las principales ventajas de Docker es que permite a los desarrolladores crear contenedores que se pueden ejecutar en cualquier sistema operativo, lo que facilita la portabilidad y la implementación de aplicaciones en diferentes entornos. Además, proporciona una forma fácil de automatizar el proceso de implementación y configuración de aplicaciones, lo que ahorra tiempo y reduce los errores. Otra ventaja es que permite a los desarrolladores crear entornos de desarrollo y prueba reproducibles y aislados, lo que significa que pueden crear y probar aplicaciones en entornos que son idénticos al entorno de producción, lo que ayuda a reducir los errores y a mejorar la calidad del software. (Docker, 2023)

Docker también es altamente escalable, lo que significa que los contenedores se pueden escalar vertical u horizontalmente para satisfacer las demandas de las aplicaciones en tiempo real. Además, Docker proporciona una forma fácil de administrar y supervisar los contenedores, lo que facilita el mantenimiento y la gestión de la infraestructura de la aplicación.

I.6.5 Motor de Base de Datos

Un motor de base de datos es el componente de software subyacente que un sistema de administración de la base de datos (SGBD) utiliza para crear, leer, actualizar y eliminar (CRUD) datos de una base de datos. Los motores de bases de datos son programas específicos que se utilizan para crear y administrar bases de datos.

PostgreSQL es un potente SGBD objeto relacional de código abierto, tiene soporte completo para claves foráneas, vistas y disparadores. Incluye la mayoría de los tipos de datos de SQL incluyendo *integer*, *boolean* y *varchar* entre otros. También soporta el almacenamiento de objetos binarios grandes como imágenes, sonido o video. (Group PostgreSQL Global Development., 2022)

I.6.6 Herramientas de modelado

Visual Paradigm es una herramienta CASE multiplataforma que contribuye al desarrollo de sistemas de software fiables, mediante un enfoque orientado a objetos. Soporta el ciclo completo de desarrollo de software y permite su documentación en diferentes formatos, empleando UML como lenguaje de

modelado (Visual Paradigm, 2022). Con el uso de esta herramienta se desarrollan los diagramas de la arquitectura de software, mapa de navegación y diagrama de despliegue.

Conclusiones del capítulo

Luego de realizar un análisis sobre los conceptos y herramientas relacionadas con el procesamiento de datos neurocientíficos, así como estudiar a detalle el contexto y necesidades específicas del CNEURO, es posible establecer las siguientes conclusiones clave:

La investigación permitió comprender el problema a resolver y su entorno, mediante la revisión de temas como neurociencias, neurotecnología y arquitecturas de clúster. Asimismo, la evaluación comparativa de soluciones existentes determinó que actualmente no existe una opción que cubra todos los requerimientos del centro de forma integral.

Por otro lado, el estado del arte facilitó precisar los componentes fundamentales que debe considerar una futura plataforma acorde a los objetivos institucionales. Además, se definió que SCRUM es la metodología óptima para guiar el desarrollo ágil de acuerdo a dichos objetivos. Finalmente, las tecnologías seleccionadas como Python, FastAPI, Vue.js y PostgreSQL, darán sustento técnico robusto y flexible a la propuesta de solución. Del mismo modo, herramientas como Nextcloud y Docker permitirán su operación en la nube de forma escalable y sencilla.

CAPÍTULO II: DISEÑO DE LA SOLUCIÓN PROPUESTA AL PROBLEMA CIENTÍFICO

En el presente capítulo se describe los elementos que se tuvieron en cuenta para el desarrollo de paquete de interfaces de la herramienta para el clúster de CNEURO. Se presentan los conceptos asociados al modelo conceptual y la representación formal del mismo. Se identifican los requisitos funcionales y no funcionales. La implementación de la solución se basa en los principios y reglas de la metodología SCRUM, utilizando las tecnologías y herramientas definidas. La metodología empleada se divide en fases y actividades dentro de estas, generándose un grupo de artefactos que permiten especificar los elementos fundamentales de la herramienta.

II.1 Descripción del problema

El Centro de Neurociencias de Cuba (CNEURO) es una institución de investigación y desarrollo dedicado a la investigación de diferentes campos en Neurociencia. El HPC o Computadora de Alto Rendimiento de CNEURO tiene como finalidad mejorar el rendimiento en la ejecución de algoritmos que requieran grandes cantidades de tiempo-máquina, y, por ende, facilitar la obtención de resultados en los procesos investigativos.

Actualmente, todos los procesos con el clúster se realizan por medio de una consola, donde los especialistas tienen que especificar manualmente los parámetros de ejecución de las tareas. En ocasiones se debe repetir el proceso debido a errores en las instrucciones lo cual atrasa la planificación de la investigación.

II.1 Propuesta de solución

La propuesta de solución es el desarrollo del paquete de interfaces para el consumo de servicios del clúster a través de la nube de la plataforma BrainSSys. Es una solución tecnológica moderna y escalable, que permitirá a los especialistas de CNEURO acceder y manipular los datos de la plataforma de una manera más eficiente y efectiva.

Para el desarrollo de las interfaces de usuario, se utilizará los *frameworks* de FastAPI y Vue.js, que es una herramienta de programación en JavaScript que permite la creación de interfaces de usuario dinámicas y responsivas. La solución de *frontend* se estructurará en componentes, que serán reutilizables y modularizados, lo que facilitará su mantenimiento y desarrollo. Además, se utilizarán librerías externas y complementos, como Vuetify, que es una librería de componentes visuales que permite la creación de interfaces modernas y estilizadas.

La solución se desplegará en un contenedor Docker, que permite la empaquetación de la aplicación y sus dependencias en un contenedor aislado que puede ser ejecutado en diferentes ambientes y sistemas operativos. Para ello, se utilizará un archivo Dockerfile que describirá los pasos necesarios para construir la imagen del contenedor, y se utilizarán herramientas como Docker Compose para simplificar el despliegue y la gestión de la infraestructura.

En cuanto a la nube de Nextcloud, se utilizará como plataforma para el despliegue de la solución. Nextcloud es una plataforma de almacenamiento en la nube que ofrece servicios de alojamiento de archivos, colaboración y sincronización. Para el despliegue, se utilizarán los servicios de contenedores y orquestación de la nube de Nextcloud, que permiten la creación y gestión de contenedores Docker de manera automatizada y escalable.

En cuanto a los servicios de BrainSSys, se utilizarán protocolos de comunicación estándar como HTTPS para la interacción entre la interfaz y los servicios de la nube. HTTPS es una variante segura del protocolo HTTP, que utiliza una capa adicional de cifrado SSL/TLS para proteger las comunicaciones entre el cliente y el servidor. La utilización de HTTPS proporciona una mayor seguridad y confidencialidad en las comunicaciones, ya que el cifrado SSL/TLS protege los datos transmitidos entre el cliente y el servidor frente a posibles ataques de interceptación o manipulación. Además, HTTPS ayuda a garantizar la autenticidad del servidor, ya que utiliza certificados digitales emitidos por autoridades de certificación confiables. Para implementar la comunicación mediante HTTPS, se deberán realizar algunos ajustes en la configuración de los servicios de la nube de BrainSSys, como la instalación y configuración de un certificado SSL/TLS emitido por una autoridad de certificación confiable. En el *frontend*, se deberá utilizar una librería de manejo de solicitudes HTTP que permita la realización de solicitudes mediante HTTPS, como Axios con la opción de configurar el protocolo de comunicación.

II.2 Requisitos de software

Los requisitos para un sistema son descripciones de lo que el sistema debe hacer: el servicio que ofrece y las restricciones en su operación. Tales requerimientos reflejan las necesidades de los clientes por un sistema que atienda cierto propósito, como sería controlar un dispositivo, colocar un pedido o buscar información. Al proceso de descubrir, analizar, documentar y verificar estos servicios y restricciones se le llama ingeniería de requerimientos. (Sommerville, 2011)

II.2.1 Requisitos funcionales

Un requisito funcional es una capacidad o condición que el sistema debe cumplir. Por lo general, estos deben incluir funciones desempeñadas por pantallas específicas, descripciones de los flujos de trabajo a ser desempeñados por la herramienta y otros requerimientos de negocio, cumplimiento, seguridad u otra índole. (Sommerville, 2011)

En el análisis realizado, se identificaron los siguientes requisitos funcionales:

Tabla 2: Requisitos Funcionales

Fuente: Los Autores

ID	Nombre	Importancia	Descripción
RF1	Autenticar usuario	70	El usuario debe autenticarse para comprobar permisos de acceso.
RF2	Registrar usuario	70	Insertar usuarios en la herramienta.
RF3	Modificar usuario	70	Actualizar la información referente a un usuario especificado.
RF4	Visualizar detalles del usuario	70	Mostrar información referente a un usuario especificado.
RF5	Eliminar usuario	70	Remover un usuario especificado de la herramienta.
RF6	Listar usuarios	70	Visualizar todos los usuarios de la herramienta.
RF7	Registrar servidor	100	Insertar un servidor en la herramienta.
RF8	Modificar servidor	90	Modificar un servidor especificado.
RF9	Visualizar detalles del servidor	90	Mostrar detalles de un servidor especificado.
RF10	Eliminar servidor	80	Remover un servidor de la herramienta.
RF11	Listar servidores	90	Visualizar todos los servidores de la herramienta.
RF12	Registrar script	100	Insertar un script en la herramienta.
RF13	Modificar script	90	Actualizar la información referente un script en la herramienta.
RF14	Eliminar script	90	Remover un script de la herramienta.
RF15	Listar scripts	90	Visualizar todos los scripts insertados previamente.

RF16	Visualizar detalles del script	90	Mostrar los detalles de un script insertado previamente.
RF17	Concatenar programas del script	100	Insertar más programas a un script.
RF18	Registrar programa	100	Insertar un programa en la herramienta.
RF19	Modificar programa	90	Actualizar la información referente a un programa.
RF20	Visualizar detalles	90	Mostrar los detalles de un programa especificado.
RF21	Eliminar programa	80	Remover de la herramienta un programa especificado.
RF22	Listar programas	100	Visualizar todos los programas insertados previamente.
RF23	Ejecutar tarea	100	Ejecutar una tarea previamente insertada.
RF24	Obtener tarea	90	Visualizar una tarea.
RF25	Detener tarea	100	Detener la ejecución de una tarea.
RF26	Comprobar tarea	90	Verificar el estado de una tarea especificada.
RF27	Obtener registro de la tarea	100	Exportar fichero con la información resultante de la tarea.
RF28	Obtener registro de errores de la tarea	80	Exportar fichero con los errores resultantes de una tarea.

II.2.1 Requisitos no funcionales

Los requisitos no funcionales (RNF), son requisitos que imponen restricciones en el diseño o la implementación como restricciones en el diseño o estándares de calidad. Son propiedades o cualidades que el producto debe tener. (Pressman, 2010)

- Seguridad:

  RNF1. La información que se maneje se mantendrá segura y confidencial, evitando el acceso no autorizado y la divulgación.

  RNF2. La seguridad estará regida por un mecanismo de control de acceso basado en roles, convenientemente asignados a los usuarios de la herramienta al momento de su creación.

  RNF3. Disponer de un mecanismo de autenticación de usuarios seguro.

- Eficiencia:

RNF4. El tiempo de respuesta de la herramienta no debe exceder de 2 segundos según *Google Page Speed Insight.* (Google, 2023)

- Usabilidad:

  RNF5. Debe tener una interfaz amigable e intuitiva, que pueda ser usada por cualquier usuario sin conocimientos especiales.

  RNF6. Los mensajes para interactuar con los usuarios y los de error deben ser lo suficientemente informativos, en idioma español y no deben revelar información interna.

II.2.2 Historias de Usuario

Las historias de usuario de Scrum son el elemento mínimo que conforma un proyecto en metodología ágil**.** Condensan los requisitos del cliente, lo definen y ayudan a comprender al equipo, el por qué están construyendo. Es uno de los elementos principales a definir, antes de comenzar un Sprint en esta metodología ágil de gestión de proyectos. Una historia de usuario bien redactada, permitirá definir los beneficios que traerá el producto a su público objetivo.

Dentro del proceso Scrum, la vinculación con el cliente comienza con una reunión donde se definen los requisitos que el cliente espera obtener del producto o servicio. Cada una de estas funciones o requisitos se describen en historias de usuario. Entonces, las historias de usuario es una descripción de una función que deberá tener el producto final (ya sea o no un software). Esta descripción está redactada en forma natural y sencilla, en lenguaje informal. Debe resumir allí, la idea de lo que espera obtener el cliente, en cada función del producto. Es decir, que debe incluir la perspectiva del usuario final. (Mancuzo, 2021)

En el marco de la investigación fueron definidos los siguientes parámetros a tener en cuenta en las historias de usuario:

- **Número:** Número asignado a la historia de usuario.
- **Nombre:** Nombre de la historia de usuario.
- **Perfil (Yo como):** Rol del usuario final.
- **Necesidad (Quiero):** El objetivo que tiene la función de software para el usuario final.
- **Propósito (Para):** El objetivo de la experiencia del usuario final con la función de software.
- **Criterios de aceptación:** Aspectos importantes de interés para el usuario final.

Para la descripción de requisitos funcionales de la propuesta de solución se define una historia de usuario para cada uno de ellos. En la Tabla 3 se muestra la HU correspondiente al requisito funcional

Registrar servidor. En el Anexo 1: Historias de Usuario pueden encontrarse las historias de usuario restantes implementadas en el desarrollo de la herramienta.

Tabla 3: Historia de Usuario: Registrar Servidor

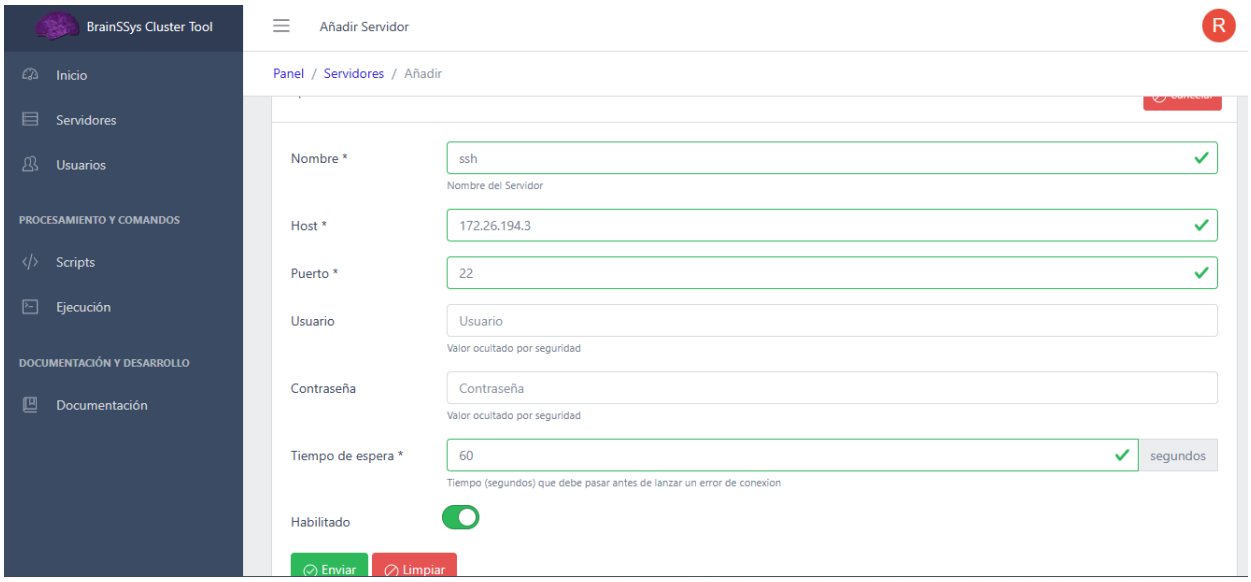
Fuente: los autores.

HU07 – Registrar servidor	
Yo como:	Especialista
Quiero:	Registrar nuevos servidores en la herramienta
Para:	Establecer una comunicación con él

Criterios de aceptación

- El campo nombre es requerido, de no ser insertado notificarlo mediante un mensaje.
- El campo host es requerido, de no ser insertado notificarlo mediante un mensaje.
- El campo puerto es requerido, de no ser insertado notificarlo mediante un mensaje. Debe de ser un valor entero entre 1 y 65535.
- El campo usuario es requerido, de no ser insertado notificarlo mediante un mensaje.
- El campo contraseña es requerido, de no ser insertado notificarlo mediante un mensaje.
- El campo *timeout* es requerido, de no ser insertado notificarlo mediante un mensaje
- Notificar al usuario en caso de que el proceso falle

Prototipo de interfaz de la historia de usuario



En el marco de este proyecto, se definieron 28 historias que describen cada uno de los requisitos funcionales identificados para la herramienta propuesta. Cada historia contiene los elementos standard recomendados como el perfil del usuario, la necesidad, el propósito y los criterios de aceptación. Esto ayudará a que el equipo comprenda fehacientemente qué funcionalidad se espera desarrollar en cada iteración, garantizando que se entregue valor al usuario final en cada incremento.

II.2.3 Planificación del Backlog

La planificación constituye la fase inicial del producto donde se discute el negocio, tiene el objetivo de tomar decisiones que impacten positivamente al producto y se especifica el propósito del proyecto. Se determina el alcance. Se comienza la creación del Backlog del producto para que el sprint siguiente contenga elementos suficientes que faciliten el trabajo. (Trigas Gallego, 2012)

II.2.4 Product Backlog

Es el inventario en el que se almacenan todas las funcionalidades o requisitos en forma de lista priorizada. Estos requisitos son los que tiene el producto y los cuales pueden incrementar en sucesivas iteraciones. (Trigas Gallego, 2012)

La Pila de Producto contiene los siguientes campos:

- ID: Identificador único, simplemente un número auto-incremental.
- Importancia: Ratio de importancia que el cliente da al requisito. Por ejemplo, 10. o 150. Más alto = más importante.
- Estimación Inicial: Valoración inicial del equipo referente a cuánto trabajo es necesario para implementar la funcionalidad, comparada con otras funcionalidades. La unidad está definida por “puntos” y usualmente corresponde a “días x persona ideales”. Lo importante no es que las estimaciones absolutas sean correctas (es decir, que un requisito de 2 puntos deba durar 2 días), sino que las estimaciones relativas sean correctas (es decir, que un requisito de 2 puntos debería durar la mitad que una funcionalidad de 4 puntos).

Tabla 4: Product Backlog.

Fuente: los autores.

ID	**Estimación Inicial**
RF1	2
RF2	2
RF3	2

RF4	2
RF5	2
RF6	2
RF7	2
RF8	2
RF9	2
RF10	2
RF11	2
RF12	6
RF13	6
RF14	2
RF15	2
RF16	2
RF17	8
RF18	8
RF19	2
RF20	2
RF21	2
RF22	2
RF23	6
RF24	2
RF25	4
RF26	4
RF27	4
RF28	4

II.2.5 Sprint Backlog

El Sprint Backlog es la lista de tareas y objetivos que el equipo define para cada Sprint de acuerdo al Product Backlog. Al comienzo de cada Sprint, el equipo se reúne para planificar qué actividades realizará en ese período. Las tareas se descomponen en elementos más pequeños, se estiman los tiempos requeridos y se asignan responsables. De esta forma, el trabajo del Sprint queda dividido en unidades manejables, permitiendo que cada integrante se enfoque en distintas responsabilidades. Las tareas

deben poder completarse en un rango de 4 a 16 horas para asegurar su viabilidad dentro del tiempo box del Sprint.

Gracias a esta planificación, es posible llevar un seguimiento preciso sobre el avance; en las revisiones diarias se verifica qué elementos del Sprint Backlog ya fueron terminados, cuáles están en progreso y cuáles aún no iniciaron. De esta manera, el equipo puede identificar desvíos tempranamente y reaccionar en consecuencia, adaptando el Sprint Backlog para maximizar el valor entregado al final del período. En conjunto, estas prácticas incrementan la transparencia y previsibilidad del trabajo. (Trigas Gallego, 2012)

Tabla 5: Product Backlog.

Fuente: Los Autores.

Sprint 0				
ID	Tarea	Asignada a	Estado	Tiempo
HU12-T1	Diseño e implementación de clases POCO (*Plain Old CLR Objects*).	Kevin Castillo Pérez	Terminada	2
HU13-T2	Migración a la base de datos.	Kevin Castillo Pérez	Terminada	2
HU12-T3	Implementación de los repositorios para el acceso a datos.	Kevin Castillo Pérez	Terminada	4
HU12-T4	Implementación de la lógica de la funcionalidad.	Kevin Castillo Pérez	Terminada	4
HU12-T5	Creación de los *endpoints*.	Kevin Castillo Pérez	Terminada	2
HU12-T6	Diseño e implementación de los DTO. (*Data Transfer Object*)	Kevin Castillo Pérez	Terminada	2
HU12-T7	Diseño del prototipado de la interfaz.	Kevin Castillo Pérez	Terminada	4
HU12-T8	Implementación de la funcionalidad en el cliente web.	Kevin Castillo Pérez	Terminada	6
HU7-T1	Diseño e implementación de clases POCO (*Plain Old CLR Objects*).	Kevin Castillo Pérez	Terminada	2
HU7-T2	Migración a la base de datos.	Kevin Castillo Pérez	Terminada	2

HU7-T3	Implementación de los repositorios para el acceso a datos.	Kevin Castillo Pérez	Terminada	4
HU7-T4	Implementación de la lógica de la funcionalidad.	Kevin Castillo Pérez	Terminada	4
HU7-T5	Creación de los *endpoints*.	Kevin Castillo Pérez	Terminada	2
HU7-T6	Diseño e implementación de los DTO. (*Data Transfer Object*)	Kevin Castillo Pérez	Terminada	2
HU7-T7	Diseño del prototipado de la interfaz.	Kevin Castillo Pérez	Terminada	4
HU7-T8	Implementación de la funcionalidad en el cliente web.	Kevin Castillo Pérez	Terminada	6
HU13-T1	Implementación de los repositorios para el acceso a datos.	Kevin Castillo Pérez	Terminada	4
HU13-T2	Implementación de la lógica de la funcionalidad.	Kevin Castillo Pérez	Terminada	4
HU13-T3	Creación de los *endpoints*.	Kevin Castillo Pérez	Terminada	2
HU13-T4	Diseño e implementación de los DTO. (*Data Transfer Object*)	Kevin Castillo Pérez	Terminada	2
HU13-T5	Diseño del prototipado de la interfaz.	Kevin Castillo Pérez	Terminada	4
HU13-T6	Implementación de la funcionalidad en el cliente web.	Kevin Castillo Pérez	Terminada	6
HU14-T1	Implementación de los repositorios para el acceso a datos.	Kevin Castillo Pérez	Terminada	4
HU14-T2	Implementación de la lógica de la funcionalidad.	Kevin Castillo Pérez	Terminada	4
HU14-T3	Creación de los *endpoints*.	Kevin Castillo Pérez	Terminada	2
HU14-T4	Diseño e implementación de los DTO. (*Data Transfer Object*)	Kevin Castillo Pérez	Terminada	2
HU14-T5	Diseño del prototipado de la interfaz.	Kevin Castillo Pérez	Terminada	4

HU14-T6	Implementación de la funcionalidad en el cliente web.	Kevin Castillo Pérez	Terminada	6
HU15-T1	Implementación de los repositorios para el acceso a datos.	Kevin Castillo Pérez	Terminada	4
HU15-T2	Implementación de la lógica de la funcionalidad.	Kevin Castillo Pérez	Terminada	8
HU15-T3	Creación de los *endpoints*.	Kevin Castillo Pérez	Terminada	2
HU15-T4	Diseño e implementación de los DTO. (*Data Transfer Object*)	Kevin Castillo Pérez	Terminada	2
HU15-T5	Diseño del prototipado de la interfaz.	Kevin Castillo Pérez	Terminada	4
HU15-T6	Implementación de la funcionalidad en el cliente web.	Kevin Castillo Pérez	Terminada	6
HU16-T1	Implementación de los repositorios para el acceso a datos.	Kevin Castillo Pérez	Terminada	4
HU16-T2	Implementación de la lógica de la funcionalidad.	Kevin Castillo Pérez	Terminada	8
HU16-T3	Creación de los *endpoints*.	Kevin Castillo Pérez	Terminada	2
HU16-T4	Diseño e implementación de los DTO. (*Data Transfer Object*)	Kevin Castillo Pérez	Terminada	2
HU16-T5	Diseño del prototipado de la interfaz.	Kevin Castillo Pérez	Terminada	4
HU16-T6	Implementación de la funcionalidad en el cliente web.	Kevin Castillo Pérez	Terminada	6
HU11-T1	Implementación de los repositorios para el acceso a datos.	Kevin Castillo Pérez	Terminada	4
HU11-T2	Implementación de la lógica de la funcionalidad.	Kevin Castillo Pérez	Terminada	4
HU11-T3	Creación de los *endpoints*.	Kevin Castillo Pérez	Terminada	2
HU11-T4	Diseño e implementación de los DTO. (*Data Transfer Object*)	Kevin Castillo Pérez	Terminada	2

HU11-T5	Diseño del prototipado de la interfaz.	Kevin Castillo Pérez	Terminada	4
HU11-T6	Implementación de la funcionalidad en el cliente web.	Kevin Castillo Pérez	Terminada	6

Para la planificación de las tareas se establecieron un total de 3 Sprint, los cuales se diseñaron con el objetivo de permitir un desarrollo completo de la herramienta. En la Tabla 4 se presenta el Sprint Backlog #0 correspondiente, mientras que en el Anexo 2: Sprint Backlogs se encuentran detallados los *Sprints* restantes que se implementaron para abordar y resolver el problema de investigación planteado.

II.3 Arquitectura de la solución

La arquitectura del software es la estructura de los componentes del programa, la manera en que estos interactúan y la estructura de datos que utilizan. El objetivo principal de la arquitectura del software es aportar elementos que ayuden a la toma de decisiones y, al mismo tiempo, proporcionar conceptos y un lenguaje común que permitan la comunicación entre los miembros de un proyecto. Para conseguirlo, la arquitectura del software construye abstracciones, materializándolas en forma de diagramas comentados. (Pressman, 2010)

Para el desarrollo de se ha definido una arquitectura base para orientar el diseño de las capas lógicas a partir de una propuesta de diseño; brindar una estructura física para soportar el código, creando así un esqueleto base; y proponer mecanismos de colaboración entre los componentes integrados en ella.

En este caso se determina el empleo del estilo “llamada y retorno”, lo cual permite obtener una estructura de programa fácil de modificar persiguiendo la escalabilidad del sistema. Según (Pressman, 2010) el empleo de este estilo posibilita además la comunicación, la coordinación y la cooperación entre los componentes y las restricciones que definen como se integran para conformar el sistema, así como los modelos semánticos que facilitan al diseñador el entendimiento de todas las partes del sistema, evitando que las variaciones realizadas a funcionalidades o componentes específicos afecten el funcionamiento general.

II.3.1 Arquitectura por capas

La arquitectura en capas es un modelo de diseño de software que separa las diferentes funcionalidades del sistema en capas o niveles. Cada capa se encarga de un conjunto de tareas específicas y se comunica con las capas adyacentes mediante interfaces. Las capas se organizan de forma horizontal y solo pueden utilizar las funciones de las capas inferiores. (Durán, 2023)

En el diseño de sistemas informáticos, en la actualidad, se emplean con frecuencia las arquitecturas multinivel, donde a cada capa o nivel, se le asigna una responsabilidad específica, dando lugar a que un cambio en una de ellas no influya directamente en la otra.

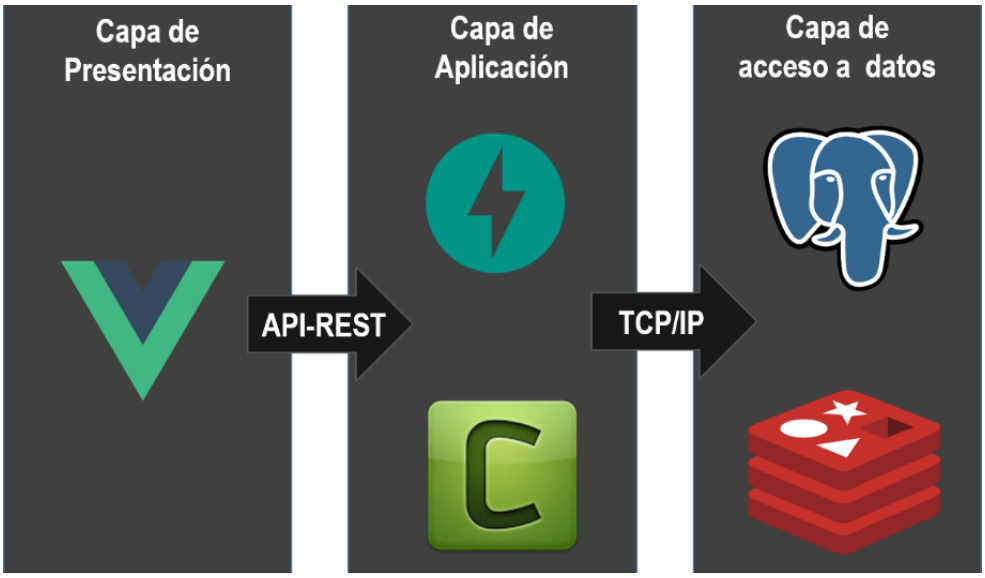


Figura 2: Arquitectura de la solución

Fuente: Elaboración propia

Capa de presentación: es donde reside la interfaz de usuario. Con la cual interactúa, comunica y captura la información del usuario dando un mínimo de proceso. Se encuentra la interfaz de usuario con todas las funcionalidades disponibles según el rol correspondiente. Es donde el *framework* Vue.js se especializa, en el diseño del *frontend.*

Capa de la aplicación: es donde residen los programas que se ejecutan, recibiendo las peticiones del usuario y enviando las respuestas tras el proceso. Esta capa se comunica con la capa de presentación y con la de datos, para solicitar al sistema administrador de base de datos el almacenamiento o recuperación de los datos. FastAPI y Celery Workers son los encargados de ejecutar las tareas que se reciban de la interfaz y devolverlas, al igual que acceso a los bancos de datos de la aplicación. La

conexión ante la capa de presentación es mediante el uso de peticiones HTTP y recursos por REST, proporciona interoperabilidad y escalabilidad al sistema permitiendo así la comunicación bidireccional y la abstracción de la lógica entre capas de forma flexible.

Capa de acceso a datos: es donde residen los datos. Está formada por dos sistemas administradores de bases de datos que realizan todo el almacenamiento, reciben solicitudes de almacenamiento o recuperación de información desde la capa de aplicación. Estos sistemas son PostgreSQL y Redis que se conectarán a través del protocolo TCP/IP a la capa de aplicaciones de forma remota y segura.

De esta forma, se obtiene un sistema escalable, flexible y robusto que permita satisfacer las necesidades del proyecto a lo largo del tiempo, optimizando su desarrollo, pruebas y evolución. El enfoque por capas brindará una base sólida para el diseño de la solución propuesta.

II.3.2 Patrón arquitectónico

Los patrones arquitectónicos ofrecen soluciones a problemas de arquitectura de software en ingeniería de software. Dan una descripción de los elementos y el tipo de relación que tienen junto con un conjunto de restricciones sobre cómo pueden ser usados. Un patrón arquitectónico expresa un esquema de organización estructural esencial para un sistema de software, que consta de subsistemas, sus responsabilidades e interrelaciones. (Pressman, 2010)

La herramienta está planteada con una arquitectura basada en el Modelo Vista Controlador (MVC), esto permite manejar de forma independiente las actividades encargadas de las vistas y las clases controladoras que como su nombre da a relucir están encargadas de controlar el funcionamiento interno del sistema. En el caso específico de Web el modelo-vista-controlador tiene como principal bondad separar los datos de una aplicación, la interfaz de usuario y la lógica de negocios en tres componentes distintos que se relacionarán para tener como resultado la herramienta final. (Buschmann, 1996)

- **Modelo**: El componente maneja lo referente a la persistencia de datos de la herramienta, las clases entidades (paquete “Modelo de datos”), el acceso a los datos (paquete “Acceso a datos”), la seguridad de los datos (paquete “Seguridad”) y los elementos necesarios para manejarlos (paquete “Modelo”).

- **Vista**: El componente representa la interfaz gráfica para la interacción con el usuario. Dentro se ubica la aplicación desarrollada en Vue.js (paquete “Vue App”) que interviene en la visualización del resultado de la comunicación con el Controlador (paquete “Vista”).

- **Controlador**: Componente que contiene las clases que interactúan con la Vista recibiendo las solicitudes de eventos de los usuarios y con el Modelo registrando los cambios realizados por el mismo (paquete “Controlador”).

La arquitectura MVVM (Modelo-Vista-Modelo de Vista) en general y Vue.js en particular se adaptan bien a la creación de aplicaciones web ricas en torno al concepto de "Componentes". Los componentes son construcciones pequeñas, autónomas y, a menudo, reutilizables que reúnen un modelo, una vista y un VM para un solo propósito bien definido. La diferencia entre MVC y MVVM está en la existencia del Modelo de Vista (VM), que es una construcción que proporciona un vínculo/interfaz entre el modelo y la vista.

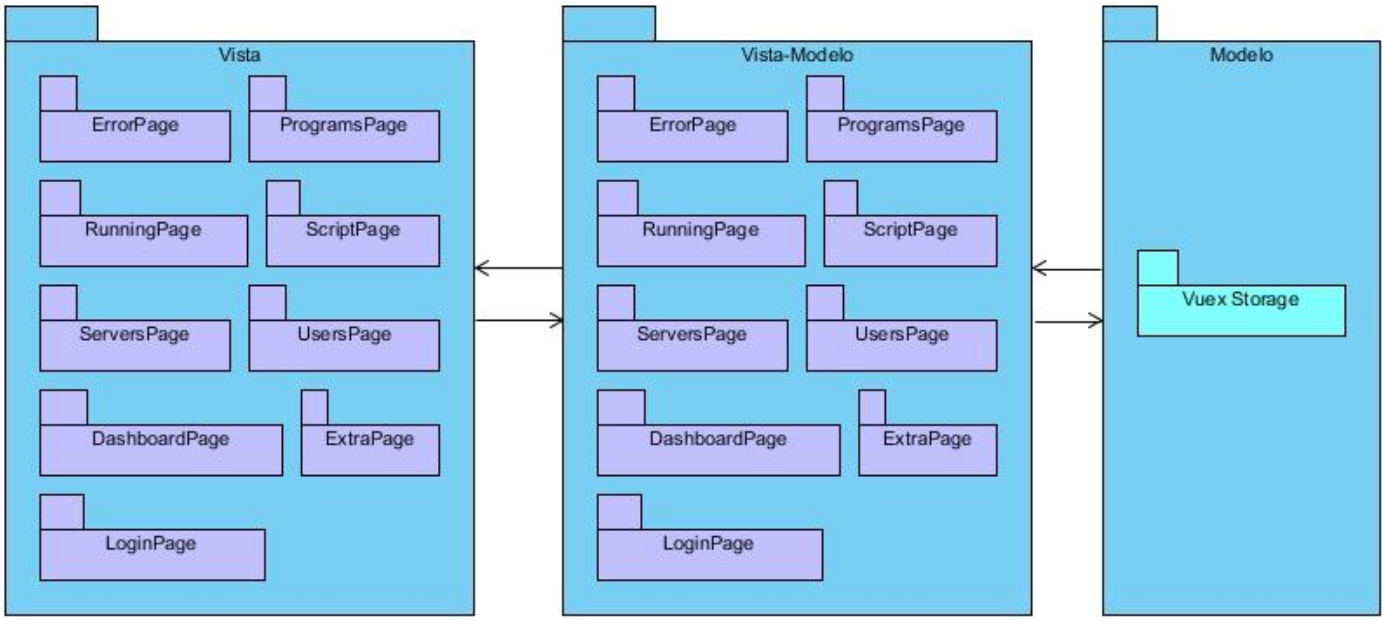


Figura 3: Modelo de la arquitectura de la herramienta (frontend).

Fuente: Elaboración propia.

En la Figura 3 se aprecia la división de la arquitectura MVVM, estos son sus componentes:

- **Modelo**: construcciones simples que contienen datos utilizados por la VM y la aplicación. El modelo no tiene lógica aparte de la validación de datos y no accede a los servicios para recuperar o guardar datos (paquete “Modelo”).

- **Vista**: representa los datos contenidos en el VM. Las vistas están activas y su estructura básica está definida por "plantillas" que son etiquetas de *template* personalizadas, etiquetas DOM (*Document Object Model*) personalizadas o HTML simple (paquete “Vista”).

- **Modelo de vista**: contienen toda la lógica del negocio necesaria para manipular los datos utilizados por la aplicación, también tienen propiedades que están vinculadas a varios elementos DOM en sus plantillas de vista para permitir el enlace de datos, además, los VM tienen métodos (*get*, *set*, *post*, *delete*) que manejan eventos DOM interceptados por directivas incrustadas dentro de la plantilla de Vista (paquete “Vista-Modelo”).

II.3.2 Patrones de diseño

La asignación de responsabilidades es la habilidad más importante en el análisis y diseño orientado por objetos, para ello tiene suma importancia la utilización de los patrones de diseño. En términos generales, un patrón es un conjunto de información que proporciona respuesta a un conjunto de problemas similares, es decir, un patrón es una solución a un problema en un contexto.

Los patrones GRASP son una serie de buenas prácticas enfocadas a la calidad del software, calidad "estructural" por llamarla de alguna manera, ya que no se ha centrado en pruebas sino en la estructura y las responsabilidades de las clases que componen el software que desarrollamos. (Carmona, 2012)

Alta cohesión: Nos dice que la información que almacena una clase debe de ser coherente y debe estar, en la medida de lo posible, relacionada con la clase. El grado de cohesión mide la coherencia de una clase, esto es, lo coherente que es la información que almacena una clase con las responsabilidades y relaciones que ésta tiene con otras clases. La clase de programa es ejemplo del patrón, va a tener comandos que estos van a poder ser asignados a la clase script para ejecutarse las tareas.

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. Se utiliza el patrón controlador mediante la creación de *endpoints* en FastAPI, los cuales actúan como controladores. Estos *endpoints* reciben las solicitudes HTTP y se encargan de coordinar la ejecución mediante la delegación de tareas a los servicios y modelos de datos. Por ejemplo, en el módulo *endpoints/users* se define la clase *UsersAPI* con métodos

como *create_user()*, *update_user()*, *delete_user()*, los cuales reciben las peticiones del *frontend*. Dichos métodos interactúan con el servicio *UserService* para realizar operaciones sobre los modelos de datos, como crear, actualizar o eliminar usuarios de la base de datos.

Creador: El patrón creador nos ayuda a identificar quién debe ser el responsable de la creación o instanciación de nuevos objetos o clases. La creación de instancias es una de las actividades más comunes en un sistema orientado a objetos. En consecuencia, es útil contar con un principio general para la asignación de las responsabilidades de creación. Si se asignan bien el diseño puede soportar un bajo acoplamiento, mayor claridad, encapsulamiento y reutilización. Siguiendo el mismo ejemplo anterior de la clase controladora, el módulo *endpoints/users* posee métodos para poder crear instancias de usuario al sistema.

Experto en información: Experto en información nos dice que la responsabilidad de la creación de un objeto o la implementación de un método, debe recaer sobre la clase que conoce toda la información necesaria para crearlo o ejecutarlo, de este modo obtendremos un diseño con mayor cohesión y cuya información se mantiene encapsulada, es decir, disminuye el acoplamiento. Entre los módulos *endpoints* se encuentra *ServersAPI*, es capaz de interactuar con el modelo *ServerModel* para realizar operaciones CRUD, concentra toda la lógica y conocimiento relacionado con los recursos de servidores (*servers*).

Polimorfismo: Polimorfismo, en programación orientada a objetos, es un concepto muy simple con el que la gente a veces se lía, polimorfismo es permitir que varias clases se comporten de manera distinta dependiendo del tipo que sean. Las clases *AsyncSSHController*, *ParallelSSHController* heredan de la clase abstracta *SSHController* e implementan los métodos abstractos de manera polimórfica

II.3.3 Mapa de navegación

Proporcionan una representación esquemática de la estructura del hipertexto, indicando los principales conceptos incluidos en el espacio de la información y las interrelaciones que existen entre ellos. La organización jerárquica de los vínculos ofrece una buena orientación a los usuarios. (Rovira, 2001)

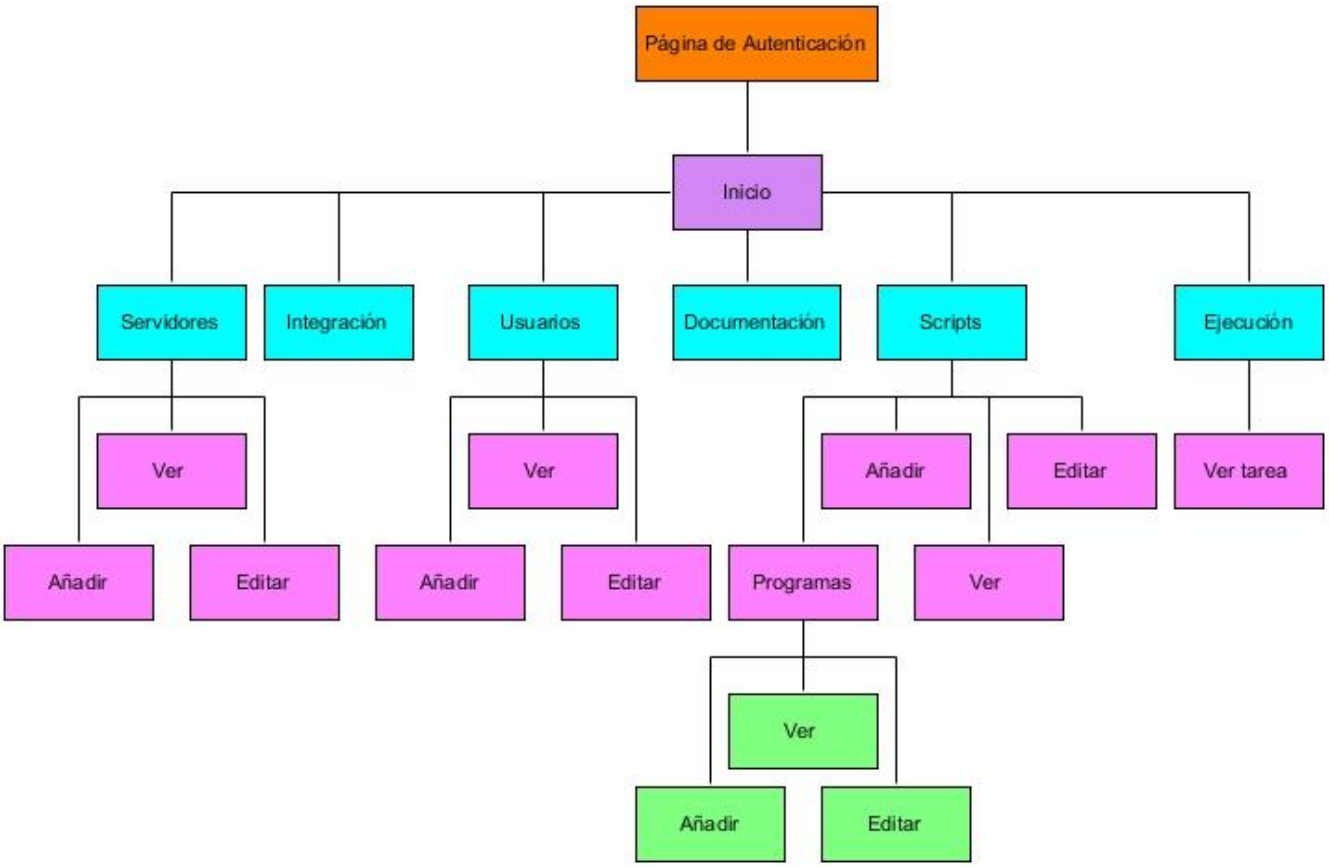


Figura 4: Mapa de Navegación.

Fuente: Elaboración propia.

A continuación, se describe de forma general la arquitectura de navegación propuesta para la interfaz de usuario de la herramienta:

- La página de Inicio presentará un menú lateral desplegable, el cual brindará acceso rápido a las secciones principales de gestión, documentación e integración.
- Dentro de la sección de gestión, las páginas de Servidores y Usuarios mostrarán tablas que listarán sus respectivas entidades, pudiéndose añadir, editar, eliminar o ver detalles de cada registro a través de botones de acción.
- La página de Scripts funcionará de manera análoga, pero a nivel de Programas, enlazando a funcionalidades de gestión sobre estos.

- La ejecución de scripts redireccionará a la página homónima donde se expondrá información relevante.

- La documentación centralizará manuales y especificaciones técnicas. La sección de integración aportará guías para extender la herramienta.

Este diseño minimalista busca optimizar la usabilidad mediante una navegación intuitiva, brindada por el menú lateral siempre disponible en todas las vistas. El objetivo es facilitar la realización de tareas comunes a los usuarios.

II.3.4 Diseño de Interfaz de Usuario

El diseño de interfaz de usuario busca establecer una interacción bidireccional efectiva entre el usuario y el sistema, mediante la aplicación de principios cognitivos y perceptuales, el diseñador conceptualiza la arquitectura de información y delimita los componentes de interacción. Esto resulta en la generación de un modelo de tareas jerarquizado que representa la estructura lógica subyacente en el modelo del dominio. Dicho modelo se concretiza en un prototipo de baja fidelidad que especifica la distribución espacial de los objetos visuales en cada vista, detallando sus estados y transiciones esperadas.

Posteriormente, el diseño de interfaz de usuario se encarga de formalizar las secuencias de interacción mediante la automatización de reglas de negocio y validaciones. Ello asegura que la experiencia de usuario cumpla con estándares cognitivos que optimicen su eficiencia y eficacia en la realización de sus objetivos, haciendo la integración entre ser humano y computador más sencilla e intuitiva. (Pressman, 2010)

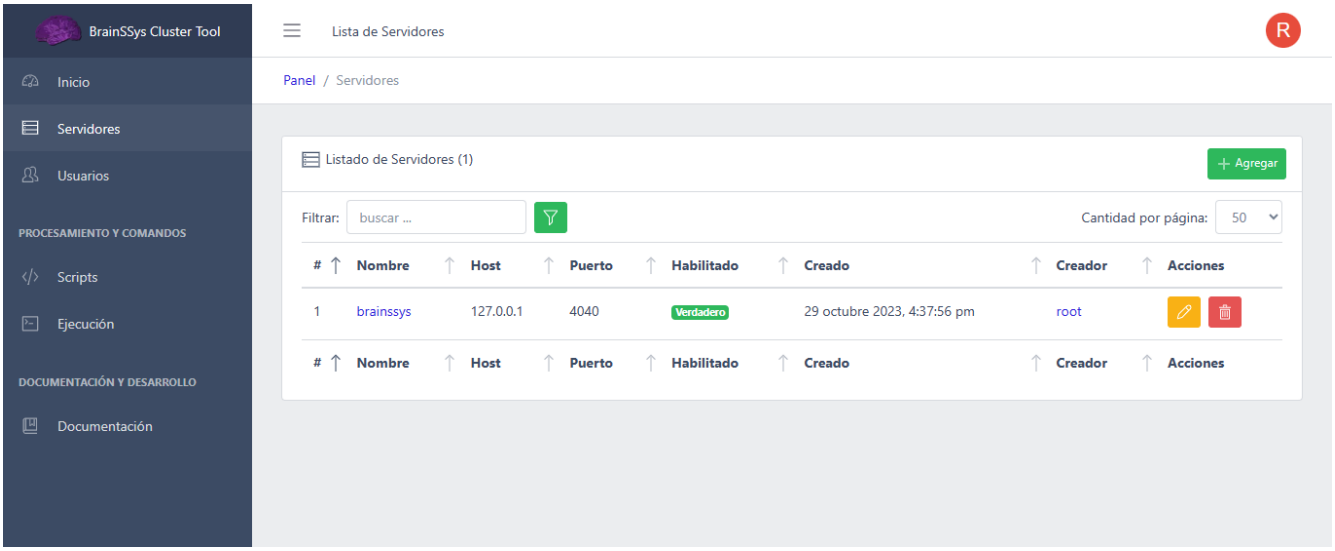


Figura 5: Prototipo de Interfaz de Usuario (RF15. Listar Servidores)

Fuente: Elaboración propia.

Tal y como se detalló en la Figura 4, se generó un mapa de navegación que representa de forma jerarquizada y modular las diversas secciones y subsecciones que conforman el sistema que se desea desarrollar, dicho modelo de tareas permite entender de forma global e intuitiva la arquitectura de información subyacente. De esta forma, el diseño de la interfaz de usuario elaborado se sustenta directamente en el análisis de tareas y la conceptualización de la arquitectura de información realizados con anterioridad. Ello asegura que la experiencia resultante optimice la realización de dichas tareas siguiendo patrones cognitivos, en concordancia con los objetivos planteados.

II.3.5 Modelo de Datos

El modelo de datos determina la estructura lógica de una base de datos y el modo de almacenar, organizar y manipular los datos. Dentro de sus propósitos se encuentra definir los datos persistentes que serán almacenados.

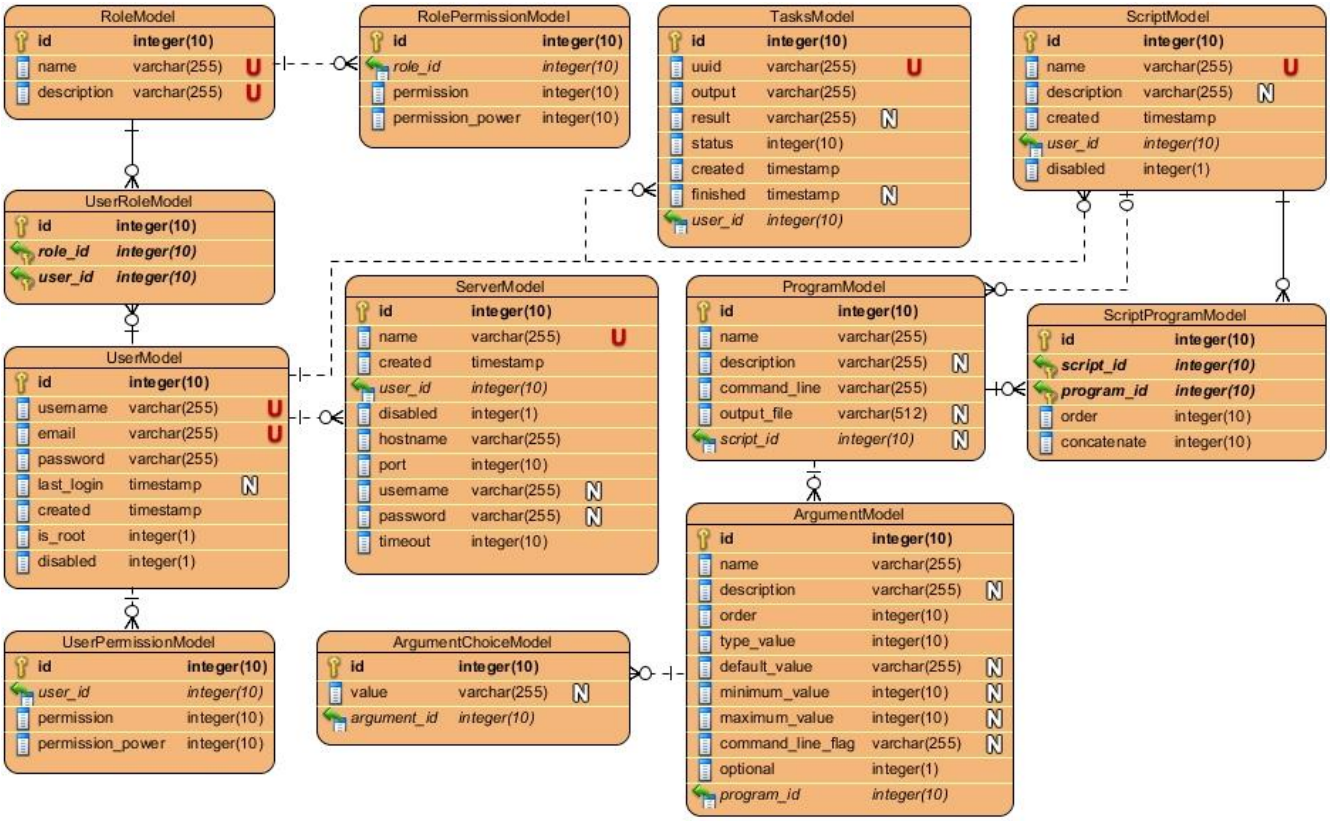


Figura 6: Modelo de datos relacional para la herramienta

Fuente: Elaboración propia.

Con el objetivo de definir las clases persistentes se identifican los conceptos, en el dominio del negocio, que persisten en el tiempo. La persistencia es la capacidad de un objeto de mantener su valor en el espacio y en el tiempo, por lo que no todos los conceptos definidos son transformados en clases de este tipo. Se generó el diagrama entidad-relación representado en la Figura 6, el cual posee un conjunto de tablas correspondientes a cada componente. Se presenta la información que va a persistir en un total de 12 tablas (*RoleModel*, *RolePermissionModel*, *UserModel*, *UserPermissionModel*, *UserRoleModel*, *TasksModel*, *ServerModel*, *ProgramModel*, *ArgumentModel*, *ArgumentChoiceModel*, *ScriptProgramModel*, *ScriptModel*) y la relación que existe entre ellas, garantizando una estructura jerárquica; para la gestión de bases de datos.

Conclusiones del capítulo

Luego de analizar el alcance y requerimientos del proyecto, es posible inferir las siguientes conclusiones:

El modelo de relaciones conceptuales y la descomposición funcional lograda aportan una comprensión integral del sistema propuesto. Asimismo, la descripción de los 28 requisitos funcionales y 6 no funcionales, proporciona una base sólida para orientar el desarrollo. La definición de 28 historias de usuario permite trasladar de manera clara y precisa las necesidades del usuario final, facilitando priorizar los incrementos de valor.

La arquitectura modular adoptada y los patrones MVC, MVVM y GRASP garantizan componentes independientes, alta cohesión y acoplamiento laxo; factores que mejorarán la calidad, mantenimiento y escalabilidad. Finalmente, el diseño de la interfaz a través de un mapa de navegación optimizará la usabilidad y experiencia del usuario final con la herramienta.

CAPÍTULO III: VALIDACIÓN DE LA SOLUCIÓN PROPUESTA

En este capítulo se detallan las especificaciones técnicas vinculadas a la implementación de la solución planteada. Se exponen las pautas de programación empleadas y la herramienta es sometida a pruebas de software, con el fin de validar que satisface los requerimientos establecidos previamente. Se realiza un proceso de *testing* que permite comprobar el correcto funcionamiento de acuerdo a la especificación funcional. De esta forma se verifica que la propuesta cumple su objetivo de dar solución al problema planteado en la investigación.

III.1 Fase de despliegue

Los diagramas de despliegue permiten modelar la disposición física o topología de un sistema; muestran las relaciones entre sus componentes y las conexiones físicas entre el hardware. Un diagrama de despliegue está compuesto por: nodos, dispositivos y conectores. (Sommerville, 2011) La Figura 7 muestra el diagrama de despliegue para la propuesta de solución.

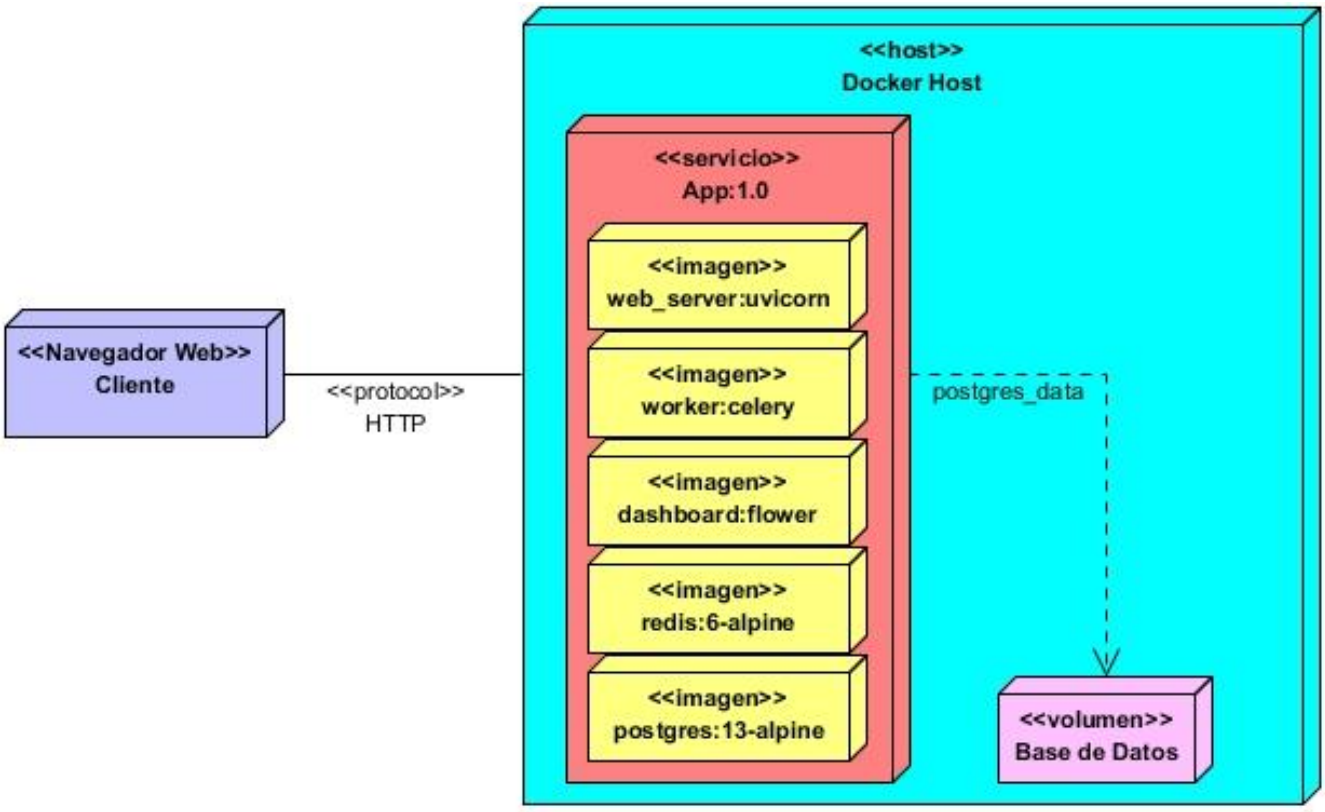


Figura 7: Diagrama de Despliegue

Fuente: Elaboración propia.

El diagrama representa el despliegue de una arquitectura de microservicios con diferentes componentes desplegados a través de Docker. Desde un lado tenemos un cliente que accede mediante una aplicación web implementada con un navegador frontal, este cliente se comunica con un host Docker donde están contenerizados los diferentes microservicios. En primer lugar, tenemos un servidor web WSGI (Interfaz de puerta de enlace de servidor web) implementado con Uvicorn que actúa como *gateway* de entrada para las solicitudes HTTP provenientes del cliente. Este servidor delega tareas asíncronas en un *worker* de tareas en segundo plano manejado por Celery y supervisado a través de un *dashboard* Flower también ejecutado en Docker, el *worker* procesa colas de tareas en Redis. Por otro lado, existe una base de datos PostgreSQL desplegada en Docker que almacena y recupera datos de forma persistente. Todos estos componentes se comunican a través de APIs RESTful (API de transferencia de estado representacional) utilizando HTTP como protocolo de transporte.

III.2 Pruebas de software

Las pruebas de software son esenciales para garantizar la calidad de cualquier producto desarrollado, a través de ellas es posible comprobar que la aplicación funciona correctamente y tal y como se había planificado, cumpliendo con los requisitos establecidos. Realizar pruebas durante todo el ciclo de desarrollo es vital, desde las primeras fases hasta el lanzamiento final. Esto permite detectar errores de manera temprana cuando aún son fáciles de solucionar, de esta forma se evitan defectos en la versión final que podrían comprometer la satisfacción del usuario.

Diferentes tipos de pruebas como las unitarias, de integración o funcionales aseguran que cada elemento funciona de manera aislada y también de forma conjunta. Asimismo, otras pruebas como las de rendimiento, estrés o regresión validan el correcto comportamiento del software ante distintas cargas de trabajo. Gracias a este proceso de *testing* se gana confianza en el producto, pues el usuario sabe que ha sido validado y probado a fondo antes de su lanzamiento. Esto redunda en beneficios económicos para la empresa, dado que se reduce el coste de posibles errores una vez en producción. (UNIR, 2022)

Una estrategia de prueba del software integra los métodos de diseño de caso de pruebas del software en una serie bien planeada de pasos que desembocará en la eficaz construcción del mismo. La estrategia proporciona un mapa que describe los pasos que se darán como parte de la prueba, indica cuándo se planean, cuándo se dan estos pasos, además de cuanto esfuerzo, tiempo y recursos consumirán. Por tanto, cualquier estrategia de prueba debe incorporar la planeación de pruebas, el

diseño de casos de pruebas, la ejecución de pruebas, la recolección y evaluación de los datos resultantes. (Pressman, 2010)

La prueba es aplicada para diferentes tipos de objetivos, en diferentes escenarios o niveles de trabajo. Se distinguen los siguientes Niveles de Pruebas. (Zapata Sánchez, 2013)

- Pruebas unitarias o de componente: se realizan a nivel de código para verificar que cada unidad funcione correctamente de forma individual. Son realizadas por el equipo de desarrollo.

- Pruebas de sistema: tienen el objetivo de verificar que el sistema completo cumple con las especificaciones. Deben ser realizadas por un equipo diferente al de desarrollo, preferiblemente externos. Se llevan a cabo en un entorno similar a producción.

- Pruebas de aceptación: son cruciales para validar que el sistema satisface las necesidades del cliente. Deben contar con la participación de usuarios finales del cliente. Se diferencian las pruebas *alpha*, en entornos del proveedor, de las pruebas beta en entornos del cliente. Estas últimas son obligatorias.

Para realizar las pruebas en la herramienta se desarrollaron las pruebas unitarias, las pruebas de sistema y las pruebas de aceptación.

III.2.1 Pruebas unitarias

Se centra en el esfuerzo de verificación de la unidad más pequeña del diseño del software, el componente o sistema de software. Tomando como guía la descripción del diseño al nivel de componentes, se prueban importantes caminos de control para describir errores dentro de los límites del módulo. El alcance restringido que se ha determinado para las pruebas de unidad limita la relativa complejidad de las pruebas y los errores que estas descubren. Las pruebas se encuentran en la lógica del procedimiento interno y en las estructuras de datos dentro de los límites de un componente. Este tipo de prueba se puede aplicar en paralelo a varios componentes. (Pressman, 2010) En el contexto de la investigación se aplica el método de caja blanca.

Método de caja blanca

La prueba de caja blanca, en ocasiones llamada “prueba de caja de vidrio”, es una filosofía de diseño de casos de prueba que usa la estructura de control descrita como parte del diseño a nivel de

componentes para derivar casos de prueba. (Pressman, 2010) Al usar los métodos de prueba de caja blanca, puede derivar casos de prueba que:

1. Garanticen que todas las rutas independientes dentro de un módulo se revisaron al menos una vez.

2. Revisen todas las decisiones lógicas en sus lados verdadero y falso.

3. Ejecuten todos los bucles en sus fronteras y dentro de sus fronteras operativas.

4. Revisen estructuras de datos internas para garantizar su validez.

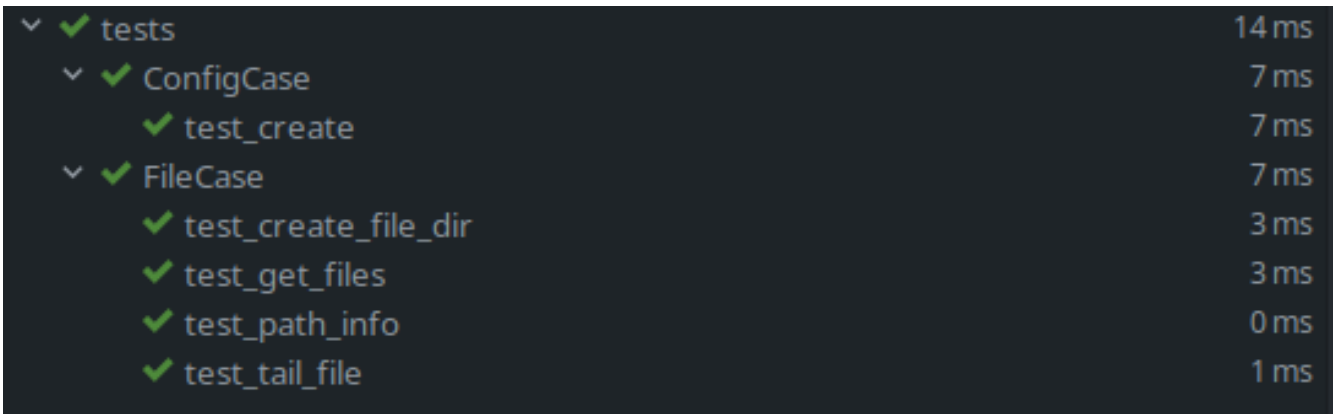


Figura 8: Resultado de las pruebas unitarias

Fuente: Elaboración propia.

En la Figura 8 se muestra el resultado obtenido de las pruebas unitarias en Python, se realizaron un total de cinco pruebas. Se utilizó el *framework* de pruebas *unittest* que ayuda a automatizar el proceso de prueba y permite ejecutar múltiples pruebas en la misma función con diferentes parámetros, verificar las excepciones esperadas y las posibles rutas críticas.

III.2.2 Pruebas del sistema

La prueba de sistema está definida por una serie de ejecuciones cuyo propósito principal es ejercitar por completo el sistema. Aunque cada prueba tenga un propósito diferente, todo el sistema funciona para verificar que los elementos se hayan integrado de manera adecuada y que se realicen las funciones asignadas. (Pressman, 2010)

Entre las técnicas de pruebas que se realizan para evaluar la funcionalidad de la herramienta, se pueden distinguir los siguientes tipos de pruebas:

- Pruebas funcionales
- Pruebas de seguridad
- Pruebas de rendimiento

Pruebas funcionales

Este tipo de prueba se enfoca en validar la correcta implementación de las necesidades del cliente. La funcionalidad puede ser vinculada a los datos de entrada y de salida. Los datos de entrada serán ejecutados y mostrarán un resultado y dicho resultado será comparado con el resultado esperado (comportamiento). (Campos, 2015) Las pruebas de función están enfocadas en los requisitos funcionales y las reglas del negocio. Estas pruebas utilizan el método de Caja Negra.

Método de caja negra

El método de caja negra, también llamadas pruebas de comportamiento, se enfocan en los requerimientos funcionales del software; es decir, las técnicas de prueba de caja negra le permiten derivar conjuntos de condiciones de entrada que revisarán por completo todos los requerimientos funcionales para un programa. (Pressman, 2010)

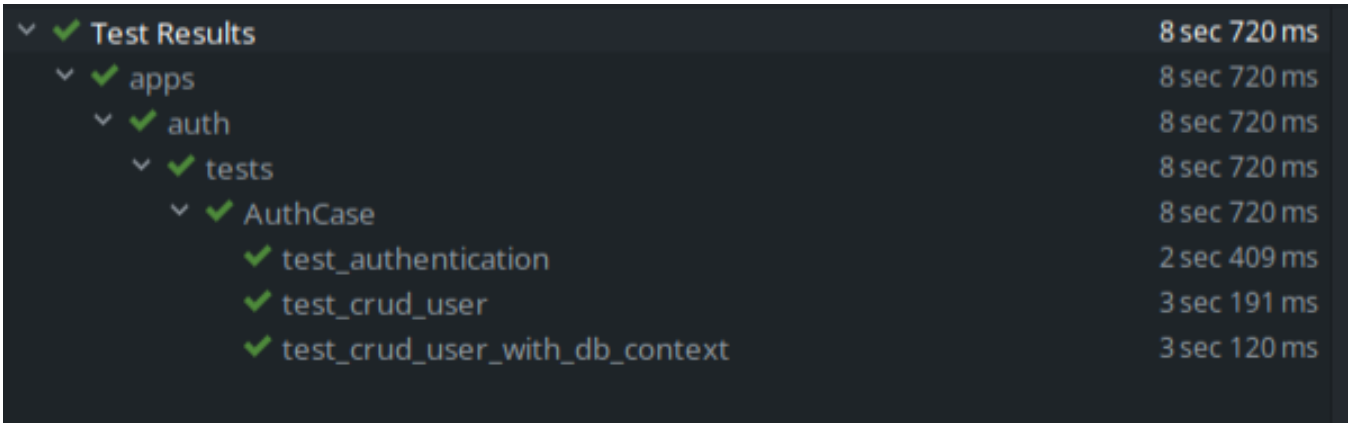


Figura 9: Resultado de las pruebas funcionales

Fuente: Elaboración propia.

En la Figura 9 se muestra el resultado de las pruebas funcionales en Python, con total de 3 pruebas realizadas. Se utilizaron las librerías disponibles en el propio *framework* “FastAPI” que proporciona un cliente API-REST para simular el proceso que realizaría un usuario y permite ejecutar múltiples pruebas en la misma función con diferentes parámetros, verificar las excepciones esperadas y las posibles rutas críticas.

Pruebas de seguridad

La prueba de seguridad intenta verificar que los mecanismos de protección que se construyen en un sistema en realidad lo protegerán de cualquier penetración impropia. Durante la prueba, quien realiza la prueba juega el papel del individuo que sea penetrar al sistema. Quien realice la prueba puede intentar adquirir contraseñas por medios administrativos externos; puede atacar el sistema con software a la medida diseñado para romper cualquier defensa que se haya construido; puede abrumar al sistema, y por tanto negar el servicio a los demás; puede causar a propósito errores del sistema con la esperanza de penetrar durante la recuperación; puede navegar a través de datos inseguros para encontrar la llave de la entrada al sistema. (Pressman, 2010)

Para la realización de las pruebas de seguridad se realizó en conjunto con las pruebas funcionales, se puede observar en la **¡Error! No se encuentra el origen de la referencia.**, en el proceso de autenticación y verificación de los permisos.

Pruebas de rendimiento

Las pruebas de rendimiento se diseñan para poner a prueba el rendimiento del software en tiempo de corrida, dentro del contexto de un sistema integrado. Esta ocurre a lo largo de todos los pasos del proceso de prueba. Incluso en el nivel de unidad, puede accederse al rendimiento de un módulo individual conforme se realizan las pruebas. Sin embargo, no es sino hasta que todos los elementos del sistema están plenamente integrados cuando puede determinarse el verdadero rendimiento de un sistema. (Pressman, 2010)

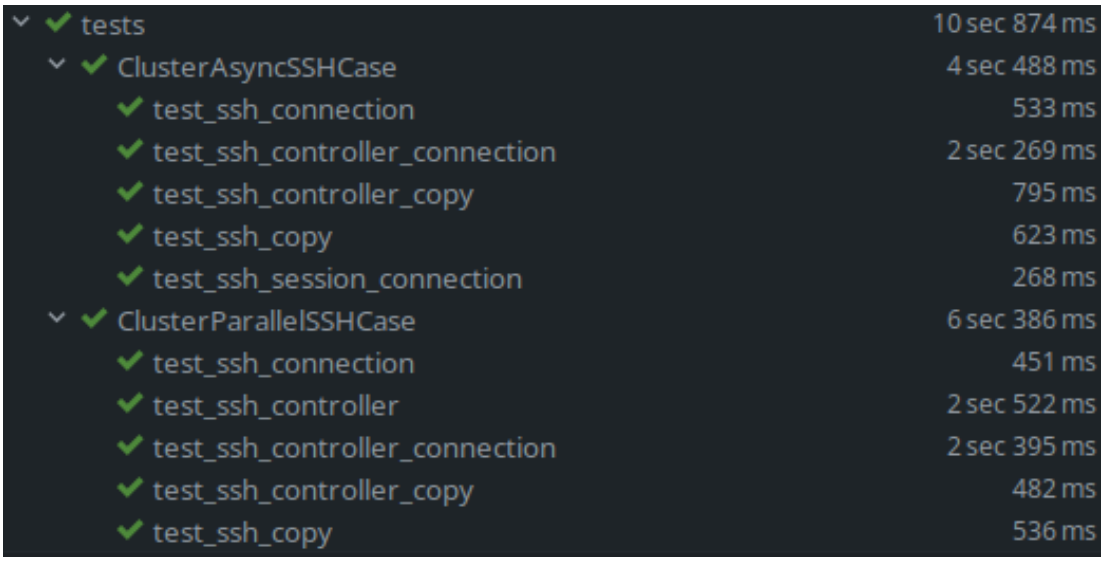


Figura 10: Resultado de las pruebas de rendimiento

Fuente: Elaboración propia.

Las pruebas de rendimiento fueron aplicadas a las librerías "*AsynSSH*" y "*ParallelSSH*" para verificar cuál dispone de un mejor rendimiento. Teniendo como resultado los observados en la Figura 10, se concluyó que la librería "*ParallelSSH*" es más rápida, pero se decidió utilizar "*AsyncSSH*" debido a todas las ventajas que proporciona.

III.2.3 Pruebas de aceptación

Cuando se construye un software para un cliente, se realiza una serie de pruebas de aceptación a fin de permitir al cliente validar todos los requerimientos. Una prueba de aceptación puede variar desde una "prueba de conducción" informal hasta una serie de pruebas planificadas y ejecutadas sistemáticamente. La prueba de aceptación puede realizarse durante un período de semanas o meses, y mediante ella descubrir errores acumulados que con el tiempo puedan degradar el sistema. La mayoría de los constructores de productos de software usan un proceso llamado prueba alfa y prueba beta para descubrir errores que al parecer solo el usuario final es capaz de encontrar. (Pressman, 2010)

Tabla 6: Caso de prueba de aceptación: Autenticar usuario.

Fuente: Los Autores.

CASO DE PRUEBA DE ACEPTACIÓN	
Historia de Usuario:	HU-01: Autenticar usuario
Descripción:	El software debe permitir iniciar sesión a los usuarios

Condiciones de ejecución:		Al entrar al sistema se mostrará una pantalla de autenticación, donde se visualiza un formulario con los campos usuario y contraseña. El usuario accede a la herramienta con sus credenciales.	
Escenarios de prueba:		**Flujo del escenario:**	**Resultados esperados:**
EP1	Insertar credenciales válidas	Se introducen las credenciales correctas y el usuario presiona el botón de iniciar sesión en el software	Se muestra el *dashboard* del software
EP2	Insertar credenciales incorrectas	El usuario introduce las credenciales incorrectas y presiona el botón de iniciar sesión	El software notifica al usuario que sus credenciales no son correctas y se mantiene en la vista de autenticación

Tabla 7: Escenarios de prueba de aceptación: Autenticar usuario.

Fuente: Los Autores

Escenarios de prueba	**Usuario**	**Contraseña**	**Administrador**	**Activo**
EP1	root	1234	verdadero	verdadero
	(V)	(V)	(V)	(V)
EP2	Dionisio	Ass465yt		
	(I)	(V)	(N/A)	(N/A)
	user	Man876		
	(V)	(I)	(N/A)	(N/A)
	victorrm	anose*		
	(I)	(I)	(N/A)	(N/A)
	norman	test		
	(I)	(I)	(N/A)	(N/A)

Tabla 8: Caso de prueba de aceptación: Registrar servidor.

Fuente: Los Autores.

CASO DE PRUEBA DE ACEPTACIÓN			
Historia de Usuario:		HU-07: Registrar servidor	
Descripción:		El sistema debe permitir a los desarrolladores el registro de nuevos servidores	
Condiciones de ejecución:		Una vez presionado el botón “Servidores” del *drawer* se mostrará una vista para la gestión de servidores. Donde aparecerá un botón de añadir servidores, al presionarlo aparece una vista con los campos de entrada para la creación de un servidor. Una vez presionado el botón de aceptar se ejecuta el proceso de creación del servidor	
Escenarios de prueba:		**Flujo del escenario:**	**Resultados esperados:**
EP1	Insertar valores válidos	Se introducen los valores que cumplen con todas las validaciones definidas para cada campo	Se añade el servidor a la lista de servidores y se muestra una notificación de éxito de la operación
EP2	Insertar valores inválidos	Hay valores en los campos que no cumplen con las validaciones especificadas en la historia de usuario	Se muestra un mensaje de error en el campo que no cumple con su validación.

Tabla 9: Escenarios de prueba de aceptación: Registrar servidor.

Fuente: Los Autores

Escenarios de prueba	**Nombre**	**Host**	**Puerto**	**Usuario**	**Contraseña**	**Time out**
EP1	brainssys	10.0.0.1	4040	root	1234	60
	(V)	(V)	(V)	(V)	(V)	(V)
EP2		10.0.0.1	22	Testing	Testing	60
	(N/A)	(V)	(V)	(V)	(V)	(V)
	Principal	10.0.0.1	100000000	Testing	3426vh	60

	(V)	(V)	(I)	(V)	(V)	(V)
	Principal	10.0.0.1	6060	Usuario-inco-rrecto	Testing	60
	(V)	(V)	(V)	(I)	(V)	(V)

Cada caso asociado a las historias de usuario "Autenticación de usuario" y "Registro de servidores", especifica los flujos normales y alternativos, así como los resultados esperados. Adicionalmente, se definen tablas con los datos de prueba categorizados por valor válido e inválido. La formalización de casos y escenarios proporciona una base para la ejecución efectiva del plan de *testing*. Esta valida que la aplicación cumple funcionalmente con las necesidades del cliente, antes de la entrega final.

III.3 Resultado de las pruebas aplicadas

La Figura 11 muestra los resultados de aplicar el método de caja negra y caja blanca, donde se ejecutaron un total de cuatro iteraciones. Además, representa el total de no conformidades identificadas por cada iteración.

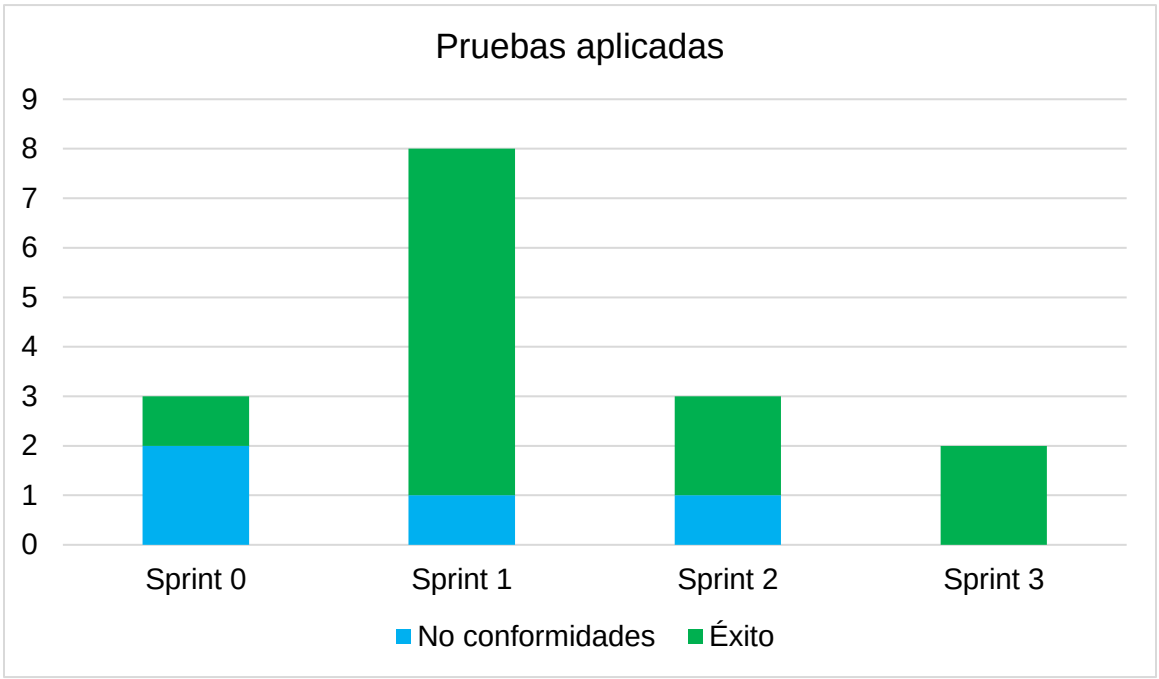


Figura 11: Resultado de las pruebas aplicadas

Fuente: Elaboración propia.

En el Sprint 0 se detectaron un total de dos no conformidades tras ejecutar una prueba de aceptación. En el Sprint 1 se implementaron las funcionalidades descritas en nueve historias de usuario, se realizaron correcciones a las dos que no superaron la primera iteración de pruebas y se ejecutó una prueba de aceptación y seis pruebas unitarias, solo una resultó no satisfactoria. Todas las no conformidades de la primera iteración fueron subsanadas.

En el Sprint 2, una vez saldados los problemas detectados en la iteración anterior, se terminó de implementar siete historias de usuario. Se repitieron las pruebas anteriores y se aplicaron otras tres con el objetivo de validar las nuevas funcionalidades encontrando una no conformidad. En el Sprint 3 no se encontraron no conformidades, validándose de esta manera el correcto funcionamiento de la herramienta propuesta. De un total de 11 pruebas realizadas durante todo el proceso de desarrollo, solo tres resultaron no satisfactorias. Finalmente, se repitieron todas las pruebas resultando satisfactorias.

Conclusiones del capítulo

El presente capítulo abordó una serie de aspectos clave para asegurar la calidad del software desarrollado.

La elaboración del diagrama de despliegue resultó fundamental para comprender a profundidad cómo deberá distribuirse el sistema en entornos reales. Esto permitió replicar dicha configuración durante las pruebas, validando el correcto funcionamiento en un ambiente lo más cercano posible al de producción. las pruebas unitarias comprobaron el correcto funcionamiento a nivel molecular, antes de integrar componentes. Por otro lado, las pruebas de sistema validaron el flujo *end-to-end* con datos reales, emulando escenarios complejos de uso.

Esto permitió identificar y enmendar fallos de manera iterativa, cumpliendo el objetivo principal de esta etapa: entregar software libre de defectos. Puede afirmarse que sin el riguroso proceso de *testing* detallado no se hubiera logrado el nivel de calidad requerido para la puesta en producción.

CONCLUSIONES FINALES

A lo largo de la presente investigación se abordó el desarrollo de una solución integral para el procesamiento de datos neurocientíficos en el Centro de Neurociencias de Cuba, siguiendo un riguroso enfoque metodológico. Entre las principales conclusiones a las que se puede arribar con la investigación desarrollada se encuentran:

- El análisis del estado del arte permitió comprender a profundidad el contexto y necesidades del CNEURO, identificando que actualmente no existe una solución integral.
- El modelado conceptual y funcional logrado, junto con la descripción rigurosa de requerimientos y casos de uso, proporcionan una base sólida para orientar el desarrollo.
- La arquitectura modular adoptada optimizará la calidad, escalabilidad y mantenimiento del sistema.
- El diagrama de despliegue y modelo de implementación aseguran la replicación de entornos para validaciones.
- Las pruebas unitarias y de integración permitieron identificar fallos de forma temprana. La herramienta desarrollada cumple los requerimientos funcionales y no funcionales definidos y la validación concluyó en resultados satisfactorios respecto a los atributos de usabilidad medidos.

RECOMENDACIONES

Para dar continuidad a la presente investigación se recomienda:

- Se recomienda evaluar la viabilidad técnica de migrar la plataforma a un entorno de nube, esto permitiría escalar recursos de manera elástica según la demanda, garantizando alta disponibilidad y redundancia. Además, facilitaría el despliegue y actualización de nuevas funcionalidades.

- Se recomienda desarrollar un algoritmo de análisis semántico capaz de identificar de forma automática las principales operaciones contenidas en los scripts de procesamiento de datos neurocientíficos. Esto permitiría a los investigadores conocer de forma intuitiva el tipo de análisis que se ejecutará al cargar determinado script, facilitando la planificación computacional. Adicionalmente, brindaría información valiosa sobre tendencias y requerimientos comunes, almacenándola en una base de conocimiento que optimice el diseño de futuros componentes.

REFERENCIAS BIBLIOGRÁFICAS

BIL. (2023). Obtenido de Brain Image Library: https://www.brainimagelibrary.org/about.html

Buschmann, F. (1996). *Pattern-Oriented Software Architecture Volume 1: A System of Patterns.*

Campos, C. (2015). *Las pruebas en el desarrollo de software.* Obtenido de http://www.ptolomeo.unam.mx:8080/xmlui/handle/132.248.52.100/7627

Carmona, J. G. (2012). SOLID Y GRASP. Buenas prácticas hacia el éxito en el desarrollo de software. Obtenido de https://juan-garcia-carmona.blogspot.com/2012/11/solid-y-grasp-buenas-practicas-hacia-el.html

CNEURO. (2023). Obtenido de Empresa Centro de Neurociencias de Cuba: https://www.cneuro.cu/index.php/quienes-somos/

Das, S., Zijdenbos, A. P., Harlap, J., Vins, D., & Evans, A. C. (2011). LORIS: A web-based data management system for multi-center studies. *Frontiers in Neuroinformatic*. Obtenido de https://doi.org/10.3389/fninf.2011.00037

Diccionario Tecnológico. (2023). Obtenido de Muy Tecnologicos: https://muytecnologicos.com/diccionario-tecnologico/cluster-de-computadoras

Digital Guide IONOS. (2023). Obtenido de https://www.ionos.es/digitalguide/servidores/know-how/hpc-high-performance-computing/

Docker. (2023). Obtenido de Docker: https://www.docker.com/

Dokmanic, I., Parhizkar, M., Raninen, A., Vetterli, M., & Schlögel, A. (2019). En *Euclidean distance matrices: An algorithmic perspective* (págs. 85-99). IEEE Signal Processing Magazine. Obtenido de https://ieeexplore.ieee.org/document/7493835

Durán, M. (2023). Qué es la arquitectura en capas, ventajas y ejemplos. *Hubspot*. Obtenido de https://blog.hubspot.es/website/que-es-arquitectura-en-capas

Economipedia. (2023). Obtenido de Economipedia: https://economipedia.com/definiciones-diccionario

Fernández, Y. (2019). API: Qué es y para qué sirve. *Xataka*. Obtenido de https://www.xataka.com/basics/api-que-sirve

GBC. (2023). Obtenido de Global Brain Consortium.: https://globalbrainconsortium.org/index.html

Google. (2023). *Google For Developers*. Obtenido de https://developers.google.com/speed/docs/insights/about?hl=es

Group PostgreSQL Global Development. (2022). PostgreSQL. *PostgreSQL*. Obtenido de https://www.postgresql.org/

HFSP. (2023). Obtenido de Human Frontier Science Program. : https://www.hfsp.org/

Iberdrola. (2019). Neurotecnología, ¿cómo revelar los secretos del cerebro humano? *Iberdrola*. Obtenido de https://www.iberdrola.com/innovacion/neurotecnologia

IBI. (2023). Obtenido de International Brain Initiative.: https://www.internationalbraininitiative.org/

IBM. (2023). ¿Qué es la computación de alto rendimiento (HPC)? *IBM*. Obtenido de https://www.ibm.com/es-es/topics/hpc#:~:text=HPC%20es%20una%20tecnolog%C3%ADa%20que%20utiliza%20cl%C3%BAsteres%20de,y%20resolver%20problemas%20complejos%20a%20velocidades%20extremadamente%20altas.

IBRO. (2023). Obtenido de International Brain Research Organization: https://ibro.org/

Maastricht University. (2023). Obtenido de Maastricht University: https://www.maastrichtuniversity.nl/

Mancuzo, G. (2021). Historias de usuario de Scrum: Plantilla y Ejemplos. *Compara Software Blog*. Obtenido de https://blog.comparasoftware.com/historias-de-usuario-de-scrum-plantilla-y-ejemplos/

McGill University. (2023). Obtenido de McGill University: https://www.mcgill.ca/

Nextcloud. (2023). Obtenido de Nextcloud: https://nextcloud.com/

Poldrack, R. A., Barch, D. M., Mitchell, J., Wager, T., Wagner, A. D., Devlin, J. T., & O'Dubhghaill, C. (2020). Toward Open Sharing of Task-Based fMRI Data: The OpenfMRI Project. *Frontiers*. Obtenido de https://doi.org/10.3389/fninf.2013.00012.

Pressman, R. S. (2010). *Ingeniería del software: un enfoque práctico.*

Python Docs. (2023). Obtenido de Python: https://docs.python.org/3/

Ramírez, S. (2023). *FastAPI*. Obtenido de FastAPI: https://fastapi.tiangolo.com/es/

Redolar-Ripoll, D. (2002). Neurociencia: la génesis de un conceptodesde un punto de vista multidisciplinar. *ResearchGate*. Obtenido de https://www.researchgate.net/publication/239929071_Neurociencia_la_genesis_de_un_concepto_desde_un_punto_de_vista_multidisciplinar

Rojas Porras, R. C. (2022). *SCRIBD.* Obtenido de SCRIBD: https://es.scribd.com/document/459210378/La-neurociencia-o-neurobiologia-es-un-campo-de-la-ciencia-que-estudia-el-sistema-nervioso-y-todos-sus-aspectos

Rovira, C. (2001). *Herramientas de ayuda a la navegación.*

RWTH-Aachen. (2023). Obtenido de RWTH International Academy: https://www.academy.rwth-aachen.de

Sherif, T., Rioux, P., Rousseau, M.-E., Kassis, N., Beck, N., Adalat, R., . . . Evans, A. C. (2014). CBRAIN: A web-based, distributed computing platform for collaborative neuroimaging research. *Frontiers in Neuroinformatics*. Obtenido de https://doi.org/10.3389/fninf.2014.00054

Sommerville, I. (2011). *Software Engineering.* Pearson.

Soriano, J. (2023). Neurotecnología: qué es, para qué sirve, y qué técnicas usa. *Psicología y Mente*. Obtenido de https://psicologiaymente.com/neurociencias/neurotecnologia

TechEdu: Definiciones de términos técnicos. (2023). Obtenido de https://hmong.es/wiki/Neuroinformatics

Trigas Gallego, M. (2012). *Metodología Scrum.* . Obtenido de https://docplayer.es/917979-Tfc-metodologia-scrum-gestion-de-proyectos-informaticos-autor-manuel-trigas-gallego-consultora-ana-cristina-domingo-troncho.html

UESTC. (2023). Obtenido de University of Electronic Science and Technology of China. : https://en.uestc.edu.cn/

UNAM, R. D. (2022). ¿Qué es un cluster? *Revista Digital Universitaria - UNAM*. Obtenido de http://www.revista.unam.mx/vol.4/num2/art3/cluster.htm

UNIR. (2022). Pruebas de software: Tipos e importancia. Obtenido de https://www.unir.net/ingenieria/revista/pruebas-software/

Visual Paradigm. (2022). Ideal Modeling & Diagramming Tool for Agile Team Collaboration. *Visual Paradigm*. Obtenido de https://www.visual-paradigm.com/

Vue.js. (2023). Obtenido de Vue.js: https://vuejs.org/

Zapata Sánchez, J. (2013). *Niveles de Prueba del Software*. Obtenido de PRUEBAS DE SOFTWARE: https://pruebasdelsoftware.wordpress.com/2013/01/21/niveles-de-prueba-del-software/

ANEXOS

ANEXO 1: HISTORIAS DE USUARIO

Tabla 10: Historia de Usuario: Autenticar usuario

Fuente: Los Autores.

HU01 – Autenticar usuario	
Yo como:	Usuario de la herramienta
Quiero:	Autenticarme en la herramienta
Para:	Tener acceso a la herramienta
Criterios de aceptación • Mostrar mensaje de usuario o contraseña incorrectos al introducir credenciales incorrectas • Devolver la petición con los datos necesarios del usuario	

Tabla 11: Historia de Usuario: Registrar usuario

Fuente: Los Autores.

HU02 – Registrar usuario	
Yo como:	Desarrollador
Quiero:	Registrar un usuario en la herramienta
Para:	Insertar un nuevo usuario en la herramienta
Criterios de aceptación • El campo usuario es requerido, de no introducirlo mostrar el mensaje: “El campo es requerido”. • En caso de que el proceso falle notificarle al usuario. • El campo email es requerido, de no introducirlo mostrar el mensaje: “El campo es requerido”. • El campo habilitado es requerido, de no seleccionar mostrar el mensaje: “El campo es requerido”. • Se debe de seleccionar al menos un permiso para los usuarios, de no hacerlo notificarle al usuario un mensaje de error. • El usuario debe de tener un rol, de no seleccionarlo notificarle al usuario que debe de seleccionar un rol.	

Tabla 12: Historia de Usuario: Modificar usuario

Fuente: Los Autores.

HU03 – Modificar usuario	
Yo como:	Desarrollador
Quiero:	Modificar usuarios
Para:	Actualizar la información de los usuarios
Criterios de aceptación • Cargar la información previamente introducida del usuario. • El campo usuario es requerido, de no introducirlo notificar al usuario mediante un mensaje. • El campo email es requerido, de no introducirlo notificar al usuario mediante un mensaje. • El campo contraseña es requerido, de no introducirlo notificar al usuario mediante un mensaje. • El campo ultimo acceso es requerido, de no introducirlo notificar al usuario mediante un mensaje.	

Tabla 13: Historia de Usuario: Visualizar detalles del usuario

Fuente: Los Autores.

HU04 – Visualizar detalles del usuario	
Yo como:	Desarrollador
Quiero:	Ver la información de un usuario
Para:	Conocer su información
Criterios de aceptación • Buscar la información mediante el usuario. • Notificar al usuario en caso de que la operación falle.	

Tabla 14: Historia de Usuario: Eliminar usuario

Fuente: Los Autores.

HU05 – Eliminar usuario	
Yo como:	Desarrollador
Quiero:	Remover un usuario de la herramienta
Para:	No tener información innecesaria
Criterios de aceptación	

• Se debe de eliminar mediante el nombre de usuario. • Notificar en caso de que la operación falle.

Tabla 15: Historia de Usuario: Listar usuario

Fuente: Los Autores.

HU06 – Listar usuarios	
Yo como:	Desarrollador
Quiero:	Ver todos los usuarios registrados
Para:	Conocer quienes se encuentran registrados en la herramienta
Criterios de aceptación • Mostrar usuario, email, contraseña, habilitado. • Notificar en caso de que la operación falle.	

Tabla 16: Historia de Usuario: Modificar servidor

Fuente: Los Autores.

HU08 – Modificar servidor	
Yo como:	Desarrollador
Quiero:	Modificar la información de un servidor
Para:	Actualizar su información
Criterios de aceptación • El campo nombre es requerido, de no ser insertado notificarlo mediante un mensaje. • El campo host es requerido, de no ser insertado notificarlo mediante un mensaje. • El campo puerto es requerido, de no ser insertado notificarlo mediante un mensaje. • El campo usuario es requerido, de no ser insertado notificarlo mediante un mensaje. • El campo contraseña es requerido, de no ser insertado notificarlo mediante un mensaje. • El campo *timeout* es requerido, de no ser insertado notificarlo mediante un mensaje. • Notificar al usuario en caso de que el proceso falle.	

Tabla 17: Historia de Usuario: Visualizar detalles del servidor

Fuente: Los Autores.

HU09 –Visualizar detalles del servidor	
Yo como:	Desarrollador

Quiero:	Ver la información de un servidor
Para:	Conocer sobre sus detalles
Criterios de aceptación • Buscar el servidor a visualizar mediante el nombre. • Mostrar nombre, host, puerto, *timeout*, habilitado. • Notificar en caso de que la operación falle.	

Tabla 18: Historia de Usuario: Eliminar servidor

Fuente: Los Autores.

HU10 – Eliminar servidor	
Yo como:	Desarrollador
Quiero:	Remover un servidor previamente insertado
Para:	No tener información innecesaria en la herramienta
Criterios de aceptación • Eliminar el servidor mediante su nombre. • Notificar en caso de que la operación falle.	

Tabla 19: Historia de Usuario: Listar servidores

Fuente: Los Autores.

HU11 – Listar servidores	
Yo como:	Desarrollador
Quiero:	Visualizar los servidores registrados
Para:	Conocer los servidores registrados
Criterios de aceptación • Mostrar por cada servidor el nombre, host, puerto, habilitado. • Notificar en caso de que la operación falle.	

Tabla 20: Historia de Usuario: Registrar script

Fuente: Los Autores.

HU12 – Registrar script	
Yo como:	Desarrollador

Quiero:	Crear un script
Para:	Que sea ejecutado cuando se le indique
Criterios de aceptación • El campo nombre es requerido, de no ser insertado notificarle al usuario mediante un mensaje. • El campo descripción es requerido. • Notificar en caso de que la operación falle.	

Tabla 21: Historia de Usuario: Modificar script

Fuente: Los Autores.

HU13 – Modificar script	
Yo como:	Desarrollador
Quiero:	Modificar un script
Para:	Actualizar la información
Criterios de aceptación • El campo nombre es requerido, de no ser insertado notificarle al usuario mediante un mensaje • El campo descripción es requerido. • Notificar en caso de que la operación falle.	

Tabla 22: Historia de Usuario: Eliminar script

Fuente: Los Autores.

HU14 – Eliminar script	
Yo como:	Desarrollador
Quiero:	Remover un script
Para:	No tener información innecesaria en la herramienta
Criterios de aceptación • Se elimina el script mediante su nombre. • Notificar en caso de que la operación falle.	

Tabla 23: Historia de Usuario: Listar scripts

Fuente: Los Autores.

HU15 – Listar scripts	
Yo como:	Desarrollador y especialista
Quiero:	Visualizar todos los scripts
Para:	Conocer todos los scripts insertados
Criterios de aceptación • Mostrar por cada script nombre, fecha de creación, creador, habilitado. • Notificar en caso de que la operación falle.	

Tabla 24: Historia de Usuario: Visualizar detalles de un script

Fuente: Los Autores.

HU16 – Visualizar detalles de un script	
Yo como:	Desarrollador y especialista
Quiero:	Ver detalles de un script
Para:	Conocer a profundidad un script
Criterios de aceptación • Buscar el script a visualizar por su nombre • Mostrar del script nombre, fecha de creación, creador, habilitado • Notificar en caso de que la operación falle	

Tabla 25: Historia de Usuario: Concatenar programas del script

Fuente: Los Autores.

HU17 – Concatenar programas del script	
Yo como:	Desarrollador
Quiero:	Añadir programas al script
Para:	Crear varios programas para un script
Criterios de aceptación • Para crear un programa al script se debe de poner un operador. • Notificar en caso de que la operación falle.	

Tabla 26: Historia de Usuario: Registrar programa

Fuente: Los Autores.

HU18 – Registrar programa	
Yo como:	Desarrollador
Quiero:	Insertar un programa
Para:	Que el investigador lo ejecute

Criterios de aceptación

- El campo nombre es requerido, de no introducirlo notificar al usuario mediante un mensaje.
- El campo descripción es requerido, de no introducirlo notificar al usuario mediante un mensaje.
- El campo comando es requerido, de no introducirlo notificar al usuario mediante un mensaje.
- El campo descripción es requerido, de no introducirlo notificar al usuario mediante un mensaje.
- El campo salida es requerido, de no introducirlo notificar al usuario mediante un mensaje.

Tabla 27: Historia de Usuario: Modificar programa

Fuente: Los Autores.

HU19 – Modificar programa	
Yo como:	Desarrollador
Quiero:	Modificar programas
Para:	Actualizar la información

Criterios de aceptación

- El campo nombre es requerido, de no introducirlo notificar al usuario mediante un mensaje.
- El campo descripción es requerido, de no introducirlo notificar al usuario mediante un mensaje.
- El campo comando es requerido, de no introducirlo notificar al usuario mediante un mensaje.
- El campo descripción es requerido, de no introducirlo notificar al usuario mediante un mensaje.
- El campo salida es requerido, de no introducirlo notificar al usuario mediante un mensaje.
- Notificar en caso de que la operación falle.

Tabla 28: Historia de Usuario: Visualizar detalles del programa

Fuente: Los Autores.

HU20 – Visualizar detalles del programa	
Yo como:	Desarrollador
Quiero:	Ver los detalles de un programa especifico
Para:	Ver sus características
Criterios de aceptación • Se mostrarán del programa el nombre, descripción, comando, salida y argumentos. • Notificar en caso de que la operación falle.	

Tabla 29: Historia de Usuario: Eliminar programa

Fuente: Los Autores.

HU21 – Eliminar programa	
Yo como:	Desarrollador
Quiero:	Remover un programa
Para:	No tener información innecesaria en la herramienta
Criterios de aceptación • El programa se especificará mediante su nombre. • Notificar en caso de que la operación falle.	

Tabla 30: Historia de Usuario: Listar programas

Fuente: Los Autores.

HU22 – Listar programas	
Yo como:	Desarrollador
Quiero:	Visualizar todos los programas existentes
Para:	Conocer los programas existentes
Criterios de aceptación • Se mostrará por cada programa su nombre, comando y argumentos. • Notificar en caso de que la operación falle.	

Tabla 31: Historia de usuario: Ejecutar tarea.

Fuente: Los Autores.

HU23 – Ejecutar tarea	
Yo como:	Especialista o desarrollador
Quiero:	Ejecutar un script
Para:	Analizar datos de neurociencia
Criterios de aceptación • El script se especificará mediante su nombre • Notificar en caso de que la operación falle	

Tabla 32: Historia de Usuario: Obtener tarea

Fuente: Los Autores.

HU24 – Obtener tarea	
Yo como:	Especialista o desarrollador
Quiero:	Ver los detalles de una tarea en ejecución
Para:	Tener conocimiento de su estado
Criterios de aceptación • La tarea se especificará mediante su id. • Notificar en caso de que la operación falle.	

Tabla 33: Historia de Usuario: Detener tarea

Fuente: Los Autores.

HU25 – Detener tarea	
Yo como:	Especialista o desarrollador
Quiero:	Detener la ejecución de una tarea
Para:	Detener el análisis actual
Criterios de aceptación • La tarea se especificará mediante su id. • Notificar en caso de que la operación falle.	

Tabla 34: Historia de Usuario: Comprobar tareas

Fuente: Los Autores.

HU26 – Comprobar tareas	
Yo como:	Especialista o desarrollador
Quiero:	Ver el estado de las tareas en ejecución
Para:	Tener conocimiento de su estado
Criterios de aceptación • Se mostrará en una tabla las tareas en ejecución con su estado. • Notificar en caso de que la operación falle.	

Tabla 35: Historia de Usuario: Obtener registro de la tarea

Fuente: Los Autores.

HU27 – Obtener registro de la tarea	
Yo como:	Especialista o desarrollador
Quiero:	Ver el resultado de una tarea
Para:	Analizar los resultados
Criterios de aceptación • Exportar a un fichero la información arrojada de la tarea. • Notificar en caso de que la operación falle.	

Tabla 36: Historia de Usuario: Obtener registro de errores de la tarea

Fuente: Los Autores.

HU28 – Obtener registro de errores de la tarea	
Yo como:	Especialista y desarrollador
Quiero:	Ver el registro de errores de una tarea
Para:	Ver qué falló
Criterios de aceptación • Exportar a un fichero con las excepciones arrojada de la tarea. • Notificar en caso de que la operación falle.	

ANEXO 2: SPRINT BACKLOGS

Tabla 37: Sprint Backlog #1

Fuente: Los Autores

Sprint #1				
ID	**Tarea**	**Asignada a**	**Estado**	**Tiempo**
HU2-T1	Diseño e implementación de clases POCO (*Plain Old CLR Objects*)	Kevin Castillo Pérez	Terminada	2
HU2-T2	Migración a la base de datos	Kevin Castillo Pérez	Terminada	2
HU2-T3	Implementación de los repositorios para el acceso a datos	Kevin Castillo Pérez	Terminada	4
HU2-T4	Implementación de la lógica de la funcionalidad	Kevin Castillo Pérez	Terminada	4
HU2-T5	Creación de los *endpoints*	Kevin Castillo Pérez	Terminada	2
HU2-T6	Diseño e implementación de los DTO (Data Transfer Object)	Kevin Castillo Pérez	Terminada	2
HU2-T7	Diseño del prototipado de la interfaz	Kevin Castillo Pérez	Terminada	4
HU2-T8	Implementación de la funcionalidad en el cliente web	Kevin Castillo Pérez	Terminada	6
HU1-T1	Implementación de los repositorios para el acceso a datos	Kevin Castillo Pérez	Terminada	2
HU1-T2	Implementación de la lógica de la funcionalidad	Kevin Castillo Pérez	Terminada	2
HU1-T3	Creación de los *endpoints*	Kevin Castillo Pérez	Terminada	2
HU1-T4	Diseño e implementación de los DTO (Data Transfer Object)	Kevin Castillo Pérez	Terminada	2

HU1-T5	Diseño del prototipado de la interfaz	Kevin Castillo Pérez	Terminada	4
HU1-T6	Implementación de la funcionalidad en el cliente web	Kevin Castillo Pérez	Terminada	4
HU3-T1	Implementación de los repositorios para el acceso a datos	Kevin Castillo Pérez	Terminada	2
HU3-T2	Implementación de la lógica de la funcionalidad	Kevin Castillo Pérez	Terminada	2
HU3-T3	Creación de los *endpoints*	Kevin Castillo Pérez	Terminada	2
HU3-T4	Diseño e implementación de los DTO (Data Transfer Object)	Kevin Castillo Pérez	Terminada	2
HU3-T5	Diseño del prototipado de la interfaz	Kevin Castillo Pérez	Terminada	4
HU3-T6	Implementación de la funcionalidad en el cliente web	Kevin Castillo Pérez	Terminada	4
HU4-T1	Implementación de los repositorios para el acceso a datos	Kevin Castillo Pérez	Terminada	2
HU4-T2	Implementación de la lógica de la funcionalidad	Kevin Castillo Pérez	Terminada	2
HU4-T3	Creación de los *endpoints*	Kevin Castillo Pérez	Terminada	2
HU4-T4	Diseño e implementación de los DTO (Data Transfer Object)	Kevin Castillo Pérez	Terminada	2
HU4-T5	Diseño del prototipado de la interfaz	Kevin Castillo Pérez	Terminada	4
HU4-T6	Implementación de la funcionalidad en el cliente web	Kevin Castillo Pérez	Terminada	4
HU5-T1	Implementación de los repositorios para el acceso a datos	Kevin Castillo Pérez	Terminada	2

HU5-T2	Implementación de la lógica de la funcionalidad	Kevin Castillo Pérez	Terminada	2
HU5-T3	Creación de los *endpoints*	Kevin Castillo Pérez	Terminada	2
HU5-T4	Diseño e implementación de los DTO (Data Transfer Object)	Kevin Castillo Pérez	Terminada	2
HU5-T5	Diseño del prototipado de la interfaz	Kevin Castillo Pérez	Terminada	4
HU5-T6	Implementación de la funcionalidad en el cliente web	Kevin Castillo Pérez	Terminada	4
HU6-T1	Implementación de los repositorios para el acceso a datos	Kevin Castillo Pérez	Terminada	2
HU6-T2	Implementación de la lógica de la funcionalidad	Kevin Castillo Pérez	Terminada	2
HU6-T3	Creación de los *endpoints*	Kevin Castillo Pérez	Terminada	2
HU6-T4	Diseño e implementación de los DTO (Data Transfer Object)	Kevin Castillo Pérez	Terminada	2
HU6-T5	Diseño del prototipado de la interfaz	Kevin Castillo Pérez	Terminada	4
HU6-T6	Implementación de la funcionalidad en el cliente web	Kevin Castillo Pérez	Terminada	4
HU8-T1	Implementación de los repositorios para el acceso a datos	Kevin Castillo Pérez	Terminada	2
HU8-T2	Implementación de la lógica de la funcionalidad	Kevin Castillo Pérez	Terminada	2
HU8-T3	Creación de los *endpoints*	Kevin Castillo Pérez	Terminada	2
HU8-T4	Diseño e implementación de los DTO (Data Transfer Object)	Kevin Castillo Pérez	Terminada	2

HU8-T5	Diseño del prototipado de la interfaz	Kevin Castillo Pérez	Terminada	4
HU8-T6	Implementación de la funcionalidad en el cliente web	Kevin Castillo Pérez	Terminada	4
HU9-T1	Implementación de los repositorios para el acceso a datos	Kevin Castillo Pérez	Terminada	2
HU9-T2	Implementación de la lógica de la funcionalidad	Kevin Castillo Pérez	Terminada	2
HU9-T3	Creación de los *endpoints*	Kevin Castillo Pérez	Terminada	2
HU9-T4	Diseño e implementación de los DTO (Data Transfer Object)	Kevin Castillo Pérez	Terminada	2
HU9-T5	Diseño del prototipado de la interfaz	Kevin Castillo Pérez	Terminada	4
HU9-T6	Implementación de la funcionalidad en el cliente web	Kevin Castillo Pérez	Terminada	4
HU10-T1	Implementación de los repositorios para el acceso a datos	Kevin Castillo Pérez	Terminada	2
HU10-T2	Implementación de la lógica de la funcionalidad	Kevin Castillo Pérez	Terminada	2
HU10-T3	Creación de los *endpoints*	Kevin Castillo Pérez	Terminada	2
HU10-T4	Diseño e implementación de los DTO (Data Transfer Object)	Kevin Castillo Pérez	Terminada	2
HU10-T5	Diseño del prototipado de la interfaz	Kevin Castillo Pérez	Terminada	4
HU10-T6	Implementación de la funcionalidad en el cliente web	Kevin Castillo Pérez	Terminada	4

Tabla 38: Sprint Backlog #2

Fuente: Los Autores

Sprint #2				
ID	**Tarea**	**Asignada a**	**Estado**	**Tiempo**
HU18-T1	Diseño e implementación de clases POCO (*Plain Old CLR Objects*)	Kevin Castillo Pérez	Terminada	2
HU18-T2	Migración a la base de datos	Kevin Castillo Pérez	Terminada	2
HU18-T3	Implementación de los repositorios para el acceso a datos	Kevin Castillo Pérez	Terminada	4
HU18-T4	Implementación de la lógica de la funcionalidad	Kevin Castillo Pérez	Terminada	4
HU18-T5	Creación de los *endpoints*	Kevin Castillo Pérez	Terminada	2
HU18-T6	Diseño e implementación de los DTO (Data Transfer Object)	Kevin Castillo Pérez	Terminada	2
HU18-T7	Diseño del prototipado de la interfaz	Kevin Castillo Pérez	Terminada	4
HU18-T8	Implementación de la funcionalidad en el cliente web	Kevin Castillo Pérez	Terminada	6
HU17-T1	Diseño e implementación de clases POCO *(Plain Old CLR Objects*)	Kevin Castillo Pérez	Terminada	2
HU17-T2	Migración a la base de datos	Kevin Castillo Pérez	Terminada	2
HU17-T3	Implementación de los repositorios para el acceso a datos	Kevin Castillo Pérez	Terminada	4
HU17-T4	Implementación de la lógica de la funcionalidad	Kevin Castillo Pérez	Terminada	4
HU17-T5	Creación de los *endpoints*	Kevin Castillo Pérez	Terminada	2

HU17-T6	Diseño e implementación de los DTO (Data Transfer Object)	Kevin Castillo Pérez	Terminada	2
HU17-T7	Diseño del prototipado de la interfaz	Kevin Castillo Pérez	Terminada	4
HU17-T8	Implementación de la funcionalidad en el cliente web	Kevin Castillo Pérez	Terminada	6
HU19-T1	Implementación de los repositorios para el acceso a datos	Kevin Castillo Pérez	Terminada	2
HU19-T2	Implementación de la lógica de la funcionalidad	Kevin Castillo Pérez	Terminada	2
HU19-T3	Creación de los *endpoints*	Kevin Castillo Pérez	Terminada	2
HU19-T4	Diseño e implementación de los DTO (Data Transfer Object)	Kevin Castillo Pérez	Terminada	2
HU19-T5	Diseño del prototipado de la interfaz	Kevin Castillo Pérez	Terminada	4
HU19-T6	Implementación de la funcionalidad en el cliente web	Kevin Castillo Pérez	Terminada	4
HU20-T1	Implementación de los repositorios para el acceso a datos	Kevin Castillo Pérez	Terminada	2
HU20-T2	Implementación de la lógica de la funcionalidad	Kevin Castillo Pérez	Terminada	2
HU20-T3	Creación de los *endpoints*	Kevin Castillo Pérez	Terminada	2
HU20-T4	Diseño e implementación de los DTO (Data Transfer Object)	Kevin Castillo Pérez	Terminada	2
HU20-T5	Diseño del prototipado de la interfaz	Kevin Castillo Pérez	Terminada	4
HU20-T6	Implementación de la funcionalidad en el cliente web	Kevin Castillo Pérez	Terminada	4

HU21-T1	Implementación de los repositorios para el acceso a datos	Kevin Castillo Pérez	Terminada	2
HU21-T2	Implementación de la lógica de la funcionalidad	Kevin Castillo Pérez	Terminada	2
HU21-T3	Creación de los *endpoints*	Kevin Castillo Pérez	Terminada	2
HU21-T4	Diseño e implementación de los DTO (Data Transfer Object)	Kevin Castillo Pérez	Terminada	2
HU21-T5	Diseño del prototipado de la interfaz	Kevin Castillo Pérez	Terminada	4
HU21-T6	Implementación de la funcionalidad en el cliente web	Kevin Castillo Pérez	Terminada	4
HU23-T1	Implementación de los repositorios para el acceso a datos	Kevin Castillo Pérez	Terminada	2
HU23-T2	Implementación de la lógica de la funcionalidad	Kevin Castillo Pérez	Terminada	2
HU23-T3	Creación de los *endpoints*	Kevin Castillo Pérez	Terminada	2
HU23-T4	Diseño e implementación de los DTO (Data Transfer Object)	Kevin Castillo Pérez	Terminada	2
HU23-T5	Diseño del prototipado de la interfaz	Kevin Castillo Pérez	Terminada	4
HU23-T6	Implementación de la funcionalidad en el cliente web	Kevin Castillo Pérez	Terminada	4
HU22-T1	Implementación de los repositorios para el acceso a datos	Kevin Castillo Pérez	Terminada	2
HU22-T2	Implementación de la lógica de la funcionalidad	Kevin Castillo Pérez	Terminada	2
HU22-T3	Creación de los *endpoints*	Kevin Castillo Pérez	Terminada	2

HU22-T4	Diseño e implementación de los DTO (Data Transfer Object)	Kevin Castillo Pérez	Terminada	2
HU22-T5	Diseño del prototipado de la interfaz	Kevin Castillo Pérez	Terminada	4
HU22-T6	Implementación de la funcionalidad en el cliente web	Kevin Castillo Pérez	Terminada	4

Tabla 39: Sprint Backlog #3

Fuente: Los Autores

Sprint #3				
ID	**Tarea**	**Asignada a**	**Estado**	**Tiempo**
HU24-T2	Implementación de la lógica de la funcionalidad	Kevin Castillo Pérez	Pendiente	2
HU24-T3	Creación de los *endpoints*	Kevin Castillo Pérez	Pendiente	2
HU24-T4	Diseño e implementación de los DTO (Data Transfer Object)	Kevin Castillo Pérez	Pendiente	2
HU24-T5	Diseño del prototipado de la interfaz	Kevin Castillo Pérez	Pendiente	4
HU24-T6	Implementación de la funcionalidad en el cliente web	Kevin Castillo Pérez	Pendiente	4
HU25-T1	Implementación de los repositorios para el acceso a datos	Kevin Castillo Pérez	Pendiente	2
HU25-T2	Implementación de la lógica de la funcionalidad	Kevin Castillo Pérez	Pendiente	2
HU25-T3	Creación de los *endpoints*	Kevin Castillo Pérez	Pendiente	2
HU25-T4	Diseño e implementación de los DTO (Data Transfer Object)	Kevin Castillo Pérez	Pendiente	2

HU25-T5	Diseño del prototipado de la interfaz	Kevin Castillo Pérez	Pendiente	4
HU25-T6	Implementación de la funcionalidad en el cliente web	Kevin Castillo Pérez	Pendiente	4
HU26-T1	Implementación de los repositorios para el acceso a datos	Kevin Castillo Pérez	Pendiente	2
HU26-T2	Implementación de la lógica de la funcionalidad	Kevin Castillo Pérez	Pendiente	2
HU26-T3	Creación de los *endpoints*	Kevin Castillo Pérez	Pendiente	2
HU26-T4	Diseño e implementación de los DTO (Data Transfer Object)	Kevin Castillo Pérez	Pendiente	2
HU26-T5	Diseño del prototipado de la interfaz	Kevin Castillo Pérez	Pendiente	4
HU26-T6	Implementación de la funcionalidad en el cliente web	Kevin Castillo Pérez	Pendiente	4
HU27-T1	Implementación de los repositorios para el acceso a datos	Michel Suarez	Pendiente	2
HU27-T2	Implementación de la lógica de la funcionalidad	Kevin Castillo Pérez	Pendiente	2
HU27-T3	Creación de los *endpoints*	Kevin Castillo Pérez	Pendiente	2
HU27-T4	Diseño e implementación de los DTO (Data Transfer Object)	Kevin Castillo Pérez	Pendiente	2
HU27-T5	Diseño del prototipado de la interfaz	Kevin Castillo Pérez	Pendiente	4
HU27-T6	Implementación de la funcionalidad en el cliente web	Kevin Castillo Pérez	Pendiente	4
HU28-T1	Implementación de los repositorios para el acceso a datos	Kevin Castillo Pérez	Pendiente	2

HU28-T2	Implementación de la lógica de la funcionalidad	Kevin Castillo Pérez	Pendiente	2
HU28-T3	Creación de los *endpoints*	Kevin Castillo Pérez	Pendiente	2
HU28-T4	Diseño e implementación de los DTO (Data Transfer Object)	Kevin Castillo Pérez	Pendiente	2
HU28-T5	Diseño del prototipado de la interfaz	Kevin Castillo Pérez	Pendiente	4
HU28-T6	Implementación de la funcionalidad en el cliente web	Kevin Castillo Pérez	Pendiente	4