Universidad de las Ciencias Informáticas

Facultad 2 y CITEC

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas.

Herramienta para la gestión de procesamientos de datos de neurociencias en el Centro de Neurociencias de Cuba

Autores:

Michel Suárez Morales

Brian Pérez López

Tutor:

Dr. C Arturo Orellana García

La Habana, noviembre de 2022

“Año 64 de la Revolución”

Declaración de autoría

Declaramos ser los únicos autores del trabajo de diploma *"Herramienta para la gestión de procesamientos de datos de neurociencias en el Centro de Neurociencias de Cuba"*, concedemos 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 conste firmo el presente documento a los 4 días del mes de noviembre del año 2022

Michel Suárez Morales

_____
Firma del Autor

Brian Pérez López

_____
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.

Dedicatoria

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

Agradecimientos

A mi familia por darme todo su apoyo y quererme por sobre todas las cosas. Gracias por ayudarme a cumplir mis objetivos como persona y estudiante. A mi padre por brindarme los recursos necesarios y estar a mi lado apoyándome y aconsejándome siempre. A mi madre por hacer de mí una mejor persona a través de sus consejos, enseñanzas y amor. A mi hermano por estar siempre presente, acompañándome para poderme realizar. A mis amistades por desvelarse conmigo para ayudarme a terminar la tesis. A Arturo por guiarme y enseñarme muchas cosas importantes de mi formación profesional aun teniendo sus propias responsabilidades en otras áreas. A Osbel por siempre estar a mi lado apoyándome y brindándome su amistad sincera. A Arianna por ser mi amiga incondicional y ayudarme en todo lo posible. A Gambou por enseñarme diferentes maneras de ver y disfrutar la vida.

Resumen

La neurociencia es una ciencia multidisciplinaria que estudia el sistema nervioso y todos sus aspectos. En este campo se genera una cuantiosa cantidad de información de investigaciones neurológicas, las cuales pueden ser procesadas gracias a la evolución de las infraestructuras computacionales y las nuevas tecnologías, y mediante el uso de la Computación de alto rendimiento (HPC). El Centro de Neurociencias de Cuba (CNEURO) es la principal institución de investigación de neurociencia en el país. La entidad dedica parte de sus esfuerzos al análisis y procesamiento de datos de neurociencias para comprender la estructura y función del cerebro. La institución presenta un conjunto de problemas entre los más significativos se pueden relacionar: la ejecución de tareas en el clúster de forma manual, los errores ocasionados por el factor humano en la ejecución de instrucciones y la inexperiencia de los especialistas en el trabajo con servidores y línea de comandos.

Es por ello que el objetivo de la presente investigación fue desarrollar una herramienta para la gestión de procesamientos de datos de neurociencias en el Centro de Neurociencias de Cuba, que mejore la usabilidad del proceso de lanzamiento de tareas en el clúster. Para su desarrollo se emplearon las siguientes tecnologías y herramientas: Visual Paradigm, Python, FastAPI, Javascript, Vue.js, WebSocket, UML, entre otras. Se validó la usabilidad de la herramienta a partir de la aplicación del método pre-experimental, demostrando resultados satisfactorios en sus cinco atributos, dando cumplimiento al objetivo planteado en la presente investigación.

Palabras clave: BrainSSys, clúster, neurociencia, procesamiento.

Abstract

Neuroscience is a multidisciplinary science that studies the nervous system and all its aspects. This field generates a large amount of information from neurological research, which can be processed thanks to the evolution of computational infrastructures and new technologies, and through the use of High-Performance Computing (HPC). The Center for Neurosciences of Cuba (CNEURO) is the main neuroscience research institution in the country. It devotes much of its efforts to the analysis and processing of neurological data to understand the structure and function of the brain. The institution presents a set of problems, among the most significant of which are: the execution of tasks in the cluster manually, the errors caused by the human factor in the execution of instructions and the inexperience of specialists in working with servers and command line.

Therefore, the objective of this research was to develop a tool for the management of neuroscience data processing at the Neuroscience Center of Cuba, which improves the usability of the process of launching tasks in the cluster. The following technologies and tools were used for its development: Visual Paradigm, Python, FastAPI, Javascript, Vue.js, WebSocket, UML and others. The usability of the tool was validated through the application of the pre-experimental method, demonstrating satisfactory results in its five attributes, fulfilling the objective set out in this research.

Keywords: BrainSSys, cluster, Neuroscience, processing

TABLA DE CONTENIDOS

Índice de figuras

Índice de tablas

Introducción

La neurociencia es un campo de la ciencia que estudia el sistema nervioso y todos sus aspectos; como podrían ser su estructura, función, desarrollo ontogenético y filogenético, bioquímica, farmacología y patología; y de cómo sus diferentes elementos interactúan, dando lugar a las bases biológicas de la cognición y la conducta. (David Bueno, 2018) Esta tiene como propósito entender cómo el encéfalo produce la marcada individualidad de la acción humana, es aportar explicaciones de la conducta en términos de actividades del encéfalo, explicar cómo actúan millones de células nerviosas individuales en el encéfalo para producir la conducta y cómo, a su vez, estas células están influidas por el medio ambiente, incluyendo la conducta de otros individuos. (Kandel et al., 1997)

Recientes desarrollos en las ciencias del cerebro permiten, como nunca antes en la historia, medir, registrar, alterar y/o manipular la actividad del cerebro. Avances similares en ingeniería de sistemas y microcircuitos ha visto nacer a la neurotecnología. “La neurotecnología es la confluencia entre la Inteligencia Artificial, ciencias de la computación y neurociencia, esta constituye un ejemplo claro de tecnociencia convergente de alto potencial innovador y disruptivo. Por neurotecnología se puede entender cualquier herramienta o técnica capaz de manipular, registrar, medir y obtener información del cerebro. La información obtenida puede permitir dar soluciones en contextos clínicos o satisfacer el deseo de curiosidad de la ciencia básica”. (Ausín et al., 2020)

Dentro de esta ciencia se pueden identificar los neurodatos, los cuáles son el conjunto de información relativa a la actividad cerebral obtenida mediante el empleo de neurotecnologías avanzadas tales como FMRI (imágenes por resonancia magnética funcional), EEG (Electroencefalograma), PET (Tomografía de emisión de positrones), entre otras. (Antonio Bardají Gálvez, 2021)

La neuroinformática es un término utilizado en relación al campo de investigación que combina la investigación en neurociencia e informática para desarrollar herramientas innovadoras destinadas a la organización de datos neurocientíficos de gran volumen y alta dimensión. También se aplica modelos computacionales para integrar y analizar estos datos para eventualmente comprender la estructura y función del cerebro. Llegó el nacimiento del campo de la neuroinformática a partir de un estudio realizado por el Nacional Academia de Ciencias, cuyo propósito fue evaluar la necesidad de crear bases de datos para compartir datos primarios de investigación en neurociencia. El estudio concluyó que la capacidad de la tecnología de la información para manejar complejidad de tales datos había madurado, y que tal programa desempeñaría un papel fundamental en la comprensión el desarrollo y la función normal del cerebro, así como el diagnóstico, tratamiento y prevención del sistema nervioso desórdenes.

Un 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. 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. (Revista Digital Universitaria - UNAM, 2022)

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, C-BIG, BIL (*Brain Image Library*) y LORIS (*Longitudinal Online Research and Imaging System*).

CBRAIN (*Canadian Brain Imaging Research Platform*) es un sistema web que permite a los investigadores de neurodatos realizar análisis computacionalmente intensivos de datos al conectarlos a instalaciones de Computación de alto rendimiento (HPC) en Canadá y en todo el mundo.(MCIN, 2022a)

Por su parte, LORIS (Sistema Longitudinal de Investigación e Imágenes en Línea) es un software de gestión de proyectos y datos basado en la web para estudios de investigación de neuroimagen. Es un marco de código abierto para almacenar y procesar datos de comportamiento, clínicos, de neuroimagen y genéticos. LORIS también facilita la gestión de grandes conjuntos de datos adquiridos a lo largo del tiempo en un estudio longitudinal o en diferentes ubicaciones en un gran estudio de varios sitios.(MCIN, 2022b)

El Centro de Neurociencias de Cuba (CNEURO) es una institución cubana de investigación y desarrollo dedicado a la investigación traslacional 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, 2022)

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, 2022)

En una entrevista realizada a los especialistas encargados de mantener el clúster y a los investigadores 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 se ven obligados a programar instrucciones en consola de forma manual para ejecutar tareas de procesamiento y análisis.
- En ocasiones se debe repetir el proceso debido a errores 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 investigación**: ¿Cómo mejorar la usabilidad del proceso de lanzamiento de tareas en el clúster del Centro Nacional de Neurociencias de Cuba?

Se define Usabilidad en el contexto de la investigación a la capacidad de un software para ser entendido, aprendido, usado y resultar atractivo para el usuario.

Objetivo general: Desarrollar una herramienta para el clúster de CNEURO que mejore la usabilidad en la gestión de las tareas de procesamiento de datos de neurociencias.

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

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. Diagnóstico de las herramientas que realicen el procesamiento y lanzamiento de tareas en el Clúster de CNEURO.
3. Desarrollo de los componentes y funcionalidades para administrar los usuarios y los permisos, así como la gestión de las diferentes tareas a ejecutar y la configuración de clúster.
4. Integración de los componentes desarrollados a la plataforma BrainsSys.
5. Validación de los resultados obtenidos aplicando los métodos definidos en 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.

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.

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.

Entre los **métodos empíricos** utilizados tenemos 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 que mejore la usabilidad del proceso de lanzamiento de tareas en el clúster para facilitar el trabajo que desarrollan los especialistas de la institución.

El documento está estructurado en tres capítulos, siendo estos:

El **Capítulo 1. Fundamentación teórica de la investigación**, trata los conceptos fundamentales sobre los diferentes elementos en la que se basa la presente investigación, incluye un análisis del estado del arte, a nivel internacional de sistemas que permitan realizar la gestión de procesamiento de tareas de Neurociencias, y se hace una descripción de las tecnologías y herramientas usadas para el desarrollo de la herramienta propuesta.

El **Capítulo 2. Análisis y diseño de la herramienta para la gestión de procesamiento de datos de neurociencias**, se describe la propuesta de solución y los componentes principales que fueron desarrollados. Para ello, se parte desde el análisis de los procesos de negocio en el Departamento de Informática de CNEURO. Teniendo en cuenta la descripción y modelación, se especifican los requisitos del sistema y se llega a la definición de estos mediante los artefactos de especificación de requisitos. Se realiza una descripción de las funcionalidades utilizando historias de usuarios. Se desarrollan las descripciones de la arquitectura de software y los patrones de diseño utilizados y, además, se muestra el modelo de datos propuesto.

Por último, en el **Capítulo III**. **Validación e implementación de la propuesta de solución**, en este capítulo se muestra todo lo relacionado a la implementación de la herramienta. Se elabora el diagrama de despliegue y, además, se hace referencia a la seguridad e integridad de los datos que se manejan.

Se muestran las estrategias y los estándares de codificación utilizados. Se realiza además la declaración de integración y la validación.

CAPÍTULO 1. Fundamentación teórica de la investigación

El presente capítulo tiene como objetivo abordar los diferentes elementos que brindan la base teórica y conceptual para el desarrollo de la investigación. Valorar de forma crítica las tendencias y tecnologías actuales, así como los antecedentes asociados al campo de acción. Se realiza un análisis de las características de los diversos repositorios de datos neurológicos existentes a nivel internacional. De este modo, se podrá realizar una correcta interpretación de la situación problemática y del problema a resolver.

1.1 Conceptos asociados al problema

Para obtener una mejor comprensión del dominio del problema, el presente epígrafe analiza los conceptos asociados al mismo.

1.1.1 Neurociencias

Las neurociencias son las ciencias multidisciplinarias que analizan el sistema nervioso para comprender las bases biológicas del comportamiento (estructura, función, desarrollo ontogenético y filogenético, bioquímica, farmacología y patología). Los neuroanatomistas estudiaron la forma del cerebro, su estructura celular y sus circuitos; los neuroquímicos estudiaron la composición química del cerebro, sus lípidos proteínas; los neurofisiólogos estudiaron las propiedades bioeléctricas del cerebro; y los psicólogos y neuropsicólogos investigaron la organización y sustratos neurales del comportamiento y la cognición.

El término neurociencia se introdujo a mediados de la década de 1960, para señalar el comienzo de una era en la que cada una de las disciplinas que las conforman trabajarían juntas de forma cooperativa, compartiendo un lenguaje común, conceptos comunes y un objetivo común: comprender la estructura y la función del cerebro normal y anormal. En la actualidad, la neurociencia abarca una amplia gama de investigaciones, desde la biología molecular de las células nerviosas (es decir, los genes que codifican las proteínas necesarias para el funcionamiento del sistema nervioso) hasta las bases biológicas del comportamiento normal y desordenado, la emoción y la cognición (es decir, las propiedades mentales con la que los individuos interactúan entre sí y con su entorno).

La neurociencia es actualmente una de las áreas científicas que crecen más rápidamente. De hecho, a veces se habla del cerebro como la última frontera de la biología. En 1971, 1100 científicos se reunieron en la primera reunión anual de la Sociedad de Neurociencia. En 2006, 25.785 científicos

participaron en la 36ª reunión anual de la sociedad, en la que se presentaron 14.268 investigaciones. (Squire et al., 2012)

1.1.1.1 Neurotecnologías

A lo largo de la historia, las nuevas herramientas han impulsado revoluciones científicas en múltiples disciplinas. Para desentrañar cómo el cerebro humano procesa la información y sustenta una amplia gama de comportamientos, se requiere una revolución científica de este tipo y un arsenal de herramientas de nueva generación que los neurocientíficos puedan desplegar en los ámbitos básico, y clínico. Capturar el registro completo de la actividad neuronal o lo que se ha llamado un "mapa de actividad cerebral" a partir de dominios espaciales y temporales es un desafío tecnológico de enormes proporciones que debe ser superado para lograr una comprensión más completa de los procesos cerebrales fundamentales y patológicos. Lanzada el 2 de abril de 2013, el objetivo clave de la iniciativa *Brain Research through Advancing Innovative Neurotechnologies* (BRAIN) es desarrollar tecnologías innovadoras para interrogar cómo interactúan las células y los circuitos del cerebro a la velocidad del pensamiento y, en última instancia, revelar los complejos vínculos entre la función cerebral y el comportamiento. Dado que la arquitectura del cerebro se extiende desde la escala de las interacciones moleculares en los billones de sinapsis hasta los miles de millones de cuerpos celulares que se conectan para formar redes locales que luego se integran en múltiples regiones del cerebro, los retos son extraordinarios. Además de estas escalas espaciales, existen escalas temporales, debido a que los circuitos cerebrales no son estáticos, sino que cambian continuamente como resultado de la actividad neuronal, la etapa de desarrollo y el envejecimiento. A pesar de esta complejidad, las tecnologías surgidas de la Iniciativa BRAIN están abriendo nuevas puertas para descifrar cómo el cerebro registra, procesa, utiliza, almacena y recupera crecientes cantidades de información. Tienen el potencial de facilitar un salto cuántico hacia la comprensión de la función cerebral y su alteración en la enfermedad, haciendo que las anormalidades del circuito sean la base del diagnóstico y la normalización de la función del circuito el objetivo de la futura intervención en los trastornos neuro/mentales/de abuso de sustancias. (Meghan C. Mott et al., 2018)

Una definición a lo anterior es la Neurotecnología. Es la confluencia entre la Inteligencia Artificial, ciencias de la computación y neurociencia, esta constituye un ejemplo claro de tecnociencia convergente de alto potencial innovador y disruptivo. Son el conjunto de métodos e instrumentos que permiten una conexión directa de dispositivos técnicos con el sistema nervioso, herramientas que sirven para analizar e influir sobre el sistema nervioso del ser humano, especialmente sobre el cerebro. Estas tecnologías incluyen simulaciones de modelos neurales, computadores biológicos y aparatos

para interconectar el cerebro con sistemas electrónicos, con el objetivo medir y analizar la actividad cerebral. (Müller & Rotter, 2017)

1.1.1.2 Neuroinformática

La neuroinformática es un área de la ciencia cuyo objetivo es integrar los datos de la neurociencia y desarrollar modernas herramientas informáticas para aumentar la comprensión de las funciones del sistema nervioso en la salud y la enfermedad. Las herramientas neuroinformáticas incluyen, entre otras, bases de datos para almacenar y compartir datos, repositorios para gestionar documentos y código fuente, y herramientas de software para analizar, modelar y simular señales e imágenes.

Esta ciencia combina la neurociencia y la informática relacionada con la neurociencia para desarrollar y aplicar herramientas y enfoques avanzados esenciales para comprender la estructura y la función del cerebro. También se considera que la neuroinformática engloba la bioinformática más tradicional y la investigación de biología de sistemas computacional recientemente establecida para la neurociencia. El campo de la neuroinformática ha crecido rápidamente hasta implicar el desarrollo de modernas herramientas basadas en las tecnologías de la información y la comunicación (TIC).

La neuroinformática reúne a científicos de todo el mundo con experiencia en informática, ingeniería de software, física, química, matemáticas, teoría de sistemas dinámicos y neurociencia. Se espera que las bases de datos de libre acceso, los modelos y los entornos de simulación de fácil manejo faciliten la investigación de funciones cerebrales complejas. En el futuro, se hará especial hincapié en el desarrollo de simuladores fáciles de usar e interoperables con otras herramientas neuroinformáticas. (Linne, 2018)

1.1.2 Procesamiento de datos de neurociencia

1.1.2.1 Computación distribuida

El término “computación distribuida” (en inglés, *distributed computing*) describe una infraestructura digital en la que una red de ordenadores conectados resuelve tareas computacionales entrantes. A pesar de que estén separados en el espacio, los ordenadores autónomos funcionan en estrecha colaboración en un proceso basado en la división de tareas. Además de los ordenadores y estaciones de trabajo más potentes del sector profesional, también pueden integrarse miniordenadores y ordenadores de escritorio de usuarios domésticos. Al estar físicamente separado, el hardware distribuido no puede utilizar una memoria compartida, así que los ordenadores participantes transmiten mensajes y datos (por ejemplo, resultados de los cálculos) a través de una red. La comunicación entre

máquinas tiene lugar a nivel local mediante una intranet (por ejemplo, en un centro de computación) o a nivel nacional y mundial a través de internet. El transporte de los mensajes se lleva a cabo mediante protocolos de internet como TCP/IP y UDP.(know-how - IONOS, 2020)

Tipos de la Computación Distribuida

La computación distribuida es un fenómeno multifacético con infraestructuras que, en ocasiones, difieren mucho. Por ello, no es fácil describir todas las variantes de la computación distribuida. Sin embargo, existen tres subcampos que se describen a menudo en el ámbito de la informática:

- Computación en la nube o *cloud computing*

  En la computación en la nube, se utiliza la computación distribuida para proporcionar a los clientes infraestructuras y plataformas altamente escalables y rentables. Los proveedores de la nube suelen ofrecer capacidad en forma de servicios alojados a los que se puede acceder a través de internet. (know-how - IONOS, 2020)

- Computación en malla o *grid computing*

  Está orientada conceptualmente a la creación de un superordenador con una enorme potencia de cálculo. Sin embargo, las tareas de computación no son procesadas por una, sino por muchas instancias. Los servidores y los PC pueden realizar diferentes tareas de manera independiente. Al realizar las tareas, la computación en malla puede acceder a los recursos de manera muy flexible. En general, los participantes ponen sus capacidades informáticas a disposición de un proyecto general por la noche, cuando su infraestructura técnica se utiliza relativamente poco.(know-how - IONOS, 2020)

- Computación en clúster o *cluster computing*

  La computación en clúster no tiene una definición claramente distinguible de las variantes de computación en la nube y en malla. Es un término más general, que se refiere a todas las modalidades que combinan ordenadores individuales y sus capacidades informáticas en un clúster (es decir, “grupo” o “conjunto”). Por ejemplo, hay clústeres de servidores, clústeres en entornos de *big data* y en la nube, clústeres de bases de datos y clústeres de aplicaciones. Además, las redes informáticas se utilizan cada vez más en la computación de alto rendimiento, que resuelve problemas informáticos particularmente complejos.(know-how - IONOS, 2020)

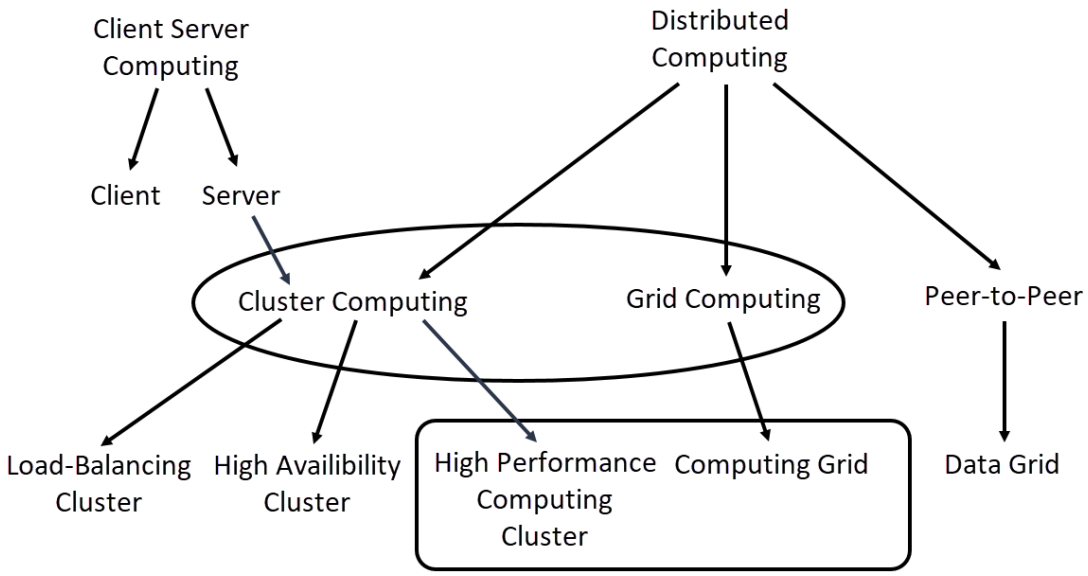


Figura 1-1 Representación de la Computación distribuida.
Fuente: (Sebastian Gräf, 2021)

Ventajas y Desventajas de la Computación Distribuida

Usar la Computación Distribuida tiene muchas ventajas, como:

- Se utilizan ordenadores económicos con los microprocesadores habituales en vez de costosos superordenadores
- Es fácil compensar el fallo de los componentes individuales
- Carga de computación compartida, permitiendo el equilibrio de carga
- Flexibilidad
- Permite el trabajo en la nube
- Escalabilidad

A pesar de sus muchas ventajas, también tiene algunos inconvenientes:

- Mayor costo de implementación y mantenimiento de una arquitectura de sistema compleja.

- En cuanto a la fiabilidad, la estrategia descentralizada tiene algunas ventajas respecto a usar una sola instancia de procesamiento. Sin embargo, al mismo tiempo, la computación distribuida puede dar lugar a problemas de seguridad**.**

- Por lo general, las infraestructuras distribuidas son también más propensas a errores, ya que hay más interfaces y posibles causas de problemas a nivel de hardware y software.

- La complejidad de la infraestructura dificulta el diagnóstico de dichos problemas y errores.

Modelo Arquitectónico

El objetivo general de este tipo de modelo es garantizar el reparto de responsabilidades entre componentes del sistema distribuido y la ubicación de dichos componentes. Las principales preocupaciones son determinar la relación entre procesos y hacer al sistema confiable, adaptable y rentable. Entre los modelos de arquitectura ampliados de los sistemas distribuidos están:

- Modelo de cliente-servidor

Modelo de diseño de software en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes. Los clientes realizan peticiones al servidor, otro programa, que le da una respuesta. Pero también un servidor puede ser cliente de otros servidores. Un buen ejemplo sería un servidor web, que es un cliente de servidor DNS. (Coulouris et al., 2011)

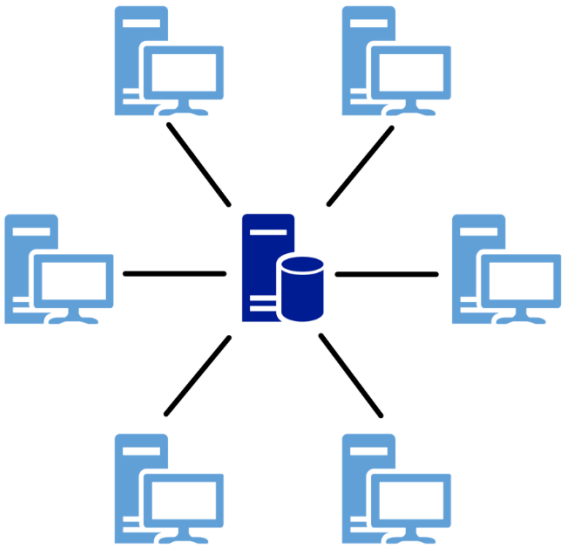


Figura 1-2 Modelo Cliente-Servidor.
Fuente: (Sebastian Gräf, 2021)

- Modelo *peer to peer*

Organiza la interacción y la comunicación de la computación distribuida a partir de aspectos descentralizados. Todos los ordenadores (también llamados nodos) tienen los mismos privilegios y realizan las mismas tareas y funciones en la red. Cada ordenador puede actuar como cliente y como servidor. Un ejemplo de arquitectura *peer to peer* es la *blockchain* de criptomonedas.(know-how - IONOS, 2020)

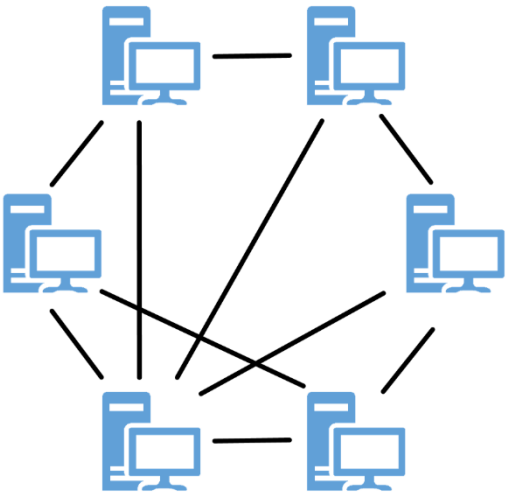


Figura 1-3 Modelo Peer to Peer.
Fuente: (Sebastian Gräf, 2021)

- Modelo de capas (arquitecturas multinivel)

En el diseño conceptual de una arquitectura de capas, los aspectos individuales de un sistema de software se distribuyen en varias capas (en inglés, *tier* o *layer*), aumentando así la eficacia y la flexibilidad de la computación distribuida. Este tipo de arquitectura del sistema, que puede diseñarse como una arquitectura de dos, tres o n capas, según el uso previsto, es bastante común en las aplicaciones web.(know-how - IONOS, 2020)

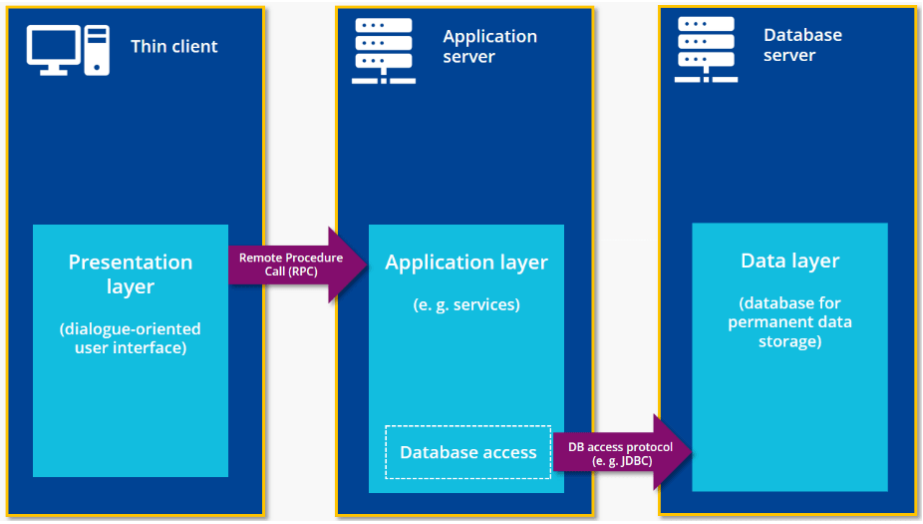


Figura 1-4 Modelo de Capas.
Fuente: (know-how - IONOS, 2020)

- Arquitectura orientada a servicios (SOA, del inglés *service-oriented architecture*)

Se centra en los servicios y en satisfacer las necesidades y procesos individuales de cada empresa. De este modo, se pueden combinar diferentes servicios en un proceso empresarial a medida. Por ejemplo, el proceso global “pedidos en línea”, en el que intervienen los servicios “entrada de pedidos”, “verificación de crédito” y “envío de factura”, está contenido en una SOA. Los componentes técnicos (servidores, bases de datos, etc.) actúan como herramientas, pero no son el centro de atención. En este concepto de computación distribuida, la prioridad es asegurara la agrupación, colaboración y organización eficaces de los servicios en aras de una gestión más eficiente y rápida de los procesos empresariales.(know-how - IONOS, 2020)

1.1.2.2 Clúster de computadoras y HPC

El término clúster (del inglés *cluster*, que significa 'grupo' o 'racimo') se aplica a los sistemas distribuidos de granjas de computadoras unidos entre sí normalmente por una red de alta velocidad y que se comportan como si fuesen un único servidor (FundéuRAE, 2017). 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.

A medida que transcurren los años y avanza la tecnología como el Internet de las cosas (IoT), la Inteligencia Artificial (IA), la ciencia de datos y las imágenes en 3D, crece exponencialmente el tamaño y la cantidad de datos con los que se debe trabajar. Como respuesta a la necesidad de poder procesar cantidades masivas de datos, surge el termino HPC (High Performance Computing).

El termino HPC, o computación de alto rendimiento implica usar la potencia de cálculo para resolver problemas complejos en ciencia, ingeniería y gestión, apoyándose en tecnologías computacionales como los clústeres, los supercomputadores o la computación paralela. Actualmente se asocia el termino HPC a la mayor parte de las ideas de computación distribuida. Por otra parte, el concepto clúster no solo se aplica a computadoras de alto rendimiento (HPC), sino también a los conjuntos de computadoras, construidas con componentes de hardware comunes y software libre.

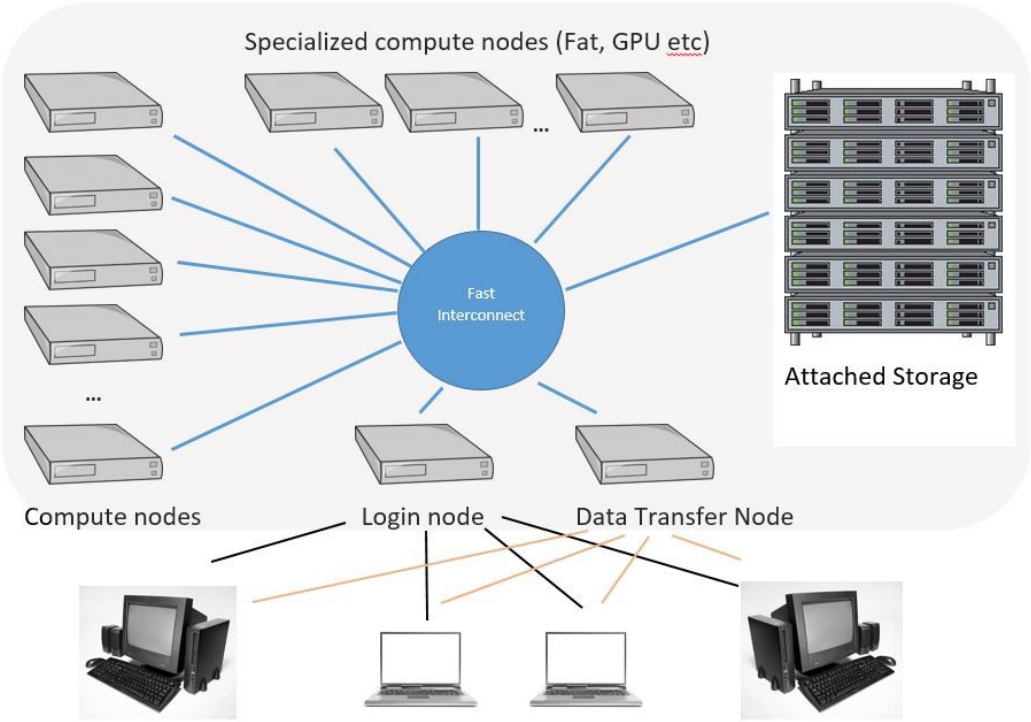


Figura 1-5 Componentes de un Clúster.
Fuente: (Iowa State University, 2022)

Clasificación de un Clúster

Los clústeres pueden clasificarse según sus características:

- HPCC (*High Performance Computing Clusters*: clústeres de alto rendimiento).

  Son clústeres en los cuales se ejecutan tareas que requieren de gran capacidad computacional, grandes cantidades de memoria, o ambos a la vez. El llevar a cabo estas tareas puede comprometer los recursos del clúster por largos periodos de tiempo.

- HA o HACC (*High Availability Computing Clusters*: clústeres de alta disponibilidad).

  Son clústeres cuyo objetivo de diseño es el de proveer disponibilidad y confiabilidad. Estos clústeres tratan de brindar la máxima disponibilidad de los servicios que ofrecen. La confiabilidad se provee mediante software que detecta fallos y permite recuperarse frente a los mismos, mientras que en hardware se evita tener un único punto de fallos.

- HT o HTCC (*High Throughput Computing Clusters*: clústeres de alta eficiencia).

Son clústeres cuyo objetivo de diseño es el ejecutar la mayor cantidad de tareas en el menor tiempo posible. Existe independencia de datos entre las tareas individuales. El retardo entre los nodos del clúster no es considerado un gran problema.

Componentes de un Clúster

Para que un clúster funcione como tal, no basta solo con conectar entre sí los ordenadores, sino que es necesario proveer de ciertos componentes y un sistema de manejo del clúster, el cual se encargue de interactuar con el usuario y los procesos que corren en él para optimizar el funcionamiento.

En general, un clúster necesita de varios componentes de software y hardware para poder funcionar:

- Nodos
- Almacenamiento
- Sistemas operativos
- Conexiones de red
- Middleware
- Protocolos de comunicación y servicios
- Aplicaciones
- Ambientes de programación paralela

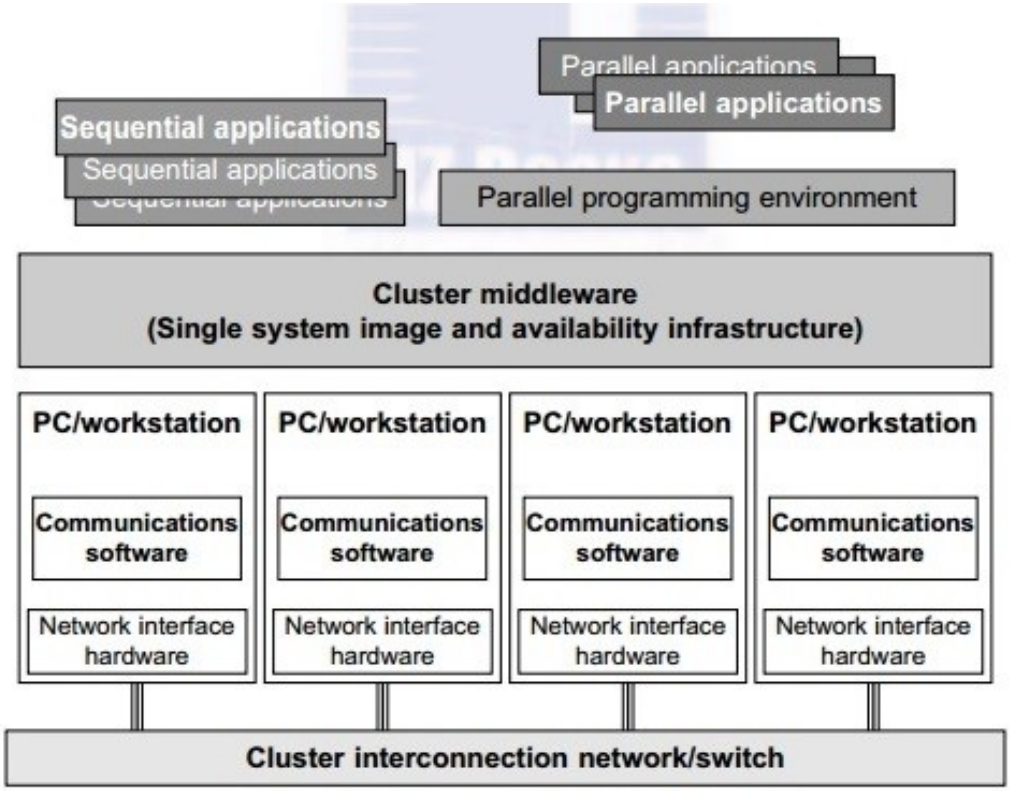


Figura 1.1-6 Componentes de un Nodo en un Clúster.

Fuente: (BrainKart Team, 2022)

1.2 Usabilidad

La norma internacional ISO 25010 hace referencia a la usabilidad y ofrece una definición de su contenido y alcance:

Capacidad del producto software para ser entendido, aprendido, usado y resultar atractivo para el usuario, cuando se usa bajo determinadas condiciones.

1.2.1 Atributos de Usabilidad

La usabilidad es una cualidad demasiado abstracta como para ser medida directamente. Para su estudio se descompone habitualmente en los siguientes cinco atributos básicos (Ferre, 2000):

Facilidad de aprendizaje: cuán fácil es aprender la funcionalidad básica del sistema, como para ser capaz de realizar correctamente la tarea que desea realizar el usuario. Se mide normalmente por el tiempo empleado con el sistema hasta ser capaz de realizar ciertas tareas en menos de un tiempo dado (el tiempo empleado habitualmente por los usuarios expertos). Este atributo es muy importante para usuarios noveles.

Eficiencia: el número de transacciones por unidad de tiempo que el usuario puede realizar usando el sistema. Lo que se busca es la máxima velocidad de realización de tareas del usuario. Cuanto mayor es la usabilidad de un sistema, más rápido es el usuario al utilizarlo, y el trabajo se realiza con mayor rapidez.

Recuerdo en el tiempo: para usuarios intermitentes (que no utilizan el sistema regularmente) es vital ser capaces de usar el sistema sin tener que aprender cómo funciona partiendo de cero cada vez. Este atributo refleja el recuerdo acerca de cómo funciona el sistema que mantiene el usuario, cuando vuelve a utilizarlo tras un periodo de no utilización.

Tasa de errores: este atributo contribuye de forma negativa a la usabilidad de un sistema. Se refiere al número de errores cometidos por el usuario mientras realiza una determinada tarea. Un buen nivel de usabilidad implica una tasa de errores baja. Los errores reducen la eficiencia y satisfacción del usuario, y pueden verse como un fracaso en la transmisión al usuario del modo de hacer las cosas con el sistema.

Satisfacción: este es el atributo más subjetivo. Muestra la impresión subjetiva que el usuario obtiene del sistema.

1.3 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*, 2022), la International Brain Initiative (*IBI*, 2022) 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*, 2022), la Universidad de Maastricht (*Maastricht*, 2022), la Universidad de Aachen (*RWTH-Aachen*, 2022), la Universidad McGill (*McGill*, s. f.) en Canadá, entre otros. También ha participado en proyectos internacionales financiados por el Human Frontier Science Program (*HFSP*, 2022) y algunos proyectos para la Organización

Internacional de Investigación del Cerebro (*IBRO*, 2022). Según el Dr. C Arturo Orellana, 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.

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.

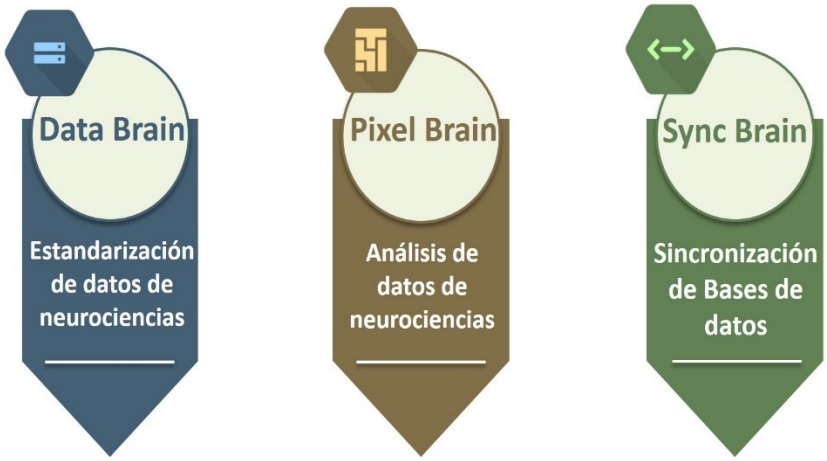


Figura 1-6 Sistemas internos de BrainSSys.
Fuente: BrainSSys.

1.4 Estado del arte de soluciones similares

A lo largo de los años han surgido a nivel mundial producto de alianzas, tratados y convenios diferentes organizaciones, soluciones y plataformas que han logrado la unidad de la mayor parte de la comunidad médica científica para el avance progresivo del desarrollo de la humanidad. A continuación, se muestran diferentes herramientas y plataformas que se considera que se asemeja a la solución deseada.

1.4.1 A nivel de Plataformas

1.4.1.1 *Canadian Brain Imaging Research Platform* (CBRAIN)

Según (Sherif et al., 2014), 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.

Los usuarios se autentican en el sistema accediendo primero a una cuenta privada. Toda la comunicación entre los clientes y la capa de servicios intermedios se realiza a través de una capa de conexión segura (SSL). CBRAIN utiliza un mecanismo de reenvío de agentes SSH bajo demanda para crear canales de comunicación entre los portales, los servidores de ejecución y los proveedores de datos. Esto tiene varias ventajas: elimina los riesgos asociados a las contraseñas o a las claves

privadas ubicadas en cualquier máquina intermedia, minimiza la duración de los túneles abiertos y permite a los administradores de la plataforma controlar cuidadosamente si los desafíos de las claves están asociados a operaciones reales de la plataforma o a posibles actividades sospechosas.

El acceso a todos los recursos de CBRAIN se gestiona mediante tres conceptos centrales: usuarios, proyectos y sitios. Los usuarios representan la cuenta de un usuario de CBRAIN. Una vez autentificados, los usuarios tienen acceso a su entorno y a los recursos a los que tienen acceso. Por defecto, los usuarios sólo tienen acceso a los datos que han añadido a través de su cuenta. La propiedad puede aplicarse a cualquier objeto dentro de CBRAIN. Esto puede incluir datos, trabajos HPC, proyectos, proveedores de datos y varios HPC.

Los proyectos definen el acceso compartido a los recursos. Todos los proveedores de datos, datos, servidores de ejecución y herramientas de CBRAIN están asociados a un proyecto. Los proyectos pueden tener uno o más usuarios como miembros, y los miembros de un determinado proyecto tendrán acceso a los recursos asociados a él. Los usuarios pueden crear y gestionar sus propios proyectos, y por defecto los recursos asociados a estos proyectos estarán disponibles sólo para el creador del proyecto.

Un sitio tendrá usuarios y proyectos asociados, y a uno o más de esos usuarios se les puede dar el papel de administrador del sitio. Un administrador de sitio tiene capacidades de administración para los recursos asociados a un sitio determinado. Pueden crear y gestionar cuentas de usuario, proyectos y otros recursos para su sitio. Básicamente, un sitio crea un subdominio administrativo en CBRAIN sobre el que uno o más administradores de sitio locales pueden tener control.

1.4.1.2 *Longitudinal Online Research and Imaging System* (LORIS)

Según (Das et al., 2011), *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.

LORIS posee dos métodos principales de acceso. Una capa de aplicación web realiza el procesamiento de alto nivel y proporciona una interfaz de usuario basada en la web, en la que los usuarios pueden conectarse a través de un protocolo seguro (SSL) para manipular o ver los datos. En el extremo posterior, los desarrolladores pueden crear herramientas de línea de comandos que operan en las estructuras de datos de la base de datos utilizando la misma interfaz de programación de aplicaciones (API) utilizada para las aplicaciones del extremo frontal. Para garantizar un control total de los accesos, LORIS está equipado con un módulo de cuentas de usuario en la capa de aplicación web, en el que cada usuario recibe una cuenta, que a su vez está vinculada a una serie de permisos. Los administradores del sistema pueden realizar todas las funciones de creación de usuarios, configuración y gestión de permisos directamente a través de la interfaz web, sin tener que acceder directamente al *back-end* de MySQL.

1.4.1.3 *Brain Image Library* (BIL)

La Biblioteca de imágenes cerebrales (*BIL*, 2022) 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.

Permite emitir identificadores de objetos digitales (DOI) para conjuntos de datos (o agrupaciones de conjuntos de datos) de forma manual y, en el futuro, automáticamente en la publicación del conjunto de datos. BIL proporciona soporte para la transferencia de datos, incluido el análisis de red para encontrar cuellos de botella en la transferencia. Además, tiene la capacidad de recibir datos en medios alternativos, incluida la cinta LTO8 y proporciona capacidad de computación de alto rendimiento (HPC) local a los datos para el procesamiento de datos previo al envío y la exploración posterior al envío.

BIL proporciona varias formas de acceder a los datos sin tener que descargarlos:

- Nodo de inicio de sesión BIL: todos los usuarios de los sistemas BIL tienen acceso de *Secure Shell* (SSH) a los nodos de inicio de sesión BIL.
- Sistema BIL VM: El sistema BIL VM es un recurso flexible que se compone de varias máquinas de memoria grandes equipadas con GPUS moderno. El sistema VM tiene capacidad para escritorio remoto, puede albergar comerciales con licencia de usuario y puede albergar puertas

de enlace web para proporcionar vistas de datos específicas del proyecto, admitir visualización basada en web y aplicaciones de computación en el lugar.

- *Bridges-2*: es un sistema informático de alto rendimiento diseñado para admitir software y entornos familiares y convenientes para usuarios de HPC tradicionales y no tradicionales. Su conjunto de sistemas interactivos altamente conectados ofrece una flexibilidad excepcional para el análisis de datos, la simulación, los flujos de trabajo y las puertas de enlace, aprovechando la interactividad, la computación paralela.

NeoCortex: es un recurso altamente innovador que acelerará el descubrimiento científico impulsado por la inteligencia artificial al acortar enormemente el tiempo requerido para el entrenamiento de aprendizaje profundo y otros enfoques de inteligencia artificial.

1.4.1.4 *C-BIG Repository*

El repositorio C-BIG (*C-BIG*, 2022) se lanzó internamente en 2019 como el repositorio de datos clínicos y bioespecímenes de *Canadian Open Neuroscience Platform* (CONP), y públicamente en noviembre de 2020 para la comunidad de investigación en general. Este repositorio se construyó utilizando LORIS para recolectar bioespecímenes y datos clínicos, de imágenes y genéticos de pacientes con enfermedades neurológicas y controles sanos.

El repositorio permite el seguimiento y la difusión de datos de muestras biológicas, sin procesar o derivados, y admite metadatos y estadísticas resumidas en múltiples niveles de control de acceso, incluido el acceso para investigadores genuinos y profesionales de la atención clínica a partir del novedoso modelo de Acceso Registrado. C-BIG presenta las mejores prácticas actuales de intercambio de datos, como FAIR, e incluye captura de procedencia, registro completamente auditable y otros esfuerzos de estandarización.

El repositorio de C-BIG es una combinación de varios componentes: un biobanco digital para el archivo y el intercambio de datos, un repositorio físico de muestras biológicas, un portal web de acceso a varios niveles, un registro de pacientes, una capa de desidentificación y una API para interacciones automatizadas. Juntos, estos presentan un ecosistema que vincula y comparte datos multimodales de numerosos laboratorios y proyectos, que abarcan varias enfermedades neurológicas.

Las precauciones están integradas en el sistema para garantizar la integridad de los datos que ingresan al repositorio digital, de modo que todos los campos deben pasar por un proceso de validación de múltiples capas para reducir el riesgo de error humano. La primera capa de validación se basa en la visualización condicional de campos según el procedimiento estándar seleccionado. Esto,

en combinación con el sistema de permisos LORIS, evita que el usuario ingrese datos de muestras incoherentes con el tipo de muestra, el protocolo de procesamiento o el sitio del paciente y las afiliaciones del proyecto. La segunda capa es el código JavaScript y HTML del lado del cliente, donde se marcan los errores de tipo de datos y se validan las listas y los rangos.

La tercera capa implica la comparación cruzada de entradas con los valores de la base de datos en todas las demás muestras, así como la validación redundante de tipos de datos para evitar intentos maliciosos de corrupción de datos. Una vez enviadas, las muestras no válidas se marcan y rechazan con un mensaje de error descriptivo que se proporciona al usuario y, por lo tanto, nunca ingresan a la base de datos. Por el contrario, las muestras válidas que ingresan a la base de datos se rastrean a través de registros de auditoría que se respaldan regularmente. Finalmente, dado que la validación automatizada no siempre puede prevenir errores de entrada humana, es un procedimiento estándar para el equipo de laboratorio de C-BIG revisar regularmente las entradas en el sistema y cotejarlas con los registros de recolección y procesamiento del laboratorio.

1.4.1.5 NeuroVault

NeuroVault (*NeuroVault*, 2022) es una plataforma web que permite a los investigadores almacenar, compartir, visualizar y decodificar mapas del cerebro humano. Este nuevo recurso puede mejorar la forma en que se presentan, difunden y reutilizan los experimentos de cartografía del cerebro humano. El repositorio facilita el depósito y el intercambio de mapas estadísticos, proporciona una visualización atractiva y una decodificación cognitiva de los mapas que pueden mejorar los esfuerzos de colaboración y la legibilidad de los resultados. Al mismo tiempo, también proporciona una API para que los investigadores de métodos descarguen los datos, realicen análisis potentes o creen nuevas herramientas (Gorgolewski et al., 2015).

Una de las características clave de NeuroVault es la facilidad para cargar y compartir mapas cerebrales estadísticos. Después de iniciar sesión, los usuarios pueden cargar una amplia gama de imágenes de neuroimagen y metadatos asociados. Luego, estos datos son configuraciones de privacidad controladas por el usuario de acceso inmediato a través de una interfaz interactiva basada en HTML y una API web REST-full integral que facilita la interoperabilidad programática con otros recursos.

El proceso de carga de NeuroVault enfatiza la velocidad y la facilidad de uso. Los usuarios pueden confiar en las cuentas de redes sociales existentes (Google o Facebook) para iniciar sesión y pueden cargar imágenes individuales o carpetas completas. Los usuarios pueden organizar sus mapas en

colecciones o agruparlos con etiquetas. Cada colección e imagen estadística en NeuroVault obtiene un enlace permanente (URL) que se puede compartir con otros investigadores o incluir en artículos u otras formas de publicación (blogs, tweets, etc.). Los usuarios pueden especificar si cada colección es pública o privada. Estos últimos tienen una URL ofuscada única que no se puede descubrir en el sitio web de NeuroVault y, por lo tanto, solo puede acceder a ella con quien el propietario decida compartir la URL. La opción de crear colecciones privadas da a los usuarios la libertad de decidir quién puede acceder a sus datos y puede facilitar un escenario en el que una colección se comparte de forma privada durante el proceso de revisión por pares antes de la publicación y luego se hace pública al aceptar un manuscrito. El uso de un tercero (como NeuroVault) para compartir datos que forman parte del proceso de revisión por pares elimina las preocupaciones sobre el anonimato de los revisores. Brinda la posibilidad de vincular una colección a un artículo a través de un DOI para promover el artículo asociado y facilitar el metaanálisis.

Para preservar los datos, se realizan copias de seguridad diarias fuera del sitio que luego se copian en otras ubicaciones. NeuroVault es gratuito y no está sujeto a acuerdos de uso de datos. Los datos están disponibles y la base de datos se puede consultar a través de la interfaz web y la API RESTful. Esta plataforma simple y moderna abre la puerta al desarrollo de métodos novedosos para extraer inferencias de una base de datos meta analítica (Gorgolewski et al., 2015).

1.4.1.6 NeuroMorpho

NeuroMorpho (*NeuroMorpho*, 2022) proporciona una interfaz gráfica fácil de usar para acceder a los datos a través de cualquier navegador web moderno. Los visitantes pueden muestrear un conjunto aleatorio de neuronas o navegar por todo el repositorio por tipo de célula, región del cerebro, especie animal o laboratorio colaborador, correspondiente a los elementos intuitivos asociados más inmediatamente con cada estudio (qué, dónde, cuál, quién). En todos los casos, los datos se pueden seleccionar y descargar o simplemente visualizar dinámicamente en una secuencia rápida con simples movimientos del cursor.

NeuroMorpho no requiere ningún registro de usuario o inicio de sesión para buscar y descargar datos. Para cada neurona en la base de datos, tanto las representaciones gráficas como los archivos planos están disponibles a través de enlaces directos para visualización y descarga. Los archivos planos incluyen el archivo de reconstrucción original proporcionado por el laboratorio de origen, la versión convertida a un formato estandarizado, el registro que detalla todas las modificaciones y un documento que enumera las notas o irregularidades restantes. Los usuarios pueden optar por descargar uno o

todos estos cuatro archivos para cualquier número de neuronas como un único archivo comprimido. NeuroMorpho también es técnicamente interoperable ya que permite el acceso externo directo a los datos a través de consultas incrustadas en URL que aceptan pares de nombre-valor que especifican la fuente de datos, la consulta SQL y el formato de salida.

1.4.2 A nivel de Solución

1.4.2.1 *Boutiques*

Boutiques es una herramienta para publicar, integrar y ejecutar automáticamente aplicaciones en plataformas informáticas. Las aplicaciones de Boutiques se resumen en una descripción JSON simple pero rica, y permiten la simulación, validación, evaluación y monitoreo específico de la aplicación de herramientas de línea de comandos. El esquema define entradas, salidas, códigos de error, grupos de entrada, contenedores y más. Aprovecha los motores de contenedores de Linux como Docker y Singularity para implementar herramientas en todas las plataformas sin ninguna instalación. Los ecosistemas computacionales como CBRAIN o VIP hacen uso de los descriptores de Boutiques para importar y administrar la implementación de canalizaciones a escala. Muchos descriptores están disponibles a través del portal de intercambio de recursos de Zenodo. (Glatard et al., 2018)

1.4.2.2 Simple Linux Utility for Resource Management (SLURM)

Slurm (*Simple Linux Utility for Resources Management*), es un sistema de gestión de tareas y de clústeres (nodos o servidores de cómputo). Slurm como sistema de gestión tiene tres tareas claves:

- Asignar a los usuarios acceso exclusivo o no exclusivo a nodos de cómputo durante un tiempo determinado para que puedan ejecutar sus tareas.
- Proporciona un *framework* que permite iniciar, ejecutar y supervisar el trabajo.
- Se encarga de arbitrar la disputa de recursos, administrando una cola de tareas pendientes.

Tenemos que SLURM consiste en un demonio central (slurmctdl) que se ejecuta en un nodo de administración, y se encarga de monitorizar los recursos y el trabajo. Puede existir otro demonio exactamente igual, que será un demonio de respaldo para asumir esa responsabilidad ante posibles fallos. En cada nodo de cómputo se ejecuta un demonio (slurmd), cuyo cometido es: esperar trabajos, ejecutar esos trabajos, devolver el resultado de ejecución de esos trabajos y esperar más trabajos. Estos demonios proporcionan comunicaciones jerárquicas tolerantes de fallos. Existe otro demonio (slurmdbd), que es opcional y que puede usarse para registrar la información relativa a la contabilidad de varios clústeres que estén administrados por SLURM en una única base de datos. (Yoo et al., 2003)

1.5 Análisis comparativo

Primeramente, a **Nivel de Plataforma**:

Tabla I-1 Análisis comparativo de las plataformas.
Fuente: los autores.

	CBRAIN	**LORIS**	**BIL**	**C-Big**	**Neuro-Vault**	**Neuro-Morpho**
Acceso al repositorio	SSL, SSH	SSL	SSH, Bridges-2, Neocórtex, Sistema VM	SSL	SSL	SSH
Acceso a los datos	Inicio de sesión.	Inicio de sesión.	Inicio de sesión.	Inicio de sesión.	Inicio de sesión.	No 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)	EEG (EEG-BIDS), MEG(MEG-BIDS)	EEG, MEG(NIFTI)	Archivos planos, imágenes y metadatos textuales (swc)
Tecnologías / Lenguajes	Ruby y Rails	Linux, Apache, MySQL, PHP (LAMP)	No especificado.	LORIS, MySQL, React, JavaScript	Python, Django, Docker, JavaScript	MySQL, Java, C ++, Matlab, Apache Tomcat
Código abierto	Si	Si	No	Si	Sí	Si
Organización / Institución a la que pertenece	Canadian Open Neuroscience Platform	Canadian Open Neuroscience Platform	Iniciativa BRAIN, Universidad de Pittsburgh	Canadian Open Neuroscience Platform	Neuroscience Information Framework	Neuroscience Information Framework

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 en su mayoría no son de código abierto, pertenecen o están asociados a instituciones u organizaciones extranjeras por lo que no cumplen con las políticas de soberanía del país. 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.

Pero la realización de este proceso posee una alta complejidad técnica por lo que obliga a los investigadores a dominar técnicas de línea de comandos en servidores Linux para dictar instrucciones y ejecutar los procesamientos en el clúster.

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 errores 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.

1.6 Ambiente de desarrollo

1.6.1 Metodología

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.

La metodología SCRUM es un marco de trabajo diseñado para lograr la colaboración eficaz de equipos en proyectos, que emplea un conjunto de reglas y artefactos y define roles que generan la estructura

necesaria para su correcto funcionamiento. SCRUM utiliza un enfoque incremental. («Revisión de metodologías ágiles para el desarrollo de software», 2013)

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. (Manuel 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.

1.6.2 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 tuvieron en cuenta los siguientes:

- C#

  C# es un lenguaje de programación multiparadigma desarrollado y estandarizado por la empresa Microsoft como parte de su plataforma .NET, es uno de los lenguajes de programación diseñados para la infraestructura de lenguaje común. Es un entorno de desarrollo de software para sistemas operativos Windows. Se puede utilizar para crear sitios y aplicaciones web en ASP.NET, aplicaciones de escritorio y aplicaciones móviles. (Johel Jiménez Rivera, 2018)

- Python

> Python es un lenguaje multiparadigma, esto significa que combina propiedades de diferentes paradigmas de programación. Principalmente es un lenguaje orientado a objetos, todo en Python es un objeto, pero también incorpora aspectos de la programación imperativa, funcional, procedural y reflexiva. (Arturo Barrera, 2019)

Por ser poderoso y rápido; juega bien con los demás; corre por todos lados; es amigable y fácil de aprender se eligió para la aplicación el lenguaje de programación Python.

Lenguaje de Modelado

UML es el acrónimo de Lenguaje Unificado de Modelado, es el lenguaje estándar para visualizar, especificar, construir y documentar los artefactos de un sistema, utilizándose para el modelado del negocio y sistemas de software. También ofrece un estándar para describir los modelos, incluyendo aspectos conceptuales como procesos de negocio, funciones del sistema, expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables. (Molina et al., 2004)

1.6.3 Tecnologías

Marco de trabajo (Framework)

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 tuvieron en cuenta los siguientes:

- Flask

> Es un “micro” Framework escrito en Python y desarrollado para simplificar y hacer más fácil la creación de Aplicaciones Web bajo el patrón MVC. (*Flask*, 2022)
>
> La palabra “micro” no quiere decir que se trate de un proyecto pequeño o que nos sirva para hacer páginas web pequeñas, al instalar Flask disponemos de las herramientas necesarias para crear una aplicación web funcional. Es probable que en algún momento se necesite una nueva funcionalidad que no se tiene de primeras con la instalación, para eso encontrarás un gran

conjunto de extensiones (*plugins*) que se pueden instalar fácilmente con Flask y que te permitirán añadirle todas las funcionalidades que necesites.

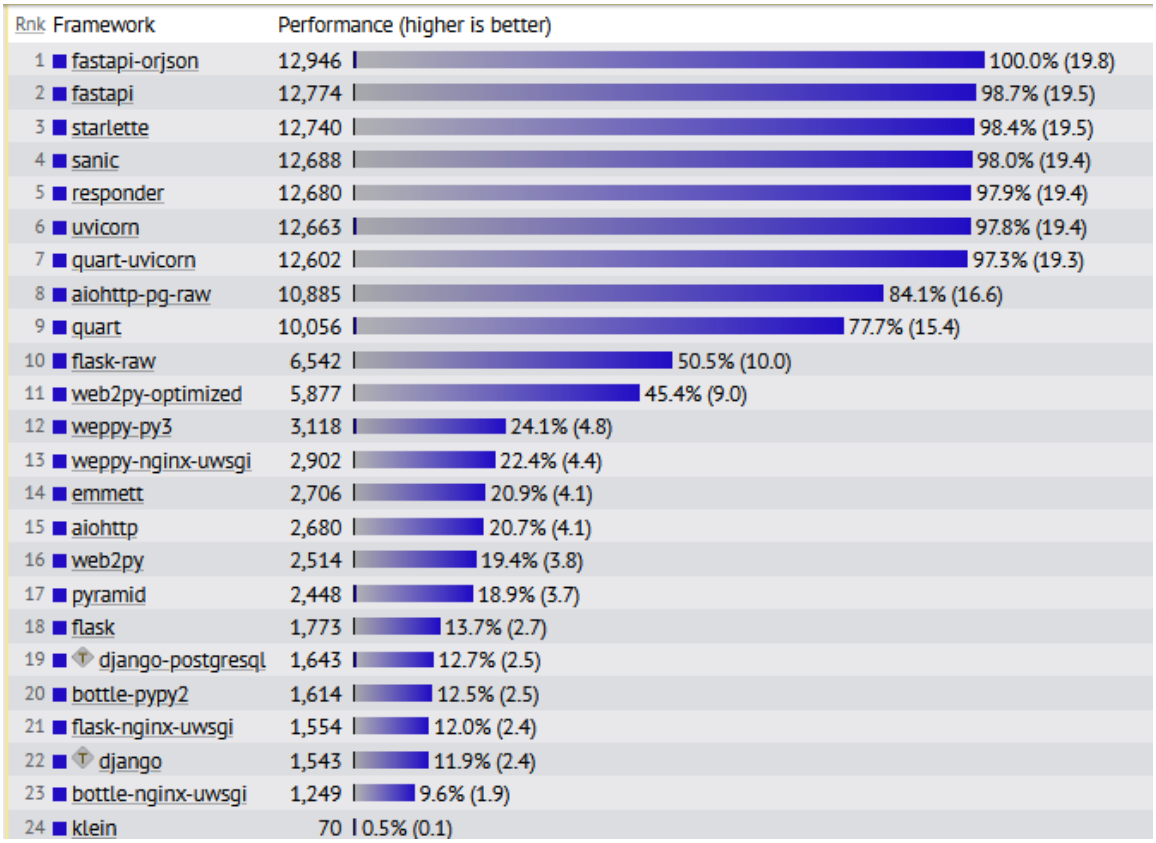
En cuanto al patrón MVC, este es una forma de trabajar que permite diferenciar y separar lo que es la vista (página HTML), el modelo de datos (los datos que va a tener la App), y el controlador (donde se gestionan las peticiones de la app web). (Epitech España, 2021)

- FastAPI

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. (Sebastián Ramírez, 2022)

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.



Rnk	Framework	Performance (higher is better)
1	fastapi-orjson	12,946 — 100.0% (19.8)
2	fastapi	12,774 — 98.7% (19.5)
3	starlette	12,740 — 98.4% (19.5)
4	sanic	12,688 — 98.0% (19.4)
5	responder	12,680 — 97.9% (19.4)
6	uvicorn	12,663 — 97.8% (19.4)
7	quart-uvicorn	12,602 — 97.3% (19.3)
8	aiohttp-pg-raw	10,885 — 84.1% (16.6)
9	quart	10,056 — 77.7% (15.4)
10	flask-raw	6,542 — 50.5% (10.0)
11	web2py-optimized	5,877 — 45.4% (9.0)
12	weppy-py3	3,118 — 24.1% (4.8)
13	weppy-nginx-uwsgi	2,902 — 22.4% (4.4)
14	emmett	2,706 — 20.9% (4.1)
15	aiohttp	2,680 — 20.7% (4.1)
16	web2py	2,514 — 19.4% (3.8)
17	pyramid	2,448 — 18.9% (3.7)
18	flask	1,773 — 13.7% (2.7)
19	django-postgresql	1,643 — 12.7% (2.5)
20	bottle-pypy2	1,614 — 12.5% (2.5)
21	flask-nginx-uwsgi	1,554 — 12.0% (2.4)
22	django	1,543 — 11.9% (2.4)
23	bottle-nginx-uwsgi	1,249 — 9.6% (1.9)
24	klein	70 — 0.5% (0.1)

Figura 1-7 Comparación entre frameworks de backend.
Fuente: (TechEmpower, 2021)

Ambos *frameworks* guardan cierta similitud en cuanto a sintaxis y se caracterizan por ser bastante ligeros y ofrecer la mínima funcionalidad. FastAPI ofrece validación, mientras que Flask no, FastAPI ofrece documentación automática, mientras que Flask no. Se eligió para la aplicación el *framework* FastAPI por todas las ventajas que ofrece y por disponer del mejor rendimiento según la Figura 1 -7.

Para la selección del marco de trabajo o librerías, en el cliente (*frontend*), se tuvieron en cuenta los siguientes:

- Vue.js: Es un *framework* progresivo lanzado en 2014 para construir interfaces de usuario. A diferencia de otros *frameworks* monolíticos, Vue está diseñado desde cero para ser utilizado incrementalmente. La librería central está enfocada solo en la capa de visualización, y es fácil de utilizar e integrar con otras librerías o proyectos existentes. Por otro lado, Vue también es perfectamente capaz de impulsar sofisticadas *Single-Page Applications* cuando se utiliza en combinación con herramientas modernas y librerías de apoyo.
- React.js: Es una biblioteca de JavaScript para construir interfaces de usuario lanzado en 2013 por Facebook. Es altamente dinámico y ofrece un gran soporte en la creación de interfaces de usuario

interactivas. No es específico del navegador y ligero. Es gratuito y elimina los problemas de rendimiento lento del DOM real ya que usa un DOM virtual. Esto permitió una mejora significativa en el rendimiento de los *frameworks* y librerías de JavaScript e hizo que React fuera completamente popular.

	React	Vue
Replace all rows	173.21 ms	168.47 ms
Create 10.000 rows	2085.16 ms	1615.71 ms
Startup time	72.4 ms	48 ms
Creating/clearing 1k rows (5 cycles)	4.93 MB	3.73 MB
Run memory	8.79 MB	6.98 MB

Figura 1-8 Comparación entre frameworks/librerías de frontend.
Fuente: (Vitaliy Ilyukha, 2022)

Ambos guardan cierta similitud en cuanto a rendimiento y flexibilidad, son adecuados para aplicaciones livianas y gozan de mucha popularidad en la comunidad de programadores. Se eligió para la aplicación el *framework* "Vue" por su simplicidad y eficiencia. De hecho, es más simple y más elegante que la mayoría de los marcos de trabajo en JavaScript que existen. Cuando codifica en Vue.js, el flujo se siente natural y todas las operaciones tienen sentido.

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.

Motor de Base 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)

1.6.4 Herramientas

Entorno integrado de desarrollo (IDE)

Permiten editar código fuente en diversos lenguajes de programación y ofrecen múltiples herramientas para facilitar el trabajo y aumentar la productividad. Los editores generalmente son programas ligeros ofreciendo al usuario productividad y experiencia de desarrollo, pero sin compilaciones. Sin embargo, los editores actuales se pueden extender, por medio de complementos que los pueden hacer llegar a ser tan avanzados como los IDE.

- Sublime Text

  Es un editor de texto y editor de código fuente, está escrito en C++ y Python para los plugin. Desarrollado originalmente como una extensión de VIM, con el tiempo fue creándose una identidad propia. Entre sus características principales se encuentra el mini mapa que consiste en un a previsualización de la estructura del código; la multiselección, hace una selección múltiple por varias secciones del archivo; multicursor, crea cursores con los que podemos escribir texto de forma arbitraria en diferentes posiciones del archivo; *multilayout*, trae 7 configuraciones de plantilla donde podemos elegir editar una sola ventana o hacer una división de hasta 4 ventanas verticales o en cuadriculas. (*Sublime Text*, 2022)

- Pycharm

  Es uno de los mejores IDE del mercado. Proporciona una finalización del código inteligente, inspecciones del código, indicación de errores sobre la marcha y arreglos rápidos, así como

refactorización de código automática y completas funcionalidades de navegación. La enorme colección de herramientas preconfiguradas de PyCharm incluye un depurador y un ejecutor de pruebas integrados, perfilador Python, un terminal integrado, integración con los principales VCS y herramientas de base de datos integradas, capacidades de desarrollo remoto con intérpretes remotos, un terminal SSH integrado e integración con Docker y Vagrant. Además de Python, PyCharm ofrece soporte de primer nivel para varios marcos de trabajo de desarrollo web Python, lenguajes de plantilla específicos, JavaScript, CoffeeScript, TypeScript, HTML/CSS, AngularJS, Node.js y más. (JetBrains, 2022)

Herramienta 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)

1.7 Conclusiones parciales del capítulo

Se constató que la investigación tiene actualidad y tiene aplicabilidad en el contexto cubano sobre todo en las neurociencias a partir del análisis del objecto de estudio en los referentes bibliográficos.

Se identificaron herramientas y tecnologías que son parcialmente solución, pero no todas se ajustan el contexto de CNEURO. No se identificó en la literatura una solución genérica que cumpla con los requisitos por las limitantes que existen hoy en día y que son necesarios para el procesamiento de las tareas en el clúster de CNEURO, siendo necesario desarrollar una herramienta que cumpla con estos requisitos y se adapte a las necesidades del centro.

Se obtuvo información valiosa sobre las características necesarias para el desarrollo de la herramienta que corresponda con las necesidades del centro, a través del análisis del estado del arte. Se seleccionó como metodología de desarrollo del software SCRUM, y las siguientes tecnologías y herramientas para el desarrollo de la propuesta de solución: Visual Paradigm, Python, FastAPI, Javascript, Vue.js, WebSocket, UML.

CAPÍTULO 2. Análisis y diseño de la herramienta para la gestión de procesamiento de datos de neurociencias.

En el actual capítulo se describe el procedimiento y los elementos que se tuvieron en cuenta para el desarrollo de la herramienta de procesamiento. Se presentan los conceptos asociados al modelo conceptual y la representación formal del mismo. Se identifican los requisitos no funcionales y funcionales. La implementación de la solución propuesta se caracteriza por basarse 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.

2.1 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. Las personas que interviene en el negocio son las siguientes:

Tabla II-2 Personas que intervienen en el negocio.
Fuente: los autores.

Nombre	**Descripción**
Desarrollador	Encargado de gestionar los servidores, scripts, usuarios, grupos y permisos.
Especialista	Encargado de ejecutar tareas de neurociencia en el clúster mediante un script.

2.2 Propuesta de solución

La propuesta ha sido diseñada para la gestión del procesamiento de datos de neurociencia a partir del estudio del funcionamiento de un Clúster. Para el uso del mismo, es necesario acceder mediante SSH al nodo maestro y ejecutar el comando o script correspondiente a la tarea a realizar. El clúster cuenta con un sistema de colas encargado de repartir las diferentes tareas por los nodos existentes. De este

modo, el usuario, a partir de instrucciones o comandos, sólo tendrá que lanzar su aplicación y el gestor de cola se encargará de encontrar nodos libres en los que ejecutar los cálculos. En caso de que estén todos ocupados, el gestor de colas se encarga de poner las tareas en espera y de lanzarla cuando haya recursos disponibles, sin necesidad de que el usuario esté conectado esperando.

La solución propuesta en esta investigación permitirá a los especialistas lanzar tareas al clúster mediante el uso de scripts publicados previamente por los desarrolladores. Un script es una descripción detallada de la estructura de un comando que permite a los especialistas, de forma simplificada, interactuar con el clúster. Para lanzar una tarea los especialistas, a partir de un script y un servidor seleccionados, deben rellenar los datos requeridos y lanzar la tarea. El servidor transforma estos datos en un comando y un *Worker* ejecuta la tarea usando SSH al clúster. Esta ejecución devolverá el registro de los resultados que visualizará el Especialista en tiempo real. La Figura 2 -9 modela el funcionamiento del proceso de la propuesta de solución.

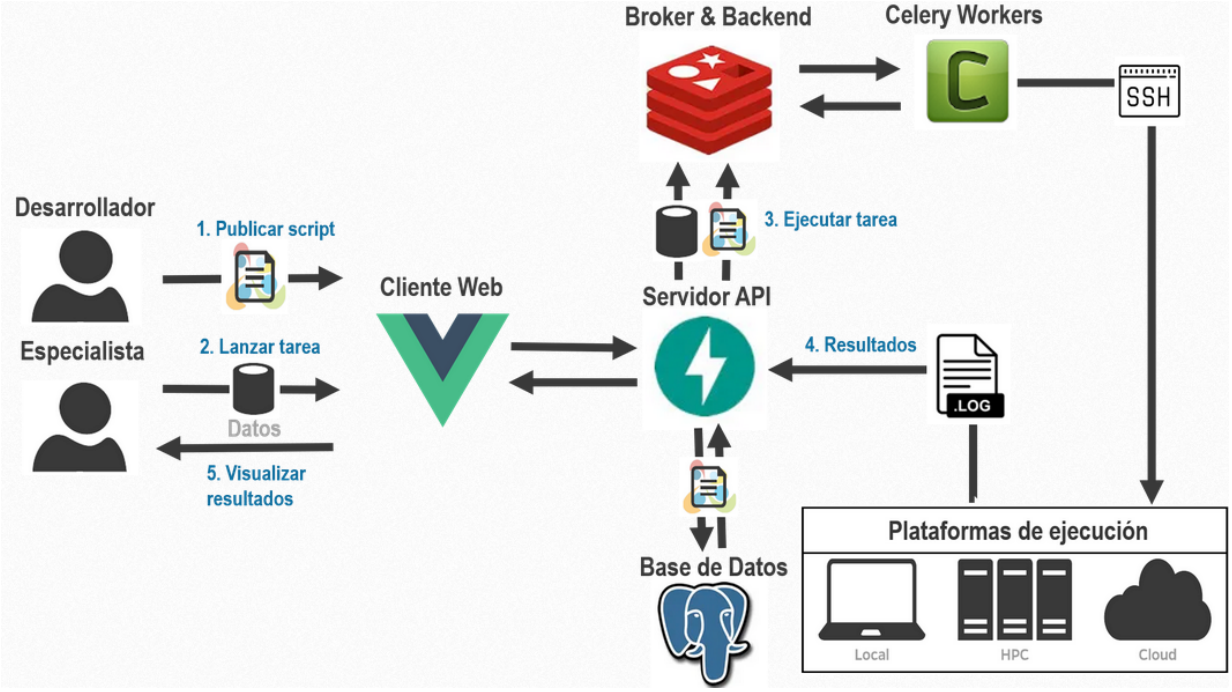


Figura 2-9 Esquema general del funcionamiento de la solución propuesta.
Fuente: los autores.

2.3 Requisitos de software

La ingeniería de requisitos es el proceso de desarrollar una especificación de software. Las especificaciones pretenden comunicar las necesidades del sistema del cliente a los desarrolladores del sistema. (Sommerville, 2011)

2.3.1 Requisitos funcionales

Un requisito funcional es una capacidad o condición que el sistema debe cumplir. (Pressman, 2010) 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.

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

RF 1. Autenticar usuario.
RF 2. Registrar usuario.
RF 3. Modificar usuario.
RF 4. Visualizar detalles del usuario.
RF 5. Eliminar usuario.
RF 6. Listar usuarios.
RF 7. Registrar servidor.
RF 8. Modificar servidor.
RF 9. Visualizar detalles del servidor.
RF 10. Eliminar servidor.
RF 11. Listar servidores.
RF 12. Registrar script.
RF 13. Modificar script.
RF 14. Eliminar script.
RF 15. Listar scripts.
RF 16. Visualizar detalles del script.
RF 17. Concatenar programas del script.
RF 18. Registrar programa.
RF 19. Modificar programa.
RF 20. Visualizar detalles del programa.
RF 21. Eliminar programa.
RF 22. Listar programas.
RF 23. Ejecutar tarea.
RF 24. Obtener tarea.
RF 25. Detener tarea.
RF 26. Comprobar tarea.
RF 27. Obtener registro de la tarea.
RF 28. Obtener registro de errores de la tarea.

2.3.2 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)

- **Interfaz:**

RNF1. La herramienta debe incluir el nombre y logo de BrainSSys.

- **Seguridad:**

RNF2. La información que se maneje se mantendrá segura y confidencial, evitando el acceso no autorizado y la divulgación.

RNF3. 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.

RNF4. Disponer de un mecanismo de autenticación de usuarios seguro.

- **Eficiencia:**

RNF5. El tiempo de respuesta de la herramienta no debe exceder de 15 segundos.

- **Usabilidad:**

RNF6. Debe tener una interfaz amigable e intuitiva, que pueda ser usada por cualquier usuario sin conocimientos especiales.

RNF7. 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.

2.3.3 Historias de Usuario

Las historias de usuario (HU) constituyen una forma de administración de requisitos sin tener que elaborar gran cantidad de documentos formales y sin requerir de mucho tiempo para administrarlos. Las mismas son escritas utilizando el lenguaje común del usuario. (Jeffries et al., 2000)

Para la descripción de requisitos funcionales de la propuesta de solución se define una historia de usuario para cada uno de ellos, para un total de 28 historias de usuario. En las tablas Tabla II-3 y Tabla II-4 se muestra la HU correspondiente al requisito funcional **Registrar servidor** y **Ejecutar tarea**. En el **Anexo 2: Historias de Usuario** pueden encontrarse las HU restantes implementadas en el desarrollo de la herramienta.

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

- **Número:** Número asignado a la HU.
- **Nombre:** Nombre de la HU.
- **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.

Tabla II-3 Historia de usuario: Registrar servidor.
Fuente: los autores.

HU07 – Registrar servidor	
Yo como:	Desarrollador
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	

Tabla II-4 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	

2.4 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. (Manuel Trigas Gallego, 2012)

2.4.1 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. (Manuel Trigas Gallego, 2012)

La Pila de Producto contiene los siguientes campos (Henrik Kniberg, 2007):

- **ID**: Identificador único, simplemente un número auto-incremental.
- **Nombre**: Descripción breve del requisito, normalmente de 2 a 10 palabras.
- **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).
- **Descripción**: Especificación de cómo funciona el requerimiento.

Tabla II-5 Product Backlog.
Fuente: los autores.

ID	Nombre	Importancia	Estimación Inicial	Descripción
RF1	Autenticar usuario	70	2	El usuario debe autenticarse para comprobar permisos de acceso.
RF2	Registrar usuario	70	2	Insertar usuarios en la herramienta.
RF3	Modificar usuario	70	2	Actualizar la información referente a un usuario especificado.
RF4	Visualizar detalles del usuario	70	2	Mostrar información referente a un usuario especificado.
RF5	Eliminar usuario	70	2	Remover un usuario especificado de la herramienta.
RF6	Listar usuarios	70	2	Visualizar todos los usuarios de la herramienta.
RF7	Registrar servidor	100	2	Insertar un servidor en la herramienta.
RF8	Modificar servidor	90	2	Modificar un servidor especificado.
RF9	Visualizar detalles	90	2	Mostrar detalles de un servidor

	del servidor			especificado.
RF10	Eliminar servidor	80	2	Remover un servidor de la herramienta.
RF11	Listar servidores	90	2	Visualizar todos los servidores de la herramienta.
RF12	Registrar script	100	6	Insertar un script en la herramienta.
RF13	Modificar script	90	6	Actualizar la información referente un script en la herramienta.
RF14	Eliminar script	90	2	Remover un script de la herramienta.
RF15	Listar scripts	90	2	Visualizar todos los scripts insertados previamente.
RF16	Visualizar detalles del script	90	2	Mostrar los detalles de un script insertado previamente.
RF17	Concatenar programas del script	100	8	Insertar más programas a un script.
RF18	Registrar programa	100	8	Insertar un programa en la herramienta.
RF19	Modificar programa	90	2	Actualizar la información referente a un programa.
RF20	Visualizar detalles	90	2	Mostrar los detalles de un programa especificado.
RF21	Eliminar programa	80	2	Remover de la herramienta un programa especificado.
RF22	Listar programas	100	2	Visualizar todos los programas insertados previamente.
RF23	Ejecutar tarea	100	6	Ejecutar una tarea previamente insertada.
RF24	Obtener tarea	90	2	Visualizar una tarea.
RF25	Detener tarea	100	4	Detener la ejecución de una tarea.
RF26	Comprobar tarea	90	4	Verificar el estado de una tarea especificada.
RF27	Obtener registro de la tarea	100	4	Exportar fichero con la información resultante de la tarea.
RF28	Obtener registro de errores de la tarea	80	4	Exportar fichero con los errores resultantes de una tarea.

2.4.2 Sprint Backlog

Es la lista de tareas que elabora el equipo durante la planificación de un sprint. Se asignan las tareas a cada persona y el tiempo que queda para terminarlas. De esta forma el proyecto queda descompuesto

en unidades más pequeñas pudiendo determinarse en qué tareas se ha avanzado. Todas las tareas deben tener un coste semejante entre 4 y 16 horas. (Manuel Trigas Gallego, 2012)

Para la planificación de las tareas se definieron un total de 3 Sprint que permiten el desarrollo completo de la herramienta. En la Tabla II-6, se muestra el **Sprint Backlog #0** correspondiente y en el **Anexo 3: Sprint Backlogs** se encuentran los *Sprints* restantes implementados para dar solución al problema de investigación.

Tabla II-6 Sprint backlog #0.
Fuente: los autores.

Sprint 0				
ID	Tarea	Asignada a	Estado	Tiempo
UH12-T1	Diseño e implementación de clases POCO.	Michel Suárez	Terminada	2
UH13-T2	Migración a la base de datos.	Michel Suárez	Terminada	2
UH12-T3	Implementación de los repositorios para el acceso a datos.	Michel Suárez	Terminada	4
UH12-T4	Implementación de la lógica de la funcionalidad.	Michel Suárez	Terminada	4
UH12-T5	Creación de los *endpoints*.	Michel Suárez	Terminada	2
UH12-T6	Diseño e implementación de los DTO. (*Data Transfer Object*)	Michel Suárez	Terminada	2
UH12-T7	Diseño del prototipado de la interfaz.	Brian Pérez	Terminada	4
UH12-T8	Implementación de la funcionalidad en el cliente web.	Brian Pérez	Terminada	6
UH7-T1	Diseño e implementación de clases POCO.	Michel Suárez	Terminada	2
UH7-T2	Migración a la base de datos.	Michel Suárez	Terminada	2
UH7-T3	Implementación de los repositorios para el acceso a datos.	Michel Suárez	Terminada	4
UH7-T4	Implementación de la lógica de la funcionalidad.	Michel Suárez	Terminada	4
UH7-T5	Creación de los *endpoints*.	Michel Suárez	Terminada	2
UH7-T6	Diseño e implementación de los DTO. (*Data Transfer Object*)	Michel Suárez	Terminada	2
UH7-T7	Diseño del prototipado de la interfaz.	Brian Pérez	Terminada	4
UH7-T8	Implementación de la	Brian Pérez	Terminada	6

	funcionalidad en el cliente web.			
UH13-T1	Implementación de los repositorios para el acceso a datos.	Michel Suárez	Terminada	4
UH13-T2	Implementación de la lógica de la funcionalidad.	Michel Suárez	Terminada	4
UH13-T3	Creación de los *endpoints*.	Michel Suárez	Terminada	2
UH13-T4	Diseño e implementación de los DTO. (*Data Transfer Object*)	Michel Suárez	Terminada	2
UH13-T5	Diseño del prototipado de la interfaz.	Brian Pérez	Terminada	4
UH13-T6	Implementación de la funcionalidad en el cliente web.	Brian Pérez	Terminada	6
UH14-T1	Implementación de los repositorios para el acceso a datos.	Michel Suárez	Terminada	4
UH14-T2	Implementación de la lógica de la funcionalidad.	Michel Suárez	Terminada	4
UH14-T3	Creación de los *endpoints*.	Michel Suárez	Terminada	2
UH14-T4	Diseño e implementación de los DTO. (*Data Transfer Object*)	Michel Suárez	Terminada	2
UH14-T5	Diseño del prototipado de la interfaz.	Brian Pérez	Terminada	4
UH14-T6	Implementación de la funcionalidad en el cliente web.	Brian Pérez	Terminada	6
UH15-T1	Implementación de los repositorios para el acceso a datos.	Michel Suárez	Terminada	4
UH15-T2	Implementación de la lógica de la funcionalidad.	Michel Suárez	Terminada	8
UH15-T3	Creación de los *endpoints*.	Michel Suárez	Terminada	2
UH15-T4	Diseño e implementación de los DTO. (*Data Transfer Object*)	Michel Suárez	Terminada	2
UH15-T5	Diseño del prototipado de la interfaz.	Brian Pérez	Terminada	4
UH15-T6	Implementación de la funcionalidad en el cliente web.	Brian Pérez	Terminada	6
UH16-T1	Implementación de los repositorios para el acceso a datos.	Michel Suárez	Terminada	4
UH16-T2	Implementación de la	Michel Suárez	Terminada	8

	lógica de la funcionalidad.			
UH16-T3	Creación de los *endpoints*.	Michel Suárez	Terminada	2
UH16-T4	Diseño e implementación de los DTO. (*Data Transfer Object*)	Michel Suárez	Terminada	2
UH16-T5	Diseño del prototipado de la interfaz.	Brian Pérez	Terminada	4
UH16-T6	Implementación de la funcionalidad en el cliente web.	Brian Pérez	Terminada	6
UH11-T1	Implementación de los repositorios para el acceso a datos.	Michel Suárez	Terminada	4
UH11-T2	Implementación de la lógica de la funcionalidad.	Michel Suárez	Terminada	4
UH11-T3	Creación de los *endpoints*.	Michel Suárez	Terminada	2
UH11-T4	Diseño e implementación de los DTO. (*Data Transfer Object*)	Michel Suárez	Terminada	2
UH11-T5	Diseño del prototipado de la interfaz.	Brian Pérez	Terminada	4
UH11-T6	Implementación de la funcionalidad en el cliente web.	Brian Pérez	Terminada	6

2.5 Arquitectura de la solución

Según (Pressman, 2010), en su forma más simple, la arquitectura del software es la estructura u organización 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.

Para el desarrollo de la herramienta 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.

2.5.1 Arquitectura por capas

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. **(Pascual, 2019)**

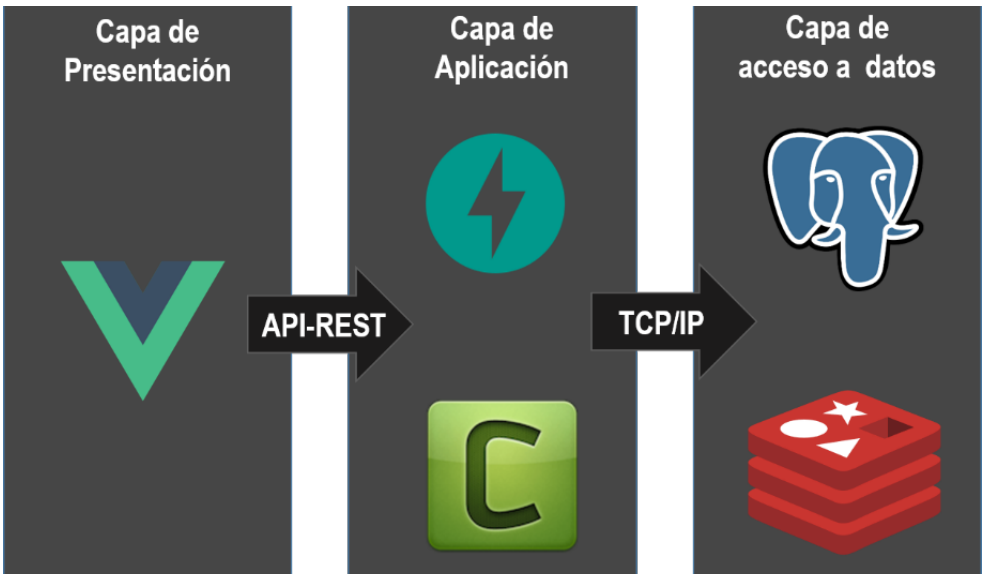


Figura 2-10 Arquitectura de la solución propuesta.
Fuente: los autores.

En la Figura 2 -10, se aprecian las siguientes **Capas o Niveles**:

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.

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.

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.

2.5.2 Patrón arquitectónico

Un patrón arquitectónico brinda la descripción de un problema particular y recurrente de diseño, que aparece en contextos de diseño específico, y presenta un esquema genérico demostrado con éxito para su solución. El esquema de solución se especifica mediante la descripción de los componentes que lo constituyen, sus responsabilidades y desarrollos, así como también la forma como estos colaboran entre sí. El patrón arquitectónico es quien define la estructura básica de la aplicación. (Camacho, E. et al., 2004)

2.5.2.1 Patrón arquitectónico Modelo – Vista – Controlador (MVC)

La herramienta está planteada con una arquitectura basada en el Modelo Vista Controlador (MVC) como se aprecia en la Figura 2 -11. 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. (Frank 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”).

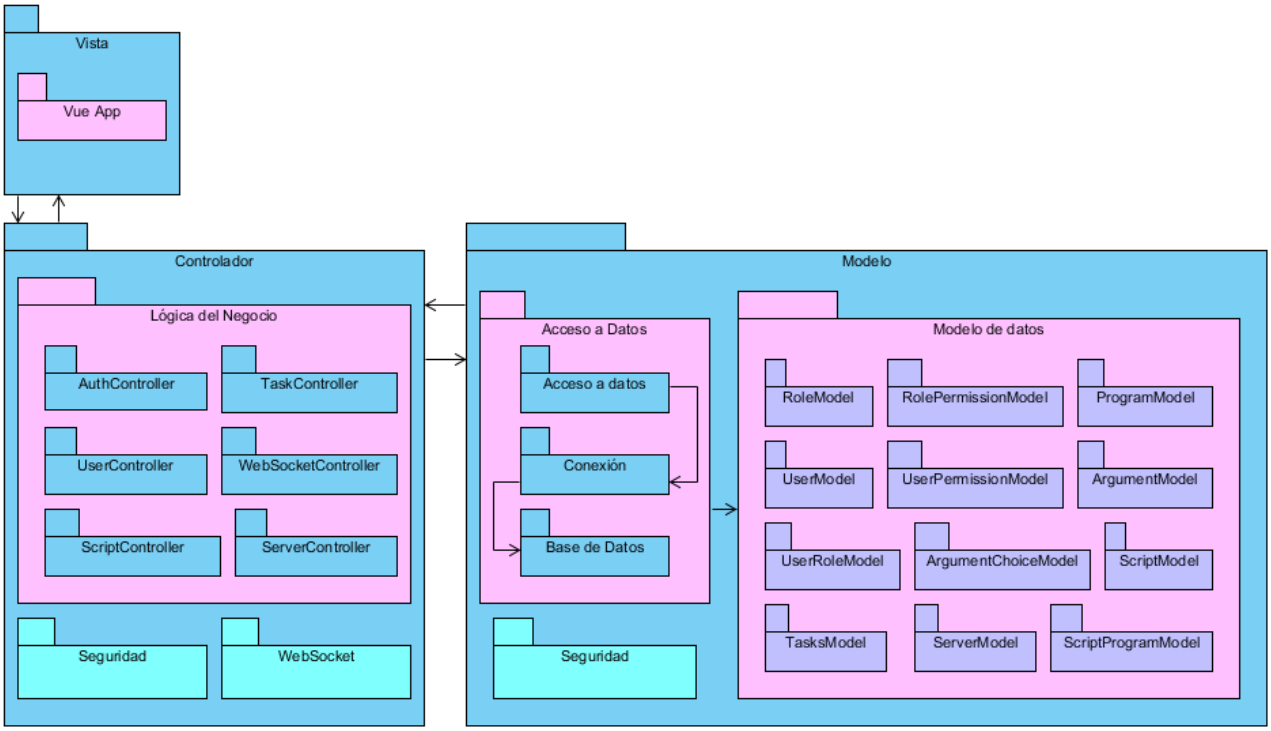


Figura 2-11 Modelo de la arquitectura de la herramienta (*backend*).
Fuente: los autores.

2.5.2.2 Patrón arquitectónico Modelo – Vista – Modelo de vista (MVVM)

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. (akin-ogundeji muyiwa, 2015)

En la Figura 2 -12 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 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" que manejan eventos DOM interceptados por directivas incrustadas dentro de la plantilla de Vista (paquete “Vista-Modelo”).

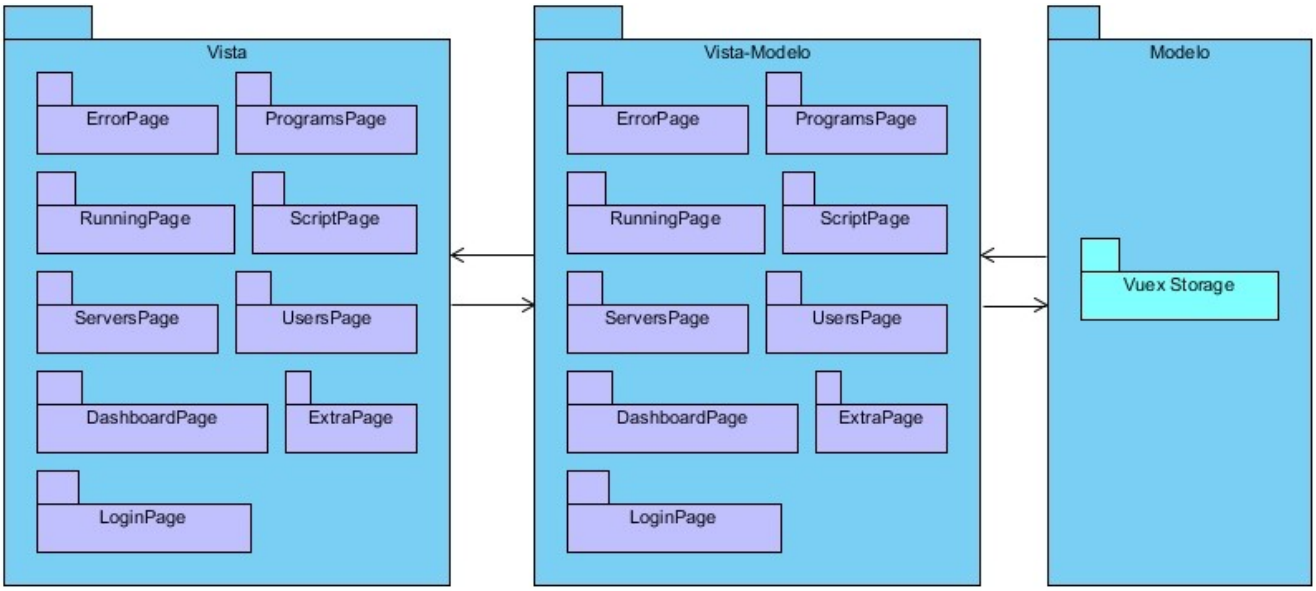


Figura 2-12 Modelo de la arquitectura de la herramienta (*frontend*).
Fuente: los autores.

2.5.3 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.

2.5.3.1 Patrón GRASP

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. (Juan García 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. (Juan García Carmona, 2012)

Bajo acoplamiento: Es la idea de tener las clases lo menos ligadas entre sí que se pueda, de tal forma que, en caso de producirse una modificación en alguna de ellas, tenga la mínima repercusión posible en el resto de clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases. También hay varios tipos de acoplamiento. (Juan García Carmona, 2012)

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. (Juan García Carmona, 2012)

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. (Juan García Carmona, 2012)

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. (Juan García Carmona, 2012)

Fabricación pura: La fabricación pura se da en las clases que no representan un ente u objeto real del dominio del problema, sino que se han creado intencionadamente para disminuir el acoplamiento, aumentar la cohesión y/o potenciar la reutilización del código. Esta surge cuando el diseñador se encuentra con una clase poco cohesiva y que no tiene otra clase en la que implementar algunos métodos. Es decir que se crea una clase "inventada" o que no existe en el problema como tal, pero que, añadiéndola, logra mejorar estructuralmente el sistema. Como contraindicación deberemos mencionar que al abusar de este patrón suelen aparecer clases función o algoritmo, esto es, clases que tienen un solo método. (Juan García Carmona, 2012)

Indirección: El patrón de indirección nos permite mejorar el bajo acoplamiento entre dos clases asignando la responsabilidad de la mediación entre ellos a una clase intermedia. (Juan García Carmona, 2012)

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. (Juan García Carmona, 2012)

2.5.4 Diseño de Interfaz de Usuario

El diseño de la interfaz de usuario crea un medio eficaz de comunicación entre los seres humanos y la computadora. Siguiendo un conjunto de principios de diseño de la interfaz, el diseño identifica los objetos y acciones de ésta y luego crea una plantilla de pantalla que constituye la base del prototipo de la interfaz de usuario. (Pressman, 2010)

La Figura 2 -13 muestra un prototipo de interfaz de usuario (UI), basada en la plantilla CoreUI, que debe tener la herramienta a implementar.

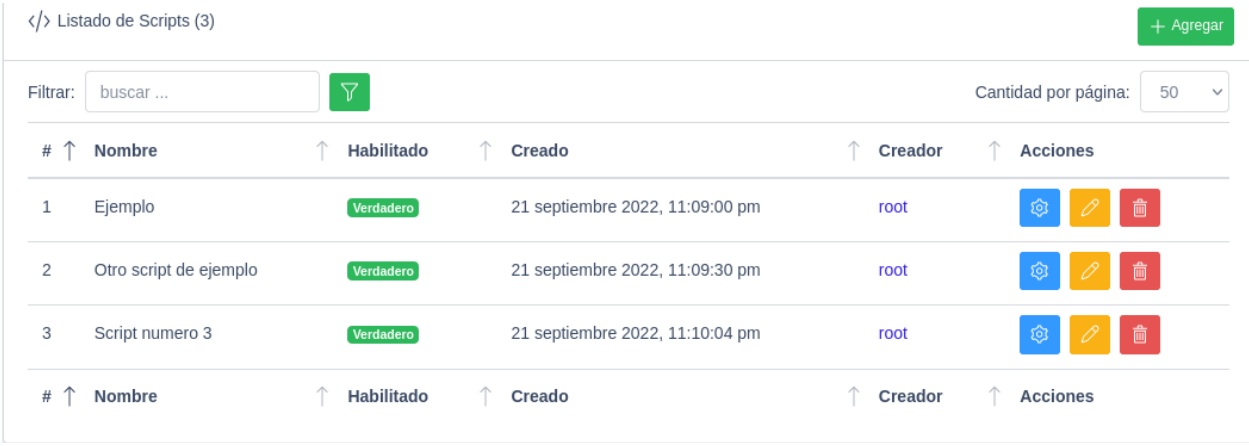


Figura 2-13 Prototipo de Interfaz de Usuario (RF15. Listar Scripts)
Fuente: los autores.

2.5.5 Mapa de Navegabilidad

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)

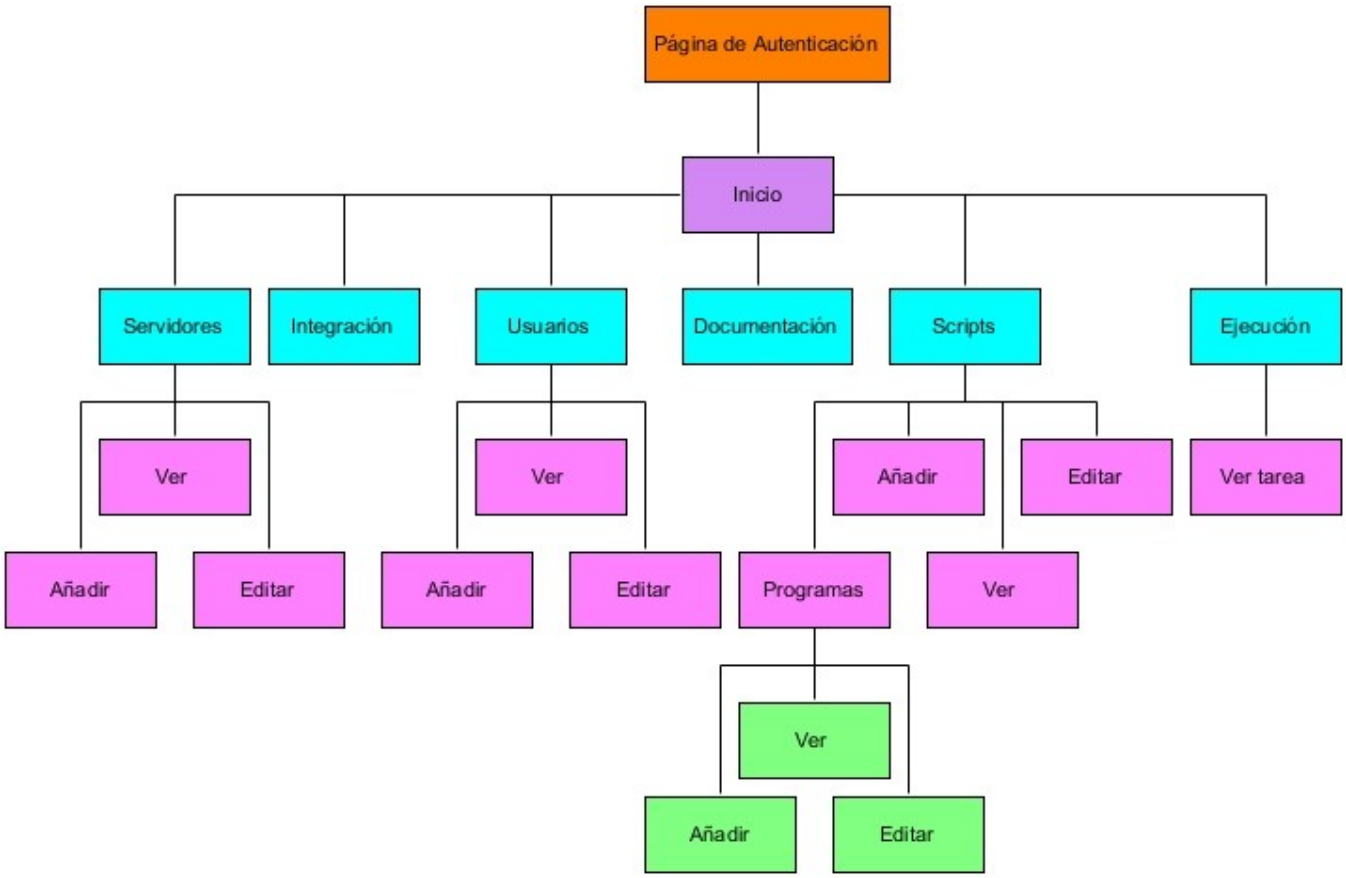


Figura 2-14 Mapa de Navegabilidad.
Fuente: los autores.

2.6 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. (Pressman, 2010)

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 (Pressman, 2010). Se generó el siguiente diagrama entidad-relación, el cual posee un conjunto de tablas correspondientes a cada componente, como se puede observar en la Figura 2 -15Figura 2 -15 Modelo de datos relacional para la herramienta.

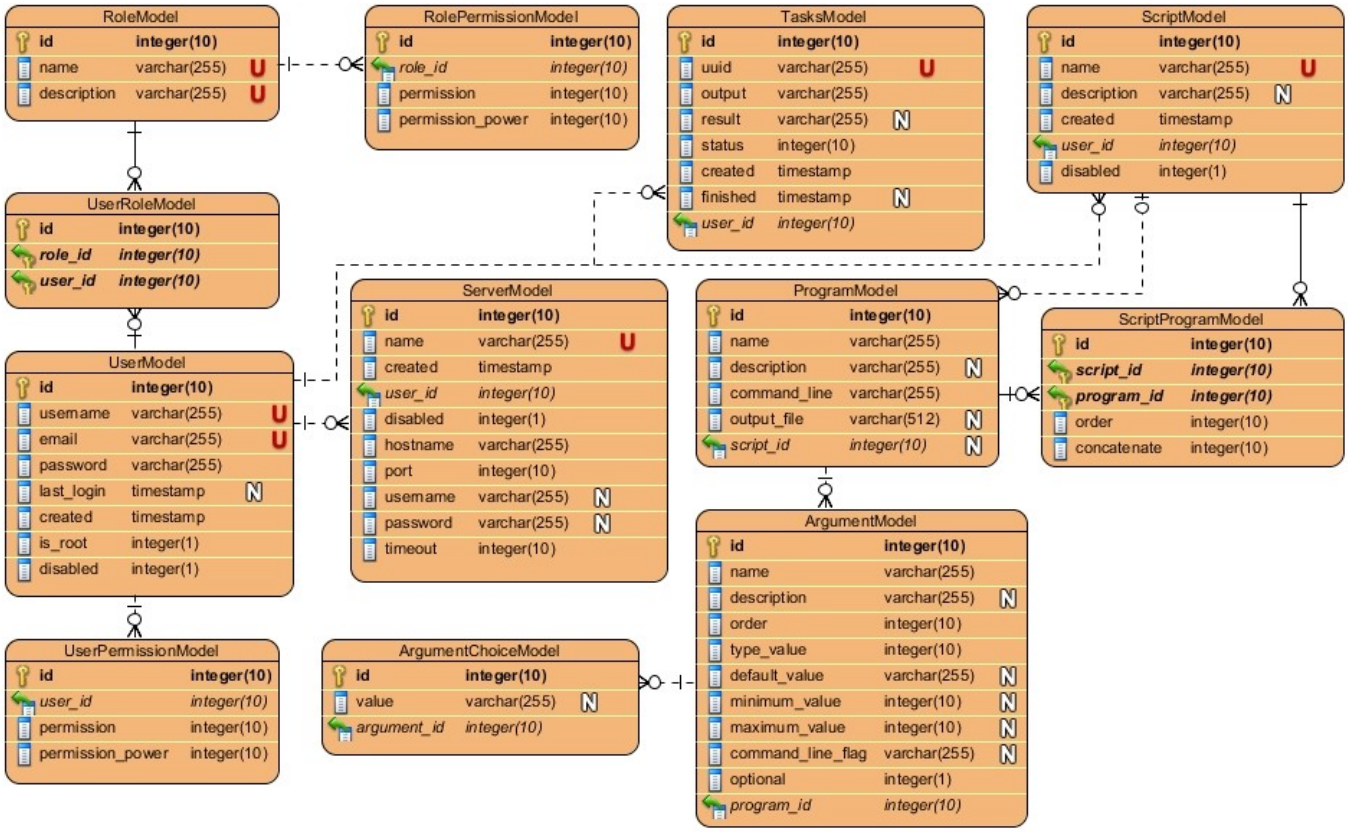


Figura 2-15 Modelo de datos relacional para la herramienta.
Fuente: los autores.

El diagrama representa el modelo de datos de la herramienta, donde 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.

La Tabla II-7 describe los atributos que contiene la entidad de la UserModel:

Tabla II-7 Atributos de la entidad UserModel
Fuente: los autores.

Atributo	Tipo	Descripción
id	integer	Id necesario en cada entidad para las referencias en las relaciones entre tablas. Es la llave principal en todas las entidades.
username	varchar	Nombre del usuario.
email	varchar	Correo electrónico del usuario.
password	varchar	Contraseña del usuario.

last_login	datetime	Ultima fecha y hora del acceso a la herramienta.
created	datetime	Fecha de creación del usuario.
is_root	integer	Permite conocer si el usuario dispone de todos los permisos de la herramienta.
disabled	integer	Permite conocer si el usuario esta deshabilitado en la herramienta.

En el expediente del proyecto BrainSSys se especifica el resto de las tablas asociadas al modelo de datos.

2.7 Conclusiones parciales

Con el análisis y diseño de la propuesta de solución se elaboró un esquema general del funcionamiento de las principales relaciones entre los conceptos asociados a la problemática. Al realizarse una correcta captura de requisitos, se obtuvieron 28 requisitos funcionales y 7 requisitos no funcionales que permiten la comprensión del sistema.

La arquitectura N capas seleccionada permitió separar los componentes en unidades independientes que facilitan el mantenimiento, desarrollo y la realización de pruebas.

Los patrones arquitectónicos MVC y MVVM, y las buenas prácticas aplicadas, con el uso de patrones GRASP, permitieron obtener una herramienta con calidad en su codificación. Con el diseño de un modelo de datos se representaron las entidades relevantes del sistema y sus relaciones.

CAPÍTULO 3. Validación e implementación de la propuesta de solución

En el presente capítulo se presentan las especificaciones asociadas a la implementación de la propuesta de solución. Se describen las pautas de codificación utilizadas, además la herramienta es sometida a un proceso de pruebas de software con el objetivo de verificar el cumplimiento de los requerimientos especificados anteriormente.

3.1 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 3 -16 muestra el diagrama de despliegue para la propuesta de solución.

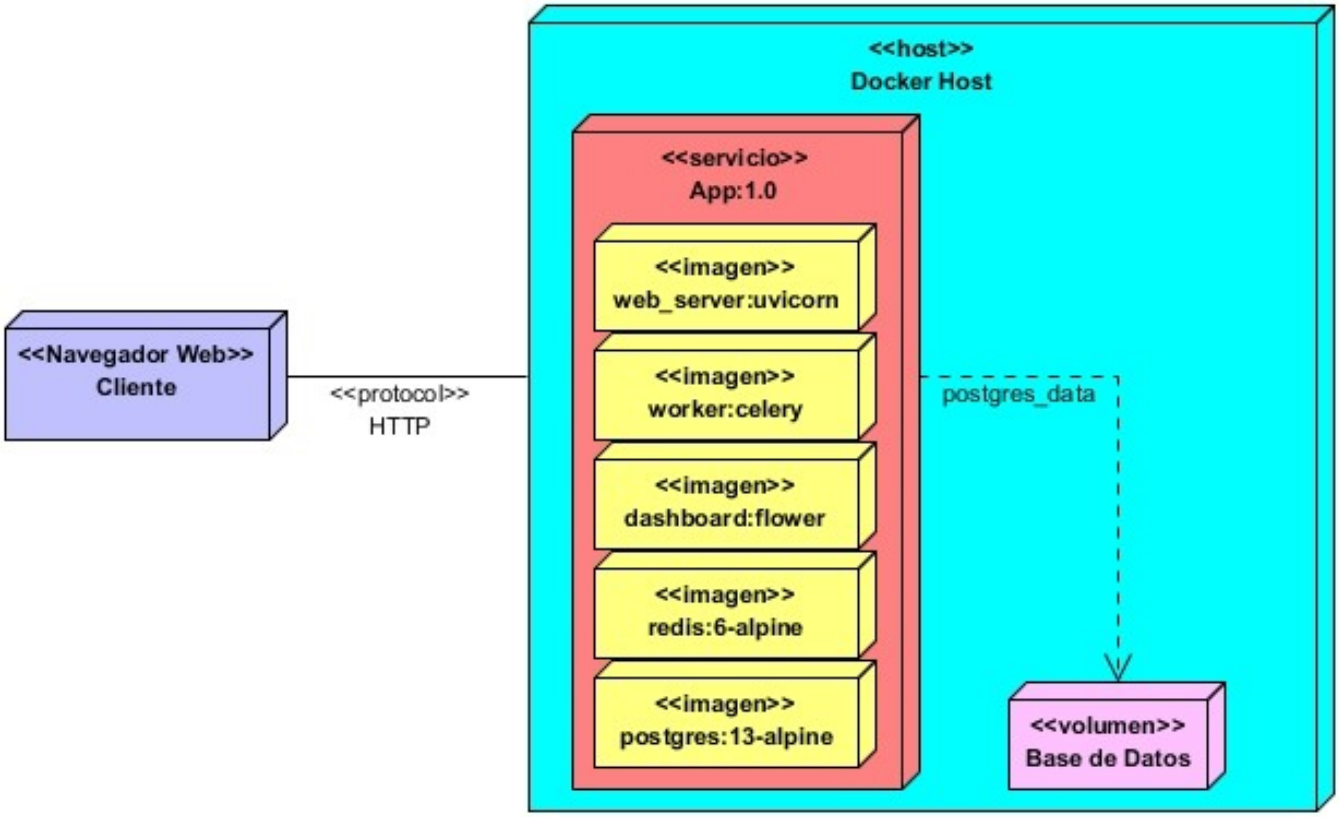


Figura 3-16 Diagrama de despliegue.
Fuente: los autores.

3.2 Estándares de codificación

Los estándares de codificación son un conjunto de convenciones (denominaciones, formatos) establecidas para la escritura de código. Estos estándares varían en dependencia del lenguaje de programación y son necesarios para los programadores y miembros del equipo de trabajo. Generalmente el software no es mantenido por los autores, por lo que debe ser revisado y/o manejado por otros. Los estándares posibilitan una mejor lectura e interpretación del software y su empleo permite aplicar un conjunto de lineamientos (Ampuero & Trujillo, 2006).

Sangría:

- Utilice 4 espacios para la sangría.

Grosor de línea:

- Cada línea de código no debe exceder los 80 caracteres en la medida de lo posible (en circunstancias especiales, puede exceder ligeramente los 80, pero la más larga no puede exceder los 120).

Razón:

- Esto es útil al ver una diferencia de lado a lado.
- Conveniente para ver el código debajo de la consola.
- Demasiado tiempo puede ser un diseño defectuoso.

Comillas:

- En pocas palabras, el lenguaje natural usa comillas dobles y las etiquetas de máquina usan comillas simples, por lo que la mayor parte del código debe usar comillas simples.
- El lenguaje natural usa comillas dobles"...". Por ejemplo, mensajes de error; en muchos casos sigue siendo Unicode, utilice "Hola mundo".
- ID de máquina Utilice comillas simples '...'. Por ejemplo, la clave en el “dict”
- Las expresiones regulares usan comillas dobles nativas "...".
- La cadena de documentos usa tres comillas dobles"""....."""".

La Figura 3 -17 muestra un fragmento del código de la herramienta donde se evidencia la aplicación de los estándares de codificación antes descritos.

```python
class AsyncSSHController(SSHController):
    """Controller connection and execution via SSH for AsyncSSH library"""
    client: SSHClientConnection = None
    task: AbortableTask = None

    async def connect(self) -> SSHClientConnection:
        """Connect to SSH Server"""
        pkey = self.config.get('pkey', None)
        if pkey is not None:
            self.client = await connect(host=self.config.get('host'), port=self.config.get('port'),
                                        client_keys=[self.config.get('pkey')],
                                        known_hosts=None)
        else:
            self.client = await connect(host=self.config.get('host'), port=self.config.get('port'),
                                        username=self.config.get('username', None),
                                        password=self.config.get('password', None),
                                        known_hosts=None)
        return self.client

    def set_task(self, task: AbortableTask) -> None:
        """Set task for check is_abort"""
        self.task = task
```

Figura 3-17 Estándares de codificación en el Controlador SSH
Fuente: los autores.

3.2.1 Principios SOLID

En ingeniería de software, SOLID (*Single responsibility, Open-closed, Liskov substitution, Interface segregation and Dependency inversion*) es un acrónimo mnemónico introducido por Robert C. Martin a comienzos de la década del 2000 que representa cinco principios básicos de la programación orientada a objetos y el diseño. Cuando estos principios se aplican en conjunto es más probable que un desarrollador cree un sistema que sea fácil de mantener y ampliar en el tiempo. Los principios SOLID son guías que pueden ser aplicadas en el desarrollo de software para eliminar código sucio provocando que el programador tenga que refactorizar el código fuente hasta que sea legible y extensible. Debe ser utilizado con el desarrollo guiado por pruebas o TDD, y forma parte de la estrategia global del desarrollo ágil de software y programación adaptativa. (Juan García Carmona, 2012)

Única responsabilidad (*Single responsibility***)**: Una clase debería concentrarse sólo en hacer una cosa de tal forma que cuando cambie algún requisito en mayor o menor medida dicho cambio sólo afecte a dicha clase por una razón. (Juan García Carmona, 2012)

Abierto/Cerrado (*Open-closed*): Las entidades de software (clases, módulos, funciones, etcétera) deberían estar abiertas a la extensión, pero cerradas a la modificación. (Juan García Carmona, 2012)

Sustitución de Liskov (*Liskov substitution*): Las funciones que utilicen punteros o referencias a clases base deben ser capaces de usar objetos de clases derivadas de éstas sin saberlo. (Juan García Carmona, 2012)

Segregación de Interfaces (*Interface segregation*): Los clientes no deberían ser forzados a depender de interfaces que no utilizan. (Juan García Carmona, 2012)

Inyección de Dependencias (*Dependency inversion*): Para conseguir robustez y flexibilidad y para posibilitar la reutilización haz que tu código dependa de abstracciones y no de concreciones, esto es, utiliza muchas interfaces y muchas clases abstractas y, sobre todo, expón, por constructor o por parámetros, las dependencias que una clase pueda tener. (Juan García Carmona, 2012)

3.3 Pruebas de software

Las pruebas de software son importantes porque aseguran el correcto cumplimiento de la funcionalidad del producto, ayudan a ganar confianza, confirman la fiabilidad del uso y previenen defectos en producción, lo cual tiene un impacto económico positivo en la empresa en cuestión. Las pruebas de software son una actividad primordial en el proceso de “aseguramiento de la calidad”. (Campos Chiu, 2015)

Realizar pruebas a una aplicación desarrollada es una actividad fundamental que toda empresa productora de software debería llevar a cabo. Las pruebas de software son un elemento clave en la garantía de la calidad del producto final.

3.3.1 Estrategia de pruebas

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 estrategia de prueba debe indicar los niveles de pruebas (ciclos) que se aplican y la intensidad o profundidad a aplicar para cada nivel de prueba definido.

3.3.2 Niveles de prueba

La prueba es aplicada para diferentes tipos de objetivos, en diferentes escenarios o niveles de trabajo. Se distinguen los siguientes Niveles de Pruebas. (Jacobson, 2000)

- **Prueba de Unidad**: Un componente es la unidad más pequeña especificada de un sistema, las pruebas se llevan a cabo tras la construcción o realización de cada componente para verificar que la implementación se esté llevando conforme a los estándares acordados. El objetivo es comprobar que el sistema, entendido como una unidad funcional, está correctamente codificado.
- **Prueba de Sistema**: Son las pruebas que se hacen cuando el software está funcionando como un todo. Es la actividad de prueba dirigida a verificar el programa final, después que todos los componentes de software y hardware han sido integrados.
- **Prueba de Aceptación**: Es la prueba final antes del despliegue del sistema. El propósito es confirmar que el sistema está terminado, que desarrolla puntualmente las necesidades de la organización y que es aceptado por los usuarios finales.

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

3.3.3 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.

Para la realización de las pruebas unitarias en Python 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. Se realizaron un total de cinco (5) pruebas, la Figura 3 -18 muestra el resultado obtenido.

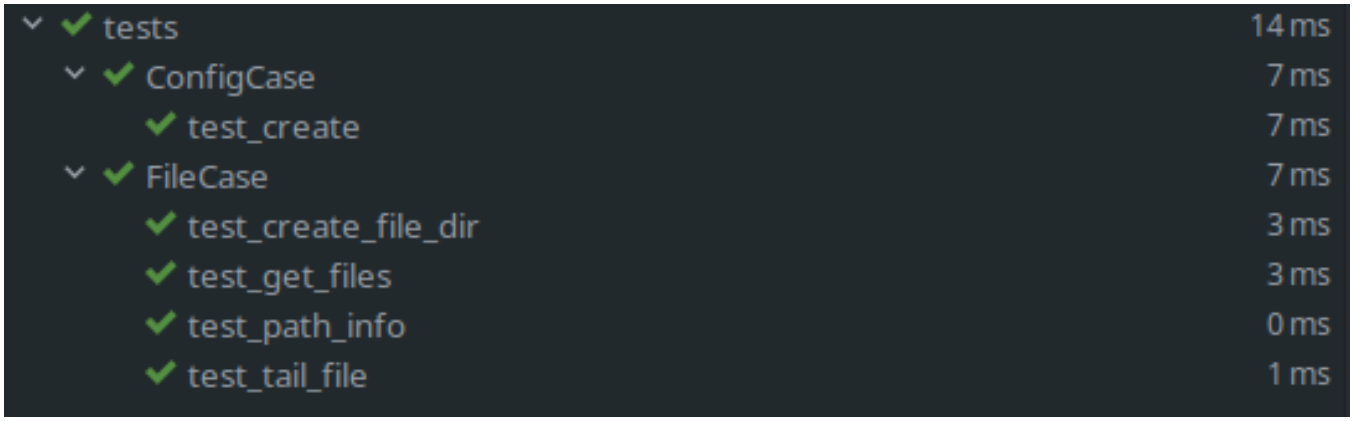


Figura 3-18 Resultado de las pruebas unitarias
Fuente: los autores.

3.3.4 Pruebas de 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), este proceso se muestra en la Figura 3 -19. (Campos Chiu, 2015)

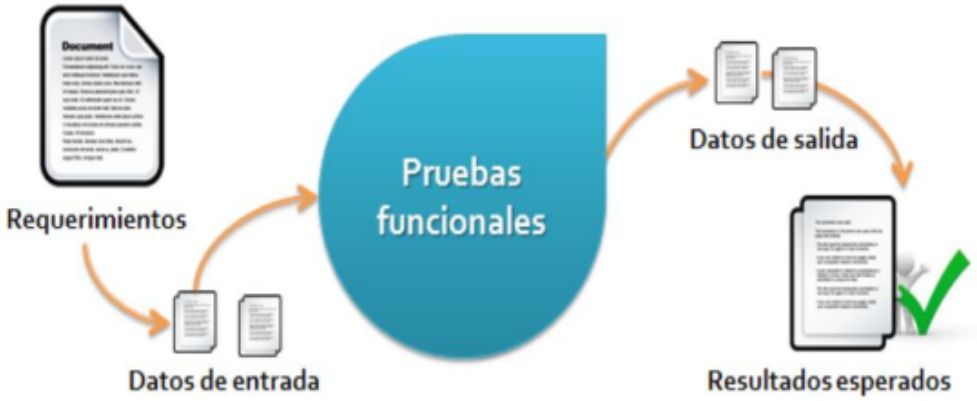


Figura 3-19 Pruebas funcionales
Fuente: CINDY CAMPOS CHIU.

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)

Para la realización de las pruebas funcionales en Python 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. Se realizaron un total de 3 pruebas, se resume en la Figura 3 -20 los resultados de las pruebas aplicadas.

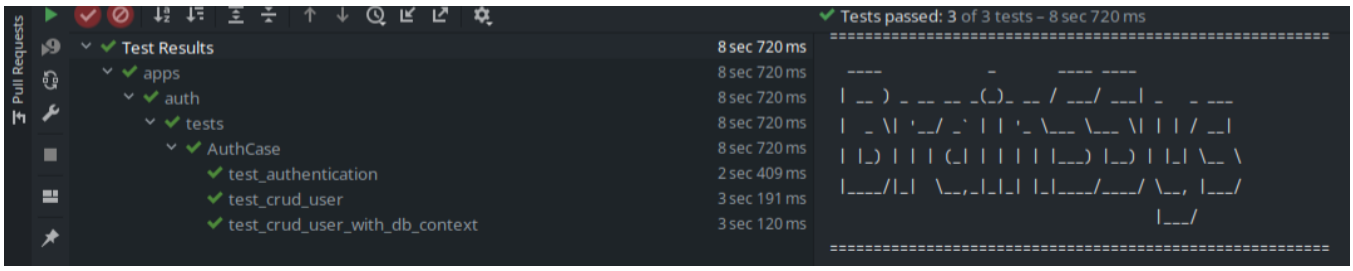


Figura 3-20 Resultado de las pruebas funcionales
Fuente: los autores.

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 Figura 3 -20, 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)

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 3 -21, se concluyó que la librería “ParallelSSH” es más rápida, pero el equipo de desarrollo decidió utilizar “AsyncSSH” debido a todas las ventajas que proporciona.

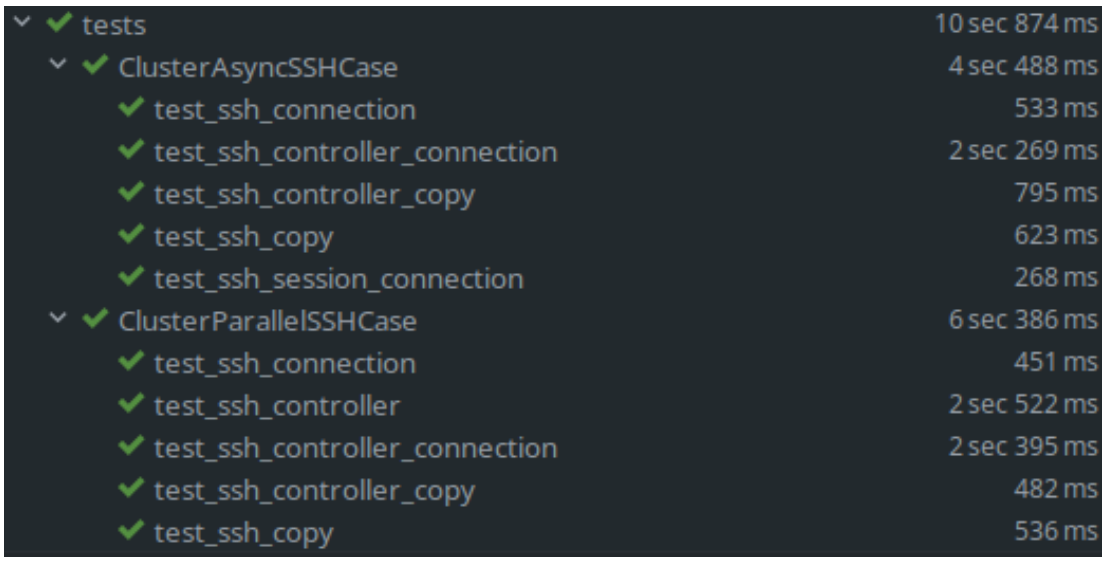


Figura 3-21 Resultado de las pruebas de rendimiento
Fuente: los autores.

3.3.5 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 III-8 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 del software	
Condiciones de ejecución:		Al entrar a la URL del software 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	Se muestra el *dashboard* del software

		usuario presiona el botón de iniciar sesión en el 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 III-9 Escenarios de prueba de aceptación: Autenticar usuario
Fuente: los autores.

Escenarios de prueba	Usuario	Contraseña	Administrador	Activo
EP1	root	password	verdadero	verdadero
	(V)	(V)	(V)	(V)
	Admin	Admin	verdadero	verdadero
	(V)	(V)	(V)	(V)
EP2	Michel	Estropajo1234		
	(I)	(V)	(N/A)	(N/A)
	Brian	TitiMan123		
	(V)	(I)	(N/A)	(N/A)
	Raikol	Prueba1*		
	(I)	(I)	(N/A)	(N/A)
	Danay	test		
	(I)	(I)	(N/A)	(N/A)

Tabla III-10 Caso de prueba de aceptación: Registrar servidor
Fuente: los autores.

CASO DE PRUEBA DE ACEPTACIÓN			
Historia de Usuario:		HU-07: Registrar servidor	
Descripción:		El software 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	Se añade el servidor a la lista de servidores y se muestra una

		validaciones definidas para cada campo	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 III-11 Escenarios de prueba de aceptación: Registrar servidor
Fuente: los autores.

Escenarios de prueba	Nombre	Host	Puerto	Usuario	Contraseña	Time out
EP1	Principal	10.0.0.1	22	Testing	Testing	60
	(V)	(V)	(V)	(V)	(V)	(V)
	Secundario	10.0.0.2	23	Admin	Root	70
	(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	Testing	60
	(V)	(V)	(I)	(V)	(V)	(V)
	Principal	10.0.0.1	156	Usuario-incorrecto	Testing	60
	(V)	(V)	(V)	(I)	(V)	(V)

3.3.6 Resultado de las pruebas aplicadas

La Figura 3 -22 muestra los resultados de aplicar el método de caja negra y caja blanca, donde se ejecutaron un total de cuatro (4) iteraciones. Además, representa el total de no conformidades identificadas por cada iteración.

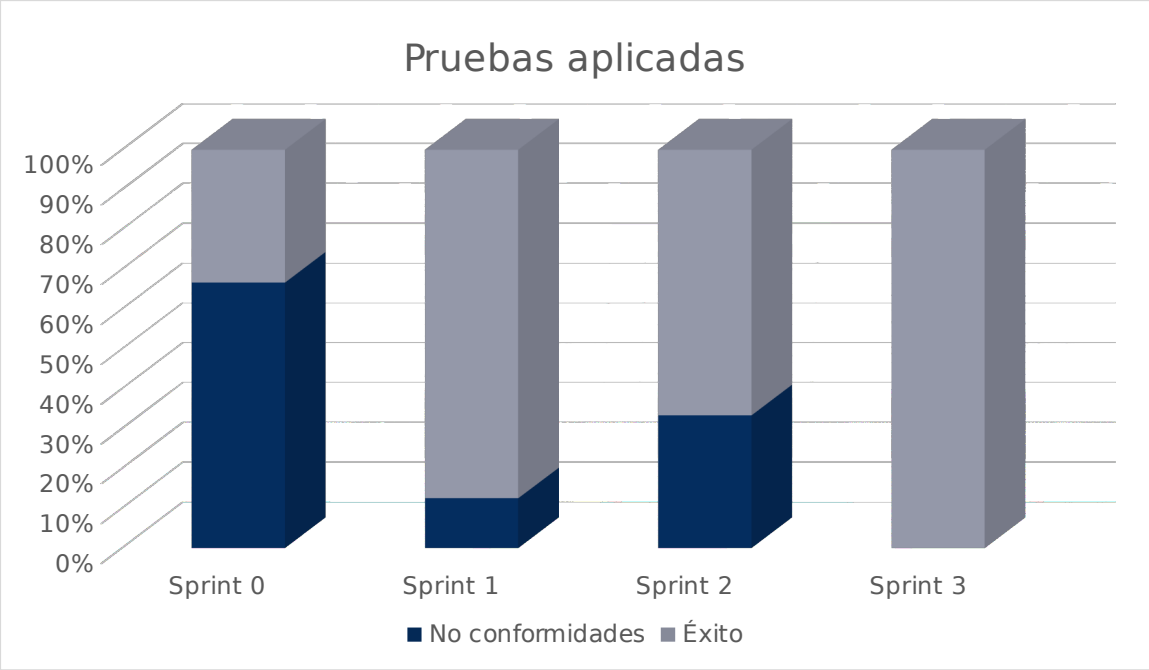


Figura 3-22 Resultado de las pruebas aplicadas
Fuente: los autores.

En la primera iteración se detectaron un total de dos (2) no conformidades tras ejecutar una prueba de aceptación. En la segunda iteración se implementaron las funcionalidades descritas en nueve (9) historias de usuario, se realizaron correcciones a las dos (2) que no superaron la primera iteración de pruebas y se ejecutó una (1) prueba de aceptación y seis (6) pruebas unitarias, solo una (1) resultó no satisfactoria. Todas las no conformidades de la primera iteración fueron subsanadas.

En la tercera iteración, una vez saldados los problemas detectados en la iteración anterior, se terminó de implementar siete (7) historias de usuario. Se repitieron las pruebas anteriores y se aplicaron otras tres (3) con el objetivo de validar las nuevas funcionalidades encontrando una (1) no conformidad. En la cuarta iteración 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 (3) resultaron no satisfactorias. Finalmente, se repitieron todas las pruebas resultando satisfactorias.

3.4 Validación de la investigación

Para la validación de la presente investigación se realizó un pre-experimento, mediante el diseño de pre-prueba y post-prueba con un solo grupo; para comprobar cómo se comporta el grado de usabilidad de la herramienta de gestión del procesamiento de datos de neurociencias, analizando un antes (sin la

herramienta) y un después (con la herramienta), además se mantuvo la heterogeneidad de los participantes con el propósito de asegurar la validez de los resultados.

Teniendo en cuenta para la pre-prueba y la post-prueba los resultados obtenidos mediante el instrumento de medición (encuesta, ver **Anexo 4**) realizado en el Centro Nacional de Neurociencias de Cuba (CNEURO), aplicadas a los profesionales y especialistas que trabajan en el clúster para medir el grado de usabilidad que tiene la herramienta de gestión del procesamiento de datos de neurociencias.

Análisis de los resultados

Se muestran los resultados del análisis, apoyado de gráficos de barras aplicado a las 10 preguntas realizadas en la encuesta para caracterizar el grado de usabilidad de la herramienta a partir de los atributos de usabilidad: facilidad de aprendizaje, eficiencia, recuerdo en el tiempo, tasa de errores, satisfacción del usuario.

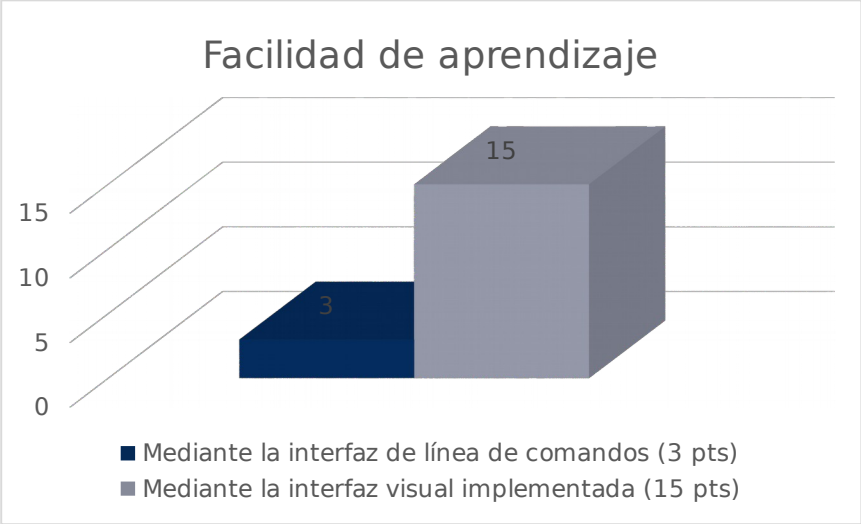


Figura 3-23 Resultado de la encuesta (Facilidad de aprendizaje)
Fuente: los autores.

La comprensión del funcionamiento de la herramienta ayuda a medir el atributo de usabilidad facilidad de aprendizaje, arrojando como resultado con un grado de satisfacción de un 16% utilizando la interfaz de línea de comandos contra un 84% utilizando la herramienta.

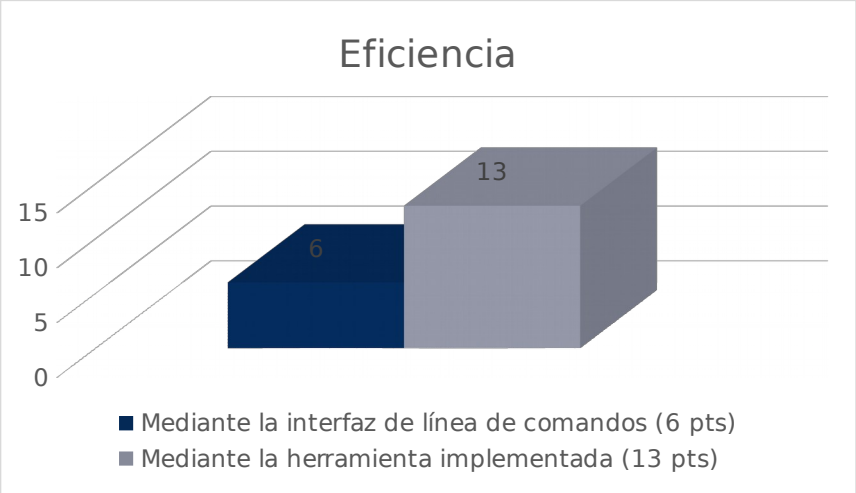


Figura 3-24 Resultado de la encuesta (Eficiencia)
Fuente: los autores.

La rapidez de ejecución de las operaciones de la herramienta ayuda a medir el atributo de usabilidad: eficiencia, arrojando como resultado con un grado de satisfacción de un 31% utilizando la interfaz de línea de comandos contra un 69% utilizando la herramienta.

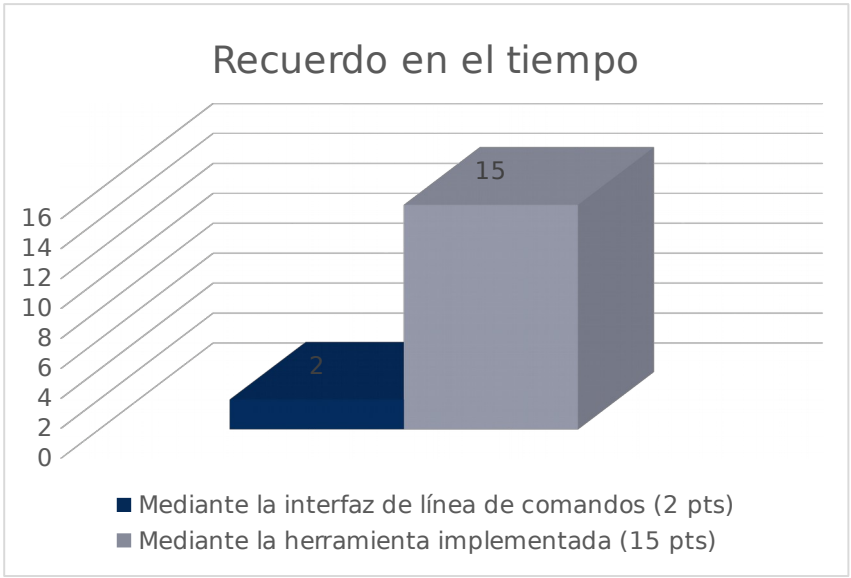


Figura 3-25 Resultado de la encuesta (Recuerdo en el tiempo)
Fuente: los autores.

La cantidad de pasos para realizar las operaciones de la herramienta ayuda a medir el atributo de usabilidad recuerdo en el tiempo, arrojando como resultado con un grado de satisfacción de un 11% utilizando la interfaz de línea de comandos contra un 89% utilizando la herramienta.

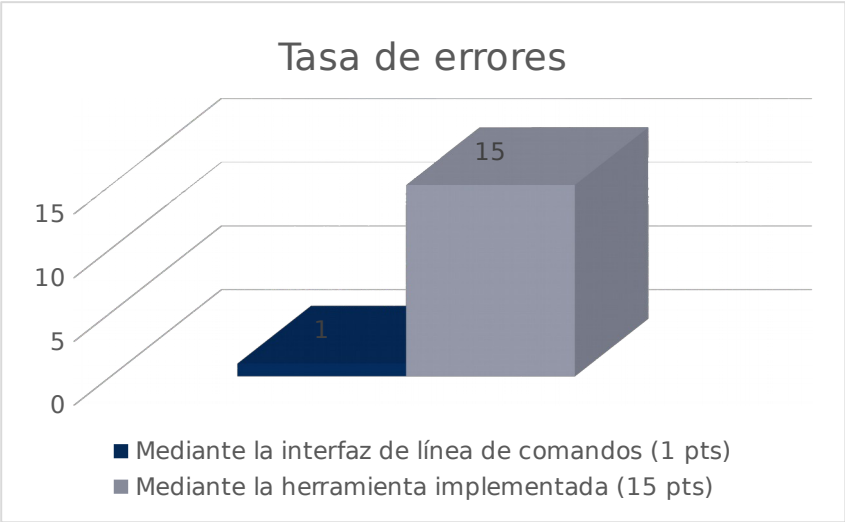


Figura 3-26 Resultado de la encuesta (Tasa de errores)
Fuente: los autores.

La regularidad con la que se cometen errores en las operaciones de la herramienta ayuda a medir el atributo de usabilidad tasa de errores, arrojando como resultado con un grado de satisfacción de un 6% utilizando la interfaz de línea de comandos contra un 94% utilizando la herramienta.

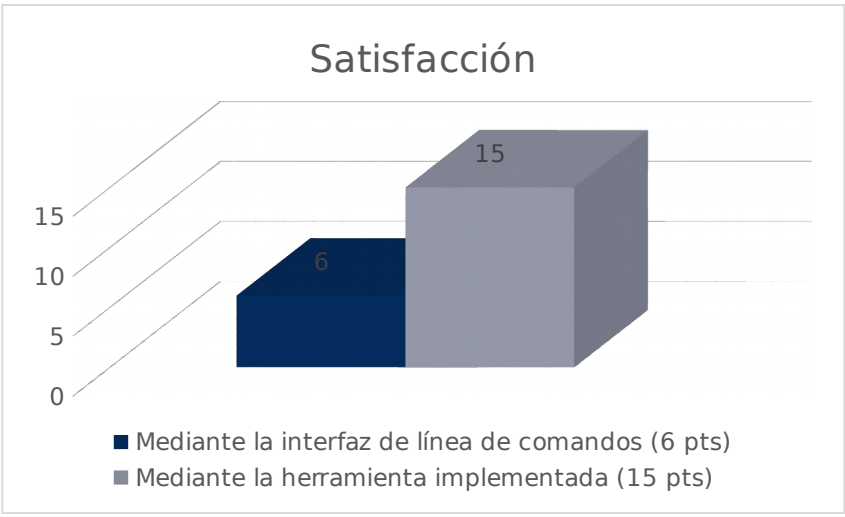


Figura 3-27 Resultado de la encuesta (Satisfacción)
Fuente: los autores.

El nivel de satisfacción al interactuar con la herramienta ayuda a medir el atributo de usabilidad satisfacción, arrojando como resultado con un grado de satisfacción de un 28% utilizando la interfaz de línea de comandos contra un 72% utilizando la herramienta.

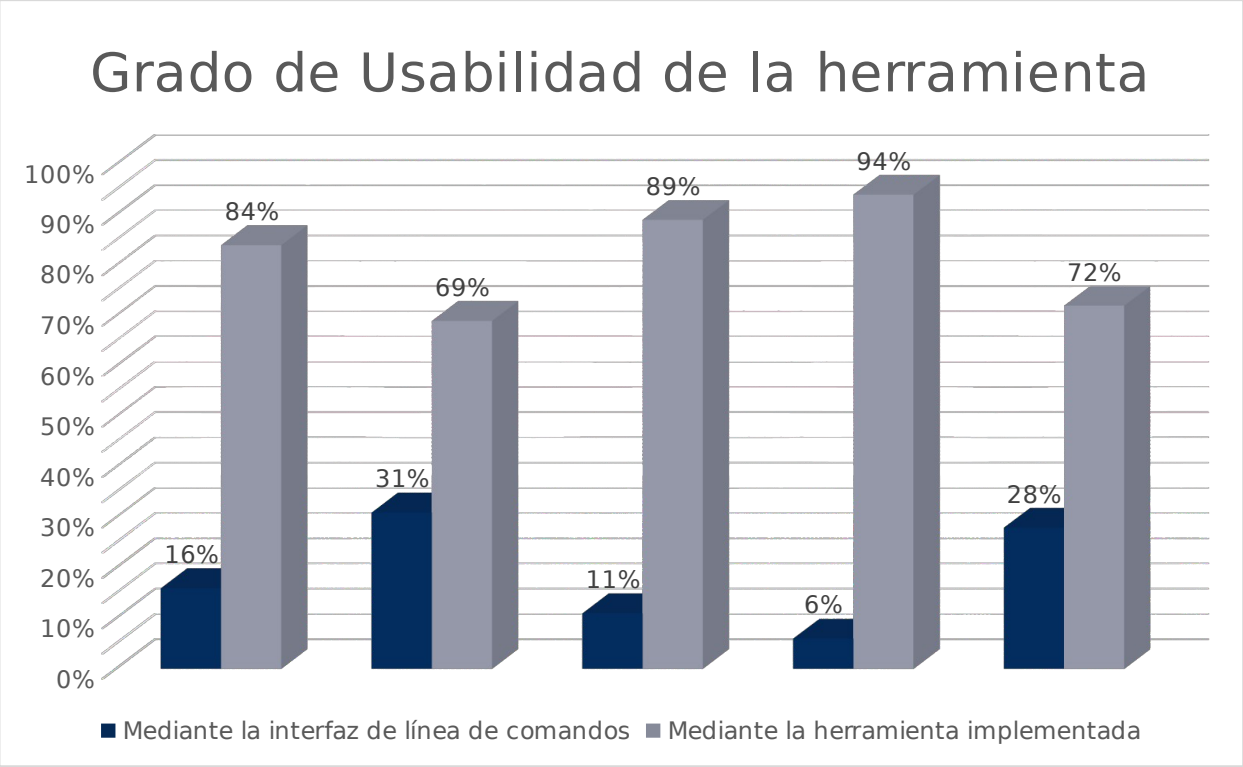


Figura 3-28 Resultados del grado de usabilidad de la herramienta.
Fuente: los autores.

La Figura 3 -28 muestra el resultado de la comparación de los atributos de usabilidad medidos, representándolo en por ciento con respecto al máximo valor posible por cada atributo. Lo que permite obtener un acercamiento al grado de usabilidad utilizando la interfaz de línea de comandos y después con la propuesta de solución. Se demuestra por cada atributo que luego de probada la propuesta de solución se le dio cumplimiento al objetivo planteado en la investigación, mejorando la usabilidad de la interfaz de línea de comando a partir de la herramienta de gestión del procesamiento de datos de neurociencias.

3.5 Conclusiones parciales

La creación del diagrama de despliegue facilitó la comprensión de la distribución del software en el ambiente de producción. La codificación de la solución se ajustó a las pautas y estándares definido en la investigación, obteniendo un código legible y robusto que permite un mantenimiento a largo plazo. La ejecución de las pruebas unitarias y de sistema comprobaron el correcto flujo de trabajo de la solución, así como la identificación y corrección de las no conformidades detectadas en cada iteración. Se validó la usabilidad de la propuesta de solución a partir de la aplicación del método pre-

experimental, demostrando resultados satisfactorios en sus cinco atributos, dando cumplimiento al objetivo planteado en la investigación.

Conclusiones generales

Durante el desarrollo de la presente investigación se ha dado cumplimiento a los objetivos planteados, construyendo una herramienta de gestión de procesamiento de datos de neurociencias para el Centro de Neurociencias de Cuba (CNEURO), que mejora la usabilidad en la ejecución de tareas en el clúster, beneficiando a los profesionales y especialistas de la institución, e incrementando la productividad del personal.

Entre las principales conclusiones a las que se puede arribar con la investigación desarrollada se encuentran:

- Las herramientas identificadas en la literatura constituyen variantes parciales de la solución del problema identificado en la presente investigación, sin embargo, poseen limitantes de usabilidad para investigadores en neurociencias.
- La definición de las herramientas, tecnologías y lenguajes, propiciaron diseñar un ambiente de desarrollo novedoso y ajustado al contexto tecnológico de CNEURO.
- Los artefactos ingenieriles elaborados al aplicar la metodología SCRUM y su posterior implementación, contribuyeron a la obtención de una aplicación informática que responde a las necesidades del cliente.
- La solución propuesta cumple los estándares de integración que garantiza el flujo de información hacia la plataforma BrainSSys, esto fue posible al adoptar patrones, estándares y buenas prácticas de desarrollo de software.
- La herramienta desarrollada propicia ejecutar tareas de procesamiento de datos en HPC desde interfaces gráficas, simplificando la complejidad técnica y aumentando su usabilidad en el entorno de CNEURO.
- La aplicación de la estrategia de validación mostró la validez de la propuesta de solución y resultados satisfactorios en los cinco atributos de usabilidad definidos en esta investigación, lo que permitió constatar el cumplimiento del objetivo general de la investigación.

Recomendaciones

Para dar continuidad a la presente investigación se recomienda:

- Implementar un migrador de descriptores Boutiques a la herramienta desarrollada para migrar los descriptores ya existentes en CBrain que utiliza la institución CNEURO.
- Incorporar una terminal SSH en tiempo real a la herramienta desarrollada para comprobar con mayor eficacia los diferentes scripts creados.

Referencias bibliográficas

akin-ogundeji muyiwa. (2015, noviembre 22). MY VUE ON MVVM. *Medium*. https://medium.com/@nohkachi/my-vue-on-mvvm-c3ffb4f0678f

Ampuero, M. A., & Trujillo, Y. L. (2006). Creando Un Profesional Con Disciplina En El Proceso De Desarrollo De Software. *Ingeniería Industrial*, *XXVII*(1), 27-30.

Antonio Bardají Gálvez. (2021, mayo 4). *El riesgo de los neurodatos: Nuevas perspectivas sobre su utilización—The Technolawgist*. https://www.thetechnolawgist.com/2021/05/04/el-riesgo-de-los-neurodatos-nuevas-perspectivas-sobre-utilizacion/

Arturo Barrera. (2019, octubre 4). *10 razones por las que debes aprender Python*. Blog | NextU LATAM. https://www.nextu.com/blog/razones-aprender-python/

Ausín, T., Morte, R., & Monasterio Astobiza, A. (2020). Neuroderechos: Derechos Humanos para las neurotecnologías. *Diario La Ley*, *43*.

BIL. (2022). Brain Image Library. https://www.brainimagelibrary.org/about.html

BrainKart Team. (2022). *Computer Clusters and MPP Architectures*. BrainKart. https://www.brainkart.com/article/Computer-Clusters-and-MPP-Architectures_11316/

Camacho, E., Jeffries, R., Anderson, A., Hendrickson, C., Cardeso, F., & Nuñez, G. (2004). *Arquitecturas de Software-Guia de Estudio*. Apr-2004.ming Installed". Addison-Wesley.

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

C-BIG. (2022). The Neuro. https://www.mcgill.ca/neuro/open-science/c-big-repository

CNEURO. (2022). *Quienes Somos – Empresa Centro de Neurociencias de Cuba*. https://www.cneuro.cu/index.php/quienes-somos/

Coulouris, G., Dollimore, J., Kindberg, T., & Blair, G. (2011). *Distributed Systems: Concepts and Design* (5th ed.). Addison-Wesley.

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 Neuroinformatics*, *5*, 37. https://doi.org/10.3389/fninf.2011.00037

David Bueno. (2018, marzo 23). Neurociencia: David Bueno explica cómo cambia nuestro cerebro al aprender. *Elisa Aribau*. https://www.elisaribau.com/neurociencia-david-bueno-explica-cambia-cerebro-al-aprender/

Epitech España. (2021, julio 8). Qué es Flask (Python) y cuáles son sus principales ventajas. *Epitech España*. https://www.epitech-it.es/flask-python/

Fernández, Y. (2019, agosto 23). *API: Qué es y para qué sirve*. Xataka. https://www.xataka.com/basics/api-que-sirve

Ferre, X. (2000). *Principios Básicos de Usabilidad para Ingenieros Software.* 39-46.

Flask. (2022). Welcome to Flask — Flask Documentation (2.1.x). https://flask.palletsprojects.com/en/2.1.x/

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

FundéuRAE. (2017, noviembre 1). *Término Clúster*. https://www.fundeu.es/recomendacion/cluster/

GBC. (2022). Global Brain Consortium. https://globalbrainconsortium.org/index.html

Glatard, T., Kiar, G., Aumentado-Armstrong, T., Beck, N., Bellec, P., Bernard, R., Bonnet, A., Brown, S. T., Camarasu-Pop, S., Cervenansky, F., Das, S., Ferreira da Silva, R., Flandin, G., Girard, P., Gorgolewski, K. J., Guttmann, C. R. G., Hayot-Sasson, V., Quirion, P.-O., Rioux, P., … Evans, A. C. (2018). Boutiques: A flexible framework to integrate command-line applications in computing platforms. *GigaScience*, *7*(5), giy016. https://doi.org/10.1093/gigascience/giy016

Gorgolewski, K. J., Varoquaux, G., Rivera, G., Schwarz, Y., Ghosh, S. S., Maumet, C., Sochat, V. V., Nichols, T. E., Poldrack, R. A., Poline, J.-B., Yarkoni, T., & Margulies, D. S. (2015). NeuroVault.org: A web-based repository for collecting and sharing unthresholded statistical maps of the human brain. *Frontiers in Neuroinformatics*, *9*, 8. https://doi.org/10.3389/fninf.2015.00008

Group PostgreSQL Global Development. (2022, junio 12). *PostgreSQL*. PostgreSQL. https://www.postgresql.org/

Henrik Kniberg. (2007). *Scrum y XP desde las trincheras*. https://pdfslide.net/documents/scrum-y-xp-desde-las-trincheras-henrik-kniberg.html

HFSP. (2022). Human Frontier Science Program. https://www.hfsp.org/

IBI. (2022). International Brain Initiative. https://www.internationalbraininitiative.org/

IBRO. (2022). International Brain Research Organization. https://ibro.org/

Iowa State University. (2022). *What is an HPC cluster | High Performance Computing*. https://www.hpc.iastate.edu/guides/introduction-to-hpc-clusters/what-is-an-hpc-cluster

Jacobson, I. (2000). *Uml El proceso unificado de desarrollo de software* (F. J. D. Muñoz, Trad.).

Jeffries, R., Hendrickson, M., Anderson, A., & Hendrickson, C. (2000). *Extreme Programming Installed*.

JetBrains. (2022). *PyCharm*. PyCharm: el IDE de Python para desarrolladores profesionales, por JetBrains. https://www.jetbrains.com/es-es/pycharm/

Johel Jiménez Rivera. (2018, agosto 6). *C#. Qué es y para qué se utiliza | n+e*. https://negociosyestrategia.com/blog/que-es-csharp/

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

Kandel, E. R., Jessell, T. M., & Schwartz, J. H. (1997). *Neurociencia y conducta*.

know-how - IONOS. (2020, octubre 13). *Distributed computing: Computación distribuida para infraestructuras digitales eficientes*. IONOS Digitalguide. https://www.ionos.es/digitalguide/servidores/know-how/que-es-la-computacion-distribuida/

Linne, M.-L. (2018). Neuroinformatics and Computational Modelling as Complementary Tools for Neurotoxicology Studies. *Basic & Clinical Pharmacology & Toxicology*, *123*(S5), 56-61. https://doi.org/10.1111/bcpt.13075

Maastricht. (2022). Maastricht University. https://www.maastrichtuniversity.nl/

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

McGill. (s. f.). McGill University. Recuperado 12 de junio de 2022, de https://www.mcgill.ca/

MCIN. (2022a). *CBrain – McGill Centre for Integrative Neuroscience*. https://mcin.ca/technology/cbrain/

MCIN. (2022b). *LORIS – McGill Centre for Integrative Neuroscience*. https://mcin.ca/technology/loris/

Meghan C. Mott, Gordon, J. A., & Koroshetz, W. J. (2018). The NIH BRAIN Initiative: Advancing neurotechnologies, integrating disciplines. *PLOS Biology*, *16*(11), e3000066. https://doi.org/10.1371/journal.pbio.3000066

Molina, J., Moreira, A., & Rossi, G. (2004). UML: El lenguaje estándar para el modelado de software. *Novática: Revista de la Asociación de Técnicos de Informática, ISSN 0211-2124, Nº. 168, 2004 (Ejemplar dedicado a: UML e Ingeniería de Modelos), pags. 4-5*.

Müller, O., & Rotter, S. (2017). Neurotechnology: Current Developments and Ethical Issues. *Frontiers in Systems Neuroscience*, *11*. https://www.frontiersin.org/articles/10.3389/fnsys.2017.00093

NeuroMorpho. (2022). NeuroMorpho.Org - a centrally curated inventory of digitally reconstructed neurons and glia. https://neuromorpho.org/

NeuroVault. (2022). NeuroVault: an open data repository for brain maps. https://neurovault.org/

Pascual, J. R. (2019, octubre 22). Capas y niveles: Diseño y confusión. *Disrupción Tecnológica*. https://www.disrupciontecnologica.com/capas-y-niveles-diseno-y-confusion/

Pressman, R. S. (2010). *Ingeniería del software un enfoque práctico* (7a.ed.). McGraw-Hill.

Revisión de metodologías ágiles para el desarrollo de software. (2013). *Prospectiva*, *11*(2), 30-39.

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

ROVIRA, C. (2001). Herramientas de ayuda a la navegación. *Temes de disseny*, *18*, 66-73.

RWTH-Aachen. (2022). RWTH International Academy. https://www.academy.rwth-aachen.de

Sebastian Gräf. (2021, agosto 9). *Conception, Construction and Evaluation of a Raspberry Pi Cluster*. https://graef.io/conception-construction-and-evaluation-of-a-raspberry-pi-cluster/

Sebastián Ramírez. (2022). *FastAPI*. https://fastapi.tiangolo.com/es/

Sherif, T., Rioux, P., Rousseau, M.-E., Kassis, N., Beck, N., Adalat, R., Das, S., Glatard, T., & Evans, A. C. (2014). CBRAIN: A web-based, distributed computing platform for collaborative neuroimaging research. *Frontiers in Neuroinformatics*, *8*, 54. https://doi.org/10.3389/fninf.2014.00054

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

Squire, L., Berg, D., Bloom, F. E., Lac, S. du, Ghosh, A., & Spitzer, N. C. (2012). *Fundamental Neuroscience*. Academic Press.

Sublime Text. (2022). Sublime Text - the sophisticated text editor for code, markup and prose. https://www.sublimetext.com/

TechEmpower. (2021, febrero 8). *TechEmpower Web Framework Performance Comparison*. Www.Techempower.Com. https://www.techempower.com/benchmarks/#section=data-r20&hw=ph&test=query&l=zijzen-64f&f=0-0-0-0-0-1ekg-295p8g-hra0hs-1ekg-0-6bk-1ekg-0-0&b=4&o=8

UESTC. (2022). University of Electronic Science and Technology of China. https://en.uestc.edu.cn/

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

Vitaliy Ilyukha. (2022). *Vue vs React: Choosing a Front-End Framework for Project*. Jelvix. https://jelvix.com/blog/js-frameworks-is-vuejs-better-than-react

Yoo, A. B., Jette, M. A., & Grondona, M. (2003). SLURM: Simple Linux Utility for Resource Management. En D. Feitelson, L. Rudolph, & U. Schwiegelshohn (Eds.), *Job Scheduling Strategies for Parallel Processing* (pp. 44-60). Springer. https://doi.org/10.1007/10968987_3

Anexos

Anexo 1: Entrevista

1. ¿Como funciona el clúster en CNEURO?

El clúster cuenta con un sistema de colas encargado de repartir las diferentes tareas por los nodos existentes. De este modo, los especialistas del centro, a partir de instrucciones por consola, sólo tendrá que lanzar sus tareas y el gestor de cola se encargará de encontrar nodos libres en los que ejecutar los cálculos. En caso de que estén todos ocupados, el gestor de colas se encarga de poner las tareas en espera y de lanzarla cuando haya recursos disponibles, sin necesidad de que el usuario esté conectado esperando.

2. ¿Cómo ejecutan tareas en el clúster?

Es necesario acceder mediante SSH al nodo maestro y ejecutar el comando o instrucción correspondiente a la tarea a realizar.

3. ¿Qué herramientas o sistemas usan en el clúster?

En el centro se emplea CBRAIN para realizar estudios y SLURM como gestor de colas en el clúster.

4. ¿Qué problemas tienen actualmente?

Los especialistas se ven obligados a programar instrucciones en consola de forma manual para ejecutar tareas de procesamiento y análisis, y, al no estar capacitados, necesitan el apoyo de un recurso humano que disponga de las habilidades técnicas necesarias.

5. ¿Qué mejoras pudieran ser añadidas?

Una herramienta de procesamiento de tareas que sea amigable e intuitiva facilitando el trabajo desarrollado por los especialistas.

Anexo 2: Historias de Usuario

Tabla 12 Historia de Usuario: Autenticar usuario
Fuente: los autores.

UH01 – 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 13 Historia de Usuario: Registrar usuario
Fuente: los autores.

UH02 – 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 14 Historia de Usuario: Modificar usuario
Fuente: los autores.

UH03 – 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 15 Historia de Usuario: Visualizar detalles del usuario
Fuente: los autores.

UH04 – 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 16 Historia de Usuario: Eliminar usuario

Fuente: los autores.

UH05 – 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 17 Historia de Usuario: Listar usuario
Fuente: los autores.

UH06 – 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 18 Historia de Usuario: Modificar servidor
Fuente: los autores.

UH08 – 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 19 Historia de Usuario: Visualizar detalles del servidor
Fuente: los autores.

UH09 –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 20 Historia de Usuario: Eliminar servidor

Fuente: los autores.

UH10 – 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 21 Historia de Usuario: Listar servidores
Fuente: los autores.

UH11 – 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 22 Historia de Usuario: Registrar script
Fuente: los autores.

UH12 – 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 23 Historia de Usuario: Modificar script
Fuente: los autores.

UH13 – 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 24 Historia de Usuario: Eliminar script
Fuente: los autores.

UH14 – 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 25 Historia de Usuario: Listar scripts
Fuente: los autores.

UH15 – 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 26 Historia de Usuario: Visualizar detalles de un script
Fuente: los autores.

UH16 – 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 27 Historia de Usuario: Concatenar programas del script
Fuente: los autores.

UH17 – 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 28 Historia de Usuario: Registrar programa
Fuente: los autores.

UH18 – 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 29 Historia de Usuario: Modificar programa
Fuente: los autores.

UH19 – 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 30 Historia de Usuario: Visualizar detalles del programa
Fuente: los autores.

UH20 – 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 31 Historia de Usuario: Eliminar programa
Fuente: los autores.

UH21 – 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 32 Historia de Usuario: Listar programas
Fuente: los autores.

UH22 – 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 33 Historia de Usuario: Obtener tarea
Fuente: los autores.

UH24 – 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 34 Historia de Usuario: Detener tarea
Fuente: los autores.

UH25 – 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 35 Historia de Usuario: Comprobar tareas
Fuente: los autores.

UH26 – 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 36 Historia de Usuario: Obtener registro de la tarea
Fuente: los autores.

UH27 – 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 37 Historia de Usuario: Obtener registro de errores de la tarea
Fuente: los autores.

UH28 – 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 3: Sprint Backlogs

Tabla 38 Sprint backlog #1
Fuente: los autores.

Sprint 1				
ID	**Tarea**	**Asignada a**	**Estado**	**Tiempo**
UH2-T1	Diseño e implementación de clases POCO	Michel Suárez	Terminada	2
UH2-T2	Migración a la base de datos	Michel Suárez	Terminada	2
UH2-T3	Implementación de los repositorios para el acceso a datos	Michel Suárez	Terminada	4
UH2-T4	Implementación de la lógica de la funcionalidad	Michel Suárez	Terminada	4
UH2-T5	Creación de los endpoints	Michel Suárez	Terminada	2
UH2-T6	Diseño e implementación de los DTO (Data Transfer Object)	Michel Suárez	Terminada	2
UH2-T7	Diseño del prototipado de la interfaz	Brian Pérez	Terminada	4
UH2-T8	Implementación de la funcionalidad en el cliente web	Brian Pérez	Terminada	6
UH1-T1	Implementación de los repositorios para el acceso a datos	Michel Suárez	Terminada	2
UH1-T2	Implementación de la lógica de la funcionalidad	Michel Suárez	Terminada	2
UH1-T3	Creación de los endpoints	Michel Suárez	Terminada	2
UH1-T4	Diseño e implementación de los DTO (Data Transfer Object)	Michel Suárez	Terminada	2
UH1-T5	Diseño del prototipado de la interfaz	Brian Pérez	Terminada	4
UH1-T6	Implementación de la funcionalidad en el cliente web	Brian Pérez	Terminada	4
UH3-T1	Implementación de los repositorios para el acceso	Michel Suárez	Terminada	2

	a datos			
UH3-T2	Implementación de la lógica de la funcionalidad	Michel Suárez	Terminada	2
UH3-T3	Creación de los endpoints	Michel Suárez	Terminada	2
UH3-T4	Diseño e implementación de los DTO (Data Transfer Object)	Michel Suárez	Terminada	2
UH3-T5	Diseño del prototipado de la interfaz	Brian Pérez	Terminada	4
UH3-T6	Implementación de la funcionalidad en el cliente web	Brian Pérez	Terminada	4
UH4-T1	Implementación de los repositorios para el acceso a datos	Michel Suárez	Terminada	2
UH4-T2	Implementación de la lógica de la funcionalidad	Michel Suárez	Terminada	2
UH4-T3	Creación de los endpoints	Michel Suárez	Terminada	2
UH4-T4	Diseño e implementación de los DTO (Data Transfer Object)	Michel Suárez	Terminada	2
UH4-T5	Diseño del prototipado de la interfaz	Brian Pérez	Terminada	4
UH4-T6	Implementación de la funcionalidad en el cliente web	Brian Pérez	Terminada	4
UH5-T1	Implementación de los repositorios para el acceso a datos	Michel Suárez	Terminada	2
UH5-T2	Implementación de la lógica de la funcionalidad	Michel Suárez	Terminada	2
UH5-T3	Creación de los endpoints	Michel Suárez	Terminada	2
UH5-T4	Diseño e implementación de los DTO (Data Transfer Object)	Michel Suárez	Terminada	2
UH5-T5	Diseño del prototipado de la interfaz	Brian Pérez	Terminada	4
UH5-T6	Implementación de la funcionalidad en el cliente web	Brian Pérez	Terminada	4
UH6-T1	Implementación de los repositorios para el acceso a datos	Michel Suárez	Terminada	2
UH6-T2	Implementación de la lógica de la funcionalidad	Michel Suárez	Terminada	2
UH6-T3	Creación de los endpoints	Michel Suárez	Terminada	2
UH6-T4	Diseño e implementación de los DTO (Data Transfer	Michel Suárez	Terminada	2

	Object)			
UH6-T5	Diseño del prototipado de la interfaz	Brian Pérez	Terminada	4
UH6-T6	Implementación de la funcionalidad en el cliente web	Brian Pérez	Terminada	4
UH8-T1	Implementación de los repositorios para el acceso a datos	Michel Suárez	Terminada	2
UH8-T2	Implementación de la lógica de la funcionalidad	Michel Suárez	Terminada	2
UH8-T3	Creación de los endpoints	Michel Suárez	Terminada	2
UH8-T4	Diseño e implementación de los DTO (Data Transfer Object)	Michel Suárez	Terminada	2
UH8-T5	Diseño del prototipado de la interfaz	Brian Pérez	Terminada	4
UH8-T6	Implementación de la funcionalidad en el cliente web	Brian Pérez	Terminada	4
UH9-T1	Implementación de los repositorios para el acceso a datos	Michel Suárez	Terminada	2
UH9-T2	Implementación de la lógica de la funcionalidad	Michel Suárez	Terminada	2
UH9-T3	Creación de los endpoints	Michel Suárez	Terminada	2
UH9-T4	Diseño e implementación de los DTO (Data Transfer Object)	Michel Suárez	Terminada	2
UH9-T5	Diseño del prototipado de la interfaz	Brian Pérez	Terminada	4
UH9-T6	Implementación de la funcionalidad en el cliente web	Brian Pérez	Terminada	4
UH10-T1	Implementación de los repositorios para el acceso a datos	Michel Suárez	Terminada	2
UH10-T2	Implementación de la lógica de la funcionalidad	Michel Suárez	Terminada	2
UH10-T3	Creación de los endpoints	Michel Suárez	Terminada	2
UH10-T4	Diseño e implementación de los DTO (Data Transfer Object)	Michel Suárez	Terminada	2
UH10-T5	Diseño del prototipado de la interfaz	Brian Pérez	Terminada	4
UH10-T6	Implementación de la funcionalidad en el cliente web	Brian Pérez	Terminada	4

Tabla 39 Sprint backlog #2
Fuente: los autores.

Sprint 2				
ID	**Tarea**	**Asignada a**	**Estado**	**Tiempo**
UH18-T1	Diseño e implementación de clases POCO	Michel Suárez	Terminada	2
UH18-T2	Migración a la base de datos	Michel Suárez	Terminada	2
UH18-T3	Implementación de los repositorios para el acceso a datos	Michel Suárez	Terminada	4
UH18-T4	Implementación de la lógica de la funcionalidad	Michel Suárez	Terminada	4
UH18-T5	Creación de los endpoints	Michel Suárez	Terminada	2
UH18-T6	Diseño e implementación de los DTO (Data Transfer Object)	Michel Suárez	Terminada	2
UH18-T7	Diseño del prototipado de la interfaz	Brian Pérez	Terminada	4
UH18-T8	Implementación de la funcionalidad en el cliente web	Brian Pérez	Terminada	6
UH17-T1	Diseño e implementación de clases POCO	Michel Suárez	Terminada	2
UH17-T2	Migración a la base de datos	Michel Suárez	Terminada	2
UH17-T3	Implementación de los repositorios para el acceso a datos	Michel Suárez	Terminada	4
UH17-T4	Implementación de la lógica de la funcionalidad	Michel Suárez	Terminada	4
UH17-T5	Creación de los endpoints	Michel Suárez	Terminada	2
UH17-T6	Diseño e implementación de los DTO (Data Transfer Object)	Michel Suárez	Terminada	2
UH17-T7	Diseño del prototipado de la interfaz	Brian Pérez	Terminada	4
UH17-T8	Implementación de la funcionalidad en el cliente web	Brian Pérez	Terminada	6
UH19-T1	Implementación de los repositorios para el acceso a datos	Michel Suárez	Terminada	2
UH19-T2	Implementación de la lógica de la funcionalidad	Michel Suárez	Terminada	2
UH19-T3	Creación de los endpoints	Michel Suárez	Terminada	2
UH19-T4	Diseño e implementación de los DTO (Data Transfer	Michel Suárez	Terminada	2

	Object)			
UH19-T5	Diseño del prototipado de la interfaz	Brian Pérez	Terminada	4
UH19-T6	Implementación de la funcionalidad en el cliente web	Brian Pérez	Terminada	4
UH20-T1	Implementación de los repositorios para el acceso a datos	Michel Suárez	Terminada	2
UH20-T2	Implementación de la lógica de la funcionalidad	Michel Suárez	Terminada	2
UH20-T3	Creación de los endpoints	Michel Suárez	Terminada	2
UH20-T4	Diseño e implementación de los DTO (Data Transfer Object)	Michel Suárez	Terminada	2
UH20-T5	Diseño del prototipado de la interfaz	Brian Pérez	Terminada	4
UH20-T6	Implementación de la funcionalidad en el cliente web	Brian Pérez	Terminada	4
UH21-T1	Implementación de los repositorios para el acceso a datos	Michel Suárez	Terminada	2
UH21-T2	Implementación de la lógica de la funcionalidad	Michel Suárez	Terminada	2
UH21-T3	Creación de los endpoints	Michel Suárez	Terminada	2
UH21-T4	Diseño e implementación de los DTO (Data Transfer Object)	Michel Suárez	Terminada	2
UH21-T5	Diseño del prototipado de la interfaz	Brian Pérez	Terminada	4
UH21-T6	Implementación de la funcionalidad en el cliente web	Brian Pérez	Terminada	4
UH23-T1	Implementación de los repositorios para el acceso a datos	Michel Suárez	Terminada	2
UH23-T2	Implementación de la lógica de la funcionalidad	Michel Suárez	Terminada	2
UH23-T3	Creación de los endpoints	Michel Suárez	Terminada	2
UH23-T4	Diseño e implementación de los DTO (Data Transfer Object)	Michel Suárez	Terminada	2
UH23-T5	Diseño del prototipado de la interfaz	Brian Pérez	Terminada	4
UH23-T6	Implementación de la funcionalidad en el cliente web	Brian Pérez	Terminada	4

UH22-T1	Implementación de los repositorios para el acceso a datos	Michel Suárez	Terminada	2
UH22-T2	Implementación de la lógica de la funcionalidad	Michel Suárez	Terminada	2
UH22-T3	Creación de los endpoints	Michel Suárez	Terminada	2
UH22-T4	Diseño e implementación de los DTO (Data Transfer Object)	Michel Suárez	Terminada	2
UH22-T5	Diseño del prototipado de la interfaz	Brian Pérez	Terminada	4
UH22-T6	Implementación de la funcionalidad en el cliente web	Brian Pérez	Terminada	4

Tabla 40 Sprint backlog #3
Fuente: los autores.

Sprint 3				
ID	**Tarea**	**Asignada a**	**Estado**	**Tiempo**
UH24-T2	Implementación de la lógica de la funcionalidad	Michel Suárez	Pendiente	2
UH24-T3	Creación de los endpoints	Michel Suárez	Pendiente	2
UH24-T4	Diseño e implementación de los DTO (Data Transfer Object)	Michel Suárez	Pendiente	2
UH24-T5	Diseño del prototipado de la interfaz	Brian Pérez	Pendiente	4
UH24-T6	Implementación de la funcionalidad en el cliente web	Brian Pérez	Pendiente	4
UH25-T1	Implementación de los repositorios para el acceso a datos	Michel Suárez	Pendiente	2
UH25-T2	Implementación de la lógica de la funcionalidad	Michel Suárez	Pendiente	2
UH25-T3	Creación de los endpoints	Michel Suárez	Pendiente	2
UH25-T4	Diseño e implementación de los DTO (Data Transfer Object)	Michel Suárez	Pendiente	2
UH25-T5	Diseño del prototipado de la interfaz	Brian Pérez	Pendiente	4
UH25-T6	Implementación de la funcionalidad en el cliente web	Brian Pérez	Pendiente	4
UH26-T1	Implementación de los repositorios para el acceso a datos	Michel Suárez	Pendiente	2
UH26-T2	Implementación de la	Michel Suárez	Pendiente	2

	lógica de la funcionalidad			
UH26-T3	Creación de los endpoints	Michel Suárez	Pendiente	2
UH26-T4	Diseño e implementación de los DTO (Data Transfer Object)	Michel Suárez	Pendiente	2
UH26-T5	Diseño del prototipado de la interfaz	Brian Pérez	Pendiente	4
UH26-T6	Implementación de la funcionalidad en el cliente web	Brian Pérez	Pendiente	4
UH27-T1	Implementación de los repositorios para el acceso a datos	Michel Suarez	Pendiente	2
UH27-T2	Implementación de la lógica de la funcionalidad	Michel Suárez	Pendiente	2
UH27-T3	Creación de los endpoints	Michel Suárez	Pendiente	2
UH27-T4	Diseño e implementación de los DTO (Data Transfer Object)	Michel Suárez	Pendiente	2
UH27-T5	Diseño del prototipado de la interfaz	Brian Pérez	Pendiente	4
UH27-T6	Implementación de la funcionalidad en el cliente web	Brian Pérez	Pendiente	4
UH28-T1	Implementación de los repositorios para el acceso a datos	Michel Suárez	Pendiente	2
UH28-T2	Implementación de la lógica de la funcionalidad	Michel Suárez	Pendiente	2
UH28-T3	Creación de los endpoints	Michel Suárez	Pendiente	2
UH28-T4	Diseño e implementación de los DTO (Data Transfer Object)	Michel Suárez	Pendiente	2
UH28-T5	Diseño del prototipado de la interfaz	Brian Pérez	Pendiente	4
UH28-T6	Implementación de la funcionalidad en el cliente web	Brian Pérez	Pendiente	4

Anexo 4: Encuesta sobre el uso de la herramienta de gestión del procesamiento de datos de neurociencias.

Encuesta aplicada a los profesionales y especialistas del Centro Nacional de Neurociencias de Cuba (CNEURO) que trabajan con el clúster, para determinar el grado de usabilidad que tiene la herramienta.

Homologías

Facilidad de aprendizaje: Cuán fácil es aprender la funcionalidad básica del sistema.

Eficiencia: El número de transacciones por unidad de tiempo que el usuario puede realizar usando el sistema.

Recuerdo en el tiempo: Para usuarios intermitentes (que no utilizan el sistema regularmente) es vital ser capaces de usar el sistema sin tener que aprender cómo funciona partiendo de cero.

Tasa de errores: El número de errores cometidos por el usuario mientras realiza una determinada tarea.

Satisfacción: Muestra la impresión subjetiva que el usuario obtiene del sistema.

1. ¿Qué tan fácil es aprender sobre la gestión de tareas sin la herramienta?

   __Muy fácil __Fácil __Normal __Difícil __Muy difícil
2. ¿Qué tan fácil es aprender sobre la gestión de tareas con la herramienta implementada?
   __Muy fácil __Fácil __Normal __Difícil __Muy difícil
3. ¿Qué tan eficiente se siente haciendo la gestión de tareas sin la herramienta?
   __ Muy insatisfecho __ Insatisfecho __ Poco satisfecho __ Satisfecho __Muy Satisfecho
4. ¿Qué tan eficiente se siente haciendo la gestión de tareas con la herramienta implementada?
   __ Muy insatisfecho __ Insatisfecho __ Poco satisfecho __ Satisfecho __Muy Satisfecho
5. ¿Qué tan fácil le resulta recordar como interactuar sin la herramienta?
   __Muy fácil __Fácil __Normal __Difícil __Muy difícil
6. ¿Qué tan fácil le resulta recordar como interactuar con la herramienta implementada?
   __Muy fácil __Fácil __Normal __Difícil __Muy difícil
7. ¿Cuál es la probabilidad de que no cometa errores sin la herramienta?
   __Siempre __Casi siempre __Pocas veces __Casi nunca __ Nunca
8. ¿Cuál es la probabilidad de que no cometa errores con la herramienta implementada?
   __Siempre __Casi siempre __Pocas veces __Casi nunca __ Nunca
9. ¿Qué tan satisfecho se siente interactuando sin la herramienta?
   __ Muy insatisfecho __ Insatisfecho __ Poco satisfecho __ Satisfecho __Muy Satisfecho
10. ¿Qué tan satisfecho se siente interactuando con la herramienta implementada?
    __ Muy insatisfecho __ Insatisfecho __ Poco satisfecho __ Satisfecho __Muy Satisfecho