Revista Cubana de Ciencias Informáticas
Vol. 17, No. 4, Octubre-Diciembre, 2023
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 43-59

Tipo de artículo: Artículo original
Temática: Desarrollo de aplicaciones informáticas
Recibido: 07/10/2023 | Aceptado: 08/11/2023

Herramienta para la gestión de colas de procesamientos de datos de neurociencias en HPC

Tool for managing neuroscience data processing queues in HPC

Arturo Orellana García[1*] https://orcid.org/0000-0002-3652-969X
Michel Suárez Morales[1] https://orcid.org/0000-0002-3169-0070
Brian Pérez López1 https://orcid.org/0000-0002-3514-2063

[1]Universidad de las Ciencias Informáticas. La Habana, Cuba.

*Autor para la correspondencia. (aorellana@uci.cu)

RESUMEN

En el campo de la neurociencia se generan crecientes volúmenes de información asociadas a 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. Lo anterior es realizado mediante el uso de comandos en una terminal segura (SSH) y requiere conocimientos técnicos que los especialistas de este campo no disponen, esto provoca errores humanos al ejecutar tareas de procesamiento. El trabajo tiene como objetivo presentar una herramienta para la gestión de colas de procesamientos de datos de neurociencias en un HPC. Se realizó un análisis documental de distintas investigaciones antecedentes que permitieron identificar las mejores variantes a tener en cuenta para la

Revista Cubana de Ciencias Informáticas
Vol. 17, No. 4, Octubre-Diciembre, 2023
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 43-59

propuesta de solución. Como ambiente de desarrollo se definieron Python, FastAPI, Javascript, Vue.js y WebSocket; además se empleó como herramienta de desarrollo PyCharm Community. Se obtuvo una herramienta que mejora la usabilidad del proceso de lanzamiento de tareas en un clúster, incrementando la productividad y disminuyendo los errores producidos por el factor humano. La gestión de las colas de procesamiento de datos de neurociencias, a partir de interfaces y la automatización de los flujos, disminuye el error humano en el lanzamiento de tareas en el servidor HPC.

Palabras clave: clúster; infraestructura; neurociencia; procesamiento.

ABSTRACT

In the field of neuroscience, increasing volumes of information associated with neurological research are generated, which can be processed thanks to the evolution of computational infrastructures and new technologies, and through the use of High-Performance Computing. This is done through the use of commands in a secure terminal (SSH) and requires technical knowledge that specialists in this field do not have, which causes human errors when executing processing tasks. The objective was to present a tool for the management of neuroscience data processing queues in an HPC. A documentary analysis of different background research was carried out to identify the best variants to take into account for the proposed solution. Python, FastAPI, Javascript, Vue.js and WebSocket were defined as the development environment; PyCharm Community was also used as the development tool. A tool was obtained that improves the usability of the task launching process in a cluster, increasing productivity and reducing errors caused by the human factor. The management of neuroscience data processing queues, based on interfaces and flow automation, reduces human error in the launching of tasks in the HPC server.

Keywords: cluster; infrastructure; neuroscience; processing.

Revista Cubana de Ciencias Informáticas
Vol. 17, No. 4, Octubre-Diciembre, 2023
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 43-59

Introducción

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 la Academia Nacional 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. (Shepherd et al., 1998; Redolfi et al., 2023)

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. Estos son aparatos o torres 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. (Willett et al., 2021; Torabzadehkashi et al., 2019)

Se han desarrollado plataformas para el procesamiento y análisis de datos de neurociencias en HPC producto de las alianzas entre diferentes países, patrocinados por proyectos y financiamientos internacionales; entre ellos CBRAIN (Sanz-Robinson et al., 2022), un sistema web para el estudio de neuro datos y LORIS (*Longitudinal Online Research and Imaging System*) o Sistema Longitudinal de Investigación e Imágenes en Línea, un software de gestión de proyectos y datos basado en la web para estudios de investigación de neuroimagen. (Poline et al., 2023)

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. (González-Losada et al., 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.

Revista Cubana de Ciencias Informáticas
Vol. 17, No. 4, Octubre-Diciembre, 2023
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 43-59

- 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 lo anterior la investigación abordó como objetivo principal desarrollar una herramienta para la gestión de colas de procesamientos de datos de neurociencias en un HPC, que contribuya a la solución de las deficiencias identificadas en la problemática.

Métodos o Metodología Computacional

Para la ejecución de la presente investigación se sigue una estrategia explicativa y se emplearon los métodos: análisis documental para obtener datos e información asociados al objeto de estudio. Se analizaron los documentos bibliográficos referentes a soluciones similares, sistemas de ejecución de procesos en un clúster, su integración, así como estándares y buenas prácticas, lo que permitió establecer los fundamentos teóricos. Mediante el método comparativo: establecer la comparación entre las diferentes librerías y marcos de trabajo, y poder establecer cuáles son los que van a utilizar para el desarrollo de la herramienta.

Resultados y discusión

La propuesta ha sido diseñada para el 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

Revista Cubana de Ciencias Informáticas
Vol. 17, No. 4, Octubre-Diciembre, 2023
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 43-59

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. En la Figura 1, se modela el funcionamiento del proceso de la propuesta de solución.

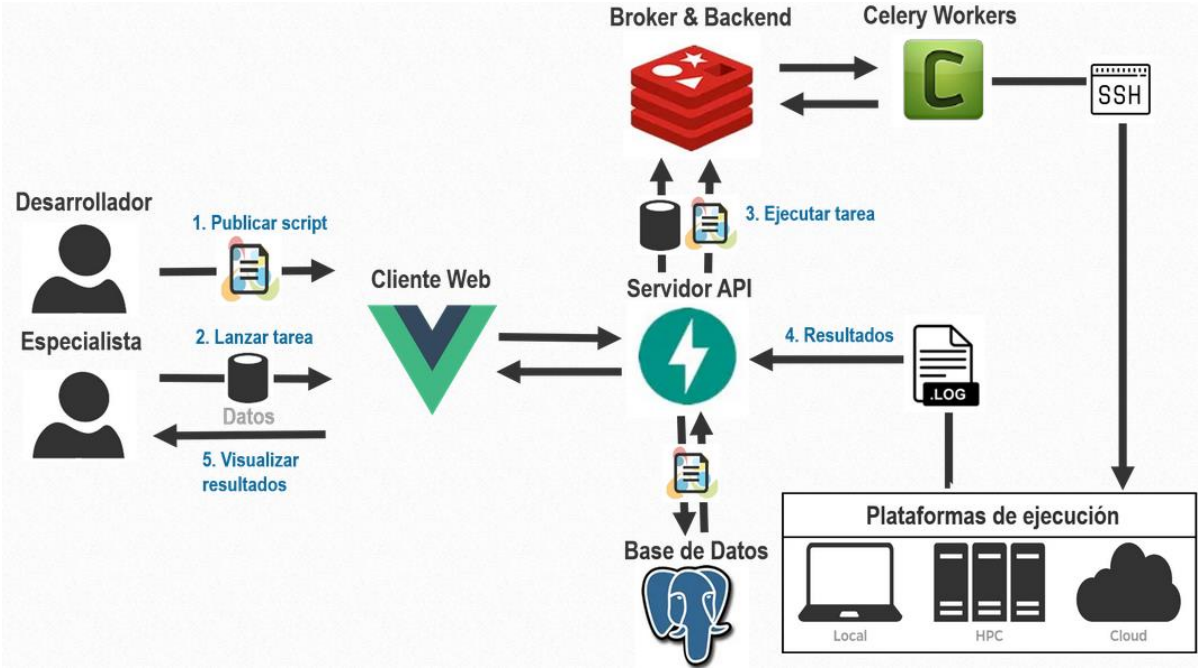

Fig. 1 – Esquema general del funcionamiento de la solución propuesta. Fuente: los autores.

Revista Cubana de Ciencias Informáticas
Vol. 17, No. 4, Octubre-Diciembre, 2023
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 43-59

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.

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. (Hernández, 2021)

Revista Cubana de Ciencias Informáticas
Vol. 17, No. 4, Octubre-Diciembre, 2023
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 43-59

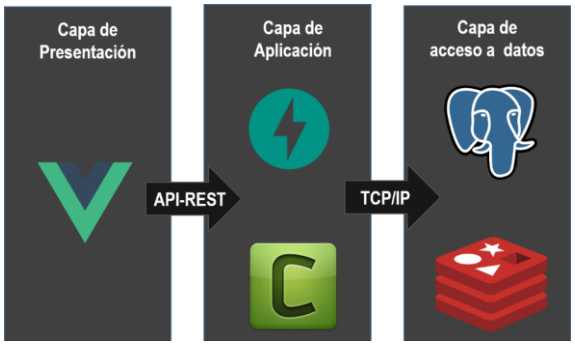

Fig. 2 – Arquitectura de la solución propuesta. Fuente: los autores.

En la Figura 2, 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.

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 3. 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.

Revista Cubana de Ciencias Informáticas
Vol. 17, No. 4, Octubre-Diciembre, 2023
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 43-59

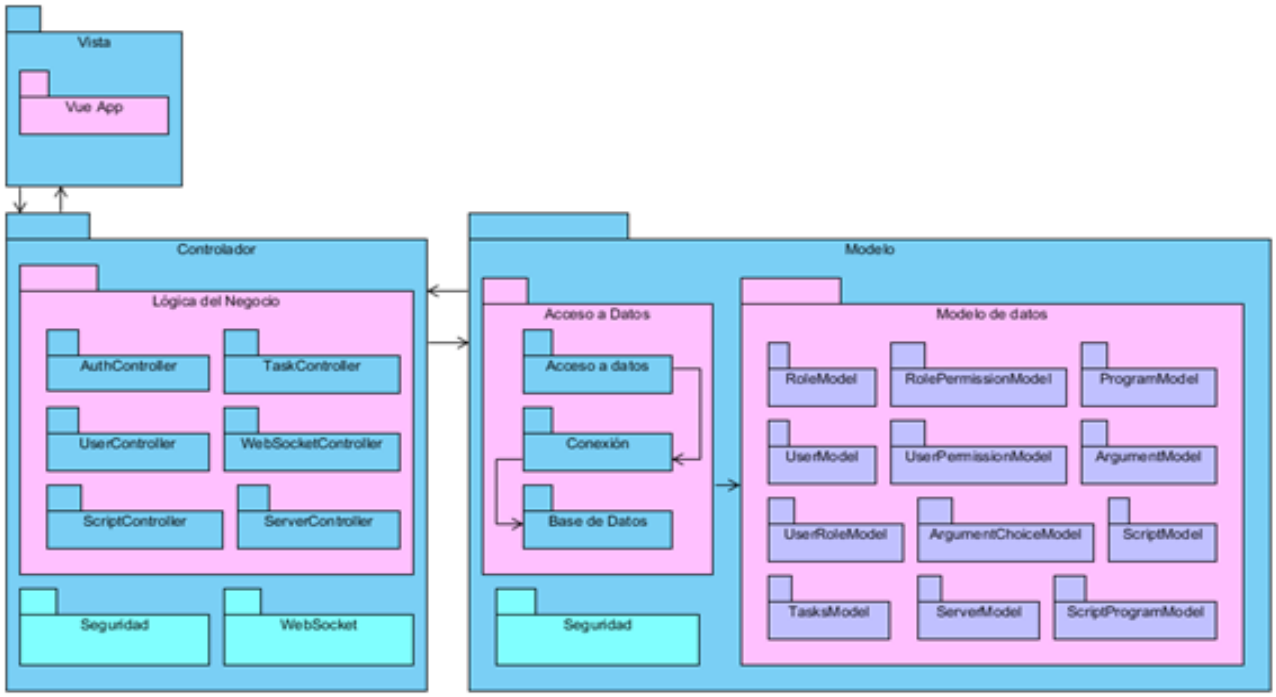

Fig. 3 – Modelo de la arquitectura del sistema (backend). Fuente: los autores.

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. (Enríquez, 2023)

- 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").

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. (Anchundia, 2022)

En la Figura 4 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").

Revista Cubana de Ciencias Informáticas
Vol. 17, No. 4, Octubre-Diciembre, 2023
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 43-59

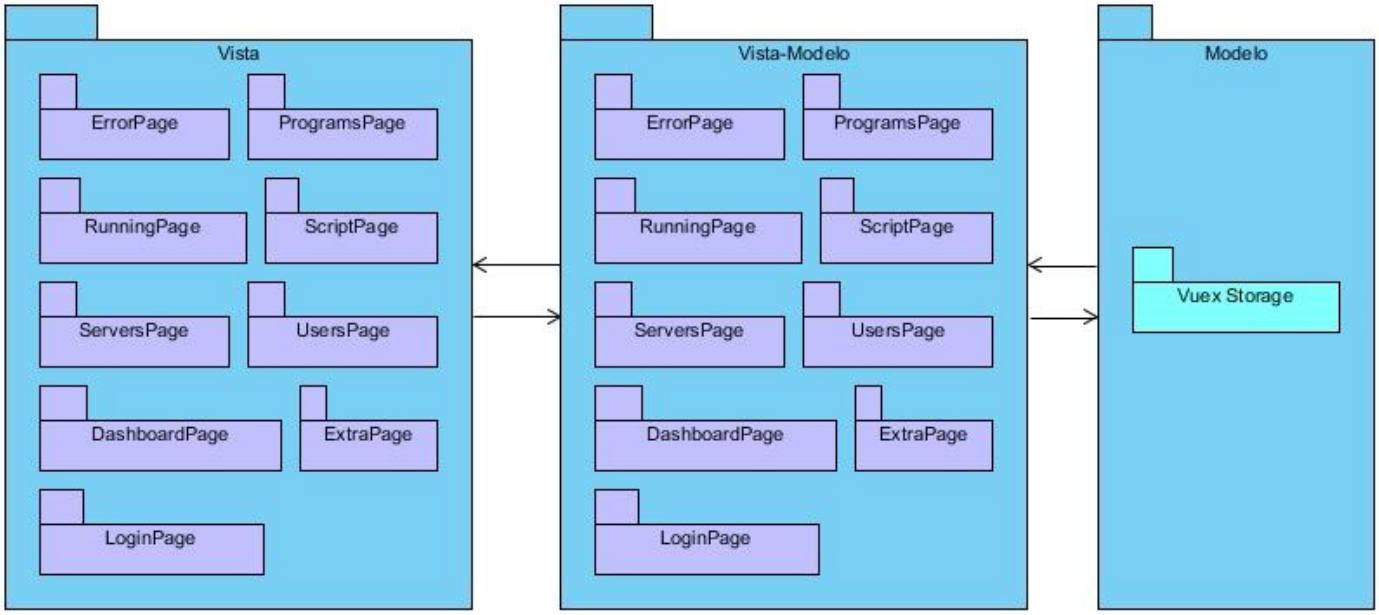

Fig. 4 – Modelo de la arquitectura del sistema (frontend). Fuente: los autores.

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. (García, 2022)

Revista Cubana de Ciencias Informáticas
Vol. 17, No. 4, Octubre-Diciembre, 2023
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 43-59

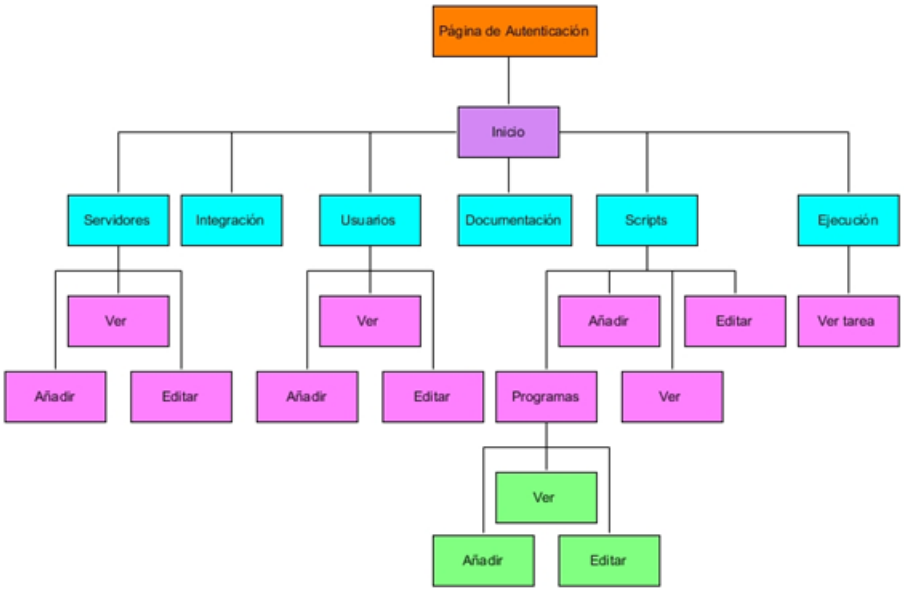

Fig. 5 – Mapa de Navegabilidad de la solución. Fuente: los autores.

Fase de Despliegue

Los diagramas de despliegue permiten modelar la disposición física o topología de un sistema; mostrando 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). En la Fig. 6, se muestra el diagrama de despliegue para la propuesta de solución.

Revista Cubana de Ciencias Informáticas
Vol. 17, No. 4, Octubre-Diciembre, 2023
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 43-59

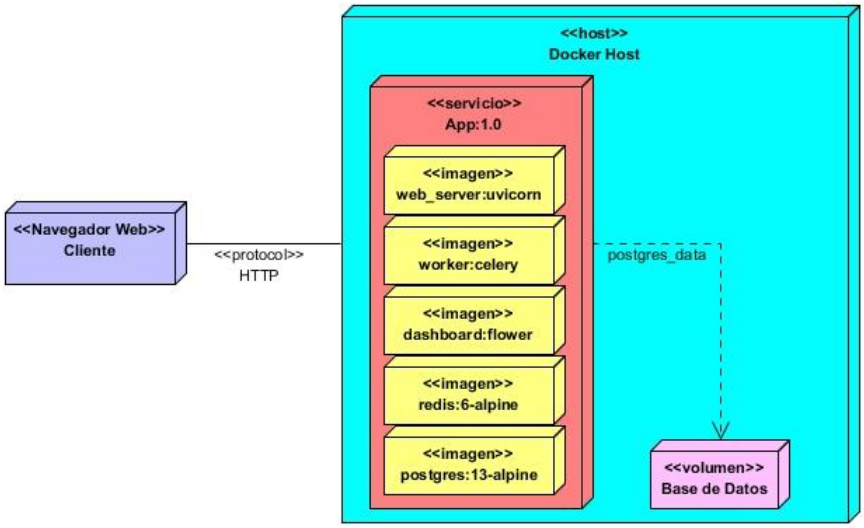

Fig. 6 – Diagrama de despliegue. Fuente: los autores.

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)

Se aplicaron pruebas por niveles como Unidad, Sistema y Aceptación (Jacobson, 2000). 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 y en todos los casos fue satisfactorio para el resultado. Entre las técnicas de pruebas que se realizaron para evaluar la funcionalidad de la herramienta, se ejecutaron pruebas funcionales, pruebas de seguridad y pruebas de rendimiento. 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 siendo positivas para la solución propuesta.

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.

Conclusiones

La definición de las herramientas, tecnologías y lenguajes, propiciaron diseñar un ambiente de desarrollo novedoso y ajustado al contexto tecnológico del clúster de neurociencias. 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, cumpliendo el objetivo general de la investigación. 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.

Referencias

Shepherd, G. M., Mirsky, J. S., Healy, M. D., Singer, M. S., Skoufos, E., Hines, M. S., & Miller, P. L. (1998). The Human Brain Project: neuroinformatics tools for integrating, searching and modeling multidisciplinary neuroscience data. Trends in neurosciences, 21(11), 460-468.

Redolfi, A., Archetti, D., De Francesco, S., Crema, C., Tagliavini, F., Lodi, R., ... & D'Angelo, E. (2023). Italian, European, and international neuroinformatics efforts: An overview. European Journal of Neuroscience, 57(12), 2017-2039.

Willett, F. R., Avansino, D. T., Hochberg, L. R., Henderson, J. M., & Shenoy, K. V. (2021). High-performance brain-to-text communication via handwriting. Nature, 593(7858), 249-254.

Torabzadehkashi, M., Rezaei, S., HeydariGorji, A., Bobarshad, H., Alves, V., & Bagherzadeh, N. (2019). Computational storage: an efficient and scalable platform for big data and hpc applications. Journal of Big Data, 6, 1-29.

Poline, J. B., Das, S., Glatard, T., Madjar, C., Dickie, E. W., Lecours, X., ... & Evans, A. C. (2023). Data and tools integration in the canadian open neuroscience platform. Scientific Data, 10(1), 189.

Sanz-Robinson, J., Jahanpour, A., Phillips, N., Glatard, T., & Poline, J. B. (2022, October). NeuroCI: Continuous Integration of Neuroimaging Results Across Software Pipelines and Datasets. In 2022 IEEE 18th International Conference on e-Science (e-Science) (pp. 105-116). IEEE.

González-Losada, C., González-Lodeiro, L. G., Canfux, A. I. B., Fernández, J. R., Camacho, H., Vazquez-Blomquist, D., & Nieto, G. E. G. (2022). Reliability of CNEURO hyssops for sample collection in the SARS-CoV-2 diagnosis. Revista Habanera de Ciencias Médicas, 21(1).

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

Hernández, L. M. A., Romero, V. A. P., González, S. A. S., & Rodríguez, J. A. V. (2021). Arquitectura REST para el desarrollo de aplicaciones web empresariales. Revista Electrónica sobre Tecnología, Educación y Sociedad, 8(15).

Enríquez, F., Fierro, S., Flores, B., Esparza, D. I., & Michelena, J. (2023). Impacto del patrón modelo vista controlador (MVC) en la seguridad, interoperabilidad y usabilidad de un sistema informático durante su ciclo de vida. EASI: Ingeniería y Ciencias Aplicadas en la Industria, 2(1), 11-16.

Anchundia Medrano, L. A. (2022). Análisis comparativo de tecnologías Front End Angular Js Vs React Js, en el modelo de procesos para el desarrollo de aplicaciones web (Bachelor's thesis, Babahoyo: UTB-FAFI. 2022).

García Cuenca, M. V. (2019). Construcción de mapas conceptuales navegables y comprensión lectora: análisis de procesos y diseño de instrucción.

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

Jacobson, I., Booch, G., & Rumbaugh, J. (2000). UML: el proceso unificado de desarrollo de software. Addison-Wesley.

Conflicto de interés

Los autores autorizan la distribución y uso de su artículo.

Contribuciones de los autores

1. Conceptualización: Arturo Orellana García.
2. Curación de datos: Michel Suárez Morales
3. Análisis formal: Michel Suárez Morales, Brian Pérez López
4. Adquisición de fondos: Arturo Orellana García
5. Investigación: Arturo Orellana García, Michel Suárez Morales, Brian Pérez López
6. Metodología: Arturo Orellana García
7. Administración del proyecto: Arturo Orellana García
8. Recursos: Arturo Orellana García, Michel Suárez Morales

9. Software: Michel Suárez Morales, Brian Pérez López
10. Supervisión: Arturo Orellana García
11. Validación: Arturo Orellana García, Michel Suárez Morales
12. Visualización: Michel Suárez Morales, Brian Pérez López
13. Redacción – Michel Suárez Morales, Brian Pérez López
14. Redacción – Arturo Orellana García, Michel Suárez Morales

Financiación

La investigación que da origen a los resultados presentados en la presente publicación recibió fondos de la Oficina de Gestión de Fondos y Proyectos Internacionales bajo el código PN305LH013-038.