



Universidad de las Ciencias Informáticas

Facultad 1

Trabajo de Diploma para optar por el título de
Ingeniero en Ciencias Informáticas.

Módulo de administración y gestión de colas y tareas de
procesamientos de datos de neurociencias para la
plataforma BrainSSys

Autor:

Dairon Canel Gómez
Dairon Fabián Sanchez Nuñez

Tutores:

Dr.C. Arturo Orellana García
MSc. Denys Buedo Hidalgo

La Habana, Noviembre de 2021

Declaración de autoría

Declaramos ser los únicos autores del trabajo de diploma “*Módulo de administración y gestión de colas y tareas de procesamientos de datos de neurociencias para la plataforma BrainSSys*”, 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 firmamos el presente documento a los ____ días del mes de _____ del año 20____.

Dairon Canel Gómez

Dairon F. Sánchez Núñez

Firma del diplomante

Firma del diplomante

Dr.C. Arturo Orellana García

MSc. Denys Buedo Hidalgo

Firma del tutor

Firma del tutor

Datos de contacto

DrC. Arturo Orellana García (aorellana@uci.cu): Se desempeña como líder del Grupo de Investigación de Minería de procesos y Asesor de Capacitación, Desarrollo e Investigación del Centro de Soluciones 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. Tutora varias tesis de grado, maestrías y doctorados enfocados al análisis de procesos de negocio, la informática médica y otras áreas del conocimiento. 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.

Dedicatoria

Resumen

Las neurociencias son un campo de la ciencia que estudia el sistema nervioso desde el funcionamiento neuronal hasta el comportamiento y cómo sus diferentes elementos interactúan. El Centro de Neurociencias de Cuba es la principal institución de investigación de neurociencia en el país y presenta un conjunto de problemas relacionados al manejo de los datos y a la seguridad. El procesamiento que se desarrolla sobre sus bases de datos se realiza de forma manual en sus servidores de HPC (*High Performance Computer*) debido a dificultades tecnológicas y financieras para acceder a los servicios que ofrecen las plataformas internacionales, por lo que los investigadores se ven obligados a programar instrucciones en consola para ejecutar tareas de procesamiento y análisis.

El objetivo de la presente investigación fue desarrollar un módulo que automatice el proceso de administración y gestión de colas y tareas de procesamiento de neurodatos para la plataforma BrainSSys. Se empleó Python como lenguaje de programación, PyQt5 como marco de trabajo y se utilizó el editor de código Sublime Text 3.

Se obtuvo como resultado un módulo que permite la gestión de las colas de procesamiento y la seguridad en la plataforma BrainSSys. Las pruebas realizadas determinaron la aceptación y la evaluación funcional del módulo, siendo en ambos casos de gran valor para la calidad de la propuesta solución.

Palabras clave: automatización, Neurociencias, neurodatos, procesamiento, seguridad

Abstract

Keywords:

Índice

Introducción	1
Capítulo 1. Fundamentación teórica de la investigación	6
1.1 Neurociencias y Neurotecnologías.....	6
1.1.1 Neurociencias.....	6
1.1.2 Neurotecnologías.....	7
1.1.3 Neuroinformática	8
1.2 Colaboración UCI-CNEURO	9
1.3 Plataformas para la gestión y análisis de datos de neurociencias.....	11
1.3.1 <i>The Canadian Brain Imaging Research Platform (CBRAIN)</i>	12
1.3.2 <i>Longitudinal Online Research and Imaging System (LORIS)</i>	13
1.3.3 <i>The Brain Image Library (BIL)</i>	13
1.3.4 <i>The C-BIG Repository</i>	14
1.3.5 <i>NeuroVault</i>	15
1.3.6 <i>NeuroMorpho</i>	17
1.3.7 Análisis comparativo de los sistemas descritos.....	17
1.4 Ambiente de desarrollo para la propuesta solución.....	18
1.5 Conclusiones parciales del capítulo	26
Capítulo 2. Propuesta de solución	27
2.1 Descripción de la propuesta solución.....	27
2.2 Requisitos del sistema	28
2.3 Definición de roles	31
2.4 Historias de usuarios	31
2.5 Análisis y diseño	34
2.5.1 Descripción de la arquitectura.....	34

2.5.2	Tarjetas CRC.....	35
2.6	Patrones de diseño.....	36
2.7	Modelo de datos.....	39
2.8	Conclusiones parciales del capítulo.....	40
Capítulo 3.	Implementación y pruebas.....	41
3.1	Estándares de codificación.....	41
3.2	Fase de desarrollo.....	42
3.3	Fase de prueba.....	45
3.4	Conclusiones parciales del capítulo.....	49
Conclusiones	51
Recomendaciones	52
Referencias bibliográficas	53
Anexos.....		58

Índice de ilustraciones

Figura 1. Estrella de Boehm y Turner. Adaptado de (Velázquez, Cabrera, Llano, & Hernández, 2012)	21
Figura 2. Fases del ciclo de desarrollo de la metodología XP (Pressman, 2010)	22
Figura 3. Modelo conceptual del sistema	27
Figura 4. Diagrama de arquitectura modelo-vista-controlador	35
Figura 6. Código de ejemplo del patrón Creador	37
Figura 7. Código de ejemplo del patrón Experto	38
Figura 8. Código de ejemplo del patrón Creador	38
Figura 5. Diagrama de entidad-relación	39
Figura 9. Código a aplicar pruebas de caja blanca	47
Figura 10. Representación del código en grafo de flujo	47
Figura 11. Casos de prueba por iteración. Fuente: elaboración propia	49

Índice de tablas

Tabla 1. Análisis Comparativo.....	17
Tabla 2. Comparación entre las metodologías ágiles SCRUM y XP (Orjuela & Rojas, 2008).....	21
Tabla 3. Requisitos funcionales	28
Tabla 4. Definición de roles.....	31
Tabla 6. HU_1 Autenticar.....	33
Tabla 7. HU_2 Administrar proceso	33
Tabla 8. HU_3 Administrar usuario	33
Tabla 9. Estimación del esfuerzo, plan de iteraciones y entrega por cada historia de usuario.....	34
Tabla 10. Tarjeta CRC asociada a la clase ProcessTable.....	36
Tabla 11. Tarjeta CRC asociada a la clase UserTable.....	36
Tabla 12. Tareas de ingeniería para la iteración 1.	43
Tabla 13. Tarea de ingeniería #1 Autenticar usuario.....	43
Tabla 14. Tarea de ingeniería #2 Crear proceso.....	43
Tabla 15. Tarea de ingeniería #3 Modificar proceso.	43
Tabla 16. Tarea de ingeniería #4 Eliminar proceso.....	43
Tabla 17. Tarea de ingeniería #5 Listar proceso.....	44
Tabla 18. Tareas de ingeniería para la iteración 2.	44
Tabla 19. Tarea de ingeniería #6 Crear usuario.....	44
Tabla 20. Tarea de ingeniería #7 Modificar usuario.....	44
Tabla 21. Tarea de ingeniería #8 Eliminar usuario.....	44
Tabla 22. Tarea de ingeniería #9 Listar usuario.....	44
Tabla 23. Tarea de ingeniería #10 Habilitar usuario.....	45
Tabla 24. Caso de prueba para la clase ProcessTable().....	47
Tabla 25. Prueba de aceptación #1 Autenticar usuario.....	48
Tabla 26. Prueba de aceptación #2 Administrar proceso	48
Tabla 27. Prueba de aceptación #3 Administrar usuario	49

Introducción

El análisis biológico del cerebro es un área multidisciplinar que abarca niveles de estudio, desde el puramente molecular hasta el específicamente conductual y cognitivo. Comprende el nivel celular (neuronas individuales), los ensambles y redes pequeñas de neuronas (como las columnas corticales) y los ensambles grandes (como los propios de la percepción visual) incluyendo sistemas como la corteza cerebral o el cerebelo, e incluso, el nivel más alto del sistema nervioso (Manes & Niro, 2014).

Las neurociencias son un campo de la ciencia que se encarga del estudio del sistema nervioso desde el funcionamiento neuronal hasta el comportamiento y de cómo sus diferentes elementos interactúan, dando lugar a las bases biológicas de la cognición y la conducta. El propósito principal de las Neurociencias es 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 (Kandell, Jessell, & Schwartz, 1997).

Un término importante a tener en cuenta en las Neurociencias es la Neuroinformática. Es utilizado en relación al campo de investigación de la neurociencia mediante la aplicación de modelos computacionales y herramientas analíticas. Tiene como objetivo facilitar de una manera estructurada y disciplinada que las nuevas tecnologías permitan a las diferentes áreas de investigación compartir datos y hallazgos para poder continuar con el estudio y entendimiento del cerebro. Estas áreas de investigación son importantes debido a la integración y análisis de grandes volúmenes de datos en detalle. Los neuroinformáticos cumplen la función de creación y desarrollo de herramientas computacionales, modelos matemáticos y bases de datos interoperables para su uso por científicos clínicos y de investigación (Adee & Sally, 2008).

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, un sistema web para el estudio de neuro datos y LORIS (Longitudinal Online Research and Imaging System) (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 (MCIN, 2021).

En Cuba también se desarrollan investigaciones con el uso de la Neuroinformática y las Neuro tecnologías, tal es el caso del Centro de Neurociencias de Cuba (CNEURO). Dicha institución está dedicada a la investigación de Neurociencias, Neurotecnología y otras tecnologías médicas, que abarca desde la investigación básica hasta el desarrollo, producción y su comercialización. La entidad lleva a cabo investigaciones de temas como neurociencia cognitiva, neuro informática y neuroimagen funcional. Entre sus áreas de especial interés se encuentran el desarrollo de nuevos métodos de neuroimagen, la búsqueda de nuevos biomarcadores y moléculas terapéuticas para la enfermedad de Alzheimer y el desarrollo de tecnologías basadas en neuro informática para el análisis de datos de electroencefalogramas (EEG) e imágenes de resonancia magnética (MRI) en condiciones cerebrales normales y patológicas (CNEURO, 2021).

CNEURO es la coordinadora del Programa Nacional de Ciencia y Tecnología de Neurociencia y Neurotecnología, por lo que trabaja en estrecha colaboración con distintas entidades para el desarrollo de investigaciones. Una de ellas es la Universidad de Ciencias Informáticas (UCI), la cual ejecuta un proyecto de investigación nacional denominado BrainSSys, para solucionar diferentes problemáticas asociadas a las tecnologías y la neuroinformática. 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.

En una entrevista realizada a 10 especialistas del centro con el objetivo de detectar los principales problemas que presenta la plataforma a la hora de controlar el procesamiento de los datos y gestionar los usuarios que la utilizan se detectaron 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 (*High Performance Computer*).
- Los investigadores se ven obligados a programar instrucciones en consola 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.

- Constituye un problema la creación de colas de procesamiento y la distribución de las tareas en los servidores debido a que no cuentan con mecanismos automatizados para realizar estas funciones, lo que constituye un proceso engorroso.
- Se pone en relieve un grupo de limitantes hacia el acceso a las bases de datos porque no se gestionan correctamente al ser directorios almacenados en carpetas, desorganizados y con baja seguridad; esto limita el éxito de las investigaciones, las cuales a menudo se desestiman por el sesgo que muestran en los resultados.

A partir de la situación problemática antes descrita se identifica el siguiente **problema a resolver**, ¿Cómo automatizar la gestión de las colas y tareas de procesamientos de las bases de datos del Centro Nacional de Neurociencias de Cuba?

El **objeto de estudio** se enmarca en el proceso de administración y gestión de colas y tareas de procesamiento de neurodatos.

El **campo de acción** se centra en la automatización del proceso de administración y gestión de colas y tareas de procesamiento de neurodatos.

Para dar solución al problema planteado se define como **objetivo general**:

Desarrollar un módulo de administración y gestión de colas y tareas de procesamientos de datos de neurociencias para la plataforma BrainSSys.

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 administración y gestión de colas y tareas en los sistemas informáticos existentes vinculados al campo de acción.
2. Análisis de plataformas internacionales para identificar las mejores alternativas de desarrollo y requisitos a tener en cuenta en la investigación.
3. Desarrollo de los componentes y funcionalidades para administrar los usuarios y los permisos, así como la gestión de colas y tareas.
4. Integración de los componentes desarrollados a la plataforma BrainSSys.
5. Validación de los resultados obtenidos a partir de los métodos definidos en la investigación.

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

- Métodos teóricos:

- Método analítico-sintético: en la presente investigación se utilizó este método para el análisis de la teoría y extracción de los principales conceptos a incluir en el marco teórico como: Sistema de Administración, Sistema de gestión de colas y tareas, plataforma digital para la estructuración y análisis de Bases de Datos estandarizadas de cerebro, entre otros y luego sintetizarlos en la propuesta de solución.
- Método histórico-lógico: se utiliza este método para investigar acerca de otras soluciones similares como: CBRAIN, LORIS, entre otros. Además, se analizó la evolución de los sistemas de análisis de neuro datos y cómo se han modernizado para lograr la informatización de la gestión del procesamiento de datos de Neurociencias.
- Método inductivo-deductivo: se utilizó en la aplicación de casos de pruebas al sistema, obteniendo conclusiones a partir de las respuestas proporcionadas por este.

-Métodos empíricos:

- Entrevista: posibilitó la recolección de información a través de conversaciones planificadas con especialistas de Neurociencias.
- Observación: se utilizó como método para validar la propuesta de solución.
- Modelación: se empleó como recurso auxiliar en la búsqueda teórica para caracterizar y representar mediante diagramas el campo de acción de la presente investigación.

Con el desarrollo del módulo de administración y gestión de colas y tareas de procesamientos de datos de neurociencias para la plataforma BrainSSys se esperan obtener los siguientes beneficios:

- La disminución del tiempo de procesamiento de los neuro datos, propiciando mejoras en el aprovechamiento de recursos.
- La posibilidad de administrar los permisos y roles de los usuarios de la plataforma BrainSSys, mejorando el orden y la prioridad de las peticiones de tareas de procesamiento de neuro datos.

La presente investigación está compuesta por tres capítulos, quedando estructurados de la siguiente manera:

Capítulo 1: Fundamentación teórica

Este capítulo contiene el marco conceptual en el que se muestran los principales elementos teóricos de la investigación. Se describen las diferentes herramientas y metodologías de desarrollo definidas para el desarrollo del sistema. Se hace referencia a los lenguajes de programación y tecnologías que apoyen

el desarrollo de la solución. Además, se realiza un análisis del proceso de administración y gestión de tareas y colas en Neurociencias.

Capítulo 2: Análisis y diseño

En este capítulo se exponen las etapas del proceso de desarrollo de software que permiten describir la propuesta de solución. Se desarrollan los artefactos ingenieriles que responden a dichas fases y se presenta el patrón arquitectónico utilizado.

Capítulo 3: Implementación y pruebas

Este capítulo hace énfasis en la implementación de las clases y subsistemas de la solución propuesta. Se presenta el modelo de datos y se describen los atributos comunes entre las entidades. Además, se realiza un estudio de los mecanismos para el tratamiento de errores y las pruebas pertinentes para la validación de dicha solución.

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; valorándose 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 plataformas de procesamiento y análisis de datos de neurociencias existentes a nivel internacional. De este modo, se puede realizar una correcta interpretación de la situación problemática y del problema a resolver.

1.1 Neurociencias y Neurotecnologías

Las neurociencias son 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 (Manes & Niro, 2014). Para la realización del estudio de esta ciencia se utilizan un conjunto de herramientas que sirven en el análisis y permiten intervenir en el sistema nervioso humano, las cuales son las llamadas neurotecnologías.

Según (Greenfield, 2002), "*Neurotecnología es toda información tecnológica y biotecnológica que afecta el cerebro.*"

1.1.1 Neurociencias

El propósito principal del estudio de la neurociencia es entender cómo el encéfalo produce la marcada individualidad de la acción humana. Aporta explicaciones de la conducta en términos de actividades del encéfalo, explicar cómo actúan millones de células nerviosas individuales para producir la conducta y cómo, a su vez, estas células están influidas por el medio ambiente. Los actuales estudiosos del cerebro, saben que para comprenderlo hay que derrumbar las barreras de las disciplinas tradicionales para mencionar apenas algunas de las áreas que han sido creadas, en gran parte para caracterizar los métodos de estudio (Rosell Aiquel, Juppet Ewing, Ramos Marquez, Ramírez Molina, & Barrientos Oradini, 2020).

Otro de los propósitos de su estudio es enfocarse en determinar los mediadores fisiológicos que subyacen a la relación entre factores sociales y salud, abre nuevas líneas de investigación, orienta sus objetivos hacia el proceso de salud-enfermedad. De ahí se desprende la necesidad de identificar la relación que guardan entre sí para obtener una visión más integral de dicho proceso (salud-enfermedad)

en su conjunto (Purves, Augustine, Fitzpatrick, Hall, & Lamantia, 2016) (Alcaraz & Gumá, 2001) (Martínez Alcántara, 2009).

En las neurociencias se distinguen varios niveles de análisis, que en orden ascendente de complejidad comprenden:

- La neurociencia molecular, que estudia la complejidad molecular del sistema nervioso y las diversas moléculas que lo componen.
- La neurociencia celular, que presta atención al estudio de cómo todas estas moléculas trabajan juntas, suministrando a las neuronas de propiedades especiales como los tipos diferentes de neuronas y sus funciones.
- La neurociencia de sistemas, que estudia los circuitos y redes neuronales complejos que llevan a cabo una función común, por ejemplo, visión, movimiento y lenguaje.
- La neurociencia cognitiva, que investiga los procesos neuronales que están a la base de los niveles superiores de la actividad mental humana, como son la conciencia de sí mismo, el lenguaje, la imaginación, la creatividad, el sentido estético, el comportamiento moral.

El estudio de las neurociencias proporciona una nueva oportunidad para comprender el comportamiento, la estructura y funciones de la mente, a través del estudio de la organización funcional del cerebro (García, 2020).

1.1.2 Neurotecnologías

Recientes desarrollos en las ciencias del cerebro permiten medir, registrar, alterar y/o manipular la actividad del cerebro. Esto se conoce de manera general como neuromodulación, que es la alteración de la actividad nerviosa por medio de la introducción de estímulos. Avances similares en ingeniería de sistemas y microcircuitos han visto nacer las neurotecnologías (Ausín, Morte, & Monasterio, 2020).

Las neurotecnologías son 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).

Algoritmos de Inteligencia Artificial aplicados en el campo de la neurociencia clínica pueden predecir y/o diagnosticar patologías neuropsiquiátricas, dispositivos para la estimulación cerebral profunda (ECP) que son utilizados para el tratamiento de múltiples enfermedades o las interfaces cerebro-ordenador (ICO) que registran la actividad del cerebro y a partir de sistemas computarizados trasladan esta información para controlar prótesis o sistemas robóticos periféricos (Müller & Rotter, 2017).

Otros ejemplos de neurotecnologías son las distintas técnicas de neuroimagen funcional y estructural, que comprende desde la Resonancia Magnética Funcional (fMRI), Tomografía por Emisión de Positrones (PET), hasta Electroencefalografía (EEG), estas permiten con distintos grados de resolución espacial y temporal medir y visualizar la actividad del cerebro. La estimulación magnética transcraneal (EMT), la estimulación de corriente directa transcraneal (ECDT) o la neuroretroalimentación son otros tipos de neurotecnologías neuromoduladoras con marcados propósitos terapéuticos (Ausín, Morte, & Monasterio, 2020).

En resumen, las neurotecnologías son un concepto que se aplica tanto a los procesos como a los dispositivos. Como tal, requiere una comprensión bastante completa de los sistemas biológicos tanto a nivel biomecánico como fisiológico (Ayers, Davis, & Alan, 2002). La información obtenida puede permitir dar soluciones en contextos clínicos o simplemente satisfacer el deseo de curiosidad de la ciencia básica (Ausín, Morte, & Monasterio, 2020). Además, el desarrollo tecnológico propicia la aparición de nuevos equipos de adquisición de imágenes y señales que abren el margen de oportunidades hacia investigaciones más específicas para el análisis del cerebro. La principal ventaja de estos estudios es que permiten obtener información del interior del cuerpo de un paciente, sin realizar un procedimiento quirúrgico (Calhoun & Adali, 2008) (Manchanda & Sharma, 2016).

1.1.3 Neuroinformática

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 (Linne, 2018).

El campo de la neuroinformática ha crecido rápidamente para involucrar el desarrollo de herramientas modernas basadas en la tecnología de la información y la comunicación (TIC) con el objetivo de ayudar a comprender la función cerebral en la salud y la enfermedad. Para lograr este objetivo, es esencial crear

grandes bases de datos donde sea posible compartir datos de neurociencia primaria, cuidadosamente estandarizados. Al igual que desarrollar herramientas de análisis poderosas y modelos computacionales (Linne, 2018).

El desarrollo de vanguardia en el campo de la neuroinformática ha otorgado avances significativos en estas últimas décadas, en estas se incluyen:

- Recopilación y mejor organización de datos de neurociencia (tanto en laboratorio húmedo como simulados) y desarrollo de bases de conocimiento.
- Desarrollo de flujos de trabajo para una mejor gestión y manejo de datos, metadatos y otros documentos relacionados con la investigación.
- Desarrollo de herramientas para la adquisición, análisis y visualización de datos automatizados y multipropósito.
- Desarrollo de herramientas y creación de repositorios para la distribución de datos.
- Desarrollo de herramientas para la teoría, cálculo y simulación de fenómenos neuronales.

1.2 Colaboración UCI-CNEURO

El Centro de Neurociencias de Cuba (CNEURO, 2021), 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 era investigar y producir tecnologías de última generación para el diagnóstico y tratamiento de las enfermedades cerebrales.

La entidad lleva a cabo investigaciones en una amplia gama de temas:

- 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

El centro cuenta con una variedad de servicios y capacidades:

- Estudios Neurofisiológicos.
- Estudios Neuroimágenes.
- Evaluaciones en modelos animales.
- Estudios de sueño.
- Evaluaciones auditivas integrales.
- Impresiones en 3D.

El centro es reconocido como un productor de dispositivos médicos avanzados en Cuba, enfocado en electroencefalografía, electromiografía, audiología y audífonos.

CNEURO es una institución de investigación que pertenece a varios consorcios de investigación internacionales como el Global Brain Consortium (GBC, 2021), la International Brain Initiative (IBI, 2021) 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, 2021), la Universidad de Maastricht (Maastricht, 2021), la Universidad de Aachen (RWTH-Aachen, 2021), la Universidad McGill (McGill, 2021) en Canadá, entre otros. También ha participado en proyectos internacionales financiados por el Human Frontier Science Program (HFSP, 2021) y algunos proyectos para la Organización Internacional de Investigación del Cerebro (IBRO, 2021). Según (Orellana, 2019), 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

Es una colaboración del Centro de Neurociencias de Cuba (CNEURO) y la Universidad de Ciencias Informáticas (UCI) con el objetivo de desarrollar una plataforma digital para la estructuración y análisis de bases de datos estandarizadas de cerebro. Para su desarrollo el equipo de proyecto se propone los siguientes **objetivos específicos**:

- Desarrollar una herramienta para la estructuración y manejo de bases de datos multimodales de neurociencias.
- Diseñar y desarrollar métodos para el procesamiento y análisis estandarizados de datos de neurociencias.
- Desarrollar una herramienta para la sincronización de bases de datos estandarizadas de cerebro con plataformas internacionales.
- Publicación y difusión de los resultados del proyecto y formación profesional de recursos humanos.

Se esperan obtener durante la implementación dos sistemas fundamentales.

Data Brain, sistema encargado de la estructuración de Bases de datos que tiene las siguientes **actividades principales**:

- Desarrollar una herramienta para la estructuración de bases de datos multimodales de cerebro.
- Desarrollar algoritmos para la conversión de datos de cerebro al estándar BIDS.
- Desarrollar herramientas para el manejo, control y seguimiento de las bases de datos de cerebro.

Pixel Brain, sistema para el análisis de imágenes y señales, este posee las siguientes **actividades principales**:

- Desarrollar algoritmos para el análisis de neuroimágenes multimodales basado en distintas Transformadas
- Desarrollar algoritmos para la identificación y estructuras metabólicas y funcionales en imágenes termográficas de cerebro aplicando Deep Learning.
- Desarrollar métodos para la integración de metadatos y datos anatómicos/fisiológicos en neuroimágenes.
- Desarrollar algoritmos para el tratamiento y limpieza de datos en bases de datos de cerebro.
- Desarrollar algoritmos para la clasificación de los datos según sus características.
- Desarrollar algoritmos para la comprensión de bases de datos sin pérdida de información.

1.3 Plataformas para la gestión y análisis de datos de neurociencias

En este epígrafe se realiza un estudio de las herramientas homólogas que permiten los procesos de administración y gestión de colas y tareas de procesamiento de neurodatos. Este análisis será útil a la hora de hacer la selección de métodos para la propuesta solución.

1.3.1 The Canadian Brain Imaging Research Platform (CBRAIN)

Según (Sherif, y otros, 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 autenticados, 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 HPCs.

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.

1.3.2 Longitudinal Online Research and Imaging System (LORIS)

Según (Das, Zijdenbos, Harlap, Vins, & Evans, 2012), *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.

1.3.3 The Brain Image Library (BIL)

La Biblioteca de imágenes cerebrales (BIL, 2021) 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.

1.3.4 The C-BIG Repository

El repositorio C-BIG (C-BIG, 2021) 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 través 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.

1.3.5 NeuroVault

NeuroVault (NeuroVault, 2021) 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, Varoquaux, Rivera, & Schwarz, 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 RESTful 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, Varoquaux, Rivera, & Schwarz, 2015).

1.3.6 NeuroMorpho

NeuroMorpho (NeuroMorpho, 2021) 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.

1.3.7 Análisis comparativo de los sistemas descritos

Tabla 1. Análisis Comparativo

	CBRAIN	LORIS	BIL	C-Big	Neuro-Vault	Neuro-Morpho
Acceso al repositorio	SSL, SSH	SSL	SSH, Bridges-2, Neocórtex, Sistema VM	SSH	SSH	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	No especificado	Linux, Apache, MySQL, PHP (LAMP)	No especificado.	LORIS, MySQL, React, JavaScript	Python, Django, Docker, JavaScript	MySQL, Java, C ++, Matlab, Apache Tomcat
Código abierto	No	No	No	No	Sí	No
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 / 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 el desarrollo de un nuevo componente, tomando en cuenta un grupo de características deseadas que presentan las plataformas anteriores en aspectos como tecnologías a emplear, acceso a la información, entre otros. Pero la realización de este proceso posee una alta complejidad técnica por lo que obliga a los investigadores a dominar técnicas de programación para dictar instrucciones para ejecutar los procesamientos.

1.4 Ambiente de desarrollo para la propuesta solución

Debido a que la propuesta solución está orientada a varios sistemas operativos, a continuación, se describen herramientas, metodologías y tecnologías usadas para el desarrollo de la aplicación.

Metodología de desarrollo

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

Para fundamentar la selección de la metodología se aplica el modelo propuesto por Barry Boehm y Richard Turner, conocido por el método de Boehm y Turner que caracteriza el proyecto de software y estima cuan ágil o prescriptivo debería ser el enfoque a utilizar a partir de cinco criterios, estos son: tamaño del equipo, criticidad del producto, dinamismo de los cambios, cultura del equipo y personal con que se cuenta. Cada uno de esos criterios tiene elementos que lo discriminan y por tanto se tienen en cuenta a la hora de seleccionar uno u otro enfoque (Velázquez, Cabrera, Llano, & Hernández, 2012).

El enfoque prescriptivo, denominado en algunas bibliografías como tradicional o pesado, busca la estructura, orden y consistencia del proyecto de desarrollo de software en cuestión (Velázquez, Cabrera, Llano, & Hernández, 2012).

El enfoque ágil, llamado también como enfoque ligero se centra en los miembros del equipo y su interacción, en la entrega rápida de versiones de software funcional, en la colaboración constante del cliente y la facilidad para manejar los cambios, dándole menor importancia a las herramientas, documentación, formalidad y planificación exhaustiva del proceso (Velázquez, Cabrera, Llano, & Hernández, 2012).

A continuación, se muestra una evaluación de las características del proyecto para determinar la idoneidad de un enfoque ágil o tradicional a partir de los siguientes criterios (Velázquez, Cabrera, Llano, & Hernández, 2012):

- **Tamaño:** se utiliza para representar el número de personas involucradas en el proyecto.

El proyecto está compuesto por cuatro integrantes, cuenta con dos estudiantes de ingeniería en ciencias informáticas, un doctor en ciencias y un máster en ciencias, para la implementación de varios requisitos funcionales que presentan cierto nivel de complejidad.

- **Criticidad:** se utiliza para evaluar la naturaleza del daño ocasionado por defectos que no hayan sido detectados al producto.

El equipo de desarrollo tiene una elevada responsabilidad con la calidad del producto a obtener, debido a que este módulo controlará la lista y las colas de procesamiento además de la lista de usuarios para la gestión de la seguridad del sistema BrainSSys, pero los errores que presente el mismo no provocará pérdidas de vidas humanas, ni un fuerte impacto social y monetario para la institución.

- **Dinamismo:** representa la rapidez con la que pueden estar cambiando los requerimientos del proyecto.

El constante cambio en los requisitos es un riesgo que se asumirá durante todo el ciclo de desarrollo del software. Para mitigarlo, deben adoptarse mecanismos que faciliten la asimilación y adaptación rápida a dichos cambios. El valor se tomó teniendo en cuenta la cantidad de funcionalidades que no pudieran cambiar a lo largo del desarrollo.

- **Personal:** representa la proporción del personal con experiencia alta, media y baja.

El equipo del proyecto cuenta con un doctor y un máster en ciencias altamente calificados para su desarrollo, también con dos estudiantes de ingeniería en ciencias informáticas que presentan conocimientos adquiridos durante la carrera sobre el tema propuesto. Determinando que el equipo de desarrollo contiene en su mayoría personal experimentado para la solución del proyecto.

- **Cultura:** las organizaciones y las personas que relaciona el proyecto pueden depender de la confianza o de la relación contractual.

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

Los elementos antes descritos evidencian organización en el desarrollo del trabajo, pero al ser un equipo pequeño no necesitan relación contractual dada la buena comunicación y confianza entre sus miembros. Cada especialista en el desempeño de su rol será libre de incorporar ideas que no afecten el diseño del sistema, ni los compromisos planificados.

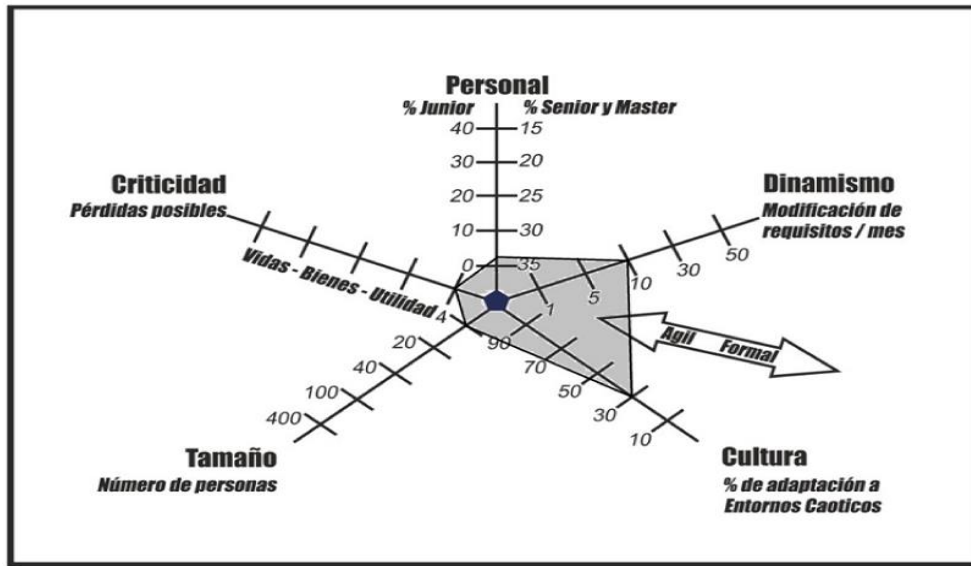


Figura 1. Estrella de Boehm y Turner. Adaptado de (Velázquez, Cabrera, Llano, & Hernández, 2012).

Metodología ágil

La metodología ágil, es una metodología de gestión de proyectos que están especialmente orientadas para proyectos que necesitan de una solución con una elevada simplificación sin dejar de lado el aseguramiento de la calidad del producto. Se centran en el factor humano y el producto, es decir, ellas le dan mayor valor al individuo, a la colaboración del cliente y al desarrollo incremental del software con iteraciones muy cortas. Utiliza ciclos de desarrollo cortos para centrarse en la mejora continua del desarrollo de un producto o servicio (Orjuela & Rojas, 2008).

Entre las metodologías ágiles más utilizadas a nivel mundial se encuentran SCRUM y XP (Orjuela & Rojas, 2008):

Tabla 2. Comparación entre las metodologías ágiles SCRUM y XP (Orjuela & Rojas, 2008)

Características	SCRUM	XP
Más enfocado en los procesos		
Más enfocado en las personas	X	X
Resultados rápidos	X	X
Cliente activo	X	X
Manejo del tiempo	X	X
Refactorización del código		X
Iterativo	X	X
Respuesta a los cambios	X	X

Por las características que se evidencian en la tabla anterior, XP cumple con siete de las ocho características y SCRUM con seis, por lo que se considera que la metodología adecuada para seguir en el proceso de desarrollo de la aplicación es XP.

Extreme Programming (XP)

La programación extrema o *Extreme Programming*, es una disciplina de desarrollo de software basada en los métodos ágiles, que evidencia principios tales como el desarrollo incremental, la participación activa del cliente, el interés en las personas y no en los procesos como elemento principal, y aceptar el cambio y la simplicidad (Pressman, 2010).

XP propone una serie de fases que al ser concluidas dan origen a una versión del producto, y cada versión es un ciclo, el cual hace parte del ciclo de vida del software. Al no tener más ciclos a ejecutar se entiende que los sistemas han cumplido con su objetivo, en caso contrario se deben seguir desarrollando ciclos para agregar la funcionalidad deseada. Cada fase del ciclo comprende lo siguiente (Pressman, 2010):



Figura 2. Fases del ciclo de desarrollo de la metodología XP (Pressman, 2010).

Fase de planeación: esta fase inicia con las historias de usuario que describen las características y funcionalidades del software. El cliente asigna un valor o prioridad a la historia, los desarrolladores evalúan cada historia y le asignan un costo el cual se mide en semanas de desarrollo (Pressman, 2010).

Fase de diseño: el proceso de diseño debe procurar diseños simples y sencillos para facilitar el desarrollo. Este proceso se apoya en el uso de tarjetas CRC (Colaborador-Responsabilidad-Clase) la

cual identifica las clases orientadas a objetos que son relevantes para el incremento del software (Pressman, 2010).

Fase de codificación: en esta fase los desarrolladores deben diseñar las pruebas de unidad que ejercitarán cada historia de usuario (Pressman, 2010).

Fase de pruebas: las pruebas de unidad deben implementarse con un marco de trabajo que permita automatizarlas, con la finalidad de realizar pruebas de integración y validación diarias, esto proporcionará al equipo un indicador del progreso y revelarán a tiempo si existe alguna falla en el sistema (Pressman, 2010).

Lenguaje y herramienta para el modelado de la solución

Las herramientas de Ingeniería de Software Asistida por Computadora (CASE, del inglés *Computer Aided Software Engineering*) se definen, según el autor Ian Sommerville, como “el software que se utiliza para ayudar a las actividades del proceso del software como la ingeniería de requerimientos, el diseño, el desarrollo de programas y las pruebas. Las herramientas CASE incluyen editores de diseño, diccionarios de datos, compiladores, depuradores. De las diversas herramientas CASE, como es el caso de: Erwin, EasyCASE, Oracle Designer y Visual Paradigm, se selecciona esta última para el desarrollo de la presente investigación, por ser de las más utilizadas para el desarrollo de sistemas en los proyectos productivos de la UCI.

Lenguaje de Modelado UML 2.0

“El modelado constituye una simplificación de la realidad donde se define lo esencial para la construcción del software con los objetivos de comunicar la estructura de un sistema complejo, especificar el comportamiento deseado del sistema, comprender mejor lo que se está desarrollando y descubrir oportunidades de simplificación y reutilización” (Jacobson, Booch, & Rumbaugh, 2000).

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 (Jacobson, Booch, & Rumbaugh, 2000).

Herramienta CASE Visual Paradigm

Se empleará Visual Paradigm en su versión 8.0, para modelar los diagramas que se generen durante el proceso de desarrollo de software.

Visual Paradigm para UML es una herramienta de modelado que soporta el modelado mediante UML y proporciona asistencia tanto a los analistas, ingenieros de software como a los desarrolladores durante todo el ciclo de vida del proyecto (SCRIBD, 2018).

La herramienta de modelado Visual Paradigm (en su versión libre) para UML en su versión 8.0 soporta el lenguaje de modelado UML, facilita el trabajo del equipo de desarrollo durante el ciclo de vida del software y permite la estandarización de la documentación que se va generando.

Gestor de Bases de Datos SQLite

Se selecciona SQLite en su versión 3.8 debido a que es una de las mejores alternativas para la persistencia de los datos sobre texto plano o ficheros en XML ya que es un sistema gestor muy ligero que posibilita la portabilidad local de los datos y no requiere de un gran procesamiento en memoria para su gestión. Android posee herramientas muy útiles que facilitan las tareas de manipulación y consulta sobre este tipo de base de datos.

SQLite es un sistema de gestión de bases de datos que enlaza con el programa pasando a ser parte integral del mismo. El programa utiliza la funcionalidad de SQLite a través de llamadas simples a subrutinas y funciones. Esto reduce la latencia en el acceso a la base de datos, debido a que las llamadas a funciones son más rápidas y seguras que la comunicación entre procesos (SQLite, 2018).

Lenguaje y herramienta para la implementación de la solución

Para el desarrollo de la solución se definieron los lenguajes y herramientas que se describen a continuación teniendo en cuenta: el conocimiento del investigador, las características de las propuestas de la solución y la metodología de desarrollo de software.

Lenguaje de programación

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 lenguajes como C# o 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. C++ es un lenguaje de programación híbrido centrado en la orientación a objetos. Por ser un lenguaje simplificado, rápido, elegante, flexible, ordenado, limpio y portable, se eligió para la aplicación el lenguaje de programación Python.

Python

Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en la legibilidad de su código. Se trata de un lenguaje de programación multiparadigma, ya que soporta parcialmente la orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, dinámico y multiplataforma (Python, 2021).

Entorno integrado de desarrollo

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

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 cuadrículas (Sublime Text, 2021).

Marco de trabajo

Un marco de trabajo o framework es un diseño abstracto orientado a objetos para un determinado tipo de aplicación, es un patrón arquitectónico que proporciona una plantilla extensible para un tipo específico de aplicaciones. Según un framework o "esquema" es un subsistema expandible de un conjunto de

servicios, es un conjunto cohesivo de interfaces y clases que colaboran para proporcionar los servicios de la parte central e invariable de un subsistema lógico.

Qt

Es un framework multiplataforma orientado a objetos ampliamente usado para desarrollar programas que utilicen interfaz gráfica de usuario, así como diferentes tipos de herramientas para la línea de comandos y consolas para servidores q no necesitan interfaz gráfica. Qt es desarrollada como un software libre y de código abierto a través de Qt Project dónde participa tanto la comunidad como desarrolladores de Nokia, Digia y otras empresas. Debido a la sencillez, robustez, rendimiento nativo, compatibilidad multiplataforma y ambas licencias, de código abierto y comerciales, muchas organizaciones en muchas partes del mundo utilizan Qt (Qt, 2021).

PyQt 5

Es una colección de interfaces de una serie de módulos y clases que unen python2.x y python3.x al marco de la aplicación Qt. PyQt puede ejecutarse en todas las plataformas compatibles con Qt como Windows OS X, IOS y Android. Los programas de GUI multiplataformas escritos por PyQt son portátiles. La documentación de PyQt en comparación a otras bibliotecas de GUI de Python es la más basta (PyQt, 2021).

1.5 Conclusiones parciales del capítulo

Se sistematizó sobre los fundamentos de teoría-investigación que permitieron constatar que existe la necesidad en Cuba de poseer esta aplicación por las limitantes que existen hoy para el procesamiento de colas en CNEURO. El análisis del estado del arte permitió obtener información valiosa sobre las características necesarias para obtener una solución actual y en correspondencia con la existente en el mercado. Además, al determinar cómo metodología de desarrollo del software el XP, se logró comprender los procesos que posee y la importancia de éstos en la realización de proyectos de gran alcance como es BrainSSys.

2.2 Requisitos del sistema

La obtención de requisitos es uno de los pasos fundamentales en el desarrollo de software. Esta tarea está encaminada a identificar los requerimientos del cliente, analizar las necesidades y especificar los requisitos de la propuesta de solución. En la presente investigación se identificaron los requisitos funcionales y no funcionales de la solución a partir de entrevistas con el cliente.

Requisitos funcionales

Los requisitos funcionales definen las acciones que debe realizar el sistema. Son capacidades o condiciones que el sistema debe cumplir, cómo debe comportarse en situaciones específicas. En algunos casos también pueden plantear explícitamente qué no debe hacer el sistema (Pressman, 2010).

Para concretar las funcionalidades a desarrollar se obtienen los siguientes requisitos funcionales:

Tabla 3. Requisitos funcionales

No	Nombre	Descripción	Prioridad	Complejidad
1	RF1. Listar procesos	La aplicación debe permitir mostrar un listado con todos los procesos	Alta	Media
2	RF2. Eliminar proceso	La aplicación debe permitir eliminar los procesos añadidos al sistema.	Media	Alta
3	RF3. Modificar proceso	La aplicación debe permitir modificar los datos de los procesos añadidos al sistema.	Media	Media

4	RF4. Añadir proceso	La aplicación debe permitir adicionar procesos a la lista de procesos.	Media	Baja
5	RF5. Adicionar usuario	La aplicación le debe permitir al administrador adicionar usuarios a la lista de usuarios.	Media	Baja
6	RF6. Listar usuarios	La aplicación debe permitir mostrar un listado con todos los usuarios.	Alta	Baja
7	RF7. Modificar usuario	La aplicación le debe permitir al administrador modificar los usuarios de la lista.	Media	Media
8	RF8. Eliminar usuario	La aplicación le debe permitir al administrador eliminar los usuarios de la lista.	Media	Alta
9	RF9. Habilitar usuario	Para mantener la seguridad de los usuarios, la aplicación debe permitir al administrador habilitar y/o deshabilitar los usuarios.	Media	Baja
10	RF10. Autenticar usuario	Para la seguridad del manejo de los datos, la aplicación debe	Alta	Media

presentar un proceso de
autenticación.

Requisitos no funcionales

Requisitos No Funcionales, son requisitos que asignan restricciones para el diseño e implementación de un sistema informático o estándares de calidad que debe cumplir el mismo. En resumen, son cualidades o propiedades con las que debe contar nuestro producto (Sommerville, 2011).

Lista de Requisitos no funcionales

RNF1- Usabilidad

- Deberá presentar botones organizados por la funcionalidad, de tal manera que permita al usuario una interacción consistente con el mismo.
- Se utilizará el idioma español para los mensajes y textos de la interfaz.
- Proveer a los formularios nombres que muestre de forma clara la función que realizan.
- Utilizar un vocabulario común para todos los textos.
- La aplicación podrá ser ejecutada desde diferentes plataformas de desarrollo

RNF2- Apariencia o Interfaz Externa

- La navegación dentro del módulo debe permitir siempre la contemplación de las funciones principales.

RNF3-Restricciones de diseño

- Mantener un sistema de codificación estándar siguiendo las pautas establecidas.

RNF4- Software

- El sistema debe ser ejecutado en cualquier plataforma.
- Google Chrome, Navegadores web Mozilla, preferiblemente Firefox en su versión superior a la 24.

RNF4-Hardware

- PC de 2.10gb de RAM

2.3 Definición de roles

Los actores del negocio son aquellas personas que interactúan con el negocio para beneficiarse de sus resultados. Ellos se agrupan en diferentes roles los cuáles son descritos a continuación:

- **Administrador:** Tiene todos los privilegios del módulo.
- **Investigador:** Visualiza la lista de procesos. Crea procesos mediante su interacción con la plataforma.

En la tabla 3 se define el rol que desempeña cada uno de los actores del negocio.

Tabla 4. Definición de roles

Actores que intervienen en el negocio	Rol que desempeñan en el módulo
Investigador	Investigador
Programador	Administrador
Jefe de Centro	Administrador
Jefe de Proyecto	Investigador

2.4 Historias de usuarios

Las Historias de Usuario (HU) es un artefacto generado por metodologías ágiles como XP, que sustituyen a los documentos de especificación funcional, y a los casos de uso. Son utilizadas con el fin de especificar los requisitos del software. Estas son escritas por el cliente, en su propio lenguaje, como descripciones cortas de lo que el sistema debe realizar. A través de un conjunto de tablas se describen brevemente las características deseadas. Contienen la información suficiente para que los desarrolladores puedan producir una estimación razonable del esfuerzo necesario para su implementación (Joskowicz, 2008).

La prioridad en el negocio:

- **Alta:** se le otorga a las HU que resultan funcionalidades fundamentales en el desarrollo del sistema, a las que el cliente define como principales para el control integral del sistema.
- **Media:** se le otorga a las HU que resultan para el cliente como funcionalidades a tener en cuenta, sin que estas tengan una afectación sobre el sistema que se esté desarrollando.

- Baja: se le otorga a las HU que constituyen funcionalidades que sirven de ayuda al control de elementos asociados al equipo de desarrollo, a la estructura y no tienen nada que ver con el sistema en desarrollo.

El riesgo en su desarrollo:

- Alta: cuando en la implementación de las HU se considera la posible existencia de errores que conlleven a la inoperatividad del código.
- Media: cuando pueden aparecer errores en la implementación de la HU que puedan retrasar la entrega de la versión.
- Baja: cuando pueden aparecer errores que serán tratados con relativa facilidad sin que traigan perjuicios para el desarrollo del proyecto.

Las HU son representadas mediante tablas divididas por las siguientes secciones:

- Número: esta sección representa el número, incremental en el tiempo, de la HU que se describe.
- Nombre de HU: identifica la HU que se describe entre los desarrolladores y el cliente.
- Modificación de HU número: sección que representa si la HU se le realizó alguna modificación con respecto al estado anterior.
- Usuario: los programadores responsables de la HU.
- Iteración asignada: número de la iteración donde va a desarrollarse la HU.
- Prioridad en negocio: se le otorga una prioridad (Alta, Media, Baja) a las HU de acuerdo a la necesidad de desarrollo.
- Riesgo en desarrollo: se le otorga una medida de (Alto, Medio, Bajo), a la ocurrencia de errores en el proceso de desarrollo de la HU.
- Puntos estimados: es el tiempo estimado en semanas que se demorará el desarrollo de la HU.
- Puntos reales: representa el tiempo que se demoró en realidad el desarrollo de la HU.
- Descripción: breve descripción de la HU.
- Observaciones: señalamiento o advertencia del sistema.

Las HU también son utilizadas para estimar el tiempo que el equipo de desarrollo tomará para realizar las entregas, deben poder ser programadas en un tiempo entre una y tres semanas. En una entrega se puede desarrollar una o más HU, esto depende solo del tiempo que demore la implementación de cada una de las mismas. Si la estimación es superior a tres semanas, debe ser dividida en dos o más historias. Si es menos de una semana, se debe combinar con otra historia.

Tabla 5. HU_1 Autenticar

Historia de usuario	
Numero: HU 1	Nombre Historia de usuario: Autenticar usuario
Usuario: Dairon Fabian Sánchez Núñez	Iteración asignada: 1
Prioridad en negocio: Alta	Puntos estimados: 2
Riesgo en desarrollo: Media	Puntos reales: 2
Descripción: Permite el acceso del usuario al sistema.	
Observaciones: El usuario tiene que llenar los campos: nombre de usuario y contraseña, de encontrar algún error, el sistema muestra una ventana emergente con la descripción del error.	

Tabla 6. HU_2 Administrar proceso

Historia de usuario	
Numero: HU 2	Nombre Historia de usuario: Administrar proceso
Usuario: Dairon Canel Gómez	Iteración asignada: 1
Prioridad en negocio: Alta	Puntos estimados: 2
Riesgo en desarrollo: Media	Puntos reales: 2
Descripción: Lista y permite al usuario crear, modificar y eliminar los procesos.	
Observaciones: Este recoge los todos los datos de cada proceso.	

Tabla 7. HU_3 Administrar usuario

Historia de usuario	
Numero: HU 3	Nombre Historia de usuario: Administrar usuario
Usuario: Dairon Canel Gómez	Iteración asignada: 1
Prioridad en negocio: Alta	Puntos estimados: 2
Riesgo en desarrollo: Media	Puntos reales: 2
Descripción: Lista y permite al administrador crear, modificar, eliminar y habilitar o deshabilitar los usuarios.	
Observaciones: Este recoge todos los datos de cada usuario.	

Estimación del esfuerzo, plan de iteraciones y entrega

La medida utilizada para la estimación del esfuerzo asociado a la implementación es el punto. Un punto equivale a una semana ideal de programación, y esta semana equivale a 5 días laborables. Esta medida generalmente toma valores de 1 a 3 puntos.

Para cada entrega fueron escogidas las HU, teniendo en cuenta el orden definido. Este plan define las historias de usuario que deben ser implementadas en cada iteración y las fechas de liberación.

En el momento en que culmina la elaboración de las HU, se inicia el proceso de creación de un plan de entrega. El cual tiene como objetivo fundamental la obtención por parte de los programadores de una estimación detallada del período de tiempo que deben tener en cuenta para la implementación.

A continuación, se muestra una tabla que define la estimación del esfuerzo por cada HU, las iteraciones en las que deben ser realizadas, y la fecha de entrega de estas:

- I: iteraciones en las que deben ser implementadas cada HU.
- A: identificador por HU.
- PE: puntos de estimación por cada HU.
- DT: duración total en semanas por cada iteración.
- E: entrega de cada HU por cada iteración.
 - FI: fecha de inicio.
 - FF: fecha final.

Tabla 8. Estimación del esfuerzo, plan de iteraciones y entrega por cada historia de usuario

I	A	HU	PE	DT	E
1	HU1	Autenticar	1	3	FI: 1 de noviembre
	HU2	Administrar proceso	2		FF: 22 de noviembre
2	HU3	Administrar usuario	2	2	FI: 22 de noviembre FF: 6 de diciembre

2.5 Análisis y diseño

El papel del diseño en el ciclo de vida de un software es facilitar la comprensión de su funcionamiento y proveer una representación o modelo del mismo con el propósito de definirlo con los suficientes detalles como para permitir su realización física. Estimula el uso de las tarjetas CRC (Clase-Responsabilidad-Colaboración) como un mecanismo eficaz para pensar en el software en un contexto orientado a objetos. Las tarjetas CRC, son el único producto del trabajo de diseño que se genera como parte del proceso XP. El objetivo es disminuir el riesgo cuando comience la implementación verdadera y validar las estimaciones originales para la historia que contiene el problema de diseño (Pressman, 2010).

2.5.1 Descripción de la arquitectura

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. La arquitectura empleada

para el desarrollo del sistema es Modelo-Vista-Controlador. Esta arquitectura divide la presentación e interacción de los datos del sistema. El sistema se estructura en tres componentes lógicos que interactúan entre sí (Sommerville, 2011):

Modelo: Maneja lo referente a la persistencia de datos de la aplicación, las clases entidades, el acceso a la red y los elementos necesarios para manejar dichos elementos. En pocas palabras contiene únicamente los datos puros de aplicación; no contiene lógica que describe cómo pueden presentarse los datos a un usuario.

Vista: Este componente representa la interfaz gráfica para la interacción con el usuario. Dentro se ubican todos los componentes que intervienen en la visualización del resultado de la comunicación con el Controlador. Presenta al usuario los datos del modelo, la vista sabe cómo acceder a los datos del modelo, pero no sabe el significado de estos datos ni lo que el usuario puede hacer para manipularlos.

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. Escucha los sucesos desencadenados por la vista (u otro origen externo) y ejecuta la reacción apropiada a estos sucesos. En la mayoría de los casos, la reacción es llamar a un método del modelo. Puesto que la vista y el modelo están conectados a través de un mecanismo de notificación, el resultado de esta acción se reflejará automáticamente en la vista.

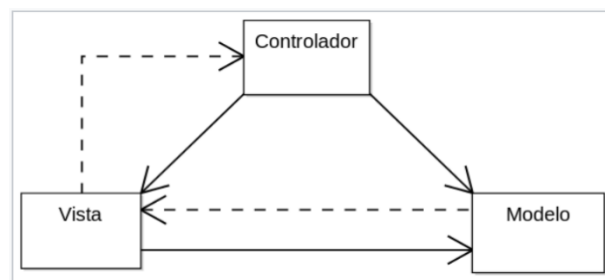


Figura 4. Diagrama de arquitectura modelo-vista-controlador.

2.5.2 Tarjetas CRC

El modelado CRC proporciona una manera sencilla de identificación y organización de las clases que son relevantes para los requerimientos de un sistema o producto (Pressman, 2010).

El modelo CRC hace uso de tarjetas de índices reales o virtuales. El objetivo es desarrollar una representación organizada de las clases. Las responsabilidades son los atributos y operaciones relevantes para la clase. Los colaboradores son aquellas clases que se requieren para dar a una clase la información necesaria a fin de completar una responsabilidad, es decir, una colaboración implica una solicitud de información o de cierta acción (Pressman, 2010).

A continuación, se describen las tarjetas CRC asociadas a cada clase de la aplicación (**Ver Anexo 3**):

Tabla 9. Tarjeta CRC asociada a la clase ProcessTable

Nombre de la clase: ProcessTable	
Responsabilidades	Clases relacionadas
login()	QApplication
editProcess()	Login
createProcess()	EditProcess
logout()	CreateProcess
elimProcess()	ListUsers
	Process
	ProcessListHandler

Tabla 10. Tarjeta CRC asociada a la clase UserTable.

Nombre de la clase: UserTable	
Responsabilidades	Clases relacionadas
createUser()	EditProcess
editUser()	CreateProcess
elimUser()	ListUsers
enableUser()	User
	UserListHandler

2.6 Patrones de diseño

Los patrones de diseño son soluciones a problemas típicos y recurrentes que se pueden encontrar a la hora de desarrollar una aplicación. Establecen un lenguaje común para programar. Identifica clases, instancias, roles, colaboraciones y distribución de responsabilidades. Provee un esquema para refinar componentes de un sistema de software y la forma en que se relacionan entre sí. Describe una estructura generalmente recurrente de comunicación de componentes que resuelve un problema de diseño general dentro de un contexto particular, lo cual permite un lenguaje de programación a alto nivel. Es la forma práctica de describir aspectos de la organización de un programa. Las características principales de los patrones de diseños son la de representar soluciones técnicas a problemas concretos, propiciar la reutilización y representar problemas frecuentes.

Patrones de Software de Asignación de Responsabilidades Generales (GRASP General Responsibilities of Assignment of Software Pattern) brindan principios generales para asignar responsabilidades y se utilizan sobre todo en la realización de diagramas de interacción (Larman C. , 2004).

Creador: Ayuda a identificar quién debe ser el responsable de la creación (o instanciación) de nuevos objetos o clases.

La nueva instancia deberá ser creada por la clase que:

- Tiene la información necesaria para realizar la creación del objeto, o
- Usa directamente las instancias creadas del objeto, o
- Almacena o maneja varias instancias de la clase
- Contiene o agrega la clase.

Una de las consecuencias de usar este patrón es la visibilidad entre la clase creada y la clase creador. Una ventaja es el bajo acoplamiento, lo cual supone facilidad de mantenimiento y reutilización. 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, encapsulación y reutilización.

```

class Process():
    def __init__(self, idprocess, taskType, desc, owner, execServer, currentStatus, workdirSize, timeSubm, lastUpdated):
        self.idprocess = idprocess
        self.taskType = taskType
        self.desc = desc
        self.owner = owner
        self.execServer = execServer
        self.currentStatus = currentStatus
        self.workdirSize = workdirSize
        self.timeSubm = timeSubm
        self.lastUpdated = lastUpdated
  
```

Figura 5. Código de ejemplo del patrón Creador.

Experto: El GRASP de experto en información es el principio básico de asignación de responsabilidades. Consiste en la asignación de una responsabilidad a la clase que cuenta con la información necesaria para llevarla a cabo. Determina cuál es la clase que debe asumir una responsabilidad a partir de la información que posee cada una. En pocas palabras la creación de un objeto o la implementación de un método, debe recaer sobre la clase que conoce toda la información necesaria para crearlo. Resultando en un diseño de alta cohesión y con una

disminución del acoplamiento. El cumplimiento de una responsabilidad requiere a menudo información distribuida en varias clases de objetos (Larman C. , 2004).

Este patrón es evidenciado en la definición de las clases de acuerdo a las funcionalidades que deben realizar a partir de la información manejada dentro del componente, como por ejemplo las clases controladoras.

En la aplicación este patrón se evidencia en las clases ProcessTable() y ListUsers() las cuales se encargan de manejar la información perteneciente a las listas de usuarios y procesos.

```

def getId(self):
    return self.idprocess

def getTaskType(self):
    return self.taskType

def getDesc(self):
    return self.desc
  
```

Figura 6. Código de ejemplo del patrón Experto.

Controlador: un controlador es un objeto de interfaz no destinada al usuario que se encarga de manejar un evento del sistema. Sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la que recibe los datos del usuario y la que los envía a las distintas clases según el método llamado. Sugiere que la lógica de negocios debe estar separada de la capa de presentación, esto permite aumentar la reutilización de código y a la vez tener un mayor control. Se recomienda dividir los eventos del sistema en un mayor número de controladores para poder aumentar la cohesión y disminuir el acoplamiento.

```

def setId(self, setElement):
    self.idprocess = setElement

def setTaskType(self, setElement):
    self.taskType = setElement

def setDesc(self, setElement):
    self.desc = setElement
  
```

Figura 7. Código de ejemplo del patrón Creador.

Alta Cohesión: en la perspectiva del diseño orientado a objetos, la cohesión o cohesión funcional es una medida que muestra que tan relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente

relacionadas que no realicen un trabajo enorme, clases con responsabilidades moderadas en un área funcional que colaboran con las otras para llevar a cabo las tareas. Indica que la información que almacena una clase debe ser coherente y debe estar (en la medida de lo posible) relacionada con la clase. Este patrón es evidenciado en la estrecha relación que guardan los nombre de las funcionalidades y variables declaradas en la clase con respecto a su función, esto se muestra en el método de la autenticación del usuario.

Bajo Acoplamiento: el acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas. Propone tener las clases lo menos ligadas entre sí, de tal forma que, en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de clases. El bajo acoplamiento soporta el diseño de clases más independientes y reutilizables, lo cual reduce el impacto de los cambios y acrecienta la oportunidad de una mayor productividad. Este patrón es evidenciado en cada una de los casos de uso del módulo, cada clase controladora maneja la información de las clases entidades que están relacionadas con las funcionalidades que se especifican en dicha clase controladora para lograr un bajo acoplamiento.

2.7 Modelo de datos

Los modelos de datos aportan la base conceptual para diseñar aplicaciones que hacen un uso intensivo de datos. El diseño de una base de datos consiste en definir la estructura de los datos que debe tener un sistema de información determinado. Para ello se suelen seguir por regla general unas fases en el proceso de diseño, definiendo para ello el modelo conceptual, el lógico y el físico. Generalmente es usado para describir la representación lógica y física de la información persistente manejada por el sistema.

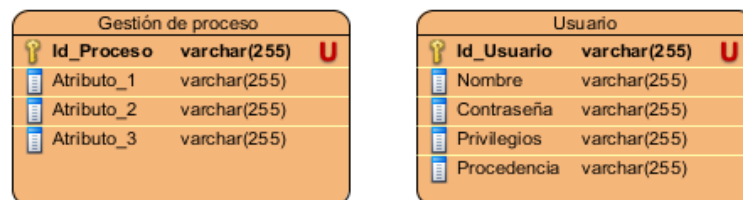


Figura 8. Diagrama de entidad-relación.

2.8 Conclusiones parciales del capítulo

Al concluir este capítulo se obtuvo la definición de los requisitos funcionales y su encapsulamiento mediante las historias de usuario, que permitieron describir las operaciones lógicas de la propuesta solución. La generación del plan de iteraciones y entrega posibilitó una adecuada planificación de las funcionalidades a desarrollar de la solución. La estructuración y organización del diseño de las funcionalidades mediante el patrón arquitectónico MVC, tarjetas CRC y modelo de datos, permitió especificar y representar los elementos abstractos tomados como referencias en la etapa de Implementación.

Capítulo 3. Implementación y pruebas

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

3.1 Estándares de codificación

La adopción de estándares de estilo y codificación son de vital importancia para asegurar la calidad del software. Un código fuente completo debe reflejar un estilo armonioso, como si un único programador hubiera escrito todo el código de una sola vez. Si bien los programadores deben implementar un estándar de forma prudente, este debe estar bien definido a nivel departamental, por tanto, al comenzar un proyecto de software es necesario establecer un estándar de codificación único para asegurarse de que todos los programadores del proyecto trabajen de forma coordinada.

Las convenciones de código o estándares de codificación son importantes para los programadores por un gran número de razones (Arias, 2014):

- El 80% del costo del código de un programa va a su mantenimiento.
- Casi ningún software es mantenido toda su vida por el autor original.
- Las convenciones de código mejoran la lectura del software lo que permite entender código nuevo de manera más óptima y rápida.
- Si distribuyes tu código fuente como un producto, necesitas asegurarte de que está bien hecho y presentado como cualquier otro producto.

A continuación, se presentan algunos de los estándares de codificación definidos y aplicados al sistema:

- Se debe utilizar como idioma el español, las palabras no se acentuarán.
- Las líneas en blanco mejoran la facilidad de lectura separando secciones de código que están lógicamente relacionadas. Se deben usar siempre dos líneas en blanco en las siguientes circunstancias:
 - Entre las secciones de un fichero fuente.
 - Entre las definiciones de clases.
- Se debe usar siempre una línea en blanco en las siguientes circunstancias:
 - Entre métodos.

- Entre las variables locales de un método y su primera sentencia.
- Antes de un comentario de bloque o de un comentario de una línea.
- Entre las distintas secciones lógicas de un método para facilitar la lectura.
- Respecto a las normas de inicialización, declaración y colocación de variables, constantes, clases y métodos:
 - Todas las instancias y variables de clases o métodos empezarán con minúscula. Las palabras internas que lo forman, si son compuestas, empiezan con su primera letra en mayúsculas. Los nombres de variables no deben empezar con los caracteres guion bajo "_" o signo de peso "\$", aunque ambos están permitidos por el lenguaje.
 - Los nombres de variables de un solo carácter se deben evitar, excepto para variables índices temporales.
 - Los nombres de las variables declaradas como constantes deben aparecer totalmente en mayúscula separando las palabras con un guion bajo ("_").
 - Los nombres de las clases deben ser sustantivos, cuando son compuestos tendrán la primera letra de cada palabra que lo forma en mayúscula. Mantener los nombres de las clases simples y descriptivas. Usar palabras completas, evitar acrónimos y abreviaturas.
 - Los métodos deben ser verbos, cuando son compuestos tendrán la primera letra en minúscula y la primera letra de las siguientes palabras que lo forman en mayúscula.
- Respecto a la indentación y longitud de la línea:
 - Se deben emplear cuatro espacios como unidad de indentación. La construcción exacta de la indentación (espacios en blanco contra tabuladores) no se especifica. Los tabuladores deben ser exactamente cada ocho espacios.
 - Evitar las líneas de más de ochenta caracteres, ya que no son manejadas bien por muchas terminales y herramientas.

3.2 Fase de desarrollo

En la fase de desarrollo se realizarán las tareas relacionadas con la implementación de cada una de las historias de usuario correspondientes a cada iteración. Además, se irá realizando una revisión del plan de iteraciones en caso de existir alguna posible modificación. A partir del desglose de cada historia de usuario, se le definieron un conjunto de tareas de programación o ingeniería que permitirán implementar exitosamente cada una de estas.

Tareas de ingeniería

A continuación, se definen las tareas de ingeniería para cada historia de usuario agrupadas por iteraciones. Estas tareas tienen como objetivo definir de manera simple cada una de las actividades necesarias para dar cumplimiento a las historias de usuario.

Tabla 11. Tareas de ingeniería para la iteración 1

Historias de usuario	Tareas de ingeniería
Autenticar	1. Autenticar usuario
Administrar proceso	1. Crear proceso 2. Modificar proceso 3. Eliminar proceso 4. Listar proceso

Tabla 12. Tarea de ingeniería #1 Autenticar usuario.

Tarea	
Número de tarea: 1	Número de historia de usuario: 1
Nombre de la tarea: Autenticar usuario	
Tipo de tarea: Desarrollo	Puntos estimados: 1
Fecha de inicio: 1 de noviembre	Fecha de fin: 8 de noviembre
Descripción: El usuario se autentica en el sistema y recibe su rol y sus privilegios.	

Tabla 13. Tarea de ingeniería #2 Crear proceso.

Tarea	
Número de tarea: 1	Número de historia de usuario: 2
Nombre de la tarea: Crear proceso	
Tipo de tarea: Desarrollo	Puntos estimados: 0.5
Fecha de inicio: 8 de noviembre	Fecha de fin: 11 de noviembre
Descripción: El usuario puede añadir un proceso a la lista de procesos.	

Tabla 14. Tarea de ingeniería #3 Modificar proceso.

Tarea	
Número de tarea: 2	Número de historia de usuario: 2
Nombre de la tarea: Modificar proceso	
Tipo de tarea: Desarrollo	Puntos estimados: 0.5
Fecha de inicio: 11 de noviembre	Fecha de fin: 14 de noviembre
Descripción: El usuario puede modificar un proceso de la lista de procesos.	

Tabla 15. Tarea de ingeniería #4 Eliminar proceso.

Tarea	
Número de tarea: 3	Número de historia de usuario: 2
Nombre de la tarea: Eliminar proceso	
Tipo de tarea: Desarrollo	Puntos estimados: 0.5
Fecha de inicio: 14 de noviembre	Fecha de fin: 19 de noviembre
Descripción: El usuario puede eliminar un proceso de la lista de procesos.	

Tabla 16. Tarea de ingeniería #5 Listar proceso

Tarea	
Número de tarea: 4	Número de historia de usuario: 2
Nombre de la tarea: Listar proceso	
Tipo de tarea: Desarrollo	Puntos estimados: 0.5
Fecha de inicio: 19 de noviembre	Fecha de fin: 22 de noviembre
Descripción: El sistema lista los procesos y el usuario los puede visualizar.	

Tabla 17. Tareas de ingeniería para la iteración 2.

Historias de usuario	Tareas de ingeniería
Administrar usuario	1. Crear usuario 2. Modificar usuario 3. Eliminar usuario 4. Listar usuario 5. Habilitar usuario

Tabla 18. Tarea de ingeniería #6 Crear usuario

Tarea	
Número de tarea: 6	Número de historia de usuario: 3
Nombre de la tarea: Crear usuario	
Tipo de tarea: Desarrollo	Puntos estimados: 0.4
Fecha de inicio: 22 de noviembre	Fecha de fin: 25 de noviembre
Descripción: El administrador puede insertar un usuario en el sistema.	

Tabla 19. Tarea de ingeniería #7 Modificar usuario

Tarea	
Número de tarea: 7	Número de historia de usuario: 3
Nombre de la tarea: Modificar usuario	
Tipo de tarea: Desarrollo	Puntos estimados: 0.4
Fecha de inicio: 25 de noviembre	Fecha de fin: 28 de noviembre
Descripción: El administrador puede modificar un usuario del sistema.	

Tabla 20. Tarea de ingeniería #8 Eliminar usuario

Tarea	
Número de tarea: 8	Número de historia de usuario: 3
Nombre de la tarea: Eliminar usuario	
Tipo de tarea: Desarrollo	Puntos estimados: 0.6
Fecha de inicio: 28 de noviembre	Fecha de fin: 2 de diciembre
Descripción: El usuario puede eliminar un usuario del sistema.	

Tabla 21. Tarea de ingeniería #9 Listar usuario

Tarea	
--------------	--

Número de tarea: 9	Número de historia de usuario: 3
Nombre de la tarea: Listar usuario	
Tipo de tarea: Desarrollo	Puntos estimados: 0.1
Fecha de inicio: 2 de diciembre	Fecha de fin: 3 de diciembre
Descripción: El sistema lista los usuarios del sistema y el administrador los visualiza.	

Tabla 22. Tarea de ingeniería #10 Habilitar usuario

Tarea	
Número de tarea: 10	Número de historia de usuario: 3
Nombre de la tarea: Habilitar usuario	
Tipo de tarea: Desarrollo	Puntos estimados: 0.5
Fecha de inicio: 3 de diciembre	Fecha de fin: 6 de diciembre
Descripción: El administrador puede habilitar o deshabilitar un usuario en el sistema.	

3.3 Fase de prueba

Las pruebas son un conjunto de actividades que se pueden planificar por adelantado y llevar a cabo sistemáticamente. Por esta razón se debe definir en el proceso de la ingeniería del software. Todo esto contribuye a elevarla calidad de los productos desarrollados y a la seguridad de los programadores a la hora de introducir cambios o modificaciones.

Uno de los pilares de la Programación extrema es el proceso de pruebas. XP anima a probar constantemente tanto como sea posible. Esto permite aumentar la calidad de los sistemas reduciendo el número de errores no detectados y disminuyendo el tiempo transcurrido entre la aparición de un error y su detección. También permite aumentar la seguridad de evitar efectos colaterales no deseados a la hora de realizar modificaciones.

La metodología XP divide las pruebas en dos grupos: pruebas unitarias, desarrolladas por los programadores, encargadas de verificar el código de forma automática y las pruebas de aceptación, destinadas a evaluar si al final de una iteración se obtuvo la funcionalidad requerida, además de comprobar que dicha funcionalidad sea la esperada por el cliente.

Se decide realizar las pruebas de aceptación a los módulos implementados, debido a que el objetivo de estas es verificar que el sistema cumpla con los requisitos establecidos por el usuario. De esta forma se puede obtener el grado de satisfacción del cliente (Joskowicz, 2008).

Pruebas unitarias

Para las pruebas unitarias se utiliza el tipo de prueba caja blanca, en donde se realiza un examen minucioso del procedimiento, comprobando los caminos lógicos del programa, los ciclos y condiciones, y examinado el estado del programa en varios puntos. Además, garantizan que:

- Se ejecutan al menos una vez todos los caminos independientes de cada interfaz.
- Se utilizan las decisiones en su parte verdadera y en su parte falsa.
- Se ejecuten todos los bucles en sus límites.
- Se utilizan todas las estructuras de datos internas.

Prueba del camino básico

La prueba de caja blanca realizada a la propuesta de solución fue la prueba del camino básico, a partir del cálculo de la Complejidad ciclomática del algoritmo a ser analizado.

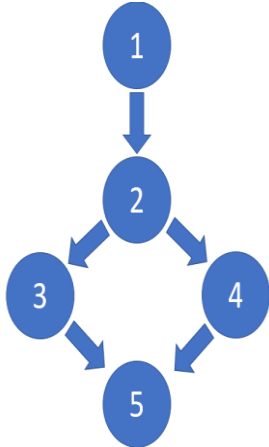
El valor calculado define el número de caminos independientes del conjunto básico de un programa. Esto indica el límite superior para el número de pruebas que se deben realizar, para asegurar que se ejecuta cada sentencia al menos una vez.

Para realizarla se enumeran las sentencias de código y a partir de ahí se elabora el grafo de flujo de esta funcionalidad. A continuación, se describen la serie de pasos a seguir:

1. Notación del grafo de flujo: se utiliza el código como base y se realiza la representación del grafo de flujo, mediante una sencilla notación. Cada construcción estructurada tiene su correspondiente símbolo.
 - Nodo: cada círculo denominado nodo, representa una o más sentencias procedimentales.
 - Arista: las flechas del grafo de flujo, denominadas aristas, representan el flujo de control y son análogas a las flechas del diagrama de flujo.
 - Región: las áreas delimitadas por aristas y nodos se denominan regiones.
2. Complejidad ciclomática: se utilizó la siguiente forma: $V(G)$, de un grafo de flujo G se define como: $V(G)=A-N+2$, donde A es el número de aristas del grafo de flujo y N es el número de nodos.
3. Determinar un conjunto básico de caminos linealmente independientes: el valor de $V(G)$ es el número de caminos linealmente independientes de la estructura de control del programa.
4. Obtención de casos de prueba: se definen los casos de prueba que forzarán la ejecución de cada camino del conjunto básico.

A continuación, se aplica la prueba al método `elimProceso()` de la clase `ProcessTable()`.

Tabla 23. Caso de prueba para la clase *ProcessTable()*

Pruebas de caja blanca	
Probado por: Dairon Canel Gómez	
Código al que se le aplica	Representación en grafo de flujo
<pre> 379 def elimProceso(self): 380 currentIndex = self.ui.treeView.selectionModel().currentIndex().row() 381 currentProcess = self.listaProcesos.getProcess(currentIndex) 382 processName = "" 383 384 att1 = currentProcess.getTaskType() 385 att2 = currentProcess.getDesc() 386 att3 = currentProcess.getOwner() 387 att4 = currentProcess.getExecServer() 388 att5 = currentProcess.getCurrentStatus() 389 att6 = currentProcess.getWorkdirSize() 390 att7 = currentProcess.getTimeSubm() 391 att8 = currentProcess.getLastUpdated() 392 393 processName = ("TASKTYPE: " + att1 + "\nDESC: " + att2 + "\nOWNER: " + att3 + 394 "\nEXECSERVER: " + att4 + "\nCURRENTSTATUS: " + att5 + "\nWORKDIRSIZE: " + att6 + 395 "\nTIMESUBM: " + att7 + "\nLASTUPDATED: " + att8) 396 397 msgElim = ("Esta seguro que desea eliminar el proceso:\n\n" + processName + "\n\n") 398 399 msgBox = QMessageBox(QMessageBox.Warning, "Eliminar proceso", msgElim, QMessageBox.NoButton, self) 400 msgBox.addButton("Eliminar", QMessageBox.AcceptRole) 401 msgBox.addButton("&Cancelar", QMessageBox.RejectRole) 402 403 if msgBox.exec_() == QMessageBox.AcceptRole: 404 self.listaProcesos.deleteProcess(self.model, currentIndex) 405 406 else: 407 self.clearList() </pre>	 <pre> graph TD 1((1)) --> 2((2)) 2 --> 3((3)) 2 --> 4((4)) 3 --> 5((5)) 4 --> 5 </pre>
<p>Figura 9. Código a aplicar pruebas de caja blanca</p>	
Complejidad Ciclomática:	Caminos independientes
$V(G) = (A - N) + 2 = 5 - 5 + 2 = 2$	1. 1 2 3 5
	2. 1 2 4 5
Caso de prueba para el camino básico No.1	
Descripción: Elimina un elemento de la lista de procesos	
Condición de ejecución: El usuario previamente selecciona un proceso y presiona el botón eliminar.	
Procedimiento prueba automatizada	
Datos de entrada:	self
Tipo de dato esperado:	void
Evaluación del caso de prueba: Satisfactoria	
Caso de prueba para el camino básico No.2	

Descripción: Cierra la ventana emergente y deselecta el elemento de la lista de procesos.	
Condición de ejecución: El usuario previamente selecciona un proceso y presiona el botón cancelar	
Procedimiento prueba automatizada	
Datos de entrada:	self
Tipo de dato esperado:	void
Evaluación del caso de prueba: Satisfactoria	

Se logró el cálculo de la complejidad ciclomática para los algoritmos de la aplicación. La misma visualiza la forma que se emplea para obtener dicho cálculo con las estructuras especificadas. Los resultados adquiridos en el cálculo de la complejidad ciclomática definen el número de caminos independientes dentro de un fragmento de código y los casos de pruebas diseñados.

Pruebas de aceptación

Las pruebas de aceptación son especificadas por el cliente, se centran en las características y funcionalidades generales del sistema que son visibles. Estas pruebas derivan de las HU que se han implementado como parte de la liberación del software. Una prueba de aceptación es como una caja negra. Cada una de ellas representa una salida esperada del sistema. Es responsabilidad del cliente verificar la corrección y toma de decisiones acerca de estas pruebas.

A continuación, se muestran representaciones de las pruebas de aceptación a realizarse:

Tabla 24. Prueba de aceptación #1 Autenticar usuario

Caso de prueba de aceptación	
Código: HU1_P1	Historia de usuario: 1
Nombre: Autenticar usuario	
Descripción: Prueba para la funcionalidad Autenticar usuario	
Condiciones de ejecución: Ninguna	
Pasos de ejecución: El usuario rellena los campos "nombre" y "contraseña" y presiona el botón aceptar.	
Resultados esperados: El usuario se ha añadido sin problemas.	

Tabla 25. Prueba de aceptación #2 Administrar proceso

Caso de prueba de aceptación	
Código: HU1_P2	Historia de usuario: 2
Nombre: Administrar proceso	
Descripción: Prueba para la funcionalidad Administrar proceso	

Condiciones de ejecución: El usuario debe haberse autenticado con anterioridad
Pasos de ejecución: El usuario puede crear, modificar o eliminar un proceso.
Resultados esperados: La lista de procesos fue administrada correctamente

Tabla 26. Prueba de aceptación #3 Administrar usuario

Caso de prueba de aceptación	
Código: HU3_P3	Historia de usuario: 3
Nombre: Administrar usuario	
Descripción: Prueba para la funcionalidad Administrar usuario	
Condiciones de ejecución: El administrador debe haber insertado previamente al menos un usuario	
Pasos de ejecución: El administrador puede crear, modificar o eliminar un usuario.	
Resultados esperados: Se administró correctamente la lista de usuarios	

Para las pruebas de aceptación se diseñaron diez casos de prueba de los cuales en la primera iteración se obtuvieron tres no conformidades de cinco casos de prueba, las cuales fueron corregidas satisfactoriamente. En la segunda iteración se obtuvieron tres casos de pruebas donde se identificó una no conformidad la cual fue rectificada de forma satisfactoria. En la tercera iteración se obtuvieron dos casos de pruebas, y no se identificaron no conformidades resultados mostrados en la gráfica siguiente:

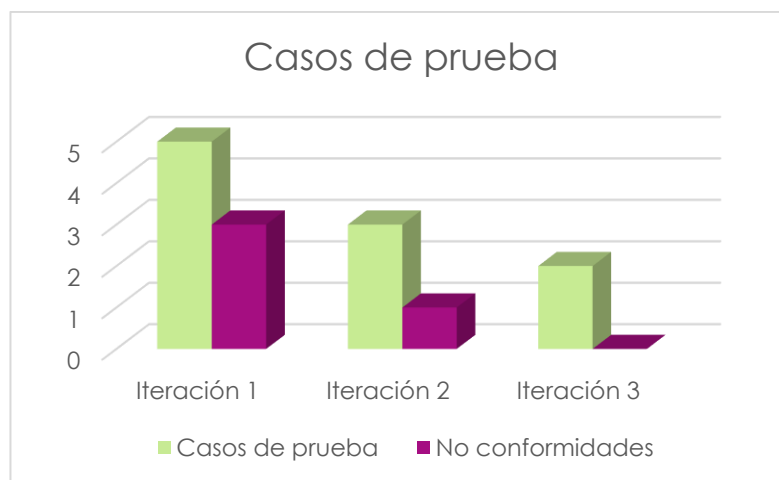


Figura 11. Casos de prueba por iteración. Fuente: elaboración propia.

3.4 Conclusiones parciales del capítulo

Los patrones de diseño utilizados GRASP: Creador, Controlador, Experto, Bajo Acoplamiento y Alta Cohesión, en conjunto con los estándares de codificación señalados permitieron estructurar la solución respondiendo a diversos problemas en la implementación de los requisitos funcionales.

La especificación de las tareas de programación permitió obtener un desglose por unidades de las funcionalidades encapsuladas mediante las historias de usuario. Los diseños de los casos de prueba de

aceptación y unidad permitieron guiar la ejecución de las actividades de verificación y validación para un correcto funcionamiento de la aplicación. Las no conformidades resultantes por iteraciones permitieron realizar un análisis incremental sobre el funcionamiento del módulo, logrando así todos los objetivos propuestos.

Conclusiones

Con el desarrollo del presente trabajo de diploma se ha dado cumplimiento al objetivo general propuesto, concluyendo lo siguiente:

- Se constató en la literatura estudiada que el tema es novedoso, actual y pertinente y además contribuyó a la elaboración de los fundamentos teórico-metodológico de la investigación, así como la identificación de los principales problemas presentes en la actualidad con respecto al ámbito de las neurociencias en Cuba, determinando características aplicables a la propuesta solución.
- El ambiente de desarrollo está acorde con el tipo de proyecto y las tendencias internacionales y permitió establecer la infraestructura tecnológica para ejecutar la propuesta solución.
- Las funcionalidades que permiten la gestión de la seguridad y las colas de procesamiento fueron implementadas a partir de la fase de análisis y diseño.
- A partir del análisis y diseño de la propuesta solución fue posible implementar las funcionalidades que permiten la gestión de la seguridad y las colas de procesamiento.
- La calidad del producto final fue validada al aplicar pruebas de software a la solución en todas sus etapas de desarrollo lo cual determinó la aceptación y la evaluación funcional del módulo.

Recomendaciones

Para dar continuidad a la presente investigación se recomienda:

- Integrar el componente a la plataforma BrainSSys.
- Identificar otros algoritmos para optimizar la gestión de colas de procesamiento.

Bibliografía

- Abraham Silberschatz, P. B. (1994). *Sistemas operativos: conceptos fundamentales*.
- Adee, & Sally. (2008). Reverse engineering the brain. *45*(6).
- Alcaraz, V. M., & Gumá, E. (2001). *Texto de Neurociencias Cognitivas*. México, DF: El Manual Moderno.
- Arias, M. (2014). Estándares de codificación.
- Ausín, T., Morte, R., & Monasterio, A. (2020). Neuroderechos: Derechos humanos para las neurotecnologías. *Diario La Ley, Sección Ciberderecho*(Nº 43), 1-7.
- Ayers, J., Davis, J., & Alan, R. (2002). *Neurotechnology for biomimetic robots*. MIT.
- BIL. (2021). *The Brain Image Library*. Retrieved from <https://www.brainimagelibrary.org>
- Bogdanov, E. (2012). *IEEE*. (Global engineering education conference (educon)) Retrieved 5 31, 2018, from A social media platform in higher education. : <https://ieeexplore.ieee.org/abstract/document/6201105/>
- Brown , M., Valdés , Y., & González, E. (2012). Repercusión del desarrollo de las neurociencias en la solución de problemas sociales. *Hum Med*, *12*(2).
- Calhoun, V., & Adali, T. (2008). Feature-based fusion of medical imaging data. *IEEE Transactions on Information Technology in Biomedicine*, *13*, 711-720.
- C-BIG. (2021). *C-BIG Repository*. Retrieved junio 26, 2021, from <https://www.mcgill.ca/neuro/open-science/c-big-repository>
- Cheung, K.-H., Lim, E., Samwald, M., Chen, H., Marengo, L., Holford, M. E., . . . Miller, P. L. (2009). Approaches to neuroscience data integration. *Brief Bioinform*, *10*(4), 345–353. doi:10.1093/bib/bbp029
- CNEURO. (2021). *Centro de Neurociencias de Cuba*. (CNEURO) Retrieved Mayo 5, 2021, from CNEURO: <https://www.cneuro.cu/>
- Cohn, M. (2006). *Agile Estimating and Planning*. Pearson Education.
- Crasto, C., & SH., K. (2007). *Neuroinformatics(Methods in molecular Biology)*. Totowa, NJ: Humana Press Inc.

- Crispin, L. (2002). *Testing Extreme Programming*. Addison Wesley Professional.
- Das, S., Zijdenbos, A., Harlap, J., Vins, D., & Evans, A. (2012). LORIS: a web-based data management system for multi-center studies. *Frontiers in neuroinformatics*.
- Elssamadisy, A. (2008). Agile Adoption Patterns. A roadmap to organizational success. *Addison Wesley Professional*.
- García, E. (2020). Neurociencia, Humanismo y Posthumanismo. *Logos Anales del Seminario de Metafísica*, 53, 9-31. doi:<http://dx.doi.org/10.5209/asem.70833>
- Garriga, A. (n.d.). *¿Qué es la metodología ágil?* Retrieved 4 4, 2018, from RECURSOS ENPROJECTMANAGEMENT: <https://www.rekursosenprojectmanagement.com/metodologia-agil/>
- GBC. (2021). *The Global Brain Consortium*. Retrieved junio 25, 2021, from GBC: <https://globalbrainconsortium.org/>
- Gorgolewski, K. J., Varoquaux, G., Rivera, G., & Schwarz, Y. (2015). NeuroVault.org: a web-based repository for collecting and sharing unthresholded statistical maps of the human brain. *Frontiers*, 9(8), 1-9. doi:10.3389/fninf.2015.00008
- Greenfield, S. (2002). Reino Unido.
- HFSP. (2021). *Human Frontier Science Program*. Retrieved junio 25, 2021, from HFSP: <https://www.hfsp.org/>
- Hofmeister, C., Nord, R., & D, S. (2000). *Applied Software Architecture*. Edtion ed.: Addison-Wesley Professional.
- IBI. (2021). *International Brain Initiative*. Retrieved junio 25, 2021, from IBI: <https://www.internationalbraininitiative.org/>
- IBRO. (2021). *International Brain Research Organization*. Retrieved junio 25, 2021, from IBRO: <https://ibro.org/>
- Jacobson, I., Booch, G., & Rumbaugh, J. (2000). *El Proceso Unificado de Desarrollo de Software*. España: Pearson Educación.
- Joskowicz, J. (2008). Reglas y prácticas en eXtreme Programming. *vol. 22*.

- Kandell, E. R., Jessell, T. M., & Schwartz, J. H. (1997). *Neurociencia y Conducta*. Madrid: Prentice Hall.
- Larman, C. (2000). UML y Patrones. Introducción al análisis y diseño orientado a objetos. *México: Prentice Hall*.
- Larman, C. (2004). *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development*. ISBN 0-13-148906-2.
- Letelier, P., & M.C, P. (2003). Metodologías Ágiles en el Desarrollo de Software: Extreme Programming(XP). *Universidad Politécnica de Valencia*.
- Linne, M.-L. (2018). Neuroinformatics and Computational Modelling as Complementary Tools for Neurotoxicology Studies. *Basic & Clinical Pharmacology & Toxicology*, 123, 56–61. doi:Doi: 10.1111/bcpt.13075
- López Moranchel, I. (15/01/2020). *Fundamentos de la técnica de imagen por resonancia magnética*.
- López Yepes, A., & Sánchez Gay, F. (1994). Fototecas digitales en prensa: formatos gráficos, entornos y sistemas informáticos. 3. Retrieved junio 25, 2021
- Maastricht. (2021). *Maastricht University*. Retrieved junio 25, 2021, from <https://www.maastrichtuniversity.nl/>
- Manchanda, M., & Sharma, R. (2016). A novel method of multimodal medical image fusion using fuzzy transform. *Journal of Visual Communication and Image Representation*, 40, 197-217.
- Manes, F., & Niro, M. (2014). *Usar el cerebro*. Barcelona: Paidós.
- Martínez Alcántara, S. (2009). El estudio de la integridad mental en su relación con el proceso de trabajo. *UAM-Xochimilco*, 5(1).
- McGill. (2021). *McGill University*. Retrieved junio 25, 2021, from McGill: <https://www.mcgill.ca/>
- MCIN. (2021). *McGill Centre for Integrative Neuroscience*. Retrieved from MCIN: <https://mcin.ca>
- Müller, O., & Rotter, S. (2017). Neurotechnology: Current Developments and Ethical Issues. *Frontiers*, 11(93), 1-4. doi:<https://doi.org/10.3389/fnsys.2017.00093>
- NeuroMorpho. (2021). *NeuroMorpho.Org*. Retrieved junio 25, 2021, from <http://neuromorpho.org/>
- NeuroVault. (2021, Enero). Retrieved Mayo 5, 2021, from <https://neurovault.org/>

- Orellana, A. (2019).
- Orjuela, A., & Rojas, M. (2008). Las metodologías de desarrollo ágil como una oportunidad para la ingeniería del software educativo. *vol. 5*(no. 2).
- Pressman, R. S. (2010). *Ingeniería de software un enfoque práctico, Séptima edición*. México, D. F.: Miembro de la Cámara Nacional de la Industria Editorial Mexicana, Reg.
- Purves, D., Augustine, G. J., Fitzpatrick, D., Hall, W. C., & Lamantia, A. S. (2016). *Neurociencia*. Bogotá: Médica Panamericana.
- PyQt. (2021). Retrieved from <https://doc.qt.io/qtforpython>
- Python. (2021). *Python Software Foundation*. Retrieved from <https://www.python.org/>
- Qt. (2021). *Qt*. Retrieved from <https://www.qt.io>
- Rew, R., & Davis, G. (1990). NetCDF: an interface for scientific data access. *IEEE Comput. Graph. Appl*, *10*, 76-82. doi:10.1109/38.56302
- Rosell Aiquel, R., Juppet Ewing, M. F., Ramos Marquez, Y., Ramírez Molina, R. I., & Barrientos Oradini, N. (2020). Neurociencia aplicada como nueva herramienta para la educación. *Opcion, Regular*(92), 792-818.
- RWTH-Aachen. (2021). *Aachen University*. Retrieved junio 25, 2021, from RWTH-Aachen: <https://www.rwth-aachen.de/>
- SCRIBD. (2018). *Visual Paradigm*. Retrieved 5 30, 2018, from Visual Paradigm: <https://es.scribd.com/document/65619842/Visual-Paradigm>
- Sherif, T., Rioux, P., Rousseau, M.-E., Kassis, N., Beck, N., Reza , A., . . . Evans, A. (2014). CBRAIN: a web-based, distributed computing platform for collaborative neuroimaging research.
- Sommerville, I. (2011). *Ingeniería de software Novena edición*. México.
- SQLite. (2018). *SQLite*. Retrieved 5 30, 2018, from <http://www.sqlite.org/index.html>
- Sublime Text. (2021). Retrieved from <https://www.sublimetext.com>
- The National Institute of Mental Health*. (n.d.). (National Institute of Mental Health) Retrieved Mayo 5, 2021, from The National Institute of Mental Health: <https://www.nimh.nih.gov/>

UESTC. (2021). *Universidad de Ciencia y Tecnología Electrónica de China*. Retrieved junio 25, 2021, from UESTC: <https://www.uestc.edu.cn/>

Velázquez, M., Cabrera, L., Llano, E., & Hernández, E. (2012). *Aplicando el método de Boehm y Turner*. Universidad de las Ciencias Informáticas. La Habana, Cuba.

Anexos

Documentos que complementan el cuerpo del trabajo, pero que no son indispensables para entenderlo. Todos deben ser referenciados en el cuerpo del documento.