Universidad de las Ciencias Informáticas

Facultad 6

Título: “Herramienta en Matlab para la obtención de información de la base de datos y ficheros electroencefalograma del proyecto Mapeo Cerebral Humano Cubano”

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

Autores: Marleysi López Duque

Raidel Oceguera Ravelo

Tutores: Ing. Yixander Yero Tarancón

Lic. Liliannes C. Matamoros Benítez

Consultante: MSc. Yaikiel Hernández Díaz

Junio, 2010

"Nunca consideres el estudio como una obligación sino como una oportunidad para penetrar en el bello y maravilloso mundo del saber."

Albert Einstein

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Marleysi López Duque

_____

Firma del Autor

Raidel Oceguera Ravelo

_____

Firma del Autor

Ing. Yixander Yero Tarancón

_____

Firma del Tutor

Lic. Liliannes C. Matamoros Benítez

_____

Firma del Tutor

DATOS DE CONTACTO

Tutor: Ing. Yixander Yero Tarancón

Ingeniero en Ciencias Informáticas

E-mail: yyero@uci.cu

Tutor: Lic. Liliannes C. Matamoros Benitez

Licenciada en Psicología

E-mail: lmatamorosb@uci.cu

Consultante: MSc. Yaikiel Hernández Díaz

Máster en Bioinformática

E-mail: yhernandezd@uci.cu

AGRADECIMIENTOS

Un sueño se hace realidad cuando se lucha día a día por él, cuando se pone todo el empeño y dedicación para alcanzarlo. Hoy se hizo realidad nuestro sueño, hoy después de muchos años de sacrificio hemos alcanzado la meta que una vez nos propusimos y por eso queremos agradecer a todas aquellas personas que de una forma u otra han estado apoyándonos en todo momento, por eso agradecemos en esta ocasión:

A nuestros padres *por su infinito amor, por el sacrificio realizado para ayudarnos a alcanzar nuestro sueño, por la confianza y apoyo que nos han brindado en todo momento. Sabemos que la vida no nos va a alcanzar para agradecerles todo lo que han hecho por nosotros.*

A nuestros hermanos *por ser tan especiales y estar presentes en los momentos que más los necesitamos, por todo lo que han hecho por nosotros y por el cariño que nos unen.*

A nuestros suegros *por acogernos y querernos como sus propios hijos.*

A nuestros tutores *Yixander y Lilianne por guiarnos, por el apoyo brindado, por ofrecernos sus conocimientos y ayudarnos a estar hoy aquí. Especialmente a Lili por cada una de las revisiones y recomendaciones realizadas al trabajo de diploma, por estar siempre presente y disponible para atendernos en cualquier momento y por su esmerada dedicación.*

Al tribunal *por sus críticas constructivas.*

A la oponente *Nara Lidia Pérez por sus recomendaciones.*

A los profes *Rasiel Aponsio (El vicio), Liusmila Nieto, Reinaldo Álvarez, Yuneimy Téllez, Yunet González, Alberto Garnache, Yaikiel Hernández y Martha D. Hernández (Martica) por brindarnos su apoyo incondicional y por aclarar nuestras dudas en los momentos más difíciles.*

A todos los profesores *por la educación y formación profesional que nos han brindado a lo largo de nuestra vida estudiantil, desde que comenzamos a leer y a escribir hasta hoy.*

A nuestros amigos *por los buenos momentos que pasamos juntos y los recuerdos que quedaron grabados para siempre, por todo el apoyo y la ayuda brindada.*

A los abuelos Cira, Mario y Librada *por todo su cariño y por preocuparse tanto por nosotros.*

A las primitas Mary, Nery y Lili *por recibirnos cada fin de semana con tanta alegría preguntándonos siempre como nos va.*

A nuestros tíos, primos y familia en general *por su preocupación por nosotros.*

A todos los que no mencionamos, pero que de una forma u otra recibimos su apoyo.

A todos ustedes, Gracias, Muchas Gracias por existir.

Marleysi y Raidel

DEDICATORIA

Quiero dedicar este Trabajo de Diploma a las personas más especiales que tengo en la vida, ellos son:

***Mis padres Marta M. Duque Pérez y Jorge L. López Rosquet**, por ser los mejores padres del mundo, por darme el don de conocer la vida, por guiarme por el mejor camino, por formar en mí la persona que hoy soy, por sus consejos, por su infinito amor, por ser insustituibles, por su dedicación, por confiar siempre en mí, por estar siempre a mi lado en los momentos que más los necesito, por toda la felicidad que me han dado, por el orgullo que siento de ser su hija y porque los quiero mucho, mucho, mucho…*

***Mi hermano Jaciel López Duque,** por ese infinito amor que existe entre nosotros, por ser el hermano ideal que toda persona anhela, por estar siempre unidos, por seguir mis pasos, por el cariño que me brinda en cada beso y abrazo, por ser mi hermanito del alma.*

***Mi esposo Raidel Oceguera Ravelo,** por estar a mi lado en todo momento, por darme fuerzas cuando creía que todo estaba perdido, por regañarme cuando lloraba innecesariamente, por cambiar mis lágrimas por sonrisas, por darme ese infinito amor que alegra cada día de mi vida, por todos los momentos lindos que hemos vivido juntos y los que quedan por vivir, por enseñarme todas las cosas bellas que tiene el mundo, por ser la persona más especial que he conocido y por querernos tanto.*

Marleysi López Duque

Dedico este Trabajo de Diploma a:

***Mis padres Gloria I. Ravelo Pérez y Reinaldo M. Oceguera Pérez,** por haberse sacrificado tanto durante todos estos años, por todo el amor que me han dado, por sus consejos, por siempre guiarme por el camino correcto, por todos los momentos felices que hemos pasado y por apoyarme en cada una de mis decisiones.*

***Mi hermano Reinaldo Oceguera Ravelo,** por su preocupación en todo momento.*

***Mi esposa Marleysi López Duque,** por su cariño y ternura, por aconsejarme cuando más lo necesitaba, por ser una persona especial, por darme apoyo en los momentos difíciles, por ser parte de mi conciencia, por los momentos lindos que hemos tenido, por llenar de felicidad mi vida y por el infinito amor que nos une.*

Raidel Oceguera Ravelo

RESUMEN

El proyecto Mapeo Cerebral Humano es un nuevo reto que enfrenta actualmente la comunidad científica internacional. El Centro de Neurociencias de Cuba (CNEURO) lidera las investigaciones en este campo en nuestro país a través del proyecto Mapeo Cerebral Humano Cubano (MCHC). Como todos los proyectos de investigación actuales, este requiere de soluciones informáticas que agilicen y mejoren los procesos, para la obtención de resultados a corto plazo. La Universidad de las Ciencias Informáticas ha estado vinculada al proyecto MCHC casi desde su surgimiento, con el objetivo fundamental de elaborar las soluciones de software que requieren los especialistas que laboran en el proyecto.

El proyecto MCHC, no sólo ha introducido novedosas técnicas, sino que es el primero a nivel internacional en añadir el mapeo de la actividad eléctrica cerebral a través del electroencefalograma (EEG), la información de estos análisis es almacenada en ficheros y los datos de las personas a las cuales se le realiza el estudio son almacenados en una base de datos implementada para el proyecto. Los especialistas de CNEURO necesitan recuperar información de los ficheros y de la base de datos con la utilización de Matlab, para realizar análisis estadísticos. El resultado de la presente investigación es una herramienta informática desarrollada en Matlab, que permite extraer información de la base de datos y de los ficheros generados en los estudios de la actividad eléctrica cerebral a través del electroencefalograma. Además, brinda la posibilidad de salvar la información extraída en formato mat, la cual es empleada en estudios posteriores.

Palabras clave: base de datos, electroencefalograma, fichero, herramienta, Matlab

ÍNDICE

INTRODUCCIÓN

Cuando se concluye el programa del genoma humano, que fuera en su tiempo un desafío colosal para la humanidad y hoy una gran hazaña, se presenta entonces para los científicos otro reto de alcance similar: “el proyecto del mapeo cerebral humano”. El hecho de hacer un mapa de cuáles son las diferentes funciones del cerebro que representa tanto la superficie cortical como los núcleos internos es, en realidad, un logro para la ciencia moderna. (1)

Debido a su gran complejidad, este proyecto se comenzó a desarrollar de forma internacional, con la colaboración de toda la comunidad científica y en el año 1993 se crea el Consorcio Internacional para el Mapeo Cerebral (ICBM, por sus siglas en inglés). El objetivo fundamental del proyecto es crear una enorme base de datos que permita agrupar y organizar toda la información del cerebro humano conocida hasta el momento, permitiéndole así a los científicos realizar estudios y profundizar en la comprensión del funcionamiento y estructura del cerebro, órgano más complejo del cuerpo humano. Estos estudios contribuyen al tratamiento precoz de enfermedades neurológicas, psicológicas, psiquiátricas y otras, con el fin de lograr una mejor calidad de vida de los pacientes.

El ICBM es el encargado de organizar y desarrollar el proyecto a escala mundial, está compuesto por 4 centros de investigación:

- La Universidad de California de Los Ángeles.
- Instituto Neurológico de Montreal.
- Universidad de Texas de San Antonio.
- Instituto de Medicina de la Universidad de Juelich y Heinrich Heine.

Además, en Asia y Europa existen centros que contribuyen con el desarrollo del proyecto. (2)

Cuba también formaba parte del proyecto internacional pero fue expulsada porque el proyecto estaba financiado por institutos norteamericanos y fondos federales. Por esa razón, Cuba a pesar de ser un país subdesarrollado y bloqueado por los Estados Unidos, decide comenzar el proyecto MCHC. Este proyecto es dirigido por los especialistas de CNEURO y persigue los mismos objetivos que el proyecto

internacional. Los resultados obtenidos hasta la fecha, han sido altamente valorados por la comunidad científica mundial y esto ha servido de impulso para el perfeccionamiento y fortalecimiento del proyecto.

CNEURO cuenta con la colaboración de algunas instituciones como son:

- Instituto de Neurología.
- Centro de Investigaciones Médico Quirúrgicas (CIMEQ).
- Centro Nacional de Genética Médica.
- Dirección de Salud y la Oficina del Carné de Identidad del municipio La Lisa (Ciudad de La Habana).
- Algunos policlínicos de San Agustín (Ciudad de La Habana).
- Universidad de las Ciencias Informáticas (UCI).

La UCI ha estado vinculada al proyecto casi desde su surgimiento en nuestro país, con el objetivo fundamental de elaborar herramientas para la gestión y almacenamiento de la información, para facilitar así la investigación a los especialistas.

El proyecto cubano de mapeo cerebral humano no solo ha introducido novedosas técnicas, sino que es el primero a nivel internacional en añadir el mapeo de la actividad eléctrica cerebral a través del electroencefalograma, el cual mide las ondas cerebrales a través de pequeños electrodos en forma de botón que se colocan sobre el cuero cabelludo. (3) La información resultante del electroencefalograma es guardada en ficheros llamados EEG, y la información de las personas a las cuales se les realiza el estudio del mapeo cerebral es almacenada en una base de datos implementada para el proyecto. Además, los especialistas de Neurociencias utilizan Matlab, software matemático con un lenguaje de programación propio, denominado lenguaje M. Este software permite el desarrollo de algoritmos, visualización de datos, análisis de datos y cálculo numérico.

En la actualidad, después de tener almacenados y organizados los datos, los especialistas necesitan extraer determinada información, que se puede encontrar tanto en la base de datos del proyecto MCHC como en ficheros resultantes del electroencefalograma, con la utilización de Matlab para posteriormente realizar estudios sobre esta información. Hasta el momento no existe ningún medio que logre la comunicación de Matlab con la base de datos y con los ficheros EEG y esto provoca que se dificulte el trabajo de los especialistas del Centro de Neurociencias.

Debido a la necesidad de dar solución a la situación planteada, se identifica como ***problema científico***: ¿cómo extraer información de la base de datos y de los ficheros electroencefalograma del proyecto Mapeo Cerebral Humano Cubano?

Este problema determinó que el ***objeto de estudio*** de esta investigación sea: proceso de obtención de la información de base de datos y de ficheros con la utilización de Matlab y se precisa como ***campo de acción***: proceso de obtención de información de la base de datos y los ficheros EEG del proyecto Mapeo Cerebral Humano Cubano con la utilización de Matlab.

Para dar solución al problema planteado se define como ***objetivo general***: desarrollar una herramienta informática en Matlab, que permita obtener información de la base de datos y de los ficheros EEG del proyecto Mapeo Cerebral Humano Cubano.

Este objetivo general se desglosa en los siguientes ***objetivos específicos***:

- Definir las funcionalidades que debe tener la herramienta.
- Diseñar la herramienta que brinde solución al problema planteado.
- Implementar la herramienta diseñada.
- Probar el correcto funcionamiento de la herramienta implementada.

Para dar cumplimiento a los objetivos se realizaron esencialmente las siguientes ***tareas***:

1. Realización de entrevistas con el cliente para comprender cuál es la información que necesitan extraer de la base de datos.
2. Revisión de las herramientas existentes para lograr la conexión entre Matlab y base de datos.
3. Investigación sobre cómo configurar Matlab para lograr la conexión con la base de datos del proyecto MCHC.
4. Análisis de los ficheros EEG proporcionados por el cliente para comprender su estructura.
5. Investigación de las herramientas para el diseño e implementación del software.
6. Selección de las herramientas apropiadas para el diseño e implementación del software.
7. Selección de la metodología adecuada para guiar el proceso de desarrollo del software.

8. Definición de los requerimientos funcionales y no funcionales del software para el correcto funcionamiento de la herramienta.
9. Realización del diseño para facilitar la implementación de la herramienta.
10. Implementación del diseño para dar cumplimiento a los requisitos funcionales.
11. Realización de pruebas de caja negra para validar el correcto funcionamiento de la solución propuesta.

El documento está estructurado de la siguiente manera: resumen, introducción, cuatro capítulos, conclusiones, recomendaciones, referencias bibliográficas, bibliografía, anexos y glosario de términos.

Capítulo 1: Fundamentación Teórica.

En este capítulo se realiza un estudio sobre las herramientas, gestor de base de datos, metodologías de desarrollo de software y lenguaje de modelado donde se hace una selección de las más apropiadas, las cuales se usarán durante el desarrollo del software.

Capítulo 2: Características del Sistema.

En este capítulo se aborda acerca de cómo debe funcionar el sistema, haciendo una descripción general del funcionamiento del mismo. Se elabora una lista de requerimientos funcionales y no funcionales que se deben tener en cuenta para la realización de la herramienta. Se identifican los actores, casos de usos del sistema y la relación existente entre ellos.

Capítulo 3: Diseño del Sistema.

En este capítulo mediante el diseño se hace una especificación de los requerimientos con el propósito de describir cómo se debe implementar el sistema. Se realiza el diagrama de clases del diseño, los diagramas de interacción y se describen las clases más relevantes. Se aborda sobre los patrones arquitectónicos y de diseño presentes en la herramienta.

Capítulo 4: Implementación y Prueba del Sistema.

En este capítulo se realiza el diagrama de componentes y se muestra la ubicación física de los nodos de procesamiento a través del diagrama de despliegue. Además, se realizan pruebas de caja negra a la herramienta para comprobar su correcto funcionamiento a través de los casos de prueba.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Introducción

En este capítulo se realiza un estudio sobre las herramientas existentes en Matlab que sean capaces de extraer información de base de datos y de ficheros. Se describe el gestor de base de datos y lenguaje de modelado que se usan durante el desarrollo del software, también se realiza un estudio de varias metodologías de desarrollo de software donde se selecciona la más apropiada de acuerdo a las características existentes durante el ciclo de vida del software. Además, en el capítulo se analizan diferentes herramientas matemáticas que puedan satisfacer las necesidades del cliente.

1.1 Herramientas matemáticas

"No se deben separar las matemáticas de la cultura, del resto de las áreas de conocimiento, porque no se puede entender el desarrollo de nuestra sociedad sin tener en cuenta a las matemáticas."[1]

Una herramienta matemática no es más que un programa o software que ejecuta diferentes acciones, tales como: análisis numérico, cálculo simbólico, estudios estadísticos y otros, estas generalmente incorporan un lenguaje de programación propio. En la actualidad existen diversas herramientas matemáticas, entre las que se encuentran: Octave, Mathematica y Matlab. A pesar de la previa selección de Matlab por parte del cliente, se realizó un estudio de otras alternativas para valorar un cambio en dicha selección.

1.1.1 Octave

Octave es un lenguaje de alto nivel desarrollado por la comunidad de software libre para el cálculo numérico, siendo su sintaxis compatible con Matlab y es ampliamente utilizado por el entorno docente. Octave es un lenguaje de programación potente para el desarrollo de algoritmos científicos. Su principal desventaja es que no posee interfaz gráfica por tanto no se utiliza para desarrollar aplicaciones para usuarios finales. (4) A pesar de que en la actualidad existe QtOctave, que es considerada como la interfaz gráfica más completa disponible para Octave, existe el inconveniente de que esta interfaz es un proyecto completamente independiente y separado, que no forma parte de Octave. Por lo general, Octave se usa para validar algoritmos y no para desarrollar aplicaciones para usuarios finales.

[1] Según Raúl Ibáñez, profesor universitario y miembro de la Real Sociedad Matemática.

1.1.2 Mathematica

Mathematica es una herramienta especializada en análisis numérico y cálculo simbólico, que incorpora un potente lenguaje de programación propio y una interfaz externa que permite salidas a C, Fortran y TEX, además de otras potentes comunicaciones con otros paquetes mediante MathLink.

Características principales:

- Realización de cálculos y simulaciones de cualquier nivel de complejidad mediante el uso de la amplia librería de funciones que tiene incorporado.
- Rápida y fácil importación y exportación de datos, que incluye imágenes y sonido, en diferentes formatos.
- Generación de documentos interactivos, independientes de la plataforma, con textos, imágenes, expresiones matemáticas, botones e hipervínculos.
- Entrada de expresiones a través del teclado o de la paleta (programable) más adecuada.
- Construcción de complejas expresiones y fórmulas con formato automático y ruptura de líneas.

Esta herramienta presenta una sintaxis complicada, por lo que el coste de aprendizaje y adaptación es relativamente alto, el principal inconveniente de Mathematica es que a veces se obtienen resultados no esperados y no se logra determinar la procedencia de estos resultados. (5)

1.1.3 Matlab

Matlab (abreviatura de ***MAT****rix* ***LAB****oratory*, “laboratorio de matrices”), es un software matemático que ofrece un entorno de desarrollo integrado (IDE) con un lenguaje de programación propio (lenguaje M). Está disponible para las plataformas Unix, Windows y Apple Mac OS X. Es una poderosa herramienta para la resolución numérica de problemas.

Entre sus prestaciones básicas se hallan: la manipulación de matrices, la representación de datos y funciones, la implementación de algoritmos, la creación de interfaces de usuario (GUI) y la comunicación con programas en otros lenguajes. Matlab dispone además de las herramientas: GUIDE (editor de interfaces de usuario – GUI por sus siglas en inglés) y Simulink (plataforma de simulación multidominio). También se pueden ampliar las capacidades de Matlab con las cajas de herramientas (*toolboxes*); y las de

Simulink con los paquetes de bloques (*blocksets*). Es un software muy usado en universidades y centros de investigación y desarrollo. (6)

Características principales

- Lenguaje de alto nivel para cálculo técnico.
- Entorno de desarrollo para la gestión de código, archivos y datos.
- Funciones matemáticas para álgebra lineal, estadística, análisis de Fourier, filtraje, optimización e integración numérica.
- Funciones gráficas bidimensionales y tridimensionales para visualización de datos.
- Herramientas para crear interfaces gráficas.
- Funciones para integrar los algoritmos basados en Matlab con aplicaciones y lenguajes externos, tales como: C/C++, FORTRAN, Java, COM y Microsoft Excel.

Principales ventajas de Matlab

- Posee una excelente ayuda incorporada.
- Es multiplataforma.
- Alta precisión en el cálculo y procesamiento de imágenes.
- Colección útil de funciones predefinidas.
- Comunidad de usuarios muy extendida.
- Potente herramienta matemática.

La principal desventaja de Matlab es que es un software propietario pero a pesar de este inconveniente se decidió el uso de esta herramienta principalmente por las necesidades y exigencias del cliente pues desea realizar estudios posteriores a través de información recopilada en ficheros con extensión mat. El hecho de que Matlab sea software propietario no es un obstáculo que impida su uso pues el cliente cuenta con la licencia de dicho software lo que posibilita su utilización sin restricciones. Otra de las razones por las cuales se usa Matlab es porque el cliente cuenta con conocimientos básicos de programación en dicha herramienta, lo que le posibilita agregarle funcionalidades que puedan surgir posteriormente. Además, es una herramienta multiplataforma, con alta precisión en el cálculo y en el procesamiento de imágenes y tiene incorporado una excelente ayuda. Se descarta la posibilidad de usar Octave debido a que las aplicaciones hechas en el mismo se ejecutan sobre consola. De modo que esto imposibilita el desarrollo

de interfaces de usuario para que este pueda interactuar con el software. Además, no se selecciona la herramienta Mathematica debido a que el costo de aprendizaje es relativamente alto, ya que presenta una sintaxis complicada.

Una vez seleccionada la herramienta matemática se investigó sobre qué características presenta dicha herramienta para la manipulación de ficheros, donde se obtuvo como resultado que Matlab incluye diferentes funciones, entre ellas se encuentran:

- fopen: función para abrir ficheros o para obtener información sobre los ficheros abiertos.
- fclose: función para cerrar uno o más ficheros abiertos.
- textread: permite leer datos de un fichero.
- feof: comprueba el fin de línea del fichero.

También, se investigó acerca de cómo obtener información de una base de datos, donde se obtuvo como resultado la existencia de un *toolbox* en Matlab llamado *Database,* el cual contiene la herramienta *Query Builder* que permite realizar consultas a bases de datos. A pesar de las ventajas que brinda Query Builder no se hace uso de la misma debido a que para interactuar con ella el usuario debe tener conocimiento sobre cómo realizar consultas SQL (*Structured Query Language*, lenguaje estructurado de consultas) y dominar el idioma inglés. Además, tiene incluida operaciones que no son de interés para el cliente. Por tal motivo se decide desarrollar una herramienta con una interfaz amigable que cumpla con los requisitos establecidos por el cliente y que sea capaz de extraer información de una base de datos que está gestionada por un sistema gestor de base de datos.

1.2 Sistema Gestor de Bases de Datos (SGBD)

Los sistemas de gestión de bases de datos, en inglés DataBase Management System (DBMS) son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan.

1.2.1 PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos, distribuido bajo licencia *Berkeley Software Distribution* (BSD, en español significa Distribución de Software Berkeley) y con su código fuente disponible libremente.

Características generales:

- Integridad referencial
- Copias de seguridad en caliente (Online/hot backups).
- Unicode.
- Juegos de caracteres internacionales.
- Completa documentación.
- Licencia BSD.
- Disponible para Linux y UNIX en todas sus variantes (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64) y Windows. (7)

1.2.2 Driver para acceder a PostgreSQL

Un driver para acceder a base de datos es una interfaz de programación de aplicaciones (API del inglés *application programming interface*) que permite la ejecución de operaciones sobre base de datos desde el lenguaje de programación determinado, independientemente del sistema operativo donde se ejecute o de la base de datos a la cual se accede, utilizando el dialecto SQL del modelo de base de datos que se utilice. (8)

Existen varios controladores (drivers) para lograr la conexión entre Matlab y base de datos, entre los que se encuentran ODBC (*Open DataBase Connectivity*, conectividad abierta a bases de datos) y JDBC (*Java DataBase Connectivity*, conectividad java a bases de datos), se selecciona el driver JBDC ya que el rendimiento es superior al de ODBC.

1.3 Metodología de desarrollo de software

En la actualidad, desarrollar un software es una tarea complicada, que exige una metodología de desarrollo para simplificar el trabajo y garantizar un producto con la calidad requerida. Una metodología propone un conjunto completo de actividades necesarias para transformar los requisitos de usuario en un producto. No existe una metodología de desarrollo de software universal pues las características de cada proyecto (tiempo de duración, equipo de desarrollo, recursos) exigen que se escoja la metodología más adecuada para favorecer el trabajo de las personas que intervienen y lograr la satisfacción del cliente con el cumplimiento de los requisitos establecidos.

1.3.1 Proceso Unificado de Desarrollo (*Rational Unified Process*, RUP)

El Proceso Unificado de Desarrollo (RUP) es el resultado de varios años de desarrollo y uso práctico en el que se han unificado técnicas de desarrollo. Es un proceso bien definido, estructurado y adaptable a las características y necesidades de cada proyecto específico. El ciclo de vida de RUP tiene tres características fundamentales: dirigido por casos de uso, centrado en la arquitectura y es iterativo e incremental. Este proceso de desarrollo de software se divide en 4 fases esenciales:

1. Inicio: el objetivo de esta fase es determinar la visión del proyecto.
2. Elaboración: el objetivo de esta fase es determinar la arquitectura óptima.
3. Construcción: el objetivo de esta fase es obtener la capacidad operacional inicial.
4. Transición: el objetivo de esta fase es alcanzar la liberación del proyecto.

En RUP se han agrupado las actividades en grupos lógicos definiéndose 9 flujos de trabajo principales. Los 6 primeros son conocidos como flujos de ingeniería y los 3 últimos como flujos de apoyo.

Flujos de trabajo:

1. Modelamiento del negocio
2. Requerimientos
3. Análisis y diseño
4. Implementación
5. Prueba
6. Instalación o Despliegue
7. Administración del proyecto
8. Administración de configuración y cambios
9. Ambiente

RUP define como sus principales elementos:

Trabajadores (“quién”): Define el comportamiento y responsabilidades (rol) de un individuo, grupo de individuos, sistema automatizado o máquina, que trabajan en conjunto como un equipo. Ellos realizan las actividades y son propietarios de elementos.

Artefactos (”qué”): Productos tangibles del proyecto que son producidos, modificados y usados por las actividades. Pueden ser modelos, elementos dentro del modelo, código fuente y ejecutables.

Flujo de trabajo (“cuándo”): Secuencia de actividades realizadas por trabajadores y que produce un resultado de valor observable.

Actividades (“cómo”): Es una tarea que tiene un propósito claro, es realizada por un trabajador y manipula elementos.

1.3.2 Programación extrema (*Extreme Programming*, XP)

Consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar al éxito del proyecto. Es utilizada en proyectos de corto plazo y pequeño equipo de desarrollo.

¿Qué es lo que propone XP?

- Empieza en pequeño y añade funcionalidad con retroalimentación continua.
- El manejo del cambio se convierte en parte sustantiva del proceso.
- El costo del cambio no depende de la fase o etapa.
- No introduce funcionalidades antes que sean necesarias.
- El cliente o el usuario se convierte en miembro del equipo. (9)

Lo fundamental en este proceso de desarrollo es lograr la comunicación entre los usuarios y los desarrolladores, la simplicidad al desarrollar, codificar los módulos del sistema y la retroalimentación del equipo de desarrollo, el cliente y los usuarios finales. Este proceso de desarrollo de software tiene como principales normas:

1. Planificación
2. Diseño

3. Codificación
4. Prueba (10)

1.3.3 Proceso Unificado Abierto (*Open Unified Process*, OpenUP)

Open Unified Process (OpenUP) es un proceso de desarrollo unificado que está basado en *Rational Unified Process* (RUP). Mantiene las mismas características de RUP pues está dirigido por casos de uso, centrado en la arquitectura y además es iterativo e incremental. El ciclo de vida de un proyecto según la metodología OpenUP se divide en 4 fases fundamentales:

1. **Concepción**: en esta fase se pretende determinar los objetivos y establecer el alcance del proyecto.
2. **Elaboración**: el propósito de esta fase es establecer la línea base de la arquitectura del sistema y proporcionar una base estable para el desarrollo de la siguiente fase.
3. **Construcción**: esta fase tiene como propósito completar el desarrollo del sistema basado en la arquitectura definida, enfocándose en el diseño, implementación y prueba de las funcionalidades para desarrollar un sistema completo.
4. **Transición**: en esta fase se asegura que el software esté listo para entregarse a los usuarios.

OpenUP propone 6 flujos de Trabajo:

1. **Requerimientos**: en este flujo de trabajo se realizan entrevistas con el cliente para comprender el problema a resolver y se definen los requerimientos.
2. **Análisis y Diseño**: se realiza el diseño de los requisitos que serán después implementados.
3. **Implementación**: en esta disciplina se realiza la implementación del sistema basándose en el diseño realizado.
4. **Prueba**: busca los defectos a los largo del ciclo de vida.
5. **Gestión del Proyecto**: involucra actividades con las que se busca producir un producto que satisfaga las necesidades de los clientes.
6. **Gestión de Configuración y Cambios**: describe cómo controlar los elementos producidos por todos los integrantes del equipo de proyecto en cuanto a: utilización y actualización, control de versiones, etcétera.

OpenUP es una metodología ágil que se aplica a proyectos de corta duración, diseñado para pequeños equipos de trabajo. Además, es un proceso mínimo y suficiente, lo que significa que sólo el contenido fundamental y necesario es incluido para lograr los objetivos del proyecto.

Existe una forma básica y fácil de manejar OpenUP que es OpenUP / Basic, objetivos más pequeños y para equipos interesados en el desarrollo ágil e iterativo. OpenUP/Basic es un FrameWork de procesos de desarrollo de software de código abierto, que permite un abordaje ágil al proceso de desarrollo de software, con sólo proveer un conjunto simplificado de contenidos, fundamentalmente relacionados con orientación, productos de trabajo, roles, y tareas. OpenUP/Basic es un proceso interactivo de desarrollo de software simplificado, completo y extensible. Es un proceso que valora los beneficios de la colaboración y de los involucrados con el resultado del proyecto, por encima de formalidades innecesarias. (11)

Después de haber realizado un estudio sobre las metodologías de desarrollo de software existentes se decidió utilizar la metodología OpenUP pues el tiempo de desarrollo del software es relativamente corto y el equipo de trabajo es pequeño. Además, esta metodología mantiene las principales características de RUP, obviando solamente las partes opcionales que RUP propone, lo que permite que el proceso sea más simple. No se hace uso de la metodología XP ya que la misma exige tener como parte del equipo al usuario final, característica no presente en el proyecto pues el cliente es un agente externo al equipo de desarrollo, además, la calidad del software de acuerdo a esta metodología no se basa en la documentación sino en la comunicación fluida con el cliente ya que después de cada fase este recibe una parte funcional, por lo que estará informado continuamente sobre el proyecto y tendrá la posibilidad de intervenir si el objetivo se desvía de sus necesidades.

1.4 Lenguaje de Modelado

El lenguaje de modelado es un conjunto estándar de símbolos y de formas que facilita la modelación de un diseño de software

1.4.1 Lenguaje Unificado de Modelado (*Unified Modeling Language*, UML)

UML es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de software. (12)

UML es un lenguaje de modelado de sistemas de software muy conocido y utilizado en la actualidad, está compuesto por diversos elementos gráficos que se combinan para conformar diagramas y además, cuenta

con reglas para combinar dichos elementos. UML es un lenguaje que permite la modelación de sistemas con tecnología orientada a objetos, pero es importante aclarar que no es una guía para realizar el análisis y diseño orientado a objetos, es decir, no es un proceso. Además, es importante destacar que un modelo UML describe lo que supuestamente hará un sistema, pero no dice cómo implementar dicho sistema.
Los diagramas más comunes del UML son los siguientes:

1. Diagramas de estructura estática: describen las propiedades estructurales del sistema.
    - Diagrama de clases: conjunto de clases, interfaces y sus colaboraciones.
    - Diagrama de objetos: conjunto de objetos y sus relaciones.
    - Diagrama de casos de uso: conjunto de casos de uso y actores y sus relaciones.
2. Diagramas de comportamiento:
    - Diagramas de interacción (secuencia y colaboración): objetos y sus relaciones, incluyendo los mensajes que pueden ser enviados entre ellos.
    - Diagrama de estados: consta de estados, transiciones, eventos y actividades.
    - Diagrama de actividad: es un tipo especial de diagrama de estados que muestra el flujo de actividades dentro de un sistema.
3. Diagramas de implementación:
    - Diagrama de componentes: organización y las dependencias entre un conjunto de componentes.
    - Diagrama de despliegue: configuración de nodos de procesamiento en tiempo de ejecución y los componentes que residen en ellos.

Se decidió utilizar el Lenguaje Unificado de Modelado (UML) pues en la actualidad es uno de los lenguajes de modelado más conocido y utilizado en todo el mundo. Además, permite visualizar, especificar, construir y documentar un software. Una de las principales razones que justifica la selección de UML es que la metodología de desarrollo seleccionada (OpenUP) lo propone como lenguaje de notación.

1.5 Herramienta CASE

Se puede definir herramienta CASE (*Computer Aided Software Engineering* y en su traducción al español: Ingeniería de Software Asistida por Computadora) a la aplicación de métodos y técnicas a través de las cuales las personas pueden modelar o diseñar sistemas por medio de programas, procedimientos y su respectiva documentación. (13)

Objetivos de las herramientas CASE:

- Mejorar la productividad en el desarrollo y mantenimiento del software.
- Aumentar la calidad del software.
- Mejorar el tiempo y coste de desarrollo y mantenimiento de los sistemas informáticos.
- Mejorar la planificación de un proyecto.
- Ayudar a la reutilización del software, portabilidad y estandarización de la documentación.
- Gestionar las fases de desarrollo del software con una misma herramienta.

Ejemplos de herramientas CASE:

- Umbrello
- ArgoUML
- Gaphor
- System Architect
- Rational Rose
- Visual Paradigm

1.5.1 Rational Rose

Rational Rose es la herramienta CASE desarrollada por los creadores de UML (Booch, Rumbaugh y Jacobson), que cubre todo el ciclo de vida de un proyecto: concepción y elaboración del modelo, construcción de los componentes, transición a los usuarios. Permite especificar, analizar, diseñar el sistema antes de codificarlo. Rational Rose establece una trazabilidad real entre el modelo (análisis y diseño) y el código ejecutable; facilita el desarrollo de un proceso cooperativo en el que todos los agentes tienen sus propias vistas de información (vista de Casos de Uso, vista Lógica, vista de Componentes y vista de Despliegue), pero comparten un mismo modelo a lo largo de todo el ciclo de vida del proyecto. (14)

1.5.2 Visual Paradigm for UML

Visual Paradigm for UML es una potente herramienta que permite visualizar y diseñar elementos de software y soporta el ciclo de vida completo del desarrollo de software. Una de sus ventajas sobre las

demás herramientas CASE es su condición multiplataforma en su edición Community, por lo que tiene la capacidad de ejecutarse sobre diferentes Sistemas Operativos. Es fácil de usar y presenta una agradable interfaz para interactuar con el usuario. Se integra con el Visio para importar imágenes del mismo para realizar los diagramas de despliegue y genera documentación para el proyecto en HTML, MS Word y PDF. Además, importa y exporta diagramas en XML y como imágenes (ya sea con extensiones jpg o png). Otra característica importante es que puede generar código y realizar ingeniería inversa a diferentes lenguajes de programación como son: Java, C++, CORBA IDL, PHP, XML Schema y ADA.

Facilita la interoperabilidad con otras herramientas CASE como el Rational Rose y se integra con las siguientes herramientas Java:

- Eclipse/IBM WebSphere
- Jbuilder
- NetBeans IDE
- Oracle Jdeveloper
- BEA Weblogic.

La herramienta CASE seleccionada fue Visual Paradigm for UML ya que permite el diseño del producto de forma rápida y con calidad, nos brinda facilidades para el diseño de los diagramas, así como su documentación. La Universidad cuenta con la licencia para su uso y además tiene la característica de ser multiplataforma. No se seleccionó la herramienta Rational Rose debido a que no se cuenta con la licencia para su uso.

1.6 Conclusiones del capítulo

Después de realizar un estudio sobre las metodologías y tecnologías actuales en el mundo necesarias para el desarrollo de la herramienta y teniendo en cuenta principalmente las exigencias y necesidades del cliente se decide optar por: Matlab como software matemático para el desarrollo de la herramienta por ser muy potente y debido al interés del cliente, OpenUP como metodología de desarrollo de software ya que es una metodología ágil para proyectos de corta duración diseñada para pequeños equipos de trabajo, utilizando como lenguaje de modelado UML pues permite visualizar los artefactos del sistema. Además, se hace uso de la herramienta CASE Visual Paradigm for UML porque presenta facilidad de uso frente a las demás y por su característica de ser multiplataforma.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

Introducción

En este capítulo se describen los conceptos de mayor importancia en el entorno donde estará el sistema. Se elabora una lista de requerimientos funcionales y no funcionales que se deben tener en cuenta para la implementación de la herramienta. Se identifican los actores, casos de uso y las relaciones existentes entre ellos. Además, se aborda acerca de cómo debe funcionar el sistema y se hace una descripción general de las actividades que serán objeto de automatización.

2.1 Modelo de Dominio

El modelo del dominio es un subconjunto del modelo de negocio y se realiza cuando no están claros los procesos o cuando no se identifican claramente los actores y trabajadores del negocio. Un Modelo del Dominio captura los tipos más importantes de objetos que existen, o los eventos que suceden en el entorno donde estará el sistema, se identifican conceptos, se definen estos conceptos y se unen o relacionan en un diagrama de clases UML.

El objetivo fundamental de este modelo es comprender y describir los conceptos más importantes dentro del contexto del sistema. (15) El modelo de dominio es una representación visual del entorno real del proyecto.

2.1.1 Descripción del Modelo de Dominio

El Centro de Neurociencias de Cuba es el encargado de la investigación, la producción y la aplicación de tecnologías avanzadas para el diagnóstico y tratamiento de enfermedades del cerebro, órgano más complejo del cuerpo humano. En este centro se desarrolla el proyecto MCHC, en el cual los especialistas que pertenecen al proyecto realizan estudios sobre información que se encuentra almacenada en una base de datos y en ficheros EEG. A partir del resultado extraído generan un reporte, el cual es utilizado para realizar posteriormente un estudio estadístico. Este estudio es de gran importancia para los especialistas del centro científico ya que permite realizar diagnósticos, hacer comparaciones entre diferentes estudios y contribuir al tratamiento precoz de enfermedades.

- **Especialistas:** personas que pertenecen al proyecto MCHC del Centro de Neurociencias.
- **BD MCHC:** base de datos del proyecto MCHC donde se almacena toda la información de las personas a las cuales se les realiza el estudio del mapeo cerebral.

- **Fichero EEG:** fichero que contiene la información resultante del electroencefalograma realizado a las personas sometidas al estudio.
- **Reporte:** fichero que almacena el resultado después de realizar determinada consulta, ya sea a la base de datos o a ficheros EEG.
- **Estudio Estadístico:** estudio que realizan los especialistas a partir del reporte generado.

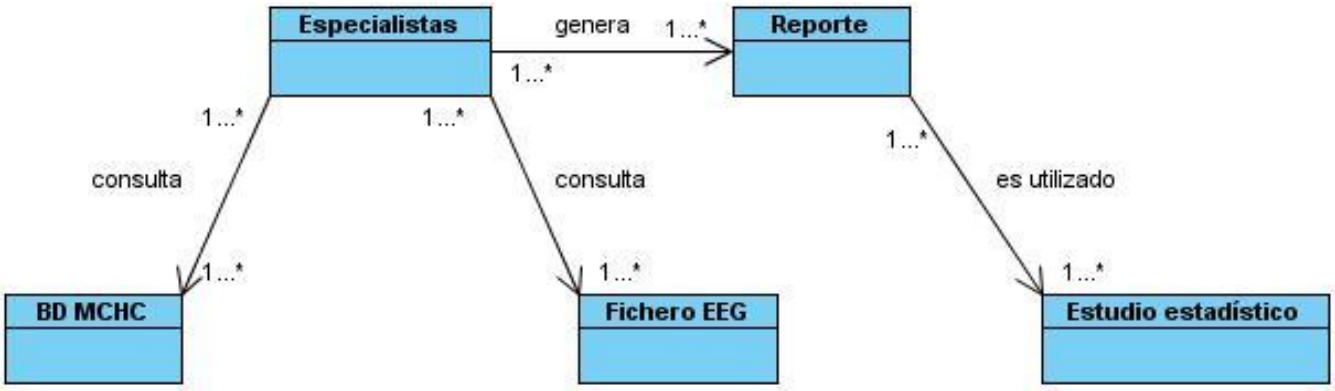


Figura 1: Modelo de Dominio

2.2 Especificación de los Requerimientos del sistema

El Instituto de Ingenieros Electricistas y Electrónicos (IEEE) define un requerimiento como:

1. Una condición o capacidad que un usuario necesita para resolver un problema o lograr un objetivo.
2. Una condición o capacidad que debe tener un sistema o un componente de un sistema para satisfacer un contrato, una norma, una especificación u otro documento formal.
3. Una representación en forma de documento de una condición o capacidad como las expresadas en 1) o en 2). (16)

2.2.1 Requerimientos Funcionales

Los requerimientos funcionales (RF) son capacidades o condiciones que el sistema debe cumplir. Además, se mantienen invariables sin importar con qué propiedades o cualidades se relacionen. Los requerimientos funcionales definen las funciones que el sistema será capaz de realizar. En la herramienta a desarrollar se identificaron los siguientes requisitos funcionales:

RF1	Registrar datos para la conexión.
RF2	Autenticar usuario.
RF3	Generar reporte de la base de datos.
RF4	Añadir condición para consultar la base de datos.
RF5	Eliminar condición para consultar la base de datos.
RF6	Salvar reporte de la base de datos.
RF7	Generar reporte de los ficheros EEG.
RF8	Añadir condición para consultar los ficheros EEG.
RF9	Eliminar condición para consultar los ficheros EEG.
RF10	Salvar reporte de los ficheros EEG.

2.2.2 Requerimientos No Funcionales

Los requerimientos no funcionales (RNF) son propiedades o cualidades que el producto debe tener. Estas propiedades o cualidades se refieren a las características que hacen al producto atractivo, usable, rápido o confiable. Por lo general los requerimientos no funcionales son fundamentales en el éxito del producto; normalmente están vinculados a los requerimientos funcionales, es decir, una vez que se conoce lo que el sistema debe hacer se puede determinar cómo ha de comportarse, qué cualidades o propiedades debe tener.

Los requerimientos no funcionales son importantes para que clientes y usuarios puedan valorar las características no funcionales del producto, pues si se conoce que el mismo cumple con toda la funcionalidad requerida, las propiedades no funcionales, como cuán usable, seguro, conveniente y agradable, pueden marcar la diferencia entre un producto bien aceptado y uno con poca aceptación.

Para la herramienta propuesta se definen los siguientes requisitos no funcionales:

Requerimientos de Software

- Sistema Operativo Windows 95 o versiones superiores
- Sistema Operativo Linux V.4 o superior, Red Hat Enterprise, Fedora Core 4 o superior, Devian 4.0 o superior.
- Debe estar instalado el software matemático Matlab 7.6 o versión superior.

Requerimientos de Hardware

- Procesador Pentium 4, Celeron, Xeon, Core, AMD64.
- 512 Megabytes (MB) de memoria RAM (Random Access Memory, por sus siglas en inglés) como mínimo (recomendado 1024MB).
- 5 Gigabytes (GB) de espacio en disco duro.

Restricciones del diseño y la implementación

- Se hace uso de la herramienta matemática Matlab.
- El lenguaje de programación que será usado para la implementación es el lenguaje m.
- Para la Base de Datos del sistema se usará PostgreSQL.

Requerimientos de Apariencia e Interfaz externa

- La herramienta debe estar diseñada con una interfaz agradable, fácil de usar y entender, de forma tal que el usuario no tenga dificultad alguna para su uso y alcanzar así una mejor aceptación del producto.

Requerimientos de Seguridad

- La información manejada por el sistema está protegida de acceso no autorizado y se encuentra disponible sólo para los usuarios que pertenezcan al proyecto y estén registrados en la base de datos. A través de la autenticación correcta se podrá obtener la información solicitada.

Requerimientos de Confiabilidad

- El sistema debe ser exacto en la información que le suministra al usuario para evitar cualquier tipo de error.

Requerimientos de Usabilidad

- La herramienta propuesta podrá ser usada por cualquier persona autorizada que pertenezca al equipo del proyecto MCHC en CNEURO y que posea conocimientos básicos en el manejo de una computadora. Debe ser una herramienta sencilla, manejable y fácil de usar para el usuario.

Requerimientos Legales

- Se utilizan herramientas de software libre y herramientas de las cuales se posee licencia para su uso y distribución.

Requerimientos de Soporte

- Se garantiza la instalación de la herramienta en el Centro de Neurociencias y se impartirá un breve adiestramiento a los especialistas que serán los futuros usuarios de la misma. Además, se le dará seguimiento al funcionamiento de la herramienta durante un período de tiempo determinado.

Requerimientos de Portabilidad

- La herramienta será multiplataforma, pues podrá ser instalada y disponer de la misma en diferentes sistemas operativos tales como Windows y Linux, sin necesidad de requerir otros recursos adicionales a los previstos.

Requerimientos Políticos-culturales

- La herramienta propuesta empleará el idioma español debido a que es el idioma rector en Cuba.
- La herramienta contará con logotipos e imágenes que se encuentren relacionadas con el carácter científico y profesional del Centro de Neurociencias.

2.3 Modelo de Casos de Uso del Sistema

El modelo de casos de uso del sistema representa las relaciones existentes entre actores y casos de uso. Los actores son terceros fuera del sistema que interactúan con él (puede ser cualquier persona, individuo, grupo, entidad, organización, máquina o sistema de información externo; con los que el sistema interactúa) y los casos de uso son fragmentos de funcionalidad que el sistema ofrece para aportar un resultado de valor para sus actores.

Se identificó como actores del sistema al Usuario y Especialista.

Actores	Descripción
Usuario	Cualquier persona que trabaje en el Centro de Neurociencias.
Especialista	Es el rol que engloba a todos los miembros del proyecto MCHC del Centro de Neurociencias. Es la persona que va a hacer uso del sistema para consultar la base de datos y los ficheros EEG con el objetivo de obtener determinada información útil para estudios posteriores.

2.3.1 Diagrama de Casos de Uso del Sistema

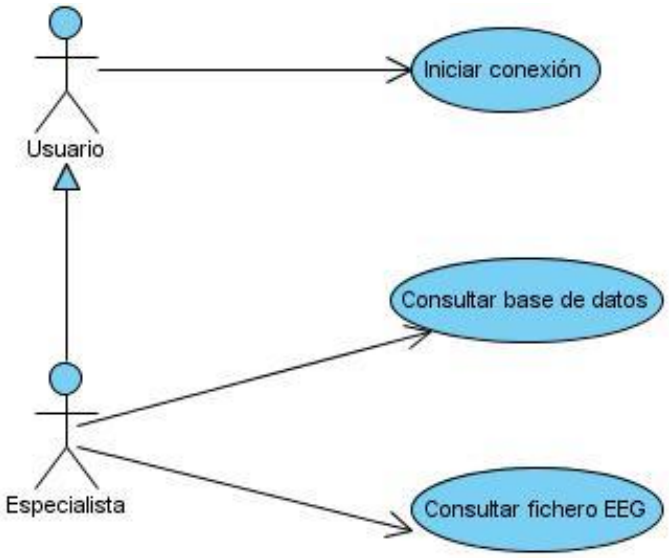


Figura 2: Diagrama de Casos de Uso del Sistema

2.3.2 Patrones de Casos de Uso

Los patrones de Casos de Uso permiten reflejar con precisión los requerimientos funcionales del sistema. Estos han mostrado ser la solución adoptada en la comunidad del desarrollo de software. Se presentan a modo de herramientas que permiten resolver los problemas que se les planteen a los desarrolladores de una forma ágil y sistemática. Con la utilización de patrones de Casos de Uso se pueden lograr mejores resultados de forma más rápida. (17)

Existen diversos patrones de Casos de Uso, entre ellos se utilizó el patrón Múltiples Actores, específicamente la categoría Roles Comunes. Este patrón se utilizó debido a que dos actores (Usuario y Especialista) inicializan un mismo caso de uso, por tal motivo se representa un actor (Usuario), que inicialice el caso de uso común (Iniciar conexión) y de él hereda el otro actor (Especialista).

2.3.3 Descripción textual de los Casos de Usos del Sistema

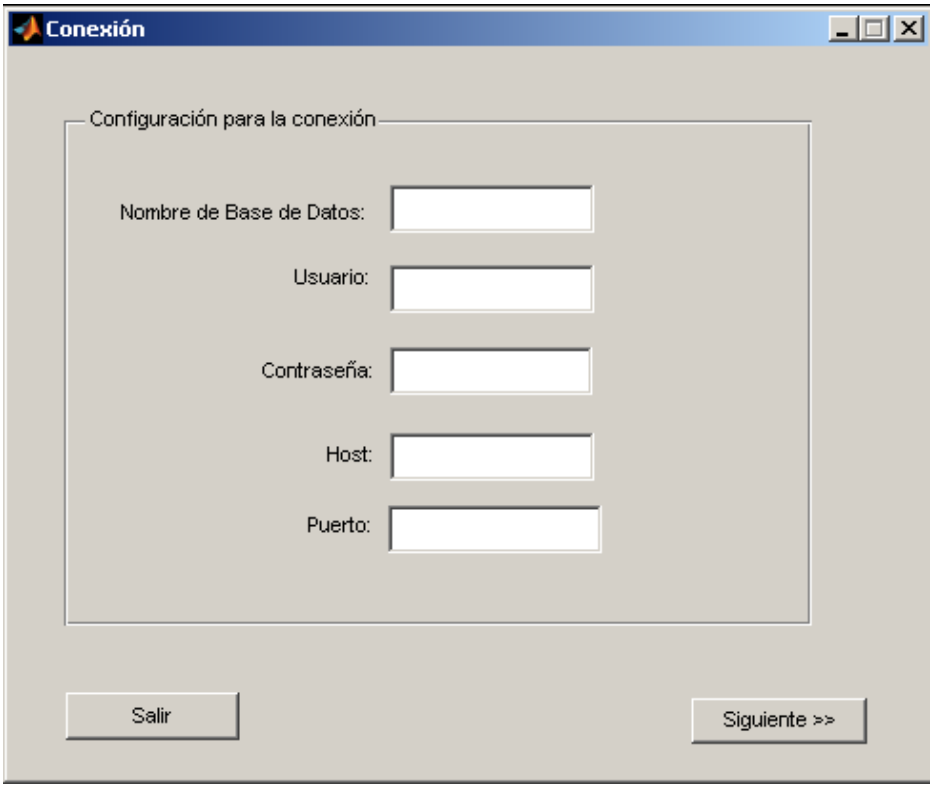
Descripción textual del Caso de Uso Iniciar conexión

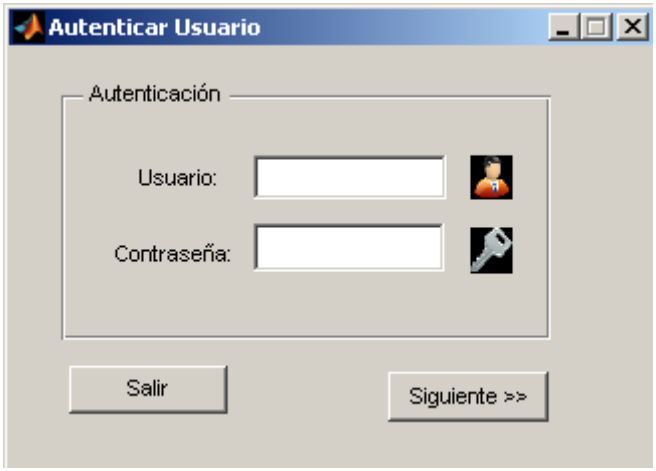
Caso de Uso	Iniciar conexión
Actor:	Usuario

Resumen:	El Caso de Uso (CU) se inicia cuando el usuario introduce los datos necesarios para lograr la conexión a la base de datos de MCHC, una vez lograda la conexión se muestra la ventana para la autenticación del usuario, donde debe introducir los datos necesarios (usuario y contraseña), después el sistema verifica los datos y si están correctos habilita la ventana para que el usuario decida qué operación realizar, finalizando así el CU.
Precondiciones:	-
Referencias:	RF1, RF2
Prioridad:	Crítico

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El usuario introduce el nombre, usuario, contraseña, puerto y host correspondientes a la Base de Datos a la cual se desea conectar y oprime el botón “Siguiente”.	2. El sistema comprueba que no exista ningún campo vacío.
	3. El sistema verifica que los datos de la conexión a la base de datos introducidos por el usuario sean correctos.
	4. El sistema muestra la interfaz correspondiente para la autenticación del usuario.
5. El usuario introduce su usuario y contraseña y oprime el botón “Siguiente”.	6. El sistema comprueba que los campos usuario y contraseña no estén vacíos.
	7. El sistema comprueba que los datos introducidos por el usuario sean correctos.
	8. El sistema muestra la ventana para seleccionar

	la operación a realizar y finaliza así el CU.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
1.1 Si presiona el botón “Salir”, sale del sistema, y finaliza así el CU.	3.1 Si existe algún campo vacío el sistema muestra un mensaje de error.
5.1 Si presiona el botón “Salir”, sale del sistema, y finaliza así el CU.	4.1 Si los datos introducidos son incorrectos el sistema muestra un mensaje de error.
	7.1 Si los campos usuario y contraseña están vacíos el sistema muestra un mensaje de error.
	8.1 Si los datos introducidos no son correctos el sistema muestra un mensaje de error.
Prototipo de Interfaz de Usuario	





Poscondiciones	El usuario queda autenticado en el sistema y se permite el acceso a la ventana donde se escoge la operación a realizar.

Descripción textual del Caso de Uso Consultar base de datos

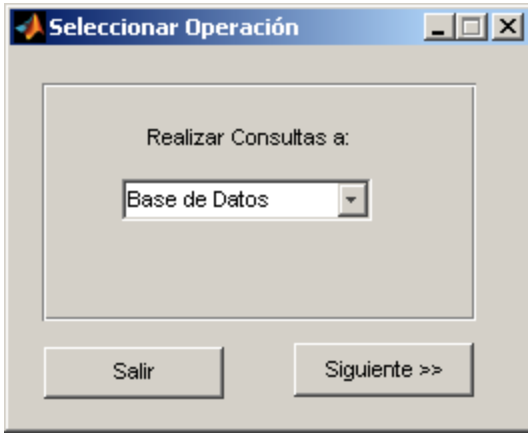
Caso de Uso	Consultar base de datos
Actor:	Especialista
Resumen:	El Caso de Uso (CU) se inicia cuando el especialista decide consultar la base de datos, luego introduce los datos correspondientes para realizar la consulta, el sistema genera un reporte y da la posibilidad de salvarlo o realizar otra consulta.
Precondiciones:	El especialista debe estar autenticado.
Referencias:	RF3, RF4, RF5, RF6
Prioridad:	Crítico

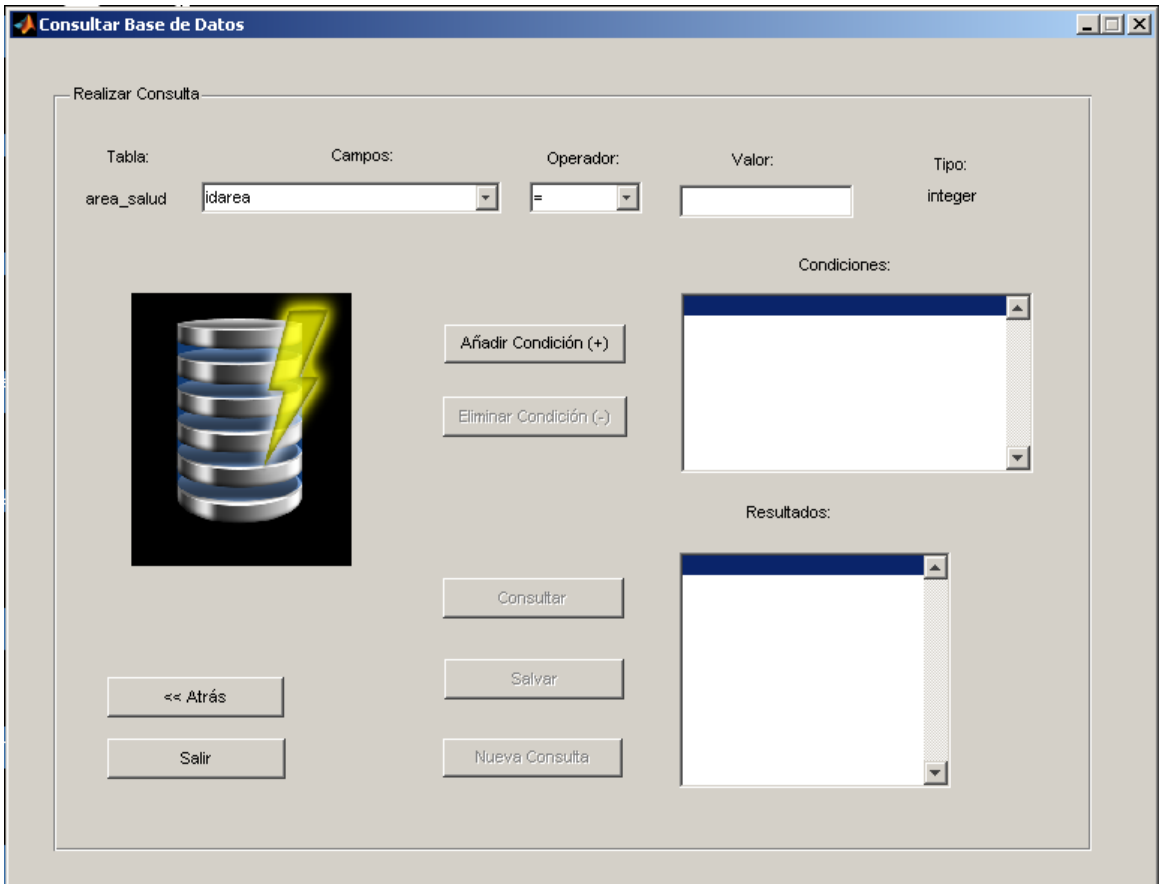
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El especialista indica realizar la consulta a la base de datos.	2. El sistema muestra la interfaz correspondiente, donde carga los campos de las tablas de la base

	de datos en el componente “Campos”, así como el tipo de dato y el nombre de las tablas.
3. El especialista introduce el dato en el campo valor para añadir la condición que se cumpla en la consulta y presiona el botón “Añadir Condición”.	4. El sistema comprueba que el campo no esté vacío.
	5. El sistema verifica que el dato introducido sea del tipo o formato correspondiente.
	6. El sistema verifica que no existe condición igual a la que se desea añadir.
	7. El sistema añade la condición.
8. El especialista selecciona una de las siguientes opciones: ✓ Añadir condición ✓ Eliminar condición ✓ Consultar ✓ Nueva consulta	9. El sistema según la operación seleccionada por el especialista realiza una de las siguientes acciones: ✓ Si el especialista selecciona la operación “Añadir Condición” ir a la actividad 3 del Flujo Normal de Eventos. ✓ Si el especialista selecciona la operación “Eliminar Condición” ir a la sección: Eliminar condición. ✓ Si el especialista selecciona la operación “Consultar” ir a la sección: Consultar. ✓ Si el especialista selecciona la operación “Nueva consulta” el sistema elimina los datos existentes y se ejecuta la actividad 3 del Flujo Normal de Eventos.

Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
1.1 Si presiona el botón “Salir”, sale del sistema y termina así el CU.	5.1 Si el campo está vacío el sistema muestra un mensaje de error.
3.1 Si presiona el botón “Salir”, sale del sistema y termina así el CU.	6.1 Si el dato introducido no tiene el tipo o formato correspondiente el sistema muestra un mensaje de error.
8.1 Si presiona el botón “Salir”, sale del sistema y termina así el CU.	7.1 Si existe una condición igual a la que se desea añadir el sistema muestra un mensaje de error.
Sección 1 “Eliminar condición”	
Acción del Actor	**Respuesta del Sistema**
1. El especialista selecciona la condición que desea eliminar.	2. El sistema elimina la condición seleccionada.
	3. El sistema verifica si existe alguna condición.
	4. Ir a la actividad 8 del Flujo Normal de Eventos.
Flujos Alternos “Sección Eliminar condición”	
Acción del Actor	**Respuesta del Sistema**
	4.1 Si no existe condición el sistema deshabilita los botones “Eliminar Condición” y “Consultar”.
	4.2 Ir a la actividad 3 del Flujo Normal de Eventos.
Sección 2 “Consultar”	
Acción del Actor	**Respuesta del Sistema**
1. El especialista presiona el botón “Consultar”.	2. El sistema realiza la consulta con las condiciones añadidas.
	3. El sistema verifica si la consulta devuelve

	resultados.
	4. El sistema muestra el reporte de la consulta y da la posibilidad de salvar el reporte, eliminar y añadir condiciones y crear nueva consulta.
5. El especialista selecciona una de las siguientes opciones: ➢ Añadir condición ➢ Eliminar condición ➢ Nueva consulta ➢ Salvar	6. El sistema según la operación seleccionada por el especialista realiza una de las siguientes acciones: ➢ Si el especialista selecciona la operación "Añadir Condición" ir a la actividad 3 del Flujo Normal de Eventos. ➢ Si el especialista selecciona la operación "Eliminar condición" se ejecuta la sección "Eliminar condición". ➢ Si el especialista selecciona la operación "Nueva consulta" el sistema elimina los datos existentes y se ejecuta la actividad 3 del Flujo Normal de Eventos. ➢ Si el especialista selecciona la operación "Salvar" se ejecuta la sección "Salvar reporte".
Flujos Alternos "Sección Consultar"	
Acción del Actor	**Respuesta del Sistema**
	4.1 Si la consulta no devuelve resultado el sistema muestra un mensaje para informar que no se encontraron datos para esa consulta.
Sección 3 "Salvar Reporte"	
Acción del Actor	**Respuesta del Sistema**
	1. El sistema muestra una ventana para seleccionar

	el directorio donde guardar el fichero del reporte.
2. El especialista selecciona la dirección donde será guardado el fichero del reporte.	3. El sistema guarda el reporte en la dirección especificada y muestra un mensaje para informar que la acción se realizó satisfactoriamente.
	4. Ir a la actividad 8 del Flujo Normal de Eventos
Prototipo de Interfaz de Usuario	
	



Poscondiciones	Se guarda el fichero con el reporte de la consulta a la base de datos.

Descripción textual del Caso de Uso Consultar fichero EEG (Ver Anexo 1:)

2.4 Conclusiones del capítulo

En este capítulo se describieron, a través del Modelo de Dominio, los conceptos más importantes dentro del contexto del sistema. Además, se identificaron 10 requisitos funcionales, agrupados en 3 casos de uso, se especificaron los requisitos no funcionales para el correcto funcionamiento del sistema y para detallar las características del producto. Se definieron los actores, los casos de uso y la relación existente entre ellos en el diagrama de casos de uso del sistema. Por último, se describieron detalladamente los casos de uso del sistema para obtener así un mayor entendimiento de los requerimientos funcionales previamente establecidos.

CAPÍTULO 3: DISEÑO DEL SISTEMA

Introducción

En este capítulo mediante el diseño se hace una especificación de los requerimientos con el propósito de describir cómo se debe implementar el sistema. Se aborda sobre los patrones arquitectónicos y de diseño que están presentes en la herramienta. Se realiza el diagrama de clases del diseño, a través del cual se muestra la estructura estática del sistema y además se describen las clases más relevantes. Para modelar los aspectos dinámicos del sistema se realizan los diagramas de interacción donde se representan las clases y sus relaciones, incluyendo los mensajes que se pueden enviar entre ellas.

3.1 Patrones

Los patrones son parejas de problema/solución con un nombre, que codifican buenos principios y sugerencias relacionados generalmente con la asignación de responsabilidades. (18)

Los patrones ayudan a construir un software sobre la experiencia colectiva de ingenieros experimentados. Estos capturan la experiencia existente y exitosa en el desarrollo de software, y ayudan a promover las buenas prácticas de diseño. Cada patrón aborda un problema específico en el diseño o implementación de un sistema de software. Existen diversos tipos de patrones, entre ellos se encuentran:

- **Patrones de arquitectura**: Expresan un esquema organizativo estructural fundamental para sistemas de software.
- **Patrones de diseño**: Expresan esquemas para definir estructuras de diseño con las cuales se construyen sistemas de software. (19)

3.1.1 Patrones Arquitectónicos

Un patrón arquitectónico define la estructura básica de una aplicación, provee un subconjunto de subsistemas predefinidos, incluyendo reglas, lineamientos para conectarlos y pautas para su organización y constituye una plantilla de construcción. La abstracción más alta en cuanto a soluciones a través de patrones, se obtiene a través del uso de patrones de arquitectura. (20)

El patrón arquitectónico que se utiliza es el patrón **Tres Capas**, debido a que los componentes de una capa sólo pueden acceder a componentes en capas inmediatamente inferiores. Este patrón simplifica la comprensión y organización del desarrollo del sistema, reduciendo las dependencias de forma que las capas más bajas no son consistentes a ningún cambio de las superiores. El patrón **Tres Capas** se divide en:

1. **Capa de presentación:** esta capa es la que permite la interacción del usuario con el sistema a través de interfaces. Esta capa se comunica únicamente con la capa lógica.
2. **Capa lógica:** se denomina capa lógica porque es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de base de datos y a los ficheros EEG la recuperación de datos.
3. **Capa de datos:** es donde residen los datos, esta capa recibe solicitudes de recuperación de información desde la capa lógica.

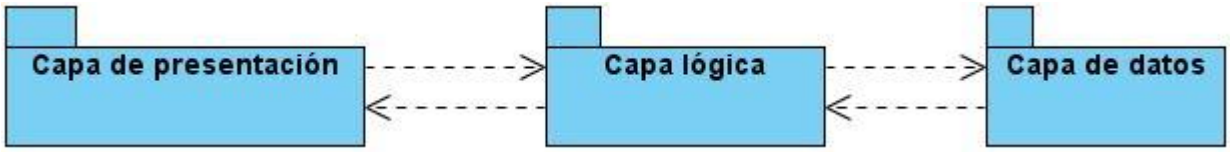


Figura 3: Patrón Tres Capas

En la capa de presentación se encuentran las clases Herramienta_MCHC, Conexión, Autenticar_Usuario, Seleccionar_Operación, Datos y Datos_Fichero.
En la capa lógica se encuentran las clases Control y Control_Fichero.
En la capa de datos se encuentra la clase Acceso_Datos, la base de datos MCHC y los ficheros EEG.

3.1.2 Patrones de Diseño

Un patrón de diseño es una solución simple y estándar para problemas específicos y comunes del diseño orientado a objetos. Es una solución basada en la experiencia y que se ha demostrado que funciona. (21)

Ventajas

- Los patrones de diseño proponen una forma de reutilizar la experiencia de los desarrolladores, para ello clasifica y describe formas de solucionar problemas que ocurren de forma frecuente en el desarrollo del software.
- Están basados en la recopilación del conocimiento de los expertos en desarrollo de software.
- Es una experiencia real, probada y que funciona. Ayuda a no cometer los mismos errores.

GRASP es un acrónimo que significa **G**eneral **R**esponsibility **A**signment **S**oftware **P**atterns (patrones generales de software para asignar responsabilidades). Los patrones **GRASP** describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. El nombre se eligió para indicar la importancia de captar (grasping) estos principios, si se quiere diseñar eficazmente el software orientado a objetos. (18)

De los patrones GRASP se utilizaron los siguientes:

Experto: este patrón plantea que se debe asignar una responsabilidad al experto en información, o sea, a la clase que cuenta con la información necesaria para cumplir la responsabilidad.

Beneficios del patrón Experto

- Se conserva el encapsulamiento, ya que los objetos se valen de su propia información para hacer lo que se les pide. Esto posibilita tener sistemas de fácil mantenimiento.
- El comportamiento se distribuye entre las clases que cuentan con la información requerida para cumplir con la responsabilidad asignada.

Una de las clases donde se evidencia la utilización del patrón Experto es la clase Acceso_Datos pues esta es la clase que contiene la información necesaria (atributos, constructor y métodos) para cumplir con sus responsabilidades (conectarse a la base de datos y ejecutar consultas).

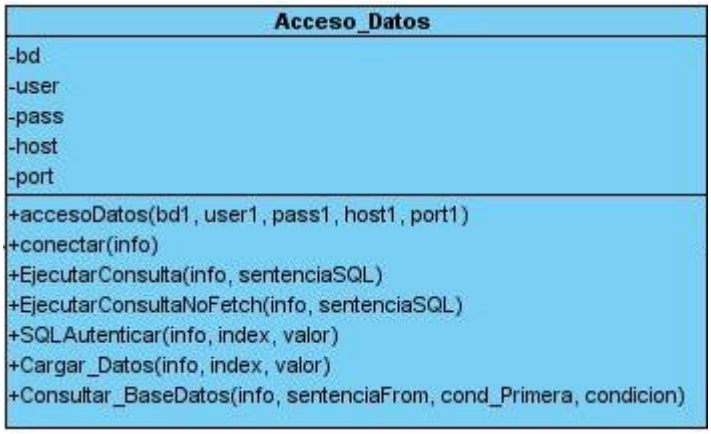


Figura 4: Clase Acceso_Datos

Creador: este patrón plantea que se debe asignar a una clase X la responsabilidad de crear una instancia de una clase Y. La creación de objetos es una de las actividades más frecuentes en un sistema orientado

a objetos. Este patrón es el encargado de guiar la asignación de responsabilidades relacionadas con la creación de objetos.

Beneficios del patrón Creador

- Se crean menos dependencias y existe mayor posibilidad de reutilización de código.

En las clases Datos_Ficheros y Conexión se evidencia la utilización del patrón Creador pues estas crean un objeto de la clase Control_Fichero y Control respectivamente con el objetivo de poder acceder a sus métodos.

Bajo acoplamiento: este patrón plantea que se debe asignar las responsabilidades de forma tal que las clases se comuniquen con el menor número de clases que sea posible. Una clase con bajo acoplamiento no depende de muchas otras.

Beneficios del patrón Bajo acoplamiento

- Fáciles de entender por separado.
- Fáciles de reutilizar.

Alta cohesión: este patrón plantea que se debe asignar una responsabilidad de modo que la cohesión siga siendo alta (una clase tiene responsabilidades moderadas). Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas, que no realicen un trabajo enorme. Una clase con baja cohesión hace muchas cosas no afines o un trabajo excesivo.

Beneficios del patrón Alta cohesión

- Mejoran la claridad y la facilidad con que se entiende el diseño.
- Se simplifican el mantenimiento y las mejoras en funcionalidad.

3.2 Modelo de Diseño

El modelo de diseño es un modelo de objetos que describe la realización de los casos de uso, y sirve como una abstracción del modelo de implementación y el código fuente.

3.2.1 Diagrama de clases del diseño

Los diagramas de clases del diseño muestran la estructura estática del sistema y en él se representan las clases, su estructura interna (atributos y métodos) y sus relaciones con otras clases.

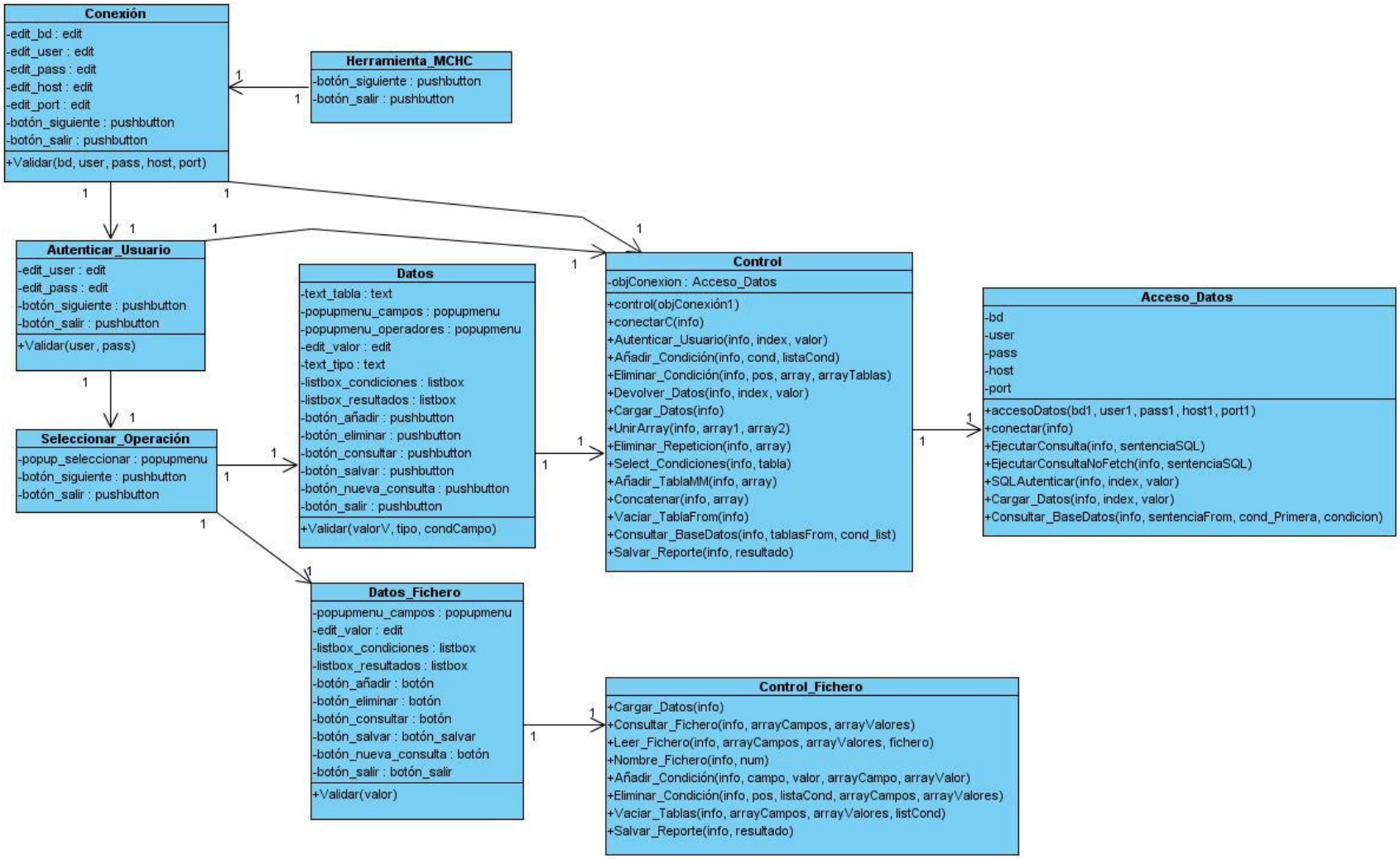


Figura 5: Diagrama de clases del diseño

Descripción de las clases más relevantes del diseño

Una clase del diseño es similar a una clase en la implementación del sistema pues tiene propiedades (atributos), comportamientos (métodos) y relaciones con otras clases.

Nombre:	Acceso_Datos
Tipo de clase:	Controladora
Atributos:	bd, user, pass, host, port
Principales Responsabilidades	
Nombre:	accesoDatos (bd1,user1,pass1,host1,port1)
Descripción:	Constructor de la clase, es el encargado de inicializar los atributos
Nombre:	Conectar (info)
Descripción:	Se encarga de conectarse a la base de datos.
Nombre:	EjecutarConsulta(info, sentenciaSQL)
Descripción:	Se encarga de ejecutar una consulta a la base de datos y devuelve los datos obtenidos.

Nombre:	**Control_Fichero**
Tipo de clase:	Controladora
Atributos:	**-**
Principales Responsabilidades	
Nombre:	Cargar_Datos(info)
Descripción:	Se encarga de devolver los campos que componen la estructura de los ficheros.
Nombre:	Consultar_Fichero(info,arrayCampos, arrayValores)
Descripción:	Se encarga de consultar todos los ficheros y devuelve un reporte con

	todos los sujetos que cumplen con las condiciones especificadas.
Nombre:	Leer_Fichero (info, arrayCampos, arrayValores, fichero)
Descripción:	Se encarga de buscar si el fichero cumple con las condiciones especificadas y en caso positivo devuelve el código del sujeto.
Nombre:	Añadir_Condición (info, campo, valor, arrayCampo, arrayValor)
Descripción:	Se encarga de añadir una nueva condición para realizar la consulta a la base de datos.
Nombre:	Eliminar_Condición(info, pos, listaCond, arrayCampos, arrayValores)
Descripción:	Se encarga de eliminar la condición seleccionada.
Nombre:	Salvar_Reporte (info, resultado)
Descripción:	Se encarga de salvar el reporte generado en la dirección especificada después de consultar los ficheros.

Nombre:	Control
Tipo de clase:	Controladora
Atributos:	objConexión
Principales Responsabilidades	
Nombre:	Control (objConexión1)
Descripción:	Constructor de la clase, es el encargado de inicializar el atributo.
Nombre:	Autenticar_Usuario (info, index, valor)
Descripción:	Se encarga de comprobar que el usuario y la contraseña sean correctos según la información existente en la base de datos.
Nombre:	Añadir_Condición (info, cond, listaCond)
Descripción:	Se encarga de añadir una nueva condición a la lista de condiciones para realizar la consulta a la base de datos.

Nombre:	Eliminar_Condición (info, pos, array, arrayTablas)
Descripción:	Se encarga de eliminar la condición seleccionada de la lista de condiciones.
Nombre:	Cargar_Datos(info)
Descripción:	Se encarga de extraer toda la información necesaria de la base de datos (nombre de las tablas, los campos y el tipo de dato de cada campo) para mostrarla en la interfaz.
Nombre:	Consultar_BaseDatos (info, tablasFrom, cond_list)
Descripción:	Se encarga de consultar la base de datos según las condiciones especificadas y devuelve un reporte con todos los sujetos que cumplen con las condiciones especificadas.
Nombre:	Salvar_Reporte (info, resultado)
Descripción:	Se encarga de salvar el reporte generado en la dirección especificada después de consultar la base de datos.

3.2.2 Diagramas de interacción

Los diagramas de interacción se utilizan para modelar los aspectos dinámicos de un sistema y muestran un conjunto de objetos y sus relaciones, incluyendo los mensajes que se pueden enviar entre ellos.

Entre los diagramas de interacción se encuentran los diagramas de secuencia.

Diagramas de secuencia

Un diagrama de secuencia destaca el orden temporal de los mensajes. Gráficamente, un diagrama de secuencia es una tabla que representa objetos que se ubican a lo largo del eje X, y mensajes ordenados según se suceden en el tiempo, a lo largo del eje Y.

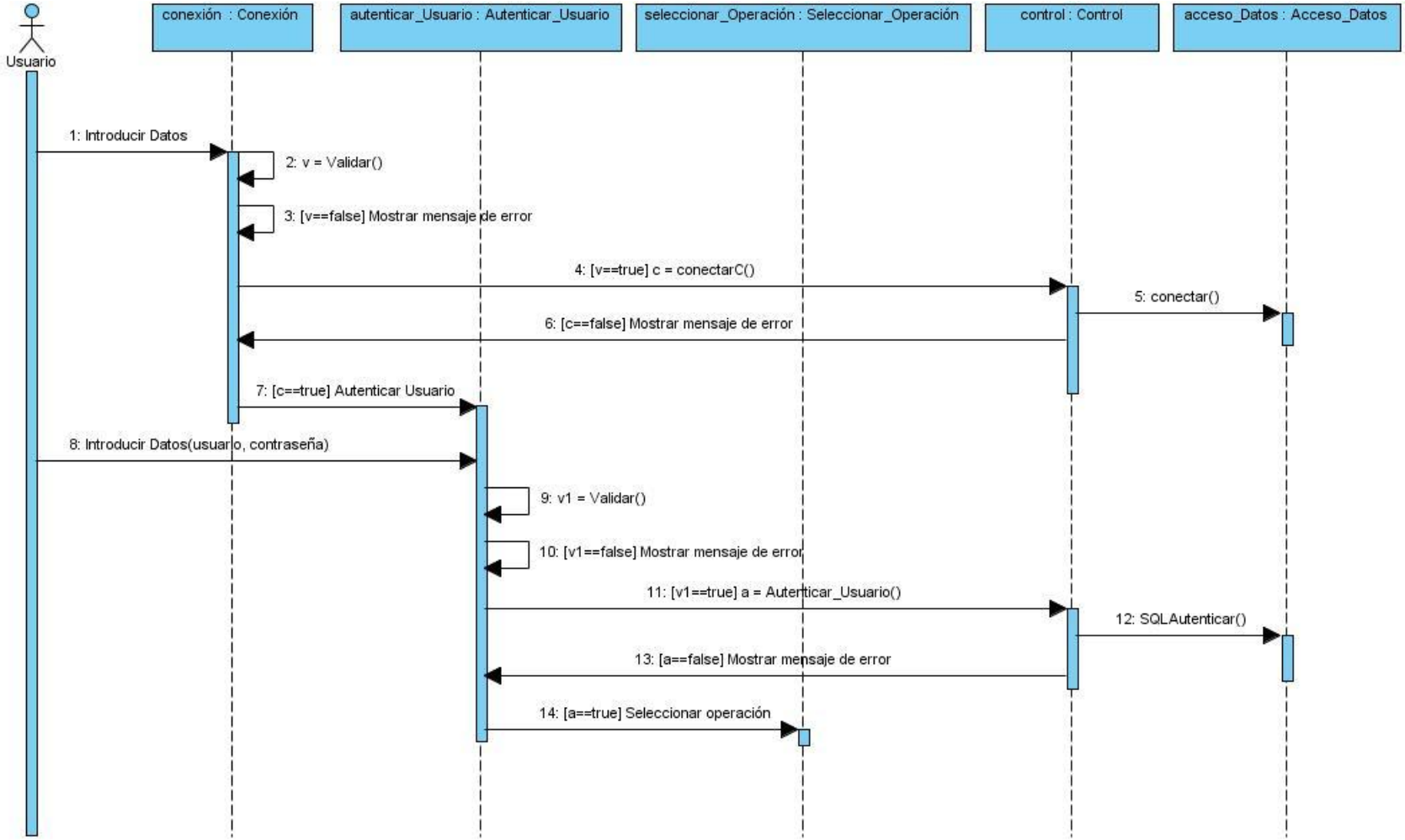


Figura 6: Diagrama de secuencia del Caso de Uso "Iniciar conexión"

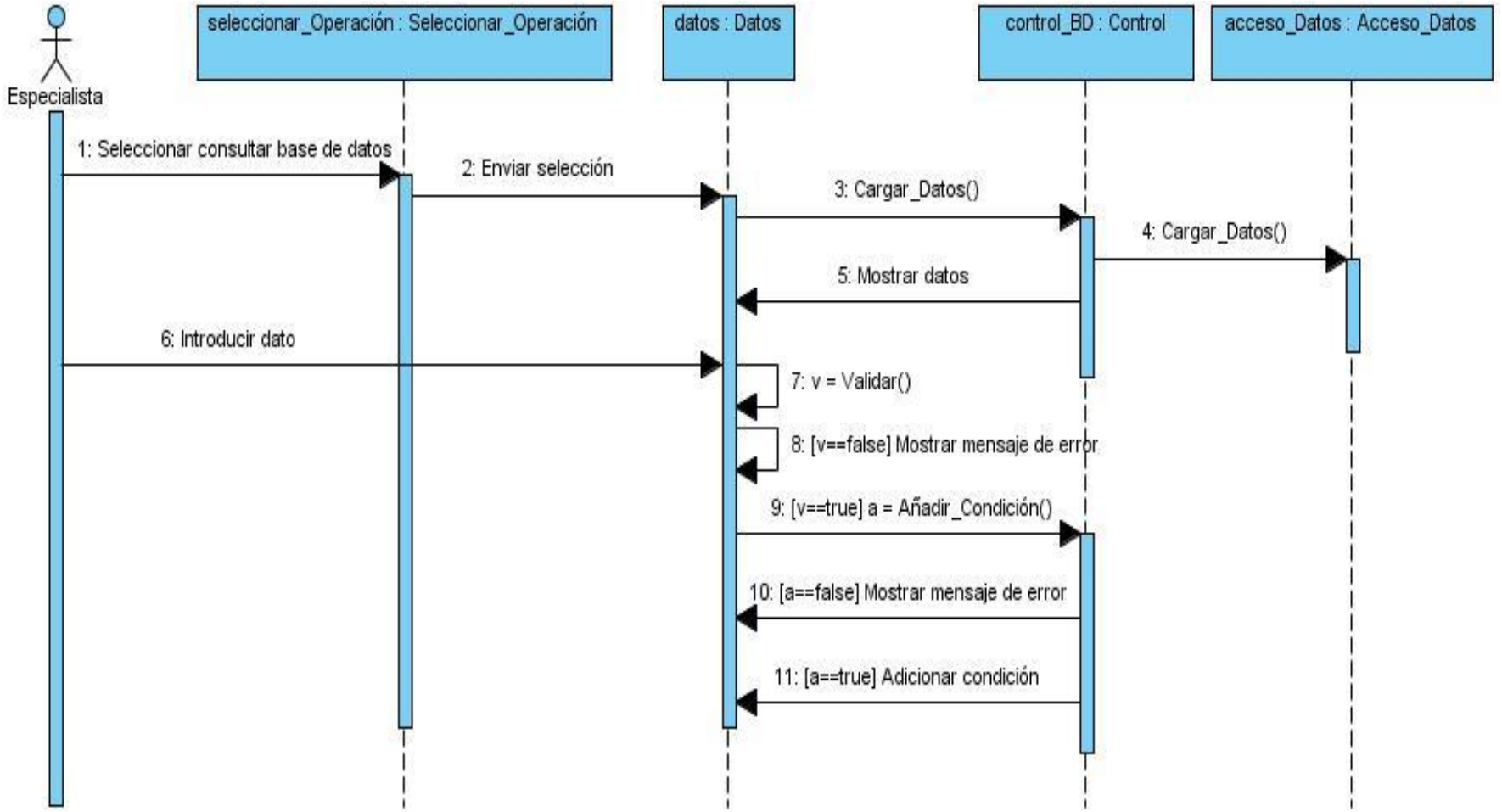


Figura 7: Diagrama de secuencia del Caso de Uso "Consultar base de datos" (Curso normal de eventos)

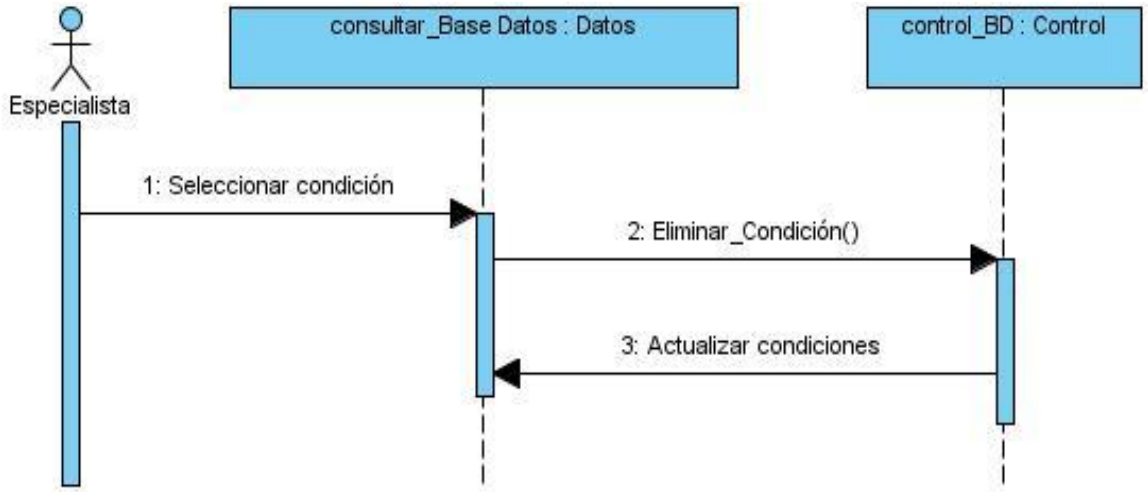


Figura 8: Diagrama de secuencia del Caso de Uso "Consultar base de datos" (Sección "Eliminar condición")

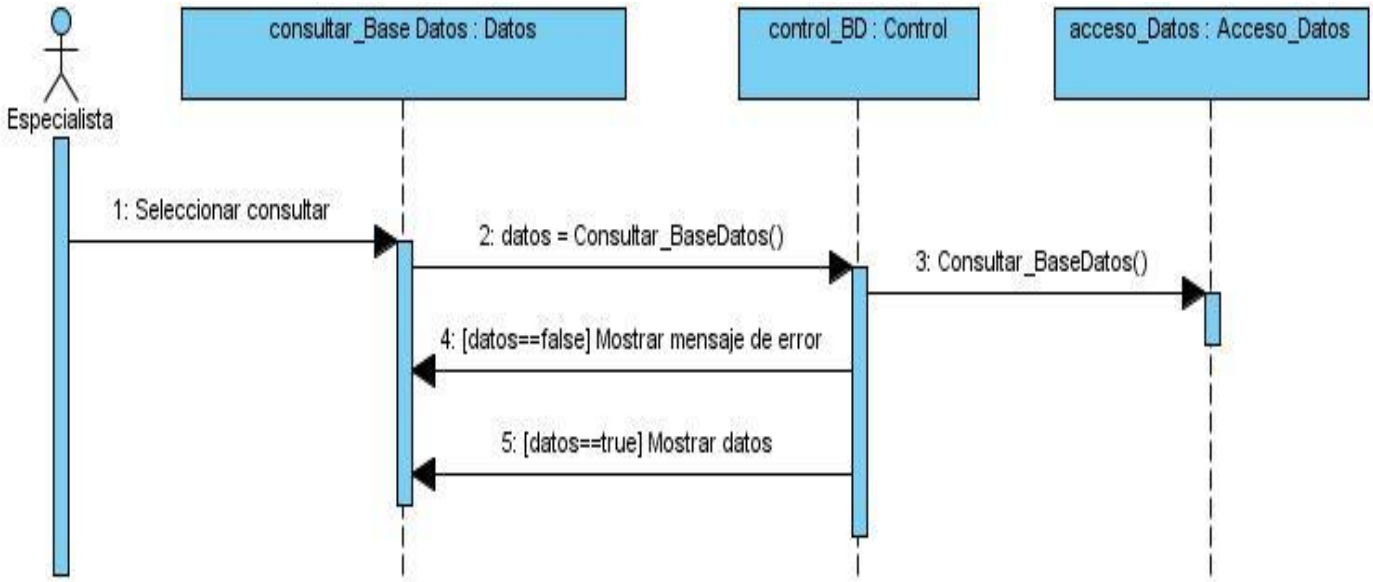


Figura 9: Diagrama de secuencia del Caso de Uso "Consultar base de datos" (Sección "Consultar")

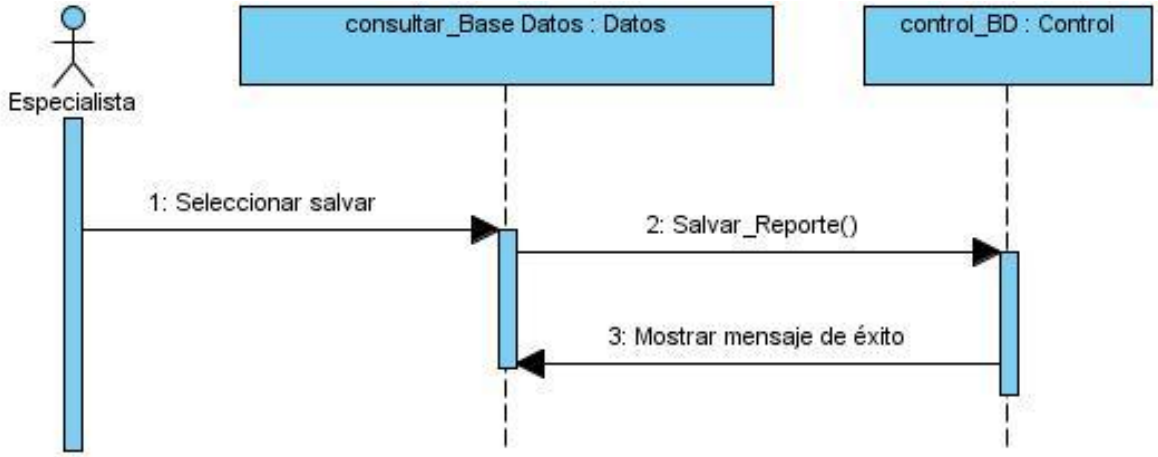


Figura 10: Diagrama de secuencia del Caso de Uso "Consultar base de datos" (Sección "Salvar Reporte")

Diagramas de Secuencia del Caso de Uso Consultar fichero EEG (Ver Anexo 2:)

3.3 Conclusiones del capítulo

En este capítulo se planteó como patrón arquitectónico, el patrón Tres capas y como patrones de diseño para asignar responsabilidades a objetos se usaron los patrones GRASP, entre ellos el Experto, Creador, Bajo acoplamiento y Alta cohesión. Además, se realizó el modelo de diseño con el propósito de describir cómo se debe implementar el sistema y se representó la estructura estática y dinámica del sistema, a través del diagrama de clases del diseño y de los diagramas de interacción (secuencia) respectivamente. También se describieron las clases más relevantes con el propósito de lograr un mejor entendimiento de las mismas.

CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBA DEL SISTEMA

Introducción

En este capítulo se realiza el diagrama de componentes para estructurar el modelo de implementación y se muestra la ubicación física de los nodos de procesamiento a través del diagrama de despliegue. Además, se realizan pruebas de caja negra para comprobar el correcto funcionamiento del software, se verifica que la entrada de datos se acepte de forma adecuada y que se produzca un resultado correcto.

4.1 Modelo de Implementación

En la implementación se comienza con el resultado del diseño y se implementa el sistema en términos de componentes. El modelo de implementación describe los componentes a construir y su organización en nodos físicos en los que funcionará el sistema. Está compuesto por los diagramas de despliegue y componentes.

4.1.1 Diagrama de componentes

El diagrama de componentes es usado para estructurar el modelo de implementación y muestra un conjunto de elementos tales como componentes y sus relaciones. Un componente es una parte del sistema que encapsula implementación y un conjunto de interfaces. Es un elemento de implementación que representa algo físico.

Existen diferentes tipos de componentes, como son:

- Ejecutable: es un programa que se puede ejecutar en un nodo.
- Biblioteca: es una biblioteca de objetos estática o dinámica.
- Tabla: es una tabla de una base de datos.
- Archivo: es un fichero que contiene código fuente o datos.
- Documento: es un documento.

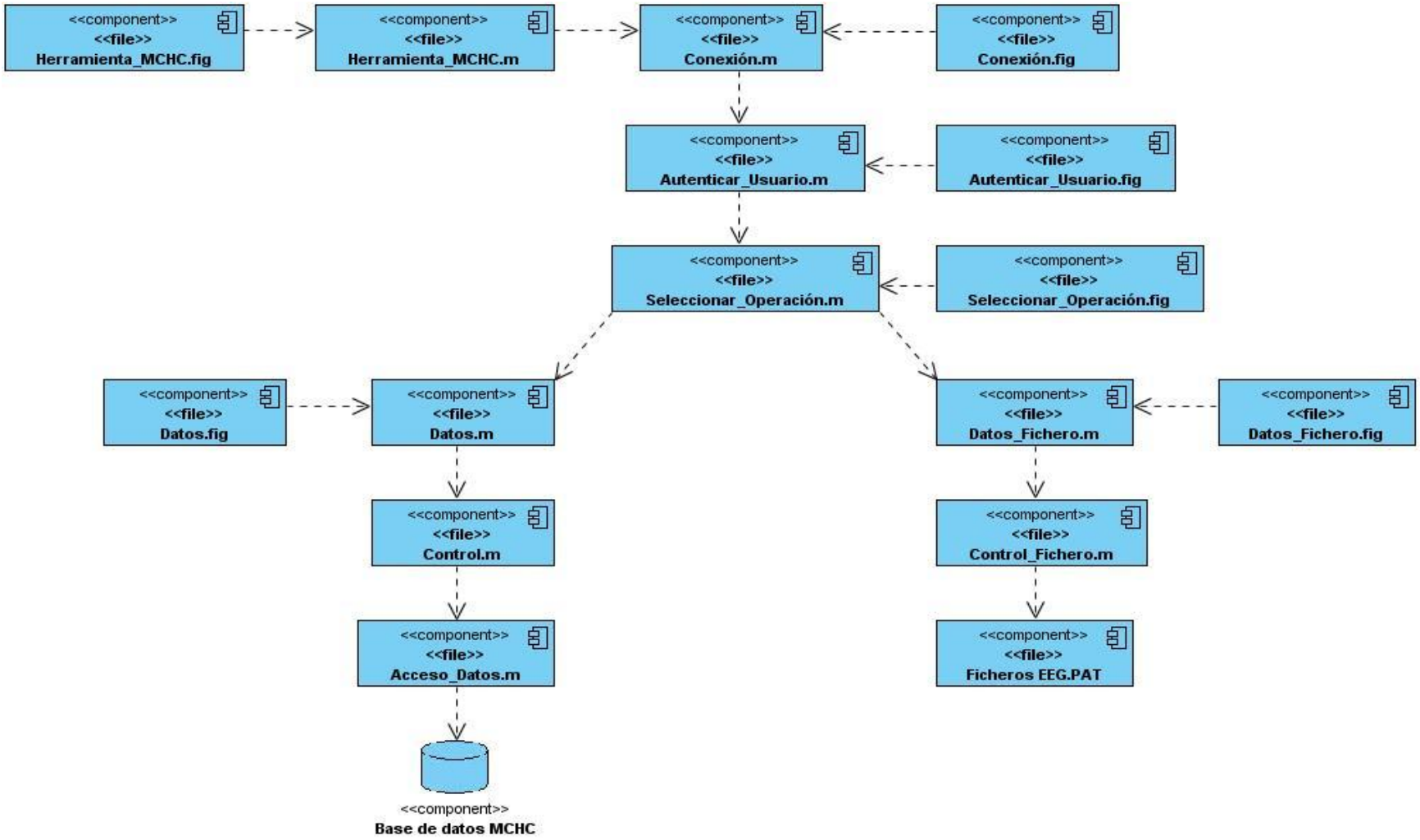


Figura 11: Diagrama de componentes

4.1.2 Diagrama de despliegue

El diagrama de despliegue muestra la configuración de los nodos de procesamiento y los vínculos de comunicación entre ellos. Contiene nodos, dispositivos y conectores.

- Nodos: elementos de procesamiento con al menos un procesador.
- Dispositivos: nodos sin capacidad de procesamiento.
- Conectores: expresa el tipo de conector o protocolo utilizado entre el resto de los elementos del modelo.

Figura 12: Diagrama de despliegue

4.2 Código fuente

El código fuente de un sistema informático es un conjunto de líneas de texto que son las instrucciones que debe seguir la computadora para así poder ejecutar las funcionalidades requeridas por el software. El código fuente varía según el lenguaje de programación, y debe ser traducido al lenguaje máquina para que sea ejecutado. A continuación se presenta el código fuente de la clase Acceso_Datos.

```
classdef Acceso_Datos
   properties
       bd;
       user;
       pass;
       host;
       port;
   end % Fin de las propiedades    (end properties)
   methods
       function cxn = Acceso_Datos(bd1,user1,pass1,host1,port1)
           cxn.bd = bd1;
```

```
            cxn.user = user1;
            cxn.pass = pass1;
            cxn.host = host1;
            cxn.port = port1;
        end
        function conn = conectar(info)
          url = strcat ('jdbc:postgresql://',info.host,':',info.port,'/',info.bd);
          conn = database(info.bd, info.user, info.pass, 'org.postgresql.Driver', url);
        end
        function consulta = EjecutarConsulta(info,sentenciaSQL)
            conn = conectar(info);
            consulta = fetch(exec(conn, sentenciaSQL));
            close(conn);
        end
    end % Fin de los Métodos
end % Fin de Clase
```

4.3 Prueba

Las pruebas son una actividad en la cual un sistema o componente es ejecutado bajo condiciones o requerimientos específicos, los resultados son observados y registrados, donde se evalúa algún aspecto del sistema o componente.

Las pruebas tienen como objetivos:

- Encontrar los defectos que puedan afectar la calidad del software.
- Validar que el software trabaje como fue diseñado.
- Validar los requisitos que debe cumplir el software.
- Validar que los requisitos fueron implementados correctamente.

4.3.1 Pruebas de Caja Negra

La prueba de caja negra se refiere a las pruebas que se llevan a cabo sobre la interfaz del software. Los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce un resultado correcto.

Pruebas de la caja negra:

- Verifican las especificaciones funcionales y no consideran la estructura interna del programa.
- Es hecha sin el conocimiento interno del producto.
- No validan funciones ocultas (por ejemplo funciones implementadas pero no descritas en las especificaciones funcionales del diseño) por tanto los errores asociados a ellas no serán encontrados.

4.3.2 Casos de Prueba

Un caso de prueba es un conjunto de entradas de pruebas, condiciones de ejecución y resultados esperados desarrollados para cumplir un objetivo en particular o una función esperada.

V: válido

I: inválido

Caso de prueba para el CU Iniciar conexión

Escenario	Variable1 Nombre de la BD	Variable2 Usuario de la BD	Variable3 Contraseña de la BD	Variable4 Puerto de la BD	Variable5 Host de la BD	Respuesta del Sistema	Resultado de la prueba	Flujo central
Registrar datos para la conexión correctamente.	V	V	V	V	V	El sistema se conecta a la base de datos y muestra la interfaz correspondiente para la autenticación del usuario.	El sistema se conecta a la base de datos y muestra la interfaz correspondiente para la autenticación del usuario.	1. Introducir los valores correctamente en todos los campos. 2. Presionar el botón **Siguiente**.
Registrar datos para la conexión incorrectamente.	I	I	I	I	I	El sistema muestra un mensaje de error.	El sistema muestra un mensaje de error.	1. Introducir los valores incorrectos en los campos. 2. Presionar el botón **Siguiente**.
	V	I	I	I	I			
	I	V	I	I	I			
	I	I	V	I	I			
	I	I	I	V	I			
	I	I	I	I	V			

	V	V	I	I	I			
	V	V	V	I	I			
	V	V	V	V	I			
	I	I	I	V	V			
	I	I	V	V	V			
	I	V	V	V	V			

Escenario	Variable1 Usuario	Variable2 Contraseña	Respuesta del Sistema	Resultado de la prueba	Flujo central
Autenticar usuario correctamente	V	V	El usuario queda autenticado en el sistema y se permite el acceso a la ventana donde se escoge la operación a realizar.	El usuario queda autenticado en el sistema y se permite el acceso a la ventana donde se escoge la operación a realizar.	1. Introducir los valores correctamente los campos. 2. Presionar el botón **Siguiente**.
Autenticar usuario incorrectamente	I	I	El sistema muestra un mensaje de error.	El sistema muestra un mensaje de error.	1. Introducir los valores incorrectos en los campos. 2. Presionar el botón **Siguiente.**
	V	I			
	I	V			

Caso de prueba del CU Consultar base de datos

Escenario	Variable1 Valor	Respuesta del Sistema	Resultado de la prueba	Flujo central
Adicionar condición correctamente.	V	Se añade la condición a la lista de condiciones que será usada para realizar la consulta a la base de datos.	Se añade la condición a la lista de condiciones que será usada para realizar la consulta a la base de datos.	1. Introducir el valor correcto en el campo. 2. Presionar el botón **Adicionar Condición**.

Adicionar condición incorrectamente	I	No se añade la condición a la lista de condiciones y se muestra un mensaje de error.	No se añade la condición a la lista de condiciones y se muestra un mensaje de error.	1. Introducir el valor incorrecto en el campo. 2. Presionar el botón **Adicionar Condición**.

Escenario	**Variable1**	**Respuesta del Sistema**	**Resultado de la prueba**	**Flujo central**
Eliminar condición.	-	Se elimina la condición seleccionada de la lista de condiciones que será usada para realizar la consulta a la base de datos.	Se elimina la condición seleccionada de la lista de condiciones que será usada para realizar la consulta a la base de datos.	1. Seleccionar en el listbox **Condiciones** la condición que se desea eliminar. 2. Presionar el botón **Eliminar Condición**.

Escenario	**Variable1**	**Respuesta del Sistema**	**Resultado de la prueba**	**Flujo central**
Consultar la base de datos.	-	Se realiza la consulta a la base de datos y se muestra el resultado en el listbox **Resultados**.	Se realiza la consulta a la base de datos y se muestra el resultado en el listbox **Resultados**.	1. Presionar el botón **Consultar**.

Escenario	**Variable1**	**Respuesta del Sistema**	**Resultado de la prueba**	**Flujo central**
Salvar el reporte de la consulta.	-	Se abre una ventana para especificar la dirección donde se	Se abre una ventana para especificar la dirección donde se	1. Presionar el botón **Salvar**. 2. Indicar la dirección donde

		desea salvar el reporte de la consulta.	desea salvar el reporte de la consulta.	se desea guardar el reporte. 3. Presionar el botón **Aceptar**.

Caso de prueba del CU Consultar fichero EEG (Ver Anexo 3:)

4.4 Conclusiones del capítulo

En este capítulo se realizó el modelo de implementación con el propósito de mostrar los componentes del sistema y sus relaciones a través del diagrama de componentes. Mediante el diagrama de despliegue se mostró la configuración de los nodos de procesamiento en tiempo de ejecución y los vínculos de comunicación entre ellos. Además, se realizó el modelo de prueba para validar que los requisitos fueron implementados correctamente y se describen los casos de prueba que verifican el correcto funcionamiento del sistema a través de las pruebas de caja negra.

CONCLUSIONES

- La correcta especificación de los requerimientos y la realización del diseño del sistema permitieron implementar la herramienta informática, utilizando el lenguaje de programación de Matlab (lenguaje m).
- Se probó el correcto funcionamiento de la herramienta, a través de las pruebas de caja negra realizadas mediante los casos de prueba.
- Como resultado del trabajo realizado se obtuvo una herramienta en Matlab, capaz de obtener información de la base de datos y de los ficheros EEG del proyecto Mapeo Cerebral Humano Cubano.

RECOMENDACIONES

A partir de las experiencias obtenidas en el desarrollo del trabajo y con el fin de lograr un aprovechamiento óptimo del resultado alcanzado se recomienda:

- Incorporar a la herramienta la funcionalidad para desarrollar un análisis estadístico de la información extraída de la base de datos y de los ficheros EEG del proyecto MCHC.
- Actualizar la herramienta con las nuevas funcionalidades que vayan surgiendo durante el avance del proyecto MCHC en el Centro de Neurociencias.
- Vincular la herramienta informática a la aplicación principal del proyecto para centralizar todos los módulos.

REFERENCIAS BIBLIOGRÁFICAS

1. **Llanes, Clara Carmen.** Mapeo cerebral humano: un reto del siglo XXI. [En línea] 19 de Septiembre de 2005. [Citado el: 18 de Noviembre de 2009.] http://www.voltairenet.org/article128202.html.

2. ICBM: International Consortium for Brain Mapping. [En línea] 2005. [Citado el: 18 de Noviembre de 2009.] http://www.loni.ucla.edu/ICBM/About.

3. El Electroencefalograma (EEG). [En línea] Health Topics, 6 de Abril de 2004. [Citado el: 28 de Noviembre de 2009.] http://www.healthsystem.virginia.edu/UVAHealth/peds_neuro_sp/eeg.cfm.

4. **J Vicente.** EUITT. [En línea] [Citado el: 18 de Diciembre de 2009.] http://www3.euitt.upm.es/taee/Congresosv2/2006/papers/2006S1B03.pdf.

5. Addlink. [En línea] 2009. [Citado el: 18 de Diciembre de 2009.] http://www.addlink.es/productos.asp?pid=1.

6. **Pijeira Cabrera, Héctor Gama.** [En línea] 19 de Enero de 2009. [Citado el: 18 de Diciembre de 2009.] http://gama.uc3m.es/index.php/the-news/64-matlab.pdf.

7. **Martínez, Rafael.** PosgreSQL-es.org. [En línea] 2009. [Citado el: 15 de Enero de 2010.] http://www.postgresql-es.org/sobre_postgresql.

8. JDBC API (Java Database Connectivity API). [En línea] [Citado el: 15 de Enero de 2010.] http://profesores.elo.utfsm.cl/~agv/elo330/2s09/lectures/JDBC/JDBC.html.

9. Extreme Programming. [En línea] 28 de Septiembre de 2009. [Citado el: 20 de Enero de 2010.] http://www.extremeprogramming.org/.

10. Extreme Programming. [En línea] [Citado el: 20 de Enero de 2010.] http://www.extremeprogramming.org/rules.html.

11. CBASQA. [En línea] 02 de Septiembre de 2008. [Citado el: 20 de Enero de 2010.] http://cbasqa.wordpress.com/2008/09/02/proceso-de-desarrollo-openup/.

12. **Booch, G, Rumbaugh, J y Jacobson, I.** *El Lenguaje Unificado de Modelado.* 2000.

13. **Périssé, M C.** *Una Metodología Simplificada.* Buenos Aires : s.n., 2001.

14. vico.org. [En línea] 2002. [Citado el: 22 de Enero de 2010.] http://www.vico.org/FormMentorOutsourcingUML.pdf.

15. **Ortiz, Kadir Hector.** eumed. [En línea] [Citado el: 15 de Marzo de 2010.] http://www.eumed.net/libros/2009c/583/Representacion%20del%20Modelo%20de%20Objetos%20de%20Dominio.htm.

16. **Durán Toro, Amador.** Departamento de Lenguajes y Sistemas Informáticos. Univ de Sevilla. [En línea] Marzo de 2006. [Citado el: 22 de Enero de 2010.] http://www.lsi.us.es/docencia/get.php?id=2005.

17. **Övergaard, Gunnar y Palmkvist, Karin.** *Use Cases Patterns and Blueprints.* 2004. ISBN/0-13-145134-0.

18. **Larman, Craig.** *UML y Patrones.* México : Prentice Hall, 1999. ISBN/970-1 7-0261-1.

19. **Welicki, León.** Patrones y Antipatrones: una Introducción - Parte I. [En línea] Microsoft Corporation, 2010 . [Citado el: 5 de Abril de 2010.] http://msdn.microsoft.com/es-es/library/bb972242.aspx#M2.

20. **Buschmann, F y Wiley, John.** *Pattern-Oriented Software Architecture.* 1996. ISBN/0471958697.

21. **Gracia, Joaquin.** Ingenieros Software. [En línea] 27 de Mayo de 2005. [Citado el: 5 de Abril de 2010.] http://www.ingenierosoftware.com/analisisydiseno/patrones-diseno.php.

BIBLIOGRAFÍA

Addlink. [En línea] 2009. [Citado el: 18 de Diciembre de 2009.] http://www.addlink.es/productos.asp?pid=1.

Avedaño, Bárbara. Cerebro humano: Trazos sobre una carta inconclusa. [En línea] Bohemia, 24 de Enero de 2008. [Citado el: 18 de Noviembre de 2009.] http://www.bohemia.cu/2008/01/23/cienciatecnologia/1-mapa.html.

Booch, G, Rumbaugh, J y Jacobson, I. *El Lenguaje Unificado de Modelado.* 2000.

Buschmann, F y Wiley, John. *Pattern-Oriented Software Architecture.* 1996. ISBN/0471958697.

CBASQA. [En línea] 02 de Septiembre de 2008. [Citado el: 20 de Enero de 2010.] http://cbasqa.wordpress.com/2008/09/02/proceso-de-desarrollo-openup/.

Chirino Cruz, Yodaysis. Revista Ciencias. [En línea] 29 de Abril de 2008. [Citado el: 5 de Diciembre de 2009.] http://www.revistaciencias.com/publicaciones/EkpAVpAFupTkLakvhJ.php. ISPN/EkpAVpAFupTkLakvhJ.

Ciclo de vida de un proyecto XP. [En línea] [Citado el: 20 de Enero de 2010.] http://oness.sourceforge.net/proyecto/html/ch05s02.html.

Durán Toro, Amador. Departamento de Lenguajes y Sistemas Informáticos. Univ de Sevilla. [En línea] Marzo de 2006. [Citado el: 22 de Enero de 2010.] http://www.lsi.us.es/docencia/get.php?id=2005.

El Electroencefalograma (EEG). [En línea] Health Topics, 6 de Abril de 2004. [Citado el: 28 de Noviembre de 2009.] http://www.healthsystem.virginia.edu/UVAHealth/peds_neuro_sp/eeg.cfm.

[En línea] [Citado el: 20 de Enero de 2010.] http://www.miliuco.net/java/jdbc/jdbc_intro.htm.

Extreme Programming. [En línea] 28 de Septiembre de 2009. [Citado el: 20 de Enero de 2010.] http://www.extremeprogramming.org/.

Extreme Programming. [En línea] [Citado el: 20 de Enero de 2010.] http://www.extremeprogramming.org/rules.html.

Gracia, Joaquin. Ingenieros Software. [En línea] 27 de Mayo de 2005. [Citado el: 5 de Abril de 2010.] http://www.ingenierosoftware.com/analisisydiseno/patrones-diseno.php.

ICBM: International Consortium for Brain Mapping. [En línea] 2005. [Citado el: 18 de Noviembre de 2009.] http://www.loni.ucla.edu/ICBM/About.

JDBC API (Java Database Connectivity API). [En línea] [Citado el: 15 de Enero de 2010.] http://profesores.elo.utfsm.cl/~agv/elo330/2s09/lectures/JDBC/JDBC.html.

J Vicente. EUITT. [En línea] [Citado el: 18 de Diciembre de 2009.] http://www3.euitt.upm.es/taee/Congresosv2/2006/papers/2006S1B03.pdf.

Larman, Craig. *UML y Patrones.* México : Prentice Hall, 1999. ISBN/970-1 7-0261-1.

Llanes, Clara Carmen. Mapeo cerebral humano: un reto del siglo XXI. [En línea] 19 de Septiembre de 2005. [Citado el: 18 de Noviembre de 2009.] http://www.voltairenet.org/article128202.html.

Machorro Reyes, Ricardo Armando. Revista upiicsa. [Online] Agosto 2004. [Cited: Febrero 25, 2010.] http://www.revistaupiicsa.20m.com/Emilia/RevMayAgo04/Machorro1.pdf.

Martínez, Rafael. PosgreSQL-es.org. [En línea] 2009. [Citado el: 15 de Enero de 2010.] http://www.postgresql-es.org/sobre_postgresql.

Matlab Summary and Tutorials. [En línea] [Citado el: 24 de Enero de 2010.] http://www.math.ufl.edu/help/matlab-tutorial/matlab-tutorial.html.

Mendoza Sánchez, María A. Informatizate. [En línea] 07 de Junio de 2004. [Citado el: 20 de Enero de 2010.] http://www.informatizate.net/articulos/metodologias_de_desarrollo_de_software_07062004.html.

Ortiz, Kadir Hector. eumed. [En línea] [Citado el: 15 de Marzo de 2010.] http://www.eumed.net/libros/2009c/583/Representacion%20del%20Modelo%20de%20Objetos%20de%20Dominio.htm.

Övergaard, Gunnar y Palmkvist, Karin. *Use Cases Patterns and Blueprints.* 2004. ISBN/0-13-145134-0.

Périssé, M C. *Una Metodología Simplificada.* Buenos Aires : s.n., 2001.

Pijeira Cabrera, Héctor Gama. [En línea] 19 de Enero de 2009. [Citado el: 18 de Diciembre de 2009.] http://gama.uc3m.es/index.php/the-news/64-matlab.pdf.

Riera, Lilliam. MINREX. [En línea] 21 de Marzo de 2009. [Citado el: 15 de Noviembre de 2009.] http://embacu.cubaminrex.cu/Default.aspx?tabid=11395.

Rodríguez Sotomayor, Daynet. Mundo en Crisis: Cuba traza su Mapa Cerebral Humano. [En línea] 8 de Noviembre de 2009. [Citado el: 18 de Noviembre de 2009.] http://mundo-en-crisis.blogspot.com/2009/11/cuba-traza-su-mapa-cerebral-humano.html.

Sanchez, María A. Mendoza. Informatizate. [En línea] 07 de Junio de 2004. [Citado el: 17 de Diciembre de 2009.] http://www.informatizate.net/articulos/metodologias_de_desarrollo_de_software_07062004.html.

Universidad de Virginia. [En línea] 6 de Abril de 2004. [Citado el: 20 de Noviembre de 2009.] http://www.healthsystem.virginia.edu/UVAHealth/peds_neuro_sp/eeg.cfm.

Vallejo Cuero, Henry Harvey. Patrones de Diseños Orientados a Objeto. [En línea] 2009. [Citado el: 5 de Abril de 2010.] http://www.slideshare.net/2008PA2Info3/patrones-de-diseo-orientados-a-objetos.

vico.org. [En línea] 2002. [Citado el: 22 de Enero de 2010.] http://www.vico.org/FormMentorOutsourcingUML.pdf.

Welicki, León. Patrones y Antipatrones: una Introducción - Parte I. [En línea] Microsoft Corporation, 2010 . [Citado el: 5 de Abril de 2010.] http://msdn.microsoft.com/es-es/library/bb972242.aspx#M2.

ANEXOS

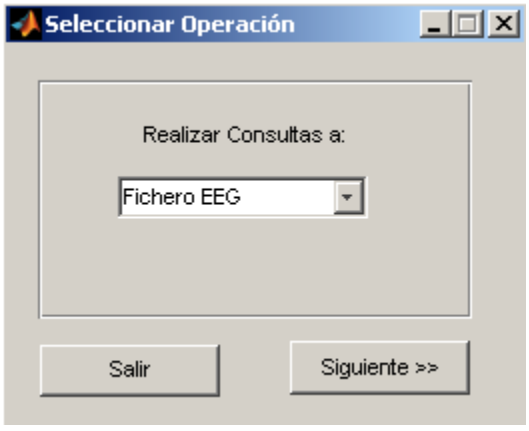
Anexo 1: Descripción textual del Caso de Uso Consultar fichero EEG

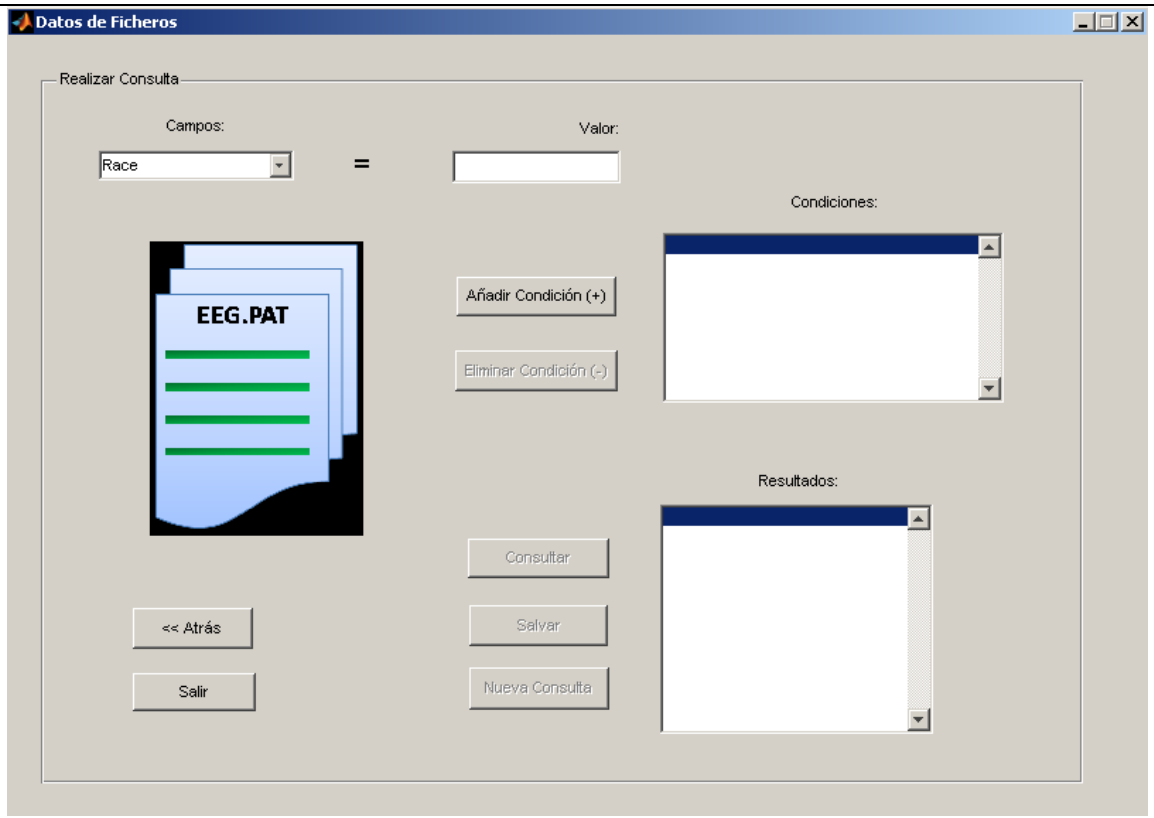
Caso de Uso	Consultar fichero EEG
Actor:	Especialista
Resumen:	El Caso de Uso (CU) se inicia cuando el especialista decide consultar los ficheros EEG, luego introduce los datos correspondientes para realizar la consulta, el sistema genera un reporte y da la posibilidad de salvarlo o realizar otra consulta.
Precondiciones:	El especialista debe estar autenticado.
Referencias:	RF7, RF8, RF9, RF10
Prioridad:	Crítico

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El especialista indica realizar la consulta a los ficheros EEG.	2. El sistema muestra la interfaz correspondiente, donde carga los campos que componen la estructura de los ficheros en el componente “Campos”.
3. El especialista introduce el dato en el campo valor para añadir la condición que se cumpla en la consulta y presiona el botón “Añadir Condición”.	4. El sistema comprueba que el campo no esté vacío.
	5. El sistema verifica que no existe condición igual a la que se desea añadir.
	6. El sistema añade la condición.
7. El especialista selecciona una de	8. El sistema según la operación seleccionada por

las siguientes opciones: ➢ Añadir condición ➢ Eliminar condición ➢ Consultar ➢ Nueva consulta	el especialista realiza una de las siguientes acciones: ➢ Si el especialista selecciona la operación “Añadir Condición” ir a la actividad 3 del Flujo Normal de Eventos. ➢ Si el especialista selecciona la operación “Eliminar Condición” ir a la sección: Eliminar condición. ➢ Si el especialista selecciona la operación “Consultar” ir a la sección: Consultar. ➢ Si el especialista selecciona la operación “Nueva consulta” el sistema elimina los datos existentes y se ejecuta la actividad 3 del Flujo Normal de Eventos.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
1.1 Si presiona el botón “Salir”, sale del sistema y termina así el CU.	5.1 Si el campo está vacío el sistema muestra un mensaje de error.
3.1 Si presiona el botón “Salir”, sale del sistema y termina así el CU.	6.1 Si existe una condición igual a la que se desea añadir el sistema muestra un mensaje de error.
8.1 Si presiona el botón “Salir”, sale del sistema y termina así el CU.	
Sección 1 “Eliminar condición”	
Acción del Actor	**Respuesta del Sistema**
5. El especialista selecciona la condición que desea eliminar.	6. El sistema elimina la condición seleccionada.
	7. El sistema verifica si existe alguna condición.

	8. Ir a la actividad 8 del Flujo Normal de Eventos.
Flujos Alternos "Sección Eliminar condición"	
Acción del Actor	**Respuesta del Sistema**
	4.1 Si no existe condición el sistema deshabilita los botones "Eliminar Condición" y "Consultar".
	4.2 Ir a la actividad 3 del Flujo Normal de Eventos.
Sección 2 "Consultar"	
Acción del Actor	**Respuesta del Sistema**
6. El especialista presiona el botón "Consultar".	7. El sistema realiza la consulta con las condiciones añadidas.
	8. El sistema verifica si la consulta devuelve resultados.
	9. El sistema muestra el reporte de la consulta y da la posibilidad de salvar el reporte, eliminar y añadir condiciones y crear nueva consulta.
10. El especialista selecciona una de las siguientes opciones: ➢ Añadir condición ➢ Eliminar condición ➢ Nueva consulta ➢ Salvar	6. El sistema según la operación seleccionada por el especialista realiza una de las siguientes acciones: ➢ Si el especialista selecciona la operación "Añadir Condición" ir a la actividad 3 del Flujo Normal de Eventos. ➢ Si el especialista selecciona la operación "Eliminar condición" se ejecuta la sección "Eliminar condición". ➢ Si el especialista selecciona la operación "Nueva consulta" el sistema elimina los datos existentes y se ejecuta la actividad 3 del Flujo Normal de Eventos.

	➢ Si el especialista selecciona la operación "Salvar" se ejecuta la sección "Salvar reporte".
Flujos Alternos "Sección Consultar"	
Acción del Actor	**Respuesta del Sistema**
	4.1 Si la consulta no devuelve resultado el sistema muestra un mensaje para informar que no se encontraron datos para esa consulta.
Sección 3 "Salvar Reporte"	
Acción del Actor	**Respuesta del Sistema**
	1. El sistema muestra una ventana para seleccionar el directorio donde guardar el fichero del reporte.
10. El especialista selecciona la dirección donde será guardado el fichero del reporte.	3. El sistema guarda el reporte en la dirección especificada.
	4. Ir a la actividad 8 del Flujo Normal de Eventos
Prototipo de Interfaz de Usuario	
	



Poscondiciones	Se guarda el fichero con el reporte de la consulta a los ficheros EEG.

Anexo 2: Diagramas de Secuencia del Caso de Uso Consultar fichero EEG

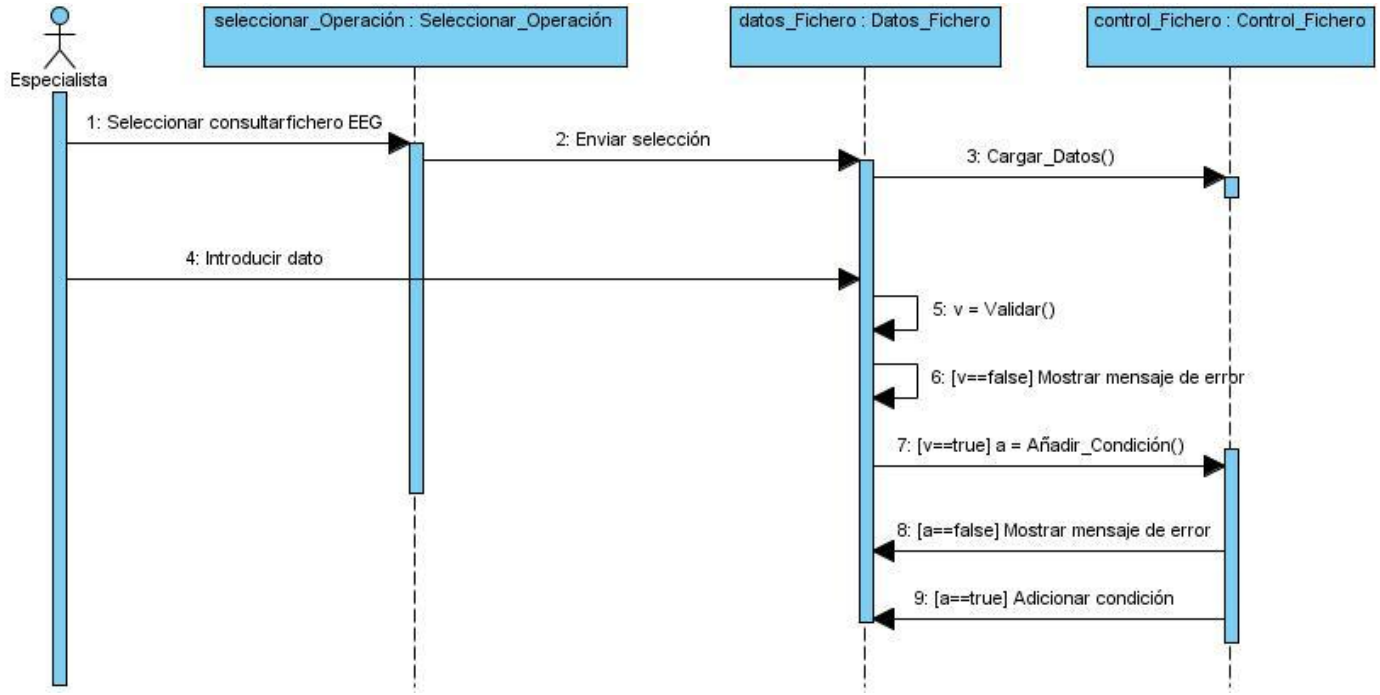


Figura 13: Diagrama de secuencia del Caso de Uso "Consultar fichero EEG" (Curso Normal de eventos)

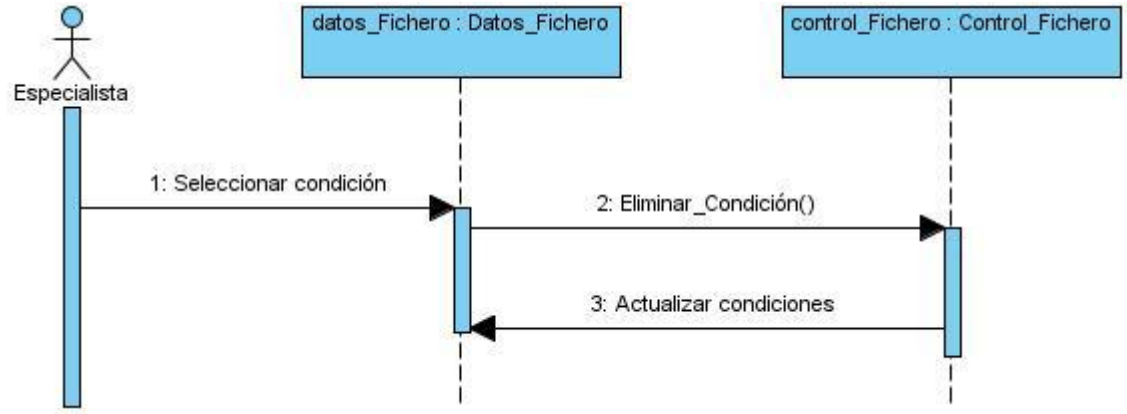


Figura 14: Diagrama de Secuencia del Caso de Uso "Consultar fichero EEG" (Sección "Eliminar condición")

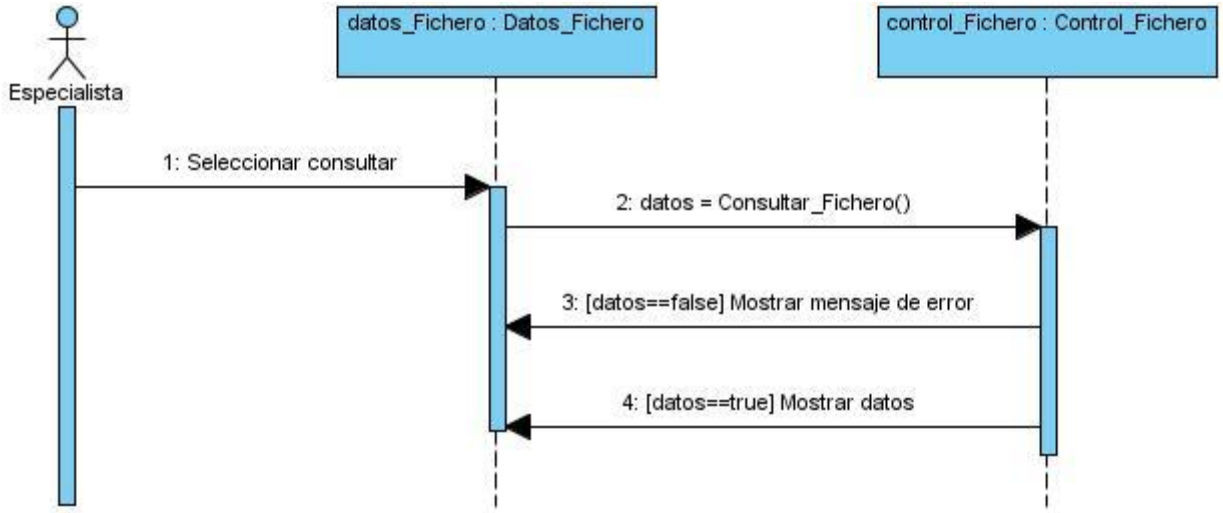


Figura 15: Diagrama de Secuencia del Caso de Uso "Consultar ficheros EEG" (Sección "Consultar")

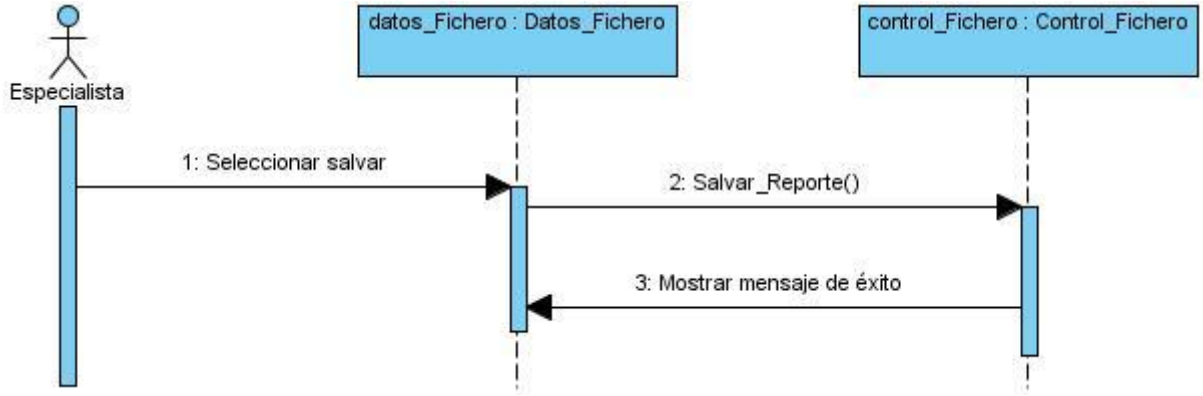


Figura 16: Diagrama de Secuencia del Caso de Uso "Consultar ficheros EEG" (Sección "Salvar reporte")

Anexo 3: Caso de prueba del Caso de Uso Consultar fichero EEG

Escenario	Variable1 Valor	Respuesta del Sistema	Resultado de la prueba	Flujo central
Adicionar condición correctamente.	V	Se añade la condición a la lista de condiciones que será usada para realizar la consulta a los ficheros	Se añade la condición a la lista de condiciones que será usada para realizar la consulta a los ficheros	1. Introducir el valor correcto en el campo. 2. Presionar el botón **Adicionar Condición**.

		EEG.	EEG.	
Adicionar condición incorrectamente	I	No se añade la condición a la lista de condiciones y se muestra un mensaje de error.	No se añade la condición a la lista de condiciones y se muestra un mensaje de error.	1. Introducir un valor anteriormente añadido en el campo. 2. Presionar el botón **Adicionar Condición**.

Escenario	**Variable1**	**Respuesta del Sistema**	**Resultado de la prueba**	**Flujo central**
Eliminar condición.	-	Se elimina la condición seleccionada de la lista de condiciones que será usada para realizar la consulta a los ficheros EEG.	Se elimina la condición seleccionada de la lista de condiciones que será usada para realizar la consulta a los ficheros EEG.	1. Seleccionar en el listbox **Condiciones** la condición que se desea eliminar. 2. Presionar el botón **Eliminar Condición**.
Escenario	**Variable1**	**Respuesta del Sistema**	**Resultado de la prueba**	**Flujo central**
Consultar la base de datos.	-	Se realiza la consulta a los ficheros EEG y se muestra el resultado en el listbox **Resultados**.	Se realiza la consulta a los ficheros EEG y se muestra el resultado en el listbox **Resultados**.	1. Presionar el botón **Consultar**.

Escenario	**Variable1**	**Respuesta del Sistema**	**Resultado de la prueba**	**Flujo central**
Salvar el reporte de la consulta.	-	Se abre una ventana para especificar la dirección donde se	Se abre una ventana para especificar la dirección donde se	1. Presionar el botón **Salvar**. 2. Indicar la dirección donde

		desea salvar el reporte de la consulta.	desea salvar el reporte de la consulta.	se desea guardar el reporte. 3. Presionar el botón **Aceptar**.

GLOSARIO DE TÉRMINOS

Base de datos: conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso.

Código abierto: significa que los programadores pueden leer, modificar y redistribuir el código fuente de un programa, éste evoluciona, se desarrolla y mejora.

Código fuente: es un conjunto de líneas de texto que son las instrucciones que debe seguir la computadora para así poder ejecutar las funcionalidades requeridas por el software.

Diagramas: representación gráfica de un conjunto de elementos. Visualizan un sistema desde diferentes perspectivas.

Driver: en su traducción al español significa controlador, un driver es un componente de software que permite la interacción entre base de datos y lenguajes de programación.

Electroencefalograma: estudio mediante el cual se mide la actividad eléctrica en el cerebro, lo que se denomina ondas cerebrales.

Fichero: se le llama fichero o archivo informático a un conjunto de información clasificada y almacenada de diversas formas para su conservación y fácil acceso en cualquier momento.

Herramienta informática: programa que ayuda al usuario analizar o buscar datos. Herramienta empleada para la construcción de otros programas o aplicaciones.

Lenguaje de programación: es el vínculo de comunicación entre el hombre y la computadora, es una técnica estándar de comunicación que permite expresar las instrucciones que han de ser ejecutadas en una computadora.

Matlab: software matemático con lenguaje de programación propio, permite el desarrollo de algoritmos, visualización de datos, análisis de datos y cálculo numérico.

Multiplataforma: es un término usado para referirse a los programas, sistemas operativos, lenguajes de programación, herramientas informáticas, u otra clase de software, que puedan funcionar en diversas plataformas. Por ejemplo, una aplicación multiplataforma podría ejecutarse en Windows, GNU/Linux y Mac OS X.

Toolbox: se conoce en español como caja de herramientas, es una colección de recursos, herramientas y materiales que pueden ser usados para realizar una tarea en especial.