UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Desarrollo de un servicio web para la medida automática del volumen cerebral a partir de imágenes de RMN

Proyecto de Fin de Carrera
Ingeniería Informática

Autor: Elena Carrascosa Beltrán
Tutor: Dr. José Vicente Manjón Herrera

Valencia, 2011

A mi abuelo Víctor, a quien le habría gustado ver que había otro ingeniero en la familia

Agradecimientos

A José Vicente Manjón, mi profesor y tutor de este proyecto, por su paciencia, su ayuda y su implicación más allá del deber de cualquier tutor. Ha sido un placer trabajar contigo.

A mis compañeros de laboratorio, por la buena compañía y por hacer mucho más llevaderas las horas de trabajo.

A Ghada y Sabina, pues sólo por llegar a conocerlas ya ha valido la pena cursar esta carrera. También a Elena, gran amiga y mejor persona. Sin vuestro apoyo no habría podido, literalmente, acabar este proyecto.

A Jorge, por ser la persona más importante de mi vida durante tanto tiempo. Hoy soy quien soy gracias a ti.

Tabla de Contenidos

Acrónimos y Abreviaturas

ASIC: Área de Sistemas de Información y Comunicaciones (UPV)
CSS: Cascading Style Sheets
FDA: U. S. Food and Drug Administration
FIFO: First-In-First-Out
GZ: GNU Zip
HTML: HyperText Markup Language
HTTP: HyperText Transfer Protocol
IBIME: Grupo de Informática Biomédica (UPV)
IIS: Internet Information Services
NIFTI: Neuroimaging Informatics Technology Initiative
PDF: Portable Document Format
PEAR: PHP Extension and Application Repository
PHP: PHP: Hypertext Processor
RAM: Random Access Memory
RAR: Roshal Archive
RMN: Resonancia Magnética Nuclear
SMTP: Simple Mail Transfer Protocol
SQL: Structured Query Language
TDD: Test-Driven Development
UPV: Universidad Politécnica de Valencia
WRA: Washington Radiology Associates
WYSIWYG: What-You-See-Is-What-You-Get

Índice de Figuras

1. INTRODUCCIÓN

1.1. Problema

La resonancia magnética nuclear o RMN es un fenómeno físico basado en las propiedades magnéticas de los núcleos atómicos, cuyo descubrimiento en 1946 por parte de Edward Purcell en Harvard y Felix Block en Standford de manera independiente los hizo merecedores del premio Nobel de Física en 1952.

La utilidad de la RMN reside en que permite alinear los campos magnéticos de diferentes núcleos atómicos en la dirección de un campo magnético externo. Dado que la respuesta a la influencia de este campo diferirá según el tipo de núcleo atómico, la RMN puede utilizarse para obtener información sobre la composición estructural de una muestra.

En los años posteriores al descubrimiento se fueron desarrollando aplicaciones para el análisis físico y químico de moléculas, y a principios de la década de los 70 se encontró un motivo para aplicar la RMN al campo de la medicina al comprobarse que los tiempos de relajación nuclear magnética de tejidos sanos y tumores diferían.

Poco después se obtuvieron las primeras imágenes de RMN, y hasta hoy en día se la considera una de las herramientas de diagnóstico médico por imagen más destacadas, siendo de uso común para el estudio de áreas como tórax, abdomen, corazón y cerebro. Este último caso es en el que se centra el presente proyecto.

Por lo detallado de las imágenes obtenidas por este procedimiento y su carácter no invasivo, la resonancia magnética cerebral es una técnica de gran ayuda para el estudio del funcionamiento del cerebro y el diagnóstico y evolución de diversos trastornos. Puede ayudar a determinar la parte del cerebro que está controlando funciones esenciales, o guiar la planificación de una cirugía o tratamiento. Es una técnica especialmente importante en la detección y control del crecimiento y comportamiento de tumores cerebrales, así como en la evaluación de los efectos de un accidente cerebrovascular o la supervisión de la evolución algunas enfermedades crónicas del sistema central nervioso.

También se usa para descartar alteraciones cerebrales, el estudio de la hipófisis y la detección de alteraciones oculares o del oído interno, entre otros.

VolBrain es una herramienta recientemente desarrollada por el Grupo de Informática Biomédica de la UPV (IBIME) destinada al análisis automático de los datos obtenidos a través de una RMN cerebral. El sistema funciona tomando unos datos de imagen previamente anonimizados en formato NIFTI como entrada y generando un informe en formato PDF que contenga los resultados del análisis de los datos, operando de modo totalmente transparente para el usuario.

El informe presenta varias tablas en las que se plasman resultados relativos a los volúmenes de los distintos tejidos que conforman las cavidades intracraneales (materia blanca, materia gris y fluido cerebroespinal). También proporciona información volumétrica de las diferentes estructuras cerebrales (cerebro, cerebelo y tronco encefálico) desglosadas por hemisferios y tejidos, así como la información volumétrica procedente de la segmentación automática de las diversas estructuras subcorticales (ventrículos laterales, núcleo caudado, putamen, tálamo, pálido, hipocampo, amígdala y núcleo accumbens).

El problema al que se enfrentará este proyecto es el de proporcionar un acceso generalizado a esta herramienta a los miembros de la comunidad científica que puedan beneficiarse de su utilidad con fines médicos o de investigación.

1.2. Objetivos

El objetivo a conseguir es la implementación de un sistema que proporcione acceso remoto a la funcionalidad que ofrece volBrain.

Para ello, será necesario conseguir una interfaz de usuario de uso sencillo y amigable que posibilite la transmisión y recepción de datos del equipo local al servidor en el que se encuentra volBrain.

En este mismo servidor se hace necesario implantar una gestión de usuarios y datos que haga posible la interacción remota con el sistema, registre en una base de datos la información pertinente y prepare los datos enviados por el usuario para su procesamiento por parte de volBrain.

Del mismo modo, hace falta un sistema de planificación de trabajos que se encargue de gestionar y atender adecuadamente las peticiones recibidas, para luego devolver el resultado al usuario sin que sea necesaria su interacción con el sistema.

1.3. Solución propuesta

La solución que se propone abordar el presente proyecto es una que cumpla de una forma precisa los objetivos enunciados en el punto anterior y sea susceptible de ser implementada con los recursos disponibles.

La creación de la interfaz de usuario se hará en forma de página web basada en estándares que muestre toda la información necesaria de forma entendible y estructurada, buscando una visión global y proporcionando a cada usuario acceso sólo a la información autorizada.

El hecho de que se implemente como una interfaz web presenta la ventaja de que el usuario puede acceder a la funcionalidad de volBrain desde cualquier equipo que cuente con una conexión a internet sin realizar ningún tipo de modificación en el entorno, simplificando la complejidad de uso del sistema al máximo.

Esta interfaz será la encargada de instruir al usuario en su funcionamiento y modo de uso, y será la herramienta a través de la que se realice la transmisión de los datos de imagen al servidor para su procesamiento.

La parte servidor de la web es la que realizará la gestión y preparación de los datos recibidos hasta dejarlos listos para ser procesados por volBrain. También llevará a cabo la gestión de los usuarios y las operaciones de almacenamiento de información y transacciones en general con una base de datos en la que se encontrará la información relevante para poder llevar a cabo todo el proceso.

Finalmente, el sistema de planificación de trabajos se implementará como un servicio del sistema que se ejecute constantemente esperando a recibir peticiones o, en su caso, monitorizando la petición en curso. Será capaz de seleccionar la siguiente petición a ser ejecutada, activar volBrain y suministrarle los datos de imagen correspondientes. El servicio también monitorizará el final del procesamiento de una petición, devolviendo entonces el informe con los resultados al usuario por correo electrónico. Así, todo el proceso tiene lugar de

forma desatendida desde el momento en que se envían los datos para su análisis.

1.4. Descripción de la memoria

La presente memoria está redactada siguiendo una estructura clásica para un proyecto de fin de carrera de una ingeniería.

La introducción hace una breve descripción del problema a resolver situándolo en un contexto que otorgue sentido a la ejecución del proyecto. A continuación se plantean unos objetivos medibles a conseguir durante el desarrollo del proyecto y se propone una solución plausible a ser desarrollada.

En el apartado de análisis se realiza una descripción más precisa del problema desde diferentes perspectivas y se enumeran los requerimientos específicos que presenta a fin de encauzar los esfuerzos del desarrollo en una dirección concreta.

La sección dedicada al diseño de la aplicación se centra en la toma de decisiones de implementación, desde la justificación de la elección de tecnologías realizada a la arquitectura escogida para la implementación, pasando por la metodología utilizada para el desarrollo de la aplicación.

Seguidamente se exponen los resultados y conclusiones de todo el proceso: las prestaciones del producto final, las dificultades presentadas a lo largo de su desarrollo y todos aquellos aspectos que no se hayan podido abordar por motivos de tiempo o recursos pero que se considere oportuno tener en cuenta como trabajo futuro.

Por último, se adjuntan las referencias bibliográficas que han sido utilizadas como fuente de información y como anexos a la memoria se incluyen un manual de usuario y un manual de administrador.

2. ANTECEDENTES

2.1. VolBrain

VolBrain es la herramienta de análisis volumétrico basada en RMN cerebral que este proyecto adapta para dar servicio a través de Internet con el fin de hacerla accesible a la comunidad científica e investigadora, tanto para su uso funcional como para su evaluación de cara a posibles mejoras.

El funcionamiento de volBrain se basa en el uso de un novedoso método basado en parches que trabaja con pequeñas secciones de la imagen de RMN en lugar de hacerlo a nivel de estructura cerebral como sucede en los métodos de distorsión de plantillas. El sistema se sirve de una biblioteca de imágenes segmentadas manualmente que son utilizadas como plantillas para realizar el etiquetado de los vóxeles de la imagen (píxeles tridimensionales) según la estructura cerebral a la que pertenecen.

El etiquetado de cada vóxel tiene lugar por comparación de la sección que lo rodea con secciones de las plantillas en las que se conoce el valor de la etiqueta de los vóxeles centrales. Para la comparación se tiene en cuenta la similaridad de intensidad entre la sección a etiquetar y cada sección plantilla sin tener en cuenta la distancia espacial entre ellas, a fin de dar mayor importancia a las muestras más relevantes.

Finalmente, se lleva a cabo un análisis volumétrico sobre las estructuras ya segmentadas atendiendo al tipo de tejido que las forman.

Esta información es muy valiosa para el diagnóstico de algunas alteraciones y enfermedades degenerativas. Como ejemplo puede consultarse el paper publicado conjuntamente por investigadores de la Universidad Politécnica de Valencia y la McGuill University de Montreal (Pierrick Coupe, Jose V. Manjón, Vladimir Fonov, Jens Pruessner, Montserrat Robles, D. Louis Collins. Patch-based Segmentation using Expert Priors: Application to Hippocampus and Ventricle Segmentation. NeuroImage, 54(2): 940-954, 2011.), en el que se realiza un análisis del hipocampo de 80 individuos sanos y los ventrículos laterales de 80 enfermos de Alzheimer mediante este método, con el claro objetivo de poder establecer un diagnóstico efectivo para nuevos pacientes.

Cabe destacar que, en el momento de la aparición de volBrain, no existía ningún software o programa de características similares en el mercado, de uso público ni privado.

2.2. Alternativas

Durante el periodo de desarrollo del presente proyecto realizó su aparición en el mercado NeuroQuant, una herramienta que, al igual que volBrain, realiza un post-procesado automático de imágenes de RMN cerebral a partir del cual se genera un nuevo conjunto de datos segmentado y etiquetado por estructuras, así como dos informes diferentes con los resultados de su análisis volumétrico.

Este software, disponible también a través de Internet, ha sido desarrollado por los laboratorios Cortechs, especializados en tecnologías de análisis de imagen cerebral, y es empleado como herramienta por la WRA (Washinton Radiology Associates), un grupo cuya misión está centrada en la detección de enfermedades mediante un amplio abanico de modalidades de imagen médica con la ayuda de nuevas tecnologías.

Uno de los principales puntos fuertes a tener en cuenta de NeuroQuant es que ha sido aprobado para su uso como dispositivo de diganóstico clínico por la FDA (Food and Drug Administration), la agencia americana encargada de preservar la salud pública mediante la supervisión y regulación de alimentos, tabaco, productos farmacéuticos y similares.

No obstante, NeuroQuant presenta también dos grandes desventajas con respecto a volBrain, ambas relacionadas con los recursos que requiere:

- A diferencia de volBrain, no se trata de un software gratuito, lo que limita el acceso al mismo para su uso y testeo.
- El tiempo de diagnóstico de NeuroQuant es de alrededor de quince minutos, mientras que volBrain emplea una media de ocho minutos en efectuar un análisis completo.

Vale la pena mencionar, aunque no se trate exactamente de un software equivalente en cuanto a objetivos, la FSL (FRMRIB Software Library). Se trata de una librería de herramientas multiplataforma para el análisis de datos de imagen de RMN. Su desarrollo se debe principalmente a miembros del Analysis Group, FRMBI, con sede en Oxford, Reino Unido.

Dentro de la librería FSL podemos encontrar FIRST, una herramienta de segmentación de imágenes de RMI cerebral basada en el uso de modelos construidos a partir de una gran cantidad de imágenes segmentadas manualmente para realizar la segmentación de las distintas estructuras subcorticales.

FSL supone una referencia interesante tanto como punto de partida como para la realización de un posible trabajo futuro en la ampliación del software volBrain, ya que incluye una extensa variedad de herramientas dignas de estudio en el procesado y segmentado de imágenes cerebrales.

2.3. Conclusiones

Podemos afirmar, tanto en el momento de acometimiento de este proyecto como en instante de su finalización, que no existe un software que realice con la eficiencia de volBrain un análisis volumétrico cerebral a partir de imágenes de RMN. Además, como se ha mencionado, se trata de una herramienta gratuita, lo que además de suponer una ventaja obvia aumenta las posibilidades de su mejora y desarrollo posteriores, que se prevé estén basados en el feedback obtenido por parte de una comunidad de usuarios mucho mayor.

3. ANÁLISIS

Este capítulo está dedicado a identificar y definir los requisitos previos del sistema que servirán de base para el posterior diseño e implementación de la aplicación. En él se define el ámbito del problema y la funcionalidad de la aplicación, para finalmente establecer unos requerimientos específicos para los diferentes elementos del proyecto.

3.1. Ámbito

El uso final de esta aplicación está previsto que tenga lugar en un entorno especializado en la materia como puede ser un hospital o centro de investigación.

Se contempla que el sistema sea utilizado por dos tipos de usuarios: usuarios generales y usuarios administradores. El usuario general simplemente se sirve de las prestaciones que le ofrece la aplicación, mientras que el usuario administrador tiene la función de gestionar el buen funcionamiento de la misma y editar y mantener su contenido.

3.2. Funcionalidad de la aplicación

La aplicación contará con una interfaz web que debe ser capaz de ejecutarse correctamente en los principales navegadores.

A través de esta interfaz los usuarios podrán registrarse en la base de datos e identificarse ante el sistema por medio de sesiones, obteniendo así acceso a la funcionalidad pertinente según el tipo de usuario.

De cara al usuario general, la interfaz supone un medio de envío de datos de imagen a volBrain, en tanto que de cara al usuario administrador es un medio de edición de contenido.

La parte servidor de la aplicación se encargará de la preparación de los datos para su procesamiento y su encolamiento en una lista de trabajos. El planificador de trabajos irá consultando esta lista para escoger un trabajo determinado de entre los pendientes cuando no haya ningún otro trabajo en proceso. Una vez escogido, el planificador lanzará volBrain y le suministrará los datos correspondientes al trabajo, monitorizando el procesamiento y tomando las acciones necesarias en caso de error. Una vez finalizado con éxito, el mismo planificador se encargará de enviar por correo electrónico al usuario el informe con los resultados del análisis.

3.3. Características de los usuarios

El usuario final medio de la aplicación es un profesional del campo de la medicina que no posee necesariamente conocimientos informáticos.

En el caso del usuario administrador sí que es conveniente que éste posea, al menos, un conocimiento más amplio sobre el funcionamiento de la aplicación, así como nociones generales de las tecnologías que ésta utiliza de cara al mantenimiento y gestión generalizados de la misma.

3.4. Especificación de requisitos

Llamamos requisitos funcionales a aquellas características requeridas del sistema que suponen una capacidad de acción del mismo o, por decirlo de otro modo, van a ser satisfechas mediante la implementación de un bloque de código.

Requisitos no funcionales son aquellas exigencias cualitativas del proyecto que se satisfacen de forma más general, bien adoptando la utilización de un determinado paradigma o lenguaje de programación, empleando un hardware concreto, etc. Estos requisitos no deben implementarse, sino cumplirse como si de una restricción se tratara.

3.4.1. Requisitos funcionales

- **Gestión de usuarios:** La aplicación debe permitir que un usuario se registre por sí mismo en la base de datos del sistema, pudiendo únicamente acceder a la funcionalidad de la aplicación los usuarios registrados. Una vez confirmado el registro, el procedimiento de baja no se contempla. Esto se hace con el fin de mantener la coherencia con la información referente al procesamiento de datos existente en la base de datos, que va directamente ligada al usuario que los suministra. El usuario puede dejar de hacer uso de la aplicación en cualquier momento sin más, pero permanecerá en la base de datos, lo cual le permitirá también volver a hacer uso de la aplicación en cualquier momento futuro sin necesidad de volver a registrarse.

- **Gestión de sesiones:** En cualquier momento debe permitirse la obtención o cierre de una sesión por parte de un usuario registrado. La obtención de una sesión garantiza que el usuario será reconocido por el sistema y le será concedido el acceso a la funcionalidad de la aplicación.

- **Envío de información:** Cualquier usuario que se haya identificado ante el sistema debe poder enviar archivos de datos de imagen anonimizados en un formato apropiado para su procesamiento por parte de volBrain.

- **Gestión de trabajos:** El sistema debe de presentar la capacidad de gestionar las distintas peticiones que recibe, registrándolas en el sistema y almacenando los datos para luego suministrarlos correcta y ordenadamente a volBrain tras asegurarse de que no hay ninguna otra petición en curso.

- **Procesamiento de la información:** La aplicación debe ser capaz de llevar a cabo las acciones necesarias para garantizar que la información suministrada por el usuario sea correctamente procesada. Tras la selección del trabajo a ejecutar debe proceder al lanzamiento de volBrain, indicándole los datos a procesar. Del mismo modo, debe de ser capaz de detectar fallos en el proceso (ya sean por parte de volBrain o de la propia aplicación) y bloqueos en el funcionamiento de volBrain, procediendo en ese caso a terminar su ejecución.

- **Comunicación con el usuario:** El sistema debe de poder enviar correos al usuario para informarle de los progresos en la gestión de su petición, avisarle ante cualquier eventualidad y proporcionarle el informe final obtenido en el caso esperado de que su petición se procese con éxito.

También debe proporcionar un mecanismo alternativo de acceso a los datos obtenidos durante las distintas etapas del procesamiento, ya que se trata de archivos de mucho mayor tamaño.

- **Edición de contenido:** Debe permitirse al usuario identificado ante el sistema como administrador la edición del contenido de las distintas secciones de la aplicación.

3.4.2. Requisitos no funcionales

- La parte cliente de la aplicación debe estar basada en una tecnología web para ser capaz de proporcionar acceso a la funcionalidad de volBrain a cualquier usuario desde un simple navegador, sin necesidad de realizar ninguna instalación o modificación en el sistema local.
- El código referente a la planificación de trabajos y gestión de su procesamiento debe ser implementado como un demonio que se ejecute continuamente en segundo plano esperando a recibir peticiones de usuario.
- La solución debe ser escalable mediante la estrategia scale-up, consistente en añadir más recursos al servidor, según las necesidades de procesamiento. También debe tenerse en cuenta la posibilidad de recurrir a la escalabilidad mediante la estrategia scale-out, consistente en añadir más servidores, en un desarrollo futuro, y buscar que en ese hipotético caso solo fueran necesarias modificaciones relativas a la gestión del balanceo de carga entre los distintos servidores.
- La interfaz de usuario ha de ser clara, amigable y de fácil manejo, puesto que el usuario puede no poseer conocimientos informáticos.
- La interfaz debe hacer las veces de página web del proyecto, ofreciendo información sobre el mismo, instrucciones sobre su utilización y presentando las secciones esperables en un entorno similar.
- El código ha de ser especialmente claro y estar bien estructurado y comentado de cara a facilitar el mantenimiento. Por el mismo motivo, deben cumplirse los estándares de las tecnologías utilizadas.
- La aplicación debe posibilitar y emplear la reutilización de código.

- El diseño ha de contemplar el uso óptimo de recursos tales como conexiones a la base de datos.
- La solución debe ofrecer adecuados niveles de servicio que garanticen la disponibilidad y la recuperación ante fallos.
- El sistema debe soportar la compresión y descompresión de archivos en los formatos ZIP, RAR y GZ.
- El equipo en el que se encuentre ubicada la parte servidor de la aplicación debe contar con un sistema operativo de 64 bits para que volBrain pueda ejecutarse.
- El servidor necesita una memoria RAM de capacidad elevada (alrededor de 8GB) dado el alto coste computacional de la ejecución de volBrain.
- Es necesario que el equipo en el que se halla el servidor cuente con un disco duro de varios terabytes de capacidad a fin de almacenar los datos de imagen enviados por los usuarios, puesto que se trata de archivos de gran tamaño.
- Se necesitará contar con un procedimiento de copias de seguridad y de restauración del sistema completo en caso de contingencia.

3.5. Casos de uso

Los casos de uso son una técnica que pretende profundizar en los requisitos funcionales del sistema, describiendo paso por paso las distintas secuencias de interacción que pueden mantenerse con la aplicación desde un punto de vista externo.

Su propósito es identificar todos los procesos de principio a fin, documentando el contexto en el que se desarrollan y ayudando así a concretar las necesidades del proyecto.

3.5.1. Actores

Los actores son entidades ajenas al sistema que pueden interactuar o comunicarse con el mismo. Más que a una entidad concreta, un actor representa un rol genérico que desempeña un papel determinado a la hora de hacer uso del sistema. Puede tratarse de un usuario humano, hardware externo u otros sistemas que intercambian información con la aplicación.

En este caso, pueden distinguirse dos tipos de actores que interactuarán con la aplicación, tratándose de ambos casos de usuarios humanos.

Por una parte están los usuarios generales, a los que se hará referencia simplemente como “usuarios”. Se trata del usuario que podrá hacer uso de la funcionalidad básica de la aplicación, enviando información que el sistema procesará para posteriormente devolverle el resultado.

Por otra, existe otro tipo de actor que desempeña el rol de usuario administrador. Este actor, aunque gozará en principio de los mismos privilegios que un usuario general, dispone además de una funcionalidad extendida ya que su interacción con el sistema responde a un objetivo totalmente distinto. Su función es la supervisión y mantenimiento del sistema, actualizando el contenido de la interfaz web e interviniendo ante cualquier incidencia del funcionamiento de la aplicación. Se espera que en principio este rol se asigne a un único usuario físico.

El comportamiento de la aplicación será distinto dependiendo del tipo de actor que la utilice, difiriendo no sólo la información mostrada sino las acciones que el actor puede realizar.

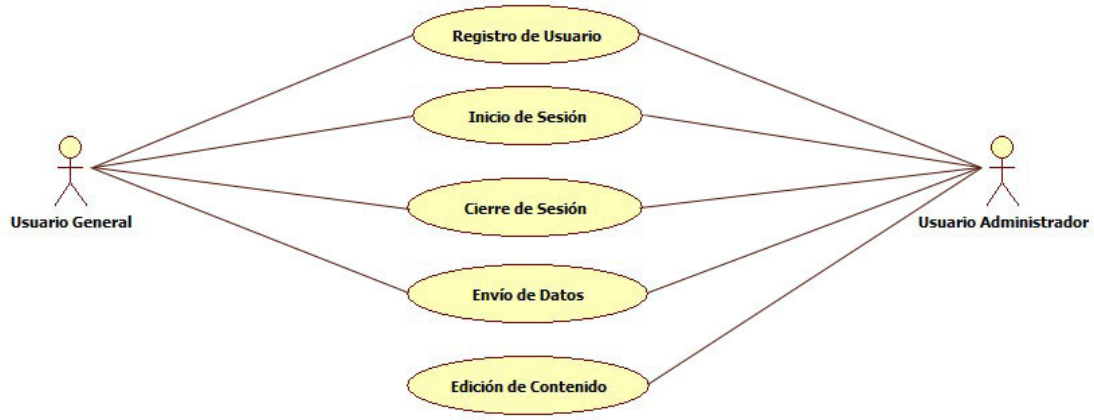


Fig. 1. Casos de Uso

3.5.2. Diagramas de casos de uso

Los diagramas de casos de uso son documentos que recogen un aspecto concreto de la funcionalidad de la aplicación, describiendo al detalle la interacción o secuencia de interacciones de un actor con la misma.

A continuación se muestran los diagramas correspondientes a los principales casos de uso del sistema.

Caso de Uso	
Código: CU1	**Nombre:** Registro de usuario
Actor/es: Usuario	
Descripción: El usuario introduce sus datos en el formulario destinado a tal efecto.	
Precondiciones: El usuario no debe estar previamente registrado.	
Flujo Principal: **1.** El actor introduce como mínimo los datos correspondientes a los campos obligatorios del formulario y pulsa el botón de envío. **2.** Los datos no están previamente registrados en el sistema, son correctamente validados y se almacenan en la base de datos. **3.** Un correo con un enlace de confirmación es enviado a la dirección de correo indicada por el actor en el formulario de registro. **4.** El actor accede a la dirección del enlace de confirmación incluido en el correo.	

	5. La cuenta se registra como habilitada en la base de datos. **6.** Se informa al actor de la confirmación de su cuenta mediante un mensaje. **7.** Si el actor ha especificado en el formulario de registro que accede a aparecer en el listado público de usuarios, se envía un correo al usuario administrador informando de su consentimiento.
Flujo Alternativo nº1:	**1.** El actor introduce como mínimo los datos correspondientes a los campos obligatorios del formulario y pulsa el botón de envío. **2.** Los datos introducidos corresponden a un usuario ya registrado en el sistema. **3.** Se informa al actor con un mensaje.
Flujo Alternativo nº2:	**1.** El actor introduce como mínimo los datos correspondientes a los campos obligatorios del formulario y pulsa el botón de envío. **2.** Los datos introducidos no son validados correctamente, o alguno de los campos de carácter obligatorio no ha sido completado. **3.** Se informa al actor con un mensaje.
Flujo Alternativo nº3:	**1.** El actor introduce como mínimo los datos correspondientes a los campos obligatorios del formulario y pulsa el botón de envío. **2.** Los datos introducidos son validados correctamente. **3.** Se produce algún error en la transacción de acceso a la base de datos. **4.** Se informa al actor con un mensaje.

Postcondiciones: El usuario queda registrado en la base de datos del sistema.

Caso de Uso	
Código: CU2	**Nombre:** Inicio de sesión
Actor/es: Usuario	
Descripción: El usuario introduce su identificador de usuario y contraseña para autenticarse en el sistema. Los datos serán introducidos en un formulario que aparecerá al pulsar en el enlace de inicio de sesión, que será claramente visible en la página.	
Precondiciones: El usuario debe estar previamente registrado.	
Flujo Principal: **1.** El actor se identifica introduciendo su usuario y contraseña y pulsando el botón de iniciar sesión. **2.** Los datos pertenecen a un usuario previamente registrado y ambos coinciden con uno de los registros almacenados en la base de datos del sistema.	
Flujo Alternativo nº1: **1.** El actor se identifica introduciendo su usuario y contraseña y pulsando el botón de iniciar sesión. **2.** Los datos no coinciden con ningún registro de la base de datos, bien por un error en la introducción de los mismos por parte del actor, bien porque el usuario no se encontraba previamente registrado. **3.** Se informa al actor con un mensaje para que introduzca los datos de nuevo o, en su caso, para que	

proceda a registrarse.
Flujo Alternativo nº2: 1. El actor se identifica introduciendo su usuario y contraseña y pulsando el botón de iniciar sesión. **2.** Se produce algún error en la transacción de acceso a la base de datos. **3.** Se informa al actor con un mensaje.
Postcondiciones: Se obtiene una sesión asociada al usuario. Una vez iniciada la sesión el enlace de inicio de sesión será sustituido por un enlace para salir de la misma.

Caso de Uso	
Código: CU3	**Nombre:** Cierre de sesión
Actor/es: Usuario	
Descripción: El usuario pulsa el enlace de cierre de sesión.	
Precondiciones: El usuario debe haber iniciado con éxito una sesión previamente.	
Flujo Principal: 1. El actor pulsa el enlace de cierre de sesión. **2.** Se destruyen las variables de sesión y se cierra la sesión.	
Postcondiciones: La sesión asociada al usuario deja de existir.	

Caso de Uso	
Código: CU4	**Nombre:** Envío de datos para su procesamiento
Actor/es: Usuario	
Descripción: El usuario selecciona un archivo local para ser enviado para su procesamiento.	
Precondiciones: El usuario debe haber iniciado con éxito una sesión previamente.	
Flujo Principal: **1.** El actor accede al área privada de la interfaz web e indica la ruta local del archivo que desea que sea procesado. Opcionalmente introduce la información adicional sobre el paciente. Para enviar el archivo pulsa sobre el botón de envío. **2.** Los datos introducidos son validados, el archivo es subido al servidor web para su procesamiento y le es asignado un identificador. El trabajo es registrado y marcado como preparado en la base de datos. **3.** Se informa al actor del éxito del envío con un mensaje.	
Flujo Alternativo nº1: **1.** El actor accede al área privada de la interfaz web e indica la ruta local del archivo que desea que sea procesado. Opcionalmente introduce la información adicional sobre el paciente. Para enviar el archivo pulsa sobre el botón de envío. **2.** El archivo indicado no es del tipo permitido o es demasiado grande. Los datos no son validados correctamente. **3.** Se informa al actor con un mensaje.	

Flujo Alternativo nº2: **1.** El actor accede al área privada de la interfaz web e indica la ruta local del archivo que desea que sea procesado. Opcionalmente introduce la información adicional sobre el paciente. Para enviar el archivo pulsa sobre el botón de envío. **2.** Los datos introducidos son validados. Se produce un error en el envío del archivo o su descompresión. **3.** Se informa al actor con un mensaje.
Flujo Alternativo nº3: **1.** El actor accede al área privada de la interfaz web e indica la ruta local del archivo que desea que sea procesado. Opcionalmente introduce la información adicional sobre el paciente. Para enviar el archivo pulsa sobre el botón de envío. **2.** Los datos introducidos son validados correctamente. **3.** Se produce algún error en la transacción de acceso a la base de datos. **4.** Se informa al actor con un mensaje.
Postcondiciones: El contenido del archivo enviado queda en el servidor a la espera de ser procesado. Los datos referentes a dicho archivo necesarios para su procesamiento quedan registrados en la base de datos del sistema.

Caso de Uso	
Código: CU5	**Nombre:** Edición de contenido
Actor/es: Usuario administrador	

Descripción: El usuario administrador realiza la edición del contenido de cualquiera de las distintas secciones informativas de la página web.
Precondiciones: El usuario debe haber iniciado con éxito previamente una sesión como usuario administrador.
Flujo Principal: 1. El actor accede a la sección de la interfaz web cuyo contenido desee editar. **2.** Se muestra un editor que incluye el contenido actual de la sección. **3.** El actor realiza los cambios deseados en el contenido de la sección y pulsa el botón de envío. **4.** Se muestra de nuevo el editor, pudiendo verse reflejados los cambios realizados sobre el contenido.
Flujo Alternativo nº1: 1. El actor accede a la sección de la interfaz web cuyo contenido desee editar. **2.** Se muestra un editor que incluye el contenido actual de la sección. **3.** El actor realiza los cambios deseados en el contenido de la sección y pulsa el botón de envío. **4.** Se produce algún error en la transacción de acceso a la base de datos. **5.** Se informa al actor con un mensaje.
Postcondiciones: Los cambios realizados al contenido quedan registrados correctamente en la base de datos.

4. DISEÑO E IMPLEMENTACIÓN

En el presente capítulo se perfilarán de un modo más práctico los requisitos definidos en la etapa de análisis, a fin de conseguir los objetivos expuestos en la introducción. Así mismo, se describirán las decisiones de implementación derivadas del estudio de estos requerimientos, definiéndolas desde distintas perspectivas.

Los principios que servirán como guías de diseño, como es habitual en el desarrollo de aplicaciones, son la simplicidad sin pérdida de funcionalidad, la idoneidad de la solución de cara a la implementación, la visión global del sistema y la capacidad de ampliación futura del mismo y facilidad de mantenimiento.

4.1. Elección de tecnologías

El proyecto a realizar presenta unas necesidades muy heterogéneas en cuanto a tecnologías se refiere. Por tanto, es de vital importancia en esta elección contemplar la compatibilidad y las posibilidades de integración de las mismas para la obtención de una aplicación robusta.

Una característica que se ha perseguido en todos los casos es que se tratara de software libre. La utilización de software libre, máxime si a su vez es de código abierto, es especialmente atractiva en el desarrollo de aplicaciones dentro del ámbito académico ya que el acceso al código promueve el intercambio de conocimiento. Del mismo modo, la posibilidad de libre modificación del código permite su adaptación a las necesidades específicas de cada proyecto, y la gratuidad de su uso es deseable dentro del marco de la actividad investigadora.

También se ha buscado que, de ser posible, se tratase de herramientas multiplataforma que posibiliten la portabilidad de la aplicación a otros sistemas en una etapa futura de desarrollo.

4.1.1. Parte Cliente

4.1.1.1. HTML

El HTML (HyperText Markup Language) es el lenguaje de marcado más extendido en la creación de páginas web. Su función es describir la estructura de la página y su contenido en forma de texto, complementándola con otros elementos como pueden ser imágenes.

Es importante incidir en que un lenguaje de marcado no es un lenguaje de programación, ya que no posee funciones aritméticas o variables, sino un lenguaje de codificación.

El marcado que proporciona HTML se realiza a través de etiquetas que designan los diferentes elementos que darán forma a la estructura del documento. Cada elemento tiene a su vez atributos y contenido, sujetos éstos a ciertas restricciones para que el documento se considere válido. Existen distintos tipos de marcado: el marcado estructural describe el propósito del texto, el marcado presentacional describe la apariencia del texto sin importar su función, y el marcado hipertextual se utiliza para enlazar partes del documento con partes del mismo u otros documentos.

La elección de HTML para este proyecto ha venido determinada por su sencillez y popularidad. No se requiere de ninguna aplicación para su desarrollo, bastando con un simple editor de texto, pero a su vez pueden emplearse editores WYSIWYG (What You See Is What You Get) que permiten la edición de HTML de modo visual y sin un conocimiento profundo del lenguaje. Ambas opciones pueden combinarse, lo que supone un incentivo interesante por facilitar el futuro mantenimiento de la aplicación.

El HTML también puede describir la apariencia de un documento hasta cierto punto empleando el marcado presentacional, pero en este caso la separación lógica elegida para el código de la aplicación propicia que se haya optado por el CSS para el diseño gráfico de la web.

4.1.1.2. CSS

CSS (Cascading Style Sheets) es un lenguaje utilizado para describir la presentación de un documento escrito en un lenguaje de marcado como pueda ser HTML o XML.

El propósito del lenguaje es habilitar la separación del contenido del documento de la presentación. La información de estilo puede encontrarse incluida en el mismo documento o bien ser adjuntada como un documento independiente recibiendo el nombre de “hoja de estilo”.

La especificación de la presentación viene descrita en forma de reglas que especifican un selector (pudiendo ser éste un tipo genérico de elemento, un identificador correspondiente a un elemento concreto o una etiqueta definida por el usuario) y una serie de pares propiedad-valor aplicables al mismo.

En las hojas de estilo CSS se especifica un esquema de prioridades que determina en qué orden se aplicarán las reglas de presentación en caso de que distintas reglas sean aplicables a un determinado elemento, siendo predecible el resultado.

La elección de CSS para especificar la presentación de la página web frente al marcado presentacional que nos ofrece HTML ha sido originada por el mayor número de ventajas que ofrece esta opción, ya que el uso de hojas de estilo proporciona más flexibilidad y control en la definición de la presentación, permite mayor accesibilidad al contenido y da la posibilidad de que una sola hoja de estilo sea usada por un gran número de documentos HTML, centralizando la presentación y estandarizando el diseño.

4.1.1.3. JavaScript

JavaScript es un lenguaje de programación interpretado de carácter dinámico que soporta los paradigmas de programación imperativo, funcional y orientado a objetos.

Su uso tiene lugar principalmente en la parte cliente de una aplicación web, aportando mejoras a la interfaz de usuario, permitiendo la ejecución de operaciones y añadiendo funcionalidad a la misma.

JavaScript es interpretado por el navegador, de modo que la respuesta a las acciones del usuario es más rápida que si el código se ejecutara en la parte servidor de la aplicación, aumentando esto la interactividad.

4.1.2. Parte Servidor

4.1.2.2. Apache

El servidor HTTP Apache en un servidor web de código abierto multiplataforma que implementa el protocolo HTTP/1.1.

Se trata del servidor HTTP de uso más extendido, siendo su uso principal el envío de páginas web tanto estáticas como dinámicas a través de internet de forma segura y confiable.

Su configuración está centralizada en unos pocos archivos, y su arquitectura es muy modular. El servidor consta de un núcleo y diversos módulos que aportan distintas partes de la funcionalidad necesaria en un servidor web, como seguridad en las comunicaciones, autenticación, compresión del contenido, envío de páginas dinámicas en lenguajes como PHP, Perl o Ruby, y control de tráfico.

Como servidor web presenta un alto rendimiento y proporciona una serie de módulos multiproceso que permiten que se ejecute en diferentes modos de ejecución relativos al proceso, proceso e hilos de forma conjunta, o a eventos, en función de las necesidades de una determinada infraestructura.

El procesado de las peticiones HTTP tiene lugar de forma consistente y confiable dentro de unos límites de tiempo razonables.

Algunas de las ventajas de Apache son su extensión de uso, modularidad, alta configurabilidad y carácter multiplataforma. Su popularidad facilitará la obtención de información a la hora de conseguir soporte para el futuro mantenimiento de la página web, y el hecho de que se trate de una aplicación de código abierto desarrollado y mantenido por una amplia comunidad de desarrolladores hace que se encuentre en un proceso de mejora continua.

En el caso del presente proyecto, se ha optado más concretamente por el paquete XAMPP, que integra el servidor Apache con una base de datos MySQL e intérpretes para los lenguajes PHP y Perl.

Esta elección ha venido motivada por distintas razones. Por una parte, el paquete incluye todas las tecnologías que se han escogido para la implementación de la parte servidor de la aplicación. Por otra, su instalación es mucho más rápida, limpia y sencilla que llevar a cabo la instalación de cada uno

de sus componentes por separado, y su configuración por defecto es bastante apropiada para el desarrollo del proyecto, siendo necesario un número mínimo de modificaciones.

A pesar de que en un principio fue diseñado como una herramienta de desarrollo local, en la práctica su uso como servidor de producción está bastante extendida, ya que con algunas pequeñas modificaciones el paquete es lo bastante seguro para ello e incluye una herramienta especial para proteger las partes más importantes del mismo.

4.1.2.2. PHP

PHP (PHP: Hypertext Preprocessor) es un lenguaje de programación multiplataforma de propósito general que está especialmente aconsejado para el desarrollo web.

Con este propósito, el código PHP puede ser incluido junto al código HTML. La interpretación del código PHP tendrá lugar en la parte del servidor por un módulo procesador de PHP. Sin embargo, dado que las características del lenguaje dan lugar a un abanico mucho más amplio de posibilidades de uso, el lenguaje también puede ser interpretado de forma autónoma a través de una interfaz de línea de comandos.

PHP tiene cientos de funciones base, que se convierten en muchísimas más en caso de hacerse uso de las extensiones del lenguaje. Así mismo, el programador puede definir funciones adicionales y tiene la posibilidad de desarrollar sus propias extensiones en C para añadir funcionalidad al mismo.

Se ha optado por PHP para programar la parte servidor de la aplicación por su completitud, idoneidad para el propósito del proyecto y sencillez de aprendizaje. Se ha tenido especialmente en cuenta la existencia de extensiones que dan soporte al API de Windows, puesto que posibilitan el desarrollo de servicios del sistema, ofreciendo así la oportunidad de programar toda la parte servidor de la aplicación en el mismo lenguaje y profundizar en el aprendizaje del mismo.

4.1.2.3. MySQL

MySQL es un sistema de gestión de bases de datos relacionales multiplataforma y multihilo que proporciona acceso simultáneo a múltiples usuarios a distintas bases de datos.

Existen APIs específicas a muchos lenguajes de programación que incluyen librerías que permiten a las aplicaciones acceder a las bases de datos MySQL y realizar transacciones.

MySQL ofrece una gran adaptabilidad permitiendo gran variedad de tipos de datos y, lo que es menos común, la posibilidad de elegir el tipo de tabla en que guardar los registros, permitiendo incluso la inclusión de distintos tipos de tablas en una misma base de datos.

Proporciona diversos motores de bases de datos, siendo MyISAM el motor proporcionado por defecto y en este caso el más adecuado a las necesidades del proyecto, puesto que en entornos de baja concurrencia en la modificación de datos, como es el caso, convierte a MySQL en una base de datos muy rápida en lectura.

En cuanto a la administración de las bases de datos MySQL se refiere, el sistema incluye herramientas de línea de comandos para tal propósito. Sin embargo, es mucho más usual descargar alguno de los diversos front-ends de administración de MySQL desarrollados por terceros como Adminer, DBEdit o phpMyAdmin.

Para la elección de MySQL como sistema de gestión de bases de datos se han tenido en cuenta su adaptabilidad, rapidez y portabilidad, así como su estrecha relación con el lenguaje PHP y la existencia de amplia documentación de apoyo. Como front-end de administración se empleará phpMyAdmin, integrado en el paquete XAMPP y de uso sencillo, intuitivo y amigable.

4.1.2.4. Filezilla

Filezilla es un servidor FTP que soporta SSL al mismo nivel de encriptación que el navegador utilizado. También soporta la compresión de archivos, lo que puede mejorar la tasa de transferencia del servidor, y permite limitar la velocidad de transferencia para evitar que se ocupe todo el ancho de banda.

Se ha tomado la decisión de emplear un servidor de archivos por preferirse la transferencia de archivos al usuario mediante el protocolo FTP a la transferencia mediante el protocolo HTTP. El uso del protocolo FTP para la descarga de archivos añade velocidad y seguridad al proceso, y permite que las descargas puedan ser continuadas tras una interrupción de la conexión.

4.2. Metodología

Una de las principales dificultades a la hora de planificar la implementación de este proyecto proviene del hecho de que existe una alta probabilidad de que los requisitos que deba cumplir el resultado final sufran modificaciones sensibles a lo largo del proceso, en especial en lo referente al añadido de nuevas funcionalidades o cambios en las ya existentes. Esto es así debido a que, al tratarse de una aplicación de uso práctico para el Grupo de Informática Biomédica de la UPV, se halla sujeta a la evolución de las necesidades de éste.

Resulta, por tanto, esencial buscar un conjunto de técnicas flexibles que permita adaptarse a una serie de requisitos cambiantes. Por suerte, este mismo problema ha estado presente en el ámbito profesional, en especial en empresas informáticas que programan para clientes externos.

Aunque en España estos conceptos aún resultan relativamente nuevos, en otros países, en especial los de habla anglosajona, es cada vez más común el empleo de metodologías ágiles, y más concretamente, de las técnicas de TDD o Test-Driven Developement.

4.2.1. Agilismo

El agilismo o desarrollo ágil del software es una metodología de trabajo basada en la división de un proyecto en iteraciones, en principio de duración relativamente corta, a lo largo de todo su ciclo de vida. Idealmente, cada iteración posee una fase de planificación, análisis de requerimientos, diseño, codificación, revisión y documentación. Al final de la misma, se pretende obtener un código, aunque no completo, sí funcional (o, lo que es lo mismo, una demo parcial que debe hallarse libre de errores).

En 2001, diecisiete representantes y críticos de distintas metodologías y modelos, convocados por Kent Beck, presentaron el manifiesto ágil, cuya filosofía se resume en cuatro prioridades:

- Individuos e interacciones sobre procesos y herramientas
- Software que funciona sobre documentación exhaustiva
- Colaboración con el cliente sobre la negociación de contratos
- Responder ante el cambio sobre el seguimiento de un plan

Aunque en el caso de este proyecto no podemos hablar de cliente, y por tanto puntos como la negociación de contratos carecen de sentido, sí existe la necesidad de un campo de colaboración y adaptación con el Grupo de Informática Biomédica. Por supuesto, la funcionalidad y la posibilidad del cambio cobran también una importancia fundamental, por lo que algunos de estos conceptos resultarán de gran utilidad.

4.2.2. TDD

Dentro del marco del agilismo, nos encontramos con una práctica que profundiza un poco más en el concepto de demos parciales. El desarrollo guiado por pruebas (Test–Driven Developement o TDD) es una práctica que requiere la escritura de pruebas previamente al inicio del desarrollo de cada iteración. Estas pruebas, conocidas como pruebas unitarias o "unit test", se centran en comprobar el correcto funcionamiento de un ejemplo (una posible petición del usuario, más concreta y específica que un caso de uso) que la iteración anterior no cumpliera. Como es obvio, en el resultado final se aprovechará el código de la iteración anterior mediante técnicas de refactorización. El algoritmo del TDD consta únicamente de tres pasos:

- Escritura de la especificación del requisito
- Implementación del código según dicho ejemplo
- Refactorización para eliminar duplicidad y hacer mejoras

Otras fuentes citan como paso necesario la comprobación de que el código no supera el ejemplo antes de la implementación del código, pero es lógico asumir que este punto es tenido en cuenta implícitamente en el propio desarrollo de ejemplos.

4.2.3. Conclusiones

Como resulta evidente, sería imposible poner en práctica todos los principios del agilismo y TDD en este proyecto de fin de carrera, para empezar porque se trata de técnicas diseñadas especialmente para grupos de trabajo, en las que se da suma importancia a algunos conceptos como que todo desarrollador sea en parte diseñador y tenga acceso a todo el software, o que el dueño del producto especifique únicamente qué quiere y nunca el modo en que lo desea ver reflejado, mientras que en este caso las decisiones del Grupo de Informática Biomédica tendrán una gran importancia en lo que aspectos como los detalles del producto final se refiere.

Además, sería probablemente poco eficiente programar pruebas unitarias para todos los ejemplos con los que tratará potencialmente. Sin embargo, no lo es tenerlos en cuenta, aunque sea a un nivel teórico a la hora de empezar a programar. Es decir, orientar el desarrollo al cumplimento de objetivos concretos con el objetivo de poder presentar resultados visibles en cortos periodos de tiempo.

En esta misma línea, y aunque se propone un diseño inicial perfectamente establecido, la división práctica en iteraciones sí resulta completamente útil, siendo además esencial la comunicación con el responsable del proyecto como paso final en cada una de ellas. En esta reunión se comprobará que realmente se están cumpliendo los objetivos pedidos y se abrirá la posibilidad de proponer modificaciones y mejoras.

Por tanto, pese a que no puede decirse que se esté empleando de forma estricta el desarrollo ágil de software o el diseño dirigido por ejemplos, sí se han tomado técnicas de estas metodologías de cara a tratar con las particulares condiciones de este proyecto, que cumple ciertas analogías con la programación para un cliente externo.

4.3. Arquitectura

El desarrollo del presente proyecto se ha llevado a cabo siguiendo el modelo de arquitectura en tres capas.

El modelo de arquitectura en tres capas consiste en la separación del código atendiendo a su propósito desde el punto de vista lógico. Así, quedarán definidas la capa de presentación, la capa de negocio y la capa de datos.

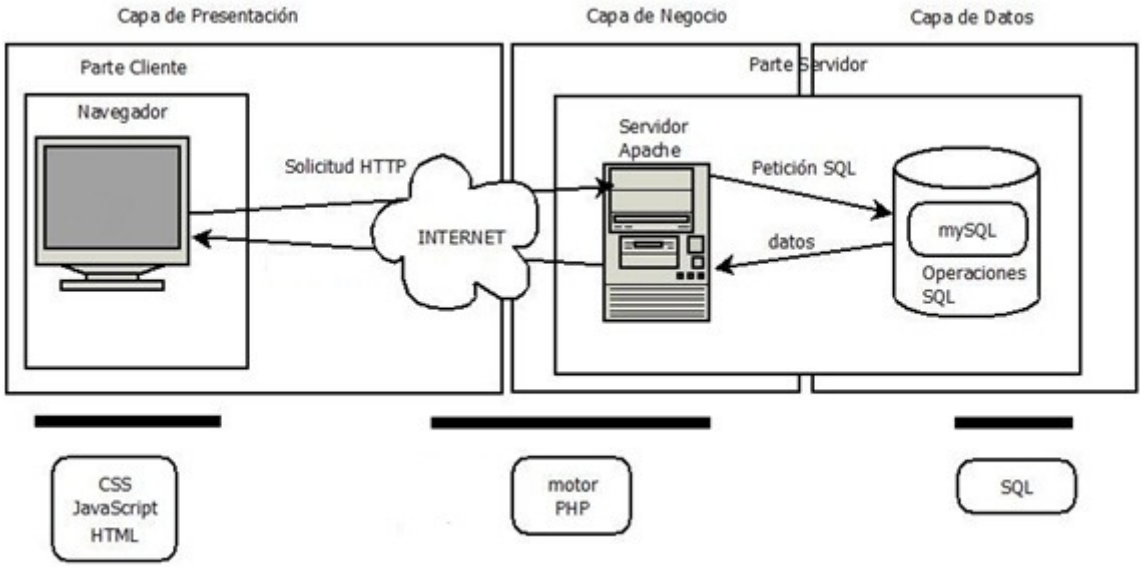


Fig. 2. Esquema conceptual de la arquitectura de la aplicación

La capa de presentación, también conocida como capa de usuario o interfaz gráfica, está constituida por la parte de la aplicación con la que el usuario interacciona directamente. Esta capa presenta al usuario la información necesaria dando un formato adecuado a los datos, y recolecta la que él proporcione a la aplicación sin llegar a entrar en el procesamiento de la misma. Las tecnologías implicadas en esta capa son: HTML, que define la estructura del sitio web, CSS, que proporciona los estilos que conferirán a la web su apariencia final y JavaScript, que realiza las operaciones necesarias en el navegador para ofrecer a la página aquellas funcionalidades gráficas que requieran una programación lógica.

La capa de negocio se centra simple y llanamente en el procesamiento de la información que recibe de la capa de presentación y la ejecución de todos los procesos necesarios para proporcionar una respuesta a la solicitud del usuario, que a su vez será devuelta a la capa de presentación. La tecnología correspondiente a esta capa en el presente proyecto es PHP, que proporciona un conjunto de funciones capaces de comunicarse con la capa de presentación y la capa de datos.

La capa de datos o persistencia es la encargada de realizar el acceso a los datos que posee la aplicación. Está formada por uno o más gestores de bases de datos que realizan todo el almacenamiento de datos y reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio. En esta capa se ha empleado una base de datos MySQL como tecnología, utilizándose el lenguaje SQL para realizar acciones sobre la misma.

La arquitectura escogida presenta varias ventajas, entre las cuales destaca la mayor facilidad para realizar cambios en el código, ya que cualquier cambio afectaría sólo a la capa oportuna. Esto facilitará el mantenimiento y soporte de la aplicación, requisito indispensable para el presente proyecto.

Otras ventajas son: una mayor flexibilidad ocasionada por la modularización del código, que minimizará los cambios en el código ya escrito a la hora de añadir nuevas funcionalidades; la posibilidad de realizar el desarrollo de las capas en paralelo, así como la de distribuir las capas en varios niveles; la obtención de aplicaciones más robustas debido al encapsulamiento; y, finalmente, un alto grado de escalabilidad que permita que la aplicación maneje más peticiones con el mismo rendimiento dependiendo sólo de la adición de nuevo hardware sin necesidad de modificar el código.

Finalmente, podemos decir que en este proyecto se ha optado por la implementación en un solo nivel, término que designa la distribución física de las capas.

Si la ubicación de cada capa se encuentra en un equipo distinto se habla de arquitectura de tres capas y tres niveles, mientras que si la distribución de las capas tiene lugar entre dos equipos (presentación y datos por separado, agrupando la lógica de negocios con cualquiera de ellos) se trata de una arquitectura de tres capas y dos niveles. Análogamente, si las tres capas están ubicadas físicamente en el mismo equipo como es el caso, hablamos de arquitectura de tres capas y un nivel.

4.3.1. Capa de presentación

La capa de presentación, como ya se ha explicado anteriormente, constituye la interfaz de usuario. El código de la capa de presentación está organizado en distintos archivos atendiendo al lenguaje en el que estén programados (HTML,

CSS o JavaScript) y a la función que realizan. Estos archivos son servidos por Apache en el servidor y se ejecutan en el navegador del equipo local del cliente.

Se ha intentado que la interfaz sea lo más amigable y sencilla de usar posible, así como que su presentación sea altamente adaptable a cualquier entorno. La estructura elegida para el layout se divide en: cabecera, cuerpo y pie de página.

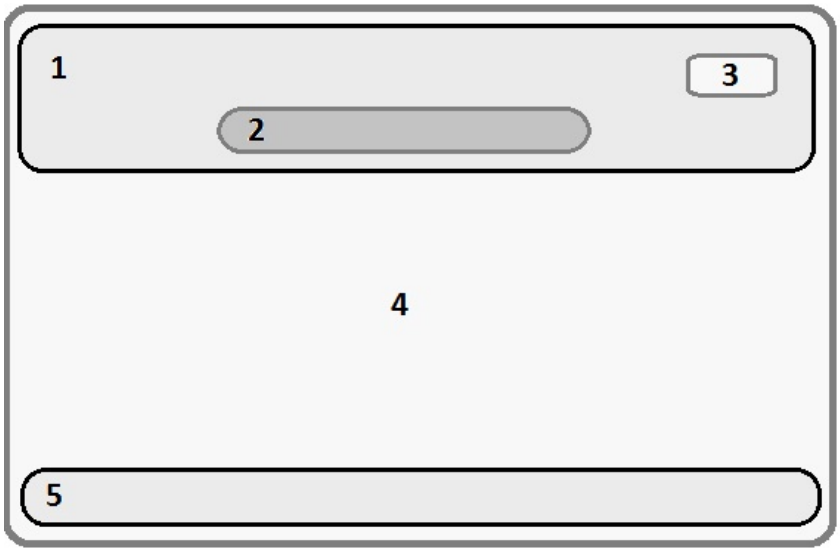


Fig. 3. Layout de la interfaz web

- **Cabecera**: La cabecera está compuesta por el banner (1), la sección de menú que permite la navegación a través de la web (2) y el link de login / logout (3), que permite identificarse ante el sistema. El diseño de la cabecera es redimensionable para ser capaz de adaptarse al tamaño de la ventana del navegador. Esta parte junto con el pie de página será común a todas las páginas, confiriendo a la web un aspecto homogéneo y reconocible.

- **Cuerpo**: El cuerpo de la página (4) es donde se mostrará el contenido web. Aquí aparecerá el texto inherente a cada sección, así como los formularios de entrada de datos en los casos que así se requiera. También se integrará el editor de contenido dentro del cuerpo de la página cuando se vaya a proceder a la administración del contenido.

- **Pie**: En el pie de página (5) solo se mostrará información referente al copyright y un vínculo a los términos de uso del sitio.

4.3.2. Capa de negocio

El código correspondiente a la lógica de negocio de la aplicación desarrollada puede dividirse en dos bloques: la parte servidor de la interfaz web y el planificador de trabajos.

La parte servidor de la interfaz web recoge la información introducida en los formularios de la capa de presentación, la valida, y la transfiere a la capa de datos donde queda registrada. También realiza la autenticación del usuario tras contrastar las credenciales introducidas por éste con las almacenadas en la capa de datos, y lleva a cabo el proceso de creación y destrucción de sesiones. Del mismo modo, realiza la transferencia de los archivos con los datos de imagen a ser procesados del equipo local del usuario al servidor en el que se encuentra volBrain, los descomprime, los prepara para ser procesados y almacena toda la información relativa al trabajo en la capa de datos.

El código de esta parte se ejecuta en el servidor bajo demanda del usuario, principalmente cuando éste realiza acciones a través de la interfaz web desde un navegador ejecutándose en su equipo local. El código se encuentra distribuido en distintos archivos PHP atendiendo a la función que realiza, y estos archivos son servidos por Apache junto a los archivos de la capa de presentación.

A diferencia de la parte servidor de la interfaz web, el planificador solo interactúa con el sistema y se comunica con la capa de datos, pero nunca con la capa de presentación.

El planificador de trabajos está implementado en un solo archivo PHP, cuyo código implementa las funciones que debe realizar el planificador, además de las llamadas necesarias para su registro en el sistema como un servicio de Windows y su posterior administración como tal. Esto es así por la necesidad de que el planificador se comporte como un proceso demonio que se ejecuta constantemente en un segundo plano esperando a que llegue una petición de procesamiento, o bien supervisando el progreso de la petición en curso.

A grandes rasgos, la funcionalidad del planificador está distribuida dentro del código en tres funciones principales:

- **lanzarTrabajo()**: El planificador se asegura de que no haya ningún trabajo en ejecución consultando los flags apropiados en la capa de datos. Si hay alguna petición en marcha, sigue comprobándolo cada pocos instantes hasta que su ejecución finalice. Una vez el sistema esté libre, consulta de nuevo los flags para ver qué trabajos están preparados para su ejecución y selecciona uno atendiendo a una estrategia FIFO. A continuación ejecuta la herramienta volBrain y le indica el trabajo que debe procesar, tras lo que activa los flags que indican que el trabajo ya está en marcha. Finalmente, envía un correo electrónico al usuario que envió el trabajo para informarle del estado de su petición.

- **enviarInforme()**: El planificador consulta los flags de la lista de trabajos en la capa de datos y selecciona uno de entre los finalizados siguiendo una estrategia FIFO. Seguidamente procede a enviar un correo electrónico al usuario que lo envió al que se adjunta el informe de resultados y el link de descarga de los archivos intermedios generados. En caso de haberse producido algún error durante el procesamiento del trabajo, el correo informará de ello y solicitará al usuario que vuelva a enviar el trabajo para ser procesado de nuevo.

- **comprobarBloqueo()**: El planificador consulta la capa de datos para comprobar si hay algún trabajo en ejecución y, de haberlo, comprueba que no lleve en marcha más de 20 minutos (se ha escogido este intervalo por ser más del doble del tiempo esperado de ejecución). Si se ha superado este límite el planificador activa el flag de error en el trabajo en ejecución y termina la ejecución de volBrain.

El resto del código del planificador, tal y como se ha comentado anteriormente, está dedicado a su consolidación como servicio de Windows (ver anexo C).

Existe una última parte de la lógica de negocio digna de ser mencionada. Como se ha podido ver, el elemento que permite la coordinación entre la interfaz web y el planificador es la capa de datos. Estos dos componentes no interaccionan directamente el uno con el otro, sino que activan y consultan los flags que indican el estado de los trabajos de forma asíncrona para decidir cuándo actuar, haciendo la capa de datos las veces de canal de comunicación.

Esto nos lleva a considerar una decisión de diseño cuidadosamente sopesada: que sea la propia herramienta volBrain la que active los flags de finalización y error de procesamiento si algo va mal durante el análisis, dada la naturaleza asíncrona de la comunicación entre procesos implementada.

A tal fin, se han añadido las líneas de código necesarias al módulo b_segment.m de la herramienta volBrain (ver anexo C).

4.3.3. Capa de datos

La capa de datos está formada por una única base de datos, de nombre volbrain, alojada en el sistema de gestión de bases de datos MySQL. Esta base de datos se compone de tres tablas: *users*, *jobs* y *content*.

4.3.3.1. Tabla *users*

La tabla *users* almacena los datos correspondientes a los usuarios del sistema, la mayoría de los cuales son introducidos en el formulario de registro y no vuelven a ser modificados. A continuación se muestra la estructura de la tabla.

	Campo	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Extra
☐	**e-mail**	varchar(40)	utf8_general_ci		No	*None*	
☐	**password**	varchar(40)	utf8_general_ci		No	*None*	
☐	**firstname**	varchar(40)	utf8_general_ci		No	*None*	
☐	**lastname**	varchar(40)	utf8_general_ci		No	*None*	
☐	**institution**	varchar(40)	utf8_general_ci		No	*None*	
☐	**department**	varchar(40)	utf8_general_ci		No	*None*	
☐	**zipcode**	varchar(8)	utf8_general_ci		No	*None*	
☐	**country**	varchar(40)	utf8_general_ci		No	*None*	
☐	**web**	varchar(40)	utf8_general_ci		No	*None*	
☐	**allows_datause**	tinyint(4)			No	1	
☐	**appear_asuser**	tinyint(4)			No	1	
☐	**date_created**	date			No	*None*	
☐	**enabled**	tinyint(4)			No	*None*	
☐	**last_visit**	timestamp		on update CURRENT_TIMESTAMP	No	CURRENT_TIMESTAMP	ON UPDATE CURRENT_TIMESTAMP
☐	**hash**	varchar(40)	utf8_general_ci		No	*None*	

Fig. 4. Tabla *users*

- El campo **e-mail**, que es la clave primaria de la tabla, contiene la dirección de correo electrónico del usuario que hará las veces de identificador del mismo.

- El campo **password** contiene la contraseña que el usuario debe utilizar para autenticarse en el sistema cifrada con el algoritmo MD5.
- El campo **firstname** contiene el nombre del usuario.
- El campo **lastname** contiene el apellido del usuario.
- El campo **institution** contiene el nombre de la institución con la que colabora el usuario.
- El campo **department** contiene el nombre del departamento al que pertenece el usuario dentro de la institución ya indicada.
- El campo **zipcode** indica el código postal correspondiente a la localización geográfica de la institución.
- El campo **country** indica el país en que se encuentra la institución.
- El campo **web** indica, en caso de existir, la URL de la página web personal del usuario o de la institución o el proyecto en el que colabora.
- El campo **allows_datause** indica que el usuario consiente que los datos de imagen médica anonimizados que envíe a volBrain para su procesamiento sean incorporados a una base de datos de archivos de imagen destinada a mejorar el funcionamiento de la aplicación.
- El campo **appear_asuser** indica que el usuario consiente que el nombre de su institución, departamento y país, en caso de haberlos indicado en el formulario de registro, aparezcan en la pestaña VolBrain Users como referencia para el resto de usuarios.
- El campo **date_created** indica el momento de creación de la cuenta, correspondiente al instante de envío del formulario de registro.
- El campo **enabled** es un booleano que indica si la cuenta está habilitada. Este campo se encuentra a 0 por defecto cuando se crea la cuenta, y se pone a 1 cuando el usuario confirma su registro pinchando en el link que la aplicación le envía por correo electrónico. Cualquier cuenta que no tenga este valor a 1 será totalmente inoperativa a pesar de estar registrada en la base de datos.

- El campo **last_visit** indica la fecha de la última visita del usuario a la web. El valor de este campo se actualiza con el valor del instante actual cada vez que el usuario obtiene una sesión. También se actualiza cada vez que se realiza una modificación en el valor de alguno de los campos de su entrada en la tabla de usuarios.
- El campo **hash** es un valor aleatorio generado en el momento de la creación de la cuenta de usuario que servirá como parámetro para algunas operaciones tales como la confirmación del registro por parte del usuario.

Ninguno de los elementos de la tabla puede ser nulo, aunque algunos pueden contener una cadena vacía indicando que no hay ningún valor para ese campo.

Los campos que en todo caso deben contener un valor válido para un correcto funcionamiento del sistema son: *e-mail, password, institution, allows_datause, appear_asuser, enabled* y *hash.*

4.3.3.2. Tabla *jobs*

La tabla *jobs* es la tabla que contiene la información necesaria para que el planificador de trabajos pueda realizar su función. Cada entrada corresponde a un trabajo que volBrain debe procesar. A algunos de sus campos se les asigna un valor en el momento de creación de la entrada, es decir, cuando el usuario transfiere el archivo, mientras que otros se van actualizando dinámicamente conforme el proceso global va avanzando. Estos últimos son los que permiten la coordinación entre la lógica de negocio de la interfaz, el planificador de trabajos y la herramienta volBrain en sí, por lo que son de suma importancia.

	Campo	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Extra
	jobnumber	int(11)			No	*None*	AUTO_INCREMENT
	jobpath	varchar(256)	utf8_general_ci		No	*None*	
	filename	varchar(256)	utf8_general_ci		No	*None*	
	user	varchar(40)	utf8_general_ci		No	*None*	
	ready	tinyint(4)			No	0	
	launched	tinyint(4)			No	0	
	finished	tinyint(4)			No	0	
	error	tinyint(4)			No	0	
	sent	tinyint(4)			No	0	
	timestamp	timestamp			No	0000-00-00 00:00:00	
	age	int(11)			Sí	*NULL*	
	sex	varchar(8)	utf8_general_ci		Sí	*NULL*	

Fig. 5. Tabla *jobs*

- El campo **jobnumber** es el número identificador con el que se hará referencia al trabajo. Se le asigna automáticamente en el momento de la creación de la entrada y como puede verse en la tabla es un valor con autoincremento. Es la clave primaria de la tabla.

- El campo **jobpath** indica el directorio del sistema en el que se guardan todos los archivos correspondientes al procesamiento de ese trabajo.

- El campo **filename** indica el nombre del archivo comprimido enviado por el usuario.

- El campo **user** indica el identificador del usuario que ha enviado el trabajo para su procesamiento.

- El campo **ready** es un valor booleano que indica que el archivo enviado por el usuario ha sido preprocesado y está listo para ser elegido por el planificador para su ejecución. Su valor por defecto es de 0 en el momento de creación de la entrada y es el código de la parte servidor de la interfaz web el que lo pone a 1 una vez lo ha dejado listo para ser procesado.

- El campo **launched** es un valor booleano que indica que ha comenzado el procesamiento del archivo. Su valor por defecto es de 0, siendo el planificador de trabajos el que pone su valor a 1 tras lanzar el trabajo y que volBrain comience a procesarlo.

- El campo **finished** es un valor booleano que indica que los datos han terminado de ser procesados por volBrain. Es el propio código de la herramienta volBrain el que pone el valor del campo a 1 tras terminar de procesar los datos. De este modo el planificador de trabajos sabe que ya puede enviar el informe de resultados del análisis al usuario.

- El campo **error** indica que ha ocurrido un error durante el análisis de los datos. Su valor por defecto es de 0, y al igual que en el campo anterior es el propio código de volBrain el que pone su valor a 1 si esto sucede. El planificador de trabajos utiliza este campo para notificar al usuario del fallo por correo electrónico.

- El campo **sent** es un booleano indica que, tras la finalización con éxito del trabajo, el informe de resultados ha sido enviado al usuario. El planificador de trabajos cambia su valor a 1 tras enviar el correo con el informe al usuario, y a partir de ese momento se considera que el trabajo ha sido totalmente completado.

- El campo **timestamp** contiene el instante en el que se inicia el procesamiento del trabajo. Su valor es actualizado al valor del instante actual por el planificador de trabajos cuando empieza el análisis de los datos. Su función es llevar la cuenta del tiempo que un trabajo se encuentra en ejecución para evitar bloqueos en el sistema. Una de las funciones del planificador es supervisar el tiempo que lleva en marcha un trabajo y, de ser este más del doble de lo razonable, terminar la ejecución de volBrain y actualizar el campo error de la entrada a 1.

- El campo **age** contiene la edad del paciente al que corresponden los datos enviados para su análisis. Este campo no es obligatorio y tan solo cumple una función informativa de cara a profundizar en el estudio del análisis por tramos de edad.

- El campo **sex** contiene el sexo del paciente al que corresponden los datos para su análisis. Al igual que en el caso anterior no se trata de un campo obligatorio y su función es meramente informativa.

4.3.3.3. Tabla *content*

La tabla *content* contiene el texto que se muestra al usuario en las diferentes secciones de la página web. En un principio este texto se encontraba codificado en el propio archivo de la página, pero tras la inclusión del editor de contenido se encuentra almacenado en esta tabla.

	Campo	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Extra
☐	**tag**	varchar(40)	utf8_general_ci		No	*None*	
☐	**text**	text	utf8_general_ci		No	*None*	

Fig. 6. Tabla *content*

- El campo **tag** contiene la etiqueta con el nombre de la sección de la página web a la que pertenece el texto.
- El campo **text** almacena el código HTML con el contenido de la sección indicada en el campo anterior.

5. RESULTADOS Y CONCLUSIONES

5.1. Descripción del desarrollo

El desarrollo del presente proyecto ha tenido una duración de un curso académico a tiempo parcial. A pesar de que este lapso se encuentra dentro de los márgenes previstos, ha resultado ser mayor que el tiempo estimado a priori.

La etapa inicial del proyecto ha consistido básicamente en la formación de la alumna en las diferentes tecnologías y lenguajes utilizados, sobre los que no se poseían conocimientos previos.

El uso del método de prueba y error, así como las reformulaciones sucesivas del diseño según se ampliaban los conocimientos de la alumna han ralentizado notablemente el avance del proyecto. Sin embargo, se considera esto una parte necesaria y positiva del proceso que ha contribuido a consolidar los conceptos aprendidos.

A lo largo del desarrollo también han tenido lugar varios cambios en los requerimientos de la aplicación, que en ocasiones han implicado también un cambio radical del diseño, a medida que se evaluaban los resultados de los avances realizados.

Más allá de lo indicado, se considera que el desarrollo del proyecto ha tenido lugar sin incidentes dignos de mención. Aunque a un ritmo más lento que lo esperado, el proceso ha evolucionado de forma adecuada hasta llegar a la obtención del producto final.

5.2. Evaluación de resultados

Teniendo en cuenta que se ha hecho uso de una metodología de desarrollo que divide el proceso en iteraciones, para la evaluación de resultados se ha diseñado una serie de pruebas de aceptación que permiten comprobar que el comportamiento de la aplicación es el esperado durante el desarrollo.

Existe una prueba de aceptación por cada flujo alternativo de cada uno de los casos de uso considerados en el apartado de análisis. En la prueba se reproduce el comportamiento de la aplicación ante la entrada de unos datos de ejemplo que cumplen las condiciones descritas en el flujo, a fin de comprobar que el estado final sea el especificado como postcondición en el caso de uso.

La ejecución de las pruebas de aceptación tiene lugar al final de cada iteración, aplicándose el subconjunto de pruebas pertinente según el bloque de código implementado. Una vez terminado el desarrollo de la herramienta, debe comprobarse de nuevo que se pasan todas las pruebas para asegurarse de que no hay problemas de integración.

En este caso, la evaluación final de todas las pruebas ha sido satisfactoria, lo que indica que la aplicación responde correctamente a todos los casos contemplados en su implementación.

En cuanto al tiempo de respuesta de la herramienta, se considera que el tiempo de respuesta de la interfaz es tan bajo como cabe esperar. El proceso de transferencia de archivos del equipo local del usuario al servidor es el único que no presenta un tiempo de respuesta constante, variando desde algunos segundos desde un equipo con una buena conexión a Internet, a varios minutos desde un equipo ubicado en el extranjero con una conexión a Internet de 128 Kbps.

El tiempo de procesamiento de los datos se encuentra ahora mismo en unos 8 minutos de media, siendo el envío del informe de resultados al usuario casi inmediato desde el momento en el que termina el procesamiento.

5.3. Conclusiones

Una vez superadas todas las fases del desarrollo, es necesario contrastar los objetivos planteados en la introducción con su cumplimiento por parte del producto obtenido.

El primer objetivo buscaba conseguir una interfaz de usuario de uso sencillo y amigable que posibilitara la interacción del usuario con el sistema. La interfaz implementada, clara e intuitiva, pone a disposición del usuario toda la información necesaria para su uso y cumple correctamente esta función.

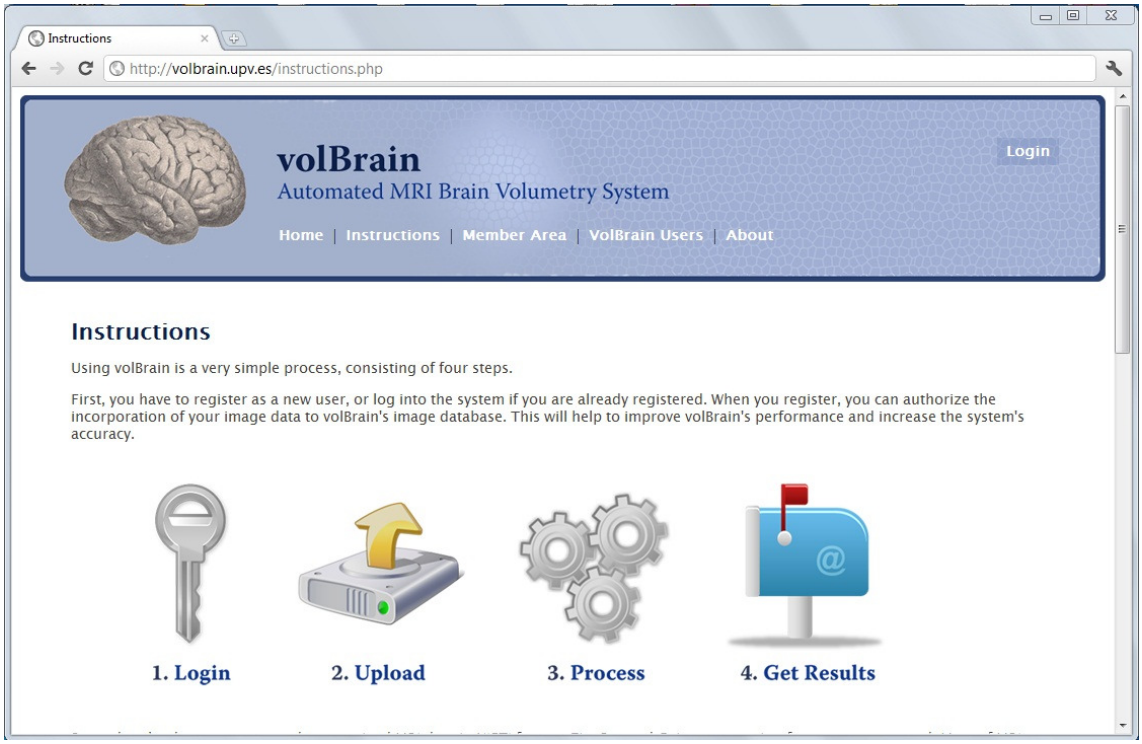


Fig. 7. Interfaz de la aplicación

Otro objetivo era conseguir una gestión de usuarios y datos que permitiera la autenticación ante el sistema, preparara los datos enviados para su procesamiento y registrara en la base de datos la información recibida. Tal y como puede comprobarse en el apartado “Diseño e implementación” de la memoria, todos estos aspectos han sido tenidos en cuenta y se han implementado correctamente, como confirma el resultado de las pruebas de aceptación realizadas sobre el producto final.

Finalmente, se planteaba la necesidad de un sistema de planificación de trabajos que gestionara y atendiera adecuadamente las peticiones recibidas. Tras un análisis detallado, este sistema ha sido implementado como un proceso demonio que realiza esta función de forma eficaz.

Por tanto, puede afirmarse que se ha conseguido desarrollar un sistema que proporciona un acceso remoto sencillo y eficiente a través de Internet a la herramienta volBrain.

5.4. Trabajo futuro

A pesar de que en este proyecto se han cubierto las necesidades básicas funcionales de la aplicación, todavía existen muchos aspectos que son susceptibles de ser considerados de cara a un desarrollo futuro.

Uno de los primeros aspectos a contemplar es el procesamiento de trabajos por lotes. Dado que uno de los atractivos de la aplicación es la capacidad de llevar a cabo el procesamiento de los datos de forma desatendida, agregar a la funcionalidad de volBrain la posibilidad de enviar un lote de trabajos y que sea el propio sistema quien se encargue de gestionar su ejecución aporta un valor añadido a tener en cuenta, ya que si se posee una gran cantidad de datos que deben ser procesados es mucho más tedioso y propenso a errores realizar el envío individual de los trabajos de forma manual.

Otra característica interesante que sería beneficioso añadir sería el paso de parámetros a volBrain a través de la interfaz de usuario. Según lo indicado en estos parámetros, la parte servidor de la aplicación se encargaría de que volBrain realizara un tipo de análisis más acorde a las necesidades individuales de cada usuario.

Un aspecto radicalmente distinto a implementar es el de análisis de tráfico y estadísticas. El propósito de este análisis es el contraste de los servicios que ofrece la web a los usuarios frente al uso que se hace de ellos, para evaluar tanto su adecuación como su rendimiento.

Con este fin puede llevarse a cabo la monitorización de diversos parámetros, tales como: número de visitas, número de usuarios, número de peticiones aceptadas por el servidor, número de trabajos por usuario, tiempo de permanencia en el sitio web, etc. Esto puede implementarse de forma manual, o bien hacer uso de alguna de las múltiples herramientas disponibles para tal propósito como puede ser Google Analytics.

Por último, en el caso de contar con un número elevado de usuarios se contempla la virtualización de los recursos y su replicación con el fin de ofrecer una mayor capacidad de cómputo y balancear la carga de trabajo entre los distintos servidores, mejorando la calidad de servicio.

La virtualización no sólo reduciría considerablemente los costes económicos, sino que optimizaría el aprovechamiento de todos los recursos hardware y facilitaría una rápida incorporación de nuevos recursos.

Esto conlleva un gran número de cambios en el sistema ya que, solo por el hecho de que la aplicación corra sobre un sistema operativo propietario, los costes de la replicación se dispararían. Por ello, deberían realizarse las modificaciones necesarias tanto en el código (especialmente el bloque del demonio que hace uso de llamadas que se comunican con el API de Windows) como en lo que al software utilizado se refiere para posibilitar el funcionamiento de la aplicación sobre un sistema operativo Unix.

ANEXO A. MANUAL DE USUARIO

En este anexo se describe con precisión el uso de la herramienta por parte de un usuario general.

El acceso a volBrain puede realizarse desde cualquier equipo que cuente con un navegador y conexión a internet. Para acceder a la interfaz web simplemente debe introducirse en el navegador la URL http://volbrain.upv.es/. Al entrar en la página se muestra la pantalla principal de volBrain, a través de la cual se puede navegar a cualquiera de los otros apartados de la web utilizando el menú de la cabecera.

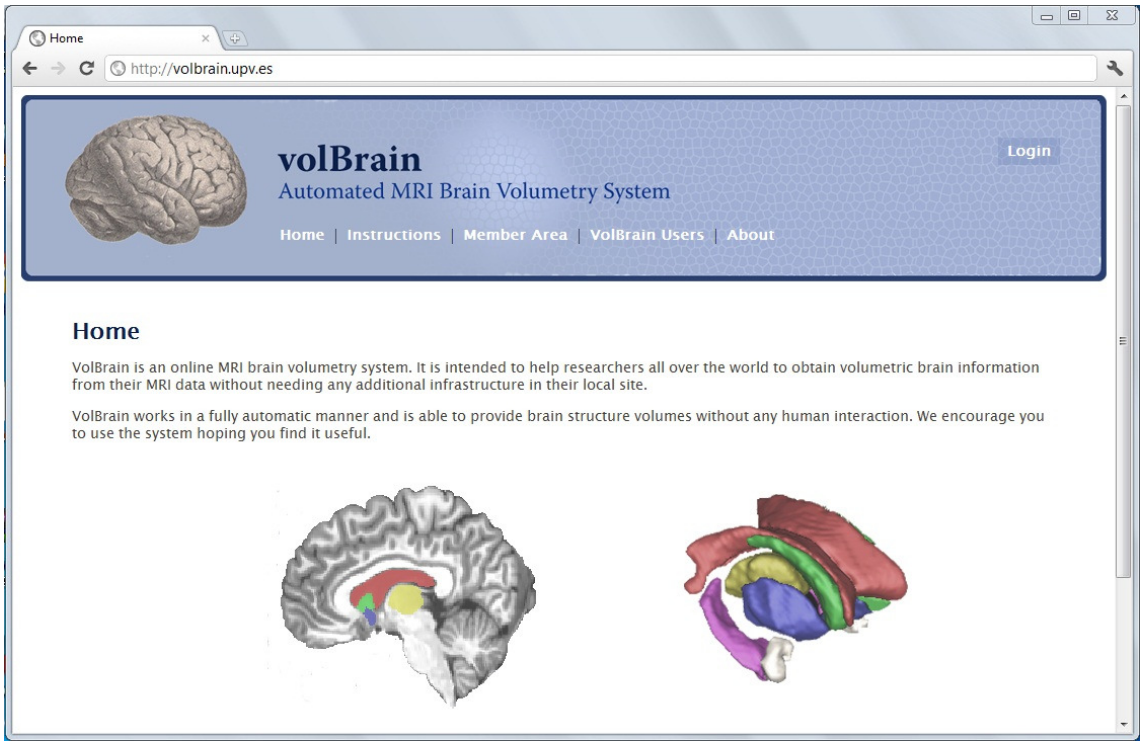


Fig. 8. Pantalla de inicio de volBrain

Todas las secciones son libremente accesibles a excepción de Member Area, cuyo propósito es el envío de archivos a volBrain para su procesamiento.

Para hacer uso de esta funcionalidad es necesario registrarse para convertirse en usuario del sistema y loguearse para obtener una sesión.

A.1. Registro

El registro como usuario se lleva a cabo pulsando el link de Login presente en la esquina superior derecha de la pantalla. Una vez aparezca el popup con el formulario de inicio de sesión, debe escogerse la opción "Click here to register".

La pantalla de registro cuenta con un formulario con campos para introducir la información que se almacena en la base de datos sobre cada usuario.

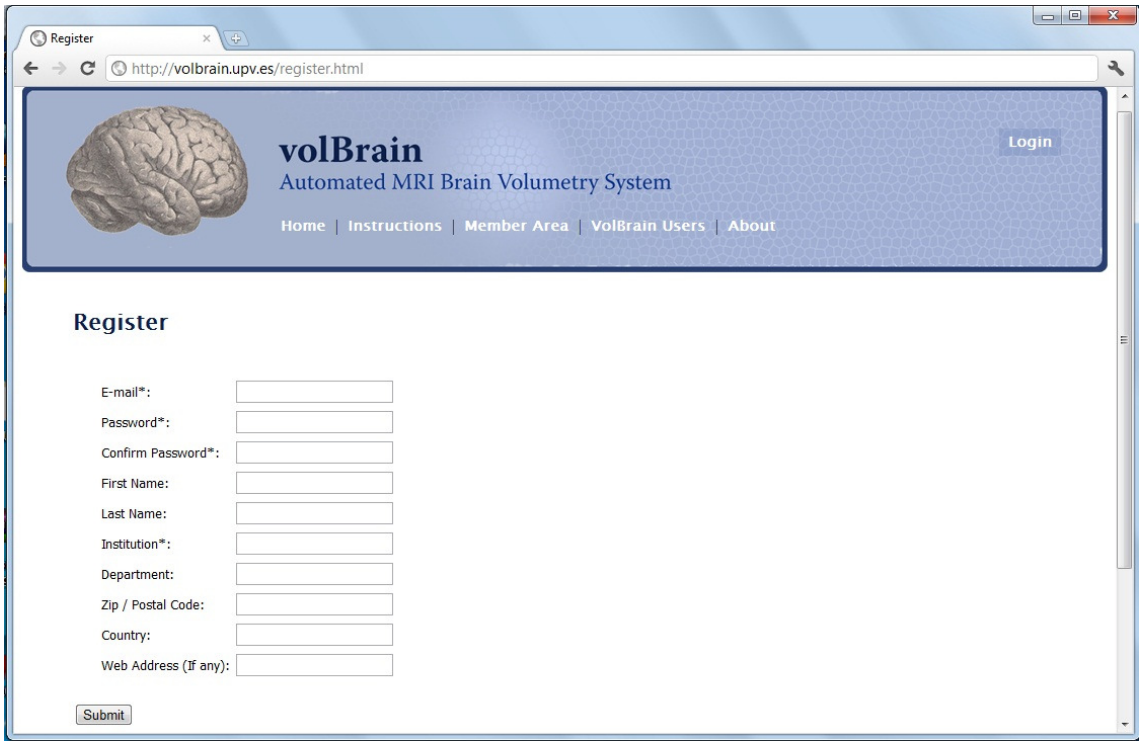


Fig. 9. Pantalla de registro de volBrain

A pesar de que todos los campos poseen nombres claramente descriptivos, es de interés indicar que campo E-mail se emplea como identificador de usuario en el sistema, además de ser la dirección de correo a la que se enviarán todas las notificaciones necesarias, y que la longitud mínima de la contraseña es de 8 caracteres. Nótese que los campos marcados con un asterisco son de carácter obligatorio.

En la parte inferior de la pantalla aparecen dos checkbox seleccionados por defecto que pueden ser deseleccionados si se desea. El primero, "I agree to my name and organization appearing in the Users section", indica que el

usuario accede a que el nombre y localización de la institución a la que pertenece aparezcan en la pestaña VolBrain Users y puedan ser vistos por cualquier usuario de la página. El segundo, "I authorize IBIME to keep my anonymized image data in a non-public database in order to improve volBrain's performance", indica que el usuario consiente que los datos de imagen que envíe para su procesamiento sean incorporados a la base de datos de conocimiento de volBrain a fin de mejorar el funcionamiento del sistema.

En caso de introducirse una dirección de correo ya registrada en la base de datos, de no coincidir la contraseña introducida y su confirmación o de no haberse introducido ningún valor en alguno de los campos obligatorios, al pulsar Submit se mostraría una pantalla de error informando del suceso.

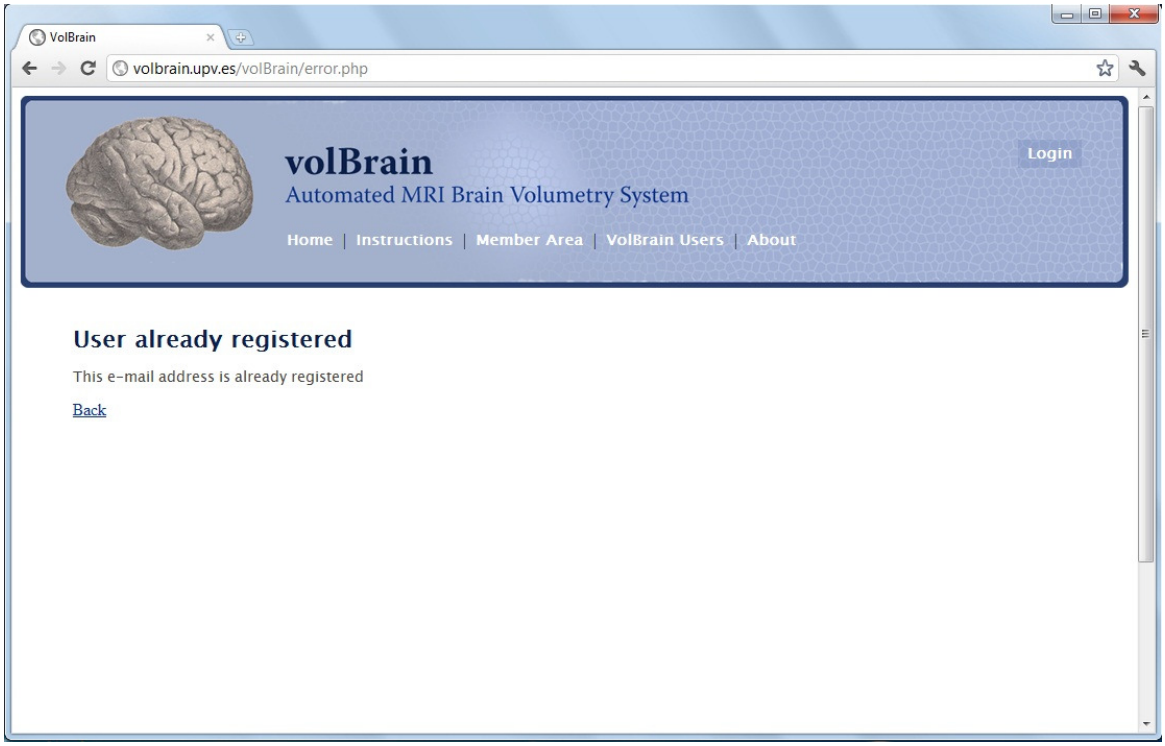


Fig. 10. Pantalla de error de volBrain

Si los datos que han sido introducidos tienen un formato correcto y se han completado todos los campos obligatorios, el sistema mostrará una pantalla informativa explicando que, para completar la operación de registro, se ha enviado un correo a la dirección indicada en el registro con un link de confirmación.

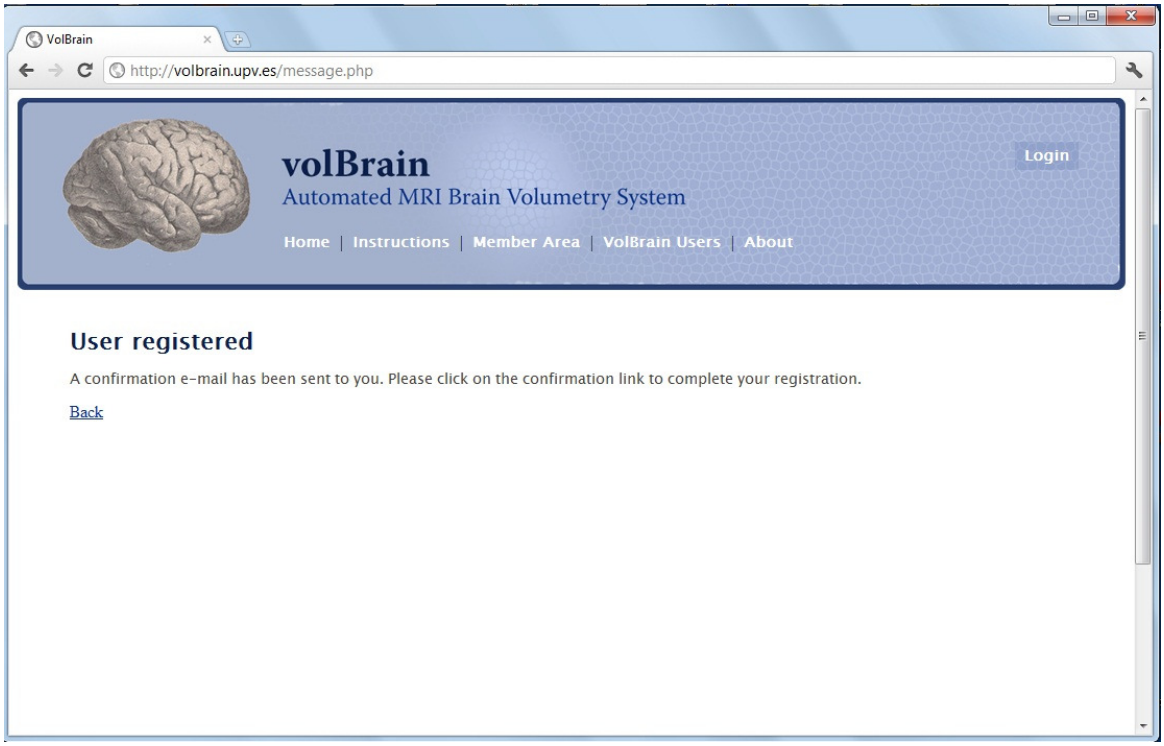


Fig. 11. Pantalla de confirmación de volBrain

El usuario debe acceder entonces a su cuenta de correo y pulsar en el link incluido, que le llevará a una pantalla en la que se confirmará que el registro se ha completado correctamente y la cuenta de usuario ya está habilitada.

A.2. Inicio y cierre de sesión

Para iniciar una sesión basta con pulsar el link Login de la parte superior derecha de la pantalla, introducir la dirección de correo y contraseña indicadas en la operación de registro y pulsar Submit.

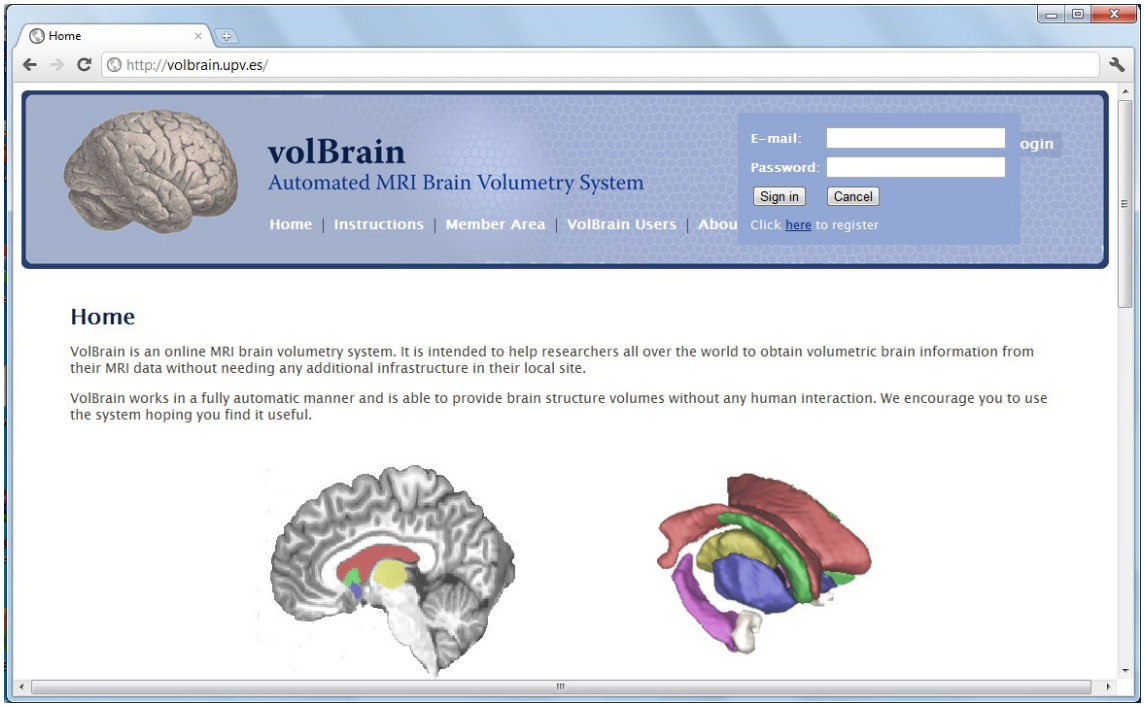


Fig. 12. Pantalla de login de volBrain

En caso de que las credenciales fueran erróneas se indicaría mediante una pantalla de error. Sin embargo, si la autenticación ante el sistema ha tenido lugar de forma satisfactoria podrá comprobarse que el link Login ha sido sustituido por el link Logout, señal inequívoca de que el usuario ha obtenido con éxito una sesión.

Para cerrar la sesión en curso, basta con pulsar el link Logout que destruirá la sesión creada, e inmediatamente será sustituido de nuevo por el link Login.

A.3. Envío de datos para su procesamiento

Una vez obtenida una sesión, ya es posible enviar datos de imagen a volBrain para que éste los procese. Esto puede ser hecho desde la pestaña Member Area, que muestra un formulario en el que introducir la ruta del archivo a enviar.

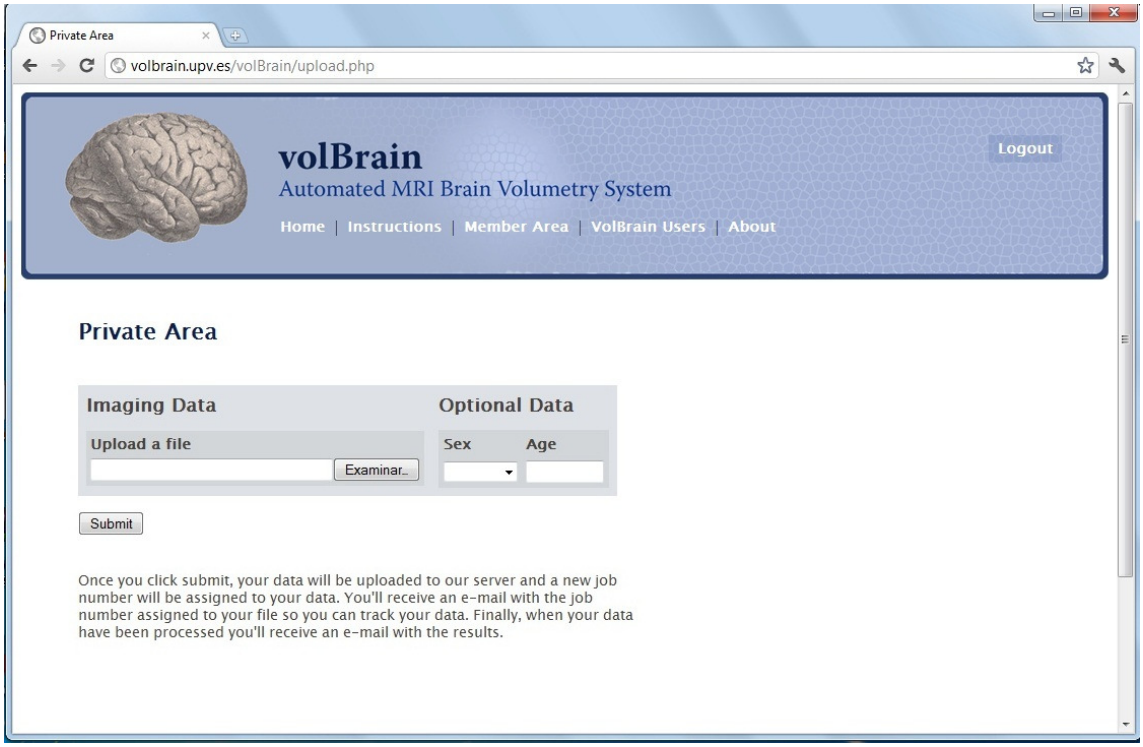


Fig. 13. Área privada de volBrain

El archivo enviado debe ser un archivo comprimido en formato RAR, ZIP o GZIP. La aplicación no aceptará otros formatos ni un tamaño de archivo mayor a 50Mb, mostrándose un mensaje de error en caso de que se intente enviar un archivo que no cumpla estas condiciones. Así mismo, el contenido del archivo comprimido debe ser obligatoriamente un archivo de datos imagen anonimizado en formato NIFTI.

En el apartado Optional Data aparece un desplegable en el que se puede seleccionar el sexo del paciente y un campo en el que indicar su edad. Estos datos no son necesarios para el procesamiento del archivo enviado, pero su inclusión facilitará la realización de posteriores estudios sobre los mismos.

Tras completar los campos, el botón Submit inicia la transferencia del archivo al servidor, que puede tardar algunos instantes en ser completada. Cuando esto suceda, se mostrará una pantalla indicando que la transferencia se ha completado con éxito y que los resultados del procesamiento serán enviados por correo electrónico al usuario.

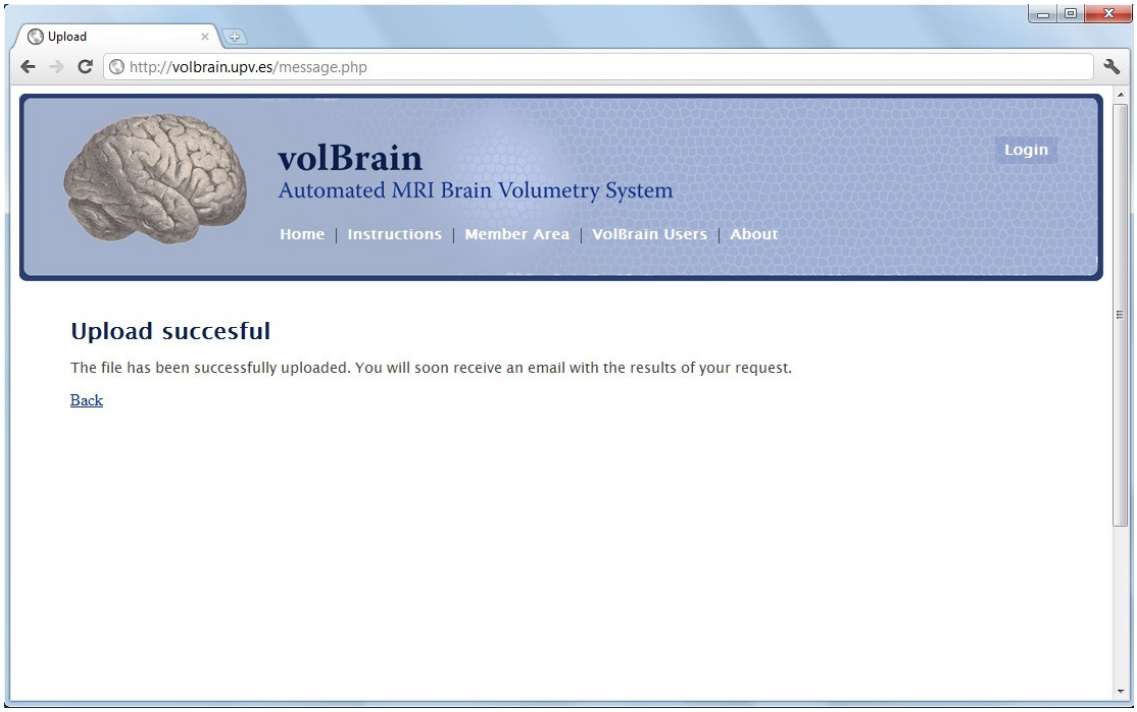
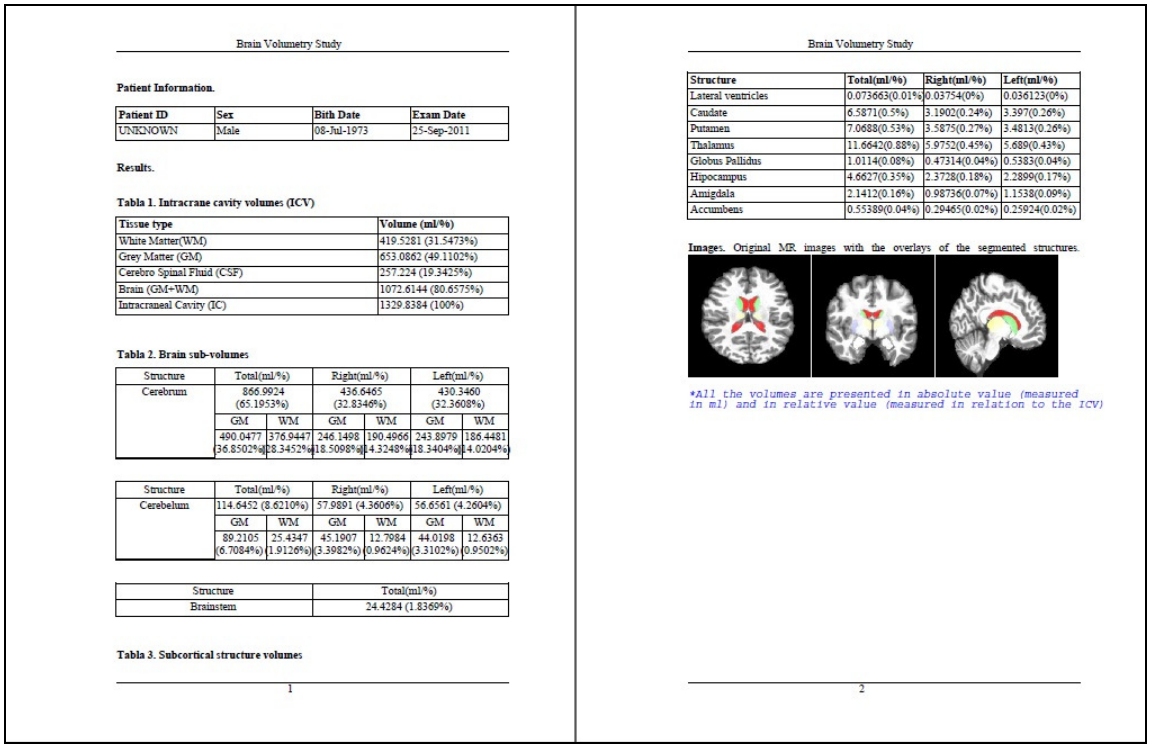


Fig. 14. Mensaje de finalización de transferencia de volBrain

El archivo transferido será descomprimido y preparado para ser procesado por la parte servidor de la aplicación. Una vez el planificador de trabajos de volBrain haya seleccionado el trabajo y los datos comiencen a procesarse, el usuario recibirá un correo electrónico informando del suceso, así como del número de identificador asignado al trabajo para poder referenciarlo en caso de acontecer alguna incidencia.

Si sucediese algún error durante el procesamiento de los datos, también se informaría por correo electrónico al usuario y se le indicaría los pasos a seguir.

Al completarse el procesamiento, se enviará al usuario un correo con un informe en formato PDF como archivo adjunto con los resultados del análisis de los datos enviados.



Brain Volumetry Study

Patient Information.

Patient ID	Sex	Bith Date	Exam Date
UNKNOWN	Male	08-Jul-1973	25-Sep-2011

Results.

Tabla 1. Intracrane cavity volumes (ICV)

Tissue type	Volume (ml/%)
White Matter(WM)	419.5281 (31.5473%)
Grey Matter (GM)	653.0862 (49.1102%)
Cerebro Spinal Fluid (CSF)	257.224 (19.3425%)
Brain (GM+WM)	1072.6144 (80.6575%)
Intracraneal Cavity (IC)	1329.8384 (100%)

Tabla 2. Brain sub-volumes

Structure	Total(ml/%)		Right(ml/%)		Left(ml/%)	
Cerebrum	866.9924 (65.1953%)		436.6465 (32.8346%)		430.3460 (32.3608%)	
	GM	WM	GM	WM	GM	WM
	490.0477 (36.8502%)	376.9447 (28.3452%)	246.1498 (18.5098%)	190.4966 (14.3248%)	243.8979 (18.3404%)	186.4481 (14.0204%)

Structure	Total(ml/%)		Right(ml/%)		Left(ml/%)	
Cerebelum	114.6452 (8.6210%)		57.9891 (4.3606%)		56.6561 (4.2604%)	
	GM	WM	GM	WM	GM	WM
	89.2105 (6.7084%)	25.4347 (1.9126%)	45.1907 (3.3982%)	12.7984 (0.9624%)	44.0198 (3.3102%)	12.6363 (0.9502%)

Structure	Total(ml/%)
Brainstem	24.4284 (1.8369%)

Tabla 3. Subcortical structure volumes

1

Brain Volumetry Study

Structure	Total(ml/%)	Right(ml/%)	Left(ml/%)
Lateral ventricles	0.073663(0.01%)	0.03754(0%)	0.036123(0%)
Caudate	6.5871(0.5%)	3.1902(0.24%)	3.397(0.26%)
Putamen	7.0688(0.53%)	3.5875(0.27%)	3.4813(0.26%)
Thalamus	11.6642(0.88%)	5.9752(0.45%)	5.689(0.43%)
Globus Pallidus	1.0114(0.08%)	0.47314(0.04%)	0.5383(0.04%)
Hipocampus	4.6627(0.35%)	2.3728(0.18%)	2.2899(0.17%)
Amigdala	2.1412(0.16%)	0.98736(0.07%)	1.1538(0.09%)
Accumbens	0.55389(0.04%)	0.29465(0.02%)	0.25924(0.02%)

Images. Original MR images with the overlays of the segmented structures.

*All the volumes are presented in absolute value (measured in ml) and in relative value (measured in relation to the ICV)

2

Fig. 15. Informe de resultados de volBrain

Así mismo, el correo contendrá un link desde el que podrán descargarse algunos de los archivos intermedios generados durante el proceso de análisis por si fueran de interés para los fines del usuario.

ANEXO B. MANUAL DE ADMINISTRACIÓN

El presente anexo tiene como fin hacer las veces de manual de instalación, configuración, administración y mantenimiento de la aplicación. En él se detallan los pasos a seguir para la instalación y puesta en marcha de la misma desde cero en un equipo servidor, y se espera que ofrezca soporte al administrador del sistema en caso de cualquier contingencia que vaya desde un simple problema de funcionamiento a una restauración completa del sistema.

B.1. Instalación

Para la instalación de la aplicación se asume como único requisito previo que el equipo servidor cuente con un sistema operativo Windows de 64 bits y la herramienta Matlab ya instalados, condición necesaria para que volBrain pueda ejecutarse.

B.1.1. Instalación de XAMPP

En primer lugar se procederá a la instalación del paquete de software XAMPP, que podemos encontrar en http://www.apachefriends.org/en/xampp.html y contiene el servidor HTTP Apache, el sistema de gestión de bases de datos MySQL y el intérprete del lenguaje de programación PHP, integrados junto a otras utilidades adicionales y parcialmente configurados.

Se recomienda la utilización del instalador para Windows por ser el método más rápido y sencillo. Simplemente hay que ejecutar el instalador y seleccionar el directorio de instalación, preferiblemente el directorio raíz para evitar problemas de permisos.

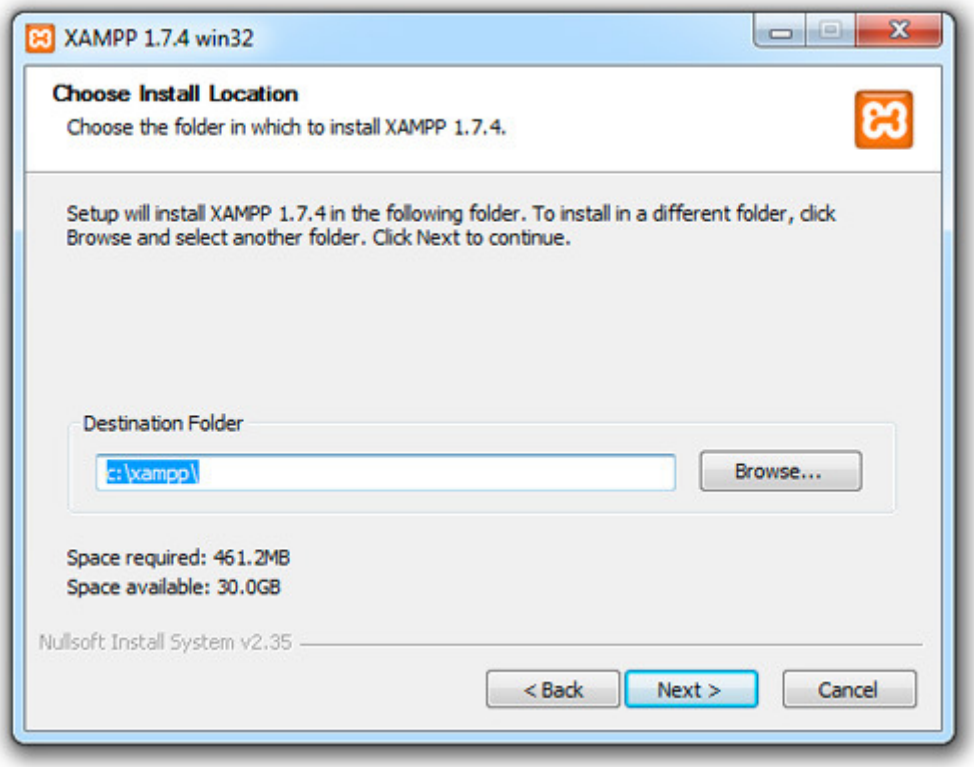


Fig. 16. Instalación de XAMPP

Una vez finalizado el proceso, en el directorio de instalación encontraremos varios ejecutables o archivos por lotes (según el caso) con nombres inequívocamente descriptivos que permiten iniciar o parar cada uno de los componentes de XAMPP por separado o bien el paquete completo.

Ejecutando el fichero xampp-control.exe accederemos a un panel de control que permite controlar los componentes de forma centralizada desde su interfaz gráfica.

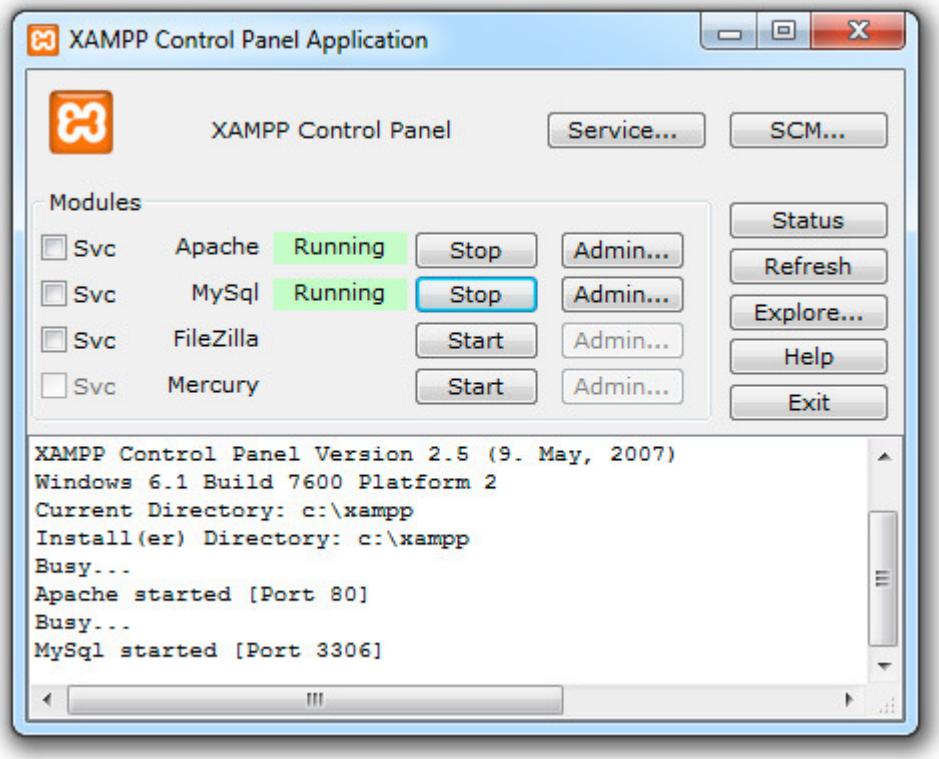


Fig. 17. Panel de control de XAMPP

En este caso, es conveniente instalar los componentes que van a ser utilizados para el funcionamiento de la aplicación como servicios del sistema, de modo que se inicien automáticamente durante el proceso de inicio del servidor, funcionando de manera continuada mientras no se indique lo contrario manualmente.

Esto se consigue marcando la casilla Scv (ver figura 17) que acompaña a cada componente en el panel de control de XAMPP, o bien haciendo uso del archivo *nombredelcomponente*_installservice.bat que encontraremos en la carpeta correspondiente a cada componente dentro del directorio de instalación.

Para comprobar que la instalación se ha completado satisfactoriamente han de iniciarse al menos los componentes Apache y MySQL y acceder a la dirección http://localhost/ o http://127.0.0.1/ donde, en caso de no haberse producido ningún problema, se mostrará una pantalla de bienvenida desde la que se puede comprobar el estado de los distintos componentes y su configuración.

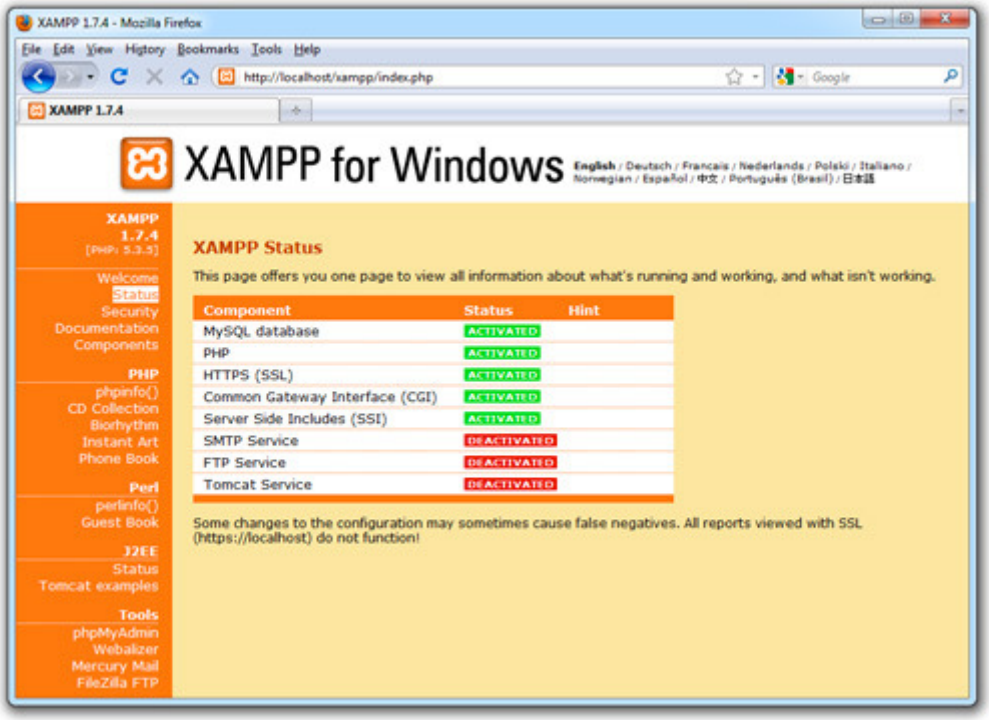


Fig. 18. Pantalla de estatus de XAMPP

Es posible que suceda que Apache se detenga inmediatamente cada vez que se intenta iniciarlo por problemas de compatibilidad con Windows 7 de 64 bits. De ser así, es posible solucionarlo editando el registro. Esto puede hacerse mediante la utilidad del sistema Ejecutar introduciendo el comando regedit. Tras abrirse el editor del registro, hay que descender por la estructura de directorios hasta "HKEY_LOCAL_MACHINE → SYSTEM → CurrentControlSet → services → HTTP". Una vez aquí, se pulsa el botón derecho del ratón sobre la parte del editor que muestra el contenido del directorio y se selecciona "Nuevo → DWORD (32 Bits)", al que se procede a asignarle el nombre NoRun. A continuación se edita el contenido de la entrada creada haciendo doble click sobre ella, asignándole un valor de 1 y pulsando Aceptar. Para finalizar hay que reiniciar XAMPP e intentar iniciar Apache, que debería arrancar correctamente esta vez.

Otra posible causa mucho más común de la imposibilidad de iniciar Apache es la existencia de otro programa en ejecución que esté haciendo uso del puerto 80, como podría ser Skype. En este caso, es suficiente finalizar la ejecución del programa para que se resuelva el problema.

B.1.2. Instalación de la extensión Win32Service

Dado que el código de la aplicación desarrollada correspondiente a la parte del planificador de trabajos crea e instala éste como un servicio de Windows, es necesaria la instalación de la extensión win32service de PHP, que permite a PHP comunicarse con el Administrador de Control de Servicios de Windows para iniciar, detener y registrar servicios, e incluso posibilita la ejecución de scripts como tales.

Para su instalación debe descargarse de alguna fuente fiable la librería php_win32service.dll, y ubicarla en el directorio de extensiones de PHP dentro del directorio de instalación de XAMPP (C:\xampp\php\ext).

Una vez la librería se encuentre en su lugar, debe editarse el fichero php.ini presente en el directorio de PHP. Dentro de la sección [PECL] agregar la siguiente línea:

```
extension=php_win32service.dll
```

Tras esto debe procederse al reinicio de XAMPP para que tenga efecto.

Para comprobar su correcta instalación, puede accederse mediante el navegador a la página de información del estado de los componentes de PHP desde el vínculo phpinfo() de la pantalla de bienvenida de XAMPP (ver figura 18). En la sección Configuration aparecerá la extensión win32service indicándose que está habilitada.

Otra opción más aconsejable, puesto que mostrará el tipo de error que se ha producido en caso de haber alguno, es ejecutar una consola de Windows, acceder al directorio de PHP de XAMPP y consultar el repositorio de extensiones de PHP (PEAR) para que informe de los paquetes instalados de la siguiente forma:

```
>pear list
```

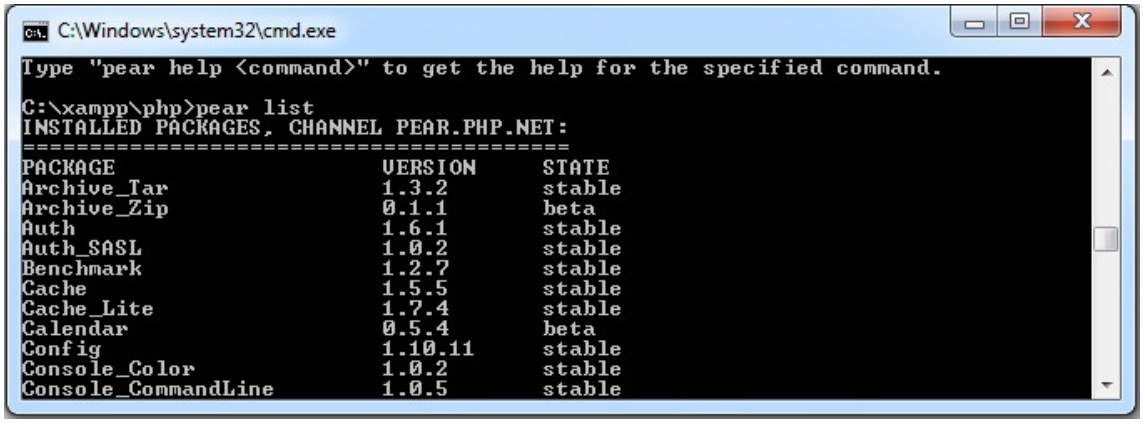


Fig. 19. Listado de extensiones de PHP

Es importante tener en cuenta que debe encontrarse una versión de la librería adecuada a la versión de PHP instalada, es decir, compilada con el mismo módulo y opciones. De no ser así, al listar los módulos instalados tal y como se ha indicado se obtendrá el siguiente error:

```
PHP Startup: win32service: Unable to initialize module
Module compiled with module API=20090626, TS, VC9
PHP compiled with module API=20090626, TS, VC6
These options need to match
```

La solución a esta situación sería tan sencilla como encontrar y descargar la versión apropiada de la librería.

B.1.3. Instalación de IIS

Las últimas versiones del sistema operativo Windows no llevan activado el componente Internet Information Services (IIS) que incorpora un servidor de correo SMTP, siendo ésta una herramienta de la que la aplicación tiene necesidad de hacer uso.

Su activación puede hacerse desde: "Panel de control → Programas → Activar o desactivar las características de Windows → Internet Information Services". Antes de aceptar hay que asegurarse de que todas las casillas del apartado Servicios World Wide Web estén seleccionadas.

Una vez activado, puede accederse al administrador de IIS desde: "Panel de control → Seguridad y mantenimiento → Herramientas administrativas → Administrador de Internet Information Services (IIS)" o bien desde la utilidad del sistema Ejecutar introducir el comando inetmgr.

B.1.4. Instalación de volBrain

La primera parte de la instalación de volBrain en el equipo es la instalación de la herramienta que realiza la segmentación automática en sí. Para ello, simplemente debe situarse la carpeta volBrain conteniendo todos los módulos y la estructura de directorios necesarios en el directorio raíz del servidor.

También será necesario descargar el conector J-Connector que permita al código Matlab hacer uso de las funciones de acceso a la base de datos MySQL desde http://www.mysql.com/downloads/connector/ y ubicarlo en el directorio de extensiones Java de Matlab (C:\Program Files\MATLAB\R2009a \java\jarext).

El siguiente paso es situar el código correspondiente a la interfaz web de la aplicación en el directorio raíz desde el que sirve las páginas el servidor Apache. Para ello, basta con situar en este directorio (C:\xampp\htdocs) los archivos con el código presentacional y funcional de la interfaz que se han desarrollado en el presente proyecto. Así mismo deberá existir en el mismo directorio una carpeta con el nombre imágenes que contenga todas las imágenes que se muestran en la parte cliente de la interfaz.

Por último, y para facilitar el acceso al mismo, situaremos el archivo demonio.php con el código del planificador de tareas en el directorio de PHP (C:\xampp\php).

La implementación del planificador de trabajos de volBrain incluye en su código las llamadas al sistema necesarias para su instalación como servicio en el sistema y su posterior administración.

Sin embargo, este registro debe realizarse manualmente mediante los comandos especificados en el código a través del intérprete de PHP, para lo que bastará el uso de una simple consola de sistema.

A través de la consola se accederá al directorio de PHP en el sistema, donde también se encuentra el archivo demonio.php con el código del planificador, y se procederá a su instalación con el comando:

```
>php.exe demonio.php install
```

Del mismo modo, en cualquier momento puede eliminarse el servicio del sistema mediante la orden:

```
>php.exe demonio.php uninstall
```

Finalmente, para poner en marcha el planificador y dejarlo corriendo, basta con ejecutar la orden:

```
>php.exe demonio.php start
```

Es **muy** importante tener en cuenta que, a pesar de que se mencione en el manual antes que otras acciones debido a su pertenencia al presente apartado, el registro del planificador de trabajos de volBrain como servicio sólo debe realizarse una vez se haya completado la configuración de todo el sistema.

B.2. Configuración

B.2.1. Configuración de PHP

La configuración de PHP se realiza principalmente en el archivo $\mathrm{php.ini}$ ubicado en el directorio de PHP. Este archivo contiene una serie de directivas que determinan el comportamiento del intérprete de PHP.

El archivo es editable con cualquier editor que trabaje con texto simple, y se encuentra dividido en secciones señaladas por una cabecera entre corchetes.

Cada directiva está formada por una pareja del tipo clave-valor sensible a las mayúsculas, y puede comentarse cualquier directiva del archivo colocando un punto y coma al inicio de la línea. Existen multitud de directivas, pero la mayoría

de ellas vienen configuradas por defecto en XAMPP con un valor adecuado a las necesidades de volBrain, de modo que se especificarán solo las modificaciones a realizar sobre la configuración inicial.

Principalmente tendremos que modificar las directivas correspondientes a las extensiones dinámicas de PHP (sección $[\mathrm{PECL}]$) y la gestión de errores (sección $[\mathrm{PHP}]$) a la configuración de correo (sección $[\mathrm{mail\ function}]$).

En la sección $[\mathrm{PECL}]$ hay que agregar las directivas:

```
extension=php_rar.dll
extension=php_win32service.dll
```

La primera ofrece la posibilidad de leer archivos en formato RAR y la segunda, cuya inclusión ya se había indicado en el apartado B.1.2., permite que PHP se comunique con el Administrador de Control de Servicios de Windows.

A su vez, en la sección $[\mathrm{mail\ function}]$ deben agregarse las directivas relativas a los parámetros de funcionamiento del servidor de correo SMTP cuando se utilice la función $\mathrm{mail}()$ de PHP:

```
SMTP = smtp.upv.es
smtp_port = 25
sendmail_from = volbrain@upv.es
```

En lo que respecta a la sección de errores, según nuestro objetivo hay dos configuraciones posibles.

Cuando se esté desarrollando la aplicación o realizando operaciones de mantenimiento, interesa mostrar los errores de ejecución por pantalla a un nivel tan estricto como sea posible, para lo que se utilizaría la siguiente configuración de directivas:

```
display_errors = On
display_startup_errors = On
log_errors = Off
html_errors = On
error_reporting = E_ALL | E_STRICT
```

Sin embargo, en condiciones normales en la que la aplicación está disponible para su uso por parte de los usuarios, lo interesante es no mostrar los errores por pantalla y guardar constancia de ellos en un log:

```
error_reporting = E_ALL & ~E_DEPRECATED
display_errors = Off
display_startup_errors = Off
log_errors = On
html_errors = Off
```

Finalmente y una vez finalizada la configuración, hay que indicarle a Windows que Apache tiene permiso para ejecutar un programa en PHP.

A fin de conseguirlo se introduce el comando services.msc en la utilidad del sistema Ejecutar y se selecciona el servicio Apache. Pinchando sobre él con el botón derecho del ratón y seleccionando la opción Propiedades, se abrirá un cuadro de configuración en el que debe seleccionarse la opción Permitir que los servicios interactúen con el escritorio en la pestaña Iniciar Sesión, y posteriormente reiniciar Apache para completar el proceso y que los cambios surjan efecto.

B.2.2. Configuración del servidor de correo SMTP

VolBrain necesita ser capaz de enviar correos electrónicos para poder comunicarse con el usuario de forma no interactiva. Para este fin, es necesario entregar los mensajes a un de correo SMTP que realice el envío.

La configuración relativa al servidor de correo SMTP se realiza desde el Administrador de IIS, al que se puede acceder tal y como se ha indicado en el apartado B.1.3.

En la sección ASP.NET de la vista principal del administrador se encuentra el elemento Correo electrónico SMTP.

Fig. 20. Administrador de Internet Information Services (IIS)

Una vez seleccionado este elemento, debe indicarse la dirección de correo electrónico del remitente de los mensajes que se envíen desde el servidor local (volbrain@upv.es), especificar el servidor SMTP al que se hará entrega de los mensajes para su envío (smtp.upv.es) y el puerto que utiliza (25), así como las credenciales del remitente en dicho servidor.

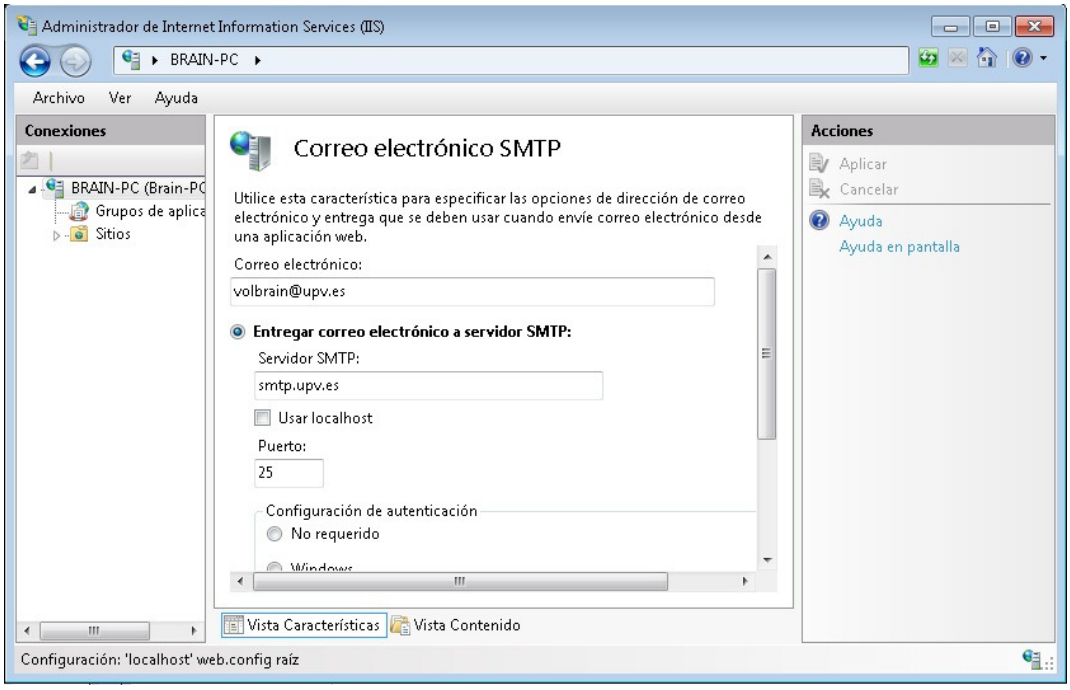


Fig. 21. Configuración del correo electrónico SMTP

B.2.3. Creación de la base de datos

La creación de la base de datos que utiliza volBrain puede realizarse de forma sencilla desde la herramienta gráfica de administración phpMyAdmin, incluida en el paquete XAMPP.

El acceso a esta herramienta se realiza desde cualquier navegador en el equipo servidor accediendo a la URL: http://localhost/phpmyadmin/.

Para facilitar la creación de la base de datos se ha creado el script volBrainDB.sql que contiene todas las sentencias SQL necesarias. De este modo, solo será necesaria su importación para que la estructura de la base de datos se cree de forma automática.

Sin embargo, antes debe crearse manualmente la base de datos en sí desde phpMyAdmin. Para ello, en la vista principal de phpMyAdmin debe introducirse el nombre que se quiera dar a la base de datos, seleccionar como cotejamiento utf8_genera_ci, y pulsar Crear.

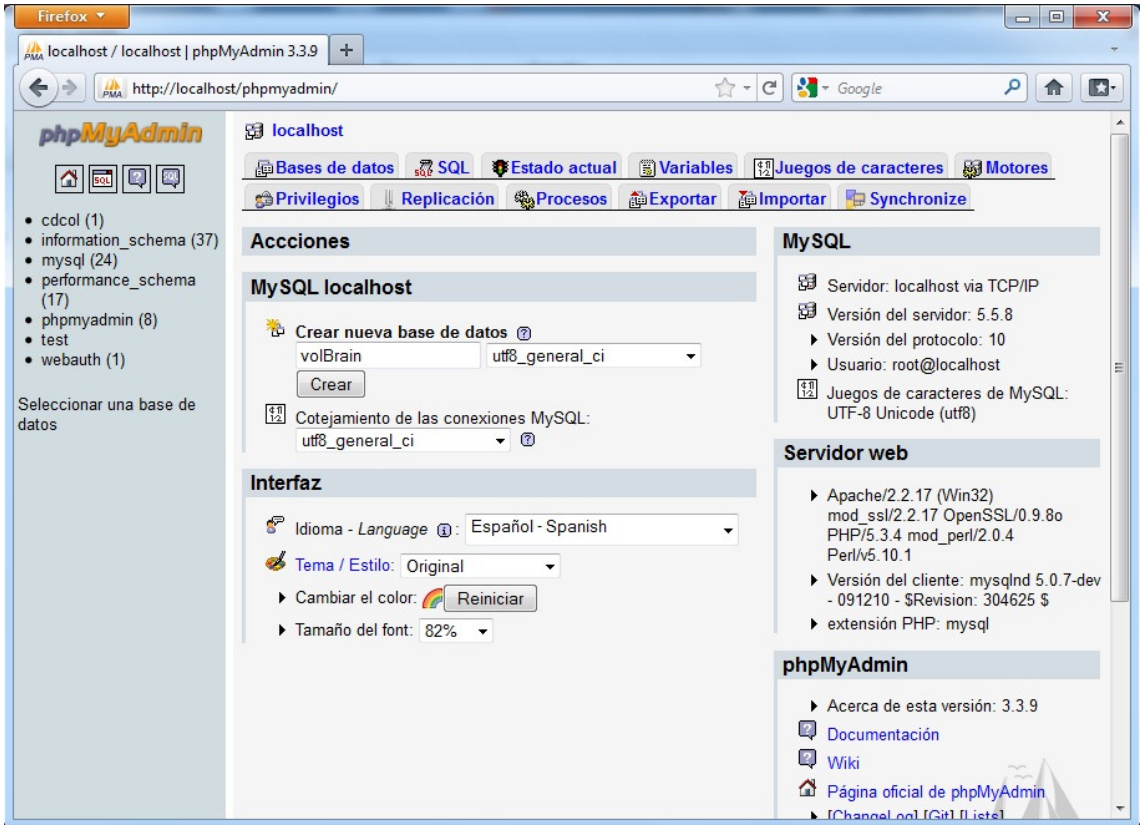


Fig. 22. Pantalla de inicio de phpMyAdmin

Una vez creada la base de datos, es posible seleccionarla en el menú de la izquierda y ejecutar el script desde la pestaña Importar. En esta pestaña sólo es necesario especificar la ruta del script, su formato y el juego de caracteres que utiliza y pulsar en Continuar, tras lo cual la base de datos tendrá la estructura deseada con las tablas y campos necesarios.

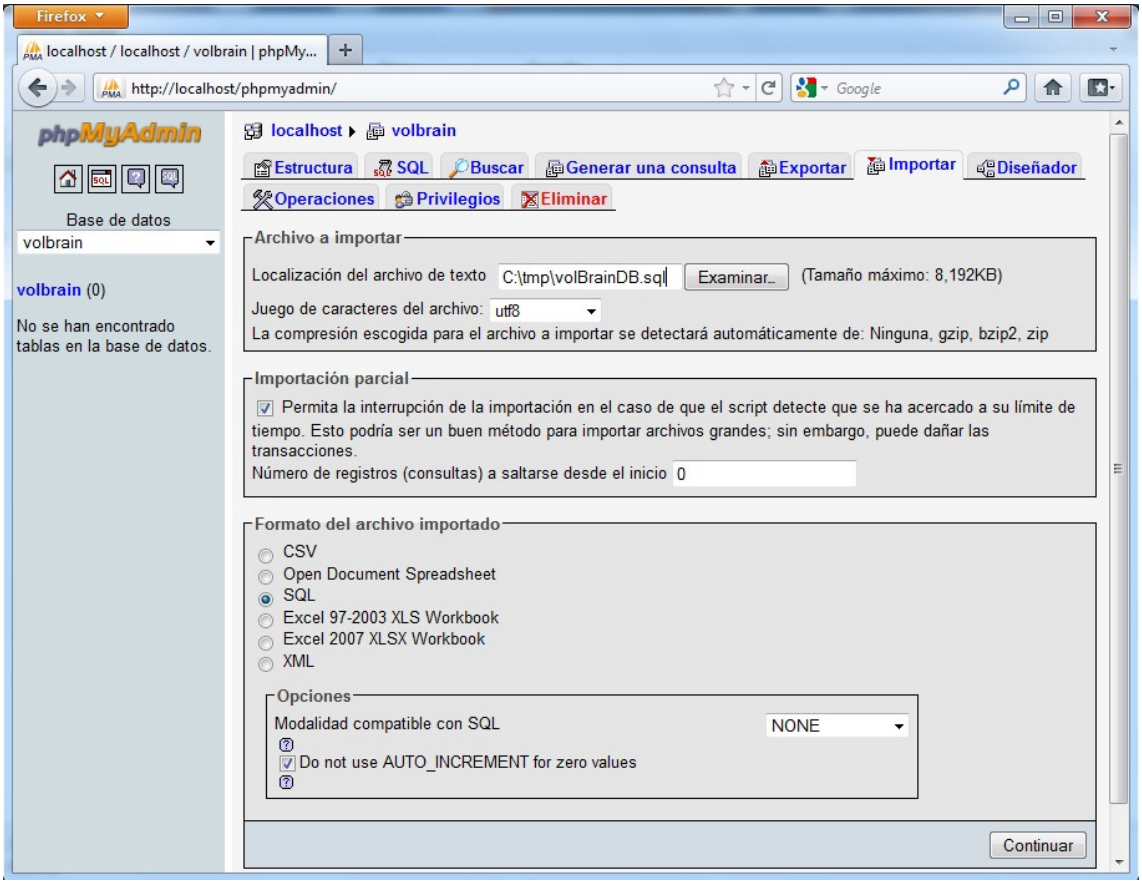


Fig. 23. Pantalla *Importar* de phpMyAdmin

A continuación se adjunta el código del script correspondiente a la creación de la estructura de la base de datos:

```
SET SQL_MODE="NO_AUTO_VALUE_ON_ZERO";
--
-- Base de datos: `volbrain`
-- --------------------------------------------------------
-- Estructura de tabla para la tabla `content`
--

CREATE TABLE IF NOT EXISTS `content` (
  `tag` varchar(40) CHARACTER SET utf8 NOT NULL,
  `text` text CHARACTER SET utf8 NOT NULL
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```

```
-- --------------------------------------------------------
--
-- Estructura de tabla para la tabla `jobs`
--

CREATE TABLE IF NOT EXISTS `jobs` (
  `jobnumber` int(11) NOT NULL AUTO_INCREMENT,
  `jobpath` varchar(256) NOT NULL,
  `filename` varchar(256) NOT NULL,
  `user` varchar(40) NOT NULL,
  `ready` tinyint(4) NOT NULL DEFAULT '0',
  `launched` tinyint(4) NOT NULL DEFAULT '0',
  `finished` tinyint(4) NOT NULL DEFAULT '0',
  `error` tinyint(4) NOT NULL DEFAULT '0',
  `sent` tinyint(4) NOT NULL DEFAULT '0',
  `timestamp` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00',
  `age` int(11) DEFAULT NULL,
  `sex` varchar(8) DEFAULT NULL,
  PRIMARY KEY (`jobnumber`)
) ENGINE=MyISAM  DEFAULT CHARSET=utf8 AUTO_INCREMENT=97 ;

-- --------------------------------------------------------
--
-- Estructura de tabla para la tabla `users`
--

CREATE TABLE IF NOT EXISTS `users` (
  `e-mail` varchar(40) NOT NULL,
  `password` varchar(40) NOT NULL,
  `firstname` varchar(40) NOT NULL,
  `lastname` varchar(40) NOT NULL,
  `institution` varchar(40) NOT NULL,
  `department` varchar(40) NOT NULL,
  `zipcode` varchar(8) NOT NULL,
  `country` varchar(40) NOT NULL,
  `web` varchar(40) NOT NULL,
  `allows_datause` tinyint(4) NOT NULL DEFAULT '1',
  `appear_asuser` tinyint(4) NOT NULL DEFAULT '1',
  `date_created` date NOT NULL,
  `enabled` tinyint(4) NOT NULL,
  `last_visit`  timestamp  NOT  NULL  DEFAULT  CURRENT_TIMESTAMP  ON
UPDATE CURRENT_TIMESTAMP,
  `hash` varchar(40) NOT NULL,
  PRIMARY KEY (`e-mail`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

B.2.4. Aspectos adicionales de la configuración

Para el acceso externo a los componentes instalados deberán abrirse los puertos utilizados por estos en el firewall de Windows. En este caso, y al hallarse el servidor dentro de la red de la UPV, será necesario solicitar al ASIC la apertura de los puertos 80 (Apache), 20 y 21 (FTP). El protocolo SMTP hace uso del puerto 25, pero no será necesario solicitar su apertura ya que tal y como se ha indicado anteriormente se hará entrega de los mensajes de correo electrónico que se desee enviar al servidor SMTP de la UPV.

B.3. Administración

En este apartado se indican algunos aspectos generales que es necesario conocer a la hora de administrar y mantener el sistema.

B.3.1. Archivo envvars.php

El archivo envvars.php ubicado en el directorio htdocs de Apache contiene los valores utilizados para las conexiones a la base de datos, así como el identificador correspondiente al usuario administrador.

De realizarse algún cambio en la estructura de tablas de la base de datos que supusiera un cambio en el valor de alguno de los parámetros utilizados para establecer una conexión, debería verse reflejado en este archivo para el correcto funcionamiento de la aplicación.

Así mismo, este archivo es el único lugar en el que es necesario realizar alguna modificación para cambiar de usuario administrador, bastando con indicar el identificador adecuado.

```
<?php

  $HOST="localhost";
  $DBUSER="root";
  $DATABASE="volbrain";
  $TABLEUSERS="users";
  $TABLEJOBS="jobs";
  $TABLECONTENT="content";
  $ADM_USER="volbrain@upv.es";

?>
```

B.3.2. Administración del planificador de trabajos

La administración del planificador de trabajos de volBrain puede realizarse a través de la consola del sistema o bien a través del administrador de servicios disponible en "Panel de control → Sistema y seguridad → Herramientas administrativas → Servicios"

Las opciones de administración se hallan implementadas en el propio código del planificador, de modo que para hacer uso de ellas debe hacerse a través del intérprete de PHP.

Para ello, se accede al directorio de PHP mediante una consola del sistema. Las distintas opciones de administración se mostrarán al ejecutar el archivo demonio.php, y la inclusión de cualquiera de las opciones en la línea de comandos como parámetro tendrá como resultado su ejecución.

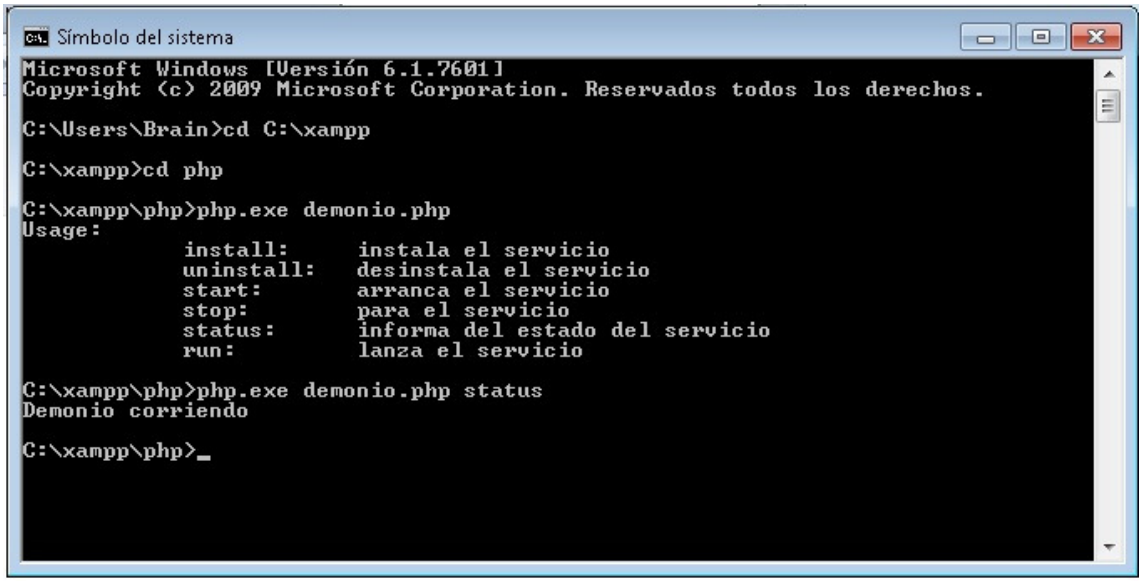


Fig. 24. Ejecución del archivo demonio.php

B.3.3. Administración de la base de datos

Cualquier modificación manual que sea necesario efectuar sobre la estructura o contenido de la base de datos de volBrain, es conveniente realizarla a través de la herramienta gráfica de administración phpMyAdmin ya mencionada con anterioridad.

Esta herramienta no sólo facilitará cualquier tarea, sino que evitará daños potenciales a la base de datos que podrían originarse a partir de su manipulación manual.

A pesar de que posee una interfaz altamente intuitiva, en caso de ser necesario puede encontrarse información más detallada sobre el manejo de la herramienta, su configuración y sus características en: http://wiki.phpmyadmin.net/pma/Welcome_to_phpMyAdmin_Wiki

B.3.4. Edición de contenido

El último aspecto general de la administración de la aplicación consiste en la actualización y mantenimiento del contenido informativo de la página web.

Para su edición, puede hacerse uso de la funcionalidad de la propia interfaz que permite al usuario administrador utilizar un editor WYSIWYG.

El proceso es muy sencillo, y basta con iniciar una sesión en la aplicación desde un navegador con las credenciales de usuario administrador. Una vez se ha obtenido la sesión, dentro de cada sección se mostrará, en lugar del texto habitual, un editor que permita manipular tanto el contenido como el formato del mismo. Para finalizar la edición es suficiente con pulsar el botón Submit, que registrará los cambios en la base de datos y recargará el contenido mostrado en pantalla, y comprobar que las modificaciones se siguen mostrando en el editor.

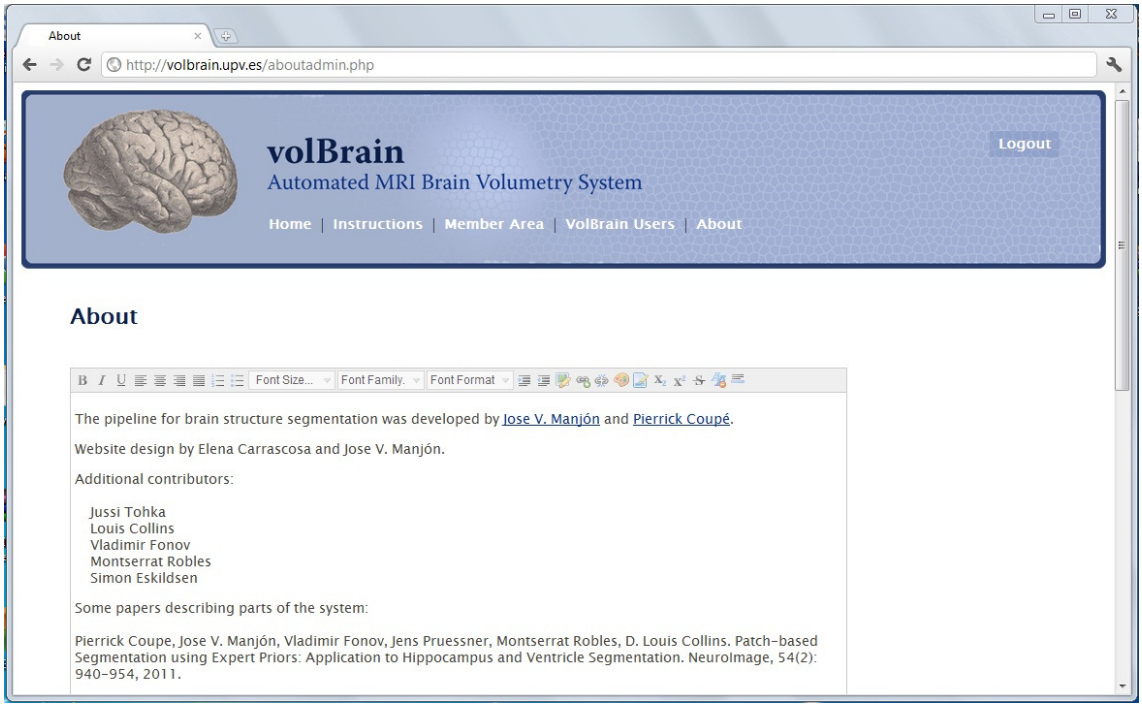


Fig. 25. Edición de contenido en volBrain

ANEXO C. CÓDIGO

En este anexo se recogen los fragmentos de código que se han considerado de especial interés en el desarrollo del presente proyecto.

C.1. Archivo *demonio.php*

```
<?php

//Comprobamos que este cargada la extension win32service de PHP, que
permite que PHP se comunique con el Service Control Manager para iniciar,
parar, registrar y eliminar servicios
//La extension (.dll) ha sido anyadida en el directorio /ext de PHP y se
ha anyadido su nombre en el fichero php.ini, en la seccion [PECL]

if(!extension_loaded('win32service'))
{
    dl('php_win32service.dll') or die('No ha sido posible cargar la
    extension php_win32service.dll');
}

$NombreServicio = "volBrain";
$NombreServicioMostrado = "VolBrainDispatcher";
$Host = "localhost";
$Usuario = "root";
$Password = "";
$DB = "volbrain";


function lanzarTrabajo()
{
   $HOST="localhost";
   $DBUSER="root";
   $DATABASE="volbrain";
   $TABLEUSERS="users";
   $TABLEJOBS="jobs";

   //Comprueba que no hay trabajos en ejecucion
   $ejecutandose =  mysql_num_rows(mysql_query("SELECT
   `jobnumber` FROM $TABLEJOBS WHERE (`launched` = 1 AND
   `finished`= 0)"));

   if (!$ejecutandose)
   {
      //Selecciona el trabajo a lanzar de entre los preparados
      $consulta = mysql_query("SELECT MIN(`jobnumber`) FROM
      $TABLEJOBS WHERE (`ready`= 1 AND `launched` = 0)");
      $resultado = mysql_fetch_array($consulta);
```

```
      //mysql_fetch_array() devuelve falso si la consulta no
      devuelve ninguna fila
      if ($resultado[0])
      {
         $jobnumber = $resultado[0];
         //Lanza el trabajo (si hay)
         $ruta_destino = 'C:\\volBrain\\results\\job'.$jobnumber;
         exec("C:\\xampp\htdocs\preproceso.bat $ruta_destino
         job$jobnumber");
         exec('C:\volBrain\modules\runseg.bat');

         //Lo marca como lanzado
         mysql_query("UPDATE $TABLEJOBS SET `launched` = 1 WHERE
         `jobnumber` = $jobnumber");

         //Anota el instante del lanzamiento
         mysql_query("UPDATE $TABLEJOBS SET `timestamp` = NOW() WHERE
         `jobnumber` = $jobnumber");

         //Buscamos el usuario que ha enviado el trabajo para
         informarle de que se ha lanzado
         $consultaB = mysql_query("SELECT `user` FROM $TABLEJOBS WHERE
         `jobnumber`= $jobnumber");
         $resultadoB = mysql_fetch_array($consultaB);
         $destinatario = $resultadoB[0];

         //Envia un correo informando de que el trabajo ha empezado a
         procesarse
         $from = "volbrain@upv.es";
         $to = $destinatario;
         $subject = "VolBrain Processing";
         $headers = "From: volbrain@upv.es";
         $message = "Dear user,\n\n VolBrain has started to process your
         data. In order to keep track of the files generated it has been
         assigned a job number, and its identifier from now on will be
         job$jobnumber.\n\n You will soon receive an e-mail informing
         about your request completion. The report will be included as an
         attachment, and a link to a file containing useful image data
         files generated in the processing will be provided.\n\n The
         volBrain Team";
         $message = wordwrap($message, 70);
         $sent = @mail($to, $subject, $message, $headers);
         if($sent)
         {
            echo "The e-mail informing that the job has been launched
            was successfully sent";
         }
         else
         {
            die("The e-mail informing that the job has been launched
            could not be sent.");
         }
         unset($resultado);
      }
      unset($jobnumber);
   }
   else { echo "Hay un proceso ejecutandose"; }
   sleep(1);
}
```

```
function enviarInforme()
{
   $HOST="localhost";
   $DBUSER="root";
   $DATABASE="volbrain";
   $TABLEUSERS="users";
   $TABLEJOBS="jobs";

   //Comprueba si hay trabajos terminados
   $finalizado =  mysql_num_rows(mysql_query("SELECT `jobnumber`
   FROM $TABLEJOBS WHERE (`finished` = 1 AND `sent`= 0)"));

   while ($finalizado)
   {
      //Selecciona un trabajo de entre los finalizados para
      informar por e-mail sobre el resultado
      $consulta1 = mysql_query("SELECT MIN(`jobnumber`) FROM
      $TABLEJOBS WHERE (`finished`= 1 AND `sent` = 0)");
      $resultado1 = mysql_fetch_array($consulta1);
      $jobnumber = $resultado1[0];

      //Consulta si el trabajo ha finalizado con error
      $consulta2 = mysql_query("SELECT `error` FROM $TABLEJOBS
      WHERE `jobnumber`= $jobnumber");
      $resultado2 = mysql_fetch_array($consulta2);
      $error = $resultado2[0];

      //Buscamos el usuario que envio el trabajo para informarle
      del resultado
      $consulta3 = mysql_query("SELECT `user` FROM $TABLEJOBS
      WHERE `jobnumber`= $jobnumber");
      $resultado3 = mysql_fetch_array($consulta3);
      $usuario = $resultado3[0];

      //Buscamos la ruta del trabajo para gestionar el envio de
      archivos
      $jobpath = mysql_fetch_array(mysql_query("SELECT `jobpath`
      FROM $TABLEJOBS WHERE `jobnumber`= $jobnumber"));
      $jobpath = $jobpath[0];

      $from = "volbrain@upv.es";
      $to = $usuario;
      $subject = "VolBrain Notification";
      $headers = "From: volbrain@upv.es";

      //Si se ha producido un error, se envia un correo al
      usuario informando del suceso y solicitando que lo vuelva
      a intentar
      if ($error)
      {
          $message = "Dear volBrain user,\n\n There was a problem
          processing your data, we apologize for the
          inconvenience. Please upload your files again.";
          $message = wordwrap($message, 70);
          $sent = @mail($to, $subject, $message, $headers);
          if($sent)
          {
             echo "The e-mail reporting the error was
             successfully sent";
```

```
        //Se marca el trabajo como enviado
        mysql_query("UPDATE $TABLEJOBS SET `sent` = 1 WHERE
        `jobnumber` = $jobnumber");
    }
    else
    {
        die("The e-mail reporting the error could not be sent.");
    }

    //Borramos los archivos correspondientes a ese trabajo que
    hayan podido quedar en el servidor
    `rmdir /S /Q $jobpath`;
}

//Si todo ha ido bien, envia un correo con el informe como
archivo adjunto
else
{
    //Preparacion previa de los archivos de resultados
    referenciados en el correo
    //Nombre del zip: jobname+timestamp.zip
    $timestamp = date(dmyHis);
    $zipname = $jobpath.'\job'.$jobnumber.$timestamp.'.zip';
    //Lista de todos los archivos a incluir en el zip
    $include = "$jobpath\mni_mf* $jobpath\\n_mni*
    $jobpath\hemilab* $jobpath\lab_n* $jobpath\csf* $jobpath\gm*
    $jobpath\wm* $jobpath\crisp*";
    //7za(programa compresor), a (add), -tzip (formato zip)
    $zipexecpath = 'C:\xampp\php\7za.exe';
    $zipexecpath a -tzip $zipname $include`;
    //Se suben los archivos al ftp
    $jobrepository = addslashes("C:\JobRepository");
    $ftppath = addslashes('C:\xampp\anonymous');
    $ftp = addslashes('ftp://volbrain.upv.es/');
    `move $zipname $ftppath`;
    $downloadlink = $ftp.'job'.$jobnumber.$timestamp.'.zip';
    $jobpath = addslashes($jobpath);
    $zipname = addslashes($zipname);
    mysql_query("INSERT INTO `prueba` (`jobrepository`,`ftppath`,
    `jobpath`,`downloadlink`,`zippath`,`ftp`) VALUES
    ('$jobrepository', '$ftppath', '$jobpath', '$downloadlink',
    '$zipname', '$ftp')");

    //Envio del mensaje
    $nl= "<br>";
     message = "Dear volBrain user, $nl You can find volBrain's
    report about your data as an attachement to this e-mail.$nl $nl
    You also can download some files related to the process at: $nl
    $nl $downloadlink $nl $nl These files will be deleted from our
    server one week from now. $nl $nl We wish you the best luck on
    your research, $nl The volBrain team";
    $semi_rand = md5(time());
    $mime_boundary = "==Multipart_Boundary_x{$semi_rand}x";
    $headers .= "\nMIME-Version: 1.0\n" .
    "Content-Type: multipart/mixed;\n" .
    " boundary=\"{$mime_boundary}\"";
    $message .= "This is a multi-part message in MIME format.\n\n" .
```

```
         "--{$mime_boundary}\n" .
         "Content-Type:text/html; charset=\"iso-8859-1\"\n" .
         "Content-Transfer-Encoding: 7bit\n\n" .
         $message . "\n\n";
         //Ruta al archivo
         $attachment = 'C:\\volBrain\\results\\job'.$jobnumber.
         '\\report.pdf';
         //Tipo de archivo
         $filetype = "application/pdf";
         //Nombre del adjunto
         $filename = "Report.pdf";
         $filedesc = fopen($attachment,'rb');
         $data = fread($filedesc,filesize($attachment));
         fclose($filedesc);
         $data = chunk_split(base64_encode($data));
         $message .= "--{$mime_boundary}\n" .
            "Content-Type: {$filetype};\n" .
            " name=\"{$filename}\"\n" .
            "Content-Transfer-Encoding: base64\n\n" .
            $data . "\n\n" .
            "--{$mime_boundary}\n";
         $message = wordwrap($message, 70);
         unset($attachment);
         unset($data);
         unset($filedesc);
         unset($filetype);
         unset($filename);

         $sent = @mail($to, $subject, $message, $headers);
         if($sent)
         {
            echo "The report was successfully e-mailed";
            //Se marca el trabajo como enviado
            mysql_query("UPDATE $TABLEJOBS SET `sent` = 1 WHERE
            `jobnumber` = $jobnumber");
         }
         else
         {
            die("The e-mail containing the report could not be
            sent.");
         }
      }
      //Comprueba si quedan trabajos terminados
      $finalizado =  mysql_num_rows(mysql_query("SELECT
      `jobnumber` FROM $TABLEJOBS WHERE (`finished` = 1 AND
      `sent`= 0)"));
   }
   sleep(1);
}

function comprobarBloqueo()
{
   $HOST="localhost";
   $DBUSER="root";
   $DATABASE="volbrain";
   $TABLEUSERS="users";
   $TABLEJOBS="jobs";
   $consulta4 = mysql_query("SELECT `jobnumber` FROM $TABLEJOBS WHERE
   (`launched`= 1 AND `finished` = 0 AND (TIMEDIFF(NOW( ),`timestamp`)
   > '00:20:00'))");
```

```
   //Si el proceso lleva mas de veinte minutos ejecutandose, lo marca
   como terminado con error y mata MATLAB
   if (mysql_num_rows($consulta4))
   {
      $resultado4 = mysql_fetch_array($consulta4);
      $bloqueado = $resultado4[0];
      mysql_query("UPDATE $TABLEJOBS SET `finished` = 1 WHERE
      `jobnumber` = $bloqueado");
      mysql_query("UPDATE $TABLEJOBS SET `error` = 1 WHERE `jobnumber`
      = $bloqueado");
      //Matar MATLAB
      TASKKILL /IM matlab.exe /T
   }
   sleep(1);
}

//Conexion a la base de datos
$Conexion = mysql_connect($Host,$Usuario,$Password) or die
(mysql_error());
mysql_select_db($DB, $Conexion) or die (mysql_error());

//Si se llama al servicio sin argumentos, mostramos un menu de ayuda
if (!isset($argv[1]))
{
   echo "Usage:
      install:\t instala el servicio
      uninstall:\t desinstala el servicio
      start:\t arranca el servicio
      stop:\t para el servicio
      status:\t informa del estado del servicio
      run:\t lanza el servicio
      ";
   exit();
}

//Instalar el servicio
if ($argv[1] == 'install')
{
   $estado = win32_query_service_status($NombreServicio);
   //Devuelve un array con la información del servicio o un código de
   error
   if($estado === 1060)
   //ERROR_SERVICE_DOES_NOT_EXIST 1060 (0x424)
   {
      //Crea una entrada para el servicio en el Service Control
      Manager (SCM)
      $creado = win32_create_service(array(
                  'service' => $NombreServicio,
            'display' => $NombreServicioMostrado,
            'description' => 'Demonio que extrae los trabajos
             destinados a volBrain de la cola de preparados y
             los lanza',
            'params' => __FILE__." run"
            ));
      //La constante __FILE__, indica la ruta completa y el
      nombre del archivo
      //El tipo de inicio por defecto es
      WIN32_SERVICE_AUTO_START
      $estado = win32_query_service_status ($NombreServicio);
```

```
      //Volvemos a comprobar el estado del servicio
      if($creado !== true)
      {
         var_dump($estado);
         //Si no se ha creado el servicio el script termina
         die('No ha sido posible crear el servicio');
      }
      else
      {
         echo "El servicio se ha creado con exito";
      }
   }
   exit();
}

//Desinstalar el servicio
else if ($argv[1] == 'uninstall')
{
   $borrado = win32_delete_service($NombreServicio);
   if ($borrado === WIN32_NO_ERROR)
   {
      echo "El servicio se ha desinstalado con exito";
   }
   else
   {
      echo "El servicio no se ha podido desinstalar";
   }
   exit();
}

//Arrancar el servicio
else if ($argv[1] == 'start')
{
   $iniciado = win32_start_service($NombreServicio);
   if ($iniciado === WIN32_NO_ERROR)
   {
      echo "El servicio se ha iniciado con exito";
   }
   else
   {
      echo "El servicio no se ha podido iniciar";
   }
   exit();
}

//Parar el servicio
else if ($argv[1] == 'stop')
{
   $parado = win32_stop_service($NombreServicio);
   if ($parado ===WIN32_NO_ERROR)
   {
      echo "El servicio se ha parado con exito";
   }
   else
   {
      echo "El servicio no se ha podido parar";
   }
   exit();
}
```

```
//Consultar el estado del servicio
else if ($argv[1] == 'status')
{
   $estado = win32_query_service_status($NombreServicio);
   if ( $estado['CurrentState'] == WIN32_SERVICE_STOPPED )
   {
      echo "Demonio parado\n";
   }
   else if ( $estado['CurrentState'] == WIN32_SERVICE_START_PENDING )
   {
      echo "Demonio esperando a ser iniciado\n";
   }
   else if ( $estado['CurrentState'] == WIN32_SERVICE_STOP_PENDING )
   {
      echo "Demonio esperando a ser parado\n";
   }
   else if ( $estado['CurrentState'] == WIN32_SERVICE_RUNNING )
   {
      echo "Demonio corriendo\n";
   }
   else if ( $estado['CurrentState'] == WIN32_SERVICE_CONTINUE_PENDING )
   {
      echo "Demonio esperando a continuar\n";
   }
   else if ( $estado['CurrentState'] == WIN32_SERVICE_PAUSE_PENDING )
   {
      echo "Demonio esperando a ser pausado\n";
   }
   else if ( $estado['CurrentState'] == WIN32_SERVICE_PAUSED )
   {
      echo "Demonio pausado\n";
   }
   else
   {
      echo "Demonio en estado desconocido\n";
   }
   exit();
}

//Si nos encontramos con cualquier otro parametro, volvemos a mostrar el
menu de ayuda
else if ($argv[1] != 'run')
{
   echo "Usage:
      install:\t instala el servicio
      uninstall:\t desinstala el servicio
      start:\t arranca el servicio
      stop:\t para el servicio
      status:\t informa del estado del servicio
      run:\t se emplea internamente para lanzar el servicio
      ";
   exit();
}

$registrado = win32_start_service_ctrl_dispatcher($NombreServicio);
//Registra el script con el Service Control Manager, de modo que pueda
actuar como un servicio con bajo el nombre indicado (se establecen
monitorizacion del servicio y facilidades de comunicacion)
```

```
if($registrado !== true)
{
   var_dump($estado);
   //Si no se ha creado el servicio, el script muere
   die('No ha sido posible registrar el servicio');
}

//Se indica al SCM que el servicio esta en marcha
win32_set_service_status(WIN32_SERVICE_RUNNING);

//Llamar periódicamente al SCM para ver si se ha pedido que se detenga el
servicio
while (win32_get_last_control_message() != WIN32_SERVICE_CONTROL_STOP)
{
   sleep(15);
   lanzarTrabajo();
   comprobarBloqueo();
   enviarInforme();
}
   //Si salimos del bucle, establecemos en el SCM que el
   servicio se ha parado
   win32_set_service_status(WIN32_SERVICE_STOPPED);
   exit();


?>
```

C.2. Archivo *b_segment.m*

El código adjunto corresponde a las líneas agregadas al archivo *b_segment.m* para la comunicación con la base de datos.

```
host = 'localhost';
user = 'root';
password = '';
db = 'volbrain';
jdbcString = sprintf('jdbc:mysql://%s/%s', host, db);
jdbcDriver = 'com.mysql.jdbc.Driver';
javaaddpath('C:\Program Files\MATLAB\R2009a\java\jarext\mysql-connector-java-
5.1.15-bin.jar')
conn = database(db, user, password, jdbcDriver, jdbcString);
if isconnection(conn)
   %Buscamos el trabajo en ejecucion (i.e. aquel lanzado pero no terminado)
   sqlquery = 'SELECT jobnumber FROM jobs WHERE launched=1 AND finished=0';
   curs = exec(conn, sqlquery);
   job = get(fetch(curs));
   jobnumber = job.Data{1};
   %Lo marcamos como terminado
   sqlquery2 = ['UPDATE jobs SET finished = 1 WHERE jobnumber =
   ',num2str(jobnumber)];
   curs2 = exec(conn, sqlquery2);
else
   disp(sprintf('Database connection failed: %s', conn.Message));
end
close(conn);
exit;

catch exception
%warning off;
host = 'localhost';
user = 'root';
password = '';
db = 'volBrain';
jdbcString = sprintf('jdbc:mysql://%s/%s', host, db);
jdbcDriver = 'com.mysql.jdbc.Driver';
javaaddpath('C:\Program Files\MATLAB\R2009a\java\jarext\mysql-connector-java-
5.1.15-bin.jar')
conn = database(db, user, password, jdbcDriver, jdbcString);
if isconnection(conn)
   %Buscamos el trabajo en ejecucion (i.e. aquel lanzado pero no terminado)
   sqlquery = 'SELECT jobnumber FROM jobs WHERE launched=1 AND finished=0';
   curs = exec(conn, sqlquery);
   job = get(fetch(curs));
   jobnumber = job.Data{1};
   %Lo marcamos como terminado
   sqlquery2 = ['UPDATE jobs SET finished = 1 WHERE jobnumber =
',num2str(jobnumber)];
   curs2 = exec(conn, sqlquery2);
   %Activamos el flag de error
   sqlquery3 = ['UPDATE jobs SET error = 1 WHERE jobnumber =
',num2str(jobnumber)];
   curs3 = exec(conn, sqlquery3);
else
    disp(sprintf('Database connection failed: %s', conn.Message));
end
close(conn);
exit;
end
```

Bibliografía

1. Ullman, Larry. *PHP 6 and MySQL 5 for dynamic web sites*. Upper Saddle River: Peachpit Press, cop. 2008.

2. Dave Shea and Molly E. Holzchlag. *The zen of CSS design: visual enlightenment for the web.* Berkeley: New Riders, cop. 2005.

3. Robbins, Jennifer Niederst. *Learning Web Design, Third Edition.* s.l. : O'Reilly Media, Inc., 2007.

4. Pilgrim, Mark. *HTML 5: up and running*. Sebastopol, Calif.: O'Reilly, 2010.

5. Meyer, Eric A. *Cascading style sheets: the definitive guide.* Beijing: O'Reilly, 2000.

6. Welling, Luke y Thomson, Laura. *PHP and MySQL Web Development, Fourth Edition.* s.l. : Addison-Wesley Professional, 2008.

7. Carey, Patrick. *Creación de páginas web con HTML.* Madrid: Paraninfo, cop. 2002.

8. Davis, Michele E. y Phillips, Jon A. *Learning PHP & MySQL, Second Edition.* s.l. : O'Reilly Media, Inc., 2007.

9. Pressman, Roger S. *Ingeniería del software, un enfoque práctico.* Santa Fe, México: McGraw Hill, 2006.

10. Blé Jurado, Carlos et al. *Diseño Ágil con TDD* [E- book]. Tenerife: Creative Commons License, 2010.

11. Pierrick Coupe, Jose V. Manjón, Vladimir Fonov, Jens Pruessner, Montserrat Robles, D. Louis Collins. *Patch-based Segmentation using Expert Priors: Application to Hippocampus and Ventricle Segmentation.* NeuroImage, 54(2): 940-954, 2011.

12. Consortium, World Wide Web. http://www.w3.org. *http://www.w3.org.* [En línea] 2010.

13. Wikipedia. http://en.wikipedia.org. *http://en.wikipedia.org*. [En línea] 2011.