UNIVERSIDAD DE ALMERÍA

ESCUELA SUPERIOR DE INGENIERÍA

DISEÑO E IMPLEMENTACIÓN DE UN PROTOTIPO DE HERRAMIENTA WEB PARA EL ENTRENAMIENTO ATENCIONAL EN NIÑOS Y NIÑAS CON TRASTORNO POR DÉFICIT DE ATENCIÓN/HIPERACTIVIDAD

PROYECTO FIN DE CARRERA

Ingeniero en Informática

Autor
José Rodríguez López

Director
Francisco Guil Reyes

Julio de 2011

Agradecimientos

Llegado a este punto y final de la carrera es cuando uno mismo siente que todo el esfuerzo realizado para llegar hasta aquí ha merecido la pena.

Son muchas las personas que de una manera u otra me han empujado a acabar, por fin, esta última etapa. Familiares y muchos amigos me han ayudado, aunque de forma muy especial mis padres y mi pareja. Ellos, siempre llenos de paciencia y seguros de mi, han estado siempre motivándome de muchas maneras para que no dejase de aprovechar esta oportunidad.

Dentro de la universidad, agradecer a mi director de proyecto, Francisco Guil, la guía, ayuda y motivación necesarias que me han sido imprescindibles durante todo el desarrollo del proyecto. También a Mayte, Fran y Raquel por su ayuda y aportaciones que sin duda han sido muy valiosas.

Muchas gracias a todos.

Índice

Capítulo 1 – Introducción.

El Trastorno por Déficit de Atención/Hiperactividad (TDAH) es la patología neuroconductual más común en la infancia. Se trata de un trastorno complejo que se corresponde con una triada sintomática caracterizada por *Hiperactividad* (hipercinesia), *Impulsividad* e *Inatención*; sintomatología que va a tener importantes repercusiones tanto en el desarrollo cognitivo como en la capacidad de aprendizaje y ajuste social del niño (*Cardo et al., 2010*). Los síntomas se presentan de modo heterogéneo en uno u otro paciente, con mayor o menor intensidad para uno u otro componente de la triada. De esta heterogeneidad se desprenden los tres subtipos del TDAH descritos en el *Manual Diagnóstico y Estadístico de los Trastornos Mentales (DSM-IV)*: (I) TDAH de tipo *Inatento*; (II) TDAH de tipo *Hiperactivo – Impulsivo*; y (III) TDAH de tipo *Combinado* (los síntomas de la triada se presentarían de forma más homogénea).

Los estudios realizados desde la aproximación de la Neurociencia Cognitiva, sugieren que el bajo rendimiento cognitivo que presentan los niños con TDAH podría ser subsidiario de un déficit primario en las denominadas "funciones ejecutivas", la cuales pueden definirse como las funciones encargadas de controlar y regular el procesamiento de la información para organizar la conducta dirigida hacia una meta, y

que dependen anatómicamente del establecimiento de conexiones de la corteza frontal con regiones del sistema límbico y con áreas cerebrales posteriores (*Romine & Reynolds, 2004*). Dentro de esta compleja constelación de funciones ejecutivas, los mecanismos atencionales juegan un papel importante, especialmente lo que desde el modelo atencional de *Michael I. Posner* se denomina Red de Control Ejecutivo o Atención Ejecutiva (*Posner, 1995*).

En esta memoria de proyecto fin de carrera se describen el análisis, diseño y aspectos destacados de la implementación de un prototipo de herramienta web para el entrenamiento de la Atención Ejecutiva en niños y niñas con trastorno por Déficit de Atención/Hiperactividad. Con este prototipo, los terapeutas que traten a esta población podrán realizar un seguimiento personalizado mediante sesiones en las que realizarán diversas tareas según un plan de trabajo establecido. Estos planes de trabajo (protocolos) podrán ser comunes entre diversos sujetos con el mismo tipo de trastorno o completamente individualizados según criterio experto

La tecnología web que se ha utilizado para el desarrollo de este prototipo de herramienta web ha sido Flash, en concreto la herramienta Flash Professional 8, junto con PHP y MySQL. Este tipo de tecnología nos ofrece la flexibilidad y adaptabilidad suficientes como para que el prototipo sea totalmente accesible vía web y además sea rico en contenido, algo imprescindible cuando las tareas que van a desarrollarse van dirigidas a niños y niñas de corta edad.

1.1. Descripción del problema y objetivos principales.

En la literatura científica no existen estudios epidemiológicos con población española, pero en estudios americanos se estima una incidencia del 4,7% para el TDAH de tipo Inatento; del 3,4 % para el de tipo Hiperactivo-Impulsivo y 4,4% para el de tipo Combinado. Se calcula que este trastorno constituye el 50% de la población psiquiátrica infantil; y es más frecuente en niños que en niñas, con un ratio que se ha llegado a considerar de 9 niños por 1 niña en la población clínica y de 4 a 1 en la población general (*Orjales, 2000*).

Aunque se han propuesto distintas teorías para explicar la sintomatología de este trastorno, hasta el día de hoy ninguna de ellas explica el espectro total de los síntomas. No obstante, el TDAH puede considerarse como un trastorno en el que los factores poligénicos desempeñan un papel importante, modulando su expresión, la combinación de varias de estas mutaciones e incluso la interacción con otros factores ambientales, perinatales y psicosociales. Estudios previos han señalado que su índice de heredabilidad oscila entre 0.55 y 0.90. Los avances en neuroimagen de la última década han aportado también numerosos hallazgos sobre la base fisiopatológica del TDAH, sugiriéndose una disfunción del circuito frontoestriatal que involucra a la corteza prefrontal y a su relación con los núcleos de la base, tálamo y cerebelo (Figura 1.1.).

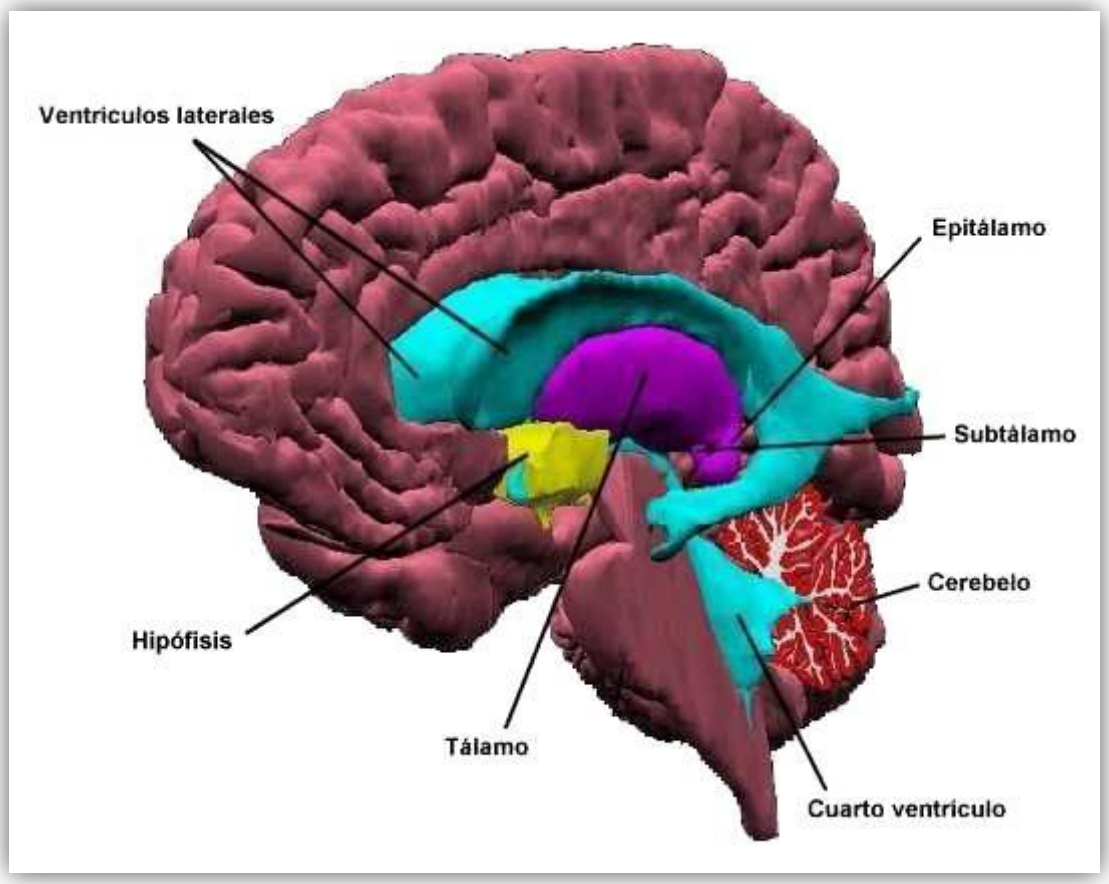


Figura 1.1 – Esquema cerebral.

Desde la aproximación de la Neurociencia Cognitiva, se ha visto que uno de los déficits cognitivos importantes que pueden presentar estos niños tiene que ver con el funcionamiento la Red de Control Ejecutivo o Atención Ejecutiva. Es importante señalar que desde esta aproximación, la Atención no se considera como una entidad unitaria sino como un sistema modular que ejerce funciones de control a través de la coordinación de diferentes redes. En este sentido, M. Posner ha propuesto una teoría integradora que defiende que el Sistema Atencional estaría compuesto, al menos, por tres redes atencionales disociables tanto anatómica como funcionalmente: la Red de Orientación, la Red de Vigilancia o Alerta y la Red de Control Ejecutivo o Atención Ejecutiva (*Posner, 1995*).

De acuerdo con esta teoría, la red de Vigilancia se encargaría de mantener un estado preparatorio o de "arousal" general, necesario para la detección rápida estímulos esperados. Aunque hay estudios que enfatizan la función tónica o duradera del estado de alerta en tareas de vigilancia, también se atribuye a esta red la función de alerta fásica o de corta duración producida por la presentación de señales de aviso que anuncian la inminente llegada de un estímulo. Las áreas corticales asociadas a esta función están lateralizadas al hemisferio derecho; concretamente en las áreas del lóbulo frontal y parietal que reciben proyecciones del Locus Coeruleus. En cuanto a la red de Orientación, ésta estaría implicada en la orientación de la atención hacia un lugar en el espacio donde aparece un estímulo potencialmente relevante bien porque posee propiedades únicas, porque es novedoso, o porque aparece de manera abrupta en la escena visual. Esta red es de suma importancia ya que nos ayuda en la búsqueda de estímulos potencialmente relevantes, favoreciendo así su procesamiento de alto nivel. Aunque existen redes similares relacionadas con otras modalidades sensoriales, la mayor parte de los datos existentes hacen referencia a la atención visual.Los resultados de estudios con técnicas de neuroimagen cerebral y de pacientes con daño cerebral muestra que las áreas cerebrales implicadas en esta función de orientación son el córtex parietal posterior, los núcleos pulvinar y reticular del tálamo y los colículos superiores. Por último, la red de Control Ejecutivo o Atención Ejecutiva sería la encargada de ejercer el control voluntario sobre el procesamiento ante situaciones que requieren algún tipo de planificación, desarrollo de estrategias, resolución de conflicto estimular o de respuesta o, en general, situaciones que impliquen la generación de una respuesta novedosa. Estudios con neuroimagen parecen converger en que las estructuras cerebrales implicadas en estas funciones de resolución de conflicto, así como en otras funciones de control, serían el cíngulo anterior y otras

áreas prefrontales relacionadas como, por ejemplo, el área dorsolateral prefrontal izquierda. Los estudios científicos basados en esta teoría, además de proporcionar un conocimiento sólido sobre las bases neurales de estas redes atencionales, también ha aportado información sobre su evolución durante el neurodesarrollo (*Rueda et al., 2004*). Así, por ejemplo, en estudios previos se ha visto que a partir de los 3 años los niños comienzan a desarrollar la Red de Atención Ejecutiva, la cual opera de manera coordinada con la memoria de trabajo en una amplia variedad de tareas cognitivas y permite regular los pensamientos y las emociones. En estudio previos, también se ha podido demostrar que un entrenamiento específico a través de tareas informatizadas podría mejorar el funcionamiento de esta red atencional (v.g. *Rueda et al., 2004; Daza et al., 2010*).

Y este es, precisamente, el principal objetivo en el que nos vamos a centrar en este proyecto: el desarrollo de una herramienta web para la gestión y control de sesiones de intervención cognitiva-conductual para niños con TDAH de entre 7 y 12 años, través de tareas informatizadas que entrenan distintos aspectos de la atención ejecutiva.

1.2. Organización de la memoria.

Esta memoria se va a organizar en diversos capítulos que vamos a explicar brevemente a continuación:

- **Capítulo 2 – Propuesta de solución:** En este capítulo se establecerán los límites del proyecto. Será aquí dónde se haga una descripción detallada tanto del problema como de la metodología aplicada para el diseño y el desarrollo del prototipo.

- **Capítulo 3 – Especificación de AT²-Training:** En este capítulo se detallará la fase de análisis del proyecto.

- **Capítulo 4 – Diseño e implementación:** Este capítulo se centrará en realizar un estudio de la fase de diseño e implementación del prototipo.

Será aquí donde se especifiquen también los diagramas de clases, secuencia e información relativa al diseño de la base de datos.

- **Capítulo 5 – Aspectos destacados de la implementación:** Este capítulo se centrará en ciertos detalles explicados específicamente ya sea porque no han sido tocados en profundidad en anteriores apartados o porque se considera que tienen cierta relevancia para ser comentados aparte.

- **Capítulo 6 – Evaluación de AT²-Training:** El objetivo de este capítulo será detallar la prueba, la implantación y la aceptación de los distintos usuarios con respecto a la solución alcanzada.

- **Capítulo 7 – Conclusiones y trabajos futuros:** Se establecerán las conclusiones a las que hemos llegado tras el desarrollo y evaluación del prototipo, además de especificar los trabajos futuros que se han ido observando durante el proceso de desarrollo.

- **Apéndices:** La memoria constará de dos apéndices; el A, que contendrá el manual de instalación para poder poner en marcha el prototipo desarrollado y el B, que contiene el manual de usuario, dónde se podrá consultar cualquier duda que tengamos a la hora de trabajar con el mismo.

1.3. Bibliografía básica.

Al final de esta memoria de proyecto fin de carrera tenemos disponible la totalidad de la bibliografía usada, pero aquí haremos referencia a la literatura básica e imprescindible que consideramos de especial referencia. Para realizar una pequeña descripción de la bibliografía usada la vamos a englobar en tres grupos:

1- **Bibliografía psicológica específica:** En este grupo se detallará toda la bibliografía que trata sobre el Trastorno por Déficit de Atención/Hiperactividad (TDAH), como por ejemplo:

- Orjales, I. (2000). Déficit de atención con hiperactividad: el modelo híbrido de las funciones ejecutivas de Barkley. *Revista Complutense de Educación, 11(1)*, 71-84.
- Posner, M.I. (1995). Attention in cognitive Neuroscience: An overview. En M.S. Gazzaniga (Ed.), *The Cognitive Neurosciences* (pp. 615–648). Cambridge Massachusetts: The Mit Press.
- Posner, M.I. et al. (2010). Training effortless attention. En: *Effortless Attention: A New Perspective in the Cognitive Science of Attention and Action*, B. Bruya (Ed.), Cambridge: MIT Press.
- Romine, C.B., Reynolds, C.R. (2004). Sequential memory: a developmental perspective on its relation to frontal lobe functioning. *Neuropsychology Review, 14*, 43-64.
- Rueda, M.R., Fan, J., McCandliss, B.D., Halparin, J.D., Gruber, D.B., Lercari, L.P., Posner, M.I. (2004). Development of attentional Networks in Childhood. *Neuropsychologia, 42(8)*, 1029-1040. 2004.

2- **Bibliografía genérica sobre la metodología:** Este grupo se centrará EN el tipo de bibliografía común a otros proyectos software, ya que es bibliografía necesaria para el desarrollo de solución software. A modo de ejemplo, tenemos:

- Roger S. Pressman. "*Ingeniería del Software. Un enfoque práctico*", McGraw-Hill, 2005.
- Sommerville, Ian. "*Ingeniería del software. 7ª edición*", Addison Wesley. 2005.

3- **Bibliografía específica sobre el desarrollo:** Este es el grupo de bibliografía que engloba los textos necesarios para conocer las distintas herramientas que se han utilizado en este proyecto para desarrollar el prototipo final. En concreto, son aquellas que hacen referencia a Flash o PHP, como por ejemplo:

- Powers, D. "PHP 5 for Flash". Friendsoft, 2005.
- Bhangal, S. y Besley, K. "Foundation Flash 8". Friendsoft, 2006.

- Vogeleer, D., Wilson, E. y Barber, L. "Macromedia Flash Professional 8 UNLEASHED". Sams, 2005.

Capítulo 2 – Propuesta de solución

En este capítulo se presentará la propuesta de solución al planteamiento inicial del problema estableciendo el dominio de la aplicación y el límite de la propuesta, además de las metodologías aplicadas, tecnologías utilizadas y la justificación de su elección frente a otras soluciones alternativas.

2.1. Dominio de la aplicación.

Como ya se ha descrito anteriormente, el Trastorno por Déficit de Atención/Hiperactividad (TDAH) es la patología neuroconductual más común en la infancia. Es un trastorno complejo en el que se hallan implicados diversos factores neuropsicológicos que provocan en el niño inatención, impulsividad y sobreactividad motora; sintomatología que va a tener importantes repercusiones tanto en el desarrollo cognitivo, emocional y social del niño, como en su capacidad de aprendizaje (*Cardo et al., 2010*).

El abordaje terapéutico del TDAH es un tema complejo y controvertido que requiere de una intervención holística, y en la que en la mayoría de las ocasiones se incluye tratamiento farmacológico y/o psicológico. Desde el ámbito de la neuropsicología infantil, la intervención en niños con TDAH enfatiza de manera importante el abordaje de los déficits cognitivos a través de programas de intervención (no farmacológicos) con entrenamiento específico de aquellos componentes o mecanismos cognitivos que se muestran más deficitarios. En este sentido, y teniendo en cuenta la literatura científica previa (sobre todo aquella que proviene del campo de la Neurociencia Cognitiva), un entrenamiento específico de los procesos de atención podría ayudar a paliar una parte importante de los déficit que presentan estos niños, ya que, además de poder contribuir a potenciar los efectos del resto de tratamientos psicológicos, en algunos casos podría llegar a ser una alternativa al tratamiento farmacológico.

El Entrenamiento Atencional (también conocido como *Attention Process Training* -APT-), tiene su origen en el campo de la Rehabilitación Cognitiva (Sohlberg & Mateer, 1989), y parte de la hipótesis de que es posible incrementar la eficacia de los procesos atencionales mediante la práctica repetitiva y sistemática de tareas conductuales que requieren la puesta en marcha de dichos procesos, pues dicha práctica puede producir adaptaciones en las redes neuronales subyacentes asociadas a esos procesos.

A pesar de que ya existe en la literatura científica algunos resultados previos que avalan la eficacia del Entrenamiento Atencional en distintas poblaciones patológicas (incluidos pacientes con daño cerebral traumático o accidentes cerebro-vasculares), una de las principales críticas que ha recibido este enfoque, es que en la mayoría de las ocasiones el entrenamiento atencional no está basado en un modelo teórico o marco conceptual de la Atención. Es importante señalar que las investigaciones comportamentales y psicofisiológicas realizadas en las últimas décadas, tanto en sujetos normales como en poblaciones clínicas, ponen de manifiesto que la atención no es una entidad unitaria sino un sistema modular que ejerce funciones de control a través de la coordinación de diferentes redes. Por lo tanto, tanto la evaluación de los procesos atencionales en las distintas poblaciones patológicas, como el desarrollo de los programas específicos de entrenamiento atencional, deberían estar basados en un modelo atencional basado en los resultados de estos estudios previos. En este sentido, el modelo atencional propuesto por M. I. Posner (ver capítulo 1),

proporciona un marco de referencia fundamental para el diseño de programas específicos de Entrenamiento Atencional. Especialmente para aquellos que van dirigidos a niños, pues este modelo, además de proporcionar un conocimiento sólido sobre las bases neurales de las redes atencionales de Alerta, Orientación y Atención Ejecutiva, aporta también información sobre su evolución durante el neurodesarrollo.

2.2. Motivaciones y límites de la propuesta.

La propuesta consiste en el diseño e implementación de un prototipo de herramienta software mediante la cual expertos terapeutas puedan planificar sesiones de intervención cognitivo-conductual a través de pruebas (en adelante, tareas) informatizadas específicamente diseñadas para el entrenamiento de la atención ejecutiva. El principal objetivo de la propuesta es que los terapeutas que vayan a trabajar con el prototipo sean capaces de planificar y gestionar sesiones de entrenamiento atencional para los distintos sujetos con los que trabajen. Dichas sesiones (de 45 minutos de duración), deberán estar compuestas por tres tareas diferentes en las cuales el niño deberá poner en funcionamiento distintos aspectos de atención ejecutiva.

Por cuestiones de familiaridad con el entorno y accesibilidad desde cualquier parte del mundo, se optará por el desarrollo de una aplicación web dinámica que haga uso de tecnología *Flash* y que tiene como núcleo una base de datos implementada en MYSQL, ambas interconectadas mediante el uso del lenguaje PHP. Todas las sesiones que se realicen y los resultados obtenidos serán registrados con el objetivo último de analizar y evaluar la evolución de los sujetos en función de sus datos comportamentales, pudiendo ser consultadas y evaluadas a posteriori por los terapeutas para diseñar planes más específicos en base a esos resultados.

Las tareas específicamente diseñadas para el entrenamiento de la atención ejecutiva serán diez, pudiéndose combinar en conjuntos de tres para cada una de las sesiones que se configuren para los planes de entrenamiento. Estas tareas formarán parte de la aplicación y estarán desarrolladas en *Flash*, tecnología que nos proporcionará la suficiente flexibilidad, adaptabilidad y atractivo visual para que las tareas cumplan el objetivo de entrenar los distintos aspectos de atención ejecutiva y

proporcionarnos la información necesaria para poder evaluar la efectividad del entrenamiento. Además, no se debe olvidar que estas tareas están enfocadas a sujetos de corta edad, por lo que el atractivo visual es una parte fundamental en el desarrollo de la herramienta.

2.3. Metodologías aplicadas.

En esta sección se va a realizar una descripción de las metodologías utilizadas para el desarrollo de la solución.

2.3.1. RUP.

El Proceso Racional Unificado (del inglés, *Rational Unified Process* ,RUP) es un proceso de desarrollo de software que, junto con el Lenguaje Unificado de Modelado (UML), constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. El objetivo principal de RUP es asegurar la producción de software de alta calidad que se ajuste a las necesidades de sus usuarios finales con unos costes y fecha de entrega predecibles.

RUP está basado en tres características principales que detallamos a continuación:

- **Dirigido por casos de uso:** Los usuarios (humanos u otros sistemas), interactúan con el sistema, y éste produce una secuencia de acciones. La interacción es un caso de uso. Se trata de un fragmento de funcionalidad del sistema, que proporciona al usuario un resultado importante. Los casos de uso representan requisitos funcionales. Todos los casos de uso juntos constituyen el Modelo de Casos de Uso, el cual describe la funcionalidad total del sistema. La estrategia para construir caso de uso es responder a la pregunta ¿Qué debe hacer el sistema para cada usuario?. Los casos de uso también guían el diseño, la

implementación y prueba del sistema. Se desarrollan en paralelo con la arquitectura y cada uno influye en el otro.

- **Centrado en arquitectura:** La arquitectura se describe mediante diferentes vistas del sistema en construcción. Incluye los aspectos estáticos y dinámicos más significativos del sistema: hardware, sistema operativo, SGBD, protocolos de comunicación, consideraciones de implantación, sistemas heredados y requisitos no funcionales como rendimiento, fiabilidad y seguridad. Es una vista del diseño completo, destacando las características más importantes y dejando a un lado los detalles. ¿Cómo se relacionan los casos de uso y la Arquitectura? Cada producto tiene una función y una forma. Cada caso de uso que representa funciones clave en el sistema se especifica en detalle y se realiza en términos de subsistemas, clases y componentes.

- **Iterativo e incremental:** Dividir en ciclos un proyecto hace que este sea más manejable. Cada ciclo consiste en una iteración que resulta en un incremento hacia la solución final. Las iteraciones son pasos en el flujo de trabajo, mientras que los incrementos hacen referencia al crecimiento del producto. Las iteraciones deben seleccionarse y ejecutarse de forma planificada. Hay dos factores para seleccionar lo que se implementará en una iteración: un grupo de Casos de Uso que amplían la utilidad del producto desarrollado hasta el momento; los riesgos que pueden presentarse. Cada iteración consta de: análisis, diseño, implementación y prueba. En cada iteración los desarrolladores identifican y especifican los Casos de Uso relevantes, crean un diseño usando la arquitectura seleccionada como guía, implementan el diseño mediante componentes, y verifican que los componentes satisfacen los Casos de Uso originales.

- **Lenguaje de modelado único:** De esta manera tenemos que UML es el único lenguaje de modelado que usamos para el desarrollo de todos los modelos.

En RUP se definen cinco actividades fundamentales que son desarrolladas a lo largo del proceso de creación del proyecto. Estas actividades son los requisitos,

análisis, diseño, implementación y prueba. Estas actividades se pueden observar de forma clara en el siguiente gráfico (Figura 2.1.).

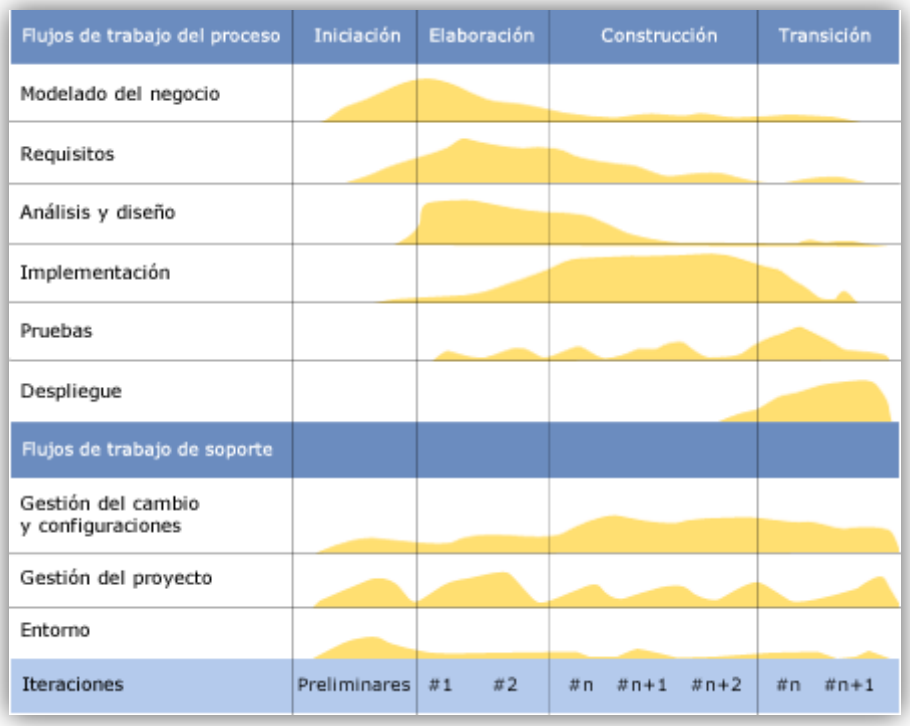


Figura 2.1 – Tareas de RUP.

Como se puede observar, son nueve las actividades, pero como fundamentales se destacan las cinco que se van a detallar a continuación:

- **Requisitos:** En esta etapa es donde se realiza un modelo del sistema que se va a desarrollar y se establecen las bases del ámbito del sistema, objetivos y funcionalidades. Toda la información reunida y tratada en esta fase es utilizada como base de información para el resto de actividades.

- **Análisis:** En esta actividad se analizan los requisitos reunidos en la fase anterior, estructurándolos en un modelo de casos de uso para obtener de esta manera una visión del sistema centrada en los requisitos funcionales.

- **Diseño:** Etapa que se utiliza para refinar el modelo de análisis obtenido en la fase anterior. Se incluyen los requisitos no funcionales y restricciones adicionales. Se puede usar un lenguaje más formal para apuntar detalles relativos a los requisitos del sistema y dividir el modelo en subsistemas, facilitando así la implementación futura.

- **Implementación:** Es en esta parte dónde se realizan todos los componentes en los que se divide el sistema a crear: objetos, ficheros, scripts, imágenes, etc. En esta actividad se van obteniendo partes funcionales del sistema final que una vez ensambladas, al finalizarlas todas, se convierten en el producto final a falta de la prueba.

- **Prueba:** Esta actividad comprende el conjunto de pruebas que se realizan a cada uno de los componentes desarrollados y al producto final para dotarlo de la calidad deseada. Como se puede observar en el gráfico, aunque es una actividad final, se realizan pruebas desde el mismo momento en que hay componentes independientes que pueden ser testados.

Como se puede observar en el diagrama anterior, también existe una serie de etapas temporales durante las cuales las actividades forman parte del proyecto en mayor o menor medida. Estas etapas son la de iniciación, elaboración, construcción y transición. En líneas generales, estas son las actividades principales que se desarrollan en cada etapa:

- **Iniciación:** En esta primera etapa se realiza todo el estudio inicial, presupuesto, costes, calendario y plan de trabajo. También se comienza a estudiar y establecer los requisitos generales del sistema a construir.

- **Elaboración:** En esta etapa se realiza principalmente el análisis y el diseño y se establece el grueso de los requisitos del sistema. Además, se van implementando los primeros y más básicos componentes con sus correspondientes pruebas. Generalmente consta de dos iteraciones, la principal y la de revisión.

- **Construcción:** En esta etapa se desarrolla el producto en sí, por lo que la implementación y las pruebas de cada componente, en menor medida, son las actividades principales. Esta etapa consta de tantas iteraciones como sean necesarias para alcanzar con calidad el producto final. En las primeras iteraciones se realizan revisiones de los requisitos que pueden desencadenar en nuevas necesidades por lo que el análisis y el diseño de esas nuevas necesidades debe ser realizado y posteriormente implementado. Esto va desapareciendo conforme se realizan nuevas iteraciones y se va alcanzando un producto final más completo.

- **Transición:** Una vez está el producto finalizado según los requisitos iniciales y los que se han ido incorporando, entramos en la última etapa. En esta etapa realizamos el despliegue o entrega del proyecto. Esta etapa consta de tantas iteraciones como sean necesarias para que el cliente quede satisfecho con el producto final, por lo que la implementación en la primera iteración suele ser aún importante. Si se observaran deficiencias o añadiesen requisitos de última hora, se deberían realizar las actividades correspondientes para la incorporación de esas mejoras. Esto podría conllevar una prolongación del tiempo de entrega y nuevas pruebas.

2.3.2. UML.

Una de las principales características que detallan al proceso de desarrollo de software que se va a utilizar, RUP, es que el lenguaje de modelado que utiliza es único, en este caso: UML.

UML es un lenguaje estándar que se utiliza para el modelado de sistemas software, tanto en la fase de análisis como de diseño de la solución.

2.4. Consideraciones acerca de las tecnologías utilizadas.

La tecnología utilizada en la realización de este prototipo debe ser aquella que pueda ofrecer un buen rendimiento gráfico para el desarrollo de las tareas y la

flexibilidad suficiente para conectar con la base de datos dónde se almacene toda la información que va a gestionar. Se debe remarcar la importancia de un manejo fluido en cuanto a gráficos puesto que el prototipo va a ser usado por sujetos de corta edad y, por lo tanto, el aspecto y respuesta del mismo debe ser acorde con éstos.

Con estas premisas se ha optado por la elección de *Flash* como tecnología de desarrollo del prototipo. Al conectar con la web, el cliente descarga el archivo de Flash (archivo de extensión swf) completo a su equipo, desde donde se ejecuta el programa íntegro, efectuando conexiones puntuales a la base de datos según la demanda o solicitando las imágenes que necesite. De esta manera nos aseguramos que la ejecución va a ser fluida siempre al cargar toda la ejecución del lado del cliente. Para poder realizar la conexión entre *Flash* y una base de datos existen múltiples posibilidades a disposición tales como *Flash Remoting,* PHP o ASP, entre otras.

En los siguientes apartados se describirán la arquitectura adoptada finalmente y el por qué de su elección frente a otras tecnologías.

2.4.1. Arquitectura del sistema.

Para este tipo de desarrollo se ha adoptado como arquitectura del sistema WAMP. WAMP es el acrónimo usado para describir un sistema de infraestructura de Internet que usa las siguientes herramientas:

- **W**indows como sistema operativo;
- **A**pache como servidor Web;
- **M**ySQL como gestor de bases de datos;
- **P**HP (en este caso), Perl o Python como lenguajes de programación.

En concreto se han utilizado las versiones 5.3.5 de PHP y la 2.2 de Apache.

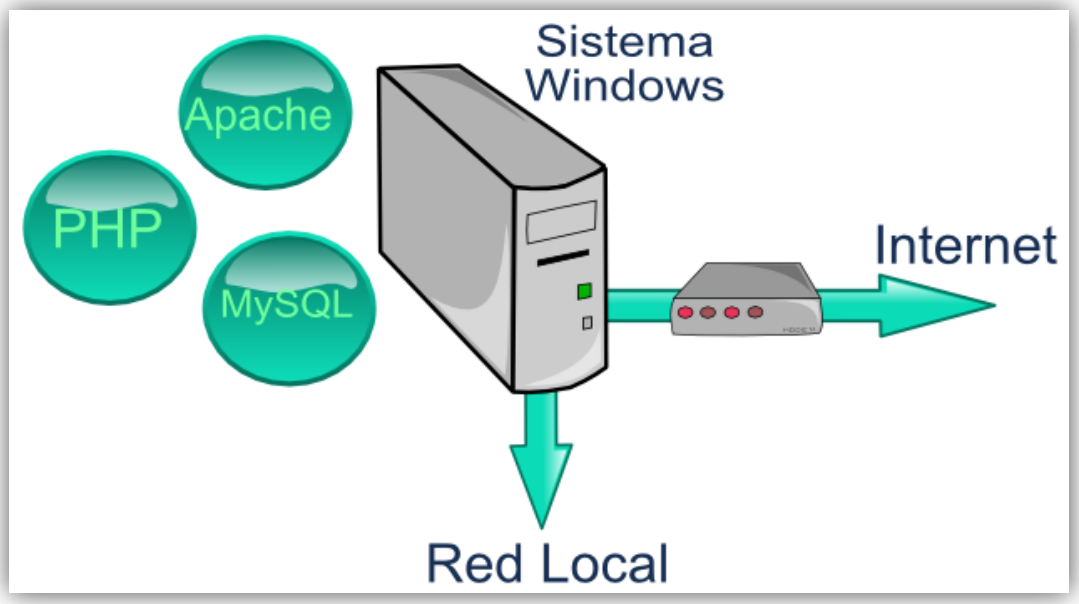


Figura 2.2 – Esquema de la arquitectura WAMP.

Como se puede observar en la Figura 2.2, en un mismo servidor se pueden concentrar el servidor Web y el de MySQL. Una arquitectura WAMP de este tipo será la implementada durante todo el desarrollo del prototipo. No obstante en una fase de implantación a un nivel mayor la base de datos y el servidor Web podrían estar en servidores independientes, repartiendo así la carga de trabajo.

La elección de esta arquitectura compuesta por Apache 2.2, MySQL y PHP 5.3 frente a otras ha sido en base a las siguientes ventajas:

- **Código abierto:** Están basadas en código abierto, es decir, el código que compone estas tecnologías es abierto y por tanto son receptoras de mejoras constantes por parte de la gran comunidad de profesionales que trabajan con ellas. También es más sencillo encontrar respuesta a las múltiples dudas que se plantean en las tareas de programación y configuración.

- **Multiplataforma:** Tanto la programación, configuración como la ejecución es posible en cualquier plataforma, ya sea Windows, Linux o Mac OS. En este caso, el desarrollo ha sido en Windows y una vez montado el servidor en este mismo sistema operativo, hace que la web

sea accesible desde cualquier navegador que tenga instalado el *plugin* de *Flash* en cualquier plataforma.

- **Seguridad:** La configuración de seguridad en los servidores Apache 2.2 hace que sean sistemas muy resistentes a ataques de virus o intrusiones no autorizadas.

- **Uso global:** Estas tres tecnologías están muy extendidas a lo largo y ancho de internet, desde pequeñas webs de aficionados hasta las webs corporativas de grandes empresas y organizaciones mundiales.

- **Coste:** El coste del uso de la combinación de estas tecnologías es cero. Que las herramientas que van a ser usadas para este desarrollo sean gratuitas es una característica imprescindible puesto que no se dispone de presupuesto destinado a tal efecto.

De esta manera es como va a estar organizado el servidor en base a las tecnologías que harán posible la publicación de la web y el almacenamiento y consulta de datos en la base de datos. Sin embargo, hay que notar la forma en la que *Flash* hará uso de PHP para el tráfico de información con la base de datos.

Cuando un usuario conecta con la página web que contiene el archivo *Flash* (swf), éste es descargado a la máquina del cliente de forma automática. Los archivos swf son auto-contenidos, es decir, que este archivo contiene todo lo necesario para su ejecución. Al ejecutarse dicho archivo, toda la carga de trabajo recae sobre el equipo cliente excepto conexiones puntuales a la base de datos o peticiones de determinados archivos. Es para las peticiones con la base de datos en las que PHP y el servidor entran en juego. Al realizar una petición a la base de datos, *Flash* realiza una llamada a un script PHP que reside en el servidor; este script es ejecutado siempre en el lado del servidor y es el que se comunica con la base de datos y, a su vez, reenvía la respuesta de nuevo al cliente. Una vez que el cliente recibe la respuesta, *Flash* la tratará según sea el caso para mostrarla convenientemente.

2.4.2. Alternativas para la implementación.

En este apartado se exponen las diferentes alternativas que se han presentado a la hora de escoger las tecnologías a usar en el desarrollo de este prototipo.

Para este tipo de desarrollo del prototipo se tuvieron dudas sobre la tecnología para desarrollar la aplicación entre HTML5 y *Flash*. HTML5 es de uso reciente y presenta mejoras sustanciales frente a HTML4. Esta nueva versión incluye nuevo soporte a imágenes 2D y 3D, además de nuevos controles para soportes multimedia, funcionalidades para arrastrar objetos, así como nuevos tipos de datos y etiquetas para manejar grandes conjuntos de datos. Es muy importante el manejo de gráficos e imágenes puesto que la aplicación está basada en gran medida en el uso de estos recursos destinados a sujetos de corta edad. Aún así, HTML5 pese a tener grandes mejoras no añade nada que *Flash* no haga desde hace tiempo y con una mejor nota en rendimiento en cuanto al manejo de gráficos e imágenes. Además, el que los archivos *Flash* sean auto-contenidos y se ejecuten en su totalidad del lado del cliente nos va a facilitar la ejecución con una mayor fluidez. Por estas razones, la herramienta Macromedia Flash Professional 8 ha sido la elegida para el desarrollo. Además, contábamos con una licencia educativa del Departamento de Lenguajes y Computación de la Universidad de Almería.

Una vez elegida la herramienta de desarrollo para el prototipo aún queda por elegir la tecnología que unirá *Flash* con la base de datos. En este caso existen varias alternativas disponibles y desde el punto de vista práctico, todas ellas válidas, por lo que serán enumeradas y comentadas una a una:

- **Flash Remoting:** Pese a ser la mejor alternativa desde el punto de vista tecnológico, es realmente rápido y está muy integrada con *Flash*, el gran coste que conlleva, aproximadamente $1000 por servidor, nos hace descartarla.

- **ASP:** Existen gran cantidad de recursos disponibles y es una alternativa veloz, pero requiere servidores Web de Microsoft y tampoco es una alternativa gratuita.

- **XML:** Los servicios web poseen una expresión de información bien estructurada pero son lentos. Además, no parece que este tipo de tecnología entre cliente-servidor vaya a tener continuidad en el futuro.

- **JSP:** Hereda todas las ventajas de Java, tiene soporte para una programación dinámica y un buen rendimiento. Es ideal para aplicaciones Web con una gran complejidad en el lado del servidor. En el caso de este prototipo la complejidad reside en el lado del cliente por lo que no necesitamos este tipo de tecnología para tan poca transferencia de datos del lado del servidor.

- **PHP:** Tiene integración con *Flash*, un rendimiento muy bueno, es una tecnología muy extendida, multiplataforma, código abierto y además es totalmente gratuita; ventajas por las cuales se ha decidido implementar este tipo de solución. Todas estas ventajas, además de de su curva de aprendizaje, ha sido las que han motivado la decisión de implementar este tipo de decisión.

Capítulo 3 – Especificación de AT²-Training

Este capítulo se centrará en la especificación del prototipo desarrollado; el plan de trabajo, los casos de uso y el diagrama de clases informal.

3.1. Plan de trabajo.

La forma de realizar la planificación del desarrollo del prototipo viene determinada por las técnicas de desarrollo de software que se determinaron con anterioridad. Una correcta planificación es muy importante para obtener un producto software de calidad en tiempos y costes determinados.

Debido a la falta de experiencia con desarrollos completos de software y que el equipo de trabajo está compuesto por una única persona (tutorada), elaborar y seguir la planificación ha sido complicado. Además, el tipo de aplicación, con partes muy diferenciadas y muy poco relacionadas desde el punto de vista de la implementación como la gestión de usuarios, la gestión de tareas (con gran contenido gráfico) y la gestión de la base de datos, remarcan la gran dificultad de dicha planificación. Partes

de esta planificación han debido ser modificadas a lo largo del desarrollo al ir ahondando en determinadas fases.

En la siguiente tabla se puede observar la planificación temporal que se ha seguido de forma más clara:

Nombre de la fase	Tiempo
Inicio del proyecto	**1 semana**
Proceso documental	**2 meses**
Recopilar bibliografía	2 semanas
Estudio de las herramientas: Flash, PHP y MySQL	4 semanas
Estudio de los fundamentos de las tareas	2 semanas
Análisis del sistema	**2 meses**
Especificar los objetivos y el alcance del sistema	2 semanas
Establecer requisitos del sistema	3 semanas
Definición de alternativas	1 semana
Análisis del sistema y casos de uso	2 semanas
Diseño del sistema	**2 meses**
Diseño de la base de datos	1 semana
Elaborar una descomposición funcional detallada del sistema	2 semanas
Diseñar el conjunto de interfaces del sistema	3 semanas
Diseñar las tareas	2 semanas
Implementación del sistema	**4 meses**
Implementación de la base de datos	2 semanas
Implementar la interfaz de usuario	3 semanas
Implementar los scripts de conexión con la base de datos	2 semanas
Implementar los formularios	2 semanas
Implementar las tareas	7 semanas
Prueba y evaluación	**1 mes**
Escribir la memoria del proyecto	**2 meses**

Tabla 3.1 – Planificación temporal del proyecto.

3.2. Funcionalidad del sistema y casos de uso.

Los casos de uso son utilizados para mostrar de manera gráfica y simplificada las tareas más relevantes que realiza el sistema a analizar. Un caso de uso es una secuencia de interacciones que suceden entre un sistema y sus actores en respuesta a un evento que inicia un actor principal sobre el propio sistema. Estos diagramas sirven para especificar la comunicación y el comportamiento de un sistema mediante su interacción con los usuarios y/u otros sistemas, es decir, que muestra la relación entre los actores y los casos de uso en un sistema. La notación utilizada en estos gráficos forma parte de la especificación de UML para el modelado de los requisitos funcionales del sistema en la fase de análisis.

En el primer diagrama (Figura 3.1), que representa el nivel inicial, se pueden observar a los tres actores principales y casos de uso del sistema:

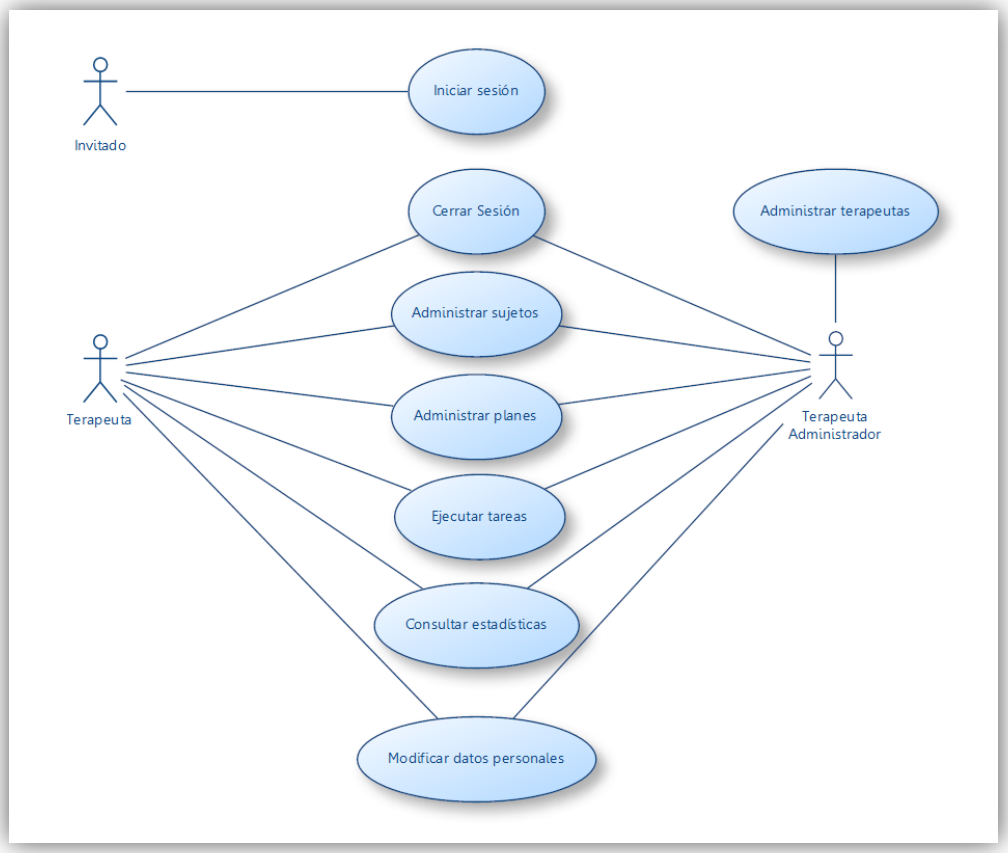


Figura 3.1 – Diagrama de casos de uso.

Se detallan a continuación los tres actores principales del sistema:

- **Invitado:** Es el actor que representa a toda persona que acceda a la web principal de la aplicación. Ante ella sólo estará disponible el formulario de inicio de sesión, por lo que deberá introducir un usuario y contraseña correctos para acceder a la aplicación. De no ser así, no podrá ir más allá de dicho formulario.

- **Terapeuta:** Es el actor que representa a toda persona que inicie sesión de forma correcta en la aplicación. Para ello deberá contar previamente con un nombre de usuario y una contraseña. Es la encargada de hacer el seguimiento de los sujetos mediante su administración, la generación de nuevos planes y ejecución de tareas de prueba. Incluso durante la ejecución de las sesiones el terapeuta estará activo, aunque en este caso ceda el puesto al sujeto que deba realizar la sesión. En detalle, estas son las operaciones que puede llevar a cabo un terapeuta:

    - **Cerrar sesión:** El terapeuta en cualquier momento puede dar por terminada la sesión iniciada en la aplicación y salirse de la misma por medio de esta operación.

    - **Administrar sujetos:** Una vez iniciada la sesión, el terapeuta puede acceder a gestionar la información de todos los sujetos que hay en la base de datos, pudiendo realizar distintas operaciones como la modificación, agregación nuevos sujetos, eliminación de los mismos, establecimiento del seguimiento de un plan y entrar en la ejecución de una sesión. En la Figura 3.2 se verán con detalle los distintos casos de uso incluidos en esta función.

    - **Administrar planes:** Al igual que con la administración de sujetos, el terapeuta tendrá la posibilidad de administrar los planes. Un plan de entrenamiento atencional constará de tantas sesiones como el terapeuta crea conveniente con un mínimo de una sesión. De esta manera, el terapeuta será capaz de realizar nuevos planes, consultar los existentes,

modificarlos, eliminarlos y validarlos. Del mismo modo que puede hacerlo con los planes también podrá realizarlo con cada una de las sesiones de las que consta el plan, pudiendo agregar nuevas, eliminar existentes y modificar información. Una sesión está compuesta por la sucesión de tres tareas, las cuales deberán ser elegidas a la hora de crear o modificar una sesión existente, no pudiendo repetirse ni ser ninguna de ellas nula. Será en la Figura 3.3 donde podrán observarse los detalles de esta función.

- **Ejecutar tareas:** Las tareas tienen el objetivo de ser parte de sesiones de entrenamiento atencional, pero podrán ser probadas de forma individual por los terapeutas que accedan correctamente al sistema. Para ello solamente se deberá elegir la duración, el nivel inicial y la tarea que se desea ejecutar.

- **Consultar estadísticas:** Las estadísticas es otra de las operaciones que puede realizar un terapeuta al introducirse correctamente en el sistema. Esta operación es una operación de consulta que recogerá la información de la base de datos para ser mostrada al terapeuta.

- **Modificar datos personales:** El terapeuta, una vez dentro del sistema, podrá modificar sus datos personales en su totalidad salvo el nombre de login, que es único e inmodificable por mantener la integridad del sistema y la condición de ser administrador si no lo es. En estos datos modificables por el terapeuta incluimos la contraseña de acceso.

- **Terapeuta administrador:** Como podemos observar en la Figura 3.1, este actor es capaz de realizar las mismas operaciones descritas para un *Terapeuta,* además de poder administrar al resto de terapeutas, esto es, agregar nuevos usuarios a la aplicación, eliminarlos y modificar su información, incluyendo la modificación

de la condición de administrador. A continuación se detalla la operación única de este actor de Administrar terapeutas:

- **Administrar terapeutas:** Cuando un usuario es administrador puede realizar cambios en el resto de usuarios del sistema. De esta manera, podrá dar de alta a nuevos usuarios, eliminar aquellos que no tengan información relevante y modificar la información personal de éstos, salvo el nombre de usuario. Incluso podrá establecer nuevos usuarios administradores y restablecer una contraseña olvidada de otro usuario a una por defecto para que sea el propio usuario quien la cambie. En la Figura 3.4 se podrán observar los detalles de esta función.

Una vez visto el nivel inicial se pasará a detallar el siguiente nivel de las funciones más importantes de ese nivel: *Administrar sujetos*, *Administrar planes* y *Administrar terapeutas*.

En primer lugar se detallará *Administrar sujetos,* cuyo diagrama se puede observar en la Figura 3.2.

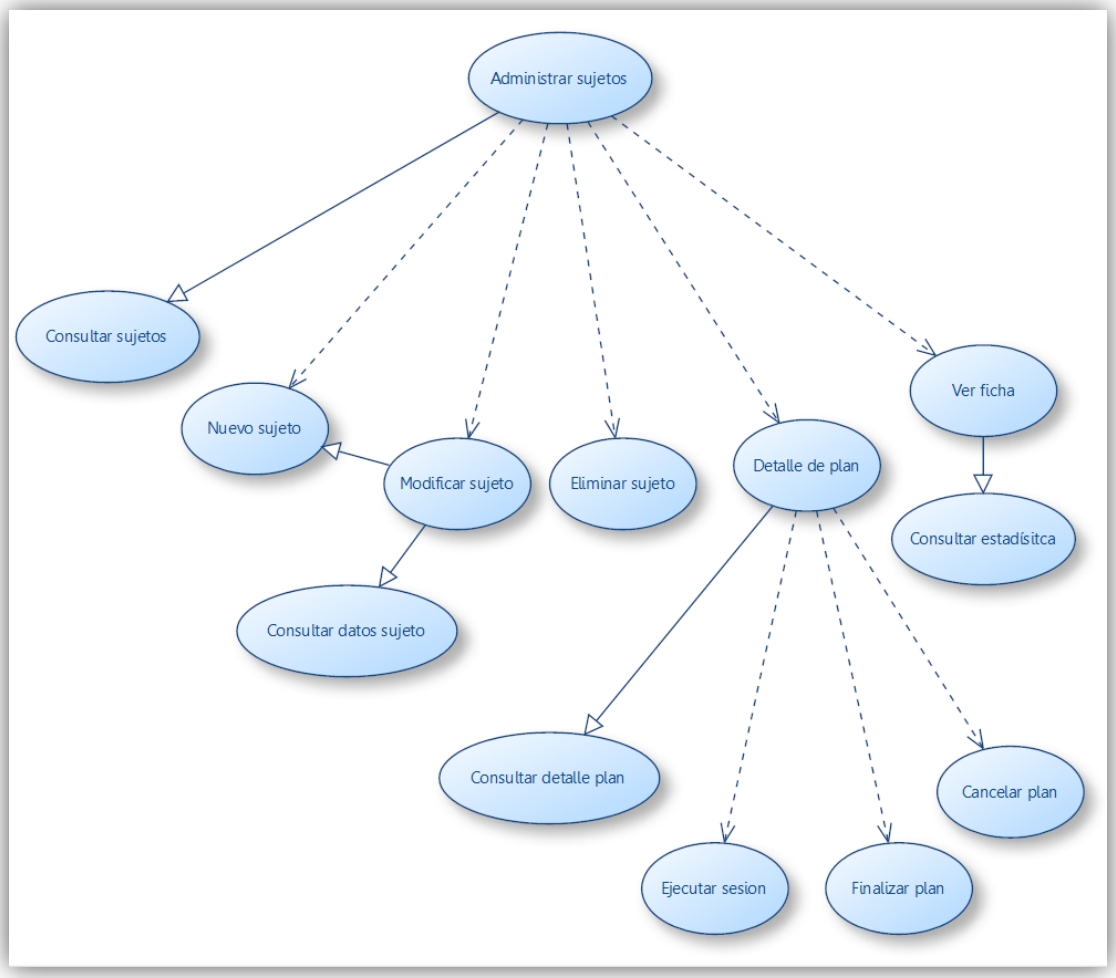


Figura 3.2 – Diagrama de casos de uso del segundo nivel de la función Administrar sujetos.

A continuación se detallarán las funciones que están presentes en la Figura 3.2, correspondiente al diagrama de casos de uso de la función *Administrar sujetos*:

- **Consultar sujetos:** Esta clase abstracta es la que proporciona el acceso a las funciones de administración de los sujetos. *Administrar sujetos* hereda de aquí la funcionalidad de mostrar un listado de sujetos que existen en la base de datos.

- **Nuevo sujeto:** Mediante esta función el terapeuta es capaz de realizar una nueva alta de un sujeto en la base de datos. Un alta hace posible la introducción de determinados datos personales del sujeto tales como el nombre, apellidos, centro, fecha de nacimiento y observaciones.

- **Modificar sujeto:** Mediante esta función, el terapeuta será capaz de realizar cambios en la información del sujeto. Esta función es una especialización de *Nuevo sujeto*, ya que realiza una operación de registro en la base de datos pero para una actualización de sus datos y no una nueva inserción. Para acceder a esta función, el terapeuta deberá entrar en la parte de administración y tras ver el listado de sujetos actuales acceder al sujeto que desea modificar, incluyendo la posibilidad de asignarle un plan de entrenamiento atencional.

    - ❖ **Consultar datos sujeto:** Esta función representa la capacidad de la función *Modificar sujeto* de mostrar la información actual y previa a la modificación del sujeto.

- **Eliminar sujeto:** También es posible la eliminación de un registro de sujetos en la base de datos. Mediante esta función, un terapeuta es capaz de eliminar el registro en su totalidad junto con los datos personales facilitados. No siempre será posible eliminar un sujeto, ya que si este ha participado en algún plan de entrenamiento atencional para mantener la integridad de los datos en la base de datos, la aplicación no permitirá eliminar ese sujeto.

- **Detalle de plan:** Si un sujeto tiene asignado un plan de entrenamiento atencional, será posible consultar los datos del plan mediante esta función. Un plan es asignado mediante la función de modificación de sujetos. Mediante esta función el terapeuta obtiene un listado de las sesiones que componen el plan actual al que está adscrito el sujeto, junto con el detalle sobre cada una de las sesiones. De esta manera se mostrará al terapeuta el estado en el que se encuentra el plan actual del sujeto, sesiones realizadas, sesiones repetidas, puntuaciones, niveles alcanzados, etc. dando la posibilidad de iniciar una nueva sesión, repetir una anterior que no haya finalizado correctamente o incluso finalizar el plan si todas las sesiones han sido superadas.

    - ❖ **Consultar detalle plan:** Esta clase abstracta es la que proporciona el acceso a las funciones de detalle del plan. *Detalle de plan* hereda de aquí la funcionalidad de mostrar el

listado de sesiones que componen el plan y los detalles de cada una de ellas.

- **Ejecutar sesión:** Una vez obtenido el listado de sesiones que componen el plan de entrenamiento, es posible elegir una de ellas y ejecutarla. Antes de iniciar la ejecución, el terapeuta deberá elegir la duración de la misma. Una vez comience la ejecución de la sesión, el terapeuta deberá ceder el puesto al sujeto para que realice la sesión, que constará de la sucesión de las tres tareas elegidas en el tiempo seleccionado.

- **Finalizar plan:** Cuando se consultan los datos sobre las sesiones, si el sistema detecta que todas las sesiones del plan han sido superadas, se dará la posibilidad de finalizar el plan de forma normal. De esta manera el sujeto pasará a no tener un plan asignado y será posible asignarle uno nuevo. La información recabada con anterioridad será posible consultarla en la ficha del sujeto. Si no se dieran las premisas anteriores no será posible finalizar el plan con normalidad.

- **Cancelar plan:** Si se desea que un sujeto esté disponible para asignarle un nuevo plan sin finalizar correctamente un plan anterior, esto es, superando todas y cada una de las sesiones de las cuales está compuesto, será posible hacerlo mediante la opción de cancelación. De esta manera, un plan queda extinto de forma alternativa a la vista anteriormente.

- **Ver ficha:** Mediante esta función se pasará a mostrar una serie de estadísticas sobre el sujeto que se desee. De esta manera, el terapeuta podrá consultar todos los datos relacionados con el sujeto que existan en la base de datos y no solo los que esté desarrollando en la actualidad.

  - **Consultar estadística:** Esta función representa la capacidad de la función *Ver ficha,* que consiste en consultar información estadística especial sobre el sujeto que se está consultando.

Una vez visto el diagrama de casos de uso correspondiente a la función *Administrar sujetos,* se verá el que corresponde a la función *Administrar planes,* en el que, tal y como se puede observar ver en la Figura 3.3, existen ciertas similitudes.

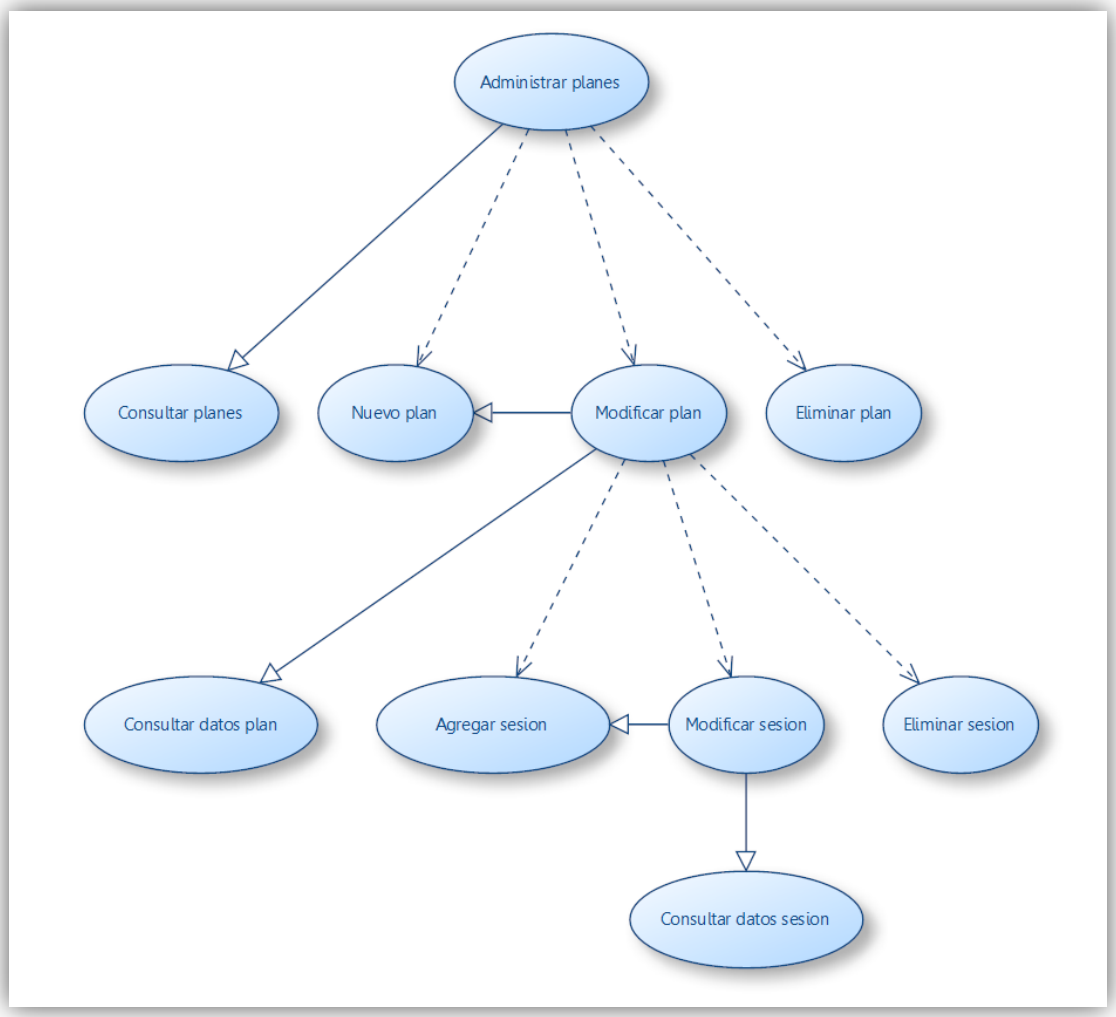


Figura 3.3 – Diagrama de casos de uso del segundo nivel de la función Administrar planes.

A continuación se detallan las funciones que están presentes en la Figura 3.3 correspondiente al diagrama de casos de uso de la función *Administrar planes*:

- **Consultar planes:** Esta clase abstracta es la que proporciona el acceso a las funciones de administración de los planes. *Administrar planes* hereda

de aquí la funcionalidad de mostrar un listado de planes que existen en la base de datos.

- **Nuevo plan:** Mediante esta función, el terapeuta es capaz de realizar un nuevo alta de un plan en la base de datos. Un alta hace posible la inserción de determinados datos como el nombre, una descripción y las observaciones que el usuario crea oportunas. El campo de validación no es posible modificarlo porque, para ello, el plan deberá constar de al menos una sesión. De esta manera, hará falta modificar dicho campo una vez cumplidos los requisitos.

- **Modificar plan:** Mediante esta función, el terapeuta será capaz de realizar cambios en la información del plan. Esta función es una especialización de *Nuevo plan*, ya que realiza una operación de registro en la base de datos pero para una actualización de sus datos y no una inserción nueva completa. Para acceder a esta función, el terapeuta deberá entrar en la parte de administración y, tras ver el listado de planes actuales, acceder al plan que desea modificar, incluyendo la posibilidad de validarlo y para que pueda ser asignado a sujetos. Si un plan ha sido validado y asignado a algún sujeto, éste no será modificable, salvo el campo de observaciones y la condición de validación, que podrá pasarse a no válido y dejar de estar disponible para futuras asignaciones a sujetos.

    - **Consultar datos plan:** Esta función representa la capacidad de la función *Modificar plan* de mostrar la información actual y previa a la modificación del plan.

    - **Nueva sesión:** Mediante esta función el terapeuta es capaz de agregar una nueva sesión a un plan en la base de datos. Un alta hace posible la inserción de determinados datos como el nombre, una descripción, observaciones, el número de orden y las tres tareas que van a componer dicha sesión. Ninguno de los campos asociados a las tareas puede dejarse sin elegir y tampoco se pueden repetir. El número de orden podrá ser 0 para poder establecer un orden con las siguientes sesiones.

Pero si al querer validar el plan, las sesiones no tienen un orden correlativo, no se podrá validar.

❖ **Modificar sesión:** Mediante esta función, el terapeuta será capaz de realizar cambios en los datos de la sesión escogida. Esta función es una especialización de *Nueva sesión*, ya que realiza una operación de registro en la base de datos pero para una actualización de sus datos y no una inserción nueva completa. Para acceder a esta función, el terapeuta deberá entrar en la parte de administración y, tras ver el listado de planes actuales, acceder al plan que desea modificar, dónde a su vez podrá ver un listado de las sesiones que componen dicho plan y así elegir la que desea modificar. Las sesiones que pertenezcan a planes no modificables no podrán ser modificadas.

- **Consultar datos sesión:** Esta función representa la capacidad de la función *Modificar sesión* de mostrar la información actual y previa a la modificación de la sesión.

❖ **Eliminar sesión:** También es posible la eliminación de un registro de la tabla de sesiones en la base de datos. Mediante esta función, el terapeuta es capaz de eliminar el registro en su totalidad junto con los datos de la sesión. No siempre será posible eliminar un plan, ya que si este ha sido asignado a algún plan que no sea modificable, no será posible modificar ni eliminar sus sesiones. Esto es así para mantener la integridad de los datos en la base de datos.

- **Eliminar plan:** También es posible la eliminación de un registro de la tabla de planes en la base de datos. Un terapeuta, mediante esta función, es capaz de eliminar el registro en su totalidad junto con los datos del plan. No siempre será posible eliminar un plan, ya que si este ha sido asignado a algún sujeto y forma parte de plan de entrenamiento

atencional, para mantener la integridad de los datos en la base de datos, la aplicación no permitirá eliminar ese plan.

Detallado el diagrama de casos de uso correspondiente a la función *Administrar planes,* se verá el que corresponde a la función *Administrar terapeutas,* detallado en la Figura 3.4. En este caso, el diagrama es más sencillo que los anteriores, pero no por ello menos importante.

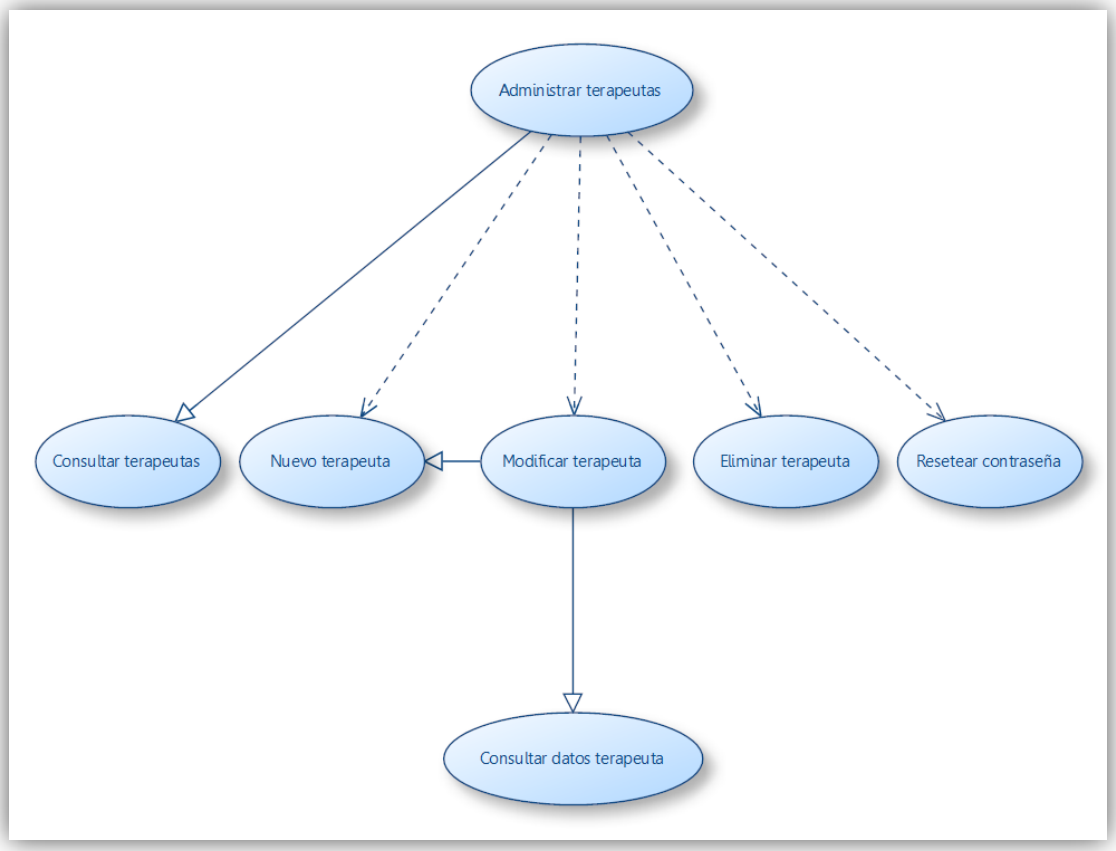


Figura 3.4 – Diagrama de casos de uso del segundo nivel de la función Administrar terapeutas.

A continuación se detallan las funciones que están presentes en la Figura 3.4 correspondiente al diagrama de casos de uso de la función *Administrar terapeutas*:

- **Consultar terapeutas:** Esta clase abstracta es la que proporciona el acceso a las funciones de administración de los terapeutas. *Administrar*

terapeutas hereda de aquí la funcionalidad de mostrar un listado de sujetos que existen en la base de datos.

- **Nuevo terapeuta:** Mediante esta función el terapeuta es capaz de dar de alta un nuevo usuario de tipo terapeuta en la base de datos. Un alta hace posible la inserción de determinados datos personales del terapeuta tales como el nombre de usuario, nombre, apellidos, ciudad, email, cargo, organización y la condición de ser administrador o no en el sistema. Una vez dada el alta de usuario, la contraseña será "1234" por defecto, debiendo ser el usuario el que cambie esta contraseña cuando lo crea conveniente.

- **Modificar terapeuta:** Mediante esta función, el terapeuta será capaz de realizar cambios en la información del resto de terapeutas. Esta función es una especialización de *Nuevo terapeuta*, ya que realiza una operación de registro en la base de datos pero para una actualización de sus datos y no una inserción nueva completa. Para acceder a esta función, el terapeuta deberá entrar en la parte de administración y, tras ver el listado de terapeutas actuales, acceder al terapeuta que desea modificar.

    - ❖ **Consultar datos terapeuta:** Esta función representa la capacidad de la función *Modificar terapeutas* de mostrar la información actual y previa a la modificación del sujeto.

- **Eliminar terapeuta:** También es posible la eliminación de un registro de terapeutas en la base de datos. Un terapeuta, mediante esta función es capaz de eliminar el registro en su totalidad junto con los datos personales facilitados. No siempre será posible eliminar un terapeuta, ya que si este terapeuta ha creado nuevos sujetos y éstos han participado en planes de entrenamiento, el sistema bloqueará el proceso de eliminación para evitar que existan datos no consistentes en la base de datos.

- **Resetear contraseña:** Los terapeutas administradores tienen la posibilidad de restablecer la contraseña perdida u olvidada de otros usuarios. En ningún caso cambian la contraseña a su criterio, solamente

pueden volverla al estado inicial de "1234", siendo el propio usuario el que la cambie en el siguiente acceso al sistema. De esta manera se evita que una contraseña olvidada impida el acceso a la aplicación y que usuarios avanzados cambien las contraseñas a voluntad.

Una vez detallado el diagrama de casos de uso correspondiente a la función *Administrar terapeutas,* se detallará el diagrama de casos de uso que corresponde a la función *Ejecutar tareas.*

Como se detalló en el diagrama de casos de uso inicial que se puede observar en la Figura 3.1., los terapeutas tienen la posibilidad de realizar pruebas con las tareas diseñadas. En general, todas las tareas tienen un mismo patrón de ejecución, es decir, se muestra un ensayo y el sujeto, en este caso terapeuta, debe responder ante ese estímulo, siendo el sistema el que muestra el feedback en función de esa respuesta dada.

La principal diferencia de esta función con *Ejecutar sesión*, como se podrá observar en la figura 3.6 donde detallaremos la mencionada función, es que el actor que da respuesta a los estímulos de ejecución es un sujeto y que el mismo sistema selecciona el nivel inicial de la ejecución, siendo el terapeuta el que configura el tiempo total de la sesión.

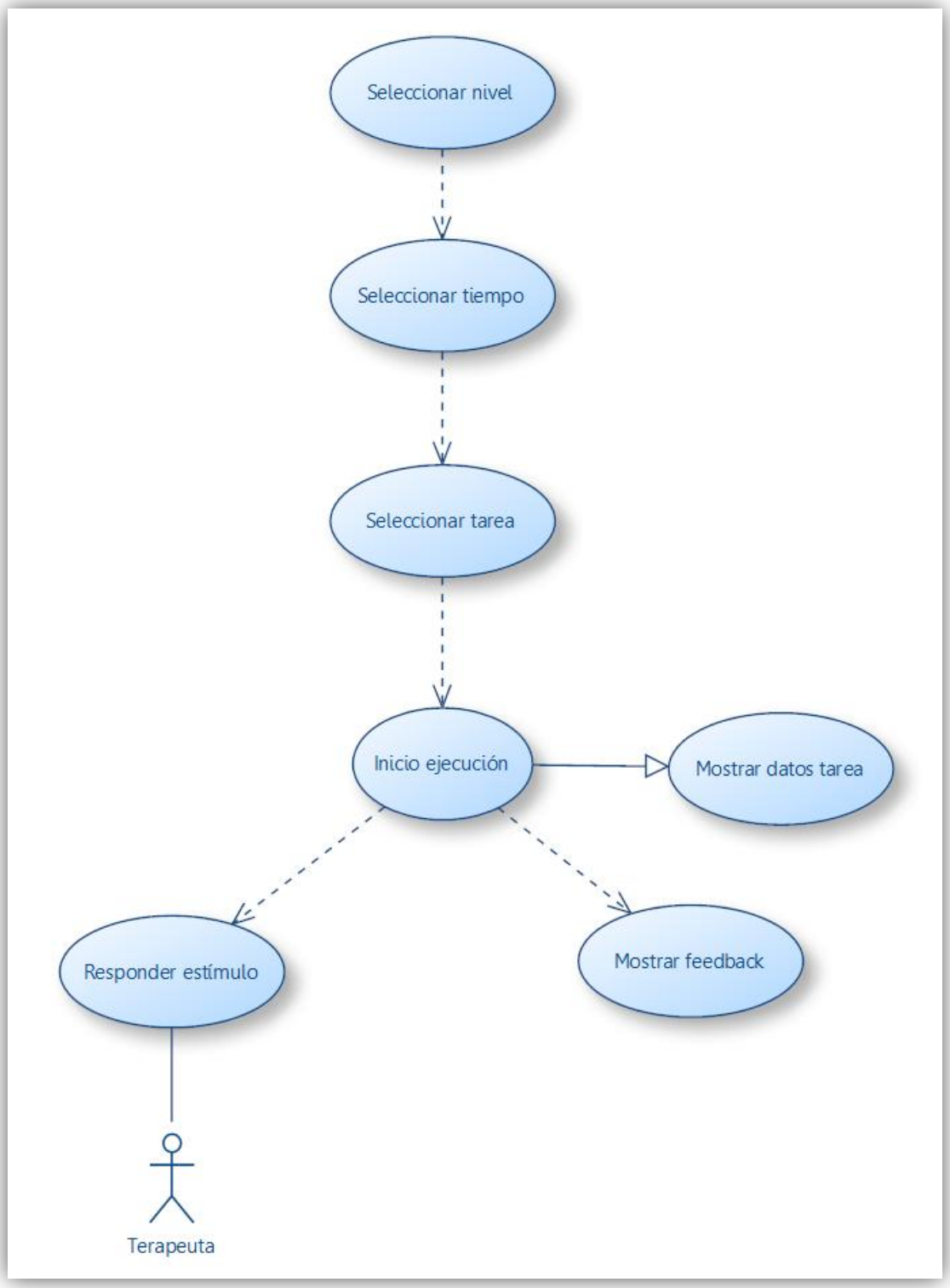


Figura 3.5 – Diagrama de casos de uso del segundo nivel de las función Ejecutar tarea.

A continuación se detallarán las funciones que están presentes en la Figura 3.5 correspondiente al diagrama de casos de uso de la función *Ejecutar tarea*:

- **Seleccionar nivel:** Una vez se está dentro del sistema de *Ejecutar tareas,* lo primero que se debe hacer es elegir el nivel inicial de la tarea. Por defecto el nivel es el 1 y el máximo el 6. Éstos son los niveles de dificultad con los que cuentan las tareas diseñadas.

- **Seleccionar tiempo:** Cuando se ha seleccionado el nivel inicial se debe hacer lo mismo con el tiempo de la tarea. Por defecto, el tiempo es de 3, expresado en minutos, y el máximo de 20.

- **Seleccionar tarea:** Una vez seleccionados los valores de los parámetros anteriores, el terapeuta debe seleccionar una de las diez tareas disponibles para ser probadas. Para seleccionar una, simplemente se debe hacer clic sobre la imagen de tarea que se desee.

- **Inicio ejecución:** Una vez configurada la aplicación, se pasa a la ejecución con una previa consulta a la base de datos para mostrar la descripción de dicha tarea.

    - **Mostrar datos tarea:** Este caso abstracto hace referencia a la capacidad de la función de modificación de consultar primero los datos de la tarea que se va a ejecutar.

- **Responder estímulo:** La tarea consta de varios niveles y cada nivel de varios ensayos. A mayor nivel, mayor dificultad, pero para subir de nivel habrá que contestar correctamente a una serie de ensayos. Por cada ensayo se muestra un estímulo que el terapeuta deberá responder para ser evaluado.

- **Mostrar feedback:** Por cada respuesta a un estímulo, sea correcta o incorrecta, el sistema mostrará un feedback. Esto es, mostrará un estímulo visual indicando si la respuesta ha sido correcta o no.

Vista la descripción del diagrama de casos de uso correspondiente a la función *Ejecutar tarea,* se mostrará el diagrama y la descripción de los casos de uso de la función *Ejecutar sesión.* Como se observa en la Figura 3.6., existen ciertas similitudes con el caso anterior, y es que salvo por los actores, son iguales (en este caso el actor principal es el sujeto, aunque la configuración previa la realiza el terapeuta).

Este diagrama de casos de uso proviene de la función *Ejecutar sesión*, perteneciente a la función *Administrar sujetos*, vista con anterioridad en el diagrama de la Figura 3.2.

Como podemos observar en la Figura 3.6., el actor principal en este caso de uso es el sujeto, puesto que se está ejecutando una de las sesiones de un plan de entrenamiento atencional directamente diseñados para ellos. Este actor no ha aparecido en ningún caso anterior puesto que es el terapeuta el que debe realizar todo el trabajo de administración.

Como se ha descrito en el caso de uso anterior, existen funciones similares. No obstante, se detalla una descripción de cada función tras el diagrama.

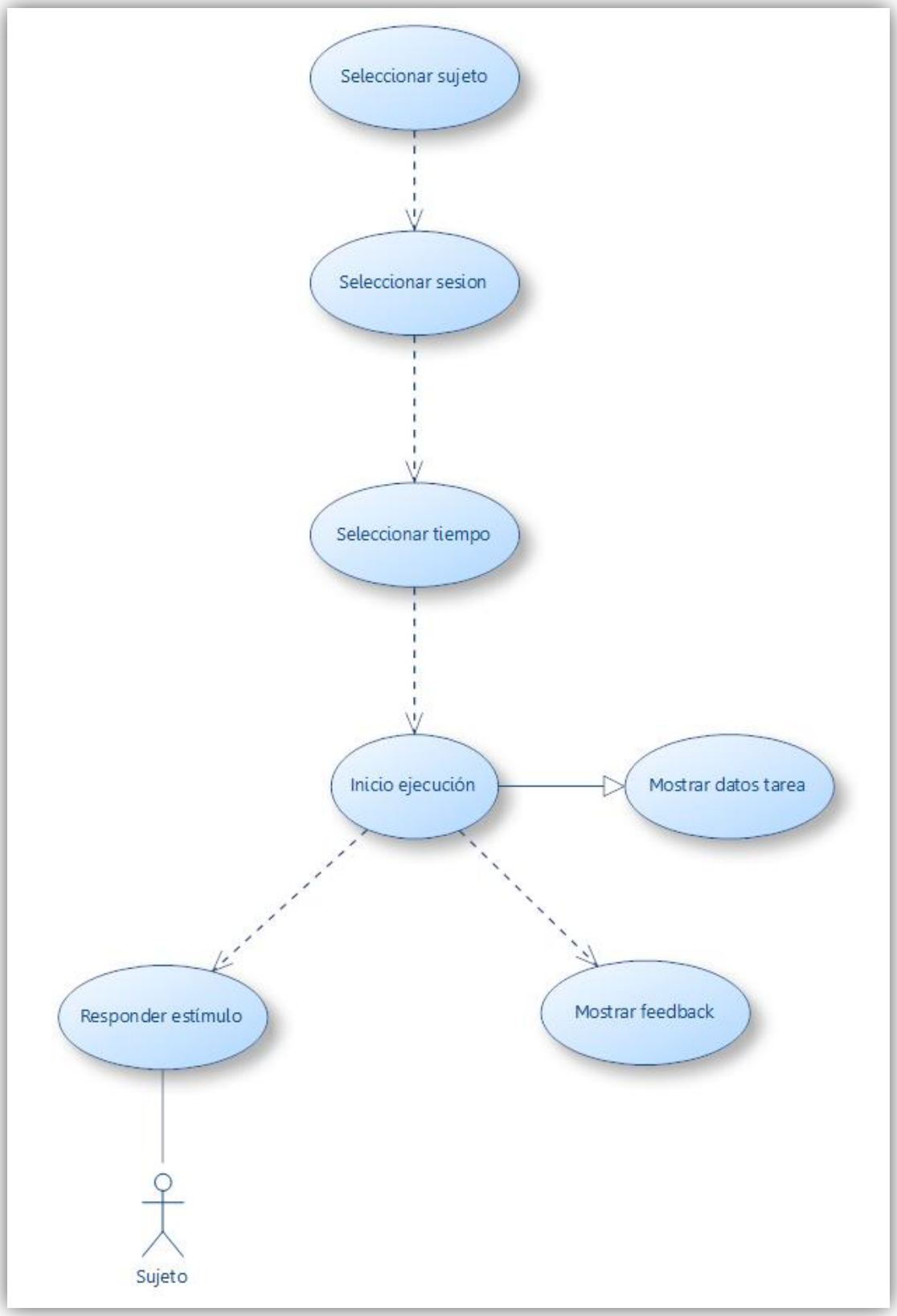


Figura 3.6 – Diagrama de casos de uso del segundo nivel de las función Ejecutar sesión.

A continuación se detallan las funciones que están presentes en la Figura 3.6 correspondiente al diagrama de casos de uso de la función *Ejecutar sesión*:

- **Seleccionar sujeto:** Una vez ejecutada la parte de administración de sujetos del sistema, para ejecutar una sesión destinada a un sujeto se debe seleccionar uno de una lista de sujetos. Una vez seleccionado y entrado en su detalle del plan, se podrá seleccionar y configurar una sesión.

- **Seleccionar sesión:** Seleccionado el sujeto, se deberá elegir una de las sesiones de las que está compuesto el plan de entrenamiento que dicho sujeto está siguiendo.

- **Seleccionar tiempo:** Cuando está seleccionada la sesión, habrá que marcar el tiempo, en minutos, que se desea que la sesión dure. En este caso, a diferencia de la función *Seleccionar tiempo* del caso de uso correspondiente a *Ejecutar tarea*, el tiempo es el total de la sesión, no el individual de la tarea.

- **Inicio ejecución:** Una vez configurado el sistema, se procede a la ejecución con una consulta previa a la base de datos para mostrar la descripción de dicha tarea. Este paso será previo a la ejecución de cada una de las tres tareas que componen la sesión.

    - **Mostrar datos tarea:** Este caso abstracto hace referencia a la capacidad de la función de modificación de consultar primero los datos de la tarea que se va a ejecutar.

- **Responder estímulo:** Una tarea de las tres que componen la sesión, consta de varios niveles y cada nivel de varios ensayos. A mayor nivel, mayor dificultad, pero para subir de nivel habrá que contestar correctamente a una serie de ensayos. Por cada ensayo se muestra un estímulo que el terapeuta deberá responder para ser evaluado.

- **Mostrar feedback:** Por cada respuesta a un estímulo, sea correcta o incorrecta, el sistema mostrará un feedback. Esto es, mostrará un

estímulo visual indicando si la respuesta ha sido correcta o no. Un ejemplo de feedback, es mostrar una imagen que simboliza un acierto ante una respuesta positiva.

3.3. Diagrama conceptual de clases.

En esta sección se detalla un diagrama de clases informal que será la base usada para completar la descripción formal en la fase de diseño en el siguiente capítulo.

El diagrama se muestra en la Figura 3.7. y en él se puede observar la estructura lógica básica de las clases que forman el prototipo de la herramienta web.

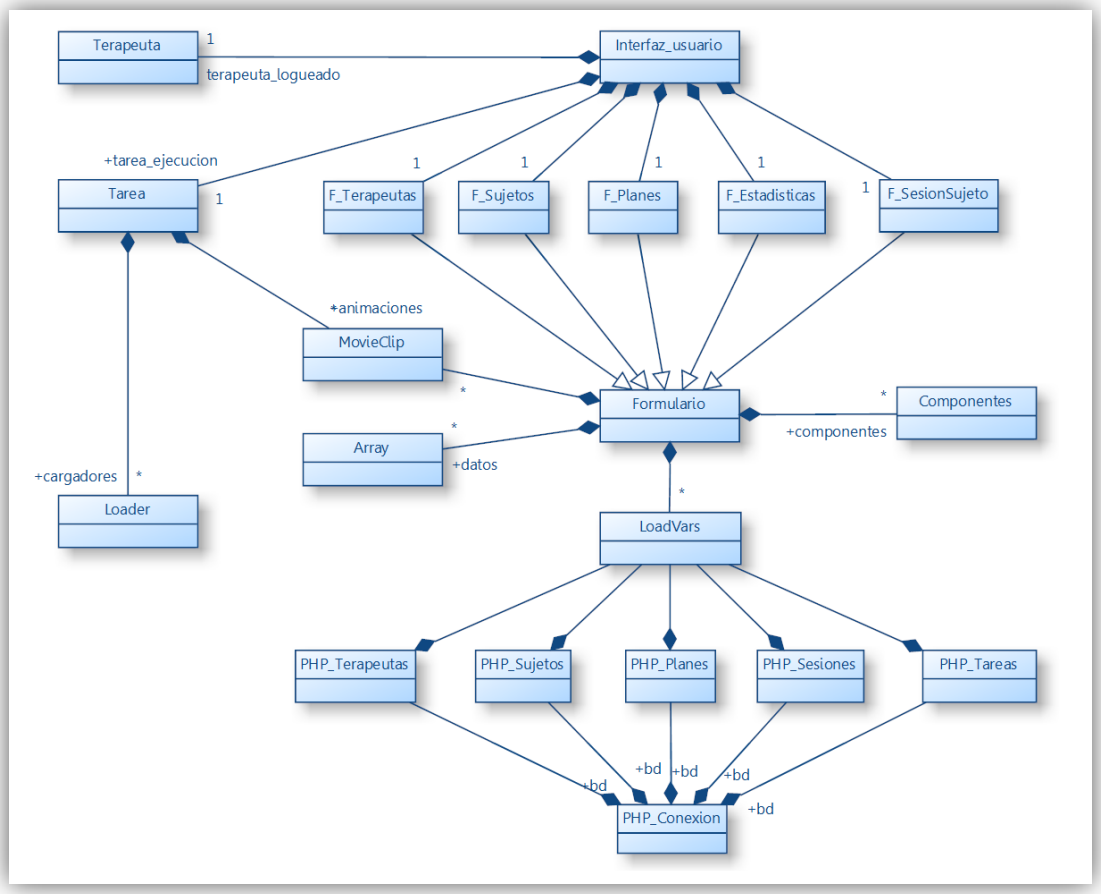


Figura 3.7 – Diagrama de clases informal.

Como se puede observar en el diagrama de clases anterior, la clase que se ha representado como Interfaz_usuario será el archivo *Flash* que contendrá al resto de estructuras utilizadas. Una de las más utilizadas es Formulario, de la que partirán los distintos formularios para interactuar con la base de datos. A la hora de realizar un formulario se han utilizado distintas estructuras que dan forma a lo que el terapeuta ve cuando interacciona con la base de datos, como botones, áreas de texto, animaciones, así como los diferentes conjuntos de datos.

Pese a existir muchos usuarios, terapeutas, y diez tareas, en el diagrama se representan sólo los que están en una única ejecución, en dicho caso solamente está el terapeuta registrado en el sistema y la tarea que se está ejecutando en un instante de tiempo determinado.

En el siguiente capítulo, Diseño e implementación, se detallará este diagrama realizando un diagrama formal de clases.

Capítulo 4 – Diseño e implementación

En este capítulo se describe el diseño del sistema, cuyo desarrollo nos servirá de base para el proceso de implementación posterior del prototipo. En el primer apartado se describe el diagrama de clases formal de la aplicación. Después se pasará a detallar el diseño de la interfaz de usuario y los diagramas de secuencia de la conexión entre la interfaz y la base de datos. Por último, se verá el diseño de la base de datos.

4.1. Diagrama de clases formal.

En las siguientes figuras, Figura 4.1 y Figura 4.2, está representado el diagrama formal de clases correspondiente al desarrollo del prototipo. Dado el gran tamaño y por mantener una calidad aceptable de la imagen, se ha decidido separar en dos figuras independientes, aunque ambas son parte del mismo diagrama.

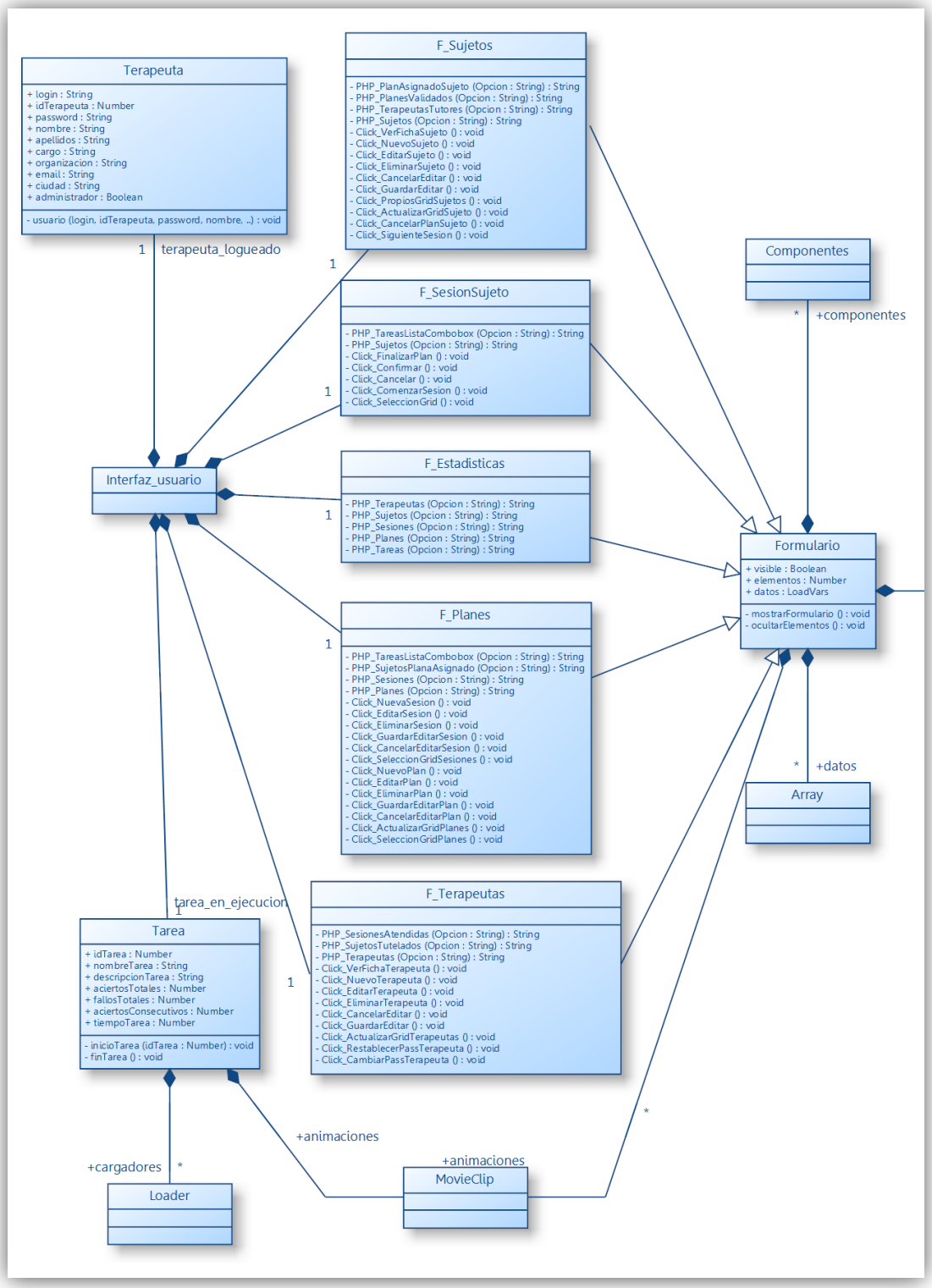


Figura 4.1 – Primera parte del diagrama formal de clases.

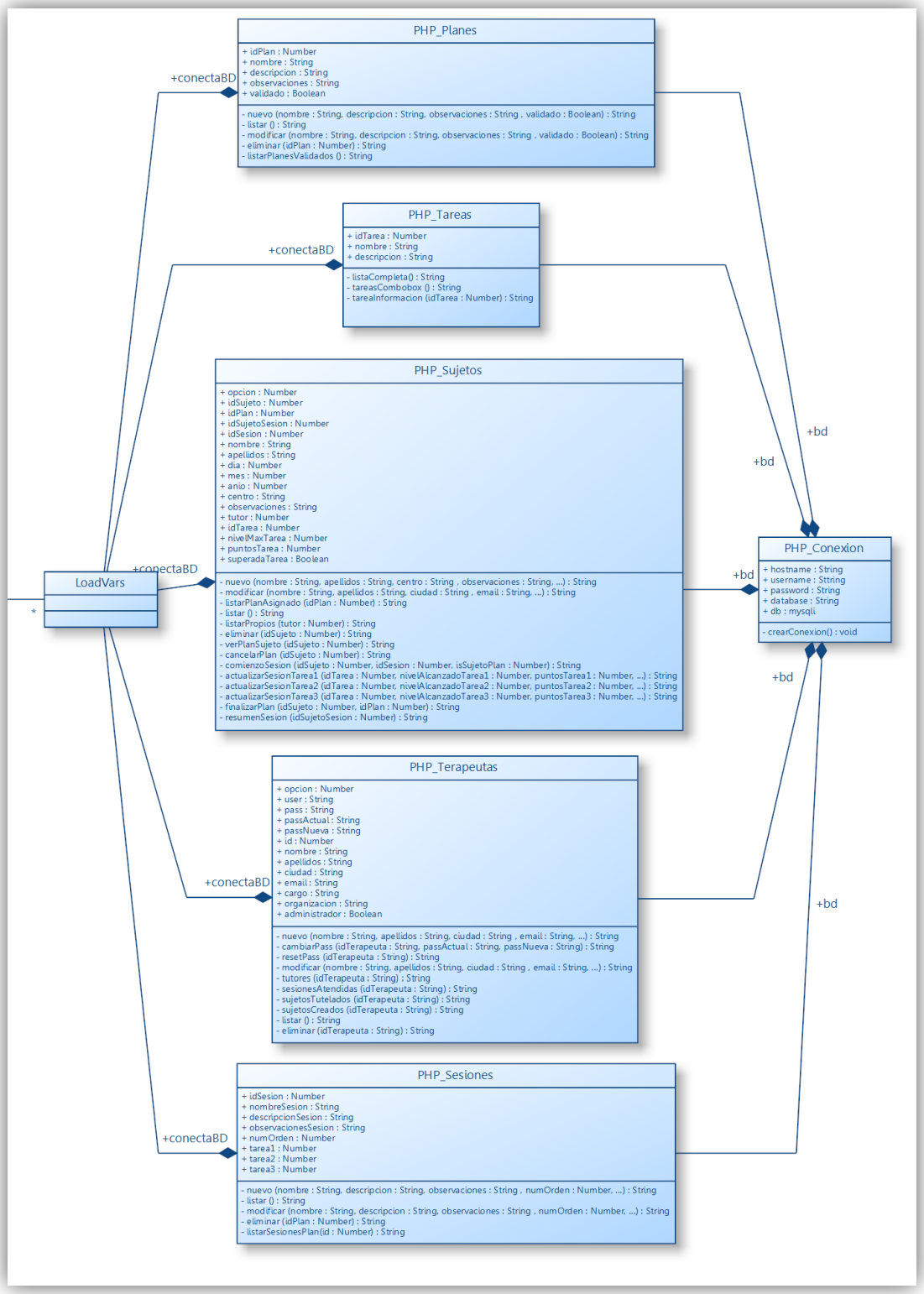


Figura 4.2 – Segunda parte del diagrama formal de clases.

Una vez observado el diagrama formal de clases, se describirán las clases representadas:

- **Terapeuta:** Esta clase representa a un usuario conectado en la aplicación web. Tiene como atributos todos los campos que se requieren y como métodos el constructor de la clase.

- **Tarea:** Esta clase representa una tarea en ejecución en la aplicación web. Como atributos tiene aquellos valores que son necesarios en su ejecución y el único método es el constructor de la clase.

- **Interfaz_usuario:** Representa el archivo *Flash* que contiene la interfaz de usuario que contiene toda la estructura de formularios y componentes para representar los datos a tratar.

- **Formulario:** Esta clase representa la estructura de datos principal, que es común al resto de formularios utilizados en la interfaz. Con ella se controlan las principales propiedades de visibilidad y muestra de datos.

- **F_Sujetos:** Esta clase representa el formulario de sujetos y, además de heredar de la clase Formulario sus propiedades, introduce nuevos métodos particulares para su correcto funcionamiento. Con este formulario se gestionará toda la información que hay almacenada en la base de datos referente a los sujetos.

- **F_SesionSujeto:** Esta clase representa el formulario del detalle del plan de los sujetos y, además de heredar de la clase Formulario sus propiedades, introduce nuevos métodos particulares para su correcto funcionamiento. Con este formulario se gestionará toda la información que hay almacenada en la base de datos referente a los sujetos y el desarrollo del plan que están llevando a cabo. Además, desde este formulario será posible lanzar sesiones.

- **F_Estadisticas:** Esta clase representa el formulario de estadísticas y además de heredar de la clase Formulario sus propiedades, introduce nuevos métodos particulares para su correcto funcionamiento. Con este formulario no se realizarán modificaciones en la base de datos pero si será posible consultar todo tipo de información.

- **F_Planes:** Esta clase representa el formulario de planes y, además de heredar de la clase Formulario sus propiedades, introduce nuevos métodos particulares para su correcto funcionamiento, como por ejemplo el mostrar u ocultar la parte de administración de las sesiones. Con este formulario se gestiona toda la información que hay almacenada en la base de datos referente a los planes y las sesiones que componen dichos planes. Desde aquí será posible validar los planes que posteriormente serán asignados a los sujetos.

- **F_Terapeutas:** Con este formulario se gestionará toda la información que hay almacenada en la base de datos referente a los terapeutas. En este formulario se administrará la información sobre los usuarios que tienen acceso a la aplicación web.

- **PHP_Conexion:** Esta clase de apoyo al resto de clases que conectarán con la base de datos para introducir u obtener datos, es la encargada de establecer la conexión con la base de datos. Tiene los atributos necesarios para establecer la conexión y el método que establece dicha conexión.

- **PHP_Planes:** Esta clase, haciendo uso de la clase PHP_Conexion, contiene todos los métodos y atributos necesarios para realizar los cambios y consultas a la base de datos que tengan información relacionada con la tabla plan.

- **PHP_Tareas:** Esta clase, haciendo uso de la clase PHP_Conexion, contiene todos los métodos y atributos necesarios para realizar

los cambios y consultas a la base de datos que tengan información relacionada con la tabla tarea.

- **PHP_Sujetos:** Esta clase, haciendo uso de la clase PHP_Conexion, contiene todos los métodos y atributos necesarios para realizar los cambios y consultas a la base de datos que tengan información relacionada con la tabla sujeto.

- **PHP_Terapeutas:** Esta clase, haciendo uso de la clase PHP_Conexion, contiene todos los métodos y atributos necesarios para realizar los cambios y consultas a la base de datos que tengan información relacionada con la tabla terapeuta.

- **PHP_Sesiones:** Esta clase, a través de PHP_Conexion, contiene todos los métodos y atributos necesarios para realizar los cambios y consultas a la base de datos que tengan información relacionada con la tabla sesión.

4.2. Diseño de interacciones.

En AT²-Training las interacciones entre la parte del cliente y el servidor son muy puntuales, centrándose en casos concretos de gestión o actualización de datos. En ningún caso se mantiene una conexión permanente de datos, sino que se realizan conexiones con la base de datos bajo demanda. Esto se realiza así porque la interacción con las pruebas se realiza exclusivamente del lado del cliente.

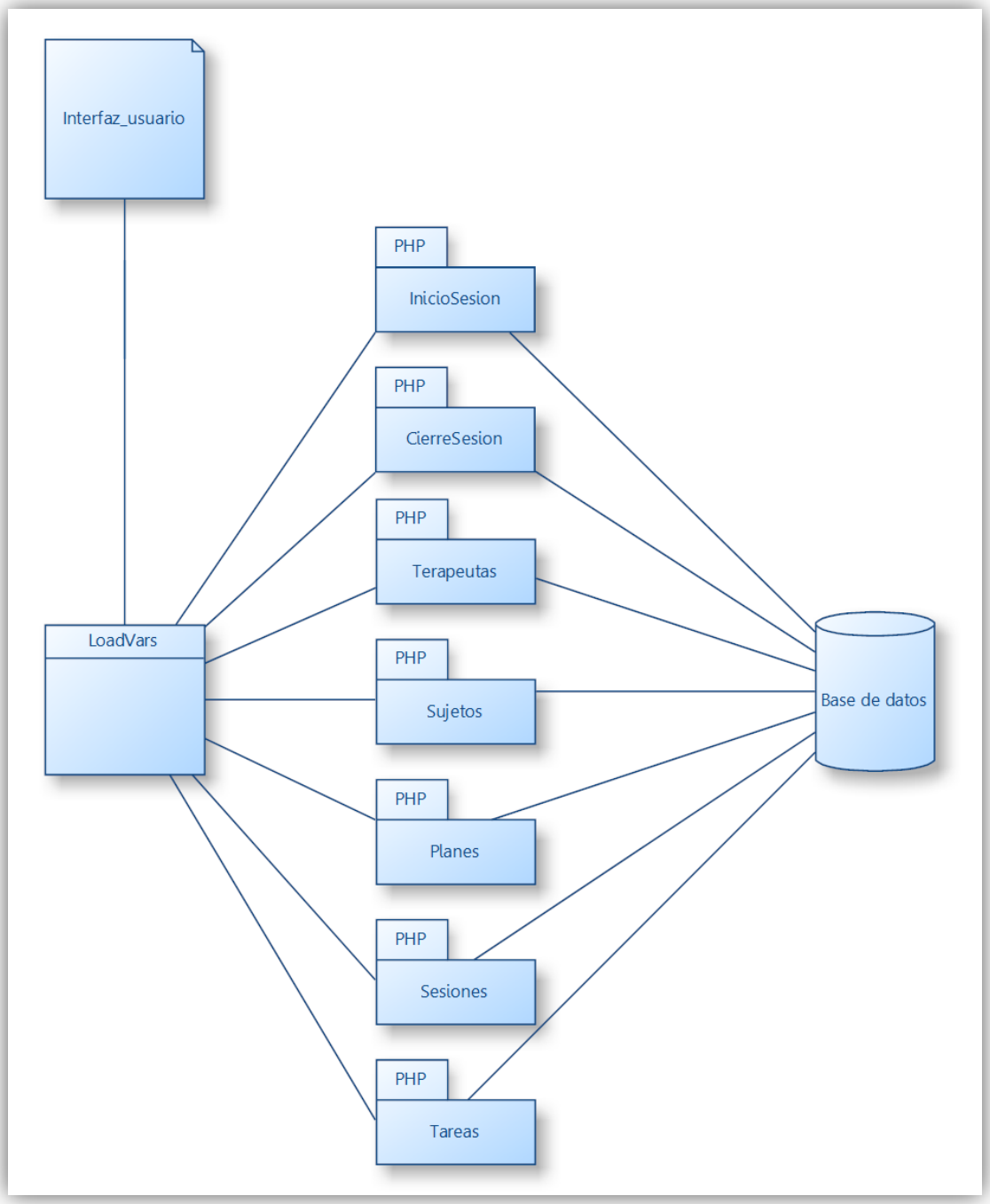


Figura 4.3 – Diagrama de interacciones.

Como se puede observar en el diagrama de interacciones representado en la Figura 4.3, la interfaz de usuario en *Flash* hace uso de la clase LoadVars para conectarse con la base de datos mediante PHP. Esta clase propia de *Flash* envía texto formateado

con GET/POST, método que resulta una gran solución para las pocas interacciones en tiempo de ejecución que se necesitan con la base de datos.

Por no hacer un diagrama ilegible por la gran cantidad de líneas de relación, no se han incluido dos páginas PHP que se usan a modo de paquete en las siete que si se ven en el diagrama: (1) Conexión, la cual establece la conexión con la base de datos; y, (2) MensajesServidor, que incluye un array con mensajes de respuesta según la operación que se haya realizado.

4.2.1. Diagramas de secuencia.

En este apartado se describen en detalle las interacciones que se realizan con la base de datos vistas en el apartado anterior.

Para ello, serán descritas todas las páginas PHP creadas y se realizará un diagrama a modo de ejemplo con una de las funciones que contienen.

En primer lugar se explicarán dos archivos PHP que hacen de soporte a los principales:

- **Conexion:** Esta página del servidor, incluida como paquete externo en el resto de archivos, realiza la conexión con la base de datos y devuelve la variable de conexión. De esta manera se evita tener que repetir el mismo código en el resto de archivos, pudiendo cambiar los valores de conexión en un solo lugar.

- **MensajesServidor:** Este script de servidor, también incluida como paquete externo en el resto de archivos, registra un *array* en el que almacena los mensajes informativos y de error que la aplicación lanzará según las conexiones. También de esta manera se consigue centralizar toda una misma información en un mismo lugar en vez de colocar cada mensaje en su archivo. De no ser así, este tipo de información estaría dispersa y sería difícil de mantener y modificar.

Una vez vistos los dos scripts que dan soporte al resto, se detallarán los scripts que hacen uso de ellas. También se podrán observar un diagrama de secuencia por cada script. Para evitar la redundancia de datos en los diagramas, los archivos de soporte no se han incluido en el tráfico de la secuencia, ya que cada vez que se requiere el uso de una aplicación PHP éstos son llamados.

- **InicioSesion:** Esta página será llamada desde el inicio de la interfaz de usuario, justo después de haber introducido los datos en el formulario del inicio de sesión. La interfaz, mediante el uso de la clase *LoadVars*, enviará el *login* y el *password* introducidos al archivo PHP. Con estos dos valores se hará la consulta a la base de datos. Una vez obtenida la respuesta, PHP comprobará el valor y si este es positivo actualizará el registro de la base de datos con los datos del terapeuta que va a entrar, el número de logins y la fecha y hora actuales. También devolverá a la interfaz todos los valores del registro para las futuras operaciones que realice el terapeuta en este inicio de sesión y los mostrará en pantalla. El diagrama correspondiente a esta página se puede observar en la Figura 4.4. En este caso, este script sólo contiene a ésta como única función.

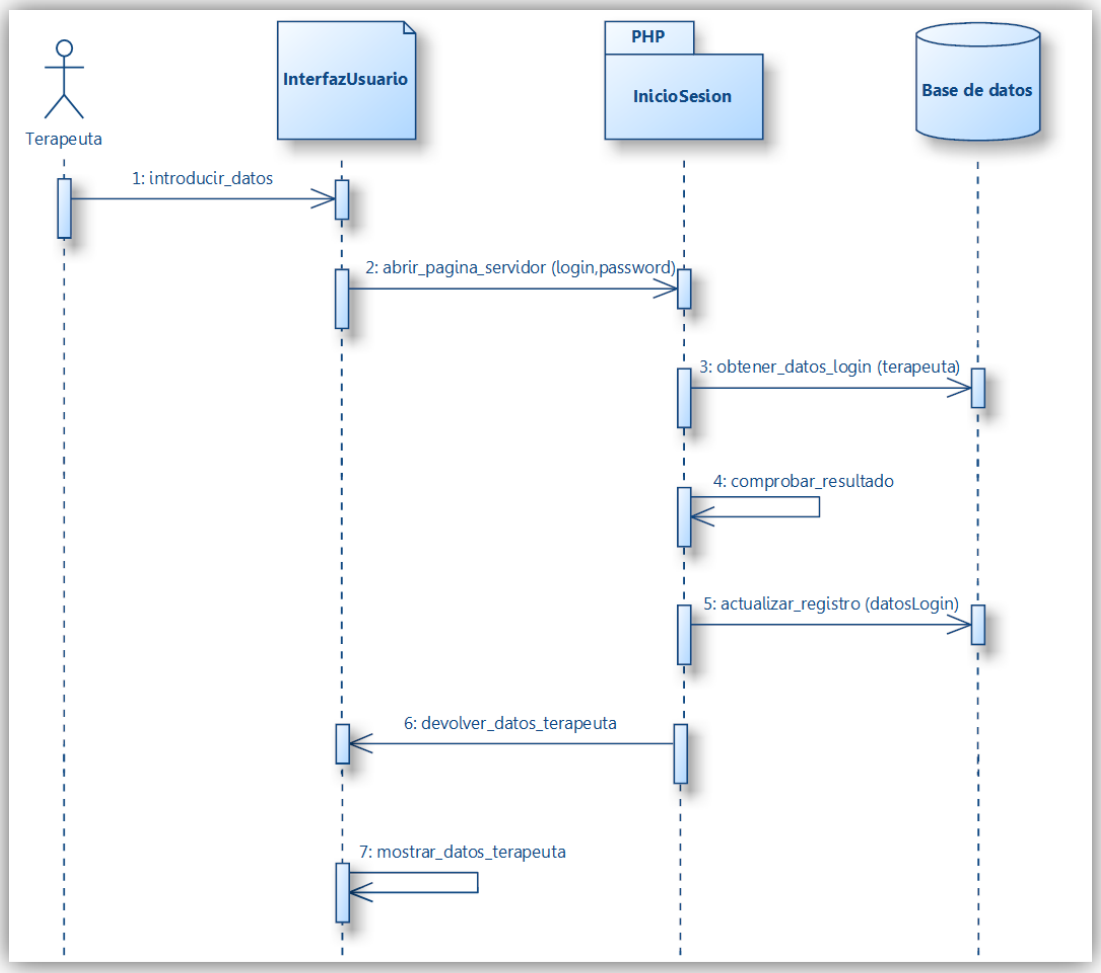


Figura 4.4 – Diagrama de secuencia de la función que inicia la sesión.

- **CierreSesion:** Para un correcto cierre de sesión y dar la posibilidad de cambiar de usuario, se deberá abrir la ventana correspondiente y aceptar el cierre de sesión. Es por esto que la interfaz hace uso de la ventana para ser mostrada en pantalla. Una vez ha sido aceptado el cierre de sesión, se actualizará el registro del terapeuta en la tabla correspondiente de la base de datos, devolviendo el mensaje positivo, y mostrando la pantalla inicial. El diagrama correspondiente a esta página se puede observar en la Figura 4.5.

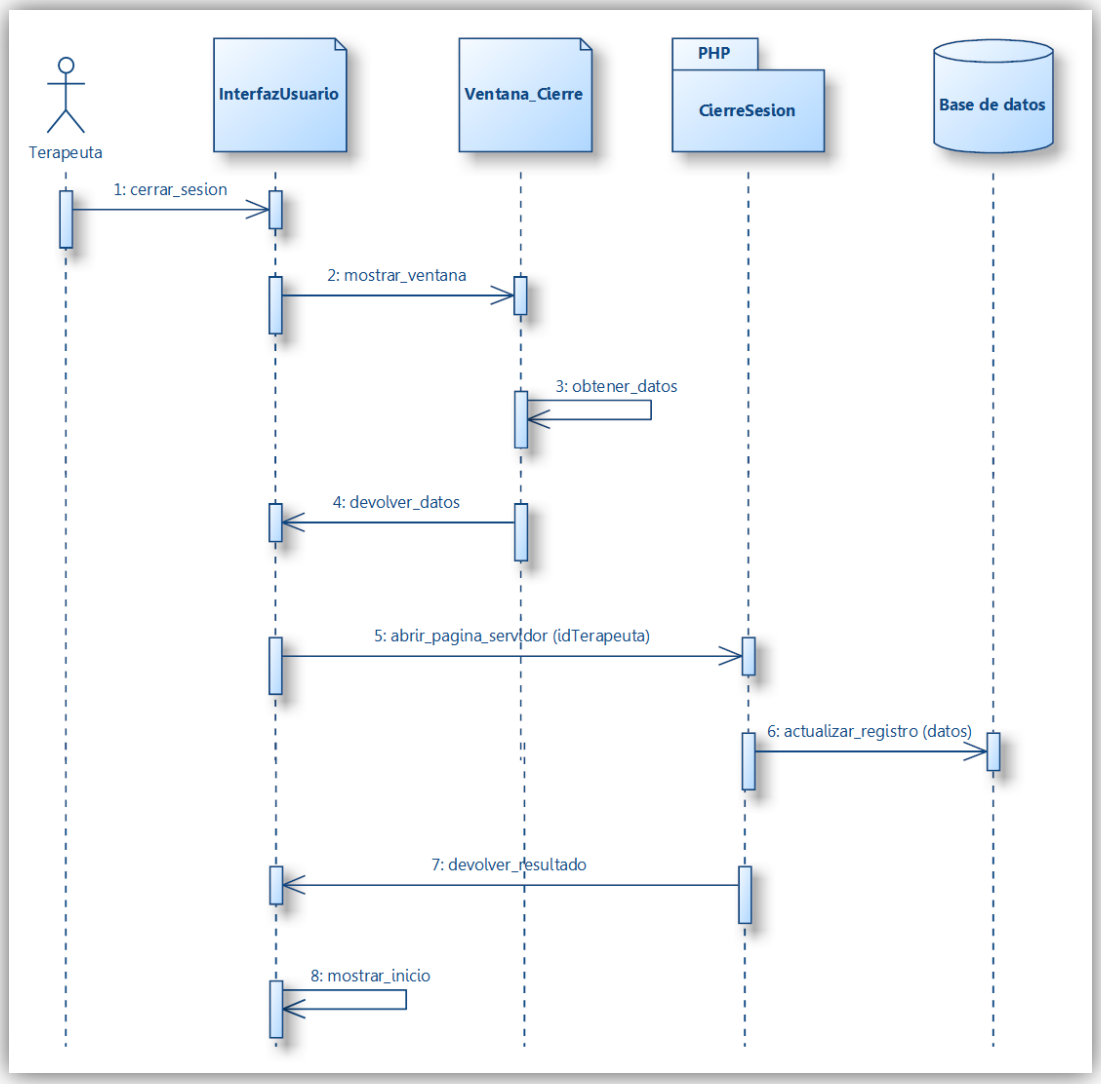


Figura 4.5 – Diagrama de secuencia de la función que cierra la sesión.

- **Terapeutas:** Este modelo contiene todas las funciones que manipulan la información contenida en los registros de la tabla terapeutas de la base de datos. Serán explicadas todas las funciones implementadas, pero en detalle se verá la que da de alta un nuevo registro, la función *Nuevo*, cuyo diagrama de secuencia se puede observar en la Figura 4.6.

  - **Nuevo:** Esta función lleva a cabo la tarea de introducir un nuevo registro en la tabla Terapeuta de la base de datos.

Como se puede observar en el diagrama de secuencia que corresponde a esta función, primero se debe abrir el formulario correspondiente a tal efecto. Una vez mostrado se introducirán los datos que se deseen y se comprobarán, desde el lado del cliente, que al menos están rellenos los datos imprescindibles y que éstos tienen una longitud adecuada. Estando todo validado se enviará la información a PHP para que éste haga el proceso de comunicación con la base de datos. Ya del lado del servidor, hará la inserción en la base de datos comprobando que el *login* introducido para éste nuevo terapeuta es único. Como respuesta a este proceso, se devuelven los datos introducidos y un listado actualizado de terapeutas existentes en la base de datos, cerrando el formulario y mostrando dicho listado. El diagrama correspondiente a esta función puede observarse en la figura 4.6.

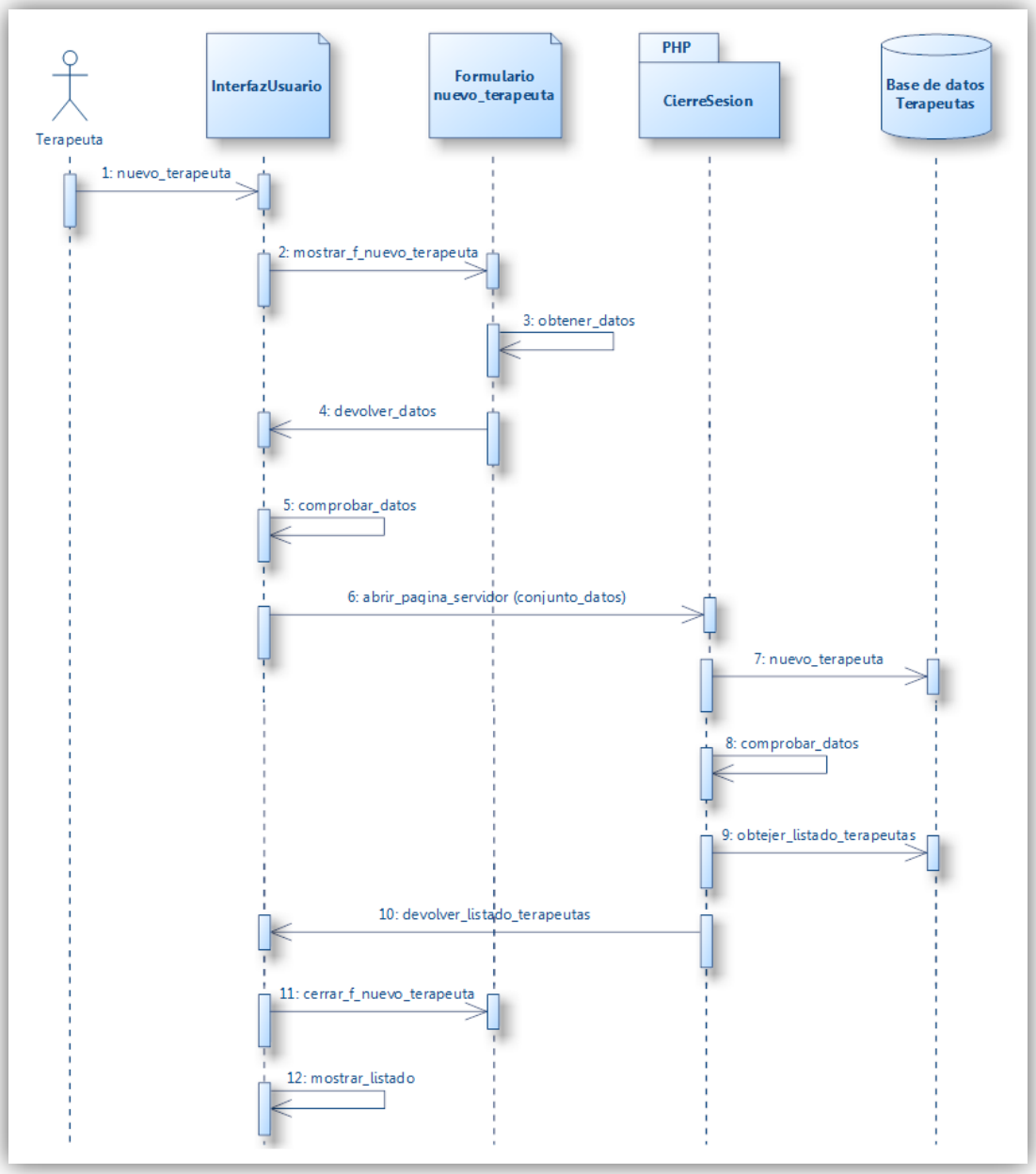


Figura 4.6 – Diagrama de secuencia de la función que da de alta un nuevo terapeuta.

- **CambiarPass:** Esta función será la encargada de cambiar la contraseña de los terapeutas que así lo deseen. Solamente podrá ser cambiada una contraseña por otra nueva por el propio terapeuta. Un administrador no podrá realizarlo, pero si resetearla, como se verá en la siguiente función. Para que se realice correctamente el cambio de contraseña, del lado

del cliente se comprobará que la contraseña nueva se ha introducido dos veces de forma correcta, y del lado del servidor que la contraseña actual y el usuario coinciden.

- **ResetPass:** A diferencia de la función anterior, ésta será llevada a cabo exclusivamente por terapeutas administradores. El objetivo principal de esta función es restablecer la contraseña de un usuario que haya olvidado su actual contraseña por una estándar, en este caso, "1234", para que sea el propio terapeuta el que pueda cambiarla por otra propia. Restablecer una contraseña simplemente actualiza el valor del campo correspondiente al terapeuta objeto de la acción a "1234".

- **Modificar:** Esta función es doblemente utilizada: se utiliza tanto para actualizar un registro terapeuta por medio de un administrador, como para actualizar los datos personales propios de un terapeuta. Las comprobaciones de la validez de los datos se realizan todas del lado del cliente, puesto que el *login* no se puede modificar y la contraseña se cambia mediante otra función.

- **SesionesAtendidas:** Esta función realiza una consulta del número de sesiones que ha atendido un terapeuta. Para ello necesita del identificativo del terapeuta y busca todas las coincidencias en la tabla *sujeto_sesion*.

- **SujetosTutelados:** Para conocer el número de sujetos (además de otra información sobre ellos) que tiene un terapeuta a su cargo (aquellos que ha dado de alta él mismo), se utiliza esta función. Para ello se realiza una búsqueda cruzada en la tabla Sujeto y Sujeto_Plan donde el identificativo de tutor es el del terapeuta sobre el que buscamos dicha información. Ésta nos devolverá un listado de sujetos con nombre e información sobre su estado actual

referente a un plan de entrenamiento: si tiene uno asignado y si este está activo.

- **Listar:** Esta función es la encargada de recoger de la base de datos toda la información de los terapeutas para elaborar un listado. Dicho listado consta de todos los campos públicos de los terapeutas, todos excepto el nombre de usuario y la contraseña. Además, también recoge el número de terapeutas que hay registrados.

- **Eliminar:** Esta función es la encargada de eliminar un terapeuta de la base de datos. Para ello deberá realizar unas comprobaciones del lado del servidor. Esta realiza una consulta especificando el ID del terapeuta a eliminar en las tablas Plan, Sesión y Sujeto, de forma que si hubiese algún registro con su ID, no permitiría el borrado de la base de datos. Esto se realiza para que la información almacenada en la base de datos no quede inconsistente y sea sólida**.**

- **Sujetos:** Esta página contiene todas las funciones que manipulan la información contenida en los registros de la tabla sujetos de la base de datos. Serán explicadas todas las funciones pero en detalle se verá la que da de alta un nuevo registro, la función *Modificar*, cuyo diagrama de secuencia se puede observar en la Figura 4.7.

  - **Nuevo:** Esta función introduce un nuevo registro en la tabla sujetos de la base de datos. Para ello se deberán introducir los datos que se deseen y, tras comprobarlos la interfaz, enviarlos al archivo PHP para que haga la introducción en la base de datos.

  - **Modificar:** Como se puede observar en la Figura 4.7, para realizar una modificación de datos de un sujeto, primero se ha de abrir el formulario de los sujetos. Una vez abierto, se realiza una consulta a la base de datos para obtener la información sobre el sujeto a modificar y, posteriormente, se

modifican los datos requeridos. Una vez modificados, éstos pasarán a la base de datos donde serán registrados y, a su vez, se devolverán para ser observados en la interfaz cerrando el formulario de modificación.

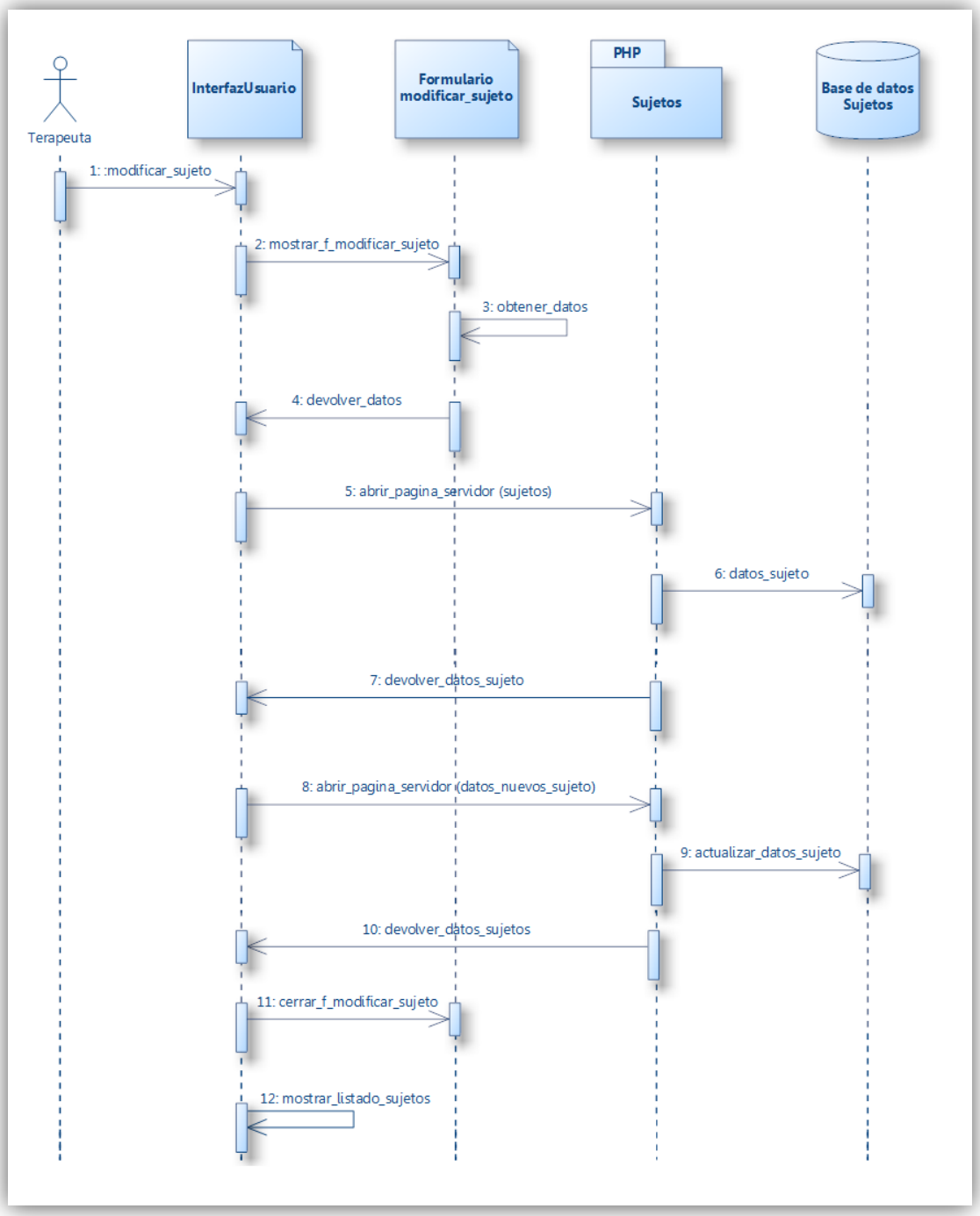


Figura 4.7 – Diagrama de secuencia de la función que modifica un sujeto.

- **ListarPlanAsignado:** Esta función es utilizada para listar los sujetos que han realizado o están realizando un mismo plan. De esta manera se conocerá la influencia de un plan sobre los sujetos y si éste es muy utilizado o repetido. Para realizar esta búsqueda se necesita el ID del plan y devuelve un listado de sujetos distintos con su nombre completo, así como el número de veces que han realizado el plan y si este está activo. La búsqueda se realiza sobre la tabala sujeto_plan.

- **Listar:** Para esta función no se requiere ningún atributo y el resultado es una lista de todos los elementos de la tabla Sujetos junto con sus atributos. Esta función la usa el formulario de los sujetos para mostrar una tabla con información sobre ellos.

- **ListarPropios:** Al igual que la función anterior, Listar, esta función devuelve la misma lista de sujetos pero filtrada por el campo de tutor. De esta manera el formulario sujetos solo mostrará los sujetos que han sido dados de alta por el terapeuta que está consultando la base de datos.

- **Eliminar:** Esta función elimina un sujeto de la base de datos. Para ello solo necesita el ID del sujeto a eliminar, devolviendo un mensaje de éxito o error según proceda. El error que se puede dar en este caso es que el sujeto tenga relaciones creadas con algún plan. Si tuviesen asignado un plan, ya no se podría eliminar, evitando así que hubiese datos inconsistentes. Esto sucede de la misma manera que con los terapeutas en su función de eliminar.

- **VerPlanSujeto:** Mediante esta función se accede a toda la información sobre el plan actual del sujeto que consultemos. Requiere del ID del sujeto a consultar y devuelve como respuesta el ID del plan asignado, la fecha de asignación, el

nombre y la fecha de inicio de la tabla plan; y la fecha de la última sesión, las sesiones realizadas y las repetidas de la tabla sujeto_sesion.

- **CancelarPlan:** Esta función cancela un plan, a petición de un terapeuta, para dejar al sujeto libre de planes y poder asignarle uno nuevo. Para realizar esta acción será necesario el ID del terapeuta que manipula la información y el ID del sujeto al que se le va a cancelar el plan. Los cambios se realizarán sobre el registro activo de la tabla sujeto_plan.

- **ListarSesionesSujetoPlan:** Mediante esta función se consulta toda la información sobre el plan que actualmente está desarrollando un sujeto. Para ello se utiliza el ID del sujeto y del plan actual de ese sujeto. Como valores de resultado se obtienen todas las sesiones de ese plan con sus tareas y puntuaciones obtenidas en cada tarea y sesión. Esta es una de las funciones más importantes por la información tan valiosa que aporta.

- **ComienzoSesion:** Cuando un sujeto comienza una sesión se hace uso de esta función. Con la ID del sujeto y del plan actual, se actualiza el registro de la tabla Sujeto_Plan para escribir el inicio del plan, si fuese la primera sesión, y se crea un nuevo registro en la tabla Sujeto_Sesion en el que se escribirán los resultados de la sesión que comenzará.

- **ActualizaSesionTarea1:** Esta función actualiza los valores de la primera tarea de la sesión: el nivel alcanzado, puntos y si ha sido superada. Para ello, utilizando los valores obtenidos en la prueba y el ID de la sesión actual, se realiza una actualización de ese registro en la tabla Sujeto_Sesion.

- **ActualizaSesionTarea2:** Realiza los mismos pasos que ActualizaSesionTarea1, pero con la segunda tarea.

- **ActualizaSesionTarea3:** Realiza los mismos pasos que ActualizaSesionTarea1, pero con la tercera tarea.

- **FinalizarPlan:** Cuando un plan ha acabado correctamente, el sujeto ha superado todas las sesiones (en uno o más intentos), el terapeuta puede finalizar correctamente el plan, sin tener que cancelarlo, para que el sujeto quede libre de plan y así poderle asignar uno nuevo. Cuando se usa esta función, el registro de plan activo actual de la tabla Sujeto_Plan de ese sujeto se actualiza con el valor de fin. Para esto solo es necesario el ID del Sujeto_Plan y del sujeto y no devuelve ningún resultado.

- **ResumenSesion:** Una vez finalizada la sesión, se muestra en pantalla un resumen de la misma en el que aparecen los valores de puntuación, tareas y puntos, además de las que han sido superadas. Para esta consulta es necesario el ID del sujeto_sesion que se ha ejecutado.

- **Planes:** Esta página contiene todas las funciones que manipulan la información contenida en los registros de la tabla Plan de la base de datos. Serán explicadas todas las funciones, pero en detalle se verá la que da de alta un nuevo registro, la función *Modificar*, cuyo diagrama de secuencia se puede observar en la Figura 4.8.

  - **Nuevo:** Esta función es usada para crear un nuevo registro en la tabla Plan. De esta manera, introducimos un nuevo plan para poder agregarle sesiones y ser asignado posteriormente a los diversos sujetos de la aplicación. Para la creación se debe introducir, al menos, el nombre del plan. El campo de fecha de creación indicará la fecha de creación y la validez será "NO" por defecto, valor que deberá cambiarse en una modificación posterior.

- **Modificación:** Esta función tiene asociado, para una explicación más detallada, un diagrama de secuencia que se puede observar en la Figura 4.8. Como se puede observar, el terapeuta comienza el proceso y a través de la interfaz llamará al formulario para mostrarlo en pantalla. Una vez está en pantalla el formulario, se realiza una consulta de la información actual del plan a modificar. Para ello se consultan registros de las tablas, Plan y Sesión, puesto que también mostrará los datos de las sesiones que componen dicho plan. Una vez validados los nuevos los campos que se van a modificar, la comprobación se realiza del lado del cliente, se envía la petición de medicación al archivo PHP correspondiente que, a su vez, requiere de una conexión a la tabla Plan de la base de datos. Realizada la modificación, el servidor MySQL envía la respuesta al PHP y éste a la interfaz de usuario, mostrando por pantalla el resultado y cerrando el formulario de modificación.

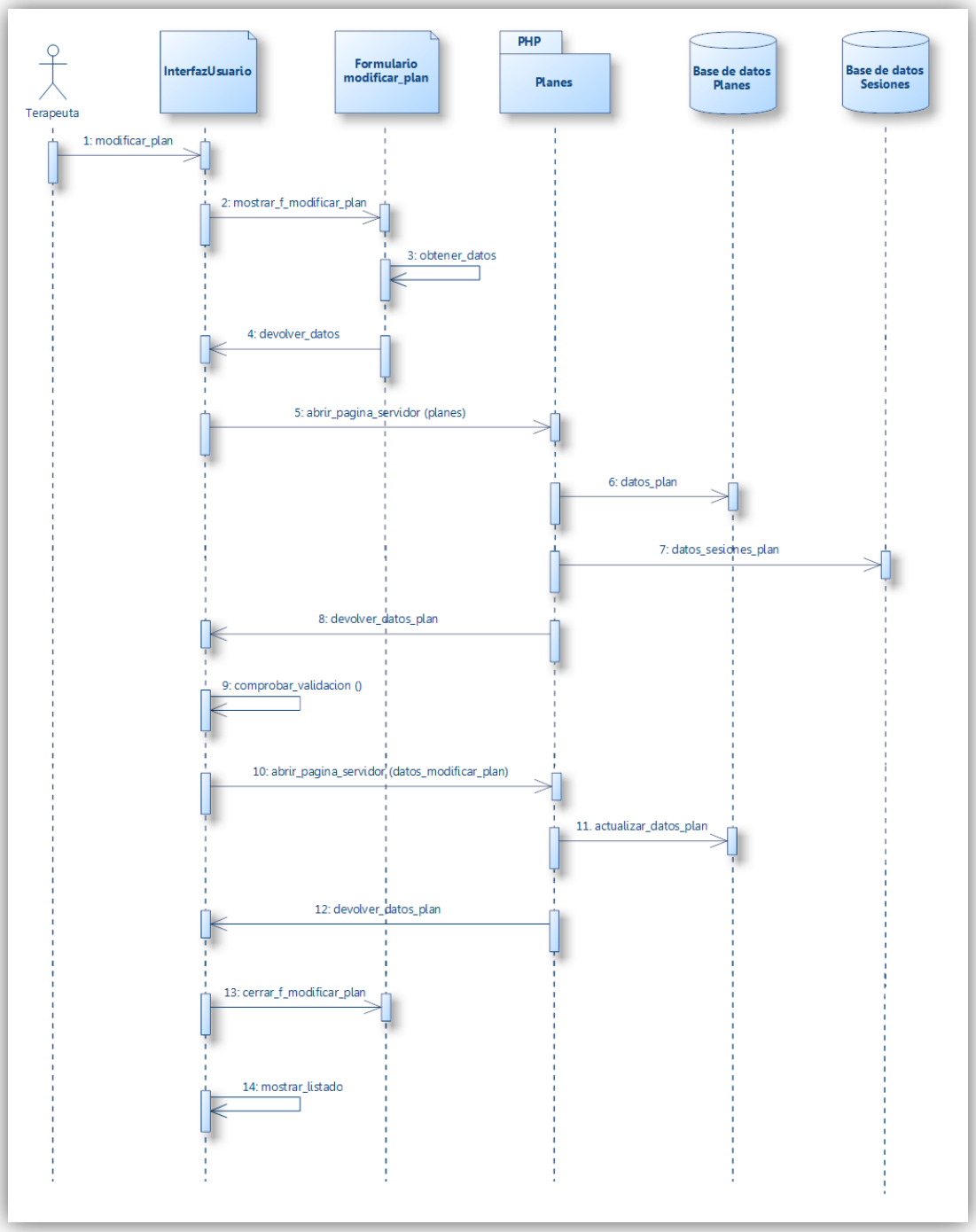


Figura 4.8 – Diagrama de secuencia de la función que modifica un plan.

- **Listar:** Para esta función no se requiere ningún atributo, siendo el resultado una lista de todos los elementos de la tabla Plan. Esta función la usa el formulario de los planes para mostrar una tabla con información sobre ellos.

- **Eliminar:** Esta función elimina un plan de la base de datos. Para ello sólo necesita el ID del plan a eliminar, devolviendo un mensaje de éxito o error según proceda. El error que se puede dar en este caso es que el plan tenga relaciones creadas con algún sujeto. Si dicho plan hubiese sido asignado previamente a uno o más sujetos ya no se podría eliminar, evitando así que hubiese datos de éste en otras tablas y el registro en planes eliminado. Esto sucede de la misma manera que con los terapeutas y sujetos en su función de eliminar.

- **ListarPlanesValidados:** Esta función se usa en el formulario de los sujetos para rellenar el listado de planes en el *combobox* de planes a asignar. Como sólo se pueden asignar planes previamente validados a sujetos, es necesario realizar una búsqueda en la tabla Plan similar al listado completo pero filtrando por aquellos planes que tienen el atributo Validado como "SI".

- **Sesiones:** Este modelo contiene todas las funciones que manipulan la información contenida en los registros de la tabla sesiones de la base de datos. Serán explicadas todas las funciones pero en detalle se verá la que da de alta un nuevo registro, la función *Modificar*, cuyo diagrama de secuencia se puede observar en la Figura 4.9.

  - **Nuevo:** Esta función crea una nueva sesión y la almacena de forma correcta en la base de datos, en la tabla de Sesión. El diagrama de secuencia completo se puede observar en la Figura 4.9. Como se puede observar, el terapeuta inicia el proceso abriendo el formulario de nueva sesión y éste, una vez en pantalla, consulta las tareas y su información en la base de datos para rellenar los componentes de selección de tareas. Una vez rellenos los datos que se deseen de la sesión, la interfaz comprueba los datos y abre la página PHP

enviando los datos que se enviarán a la base de datos para almacenar el nuevo registro. Como resultado a esta entrada de datos, se obtiene un listado actualizado de sesiones. Una vez recibido dicho listado en la interfaz, se cerrará el formulario de entrada de datos mostrando en pantalla el listado.

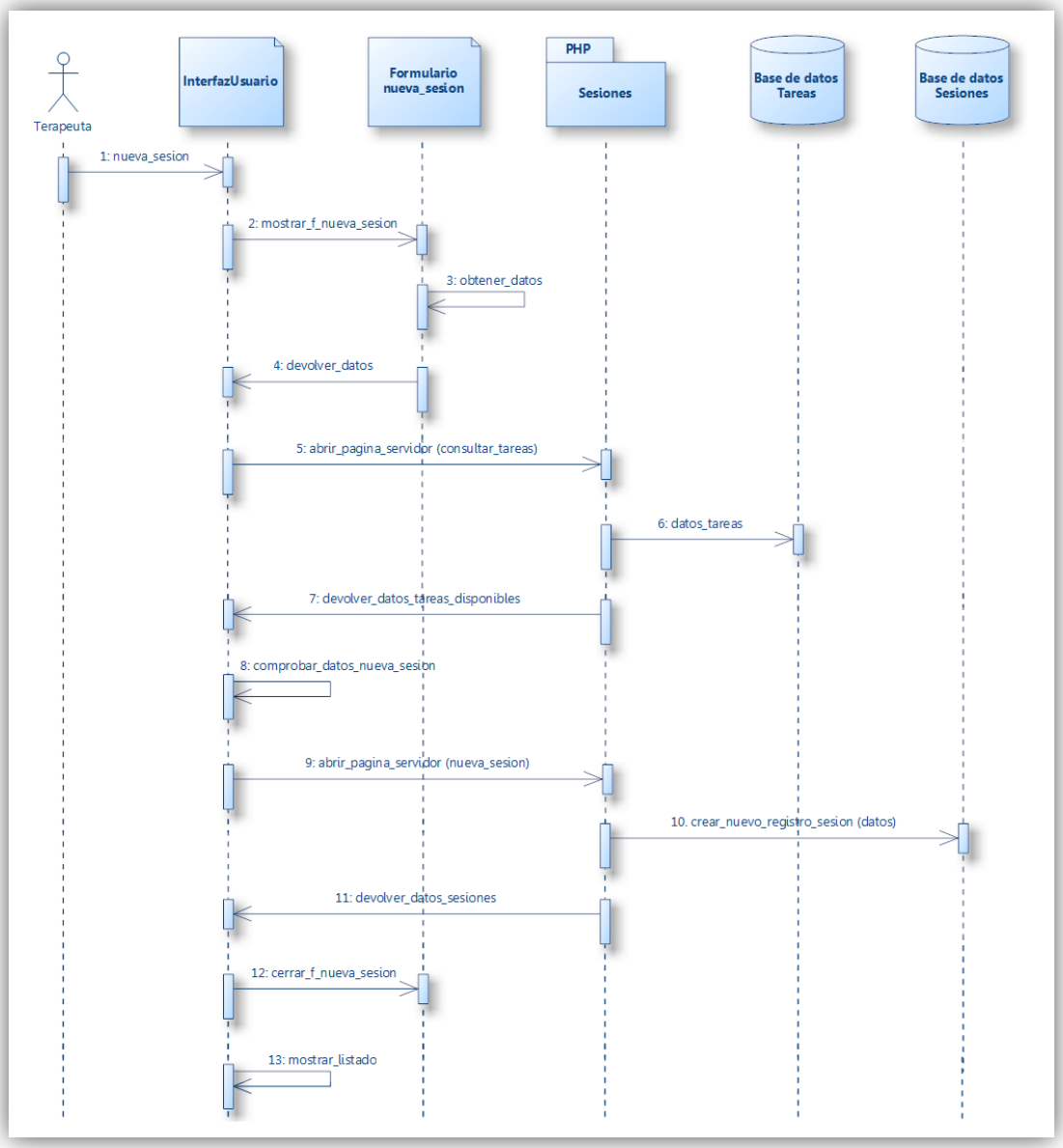


Figura 4.9 – Diagrama de secuencia de la función que crea una nueva sesión.

- **Modificar:** Esta función se utiliza para modificar un registro de la tabla Sesión de la base de datos. Para realizar dicha modificación de forma completa, es decir, todos los campos del registro, el plan debe ser modificable. Si un plan no es modificable, es que ha sido asignado al menos una vez a algún sujeto. Salvo en el campo observaciones, el resto de atributos no son modificables. Si el plan se puede modificar, todo será modificable sin restricciones.

- **Listar:** Para esta función no se requiere ningún atributo y el resultado es una lista de todos los elementos de la tabla Sesión, junto con todos sus atributos. Esta función la usa el formulario de las sesiones para mostrar una tabla con información sobre ellos. Esta función se utiliza para mostrar la información sobre todos los planes dentro del formulario asociado.

- **ListarSesionesPlan:** Esta función devuelve todas las sesiones que pertenecen a un plan. En esencia, realiza el mismo tipo de consulta que "Listar" pero filtrando por el ID del plan a consultar y ordenando la lista por el número de orden dentro del plan. Esta función se utiliza para mostrar la información sobre las sesiones de un plan en el formulario de planes.

- **Eliminar:** Esta función elimina una sesión de la base de datos. Para ello solo necesita el ID de la sesión a eliminar, devolviendo un mensaje de éxito o error según proceda. El error que se puede dar en este caso es que la sesión tenga relaciones creadas con algún sujeto. Si dicha sesión hubiese sido asignada previamente a uno o más sujetos, ya no se podría eliminar, evitando así que hubiese datos de ésta en otras tablas y el registro en sesiones eliminado. Esto sucede de la misma manera que con los terapeutas y sujetos en su función de eliminar.

- **Tareas:** Este modelo contiene todas las funciones que manipulan la información contenida en los registros de la tabla Tareas de la base de datos. Serán explicadas todas las funciones que componen esta página una a una. A diferencia del resto de modelos que contienen funciones de modificación y consulta sobre algunas tablas de la base de datos, en esta sólo hay funciones de consulta. Las tareas que existen son diez y no pueden ser modificadas por ningún usuario.

  - **ListaCompleta:** Esta función nos devolverá un listado completo de toda la información disponible en la tabla Tarea. No se realiza ningún tipo de filtrado de la información sobre la tabla Tarea.

  - **ListaCombobox:** Esta función devuelve un listado del identificador, atributo idTarea, y el nombre de las tareas de la tabla Tarea. Esta función se utiliza para mostrar en un componente de formulario el listado completo de tareas para que el terapeuta sea capaz de seleccionar aquellas que desee para sus sesiones.

  - **TareaInformacion:** Con esta función se realiza una consulta de los campos nombreTarea y descripcionTarea dentro de la tabla Tarea de la base de datos. Se usa esta función al comienzo de cada ejecución de una tarea, mostrando en pantalla el nombre y la descripción para que el sujeto conozca el funcionamiento de dicha tarea.

4.3. Diseño de la base de datos.

La base de datos utilizada en la aplicación web e implementada en MySQL consta de siete tablas relacionadas. Para mostrar en qué forma están relacionadas se ha realizado un diagrama de la estructura de la base de datos, se puede observar en la Figura 4.10.

En el diagrama, cada una de los siete rectángulos representa una tabla y las flechas las relaciones entre las mismas. El nombre de cada tabla aparece en la cabecera del rectángulo y en el cuerpo el listado de atributos. Los símbolos que preceden a cada atributo representan:

-> Clave primaria

+ -> Clave externa

- -> Atributo descriptivo

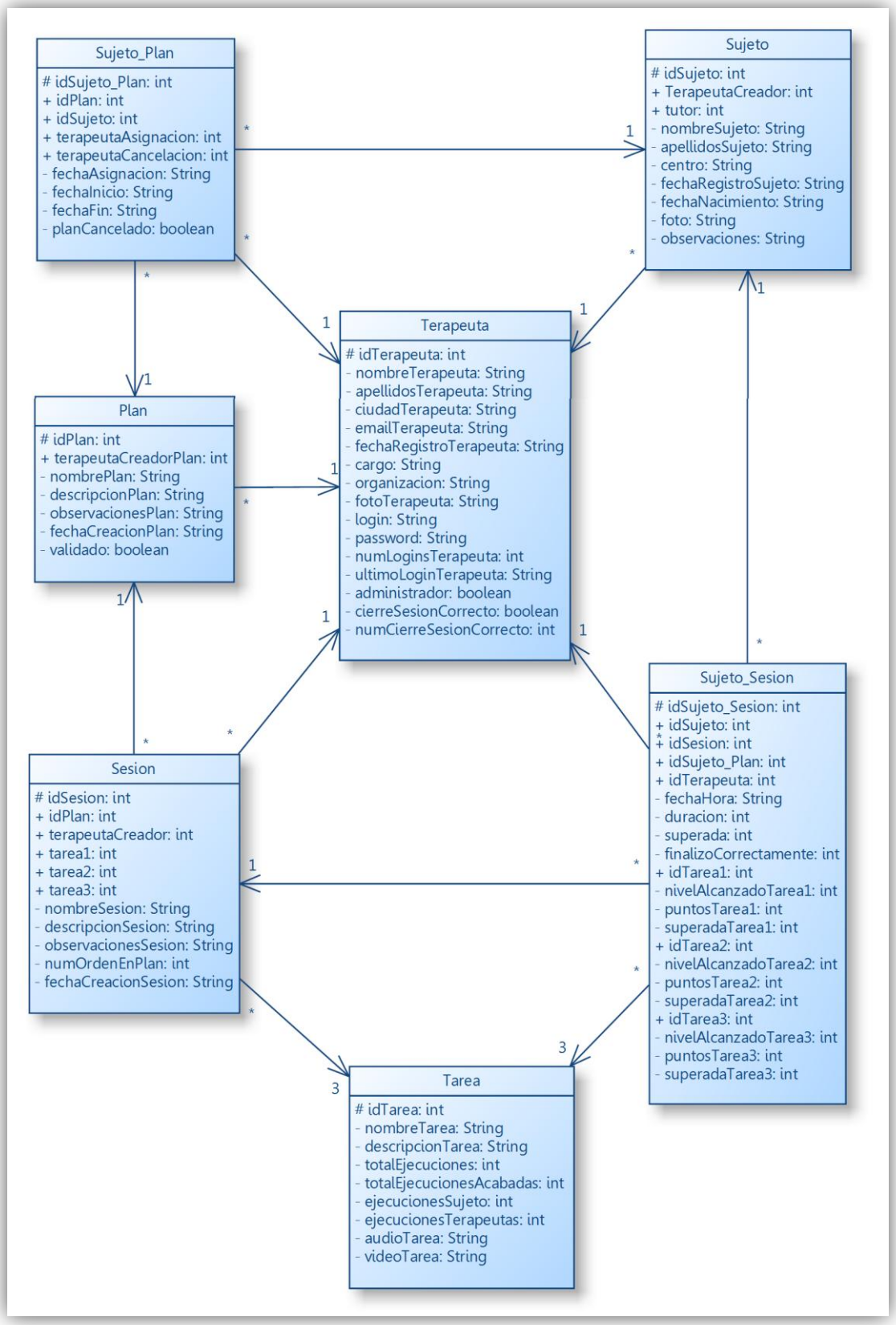


Figura 4.10 – Diagrama estructural de la base de datos.

En base al diagrama presentado en la Figura 4.10, se realizará una descripción de las tablas y sus atributos.

- **Terapeuta:** Es una de las tablas principales. Describe y almacena a los usuarios que van a hacer uso de la herramienta web y, además, está relacionada con cinco de las seis tablas restantes. Todos los cambios que se realicen en el resto de tablas llevan asociados el identificador del usuario que realizó estos cambios, por ello este identificador está presente en la mayor parte de las tablas de la base de datos. Además de contener la información de la persona, también contiene la información del usuario del sistema, como el nombre de usuario (login) y la contraseña.

- **Sujeto:** Esta tabla contiene la información referente a los sujetos que van a realizar los entrenamientos atencionales en la herramienta. Por este motivo, en dicha tabla se pueden establecer hasta dos relaciones con la tabla Terapeuta, de forma que una es para identificar al que registró al sujeto en la base de datos y la otra para conocer al responsable de su entrenamiento. Actualmente, ambos campos son en el mismo, pero se puede ampliar de forma que sean dos terapeutas distintos.

- **Tarea:** En esta tabla se guarda toda la información referente al identificador, nombre y descripción de cada tarea, así como determinadas estadísticas y la inclusión de rutas para un audio o video de apoyo al terapeuta. Algunas de estas características se han incluido para facilitar una ampliación futura de la herramienta.

- **Plan:** Aquí serán almacenados todos los planes de entrenamiento que se planifiquen. Esta tabla contiene el identificador del terapeuta que creó el plan junto con algunos campos descriptivos. En especial hay que nombrar el campo "validado", pues es el que indicará si el plan está disponible para ser asignado a sujetos, o por el contrario, no lo está. Este campo será modificable aún estando ya asignado, haciendo de esta manera que deje de estar seleccionable.

- **Sesion:** Puesto que un plan contiene muchas sesiones, en esta tabla almacenaremos tantas sesiones distintas como se hayan creado para los planes. No existe un límite establecido de sesiones por cada plan, de manera que un plan puede ser tan extenso como el terapeuta crea conveniente. Como una sesión es creada por un terapeuta para ser asignada a un plan y ésta a su vez debe estar compuesta de tres tareas, está relacionada con esas tres tablas. En especial, un registro de esta tabla contiene tres campos identificativos de la tabla tarea, uno por cada tarea de las tres que componen una sesión.

- **Sujeto_Plan:** Para establecer una relación fuerte entre un sujeto y un plan necesitamos una tabla intermedia. Esto es así porque un mismo sujeto puede haber realizado varios planes y un mismo plan puede haber sido realizado por varios sujetos. Por lo tanto, cada vez que asignamos un plan nuevo a un sujeto, se crea un nuevo registro en esta tabla. De este registro sacaremos información sobre el plan al que se ha asignado el sujeto, el terapeuta que ha hecho dicha asignación, si el plan fue cancelado, el terapeuta que canceló el plan, sujeto al que pertenece y las fechas de asignación, inicio y fin.

- **Sujeto_Sesion:** Al igual que ocurre con los planes y los sujetos, con los sujetos y las sesiones hay una relación de muchos a muchos, es decir, cada sujeto puede realizar varias sesiones y una sesión puede ser realizada por muchos sujetos. En este caso tenemos una de las tablas que más información va a aportar a los terapeutas, dado el tipo y el número de campos que la componen. Esta tabla está relacionada con el sujeto que va a realizar la sesión, el terapeuta que va a supervisarla, la sesión a realizar y las tareas que van a participar. En este caso existe una duplicidad de datos intencionada, ya que los tres identificativos de las tareas están tanto en la tabla que describe la sesión como en la tabla que describe la sesión por sujeto. Esta decisión de diseño se llevó a cabo para facilitar la búsqueda de datos estadísticos exclusivamente en esta tabla y no tener que hacer consultas con tablas cruzadas. Como se puede observar, además de

los datos anteriores también se almacena la duración, fecha y hora, si fue superada y los puntos y niveles alcanzados de cada tarea.

Capítulo 5 – Aspectos destacados de la implementación

En este capítulo se describen las partes más importantes de la aplicación (desde el punto de vista de la implementación), centrándonos en la estructura de la interfaz y en cómo se ha implementado la interacción con el usuario. Se detallan todos los recursos utilizados y la forma en que se han usado dichos recursos para hallar diversas soluciones mediante *ActionScript 2*, lenguaje de programación de *Flash*. También se realiza una descripción de cada tarea y la forma en la que ha sido implementada.

5.1. Estructura de la aplicación.

La aplicación consta de tres partes diferenciadas de recursos: (1) el archivo Flash, que contiene la interfaz de usuario y que se descarga y ejecuta directamente en el navegador web del cliente; (2) los archivos PHP, que serán requeridos para conectarse con la base de datos y que serán ejecutados en el lado del servidor; y, por último, (3) diversos recursos multimedia como imágenes o animaciones, que serán

requeridos por el archivo Flash para ser mostrados en las diversas tareas implementadas.

El archivo *Flash* será descrito ampliamente en el siguiente apartado y en él se detallan las partes en las que se divide la interfaz y cómo está estructurada dentro de *Flash.*

Los archivos PHP ya han sido descritos con anterioridad en el apartado 4.3 junto con los diagramas de secuencia. No obstante, se hará referencia a ellos y se mostrarán ejemplos de código en la descripción de detalles de programación de los formularios.

Las imágenes y animaciones externas a *Flash* están localizadas en diversos directorios situados en el servidor donde se aloja la aplicación. Generalmente, pertenecen a tareas y se ha decidido colocarlos de forma externa ala aplicación *Flash* por dos motivos: (1) el primero es para hacer un archivo swf (*Flash*) más pequeño en tamaño, así el cliente podrá manejarlo con mayor soltura y sin necesidad de esperar un mayor tiempo de carga; (2) y la segunda, para poder manipular las imágenes de forma más sencilla, sin necesidad de recompilar de nuevo el código de la aplicación principal si se desean modificar los estímulos. A pesar de que la segunda razón puede parecer un problema añadido, ya que desde la aplicación se deben cargar imágenes externas, se ha reducido al máximo el tamaño de estos archivos para favorecer una carga rápida y sin saltos.

5.2. Diseño e implementación de la interfaz de usuario en Flash.

En *Flash* se puede desarrollar cualquier tipo de aplicación o animación, desde pequeños cortos de animación a aplicaciones webs profesionales con información dinámica. El esquema de desarrollo en *Flash* se fundamenta en el concepto de línea temporal y, en el uso de símbolos. Además, tiene su propio lenguaje de programación, *ActionScript*, que permite modificar y adaptar todo el contenido a aquello que se desee obtener.

La línea temporal representa el paso del tiempo en todo desarrollo *Flash.* Se compone de fotogramas donde, en cada fotograma, se pueden alojar símbolos y

código, dentro o fuera de éstos símbolos. Además, todo proyecto *Flash* tiene como propiedad una velocidad dada en fotogramas por segundo. De esta manera, conseguimos desplazarnos a lo largo de la línea temporal para crear el efecto de movimiento que deseamos. Mediante código se controla la línea temporal, pudiendo hacer que ésta se detenga en cualquier momento. De esta manera, se puede realizar una aplicación con distintas ventanas repartidas a lo largo de la línea temporal y saltar de una a otra mediante elementos con los que el usuario interactúa.

A su vez, para estructurar cada fotograma, se puede hacer una división por capas para disponer los elementos dentro del fotograma, haciendo que unos estén superpuestos con otros en función de la profundidad de dicha capa. De esta manera es como se ha dividido el espacio dentro de cada ventana.

Un símbolo es cualquier elemento gráfico que esté dentro de una película *Flash*. A su vez, se pueden dividir en tres tipos según su comportamiento:

- **Botones:** Son elementos gráficos que ofrecen interacción con el usuario. Pueden tener hasta tres estados distintos según estén en reposo, el ratón encima o el ratón presionándolo, además de establecer una zona para su activación.

- **Gráficos:** Son todos aquellos símbolos que representan elementos gráficos estáticos. Un ejemplo de este tipo de elementos sería un círculo o una línea estática.

- **MovieClips:** A diferencia de los gráficos, éstos representan elementos gráficos dinámicos y tienen su propia línea temporal independiente de la principal. Un MovieClip puede contener otros MovieClips, gráficos y botones.

Una vez conocidos los elementos y la forma de colocarlos a lo largo de la línea temporal se verá la forma en la que se ha dispuesto todo para el desarrollo de este prototipo.

En este proyecto, transversalmente, es decir, a lo largo de la línea temporal, se han dispuesto diversas ventanas para mostrar la barra de carga, el formulario de inicio

de sesión (Figura 5.1), otros formularios y cada una de las tareas; y longitudinalmente, las capas, se han dividido en pantallas informativas y de error, pie, cuerpo, cabecera, menús y fondo.

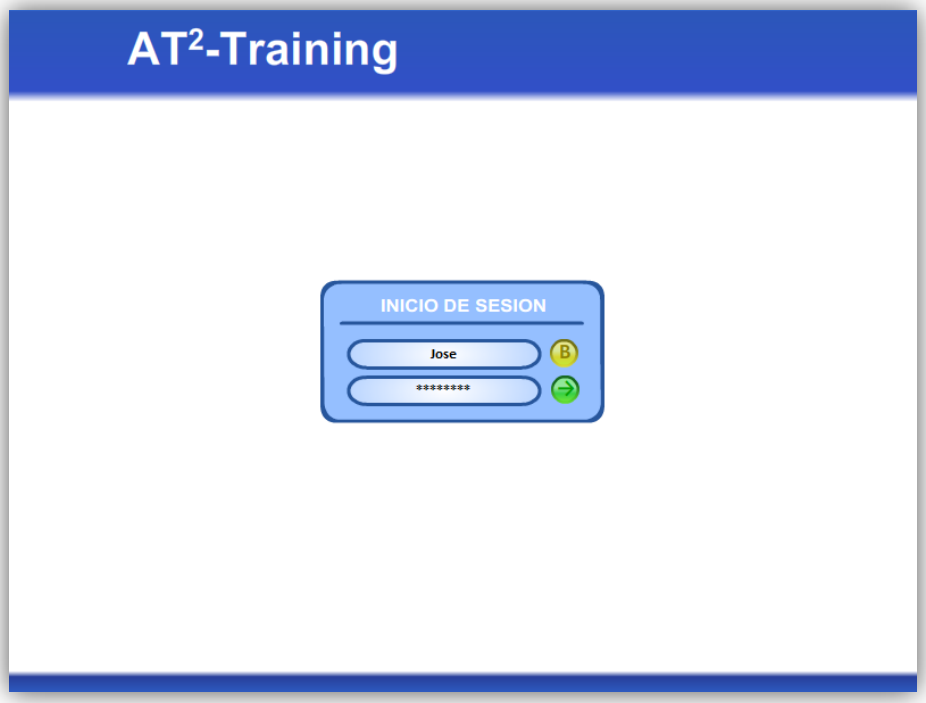


Figura 5.1 – Pantalla con el formulario de inicio de sesión.

Una vez se ha accedido a la aplicación web, tras introducir correctamente un nombre de usuario y su correspondiente contraseña en el formulario de inicio de sesión (Figura 5.1), se muestra la parte de gestión de la aplicación, la cual tiene una estructura muy definida y uniforme.

Como se puede observar en la siguiente figura (Figura 5.2), la parte de gestión está dividida en tres zonas con las que el usuario puede interactuar:

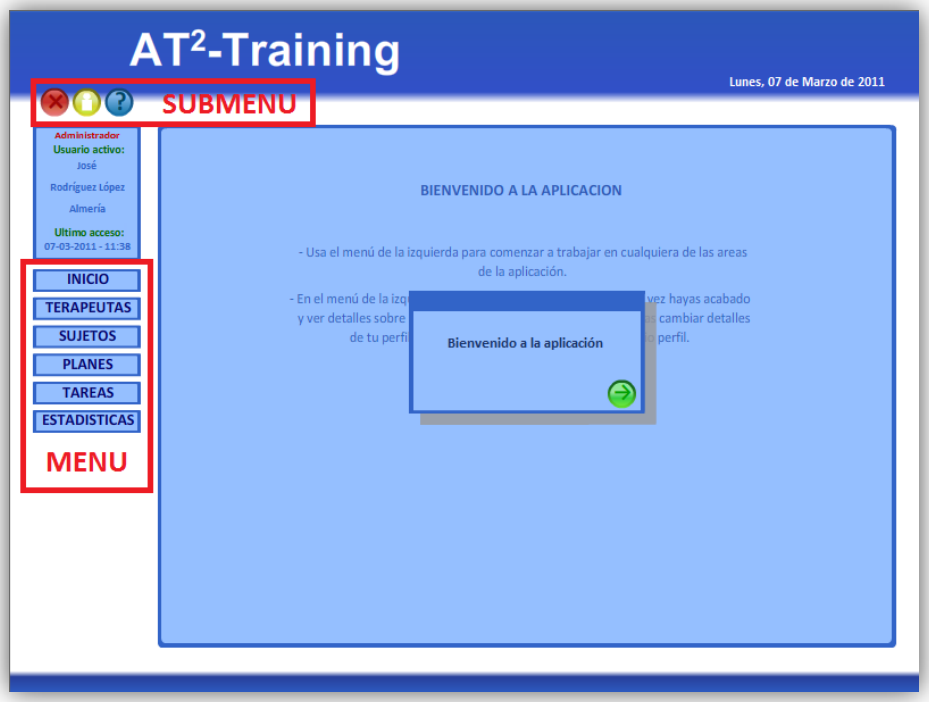


Figura 5.2 – División en secciones de la aplicación.

- **Submenú:** Situado en la parte superior de la aplicación, contiene tres botones que son utilizados para cerrar la sesión actual, ver el perfil del usuario que ha accedido al sistema y acceder al pdf de ayuda.

- **Menú:** Está situado en la parte izquierda y da acceso a las diferentes partes de la aplicación.

- **Contenido:** Es la zona central de la aplicación, enmarcada en un recuadro azul. En ella irán apareciendo los distintos formularios u opciones según el usuario vaya interactuando con el sistema.

Estas partes en las que está dividida la estructura interactiva de la aplicación no son las únicas. La cabecera, el pie, el fondo y las ventanas, ocultas, también forman parte de la aplicación pero no son interactivas. Todas estas partes, interactivas o no, forman las capas en las que se distribuye *Flash* para organizar la programación de la aplicación.

Esta estructura varía a la hora de realizar una sesión con un sujeto o cuando se ejecutan las diferentes tareas.

Cuando se muestra la información de la tarea que va a ser ejecutada así como el resumen de la sesión ejecutada la pantalla sólo muestra esta información. Esto se realiza de esta manera porque en esos momentos el sujeto, persona de corta edad, puede estar solo delante de la aplicación y toda la parte de gestión debe estar oculta para él. En la siguiente imagen (Figura 5.3) se puede observar un ejemplo de la pantalla informativa sobre una tarea.

Figura 5.3 – Información previa a la ejecución de una tarea.

Durante la ejecución de una tarea, cualquiera, la pantalla se vuelve a dividir en tres zonas bien diferenciadas y distintas al resto pantallas anteriormente descritas. Una tarea tiene una cabecera, en la que podemos ver el nombre de la tarea, nivel actual, tiempos y ensayo actual; un pie, donde se sitúan los botones de salir (oculto en la ejecución de una sesión), pausa y puntuación; y por último, la zona central, dónde se ejecuta la tarea en sí. Podemos observar esta división en la Figura 5.4, en la que se muestra la ejecución de una de las diez tareas realizadas.

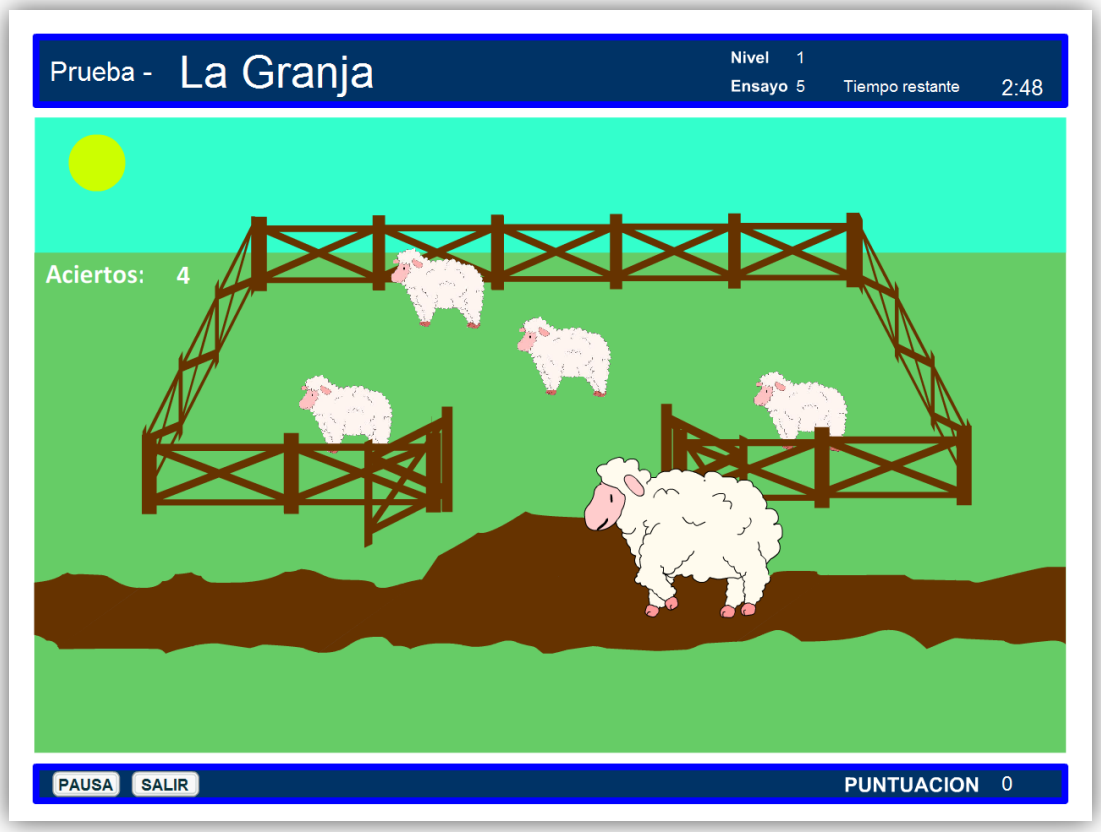


Figura 5.4 – Ejemplo de ejecución de una de las tareas.

5.2.1 Diseño de los formularios.

Para implementar todos los formularios se han utilizado los componentes prediseñados de *Flash* tales como botones, áreas de texto, introducción de datos, tablas de datos, campos de selección, etc. así como otros elementos gráficos diseñados para completar una visión más atractiva de la aplicación.

Los componentes prediseñados resultan extremadamente útiles para el manejo y muestreo de la información además de ser fácilmente adaptables y configurables en forma, aspecto y capacidad de gestión de los datos que tratan.

En la siguiente imagen se observa cómo están dispuestos los diferentes componentes en uno de los formularios (Figura 5.5).

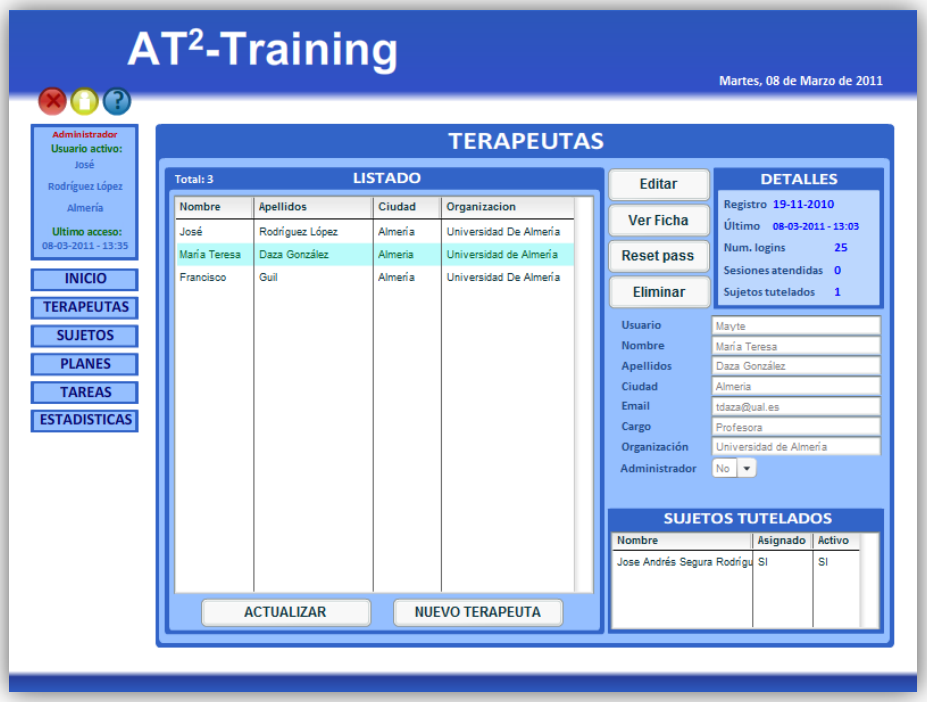


Figura 5.5 – Vista de ejemplo de unos de los formularios.

A continuación se detallan algunos de los componentes utilizados:

- **Button:** Es un botón común compilado por *Flash* y editable. A diferencia de otros botones exclusivamente realizados para esta aplicación, los instanciados por esta clase compilada tienen un aspecto común y son los que se han utilizado dentro de los formularios.

- **ComboBox:** Esta clase representa los selectores de opciones desplegables. Llevan asociado un *array* de datos para introducir los valores que almacenen, sean mostrados o no. Por ejemplo, en el formulario de los sujetos, tenemos un *ComboBox* para seleccionar el Plan que queremos asignar al sujeto; para ello se realiza una consulta a la base de datos y se le asigna un *array* de dos columnas, una con el ID del plan, que corresponde a los datos, y otra el nombre del plan, que

corresponde a la etiqueta. Así, al seleccionar el plan por su nombre, de forma oculta también estamos vinculando el ID, que será necesario para establecer correctamente la relación.

- **TextInput:** Este componente representa un campo de texto y es utilizado para introducir y mostrar en pantalla el texto de los distintos campos de información. Gracias a él introducimos los datos en la base de datos y los mostramos en pantalla para ser editados. Para mostrar información no editable se ha utilizado la herramienta de texto asociándola a los diferentes atributos, no este componente.

- **DataGrid:** Esta es la clase que representa las tablas de datos y al igual que el componente *ComboBox* también lleva asociado un *array* de datos pero con la posibilidad de mostrar más columnas y poder reordenarse dinámicamente.

Como se detallará en el siguiente apartado, el comportamiento de los formularios están programados mediante el código *actionScript*, por ejemplo, para ocultar determinados componentes según el paso en el que se encuentre el usuario o incluso, según sea el tipo de usuario. Los administradores tienen más posibilidades de edición de registros en el formulario de administración de terapeutas.

5.2.2. Implementación de los formularios.

Toda la programación se realiza en el lenguaje específico de *Flash*, es decir, *actionScript*. Cada formulario tiene el código asociado en el fotograma en el que se encuentra, de forma que al pasar a ese fotograma en concreto, se ejecuta el código correspondiente.

La estructura básica del código en estos fotogramas que se ha seguido es la siguiente: declaración de variables, funciones de conexión con la base de datos, eventos asociados a los componentes, funciones que realizan esos eventos y, por último, personalización de los componentes y llamada a las funciones que inicializan la ejecución del formulario.

En la parte de la declaración de variables, como se puede observar en la Figura 5.6, también se colocan determinados atributos de objetos en los valores óptimos para ese formulario. Lo que se observa en la Figura 5.6 es la configuración de la posición de las ventanas de información y de error, así como la propiedad de visibilidad de éstas asignada a falso, con el objetivo de que no sean visibles. Además, se puede ver como se declara una variable que será la encargada de almacenar la acción a realizar al hacer clic en el botón de aceptar.

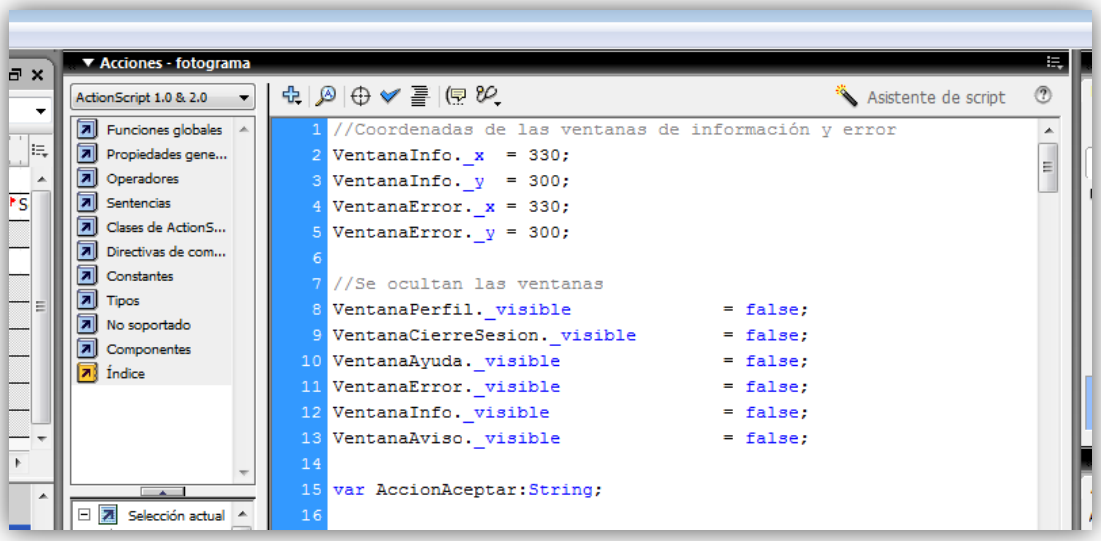


Figura 5.6 – Parte inicial del código de los formularios.

La parte de la conexión con la base de datos, mediante el uso de la clase LoadVars y los archivos PHP, por su importancia, serán explicados en el siguiente apartado, el 5.2.3. Comunicación con la base de datos.

La parte en la que se hace referencia a la declaración de eventos para la gestión de éstos se puede ver en la Figura 5.7. Con estas líneas se describe qué función se debe ejecutar cuando se hace clic en cada uno de esos componentes. Esas diez funciones que se ven en la figura son descritas en la siguiente parte del código.

```
197 BTN_NuevoTerapeuta.addEventListener("click", Click_NuevoTerapeuta);
198 BTN_EditarTerapeuta.addEventListener("click", Click_EditarTerapeuta);
199 BTN_EliminarTerapeuta.addEventListener("click", Click_EliminarTerapeuta);
200 BTN_CancelarEditar.addEventListener("click", Click_CancelarEditar);
201 BTN_GuardarEditar.addEventListener("click", Click_GuardarEditar);
202 BTN_ActualizarTerapeuta.addEventListener("click", Click_ActualizarGridTerapeutas);
203 BTN_RestablecerPassTerapeuta.addEventListener("click", Click_RestablecerPassTerapeuta);
204 BTN_CambiarPassTerapeuta.addEventListener("click", Click_CambiarPassTerapeuta);
205 BTN_VerFichaTerapeuta.addEventListener("click", Click_VerFichaTerapeuta);
206 GridTerapeutas.addEventListener("change", Click_SeleccionGrid);
```

Figura 5.7 – Parte de la gestión de eventos del código de los formularios.

La siguiente parte consiste en, declarar todas las funciones que se añaden como eventos de ratón. En el ejemplo que hemos tomado en la Figura 5.8 se describe la función que se ejecuta cuando se hace clic en el botón de Nuevo Terapeuta. Como se puede observar, se habilitan o deshabilitan los componentes según se vaya a hacer uso de ellos o no. Ahí entran todos, ya sean campos de texto, botones o tablas de datos. De esta manera, se establece que sólo se vea en pantalla aquello con lo que el usuario debe interaccionar, evitando lanzar determinados mensajes de error al usuario, no dándole la posibilidad de realizar determinadas acciones. Por ejemplo, hacer clic en el botón de Eliminar cuando se está introduciendo un nuevo terapeuta. Ocultando el botón de Eliminar, se evita tener que mostrar ese posible mensaje de error, ya que nunca se va a poder hacer clic cuando se está introduciendo datos para uno nuevo. A la variable "*AccionAceptar*" se le asigna el valor de "*Nuevo*", para que cuando se haga clic en el botón de aceptar, el programa lance la función que permite enviar los datos a la base de datos.

```
277 function Click_NuevoTerapeuta()
278 {
279     Click_CancelarEditar();
280     MostrarCamposTerapeuta();
281 
282     AccionAceptar                = "Nuevo";
283     GridTerapeutas.enabled       = false;
284     GridSujetosTutelados.enabled = false;
285     TextoNuevoTerapeuta.text     = "NUEVO TERAPEUTA";
286 
287     OutUser.enabled              = true;
288     OutUser.editable             = true;
289     OutNombre.enabled            = true;
290     OutNombre.editable           = true;
291     OutApellidos.enabled         = true;
292     OutApellidos.editable        = true;
293     OutCiudad.enabled            = true;
294     OutCiudad.editable           = true;
295     OutEmail.enabled             = true;
296     OutEmail.editable            = true;
297     OutCargo.enabled             = true;
298     OutCargo.editable            = true;
299     OutOrganizacion.enabled      = true;
300     OutOrganizacion.editable     = true;
301     OutAdmin.enabled             = true;
302 
303     BTN_GuardarEditar.visible            = true;
304     BTN_CancelarEditar.visible           = true;
305     BTN_ActualizarTerapeuta.enabled      = false;
306     BTN_NuevoTerapeuta.enabled           = false;
307     BTN_EditarTerapeuta.visible          = false;
308     BTN_EliminarTerapeuta.visible        = false;
309     BTN_RestablecerPassTerapeuta.visible = false;
310     BTN_VerFichaTerapeuta.visible        = false;
311 
312     OutAdmin.selectedIndex = 0;
313 }
```

Figura 5.7 – Parte de la declaración de funciones de eventos.

La última parte en la que se divide el código de los formularios hace referencia a los atributos de los componentes. En algunos, como en la introducción de datos, se restringe la entrada de determinados caracteres como, por ejemplo, en el campo nombre, donde no se pueden introducir números, de esta manera tenemos un control mayor sobre los datos y, al igual que se hace ocultando botones, se evita tener que lanzar mensajes de error alertando de esa entrada errónea de caracteres. Se puede ver un ejemplo de esa parte del código en la Figura 5.8.

```
731 GridSujetosTutelados.editable           = false;
732 GridSujetosTutelados.resizableColumns   = false;
733 GridSujetosTutelados.headerHeight       = 15;
734 GridSujetosTutelados.showHeaders        = true;
735 GridSujetosTutelados.sortableColumns    = true;
736 GridSujetosTutelados.setStyle("color", "#DarkBlue");
737 GridSujetosTutelados.addColumn("Nombre");
738 GridSujetosTutelados.addColumn("Asignado");
739 GridSujetosTutelados.addColumn("Activo");
740 GridSujetosTutelados.getColumnAt(0).width = 125;
741 GridSujetosTutelados.getColumnAt(1).width = 53;
742 GridSujetosTutelados.getColumnAt(2).width = 40;
743
744 PHP_Terapeutas("Listar");
745
746 stop();
```

Figura 5.8 – Parte final del código de los formularios.

Observando la Figura 5.8, en el caso concreto del componente de la tabla de datos "*GridSujetosTutelados*", se establecen determinadas propiedades y estilos gráficos, además de las columnas que van a ser visibles y su anchura. Para finalizar el código de un formulario se hace una llamada a la función que se ejecuta inicialmente, en este caso *PHP_Terapeutas("Listar"),* y que en este caso lo que hará es inicializar la tabla de datos de terapeutas colocando toda la información de los terapeutas que hay en la base de datos. Por último, se coloca la instrucción *stop();* que para la ejecución de la película *Flash* para que no continúe su avance.

5.2.3. Comunicación con la base de datos.

Para establecer una comunicación válida con la base de datos, primero se debe almacenar en un objeto aquellos datos que necesitamos. La clase de *Flash Professional 8*, llamada *LoadVars*, es la que va a permitir realizar esta conexión con el archivo PHP y éste, a su vez, será el que comunique con la base de datos, transportando la información de nuevo a *Flash* mediante la misma clase. De esta manera, la clase *LoadVars* es usada de forma bidireccional, tanto para enviar datos como para recibirlos.

Para realizar todo esto, primero se deben declarar las dos variables de tipo *LoadVars* que se van a necesitar, una para enviar datos y otra para recibirlos. Una vez declaradas las variables que se necesitan, se hace un llamamiento al método "*sendAndLoad*" de la variable de envío y se colocan como parámetros de entrada la dirección web del archivo PHP que va realizar la conexión, variable que recibirá los datos y el método de envío de los datos, en este caso, POST.

En la siguiente figura, la Figura 5.9, se puede analizar ver un ejemplo de este tipo de código:

```
//**************************************************************************************************************************************************
// PHP DE SESIONES ATENDIDAS POR TERAPEUTA
//**************************************************************************************************************************************************

function PHP_SesionesAtendidas(OpcionesSesionesAtendidas:String)
{
    enviarDatosPHPSesionesAtendidas.Opcion  = OpcionesSesionesAtendidas;

    enviarDatosPHPSesionesAtendidas.ID      = DatosTerapeuta.ID;

    enviarDatosPHPSesionesAtendidas.sendAndLoad(Servidor + "CodigoPHP/Terapeutas.php?"+Math.random(), recibePHP_SesionesAtendidas, "POST");
    trace("Opcion Sesiones atendidas de ida: "+OpcionesSesionesAtendidas);
}

recibePHP_SesionesAtendidas.onLoad = function(exito)
{
   if (exito)
   {
        trace("Opcion  Sesiones atendidas de vuelta: "+ this.OpcionesTerapeuta);

        switch (this.OpcionesTerapeuta)
        {
            case "SesionesAtendidas":
                    OutSesionesAtendidas.text = this.NumSesionesAtendidas;
                break;

            default:
                    VentanaError.MensajeError.text = "Error desconocido al recibir la información de las sesiones atendidas.";
                    VentanaError._visible = true;
                    Click_CancelarEditar();
                break;
        }

   }
        else
        {
            VentanaError.MensajeError.text = "Error al enviar la informacion, no se puede acceder al archivo Terapeutas.php.";
            VentanaError._visible = true;
        }
} // FIN recibePHP_Terapeutas
```

Figura 5.9 – Código de envío y recepción de datos con la clase LoadVars.

Como se puede observar en la figura anterior, se ha separado todo en dos funciones: la primera es llamada para enviar datos al archivo PHP y es la que se encarga de llamar al método "sendAndLoad", con sus correspondientes parámetros; y la segunda es la que se encarga de realizar el tratamiento de los datos al recibirlos. En este caso se envían dos variables, la Opción, que nos llevará al caso correspondiente

dentro del archivo PHP; y el ID del terapeuta del que se va a realizar la consulta SQL que se enviará a la base de datos.

En las siguientes figuras se describirán como están estructurados los archivos PHP que contienen las funciones.

En la primera figura, Figura 5.10, se ve como comienza el código PHP del archivo "*Terapeutas.php*". Primero se realiza la conexión a la base de datos y la lectura de un array de mensajes; son otros archivos PHP que se importan como soporte. Posteriormente se realiza una lectura *POST* de las variables enviadas desde *Flash*. Dependiendo de la función muchas estarán vacías porque no serán usadas. Para continuar se ejecuta un "*switch*" con la variable "Opcion", la cual irá directamente al caso que se requiere.

```
<title>Funciones de terapeuta</title>
<?PHP

    // Conexión con la base de datos
    include "Conexion.php";
    include "MensajesServidor.php";

    $Opcion         = $_POST["Opcion"];

    $User           = $_POST["User"];
    $Pass           = $_POST["Pass"];
    $PassActual     = $_POST["PassActual"];
    $PassNueva      = $_POST["PassNueva"];
    $ID             = $_POST["ID"];

    $Nombre         = ucwords(strtolower(utf8_decode($_POST["Nombre"])));
    $Apellidos      = ucwords(strtolower(utf8_decode($_POST["Apellidos"])));
    $Ciudad         = ucwords(strtolower(utf8_decode($_POST["Ciudad"])));
    $Email          = utf8_decode($_POST["Email"]);
    $Cargo          = utf8_decode($_POST["Cargo"]);
    $Organizacion   = utf8_decode($_POST["Organizacion"]);

    $Administrador  = $_POST["Administrador"];

    switch ($Opcion)
    {
```

Figura 5.10 – Código PHP común entre los diferentes archivos de conexión.

Una vez dentro del caso del "*switch*" que vamos a ejecutar, y tal como se puede observar en la Figura 5.11, realizamos la consulta SQL y la enviamos a la base de datos mediante la conexión establecida en el paso anterior. Como la respuesta es una misma cadena en la que se introducen variables separadas por el símbolo "&", inicializamos esa cadena y la terminamos como finaliza el archivo PHP, Figura 5.12.

```php
//*****************************************************************************************************************************
//*****************************************************************************************************************************
/** Función SESIONES ATENDIDAS POR TERAPEUTA

Parámetros:
- ID del terapeuta a consultar.

Devuelve:
- El número de sesiones atendidas por un terapeuta.

Pautas a seguir:
- Consulta en la base de datos mediante el ID del terapeuta que se va a consultar
  para contar las coincidencias dentro de la tabla sujeto_sesion.

**/
case "SesionesAtendidas":

    $sql = "SELECT  COUNT(idTerapeuta) AS NumSesionesAtendidas
            FROM sujeto_sesion
            WHERE idTerapeuta = '$ID'";

    $resultados = $db -> query($sql);

    // Extracción de los datos de los terapeutas
    $fila = $resultados -> fetch_assoc();
    $respuesta .= '&NumSesionesAtendidas='.$fila["NumSesionesAtendidas"];

    break;
    // FIN SESIONES ATENDIDAS POR TERAPEUTA
```

Figura 5.11 – Código PHP correspondiente a la función SesionesAtendidas.

```php
    default:

        $CodigoServidor = 10; //"Opcion incorrecta: Terapeutas.php"

        break;
} // Fin del switch

$respuesta .=   '&OpcionesTerapeuta='.$Opcion.
                '&MensajeServidor='.$MensajeServidor[$CodigoServidor];

$db -> close();

echo utf8_encode($respuesta);

?>
```

Figura 5.12 – Código PHP correspondiente al final del archivo.

Como se puede observar en la última figura, Figura 5.12, el "switch" finaliza con el caso por defecto enviando un mensaje de error, ejecutándose si la opción enviada desde *Flash* no estuviese contemplada. También se ve que se termina de completar la respuesta, se cierra la conexión de la base de datos y se devuelve la respuesta.

5.3. Diseño de las tareas.

En total se han implementado diez tareas que serán utilizadas para el entrenamiento atencional de los sujetos. Cada una de estas tareas se ha desarrollado dentro de un fotograma en concreto. De esta manera, podemos saltar a cada una de ellas de forma independiente.

Para llevar a cabo las tareas se han utilizado algunos componentes para la carga de imágenes, botones, gráficos y animaciones con *MovieClips*.

Todas las tareas se basan en una respuesta a un estímulo visual, algunas tienen un tiempo límite y otras no. Para incluir estas imágenes se ha optado por almacenarlas de forma externa al programa, ya que incluir todas estas imágenes haría más largo el tiempo de carga de la aplicación y más compleja la edición de éstas para la etapa de pruebas del prototipo. Pese a que no era un requisito funcional, las imágenes pueden ser alteradas de forma externa para poder ajustar la dificultad de las pruebas. Como se explicará más adelante, también determinados parámetros podrán ser revisados.

Como se ha detallado en el apartado 5.2, durante la ejecución de las tareas existen tres partes visuales: la cabecera, el contenido de la tarea y el pie (Figura 5.4).

En la cabecera se muestra el nombre de la tarea, el ensayo actual, el nivel actual y el tiempo de sesión. Esta parte está compuesta por gráficos estáticos y campos de texto que muestran las variables de la tarea.

En el contenido de la tarea tendremos todo tipo de MovieClips animados con las imágenes que cargará externamente. Para la carga de imágenes se hace uso del componente “Loader”, gráficos estáticos para el decorado y fondo, y en determinados casos, botones.

El pie contiene el botón de pausa y salir y la puntuación obtenida. La puntuación varía en función del número de niveles que se superen. Cada nivel superado supondrá un punto extra.

En los siguientes apartados se describirán las partes más importantes de las tareas y el funcionamiento de cada una de ellas.

5.3.1. Diseño común de las tareas.

Al ejecutar una tarea se realiza un paso previo que es el de mostrar la información, nombre y descripción de dicha tarea. En este paso previo, que es común a todas y cada una de las tareas, se realiza una consulta a la base de datos de la misma manera que se hace para gestionar la información en los formularios.

Determinadas funciones que manejan variables comunes con todas las tareas, como pueden ser los temporizadores y las puntuaciones, han sido declaradas en el primer fotograma de la aplicación. Esto se ha realizado de esta manera porque son funciones genéricas que deben estar disponibles para toda la aplicación.

Entre estas funciones destacan:

- **Temporizadores:** Actualiza en tiempo real las variables de tiempo que se muestran en la cabecera.

- **FinTarea:** Esta compleja e importante función es ejecutada cada vez que finaliza una tarea. Es la encargada de actualizar los valores en la base de datos durante la ejecución de una sesión y de saltar a la siguiente tarea. En caso de que se esté ejecutando una prueba, esta función redirigiría la aplicación al menú de tareas.

- **Pausa:** Se encarga de pausar la ejecución de las tareas, especialmente del tiempo de sesión.

- **Eventos de teclas:** Sin ser una función en concreto, se han implementado los eventos de las teclas de dirección y espacio para poder interactuar con algunas de las tareas que así lo han requerido. Así, estas funciones son especificadas para cada tarea en concreto, ya que según la tarea, una misma tecla puede realizar distintas acciones.

Es importante señalar el uso de la función de *Flash*, llamada "*setInterval*". Esta función (similar a la función de *javaScript* que lleva el mismo nombre) establece una llamada periódica en milisegundos a la función que se desee. Por ejemplo, para mostrar por pantalla el tiempo transcurrido, se hace una llamada periódica a la función

"Temporizadores" cada mil milisegundos, es decir, cada segundo. En las tareas se usa esta función para pasar de función en función e ir mostrando imágenes y resultados de forma secuencial.

Todas las tareas tienen en común diversas variables como son el nivel actual, el ensayo actual, el número de errores totales, aciertos totales y aciertos consecutivos. Estos valores se van actualizando cada vez que se realiza la comprobación de respuesta de un ensayo.

Todas las tareas cuentan con seis niveles de dificultad y, dentro de cada nivel, ensayos, algunas seis ensayos y otras diez, a excepción de la tarea de El Bosque, que cada nivel tiene un único ensayo. En la descripción de cada tarea se verán los requisitos para pasar de nivel.

Al margen de existir recursos externos (imágenes) de algunas tareas, todas tienen un archivo de configuración (texto plano) para poder ajustar determinados valores de variables como, por ejemplo, tiempos de respuesta, de feedback o velocidad. Alguno de estos archivos es más extenso y contiene las imágenes que se muestran en cada ensayo. Estos recursos se encuentran dentro de la carpeta correspondiente en el árbol de directorios de la aplicación.

Unas funciones comunes en todas las tareas, aunque personalizadas según el caso en su propio código, son la que inicia la ejecución, llamada tras finalizar la cuenta atrás, la que muestra la imagen de feedback y la que evalúa la respuesta que se ha dado, actualizando los valores de ensayos y niveles.

5.3.2. Implementación de las tareas.

En este apartado se describen todas y cada una de las tareas realizadas, así como una pequeña descripción de las funciones y métodos utilizados para su implementación al margen de aquellos comunes, vistos en el apartado anterior.

Este apartado está subdividido a su vez en diez apartados que corresponden a las diez tareas implementadas.

5.3.2.1 Retratos.

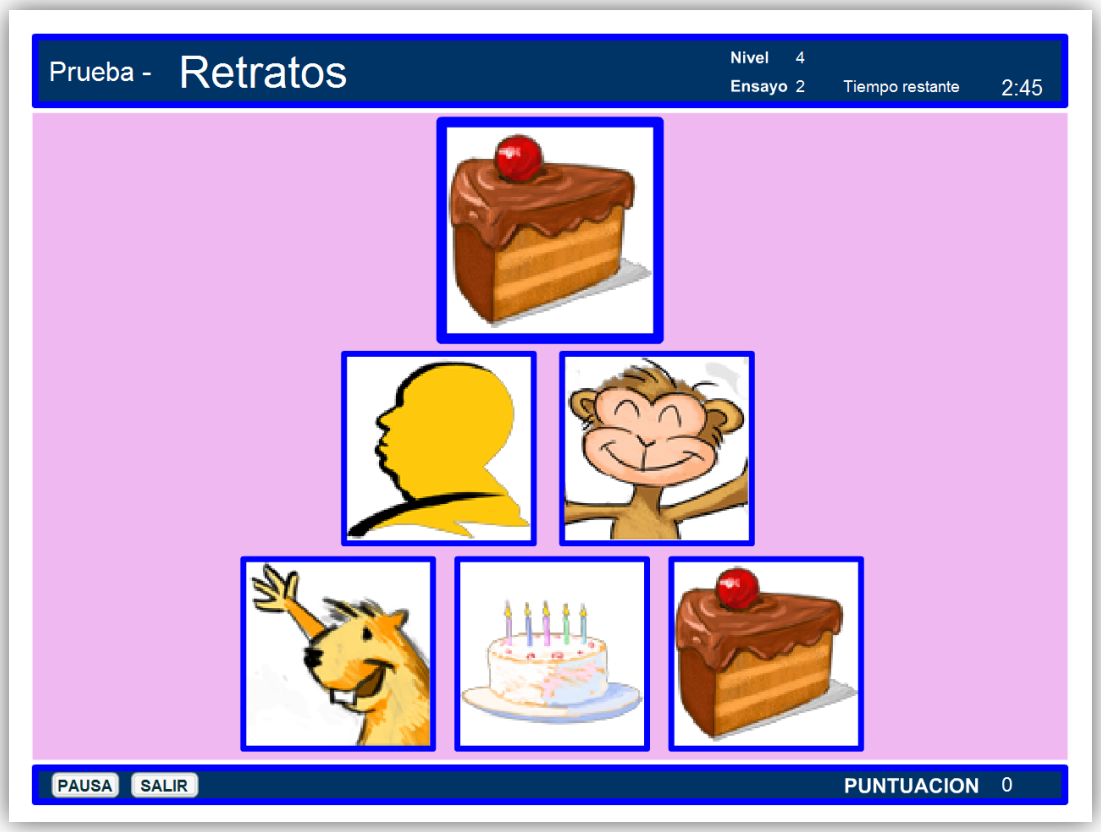


Figura 5.13 – Vista de ejemplo de la tarea Retratos.

Dado un estímulo de muestra (imagen situada en la parte superior), el niño deberá seleccionar el estímulo de comparación igual a la muestra dentro del conjunto de estímulos que aparecen debajo.

Consta de seis niveles de dificultad y seis ensayos por nivel. Para superar cada nivel se han de contestar correctamente cuatro ensayos consecutivos. El aumento de dificultad por cada nivel consiste en incrementar las imágenes de respuesta en una, comenzando en dos imágenes en el nivel uno y terminando en siete en el nivel seis.

En el archivo de configuración de esta tarea se pueden modificar el tiempo de feedback, los tiempos de muestra de la imagen superior (aunque posteriormente no desaparezca) y la imagen que será la correcta de cada ensayo.

En cuanto a la implementación, esta consta de siete botones que representan las siete posibles respuestas y que muestra según sea el nivel actual. Hay un objeto *MovieClip* que representa la imagen principal y que es igual a una de las que aparece en los botones y que es, precisamente, la que el sujeto debe seleccionar.

El inicio de la ejecución comienza cargando las imágenes del ensayo actual y mostrando la imagen de muestra. Pasado el tiempo de muestra, aparecen todas las imágenes, manteniendo la imagen de muestra en pantalla y esperando una respuesta por parte del sujeto. Una vez se ha hecho clic sobre la imagen, se evalúa, se actualizan valores (como por ejemplo, los aciertos consecutivos o errores cometidos) y se muestra la imagen de feedback (distinta según la respuesta sea correcta o incorrecta) sobre el estímulo seleccionado. Pasado el tiempo de feedback se eliminan todas las imágenes de pantalla y se vuelve a comenzar con los nuevos valores de ensayo y nivel.

5.3.2.2. El bosque.

Figura 5.14 – Vista de ejemplo de la tarea El Bosque.

Esta tarea consiste en mover una figura (un perro) con las teclas (o un joystick que genere eventos de teclado) para atrapar la pelota una vez que ha salido del árbol. La pelota irá rebotando por las distintas partes del escenario hasta entrar en el árbol. Una vez dentro, su trayectoria será recta y llevará velocidad constante, por lo que el sujeto deberá inferir el lugar por el que saldrá del árbol y situar el perro en esa posición para poder atraparla. Una vez la pelota ha salido del árbol, ésta permanecerá inmóvil durante un tiempo límite esperando ser atrapada. En caso de no ser atrapada, el nivel se repetirá y, en caso de atraparla, se pasará al siguiente nivel.

Consta de seis niveles de dificultad. El aumento de dificultad por cada nivel reside en determinados factores como el tiempo de respuesta, la inclusión de otras

pelotas dentro del árbol para que distraigan al sujeto (distractores) e incluso de la desaparición de la pelota mientras esté atravesando el árbol.

En el archivo de configuración de esta tarea se pueden modificar el tiempo de feedback, los tiempos de respuesta de cada nivel y la velocidad de la pelota.

En cuanto a la implementación, esta tarea consta de varios objetos tipo *MovieClip*: el árbol, el perro y las pelotas. El árbol es el único de todos ellos que no posee animación pero si interacción con el resto, puesto que el perro no puede entrar en él y las pelotas distractoras no pueden salir de él.

El inicio de la ejecución comienza estableciendo la posición y dirección de la pelota. Una vez establecido el nivel, comienza pasados dos segundos y es cuando se activan los eventos de teclado. Una vez comenzada la tarea, cada diez milisegundos se llama a la función que refrescará los elementos en pantalla. Una vez se detecta que la pelota ha colisionado con el árbol, ésta pasa a modo "DentroArbol" para que cuando vuelva a estar fuera, se detenga y se active la opción de ser atrapada. Una vez atrapada, o no, muestra el feedback y el nivel comienza con los nuevos valores según sea la respuesta.

5.3.2.3. Los pájaros.

Figura 5.15 – Vista de ejemplo de la tarea Los Pájaros.

Esta tarea de tipo *Stroop*, consiste en pulsar la tecla (izquierda, derecha, arriba o abajo) correspondiente a la dirección a la que mira el pájaro central de un grupo de cinco pájaros.

Consta de seis niveles de dificultad y diez ensayos por nivel. Para superar cada nivel se han de contestar correctamente a los ensayos siete, ocho, nueve y diez o tras ellos, correctamente a tres consecutivos que se eligen de forma aleatoria. El aumento de dificultad por cada nivel reside en incrementar el porcentaje de imágenes no congruentes, es decir, aquellas en las que el pájaro central mira hacia el lado contrario del resto.

En el archivo de configuración de esta tarea se pueden modificar el tiempo de feedback, el tiempo que habrá entre un ensayo y otro y la imagen a mostrar.

En cuanto a la implementación, esta tarea consta de cinco componentes "Loader" (cargadores de imágenes) ocultos. Según el ensayo y la imagen a mostrar será visible uno u otro (dentro de las cuatro posibilidades) y, para mostrar la imagen de respuesta, el quinto cargador. También posee otros elementos gráficos de adorno tales como las nubes y el fondo.

El inicio de la ejecución comienza cargando la imagen del nivel ensayo y mostrándola en pantalla. Una vez mostrada, ésta se mantiene en pantalla esperando una respuesta por parte del sujeto. Una vez se ha seleccionado una respuesta, se evalúa, se actualizan valores y se muestra la imagen de feedback sobre la imagen. Pasado el tiempo de feedback se elimina la imagen de pantalla y se vuelve a comenzar con los nuevos valores de ensayo y nivel.

5.3.2.4. Retratos con demora.

Figura 5.16 – Vista de ejemplo de la tarea Retratos con demora.

Esta tarea es muy parecida a la primera tarea descrita, Retratos. La única diferencia que existe con esa tarea es que, mientras que en la primera la imagen de muestra era fija durante todo el ensayo, en esta, la imagen de muestra se oculta tras un tiempo de exposición. Así pues, esta tarea consiste en hacer clic sobre la imagen, de todas las que aparecen, que es igual a la que se mostró, en un evento anterior.

Consta de seis niveles de dificultad y seis ensayos por nivel. Para superar cada nivel se han de contestar correctamente cuatro ensayos consecutivos. El aumento de dificultad por cada nivel reside en incrementar las imágenes de respuesta en una, comenzando en dos imágenes en el nivel uno y terminando en siete en el nivel seis.

En el archivo de configuración de esta tarea se pueden modificar el tiempo de feedback, los tiempos de exposición de la imagen de muestra y la imagen que será la correcta de cada ensayo.

En cuanto a la implementación, esta consta de siete botones que representan las siete posibles respuestas y que muestra según sea el nivel actual. Hay un objeto tipo *MovieClip* que representa la imagen principal y que es igual a una de las que aparece en los botones y que, precisamente, es la que el sujeto debe seleccionar como respuesta correcta.

El inicio de la ejecución comienza cargando las imágenes del nivel ensayo y mostrando la imagen de muestra. Pasado el tiempo de exposición, la imagen de muestra se oculta y se muestran todas las posibles respuestas. Las respuestas se mantienen en pantalla hasta que el sujeto responda. Una vez se ha hecho clic sobre la imagen, se evalúa, se actualizan valores y se muestra la imagen de feedback sobre la respuesta elegida. Pasado el tiempo de feedback, se eliminan todas las imágenes de pantalla y se vuelve a comenzar con los nuevos valores de ensayo y nivel.

5.3.2.5. Stroop numérico.

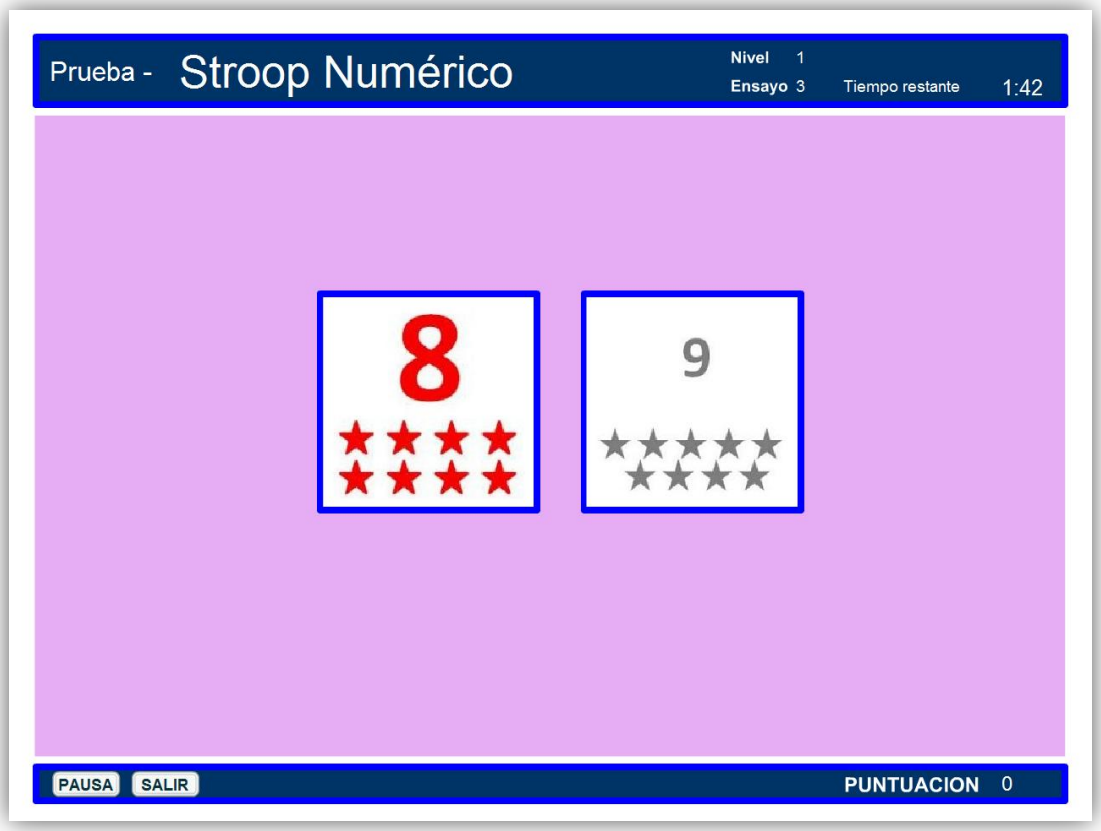


Figura 5.17 – Vista de ejemplo de la tarea Stroop Numérico.

Esta tarea consiste en la muestra de dos imágenes con cantidades, ya sean numéricas o simbólicas, con diferentes tamaños de representación. Así pues se puede dar el caso en el que aparezca un nueve pequeño o nueve estrellas pequeñas y un uno grande o una estrella grande. En cualquier caso, el sujeto debe discriminar el tamaño y responder el mayor valor según la cantidad representada.

Consta de seis niveles de dificultad y seis ensayos por nivel. Para superar cada nivel se han de contestar correctamente cuatro ensayos consecutivos. El incremento de dificultad reside en la variación del tipo de estímulos mostrados. En los dos primeros niveles, los estímulos estarán compuestos por imágenes con estrellas y números; en los niveles tres y cuatro sólo de números; y en los dos últimos niveles solo de estrellas.

En el archivo de configuración de esta tarea se pueden modificar el tiempo de feedback, el tiempo entre ensayos y las imágenes que se sitúan en la parte izquierda y derecha en cada ensayo.

En cuanto a la implementación, esta tarea consta de dos componentes "Loader" sobre un fondo estático. En cada componente de carga se mostrará una de las imágenes y será mediante el teclado con lo que se recoja la respuesta del sujeto.

El inicio de la ejecución comienza cargando las imágenes del ensayo actual, esperando sin un tiempo límite la respuesta por parte del sujeto. Una vez se ha tecleado una respuesta, se evalúa, se actualizan valores y se muestra la imagen de feedback sobre la respuesta elegida. Pasado el tiempo de feedback, se eliminan todas las imágenes de pantalla y se vuelve a comenzar con los nuevos valores de ensayo y nivel.

5.3.2.6. Actualización.

Figura 5.18 – Vista de ejemplo de la tarea Actualización.

Esta tarea consiste en la muestra de una imagen y en contestar SI o NO dependiendo de si esa imagen ha sido mostrada dos ensayos atrás. Se trata de una tarea N-back (Para N=2), diseñada para el entrenamiento de la Memoria de Trabajo.

Consta de seis niveles de dificultad y seis ensayos por nivel. Para superar cada nivel se han de contestar correctamente cuatro ensayos consecutivos. El incremento de dificultad reside en el uso más reiterativo de imágenes parecidas en ensayos más cercanos.

En el archivo de configuración de esta tarea se pueden modificar el tiempo de feedback, el tiempo entre ensayos y la imagen que será mostrada en cada uno de los ensayos.

En cuanto a la implementación, esta tarea consta de varios gráficos estáticos, dos botones de respuesta y una animación (la cinta transportadora), que contiene además un componente de carga de imágenes que se mueve a lo ancho de la pantalla.

El inicio de la ejecución comienza cargando la imagen del ensayo actual e iniciándose el movimiento a través de la cinta transportadora. Al llegar al centro de la cinta, la imagen se parará y esperará una respuesta del sujeto sin tiempo límite. Una vez se ha hecho clic sobre una de las dos posibles respuestas, ésta será evaluada y, actualizándose valores (como por ejemplo, los aciertos consecutivos o errores cometidos) será mostrada la imagen de feedback sobre la respuesta elegida. A la vez, la imagen seguirá su camino sobre la cinta y, al llegar al lado derecho de la pantalla se esperará un tiempo establecido a que comience el nuevo ensayo con los valores nuevos.

5.3.2.7. La Granja.

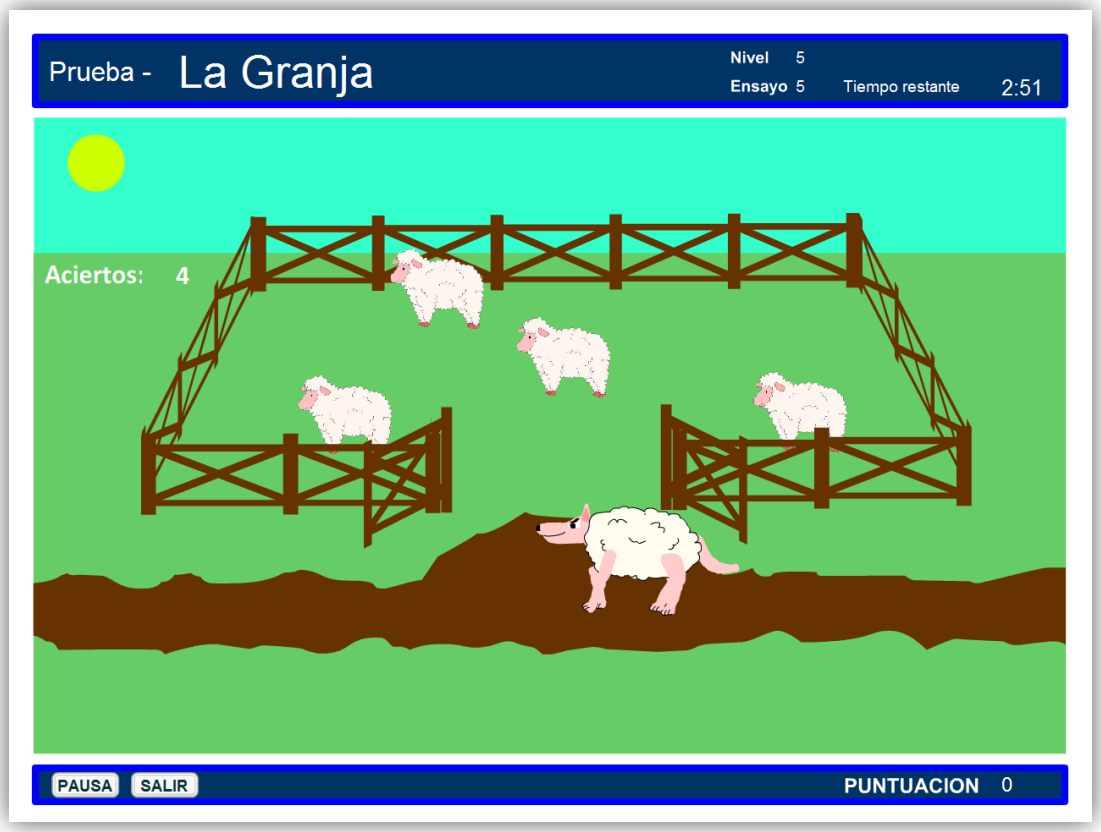


Figura 5.19 – Vista de ejemplo de la tarea La Granja.

Esta tarea consiste en introducir la mayor cantidad posible de ovejas dentro del corral evitando introducir a los lobos dentro por motivos obvios. Para ello, el sujeto tendrá que pulsar la tecla "espaciadora" justo cuando la oveja se sitúe justo enfrente de la puerta del corral.

Consta de seis niveles de dificultad. Para superar cada nivel se han de contestar correctamente cinco ensayos consecutivos. Aunque este valor es por defecto, se puede configurar. El incremento de dificultad reside, por un lado, en la disminución del tiempo para contestar. Por otro lado, en que la imagen del lobo se va pareciendo a la oveja (véase Figura 5.19). Y por último, en los dos últimos niveles, el lobo comienza con el aspecto de una oveja y, tras un tiempo de parpadeo (tiempo que transcurre entre que aparece la imagen de la oveja hasta que esta cambia), cuando el animal está cerca

de la puerta del corral, éste se transforma en lobo. De esta manera el tiempo de reacción para evaluar la respuesta es menor.

En el archivo de configuración de esta tarea se pueden modificar los tiempos de respuesta, el tiempo de parpadeo, aciertos necesarios para pasar de nivel y la proporción de ovejas respecto a los lobos.

En cuanto a la implementación, esta tarea consta de varios gráficos estáticos y animaciones. El corral es una animación en la que se van sumando ovejas y el camino es otra animación con un componente de carga en el que se van mostrando las imágenes.

El inicio de la ejecución comienza cargando la imagen del ensayo actual, iniciándose el movimiento de la oveja (o del lobo) de derecha a izquierda. Pasado un tiempo, las puertas del corral se abren y es cuando el sujeto debe contestar pulsando la tecla espaciadora (si es una oveja), u omitiendo la acción (si es un lobo), para dejarlo pasar. La respuesta será evaluada al pulsar la tecla o al desaparecer la imagen por la izquierda. Una vez se haya evaluado la respuesta, se actualizarán los valores y la animación del corral, mostrando una oveja más o ninguna si se ha errado o comenzado un nuevo nivel. Tras esto, volverá a salir una imagen por la derecha hasta que se termine el tiempo o se superen todos los niveles.

5.3.2.8. Las Caras.

Figura 5.20 – Vista de ejemplo de la tarea Las Caras.

Esta tarea consiste en teclear la flecha (izquierda o derecha) correspondiente a la dirección hacia donde miran los ojos de las caras que se muestran de forma alterna en tres posiciones distintas: izquierda, centro y derecha.

Consta de seis niveles de dificultad y diez ensayos por nivel. . Para superar cada nivel se han de contestar correctamente a los ensayos siete, ocho, nueve y diez o, tras ellos, correctamente a tres consecutivos que se eligen de forma aleatoria. El aumento de dificultad por cada nivel reside en incrementar el porcentaje de imágenes no congruentes, es decir, aquellas en las que la imagen mira al lado contrario de donde está situada. Por ejemplo, mirar a la derecha habiendo salido en el recuadro de la izquierda.

En cuanto a la implementación, esta tarea consta de tres componentes "Loader" (cargadores de imágenes) ocultos.

El inicio de la ejecución comienza cargando la imagen del nivel ensayo y mostrándola en pantalla. Una vez mostrada, ésta se mantiene en pantalla esperando una respuesta por parte del sujeto. Una vez se ha tecleado una respuesta, se evalúa, se actualizan valores y se muestra la imagen de feedback correspondiente. Pasado el tiempo de feedback, se elimina la imagen de pantalla y se vuelve a comenzar con los nuevos valores de ensayo y nivel.

5.3.2.9. Los Peces.

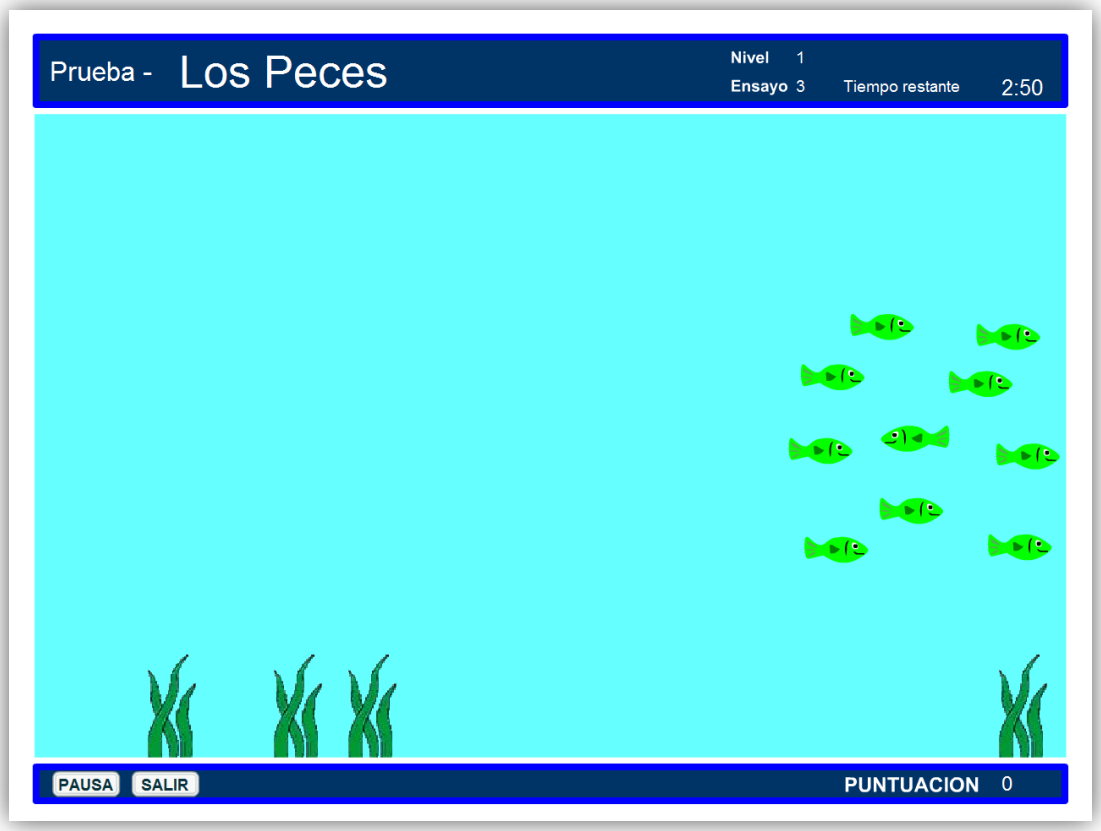


Figura 5.21 – Vista de ejemplo de la tarea Los Peces.

Esta tarea es muy parecida a la tarea de Los Pájaros con la diferencia de que la imagen central puede estar en cualquiera de las cuatro orientaciones posibles: izquierda, derecha, arriba y abajo. En Los Pájaros, las imágenes horizontales solo podían estar orientadas a izquierda-derecha y las verticales arriba-abajo. Al igual que en Los Pájaros, los sujetos deben abstraerse del resto de peces distractores y responder pulsando la tecla correspondiente a la orientación del pez central.

Consta de seis niveles de dificultad y diez ensayos por nivel. Para superar cada nivel se han de contestar correctamente a los ensayos siete, ocho, nueve y diez o, tras ellos, correctamente a tres consecutivos que se eligen de forma aleatoria. El aumento de dificultad por cada nivel reside en incrementar el porcentaje de imágenes no

congruentes, es decir, aquellas en las que el pez central mira en una dirección distinta al resto de peces.

En el archivo de configuración de esta tarea se pueden modificar el tiempo de feedback, el tiempo que habrá entre un ensayo y otro y la imagen a mostrar.

En cuanto a la implementación, esta tarea consta de cinco componentes "Loader" (cargadores de imágenes) ocultos. Según el ensayo y la imagen a mostrar será visible uno u otro, de las cuatro posibilidades, y para mostrar la imagen de respuesta el quinto cargador. También posee otros elementos gráficos de adorno tales como las algas y el fondo.

El inicio de la ejecución comienza cargando la imagen del nivel ensayo y mostrándola en pantalla. Una vez mostrada, ésta se mantiene en pantalla esperando una respuesta por parte del sujeto. Una vez se ha tecleado una respuesta, se evalúa, se actualizan valores y se muestra la imagen de feedback sobre la imagen. Pasado el tiempo de feedback, se elimina la imagen de pantalla y se vuelve a comenzar con los nuevos valores de ensayo y nivel.

5.3.2.10. Reciclaje.

Figura 5.22 – Vista de ejemplo de la tarea Reciclaje.

Esta tarea consiste en la muestra de una imagen que se transporta sobre la cinta transportadora. Cuando la imagen está parada en el centro de la cinta, el sujeto pulsará la tecla espaciadora, si la imagen mostrada no se corresponde con la que se busca, que será especificada al comienzo de cada nivel. En este caso, en los dos primeros niveles se deben dejar pasar a las botellas; en los niveles tres y cuatro a los tetrabrikes; y en los dos últimos a las cajas.

Consta de seis niveles de dificultad y seis ensayos por nivel. Para superar cada nivel se han de contestar correctamente cinco ensayos consecutivos, aunque este valor es configurable. El incremento de dificultad reside en la disminución del tiempo de respuesta.

En el archivo de configuración de esta tarea se pueden modificar el tiempo de respuesta de cada nivel, el número de aciertos consecutivos necesarios para pasar de nivel y la proporción de basura descartable con respecto de la reciclable.

En cuanto a la implementación, esta tarea consta de varios gráficos estáticos y varias animaciones: (1) la cinta transportadora, que contiene además un componente de carga de imágenes que se mueve a lo ancho de la pantalla; y (2) la trampilla, que se abre para dejar pasar los elementos descartados.

El inicio de la ejecución comienza cargando la imagen del ensayo actual e iniciándose el movimiento a través de la cinta transportadora de izquierda a derecha. Al llegar al centro de la cinta, la imagen se parará y esperará una respuesta del sujeto con un tiempo límite. Si hay respuesta, pulsando la tecla espaciadora, se abrirá la trampilla y el elemento caerá al fondo. Si no hay respuesta, el elemento seguirá su camino por la cinta hasta llegar al final de la misma. En cualquiera de los dos casos, la respuesta se evaluará después de finalizar la animación, se actualizan valores y se mostrará el feedback de la respuesta. Pasado el tiempo de feedback se elimina la imagen de pantalla y se vuelve a comenzar con los nuevos valores de ensayo y nivel.

Capítulo 6 – Evaluación de AT²-Training

La prueba del software es un elemento de un tema más amplio que, a menudo, es conocido como verificación y validación (V & V). La verificación se refiere al conjunto de actividades que aseguran que el software implementa correctamente una función específica. La validación se refiere a un conjunto diferente de actividades que aseguran que el software construido se ajusta a los requisitos del cliente.

6.1. Validación

Como se ha descrito anteriormente, la validación se refiere al conjunto de actividades que aseguran que el software ha sido construido en base a los requisitos y necesidades del cliente y los usuarios finales. En esta fase son ellos, cliente y usuarios, quienes deben realizar una gran parte de la actividad aportando las respuestas necesarias para que los desarrolladores encaucen el software. Los desarrolladores por su parte también deben ayudar en base a su experiencia a que el cliente sepa plasmar esas necesidades que llevaran a construir el producto. Una de las alternativas más común a la hora de ayudar a los clientes a ver esas necesidades es que los

desarrolladores presenten prototipos intermedios con diversas soluciones. De esta manera, la validación se ha realizado dentro de la actividad fundamental de requisitos. En este caso, clientes y usuarios tienen conocimientos suficientes para comprender las notaciones utilizadas, por lo tanto se han podido utilizar en la validación de los modelos de análisis, es decir, en la validación de los casos de uso y en los diagramas de clases, los cuales son modificados por los clientes y reenviados a los desarrolladores. Este ha sido el caso de AT²-Training, ya que los requisitos han sido definidos por ingenieros informáticos con los conocimientos suficientes de UML, y la destreza necesaria para la revisión de los prototipos y de los diagramas sin supervisión alguna.

Las actividades de validación al ser realizadas conllevan generalmente nuevos requisitos, esto es debido a que a medida que se perfila el sistema, aparecen nuevas necesidades que antes se encontraban ocultas, sobre todo cuando se van presentando prototipos intermedios. Para este desarrollo se han llevado a cabo tres iteraciones. En la segunda y la tercera iteración ya estaba disponible un prototipo evaluable por el cliente. El resultado final fue la obtención de un catálogo de requisitos válido, casos de uso, y parte de la especificación de las clases de análisis, también validadas por los clientes, que permitió abordar el diseño de la aplicación final.

6.2. Verificación

Las actividades de verificación son aquellas actividades que aseguran que el software que ha sido desarrollado desempeña correctamente las funciones para las que ha sido desarrollado. De esta manera, una vez finalizado el desarrollo, éste debe ser probado de forma exhaustiva para descubrir y corregir el máximo de errores antes de su entrega final. En este paso se deben diseñar casos de prueba que tengan una alta probabilidad de encontrar errores. Estas pruebas, llamadas pruebas de defectos, se suelen acompañar de las pruebas estadísticas que se utilizan para verificar el desempeño de la fiabilidad del programa y comprobar cómo trabaja bajo condiciones operacionales, constatando los accesos de los usuarios y su frecuencia. Éstas últimas se han puesto en práctica durante las tareas de aceptación por parte de los clientes/usuarios.

La prueba de defectos consiste en dos etapas: las pruebas del componente (realizadas por el desarrollador) y las pruebas de integración (realizadas por un equipo independiente). La primera de las etapas comprende las pruebas de funcionamiento de los componentes claramente identificables, que son funciones o grupos de métodos agrupados en módulos o en objetos. Durante las pruebas de integración, que pare este desarrollo también han sido realizadas por el desarrollador debido a la limitación de recursos, los componentes se integran para formar subsistemas o el sistema completo. En esta etapa las pruebas se enfocan en las interacciones entre componentes y en la funcionalidad y el desempeño del sistema como un todo.

Para el tipo de aplicación desarrollada en este caso existen dos tipos de componentes bien diferenciados, la película *Flash* y los objetos. Para los componentes orientados a objetos que se han desarrollado han sido aplicados tres niveles de pruebas:

1. Probar las operaciones individuales asociadas con los objetos. Se han aplicado técnicas de caja blanca y caja negra.

2. Probar clases de objetos individuales, aplicando pruebas de caja negra y teniendo en cuenta la herencia y el polimorfismo entre clases.

3. Probar cluster de objetos. La integración estricta descendente o ascendente no es apropiada para crear grupos de objetos relacionados y se utiliza un enfoque basado en casos de uso. Cada caso de uso describe un escenario de uso del sistema y un cluster de objetos que se prueban de forma integrada. Como un ejemplo para este nivel tenemos la creación o eliminación de planes en el sistema de gestión.

6.3. Aceptación de los usuarios

Una vez se tiene la especificación del sistema que se va a construir, se ha de tener presente que el prototipo implementado debe ser aceptado por los usuarios finales, en este caso por las personas encargadas de realizar las pruebas sobre los

sujetos. Para conocer la opinión de los responsables de dicha labor se han elaborado una serie de cuestionarios que proporcionarán el feedback necesario para la mejora del prototipo. Los resultados de dichos cuestionarios aún no han sido recibidos.

6.3.1. Cuestionario de validación: explicación y funcionalidad

Para conocer si el sistema satisface las necesidades y requisitos de los usuarios, además de si tienen una interacción óptima con el sistema, se les ha presentado el siguiente cuestionario:

Pregunta	Valoración (1 - 10)	Comentario
¿Considera que la distribución de contenidos es correcta?		
Al introducir nueva información, ¿sabe exactamente cómo y dónde hacerlo?		
Cuando se realiza una tarea con el sistema, ¿cree que puede acceder a toda la información que necesita?		
El sistema le ayuda con información adecuada (mensajes de error, avisos…)		
¿Cree que el sistema le permite hacer todo lo que necesita?		
El sistema responde de forma apropiada a las acciones del usuario.		
El sistema justifica por qué ha solicitado una determinada información.		
Los mensajes de explicación son adecuados		
El sistema explica convenientemente las situaciones especiales.		

Tabla 6.1 – Cuestionario de validación

6.3.2. Cuestionario sobre oportunidad y previsiones futuras

Para conocer la utilidad sobre este tipo de sistemas a analistas y desarrolladores, ponemos el siguiente cuestionario a disposición de los usuarios para que sean éstos los que puedan realizar cualquier tipo de consideración sobre la oportunidad e implantación de este tipo de sistemas en las organizaciones.

Pregunta	Comentario
¿Hacia quién cree que va dirigida esta aplicación?	
En el desarrollo de su proyecto software, ¿para qué cree que le puede ayudar?	
¿Qué problema cree que resuelve mejor esta aplicación?	
¿En qué otros temas cree que se puede usar este tipo de herramientas basadas en internet?	
¿Conoce algún otro sistema o herramienta que tenga una finalidad parecida?	

Tabla 6.2 – Cuestionario sobre oportunidad y previsiones futuras

6.3.3. Críticas, mejoras, sugerencias y errores detectados

Para poder recibir un feedback valioso y de calidad acerca de la funcionalidad del prototipo desarrollado, ponemos a disposición de los usuarios el siguiente cuestionario en el que podrán justificar cualquier aportación que puedan hacer.

Sugerencia	Objetivo	Justificación
Se ha encontrado el error Añadiría Mejoraría Falta No hace No es correcto Aclararía Cambiaría …		
Se ha encontrado el error Añadiría Mejoraría Falta No hace No es correcto Aclararía Cambiaría …		

Tabla 6.3 – Cuestionario de críticas, mejoras, sugerencias y errores detectados

Capítulo 7 – Conclusiones y trabajos futuros

En este capítulo veremos las conclusiones a las que hemos llegado tras la realización de la aplicación web y aportaremos trabajos futuros que se puedan realizar en base al proyecto aquí especificado.

7.1. Conclusiones

Pasaremos a detallar de forma breve cada una de las conclusiones a las que hemos llegado tras la finalización y puesta en marcha de la aplicación:

- Se ha diseñado e implementado un prototipo de herramienta web mediante la cual expertos terapeutas puedan planificar sesiones de intervención cognitivo-conductual a través de pruebas informatizadas específicamente diseñadas para el entrenamiento de la atención ejecutiva.

- La herramienta, a la que hemos llamado AT²-Training, nos permite llevar un control preciso sobre la evolución de cada uno de los sujetos que van a realizar el proceso de entrenamiento atencional.

- Se ha utilizado Flash como tecnología web dinámica para el desarrollo del proyecto, lo que ha permitido, de manera sencilla, implementar una interfaz rica, intuitiva y versátil para el desarrollo de las sesiones y tareas que permitirán el entrenamiento de los sujetos.

  Al ser una aplicación web se permite que el acceso a la misma sea posible desde cualquier lugar que tenga acceso a Internet.

7.2. Trabajos futuros propuestos

A lo largo del proceso de creación de la herramienta se han ido planteando diversas ideas y nuevas funcionalidades que harían mucho más completo el diseño inicial, pero que no han podido ser implementadas por salirse del planteamiento inicial y por la complejidad de implantación que aportaban al proyecto una vez superadas determinadas fases del mismo.

Pese a ello, consideramos interesante incluir un pequeño esbozo sobre futuras mejoras a la aplicación que harán posible un mayor control sobre el seguimiento de los sujetos y ampliación de sus funcionalidades, básicas o avanzadas.

A continuación mostraremos algunos de los posibles trabajos que se pueden realizar en base al proyecto finalizado:

- Aumentar el número de tareas disponibles para incrementar las posibilidades en la variedad de planes de entrenamiento atencional e incluso en otros ámbitos de la psicología. Aquí queremos destacar que la inclusión de nuevas tareas, requisito con el que partió este proyecto, es bastante sencillo desde el punto de vista de la programación, debido a la alta modularidad con la que se ha diseñado la herramienta.

- Implementar un sistema de recogida de feedback específico de cada tarea. De esta manera podremos conocer el tiempo de reacción y/o respuesta de cada sujeto ante cada estímulo.

- Implementar un módulo inteligente que aprenda de datos históricos (ejecuciones anteriores) mediante técnicas de minería de datos. La inclusión de este módulo dará lugar a la obtención de una herramienta web adaptativa que ayude a los terapeutas en la planificación de sesiones, adaptando los planes previamente establecidos según las características particulares y específicas de cada sujeto.

Apéndice A – Manual de instalación.

Para poder utilizar la aplicación web desarrollada AT²-Training, en este apartado se especificarán qué herramientas se deben instalar y dónde se pueden conseguir, además de la configuración necesaria y óptima para que todo funcione correctamente.

Para el buen funcionamiento de la aplicación se necesita un servidor web, PHP y un gestor para la base de datos MYSQL; y es XAMPP, en su versión para Windows, una compilación de software libre para el desarrollo de aplicaciones web el que se ha usado en sus últimas versiones. XAMPP aúna varias herramientas libres para el desarrollo y funcionamiento de varias tecnologías entre los que usaremos Apache 2.2, MySQL 5.5.8, PHP 5.3.5 y phpMyAdmin 3.3.9 (última versión disponible en el momento de la creación de este apéndice).

El sistema operativo utilizado durante el desarrollo y sobre el que se han montado estas herramientas es Microsoft Windows 7 Professional.

Como curiosidad, indicar que el acrónimo XAMPP significa: **X** (para cualquiera de los diferentes sistemas operativos), **A**pache, **M**ySQL, **P**HP, **P**erl. En este caso, al usar Windows como sistema operativo, la infraestructura tomada se denomina WAMP.

Estos son los pasos que seguiremos para la instalación:

- Descarga de XAMPP
- Instalación de XAMPP
- Configuración de XAMPP
- Instalación y configuración de AT²-Training

A.1. Descarga de XAMPP.

La web oficial de XAMPP, donde está siempre disponible la última versión, se encuentra en la dirección www.xampp.org.

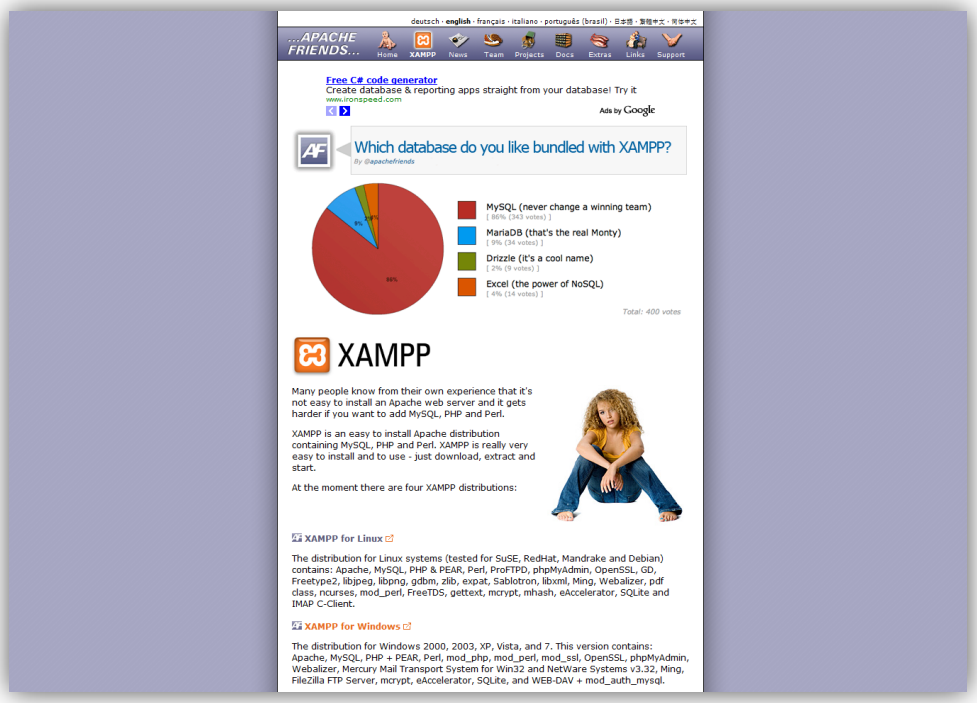


Figura A.1 – Sitio web de XAMPP.

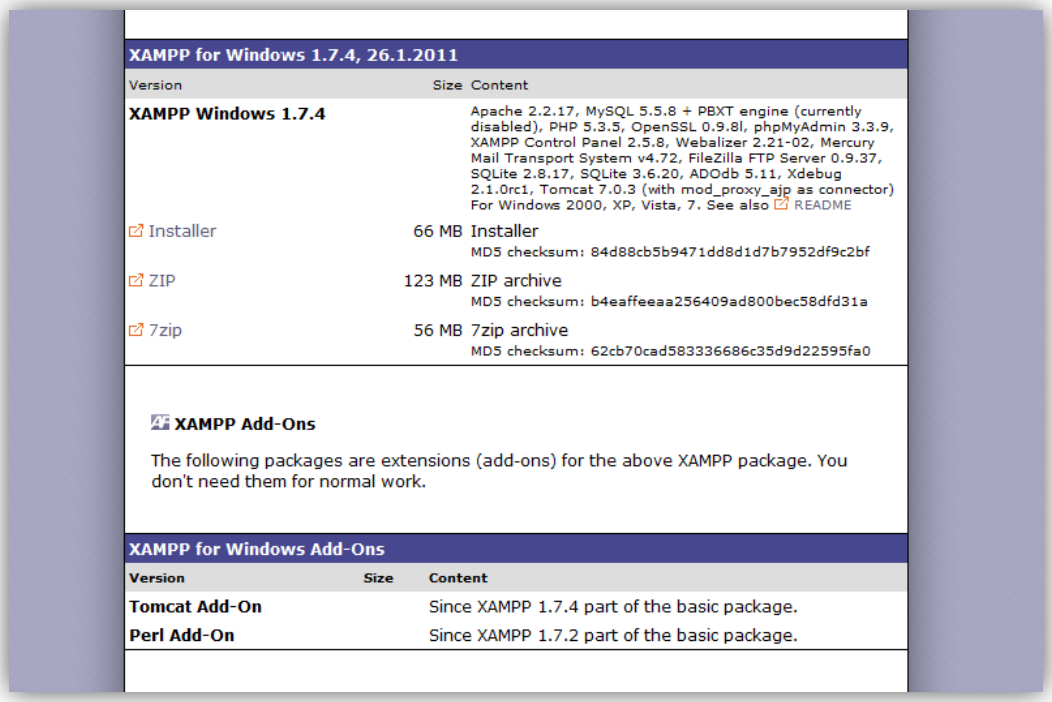
Una vez se acceda a la dirección de la web de XAMPP (Figura A.1), solamente se deberá hacer clic en "XAMPP for Windows", es ahí donde se describe de qué se compone la descarga de XAMPP, la versión actual y las versiones de sus componentes.

Como se puede ver en el enlace (Figura A.2), el paquete contiene más herramientas de las imprescindibles para el desarrollo del proyecto, pero se hará uso y se configurará con las que nombramos anteriormente. En la misma figura, debajo de

"download" hay dos enlaces, se deberá hacer clic en el primero, "XAMPP", para ir a la zona de la web dónde se seleccionará el tipo de archivo a descargar.

Una vez hecho clic en XAMPP se pueden ver los enlaces (véase la Figura A.3) pudiendo proceder con la descarga del "Installer" haciendo clic en su nombre.

Figura A.2 – XAMPP for Windows.



XAMPP for Windows 1.7.4, 26.1.2011

Version	Size	Content
XAMPP Windows 1.7.4		Apache 2.2.17, MySQL 5.5.8 + PBXT engine (currently disabled), PHP 5.3.5, OpenSSL 0.9.8l, phpMyAdmin 3.3.9, XAMPP Control Panel 2.5.8, Webalizer 2.21-02, Mercury Mail Transport System v4.72, FileZilla FTP Server 0.9.37, SQLite 2.8.17, SQLite 3.6.20, ADOdb 5.11, Xdebug 2.1.0rc1, Tomcat 7.0.3 (with mod_proxy_ajp as connector) For Windows 2000, XP, Vista, 7. See also README
Installer	66 MB	Installer MD5 checksum: 84d88cb5b9471dd8d1d7b7952df9c2bf
ZIP	123 MB	ZIP archive MD5 checksum: b4eaffeeaa256409ad800bec58dfd31a
7zip	56 MB	7zip archive MD5 checksum: 62cb70cad583336686c35d9d22595fa0

XAMPP Add-Ons

The following packages are extensions (add-ons) for the above XAMPP package. You don't need them for normal work.

XAMPP for Windows Add-Ons

Version	Size	Content
Tomcat Add-On		Since XAMPP 1.7.4 part of the basic package.
Perl Add-On		Since XAMPP 1.7.2 part of the basic package.

Figura A.3 – Enlaces de descarga.

A.2. Instalación de XAMPP

Una vez que se ha descargado el archivo en el equipo se procederá a ejecutarlo. Cuando dé comienzo la instalación se deberá hacer clic en "*Next* >" con los valores por defecto de la instalación del archivo ejecutable en las ventanas que aparecen como en las Figuras A.4 y A.5.

Al llegar a la pantalla que describe la Figura A.6. también se recomienda dejar todo por defecto, haciendo clic en "Install" para proceder a su instalación. Por supuesto, estas opciones por defecto son modificables sin problema para el funcionamiento de esta aplicación. Los accesos directos se pueden omitir y la instalación de las herramientas como servicio se puede activar, pero para el entorno de desarrollo es no es necesario.

Una vez llegados al punto de la Figura A.7. la instalación habrá terminado y pudiendo ejecutar la consola principal para configurar las aplicaciones.

Figura A.4 – Instalador de XAMPP.

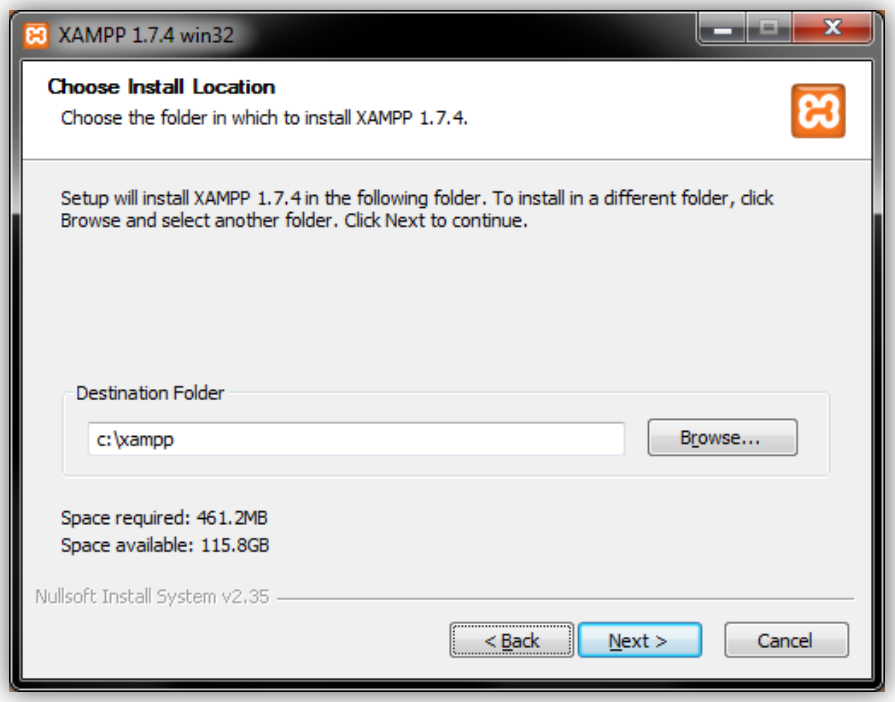


Figura A.5 – Directorio de instalación de XAMPP.

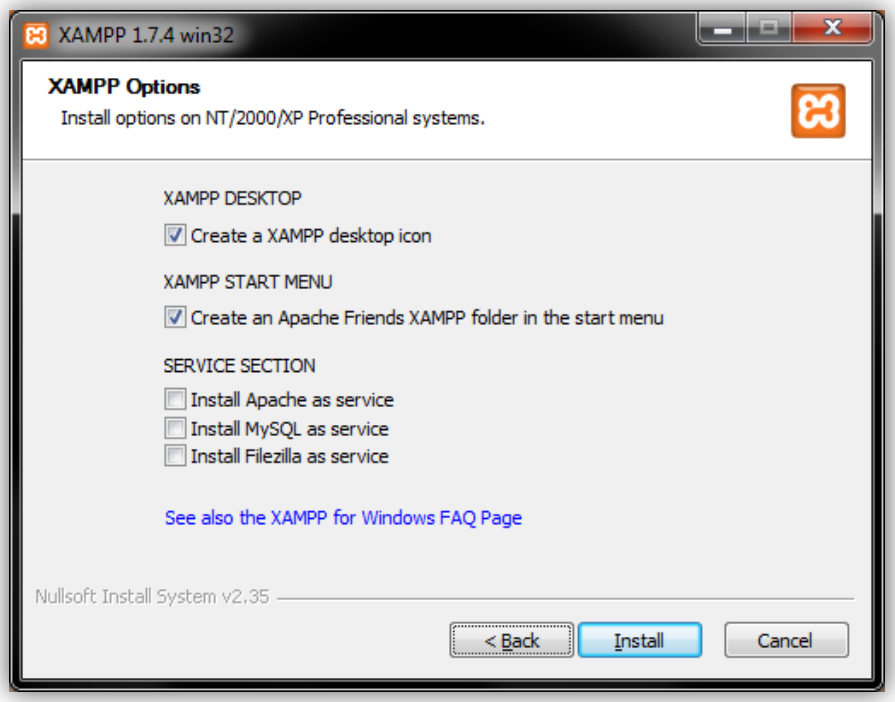


Figura A.6 – Opciones de instalación de XAMPP.

Figura A.7 – Fin de la instalación de XAMPP.

A.3. Configuración de XAMPP

Una vez instalado XAMPP se debe iniciar el programa principal de XAMPP, el *"XAMPP Control Panel Application"* (Figura A.8.).

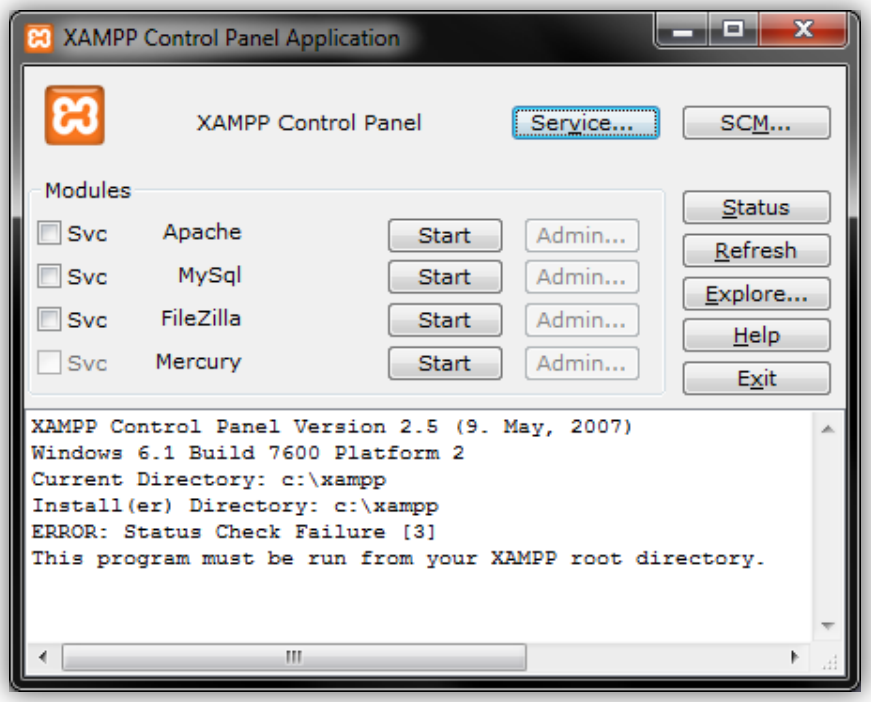


Figura A.8 – Vista principal del XAMPP Control Panel Application.

Desde esta consola principal se pueden activar y desactivar los distintos servicios de las herramientas que se han instalado, así como lanzar las herramientas de administración de las mismas.

Primero se van a activar los servicios Apache y MySql haciendo clic en el botón "*Start*" que hay al lado de sus nombres.

Una vez estén activos ambos servicios (Figura A.9.), se tendrá que hacer clic en la administración de Apache, botón "*Admin...*" que hay a la derecha de Apache o abrir un navegador web e introducir como dirección: http://localhost/. Ahí se verá la configuración actual y más básica de distintos elementos, así como el establecimiento de la seguridad de todo el sistema.

Una vez dentro de la web de administración (Figura A.10), se podrá acceder a la web principal, pudiendo consultar el estado haciendo clic en la opción "Estado", situado en la parte superior izquierda.

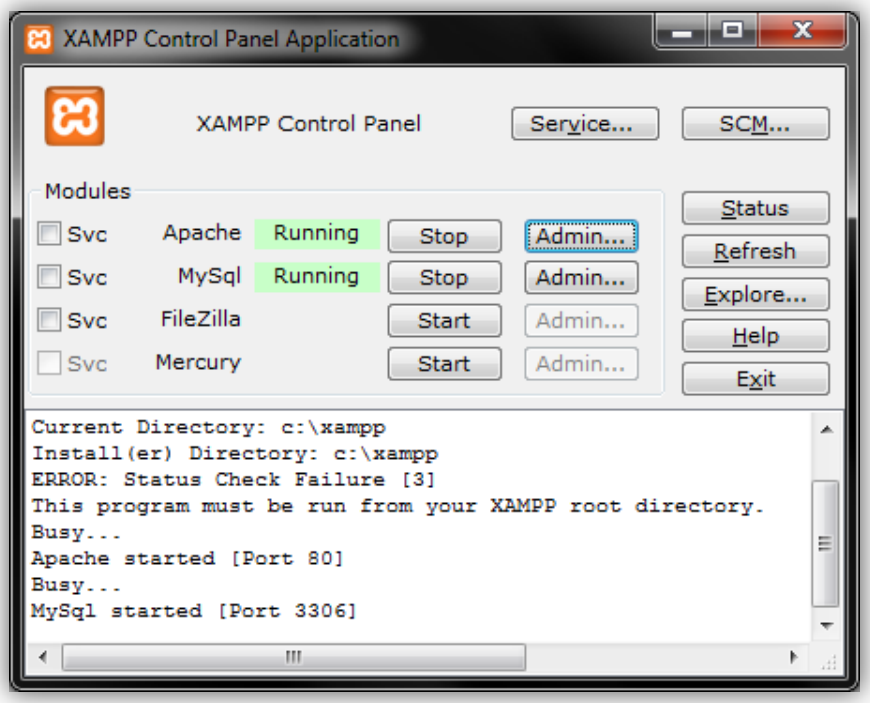


Figura A.9 – XAMPP Control Panel con servicios activos.

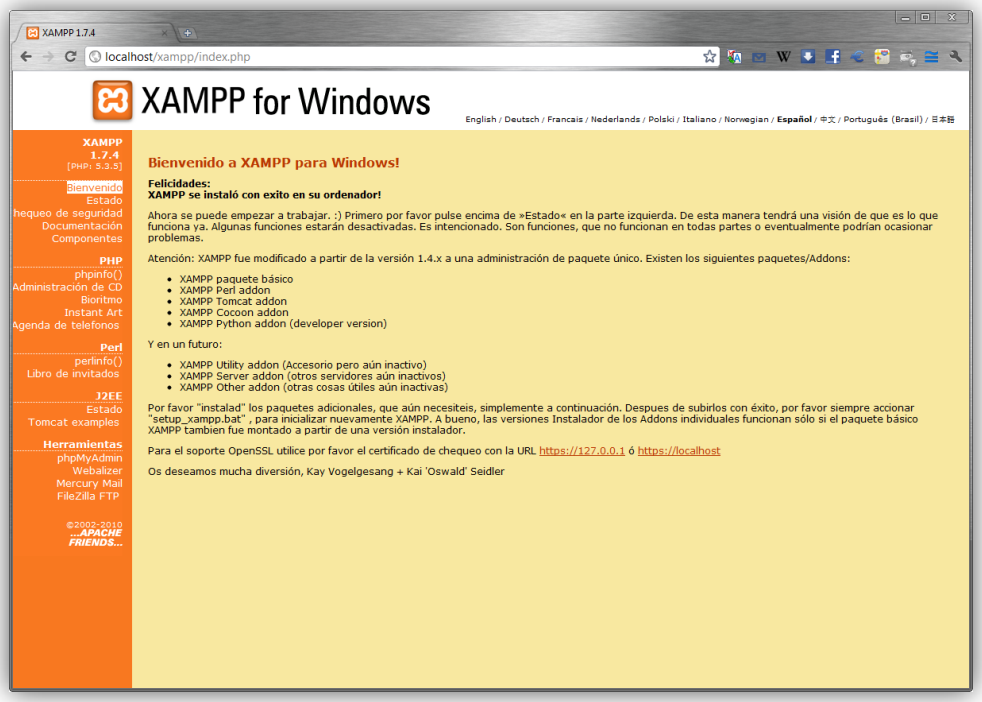


Figura A.10 – Web principal de administración de XAMPP.

Una vez hecho clic en “Estado” se podrá ver cómo se encuentran actualmente los servicios:

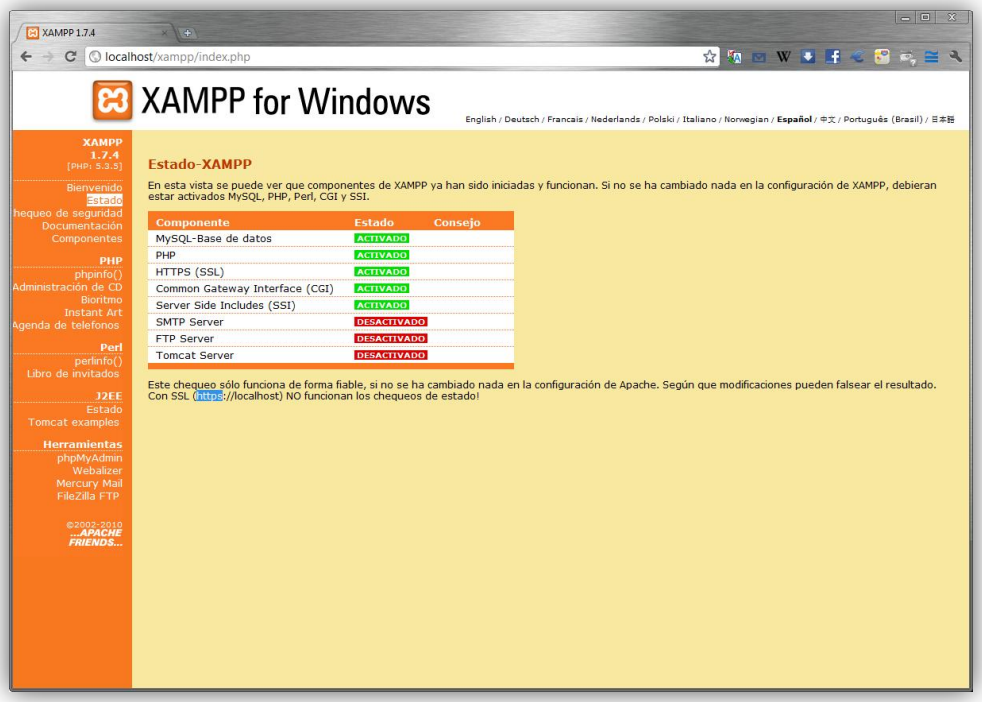


Figura A.11 – Estado de XAMPP.

Como se puede observar, se encuentran activos los servicios de MySQL, PHP, HTTPS, CGI y SSI. No han sido activados ni FTP, ni SMTP, ni Tomcat, ya que no serán necesarios.

También será necesario chequear la seguridad del sistema, ya que hay que tener en cuenta que al estar todo activo, nuestro equipo se encuentra en riesgo. Las mismas webs de administración que manejamos son mostradas a todos aquellos que introduzcan nuestra IP desde el exterior. Por defecto, XAMPP tiene todo abierto y esto es potencialmente inseguro, por lo que se deben establecer contraseñas de *root* a todos los servicios que están expuestos.

XAMPP tiene una parte de chequeo de seguridad que nos permitirá conocer el estado de las contraseñas y su vulnerabilidad. Para ello, se hará clic en "Chequeo de seguridad", opción que se encuentra debajo de "Estado".

En la Figura A.12 se puede observar cómo se encuentra actualmente la seguridad del sistema que hemos montado.

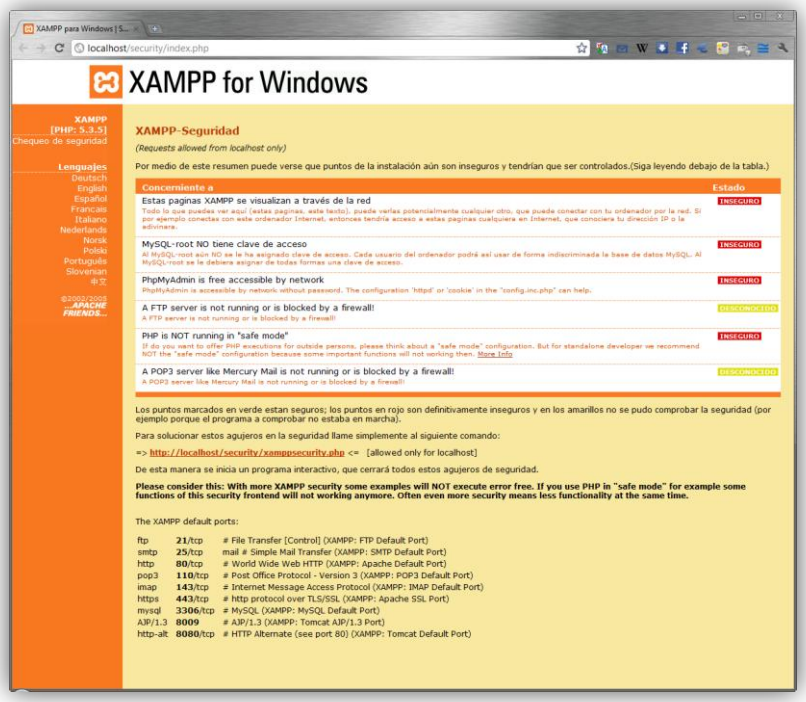


Figura A.12 –Chequeo de seguridad.

Como el propio programa nos indica (Figura A.12), se debe hacer clic en la url http://localhost/security/xamppsecurity.php para acceder a un formulario específico dónde modificar las contraseñas de algunos servicios en riesgo (Figura A.13).

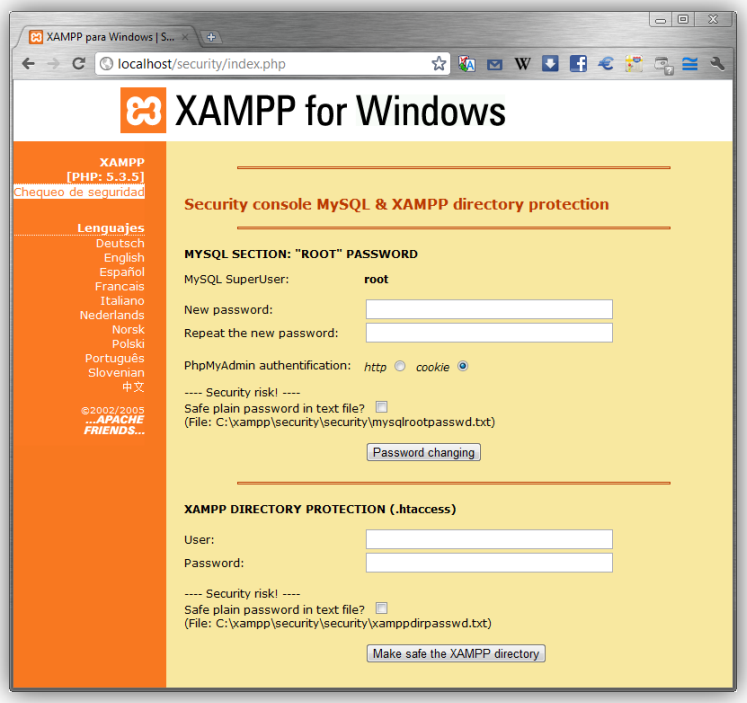


Figura A.13 –Consola de seguridad de MySQL y XAMPP.

Una vez situados en la pantalla de la consola de seguridad (Figura A.13), se introducirá una contraseña válida en ambos apartados. Primero arriba y aceptando el cambio de contraseña y después abajo haciendo seguro el directorio de XAMPP. Una vez realizadas y validadas ambas acciones, se hará clic en "Chequeo de seguridad" de nuevo (arriba a la izquierda también en la figura A.13) y pudiendo ver el estado de la seguridad después de estos cambios (Figura A.14).

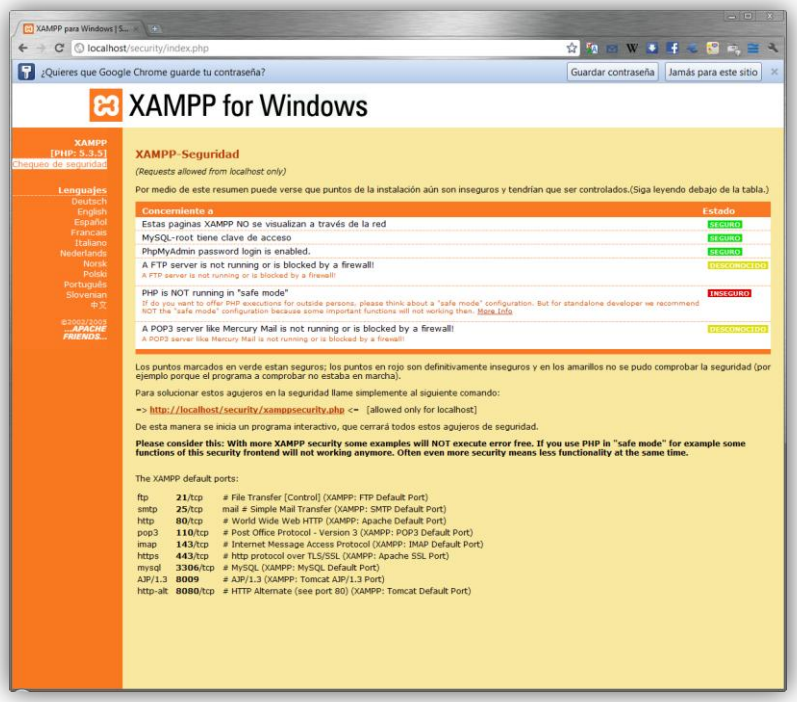


Figura A.14 –Comprobación de seguridad tras los cambios.

Como se puede comprobar en la imagen anterior, ahora todo es seguro excepto PHP, que no es que no sea seguro, sino que no corre en modo seguro. No obstante, se dejará que esto continúe así porque configurar PHP para que trabaje en modo seguro puede hacer que funciones importantes trabajen de forma no correcta, lo cual sería contraproducente para el correcto funcionamiento de nuestra aplicación web y las funciones desarrolladas en PHP para la conexión con la base de datos. Hechos los cambios anteriores, ahora es más inaccesible nuestro sistema desde el exterior, cuestión que es deseable.

Tras la instalación y los cambios anteriormente comentados, con los servicios activos y seguros ahora sólo basta configurar los archivos de la aplicación web y ver que todo funciona con normalidad.

A.4. Instalación y configuración de AT²-Training

Lo que se debe hacer para configurar la aplicación de AT2-Training es copiar la base de datos de MySQL y los archivos propios de la aplicación en la ruta correcta dentro del directorio de XAMPP.

Primero comenzaremos por la base de datos que contiene la estructura básica y algunos registros de prueba introducidos previamente durante el desarrollo de la aplicación.

La información de la base de datos de la aplicación se encuentra en "proyecto", carpeta situada dentro del directorio BD. Esta carpeta se debe copiar en la ruta "C:\xampp\mysql\data" (ver Figura A.15).

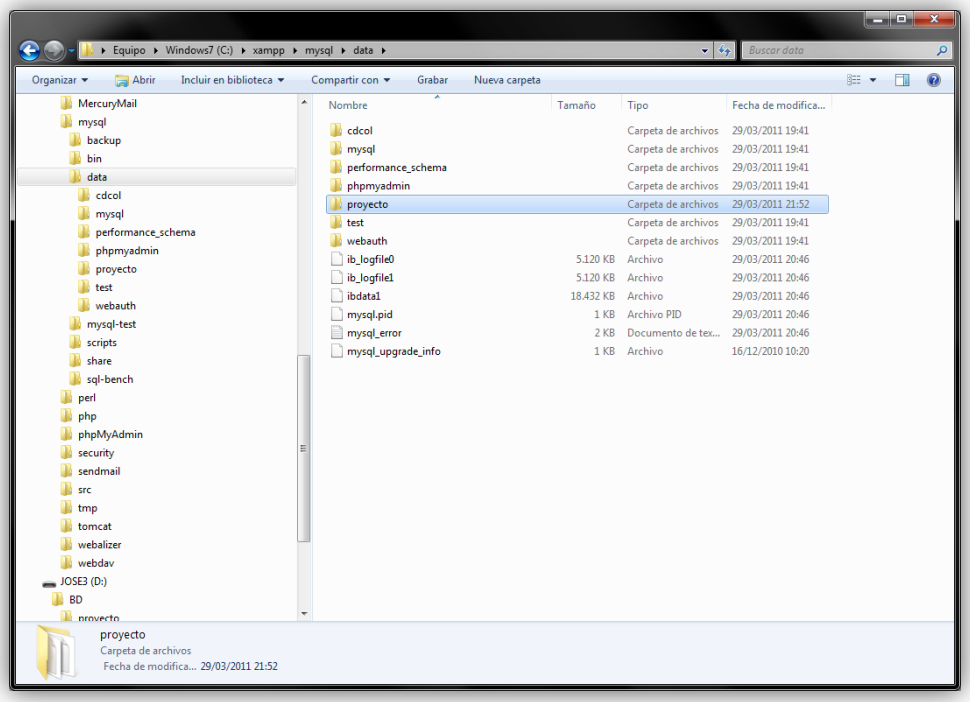


Figura A.15 –Copia de los archivos de la base de datos.

Los archivos de la aplicación web en sí se encuentran en la carpeta "Proyecto" debiendo copiarse en la ruta "C:\xampp\htdocs" (ver Figura A.16).

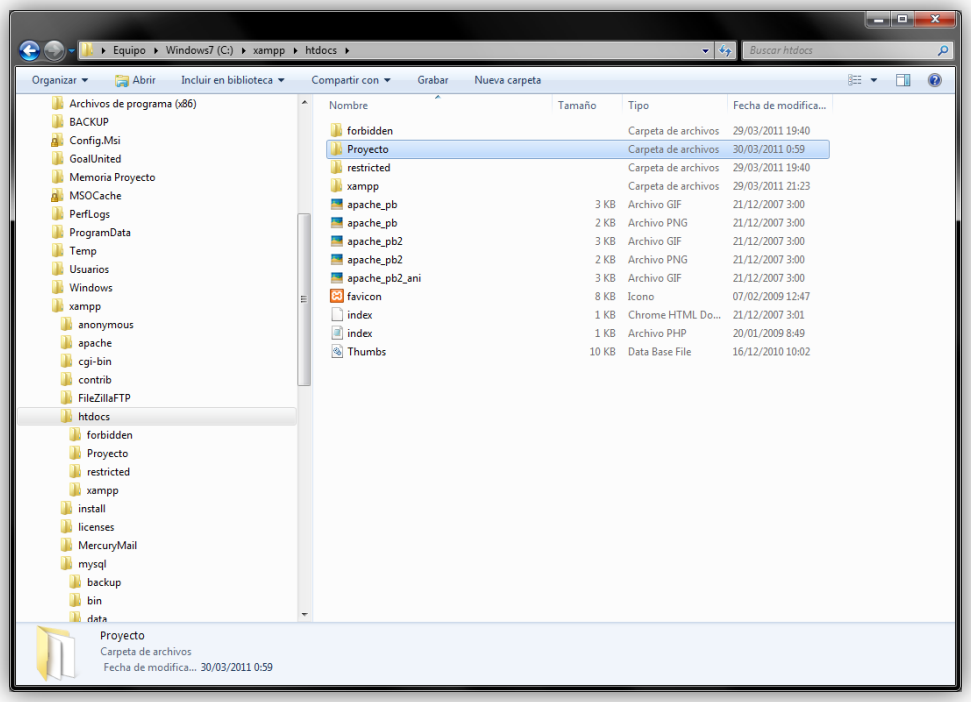


Figura A.15 –Copia de los archivos de la aplicación web.

Existen dos archivos básicos de configuración de la aplicación: Configuración.txt (situado en el directorio raíz del proyecto) y Conexion.php (situado en la carpeta CodigoPHP).

En el primero está establecida la variable que controla cuál es el servidor dónde se aloja la web y en el segundo los datos referentes a la conexión con la base de datos: el usuario y la contraseña genéricos de MySQL y la dirección del servidor donde se encuentra MySQL instalado. Si se desea hacer que funcione de forma local, todos los valores son los idóneos salvo el usuario y la contraseña para la conexión con la base de datos (en el archivo Conexion.php), que puede ser *root* y la contraseña que se ha configurado en el apartado A.3 u otro usuario que hayamos creado. Si queremos que funcione en modo red, se debe introducir en el primer archivo (Configuración.txt) la dirección IP completa de la máquina donde se ha instalado la aplicación y hacer también el cambio en el usuario y la contraseña dentro del archivo Conexion.php.

Una vez realizados estos ajustes, la web se deberá mostrar correctamente en nuestro navegador web introduciendo la dirección dónde se encuentre; si es en ámbito local, bastará con: http://localhost/proyecto/.

En siguiente apéndice se podrá consultar una guía a modo de manual de usuario acerca del funcionamiento completo de la aplicación.

Apéndice B – Manual de usuario

Esta sección tiene como objeto ser una guía de referencia para el usuario de la aplicación web que se ha desarrollado.

B.1. Inicio

Se podrá acceder a la página web inicial con un navegador compatible, es decir, cualquier navegador que tenga el *plugin* de Flash instalado. Tras la muestra de la barra de carga y la animación de la cabecera y el pié, se podrá observar un pequeño y típico formulario de identificación, dónde se deberá especificar nombre de usuario y contraseña. (Figura B.1)

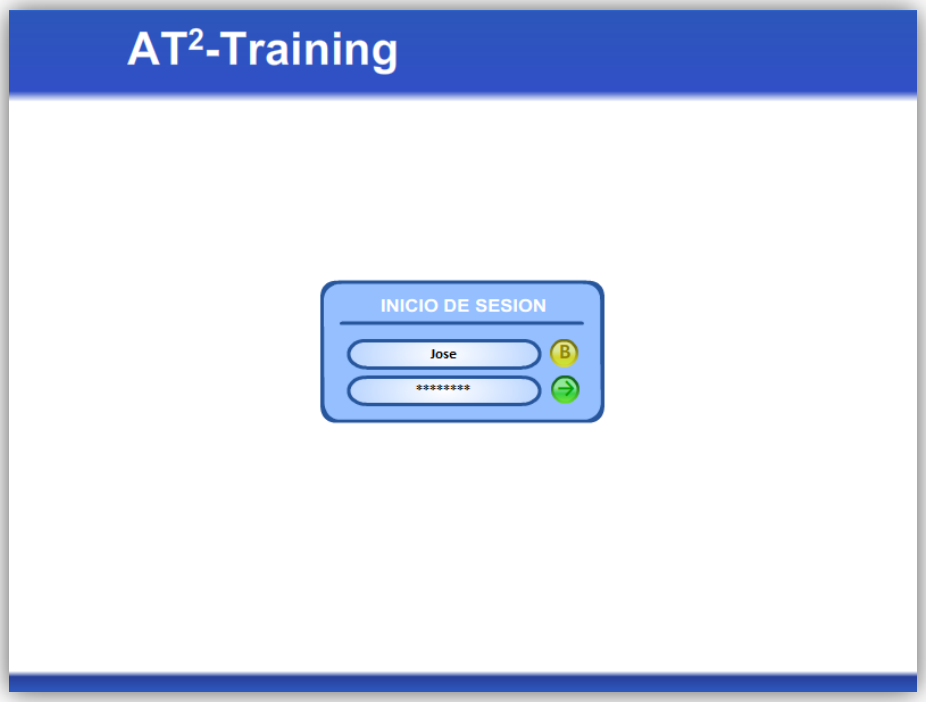


Figura B.1 – Inicio de sesión.

Para acceder a la web es necesario poseer ya un nombre de usuario puesto que no es una web pública. Existen distintos usuarios base (administradores) que deben dar de alta a nuevos usuarios que vayan a hacer un uso responsable de la web, asegurando siempre un correcto tratamiento de los datos que almacenan.

Una vez que se ha especificado el nombre de usuario y la contraseña se debe hacer clic en el botón verde para que la aplicación compruebe en la base de datos que es todo correcto y pueda acceder a la aplicación principal. El botón amarillo borra ambos campos.

Si el nombre de usuario o la contraseña fuesen incorrectos, se mostraría la pantalla de error, dando la posibilidad de cambiarlos y reintentar el acceso. No se ha separado el mensaje de error en dos ventanas distintas,(error en nombre de usuario y en contraseña independientes), puesto que esto haría posible que alguien ajeno a la aplicación probase distintas formas de acceso para intentar acceder.

Una vez que se ha accedido a la web, se podrá ver la pantalla principal de la aplicación con los datos del usuario con el que se ha accedido en la parte superior izquierda.

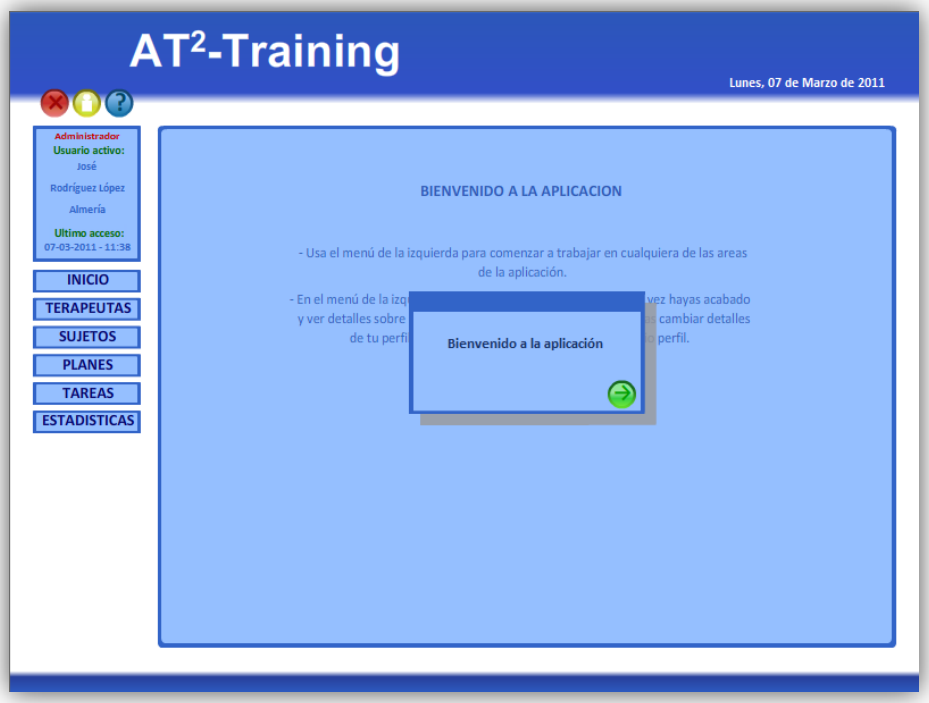


Figura B.2 – Pantalla inicial.

Como se puede observar en el recuadro de la información del usuario activo (Figura B.2), si el usuario es de tipo administrador, esto será mostrado con la palabra Administrador en rojo sobre su información, aparte de su nombre y apellidos y la fecha y hora del último acceso a la aplicación. En la siguiente sección se podrá ver las diferencias entre un usuario estándar y un usuario administrador

B.1.1. Administradores

Sólo existe una diferencia importante entre un usuario administrador y un usuario estándar para el manejo de esta aplicación. El usuario administrador podrá modificar, eliminar y agregar nuevos terapeutas a la aplicación. Será el que controle el acceso a la misma pudiendo resetear las contraseñas de otros usuarios y su información. También podrá establecer a un usuario estándar como administrador.

Un usuario que no sea administrador podrá realizar todos los cambios que desee en el resto de aspectos de la aplicación. Por lo tanto, todos los usuarios tienen la misma capacidad de modificación de sujetos, planes y sesiones, además de acceso a la prueba de tareas de entrenamiento atencional.

En el apartado 3.1, Administración de Terapeutas, se comentarán las diferencias de la interfaz en el caso de ser administrador y de no serlo.

B.2. Menú y submenú

Al acceder con nuestro nombre de usuario y contraseña se podrán observar dos partes de menús bien diferenciadas en la parte central izquierda y en la parte superior izquierda (Figura B.3).

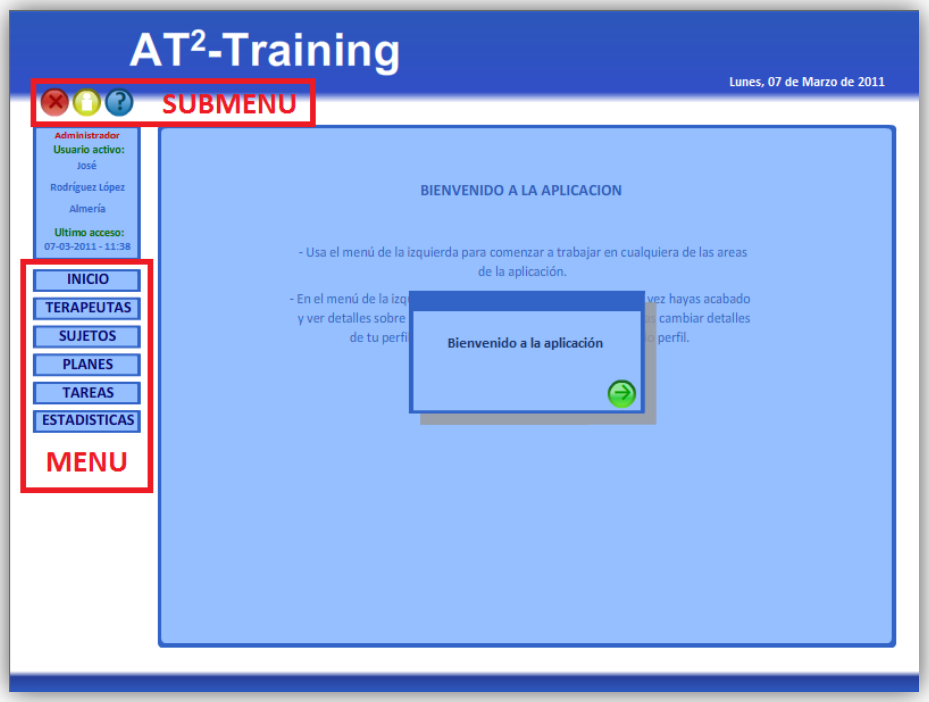


Figura B.3 – Menú y submenú.

B.2.1. Menú

El menú principal está dispuesto en la zona central izquierda. Desde ahí accederemos a las distintas partes de la aplicación para administrarla, hacer el seguimiento de los sujetos y lanzar las ejecuciones de sesiones y tareas.

A continuación describiremos el contenido de cada una de las secciones representadas en el menú principal:

- **INICIO:** Nos mostrará en la parte central de la aplicación la pantalla de bienvenida.
- **TERAPEUTAS:** En esta sección estará la pantalla de administración de terapeutas. Se muestra un listado de los que existen para mostrar información sobre ellos y editarlos o crear nuevos terapeutas. También se modificarán desde aquí los datos propios de cada terapeuta. Si éste es administrador además podrá modificar los del resto e incluso resetear las contraseñas.
- **SUJETOS:** Desde aquí se realizará la administración completa de los sujetos. Se crearán nuevos, modificarán y eliminarán. También será desde dónde se asignen planes a cada usuario y se realice su seguimiento, lanzando las sesiones que se van a realizar.
- **PLANES:** Para administrar los planes, crear, modificar y eliminar, deberemos entrar en esta sección. Por cada plan se podrán crear las sesiones que los compongan.
- **TAREAS:** En esta parte se podrán probar de forma independiente las tareas que los sujetos realizarán en las sesiones. Antes de hacer clic sobre la tarea que se quiera realizar se podrá seleccionar el tiempo y el nivel desde el que se quiere partir. De no cambiarse, la tarea se iniciará con los valores estándar de 1 minuto y desde el nivel 1.

B.2.2. Submenú

El submenú está colocado en la parte superior izquierda y se compone de tres botones:

- **ROJO – Cierre de sesión:** Este botón se usará para salir correctamente de la aplicación y nos mostrará una pequeña pantalla con dos botones: uno verde de confirmación de la acción de cierre de sesión y otro rojo para la cancelación de ese acción. Una vez aceptado el cierre de sesión nos mostrará la pantalla de inicio de sesión (Figura B.1).

- **AMARILLO – Información del usuario activo:** Al hacer clic sobre este botón se nos mostrará una ventana con la información correspondiente al usuario actual. Si el usuario desease cambiar o agregar parte de su información, deberá acceder a su usuario desde el menú Terapeutas.

- **AZUL – Ayuda:** Este botón nos mostrará una pantalla con un enlace a este manual en formato PDF que se abrirá en una nueva ventana o pestaña del navegador que estemos usando para la aplicación.

B.3. Administración de terapeutas

En este apartado veremos todas las pautas a seguir para introducir nuevos terapeutas o modificar y eliminar los existentes. Además se verán determinadas estadísticas sobre cada uno de los terapeutas y un listado de los sujetos que han creado.

Para entrar en esta sección deberemos hacer clic en el botón Terapeutas que se encuentra en el menú principal, en la parte izquierda de la pantalla de la aplicación. Al hacer clic sobre dicho botón nos aparecerá la siguiente imagen (Figura B.4).

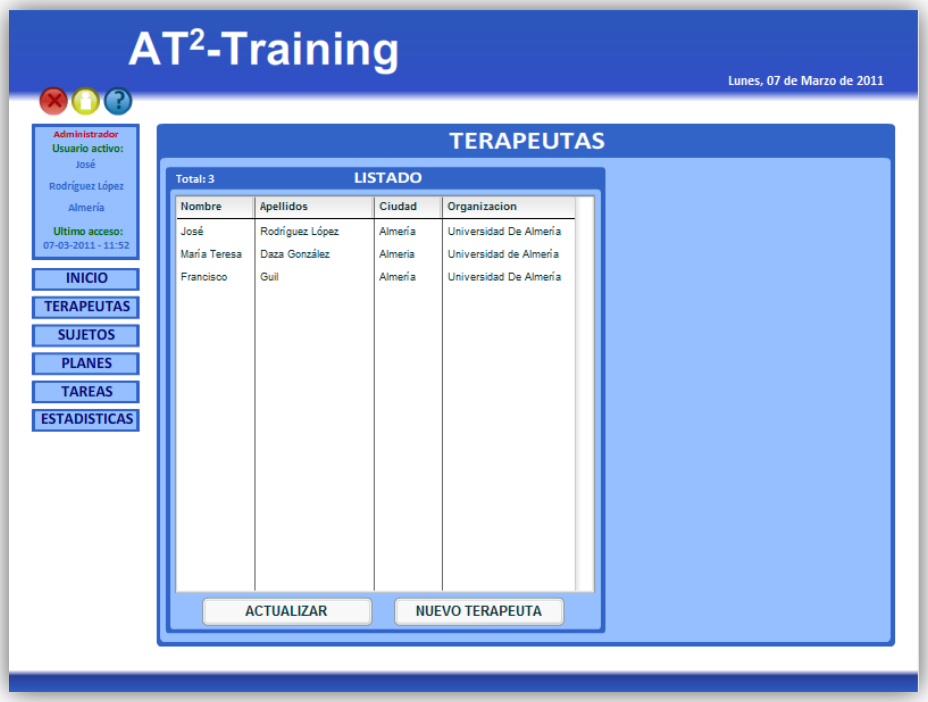


Figura B.4 – Vista general de terapeutas.

Como podemos observar en la imagen (Figura B.4), se mostrará un listado de todos los terapeutas activos y dos botones. Si hacemos clic en el botón de actualizar, la lista se actualizará, aunque se mostrará nueva información sólo si otro terapeuta o terapeutas han realizado cambios de forma simultánea desde otros equipos, ya que cada cambio que realicemos en nuestra sesión actualizará el listado de forma automática.

B.3.1. Creación de terapeutas

En la parte inferior derecha del listado de terapeutas encontramos el botón que nos permite introducir un nuevo terapeuta en la base de datos. Al hacer clic nos mostrará los campos de edición para introducir los datos del nuevo terapeuta (Figura B.5). Este botón permanecerá oculto si el usuario fuese estándar y no administrador, de ésta manera evitamos el uso excesivo de mensajes de error que se tendrían que mostrar si el usuario fuese estándar y quisiese agregar un nuevo terapeuta.

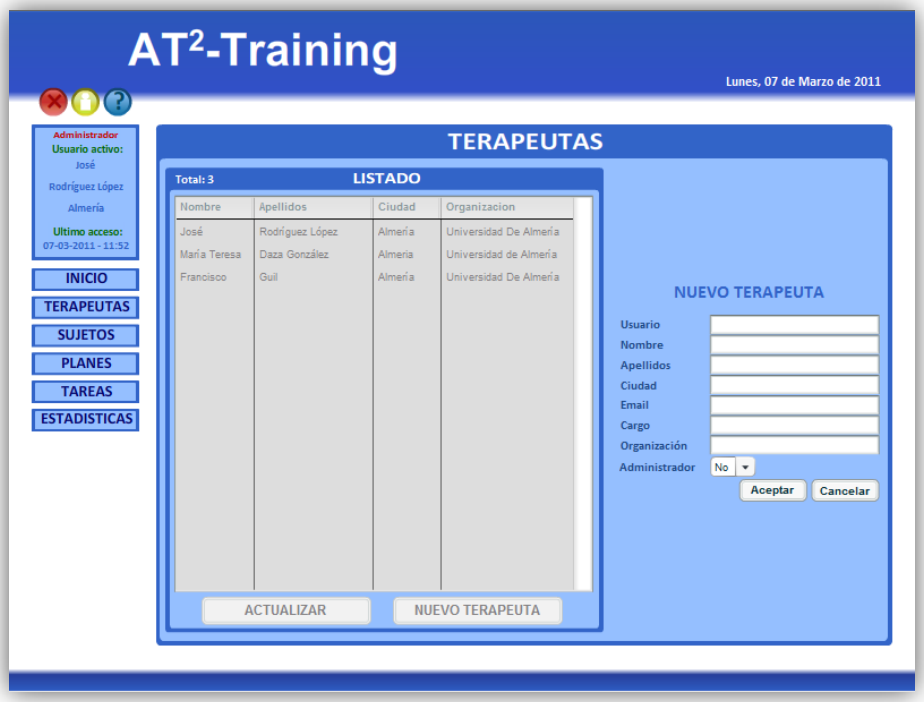


Figura B.5 – Formulario de nuevo terapeuta.

Como podemos observar en la imagen anterior (Figura B.5), al hacer clic sobre el botón de nuevo terapeuta no sólo se despliega el formulario de introducción de datos, también los botones y el listado de terapeutas se desactivan para que el usuario centre la atención sobre los nuevos campos y para evitar posibles errores derivados de una interfaz demasiado permisiva.

Existen determinadas restricciones en los campos a introducir cuando se va a crear un nuevo usuario:

- **Usuario**: Es obligatorio y debe tener una longitud mínima de 4 caracteres. Además debe ser único, por lo que si se intenta introducir uno ya existente, se mostrará un mensaje de error alertando del problema.
- **Nombre**: Es obligatorio, solo se permiten letras, mayúsculas y minúsculas, y debe tener una longitud mínima de 3 caracteres.
- **Apellidos**: Es de uso libre y solo se permiten letras, mayúsculas y minúsculas.
- **Ciudad**: Es de uso libre y solo se permiten letras, mayúsculas y minúsculas.

- **Email**: Es de uso libre.
- **Cargo**: Es de uso libre.
- **Organización**: Es de uso libre.
- **Administrador**: Sólo permite Si y No y ya está restringido por un campo de texto tipo Combobox.

En los campos que tienen restricciones de números, no todos permiten su escritura, directamente la aplicación evita escribirlos cuando es pulsada una tecla numérica, al igual que otros símbolos.

Como podemos observar, la contraseña de acceso a la aplicación no se introduce puesto que estamos creando el usuario de acceso de otra persona. Así pues, la contraseña por defecto será "1234" y deberá ser el propio usuario el que cambie la contraseña por la que desee cuando así lo crea oportuno.

Una vez hayamos rellenado los campos obligatorios y los que consideremos necesarios para identificar al nuevo terapeuta deberemos hacer clic en aceptar para que se hagan las últimas comprobaciones y así almacenar los datos en la base de datos.

Por defecto, una vez creado el nuevo usuario se mostrará un mensaje informativo y se actualizará la lista de terapeutas mostrando el nuevo que se ha creado.

B.3.2. Vista, modificación y eliminación de usuarios

Una vez hemos accedido al apartado de Terapeutas de la aplicación y estemos observando el listado de los que hay actualmente (Figura B.4), para obtener una vista de cualquiera de ellos sólo deberemos hacer clic en su línea.

Como nota, resaltar que el listado es totalmente dinámico y permite ordenar alfabéticamente, de forma tanto ascendente como descendente, por cualquiera de las cabeceras (Nombre, Apellidos, Ciudad u Organización).

Habiendo hecho clic en un terapeuta del listado se mostrará la información disponible de éste y las acciones disponibles a realizar sobre él, éstas dependerán de si el usuario es administrador (Figura B.6) o no lo es (Figura B.7).

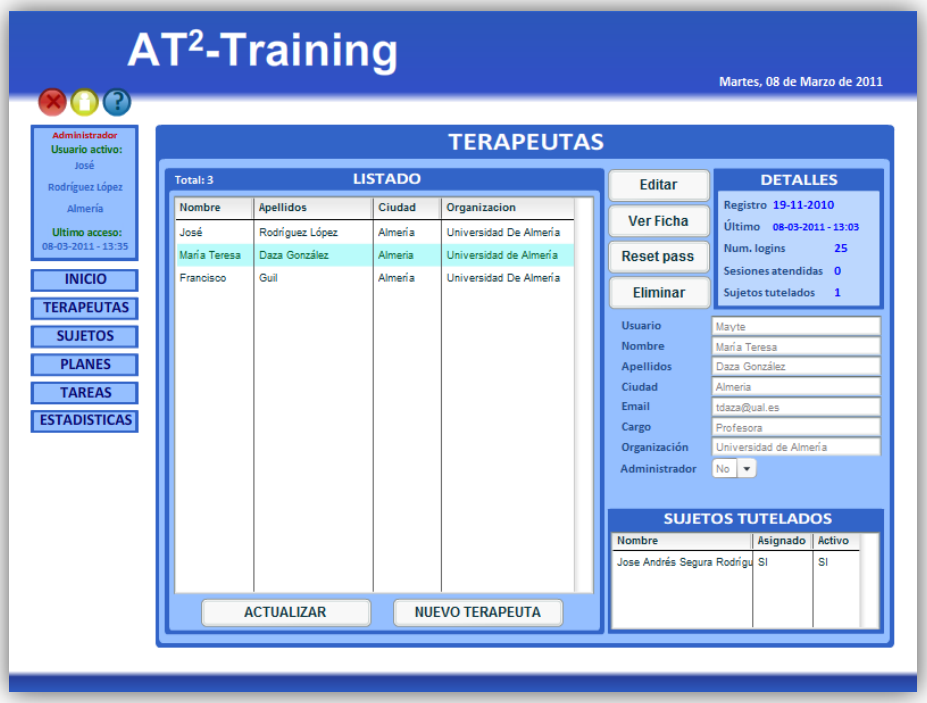


Figura B.6 – Vista de un Terapeuta con usuario administrador.

En la Figura B.6. Vemos las acciones que tendría disponibles el usuario actual, administrador, en el caso de hacer clic sobre otro terapeuta. Como se puede observar, puede editarlo, ver su ficha de estadísticas (la aplicación saltará al apartado de estadísticas y mostrará ahí su ficha), resetear su contraseña (siempre por defecto a "1234") y eliminarlo.

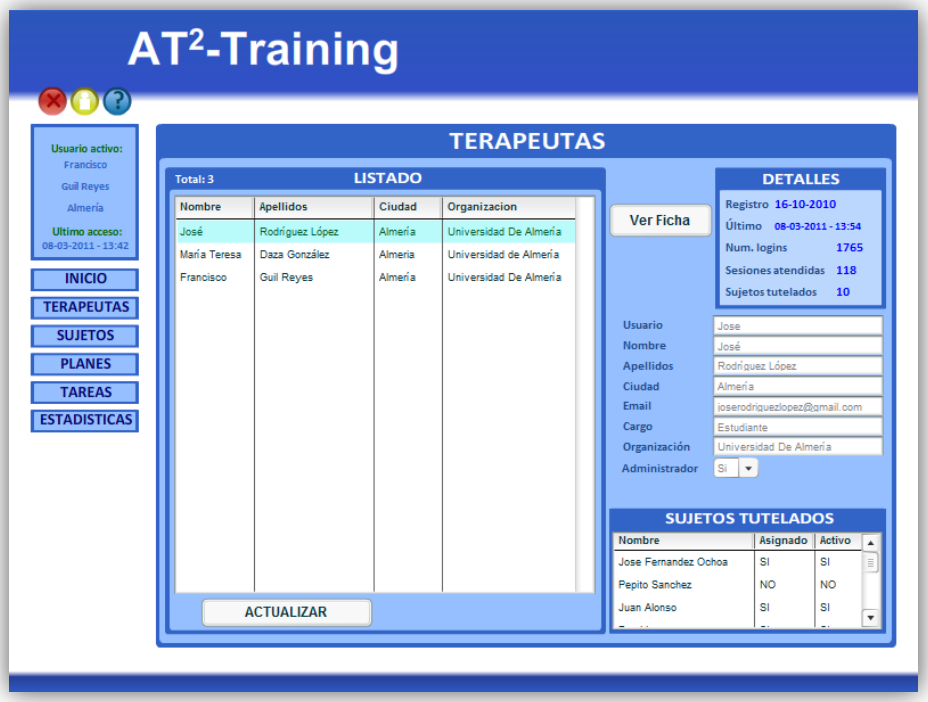


Figura B.7 – Vista de un Terapeuta con usuario estándar.

En la Figura B.7. Vemos las acciones que tendría disponibles el usuario actual, en este caso no es administrador, cuando hace clic sobre otro terapeuta. Como se puede observar, sólo puede hacer clic para ver su ficha de estadísticas y observar toda la información, que es independiente del tipo de usuario que sea.

Los detalles sobre la información del usuario son visibles arriba a la derecha, en ese recuadro se puede observar la fecha de registro, su último acceso a la aplicación, el número de veces que ha accedido, las sesiones atendidas a sujetos y el número de sujetos que ha creado, que tutela.

El listado que aparece abajo a la derecha es una relación de todos los sujetos que ese terapeuta ha creado con nombre completo y si tienen un plan asignado y comenzado.

Cuando un usuario se selección a sí mismo (Figura B.8), independientemente de que sea estándar o administrador, puede cambiar su contraseña, editarse y ver su ficha. En el caso de que sea administrador, el botón de eliminar se desactiva, puesto que no

es posible eliminarse uno mismo de la aplicación, evitando así posteriores mensajes de error.

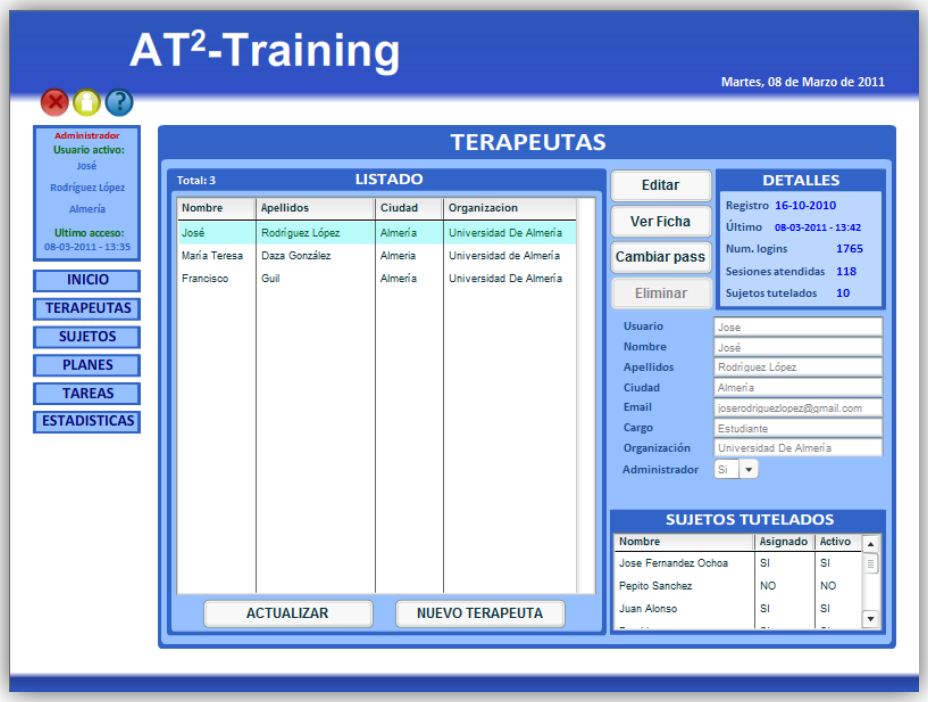


Figura B.8 – Vista propia de un Terapeuta.

Cuando queremos editar un terapeuta solo debemos hacer clic en el botón de editar y así se activarán los campos correspondientes a editar. No será posible editar el nombre de usuario único y las reglas de edición de los campos son las mismas que cuando se crea un terapeuta nuevo (Ver apartado B.3.1. Creación de terapeutas).

Por razones de seguridad, si el usuario activo es estándar, al editar su propia información no podrá establecerse a sí mismo como usuario administrador. El campo combobox continuará desactivado en ese caso.

Si el botón eliminar estuviese activo (el usuario activo es administrador y no se ha seleccionado a sí mismo), podremos eliminarlo sólo el terapeuta a eliminar no ha realizado cambios en la base de datos, es decir, no ha creado sujetos, planes o sesiones. De esta manera nos aseguramos de mantener una información estadística sólida sobre la base de datos.

En el caso de querer cambiar la contraseña aparecerá un nuevo formulario en la zona de detalles como se puede observar en la Figura B.9.

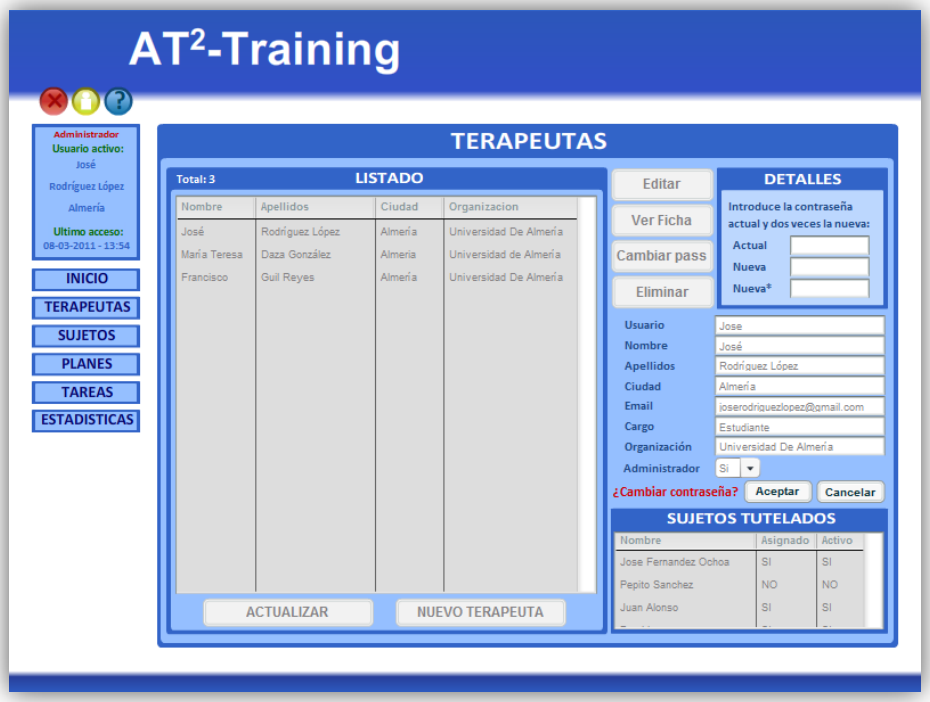


Figura B.9 – Formulario de cambio de contraseña.

Para realizar correctamente un cambio de contraseña se debe escribir la actual y dos veces la nueva para evitar errores de escritura. La contraseña debe tener una longitud mínima de 4 caracteres y su composición es libre.

B.4. Administración de sujetos

En este apartado de la aplicación tendremos la posibilidad de realizar sobre los sujetos todas las acciones necesarias para administrarlos y hacerles el seguimiento del plan que estén siguiendo.

Antes de continuar, debemos aclarar que no hay diferencias de administración de sujetos siendo los usuarios administradores o no, a partir de este punto todas las acciones son posibles por todos los usuarios.

Una vez hayamos hecho clic en el botón de Sujetos del menú principal pasaremos a ver el listado total de sujetos (Figura B.10).

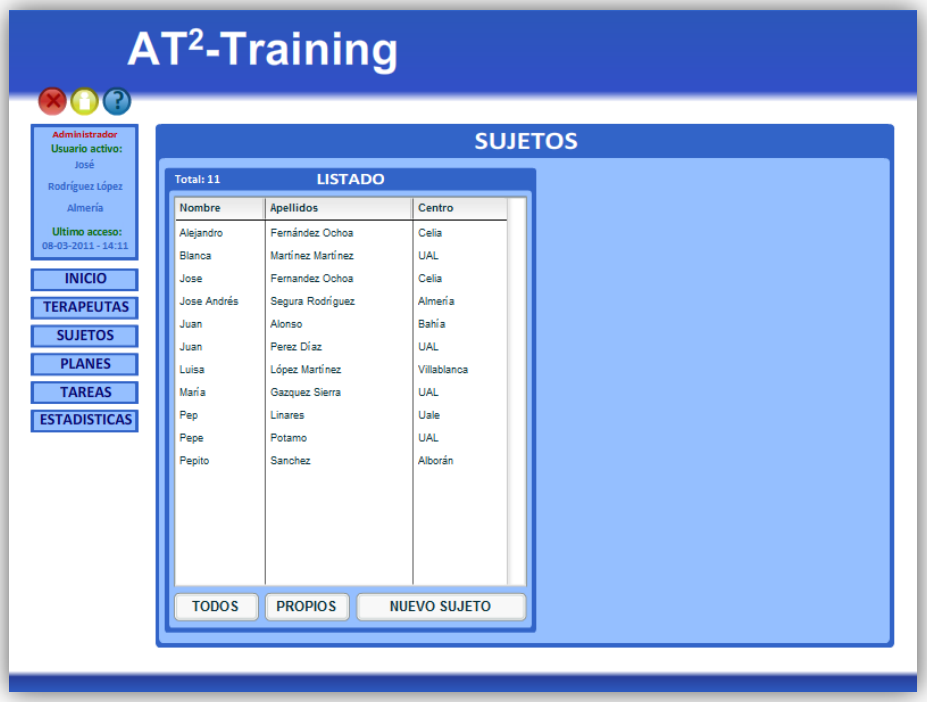


Figura B.10 – Vista general de sujetos.

En este caso tendremos tres botones disponibles, dos de ellos para el filtrado del listado y el tercero para abrir el formulario de creación de un nuevo sujeto.

El botón de filtrado Todos nos mostrará la lista de todos los sujetos que se encuentran en la base de datos y el botón propios nos mostrará el listado de los sujetos que ha creado el usuario activo de la sesión.

B.4.1. Creación de sujetos

Al hacer clic en el botón de Nuevo Sujeto, aparecerá el formulario de creación del nuevo sujeto, muy similar al de creación de un nuevo terapeuta. A la vez que se muestra el formulario se desactivarán los botones y el listado. Podemos ver cómo quedaría en la siguiente figura. (Figura B.11).

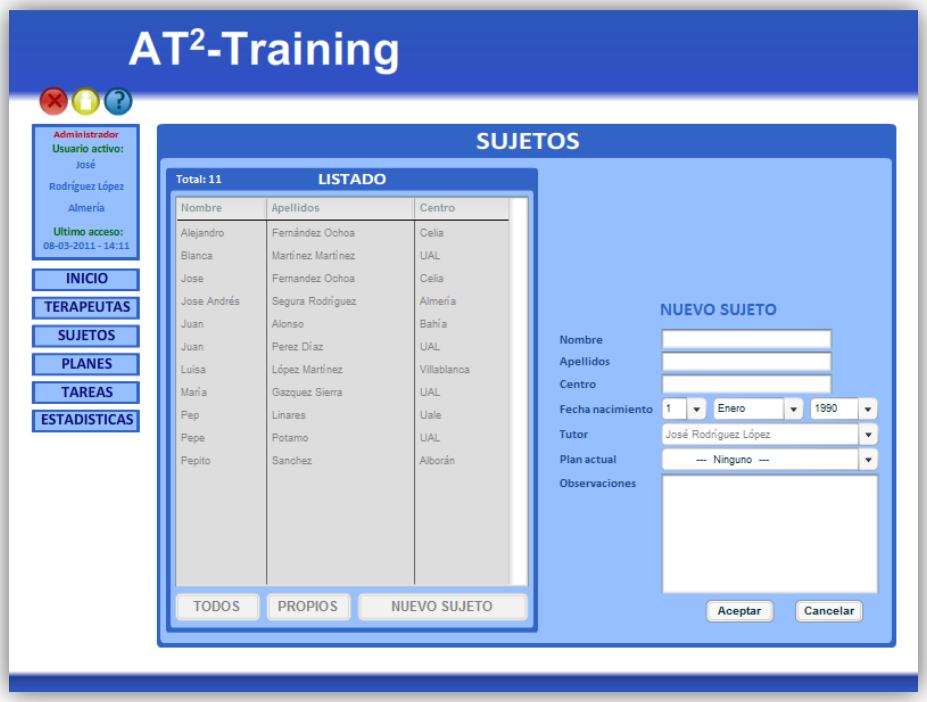


Figura B.11 – Formulario de nuevo sujeto.

Existen determinadas restricciones en los campos a introducir cuando se va a crear un nuevo usuario:

- **Nombre**: Es obligatorio, solo se permiten letras, mayúsculas y minúsculas, y debe tener una longitud mínima de 2 caracteres.
- **Apellidos**: Es de uso libre y solo se permiten letras, mayúsculas y minúsculas.
- **Centro**: Es de uso libre.
- **Fecha de nacimiento**: Se debe ajustar a los campos que hay y además debe ser una fecha correcta. La aplicación alertará si se introduce una fecha incorrecta (Por ejemplo, 31 de Febrero).
- **Tutor**: Es un campo fijo dependiente del usuario activo que esté creando al sujeto.
- **Plan**: Se limita a la elección de uno de los planes que hayan sido validados (Ver apartado B.5.3 Validación de un plan).

- **Organización**: Es de uso libre.

Una vez se hayan introducido los datos que se deseen, sólo debemos darle a aceptar. Si todos los datos son correctos se actualizará correctamente la base de datos y se mostrará un mensaje informativo a la vez que se actualiza el listado de los sujetos existentes.

B.4.2. Vista, modificación y eliminación de sujetos

Una vez estamos dentro de la sección de sujetos (Figura B.10), solamente debemos hacer clic sobre la línea del sujeto del que queramos consultar su información como muestra la siguiente figura (Figura B.12).

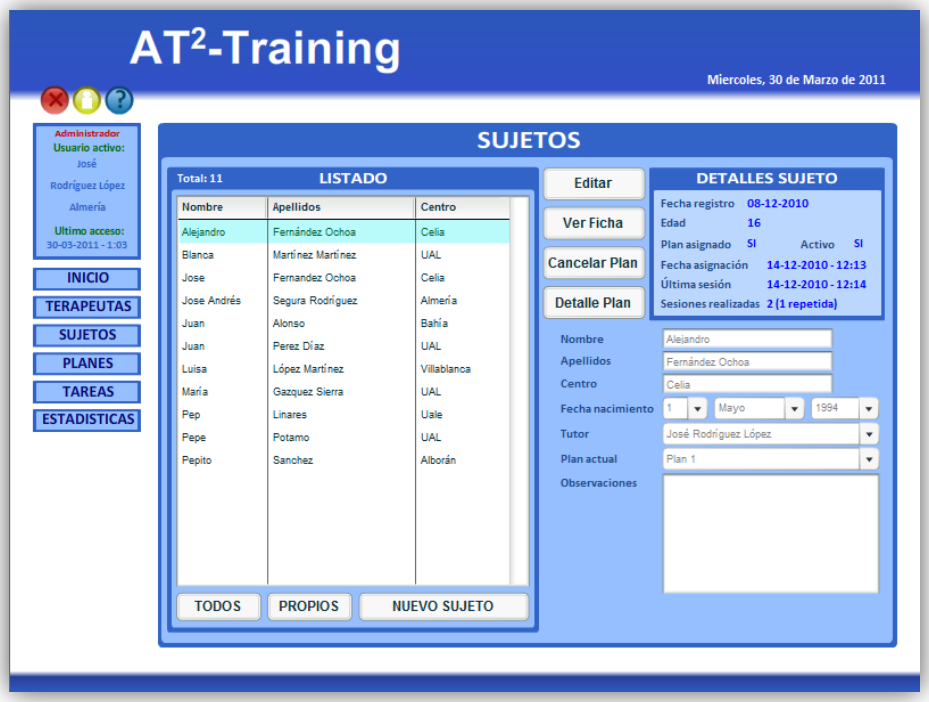


Figura B.12 – Vista de un sujeto con plan asignado.

Como podemos observar en este caso el sujeto ya tiene un plan asignado y que ya está activo, es decir, que ya se realizó al menos una sesión. En detalles se puede ver la fecha de registro, la edad, si tiene un plan asignado, activo y los detalles de éste.

Además, se habilita la interacción con los botones para la administración de los sujetos. Según esté el sujeto con un plan asignado o no, se verán unos botones u otros. En la Figura B.12 vemos un sujeto con plan asignado y los botones Editar, Ver ficha, Cancelar plan y detalle plan activados. Si este mismo sujeto no tuviese un plan asignado el botón Cancelar plan estaría deshabilitado y el botón detalle plan sería el de eliminar (Figura B.13). No obstante, dependiendo del caso en el que se encuentre un sujeto se puede o no eliminar. Si un sujeto ya tiene estadísticas de planes y sesiones no se puede eliminar para mantener la máxima información disponible en la base de datos.

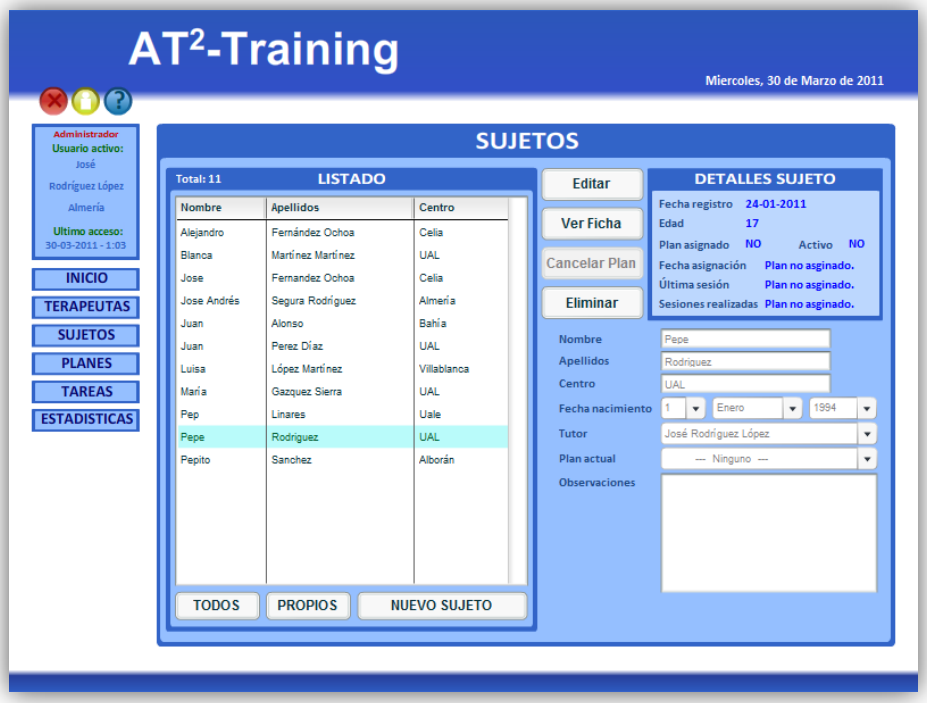


Figura B.13 – Vista de un sujeto sin plan asignado.

Para la edición o modificación de los datos de un sujeto sólo debemos hacer clic en el botón Editar y tendremos habilitados todos los campos. La edición de los campos se rige por las mismas normas que la creación de un sujeto nuevo.

Si se edita un sujeto con plan asignado, éste no podrá ser modificado. Para cambiar el plan de un sujeto antes se debe terminar el plan, superando todas las sesiones, o cancelarlo perdiendo la posibilidad de acabarlo posteriormente.

B.4.3. Asignación de plan

Para asignar un plan debemos editar el sujeto y seleccionar el plan que deseemos que siga dicho sujeto. Para ello debemos seleccionarlo con el combobox habilitado a tal efecto. Hay que tener en cuenta que para que el plan sea seleccionable debe ser validado con anterioridad. Más adelante veremos cómo validar un plan y las restricciones que tiene el hacerlo (Apartado B.5.3. Validación de un plan).

B.4.4. Seguimiento de un plan

Una vez hemos asignado un plan como hemos visto en el anterior apartado, podemos ver la evolución del plan actual haciendo clic en el botón Detalle Plan. Una vez hecho esto nos aparecerá una vista parecida a la siguiente imagen (Figura B.14).

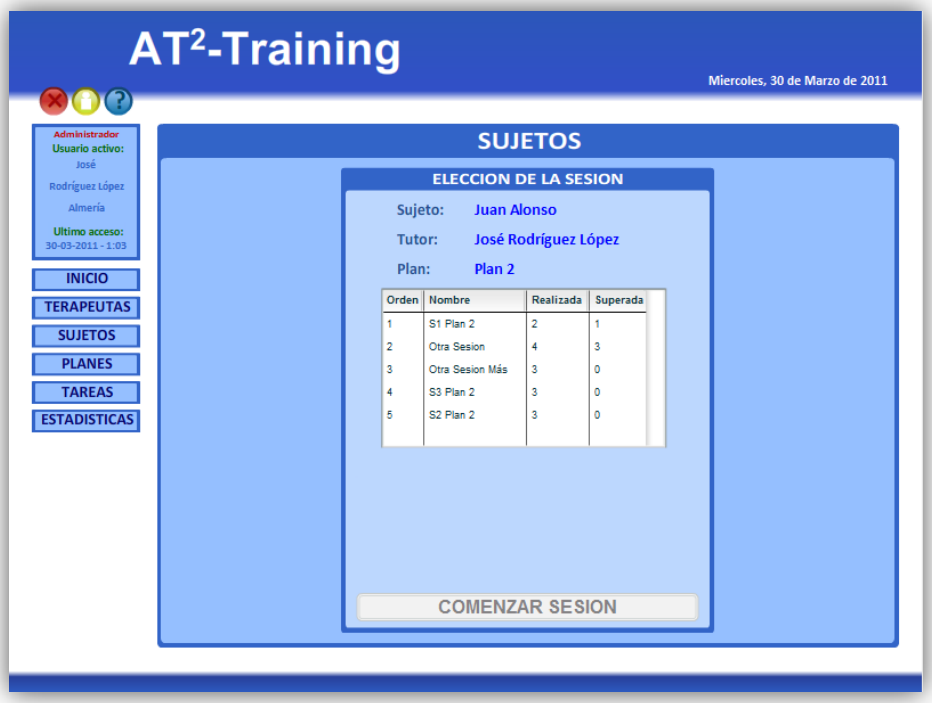


Figura B.14 – Vista del detalle del plan de un sujeto.

Aquí podemos observar (Figura B.14) el nombre del sujeto actual, su tutor y el nombre del plan que está realizando. Además, en el listado vemos las sesiones de las que está compuesto el plan, el número de veces que se han realizado cada una de ellas y el número de veces que han sido superadas. También aparece el botón Comenzar

sesión deshabilitado, para habilitarlo será necesario seleccionar una de las sesiones disponibles.

Si todas las sesiones estuviesen finalizadas con éxito al menos una vez, aparecería un botón de finalización de plan que nos da la posibilidad de dar por finalizado ese plan.

Para ver la progresión actual de cada sesión y para poder lanzar una de ellas, debemos seleccionarla del listado. Una vez está seleccionada podemos ver qué tareas la componen, los puntos máximos obtenidos en una sesión de cada una de ellas, el nivel máximo alcanzado y el número de veces que la ha superado (Figura B.15).

Esta información es muy relevante para conocer en qué tareas el sujeto se desenvuelve mejor y si hay alguna que es especialmente difícil para él.

Figura B.15 – Vista del detalle de una sesión de un sujeto.

B.4.5. Comienzo de una sesión

Estando dentro del detalle del plan y habiendo seleccionado una de las sesiones de las cuales el plan esté compuesto, se habilitará el botón comenzar sesión (Figura B.15). Para realizar esa sesión solamente deberemos hacer clic en el botón

Comenzar Sesión, seleccionar el tiempo que queremos que dure dicha sesión, confirmar el comienzo y ceder el puesto al sujeto para que la realice (Figura B.16).

La duración de la sesión se da en minutos y debe ser un número múltiplo de tres. Para evitar errores de introducción de datos, la duración se selecciona en un desplegable con valores desde un mínimo de 3 minutos a un máximo de 90.

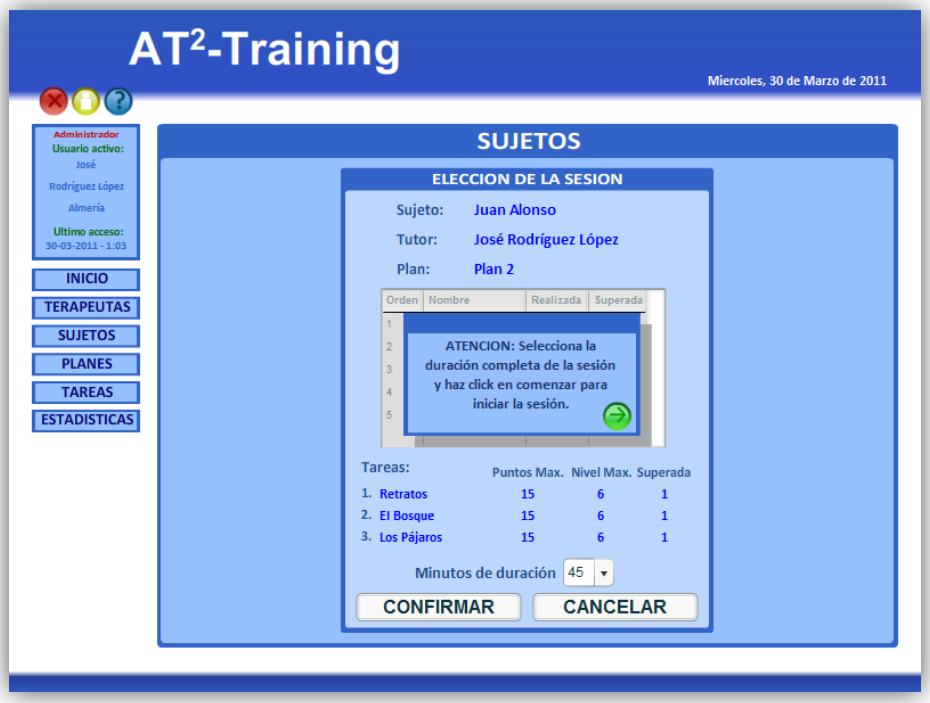


Figura B.16 – Lanzamiento de una sesión.

B.4.6. Funcionamiento básico de una sesión

Una vez hemos lanzado la sesión, ésta mostrará una pantalla de descripción de la primera tarea, el nivel en el que comenzará y la duración máxima de la misma en minutos (Figura B.17). Tras la lectura, haremos clic en Comenzar para que comience la tarea descrita.

Toda sesión debe estar compuesta por tres tareas, más adelante en este mismo manual veremos cómo configurar planes y sesiones, por ello el tiempo total de la sesión se reparte entre las tres tareas.

Para superar una tarea deberemos superar los 6 niveles de dificultad, dependiendo de la tarea, los niveles se superan con una serie de requisitos variables. Si

una tarea es superada antes del tiempo límite, se pasará a la siguiente tarea. El tiempo que resta se acumulará para ser usado al final, bien para superar tareas no superadas o para repetir las superadas y acumular más puntos extra. Obtendremos un punto por cada nivel superado.

Figura B.17 – Descripción y lanzamiento de una tarea.

Una vez ha finalizado el tiempo total de la sesión, la tarea en curso se da por finalizada mostrando un mensaje de que el tiempo acabó y salta a una pantalla mostrando un resumen completo de la sesión. Ahí podremos ver los puntos obtenidos, nivel alcanzado y tiempo empleado en cada tarea (Figura B.18). El tiempo base es el tiempo que se ha estado por primera vez en la tarea y el extra el tiempo dedicado una vez se ha pasado por las tres primeras tareas.

En dicha pantalla sólo hay botón, Salir, que hará que la aplicación vaya directamente al formulario de login. Esto se ha decidido así porque el equipo está en manos del sujeto cuando acaba la sesión y éste no debe usar la interfaz con otro fin que no sea el de realizar las sesiones.

Figura B.18 – Resumen de sesión.

B.4.7. Cancelación de una sesión

Como hemos visto con anterioridad, para que a un sujeto se le pueda asignar un plan, éste no debe tener uno ya asignado. Si ya tiene un plan asignado es necesario acabar el actual superando todas las sesiones o cancelando el actual.

Como es obvio, si se cancela un plan, no se puede volver a continuarlo. En las estadísticas aparecerá como que el plan ha sido cancelado y se deberá comenzar uno nuevo, el mismo u otro, pero desde cero.

Dada la importancia de la cancelación de un plan, la aplicación mostrará un mensaje de advertencia y necesitará de una confirmación de la acción.

B.5. Administración de planes y sesiones

En este apartado veremos cómo realizar la configuración de un plan y la inclusión de sesiones. Aunque más adelante se repetirá, los planes y las sesiones de dichos planes no podrán ser modificados una vez el plan haya sido asignado, y por

tanto comenzado, a un sujeto. Esto es de suma importancia, puesto que si se pudiese modificar tras el uso del plan, los datos de las sesiones realizadas con anterioridad no serían congruentes con los que hubiese después de haber hecho el cambio. Por tanto, si un plan se decidiese cambiar una vez usado, no se podría, se debería de crear un nuevo plan similar o no.

La primera pantalla que visualizamos cuando hacemos clic en el botón de planes del menú principal es parecida a la de sujetos y terapeutas, pero esta vez con un listado de planes existentes (Figura B.19).

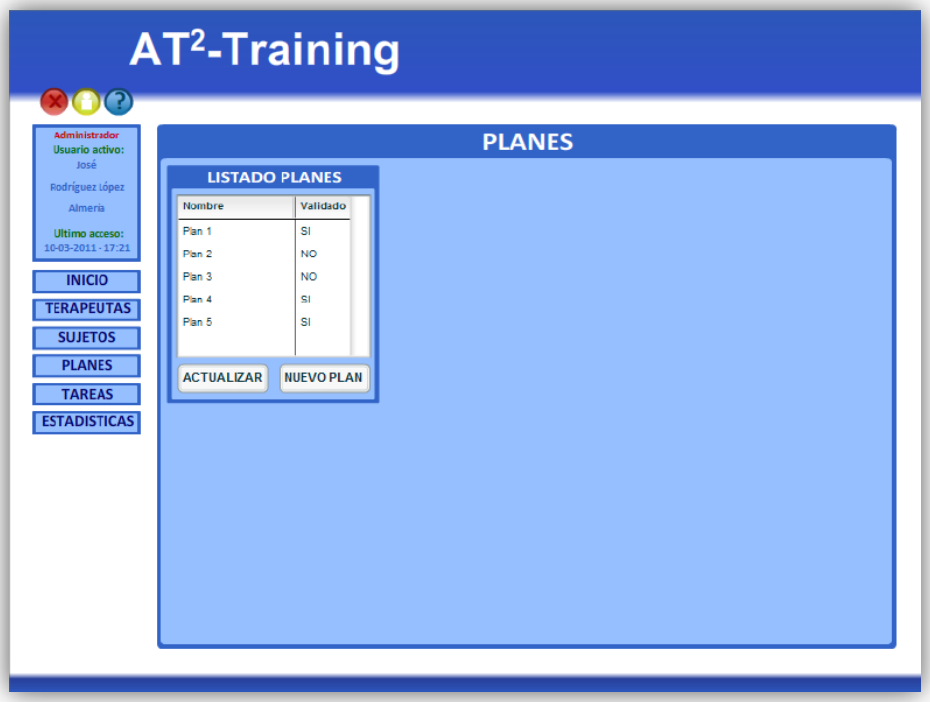


Figura B.19 – Vista general de planes.

Como podemos observar en la figura anterior, en el mismo listado, a la derecha del nombre de cada plan, aparece si éste está validado o no.

En cuanto a los botones, solo dos, actualizar, para actualizar la tabla y nuevo plan, para crear un nuevo plan desde cero.

B.5.1. Creación de un plan

Para crear un plan solamente debemos hacer clic sobre el botón Nuevo Plan y nos aparecerá el formulario de creación de un plan (Figura B.20). En la figura podremos observar que requerimos de menos información para crearlo que un sujeto o un terapeuta. Los campos a rellenar son solamente tres y el cuarto, el de validación, está deshabilitado: esto es así en base a una serie de requisitos que veremos en el apartado B.5.5. Validación de un plan. Los campos a rellenar son:

- **Nombre**: Es obligatorio y debe tener una longitud mínima de 2 caracteres.
- **Descripción**: Es de uso libre.
- **Observaciones**: Es de uso libre.

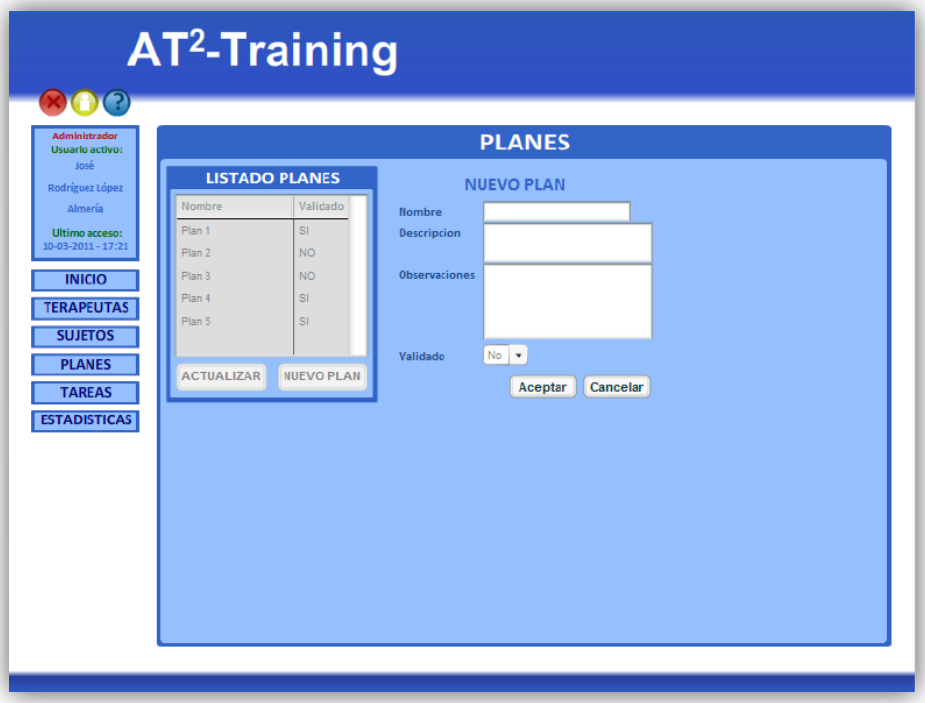


Figura B.20 – Formulario de creación de un nuevo plan.

Una vez rellenos los campos que se deseen, solamente debemos hacer clic en aceptar y tras aparecer un mensaje de confirmación de que se ha creado correctamente el nuevo plan, se actualizará el listado mostrando el nuevo.

Para agregar sesiones al plan que hemos creado deberemos seleccionar el plan y agregarle sesiones. Esto lo veremos en el apartado B.5.3. Creación de una sesión.

B.5.2. Vista, modificación y eliminación de un plan

Una vez hemos seleccionado el plan que deseamos visualizar en el listado nos aparecerá una vista de sus datos: datos básicos del plan, detalles del plan y las sesiones que lo componen (Figura B.21).

En los detalles veremos la fecha en la que se registró el plan, el terapeuta que lo creó (aunque cualquier terapeuta puede modificarlo y agregarle sesiones) y los sujetos que tienen el plan asignado, tanto el numero como el listado. En el listado la columna T representa el total de veces que el sujeto ha realizado ese plan y si lo tiene activo actualmente.

Para modificar un plan solo es necesario hacer clic en editar plan y los campos se habilitarán para su modificación y posterior registro una vez hayamos aceptado los cambios. Al editar si podremos colocar el plan como validado siempre que cumpla los requisitos que veremos en el apartado B.5.5. Validación de un plan. También está disponible la opción de eliminar el plan, aunque en este caso se ve deshabilitado porque ya hay al menos un sujeto haciendo uso de este plan. Sólo podremos eliminar aquellos planes que no tengan datos de sujetos registrados.

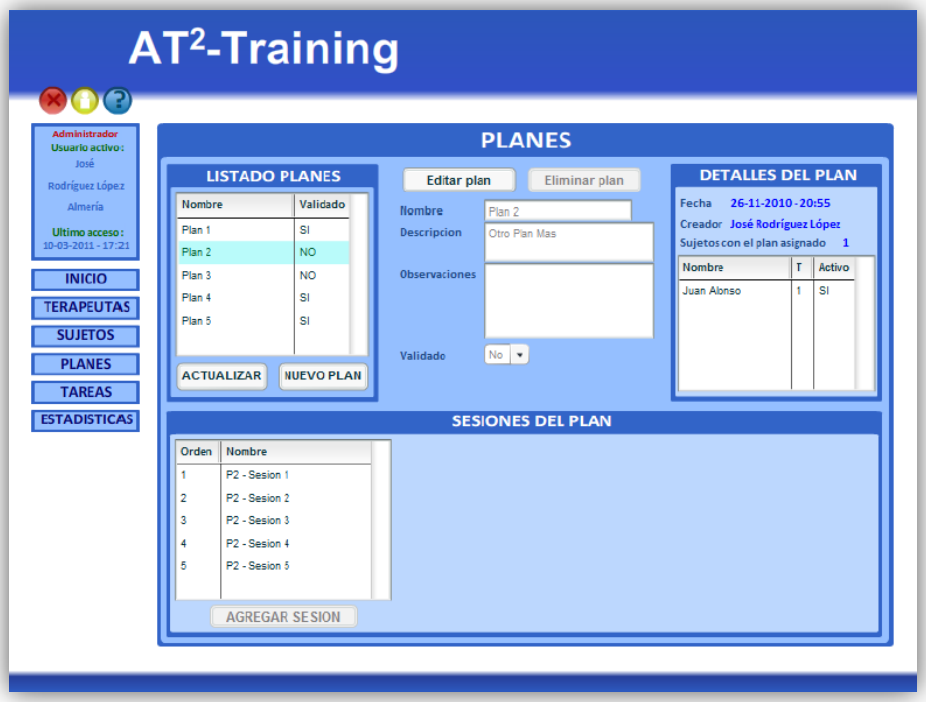


Figura B.21 – Vista detallada de un plan.

B.5.3. Creación de una sesión

Para agregar una sesión a un plan basta con darle al botón Agregar Sesión, que aparece en el listado de sesiones una vez hemos hecho clic sobre alguno de los planes. Debemos hacer hincapié en que si un plan ya ha sido asignado a sujetos, no se podrán agregar nuevas sesiones. Prueba de ello es el plan de la figura B.21, el botón de Agregar Sesión está deshabilitado.

Tras clicar en el botón Agregar Sesión se desplegará el formulario de creación de la sesión (Figura B.22).

Como observamos, el formulario consta de siete campos, tres de ellos iguales a los usados para crear un plan y los otros para establecer el número de orden dentro del plan (por defecto consecutivo al último o al hueco que exista) y tres para cada una de las tareas de las que se compondrá la sesión.

- **Nombre**: Es obligatorio y debe tener una longitud mínima de 2 caracteres.
- **Descripción**: Es de uso libre.

- **Observaciones**: Es de uso libre.
- **Número de orden**: A elegir entre el siguiente disponible o cero, para poder cambiar el orden de las existentes.
- **Tarea 1**: A elegir de una de las tareas disponibles sin repetirse con otra tarea de la misma sesión.
- **Tarea 2**: A elegir de una de las tareas disponibles sin repetirse con otra tarea de la misma sesión.
- **Tarea 3**: A elegir de una de las tareas disponibles sin repetirse con otra tarea de la misma sesión.

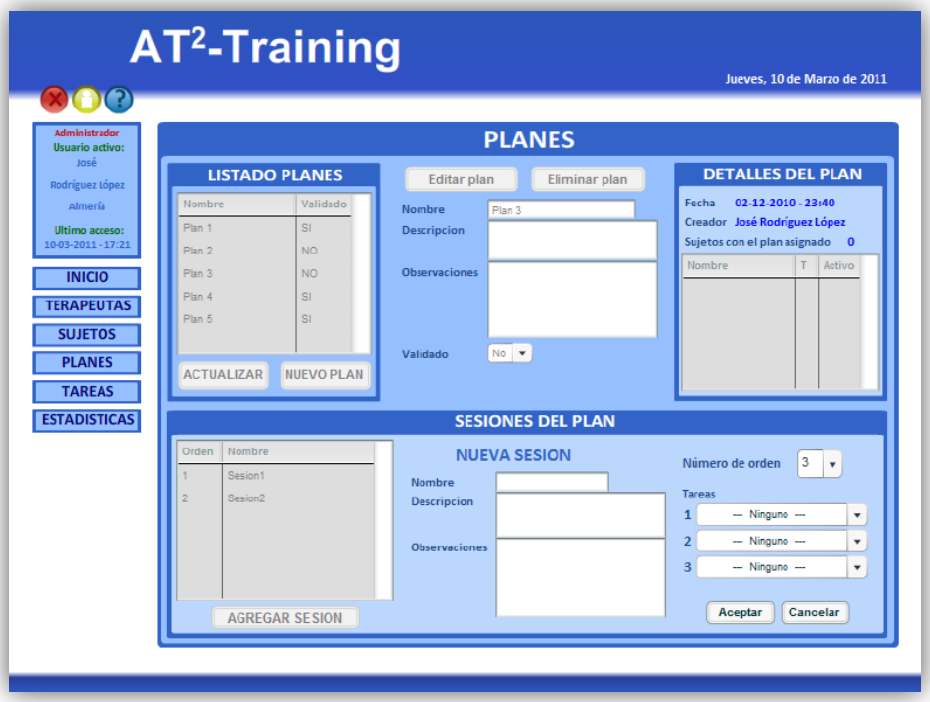


Figura B.22 – Formulario de creación de sesión.

Para que sea válida la sesión se ha de tener en cuenta que las tres tareas deben ser elegidas, no vale elegir solo dos, y todas ellas distintas.

Una vez hecho todo esto, la sesión será guardada y mostrada en la lista en el orden que le hayamos marcado.

B.5.4. Vista, modificación y eliminación de una sesión

Una vez estamos en la vista de detalle de un plan (Figura B.21) podemos hacer clic para seleccionar cada una de las sesiones que componen dicho plan (Figura B.23).

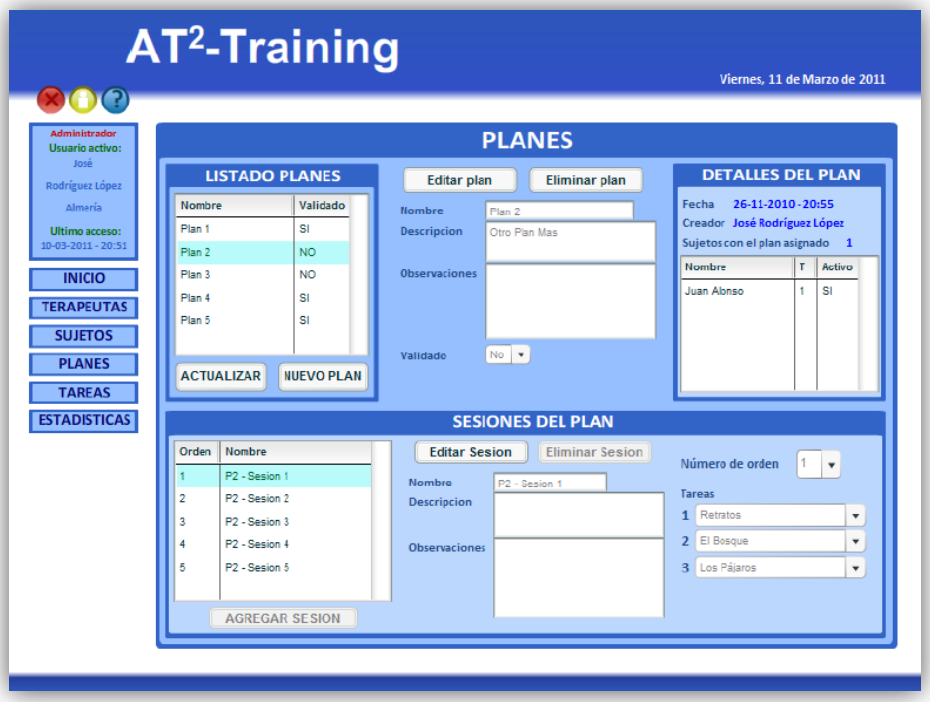


Figura B.23 – Vista de detalle de una sesión.

Para poder editar una sesión tenemos la misma restricción que con los planes, es decir, las sesiones cuyo plan ha sido asignado, no podrán editarse, salvo las observaciones. En el campo de observaciones podremos introducir los cambios que queramos.

Para eliminar la sesión sucede exactamente lo mismo que con la edición, así pues no podremos eliminar las sesiones que participen en planes ya asignados.

B.5.5. Validación de un plan

La validación de un plan es el paso último a la creación y edición de un plan. Cuando un plan está validado pasa a estar disponible para ser asignado a un sujeto, si el plan no está validado no aparecerá en dicha lista.

Para que un plan sea posible marcarlo como validado debe tener al menos una sesión válida y en un orden correcto. Las sesiones que compongan el plan deben empezar con número de orden uno y acabar con un número de orden igual al número total de sesiones que tenga el plan.

A la hora de realizar un plan, las sesiones aparecerán con el orden fijo que se haya elegido, pero se podrán realizar en el orden que se desee.

Es importante acabar correctamente el plan antes de validarlo y posteriormente asignarlo a un sujeto puesto que, como ya se ha remarcado, el plan no podrá ser modificado con posterioridad.

B.6. Tareas

Existen diez tareas distintas para formar las sesiones de trabajo de los sujetos. Cada una de ellas juega un papel distinto en el entrenamiento atencional del sujeto y por tanto tienen una mecánica diferente aunque sistemas de puntuación parecidos.

En general, una tarea se compone de seis niveles distintos de dificultad y para superarla se deben superar todos y cada uno de ellos. Para la realización de la tarea tenemos un tiempo límite antes del cual debemos intentar de superar la prueba.

Cada nivel de las tareas se suele componer de ensayos (la única excepción es la tarea de El Bosque, que cada nivel tiene un solo ensayo), algunas tienen 6 ensayos y otras 10. Para poder pasar de nivel será necesario superar un determinado número de ensayos o unos determinados ensayos.

B.6.1. Inicio de una prueba de tarea

Para que un terapeuta pueda realizar una prueba de la tarea que quiera, debe ir a Tareas en el menú principal y verá la vista general de las diez tareas (Figura B.24).

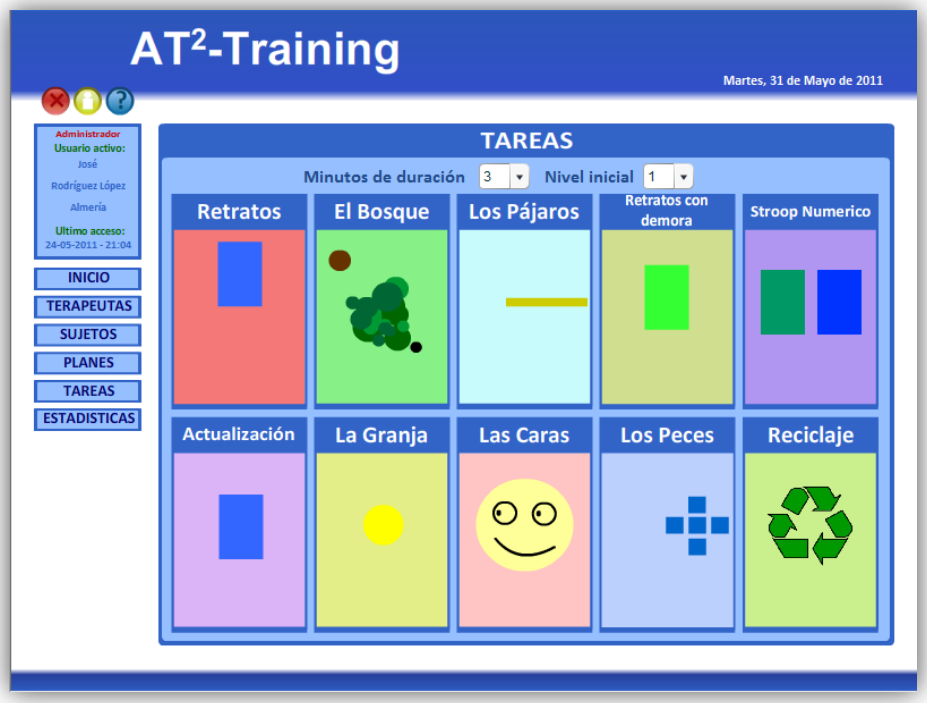


Figura B.24 – Vista general de tareas

Aquí podrá elegir el nivel de comienzo y el tiempo, en minutos, de la prueba. Una vez elegidos estos dos parámetros, solo deberá hacer clic en el icono de la tarea que quiera probar y la aplicación le mostrará la descripción de dicha tarea (Figura B.17).

Una vez acabada la prueba, por finalización de tiempo o superada, volveremos al menú de tareas para realizar otra prueba si deseamos.

B.6. Estadísticas

Haciendo clic en Estadísticas, en el menú principal, accederemos a toda la información disponible en la base de datos sobre Terapeutas, Sujetos y Planes. Es aquí donde podremos consultar la información de Planes anteriores de cualquier sujeto, ya que la vista de detalle de plan solo nos aportaba información sobre el plan en curso.

Bibliografía

[1] Roger S. Pressman. "*Ingeniería del Software. Un enfoque práctico*", McGraw-Hill, 2005.

[2] Sommerville, Ian. "*Ingeniería del software. 7ª edición*", Addison Wesley. 2005.

[3] Booch, G., Jacobson, I. y Rumbaugh, J. "*El Lenguaje Unificado de Modelado*", Addison Wesley, 2001.

[4] Bhangal, S. y Besley, K. "Foundation Flash 8". Friendsoft, 2006.

[5] Powers, D. "PHP 5 for Flash". Friendsoft, 2005.

[6] Moock, C. "Essential ActionScript 2.0". O'Reily, 2004.

[7] Vogeleer, D., Wilson, E. y Barber, L. "Macromedia Flash Professional 8 UNLEASHED". Sams, 2005.

[8] Cantwell, D.P. (1996). Attention Deficit Disorder: A review of the past 10 years. *Journal of American Child and Adolescence Psychiatry, 35*, 8, 97-987.

[9] Cardo, E., Nevoy, A., Redondo, M., Melero, A., De Azua, B., García-De la Banda, G. y Servera, M. (2010). Trastorno por déficit de atención/hiperactividad: ¿un patrón evolutivo? *Revista de Neurología, 50 (Supl 3)*, 143-147.

[10] Daza, M.T. y García-Giménez, N. (2010). Estudio atencional en niños con riesgo en los principales rasgos del TDAH. *Revista de Neurología, 51*, 244-244.

[11] Daza, M.T., Gonzálvez, E., Escobosa, J.A., Del Águila, E.M. y Agis, I.F. (2010). Programa de intervención neuropsicológica en niños de 7 a 11 años: Entrenamiento de la atención ejecutiva. *Revista de Neurología, 50(9)*, 568-568.

[12] Diamond, A. et al. (2007) Preschool improves cognitive control. *Science, 318*, 1387–1388

[13] Fernández-Mayoralas, D.M., Fernández-Jaén, A., García-Segura, J.M. y Quinones-Tapia, D. (2010). Neuroimagen en el trastorno por déficit de atención/hiperactividad. *Revista de Neurología, 50 (Supl 3)*, 125-133.

[14] Gillis, J.J., Gilger, J.W., Pennigton, B.F. y Defries, J.C. (1992). Attention deficil disorder in reading disabled twins: Evidence for a genetic etiology. *.Iournal ofAbnormal Child Psychology,* 20, 303-3 15.

[15] Gizer, I.R., Ficks, C., Waldman, I.D. (2009). Candidate gene studies of ADHD: a meta-analytic review. *Human Genetics, 126*, 51-90.

[16] Johnson, K.A., Robertson, I.H., Barry, E., Mulligan, A., Dáibhis, A., Daly, M., Watchorn, A., Gill, M. y Bellgrove, M.A.(2008). Impaired conflict resolution and alerting in children with ADHD: evidence from the Attention Network Task (ANT). *Journal of Child Pychology an Psychiatry, 49(12)*, 1339-1347.

[17] McDonald, A.W., Cohen, J.D., Stenger, V.A. y Carter, C.S. (2000). Dissociating the role of the dorsolateral prefrontal and anterior cingulated cortex in cognitive control. *Science*, *288*, 1835-1838.

[18] Orjales, I. (2000). Déficit de atención con hiperactividad: el modelo híbrido de las

funciones ejecutivas de Barkley. *Revista Complutense de Educación,11(1)*, 71-84.

[19] Posner, M.I. et al. (2010). Training effortless attention. En: *Effortless Attention: A New Perspective in the Cognitive Science of Attention and Action*, B. Bruya (Ed.), Cambridge: MIT Press.

[20] Posner, M.I. y Dehaene, S. (1994). Attentional networks. *Trends in Neuroscience*, *17*, 75-79.

[21] Posner, M.I. y Petersen, S.E. (1990). The attention system of the human brain. *Annual Review of Neuroscience, 13*, 25-42.

[22] Posner, M.I. y Rothbart, M.K. (1991). Attentional mechanisms and conscious experience. En A.D. Milner y M.D. Rugg (Eds.), *The neuropsychology of consciousness* (pp. 91-112), London: Academic Press.

[23] Rueda, M.R., Fan, J., McCandliss, B.D., Halparin, J.D., Gruber, D.B., Lercari, L.P., Posner, M.I. (2004). Development of attentional Networks in Childhood. *Neuropsychologia, 42(8)*,1029-1040. 2004.

[24] Rueda, M.R., Rothbart, M.K., McCandliss, B.D., Saccomanno, L. y Posner, M.I. (2005). Training, maturation, and genetic influences on the development of executive attention. *Proceedings of the National Academy of Sciences,102*, 14931–14936.

[25] Rueda, M.R., Posner, M.I. & Rothbart, M.K. (2004) Attentional control and self regulation. In R.F. Baumeister & K.D. Vohs (Eds). Handbook of SelfRegulation: Research, Theory, and Applications. New York: Guilford Press Ch. 14, 283-300