UNIVERSIDAD POLITÉCNICA DE MADRID
Escuela Técnica Superior de
Ingeniería y Sistemas de Telecomunicación

PROYECTO FIN DE GRADO

PLATAFORMA MÉDICA PARA EL ENTORNO DE VIDEOJUEGO TERAPÉUTICO "BLEXER"

MÓNICA JIMÉNEZ RAMOS

Grado en Ingeniería de Telemática

Julio 2017

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA Y SISTEMAS DE TELECOMUNICACIÓN

PROYECTO FIN DE GRADO

TÍTULO: PLATAFORMA MÉDICA PARA EL ENTORNO DE VIDEOJUEGO TERAPÉUTICO ''BLEXER''

AUTOR: MÓNICA JIMÉNEZ RAMOS

TITULACIÓN: GRADO EN INGENIERÍA TELEMÁTICA

TUTOR: MARTINA ECKERT

DEPARTAMENTO: TEORÍA DE LA SEÑAL Y COMUNICACIONES

VºBº

Miembros del Tribunal Calificador:

PRESIDENTE: MARIA LUISA MARTÍN RUIZ

TUTOR: MARTINA ECKERT

SECRETARIO: ENRIQUE RENDÓN ANGULO

Fecha de lectura: 13 de Julio de 2017

Calificación:

El Secretario,

AGRADECIMIENTOS

A toda mi familia, por estar absolutamente siempre.

A mi tutora Martina, por toda su confianza y cariño.

A Agata, Alfonso, Marta, Goyo, Cris y Joel,
por acompañarme en este viaje.

RESUMEN

Este Proyecto de Fin de Grado está enmarcado dentro de la investigación de la Dra. Martina Eckert, sobre juegos serios para rehabilitación e interfaces naturales para personas con discapacidad, predominantemente niños y adolescentes. Tras varios años, la línea de investigación se ha encauzado en torno a videojuegos de rehabilitación con la cámara Kinect de Xbox, de Microsoft. No obstante, algunas ramificaciones afluentes se inclinan al uso de otros dispositivos para terapia, como teléfonos inteligentes o gafas de Realidad Virtual.

Conforme se han desarrollado diversos videojuegos, surge la necesidad de una base de datos para almacenar resultados y poder llevar un seguimiento de la rehabilitación, así como de un mecanismo que posibilite que especialistas en fisioterapia puedan adaptar los videojuegos a la necesidad del paciente.

En este Proyecto de Fin de Grado se ha implementado una plataforma Web que se ha bautizado como “Blexer-Med” y consiste en una herramienta de trabajo para profesionales de fisioterapia y rehabilitación, para la gestión de la configuración de futuras sesiones y la gestión de resultados. La plataforma está diseñada de modo que se pueden administrar los juegos que están disponibles en un momento determinado, lo que la convierte en escalable en un futuro sin necesidad de reprogramarla para añadir videojuegos. También se han establecido una serie de roles entre los usuarios de la plataforma, para darle fiabilidad y robustez, ya que al tratarse de un aplicación que maneja datos personales de pacientes, debe ofrecer seguridad. Por ello también se han utilizado algunas técnicas de cifrado, y autenticación, y se ha establecido una división de pacientes y personal autorizado en centros médicos.

Por otra parte, se ha modificado las características del *middleware* “Chiro” –desarrollado por el ingeniero Ignacio Gómez-Martinho en su Proyecto de Fin de Grado, dentro de la misma investigación que el actual proyecto– que comunica los múltiples dispositivos (Kinect, gafas de Realidad Virtual, *Smartphones*, etc.) con los videojuegos. La modificación del *middleware* se ha realizado con el fin de que éste se descargue las configuraciones hechas por un terapeuta cuando un usuario empiece una partida, y del mismo modo envíe los resultados a la plataforma Blexer-Med cuando la partida termine.

La combinación de la plataforma web con el middleware Chiro dotan a esta investigación de la suficiente modularidad como para que cualquiera de ellos en un futuro sea modificado e incluso reemplazado por otro componente sin afectar al resto del entorno de los videojuegos. Así, con este Proyecto de Fin de Grado se proporciona una solución a una necesidad real que permitirá a la investigación global seguir su curso.

ABSTRACT

This Final Degree Project is taking part in the research activities of Dr. Martina Eckert, which treat with serious games for rehabilitation and natural interfaces for disabled persons, preferably children and teenagers. The focus of her research is currently set on exergames (Exercise games) using the Xbox Kinect camera (Microsoft), and other devices like smartphones or Virtual Reality glasses. As a game engine and modelling tool, the free available Blender software is used.

In this sense, a modular middleware that transmits the Kinect data to the Blender game engine was developed in former projects. Also, four mini-games have been realised and tested. This environment is called "Blexer" (Blender Exergames). During the present year, in parallel to this project, two students were integrating the mini-games into a complete adventure game. As the final aim is that the game (and other future games) could be played at the patient's homes and the therapist could supervise the players and adapt the difficulties from a distance, the need for a web-based platform with integration of a database aroused.

It was the objective of the present Project to design and develop this web platform and database, such that it would work together with the Blexer environment (middleware and games). As it is representing the medical access to the games, it was called "Blexer-Med".

The resulting platform is a complete working tool for professionals who will be able to manage the sessions of the patients and analyse the results. The games can be reconfigured in any moment, new games can be added, patients and therapists can be registered or withdrawn. Three different user roles are available (therapist, administrator of a centre and super administrator). To assure the confidentiality of the patient's data, authentication and encryption techniques have been applied.

To enable the communication between the platform and the middleware, the code has been modified in a place, that a file exchange for configuration data and results takes place when an internet connection is available. The programming is realised in a modular way, such that future modifications could be integrated without disturbing the rest of the system. In this way, this Final Degree Project contributes to an important solution to a real need.

INDICE DE CONTENIDOS

INDICE DE ILUSTRACIONES

1. Introducción y objetivos

Este Proyecto de Fin de Grado forma parte del trabajo de investigación de la Dra. Martina Eckert, sobre fisioterapia y rehabilitación para personas con discapacidad mediante juegos serios e interfaces naturales. Durante ya casi cuatro años han colaborado múltiples alumnos en el proyecto "Blexer" (*Blender Exergames*) que tiene como objetivo lograr que personas con discapacidad motora hagan ejercicios físicos mediante juegos. Merece una mención especial Ignacio Gómez-Martinho, quien hizo una grandísima aportación. En su periodo de prácticas y posteriormente en su PFG realizó la base de todo: un *middleware* llamado "Chiro"[1] que transmite los datos obtenidos de distintos dispositivos como p.ej. la cámara Kinect, gafas de realidad virtual, *smartphones*, etc. a los juegos virtuales, en los cuales estos señales sirven para controlar objetos o personajes. Como juegos, hasta el inicio de este proyecto, se crearon cuatro mini-juegos que sirven para practicar diferentes movimientos de brazo, tipo remar, escalar, golpear y volar.

El actual proyecto se realiza en paralelo al proyecto de dos estudiantes cuya labor es la integración de los mini-juegos en una historia completa, envolvente y motivadora; de esta manera, los mini-juegos serán ejercicios particulares del juego completo, que deben ser realizados para avanzar. Un aspecto muy importante de los ejercicios es su adaptabilidad a personas con diferentes necesidades y capacidades. Para conseguirlo, ciertos parámetros deben ser configurables, como p.ej. el tiempo de realización de un ejercicio determinado o el número de veces que hay que realizar el movimiento. Para que esta adaptación pueda ser realizada por parte de un experto (el médico rehabilitador o fisioterapeuta) y en cualquier momento, y el paciente use el juego en su propia casa sin acudir a la clínica, el presente proyecto tiene como objetivo realizar una plataforma web con base de datos, que se comunica con el *middleware*, para que este transmita los parámetros de configuración establecidos por el experto al juego. También es importante que el especialista esté informado de los resultados de los ejercicios con el fin de prevenir efectos negativos y pueda modificar la configuración establecida para el ejercicio cuando lo considere oportuno. Por esa razón, la monitorización de resultados es otra de las finalidades de este PFG, pues resulta necesaria para poder cuantificar los progresos de un paciente.

La plataforma web, como herramienta de trabajo para los especialistas, debe consistir en una interfaz gráfica amigable para que terapeutas puedan realizar configuraciones de ejercicios y visualizar sus resultados y, por consiguiente, debe contar también con una base de datos robusta. Así, sumando los elementos mencionados, el entorno de videojuegos Blexer estaría constituido por los dispositivos (Kinect, *smartphone*,…), los videojuegos, el *middleware*, la plataforma web, y la base de datos. Con esto se lograría el desacoplamiento de las partes de forma que se podría modificar o incluso sustituir alguna de ellas sin afectar sustancialmente implementación de las demás.

La base de datos elegida para este proyecto es MySQL por ser, sobre todo, muy fácil de aprender a utilizar, pero también por tener una versión de libre distribución, por la cantidad de documentación que hay al respecto, por ser escalable y sencilla de modificar, y por ser multiplataforma (da soporte varios sistemas operativos como Windows, Linux y Mac), que puede ser útil en un futuro.

Por otro lado, se ha determinado que la plataforma web se basará en un *Web Service* al que podrá acceder tanto el terapeuta desde su navegador como el *middleware* Chiro desde el PC del paciente. Este Web Service estará implementado en Java, el lenguaje orientado a objetos por excelencia. Esta elección se fundamenta en la disponibilidad de librerías de Java para desarrollo Web así como para acceso a bases de datos SQL (*Structured Query Language*).

Se desea que la plataforma sea un utensilio de trabajo para los especialistas, pudiendo gestionar la información personal de pacientes, centros médicos, intercambiar comentarios con otros terapeutas, etc. Además, se pretende establecer unos roles en los usuarios de la plataforma para implantar unos permisos y así ofrecer una mayor protección de datos.

Con este proyecto se espera dar un fuerte empujón al proyecto en cuanto al abanico de posibilidades para terapeutas, mejorando las sesiones de rehabilitación de los pacientes y promoviendo su autonomía a la hora de jugar. Así, se puede concluir que los objetivos principales de la plataforma Blexer-Med son:

- Establecer una herramienta de trabajo a terapeutas y personal de rehabilitación que ofrezca, al menos, los siguientes servicios:
    - Gestión de información personal de pacientes.
    - Configuración los ejercicios en función del individuo que vaya a realizarlos.
    - Almacenamiento y visualización de resultados de los ejercicios.
    - Interfaz gráfica amigable para personal médico.
- Creación de una base de datos robusta y escalable en el futuro.
- Roles en los usuarios de la plataforma para convertirla en segura y fiable.
- En definitiva, aumentar la eficiencia de las sesiones de rehabilitación y promover la autonomía del paciente permitiendo que pueda jugar desde casa.

2. Antecedentes

El marco tecnológico de este Proyecto de Fin de Carrera es el ámbito *e-Health o* e-Salud, término recientemente acuñado para las TIC (Tecnologías de la Información y la Comunicación) en el sector de las Ciencias de la Salud. *e-Health* engloba diversos productos como aplicaciones móviles, dispositivos *wearables*, videojuegos, *Big Data*, IoT (*Internet of Things*), entre otros, para ofrecer servicios de prevención, diagnóstico, seguimiento, e incluso gestión de la salud vía telemática, mejorando la eficacia de sistemas sanitarios.

"En opinión del Dr. Sergio Vañó, presidente de la Asociación de Investigadores en eSalud (AIES) [2], la e-Salud supone una <<transformación radical de la sanidad y, por ello, es necesario una evaluación de la eficacia y la seguridad de los sistemas de eSalud, con el objetivo de que los profesionales sanitarios>> estén preparados y, los datos proporcionados por los dispositivos de monitorización, puedan integrarse en la asistencia sanitaria". [3]

La investigación de la que este PFG forma parte utiliza juegos serios para rehabilitación de personas discapacitadas. La gamificación –que en este proyecto se aplica- es una tendencia que se utiliza cada vez más en *e-Health*, aunque también en otros campos como marketing, negocios o aprendizaje. Es una técnica que utiliza herramientas que incitan a "jugar", ganar puntos, esforzarse, a los usuarios de un servicio determinado a la hora de realizar una tarea. Por ejemplo, las votaciones o comentarios en una web de alquiler de viviendas vacacionales u otros negocios colaborativos. Esto puede tener el fin de fidelizar a los clientes de una empresa, o como es el caso de esta investigación: motivar a personas a hacer rehabilitación de una forma divertida.

Un proyecto muy similar al Blexer es el proyecto REWIRE [4] (*Rehabilitative Layout in Responsive Home Environment)*, iniciativa de teleRehabilitación en pacientes que han sufrido un ictus. Este proyecto utiliza también Kinect, combinándola con sensores de movimiento colocados en las extremidades del paciente, y con una balanza de equilibrio.

Existen otras aplicaciones dedicadas a la fisioterapia como el software Mira [5] creado por la fundación del mismo nombre en Reino Unido. Mira se sirve igualmente de la cámara Kinect, y una plataforma de videojuegos para pacientes en recuperación de una cirugía o lesión. Asimismo cabe destacar el software MOTMI [6], de la universidad UNSAM de Buenos Aires, que permite grabar en video las sesiones de rehabilitación. A diferencia del entorno de este proyecto, los ejercicios de MOTMI están orientados mayormente a adultos.

En una línea de investigación similar, Investigadores de la UPM, Universidad de Alcalá y Universidad Autónoma de Madrid colaboran en el proyecto EDUCERE [7], un sistema de detección precoz de trastornos de desarrollo en niños mediante juegos serios.

Más cercano a este proyecto se encuentra el proyecto de investigación Sonríe, por la profesora Marisa Martín Ruiz y sus alumnos (Universidad Politécnica de Madrid) que consiste en una serie de juegos para ejercitar los músculos faciales en jóvenes con parálisis cerebral.

También tutelados por esta profesora, los sistemas Gades y Pegaso ayudan a educadores y pediatras a localizar trastornos del lenguaje en niños de manera temprana. Por

otro lado, la plataforma Galatea [8], de Tania Matamoros (ETSIST, UPM) se centra en la evolución del conocimiento: a partir de los resultados de Gades y Pegaso, así como del consenso entre expertos, poder deliberar una decisión.

Un producto certificado por la Unión Europea y ya implantado en algunos hospitales es VirtualReb, plataforma que usa Kinect para tratar enfermedades neurodegenerativas y daños cerebrales [9].

Como se puede observar, casi todos los proyectos mencionados son trabajos de investigación, pues no hay abundantes plataformas de juegos serios para rehabilitación comercializados y puestos en uso en centros médicos.

3. Requisitos de la plataforma

La plataforma Web Blexer-Med consiste en una herramienta anexa a unos videojuegos de rehabilitación para personas con discapacidad. A través de ella, los terapeutas podrán principalmente:

- ✓ Adaptar diferentes parámetros de dificultad de los videojuegos a la necesidad de cada paciente individual.
- ✓ Almacenar y consultar los resultados de las partidas jugadas por los pacientes

3.1. Tipos de usuarios

Aunque en la investigación global de la que este PFG forma parte, todo gira en torno a los pacientes, Blexer-Med estará destinada a terapeutas con el fin de que estos puedan supervisar la rehabilitación de los pacientes que hagan uso de los videojuegos. Los **terapeutas**, como usuarios de la plataforma, estarán asociados a un centro, del mismo modo que los pacientes con los que trabajen, y un especialista solo puede acceder a los pacientes de su mismo centro.

Por seguridad de la plataforma, será requerido que los terapeutas tengan autorización para utilizar Blexer-Med. La plataforma por tanto necesita un tipo de usuario que gestione cada centro y dé de alta/baja a los terapeutas de dicho centro. Este usuario se llamará **administrador del centro** y tendrá dominio sobre un único centro médico. Como no resulta conveniente restringir la escala de la plataforma, podrá haber varios administradores por cada centro. En cualquier caso, por cada centro debe haber al menos un administrador.

Por consecuente, es necesaria también la existencia de otro tipo de usuario que gestione los administradores de cada centro y tenga una visión global de todos los centros. Este será el **superadministrador de la plataforma** y tendrá privilegios para añadir nuevos centros a la plataforma y los correspondientes usuarios administradores. De nuevo, para no limitar la escala del sistema, podrá haber varios superadministradores en el sistema.

En adelante se denominará “usuarios” a superadministradores, administradores, y terapeutas, pues son los que van a hacer uso de la plataforma. La pirámide de usuarios de la Figura 1 tiene el fin de establecer unos permisos y restricciones en las tareas que puede realizar cada persona, que por seguridad no deben ser las mismas.

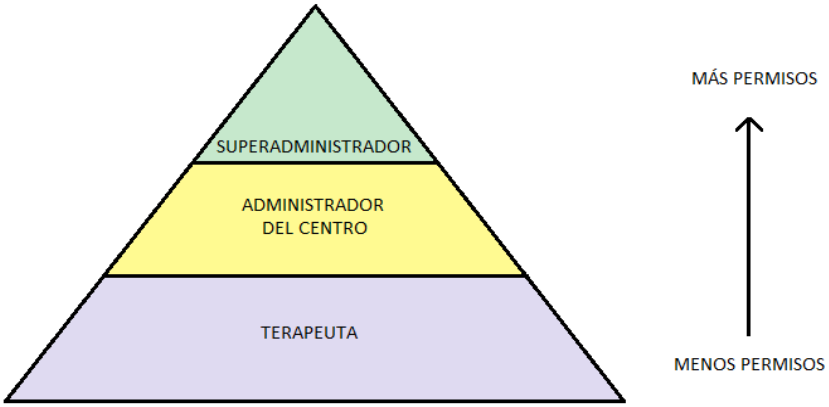


Figura 1 - Pirámide de permisos de usuario

En resumen, para separar los pacientes y terapeutas por centros médicos, habrá un superadministrador que pueda dar de alta estos centros, pudiendo haber varios superadministradores en el sistema. Después, cada centro tendrá uno o varios administradores, dados de alta por el superadministrador, que podrán dar de alta a su vez a terapeutas y pacientes. Por último, los terapeutas podrán dar de alta pacientes y gestionar los ejercicios que quieran que éstos hagan. Se muestra en la Figura 2 la estructura que tendría el sistema para N centros dados de alta.

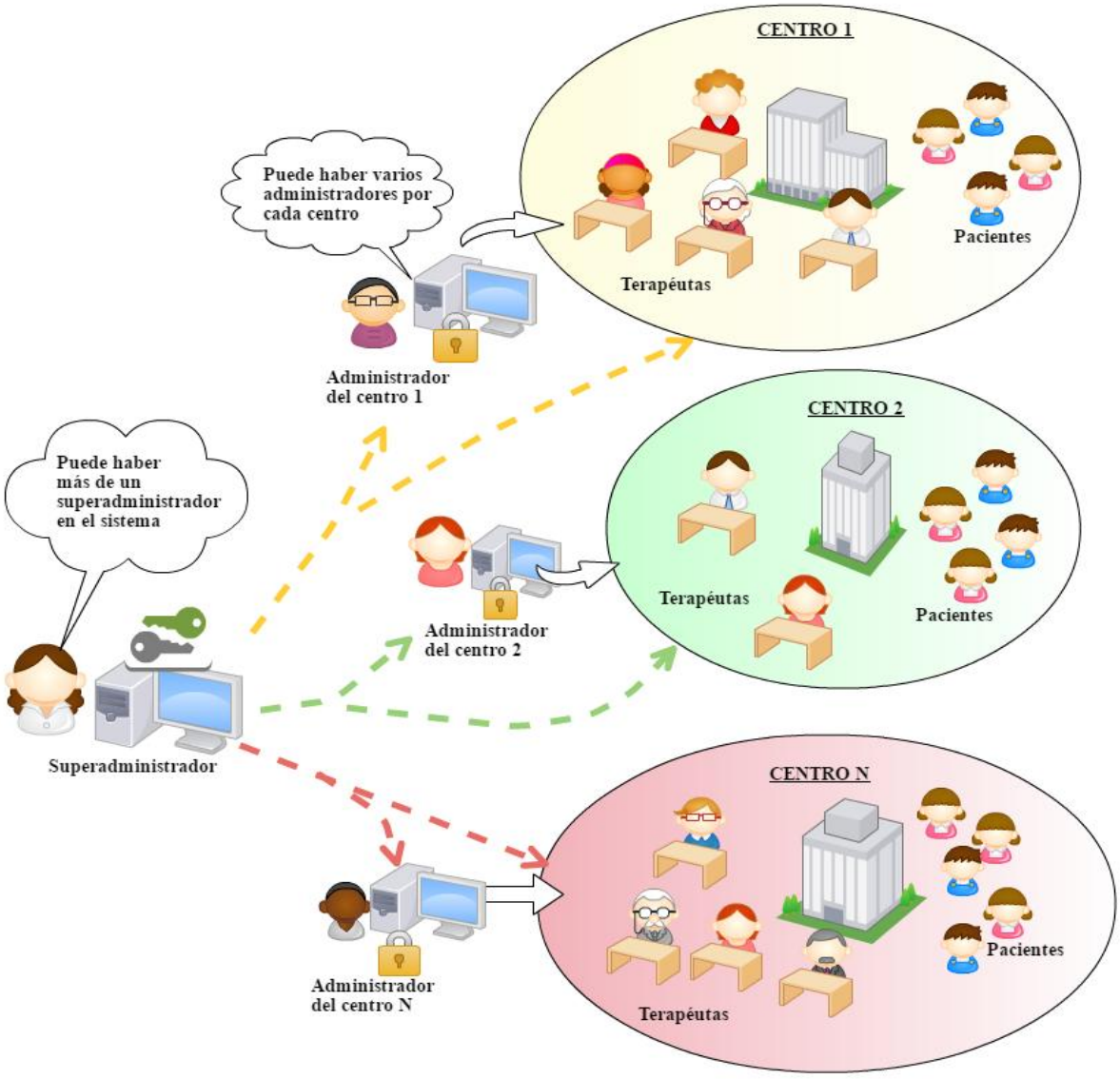


Figura 2 - Tipos de usuarios para N centros médicos

A continuación se profundiza en las acciones permitidas a cada usuario, empezando por el superadministrador. Entre las tareas encomendadas a este usuario se deben diferenciar las relacionadas con los centros médicos, o bien con los administradores de cada centro, o con la gestión de los videojuegos.

Se presenta en la Figura 3 el diagrama UML (*Unified Modelling Language)* de casos de uso de un superadministrador. UML es un estándar de modelado orientado a objetos para

el diseño de sistemas complejos así como una base para metodologías de desarrollo de software. Hay varios tipos de diagramas UML como los de secuencia, de actividad, de estado, y los de casos de uso entre otros, para visionar un sistema desde distintos puntos de vista. Los diagramas de casos de uso son muy importantes para esta situación porque sirven para explicar de manera muy sencilla quién utiliza un determinado sistema y todos los usos que pueden darle.

Para la comprensión de este diagrama se adelanta que un juego debe estar formado por uno o más ejercicios u que éstos últimos serán configurables por el terapeuta a través de unos parámetros.

Figura 3 - Diagrama de casos de uso del superadministrador

En cuanto a los administradores del centro, se espera que tengan la posibilidad de modificar la información del centro si fuese oportuno, así como de administrar el alta/baja de terapeutas y pacientes de su centro –y no de ningún otro centro–. En la Figura 4 se especifican las acciones permitidas para el administrador.

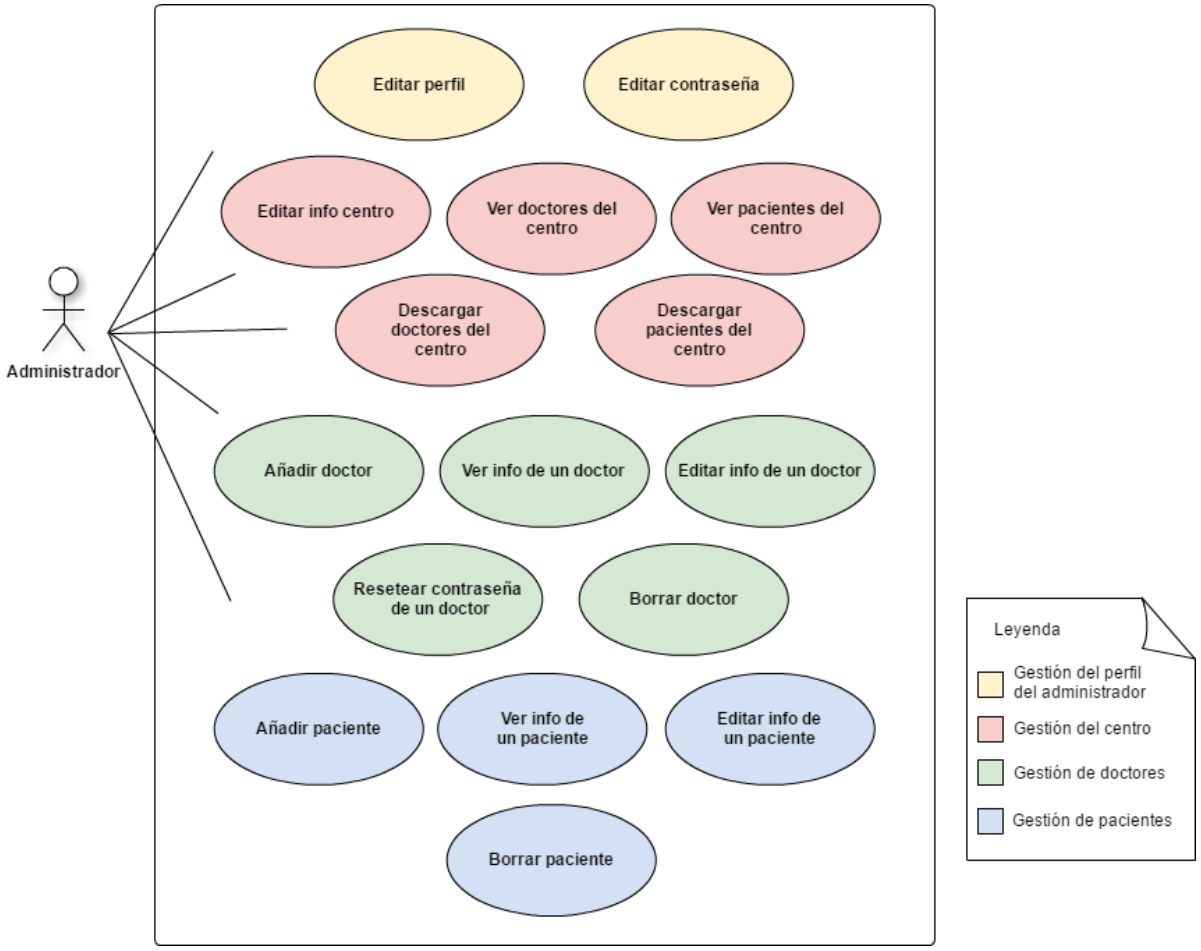


Figura 4 - Diagrama de casos de uso del administrador

Finalmente, el terapeuta es el usuario con menos privilegios en la pirámide de permisos. No obstante, como se puede observar en el diagrama de la Figura 5, es el único que tiene permisos para cambiar la configuración de los juegos para los pacientes, o para interactuar con otros terapeutas a través de comentarios.

Cabe resaltar que los casos de uso de terapeuta "Configurar un ejercicio" y "Ver resultados de una configuración" son los objetivos principales de este proyecto. El resto de usuarios y de casos de uso son complementarios, o derivados de necesidades para cumplir estos dos.

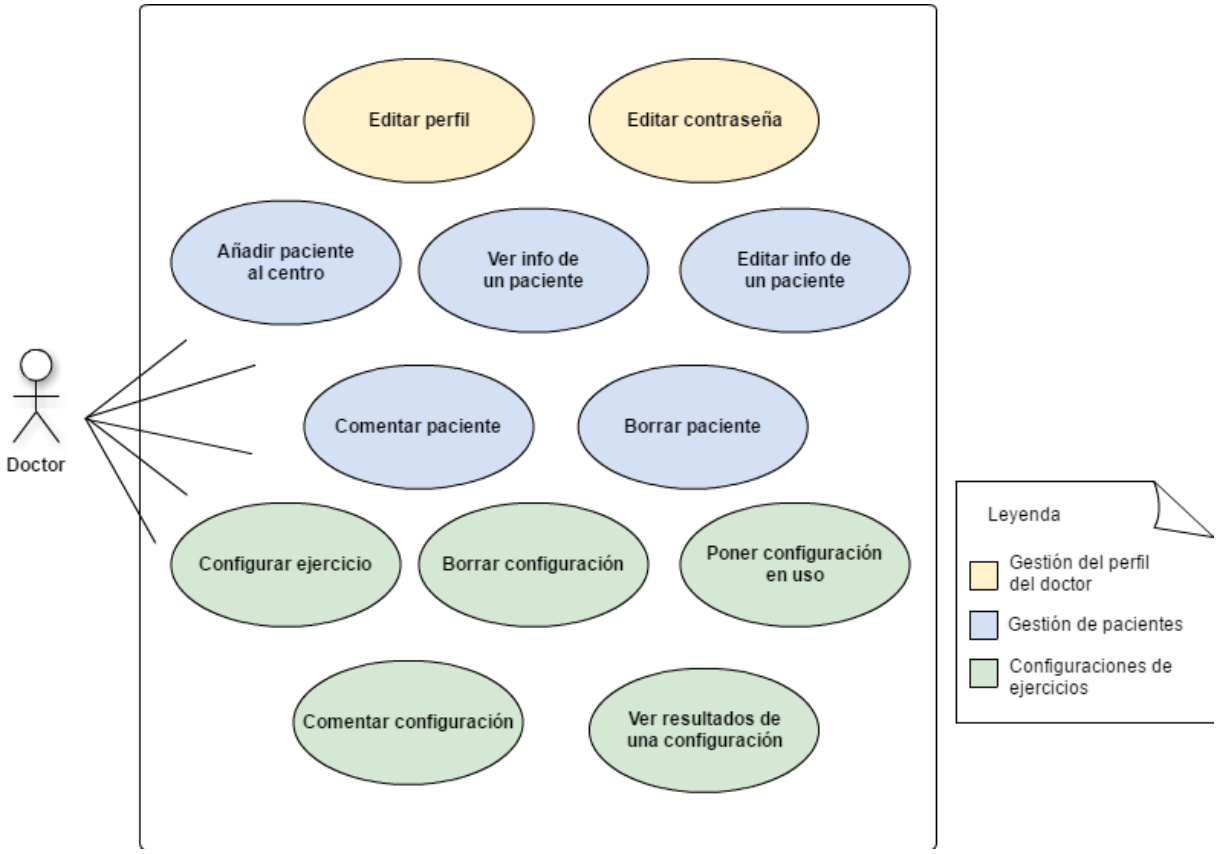


Figura 5 - Diagrama de casos de uso de un terapeuta

3.2. Juegos y ejercicios

Es sustancial tener en cuenta que paralelamente a este proyecto se está realizando un juego multiaventura para rehabilitación, el cual se pretende que sea configurable desde la plataforma Web. Por ello, la plataforma Blexer-Med debe ser modular de manera que en un futuro se puedan añadir más juegos, considerando que los videojuegos pueden variar –suprimir escenas, añadir nuevas, crear nuevos personajes, etc. –.

Con la intención de que conforme la investigación avance y los videojuegos varíen no haya que reprogramar la base de la plataforma, Blexer-Med debe ofrecer al **superadministrador** la posibilidad de dar de alta juegos, modificarlos o borrarlos si fuese oportuno. Además, a cada juego se debe poder añadir, modificar, o borrar ejercicios para el caso en el que el juego sea multiaventura o conste de distintas fases.

Dado que el objetivo principal de esta plataforma es poder configurar los distintos juegos a las necesidades de cada paciente, inicialmente era un requisito que, para los ejercicios que añadiese el superadministrador, el terapeuta pudiese configurar cuatro parámetros:

- Objetivo de número de movimientos a realizar
- Precisión de los movimientos
- Tiempo límite de cada partida para evitar sobrecarga
- Límite de movimientos para evitar sobrecarga

Esta idea evolucionó ya que es un modelo de configuración demasiado estricto y hay ejercicios que pueden necesitar otros parámetros diferentes. Finalmente se dejará a libre elección del superadministrador qué parámetros desea que el terapeuta pueda configurar. Esta elección no puede ser arbitraria si no que deben ser parámetros que obtengan información manejable por el funcionamiento interno del videojuego. El superadministrador debe estar en contacto con el desarrollador de los videojuegos pues la composición de cada videojuego que se añade a la plataforma Blexer-Med va de la mano de la implementación de éste y hay ciertas directrices a tener en cuenta –más adelante se detallarán–.

Los terapeutas podrán hacer configuraciones individuales para cada paciente de su centro y para los distintos ejercicios –según los parámetros establecidos por el superadministrador–. Podrán también consultar el histórico de configuraciones que tiene cada paciente y decidir con qué configuración quiere que sean jugadas las próximas partidas, poner un comentario, o borrar una configuración.

Otro de los objetivos de este PFG es la monitorización, visualización y almacenamiento de los resultados de las partidas. Los resultados de los pacientes de un centro estarán a disposición de los terapeutas de dicho centro en un historial, con el fin de que se pueda observar la progresión/regresión de un paciente de forma cuantificada. Se podrán descargar los resultados en un fichero CSV para poder trabajar con ellos en otra aplicación, siendo esto una gran ventaja de la plataforma Blexer-Med.

4. Solución propuesta

4.1. Estructura del proyecto

El punto de partida de este proyecto es el PFG de Ignacio Gómez-Martinho: "Desarrollo e implementación de *middleware* entre Blender, Kinect y otros dispositivos". Como se ha comentado en el capitulo 2 (Antecedentes), este alumno realizó varios mini-juegos en el entorno de desarrollo Blender, junto con un *middleware* llamado Chiro para conectar Blender con múltiples dispositivos. En la Figura 6 se plasma de forma muy simplificada la arquitectura del entorno de los videojuegos antes de comenzar este PFG.

Se puede observar que la comunicación entre Blender y el *middleware* se produce a través del Protocolo UDP (*User Datagram Protocol*) y del Protocolo OSC (*Open Sound Protocol)*. El primero es un protocolo de transferencia de datagramas (paquete de datos) a través de la red. El segundo sirve para el envío de información multimedia, de forma que los paquetes OSC se encapsulan en paquetes UDP.

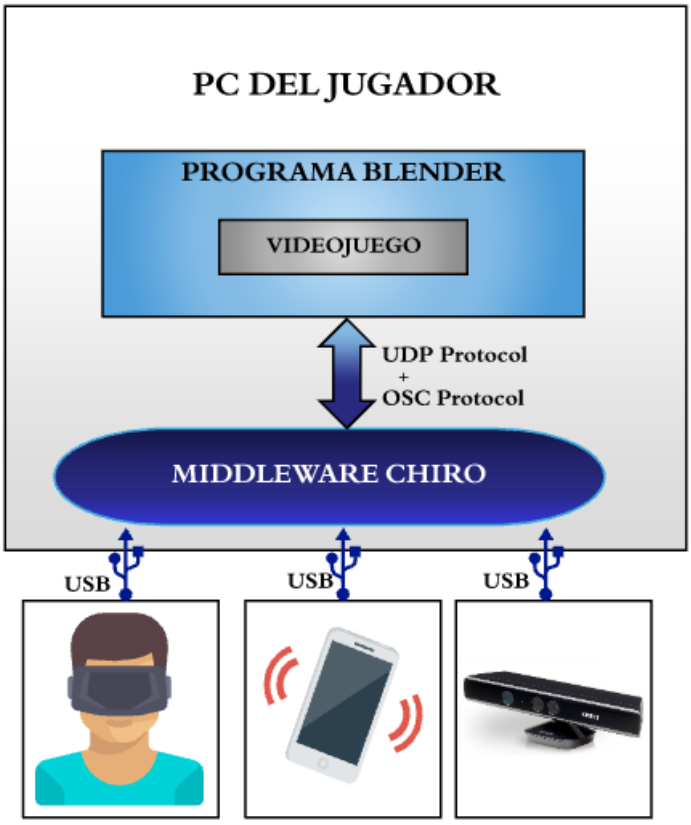


Figura 6 - Punto de Partida. Arquitectura.

Con el propósito de personalizar las partidas a los jugadores del videojuego se ha añadido a esta arquitectura una plataforma Web desplegada en un Servidor Web Apache Tomcat. Dicha plataforma es una interfaz gráfica para que el terapeuta no se vea obligado a usar Blender para cambiar la dificultad del videojuego, ya que Blender es un entorno de desarrollo y requiere conocimientos de programación en lenguaje C#.

Para almacenar la información de usuarios, pacientes, configuraciones, resultados de los juegos, etc., se ha utilizado una base de datos MySQL que reside en un servidor MySQL.

De cara a este proyecto se puede dividir las comunicaciones en tres partes:

1) Interacción de usuarios con el servidor Web para acceder a la plataforma.
2) Comunicación del *middleware* Chiro con la plataforma para recibir configuraciones de juegos y enviar resultados.
3) Comunicación del Servidor con la Base de datos para obtener la información que el Servidor responde a Chiro o al usuario.

Las tres comunicaciones, representadas en la Figura 7, se basan en el modelo Cliente/Servidor. Éste es un sistema distribuido en el que el cliente solicita un recurso al servidor y éste le responde.

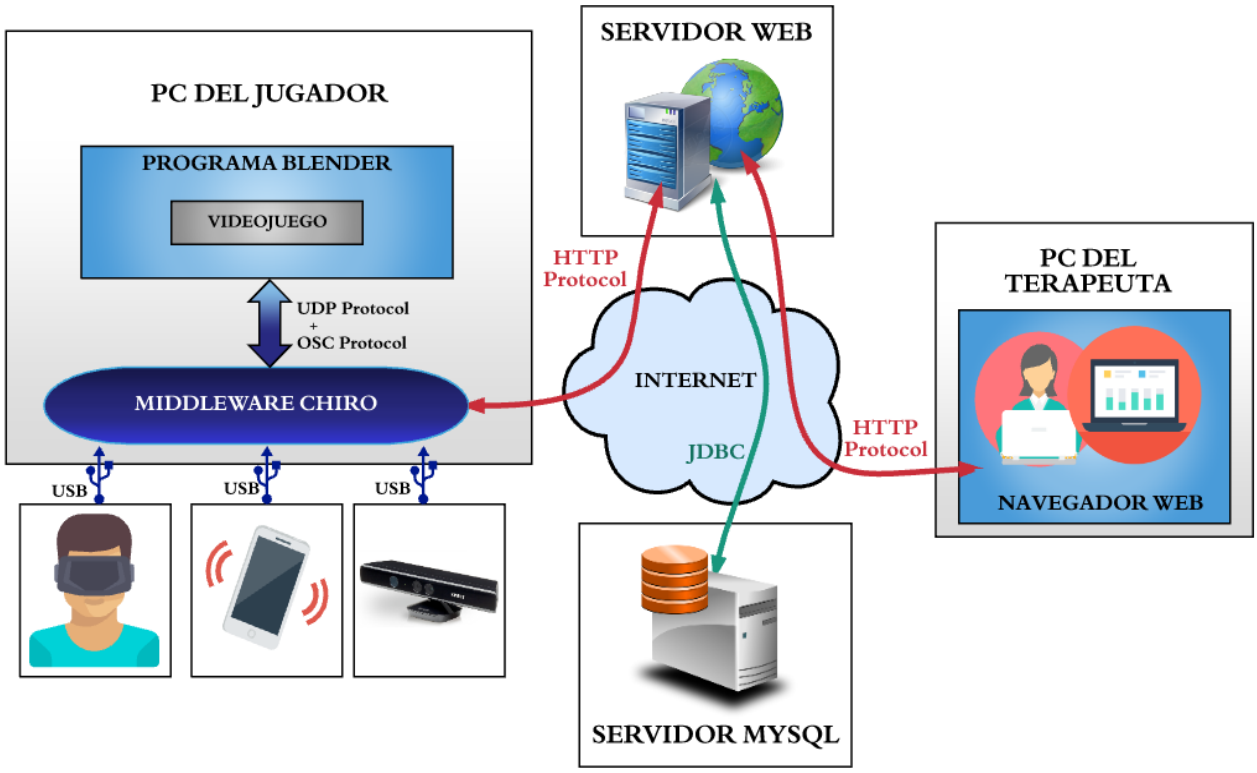


Figura 7 - Estructura del entorno y comunicaciones

Por un lado, el navegador (Cliente Web) del especialista carga la aplicación (Servidor Web) mediante peticiones/respuestas HTTP (*Hypertext Transfer Protocol*). Éste es un protocolo para la transferencia de texto plano.

Por otro lado, Chiro además de todo lo que hacía en el punto de partida se comportará como Cliente Web para solicitar a la aplicación la configuración de los ejercicios y, del mismo modo, enviar los resultados.

En ambos casos, cuando el Servidor Web recibe una petición, como puede ser el inicio de sesión del terapeuta o cuando Chiro solicita un recurso a la plataforma, se accede desde la

plataforma Web (Cliente MySQL) a la base de datos MySQL (Servidor MySQL) en forma de petición/respuesta.

En los posteriores puntos se detalla la implementación de la base de datos y el servidor Web, empezando por la base de datos ya que es un elemento aislado de pacientes y usuarios y se accede a ella a través del servidor Web.

4.2. Base de datos

4.2.1. Tecnología utilizada

Las bases de datos modernas se pueden dividir en bases de datos relacionales y bases de datos no relacionales. Las primeras, están estructuradas en tablas que almacenan los datos y están vinculadas entre sí. Una tabla tiene una o más columnas que la definen, y filas que representan los registros almacenados en la BD (Base de Datos).

Las bases de datos no relacionales, no requieren que los datos se almacenen en tablas, sino que además tienen otras formas como "clave-valor", *BigTable*, *arrays*, grafos u otras. Éstas tienen la ventaja de una gran escalabilidad y por su flexibilidad son recomendadas cuando la estructura de los datos manejados no sigue un patrón fijo.

Para este proyecto se ha optado por una BD **relacional** por su sencillez para realizar operaciones sobre la ella –inserción, consulta, modificación y borrado de elementos– y la atomicidad de éstas. Esto es: cuando se solicita una operación (ej. insertar un elemento), o se ejecuta por completo o, si se produce algún fallo, se revierten los cambios que se hayan producido, hasta dejar la BD en el estado previo a comenzar esta operación. De esta manera se garantiza la integridad de los datos, mientras que en las bases de datos no relacionales no se garantiza la atomicidad de operaciones, y por tanto tampoco la integridad de los datos.

La base de datos relacional elegida para este proyecto es **MySQL**. "Es la base de datos más popular del mercado. Gracias a su rendimiento probado, a su fiabilidad y a su facilidad de uso, MySQL se ha convertido en la base de datos líder elegida para las aplicaciones basadas en web y utilizada por propiedades web de perfil alto, como Facebook, Twitter, YouTube…".[10]

Se ha utilizado la versión **Community Edition** de MySQL por ser de uso libre y estar disponible para alrededor de 20 plataformas y sistemas operativos como Linux, Mac y Windows. Tiene implementados conectores para diferentes lenguajes, y en este proyecto se usa como conector JDBC (*Java Database Connectivity*), una API (*Application Programming Interface*) de Java para acceder desde el servidor Web a la base de datos MySQL.

En MySQL las relaciones entre las tablas de una base de datos forman el modelo de la base de datos. En el modelo se declaran las distintas tablas que componen la BD, los tipos de datos almacenados en cada columna de cada tabla, y se define la relación que hay entre dichas tablas. En MySQL, los principales tipos de datos son numéricos (*int*, *double*, *decimal*,…), alfanuméricos (*varchar*, *text*,…), fechas (*date*, *timestamp*,…) y booleanos.

En las tablas de una BD, el concepto de **Clave Primaria** (PK, *Primary Key*) es fundamental puesto que es el campo/conjunto de campos de una tabla que identifican cada registro y no pueden repetirse. Por ejemplo, en la tabla Persona la PK sería la columna DNI,

pero podría haber dos o más PK, como el Nº de Pasaporte, siendo ambos valores únicos para cada persona.

La relación entre dos tablas se establece a través de una **Clave Foránea** (FK, *Foreign Key*) que es una columna de una tabla A que apunta a una columna de una tabla B llamada **Clave Referenciada**. Por ejemplo, en la base de datos de la Figura 8 formada por las tablas Libro y Autor, por un lado la PK identificativa del libro es la tupla {ISBN, Dni_autor} y la PK del autor es su DNI, y por otro lado la FK de la relación entre las tablas es la columna Dni_autor de la tabla Libro que apunta a DNI de la tabla Autor.

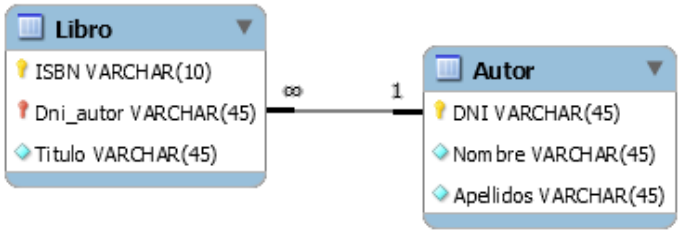


Figura 8 - Ejemplo explicativo sobre PK y FK

La FK de un registro no tiene porqué ser única, porque como se observa en el la figura anterior, el mismo Dni_autor puede estar presente en varios libros.

Las relaciones entre tablas pueden clasificarse principalmente de dos maneras:

- **Identificadoras/ No identificadoras**:
  - Una relación identificadora es aquella en la que la PK de una tabla A lleva implícita la PK de una tabla B, y por tanto un registro de la tabla A está condicionado a la existencia de un registro de la tabla B. Se representa por una línea continua. El ejemplo de la Figura 8 es una relación identificadora porque en ese caso no puede existir un libro sin el autor correspondiente.
  - Una relación no identificadora es aquella en la que la PK de la tabla B está incluida en la tabla A pero no es una PK de la tabla A, simplemente porque B no define a A. Se representa con línea discontinua. Es importante saber que una relación no identificadora puede ser obligatoria o no obligatoria. Por ejemplo, la tabla Persona de la Figura 9 tiene dos FK: Población y Estado_Civil, que no son PK de Persona, ya que una persona no está identificada por su Población ni por su Estado Civil. Sin embargo, en este ejemplo concreto, es obligatorio que una persona pertenezca a una población -este campo no puede valer NULL en la BD–, pero no es obligatorio que tenga estado civil –puede valer NULL en la BD–.

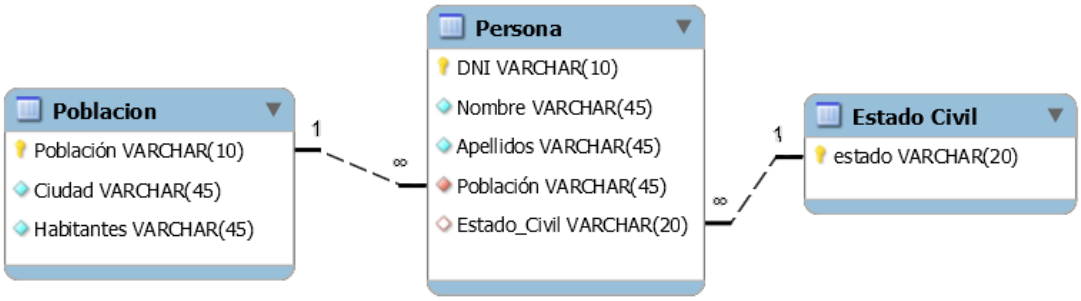


Figura 9 - Ejemplo de relaciones no identificadoras obligatorias/no obligatorias

- Por **cardinalidad**:
  - Relación "uno a uno":
    Un objeto de la tabla A está relacionado con un único de la tabla B y viceversa. Ej.: un coche tiene una matrícula única y una matrícula está asociada a un solo coche.
  - Relación "uno a varios":
    Un objeto de la tabla A puede estar relacionado con ninguno, uno, o varios objetos de la tabla B. Un objeto de la tabla B solo se está asociado a un objeto de la tabla A. Ej.: Un vendedor puede tener una venta, varias, o ninguna, pero una venta está hecha por solo un vendedor.
  - Relación "varios a varios":
    Un registro de la tabla A puede estar relacionado con un registro de la tabla B, varios, o ninguno, y viceversa. Ej.: Un alumno puede estar cursando varias materias a la vez, y estas materias estar siendo cursadas por varios alumnos.

Como último ejemplo para comprender las claves primarias, claves foráneas, identificación, y cardinalidad de MySQL, se muestra en la Figura 10 el modelo de base de datos de una biblioteca. Éste podría estar formado por las tablas Libro, Usuario y Préstamo relacionadas entre sí. La tabla Libro podría tener las columnas: código de libro (PK), nombre, autor y editorial. La tabla usuario podría tener los campos: DNI (PK), nombre y apellidos. La tabla Préstamos podría tener los campos: número de pedido (PK), código de libro (FK), DNI del usuario (FK), fecha de préstamo y fecha límite de devolución. Cabe destacar que en la tabla Préstamo, código_libro y dni_usuario además de FK son PK porque un préstamo está definido por el usuario y el libro que éste arrenda. Son por tanto, dos relaciones identificadoras. En cuanto a cardinalidad, un préstamo corresponde a un solo libro y a una única persona, mientras que una persona y un libro pueden figurar en N préstamos. En las figuras 11 y 12 se muestran posibles registros de las tablas Libro y Usuario

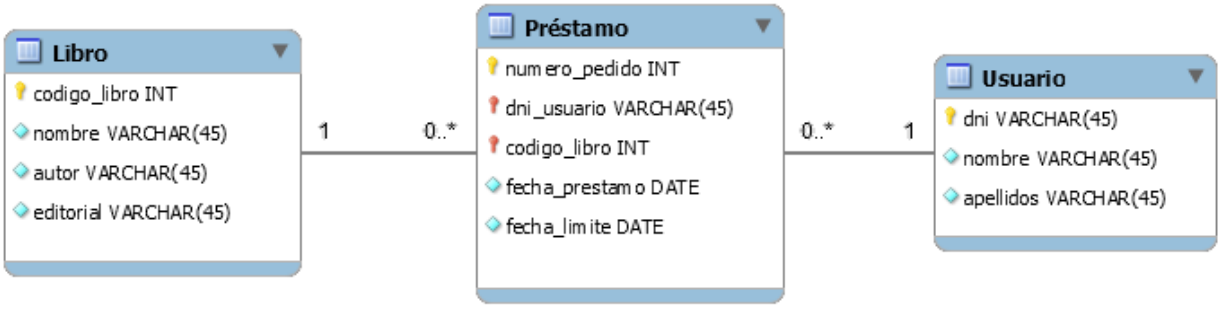


Figura 10 - Ejemplo de modelo de base de datos de una biblioteca

codigo_libro	nombre	autor	editorial
1	Manual De Fisioterapia En Traumatología	Esther Díaz Mohedo	Elsevier
2	MANUAL DE TÉCNICAS DE FISIOTERAPIA	R. Haarer-Becker	Paidotribo
3	Ejercicios de rehabilitación II: memoria	Armando Estévez González	Lebón
4	Rehabilitación ortopédica clínica	S.B. Brotzman	Elsevier

Figura 11 - Ejemplo de tabla Libro con 4 registros.

dni	nombre	apellidos
10174895K	Lucía	Soler Benitez
15478950M	Teresa	Cabrera Lozano
50201546J	Daniel	Herrera Pascual
50623248X	Rafael	Parra Esteban
84609547M	Sergio	Crespo Mora

Figura 12 - Ejemplo de tabla Usuario con 5 registros

A continuación, el posible contenido de la tabla Préstamo, con los préstamos desde marzo 2016 hasta junio 2017, con importancia de los campos dni_usuario y codigo_libro que son PK y FK.

numero_pedido	dni_usuario	codigo_libro	fecha_prestamo	fecha_limite
1	10174895K	2	2016-03-14	2016-03-30
2	10174895K	4	2016-05-27	2016-06-20
3	84609547M	4	2016-07-05	2016-08-25
4	15478950M	4	2016-09-13	2016-10-25
5	84609547M	3	2016-11-06	2016-12-03
6	50201546J	3	2017-05-06	2017-06-20

Figura 13 - Ejemplo de contenido de la tabla Préstamo

4.2.2. Solución para implementación de base de datos

La plataforma Blexer-Med tiene tres tipos de usuarios, con sus respectivas interfaces gráficas, permisos y utilidades. Por tanto las primeras tablas que han sido requeridas en la base de datos son las tablas para almacenar la información de supervisores, administradores y

terapeutas. Puesto que será un supervisor quien dé de alta centros médicos y les asigne administradores, se ha creado además una tabla para almacenar los registros de los centros.

La información que se guarda acerca de un supervisor es su *login* y *password*, porque se entiende a este individuo como una cuenta más que como una persona física con nombre y apellidos. De un administrador se van a guardar su *login* y *password* para el acceso a la plataforma Web, pero también nombre, apellidos e *email*, y el centro que éste gestiona. Igualmente, para un terapeuta se almacenan *login*, *password*, nombre, apellidos, *email*, y el identificador del centro al que está adscrito.

Para llevar un control sobre quién da de alta qué, se almacenará en los centros y administradores un campo para el superadministrador que los creó, y en cada terapeuta, un campo con qué administrador le dio de alta. Esta información no debe ser obligatoria, pues de serlo se borraría un registro si la persona que lo añadió al sistema se da de baja. En la Figura 14 se muestran las columnas que tendrán las cuatro tablas.

Se puede ver en la figura que cada registro de las cuatro tablas tiene un identificador único: id_supervisor, id_admin, id_clinician e id_center. También las cuatro tienen una columna time_stamp que sirve para guardar la fecha exacta en la que se añadió un registro a la base de datos.

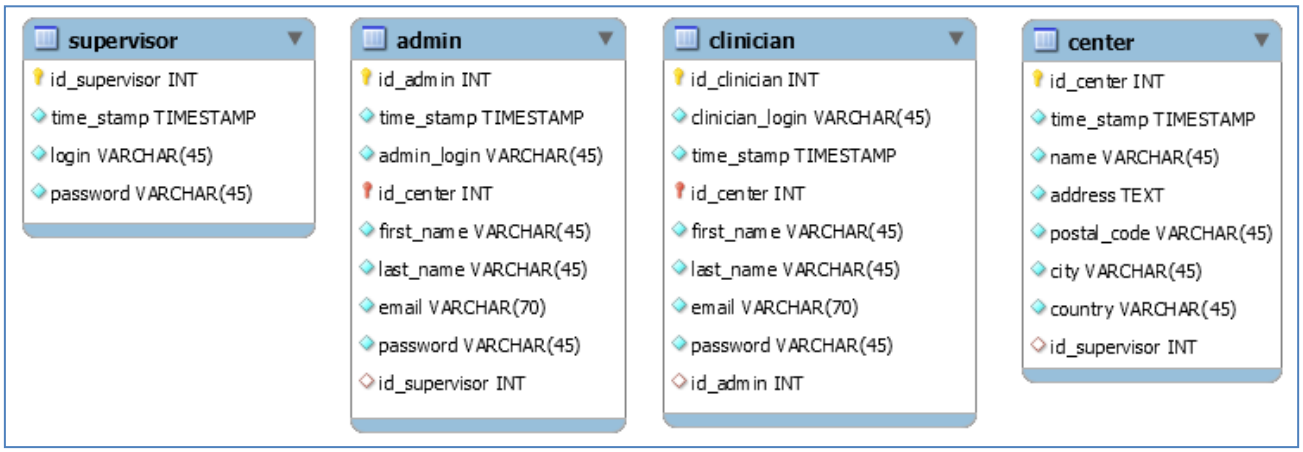


Figura 14 - Tablas de Supervisor, Administrador, Terapeuta y Centro.

Las cuatro tablas anteriores son necesarias para crear una infraestructura de centros médicos y personas responsables de la rehabilitación de pacientes. En consecuencia, se ha añadido una tabla para recoger la información de los pacientes (Figura 15). De cada paciente interesa almacenar su nombre, apellidos, fecha de nacimiento, observaciones médicas, teléfono de contacto y email de contacto. Los tres últimos no han sido considerados obligatorios. Lógicamente hace falta una PK, que identifique unívocamente a cada paciente, y una columna para el centro al que pertenece. El paciente no tiene acceso a la plataforma Web, pero sí deberá establecer una conexión con el servidor cuando vaya a jugar a los videojuegos para descargar su configuración. Así pues, se han incorporado las columnas *login* y *password*, que se utilizarán para iniciar sesión desde el *middleware* Chiro. Por último, se almacenará la fecha de alta de este usuario en la columna *time_stamp*.

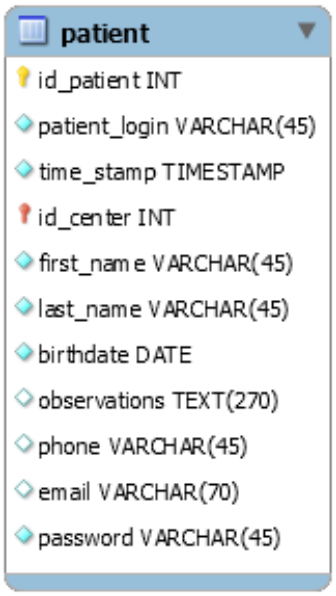


Figura 15 - Tabla Paciente

Se quiere que, además de la configuración de ejercicios, un terapeuta pueda poner notas o comentarios en el perfil de un paciente, y así pueda haber también de algún modo una interacción entre terapeutas. Por consiguiente, existe en la base de datos una tabla, que se muestra en la Figura 16, para guardar los comentarios de los terapeutas.

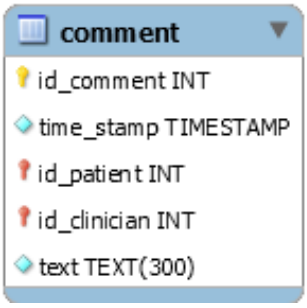


Figura 16 - Tabla Comentario

Cada comentario tendrá un identificador único, una fecha de creación, el texto propio del comentario y estará asociado a un paciente y a un terapeuta.

Como se ha explicado en el apartado 3.2. “Juegos y Ejercicios” sobre los requisitos de la plataforma, un juego está formado por uno o más ejercicios que se añaden, modifican o borran desde la interfaz del superadministrador. Se ha establecido la tabla Juego (Figura 17) con un identificador único, una fecha de creación, un título y una descripción.

Desde el punto de vista del envío de configuraciones al videojuego, el *middleware* Chiro descargará los datos de configuraciones. Para solicitar las configuraciones de los ejercicios de un juego en concreto, hará falta un identificador único para este juego, lo cual produce un inconveniente: el desarrollador del videojuego debe conocer de antemano el identificador único del juego por el que debe preguntar para descargar configuraciones. Sin embargo, el identificador del juego es un número entero que se genera automáticamente cuando el juego se inserta en la BD y para insertarlo el videojuego ya debe estar listo para su

uso. Es decir, el desarrollador del videojuego va a necesitar el identificador antes de que éste exista porque aún no se habrá insertado el juego en la BD.

La solución encontrada al problema anterior es insertar en la tabla Juego una columna con un "código de juego" alfanumérico, de 5 caracteres, que pueda ser establecido en mutuo acuerdo por el desarrollador del juego y superadministrador. Así, el desarrollador del juego no necesita que el juego esté insertado en la BD para conocer el código con el que tiene que descargar configuraciones. Además, que el código sea alfanumérico lo convertirá en más intuitivo que un identificador numérico en el caso de que haya gran cantidad de juegos en un futuro.

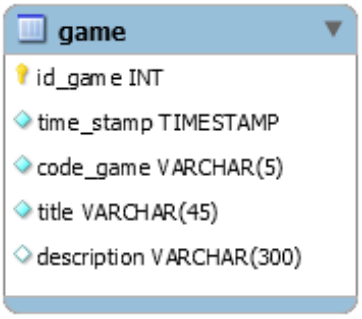


Figura 17 - Tabla Juego

Se hace uso de la tabla Ejercicio de la Figura 18 para registrar las partes, al menos una, del videojuego. De cada ejercicio se guarda un identificador único (numérico y automático), el nombre del ejercicio, la descripción del ejercicio, la fecha de creación y un código de ejercicio alfanumérico –que se establece entre desarrollador del videojuego y superadministrador para el envío de configuraciones al *middleware* Chiro, del mismo modo que el código de juego de la tabla Juego–. Además, se guarda el identificador del juego al que pertenece cada ejercicio.

Como aparece en los requisitos de juegos y ejercicios, el superadministrador debe poder elegir hasta cuatro parámetros del videojuego que quiere que sean configurables por el terapeuta. Para que esto sea posible, la tabla Ejercicio tiene cuatro columnas –no obligatorias– para los nombres de estos parámetros y cuatro columnas para que el superadministrador introduzca una descripción sobre qué significan estos parámetros y qué se espera recibir. En la figura 18 se aprecian todas las columnas mencionadas previamente.

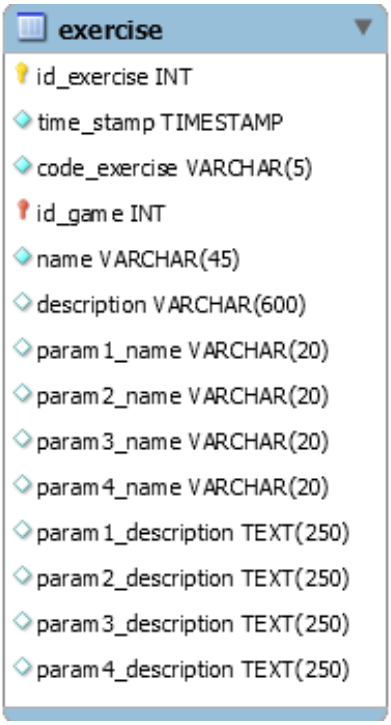


Figura 18 - Tabla Ejercicio

La configuración de cada ejercicio por parte de un terapeuta se realiza individualmente para un paciente, y se puede hacer tantas configuraciones quiera mientras que haya solo una en uso. La configuración en uso será la que se descargue el *middleware* Chiro a la hora de jugar al videojuego. Representada en la Figura 19, cada configuración tiene un identificador único, una fecha de creación, el identificador del paciente al que pertenece dicha configuración, y el del ejercicio para el que está dispuesta, así como cuatro campos para guardar los valores introducidos por el terapeuta. Además guarda el identificador del especialista que realizó la configuración, no siendo éste obligatorio, porque en el caso de que se dé de baja ese terapeuta, se pondrá este campo a *null*. De otra forma, se borraría la configuración y esto no es deseable. También en esta tabla se encuentra el campo booleano "en uso" que valdrá *true* si la configuración está en uso y *false* si está alguna otra. Como valor añadido, hay una columna Comentario de tipo texto para una breve anotación sobre la configuración.

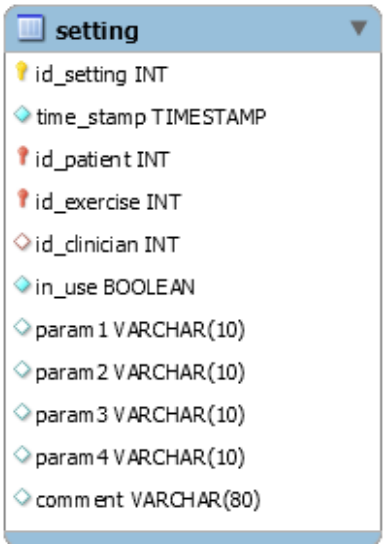


Figura 19 - Tabla Configuración

Por último, los resultados recibidos del videojuego se almacenan en la tabla Partidas, que contiene parte de la información de la configuración con la que se ha jugado. Esto se debe a que en algunas ocasiones, como en caso de fallo de conexión entre el *middleware* Chiro y el servidor Web, el videojuego utiliza la última configuración guardada en el PC. Cuando el *middleware* pueda enviar los resultados al servidor Web, la base de datos podría haber cambiado, por ejemplo que un terapeuta hubiese borrado la configuración con la que se ha jugado las últimas partidas. Dado que no se quiere descartar ningún resultado, pero carece de sentido almacenar un resultado relativo a una configuración y que esta haya sido borrada, el propio resultado lleva integrada la información de la configuración. En definitiva, la tabla Partida, tiene tres tipos de campos:

- Identificativos: identificador de la partida, fecha de la partida, identificador del paciente que ha jugado, identificador del ejercicio que ha jugado y código del ejercicio que ha jugado. Aunque en la plataforma actual no se contempla la posibilidad de guardar el resultado de un ejercicio si éste ha sido borrado, el código del ejercicio se conserva por si en un futuro se plantea esta necesidad, del mismo modo que se conserva la información de configuración.
- Configuración de la partida: identificador de la configuración con la que partida ha sido jugada y los cuatro valores de la configuración que puso el terapeuta. El identificador no es obligatorio por si, como se explica anteriormente, se borra una configuración, no se borre también las partidas asociadas. Si el juego tenía menos de cuatro parámetros a configurar, o ninguno, el valor de los parámetros que no había que configurar valdrá 0.
- Datos de la partida: La información que se ha considerado interesante guardar de la partida en esta solución han sido la duración de la partida, el número de movimientos, los movimientos correctos, y un dato booleano sobre si se ha superado o no. Conforme se avance en el desarrollo de los videojuegos estos parámetros podrán verse alterados.

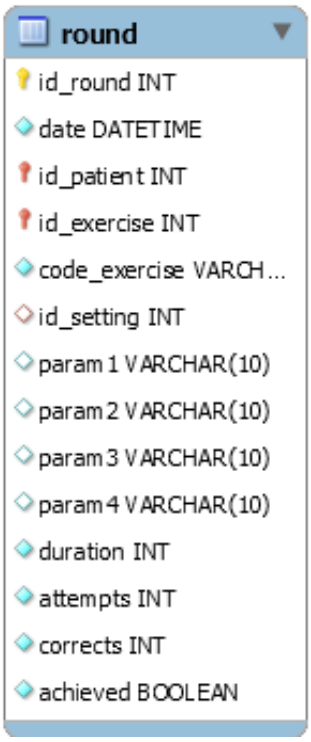


Figura 20 - Tabla Partida

Con esta concluyen las tablas empleadas en la base de datos. En la Figura 23 se muestra el diagrama EER (*Enhanced Entity-Relationship*) con la relación entre todas las tablas utilizadas. Previamente, en las figuras 21 y 22 se exponen los dos tipos de relaciones utilizados: la primera, identificadora 1:N (Uno a varios) y la segunda, no identificadora y no obligatoria 1:N.

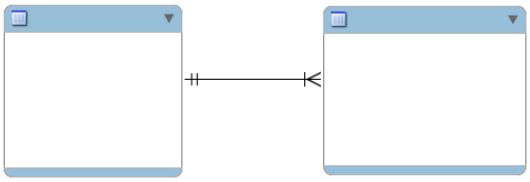


Figura 21 - Relación identificadora 1:N

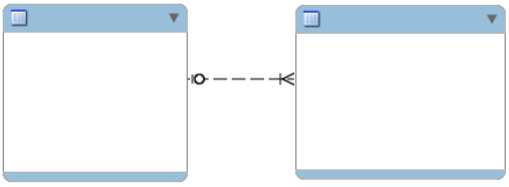


Figura 22 - Relación no identificadora y no obligatoria 1:N

En el siguiente diagrama EER, cuando dos tablas estén relacionadas identificadoramente, cuando se borre un objeto de la tabla "padre" se borrarán también sus "hijos". Ej.: Cuando se borre un juego se borrarán los ejercicios de este juego. Si por el contrario dos tablas están relacionadas no identificadoramente, si se borra el objeto "padre", no se borrará el "hijo. Ej.: Cuando se dé de baja un terapeuta, no se borrarán las configuraciones hechas por él.

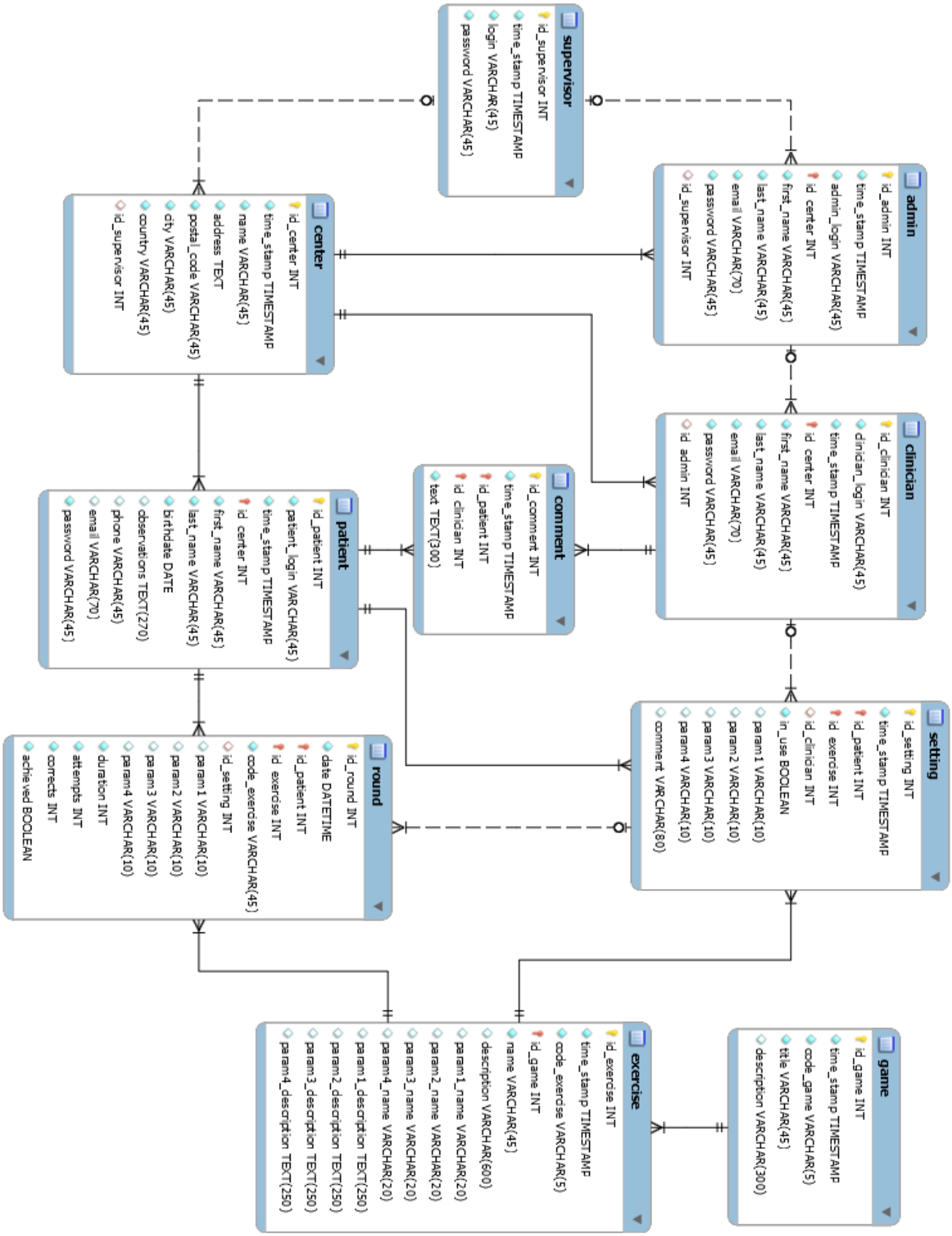


Figura 23 - Modelo de la base de datos de Blexer-Med

4.3. Plataforma Web

Los sistemas informáticos se puede abordar como una sencilla estructura de capas, y así se va a plantear (Figura 24) la plataforma Web núcleo de este proyecto. Esta arquitectura es la representación típica de un modelo Cliente-Servidor, por el desacoplamiento de las partes. Actualmente en diseño de aplicaciones por capas, el más utilizado es el de tres niveles. En primer lugar la Capa de Presentación es la que presenta la información al usuario de la plataforma y captura los eventos provocados por el usuario –*click* en un botón, enviar un formulario, solicitar una página concreta, etc. -. En segundo lugar, la Capa de Negocio es la que procesa las peticiones del usuario y accede a los datos de la base de datos para enviárselos a la Capa de Presentación y que ésta los muestre al usuario. Por último, la Capa de Datos es la capa en la que residen los datos. Esta capa es la encargada de almacenar/recuperar los datos al recibir solicitudes de la Capa de Negocio.

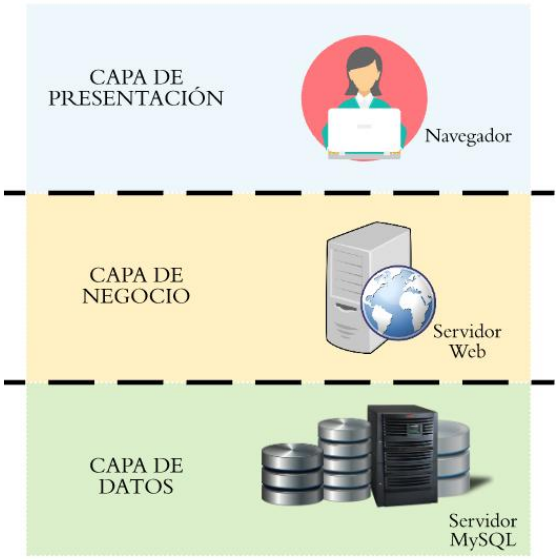


Figura 24 - Capas de la plataforma Web

Respecto a la Capa de Datos, ésta contiene la base de datos MySQL cuya implementación se ha detallado en el punto 4.2 “Base de Datos”, por lo que en este apartado únicamente se tratará la comunicación con ella.

4.3.1. Arquitectura de capas: Capa de Presentación

La capa de presentación representa la interfaz gráfica con la que interacciona el usuario, en este caso terapeutas, administradores y superadministradores. Esta capa está formada por dos bloques básicos: Componentes de Presentación y Componentes de Procesado.

El sistema podría ser una aplicación de escritorio, una plataforma Web, un programa, u otro, y los componentes de presentación son los que permiten al usuario interactuar con el sistema. “Dibujan” y dan formato a la información que se quiere presentar, y capturan la entrada/salida de datos –por ejemplo, en un formulario–. Estos elementos pueden ser de muchos tipos: botones, etiquetas, cuadros de texto, tablas, hipervínculos, imágenes, etc.

En desarrollo Web, como es el caso de la plataforma Blexer-Med, los componentes de presentación están especificados en **HTML** (*Hypertext Markup Language*), un lenguaje de marcado derivado del XML (*Extensible Markup Language*). Para este proyecto se ha utilizado la versión HTML5. Un documento html5 está formado por cabecera y cuerpo, y el navegador Web se encarga de interpretar este documento, que previamente descarga desde un servidor Web, y mostrarlo en la pantalla del dispositivo. En la Figura 25 aparece un ejemplo de documento html5 y de cómo se visualizaría en el navegador.

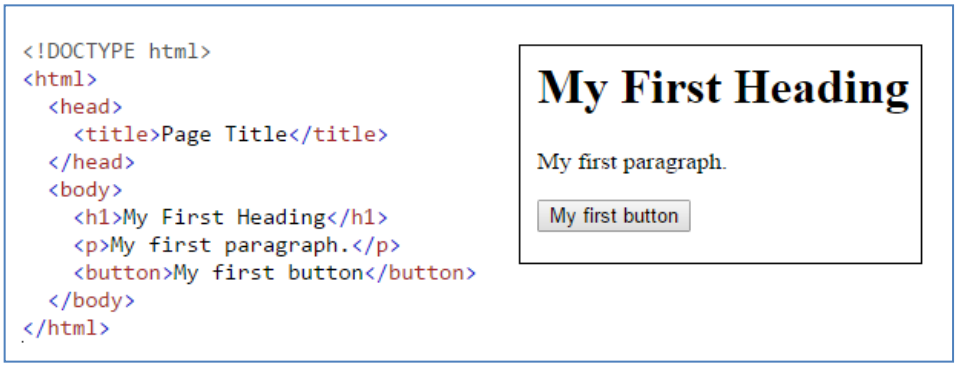


Figura 25 - Ejemplo de documento html y visualización

En el cuerpo de la figura anterior hay una etiqueta *h1* (para títulos), una *p* (para párrafos), y *button* (botones). Generalmente los componentes de presentación se acompañan con hojas de estilo **CSS** (*Cascading Style Sheets*) para dar el formato deseado a la Web. En la Figura 26 se muestra el mismo documento html de antes, que esta vez importa la hoja de estilos *styles.css*. A su izqda. se puede observar el contenido de dicha hoja de estilos, y finalmente cuál sería su visualización en el navegador.

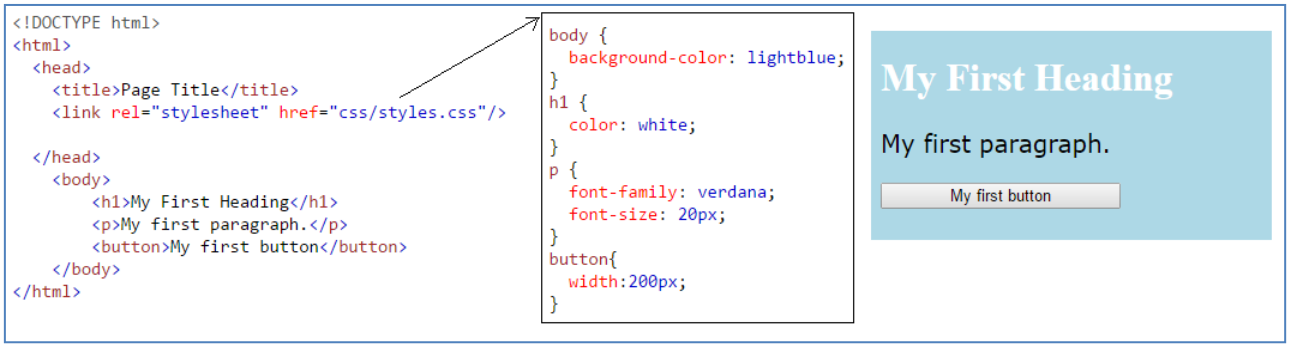


Figura 26 - Ejemplo de documento html con css y visualización

En el segundo bloque básico de la Capa de Presentación están los componentes de procesado. Estos componentes sirven para interacciones con el usuario que desencadenan efectos en la interfaz gráfica –Ej. Cambiar el color de un texto cuando se pulsa un botón– o para el tratamiento de datos, cuyo funcionamiento se estudiará a continuación. Son comúnmente utilizados los lenguajes de procesado PHP (*Hypertext Preprocessor*) o JS (*JavaScript*). Aunque ambos lenguajes van embebidos en el documento html, JavaScript se ejecuta en el lado del cliente (navegador) y PHP en el lado del servidor Web. A raíz de esto se ha elegido **JavaScript** para el procesado de elementos, porque al ejecutarse en front-end (lado del cliente) se ahorra tiempo de petición/respuesta al servidor, por ejemplo a la hora de validar

formularios, y la respuesta al usuario es prácticamente inmediata. A través de JavaScript, el html –a pesar de ser estático– puede ser accedido y modificado y así tener una interfaz dinámica.

Un ejemplo en la Figura 27 muestra el mismo documento de las dos figuras anteriores, pero ahora importa un archivo JavaScript *javascript.js* que tiene una función llamada "showAlert()". Esta función recoge la fecha instantánea del sistema y la asigna a una variable. A continuación muestra un diálogo con un mensaje que muestra la fecha recogida. Desde el documento html, en la etiqueta <*button*> , en el atributo *onclick*, se llama a la función *showAlert* definida anteriormente. Así, cada vez que se pulse sobre el botón se produce un evento *onclick* que ejecuta la función y muestra el diálogo con la fecha.

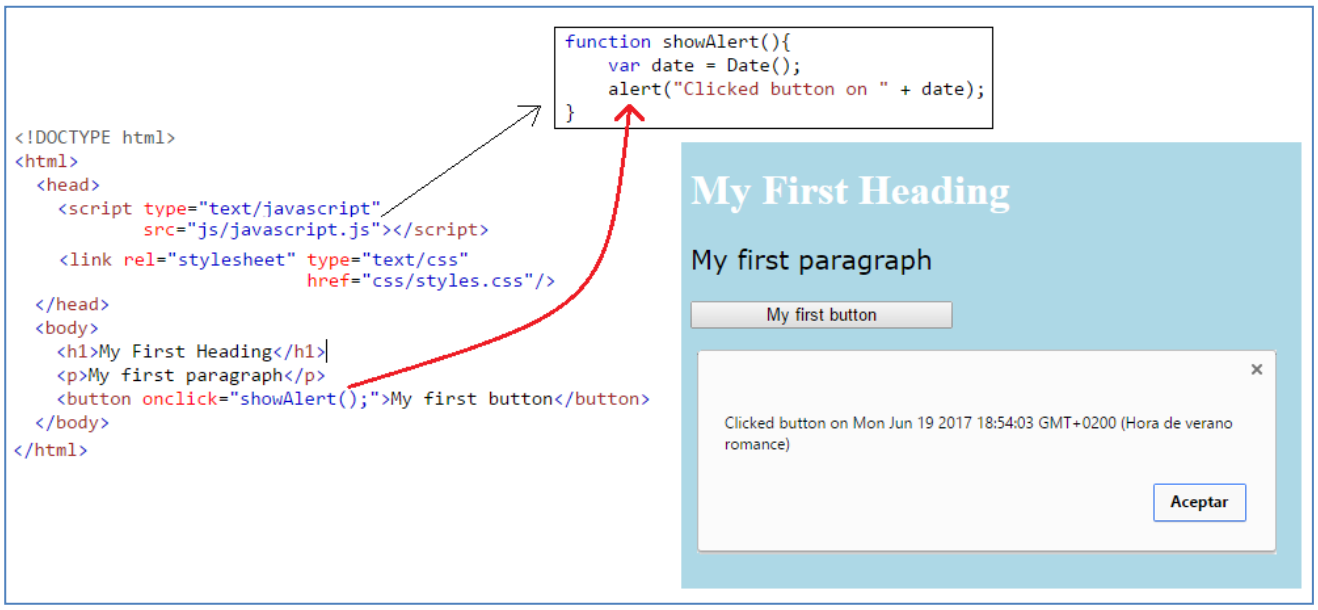


Figura 27 - Ejemplo de documento html (con css y JavaScript) y visualización

Para este proyecto se ha utilizado la librería **jQuery** de JavaScript que facilita el uso de este lenguaje con algunas expresiones. Por ejemplo, suponiendo que el botón del ejemplo anterior tiene un identificador "button1":

```
<button id="button1" onclick="showAlert();"> My first button </button>
```

y desde una función cualquiera se quiere seleccionar este botón, en JavaScript se haría:

```
var elemento = document.getElementById("button1");
```

mientras que con jQuery:

```
var elemento = $("# button1");
```

El símbolo "$" es la abreviatura de "jQuery" y el símbolo "#" significa que la palabra que lo sucede es el identificador de un elemento.

Que JavaScript se ejecute en el lado del cliente no quiere decir que no pueda interactuar con el servidor. La interfaz gráfica de Blexer-Med, desde los scripts de JavaScript envía frecuentemente peticiones **HTTP** al servidor para solicitar la información de un paciente, guardar una configuración, borrar un comentario, y demás acciones que ofrece la plataforma.

HTTP es un protocolo para la transferencia de hipertexto vía web y existen múltiples métodos para su transmisión, pero esta plataforma solo utiliza dos: *GET* para obtener un recurso, y *POST* para enviar datos y que estos sean procesados en el servidor. Para hacer peticiones HTTP con jQuery, se hace uso de dos métodos: *getJson* y Ajax (*Asynchronous JavaScript and XML*) a los que se les pasa como parámetro la URL (*Uniform Resource Locator*) del recurso al que se quiere acceder.

getJson y Ajax son equivalentes cuando se trata de hacer un una petición GET HTTP, pero con Ajax se puede hacer además POST:

- GET con getJson:

```
$.getJSON(url)
```

- GET con Ajax (básico):

```
$.ajax({
        type : "GET",
        dataType: "json",
        url: url
       });
```

- POST con Ajax (básico):

```
$.ajax({
      type : "POST",
      url : url,
      });
```

Las capas de presentación y negocio se han desarrollado en el entono de desarrollo Eclipse JEE Luna para Java. La capa de presentación está implementada en un conjunto de documentos html, js, imágenes, hojas de estilo en la carpeta *WebContent* del proyecto de Eclipse. Estos se disponen según la estructura de la Figura 28. Los documentos html están en la raíz *WebContent*, y los css, fuentes e imágenes en los directorios de mismo nombre, y los scripts de JavaScript en *js*.

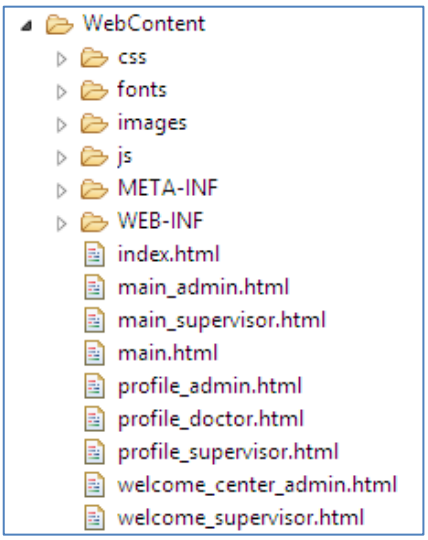


Figura 28 - Capa de Presentación en Blexer-Med: Documentos y directorios

Dentro del directorio *js*, y luego dentro del directorio *functions* (Figura 29), hay otro árbol de directorios con una rama para las funciones JavaScript de autenticación de usuarios,

otra para las funciones que se requieren en los html –separados por el tipo de usuario y el tipo de funcionalidad que ofrecen-. Por ejemplo, si se buscase la función por la que el superadministrador solicita añadir un centro médico al sistema esta estará en *functions/templates/supervisor_view/centers.js*, o si se necesita la función que consulta los resultados de las partidas de un usuario se debe buscar en *functions/templates/doctor_view/rounds_results.js*, ya que el terapeuta es el único que puede consultar los resultados. En los distintos html se importa únicamente los js de los que hace uso.

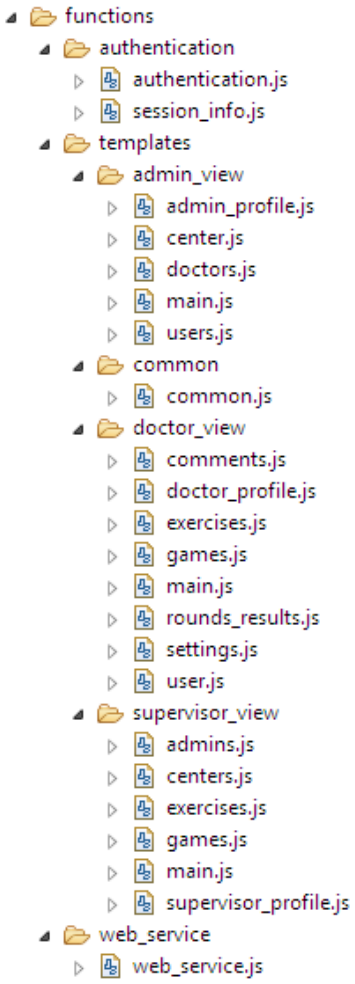


Figura 29 - Estructura de scripts utilizados en Blexer-Med

En resumen, la Capa de Presentación de Blexer-Med tiene componentes de presentación (HTML y CSS) y componentes de procesado (JavaScript). La Capa de Presentación y la Capa de Negocio se comunican entre sí mediante peticiones HTTP que se realizan usando funciones de jQuery para solicitar/enviar un recurso a la Capa de Negocio. Dichas funciones devuelven texto plano a interpretar por la capa de Presentación y mostrar al usuario de forma amigable.

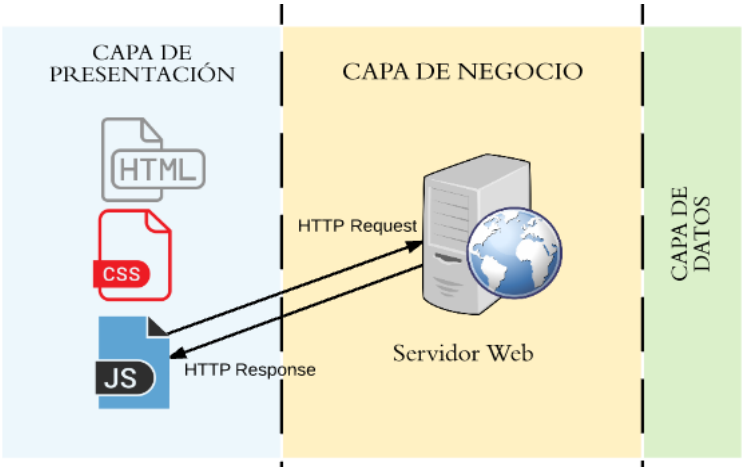


Figura 30 - Esquema de Capa de Presentación de Blexer-Med

4.3.2. Arquitectura de capas: Capa de Negocio

"La capa de negocio está formada por la lógica de aplicación, que prepara datos para su envío a la capa de cliente y procesa solicitudes desde la capa de cliente […]. La capa de negocio consiste en la lógica que realiza las funciones principales de la aplicación: procesamiento de datos, implementación de funciones de negocios, coordinación […] y administración de recursos externos como, por ejemplo, bases de datos o sistemas heredados." [11].

En Blexer-Med, la Capa de Negocio está programada en lenguaje **Java** pero otros muy comunes son Django, Ruby, PHP, y de nuevo JavaScript. Para la comprensión de este apartado es primordial saber que Java es un lenguaje orientado a objetos, y esto significa que se basa en la existencia de clases que tienen atributos y métodos. Una clase de Java es similar a un molde o plantilla con la que se pueden crear objetos del mismo tipo. "Un objeto es un ejemplar concreto de una clase, que se estructura y comporta según se definió en la clase. Al proceso de crear un objeto se le llama generalmente instanciar una clase."[12]. Los métodos de una clase Java son parecidos a las funciones de JavaScript, pueden recibir o no parámetros y pueden devolver o no un resultado. En la Figura 31 se utiliza la representación más común de una clase. En este ejemplo, la clase Coche tiene tres atributos privados (marcados con un +) y cuatro métodos públicos (marcados con -). Los tres primeros no reciben parámetros y devuelven un número entero, mientras que el último recibe un número entero pero no devuelve nada (void). Que un atributo o método sea público determina que sea accesible desde fuera de la propia clase o no. La sintaxis de los métodos utilizada es:

- *Nombre del método (parámetro: tipo de parámetro): tipo de resultado*

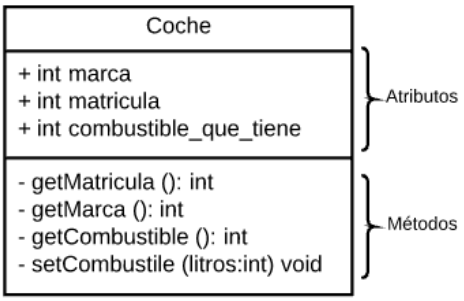


Figura 31 - Ejemplo de clase Java

Para que la Capa de Negocio sea accesible a través de HTTP se ha implementado un servicio web **RESTful**. Este es el nombre que reciben los sistemas distribuidos que siguen la arquitectura REST (*Representational State Transfer*), como la Word Wide Web. REST especifica restricciones en la arquitectura como una interfaz uniforme y si se aplica a un servicio web induce propiedades deseables tales como rendimiento, escalabilidad y modificabilidad. El estilo REST está diseñado para una arquitectura Cliente/Servidor y utiliza un protocolo de comunicación sin estado, normalmente HTTP. En esta arquitectura, los datos y funcionalidades del lado del servidor son considerados recursos y son accesibles a través de URLs (*Uniform Resource Locator*), conocidos típicamente como "enlaces" en la Web. Clientes y servidores intercambian de las representaciones de los recursos del WS (*Web Service*) mediante una interfaz y protocolo estandarizados. [13]

La interfaz elegida a través de la cual se accede al servicio web es la API **JAX-RS (***Java API for RESTful Web Services*), "diseñada para facilitar el desarrollo de aplicaciones de arquitectura REST" [14]. El *framework* **Jersey** es la implementación de referencia de JAX-RS, es decir, es el estándar del que las demás implementaciones derivan.

Para que el servicio Web intercepte las peticiones HTTP, Jersey dispone de un **Servlet** de la clase *ServletContainer*. Un Servlet sirve para extender las capacidades de un servidor que aloja aplicaciones que siguen el modelo de petición/respuesta. El **ServletContainer** de Jersey recibe y analiza las peticiones HTTP entrantes y selecciona la clase Java y método al que derivar la operación. En el fichero */WEB-INF/web.xml* del proyecto Java se debe detallar el servlet que tiene que atender las peticiones. A continuación se muestra el *web.xml* de Blexer-Med, donde primero se crea el servlet y se le da el nombre "web-service-servlet" y luego se mapea el servlet a una url:

```
<servlet>
      <servlet-name>web-service-servlet</servlet-name>
      <servlet-class>
            com.sun.jersey.spi.container.servlet.ServletContainer
      </servlet-class>
</servlet>
<servlet-mapping>
      <servlet-name>web-service-servlet</servlet-name>
      <url-pattern>/ws/*</url-pattern>
```

```
</servlet-mapping>
```

De esta manera el servlet atenderá una petición cuando el servidor reciba una petición con la forma *http://dominio:puerto/nombre_aplicación/url_servlet/*.

Para Blexer-Med, el dominio es "blexer-med.citsem.upm.es", el puerto es el de Tomcat (8080), el nombre de aplicación "blexer-med", y el patrón de url del servlet dado en *web.xml* es "/ws/*". Así, el servlet interceptará las peticiones dirigidas a

http://blexer-med.citsem.upm.es:8080/blexer-med/ws/ .

Además, para que la Capa de Presentación desde el JavaScript solicite un recurso concreto, éste debe estar especificado en la url:

http://dominio:puerto/nombre_aplicación/url_servlet/recurso/parametro1/.../parametroN

En función de la url recibida**, ServletContainer** "escanea" las clases Java de la aplicación para identificar a qué método va dirigida. Esta selección está basada en anotaciones de la API JAX-RS incluidas en la clase Java y en los métodos. Algunas de las anotaciones más importantes son:

- @PATH: Indica la ruta de acceso a una clase o un método de una clase
- @PathParam: Relaciona un parámetro recibido en la url con un parámetro de un método Java.
- @POST, @GET , @PUT , @DELETE: Indican el tipo de petición HTTP
- @Consumes , @Produces: Se refiere al tipo de parámetro que espera recibir o el tipo de resultado. En esta aplicación se han utilizado los tipos "application/json" -formato JSON (*JavaScript Object Notation*)- y "MediaType.*TEXT_PLAIN"* (texto plano).

En la Figura 32 se presenta un esquema de los componentes de la Capa de Negocio que permiten el acceso al servicio Web desde la Capa de Presentación.

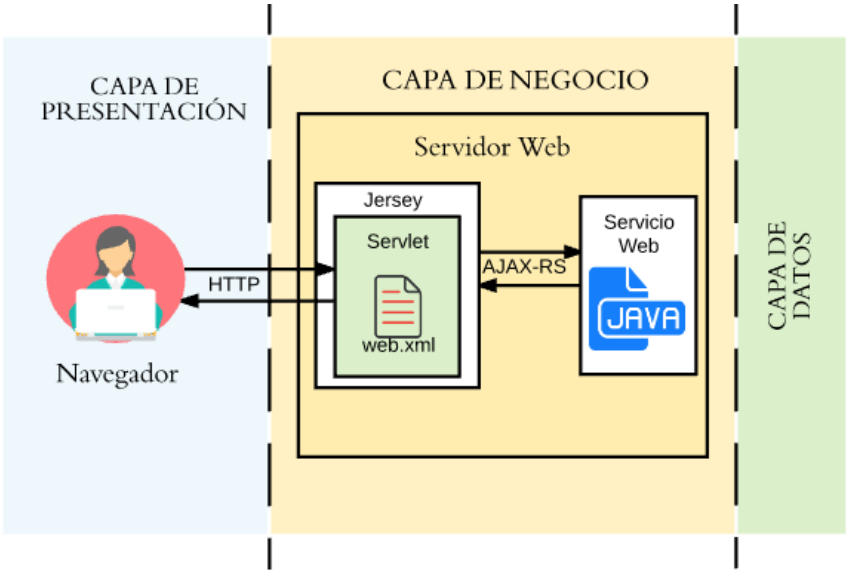


Figura 32 - Esquema de acceso a Capa de Negocio

En la plataforma Blexer-Med, solo hay una clase Java con métodos a los que puede acceder la máquina Cliente desde JavaScript y es ***WebService.java***. A continuación se puede observar como está declarada esta clase y uno de sus métodos:

```
@Path("/")
public class WebService {
	@GET
	@Produces("application/json")
	@Path("getDoctorById/{id}")
	public String getDoctorById(@PathParam("id")int id_doctor){
		String json=null;
		… (otras operaciones)
		return json;
	}
	… (otros métodos)
}
```

Por lo tanto, para obtener la información sobre el doctor con identificador 1000, desde la Capa de Presentación se haría una petición a la url:

http://blexer-med.citsem.upm.es/blexer-med/ws/getDoctorById/1000 que se haría invocando el método *$.getJson(url).*

En resumen, con los métodos de JavaScript *getJson* y *ajax* que se han mencionado en la Capa de Presentación, se accede a un método de un servicio web, que es localizado por un servlet que intercepta la petición HTTP. Dicho método del servicio web, en Java, es el que debe devolver una respuesta al cliente.

El servicio web de esta plataforma, implementado por *WebService.java* tiene los métodos que aparecen en la Figura 33, todos ellos accesibles mediante url. Por conveniencia, la ruta de acceso a cada método (@Path) es la misma que el nombre del método.

Cada método tiene una funcionalidad que ofrece a la Capa de Presentación y que indirectamente utiliza el terapeuta cuando realiza una acción en la interfaz de usuario. Hay métodos para añadir nuevos registros a cada una de las tablas de la base de datos, para consultarlos, modificarlos en el caso de que sea posible según el diseño de l plataforma, o borrarlos. Además, hay métodos para autenticar a los usuarios de la plataforma (terapeutas, administradores y superadministradores) y tres métodos que actualmente solo se usan desde el middleware Chiro: *authenticateUser* (para autenticar un paciente), *loadSettingsByUser* (para cargar las configuraciones en uso de todos los ejercicios para un usuario) y *saveResult* (para guardar un resultado de una partida enviado por Chiro a la plataforma).

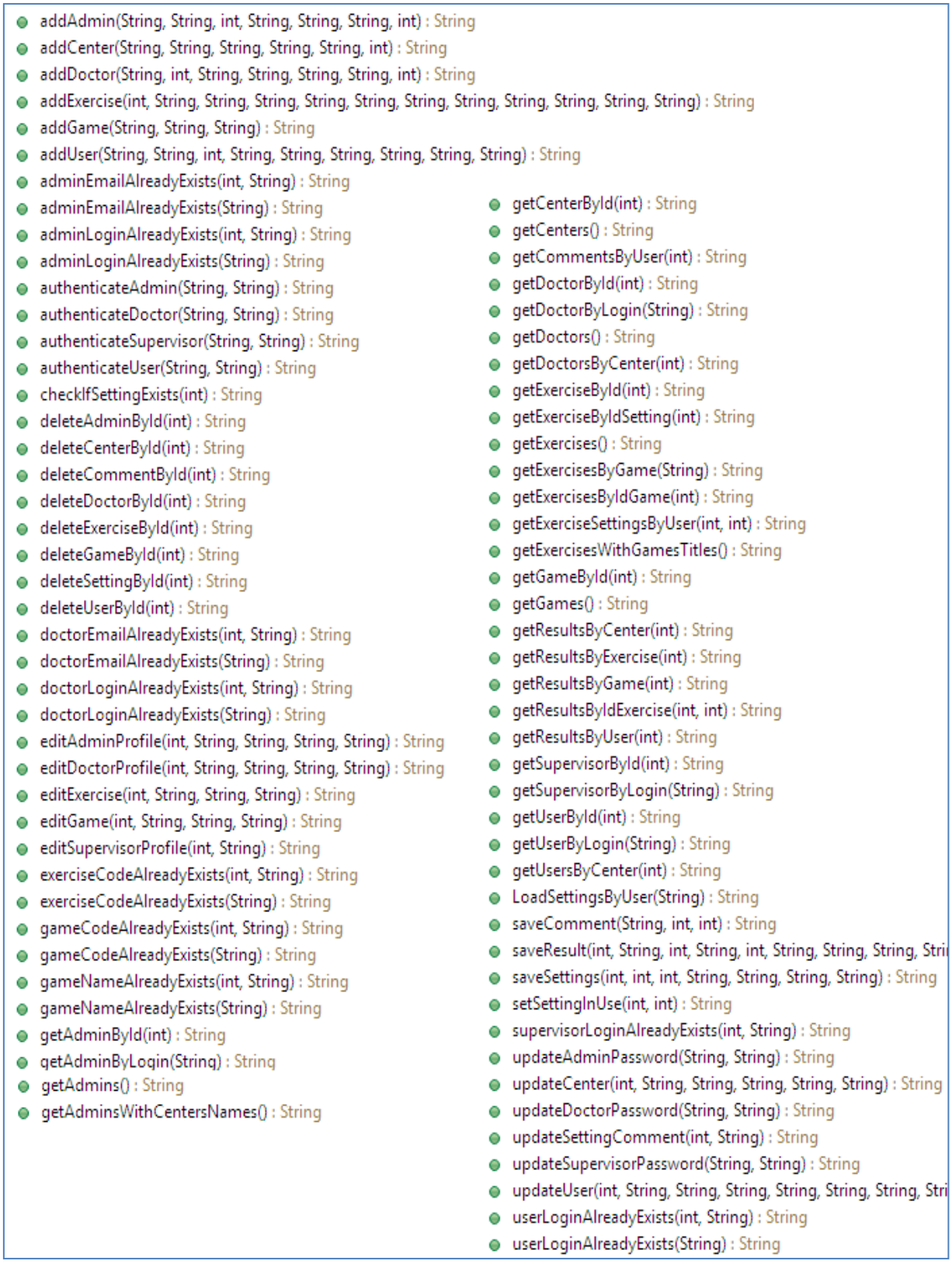


Figura 33 - Métodos del Servlet WebService.java

La clase *WebService.java* que representa el servicio web no interactúa directamente con la base de datos MySQL sino que utiliza unas clases intermediarias enviar las peticiones a la BD. Estas clases se han ubicado en el paquete "Management" del proyecto de Eclipse

(Figura 34), junto al paquete "WebService" que contiene WebService.java. En la figura, las clases ObjectManager representan las clases intermediarias.

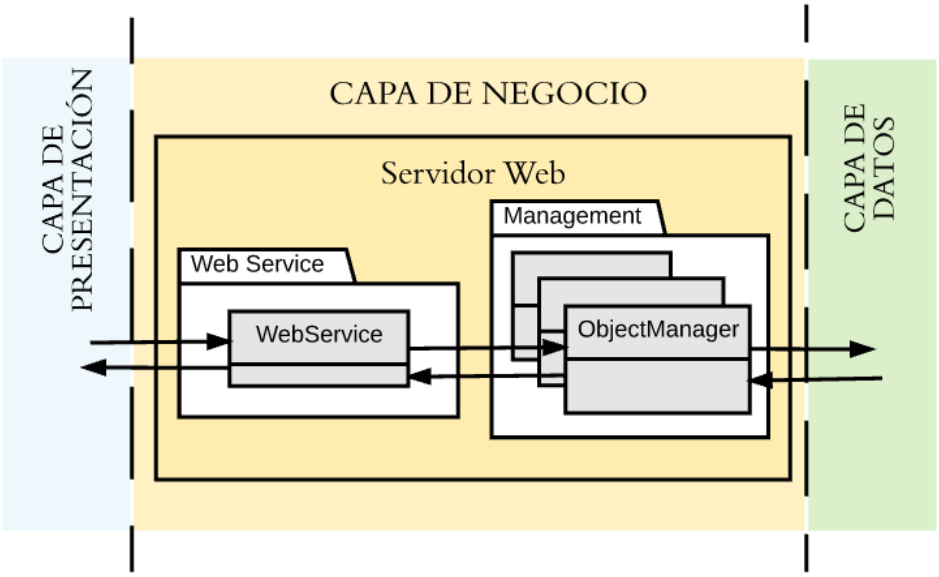


Figura 34 - Ubicación del paquete Management

Se han generado nueve clases intermediarias que se han llamado clases de Gestión de Objetos y que se resumen a continuación:

- **SupervisorManager**: Para el envío de peticiones a la BD relacionadas con los superadministradores (autenticación de supervisor, consultar supervisor dado su *login*, consultar supervisor dado su *id*, comprobar si un *login* existe en la base de datos, editar la información de supervisor, resetear contraseña). El esquema de los métodos de la clase SupervisorManager aparece en la figura 35. En esta y en las demás clases de gestión de objetos existe un constructor vacío para poder instanciarlas.

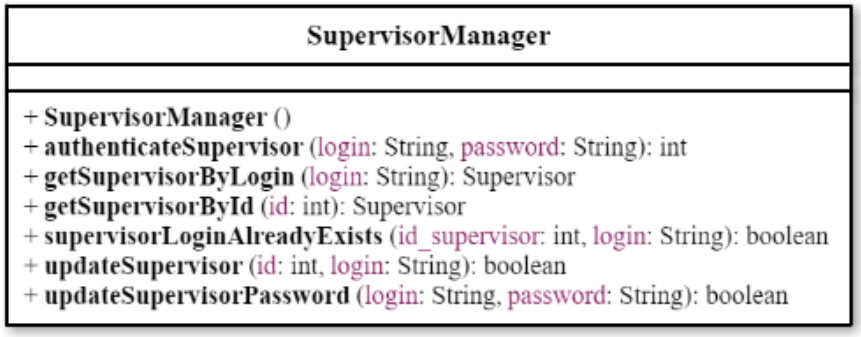


Figura 35 - Esquema de la clase SupervisorManager

- **AdminManager**: Esta clase cuyo esquema aparece en la figura 36 tiene los métodos para gestionar los administradores del sistema: autenticar, consultar administrador por su *login*, consultar administrador por su *id*, consultar si existe un administrador con un *email* dado, consultar si existe un administrador con un *login* dado, dar de alta/baja un administrador, editar su información, resetear su contraseña y obtener los administradores del sistema.

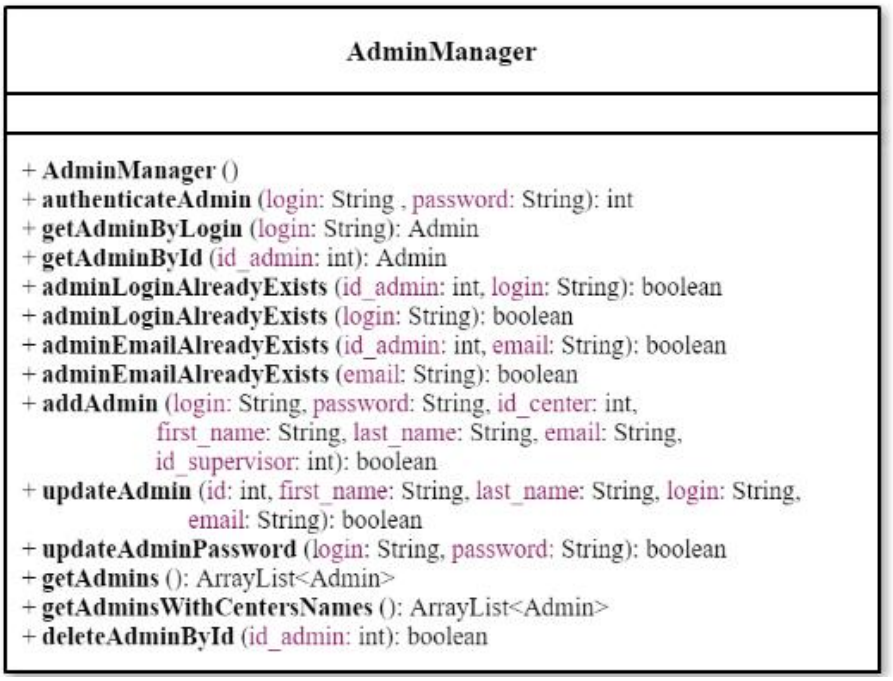


Figura 36 - Esquema de la clase AdminManager

- **DoctorManager**: Clase con los métodos necesarios para gestionar los terapeutas del sistema (autenticar, consultar terapeuta por su *login* o su *id*, consultar si existe un terapeuta con un *login* o *email* dado, dar de alta, modificar o borrar un terapeuta, resetear su contraseña, consultar los terapeutas de un centro o todos los de la base de datos).

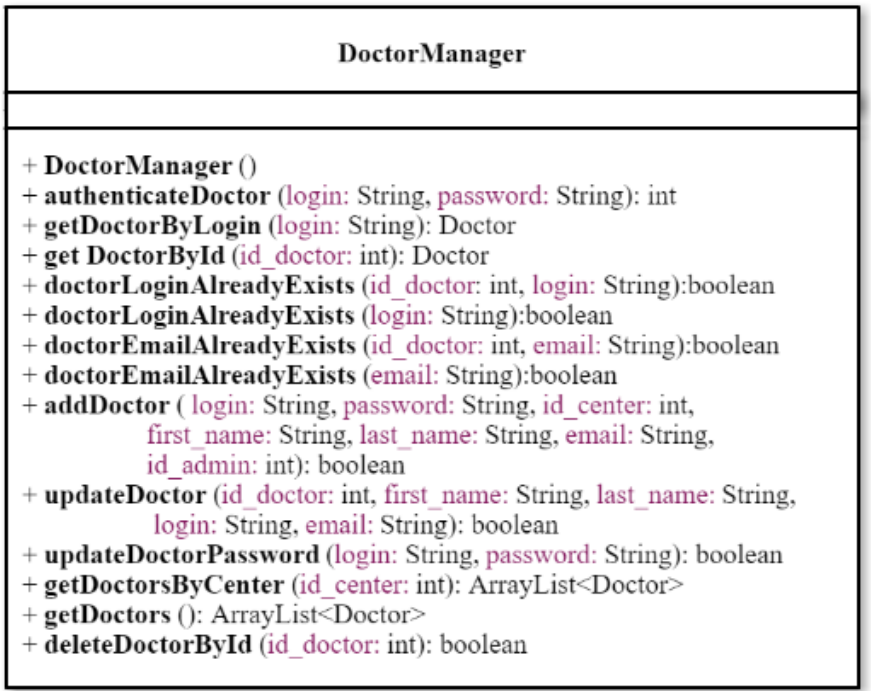


Figura 37 - Esquema de la clase DoctorManager

- **UserManager**: Los métodos de la clase UserManager (Figura 38) permiten trabajar con los pacientes de la plataforma Blexer-Med: autenticar un paciente (acceso desde *middleware* Chiro), solicitar los pacientes por un centro médico o por su *id* o *login*, añadir/editar/borrar un paciente, comprobar si existe un paciente con un *login* determinado, y gestionar comentarios de terapeutas (añadir, borrar).

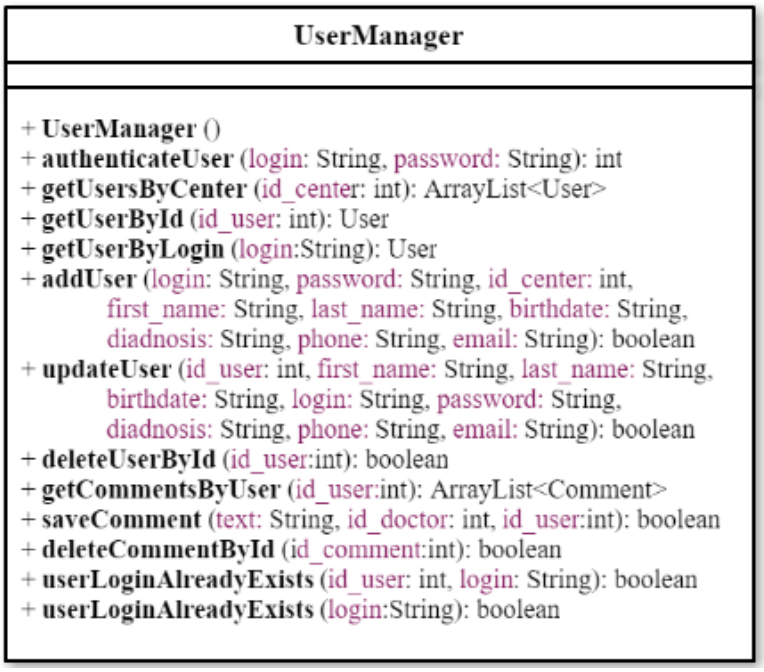


Figura 38 - Esquema de la clase UserManager

- **CenterManager**: A través de esta clase se puede añadir, editar y borrar un centro médico de la base de datos. También, consultar todos los centros del sistema o solo uno dado su *id:*

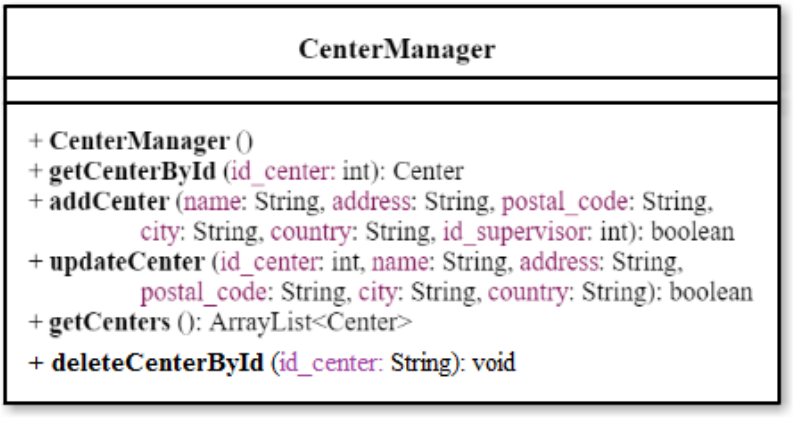


Figura 39 - Esquema de la clase CenterManager

- **GameManager**: Esta clase representada en la Figura 40 tiene métodos para gestionar los juegos de la BD como consultar todos los juegos del sistema, o seleccionar uno por su *id* o código, añadir, editar o borrar un juego, o comprobar si ya existe un registro de un juego con un nombre o código determinado.

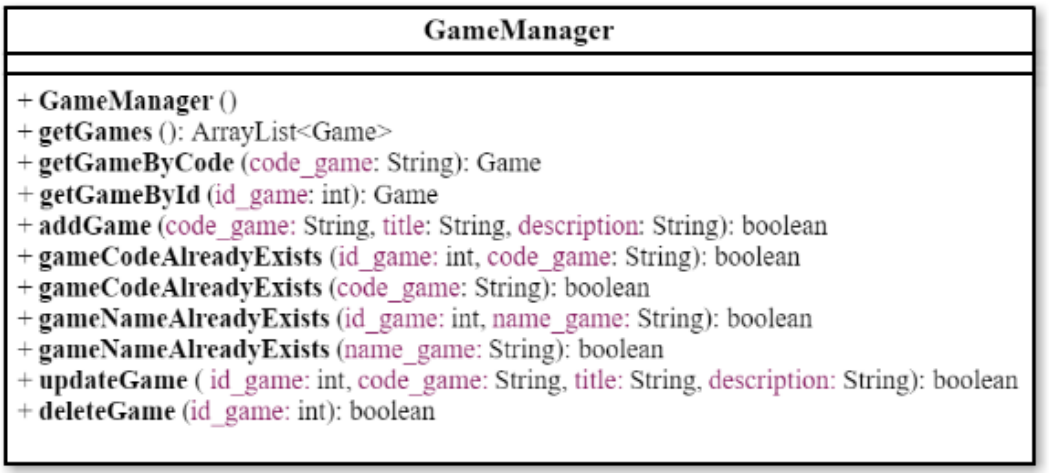


Figura 40 - Esquema de la clase GameManager

- **ExerciseManager**: Los métodos que interactúan sobre los ejercicios y los resultados de las partidas almacenados en BD se encuentran en la clase ExerciseManager. En cuanto a los ejercicios hay métodos para consultar todos o solo un ejercicio, comprobar si existe un ejercicio con un determinado código y como es habitual: añadir, editar y borrar ejercicios. Por otro lado, los métodos para gestionar los resultados son los presentes en la Figura 41: añadir resultados y consultarlos según el ejercicio, o según el usuario, o según el ejercicio y el usuario.

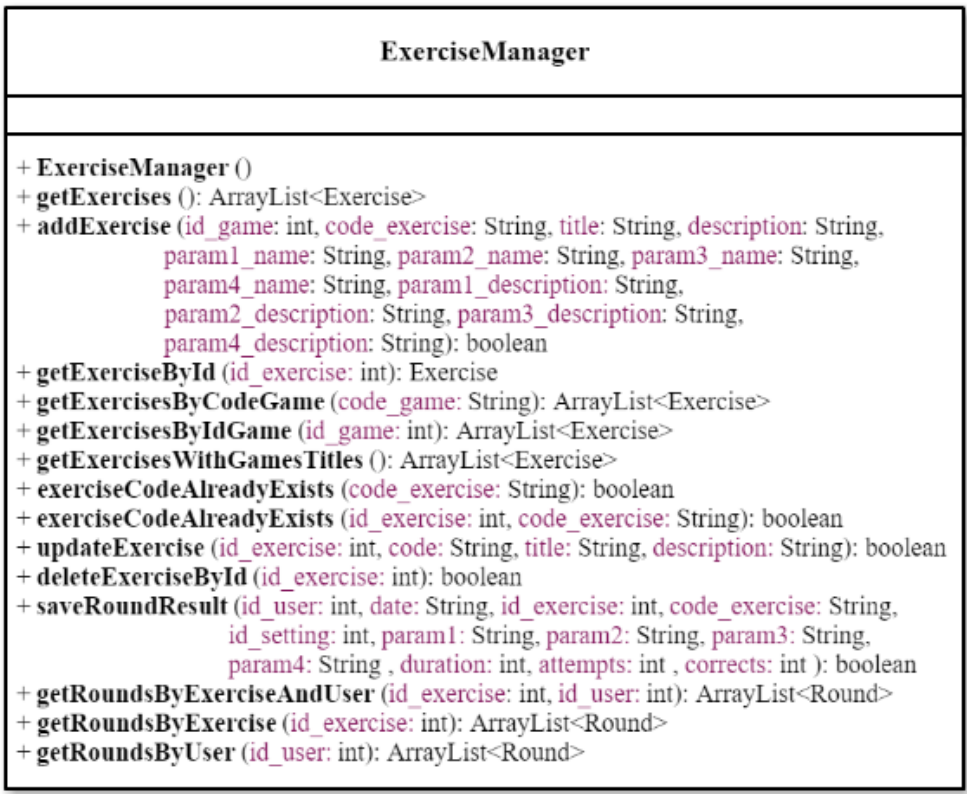


Figura 41 - Esquema de la clase ExerciseManager

- **SettingManager**: La clase SettingManager contiene los métodos para acceder a la BD y consultar, añadir, editar o borrar configuraciones. Además se puede consultar la

configuración que está en uso en un momento determinado, así como modificarla, y obtener la última y penúltima configuración que se ha hecho por orden cronológico.

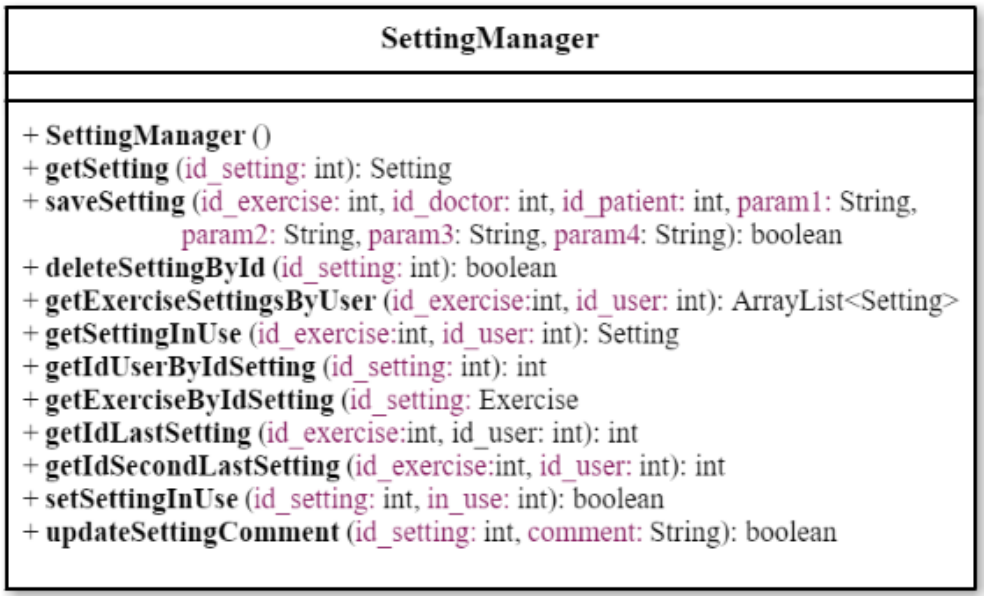


Figura 42 - Esquema de la clase SettingManager

- **SecurityManager**: Esta clase no interactúa con la base de datos, sino que es utilizada por las clases SupervisorManager, AdminManager y DoctorManager. Estas clases la instancian a la hora de autenticar un usuario, añadir uno nuevo, o resetear su contraseña. SecurityManager contiene un método *getHash* al que se le pasa como parámetro un *string* y lo convierte en un resumen MD5. La codificación del MD5 de una cadena de caracteres –en este caso la contraseña de un usuario- es representada con un *Hash*: una combinación de 32 símbolos hexadecimales (números del 0 al 9 y letras de la 'a' a la 'f'). Gracias a este método, es posible guardar el *hash* en la base de datos en lugar de la contraseña en texto plano. Se puede calcular el resumen MD5 de cualquier cadena de caracteres, pero no se puede deshacer. Si alguien averigua el *hash*, no se puede conocer la cadena original.

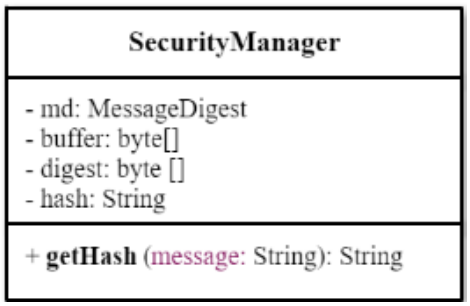


Figura 43 - Esquema de la clase SecurityManager

Estas nueve clases intermediarias se han agrupado en un paquete del proyecto de Eclipse llamado "Management", y a su vez hacen uso de otras clases para establecer la conexión con la BD. Estas clases necesarias para la comunicación con la BD se han situado en el paquete "DBConnection" que se muestra en la Figura 44.

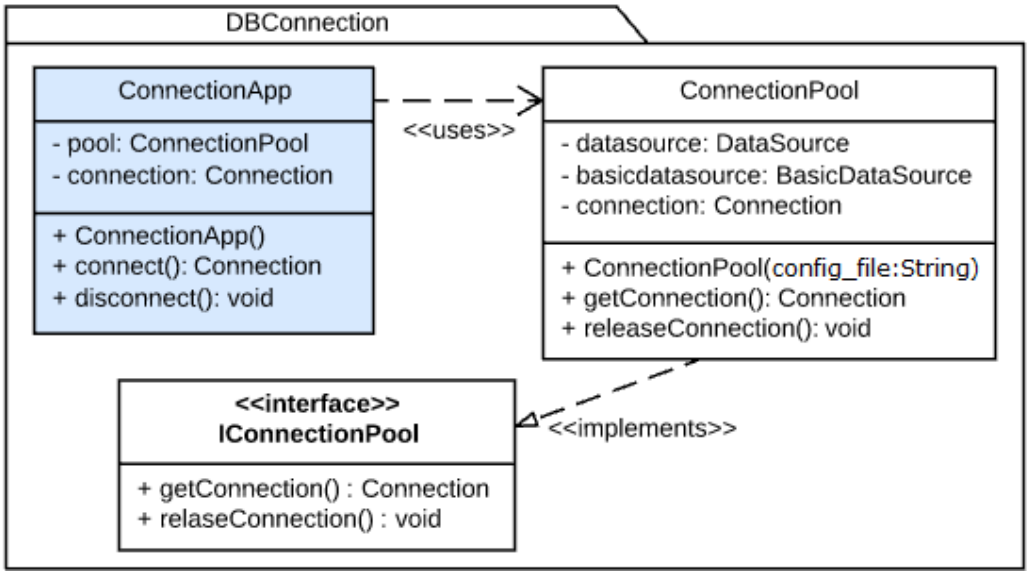


Figura 44 - Paquete DBConnection para comunicación con BD

Todas las clases intermediarias –menos SecurityManager- que se han explicado previamente extienden o heredan de clase *ConnectionApp*. "La herencia es un mecanismo que permite la definición de una clase a partir de la definición de otra ya existente. La herencia permite compartir automáticamente métodos y datos entre clases, subclases y objetos.".[15] ConnectionApp tiene los siguientes métodos –y por tanto las clases intermediarias también los tienen, pues los heredan- para manejar la conexión con la BD:

- *ConnectionApp*: Constructor de la clase que crea un objeto de la clase *ConnectionPool* que aparece en la Figura 44.

```
public ConnectionApp(){
      pool=new ConnectionPool("settingsDB");

}
```

A este objeto se le pasa en el constructor nombre del fichero de configuración para acceder a la base de datos, *settingsBD.properties*, en el que se especifica la url de acceso a la BD, el usuario para acceder, y su contraseña. En la url, hay que indicar el nombre de la BD a la que se quiere acceder. El contenido del fichero de configuración es el siguiente:

```
url                                                              =
     jdbc:mysql://localhost:3306/blexerdb?verifyServerCertificat
     e=
      false&useSSL=false&requireSSL=false

user = admin

password = admin
```

- *Connect*: Método que solicita la conexión con la BD.

```
public void connect(){

      … (instrucciones)

      this.connection=pool.getConnection();
```

```
        … (instrucciones)
    }
```

- *Disconnect:* Método que solicita la liberación la conexión con la BD.

```
    public void disconnect(){
        … (instrucciones)
        pool.releaseConnection();
        … (instrucciones)
    }
```

En el código de los métodos *connect* y *disconnect* se recurre a un objeto llamado "pool*"* de la clase *ConnectionPool*. Dicha clase implementa la interfaz *IconnectionPool*, como se puede observar en la Figura 44. Una interfaz es un conjunto de métodos con cuerpos vacíos de manera que cuando una clase implementa la interfaz, debe definir el comportamiento de todos los métodos que tenga esa interfaz. Así, si varias clases implementan una interfaz, los métodos pueden funcionar de manera distinta.

En la Figura 44, la clase *ConnectionPool* tiene tres atributos privados y tres métodos públicos que son utilizados desde la clase *ConnectionApp*:

- *ConnectionPool*: Constructor que recibe el nombre del fichero de configuración del que se ha mencionado el contenido anteriormente. El constructor obtiene los parámetros del fichero de configuración y establece la conexión con la base de datos:

```
    public ConnectionPool(String configFile){
        ResourceBundle
config=ResourceBundle.getBundle(configFile);
        basicdatasource= new BasicDataSource();

basicdatasource.setDriverClassName("com.mysql.jdbc.Driver");
        basicdatasource.setUrl(config.getString("url"));
        basicdatasource.setUsername(config.getString("user"));

basicdatasource.setPassword(config.getString("password"));
        datasource=basicdatasource;
        connection = datasource.getConnection();
    }
```

- *getConnection*: Método que devuelve la conexión creada en el constructor. Implementa el método vacío de la interfaz *IconnectionPool*:

```
    public Connection getConnection() {
        return this.connection;
    }
```

- *releaseConnection*: Método que cierra la conexión. Implementa el método vacío de la interfaz IconnectionPool:

```
public synchronized void releaseConnection() throws SQLException {
        if(connection!=null){
            connection.close();
            connection=null;
            basicdatasource.close();
        }
    }
```

Volviendo al método de ejemplo *getDoctorById* de *WebService.java*, ahora sabiendo que WebService utiliza clases intermedias para acceso a la BD –para este caso usa *DoctorManager*-, en este método se instancia *DoctorManager* y se llama a los métodos *connect* y *disconnect* (heredado de *ConnectionApp*):

```
public class WebService {
    @GET
    @Path("getDoctorById/{id}")
    public String getDoctorById(@PathParam("id")int id_doctor){
        DoctorManager doctorManager = new DoctorManager();
        doctorManager.connect();
        String json=null;
        … (instrucciones que solicitan a DoctorManager la info del terapeuta)
        doctorManager.disconnect();
        return json;
    }
    … (otros métodos)
}
```

Entran en escena un grupo de clases auxiliares que se han utilizado para manejar objetos entre la clase WebService y las clases intermediarias –DoctorManager, AdminManager, etc.-. Concretamente, existe una clase auxiliar para cada tabla de la BD. Estas clases tienen atributos privados análogos a las columnas de las tablas correspondientes de la base de datos. En la Figura 45 se muestra el paquete "Tables" donde se hallan estas clases auxiliares, y en la Figura 46 la estructura final de paquetes utilizada para la plataforma web Blexer-Med.

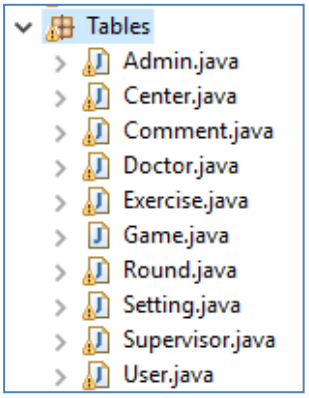


Figura 45 - Paquete Tables con clases auxiliares

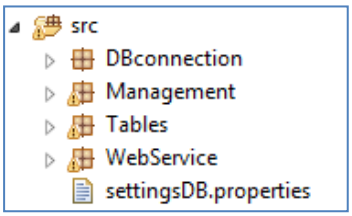


Figura 46 - Paquetes de la Capa de Negocio de Blexer-Med

De nuevo con el ejemplo de *getDoctorById* de *WebService.java* –se recuerda que los métodos de *WebService* son accedidos desde JavaScript en el navegador del usuario–, *WebService* instancia un objeto de tipo *DoctorManager*, al que solicita a la información de un doctor. En el código siguiente se detalla como la clase *DoctorManager*, en su método *getDoctorById*, crea un objeto de la clase auxiliar *Doctor*, y le da asigna a sus atributos los valores devueltos por la BD.

```
public class DoctorManager{
      public Doctor getDoctorById(int id_doctor){
            Doctor doctor = new Doctor();
            … (operaciones que solicitan a la BD la info del
doctor)
            doctor.setId("id_ejemplo");
            doctor.setLogin("login_ejemplo");
            doctor.setFirstName("nombre_ejemplo");
            doctor.setLastName("apellidos_ejemplo");
            doctor.setPassword("password_ejemplo");
            doctor.setEmail("email_ejemplo");
            doctor.setIdCenter("id_centro_ejemplo");
            return doctor;
      }
      … (otros métodos)
}
```

Este método, devuelve el objeto de clase *Doctor* que *WebService* debe recoger, y convertir en JSON –usando la clase Gson y su método toJSon– para devolver al JavaScript del navegador del cliente para poder mostrar al usuario:

```
public class WebService {


	@GET
	@Produces("application/json")
	@Path("getDoctorById/{id}")
	public String getDoctorById(@PathParam("id")int id_doctor){
		DoctorManager doctorManager = new DoctorManager();
		doctorManager.connect();
		String json=null;
		Doctor doctor = doctorManager.getDoctorById(id_doctor);
		if(doctor!=null){
			Gson gson = new Gson();
			json = gson.toJson(doctor);
		}
		doctorManager.disconnect();
		return json;
	}
	… (otros métodos)
}
```

Por último, cuando *WebService* le pide a una clase intermediaria (por ej. *DoctorManager*) que realice una operación sobre la BD, dicha clase intermediaria para acceder a la Capa de Datos crea un *PreparedStatement*, lo ejecuta –enviándolo a la BD-, y recibe un *ResultSet*.

Un *PreparedStatement* es una clase de Java que contiene un *statement* o sentencia SQL que ya ha sido compilada. Una sentencia SQL es la instrucción que va a ejecutar la BD: *INSERT* para añadir, *DELETE* para borrar, *UPDATE* para editar o *SELECT* para consultar. El formato para crear un *PreparedStatement* es:

PreparedStatement statement = conexiónEstablecidaConDB.prepareStatement(“string sentencia sql”);

El método “prepareStatement” de la clase *Connection* es el que compila la sentencia SQL. Después, se ejecuta el *PreparedStatement*, y esto devuelve un *ResultSet*. La clase *ResultSet*, permite acceder a los datos devueltos por la BD con *getString(“nombre de la columna de la tabla correspondiente”), getInt(“nombre Columna”), getBoolean(“nombre Columna*”), getDate(“nombre Columna”), etc.

Así, el método *getDoctorById* de *DoctorManager* solicita los datos a la BD de la siguiente manera, crea con ellos un objeto de clase *Doctor*, y lo devuelve en la sentencia return:

```
public Doctor getDoctorById(int id_doctor){
        Doctor doctor=null;
        PreparedStatement statement=null;
        ResultSet result=null;
        statement = this.connection.prepareStatement( "SELECT *FROM clinician
                    WHERE id_clinician='"+id_doctor+"';" );
        result = statement.executeQuery();
        if(result!=null){
              doctor = new Doctor();
              doctor.setId( new Integer( result.getInt("id_clinician") ) );
              doctor.setLogin( result.getString("clinician_login") );
              doctor.setFirstName( result.getString("first_name") );
              doctor.setLastName(result.getString("last_name") );
              doctor.setPassword( result.getString("password") );
              doctor.setEmail( result.getString("email") );
              doctor.setIdCenter( new Integer( result.getInt("id_center")));
              result.close();
        }
        statement.close();
        return doctor;
}
```

Para terminar, en la 47 se indican las clases Java que componen la Capa de Negocio de la plataforma y los cuatro paquetes a los que pertenecen, así como el fichero de configuración que contiene los datos de acceso a la BD. Las clases *ChiroSettingResponse* y *SettingInfo* se explican en el apartado 4.4 “Middleware Chiro” porque es el que hace uso de ellas. En la figura 48 se resumen la relación entre las clases que forman la Capa de Negocio.

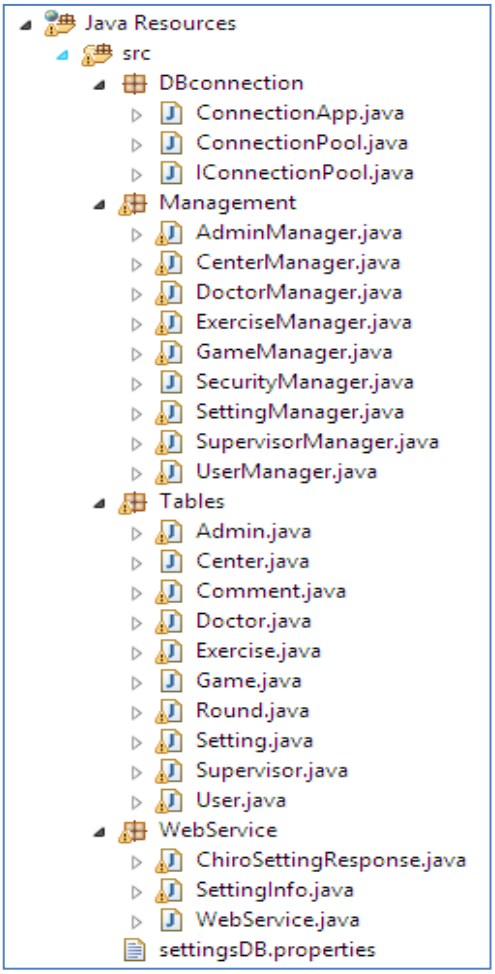


Figura 47 - Detalle de las clases que componen la Capa de Negocio de Blexer-Med

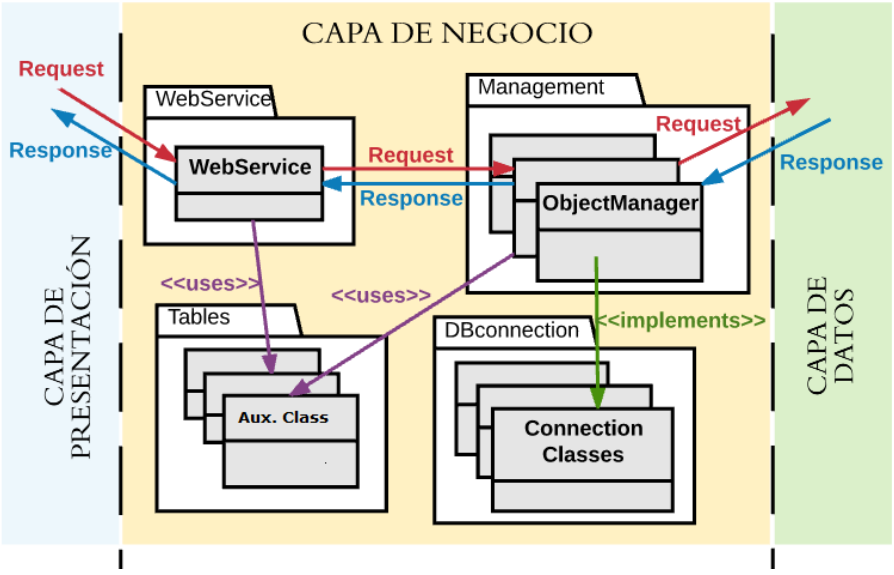


Figura 48 - Relación entre las clases que componen la Capa de Negocio de Blexer-Med

4.3.3. Control de sesiones

Al acceder a la URL de la plataforma web, la página de bienvenida es la página de autenticación del terapeuta que se muestra en la Figura 49. En ella se puede ver un breve formulario para introducir *login* y *contraseña*, junto al logo de Blexer-Med. En el *footer* de la página, bajo los logos de ETSIST, CITSEM y UPM, hay dos enlaces para cambiar a la vista de administrador del centro o de superadministrador.

Figura 49 - Página de bienvenida de Blexer-Med y autenticación para terapeutas

En relación con lo estudiado en los apartados 4.3.1 "Arquitectura de capas: Capa de Presentación" y 4.3.2 "Arquitectura de capas: Capa de Negocio" se presenta el diagrama de UML de secuencia de la Figura 50 con el proceso de autenticación de un terapeuta. "Un diagrama de secuencia es una forma de diagrama de interacción que muestra los objetos como líneas de vida y sus interacciones en el tiempo representadas como mensajes dibujados como flechas desde la línea de vida origen hasta la línea de vida destino." [16]. En el diagrama de la Figura 50 se detallan las acciones que se desencadenan al pulsar sobre el botón "Enter" del formulario de autenticación.

Cuando el usuario hace *click* sobre el botón "Enter" se produce un evento capturado por el manejador *onClick* asociado a este botón. Este manejador ejecuta la función *authenticateDoctor* del script *autentication*.js, como se observa en el diagrama de secuencia. Dicha función realiza una petición *HTTP GET* a la URL del WS, concretamente al recurso "authenticateDoctor", al que le pasa como parámetros el *login* y contraseña dentro de la url.

WebService.java, recibe la petición –gracias al Servlet de Jersey que localiza el recurso- y llama al método *authenticateDoctor* de *DoctorManager*. Éste, solicita a la clase

SecurityManager el resumen MD5 de la contraseña escrita por el usuario. Cuando *SecurityManager* le devuelve el resumen, *DoctorManager* envía una sentencia SQL a la BD para obtener los datos del usuario que intenta autenticarse. Dado que la contraseña en la base de datos se guarda codificada con MD5, *DoctorManager* compara la contraseña obtenida de la BD y el resumen devuelto por *SecurityManager*. Si es la misma, *DoctorManager* devuelve '1' a WebService.java. Por el contrario, si no es la misma o si no existe un usuario con el login dado, devuelve '-1'. En caso de que haya sido imposible conectar con la BD, se producirá una excepción, que *DoctorManager* capturará, y reenviará a *WebService*.

WebService.java analiza el resultado devuelto por DoctorManager: Si el resultado es '1' (caso representado en la figura), WebService responderá "OK" al HTTP GET, si el resultado es '-1', WebService responderá "NOK", y si se produjo una excepción en la conexión con la BD, WebService responderá "ERR".

Por último, el método *authenticateDoctor* de autentication.js, examina la respuesta y si esta es "OK", almacena los datos de sesión login y password y redirige la página de autenticación a la página principal del terapeuta.

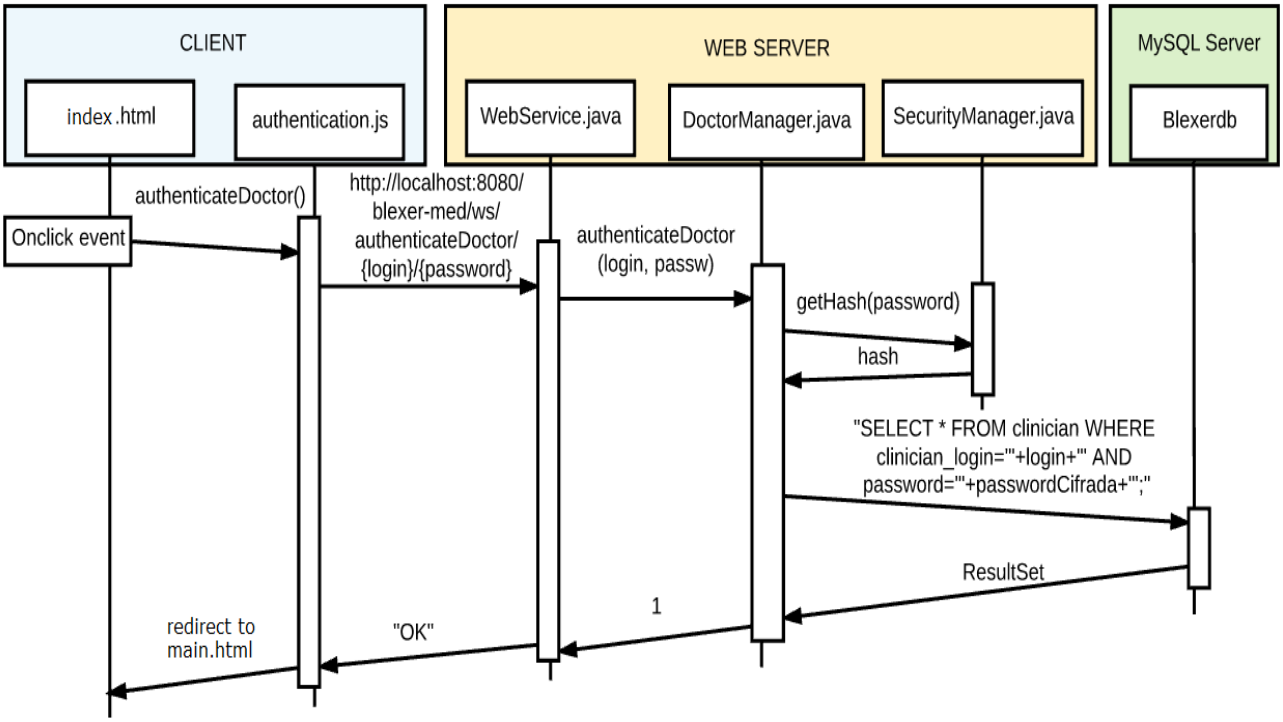


Figura 50 - Diagrama UML de secuencia de autenticación de un terapeuta

En el caso de que el nombre de usuario o la contraseña sean incorrectos –authentication.js recibe "NOK"- se muestra un aviso emergente de la Figura 51 (izquierda). Si ha habido un problema de conexión con la BD, el mensaje sería el de la derecha.

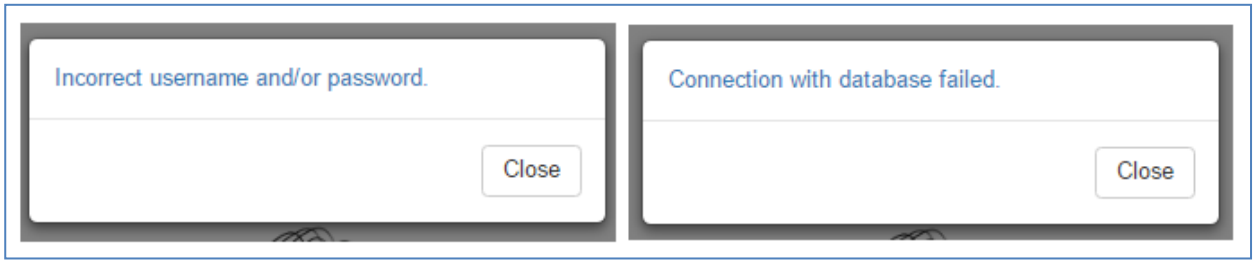


Figura 51 - Error en autenticación: mensajes emergentes.

Las vistas de autenticación para administradores de centros médicos, y para el superadministrador son muy similares, únicamente cambian la cabecera y los enlaces del *footer*, que dan la opción de volver a la vista de terapeuta (figuras 52 y 53).El diagrama de secuencia es análogo al de autenticación del terapeuta, pero el recurso de *WebService* al que accede el script de authentication.js es "authenticateAdmin" o "authenticateSuperadmin" y las clases intermediarias que acceden a la BD son AdminManager o SupervisorManager respectivamente. La gestión de resultados y mensajes de error es la misma.

Figura 52 - Página de autenticación para superadministrador

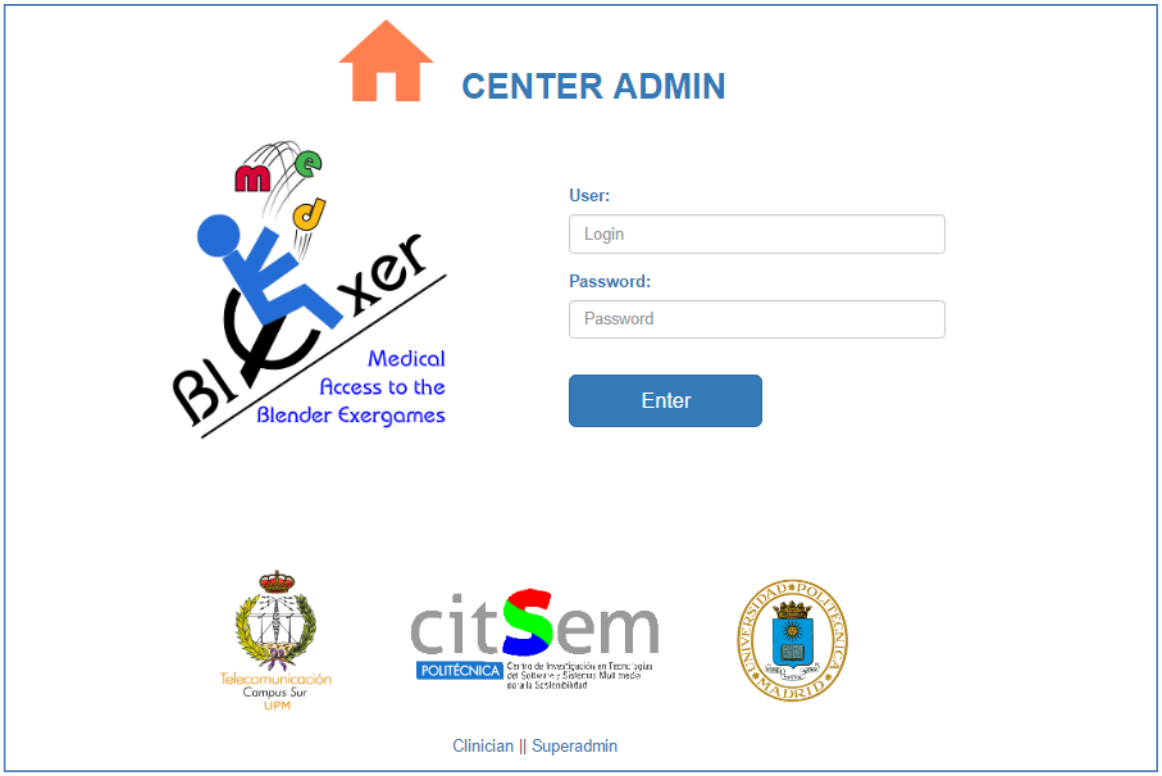


Figura 53 - Página de autenticación para superadministrador

4.3.4. Interfaz gráfica para superadministrador

Cuando un superadministrador se autentica correctamente, la aplicación lo conduce a la página principal que se visualiza en la Figura 54. Ésta presenta una cabecera o barra de navegación con tres botones: a la derecha uno con el nombre del superadministrador para ir a la ventana de edición de cuenta y otro en verde para cerrar sesión, y a la izquierda uno con el logo de Blexer-Med para volver a la página principal.

La ventana principal funciona como un "escritorio" y todo lo que puede hacer el superadministrador se desarrolla en esta ventana, salvo si va a editar su cuenta, que pasa a otra página. Así pues, la ventana principal tiene cuatro posibles vistas accesibles a través de cuatro botones a modo de menú: gestión de los centros de todo el sistema, gestión de los administradores de cada centro, gestión de juegos de la plataforma y por último gestión de los ejercicios de cada juego. El aspecto de la ventana de trabajo del superadministrador de la plataforma se muestra en la figura 54.

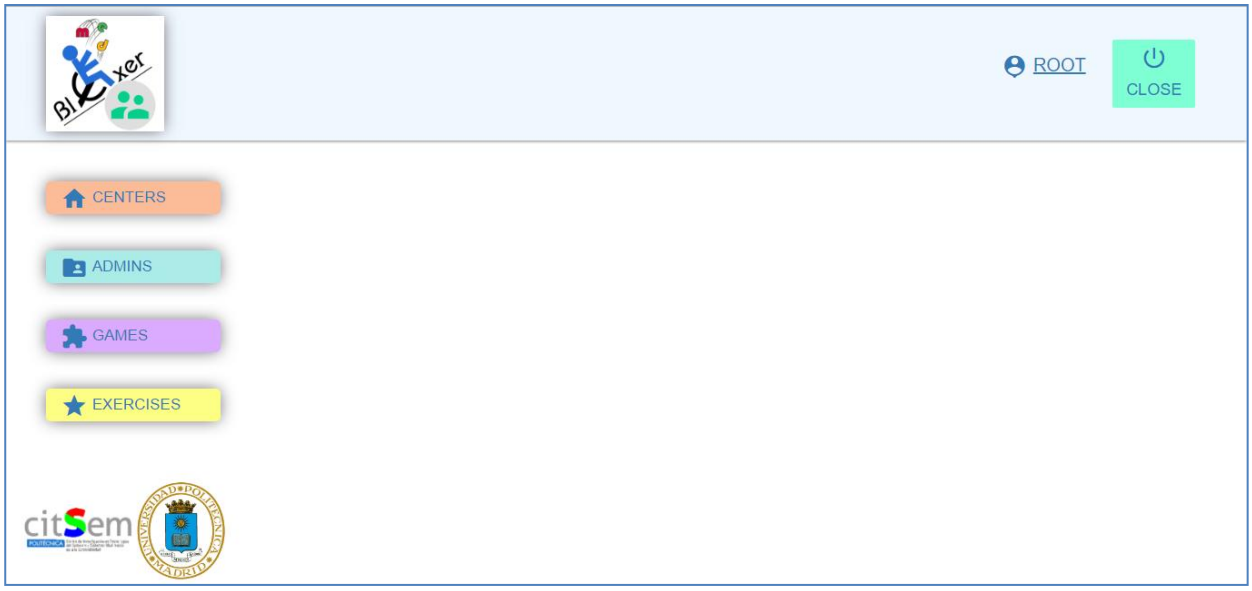


Figura 54 - Superadministrador: Página principal

La gestión de la cuenta del superadministrador se puede hacer desde la página de perfil (Figura 55), a la que se accede haciendo *click* sobre el nombre del superadministrador en la cabecera de la página principal. Esta página de configuración es la misma para superadministrador, administrador y terapeutas. Existen dos posibilidades en esta página: modificar el perfil y modificar la contraseña (ver *Anexo A.I. Gestión de la cuenta de superadministrador*).

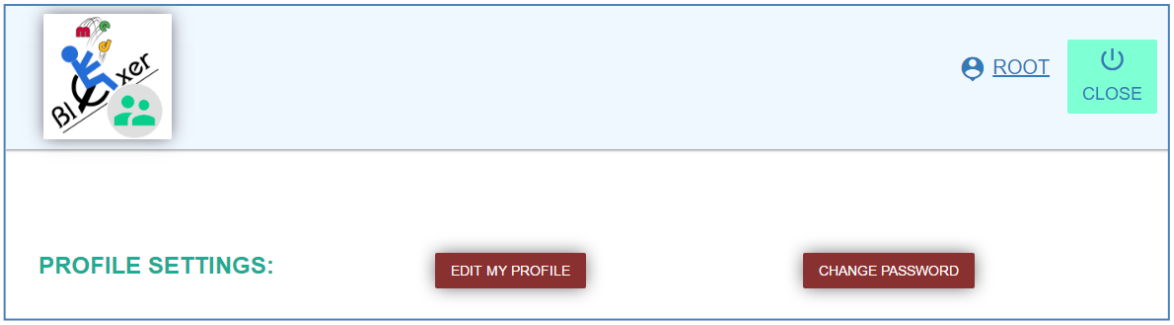


Figura 55 - Superadministrador: perfil

En la página principal, el primer botón del menú conduce a la vista de los centros médicos. En ella, una tabla con la información de todos los centros del sistema ocupa todo el ancho del escritorio, como se observa en la Figura 56. Aunque en la captura solo hay dos centros médicos de ejemplo, un desplegable en la parte superior izquierda de la tabla da a elegir hasta cuantos centros se quieren mostrar por cada página de la tabla: 10, 25, 50 o 100. Este desplegable está presente en todas las tablas que suceden a esta.

En la tabla de la figura, hay unas columnas dedicadas a información del centro –identificador único, nombre, dirección, código postal, ciudad y país-, otra columna que indica al superadministrador que dio de alta este centro, y dos últimas columnas para las opciones de "Editar centro" y "Borrar centro". Para más información y figuras, visitar *Anexo A.II. Gestión de centros médicos*.

- Es muy importante destacar que si se solicita borrar un centro, un paciente, un juego o un ejercicio, se ofrece al usuario la opción de descargar los resultados de partidas jugadas por los usuarios del centro a borrar (o jugadas por el paciente a borrar, etc.) para no perder información relevante.

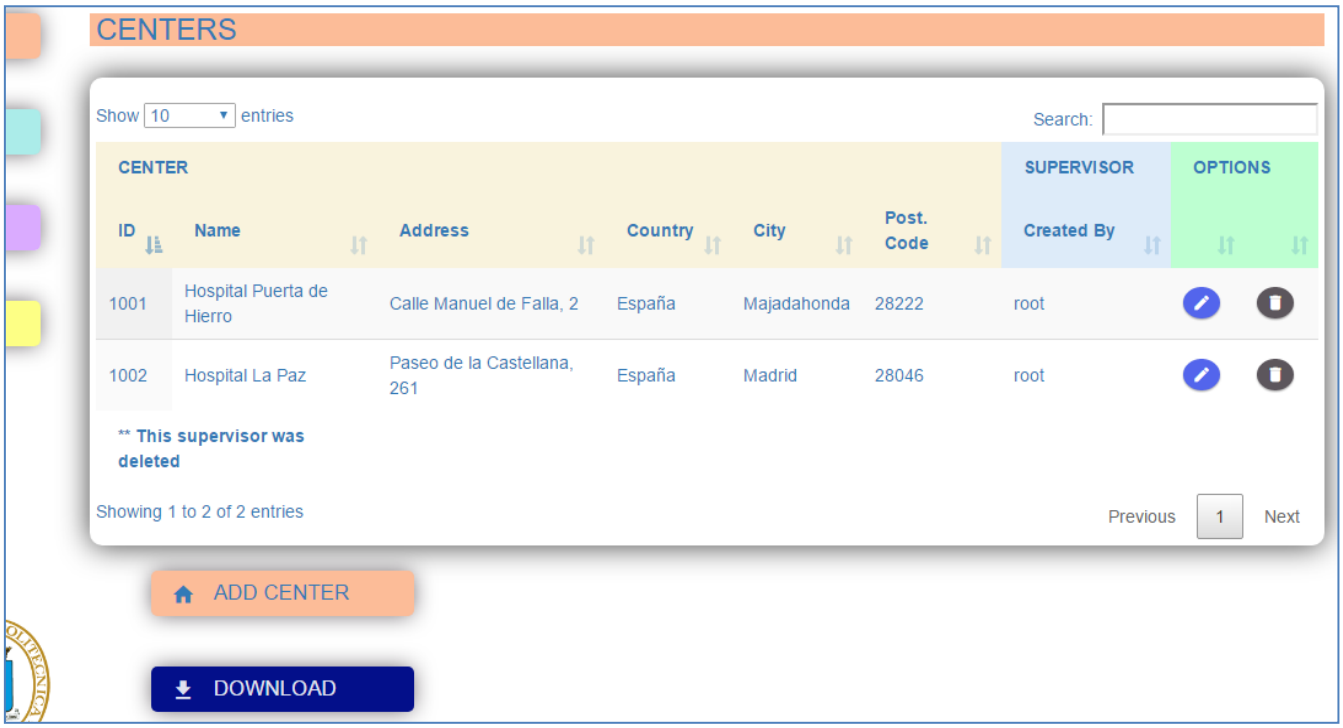


Figura 56 - Superadministrador: Vista de centros médicos del sistema

Además dos botones debajo de la tabla dan las opciones de "Añadir centro" y "Descargar todos los centros del sistema" (*Anexo A.II. Gestión de centros médicos*).

En segundo lugar, la vista de administradores de los centros médicos es accesible mediante el segundo botón del menú. Al entrar en esta vista, se visualiza una tabla similar a la tabla de los centros, aunque con otro aspecto y distintas columnas (Figura 57). Cinco columnas detallan la información del administrador: identificador, nombre de usuario, nombre, apellidos y *email*. A continuación hay dos columnas para la información del centro –identificador y nombre del centro- al que está asignado dicho administrador. Igual que en la anterior tabla de centros médicos, en esta también hay una columna para el superadministrador que dio de alta a cada administrador. Por último, como se puede ver en la figura, se ofrecen al superadministrador tres posibilidades a realizar sobre un administrador concreto: "Editar", "Restablecer contraseña" o "Dar de baja*"* (*Anexo A.III. Gestión de administradores de los centros*).

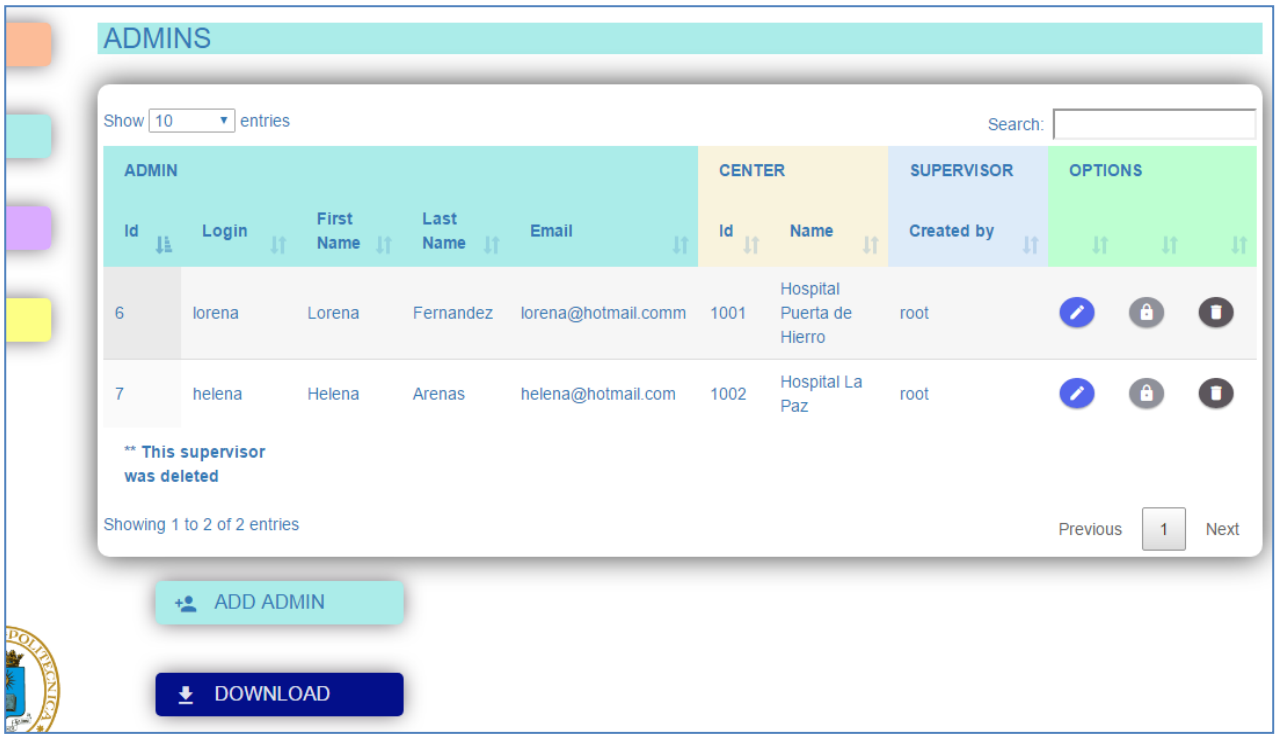


Figura 57 - Superadministrador: Vista de administradores del sistema

El tercer botón del menú del superadministrador conduce a la vista de gestión de juegos que se observa en la figura 58.

Figura 58 - Superadministrador: Vista de los juegos del sistema

En este caso, hay tres juegos de ejemplo: "Phiby's Adventures", "Cars Race" y "Fly around the world". El juego multiaventura "Phiby's Adventures" está siendo desarrollado por dos alumnas de la ETSIST (Yadira Peláez y Cristina Esteban) en su PFG, los otros dos son ficticios. Para cada juego del sistema se muestra su identificador único, un código de hasta cinco caracteres que pueden ser alfanuméricos –este código será utilizado por el *middleware* Chiro-, el título del juego y una descripción. Además, cada juego se puede modificar o eliminar (ver *Anexo A.IV. Gestión de juegos*). Igual que para centros y administradores, hay un botón bajo la tabla para añadir un juego al sistema, y otro para descargar la información de los juegos que haya en un momento determinado. El formulario de "Añadir juego" se representa en la Figura 59, donde se pide obligatoriamente el código de 5 caracteres alfanumérico mencionado anteriormente, un título para el juego y una descripción.

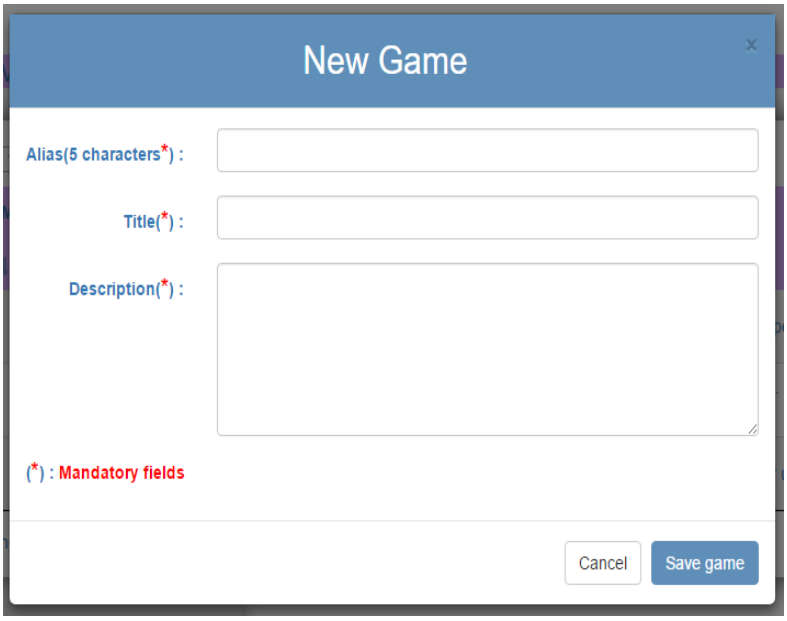


Figura 59 - Superadministrador: Formulario para añadir un juego

Para terminar, el superadministrador es el encargado de gestionar los ejercicios correspondientes a cada juego. En la vista de gestión de ejercicios que aparece en la Figura 60 se muestra la tabla de posibles ejercicios del sistema.

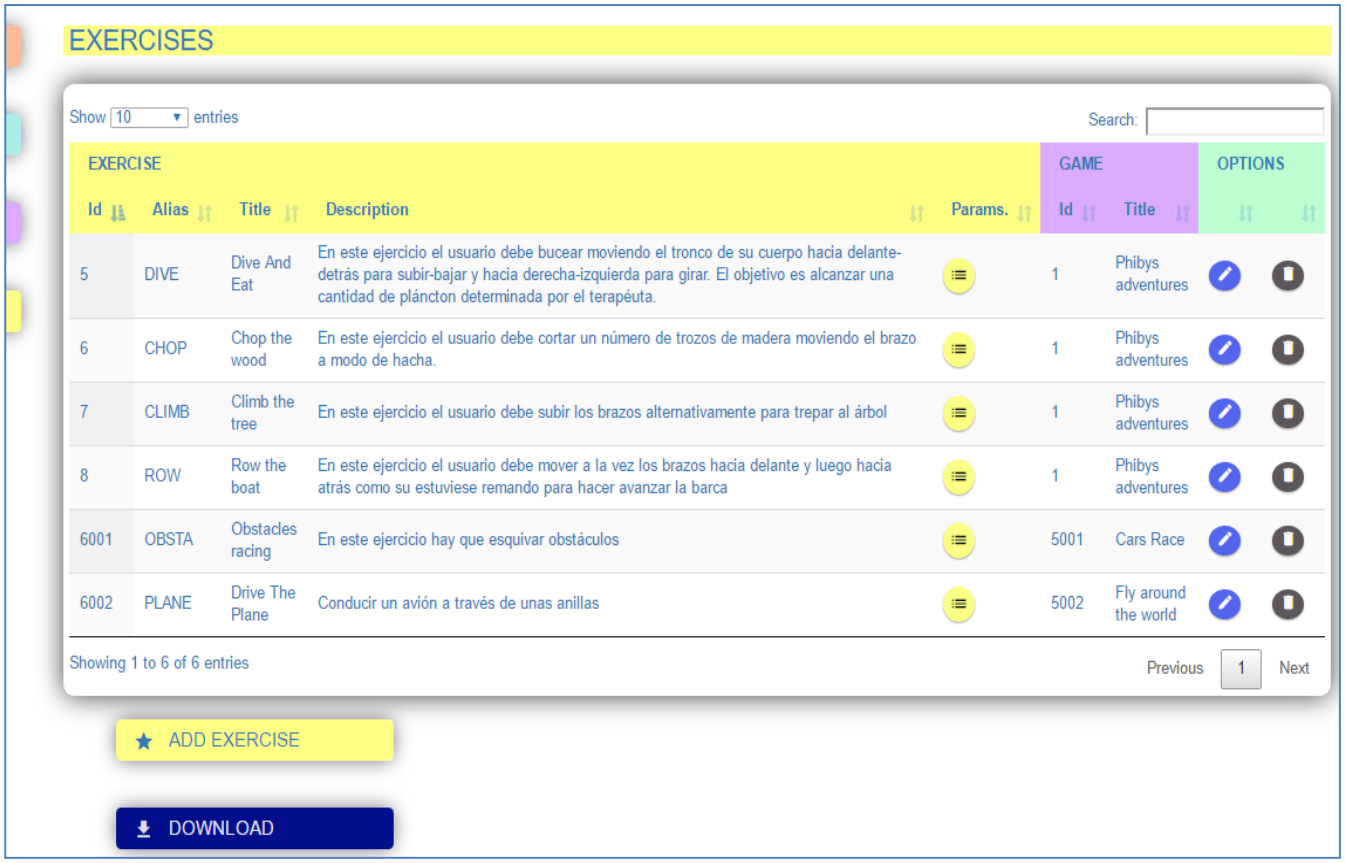


Figura 60 - Superadministrador: Vista de ejercicios de todos los juegos del sistema

Las primeras columnas –con cabecera en amarillo– muestran la información propia del ejercicio (id, código para Chiro, título y descripción), y un botón para consultar los

parámetros configurables de cada ejercicio, establecidos por el administrador al crear el ejercicio. En la tabla también se observa a qué juego pertenece cada ejercicio, y las opciones de añadir o borrar (*Anexo A.V. Gestión de ejercicios*). Lógicamente se pueden añadir ejercicios nuevos a un juego, y descargar la información de los juegos del sistema.

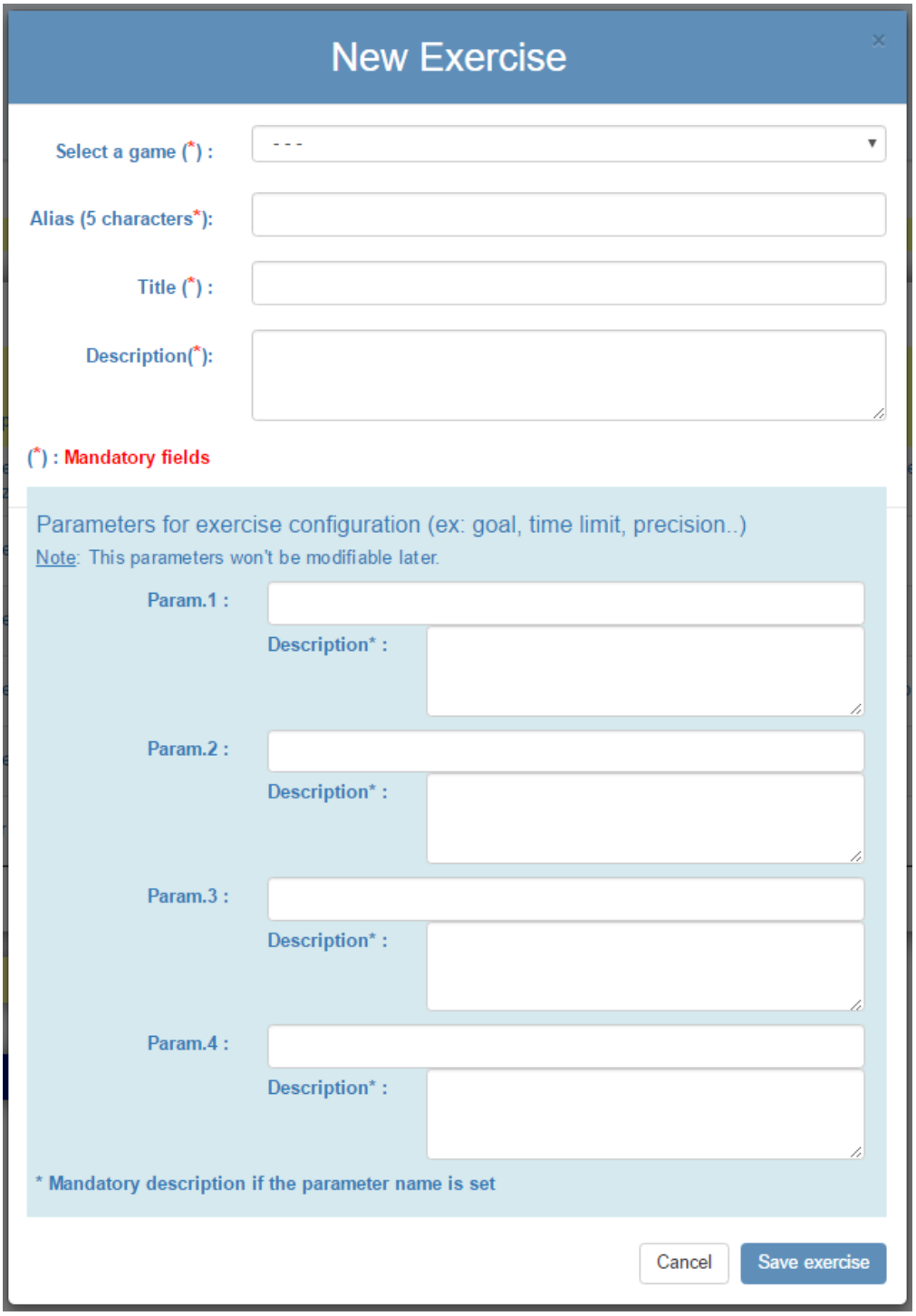


Figura 61 - Superadministrador: Formulario para crear ejercicio

El formulario de creación de un ejercicio se muestra en la figura 61. Este formulario de "Añadir ejercicio" es de las partes más importantes del proyecto, pues aquí es donde el superadministrador (de la mano de un especialista y del desarrollador de los videojuegos)

establece qué parámetros va a poder configurar el terapeuta. Cumpliendo así el objetivo del proyecto de que los juegos sean configurables, y de que cada juego pueda tener una configuración distinta.

Los parámetros de configuración establecidos en el momento de creación del ejercicio no serán modificables, puesto que el desarrollo del videojuego se hará en torno a los parámetros iniciales y podría ocurrir un error grave si los parámetros cambiasen en la plataforma pero no en la programación del videojuego. Para cada parámetro, el superadministrador debe escribir una breve explicación sobre su significado, que podrá consultar tanto él como los terapeutas a la hora de hacer una configuración.

4.3.5. Interfaz gráfica para administrador de un centro

La actividad del administrador de un centro tiene lugar, igual que la del superadministrador, en una página principal a modo de escritorio. En la Figura 62 se presenta la interfaz gráfica del administrador de un centro. Tiene una cabecera en la parte superior con los mismos tres botones que la página de superadministrador: uno con el nombre del administrador –que conduce a la página de perfil-, otro a la izquierda con el icono de Blexer-Med –para volver a la página principal- y por último uno para cerrar sesión. Además, la cabecera incluye el nombre del centro al que está asociado este administrador. En el "escritorio" del administrador, como se puede ver en la figura 62, hay un menú a la izquierda, donde puede elegir qué vista quiere mostar: "Gestión del centro", "Gestión de terapeutas del centro" o "Gestión de pacientes del centro".

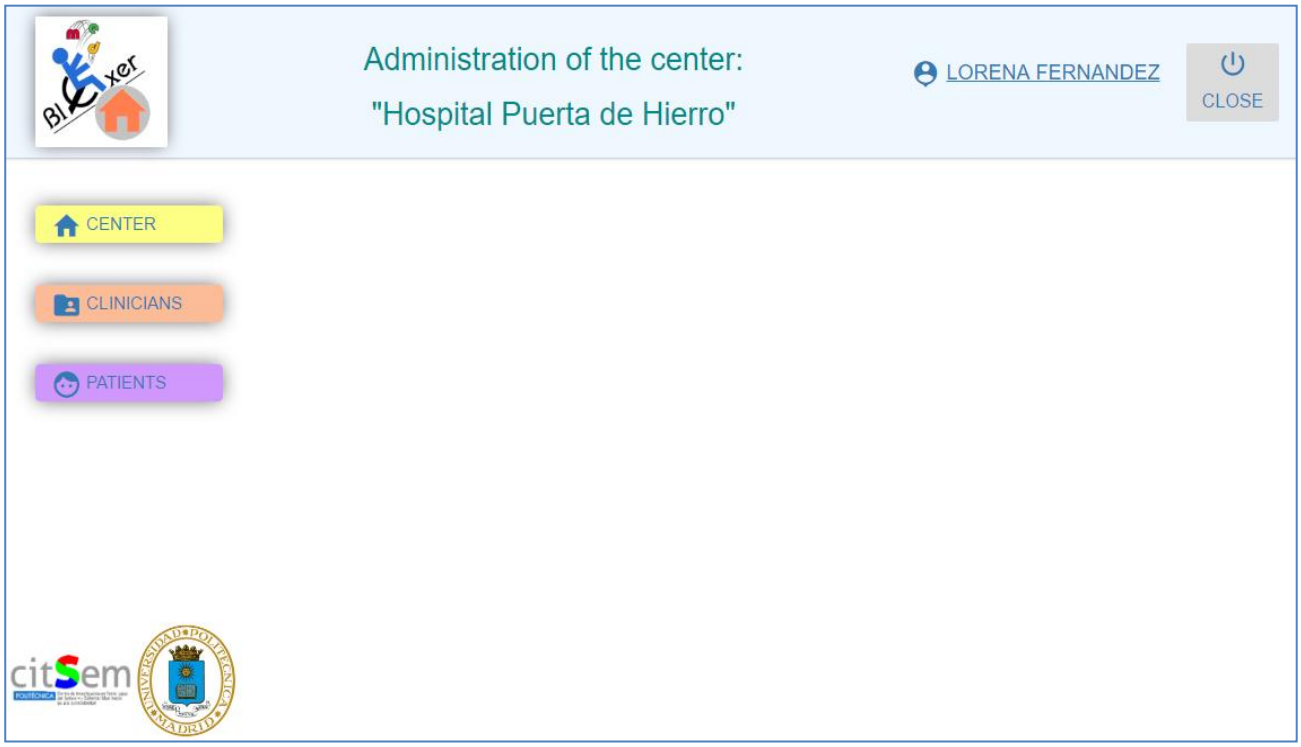


Figura 62 - Administrador: Página principal

En la Figura 63 aparece cómo se visualiza la información de un centro. En la parte inferior derecha un botón permite acceder a un formulario para editar la información del centro (ver *Anexo B.II. Gestión del centro médico*).

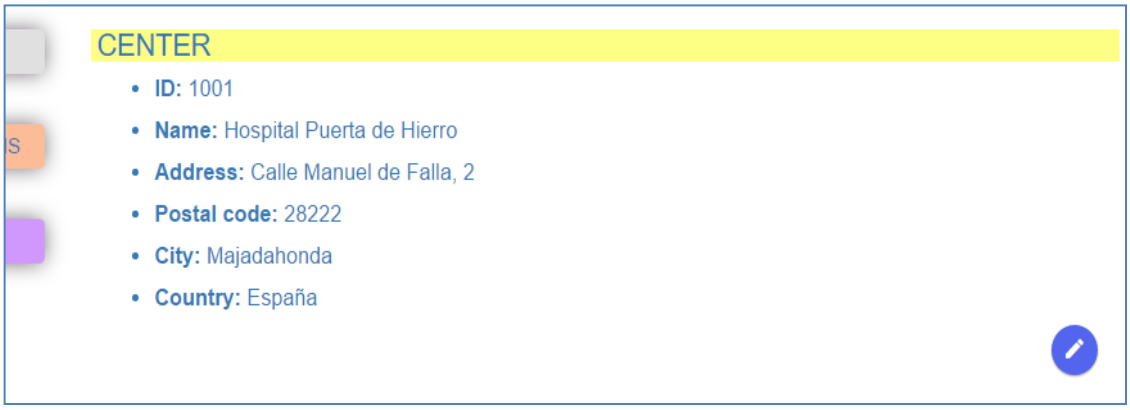


Figura 63 - Administrador: Vista de información del centro

Sobre la gestión de terapeutas, el administrador de un centro puede consultar todos los terapeutas del centro y la información de cada uno en una tabla como la de la Figura 64. En esta tabla se puede ver también que administrador dio de alta a cada terapeuta.

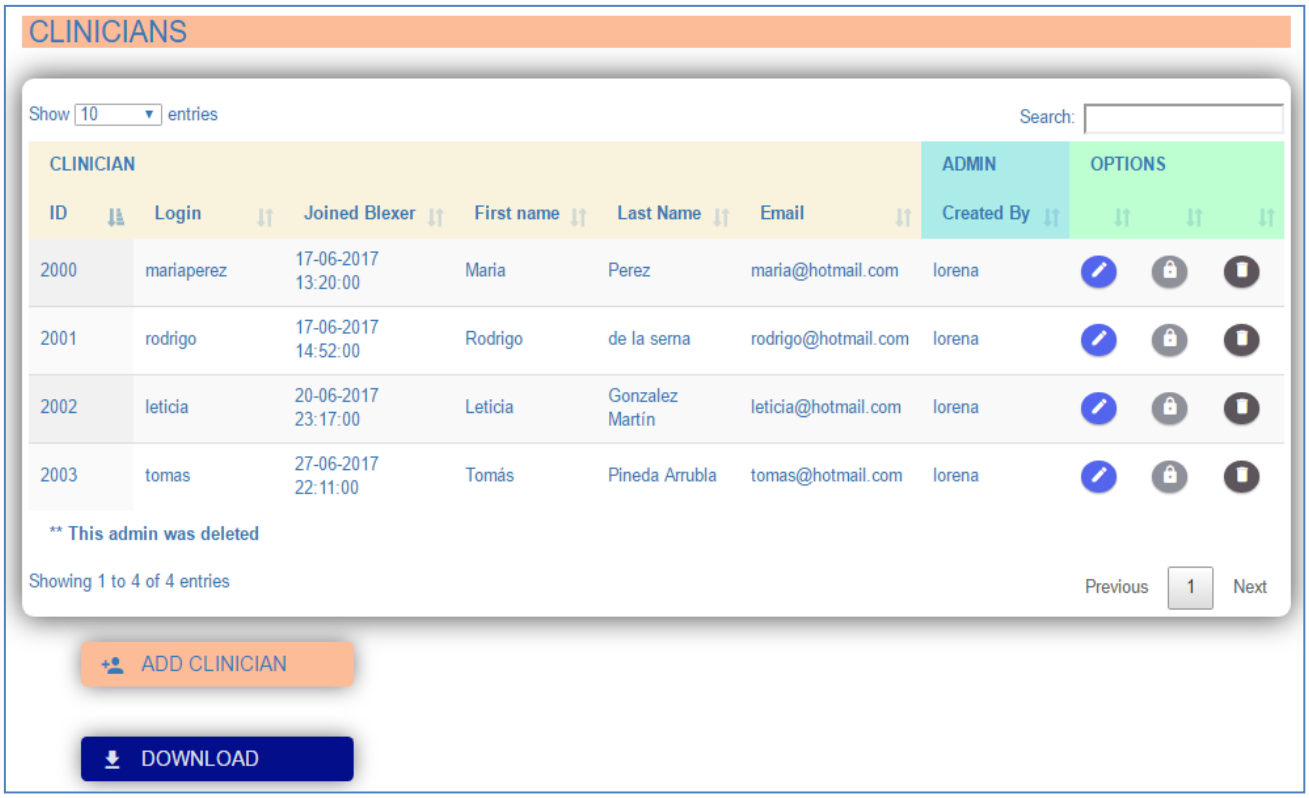


Figura 64 - Administrador: Vista de todos los terapeutas del centro

Desde esta perspectiva de "Gestión del terapeuta" se puede dar de alta un terapeuta en el centro médico, editarlo, restablecer su contraseña, y darlo de baja. Asimismo, con permisos de administrador es posible descargar una lista de todos los terapeutas de un centro (Manual de usuario disponible en *Anexo B.III. Gestión de terapeutas).*

Como administrador del centro, también se pueden consultar los pacientes del centro, mediante una tabla que aparece en la Figura 65. Igualmente, se ofrece añadir nuevos pacientes, modificar sus datos personales, o darlos de baja, así como descargar una lista de pacientes del centro (*Anexo B.IV. Gestión de pacientes*).

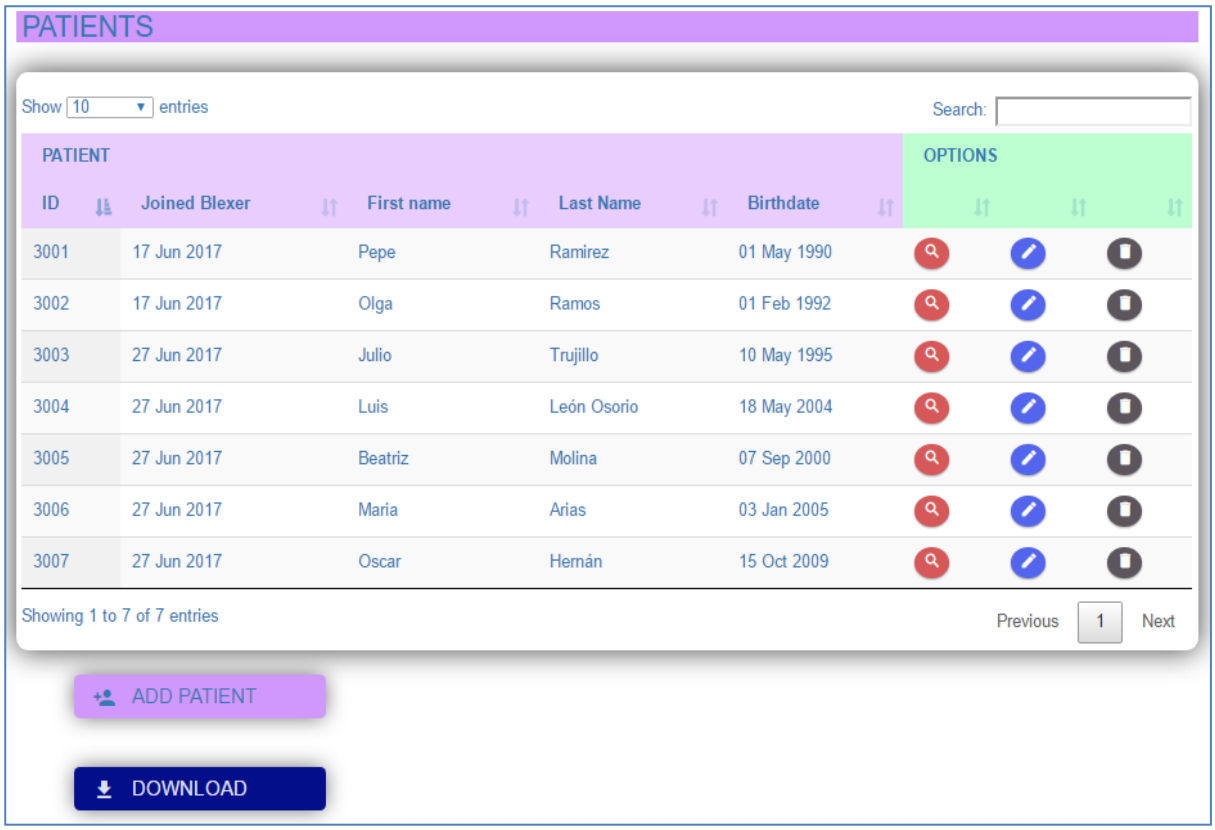


Figura 65 - Administrador: Visualizar pacientes del centro

Cuando un administrador solicita dar de baja a un paciente, si éste tiene resultados guardados de partidas jugadas, se ofrece la posibilidad de descargar estos resultados.

4.3.6. Interfaz gráfica para terapeutas

La interfaz gráfica para terapeutas es parecida a las de administrador y superadministrador, aunque es la que más difiere de las otras dos. Se muestra en la Figura 66 el aspecto de la página del terapeuta.

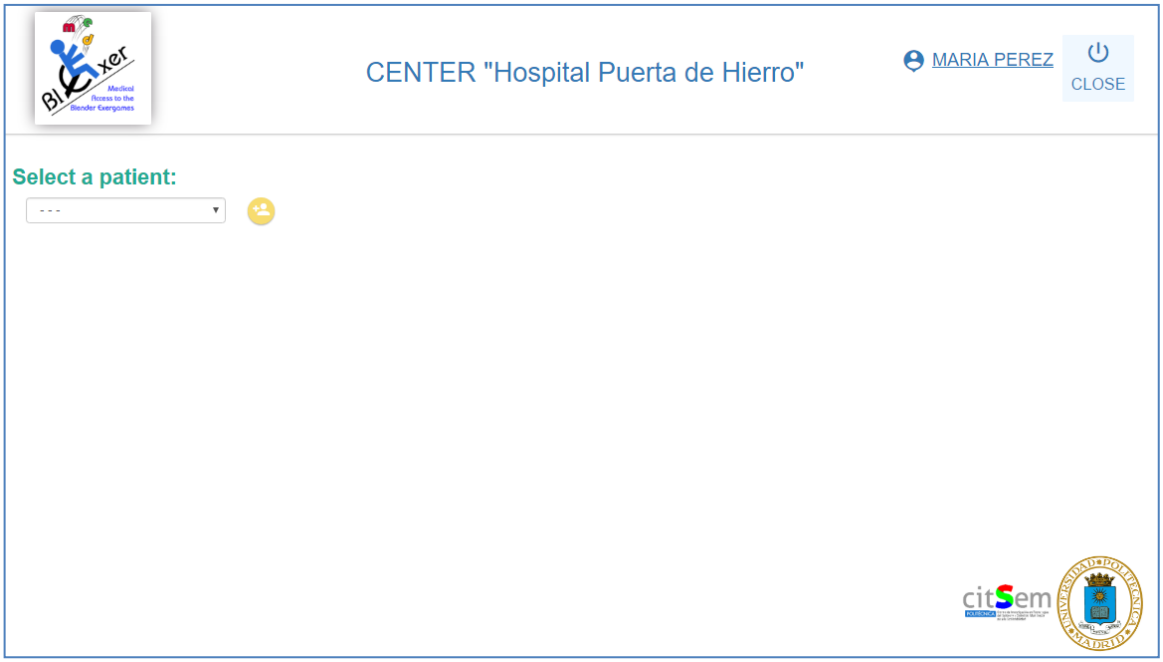


Figura 66 - Terapeuta: Página principal

Un terapeuta puede gestionar un paciente, sus configuraciones y resultados. En primer lugar, debe seleccionar el paciente que desea gestionar en el desplegable (Figura 67) que muestra todos los pacientes del centro. A la derecha del desplegable se encuentra la opción para añadir un nuevo paciente al centro.

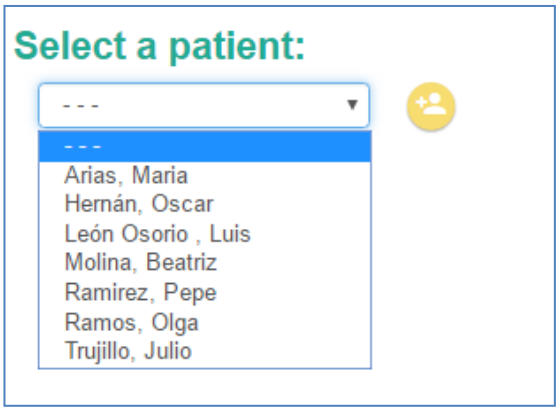


Figura 67 - Terapeuta: Desplegable de pacientes

Una vez que el terapeuta selecciona un paciente, el escritorio toma la forma de la Figura 68. En el centro, se observa la información del paciente seleccionado (ampliado en la Figura 69). Debajo, los comentarios –si los hay- de los terapeutas (ampliado en la Figura 70). En esta página, el terapeuta podrá editar la información de un terapeuta o darlo de baja. Además podrá poner un comentario en el perfil de este paciente, y visualizar el historial de comentarios, tanto suyos como de otros terapeutas.

A la derecha y colocados en columna, tres botones forman el menú del terapeuta. El primero sirve para mostrar la información de un paciente y los comentarios mencionados anteriormente. El segundo, para ver el historial de configuraciones de cada ejercicio para el paciente seleccionado. El tercero, para ver los resultados de las partidas jugadas.

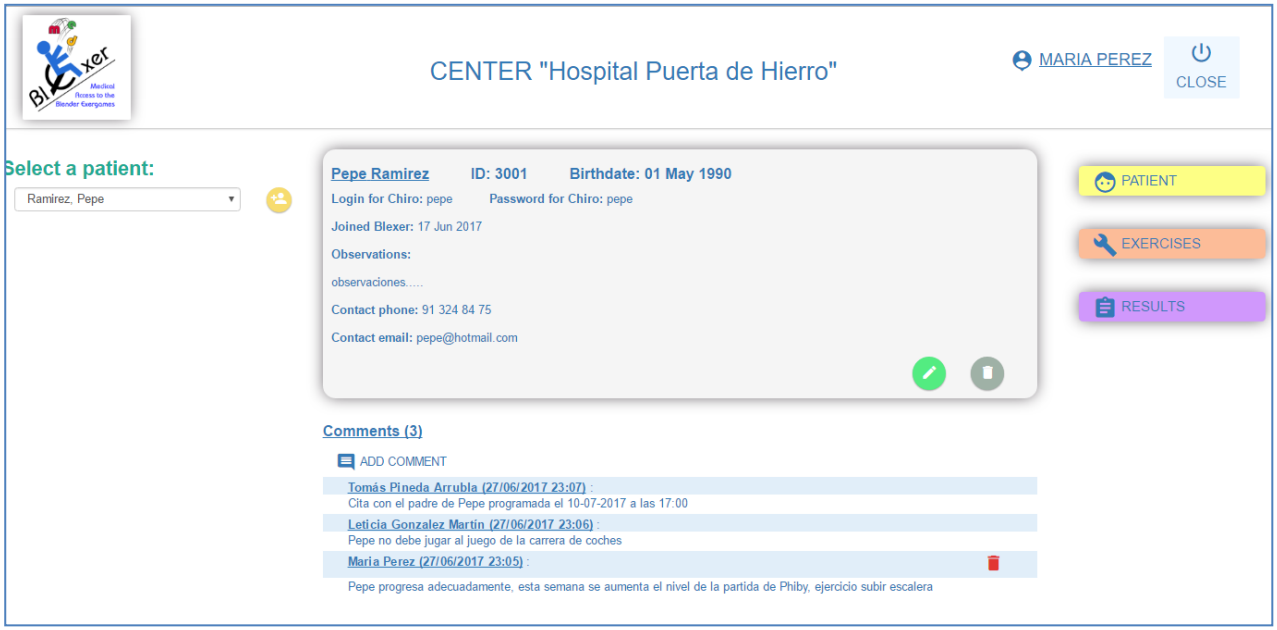


Figura 68 - Terapeuta: Vista de paciente seleccionado

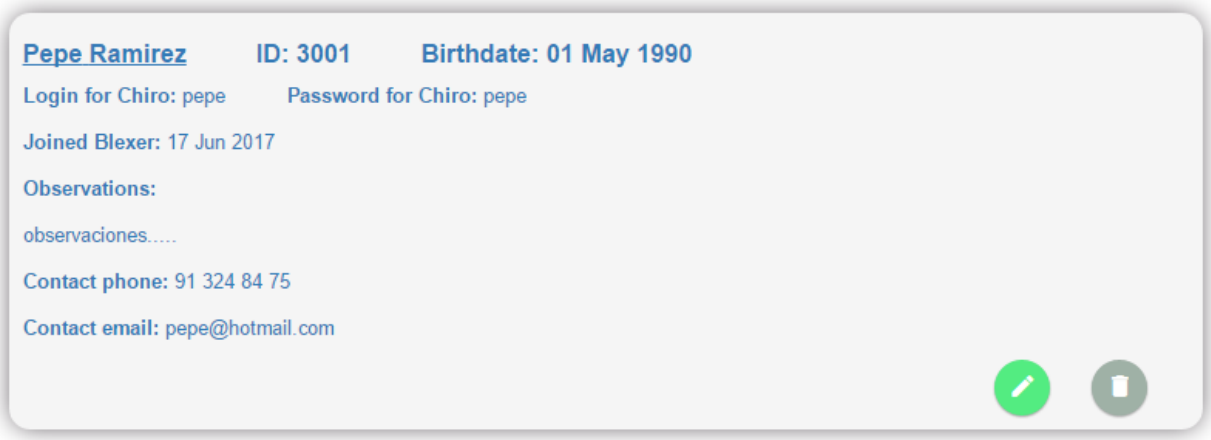


Figura 69 - Terapeuta: Detalle de información del paciente

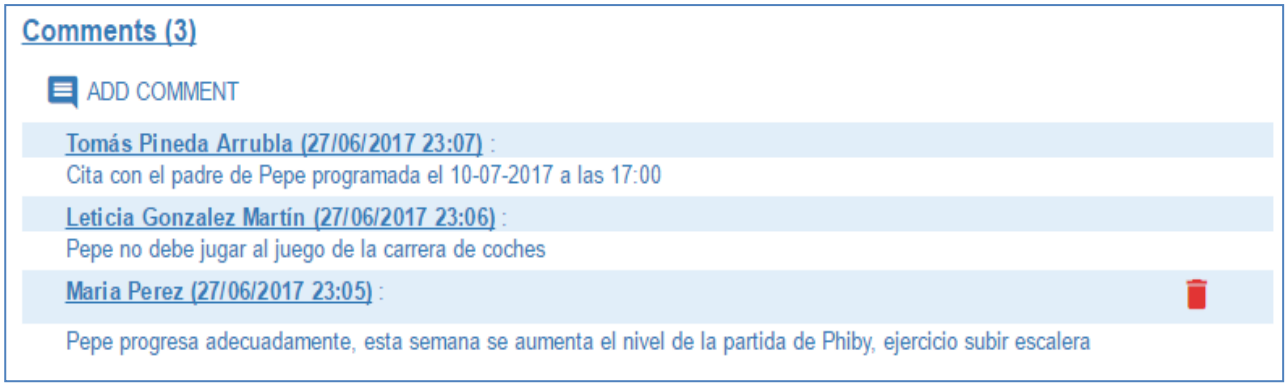


Figura 70 - Terapeuta: Comentarios de terapeutas en un paciente

Si el terapeuta selecciona en el menú la opción "Configuraciones", se oculta automáticamente la información del usuario y los comentarios de terapeutas, y en su lugar se coloca un desplegable para seleccionar el juego que se quiere configurar. Cuando se elige un

juego, debajo del desplegable se muestra una tabla con los ejercicios de este juego (Figura 71) y sus respectivos datos (identificador, código, título y descripción).

A la derecha de cada fila de la tabla de la Figura 71, hay un botón para consultar el historial de configuraciones de ese ejercicio o definir una nueva. Cuando se pulsa dicho botón, aparece un modal –ventana emergente- como el representado en la Figura 72. En el título del modal están escritos el nombre del paciente y el ejercicio seleccionados. Debajo hay un formulario para hacer una nueva configuración para este paciente y este ejercicio. Como se puede ver en la figura 72, los campos del formulario coinciden con los parámetros de configuración que estableció el superadministrador al añadir el ejercicio al sistema. El terapeuta puede consultar qué significa cada parámetro en el icono con una interrogación, a la derecha del campo que quiera consultar.

Si aún no se ha hecho ninguna configuración, el videojuego contará con una por defecto. Si se hace una configuración por primera vez, ésta se pondrá "en uso" y será la utilizada para las próximas partidas del videojuego. Si hay más de una configuración, el terapeuta podrá elegir cuál estará "en uso". En el caso de que se elimine la configuración que está en uso –solo el terapeuta que la hizo puede borrarla- se pondrá "en uso" la configuración que se hizo más recientemente.

Para las configuraciones que el terapeuta autenticado hizo, éste puede escribir, modificar o eliminar un comentario. (Ver *Anexo C.III. Configuración de ejercicios*).

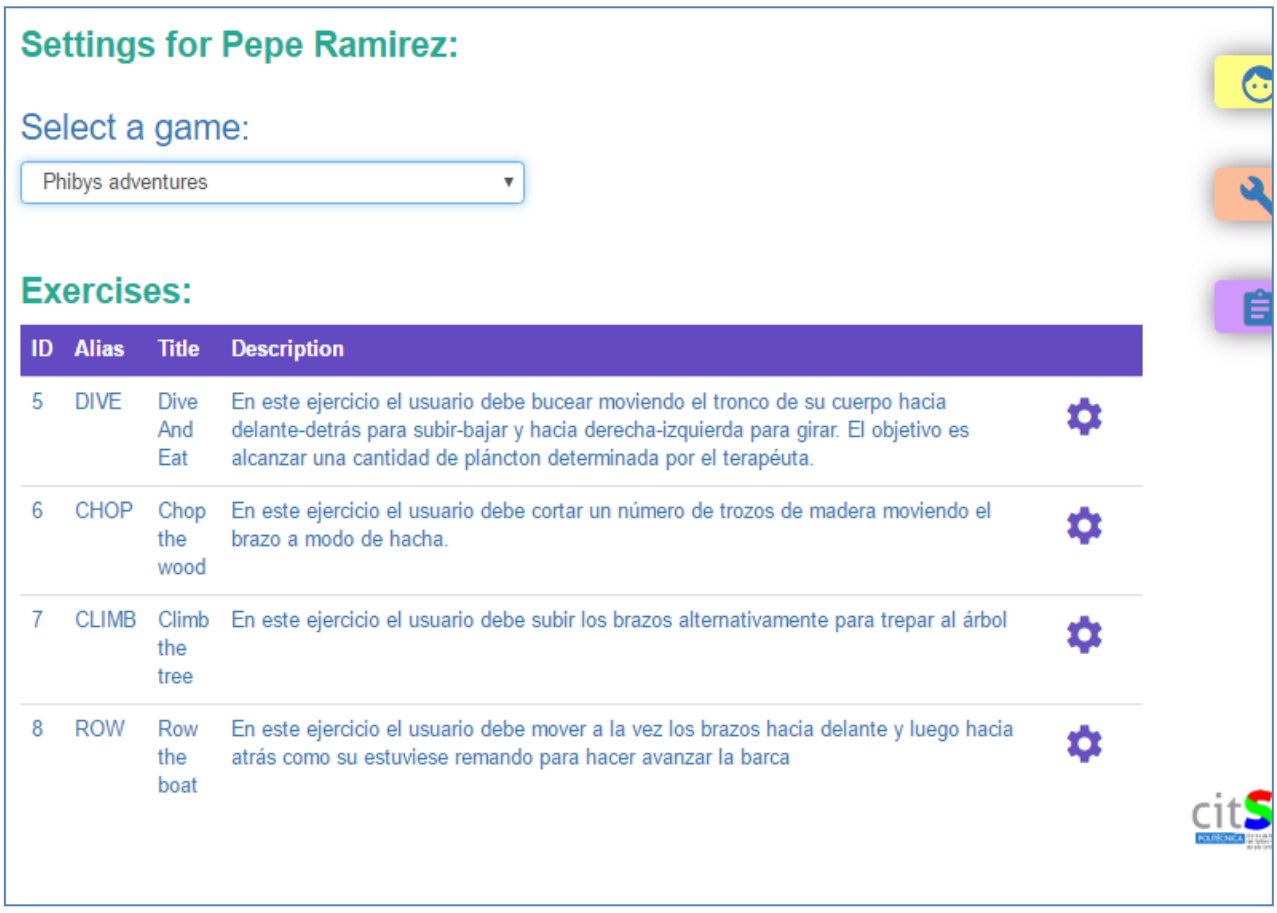


Figura 71 - Terapeuta: Vista de ejercicios de un juego

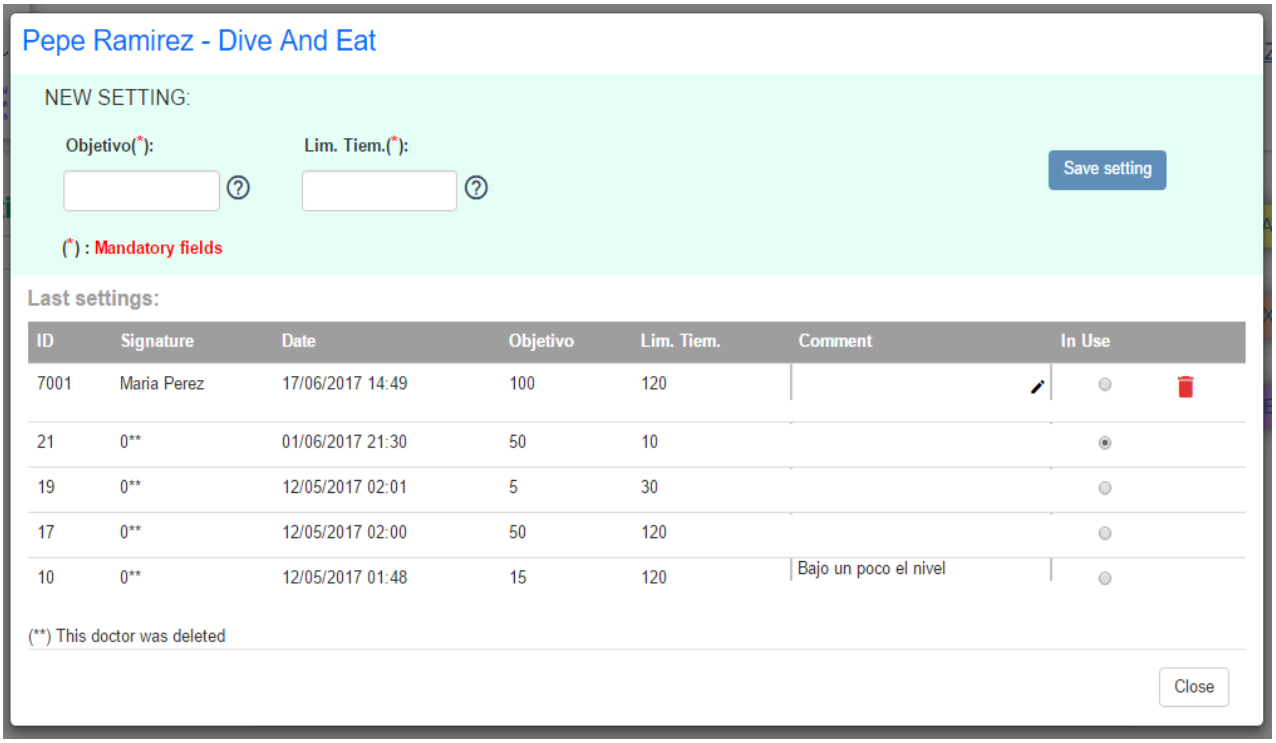


Figura 72 - Terapeuta: Vista de configuraciones de un ejercicio para un usuario

Una de las funcionalidades más importantes de la plataforma, junto con realizar la configuración de los videojuegos es que el terapeuta pueda ver los resultados de las partidas jugadas por los pacientes. En la Figura 73 se muestra la tabla que se visualiza al pulsar el tercer botón ("Resultados") del menú del terapeuta. Previamente a que se visualice la tabla, el terapeuta debe elegir el juego y el ejercicio del que quiere ver los resultados. En la tabla que se puede ver en la Figura 73, aparece el identificador de cada partida y la fecha y hora en la que se jugó. También se observan los datos de la configuración con la que se jugó –en la cabecera de la tabla se pueden ver el nombre de los parámetros que estableció el superadministrador-. Por último, dos columnas muestran la duración de la partida y el número de movimientos correctos. Estos dos parámetros de resultados, podrán variar en un futuro. Son los elegidos para esta solución propuesta.

Para terminar con la interfaz gráfica del terapeuta, cabe decir que tiene la posibilidad de descargar los resultados de un ejercicio (*Anexo C.IV. Consultar resultados de un ejercicio*), pulsando el botón "descargar resultados" bajo la tabla de resultados. Con este servicio, se ofrece al terapeuta la oportunidad de recolectar los datos de las partidas y almacenarlos o trabajar con ellos con otra aplicación de cálculo o realización de gráficas.

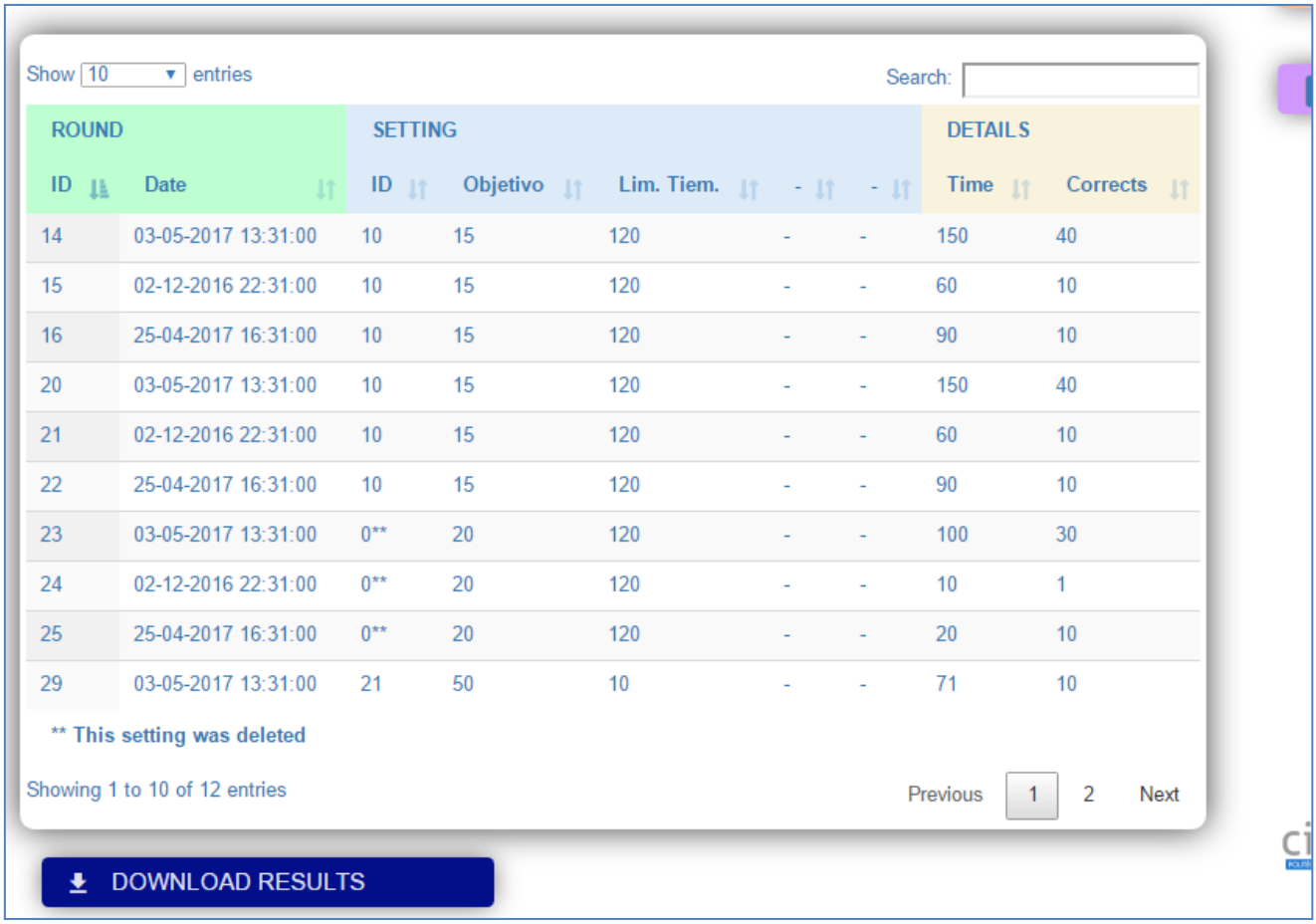


Figura 73 - Terapeuta: Vista de resultados de ejercicio

4.3.7. Mecanismos de seguridad

En primer lugar, la plataforma Blexer-Med debe ofrecer confidencialidad sobre la información personal que maneja, tanto de especialistas como de pacientes. Por eso se han establecido los roles de usuarios de manera que unos tengan permisos para ciertas acciones y otros no. Para securizar los datos de inicio de sesión, concretamente la contraseña, ésta se guarda en la BD en forma de resumen MD5 (Visto en el apartado 4.3.2. "Arquitectura de capas: Capa de Negocio" en la descripción de la clase Java *SecurityManager*). El resumen MD5 de una palabra es un cifrado único y unidireccional, es decir, no puede hacerse a la inversa. Se puede calcular el resumen MD5 de una palaba tantas veces como haga falta, y siempre será el mismo. Sin embargo, si se averigua un resumen MD5, desconociendo de qué palabra fue sacado, no se puede saber la palabra original.

Otra medida de seguridad implementada es que no se pueda acceder mediante url a una página sin estar autenticado. Esto se ha logrado almacenando el nombre de usuario autenticado en una variable que se elimina cuando se cierra la sesión. Cuando se accede por url a una página que no es la página de *login*, se comprueba esta variable de sesión para ver si hay un usuario autenticado. Si se intenta acceder directamente a la página principal o de perfil de un usuario sin estar autenticado, la aplicación "expulsa" al usuario a la página de autenticación.

Por otra parte, ya se ha visto que las funciones de JavaScript –que se utilizan en la Capa de Presentación para enviar peticiones al Web Service– están separadas en documentos ".js" según el usuario al que presten servicio (Figura 29 - Estructura de scripts utilizados en Blexer-Med). Dado que en el documento html que construye la interfaz de usuario en el navegador se importan solo los ".js" que se utilizan, el resto quedan inaccesibles. De esta forma, para los tres tipos de usuarios, cuando uno de ellos está autenticado (por ej. un terapeuta), la lógica de JavaScript de los otros dos (ej. administrador y superadministrador) queda oculta.

En segundo lugar, es fundamental procurar que la información que se envía a la base de datos sea consistente. Por ejemplo, a la hora de dar de alta un paciente, en el campo "fecha de nacimiento", es importante que se introduzca una fecha (con el formato requerido) y no otro tipo de dato. En caso contrario, la base de datos produce una excepción, que también es muy importante capturar para poder mostrar un mensaje de error al usuario.

Para la validación de los campos de un formulario rellenados por el usuario, se ha utilizado un *plugin* de jQuery, "jQuery Validate". A este plugin, se le indican los valores que espera recibir un formulario y si son requeridos o no, y los mensajes de error que deben mostrarse en caso de que no se cumpla algún requerimiento. Por ejemplo, la validación del formulario "Editar perfil" en la página de administrador se implementa de la siguiente forma:

```
$("#add_game_form").validate({
      rules: {
            title_game_add: {
                  required: true,
                  maxlength: '45'
            },
            code_game_add: {
                  required: true,
                  maxlength: '5'
            },
            description_game_add: {
                  required: true,
                  maxlength: '600'
            }
      },
      messages: {
            title_game_add: {
                  required: "Title required",
                  maxlength:  "Title  must  have  less  than  46
            characters"
            },
            code_game_add: {
```

```
                    required: "Alias required",
                    maxlength: "Alias must have less than 6 characters"
                },
                description_game_add:{
                    required: "Description required",
                    maxlength: "Descript. must have less than 300
                characters"
                }
            }
        });
```

En el código anterior, se le pasa a la función *Validate* las reglas a seguir para cada campo del formulario (al ser campos de tipo texto, se les pasa si son requeridos o no y el número máximo de caracteres) y los mensajes que se desea mostrar en caso de que no se cumplan esas reglas. El resultado es el de la Figura 74, donde aparecen diferentes mensajes de error si se pulsa sobre "Save game".

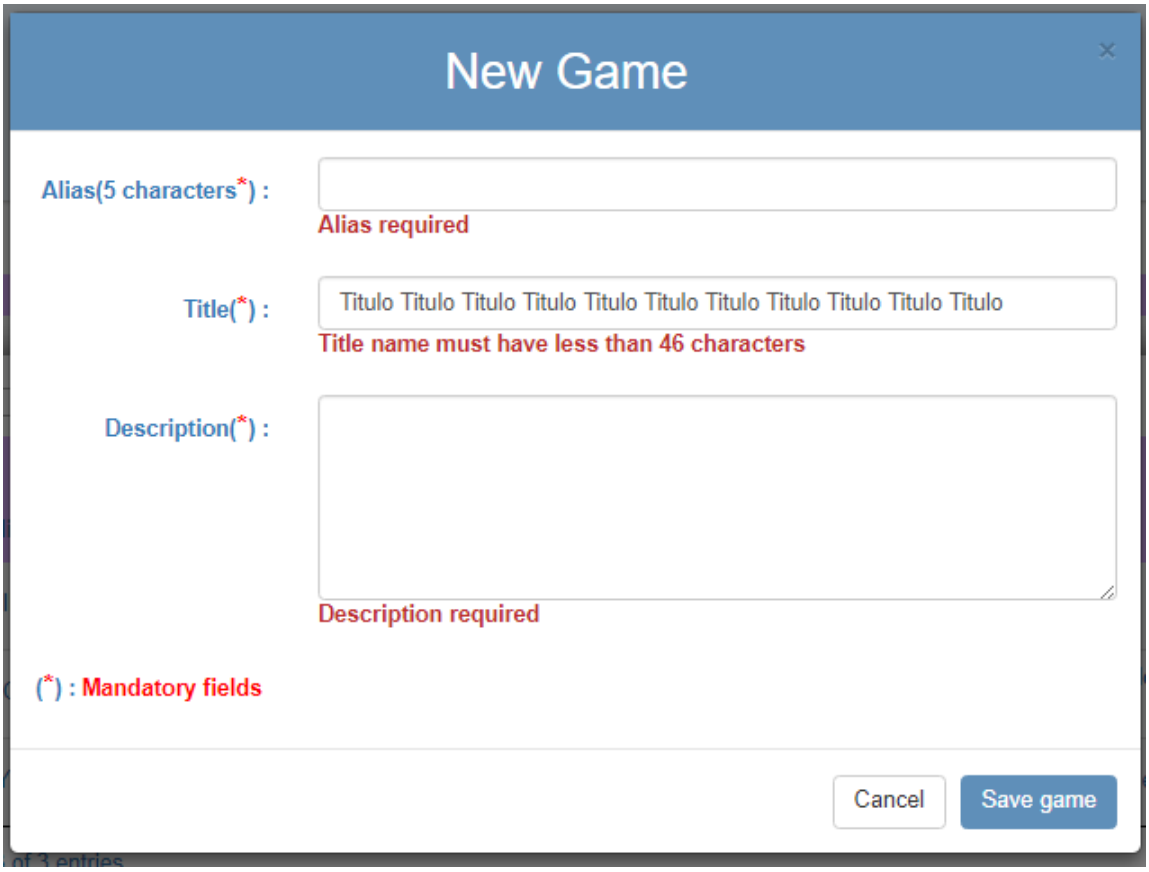


Figura 74 - Uso de jQuery Validate

Se han llevado a cabo otro tipo de comprobaciones, desde los scripts y no con jQuery Validate, como por ejemplo:

- ✓ Que haya variado algún campo en los formularios de edición de objetos. Ej. No se ha modificado ningún campo en el formulario "Editar centro" (Figura 75).

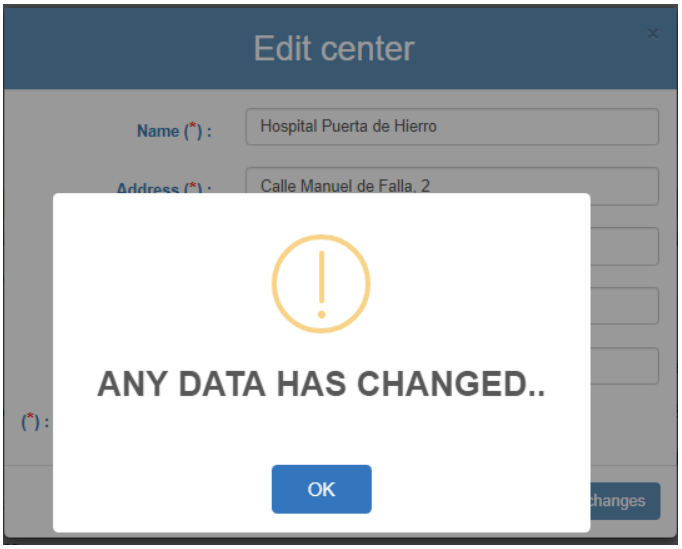


Figura 75 - Error en edición de un objeto

- ✓ Al crear o editar un objeto, si un *login*/*email*/código debe ser único, que no esté repetido para otro objeto. Ej. Al crear un administrador, el *login* está repetido (Figura 76).

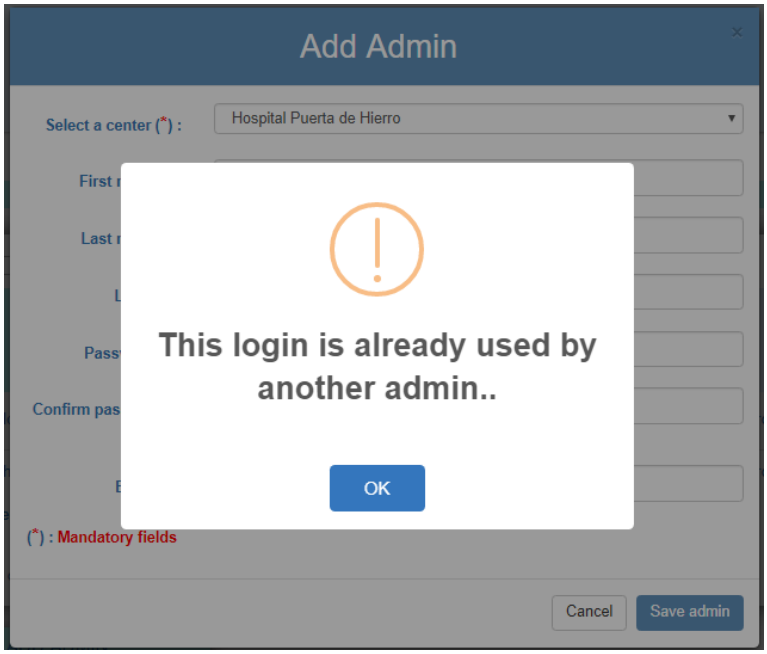


Figura 76 - Mensaje de error por login repetido

- ✓ Para restablecer la contraseña de un usuario, se realizan las siguientes comprobaciones:
  - La contraseña antigua proporcionada es correcta
  - La nueva contraseña se ha confirmado debidamente
  - La nueva contraseña no es la misma que la antigua

4.4. Middleware Chiro

"Conceptualmente, Chiro es un programa muy simple: no es ni más ni menos que una interfaz gráfica consistente en una ventana a la que va añadiendo distintas pestañas. Cada una de estas pestañas contiene el código y los controles de la comunicación entre Blender y un dispositivo específico de NUI (*Natural User Interface*). […] Actualmente Chiro no es capaz de interactuar con las pestañas de otra forma que no sea inicializarlas o destruirlas, por lo que éstas deberán funcionar de forma autónoma y contener en su interfaz todos los controles necesarios para que el usuario influya en la comunicación". [8]

El entorno del *middleware* está formado por varios proyectos de C#, uno para el contenedor (este proyecto se llama "Chiro"), y otros para la creación de pestañas para cada dispositivo. Para cada dispositivo que se quiere comunicar con Chiro, se debe crear un proyecto (el proyecto para la Kinect se llama "KinectForms"). En *KinectForms* se implementa el comportamiento de la pestaña para Kinect que se observa en la Figura 77.

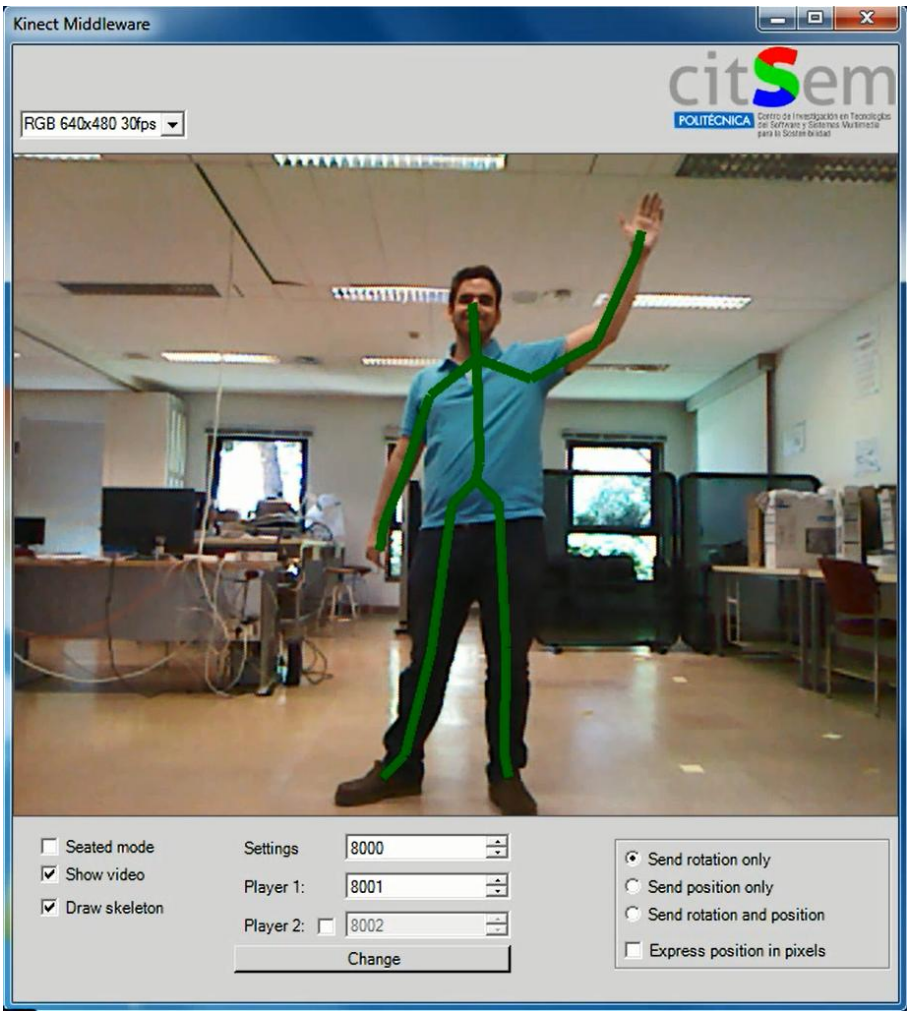


Figura 77- Interfaz gráfica de la KinectWindow [8]

"Para que el contenedor incorpore a su interfaz la pestaña […], ésta debe estar listada en el archivo que Chiro lee al iniciarse" [8] (*TabList.bin*). Se requiere el uso de otros

proyectos, “WindowReference” y “WindowRefMaker”, para la instalación de la pestaña de la Figura 77 en el contenedor Chiro.

WindowReference contiene las clases necesarias para crear pestañas: *WindowRef* y *WindowList*. La clase *WindowRef* posee los atributos que constituyen una pestaña (título de la pestaña, ubicación de la biblioteca donde está definido el middleware, objeto que contiene el middleware –por ejemplo *KinectForm*-, la existencia o no de un ejecutable para instalar bibliotecas requeridas, y ubicación de dicho ejecutable –si existe-). La clase *WindowList* sirve para instanciar un mapa de objetos clase *WindowRef*.

WindowRefMaker es el proyecto cuya aplicación principal crea tantos objetos de tipo *WindowRef* como pestañas se quieren añadir a Chiro, añade todas esas pestañas a un mapa *WindowList*, lo serializa, y lo escribe en el fichero *TabList.bin*. Así el contenedor Chiro, en su aplicación principal, puede acceder a los datos que contiene cada *WindowRef* desde el mismo fichero y mostrar las pestañas correspondientes a cada dispositivo.

4.4.1. Adaptación de Chiro al entorno de Blexer-Med

Con la creación de la plataforma Web, se ha asignado una nueva misión a Chiro que es, en primer lugar, descargar de la plataforma las configuraciones hechas por los terapeutas para un usuario y, en segundo lugar, enviar los resultados de las partidas jugadas por un usuario.

Para ello, se ha añadido a la aplicación principal de Chiro un breve formulario (Figura 78) para la autenticación del paciente. Este nombre de usuario y contraseña son determinados al dar de alta el paciente en un centro (por parte de un terapeuta o administrador) y son modificables por un terapeuta o administrador en cualquier momento desde la plataforma.

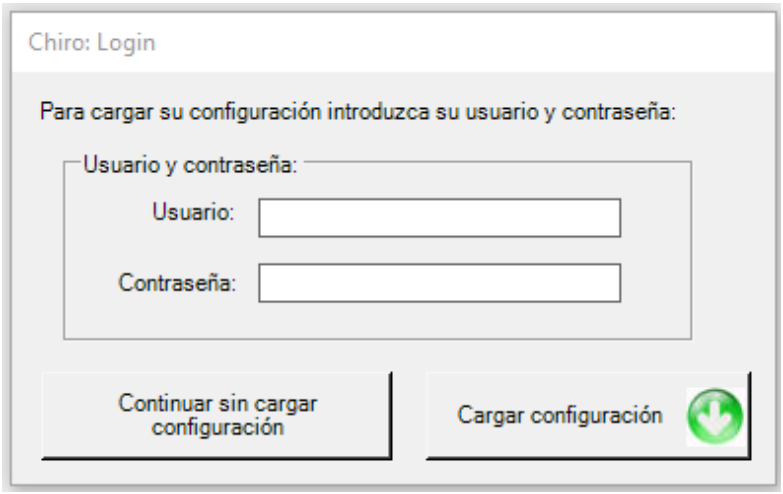


Figura 78 - Formulario de autenticación del paciente en Chiro

Como se puede ver en la figura 78, no es obligatorio autenticarse pues no es deseable que existan restricciones para jugar, aunque sí para descargar las configuraciones. Si se pulsa sobre “continuar sin cargar configuración” el juego deberá usar una configuración por defecto. Si por el contrario se selecciona “Cargar configuración”, se realiza el proceso de autenticación que aparece en el diagrama UML de secuencia de la figura 79, similar al de autenticación de un terapeuta, administrador o superadministrador en la plataforma Web.

Se puede observar que el desencadenante es el evento *onClick* al pulsar el botón "Cargar Configuración" que a continuación realiza una petición HTTP al Web Service y éste solicita a la clase intermediaria UserManager que realice la autenticación en la BD.

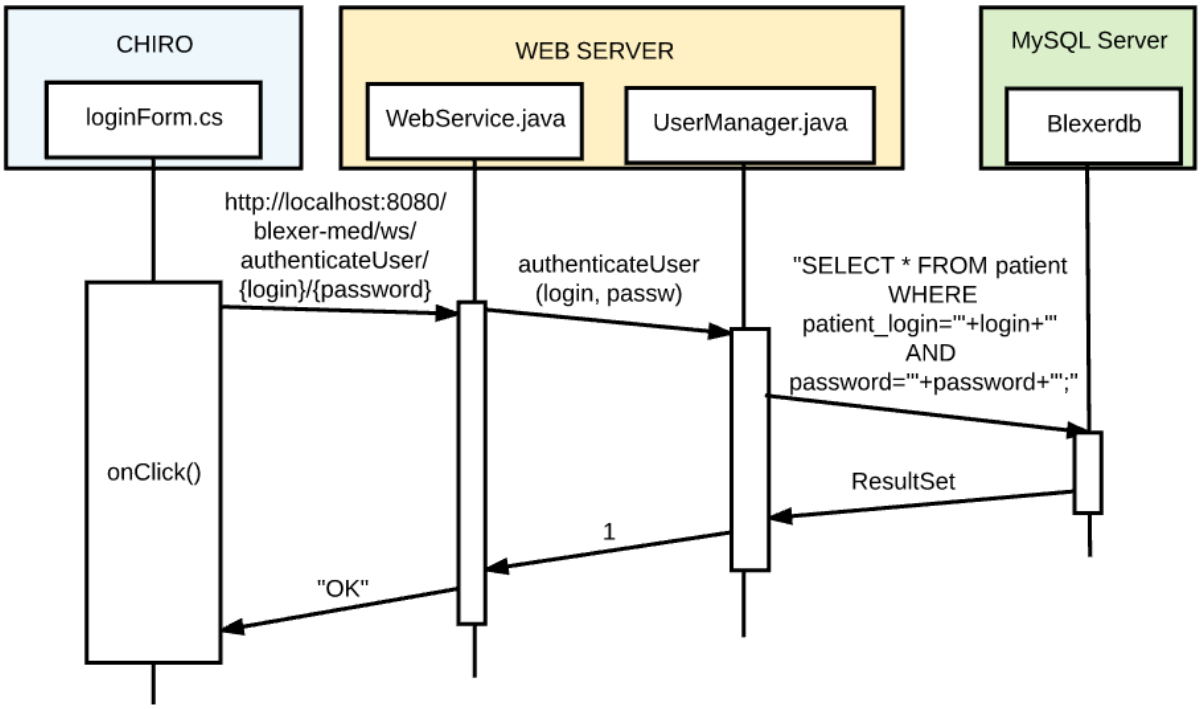


Figura 79 - Diagrama de secuencia de autenticación de paciente en Chiro

En caso de que la contraseña o usuario sean incorrectos, UserManager devolverá "-1" y si hubo un error de conexión, se producirá una excepción. Cuando el resultado es "-1", WebService responde "NOK" y si se produce la excepción en la conexión responde "ERR", y el evento *onClick*, según la respuesta recibida a la petición HTTP muestra un mensaje de la Figura 80. Al pulsar sobre "Aceptar", sigue activo el formulario de autenticación. El usuario puede reintentar o "Continuar sin cargar configuración".

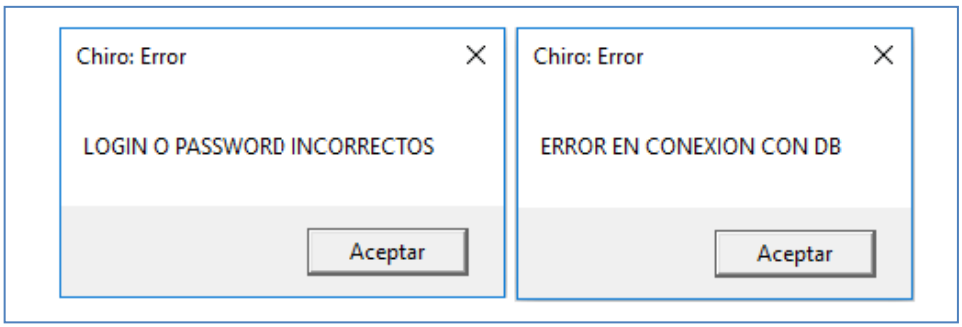


Figura 80 - Mensajes de error en autenticación del paciente en Chiro

Cuando UserManager devuelve "1", como es el caso de la Figura 79, WebService responderá "OK" a la petición HTTP. Entonces Chiro, después de haber autenticado al usuario, inicia las siguientes acciones en orden:

1) Descargar la configuración de los ejercicios para el usuario autenticado (explicación en detalle en apartado 4.4.2. “Descarga de configuraciones para un usuario de Blexer-Med”)
2) Comprobar si hay resultados sin enviar de partidas anteriores asociadas al usuario autenticado. (ver 4.4.3 “Envío de resultados de ejercicios a Blexer-Med”)

Por último, cuando se cierra la ventana de Chiro para terminar de jugar, la aplicación vuelve a comprobar si hay resultados por enviar para este usuario, en cuyo caso los envía a la plataforma.

4.4.2. Descarga de configuraciones para un usuario de Blexer-Med.

Para descargar la configuración, Chiro –en C#- envía una segunda petición HTTP GET al servicio web –en Java-, al recurso *LoadSettingsByUser* al que le pasa como parámetro el login introducido en el formulario. Cuando el método *LoadSettingsByUser* de *WebService* recibe la petición, solicita a *SettingManager* la configuración que está “en uso” para cada juego y para cada ejercicio.

El método *LoadSettingsByUser* recopila los datos de cada configuración en uso de cada ejercicio y crea un objeto de clase *SettingInfo* (Figura 81) para cada una, que contiene el identificador de la configuración, y los valores propios de esa configuración.

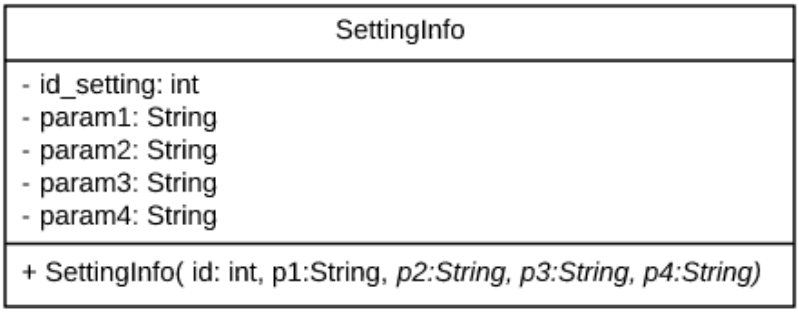


Figura 81 - Representación de la clase SettingInfo.java

A continuación *LoadSettingsByUser*, crea un **mapa de ejercicios para cada juego**. Habrá tantos mapas de ejercicios como juegos haya en el sistema. Un mapa está formado por un conjunto de ítems, cada uno de ellos con la forma “clave-valor”. En este caso, la clave que identifica cada ejercicio es un array de tres cadenas de caracteres: *“[id_ejercicio, código_ejercicio, título de ejercicio]*” y el valor es el objeto de clase *SettingInfo* creado anteriormente para cada ejercicio.

Después *LoadSettingsByUser* crea **un mapa de juegos**, y de nuevo cada entrada del mapa tendrá la forma “clave-valor”. La clave para cada juego es el array “*[id_juego, código_juego, título_juego]”* y el valor es el mapa de ejercicios de dicho juego. Con este mapa de juegos, *LoadSettingsByUser* crea un objeto de la clase auxiliar *ChiroSettingResponse*, que contiene también el identificador del usuario, su login, nombre y apellidos.

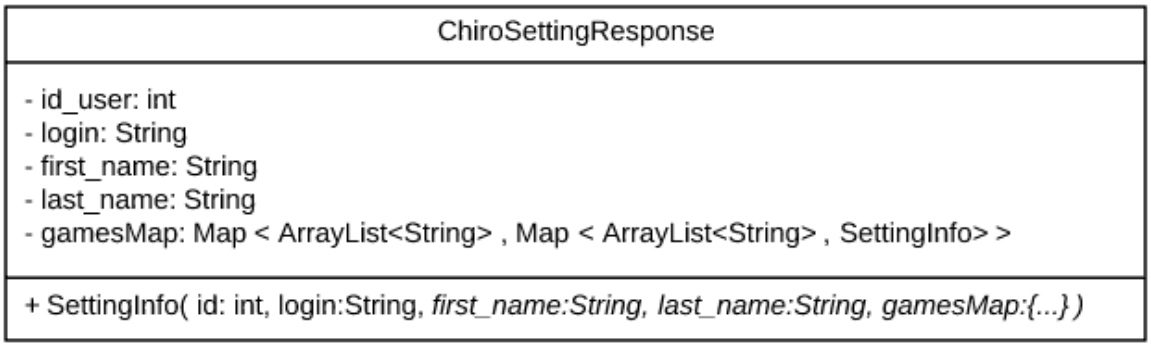


Figura 82 - Representación de la clase ChiroSettingResponse.java

La nomenclatura para declarar un mapa es "*Map<tipo de clave, tipo de valor>"*. De ahí que el mapa de ejercicios sea "Map<ArrayList<String>,SettingInfo>" y que el mapa de juegos sea "Map<ArrayList<String>, Map<ArrayList<String>,SettingInfo>>".

Este objeto de tipo ChiroSettingResponse es el que se envía desde el servicio Web a Chiro –como respuesta a la petición HTTP GET– serializado en formato JSON (Ver *Anexo D: Formato JSON de configuraciones descargadas*). Chiro, tal cual recibe el JSON, lo escribe en un fichero de texto en "*Chiro/users-settings/login_usuario_autenticado.txt*", listo para usar por otra aplicación o por el propio videojuego.

4.4.3. Envío de resultados de ejercicios a Blexer-Med.

Cuando un usuario inicia sesión en Chiro o cierra la aplicación, el programa principal consulta si existe un fichero de texto "*Chiro/users-results/login_usuario_autenticado.txt*" que contenga resultados. En caso de que exista el fichero, los resultados deben estar escritos en un formato determinado (*Anexo E: Formato JSON de resultados guardados*). La aplicación lee el JSON contenido en este fichero y trata de deserializarlo en un objeto de la clase *UserResults* en C# (Figura 83). Esta clase tiene dos atributos, un identificador del usuario que corresponde con el del autenticado, y un array de resultados, que deserializa en objetos de clase *ExerciseResults* (Figura 84).

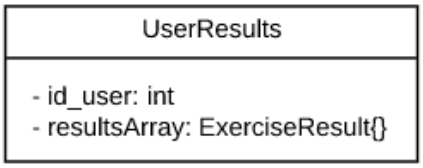


Figura 83 - Clase UserResults

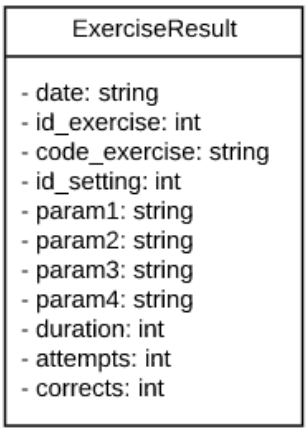


Figura 84 - Clase ExerciseResult

De cada resultado, se espera obtener la información de la partida (fecha, duración, número de movimientos, y movimientos correctos), datos sobre el ejercicio asociado a esta la partida (*id* y código de ejercicio) e información redundante sobre la configuración con la que se jugó esta partida (identificador y los valores de los cuatro parámetros).

Si se da el caso de que ocurra un problema al deserializar el JSON (SerializationException), es porque los resultados no están escritos en el formato correcto en el fichero, y aparece el mensaje emergente de la Figura 85.

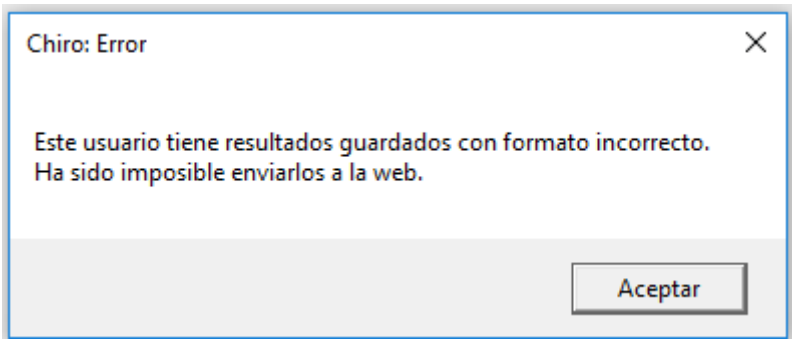


Figura 85 - Mensaje de error en formato de fichero de resultados

Si se ha deserializado el JSON correctamente, Chiro recorre el array de resultados *resultsArray* de la Figura 83 y uno a uno, los envía a la plataforma a través de una petición HTTP POST al recurso *saveResult.* Chiro envía un resultado, espera una respuesta y si la respuesta es "OK", borra el resultado de *resultsArray* –solo borra ese resultado-, y procede a enviar el siguiente.

El método *saveResult* del servicio web comprueba que la información sea consistente –Ej. Que el *id* de la configuración que se pasa como parámetro pertenezca a ese usuario, o que los parámetros redundantes coincidan con los que un día estableció el terapeuta-. Si hay un error en la consistencia de los datos, *saveResults* responderá a Chiro "SQL_ERROR" y éste mostrará el mensaje de la Figura 86. Si en cambio hubiese un error en la conexión con la BD, *saveResults* devolvería "CONEX_ERR" y se mostraría el mensaje de la Figura 80.

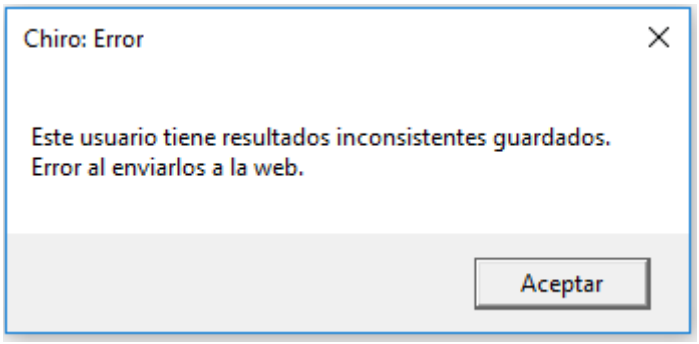


Figura 86 - Error en consistencia de los resultados

Si se produce cualquiera de los dos errores (inconsistencia en los datos o error en la conexión con BD), los resultados no se eliminan de *resultsArray*. Cuando Chiro termina de recorrer el array de resultados, sobrescribe el fichero de resultados del usuario, eliminando del fichero los resultados que se han enviado correctamente –para evitar resultados duplicados en la BD- y dejando los que no se han conseguido enviar. En futuras conexiones de Chiro se intentará enviarlos de nuevo.

- Es importante que desde la aplicación del videojuego, cuando se escriba en el fichero de resultados, no se sobrescriban los anteriores, si no que se añada cada nuevo resultado al final del fichero, ya que si hay resultados en el fichero es porque no se han enviado aún a la plataforma.

5. Conclusiones

El objetivo primordial de este proyecto era construir una plataforma Web a través de la cual terapeutas y especialistas en rehabilitación pudiesen gestionar los ejercicios de los juegos del entorno de Blexer (Blender Exergames), esto es, adaptarlos individualmente a cada jugador y monitorizar los resultados a fin de optimizar la eficiencia de las sesiones de fisioterapia.

Este objetivo se ha logrado gracias a la implementación de una interfaz gráfica para uso de terapeutas, accesible desde el navegador del personal médico, que a su vez accede un Servicio Web instalado en un servidor Apache Tomcat. Además se ha hecho uso de una base de datos MySQL para el almacenamiento de información personal de profesionales y pacientes, historial de configuraciones y resultados de los ejercicios.

Se han diseñado tres roles para los usuarios de la plataforma: terapeuta, administrador –estos clasificados en centros médicos- y superadministrador de la plataforma, con distintos permisos cada uno, y se han utilizado algunas técnicas de encriptado para securizar las contraseñas, con el propósito de convertirla en segura y confidencial, otro de los objetivos iniciales.

Con el fin de potenciar la escalabilidad de la plataforma, se ha estructurado la base de datos de manera que un juego consta de uno o más ejercicios, con la intención de que se puedan añadir, modificar o borrar si es necesario, permitiendo adaptar los juegos y ejercicios disponibles al desarrollo de la investigación global.

Por otro lado, se ha modificado el *middleware* "Chiro", instalado en el PC del paciente, que el ingeniero Ignacio Gómez-Martinho implementó en su Proyecto de Fin de Grado, mediante el cual se establece una comunicación entre los videojuegos del entorno Blexer y los distintos dispositivos con los que se puede jugar (cámara Kinect, gafas VR y samartphones). La modificación de Chiro incluye la autenticación del paciente, y posterior descarga de configuraciones de los distintos ejercicios y envío de resultados a la plataforma Web, a través del Servicio Web antes mencionado.

La realización de este Proyecto de Fin de Carrera permitirá continuar con la investigación global de la que forma parte, permitiendo la realización de pruebas, y, con la participación de personal médico, interpretar los resultados.

Conforme se ha desarrollado este proyecto, han surgido algunas ideas que sería interesante incorporar en un futuro. Éstas se detallan en el capítulo 6. "Trabajos Futuros".

6. Trabajos futuros

Los trabajos futuros que se proponen para el entorno de Blexer-Med se han clasificado según el usuario al que afectaría y el carácter de la propuesta:

- Interfaz gráfica del terapeuta:
  - Asociación de cada paciente a un terapeuta de forma que los demás no tengan acceso a sus datos a menos que dicho terapeuta o el administrador les autorice.
  - Posibilidad de habilitar/deshabilitar un juego para un paciente, determinar cuántas veces a la semana debe practicar, etc.
  - Mostrar gráficas con los resultados de las partidas.
  - Previsualización del juego y de la configuración, como ya se hace en el proyecto REWIRE mencionado en antecedentes.
- En la interfaz gráfica del superadministrador:
  - Elección de qué tipo de dato (número entero, texto, número de caracteres, valor máximo…) se espera recibir en los parámetros de configuración cuando el superadministrador da de alta un ejercicio. Actualmente todos los parámetros se tratan como texto.
  - Posibilidad de escoger si un parámetro de configuración es obligatorio o no a rellenar por el terapeuta. (De momento todos son obligatorios).
- Para todas las interfaces graficas:
  - Sistema de Logs para terapeutas, administradores, superadministradores con el fin de garantizar no repudio de las acciones.
- Seguridad:
  - Utilizar JSP (*Java Server Pages*) para ocultar el código JavaScript y que no se pueda inspeccionar desde el navegador. Esto se debe a que JSP se ejecuta en el servidor y JavaScript en el cliente.
  - Proteger los ficheros de configuraciones y resultados para que no puedan ser borrados accidentalmente en el PC del paciente o modificados manualmente sin permiso.
- Trabajos futuros que no competen únicamente a la plataforma Web:
  - Mostrar videos en la web de las partidas jugadas por lo pacientes.
  - Elección de las partes del cuerpo con que puede jugar cada paciente a cada juego, para no restringirle el acceso a algunos juegos.
  - Comunicación entre el paciente y el terapeuta, en forma de mensajes o chats.

7. Bibliografía

- M. Pirovano, R. Mainetti, G. Baud-Bovy, P.L. Lanzi, and N. A. Borghese, "Intelligent Game Engine for Rehabilitation (IGER)", *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 8, no. 1, March 2016.

- N. J. Nunes, B. Selic, A. Rodrigues da Silva y A. T. Alvarez, *UML Modeling Languages and Applications*, Springer, 22 feb. 2005.

- Lamyae Sardi, Ali Idri, José Luis Fernández-Alemán, "Journal of Biomedical Informatics",Volume 71, July 2017, Pages 31–48

- MySQL Community Edition disponible en: https://www.mysql.com/products/community/ [último acceso 28 de junio 2017].

- Fisiofocus, "Cinto Apps imprescindibles para fisioterapeutas". Artículo : http://www.fisiofocus.com/es/articulo/cinco-apps-imprescindibles-para-fisioterapeutas, [último acceso 28 de junio 2017].

- Freepic, Flaticon - Base de datos para iconos libres http://www.flaticon.com/ [último acceso 28 de junio 2017].

8. Referencias

[1] Ignacio Gómez-Martinho González, "Desarrollo e implementación de middleware entre Blender, Kinect y otros dispositivos", Proyecto de Fin de Grado, Universidad politécnica de Madrid. Julio 2016.

[2] Asociación de Investigadores en eSalud (AIES) http://aiesalud.com/ [último acceso 28 de junio 2017].

[3] "La eSalud" Página oficial. http://laesalud.com/ [último acceso 28 de junio 2017].

[4] REWIRE, proyecto parcialmente financiado por la Comisión Europea en el 7º programa marco, FP7-ICT-2011 Call 7 - Personal Health System (PHS), https://sites.google.com/site/projectrewire/ [último acceso 28 de junio 2017].

[5] MIRA Rehab, Proyecto de Rehabilitación, página oficial:

http://www.mirarehab.com/ , [último acceso 28 de junio 2017].

[6] MOTMI, Plataforma de Rehabilitación Virtual. Página oficial:

http://motmi.rehab/es/, [último acceso 28 de junio 2017].

[7] Educere, Proyecto de I+D+i por la UPM, UAM, UA, página oficial:

https://educeremus.wordpress.com/ [último acceso 28 de junio 2017].

[7] Tania Matamoros Santamaría, "Plataforma de consenso entre profesionales para evolución de conocimiento experto", Proyecto Fin de Grado, Universidad Politécnica de Madrid. Junio 2016.

[9] VirtualRehab http://www.virtualrehab.info [último acceso 28 de junio 2017].

[10] https://www.oracle.com/es/mysql [último acceso 28 de junio 2017].

[11] https://docs.oracle.com/cd/E19528-01/820-0888/auto22/index.html [último acceso 28 de junio 2017].

[12] https://www.etsisi.upm.es/sites/default/files/curso_2013_14/MASTER/MIW.JEE.POOJ.pdf [último acceso 28 de junio 2017].

[13] http://docs.oracle.com/javaee/6/tutorial/doc/gijqy.html [último acceso 28 de junio 2017].

[14] http://docs.oracle.com/javaee/6/tutorial/doc/gilik.html#ginpw [último acceso 28 de junio 2017].

[15] http://profesores.fi-b.unam.mx/carlos/java/java_basico3_4.html [último acceso 28 de junio 2017].

[16] http://www.sparxsystems.com.ar/resources/tutorial/uml2_sequencediagram.html [último acceso 28 de junio 2017].

Anexo A: Manual de Usuario Superadministrador

A.I. Gestión de la cuenta de superadministrador

- Editar perfil: El superadministrador del sistema es una figura, no una persona con nombre y apellidos, por lo que solo se puede editar su nombre de usuario. El nombre se usuario es único para cada superadministrador y aparecerá un mensaje de error si se intenta modificar a uno que ya existe.

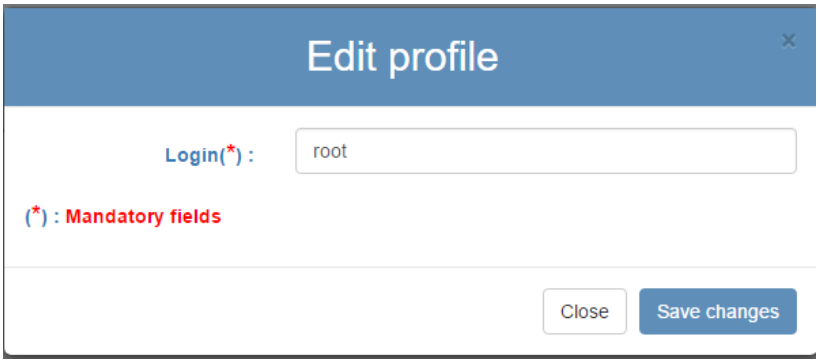


Figura 87 - Formulario para editar perfil del superadministrador

- Restablecer contraseña: Este formulario es el mismo para editar la contraseña de superadministradores, administradores y usuarios. Hay que introducir la contraseña vigente, introducir una nueva y confirmarla. La plataforma primero comprueba que la contraseña vigente sea correcta y luego que la nueva contraseña sea igual que la confirmación. Por último, comprueba que la nueva contraseña no sea la misma que la antigua.

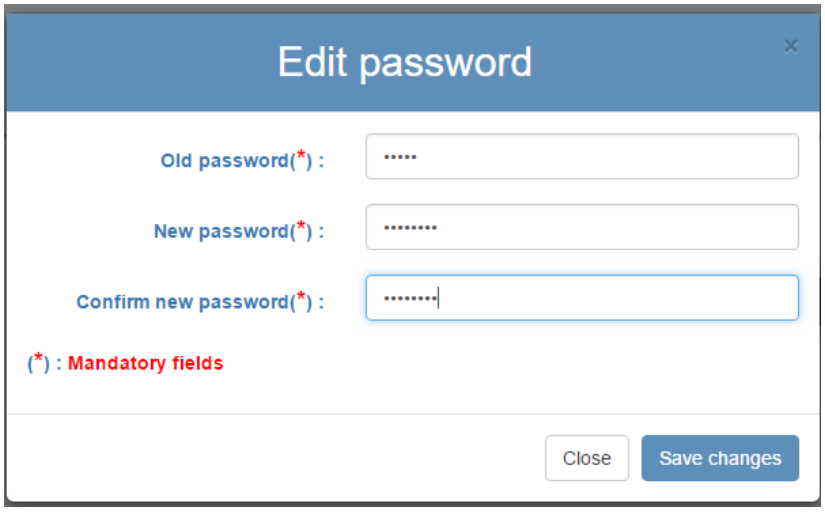


Figura 88 - Formulario para resetear contraseña del superadministrador

A.II. Gestión de centros médicos

- Descargar centros: Bajo la tabla que muestra los centros se le da la opción al superadministrador de descargar la información de todos los centros. Al pulsar sobre este botón, automáticamente se descarga un documento de Excel con la siguiente estructura.

	A	B	C	D	E	F	G
1	Id center	Creation date	Name	Address	Postal code	City	Country
2	1001	10-05-2017 19:40:00	Hospital Puerta de Hierro	Calle Manuel de Falla, 2	28222	Majadahonda	España
3	1002	12-05-2017 00:54:00	Hospital La Paz	Paseo de la Castellana, 261	28046	Madrid	España

Figura 89 - Descarga de centros médicos del sistema

- Añadir centro: Al pulsar sobre añadir un centro se despliega el siguiente formulario que solicita el nombre del centro, su dirección, código postal, ciudad y país, siendo obligatorios todos los campos.

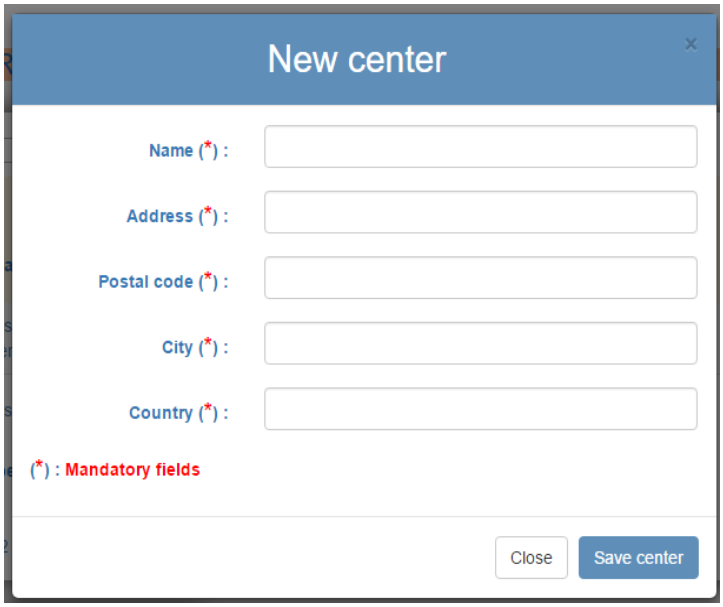


Figura 90 - Formulario añadir centro

- Editar centro: Cuando el superadministrador solicitar la edición de un centro concreto, aparece este formulario que se autocompleta con la información del centro seleccionado. Para enviar el formulario, se comprueba que al menos un dato haya cambiado.

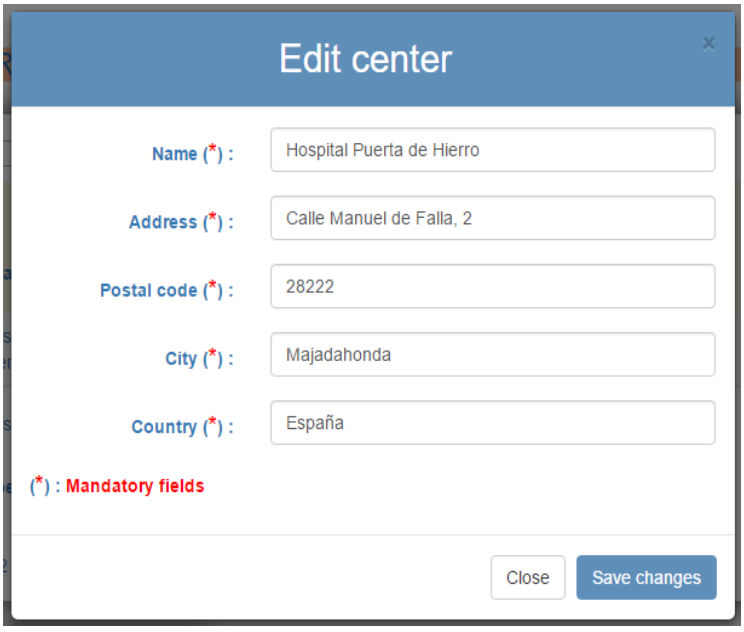


Figura 91 - Formulario editar centro

- Borrar centro: Por seguridad, cuando se solicita borrar un centro, aparece un aviso de confirmación –primer mensaje de la Figura 92-. Si se confirma que se quiere borrar dicho centro y además en ese centro hay usuarios con resultados guardados de partidas jugadas –los usuarios, configuraciones y resultados también se borrarán-, se da la opción de descargar los resultados de estos usuarios para todos los juegos y ejercicios (segundo mensaje de la Figura 92)

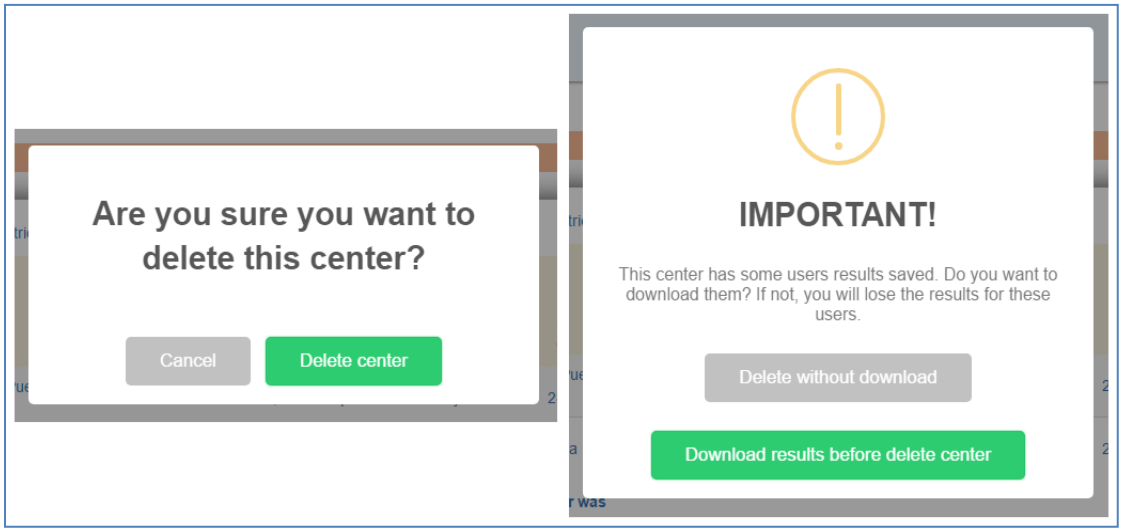


Figura 92 - Mensajes de aviso antes de borrar centro

A.III. Gestión de administradores de los centros

- Asignar administrador a un centro: La operación de asignar un administrador a un centro, se realiza a través del formulario de la Figura 93, donde el superadministrador debe elegir a qué centro quiere asociar este administrador.

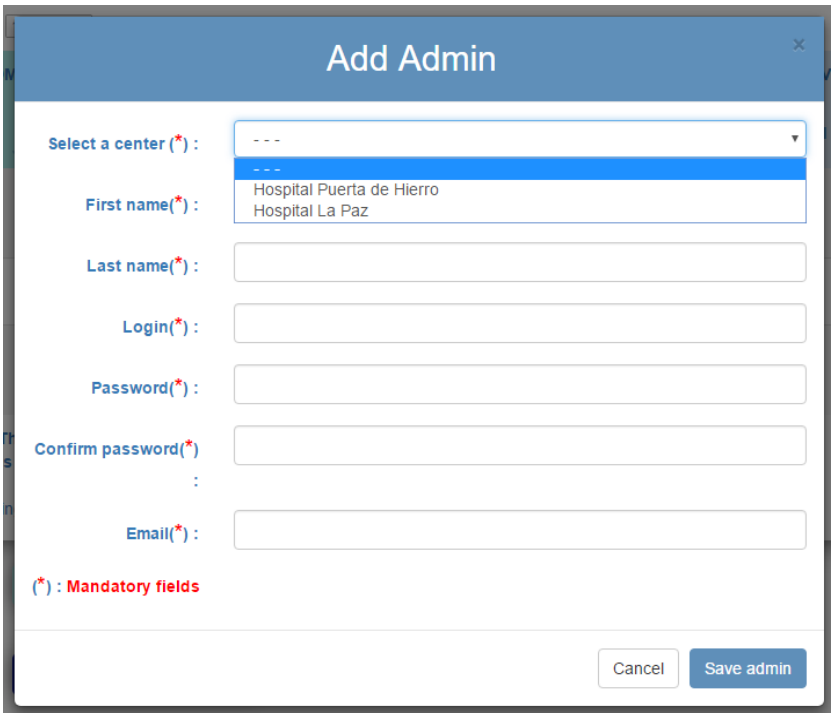


Figura 93 - Añadir administrador

- Restablecer contraseña de administrador: Para restablecer la contraseña del administrador, desde la interfaz de superadministrador, no es necesario insertar la contraseña antigua, sino directamente la nueva contraseña y confirmación.

Figura 94 - Resetear contraseña de admin.

- Descargar información de los administradores del sistema: La información que se descarga sobre los administradores del los centros aparece en la figura 95.

	A	B	C	D	E	F	G
1	Id admin	First Name	Last Name	Login	Email	Id center	Center
2	6	Lorena	Fernandez	lorena	lorena@hotmail.comm	1001	Hospital Puerta de Hierro
3	7	Helena	Arenas	helena	helena@hotmail.com	1002	Hospital La Paz
4							

Figura 95 - Descarga de administradores del sistema

A.IV. Gestión de juegos

- Descargar todos los juegos: La información que se descarga acerca de los juegos es la siguiente:

	A	B	C	D
1	Id game	Alias	Title	Description
2	5001	RACE	Cars Race	En este juego hay que cond...
3	5002	FLY	Fly around the world	En este ejercicio Juego hay ...
4	1	PHIBY	Phibys adventures	Este juego consta de vario....
5				

Figura 96 - Descargar información de los juegos del sistema

- Editar juego: El formulario de edición de un juego es muy similar al de añadir un juego, pero en este se autocompletan los campos del juego seleccionado para modificar. Todos los campos del formulario (Figura 97) son requeridos.

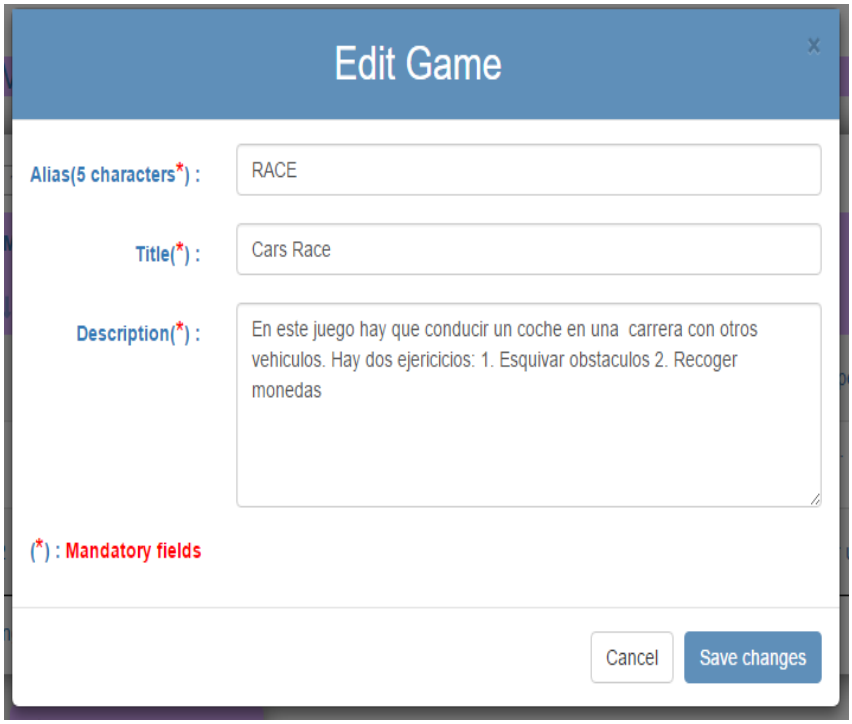


Figura 97 - Formulario para editar juego

A.V. Gestión de ejercicios

- Descargar todos los ejercicios del sistema: De todos los ejercicios de todos los juegos se descarga la siguiente información:

  - *Exercise id*
  - *Exercise alias*
  - *Exercise title*
  - *Exercise description*
  - *Game id*
  - *Game title*
  - *Parameter1 name*
  - *Parameter1 description*
  - *Parameter2 name*
  - *Parameter2 description*
  - *Parameter3 name*
  - *Parameter3 description*
  - *Parameter4 name*
  - *Parameter4 description*

- Consultar parámetros de configuración de un ejercicio: Cuando en la tabla de la vista de ejercicios del superadministrador se solicita ver los parámetros de configuración, se visualiza una ventana emergente con el siguiente aspecto:

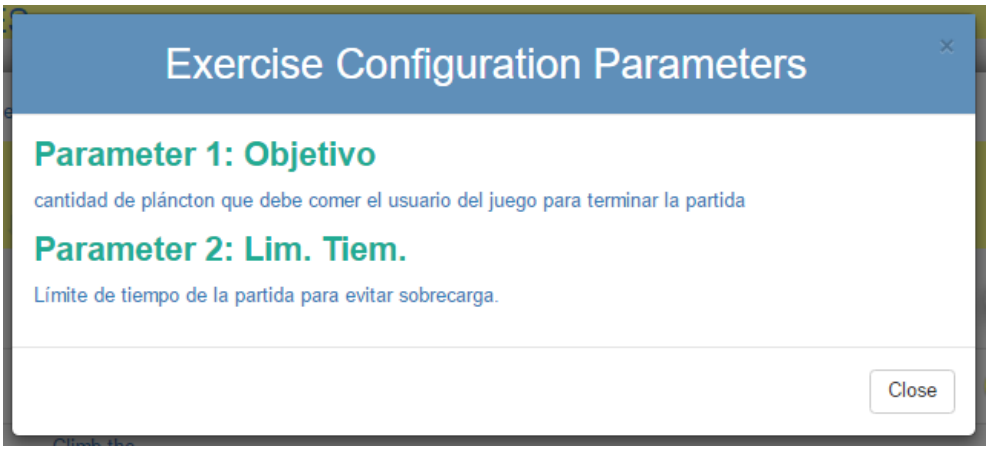


Figura 98 - Consultar parámetros de un ejercicio

- Editar ejercicio: De un ejercicio son editables el código de 5 caracteres, tu título y su descripción (Figura 99), pero no los parámetros de configuración como se ha explicado en el apartado 4.3.4. “Interfaz gráfica para superadministrador”

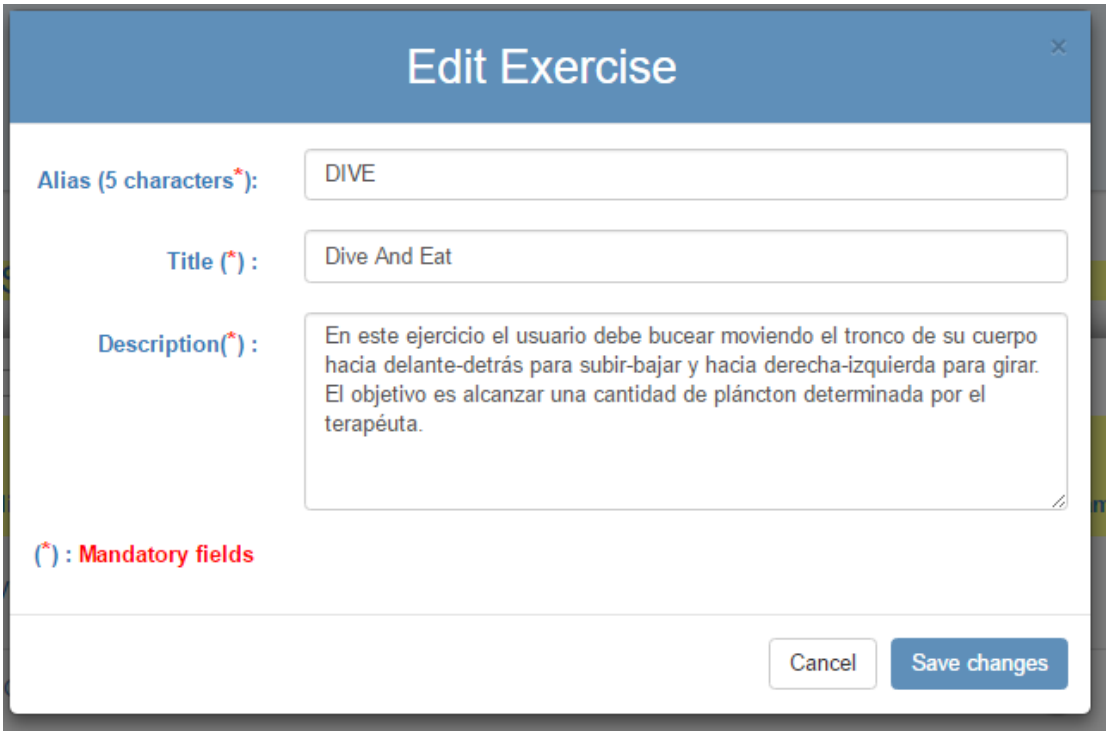


Figura 99 - Formulario para editar ejercicio

Anexo B: Manual de Usuario para el Administrador

B.I. Gestión de la cuenta de administrador

- Editar perfil: A diferencia del superadministrador, el administrador sí representa una persona física, por lo que su perfil tiene nombre, apellidos, nombre de usuario y *email*. Todos los campos son obligatorios, como se observa en la Figura 100. El nombre de usuario y el email no deben coincidir con los de otro terapeuta, en cuyo caso aparecerá un mensaje de error.

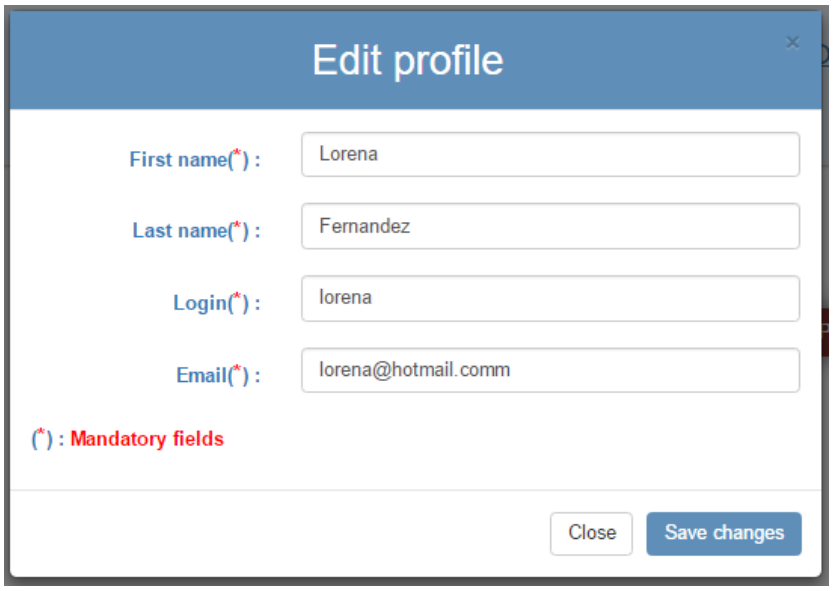


Figura 100 - Formulario para editar perfil de administrador

- Restablecer contraseña: El formulario es el mismo que para restablecer la contraseña del superadministrador (Figura 88).

B.II. Gestión del centro médico

- Editar información del centro: El formulario es el mismo que el de "Editar centro" desde la cuenta de superadministrador (Figura 91).

B.III. Gestión de terapeutas

- Descargar información de los terapeutas del centro: Al pulsar sobre el botón "Descargar terapeutas" bajo la tabla de terapeutas del administrador de un centro, se descarga automáticamente un documento (Figura 101) con la información de todos los terapeutas de dicho centro.

	A	B	C	D	E
1	Id clinician	Last Name	First Name	Login	Email
2	2000	Perez	Maria	mariaperez	maria@hotmail.com
3	2001	de la serna	Rodrigo	rodrigo	rodrigo@hotmail.com
4	2002	Gonzalez Mai	Leticia	leticia	leticia@hotmail.com
5	2003	Pineda Arrubl	Tomás	tomas	tomas@hotmail.com
6					

Figura 101 - Descargar información de terapeutas del centro

- Añadir terapeuta al centro: Para añadir un nuevo terapeuta a un centro, desde la cuenta de un administrador, se requiere su nombre, apellidos, nombre de usuario, contraseña y *email*. Se mostrará un mensaje de error en caso de que el nombre de usuario o el *email* estén asignados a otro terapeuta.

Figura 102 - Formulario para añadir terapeuta

- Editar información de un terapeuta: La información del terapeuta es editable desde el formulario de la Figura 103. Antes de enviar la información, la plataforma comprueba que al menos un dato haya sido modificado.
- Restablecer contraseña de un terapeuta: El formulario es el mismo que tiene el superadministrador para restablecer la contraseña un administrador (Figura 94). No hace falta conocer la contraseña antigua del terapeuta pues el administrador tiene permisos para restablecerla.

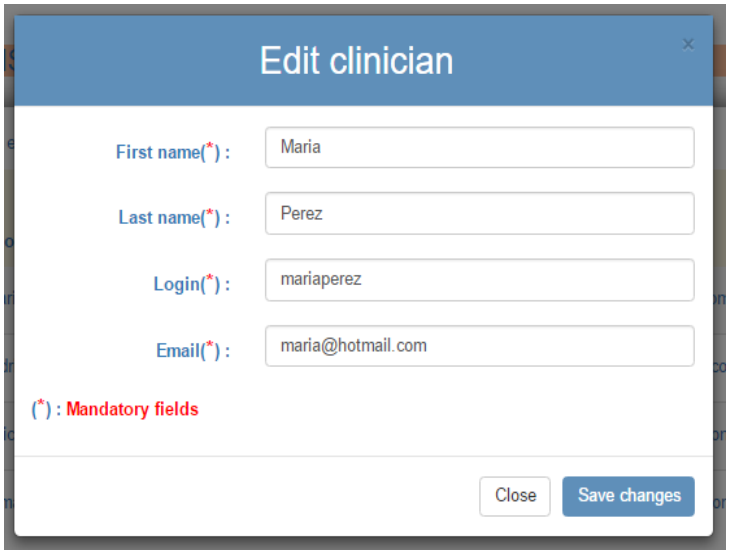


Figura 103 - Formulario editar terapeuta

B.IV. Gestión de pacientes

- Consultar información de un paciente del centro: Cuando se pulsa sobre el botón "Ver información del paciente" se despliega el diálogo de la Figura 104. En este, se puede observar la misma información que en la tabla de pacientes y además su nombre de usuario y contraseña –para acceder al *middleware* Chiro y descargar las configuraciones de un ejercicio-, las observaciones, teléfono de contacto y *email*.

Figura 104 - Ver información de un paciente

- Añadir paciente al centro: Para añadir un paciente al centro –se puede hacer con permisos de administrador y de terapeuta- se despliega el formulario de la Figura 105, donde son obligatorios los campos "nombre, "apellidos", "fecha de nacimiento", "nombre de usuario para acceder a Chiro" y "contraseña para acceder a Chiro". Por el contrario, no son obligatorios los campos "observaciones", "teléfono" y "email", aunque siempre son modificables a posteriori. La plataforma comprobará que el nombre de usuario no esté ocupado por otro paciente.

Figura 105 - Añadir paciente

- Editar información de un paciente: Los campos son exactamente los mismos que en el formulario "Añadir paciente" pero el contenido se autocompleta con la información del paciente a modificar. De nuevo se comprueba que algún dato haya sido modificado y que el nombre de usuario no esté repetido.
- Borrar cuenta de un paciente: Igual que al borrar cualquier elemento de la plataforma, se despliega un diálogo solicitando confirmación (Figura 92). No obstante, si el paciente que se solicita borrar tiene resultados almacenados, un segundo diálogo propone al administrador descargar los resultados guardados para este paciente.
- Descargar información de los pacientes del centro: La información de los pacientes de un centro se descarga en el formato de la Figura 106.

	A	B	C	D	E	F	G	
1	Id patient	Joined Blexer	Last Name	First Name	Login	Birthdate	Phone	Email
2	3006	27 Jun 2017	Arias	Maria	maria	03 Jan 2005		
3	3007	27 Jun 2017	Hernán	Oscar	oscar	15 Oct 2009	654654654	
4	3004	27 Jun 2017	León Osorio	Luis	luis	18 May 2004		
5	3005	27 Jun 2017	Molina	Beatriz	beatriz	07 Sep 2000		
6	3001	17 Jun 2017	Ramirez	Pepe	pepe	01 May 1990	91 324 84 75	pepe@h
7	3002	17 Jun 2017	Ramos	Olga	olga	01 Feb 1992		
8	3003	27 Jun 2017	Trujillo	Julio	julio	10 May 1995	654654654 (madre de Julio)	
9								

Figura 106 - Descargar información de pacientes del centro

Anexo C: Manual de Usuario para el Terapeuta

C.I. Gestión de la cuenta de terapeuta

- Editar perfil: El formulario es el mismo que para editar la información del administrador (Figura 93), pues un terapeuta es una persona física igual que el administrador y al contrario que el superadministrador.
- Restablecer contraseña: El formulario es el mismo que para restablecer la contraseña del superadministrador (Figura 94) y del administrador.

C.II. Gestión de pacientes

- Añadir un paciente al centro: Mismo formulario que para dar de alta un paciente desde una sesión de administrador (Figura 105)

C.III. Configuración de ejercicios

- Eliminar una configuración: Al eliminar una configuración hecha por un terapeuta, se pide una confirmación como el mensaje de la Figura 92.
- Consultar los parámetros de configuración de un ejercicio: El terapeuta podrá consultar la descripción de los parámetros con los que puede configurar un ejercicio en caso de que el nombre no sea lo suficientemente explicativo. Esta descripción es la que proporcionó el terapeuta en el formulario de "Añadir ejercicio".

Figura 107 - Consulta de significado de parámetro "Objetivo"

- Escribir/editar/borrar un comentario sobre una configuración: La plataforma permite al terapeuta que hizo una configuración –únicamente a este terapeuta- añadir un comentario o anotación sobre dicha configuración. Posteriormente, este comentario se puede configurar e incluso borrar.

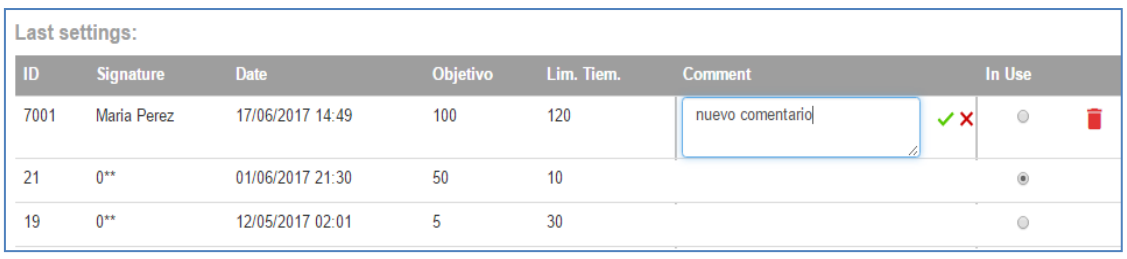
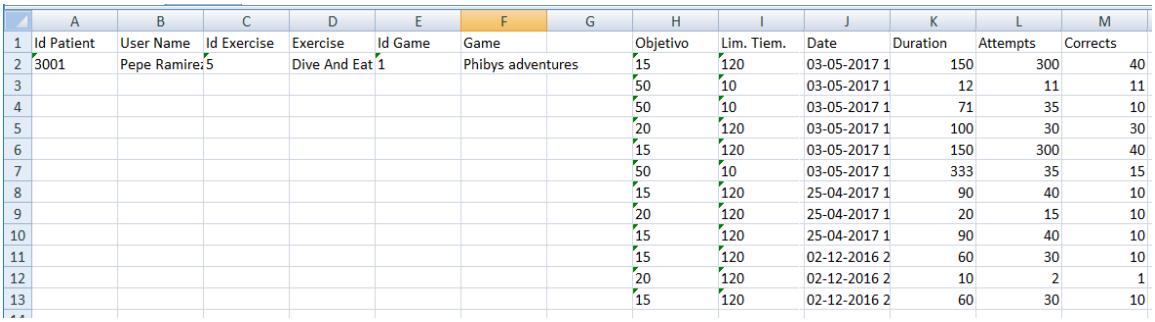


Figura 108 - Detalle de comentario en una configuración

C.IV. Consultar resultados de un ejercicio

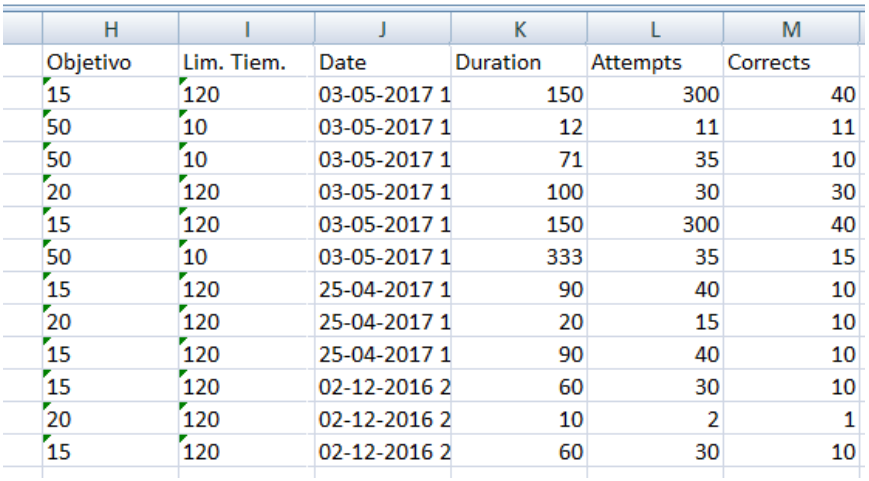
- Descargar resultados: Esta última es una de las funcionalidades más interesantes de la plataforma, poder descargar los resultados de la rehabilitación de un paciente, con el fin de poder cuantificar los avances o recesos. De esta forma, se puede trabajar con los datos en otros programas de cálculo u otras herramientas. Al descargar los resultados de un ejercicio para un paciente concreto, se descarga la información básica del paciente y la información mínima para identificar el ejercicio y el juego al que pertenece. Para cada partida jugada, se descarga la información de la configuración con la que se jugó.



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Id Patient	User Name	Id Exercise	Exercise	Id Game	Game		Objetivo	Lim. Tiem.	Date	Duration	Attempts	Corrects
2	3001	Pepe Ramire:	5	Dive And Eat	1	Phibys adventures		15	120	03-05-2017 1	150	300	40
3								50	10	03-05-2017 1	12	11	11
4								50	10	03-05-2017 1	71	35	10
5								20	120	03-05-2017 1	100	30	30
6								15	120	03-05-2017 1	150	300	40
7								50	10	03-05-2017 1	333	35	15
8								15	120	25-04-2017 1	90	40	10
9								20	120	25-04-2017 1	20	15	10
10								15	120	25-04-2017 1	90	40	10
11								15	120	02-12-2016 2	60	30	10
12								20	120	02-12-2016 2	10	2	1
13								15	120	02-12-2016 2	60	30	10

Figura 109 - Descargar resultados de un ejercicio y un usuario

Los datos ampliados de la configuración y los resultados de la partida aparecen en la Figura 110.



H	I	J	K	L	M
Objetivo	Lim. Tiem.	Date	Duration	Attempts	Corrects
15	120	03-05-2017 1	150	300	40
50	10	03-05-2017 1	12	11	11
50	10	03-05-2017 1	71	35	10
20	120	03-05-2017 1	100	30	30
15	120	03-05-2017 1	150	300	40
50	10	03-05-2017 1	333	35	15
15	120	25-04-2017 1	90	40	10
20	120	25-04-2017 1	20	15	10
15	120	25-04-2017 1	90	40	10
15	120	02-12-2016 2	60	30	10
20	120	02-12-2016 2	10	2	1
15	120	02-12-2016 2	60	30	10

Figura 110 - Resultados de un ejercicio

Anexo D: Formato JSON de configuraciones descargadas

Según la estructura de clases explicada en el apartado 4.4.2. "Descarga de configuraciones para un usuario de Blexer-Med", Chiro recibe la configuración de un usuario en el siguiente formato.


```
{
"id_user":3001,
"login":"pepe",
"first_name":"Pepe",
"last_name":"Ramirez",
"gamesMap":{
	"[5001, RACE, Cars Race]":{},
	"[5002, FLY, Fly around the world]":{
		"[6002, PLANE, Drive The Plane]":{
"id_setting":7003,
"param1":"200",
"param2":"null",
"param3":"null",
"param4":"null"
}
	},
	"[1, PHIBY, Phibys adventures]":{
		"[6, CHOP, Chop the wood]":{
"id_setting":12,
"param1":"50",
"param2":"60",
"param3":"null",
"param4":"null"
},
		"[8, ROW, Row the boat]":{
"id_setting":20,
"param1":"60",
"param2":"150",
"param3":"null",
"param4":"null"
},
		"[5, DIVE, Dive And Eat":{
"id_setting":21,
"param1":"50",
"param2":"10",
"param3":"null",
"param4":"null"
},
		"[7, CLIMB, Climb the tree]":{
"id_setting":18,
"param1":"14",
"param2":"30",
"param3":"null",
"param4":"null"
}
		}
	}
}
```

En amarillo se indica la información de usuario, en azul, la información de cada juego (mapa de juegos) y en verde, la información de cada ejercicio (un mapa de ejercicios para cada juego). Para cada ejercicio, se indica la configuración en uso –si hay una configuración hecha-. Los parámetros cuyo valor es "null" no son configurables por el terapeuta porque así lo decidió el superadministrador. El fichero se escribirá en la ruta "Chiro/users-settings" bajo el nombre *"login_usuario.*txt"

Anexo E: Formato JSON de resultados guardados

El fichero de resultados que Chiro espera leer –en el caso de que haya resultados guardados- debe tener el formato que aparece en la Figura 111. El fichero debe estar situado en la ruta "Chiro/users-results/" y llamarse "*login_usuario.txt*"

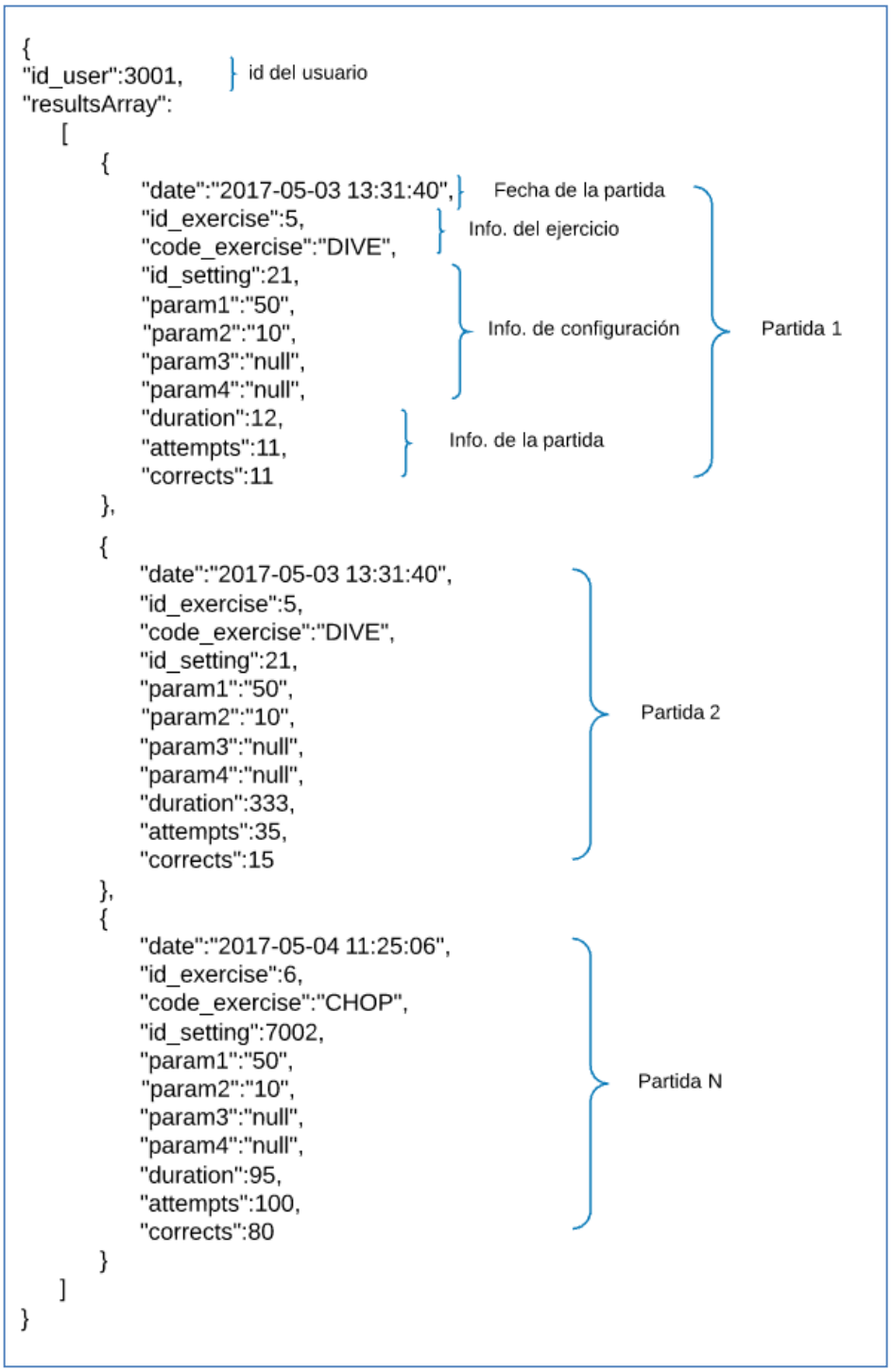


Figura 111 - Formato de fichero de resultados de un usuario

ANEXO F: Costes del Proyecto

Presupuesto de los costes generales del proyecto:

- Equipo……..... 1300 €
-
  - Interl Core i7-4790 a 3.6 GHz
  - 16GB RAM DDR3
  - 1000 GB Disco Duro 5400 rpm
  - 128 GB SSD
  - 4 GB ATI Radeon

- Material de oficina…….. 200€

- Horas de trabajo de un Ingeniero de Telecomunicaciones…….. 24.000€
  - 8 meses
  - 25 horas semanales
  - 30 €/hora

Total……. 25.500 €