Sistemas informáticos

Curso 2005-2006

Sistema CBR para presentación de entrenamientos físicos personalizados en Internet

Mónica González Jenal
Silvia Martín Alejo
Raquel Ramos López

Dirigido por:
Profesor Luis Hernández Yáñez
Departamento Sistemas Informáticos y Programación

Facultad de Informática
Universidad Complutense de Madrid

"Puedes llegar a cualquier parte,
siempre que andes lo suficiente."

Lewis Carrol

AGRADECIMIENTOS

Queremos agradecer a Juan Antonio Recio García y a Luis Diez Hernández su inestimable ayuda e infinita paciencia, a la que bien hemos puesto a prueba, en todo lo relacionado con la herramienta jCOLIBRI y el proyecto en general.

Además queremos darle las gracias a Albert Meco por su tiempo y valiosos consejos en lo referente al diseño en 3D (¡Algún día le haremos la competencia a Pixar!).

Mil gracias también a Luis Hernández Yáñez por prestarse a dirigir este proyecto atípico, demostrando así que en esta facultad es posible materializar una idea propuesta por alumnos. ¡Nunca olvidaremos sus mensajes de ánimo, ni escribir un punto al final de cada frase!

Por último, pero no menos importante, gracias a nuestras familias, compañeros y amigos. Por su cariño, apoyo y humor en los momentos en que más lo necesitábamos.

A todos: ¡Gracias!

Resumen

El proyecto ha consistido en el desarrollo de un sistema experto que asesora a los usuarios sobre entrenamientos físicos personalizados que resulten adecuados para cada condición física y cada objetivo de entrenamiento.

El sistema hace uso de razonamiento basado en casos (*Case Based Reasoning*), siendo el dominio de conocimiento el entrenamiento médico-deportivo. Los casos describen tanto las características de los usuarios, como los entrenamientos que les corresponden. Los entrenamientos completos están formados secuenciando apropiadamente unidades de entrenamiento individuales.

Los usuarios acceden al sistema por medio de una interfaz Web que permite tanto consultar la realización de ejercicios individuales como la obtención de un entrenamiento adaptado. La presentación que se muestra a los usuarios incluye, además de la explicación de cada unidad de entrenamiento, modelos que ejemplarizan las distintas posiciones que debe adoptar el cuerpo para ejecutar un ejercicio correctamente. Esos modelos consisten en representaciones humanoides creadas con programas de diseño gráfico de 2D y 3D.

Palabras clave: Sistemas expertos, Razonamiento basado en casos, Entrenamiento deportivo, Gráficos 2D y 3D, Interfaces Web.

Abstract

The aim of this project was to develop an expert system that advises the users on the customized physical training that suits best each physical condition as well as training goals.

The systems works with case-based reasoning (CBR) and the knowledge used is of the medical - sports training domain. The cases describe the user's characteristics and the advised training programs, which are created by sequencing individual training units in an appropriate way.

The users access the system via a web interface that allows them to consult how a specific exercise has to be executed as well as to request a training program generated to suit their individual needs. The presentation includes an explanation of each training unit and graphic models that visualize and exemplify the different positions the body has to adopt to perform the exercise correctly. These models are humanoid representations created with software for the design 2D and 3D graphics.

Keywords: Expert Systems, Case Based Reasoning, Sport Training, 2D and 3D Graphics, Web Interfaces.

ÍNDICE

1. INTRODUCCIÓN

El bienestar físico es uno de los asuntos que más preocupa a la sociedad actual. Bien es sabido que nuestra vida es cada vez más sedentaria y nuestro tiempo libre cada vez más limitado y difícil de compatibilizar con actividades deportivas o entrenamientos de mantenimiento. Por otra parte, este bienestar físico implica un cuidado de nuestro cuerpo que repercute positiva y directamente tanto en nuestra salud como en nuestra actividad diaria personal y profesional.

Por estos motivos y reconociendo que las nuevas tecnologías proporcionan herramientas suficientes para la creación de un sistema basado en el conocimiento de un entrenador experto, decidimos proponer un proyecto de Sistemas Informáticos que tuviese disponibilidad y accesibilidad máxima para atender las necesidades de los sectores de población más sendentarios. Atendiendo de forma personalizada a su demanda de entrenamientos deportivos y actividad física.

Así pues, las siguientes páginas narran la investigación, el análisis, la toma de decisiones, el diseño, la implementación, las pruebas y las dificultades que hemos encontrado a lo largo de todo el proceso que implica la creación de nuestro entrenador personal particular.

Las secciones que siguen a continuación organizan toda la información relativa a este proceso de la siguiente forma: primero se especifican los requisitos iniciales y objetivos. Segundo, se investigan las principales tecnologías que ofrece el mercado y se continúa con la adquisición del conocimiento experto necesario para el óptimo funcionamiento del sistema de entrenamiento personal. Después, se sigue con la implementación: base de datos, módulo de usuario, razonamiento basado en casos, comunicación entre la lógica basada en casos y el módulo de usuario y por último la salida del sistema. Finalmente llegamos a la sección de conclusiones para terminar con el trabajo futuro que ampliaría y mejoraría el sistema.

Para satisfacer las exigencias y la curiosidad del lector escribimos también una última sección con toda la bibliografía y guía de consultas en las que se apoya nuestro trabajo.

2. ESPECIFICACIÓN DE REQUISITOS Y OBJETIVOS

1.1.INTRODUCCIÓN

Esta sección se divide en tres partes principales: la especificación de requisitos inicial del sistema, los objetivos que implica esta especificación, y finalmente los objetivos que cumplimos.

1.2.ESPECIFICACIÓN DE REQUISITOS

Nuestro proyecto de Sistemas Informáticos tiene como fin principal crear el prototipo de una herramienta que evalúe las características y los objetivos de los usuarios a la hora de someterse a un entrenamiento físico para ofrecerles una sesión de entrenamiento personalizada y adecuada a sus necesidades. Además, el sistema debe tener máxima accesibilidad y disponibilidad por lo que su uso será vía Web.

Por otra parte, el sistema presentará la información en una interfaz gráfica amigable que incluye componentes gráficos tanto 2D como 3D, siendo estos diseños originales nuestros.

El sistema está compuesto por los siguientes módulos:

- Adquisición de conocimiento experto.
- Interfaz y componentes gráficos del sistema.
- Lógica central del sistema.
- Comunicación entre la interfaz y la lógica del sistema.

Los requisitos relativos a cada uno de estos módulos son los siguientes:

Adquisición del conocimiento experto

- La adquisición de toda la información relativa a entrenamientos deportivos, características físicas de los usuarios, objetivos del entrenamiento y cuerpo humano debe ser proporcionada por personas o material altamente cualificado.
- Toda esta información debe usarse del mismo modo en que lo haría un experto. Por ello es un requisitito imprescindible incorporar este conocimiento a la lógica del sistema.

Interfaz y componentes gráficos del sistema

- La interfaz debe proporcionar toda la información en el formato requerido por la lógica central del sistema para recomendar el entrenamiento.
- Del mismo modo debe presentar visualmente el conjunto de ejercicios que componen el entrenamiento que el sistema recomienda al usuario concreto.

- La interfaz permite registrar a un usuario, modificar sus datos, eliminarle del sistema, consultar ejercicios, consultar ejecución de pruebas físicas, consultar entrenamientos y finalmente consultar la evolución de un usuario.
- La interfaz del sistema debe ser accesible, amigable e intuitiva.
- Los componentes gráficos diseñados por nosotras tienen como objetivo principal ser imágenes autoexplicativas de la parte del cuerpo que involucra el ejercicio o el modo de ejecución del ejercicio representado.

Lógica central del sistema

- La lógica central del sistema debe recuperar toda la información pertinente según el conocimiento experto para la correcta recomendación de los ejercicios.
- La lógica central del sistema debe estar compuesta de una base de conocimiento suficiente y adecuada a las características generales de los individuos tipo de la población.
- La lógica central del sistema debe evaluar qué caso es más similar a otro caso dado abstrayendo el modo en que lo haría un experto.
- La lógica central del sistema debe evaluar cuando el entrenamiento recomendado a un usuario debe ser diferente al de otro dado. En este caso es capaz de adaptar el entrenamiento a las características del nuevo usuario.
- La lógica central del sistema debe evaluar el resultado de la modificación decidiendo si el entrenamiento adaptado beneficia o satisface al usuario que se le recomendó.
- La lógica central del sistema debe evaluar cuando los casos adaptados enriquecen el contenido de la base de casos y almacenarlos para su posterior uso.
- La lógica central del sistema debe permitir tener una base de casos fácilmente mantenible.
- El sistema es fiable y recomienda ejercicios personalizados al usuario concreto del mismo modo en que lo haría un experto.

Comunicación entre la interfaz y la lógica del sistema

- La información transferida entre la interfaz y la lógica del sistema debe ser siempre “entendible” y adecuada para cada una de las partes.
- Nunca se perderá información transferida.

1.3.OBJETIVOS INICIALES

Como hemos indicado nuestro proyecto de Sistemas Informáticos responde a la necesidad de crear entrenamientos deportivos adecuados a las características personales de cada individuo concreto y a los objetivos que define cuando se propone entrenar.

El objetivo principal de cada uno de estos entrenamientos es que el usuario no sienta ninguna diferencia entre un entrenamiento propuesto por un entrenador profesional y el entrenamiento dado por el sistema. Por este motivo la lógica principal de la aplicación se basa en un sistema experto de razonamiento basado en casos apoyado en el uso de ontologías.

Por otra parte y para permitir la máxima disponibilidad de este sistema experto, adecuaremos el sistema al entorno de Internet. De este modo estaría disponible veinticuatro horas al día todos los días del año. Esta ventaja responde también a la necesidad de adaptarnos al máximo a las demandas del usuario ya que es él quién decide en que momento realizar el entrenamiento e Internet permite que el sistema esté disponible desde cualquier parte.

El soporte inteligente que debe tener el sistema es el siguiente:

- Base de casos que represente el conocimiento experto.
- Uso de conocimiento experto para encontrar el caso más similar al del usuario.
- Adaptación de la solución del caso si las características no son lo suficientemente adecuadas.
- Aprendizaje de los casos que enriquezcan el conocimiento de la base de casos.

Además, el sistema debe presentar una interfaz gráfica vía Web con elementos de diseño 2D y 3D originales.

El sistema pretende además realizar un seguimiento de la evolución del usuario y su entrenamiento.

1.4.OBJETIVOS CUMPLIDOS

Excepto el objetivo de realizar el seguimiento de la evolución del usuario y su entrenamiento hemos cumplido teóricamente todos los objetivos que nos planteamos. Es decir, hemos analizado las características de cada uno de ellos, hemos pensado distintas estrategias para su solución y finalmente nos hemos decantado por la más adecuada para proceder a su desarrollo y prueba.

Sin embargo, debido a las dificultades técnicas que nos planteó usar una versión Beta de la herramienta en la que fundamentábamos la lógica del sistema no pudimos terminar el desarrollo de todas las partes previstas. Concretamente, la adaptación y el aprendizaje están pendientes de finalizar. No obstante, este trabajo queda abierto y una vez que estas dificultades estén resueltas, el tiempo para tener lista estas partes será mínimo.

Por otra parte, mencionamos que el sistema tendría disponibilidad y accesibilidad máxima vía Web. De momento, el sistema no es altamente disponible debido a la misma causa: la versión Beta que implementa la herramienta en la que basamos la lógica central del sistema, de momento sólo permite trabajar en modo monousuario. En cualquier caso, el Grupo de Aplicaciones de la Inteligencia Artificial que la desarrolla la herramienta la dotará de esta funcionalidad en futuras publicaciones. Estas y otras cuestiones se analizarán en detalle en secciones posteriores.

3. FASE DE INVESTIGACIÓN

1.5.TECNOLOGÍAS [1]

1.5.1.JSP VS ASP

Cuando estudiamos las tecnologías que podíamos utilizar para codificar las páginas Web interactivas, en principio estuvimos pensando en usar ASP (*Active Server Pages*), ya que ASP y JSP (*Java Server Pages*) sirven para hacer, más o menos, el mismo tipo de aplicaciones Web. Pero investigando con más detenimiento descubrimos que existen bastantes diferencias entre las dos opciones. A continuación pasamos a detallar algunas de ellas:

JSP sigue la filosofía de la arquitectura JAVA de "escribe una vez y ejecuta donde quieras". La implantación de ASP está limitada para arquitecturas basadas en tecnología Microsoft.

JSP se puede ejecutar en los sistemas operativos y servidores Web más populares, como por ejemplo Apache, Netscape o Microsoft IIS. Mientras que ASP sólo tiene soporte nativo para los servidores IIS y Personal Web Server, que son los dos servidores Web para sistemas Microsoft. Los componentes JSP son reutilizables en distintas plataformas (UNIX, Windows).

La tecnología JSP usa Java como lenguaje de Script mientras que ASP usa VBScript o Jscript. Java es un lenguaje más potente y escalable que los lenguajes de Script. Las páginas JSP son compiladas en Servlets por lo que actúan como una puerta a todos los servicios Java de Servidor y librerías Java para aplicaciones http. Java facilita el trabajo del desarrollador ayudando por ejemplo a proteger el sistema contra "caídas" mientras que las aplicaciones ASP sobre sistemas NT son más susceptibles a sufrirlas. También protege contra fallos de memoria y simplifica el duro trabajo de buscar los fallos de perdida de punteros de memoria que ralentizan el funcionamiento de una aplicación. Las aplicaciones que usan JSP tienen un mantenimiento más fácil que las que usan ASP.

1.5.2.JAVA BEANS

Como tecnología de soporte para el mantenimiento de las páginas Web decidimos usar un *JavaBean* o *bean* que es un componente hecho en software que se puede reutilizar y que puede ser manipulado visualmente por una herramienta de programación en lenguaje Java.

Las propiedades de un *bean* pueden examinarse y modificarse mediante métodos o funciones miembro, que acceden a dicha propiedad, y pueden ser de dos tipos:

- Método ***get****ter*: lee el valor de la propiedad.
- Método ***set****ter*: cambia el valor de la propiedad.

1.5.3.SERVIDORES

1.5.4.Blazix

Para poder trabajar con JSPs es necesario un servidor Web. Como no teníamos experiencia en ese ámbito preguntamos al profesor del proyecto, quien nos recomendó probar con Blazix. Este servidor se ha utilizado en los seminarios de aplicaciones Web impartidos en nuestra facultad, porque ocupa muy poco espacio (solamente 1.5MB) y resulta fácil de instalar y aprender a manejar.

En la página Web oficial [2] encontramos la siguiente documentación:

Blazix is a high-performance full-featured Java application server. Blazix download size is a mere 1.5 megabytes. This low footprint, combined with the high precision code of Blazix, naturally results in very high performance. With Blazix, it is not necessary to purchase expensive hardware merely to recompense for inefficiencies in the application server.Because the design of Blazix combines elegance with sophistication, Blazix is also very easy for developers to work with. This translates into reduced development time and less bugs. Blazix can be used as an Application Server or as a full Web Server (serving HTML files and images in addition to the standard application server workload.) […]

En la sección de preguntas frecuentes encontramos la siguiente información adicional, que nos resultó de gran interés:

Q. Is IIOP supported for Blazix EJB?
A. Yes. The default protocol used in Blazix EJB is JRMP, because Blazix works both JDK 1.2 and above, and JDK 1.2 doesn't include IIOP. […]

Q. Which versions of technology specifications does Blazix comply with?
A. Blazix complies with EJB 1.1, Servlets 2.3, JSP 1.2 and JMS 1.0.2 specifications.

1.5.5.Tomcat

Problemas y soluciones

Aunque Blazix resultó muy útil en las pruebas iniciales, permitiéndonos ver el funcionamiento de los primeros JSPs que diseñamos, a la larga no resultó ser la mejor opción.

El principal problema fue que Blazix no era compatible con jCOLIBRI, que es la herramienta que genera los entrenamientos que posteriormente deben ser mostrados a través de los JSPs. El problema concreto era que jCOLIBRI necesitaba unos permisos y licencias que Blazix no habilitaba. Dado que el proceso de razonamiento llevado a cabo

en jCOLIBRI era fundamental para nuestro sistema, decidimos utilizar otro servidor Web para resolver este problema.

La elección de Tomcat

Tomcat es otro servidor de páginas Web, pero mucho más potente y flexible que Blazix y, además, compatible con jCOLIBRI.

Documentándonos sobre él [3] dimos con la siguiente información que resume bastante bien sus características:

Apache Tomcat *(formerly under the Apache Jakarta Project; Tomcat is now a top level project) is a web container developed at the Apache Software Foundation. Tomcat implements the servlet and the JavaServer Pages (JSP) specifications from Sun Microsystems, providing an environment for Java code to run in cooperation with a web server. It adds tools for configuration and management but can also be configured by editing configuration files that are normally XML-formatted. Because Tomcat includes its own HTTP server internally, it is also considered a standalone web server.*

Maintainer:	Apache Software Foundation
Stable release:	5.5.17 (26 Apr 2006) [+/-]
Preview release:	5.0.30 (date unknown) [+/-]
OS:	Cross-platform
Use:	Web container
License:	Apache license
Website:	tomcat.apache.org

Observación: La versión de Tomcat que utilizamos en este proyecto ha sido la 5.5

1.6.CBR

1.6.1.INTRODUCCIÓN A RAZONAMIENTO BASADO EN CASOS

Introducción

Un sistema CBR es un sistema de razonamiento basado en casos en la que la experiencia está en embebida en una biblioteca de casos pasados en vez de estar codificada con reglas clásicas. Cada caso contiene una descripción del problema, una solución y/o una salida. El conocimiento y razonamiento experto que se usa para resolver el problema no está grabado, pero está implícito en la solución.

Así, el problema actual se resuelve comparándolo con los casos de la base de casos y seleccionando el o los casos más similares. Los casos seleccionados se usan para sugerir una solución que se puede reutilizar y probar para comprobar su éxito. Si fuese necesario, además se puede revisar la solución. Finalmente, el problema actual, así como su solución, se pueden aprender y ser guardados como un nuevo caso [4].

El razonamiento basado en casos es similar al razonamiento humano, por este motivo muchas personas encuentran más apropiado esta forma de trabajar, ya que usa ejemplos en vez de conclusiones sacadas de su contexto. Una biblioteca de casos también es un recurso colectivo que permite a todo el mundo dentro de una corporación usar la biblioteca para tratar un problema nuevo. Veamos los siguientes ejemplos de razonamiento similar al humano

- Los médicos buscan conjuntos de síntomas que identifican al paciente con algún conjunto de casos previos.
- Los ingenieros toman muchas de sus ideas de soluciones previas ya construidas con éxito.
- Los programadores expertos reutilizan esquemas más o menos abstractos de las soluciones [5].

Por este motivo, desde los años 90 el área de interés de los CBR se ha ampliado desde un punto de vista tanto académico como comercial. Se desarrollan herramientas más maduras y aplicaciones enfocadas a conferencias.

Creemos que también es conveniente indicar que razonamiento CBR es un término genérico que se suele usar para describir técnicas que incluyen pero que no están limitadas a razonamiento basado en casos [4].

Pasemos ahora a analizar cuáles son las principales ventajas de este tipo de sistemas:

- Sencillez: resulta más sencillo adquirir nuevos casos que descubrir reglas y generalizaciones nuevas. Viéndolo en los ejemplos anteriores esto es: es más sencillo anotar un conjunto de síntomas con un diagnóstico (caso) que inferir las reglas que evaluamos teniendo un conjunto de síntomas dar ese diagnóstico.
- Rapidez: es más rápido reutilizar una solución previa que obtener la solución desde cero
- Los casos ayudan a un razonador a concentrarse en los aspectos importantes de un problema, al identificar las características definitorias
- Los casos pueden proporcionar también “información negativa”, alertando sobre posibles fallos o excepciones
- Mantenimiento de la base de conocimiento: los usuarios pueden añadir nuevos casos sin ayuda de los expertos (en nuestro caso los usuarios no serán

conscientes de estar añadiendo casos nuevos. Esta es una funcionalidad del propio sistema) [5].

La técnica CBR

Empezaremos a describir la técnica CBR con el conocimiento que incluyen los casos. Un caso puede verse como *"un fragmento de conocimiento que representa una experiencia y que enseña una lección para conseguir los objetivos del razonador"* [4]. Debemos ser conscientes de que no todas las situaciones "enseñan una lección": así que distinguiremos entre casos redundantes o casos cubiertos por el conocimiento general. Como dijimos un caso incluye:

- Descripción de la situación o el problema.
  - Objetivos, restricciones para la consecución de los objetivos.
  - Características de la situación.
- Solución.
  - La solución.
  - Cómo se obtuvo la solución.
  - Justificaciones de las decisiones tomadas en la solución.
- Resultado.
  - Si tuvo éxito o no.
  - En caso de fallo ¿por qué fallo? ¿qué estrategia de reparación se aplicó?
- Modelos del dominio (ontologías) que se utilizan en las distintas fases del CBR para el cómputo de similitud y búsqueda de sustitutos.

Todos los métodos de razonamiento basado en casos tienen un proceso común que consta de las siguientes etapas:

- Selección del caso o casos más similares en comparación con la biblioteca de casos pasados.
- Reutilización del caso seleccionado para intentar resolver el problema actual.
- Revisión y adaptación de la solución propuesta si fuera necesario.
- Aprendizaje de la solución final como un nuevo caso.

Ahora explicaremos más detalladamente el funcionamiento de estas cuatro fases:

Selección o recuperación de un caso

Los principales pasos que se siguen para la recuperación de un caso son los siguientes:

Primero se hace una valoración de la situación, se determinan las características que permiten encontrar casos relevantes para hacer una búsqueda en la memoria y encontrar los casos que guardan una similitud por encima de un cierto umbral con el caso a tratar. Esto también se conoce como comparación superficial. El procedimiento de búsqueda depende de la organización o la estructura de los casos. En ocasiones, los casos se organizan automáticamente utilizando métodos de aprendizaje máquina.

Segundo, se procede a la ordenación o ranking de los casos recuperados. Esto es ya una comparación más elaborada. Se puede usar para seleccionar el caso mejor. La representación de los casos puede ser variada. Por ejemplo:

- Listas de pares atributo-valor
- Registros de una base de datos
- Sistemas de marcos y redes semánticas, modelos de memoria
- Textos estructurados

La organización de la base de casos también.

- Lineal
- Árboles de decisión
- Árboles k-d
- Redes de activación
- Redes de recuperación de casos

Reutilización de casos

Se utiliza el conocimiento incluido en el caso recuperado para resolver o clasificar el problema o situación actual. Generalmente, se puede elegir entre dos opciones principales: reutilizar el caso sin modificarlo, es decir, la solución es válida o es el usuario quien se encarga de adaptarla o interpretarla. O bien reutilizar el caso modificándolo. Esto se logra adaptando la solución (resolución de problemas) o la justificación (interpretación de situaciones) de los casos recuperados.

Revisión de los casos

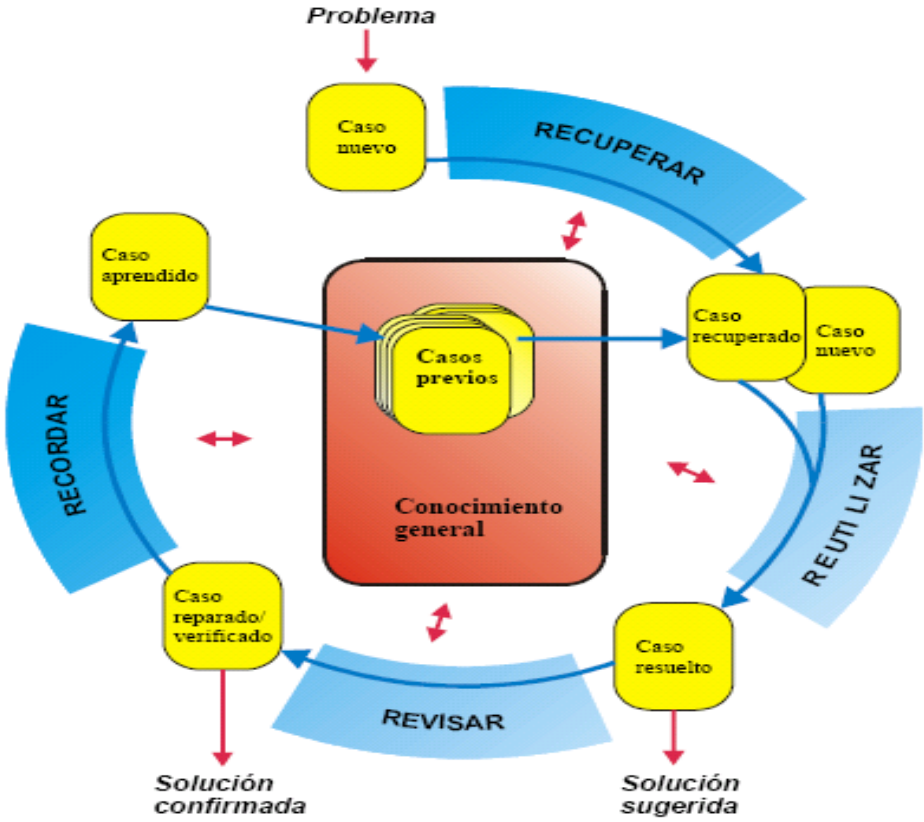
Esta etapa es necesaria para responder a la siguiente pregunta: ¿Es correcta la solución propuesta? Esta evaluación de la solución normalmente se realiza fuera del sistema CBR mediante la respuesta de un experto, el usuario o mediante una simulación.

Recuerdo – Aprendizaje de los casos

Como se puede suponer en CBR el razonamiento y el aprendizaje están íntimamente ligados. Gracias a esto el sistema mejora con el uso al ir adquiriendo nuevas

experiencias que integra adecuadamente. Esto también permite mejorar la eficiencia del sistema al disponer de más casos a partir de los cuales obtener soluciones. Por el contrario también corremos el riesgo de llegar a una base de casos tan grande que no sea mantenible fácilmente.

El siguiente gráfico [5] muestra el ciclo que acabamos de explicar: tenemos una base de casos previos. Nos llega un problema (caso nuevo) y lo comparamos con los casos previos que recuperamos (no recuperamos todos los casos, más adelante veremos que criterios seguiremos en nuestro proyecto para la recuperación de casos). Se usa el caso recuperado para sugerir una solución, se revisa (la fase de revisión no existe en todos los sistemas pero puede ser hecha a posteriori por el usuario, por ejemplo) y según esto se recuerda incluyéndolo en la nueva base de casos.



Hay una gran variedad de métodos para organizar seleccionar, utilizar e indexar el conocimiento guardado en los casos. La recuperación de un caso comienza con la descripción del problema y termina cuando se encuentra el caso más similar. Las tareas en las que se divide esta etapa consisten en [4]:

- Identificar el conjunto de los descriptores de problemas significativos.
- Evaluar el caso y devolver un conjunto de casos suficientemente similares.
- Seleccionar el mejor caso de los casos devueltos.

Algunos sistemas recuperan casos basándose en las similitudes sintácticas de la descripción. Sin embargo, los sistemas más avanzados usan similitudes semánticas.

La modificación de la solución del caso seleccionado se centra en identificar las diferencias entre el caso recuperado y el caso actual e identificar la parte del caso seleccionado que puede transformarse en el nuevo caso. Generalmente la solución del caso recuperado se puede usar directamente como solución del nuevo caso.

Por otra parte, es necesaria la revisión de la solución generada por el procedimiento de modificación para comprobar si realmente es correcta y darle al sistema la oportunidad de aprender de la experiencia.

El aprendizaje del caso es el proceso de incorporar la información útil del nuevo caso a la biblioteca de casos. Esto implica decidir que información y en qué forma se debe retener, cómo indexar el caso para su posterior recuperación y cómo integrarlo en la librería.

Finalmente podemos decir que toda buena herramienta CBR soporta estas cuatro fases y proporciona una buena variedad de mecanismos que permiten su uso combinado en caso de ser necesario. Además, la herramienta debe permitir el manejo de grandes bibliotecas de casos con tiempo de recuperación lineal con respecto al tamaño de la base de casos en el caso peor [5].

Aplicaciones

Desde los 90 los sistemas CBR se han aplicado a muchas herramientas comerciales creando así aplicaciones en una gran variedad de dominios. A continuación describimos los principales [4]:

- Diagnóstico: sistemas CBR de diagnóstico médico que basándose en la similitud de los síntomas de la base de casos con el caso actual seleccionan el que más se adecue para realizar un diagnóstico.
- Help-Desk: estos sistemas se usan en el área de servicio al cliente para resolver problemas con productos o servicios
- Evaluación: sistemas para determinar el valor de variables por comparación con el valor de una variable conocida similar. Las tareas de evaluación son bastante comunes en los dominios de marketing y finanzas.
- Soporte de decisión: ayudan a la hora de tomar una decisión frente a un problema complejo basándose en las decisiones tomadas con problemas análogos. CBR es especialmente bueno en consultas estructuradas en documentos modulares y no homogéneos.
- Diseño: sistemas que ayudan a desarrollar diseños industriales y arquitectónicos. Estos sistemas sólo asisten al usuario en una parte del proceso de diseño, que es el de seleccionar casos pasados. Por tanto, se debe usar de forma combinada con otra técnica de razonamiento para soportar el proceso de diseño completo. A continuación incluimos una lista con los dominios, características y herramientas que nos parecen más interesantes teniendo en cuenta nuestro proyecto.

Comercio electrónico

- Contenido estándar: un catálogo de productos y un mecanismo de consulta a base de datos.
- En las tiendas electrónicas no hay dependiente, según el tipo de producto, puede ser complicado encontrar lo que necesitamos.
- El CBR proporciona mecanismos de recuperación aproximada.

Sistemas académicos no comerciales

Dificultades con las que se enfrentan:

- Representaciones sofisticadas con estructuras complicadas para las que no se suele disponer de herramientas de soporte
- La obtención y codificación de los casos es compleja. Normalmente se debe ocupar un experto. Esto reduce el tamaño de las bases de casos
- No existen técnicas generales de indexación y recuperación sobre casos con estructura compleja. Se desarrollan procedimientos ad-hoc.
- La adaptación resulta casi imprescindible, con el consiguiente problema de adquisición de conocimiento.
- No es fácil satisfacer a los usuarios. Los sistemas implantados suelen dejar la adaptación en sus manos.
- Las herramientas comerciales de CBR ofrecen poca ayuda. No existen metodologías aceptadas y bien definidas para este tipo de sistemas.

Planificación

- La herramienta Bolero planifica las pruebas a realizar para llegar a un diagnóstico.
- La herramienta CHEF planifica recetas de comida china.
- Coach planifica jugadas de fútbol americano.

Diseño

- Los casos tienen una representación compleja.
- La adaptación es difícil porque existe más de un diseño posible.
- A los expertos les resulta difícil prever el efecto de las modificaciones, con lo que se complica la obtención de reglas de adaptación
- La herramienta Julia diseña menús.
- La herramienta Archie ayuda en el diseño arquitectónico de alto nivel.
- La herramienta Clavier diseña la carga de un horno autoclave.

Diagnóstico

- La herramienta Protos diagnostica enfermedades del oído.

- La herramienta Casey diagnostica enfermedades cardiovasculares.
- La herramienta Cascade diagnostica problemas en sistemas VMS.
- La herramienta Caseline ayuda a los ingenieros de British Airways a diagnosticar y solucionar fallos en los Boeing 747.

Enseñanza

- La herramienta Spiel para el aprendizaje de habilidades sociales: negocios y diplomacia.
- La herramienta Hypo para la enseñanza de razonamiento legal.
- Institute for the Learning Science: case based teaching (Roger Schank).

¿Cuándo es apropiado usar un sistema CBR?

Las características del dominio que indican que una aproximación CBR es apropiada para resolver el problema son las siguientes:

- Hay registros de problemas similares existentes y resueltos
- Los casos pasados se ven como un conjunto que debe guardarse
- Recordar la experiencia pasada es útil
- Los especialistas hablan del dominio dando ejemplos
- La experiencia es al menos tan preciada como el conocimiento teórico
- Los problemas tienden a repetirse
- Los casos similares se resuelven con soluciones similares

Los sistemas de razonamiento basado en casos se usan normalmente cuando a los expertos les resulta difícil articular procesos para resolver los problemas. Esto es porque la adquisición del conocimiento es tremendamente difícil para estos dominios y es probable que se produzcan resultados imprecisos o incompletos. Cuando se usa un sistema sobre razonamiento de casos, la adquisición del conocimiento se puede limitar a establecer cómo caracterizar los casos.

El razonamiento basado en casos además se puede desarrollar de forma incremental y el mantenimiento de la biblioteca de casos es además relativamente sencillos y puede llevarse a cabo por un experto del dominio.

Tras describir todas estas características podemos ver que realmente un sistema CBR se ajusta muy bien a las necesidades de nuestro sistema experto. Por otra parte, una vez tomada esta decisión de diseño, nos restaba evaluar las diferentes herramientas CBR que se encontraban a nuestra disposición para elegir la que mejor se adaptase a nuestras necesidades. Éste es tema que tratamos en la siguiente sección.

1.6.2.JCOLIBRI

jCOLIBRI es una herramienta que desarrolla el grupo de investigación GAIA, de la Facultad de Informática de la Universidad Complutense de Madrid [6]. Esta

herramienta integra aplicaciones que combinan conocimiento específico de casos con modelos de conocimiento del dominio general.

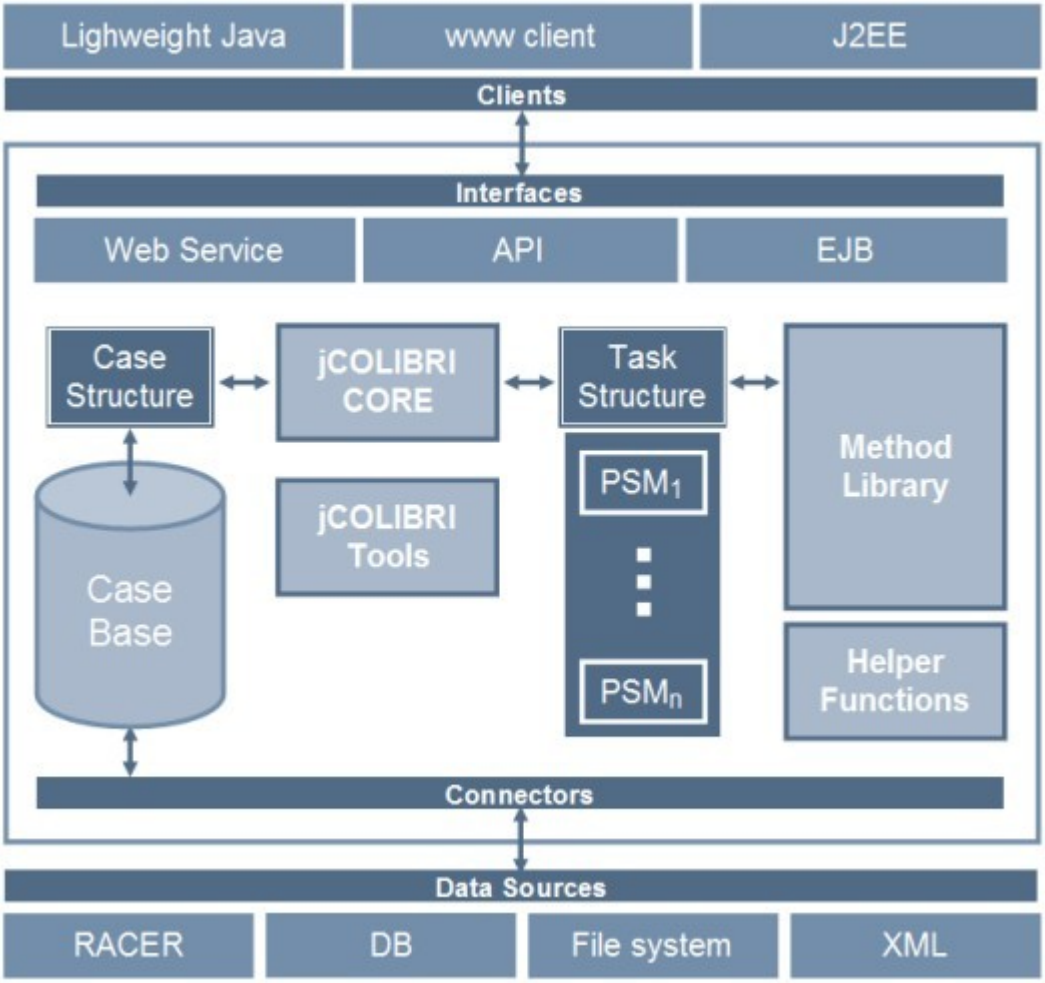
El dominio de esta herramienta se ajustaba a nuestras necesidades y creímos que implicaba una contribución interesante: probar y ampliar una herramienta desarrollada por la facultad, mientras que nos era útil para desarrollar la lógica de nuestro proyecto. Estos fueron los motivos para decantarnos por esta opción. Además, con el uso de jCOLIBRI probamos que satisface su carácter didáctico.

Al tomar esta decisión fuimos también conscientes por otra parte que basaríamos nuestro proyecto en la lógica de una herramienta de versión Beta que además es monousuario. Sin embargo, debido a que esta herramienta se ha ido y se irá mejorando durante el transcurso del tiempo, nos aseguramos que si bien en un primer momento el proyecto no tiene la funcionalidad más amplia posible o nos ha costado más trabajo llegar a ella, tiene las bases para tenerla en un futuro próximo. Además, con el uso de jCOLIBRI contribuimos también en la medida de nuestras posibilidades a mejorar y satisfacer sus fines educativos.

Ahora pasamos describir en más profundidad la primera versión que nos encontramos de jCOLIBRI y la versión posterior que se realizó entre otras cosas para dotar a la herramienta de una funcionalidad suficiente para satisfacer las necesidades de nuestro entrenador personal.

1.6.3.Primera Versión

COLIBRI significa (*Cases and Ontology Libraries Integration for Building Reasoning Infrastructures*) y sirve para el diseño intensivo del conocimiento en sistemas CBR. COLIBRI se basa en la adquisición del conocimiento de una biblioteca de ontologías independientes y en el uso de CBROnto, que es una ontología con la terminología común de CBR que guía la representación de casos y permite la descripción de un Método de Solución de Problemas (PSMs) flexible, genérico y reutilizable para resolver las típicas tareas de CBR [6].



jCOLIBRI es una evolución tecnológica de COLIBRI que incorpora en una arquitectura distribuida (como la que muestra la figura anterior) una máquina de DLs, un GUI para ensamblar un sistema CBR desde componentes reutilizables y un *framework* orientado a objetos en Java. El diseño del *framework* comprende una jerarquía de clases Java y un conjunto de ficheros XML organizados alrededor de los siguientes elementos [6]:

- **Tareas y Métodos.** Los ficheros XML describen las tareas soportadas por el *framework* así como los métodos para resolver esas tareas.
- **Base de Casos.** Define diferentes conectores para soportar varios tipos de persistencia de casos, desde un sistema de ficheros a una base de datos. Además, soporta un número de posibles organizaciones en memoria de la Base de Casos.
- **Casos.** Incluye varias interfaces y clases en el *framework* que proporcionan una representación abstracta de los casos que soportan cualquier tipo de la estructura del caso actual.
- **Método para la resolución de problemas.** Es el código que soporta los métodos incluidos en el *framework*. jCOLIBRI está diseñado para soportar fácilmente la construcción de diferentes tipos de sistemas CBR sacando provecho del paradigma de división de tareas y métodos descrito. Construir un sistema CBR es un proceso de configuración donde se seleccionan las tareas que el sistema debe satisfacer. Para cada tarea asigna el método que hará el trabajo.

Idealmente, el diseñador del sistema encuentra todas las tareas y métodos necesarios, así que solo tendría que programar la representación de los casos. Sin embargo, en una situación más realista hay un conjunto de métodos nuevos que se podrían necesitar y, menos probablemente, alguna tarea nueva. Como jCOLIBRI está diseñado como un *framework* extensible, los nuevos elementos se pueden integrar con la infraestructura disponible siempre y cuando sigan el diseño del *framework*.

Así, en nuestro caso hemos programado e incluido todos aquellos métodos que eran necesarios para el desarrollo de nuestro proyecto y que no estaban disponibles en la versión publicada.

Resaltamos que uno de los mayores problemas de los *frameworks* es aprender a usarlos. Para aliviar el esfuerzo de instanciar un *framework*, jCOLIBRI permite una herramienta de configuración semiautomática que guía el proceso de instanciación a través de una interfaz gráfica. Esta interfaz se crea dinámicamente para reflejar el contenido actual de la ontología y de las tareas y métodos que descansan sobre los ficheros XML que describen la restricción de las tareas y los métodos beneficiándose de las facilidades implementadas en Java. La configuración de un sistema CBR que usa esta interfaz consiste en el siguiente proceso:

- Definir la estructura de casos, la fuente de los casos y la organización de la base de casos.
- Mientras que el sistema no esté completo, hay que seleccionar una de las tareas que no tiene método asignado y seleccionar y configurar un método para cada una de esas tareas. Al comienzo el árbol de tareas sólo tiene un elemento, CBRTask, que se soluciona mediante un método de descomposición que resulta en tareas adicionales. Las restricciones de tarea/método son seguidas durante el proceso de configuración de tal forma que solo se aplican a métodos dados en el contexto que ofrece el diseñador del sistema.
- Una vez que el sistema está configurado, el código de configuración se genera corriendo el sistema CBR en un sistema disponible. La herramienta de configuración también proporciona una interfaz por defecto para correr el sistema CBR configurado, aunque en un ajuste real la interfaz específica de la aplicación debe desarrollarse, como ocurrirá en nuestro caso.

El *framework* de implementación evoluciona con los métodos nuevos que se incluyen. Ya hay una versión pública disponible en: http://jcolibri-cbr.sourceforge.net. jCOLIBRI intenta aplicar las mejores ideas de COLIBRI a los diferentes tipos de sistemas CBR con:

- Tipos diferentes de representación de casos (pares de valor-atributo simple, textos, representaciones orientadas a objetos, semi-estructurados…).
- Estructuras diferentes de organización de la base de casos (simple, jerárquica, basada en aprendizaje máquina, ...),
- Conjuntos distintos de tareas (como sistemas de sólo extracción)
- Métodos diferentes de resolución de tareas, desde métodos de conocimiento simples hasta complejos.

En resumen, podemos decir que el objetivo principal de jCOLIBRI es proporcionar un *framework* de referencia para el desarrollo de CBR que pueda crecer con las contribuciones de la comunidad. Esta referencia sirve para propósitos pedagógicos y como base de la implementación para prototipar sistemas CBR y comparando distintas aproximaciones CBR al problema dado.

1.6.4.Segunda Versión

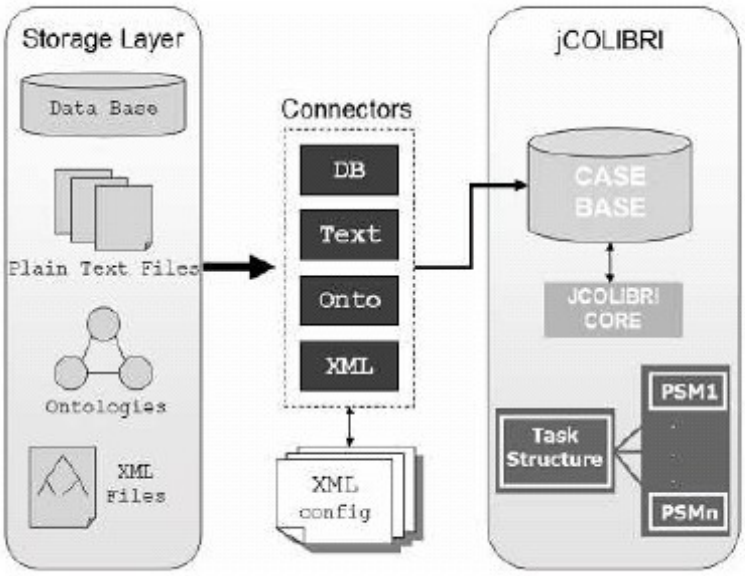
La segunda versión de jCOLIBRI incorpora la capacidad de razonamiento de la Lógica Descriptiva (DLs). Con esta expansión jCOLIBRI adquiere conocimiento de domino sobre ontologías que permite el desarrollo de aplicaciones CBR con conocimiento intensivo (*KI-CBR Knowledge Intensive CBR*). Esta aproximación resuelve el problema tradicional de adquisición del conocimiento en los sistemas CBR y mejora la funcionalidad del *framework* con nuevos PSMs para seleccionar y adaptar casos [7].

Concretamente esta ampliación nos proporciona las nuevas funcionalidades que enumeramos a continuación. La primera es que el tipo "conceptos" que incluye permite indicar a los desarrolladores el atributo que va a ser una instancia de la ontología. Este tipo de datos se puede almacenar también en cualquier clase de persistencia como una cadena de caracteres. Usa Jena para conectar el atributo con la instancia cargada por el razonador de DLs.

Esto permite que posteriormente podamos definir la adaptación de las soluciones de los distintos casos de una forma mucho más aproximada ya que tenemos en cuenta su estructura jerárquica. Diferentes cálculos relativos a la posición de los nodos en esta estructura permiten también el cómputo de similitud entre los distintos nodos o ejercicios que componen un entrenamiento.

La segunda mejora que añade esta nueva versión es la definición de un lenguaje general basado en reglas que usaremos para definir las premisas bajo las que se debe hacer la adaptación de la solución de un caso.

A continuación mostramos una figura con la arquitectura de la nueva versión de jCOLIBRI [7].



1.7.ONTOLOGÍA

1.7.1.Ontology Web Language (OWL)

OWL es el acrónimo de *Ontology Web Language* (Lenguaje Web para Ontologías). Este lenguaje esta diseñado para usarse con aplicaciones que necesitan procesar el contenido de la aplicación en vez de presentar simplemente la información a los usuarios. OWL permite mayor interpretabilidad del contenido de la Web que XML, RDF, y RDF Schema (RDF-S) proporcionando un vocabulario adicional, así como una semántica formal. OWL está compuesto de tres sublenguajes: OWL Lite, OWL DL, y OWL Full [8].

Introducción

OWL se usa cuando la información contenida en los documentos necesita procesarse en las aplicaciones como oposición a las situaciones en las que el contenido simplemente necesita presentarse a humanos. OWL puede usarse explícitamente para representar el significado de términos en vocabularios y las relaciones entre estos términos. La representación de estos términos y sus relaciones es lo que conocemos como ontología. OWL tiene más utilidades para expresar el significado y la semántica que XML, RDF, y RDF-S. Por tanto, OWL es capaz de ir más allá que estos lenguajes en su habilidad de representar la información de la máquina interpretando el contenido de la Web. OWL es una revisión del lenguaje de ontología Web DAML+OIL [9] que incorpora las lecciones aprendidas del diseño y la aplicación de DAML+OIL.

¿Por qué usar OWL?

La Web Semántica es una visión de futuro de la Web en la que se le da significado explícito a la información para facilitar que las máquinas hagan un procesamiento automático e integren la información disponible en la Web. La Web semántica se construye sobre la habilidad de los XML para definir esquemas de etiquetas adaptadas y una aproximación flexible para representar datos en RDF. El primer nivel sobre RDF requerido para la Web Semántica es una ontología que describa formalmente el significado y la terminología usada en los documentos de la Web. Sin embargo, el principal motivo para usar OWL en nuestro proyecto es porque es el lenguaje sobre el que se apoya la construcción de la ontología que creamos con la herramienta Protége. En la siguiente sección profundizaremos en este tema. Antes veremos algunos de los aspectos relacionados con OWL que consideramos básicos y que hemos estudiado para el desarrollo del proyecto.

- XML [10] proporciona una sintaxis para documentos estructurados, pero no impone restricciones semánticas en el significado de estos documentos.
- XML Schema [11] es un lenguaje para restringir la estructura de los documentos XML y extender el XML con tipos de datos.
- RDF [12] es un modelo de datos para objetos y relaciones entre ellos que proporciona una semántica simple para este modelo de datos. Estos modelos de datos se pueden representar en una sintaxis XML.
- RDF Schema [13] es un vocabulario para describir las propiedades y clases de los recursos RDF con semánticas para la generación de jerarquías de dichas propiedades y clases.
- OWL añade además vocabulario para describir las propiedades y clases: entre otras relaciones entre clases, cardinalidad, equidad, una capacidad de edición más rica, características de propiedad y clases enumeradas.

Los tres sublenguajes de OWL

OWL proporciona tres sublenguajes expresivos diseñados para usarse por comunidades específicas de implementadores y usuarios.

- OWL Lite es para los usuarios que necesitan una clasificación jerárquica y restricciones simples.
- OWL DL es para aquellos usuarios que quieren maximizar la expresividad mientras garantizan que todas las conclusiones son computables y decidibles.
- OWL Full está diseñado para usuarios que quieren maximizar la expresividad y la libertad sintáctica de RDF sin garantías de computación. Por ejemplo, en OWL Full una clase se puede tratar simultáneamente como una colección de individuos y como un individuo por si sólo, ya que permite aumentar la significado del vocabulario predefinido. Es improbable que cualquier razonamiento software soporte completamente el razonamiento para cada característica de OWL Full.

Cada uno de estos sublenguajes es una extensión de su predecesor más simple. Se soporta el siguiente conjunto de relaciones, pero no sus inversas.

- Cada ontología OWL Lite legal es una ontología OWL DL legal.
- Cada ontología OWL DL legal es una ontología OWL Full legal.
- Cada ontología OWL Lite conclusión es una ontología OWL DL conclusión.
- Cada ontología OWL DL conclusión es una ontología OWL Full conclusión.

Como desarrolladores de ontologías que adoptan OWL hemos considerado qué sublenguaje se adapta mejor a nuestras necesidades. La opción entre OWL Lite y OWL DL depende de si se quiere usar una extensión que proporcione construcciones más expresivas que las proporcionadas por OWL DL. La elección entre OWL DL y OWL Full depende principalmente de hasta qué punto se necesitan opciones de meta-modelado del RDF Schema (por ejemplo clases definidas o adjuntar propiedades a las clases). Cuando usamos OWL Full en comparación con OWL DL, el soporte del razonamiento es menos predecible porque la implementación de OWL Full completa no existe en la actualidad.

Decidimos usar una versión de OWL Full ya que se puede ver como una extensión de RDF, mientras que OWL Lite y OWL DL son vistas como extensiones de una vista restringida de RDF. Cada documento OWL (Lite, DL, Full) es un documento RDF y todos los documentos RDF son documentos OWL Full, pero sólo algunos documentos RDF pueden ser documentos OWL Lite o OWL DL legales. Por este motivo, hay que ser cuidadosos a la hora de migrar la ontología que definimos a documentos RDF de OWL.

Resumen de los lenguajes

Antes de explicar en detenimiento la versión de OWL Full describiremos las principales características de los otros sublenguajes.

Resumen de OWL Lite

Descripción de OWL Lite

Ahora veremos una descripción informal de las principales características del lenguaje OWL Lite. No profundizaremos en la sintaxis específica de estas características. Para un mayor conocimiento remitimos a la documentación en la que nos basamos [8] [14].

OWL Lite sólo usa algunas características del lenguaje OWL y tiene el uso de características más limitado que OWL DL o OWL Full. Por ejemplo, en OWL Lite las clases sólo se pueden definir en términos de las llamadas superclases que a su vez, no pueden ser expresiones arbitrarias, y de las cuales, sólo se pueden usar algunas restricciones de clase. La equivalencia entre las relaciones de clases, entre subclases y clases, sólo se permiten entre clases definidas y no entre expresiones de clase cualquiera. Del mismo modo, las restricciones en OWL Lite sólo usan clases definidas. OWL Lite tiene la cardinalidad limitada entre 0 y 1.

Características de OWL Lite RDF Schema

Se incluyen las siguientes características de OWL Lite relacionadas con RDF Schema.

- Class: [15] Una clase define un grupo de individuos que comparten el mismo conjunto de propiedades. Las clases se organizan en jerarquías usando subClassOf [16]. La clase sobre más general sobre la que se construye el resto se llama Thing [17]. Es la superclase define todos los individuos y clases de todas las clases OWL. También que no tiene instancias y que es una subclase de todas las clases OWL: es la clase definida Nothing [18].
- rdfs:subClassOf [19]: la jerarquía de clases se crea haciendo una más sentencias que indiquen que una clase es subclase de otra clase.
- rdf:Property [20]: las propiedades se pueden usar para hacer explícitas relaciones entre individuos o desde datos de individuos.
- rdfs:subPropertyOf [21]: la jerarquía de propiedades se puede crear haciendo una o más sentencias que indiquen qué propiedad es una subpropiedad de una o más propiedades.
- rdfs:domain [22]: el dominio de una propiedad limita los individuos a los que se puede aplicar una propiedad. Si una propiedad relaciona dos individuos y la propiedad tiene una clase como uno de sus dominios, entonces el individuo debe pertenecer a la clase.
- rdfs:range [23]: el rango de propiedades limita los individuos que la propiedad puede tener como su valor. Si una propiedad relaciona un individuo con otro individuo y la propiedad tiene una clase como su rango entonces el otro individuo debe pertenecer al rango de la clase.
- Individual [24]: los individuos son instancias de clases, y propiedades que se pueden usar para relacionar un individuo con otro.

Restricciones de propiedades en OWL Lite

OWL Lite permite restricciones que se pueden poner como propiedades y pueden usarse a su vez como instancias de una clase. Estos tipos (y también las restricciones de cardinalidad) se usan en el contexto de un owl:Restriction [24]. El elemento owl:onProperty [25] indica la restricción de propiedad. Las dos restricciones anteriores limitan como pueden usarse los valores mientras las siguientes restricciones limitan cuantos valores pueden usarse.

- allValuesFrom [26]: esta restricción es una propiedad con respecto a una clase. Esto significa que esta propiedad en esta clase en particular tiene una restricción asociada en el rango local. Por tanto, si una instancia de esta clase está relacionada con la propiedad de un segundo individuo, entonces el segundo individuo se puede inferir para ser instancia de la restricción del rango local de la clase.
- someValuesFrom [25]: esta restricción se define en una propiedad con respecto a una clase. Una clase particular tiene una restricción en una propiedad que al menos un valor para esa propiedad es de un determinado tipo.

Cardinalidad restringida en OWL Lite

OWL Lite incluye una forma limitada de restricciones de cardinalidad. Las restricciones de cardinalidad de OWL (y OWL Lite) se refieren a restricciones locales porque son definidas en propiedades respecto a una clase particular. Así, las restricciones limitan la cardinalidad de las propiedades sobre instancias de esa clase. Las restricciones de cardinalidad de OWL Lite están limitadas porque sólo permiten sentencias acerca de valores de cardinalidad de 0 ó 1.

- minCardinality [27]: la cardinalidad se especifica en una propiedad con respecto a una clase particular. Si se especifica cardinalidad mínima de 1 en una propiedad con respecto a una clase, entonces cualquier instancia de esa clase se podrá relacionar al menos con un individuo por esta propiedad. Esta restricción es otra forma de decir que la propiedad requiere que se tenga un valor para todas las instancias de la clase.
- maxCardinality [28]: la cardinalidad se especifica en una propiedad con respecto a una clase particular. Si la cardinalidad máxima de 1 se especifica en una propiedad respecto a una clase, entonces cualquier instancia de esa clase puede relacionarse como mucho con un individuo por esa propiedad. La restricción de cardinalidad máxima a veces se llama también propiedad funcional o de unicidad
- cardinality [29]: la cardinalidad se proporciona como una comodidad cuando es útil para especificar que una propiedad en una clase tiene ambas: cardinalidad mínima 0 y cardinalidad máxima 0 o cardinalidad mínima 1 y cardinalidad máxima 1.

Tipos de datos OWL

OWL usa mecanismos de RDF para valores de datos. Para información más detallada de como construir tipos de datos OWL recomendamos la siguiente referencia [30].

Cabecera de información OWL Lite

OWL Lite permite también adjuntar información a las ontologías. Para ver más ejemplos sobre esto remitimos a la siguiente referencia [13], [14].

Anotación de propiedades OWL Lite

OWL Lite permite hacer anotaciones en las clases, propiedades, individuos y cabeceras de ontologías. El uso de estas notaciones es objeto de determinadas restricciones. La siguiente referencia proporciona más detalles sobre las anotaciones [31].

Descripción incremental del lenguaje OWL DL y OWL Full

OWL DL y OWL Full usan el mismo vocabulario aunque OWL DL está sometido a más restricciones. Aproximadamente, OWL DL requiere tipo de separación, es decir, una clase puede ser no sólo un individuo o una propiedad, sino también un individuo o

una clase. Esto implica que las restricciones no se puedan aplicar al lenguaje de elemento de OWL mismo (algo que sí está permitido en OWL Full). Además, OWL DL requiere que las propiedades sean bien ObjectProperties o DatatypeProperties. Las últimas son relaciones entre instancias de clases, literales RDF y esquemas de tipos de datos XML, mientras que las primeras son relaciones entre instancias de dos clases. Para más información sobre este tema recomendamos la referencia [32] que explica claramente las distinciones y limitaciones.

Ahora pasamos a describir las construcciones que extienden el lenguaje de OWL Lite en OWL DL y OWL Full.

1. oneOf: [33] (clases enumeradas): estas clases se describen enumerando los individuos que las componen. Los miembros de la clase son exactamente el conjunto de individuos enumerados.
2. hasValue: [34] (valores de propiedades): una propiedad puede requerir tener un determinado individuo como valor.
3. disjointWith: [35] las clases se pueden definir como disjuntas. Por ejemplo, hombre, mujer.
4. unionOf, complementOf, intersectionOf [36] (combinaciones booleanas): OWL DL y OWL Full permiten combinaciones arbitrarias de clases y restricciones booleanas como la unión, el complemento y la intersección.
5. minCardinality, maxCardinality, cardinality [37] (cardinalidad completa): mientras que en OWL Lite la cardinalidad está restringida a ser al menos, como mucho ó exactamente 1 ó 0, OWL permite definir las especificación de la cardinalidad para enteros arbitrarios no negativos.
6. complex classes: en muchos constructores OWL Lite restringe la sintaxis a nombres de clases simples, por ejemplo en las sentencias subClassOf o equivalentClass. OWL Full extiende esta restricción para permitir restricciones arbitrarias de clases más complejas que consisten en clases enumeradas, restricción de propiedades y combinaciones boolenas. Además, OWL Full, permite usar clases como instancias.

En resumen, diremos que usamos OWL ya que es el estándar más desarrollado para la creación de ontologías.

1.7.2.Protégé

Esta sección describe la herramienta elegida para la creación de la ontología. Al igual que en la asignatura Ingeniería de Sistemas Basados en el Conocimiento nos hemos basado en Protégé que es una plataforma gratuita de código abierto que proporciona a una comunidad de usuarios creciente un conjunto de herramientas para la construcción de modelos del dominio y aplicaciones basadas en el conocimiento con ontologías.

El núcleo de esta herramienta implementa un amplio conjunto de estructuras para el modelado del conocimiento y acciones que soporten la creación, visualización y manipulación de ontologías en varios formatos de representaciones. Además, permite la

personalización de Protégé para proporcionar un soporte de dominio amigable al crear modelos de conocimiento y entrada de datos. Por otra parte, también se puede extender con una arquitectura plug-in y API basado en Java para construir herramientas y aplicaciones basadas en el conocimiento.

Ahora pasaremos a explicar qué es exactamente una ontología. Podemos verla como un conjunto de conceptos y relaciones que son importantes en un dominio concreto que proporciona el vocabulario para que ese dominio sea también una especificación computable del significado y términos usados en el vocabulario. El rango de las ontologías va desde taxonomías, clasificaciones y esquemas de bases de datos, hasta teorías completamente axiomatizadas. Anecdóticamente nos gustaría resaltar que en los últimos años, las ontologías se han adoptado en muchos negocios y comunidades científicas como la forma de compartir, reutilizar y procesar conocimiento del dominio. Por este motivo, las ontologías son ahora el núcleo de muchas aplicaciones como portales de conocimiento científico, gestión de información e integración de sistemas, comercio electrónico y servicios Web semánticos.

Véamos ahora cuáles son las principales vías para modelar ontologías con la plataforma de Protégé:

- Protégé-Frames Editor [38] permite a los usuarios construir y poblar la ontología según el protocolo Open Knowledge Base Connectivity (OKBC) [39]. En este modelo, una ontología está compuesta de un conjunto de clases organizadas en una jerarquía para representar un dominio de conceptos salientes, un conjuntos de atributos asociados a clases que describen sus propiedades y relaciones y un conjunto de instancias de esas clases – ejemplares individuales de conceptos que tienen valores específicos para unas propiedades.
- Protégé-OWL Editor permite a los usuarios construir ontologías para la Web Semántica, en particular en el W3C's Web Ontology Language (OWL) [8]. "Una ontología OWL puede incluir descripciones de clases, propiedades y sus instancias. Dada dicha ontología, la semántica formal de OWL especifica como derivar sus consecuencias lógicas, por ejemplo, sino están presentes explícitamente en la ontología, pero la semántica lleva consigo. Estos hechos implícitos se pueden basar en un documento sencillo o múltiples documentos distribuidos que se combinan definiendo los mecanismos OWL" [14].

El editor Protégé-Frames proporciona una interfaz de usuario completa y un conocimiento de servidor que soporta usuarios que construyen y almacenan ontologías de dominio basado en estructura, personalización de formularios de entrada de datos y ejemplos de entrada de datos. Protégé-Frames implementa un modelo de conocimiento que es compatible con el protocolo OKBC que mencionamos anteriormente.

Características que incluye Protégé-Frames:

- Una amplia gama de elementos de interfaz de usuario que se pueden personalizar para modelar el conocimiento y meter datos en formularios de dominio amigables.
- Una arquitectura plug-in que se puede extender con elementos personalizados, como componentes gráficos como gráficos y tablas; elementos multimedia como sonido, imágenes y vídeo; varios formatos de almacenamiento: RDF, XML, HTML... y herramientas adicionales de soporte.
- El API de Java hace que sea posible tener plug-ins y otras aplicaciones que accedan, usen y muestren ontologías creadas con Protégé-Frames.

Para un conocimiento más profundo sobre este tema recomendamos las siguientes referencias: [41] [42] [43].

El editor Protégé-OWL editor permite al usuario:

- Cargar y salvar ontologías OWL y RDF.
- Editar y visualizar clases, propiedades y reglas.
- Definir clases de características lógicas como expresiones OWL.
- Ejecutar razonadores como la descripción lógica de clasificadores. En la siguiente sección explicaremos los razonadores con más detalle.
- Editar individuos OWL para marcado de Web Semántica.

La arquitectura flexible de Protégé-OWL facilita la extensión de la herramienta. Protégé-OWL está íntimamente integrado con JENA y tiene un API de código abierto en Java para el desarrollo y personalización de los componentes de interfaz de usuario o servicios arbitrarios de Web Semántica.

Para un mayor conocimiento de esta herramienta recomendamos las siguientes referencias: [44] [45] [46].

1.7.3.Racer y Jena

1.7.4.Racer y OWL

Una vez que hemos editado la ontología necesitamos una herramienta que sea capaz de comprobar la corrección del sistema y nos sea útil para su análisis. En nuestro caso elegimos Racer, ya que es la herramienta que se nos proporcionó en la facultad y era la más conveniente del mercado. A continuación, introducimos Racer explicando cuales son los hechos principales que soportan esta afirmación y las características de razonamiento en las que se basa.

RACER o RacerPro es el nombre que recibe el primer razonador OWL del Mercado. Apareció en el 2002 y se ha mejorado desde entonces. Mientras que otros razonadores han intentado lograr una velocidad comparable, Racer todavía es unos de los sistemas disponibles de razonamiento para OWL más rápido [47] [48].

Podemos decir que Racer se considera uno de los principales agentes de razonamiento para la Web semántica. Actualmente, soporta una amplia gama de servicios de inferencia de ontologías que se especifican en OWL. Sus principales funcionalidades son las siguientes: por varios clientes con editores, herramientas de desarrollo y visualización de ontologías. También, es el primer prototipo basado en la Web para la exploración y análisis de ontologías OWL.

Como hemos explicado en secciones anteriores la Web semántica define retos importantes para la representación del conocimiento y la inferencia de sistemas. Recientemente se han propuesto varios estándares para la representación de lenguajes. Unos de estos estándares son RDF y RDF Schema. La expresividad de estos lenguajes es limitada y como vimos, OWL presenta una expresividad mucho más rica. Además también proporciona medios para tratar con tipos de datos conocidos desde los lenguajes de programación.

Después de este pequeño resumen podemos empezar a comprender Racer más en profundidad. Como ya hemos anticipado Racer es un razonador para OWL. Principalmente, lee bases de conocimiento OWL desde archivos locales o desde servidores remotos. Además, otros programas cliente que necesiten servicios de inferencia pueden comunicarse con el servidor Racer vía protocolos TCP. Racer también proporciona una interfaz basada en TCP para pedir y recibir instrucciones y consultas. La ventaja principal es que los usuarios pueden escribir consultas que se envían directamente al servidor Racer. No obstante, la interfaz Racer TCP también pudiese acceder fácilmente desde aplicaciones Java o C++. De hecho, hay un API disponible para ambos lenguajes.

Una de las características principales de Racer es que lee directamente documentos OWL y los representa como bases de conocimiento en lógica descriptiva.

En términos de lógica descriptiva una tupla está formada por un T-box y un A-box que se refieren a la base de conocimiento. Un individuo es un objeto específico. Por ejemplo, para expresar el hecho que todos los humanos vienen de un solo individuo Adam, hay que referirse a un individuo en un concepto (y un T-box). Sólo parte de la expresividad de los individuos mencionados en los conceptos se puede capturar con A-boxes. Sin embargo, en la práctica existe una aproximación directa ya que las ontologías se puede generar desde un documento OWL

A continuación describimos la selección de consultas soportadas.

- Consistencia de conceptos: ¿se describe el conjunto de objetos como un conjunto de conceptos vacío?

- Clasificación de conceptos: ¿hay un subconjunto de relaciones entre el conjunto de objetos que se puedan describir por dos conceptos?

- Encuentra todas las inconsistencias de conceptos mencionadas en T-box. Algunas de estas inconsistencias provienen de errores de modelado.

- Determina los padres e hijos de un concepto. Los padres de un concepto son los nombres de conceptos más específicos denotados en un T-box que clasifican el concepto. Los hijos de un concepto son los conceptos más generales que contiene el T-box que clasifica el concepto.

Cuando se necesita un concepto como argumento de una consulta, no sólo son posibles los nombres predefinidos sino también un A-box. Entre otros, los siguientes tipos de consultas A-box son posibles:

- Comprueba la consistencia de un A-box: ¿las restricciones dadas en un A-box y un T-box se contradicen?

- Prueba de instancias: ¿es el objeto para el cual un individuo es un miembro del conjunto de objetos descritos por un cierto concepto de la consulta? Entonces el individuo es una instancia del concepto de la consulta.

- Selección de instancias: encuentra todos los individuos desde un A-box tales que los objetos que simbolizan prueban ser miembros de un conjunto de objetos descritos por un cierto concepto.

- Computación de diferentes tipos de un individuo. Encuentra el concepto más específico desde una T-box de la cual el individuo es instancia.

- Computa si se satisfacen los requisitos que hacen que se referencie a un individuo.

Estas nociones sobre lógica descriptiva hacen que se pueda demostrar como muchos servicios de inferencia resuelven problemas con bases de conocimiento OWL.

También es interesante conocer las técnicas de optimización para la consulta de ontologías respecto T-boxes y A-boxes y valores concretos que se han desarrollado, implementado e investigado con el sistema Racer. Uno de los objetivos de diseño de Racer es seleccionar automáticamente el estado de la técnica de optimización que es aplicable a la entrada actual.

Por otra parte, la persistencia se da de un modo similar a los sistemas de base de datos. Para resolver consultas a T-boxes y A-boxes Racer computa y usa estructuras de datos complicadas. Estas estructuras son internas y pueden ser T-boxes y A-boxes y se pueden procesan para una consulta que se salva de disco para tener acceso rápido y más tarde se reusar si se reinicia el servidor Racer [47].

Finalmente y como resumen, diremos que Racer es un agente de razonamiento para OWL que proporciona servicios para la exploración y análisis de ontologías [47].

1.7.5.Jena

Al igual que necesitábamos Racer para analizar la ontología necesitamos una herramienta que enlace el uso de ontologías con la herramienta (jCOLIBRI) que desarrolla el núcleo principal de la aplicación que forma nuestro proyecto.

Concretamente Jena, que es una herramienta para desarrollar aplicaciones en la Web semántica. El único prerrequisito para usar Jena es que el código de la aplicación sea en Java. Por otra parte los principales objetivos para usar Jena son los siguientes [49]:

- Cargar archivos de ontologías en Java,
- Escribir un gráfico de RDF como un archivo RDF/XML,
- Navegar en un gráfico RDF.

Podemos usar Jena gracias a la posibilidad de convertir el archivo OWL Full de nuestra ontología en archivo RDF. En general OWL nos permite expresar todo lo que se puede expresar como un RDFS (por tanto RDF) y mucho más [50].

1.8.BASE DE DATOS

1.8.1.MySQL

Desde el comienzo de nuestro proyecto era evidente que íbamos a necesitar una base de datos (BD). Por ello, uno de los puntos de nuestra fase de investigación consistió en determinar qué sistema de gestión de bases datos era el más adecuado para nuestros propósitos.

1.8.2.La elección del sistema de gestión de la base de datos

Los sistemas de gestión de bases de datos que conocíamos eran ACCESS, ORACLE, SQL Server y MySQL. De todos ellos, solamente MySQL está desarrollado bajo la filosofía de código abierto. Este último aspecto nos pareció muy importante, pero no por ello dejamos de estudiar las demás posibilidades.

La primera opción que descartamos fue ACCESS debido a que es un sistema orientado a usuarios y no a desarrolladores, por lo que no posee el alcance ni la flexibilidad de las otras opciones.

Además Access, ORACLE y SQL Server tienen el inconveniente de ser de pago y, al menos los dos últimos, requieren mucho espacio de memoria. Todo ello nos condicionaba y limitaba a la disponibilidad de los laboratorios de la facultad con licencia para la herramienta.

Siguiendo nuestra preferencia inicial, MySQL acabó siendo la elegida para nuestro sistema. Pero antes de tomar la decisión recopilamos información para asegurarnos de que el software realmente cubriría las necesidades del proyecto.

1.8.3.Más información sobre MySQL

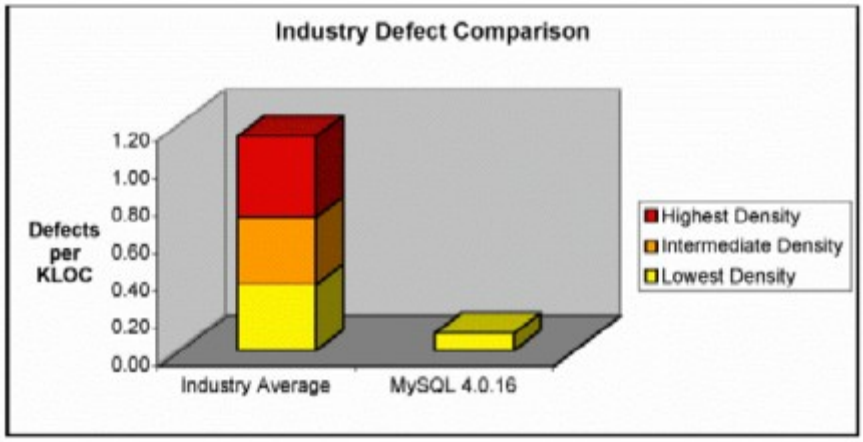
En la página oficial de MySQL [51] obtuvimos los siguientes datos.

La calidad del código

A study by Reasoning, the leading provider of automated software inspection (ASI) services, concluded that the code quality of MySQL ranks higher than commercial equivalents.

Specifically, Reasoning found that:

- *MySQL code quality was 6x better than that of comparable proprietary code*
- *MySQL benefits from the large communities of programmers who "battle test" the code*
- *MySQL benefits from users who not only report bugs, but track down their root cause and fix them.*



MySQL has a defect density that is about six times lower than comparable proprietary projects.

A follow up study by Coverity found that MySQL had an average of one bug in every 4,000 lines of code --results that are more than four times better than is typical with commercial software.

MySQL has used Coverity products to detect and fix:

- *Crash Causing Defects*
- *Performance Degradation*
- *Security Vulnerabilities*

MySQL Network certified software is tested on more than ten different platforms using tools from Coverity and Klocwork.

10 razones para instalar MySQL

Además, la página listaba 10 razones para instalar MySQL, que nos parecieron muy convincentes.

1. Scalability and Flexibility
The MySQL database server provides the ultimate in scalability, sporting the capacity to handle deeply embedded applications with a footprint of only 1MB to running massive data warehouses holding terabytes of information. Platform flexibility is a stalwart feature of MySQL with all flavors of Linux, UNIX, and Windows being

supported. And, of course, the open source nature of MySQL allows complete customization for those wanting to add unique requirements to the database server.

2. High Performance

A unique storage-engine architecture allows database professionals to configure the MySQL database server specifically for particular applications, with the end result being amazing performance results. […]

3. High Availability

Rock-solid reliability and constant availability are hallmarks of MySQL, with customers relying on MySQL to guarantee around-the-clock uptime. MySQL offers a variety of high-availability options from high-speed master/slave replication configurations, to specialized Cluster servers offering instant failover, to third party vendors offering unique high-availability solutions for the MySQL database server.

4. Robust Transactional Support

MySQL offers one of the most powerful transactional database engines on the market. Features include complete ACID (atomic, consistent, isolated, durable) transaction support, unlimited row-level locking, distributed transaction capability, and multi-version transaction support where readers never block writers and vice-versa. Full data integrity is also assured through server-enforced referential integrity, specialized transaction isolation levels, and instant deadlock detection.

5. Web and Data Warehouse Strengths

MySQL is the de-facto standard for high-traffic web sites because of its high-performance query engine, tremendously fast data insert capability, and strong support for specialized web functions like fast full text searches. […]Other features like main memory tables, B-tree and hash indexes, and compressed archive tables that reduce storage requirements by up to eighty-percent make MySQL a strong standout for both web and business intelligence applications.

6. Strong Data Protection

Because guarding the data assets of corporations is the number one job of database professionals, MySQL offers exceptional security features that ensure absolute data protection. In terms of database authentication, MySQL provides powerful mechanisms for ensuring only authorized users have entry to the database server, with the ability to block users down to the client machine level being possible. SSH and SSL support are also provided to ensure safe and secure connections. A granular object privilege framework is present so that users only see the data they should, and powerful data encryption and decryption functions ensure that sensitive data is protected from unauthorized viewing. Finally, backup and recovery utilities provided through MySQL and third party software vendors allow for complete logical and physical backup as well as full and point-in-time recovery.

7. Comprehensive Application Development

One of the reasons MySQL is the world's most popular open source database is that it provides comprehensive support for every application development need. [...] MySQL also provides connectors and drivers (ODBC, JDBC, etc.) that allow all forms of applications to make use of MySQL as a preferred data management server. It doesn't matter if it's PHP, Perl, Java, Visual Basic, or .NET, MySQL offers application developers everything they need to be successful in building database-driven information systems.

8. Management Ease

MySQL offers exceptional quick-start capability with the average time from software download to installation completion being less than fifteen minutes. This rule holds true whether the platform is Microsoft Windows, Linux, Macintosh, or UNIX. Once installed, self-management features like automatic space expansion, auto-restart, and dynamic configuration changes take much of the burden off already overworked database administrators. [...]

9. Open Source Freedom and 24 x 7 Support

[...] worries can be put to rest with MySQL as complete around-the-clock support as well as indemnification is available through MySQL Network. MySQL is not a typical open source project as all the software is owned and supported by MySQL AB, and because of this, a unique cost and support model are available that provides a unique combination of open source freedom and trusted software with support.

10. Lowest Total Cost of Ownership

[...] Accomplished through the use of the MySQL database server and scale-out architectures that utilize low-cost commodity hardware, corporations are finding that they can achieve amazing levels of scalability and performance, all at a cost that is far less than those offered by proprietary and scale-up software vendors. In addition, the reliability and easy maintainability of MySQL means that database administrators don't waste time troubleshooting performance or downtime issues, but instead can concentrate on making a positive impact on higher level tasks that involve the business side of data.

Centros y empresas que utilizan MySQL

Lógicamente esta información es muy favorable a MySQL, ya que la proporciona la propia empresa que se encarga de difundir el software. Pero hubo otros datos que nos convencieron de la calidad de este producto. El más importante es el número y tipo de empresa que hacen uso de MySQL. Entre ellas se encuentran compañías, centros y grupos de investigación tan destacadas como: Bayer, Colgate, Ensembl Genome Browser, The Institute for Genomic Research, AIRBUS/EADS, French Ministry of Defense, Los Alamos National Laboratory, Google, Lycos Europe, Yahoo!, MIT Lincoln Lab, University of California, Berkeley, Chicago Mercantile Exchange, CLASS / Credit Lyonnais, HypoVereinsbank, Lloyds TSB Bank, City of New York, Deutsche Post , NASA, State of Michigan, State of Minnesota, State of New York, State of Rhode

Island, U.S. Census Bureau, United Nations FAO, UNICEF, Braun, DaimlerChrysler, Epson, Toyota France, Yamaha, Associated Press, BBC Technology, Bloomberg L.P., Chicago Sun-Times, Wikipedia, Fedora Project, OpenOffice, Apple, BMC, Dell, Hewlett-Packard, Intel, McAfee, Motorola, Sony Deutschland GmbH, Sun Microsystems, Texas Instruments, Xerox Research Centre Europe, Alcatel, AT&T Wireless, Ericsson, France Telecom, Nokia , Siemens, Continental Airlines, Lufthansa Systems, etcetera.

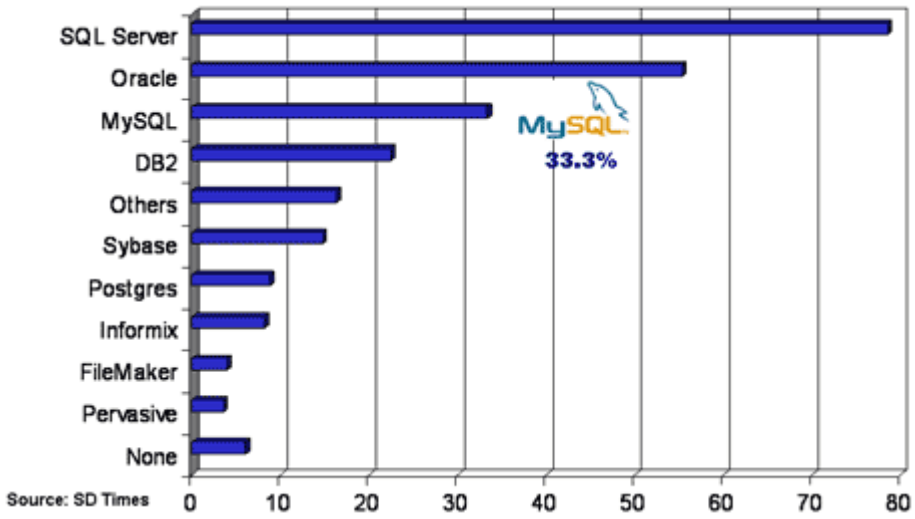
La cantidad y reputación de estas empresas nos confirmó que se trataba de un sistema que cumplía ciertos niveles de calidad.

La cuota de mercado

Además encontramos información sobre el número de sistemas que trabajan con MySQL.

MySQL is not only the world's most popular open source database, it's also the fastest growing database in the industry. With more than 8 million active installations and 50,000 downloads per day, MySQL continues to be the choice for a broad range of developers, database administrators and IT managers who want a high performance database that is reliable and easy to use.

In an article "Relational Databases Rule the Roost" published in SD Times in July 2004, MySQL was identified as the number 3 "Top Deployed Database" in a survey of 934 readers. MySQL was more broadly deployed than DB2, Informix, Postgres or Sybase. Respondents cited familiarity with the database, reputation of the database vendor and lowest development costs as the top 3 factors that led to their database choice.



MySQL is the #3 database among those polled

Las implicaciones de la licencia GPL

La licencia GPL de MySQL obliga a distribuir cualquier producto derivado bajo esa misma licencia. Si un desarrollador desea incorporar MySQL en su producto pero no desea distribuirlo bajo licencia GPL, puede adquirir la licencia comercial de MySQL que le permite hacer justamente eso [52].

1.8.4.SQLYOG

La necesidad de un interfaz

Para facilitar el trabajo con MySQL decidimos utilizar un interfaz que resultara más amigable que el modo consola de MySQL. Descubrimos SQLYog, que dispone de una versión gratuita en su página Web [52]. Como no teníamos experiencia previa con un interfaz para bases de datos y nos gustó el que habíamos descargado, decidimos utilizarlo para el proyecto.

En su página Web se puede leer la siguiente información sobre la compañía.

Webyog Softworks Pvt. Ltd., a privately held company based in Bangalore, India, creates innovative data management tools for thousands of customers in 84 countries, ranging from enterprises to small businesses and home users.

Founded in 2001, the company provides best-of-breed data management tools for managing popular open-source databases. Webyog consistently receives top ratings and recommendations by respected third-party media and product reviewers.

More than 4500 customers worldwide have selected Webyog to help them smartly manage their databases. Webyog is a MySQL Network Gold Certified Partner and works closely with MySQL AB to deliver the next generation of productivity tools for MySQL.

1.9.DISEÑO

En este apartado queremos explicar todo lo referente al diseño. Esto incluye tanto la estructura de páginas Web como nuestro trabajo con imágenes en 2 y 3 dimensiones.

Nuestro objetivo era crear un entorno agradable e intuitivo que sirviera para obtener y presentar información al usuario. Aunque la función del interfaz es principalmente estética, su importancia no debe ser despreciada ya que nos movemos en un mundo gobernado por las imágenes. Consideramos que un buen diseño es la carta de presentación de un sistema, especialmente si se trata de retener la atención de las personas que navegan por Internet.

La versión final de nuestra Web cumple su función de interfaz con el resto del sistema, ocultando al usuario la lógica e implementación, pero haciendo uso de ellas para adaptarse dinámicamente a las peticiones y proporcionar un servicio individualizado. Aunque los gustos y criterios estéticos pueden ser diversos, consideramos que hemos cumplido con nuestro objetivo.

1.9.1.INTERFAZ GRÁFICA

A la hora de desarrollar una Web es muy importante mantener una clara distinción entre lo que se quiere mostrar y cómo se va a mostrar. Esta separación entre contenido y aspecto es lo que hace que las Webs sean mantenibles y exista un mismo criterio estético en todas las páginas, haciendo que el interfaz sea fácil de ampliar. Para cumplir este objetivo hemos seguido los pasos relatados a continuación.

1.9.2.PLANTILLA WEB: CONTENIDO, CONDICIONES Y LIMITACIONES

Inicialmente contemplamos tanto la posibilidad de crear la Web desde cero como la de utilizar alguna de las plantillas de libre distribución que hay en la red. Por ello estuvimos visitando distintos portales que ofrecen este tipo de esquemas, entre los que encontramos uno que nos gustó mucho.

Finalmente optamos por una vía intermedia que consistía en utilizar una plantilla, pero adaptándola a nuestras necesidades. Aunque esto suponía una gran cantidad de modificaciones, nos pareció la mejor decisión por varias razones. Primordialmente porque la reutilización de código, aunque se trate de un lenguaje de marcado, nos parece lo más sensato para ahorrar tiempo y trabajo inútil. Además, nuestra experiencia en diseño Web nos ha demostrado que si hubiéramos dedicado el mismo tiempo a crear la Web desde la nada no hubiéramos obtenido una Web tan elaborada.

La plantilla

Obtuvimos la plantilla de un portal llamado AUHWeb [53]. Allí encontramos bastantes plantillas que nos gustaron, aunque finalmente nos decantamos por la número 80. Los detalles indicados en la página son los siguientes:

La carpeta descargada contenía un fichero HTML de ejemplo, una hoja de estilo CSS, una carpeta con las imágenes que se ven por pantalla, una carpeta con la imagen de encabezado en blanco y un fichero TXT con información adicional.

Las condiciones

Al margen de la publicidad, en el fichero de texto se detallaba lo siguiente:

Template 80

FreeTemplates4all.com - Free html web templates.
Part of JoshsfsWeb.
(c) copyright 2005

Thank you for downloading this template!
Read the index.htm for help on editing the look, etc. of this template. For more info email us at templates@freetemplates4all.com

Terms
1) You can edit this template as much or as little as you like.
2) You must NOT pass this work of as your own
3) You must not re-distribute this template without proper permission.
4) This template is free to use as long as a link if left back to www.freetemplates4all.com.

Thanks!
www.freetemplates4all.com

Hemos cumplido estrictamente todas las condiciones, incluyendo un enlace a su página desde nuestra sección "Enlaces de interés".¡Y, por supuesto, lo hemos cambiado mucho! Estos cambios no sólo se deben a criterios estéticos, sino en gran parte a las limitaciones del material de partida. A continuación explicaremos las modificaciones que hemos realizado y las razones por las que tuvimos que hacerlas.

Las imágenes de la plantilla

Como ya mencionamos antes, la plantilla incluía dos carpetas con imágenes que pueden ser utilizadas directamente, pero lógicamente sólo si los encabezados que se necesitan son los proporcionados y se quieren en inglés. Como este no era nuestro caso, fue necesario generar todas las imágenes que contienen texto a partir de la única imagen en blanco que proporcionaba la plantilla. El proceso exacto está detallado en la sección de "Diseño en entorno 2D", aunque hemos querido mencionar el problema aquí, puesto que se deriva de la plantilla.

La hoja de estilos

La hoja de estilos de la que partimos especificaba el formato del texto de salida una propiedad de las filas de las tablas y las propiedades de los enlaces. Esto era bastante limitado para lo que nos disponíamos a hacer, de modo que tuvimos que ampliar la hoja de estilos.

La estructura de la plantilla

El documento HTML consistía en una serie de tablas anidadas que contenían enlaces e imágenes de fondo y cabecera. Esta organización permite borrar e insertar fácilmente nuevos elementos, que es lo que hicimos a continuación.

1.9.3.NUESTRO DISEÑO

Nuestro esfuerzo de diseño se puede subdividir en distintas secciones. Por una parte hubo que cambiar la estructura de la página en sí, así como generar varios tipos de página distintos y enlazarlos correctamente. Además tuvimos que ampliar la hoja de estilo para crear un estilo personalizado y coherente para toda nuestra Web. Todo esto está explicado en el apartado "diseño de la Web".

Por otra parte está el tratamiento de imágenes, que a su vez se puede subdividir en dos categorías. Una es la de las imágenes estructurales, que componen el fondo de la página Web. Su función es completamente estética y gracias a ellas la Web parece una colección de cajitas azules de bordes redondeados que contienen la información. Este tipo de imágenes pertenecen de forma estática a la Web.

La segunda categoría de imágenes está relacionada con la información que queremos mostrar y se toma de forma dinámica de la base de datos, según las necesidades de cada usuario.

La diferencia entre los tipos de imágenes es más bien conceptual, ya que en el fondo todas son ficheros ".jpeg" o "gif". Pero el proceso de desarrollo que hemos seguido y las técnicas que hemos utilizado para crearlas difieren, de modo que hemos considerado conveniente explicarlas en dos apartados distintos. "Diseño en entorno 2D" para las imágenes estructurales y "Diseño en entorno 3D" para las imágenes deportivas.

1.9.4.DISEÑO DE LA WEB

La herramienta

Existen multitud de herramientas para la edición y creación de páginas Web. Nosotras hemos utilizado Macromedia Dreamweaver, ahora propiedad de Adobe. Esta decisión se debe simplemente a que ya teníamos experiencia con el uso de esta herramienta.

Modificaciones sobre la hoja de estilos (CSS) original

Una de las cosas que la hoja de estilo original no contenía, eran indicaciones para los distintos tipos de encabezados ("headers") que el lenguaje de marcado permite. Creamos cuatro niveles con distintos formatos (colores en diversos tonos de azul o negro, peso, posición, tamaño, etc.)

Además hubo que definir tres tipos distintos de enlaces. Todos ellos tienen un color azul extraído del encabezado (t1 el #0000FF t2 y t3 el #3399FF) y pasan a un color de contraste anaranjado o negro cuando el usuario pasa el cursor sobre ellos (t1 y t2 a #FF3300 y t3 al negro). Ninguno de los enlaces está subrayado ni queda marcado por haber sido pulsado con anterioridad, puesto que estas convenciones están en desuso y no encajaban con nuestro diseño.

Asimismo definimos estilos para distintos tipos de párrafos y corregimos, también desde la hoja de diseño, peculiaridades como la repetición de imágenes a lo largo de los ejes evitando así que los encabezados y fondos se repitieran a lo largo del eje x, desfigurando nuestro diseño.

Modificaciones sobre la página (HTML) original

La página Web original consistía en diversas tablas anidadas. Utilizamos este fichero como base para crear nuestras propias páginas, añadiendo o quitando los bloques que consideramos necesarios.

Sistema de enlace de páginas

Las páginas siguen el siguiente sistema de enlaces.

La distinción entre usuarios registrados y visitantes es importante, pues dependiendo de ello se mostrarán unas páginas u otras. Por defecto se presentarán las páginas de visitante y solamente si el usuario se registra tendrá la posibilidad de ver las otras.

Las páginas de visitantes son meramente informativas, de modo que para poder disfrutar de los servicios, como la generación de una sesión de entrenamiento a medida, el usuario tendrá que registrarse.

Interacción: incorporación de JSPs

Todas las acciones relacionadas con las páginas, como por ejemplo distinguir los tipos de usuario, están hechas con JSPs que toman los datos del usuario y muestran la información generada por el sistema.

El código JSP está embebido dentro del código HTML y relaciona la interfaz gráfica con el resto de nuestro sistema.

Una vez explicada la estructura de la página Web, pasemos a ver lo relacionado con las imágenes que la componen o están contenidas en ella.

1.9.5.DISEÑO EN ENTORNO 2D

Funcionalidad

Como ya hemos comentado antes, la funcionalidad de las imágenes que diseñamos en un entorno 2D es principalmente estética. Sin embargo, las imágenes que contienen texto también tienen cierta función informativa y contribuyen a que la página sea más comprensible para el usuario. Para que puedan cumplir esta función es necesario que el texto esté en el idioma adecuado y que el mensaje que contenga sea el correcto. Nada de esto era así en las imágenes con texto que nos proporcionó la plantilla, de modo que tuvimos que rediseñarlas todas.

Las imágenes 3D originales

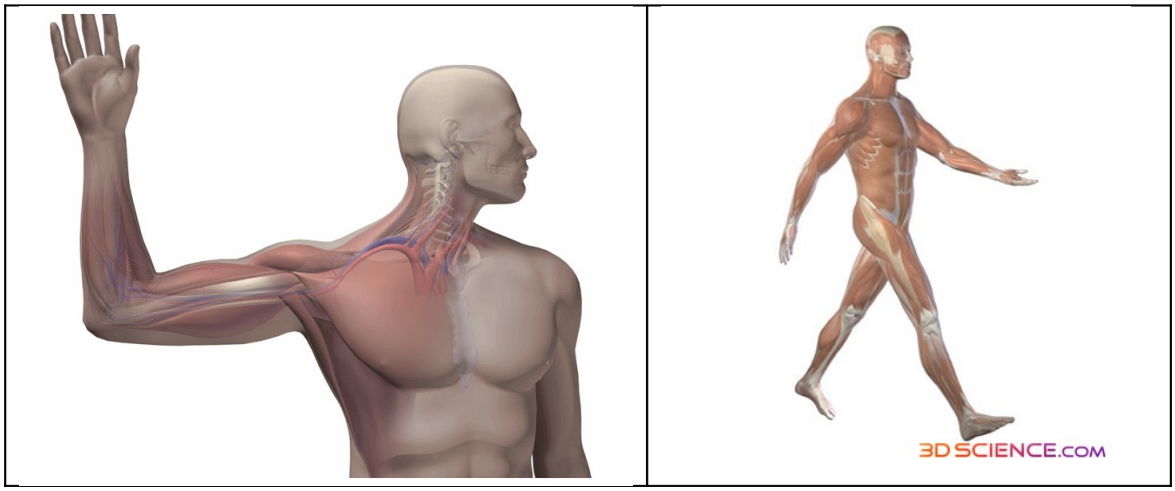
Las imágenes que hemos llamado "de estructura" se pueden dividir en tres tipos:

- Encabezados (con texto)

- Fondos (sin texto)

- Bajos (sin texto)

También tenemos algunas imágenes decorativas que pertenecen a nuestra Web de forma estática. Las obtuvimos de la empresa Zygote Media Group Inc. [54] [55], que se dedicada entre otras cosas a la creación de imágenes 2 y 3D relacionadas con la biomedicina. Esta empresa pone ciertas imágenes a disposición del público bajo licencia *open source*, que fueron las que tomamos para nuestro proyecto. A continuación están algunas de ellas:



Formato y tamaño de las imágenes

Las imágenes están todas en formato GIF o JPEG, pues son los más habituales en las páginas Web debido al grado de compresión que permiten. De hecho, las imágenes de estructura ocupan todas entre 2 y 15 Ks, a excepción del encabezado que es de 51Ks. Utilizamos las imágenes decorativas en pocas páginas, pero aun así, su tamaño suele estar entre los 60 y los 150Ks. Es importante que el tamaño de las imágenes no sea excesivo, para no ralentizar la comunicación con el usuario y que este pierda el interés en la Web.

La herramienta

Solamente tuvimos que hacer modificaciones sobre las imágenes de la plantilla, para lo que utilizamos como herramienta Photoshop 7.0 de Adobe. La elección de esta herramienta también se debe a que teníamos experiencia previa con ella y sabíamos que nos permitiría realizar todos los cambios que pudieran hacer falta.

Las modificaciones

El problema básico era que nos gustaba el diseño de los encabezados, pero las imágenes suministradas por la plantilla eran simples JPEGs que no permitían borrar las letras del texto que incluían. Lógicamente esto es así para que se contraten los servicios de la compañía que, gratuitamente, suministra las plantillas. Pero al margen de todos los encabezados con texto como este:

La plantilla incluía una única imagen en blanco que fue la que utilizamos como punto de partida.

Con ayuda de nuestra herramienta seleccionamos la forma de los otros encabezados y la "recortamos" sobre este fondo. De este modo, y escalando encabezados, imágenes de fondo e imágenes del borde inferior, obtuvimos plantillas de todos los elementos e incluso creamos los nuestros propios.

Photoshop permite además trabajar con capas, de modo que superpusimos distintas capas de texto sobre estas plantillas, obteniendo así los encabezados que queríamos. Las plantillas están todas en el formato propio de Photoshop, puesto que así permiten futuras modificaciones, pero también las hemos guardado en formato JPEG y GIF para comprimirlas y poder utilizarlas en la Web. El resultado es el siguiente:

Como tampoco nos convenció la forma del encabezado principal, decidimos transformarlo con algunas de las opciones que permite la herramienta, obteniendo así una forma que nos gustaba más. Luego le añadimos el nombre de nuestro proyecto y el eslogan con distintas letras y formatos, obteniendo así la imagen definitiva:

Con esto teníamos todas las imágenes necesarias para nuestra Web.

Todo lo explicado hasta ahora trataba sobre las imágenes diseñadas o retocadas con programas 2D. A continuación explicamos el trabajo que realizamos en un entorno de diseño 3D.

1.9.6.DISEÑO EN ENTORNO 3D

1.9.7.Introducción

Funcionalidad

La funcionalidad de las imágenes que diseñamos en el entorno 3D era la de explicar de forma gráfica y sencilla los ejercicios de gimnasia que queríamos presentar a los usuarios. Estas imágenes están en la base de datos y son elegidas de forma dinámica según el entrenamiento que se diseñe para el usuario.

¿Por qué 3D?

La razón por la que elegimos un entorno 3D es que una vez creado el muñeco, se le podía colocar en las distintas posturas en vez de tener que rediseñarlo todo, como ocurriría en un entorno 2D. Además, este tipo de diseño permitía colocar la cámara en distintos puntos, permitiendo una visión de diversos ángulos de una sola postura. Además, la posibilidad de jugar con luces, sombras, colores y texturas permitía crear imágenes con profundidad y, por tanto, escenas bastante reales.

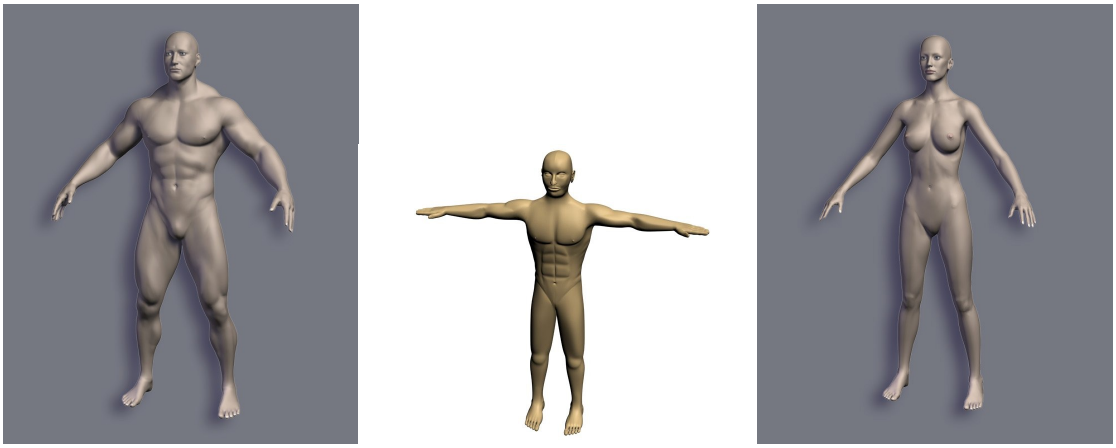
1.9.8.Planteamiento inicial

El modelo exacto

Nuestro planteamiento inicial era utilizar un muñeco lo más realista posible del cuerpo humano. Dado que no teníamos ninguna experiencia en diseño 3D, desde el principio pensamos en utilizar un modelo de libre distribución al que sólo tuviéramos que manipular pero no diseñar.

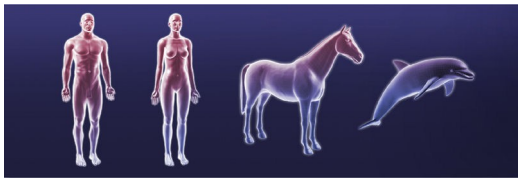
Para ello estuvimos investigando mucho tiempo en la red, especialmente en sitios relacionados con el diseño 3D. Lamentablemente estas técnicas se han utilizado sobre todo para la creación de videojuegos, de modo que la mayoría de las Webs están repletas de modelos de seres imaginarios, desproporcionados y repletos de detalles innecesarios para nuestro propósito, con los que nuestros usuarios no se habrían podido identificar.

De entre ellos, los siguientes son ejemplos de los modelos más aceptables que encontramos:



Finalmente encontramos la compañía Zygote y a través de sus enlaces obtuvimos unos modelos realistas y manipulables que podíamos utilizar para el proyecto. La razón por la que nos decantamos por ellos fue que crean sus modelos 3D trabajando con formas humanas escaneadas, atlas anatómicos ilustrados e imágenes de cadáveres.

Desafortunadamente los modelos de músculos, esqueleto, órganos internos etcétera en 3D son de pago. Pero los modelos de la forma humana, que eran los que más nos interesaban, sí son *open source* y no hubo problemas para obtenerlos. De hecho, hasta hay modelos de pose ("*poser forms*") que se adaptan específicamente a nuestros propósitos.



Los formatos

Los modelos *open source* que obtuvimos de Internet estaban en formato OBJ, de modo que resultaba fácil importarlos desde cualquiera de las herramientas de modelado 3D que existen.

La herramienta

Dado que no teníamos experiencia en diseño 3D, tuvimos que encontrar una herramienta que nos permitiera manipular los modelos. Nuestra intención era, ya que partíamos de cero, utilizar un programa de software libre de modo que investigamos qué opciones había.

En Internet encontramos mucha información sobre todo tipo de programas, desde los conocidos Maya y 3D Estudio Max, de pago, hasta todo tipo de *freeware* y *shareware*. Estos son algunos de los programas de diseño 3D que consideramos: AC3D, DX Studio, AC3D, Daz Studio, 2D/3D SunXi Viewer, 3D Canvas, 3D Render, 3DOM, 4D Blue, Anim8or, Aurora Beta Version 8.4 , AutoQ3D, Aztec 3D, Behemot, Breeze Designer, CADMAI, Calimax Modeler, DMesh, Direct 3D, Wizard, CharacterFX, Blender, etc

El primer programa que pensamos utilizar fue “Character FX” [56], pero dio problemas a la hora de descargarse y el hecho de que se tratara de una versión de prueba con sólo 30 días de validez nos acabó echando atrás.

Nos seguimos documentando y encontramos muchas referencias sobre Blender, que hoy en día es una de las más serias competencias a programas de pago como Maya. En aquel momento desconocíamos esto, de modo que los comentarios de otros internautas sobre este software [57] resultaron determinantes. La mayoría le daban una puntuación muy alta a Blender y lo describían como una herramienta muy amplia y con infinidad de posibilidades, aunque difícil de aprender a usar. Finalmente la versatilidad de Blender y su amplia comunidad de usuarios primaron sobre la complejidad de su interfaz y decidimos utilizar esta herramienta. La versión que utilizamos para el proyecto es la 2.40, aunque actualmente ya se puede descargar la 2.41 de la página oficial [58].

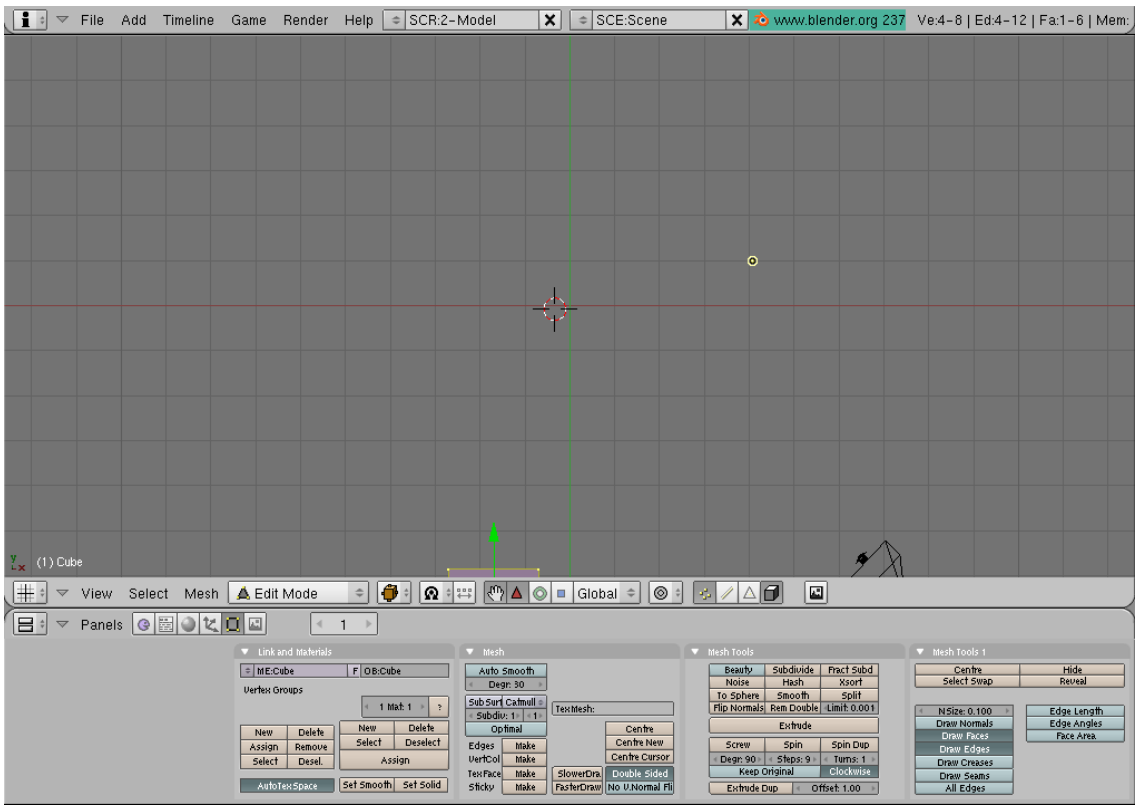
Documentación de Blender

El interfaz de Blender no resulta nada intuitivo y menos si no se tiene ninguna experiencia en el ámbito del diseño 3D. La complejidad de este tipo de programas es extraordinaria, así que el habitual acercamiento tipo “prueba y error” no era viable en este caso. Hay demasiados botones, pantallas, paneles y menús como para saber por dónde comenzar si no es con la ayuda de un manual.

Afortunadamente, la comunidad de usuarios ha elaborado diversas páginas de apoyo así como un manual bastante extenso cuya versión inglesa se titula "Blender 3D: From Noob to Pro”. Este manual nos guió en nuestros inicios y resultó de gran valor. Está escrito por diferentes personas y salpicado de comentarios de otros usuarios, lo que hace que su lectura resulte más amena, aunque no siempre está actualizado respecto a la última versión de Blender ni tiene explicaciones para todas las cosas que se pueden hacer. Tanto la herramienta como su documentación están en constante desarrollo, lo cual supuso ciertos problemas en las partes menos documentadas, aunque el foro de diseñadores [59] suplió con creces lo que faltaba en el manual.

Primeros pasos en Blender: Nociones básicas

El interfaz de Blender tiene el siguiente aspecto:



En la imagen superior se puede ver el "*3D Viewport*" que es donde se visualiza el trabajo de diseño, el menú superior y la ventana de botones, que a su vez se subdivide en módulos que cambian según la opción elegida. Toda estas opciones pueden resultar muy confusas, pero una vez se sabe manejar resultan verdaderamente prácticas.

Nuestro primer intento fue importar el modelo humano que habíamos obtenido, lo cual no supuso demasiados problemas. Pero rápidamente nos dimos cuenta de que no teníamos suficiente conocimiento para manejar el modelo, así que decidimos seguir paso a paso las indicaciones del manual para familiarizarnos con el entorno.

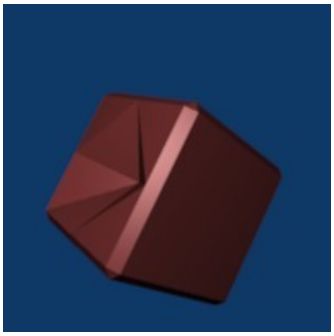
No queremos entrar en demasiados detalles y que la explicación se convierta en un segundo manual, pero hay ciertas nociones básicas necesarias para entender el trabajo que hemos realizado. Hemos tratado de resumirlas a continuación.

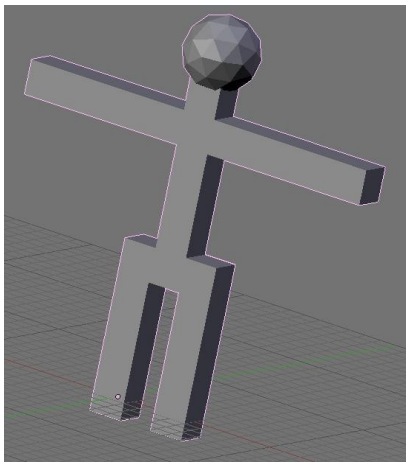
Una de las cosas que más nos llamaron la atención a la hora de utilizar Blender es que prácticamente cada botón y combinación de teclas tienen un significado. Esto es un arma de doble filo, pero con el tiempo se aprende a utilizar un pequeño número de ellas. Las más útiles son los números del teclado numérico, que sirven para volver a vistas habituales como frontal y lateral. Aunque hay un modo de adaptar los teclados de los portátiles, ya que no suelen tener teclado numérico, nosotras nos decantamos por utilizar un ordenador de sobremesa (los números que se encuentran sobre las teclas no tienen la misma función).Otras teclas útiles son: G para coger objetos ("*grab*"), S para escalarlos ("*scale*") y R para rotar ("*rotate*"). Con F12 se hacen los *renderizados* y con el ratón se pueden alejar, mover o girar tanto los paneles como el *viewport*.

Una vez seleccionado un objetos se pueden visualizar los ejes X, Y o Z pulsando la tecla correspondiente. Esto resulta muy útil para restringir una rotación a un solo eje, cosa de la que hicimos mucho uso a la hora de crear las poses.

Finalmente queda por explicar que hay distintos modos en los que se pueden visualizar los objetos, aunque para este proyecto sólo hemos hecho uso de 3: el modo objeto, el modo edición y el modo de pose.

Siguiendo el manual aprendimos a modelar objetos a partir de las formas básicas que Blender proporciona, añadir luces, texturas y finalmente a animar un muñeco.
A continuación están algunas imágenes de aquellos primeros intentos:





Aplicar lo averiguado a nuestro modelo

Una vez habíamos aprendido a manejar la herramienta, decidimos poner en práctica nuestro plan de diseño. Este pasaba por aplicar todos los pasos necesarios para la animación al modelo que habíamos obtenido de Zygote. Estos pasos son los siguientes:

- Crear el modelo
- "*Rigging*"
- "*Skinning*"
- Posar
- *"Renderizar"* o crear un video

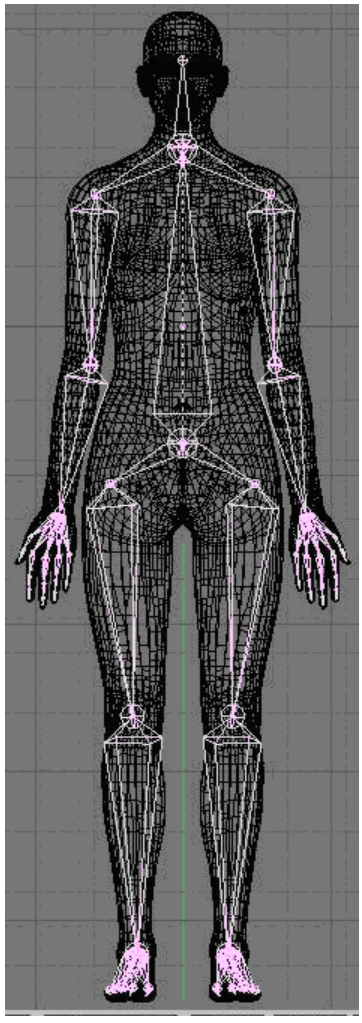
Rigging

Puesto que ya teníamos el modelo, pasamos directamente al segundo punto, que se denomina "*rigging*" y consiste en añadir una armadura al objeto 3D que se desea manipular. Para entender esto hay que saber que los modelos no son más que puntos (vértices) que están unidos y forman una malla dentro de un espacio de tres dimensiones. Pero por mucho que tengan forma de brazos o piernas, no contienen información respecto al tipo de desplazamiento que pueden hacer. Técnicamente los

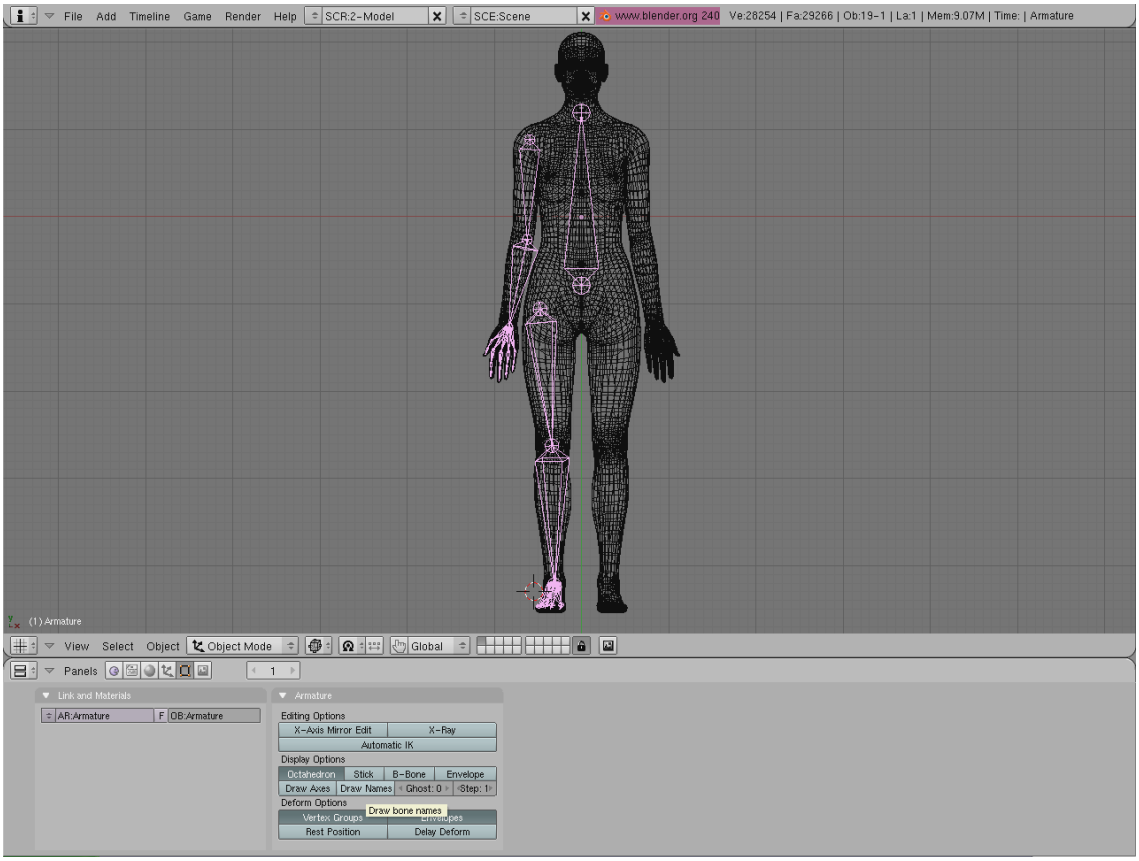
puntos se pueden mover de forma individual o colectiva, pero para conseguir algo parecido a un movimiento hay que aplicar técnicas más sofisticadas.

Dentro del *viewport* se pueden situar muchos objetos que luego resultan invisibles al hacer un *renderizado*, como por ejemplo las fuentes de luz de las que sólo se ve el efecto que producen. Otro de ésos objetos es la armadura. La armadura es una estructura que se asocia a una malla y lleva información sobre los vértices con los que se relaciona. Esto permite que luego no haya que mover cada punto de la malla para recrear un ejercicio, sino que baste con mover la armadura y la malla se deformará consecuentemente. En la práctica esto resulta bastante más complicado de llevar a cabo, pero eso ya lo explicaremos más adelante.

Lo primero fue añadir la armadura que, ya que se trata de un modelo humano, se podría llamar también "esqueleto". Este esqueleto está compuesto por huesos que se encuentran dentro de una jerarquía. De modo que si el hueso del brazo es padre del hueso de la mano, si movemos el brazo la mano seguirá el movimiento. Esto se aprende fácilmente con un simple muñeco de prueba, pero resulta más complejo a la hora de reproducir el esqueleto humano. Un error que cometimos fue creer que una buena imitación del esqueleto humano haría que nuestro modelo se moviera de una forma muy natural. Este tipo de razonamiento puede parecer muy lógico, pero un experto en diseño 3D sabe que para conseguir un movimiento natural a veces es mejor colocar huesos en sitios donde no los hay, omitir otros o incluso situar algunos huesos fuera de la malla.
Evidentemente nosotras no sabíamos esto, de modo que nos ceñimos a nuestro plan inicial y creamos la siguiente estructura:

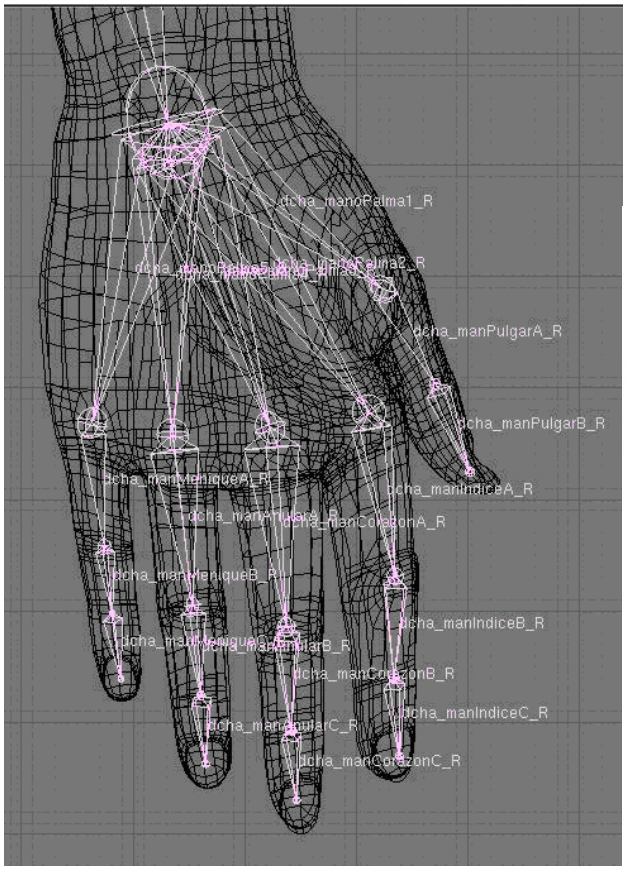


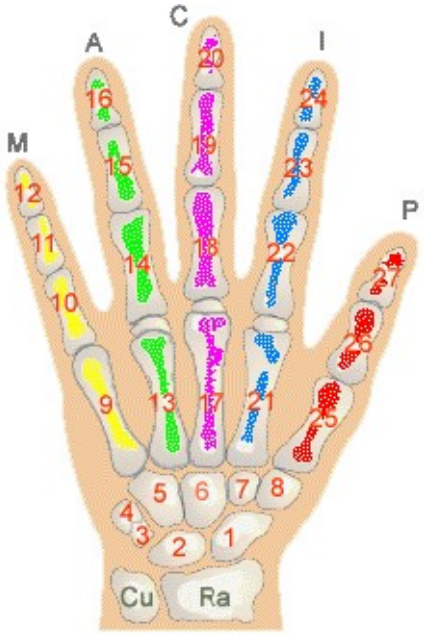
Creamos este esqueleto paso a paso, partiendo del hueso de la columna y desarrollando a partir de él la jerarquía de huesos para un lado del cuerpo. La idea que hay detrás de diseñar sólo una mitad del cuerpo es, que los modelos suelen ser simétricos y por tanto interesa invertir mucho tiempo en hacer muy bien un lado y luego utilizar la herramienta de duplicado y espejado para obtener el otro lado. De hecho, Blender hasta es capaz de adaptar los nombres, siempre que se siga su nomenclatura para distinguir los lados. De este modo se ahorra tiempo y se obtienen modelos perfectamente simétricos. Esto es algo que aprendimos haciendo los ejercicios propuestos en el manual y que aplicamos también al esqueleto. De modo que a medio camino teníamos una situación parecida a esta:

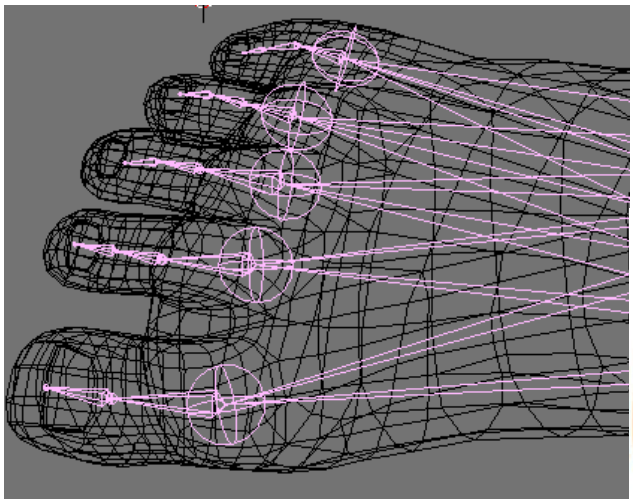


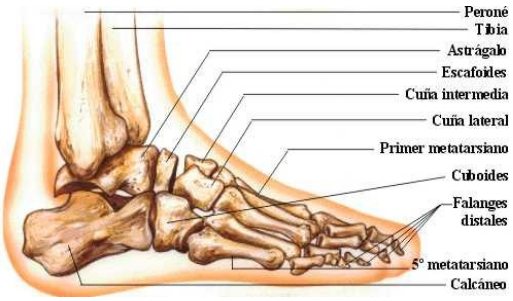
Un buen ejemplo del nivel de detalle que alcanzamos con el esqueleto son las manos y los pies. Para poder crear esta armadura nos fue imprescindible documentarnos sobre lo detalles de la anatomía humana, al menos en lo que a huesos se refiere. De hecho, en la primera versión que hicimos de la armadura, los huesos tenían sus nombres reales. Pero esto no lo mantuvimos porque resultaba confuso para nosotras mismas y, sobre todo, porque Blender mostraba los listados de huesos en un orden un tanto particular para el que convenía renombrarlos. (Esto es una desventaja respecto a Maya, que muestra los huesos por jerarquías y permite, además, reorganizarlos de distintos modos).

Las próximas imágenes son capturas de pantalla de la mano con sus huesos y nombres de una de nuestras versiones intermedias, así como un recorte de la imagen de uno de los pies del mismo muñeco.









Skinning

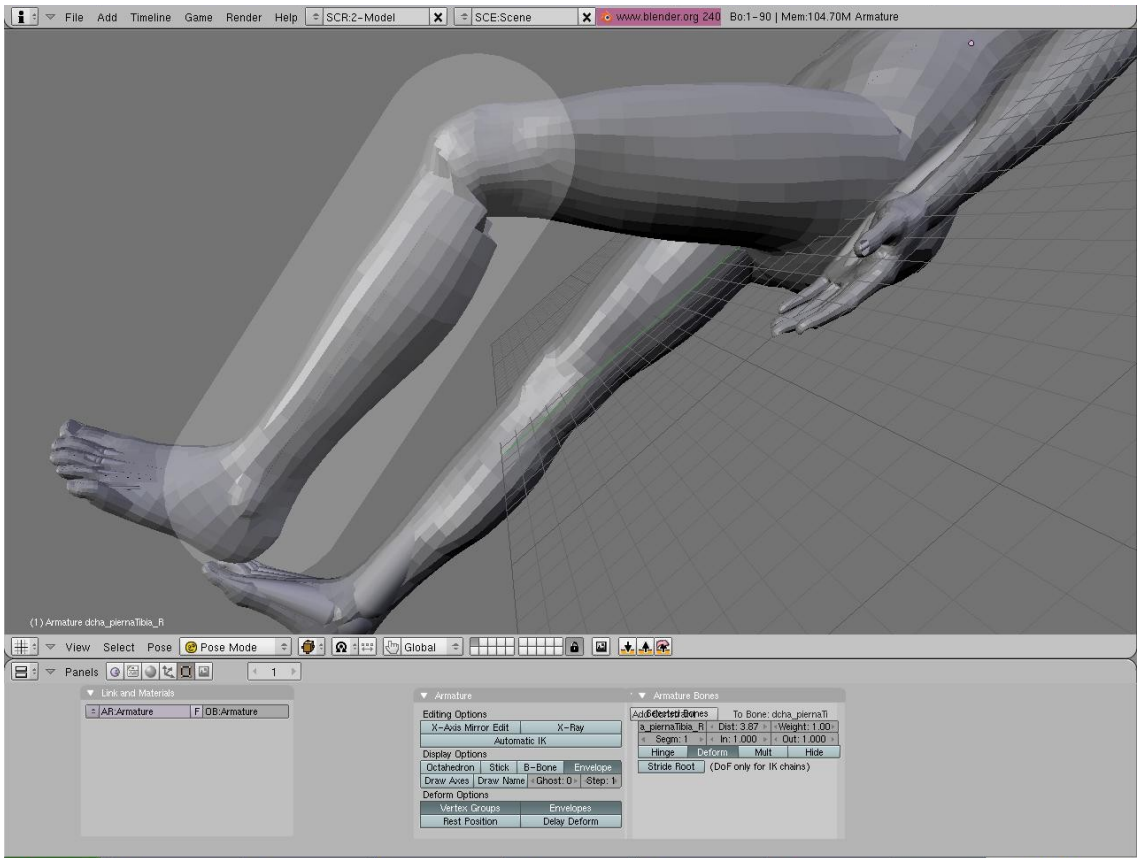
"*Skinning*" se refiere al paso en el que se establece la relación ente los vértices y los huesos de la armadura. Para poder hacer esto, es necesario relacionar previamente el objeto armadura y el objeto malla. Esto se consigue mediante la relación de "emparentado". Lo siguiente que hay que hacer es indicarle a Blender qué vértices se deben mover cuando se mueve un hueso determinado. Para ello hay distintas opciones.

La más habitual es que el programa cree grupos de vértices a partir de los puntos más cercanos al hueso. Este criterio de distancia es práctico para modelos simples, pero no

lo suficientemente útil para un modelo tan complejo como el nuestro. El problema de esta técnica es que ciertos vértices quedan sin asignar, de modo que al mover el brazo los puntos de codo pueden quedar atrás porque Blender no haya considerado que pertenecen al brazo. También puede ocurrir que se relacionen vértices que no deberían moverse, produciendo un efecto indeseado a la hora de posar.

Los vértices afectados por el movimiento de un hueso también se pueden determinar mediante la técnica de "envoltorios". Esto es una supuesta mejora que se ha incluido en Blender 2.40 pero que no nos resultó útil porque no permite alcanzar un nivel de detalle muy alto.

En esta imagen se ve un envoltorio, que es la zona ovalada de color gris claro.

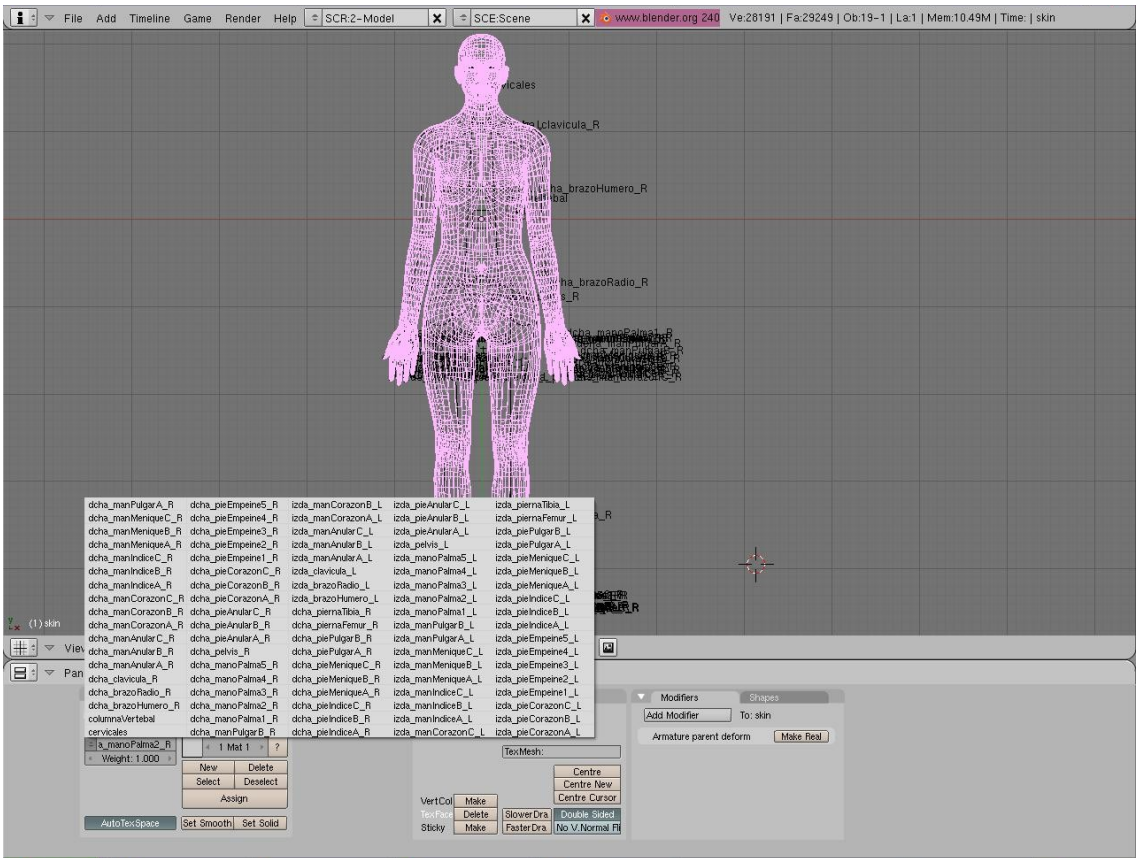


Otra opción es seleccionar los vértices afectados personalmente y relacionarlos con los huesos. No es necesario hacerlo de forma individual, ya que Blender dispone de distintas herramientas para seleccionar grupos de vértices que son extremadamente útiles. Además existe la posibilidad de ver y manipular una malla como un conjunto de vértices, aristas o incluso planos.

Nosotras utilizamos una técnica híbrida, permitiendo a Blender crear los grupos de vértices para luego retocarlos utilizando las opciones anteriormente descritas. Para ello hay que elegir cada uno de los huesos creados y determinar su zona de influencia.

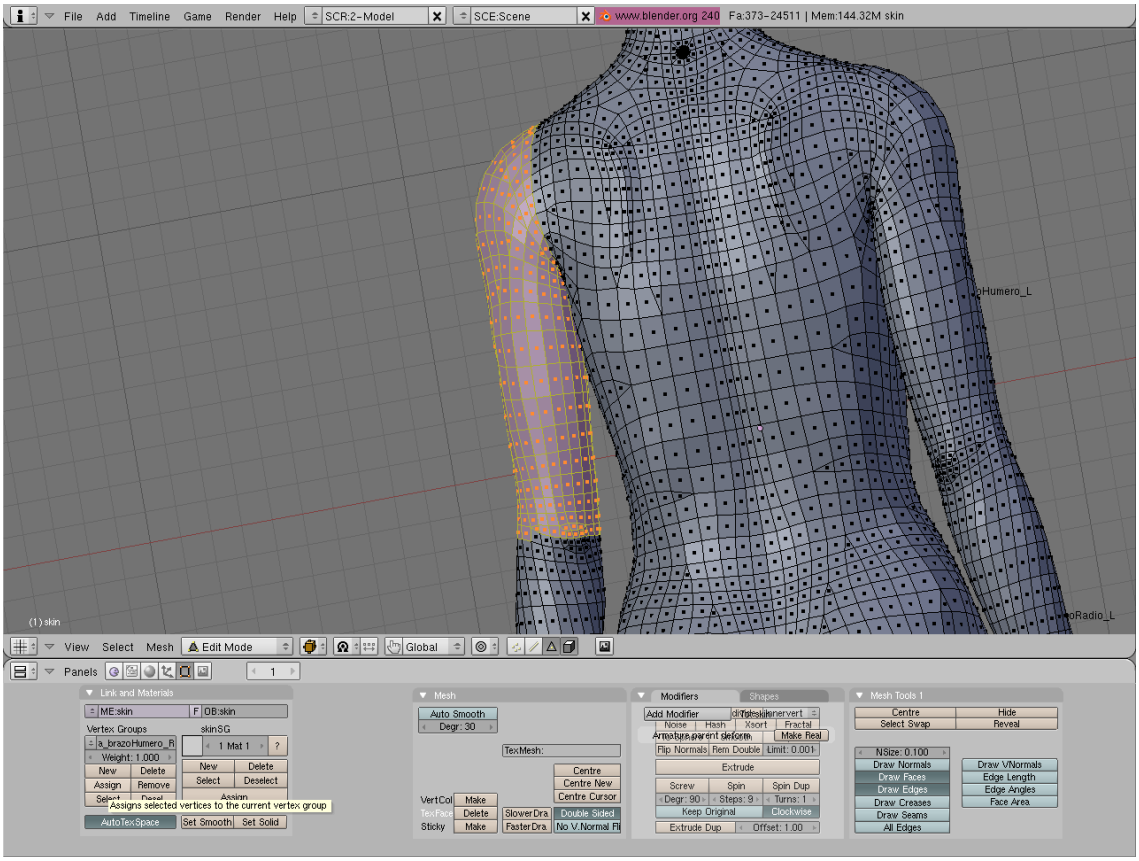
En la imagen siguiente se pude ver la malla y el submenú con los nombres de todos los huesos que se despliega en el panel de herramientas. Precisamente por la forma en que Blender muestra los huesos en este desplegable decidimos nombrarlos de forma distinta

en nuestra segunda versión, desechando su renombrado automático. Esto suponía un esfuerzo añadido, pero al menos los lados izquierdos y derechos se encuentran agrupados para el resto del tiempo que se trabaja con el modelo. (Esto es un aspecto de Blender que debería ser mejorado en las futuras versiones.)



En la siguiente imagen se pueden ver los vértices relacionados con el hueso que llamamos “espinilla_L”.

Como mencionamos antes, también se puede visualizar la malla no como vértices sino como colección de planos. Además, se puede cambiar de modo sólido a transparente con pulsar una tecla (Z). La imagen siguiente muestra esto mismo cuando definíamos la zona de influencia del húmero.



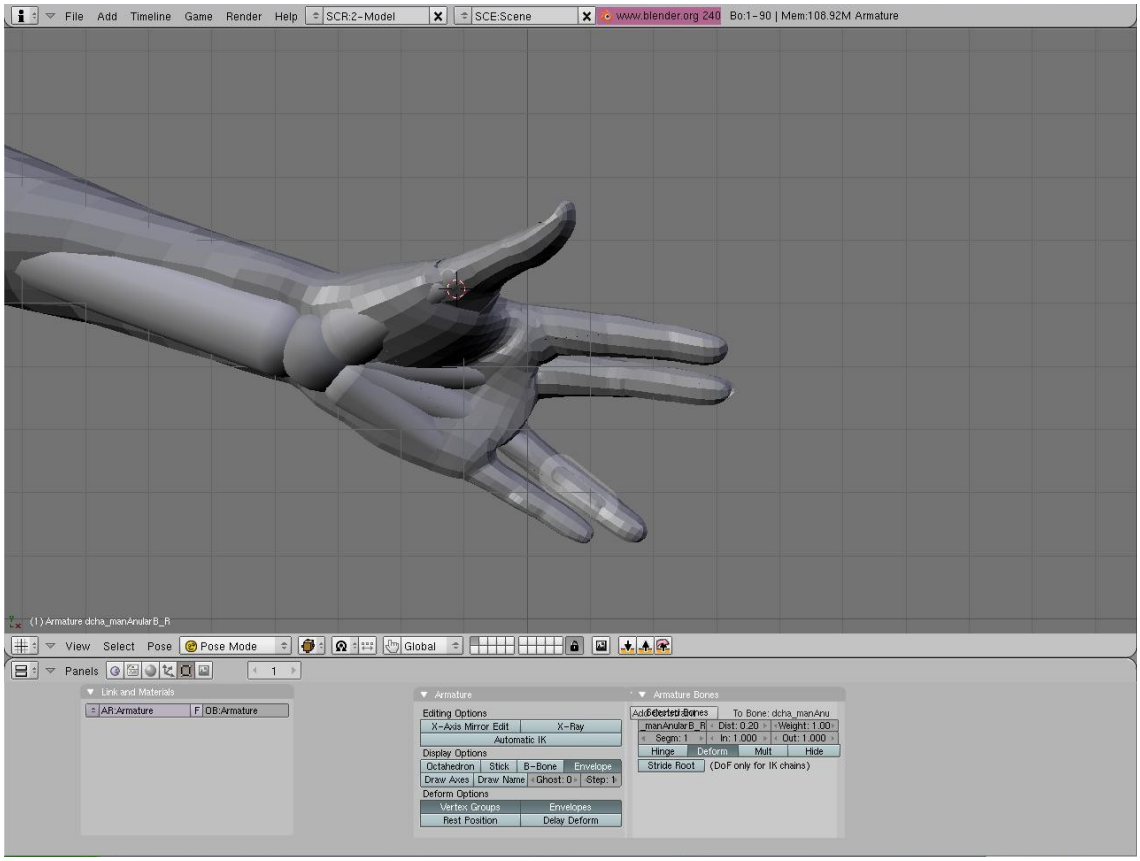
Posado

Teóricamente, una vez realizado el *rigging* para cada hueso del cuerpo se puede utilizar el modelo para la fase de posado. En la práctica lo más recomendable es ir cambiando de un modo a otro para ver el resultado del *rigging*. Esto hace que la definición de la zona de influencia sea una tarea muy laboriosa. Pero es mucho más sencillo comprobar si un hueso mueve los vértices que debe nada más definirlo, que esperar a finalizar la fase de *rigging* y empezar a mover todo el esqueleto luego.

Aun trabajando con mucho cuidado, a la hora de hacer el posado siempre surgen nuevos problemas. Uno de los que nos encontramos era que lo que habíamos considerado una malla única no era tal. Lo que hace la función de piel es ciertamente un solo objeto, pero los ojos estaban hechos a partir de varios objetos y las uñas de las manos también eran un elemento a parte. Estos elementos no se veían afectados por el movimiento de los huesos, provocando imágenes bastante curiosas. (Por ejemplo uñas suspendidas en el aire o una cabeza alejada de los ojos.) Este problema tiene varias soluciones posibles. Las mallas se podrían emparentar entre si, por ejemplo las uñas con la piel, o incluso relacionar el esqueleto con ésos otros objetos. Nosotras aplicamos la técnica que nos pareció más conveniente en cada caso.

Una vez resueltos todos los contratiempos, teníamos un modelo manipulable aunque no siempre el posado daba el resultado que esperábamos. Las siguientes son algunas capturas de pantalla que explican muy bien la situación a la que llegamos.

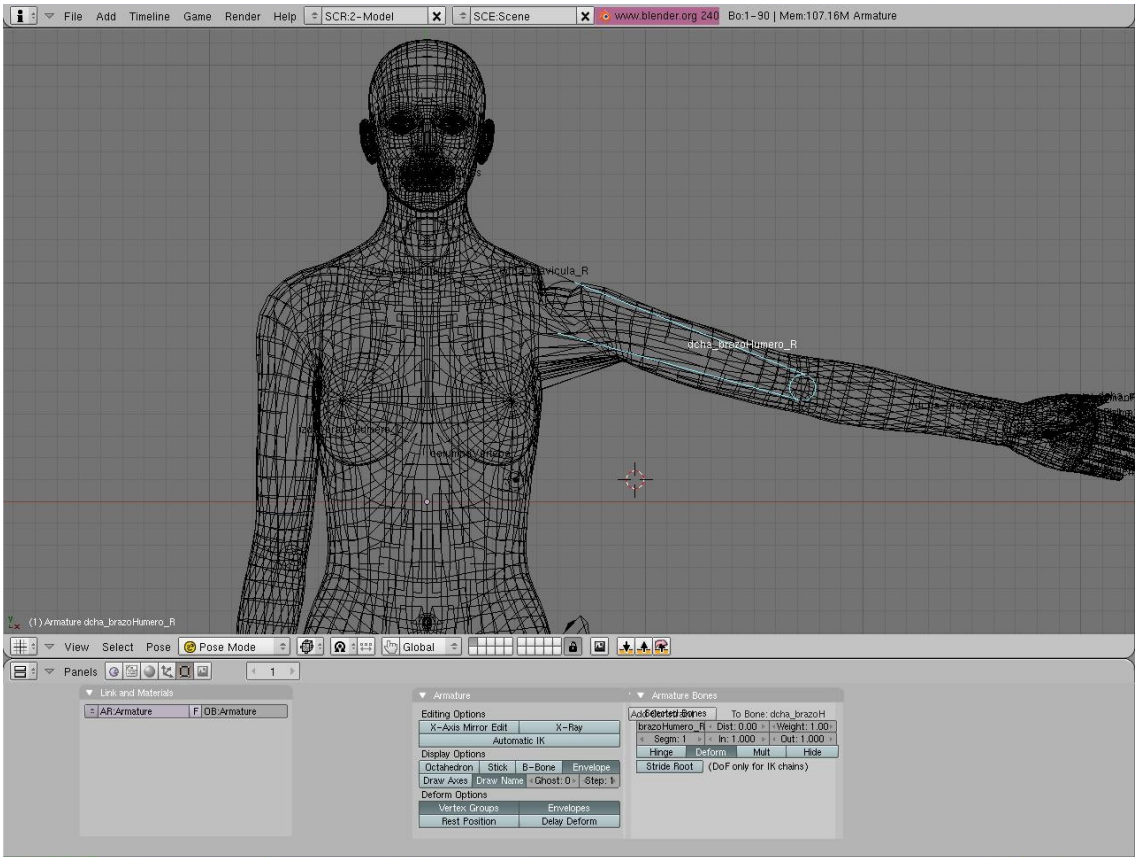
En algunos casos, tras muchos ajustes obtuvimos zonas que se movían perfectamente, dando el resultado esperado.

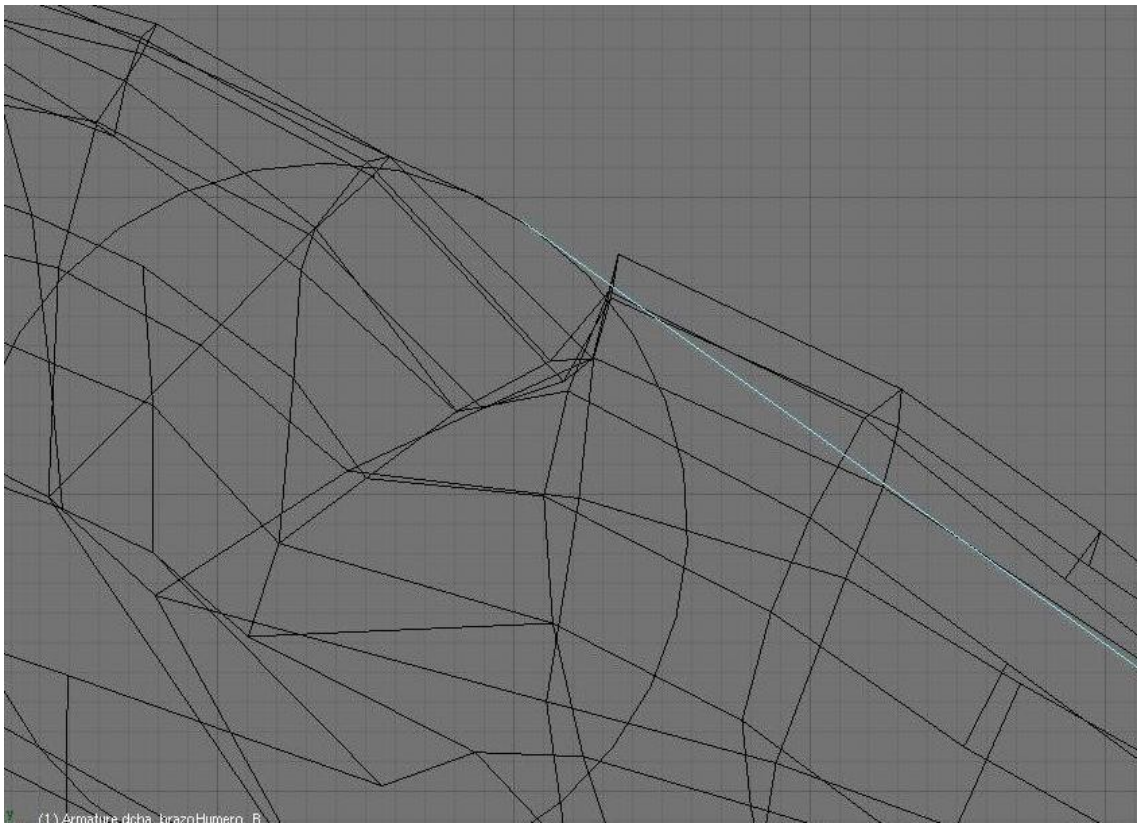


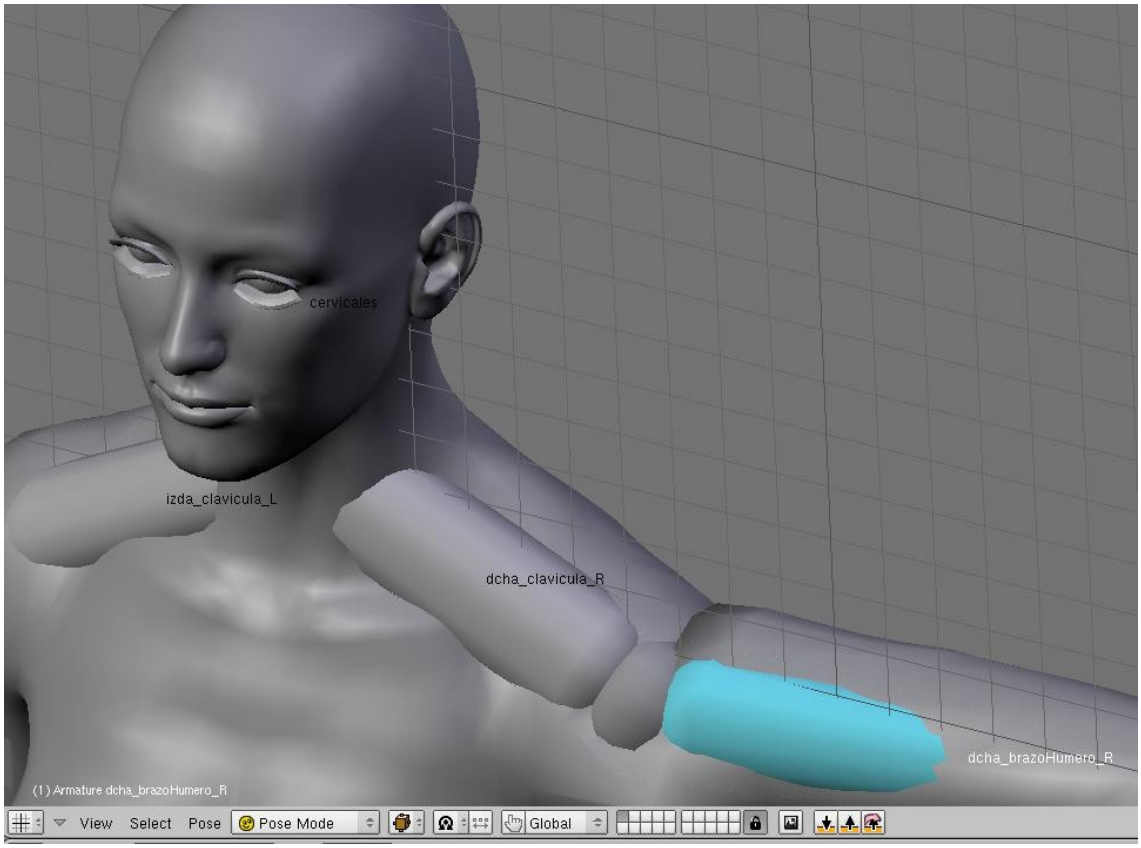
Pero en otros, por mucho que retocáramos los vértices afectados, la malla no se deformaba de forma natural.

En este *render* se puede apreciar cómo mover el brazo supone un corte exacto entre las zonas que corresponden al brazo y las que no.

Las imágenes siguientes son capturas de pantalla de distintos momentos en los que intentamos resolver el problema. Nótese, que al tratarse de capturas de pantalla en algunas se ven los huesos y el problema puede quedar algo enmascarado. Pero a la hora de realizar un *renderizado* de la escena estos huesos no se ven y el problema reaparece.


File Add Timeline Game Render Help SCR:2-Model SCE:Scene www.blender.org 240 Bo:1-90 | Mem:107.16M Armature
dcha_brazoHumero_R
(1) Armature dcha_brazoHumero_R
View Select Pose Pose Mode Global
Panels
Link and Materials
AR:Armature OB:Armature
Armature
Editing Options
X-Axis Mirror Edit X-Ray
Automatic IK
Display Options
Octahedron Stick B-Bone Envelope
Draw Axes Draw Name Ghost: 0 Step: 1
Deform Options
Vertex Groups Envelopes
Rest Position Delay Deform
Armature Bones
To Bone: dcha_brazoH
brazoHumero_R Dist: 0.00 Weight: 1.00
Segm: 1 In: 1.000 Out: 1.000
Hinge Deform Mult Hide
Stride Root (DoF only for IK chains)


(1) Armature dcha_brazoHumero_R



Renderizado

Renderizar es un término que se refiere crear una imagen 2D a partir de una en tres dimensiones. La idea es que los objetos dentro del *viewport* componen una escena. Estos objetos pueden ser directamente visibles y tener colores y texturas, como por ejemplo la malla de nuestro modelo. También hay objetos que se encuentran en la escena, pero luego no se ven al *renderizar*. Esto son, por ejemplo, las luces. Dependiendo de la escena, incluimos una o varias fuentes de luz de distintos tipos. Otro elemento crucial es la cámara. Hacer un *renderizado* equivale a "tomar una foto". Por ello es importante que la cámara enfoque lo que se desea mostrar y esté en una posición buena respecto a los elementos de iluminación y la malla.

Existe la posibilidad de que la cámara siga a un objeto, de modo que este siempre queda en el centro de la "lente" aunque se mueva la cámara. Esta técnica, llamada "*tracking*", puede ser útil y la utilizamos al principio, pero en el caso de los ejercicios deportivos no es demasiado buena. Esto se debe a que el modelo llega a deformarse de manera un tanto extrema y a que la cámara enfoca el punto central de la malla. Esto obligaba muchas veces a situar la cámara extremadamente lejos de la malla, para que saliera el modelo completo. Finalmente decidimos encargarnos personalmente de situar la cámara en el ángulo y posición correctos para obtener la imagen más deseable.

En Blender existen ciertas opciones de *renderizado* por defecto, de modo que el color del "mundo", es decir, del fondo, es siempre azul oscuro. Modificarlo a transparente puede dar muchos problemas y en el foro de esta herramienta nos desaconsejaron hacer

esto. Pero más adelante nos decidimos a cambiar el fondo a blanco, para que encajara mejor con la página Web en la que acabarían mostrándose los *renders*.

1.9.9.Problemas y soluciones

Aunque habíamos logrado hacer *renderizados* de nuestro modelo, no estábamos satisfechas con todos los resultados obtenidos. Llegados a este punto nos encontrábamos ante un problema cuya solución ya no podíamos encontrar en el manual. Precisamente lo referente a la animación no está aún bien documentado en dicho manual, lo que dificultaba nuestra situación, de modo que decidimos consultar a alguien con experiencia en el mundo del diseño 3D.

Camino hacia la solución

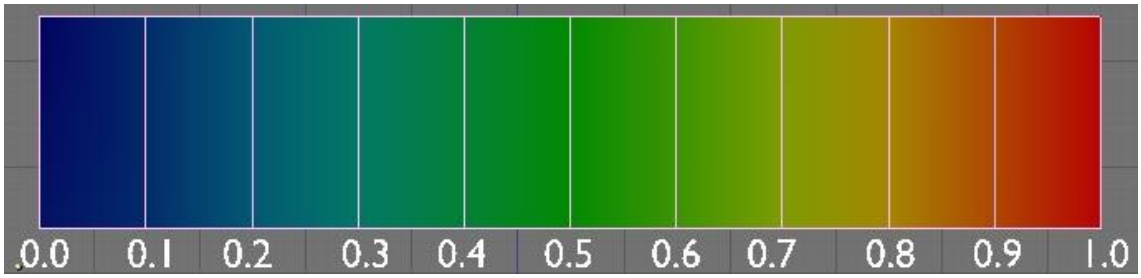
Finalmente nos pusimos en contacto con Albert Meco, un alumno de nuestra facultad que realizó un curso de Maya, cuyo proyecto está también relacionado con el diseño en tres dimensiones y que colabora con el master de desarrollo de videojuegos que se imparte en la FDI. Su comentarios y recomendaciones nos sirvieron para entender mejor cómo se enfrentan a este tipo de problemas personas con más experiencia en el mundo del diseño. Además nos indicó una solución al problema de la deformación de la malla.

Este problema derivaba de la "costura" exacta entre los vértices que se veían afectados por el movimiento de un hueso y los que no. Este problema se puede solucionar mediante el "pesado" de los vértices. La herramienta de pesado está disponible en Blender y sabíamos de su existencia, pero ni la documentación es muy extensa ni habíamos comprendido todo lo que implicaba.

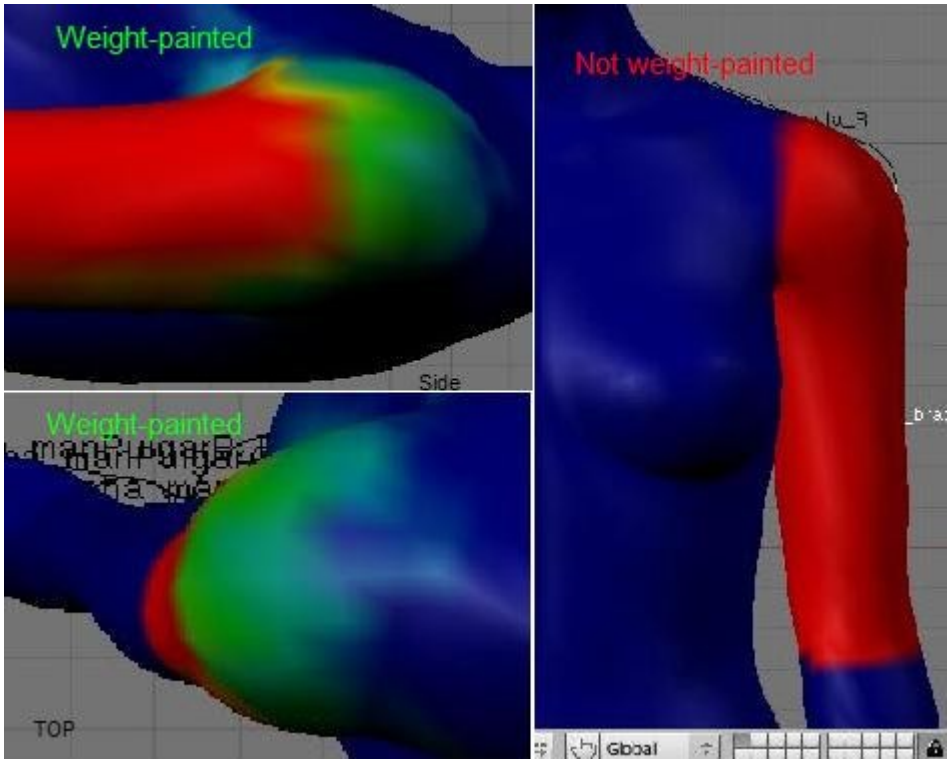
El pesado de los vértices

La idea de pesar los vértices es poder variar la fuerza con la que influye el movimiento de un hueso sobre un vértice. De este modo, un vértice situado sobre el brazo deberá tener un peso muy alto, pues deberá moverse prácticamente tanto como el propio hueso que provoca el desplazamiento. Sin embargo, un vértice del hombro deberá verse afectado en menor medida. De este modo, la deformación de la malla es más progresiva y se evitan las incisiones que ocurrían en nuestro modelo. Sin embargo, cuando se crean las zonas de influencia los valores por defecto son uno, si el vértice está dentro de la zona, o cero si no lo está. Para obtener una deformación suave hay que conocer la herramienta de pesado y aplicarla de forma explícita, aunque este paso no esté descrito en el manual.

El pesado se visualiza en Blender mediante colores y la técnica consiste en pintar el modelo por zonas tratando de conseguir el peso correcto en el vértice adecuado. El espectro en el que se reparten los pesos es el siguiente.



Un acercamiento al pesado correcto de una malla se puede ver en la imagen siguiente, que nos proporcionó como sugerencia otro miembro del foro de Blender, en base a nuestro modelo 3D.



Blender vs. Maya

Tanto Blender como Maya tienen herramientas de pesado y muchas de las técnicas básicas de diseño 3D en común. Pero también difieren en algunos aspectos que Albert nos comentó. Esto nos llevó a valorar la posibilidad de migrar nuestro trabajo de diseño a Maya. Finalmente nos dimos un plazo para intentar resolver el problema en Maya o, sino, buscar un nuevo enfoque para nuestro diseño. Quizás resulte extraño que no siguiéramos utilizando Blender, pero las siguientes razones nos inclinaron a favor del cambio.

- El pesado de vértices es una tarea muy laboriosa, que no responde a una ciencia exacta sino que se aprende y mejora a medida que se realizan distintos modelos. Sin embargo, nosotras carecíamos de esta experiencia y también del tiempo para

probar cientos de veces el peso más oportuno para cierto grupo de vértices. Evidentemente este problema es inherente al pesado y no cambia por utilizar una herramienta u otra. Pero la gran diferencia entre Blender y Maya es que Maya permite el espejado de los pesos. Es decir, que una vez realizado correctamente el pesado de los vértices de un brazo, el pesado de los vértices del otro brazo se puede obtener por simetría. Esto no es sólo una ventaja respecto al tiempo invertido, sino que también el resultado es mucho mejor, más simétrico, del que jamás se podría obtener al hacer ambos lados de forma separada (al menos para una malla de estas características).

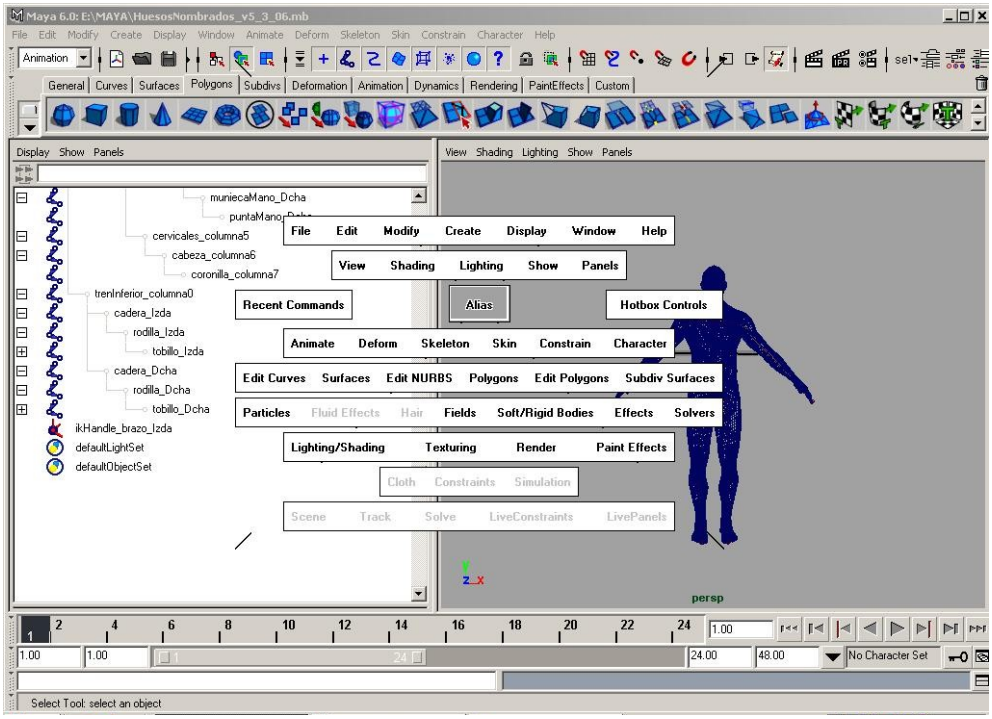
- Además Maya permite exportar pesados, que guarda en ficheros separados de las mallas. Esto permite tener distintos pesados para una sola malla y poder aplicar el que más convenga en cada momento e incluso aplicar el pesado a un modelo similar.
- Otro punto a favor de Maya es su herramienta "IK-handle". Esta permite seleccionar distintos huesos y relacionarlos de modo que, cuando se quiere realizar un posado, el programa calcula la posición más probable de todos los huesos involucrados. Es difícil explicar esto en texto, pero se podría describir así: Si en Blender queremos que el modelo se apoye contra la pared con el brazo doblado, será necesario colocar primero el brazo en la posición, luego el antebrazo, la muñeca y finalmente la mano. En Maya se pueden relacionar todos estos huesos con un IK-Handle de modo que bastaría mover la mano hacia atrás para que el programa sitúe el resto de huesos en la posición que el diseñador más probablemente desea. Es posible que Blender también tenga está opción o que sus futuras versiones la incluyan, pero la falta de documentación en este aspecto equivalía, para nosotras, a no contar con ello. Así que la existencia de los IK-Handles nos hizo pensar que el tiempo que invertiríamos en migrar a otra herramienta se vería compensado con el tiempo que ahorraríamos a la hora de realizar los posados.
- Finalmente, también gracias a los consejos de Albert, entendimos que la armadura no tenía por qué parecerse a la de un humano para obtener los resultados más favorables. De modo que decidimos crear una armadura más adecuada. Esto suponía comenzar desde cero con el trabajo de animación y, ya que íbamos a hacer este esfuerzo, decidimos aprovechar las ventajas de Maya.

Observación: Maya es una herramienta de pago, pero está disponible en los ordenadores del laboratorio número 11 de nuestra facultad.

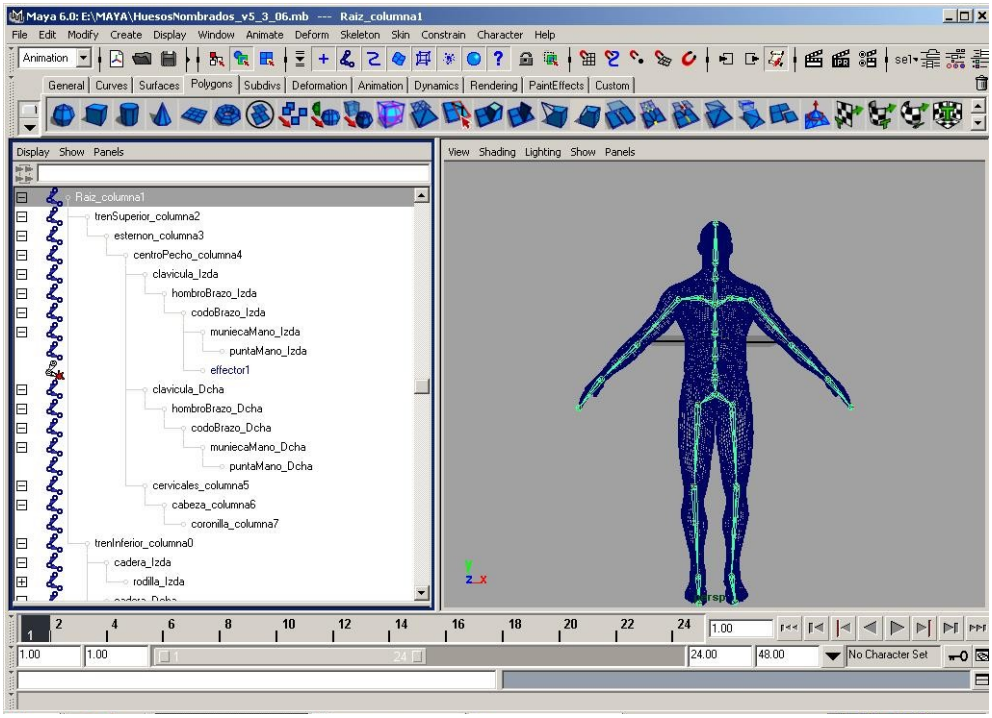
Trabajando con Maya

Al igual que Blender, Maya tiene la mayoría de sus comandos accesibles desde el teclado. El aprendizaje de esta nueva herramienta se planteaba aún más laborioso que el que tuvimos que completar para saber manejar Blender, ya que Maya es un software aún más amplio.

Un buen ejemplo es la cantidad de menús que aparecen en pantalla con sólo pulsar la tecla de espacio, como se puede apreciar en la siguiente imagen.

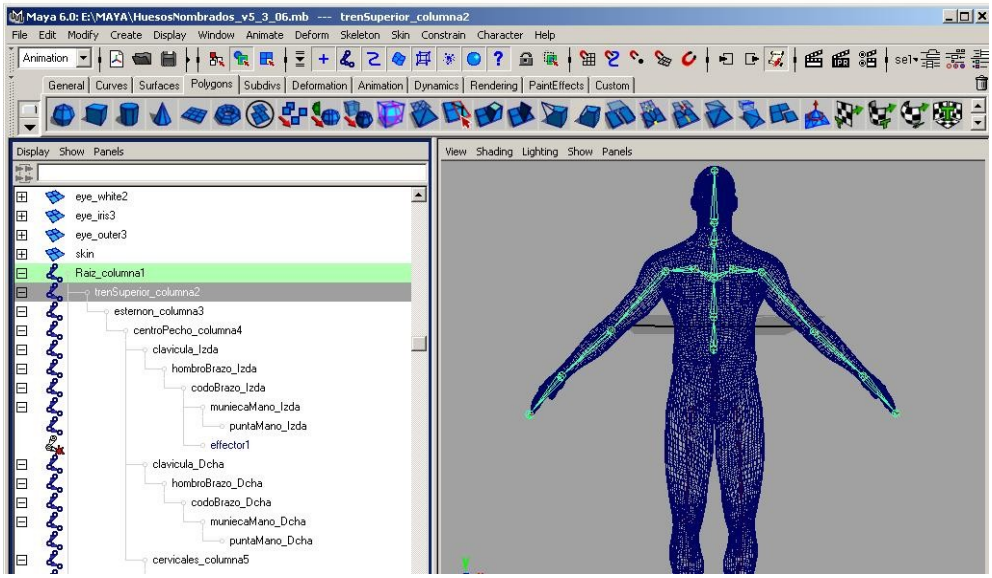


Una ventaja es que disponíamos del valioso asesoramiento de Albert y de mucho conocimiento relacionado con el diseño 3D. Pero entender lo que hay que hacer no equivale a saber manejar una nueva herramienta, de modo que la mayor parte del tiempo que nos habíamos dado para obtener resultados con Maya la invertimos en volver a crear y nombrar un esqueleto.

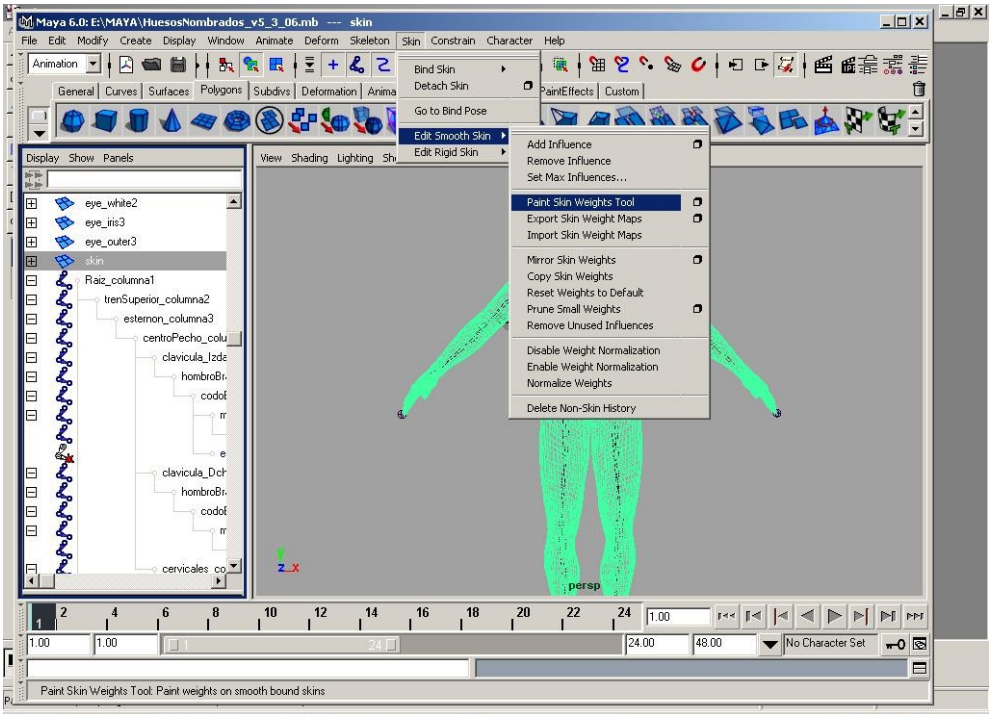


Este trabajo nos permitió apreciar ciertas ventajas de Maya, como la ordenación de los huesos por jerarquías. Esta jerarquía permite seleccionar la raíz de una subjerarquía en

el panel izquierdo y trabajar con todos los huesos que dependían de ella. En la siguiente imagen se puede ver cómo queda la pantalla cuando se selecciona el tren superior.



Una vez completado el esqueleto, decidimos continuar con el pesado. Maya dispone de múltiples opciones, como se puede apreciar en la siguiente imagen.



Sin embargo, el trabajo de pesado resultó muy complicado. No por la tarea en si, sino por nuestro desconocimiento de todos los demás aspectos de Maya. A diferencia de Blender, en Maya no habíamos trabajado con modelos ni problemas más simples para familiarizarnos con el entorno, porque pensamos que no sería necesario. Pero nuestro desconocimiento del medio acabó ralentizando sumamente el trabajo. El plazo de

tiempo que nos habíamos dado para obtener resultados con Maya estaba concluyendo y, aunque teníamos la certeza de estar yendo por el camino correcto, no nos convencía la estrategia que estábamos siguiendo.

Seguimos preguntando y documentándonos, hasta que finalmente dimos con el verdadero origen del problema. Como ya hemos explicado, la deformación que sufría nuestra malla se podía resolver mediante el pesado. Pero esta labor es extremadamente laboriosa para mallas con un número de vértices muy elevado, y este era precisamente nuestro caso. Aunque las técnicas eran las correctas, el modelo que habíamos elegido no era el más apropiado para el tipo de labor que queríamos realizar.

Aunque habíamos aprendido que para los videojuegos se utilizan modelos con un número menos de vértices para obtener buenas prestaciones, creíamos que esto sólo se aplicaba a situaciones en las que el movimiento se tiene que generar de forma dinámica. Pero también para nuestro caso resultaba contraproducente trabajar con un modelo de tan alto nivel de detalle, especialmente porque esto dificultaba enormemente el pesado.

Las herramientas de diseño 3D cuentan con distintas opciones de suavizado que permiten crear modelos de contornos suaves a partir de un diseño más tosco. Esto permite a los diseñadores trabajar con modelos de menor número de vértices sin tener que renunciar luego a un acabado suave a la hora de *renderizar*. Sin embargo nuestro modelo tenía ya un nivel de detalle tan extraordinario a nivel de vértices, que las herramientas de suavizado a penas eran necesarias.

En los ejercicios iniciales que realizamos para aprender a utilizar Blender aprendimos a usar estas herramientas, así que llegados a este punto tuvimos que decidir si continuar con el modelo detallado o crear uno propio.

Estuvimos sopesando las ventajas e inconvenientes de ambas opciones, pues era una decisión difícil. Por una parte queríamos conservar el aspecto del modelo detallado y no perder las horas invertidas ya en él. Pero el trabajo de pesado era muy tedioso, requería mucho tiempo y no teníamos la certeza de obtener el resultado deseado a tiempo. De modo que al final decidimos crear un modelo más simple y aplicarle luego las técnicas de suavizado.

Otras razones que justificaban esta decisión fueron que conceptualmente no es necesario ver un modelo detallado para entender un ejercicio e incluso un modelo más básico evita que los usuarios se pierdan en detalles insignificantes. Además nos gustaba la idea de trabajar con un modelo diseñado por nosotras, ahora que habíamos aprendido tantas cosas sobre el modelado en 3D que, de otro modo, no habríamos podido demostrar en el proyecto. Finalmente, un modelo más simple evitaba los problemas de pesado, puesto que nuestro nuevo diseño iba a carecer de articulaciones, lo que nos hacía ganar mucho tiempo. La ausencia de detalles como dedos y articulaciones también facilitaba los problemas de posado, de modo que decidimos que podíamos prescindir del IK-Handle de Maya.

De todos modos decidimos incluir imágenes del modelo detallado en la base de datos, ya no para mostrar movimientos sino para explicar las zonas afectadas por cada ejercicio. Esto nos pareció un compromiso bueno, puesto que de este modo el modelo detallado no desaparecía completamente del proyecto.

1.9.10.Planteamiento final

Una vez que habíamos decidido crear nuestro propio modelo en Blender, puesto que teníamos mucha más soltura con esta herramienta, nos pusimos manos a la obra. Teníamos volver a realizar todos los pasos necesarios para una animación y lo hicimos del siguiente modo:

Creación de la malla

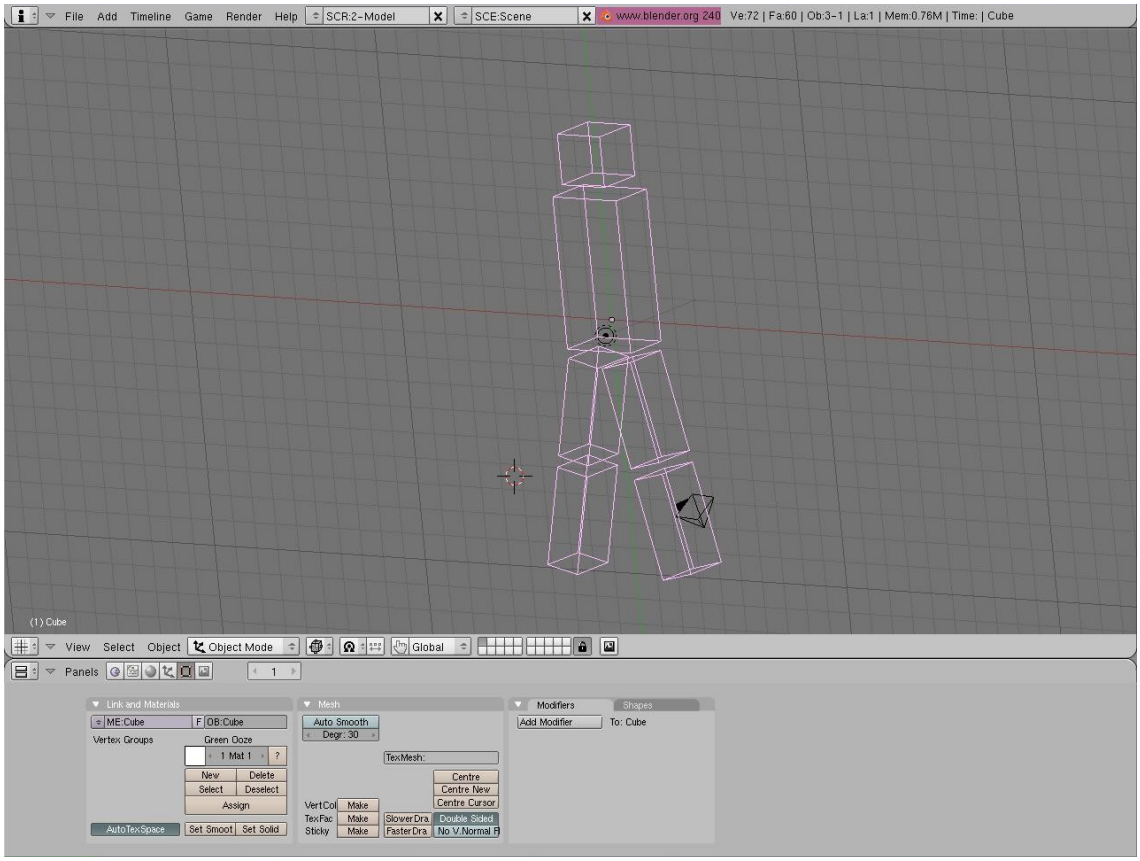
Creamos nuestro modelo a partir del cubo que Blender tiene por defecto en su pantalla inicial. Para ello seleccionamos vértices, los duplicamos, los movimos y los reordenamos en el espacio. Utilizamos mucho la herramienta "extrude" así como rotaciones, movimientos a lo largo de los ejes, espejados, bloqueos, etcétera

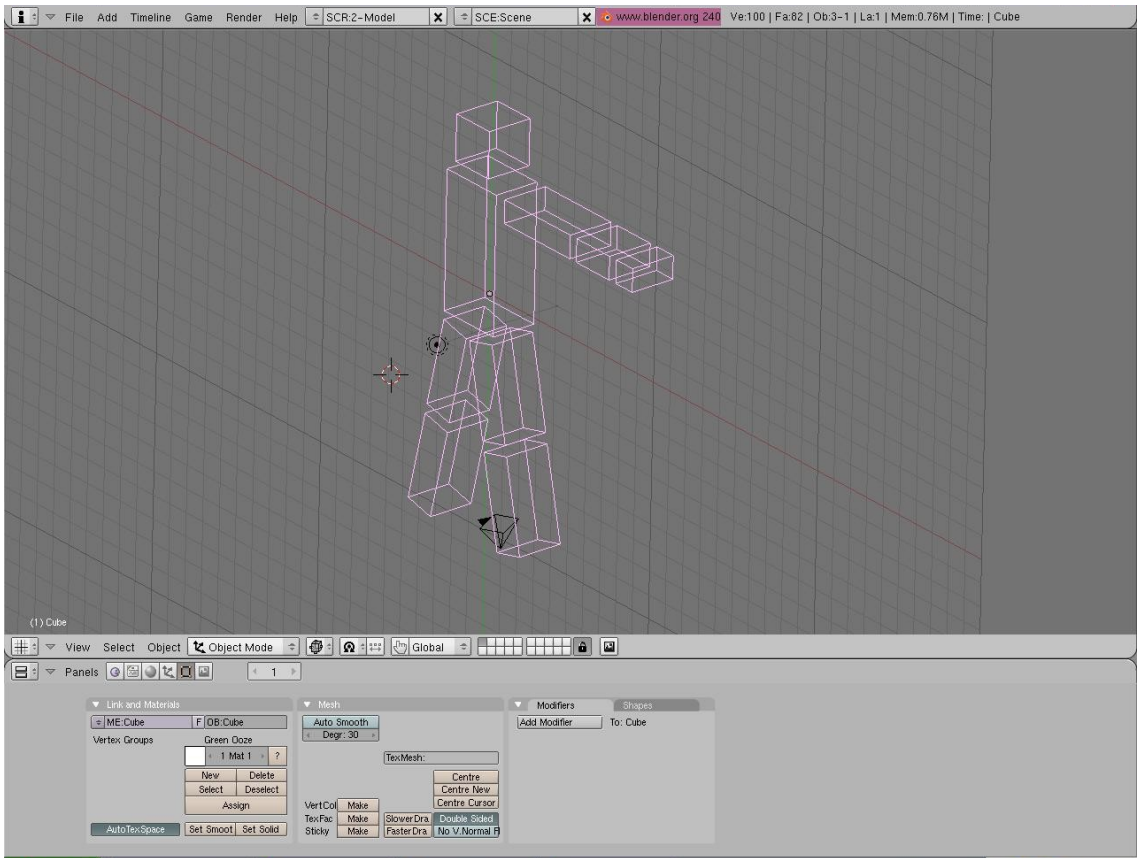
Nuestro nuevo modelo debía carecer de articulaciones, así que brazos, piernas, pies, manos, tronco y cabeza debían ser elementos no conexos, pero parte de un mismo objeto. Esto lo conseguimos manipulando los cubos de la forma anteriormente descrita así como utilizando algunas de las formas que Blender pone a disposición del diseñador. Estas formas fueron esferas para cabeza y ojos, así como un cilindro que modificamos significativamente para obtener la sonrisa de nuestro muñeco.

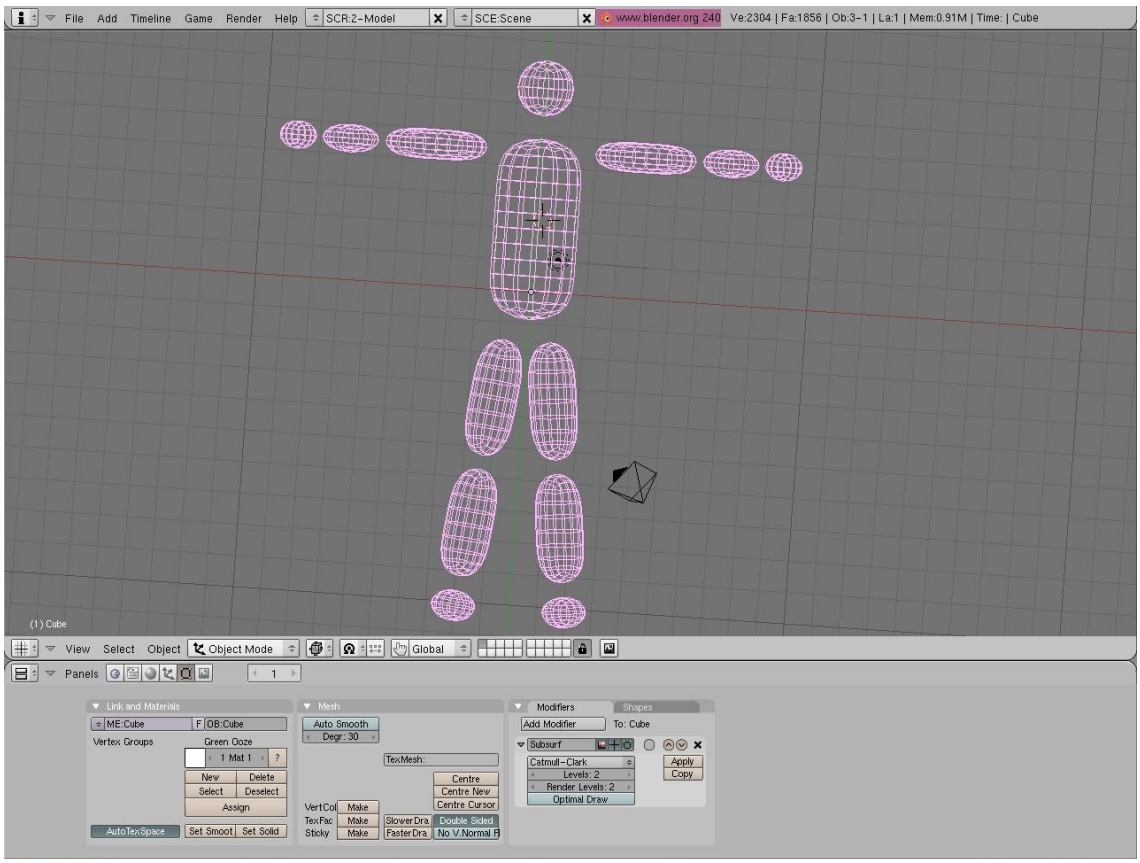
Finalmente teníamos un modelo sencillo y manejable que servía para explicar los ejercicios. Pero su acabado era aún muy tosco, de modo que utilizamos técnicas de subdivisión y suavizado que concedieron a nuestro modelo su aspecto liso y redondeado. Una vez que ve obtuvimos esto invertimos algo más de tiempo en modelar más detalladamente la forma de ciertos elementos, dando mayor volumen a los muslos, tronco y brazos, restando volumen a la parte inferior de las espinillas, ahuecando las manos, etcétera.

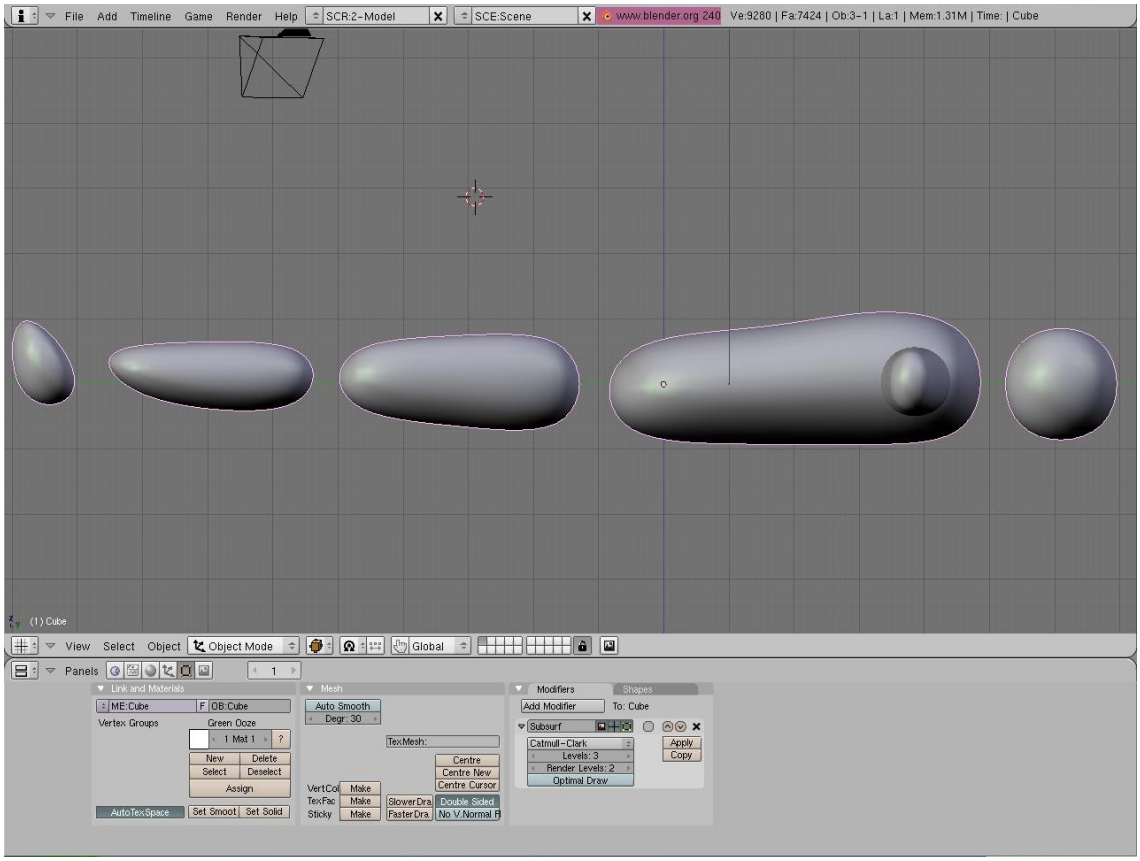
Estos detalles, al igual que los ojos y la boca tienen, más allá de la finalidad artística, el objetivo de permitir al usuario distinguir entre la imagen frontal y la de espaldas del modelo, facilitando así la comprensión del ejercicio.

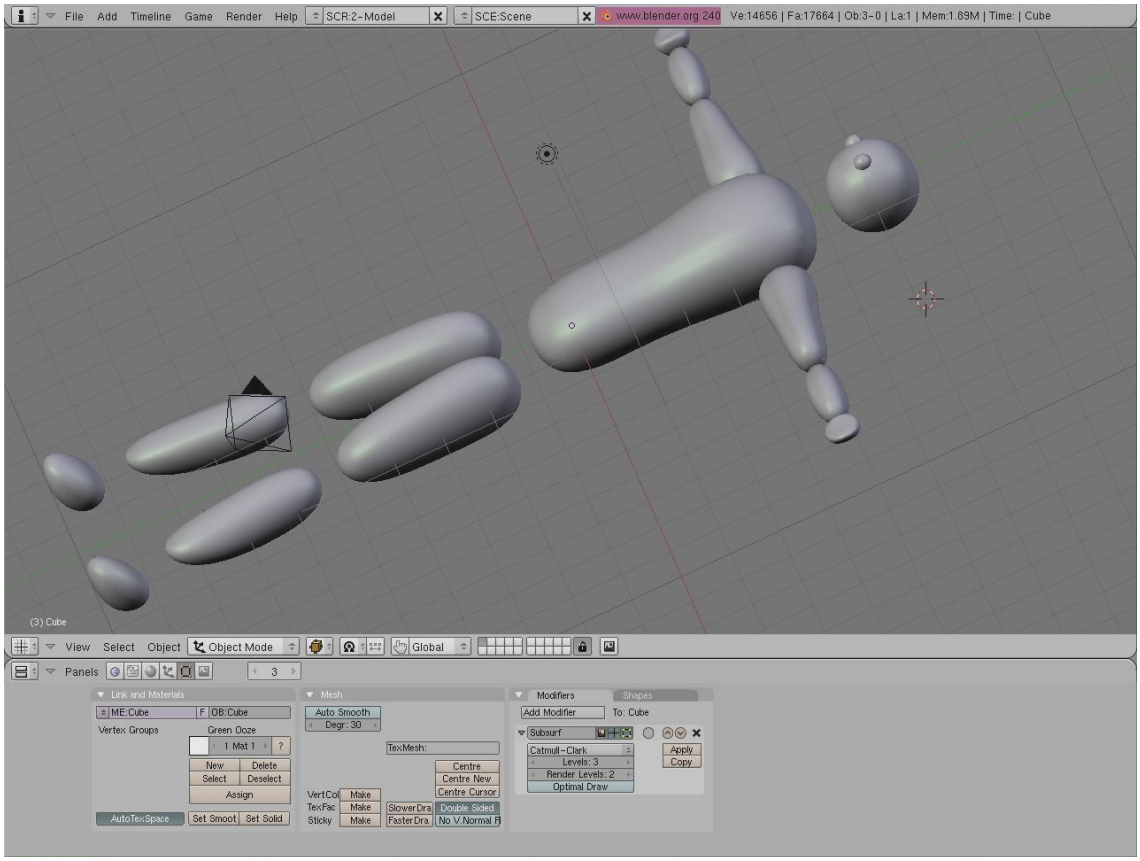
Las siguientes imágenes son capturas de pantalla de distintos momentos a lo largo del proceso de diseño.

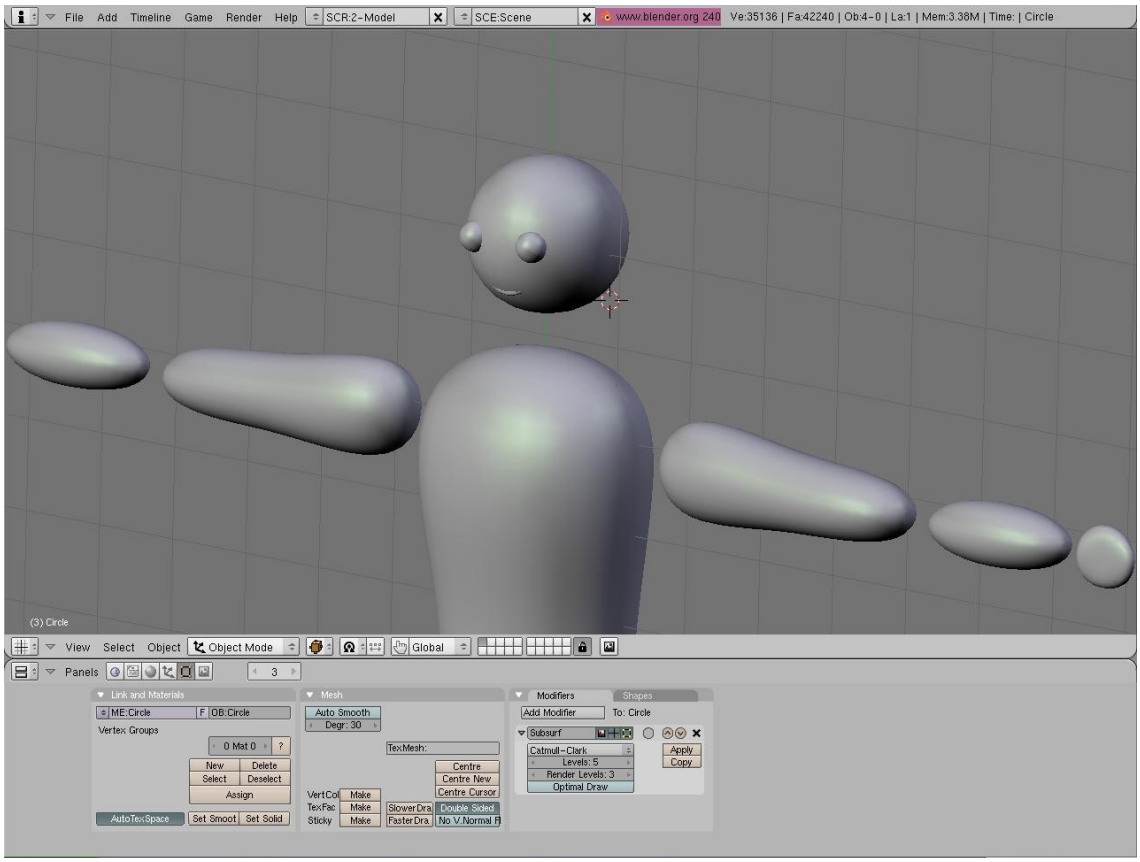

File Add Timeline Game Render Help SCR:2-Model SCE:Scene www.blender.org 240 Ve:72 | Fa:60 | Ob:3-1 | La:1 | Mem:0.76M | Time: | Cube
(1) Cube
View Select Object Object Mode Global
Panels 1
Link and Materials
ME:Cube F OB:Cube
Vertex Groups Green Ooze
1 Mat 1
New Delete
Select Deselect
Assign
AutoTexSpace Set Smoot Set Solid
Mesh
Auto Smooth
Degr: 30
TexMesh:
Centre
Centre New
Centre Cursor
VertCol Make
TexFac Make Slower Dra Double Sided
Sticky Make Faster Dra No V.Normal F
Modifiers Shapes
Add Modifier To: Cube

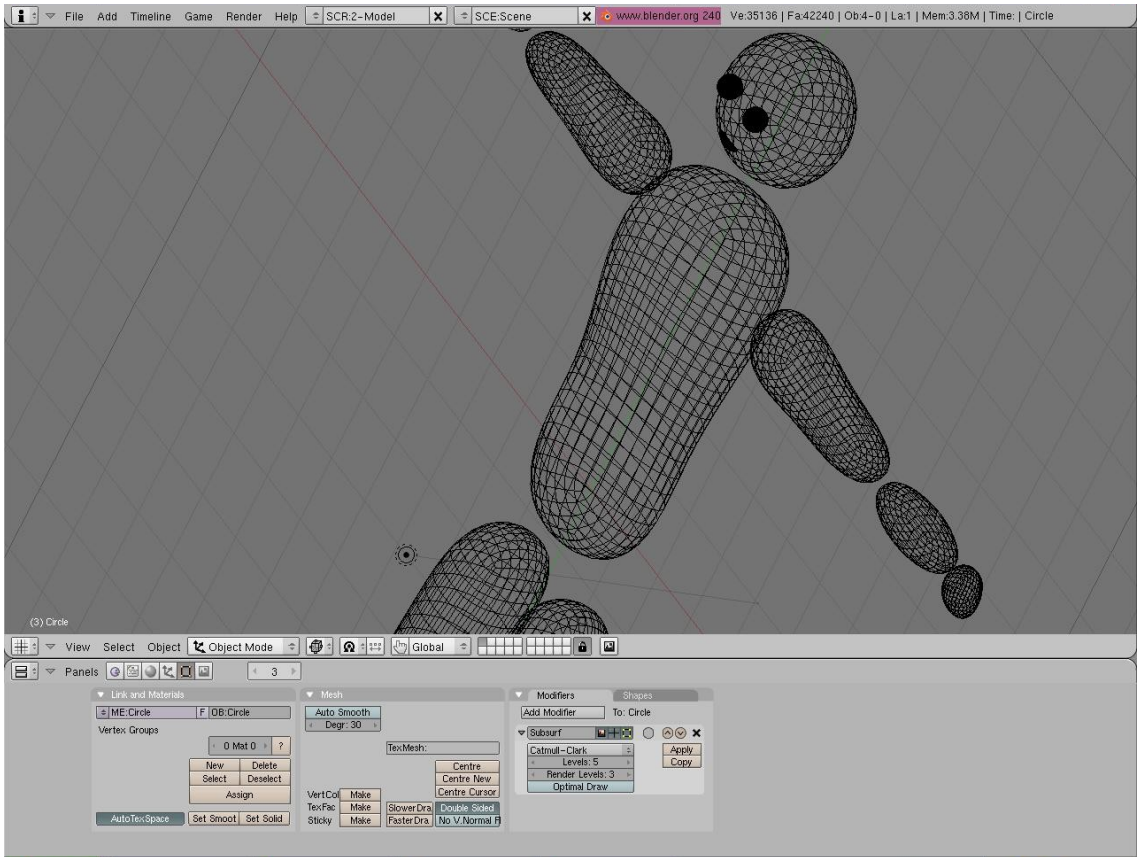

File Add Timeline Game Render Help SCR:2-Model SCE:Scene www.blender.org 240 Ve:100 | Fa:82 | Ob:3-1 | La:1 | Mem:0.76M | Time: | Cube
(1) Cube
View Select Object Object Mode Global
Panels 1
Link and Materials
ME:Cube F OB:Cube
Vertex Groups Green Ooze
1 Mat 1
New Delete
Select Deselect
Assign
AutoTexSpace Set Smoot Set Solid
Mesh
Auto Smooth
Degr: 30
TexMesh:
Centre
Centre New
Centre Cursor
VertCol Make
TexFac Make Slower Dra Double Sided
Sticky Make Faster Dra No V.Normal F
Modifiers Shapes
Add Modifier To: Cube

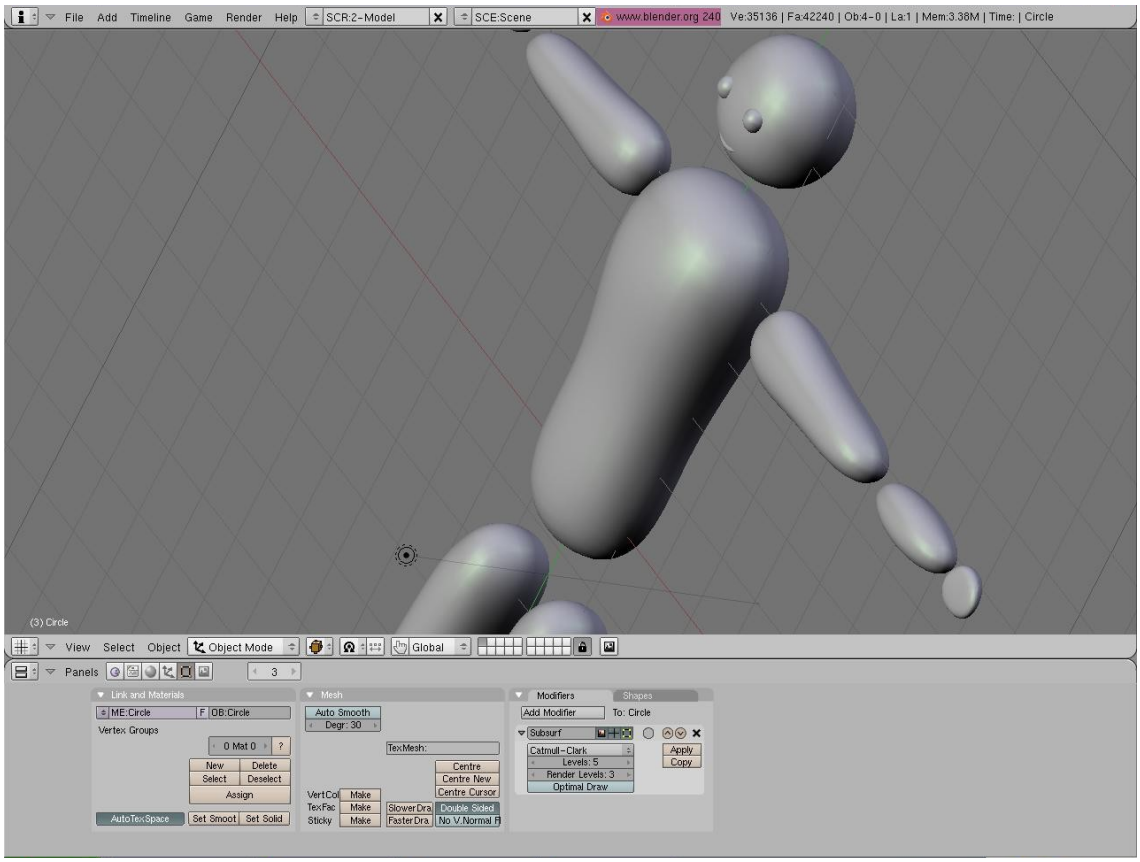

File Add Timeline Game Render Help SCR:2-Model SCE:Scene www.blender.org 240 Ve:2304 | Fa:1656 | Ob:3-1 | La:1 | Mem:0.91M | Time: | Cube
(1) Cube
View Select Object Object Mode Global
Panels
Link and Materials
ME:Cube F OB:Cube
Vertex Groups Green Ooze
1 Mat 1
New Delete
Select Deselect
Assign
AutoTexSpace Set Smoot Set Solid
Mesh
Auto Smooth
Degr: 30
TexMesh:
Centre
Centre New
Centre Cursor
VertCol Make
TexFac Make
Sticky Make
Slower Dra Double Sided
Faster Dra No V.Normal F
Modifiers Shapes
Add Modifier To: Cube
Subsurf
Catmull-Clark
Levels: 2
Render Levels: 2
Optimal Draw
Apply
Copy


File Add Timeline Game Render Help SCR:2-Model SCE:Scene www.blender.org 240 Ve:9280 | Fa:7424 | Ob:3-1 | La:1 | Mem:1.31M | Time: | Cube
(1) Cube
View Select Object Object Mode Global
Panels
Link and Materials
ME:Cube F OB:Cube
Vertex Groups Green Ooze
1 Mat 1
New Delete
Select Deselect
Assign
AutoTexSpace Set Smoot Set Solid
Mesh
Auto Smooth
Degr: 30
TexMesh:
Centre
Centre New
Centre Cursor
VertCol Make
TexFac Make
Sticky Make
Slower Dra Double Sided
Faster Dra No V.Normal F
Modifiers Shapes
Add Modifier To: Cube
Subsurf
Catmull-Clark
Levels: 3
Render Levels: 2
Optimal Draw
Apply
Copy

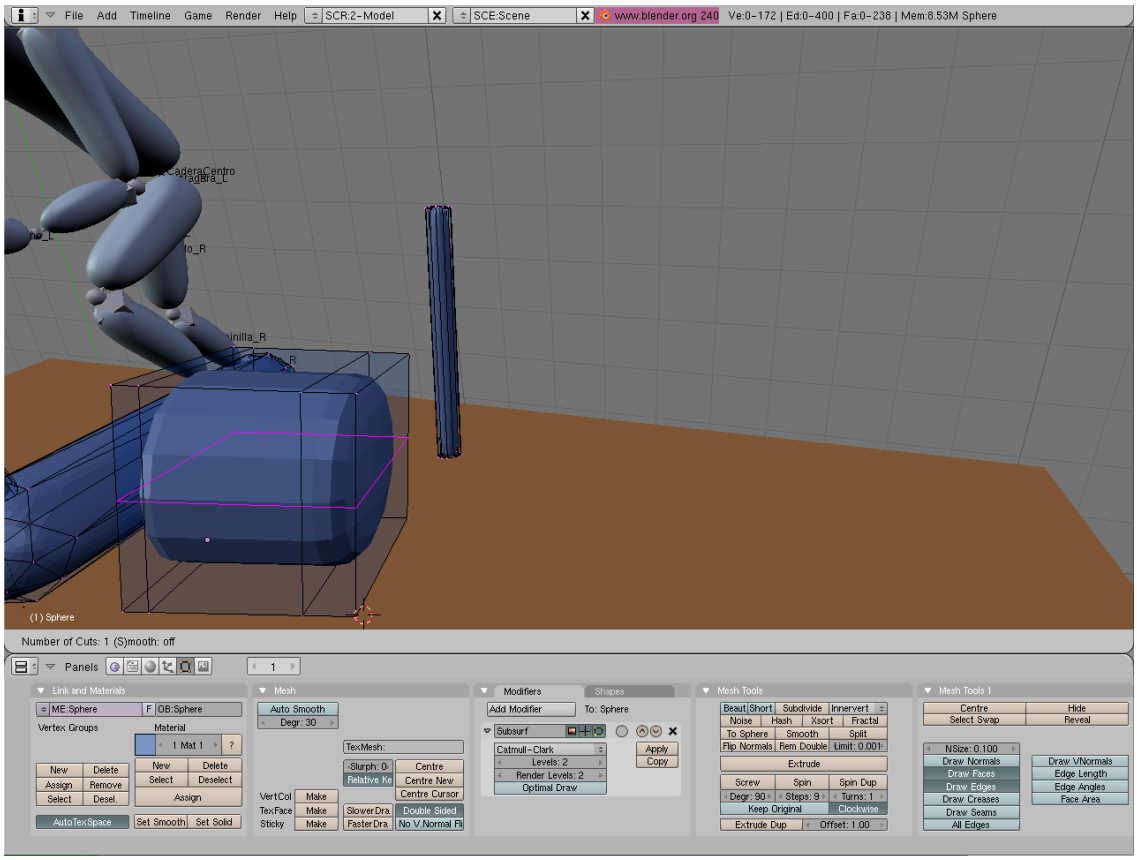

File Add Timeline Game Render Help SCR:2-Model SCE:Scene www.blender.org 240 Ve:14656 | Fa:17664 | Ob:3-0 | La:1 | Mem:1.89M | Time: | Cube
(3) Cube
View Select Object Object Mode Global
Panels 3
Link and Materials
ME:Cube OB:Cube
Vertex Groups
Green Ooze
1 Mat 1
New Delete
Select Deselect
Assign
AutoTexSpace Set Smoot Set Solid
Mesh
Auto Smooth
Degr: 30
TexMesh:
Centre
Centre New
Centre Cursor
VertCol Make
TexFac Make
Sticky Make
Slower Dra
Faster Dra
Double Sided
No V.Normal Fl
Modifiers
Shapes
Add Modifier
To: Cube
Subsurf
Catmull-Clark
Levels: 3
Render Levels: 2
Optimal Draw
Apply
Copy

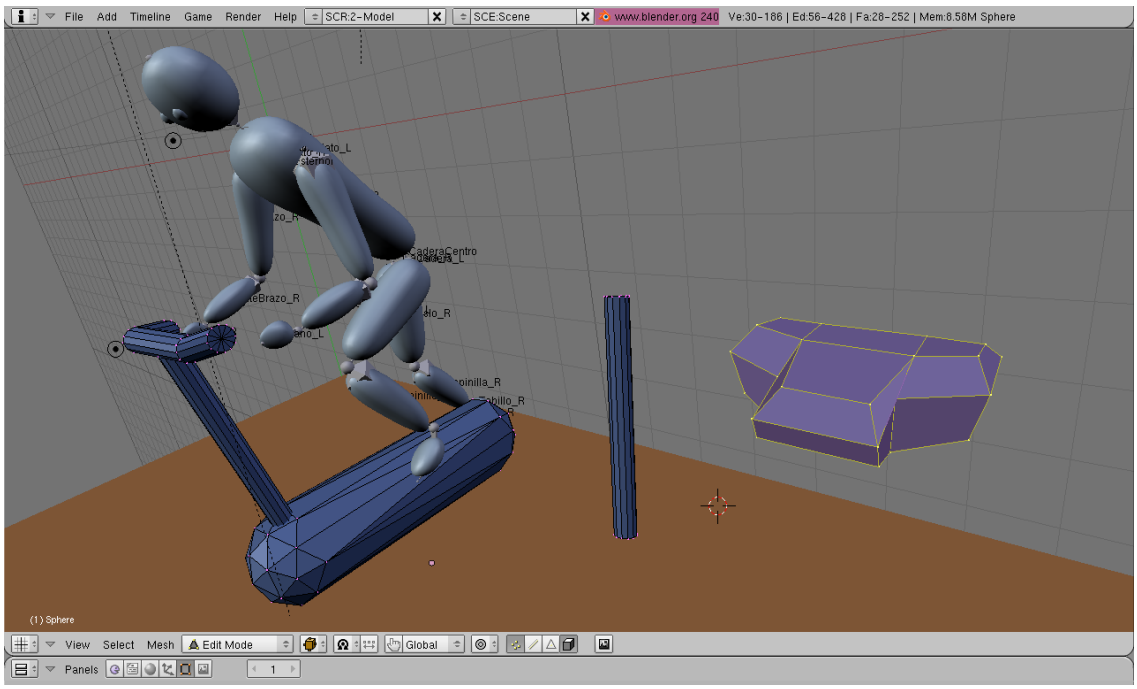

File Add Timeline Game Render Help SCR:2-Model SCE:Scene www.blender.org 240 Ve:35136 | Fa:42240 | Ob:4-0 | La:1 | Mem:3.38M | Time: | Circle
(3) Circle
View Select Object Object Mode Global
Panels 3
Link and Materials
ME:Circle OB:Circle
Vertex Groups
0 Mat 0
New Delete
Select Deselect
Assign
AutoTexSpace Set Smoot Set Solid
Mesh
Auto Smooth
Degr: 30
TexMesh:
Centre
Centre New
Centre Cursor
VertCol Make
TexFac Make
Sticky Make
Slower Dra
Faster Dra
Double Sided
No V.Normal Fl
Modifiers
Shapes
Add Modifier
To: Circle
Subsurf
Catmull-Clark
Levels: 5
Render Levels: 3
Optimal Draw
Apply
Copy

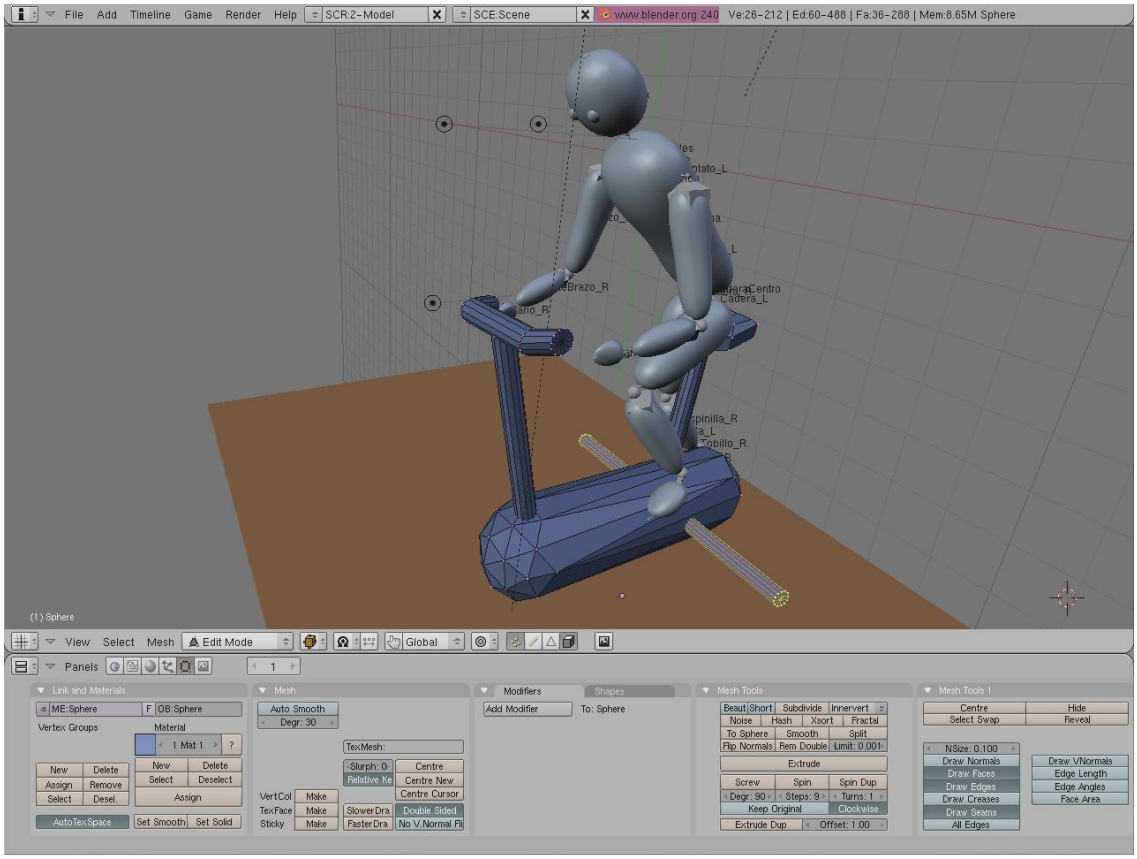

File Add Timeline Game Render Help SCR:2-Model SCE:Scene www.blender.org 240 Ve:35136 | Fa:42240 | Ob:4-0 | La:1 | Mem:3.38M | Time: | Circle
(3) Circle
View Select Object Object Mode Global
Panels 3
Link and Materials
ME:Circle F OB:Circle
Vertex Groups
0 Mat 0
New Delete
Select Deselect
Assign
AutoTexSpace Set Smoot Set Solid
Mesh
Auto Smooth
Degr: 30
TexMesh:
Centre
Centre New
Centre Cursor
VertCol Make
TexFac Make
Sticky Make
Slower Dra Double Sided
Faster Dra No V.Normal Fl
Modifiers Shapes
Add Modifier To: Circle
Subsurf
Catmull-Clark
Levels: 5
Render Levels: 3
Optimal Draw
Apply
Copy

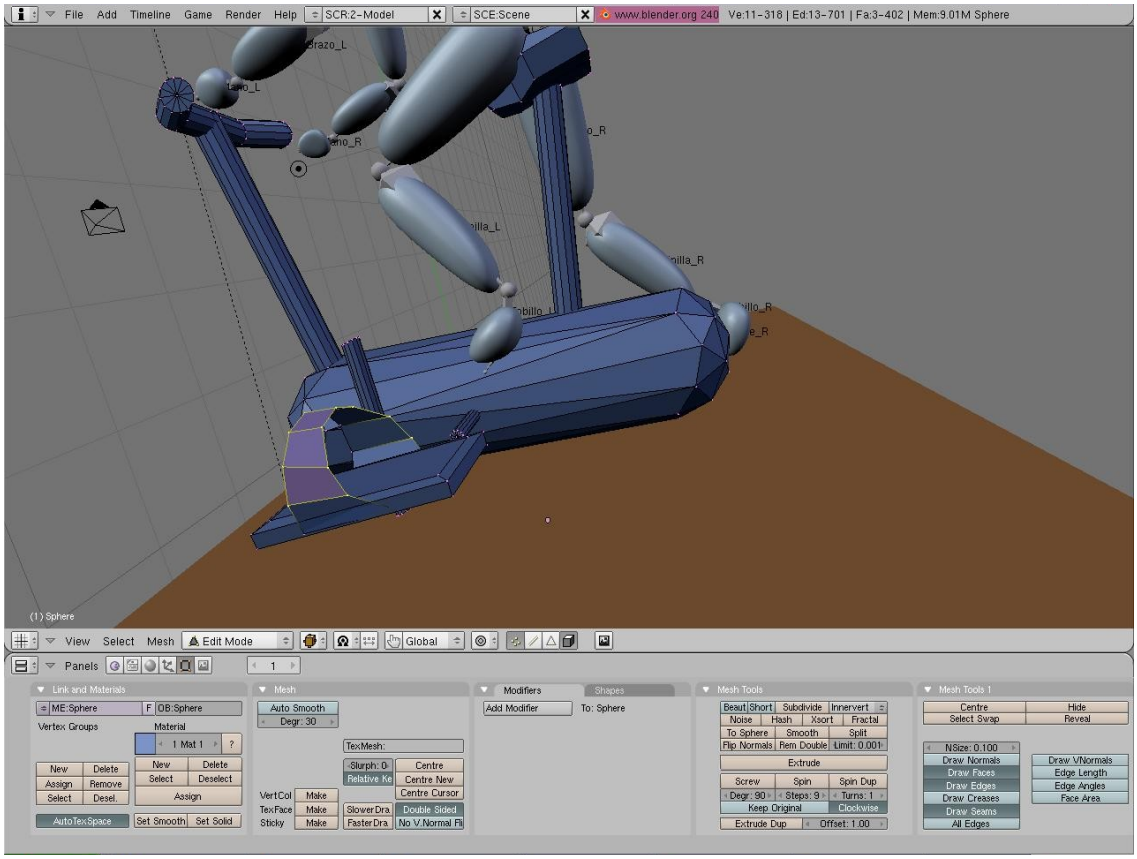

File Add Timeline Game Render Help SCR:2-Model SCE:Scene www.blender.org 240 Ve:35136 | Fa:42240 | Ob:4-0 | La:1 | Mem:3.38M | Time: | Circle
(3) Circle
View Select Object Object Mode Global
Panels 3
Link and Materials
ME:Circle F OB:Circle
Vertex Groups
0 Mat 0
New Delete
Select Deselect
Assign
AutoTexSpace Set Smoot Set Solid
Mesh
Auto Smooth
Degr: 30
TexMesh:
Centre
Centre New
Centre Cursor
VertCol Make
TexFac Make
Sticky Make
Slower Dra Double Sided
Faster Dra No V.Normal Fl
Modifiers Shapes
Add Modifier To: Circle
Subsurf
Catmull-Clark
Levels: 5
Render Levels: 3
Optimal Draw
Apply
Copy

Más adelante también resultó necesario crear aparatos de gimnasia para el modelo. Hicimos esto con las mismas técnicas con que hicimos el modelo anterior. Las siguientes imágenes son también pasos intermedios de ese proceso de creación.






File Add Timeline Game Render Help SCR:2-Model SCE:Scene www.blender.org 240 Ve:26-212 | Ed:60-488 | Fa:36-288 | Mem:8.65M Sphere
(1) Sphere
View Select Mesh Edit Mode Global
Panels
Link and Materials
Mesh
Modifiers
Shapes
Mesh Tools
Mesh Tools 1

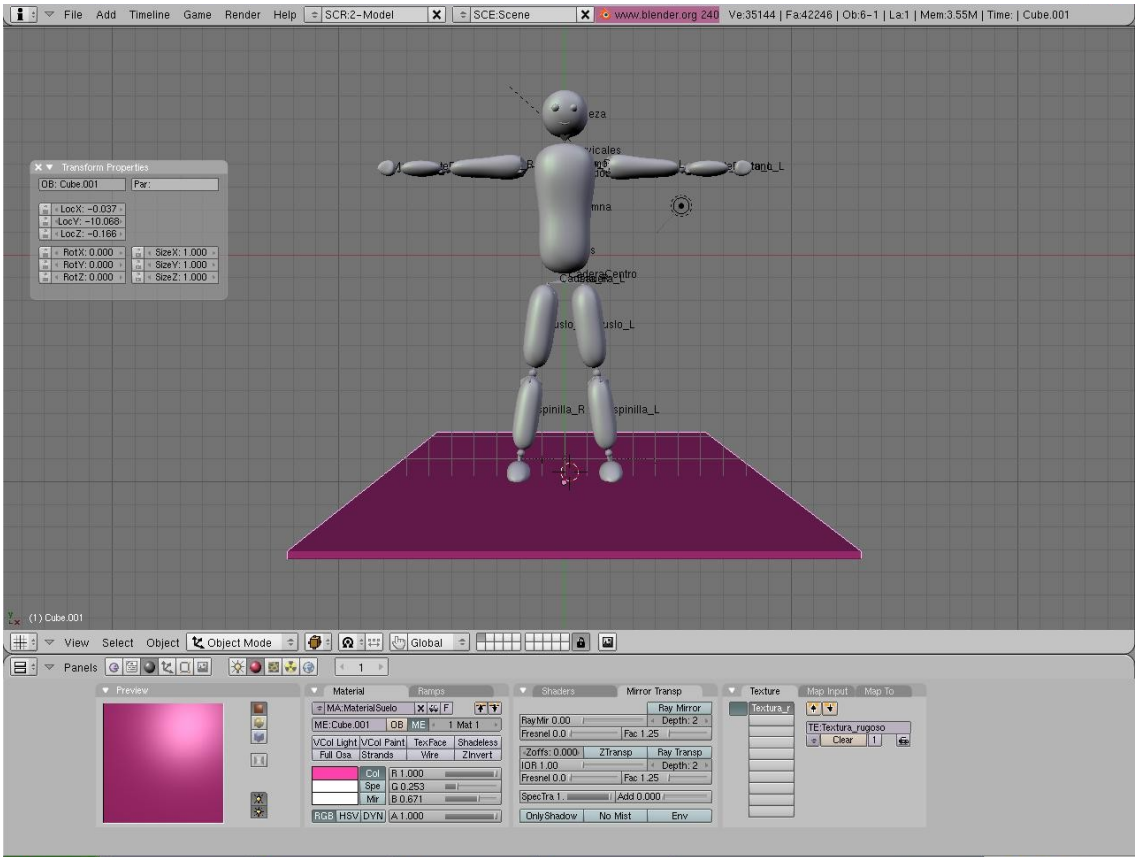

File Add Timeline Game Render Help SCR:2-Model SCE:Scene www.blender.org 240 Ve:11-318 | Ed:13-701 | Fa:3-402 | Mem:9.01M Sphere
(1) Sphere
View Select Mesh Edit Mode Global
Panels
Link and Materials
Mesh
Modifiers
Shapes
Mesh Tools
Mesh Tools 1

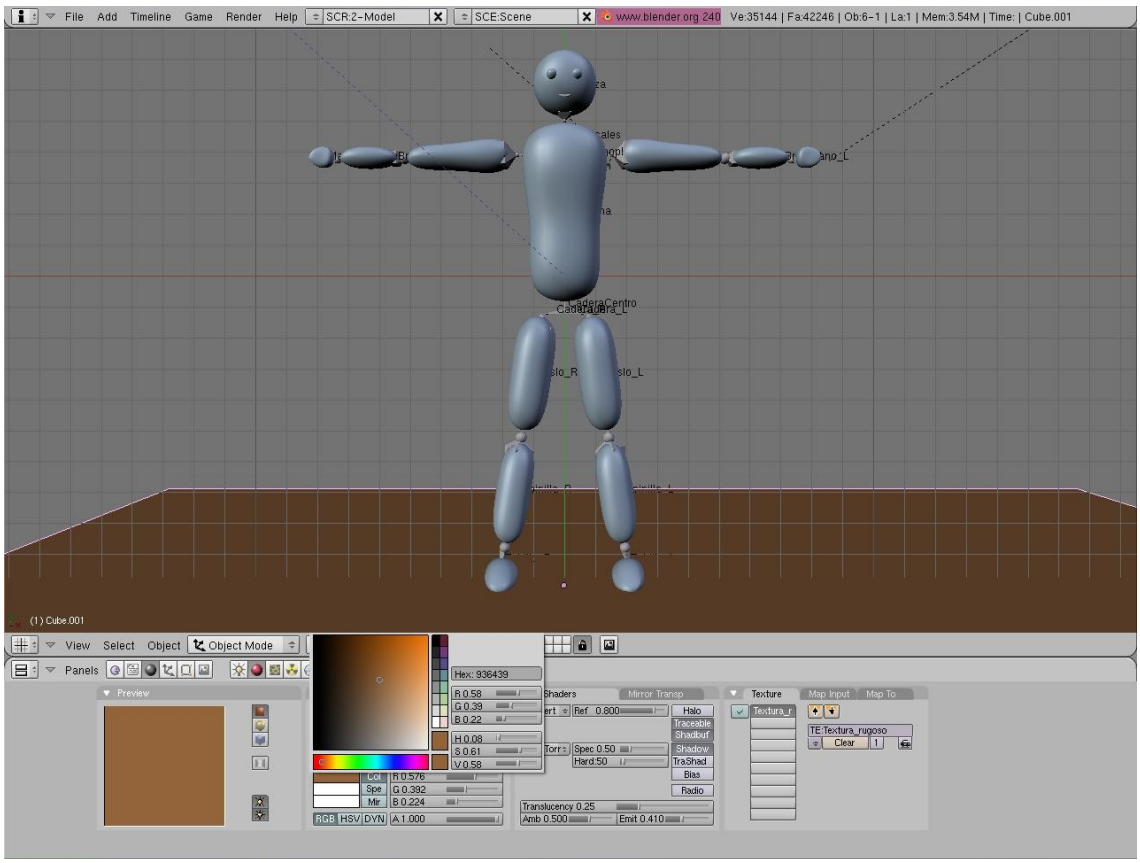
Texturas y colores

Cuando estuvimos satisfechas con el aspecto final de nuestro modelo empezamos a aplicarle texturas y color. También creamos un suelo, porque consideramos que era importante para el usuario tener una referencia dentro del espacio 3D, al que también hubo que dar formato.

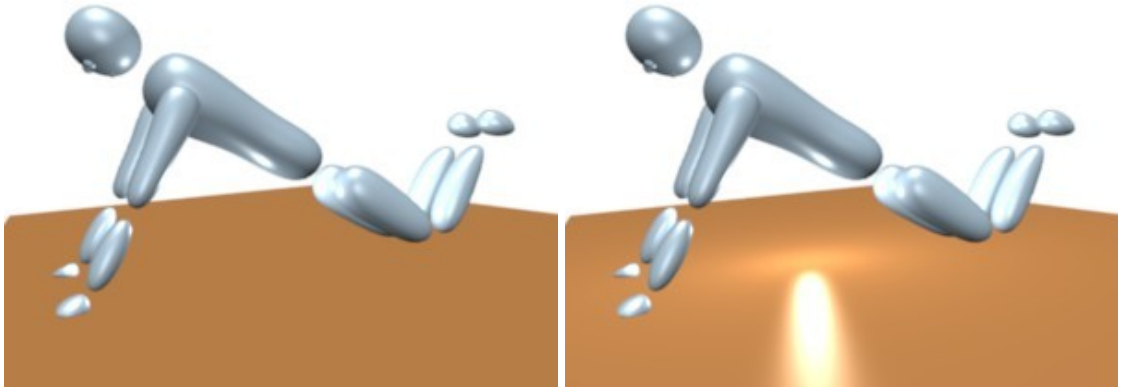
Aunque hicimos esto tras crear las mallas, los colores sufrieron algunas modificaciones más adelante, puesto que no nos gustaba cómo quedaba una vez realizado el *renderizado*.

Las imágenes siguientes reflejas algunos momentos en los que estábamos probando distintos formatos.

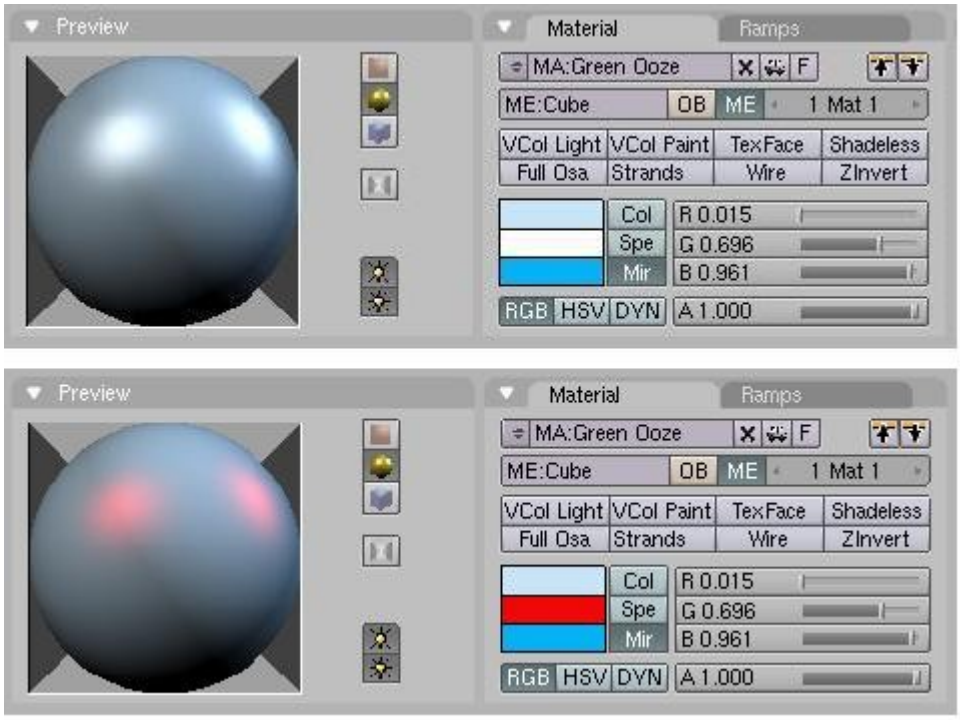




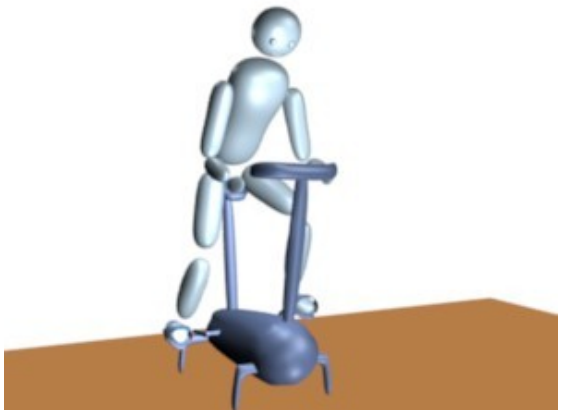
La versión final tiene un fondo blanco y mallas con las siguientes características.
El suelo es de color Hex: 936439 (marrón) y no refleja la luz, puesto. La diferencia entre crear un material reflectante o no se ve en las siguientes imágenes.



El muñeco es de color Hex: C8EF6 con reflejos en blanco. En Blender se puede regular hasta el color de las sombras o reflejos. La próxima imagen es un montaje de dos recortes del panel de botones en el que se ven los reflejos una vez en blanco y la otra en rojo. (La imagen superior corresponde a la configuración de nuestro muñeco.)



Posteriormente creamos aparatos de deporte como una cinta de correr, una bicicleta o pesas. Todos ellos tienen también textura reflectante y son de color azul oscuro (Hex: 7C8DE1), aunque la percepción del color cambia según la intensidad de la luz y el ángulo con el que caen los rallos (comportamiento semejante al del mundo real).

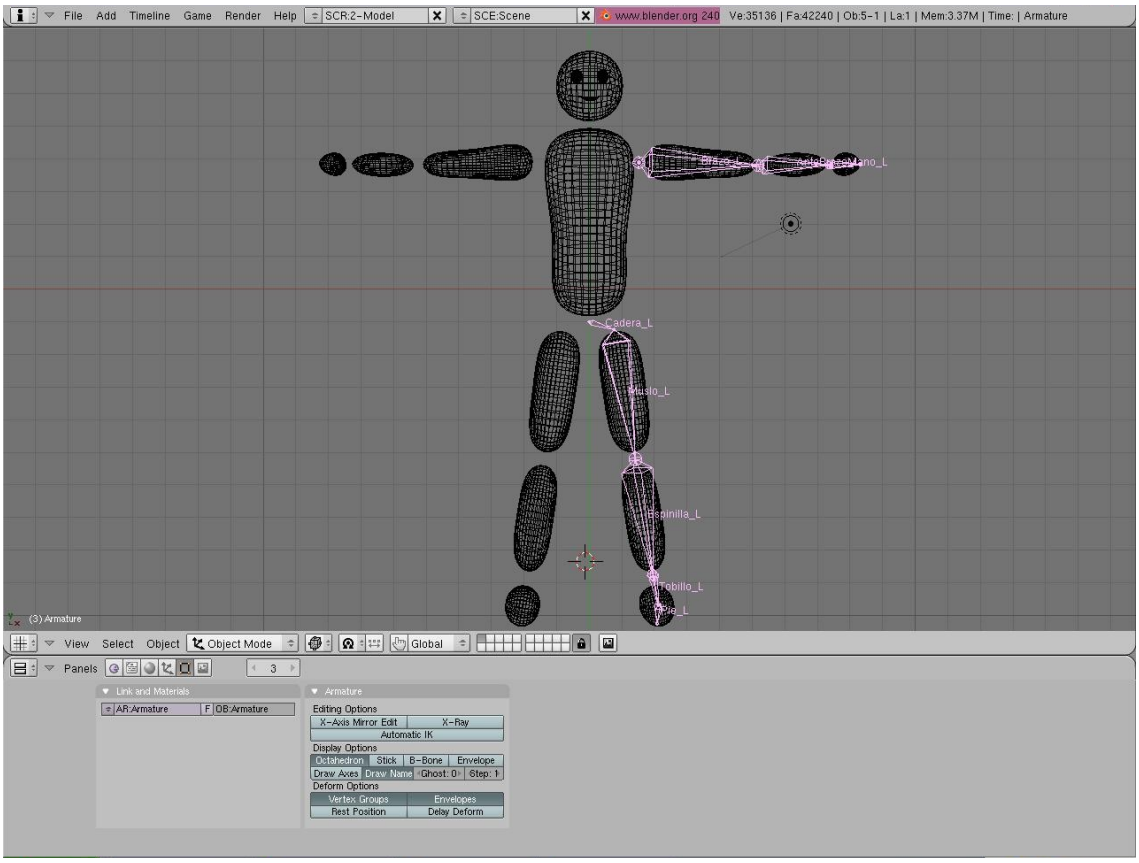


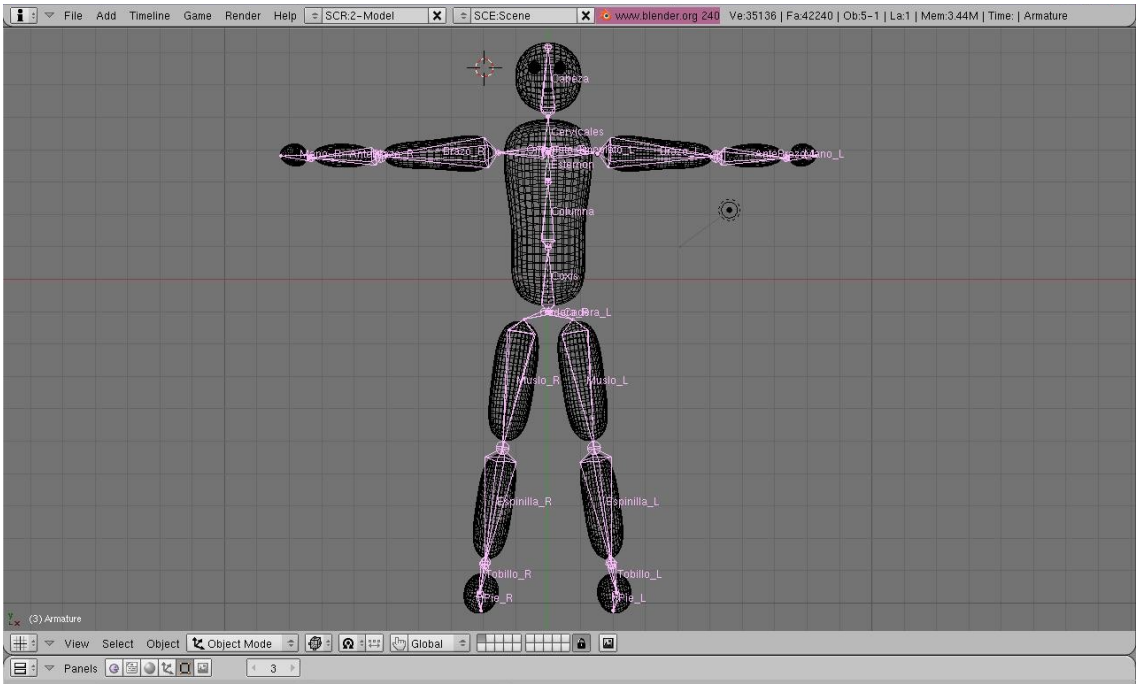
Una vez concluido esto, el resto de los pasos eran iguales a los que realizamos con nuestro primer modelo, por lo que los explicaremos de forma más resumida.

***Rigging* (el esqueleto)**

Puesto que nuestro modelo era muy sencillo, decidimos crear una armadura que simplemente relacionara los distintos elementos que lo componían. Además, emparentamos los objetos, (ojos, boca y malla) para evitar que ninguno se quedara atrás cuando se moviera el hueso que correspondía a la cabeza. Puesto que la primera vez Blender nos dio problemas al intentar espejar huesos que luego coincidían, por ejemplo

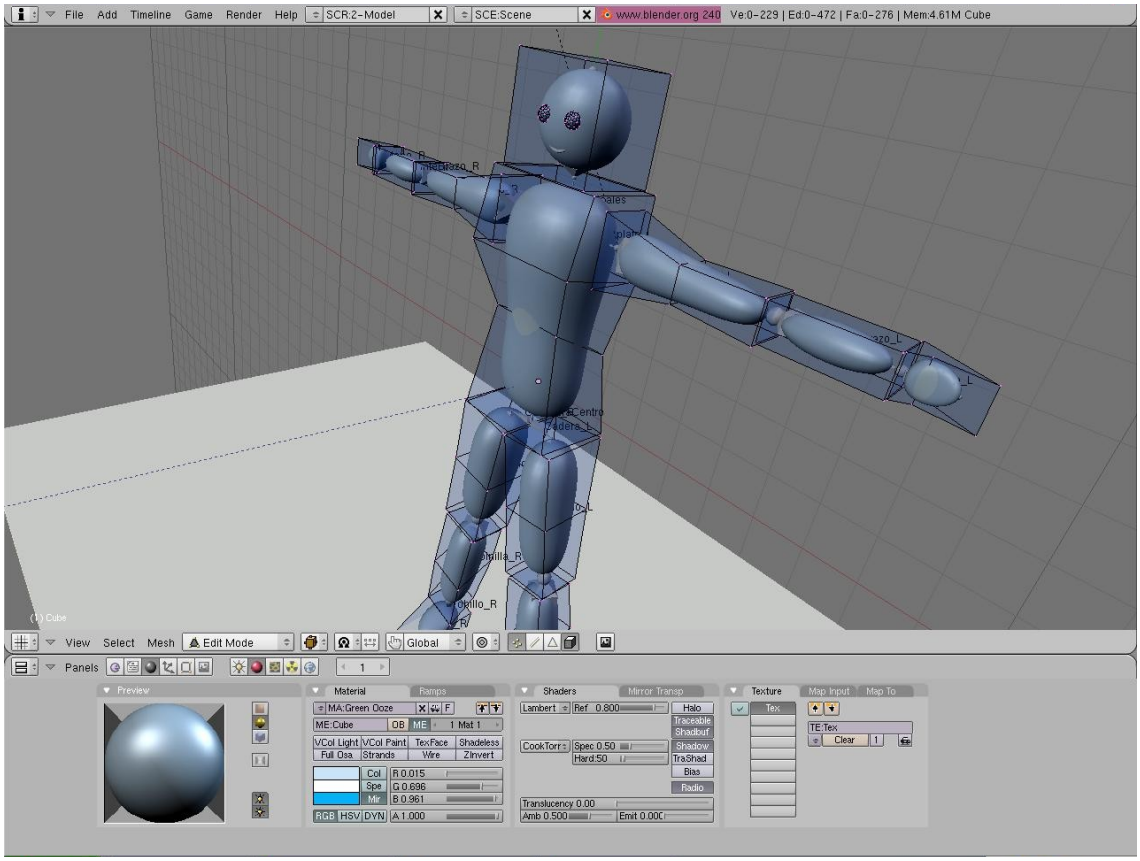
la columna y la cabeza, esta vez comenzamos el diseño de la armadura por las extremidades. En las siguientes imágenes se pueden apreciar claramente el proceso de diseño.





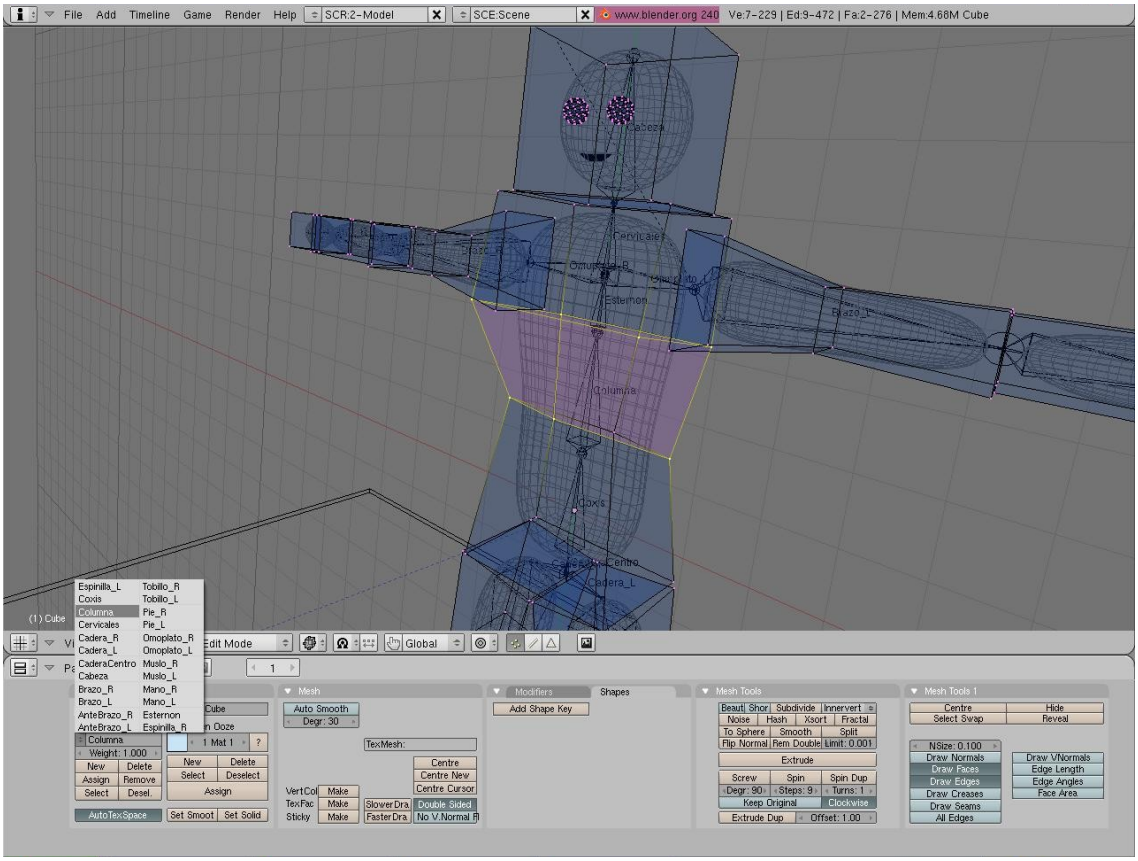
Skinning **(relación entre los vértices y la armadura)**

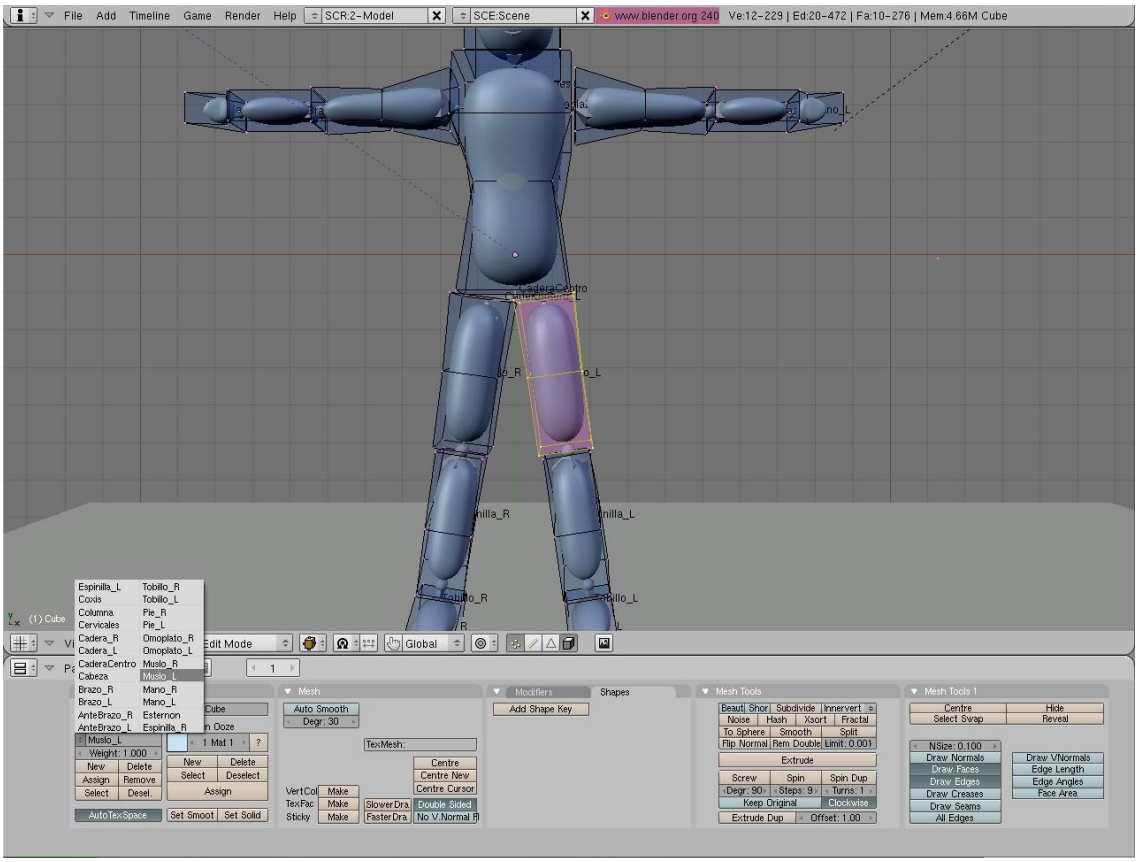
Aunque las herramientas de suavizado nos habían permitido crear un modelo de apariencia detallada, el número de vértices que lo componen y su posición seguían siendo muy sencillos. Esto es el modo en que trabajan los diseñadores 3D, puesto que facilita enormemente las distintas labores de diseño. La siguiente imagen muestra esta doble caracterización.



Además, la ausencia de articulaciones nos permitía prescindir del trabajo de pesado. De modo que realizamos el *skinning* creando nosotras mismas, desde el comienzo, los grupos de vértices y asignándoles un peso de 1 a los que queríamos que se movieran. Una vez completada esta fase, el modelo estaba ya listo para ser utilizado.

A continuación incluimos algunas capturas de pantalla que enseñan distintos momentos de este proceso. En ellas se puede ver el modelo, tanto en forma sólida como de malla, y los vértices que le componen. La opción de ver los nombres de los huesos está activada, puesto que facilita la labor de diseño. En un menú del panel de botones de elige el hueso con el que se desea trabajar y luego se utilizan los botones correspondientes para tratar (añadir, quitar, seleccionar, etcétera) los vértices correspondientes.


File Add Timeline Game Render Help SCR:2-Model SCE:Scene www.blender.org 240 Ve:7-229 | Ed:9-472 | Fa:2-276 | Mem:4.68M Cube
Cabeza
Cervicales
Esternon
Brazo_L
Columna
Coxis
Cadera_L
Edit Mode
Global
Mesh
Modifiers
Shapes
Mesh Tools
Mesh Tools 1
Add Shape Key
Extrude


File Add Timeline Game Render Help SCR:2-Model SCE:Scene www.blender.org 240 Ve:12-229 | Ed:20-472 | Fa:10-276 | Mem:4.66M Cube
CaderaCentro
Edit Mode
Global
Mesh
Modifiers
Shapes
Mesh Tools
Mesh Tools 1
Add Shape Key
Extrude

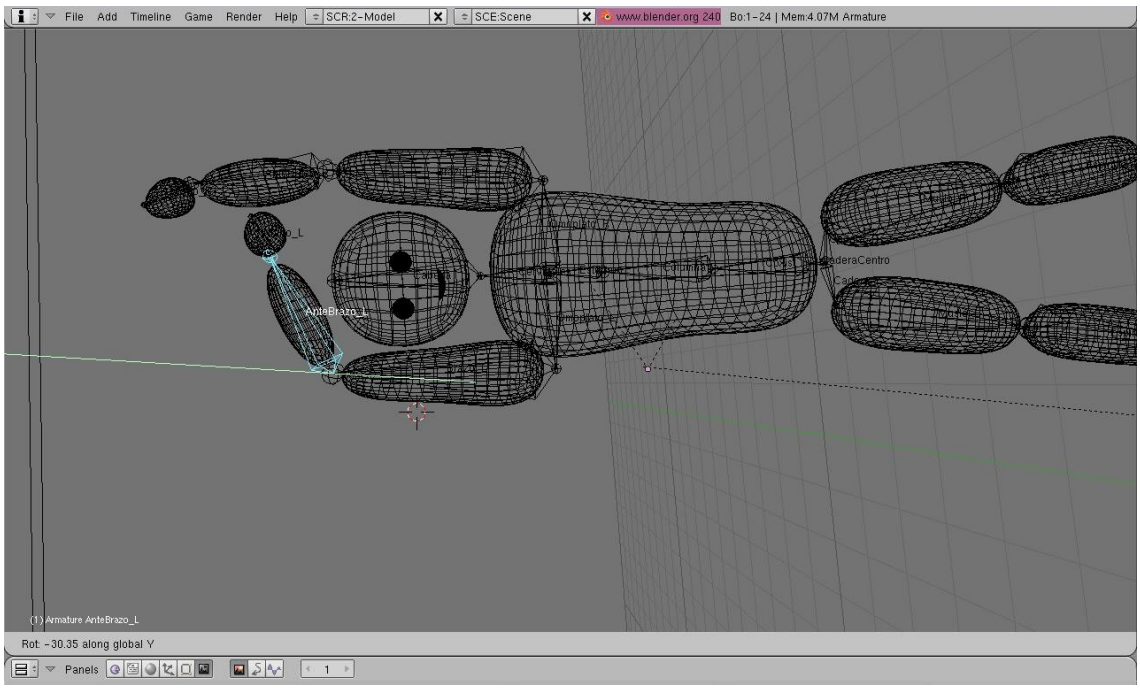
Posing

Con este nuevo diseño conseguimos evitar los problemas que aparecieron con el diseño inicial a la hora de realizar el posado. Así que ahora ya sólo nos teníamos que concentrar en crear escenas idóneas para los ejercicios y generar *renderizados* que luego pudiéramos incluir en la base de datos.

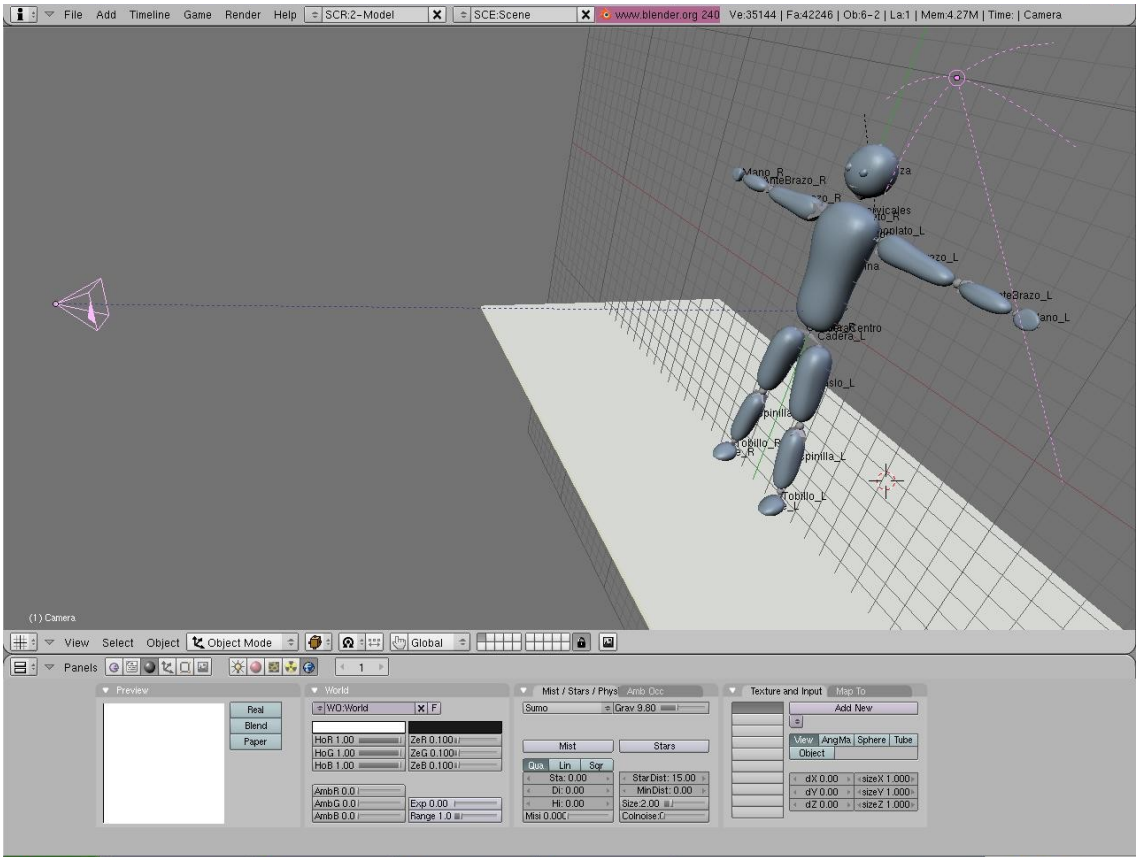
"*Posing*" sólo se refiere a la ubicación de los componentes del modelo, pero es cuando se realiza el *renderizado*, cuando se ve la necesidad de añadir más fuentes de luz, modificar su intensidad o recolocar la cámara. Esto se hace en modo objeto o edición, pero lo describimos bajo este punto, porque nosotras realizamos estos ajustes a la vez que determinábamos las poses del modelo.

La idea era colocar al modelo en posiciones representativas de ejercicios deportivos. Tomamos como referencia las ilustraciones de una reconocida revista, Sport Life [60], así como ejercicios que conocíamos por nuestra propia experiencia. Aunque no disponíamos de las ventajas del IK-Handle de Maya, descubrimos unas opciones muy interesante que facilitaron en gran medida la fase de posado. Una de las posibilidades que ofrece Blender, es bloquear el movimiento que se desea realizar a lo largo de un eje. De modo que para colocar al muñeco en una posición, lo único que teníamos que hacer era desplazar o rotar sus componentes a lo largo de los tres ejes. Para bloquear un eje basta pulsar la tecla con su nombre (x,y,z) una vez elegido el tipo de operación que se desea realizar sobre la componente. Entonces el eje aparece dibujado en el espacio, facilitando así la comprensión de lo que ocurre dentro del *viewport*.

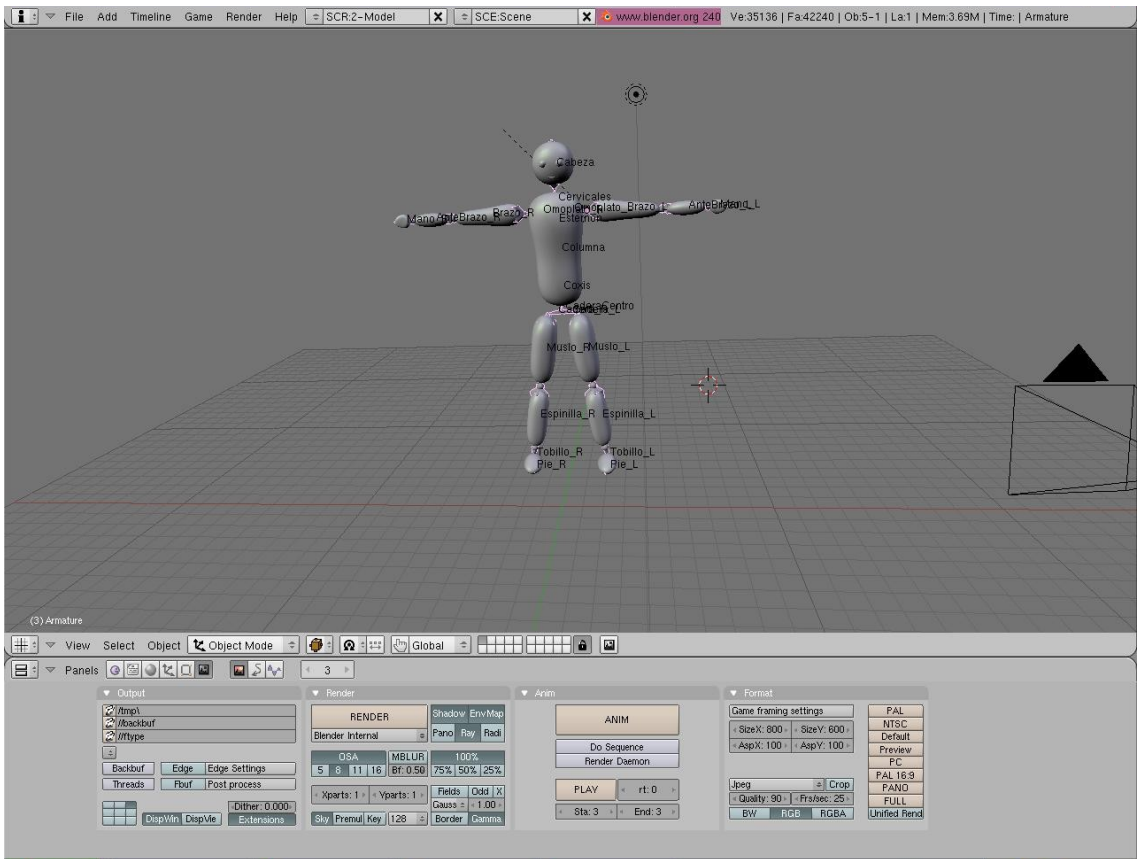
La siguiente imagen recoge esta situación. En ella se puede ver el hueso seleccionado resaltado en azul y el eje respecto al que se desea mover en color verde.



Como hemos comentado anteriormente, una escena se compone de muchos objetos. Estos pueden estar en distintas capas, lo que se suele hacer cuando el número de objetos es muy elevado. En nuestro caso optamos por dejarlos todos en la misma capa, de modo que en las próximas capturas de pantalla se pueden apreciar tanto las luces y la cámara como las mallas. Estas imágenes deberían aclarar el concepto de escena.

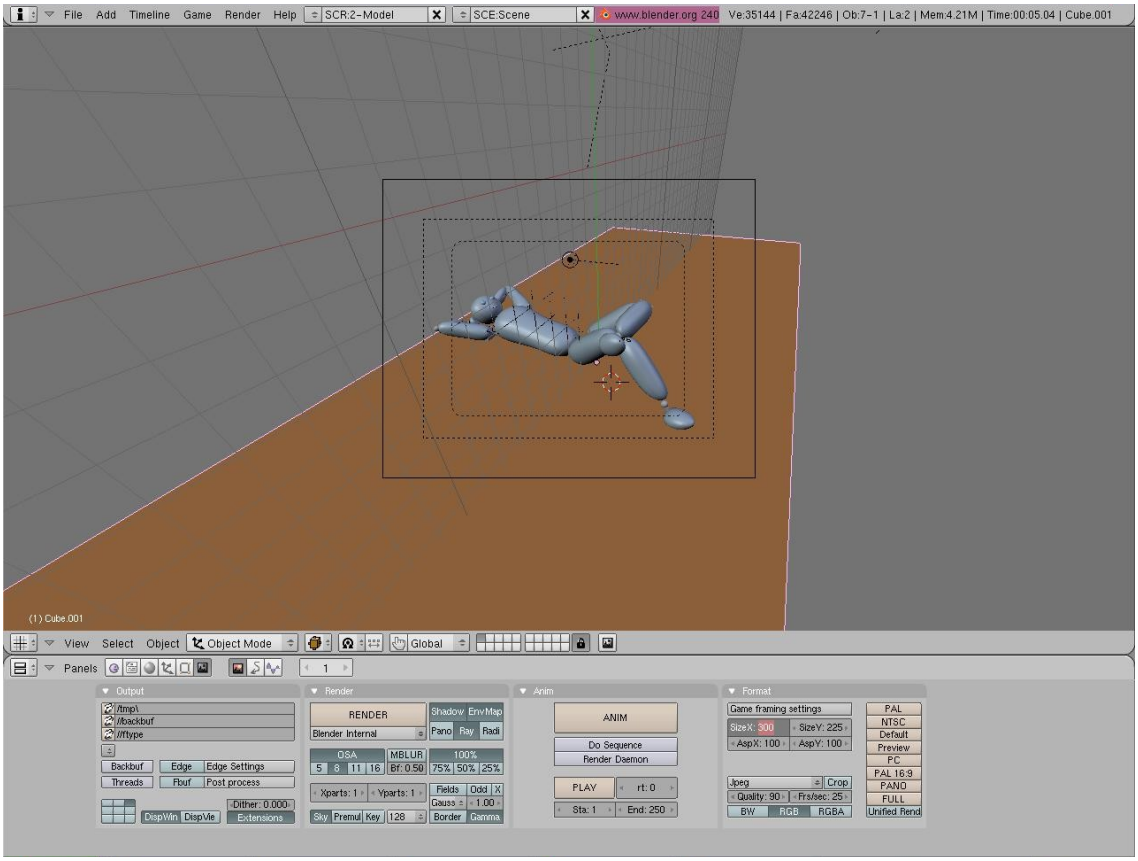


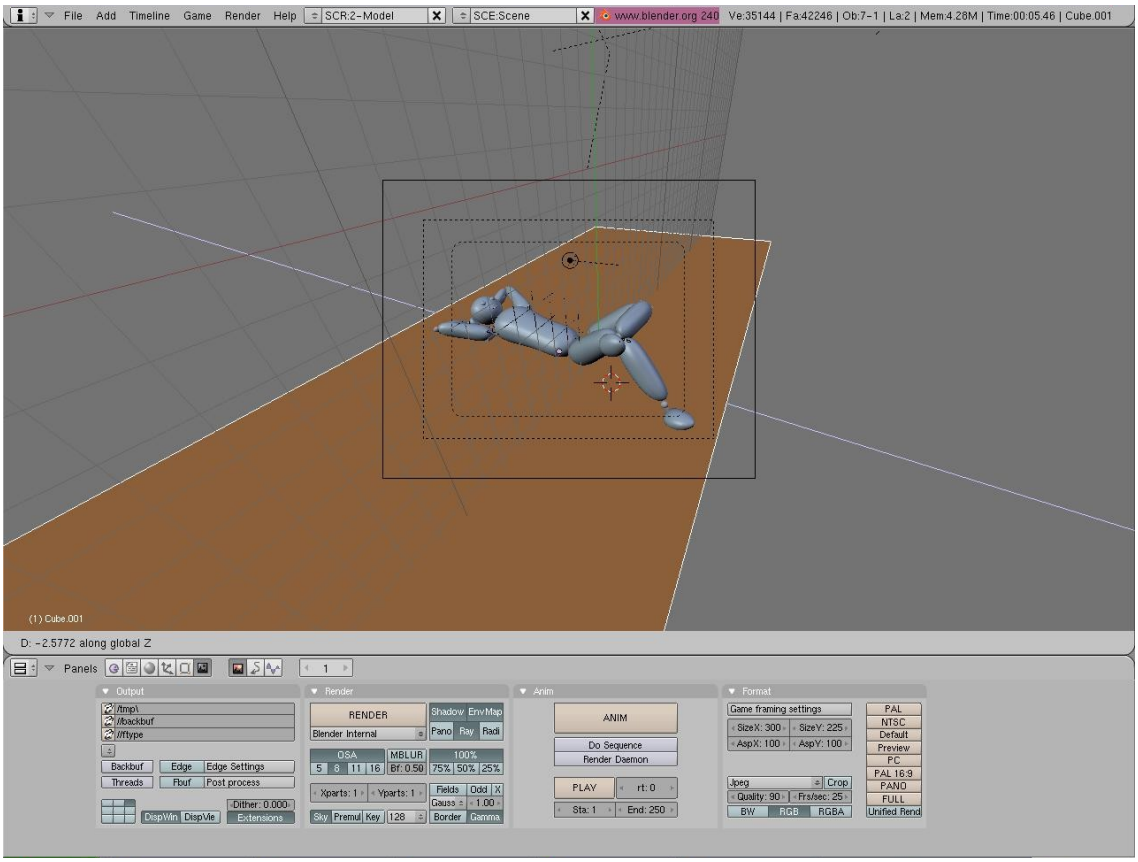
En la imagen superior, la cámara y la fuente de luz están seleccionadas. En Blender, esto significa que aparecen en color rosa. Estos elementos no se ven al hacer un *renderizado*, pero su correcta colocación es determinante para obtener una imagen 2D de calidad.



En Blender existen distintos tipos de fuentes de luz, desde focos a luces de rayos paralelos, etcétera También se puede regular su intensidad, situación e inclinación, de modo que las posibilidades son infinitas. En las dos imágenes anteriores se pueden ver distintas fuentes de luz. Dependiendo de la postura del muñeco, se creaban más sombras o zonas oscuras de modo que la colocación de las luces era algo que había que hacer de forma individualizada.

Elegir el ángulo idóneo para la cámara era algo que tampoco podíamos dejar en manos de Blender, de modo que movíamos la cámara según las características de la escena. Puesto que la cámara es un objeto más, se puede mover con el bloqueo de ejes, al igual que los componentes de la malla. Además existe la opción de ver la escena desde dentro de la cámara, sin necesidad de realizar un *renderizado* e incluso rotarla sobre ciertos ejes desde ése modo. Esto permite apreciar el recorte de la escena que supondrá el *render*, aunque no da información sobre el efecto de las luces. La siguiente imagen muestra la vista desde la cámara.

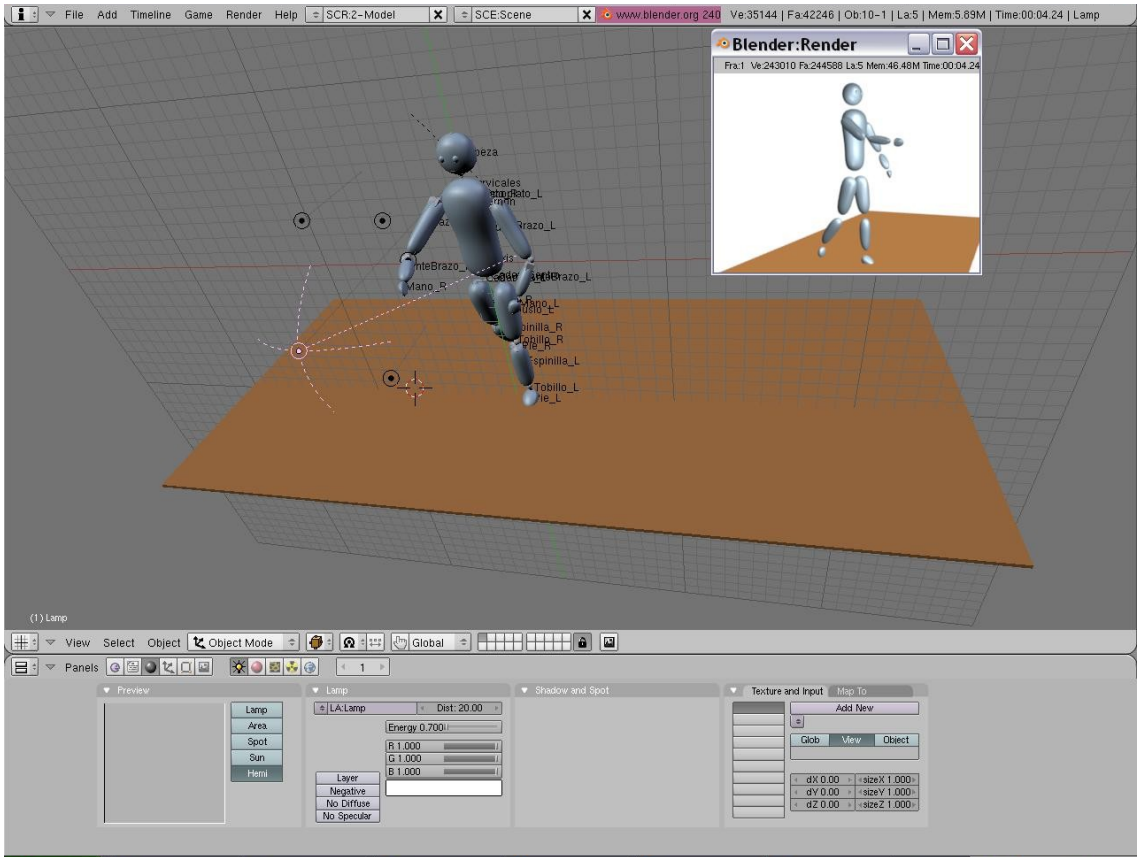

File Add Timeline Game Render Help SCR:2-Model SCE:Scene www.blender.org 240 Ve:35144 | Fa:42246 | Ob:7-1 | La:2 | Mem:4.21M | Time:00:05.04 | Cube.001
(1) Cube.001
View Select Object Object Mode Global
Panels
Output
Render
Anim
Format
RENDER
ANIM
Do Sequence
Render Daemon
PLAY


File Add Timeline Game Render Help SCR:2-Model SCE:Scene www.blender.org 240 Ve:35144 | Fa:42246 | Ob:7-1 | La:2 | Mem:4.28M | Time:00:05.46 | Cube.001
(1) Cube.001
D: -2.5772 along global Z
Panels
Output
Render
Anim
Format
RENDER
ANIM
Do Sequence
Render Daemon
PLAY

Aunque se visualice la escena desde dentro de la cámara, esto no impide que se puedan manipular los objetos del mundo. (La palabra"mundo" es terminología Blender y se utiliza para referirse al entorno en el que se ubican todos los elementos.)

Normalmente no utilizamos esta opción para manipular el modelo, pero muchas veces movimos el suelo, puesto que desde el ángulo de la cámara daba la sensación de que el modelo sobresalía. En la imagen anterior se puede ver claramente un bloque de eje a lo largo del cual habíamos movido el suelo.

Cuando considerábamos que la escena estaba completada realizábamos un *renderizado*. Las opciones del *renderizado* son muy diversas. Se puede especificar, por ejemplo, que se desea un nivel de suavizado mayor que con el que se trabaja normalmente. De este modo se ahorra tiempo de procesamiento durante el diseño, al trabajar con un modelo más tosco, sin tener que prescindir de unos *renderizados* de alta calidad. También se pueden especificar el formato y las dimensiones de la imagen que se desea obtener. Nosotras elegimos JPEG, por su nivel de compresión y uso habitual en páginas Web, y unas dimensiones de 300 x 225. La siguiente captura de pantalla muestra una escena y la imagen *renderizada* que Blender genera y coloca, a petición nuestra, en la esquina superior derecha.



Aunque se haya construido la escena con mucho cuidado, el resultado del *renderizado* no siempre es el deseado. Esto obliga a volver a la escena y retocar o añadir luces o la posición de la cámara. Esto supone un trabajo laborioso, pero que se agiliza mucho en cuanto se adquiere cierta práctica.

Una forma de agilizar este proceso es disponer de una buena organización. Muchos ejercicios son bastante similares, de modo que una vez se han colocado todos los elementos de la escena correctamente sólo hace falta mover algún componente del modelo para obtener otro ejercicio. De este modo se ahorra mucho tiempo.

Nosotras hemos clasificado nuestras imágenes en distintos niveles. Tenemos un grupo de ficheros que contienen las posiciones básicas: de pie, sentado, tumbado boca arriba, boca abajo, etcétera A un segundo nivel, tenemos posiciones básicas ya más detalladas para abdominales, ejercicios de piernas etcétera El esfuerzo inicial para crear todos estos ficheros es considerable, pero una vez se han obtenido y clasificado, lo más sencillo es buscar un ejercicio similar al que se quiere diseñar y adaptarlo. De modo que, cuantos más ejercicios se crean, más fácil resulta añadir nuevos.

Video

Hay distintas maneras de crear una animación en Blender, pero una de ellas es bastante sencilla. Para ello basta situarse en el modo posado y hacer uso del numerador de escenas. Esto es una combinación de botones que permiten numerar las escenas a lo largo del tiempo. Hasta el momento hemos explicado solamente los posados, que se realizan por defecto en el instante uno. Para generar un video se puede colocar al modelo en una posición concreta, bloquearlo y asociarlo a ése instante del tiempo. Luego hay que avanzar el contador unas cuantas escenas y colocar el modelo en la posición que tendría unos instantes más adelante, dentro del movimiento que se quiere explicar. Este proceso se repite para varios momentos de un movimiento. Un ejemplo sería colocar al muñeco de cuclillas en el instante uno, con las piernas algo más estiradas en el instante cinco, prácticamente incorporado en el instante diez y completamente de pie en el instante quince.

Basta visualizar el panel de botones dedicado a la animación (F10) para poder especificar las características de la animación y Blender se encargará de generar una enlazando los distintos instantes definidos anteriormente. Para ello es importante indicarle el instante inicial y final, así como el formato de salida. Este puede ser desde un formato de video, como AVI raw hasta incluso JPEG, lo que provoca que Blender genere tantos *renderizados* como instantes de tiempo tenga la animación (como el modo ráfaga de una cámara normal). Todos los formatos tienen su interés, aunque nosotras hemos utilizado solamente los dos comentados en este párrafo.

La desventaja de las animaciones es que, al margen de necesitar mucho más tiempo para ser generadas, ocupan un mayor espacio de memoria y sobrecargan la transmisión de datos con los usuarios. Además, aunque puedan resultar entretenidas, no aportan mucha información adicional ya que con una o dos instantáneas se pueden entender perfectamente cualquiera de los ejercicios propuestos. Estas son las razones por las que a penas hemos incluido animaciones en nuestra base de datos.

1.9.11.EJEMPLOS

A continuación mostramos algunos ejemplos de imágenes definitivas que incluimos en nuestra base de datos.

1.10.MODELO VISTA-CONTROLADOR

Como patrón de diseño hemos utilizado el MCV (Modelo Vista Controlador) que es un patrón de ingeniería del software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de de control en tres componentes distintos.

El patrón MVC se ve frecuentemente en aplicaciones Web, donde la vista son los JSP, y el modelo son los *beans*. A continuación pasamos a explicar con más detalle en qué consiste cada una de las partes de MVC.

- **Modelo**: Esta es la representación específica del dominio de la información sobre la cual funciona la aplicación. El modelo es otra forma de llamar a la capa de dominio. La lógica de dominio añade significado a los datos; por ejemplo, calculando si hoy es el cumpleaños del usuario o los totales, impuestos o portes en un carrito de la compra.
- **Vista**: Presenta el modelo en un formato adecuado para interactuar, usualmente un elemento de interfaz de usuario.
- **Controlador**: Responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista.

Nuestra aplicación utiliza un mecanismo de almacenamiento persistente que es la base de datos.

Es común pensar que una aplicación tiene tres capas principales: presentación (IU), dominio, y acceso a datos. En MVC, la capa de presentación está partida en controlador y vista. La principal separación es entre presentación y dominio; la separación entre la vista y el controlador es menos clara. El flujo que sigue el control generalmente es el siguiente:

1. El usuario interactúa con la interfaz de usuario a través de las paginas Web.
2. El controlador recibe la notificación de la acción solicitada por el usuario.
3. El controlador accede al modelo, actualizándolo, posiblemente modificándolo de forma adecuada a la acción solicitada por el usuario (por ejemplo, el controlador actualiza el carro de la compra del usuario). Los controladores complejos están a menudo estructurados usando un patrón de comando que encapsula las acciones y simplifica su extensión.
4. El controlador delega a los objetos de la vista la tarea de desplegar la interfaz de usuario. La vista obtiene sus datos del modelo para generar la interfaz apropiada para el usuario donde se refleja los cambios en el modelo. El modelo no debe tener conocimiento directo sobre la vista. Sin embargo, el patrón de observador puede ser utilizado para proveer cierta indirección entre el modelo y la vista, permitiendo al modelo notificar a los interesados de cualquier cambio. Un objeto vista puede registrarse con el modelo y esperar a los cambios, pero aun así el modelo en sí mismo sigue sin saber nada de la vista. La interfaz de usuario espera nuevas interacciones del usuario, comenzando el ciclo nuevamente.

4. FASE DE ADQUISICIÓN DE CONOCIMIENTO

1.11.INFORMACIÓN DEPORTIVA

1.11.1.INTRODUCCIÓN

Desde el principio se nos planteaban dos cuestiones cruciales para el funcionamiento de nuestro sistema: Clasificar a los usuarios y elaborar un programa de entrenamiento adecuado. Estas dos cuestiones surgieron muy pronto, pues están fuertemente relacionadas con el razonamiento basado en casos (CBR) que habíamos decidido seguir. Al comenzar a diseñar esta parte del sistema pudimos concretar muy bien qué información nos faltaba y preparar adecuadamente las entrevistas con el experto. En total realizamos dos entrevistas y alguna consulta adicional. El resultado de todo ello está descrito a continuación.

1.11.2.ENTREVISTAS CON EL EXPERTO DEPORTIVO

El experto

Desde el comienzo tuvimos claro que necesitábamos a un experto en materia deportiva, para que nos asesorara y nos proporcionara la información de partida que necesitábamos para el sistema. El experto al que hemos consultado ha sido Carlos Victoria Pfleiderer, licenciado en Ciencias de la Actividad Física y del Deporte por la Universidad Politécnica de Madrid. Además de tener el título de Entrenador Nacional de Atletismo ha sido 6 veces campeón de Madrid de Atletismo, tercer clasificado en el Campeonato Nacional de Austria en el año 2005 y trabaja actualmente como entrenador personal. Esto le convertía en el candidato ideal para asesorarnos, a lo que accedió de manera entusiasta.

La problemática de clasificar a los usuarios

En un principio habíamos considerado que el cliente podría introducir su nivel de preparación física personalmente, valorándolo con un número del 1 al 10. Pero esto era un planteamiento excesivamente simplista que tuvimos que descartar por varias razones.

En cierto modo, nuestro sistema se basa en la confianza, pues no tenemos forma de comprobar si el cliente es sincero, por ejemplo respecto a si realiza o no los ejercicios propuestos. Pero la cuestión de la valoración personal es especialmente importante, por ser la base respecto a la cual el sistema diseña los entrenamientos.

La clave de nuestra propuesta, es desarrollar un sistema que sepa adaptar los entrenamientos a los usuarios, de modo que no podíamos permitir que se generaran en base a un dato erróneo. Además es muy probable que los usuarios, aun sin querer

engañar al sistema conscientemente, introduzcan valores incorrectos por tener una percepción errónea de su estado de salud, como averiguamos al documentarnos sobre este tema [61].

Otro aspecto de este mismo problema es que distintos usuarios conceden distinto grado de importancia a los factores relacionados con su estado físico. La falta de un criterio único imposibilitaría al sistema clasificar a los usuarios de manera correcta, corrompiendo así el razonamiento basado en casos desde el comienzo. Otra razón adicional es la falta de información que supondría un único número para el sistema CBR a la hora de calcular parecidos entre usuarios.

Finalmente todas estas reflexiones nos obligaron a crear una definición propia de "estado físico" que fijara los datos que lo componían y su importancia relativa. Las entrevistas con el experto nos ayudaron a concretar esta definición.

Nuestra definición final de estado físico incluye diversas características del usuario (edad, altura, etcétera) así como los resultados de pruebas concretas que se le plantean al usuario a la hora de darse de alta en el sistema. Esta definición está explicada con mayor nivel de detalle en el apartado de CBR de esta memoria, ya que es allí donde se utiliza esta información.

La problemática de elaborar de un programa de entrenamiento

Aunque todos sabemos realizar algunos ejercicios, este conocimiento dista mucho del necesario para diseñar un entrenamiento correcto. Una sesión de entrenamiento es mucho más que un conjunto de ejercicios presentados en orden aleatorio. Las entrevistas con nuestro experto nos ayudaron a comprender el orden en que se deben realizar los ejercicios y la relación que debe existir entre ellos.

La información que nos permitió resolver estas dos grandes problemáticas está detallada a continuación.

1.11.3.INFORMACIÓN

1.11.4.EVAULUAR LA FORMA FÍSICA DE LOS USUARIOS

Información sobre la valoración de la forma física de los usuarios

Antes de nuestra primera reunión con el experto habíamos recopilado ya información sobre pruebas que habitualmente se hacen a las personas que van a comenzar un entrenamiento. Esta información la obtuvimos de una revista deportiva [62] y la pasamos en su mayor parte a ordenador.

En esta primera fase descartamos algunas pruebas por razones como el material que se necesita para realizarlas (balones medicinales, etcétera), que estuvieran pensadas para deportistas de elite o simplemente porque consideramos que un usuario medio no estaría dispuesto a realizarlas (porque involucraban localizar una pista de atletismo de ciertas características sobre la que correr una distancia concreta, etcétera).

La segunda fase corresponde a las entrevistas, en las cuales estuvimos considerando las ventajas e inconvenientes de las pruebas recopiladas. Con ayuda del experto acabamos teniendo un conjunto interesante que nos permitía, eligiendo pruebas de distintas categorías, obtener una información bastante precisa sobre el estado físico del usuario.

A continuación se encuentra el listado de pruebas con las observaciones del experto que obtuvimos al finalizar la segunda fase. Nótese que no todas las pruebas descritas se consideraron aptas para los usuarios y que nuestra recopilación sólo es un subconjunto de todas las posibles pruebas que se pueden realizar.

1.11.5.Las pruebas físicas

1. Pruebas para valorar la movilidad articular

1.1. Test de Grauus-Weber (tocar el suelo)

Objetivo: Medir la amplitud articular del tronco.

El ejercicio consiste en: de pie, con los pies juntos, se empieza a flexionar el tronco hacia delante dejando colgar los brazos lateralmente con la intención de tocar el suelo sin flexionar las rodillas. Se mantiene la posición al menos durante 3 segundos. No se admiten rebotes ni movimientos bruscos.

Puntuación: 10 si se toca el suelo. Se resta 1 punto por cada centímetro que se aleja de este.

→ INCLUIDA EN LA VERSIÓN FINAL DEL SISTEMA.

1.2. Test de extensión de tronco

Objetivo: Medir la hiperextensión del tronco.
El ejercicio consiste en: en posición boca abajo, un ayudante sujeta al evaluado por los tobillos. El evaluado tiene las dos manos agarradas por la espalda y debe elevar lo máximo posible el tronco, separándolo del suelo. El entrenador medirá desde la parte superior hasta el suelo.

→ DESCARTADO: por necesitar la colaboración de una segunda persona.

1.3. Test del cajón

Objetivo: Medir la flexibilidad combinada de la espalda, cadera y región posterior del muslo.

El ejercicio consiste en: se utiliza un cajón que tiene una escala de medición o simplemente algún elemento que haga de tope para los pies (se puede utilizar una cinta métrica).
Se fijan las plantas de los pies separados a la anchura de los hombros y las rodillas bloqueadas en extensión. Se extienden los brazos hacia delante y las manos se colocan sobre el cajón. El deportista avanza hacia delante a lo largo de la escala de medición cuatro veces y determina la distancia que puede alcanzar. Se ha de mantener la posición durante 3 segundos en el último intento.

Se utiliza un cajón de las siguientes medidas: largo: 35 cm. Ancho: 45 cm. Alto: 32 cm.

En la parte superior del cajón se acopla una cinta métrica que permitirá realizar la medida en centímetros y debe sobresalir 15 cm. del cajón. Es decir, un resultado de 15 cm. en este test equivale a llegar a la altura del pie.

→ DESCARTADO: por la necesidad de un cajón de dimensiones concretas.

Valores de referencia para adultos (en cm.):

Clasificación	Hombres	Mujeres
Baja	< 14,0	< 30,0
Regular	14,0 – 24,0	30,0 – 33,0
Media	24,1 – 35,0	33,1 – 37,0
Buena	35,1 – 45,0	37,1 – 41,0
Excelente	> 45,0	> 41,0

2. Pruebas para valorar la fuerza

2.1. Determinación de la fuerza máxima para cualquier grupo muscular

No destinado a adolescentes, personas con poca experiencia o sujetos sedentarios.

→ DESCARTADO: por riesgo de lesión.

2.2. Determinación de la potencia del tren inferior

2.2.1. Detente vertical

El ejercicio consiste en:
1. Dibujar o fijar una escala métrica en la pared.
2. El deportista se colocará al lado de esta y levantará uno de los brazos, dejando una marca en el punto más alto que llegue.

3. Seguidamente se coloca sobre el lugar señalado con las piernas ligeramente separadas.
4. Desde la posición de flexión de rodillas, y sin hacer contramovimiento, se salta lo más arriba que se pueda y se marca con la mano, en la pared, el punto más alto donde se ha llegado.

Medida: la medida es la diferencia entre la maca de hacer el salto y la primera medición.

Tabla con valores de referencia:

Puntos	Hombres	Mujeres
10	70 o más	60 o más
9	65	56
8	58	52
7	54	48
6	50	44
5	46	40
4	42	36
3	38	32
2	34	28
1	30	25
0	Menos de 30	Menos de 25

→ INCLUIDA EN LA VERSIÓN FINAL DEL SISTEMA.

2.2.2. Detente horizontal

El ejercicio consiste en:

1. El deportista se coloca derecho, con los pies ligeramente separados y las puntas detrás de la línea de salida.
2. Se hace una flexión de rodillas y se llevan los brazos hacia atrás para preparar el salto.
3. A partir de ahí se ejecuta el test, saltando lo más lejos que se pueda y llevando los brazos hacia delante. Se ha de caer con los dos pies al mismo tiempo.
4. Se dibujará la última la última marca dejada por las zapatillas y se medirá la distancia de la línea de salida hasta la marca.

Medida: se deben realizar dos intentos y al final se conservará la mejor marca.

Tabla con valores de referencia:

Puntos	Hombres	Mujeres
10	2,85	2,35
9	2,70	2,25
8	2,65	2,20
7	2,60	2,15

Puntos	Hombres	Mujeres
6	2,55	2,10
5	2,50	2,05
4	2,47	2,00
3	2,45	1,95
2	2,40	1,90
1	2,30	1,80
0	2,20	1,70

→ NO INCLUIDA EN LA VERSIÓN FINAL DEL SISTEMA. Aunque es una prueba aceptable, ya hemos elegido otra para valorar la fuerza del tren inferior del usuario.

1.11.6.

2.3. Determinación de la fuerza explosiva global

El ejercicio consiste en el lanzamiento de un balón medicinal.

Nota: El peso del balón medicinal varía: mujeres: 3kg, hombres: 5kg

→ DESCARTADO: por el material necesario para realizar la prueba.

2.4. Determinación de la fuerza resistencia

2.4.1. Flexiones o fondos de brazos

El ejercicio consiste en: hacer la mayor cantidad posible de flexiones. No hay límite de tiempo.

1. El deportista debe colocarse con los brazos extendidos, separados la anchura de los hombros. El tronco y las piernas deben formar una línea recta.
2. Se considera un ejercicio completo cuando se llega dala extensión total de brazos a la flexión de codos de 90º grados. El movimiento ha de ser continuo durante la flexo-extensión y no debe haber ningún contacto entre el suelo y el cuerpo, a excepción de las manos y las puntas de los pies claro.

Nota: Las mujeres pueden realizar el ejercicio con las rodillas apoyadas en el suelo.

Medida: número de flexiones completas realizadas de forma consecutiva.

Tabla con valores de referencia:

Fondos completos					
Edad	Valoración				
	Excelente	Muy bien	Bien	Medio	Bajo
20 – 29	> 54	45 – 54	35 – 44	20 – 35	< 20
30 – 29	> 44	35 – 44	25 – 34	15 – 24	< 15

40 – 49	> 39	30 – 39	20 – 29	12 -19	< 12
50 – 59	> 34	25 – 34	15 – 24	8- 14	< 8
60 +	> 29	20 – 29	10 – 19	5 – 9	<5

Fondos modificados					
Edad	Valoración				
	Excelente	Muy bien	Bien	Medio	Bajo
20 – 29	> 48	34 – 38	17 – 33	6 – 16	< 6
30 – 29	> 39	25 – 39	12 – 24	4 – 11	< 4
40 – 49	> 34	20 – 34	8 – 9	3 – 7	< 3
50 – 59	> 29	15 - 29	6 -14	2 – 5	< 2
60 +	> 19	5 -19	3- 4	1- 2	< 1

→ INCLUIDA EN LA VERSIÓN FINAL DEL SISTEMA.

2.4.2. Abdominales en 30 segundos

El ejercicio consiste en: realizar la mayor posible cantidad de abdominales en un plazo de 30 segundos. Se trata de medir la resistencia de los músculos abdominales.

1. El deportista se colocará boca arriba con las piernas flexionadas 90 grados, los pies ligeramente separados y los dedos por detrás de la nuca.
2. Se debe completar el mayor número de veces el ciclo flexión-extensión de cadera, tocando las rodillas con los codos en la flexión y el suelo con la espalda en la extensión.

Medida: el número de abdominales completos realizados en el tiempo disponible.

Tabla con valores de referencia para adultos:

	Valoración				
	Excelente	Sobre la media	En la media	Debajo de la media	Pobre
Hombre	> 30	26 – 30	20 – 25	17 – 19	< 16
Mujer	> 25	21 - 25	15 – 20	9 – 4	< 8

→ INCLUIDA EN LA VERSIÓN FINAL DEL SISTEMA.

3. Pruebas para valorar la velocidad de desplazamiento

3.1. Velocidad 10 x 5

Objetivo: Medir la velocidad de desplazamiento y la agilidad.

El ejercicio consiste en:

1. Se necesitan dos líneas paralelas en una superficie plana, separadas 5 metros la una de la otra. El deportista se debe colocar tras una de ellas sin tocarla.
2. Al oír la señal deberá salir a toda velocidad para pisar la línea contraria. Cada línea debe ser pisada 5 veces en total. La línea de salida también se cuenta.
3. El ejercicio finaliza cuando el deportista ha de atravesar y pisar detrás de la que fue la línea de salida. Cuando esto haya ocurrido se pulsa el cronómetro.

Medida: el tiempo requerido para completar el ejercicio.

Tabla con valores de referencia para adultos:

Puntos	Hombres	Mujeres
10	5” 95	6” 35
9	6” 15	6” 65
8	6” 35	6” 95
7	6” 55	7” 25
6	6” 75	7” 55
5	6” 95	7” 85
4	7” 15	8” 15
3	7” 40	8” 45
2	7” 65	8” 75
1	7” 90	9” 05
0	8” 25	9” 35

→ DESCARATDO: porque finalmente la velocidad no forma parte de los criterios necesarios para clasificar a los usuarios.

3.2. Carrera de 40m

Objetivo: Medir la velocidad de desplazamiento máxima en la distancia de 40 metros.

El ejercicio consiste en:

1. La salida se realiza de pie, con los ambos pies a la misma altura. Ni se puede adelantar uno al otro ni está permitido impulso previo.
2. El tiempo empieza a correr en cuanto uno de los pies se despega del suelo.
3. El tiempo se para cuando el pecho del deportista atraviesa la marca de los 40 metros.

Medición: el tiempo requerido.

Valores de referencia para adultos de 19 a 25 años:

Excelente	Muy bueno	Bueno	Regular	Suficiente	Malo	Muy malo
4” 50	4” 80	5” 10	5” 40	5” 70	5” 90	Más de 6”

→ DESCARTADO: porque los valores de referencia proporcionados no cubren todo el rango posible de usuarios.

4. Pruebas para valorar la resistencia cardiovascular

4.1. Prueba de Cooper

Objetivo: Medir la resistencia aeróbica a media duración.

Nota: es una prueba estándar con la que se valora a todos los niveles.

El ejercicio consiste en:

1. Salida de pie.
2. Se corre durante 12 minutas la mayor distancia posible sobre una pista preparada.

Medida: la distancia recorrida.

Valores de referencia:

Excelente	Bien	Suficiente	Mal	Muy mal
Más de 2.800	2.400 – 2.800	2.000 – 2.400	1.600 – 2.000	Menos de 1.600

→ NO INCLUIDA EN LA VERSIÓN FINAL DEL SISTEMA. Aunque es una prueba aceptable, finalmente utilizamos el test de índice de Ruffier para determinar la resistencia cardiovascular, ya que no exige abandonar el domicilio o lugar donde se realizan las demás pruebas.

4.2. Saltos laterales de cajón

Objetivo: Medir la resistencia aeróbica de larga y media duración.

El ejercicio consiste en:

1. Durante 90 segundos el deportista ha de saltar el mayor número de veces que le sea posible, de un lado a otro de un cajón o plinto.
2. En cada salto ha de posar ambos pies sobre el cajón. Es decir: derecha, encima, izquierda, encima, derecha, encima, etcétera
3. El ejercicio se comienza de pie, a un lado del cajón.

Nota: Las características del cajón a tener en cuenta son: 40 cm. de alto.

Tabla con valores de referencia para edades entre 19 – 27 años:

Hombres	140 + / -6
Mujeres	84 + / - 6

→ DESCARTADO: porque los valores de referencia proporcionados no cubren todo el rango posible de los usuarios.

4.3. Test de índice de Ruffier

Objetivo: Medir la adaptación cardiovascular al esfuerzo.
El ejercicio consiste en:

1. Tomar las pulsaciones en reposo (Pr).
2. El deportista, colocado de pie, con la espalda recta y manos en la cadera debe realizar 30 flexo-extensiones de piernas en 45 segundos. Tras finalizar el ejercicio se toman de nuevo las pulsaciones (Pe).
3. Se espera un minuto y se vuelven a medir las pulsaciones (Pm).

Nota: La toma de pulsaciones se realiza en 15 segundos. El número de pulsaciones registradas se multiplica por 4.

Cálculo: Índice de Ruffier = ((Pr + Pe + Pm) – 200) / 10

Valoración:

Excelente	Muy buena	Buena	Débil o moderada	Mala
< 0	0 – 5	5 - 10	10 – 15	> 15 (21, 22)

→ INCLUIDA EN LA VERSIÓN FINAL DEL SISTEMA.

4.4. Test del kilómetro

Objetivo: Valorar la resistencia aeróbica-anaeróbica.

El ejercicio consiste en: Correr 1km en el menor tiempo posible.

Medición: Tiempo Requerido.

Tabla con valores de referencia:

	Edad			
Valoración	18 – 30 años	31 – 40 años	41 – 55 años	Más de 55 años
Excelente	Menos de: 3:15	Menos de: 3:20	Menos de: 3:30	Menos de: 3:45
Muy buena	3:16 – 3:30	3:20 – 3:35	3:30 – 3:50	3:46 – 4:00
Buena	3:31 – 3: 50	3:36 – 3:55	3:51 – 4:15	4:01 – 4:25
Normal	3:51 – 4:15	3:56 – 4:20	4:16 – 4:50	4:26 – 5:00
Regular	4:16 – 4:50	4:21 – 4:50	4:51 – 5:30	5:01 – 5:40
Mala	4:51 – 5:30	4:51 – 5:30	5:30 – 6:30	5:41 – 6:45
Suspenso	Más de: 5:30	Más de: 5:30	Más de: 6:30	Más de: 6:45

→ NO INCLUIDA EN LA VERSIÓN FINAL DEL SISTEMA. Aunque es una prueba aceptable, finalmente nos hemos inclinado por el test de Ruffier.

5. Pruebas para valorar el equilibrio estático

Objetivo: Evaluar la capacidad de mantener una posición estática en el tiempo.

El ejercicio consiste en:

1. El deportista debe ponerse en pie con las manos en las caderas.
2. A continuación ha de levantar una pierna, colocando los dedos del pie elevado sobre la rodilla de la pierna contraria (la que sirve de apoyo).
3. Por último debe elevar el talón de la pierna estirada y mantener el equilibrio sobre la punta.

Nota: el ejercicio termina cuando o bien el talón toca el suelo o bien el pie que está apoyado sobre la rodilla se despega de ella.

Medida: tomando el número de segundos que logra mantener ésa postura con cada pierna, se realiza la media y se obtiene el valor que corresponde al deportista evaluado.

Valoración:

Excelente	Bien	Medio	Regular	Malo
Más de 50”	40 – 49”	26 – 39”	10 – 25”	Menos de 10”

→ NO INCLUIDA EN LA VERSIÓN FINAL DEL SISTEMA. Aunque es una prueba aceptable, el equilibrio no es un dato relevante para los aspectos de la condición física que contemplamos en nuestro sistema.

6. Otros test físicos que miden el nivel de adaptación y condición cardiovascular

6.1. Test de Balke

Objetivo: Valora la resistencia aeróbica de media duración. Determinar el VO_2 máximo.
Destinado a: atletas entrenados.

El ejercicio consiste en: (es una variante del test de Cooper)
Recorrer la mayor distancia posible en 15 minutos.

La tabla del VO_2 máximo estimado es:

Distancia recorrida	VO_2 máximo estimado
6.000 metros	80 ml / kg / min
5.600 metros	75 ml / kg / min
5.200 metros	70 ml / kg / min
4.800 metros	66 ml / kg / min
4.400 metros	61 ml / kg / min
4.000 metros	56 ml / kg / min
3.600 metros	52 ml / kg / min
3.200 metros	47 ml / kg / min
2.600 metros	40 ml / kg / min
2.200 metros	35 ml / kg / min
1.800 metros	31 ml / kg / min

→ DESCARTADO: por estar destinado sólo a atletas entrenados.

6.2. Test de Rockport

Objetivo: Determinar el VO_2 máximo en sujetos de baja condición física.

Destinado a: sujetos de baja condición física.

El ejercicio consiste en: recorrer andando, según el ritmo de cada persona, una milla (1609,3 metros).

Medición: frecuencia cardiaca al terminar el ejercicio y tiempo empleado en el.

El cálculo es:
VO_2 máximo aproximado = 132,6 – (0,17 * Peso corporal [en kg]) - (0,39 * edad) + (6,31 * Sexo) – (3,27 * tiempo [en minutos]) – (0,156 * frecuencia cardiaca justo al acabar)

Nota: sexo: mujeres: 0, hombres: 1

→ NO INCLUIDA EN LA VERSIÓN FINAL DEL SISTEMA. Aunque es una prueba aceptable, los datos que proporciona no son de utilidad para el sistema.

6.3. Test de Burpee

Objetivo: Medir la resistencia anaeróbica.

El ejercicio consiste en: Realizar el siguiente ejercicio el mayor número de veces posible en un minuto. El ejercicio consiste en encadenar las siguientes posiciones:

- Posición 1: De pie, con los brazos colgando.
- Posición 2: Piernas flexionadas (en cuclillas).
- Posición 3: Con apoyo de manos en el suelo se realiza una extensión de piernas.
- Posición 4: Flexión de piernas y vuelta a la posición 2.
- Posición 5: Extensión de piernas y vuelta a la posición 1.

Nota: Se considera un ejercicio completo cuando el alumno, partiendo de la posición 1 pasa a la 5 realizando las posiciones 2, 3 y 4.

La tabla es:

Menos de 30	De 30 a 40	De 40 a 50	De 50 a 60	Más de 60
Malo	Normal	Bueno	Muy bueno	Excelente

→ NO INCLUIDA EN LA VERSIÓN FINAL DEL SISTEMA, aunque es una prueba aceptable.

6.4. Prueba de esfuerzo para conocer la condición cardiovascular

Objetivo: Valora la resistencia aeróbica de media duración. Determinar el VO2 máximo.

El ejercicio consiste en:

1. Andar a ritmos suave durante diez minutos.
2. Luego cubrir 1.600 metros a la máxima velocidad, pero sin correr. Registrar el tiempo que se necesita para cubrir esta distancia.
3. Nada más terminar esto reducir el ritmo pero seguir caminando.
4. Tomar el pulso justo cuando se empieza a andar más suavemente.

El cálculo consiste en:
A = edad [en años] * 0.388
B = peso [en kg] * 0.169
C = tiempo en 1.600 metros [en minutos] * 0,156
VO2 máximo aproximado [ml/kg/min] = 139.168 – (A + B + C + D)

La tabla es:

Edad	De 20 a 29	De 30 a 39	De 40 a 49	De 50 a 59
Excelente	Más de 51	Más de 48	Más de 46	Más de 42
Bueno	49 a 51	46 a 48	44 a 46	40 a 42
Medio	42 a 48	39 a 45	37 a 43	33 a 39
Bajo la media	Menos de 41	Menos de 38	Menos de 36	Menos de 32

→ NO INCLUIDA EN LA VERSIÓN FINAL DEL SISTEMA, aunque es una prueba aceptable.

1.11.7.Recomendaciones para la realización de ejercicio

Información sobre cómo construir una sesión de entrenamiento

Una vez habíamos determinado las pruebas con las que clasificar a los usuarios, nos centramos en cómo diseñar una sesión de entrenamiento. A continuación se encuentra el texto que resume la información que obtuvimos del experto en las reuniones y nuestros sucesivos planteamientos para crear un esquema que se amoldara a la parte CBR de nuestro sistema.

1.11.8.Información general

1. Edad y pulsaciones recomendadas
Fórmula: Pulsaciones máximas = 220 – edad

2. Relación protagonista / antagonista entre grupos musculares

- Abdominales / Lumbares: 4 a 1

3. Horario idóneo para entrenar
Mejores horas para entrenar: 9:00, 18:00 a 21:00

4. Estructura del programa básico de entrenamiento de 60min

Duración	Componentes	
15 min.	Calentamiento	General
		Específico
35 min.	Parte principal	
10 min.	Vuelta a la calma	General
		Específico

"General" y "Específico"

- Musculatura tónica (general, para andar erguido): función de sostén.
  Es la que se debe calentar y estirar en la parte "General".

- Musculatura fásica (más específica): función de movimiento.
  Es la que se debe calentar y estirar según lo que se vaya a hacer / haya hecho en la parte principal.

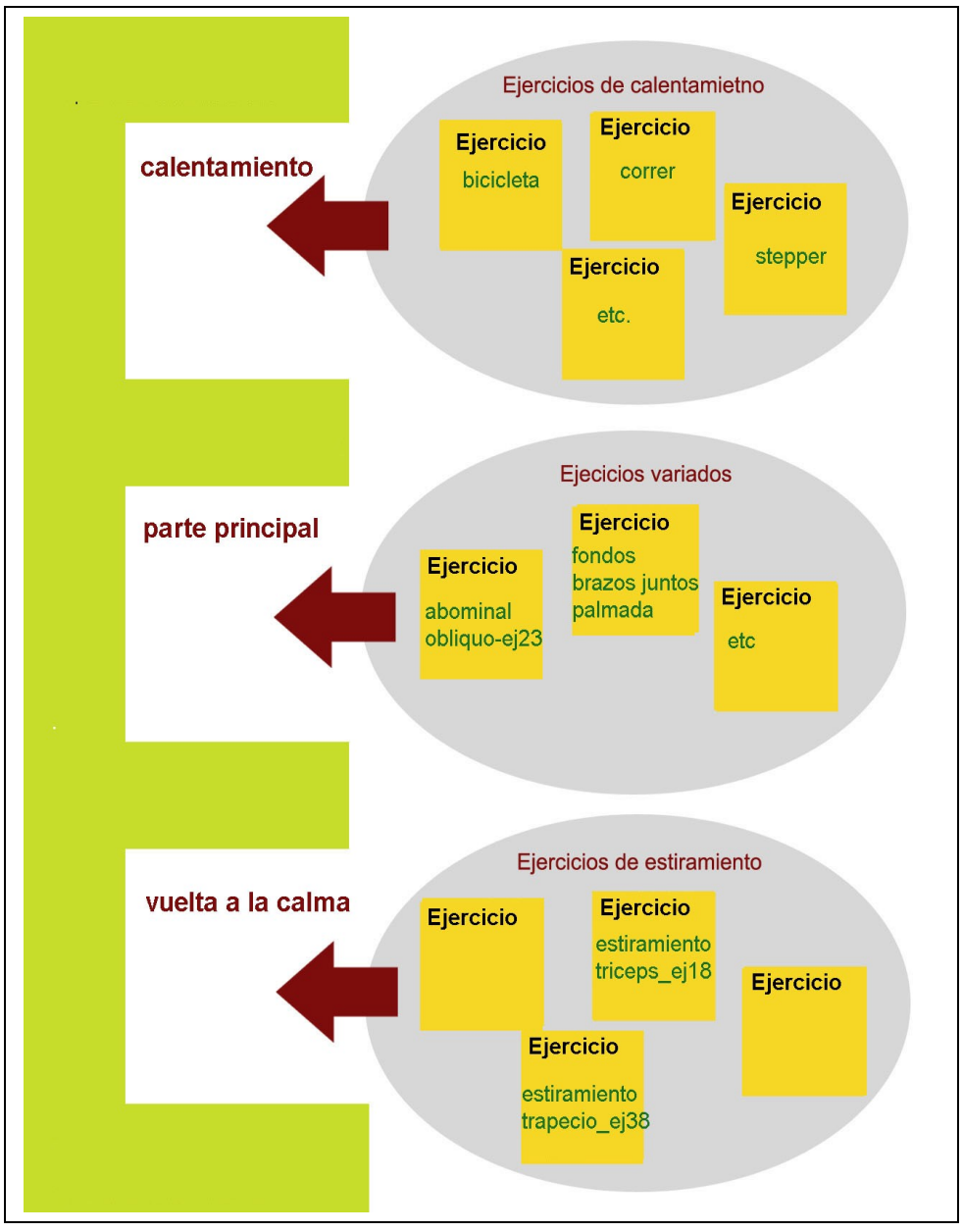
1.11.9.4.1.4. LAS SESIONES DE ENTRENAMIENTO

1.11.10.4.1.4.1. Las sesiones de entrenamiento

Planificar una sesión: Idea general

A groso modo, una sesión de entrenamiento consiste en calentamiento, parte principal y vuelta a la calma. En cada sección se pueden realizar varios ejercicios.

El siguiente esquema explica este primer acercamiento:

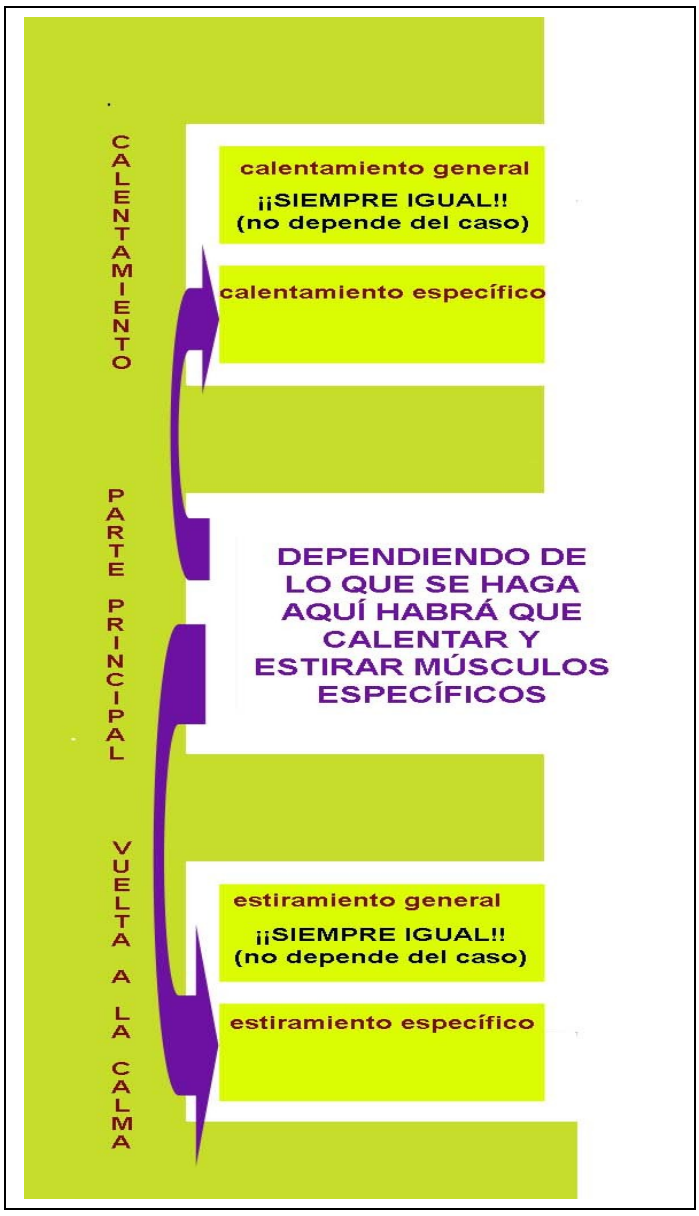


Planificar una sesión: Versión mejorada

Si además tenemos en cuenta lo explicado al principio, referente a la musculatura tónica y fásica, se deduce lo siguiente:

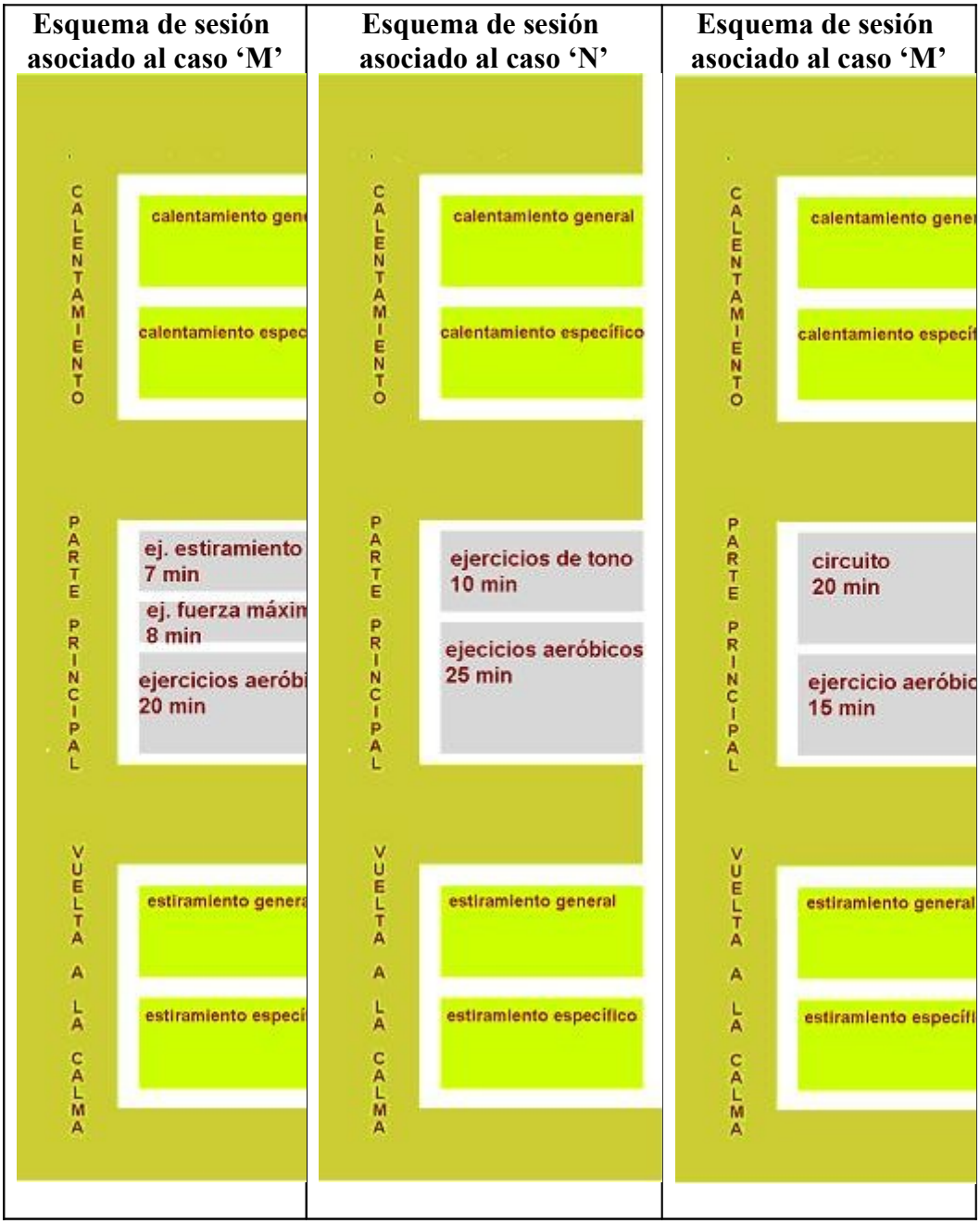
- La parte general (tanto del calentamiento como de la vuelta a la calma) va a ser exactamente igual para todas las sesiones de todos los casos.
- La parte específica (tanto del calentamiento como de la vuelta a la calma) dependerá de los músculos que se muevan en la parte principal. Bastará con buscar en la BD ejercicios que los estiren / sirvan para calentar.

Conclusión: Lo único que verdaderamente hay que pensar es la parte principal.
Esto se puede reflejar con el siguiente esquema:



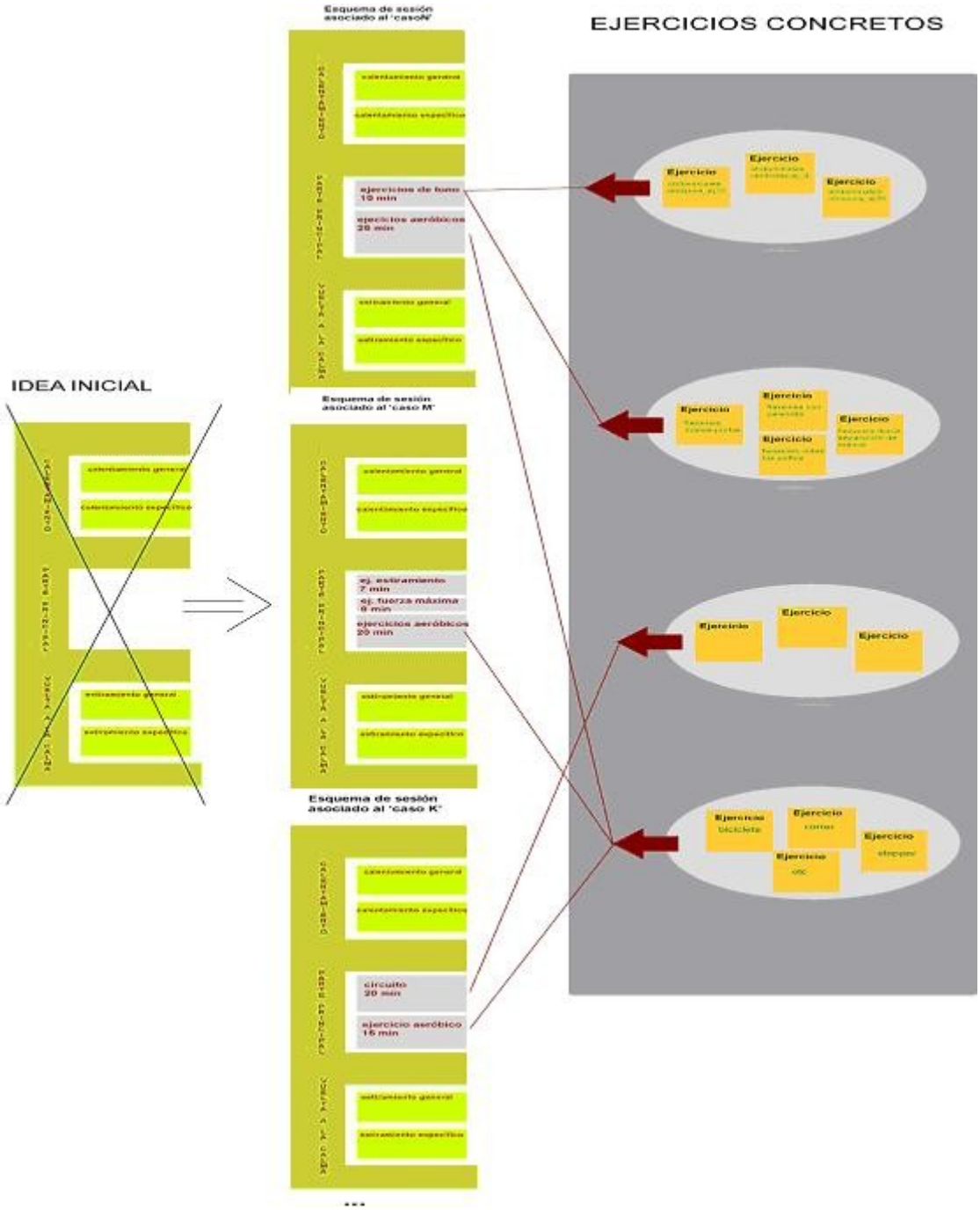
Planificar una sesión orientada a los casos de CBR

El planteamiento anterior es correcto, pero no tiene en cuenta objetivos. Para crear un entrenamiento diferente, orientado a los casos (es decir: que tuviera en cuenta las características y objetivos de los usuarios) tuvimos que desarrollar varios formatos de sesiones de entrenamiento.
Lo que hicimos fue pasar a un nivel intermedio entre el esquema general y los ejercicios disponibles en la base de datos. A este nivel de detalle ya se distinguen distintas estructuras de entrenamiento según la condición física y el objetivo de cada caso, pero se conserva la flexibilidad de incluir un u otro ejercicio específico. Esto también facilitará las adaptaciones que haya que realizar en nuestro sistema CBR.
El siguiente esquema explica la evolución de nuestro diseño:



Nota: Estos esquemas son meramente ejemplos y no se corresponden necesariamente con sesiones reales existentes en nuestra base de casos.

En relación con los puntos anteriores, este nuevo diseño se ubica en un nivel de abstracción intermedio, como se pude apreciar en la imagen siguiente.



El esquema de sesión correspondiente a cada caso tiene una zona principal distinta. Esto se debe a que, ahora sí, existe una planificación enfocada a la condición física y al objetivo del caso. Es la parte principal (y las zonas específicas) lo que varía de caso en caso y lo que, en definitiva, la nos permite proporcionar un entrenamiento específico, adaptado al cliente.

1.12.INFORMACIÓN NUTRICIONAL

1.12.1.INTRODUCCIÓN

Al comienzo de nuestro proyecto diseñamos la parte central del sistema, pero también esbozamos distintas ampliaciones posibles. La decisión de descartar o no las ampliaciones la tomamos más adelante, en función de los diferentes factores (tiempo, recursos, etcétera) que influyen habitualmente en el desarrollo de un proyecto.

Por eso inicialmente recopilamos una gran cantidad de información no exclusivamente de carácter deportivo. Nos pareció especialmente interesante la estrecha relación que hay entre el deporte y la nutrición a la hora de tratar de obtener ciertos resultados. Aunque no sabíamos si la nutrición acabaría formando parte del proyecto, consideramos que merecía la pena invertir cierto esfuerzo en averiguar más sobre esta temática. Esto decisión la tomamos en base al tiempo del que disponíamos entonces así como por la convicción de que un mayor nivel de información nos permitiría valorar, más adelante, el trabajo y las complicaciones que supondría ampliar el sistema en este sentido.

1.12.2.ENTREVISTAS CON LA EXPERTA EN NUTRICIÓN

La experta

Por todo lo explicado anteriormente, nos pusimos en contacto con una experta que nos pudiera asesorar en este aspecto. María Isabel Espinosa Salinas es Diplomada en Nutrición Humana y Dietética por la Universidad Autónoma de Madrid y colaboró amablemente con nosotras en esta etapa de adquisición del conocimiento.

La información nutricional

El campo de la alimentación es increíblemente extenso, por lo que nuestros esfuerzos iniciales iban dirigidos a catalogar al usuario según un criterio nutricional. Para ello recopilamos, con ayuda de la experta, diversa información sobre las características de las personas y las conclusiones sobre su estado físico, a las que se puede llegar a partir de ellas. A continuación se encuentran la información que recopilamos.

1.12.3.INFORMACIÓN

1.12.4.MEDICIONES PARA DETERMINAR LAS CARACTERÍSTICAS DE LOS USUARIOS [63]

Dado que la percepción propia algo subjetivo, el objetivo de estas mediciones es proporcionar al sistema datos más fiables sobre los usuarios. (Obviamente esto presupone que el cliente no miente sobre las medidas.)

1. Determinar el gasto energético total (GET)

1.1. Metabolismo Basal (MB)
Qué es: Lo que consume nuestro cuerpo en reposo
Cálculo:

Hombres	MB = 1,0 [Kcal.] * 24 [horas] * peso [Kg.]
Mujeres	MB = 0,9 [Kcal.] * 24 [horas] * peso [Kg.]

Nota: También está la fórmula de la tasa de metabolismo basal (TMB) de Harris – Bennedict, que se calcula así:

Hombres	TMB = 66,473 + (5*T[cm.]) + (13,7*P[Kg.]) – (6,8*E[años])
Mujeres	TMB = 655,096 + (1,8*T[cm-]) + (9,6*P[Kg.]) – (4,7*E[años])

Siendo: T talla (es decir: altura,) P peso y E edad.
Observación: TMB y MB son lo mismo.

1.2. Actividad Física (AF)
Qué es: Se compone del gasto energético que suponen las actividades deportivas y la actividad laboral.
Cálculo: $\Sigma_{\text{de actividades}}$ (factor de esfuerzo * número de [horas] * peso de la persona [Kg.]).
Nota: existen tablas con factores de esfuerzo para distintas tareas.

1.3. Termogénesis (T)
Qué es: El gasto calórico que supone mantener la temperatura del cuerpo.

1.4. GET
Qué es: Es el gasto calórico total (que tiene una persona al día).
Cálculo ideal: GET = MB + AF + T
Cálculo aproximado: GET = MB + (MB*0,2) + (MB * 0,1)
Nota: el 0,2 es un factor de actividad media, el factor 0,1 es otra aproximación.

2. Determinar el Índice de Masa Corporal

2.1. IMC

Qué es: Un valor de referencia, respecto a la población general.
Cálculo: IMC = peso de la persona [Kg.] / (altura [m])^2

2.2. Clasificación del cliente

Intervalo de peso	Clasificación
< 19	Peso insuficiente
[19 – 25)	Normopeso (peso normal)
[25 – 27)	Sobrepeso
[27 – 30)	Sobrepeso (grado 1)
[30 – 40)	Sobrepeso (grado 2)
[30 -35)	Obesidad (grado 1)
[35 – 40)	Obesidad (grado 2)
[40 – 50)	Obesidad mórbida
> 50	Obesidad tipo 4 extrema

→ ESTE CRITERIO ESTÁ INCLUIDO EN LA VERSIÓN FINAL DEL SISTEMA.

3. Determinar el peso ideal

Qué es: Esto es otra forma de determinar el peso ideal (en vez de con el IMC y mirando en las tablas). Observación: talla y altura son lo mismo.

Intervalo de edad		Cálculo de peso ideal
Adultos	Hombres	Talla [cm.] – 100 – (Talla [cm.] – 150) / 4
	Mujeres	Talla [cm.] – 100 – (Talla [cm.] – 150) / 2
14 – 18 años		Talla – 110
6 – 14 años		(Talla – 125) / 2
2 - 6 años		Talla [cm.] – 100 – (Talla [cm.] – 123) * 0,7

4. Nivel de riesgo

Existe relación entre el riesgo de presentar ciertas enfermedades y el perímetro de la cintura. Con este dato pretendemos llamar la atención del usuario sobre estos posibles problemas.

4.1. Tipos de enfermedades cuyo nivel de riesgo determinamos

- Diabetes tipo 2
- Hipertensión Arterial (HTA)
- Enfermedades cardiovasculares

1.12.5.

4.1. Clasificación del cliente

Hombres	Mujeres	Riesgo
0 – 95	0 – 82	Normal
95 – 102	82 – 90	Valores de riesgo
>102	> 90	Valores de riesgo elevado

5. Peso ajustado

5.1. Peso al que se debe llegar
Qué es: El peso alcanzable si el deseable no lo es.
Cálculo: peso ideal + ((peso actual – peso ideal) / 4).

1.12.6.DATOS NUTRICIONALES

1. Kcal. producidas por un gramo de…

1 gramo de	Supone tantas Calorías
Proteínas	4
Glúcidos (=Hidratos de carbono)	4
Fibra	2
Lípidos	9
Alcohol	7

Definición: Una Kilocaloría es la energía que se necesita para subir en un grado la temperatura de un litro de agua.

1.12.7.GASTO ENERGÉTICO SEGÚN LAS ACTIVIDADES REALIZADAS [64]

1.12.8.Tabla de gasto energético [65]

Actividades		Cal / minuto
Sueño		1,13
Siesta		1,4
Aseo personal	Lavarse, vestirse, afeitarse	3,0
Comidas		1,6
Desplazamientos a pie	Andando sobre superficie lisa (4,8 Km./h)	4,0
	Subiendo y bajando escaleras (15 cm.)	6,2

Actividades		Cal / minuto
Recreativas	Tendidos en reposo	1,4
	Sentados en reposo	1,6
	Sentados oyendo la radio	1,7
	Sentados escribiendo	1,9
	De pie, descansando	1,8
	Bailar	6,0
	Cine	2,0
Desplazamientos en vehículos	Metro, tranvía, autobús	2,0
	Conducir automóvil	2,8
	Conducir moto	3,4
	Conducir bicicleta (a su paso)	6,6
	Conducir bicicleta (14 km/h)	8,2
Deportivas	Ejercicios gimnásticos	3,5
	Balonvolea	3,5
	Fútbol	8,9
	Baloncesto	9,0
	Remar	4,1
	Nadar (18 km/h)	5,0
	Esquiar (6 km/h)	9,9
	Esgrima	6,0
Domésticas	Coser a mano	1,7
	Coser a máquina	2,8
	Barrer	1,7
	Cepillar calzado	2,2
	Lavar prendas pequeñas	3,0
	Pelar patatas	2,9
	Fregar suelos de rodillas	5,9
	Fregar suelos de pie	3,6
	Hacer camas	3,9
	Planchar	4,2
Ocupación	Estudio y clase	2,0

1.12.9.Un posible formato de formulario

Actividad		Total tiempo empleado		Cal / minuto	Total calorías
		Horas	Minutos		
Sueño					
Duermevela y siesta					
Aseo personal					
Trabajo doméstico					
Comidas					
Desplazamientos	Andando				
	En vehículo				
Recreo sedentario					

Deportes				
Total				

Nota: Los campos en verde claro los puede calcular el programa si dispone de la tabla anterior. Los campos en blanco los debe rellenar el cliente, eligiendo dentro de las distintas categorías actividades cuyo gasto energético tengamos registrado en la tabla.

El objetivo es calcular el gasto energético diario (promedio) y con eso obtener el gasto energético semanal. Una vez obtenidos ésos datos se sabe el aporte calórico semanal que necesita ésa persona y a partir de allí se puede elaborar una dieta.

Uso de la información nutricional

Más adelante las limitaciones de tiempo nos obligaron a descartar ampliaciones de carácter nutricional, pero parte de la información recompilada (como la fórmula del IMC) sí acabaron formando parte de nuestro sistema.

1.13.ESTRUCTURA DE LA BASE DE DATOS

Hemos desarrollado la base de datos (BD) de forma incremental, de modo que desde el primer diseño hasta la versión definitiva esta sufrió diversas modificaciones. En este punto queremos describir nuestro primer acercamiento al diseño de la estructura de la base de datos, porque al comienzo del proceso de diseño éste está fuertemente influenciado por la información adquirida en la fase de adquisición del conocimiento. Más adelante los cambios de la base de datos se deben cada vez más a necesidades de implementación, pero inicialmente el objetivo es recoger los datos que consideramos necesarios, de la forma más adecuada. Estas primeras etapas de diseño son la que vamos a explicar a continuación. Para ello reproducimos los textos que escribimos en aquel momento y que reflejan tanto la información recopilada como las alternativas que se nos planteaban.

1.13.1.FASE 0: CONSIDERACIONES GENERALES (*BRAINSTORMING*)

Antes de entrar en el diseño queremos anotar las características que, a priori, consideramos necesarias en la base de datos.

Observación: Una posible ampliación consiste en que el sistema cree frases, ya sean audio o texto, a medida del cliente. Más adelante decidiremos si lo implementaremos, pero en este primer acercamiento vamos a incluir todo lo que creemos puede ser de utilidad. Lo mismo sucede con los idiomas y demás opciones que aun están por decidir.

1.13.2.POSIBLES CAMPOS DE LA BASE DE DATOS

Información sobre la persona (cliente) que realiza los ejercicios

DNI → puede servir como clave.
nombre → útil para incluirlo en los mensajes, dando la sensación de un trato personal.
fecha de nacimiento → para calcular su edad, cálculos de calorías, cuestiones legales…
sexo → para cálculos de calorías, tipos de ejercicios, forma de hablarle (femenino/masculino).
presupuesto deportivo → cantidad de dinero que está dispuesto/a a gastar al mes en material deportivo (permite sugerencias) (este campo sería opcional).
idioma → si damos la posibilidad de elegir varios idiomas (inglés o castellano).
clave de acceso → (puede ser el DNI o una contraseña).

Información sobre el equipamiento

clave → Si existe una clave inherente a los productos, en caso de desarrollar el sistema para una empresa de material deportivo, se podría utilizar esta como clave.

nombre del equipamiento → para utilizarlo en las instrucciones ("coja las pesas / cintas / etcétera").
género → femenino o masculino, para la construcción de la frase ("la/el/los/las").
cualidades → anotaciones, por ejemplo si son pesas pues: de 1kg, de 4kg, …
(Tal vez como un atributo multislot que detalle distintas características.)
precio → especialmente si el entrenador deportivo está ubicado en el portal de una empresa dedicada a la venta de material deportivo.

Información sobre un ejercicio

clave → puede ser autogenerada
nombre del ejercicio → puede estar dentro de ciertas categorías (abdominales, etcétera).
descripción → texto(s), audio, imagen(es), etcétera.
grado de dificultad → dentro de una escala preestablecida.
origen → a elegir dentro de ciertas categorías (dar la posibilidad de crear categorías).
género → femenino o masculino, para la construcción de la frase ("la/el/los/las").

Información sobre una articulación

nombre de la articulación → toma de una lista (puede ser clave).
lado y/o localización → si queremos distinguir la rodilla izquierda de la derecha por ejemplo o ayudar a localizarla dentro de la anatomía (en tal caso sería necesario incluir este campo en la clave).
tipo de articulación → debería autocompletarse en base a la articulación indicada.
género → femenino o masculino, para la construcción de la frase ("la/el/los/las").

Observación: Esta información tiene sentido si el entrenamiento puede ir dirigido a personas con un problema en alguna articulación, ya sea para evitar utilizarla o para fortalecerla.

Información sobre un músculo

nombre del músculo → se toma de una lista (puede ser clave).
grupo → debería autocompletarse en base a la articulación indicada (para evitar errores).
lado y/o localización → si queremos distinguir el tríceps izquierdo del derecho por ej (en tal caso sería necesario incluir este campo en la clave).
género → femenino o masculino, para la construcción de la frase ("la/el/los/las") (debería autocompletarse).

Observación: Incluir y distinguir el lado puede ser favorable para evitar utilizar músculos lesionados, pero complica la relación con el concepto ejercicio, pues este está descrito de forma genérica tanto para los músculos de un lado como para los del otro.
Estudiaremos cómo resolver este problema. Una posibilidad sería incluir el concepto de lesión en la BD.

Observación: En la BD se pueden incluir nuevos usuarios, equipamiento, ejercicios, etcétera Pero los datos sobre los músculos y articulaciones deben estar incluidos desde el principio. Puesto que se trata de un conjunto finito que no sufre alteraciones, en principio estos datos no se deben poder modificar.

Relaciones entre los distintos conceptos

- Será necesario saber de qué equipamiento dispone el usuario.
- También habrá que registrar qué ejercicios realiza, cuántos y en qué día(s) / hora(s). Esta información se podrá utilizar para realizar estadísticas y etcétera que pueden resultar de interés tanto para el cliente como para la mejora de la BD.
- Otro dato importante será el material necesario para realizar un ejercicio (aunque puede haber ejercicios que no requieran material específico).
- Será muy importante saber qué músculos, articulaciones se ven involucrados en la realización de un ejercicio (tal vez hasta en qué grado y de qué modo).

Para diseñar un entrenamiento equilibrado y completo sería interesante saber qué músculos son opuestos y así poder incluir en una sesión de entrenamiento ejercicios que se compensen.

1.13.3.POSIBLES CONSULTAS A LA BD

Aunque la base de datos pueda responder a muchas preguntas, hemos anotado algunas de las más relevantes para que nos sirvan de guía a la hora de hacer el diseño.

- ¿Qué Ejercicios puede realizar una Persona, teniendo en cuenta el Equipamiento de que dispone?
- ¿Qué Equipamiento requiere un Ejercicio?
- ¿Qué ejercicios ha realizado una persona?
- ¿Qué Articulaciones y Músculos involucra un Ejercicio?
- ¿Cuál o cuáles son los Músculos opuestos a uno dado? [Sólo nos interesan los nombres.]
- Dado un Ejercicio, hallar los Ejercicios que involucran a los Músculos opuestos a los ejercitados por el primer Ejercicio.
  [Por ejemplo, averiguar los ejercicios que fortalecen los músculos opuestos a los entrenados por cierto ejercicio de abdominales.]
- Dado un grupo muscular, obtener todos los Ejercicios que lo involucran.
- Obtener todos los Ejercicios de un determinado grado de dificultad.
- Dado un determinado origen, hallar todos los Ejercicios que provengan de él.
- Averiguar el número de repeticiones que una Persona ha realizado de un Ejercicio, en un intervalo de tiempo.
- Averiguar cuantos ejercicios hay en la BD.
- Averiguar de qué presupuesto deportivo disponen [por ejemplo] los hombres con sueldo mayor a 1000 euros, como media.

- Averiguar la media de repeticiones de abdominales que ha realizado el usuario con DNI '10000000'.
- Creamos una vista sobre los datos de las personas incluidas en nuestra BD, pero permitiendo que sólo se vean aquellos que no son confidenciales.
- Creamos una vista en la que se ve el nombre de una persona, los ejercicios que ha realizado, en que fechas y el número de repeticiones.

Observación: Otro tipo de consulta interesante puede tener que ver con el dolor en algunas partes del cuerpo. Esto puede estar motivado porque los músculos son muy débiles, entonces podemos proponer ejercicios que fortalezco esos músculos.

Problema: No estaremos seguras de estar aconsejando bien, puesto que la causa del dolor puede ser muy diversa y hasta desconocida o mal interpretada por el usuario.

Ventaja: Comercialmente sería útil saber qué tipo de problemas físicos tienen los usuarios y con qué ejercicios tratan de resolverlos, para desarrollar o mejorar los utensilios que mejoren estas zonas.

1.13.4.FASE 1: DISEÑO DE LA BD: 1ª VERSIÓN

1.13.5.DIAGRAMA ENTIDAD-RELACIÓN

La idea de un sistema informático relacionado con el deporte es algo que siempre nos ha interesado, por lo que ya habíamos realizado algún acercamiento a lo que debería ser el diseño de la base de datos en otras asignaturas. Partimos de ése diseño para llegar a uno adaptado a nuestro proyecto.

1.13.6.ESQUEMA RELACIONAL

Observación: Las claves están subrayadas y las claves ajenas están en un círculo.

Persona (DNI, nombreP, fechaNac, sexo, presupuestoDeport)
Equipamiento (nombreEq, cualidades)
Ejercicio (nombreEj, descripción, imagen, gradoDificultad, origen)
Articulación (nombreA, tipoArticulación)
Músculo (nombreM, grupo)

DisponeDe (DNI, nombreEq)

Realiza (DNI, nombreEj, fecha, numRepeticiones)

Requiere (nombreEq, nombreEj)

Involucra (nombreEj, nombreA, nombreM, modo, grado)

Opuesto (nombreM1, nombreM2)

1.13.7.RESTRICCIONES

La relación Realiza contiene un atributo fecha que determina el día en que una Persona practicó un ejercicio. Lógicamente, el valor de ése atributo fecha debe ser mayor que el del atributo fechaNac de la Persona que realizó el Ejercicio.

1.13.8.FASE 2: DISEÑO DE LA BD: 2ª VERSIÓN

Modificamos algunas características y añadimos otras nuevas para obtener progresivamente una BD que se ajustara a las necesidades de nuestro problema.

1.13.9.MODIFICACIONES RESPECTO A LA VERSIÓN ANTERIOR

Campos añadidos

Persona
forma física: nos ayuda a clasificar a las personas por principiante, experto,… y así recomendarle ejercicios más adecuados a sus características.
fecha de alta: puede ser interesante distinguir a los usuarios más antiguos o simplemente saber desde cuándo siguen los entrenamientos diseñados por el generador de presentaciones.

clave de acceso: para entrar en la aplicación. También cabe la posibilidad de entrar como visitante, pero con otros permisos. Aun no lo hemos decidido.
idioma: no estaba incluido en la versión anterior.
tipo: distingue si se trata de un cliente o un experto.
alergias: se eligen de una lista desplegable.

Equipamiento
clave, **género** y **precio** por razones que ya hemos indicado en la introducción. Además, sustituimos el campo cualidades por los siguientes:
peso: describe el peso del aparato.
material: describe el material. Consideramos este dato para evitar alergias.
otros: resume otras características**.**

Ejercicio
clave: por razones que ya hemos indicado en la introducción.
género: por razones que ya hemos indicado en la introducción.
tiempo de una repetición: porque es necesario saber cuánto dura, aproximadamente, una repetición para poder calcular el tiempo total de cada ejercicio en función de las repeticiones y con eso (y otros tiempos) la duración de la sesión de entrenamiento.
Además, hemos quitado el campo "descripción" que era muy genérico y hemos creado cuatro nuevas Entidades relacionadas con el ejercicio: texto, audio, imagen y video.

Músculo, Hueso y Articulación
Localización: especifica la zona en la que se encuentran. En los casos que había grupo lo hemos sustituido por este atributo. Los posibles valores de localización son cabeza, tronco, extremidad superior y extremidad inferior.

Realiza
El campo **Fecha** lo hemos a cambiado a tipo Fecha/Hora, para permitir la realización de más de una sesión de entrenamiento al día.

Involucra
Al haber creado Hueso como nueva entidad, Involucra necesita también NombreH como clave ajena.

Opuesto pasa a llamarse Antagonista.

Las nuevas entidades

Texto
claveTexto: necesaria (puede ser autogenerada).
ficheroTexto: el propio texto.
idiomaTexto: el idioma del propio texto.
tipoTexto: imprescindible, adicional, explicativo, etcétera.

Observación: esto puede cobrar especial importancia si la pantalla por la que se ve a mostrar el ejercicio es muy reducida. Además de servir como criterio de selección

también nos puede indicar dónde ubicar el texto dentro del interfaz, por tanto es importante distinguir los distintos tipos de texto que pueden ir asociados aun ejercicio.
Audio
claveAudio: necesaria (puede ser autogenerada).
ficheroAudio: el propio fichero de audio.
idiomaAudio: el idioma del propio audio.
tipoAudio: música o narración (autogenerada a partir de texto o de un narrador).

Imagen
claveImagen: necesaria (puede ser autogenerada).
ficheroImagen: el propio fichero con la imagen.
tipoImagen: imprescindible, adicional, explicativo, etcétera.

Video
claveVideo: necesaria (puede ser autogenerada).
ficheroVideo: el propio fichero de video.
idiomaVideo: el idioma del propio video.
tipoVideo: imprescindible, adicional, explicativo, etcétera.

Hueso
nombreH: toma de una lista (puede ser clave).
localización: este campo describe la zona en la que se encuentra el hueso. Por ejemplo, mano.

Nuevas relaciones

Explicado relaciona Texto con Ejercicio.

Contado relaciona Audio con Ejercicio.

Visualizado relaciona Imagen con Ejercicio.

Animado relaciona Vídeo con Ejercicio.

Amplia Ejercicio se relaciona consigo mismo mediante "Amplia". Esta relación tiene un nuevo atributo "efecto". Amplia relaciona un ejercicio con un ejercicio auxiliar o explicativo. Este ejercicio puede ser contraproducente y se mostrará para apoyar la explicación de la correcta realización del ejercicio. Por ello hemos incluido el campo efecto.

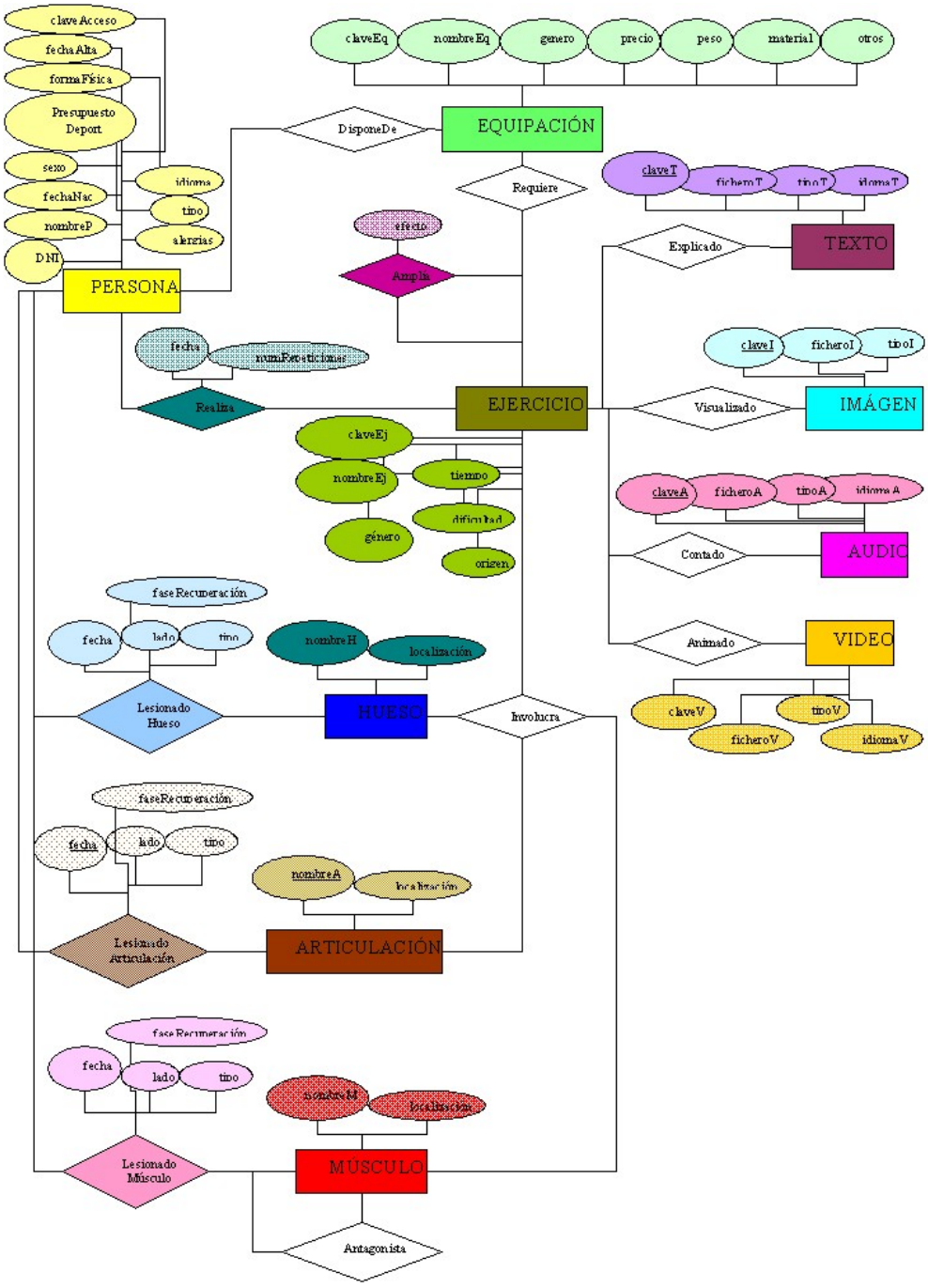
Lesionado Hueso Relación entre las entidades Persona y Hueso. Campos: fecha, lado, tipo, faseRecuperación.
Fecha: especifica la fecha de la lesión.
Lado: izquierdo/derecho.
Tipo: fisura, fractura, etcétera.
FaseRecuperación: recuperado, reposo, tratamiento, etcétera.

Lesionado Músculo

Relación entre las entidades Persona y Músculo. Campos: fecha, lado, tipo, faseRecuperación.
Fecha: especifica la fecha de la lesión.
Lado: izquierdo/derecho.
Tipo: inflamado, etcétera.
FaseRecuperación: recuperado, reposo, tratamiento...

Lesionado Articulación
Relación entre las entidades Persona y Articulación. Campos: fecha, lado, tipo, faseRecuperación.
Fecha: especifica la fecha de la lesión.
Lado: izquierdo/derecho.
Tipo: torcedura, esguince, etcétera.
FaseRecuperación: recuperado, reposo, tratamiento...

1.13.10.SIGUIENTE DIAGRAMA ENTIDAD-RELACIÓN



1.13.11.SIGUIENTE ESQUEMA RELACIONAL

Persona (DNI, nombreP, fechaNac, sexo, presupuestoDeport, forma física, fecha de alta, claveAcceso, idioma, tipo, alergias)
Equipamiento (claveEq, nombreEq, género, precio, peso, material, otros)
Ejercicio (claveEj, nombreEj, género, tiempo de una repetición, dificultad, origen)

Articulación (nombreA, tipoArticulación, localización)
Músculo (nombreM, localización)
Hueso (nombreH, localización)
Texto (claveTexto, ficheroTexto, tipoTexto, idiomaTexto)
Imagen (claveImagen, ficheroImagen, tipoImagen)
Audio (claveAudio, ficheroAudio, tipoAudio, idiomaAudio).
Video (claveVideo, ficheroVideo, tipoVideo, idiomaVideo)

DisponeDe (DNI, claveEq)
Realiza (DNI, claveEj, fecha, numRepeticiones)
Requiere (claveEq, claveEj)
Involucra (claveEj, nombreA, nombreM, nombreH, modo, grado)
Opuesto (nombreM1, nombreM2)
Explicado(ClaveEj, ClaveT)
Contado(ClaveEj, ClaveA)
Visualizado(ClaveEj, ClaveI)
Animado (ClaveEj, ClaveV)
Amplia (claveEj1, claveEj2, efecto)
LesionadoHueso (fecha, lado, tipo, faseRecuperación, DNI, nombreH)
LesionadoMusculo (fecha, lado, tipo, faseRecuperación, DNI, nombreM)
LesionadoArticulación (fecha, lado, tipo, faseRecuperación, DNI, nombreA)

1.13.12.OBSERVACIONES

Este es el esquema con el que empezamos a trabajar, pero luego sufrió una gran cantidad de modificaciones debido a cuestiones, principalmente de implementación. Estos cambios y la base de datos final están explicados en el apartado correspondiente dentro del tema de desarrollo.

1.14.DESARROLLO DE LA ONTOLOGÍA

La ontología que ocupa nuestro proyecto no está basada en otra ontología existente, sino en un diseño original que adapta las principales características de clasificación de un conjunto de entrenamientos y cuyo objetivo principal es usarse junto con la herramienta jCOLIBRI en nuestro sistema CBR experto. Esta ontología clasifica un conjunto de entrenamientos deportivos según las características principales de los ejercicios que incluyen.

Su diseño distingue: diferentes tipos de entrenamiento, las partes de un entrenamiento, ejercicios que componen un entrenamiento, las partes del cuerpo que ejercitan, etc.

Para la creación de la ontología partimos inicialmente del tutorial de la pizzería que ofrece Protégé. Más tarde cambiamos el diseño y comenzamos desde cero ya que el campo de conocimiento sobre el que versa nuestra ontología pertenece un campo muy específico y no disponíamos de ningún ejemplo que incluyese todas las características que necesitábamos. Por este motivo, hemos ido desarrollando varias versiones que se han ampliado y modificado progresivamente bajo la supervisión de Belén Díaz, profesora de Ingeniería de Sistemas Basados en el Conocimiento. El proceso ha sido largo debido a que las necesidades del sistema CBR se han modificado en el transcurso de este tiempo.

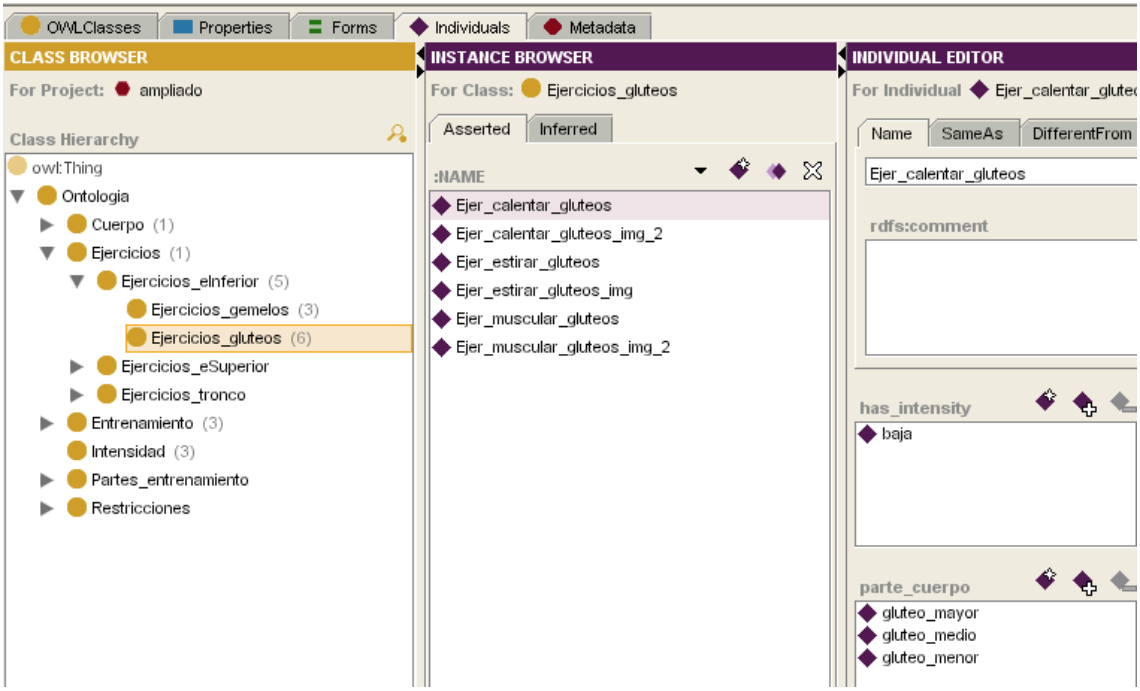
La estructura principal de la ontología está compuesta de las siguientes subclases:

- Cuerpo: clasifica los principales músculos según la zona del cuerpo donde se encuentran.
- Intensidad: detallan las distintas intensidades que pueden tener los ejercicios de un entrenamiento.
- Entrenamiento: distingue los distintos tipos de entrenamiento que conocemos.
- Partes de un entrenamiento: explica las distintas partes que componen un entrenamiento.
- Ejercicios: clasificación de los ejercicios de un entrenamiento según la clasificación que hemos hecho de los músculos del cuerpo.
- Restricciones: límite del número de ejercicios.

La siguiente figura muestra la estructura principal de la ontología.

Ahora, profundizaremos un poco más en cada clase. Empezamos por la clase Cuerpo que está dividida en las siguientes zonas: cabeza, tronco, extremidades superiores y extremidades inferiores. Cada una de estas zonas está instanciada con los músculos más característicos de las mismas.

Veamos la clasificación de Ejercicios, que es más completa:



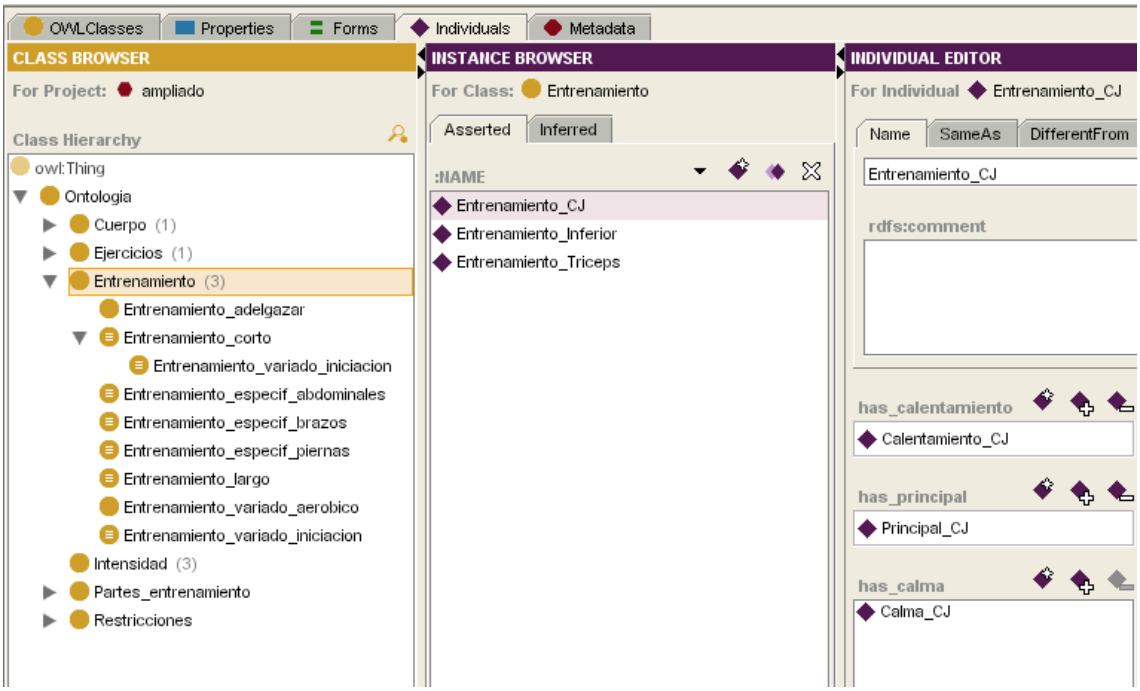
Hemos dividido los ejercicios según la zona del cuerpo que implican, para esto usamos la primera clasificación que definimos: Cuerpo. También los clasificamos según su nivel de intensidad, más tarde los incluiremos en la parte del entrenamiento en la que se aplican para formar un entrenamiento completo.

Hemos asociado las instancias de ejercicios en varias clases para facilitar la adaptación de un entrenamiento cuando usemos la ontología en el sistema CBR. Veamos esto con un ejemplo.
Supongamos que tenemos un entrenamiento muy sencillo para fortalecer los bíceps. Este entrenamiento se compone de un conjunto de ejercicios: el primero para calentar los bíceps. Después, una secuencia de ejercicios para fortalecerlo (parte principal del entrenamiento) y finalmente otro conjunto de ejercicios para volver a la calma estirando el bíceps. La modificación deseable en nuestro sistema será que si un usuario padece una lesión en el bíceps y el sistema CBR reconoce como caso más similar posible uno que recomienda ejercitarlos, la modificación se pueda dar gracias a las relaciones, proporcionando ejercicios de calentamiento y vuelta a la calma que impliquen directamente los mismos músculos que aquellos que se han trabajado durante la parte principal del entrenamiento. En este caso, como el usuario tenía una lesión en el bíceps, el sistema le recomendará un entrenamiento lo más similar posible, pero sustituyendo los ejercicios que ejercitan este músculo en la parte de calentamiento y vuelta a la calma

por otros que trabajen el músculo más similar según la definición de la ontología. Lo mismo ocurrirá con los de la parte principal del entrenamiento.

Del mismo modo, esta ontología define las relaciones necesarias para dar lugar a un entrenamiento coherente, es decir, un entrenamiento que caliente y haga vuelta a la calma de los músculos trabajados en la parte principal.

Según la clasificación que hemos explicado y que se puede observar a continuación podríamos seguir creando subclases indefinidamente, añadiendo más músculos. Por este motivo podemos decir que hemos hecho una ontología fácilmente ampliable con conocimiento experto del dominio. Análogamente la clasificación de los entrenamientos también es fácilmente ampliable aumentando el conocimiento experto que poseemos del sistema. De la misma forma, las partes del entrenamiento también se pueden ampliar, añadiendo más instancias.



Las restricciones se incluyeron para limitar el número de ejercicios de las partes de los entrenamientos por la hipótesis del mundo abierto, para que el sistema fuese capaz de realizar inferencias.

Podemos clasificar las clases que describe nuestra ontología dentro de dos categorías principales: primitivas y definidas. Las clases primitivas son aquellas que tienen restricciones que delimitan su definición de forma necesaria pero no suficiente en el sentido de la pertenencia de un miembro a dicha clase. Podemos decir que una categoría de una ontología es primitiva si no puede definirse en términos de otras categorías en la misma ontología. El significado de una primitiva no está determinado por una definición con una forma cerrada, sino por axiomas que especifican cómo se relaciona con otras primitivas. En el caso de las clases definidas las restricciones tienen carácter de necesarias y suficientes.

Veamos un ejemplo de estas clases en nuestra ontología. La clase entrenamiento es un concepto primitivo ya que sólo tiene condiciones necesarias. Mientras que todas las subclases de entrenamiento (entrenamiento adelgazar, entrenamiento corto, entrenamiento específico abdominales, entrenamiento específico de brazos, entrenamientos específico de piernas, entrenamiento largo, entrenamiento suave, entrenamiento variado aeróbico, entrenamiento variado iniciación) son definidas, ya que tienen condiciones tanto necesarias como suficientes.

Debido a lo sencillo que resulta clasificar las clases como primitivas o definidas según las pautas que hemos dado y a la extensión de nuestra ontología creemos que es suficiente con citar un ejemplo de ambas. Para conocer exactamente a qué tipo pertenece cada clase, remitimos al lector a la ontología en sí. Una vez allí, bastará con pasar el cursor por encima de cada clase para ver si sus condiciones son necesarias o necesarias y suficientes. Así podemos catalogar la clase como primitiva o como definida respectivamente.

Pruebas de las capacidades de razonamiento del sistema

La ontología actual clasifica entrenamientos con ayuda de Racer. Las pruebas de inferencia de instancias demandan bastante tiempo (varios minutos) porque el volumen que tomó la ontología en una de las últimas versiones hace que el razonador trabaje bastante.

El sistema está preparado para poder adaptar entrenamientos en caso de no poder trabajar con un músculo en concreto. Como introdujimos anteriormente el entrenamiento está dividido en 3 partes: calentamiento, parte principal y vuelta a la calma. El calentamiento y la vuelta a la calma dependen de la parte principal (gracias a una relación de dependencia), para poder seleccionar un entrenamiento sólo indicando la parte principal, y que el sistema elija un calentamiento y una vuelta a la calma adecuado para el mismo.

Nótese que aunque el concepto es que el calentamiento y la vuelta a la calma dependen de la parte principal, en la ontología está indicado en sentido contrario (la parte principal depende de los otros dos).

Puntos fuertes de la ontología

El principal es que es un gran reflejo de conocimiento experto útil y real para clasificar un problema complejo. Esta clasificación es tan amplia que abarca desde músculos del cuerpo humano hasta detalles de ejercicios pasando por conceptos como intensidad y partes de un entrenamiento. La gestión de toda esta información y la necesidad de clasificarla en varios niveles jerárquicos así como las relaciones y restricciones entre diversas clases e instancias hace que sea un buen ejemplo de ontología y un trabajo original adecuado a las características de nuestro sistema CBR.

Dificultades

No se ha podido incluir explícitamente la división del entrenamiento en parte principal, calentamiento y vuelta a la calma debido a la hipótesis del mundo abierto. Hemos tenido ciertas dificultades debido a esta hipótesis, ya que no inferenciaba ciertas clases que para nosotras era obvio que debían aparecer.

Ampliaciones

Va a ser necesaria la ampliación de la ontología para el uso en el proyecto, aunque la estructura básica está hecha. Para ello hay que incluir más clases de entrenamiento, más instancias, y añadir sus dependencias con los calentamientos y vueltas a la calma.

Las ampliaciones más sencillas son las indicadas con anterioridad: ampliación de las subclases e instancias de los ejercicios y los entrenamientos.

Cuando probamos la ontología con la nueva versión de jCOLIBRI que incorpora el manejo de lógica descriptiva nos dimos cuenta que el sistema detectaba inconsistencias en la ontología. Después de evaluar los motivos y de realizar diversos cambios, dedujimos que era un problema de tamaño de la ontología. Según esto, la capacidad máxima de razonamiento que permite esta versión de jCOLIBRI con esta ontología concreta es la que presentamos. Si añadimos un ejercicio más a cada una de las partes del entrenamiento el sistema comienza a fallar. Dejamos este apartado abierto a la investigación para analizar en profundidad las causas y encontrar una solución a este inconveniente.

5. FASE DE DESARROLLO

1.15.BASE DE DATOS

1.15.1.PROBLEMAS Y SOLUCIONES

En el apartado sobre la adquisición de conocimiento explicamos nuestros primeros acercamientos al diseño de la base de datos. Al tratarse de un proceso incremental, obtuvimos multitud de versiones intermedias que solamente se diferenciaban entre ellas en algunos detalles. Consideramos que no merece la pena describir todos estos cambios menores, de modo que hemos optado por resumir los problemas más graves que encontramos a lo largo del proceso de desarrollo y documentar las soluciones que les dimos.

Como hemos explicado con anterioridad todo el proceso de desarrollo de la base de datos se ha hecho usando SQLYog. A la hora de crear relaciones entre tablas usando esta herramienta nos dimos cuenta que sólo permitía relaciones con un tipo de tablas concretas: las InnoDB. Esta apreciación no nos resultó obvia desde un primer momento, ya que nunca nos habíamos enfrentado a un problema semejante en nuestra experiencia previa con bases de datos.

El resto de las modificaciones que ha sufrido la base de datos están directamente relacionadas con los cambios del diseño o las demandas del sistema CBR. Así pues la motivación de las nuevas tablas introducidas y sus relaciones correspondientes respecto de la segunda versión son las siguientes:

- Estado físico: guarda en la BD los campos relativos a las características físicas de un usuario.
- Tiene estado: relaciona la tabla anterior con el usuario concreto que tiene ese estado físico (se puede dar que más de un usuario tenga el mismo estado) y con la fecha en la que presenta el estado. El estado físico de un usuario puede variar a lo largo del tiempo en función del entrenamiento o de diversos aspectos externos.

Motivación: necesitábamos incluir estas dos tablas en el sistema porque queríamos que el sistema permitiera al usuario ver su evolución.

- Idencaso: esta tabla también relaciona el usuario con su estado físico y le asigna una clave autonúmerica.

Motivación: jCOLIBRI necesita que todos los casos estén identificados por un identificador de caso (caseId). El sistema absorbe este dato para su uso interno sin que sea necesario mapearlo con ningún caso de la estructura de casos. Nótese, que no es necesario incluir la fecha del estado físico del usuario debido a que es el campo caseId quién lo identifica unívocamente.

- Sesión: tabla necesaria para almacenar la información relativa a una sesión de entrenamiento.

- Entrena: relaciona la información de sesión con el usuario concreto y la fecha.

Motivación: estas dos tablas son necesarias porque la persistencia del sistema es en la BD. En otras palabras, es imprescindible almacenar la solución, en nuestro caso el entrenamiento, que devuelve el sistema CBR.

1.15.2.ESQUEMA FINAL DE LA BASE DE DATOS

El diseño de la versión final de nuestra base de datos ya satisface los requisitos del sistema. A continuación mostramos un esquema de todas las tablas y sus correspondientes campos.

1.15.3.LAS TABLAS DE LA BASE DE DATOS

La base de datos se llama "trainer" y sus tablas son:

amplia	entrena	Involucra	realiza
animado	equipamiento	lesionadoarticulacio	requiere
articulacion	estadofisico	n	sesion
audio	explicado	lesionadohueso	texto
contado	hueso	lesionadomusculo	tieneestado
disponede	idencaso	musculo	video
ejercicio	imagen	opuesto	visualizado
		persona	

1.15.4.DETALLES DE CADA TABLA

En las siguientes páginas presentamos un esquema por cada tabla, que recoge toda la información referente a los campos que la componen.

amplia

Fields

Field	Type	Collation	Null	Key	Default	Extra	Privileges	Comment
claveEj1	decimal(9,0)	(NULL)	NO	PRI			select,insert,update,references	
claveEj2	decimal(9,0)	(NULL)	NO	PRI			select,insert,update,references	
efecto	enum('positivo','negativo')	latin1_swedish_ci	YES		(NULL)		select,insert,update,references	

Indexes

Table	Non unique	Key name	Seq in index	Column name	Collation	Cardinality	Sub part	Packed	Null	Index type	Comment
amplia	0	PRIMARY	1	claveEj1	A	2	(NULL)	(NULL)		BTREE	
amplia	0	PRIMARY	2	claveEj2	A	2	(NULL)	(NULL)		BTREE	
amplia	1	FK_amplia	1	claveEj2	A	2	(NULL)	(NULL)		BTREE	

Foreign Key Relationships

FK Id	Reference Table	Source Column	Target Column	Extra Info
amplia_ibfk_1	ejercicio	`claveEj1`	`claveEj`	ON DELETE NO ACTION ON UPDATE NO ACTION,
amplia_ibfk_2	ejercicio	`claveEj2`	`claveEj`	ON DELETE NO ACTION ON UPDATE NO ACTION

animado

Fields

Field	Type	Collation	Null	Key	Default	Extra	Privileges	Comment
claveEj	decimal(9,0)	(NULL)	NO	PRI			select,insert,update,references	
claveV	double	(NULL)	NO	PRI			select,insert,update,references	

Indexes

Table	Non unique	Key name	Seq in index	Column name	Collation	Cardinality	Sub part	Packed	Null	Index type	Comment
animado	0	PRIMARY	1	claveEj	A	2	(NULL)	(NULL)		BTREE	
animado	0	PRIMARY	2	claveV	A	2	(NULL)	(NULL)		BTREE	
animado	1	FK_animado	1	claveV	A	2	(NULL)	(NULL)		BTREE	

Foreign Key Relationships

FK Id	Reference Table	Source Column	Target Column	Extra Info
animado_ibfk_1	video	`claveV`	`claveV`	ON DELETE NO ACTION ON UPDATE NO ACTION,
animado_ibfk_2	ejercicio	`claveEj`	`claveEj`	ON DELETE NO ACTION ON UPDATE NO ACTION

articulacion

Fields

Field	Type	Collation	Null	Key	Default	Extra	Privileges	Comment
nombreA	varchar(100)	latin1_swedish_ci	NO	PRI			select,insert,update,references	
localizacion	enum('cabeza','tronco','eSuperior','eInferior')	latin1_swedish_ci	YES		(NULL)		select,insert,update,references	
tipoA	varchar(150)	latin1_swedish_ci	YES		(NULL)		select,insert,update,references	

Indexes

Table	Non unique	Key name	Seq in index	Column name	Collation	Cardinality	Sub part	Packed	Null	Index type	Comment
articulacion	0	PRIMARY	1	nombreA	A	56	(NULL)	(NULL)		BTREE	

audio

Fields

Field	Type	Collation	Null	Key	Default	Extra	Privileges	Comment
claveA	double	(NULL)	NO	PRI	(NULL)	auto_increment	select,insert,update,references	
ficheroA	blob	(NULL)	YES		(NULL)		select,insert,update,references	
tipoA	enum('1','2','3','4','5')	latin1_swedish_ci	YES		(NULL)		select,insert,update,references	
idioma	enum('castellano','ingles')	latin1_swedish_ci	YES		(NULL)		select,insert,update,references	

Indexes

Table	Non unique	Key name	Seq in index	Column name	Collation	Cardinality	Sub part	Packed	Null	Index type	Comment
audio	0	PRIMARY	1	claveA	A	1	(NULL)	(NULL)		BTREE	

contado

Fields

Field	Type	Collation	Null	Key	Default	Extra	Privileges	Comment
claveEj	decimal(9,0)	(NULL)	NO	PRI			select,insert,update,references	
claveA	double	(NULL)	NO	PRI			select,insert,update,references	

Indexes

Table	Non unique	Key name	Seq in index	Column name	Collation	Cardinality	Sub part	Packed	Null	Index type	Comment
contado	0	PRIMARY	1	claveEj	A	0	(NULL)	(NULL)		BTREE	
contado	0	PRIMARY	2	claveA	A	0	(NULL)	(NULL)		BTREE	
contado	1	FK_contado	1	claveA	A	0	(NULL)	(NULL)		BTREE	

Foreign Key Relationships

FK Id	Reference Table	Source Column	Target Column	Extra Info
contado_ibfk_1	audio	`claveA`	`claveA`	ON DELETE NO ACTION ON UPDATE NO ACTION,
contado_ibfk_2	ejercido	`claveEj`	`claveEj`	ON DELETE NO ACTION ON UPDATE NO ACTION

disponede

Fields

Field	Type	Collation	Null	Key	Default	Extra	Privileges	Comment
dni	decimal(9,0)	(NULL)	NO	PRI			select,insert,update,references	
claveEq	int(11)	(NULL)	NO	PRI			select,insert,update,references	

Indexes

Table	Non unique	Key name	Seq in index	Column name	Collation	Cardinality	Sub part	Packed	Null	Index type	Comment
disponede	0	PRIMARY	1	dni	A	4	(NULL)	(NULL)		BTREE	
disponede	0	PRIMARY	2	claveEq	A	4	(NULL)	(NULL)		BTREE	
disponede	1	FK_disponede	1	claveEq	A	4	(NULL)	(NULL)		BTREE	

Foreign Key Relationships

FK Id	Reference Table	Source Column	Target Column	Extra Info
disponede_ibfk_2	equipacion	`claveEq`	`claveEq`	ON DELETE NO ACTION ON UPDATE NO ACTION,
disponede_ibfk_3	persona	`dni`	`dni`	

ejercicio

Fields

Field	Type	Collation	Null	Key	Default	Extra	Privileges	Comment
claveEj	decimal(9,0)	(NULL)	NO	PRI			select,insert,update,references	
nombreEj	varchar(100)	latin1_swedish_ci	NO				select,insert,update,references	
genero	enum('femenino','masculino')	latin1_swedish_ci	NO				select,insert,update,references	
tiempo	float	(NULL)	NO		0		select,insert,update,references	
dificultad	enum('1','2','3','4','5')	latin1_swedish_ci	YES		(NULL)		select,insert,update,references	
origen	varchar(100)	latin1_swedish_ci	YES		(NULL)		select,insert,update,references	
repeticiones	decimal(9,0)	(NULL)	NO	PRI			select,insert,update,references	

Indexes

Table	Non unique	Key name	Seq in index	Column name	Collation	Cardinality	Sub part	Packed	Null	Index type	Comment
ejercido	0	PRIMARY	1	claveEj	A	114	(NULL)	(NULL)		BTREE	
ejercido	0	PRIMARY	2	repeticiones	A	114	(NULL)	(NULL)		BTREE	

entrena

Fields

Field	Type	Collation	Null	Key	Default	Extra	Privileges	Comment
claveSesion	double	(NULL)	NO	PRI			select,insert,update,references	
dni	decimal(9,0)	(NULL)	NO	PRI			select,insert,update,references	
fecha	datetime	(NULL)	NO	PRI			select,insert,update,references	

Indexes

Table	Non unique	Key name	Seq in index	Column name	Collation	Cardinality	Sub part	Packed	Null	Index type	Comment
entrena	0	PRIMARY	1	claveSesion	A	5	(NULL)	(NULL)		BTREE	
entrena	0	PRIMARY	2	dni	A	5	(NULL)	(NULL)		BTREE	
entrena	0	PRIMARY	3	fecha	A	5	(NULL)	(NULL)		BTREE	
entrena	1	FK_entrena	1	dni	A	5	(NULL)	(NULL)		BTREE	

Foreign Key Relationships

FK Id	Reference Table	Source Column	Target Column	Extra Info
entrena_ibfk_1	persona	`dni`	`dni`	ON DELETE NO ACTION ON UPDATE NO ACTION,
entrena_ibfk_2	sesion	`claveSesion`	`claveSesion`	ON DELETE NO ACTION ON UPDATE NO ACTION

equipacion

Fields

Field	Type	Collation	Null	Key	Default	Extra	Privileges	Comment
claveEq	int(11)	(NULL)	NO	PRI			select,insert,update,references	
nombreEq	varchar(50)	latin1_swedish_ci	NO				select,insert,update,references	
genero	enum('femenino','masculino')	latin1_swedish_ci	YES		(NULL)		select,insert,update,references	
precio	float	(NULL)	YES		(NULL)		select,insert,update,references	
pesoEquipadon	int(11)	(NULL)	YES		(NULL)		select,insert,update,references	peso dado en gramos
material	varchar(50)	latin1_swedish_ci	YES		(NULL)		select,insert,update,references	
otros	varchar(100)	latin1_swedish_ci	YES		(NULL)		select,insert,update,references	

Indexes

Table	Non unique	Key name	Seq in index	Column name	Collation	Cardinality	Sub part	Packed	Null	Index type	Comment
equipacion	0	PRIMARY	1	claveEq	A	3	(NULL)	(NULL)		BTREE	

Estadofisico

Fields

Field	Type	Collation	Null	Key	Default	Extra	Privileges	Comment
clave	double unsigned	(NULL)	NO	PRI	(NULL)	auto_increment	select,insert,update,references	
peso	int(11)	(NULL)	YES		(NULL)		select,insert,update,references	
altura	int(11)	(NULL)	YES		(NULL)		select,insert,update,references	altura expresada en cm.
pResistencia	int(11)	(NULL)	YES		(NULL)		select,insert,update,references	
pFlexibilidad	int(11)	(NULL)	YES		(NULL)		select,insert,update,references	
tSesion	int(11)	(NULL)	YES		(NULL)		select,insert,update,references	tiempo expresado en minutos
frecuencia	enum('1','2','3','4','5','6','7')	latin1_swedish_ci	YES		(NULL)		select,insert,update,references	numero de dias de entrenamientos a la semana: 1, una vez a la semana. 7 todos los dias
estado	enum('1','2','3','4','5')	latin1_swedish_ci	YES		(NULL)		select,insert,update,references	
objetivo	enum('perderGrasa','mejorarResistencia','aumentarVolumenMuscular','mejorarTonicidad','mejorarFlexibilidad','mejorarCondicionIntegral')	latin1_swedish_ci	YES		(NULL)		select,insert,update,references	
edad	int(11)	(NULL)	YES		(NULL)		select,insert,update,references	
IMC	int(11)	(NULL)	YES		(NULL)		select,insert,update,references	
pFuerzaSuperior	int(11)	(NULL)	YES		(NULL)		select,insert,update,references	
PFuerzaAbdominal	int(11)	(NULL)	YES		(NULL)		select,insert,update,references	
PFuerzaInferior	int(11)	(NULL)	YES		(NULL)		select,insert,update,references	

Indexes

Table	Non unique	Key name	Seq in index	Column name	Collation	Cardinality	Sub part	Packed	Null	Index type	Comment
estadofisico	0	PRIMARY	1	clave	A	5	(NULL)	(NULL)		BTREE	

Explicado

Fields

Field	Type	Collation	Null	Key	Default	Extra	Privileges	Comment
claveEj	decimal(9,0)	(NULL)	NO	PRI			select,insert,update,references	
claveT	double	(NULL)	NO	PRI			select,insert,update,references	

Indexes

Table	Non unique	Key name	Seq in index	Column name	Collation	Cardinality	Sub part	Packed	Null	Index type	Comment
explicado	0	PRIMARY	1	claveEj	A	29	(NULL)	(NULL)		BTREE	
explicado	0	PRIMARY	2	claveT	A	29	(NULL)	(NULL)		BTREE	
explicado	1	FK_explicado	1	claveT	A	29	(NULL)	(NULL)		BTREE	

Foreign Key Relationships

FK Id	Reference Table	Source Column	Target Column	Extra Info
explicado_ibfk_1	texto	`claveT`	`claveT`	ON DELETE NO ACTION ON UPDATE NO ACTION,
explicado_ibfk_2	ejercicio	`claveEj`	`claveEj`	ON DELETE NO ACTION ON UPDATE NO ACTION

Hueso

Fields

Field	Type	Collation	Null	Key	Default	Extra	Privileges	Comment
nombreH	varchar(100)	latin1_swedish_ci	NO	PRI			select,insert,update,references	
localizadon	enum('cabeza','tronco','eSuperior','eInferior')	latin1_swedish_ci	YES		(NULL)		select,insert,update,references	

Indexes

Table	Non unique	Key name	Seq in index	Column name	Collation	Cardinality	Sub part	Packed	Null	Index type	Comment
hueso	0	PRIMARY	1	nombreH	A	57	(NULL)	(NULL)		BTREE	

idencaso

Fields

Field	Type	Collation	Null	Key	Default	Extra	Privileges	Comment
caseId	char(1)	latin1_swedish_ci	NO	PRI			select,insert,update,references	
dni	decimal(9,0)	(NULL)	YES	MUL	(NULL)		select,insert,update,references	
clave	double	(NULL)	YES	MUL	(NULL)		select,insert,update,references	

Indexes

Table	Non unique	Key name	Seq in index	Column name	Collation	Cardinality	Sub part	Packed	Null	Index type	Comment
idencaso	0	PRIMARY	1	caseId	A	4	(NULL)	(NULL)		BTREE	
idencaso	1	dni	1	dni	A	4	(NULL)	(NULL)	YES	BTREE	
idencaso	1	clave	1	clave	A	4	(NULL)	(NULL)	YES	BTREE	

Foreign Key Relationships

FK Id	Reference Table	Source Column	Target Column	Extra Info
idencaso_ibfk_1	persona	`dni`	`dni`	ON DELETE NO ACTION ON UPDATE NO ACTION,
idencaso_ibfk_2	estadofisico	`clave`	`clave`	ON DELETE NO ACTION ON UPDATE NO ACTION

imagen

Fields

Field	Type	Collation	Null	Key	Default	Extra	Privileges	Comment
claveI	double	(NULL)	NO	PRI	(NULL)	auto_increment	select,insert,update,references	
ficheroI	blob	(NULL)	YES		(NULL)		select,insert,update,references	
tipoI	enum('1','2','3','4','5')	latin1_swedish_ci	YES		(NULL)		select,insert,update,references	

Indexes

Table	Non unique	Key name	Seq in index	Column name	Collation	Cardinality	Sub part	Packed	Null	Index type	Comment
imagen	0	PRIMARY	1	claveI	A	1	(NULL)	(NULL)		BTREE	

involucra

Fields

Field	Type	Collation	Null	Key	Default	Extra	Privileges	Comment
claveEj	decimal(9,0)	(NULL)	NO	PRI			select,insert,update,references	
nombreA	varchar(100)	latin1_swedish_ci	NO	PRI			select,insert,update,references	
nombreM	varchar(100)	latin1_swedish_ci	NO	PRI			select,insert,update,references	
nombreH	varchar(100)	latin1_swedish_ci	NO	PRI			select,insert,update,references	
grado	enum('1','2','3','4','5')	latin1_swedish_ci	YES		(NULL)		select,insert,update,references	
modo	enum('1','2','3','4','5')	latin1_swedish_ci	YES		(NULL)		select,insert,update,references	

Indexes

Table	Non unique	Key name	Seq in index	Column name	Collation	Cardinality	Sub part	Packed	Null	Index type	Comment
involucra	0	PRIMARY	1	claveEj	A	0	(NULL)	(NULL)		BTREE	
involucra	0	PRIMARY	2	nombreA	A	0	(NULL)	(NULL)		BTREE	
involucra	0	PRIMARY	3	nombreM	A	0	(NULL)	(NULL)		BTREE	
involucra	0	PRIMARY	4	nombreH	A	0	(NULL)	(NULL)		BTREE	
involucra	1	FK_involucra_2	1	nombreA	A	0	(NULL)	(NULL)		BTREE	
involucra	1	FK_involucra	1	nombreH	A	0	(NULL)	(NULL)		BTREE	
involucra	1	FK_involucra_3	1	nombreM	A	0	(NULL)	(NULL)		BTREE	

Foreign Key Relationships

FK Id	Reference Table	Source Column	Target Column	Extra Info
involucra_ibfk_1	ejercicio	`claveEj`	`claveEj`	ON DELETE NO ACTION ON UPDATE NO ACTION,
involucra_ibfk_2	articulacion	`nombreA`	`nombreA`	
involucra_ibfk_3	hueso	`nombreH`	`nombreH`	ON DELETE NO ACTION ON UPDATE NO ACTION,
involucra_ibfk_4	musculo	`nombreM`	`nombreM`	ON DELETE NO ACTION ON UPDATE NO ACTION

lesionadoarticulacion

Fields

Field	Type	Collation	Null	Key	Default	Extra	Privileges	Comment
dni	decimal(9,0)	(NULL)	NO	PRI			select,insert,update,references	
nombreA	varchar(100)	latin1_swedish_ci	NO	PRI			select,insert,update,references	
fecha	date	(NULL)	NO	PRI			select,insert,update,references	
lado	enum('izquierdo','derecho')	latin1_swedish_ci	YES		(NULL)		select,insert,update,references	
tipo	varchar(100)	latin1_swedish_ci	YES		(NULL)		select,insert,update,references	
faseRecuperacion	varchar(100)	latin1_swedish_ci	YES		(NULL)		select,insert,update,references	

Indexes

Table	Non unique	Key name	Seq in index	Column name	Collation	Cardinality	Sub part	Packed	Null	Index type	Comment
lesionadoarticulacion	0	PRIMARY	1	dni	A	4	(NULL)	(NULL)		BTREE	
lesionadoarticulacion	0	PRIMARY	2	nombreA	A	4	(NULL)	(NULL)		BTREE	
lesionadoarticulacion	0	PRIMARY	3	fecha	A	4	(NULL)	(NULL)		BTREE	
lesionadoarticulacion	1	FK_lesionadoarticulacion	1	nombreA	A	4	(NULL)	(NULL)		BTREE	

Foreign Key Relationships

FK Id	Reference Table	Source Column	Target Column	Extra Info
lesionadoarticulacion_ibfk_1	persona	`dni`	`dni`	ON DELETE NO ACTION ON UPDATE NO ACTION,
lesionadoarticulacion_ibfk_2	articulacion	`nombreA`	`nombreA`	ON DELETE NO ACTION ON UPDATE NO ACTION

lesionadohueso

Fields

Field	Type	Collation	Null	Key	Default	Extra	Privileges	Comment
dni	decimal(9,0)	(NULL)	NO	PRI			select,insert,update,references	
nombreH	varchar(100)	latin1_swedish_ci	NO	PRI			select,insert,update,references	
fecha	date	(NULL)	NO	PRI			select,insert,update,references	
lado	enum('izquierdo','derecho')	latin1_swedish_ci	YES		(NULL)		select,insert,update,references	
tipo	varchar(100)	latin1_swedish_ci	YES		(NULL)		select,insert,update,references	
faseRecuperacion	varchar(100)	latin1_swedish_ci	YES		(NULL)		select,insert,update,references	

Indexes

Table	Non unique	Key name	Seq in index	Column name	Collation	Cardinality	Sub part	Packed	Null	Index type	Comment
lesionadohueso	0	PRIMARY	1	dni	A	4	(NULL)	(NULL)		BTREE	
lesionadohueso	0	PRIMARY	2	nombreH	A	4	(NULL)	(NULL)		BTREE	
lesionadohueso	0	PRIMARY	3	fecha	A	4	(NULL)	(NULL)		BTREE	
lesionadohueso	1	FK_lesionadohueso	1	nombreH	A	4	(NULL)	(NULL)		BTREE	

Foreign Key Relationships

FK Id	Reference Table	Source Column	Target Column	Extra Info
lesionadohueso_ibfk_1	hueso	`nombreH`	`nombreH`	ON DELETE NO ACTION ON UPDATE NO ACTION,
lesionadohueso_ibfk_2	persona	`dni`	`dni`	ON DELETE NO ACTION ON UPDATE NO ACTION

lesionadomusculo

Fields

Field	Type	Collation	Null	Key	Default	Extra	Privileges	Comment
dni	decimal(9,0)	(NULL)	NO	PRI			select,insert,update,references	
nombreM	varchar(100)	latin1_swedish_ci	NO	PRI			select,insert,update,references	
fecha	date	(NULL)	NO	PRI			select,insert,update,references	
lado	enum('izquierdo','derecho')	latin1_swedish_ci	YES		(NULL)		select,insert,update,references	
tipo	varchar(100)	latin1_swedish_ci	YES		(NULL)		select,insert,update,references	
faseRecuperacion	varchar(100)	latin1_swedish_ci	YES		(NULL)		select,insert,update,references	

Indexes

Table	Non unique	Key name	Seq in index	Column name	Collation	Cardinality	Sub part	Packed	Null	Index type	Comment
lesionadomusculo	0	PRIMARY	1	dni	A	4	(NULL)	(NULL)		BTREE	
lesionadomusculo	0	PRIMARY	2	nombreM	A	4	(NULL)	(NULL)		BTREE	
lesionadomusculo	0	PRIMARY	3	fecha	A	4	(NULL)	(NULL)		BTREE	
lesionadomusculo	1	FK_lesionadomusculo	1	nombreM	A	4	(NULL)	(NULL)		BTREE	

Foreign Key Relationships

FK Id	Reference Table	Source Column	Target Column	Extra Info
lesionadomusculo_ibfk_1	musculo	`nombreM`	`nombreM`	ON DELETE NO ACTION ON UPDATE NO ACTION,
lesionadomusculo_ibfk_2	persona	`dni`	`dni`	ON DELETE NO ACTION ON UPDATE NO ACTION

musculo

Fields

Field	Type	Collation	Null	Key	Default	Extra	Privileges	Comment
nombreM	varchar(100)	latin1_swedish_ci	NO	PRI			select,insert,update,references	
localizacion	enum('cabeza','tronco','eSuperior','eInferior')	latin1_swedish_ci	YES		(NULL)		select,insert,update,references	

Indexes

Table	Non unique	Key name	Seq in index	Column name	Collation	Cardinality	Sub part	Packed	Null	Index type	Comment
musculo	0	PRIMARY	1	nombreM	A	162	(NULL)	(NULL)		BTREE	

opuesto

Fields

Field	Type	Collation	Null	Key	Default	Extra	Privileges	Comment
nombreM1	varchar(100)	latin1_swedish_ci	NO	PRI			select,insert,update,references	
nombreM2	varchar(100)	latin1_swedish_ci	NO	PRI			select,insert,update,references	

Indexes

Table	Non unique	Key name	Seq in index	Column name	Collation	Cardinality	Sub part	Packed	Null	Index type	Comment
opuesto	0	PRIMARY	1	nombreM1	A	0	(NULL)	(NULL)		BTREE	
opuesto	0	PRIMARY	2	nombreM2	A	0	(NULL)	(NULL)		BTREE	
opuesto	1	FK_opuesto	1	nombreM2	A	0	(NULL)	(NULL)		BTREE	

Foreign Key Relationships

FK Id	Reference Table	Source Column	Target Column	Extra Info
opuesto_ibfk_1	musculo	`nombreM1`	`nombreM`	ON DELETE NO ACTION ON UPDATE NO ACTION,
opuesto_ibfk_2	musculo	`nombreM2`	`nombreM`	ON DELETE NO ACTION ON UPDATE NO ACTION

persona

Fields

Field	Type	Collation	Null	Key	Default	Extra	Privileges	Comment
dni	decimal(9,0)	(NULL)	NO	PRI			select,insert,update,references	
nombreP	varchar(100)	latin1_swedish_ci	YES		(NULL)		select,insert,update,references	
fechaNac	date	(NULL)	YES		(NULL)		select,insert,update,references	
sexo	enum('femenino','masculino')	latin1_swedish_ci	YES		(NULL)		select,insert,update,references	
presupuesto	float	(NULL)	YES		(NULL)		select,insert,update,references	
fechaAlta	date	(NULL)	YES		(NULL)		select,insert,update,references	
claveAcceso	varchar(10)	latin1_swedish_ci	YES		(NULL)		select,insert,update,references	
idioma	enum('castellano','ingles')	latin1_swedish_ci	YES		(NULL)		select,insert,update,references	
tipo	enum('experto','cliente','administrador')	latin1_swedish_ci	YES		(NULL)		select,insert,update,references	
email	varchar(100)	latin1_swedish_ci	YES		(NULL)		select,insert,update,references	
altura	decimal(3,2)	(NULL)	NO	PRI			select,insert,update,references	

Indexes

Table	Non unique	Key name	Seq in index	Column name	Collation	Cardinality	Sub part	Packed	Null	Index type	Comment
persona	0	PRIMARY	1	dni	A	15	(NULL)	(NULL)		BTREE	
persona	0	PRIMARY	2	altura	A	15	(NULL)	(NULL)		BTREE	

realiza

Fields

Field	Type	Collation	Null	Key	Default	Extra	Privileges	Comment
dni	decimal(9,0)	(NULL)	NO	PRI			select,insert,update,references	
claveEj	decimal(9,0)	(NULL)	NO	PRI			select,insert,update,references	
fecha	datetime	(NULL)	NO	PRI			select,insert,update,references	
numRepeticiones	int(11)	(NULL)	YES		(NULL)		select,insert,update,references	

Indexes

Table	Non unique	Key name	Seq in index	Column name	Collation	Cardinality	Sub part	Packed	Null	Index type	Comment
realiza	0	PRIMARY	1	dni	A	3	(NULL)	(NULL)		BTREE	
realiza	0	PRIMARY	2	claveEj	A	3	(NULL)	(NULL)		BTREE	
realiza	0	PRIMARY	3	fecha	A	3	(NULL)	(NULL)		BTREE	
realiza	1	FK_realiza	1	claveEj	A	3	(NULL)	(NULL)		BTREE	

Foreign Key Relationships

FK Id	Reference Table	Source Column	Target Column	Extra Info
realiza_ibfk_1	persona	`dni`	`dni`	ON DELETE NO ACTION ON UPDATE NO ACTION,
realiza_ibfk_2	ejercicio	`claveEj`	`claveEj`	ON DELETE NO ACTION ON UPDATE NO ACTION

Requiere

Fields

Field	Type	Collation	Null	Key	Default	Extra	Privileges	Comment
claveEq	int(11)	(NULL)	NO	PRI			select,insert,update,references	
claveEj	decimal(9,0)	(NULL)	NO	PRI			select,insert,update,references	

Indexes

Table	Non unique	Key name	Seq in index	Column name	Collation	Cardinality	Sub part	Packed	Null	Index type	Comment
requiere	0	PRIMARY	1	claveEq	A	1	(NULL)	(NULL)		BTREE	
requiere	0	PRIMARY	2	claveEj	A	1	(NULL)	(NULL)		BTREE	
requiere	1	FK_requiere	1	claveEj	A	1	(NULL)	(NULL)		BTREE	

Foreign Key Relationships

FK Id	Reference Table	Source Column	Target Column	Extra Info
requiere_ibfk_1	equipacion	`claveEq`	`claveEq`	ON DELETE NO ACTION ON UPDATE NO ACTION,
requiere_ibfk_2	ejercicio	`claveEj`	`claveEj`	ON DELETE NO ACTION ON UPDATE NO ACTION

Sesion

Fields

Field	Type	Collation	Null	Key	Default	Extra	Privileges	Comment
claveSesion	double	(NULL)	NO	PRI	(NULL)	auto_increment	select,insert,update,references	
entrenamiento	text	latin1_swedish_ci	YES		(NULL)		select,insert,update,references	

Indexes

Table	Non unique	Key name	Seq in index	Column name	Collation	Cardinality	Sub part	Packed	Null	Index type	Comment
sesion	0	PRIMARY	1	claveSesion	A	3	(NULL)	(NULL)		BTREE	

Texto

Fields

Field	Type	Collation	Null	Key	Default	Extra	Privileges	Comment
claveT	double	(NULL)	NO	PRI	(NULL)	auto_increment	select,insert,update,references	
ficheroT	blob	(NULL)	YES		(NULL)		select,insert,update,references	
tipoT	enum('1','2','3','4','5')	latin1_swedish_ci	YES		(NULL)		select,insert,update,references	
idioma	enum('castellano','ingles')	latin1_swedish_ci	YES		(NULL)		select,insert,update,references	

Indexes

Table	Non unique	Key name	Seq in index	Column name	Collation	Cardinality	Sub part	Packed	Null	Index type	Comment
texto	0	PRIMARY	1	claveT	A	29	(NULL)	(NULL)		BTREE	

Tieneestado

Fields

Field	Type	Collation	Null	Key	Default	Extra	Privileges	Comment
dni	decimal(9,0)	(NULL)	NO	PRI			select,insert,update,references	
clave	double	(NULL)	NO	PRI			select,insert,update,references	
fecha	datetime	(NULL)	YES		(NULL)		select,insert,update,references	

Indexes

Table	Non unique	Key name	Seq in index	Column name	Collation	Cardinality	Sub part	Packed	Null	Index type	Comment
tieneestado	0	PRIMARY	1	dni	A	4	(NULL)	(NULL)		BTREE	
tieneestado	0	PRIMARY	2	clave	A	4	(NULL)	(NULL)		BTREE	
tieneestado	1	FK_tieneestado	1	clave	A	4	(NULL)	(NULL)		BTREE	

Foreign Key Relationships

FK Id	Reference Table	Source Column	Target Column	Extra Info
tieneestado_ibfk_1	persona	`dni`	`dni`	ON DELETE NO ACTION ON UPDATE NO ACTION,
tieneestado_ibfk_2	estadofisico	`clave`	`clave`	ON DELETE NO ACTION ON UPDATE NO ACTION

video

Fields

Field	Type	Collation	Null	Key	Default	Extra	Privileges	Comment
claveV	double	(NULL)	NO	PRI	(NULL)	auto_increment	select,insert,update,references	
ficheroV	blob	(NULL)	YES		(NULL)		select,insert,update,references	
tipoV	enum('1','2','3','4','5')	latin1_swedish_ci	YES		(NULL)		select,insert,update,references	
idioma	enum('castellano','ingles')	latin1_swedish_ci	YES		(NULL)		select,insert,update,references	

Indexes

Table	Non unique	Key name	Seq in index	Column name	Collation	Cardinality	Sub part	Packed	Null	Index type	Comment
video	0	PRIMARY	1	claveV	A	1	(NULL)	(NULL)		BTREE	

visualizado

Fields

Field	Type	Collation	Null	Key	Default	Extra	Privileges	Comment
claveEj	decimal(9,0)	(NULL)	NO	PRI			select,insert,update,references	
claveI	double	(NULL)	NO	PRI			select,insert,update,references	

Indexes

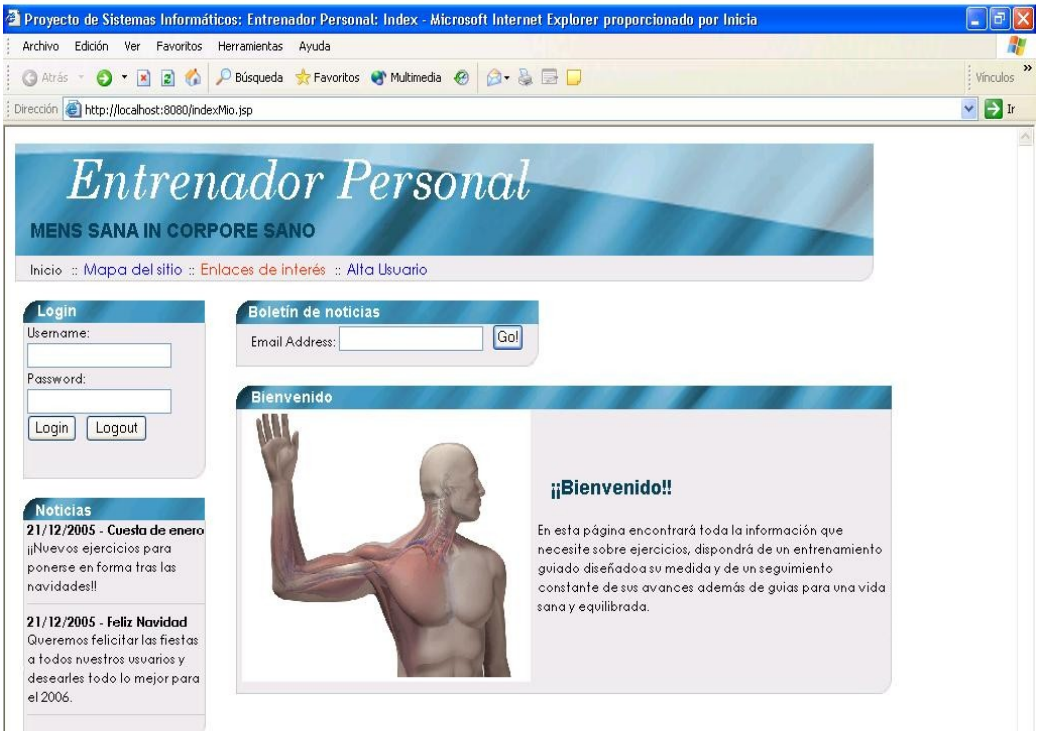
Table	Non unique	Key name	Seq in index	Column name	Collation	Cardinality	Sub part	Packed	Null	Index type	Comment
visualizado	0	PRIMARY	1	claveEj	A	2	(NULL)	(NULL)		BTREE	
visualizado	0	PRIMARY	2	claveI	A	2	(NULL)	(NULL)		BTREE	
visualizado	1	FK_visualizado	1	claveI	A	2	(NULL)	(NULL)		BTREE	

Foreign Key Relationships

FK Id	Reference Table	Source Column	Target Column	Extra Info
visualizado_ibfk_2	ejercicio	`claveEj`	`claveEj`	ON DELETE NO ACTION ON UPDATE NO ACTION,
visualizado_ibfk_3	imagen	`claveI`	`claveI`	ON DELETE NO ACTION ON UPDATE NO ACTION

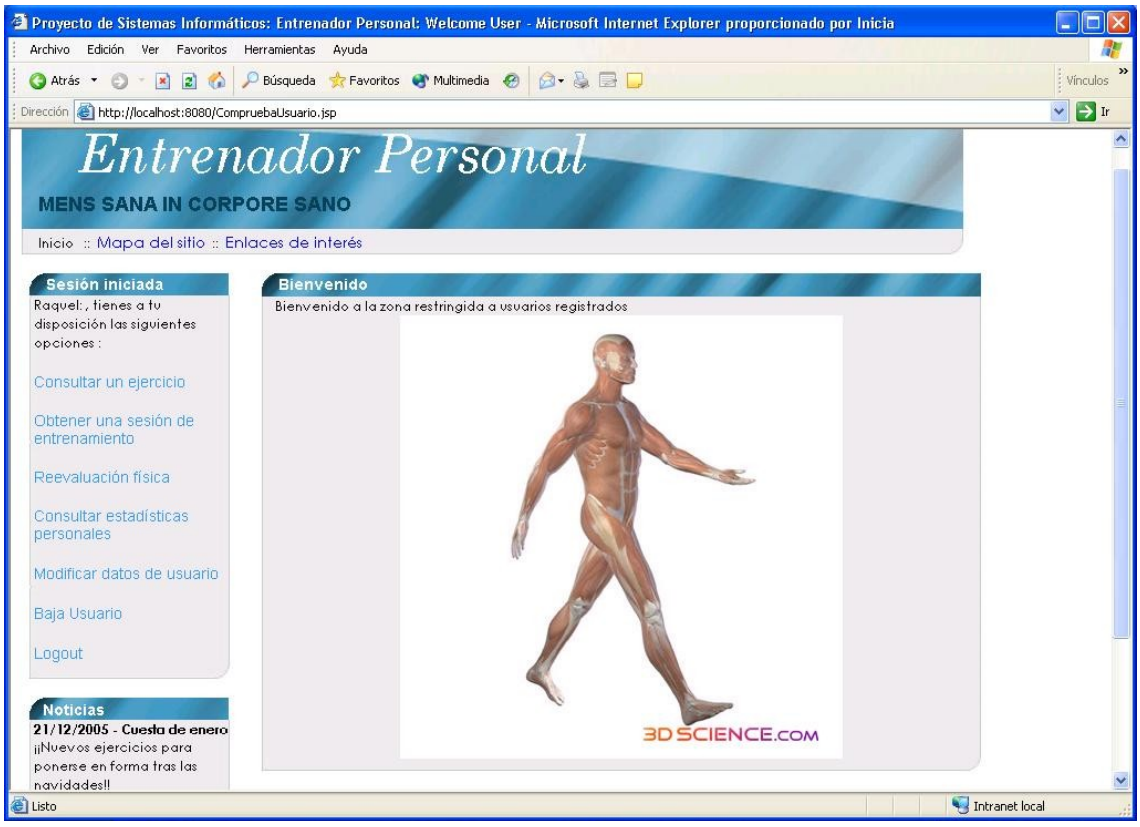
1.16.MÓDULO USUARIO

El usuario puede interactuar con el sistema a través de los JSPs. Hemos dividido los JSPs dependiendo de si el usuario está o no registrado. Si el usuario está registrado tendrá acceso a todas las prestaciones que le proporciona la herramienta como puede ser: mostrarle un ejercicio, una sesión de entrenamiento etcétera .Si por el contrario no está registrado, tenemos unas páginas que podrá ver sin necesidad de realizar tal acción. El primer JSP que el usuario va a encontrar cuando inicie el programa va a ser siempre la página de entrada general (llamada "*index"*). En dicho JSP el usuario tiene varias opciones que a continuación pasamos a detallar, el aspecto visual que tiene dicho JSP:



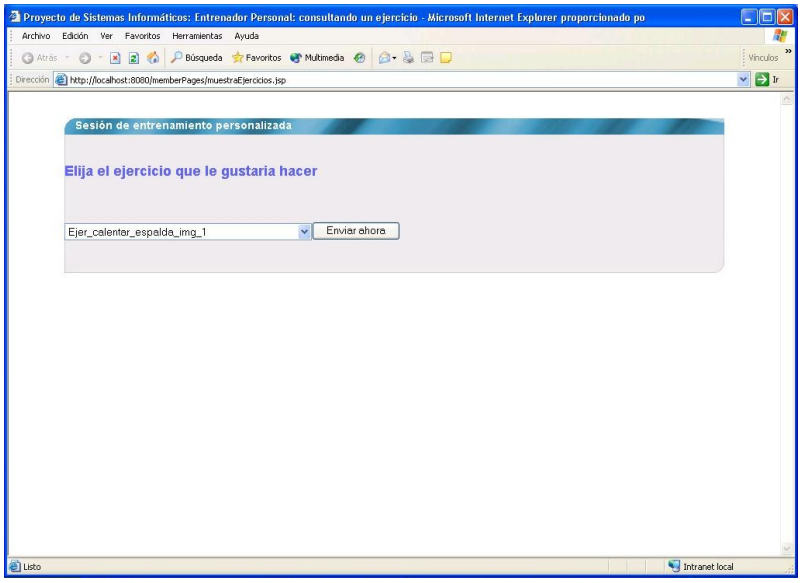
Como puede verse el usuario tiene varias opciones dependiendo si está o no dado de alta.

Si está dado de alta, entrará directamente a su sesión donde podrá consultar un ejercicio, una sesión de entrenamiento, reevaluar su condición física, consultar sus estadísticas personales, modificar sus datos, darse de baja etcétera.

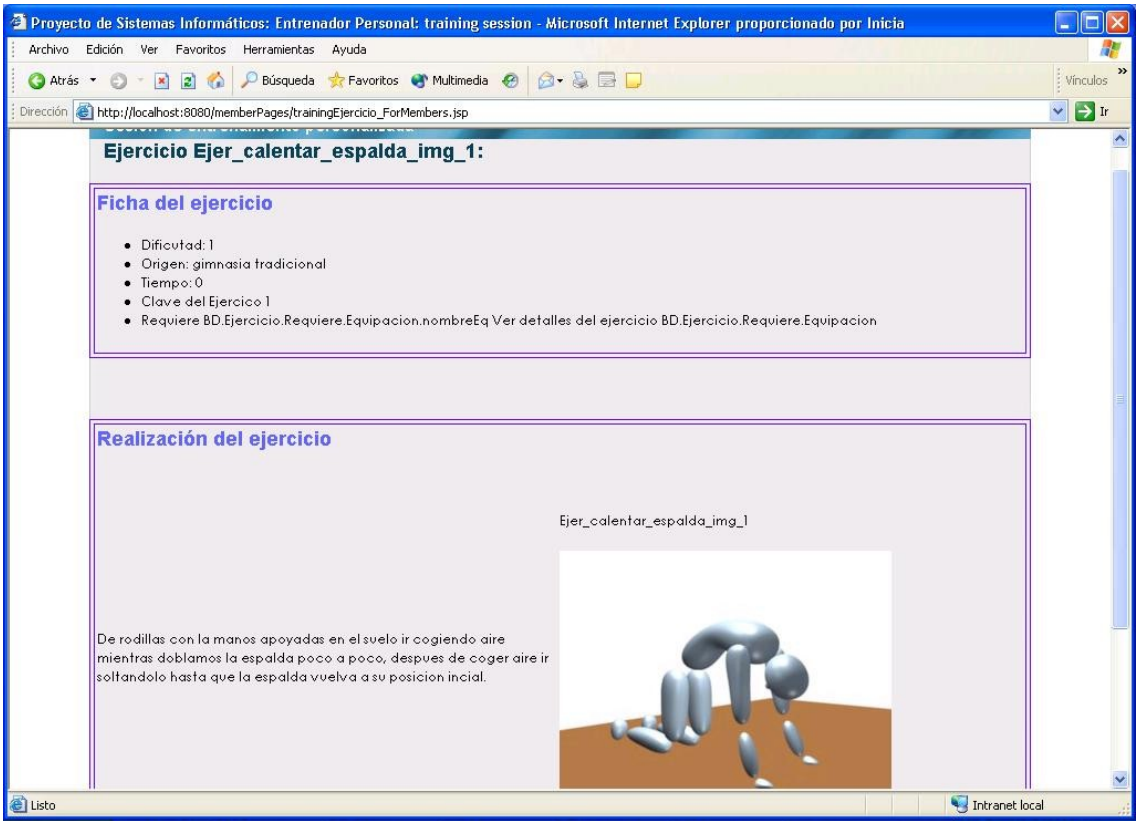


Si por el contrario el usuario no está dado de alta y desea darse de alta lo que hacemos es pedirle que rellene un formulario para conocer sus datos personales así como conocer su perfil físico porque a parte de ponerle la sesión de entrenamiento que se adapte a su condición física necesitamos saber la evolución que ha ido adquiriendo utilizando nuestro sistema de entrenamiento.

Si el usuario desea consultar un ejercicio puede elegir cualquier ejercicio disponible en nuestra base de datos:



Una vez que ha elegido el ejercicio el sistema le muestra la ficha de ese ejercicio:



Si el usuario quiere ver un entrenamiento completo, lo que visualizaría sería una lista de ejercicios.

1.17.SISTEMA DE RAZONAMIENTO BASADO EN CASOS

Como ya introdujimos en secciones anteriores al igual que todo sistema CBR en general, el nuestro se descompone en un preciclo, un ciclo CBR y un posciclo que a su vez se subdivide en conjuntos de tareas tal y como sigue a continuación:

- Preciclo:
  - Tarea para obtener los casos
- Ciclo CBR:
  - Tarea para obtener la consulta de casos
  - Tarea para seleccionar el caso
    - Tarea para seleccionar los casos que funcionan
    - Tarea para computar la similitud
    - Tarea para seleccionar el caso mejor
  - Tarea para la reutilización del caso
  - Tarea para la revisión del caso
  - Tarea de aprendizaje

- Postciclo

El preciclo consta básicamente de la configuración del sistema para conectarlo con la base de datos de donde lee las características del caso mediante un mapeo. El ciclo consta de:

Primero, una tarea para obtener la consulta de casos. Esta tarea básicamente lee la estructura del caso mediante el archivo .xml que define la estructura del caso.

Segundo, pasamos a las tareas para seleccionar el caso más similar posible. En este punto seguimos la siguiente secuencia de tareas: primero una tarea para seleccionar los casos que funcionan. Esta tarea lee todos los casos almacenados en la base de casos. Segundo, una tarea para computar la similitud que existe entre la consulta y cada caso almacenado en la base de casos. Y por último, una tarea que selecciona el caso más similar posible.

Tercero, tenemos una tarea de reutilización del caso más similar posible en caso de que el grado de similitud no sea suficiente con el del caso.

Cuarto, una tarea que revisa el caso para proponer una solución que se adapte bien al caso y quinto, otra para aprender esa nueva solución si el resultado es positivo.

Además de estas tareas típicas de cualquier sistema CBR esta sección aborda también la comunicación Proxy entre el sistema CBR y la página Web e ilustra todo lo anterior con un ejemplo completo de salida del sistema.

Comenzaremos describiendo el recuperador de información.

1.17.1.RI (RECUPERADOR DE INFORMACIÓN)

1.17.2.Introducción

El recuperador de información tiene como principal objetivo modelar la información necesaria para describir un caso, su solución y su resultado. De esta forma nos aseguramos que si el conocimiento empleado se aplica de forma lógica y coherente con las especificación del conocimiento por parte del experto tendremos un sistema que sea capaz de hacer las mismas recomendaciones que haría este experto bajo las mismas premisas.

1.17.3.Especificación de requisitos

Los requisititos del recuperador de la información no se corresponden exactamente con los requisitos de entrada al sistema. El motivo es lógico, ya que tenemos una interfaz Web será allí dónde hagamos las comprobaciones necesarias sobre el rango de los datos que introduce el usuario para ver si se adecuan a cómo los hemos definido en el sistema.

Así pues, el sistema recibirá sólo los datos que vaya a evaluar y en una forma adecuada a la que necesita para poder evaluarla. En otras palabras, tenemos flujos de información desde la página Web a la BD y desde allí al sistema CBR. Todos estos flujos de información deben ser compatibles.

Además, el recuperador de información debe satisfacer que en la BD se rellenen todos aquellos campos clave para que se almacene la información que el sistema CBR necesita para su funcionamiento.

1.17.4.Conocimiento utilizado

El conocimiento utilizado se puede analizar desde dos puntos de vista:

- Conocimiento experto necesario para el desarrollo del sistema.
- Conocimiento de usuario para usar el sistema.

Como hemos apuntado con anterioridad, hemos recurrido a la ayuda de un licenciado en INEF (Instituto Nacional de Educación Física) y nos hemos documentado exhaustivamente con libros, revistas y páginas Web del campo de la educación física y el deporte para abordar el primer punto.

Toda esta información nos ha sido útil para diseñar un sistema sencillo y amigable capaz de recomendar un entrenamiento eficaz a las necesidades y objetivos del usuario según sus características físicas reduciendo así el conocimiento necesario de usuario para usar el sistema.

1.17.5.Base de Casos

La base de casos presenta una estructura compuesta de: descripción, solución y resultado. A continuación, procedemos a explicar cada una de estas partes y cómo se subdividen.

Descripción

Los casos tienen las siguientes características:

- Sexo: enumerado del tipo género (masculino, femenino) con el sexo del usuario.
- Edad: entero con la edad del usuario.
- Altura: entero con la altura del usuario.
- Peso: entero con el peso del usuario.
- IMC: Índice de Masa Corporal. Se calcula respecto la altura y el peso del usuario.
- Objetivos: enumerado que define el objetivo que quiere lograr el usuario con el entrenamiento. Este enumerado puede tener uno del siguiente conjunto de valores: perder grasa, mejorar resistencia, aumentar volumen muscular, mejorar tonicidad, mejorar flexibilidad y mejorar condición integral.

- Músculo: texto que indica los músculos que tiene lesionados el usuario.
- Articulación: texto que indica las articulaciones que tiene lesionadas el usuario.
- Hueso: texto que indica los músculos que tiene lesionados el usuario.
- Articulación Localización: enumerado que indica la parte del cuerpo donde se encuentra la lesión. Este enumerado tiene uno de los siguientes valores: cabeza, extremidades superiores, extremidades inferiores y tronco.
- Músculo Localización: enumerado que indica la parte del cuerpo donde se encuentra la lesión. Este enumerado tiene uno de los siguientes valores: cabeza, extremidades superiores, extremidades inferiores y tronco.
- Hueso Localización: enumerado que indica la parte del cuerpo donde se encuentra la lesión. Este enumerado tiene uno de los siguientes valores: cabeza, extremidades superiores, extremidades inferiores y tronco.
- Nombre equipamiento: texto que indica la equipamiento de la que dispone el usuario para realizar el entrenamiento.
- Peso equipamiento: entero que indica el peso de la equipamiento del usuario.
- Pruebas resistencia: entero con el resultado de las pruebas de resistencia.
- Pruebas flexibilidad: entero con el resultado de las pruebas de flexibilidad.
- Pruebas fuerza parte inferior: entero con el resultado de las pruebas de fuerza de las extremidades inferiores.
- Pruebas fuerza parte superior: entero con el resultado de las pruebas de fuerza de las extremidades superiores.
- Pruebas fuerza parte abdominal: entero con el resultado de las pruebas de fuerza de la zona abdominal.

Nota: según esta descripción del caso contemplamos la posibilidad de que un usuario tenga lesionado más de un hueso, músculo o articulación.

Solución

La solución consta del entrenamiento adecuado a las características físicas de ese usuario. El entrenamiento solución es del tipo de uno de los conceptos que definimos en la ontología.

Resultado

Indica si el entrenamiento solución a las características del usuario tiene resultados positivos o negativos

Casos en nuestra base de casos

Nuestra base de casos inicial del prototipo que presentamos se compone de cinco casos. Todos ellos son inventados, sin embargo debido a la documentación exhaustiva que hemos llevado a cabo podrían corresponderse sin ningún problema con casos muy reales.

Caso Ejemplo

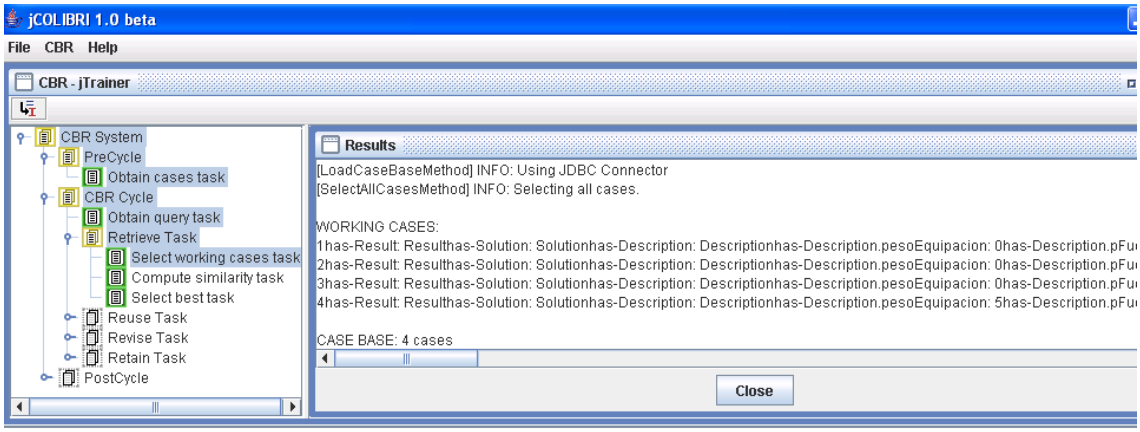
has-Result:
Resulthas-Solution: Entrenamiento_CJ
Solutionhas-Description:
Descriptionhas-Description.pesoEquipacion: 0
has-Description.pFuerzaInferior: 34
has-Description.nombreEquipacion: Cinta
has-Description.sexo: masculino
has-Description.artLocalizacion: null
has-Description.pFuerzaAbdominal: 16
has-Description.altura: 160
has-Description.articulacion: ninguno
has-Description.pFlexibilidad: 24
has-Description.pFuerzaSuperior: 35
has-Description.objetivos: aumentarVolumenMuscular
has-Description.musculo: cuadriceps
has-Description.hueLocalizacion: null
has-Description.pesoPersona: 50
has-Description.musLocalizacion: eInferior
has-Description.pResistencia: 3
has-Description.hueso: ninguno
has-Description.IMC: 23
has-Description.edad: 25

1.17.6.Implementación

Para implementar la base de casos hemos usado el asistente de jCOLIBRI tanto para crear la estructura del caso como para hacer el conector que mapea esta estructura con la base de datos

1.17.7.Ejemplo del funcionamiento de la aplicación

En este punto todavía no tenemos el sistema completo funcionando, sin embargo el sistema nos permite realizar una consulta seleccionando distintos atributos y ver que la consulta se realiza correctamente y que cargamos todos los casos almacenados en la base de casos.



[DataTypePkgReg] INFO: Loading Data Type: Integer
[DataTypePkgReg] INFO: Loading Data Type: Double
[DataTypePkgReg] INFO: Loading Data Type: Boolean
[DataTypePkgReg] INFO: Loading Data Type: String
[DataTypePkgReg] INFO: Loading Data Type: File
[DataTypePkgReg] INFO: Loading Data Type: StringEnumeration
[DataTypePkgReg] INFO: Loading Data Type: HolidayTypesEnum
[DataTypePkgReg] INFO: Loading Data Type: RegionsEnum
[DataTypePkgReg] INFO: Loading Data Type: TransportationsEnum
[DataTypePkgReg] INFO: Loading Data Type: SeassonsEnum
[DataTypePkgReg] INFO: Loading Data Type: Genero
[DataTypePkgReg] INFO: Loading Data Type: Idioma
[DataTypePkgReg] INFO: Loading Data Type: LesionLocalizacion
[DataTypePkgReg] INFO: Loading Data Type: Edad
[DataTypePkgReg] INFO: Loading Data Type: Objetivos
[CBRCore] INFO: Loading package: Core
[CBRCore] INFO: Loading package: User Components

1.17.8.Pruebas

Las pruebas que hemos realizado en esta sección han consistido en asegurarnos que se creara una estructura de datos correcta y que los tipos de datos enumerados se definiesen también de forma apropiada. Por otro lado hemos comprobado que el mapeo de la base de datos se hiciese de forma conveniente y que finalmente se cargaran todos los atributos relativos a la consulta y todos los casos almacenados en la base de casos.

1.17.9.Conclusiones

El sistema recuperador de la información aglutina el conocimiento experto necesario para hacer consultas que satisfacen nuestra especificación de requisitos y las necesidades de implementación del sistema.

1.17.10.SIMILITUD ENTRE CASOS

Para calcular la similitud entre un caso y la consulta del usuario debemos distinguir entre:

- La función de similitud global que en nuestro caso multiplica la similitud local o intermedia de cada atributo por un peso y luego los suma.

- Las funciones de similitud de los atributos. Aquí distinguimos también dos tipos de funciones según los atributos:
    - Los atributos que tienen carácter local, es decir, podemos calcular la similitud entre el atributo de la consulta y su homólogo de forma directa o mediante una función simple.
    - Los atributos que cuya similitud tienen un carácter entre local y global, es decir para poder calcular la similitud entre un atributo de la consulta y su homólogo en el caso de la base de casos necesitamos conocer el valor de otros atributos en la estructura de casos. Sólo de esta forma podemos determinar el grado de similitud entre esos dos atributos.

Las siguientes secciones profundizan y amplían este tema.

1.17.11.SIMILITUD LOCAL

1.17.12.5.3.2.1.1. Introducción

Definimos similitud local como aquella cuyo carácter relaciona sólo un par de atributos homólogos, uno de la consulta del usuario y otro de un caso almacenado en la base de casos.

1.17.13.5.3.2.1.2. Especificación de requisitos

La función de similitud local es diferente según sea el tipo de los atributos y de que forma deban usarse posteriormente:

- Sexo es un enumerado y su función de similitud local viene dada por la igualdad
- Objetivos también es un enumerado, sin embargo su función de similitud local viene dada por una función que implementa el conocimiento obtenido por el experto. Veremos su detalle en la siguiente subsección.
- El atributo IMC es un entero que usa una función de similitud local que también es implementada gracias al conocimiento obtenido por parte del experto. Veremos también su detalle en la siguiente subsección.
- El nombre del equipamiento, el hueso, la articulación y el músculo son textos. Para todos ellos la similitud local se corresponde con la igualdad de textos.
- El Peso del equipamiento es un entero y su función de similitud es local son intervalos.
- La localización de la lesión ya sea en un hueso, articulación o músculo se define mediante la igualdad.

1.17.14.5.3.2.1.3. Conocimiento utilizado

El conocimiento experto usado en esta parte se corresponde principalmente con los atributos Objetivos e IMC.

La similitud entre el atributo Objetivos la calculamos respecto la función que hemos definido para este propósito según la información que obtuvimos del experto.

Los objetivos principales que hemos definido son los siguientes:

- Perder grasa
- Mejorar resistencia
- Aumentar masa muscular (volumen) [y condición física general]
- Mejorar tonicidad [y perder grasa]
- Flexibilidad
- Mejorar condición física general (salud)

La similitud entre estos objetivos la podemos definir así:

- Perder grasa VS Mejorar resistencia: 90%
- Mejorar condición general VS Resto: 90%
- Mejorar volumen VS Tonicidad: 40%
- Otros casos: 0%

La función que define el valor de similitud respecto al IMC equivale a la representación de la tabla de IMC que encontramos en la parte de conocimiento experto.

1.17.15.5.3.2.1.4. Implementación

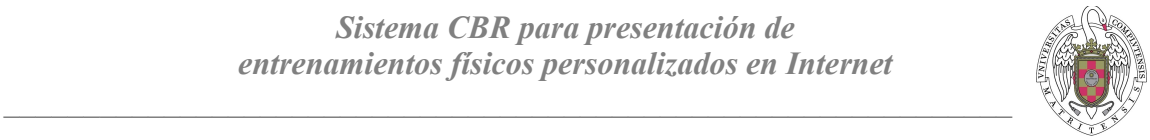
La función de similitud para objetivos implementa el comportamiento descrito previamente, la única diferencia es que usamos el tanto por uno correspondiente al porcentaje. Por ejemplo, 0.9 en vez de 90%

La función de similitud para el IMC implementa una tabla similar a la siguiente:

Clasificación	Nota	Hombres	Mujeres
Baja	0.15	< 14,0	< 30,0
Regular	0.35	14,0 – 24,0	30,0 – 33,0
Media	0.55	24,1 – 35,0	33,1 – 37,0
Buena	0.75	35,1 – 45,0	37,1 – 41,0
Excelente	0.95	> 45,0	> 41,0

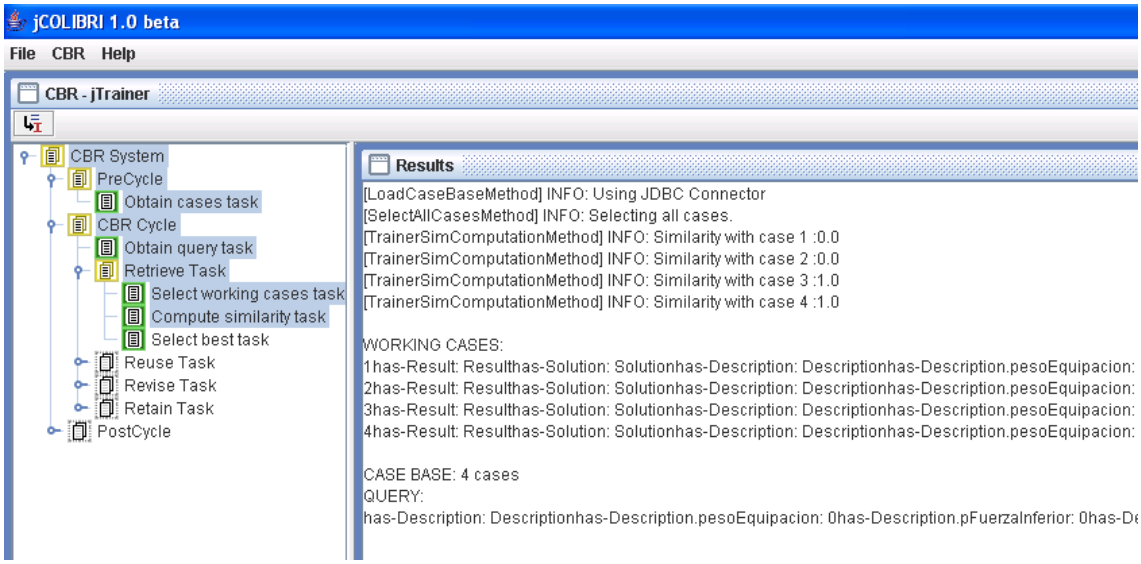
La función de similitud local que usan los atributos de tipo texto es EqualsStringIgnoreCase que ya trae implementada jCOLIBRI.

La función para el equipamiento también estaba ya implementada en la herramienta. Nos referimos concretamente a intervalos de uno.



1.17.16.5.3.2.1.5. Ejemplo del funcionamiento de la aplicación

A continuación mostramos un ejemplo en el que podemos comprobar que efectivamente calculamos de forma correcta la similitud referente sólo al atributo sexo. Consultamos sólo el atributo sexo y pedimos que sea femenino. En nuestra base de casos los casos uno y dos son masculino y los restantes femeninos. Según el comportamiento especificado podemos comprobar que la figura muestra evidentemente un comportamiento correcto.



1.17.17.5.3.2.1.6. Pruebas

Hemos hecho una consulta para cada atributo con función de similitud local comprobando que el resultado era el esperado según el caso con el que estábamos comparando.

1.17.18.5.3.2.1.7. Conclusiones

El cómputo de la similitud local se hace de forma correcta para cada atributo que definimos y satisface las restricciones impuestas por el conocimiento experto así como los requisitos de especificación y las necesidades de implementación del sistema.

1.17.19.

1.17.20.SIMILITUD INTERMEDIA (LOCAL-GLOBAL)

1.17.21.Introducción

La similitud de los restantes atributos de la descripción del caso no se puede realizar de un modo tan simple como vimos en la sección anterior, sino que necesita comparar pares de atributos de la consulta y computar la similitud en base a ese mismo par con cada caso guardado en la estructura de casos. Así el valor de la edad, la fuerza inferior, la flexibilidad y la fuerza abdominal los calculamos junto con el sexo mientras que fuerza superior según la edad del individuo.

Lo ideal hubiera sido describir estos pares de atributos como atributos multivaluados, pero esta funcionalidad todavía no estaba disponible en jCOLIBRI. Sin embargo, esta solución muestra resultados plenamente satisfactorios como veremos en el apartado de pruebas.

1.17.22.Especificación de requisitos

Los requisitos para computar este tipo de similitud son los siguientes:

- La similitud de la edad se debe computar junto con el sexo del individuo.
- La similitud de la fuerza inferior se debe computar junto con el sexo del individuo.
- La similitud de la flexibilidad se debe computar junto con el sexo del individuo.
- La similitud de la fuerza abdominal se debe computar junto con el sexo del individuo.
- La similitud de la fuerza superior se debe computar junto con el sexo del individuo.

1.17.23.Conocimiento utilizado

Al igual que en el caso anterior todo el conocimiento utilizado proviene por parte del experto y la documentación consultada. Las tablas que implementan estas funciones de similitud son similares a las descritas en la parte de conocimiento pero incluimos leves cambios para traducir la nota de una cadena de caracteres a un valor numérico. A continuación mostramos cada una de las tablas con las notas usadas correspondientes.

1.17.24.Implementación

Tablas para el cálculo de la similitud de la edad respecto al sexo.

Intervalo	Hombre
15-20	0.9
20-25	1
25-30	0.9
30-40	0.8
40-45	0.7
45-50	0.6
50-55	0.5
55-60	0.4
60-65	0.3
65-75	0.2
>75	0.1

Intervalo	Mujer
15-18	0.9
18-23	1
23-33	0.9
33-38	0.8
38-43	0.7
43-48	0.6
48-53	0.5
53-58	0.4
58-63	0.3
63-73	0.2
>73	0.1

Tabla para el cálculo de la similitud de la flexibilidad respecto al sexo.

Clasificación	Nota	Hombres	Mujeres
Baja	0.15	< 14,0	< 30,0
Regular	0.35	14,0 – 24,0	30,0 – 33,0
Media	0.55	24,1 – 35,0	33,1 – 37,0
Buena	0.75	35,1 – 45,0	37,1 – 51,0
Excelente	0.95	> 45,0	> 51,0

Tabla para el cálculo de la similitud de la fuerza inferior respecto al sexo.

Puntos	Hombres	Mujeres
1	70 o más	60 o más
0.9	65	56
0.8	58	52
0.7	54	48
0.6	50	44
0.5	46	40

Puntos	Hombres	Mujeres
0.4	42	36
0.3	38	32
0.2	34	28
0.1	30	25
0.0	Menos de 30	Menos de 25

Tabla para el cálculo de similitud de la fuerza superior respecto de la edad.

Fondos completos					
Edad	Valoración				
	Excelente 0.95	Muy bien 0.75	Bien 0.55	Medio 0.35	Bajo 0.15
20 – 29	> 54	45 – 54	35 – 44	20 – 35	< 20
30 – 29	> 44	35 – 44	25 – 34	15 – 24	< 15
40 – 49	> 39	30 – 39	20 – 29	12 -19	< 12
50 – 59	> 34	25 – 34	15 – 24	8- 14	< 8
60 +	> 29	20 - 29	10 - 19	5 – 9	<5

Tabla para el cálculo de la similitud de la fuerza abdominal respecto al sexo.

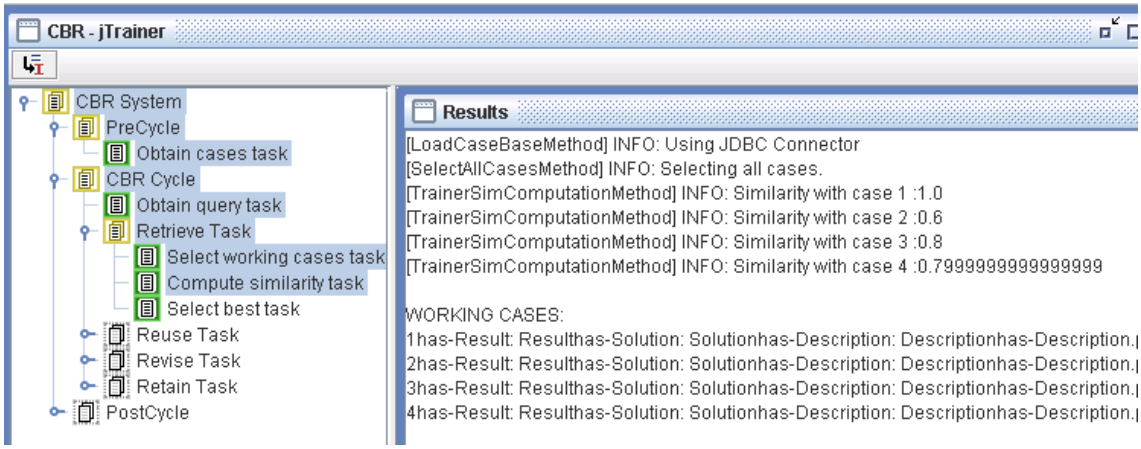
	Valoración				
	Excelente 0.95	Muy bien 0.75	Bien 0.55	Medio 0.35	Bajo 0.15
Hombre	> 30	26 – 30	20 – 25	17 – 19	< 16
Mujer	> 25	21 - 25	15 - 20	9 - 4	< 8

Tabla para el cálculo de la similitud de la resistencia respecto al sexo.

Excelente 0.95	Muy bien 0.75	Bien 0.55	Medio 0.35	Bajo 0.15
< 0	0 - 5	5 - 10	10 - 15	> 15 (21, 22)

1.17.25.Ejemplo del funcionamiento de la aplicación

A continuación mostramos un ejemplo en el que podemos comprobar que efectivamente calculamos de forma correcta la similitud referente a uno de estos pares de atributos. En la consulta pedimos los casos con fuerza superior 35 y edad 25. Estos valores se corresponden exactamente el primer caso de la base de casos.



1.17.26.Pruebas

Hemos hecho una consulta por cada tabla descrita teniendo en cuenta sólo el cálculo de un par de atributos. De este modo, comprobamos que el resultado es el esperado según el caso con el estábamos comparando.

1.17.27.Conclusiones

El cómputo de la similitud intermedia se hace de forma correcta para cada atributo que definimos y satisface las restricciones impuestas por el conocimiento experto así como los requisitos de especificación y las necesidades de implementación del sistema.

1.17.28.SIMILITUD GLOBAL

1.17.29.Introducción

La similitud global es la suma de todas las similitudes locales e intermedias multiplicadas por su peso según la función que describimos en la sección de conocimiento utilizado.

1.17.30.Especificación de requisitos

La función de similitud global debe describir de un modo realista la similitud entre una consulta y un caso de la base de casos dando lugar a un orden de semejanza de casos igual al que haría un experto.

1.17.31.Conocimiento utilizado

El cálculo de la similitud se realiza según la función matemática que describimos a continuación:

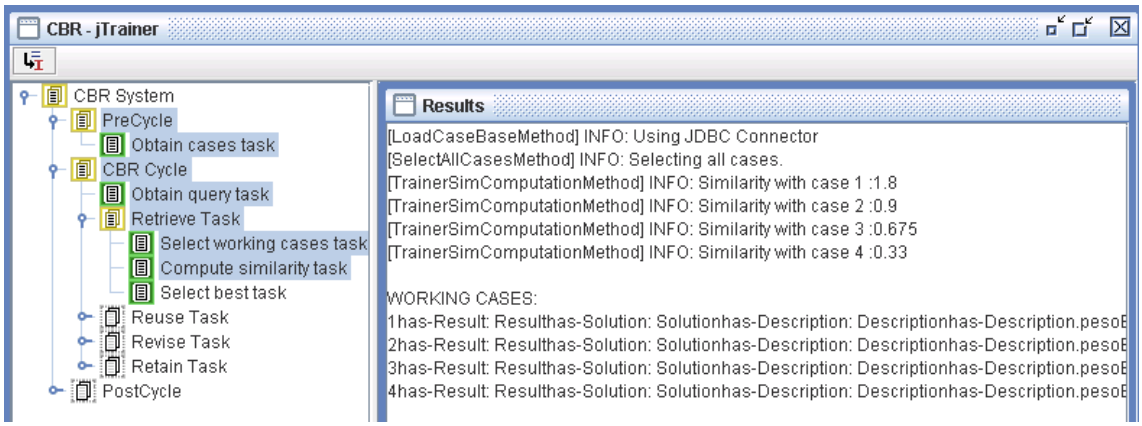
res = 0.08 * (resHuesoLocalizacion + resArticulacionLocalizacion + resMusculoLocalizacion + resHueso + resArticulacion + resMusculo)
+ 0.02 * (resPesoEq + resNombreEq) + 0.5 * (0.4 * resSexo + 0.4 * resSexoEdad + (resSexoFuerzaInferior + resEdadFuerzaSuperior + resSexoFlexibilidad + resResistencia + resSexoFuerzaAbdominal)/ 5 + 0.4 * resIMC)
+ 0.5 * resObjetivos;

1.17.32.Implementación

La implementación es muy sencilla; no es más que la ecuación anterior.

1.17.33.Ejemplo del funcionamiento de la aplicación

A continuación mostramos un ejemplo en el que podemos comprobar que efectivamente calculamos de forma correcta la similitud global. En este ejemplo introducimos exactamente los mismo datos con los cuenta el caso uno en la base de casos.



1.17.34.Pruebas

Hemos hecho una consulta por cada tabla descrita y teniendo en cuenta sólo el cálculo de un par de atributos. De este modo, comprobamos que el resultado es el esperado según el caso con el estábamos comparando.

1.17.35.Conclusiones

La similitud global se hace de forma correcta para el conjunto de todos los atributos y la estructura general del caso satisfaciendo las restricciones impuestas por el conocimiento experto así como los requisitos de especificación y las necesidades de implementación del sistema.

1.17.36.SISTEMA PARA LA OBTENCIÓN DE UN ENTRENAMIENTO ADECUADO A LAS CARACTERÍSTICAS DEL USUARIO

1.17.37.Introducción

Esta sección analiza en profundidad las fases del desarrollo necesario para que el sistema obtenga un entrenamiento adecuado a las características del usuario.

Como hemos visto la solución de un caso es un entrenamiento que es a su vez un tipo concepto de nuestra ontología. Para llegar a este punto trabajar con la segunda versión de jCOLIBRI es un factor clave ya que permite el uso de lógica descriptiva.

A continuación detallamos la especificación de requisitos que implican las distintas fases que ha comprendido el proceso de cálculo de la solución de un caso, el conocimiento utilizado, la implementación de las fases, un ejemplo de funcionamiento del sistema, las pruebas a las que le hemos sometido y finalmente las conclusiones después de todo este ciclo de trabajo. Comenzamos pues con la especificación de requisitos.

1.17.38.Especificación de requisitos

1.17.39.Primera fase: Usuario VS Base de Casos

La primera fase consta únicamente de la devolución directa de la solución del ejercicio que tiene una similitud mayor con el caso del usuario que hace la consulta al sistema. Es interesante darse cuenta que en este punto no distinguimos que el grado de esta similitud sea o no adecuado. Es decir, que se ajuste a un cierto valor o umbral que defina que ese caso es lo suficientemente similar al del usuario para afirmar que la solución es cien por cien adecuada a su situación. Por tanto, esta fase no es más que un paso intermedio en el desarrollo del sistema ya que como hemos avanzado en numerosas ocasiones un requisito fundamental de nuestro sistema es que se adecue a cada usuario de forma personalizada.

1.17.40.Segunda fase: Modificación Caso

La segunda fase se beneficia de la posibilidad de adaptación que ofrece la nueva versión de jCOLIBRI. Concretamente, el objetivo principal al definir la solución como una ontología es facilitar la definición de la reutilización y enriquecer semánticamente los métodos de adaptación.

Para adaptar la solución de un caso es necesario sustituir o eliminar alguno de los ejercicios que componen el entrenamiento. La búsqueda de estos sustitutos lleva a cabo una búsqueda especializada que se aprovecha de la organización de la base de conocimiento de lógica descriptiva y aplica las funciones de similitud relativas a la ontología a los sustitutos para encontrar el sustituto más similar posible. En este punto es interesante tener en cuenta que los sustitutos de los nodos pueden depender de otros nodos de la solución. A continuación realizamos una especificación más en detalle para aclarar estos conceptos.

Como hemos dicho el núcleo principal de un entrenamiento es su parte principal. Los ejercicios de calentamiento y vuelta a la calma deben ser acordes con los ejercicios que se realizan en la parte principal. Así, si la parte principal muscula las extremidades, los ejercicios de calentamiento y vuelta a la calma deben ser también para las extremidades.

Un ejemplo de posible modificación es el siguiente:

- El caso que tiene una similitud más alta con la consulta tiene como solución un entrenamiento para fortalecer las extremidades.
- El usuario que hace la consulta tiene una lesión en el bíceps.
- La modificación necesaria es la siguiente:
  - Sustituir los ejercicios de la parte principal que estén relacionados con el bíceps por otros que no lo impliquen.
  - Sustituir los ejercicios de calentamiento y vuelta a la calma por ejercicios que trabajen los mismos músculos que los nuevos músculos introducidos en la parte principal.

Otro ejemplo de posible modificación es el siguiente:

- El caso que tiene una similitud más alta con la consulta tiene como solución un entrenamiento de intensidad alta. Sin embargo, el grado de similitud entre ese caso y la consulta es menor que el umbral.
- La modificación necesaria sería la siguiente:
  - Sustituir los mismos ejercicios pero con intensidad moderada.

Por último nos podemos encontrar también en una situación en la que sea necesario hacer ambos tipos de modificaciones. Así nos encontraríamos en el siguiente escenario:

- El caso más similar y la consulta tienen un grado de similitud inferior al umbral. El entrenamiento solución es un entrenamiento general de intensidad suave. El usuario que hace la consulta tiene lesionado el radio y sus características físicas

son mejores que las de la descripción del caso.

- Modificaciones:
  - Sustituir los ejercicios de intensidad suave por los mismos ejercicios con intensidad moderada.
  - Sustituir los ejercicios que impliquen el área lesionada por otros lo más similares posibles. Es decir, los ejercicios que se encuentren más cerca en la ontología.
  - Quitar los ejercicios de calentamiento y vuelta a la calma que impliquen el área lesionada.
  - Introducir ejercicios de calentamiento y vuelta a la calma que trabajen los músculos de los nuevos ejercicios incorporados.

Especificación de las modificaciones

1. El caso que tiene similitud más alta está por encima del umbral.
   1.1. El usuario que hace la consulta no tiene ninguna lesión que dificulte la ejecución del entrenamiento. Entonces, no es necesaria ninguna modificación. Devolvemos la solución tal y como indicamos en la primera fase.
   1.2. El usuario que hace la consulta tiene alguna lesión que dificulte la realización del entrenamiento. Entonces:
      1.2.1. Sustituir los ejercicios que impliquen las partes lesionadas por los más similares que no las impliquen según el árbol de la ontología.
      1.2.2. Eliminar los ejercicios de calentamiento y vuelta a la calma que implican partes lesionadas.
      1.2.3. Introducir ejercicios de calentamiento y vuelta a la calma que trabajen los músculos de los nuevos ejercicios introducidos.
2. El caso que tiene similitud más alta está por debajo del umbral.
   2.1. Las características físicas del usuario que hace la consulta son entre un 20% y un 55% mejores que las del caso seleccionado. Entonces:
      2.1.1. Sustituir los ejercicios de la parte principal de la solución por los mismos pero con el siguiente nivel de intensidad.
      2.1.2. Proceder según 1.2 si el usuario de la consulta tiene alguna lesión.
   2.2. Las características físicas del usuario que hace la consulta son más de un 55% mejores que las del caso seleccionado. Entonces:
      2.2.1. Sustituir los ejercicios de la parte principal de la solución por los mismos pero con dos niveles más de intensidad.
      2.2.2. Proceder según 1.2 si el usuario de la consulta tiene alguna lesión.
   2.3. Las características físicas del usuario que hace la consulta son entre un 20% y un 55% peores que las del caso seleccionado. Entonces:
      2.3.1. Sustituir los ejercicios de la parte principal de la solución por los mismos pero con un nivel de intensidad menor.

2.3.2. Proceder según 1.2 si el usuario de la consulta tiene alguna lesión.

2.4. Las características físicas del usuario que hace la consulta son más de un 55% peores que las del caso seleccionado. Entonces:

2.4.1. Sustituir los ejercicios de la parte principal de la solución por los mismos pero con dos niveles de intensidad menos.

2.4.2. Proceder según 1.2 si el usuario de la consulta tiene alguna lesión.

1.17.41. Tercera fase: Aprendizaje - Resultado

La tercera fase consiste en aprender aquellos casos nuevos que aporten información relevante a la base de casos. Las soluciones que se presentan a los usuarios del sistema serán, bien una de las soluciones que presenten estos casos, o una de estas soluciones modificadas. Guardaremos sólo los casos cuyas soluciones hayan sido adaptadas y que tengan resultados positivos. Los resultados son evaluados por el usuario del sistema. Obtenemos esta información vía Web, preguntándole después de cada entrenamiento si el resultado de la sesión le ha parecido satisfactorio.

Por tanto, debemos ser cuidadosos a la hora de realizar el mantenimiento de la base de casos. Principalmente porque tener una base de casos con muchos casos diferentes y similares no aporta mayor riqueza al sistema sino que aumenta su ineficiencia haciendo que el tiempo que se tarda en procesar las consultas sea mayor.

Por este motivo proponemos un límite de 200 casos a la base de casos. Una vez que la base alcance estas dimensiones la librería de casos puede seguir incorporando casos nuevos, pero siempre eliminando previamente aquellos casos de la librería que hayan resultado menos útiles para el funcionamiento del sistema desde que llevan dos meses en la base de casos. De este modo, nos aseguramos que los casos que se han incorporado recientemente permanezcan en el sistema tiempo suficiente para comprobar si realmente mejoran la riqueza de la base de casos.

1.17.42. Conocimiento utilizado

Al igual que en secciones anteriores usamos el conocimiento experto descrito en la sección 4 para especificar, diseñar, implementar y probar estas fases.

1.17.43. Implementación

1.17.44. Primera fase: Usuario vs. Base de casos

La primera fase extiende la implementación desde el cálculo de similitud y la elección del caso más similar posible, añadiendo la parte que muestra la solución del sistema vía Web. Detallamos los detalles relativos a esta implementación en la sección 5.4.

1.17.45.Segunda fase: Modificación del caso

Para satisfacer la especificación de esta fase necesitamos un método que desarrolle varias transformaciones en una instancia que represente la solución de un caso. Para implementar esta funcionalidad de una forma extensible usamos el lenguaje de adaptación de la ontología. Este lenguaje permite crear reglas para adaptar soluciones y almacenar reglas textuales que se puedan cargar, interpretar y generar con el método de adaptación.

Estas reglas se componen de tres partes: la primera es identificar la instancia a adaptar siguiendo la cadena <concept.relation>. La segunda es evaluar que la regla tenga una condición para decidir si la adaptación se debe llevar a cabo. Por último, debemos llevar a cabo la adaptación si corresponde.

A continuación mostramos la sintaxis de estas reglas de acción y su significado:

IDONTO:= /(Conceptoo/Relación)* /Concepto
IDPROPERTY:= (Relación/Concepto)*
IDCASE:= "CASE."(Atributo[.])*
RULE:= IDONTO,@,CONDITION,@,ADAPTATION
CONDITION:= (IDPROPERTY (=|!=) IDCASE) | (IDCASE (=|!=) String)
| [not] (IDPROPERTY instanceOf Concepto)
ADAPTATION:= SUBSTITUTION | MODIFY [FOLLOWDEPENDENCIES Relación]
SUBSTITUTION:= "SUBSTITUTE" [#CONDITION#]
MODIFY:= DIRECTMODIFICATION | ANYOTHERINSTANCEMODIFICATION
DIRECTMODIFICATION:= "DIRECT":IDPROPERTY:instance
ANYOTHERINSTANCEMODIFICATION:= "ANYOTHERINSTANCEOF":IDPROPERTY:concepto
[#CONDITION#]

- **IDONTO** identifica el camino hasta la instancia a adaptar.
- **IDPROPERTY** identifica la propiedad de un concepto.
- **IDCASE** identifica un atributo del caso.
- **SUBSTITUTION** sustituye la instancia por otra elegida al azar. Acepta que se pueda incluir una condición para sustituir la instancia que se debe obedecer.
- **DIRECTMODIFICATION** sustituye un atributo de una instancia con la instancia indicada por el desarrollador.
- **ANYOTHERINSTANCEMODIFICATION** sustituye un atributo de la instancia por otra instancia de un concepto. Acepta una condición para sustituir la instancia.
- **FOLLOWDEPENDENCIES** aplica la regla recursivamente a las instancias relacionadas con la relación especificada.

El método que hemos indicado soporta las siguientes adaptaciones:

- Sustitución de/o por otra instancia especificada por el desarrollador. Además es posible indicar las condiciones que sustituyen la instancia a la que obedecen.

- Sustitución directa de una instancia que se relacione con/o usando una propiedad de la ontología. Esta opción también permite la sustitución directa de una instancia.

Además, estas reglas permiten seguir dependencias (es decir, propiedades) que relacionen instancias para desarrollar el proceso de adaptación recursivamente.

Recordamos que las reglas se componen de tres partes:

- identificación de la instancia para la adaptación,
- condición para evaluar el desarrollo de la adaptación y
- modificación de la instancia.

Estas reglas se almacenan en un archivo de texto que se puede cargar y ejecutar en tiempo de ejecución.

Según estas consideraciones la modificación de sustituir los ejercicios del entrenamiento relativos a la parte del cuerpo que el usuario tiene lesionada por otros de otra parte del cuerpo se podría llevar a cabo con las siguientes reglas:

```
/Entrenamiento/has_principal/Principal/has_exercise/Ejercicios@
parte_cuerpo/eInferior == CASE.Description.musculo@
ANYOTHERINSTANCEOF:has_exercise/Cuerpo:Cuerpo
#has_exercise/Cuerpo not instanceof eInferior#

/Entrenamiento/has_principal/Principal/has_exercise/Ejercicios@
parte_cuerpo/ESuperior == CASE.Description.musculo@
ANYOTHERINSTANCEOF:has_exercise/Cuerpo:Cuerpo
#has_exercise/Cuerpo not instanceof eSuperior#

/Entrenamiento/has_principal/Principal/has_exercise/Ejercicios@
parte_cuerpo/Tronco == CASE.Description.musculo@
ANYOTHERINSTANCEOF:has_exercise/Cuerpo:Cuerpo
#has_exercise/Cuerpo not instanceof Tronco#

/Entrenamiento/has_principal/Principal/has_exercise/Ejercicios@
parte_cuerpo/Cabeza == CASE.Description.musculo@
ANYOTHERINSTANCEOF:has_exercise/Cuerpo:Cuerpo
#has_exercise/Cuerpo not instanceof Cabeza#
```

Debido a que la última versión de jCOLIBRI se nos entregó con muy poca antelación a la entrega del proyecto y a que detectamos pequeños errores en la implementación de la herramienta que debían ser resueltos para que el sistema funcionase, nos ha resultado imposible probar las reglas de modificación anteriores y crear otras nuevas. Por este motivo, tampoco tenemos implementación de la tercera fase. Todo lo que ha quedado pendiente se realizará más adelante.

1.18.COMUNICACIÓN PROXY: CBR – MÓDULO USUARIO [66]

Esta sección explica cómo conectar el framework jCOLIBRI con el servidor Web TOMCAT para poder utilizar GUIs Web.

Normalmente cuando se quiere unir una aplicación existente en TOMCAT se empaqueta como un .jar y se importa/utiliza desde la aplicación Web. Sin embargo esta opción tiene algunos problemas: jCOLIBRI no se puede empaquetar en un .jar (al menos la versión 1) y además el incluir el .jar a la aplicación Web significa que el flujo de control se centralizará en los JSPs.

Por lo tanto debe ser jCOLIBRI el que pueda invocar a los JSPs y que TOMCAT sea simplemente una aplicación llamada desde nuestro framework, no al revés.

Para interconectar TOMCAT con jCOLIBRI (o cualquier otra aplicación java) podemos utilizar varias opciones: RMI, JMS, sockets,… El problema de estas tecnologías es que requieren cierto proceso de aprendizaje. Para evitarlo vamos a unir TOMCAT y jCOLIBRI en la misma aplicación java. Es decir, ambas se ejecutarán en la misma máquina virtual, por lo que podrán comunicarse directamente invocando a los métodos de los objetos.

Esto funciona, entre otras cosas porque TOMCAT cuando se arranca crea su propia hebra. Por lo tanto, jCOLIBRI puede iniciarlo y seguir su funcionamiento normalmente.

En resumen, jCOLIBRI y TOMCAT se ejecutan en la misma MV (en hebras distintas) y se comunican directamente. La idea es que jCOLIBRI arranque TOMCAT en el preciclo dentro de la función precycle() del código generado. TOMCAT presentará los JSPs necesarios para introducir la query y llamará al método cycle() con la consulta a utilizar. Esta parte requiere modificar levemente ese método. Una vez computada la solución, este mismo método cycle devolverá la información a presentar al usuario. El JSP que invocó a cycle() presentará la información obtenida.

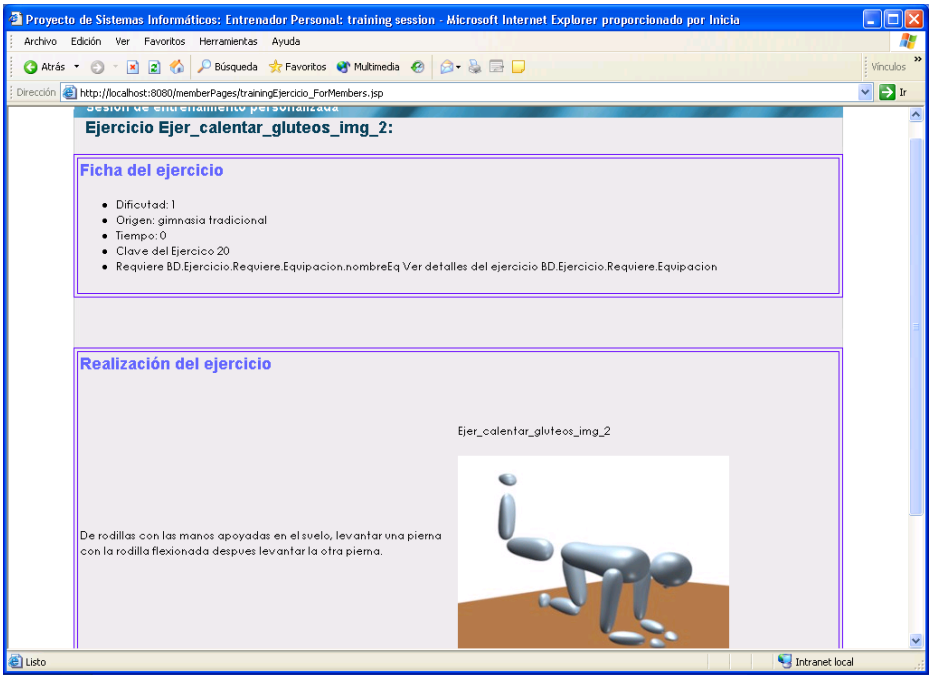
Vamos a hacer una aplicación muy básica que demuestre estas ideas:

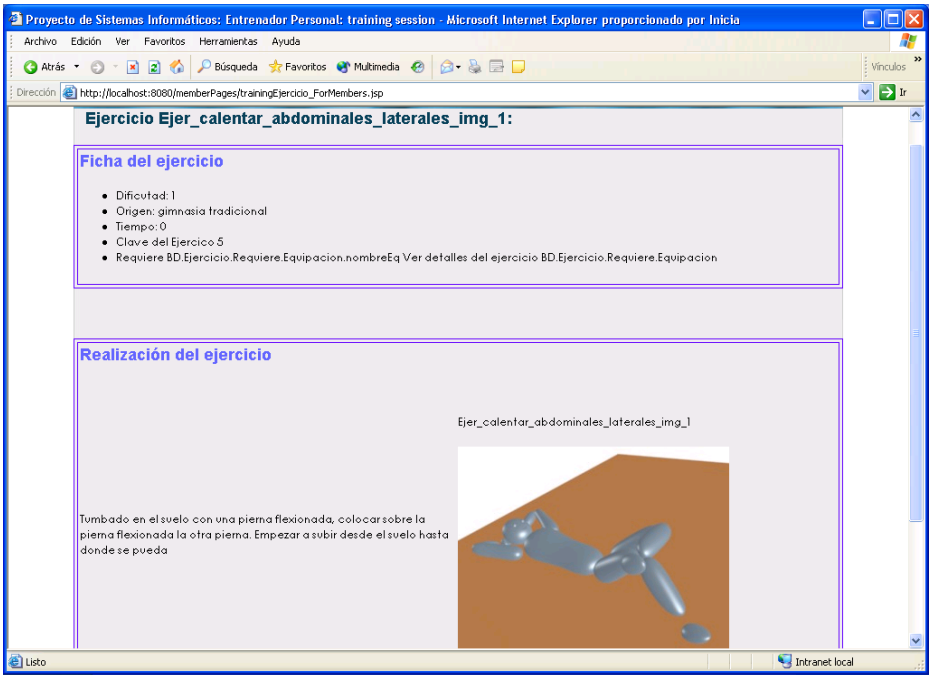
1. Utilizaremos eclipse creando dos proyectos: uno con la clase proxy que une las dos aplicaciones y otro proyecto que en la práctica sería jCOLIBRI.
2. El primer proyecto se llama “jcolibriWebProxy” y contiene sólo una clase jCOLIBRIproxy con métodos estáticos que podrán ser invocados desde los JSP y que llamarán a jCOLIBRI.
3. Luego creamos nuestro segundo proyecto “prueba” que más tarde se sustituirá por jCOLIBRI. En él tenemos el código que arranca y para TOMCAT.
4. En el build path de jcolibriWebProxy, añadimos el proyecto prueba para poder llamar a sus métodos.
5. En la clase jCOLIBRIproxy codificamos los métodos estáticos que necesitemos para manejar las clases de prueba o jCOLIBRI.

6. Luego empaquetamos jCOLIBRIproxy en un jar y lo copiamos al directorio shared/libs de TOMCAT.
7. Modificamos el build path de prueba para incluir ese .jar. Tomcat cargará automáticamente ese .jar al ejecutar los JSPs y como también lo tenemos incluido en el build path de prueba podremos acceder a esa clase desde su código.
8. Creamos los JSPs que llaman a la clase proxy.

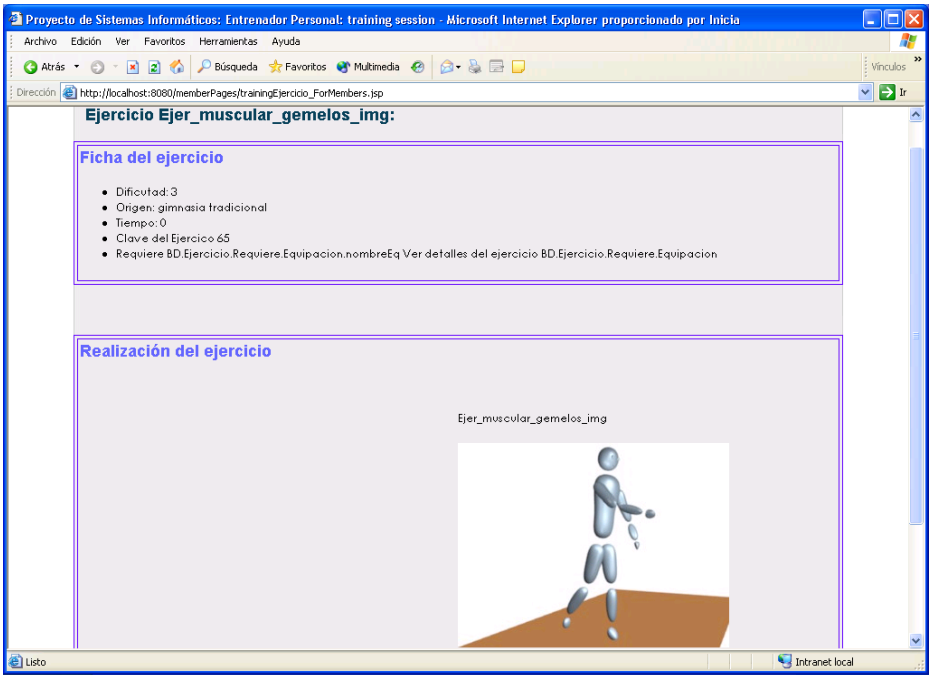
1.19.EJEMPLO DE SALIDA DEL SISTEMA

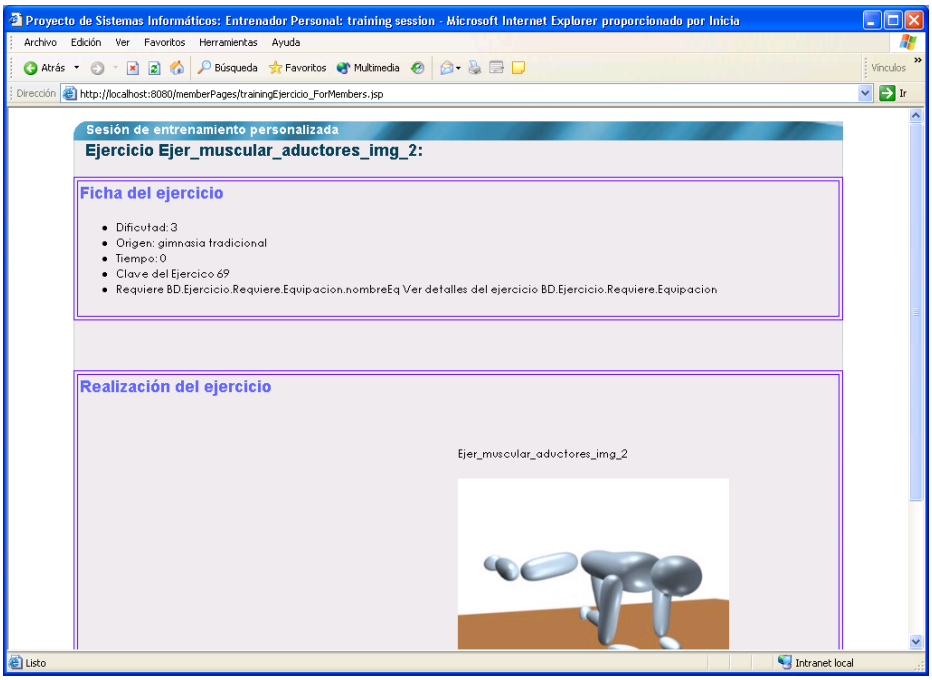
Calentamiento:



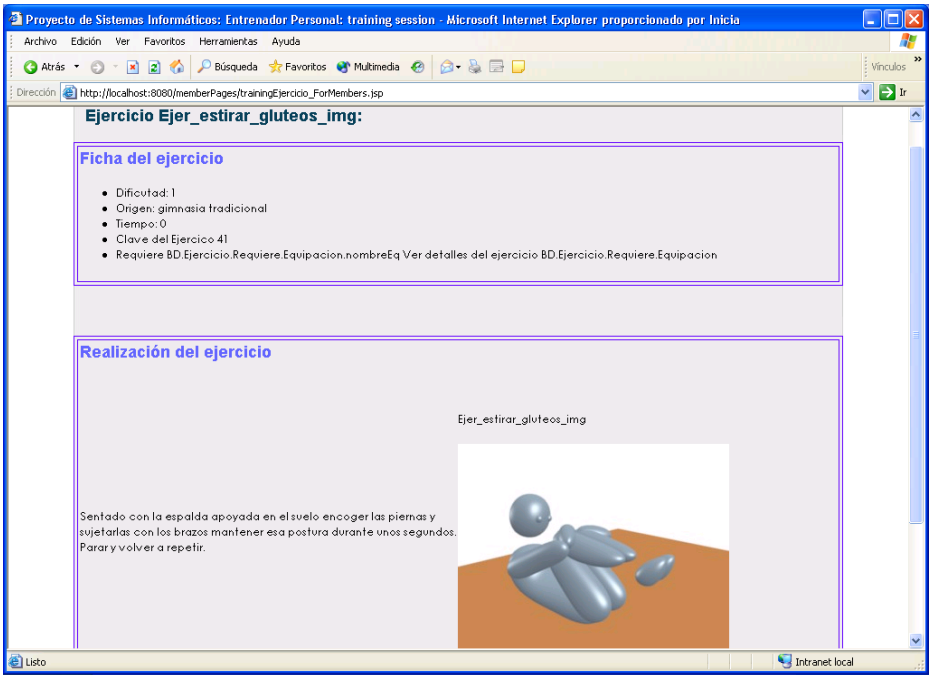


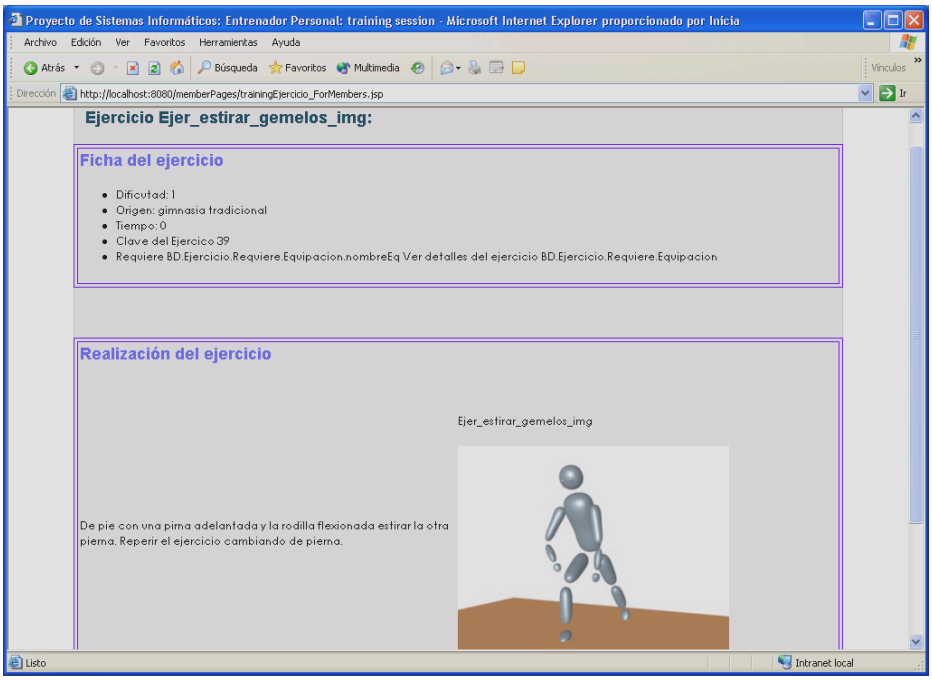
Muscular:

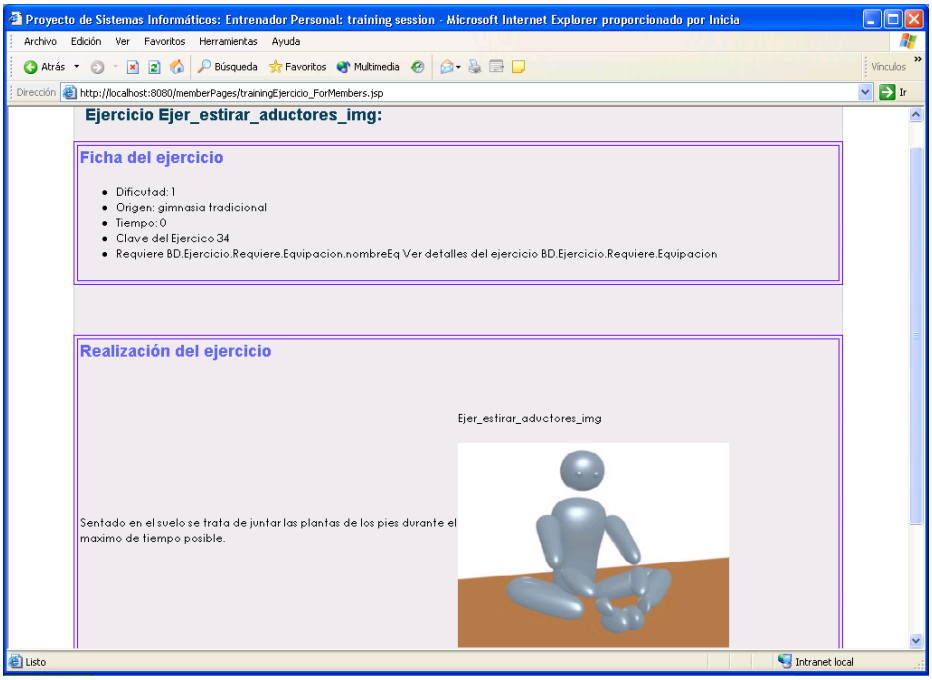

Proyecto de Sistemas Informáticos: Entrenador Personal: training session - Microsoft Internet Explorer proporcionado por Inicia
Archivo Edición Ver Favoritos Herramientas Ayuda
Atrás Búsqueda Favoritos Multimedia
Dirección http://localhost:8080/memberPages/trainingEjercicio_ForMembers.jsp
Ejercicio Ejer_muscular_gemelos_img:
Ficha del ejercicio
Dificutad:3
Origen: gimnasia tradicional
Tiempo:0
Clave del Ejercicio 65
Requiere BD.Ejercicio.Requiere.Equipacion.nombreEq Ver detalles del ejercicio BD.Ejercicio.Requiere.Equipacion
Realización del ejercicio
Ejer_muscular_gemelos_img
Listo
Intranet local

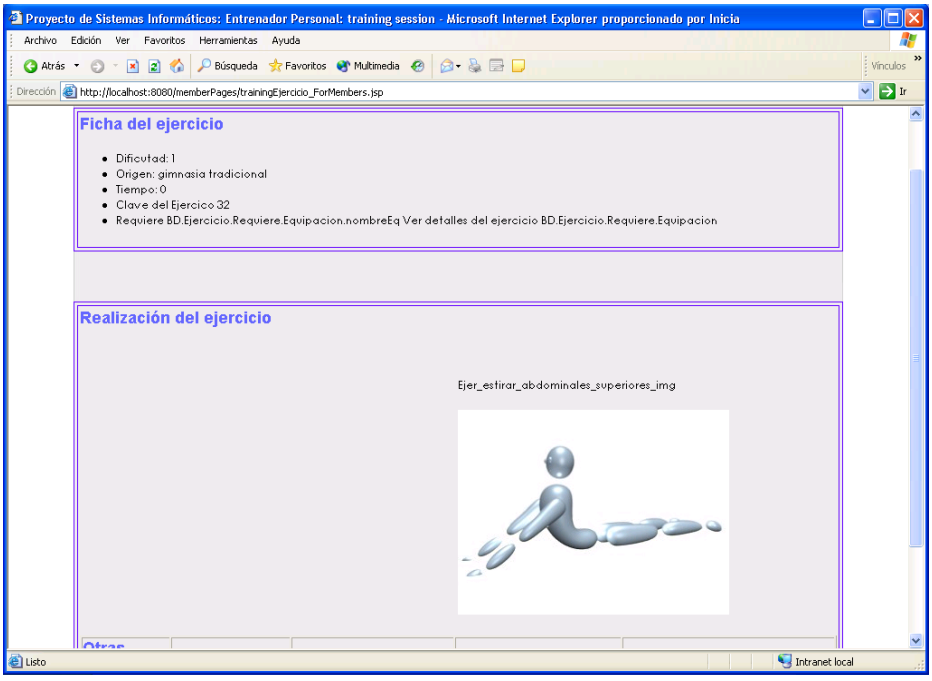

Proyecto de Sistemas Informáticos: Entrenador Personal: training session - Microsoft Internet Explorer proporcionado por Inicia
Archivo Edición Ver Favoritos Herramientas Ayuda
Atrás Búsqueda Favoritos Multimedia
Dirección http://localhost:8080/memberPages/trainingEjercicio_ForMembers.jsp
Sesión de entrenamiento personalizada
Ejercicio Ejer_muscular_aductores_img_2:
Ficha del ejercicio
Dificutad:3
Origen: gimnasia tradicional
Tiempo:0
Clave del Ejercicio 69
Requiere BD.Ejercicio.Requiere.Equipacion.nombreEq Ver detalles del ejercicio BD.Ejercicio.Requiere.Equipacion
Realización del ejercicio
Ejer_muscular_aductores_img_2
Listo
Intranet local

Vuelta a la calma:






Proyecto de Sistemas Informáticos: Entrenador Personal: training session - Microsoft Internet Explorer proporcionado por Inicia
Archivo Edición Ver Favoritos Herramientas Ayuda
Atrás Búsqueda Favoritos Multimedia
Dirección http://localhost:8080/memberPages/trainingEjercicio_ForMembers.jsp
Ejercicio Ejer_estirar_aductores_img:
Ficha del ejercicio
Dificutad: 1
Origen: gimnasia tradicional
Tiempo: 0
Clave del Ejercico 34
Requiere BD.Ejercicio.Requiere.Equipacion.nombreEq Ver detalles del ejercicio BD.Ejercicio.Requiere.Equipacion
Realización del ejercicio
Ejer_estirar_aductores_img
Sentado en el suelo se trata de juntar las plantas de los pies durante el maximo de tiempo posible.
Listo
Intranet local


Proyecto de Sistemas Informáticos: Entrenador Personal: training session - Microsoft Internet Explorer proporcionado por Inicia
Archivo Edición Ver Favoritos Herramientas Ayuda
Atrás Búsqueda Favoritos Multimedia
Dirección http://localhost:8080/memberPages/trainingEjercicio_ForMembers.jsp
Ficha del ejercicio
Dificutad: 1
Origen: gimnasia tradicional
Tiempo: 0
Clave del Ejercico 32
Requiere BD.Ejercicio.Requiere.Equipacion.nombreEq Ver detalles del ejercicio BD.Ejercicio.Requiere.Equipacion
Realización del ejercicio
Ejer_estirar_abdominales_superiores_img
Listo
Intranet local

6. CONCLUSIONES

Al iniciar el proyecto nos planteamos satisfacer la demanda de entrenamientos deportivos personalizados por parte de un sector determinado de la población de forma eficaz, disponible y accesible.

Así, el proyecto en el que nos embarcamos comenzaba implicando un proceso de análisis de las tecnologías para su desarrollo profundizando en sus características y funcionalidades que nos permitía evaluar la oferta actual del mercado y tomar las decisiones que definen el uso de tecnologías a lo largo de todo el proceso de desarrollo.

Por otra parte el dominio sobre el que trabajamos hizo necesaria la adquisición de conocimiento experto y su abstracción de forma que constituyera la lógica del sistema que finalmente daría lugar a los entrenamientos deseados.

Creemos que el sistema destaca principalmente en lo novedoso de la idea, lo vistoso del diseño y los componentes gráficos, la flexibilidad que permite adaptación y el aprendizaje que enriquece la base de conocimiento. La granularidad con la que trabajamos resulta fundamental, puesto que analiza al detalle la similitud entre los casos mejorando notablemente la recomendación de un entrenamiento al usuario concreto.

Por este motivo, creemos que el factor clave de nuestro sistema es la parte de adaptación de la solución y su posterior aprendizaje siempre con una visión clara de adaptación al usuario.

Las dificultades encontradas están directamente relacionadas con la nueva versión de la herramienta jCOLIBRI que no está totalmente depurada. Esperamos que se resuelva muy pronto este asunto para aprovechar al máximo todas las funcionalidades que ofrece.

Debido también a que la lógica central del sistema se basa en la versión Beta de esta herramienta en desarrollo, hemos aprendido a desenvolvernos y a expresar las necesidades de nuestro sistema en términos entendibles por el Grupo de Aplicaciones de Inteligencia Artificial que se encarga de su mantenimiento y desarrollo. Esto ha sido realmente interesante ya que nos ha proporcionado una visión detallada de los flujos de información que están presentes entre los distintos miembros involucrados en el desarrollo de una herramienta. Por otra parte, hemos sido también conscientes del esfuerzo de todas las personas implicadas en su desarrollo y el apoyo que nos han prestado.

Finalmente, diremos que en la medida de nuestras posibilidades esperamos haber aportado una visión práctica como usuarias que ayude a mejorar jCOLIBRI y a probar que satisface sus fines educativos.

7. TRABAJO FUTURO: AMPLIACIONES Y MEJORAS

Como observamos al leer esta memoria nuestro proyecto implica las siguientes partes diferenciadas:

- Interfaz y diseño gráfico de los componentes visuales del sistema.
- Lógica central del sistema formada por la ontología y jCOLIBRI.
- Comunicación entre la interfaz y la lógica del sistema.

Siguiendo esta misma estructura explicaremos cuáles son las posibles ampliaciones y mejoras al sistema que quedan abiertas como trabajo futuro para otros grupos.

Interfaz y diseño de componentes gráficos

Todos los elementos de diseño necesarios para la presentación vía Web de entrenamientos deportivos está completamente diseñados, de modo que las mejoras en este apartado estarían enfocadas principalmente a ampliar el número de imágenes guardadas en la base de datos.

También hemos incluido la posibilidad de asociar videos a los ejercicios, para lo que hemos realizado algún ejemplo de prueba. Estos elementos resultan muy vistosos y entretenidos para los usuarios, de modo que una tarea a realizar en el futuro podría ser ampliar este aspecto, especialmente en lo que se refiere a videos en los que interactúan varios muñecos 3D. Esto podría ser útil para explicar ejercicios que involucren a dos personas o ejemplos de artes marciales como la capoeira, aunque supondría profundizar en las posibilidades que ofrece la herramienta de diseño elegida.

Además, esta herramienta incorpora multitud de opciones de animación, especificación de trayectorias, inclusión de canales de música, etcétera, así como la posibilidad de trabajar con scripts y mucho más. Sería interesante, por tanto, profundizar aun más en las posibilidades que plantea Blender y tratar de averiguar si existen métodos más eficientes para realizar el trabajo así como tantear los límites de la herramienta para producir nuevos elementos gráficos que mejoren nuestro sistema.

Lógica central del sistema: Ontología y jCOLIBRI

Ontología

Para aprovechar al máximo las posibilidades que ofrece el sistema es necesario tener una ontología que cuente con un mayor número de entrenamientos y que éstos sean más exhaustivos. Es decir, que cuenten con un mayor número de ejercicios. De este modo será posible hacer recomendaciones más personalizadas.

jCOLIBRI

Precisamente porque jCOLIBRI no soporta un tamaño de ontología determinado es necesario un trabajo para solventar esta dificultad. Principalmente las dos vías principales que encontramos para resolver este problema son las siguientes: bien buscar la solución en el lado de la herramienta o bien estudiar que características de una ontología se soportan bien con jCOLIBRI y trabajar en base a ellas.

El hecho de tener más entrenamientos nos permitiría una riqueza de casos mucho mayor permitiendo así ampliar la base de casos y mejorar a su vez la personalización.

Del mismo modo el análisis del sistema para introducir nuevas reglas de modificación y su posterior aprendizaje mejoraría la riqueza semántica de la librería de casos.

8. BIBLIOGRAFÍA

[1] Tema 11, transparencias Luis Hernández Yañez. Departamento de Sistemas Informáticos. Facultad de Informática. Universidad Complutense de Madrid. http://www.fdi.ucm.es/profesor/luis/Java/index.html
[2] Blazix
[3] Tomcat. http://wiki.apache.org/tomcat/PoweredBy
[4] http://www.aiai.ed.ac.uk/links/cbr.html
[5] Transparencias Ingeniería de Sistemas Basados en el Conocimientos. Belén Díaz Agudo. Facultad de Informática. Universidad Complutense de Madrid. http://www.fdi.ucm.es/profesor/belend/ISBC/
[6] Grupo de Aplicaciones de Inteligencia Artificial. http://gaia.fdi.ucm.es/grupo/projects/jcolibri/
[7] Ontology based CBR with jCOLIBRI. Juan A. Recio-García, Belén Díaz-Agudo, Pedro González-Calero, Antonio Sánchez-Ruiz-Granados. Dep. Sistemas Informáticos y Programación. Universidad Complutense de Madrid.
[8] OWL Web Ontology Language: http://www.w3.org/TR/owl-features/
[9] DAML+OIL Web Ontology Language: http://www.w3.org/TR/2004/REC-owl-features-20040210/#DAMLReference
[10] Extensible Markup Language XML: http://www.w3.org/XML/
[11] XML Schema: http://www.w3.org/XML/Schema
[12] RDF: http://www.w3.org/TR/2002/WD-rdf-concepts-20021108/
[13] RDF Schema: http://www.w3.org/TR/2002/WD-rdf-schema-20021112/
[14] OWL Guide: http://www.w3.org/TR/owl-guide/
[15] OWL – Guide: OWL Class: http://www.w3.org/TR/owl-guide/#owl_Class
[16] OWL – Guide: subClassOf: http://www.w3.org/TR/2004/REC-owl-features-20040210/#subClassOf
[17] OWL – Guide: Class definition: Thing http://www.w3.org/TR/owl-guide/#DefiningSimpleClasses
[18] OWL – Guide: Simple Classes Nothing. http://www.w3.org/TR/owl-guide/#DefiningSimpleClasses
[19] OWL – Guide: Properties: rdf:Property: http://www.w3.org/TR/owl-guide/#DefiningProperties
[20]: OWL – Guide: Properties: rdfs:subPro http: //www.w3.org/TR/owl-guide/#DefiningPropertiespertyOf
[21]: OWL – Guide: rdfs:domain. http://www.w3.org/TR/owl-guide/#term_domain
[22]: OWL – Guide: rdfs:range.
[23]: OWL – Guide: Terms: Individual: http://www.w3.org/TR/owl-guide/#term_individual
[24]: OWL – Guide: owl:Restriction http://www.w3.org/TR/owl-guide/#PropertyRestrictions
[25]: OWL – Guide: owl:onProperty: http://www.w3.org/TR/owl-guide/#owl_someValuesFrom
[26]: OWL – Guide: allValuesFrom: http://www.w3.org/TR/owl-guide/#owl_allValuesFrom
[27]:OWL Guide – minCardinality: http://www.w3.org/TR/owl-guide/#owl_minCardinality

[28]: OWL – Guide: maxCardinality: http://www.w3.org/TR/owl-guide/#owl_minCardinality
[29]: OWL – Guide: cardinality: http://www.w3.org/TR/owl-guide/#owl_cardinality
[30]: OWL – Guide section on datatypes: http://www.w3.org/TR/owl-guide/#Datatypes1
[31] Section on Annotations in the OWL Reference for details. http://www.w3.org/TR/owl-ref/#Annotations
[32] OWL Semantics and Abstract Syntax http://www.w3.org/TR/owl-semantics/
[33] OWL Guide: oneOf: http://www.w3.org/TR/owl-guide/#owl_oneOf
[34] OWL Guide: hasValue: http://www.w3.org/TR/owl-guide/#owl_hasValue
[35] OWL Guide: disjointWith: http://www.w3.org/TR/owl-guide/#owl_disjointWith
[36] OWL Guide: unionOf, complementOf, intersectionOf http://www.w3.org/TR/owl-guide/#owl_unionOf
[37] OWL Guide: minCardinality, maxCardinality, cardinality http://www.w3.org/TR/owl-guide/#owl_cardinality
[38] Protégé-Frames editor http://protege.stanford.edu/overview/protege-frames.html
[39] Open Knowledge Base Connectivity protocol (OKBC) http://www.ai.sri.com/~okbc/
[40] Protégé-OWL editor http://protege.stanford.edu/overview/protege-owl.html
[41] Protégé User's Guide http://protege.stanford.edu/doc/users_guide/index.html
[42] Protégé-Frames FAQ. http://protege.stanford.edu/doc/faq.html
[43] Documentation section of the Protégé-OWL Web site. http://protege.stanford.edu/plugins/owl/documentation.html
[44] Ontology Development 101 guide to building a frame-based ontology, the Protégé-Frames. http://protege.stanford.edu/publications/ontology_development/ontology101.html
[45] Protégé OWL Tutorial: http://www.co- ode.org/resources/tutorials/ProtegeOWLTutorial.pdf
[46] Protégé-OWL FAQ. http://protege.stanford.edu/plugins/owl/protege-owl-faq.html
[47] Racer: An OWL Reasoning Agent for the SemanticWeb. Volker Haarslev and Ralf Möller. Concordia University, Montreal, Canada (haarslev@cs.concordia.ca). University of Applied Sciences, Wedel, Germany (rmoeller@fh-wedel.de) http://citeseer.ist.psu.edu/cache/papers/cs2/729/http:zSzzSzwww.sts.tu-harburg.dezSzpaperszSz.zSz2003zSzHaMo03d.pdf/haarslev03racer.pdf
[48] Racer : http://www.sts.tu-harburg.de/~r.f.moeller/racer/
[49] Jena: http://jena.sourceforge.net/tutorial/index.html
[50] Jena – Ontology: http://jena.sourceforge.net/ontology/
[51] La página oficial de MySQL: http://www.mysql.com
[52] La página Web SQL YOG. http://www.webyog.com/
[53] Llegamos al portal AUHweb a través de www.freetemplates4all.com, aunque este enlace nos redireccionó a http://www.auhwebonline.com/
[54] La página Web de la empresa Zygote Media Group Inc. http://www.zygote.com
[55] El enlace desde Zygote Media Group Inc que más nos sirvió fue: http://3dscience.com
[56] El software de diseño 3D, Character FX, se puede descargar bajo http://www.insanesoftware.de
[57] Los comentarios de otros internautas sobre software de diseño 3D que consultamos se encuentran bajo la siguiente dirección.

http://www.3dlinks.com/Rating.cfm?linkid=1479
[58] La página oficial de Blender es http://www.blender.org
[59] El foro de diseñadores de Blender se encuentra bajo http://blenderartists.org/forum/

[60] Sport Life es una publicación de Motropress-Ibérica, de carácter divulgativo, muy conocida en el ámbito del deporte. El experto que nos recomendó la revista y con el que realizamos distintas entrevistas es Carlos Victoria, licenciado en INEF por la UPM.
[61] La problemática de la percepción distorsionada del estado físico es uno de los temas explicados en el libro "Musculación. El Entrenamiento Personalizado" de Ramón Lacaba Velasco, Gráficas Sicilia S.L. utilizado en el curso de "Entrenador Nacional de Fisicoculturismo y Musculación".
[62] La revista deportiva a la que nos referimos en este apartado es Sport Life.
[63] Fuente: Apuntes de "Medicina, Deporte y Salud". Asignatura de libre elección de la UCM, impartido por el departamento de medicina deportiva de la facultad de medicina.
[64] Datos tomados de PASSMORE y DURNIN, Physiol, Rews, 1955.
Obtenidos a través de la Catedra de Fisiología de la facultad de medicina de la UCM. Prof. A. Gallego.
[65] Fuente: Apuntes de Nutrición y Dietética de la UAM
[66] Ejecutando Tomcat desde jCOLIBRI. Juan Ant. Recio García. GAIA – Group for Artificial Intelligence Applications. Facultad de Informática. Universidad Complutense de Madrid

Concesión de derechos

Los abajo firmantes autorizan a la Universidad Complutense de Madrid a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la propia memoria como la documentación y/o el prototipo desarrollado.

Mónica González Jenal	Silvia Martín Alejo	Raquel Ramos López

En Madrid, a 7 de Julio del año 2006.