Universitat
Autònoma
de Barcelona

(3520: Implementació d'un modul de generació d'exercicis per un sistema d' e-learning)

Memoria del Proyecto de Fin de Carrera
Ingenieria en Informática
realizado por
Sergio Rodríguez Perez
y dirigido por
Robert Benavente Vidal y *Daniel Ponsa Mussarra*
Bellaterra, 13 de septiembre de 2011

CERTIFICACIÓ DE DIRECCIÓ

El sotasignat, *Robert Benavente Vidal*

Professor/a de l'Escola Tècnica Superior d'Enginyeria de la UAB,

CERTIFICA:

Que el treball a què correspon aquesta memòria ha estat realitzat sota la seva direcció per en *Sergio Rodríguez Pérez*

I per tal que consti firma la present.

Signat:

Bellaterra, 13 de septiembre de 2011

Índice

1. Introducción

Durante las últimas décadas hemos visto la rápida evolución de la tecnología que ha permitido que la gran mayoría de personas tengan acceso tanto a un ordenador personal como a una conexión a internet. Este desarrollo en las tecnologías esta tendiendo una gran influencia en el mundo educativo, ya que forma parte de una nueva herramienta de trabajo que permite a los estudiantes y profesores poderse comunicar "en línea" y colaborar, además de proporcionales la capacidad de acceder a una gran cantidad de información.

A este sistema de enseñanza o aprendizaje que utiliza las tecnologías proporcionadas por Internet se le conoce con el nombre de e-Learning.[1] Gracias a esta tecnología, los profesores pueden gestionar cursos "en línea" sin estar sometidos a una situación geográfica ni horario específico.

En nuestro caso los alumnos se conectan a los cursos mediante un dispositivo con conexión a una red privada (intranet) donde se hospeda el curso. E-Learning presenta una serie de pros y contras que deben tenerse en cuenta [2]. Entre las ventajas que presenta, podemos destacar las siguientes:

- Se facilita la actualización, puesta al día y mantenimiento del curso.
- Permite el abaratamiento de costes para obtener información.
- Facilita al estudiante poder participar cuando lo vea posible.
- Facilita una formación en el momento que se necesita.
- Favorece el aprendizaje mediante elementos multimedia.
- Facilita una formación en grupo y colaborativa.
- Ofrece la posibilidad de establecer un seguimiento de las actividades que ha realizado el alumnado mediante registrando la información en el servidor.
- Ahorra costes en desplazamientos.

Por contra, nos encontramos con algunos inconvenientes a tener en cuenta:

- Depende de una conexión a Internet.

- El profesor necesita invertir más tiempo para atender a sus alumnos que por la forma tradicional.
- Es necesario tener nociones tecnológicas para dar uso a la aplicación.
- Requiere que los estudiantes sepan estudiar de forma autónoma.
- La calidad del aprendizaje se puede deteriorar con el tiempo si no se sigue el curso de forma regular.
- Aun predominan los cursos asistenciales y se desconfía de esta modalidad de enseñanza.

Utilizar el ordenador como herramienta educativa nos proporciona y facilita la tarea de transmitir los conocimientos, aportar ejemplos prácticos y controlar el aprendizaje del alumno.

El proyecto nace con el objetivo de unificar y continuar dos proyectos que fueron realizados anteriormente, dedicados al diseño e implementación de un sistema de e-learning para escuelas de primaria. Este sistema se conoce con el nombre de STAD[3].

El primer proyecto [4] llamado Módulo de Gestión del sistema STAD, tenía como objetivo el diseño de una página web donde poder realizar la gestión de alumnos, profesores y sus vinculaciones con las diferentes asignaturas que se desarrollaran en una escuela. Por otro lado, se inició un segundo proyecto[5], llamado Núcleo del sistema de evaluación de alumnos, se enfoco en plantear un motor generador de ejercicios que mediante un formato establecido generará ejercicios de forma automática y con valores aleatorios sin necesidad de tener almacenada la información en una base de datos (BBDD).

1.1. Estado del Arte

Como hemos visto anteriormente STAD, se circunscribe dentro del ámbito de las aplicaciones e-learning. Entre las herramientas más utilizadas para los sistemas e-learning están los sistemas de Administración de Aprendizaje o LMS. A continuación describimos en que consiste LMS y veremos si STAD tiene sentido, teniendo en cuenta las herramientas que existen en la actualidad.

1.1.1. LMS (*Learning Management System*)

Un LMS es una aplicación software instalada en el servidor, utilizado para administrar, distribuir y controlar las actividades del e-learning de una organización, en nuestra caso, una escuela.

Es el lugar donde los alumnos y profesores se conectan a través de internet para descargar contenidos, ver el contenido de las asignaturas, enviar un correo al profesor, foro de debates, videoconferencia, chats, etc...

Así mismo, LMS consta de un entorno de aprendizaje, al que acceden los alumnos, profesores, administradores y un entorno de administración, en el cual se configuran los cursos, se registran los alumnos, se añaden contenidos, se habilitan servicios, etc.

Pero un LMS por sí solo no genera contenido de los cursos. Para generar el contenido se requiere un generador de contenidos para cursos conocido con las siglas LCMS (Learning Content Management System), que proporciona el medio necesario para crear y reutilizar contenido de e-Learning y reducir esfuerzos de desarrollar duplicados innecesarios.

Muchas empresas se han lanzado a producir distintas aplicaciones LCMS, algunas de pago, otras de libre distribución. Se presentan ahora una selección de los programas más relevantes en el mercado actual.

1.1.2. Sistemas LCMS de pago

Estos sistemas, al ser de pago, carecen de la posibilidad de ser adaptados, modificados o ampliados por un desarrollador externo, lo que limita su funcionalidad a lo que llegue a ofrecer el sistema. Los sistemas LCMS que más destacan en este terreno son *Blackboard Learning System*, *WebCT (aunque ahora está fusionado con Blackboard)* y *Desire2Learn.*

- Blackboard Learning System[6] es un entorno de manejo de cursos. Actualmente esta plataforma está siendo usada a nivel mundial por diversas instituciones relacionadas con la educación. Su principal propósito es añadir elementos online emulando cursos tradicionales, y desarrollar cursos completamente virtuales con pocas o ninguna reunión cara a cara.

- Los productos de Desire2Learn[7] incluyen una plataforma basada en páginas de internet que combina Sistemas de Manejo de Aprendizaje, Sistemas de Administración de Contenido, un Depósito de Objetos (Base de Datos) de Aprendizaje y otras herramientas para la educación en línea.

1.1.3. Sistemas LCMS Open-Source

A diferencia de los LCMS de pago mostrados anteriormente, estos sistemas LCMS gratuitos opensource ofrecen, entre otras cosas, el código fuente del producto liberado para permitir la modificación y actualización por parte de un desarrollador externo, y se ofrecen de forma gratuita, dejando al cliente el único requerimiento de poseer un servidor adaptado para poder alojar el sistema.

Entre los disponibles, los que más destacan en el mercado actual son *Moodle*, *Sakai Project* y *Claroline* .

- Moodle (Modular Object-Oriented Dynamic Learning Enviroment) [8] es un LCMS cuya meta principal es ayudar a los instructores para crear cursos online con muchas oportunidades para una interacción enriquecedora. Está bajo licencia GNU GPL[9]. A día de hoy, Moodle es el LCMS más utilizado del mundo, con más de 40.000 sitios registrados que incluyen en total más de 2.500.000 de cursos, entre los cuales se incluye la Universidad Autónoma de Barcelona.

- Sakai Project [10] es otro LCMS bajo la licencia de la comunidad educativa. Es una aplicación muy potente que está integrada por bastantes herramientas. Estas nos dan la oportunidad de realizar tareas sobre los alumnos. La aplicación se divide en "sitios", los cuales son utilizados como asignaturas, por lo que tenemos que un sitio es igual a una asignatura. Y dentro de cada una de éstas se pueden activar las herramientas que cada profesor necesite, o crea que son necesarias. Si entramos en el nivel técnico, la aplicación esta desarrollada sobre Java principalmente.

- Claroline [11] es una plataforma de aprendizaje y trabajo virtual (eLearning y eWorking) de código abierto y software libre que permite a los formadores construir eficaces cursos online y gestionar las actividades de aprendizaje y colaboración en la web.

La decisión de crear el sistema STAD, ya venía dada al comienzo del proyecto porque los dos proyectos anteriores a este, tomaran en su día la decisión de que los LCMS que existen en el mercado no cumplían con las necesidades que STAD quiere alcanzar.

Después de haber hecho un nuevo análisis para comprobar la posibilidad de que existan nuevas alternativas, vemos que actualmente tampoco existe ningún sistema que se ajuste a lo que queremos alcanzar en este proyecto. En el apartado 1.4 explicaremos el motivo que hace que STAD sea diferente a los sistemas actuales.

1.2. TOAM y STAD

El sistema STAD esta basado en una antigua aplicación didáctica que fue desarrollada durante los años 80, cuyo nombre era TOAM [12] (Computer-Assisted Testing and Practice) . Este sistema fue desarrollado por el Centro de Educación Tecnológica de Israel (CET), organización que tiene como objetivo mejorar el nivel educativo en Israel mediante el uso adecuado de la tecnología. Este sistema se implanto adaptando los programas aritméticos con la intención de resolver los problemas de aritmética para aquellos alumnos que tenían carencias en el aprendizaje.

Este sistema estaba pensado para que los alumnos pudieran practicar ejercicios acordes a su nivel de manera que pudiesen evolucionar a su ritmo. Los alumnos eran evaluados en una sesión de tiempo determinado, los resultados obtenidos por cada alumno eran almacenados en el sistema y evaluados para poder determinar su nivel.

Los ejercicios que se le presentaban al alumno dependían del nivel alcanzado, con los resultados que se iban obteniendo. La información obtenida de las sesiones de ejercicios era reportada a los profesores mediante informes que mostraban los puntos fuertes y débiles de los alumnos, evitando que el profesor tuvieran que realizar el esfuerzo de pensar ejercicios adaptados al alumno, así como corregirlos.

TOAM permitía registrar todos los alumnos existentes en la escuela, así como todos los resultados prácticos realizados durante los cursos, aconteciendo como una herramienta adicional de soporte para el centro. Llevándose a cabo todos los temarios que contenía el sistema, se garantizaba en parte que el alumno había adquirido todos los conocimientos aritméticos que se debe adquirir durante los primeros años de formación.

El primer sistema TOAM fue instalado en Israel en Octubre de 1977. Los resultados registrados en los seis sistemas instalados durante los cursos 1977 y 1980 fueron muy satisfactorios ya que la media de progreso de los estudiantes israelís en la asignatura de Matemáticas se doblo desde la introducción del sistema.

A partir de estos resultados, fue instalado en 120 escuelas mas repartidas por Israel, con 70.000 estudiantes utilizando en las asignaturas de Matemáticas, Comprensión de texto, ingles, Mecanografía y Informática. Además se desarrollaron versiones en ingles que fueron utilizadas en Sudáfrica, España y Latinoamérica.

La escuela Pere Viver de Terrassa fue seleccionada por la Consejería de Educación de la Generalitat como uno de los 10 centros de Catalunya donde implantar una experiencia durante el curso 1984-85 de enseñanza asistida por ordenador mediante el sistema TOAM. Esta experiencia se convirtió en un soporte básico a la dinamización de la enseñanza de las matemáticas y en la introducción de nuevas tecnologías en los centros educativos.

Actualmente el sistema TOAM se encuentra en condiciones muy precarias ya que es un sistema obsoleto y sin mantenimiento, por ese motivo ha surgido la idea de construir una replica actualizada de este sistema aprovechando la tecnología actual para mejorar los inconvenientes que presentaba el anterior sistema. Los cuales los explicamos a continuación:

- Terminales específicos: TOAM es complejo porque utiliza terminales diseñados exclusivamente para este uso, como un ordenador central con un sistema operativo también exclusivo que hace que su sistema de mantenimiento dependa de un especialista. La existencia de nuevas tecnologías permite la utilización de PC's que abaratan el mantenimiento del sistema.
- Sistema Cerrado: El sistema no es flexible debido a que contiene un conjunto finito de posibles ejercicios por materia y no existe la posibilidad de incorporar nuevos ejercicios. Uno de los objetivos de este proyecto es poder diseñar un módulo que permita de manera sencilla la creación de nuevos ejercicios siendo así un sistema abierto e infinito número de ejercicios realizando variaciones a partir de un mismo ejercicio.
- Apariencia: la interficie grafica en la cual interactúan los alumnos y los profesores es poco atractiva y poco intuitiva para los alumnos de primaria. Actualmente existen muchas posibilidades para desarrollar interficies atractivas que propicien una mayor atención y motivación del alumnado. Así un buen diseño de pantallas puede hacer mas fácil y intuitivo el sistema.

- Gestión sistema: Las operaciones administrativas se realizan mediante comandos por lo que se convierte en una tarea laboriosa y poco intuitiva. Por ese motivo se plantea realizar un módulo administrativo.

Por los motivos expuestos anteriormente se decidió la creación de un nuevo sistema STAD. Hasta el momento se han realizado dos proyecto, uno focalizado en el módulo administrativo y el segundo proyecto planteo el módulo de ejercicios, sin llegar a desarrollarse ya que esto ha sido implementado en este proyecto.

1.3. Objetivos

En este capitulo enumeraremos los diferentes objetivos que se pretenden alcanzar en este proyecto.

1. Evaluar, rediseñar e implementar la idea que se propuso en el segundo proyecto nombrado anteriormente (Núcleo del sistema de evaluación de alumnos), para darle la capacidad al sistema STAD de poder gestionar y crear nuevos modelos de ejercicios. Así obtenemos un entorno mas abierto y modulable.

2. Para gestionar los ejercicios se han planteado dos conceptos necesarios para que exista una alta capacidad de personalización de los ejercicios. Los conceptos son:

   - Ejercicios tipo: se trata de una definición general en la cual se definen los parámetros que van a ser necesarios, para llamar a una función generadora. La función generadora es el código fuente que es donde se realizan los cálculos para generar los ejercicios.
   - Ejercicios concretos: Es la definición de los limites de la función generadora para poder crear un ejercicio personalizado a nuestras necesidades.

   Teniendo en cuenta esta idea el sistema adquiere la capacidad de generar un número ilimitado de ejercicios basándose en un modelo diseñado previamente. Este concepto hace de STAD un sistema único respecto a los aplicaciones e-learning que existen en la actualidad. Más adelante en el diseño veremos en que consiste esta idea.

3. Implementación de un nuevo módulo de ejercicios. Este módulo es el encargado de la gestión de los ejercicios y pretender ser una herramienta para poder dar de alta ejercicios tipo y ejercicios concretos.

4. Actualización de la tecnología. El módulo de gestión fue implementado durante los años 2003/04 por lo que es necesario realizar una revisión de la aplicación que existe actualmente ya que se han encontrados implementaciones que son obsoletas en la actualidad.

5. Integración. El nuevo módulo de ejercicios tiene que estar integrado con el módulo de gestión existente, teniendo en cuenta la BBDD diseñada por los anteriores proyectos para el módulo STAD. Es de vital importancia este punto, ya que si no existe integración entre ambos módulos no obtendríamos un sistema completo.

6. Núcleo generador y evaluador. Es el encargado de auto-generar en el momento que el alumno inicie una sesión de ejercicios. El núcleo generador se nutrirá de los ejercicios concretos que han sido definidos por los profesores en el módulo de ejercicios, para poder ofrecer de forma aleatoria un gran número de ejercicios al alumno acorde con su nivel académico.

7. Ampliación de ejercicios. Generar una gran cantidad de ejercicios tipo para ofrecer un abanico de posibilidades al profesor y poder ofrecer así un sistema provechoso para los estudiantes.

8. Control de la LOPD(Ley Orgánica de protección de datos). Posibilidad de configuración de la aplicación para poderlo cumplir.

9. Modularizar el sistema para posibles migraciones a B-learning (enseñanza mediante dispositivos móviles.)

Por último una de las premisas que han de seguir el proyecto son: intentar modificar lo mínimo posible la BBDD definida en los anteriores proyectos, así como mantener el estilo actual de la página web del módulo de gestión. En caso de realizarse modificaciones en la BBDD habría que revisar el módulo ya existente para adaptarlo a las nuevas necesidades y permitir a la aplicación la posibilidad de poder ser traducida a diferentes idiomas de manera rápida y sencilla.

1.4. Descripción de la solución

Hemos hablado sobre diseñar un nuevo módulo de ejercicios, pero no hemos dicho que debe hacer esta módulo. A continuación pasamos a describir a grandes rasgos que es lo que tiene que hacer la solución que proponemos.

1. Control de acceso de los usuarios teniendo en cuenta los perfiles que van a ser definidos en la aplicación.
2. Módulo de gestión de ejercicios, integrado en la aplicación STAD.
3. Motor evaluar y lanzador de ejercicios encargado de generar ejercicios e interactuar con toda la información existente en la BBDD.
4. Control estadístico sobre ejercicios realizados por los alumnos y calculo del nivel del alumno.
5. Integración del módulo de gestión que ya esta en funcionamiento actualmente, con el nuevo módulo de ejercicios.

1.5. Planificación del proyecto

El proyecto fue iniciado aproximadamente el 13 marzo de 2011 y se prevé finalizarlo el 13 de septiembre de 2011. Durante este periodo se ha estimado una dedicación semanal de 20h, organizadas de manera flexible.

El proyecto se desarrollará siguiendo las siguientes etapas:

1. Documentación: recabar información de los proyectos que ha sido realizados con anterioridad para tener una visión global.
2. Análisis de la aplicación y requerimientos: Analizar las diferentes módulos de la aplicación y exponer los objetivos que se han marcado en el proyecto.
3. Diseño: Plantear como va a ser diseñada la BBDD, y el módulo de ejercicios.
4. Implementación: Definir los herramientas a utilizar e iniciar el desarrollo de la aplicación.

5. Fase de pruebas: Realizar test en la aplicación para detectar posibles fallos.

6. Mejoras y correcciones: Solventar los problemas encontrados durante las pruebas.

7. Redacción: Redactar memoria del proyecto.

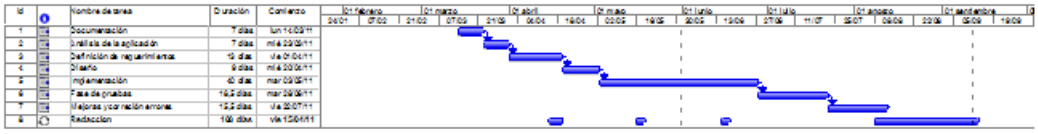


Figura 1: Diagrama de Gantt

2. Análisis y requerimientos

En este apartado se analizan los módulos principales del sistema STAD.

Primero analizaremos el módulo de gestión administrativa del sistema que actualmente ya está en funcionamiento. Posteriormente describiremos y diseñaremos el módulo de gestión de ejercicios, el cual va a ser desarrollado en este proyecto. Finalizaremos el análisis explicando un nuevo concepto que ha sido necesario incluir en desarrollo para poder integrar estos dos módulos que es la gestión de roles.

2.1. Módulo de gestión

Con este módulo el profesor o administrador del sistema tendrá la posibilidad de por un lado, realizar las tareas de gestión y mantenimiento de datos necesarios para el funcionamiento de STAD y por otro lado, generar diferentes tipos de informes y estadísticas sobre el progreso académico de los alumnos.

Las tareas principales que se pueden realizar mediante este módulo son:

- Gestión de escuelas
- Gestión de clases
- Gestión de alumnos
- Gestión de profesores
- Gestión de informes
- Gestión de roles

A continuación pasamos a explicar las funcionalidades y mediante la utilización de los casos de uso determinaremos los requisitos funcionales del sistema.

2.1.1. Gestión de escuelas

Este módulo es el encargado de definir las diferentes escuelas que comparten el sistema. Mediante este módulo se pueden crear y definir ciertos parámetros necesarios para el funcionamiento de la escuela, que son los siguientes:

- Los datos descriptivos referentes a la escuela.
- El número de lecciones a realizar por el alumno para determinar el nivel adquirido por el alumno.
- La duración en segundo de cada ejercicio a resolver
- Los minutos de duración de cada lección.

Algunos de estos parámetros están definidos en niveles inferiores, como puede ser a nivel de asignatura o nivel de alumno, de no estar definidos en estos niveles, se obtendrá la información definida para la escuela. El funcionamiento de la aplicación se inicia accediendo a la selección de la escuela, que dependiendo del rol del usuario permitirá crear y definir toda la información referente a la escuela o únicamente permite modificar el numero de lecciones a realizar y el tiempo de duración de una sesión de ejercicios.

Una vez dada de alta la escuela se puede acceder a los módulos de gestión de clases, gestión de alumnos, gestión de profesores, gestión de listados y gestión de ejercicios. Pasamos a ver el caso de uso para el módulo de escuelas.

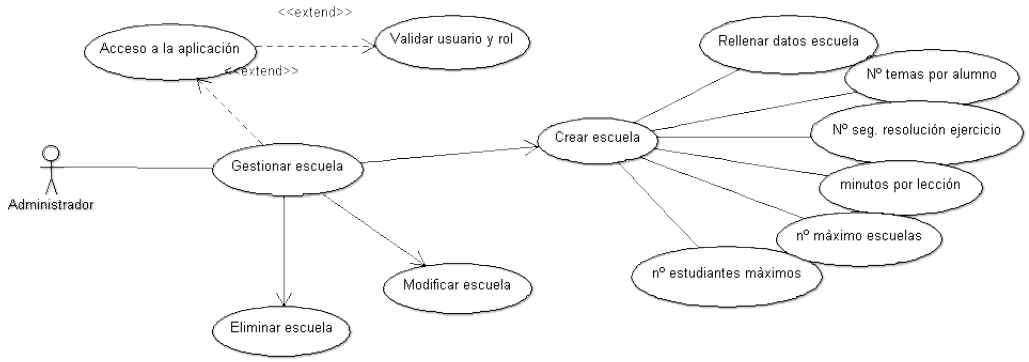


Figura 2: Caso de uso Gestión escuela por parte del administrador

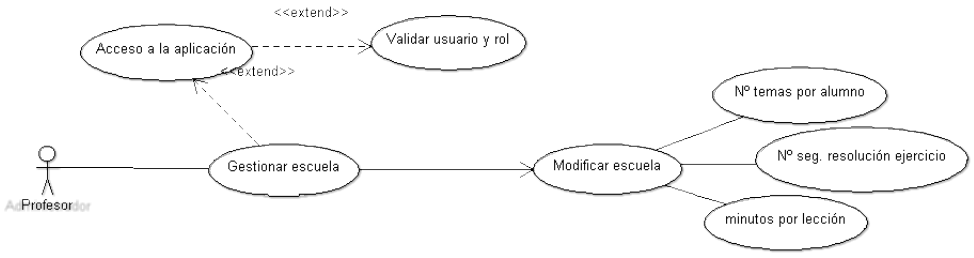


Figura 3: Caso de uso Gestión escuela por parte del profesor

2.1.2. Gestión de clases

Para cada escuela se podrán crear diferentes clases, así como modificarlas o darlas de baja. Con la gestión de clases se podrán asociar los alumnos de la escuela con la clase seleccionada y asignar materias y profesores a ésta.

La gestión de alumnos dentro de una clase permite realizar las siguientes funciones:

- Dar de alta un alumno en la clase. De este modo estará asociado con asignaturas que se imparten en la clase seleccionada.
- Editar los datos de uno o varios alumnos de la clase seleccionada. Se visualiza las asignaturas de cada uno de los alumnos seleccionados y se podrá modificar los valores de sus parámetros.
- Cambiar uno o varios alumnos de clase asignándoles a otro grupo de la propia escuela.
- Añadir alumnos que pertenecen a la escuela y no tienen asociado ningún grupo.
- Desvincular alumnos de la clase seleccionada.

La gestión de asignaturas y profesores permite realizar las siguientes funciones:

- Asignar la asignatura y el profesor a la clase seleccionada y definir o modificar los parámetros de la asignatura que realiza el profesor en la clase. Los parámetros son: Nivel, Número de lecciones, Minutos por lección y Segundos por ejercicio.
- Eliminar las asignaturas que tiene el profesor asignada a una clase.

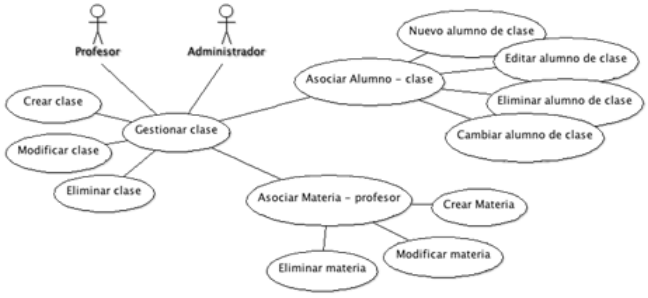


Figura 4: Caso de uso Gestión de clases

2.1.3. Gestión de alumnos

Las funciones que se pueden realizar en la gestión de alumnos son las siguientes:

- Dar de alta los alumnos dentro de la escuela.
- Edición de los datos de uno o varios alumnos de la escuela, donde podremos modificar los datos personales, la clase que pertenecen y los parámetros de las materias que los asocian con las clases. Para todas estas funciones se podrán seleccionar todos los alumnos de la escuela aquellos que no estén asignados a ningún grupo o aquellos que pertenecen a un grupo en concreto.
- Eliminación de uno o varios alumnos de la escuela.

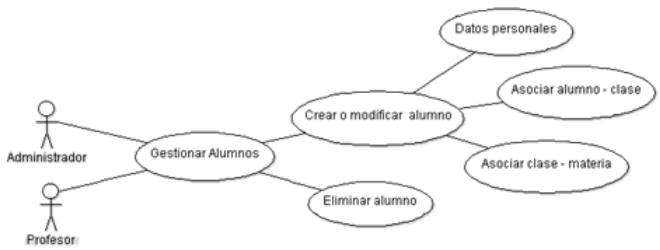


Figura 5: Caso de uso Gestión de alumnos

2.1.4. Gestión de profesores

Las funciones que se pueden realizar en la gestión de profesores son las siguientes:

- Registrar nuevos profesores en la escuela seleccionada.
- Editar los datos de uno o varios profesores de la escuela, donde podremos modificar los datos personales del
- profesor, las clases en las que se puede registrar y las materias en el que está asignado.
- Eliminar los profesores existentes en la escuela.

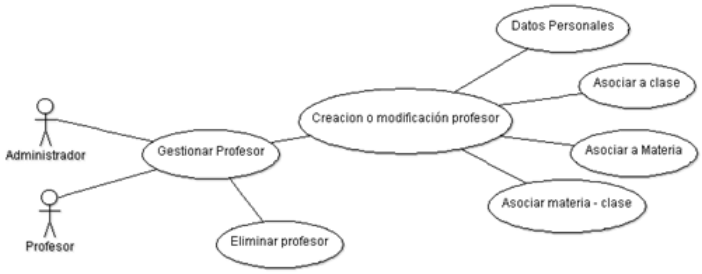


Figura 6: Caso de uso Gestión de profesores

2.1.5. Gestión de informes

La gestión de informes pretende gestionar gráficamente la evolución de los alumnos para cada una de las clases y escuelas existentes, a continuación describimos los informes existentes:

- Informe de progreso de la clase. Muestra por cada alumno de la clase y la materia seleccionada, el total de respuestas correctas respecto al total de ejercicios realizados durante la última lección, así como el nivel de los temas de la asignatura.
- Informe de progreso del alumno. Mostrar la información correspondiente a un alumno seleccionado.
- Informe de progreso de una materia. Detalla el nivel promedio obtenido para una materia determinada de todas las clases que realizan la materia.

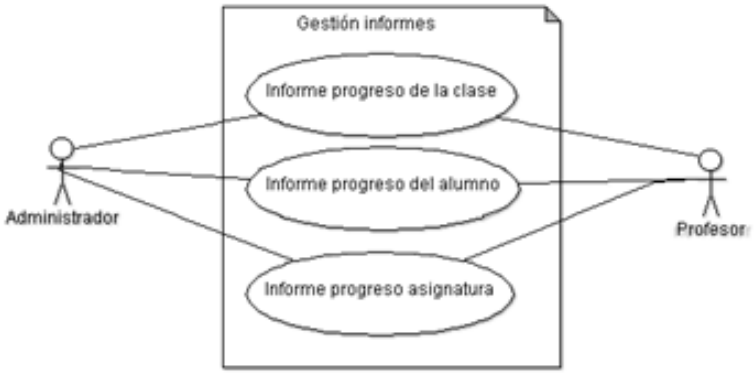


Figura 7: Caso de uso Gestión de informes

2.1.6. Gestión de roles

Éste nuevo módulo ha sido creado por las nuevas necesidades que se requieren en la aplicación ante la creación del nuevo módulo de ejercicios. Incluir roles a los diferentes usuarios es necesario porque hay que controlar el acceso a la aplicación debido a que alguno de los módulos son sensibles y no deben ser visibles i/o accesibles para todos los usuarios, por ese motivo hemos definido los siguientes roles:

- Administrador: Plena capacidad de modificación de toda la aplicación por completo.
- Profesores: Capacidad de gestionar toda la información referente a la escuela, exceptuando la gestión de la propia escuela, debido a que un profesor no debe tener permisos para modificar o eliminar una escuela. También hay que tener el cuenta que los profesores únicamente podrán modificar o eliminar ejercicios creados por ellos mismos. En el apartado 2.2.2 , explicaremos mas detalladamente en que consiste.
- Alumnos: Únicamente podrán acceder a módulo de lanzador − evaluador de ejercicios.

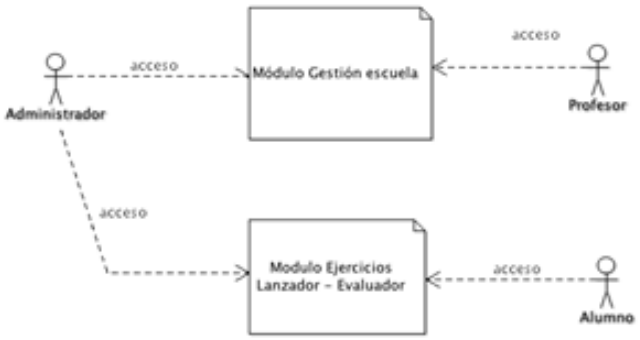


Figura 8: Caso de uso Gestión de roles

2.2. Módulo de ejercicios

En este capítulo describiremos el funcionamiento general del módulo. Este análisis se dividiendo en:

- Generación de ejercicios
- Gestión de ejercicios
- Ejecución de ejercicios.

2.2.1. Módulo generación de ejercicios

El objetivo del nuevo módulo generador de ejercicios es la posibilidad de generar nuevos tipos de ejercicios de manera que el sistema no sea un sistema cerrado, sino que sea capaz de evolucionar con las nuevas necesidades pedagógicas. La idea que hace que este proyecto sea original es la posibilidad de ofrecer al profesorado un conjunto de herramientas para que sean capaces de ampliar la base de datos con nuevos ejercicios concretos a partir de los ejercicios tipos que ya existen definidos o crear ejercicios tipos nuevos con cierta facilidad. Este concepto hace que esta se convierta en una aplicación flexible por su capacidad de personalizar los ejercicios según desee los profesores.

Idear un sistema de generación de nuevos ejercicios tipo es un problema muy complejo, provocado por la multitud de nuevos ejercicios tipo que pueden llegar a existir, aspecto que hace difícil definir un método global para todos ellos.

Por otro lado, el potencial que se le quiere proporcionar a la herramienta, es plantear que si ya existe un ejercicios tipo definido, se puedan crear ejercicios concretos de los ejercicios tipos de manera fácil debido a como va a ser diseñada la estructura de la Base de datos que ha sido implementada en el nuevo sistema.

Teniendo en cuenta esta idea, ahora mostramos qué pasos son necesarios para crear un nuevo ejercicio:

1. Creación de la función generadora: En este punto es donde se define la función encargada de calcular o generar la solución al ejercicio, técnicamente, es el código fuente donde se establece los patrones que se van a definir para el ejercicio.

2. Inserción de un ejercicio tipo: se incluye un nuevo registro en la base de datos, donde se definen como tiene que ser el formato de campos para poder realizar la llamada a la función generadora y como va a ser el enunciado que va a ser mostrado en el momento que el alumno realice un ejercicio.

3. Inserción de un ejercicio concreto basado en un ejercicio tipo existente. Aquí se incluye un nuevo registro en la tabla que especifica para los ejercicios concretos, que va a ser interpretado por la función generadora. En este punto se define y se marcan los límites que serán utilizados en el momento de llamar a la función generadora.

A continuación mostramos un sencillo ejemplo para ver de forma mas claro, de cómo a partir de un ejercicio tipo se pueden crear un número ilimitado de ejercicios concretos.

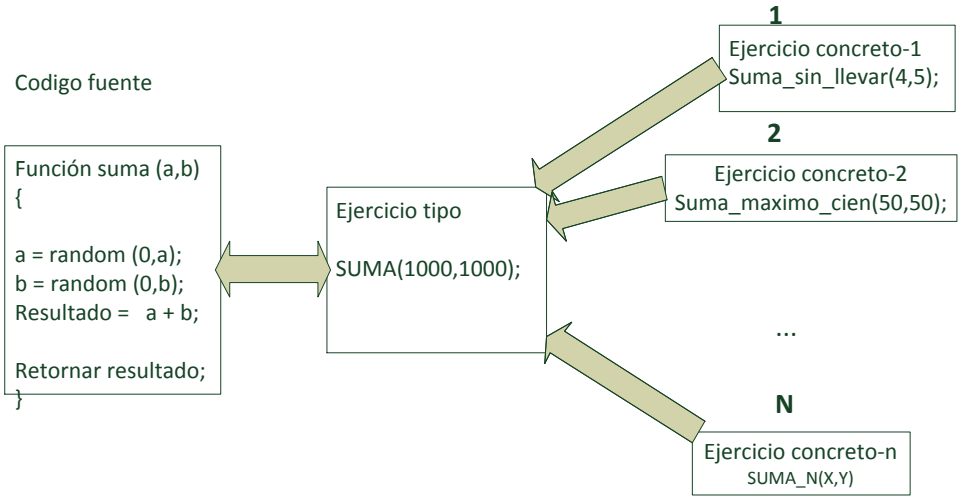


Figura 9: N Ejercicios concretos a partir de un ejercicio tipo

Como hemos visto anteriormente, crear un nuevo ejercicio a partir de un modelo de ejercicio es relativamente fácil , pero si no existe este modelo es necesario realizar una implementación en la aplicación para contenerlo. Hemos analizado la tarea de automatizar la generación de nuevos modelos de ejercicios pero hemos llegado a la conclusión que es una tarea muy difícil, por no decir imposible. Esto viene provocado

porque una de las premisas de la aplicación es poder generar de manera aleatoria una infinidad de ejercicios partiendo siempre de un modelo, después de estudiar diferentes modelos de ejercicios hemos llegado a la conclusión de que no todos los modelos de ejercicios pueden ser creados de manera aleatoria y permitir infinitos ejercicios.

A continuación exponemos los dos modelos que han sido implementados en el proyecto:

- Ejercicios modelo tipo Test: Este ejercicio consiste en mostrar un enunciado donde existen diversas respuesta, donde solo una de ellas es correcta. Todo esto presentado por pantalla.
- Ejercicios modelo dar respuesta: Este ejercicio consiste en mostrar un enunciado y que el alumno por su parte interactúe y de una solución.

Otros modelos de ejercicios que han sido planteados pero que no han sido implementados son:

- Ejercicio de correspondencia entre listas: Este ejercicio consiste básicamente a relacionar elementos de dos listas de elementos presentados por pantalla.
- Ordenación de elementos: Este tipo de ejercicio consiste en poner un orden de lista de elementos según un criterio definido.
- Comprensión de texto: Este tipo de ejercicio consistiría en presentar un texto donde existen espacios vacíos, que han de ser rellenados por el alumno.

2.2.2. Módulo de gestión de ejercicios

Con este módulo el profesor y el administrador del sistema tendrá la posibilidad de, por un lado, crear nuevos ejercicios tipo y por otro poder crear nuevos ejercicios concreto nutriéndose de los ejercicios tipo.

Módulo de gestión de ejercicios tipo: Este módulo es el encargado de definir, modificar y eliminar los diferentes tipos de ejercicios que van a contener la aplicación, dicho de otro modo es la base de conocimiento que contiene la aplicación.

Ahí que matizar que los profesores únicamente tendrá el privilegio de modificar o eliminar aquellos ejercicios tipo que han sido creados por ellos mismos. Hemos definido este concepto para que no exista intrusismo entre los propios profesores, ya que si se permitiera que todos los profesores tuvieran acceso a todas los ejercicios, provocaría que entre ellos se modificarían los ejercicios que se quieren impartir en sus clases.

La gestión de ejercicios tipo permite realizar las siguientes funciones:

- Dar de alta un nuevo ejercicio tipo, De este modo registramos un nuevo ejercicio, relacionando con el código fuente creado en el servidor.
- Editar los datos de los ejercicios tipo.
- Eliminar los ejercicios tipo.

A continuación mostramos el caso de uso de gestión de ejercicio tipo. Existen dos modelos de caso de uso, por un lado, caso de uso del administrador posee el privilegio de modificar y eliminar todos aquellos ejercicios tipo que existen en la aplicación y por otro lado los privilegios de los profesores que únicamente pueden acceder a sus propios ejercicios (ejercicios tipo propios).

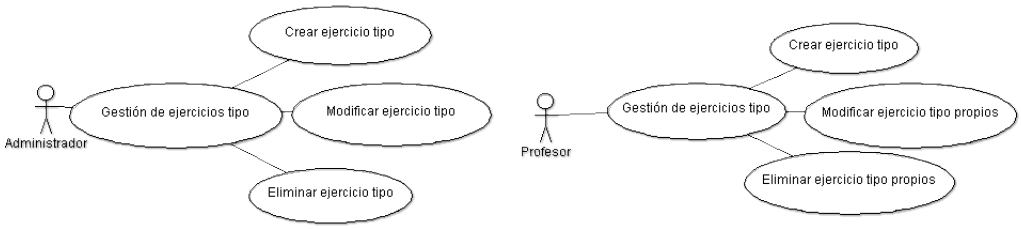


Figura 10: Caso de uso Módulo de gestión ejercicio tipo

Módulo de gestión de ejercicios concreto: La gestión de ejercicios concretos permite la posibilidad de seleccionar cualquiera de los ejercicios tipos definidos en el sistema y poder definir un ejercicio personalizado teniendo en cuenta las propias limitaciones que se están definidas en el ejercicio tipo, esta personalización del ejercicio permite al profesor definir las necesidades pedagógicas que necesitan sus alumnos en todo momento. Igual que en los ejercicios tipo los profesores únicamente tendrán acceso a sus propios ejercicios, para evitar lo comentado anteriormente.

A continuación mostramos que funciones permite realizar la gestión de ejercicios concretos:

- Dar de alta un nuevo ejercicio concreto. Basándonos en un ejercicio tipo que este definido en la BBDD.
- Editar los datos de los ejercicios concretos.
- Eliminar los ejercicios concretos.

Igual que el apartado anterior, el administrador tendrá la capacidad de modificar o eliminar todos los ejercicios concretos que existen en el aplicativo, en cambio, los profesores únicamente tendrán la capacidad de modificar o eliminar los ejercicios concretos definidos por ellos mismo (ejercicios concretos propios) . Por ese motivo se muestran los dos modelos de caso de uso siguientes:

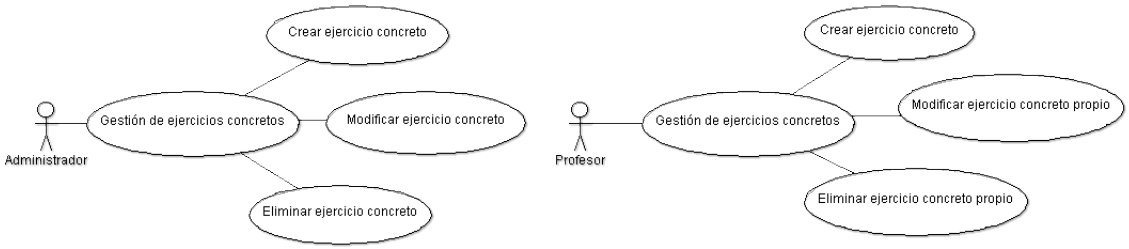


Figura 11: Caso de uso Módulo de gestión ejercicio concreto

2.2.3. Módulo Lanzador y evaluador de ejercicios

A continuación explicaremos qué detalles hemos de tener en cuenta para realizar el módulo lanzador − generador de ejercicios.

El módulo lanzador es el motor encargado de generar de forma aleatoria una serie de ejercicios acorde con el nivel del alumno. Para poder acceder tenemos que considerar las siguientes premisas:

- El usuario que entra en la aplicación tiene que tener permisos de alumno para acceder a este módulo, esto simplemente significa que el profesor haya dado de alta al alumno en la aplicación de gestión.
- Seguidamente se obtiene el nivel del alumno, las materias y los temas que tiene asignados para poder realizar ejercicios acordes a su nivel y materias asignadas.
- Por último se gestiona que tiempo de sesión tiene definido el alumno que viene marcado por la siguiente pauta:
  - Primero comprobamos si el tema de la materia tiene informado el tiempo de la sesión, si no es así, vamos a buscarlo a la materia y en último caso lo iremos a buscar la información a la definición de la escuela.
- Alcanzados los requisitos anteriormente descritos, iniciaríamos la sesión de ejercicios teniendo en cuenta que usuario es, el nivel que tiene y que tiempo va estar realizando ejercicios.
- Al iniciarse la sesión se recogen todos los ejercicios concretos que existan en la base de datos que cumplan con el nivel de alumno y la materia−tema seleccionado.
- Con todos los ejercicios seleccionados en el punto anterior, vamos eligiendo de forma aleatoria un ejercicio que será solucionado por el alumno, hay que tener en cuenta que cada ejercicio tiene definido un tiempo máximo de respuesta.
- Si el ejercicio ha sido resuelto correctamente, registraremos en la base de datos este hecho, y pasaremos a generar un nuevo ejercicio aleatoriamente basándonos en la batería que tenemos.

- De lo contrario si el alumno responde incorrectamente o ha excedido el tiempo máximo para resolver un ejercicio, registraremos el fallo en su primer intento, y le daremos la posibilidad de intentar resolver el ejercicio en 3 intentos, si no responde en esos intentos se almacenará esta información y volveremos a generar un nuevo ejercicio basándonos en la batería que tenemos.

Toda esta casuística se irá sucediendo continuamente hasta que se finalice el tiempo de sesión del cual dispone el alumno.

3. Diseño

Ahora pasamos a ver como está diseñada la aplicación para poder ofrecer todos los requerimientos especificados. Por un lado veremos como está diseñada actualmente la base de datos encargada de guardar todos los datos y por otro mostraremos mas detalladamente como se ha diseñado el módulo lanzador − generador de ejercicios. Por último mediante un ejemplo mostraremos como se da de alta un ejercicio tipo y un ejercicio concreto en la base de datos.

Por último cabe destacar que no se realizará un diseño de la interfaz web porque se pretende mantener el estilo que se inició en el proyecto que gestiona el módulo administrativo de la escuela, mantener diseño propicia que no se vea afectada la integridad de la aplicación STAD.

Primero de todos mostraremos un esquema general de la aplicación para ver como está organizado el entorno de la aplicación.

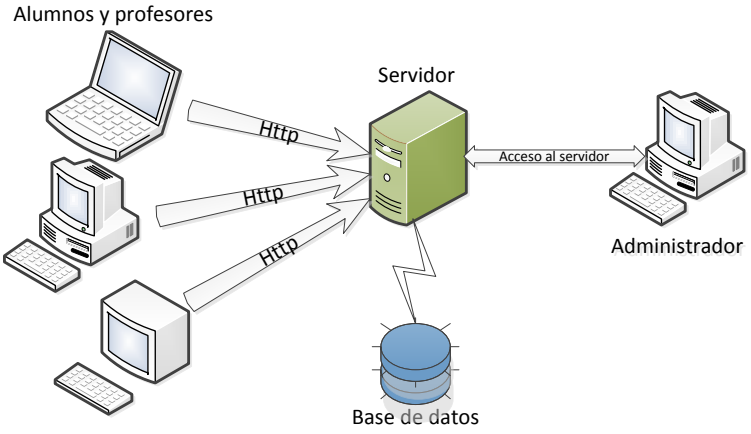


Figura 12: Esquema General

3.1. Diseño de la Base de datos de la aplicación

En este apartado se expone la base de datos de la aplicación [Figura13]. Este diseño fue realizado por los anteriores proyectistas que iniciaron la implementación de STAD y como se ha comentado en la introducción, se ha de intentado modificar

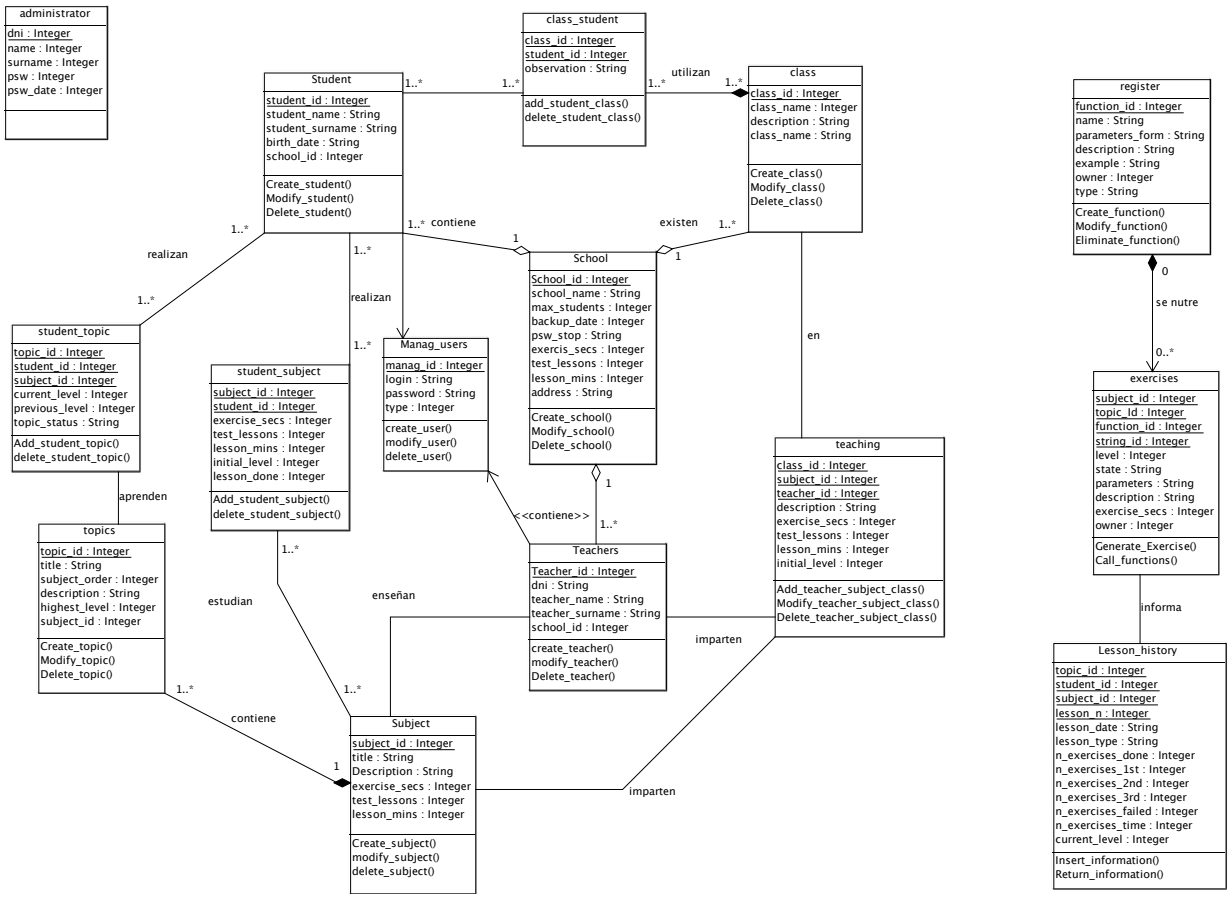
lo menos posible para no tener que modificar en exceso el módulo de gestión que ya existe en la actualidad.

Durante el desarrollo del proyecto está BBDD ha sufrido varios cambios, provocados por los siguientes motivos:

- Inicialmente en la aplicación no se había planteado una gestión de roles, es decir, que existieran diferentes tipos de usuarios con diferentes permisos. Esto ha causado que algunas tablas se vieran afectas para poder definir el rol que tenía cada uno de los usuarios dentro la aplicación y para controlar en todo momento que tipo de usuario está accediendo a STAD. Los cambios que han sufrido la base de datos son los siguientes:
  - Se ha incluido los campos: *manag_id* y *type* en la tabla donde se registran todos los usuarios (*Manag_users−manag_id* y *Manag_users−type*). *Manag_id* ha sido utilizado para poder relacionar cada usuario con su tabla correspondiente, en el caso del alumno es: *student−student_id* y en el caso de los profesores: *teachers−teacher_id*. Luego el campo type se ha utilizada para diferenciar el rol de cada usuario.
  - Ha sido incluido el campo: *format* en la tabla donde se registrar los ejercicios tipo para poder controlar el modelo de ejercicio que queremos mostrar. (*register_format*).
  - Ha sido añadidos el campo: owner en las tablas registro de los ejercicios tipo y ejercicios concretos para controlar que usuario ha registrado el ejercicio. Los campos incluidos en la BBDD son: *register−owner* y *exercise−owner*.
- El otro motivo por el cual se ha modificado la base de datos ha sido el rediseño que ha sufrido la idea inicial del módulo de ejercicios. Este cambio ha sido debido por cómo debía ser almacenado el código fuente de las funciones, inicialmente se planteo almacenar el código de las funciones en un campo de la BBDD, pero estoy implicaba una serie de limitaciones:
  - los campos de BBDD tiene un tamaño máximo, por lo que el código de la funciones no podrían exceder ese número.
  - En caso de corromperse la BBDD, se perdería toda la información existente.

Finalmente hemos tomado la decisión de crear un archivo por cada una de las ejercicios tipo que se deseen crear y estos archivos serán almacenados en una directorio del servidor que ha sido creada para ese propósito, por lo tanto cabe vez que se creará una función esta se almacenada en una ubicación concreta.

Ahora pasamos a ver la estructura del sistema completo mostrando los atributos y las relaciones que existen entre ellas, así como sus funcionalidades.


administrator
dni : Integer
name : Integer
surname : Integer
psw : Integer
psw_date : Integer
Student
student_id : Integer
student_name : String
student_surname : String
birth_date : String
school_id : Integer
Create_student()
Modify_student()
Delete_student()
class_student
class_id : Integer
student_id : Integer
observation : String
add_student_class()
delete_student_class()
utilizan
class
class_id : Integer
class_name : Integer
description : String
class_name : String
Create_class()
Modify_class()
Delete_class()
register
function_id : Integer
name : String
parameters_form : String
description : String
example : String
owner : Integer
type : String
Create_function()
Modify_function()
Eliminate_function()
contiene
existen
realizan
realizan
School
School_id : Integer
school_name : String
max_students : Integer
backup_date : Integer
psw_stop : String
exercis_secs : Integer
test_lessons : Integer
lesson_mins : Integer
address : String
Create_school()
Modify_school()
Delete_school()
en
se nutre
student_topic
topic_id : Integer
student_id : Integer
subject_id : Integer
current_level : Integer
previous_level : Integer
topic_status : String
Add_student_topic()
delete_student_topic()
student_subject
subject_id : Integer
student_id : Integer
exercise_secs : Integer
test_lessons : Integer
lesson_mins : Integer
initial_level : Integer
lesson_done : Integer
Add_student_subject()
delete_student_subject()
Manag_users
manag_id : Integer
login : String
password : String
type : Integer
create_user()
modify_user()
delete_user()
exercises
subject_id : Integer
topic_id : Integer
function_id : Integer
string_id : Integer
level : Integer
state : String
parameters : String
description : String
exercise_secs : Integer
owner : Integer
Generate_Exercise()
Call_functions()
aprenden
<<contiene>>
teaching
class_id : Integer
subject_id : Integer
teacher_id : Integer
description : String
exercise_secs : Integer
test_lessons : Integer
lesson_mins : Integer
initial_level : Integer
Add_teacher_subject_class()
Modify_teacher_subject_class()
Delete_teacher_subject_class()
topics
topic_id : Integer
title : String
subject_order : Integer
description : String
highest_level : Integer
subject_id : Integer
Create_topic()
Modify_topic()
Delete_topic()
Teachers
Teacher_id : Integer
dni : String
teacher_name : String
teacher_surname : String
school_id : Integer
create_teacher()
modify_teacher()
Delete_teacher()
estudian
enseñan
imparten
informa
Lesson_history
topic_id : Integer
student_id : Integer
subject_id : Integer
lesson_n : Integer
lesson_date : String
lesson_type : String
n_exercises_done : Integer
n_exercises_1st : Integer
n_exercises_2nd : Integer
n_exercises_3rd : Integer
n_exercises_failed : Integer
n_exercises_time : Integer
current_level : Integer
Insert_information()
Return_information()
contiene
Subject
subject_id : Integer
title : String
Description : String
exercise_secs : Integer
test_lessons : Integer
lesson_mins : Integer
Create_subject()
modify_subject()
delete_subject()
imparten

3.2. Diseño del módulo lanzador y evaluador de ejercicios

Para entender el módulo lanzador − evaluador de ejercicios necesitamos definir diferentes conceptos y rutinas que van a ser llamadas dentro del módulo.

- Tiempo de sesión. Es el tiempo que dura una sesión de ejercicios. Una vez finalizado el tiempo la sesión de ejercicios finaliza. Para obtener o calcular este tiempo se consultará la relación que define el alumno con la materia (*student_subject−lesson_mins*), en caso de no encontrar información, el siguiente criterio es comprobar si está definido en la definición de la asignatura (*subject−lesson_mins*), concepto que se puede relacionar con el ejercicio que esta realizando el alumno. Por último, si no se a podido establecer dicho tiempo con los anteriores parámetros se consulta la escuela, donde se define de manera obligatorio el tiempo que debe durar una sesión.
- Nivel : Este concepto indica la dificultad del ejercicio que puede realizar el alumno. Esta información la obtenemos de la relación que tiene definido el alumno con el temario del cual quiere realizar un ejercicio (*student_topic*). Con el nivel podremos realizar una filtro de los ejercicios que son idóneos para el alumno y para poder tener información de cúal ha sido el desarrollo del alumno durante el curso. El nivel se calcula del siguiente modo:
  - Acertar el ejercicio en el primer intento, se obtienen 5 puntos.
  - Acertar el ejercicio en el segundo intento, se obtienen 3 puntos.
  - Acertar el ejercicio en el tercer intento, se obtiene 1 punto.
  - Se ha marcado que los alumnos como mínimo deben realizar 100 ejercicios por cada nivel y para poder subir de nivel al menos debe alcanzar los 350 puntos.
- tiempo de ejercicio: Es el tiempo que se le da al alumno para resolver un ejercicio. Esta información se obtiene del ejercicio que se le propone al alumno, en caso de no encontrar la información, se busca en la relación que tiene definido el alumno con el temario, en caso de no encontrarlo, este se encontraría definido en la escuela.
- Intentos: Se ha definido a nivel global de la aplicación que el máximo número de intentos para resolver un ejercicio es de tres intentos.

Los conceptos de tiempo de sesión y tiempo de ejercicio, que acabamos de describir anteriormente se encuentra definida en diferentes niveles para poder dar la capacidad de personalizar las sesiones de ejercicios, así obtenemos una aplicación adaptada a cada alumno.

A continuación mostraremos en formato pseudocódigo el módulo lanzador−generador de ejercicios.

INICIO

1. Entrada de datos por parte del alumno que desea iniciar una sesión de ejercicios. (Internamente la aplicación valida que se trata de un alumno).
2. Selección de la materia. (Se cargan los datos referentes a los ejercicios concretos que utilizan esta materia).
3. Selección de tema. (Se filtran los ejercicios concretos que son utilizados para ese tema y que correspondan al nivel de alumno).
4. Iniciamos el tiempo de sesión y definimos un nueva sesión de ejercicios en la tabla: *lesson_history*.
5. Mientras el tiempo transcurrido sea inferior al tiempo de sesión, se inicia el siguiente proceso:
   - *a*) Seleccionar un ejercicio aleatoriamente que se encuentre activo para ser generado e iniciamos contador de intentos
   - *b*) Identificado el ejercicio donde obtenemos el tiempo máximo para resolver el ejercicio, la solución y posibles respuestas incorrectas.
   - *c*) Recibimos respuesta. Incrementamos contador de intentos limitándolo a 3 intentos.
   - *d*) Respuesta acertada. Retornamos al punto 5.1 y guardamos un histórico informando en que intento ha sido resuelto el ejercicio. Recalculamos el nivel del alumno.
   - *e*) Respuesta errónea. Comprobamos el numero de intentos.
   - *f*) Si el número de intentos es menor a 3. Retornamos al punto 5.2
   - *g*) Si el número de intentos es igual a 3. Guardamos los datos en la tabla de histórico de lección conforme hemos fallado un ejercicio. Retornamos al punto 5.1 para generar un nuevo ejercicio.

6. Al finalizar el tiempo de sesión se recalcula el nivel del alumno para verificar si ha aumentado de nivel.

7. Guardamos información referente a la sesión realizada.

   - *Lesson_history*, los ejercicios que han sido realizados, el número de intento en el cual se ha resuelto cada uno, y el nivel que tiene el alumno.
   - *Student_topic*: actualizamos la tabla en caso de que el alumno haya aumentado el nivel.
   - *Student_subject*: almacenado el numero de sesiones que llevamos realizadas para la asignatura y temario.

8. Mostramos los resultados obtenidos en la sesión cursada.

FIN

A continuación mostramos el diagrama de secuencia[figura 14] del módulo evaluador −lanzador.

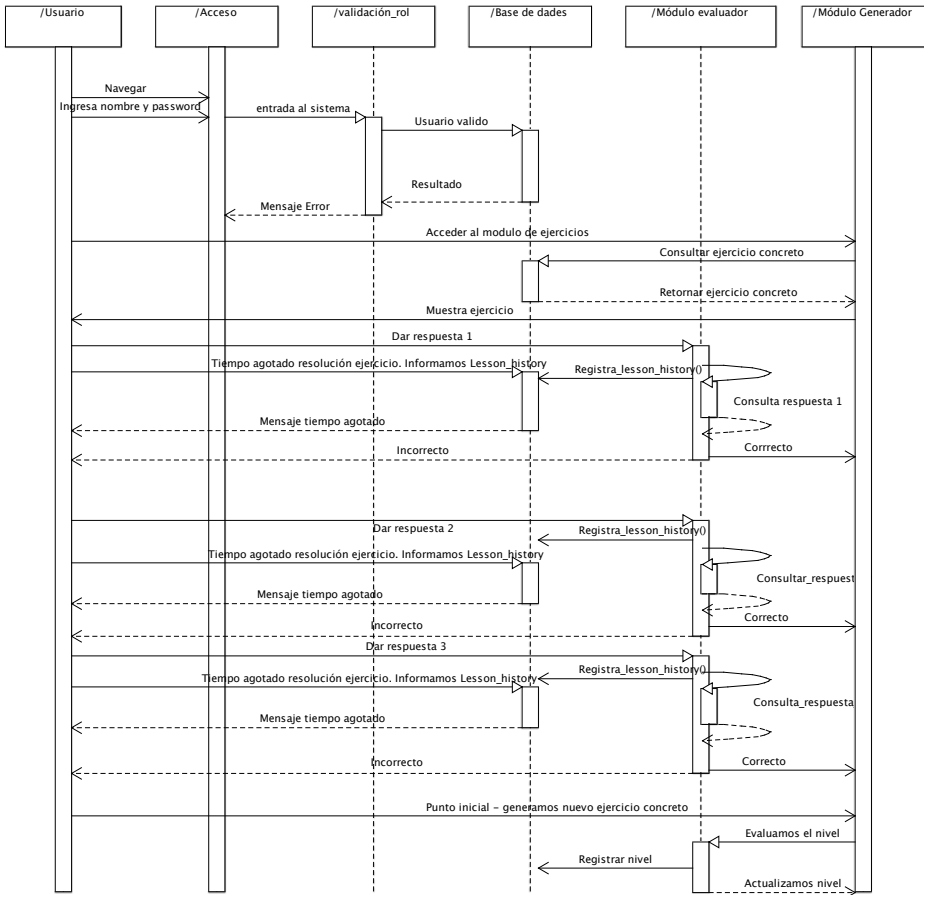


Figura 14: Diagrama de secuencia

3.3. Ejemplo de registrar un ejercicio en la base de datos

Vamos a mostrar como están definidas las tablas utilizadas para guardar la información del módulo de ejercicios y a continuación lo ejemplificaremos con un ejemplo.

Esta tabla es la encargada de almacenar todos los ejercicios tipo que se definan en la aplicación:

Cuadro 1: Estructura de la tabla register

Campo	Tipo	Nulo	Descripción
function_id	int(4)	No	
name	varchar(30)	Sí	NULL
parameters_form	varchar(255)	Sí	NULL
description	varchar(255)	Sí	NULL
example	varchar(255)	Sí	NULL
owner	int(5)	No	
type	varchar(2)	Sí	NULL

Descripción de los campos de la tabla register:

- Function_id: Identificador del ejercicio tipo, se define como campo primario.
- Name: Nombre que define la función.
- Parameters_form: Descripción del formato que va ser utilizado para llamar a la función que genera el ejercicio.
- Description: Describimos que posibilidades ofrece este ejercicio tipo.
- Example: Muestra como tiene que ser defino los parámetros para ser utilizados en el ejercicio concreto.
- Owner: Propietario que ha registrado el ejercicio tipo.
- Type: Define el modelo del ejercicio. Como hemos comentado anteriormente si, es un ejercicio tipo test o si es un tipo de ejercicio donde se espera 1 respuesta.

La tabla Exercise es la encargada de almacenar todos los ejercicios concretos que se definen en STAD:

Cuadro 2: Estructura de la tabla exercise

Campo	Tipo	Nulo	Predeterminado
subject_id	int(3)	No	
topic_id	int(4)	No	
level	int(3)	Sí	NULL
state	varchar(1)	Sí	NULL
function_id	int(4)	No	
string_id	int(4)	No	
parameters	varchar(255)	Sí	NULL
description	varchar(255)	Sí	NULL
exercise_secs	int(3)	Sí	NULL
owner	int(5)	No	

Descripción de los campos de la tabla exercise:

- Subject_id: Identificador numérico de la asignatura, campo primario.
- Topic_id: Identificador numérico del tema, campo primario.
- Level: Nivel necesario para poder realizar el ejercicio.
- State: Estado del ejercicio. Puede ser abierto o cerrado.
- Function_id: Identificar del ejercicio tipo que ha sido definido en la tabla register.
- String_id: Identificar del ejercicio concreto.
- Parameters: Parámetro donde se define como va a ser llamado al ejercicio tipo. Esto permite personalizar el ejercicio.
- Description: Enunciado que va a ser mostrado por pantalla.
- Exercise_secs: Segundo que se dispone para resolver el ejercicio.
- Owner: Propietario que ha registrado el ejercicio concreto.

Ahora mostramos un ejemplo de cómo se registraría un ejercicio tipo y un ejercicio concreto desde el módulo de gestión de ejercicios, para ello damos por hecho que hemos creado un archivo en el servidor de la aplicación con el código fuente del ejercicio [Anexo 8.1].

En este caso queremos recrear una ejercicio tipo donde se puedan realizar todo tipo de operaciones aritméticas simples: sumas, restas, multiplicaciones y divisiones.

Así sería inserción del ejercicio tipo en la base de datos:

- Function_id: 1
- Name: Operaciones
- Parameters_form: operador; operador1_min; operador1_max; operador2_min; operador2_max; Limite;
- Description: Ejercicio operaciones aritmeticas (suma, resta, multiplicación, division)
- Exemple: *;0;100;0;100;10000;
- Owner: 1;
- Type: 1;

Así sería inserción del ejercicio concreto en la base de datos:

- Subject_id : 1(Asignatura: matemáticas)
- Topic_id: 1 (Tema: Suma)
- Level: 5
- State: O
- Function_id: 1
- String_id: 1
- Parameters: +;0;10;0;10;20

- Description: Suma de operando1 + operando2:
- Exercise_secs: 10
- Owner: 1 (identificar del profesor)

Después de esto, cuando un alumno realizaré este ejercicio concreto se generaría la siguiente pantalla:

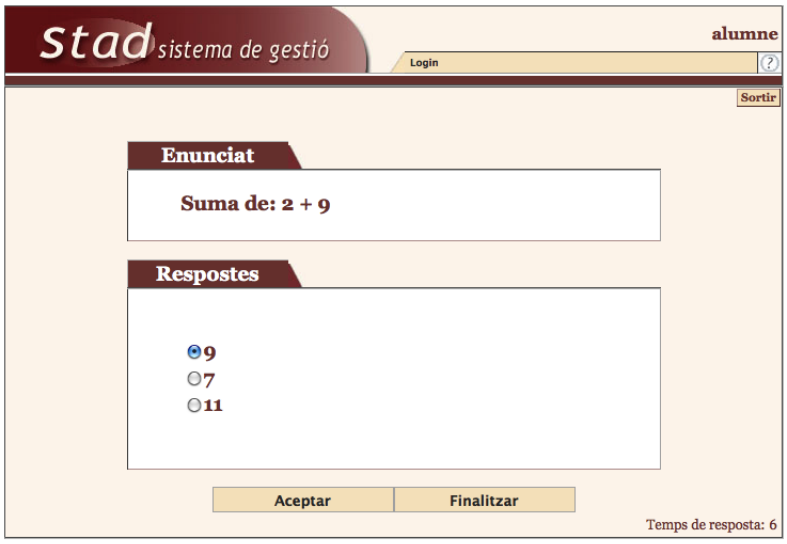


Figura 15: Ejemplo: 1

Otro ejemplo de ejercicio concreto a partir del ejercicio tipo sería:

Así sería inserción del ejercicio concreto en la base de datos:

- Subject_id : 1(Asignatura: matemáticas)
- Topic_id: 1 (Tema: Resta)
- Level: 5
- State: O
- Function_id: 1

- String_id: 2
- Parameters: -;-5;10;-5;10;10
- Description: Resta de operando1 + operando2:
- Exercise_secs: 10
- Owner: 1 (identificar del profesor)

Figura 16: Ejemplo: 2

4. Implementación

En este punto explicaremos cuales han sido las herramientas utilizadas, así como los lenguajes de programación utilizados para realizar la aplicación web.

Para seleccionar las herramientas involucradas en el desarrollo hemos tenido en cuenta una de las premisas que era mantener las herramientas que se utilizaron en los proyectos anteriores por lo aquí presentaremos y justificaremos porque motivo fueron seleccionadas.

4.1. Sistema Operativo

Teniendo en cuenta que nuestra aplicación funcionará sobre un entorno web, no es de vital importancia conocer que sistema operativo es necesario utilizar, ya que actualmente la gran mayoría de estos, están desarrollados con conectividad a internet. En nuestro casos hemos seleccionado Ubuntu que es una distribución GNU/Linux[10], que contiene un núcleo Linux y actualmente es uno de los Sistemas Operativos mas populares, además tiene un sistema de instalación intuitivo y de fácil instalación. Hemos seleccionado este S.O porque mantiene las siguientes premisas que fueron marcadas por los proyectos anteriores:

- Sistema operativo de coste 0.
- Capacidad para hospedar un servidor.
- Por su popularidad y la gran comunidad que desarrolla este S.O, existen aplicaciones potentes que dan soporte al sistema.

4.2. Servidor Web

El servidor web es un programa informático que procesas peticiones del lado del servidor. Generalmente utiliza el protocolo http para realizar estas comunicaciones y atender de este modo las peticiones de los navegadores web.

En los anteriores proyecto que datan de 2003 se tomo la decisión de utilizar Apache. Vamos a ver si actualmente Apache es una opción recomendable.

El servidor http Apache[13] es un servidor web de código abierto para plataformas Unix, Microsoft y Macintosh. Su desarrollo comenzó el año 1995 basándose inicialmente en NCSA HTTPd 1.3[14] que mas tarde fue reescrito por completo. Apache presenta entre otras características altamente configurables, su bases de datos de autentificación y es desarrollado dentro del proyecto HTTP Server de la Apache Software Foundation.

Apache tiene amplia aceptación en la red desde 1996 es el servidor HTTP más usado y alcanzó su máxima cuota de mercado en 2005 siendo el servidor empleado en el 70 % de las paginas web en el mundo. Cabe decir, que durante los últimos años ha bajado un poco estas estadísticas.

4.3. Base de datos

Igual que en el punto anterior analizaremos porqué en los proyectos anteriores se seleccionó MySQL [15] como gestor de base de datos.

MySQL es un sistema gestor de base de datos relacional, multihilo y multiusuario con más de seis millones de instalaciones. MySQL, desde enero de 2008, es una subsidiaria de Sun Microsystems[16] y está a su vez en Oracle Corporation desde abril de 2009 y desarrollando MySQL como software libre en esquema de licencia dual. Por un lado se ofrece como GNU para los usuarios, pero también existen licencias privativas para aquellas empresas que lo deseen.

Las ventajas destacables del gestor MySQL són:

- Velocidad en procesar operaciones.
- Bajo consumo.
- Robustez demostrada por su amplia utilización.
- Su integración con PHP.

4.4. Lenguajes de programación

En este apartado veremos los diferentes lenguajes de programación que han sido utilizados para desarrollar STAD, hemos mantenido los lenguajes que han sido utilizados en anteriores proyectos para que la aplicación siga manteniendo las mismas características.

4.4.1. PHP

PHP es un lenguaje interpretado de alto nivel, embebido en paginas HTML[17] y ejecutado en el servidor, que nos permite desarrollar paginas webs dinámicas. Fue implementado por Rasmus Lerdorf en 1994, pero la implementación principal es producida por *The PHP Foundation* y sirve como standard de facto para PHP.

PHP puede ser desplegado en la mayoría de servidores web y en casi todos los sistemas operativos, sin coste alguno. Este lenguaje se encuentra instalado en mas de 20 millones de sitios web, conjuntamente con Apache es una de las infraestructuras mas utilizadas como servidores web.

Las principales ventajas que ofrece PHP son:

- Multi plataforma.
- Fácil conexión con las mayoría de gestores de base de datos actuales.
- Capacidad de expansión por la gran cantidad de módulos.
- Software libre.
- Permite control de excepciones.

4.4.2. Lenguaje Web

HTML (*HyperText Markup Language* : lenguaje de marcado de hipertexto), es el lenguaje de marcado predominante para la elaboración de páginas web. Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes. HTML también puede describir, hasta un cierto punto, la apariencia de un documento.

A pesar de su gran popularidad, HTML tiene varias restricciones, debido principalmente a sus etiquetas fijas. Esta restricción rompe la premisa que se planteo inicialmente que era la posibilidad de traducir la aplicación STAD a otros idiomas sin que suponga grandes modificaciones en la implementación ni grandes conocimientos para programarlo.

Por lo tanto, para agilizar la posible tarea de traducción, se planteo la necesidad de separar el contenido de la pagina web de su representación, por lo que se decidió en el proyecto de gestión y administración de STAD la utilización de la estructuración XML/XSL. De este modo la traducción de la aplicación consistiría únicamente los textos de los ficheros que nada mas incluye el contenido de la pagina web.

A continuación presentaremos en que consiste XML y XSL y seguidamente expondremos porque la utilización de estas herramientas conjuntas nos permite alcanzar nuestra objetivo.

XML [18] (*Extensible Markup Language*: Lenguaje extensible de marcas) es un lenguaje de esquema utilizado para describir la estructura y las restricciones de los contenidos de los documentos XML de una forma muy precisa.

Veamos algunas ventajas[19] del XML en algunos campos prácticos.

- Comunicación de datos. Si la información se transfiere en XML, cualquier aplicación podría escribir un documento de texto plano con los datos que estaba manejando en formato XML por lo que otra aplicación podrá recibir esta información para poder trabajar con ella.
- Migración de datos. Si tenemos que mover los datos de una base de datos a otra sería muy sencillo si las dos trabajan en formato XML.
- Aplicaciones web. Hasta ahora cada navegador interpreta la información a su manera y los programadores de web tenemos que hacer unas cosas u otras en función del navegador del usuario. Con XML tenemos una sola aplicación que maneja los datos y para cada navegador o soporte podremos tener una hoja de estilo o una similar para aplicarle el estilo adecuado. Si mañana nuestra aplicación debe correr en WAP solo tenemos que crear una nueva hoja de estilo o similar.

XSL[20](*Extensible Stylesheet Language*: lenguaje extensible de hojas de estilo")es una familia de lenguajes basados en el estándar XML que permite describir cómo la

información contenida en un documento XML cualquiera puede ser transformada o formateada para su presentación en un medio. En resumen podemos decir que XSL es el encargado de dar formato a los elementos de los documentos, por lo que conseguir aislar el contenido de la pagina web de su estructura, cumpliendo así el requerimiento que ha hecho necesario utilizar este concepto A continuación mostramos como ha sido generada la pantalla inicial de la aplicación realizando un parser de XML con XSL que nos permite la generación de paginas web, manteniendo en todo momento un estilo en la aplicación.

```
<?xml version="1.0" encoding='iso-8859-1'?>
<?xml-stylesheet href="../xsl/index.xsl" type="text/xsl"?>

<login_page>
  <login_box>
    <login_title>Login</login_title>
    <login_input input_type="text" input_name="p_user" maxsize="10">
       <login_name>Nom Usuari</login_name>
       <login_size>12</login_size>
    </login_input>
    <login_input input_type="password" input_name="p_password" maxsize="6">
       <login_name>Paraula Clau</login_name>
       <login_size>12</login_size>
    </login_input>

  </login_box>
</login_page>
```

Figura 17: Codigo XML

```
    <xsl:template match="login_box">

        <table cellspacing="0" width="230" align="center" class="login_area">

            <xsl:for-each select="login_input">
                <tr>
                    <td width="50%" height="25" class="input_field" >
                        <xsl:value-of select="login_name"/>
                    </td>
                    <td align="left">
                        <input class="field">
                            <xsl:attribute name="name">
                                <xsl:value-of select="@input_name"/>
                            </xsl:attribute>
                            <xsl:attribute name="type">
                                <xsl:value-of select="@input_type"/>
                            </xsl:attribute>
                            <xsl:attribute name="size">
                                <xsl:value-of select="login_size"/>
                            </xsl:attribute>
                            <xsl:attribute name="maxlength">
                                <xsl:value-of select="@maxsize"/>
                            </xsl:attribute>
                        </input>
                    </td>
                </tr>
            </xsl:for-each>
            <tr>
                <td colspan="2" height="10"></td>
            </tr>
            <xsl:apply-templates select="login_submit"/>
            <tr>
                <td colspan="2" height="15"></td>
            </tr>
        </table>
    </xsl:template>
</xsl:stylesheet>
```

Figura 18: Codigo XSL

Y la pantalla resultante es:

Figura 19: Pantalla inicial STAD

4.4.3. Javascript

Se utiliza principalmente en su forma del lado del cliente (client-side), implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas. Todos los navegadores modernos interpretan el código JavaScript integrado en las páginas web. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del Document Object Model (DOM). En STAD ha sido utilizado para mostrar mensajes en caso de seleccionar o indicar casos erróneos, por ejemplo cuando un alumno selecciona una respuesta incorrecta se le muestra un mensaje de error indicándoselo.

4.5. Otras herramientas

4.5.1. Xampp

XAMPP[21] ha sido la herramienta que nos a permitio de manera fácil instalar y gestionar Apache, MySQL y PHP. Xampp es un servidor independiente de plataforma, software libre, que consiste principalmente en la base de datos MySQL, el servidor web Apache y los intérpretes para lenguajes de script: PHP y Perl. El nombre proviene del acrónimo de X (para cualquiera de los diferentes sistemas operativos), Apache, MySQL, PHP, Perl. El programa está liberado bajo la licencia GNU y actúa como un servidor web libre, fácil de usar y capaz de interpretar páginas dinámicas. Actualmente XAMPP está disponible para Microsoft Windows, GNU/Linux, Solaris y MacOS X.

4.5.2. Netbeans

Netbeans[22] ha sido el entorno de desarrollo para generar la aplicación STAD, esta aplicación forma parte de un proyecto de código abierto de gran éxito con una gran base de usuarios, una comunidad en constante crecimiento, y con cerca de 100 socios en todo el mundo en el que destaca Sun MicroSystems. Esta herramienta ha sido la base para programar en los lenguajes: PHP, XML,XSL y Javascript que ha sido necesario para crear la aplicación STAD.

4.5.3. Argo UML

ArgoUML [23]es una aplicación de diagramado de UML escrita en Java y publicada bajo la Licencia BSD. Dado que es una aplicación Java, está disponible en cualquier plataforma soportada por Java. Con este software se han realizado los diferentes diagramas UML que han sido expuestos durante la realización de esta memoria.

5. Pruebas y interficie gráfica

En este capitulo se pretender realizar diferentes pruebas para verificar y analizar posibles errores que hayan sido encontrado durante la implementación del modulo de ejercicios.

En una primera fase, mientras se realizaba el estudio de módulo de gestión se ha ido comprobando que existencia pequeños errores, provocando por el cambio de versión en los lenguajes de programación, por ejemplo existencia funciones PHP que habían pasado a ser obsoletas por lo que fue necesario rehacer algunas partes. Igualmente ocurrió con el formato de composición XML y XSL, ya que en el año que fue implementado (año 2003), era una tecnología relativamente nueva y poco utilizada y que ha sufrido diferentes cambios en el transcurso de estos últimos años.

En una segunda fase, se ha ido realizando pruebas aisladas que han sido realizadas mientras se iban completando los diversos requerimientos. Se podría decir, que ha sido una fase de pruebas acompañada de la fase de implementación.

Por último se ha decido utilizar los tres roles existentes que permite la aplicación, para observar posibles fallos en la aplicación. Aprovechamos esta fase de pruebas para mostrar de nuevo las características de STAD y como ha sido integrado con la aplicación existente.

5.1. Interficie gráfica

En este punto pasamos a ver las diferentes interficies que han sido creadas en la aplicación STAD. Por un lado, veremos el módulo de gestión de ejercicios que ha sido integrado en al módulo de gestión administrativa existe y por otro lado veremos la parte del alumno donde se muestra un ejemplo de ejecución de ejercicios.

5.1.1. Gestión de ejercicios

Como se ve en la figura 20 se han creado dos nuevos accesos que permiten la creación de ejercicios tipo y ejercicios concretos.

Figura 20: Pantalla principal de administración de la escuela

Esta figura 21 podemos ver como podremos crear, modificar y eliminar los ejercicios tipo que hayan sido creados en BBDD(controlando los permisos de modificación y borrado).

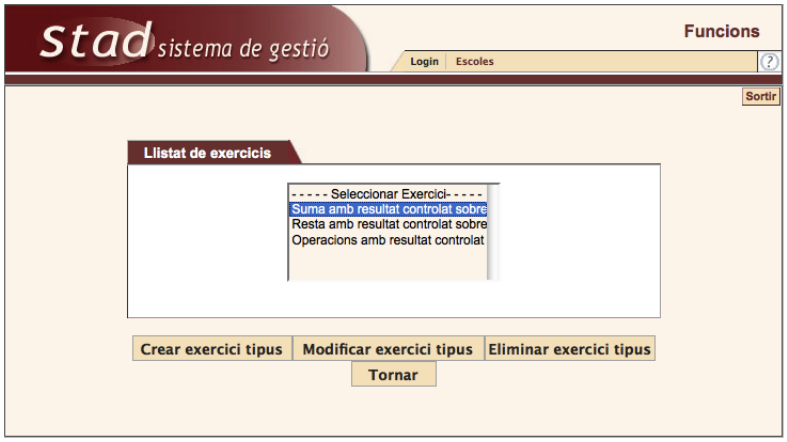


Figura 21: Gestion de ejercicios tipo

En la figura 22, se muestra la definición de un ejercicio concreto, a partir de un ejercicio tipo:

Figura 22: Alta ejercicio concreto

5.1.2. Ejecución del módulo lanzador – evaluador de ejercicios

Iniciado el acceso, vemos como el alumno puede seleccionar la materia y el tema.

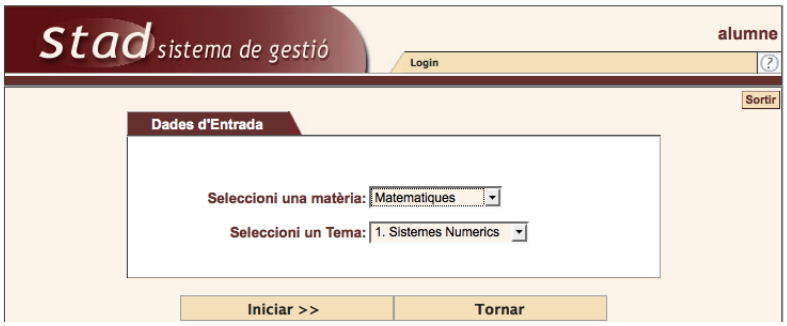


Figura 23: Selección de materia y tema

Seguidamente a la selección de la materia y tema al alumno se le irán sucediendo una serie de ejercicios generador por el módulo lanzador, a su vez el módulo evaluador será el encargado de validar si responde correctamente los ejercicios, si lo hace en el tiempo estimado, además todos estos pasos serán registrados en la BBDD.

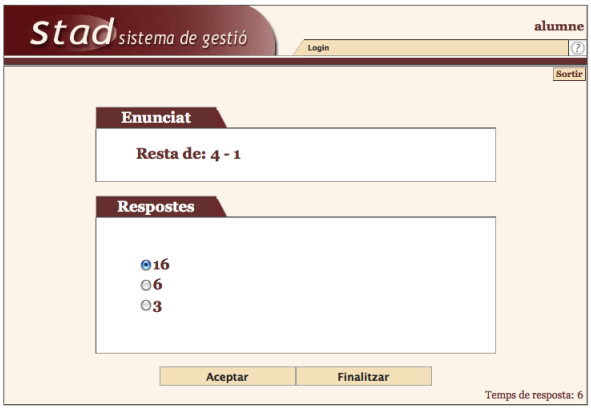


Figura 24: Modelo de ejercicio Test

A continuación mostramos el otro modelo de ejercicio.

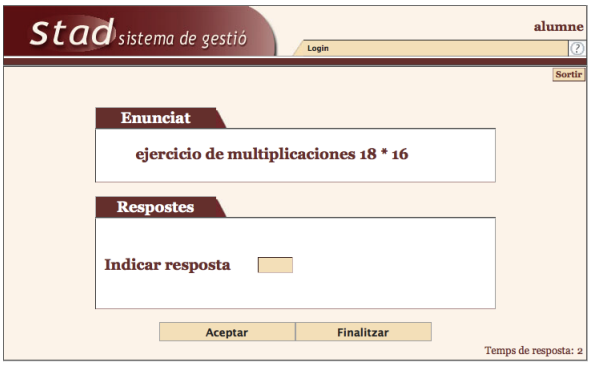


Figura 25: Modelo de ejercicio respuesta

Por último, al finalizar el tiempo se le muestra al alumno un resumen de la sesión.

Figura 26: Resultados de la sesión

6. Conclusiones

Llegado ha este punto es momento de valorar si se han alcanzados los objetivos marcados inicialmente, y si la implementación ha cubierto las expectativas que se pretendían alcanzar.

El hecho de realizar una toma de requerimientos previa a la realización del proyecto ha sido de vital importancia para poder tener una visión global de las necesidades reales que se quiere alcanzar en el sistema STAD, pese a todo ha habido momentos en que se ha tenido que replantear ciertas decisiones y que gracias al análisis de la aplicación se han podido solventar rápidamente, y así conseguir de esta manera alcanzar las hitos marcados.

Los objetivos alcanzados durante el proyecto han sido los siguientes:

- Evaluar y rediseñar la aplicación para darle la capacidad de gestionar ejercicios.
- Implementar un nuevo módulo de ejercicios que este integrado con el módulo de gestión existente.
- Actualizar las diferentes tecnologías que han sido utilizadas en los proyectos anteriores, para que pueda ser utilizada la aplicación en la actualidad.
- Desarrollar un nuevo módulo generador y evaluador de ejercicios, cumpliendo con los requisitos que estaban definidos por la base de datos.

Los objetivos que no han sido alcanzados por su complejidad o por ser inviables en este proyecto son:

- Ampliación de ejercicios tipos. Finalmente no han sido posible crear una gran variedad de modelos de ejercicio ya que finalmente han sido creados dos modelos.

En el aspecto técnico destacar el no conocimiento de algunas herramientas y lenguajes de programación que se han utilizado para desarrollar y con cierta seguridad algunos de los problemas encontrados han sido debidos al estar poco familiarizado con herramientas de desarrollo web, provocando que se tuvieran que realizar cambios varias veces para cumplir con los requisitos definidos.

A pesar de que el trabajo ha sido largo y me he encontrado con diversas dificultades a superar, ha sido muy satisfactorio y enriquecedor poder ver finalizado el proyecto, obteniendo una experiencia personal y profesional al crear un programa con la ayuda del profesor. Y ver que su funcionamiento puede ser una herramienta de trabajo muy completa para escolares que deseen ampliar o reforzar conocimientos.

6.1. Vías de mejora

Con este proyecto se llevado a cabo el "corazón" del sistema, que podrá ir creciendo gracias a las aportaciones de los usuarios y principalmente a la dedicación que quieran aportar a la aplicación los docentes.

Obviamente donde se puede ampliar más este sistema es en la creación de un fondo ampliado de ejercicios operativos para los alumnos, en este proyecto se han definido unos pocos ejercicios para poder dar el ejemplo del desarrollo.

Las mejoras y modificaciones vendrán en el momento en que se decida implantar STAD en una escuela, ya que mediante la utilización de la herramienta se verán cuales son los puntos fuertes y líneas a mejorar en el proyecto. Por lo que sería necesario una comunicación con los usuarios para ir readaptando las aplicación a las nuevas necesidades que aparezcan. Decir que existe la intención de buscar algunas escuelas interesadas en probar el sistema.

Otra posible mejora de la aplicación es dar al módulo de gestión de ejercicios la posibilidad de poder incrustar un editor PHP, para poder generar el código fuente de la aplicación sin la necesidad de utilizar una herramienta externa.

7. Bibliografia

Referencias

[1] http://es.wikipedia.org/wiki/Aprendizaje_electr

[2] J. Cabero. 2006. Bases pedagógicas del e−learning. Disponible en URL: http://www.uoc.edu/rusc/3/1/dt/esp/cabero.html

[3] STAD:Servicio telemático aplicado a la didáctica.

[4] Autor (Valles Castro, Sandra) STAD:El módulo de soporte al Professor pel sistema, Julio de 2004 (05/05/2011)

[5] Autor(Chacón Rey, Antoni) STAD:Núcleo del sistema evaluador de alumnos, septiembre 2003 (05/05/2011)

[6] Pagina oficial: Blackboard, http://www.blackboard.com

[7] Pagina oficial:Desire2Learn, http://www.desire2learn.com/

[8] Pagina oficial:Moodle, http://moodle.org)

[9] GNU/Linux : http://es.wikipedia.org/wiki/GNU/Linux

[10] Pagina oficial: Sakai Project http://sakaiproject.org/

[11] Pagina oficial:Claroline, http://www.claroline.net)

[12] TOAM - Referencia a documentación de otros proyectos

[13] Pagina oficial:Apache http://www.apache.org/

[14] NCSA http://es.wikipedia.org/wiki/NCSA_HTTPd

[15] Pagina oficial:MySQL, http://www.mysql.com/

[16] Pagina oficial:SunMicrosystems, http://www.oracle.com/es/index.html

[17] Pagina oficial:HTML, http://www.w3schools.com/html/

[18] Pagina oficial:XML, http://www.w3c.es/divulgacion/guiasbreves/tecnologiasxml

[19] Autor (Alvarez, Miguel Angel), Objetivos y usos del XML, 21 de junio de 2001 (11/09/2011). http://www.desarrolloweb.com/articulos/460.php

[20] Pagina oficial:XSLT, http://www.w3schools.com/xsl/

[21] Pagina oficial:DOM, http://www.w3.org/DOM/

[22] Pagina oficial:Xampp, http://www.apachefriends.org/es/xampp.html

[23] Pagina oficial:Netbeans, http://netbeans.org/

[24] Pagina oficial:ArgoUML,

[25] ttp://argouml.tigris.org/

8. Anexos

8.1. Codigo fuente

A continuación mostramos el código fuente de unos de los ejercicios que ha sido implementado durante el proyecto. Esta función genera ejercicios de operaciones basicas de sumar, resta, multiplicar y dividir.Aleatoriamente a partir de un operando dado, una serie de parámetros y una limitación en el valor del resultado.

Código PHP.

```
<?php

function operaciones($parameters) {

    switch ($parameters[0]) {
        case '+':
            do {
                $operando1 = rand($parameters[1], $parameters[2]);
                $operando2 = rand($parameters[3], $parameters[4]);
                $solucion = $operando1 + $operando2;
            } while ($solucion > $parameters[5]);

            break;
        case '-':
            do {

                $operando1 = rand($parameters[1], $parameters[2]);
                $operando2 = rand($parameters[3], $parameters[4]);
                $solucion = $operando1 - $operando2;
            } while ($solucion > $parameters[5] && $solucion >= '0');

            break;
        case '*':
            do {

                $operando1 = rand($parameters[1], $parameters[2]);
                $operando2 = rand($parameters[3], $parameters[4]);
                $solucion = $operando1 * $operando2;
            } while ($solucion > $parameters[5] );

            break;
        case '/':
            do {

                $operando1 = rand($parameters[1], $parameters[2]);
                $operando2 = rand($parameters[3], $parameters[4]);
                $solucion = $operando1 / $operando2;
            } while ($solucion > $parameters[5] && is\_int($solucion));
```

```
            break;

        default:
            break;
    }
    $a\_solucion = array($operando1, $operando2, $solucion);

    do {
        $valor1 = mt\_rand($solucion, $parameters[5]);
        $valor2 = mt\_rand(mt\_rand(-100,100), $solucion);
    } while ($solucion == $valor1 or $solucion == $valor2 or $solucion == $valor2);

    $creado = array($solucion, $valor1, $valor2);
    shuffle($creado);
    array\_unshift($creado, $solucion);
    $enunciado = $operando1 . "␣" . $parameters[0] . "␣" . $operando2;
    array\_push($creado, $enunciado);

    return $creado;
}

?>
```

8.2. Archivo SQL para la generación de la base de datos

```
-- phpMyAdmin SQL Dump
-- version 3.2.4
-- http://www.phpmyadmin.net
--
-- Servidor: localhost
-- Tiempo de generación: 13-09-2011 a las 11:40:44
-- Versión del servidor: 5.1.44
-- Versión de PHP: 5.3.1

SET SQL\_MODE="NO\_AUTO\_VALUE\_ON\_ZERO";


/*!40101 SET @OLD\_CHARACTER\_SET\_CLIENT=@@CHARACTER\_SET\_CLIENT */;
/*!40101 SET @OLD\_CHARACTER\_SET\_RESULTS=@@CHARACTER\_SET\_RESULTS */;
/*!40101 SET @OLD\_COLLATION\_CONNECTION=@@COLLATION\_CONNECTION */;
/*!40101 SET NAMES utf8 */;

--
-- Base de datos: `stad`
--

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `administrator`
--

CREATE TABLE IF NOT EXISTS `administrator` (
  `dni` varchar(9) COLLATE latin1\_spanish\_ci NOT NULL,
  `name` varchar(50) COLLATE latin1\_spanish\_ci NOT NULL,
  `surname` varchar(100) COLLATE latin1\_spanish\_ci DEFAULT NULL,
  `password` varchar(6) COLLATE latin1\_spanish\_ci NOT NULL,
  `psw\_date` date DEFAULT NULL,
  PRIMARY KEY (`dni`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1 COLLATE=latin1\_spanish\_ci;

--
-- Volcar la base de datos para la tabla `administrator`
--


-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `classes`
--

CREATE TABLE IF NOT EXISTS `classes` (
  `class\_id` int(4) NOT NULL AUTO\_INCREMENT,
  `class\_name` varchar(25) CHARACTER SET latin1 COLLATE latin1\_spanish\_ci NOT NULL,
  `description` varchar(200) CHARACTER SET latin1 COLLATE latin1\_spanish\_ci DEFAULT NULL,
  `school\_id` int(4) NOT NULL,
  PRIMARY KEY (`class\_id`,`school\_id`)
```

```
) ENGINE=MyISAM  DEFAULT CHARSET=latin1 AUTO\_INCREMENT=7 ;

--
-- Volcar la base de datos para la tabla `classes`
--

INSERT INTO `classes` (`class\_id`, `class\_name`, `description`, `school\_id`) VALUES
(1, '1er A', 'Primer Curs, grup A', 3),
(2, '1er B ', 'Primer Curs, grup B', 3),
(3, '1er C', 'Primer Curs, grup C', 3),
(4, '2on A  ', 'Segon Curs, grup A', 3),
(5, '2on B', 'Segon Curs, grup B', 3),
(6, '2on C ', 'Segon Curs, grup C', 3);

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `class\_student`
--

CREATE TABLE IF NOT EXISTS `class\_student` (
  `class\_id` int(4) NOT NULL,
  `student\_id` int(5) NOT NULL,
  `observation` varchar(200) CHARACTER SET latin1 COLLATE latin1\_spanish\_ci DEFAULT NULL,
  PRIMARY KEY (`class\_id`,`student\_id`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

--
-- Volcar la base de datos para la tabla `class\_student`
--

INSERT INTO `class\_student` (`class\_id`, `student\_id`, `observation`) VALUES
(6, 1, NULL),
(6, 2, NULL),
(6, 21, NULL),
(6, 4, NULL),
(6, 5, NULL),
(6, 6, NULL),
(6, 7, NULL),
(6, 8, NULL),
(6, 9, NULL),
(1, 10, NULL),
(6, 27, NULL),
(6, 3, NULL);

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `exercise`
--

CREATE TABLE IF NOT EXISTS `exercise` (
  `subject\_id` int(3) NOT NULL,
  `topic\_id` int(4) NOT NULL,
  `level` int(3) DEFAULT NULL,
  `state` varchar(1) CHARACTER SET latin1 COLLATE latin1\_spanish\_ci DEFAULT NULL,
  `function\_id` int(4) NOT NULL,
  `string\_id` int(4) NOT NULL AUTO\_INCREMENT,
  `parameters` varchar(255) CHARACTER SET latin1 COLLATE latin1\_spanish\_ci DEFAULT NULL,
```

```
  `description` varchar(255) CHARACTER SET latin1 COLLATE latin1\_spanish\_ci DEFAULT NULL,
  `exercise\_secs` int(3) DEFAULT NULL,
  `owner` int(5) NOT NULL,
  PRIMARY KEY (`subject\_id`,`topic\_id`,`function\_id`,`string\_id`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1 AUTO\_INCREMENT=1 ;

--
-- Volcar la base de datos para la tabla `exercise`
--

INSERT INTO `exercise` (`subject\_id`, `topic\_id`, `level`, `state`, `function\_id`, `string\_id`,
(1, 1, 10, 'O', 1, 1, '+;0;10;0;10;20', 'Suma de:', 10, 2),
(1, 1, 10, 'O', 2, 1, '-;0;10;0;10;20', 'Resta de:', 10, 2),
(1, 1, 10, 'O', 3, 1, '*;0;10;0;10;100', 'Operacions:', 10, 2),
(1, 1, 10, 'O', 3, 2, '*;15;20;15;20;400;', 'ejercicio de multiplicaciones', 10, 2);

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `lesson\_history`
--

CREATE TABLE IF NOT EXISTS `lesson\_history` (
  `lesson\_date` date DEFAULT NULL,
  `lesson\_type` varchar(1) CHARACTER SET latin1 COLLATE latin1\_spanish\_ci DEFAULT NULL,
  `lesson\_n` int(4) NOT NULL AUTO\_INCREMENT,
  `n\_exercises\_done` int(4) DEFAULT NULL,
  `n\_exercises\_1st` int(4) DEFAULT NULL,
  `n\_exercises\_2nd` int(4) DEFAULT NULL,
  `n\_exercises\_3rd` int(4) DEFAULT NULL,
  `n\_exercises\_failed` int(4) DEFAULT NULL,
  `n\_exercises\_time` int(4) DEFAULT NULL,
  `current\_level` int(2) DEFAULT NULL,
  `subject\_id` int(3) NOT NULL,
  `topic\_id` int(4) NOT NULL,
  `student\_id` int(5) NOT NULL,
  PRIMARY KEY (`subject\_id`,`topic\_id`,`student\_id`,`lesson\_n`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1 AUTO\_INCREMENT=1 ;

--
-- Volcar la base de datos para la tabla `lesson\_history`
--

INSERT INTO `lesson\_history` (`lesson\_date`, `lesson\_type`, `lesson\_n`, `n\_exercises\_done`, `
('2011-09-13', 'T', 1, 6, 4, 0, 1, 1, 0, 10, 1, 1, 3);

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `manag\_users`
--

CREATE TABLE IF NOT EXISTS `manag\_users` (
  `manag\_id` int(5) NOT NULL AUTO\_INCREMENT,
  `login` varchar(10) CHARACTER SET latin1 COLLATE latin1\_spanish\_ci NOT NULL,
  `password` varchar(8) CHARACTER SET latin1 COLLATE latin1\_spanish\_ci NOT NULL,
  `type` varchar(2) CHARACTER SET latin1 COLLATE latin1\_spanish\_ci NOT NULL,
  PRIMARY KEY (`manag\_id`)
) ENGINE=MyISAM  DEFAULT CHARSET=latin1 AUTO\_INCREMENT=29 ;
```

```
--
-- Volcar la base de datos para la tabla `manag\_users`
--

INSERT INTO `manag\_users` (`manag\_id`, `login`, `password`, `type`) VALUES
(5, 'Angel', 'angel', '2'),
(6, 'Maria', 'maria', '2'),
(7, 'Elisa', 'elisa', '2'),
(10, 'Margarita', 'Margarit', '3'),
(11, 'Joan', 'Joan', '3'),
(12, 'Marc', 'Marc', '3'),
(13, 'Maria', 'Maria', '3'),
(14, 'Paula', 'Paula', '3'),
(15, 'Sergi', 'Sergi', '3'),
(16, 'Adria', 'Adria', '3'),
(17, 'Lucia', 'Lucia', '3'),
(18, 'Alex', 'Alex', '3'),
(19, 'Gisela', 'Gisela', '3'),
(20, 'Jan', 'Jan', '3'),
(21, 'Daniel', 'Daniel', '3'),
(22, 'David', 'David', '3'),
(23, 'Anna', 'Anna', '3'),
(24, 'Mariona', 'Mariona', '3'),
(1, 'admin', 'admin', '1'),
(2, 'sergio', 'sergio', '2'),
(3, 'alumne', 'alumne', '3'),
(28, 'robert', 'robert', '2'),
(27, 'manuel', 'serser', '3');

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `register`
--

CREATE TABLE IF NOT EXISTS `register` (
  `function\_id` int(4) NOT NULL AUTO\_INCREMENT,
  `name` varchar(30) CHARACTER SET latin1 COLLATE latin1\_spanish\_ci DEFAULT NULL,
  `parameters\_form` varchar(255) CHARACTER SET latin1 COLLATE latin1\_spanish\_ci DEFAULT NULL,
  `description` varchar(255) CHARACTER SET latin1 COLLATE latin1\_spanish\_ci DEFAULT NULL,
  `example` varchar(255) CHARACTER SET latin1 COLLATE latin1\_spanish\_ci DEFAULT NULL,
  `owner` int(5) NOT NULL,
  `type` varchar(2) CHARACTER SET latin1 COLLATE latin1\_spanish\_ci DEFAULT NULL,
  PRIMARY KEY (`function\_id`)
) ENGINE=MyISAM  DEFAULT CHARSET=latin1 AUTO\_INCREMENT=17 ;

--
-- Volcar la base de datos para la tabla `register`
--

INSERT INTO `register` (`function\_id`, `name`, `parameters\_form`, `description`, `example`, `owne
(1, 'suma', 'operacio;operando1\_inf;operando1\_sup;operando2\_min;operando2\_max;Limit\_resultat;'
(2, 'resta', 'operacio;operando1\_inf;operando1\_sup;operando2\_min;operando2\_max;Limit\_resultat;
(3, 'operaciones', 'operacio;operando1\_inf;operando1\_sup;operando2\_min;operando2\_max;Limit\_re

-- --------------------------------------------------------

--
```

```
-- Estructura de tabla para la tabla `schools`
--

CREATE TABLE IF NOT EXISTS `schools` (
  `school\_id` int(4) NOT NULL AUTO\_INCREMENT,
  `school\_name` varchar(50) CHARACTER SET latin1 COLLATE latin1\_spanish\_ci NOT NULL,
  `address` varchar(200) CHARACTER SET latin1 COLLATE latin1\_spanish\_ci DEFAULT NULL,
  `max\_students` int(4) DEFAULT NULL,
  `backup\_date` date DEFAULT NULL,
  `psw\_stop` varchar(6) CHARACTER SET latin1 COLLATE latin1\_spanish\_ci DEFAULT NULL,
  `exercise\_secs` int(3) DEFAULT NULL,
  `test\_lessons` int(3) DEFAULT NULL,
  `lesson\_mins` int(3) DEFAULT NULL,
  PRIMARY KEY (`school\_id`)
) ENGINE=MyISAM  DEFAULT CHARSET=latin1 AUTO\_INCREMENT=5 ;

--
-- Volcar la base de datos para la tabla `schools`
--

INSERT INTO `schools` (`school\_id`, `school\_name`, `address`, `max\_students`, `backup\_date`, `p
(1, 'Collegi␣Elisa␣Badia', 'Rda.␣de␣l''Est,␣s/n,␣Barbera', 1400, NULL, NULL, NULL, NULL),
(2, 'Escola␣del␣Carme', 'C/Josep␣Renom,␣s/n,␣Sabadell', 1200, NULL, NULL, NULL, NULL),
(3, 'Collegi␣Salessians', 'Avda.␣Sabadell,␣s/n,␣Sabadell', 1200, NULL, NULL, 10, 50, 1);

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `students`
--

CREATE TABLE IF NOT EXISTS `students` (
  `student\_id` int(5) NOT NULL AUTO\_INCREMENT,
  `student\_name` varchar(50) CHARACTER SET latin1 COLLATE latin1\_spanish\_ci NOT NULL,
  `student\_surname` varchar(100) CHARACTER SET latin1 COLLATE latin1\_spanish\_ci NOT NULL,
  `birth\_date` date DEFAULT NULL,
  `observation` varchar(200) CHARACTER SET latin1 COLLATE latin1\_spanish\_ci DEFAULT NULL,
  `school\_id` int(4) NOT NULL,
  PRIMARY KEY (`student\_id`,`school\_id`)
) ENGINE=MyISAM  DEFAULT CHARSET=latin1 AUTO\_INCREMENT=28 ;

--
-- Volcar la base de datos para la tabla `students`
--

INSERT INTO `students` (`student\_id`, `student\_name`, `student\_surname`, `birth\_date`, `observa
(10, 'Margarita', 'Garcia␣Guerra', '1998-10-15', '', 3),
(11, 'Joan␣Josep', 'Losada␣Puig', '1998-01-07', NULL, 3),
(12, 'Marc', 'Puigventos␣Rovira', '1998-03-08', NULL, 3),
(13, 'Maria', 'Castro␣Torres', '1998-11-25', NULL, 3),
(14, 'Paula', 'Xicoy␣Canadell', '1998-07-31', NULL, 3),
(15, 'Sergi', 'Murillo␣Perez', '1998-10-15', NULL, 3),
(16, 'Adrian', 'Domenich␣Vendrell', '1998-10-15', NULL, 3),
(17, 'Lucia', 'Roca␣Lorenzo', '1998-06-23', NULL, 3),
(18, 'Alex', 'Corbalan␣Cots', '1998-05-15', NULL, 3),
(19, 'Gisela', 'Simon␣Gumbert', '1998-02-02', NULL, 3),
(20, 'Jan', 'Figuera␣Ridao', '1998-04-19', NULL, 3),
(21, 'Daniel', 'Aguilar␣Martinez', '1998-12-25', '', 3),
(22, 'David', 'Sierra␣Tena', '1998-10-07', NULL, 3),
```

```
(23, 'Anna', 'Gimeno Xipell', '1998-08-07', '', 3),
(24, 'Mariona', 'Soy Martinez', '1998-06-10', NULL, 3),
(3, 'alumne', 'alumne', '1998-06-10', 'Pruebas realizar ejercicios', 3),
(26, 'sergio', 'Rodriguez Perez', '1983-08-25', 'Proyectista', 3),
(27, 'manuel', 'ignacio', '1983-08-25', 'pruebas', 3);

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `student_subject`
--

CREATE TABLE IF NOT EXISTS `student_subject` (
  `exercise_secs` int(3) DEFAULT NULL,
  `test_lessons` int(3) DEFAULT NULL,
  `lesson_mins` int(3) DEFAULT NULL,
  `initial_level` int(3) DEFAULT NULL,
  `lessons_done` int(3) DEFAULT NULL,
  `subject_id` int(4) NOT NULL,
  `student_id` int(5) NOT NULL,
  PRIMARY KEY (`subject_id`,`student_id`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

--
-- Volcar la base de datos para la tabla `student_subject`
--

INSERT INTO `student_subject` (`exercise_secs`, `test_lessons`, `lesson_mins`, `initial_level`
(10, 15, 20, 20, NULL, 1, 1),
(10, 15, 20, 21, NULL, 1, 2),
(10, 15, 20, 22, NULL, 1, 4),
(10, 15, 20, 20, NULL, 1, 5),
(10, 15, 20, 20, NULL, 1, 6),
(10, 15, 20, 20, NULL, 1, 7),
(10, 15, 20, 20, NULL, 1, 8),
(10, 15, 20, 20, NULL, 1, 9),
(10, 14, 20, 18, NULL, 3, 21),
(10, 12, 20, 18, NULL, 2, 1),
(10, 12, 20, 18, NULL, 2, 2),
(10, 50, 1, 10, NULL, 1, 3),
(10, 12, 20, 18, NULL, 2, 4),
(10, 12, 20, 18, NULL, 2, 5),
(10, 12, 20, 18, NULL, 2, 6),
(10, 12, 20, 18, NULL, 2, 7),
(10, 12, 20, 18, NULL, 2, 8),
(10, 12, 20, 18, NULL, 2, 9),
(10, 14, 20, 18, NULL, 2, 21),
(10, 14, 20, 16, NULL, 3, 1),
(10, 14, 20, 16, NULL, 3, 2),
(10, 14, 20, 16, NULL, 3, 4),
(10, 14, 20, 16, NULL, 3, 5),
(10, 14, 20, 16, NULL, 3, 6),
(10, 14, 20, 16, NULL, 3, 7),
(10, 14, 20, 16, NULL, 3, 8),
(10, 14, 20, 16, NULL, 3, 9),
(10, 15, 20, 20, NULL, 1, 21),
(10, 15, 20, 20, NULL, 1, 27),
(10, 14, 20, 18, NULL, 2, 27),
(10, 14, 20, 18, NULL, 3, 27);
```

```
-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `student\_topic`
--

CREATE TABLE IF NOT EXISTS `student\_topic` (
  `current\_level` int(3) DEFAULT NULL,
  `previous\_level` int(3) DEFAULT NULL,
  `topic\_status` varchar(1) CHARACTER SET latin1 COLLATE latin1\_spanish\_ci DEFAULT NULL,
  `topic\_id` int(4) NOT NULL,
  `student\_id` int(5) NOT NULL,
  `subject\_id` int(3) NOT NULL,
  PRIMARY KEY (`topic\_id`,`student\_id`,`subject\_id`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

--
-- Volcar la base de datos para la tabla `student\_topic`
--

INSERT INTO `student\_topic` (`current\_level`, `previous\_level`, `topic\_status`, `topic\_id`, `s
(26, 24, 'O', 1, 1, 1),
(25, 24, 'O', 2, 1, 1),
(10, 10, 'O', 3, 1, 1),
(26, 24, 'O', 4, 1, 1),
(26, 24, 'O', 5, 1, 1),
(26, 24, 'O', 8, 1, 1),
(26, 24, 'O', 9, 1, 1),
(26, 24, 'O', 14, 1, 1),
(23, 24, 'O', 1, 2, 1),
(26, 24, 'O', 2, 2, 1),
(26, 24, 'O', 3, 2, 1),
(26, 24, 'O', 4, 2, 1),
(28, 24, 'O', 5, 2, 1),
(34, 24, 'O', 6, 2, 1),
(33, 30, 'O', 8, 2, 1),
(33, 30, 'O', 9, 2, 1),
(31, 30, 'O', 14, 2, 1),
(10, 24, 'O', 1, 3, 1);

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `subjects`
--

CREATE TABLE IF NOT EXISTS `subjects` (
  `subject\_id` int(4) NOT NULL AUTO\_INCREMENT,
  `title` varchar(50) COLLATE utf8\_spanish\_ci NOT NULL,
  `description` varchar(200) COLLATE utf8\_spanish\_ci DEFAULT NULL,
  `exercise\_secs` int(3) DEFAULT NULL,
  `test\_lessons` int(3) DEFAULT NULL,
  `lesson\_mins` int(3) DEFAULT NULL,
  PRIMARY KEY (`subject\_id`)
) ENGINE=MyISAM  DEFAULT CHARSET=utf8 COLLATE=utf8\_spanish\_ci AUTO\_INCREMENT=4 ;

--
-- Volcar la base de datos para la tabla `subjects`
```

```
--

INSERT INTO `subjects` (`subject\_id`, `title`, `description`, `exercise\_secs`, `test\_lessons`, `
(1, 'Matematiques', NULL, NULL, NULL, NULL),
(2, 'Angles', NULL, NULL, NULL, NULL),
(3, 'Ciencies', NULL, NULL, NULL, NULL);

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `teachers`
--

CREATE TABLE IF NOT EXISTS `teachers` (
  `teacher\_id` int(4) NOT NULL AUTO\_INCREMENT,
  `dni` varchar(9) CHARACTER SET latin1 COLLATE latin1\_spanish\_ci NOT NULL,
  `teacher\_name` varchar(50) CHARACTER SET latin1 COLLATE latin1\_spanish\_ci NOT NULL,
  `teacher\_surname` varchar(100) CHARACTER SET latin1 COLLATE latin1\_spanish\_ci NOT NULL,
  `school\_id` int(4) NOT NULL,
  PRIMARY KEY (`teacher\_id`,`school\_id`)
) ENGINE=MyISAM  DEFAULT CHARSET=latin1 AUTO\_INCREMENT=29 ;

--
-- Volcar la base de datos para la tabla `teachers`
--

INSERT INTO `teachers` (`teacher\_id`, `dni`, `teacher\_name`, `teacher\_surname`, `school\_id`) VA
(1, '40566989P', 'Angel', 'Riera Boix', 3),
(2, '43966999K', 'Sergio', 'Sergio', 3),
(3, '33814533E', 'Elisa', 'Serrano Torres', 3),
(28, '123123456', 'robert', 'robert', 3);

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `teaching`
--

CREATE TABLE IF NOT EXISTS `teaching` (
  `description` varchar(200) DEFAULT NULL,
  `exercise\_secs` int(3) DEFAULT NULL,
  `test\_lessons` int(3) DEFAULT NULL,
  `lesson\_mins` int(3) DEFAULT NULL,
  `initial\_level` int(3) DEFAULT NULL,
  `class\_id` int(4) NOT NULL,
  `subject\_id` int(4) NOT NULL,
  `teacher\_id` int(4) NOT NULL,
  PRIMARY KEY (`class\_id`,`subject\_id`,`teacher\_id`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

--
-- Volcar la base de datos para la tabla `teaching`
--

INSERT INTO `teaching` (`description`, `exercise\_secs`, `test\_lessons`, `lesson\_mins`, `initial\
(NULL, 10, 15, 20, 20, 6, 1, 1),
(NULL, 10, 12, 20, 18, 6, 2, 2),
(NULL, 10, 12, 20, 18, 6, 3, 2),
(NULL, 10, 14, 20, 16, 6, 3, 3),
```

```
(NULL, 10, 15, 20, 20, 5, 3, 1),
(NULL, 100, 100, 100, 100, 4, 1, 2),
(NULL, 10, 14, 20, 16, 6, 2, 3),
(NULL, NULL, NULL, NULL, NULL, 6, 2, 1);

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `topics`
--

CREATE TABLE IF NOT EXISTS `topics` (
  `topic\_id` int(4) NOT NULL AUTO\_INCREMENT,
  `title` varchar(50) NOT NULL,
  `subject\_order` int(11) NOT NULL,
  `description` varchar(200) DEFAULT NULL,
  `highest\_level` int(3) DEFAULT NULL,
  `lowest\_level` int(3) DEFAULT NULL,
  `school\_id` int(4) DEFAULT NULL,
  `subject\_id` int(4) DEFAULT NULL,
  PRIMARY KEY (`topic\_id`)
) ENGINE=MyISAM  DEFAULT CHARSET=latin1 AUTO\_INCREMENT=16 ;

--
-- Volcar la base de datos para la tabla `topics`
--

INSERT INTO `topics` (`topic\_id`, `title`, `subject\_order`, `description`, `highest\_level`, `low
(1, 'Sistemes Numerics', 1, NULL, 76, 10, 3, 1),
(2, 'Suma/Resta fins a 20', 2, NULL, 29, 10, 3, 1),
(3, 'Suma Vertical', 3, NULL, 47, 25, 3, 1),
(4, 'Resta Vertical', 4, NULL, 49, 26, 3, 1),
(5, 'Multiplicacio i Divisio fins a 100', 5, NULL, 37, 20, 3, 1),
(6, 'Multiplicacio Vetical', 6, NULL, 54, 33, 3, 1),
(7, 'Divisio Llarga', 7, NULL, 59, 37, 3, 1),
(8, 'Aritmetica Mental', 8, NULL, 79, 25, 3, 1),
(9, 'Ecuacions i Leis de la Aritmetica', 9, NULL, 79, 17, 3, 1),
(10, 'Mesures i Pes', 10, NULL, 74, 40, 3, 1),
(11, 'Fraccions Simples', 11, NULL, 79, 40, 3, 1),
(12, 'Decimals i Percentatges', 12, NULL, 79, 50, 3, 1),
(13, 'Sencers (Positius i Negatius)', 13, NULL, 79, 50, 3, 1),
(14, 'Problemes', 14, NULL, 69, 20, 3, 1),
(15, 'Divisibilitat , Factorització i Potencies', 15, NULL, 79, 44, 3, 1);
```

(Resumen, resúm, summary)

En este proyecto se ha creado un nuevo módulo de gestión y generación de ejercicios que ha sido integrado a la aplicación STAD. STAD es una aplicación e-learning que sirve como sistema de aprendizaje destinado a la educación primaria.

Con el nuevo módulo de ejercicios se ha ideado un nuevo concepto que permite generar un número ilimitado de ejercicios y que permite gestionar de manera sencilla estos ejercicios.

A aquest projecte s'ha creat un nou módul de gestió i generació d'exercicis que ha sigut integrat a la aplicació STAD. STAD es una aplicació e-learning que serveix com a sistema d'aprenentatge destinat a l'educació primaria.

Amb el nou módul de exercicis s'ha ideat un nou concepte que permet generar un nombre ilimitat d'exercicis i permet gestionar de forma sencilla aquests exercicis.

This project has created a new management module and generation of exercises has been integrated into the application STAD. STAD is an application e-learning that serves as a learning system for primary education.

With the new module has exercises devised a new concept for generating an unlimited number of exercises and allows to easily manage the creation of these.