PROYECTO FINAL DE CARRERA

JLAB - Módulo de Integración de los Simuladores de Comunicaciones LAVICAD en Moodle

(JLAB - Moodle Activity for LAVICAD Communications Simulators integration in Moodle)

Estudios: Ingeniería de Telecomunicación

Autora: Mar Pérez Casares

Directora: Margarita Cabrera Bean

Año: 2016

Índice General

Lista de Figuras

Colaboraciones

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Departament de Teoria del Senyal i Comunicacions

Agradecimientos

A mis padres por su esfuerzo,

a mi marido por sus ánimos y paciencia

y a Marga por su comprensión y apoyo.

Resum del Projecte

El proyecte JLab es la continuació del treball realitzat pels companys del grup GILAVBIR per aconseguir un laboratori de comunicacions virtual accessible des del campus virtual de la UPC, Atenea, que és una plataforma Moodle.

Inicialment el projecte LAVICAD es va encarregar de desenvolupar una sèrie d'applets que implementen simuladors de diversos sistemes de comunicació donant la possibilitat a l'usuari de configurar i avaluar cada etapa del sistema.

L'objectiu de JLab és integrar aquests simuladors en Moodle per oferir als alumnes de les diverses enginyeries l'opció de realitzar els experiments a distancia i permetre al professorat proposar experiments i realitzar el seu seguiment des del mateix portal Atenea.

Resumen del Proyecto

El proyecto Jlab que es la continuación del trabajo realizado por otros compañeros del grupo GILABVIR para conseguir un laboratorio de comunicaciones virtual accesible desde el campus virtual de la UPC, Atenea, que es una plataforma Moodle.

Inicialmente el proyecto LAVICAD se encargó de desarrollar una serie de applets que implementan simuladores de varios sistemas de comunicación permitiendo al usuario configurar y evaluar cada etapa del sistema.

El objetivo de JLAB es integrar estos simuladores en Moodle para ofrecer a los alumnos de las diversas ingenierías la opción de realizar los experimentos a distancia y permitir al profesorado proponer experimentos y realizar su seguimiento desde el mismo portal Atenea.

Abstract

The JLAB project is the continuation of the work done by other colleagues in the GILABVIR group for a virtual communications lab accessible from the virtual campus of the UPC, Athena, a Moodle platform.

LAVICAD project initially was responsible for developing a series of applets simulators that implement various communication systems allowing the user to configure and evaluate each stage of the system .

The JLAB goal is to integrate these simulators in Moodle to offer engineering students the option to perform experiments remotely and allow teachers to propose experiments and track them from the same portal Athena.

1. Introducción

El imparable desarrollo tecnológico iniciado en el siglo XX en Electrónica, Informática y Telecomunicaciones ha dado paso al siglo XXI con su mayor exponente, el vertiginoso crecimiento de INTERNET. Las tecnologías de la Información y Comunicación (TICs) han convertido el acceso a la información y a la comunicación en global, descentralizada, continua e instantánea y a nuestra sociedad en la sociedad de la Información y/o el Conocimiento.

El resultado es la transformación de todos los ámbitos de la sociedad incluido el mundo educativo universitario que debe evolucionar continuamente para adaptarse.

Los factores de cambio de la cultura universitaria [1] son:

- Aprovechamiento de las TIC
- Acceso generalizado de estudiantes
- Necesidad de formación continua "a medida"
- Mayor exigencia de calidad y flexibilidad
- Transformación de la función docente
- Gestión universitaria más descentralizada
- Investigaciones multidisciplinares y grupales
- Mayor presión competitiva

Es práctica ya consolidada que las universidades presenciales complementen sus servicios habituales con un "campus virtual" donde la institución, los profesores y los alumnos puedan compartir información y facilitar trámites administrativos.

En el caso de las ingenierías se plantea además la búsqueda de herramientas que ofrezcan a sus alumnos la posibilidad de la experimentación a distancia. Ofrece un espacio compartido donde los profesores puedan proponer prácticas y los alumnos realizarlas pero sin limitaciones de horarios ni distancias, es decir, ofrece un laboratorio virtual a profesores y alumnos.

1.1 Laboratorio Virtual de la UPC, Atenea

La Universidad Politécnica de Cataluña (UPC) hace frente a este reto con una serie de proyectos de investigación en Innovación Educativa. Uno de estos proyectos es RIMA (Investigación e Innovación en Metodologías de Aprendizaje) [2] que se desarrolló para promover la innovación en el desarrollo de metodologías de aprendizaje aplicadas a la educación.

Dentro del marco de investigación del proyecto RIMA surgió el grupo de interés GILABVIR (Grup d'Interés en Laboratoris Virtuals I Remots) [3]. El grupo GILABVIR está formado por miembros de la comunidad educativa involucrados en diferentes laboratorios de asignaturas de Ingeniería. Todos ellos se caracterizan por el uso de experimentos reales y/o simulados accesibles desde la plataforma de Campus Digital de la UPC basada en Moodle, ATENEA [5]. Los experimentos de los diferentes cursos de laboratorio suelen seguir la siguiente secuencia de pasos:

1) El estudiante diseña el experimento y configura los parámetros on-line, ya sea desde casa o desde el ordenador de la clase.

2) El experimento es ejecutado.

3) Los diferentes resultados (numéricos, gráficos, imágenes, etc.) son visualizados y opcionalmente grabados en el ordenador personal del estudiante y también opcionalmente grabados en el servidor de ATENEA.

Se han llegado a integrar once laboratorios en GILABVIR [4] y se utilizan en asignaturas correspondientes a los planes de estudios de: Ingeniería Electrónica, Ingeniería de Telecomunicaciones, Ingeniería Informática, Ingeniería Industrial e Ingeniería Civil. Los aspectos técnicos y didácticos de todos ellos han sido recogidos y clasificados. Los dos objetivos principales de este grupo son detectar las necesidades comunes de las diferentes soluciones técnicas de los diferentes laboratorios y diseñar nuevas metodologías educativas que usen actividades formativas basadas en laboratorios virtuales y remotos.

El principal objetivo de GILABVIR es la implementación de una aplicación informática que una todos los laboratorios virtuales y remotos en el campus digital de la UPC (plataforma Moodle) y permitir a los alumnos ejecutar experimentos y a los profesores proponerlos, monitorizarlos y evaluar estos experimentos.

1.2 Plataforma de Formación Moodle

El Campus Virtual de la UPC, Atenea, está basado en la plataforma web de formación Moodle [6].

Moodle es un Sistema de Gestión de Cursos (Course Management System, CMS), también conocido como Sistema de Gestión del Aprendizaje (Learning Management System, LMS) o

Entorno Virtual de Aprendizaje (Virtual Learning Environment, VLE). Es una aplicación web orientada a los educadores para crear portales de formación a distancia efectivos y colaborativos.

Moodle está basado en Código Abierto, por lo que es distribuido y modificado libremente. Se basa en la idea de que si los programadores (en Internet) pueden leer, modificar y redistribuir el código fuente de un programa, éste evoluciona, se desarrolla y mejora. Los usuarios lo adaptan a sus necesidades, corrigen sus errores a una velocidad impresionante, mayor a la aplicada en el desarrollo de software convencional o cerrado, dando como resultado la producción de un mejor software. La primera versión se publicó en 2002 y en la actualidad, 2016, ya van por la versión 3.0 contando con más de 86 millones de usuarios en todo el mundo.

Registered sites	73,703
Countries	229
Courses	9,611,137
Users	86,024,963
Enrolments	251,909,086
Forum posts	174,197,636
Resources	86,926,495
Quiz questions	437,533,517

Figura 1: Estadísticas de uso de Moodle en todo el mundo.

A nivel pedagógico, Moodle está diseñado para soportar un modelo educativo constructivista, enfatizando que los que aprenden (y no solo los profesores) puedan contribuir en la experiencia formativa [6].

La palabra Moodle fue originalmente un acrónimo de Entorno de Aprendizaje Modular Dinámico y Orientado a Objetos (Modular Object-Oriented Dynamic Learning Environment).

Moodle es modular de base y puede ser ampliado mediante la creación de módulos específicos que aporten nuevas funcionalidades al portal. La infraestructura de Moodle soporta una gran variedad de

tipos de módulos: Actividades, Recursos, Preguntas, Campos de Datos, Temas gráficos, métodos de autenticación, filtros de contenido y hechos a medida [6].

1.3 Laboratorio Virtual de Comunicaciones LAVICAD

Dentro de este marco de innovación educativa en la Escuela Técnica Superior de Ingeniería de Telecomunicación de Barcelona (ETSETB) un grupo de profesores del departamento de TSC (Teoría de la Señal y Comunicación) ha promovido el desarrollo del proyecto LAVICAD [12], consistente en un laboratorio virtual de comunicaciones analógicas y digitales basado en applets de Java.

La motivación de este proyecto fin de carrera es conseguir que estos simuladores se comuniquen con la plataforma Moodle de la UPC, Atenea, para enviar a los profesores los resultados de las simulaciones en tiempo real y permitir de este modo obtener de forma automática más elementos de evaluación en el contexto de una asignatura.

En el momento en que se inició el presente proyecto final de carrera el grupo GILABVIR tenía en marcha un proyecto más amplio que tenía como objetivo crear un sistema estándar para integrar cualquier laboratorio virtual o remoto dentro del campus ATENEA, el proyecto iLabViR (Integración de Laboratorios Virtuales y Remotos) [13]. En este proyecto ha colaborado la UPC y la consultora Everis y han contado con el apoyo económico del Estado mediante el Plan Avanza 2. Actualmente el proyecto iLabViR se ha completado e integrado con éxito dentro de ATENEA.

El proyecto final de carrera que presentamos en esta memoria se ha desarrollado de forma independiente del proyecto iLabViR y se centra exclusivamente en la integración de los simuladores virtuales LAVICAD.

Este proyecto lo presenté en 2010 en el VI Congreso Internacional de Docencia Universitaria e Innovación (CIDUI) en unas jornadas reservadas al proyecto RIMA (Investigación e Innovación en Metodologías de Aprendizaje) de la UPC donde se expusieron diversas experiencias docentes basadas en laboratorios virtuales y remotos.

1.4 Estructura de la memoria

La estructura de la presente memoria se divide en los siguientes capítulos: En este primer capítulo se recoge la introducción al proyecto.

En el capítulo 2 se introduce la visión del proyecto en términos de motivación, objetivos atribuidos y el alcance que comprende.

En el capítulo 3 se realiza una descripción del proyecto introduciendo los conceptos principales en los que se basa. Se describe así Moodle como un paquete de software open-source para producir cursos basados en tecnología web permitiendo a estudiantes y alumnos la formación a distancia. Además, se define el concepto de módulo de actividades de Moodle como un complemento que se puede incorporar a una plataforma Moodle para proporcionar nuevas actividades que proponer a los alumnos. Y, por último, se introduce el concepto de JLAB como el módulo de actividades para integrar los simuladores LAVICAD en el portal Moodle de la UPC, Atenea.

El capítulo 4 incluye el análisis de los requerimientos del proyecto. Se distinguen los requerimientos funcionales que describen el comportamiento esperado del software, de los requisitos no funcionales que se centran en especificar condiciones concretas en el diseño y la implementación. Se ha utilizado la técnica de documentación de requisitos basada en Casos de Uso donde se muestra gráficamente la interacción entre los usuarios y el sistema.

En el capítulo 5 se describe el diseño elegido para cubrir todos los requisitos definidos en el análisis. El capítulo empieza con una descripción de las tecnologías implicadas en el proyecto. Por un lado, la arquitectura del portal Moodle donde debemos integrar nuestro proyecto. Por otro lado, se detalla la arquitectura de los simuladores LAVICAD que debemos conocer para adaptarlos a los requisitos definidos. Una vez conocemos el entorno técnico del proyecto se detallan los diseños elegidos para cubrir los requisitos definidos.

El capítulo 6 se centra en la implementación de cada una de las funcionalidades. Cada apartado engloba una funcionalidad y se detallan los pasos seguidos para su consecución.

El capítulo 7 explica el despliegue de la aplicación desarrollada para su publicación final. Se detalla el entorno requerido con sus requisitos mínimos y recomendados. También se incluye un manual de instalación y puesta en marcha.

En el capítulo 8 se muestra un ejemplo concreto de una práctica JLab dentro de una asignatura de Comunicaciones Digitales. Se detalla la simulación de un sistema QAM etapa por etapa con el objetivo de ver los resultados almacenados en Moodle.

El capítulo 9 explicamos los resultados obtenidos con la realización del proyecto.

Por último, el capítulo 10 incluye las referencias bibliográficas empleadas en el transcurso del proyecto.

2. Objetivos del proyecto

2.1 Introducción

Este capítulo está dedicado a presentar la descripción general del proyecto dividida en tres apartados: motivación, objetivos y alcance.

2.2 Motivación

Las tecnologías de la información y comunicación (TICs) y, en mayor medida, internet, están modificando el acceso a la información en todos los ámbitos de la vida. La educación superior también debe aprovechar estas nuevas herramientas para mejorar el proceso educativo y para adaptarse a las nuevas necesidades de las generaciones de estudiantes que han nacido en un entorno donde la información es global e instantánea.

Los estudios de ingeniería se plantean el uso de nuevas formas de trasladar la experimentación a sus estudiantes, una solución son los laboratorios remotos y virtuales. Con el objetivo de promocionar la investigación de nuevos métodos de aprendizaje aplicados a la enseñanza de la ingeniería, la UPC ha creado el grupo de investigación RIMA [2]. Dentro de este grupo se ha creado el grupo de investigación GILABVIR [3] que trabaja en el desarrollo de diversos laboratorios virtuales y remotos. Concretamente en el proyecto LAVICAD [12] se han desarrollado una serie de simuladores de varios sistemas de comunicación permitiendo al estudiante configurar y evaluar cada etapa del sistema.

El objetivo es integrar estos simuladores dentro del campus virtual de la UPC, ATENEA [5], que es una plataforma Moodle, para ofrecer a los alumnos de las diversas ingenierías la opción de realizar los experimentos a distancia y permitir al profesorado proponer experimentos y realizar su seguimiento desde el mismo portal Atenea.

Ya se ha realizado un proyecto que es capaz de enviar los resultados de las simulaciones a Moodle modificando el módulo JClic para ello [14]. Pero este sistema requiere hacer una instalación de JClic para cada simulador que se quiera usar. Además, almacena los resultados en una tabla de Moodle pero sólo indicando la persona y el simulador, no el curso y la actividad donde se ha realizado.

Con la idea de ampliar la funcionalidad del JClic modificado surge el presente proyecto, JLab. Un módulo de Moodle que permita:

- Gestionar un banco de simuladores compartido por todos los profesores.
- Añadir, modificar y eliminar los simuladores de forma sencilla.

- Grabar los resultados de las prácticas distinguiendo por curso, práctica y usuario.
- A los profesores indicar si quieren almacenar los resultados o no y entre qué fechas.
- A los profesores visualizar los resultados de las prácticas desde el mismo portal Moodle.
- Descargar en Excel los resultados de una práctica concreta.

2.3 Objetivos

El desarrollo del presente proyecto ha sido promovido por el Departamento de Teoría de la Señal y Comunicaciones (TSC) de la Escuela Técnica Superios de Telecomunicación de la UPC como continuación del proyecto LAVICAD en el que se desarrolló una plataforma online para la implementación de simuladores de sistemas de comunicación. El principal objetivo de este proyecto es integrar estos simuladores dentro del portal formativo Moodle de la UPC, ATENEA.

Teniendo en cuenta el funcionamiento de la plataforma Moodle se ha decidido realizar la integración implementando un módulo de actividades de Moodle, al que hemos denominado módulo JLAB. Un módulo es un complemento que se instala en un portal Moodle y que añade a éste un nuevo tipo de prácticas que los profesores podrán proponer a sus alumnos.

El módulo JLAB ha de permitir a los profesores proponer a sus alumnos prácticas de cualquier simulador LAVICAD dentro de una asignatura en Moodle donde los alumnos puedan ejecutarlo. Además, se pretende ampliar los recursos de evaluación de los profesores dándoles acceso a los resultados de las simulaciones realizadas, para ello se realizará una modificación del núcleo de los simuladores LAVICAD para que envíe los resultados al servidor de Moodle. JLAB almacenará los resultados recibidos y quedaran accesibles a los profesores desde el mismo portal.

2.3.1 Implementar Módulo de Actividades JLAB

El diseño del módulo JLAB tiene como objetivo principal la integración de los simuladores LAVICAD en Moodle ofreciendo a alumnos y profesores un nuevo tipo de actividad que realizar en sus cursos, así como la integración en Moodle de herramientas de evaluación de dichas prácticas.

Cada simulador LAVICAD es un fichero independiente que se puede publicar en una página web para su ejecución. El módulo JLAB ha de permitir gestionar los ficheros de los simuladores de manera que no haya duplicidad de ficheros, es decir, evitar que un profesor tenga que subir al servidor el fichero del simulador cada vez que quiera proponerlo en una práctica a sus alumnos. Para resolverlo el módulo incluirá un repositorio compartido de simuladores que los profesores podrán gestionar y que enlazará con los simuladores que se podrán ofrecer en las prácticas JLAB.

El módulo JLAB ha de almacenar en una base de datos los resultados de las simulaciones de los alumnos. Un simulador LAVICAD representa un sistema de comunicación por etapas, cada etapa tiene unos parámetros que el alumno deberá configurar hasta obtener la señal de salida solicitada por el profesor, esta generación de la señal de salida también genera unos resultados que serán los que se envíen al servidor. Además de los resultados de la simulación de la etapa se necesita almacenar la información contextual de la simulación: los resultados que se envían, los valores de cada resultado, la etapa a la que pertenecen dichos resultados, el simulador correspondiente, el alumno que lo envía, así como la actividad y curso a la que pertenece el simulador.

Una vez se dispone de la información de las prácticas almacenadas en la base de datos, JLAB ha de implementar una página donde los profesores puedan ver los resultados enviados por sus alumnos y descargarlos en un archivo en formato hoja de cálculo.

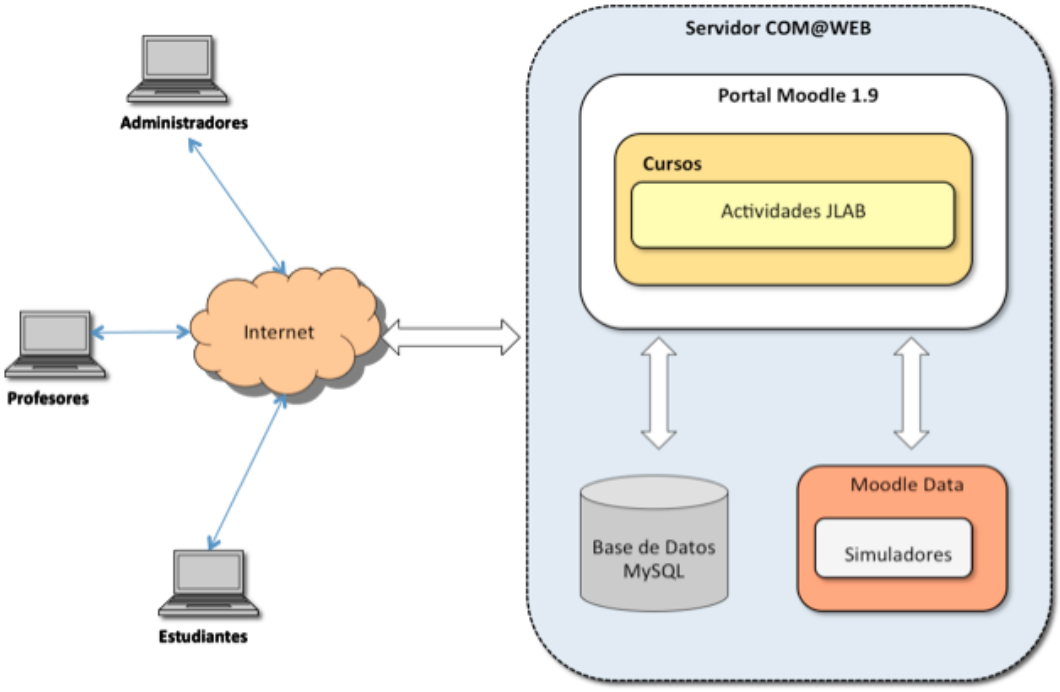


Figura 2 - Arquitectura del módulo JLAB.

La arquitectura del módulo JLab mostrado en la Figura 3 muestra que las actividades JLab se crean dentro de Cursos en el portal Moodle. También muestra que se comunica con la Base de datos MySQL de Moodle. Y, finalmente, que los simuladores están accesibles desde los cursos, pero almacenados fuera de éstos, en concreto en el espacio Moodle Data que ofrece Moodle para guardar datos adicionales.

2.3.2 Modificación Aplicativo Contenedor LAVICAD

La estructura base de la plataforma LAVICAD consiste en una aplicación o plataforma general única denominada aplicativo Contenedor, reprogramable y configurable para emular desde sistemas muy sencillos y básicos, hasta aplicaciones más complejas y cotidianas, como por ejemplo el sistema de comunicación sin cables (WiFi) o el de televisión digital terrestre (TDT).

Mediante este aplicativo Contenedor se han programado diferentes sistemas de comunicación que permiten observar y analizar el avance secuencial de las señales implicadas a través de diferentes etapas.

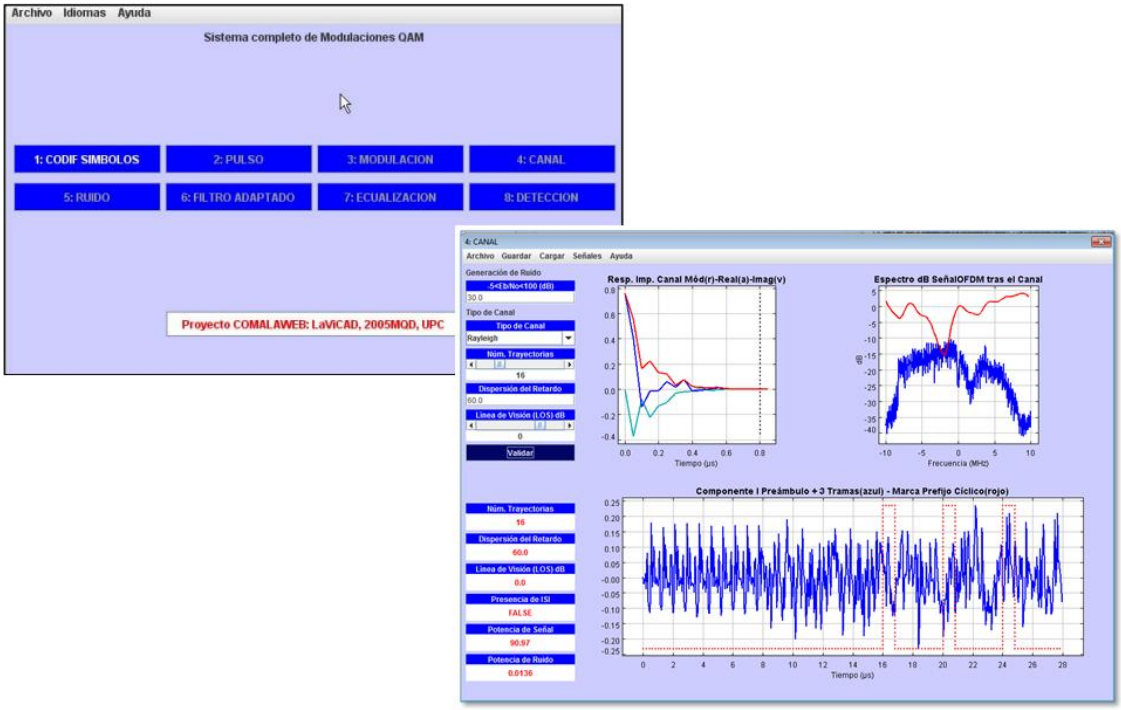


Figura 3 - Simulador QAM: Etapas y detalle de la etapa CANAL.

Actualmente se han desarrollado los siguientes sistemas de comunicación utilizando la plataforma LAVICAD:

1. Sistema completo de modulaciones QAM (Quadrature an Amplitude Modulation)
2. Sistema completo WiFi 802.11a/g basado en OFDM (Orthogonal Frequency Division Multiplexing)
3. Sistema completo de DVB-T (Digital Video Broadcasting)
4. Codificador convolucional y decodificador de Viterbi.
5. Sistema AMFM

Cada simulador consiste en un applet [10] de java que los usuarios se pueden descargar desde internet, por ejemplo, desde la plataforma formativa Atenea y ejecutarlo localmente en su ordenador. Un applet es una aplicación escrita en lenguaje JAVA que se puede incrustar en una página HTML, lo que facilita su distribución y ejecución.

A pesar de que los applets se pueden incluir en una página web, ambos están aislados el uno del otro no pudiendo compartir información, por lo que si se requiere algún tipo de intercambio de información ha de implementarse su envío y/o recepción.

El módulo JLAB debe recibir los resultados de las simulaciones realizadas por los alumnos, así como la información contextual de estos resultados como son el simulador que está ejecutando, el curso asociado y los datos del alumno que la realiza. Por lo que se deberá

modificar la base de los simuladores LAVICAD para que envíen esta información al servidor.

En la parte del servidor, se deberá implementar un receptor que esté siempre a la escucha y que procese la información recibida y la almacene en la base de datos.

La información contextual que deben enviar los simuladores LAVICAD al módulo JLAB deberá enviarse al simulador en el momento de la carga del applet, ya que una vez se ha descargado la aplicación, ésta ya no puede acceder a la información contextual en la que se está ejecutando. De modo que también se deberán modificar los simuladores para que puedan recibir esta información.

2.4 Alcance del proyecto

Dentro de este proyecto se engloba el diseño e implementación de una solución que cubra los objetivos definidos en el apartado anterior. A continuación, se detallan los puntos a desarrollar.

- Implementación del módulo de actividades de Moodle compatible con la versión de Moodle 1.9 o superior basándonos en la estructura de ficheros recomendada por la comunidad que ha desarrollado Moodle.

- Diseño, implementación y configuración de las tablas de la base de datos necesarias para cubrir las necesidades de almacenamiento indicadas en los objetivos, siguiendo las recomendaciones de uso de la base de datos de Moodle.

- Modificación del Aplicativo Contenedor para que se comunique con el módulo JLAB recibiendo y enviado los datos de las simulaciones y su información de contexto.

- Implementación dentro del módulo JLAB de un componente receptor de la información que envían los simuladores LAVICAD para procesarla y almacenarla en la base de datos.

3. Contexto del Proyecto

3.1 Introducción

En este capítulo se analizan más en profundidad los dos pilares en los que se asienta el proyecto: la plataforma de enseñanza online Moodle con la que está hecho el portal ATENEA y los simuladores virtuales de comunicaciones LAVICAD que queremos integrar en el portal ATENEA.

Los analizaremos desde tres puntos de vista. Primero veremos la capa funcional para entender los conceptos con los que trabajan. Después entraremos en la arquitectura que nos permitirá entender qué tendremos que modificar para conseguir los objetivos del proyecto. Y, finalmente, haremos una introducción a las tecnologías con las que se han implementado.

Todo ello con el objetivo de entender mejor la solución planteada en este proyecto: el módulo de actividades JLAB. La implementación de un módulo de actividades de Moodle que permite a profesores y alumnos interactuar con los simuladores LAVICAD dentro del portal ATENEA.

3.2 Plataforma de enseñanza a distancia MOODLE

En este apartado se explica el funcionamiento de la plataforma de enseñanza online Moodle. Se divide en tres áreas: la funcional donde se explican los conceptos con los que trabaja Moodle; la arquitectura para ver cómo se organiza el portal; y por último la capa tecnológica para entender las tecnologías que utiliza.

3.2.1 Descripción Funcional

A continuación veremos los conceptos básicos que se utilizan en Moodle: Curso, Actividad, Contexto, Capacidad, Permiso y Rol.

3.2.1.1 Curso

Moodle es un portal formado por cursos que son básicamente páginas que contienen el material de aprendizaje que los profesores quieren compartir con sus alumnos. Un profesor en un curso de Moodle puede seleccionar elementos de tres tipos diferentes los cuales ayudan al proceso de aprendizajes. Esos elementos son las Actividades, los Recursos y los Bloques [6].

3.2.1.2 Actividad

Una Actividad en Moodle es una funcionalidad que permite a los estudiantes aprender interactuando con los demás o con el profesor. Deben, por ejemplo, contribuir en un foro, subir un ejercicio, responder preguntas en un test o colaborar conjuntamente en una wiki. Las Actividades pueden ser puntuadas.

Un profesor puede añadir actividades activando la edición y añadiendo una actividad de entre las disponibles.

Moodle tiene un número de actividades consideradas estándar y también hay otras extras disponibles para que el administrador del portal las descargue e instale.

Además, existe la opción de crear nuestra propia actividad implementando un módulo de Actividades que luego se incluirá en el portal a modo de complemento .

3.2.1.4 Contexto

Un sitio Moodle está estructurado en niveles o Contextos jerárquicos [8] que son espacios en Moodle donde pueden asignarse Roles. Esta estructura permite administrar de forma clara y racional las capacidades y los permisos que los usuarios tienen en los diferentes recursos de Moodle.

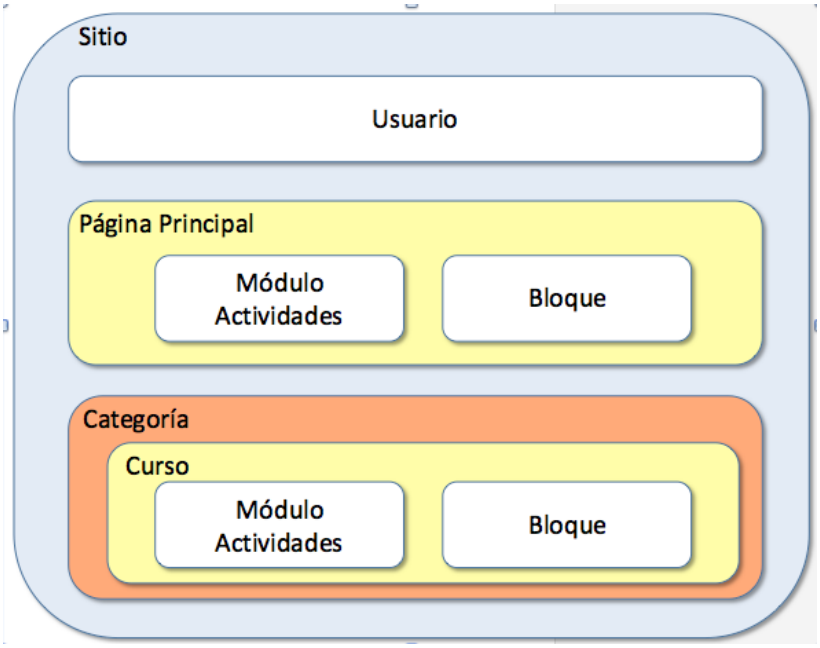


Figura 4: Contextos jerárquicos de Moodle.

Así, el contexto de mayor nivel jerárquico se denomina Sitio y corresponde al servidor Moodle. En este contexto el único que tiene definidas capacidades y activados los permisos de forma total es el Administrador del sitio.

A nivel de profesor o de alumno, el contexto de mayor jerarquía es la Página Principal, página inicial donde el usuario puede ver las listas de todos los cursos contenidos en el sitio, o sólo aquellos a los

que tiene acceso, según estén configurados los permisos en esta página.

Cada sitio Moodle contiene cursos que pueden estar organizados en agrupaciones llamadas Categorías de cursos. La Categoría es el siguiente nivel de contexto.

Más abajo en la jerarquía tenemos el Curso como tal, y finalmente, aparece un nuevo nivel dentro del curso, que son los Bloques, Recursos y Actividades que lo configuran.

Las modificaciones de permisos en un contexto se heredan en los contextos de nivel inferior, por lo que una modificación en un permiso, por ejemplo, de categoría de cursos, afecta a todos los cursos contenidos en esta categoría.

3.2.1.5 Capacidad

Una capacidad o habilidad es una posible acción unitaria que un usuario puede realizar sobre un elemento de Moodle; así, un usuario puede tener la capacidad de modificar su contraseña, de ver un recurso, de borrar este recurso, de añadir un recurso nuevo, responder a una entrada en un foro, etc. En Moodle hay más de 200 habilidades y en cada nueva versión van apareciendo nuevas, según crecen las funcionalidades del producto.

3.2.1.6 Permiso

Las capacidades se gestionan mediante Permisos [8]. Un permiso es la autorización que se da a un usuario para ejercer una capacidad que tiene asignada en un determinado contexto. Por ejemplo, un usuario con rol de alumno tiene la capacidad de responder a una entrada del foro, pero en un foro determinado podemos negarle esa capacidad modificando el permiso.

Cada rol tiene asignados unos permisos por omisión definidos por Moodle, pero estos permisos pueden ser modificados en un determinado contexto por aquellos usuarios que tengan la capacidad y el permiso de modificar el rol en ese contexto.

3.2.1.7 Rol

Podríamos definir un rol como un usuario tipo o un modelo de usuario que queda definido por un conjunto de capacidades y de permisos en los diferentes contextos de Moodle. Dicho de otra manera, cuando hablamos del rol de Profesor nos estamos refiriendo a todo aquello que puede hacer y le dejamos hacer a un usuario genérico que denominamos Profesor en cada uno de los contextos de un sitio Moodle.

Por lo tanto, para configurar lo qué puede hacer un nuevo usuario, basta con asignarle el rol correspondiente [8].

3.2.2 Arquitectura

A continuación se detalla la arquitectura de un portal Moodle.

3.2.2.1 Arquitectura global

La arquitectura de Moodle se compone de un servidor web con PHP habilitado, una base de datos relacional como pueden ser MySQL, PostgreSQL, Microsoft SQL Server u Oracle y por último un sistema de almacenamiento de ficheros donde dejar los archivos que se suban y se generen en el sistema, el denominado MoodleData.

Moodle está organizado en una aplicación central básica en la que se integran múltiples módulos que le aportan la funcionalidad específica necesaria en cada instalación. Moodle está diseñado para ser altamente extensible y altamente customizable sin modificar las librerías del núcleo. Esto permite asegurar el correcto funcionamiento de las futuras actualizaciones.

Los módulos en Moodle pueden ser de diferentes tipos, cada uno de ellos con una API (Application Programming Interface) específica que permite la comunicación entre el núcleo y el módulo. Los diferentes tipos de módulos disponibles en Moodle son:

- Actividades: cuestionarios, foros, etc.
- Recursos: enlaces, etc.
- Bloques
- Temas
- Paquetes de lenguajes
- Autenticación

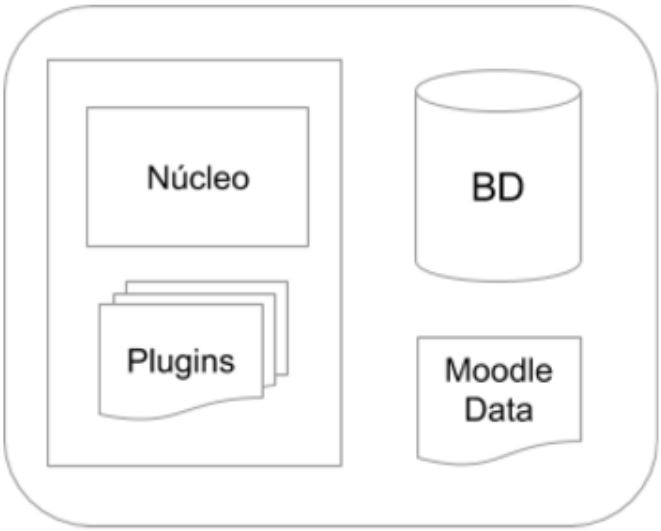


Figura 5: Arquitectura Portal Moodle.

El código fuente de Moodle está organizado en dos capas: la capa de presentación en HTML y la lógica de negocio en scripts PHP.

Físicamente, un módulo es una carpeta con los scripts PHP necesarios que se copia dentro de las carpetas de Moodle. Al seguir la estructura y la API de Moodle, el módulo se integrará perfectamente en el portal Moodle.

3.2.2.1 Módulos de Actividades

Un módulo de actividades [7] es un tipo de módulo de Moodle que le aporta la posibilidad de añadir prácticas dentro de un curso. La instalación básica de Moodle incorpora diversos módulos de actividades básicos como son cuestionarios y foros.

Moodle dispone, además, de una API para la creación de nuevos módulos de actividades y de una estructura de ficheros modelo para su implementación. Mediante la modificación de estos ficheros podemos dar la funcionalidad que necesitemos al módulo.

3.2.3 Capa Tecnológica

En este apartado veremos más en detalle las diferentes tecnologías utilizadas en un portal Moodle: PHP y MySQL.

3.2.3.1 Lenguaje de programación PHP

Moodle es una página web dinámica implementada en el lenguaje de programación PHP [9].

PHP (Hypertext Pre-processor) es un lenguaje de programación interpretado [9], es decir, que está diseñado para ser ejecutado

mediante un intérprete a diferencia de los lenguajes compilados en los que el código fuente se traduce a un ejecutable. PHP se usa principalmente para la interpretación del lado del servidor, pero actualmente puede ser utilizado desde línea de comandos o para la creación de aplicaciones de escritorio utilizando las librerías Qt o GTK+.

Puede ser desplegado en la mayoría de servidores web y en casi todos los sistemas operativos y plataformas sin costo alguno ya que está publicado como software libre.

Cuando un cliente hace una petición al servidor para que le envíe una página web, el servidor ejecuta el intérprete de PHP, éste procesa el script solicitado que generará el contenido de manera dinámica (por ejemplo, obteniendo información de una base de datos). El resultado es enviado por el intérprete al servidor, quien a su vez se lo envía al cliente en formato HTML para que lo interprete el navegador web.

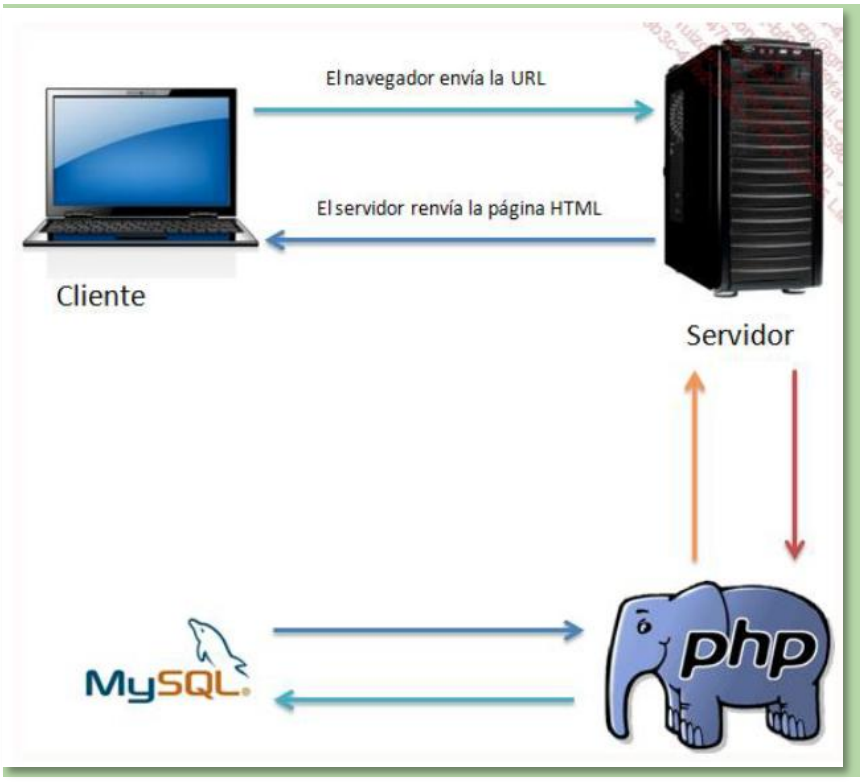


Figura 6: Arquitectura cliente servidor PHP.

3.2.3.2 Motor de base de datos relacional MySQL

Moodle utiliza una base de datos relacional [9] para almacenar toda la información del sitio.

Una base de datos se define formalmente como un conjunto de datos estructurados, fiables y homogéneos, organizados independientemente en máquina, accesibles en tiempo real, compartibles por usuarios concurrentes que tienen necesidades de información diferentes y no predecibles en el tiempo.

En una computadora existen diferentes formas de almacenar información. Esto da lugar a distintos modelos de organización de la base de datos: jerárquico, red, relacional y orientada objeto.

Las bases de datos relacionales están constituidas por una o más tablas que contienen la información ordenada de una forma organizada y que cumplen las siguientes leyes básicas:

- Generalmente, contendrán muchas tablas.
- Una tabla sólo puede contener un número fijo de campos.
- Los nombres de los campos de una tabla son distintos.
- Cada registro de la tabla es único.
- El orden de los registros y de los campos no está determinado.
- Para cada campo existe un conjunto de valores posible.

Diseño

El primer paso para crear un base de datos es planificar el tipo de información que se quiere almacenar en la misma, teniendo en cuenta dos aspectos: la información disponible y la información que necesitamos.

La planificación de la estructura de la base de datos, en particular de las tablas, es vital para la gestión efectiva de la misma. El diseño de la estructura de una tabla consiste en una descripción de cada uno de los campos que componen el registro y los valores o datos que contendrá cada uno de esos campos.

Los campos son los distintos tipos de datos que componen la tabla, por ejemplo: nombre, apellido, domicilio.

Los registros constituyen la información que va contenida en los campos de la tabla.

Lenguaje de Base de Datos SQL

Para manipular la información utilizamos un lenguaje relacional, actualmente se cuenta con dos lenguajes formales: el álgebra

relacional y el cálculo relacional. El álgebra relacional permite describir la forma de realizar una consulta, en cambio, el cálculo relacional sólo indica lo que se desea devolver.

El lenguaje más común para construir las consultas a bases de datos relacionales es SQL (Structured Query Language) [30], un estándar implementado por los principales motores o sistemas de gestión de bases de datos relacionales.

Existe software exclusivamente dedicado a tratar con bases de datos relacionales. Este software se conoce como SGBD (Sistema de Gestión de Base de Datos relacional) o RDBMS (Relational Database Management System) [9].

3.3 Simuladores LAVICAD

En este apartado veremos en detalle el funcionamiento de los simuladores LAVICAD. Empezaremos con una descripción funcional, continuaremos con la arquitectura y acabaremos con la capa tecnológica.

3.3.1 Descripción Funcional

Los simuladores LaViCAD (Laboratorio Virtual de Comunicaciones Analógicas y Digitales) consisten en applets que representan sistemas o subsistemas de comunicaciones permitiendo observar y analizar el avance secuencial de las señales implicadas a través de las diferentes etapas del sistema.

Además, ofrece una plataforma llamada Aplicativo Contenedor que incluye el interfaz gráfico, así como las librerías y funcionalidades a nivel genérico. Permitiendo al desarrollador de un nuevo sistema de comunicaciones preocuparse sólo de configurar el sistema y de implementar el procesado de señal.

Mediante LaViCAD se emulan sistemas y subsistemas de comunicaciones. Todo sistema de comunicaciones parte de una fuente de información, medible a través de una señal continua en tiempo o de una serie de datos. Esta señal se procesa a través de diferentes sistemas: transmisor, canal y receptor. El subsistema transmisor y el subsistema receptor se hallan formados a su vez por diferentes subsistemas funcionales o etapas cada uno de ellos realizando un procesado o transformación diferente sobre la señal. El canal, a su vez, también se puede modelar parcialmente con un sistema que provoque en la señal los mismos efectos que éste.

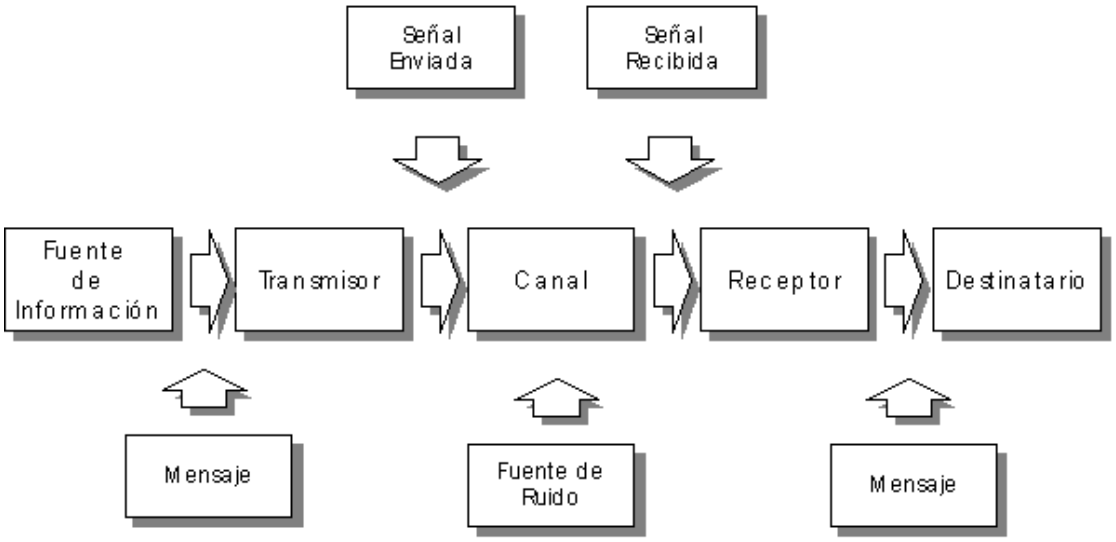


Figura 7: Esquema de comunicación.

LAVICAD emula el recorrido de esta señal a través de las diversas etapas. Cada etapa del emulador LAVICAD puede ser accionada por el usuario, siempre avanzando de forma secuencial. Con ello se abre una pantalla en la que se deben validar y/o cambiar los diferentes parámetros de configuración de etapa y apareciendo así toda una serie de resultados correspondientes a las señales de interés de las etapas en cuestión.

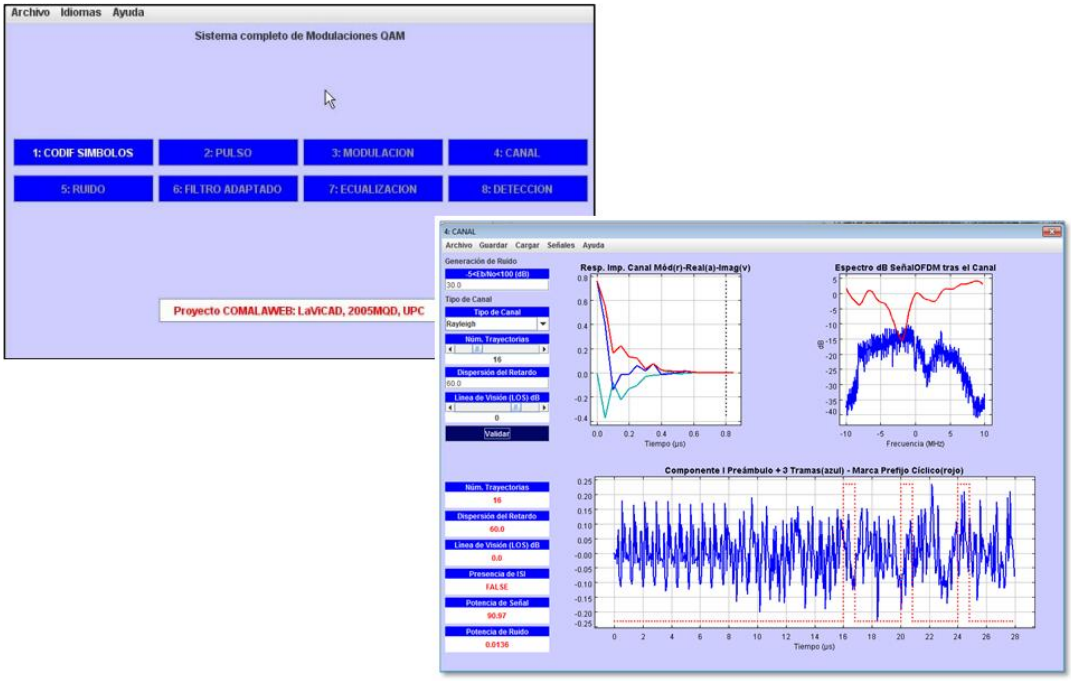


Figura 8: Etapas y detalle de una etapa de un sistema QAM implementado en un simulador LAVICAD.

3.3.2 Arquitectura

Cada sistema de comunicación a simular consiste en un fichero *sistema.jar* donde está definido e implementado el sistema y, por otro lado, un fichero común a todos los sistemas que se encarga de interpretar la configuración y construir gráficamente las etapas del sistema y ejecutar el procesado implementado para cada una de ellas en el sistema, es el llamado *Aplicativo Contenedor (contenedor.jar)*

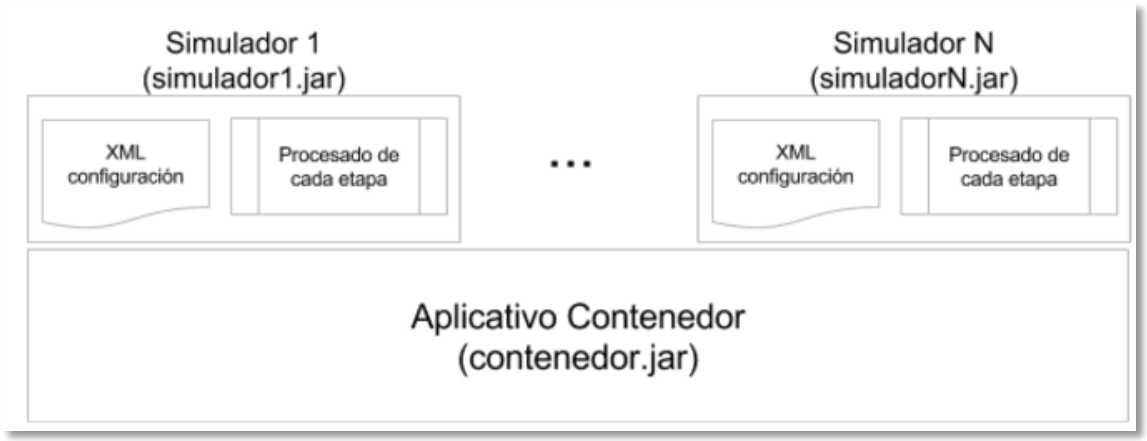


Figura 9: Arquitectura de un simulador LAVICAD.

Otras librerías necesarias para el funcionamiento de los simuladores y que también son comunes a todos ellos son:

- *plotmapplet.jar:* Librería utilizada para la representación gráfica de las señales.

- *SciLib.jar:* Librería gratuita científica de funciones matemáticas utilizada para realizar ciertas funciones de procesado, desarrollada por Ostfold College Lab.

3.3.2.1 Aplicativo Contenedor

El aplicativo Contenedor es un programa implementado en JAVA que proporciona una herramienta genérica para el desarrollo de cualquier simulador para su disposición libre en la red.

Es el núcleo de los simuladores LAVICAD ya que implementa la interfaz gráfica común a todos los simuladores, así como los métodos de configuración y ejecución, recogida de parámetros y generación de resultados, abstrayendo al programador del simulador de aquellos aspectos que se separan puramente de las operaciones de procesado de señal.

Y dispone de unos ficheros de configuración genéricos que permiten definir y configurar un determinado sistema mediante XML's.

- Número de etapas
- Parámetros de entrada de cada etapa y sus validaciones
- Parámetros de salida de cada etapa
- Señales que se representarán en cada etapa
- Señales que cada etapa permitirá guardar y/o cargar de un fichero
- Ayudas de cada etapa
- Idiomas disponibles en cada etapa

El programador del sistema de comunicación debe configurar mediante estos ficheros el sistema que quiere implementar.

Una vez configurado debe escribir el código JAVA necesario para realizar el procesado de cada etapa.

Al finalizar la configuración e implementación se compila todo en un fichero sistema.jar listo para ser embebido en una página web para su ejecución.

3.3.2.2 Simulador de comunicaciones

El simulador de comunicaciones consiste en la definición del sistema de comunicación utilizando el XML de configuración del aplicativo contenedor. Así como la implementación en java del procesado requerido por cada etapa del sistema. Todo ello se distribuye en un archivo de java con extensión .jar

3.3.3 Capa Tecnológica

En este apartado explicaremos el funcionamiento de los Java Applets que es la tecnología con la que están implementados los simuladores LAVICAD.

3.3.3.1 Java Applets

Java [10] es un lenguaje de programación de alto nivel orientado a objetos creado por James Gosling en 1995.

Se denomina lenguaje de alto nivel porque permite expresar los algoritmos de una manera adecuada a la capacidad cognitiva humana en lugar de adaptarse a la capacidad ejecutora de las máquinas.

La programación orientada a objetos (POO) [10] es un paradigma que utiliza objetos como elementos fundamentales en la construcción de la solución. Un objeto es una abstracción de algún hecho o ente del mundo real que tiene atributos que representan sus características o propiedades y métodos que representan su comportamiento o acciones que realizan. Todas las propiedades y métodos comunes a los objetos se encapsulan o se agrupan en clases. Una clase es una plantilla o un prototipo para crear objetos, por eso se dice que los objetos son instancias de clases.

La POO difiere de la programación estructurada tradicional [10], en que los datos y los procedimientos están separados y sin relación, ya que lo único que se busca es el procesamiento de unos datos de entrada para obtener otros de salida. La programación estructurada anima al programador a pensar sobre todo en términos de procedimientos o funciones, y en segundo lugar en las estructuras de datos que esos procedimientos manejan. En la programación estructurada sólo se escriben funciones que procesan datos. Los programadores que emplean POO, en cambio, primero definen objetos para luego enviarles mensajes solicitándoles que realicen sus métodos por sí mismos.

Una de las características diferenciadoras del lenguaje Java es que es multiplataforma, esto permite desarrollar aplicaciones que se pueden ejecutar en los diferentes sistemas operativos actuales: Unix, Linux, Windows y Mac. Para conseguirlo, el código fuente de Java se compila en lo que se denomina un *bytecode* [10] que es independiente del sistema operativo. Para conseguir el código máquina, que permite que el sistema operativo ejecute la aplicación, el sistema requiere de la Máquina Virtual de Java [10] que es específica para cada sistema operativo.

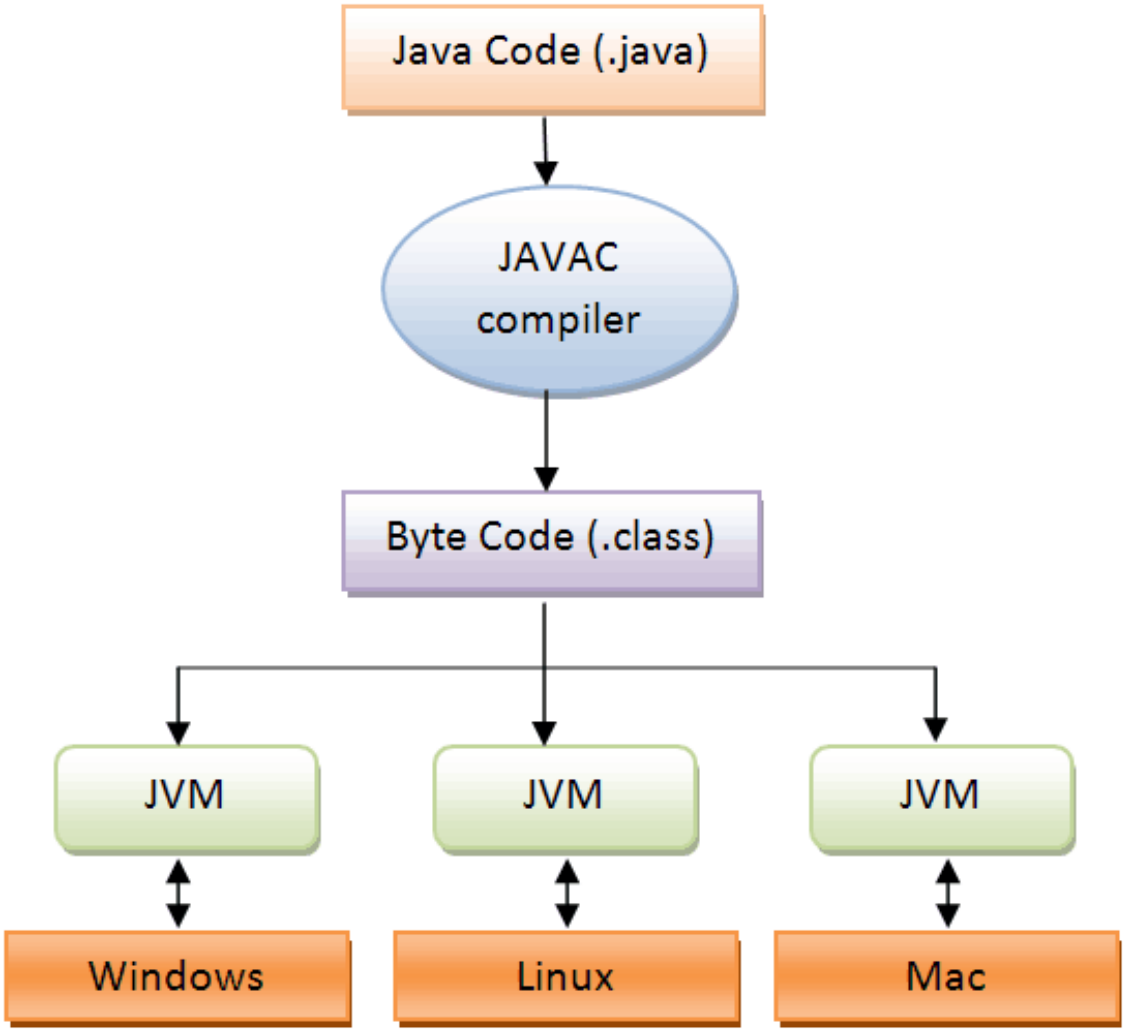


Figura 10: Fases de compilación del código Java.

Java puede ser usado para crear dos tipos de programas: aplicaciones independientes y applets [10].

Las aplicaciones independientes se comportan como cualquier otro programa escrito en cualquier lenguaje, como por ejemplo el navegador de Web HotJava, escrito íntegramente en Java.

Por su parte, las applets son pequeños programas que aparecen embebidos en las páginas Web, como aparecen los gráficos o el texto, pero con la capacidad de ejecutar acciones muy complejas, como animar imágenes, establecer conexiones de red, presentar menús y cuadros de diálogo para luego emprender acciones, etc.

Los simuladores LAVICAD son applets implementados en Java.

3.4 Conclusiones

En este apartado presentamos las conclusiones del capítulo. Propondremos una solución para integrar los Simuladores LAVICAD en Moodle. Y explicaremos la metodología utilizada para llevar a cabo el proyecto.

3.4.1 Solución propuesta

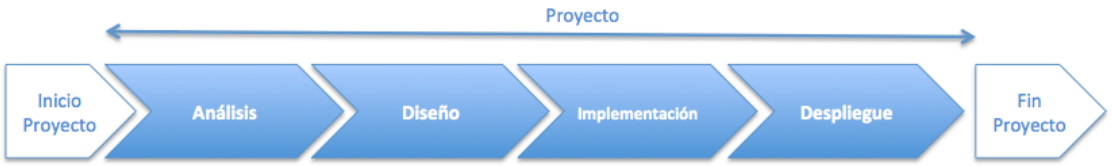
El objetivo de este proyecto es permitir ejecutar simuladores LAVICAD dentro del portal Moodle.

En este capítulo hemos visto, por un lado, que Moodle es un portal web que permite añadir funcionalidades extras al portal mediante la creación de módulos de actividades y que dispone de una base de datos relacional. Por otro lado, hemos visto que los simuladores LAVICAD son applets de java que se pueden ejecutar en una página web.

Por ello se ha decidido crear un módulo de actividades de Moodle que ejecute los simuladores LAVICAD y almacene los resultados de los alumnos en la base de datos de Moodle. A este módulo lo hemos llamado JLab, de Laboratorio en Java.

3.4.2 Metodología de desarrollo

Teniendo en cuenta que este proyecto consiste en desarrollar un programa informático, se ha seguido una metodología de desarrollo [11] con el fin de obtener un resultado de calidad y que cubra todos los objetivos definidos.



Los procesos que se realizan en todo desarrollo de un aplicativo son:

- **Análisis**: Proceso de definición del problema a resolver mediante la recogida y el análisis de los requisitos que debe cumplir el producto final.
- **Diseño**: Proceso de definir la arquitectura de la solución al problema planteado.
- **Implementación**: Proceso de construir una solución funcional al problema. Consiste en codificar la solución mediante un lenguaje de programación.
- **Despliegue**: Proceso de entrega de la solución final y puesta en marcha en un entorno de producción.

4. Análisis

4.1 Introducción

Este capítulo se centra en la fase de análisis del proyecto.

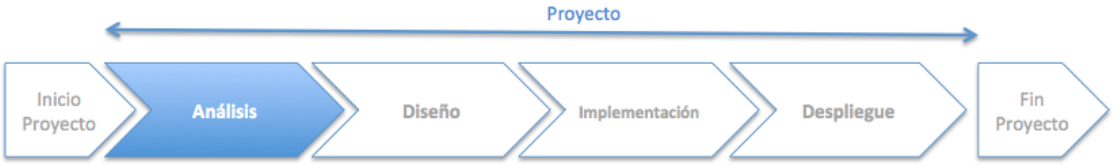


Figura 11: Fase de Análisis de un proyecto de software.

Se recogen los requisitos que debe cubrir el proyecto JLAB. Se distinguen los requisitos funcionales que describen el comportamiento esperado de la aplicación, de los requisitos no funciones que se centran en especificar condiciones concretas en el diseño y la implementación.

Se ha utilizado la técnica de documentación de requisitos basada en Casos de Uso [11] donde se muestra gráficamente la interacción entre los usuarios y el sistema.

4.2 Requisitos Funcionales

En este apartado se presentan los requisitos funcionales que deberán ser satisfechos por el sistema. Los requisitos funcionales pretenden dar una visión global de todas las funcionalidades que debe ofrecer el módulo JLAB.

4.2.1 REQ01: Gestionar Repositorio de Simuladores

Los simuladores LAVICAD, tal y como se explican en el Capítulo 3, se componen de dos ficheros java: un fichero común a todos los simuladores llamado Contenedor.jar; y un fichero específico para cada sistema de comunicación al que llamaremos sistema.jar donde se definen las etapas concretas de cada sistema de comunicación.

JLAB ha de incluir en sus ficheros el Contenedor.jar, pero ha de permitir a los profesores y administradores del portal gestionar y compartir los ficheros específicos de los sistemas de comunicación que se vayan a utilizar en las actividades JLAB.

Para ello JLAB ha de habilitar una carpeta dentro de Moodle donde dejar los ficheros, a la que llamaremos Repositorio de Simuladores. Los profesores y los Administradores del sitio Web han de tener acceso a esta carpeta entrando desde la web, ya que no se conectarán directamente al sistema de archivos del servidor.

Desde JLAB podrán añadir simuladores al Repositorio, modificarlos y eliminarlos del Repositorio.

Los simuladores han de poder asociarse a una actividad, por lo que las actividades JLAB deben permitir seleccionar un simulador entre los disponibles en el repositorio. Para tener un control de los simuladores disponibles, se ha de mantener en la Base de Datos del módulo JLAB un registro por cada simulador disponible en el Repositorio.

JLab debe ofrecer una página de Gestión del Repositorio donde ha de mostrar un listado de los simuladores disponibles indicando la siguiente información:

- Nombre del simulador.
- Descripción del simulador.
- Visibilidad (visible/oculto en la página de nueva actividad JLAB).
- Número de actividades que lo utilizan.
- Fecha de subida del simulador al repositorio.
- Nombre de la persona que ha subido el simulador al repositorio.

Además, la página de Gestión del Repositorio ha de incluir un formulario para cada gestión:

- Añadir nuevos simuladores.
- Modificar un simulador ya existente en el repositorio
- Eliminar simuladores del repositorio.

4.2.2 REQ02: Configurar la actividad: simulador, guardar y fechas de guardado

JLAB ha de permitir a los profesores proponer cualquiera de los simuladores LAVICAD que estén disponibles en el repositorio común de JLAB en una actividad de un curso. Para ello se ha de implementar una configuración específica de una actividad de Moodle para que permita al profesor seleccionar el simulador que se quiere ejecutar en la actividad.

Por lo que en la pantalla de configuración de una actividad de Moodle se han de añadir las siguientes opciones:

- Desplegable con todos los simuladores LAVICAD disponibles en un repositorio compartido por todos los profesores. El profesor deberá elegir uno de los simuladores, que será el que se ejecute cuando se entre en la actividad.
- Indicador para marcar si se desea almacenar en la base de datos los resultados de las simulaciones.
- En caso de desear almacenar los resultados se ha de poder indicar unas fechas de inicio y fin de grabación. Esto permitirá limitar la información guardada en base de datos y también limitar el tiempo de validez de las simulaciones sin tener que quitar el acceso al simulador. Fuera del periodo de validez el simulador continuará estando disponible para su ejecución, pero los resultados no servirán para la evaluación del curso.

4.2.3 REQ03: Carga dinámica de un simulador en una actividad de Moodle

El módulo JLAB ha de permitir la ejecución de múltiples simuladores LAVICAD. Será el profesor quien indique en cada actividad el simulador que se ha de ejecutar.

Por ello, en el momento en que un usuario entra en un curso de Moodle y en una actividad concreta, JLAB ha de poder obtener el simulador que ha de cargar y la ruta del fichero que ha de incrustar en la página de la actividad para que el usuario vea el applet del simulador en el portal Moodle.

4.2.4 REQ04: Guardar resultados de las simulaciones

El módulo JLAB ha de almacenar los resultados obtenidos en la simulación de cada etapa de los sistemas de comunicación que se ejecuten dentro de una actividad JLAB. Con el objetivo de poder consultar esta información, los resultados irán asociados a la información de contexto de esta ejecución como son el curso, la actividad, el usuario, el simulador y la etapa a la que corresponden los resultados.

Para ello, en el momento en el que un usuario (profesor o estudiante) solicita la ejecución de un simulador dentro de una actividad JLAB, el sistema ha de enviar al cliente el applet del simulador incluyendo la siguiente información:

- Identificador del curso donde se realiza la actividad.
- Identificador de la actividad en la que se ejecuta el simulador.
- Identificador del simulador que se está ejecutando.
- Identificador del usuario que solicita la ejecución del simulador.

El estudiante configurará los parámetros de cada etapa, ejecutará la simulación y en ese momento se han de enviar al servidor los resultados obtenidos junto con la información contextual recibida inicialmente. Se enviará al servidor la siguiente información:

- Información contextual: curso, actividad, simulador, usuario.
- Nombre de la etapa a la que pertenecen los resultados.
- Lista de los resultados en formato:
    - Nombre Resultado
    - Valor Resultado

El módulo JLAB recibirá la información y la almacenará en la base de datos.

4.2.5 REQ05: Mostrar resultados online

El sistema ha de guardar en la base de datos los resultados de las simulaciones realizadas por los alumnos en las actividades JLAB. También ha de permitir que un profesor pueda consultar estos resultados desde el mismo portal.

Por lo que en la página de una actividad se ha de mostrar la información de los resultados, pero sólo a los profesores. La información a mostrar ha de ser un listado donde se indique:

- El nombre del estudiante que ha realizado la simulación.
- El nombre del simulador ejecutado.
- El nombre de la etapa a la que pertenecen los resultados
- Lista de pares “nombre resultado” – “valor resultado”.

4.2.6 REQ06: Descargar resultados en una hoja de cálculo

JLAB ha de almacenar y mostrar a los profesores los resultados de las simulaciones realizadas por sus alumnos en el contexto de una actividad. Esta información se ha de poder descargar en formato hoja de cálculo desde la misma pantalla donde se visualizan los resultados en línea.

La información a incluir ha de ser la misma que se muestra por pantalla, por lo que será un listado con:

- El nombre del estudiante que ha realizado la simulación.
- El nombre del simulador ejecutado.
- El nombre de la etapa a la que pertenecen los resultados.

- Lista de pares “nombre resultado” – “valor resultado”.

4.2.7 REQ07: Configurar la IP del servidor para la recepción de resultados

JLAB ha de almacenar los resultados de las simulaciones que realizan los estudiantes en las actividades JLAB. Los simuladores LAVICAD están implementados con applets de JAVA que se pueden incrustar en una página web pero que no puede compartir información directamente con ésta.

Para permitir la comunicación entre el applet y el módulo JLAB, éste ha de implementar un complemento que envíe y reciba la información del Applet.

El applet necesita que se le indique la IP del servidor donde ha de enviar la información. Para evitar tener que modificar el código fuente cada vez que se quiere instalar el módulo JLAB en un servidor diferente, se ha de implementar en la página de configuración del módulo JLAB la especificación de la IP y su configuración en el applet.

El módulo detectará automáticamente la IP del servidor donde está alojado, pero ha de permitir al Administrador del Moodle especificar otra IP.

4.2.8 REQ08: Control de acceso

El módulo JLAB ha de controlar en cada página si el perfil del usuario conectado tiene privilegios para entrar a dicha página. En Moodle existen los perfiles Profesor, Estudiante y Administrador.

En JLAB los usuarios con perfil Profesor podrán realizar las siguientes acciones:

- Gestionar el Repositorio de Simuladores
- Crear nuevas actividades JLAB dentro de los cursos en los que tenga permisos de profesor.
- Ejecutar los simuladores LAVICAD dentro de una actividad JLAB de uno de sus cursos.
- Visualizar y descargar los resultados de las actividades de sus cursos.

El módulo ha de controlar que los alumnos solo puedan ejecutar los simuladores LAVICAD dentro de una actividad de un curso al que pertenezcan.

Los Administradores del sistema podrán realizar todas las acciones, por lo que no se controlará su acceso al módulo.

4.3 Requisitos No Funcionales

4.3.1 REQ09: Portabilidad: Módulo de Moodle 1.9 o superior

En el momento de la realización del presente proyecto la versión del Moodle del Portal ATENEA de la UPC es la versión 1.9. Por lo que el módulo JLAB se debe implementar de manera que se pueda instalar en un Moodle versión 1.9 o posterior.

4.3.2 REQ10: Compatibilidad con los simuladores de LAVICAD ya existentes

JLAB debe mantener la compatibilidad con los simuladores LAVICAD disponibles actualmente para su correcta ejecución. En el apartado 1.3 vimos la lista de simuladores implementados en el momento del desarrollo del presente proyecto.

4.4 Casos de Uso

Una vez recogidos los requisitos de la aplicación los hemos analizado y hemos identificado las diversas interacciones entre los usuarios y el sistema para cumplirlos. Para documentar dicha información hemos elegido el lenguaje de modelado UML (*Unified Modeling Language*) [11] que sirve para representar sistemas de software. Dentro del UML vamos a utilizar el diagrama de Casos de Uso que representa en un gráfico simple el conjunto de Casos de Uso y las interacciones con los usuarios. También detallaremos cada uno de los Casos de Uso.

4.4.1 Definiciones

UML (Lenguaje Unificado de Modelado)

El Lenguaje Unificado de Modelado (LUM o UML, por sus siglas en inglés, *Unified Modeling Language*) [11] es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio, funciones del sistema y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables.

Caso de Uso

Un caso de Uso [11] es una lista de pasos que definen las interacciones entre un usuario (en UML se conoce como Actor) y el sistema para conseguir un objetivo.

Diagrama de Casos de Uso

En el ámbito del UML, un diagrama de Caso de Uso es una representación gráfica simple de la interacción de los usuarios con el sistema, representando las especificaciones de los casos de uso.

El diagrama utiliza una serie de componentes para la representación del modelo.

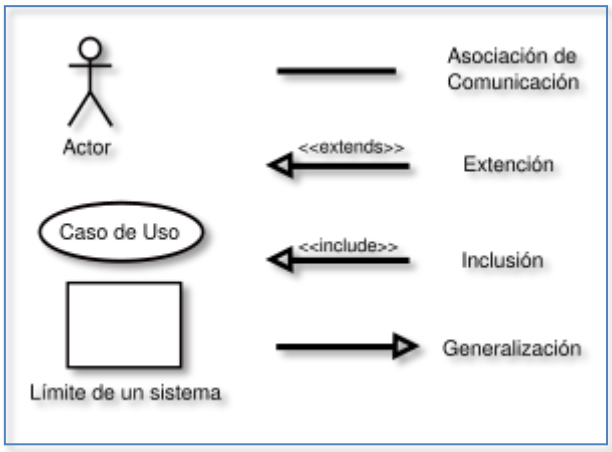


Figura 12: Componentes de un Diagrama de Casos de Uso

Actor: Se le llama actor a toda entidad externa al sistema que guarda una relación con éste y que le demanda una funcionalidad.

Caso de Uso: Es una operación específica que se realiza tras una orden de algún agente externo.

Límite de un sistema: Sirve para identificar las operaciones y actores que forman parte del sistema de las que son externas.

Relaciones:

- **Asociación de comunicación**: Indica invocación desde un actor o caso de uso a otra operación (caso de uso).
- **Generalización:**
  - **Extensión**: Indica que un caso de uso es similar a otro.
  - **Inclusión:** Utilizado cuando un caso de uso tiene características comunes a varios casos de uso.

4.4.2 Diagrama de Casos de Uso de JLAB

Una vez tenemos los requisitos definidos vamos a identificar los actores que intervienen en estos requisitos: el Administrador, el Profesor y el Alumno.

Además, debemos extraer de los requisitos las acciones que debe poder realizar cada uno de los Actores en el Sistema a implementar:

- El Administrador debe poder instalar JLAB en Moodle.
- El Administrador debe poder configurar ciertos parámetros de JLAB.
- El Profesor debe poder visualizar un listado de los simuladores disponibles en el Repositorio de Simuladores.
- El Profesor debe poder añadir un simulador al Repositorio de Simuladores.
- El Profesor debe poder modificar el fichero de un simulador en el Repositorio de Simuladores.
- El Profesor debe poder eliminar un simulador del Repositorio de Simuladores.
- El Profesor debe poder asociar un Simulador a una Actividad.
- El Profesor debe poder indicar en la actividad si se han de guardar o no los resultados de las simulaciones y entre qué fechas se guardarán.
- El profesor debe poder ver una lista de las actividades JLAB creadas.
- El profesor debe poder ver online los resultados de las actividades.
- El Profesor debe poder descargar en una hoja de cálculo los resultados de las simulaciones de una actividad.
- Un Estudiante debe poder ejecutar el simulador asociado a una actividad.
- Un Estudiante debe poder enviar los resultados de las simulaciones al servidor para que se guarden.

Una vez tenemos identificadas todas las interacciones entre los actores y el sistema, lo representamos mediante el diagrama de Casos de Uso.

Figura 13: Diagrama de Casos de Uso de JLab.

4.4.3 Especificaciones de Casos de Uso

En este apartado se detalla cada uno de los Casos de Uso identificados en el apartado anterior. Para ello se ha definido una plantilla de ficha de caso de uso donde se rellenará la siguiente información:

- **Título**: Nombre del caso de uso.

- **Descripción:** Explicación breve del caso de uso.
- **Actores:** Usuarios que intervienen en el caso de uso.
- **Pre-Condiciones:** Condiciones previas que se deben cumplir antes de la realización del caso de uso para su correcto funcionamiento.
- **Flujo Normal de Eventos:** Descripción paso a paso de las acciones de los usuarios y la respuesta del sistema para el caso de funcionamiento normal.
- **Flujos Alternativos de Eventos:** Descripción paso a paso para situaciones anómalas que también deben de estar previstas y controladas.
- **Post-Condiciones:** Condiciones que se deben cumplir después de la realización del caso de uso.

4.4.3.1 Caso de Uso: Instalar JLAB en Moodle

Caso de Uso:	**Instalar JLAB en Moodle**
Descripción: Permite instalar el módulo JLab en un portal Moodle	
Actores: Usuario con perfil Administrador	
Pre-condiciones: Estar logado en Moodle con perfil Administrador. El Moodle donde se instalará el JLab ha de ser versión 1.9 o superior.	
Flujo Normal de Eventos: 1. El usuario copia la carpeta del módulo JLab en la carpeta 'mod' del servidor Moodle. 2. El usuario entra en Moodle con perfil Administrador. 3. El usuario entra en la página de Administración del portal Moodle y pulsa 'Actualizar'. 4. Moodle detecta automáticamente el nuevo módulo y crea las tablas definidas en la base de datos y registra el nuevo tipo de actividad.	
Post-condiciones: Se ha instalado correctamente el módulo JLab en Moodle por lo que ya se pueden crear actividades JLab.	

4.4.3.2 Caso de Uso: Configurar IP del módulo JLAB

Caso de	**Configurar IP del módulo JLAB**

Uso:	
Descripción: Permite configurar la IP para la comunicación de los applets LAVICAD y el módulo JLab.	
Actores: Usuario con perfil Administrador	
Pre-condiciones: • Estar logado en Moodle con perfil Administrador.	
Flujo Normal de Eventos: 1. El usuario entra en Moodle con perfil Administrador. 2. El usuario entra en la página de Administración del módulo JLab. 3. El sistema cargará la página estándar de configuración de un módulo y, además, permitirá definir la IP del servidor donde se aloja el módulo. 4. El sistema detectará automáticamente la IP del servidor pero permitirá al usuario modificarla y guardarla.	
Post-condiciones: Se ha configurado la IP del servidor donde los applets enviarán los resultados de las simulaciones que validen los estudiantes.	

4.4.3.3 Caso de Uso: Añadir Simulador al Repositorio

Caso de Uso:	**Añadir Simulador al Repositorio**
Descripción: Añadir el fichero de un simulador LAVICAD al repositorio común del módulo JLAB.	
Actores: Usuario logado con perfil Profesor.	
Pre-condiciones: El usuario debe haberse logado en Moodle con perfil profesor.	
Flujo normal de Eventos: 1. El usuario entra en la configuración del módulo JLab y pulsa 'Repositorio de Simuladores'. 2. El sistema muestra la ventana de Gestión del Repositorio. 3. El usuario rellena el nombre del simulador, la descripción y selecciona el fichero de simulador a subir y pulsa 'Salvar Simulador'. 4. El sistema comprueba el fichero subido por el usuario y lo registra en la base de datos. 5. El sistema refresca la lista de simuladores para mostrar el nuevo simulador incorporado en el repositorio.	
Flujo Alternativo:	

4. El sistema comprueba el fichero del simulador y detecta los siguientes errores: • Tipo de fichero incorrecto: sólo se admiten ficheros con extensión ".jar". • Tamaño del fichero mayor al permitido por la configuración del sistema. • Fichero subido vacío. 5. El sistema informa al usuario del error producido para que pueda corregirlo y volver a guardar la información.
Post-condiciones: Se ha añadido el fichero en el repositorio de Simuladores y se ha registrado el simulador en la base de datos.

4.4.3.4 Caso de Uso: Modificar Simulador del Repositorio

Caso de Uso:	**Modificar Simulador del Repositorio**
Descripción: Modificar los datos de un simulador del repositorio y/o cambiar el fichero del simulador.	
Actores: Usuario logado con perfil Profesor.	
Precondiciones: El usuario debe haberse logado en Moodle con perfil Profesor.	
Flujo normal de Eventos: 1. El usuario entra en la configuración del módulo JLab y pulsa 'Repositorio de Simuladores'. 2. El usuario selecciona del desplegable el simulador que desea modificar. 3. El sistema carga los datos del simulador seleccionado: Nombre, Descripción, Visibilidad. 4. El usuario sólo modifica los datos del simulador y pulsa 'Guardar'. 5. El sistema guarda los cambios en la base de datos y refresca la tabla de simuladores para mostrar la información actualizada.	
Flujo Alternativo 1: 4. El usuario sube un nuevo fichero ".jar". 5. El sistema comprueba que el fichero es correcto y sobrescribe el anterior fichero.	
Flujo Alternativo 2: 4. El sistema comprueba el fichero del simulador y detecta los siguientes errores: • Tipo de fichero incorrecto: sólo se admiten ficheros con extensión .jar • Tamaño del fichero mayor al permitido por la	

configuración del sistema. • Fichero subido vacío. 5. El sistema informa al usuario del error producido para que pueda corregirlo y volver a guardar la información.
Post-condiciones: Se han modificado los datos y/o fichero del simulador y se han registrado los cambios en la base de datos.

4.4.3.5 Caso de Uso: Eliminar Simulador del Repositorio

Caso de Uso:	**Eliminar Simulador del Repositorio**
Descripción: Permite eliminar un simulador del repositorio.	
Actores: Usuario logado en Moodle con perfil Profesor.	
Precondiciones: Estar logado en Moodle con perfil Profesor.	
Flujo normal de Eventos: 1. El usuario entra en la configuración del módulo JLab y pulsa 'Repositorio de Simuladores'. 2. El sistema muestra la ventana de Gestión del Repositorio. 3. El usuario selecciona en el desplegable el simulador que desea eliminar. 4. El sistema comprueba si el simulador está asociado a alguna actividad. 5. El simulador no se está usando en ninguna actividad por lo que se puede eliminar el fichero ".jar" del repositorio y eliminar el registro de la base de datos.	
Post-condiciones: Simulador eliminado del repositorio y de la base de datos.	

4.4.3.6 Caso de Uso: Añadir Actividad JLAB a un curso

Nombre:	**Añadir Actividad JLab a un curso**
Descripción: Permite añadir una actividad JLab a un curso.	
Actores: Usuario logado en Moodle con perfil Profesor	
Precondiciones: Estar logado en Moodle. Ser profesor del curso donde se crea la actividad.	
Flujo Normal de Eventos: 1. El usuario selecciona 'Añadir Nueva Actividad JLAB'. 2. El sistema carga la página de configuración de una actividad JLAB. 3. El actor rellena los campos comunes a todos los tipos de	

actividades: Nombre, Descripción, Observaciones. 4. El actor rellena los campos específicos de una actividad JLAB: - Selecciona uno de los simuladores disponibles en el repositorio. - Marca si se han de guardar los resultados de las simulaciones y entre qué fechas. 5. El actor pulsa 'Guarda Cambios'. 6. El sistema comprueba que los datos introducidos sean correctos. 7. El sistema crea una nueva actividad en la base de datos y le asocia el simulador seleccionado.
Flujo Alternativo de Eventos: 6. El sistema detecta algún error, informa al usuario y le permite corregir los cambios antes de volver a guardar.
Post-condiciones: Se ha guardado la configuración de la actividad en la base de datos.

4.4.3.7 Caso de Uso: Modificar configuración Actividad JLAB

Nombre:	**Modificar configuración Actividad JLab**
Descripción: Modificar la configuración de una actividad JLab.	
Actores: Usuario logado en Moodle con perfil Profesor	
Precondiciones: Estar logado en Moodle con perfil profesor. Ser profesor del curso al que pertenece la actividad.	
Flujo Normal de Eventos: 1. El usuario entra en la actividad de un curso. 2. El sistema carga la página de configuración de la actividad seleccionada. 3. El usuario puede modificar los parámetros de la actividad: Nombre, Descripción, Observaciones, Simulador, Fechas de guardado. 4. El usuario pulsa 'Guardar Cambios'. 5. El sistema comprueba que los datos modificados sean correctos y guarda los cambios en la Base de Datos.	
Flujo Alternativo de Eventos: 5. El sistema detecta algún error en los datos modificados por lo que informa al usuario y le permite corregir los errores.	
Post-condiciones: Se han guardado en la base de datos los cambios en la configuración de la actividad.	

4.4.3.8 Caso de Uso: Eliminar Actividad JLAB

Nombre:	**Eliminar Actividad JLab**
Descripción: Permite eliminar una actividad ya creada.	
Actores: Usuario logado en Moodle con perfil Profesor.	
Precondiciones: Ser profesor del curso al que pertenece la actividad.	
Flujo Normal de Eventos: 1. El usuario entra en la actividad. 2. El usuario pulsa eliminar. 3. El sistema elimina la actividad y todos sus datos asociados, como son los resultados que se hayan enviado.	
Post-condiciones: La actividad ha sido eliminada.	

4.4.3.9 Caso de Uso: Cargar Actividad JLAB

Caso de Uso:	**Cargar actividad JLab**
Descripción: Cargar una actividad JLab ejecutando el simulador asociado.	
Actores: Usuario con perfil Estudiante, Profesor o Administrador.	
Precondiciones: Estar logado en Moodle con perfil Estudiante, Profesor o Administrador y pertenecer al curso de la actividad.	
Flujo Normal de Eventos: 1. El usuario entra en un curso de Moodle 2. El usuario pulsa en una actividad JLAB del curso. 3. El sistema consulta en la base de datos la ruta del fichero del simulador asociado a la actividad solicitada. 4. El sistema incrusta el fichero del simulador dentro de la página de la actividad 5. El navegador del usuario comprueba que el cliente permite la ejecución de applets de java y se ejecuta el simulador correctamente.	
Flujo Alternativo de Eventos: 5. El navegador del cliente no permite la ejecución de applets de java por lo que no se podrá ejecutar el simulador. Deberá refrescar la página y permitir el uso de java applets.	
Post-condiciones: Applet del simulador LAVICAD ejecutándose dentro de la página de la actividad JLAB.	

4.4.3.10 Caso de Uso: Guardar resultados de una simulación

Caso de Uso:	**Guardar resultados de una simulación**
Descripción: Recepción y guardado en base de datos de los resultados de las simulaciones enviadas por los alumnos.	
Actores: Estudiante	
Pre-Condiciones: El usuario ha de estar logado en Moodle con perfil estudiante. El usuario ha de estar dentro de una actividad JLab y ejecutando el simulador LAVICAD desde su navegador.	
Flujo Normal de Eventos: 1. El usuario configura los parámetros de una etapa del simulador y pulsa 'Validar'. 2. El applet de java envía al servidor los resultados y la información contextual: • Identificador del curso • Identificador de la actividad • Identificador del simulador • Nombre de la etapa • Nombre de los parámetros resultantes • Valores de los parámetros resultantes 3. Si el applet tiene bien configurada la IP del servidor, los resultados llegarán al módulo JLab. 4. El módulo JLab comprobará en la configuración de la actividad si ha de guardar los resultados y los almacenará en la base de datos. 5. Si ese usuario ya había enviado antes resultados de la misma actividad y etapa, se sobrescribirán, quedando almacenada siempre la última versión enviada.	
Flujo Alternativo de Eventos 1: 3. El applet no tiene bien configurada la IP del servidor por lo que los datos no llegarán al servidor.	
Flujo Alternativo de Eventos 2: 4. La configuración de la actividad indica que no se han de guardar los resultados de la simulación por tener la opción de guardado desactivada o por estar fuera del margen de fechas de validez.	
Post-condiciones: Resultados de la simulación guardados en la base de datos en caso de que la configuración así lo indique y la IP del servidor esté bien configurada en el módulo JLAB.	

4.4.3.11 Caso de Uso: Mostrar los resultados de una actividad JLAB

Caso de Uso:	**Mostrar los resultados de una actividad JLab**
Descripción: Permite visualizar en la página de una actividad una tabla con los resultados de las simulaciones enviados por los alumnos.	
Actores: Usuario con perfil Profesor.	
Pre-condiciones: El usuario ha de estar logado en Moodle y tener perfil profesor. Ser profesor del curso al que pertenece la actividad. La actividad ha de tener resultados almacenados en la base de datos.	
Flujo Normal de Eventos: 1. El usuario entra en una actividad JLAB y selecciona la opción 'Resultados'. 2. El sistema consulta en la base de datos los resultados almacenados para ese curso y actividad concreta. 3. El sistema muestra una tabla con los resultados y su información contextual: nombre del estudiante, nombre de la etapa, nombre del parámetro, valor del parámetro y fecha de envío del resultado.	
Post-condiciones: Se muestran los resultados de las simulaciones enviadas por los alumnos de una actividad.	

4.4.3.12 Caso de Uso: Descargar resultados de una actividad

Caso de Uso:	**Descargar resultados de una actividad**
Descripción: Permite descargar una hoja de cálculo con los resultados de una actividad enviados por los alumnos.	
Actores: Usuarios con perfil Profesor.	
Pre-condiciones: Ser profesor del curso al que pertenece la actividad. La actividad ha de tener resultados almacenados en la base de datos.	
Flujo Normal de Eventos: 1. El usuario entra en una actividad JLAB y selecciona la	

opción 'Resultados'. 2. El sistema muestra la lista de resultados de la simulación de cada etapa del simulador asociado a la actividad realizados por los estudiantes. 3. El usuario pulsa 'Descargar Resultados'. 4. El sistema genera un archivo en formato hoja de cálculo con los resultados que está viendo el usuario por pantalla y lo envía al navegador del cliente.
Post-condiciones: El profesor se ha descargado una hoja de cálculo con los resultados de las simulaciones enviadas por los alumnos.

4.5 Conclusiones

En este capítulo se han recogido los requisitos que se esperan del módulo JLAB, tanto funcionales como no funcionales.

De todos los requisitos definidos se han identificado los actores y las interacciones entre los actores y el sistema JLAB. Y se han representado mediante un diagrama de Casos de Uso.

Finalmente se ha especificado en detalle cada uno de los casos de uso.

Una vez se ha realizado este análisis detallado de lo que se espera del módulo JLAB, ya se puede proceder a la siguiente fase en un desarrollo de software, al diseño de cada uno de los casos de uso. Tal y como se verá en el próximo capítulo.

5. Diseño

5.1 Introducción

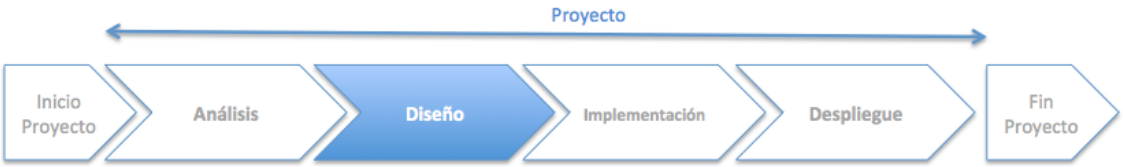


Figura 14: Fase de Diseño de un Proyecto.

Este capítulo tiene como objetivo plasmar un diseño de aplicación que cumpla con todos los requisitos definidos en el capítulo anterior.

En este capítulo se definirá la arquitectura técnica que se ha seleccionado para la implementación del módulo JLAB. También se definirá el diseño de las tablas que añadirá JLAB a la base de datos de Moodle. Y se definirán los cambios e implementaciones a realizar en los ficheros de un módulo de actividades de Moodle para cumplir con los requerimientos establecidos.

5.2 Diseño Arquitectura de JLAB

La arquitectura de la solución propuesta integra en un portal Moodle la gestión y ejecución de los simuladores LAVICAD, así como el almacenamiento de los resultados que obtienen los alumnos.

JLAB se ha diseñado a partir de un módulo de actividades de Moodle estándar al que se le han de realizar múltiples cambios para poder cubrir todos los requisitos definidos en el capítulo 5.

El diseño también incluye la adaptación del archivo base de los simuladores LAVICAD, el Aplicativo Contenedor, para que envíe al servidor Moodle los resultados de las simulaciones incluyendo la información contextual (alumno, actividad, curso) asociada. La modificación ha de permitir que funcionen correctamente los simuladores ya implementados.

La arquitectura de JLab es la arquitectura estándar de un módulo de actividades de Moodle. Por un lado, tenemos la estructura de ficheros del módulo que se integra dentro de los ficheros del portal Moodle. Estos ficheros son los que se deberán adaptar a los requisitos definidos.

Por otro lado, tenemos las tablas de JLAB que se añaden al esquema general de la base de datos de Moodle. Estas tablas han de almacenar los resultados de las simulaciones de los alumnos de manera que se puedan consultar fácilmente desde Moodle.

Y, finalmente, JLAB gestiona el repositorio de simuladores utilizando el lugar definido por Moodle para el almacenamiento de recursos propios de los módulos, el Moodle Data. El uso de este repositorio permite controlar y unificar las versiones de los simuladores disponibles en el módulo JLAB. Véase la Figura 3. Arquitectura del módulo JLab.

En los siguientes apartados se plantea la solución propuesta para resolver cada uno de los casos de uso identificados en el capítulo anterior utilizando la arquitectura elegida.

5.3 Diseño Base de Datos de JLab

En este apartado se define el diseño de las tablas de JLAB para almacenar la información necesaria para cubrir todos los requisitos. El diseño de las tablas ha de permitir, además, su perfecta integración en la base de datos de Moodle. Por lo que deberán seguir los criterios de Moodle.

5.3.1 Requisitos de las tablas según Moodle

Moodle obliga a que todos los módulos de actividades incluyan una tabla con el nombre del módulo. Además, todas las tablas deberán tener el prefijo "mdl_". Por lo que en nuestro caso deberemos añadir como mínimo una tabla llamada ***mdl_jlab*** para almacenar todas las instancias que se creen de nuestra actividad. Esta tabla debe incluir como mínimo tres campos:

1. **Id**: id de la instancia de JLab.
2. **Course**: Curso al que pertenece la instancia de la actividad JLab.
3. **Name:** Nombre de la instancia de JLab.

A partir de aquí se pueden añadir más campos y más tablas según la funcionalidad deseada. En el siguiente apartado se detalla el proceso de diseño para conseguir todas las tablas necesarias en JLab.

5.3.2 Diseño de las tablas de JLab

En este apartado vamos a detallar el proceso seguido para el diseño de las tablas del módulo JLab. Para ello partiremos del modelo Entidad-Relación que se deduce de los casos de uso definidos en el capítulo anterior y de ahí haremos la inferencia del modelo de datos.

5.3.2.1 Modelo Entidad-Relación

Vamos a ayudarnos del Modelo Entidad-Relación para diseñar las tablas. Un diagrama o modelo entidad-relación (a veces denominado por sus siglas en inglés, *E-R* "Entity Relationship", o del español *DER* "Diagrama de Entidad Relación") es una herramienta para el modelado de datos que permite representar las entidades relevantes

de un sistema de información, así como sus interrelaciones y propiedades. Sus elementos fundamentales son las entidades y las relaciones.

Una **entidad** caracteriza a un tipo de objeto, real o abstracto, del problema a modelar. Toda entidad tiene existencia propia, es distinguible del resto de las entidades, tiene nombre y posee **atributos** definidos en un dominio determinado. Una entidad es todo aquello de lo que se desea almacenar información. En el diagrama E-R las entidades se representan mediante **rectángulos** y los atributos mediante **elipses**.

Una **relación** es una asociación o relación matemática entre varias entidades. Las relaciones también se nombran. Se representan en el diagrama E-R mediante flechas y **rombos.** Cada entidad interviene en una relación con una determinada **cardinalidad**. La cardinalidad (número de instancias o elementos de una entidad que pueden asociarse a un elemento de la otra entidad relacionada) se representa mediante una pareja de datos, en minúsculas, de la forma *(cardinalidad mínima, cardinalidad máxima)*, asociada a cada uno de las entidades que intervienen en la relación. Son posibles las siguientes cardinalidades: *(0,1), (1,1), (0,n), (1,n), (m,n)*. También se informa de las cardinalidades máximas con las que intervienen las entidades en la relación.

El **tipo de relación** se define tomando los máximos de las cardinalidades que intervienen en la relación. Hay cuatro tipos posibles:

1. Una a una (1:1). En este tipo de relación, una vez fijado un elemento de una entidad se conoce la otra. Ejemplo: nación y capital.
2. Una a muchas (1:N). Ejemplo: cliente y pedidos.
3. Muchas a una (N:1). Simetría respecto al tipo anterior según el punto de vista de una u otra entidad.
4. Muchas a muchas (N:N). Ejemplo: productos y pedidos.

Toda entidad debe ser unívocamente identificada y distinguible mediante un conjunto de atributos (quizás un solo atributo) denominado **identificador** o **clave principal o primaria**. Puede haber varios posibles identificadores para una misma entidad, en cuyo caso se ha de escoger uno de ellos como identificador principal siendo el resto identificadores alternativos. Ejemplo: dni y número de seguridad social de una persona.

5.3.2.2 Diagrama Entidad-Relación del JLAB

En el caso del módulo JLAB, si analizamos los casos de uso podemos identificar las siguientes Entidades:

- Curso
- Administrador
- Profesor
- Alumno
- Actividad JLab
- Simulador
- Resultado de la simulación

La relación entre ellos se puede representar mediante el siguiente diagrama Entidad-Relación.

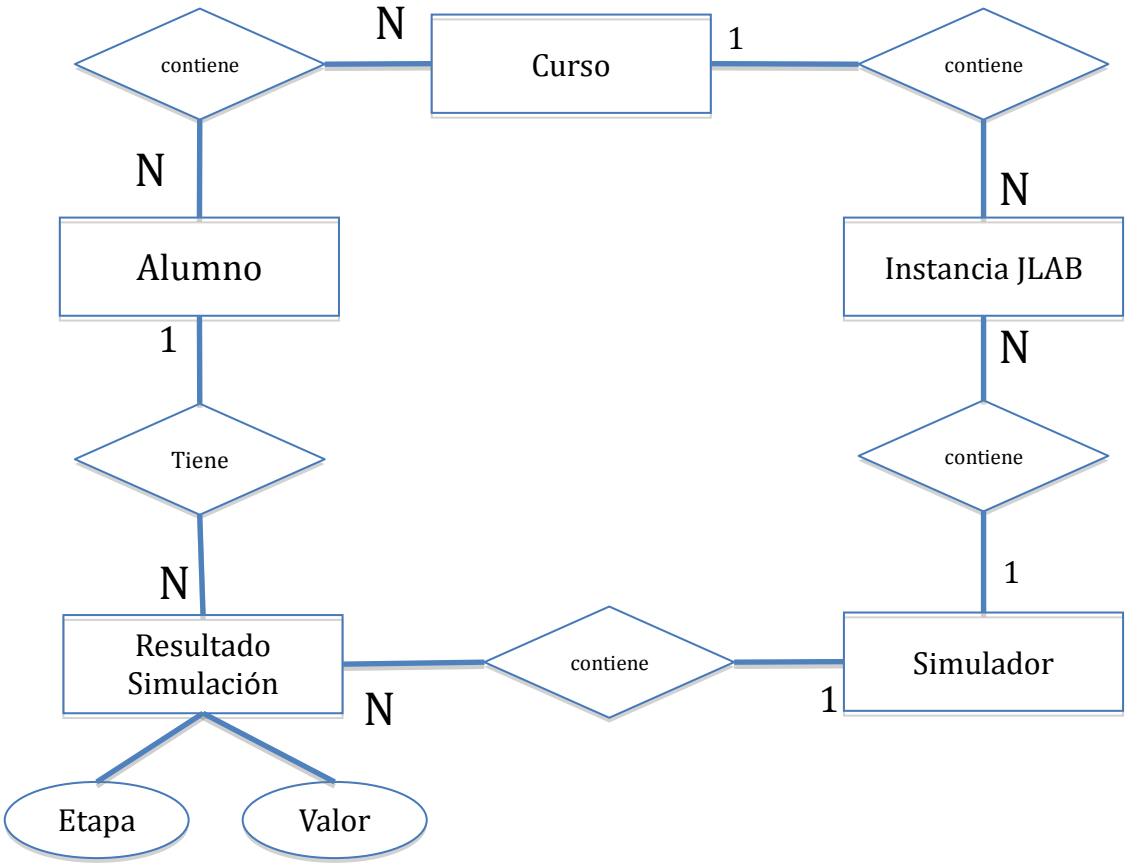


Figura 15: Diagrama Entidad-Relación de JLAB.

Las relaciones que se muestran en el diagrama ER son:

- Alumno – Curso (N:N) :

1 alumno puede estar en N cursos.

1 curso puede contener a N alumnos.

- Curso – Instancia JLab (1:N):

1 curso puede tener N instancias de actividades JLab.

1 instancia de JLab solo puede estar en 1 curso.

- Instancia JLab –Simulador (N:1):

1 Instancia sólo puede tener 1 simulador.

1 simulador puede estar en N instancias de JLab.

- Simulador – Resultados Simulación (1:N):

1 simulador puede tener N resultados.

1 resultados sólo puede pertenecer a 1 simulador.

- Alumno – Resultados Simulación (1:N):

1 Alumno puede tener N resultados de simulaciones.

1 Resultado de simulación sólo puede ser de 1 alumno.

- Cada Resultado tienen 2 atributos además del simulador y el alumno, que son la etapa a la que pertenece el resultado y el valor del resultado.

5.3.2.3 Modelado de Datos

Una vez tenemos el diagrama E-R podemos realizar la transformación al modelo de Tablas siguiendo las siguientes reglas básicas:

1. Cada **conjunto de entidades fuertes** se representa con una tabla, cuyas columnas corresponden a los atributos de las entidades. Se define como entidad fuerte aquella que puede ser identificada unívocamente sin participar en ninguna relación. Por ejemplo, la entidad cliente puede existir por sí misma.
2. Cada **conjunto de entidades débiles** se representa con una tabla, con una columna por cada atributo de las entidades más una columna por cada atributo de la clave primaria de la entidad fuerte de la cual el conjunto de entidades débil depende. Una entidad débil es que aquella que no puede existir sin participar en una relación. Por ejemplo, la entidad línea de factura no puede existir sin una factura.

3. Cada **relación "uno a varios"** se representa incluyendo en la tabla del extremo "varios'' las columnas de la clave primaria del extremo "uno"'.
4. Cada **relación "varios a varios"** y toda relación que involucre más de dos conjuntos de entidades se representa con una tabla, la cual tiene una columna por cada atributo de las claves primarias de los conjuntos de entidades a los que está ligada, más una columna por cada atributo descriptivo de la relación.

5.3.2.4 Modelo de Datos de JLAB

Siguiendo el criterio del Modelado de Datos definido en el apartado anterior vamos a identificar del diagrama E-R de JLAB cada uno de los elementos clave para la creación de las tablas.

En el diagrama E-R de JLab podemos identificar las siguientes Entidades Fuertes que se corresponderán cada una de ellas a una tabla independiente.

- Curso
- Alumno
- Profesor
- Administrador
- Actividad JLAB
- Simulador

Las Entidades Débiles son aquellas que necesitan de entidades fuertes para existir, por ejemplo, los Resultados de la Simulación, que relaciona al alumno con un simulador.

- Resultado Simulación

En la base de datos de Moodle ya existe una tabla para los Cursos y otra tabla para los Usuarios (como son Administradores, Profesores y Alumnos) a las que tendrá acceso el módulo JLAB, por lo que no necesitamos incluirlas en nuestro diseño. El resto de tablas sí las tendremos que añadir.

Además, Moodle, indica en sus especificaciones que los nombres de las tablas han de seguir un patrón, han de comenzar por "mdl_<nombre_del_módulo>". Y, como mínimo, ha de incluir una tabla que almacene las instancias que se vayan creando del módulo, llamada "mdl_<nombre_del_módulo>".

De manera que las tablas a añadir son:

mdl_jlab		
id	BIGINT(10)	Identificador único de una instancia de jlab.
name	VARCHAR(45)	Nombre de la instancia o práctica.
course	BIGINT(10)	Identificador del curso al que pertenece la práctica.

simulator	BIGINT(10)	Identificador del simulador que incluye la práctica. Coincide con el nombre de la carpeta dentro de simuladores que contiene los archivos de dicho simulador.
rec	TINYINT	Indica si se han de grabar en base de datos los resultados que obtienen los usuarios al utilizar el simulador.
rec_start	DATETIME	Fecha y hora en la que se empiezan a grabar los resultados que obtienen los alumnos.
rec_end	DATETIME	Fecha y hora en la que se dejan de grabar los resultados.
Description	TEXT	Descripción de la práctica.

mdl_jlab_simulators		
id	BIGINT	Identificador único del simulador. Coincide con la carpeta que contiene los archivos del applet: /jlab/simuladores/<id>
name	VARCHAR(45)	Nombre del simulador.
description	VARHCAR(45)	Descripción del simulador.
time_created	DATETIME	Fecha de inserción del simulador en el Repositorio de Simuladores de JLab.
time_modified	DATETIME	Fecha de modificación del simulador.
created_by	BIGINT(10)	Identificador del usuario que ha insertado el simulador en el Repositorio de Simuladores.
modified_by	BIGINT(10)	Identificador del usuario que ha modificado la información del simulador.
Visible	BIGINT(10)	Indica si el simulador se ha mostrar como opción en la creación de nuevas prácticas o no. 1;0.

mdl_jlab_results		
id	BIGINT	Identificador del resultado
jlab	BIGINT	Identificador de la práctica a la que pertenece el resultado.
user	BIGINT	Identificador del usuario que ha obtenido los resultados.
etapa	VARHCAR(200)	Nombre de la etapa del simulador al que pertence el resultado.
valor1	VARCHAR(200)	Resultado 1 con el formato: <nombre_resultado>:<valor_resultado>
valor2	VARCHAR(200)	Resultado 2 con el formato: <nombre_resultado>:<valor_resultado>
Valor3	VARCHAR(200)	Resultado 3 con el formato: <nombre_resultado>:<valor_resultado>

Valor4	VARCHAR(200)	Resultado 4 con el formato: <nombre_resultado>:<valor_resultado>
Valor5	VARCHAR(200)	Resultado 5 con el formato: <nombre_resultado>:<valor_resultado>

Una vez definidas las tablas y sus relaciones el esquema de base de datos es:

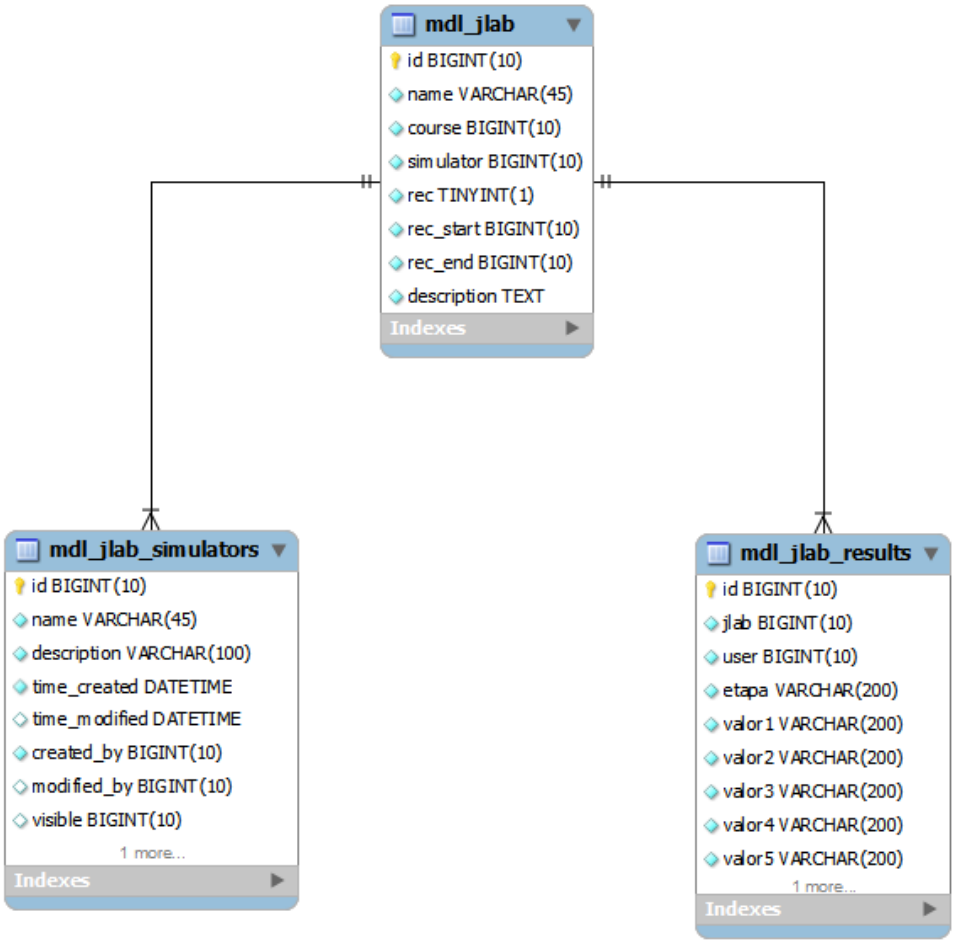


Figura 16: Esquema relacional de las tablas propias del módulo JLAB.

5.4 Diseño de los Casos de Uso

5.4.1 Diseño Instalación JLAB en Moodle

JLAB es un módulo de actividades de Moodle estándar, por lo que la instalación se realiza copiando una carpeta llamada igual que el módulo, es decir, jlab, que contenga todos los ficheros de un módulo modificados según la funcionalidad que se desee ofrecer, en el directorio del portal Moodel /mod.

Por otro lado, Moodle crea las tablas asociadas al módulo durante la instalación. Para ello lee el archivo incluido en el módulo db/install.xml. En este fichero deberemos dejar especificadas las tablas del módulo. En el próximo capítulo veremos el contenido exacto del fichero.

Una vez copiada la carpeta del módulo en el servidor de Moodle, solo será necesario entrar en el portal Moodle como Administrador, en Administración del Sitio > Notificaciones, y Moodle instalará el módulo y creará las tablas.

5.4.2 Diseño Configuración IP del módulo JLAB

Todo módulo de actividades de Moodle incluye una página de configuración.

Modificaremos esta página para incluir un campo donde el Administrador pueda indicar la IP del servidor donde se aloja Moodle. Esta IP se utilizará para comunicar los simuladores con el servidor para enviar los resultados.

También se mostrará en esta página la ruta física del Repositorio de Simuladores, ya que puede cambiar según la versión de Moodle y según la instalación concreta.

5.4.3 Diseño Repositorio de Simuladores

En este apartado plasmaremos el diseño de tres casos de uso relacionados con la gestión del Repositorio de Simuladores:

- Añadir Simulador al Repositorio
- Modificar Simulador del Repositorio
- Eliminar Simulador del Repositorio

El principal objetivo del Repositorio de Simuladores de JLab es ofrecer a los profesores una zona dentro de Moodle donde poder almacenar los ficheros de los simuladores que van a utilizar en las actividades de los cursos para poder compartirlos entre todos los profesores. Además, ha de ofrecer una página accesible por los profesores donde poder visualizar los simuladores disponibles, así como subir nuevos simuladores, modificar los ficheros de simuladores ya existentes y poder eliminarlos.

Para almacenar los ficheros vamos a utilizar la carpeta MoodelData que ofrece Moodle para subir ficheros desde Moodle. En la carpeta MoodleData vamos a crear una estructura de carpetas para almacenar los simuladores y vamos a utilizar los identificadores de los simuladores en la base de datos para nombrar las carpetas de cada simulador. La estructura de carpetas es la que se indica a continuación:

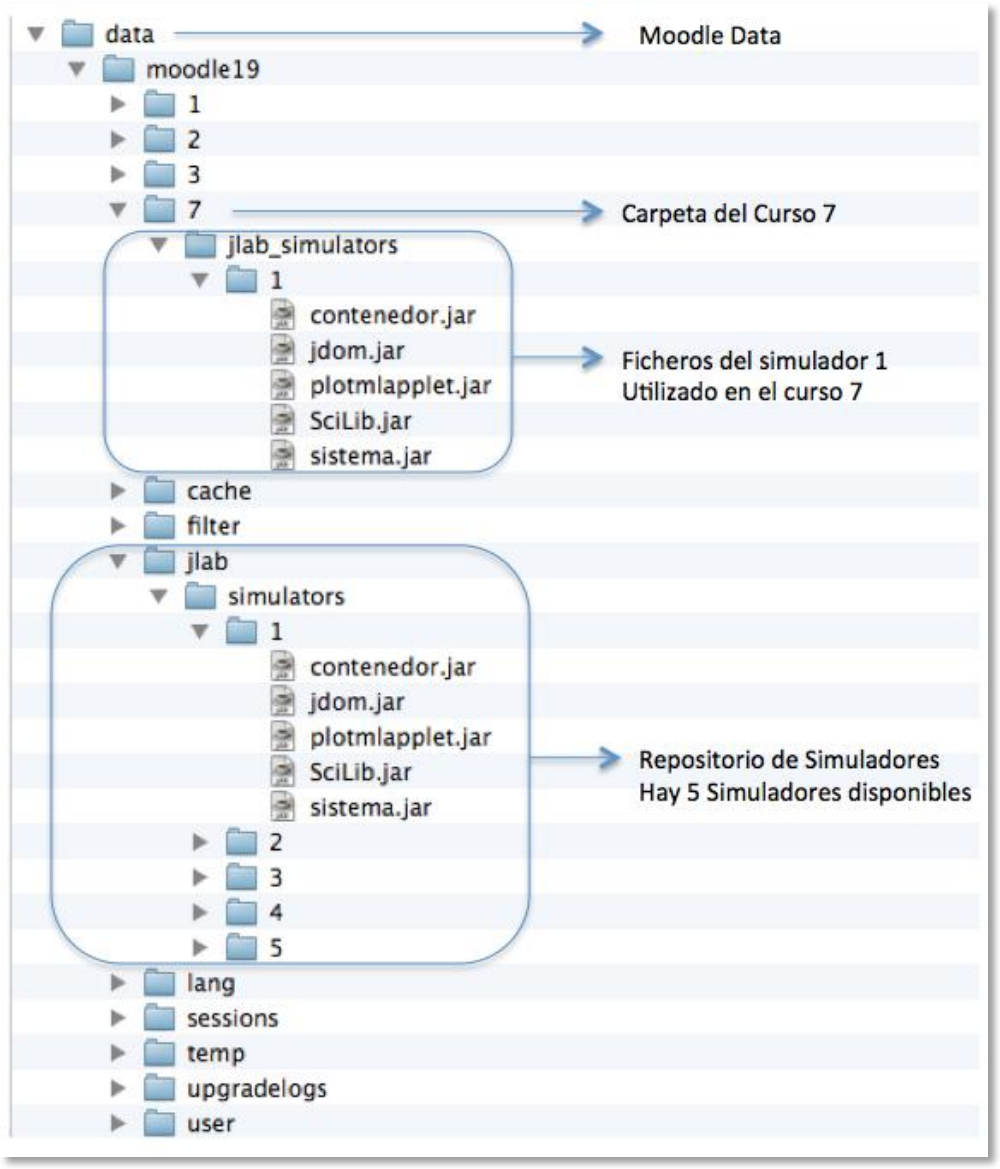


Figura 17: Ejemplo de estructura de carpetas de los simuladores en Moodle Data. Detalle del Repositorio y los simuladores de un curso.

Como se ha comentado anteriormente, un simulador LAVICAD consta de varios ficheros comunes a todos los simuladores y un fichero específico para cada simulador. JLab se ha diseñado para que el profesor solo tenga que subir el fichero específico del simulador, al que denominaremos *Sistema.jar*, el resto de ficheros los copiará automáticamente el módulo JLab de una carpeta incluida en la estructura raíz del propio módulo JLAB.

Además de gestionar los ficheros de los simuladores también se registrarán los simuladores en la base de datos. De modo que cada simulador que se suba al repositorio se almacenará en la base de

datos. Tenerlos en la base de datos permite a JLab conocer los simuladores disponibles tanto para listarlos en la página de gestión del Repositorio como en el momento de seleccionar un simulador al crear una actividad JLab.

En la versión 1.9 de Moodle utilizada en el desarrollo de este proyecto existe una limitación que tenemos que solventar. Moodle sólo permite a los usuarios de los cursos acceder al contenido de MoodleData ubicado bajo la estructura de carpetas *<MoodleData>/<Id_Curso>.*

Por lo que un usuario de una actividad JLab no podrá ejecutar directamente un simulador ubicado en nuestro Repositorio Principal de simuladores. Para que sean accesibles, JLab deberá realizar una copia de los simuladores desde el Repositorio Principal al Repositorio del Curso. La estructura de carpetas del Repositorio JLab del Curso es *<MoodleData>/<id_Curso>/<jlab>/<id_simulador>.* En los repositorios de cada curso solo se van a copiar los simuladores propuestos en ese curso, y sólo se copiarán una única vez.

El mantenimiento del Repositorio del Curso lo realiza JLab de forma transparente al usuario.

5.4.4 Diseño Gestión Actividades JLAB

En este apartado se incluye el diseño de los casos de uso:

- Añadir Actividad JLAB a un curso
- Modificar configuración Actividad JLAB
- Eliminar Actividad JLAB

Toda actividad de Moodle dispone de una página de creación donde se configuran los parámetros básicos de la actividad como son el nombre, una descripción y los permisos de acceso.

A la configuración básica debemos añadirle los parámetros propios del módulo JLAB. Se deberá poder seleccionar un simulador de los disponibles en el repositorio de simuladores, sin necesidad de tener que subir ningún fichero. El simulador seleccionado será el que se muestre a los estudiantes cuando entren en la actividad.

Además, se deberá poder indicar si se desea guardar los resultados de las simulaciones en la base de datos o no. Tal y como se ha explicado en el capítulo 3, los simuladores LAVICAD están formados por etapas con parámetros de entrada y salida. Los estudiantes deben definir los parámetros de entrada para conseguir la señal de salida solicitada en el ejercicio y los valores de los parámetros de salida. Éstos resultados serán los que se guarden o no en la base de datos según la configuración de la actividad.

En caso de desear guardar los resultados se ha de poder indicar entre qué fechas se guardarán los datos. Fuera de estas fechas se podrá continuar ejecutando el simulador, pero no se almacenarán los datos.

Esta configuración se almacenará en la base de datos de manera que se podrá consultar en el momento de la ejecución.

Para eliminar una instancia de una actividad JLab de un curso se ha de proceder como con cualquier otra actividad, se ha de entrar en el curso en modo edición y en el listado de actividades pulsar eliminar.

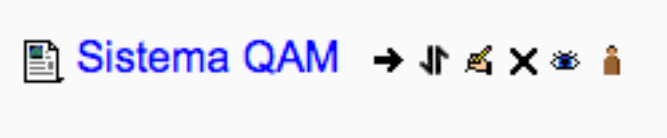


Figura 18: Detalle de una actividad JLab donde se muestran las acciones disponibles.

Al borrar la instancia de la actividad, también se deberán borrar de la base de datos los resultados almacenados de dicha actividad, así como todos los ficheros del simulador copiados en la carpeta del curso en MoodleData.

5.4.5 Diseño Carga Actividad JLAB

El principal objetivo del simulador JLab es permitir la ejecución de cualquier simulador Lavicad dentro de los cursos de Moodle. Para ello hemos creado un módulo de actividades que permite proponer a los alumnos de un curso la ejecución de un simulador en el contexto de una actividad. De modo que al acceder a la página principal de una actividad JLab muestre al usuario el simulador configurado previamente por el profesor.

La configuración del profesor se almacena en la base de datos. Datos como el simulador seleccionado y la forma de grabado de la información son accesibles por el módulo en cualquier momento.

La ejecución de un applet de java dentro de una página web requiere incluir un código donde se indica la ruta física del fichero .jar a ejecutar. JLab ha de consultar en la base de datos el simulador que le corresponde a la actividad y modificar el código de carga del applet dinámicamente para que se le cargue al usuario el simulador correcto.

5.4.6 Diseño Guardar Resultados del Simulador

JLab tiene como requisito poder almacenar los resultados de las simulaciones en la base de datos de Moodle. Los resultados se tendrán que almacenar indicando el contexto de estos resultados, es

decir, el curso de Moodle al que pertenecen, la actividad asociada y el usuario que los envía, así como la etapa concreta del simulador. Por lo que el simulador tendrá que enviar toda esta información al servidor.

Debido a que los simuladores son applets de Java que se ejecutan en el navegador del usuario cliente de forma aislada de la web que lo aloja, éstos no pueden consultar la información de contexto durante la ejecución.

La información contextual se tendrá que incorporar al simulador antes de que el usuario se descargue el applet. Para ello disponemos del fichero contenedor.jar que, como ya hemos comentado, es el núcleo común de todos los simuladores LAVICAD. Vamos a modificar el contenedor para que almacene la información contextual.

También vamos a modificar el contenedor para que, al ejecutar una simulación, envíe automáticamente los resultados y su contexto al servidor en un formato entendible por el servidor. En el servidor tendremos un servicio que estará a la espera de recibir esta información en cualquier momento y que se encargará de interpretarla, analizarla y almacenarla en la base de datos si la configuración así lo indica.

Por lo tanto, para poder almacenar los resultados en la base de datos vamos a modificar el contenedor de los simuladores para que almacene la información de contexto y luego la envíe junto con los resultados al servidor Moodle.

5.4.7 Diseño Visualización de resultados

Este apartado incluye el diseño de los casos de uso:

- Mostar los resultados de una actividad JLAB
- Descargar resultados de una actividad

Uno de los principales objetivos del módulo JLab es ofrecer a los profesores una herramienta ágil para visualizar los resultados que van obteniendo los alumnos en las simulaciones propuestas en los cursos.

Se modificarán los simuladores para que envíen los resultados al servidor, éste se encarga de almacenarlos en la base de datos. Ahora sólo falta mostrar los resultados de forma ordenada.

Para ello vamos a utilizar la página principal una actividad de Moodle. En esta página ya mostramos el simulador para su ejecución, pero vamos a incorporar un sistema de pestañas para permitir a los profesores visualizar además los resultados que han enviado los alumnos de las ejecuciones del simulador de esta actividad.

En la pestaña de resultados incluiremos una tabla donde se mostrará el último resultado enviado por cada alumno indicando la siguiente información:

- Nombre y apellidos del alumno.
- Etapa del simulador al que corresponden los resultados.
- Nombre del resultado.
- Valor del resultado para el parámetro indicado.

Los datos mostrados en la tabla también se podrán descargar en una hoja de cálculo, por lo que deberemos implementar el proceso que lo permita.

5.5 Conclusiones

En este capítulo hemos definido cómo se ha implementado cada uno de los casos de uso definidos en el capítulo 5.

En el siguiente capítulo detallaremos la implementación de cada uno de los diseños expuestos.

6. Implementación

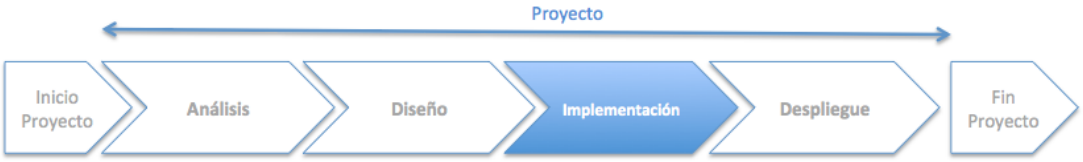


Figura 19: Fase de Implementación de un proyecto de software.

En este capítulo se va a explicar cómo se construye el módulo JLab y se explican las implementaciones y modificaciones realizadas para cubrir cada uno de los casos de uso requeridos siguiendo las directrices de diseño del capítulo anterior.

6.1 Configuración del módulo

JLab es un módulo de actividades de Moodle [7]. Tal y como hemos visto en el capítulo 3, Moodle ofrece la posibilidad de ampliar la funcionalidad de su portal web añadiendo nuevas actividades que se podrán ofrecer a los alumnos.

Para crear un nuevo módulo de actividades, Moodle ofrece un conjunto de archivos base que se pueden modificar para conseguir la funcionalidad deseada siguiendo unos criterios para que se pueda integrar en el portal Moodle . Una vez modificados estos ficheros, para añadirlos a una instalación de Moodle, basta con copiar la carpeta con los ficheros en una carpeta del servidor de Moodle y lanzar un proceso de carga del módulo desde la web de Moodle.

La plantilla de los ficheros que componen un módulo de actividades se puede descargar desde el portal de Moodle.org y los ficheros principales son los siguientes: [7].

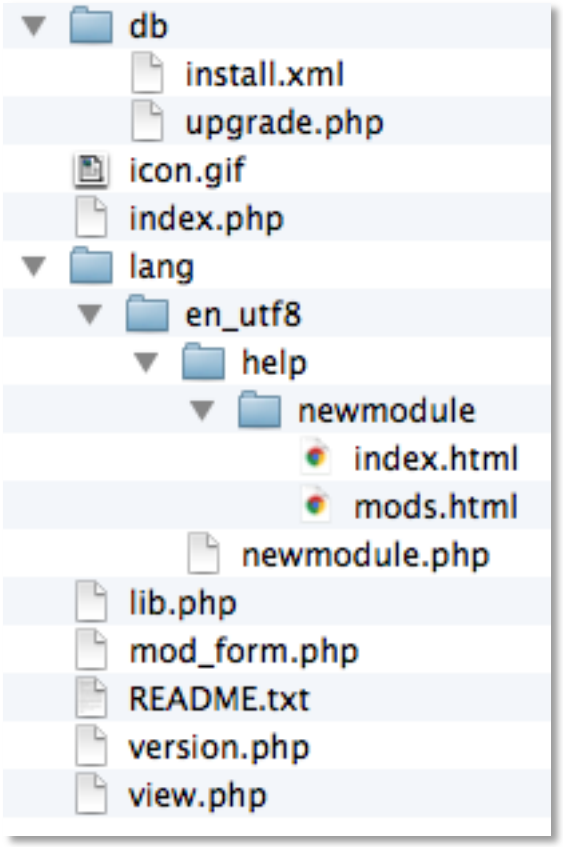


Figura 20: Estructura básica de un módulo de actividades de Moodle 1.9.

Cada fichero tiene una función específica, algunos de ellos son de uso imprescindible para la integración del módulo en Moodle y otros ofrecen funcionalidades complementarias de uso opcional:

- **db/install.xml** es el fichero xml que describe las tablas que necesita el módulo.
- **db/upgrade.php** es el fichero donde se escriben los cambios de estructura de las tablas del módulo.
- **Icon.gif** es un icono de 16px por 16px que sirve para identificar cada instancia del módulo.
- **Index.php** es una página web donde se listan todas las instancias del módulo en un curso.
- **lang** es una carpeta reservada para los paquetes de idiomas.
- **lib.php** es el fichero con las funcionalidades principales del módulo.
- **mod_form.php** es el fichero donde se describe el formulario de creación y edición del módulo.

- **version.php** es el fichero donde se indica la versión del módulo. Necesario en conjunción con upgrade.php para realizar modificaciones en las tablas.
- **view.php** es el primer código ejecutado por el módulo cuando se entra en la actividad.

Para conseguir implementar todas las funcionalidades definidas en el capítulo anterior hemos tenido que añadir más ficheros al módulo. Quedando la estructura de carpetas del módulo JLab de la siguiente forma:

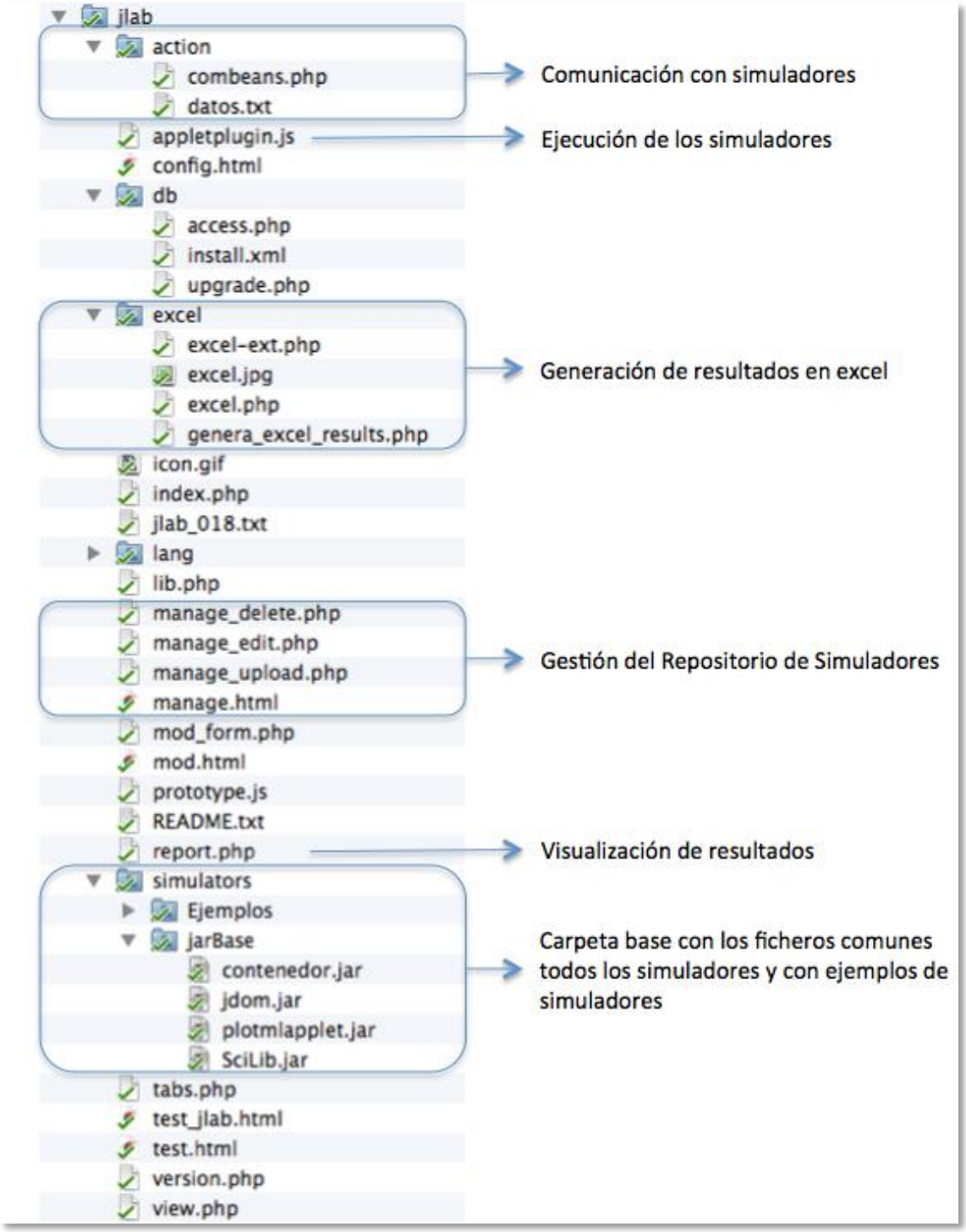


Figura 21: Estructura completa del módulo JLab donde se resaltan los ficheros añadidos a la estructura base.

Una vez se ha configurado todo correctamente añadimos nuestro módulo a Moodle simplemente copiando la estructura de carpetas del módulo dentro de la carpeta del portal web */mod/*.

Una vez copiado el módulo en el portal de Moodle, se ha de ir a la administración del portal Moodle para registrar el módulo e integrar sus tablas en la base de datos de Moodle.

En los siguientes apartados explicaremos el funcionamiento de los ficheros implicados en cada funcionalidad del módulo:

- Configuración de la base de datos:
  - *db/install.xml*
- Seguridad:
  - *db/access.php*
- Gestión del Repositorio de simuladores:
  - *manage.html*
  - *manage_upload.php*
  - *manage_edit.php*
  - *manage_delete.php*
- Configuración de una nueva actividad:
  - *mod.html*
  - *mod_form.php*
- Ejecución de los simuladores LAVICAD:
  - *View.php*
  - *appletplugin.js*
- Comunicación con simuladores
  - *action/combeans.php*
- Visualización de los Resultados:
  - *report.php*
- Descarga de los resultados en Hoja de Cálculo:
  - *Excel/Excel-ext.php*
  - *Excel.php*
  - *genera_excel_results.php*
- Fichero con funciones y consultas a base de datos
  - *lib.php*

La relación entre los ficheros que componen el módulo y las funcionalidades a implementar se pueden ver de forma gráfica en el diagrama funcional siguiente:

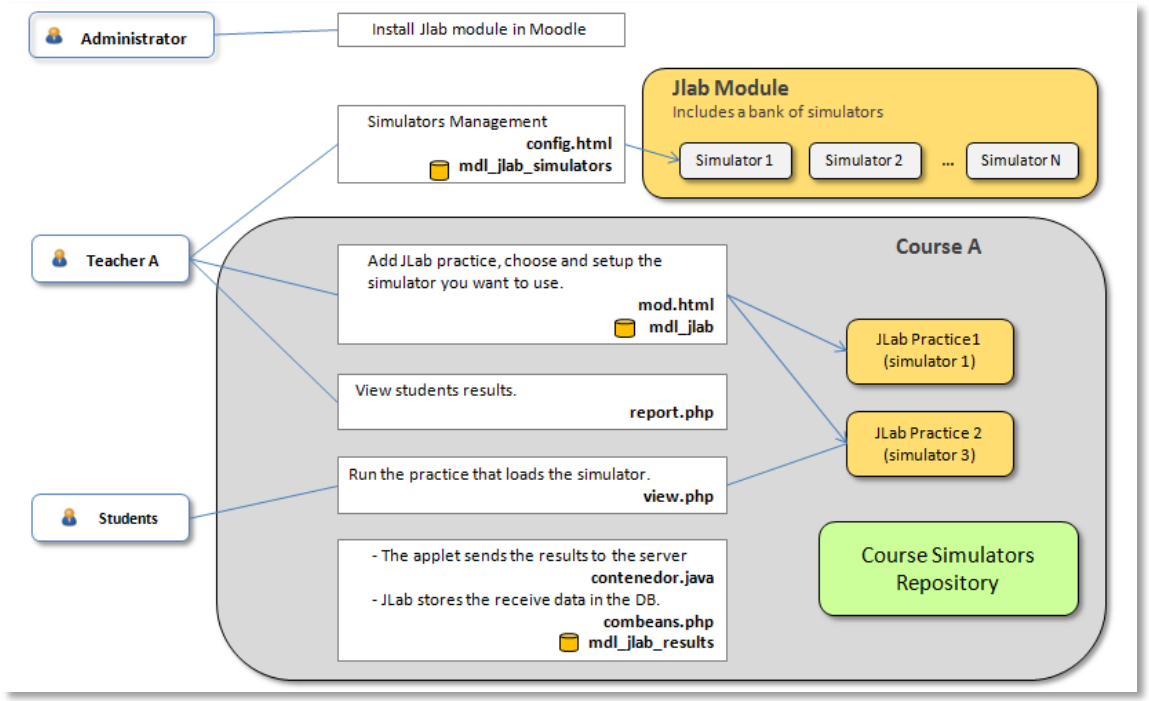


Figura 22: Diagrama funcional de JLab

6.2 Configuración Base de datos

JLab necesita almacenar en la base de datos de Moodle los simuladores disponibles, los simuladores asociados a las actividades y los resultados que obtienen los alumnos ejecutando estos simuladores. En el capítulo anterior diseñamos la estructura de tablas necesarias para el módulo JLab. Puede verse la estructura en la Figura 16.

Para integrar estas tablas en la propia base de datos de Moodle debemos indicar en el fichero install.xml las tablas y campos a crear. Una vez creado, se ha de copiar el fichero en la carpeta */mod/jlab/db*. Moodle leerá este fichero en la instalación del módulo e insertará en la base de datos de Moodle las tablas indicadas en el fichero.

Una forma sencilla de obtener el fichero install.xml es crear las tablas en una base de datos MySQL y utilizar el XMLDB Editor de Moodle, disponible en el apartado de Administración del Portal de Moodle. Este editor se conecta a la base de datos, detecta las tablas, campos, índices y tipos de datos, y los plasma en el xml. Se puede modificar la estructura del xml desde el mismo editor.

El fichero XML con la definición de las tablas de JLAB queda como sigue:

Tabla mdl_jlab.

```
<TABLES>
  <TABLE NAME="jlab" COMMENT="jlab table retrofitted from MySQL" NEXT="jlab_simulators">
    <FIELDS>
      <FIELD NAME="id" TYPE="int" LENGTH="10" NOTNULL="true" UNSIGNED="true" SEQUENCE="true" ENUM="false" NEXT="name"/>

      <FIELD NAME="name" TYPE="char" LENGTH="45" NOTNULL="true" SEQUENCE="false" ENUM="false" PREVIOUS="id" NEXT="description"/>

      <FIELD NAME="description" TYPE="text" LENGTH="small" NOTNULL="true" SEQUENCE="false" ENUM="false" PREVIOUS="name" NEXT="course"/>

      <FIELD NAME="course" TYPE="int" LENGTH="10" NOTNULL="true" UNSIGNED="true" SEQUENCE="false" ENUM="false" PREVIOUS="description" NEXT="simulator"/>

      <FIELD NAME="simulator" TYPE="int" LENGTH="10" NOTNULL="true" UNSIGNED="true" SEQUENCE="false" ENUM="false" PREVIOUS="course" NEXT="rec"/>

      <FIELD NAME="rec" TYPE="int" LENGTH="1" NOTNULL="true" UNSIGNED="true" DEFAULT="0" SEQUENCE="false" ENUM="false" PREVIOUS="simulator" NEXT="rec_start"/>

      <FIELD NAME="rec_start" TYPE="int" LENGTH="10" NOTNULL="true" UNSIGNED="true" SEQUENCE="false" ENUM="false" PREVIOUS="rec" NEXT="rec_end"/>

      <FIELD NAME="rec_end" TYPE="int" LENGTH="10" NOTNULL="true" UNSIGNED="true" SEQUENCE="false" ENUM="false" PREVIOUS="rec_start"/>

    </FIELDS>
    <KEYS>
      <KEY NAME="primary" TYPE="primary" FIELDS="id"/>
    </KEYS>
  </TABLE>
```

Figura 23: Código XML de definición de la tabla mdl_jlab.

- Tabla mdl_jlab_simulators

```
  <TABLE NAME="jlab_simulators" COMMENT="jlab_simulators table retrofitted from MySQL" PREVIOUS="jlab" NEXT="jlab_results">
    <FIELDS>
      <FIELD NAME="id" TYPE="int" LENGTH="10" NOTNULL="true" UNSIGNED="true" SEQUENCE="true" ENUM="false" NEXT="name"/>

      <FIELD NAME="name" TYPE="char" LENGTH="45" NOTNULL="true" SEQUENCE="false" ENUM="false" PREVIOUS="id" NEXT="description"/>

      <FIELD NAME="description" TYPE="char" LENGTH="100" NOTNULL="true" SEQUENCE="false" ENUM="false" PREVIOUS="name" NEXT="time_created"/>

      <FIELD NAME="time_created" TYPE="datetime" NOTNULL="true" SEQUENCE="false" ENUM="false" PREVIOUS="description" NEXT="time_modified"/>

      <FIELD NAME="time_modified" TYPE="datetime" NOTNULL="false" SEQUENCE="false" ENUM="false" PREVIOUS="time_created" NEXT="created_by"/>
```

```
    <FIELD NAME="created_by" TYPE="int" LENGTH="10" NOTNULL="true"
UNSIGNED="true" SEQUENCE="false" ENUM="false" PREVIOUS="time_modified"
NEXT="modified_by"/>

    <FIELD NAME="modified_by" TYPE="int" LENGTH="10" NOTNULL="false"
UNSIGNED="true" SEQUENCE="false" ENUM="false" PREVIOUS="created_by"
NEXT="visible"/>

    <FIELD NAME="visible" TYPE="int" LENGTH="10" NOTNULL="false"
UNSIGNED="true" DEFAULT="1" SEQUENCE="false" ENUM="false"
PREVIOUS="modified_by"/>

   </FIELDS>
   <KEYS>
    <KEY NAME="primary" TYPE="primary" FIELDS="id"/>
   </KEYS>
  </TABLE>
```

Figura 24: Código XML de definición de la tabla mdl_jlab_simulators.

Tabla mdl_jlab_results.

```
  <TABLE NAME="jlab_results" COMMENT="jlab_results table retrofitted from MySQL"
PREVIOUS="jlab_simulators">
   <FIELDS>
    <FIELD NAME="id" TYPE="int" LENGTH="10" NOTNULL="true" UNSIGNED="true"
SEQUENCE="true" ENUM="false" NEXT="jlab"/>

    <FIELD NAME="jlab" TYPE="int" LENGTH="10" NOTNULL="true" UNSIGNED="true"
SEQUENCE="false" ENUM="false" PREVIOUS="id" NEXT="user"/>

    <FIELD NAME="user" TYPE="int" LENGTH="10" NOTNULL="true" UNSIGNED="true"
SEQUENCE="false" ENUM="false" PREVIOUS="jlab" NEXT="etapa"/>

    <FIELD NAME="etapa" TYPE="char" LENGTH="200" NOTNULL="true"
SEQUENCE="false" ENUM="false" PREVIOUS="user" NEXT="valor1"/>

    <FIELD NAME="valor1" TYPE="char" LENGTH="200" NOTNULL="true"
SEQUENCE="false" ENUM="false" PREVIOUS="etapa" NEXT="valor2"/>

    <FIELD NAME="valor2" TYPE="char" LENGTH="200" NOTNULL="true"
SEQUENCE="false" ENUM="false" PREVIOUS="valor1" NEXT="valor3"/>

    <FIELD NAME="valor3" TYPE="char" LENGTH="200" NOTNULL="true"
SEQUENCE="false" ENUM="false" PREVIOUS="valor2" NEXT="valor4"/>

    <FIELD NAME="valor4" TYPE="char" LENGTH="200" NOTNULL="true"
SEQUENCE="false" ENUM="false" PREVIOUS="valor3" NEXT="valor5"/>

    <FIELD NAME="valor5" TYPE="char" LENGTH="200" NOTNULL="true"
SEQUENCE="false" ENUM="false" PREVIOUS="valor4"/>
   </FIELDS>
   <KEYS>
    <KEY NAME="primary" TYPE="primary" FIELDS="id"/>
   </KEYS>
  </TABLE>
 </TABLES>
```

Figura 25: Código XML de definición de la tabla mdl_jlab_results.

6.3 Seguridad

Tal y como se explicó en el capítulo 3, en Moodle los permisos se gestionan definiendo capacidades [7], que son combinaciones de permisos para los diferentes perfiles de usuario.

En el fichero *mod/jlab/db/Access.php* se definen las capacidades específicas para el módulo. En el módulo JLab hemos creado 2 capacidades:

- Visión:
  - Permiso de Lectura: 'read'
  - En el contexto de todo el módulo: CONTEXT_MODULE
  - Para los perfiles: invitado, estudiante, profesor y administrador.
- Gestión:
  - Permiso de Escritura: 'write'
  - En el contexto de todo el módulo: CONTEXT_MODULE
  - Para los perfiles: profesor y administrador.

6.4 Creación Instancia Actividad JLAB

Una instancia de JLab es una actividad JLab añadida a un curso. Al añadir cualquier actividad a un curso se abre una página de configuración común a todas las actividades donde se pide al profesor el nombre y la descripción con el que los alumnos verán la actividad.

Para añadir parámetros de configuración adicionales, debemos implementar el fichero */mod/jlab/mod.html*. Moodle inscrustará el código que pongamos en ese fichero en la página de configuración de la actividad.

En el caso de JLAB debemos configurar los siguientes parámetros:

- **Simulador**: Identificador de uno de los simuladores disponibles en el Repositorio.
- **Guardar Resultados**: Indica si se quieren almacenar en la base de datos los resultados de las ejecuciones del simulador
- **Fecha de Inicio y Fin**: Fechas entre las que sí se guardarán los resultados en la base de datos. Fuera de este intervalo no se guardarán.

El formulario *mod.html* utiliza el script *mod_form.php*. En este script se ha implementado la obtención de la lista de simuladores disponibles usando la función JLab_get_simulators() implementada en *lib.php*.

En *moodle/mod/jlab/lib.php* están todas las funciones que interactúan con las tablas de nuestro módulo JLab en la base de datos.

La función JLab_get_simulators() devuelve un array de objetos simulador con el id y el nombre de los simuladores disponibles.

En el fichero *mod_form.php* se construyen de manera dinámica los elementos a añadir en el formulario, indicando en cada elemento el nombre del campo de la tabla *mdl_jlab* en el que se volcará el dato introducido por el usuario.

Una cabecera para el grupo de parámetros específicos de JLab.

Un desplegable que muestre la lista de simuladores disponibles en el Repositorio. El Id del simulador seleccionado por el usuario se almacenará en el campo 'simulator'.

Un selector donde poder marcar si se guardan o no los resultados. La marca del usuario se almacenará en el campo 'rec' de la tabla *mdl_jlab*.

Y dos selectores de fecha para indicar la fecha de inicio y la de fin del guardado de los resultados. Los campos asociados son 'rec_start' y 'rec_end'.

El resultado final es la inclusión del apartado JLab en la página de configuración de la actividad.

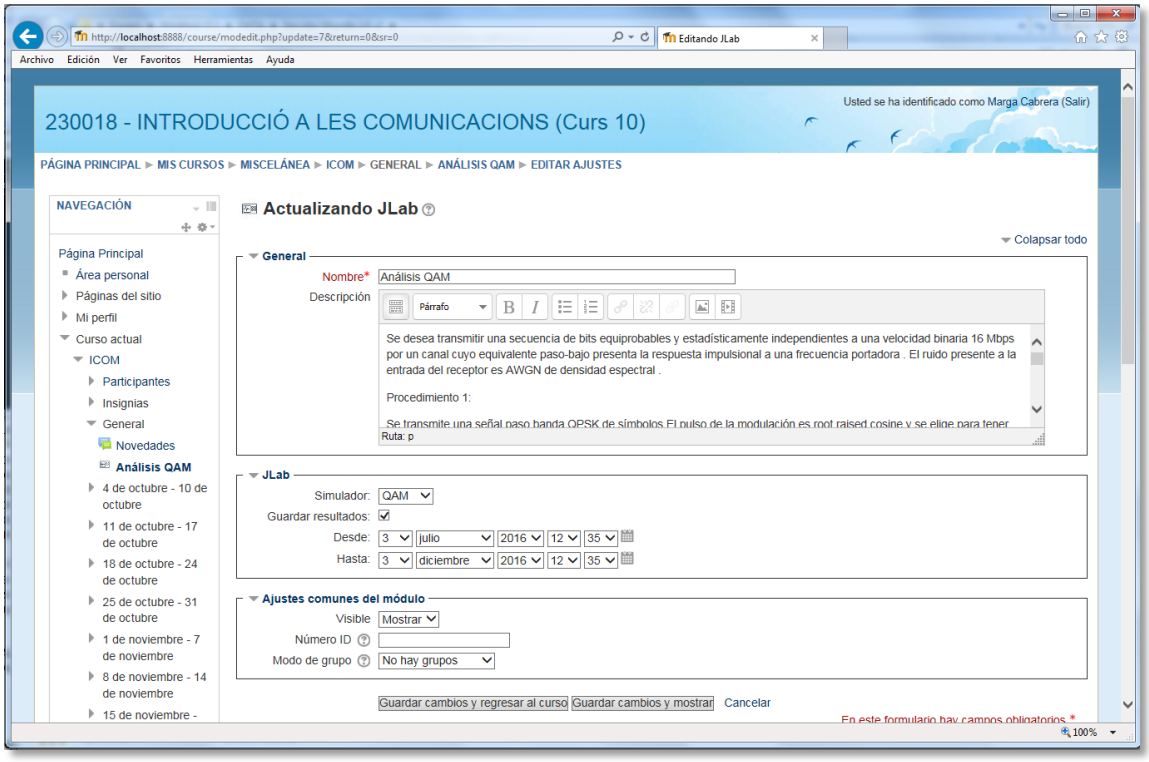


Figura 26: Página de creación de una actividad JLab.

Una vez configurados todos los datos de la nueva actividad y se pulsa guardar, Moodle guarda en una tabla propia de Moodle, *mdl_course_modules*, un registro para la nueva actividad, indicando el id del curso (course), id del módulo (module) y un id de instancia (instance).

El campo *instance* será equivalente al id de la actividad en nuestra tabla de instancias *mdl_jlab*.

En el caso de la creación de una nueva instancia de un módulo JLab, Moodle necesita que en el fichero /mod/jlab/*lib.php* estén definidas las funciones de inserción y actualización de la tabla mdl_jlab.

Al crear la instancia de la actividad, Moodle llama también a la función *jlab_add_instance* de nuestro fichero *lib.php* y le pasa un objeto con la información rellenada por el usuario en el formulario:

- Id de la actividad creada
- Descripción de la actividad.
- Id del Curso al que pertenece la actividad
- Id del Simulador seleccionado
- Si se guardan o no los resultados
- Rango de fechas para guardar los resultados

La información almacenada en la base de datos servirá para ejecutar el simulador cuando los alumnos entren en la actividad.

6.5 Visualizar Instancia Actividad JLAB

Todas las actividades de Moodle tienen una página donde se visualiza la actividad a realizar, es el fichero *View.php*.

En el caso de JLab lo hemos modificado para incluir tres pestañas:

- **Repositorio de Simuladores**: Pestañas desde donde se puede gestionar el repositorio. Sólo visible por profesores y el Administrador.
- **Simulador**: Pestaña donde se cargar el applet del simulador. Esta pestaña será visible por todos los usuarios.
- **Resultados**: Pestañas donde se visualizan los resultados de las ejecuciones de las simulaciones de los alumnos. Sólo será visible por profesores y por el administrador.

La página View.php recibe como parámetro en la URL el id de la instancia de JLab. A partir de este id podemos obtener toda la información de la actividad consultando a la base de datos.

A continuación la página comprueba los permisos del usuario conectado. Si es un alumno sólo puede visualizar la pestaña Simulador. Por lo que cargará el simulador asociado a la instancia. Véase el apartado 6.5.2 Ejecución de un simulador en una Actividad.

En caso de ser un profesor, que puede acceder a las tres pestañas (Simulador, Resultados, Repositorio), la página detecta la pestaña seleccionada y carga el contenido correspondiente.

A continuación, se detalla la implementación de cada pestaña.

6.5.1 Gestión del Repositorio de simuladores

El Repositorio de Simuladores es una carpeta accesible para todos los profesores donde podrán añadir simuladores LAVICAD para que todos los profesores puedan utilizarlos en sus cursos.

Por seguridad, esta carpeta, donde los usuarios de la web podrán escribir, no está dentro de la estructura de carpeta de la web Moodle, sino que está en una carpeta llamada MoodleData separada del resto y que incluso puede estar en un servidor diferente.

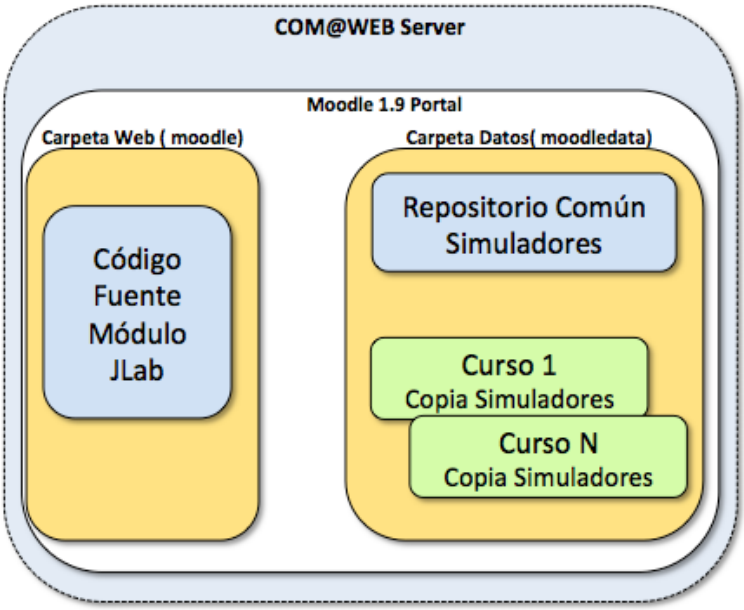


Figura 27: Arquitectura de JLab donde se muestra la ubicación del código fuente y del Repositorio de simuladores.

En cada instalación de Moodle, la ruta de la carpeta MoodleData puede variar. Para evitar errores, Moodle ofrece la variable global *$CFG->dataroot* que hace referencia a esta ruta y que podremos utilizar en el código de nuestro módulo.

Para darle al usuario la posibilidad de gestionar los simuladores del repositorio hemos implementado la página web *manage.html* que ofrece cuatro funcionalidades principales:

- Lista de los simuladores disponibles
- Formulario para añadir un nuevo simulador al repositorio y almacenarlo en la base de datos.
- Formulario para modificar los datos de un simulador ya existente
- Formulario para eliminar un simulador del repositorio.

6.5.1.1 Listado de simuladores del Repositorio

La página *manage.html* utiliza el script *manage_upload.php* para consultar en la base de datos los simuladores registrados en la tabla *mdl_jlab_simulators* y construir una tabla con todos los datos:

- **Id**: Identificador único del simulador
- **Nombre**: Nombre del simulador especificado por el usuario
- **Descripción**: Descripción del simulador
- **Modificado Por**: Usuario que modificó el registro del simulador por última vez.
- **Fecha de Modificación**: Fecha de la última modificación del registro.
- **Creado por**: Usuario que añadió el simulador al repositorio.
- **Fecha de Creación**: Fecha en la que se añadió el simulador al repositorio.
- **Instancias**: Nº de actividades que utilizan el simulador.
- **Visible**: Indica si el simulador está visible o no para añadirlo a nuevas actividades.

Primero obtenemos la lista de simuladores disponibles en el Repositorio y todos los datos necesarios para rellenar la tabla en la página.

Construimos una tabla en la página con los campos que se quieren mostrar.

Rellenamos la tabla con los datos de los simuladores y dibujamos la tabla en la página.

A continuación, se muestra un ejemplo de página de una instancia de una actividad donde se ha seleccionado la pestaña del

Repositorio de Simuladores y se puede ver la tabla con los simuladores disponibles.

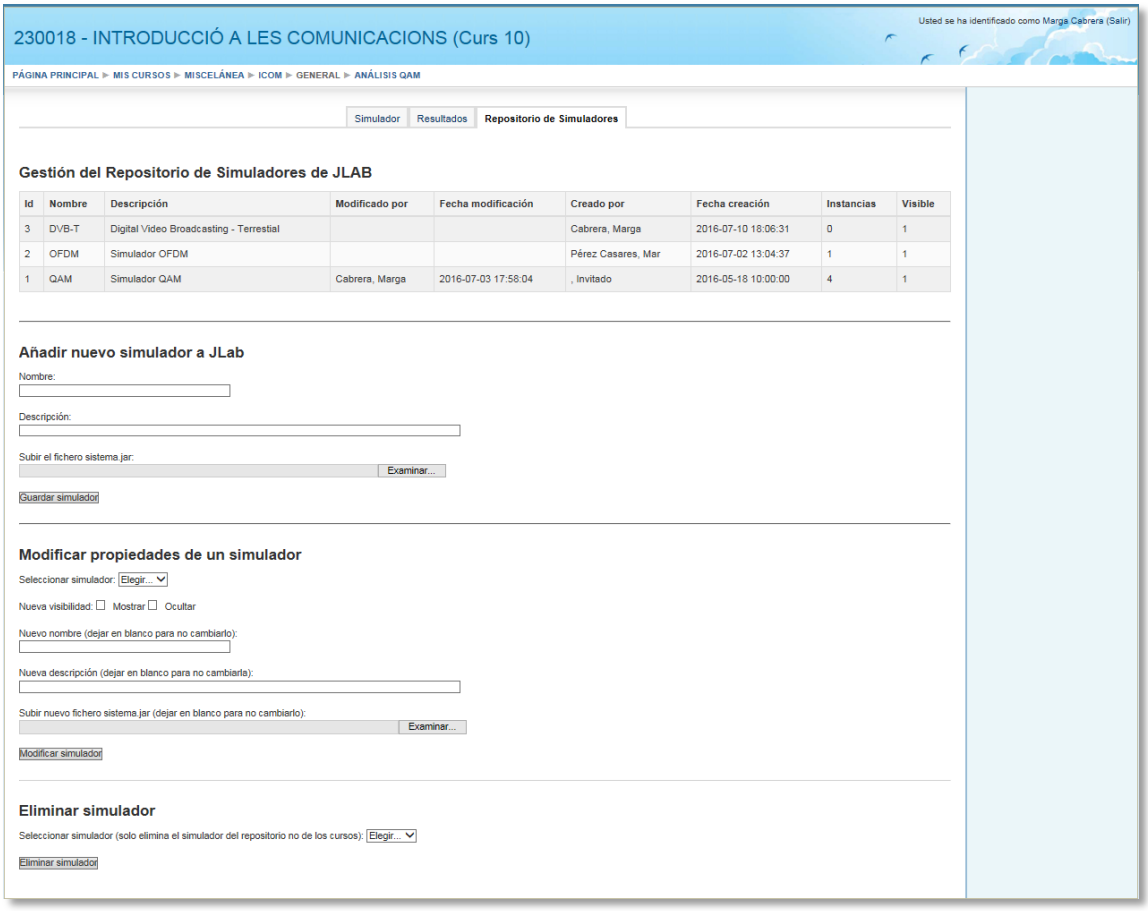


Figura 28: Página de gestión del Repositorio de Simuladores de JLab.

6.5.1.2 Añadir simulador al Repositorio

En la misma página manage.html se incluye un formulario cuyo submit se enlaza con el fichero *mod/jlab/manage_upload.php* que permite a un usuario con perfil profesor añadir un nuevo simulador al Repositorio y registrarlo en la base de datos.

El formulario pide al usuario los datos necesarios para añadir un registro nuevo en la tabla *mdl_jlab_simulators*:

- **Nombre**: nombre del simulador que servirá para seleccionarlo en una nueva actividad.
- **Descripción**: Descripción del simulador.
- **Fichero**: Permite al usuario seleccionar un fichero sistema.jar de su ordenador y subirlo al Repositorio.

Dentro del fichero View.php, la parte de código correspondiente al formulario de inserción de nuevos simuladores es el siguiente y utiliza el script *manage_upload.php*.

El formulario para añadir un simulador queda como se muestra en la Figura siguiente.

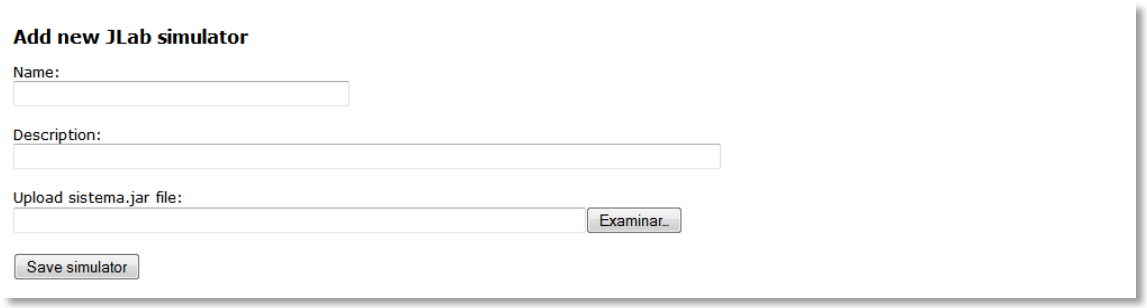


Figura 29: Página de Gestión del Repositorio de Simuladores JLab. Detalle del formulario para añadir nuevos simuladores.

Una vez el usuario pulsa 'Guardar simulador', el formulario ejecuta el código en *mod/jlab/manage_upload.php*, que inserta en la base de datos un nuevo simulador y copia el fichero subido por el usuario en la carpeta MoodleData dentro de una carpeta con nombre el Id del simulador. En esa carpeta copiamos también el resto de ficheros comunes a todos los simuladores (contenedor.jar, jdom.jar,plotmapplet.jar, sciLib.jar).

Si al subir el fichero se produce algún error, se detecta y se informa al usuario y no se guarda ningún fichero ni se inserta ningún registro en la tabla mdl_jlab_simulators. Los errores capturados son:

- No se admiten ficheros mayores a 5 MB.

- Si el fichero no se ha subido por completo, se rechaza.

- Si no se ha seleccionado ningún fichero también se avisa al usuario.

6.5.1.3 Modificar Simulador del Repositorio

La página *manage.html* también incluye un formulario enlazado al fichero *mod/jlab/manage_edit.php* que permite a los profesores modificar los datos de un simulador ya almacenado en el Repositorio.

Permite modificar:

- La visibilidad del simulador en nuevas actividades.
- El nombre del simulador almacenado en la base de datos
- La descripción del simulador

- el fichero sistema.jar, permite subir un nuevo fichero que sobrescribirá el anterior.

El formulario que pide los datos queda como se muestra en la Figura siguiente.

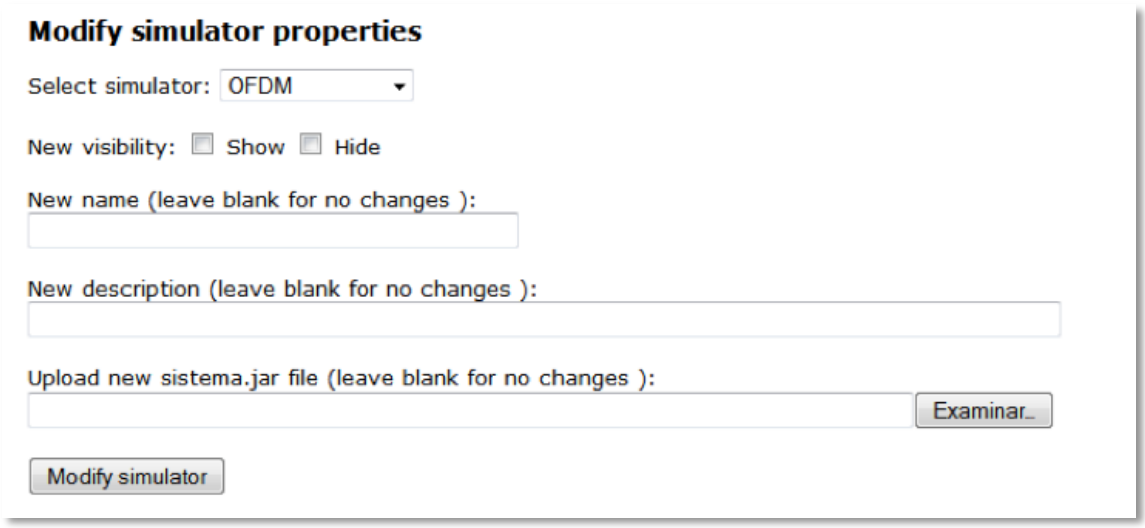


Figura 30: Página de Gestión del Repositorio de Simuladores JLab. Detalle del formulario para modificar un simulador.

Una vez se pulsa 'Modificar Simulador' se ejecuta el código del fichero *mod/jlab/manage_edit.php* que recoge los datos introducidos por el usuario en el formulario y que se han enviado por POST al fichero PHP y actualiza el registro del simulador en la tabla *mdl_jlab_simulators*.

Actualiza los cambios en un objeto simulador y se aplican los cambios en la base de datos, en la tabla mld_jlab_simulators.

Si el fichero es correcto, se sobrescribe el fichero sistema.jar almacenado en el

Una vez se ha finalizado se informa al usuario del resultado de la acción.

6.5.1.4 Eliminar Simulador del Repositorio

Por último, el fichero manage.html incluye un formulario enlazado al fichero *mod/jlab/manage_delete.php* que permite eliminar un simulador del repositorio.

Es importante destacar que no elimina el simulador de los cursos que lo utilicen, sólo lo borra del repositorio, lo que implica que no se podrá ofrecer en nuevas actividades. Los simuladores de una

actividad concreta están en una carpeta separada, como vimos en el apartado 6.4 Creación Instancia Actividad JLab.

El formulario sólo pide que seleccione uno de los simuladores disponibles en el repositorio.

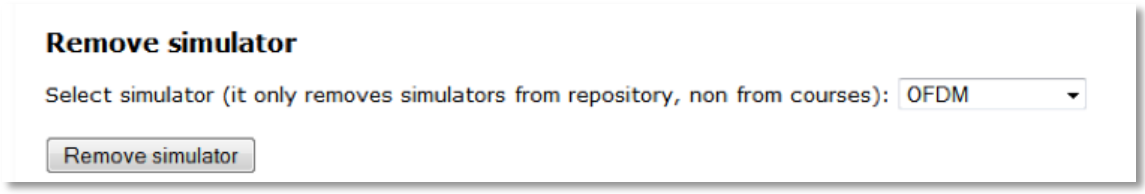


Figura 31: Página de Gestión del Repositorio de Simuladores JLab. Detalle del formulario para eliminar un simulador.

Una vez el usuario pulsa 'Eliminar simulador' se elimina la carpeta de MoodleData que contiene los ficheros del simulador eliminado y además se borra el registro de la base de datos. Por lo que el simulador ya no aparecerá en la lista de simuladores ni estará disponible para nuevas actividades.

Recogemos el id del simulador a eliminar de los parámetros que nos llegan por POST.

Si el identificador no es nulo, borramos el simulador de la base de datos.

A continuación, borramos los ficheros del simulador del Repositorio de Simuladores.

Al finalizar se informa al usuario del resultado de la acción.

6.5.2 Ejecución de un Simulador en una Actividad

La ejecución del simulador se realiza en la pestaña Simulador de la actividad, en la página *View.php*.

Una vez se ha determinado que se tiene que ejecutar el simulador, se ejecuta el siguiente código javascript que se encarga de llamar a la función *writePlugin* implementada en *appletplugin.js*.

A la función *writeplugin* se le pasa como parámetros toda la información de contexto de la actividad: id de la actividad, id del usuario que lo ejecuta, id del simulador a cargar, si se han de guardar resultados, la IP del servidor y la ruta de los ficheros .jar.

La información de contexto está accesible en la página *View.php*, pero no lo estará en el applet del simulador porque los applets no pueden acceder a los datos de la página que los aloja. Y

los datos de contexto son necesarios en el Applet para poder enviar los resultados al servidor indicando de qué curso, actividad y alumno son. Para salvar este problema se ha tenido que modificar el fichero *Contenedor.jar* para recibir los parámetros de contexto. Esto lo veremos más en detalle en el apartado siguiente.

La función writePlugin está implementada en el fichero *appletplugin.js* y lo que hace es escribir el código javascript necesario para ejecutar en el cliente el applet del simulador que se indica en el parámetro jarbase. Además, le pasa todos los parámetros de contexto al applet.

Una vez se ejecuta el código javascript indicado más arriba se muestra el simulador dentro de la página de la actividad JLab.

Figura 32: Página View.php donde se muestra la ejecución del applet del simulador dentro del portal Moodle.

6.5.3 Visualización de Resultados

los resultados de las simulaciones de los alumnos de una actividad se muestran dentro de la misma actividad, en una pestaña sólo visible para profesores y administradores.

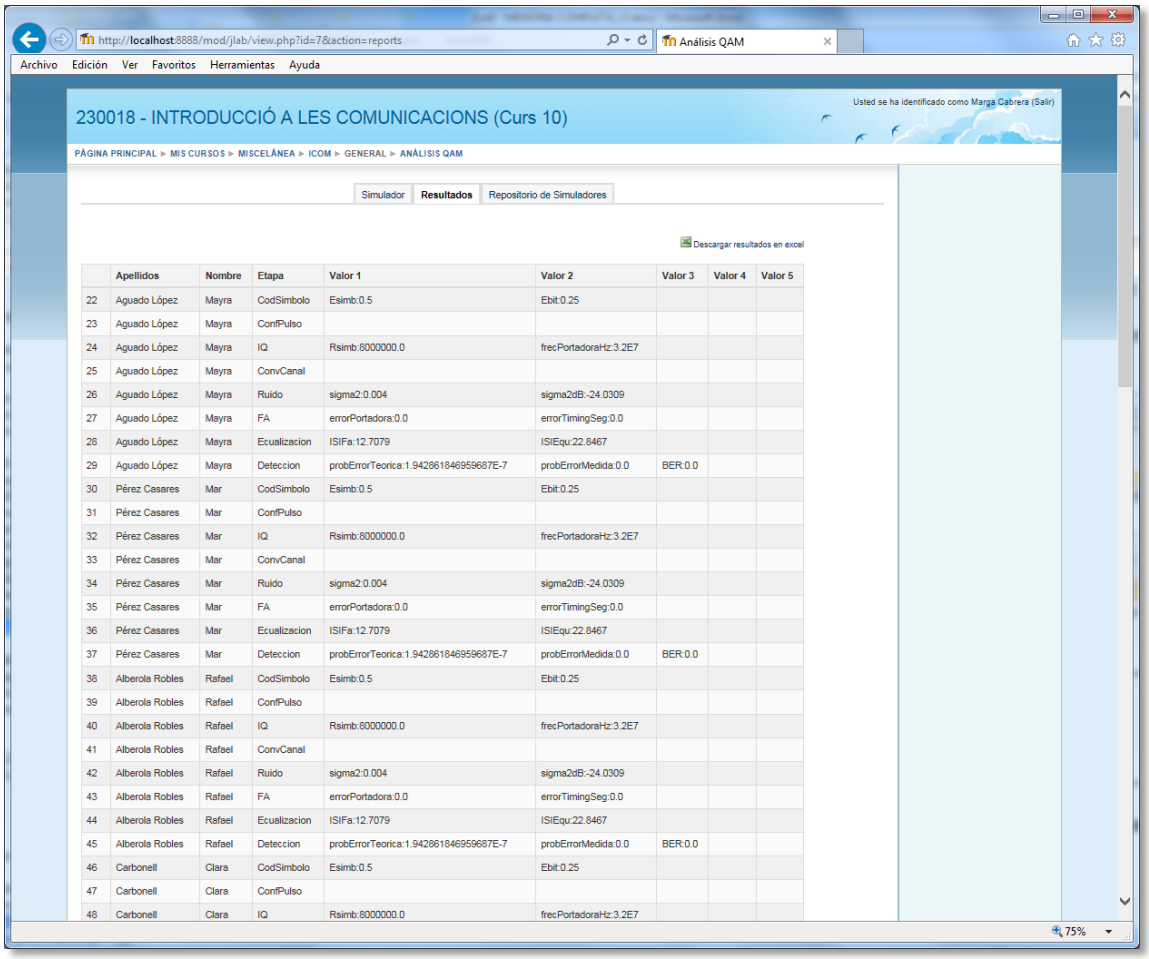
Al solicitar los resultados de una actividad, la página *View.php* consulta en la base de datos, en la tabla *mdl_jlab_results*, todos los resultados almacenados de la actividad y los muestra en formato tabla.

Los datos que muestra para cada resultado son:

- **Apellidos**: Apellido del alumno que ha realizado la simulación.
- **Nombre**: Nombre del alumno.
- **Etapa**: Nombre de la Etapa del simulador al que pertenecen los resultados.
- **Valor 1 a 5**: muestran el nombre del parámetro y el valor. Se admiten hasta 5 parámetros por etapa.

El script */mod/jlab/report.php* consulta los resultados almacenados de la actividad y todos los datos necesarios. Construye una tabla de datos y la rellena con los datos obtenidos. Añade un enlace a la descarga de los datos en una hoja de cálculo. Y finalmente, se muestra la tabla en la página.

Los resultados que se ven son los de la última ejecución que ha enviado el alumno, es decir, que cada vez que un alumno envía un resultado de una etapa, se sobrescribe cualquier resultado que hubiera enviado con anterioridad.



	Apellidos	Nombre	Etapa	Valor 1	Valor 2	Valor 3	Valor 4	Valor 5
22	Aguado López	Mayra	CodSimbolo	Esimb:0.5	Ebit:0.25			
23	Aguado López	Mayra	ConfPulso					
24	Aguado López	Mayra	IQ	Rsimb:8000000.0	frecPortadoraHz:3.2E7			
25	Aguado López	Mayra	ConvCanal					
26	Aguado López	Mayra	Ruido	sigma2:0.004	sigma2dB:-24.0309			
27	Aguado López	Mayra	FA	errorPortadora:0.0	errorTimingSeg:0.0			
28	Aguado López	Mayra	Ecualizacion	ISIFa:12.7079	ISIEqu:22.8467			
29	Aguado López	Mayra	Deteccion	probErrorTeorica:1.942861846959687E-7	probErrorMedida:0.0	BER:0.0		
30	Pérez Casares	Mar	CodSimbolo	Esimb:0.5	Ebit:0.25			
31	Pérez Casares	Mar	ConfPulso					
32	Pérez Casares	Mar	IQ	Rsimb:8000000.0	frecPortadoraHz:3.2E7			
33	Pérez Casares	Mar	ConvCanal					
34	Pérez Casares	Mar	Ruido	sigma2:0.004	sigma2dB:-24.0309			
35	Pérez Casares	Mar	FA	errorPortadora:0.0	errorTimingSeg:0.0			
36	Pérez Casares	Mar	Ecualizacion	ISIFa:12.7079	ISIEqu:22.8467			
37	Pérez Casares	Mar	Deteccion	probErrorTeorica:1.942861846959687E-7	probErrorMedida:0.0	BER:0.0		
38	Alberola Robles	Rafael	CodSimbolo	Esimb:0.5	Ebit:0.25			
39	Alberola Robles	Rafael	ConfPulso					
40	Alberola Robles	Rafael	IQ	Rsimb:8000000.0	frecPortadoraHz:3.2E7			
41	Alberola Robles	Rafael	ConvCanal					
42	Alberola Robles	Rafael	Ruido	sigma2:0.004	sigma2dB:-24.0309			
43	Alberola Robles	Rafael	FA	errorPortadora:0.0	errorTimingSeg:0.0			
44	Alberola Robles	Rafael	Ecualizacion	ISIFa:12.7079	ISIEqu:22.8467			
45	Alberola Robles	Rafael	Deteccion	probErrorTeorica:1.942861846959687E-7	probErrorMedida:0.0	BER:0.0		
46	Carbonell	Clara	CodSimbolo	Esimb:0.5	Ebit:0.25			
47	Carbonell	Clara	ConfPulso					
48	Carbonell	Clara	IQ	Rsimb:8000000.0	frecPortadoraHz:3.2E7			

Figura 33: Página de visualización de los resultados de una actividad JLab.

6.5.4 Descarga de resultados en Hoja de Cálculo

La descarga de los resultados de una actividad en una hoja de cálculo está accesible sólo a los profesores desde la página de la actividad, en la pestaña "Resultados", mediante el enlace "Descargar resultados en Excel".

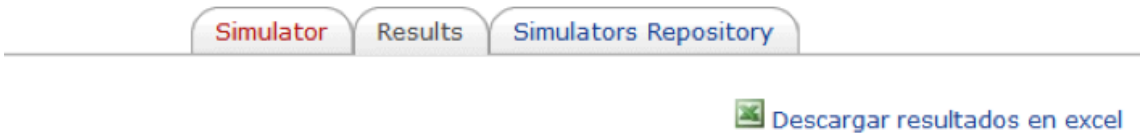


Figura 34: Enlace a la descarga de los resultados en excel.

Este enlace, en realidad lo que hace es llamar al fichero */mod/jlab/Excel/genera_excel_results.php* pasándole el id de la actividad.

Lo primero que se hace en el fichero *genera_excel_results.php* es cargar unas librerías [47] que generan un fichero Excel con los datos que se le pasen.

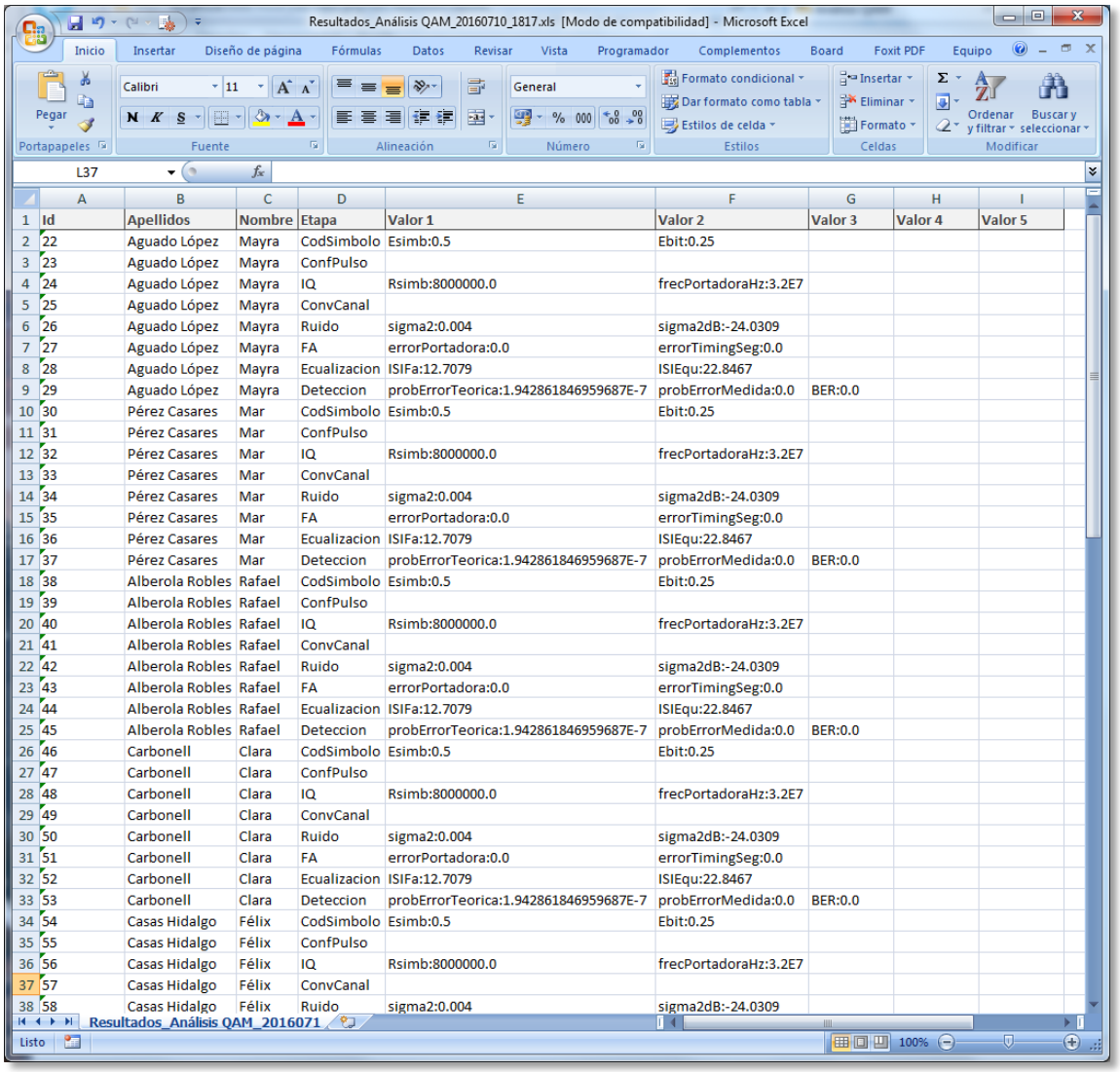
Para generar los datos, el primer paso es leer el Id de la actividad cuyos resultados se quieren descargar de los parámetros de la URL.

Con ese Id vamos a consultar en la base de datos los resultados almacenados de la actividad cargada, además queremos mostrar el nombre y apellidos de los usuarios, así como el nombre del simulador. Para ello debemos cruzar los datos de tres tablas: user, jlab y jlab_results.

A continuación, construimos una matriz con los datos. Cada fila será un resultado de una etapa. Y para cada fila tendremos varios campos: apellidos del estudiante, nombre del estudiante, nombre de la etapa y resultado de cada valor.

Para descargar los resultados en una hoja de cálculo utilizamos las librerías de Excel importadas al inicio, que se encargan de volcar en un fichero Excel con el nombre que le indiquemos, $file_name, los datos que se le pasen, $result.

Y, finalmente, se abre la hoja de cálculo.



Resultados_Análisis QAM_20160710_1817.xls [Modo de compatibilidad] - Microsoft Excel

Id	Apellidos	Nombre	Etapa	Valor 1	Valor 2	Valor 3	Valor 4	Valor 5
22	Aguado López	Mayra	CodSimbolo	Esimb:0.5	Ebit:0.25			
23	Aguado López	Mayra	ConfPulso					
24	Aguado López	Mayra	IQ	Rsimb:8000000.0	frecPortadoraHz:3.2E7			
25	Aguado López	Mayra	ConvCanal					
26	Aguado López	Mayra	Ruido	sigma2:0.004	sigma2dB:-24.0309			
27	Aguado López	Mayra	FA	errorPortadora:0.0	errorTimingSeg:0.0			
28	Aguado López	Mayra	Ecualizacion	ISIFa:12.7079	ISIEqu:22.8467			
29	Aguado López	Mayra	Deteccion	probErrorTeorica:1.942861846959687E-7	probErrorMedida:0.0	BER:0.0		
30	Pérez Casares	Mar	CodSimbolo	Esimb:0.5	Ebit:0.25			
31	Pérez Casares	Mar	ConfPulso					
32	Pérez Casares	Mar	IQ	Rsimb:8000000.0	frecPortadoraHz:3.2E7			
33	Pérez Casares	Mar	ConvCanal					
34	Pérez Casares	Mar	Ruido	sigma2:0.004	sigma2dB:-24.0309			
35	Pérez Casares	Mar	FA	errorPortadora:0.0	errorTimingSeg:0.0			
36	Pérez Casares	Mar	Ecualizacion	ISIFa:12.7079	ISIEqu:22.8467			
37	Pérez Casares	Mar	Deteccion	probErrorTeorica:1.942861846959687E-7	probErrorMedida:0.0	BER:0.0		
38	Alberola Robles	Rafael	CodSimbolo	Esimb:0.5	Ebit:0.25			
39	Alberola Robles	Rafael	ConfPulso					
40	Alberola Robles	Rafael	IQ	Rsimb:8000000.0	frecPortadoraHz:3.2E7			
41	Alberola Robles	Rafael	ConvCanal					
42	Alberola Robles	Rafael	Ruido	sigma2:0.004	sigma2dB:-24.0309			
43	Alberola Robles	Rafael	FA	errorPortadora:0.0	errorTimingSeg:0.0			
44	Alberola Robles	Rafael	Ecualizacion	ISIFa:12.7079	ISIEqu:22.8467			
45	Alberola Robles	Rafael	Deteccion	probErrorTeorica:1.942861846959687E-7	probErrorMedida:0.0	BER:0.0		
46	Carbonell	Clara	CodSimbolo	Esimb:0.5	Ebit:0.25			
47	Carbonell	Clara	ConfPulso					
48	Carbonell	Clara	IQ	Rsimb:8000000.0	frecPortadoraHz:3.2E7			
49	Carbonell	Clara	ConvCanal					
50	Carbonell	Clara	Ruido	sigma2:0.004	sigma2dB:-24.0309			
51	Carbonell	Clara	FA	errorPortadora:0.0	errorTimingSeg:0.0			
52	Carbonell	Clara	Ecualizacion	ISIFa:12.7079	ISIEqu:22.8467			
53	Carbonell	Clara	Deteccion	probErrorTeorica:1.942861846959687E-7	probErrorMedida:0.0	BER:0.0		
54	Casas Hidalgo	Félix	CodSimbolo	Esimb:0.5	Ebit:0.25			
55	Casas Hidalgo	Félix	ConfPulso					
56	Casas Hidalgo	Félix	IQ	Rsimb:8000000.0	frecPortadoraHz:3.2E7			
57	Casas Hidalgo	Félix	ConvCanal					
58	Casas Hidalgo	Félix	Ruido	sigma2:0.004	sigma2dB:-24.0309			

Figura 35: Hoja de cálculo con los resultados de una actividad JLab.

Una de las ventajas de que la hoja de cálculo se genere en tiempo de ejecución es que contendrá la información actualizada en ese mismo instante.

Además, el fichero sólo se muestra. Será el profesor quien lo guarde en su sistema de archivos propio. Por lo que en Moodle no se guarda ningún fichero Excel.

6.6 Envío de resultados a JLAB

La comunicación entre el applet del simulador y la actividad JLab se ha realizado partiendo del trabajo desarrollado en el departamento de Teoría de Señal y Comunicaciones [14] donde se enviaba el usuario del applet al Moodle. En este proyecto hemos modificado la clase Contenedor y la clase Etapa de LAVICAD para

incluir además el curso, la actividad, la etapa y pares clave-valor con los resultados de la etapa.

A continuación, se muestra el diagrama secuencial del proceso de comunicación entre los simuladores y JLab.

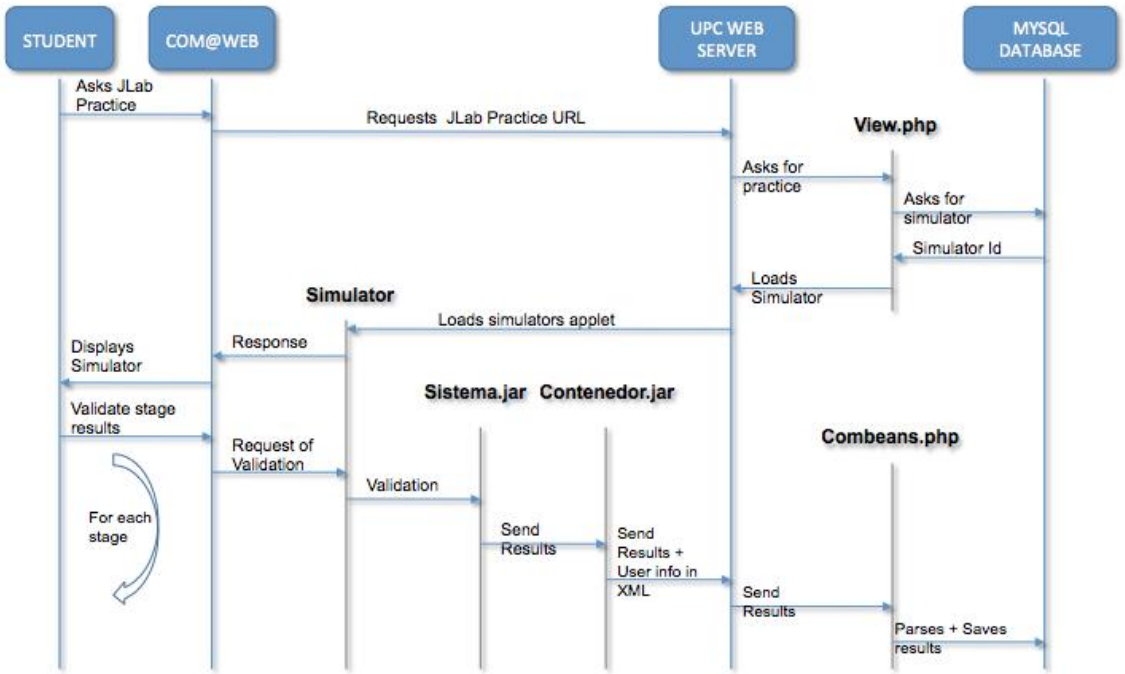


Figura 36: Diagrama secuencial de la comunicación entre el JLab y las etapas de los simuladores LAVICAD.

Tal y como se ha visto en el apartado anterior, cuando el alumno entra en una actividad JLab, se ejecuta la página *View.php* que consulta en la base de datos el simulador asociado a la actividad.

La página *View.php* se ejecuta en el servidor, por lo que tiene acceso a la información de contexto (curso, actividad, usuario). Con esta información llama a la función *writePlugin* del fichero javascript *appletplugin.js* que se encarga de embeber un objeto que permite ejecutar el applet del simulador dentro de la página *View.php.* Al simulador se le pasa como parámetro los datos de contexto.

Una vez hecho esto, se ejecuta el applet del simulador en el cliente, en el ordenador del alumno. El alumno puede ejecutar las diferentes etapas del simulador, modificar los parámetros de entrada y pulsará Validar datos de la etapa. Este proceso genera las señales de salida y los resultados de salida.

El proceso de Validar lo que hace es comprobar si está activado el almacenamiento de los resultados y, si es afirmativo, envía al servidor de Moodle, en concreto al fichero /mod/jlab/action/*combeans.php*, los datos de contexto junto con los valores de cada parámetro de la etapa en un fichero XML.

El fichero *combeans.php* recibe el fichero XML, lo parsea y almacena los datos en la tabla *mdl_jlab_results* de la base de datos.

Una vez almacenados los resultados ya están disponibles para su consulta por parte del profesor del curso al que pertenezcan.

Para conseguir que esta comunicación funcione se han tenido que realizar varias modificaciones en varios desarrollos previos. A continuación, detallamos cada uno de ellos.

6.6.1 Appletpluggin.js

Se ha modificado el fichero javascript *appletpluggin.js* para incluir el id de la actividad, el id del usuario, el id del curso y el parámetro que indica si se han de guardar o no los resultados de esa ejecución. Esta información es la que llamamos datos de contexto de la simulación. Además, se incluye la IP del servidor para que el simulador sepa dónde enviar los datos.

6.6.2 Etapa.java del proyecto LAVICAD

Se ha modificado la clase *Etapa.java* del proyecto LAVICAD para añadir también nuevos atributos que alojen los datos de contexto.
Y para modificar el envío de datos al servidor para incluir éstos nuevos datos.

6.6.3 Contenedor.java del proyecto LAVICAD

Se ha modificado la clase *Contenedor.java* de LAVICAD para recibir los nuevos parámetros de contexto vía web al cargar el applet del simulador y encargarse de rellenar los atributos de contexto de cada etapa.

El archivo *contenedor.java* es el primer archivo que se procesa cuando se ejecuta el applet. Es el que recibe los parámetros identificativos de la práctica y el usuario. Estos parámetros deben pasar a las etapas.

6.6.4 Combeans.php

Se ha modificado el receptor de datos */mod/jlab/action/combeans.php* para incluir el id de la actividad JLab a la que pertenecen los resultados recibidos.
Leemos los parámetros que llegan y los guardamos en un objeto Simulador, $mdl_jlab_results.

Borramos cualquier resultado previamente guardado para este usuario, simulador y etapa.
Y, finalmente, insertamos el objeto en base de datos, en la tabla mdl_jlab_results.

6.7 Conclusiones

En este capítulo hemos visto cómo se crea el módulo de actividades JLAB y cómo se añaden a la base de datos de Moodle las tablas necesarias para su funcionamiento.

También se ha explicado cómo se configura el Repositorio de Simuladores y los desarrollos realizados para su completa gestión.

Se ha detallado la implementación del formulario de nueva actividad donde se crea la actividad de Moodle asociando un simulador.

Y hemos visto también los cambios que se han realizado en el fichero Contenedor de los simuladores LAVICAD, el fichero común a todos los simuladores, para incluir los parámetros de contexto de la simulación (curso, actividad, alumno) y para que el simulador pueda enviar los resultados de la simulación al servidor de Moodle. Donde tenemos un receptor, el fichero combeans.php, que se encarga de almacenarlos en la base de datos.

Por último, se ha detallado la implementación de la visualización y descarga de los resultados.

Llegado a este punto del proyecto ya tenemos el módulo JLab implementado. En el siguiente capítulo veremos cómo se realiza el despliegue en un servidor de Moodle.

7. Despliegue

7.1 Introducción

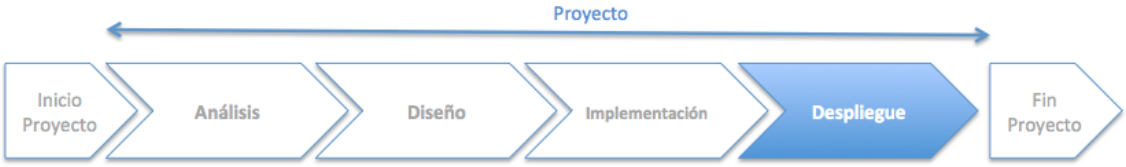


Figura 37: Fase de Despliegue de un proyecto de software.

En este capítulo se explica cómo instalar el módulo JLab en un portal Moodle.
Además veremos un ejemplo práctico del uso de JLab en un curso de Comunicaciones de la facultad ETSETB de la UPC.

7.2 Instalación y puesta en marcha

Partimos de una instalación de Moodle 1.9 o superior ya funcionando.

El módulo de actividades JLab lo tiene que instalar un administrador del servidor Moodle, ya que necesita acceso a las carpetas principales del Moodle.

La instalación de JLab se realiza siguiendo los siguientes pasos:

1. Copiar la carpeta jlab en moodle/mod/
2. Entrar en la web de moodle, en el apartado de Administración y pulsar Notificaciones.

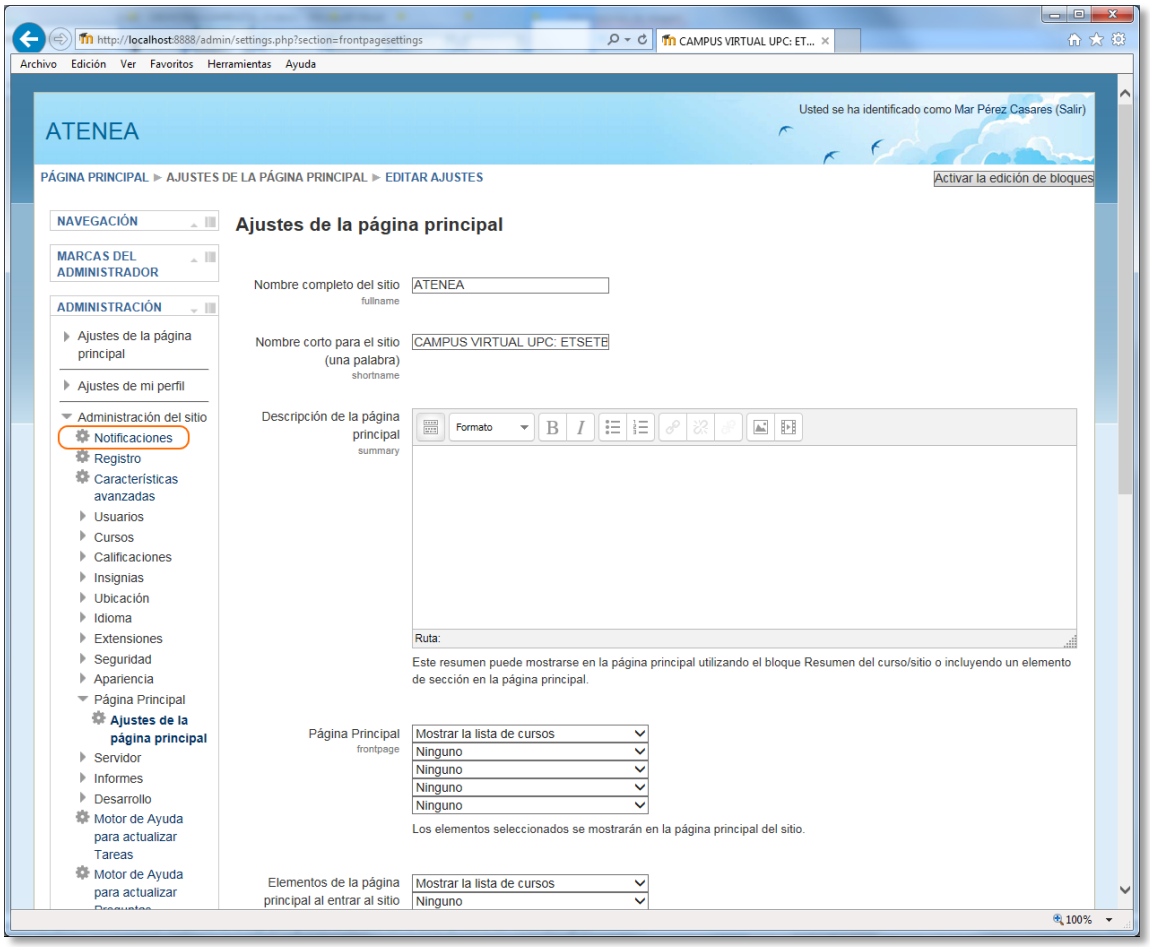


Figura 38: Instalación de JLab en Moodle. Site Administration > Notifications.

El proceso de Notificaciones instalará el módulo de actividades en Moodle y añadirá las tablas definidas en el fichero *mod/jlab/db/install.xml*. Mostrando el estado de la creación de cada tabla.

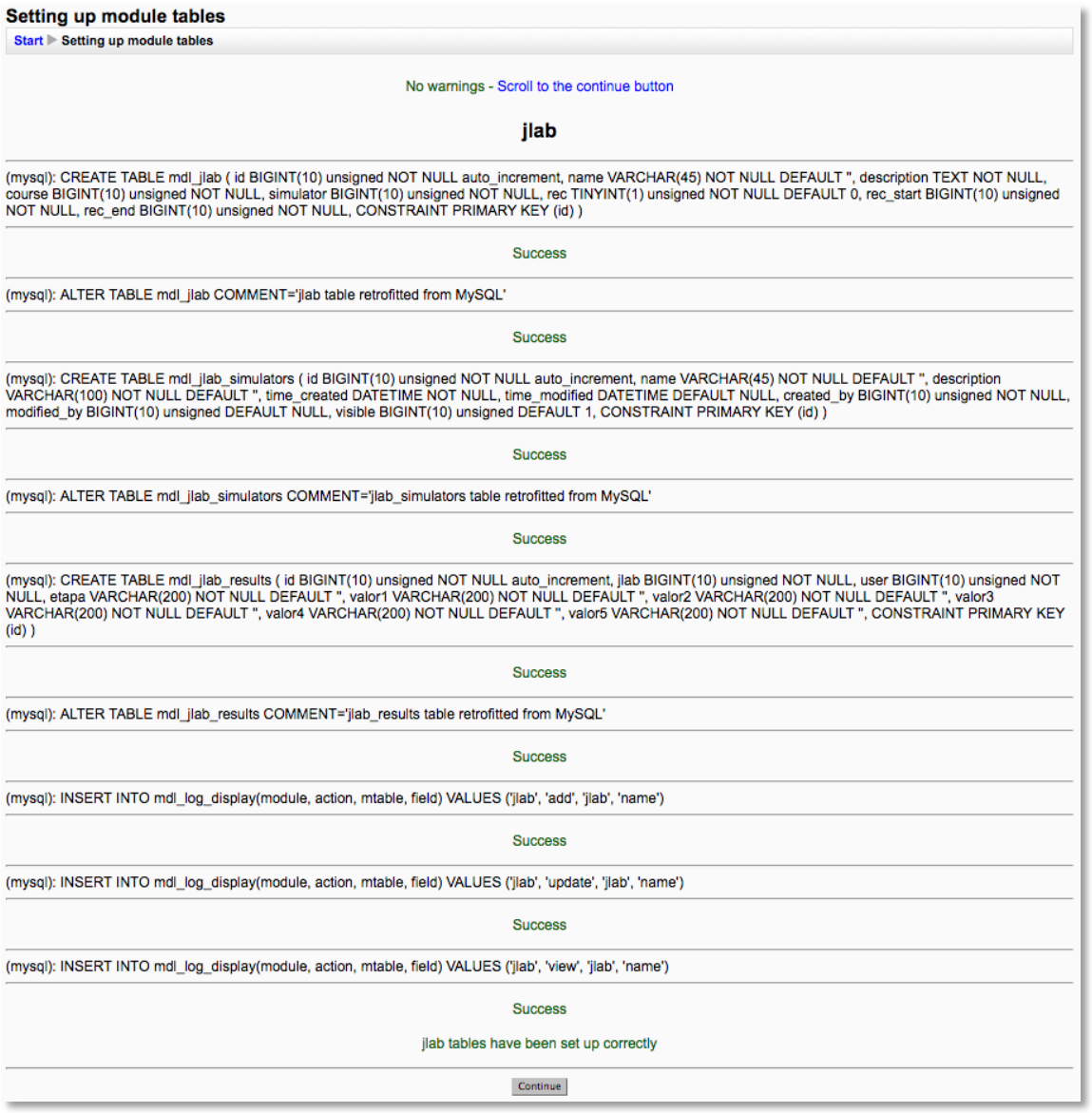


Figura 39: Resultado de la instalación del módulo JLab.

Una vez se ha terminado la instalación con éxito se debe comprobar que aparece una nueva actividad. Para ello se debe ir a Administración del Sitio > Modulos > Actividades. Y tiene que aparecer JLab.

Si se pulsa en JLab se muestra la pantalla de configuración del módulo donde se indica:

- **IP del servidor**: IP del servidor necesaria para comunicar los simuladores applet con Moodle.

- **Repository**: Indica la carpeta donde se aloja el repositorio de simuladores.

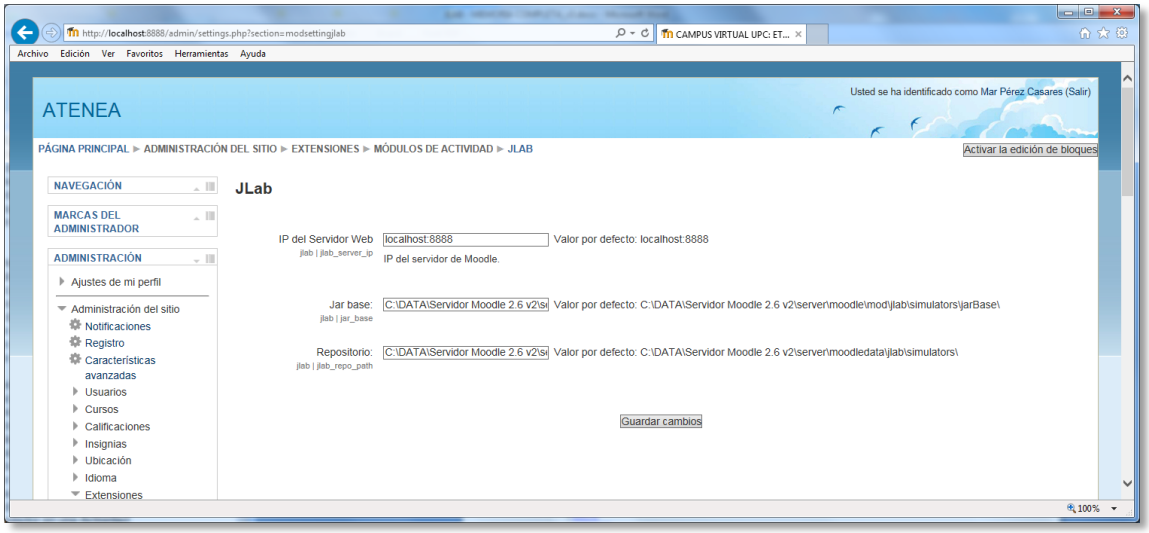


Figura 40: Listado de las Actividades instaladas en Moodle.

Una vez instalado el módulo JLab en Moodle, ya aparecerá como una actividad más para crear en los cursos.

8. Ejemplo de Práctica JLab en un Curso

En ese apartado vamos a mostrar un ejemplo práctico de uso del módulo JLAB apropiado para la asignatura de Introducción a las Comunicaciones (ICOM) de los estudios de grado en Enginyeria de Tecnologies i Serveis de Telecomunicació de la escuela ETSETB de la UPC.

8.1 Práctica QAM

El profesor de la asignatura 'INTRODUCCIÓ A LES COMUNICACIONS' quiere proponer un ejercicio donde se trabaje con el sistema de comunicaciones QAM.

El enunciado de la práctica es el siguiente:

Se desea transmitir una secuencia de bits equiprobables y estadísticamente independientes a una velocidad binaria $r_b =$ 16 Mbps por un canal cuyo equivalente paso-bajo presenta la respuesta impulsional $b_{h_c}(t) = \delta(t) - 0.2\delta(t-T)$ a una frecuencia portadora $f_c = 64MHz$. El ruido presente a la entrada del receptor, $w(t)$, es AWGN de densidad espectral $S_w(f) = \frac{N_0}{2}$; $N_0 = 10^{-6} mwatt/Hz$.

Procedimiento 1:

Se transmite una señal paso banda QPSK de símbolos $\pm 1 \pm j$ El pulso de la modulación $p(t)$ es root raised cosine y se elige para tener un ancho de banda $B_p = 5MHz$ y energía $E_p = 0.06324mjul$

Se pide:

a) Dibuje el espacio de señal y halle la energía media transmitida por símbolo E_s la energía media transmitida por bit E_b . Etapa 1
b) Calcule el parámetro de roll-off α para que el ancho de banda del pulso sea $B_p = 5MHz$ y visualice la transformada de Fourier del pulso en dB Etapa 2
c) Halle la frecuencia portadora en función de la velocidad de símbolo *r*. Etapa 3
d) Dibuje la respuesta impulsional del equivalente paso bajo del canal Etapa 4
e) Halle el cociente $\frac{E_b}{N_0}$ en dB Etapa 5
f) Dibuje el espacio de señal a la salida del filtro adaptado con y sin ruido. Etapa 6
g) Diseñe un ecualizador forzador de ceros de dos coeficientes y halle la ISI residual en dB a la entrada y a la salida del filtro adaptado. Etapa 7
h) Calcule la probabilidad de error. Etapa 8

En la siguiente imagen se muestra la pantalla de configuración de la actividad JLab, donde el profesor selecciona el simulador QAM, Indica que quiere almacenar los resultados de los alumnos que ejecuten el simulador entre Marzo y Julio de 2016. De esta formalas ejecuciones realizadas fuera de este plazo no se almacenarán en Moodle.

Una vez se guarda la configuración de la actividad, aparece un enlace a la práctica creada 'Análisis QAM' en la página principal del curso, tal y como se muestra en la imagen siguiente.

Figura 41: Portal Moodle donde se puede acceder a la nueva práctica JLab creada 'Análisis QAM'.

El enlace a la práctica es visible por el profesor y todos los alumnos del curso. Si pulsan el enlace 'Análisis QAM' se abre la página del curso. En caso de ser el profesor verá tres pestañas (Simulador, Resultados, Repositorio), tal y como se muestra en la magen siguiente.

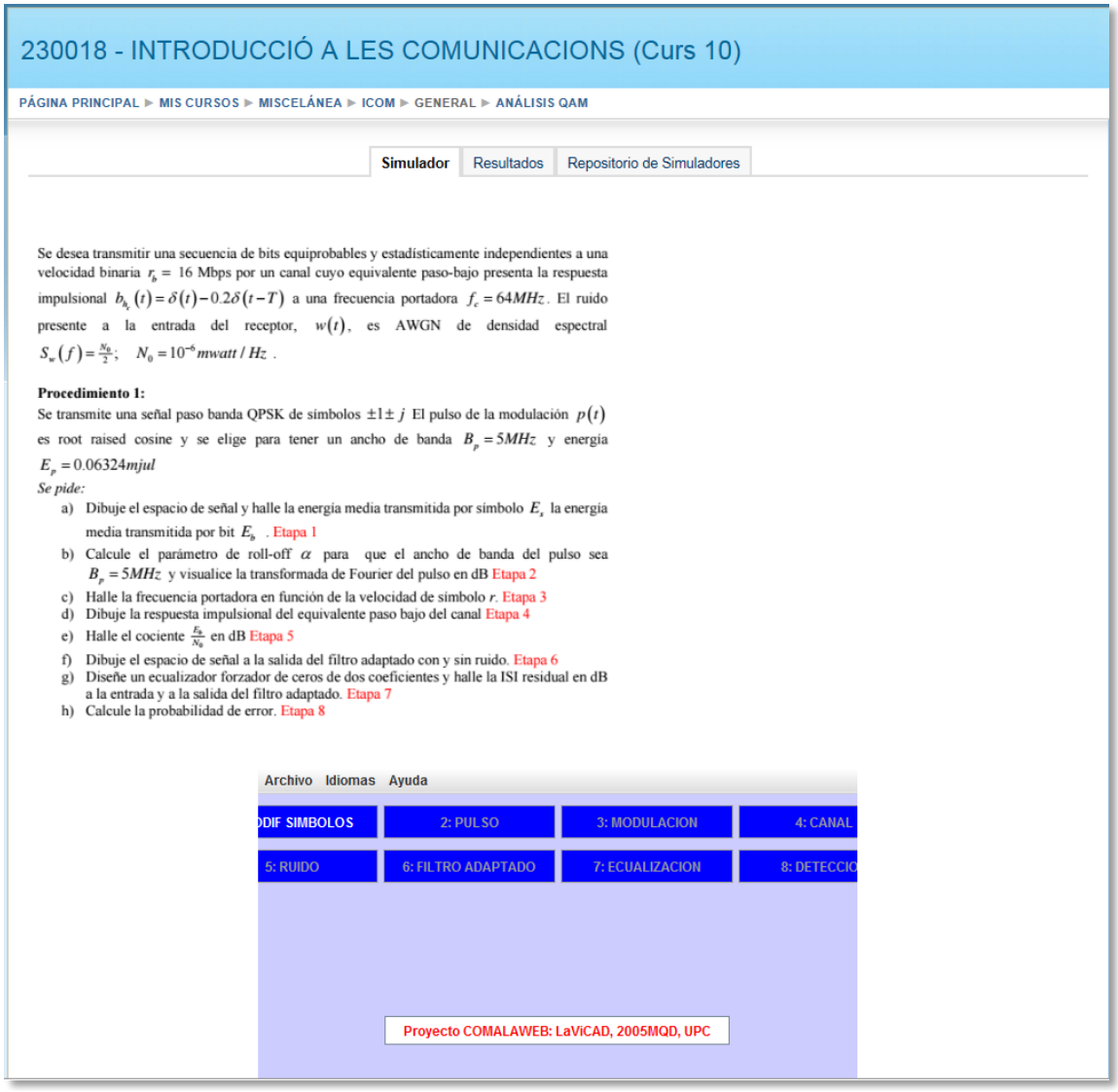


Figura 42: Enunciado de la práctica JLab usando el simulador QAM.

Veamos el enunciado de la práctica en detalle.

Se desea transmitir una secuencia de bits equiprobables y estadísticamente independientes a una velocidad binaria r_b= 16 Mbps por un canal cuyo equivalente paso-bajo presenta la respuesta impulsional b_{h_c} (t) = δ(t) - 0.2δ(t-T) a una frecuencia portadora f_c =64MHz. El ruido presente a la entrada del receptor, w(t), es AWGN de densidad

espectral $S_w(f) = \frac{N_0}{2}$; $N_0 = 10^{-6}\, mwatt/Hz$.

Procedimiento 1:

Se transmite una señal paso banda QPSK de símbolos $\pm 1 \pm j$. El pulso de la modulación $p(t)$es raised cosine y se elige para tener una ancho de banda $B_p = 5MHz$ y energía $E_p = 0.06324\, mjul$.

El sistema QAM implementado en el simulador LAVICAD se divide en las siguientes Etapas.

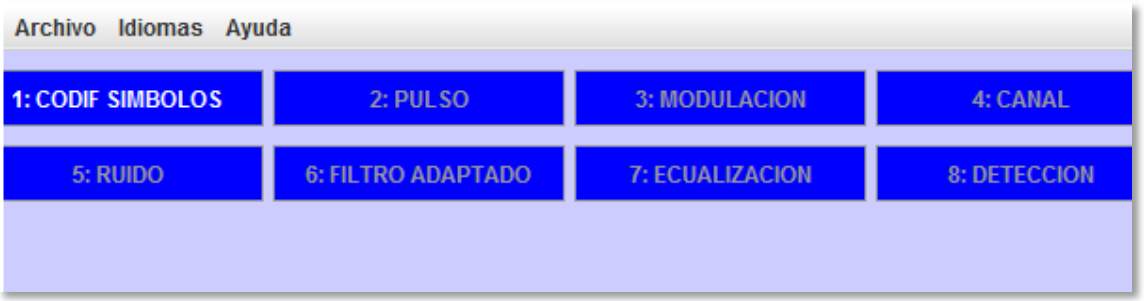


Figura 43: Etapas del simulador QAM implementado en el simulador LAVICAD.

- **Etapa 1**: Codificación de Símbolos
- **Etapa 2**: Conformación de pulso
- **Etapa 3**: Modulación I-Q
- **Etapa 4**: Convolución Canal
- **Etapa 5**: Suma de Ruido
- **Etapa 6**: Demodulación y filtro adaptado
- **Etapa 7**: Ecualización
- **Etapa 8**: Decodificación de símbolo

8.1.1 Etapa 1. Codificación de Símbolo

Esta etapa implementa la generación de la secuencia de bits y su posterior mapeo de señal para dar lugar a una secuencia de símbolos. Permite analizar los diferentes mapas de señal de las codificaciones propuestas, así como el comportamiento de la correlación de símbolos en comparación a su versión teórica.

Figura 44: Parámetros de entrada de la Etapa 1. Codificación de Símbolos del simulador QAM.

Parámetros de entrada

Grupo Generación de Bits:

- **Nbits**: Número de bits: fija la longitud de la señal binaria que se desea generar.

- **Bits**: Tipo de Generación: Permite elegir entre aleatoria o introducida por fichero.

- **Pbit0**: Probabilidad de bit 0: variable que determina la probabilidad de aparición del bit '0' en caso de utilizar generación aleatoria.

Grupo Codificación de Símbolos

- **Cod**.: Tipo de Codificación: Permite elegir entre: Polar, Unipolar, Bipolar, QAM, PSK (Phase Shift Keying), APSK (Amplitude and Phase Shift Keying).

- **Bits. Simb.**: Número de bits por símbolo: puede tomar valores entre 1 y 8. En QAM sólo admite valores pares.

- **Dist. Símbolo**: Distancia mínima entre símbolos.

Vamos a definir cada parámetro de entrada para cumplir con los datos del enunciado.

a) Dibuje el espacio de señal y halle la energía media transmitida por símbolo E_s y la energía media transmitida por bit E_b.

Bits: los dejamos a 1000.

Bits: Generación aleatoria ya que nos piden que sean estadísticamente independientes.

Pbit0: 0,5 para que sean bits equiprobables.

Para transmitir una señal paso banda QPSK de símbolos $\pm 1 \pm j$ debemos seleccionar la codificación QAM y 2 bits por símbolo.

Una vez configurados los parámetros indicados, pulsamos Validar y el simulador genera los siguientes resultados.

Resultados

Resultados Gráficos

- Espacio de señal transmitido. Muestra los puntos en la constelación que adquieren los símbolos transmitidos.

- Correlación de símbolos. Parte real de la correlación comparada con la correlación de símbolo teórica, mostrando una comparativa entre la versión teórica y la simulación de la correlación.

Resultados numéricos

- Energía de Símbolo $E_s = 0{,}5$

- Energía de bit $E_b = 0{,}25$ ya que codificamos cada símbolo con 2 bits.

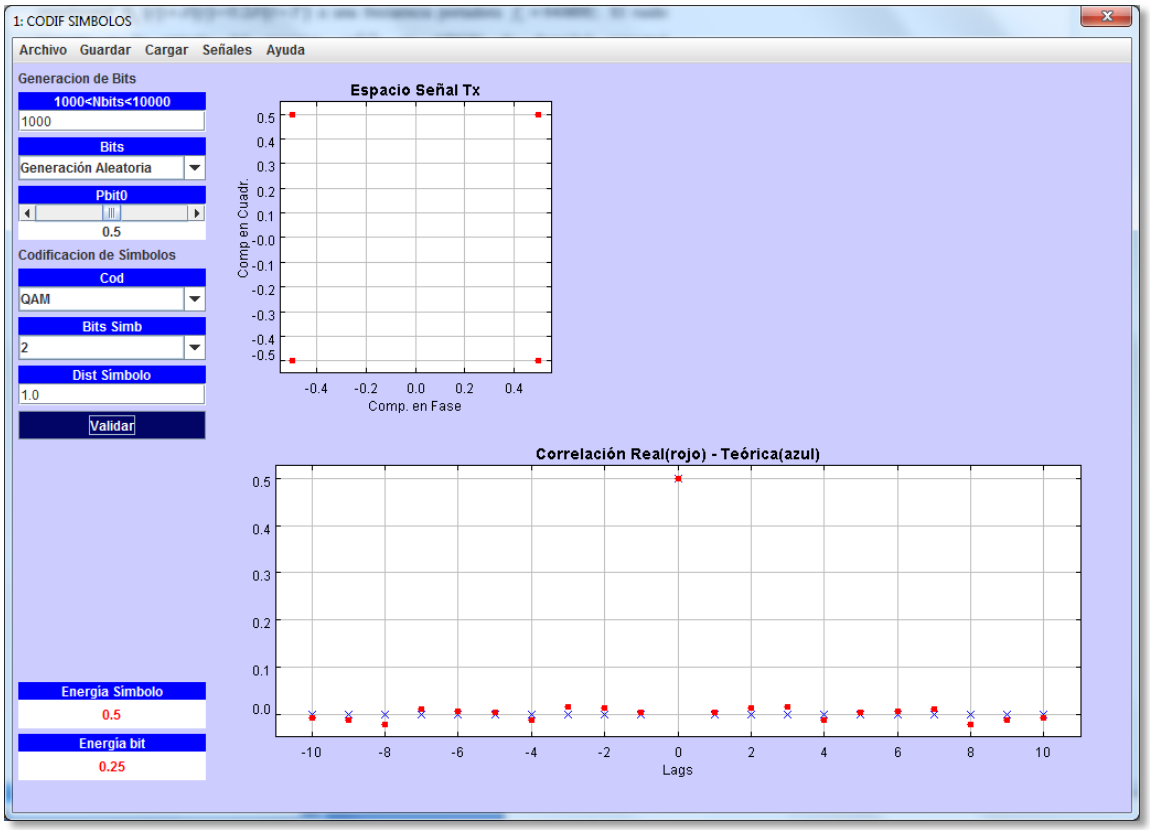


Figura 45: Resultados de la Etapa 1. Codificación de Símbolos del simulador QAM.

Una vez validada la primera etapa el simulador permite entrar en la siguiente etapa, el Conformador de Pulso.

Los resultados a enviar a Moodle son:

Etapa: CodSimbolo

Valor 1:Esimb:0.5

Valor 2:Ebit:0.25

8.1.2 Etapa 2. Conformación del Pulso

La segunda etapa del sistema se encarga de configurar el pulso que utilizaremos para la conformación de los símbolos, la operación conformación.

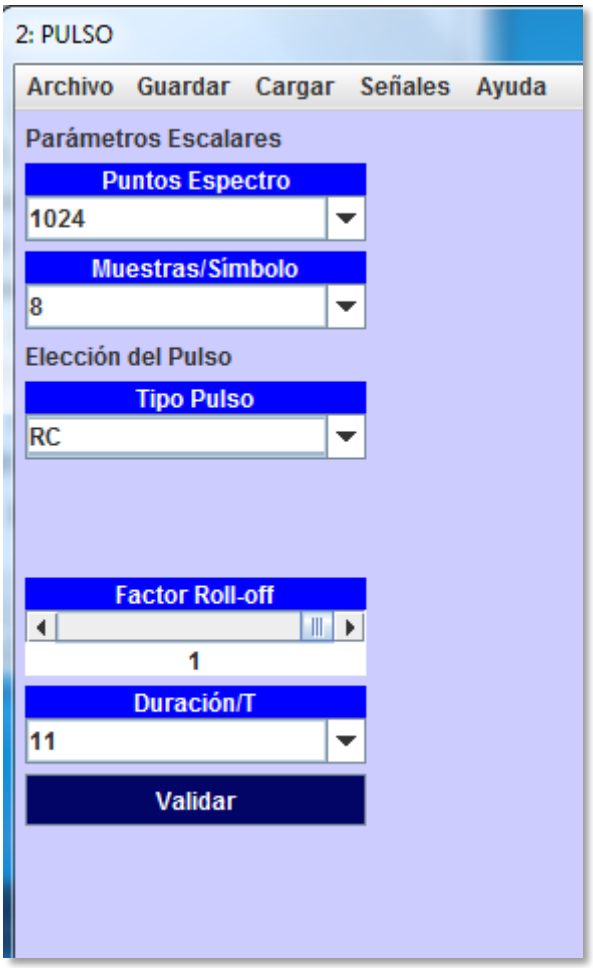


Figura 46: Parámetros de entrada de la Etapa 2. Conformación del Pulso del simulador QAM.

Parámetros de entrada

- **Puntos Espectro**: Número de puntos para el espectro de la nFFT. Esta variable determina el número de muestras que se utilizará en el cálculo de las FFT's para la representación espectral.

- **Muestras/Símbolo**: Número de muestras con el que representaremos cada símbolo, afecta directamente en la representación temporal del pulso y en la de la señal que contiene la información.

- Tipo de pulso: Desplegable donde se puede elegir RC (Raised Cosine), Rectangular o cualquier otro que introduzcamos por fichero.

- **Factor roll-off**: Parámetro del RC.

- **Duración/T**: Duración del pulso conformador en términos de tiempo de símbolo T_s.

El enunciado de la práctica pide:

b) Calcule el parámetro de roll-off α para que el ancho de banda del pulso sea $B_p = 5\,MHz$ y visualice **la transformada de Fourier del pulso en dB.**

$Ancho\ de\ bana\ del\ Raised\ Cosine: B_{p=\frac{r}{2}+\beta}$

$Factor\ de\ roll-off\ del\ Riased\ Cosine: \alpha = \frac{\beta}{r/2}$

A una velocidad de bit de 16 bps, teniendo en cuenta que cada símbolo son 2 bits, tenemos que $r = 8 \cdot 10^6\ simbolos/s$

Si el ancho de banda son 5 MHz, $\beta = 5 \cdot 10^6 - \frac{1}{2} \cdot 8 \cdot 10^6 = 10^6$

Por lo que el factor de roll-off será $\alpha = \frac{10^6}{\frac{1}{2} \cdot 8 \cdot 10^6} = 0{,}25$

Configuramos el pulso RC con un factor de roll-off de 0,25 y al pulsar Validar el simulador genera los siguientes resultados.

Resultados

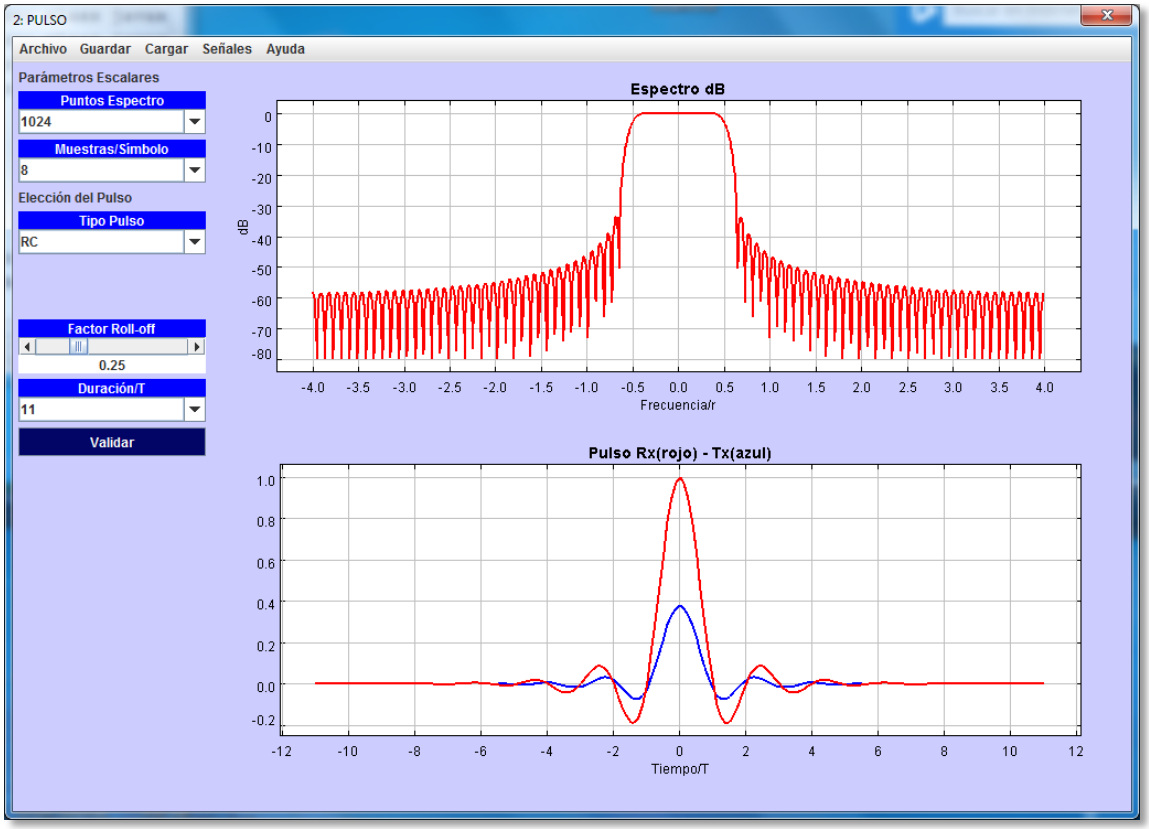


Figura 47: Resultado Etapa 2. Conformación de Pulso del simulador QAM.

- **Espectro en dB**: Muestra el espectro del pulso con la amplitud expresada en dB. Se calcula como el $10 \cdot \log_{10}$ del espectro lineal.

- **Pulso Transmitido y Recibido**: En la misma gráfica se muestra en azul el pulso transmitido que es la representación temporal del pulso diseñado en esta etapa para realizar la conformación; y en rojo el recibido que es la representación temporal del pulso recibido a la salida del filtro de recepción.

Esta etapa no tiene resultados numéricos que enviar, por lo que a Moodle le llegarán los valores de los resultados a vacios.

Etapa: ConfPulso

Valor 1:

Valor 2:

8.1.3 Etapa 3. Modulación I-Q

En esta etapa se realizan las operaciones de procesado para obtener la señal equivalente paso bajo compleja.

Parámetros de entrada

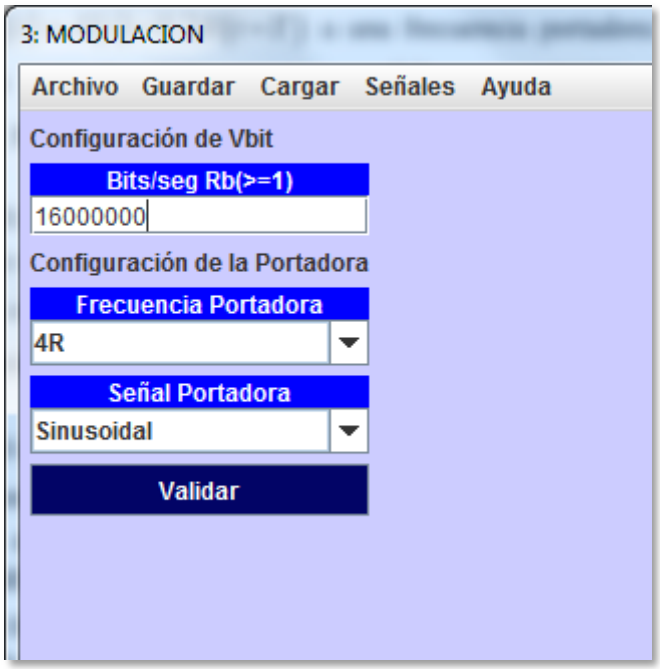


Figura 48: Parámetros de la Etapa 3. Modulación I-Q del simulador QAM.

- **Bits/seg**: Velocidad en bits por segundo.
- **Frecuencia portadora**: el simulador sólo admite 1, 2, 3 o 4 veces la velocidad de símbolo, o el valor nulo que implicaría modulación en

banda base. La velocidad de símbolo se calcula a partir de la velocidad de bit escogida en el parámetro anterior.

- **Señal portadora**: permite seleccionar la forma de onda de la portadora entre sinusoidal y rectangular.

El enunciado de la práctica pide:

c) Halle la frecuencia portadora en función de la velocidad de símbolo r.

En el enunciado se define $R_b = 16\,Mbps$ y una frecuencia portadora $f_c = 64\,MHz$.

Por lo que configuraremos la frecuencia portadora como ${}^{64}/_{16} = 4\,R$

Resultados

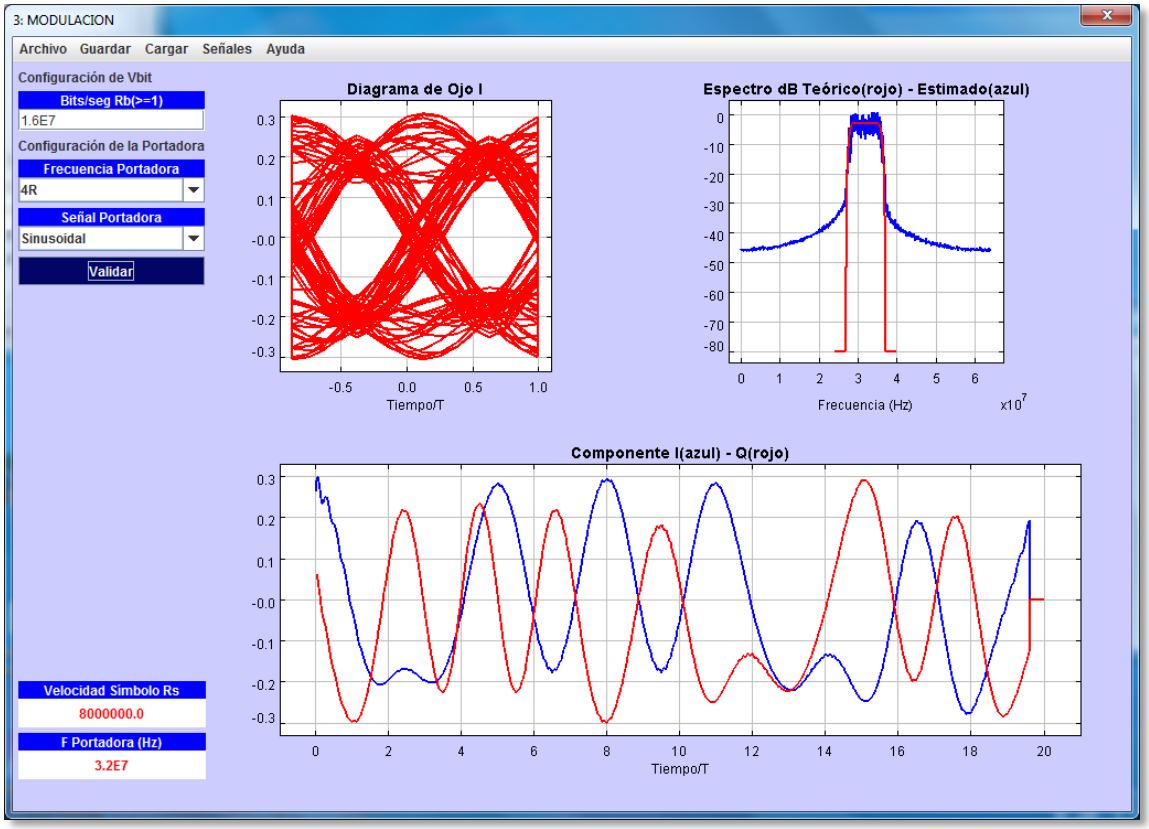


Figura 49: Resultados de la Etapa 3. Modulación I-Q del simulador QAM.

Gráficas

- Diagrama de ojo I: diagrama de ojo de la parte real de la señal paso bajo.

- Espectro dB: Comparación del espectro teórico y estimado en dB.

- **Componente I -Q** : se solapan la parte real de la señal paso bajo (azul) y la parte imaginaria (rojo).

Resultados numéricos

- Velocidad de símbolo

- Frecuencia portadora

Los resultados a enviar a Moodle son:

Etapa: IQ

Valor 1:Rsimb:8000000.0

Valor 2:frecPortadoraHz:3.2E7

8.1.4 Etapa 4. Convolución Canal

En esta etapa se implementa la convolución de la señal equivalente paso bajo resultante de la etapa anterior con la respuesta impulsional del canal seleccionado.

Parámetros de entrada

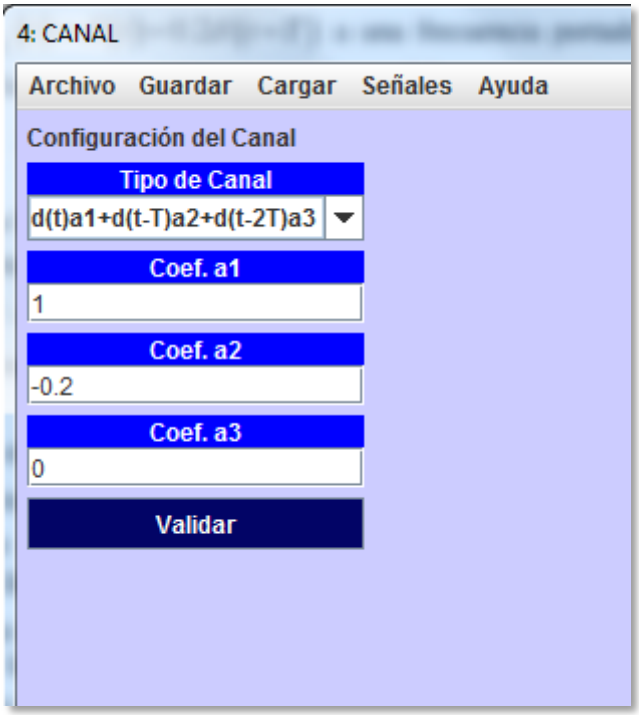


Figura 50: Parámetros de entrada de la Etapa 4. Canal del simulador QAM.

- **Tipo de canal**: Permite elegir entre un canal ideal y el canal con eco.

- **Coeficientes**: del canal con eco.
El enunciado de la práctica pide:

d) Dibuje la respuesta impulsional del equivalente paso bajo del canal.

El enunciado indica que el canal tiene la respuesta impulsional

$$b_{h_c} = \delta(t) - 0.2\delta(t-T)$$

Por lo que los coeficientes serán 1, -0.2 y 0.

Resultados

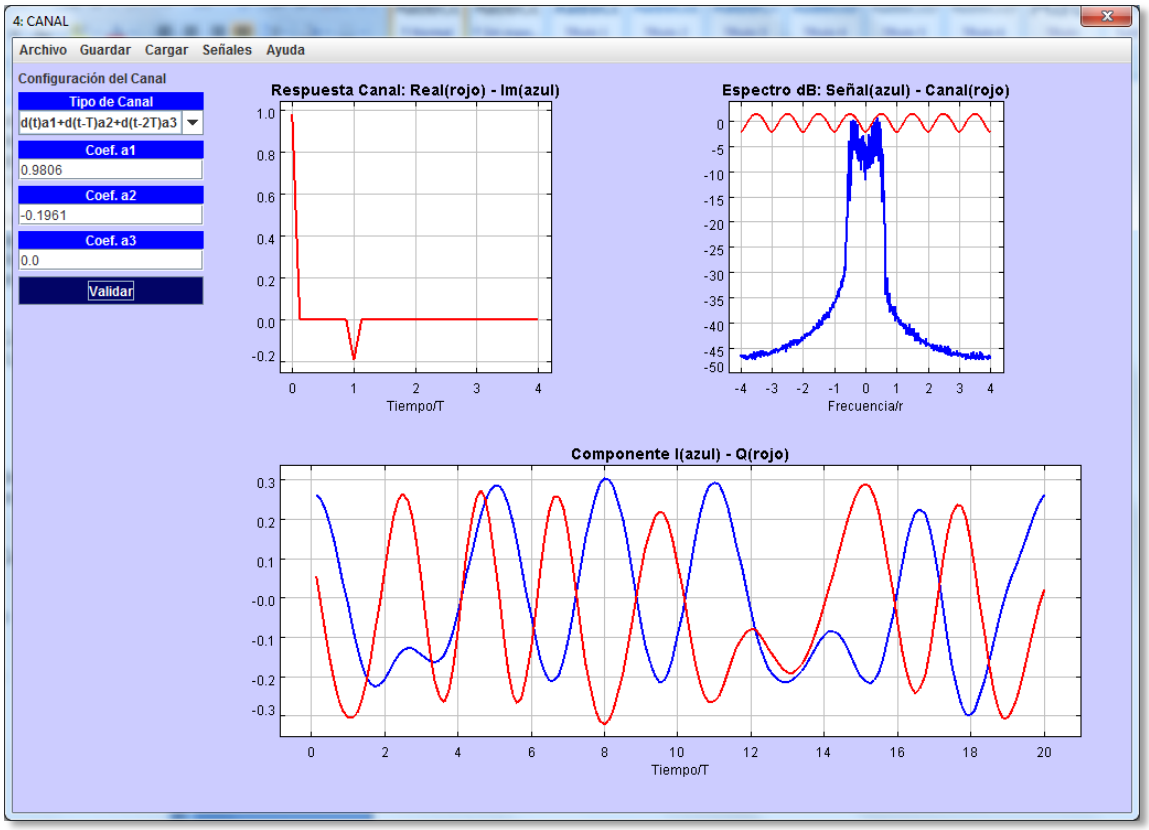


Figura 51: Resultados de la Etapa 4. Canal del simulador QAM.

- **Respuesta impulsional**: representación directa de la respuesta impulsional del canal seleccionado por el usuario.
- **Espectro en dB**: calculado como $10 \cdot log_{10}$del espectro lineal de la señal transmitida en azul y como $20 \cdot log_{10}$ de la función de transferencia del canal.
- **Señal I-Q**: representación de los 20 primeros símbolos de la señal paso bajo tras la convolución con el canal. En azul la parte real de la señal y en rojo la parte imaginaria de la señal.

Esta etapa no tiene resultados numéricos que enviar, por lo que a Moodle le llegarán los valores de los resultados a vacios.

Etapa: Convolucion

Valor 1:

Valor 2:

8.1.5 Etapa 5. Suma de Ruido

Esta etapa genera ruido con densidad de probabilidad gaussiana y densidad espectral blanca y lo suma a la señal.

Parámetros de entrada

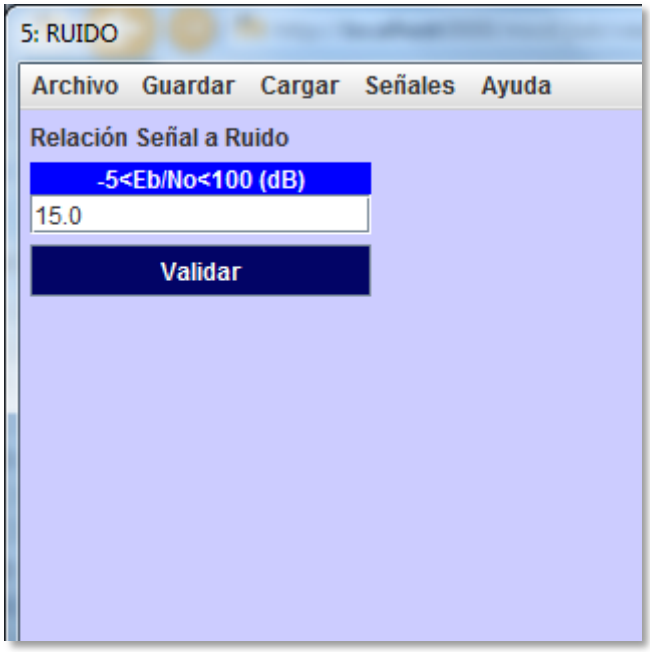


Figura 52: Parámetros de entrada de la Etapa 5. Suma de Ruido del simulador QAM.

- $^{E_b}/_{N_0}$: Es el cociente entre la energía media transmitida por bit (E_b) y la densidad espectral de ruido (N_0). Se trata del valor en dB de este parámetro. La energía de bit queda determinada en la primera etapa, mediante este valor estamos escogiendo el nivel de ruido que queremos simular.

El enunciado de la práctica pide:

d) Hallar la potencia de ruido para una $^{E_b}/_{N_0} = 15\ dB$.

$$E_b/N_0 = 10^{15/10} = 10^{1,5} \text{ jul}$$

$$\sigma_n^2 = \frac{1}{2}\frac{E_s}{k \cdot E_b/N_0} = 0{,}004 \text{ W} = -24{,}0309 \text{ dB}$$

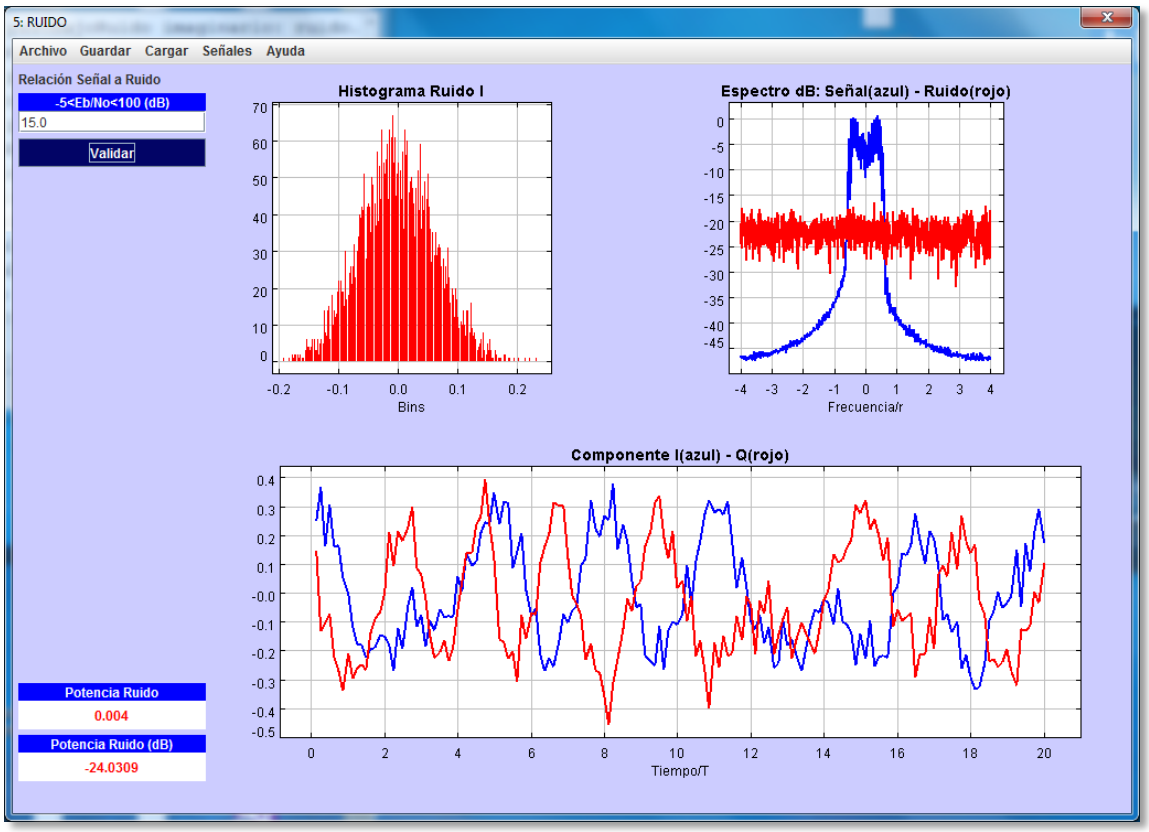


Figura 53: Resultado de la Etapa 5. Suma de Ruido del simulador QAM.

Resultados

Gráficas

- **Histograma Ruido I**: Histograma de la parte real de la señal de ruido generada.

- **Espectro en dB**: $10 \cdot log_{10}$ de la densidad espectral del ruido (rojo) y la señal antes de ser afectada por el ruido (azul).

- **Señal I-Q**: parte real (azul) y parte imaginaria de la señal.

Resultados numéricos

- Potencia de ruido: 0,004 W.

- **Potencia de ruido (dB):** -24,0309 dB.

Los resultados a enviar a Moodle son:

Etapa: Ruido

Valor 1:sigma2:0.004

Valor 2: sigma2dB:-24.0309

8.1.6 Etapa 6. Demodulación y Filtro Adaptado

En la primera etapa en recepción se simula el paso por el filtro adaptado y las operaciones para obtener la señal muestreada. Dado que el sistema trabaja con el equivalente paso bajo, el primer paso de la etapa: demodulador I - Q, se limita a aplicar posibles errores de portadora.

Parámetros de entrada

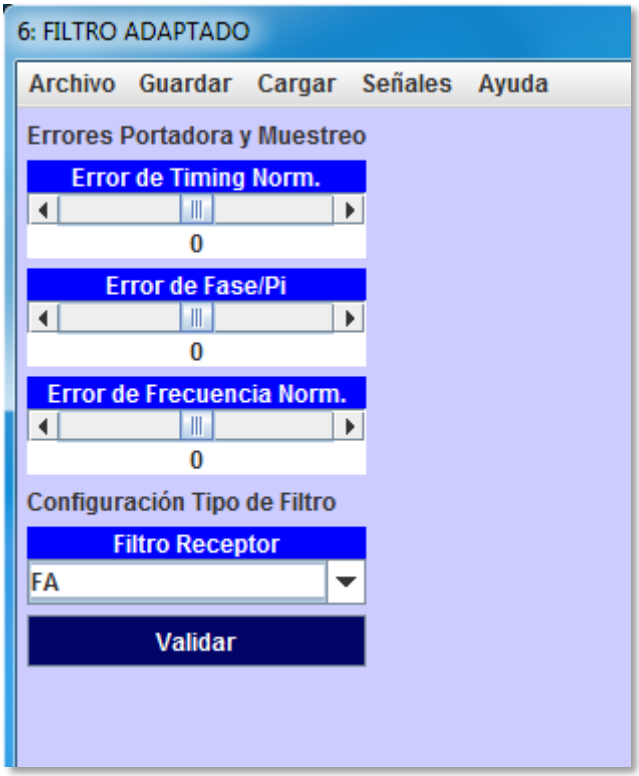


Figura 54: Parámetros de entrada de la Etapa 6. Filtro Adaptado del simulador QAM.

- **Error de muestreo normalizado**: error de muestreo añadido al instante óptimo de muestreo. Normalizado al periodo de muestreo.

- **Error de fase normalizado**: error de fase a introducir a la portadora en recepción. Normalizado a π.

- **Error de frecuencia normalizado**: error a añadir a la frecuencia de la portadora para demodular en recepción. Normalizado a la frecuencia de muestreo.

- **Filtro receptor**: permite seleccionar el diseño de un filtro adaptado o un filtro de respuesta impulsional arbitraria que el usuario introduce por fichero.

El enunciado de la práctica pide:

f) Dibuje el espacio de señal a la salida del filtro adaptado con y sin ruido.

El enunciado no indica nada de añadir errores de muestreo, así que dejamos todos los parámetros de entrada a 0.

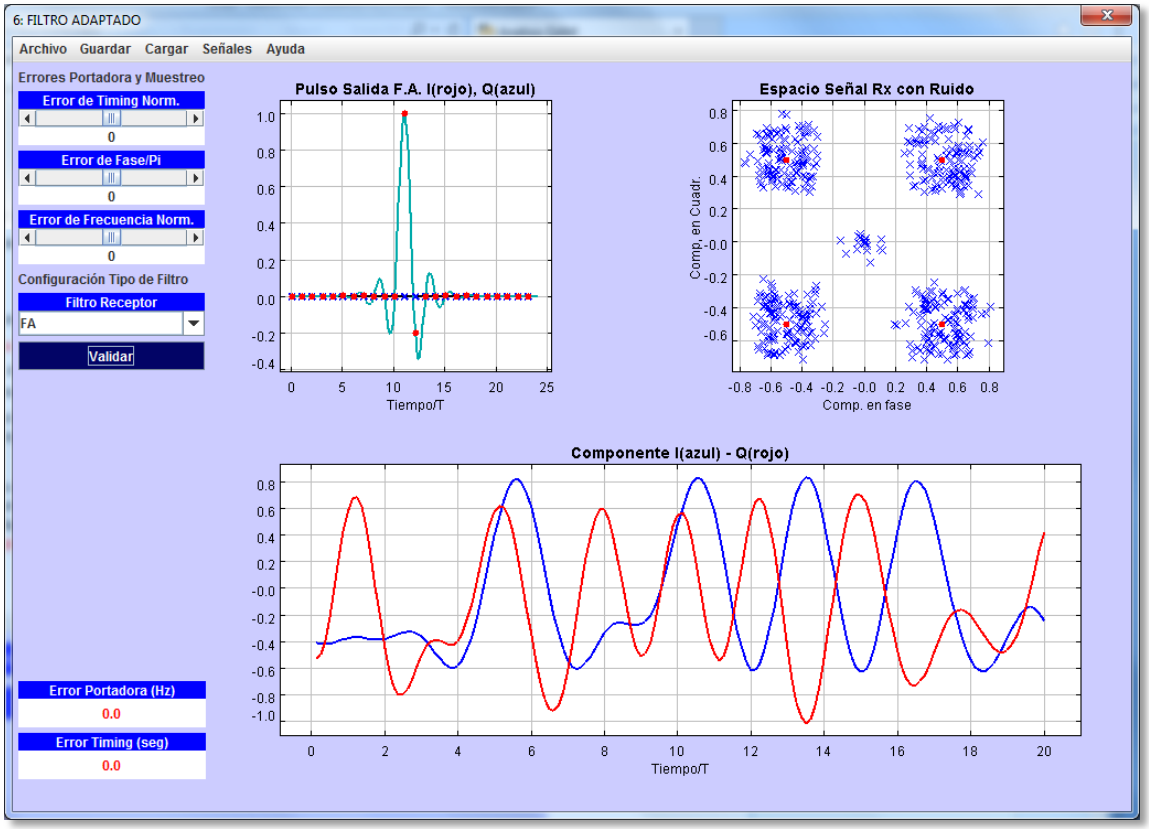


Figura 55: Resultados de la Etapa 6. Filtro Adaptado del simulador QAM. Con Ruido.

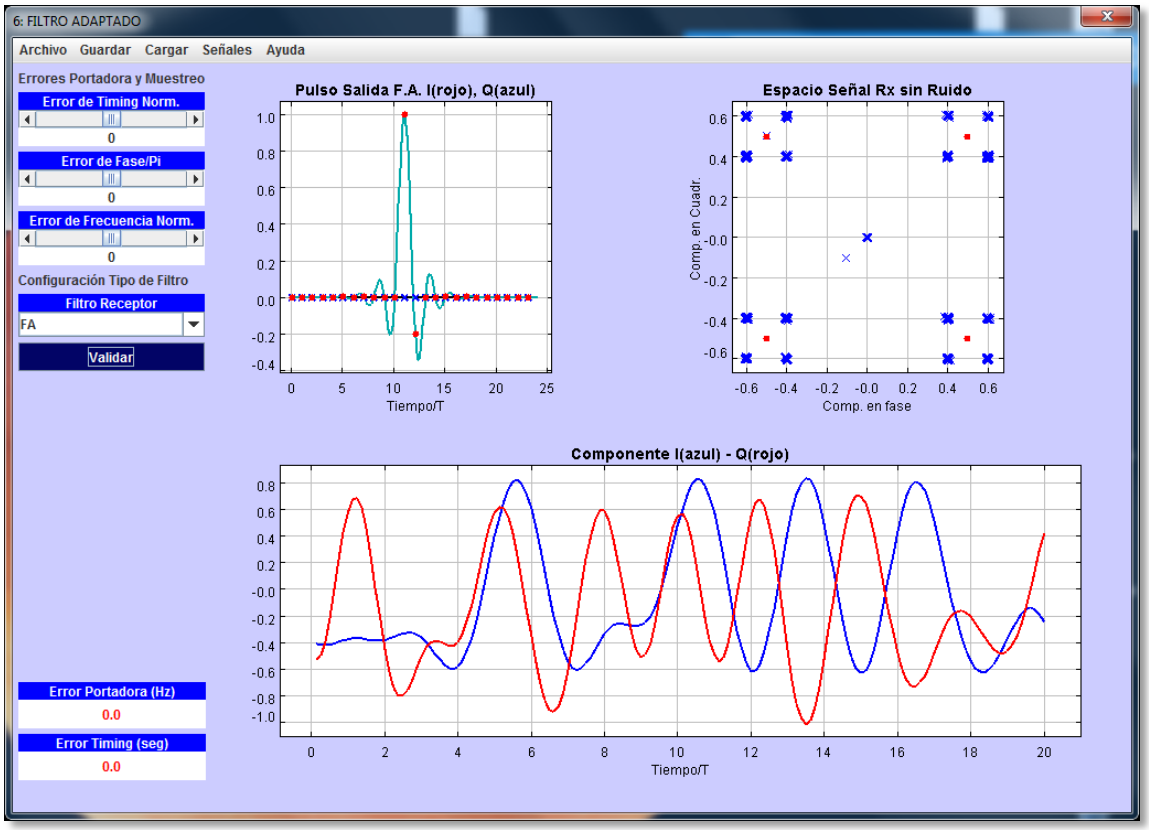


Figura 56: Resultados de la Etapa 6. Filtro Adaptado del simulador QAM. Sin Ruido.

Resultados

Gráficas

- Pulso a la salida del filtro adaptado: Parte real e imaginaria de la versión continua del pulso y la versión muestreda.

- Espacio de Señal Rx con ruido: representa las muestras de la señal a la salida del muestreador considerando también el ruido añadido en la etapa 5.

- Espacio de Señal Rx sin ruido: todas las muestras de la señal recibida a la salida de la etapa sin considerar el ruido añadido en la etapa 5

Resultados numéricos

- Error de portadora expresado en Hz.

- Error de timming expresado en segundos.

Ambos dan 0 porque no hemos añadido errores de muestreo.

Los resultados a enviar a Moodle son:

Etapa: FA

Valor 1: errorPortadora:0.0

Valor 2: errorTimingSeg:0.0

8.1.7 Etapa 7. Ecualización

Esta etapa se inicia con el diseño del filtro ecualizador a partir de los parámetros de entrada. Tras configurarlo, el procesado de la etapa se limita a convolucionar la señal que proviene del filtro adaptado con el ecualizador.

Parámetros de entrada

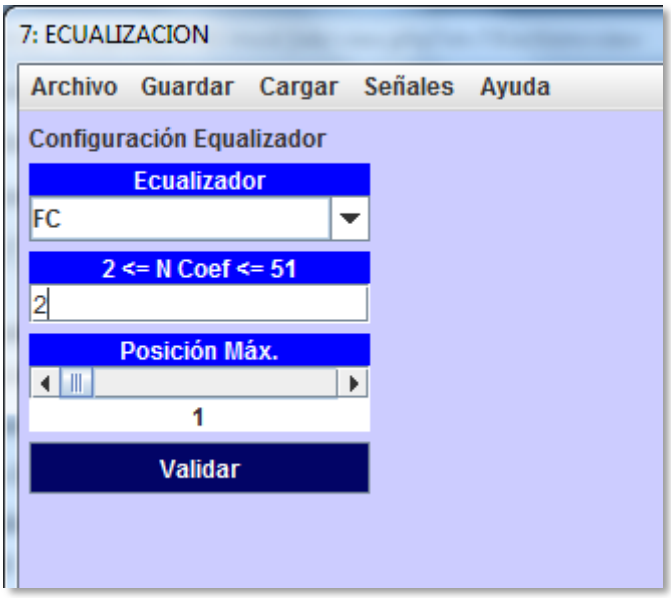


Figura 57: Parámetros de entrada de la Etapa 7. Ecualización del simulador QAM.

- **Tipo de ecualizador**: La variable ofrece cuatro opciones: no ecualizar, alternativa de especial interés cuando se trabaja con canal ideal; filtro forzador de ceros (FC), filtro de Wiener con criterio de mínimo error cuadrático medio (MMSE) y cualquier filtro de tipo FIR cuya respuesta impulsional que sea introducido por el usuario mediante fichero.
- **Número de coeficientes**: Se trata de la longitud del filtro FIR que forma el ecualizador, es un parámetro acotado entre 2 muestras y 51.
- **Posición del máximo**: Selecciona en qué posición se situará el máximo del pulso ecualizado.

El enunciado de la práctica pide:

g) Diseñe un ecualizador forzador de ceros de dos coeficientes y halle la ISI residual en dB a la entrada y a la salida del filtro adaptado.

Tipo de Ecualizador= FC (Forzador de ceros)

Número de coeficientes = 2

Resultados

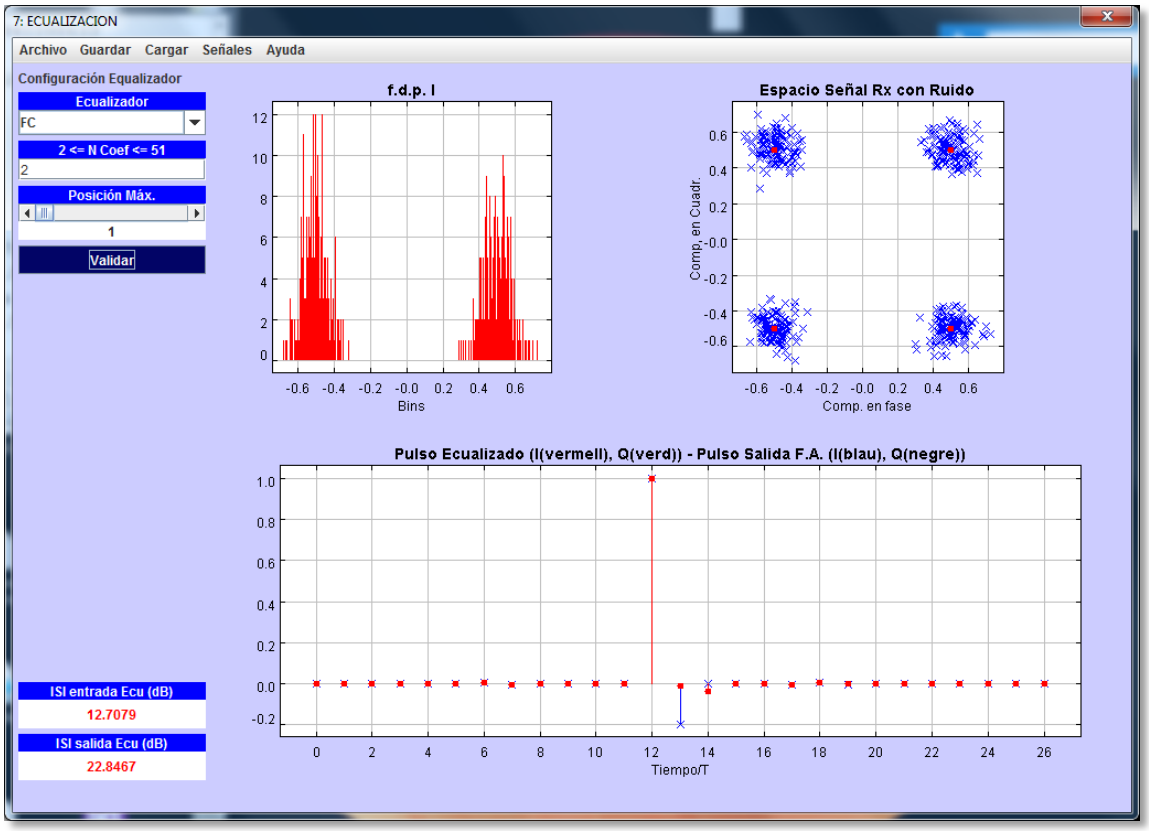


Figura 58: Resultado Etapa 7. Ecualización del simulador QAM.

- **Histograma teórico I**: se trata de la representación de la función de densidad de probabilidad teórica correspondiente a la parte real de dicha señal.

- **Mapa de señal con ruido**: representa las muestras de la señal a la salida del ecualizador considerando también el ruido añadido en la etapa 5.

- **Pulso a la salida** del filtro adaptado comparado con el ecualizado, en escala lineal: compara el pulso equivalente a la entrada y salida del ecualizador.

Resultados numéricos

- ISI en dB a la salida del filtro adaptado

- ISI en dB a la salida del ecualizador

Los resultados a enviar a Moodle son:

Etapa: Ecualización

Valor 1: ISIFa:12.7079

Valor 2: ISIEqu:22.8467

8.1.8 Etapa 8. Detección

La detección de los símbolos es la última etapa de este sistema. En esta etapa se realizan las siguientes operaciones:

- Detección de los símbolos que provienen del ecualizador

- El demapping de los símbolos que consiste en traducir los símbolos en bits a partir de la asignación inicial que se ha dado en transmisión.

- Cálculo de la probabilidad de error a nivel de símbolo y de bit

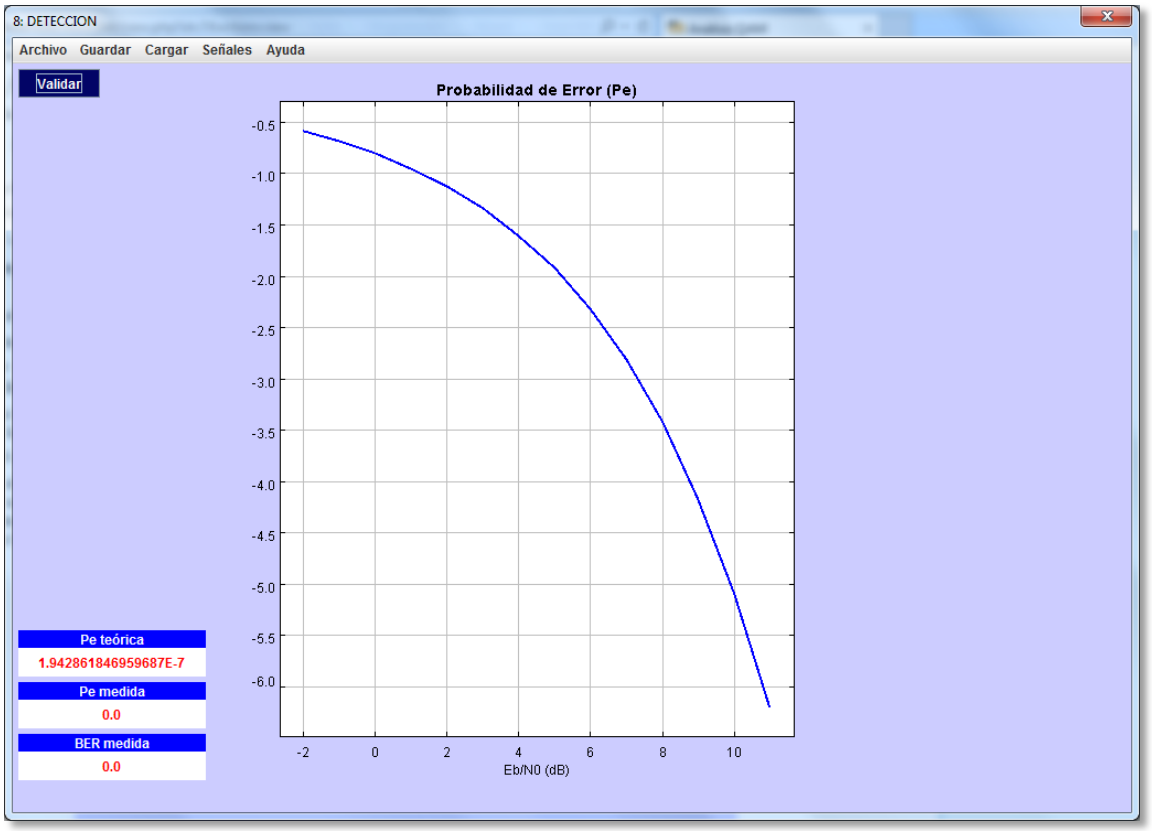


Figura 59: Resultados Etapa 8. Detección, última etapa del simulador QAM.

Gráficas

- Probabilidad de error teórica: relación teórica entre el cociente de potencias
E_b / N_0 (dB) y el $10 \cdot \log_{10}$ de la probabilidad de error. Sobre esta curva se dibuja un punto que representa la probabilidad de error medida para la E_b / N_0 concreta seleccionada en la etapa 5. Se proporciona así una comparación entre la probabilidad de error obtenida a partir de la simulación con la gráfica teórica.

Resultados numéricos

- Probabilidad de error teórica: muestra la probabilidad de error de símbolo calculada de forma teórica.

- Probabilidad de error medida: es la probabilidad de error de símbolo medida.

- BER medida: se trata de la probabilidad de error de bit medida

Los resultados a enviar a Moodle son:

Resultados etapa 8:

Pe teórica = 1,942·E-7

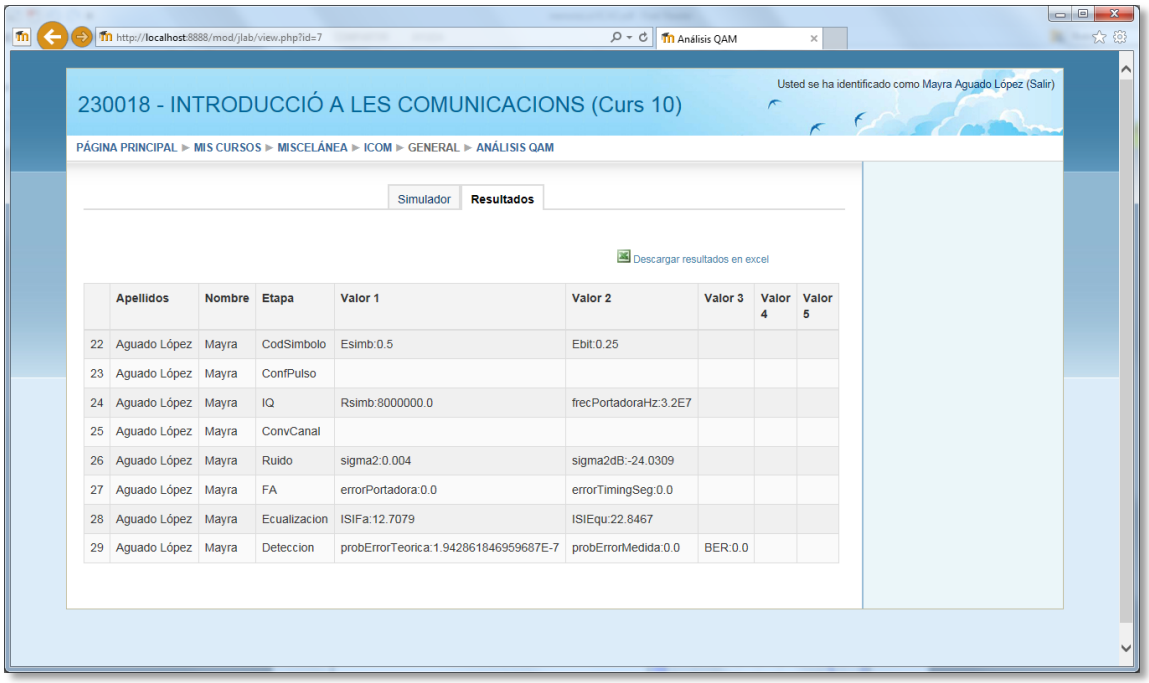
Pe media = 0

BER media = 0

8.2 Resultados recibidos en Moodle

Los resultados de cada etapa que se han ido enviando a Moodle se pueden visualizar desde la actividad 'Análisis QAM' en la pestaña 'Resultados'.

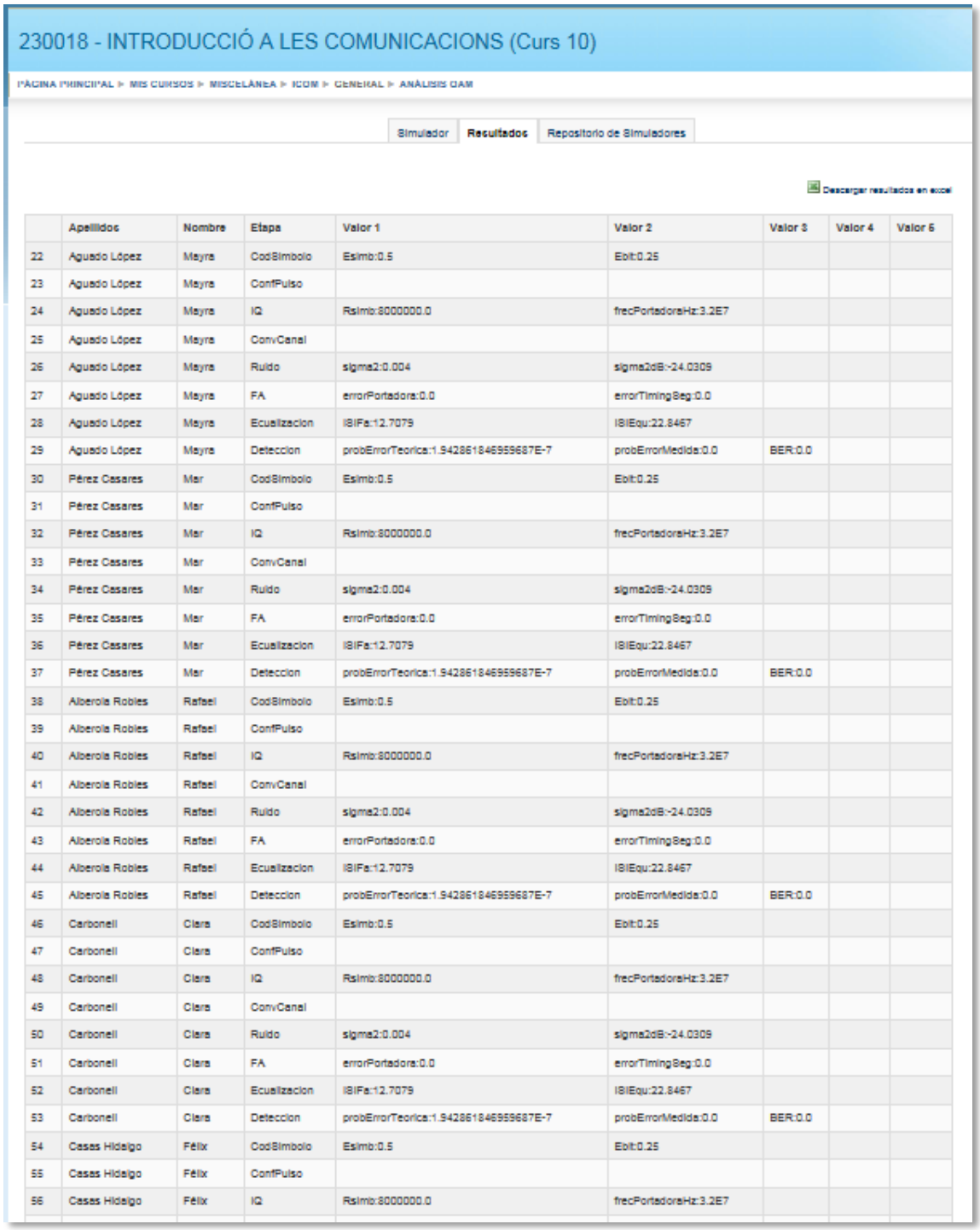
Los alumnos sólo verán los resultados de la última simulación que ellos mismo han realizado, tal y como se muestra en la siguiente imagen.



	Apellidos	Nombre	Etapa	Valor 1	Valor 2	Valor 3	Valor 4	Valor 5
22	Aguado López	Mayra	CodSimbolo	Esimb:0.5	Ebit:0.25			
23	Aguado López	Mayra	ConfPulso					
24	Aguado López	Mayra	IQ	Rsimb:8000000.0	frecPortadoraHz:3.2E7			
25	Aguado López	Mayra	ConvCanal					
26	Aguado López	Mayra	Ruido	sigma2:0.004	sigma2dB:-24.0309			
27	Aguado López	Mayra	FA	errorPortadora:0.0	errorTimingSeg:0.0			
28	Aguado López	Mayra	Ecualizacion	ISIFa:12.7079	ISIEqu:22.8467			
29	Aguado López	Mayra	Deteccion	probErrorTeorica:1.9428618469596 87E-7	probErrorMedida:0.0	BER:0.0		

Figura 60: Resultados simulación QAM vista por un alumno.

En cambio, el profesor verá los resultados de todos los alumnos, tal y como se muestra en la siguiente imagen.



230018 - INTRODUCCIÓ A LES COMUNICACIONS (Curs 10)

PÁGINA PRINCIPAL ▸ MIS CURSOS ▸ MISCELÁNEA ▸ ICOM ▸ GENERAL ▸ ANÁLISIS QAM

Simulador | Resultados | Repositorio de Simuladores

Descargar resultados en excel

	Apellidos	Nombre	Etapa	Valor 1	Valor 2	Valor 3	Valor 4	Valor 5
22	Aguado López	Mayra	CodSimbolo	Esimb:0.5	Ebit:0.25			
23	Aguado López	Mayra	ConfPulso					
24	Aguado López	Mayra	IQ	Rsimb:8000000.0	frecPortadoraHz:3.2E7			
25	Aguado López	Mayra	ConvCanal					
26	Aguado López	Mayra	Ruido	sigma2:0.004	sigma2dB:-24.0309			
27	Aguado López	Mayra	FA	errorPortadora:0.0	errorTimingSeg:0.0			
28	Aguado López	Mayra	Ecualizacion	IBIFa:12.7079	IBIEqu:22.8467			
29	Aguado López	Mayra	Deteccion	probErrorTeorica:1.9428618469596 87E-7	probErrorMedida:0.0	BER:0.0		
30	Pérez Casares	Mar	CodSimbolo	Esimb:0.5	Ebit:0.25			
31	Pérez Casares	Mar	ConfPulso					
32	Pérez Casares	Mar	IQ	Rsimb:8000000.0	frecPortadoraHz:3.2E7			
33	Pérez Casares	Mar	ConvCanal					
34	Pérez Casares	Mar	Ruido	sigma2:0.004	sigma2dB:-24.0309			
35	Pérez Casares	Mar	FA	errorPortadora:0.0	errorTimingSeg:0.0			
36	Pérez Casares	Mar	Ecualizacion	IBIFa:12.7079	IBIEqu:22.8467			
37	Pérez Casares	Mar	Deteccion	probErrorTeorica:1.9428618469596 87E-7	probErrorMedida:0.0	BER:0.0		
38	Alberola Robles	Rafael	CodSimbolo	Esimb:0.5	Ebit:0.25			
39	Alberola Robles	Rafael	ConfPulso					
40	Alberola Robles	Rafael	IQ	Rsimb:8000000.0	frecPortadoraHz:3.2E7			
41	Alberola Robles	Rafael	ConvCanal					
42	Alberola Robles	Rafael	Ruido	sigma2:0.004	sigma2dB:-24.0309			
43	Alberola Robles	Rafael	FA	errorPortadora:0.0	errorTimingSeg:0.0			
44	Alberola Robles	Rafael	Ecualizacion	IBIFa:12.7079	IBIEqu:22.8467			
45	Alberola Robles	Rafael	Deteccion	probErrorTeorica:1.9428618469596 87E-7	probErrorMedida:0.0	BER:0.0		
46	Carbonell	Clara	CodSimbolo	Esimb:0.5	Ebit:0.25			
47	Carbonell	Clara	ConfPulso					
48	Carbonell	Clara	IQ	Rsimb:8000000.0	frecPortadoraHz:3.2E7			
49	Carbonell	Clara	ConvCanal					
50	Carbonell	Clara	Ruido	sigma2:0.004	sigma2dB:-24.0309			
51	Carbonell	Clara	FA	errorPortadora:0.0	errorTimingSeg:0.0			
52	Carbonell	Clara	Ecualizacion	IBIFa:12.7079	IBIEqu:22.8467			
53	Carbonell	Clara	Deteccion	probErrorTeorica:1.9428618469596 87E-7	probErrorMedida:0.0	BER:0.0		
54	Casas Hidalgo	Félix	CodSimbolo	Esimb:0.5	Ebit:0.25			
55	Casas Hidalgo	Félix	ConfPulso					
56	Casas Hidalgo	Félix	IQ	Rsimb:8000000.0	frecPortadoraHz:3.2E7			

Figura 61: Resultados simulación QAM vista por el profesor.

9. Conclusiones

Partiendo de la idea de adaptar las prácticas de laboratorio de comunicaciones a las nuevas tecnologías de la información, se ha desarrollado el presente proyecto que aúna los simuladores de comunicaciones LAVICAD implementados con applets de java y el portal educativo Moodle, creando el módulo de actividades de Moodle JLab.

La gran aportación de JLab es permitir que los resultados obtenidos por los alumnos en experimentos virtuales lleguen al profesor a través del portal Moodle.

El hecho de ser un módulo estándar de Moodle permite una instalación sencilla y una integración completa en portales Moodle 1.9 o superior. JLab añade funcionalidad a Moodle permitiendo a los profesores ofrecer un nuevo tipo de actividades a sus alumnos, en este caso, actividades en las que podrán configurar y simular un sistema de comunicaciones viendo las señales de entrada y salida como si estuvieran en un laboratorio real.

Existen varios sistemas de comunicación implementados con el sistema LAVICAD que se pueden añadir al Repositorio de Simuladores. El Repositorio de Simuladores ofrece un espacio compartido por todos los profesores dentro del propio Moodle donde se pueden ir añadiendo nuevos simuladores.

JLab permite a los profesores tener control de las simulaciones que se proponen a los estudiantes en todo momento, pudiendo indicar si se quieren almacenar los resultados de las simulaciones o no y entre qué periodos de fechas se quieren almacenar. Pasada la fecha límite indicada JLab ya no almacenará ningún resultado enviado.

La consulta de los resultados también se realiza desde el propio Moodle. Los profesores pueden ver para cada alumno los datos configurados en cada etapa del sistema de comunicación. Con lo que podrán evaluar el grado de comprensión del alumno.

Desde el punto de vista del alumno, JLab ofrece un laboratorio virtual accesible desde cualquier ordenador en cualquier momento donde poder experimentar los conceptos teóricos jugando con los parámetros de las etapas de los simuladores y viendo los resultados obtenidos.

Por lo que creemos se han cubierto los objetivos a conseguir y se ha creado una herramienta que esperemos motive tanto a alumnos como a profesores a utilizarla como apoyo en sus clases de comunicaciones y que sirva para fomentar el desarrollo de nuevos

simuladores LAVICAD y tener un abanico más amplio de sistemas con los que experimentar.

10. Referencias

Bibliografía

[1] Dr. Pere Marquès Graells."Impacto de las TIC en la enseñanza universitaria". 2000. Deprt. Pedagogía Aplicada, Facultad de Educación, UAB.http://ddd.uab.cat/pub/dim/16993748n11a5.pdf

[2] Portal del Proyecto RIMA. "Recerca i Innovació en Metodologies de l'Aprenentatge". http://www.upc.edu/rima/

[3] Portal del Grupo GILABVIR. http://www.upc.edu/rima/grups/gilabvir

[4] Proyectos participantes en el Grupo GILABVIR. http://www.upc.edu/rima/grupos/gilabvir-grup0-de-interes-laboratorios-virtuales-y-remotos/proyectos-participantes

[5] Portal ATENEA, Campus Virtual de la UPC. http://atenea.upc.edu/

[6] William Rice. "Moodle 1.9 E-learning Course Development". Ed. Packt Publishing. Junio 2008.

[7] Jonathan Moore, Michael Churchward. "Moodle 1.9 Extension Development". Packt Publishing. Abril 2010.

[8] Alex Buchner. "Moodle Administration". Ed. Packt Publising. Septiembre 2008.

[9] Robin Nixon. "Learning PHP, MySQL & Javascript". O'Reilly.Diciembre 2014.

[10] Elizabeth Sugar Boese. "An introduction to Programming with Java Applets". Ed. Jones and Barlett. Abril 2009.

[11] Dan Pilone, Neil Pitman. "UML 2.0 in a Nutshell". O'Reilly. 2005.

[12] Portal del proyecto LAVICAD. http://comweb.upc.edu/

[13] Portal del Proyecto iLabViR. "Integración Laboratorios Virtuales y Remotos". http://ilabvir.upc.edu/

[14] Xavier Giró, Silvia Cortés. Margarita Cabrera Beán, Miguel Ángel Farré, Pedro-Christian Espinosa Fricke - "Comunicación Contenedor java-moodle" Dpto. de Teoria del Senyal i Comunicacions. UPC. Abril 2009.

[15] Ignatius Teo . Librería PHP para la exportación de datos a Excel. Enero de 2005.

[16] John G.Proakis. "Digital Communications". McGraw-Hill. 2001