UNIVERSIDAD DE CONCEPCION
Facultad de Ingeniería
Departamento de Ingeniería Informática
Y Ciencias de la Computación

Patrocinante: Jorge López Reguera
Comisión: César González Castillo
Yussef Farrán Leiva

CREACIÓN DE API PARA GESTIÓN DE PROBLEMAS Y ADMINISTRACIÓN DE AULA VIRTUAL PARA PLATAFORMA DE EVALUACIÓN AUTOMÁTICA DE CURSOS DE PROGRAMACIÓN

MAURICIO DAVID ECHAVARRÍA ARANEDA

Informe de Memoria de Título
Para optar al Título de

Ingeniero Civil Informático

Octubre 2016

Sumario

El presente proyecto consiste en la construcción de una mejora a la plataforma de apoyo al aprendizaje con evaluación automática que posee el Departamento de Ingeniería Informática y Ciencias de la Computación de la Universidad de Concepción. Se abordará la perspectiva del docente dentro de la plataforma. El proyecto se divide en dos partes: una API para gestión de problemas y una interfaz de administración de aula virtual.

La API de gestión de problemas consiste en una interfaz web que permite la creación y edición de los problemas con los que cuenta la plataforma. Se crearon cuatro editores: uno para probadores estáticos, uno para probadores dinámicos, uno para los enunciados y uno para soluciones de referencia.

La interfaz de administración de aula virtual consiste en una interfaz web que permite agregar alumnos, crear cursos, crear actividades, asociar alumnos a cursos, asociar actividades a cursos, y obtener calificaciones de los alumnos.

Para este proyecto se utilizó una modificación de la metodología de desarrollo de tipo ágil SCRUM. Esto debido a la presencia de un único desarrollador en el equipo de trabajo. Este modelo es utilizado normalmente en presencia de un equipo de desarrollo compuesto por varias personas, donde se asignan roles específicos. En este caso, el único miembro asumió todos los roles.

El producto final es una máquina virtual que contiene la plataforma completa más la API de gestión de problemas y la interfaz de administración del aula virtual.

Contenido

Capítulo 1: Introducción 7
1.1 Contextualización 8
1.2 Objetivo general 10
1.3 Objetivos específicos 10
1.4 Recursos necesarios 10
1.5 Entregable 11
Capítulo 2: Análisis del problema 12
2.1 Análisis de la situacion actual de la plataforma 13
2.2 Metodología de desarrollo 15
Capítulo 3: Diseño API para Edición de problemas 17
3.1 Introducción 18
3.2 Requisitos funcionales de la API de edición de problemas 18
3.3 Diagrama de casos de uso de la API de edición de problemas 19
3.4 Casos de uso de la API de edición de problemas 20
3.5 Base de datos para almacenamiento de problemas 21
3.5 Modelo MER de la base de datos de problemas 22
3.6 Mockups de la API de edición de problemas 23
Capítulo 4: Desarrollo API para Edición de problemas 27
4.1 Introducción 28
4.2 Lista de problemas 28
4.3 Barra de selección de los editores 29
4.4 Editor de probadores dinámicos 29
4.5 Editor de probadores estáticos 30
4.6 Editor de enunciados 30
4.7 Editor de soluciones de referencia 31
4.8 Programa PHP de guardado 31
4.9 Programas PHP de opciones para las soluciones 32
Capítulo 5: Diseño de Ambiente de Administración del Aula Virtual 33
5.1 Introducción 34
5.2 Requisitos funcionales del ambiente de administración 34
5.3 Diagrama de casos de uso del ambiente de administración 35

5.4 Casos de uso del ambiente de administración 36
5.5 Mockups del ambiente de administración 38
Capítulo 6: Desarrollo de Ambiente de Administración de Aula Virtual 43
6.1 Introducción 44
6.2 Vista inicial del ambiente de administración 45
6.3 Creación y manejo de actividades 45
6.4 Gestión de alumnos 46
6.5 Gestión de cursos 46
6.6 Estilo CSS 47
Capítulo 7: Pruebas y Resultados 48
7.1 Pruebas técnicas de Software 49
7.2 Comparativa entre el producto final y el funcionamiento previo 49
7.3 Validación del producto 51
Conclusiones 53
Referencias 55
Anexo 1: Casos de uso de la plataforma de administración de problemas 56
Anexo 2: Vistas reales de la API de edición de problemas. 57
Anexo 3: Casos de uso de la plataforma de administración del aula virtual 62
Anexo 4: Vistas reales del ambiente de gestión de la plataforma de evaluación. 64
Anexo 5: Errores técnicos encontrados durante las pruebas de software. 69

Imágenes

Figura 2.1 Vista expandida de la carpeta de problemas 13
Figura 3.1 Diagrama de casos de uso de la API de administración de problemas 19
Figura 3.2 Modelo MER de la base de datos para almacenar los problemas 22
Figura 3.3 Mockup de la vista del navegador de problemas 23
Figura 3.4 Mockup de la vista de la barra superior presente en los editores 23
Figura 3.5 Mockup de la vista del editor dinámico simplificado 24
Figura 3.6 Mockup de la vista del editor dinámico avanzado 24
Figura 3.7 Mockup de la vista de un editor de probador estático 25
Figura 3.8 Mockup de la vista del editor de programa solución de referencia 25
Figura 3.9 Mockup de la vista del editor de enunciado 26
Figura 5.1 Diagrama de casos de uso de la plataforma de administración del aula virtual 35
Figura 5.2 Mockup de la vista general de la interfaz de administración del aula virtual 38
Figura 5.3 Mockup de la vista de la opción Crear Actividad 39
Figura 5.4 Mockup de la vista de la opción Ver Actividades 39
Figura 5.5 Mockup de la vista del detalle de una actividad 40
Figura 5.6 Mockup de la vista de la opción Agregar Alumnos 40
Figura 5.7 Mockup de la vista de la lista de cursos 41
Figura 5.8 Mockup de la vista de detalle de un curso 41
Figura 5.9 Mockup de la vista para agregar alumnos a un curso 42
Figura 7.1 Vista de la carpeta de problemas del sistema anterior 50
Figura 22.1 Vista real de la lista de problemas 57
Figura 22.2 Vista real del editor básico de probadores dinámicos 58
Figura 22.3 Vista real del editor completo de probadores dinámicos 59
Figura 22.4 Vista real del editor de probadores estáticos 60
Figura 22.5 Vista real del editor de enunciados 60
Figura 22.6 Vista real del editor de soluciones 61
Figura 24.1 Vista real inicial del ambiente de administración del aula virtual 64
Figura 24.2 Vista real de la lista de actividades 64
Figura 24.3 Vista real del detalle de una actividad 65
Figura 24.4 Primera vista real de crear actividad 65
Figura 24.5 Segunda vista real de crear actividad 65

Figura 24.6 Vista real de la lista de alumnos 66
Figura 24.7 Vista real del detalle de un alumno 66
Figura 24.8 Vista real de agregar alumnos 67
Figura 24.9 Vista real de la lista de cursos 67
Figura 24.10 Vista real del detalle de un curso 68
Figura 24.11 Vista real de agregar alumnos a un curso 68
Figura 25.1 Capturas de pantallas del error de dimensión en pantallas pequeñas. 70

Tablas

Tabla 4.1 resumen de los archivos que componen la API de gestión de problemas 28
Tabla 6.1 resumen con los archivos que componen el ambiente de administración de aula virtual 44

Capítulo 1: Introducción

1.1 Contextualización

El Departamento de Ingeniería Informática de la Universidad de Concepción cuenta con una plataforma de apoyo al aprendizaje con evaluación automática para cursos de programación. Esta plataforma se diseñó para ser usada en aula y evaluar a un grupo de alumnos en particular, y también se encuentra disponible vía Internet para que cualquier estudiante la utilice.

El objetivo de la plataforma es apoyar el aprendizaje del alumno presentándole una serie de problemas de programación, luego éste debe generar un programa solución, que es evaluado por un centro de evaluación computarizado de dos formas: una estática y una dinámica.

La prueba estática, consiste en analizar el código fuente del estudiante, y determinar si cumple con los estándares del lenguaje en uso, además se analiza la forma en que se resuelve un problema, como por ejemplo si se imprime un conjunto de líneas repetitivas haciendo o no uso de ciclos iterativos.

La prueba dinámica se basa en el concepto de *Testing de software* [1], ya que se hacen pruebas en tiempo de ejecución al software creado por los estudiantes, y determina si lo que realiza lo hace en forma correcta o errónea.

Según Ammann y Offutt [1], un *ingeniero de pruebas* es un profesional de las tecnologías de la información que está a cargo de una o más actividades de prueba técnicas, incluyendo diseño de entradas de prueba, producir valores para los casos de prueba, analizar resultados y reportarlos a los desarrolladores y administradores. En el caso de esta plataforma, el *ingeniero de pruebas* corresponde a la prueba dinámica que se realiza al software desarrollado por el alumno.

También los autores indican que cualquier ingeniero involucrado en el desarrollo de un software debería saber que en algún momento tendrá que “ponerse el sombrero” de un ingeniero de pruebas, por lo que esta plataforma también busca que el alumno pruebe su software por su cuenta antes de solicitar la revisión computarizada.

El apoyo al aprendizaje busca varios objetivos, tanto para el estudiante como para el profesor. Por el lado del alumno, se intenta hacer que ejercite con múltiples problemas, a la vez que evalúa el modelo mental del alumno para retroalimentarlo con información personalizada. Mientras que al docente le permite calificar al alumno y capturar información acerca de nuevos modelos mentales.

La principal característica de este sistema es su orientación a la docencia didáctica [2], por lo que sus problemas son relativamente pequeños, es decir, que pueden ser resueltos en poco tiempo. Además, tiene otras como:

- La complejidad de los problemas es incremental, de modo que el alumno empieza ejercitando con el más simple y termina desarrollando el más complejo.
- Entrega retroalimentación personalizada al alumno.
- Permite al docente la detección de modelos mentales erróneos, entregándole un patrón común, permitiendo que en el uso futuro estos errores sean orientados automáticamente hacia lo correcto.

El uso de la plataforma se realiza desde dos perspectivas: una desde el docente y otra desde el alumno. La perspectiva del docente es la siguiente:

- Al iniciar el semestre debe agregar al sistema la lista de alumnos, y luego crear un curso al cual asignarle dichos estudiantes.
- Para realizar cada sesión el docente debe crearla para luego asignarle los problemas deseados de los que ya existen o bien crear nuevos problemas.
- Al iniciar la clase, el docente debe lanzar la sesión de evaluación.
- Durante la sesión, se muestra al docente el comportamiento general del curso, los problemas que han resuelto en la sesión, las calificaciones obtenidas y los casos de estudiantes que se encuentren en dificultades
- Al finalizar la clase, el docente debe detener la sesión en el sistema.

La perspectiva del alumno se desarrolla de la siguiente manera:

- Los alumnos inician sesión en la plataforma ingresando matricula y curso.
- Se presenta al estudiante una serie de problemas que deben ser resueltos. El alumno elige uno del listado, lo resuelve y luego sube la solución al sistema.
- La plataforma retroalimenta al alumno indicándole las metas obtenidas, una descripción de sus errores y lo orienta hacia la solución correcta.
- Se continua resolviendo todos los problemas del listado hasta completarlos todos o hasta que se termine el tiempo determinado por el docente.

Esta plataforma, desde un principio fue desarrollada para ser amigable con el usuario estudiante, pero para el usuario docente este aspecto se ha ido postergado. Los principales aspectos no desarrollados o posibles de mejora son los siguientes:

- Agregar un nuevo problema significa crear casi desde cero un probador estático y uno dinámico, debido a que no existe ningún ambiente de reutilización de probadores.
- Crear un probador estático, actualmente, implica crear una serie de expresiones regulares que permitan detectar errores a nivel de lenguaje.
- Crear un probador dinámico implica determinar casos de prueba que logren hacer que el programa falle para así detectar todos los posibles errores que puedan ocurrir en el tiempo de ejecución *[1].*
- Se deben identificar automáticamente los posibles modelos mentales erróneos en los que puedan caer los alumnos durante el desarrollo de cada problema [2].

- Se deben crear "*minitest*" para orientar a los alumnos en la búsqueda de la solución.
- La administración del aula virtual se hace directamente en MySQL, lo cual obliga al docente a conocer la estructura de la base de datos
- Todos los problemas están almacenados como archivos y para modificar un problema, hay que buscar la carpeta del problema y luego editar directamente los archivos contenidos en ella
- Para iniciar una actividad, hay que crearla previamente en MySQL y luego lanzarla conociendo el ID de ella.

En resumen, es necesario conocer a fondo el sistema para usarlo correctamente. En efecto, los docentes lo han hecho evitan agregar nuevos problemas dada la complejidad que significa administrarlo.

1.2 Objetivo general

Diseñar y construir las funcionalidades necesarias para dotar a la plataforma de evaluación automática de un ambiente de gestión de nuevos problemas, la edición de los ya existentes, y la administración del aula virtual para cursos de programación, de modo que pueda ser usado por cualquier docente sin que deba conocer los detalles de implementación

1.3 Objetivos específicos

- Crear una API para producir problemas de programación, probadores estáticos y probadores dinámicos.
- Construir una interfaz que elimine la necesidad de trabajar en lenguajes de implementación de la plataforma al gestionar el aula virtual.
- Generar las condiciones necesarias para que la plataforma sea más adaptativa, permitiendo la clasificación de problemas por criterios tales como tema o complejidad

1.4 Recursos necesarios

Considerando que la plataforma actualmente existe como una máquina virtual basada en Linux, MySQL y PHP, se necesitará como principal recurso una máquina que haga de servidor para la plataforma. Esta máquina tendrá que tener instalada alguna distribución servidora de Linux como Ubuntu Server. Además tendrá que tener el servidor web Apache, PHP, MySQL, phpMyAdmin, GCC, JRE, FLEX, JACC.

Al ser una plataforma en la que los alumnos trabajan remotamente, se requiere un entorno LAN que permita la conexión entre el servidor y las máquinas que utilizan los alumnos.

1.5 Entregable

Como producto entregable, se tendrá un archivo de máquina virtual, exportado mediante el software de virtualización VirtualBox, que contendrá la plataforma instalada más el desarrollo de este proyecto. Esta máquina podrá ser instalada en cualquier computador que tenga instalado VirtualBox y que posea al menos 2GB de memoria RAM y 8GB de espacio en disco disponible. Estos requerimientos de sistema se justifican de la siguiente manera:

- Una máquina virtual Ubuntu con entorno gráfico necesita al menos 512MB [4] de memoria RAM para funcionar correctamente, y adicionalmente otros 512MB para la plataforma de evaluación automática, lo que suma un total de 1GB. A esto se debe adicionar la memoria necesaria por el sistema host, la cual en general corresponde a 1GB
- La máquina virtual que contiene la plataforma de evaluación automática, una vez instalada, tiene un peso de 5.13GB, mientras que el archivo de importación de la máquina utiliza 2.32GB extras. Esta suma da 7.45GB, que es el mínimo necesario y se da durante la instalación, ya que una vez instalada se puede remover el archivo importado.

Capítulo 2: Análisis del problema

2.1 Análisis de la situacion actual de la plataforma

Los problemas se encuentran almacenados como archivos ubicados en carpetas, como se muestra en la figura 2.1. Cada problema cuenta con dos programas probadores: uno escrito en Shell de Linux que cumple el rol de probador estático y otro en lenguaje C que actúa como probador dinámico. Posee también un archivo de texto plano que contiene el enunciado y otro con el nombre del problema. Finalmente existen uno o varios programas en lenguaje C que representan soluciones al problema. El número identificador del problema corresponde al nombre de la carpeta que contiene sus archivos.

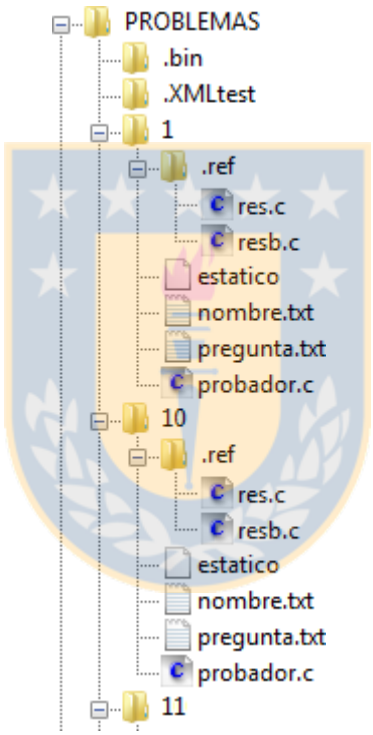

Figura 2.1 Vista expandida de la carpeta de problemas

Un probador estático está compuesto por dos fragmentos de código. El primero de ellos muestra al estudiante una comparación entre la salida de su programa y la salida esperada. El segundo fragmento analiza la forma en que esa salida fue generada.

Para mostrar al alumno la comparación de salidas, el probador genera código HTML utilizando un archivo de salida generado por el programa del estudiante, y otro archivo de salida generado por la solución de referencia.

Para analizar la forma en que el programa del alumno generó la salida, el probador analiza el código en busca de ciertas expresiones como ciclos “*for*” o funciones como

"*printf"* para determinar, por ejemplo, si el alumno generó una secuencia de números usando ciclos iterativos o mediante el abuso de sentencias "*printf"*.

Un probador dinámico es un programa que ejecuta la solución del alumno y supervisa que su ejecución no produzca errores fatales, además de generar un registro de la salida de dicho programa. También crea una salida de referencia utilizando el programa solución almacenado en el sistema. Si el problema requiere alguna entrada para poder generar la salida, ésta es generada también por el probador dinámico.

Existe una gran variedad de probadores dinámicos según requiere cada problema. Algunos incluyen funciones adicionales, y otros tienen más de una meta.

De los programas solución de cada problema, uno representa una solución correcta y el resto son soluciones de prueba utilizadas para encontrar algún error típico en el software del alumno que haya sido detectado en el pasado para el problema en particular Estas diversas soluciones representan los variados casos de pruebas que debería usar un Ingeniero de Testing [1]. Es importante destacar que cada problema tiene exactamente un archivo con la solución correcta, y puede o no contener soluciones con errores típicos. En caso de existir más de una solución, es necesario analizar cada una de ellas para determinar cuál es la correcta.

Actualmente, para crear un nuevo problema se debe navegar entre las diversas carpetas para encontrar alguno similar a lo que el usuario desee crear. Luego se debe seleccionar y copiar los archivos en una nueva carpeta. Finalmente se editan los archivos de los editores estático y dinámico, además del nombre, enunciado y el o los programas solución de referencia.

La base de datos que actualmente posee el sistema se enfoca principalmente en almacenar la información del aula virtual. Esto es, almacenar actividades, alumnos, calificaciones, soluciones generadas por los estudiantes, entre otros. Esta base de datos no cuenta con ningún tipo de documentación, por lo que el modelo MER es inexistente dentro de ésta. Debido a esto, resulta tedioso realizar un seguimiento de su estructura.

El manejo del aula virtual se realiza de diversas maneras, dependiendo de la acción que se desee realizar:

- Para ingresar nuevos estudiantes a la base de datos, se debe ejecutar la siguiente instrucción SQL en el motor de base de datos,

```
INSERT INTO `aulaVirtual`.`alumno` (`nmat`,`nombre`)
        VALUES ('00', 'aa'), ('01', 'bb')...;
```

- Para crear una actividad, se debe insertar en la tabla "actividad" una nueva entrada, para luego asignarle problemas. Estas relaciones se almacenan en la tabla "problema_actividad".

- Para lanzar una actividad, existe un programa en Shell, el cual dado un ID de actividad, desactiva cualquier otra que se encuentre en curso, inicia una nueva con el ID indicado y le asigna un *"timestamp"*. Este programa debe ser ejecutado desde la terminal de Linux, debido a que el ID de actividad se entrega como un argumento.
- La obtención de notas se realiza desde la interfaz de reporte al docente, desde donde se obtiene una captura de pantalla, o bien se guarda la página web tal como se encontraba al momento de finalizar la sesión. Estos datos son también almacenados en la tabla calificaciones de la base de datos, y para obtenerlas se requiere la siguiente consulta SQL:

```
SELECT * FROM `calificaciones` WHERE

`actividad` = [N°_de_actividad];
```

2.2 Metodología de desarrollo

Para poder establecer una metodología de desarrollo, primero se debe caracterizar el proyecto. Para ello, los aspectos claves a considerar son los siguientes:

- El equipo de trabajo solo cuenta con una persona, que será la encargada de diseñar y desarrollar el proyecto (el autor de este informe).
- Se trabajará sobre un sistema ya existente, formado por dos perspectivas distintas mencionadas en la Introducción, y de las cuales el desarrollador conoce solo una (la del alumno), siendo inexperto en la segunda que es el foco de este proyecto.
- El producto final debe ser un prototipo funcional y no solo un diseño conceptual.
- Se proponen pasos como toma de requerimientos, diseño conceptual, implementación, e integración de los productos a desarrollar.

En base a esto, se puede descartar algunas metodologías de desarrollo:

- RUP no es aplicable dado que la documentación orientada a intercomunicación dentro del equipo no tiene sentido al ser un único desarrollador.
- El método ágil XP tampoco es aplicable porque éste propone programación por pares.

Se presentan ciertas aproximaciones a algunos de los modelos existentes:

- Modelo en Cascada, el cual propone una linealidad de pasos para conseguir el objetivo final, entre los cuales se encuentran los pasos propuestos en este proyecto.
- Modelo Scrum, que propone un esquema iterativo e incremental, además de asignar roles a los integrantes del equipo de trabajo.

Si se considera el modelo Scrum por sí solo, se debe tener en cuenta lo siguiente:

- Se proponen roles, pero como se dijo, el equipo de trabajo se compone de solo una persona. Sin embargo, esta única persona podría ser capaz de desarrollar todas las tareas que propone el modelo.
- La tarea de *Scrum Master* la desempeñaría sobre sí mismo, por lo que no tendría sentido hablar de este cargo.
- El *stakeholder* es necesario para un método ágil como Scrum, pero el desarrollador puede tomar este cargo sin ningún problema.
- Los *sprints* quizá se definan un poco largos, porque vendrían a ser dos, uno para la API de gestión de problemas y otro para la plataforma de administración del aula virtual.

Si se considera el modelo en cascada por sí solo existe la probabilidad de que durante el desarrollo de la plataforma de administración de aula virtual se detecte un error en el diseño o la programación de la API de gestión de problemas, lo cual implicaría un rediseño y reprogramación, lo cual tiene un alto costo.

Se concluye que la metodología a seguir es una adaptación del modelo Scrum, a la cual se le incorporan los pasos del diseño en cascada, además de definir todos los cargos sobre el único desarrollador. Esto no es nuevo, pues el ingeniero informático Oscar Herrera realiza una adaptación similar a la hecha en este proyecto en su memoria de título [3].

Capítulo 3: Diseño API para Edición de problemas

3.1 Introducción

El primer objetivo del presente trabajo consiste en diseñar una API que le permita al usuario docente la gestión de los problemas de la plataforma de apoyo al aprendizaje con evaluación automática. Esta deberá permitir la creación y edición de problemas de manera amigable, y el almacenamiento de ellos en una base de datos para así eliminar la necesidad de tener que trabajar con archivos y carpetas.

3.2 Requisitos funcionales de la API de edición de problemas

Se han establecido los siguientes requisitos funcionales para la plataforma de creación y edición de problemas:

- Una ventana que permita seleccionar si se desea crear un problema nuevo o editar un problema existente. También debe permitir ver la lista completa de problemas.
- Dos editores de probadores, uno para estáticos y otro para dinámicos. Estos deben contar con un cuadro donde se pueda ver y editar el código fuente de cada uno, además de un panel que muestre filtros para agregarlos al código.
- Un editor para generador de E/S de referencia, que debe contar con un cuadro donde se pueda ver y editar el código fuente. No requiere del panel de filtros mencionado anteriormente dado que el generador de E/S de referencia es una alternativa de solución al problema que puede ser codificado fácilmente por el docente que utilice el sistema.
- Un editor para enunciados de problemas, que debe contar con un panel de texto que permita su escritura.
- Una lista completa de problemas para navegar en ella.
- Permitir el rápido cambio entre cada editor.

3.3 Diagrama de casos de uso de la API de edición de problemas

En base a los requisitos anteriores, se generó el diagrama de casos de uso de la figura 3.1. En él se puede apreciar que el docente tendrá dos opciones básicas: crear un nuevo problema y editar un problema existente. En ambos casos, podrá crear o editar cualquiera de las 4 componentes de un problema.

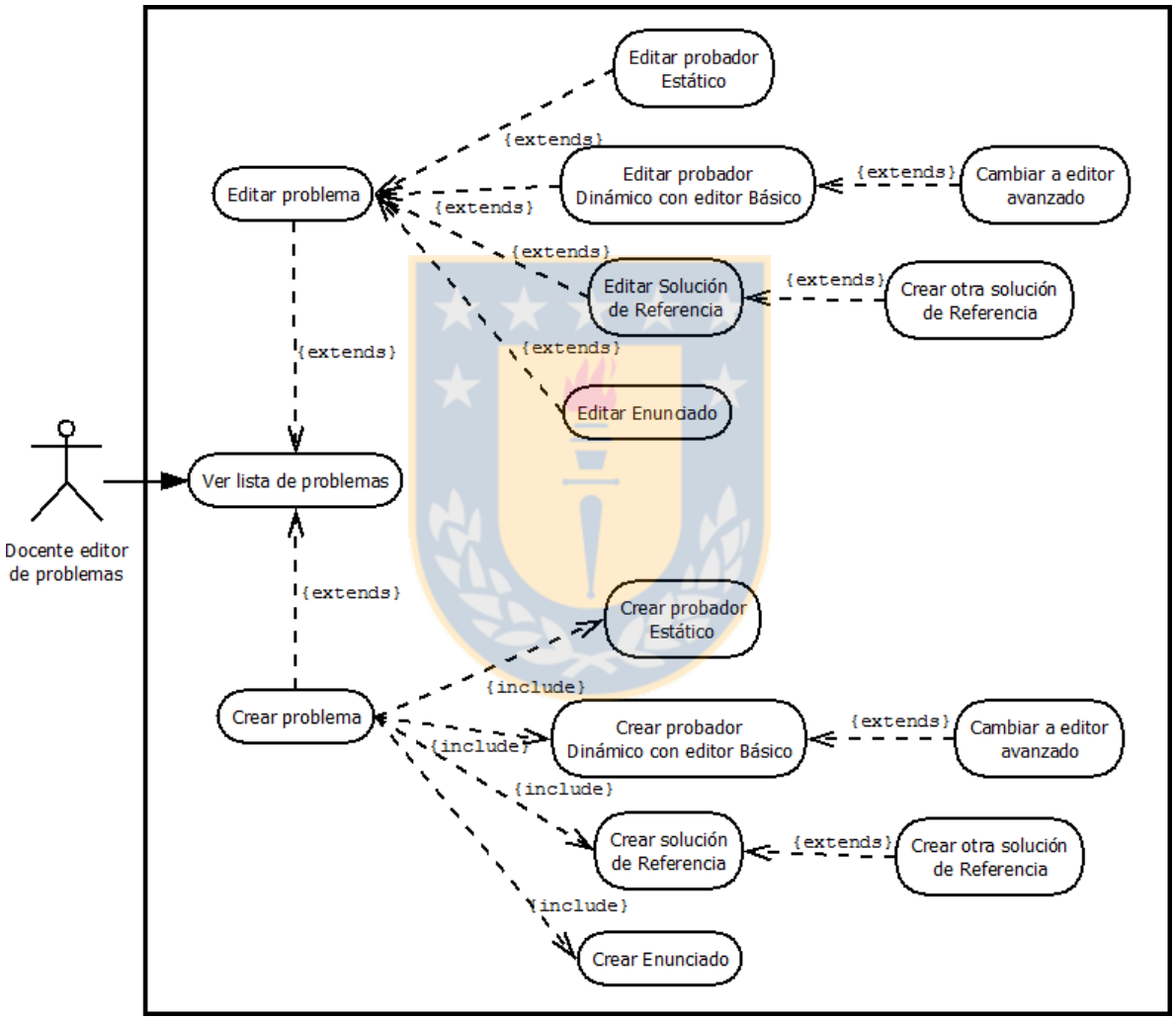

Figura 3.1 Diagrama de casos de uso de la API de administración de problemas

Para editar un problema ya almacenado, el profesor podrá navegar por el listado de problemas y elegir cualquiera de ellos haciendo clic en el nombre correspondiente. Esta acción abrirá los editores de problemas, mostrando en primera instancia el editor de enunciado.

Para crear un problema, el usuario deberá escoger la categoría, luego asignar un nombre y hacer clic en el botón de crear problema. Esto desplegará los editores, mostrando en primera instancia, al igual que en el caso de editar, el editor de enunciado.

3.4 Casos de uso de la API de edición de problemas

A continuación se presentan los principales casos de uso del diagrama anterior. Otros diagramas se encuentran en el anexo 1: casos de uso de la plataforma de administración de problemas.

Caso de uso	Editar un problema.	
ID	1.	
Precondición	El docente está viendo la lista de problemas.	
Resultado	El docente completa la edición de un problema.	
Actores e Intereses	Docente: quiere editar un problema.	
Actores principales	Docente.	
Secuencia normal	**Paso**	**Acción**
	1	El docente hace clic en el nombre del problema que desea editar.
	2	El sistema le muestra el editor al docente.
	3	El docente selecciona el tipo de editor que desea.
	4	El sistema le muestra el editor seleccionado.
	5	El docente hace la edición seleccionada y guarda cambios.
	6	El sistema le indica al docente que se guardaron los cambios.
Excepciones	3	Como el editor muestra por defecto el editor de enunciado, si el docente solo desea modificar el enunciado no necesita escoger el tipo de editor.
	6	Ocurre algún error en el sistema y los cambios no se guardan.
Comentarios		

Caso de uso	Crear un problema.	
ID	2.	
Precondición	El docente está viendo la lista de problemas.	
Resultado	El docente crea el problema.	
Actores e Intereses	Docente: desea crear un nuevo problema.	
Actores principales	Docente.	
Secuencia normal	**Paso**	**Acción**
	1	El docente da un nombre al problema nuevo.
	2	El docente escoge la categoría del problema.
	3	El docente crea el problema.
	4	El sistema le muestra al docente el editor de enunciado.
	5	El docente crea el enunciado, guarda y escoge el editor de probador estático.
	6	El sistema le muestra al docente el editor de probador estático.
	7	El docente crea el probador estático, guarda y escoge el editor de probador dinámico.
	8	El sistema le muestra al docente el editor de probador dinámico
	9	El docente crea el probador dinámico, guarda y escoge el editor de solución
	10	El sistema le muestra al docente el editor de soluciones
	11	El docente crea el editor de soluciones y guarda.
Excepciones	5-11	El docente puede escoger partir por cualquier editor, pues no es necesario que todas las componentes estén hechas para guardarlo, pero sí es necesario que estén todas las componentes hechas para poder usar el problema en una sesión.
Comentarios		

3.5 Base de datos para almacenamiento de problemas

El sistema actual cuenta con una base de datos, que contiene información del aula virtual, esto es, información sobre los cursos, alumnos, respuestas subidas por cada uno, entre otros. Sin embargo, no se almacena en ésta información sobre los problemas como probadores, enunciado ni nombre, ya que como se mencionó en el análisis actual de la plataforma, dicha información se almacena mediante carpetas y archivos.

Este hecho sugiere la posibilidad de crear una nueva base de datos, destinada a guardar toda la información de los editores y los problemas, de esta manera se mantendrán separados el ambiente editor del aula virtual.

Las características que debe tener esta base de datos son las siguientes:

- Capacidad de almacenar todos los problemas, incluyendo enunciados, probadores estáticos, probadores dinámicos y generadores de E/S de prueba

- Capacidad de clasificar los problemas por categoría.
- Capacidad de almacenar plantillas de probadores para reutilizarlos al momento de crear un nuevo problema
- Capacidad de almacenar los filtros que estarán disponibles en los editores de problemas

3.5 Modelo MER de la base de datos de problemas

En base a lo anterior, el modelo MER generado para la base de datos es el de la figura 3.2:

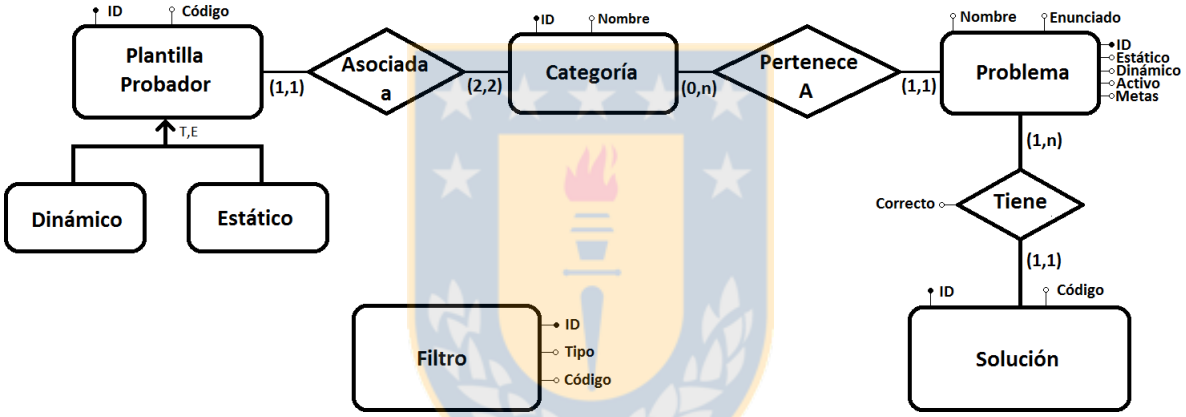

Figura 3.2 Modelo MER de la base de datos para almacenar los problemas

Como es posible apreciar, hay una entidad llamada “Filtro” que no se relaciona con el resto del modelo. Esta tabla está destinada a almacenar una lista de filtros que aparecerán en el editor de probadores estáticos, pero estos no guardan relación con el resto de las tablas, Además se considera innecesario crear una base de datos para almacenar una única tabla.

El atributo "Activo" incluido en la entidad "Problema”, se agrega para indicar si un problema puede o no ser utilizado. Esto con el fin de prevenir que problemas no terminados estén disponibles al momento de crear una nueva sesión de evaluación.

El modelo cuenta con las siguientes restricciones no modeladas:

- De las dos plantillas asociadas a cada categoría, una debe ser de probador estático y otra de probador dinámico.
- Cada problema debe tener exactamente una solución correcta.
- Cada problema puede tener ninguna o varias soluciones incorrectas

- Los atributos “Estático” y “Dinámico” pueden estar vacíos, sin embargo para poder utilizar el problema estos campos deben estar completos.

3.6 Mockups de la API de edición de problemas

Para el sistema de edición de problemas, se generó una serie de mockups: uno para el navegador de problemas, uno para el editor de probadores estáticos, uno para el editor de probadores dinámicos, uno para el editor del enunciado del problema, y uno para el editor del programa solución de referencia.

El primero de ellos corresponde a la figura 3.3. En él se muestra cómo verá el usuario la lista de problemas, donde verá los ID, los nombres de cada problema, la categoría a la cual pertenece cada uno, y si está activo o no. Esta lista será ordenable por cualquiera de las cuatro columnas. En la parte superior se incluyen las opciones para crear un nuevo problema.

Figura 3.3 Mockup de la vista del navegador de problemas

Cada editor contiene una barra de enlaces en la parte superior que permite cambiar entre cada uno de ellos, además de otras opciones como volver al listado de problemas. Ésta se muestra en la figura 3.4.

Figura 3.4 Mockup de la vista de la barra superior presente en los editores

El editor de probador dinámico tiene dos presentaciones distintas: una simplificada y otra completa. La presentación simplificada muestra dos bloques: uno para agregar variables y funciones externas y otro para crear un generador de entradas, quedando el resto del código oculto al usuario; además muestra una descripción de lo que debería contener cada bloque a modo de ayuda al usuario. La presentación completa muestra el código del

probador en su totalidad, permitiendo al usuario hacer cambios más complejos que el editor simplificado no permite. Ambas vistas corresponden a las figuras 3.5 y 3.6.

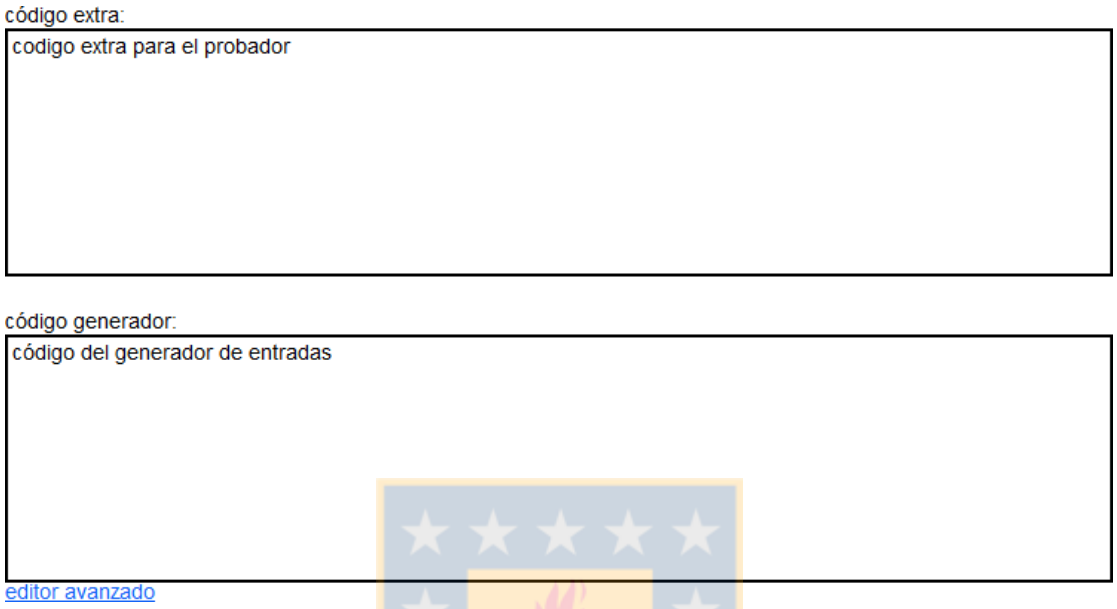

Figura 3.5 Mockup de la vista del editor dinámico simplificado

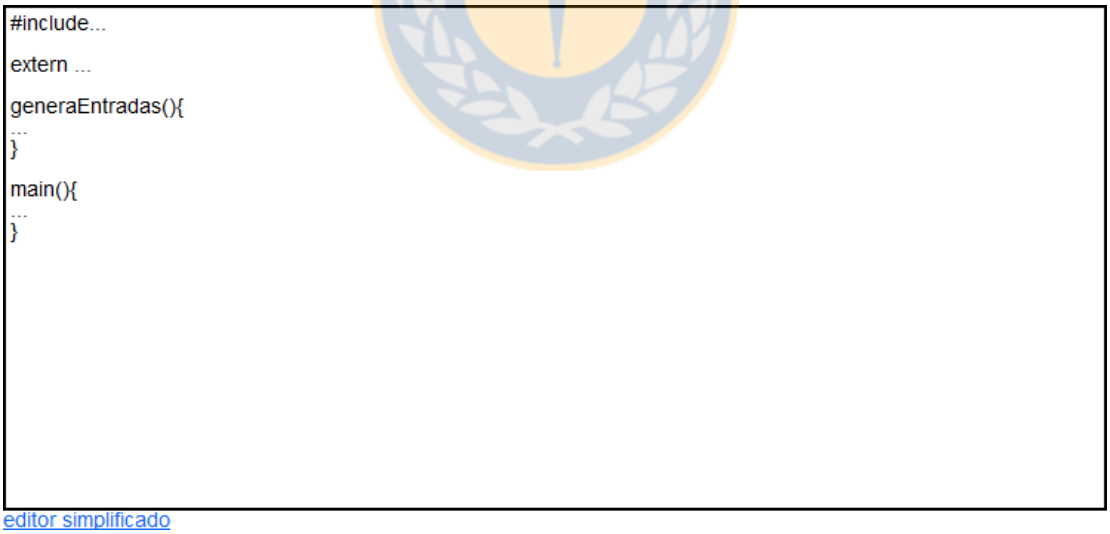

Figura 3.6 Mockup de la vista del editor dinámico avanzado

En el editor de probadores estático, como se puede apreciar en la figura 3.7, se muestra a la derecha un panel de texto que contendrá el código fuente del probador, y a la izquierda se presenta una lista con plantillas de filtros para agregar al código.

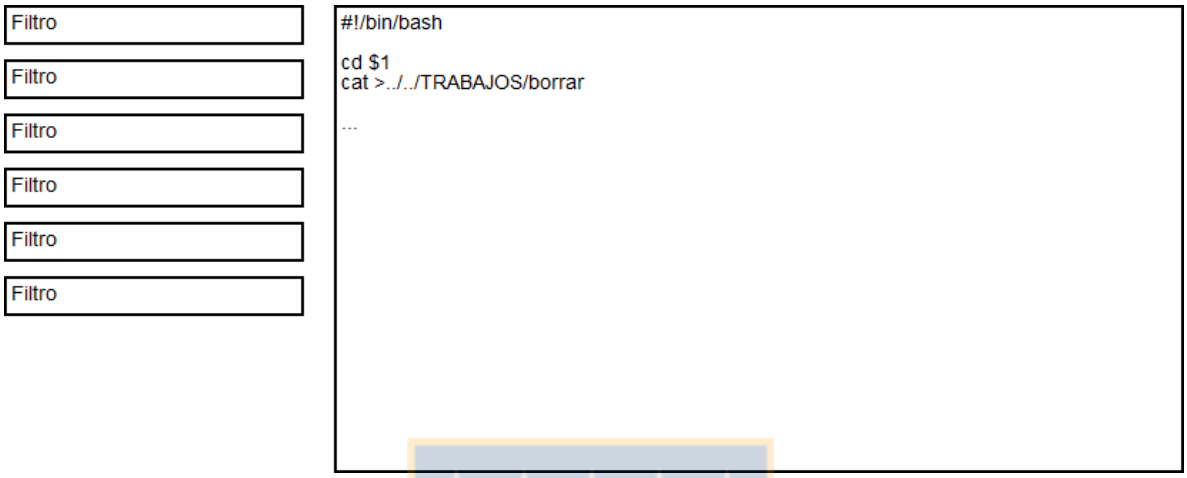

Figura 3.7 Mockup de la vista de un editor de probador estático

El editor de programas soluciones cuenta con dos bloques: uno a la izquierda, que contiene opciones para elegir la solución a editar, crear nuevas o eliminar alguna, y uno a la derecha que muestra el código del programa y permite editarlo. La figura 3.8 muestra esta división.

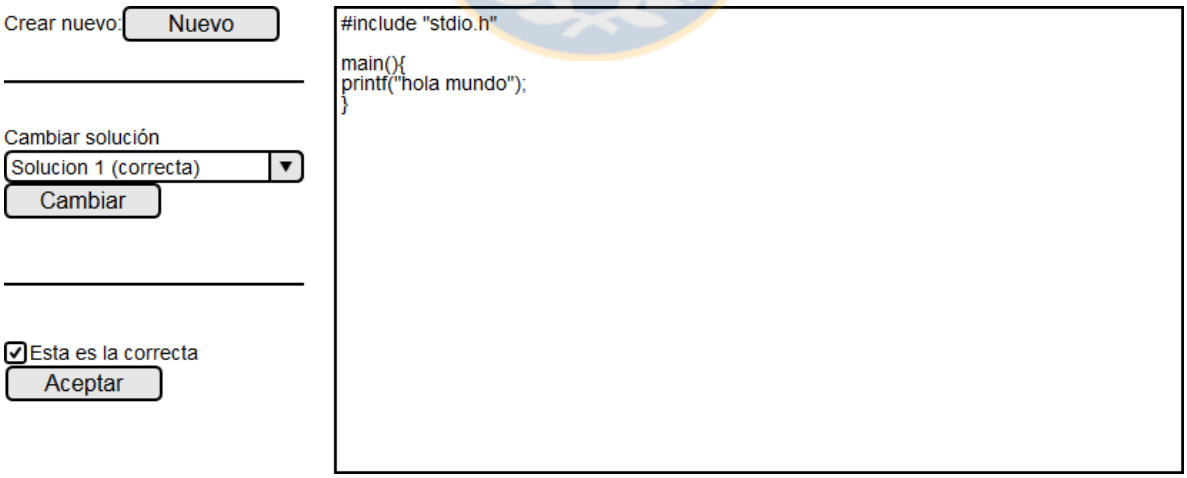

Figura 3.8 Mockup de la vista del editor de programa solución de referencia

El editor del enunciado es más simple, dado que solo contiene la caja de texto para escritura. Se deja de esta manera porque el texto del enunciado es simplemente texto escrito en español. Esta simpleza se aprecia en la figura 3.9.

Haga un programa que escriba en su salida estandar:

Hola mundo

Figura 3.9 Mockup de la vista del editor de enunciado

Capítulo 4: Desarrollo API para Edición de problemas

4.1 Introducción

La API de edición fue construida en los lenguajes HTML y PHP. Además se utilizó JavaScript y jQuery para generar la tabla que organiza los problemas. Las vistas reales de esta API se encuentran en el anexo 2 al final de este informe.

En cuanto a los archivos, se utilizaron varios programas PHP. Cada editor cuenta con su propio archivo, así como también la tabla y el programa de guardado. Además se deja en un archivo aparte la barra superior con opciones presente en los editores, esto porque para todos es la misma y así si se requiere modificar se modifica una sola vez. El editor de soluciones cuenta con varios formularios, por lo que cada uno de ellos tiene su propio programa PHP. También se creó un archivo CSS que cubre toda la API.

Nombre de archivo	Descripción
tabla.php	Despliega la lista de problemas. Es el punto de partida de la API de edición de problema.
datos_de_la_conexion.php	Archivo que contiene el usuario y la contraseña de la base de datos, se incluye cuando se requiere una conexión.
cabecera_editores.php	Contiene la barra de selección de los editores.
editor_dinamico.php	Código del editor de probadores dinámicos de código reducido.
editor_dinamico_full.php	Código del editor de probadores dinámicos de código completo.
editor_estático.php	Código del editor de probadores estáticos.
editor_enunciado.php	Código del editor de enunciados.
editor_solucion.php	Código del editor de soluciones.
guardar.php	Programa que controla el guardado de cada editor.
elegir_solucion.php	Programa que controla el cambio entre las distintas soluciones de un problema.
eliminar_solucion.php	Programa que elimina una solución de un problema.
marcar_correcto.php	Programa que marca una solución como correcta.
nueva_solucion.php	Programa que crea una nueva solución para un problema
nuevo_problema.php	Programa que crea un nuevo problema.
estilo.css	Contiene todas las opciones de estilo de la API.

Tabla 4.1 resumen de los archivos que componen la API de gestión de problemas

4.2 Lista de problemas

La lista de problemas es generada mediante un jQuery llamado Tablesorter. Es un script que se usa en conjunto con una tabla en HTML y que permite ordenar las filas de la tabla por alguna de las columnas haciendo clic en cualquiera de los títulos de columna.

Para generar la tabla, se escriben directamente los títulos de columna, luego se ejecuta una consulta a la base de datos que obtiene todos los problemas, y finalmente genera las filas usando cada uno de los resultados de la consulta.

La barra de la parte superior, que muestra opciones para crear un nuevo problema, es un formulario en HTML que captura el nombre y la categoría. Éste ejecuta un programa en PHP que es el que crea el problema en la base de datos.

4.3 Barra de selección de los editores

La barra superior que se muestra en los editores es un archivo PHP que es incluido dentro del código de cada uno. Éste es parte de un formulario HTML que incluye enlaces a los otros editores, un enlace a la lista de problemas, el botón de guardado y una casilla de verificación que se utiliza para marcar el problema como utilizable.

Este archivo mezcla reiteradamente código HTML y PHP, por lo que para hacerlo más legible se optó por desplegar todo el código HTML mediante PHP usando instrucciones "`echo`".

4.4 Editor de probadores dinámicos

El editor de probadores dinámico está dividido en dos archivos distintos. Uno para el básico y otro para el completo. Ambos archivos cargan el probador desde la base de datos mediante PHP, sin embargo la forma de mostrarlo es distinta.

Para el probador dinámico se estableció una plantilla general. Ésta se utiliza para un gran número de problemas, ya que basta con agregarle funciones o variables y establecer un generador de entradas de prueba. A esta plantilla se le incorporó a marca "`/*S#%*/`", que es utilizada por el editor básico para dividir el código del probador y poder mostrar el fragmento que corresponde en cada caja de texto. Cada probador debe contener cuatro de estas marcas. Las primeras dos indican el comienzo y el fin del espacio para agregar variables y funciones, mientras que las otras dos indican el comienzo y el fin del espacio para crear el generador de referencias. El código que esta fuera de las marcas es código que en general no se modifica. En el editor avanzado, que muestra el código completo, esta marca no presta ninguna utilidad, por lo que al estar incluida en el probador es mostrada al usuario.

La plantilla mencionada no es única. Existen problemas que trabajan con más de una meta, y éstas últimas pueden ser de distinto tipo. Se estima, en base a los problemas

observados, que es posible generar alrededor de 4 plantillas distintas para distintas categorías.

El editor básico implementa un formulario con dos cajas de texto, una en la parte superior y otra en la parte inferior. La superior es para agregar funciones y variables, mientras que la inferior es para crear un generador de entradas de prueba. El editor obtiene el probador desde la base de datos y lo divide en fragmentos utilizando la marca mencionada. Se deberían generar cinco fragmentos, de los cuales el editor muestra solo dos, ocultando el resto. Si no encuentra todas las marcas, obliga al usuario a utilizar el editor avanzado ya que se hace imposible determinar dónde dividir el código. Al guardar los cambios, el editor le entrega al programa PHP de guardado solo los fragmentos mostrados al usuario, donde son posteriormente concatenados con el resto del probador.

A diferencia del editor dinámico básico, el completo implementa solo una caja de texto que contiene el código completo del probador, incluidas las marcas de división. Estas eventualmente pueden ser borradas por el usuario, ya sea por descuido o deliberadamente. Para evitar esto, se establece una advertencia para indicarle al usuario que no debe hacerlo. En el caso de que lo haga, puede en el futuro agregarlas manualmente, pero corriendo el riesgo de insertarlas donde no corresponde. Este editor le entrega el código completo al programa de guardado, junto con el ID del problema.

4.5 Editor de probadores estáticos

El editor de probadores estáticos es un archivo PHP que implementa una formulario HTML con una caja de texto que muestra el código completo del probador. Carga desde la base de datos el probador y lo muestra al usuario para que lo modifique. Al guardar, le entrega el código completo al programa de guardado, junto con el ID del problema. Este editor está pensado para que en un futuro permita agregar filtros de manera fácil al código, o bien explorar un listado de filtros, por lo que incluye un panel a la izquierda conteniendo un acordeón.

4.6 Editor de enunciados

El editor de enunciados es un formulario HTML que contiene una caja de texto. Su funcionamiento es similar al de probadores estáticos. No tiene ninguna función especial debido a que recibe simplemente texto en español.

4.7 Editor de soluciones de referencia

El editor de soluciones de referencia está compuesto por varios formularios. El primero de ellos es el formulario del editor de la solución actual, y los otros son los que manejan las opciones de todas las soluciones del problema en el cual se está trabajando.

El panel de edición de la solución no cuenta con ninguna opción, ya que el botón de guardado está situado en la barra de selección de los editores. El panel de manejo de las soluciones del problema cuenta con varios botones de tipo “submit” para las distintas opciones. Cada formulario cuenta con su propio PHP de control.

En primer lugar se obtienen los datos del problema con el que se está trabajando, luego comprueba si la variable “$id_sol” está o no definida. Después carga la solución y crea el panel de opciones de las soluciones del problema.

Que la variable “$id_sol” esté definida significa que al editor de soluciones se entró desde sí mismo, esto es, desde el panel de opciones de la izquierda; en este caso se carga la solución requerida. Que no esté definida significa que al editor se entró desde la barra de selección de editor; en este caso se carga siempre la solución correcta.

El panel de opciones cuenta con cuatro partes: la primera es un botón que crea una nueva solución vacía, la segunda es un menú que permite elegir cuál de todas las soluciones del problema se desea trabajar, la tercera es una caja de verificación, que al activarla permite establecer la solución actual como correcta, y la cuarta permite eliminar la solución actual.

4.8 Programa PHP de guardado

El programa de guardado, construido en PHP, controla el funcionamiento de todos los formularios de edición del sistema. Primero éste determina desde qué tipo de editor se está invocando. Esto lo hace mediante la variable “$tipo”, que llega por método POST desde los editores. También al inicio recibe el ID del problema para saber a cuál de todos se debe aplicar la modificación. De acuerdo a eso, el funcionamiento es el siguiente:

- Si el editor es el estático, obtiene el contenido de la caja de texto por POST y actualiza en la base de datos el probador estático del problema solicitado.
- Si el editor es el dinámico básico, obtiene primero el probador dinámico completo de la base de datos, luego lo divide de acuerdo a las marcas para obtener el código no mostrado y descartando los fragmentos mostrados, obtiene mediante POST los códigos de las cajas de texto, realiza la concatenación de los códigos no

mostrados más los editados en el orden correcto, y actualiza la base de datos con el código concatenado.

- Si el editor es el dinámico avanzado, trabaja similar al caso del editor estático, pero actualizando el campo dinámico del problema solicitado.
- Si el editor es el de enunciado, trabaja similar al caso del editor estático, pero en este caso actualiza el campo enunciado del problema solicitado.
- Si el editor es el de soluciones, trabaja similar al caso del editor estático, pero actualizando la solución en la tabla de soluciones.

Una vez que los cambios han sido guardados, se le informa al usuario que la operación fue exitosa, o bien que no se ha guardado nada porque no hay cambios. Luego se redirige al editor en el cual se estaba antes de guardar. En el caso del editor de soluciones, carga la solución que se estaba editando antes de guardar.

4.9 Programas PHP de opciones para las soluciones

Como se mencionó anteriormente, el editor de soluciones cuenta con cuatro formularios extra para las opciones de las soluciones de un problema, los cuales se describen a continuación:

Para crear una nueva, existe el programa “nueva_solucion.php” que inserta una nueva fila en la tabla de soluciones, obtiene el nuevo ID, crea la relación entre la solución y el problema asociado y redirige de vuelta al editor con la nueva solución.

Para cambiar a otra solución, existe el programa “elegir_solucion.php” que simplemente redirige al editor de soluciones con el ID solicitado.

Para establecer una solución como correcta, existe el programa “marcar_correcto.php” que primero determina si se activó la casilla de verificación. En caso afirmativo marca como no correcta todas las soluciones del problema, para luego marcar como correcta la solicitada. En ambos casos, afirmativo y negativo, redirige de vuelta al editor con la solución actual.

Para eliminar una solución, existe el programa “elimina_solucion.php”. Lo primero que hace es determinar si la que se desea eliminar tiene la marca de correcta. Esto porque no se puede eliminar una solución si es la correcta. En caso afirmativo le indica al usuario que la operación no está permitida y redirige a la solución actual. En caso negativo, primero elimina la relación entre el problema y la solución, y luego la elimina. Para volver al editor, busca aquella con la marca de correcta y redirige con ella.

Capítulo 5: Diseño de Ambiente de Administración del Aula Virtual

5.1 Introducción

El segundo objetivo del presente trabajo consiste en diseñar una interfaz para administrar el aula virtual. Esta deberá permitir la gestión de la plataforma de manera amigable, y así eliminar la necesidad de conocer cómo funciona la base de datos que almacena a alumnos, cursos y actividades.

5.2 Requisitos funcionales del ambiente de administración

La interfaz de administración del aula virtual debe permitir las siguientes acciones:

- Agregar nuevos alumnos al sistema, ya sea de manera manual uno a uno o bien en forma automatizada mediante un archivo de lista
- Crear una nueva actividad y asignarle problemas
- Iniciar y detener una actividad en particular
- Crear y ver la lista de cursos y los alumnos de cada uno
- Ver las calificaciones de cada alumno en cada evaluación
- Integración con la API de edición de problemas, y permitir la modificación de alguno sin interrumpir el trabajo

5.3 Diagrama de casos de uso del ambiente de administración

En base a la descripción anterior, se generó el diagrama de casos de uso de la figura 5.1. En él se puede apreciar que el docente tendra cinco opciones desde la vista principal. El resto de acciones estarán dentro de la consulta por la lista de cursos del sistema.

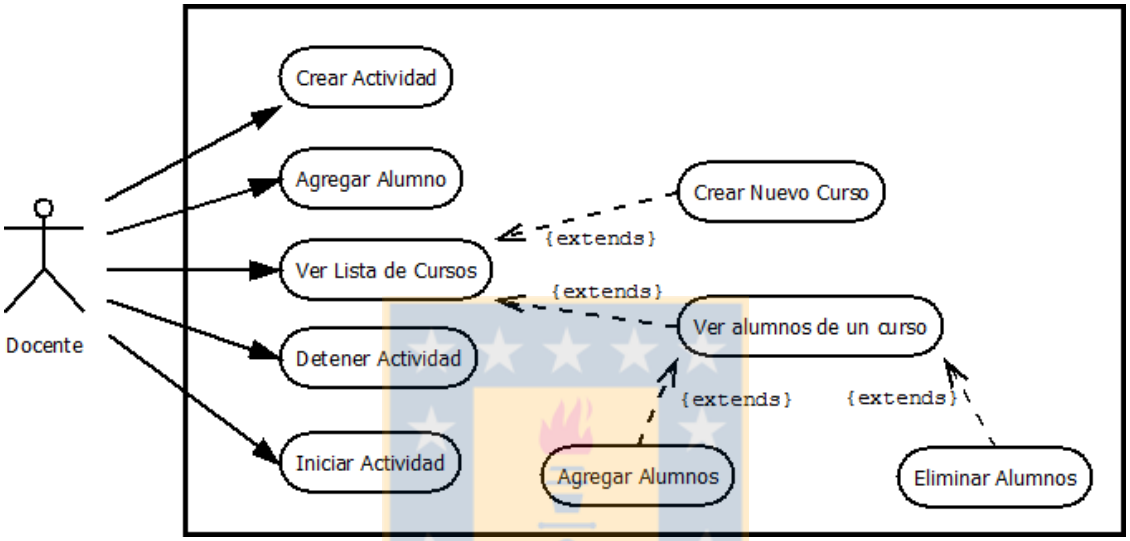

Figura 5.1 Diagrama de casos de uso de la plataforma de administración del aula virtual

Al consultar la lista de cursos, el docente tendrá acceso a crear un nuevo curso y a consultar la lista de alumnos de un curso en particular. Al hacer esto último, también se mostrarán las notas obtenidas en cada evaluacion por cada estudiante. Si desea agregar o eliminar alumnos de un curso, el usuario puede indicárselo al sistema desde la lista del curso.

Dentro de todas las opciones, también se encuentra el acceso a la API de edición de problemas. Estos accesos estarán dentro de la lista de acciones de la vista principal, desde la lista de problemas en crear actividad, y en la vista del detalle de una actividad. De ellos, el primero llevara a la lista de problemas almacenados en el sistema, mientras que el resto serán enlaces directos presentes en el nombre del problema y que llevarán al usuario al editor de enunciado.

5.4 Casos de uso del ambiente de administración

Caso de uso	Crear una actividad.	
ID	1.	
Precondición	El docente está en algún lugar de la plataforma de administración del aula virtual.	
Resultado	El docente crea una nueva actividad.	
Actores e Intereses	Docente: quiere crear una actividad.	
Actores principales	Docente.	
Secuencia normal	**Paso**	**Acción**
	1	El docente hace clic en la opción “Crear nueva actividad”.
	2	El sistema le pregunta al docente para qué curso desea crear una nueva actividad.
	3	El docente le indica al sistema el curso deseado.
	2	El sistema crea una nueva actividad y le muestra al docente el listado de problemas para que los agregue.
	3	El docente elige el curso para el que estará disponible la actividad, selecciona un tema, y escoge problemas, marcando las casillas verificadoras correspondientes; luego acepta.
	4	El sistema crea asociaciones entre cada problema seleccionado y la nueva actividad, y le informa al docente que se ha creado correctamente
Excepciones		Ninguna
Comentarios		

Caso de uso	Agregar alumno	
ID	2.	
Precondición	El docente está en algún lugar de la plataforma de administración del aula virtual.	
Resultado	El docente agrega uno o más alumnos.	
Actores e Intereses	Docente: quiere agregar uno o varios alumnos.	
Actores principales	Docente.	
Secuencia normal	**Paso**	**Acción**
	1	El docente hace clic en la opción “Agregar alumnos”
	2	El sistema le muestra al docente la vista para agregar alumnos.
	3	El docente ingresa el nombre y el número de matrícula del alumno nuevo y confirma.
	4	El sistema agrega al alumno y le indica que la operación concluyó.
Excepciones	3	El docente decide agregar alumnos mediante una lista que puede ser pegada como texto o bien subida como un archivo de texto.
Comentarios		

Caso de uso	Iniciar/detener actividad.	
ID	3.	
Precondición	El docente está viendo la lista de actividades.	
Resultado	El docente Inicia o detiene una actividad.	
Actores e Intereses	Docente: quiere iniciar o detener una actividad.	
Actores principales	Docente.	
Secuencia normal	**Paso**	**Acción**
	1	El docente escoge una actividad y oprime el botón de iniciar o detener actividad.
	2	El sistema le indica al docente una advertencia de confirmación.
	3	El docente confirma que desea iniciar o detener esa actividad.
	4	El sistema realiza los cambios y le indica al sistema que la operación tuvo éxito.
Excepciones		
Comentarios	El iniciar una actividad provocará que cualquier otra actividad que se encuentre iniciada sea detenida. Esto se le informa al docente en el aviso de confirmación.	

5.5 Mockups del ambiente de administración

En base a la descripción de lo requerido hecha anteriormente, se plantean una serie de mockups que a continuación se presentan.

En la figura 5.2 se observa la presentación general de la interfaz, donde es posible apreciar que contará con un menú lateral con todas las opciones que estarán disponibles, además de una zona de trabajo que variará con cada función escogida.

Figura 5.2 Mockup de la vista general de la interfaz de administración del aula virtual

La figura 5.3 corresponde a la primera opción, “Crear actividad”. En ella el sistema mostrará al usuario un menú desplegable que permite elegir el tema de la actividad, y además muestra la lista de problemas disponibles. El objetivo de la primera columna es marcar los problemas que se incluirán en la actividad, mientras que el de la última es indicarle al sistema cuál será el puntaje que obtendrá el alumno al resolver correctamente el ejercicio.

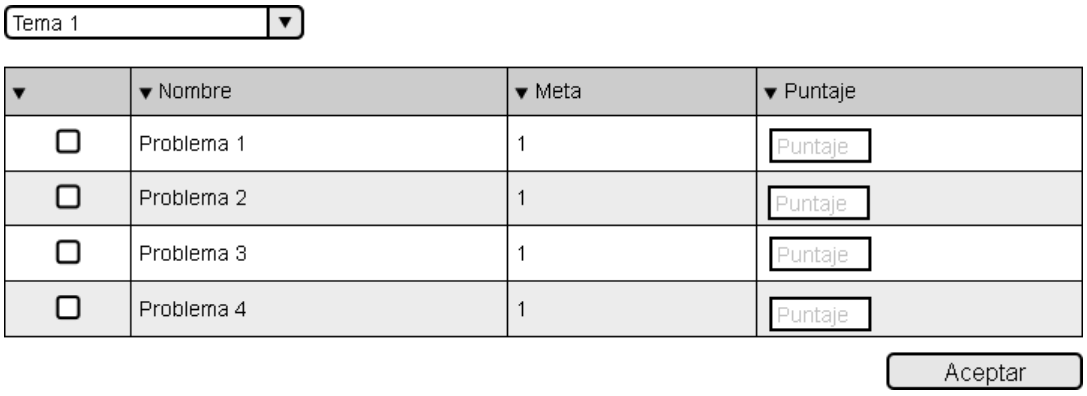

Figura 5.3 Mockup de la vista de la opción Crear Actividad

Cuando el usuario desee revisar las actividades creadas, el sistema le mostrará la lista de la figura 5.4. En ella podrá ver el estado de las actividades en la tercera columna, a la vez que podrá cambiarlo si lo desea con los botones de la cuarta columna. También podrá ver qué problemas tiene asociado cada una haciendo clic en el nombre del tema.

Figura 5.4 Mockup de la vista de la opción Ver Actividades

En la figura 5.5 se presenta la vista del detalle de cada actividad. Ésta permitirá al usuario editar un problema haciendo clic en el nombre correspondiente lo que abrirá una ventana nueva con los editores creados en los capítulos anteriores de este proyecto. Se abrirá una ventana nueva para así mantener el trabajo actual del aula virtual y evitar que el usuario tenga que oprimir el botón atrás en el navegador para volver.

▼ ID	▼ Problema	▼ Meta	▼ Puntaje
1	Problema 1	1	1
2	Problema 2	1	1
3	Problema 3	1	1
4	Problema 4	1	1

Figura 5.5 Mockup de la vista del detalle de una actividad

Si el usuario desea agregar alumnos al sistema, tendrá tres opciones para hacerlo. Estas se muestran en la figura 5.6. La primera de ellas es ingresar un solo alumno completando los campos de matrícula y nombre. La segunda opción es cargar un archivo de texto con algún formato específico que el sistema pueda leer, y agregar de esta manera un lote de estudiantes de manera sencilla. La última alternativa consiste en una caja de texto en la cual el usuario puede ingresar uno o varios alumnos en algún formato específico.

El formato planificado para las opciones dos y tres se establece como un alumno por línea, colocando en primer lugar el número de matrícula, luego una coma, y luego el nombre completo del alumno. Adicionalmente, para la segunda alternativa, se establece que el archivo debe ser de texto simple, es decir, con extensión "`.txt`".

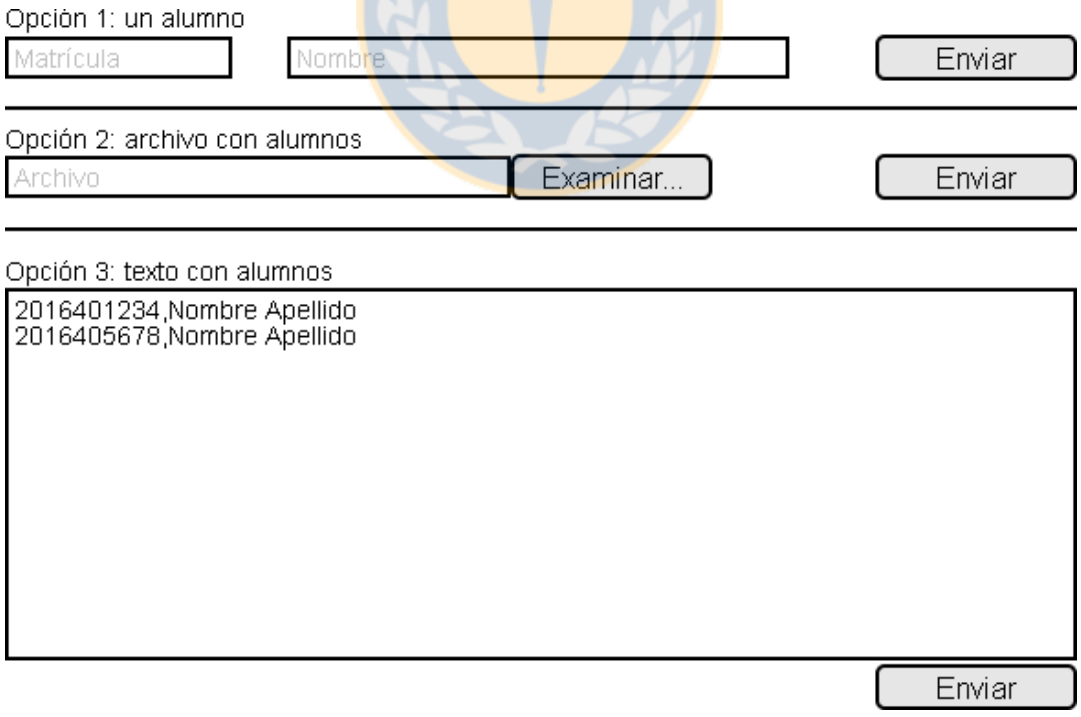

Figura 5.6 Mockup de la vista de la opción Agregar Alumnos

Cuando el usuario desee ver la lista de cursos, el sistema le mostrará la vista de la figura 5.7, que contiene en su parte superior cuatro cuadros de texto y un botón de confirmación que cumplen la función de agregar un nuevo curso al sistema. Además le mostrará la lista de cursos, y le permitirá acceder al detalle de alguno pulsando en su nombre.

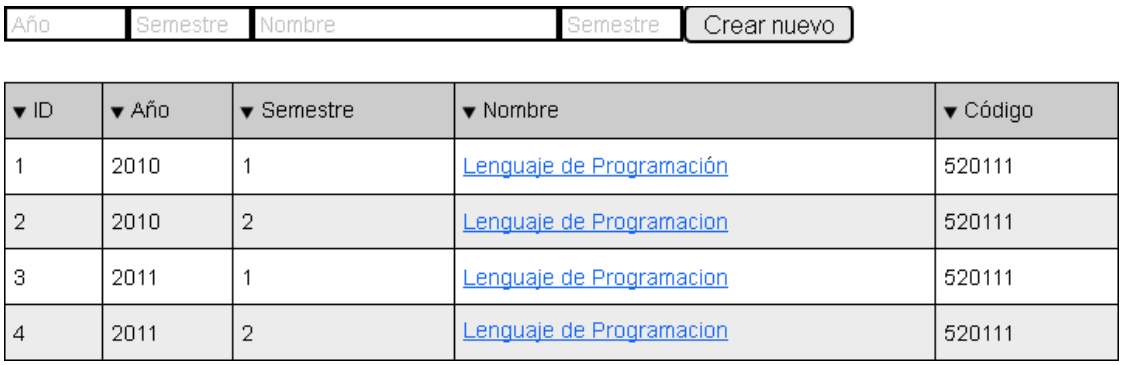

Figura 5.7 Mockup de la vista de la lista de cursos

Al acceder al detalle de algún curso, el usuario podrá observar los alumnos que están en él, las notas de cada uno, y además podrá eliminar o agregar estudiantes. Para removerlos se deberán marcar las casillas verificadoras correspondientes y pulsar el botón de eliminación de la parte inferior. Junto a este botón, existe otro que lleva a la vista de selección de alumnos para agregar nuevos. La figura 5.8 muestra el detalle de un curso.

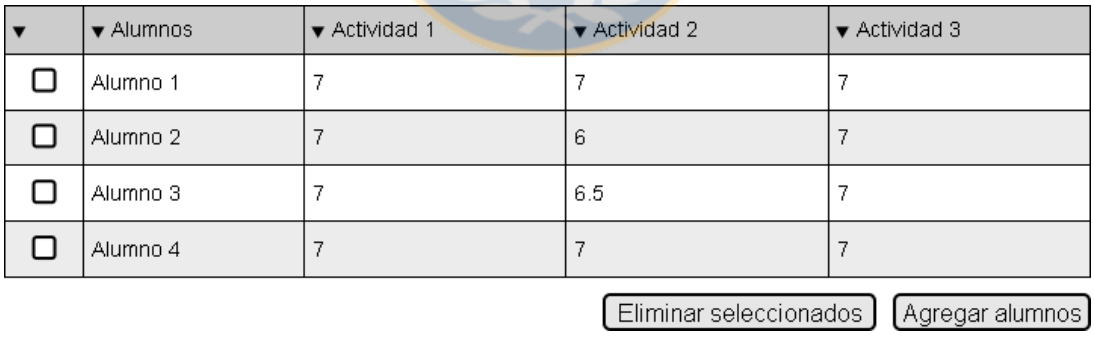

Figura 5.8 Mockup de la vista de detalle de un curso

La vista de la figura 5.9 cumple la función de agregar alumnos a un curso. Permitirá cumplir este objetivo para ambos casos de agregación, es decir, cuando el usuario cree un curso nuevo y cuando desee incluir a alumnos en un curso ya existente. La lista mostrará todos los alumnos del sistema, y en el caso de modificar un curso no mostrará a aquellos que ya estén en él. Para seleccionarlos, se deberán marcar las casillas de verificación correspondientes y luego el botón de aceptar de la parte inferior.

▼	▼ Nombre	▼ Matrícula
☐	Alumno 1	2016401234
☐	Alumno 2	2016405678
☐	Alumno 3	2016409012
☐	Alumno 4	2016403456

Aceptar

Figura 5.9 Mockup de la vista para agregar alumnos a un curso

Capítulo 6: Desarrollo de Ambiente de Administración de Aula Virtual

6.1 Introducción

Al igual que la API de gestión de problemas, el ambiente de administración de aula virtual se construyó como una página web utilizando HTML y PHP, además de JavaScript y jQuery para las tablas, en específico el plugin Tablesorter. Las vistas reales del ambiente se pueden ver en el anexo 4.

En cuanto a los archivos, todo está hecho en formato PHP, excepto el estilo que es un CSS compartido con la API de gestión de problemas. Cada vista tiene su propio fichero PHP que genera las vistas y realiza las consultas a las bases de datos. Además, existen otros archivos PHP que son los encargados de hacer las modificaciones a las bases de datos cuando corresponde. También existe un archivo, compartido por todas las vistas, que genera el menú de la izquierda.

Nombre de archivo	Descripción
inicio.php	Página de inicio del ambiente de administración del aula virtual.
ver_actividades.php	Muestra las actividades creadas en el sistema.
detalle_actividad.php	Muestra los problemas que están asociados a una actividad.
crear_actividad.php	Inicia la creación de una actividad, le pide al usuario el curso donde se creará una.
crear_actividad2.php	Segunda parte de la creación de una actividad, solicita los problemas que incluirá la actividad
ver_alumnos.php	Muestra la lista de alumnos que tiene el sistema.
detalle_alumno.php	Muestra las calificaciones que ha obtenido un determinado alumno
agregar_alumnos.php	Muestra las opciones disponibles para agregar alumnos al sistema.
cursos.php	Muestra la lista de cursos del sistema y permite crear nuevos cursos.
detalle_curso.php	Muestra los alumnos presentes en un curso y las calificaciones obtenidas por cada uno en cada actividad.
agregar_a_curso.php	Permite agregar alumnos a algún curso.
nueva_actividad.php	Controla la creación de una actividad
inicia_detiene.php	Controla el inicio y la detención de una actividad
nuevo_alumno_uno.php	Controla la creación de un alumno
nuevo_alumno_archivo.php	Controla la creación de varios alumnos mediante un archivo de texto
nuevo_alumno_varios.php	Controla la creación de varios alumnos mediante texto ingresado a mano
nuevo_curso.php	Controla la creación de un curso
nuevo_alumno_curso.php	Controla la vinculación de un alumno a un curso
elimina_alumno_curso.php	Controla la remoción de un alumno de un curso
cambia_detalle_curso.php	Recarga la página de detalle del curso con otra actividad

Tabla 6.1 resumen con los archivos que componen el ambiente de administración de aula virtual

6.2 Vista inicial del ambiente de administración

La vista inicial solo sirve como punto de partida para el ambiente de administración de aula virtual. Está compuesta de dos archivos PHP. El primero es la vista general, mientras que el segundo es el menú de opciones de la izquierda.

El menú de selección es utilizado por todas las vistas de la plataforma. Debido a esto se dejó en un archivo aparte, ya que así se facilita su modificación, en el caso de que en algún momento se decida agregar nuevas opciones. Mezcla reiteradamente código PHP y HTML, por lo que se optó por desplegar todo mediante instrucciones "`echo`" para hacerlo más legible.

6.3 Creación y manejo de actividades

La vista de actividades es un archivo PHP que genera una tabla con todas las actividades registradas en el sistema. Cada entrada de la tabla es un formulario que implementa solo un botón de iniciar o detener, de acuerdo a si está detenida o iniciada, respectivamente. Todos funcionan con el mismo programa PHP llamado "inicia_detiene.php" el cual, al pulsar el botón de aceptar, recibe el ID de la actividad y la detiene o inicia según sea el caso. Además, el nombre de la actividad es un enlace al detalle de la misma.

El detalle de una actividad utiliza el id de la misma para obtener los problemas asociados a ella, y los muestra mediante una tabla que usa el plugin Tablesorter para ordenar las filas. El nombre de cada problema es un enlace que carga el editor de enunciado de dicho problema, y utiliza su ID para identificación.

La creación de actividades está compuesta por dos pantallas. La primera es un formulario en HTML que permite seleccionar el curso mediante una lista de opciones desplegable. Al aceptar, se abre la segunda vista que también es un formulario en HTML, que está compuesto por dos selecciones: una lista desplegable para el tema de la actividad y una tabla con la lista de problemas para seleccionar. Cada entrada de la tabla contiene una casilla de verificación para seleccionar cada problema y un espacio de texto para establecer el puntaje de cada uno. Ambas selecciones tienen asociado el ID del problema, para así poder detectar cuáles han sido seleccionados y a qué problema corresponde cada puntaje indicado. La tabla hace uso del plugin Tablesorter para facilitar el ordenamiento de las filas.

6.4 Gestión de alumnos

La vista de alumnos es una tabla en HTML que carga toda la lista de alumnos mediante una consulta a la base de datos y haciendo uso del plugin TableSorter para el ordenamiento de las filas. Cada nombre de alumno es un enlace al detalle del mismo, identificado mediante su número de matrícula.

Para generar el detalle de un alumno, primero se utiliza su número de matrícula para mostrar su nombre. Luego se identifica si éste tiene alguna calificación. En caso negativo se muestra solo un mensaje indicando esta situación. En caso afirmativo, se despliega una tabla con Tablesorter donde se muestra el nombre de la actividad y su calificación.

La vista para agregar alumnos está compuesta por tres formularios HTML, uno para cada una de las tres opciones de agregación. El primero recibe los datos de un alumno con dos cajas de texto para entregárselas al programa “nuevo_alumno_uno.php” que hace la inserción en la base de datos. El segundo recibe un archivo de texto con los alumnos, y al ser subido es procesado por el programa “nuevo_alumno_archivo.php”. El tercero es una caja de texto grande que contendrá varios alumnos, uno por fila, para luego ser insertados por el programa “nuevo_alumno_varios.php”.

El programa “nuevo_alumno_uno.php”, simplemente recibe el nombre y la matrícula del alumno para luego hacer una inserción en la base de datos. Los programas “nuevo_alumno_archivo.php” y “nuevo alumno_varios.php” son muy similares, dado que ambos reciben el texto con la lista de alumnos, lo dividen por línea, luego cada línea es dividida para obtener matricula y nombre, y luego se inserta en la base de datos. La diferencia es la forma de recibir el texto, ya que “nuevo_alumno_archivo.php” recibe la información como un archivo de texto mientras que “nuevo_alumno_varios.php” recibe el contenido de una caja de texto del formulario.

6.5 Gestión de cursos

La vista de cursos está compuesta por dos partes. La primera es un formulario que permite la creación de un nuevo curso, este recibe los datos para luego, al hacer clic en aceptar, entregárselos al programa “nuevo_curso.php” el cual construye la instrucción SQL para luego realizar la inserción en la tabla “curso” de la base de datos. La segunda es una tabla con Tablesorter que muestra la lista de cursos que actualmente hay en el sistema. El nombre de cada curso es un enlace a la vista del detalle de ese curso.

El detalle de cada curso está formado por tres formularios HTML. El primero de ellos se utiliza para escoger la actividad de la cual se mostrarán las notas. Se genera utilizando un JOIN que permite obtener todas las actividades de un curso junto con las calificaciones, y

el cambio es realizado por el programa “cambiar_detalle_curso.php” que simplemente contiene una redirección html. El segundo es el listado de alumnos, mostrados en una tabla Tablesorter, que incluye la matrícula, el nombre, y la calificación de la actividad seleccionada, además de una casilla que permite marcar al alumno para eliminarlo del curso. El tercer formulario está al final de la vista, y corresponde a un botón que permite agregar alumnos al curso, este captura el ID del curso para usarlo en la siguiente vista.

La eliminación de estudiantes de un curso está controlada por el programa “elimina_alumno_curso.php”, el cual obtiene desde la base de datos la lista completa de alumnos pertenecientes al curso, y luego realiza la comprobación por cada uno si se ha marcado la casilla de eliminar del formulario de detalle de curso. En caso afirmativo, construye la instrucción SQL para luego realizar la eliminación en la tabla “lista_curso”. Al mismo tiempo, realiza un conteo de instrucciones ejecutadas correctamente para determinar si toda la eliminación ha sido completada o si ha habido algún error. Esto es notificado al usuario una vez terminado el proceso.

Para agregar alumnos a un curso, se generó una pantalla compuesta por un formulario HTML que muestra la lista completa de alumnos utilizando una tabla con Tablesorter, en la que se puede seleccionar a cada estudiante marcando la casilla adjunta a cada uno. Estas se identifican el número de matrícula del alumno. Al hacer clic en aceptar, los datos del formulario son entregados al programa “nuevo_alumno_curso.php”, el cual funciona de manera muy similar al de eliminación, pero con dos diferencias. La primera es la lista obtenida, que en este caso corresponde a los estudiantes que no están presentes en el curso, y la segunda es que en este caso la instrucción a ejecutar es de tipo “INSERT”.

6.6 Estilo CSS

Como archivo de estilo, se utilizó el mismo .CSS creado para la API de gestión de problemas. A éste se le agregaron las entradas “cabeza_inicio” para el título principal de todas las vistas, “panel_aula” para el panel de opciones de la izquierda, y “contenido_aula” para el contenido de cada opción del ambiente de administración del aula virtual.

Capítulo 7: Pruebas y Resultados

7.1 Pruebas técnicas de Software

Para realizar pruebas al software creado, se generó un plan de trabajo consistente en dos partes. La primera propone utilizar el poblado de la base de datos como casos de prueba de la API de gestión de problemas, esto es, crear varios problemas en el editor, y que estos sean los mismos que ya existen como archivos y carpetas. Luego de eso, se propone hacer lo mismo en el ambiente de gestión de aula virtual, creando e iniciando/deteniendo actividades, creando nuevos alumnos, asociándolos y eliminándolos de los cursos existentes, y creando nuevos cursos.

Durante este procedimiento fueron encontrados diversos errores de programación, como problemas al guardar ciertos caracteres. Los detalles de estos problemas se pueden encontrar en el Anexo 5: Errores técnicos encontrados durante las pruebas de software. Todos estos problemas fueron solucionados con éxito.

7.2 Comparativa entre el producto final y el funcionamiento previo

Con el fin de realizar una validación inicial de los resultados obtenidos, se presenta a continuación una comparación entre la nueva interfaz creada y el método anterior de operación del sistema.

La edición de problemas ahora es más simple. Esto debido a que anteriormente los problemas estaban almacenados en archivos y carpetas, donde el nombre de la carpeta era el ID del problema, y por lo tanto no era posible saber qué problema tiene cada una sin tener que entrar a ellas. Esto significa que se debe buscar en cada una leyendo los archivos “nombre.txt” y/o “pregunta.txt” para saber de qué problema se trata hasta dar con el requerido. Más aún, durante la realización de la API de gestión de problemas, se tuvo que construir un pequeño programa en Shell que permitió generar un archivo TXT con la lista de nombres de problemas y sus ID asociados, para así saber en qué consistía cada problema. La nueva lista de problemas que se construyó permite visualizar el nombre de cada problema junto a su ID. En la figura 7.2 de la carpeta de problemas se puede apreciar que es imposible saber de qué se trata cada uno sin antes entrar y ver el nombre o el enunciado de ellos. En el anexo 2 se puede encontrar una vista de la nueva tabla de problemas.

La creación de problemas también se ve afectada por el problema anterior, debido a que al no existir un ambiente de reutilización, era necesario explorar las carpetas hasta encontrar uno que fuera similar a lo que uno necesita. La nueva interfaz, aparte de permitir la búsqueda de un problema similar de manera rápida, permite crear nuevos problemas mediante plantillas prediseñadas.

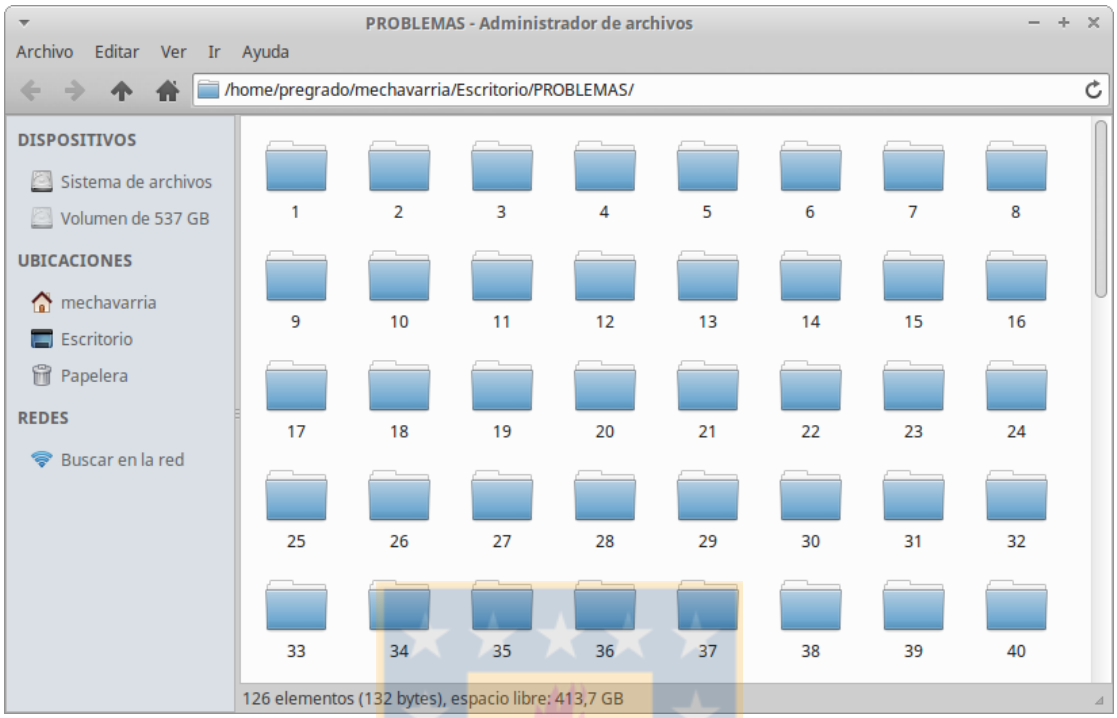

Figura 7.1 Vista de la carpeta de problemas del sistema anterior

Para iniciar una actividad, era necesario conocer el ID de esta, para luego ejecutar un programa en Shell al cual se le entrega como argumento dicho número. Para conocer el ID era necesario entrar a la base de datos y buscar en la tabla actividad la deseada. Para saber qué problemas están asociados a la actividad a iniciar, era necesario relacionar los temas y los problemas y compararlo con el tema asociado a la actividad deseada. Ahora las actividades se pueden iniciar con tan solo un par de clics, y se puede determinar qué problemas están asociados a ella haciendo clic en su nombre, lo cual libera al usuario de tener que acceder a la base de datos.

La creación de una nueva actividad requería conocer en detalle cada una de las tablas y cómo estas se relacionan para saber dónde agregar las entradas correspondientes, tanto como para el registro de la nueva actividad como para los problemas que estarán asociados a ella. Como se mencionó anteriormente, los problemas estaban en carpetas por lo que saber cuál era cada uno era tedioso. Con el nuevo ambiente de administración las consultas necesarias se manejan de manera automática, y el usuario solo tiene que marcar los problemas deseados, identificados por su nombre, y rellenar los campos solicitados. Para facilitar las cosas, además el usuario puede hacer clic en el problema y esto lo cargará en el editor, lo que permite ver sus detalles.

El agregado de alumnos debía hacerse mediante una consulta en MySQL. Luego de la implementación del ambiente, la consulta que agrega los alumnos se genera de manera automática con los datos entregados por el usuario.

También debía hacerse en MySQL la vinculación de un alumno a un curso, para lo cual era necesario conocer su matrícula y el ID del curso. La nueva interfaz permite realizar esta tarea de manera fácil desde la lista de alumnos de un curso, indicándole al sistema su intención y luego marcando a los alumnos del listado, identificados con su matrícula y su nombre, lo que libera al usuario de tener que relacionar matrículas con nombres al realizar las inserciones.

Para resumir la comparación, se puede concluir que los docentes que han utilizado la plataforma deben conocer el sistema muy a fondo, al punto de saber los nombres de las columnas de las tablas para poder realizar consultas, además del tiempo que lleva encontrar un problema en particular. Esto se elimina con la nueva interfaz, debido a que todas las consultas se construyen internamente y el usuario solo completa campos vacíos o marca casillas de verificación para administrar el aula, además los problemas son de fácil navegación a través de una tabla que los identifica por nombre y no por ID

7.3 Validación del producto

Para realizar una validación final del software creado, se mostró el resultado a uno de los docentes del Departamento de Ingeniería Informática y Ciencias de la Computación, y principal usuario de la plataforma. En la ocasión, se mostraron todas las funcionalidades que tiene el sistema, tanto de la API de edición de problemas, como del ambiente de gestión de aula virtual. Su opinión fue que en general le gustó la interfaz.

Para validar el software con otros usuarios, éste fue mostrado a otro profesor del Departamento, indicándole en primer lugar en qué consiste la plataforma de evaluación automática, para luego mostrarle todas las funciones y características creadas, y haciendo hincapié en las mejoras obtenidas con este producto con respecto a la forma de trabajo previa a esta implementación. Para evaluar su percepción del trabajo, una vez concluida la muestra se le planteó las siguientes tres preguntas:

- ¿Qué impresión tiene acerca de la antigua interfaz?
- ¿Cree que usted estaría dispuesto a usar esta plataforma con esa interfaz?
- ¿Qué opinión tiene respecto de la nueva interfaz?

El docente manifestó haber utilizado la plataforma en el pasado pero que dejó de usarla hace cerca de dos años por todos los problemas de administración que conlleva, y que no volverá a usarla en las condiciones actuales de administración. En su opinión, la plataforma está hecha para que solo su creador pueda utilizarla debido a que se debe trabajar muy a bajo nivel, y es tan mala su forma de administrar que cualquier mejora a esto será bien recibido. Su uso fue para cursos de programación en MATLAB, por lo que algunas funciones implementadas no aplican para este caso, sin embargo, manifestó que si al producto creado en este proyecto se agregan funciones para trabajar con dicho

lenguaje, consideraría volver a utilizarla, dado que elimina la necesidad de trabajar a bajo nivel. A raíz de esta muestra, se sugirió crear una nueva forma de agregar alumnos, mediante una planilla Excel.

Se logró obtener también la opinión de un alumno que actualmente es ayudante en la asignatura Programación II, ramo donde se utiliza la plataforma pero en lenguaje JAVA. Él manifestó conocer parcialmente la forma de administrar la plataforma. Su opinión fue que *"es un poco complicado llegar a las cosas para alguien externo […], alguien que no conoce el sistema se complica"*. Indicó que él como docente utilizaría la plataforma pero le tomaría un tiempo para aprender a utilizarla. Con respecto a la nueva interfaz, señaló que es mucho mejor que lo que estaba antes, que es mucho más fácil llegar a las cosas y que es más intuitivo que trabajar a bajo nivel, además indicó que le tomaría mucho menos tiempo aprender a utilizarla con esta nueva interfaz.

Conclusiones

Se ha conseguido crear un conjunto de herramientas que facilitan la gestión de la Plataforma al Apoyo Docente con Evaluación Automática con que cuenta el Departamento de Ingeniería Informática y Ciencias de la Computación de la Universidad de Concepción. Forman parte de este conjunto una API para edición de problemas construida en HTML y PHP, y una interfaz web de gestión de alumnos, cursos y actividades.

Existe dentro de la API de edición de problemas un editor de probadores dinámicos para el cual fue necesario crear una plantilla. En base a los problemas existentes observados, se puede decir que existen alrededor de cuatro plantillas que pueden ser creadas, sin embargo para efectos de prototipo solo se generó una. Queda como trabajo futuro deducir y crear más plantillas.

También existe dentro de la API un editor de probadores estáticos, para el cual se implementó solo un editor de código. En base a lo observado en los problemas existentes, estos probadores utilizan una serie de filtros con los cuales se puede generar una biblioteca de filtros y realizar una mejora al editor incorporando este repositorio, lo cual queda como trabajo futuro.

Un punto importante que se observó es que la plataforma existente está implementada para trabajar teniendo los problemas como archivos y carpetas, por lo que para acceder a los problemas almacenados en la nueva base de datos habrá que modificar la plataforma para que se adapte a ella. Esto podría implicar la realización de cambios en los probadores estáticos y dinámicos para que también se adapten a esta nueva forma de trabajo. Para evitar esto, se sugiere realizar una función que, al iniciar una actividad, escriba los problemas desde la base de datos a la carpeta PROBLEMAS, y que al finalizarla éstos sean borrados.

Con respecto al uso del sistema para otros lenguajes, hay ciertas modificaciones que se deben hacer, como por ejemplo un creador de temas, y una diferenciación a cada problema para saber a qué curso o a qué lenguaje pertenecen, ya que al crear una actividad para, por ejemplo MATLAB, salen también los problemas de C.

Otro punto importante es que dado que no se realizó un diseño previo a la implementación de la plataforma existente, realizar modificaciones en ella resulta complejo, por lo que en algún momento será necesario realizar un rediseño documentado de ella.

La adaptación de la metodología Scrum mostró funcionar bien para ese proyecto, debido a que ha permitido desarrollar ambos entornos de administración de manera correcta y ordenada, esto es, logrando desarrollar primero una etapa y luego la otra. Se debe considerar además la inexperiencia del equipo de trabajo, compuesto por una sola persona, en el desarrollo sobre sistemas ya existentes.

Referencias

[1] Ammann, Paul, Offutt, Jeff. ***"Introduction to Software Testing"*** Cambridge University Press, Enero 2008, ISBN 9780521880381

[2] López Reguera, Jorge, Hernández Rivas, Cecilia and Farrán Leiva, Yussef. "***Una plataforma de evaluación automática con una metodología efectiva para la enseñanza/aprendizaje en programación de computadores***" Ingeniare. Rev. chil. ing., Agosto 2011, vol.19, no.2, p.265-277. ISSN 0718-3305

[3] Herrera Rocha, Oscar. **"Construcción de sistema Game-Engine orientado al desarrollo de videojuegos en perspectiva 2.5D"** Universidad de Concepcion, Facultad de Ingeniería, Departamento de Ingeniería Informática y Ciencias de la Computación, Mayo 2010

[4] Sitio web de ayuda de Ubuntu, donde se detallan los requerimientos de sistema: https://help.ubuntu.com/community/Installation/SystemRequirements

Anexo 1: Casos de uso de la plataforma de administración de problemas

Caso de uso	Cambiar a editor avanzado.	
ID	3.	
Precondición	El docente está editando o creando un probador dinámico usando el editor básico.	
Resultado	El docente utiliza el editor avanzado.	
Actores e Intereses	Docente: desea usar editor avanzado.	
Actores principales	Docente.	
Secuencia normal	**Paso**	**Acción**
	1	El docente escoge editar el probador con el editor avanzado.
	2	El sistema le recuerda al docente que si realizó cambios debe guardarlos, y le pregunta si desea continuar.
	3	El docente ya guardó cambios y escoge continuar.
	4	El sistema le muestra al docente el probador en el editor avanzado.
Excepciones	3	El docente no ha guardado cambios por lo que escoge no continuar para guardar cambios.
Comentarios		

Caso de uso	Crear otra solución de referencia.	
ID	4.	
Precondición	El docente se encuentra en el editor de soluciones.	
Resultado	El docente ha creado una nueva solución de referencia.	
Actores e Intereses	Docente: desea crear una nueva solución de referencia.	
Actores principales	Docente.	
Secuencia normal	**Paso**	**Acción**
	1	El docente escoge crear una nueva solución.
	2	El sistema le recuerda al docente que si realizó cambios debe guardarlos, y le pregunta si desea continuar.
	3	El docente ya guardó cambios y escoge continuar.
	4	El sistema crea la nueva solución y le muestra al docente el editor con la nueva solución en blanco.
	5	El docente crea la solución y guarda los cambios.
Excepciones	3	El docente no ha guardado cambios por lo que escoge no continuar para guardar cambios.
Comentarios		

Anexo 2: Vistas reales de la API de edición de problemas.

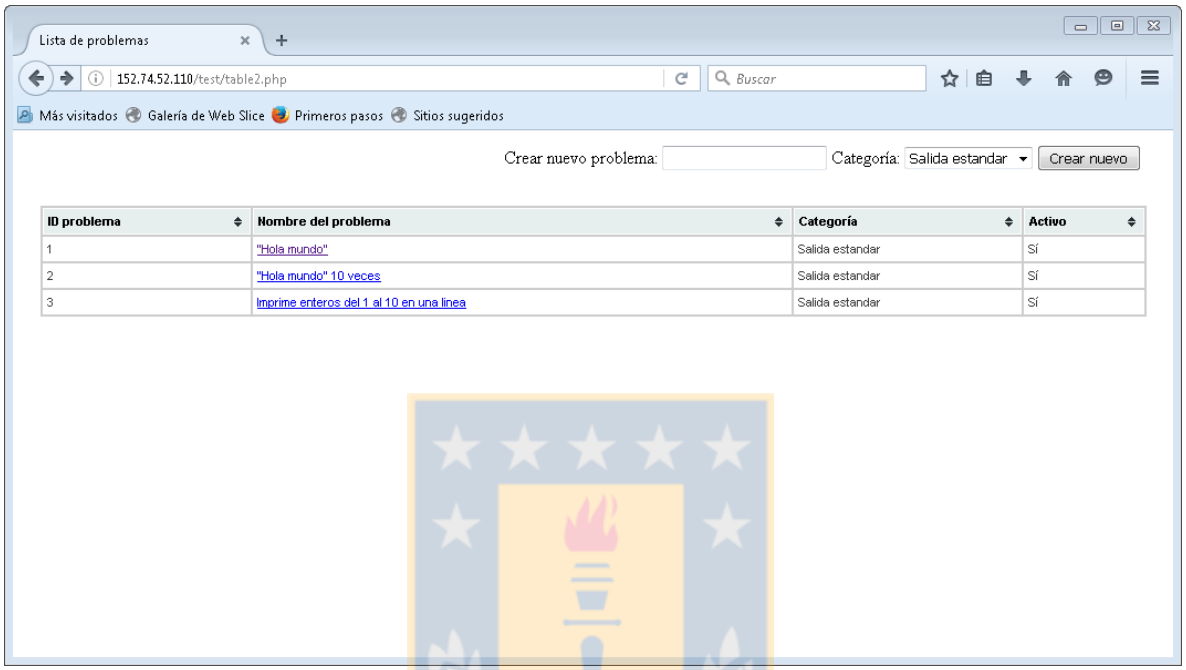

Figura 22.1 Vista real de la lista de problemas

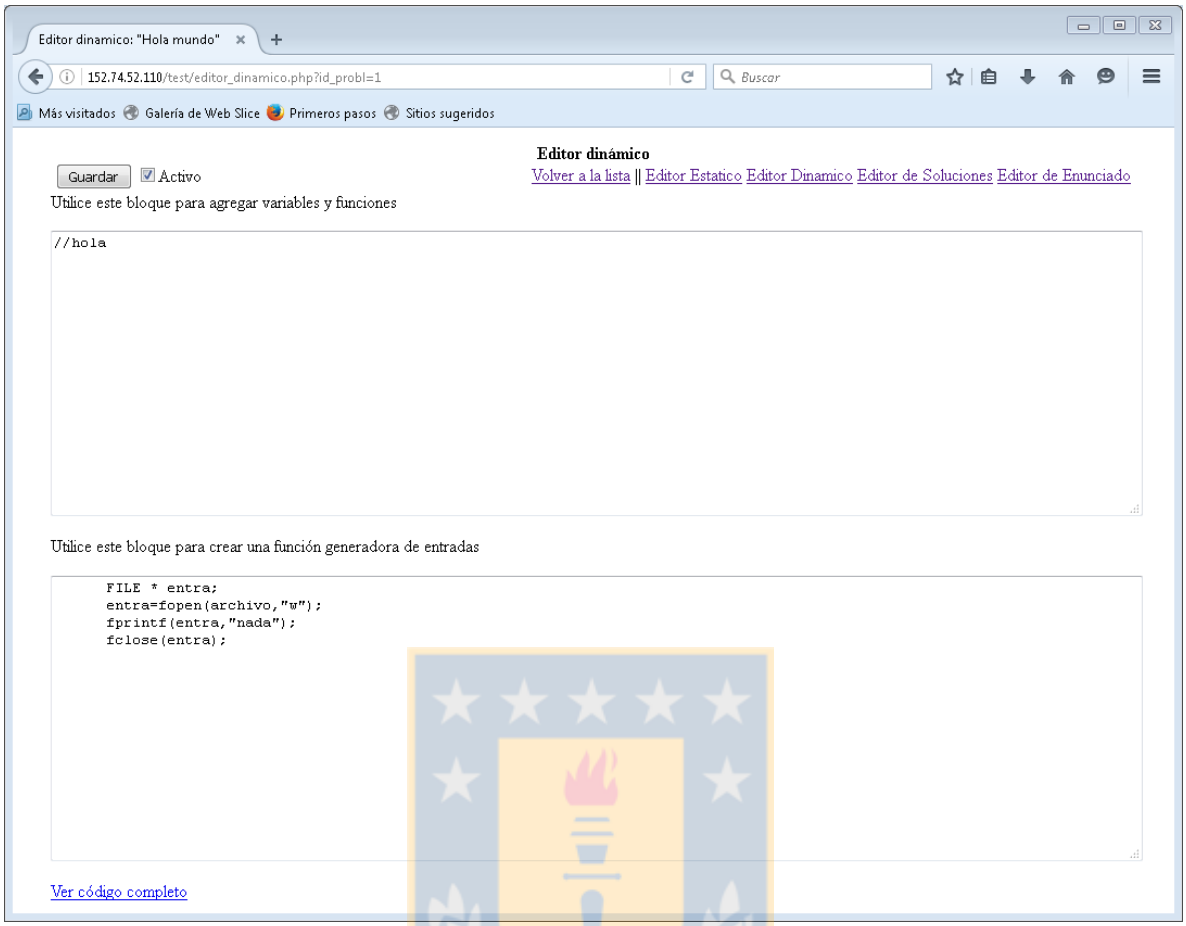

Figura 22.2 Vista real del editor básico de probadores dinámicos

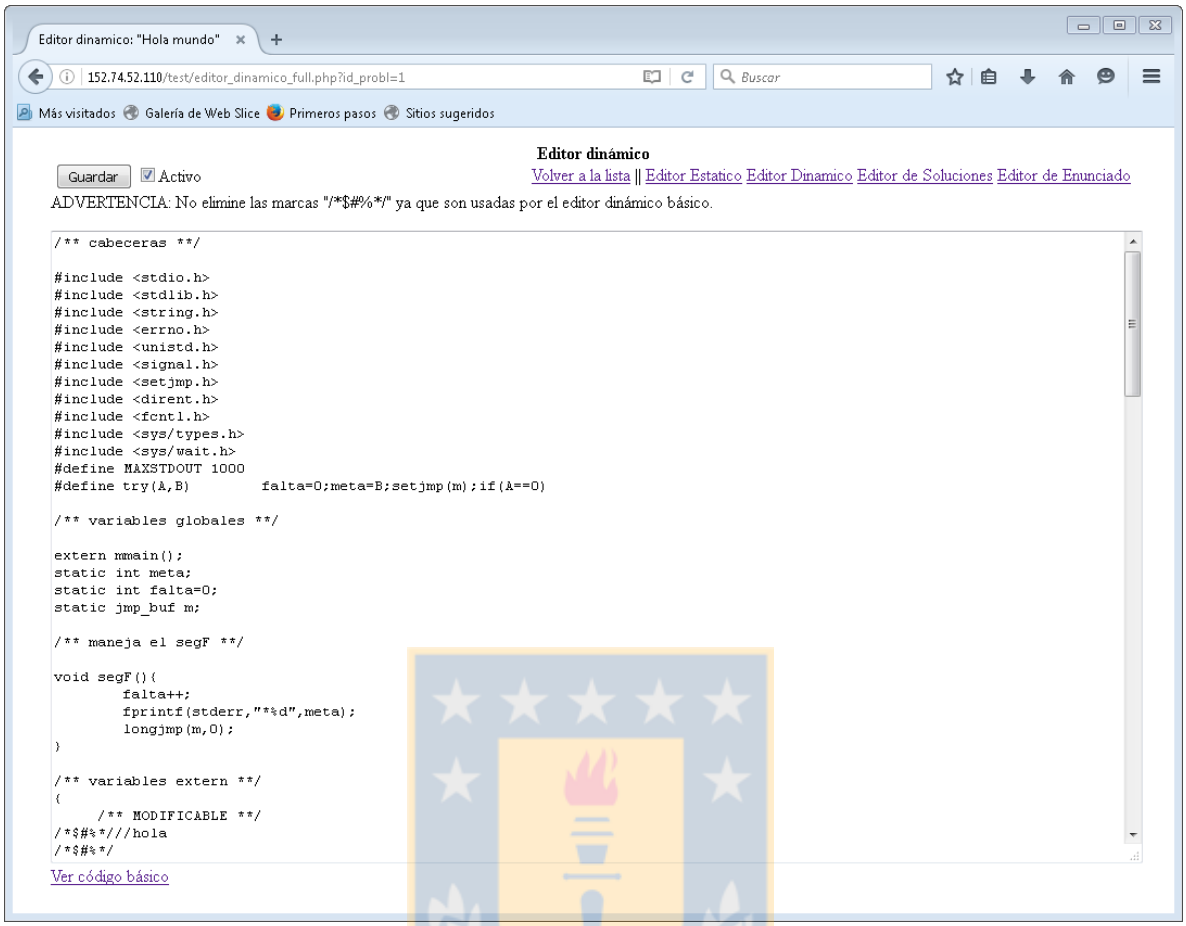

Figura 22.3 Vista real del editor completo de probadores dinámicos

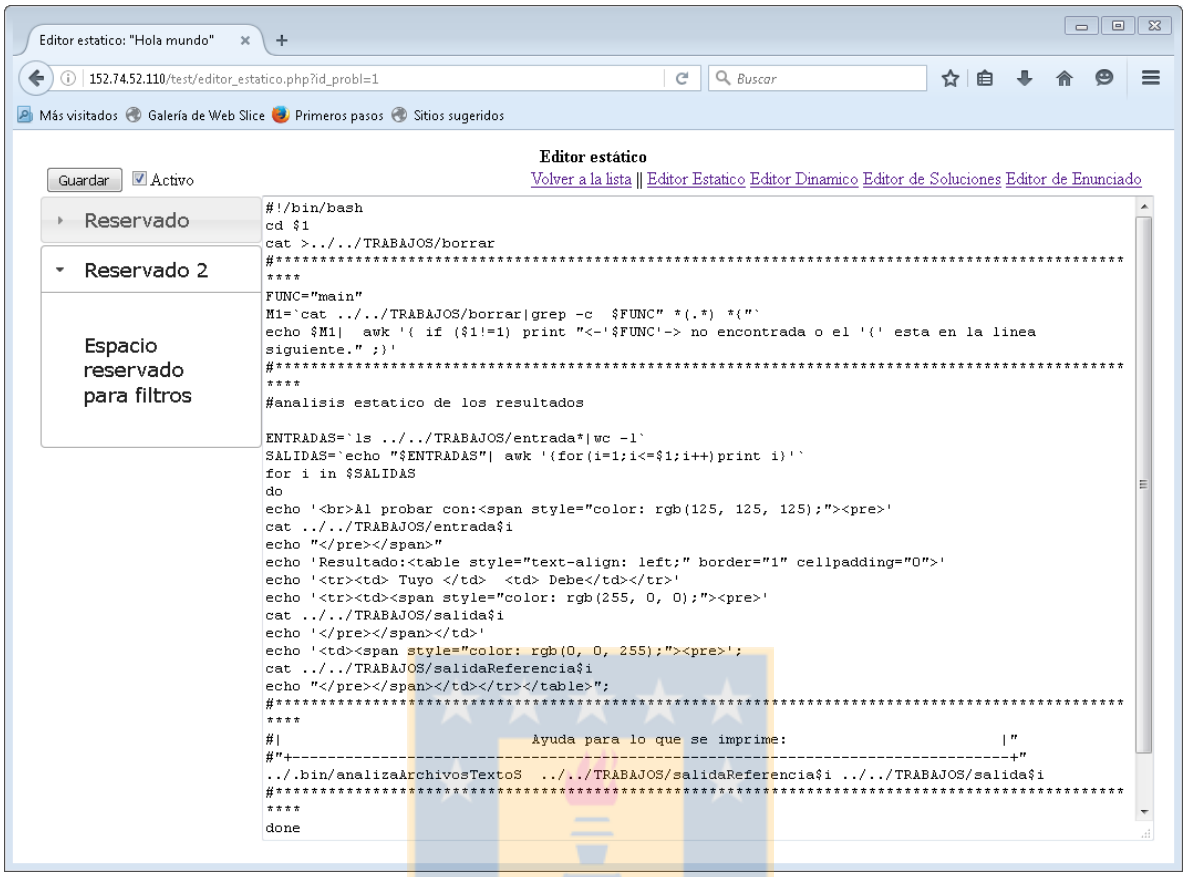

Figura 22.4 Vista real del editor de probadores estáticos

Figura 22.5 Vista real del editor de enunciados

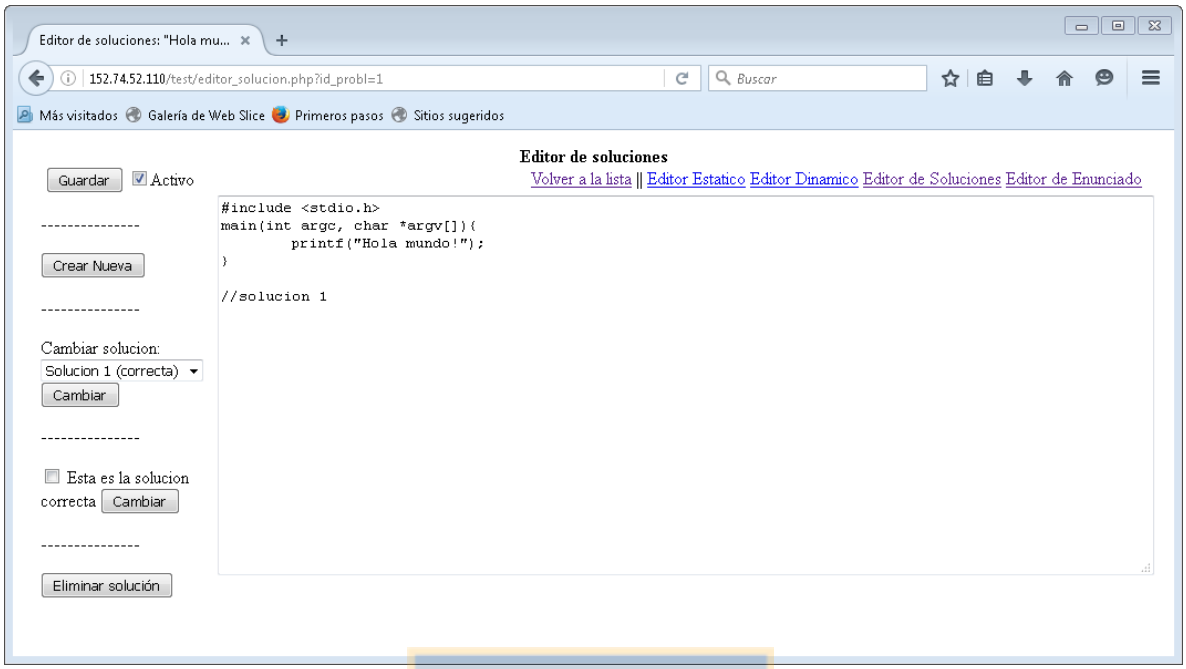

Figura 22.6 Vista real del editor de soluciones

Anexo 3: Casos de uso de la plataforma de administración del aula virtual

Caso de uso	Crear curso.	
ID	4.	
Precondición	El docente está viendo la lista de cursos.	
Resultado	El docente crea un nuevo curso.	
Actores e Intereses	Docente: quiere ingresar un nuevo curso al sistema.	
Actores principales	Docente.	
Secuencia normal	**Paso**	**Acción**
	1	El docente ingresa los datos del nuevo curso y confirma la creación.
	2	El sistema crea el curso en la base de datos y le muestra al docente la lista de los alumnos.
	3	El docente selecciona los alumnos que pertenecerán a este curso marcando las casillas de verificación correspondiente y envía la información al sistema.
	4	El sistema asocia los alumnos al nuevo curso y le indica al docente que se ha completado la operación.
Excepciones		Ninguna.
Comentarios		

Caso de uso	Agregar alumnos a un curso.	
ID	5.	
Precondición	El docente está viendo la lista de cursos. El curso ya está creado.	
Resultado	El docente agrega uno o varios alumnos a un curso.	
Actores e Intereses	Docente: quiere agregar alumnos a un curso existente.	
Actores principales	Docente.	
Secuencia normal	**Paso**	**Acción**
	1	El docente selecciona el curso al que desea modificar.
	2	El sistema le muestra la lista de alumnos actual del curso.
	3	El docente le indica al sistema que desea agregarle alumnos al curso.
	4	El sistema le muestra el listado de alumnos sin los que ya están en el curso, para que escoja a los nuevos.
	5	El docente marca las casillas verificadoras de cada alumno que desee agregar y confirma la operación.
	6	El sistema le indica al docente que se guardaron los cambios y le muestra el listado actualizado.
Excepciones	4	Todos los alumnos del sistema ya están en el curso, por lo que la lista se genera vacía; se le indica al docente esto y se le devuelve inmediatamente a la lista del curso
Comentarios		

Caso de uso	Eliminar alumnos de un curso.	
ID	6.	
Precondición	El docente está viendo la lista de cursos.	
Resultado	El docente elimina uno o varios alumnos a un curso.	
Actores e Intereses	Docente: quiere eliminar alumnos de un curso existente.	
Actores principales	Docente.	
	Paso	**Acción**
	1	El docente selecciona el curso al que desea removerle alumno.
	2	El sistema le muestra la lista de alumnos actual del curso.
	3	El docente marca las casillas verificadoras de los alumnos que serán eliminados del curso y pulsa el botón de eliminar.
	4	El sistema quita los alumnos del curso y le indica al docente que la operación tuvo éxito.
Excepciones		Ninguna
Comentarios		

Anexo 4: Vistas reales del ambiente de gestión de la plataforma de evaluación.

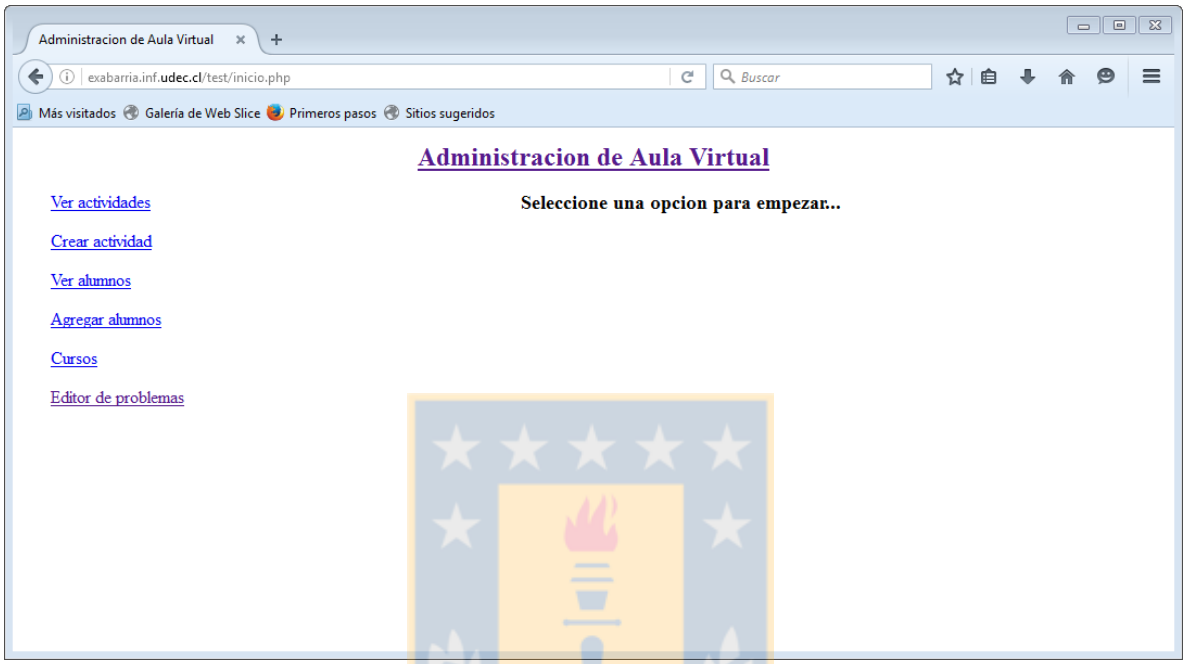

Figura 24.1 Vista real inicial del ambiente de administración del aula virtual

Figura 24.2 Vista real de la lista de actividades

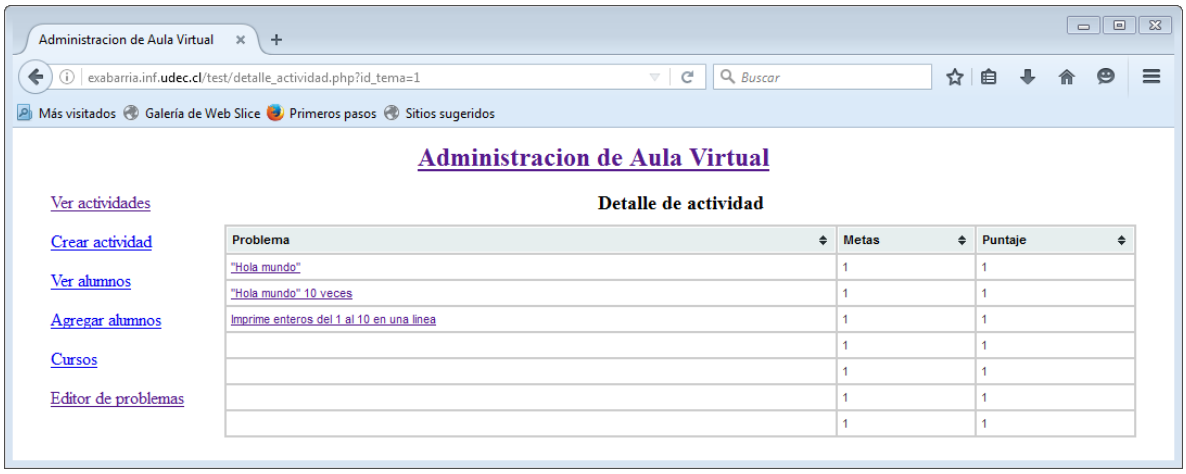

Figura 24.3 Vista real del detalle de una actividad

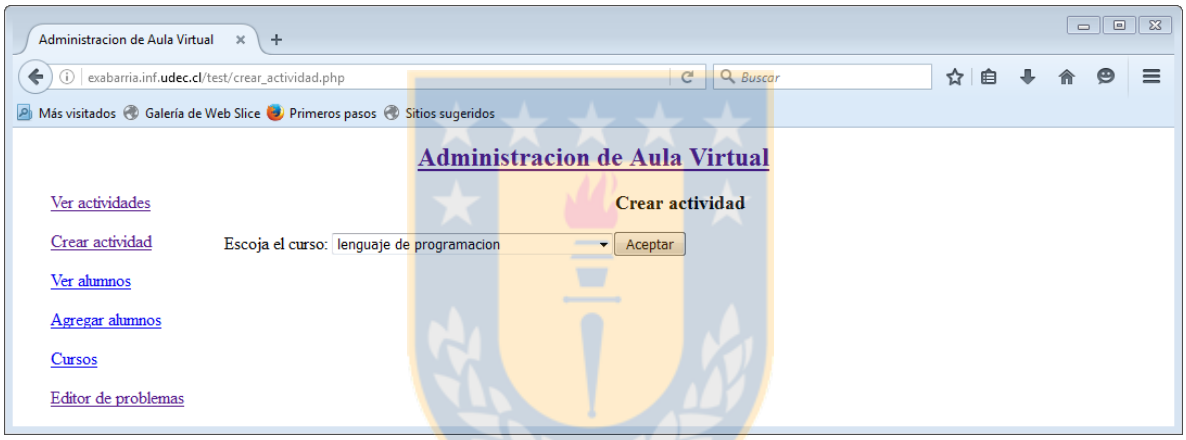

Figura 24.4 Primera vista real de crear actividad

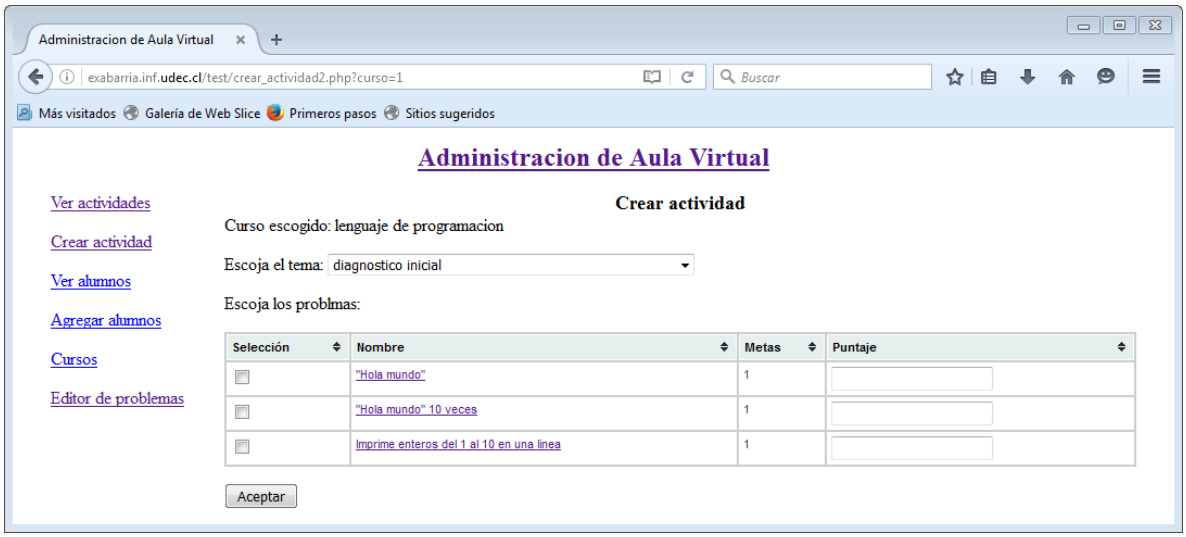

Figura 24.5 Segunda vista real de crear actividad

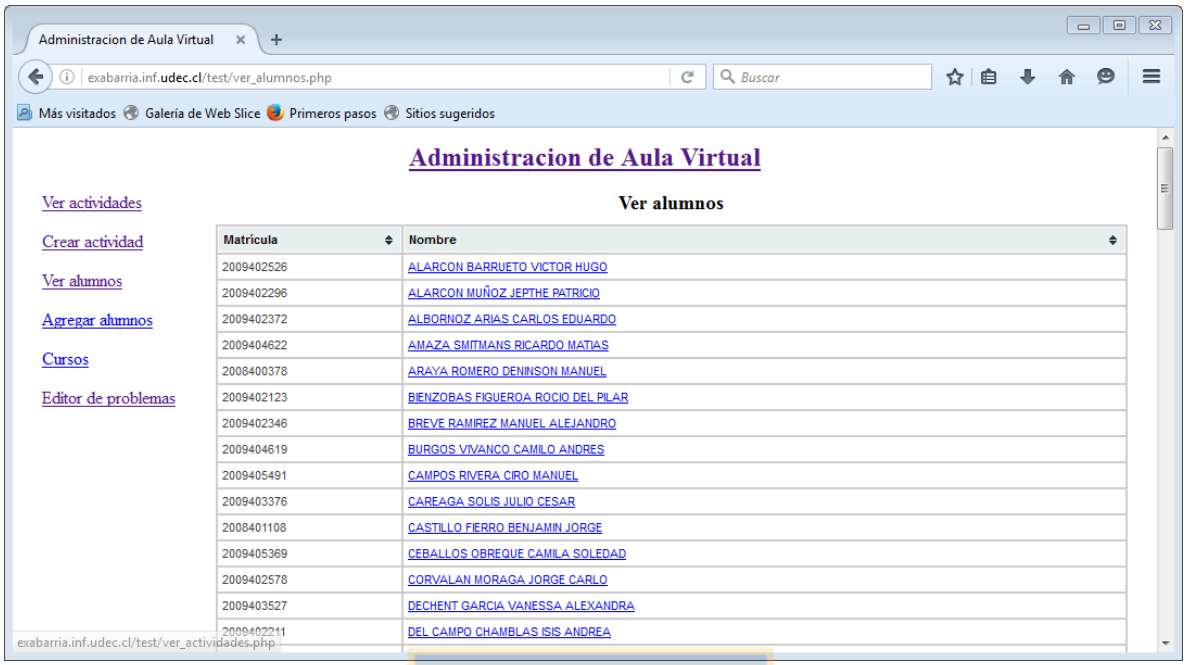

Figura 24.6 Vista real de la lista de alumnos

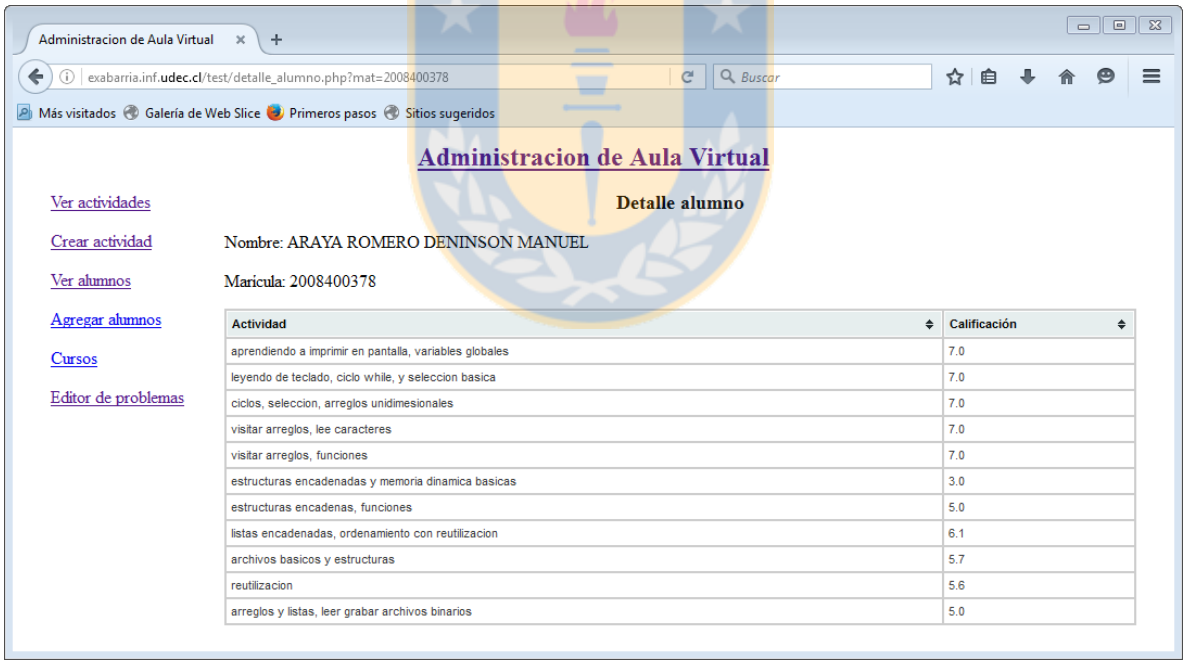

Figura 24.7 Vista real del detalle de un alumno

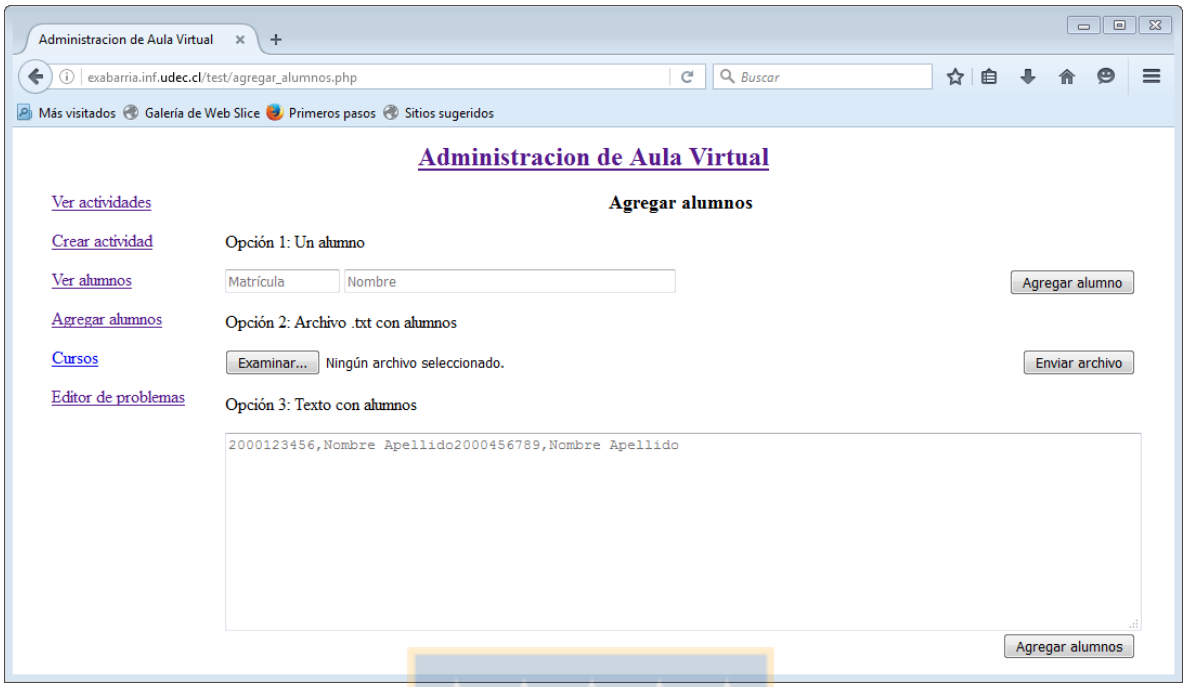

Figura 24.8 Vista real de agregar alumnos

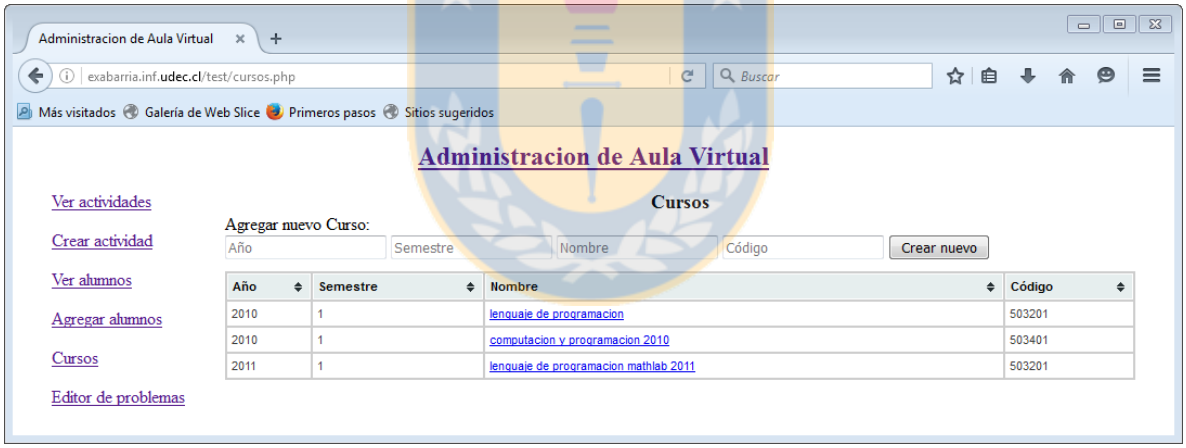

Figura 24.9 Vista real de la lista de cursos

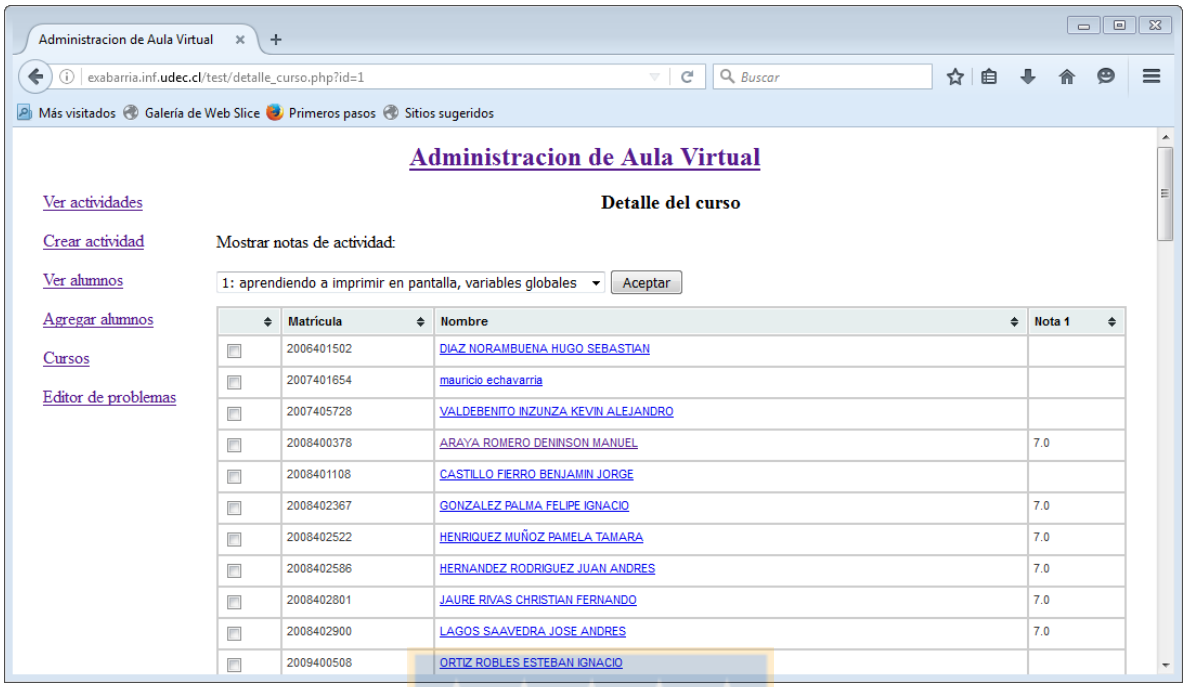

Figura 24.10 Vista real del detalle de un curso
No se aprecian los botones al final de la lista

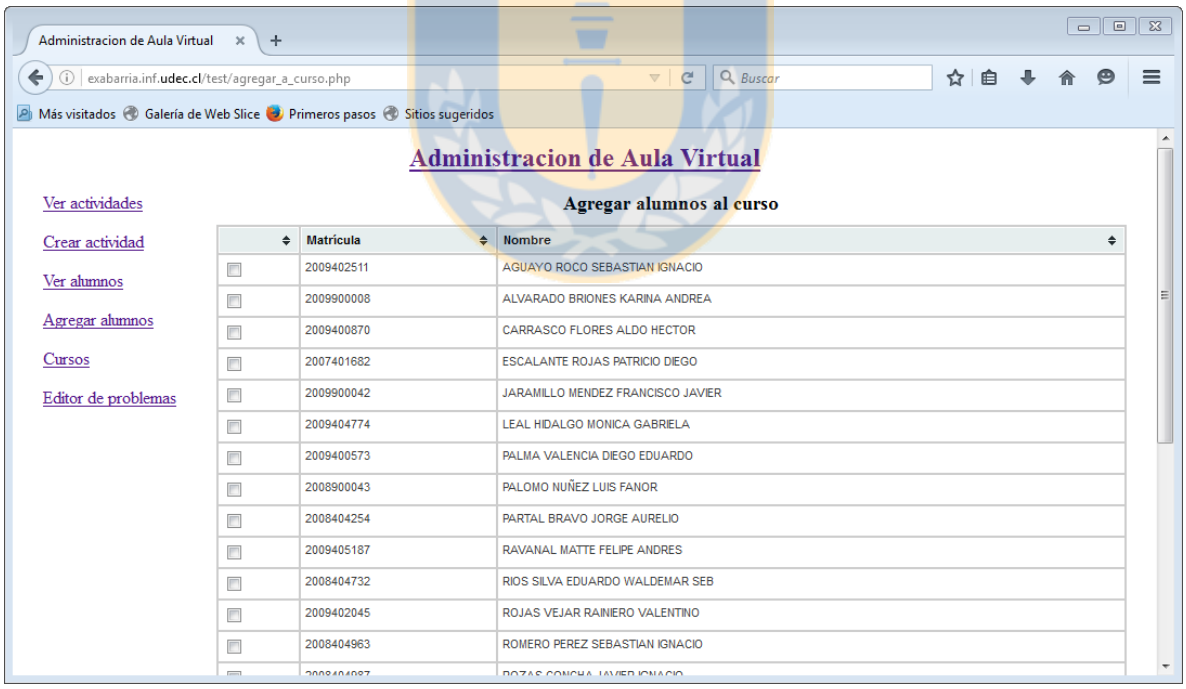

Figura 24.11 Vista real de agregar alumnos a un curso
No se aprecia el botón de aceptar al final de la tabla

Anexo 5: Errores técnicos encontrados durante las pruebas de software.

- Al iniciar el poblado de la base de datos, se observó que algunos problemas generaban error al ser guardados, específicamente al guardar los probadores estáticos y dinámicos. Luego de un análisis al programa de guardado y a los mismos probadores, se descubrió que las comillas simples y dobles generaban una consulta SQL ambigua, dado que las comillas contenidas en el código eran interpretadas como parte del lenguaje SQL y no como parte del texto a guardar. Para solucionar esto, se tuvo que modificar el programa *"guardar.php"* agregándole lo que se denominó como "limpiador de contenido". Esto consiste en analizar el código fuente del probador en búsqueda de comillas simples para anteponerles el carácter "\", Esto provoca que MySQL lea estas comillas como parte del texto que se debe guardar y no como parte de la sintaxis. La consulta está construida con comillas dobles, por lo que solo las comillas simples interfieren, y es por esto que solo estas últimas son modificadas.
- Se observó que el error anterior también afectaba a los caracteres "\" contenidos en algunas instrucciones como *"printf"*. Por lo que también se agregó al limpiador de contenido la búsqueda de "\\" para cambiarlos por "\\\\".
- También se observó que durante la construcción del editor dinámico y de la plantilla generada, esta se alteró debido al mismo error anterior, por lo que tuvo que ser remplazada por un respaldo.
- Al abrir el ambiente de gestión de aula virtual desde un computador, no había ningún problema, pero al abrirlo desde dispositivos con pantallas pequeñas como la de un teléfono móvil, éste se deformaba, quedando el panel derecho por debajo del izquierdo, como se aprecia en la figura 7.1. Esto se debió a que en el archivo CSS se definieron los anchos de los espacios como pixeles fijos y no como porcentajes. Realizando este cambio se solucionó el problema. Esto también afectaba a las vistas con pantalla dividida de la API de edición de problemas.
- Luego de agregar algunos problemas, se procedio a probar el funcionamiento de la creación de actividades, instancia en la que se observó que en vez de mostrar solo los problemas activos, se mostraban todos. Para solucionar esto, se modificó el programa *"crea_actividad2.php"* agregándole una comprobación al inicio del ciclo que imprime los problemas, que pregunta si el campo "activo" del problema está en cero, y en caso afirmativo se salta la iteración mediante la instrucción *"continue"*.

Figura 25.1 Capturas de pantallas del error de dimensión en pantallas pequeñas.
A la izquierda se muestra el error. A la derecha, se muestra problema solucionado.

- Un error menor que fue encontrado consiste en las múltiples escrituras de variables utilizadas para *debug* que quedaron activas. Estas líneas de código no fueron eliminadas, sino solo comentadas.
- Al agregar o eliminar alumnos de un curso, se muestra un mensaje de confirmación con la cantidad de estudiantes agregados correctamente y también con la cantidad de errores presentados. Este conteo tenía un error que provocaba números negativos cuando se producía algún error. Esto fue corregido modificando la forma en que son contadas las operaciones correctas y erróneas.
- Al crear una nueva actividad, el sistema le pregunta al usuario por la cantidad de puntos que tendrá asociado cada problema. Esto, en conjunto con las metas, eran asignados por defecto como 1, por lo que daba igual el valor que el usuario deseara, siempre quedaban ambos valores en 1. Esto fue corregido desactivando el valor por defecto.