Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Desarrollo de un portal web para la dinamización de resolución de problemas de programación

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Vicent Castell Puertas

Tutor: Alicia Villanueva

2018-2019

Resumen

Este proyecto explica todo el proceso de diseño y creación de una aplicación para la ayuda y motivación de los alumnos con pocos conocimientos de programación. Representa una pequeña competición, a nivel de asignaturas, en la que los alumnos realizan una serie de ejercicios propuestos por el profesor con el fin de obtener la puntuación más alta. Esta plataforma está enfocada a los ejercicios de programación en lenguajes poco comunes impartidos en la universidad como pueden ser Haskell o Prolog, aunque no existe ninguna restricción a ningún lenguaje, ya que la aplicación no ejecuta código.

Creemos que crear un entorno de sana competición entre los alumnos, puede ampliar tanto la motivación a la hora de aprender por parte de los alumnos como las relaciones sociales en el grupo de clase, evitando que los alumnos puedan tener una mejor experiencia a la hora de aprender nuevos lenguajes.

Palabras clave: Plataforma web; enseñanza de lenguajes de programación; xampp; jsp; mysql; NetBeans; UML; Servlet; controlador.

Abstract

This project explains the entire process of designing and creating an application improve the students motivation toward programming languages learning with little knowledge about programming. It represents a sort of competition, in which students solves a series of exercises proposed by the teacher in order to obtain the highest score. This platform is focused on programming exercises in languages different from Java which is the first programming language that is introduced in the GII of the UPV such as Haskell or Prolog, although there is no restriction to any language.

We think that by a healthy environment of competition among them can improve both the students motivation when learning and the social relationships in the class group, building a better experience for students when learning a new programming language.

Keywords : Web Platform; programming languages learnings; xampp; jsp; mysql; NetBeans; UML; Servlet; controller.

Resum

Aquest projecte explica tot el proces de diseny i creació d'una aplicació per a l'ajuda i motivació dels alumnes amb pocs coneixements de programació. Representa una xicoteta competició, a nivell d'assignatures, en la qual els alumnes realitzen una serie d'exercicis, proposats pel professor amb el fí d'obtindre la puntuació mes alta. Aquesta plataforma està enfocada als exercicis de programació amb llenguatges poc comúns impartits a la Universistat com poden ser Haskell o Prolog, encara que no existeix cap tipus de restricció respecte al llenguatge, ja que no ejectuta còdi.

Creiem que crear un entorn de sana competició entre els alumnes pot ampliar tant la motivació a l'hora d'aprendre per part dels alumnes com les relacions socials en el grup de classe, evitant que els alumnes puguen tindre una millor experiència a l'hora d'aprendre nous llenguatges.

Paraules Clau: Plataforma web; ensenyança de lenguatges de programació; xampp; jsp; mysql; NetBeans; UML; Servlet; controlador.

Agradecimientos

Agradezco a muchas personas que este trabajo se haya podido realizar, son tantas las personas a las que debería mencionar que no se si voy a poder mencionarlas a todas.

En primer lugar, querría agradecer especialmente toda esta experiencia y todo este trabajo a mi profesora, ayudante y participe de este TFG a mi profesora Alicia Villanueva ya que sin ella no habría podido realizar y terminar todo esto. Agradecerle tanto la ayuda en este proyecto como durante todo mi periodo en la universidad desde que me impartió una asignatura, me ha ayudado tanto en aspectos de la propia asignatura como en asuntos ajenos a ella dentro de la universidad, a parte de brindarme la oportunidad de ser mi tutora de prácticas de empresa, cosa que me ha ayudado y mucho a meterme en este maravilloso mundo.

Agradecer también, como no, a mis padres por todo el apoyo, porque, aunque yo he decaído varias veces, ellos siempre han estado ahí para ayudarme a poder seguir y poder cultivarme un futuro en el mundo de la informática. Gracias por todos esos momentos de ayuda, por todas esas veces que hemos discutido, por mi bien, que he han hecho abrir los ojos en ciertos aspectos.

A mi novia, por estas siempre a mi lado, por todo el apoyo y todas las horas dedicadas a la búsqueda de información y por todos los momentos que me ha ayudado a levantarme cuando no podía mas en situaciones durante el curso que me planteaba dejar de estudiar esta carrera

Y sobre todo, a esos compañeros de la universidad, Antoni, Dani, David, Raquel, Fenollar, Altea, Ricardo, … y muchos más, por todos los momentos vividos en esta estupenda etapa de mi vida. Amigos que estarán ahí siempre y que me tendrán para lo que necesiten.

Mención especial para Sergio Vacas y Juan Manuel Isaac, por esas horas dedicadas en Discord y Anydesk ayudándome a resolver dudas ya perfeccionar la aplicación, saben que después de la entrega de este trabajo, su deuda será saldada con una buena cena.

Seguramente me deje a alguien, porque en esta etapa universitaria he conocido a mucha gente fantástica que han hecho mas llevadera este tramo de mi vida.

Tabla de contenidos

Contenido

Índice de Tablas

Índice de Ilustraciones

1. Introducción

Este trabajo está inspirado en la plataforma de internet llamada https://adventofcode.com/ [1], cuyo propósito es que un conjunto de personas compita entre ellos resolviendo diferentes problemas del ámbito de la programación.

Nuestro objetivo es implantar un sistema parecido en el ámbito estudiantil de una asignatura de la UPV. Este proyecto puede servir para incentivar a los alumnos a competir entre ellos, a nivel de grupos, para ver quién es capaz de resolver más problemas o de llegar más lejos en la "competición".

El funcionamiento es simple: el profesor da de alta una serie de enunciados. Cada uno de estos ejercicios tendrá asociados una serie de test de entrada-salida. Desde el punto de vista del estudiante, cada problema a resolver consta de un enunciado con su respectiva entrada (input).

El alumno tiene acceso a todos los enunciados que el profesor decida mostrar. El profesor puede esconder los ejercicios para poder editarlos. El alumno, al acceder a un ejercicio disponible puede ver el enunciado junto a un parámetro de entrada(input).

El alumno resolverá el problema utilizando herramientas que quedan al margen de la plataforma. Cuando haya resuelto el ejercicio, deberá introducir su respuesta. Si la respuesta es correcta, es decir, coincide con la asignada por el profesor a ese parámetro de entrada, el alumno obtendrá su correspondiente puntuación y podrá intentar resolver otro ejercicio de los mostrados por el profesor. En caso de no ser correcto, puede volver a intentar solucionar el ejercicio sin ningún tipo de restricción.

Es importante remarcar que el sistema no compila código en la plataforma. El sistema solo compara los valores introducidos por el profesor con los introducidos por el alumno. En un apartado hablamos de esta alternativa, la cual consideramos una desventaja porque aumenta el coste computacional del programa, el mantenimiento es mucho más costoso y solo nos permite centrarnos en un único lenguaje de programación.

El profesor puede consultar en todo momento el progreso de los alumnos que han ingresado en la aplicación, así como la cantidad de usuarios que han resuelto cada problema. Los propios alumnos tienen acceso a un ranking entre todos ellos.

Esta aplicación es un mero acompañamiento a la enseñanza de un lenguaje de programación. En un futuro podría ser útil para aprender a programar, incluso puede ser utilizado para decidir aspectos de la nota del alumno, aunque este no es su principal propósito.

1.1Motivación

Uno de los motivos principales por los que me decidí a realizar este trabajo, fue la utilidad a nivel educativo que se le puede dar, y la ayuda que puede ofrecer a futuros alumnos de asignaturas de programación que, como yo, cuando entraron a la carrera desconocían cualquier tipo de lenguaje.

Cuando se me planteó esta idea, pensé en las dificultades que tuve para poder realizar asignaturas como Programación (PRG) o Lenguajes, tecnologías y paradigmas de la programación (LTP). Esta herramienta puede ayudar y puede motivar a gente que tiene dificultad en estos campos.

Dependiendo del planteamiento del profesor, esta aplicación puede llegar a ser una competición entre los alumnos, permitiendo aumentar las relaciones sociales con sus compañeros, como a la vez motivarse para trabajar un poco y conseguir más puntos que otro alumno.

A nivel de profesorado, esta herramienta no está pensada para que sea considerada en la nota del alumno, pero sí que se podría llegar a utilizar como ayuda para la toma de decisiones de un profesor a la hora de incidir en algún tipo de ejercicio problema para el que los alumnos tengan más dificultades.

1.2 Objetivos

El objetivo general de este proyecto es el desarrollo de una plataforma web para incentivar a los alumnos de las asignaturas de programación a seguir programando, así como ayudar a fomentar las relaciones sociales de dichos alumnos creando un ámbito sano de competición entre ellos.

Los objetivos específicos para lograr el objetivo general son los siguientes:

- Crear una plataforma atractiva para el alumno, que lo motive a seguir practicando
- Poder empezar a implantar elementos que puedan facilitar la introducción de dicha aplicación en la plataforma PolifotmaT.
- Guardar datos propios de cada alumno dentro de la base de datos, con el objetivo de que cada uno de los alumnos tenga unos parámetros asignados dentro de cada ejercicio.
- Asignar de forma aleatoria a los alumnos los inputs creados por el profesor para evitar la posible falsificación de datos por parte de los usuarios.

1.3 Metodología Utilizada

Para este trabajo hemos decidido utilizar una metodología de diseño de software denominada metodología en cascada.

La metodología en cascada es un modelo lineal de diseño de software en el cual se emplea un proceso secuencial, es decir, se divide todo el proceso de diseño en varias fases, cada una dependiente directamente de la anterior.

El proceso definido en la metodología en cascada es el siguiente:

1. Análisis de requisitos.
2. Diseño.
3. Implementación.
4. Verificación.
5. Mantenimiento.

Desde un primer momento se hicieron reuniones para establecer los requisitos que debía cumplir nuestra aplicación. Seguidamente, una vez establecidos, se procedió al diseño tanto del sistema como del programa, todo esto dependiendo de los requisitos previamente especificados. Una vez teníamos hecho el diseño, se dio paso a la fase de implementación, la cual una vez finalizada fue entregada a la persona responsable para la fase de verificación, en la cual el cliente se encarga de comprobar que se ha realizado todo en base a lo establecido anteriormente. Si todo está bien y todo funciona correctamente, se pasa a la fase de mantenimiento, en la cual se trabaja en la resolución de posibles errores. Para la implementación de alguna ampliación o alguna modificación importante de funcionalidad, utilizamos el mismo método utilizado hasta ahora, pero solo lo aplicaremos a las modificaciones necesarias del sistema.

1.4 Impacto Esperado

En el ámbito estudiantil, no tenemos ninguna plataforma o aplicación que nos permita plantear este tipo de inferencia. La plataforma “https://adventofcode.com/” que nos permite hacer algo parecido, pero en un nivel mucho más avanzado en cuanto a retos y problemas plantados y para nada integrado en el mundo estudiantil. Con la creación de esta plataforma, pretendemos fomentar el conocimiento y ayudar a los alumnos con conocimientos de programación a realizar ejercicios sencillos, con el fin de obtener una puntuación en un ranking global entre los alumnos.

Desde el punto de vista del profesor, puede ser una herramienta para crear un ambiente agradable que fomente la participación y la mejora continua en las clases que imparte.

2. Estado del arte

Uno de los principales propósitos de este TFG es conseguir que nuestro sistema esté integrado en la lista de aplicaciones disponibles en la Learning Tools Interoperability (LTI), cumpliendo el estándar de comunicación establecido por Sakai, ya que actualmente no existe ningún sistema en esta lista con las características que nosotros planteamos.

Todo esto nos permitiría poder implantar este trabajo como un plug-in en la aplicación propia de la Universitat Politècnica de València, PoliformaT.

2.1 Learning Tools Interoperability

El Learning Tools Interoperability (LTI) [2] es un estándar de especificaciones utilizado para permitir la integración de aplicaciones de terceros en plataformas educativas.

En lugar de que cada aplicación de sistemas se desarrolle en base a un sistema de gestión de aprendizaje (LMS) [3] teniendo que gestionar el mantenimiento y adaptando cambios en base a modificaciones de la plataforma, LTI permite que nuestro sistema sea compatible con todo tipo de plataformas LMS (que acepten LTI), siendo muy sencillo aplicar la aplicación a cualquier sistema educativo.

Para entenderlo mejor podríamos comparar esta aplicación con un dispositivo USB. Hace unos años cada tipo de periférico o sistema de almacenamiento externo en un PC tenía sus propias especificaciones y puertos de conexión. Ahora, gracias a la existencia del USB, todos estos periféricos han unificado especificaciones de conexión físicos, permitiendo conectarse todos mediante este tipo de puerto.

Cada día se suman más aplicaciones a la lista de LTI, pero no hemos conseguido encontrar ningun con las mismas especificaciones que la nuestra. Existen sistemas parecidos de realización de examen, como DigiExam y otros de gestión de tareas para casa como CircleIn.

2.2 DigiExam

DigiExam [4] es una plataforma disponible en modo online y offline, que permite a los profesores una nueva manera de realizar exámenes.

Este sistema consiste en una plataforma única para profesores en la cual pueden crear exámenes, tanto de manera individual como de manera cooperativa con otros profesores y permite la realización de éstos por parte de los alumnos desde el mismo sistema.

Contiene un gran conjunto de baterías de preguntas y estilos para mostrar, permitiendo así a los profesores ahorrar hasta un 40% del tiempo invertido en la creación de exámenes.

Es una ventaja que tengan versión offline, ya que no siempre se tiene acceso a internet, y poder trabajar con unas prestaciones bastante parecidas a las que te ofrece la versión online, pero sin tener conexión, puede facilitar mucho el trabajo a los profesores.

A parte del sistema integrado de creación de exámenes, DigiExam permite a los profesores, monitorizar los ordenadores de los alumnos, detectando otros programas ejecutándose en paralelo al mismo tiempo que el alumno está realizando el examen.

Las diferencias con nuestro sistema, es que en DigiExam los profesores utilizan los ejercicios planteados para puntuar y evaluar a los alumnos, además, estos ejercicios son corregidos manualmente por el profesor después de realizar el ejercicio.

2.3 CircleIn

CircleIn [5] es una plataforma creada por y para alumnos con el objetivo de compartir tareas de clase y apuntes que puedan necesitar otros alumnos.

Basado principalmente en un sistema Canvas, permite a los alumnos comunicaciones peer to peer (P2P)[1] para el intercambio de material educativo, de esta manera se puede acceder a toda esa información creada y compartida por alumnos desde cualquier sitio con acceso a internet.

Para promover que los alumnos suban cosas y compartan su material educativo, la plataforma integra un sistema de puntuaciones en base a la cantidad de materias que se comparta. Dependiendo del material compartido y de la cantidad de material, se obtiene cierta cantidad de puntos que después el alumno puede canjear por premios muy bien adaptados a los tiempos en los que vivimos, como son entre otros, tarjetas de descuento de Amazon o tarjetas de regalo para Starbucks.

Además, la plataforma también premia la colaboración con la propia aplicación y da puntos a los alumnos que inviertan tiempo apoyando a la comunidad y moderando el contenido compartido.

La integración con LTI permite al alumno cargar la aplicación CicleIn desde el propio sistema de la Universidad solo pulsando en el enlace donde te redirecciona a la ventana del sistema.

[1] Método más común de compartir información y datos vía web sin importar la plataforma utilizada.

3. Entorno de desarrollo

En este capítulo se va a describir la tecnología que se ja usado para el desarrollo de este proyecto. Se explicará también como se pueden aplicaciones web con Java utilizando el lenguaje de programación de JSP en el entorno de compilación de NetBeans junto con un servicio de servidor de base de datos, en este caso un servicio Apache creado gracias a XAMPP. En la sección 3.1 vamos a hablar un poco del compilador, en la sección 3.2 explicaremos como funciona y como se estructura el servicio ofrecido por XAMPP y en la sección 3.3 hablaremos del lenguaje de programación JSP.

3.1 NetBeans 8.2

Netbeans [6] es un entorno de programación pensado para el lenguaje Java [7], es un entorno de libre uso, es decir, que no se necesita de un desembolso económico para la obtención de una licencia.

Esta plataforma permite trabajar con diferentes capas de programación, una ventaja respecto a otros entornos, como podrían ser eclipse [8] o BlueJ [9]. En mi caso lo he utilizado para el diseño de tres capas, visualización web (html), comunicación con la base de datos (java y mysql) y los llamados “servlets” utilizados en JSP para la comunicación de los dos anteriores.

El motivo de utilizar la versión 8.2 de este IDE, sin ser ésta la última versión del mismo es porque en las siguientes versiones a esta (de la 8.2 hasta la actual 11.0) NetBeans empieza una evolución al ser comprado o ayudado por Apache. Al desconocer lo que podía ocurrir y los cambios que podía conllevar, preferí trabajar con una versión más fiable, con la que ya había trabajado anteriormente, tanto durante la carrera como en mis proyectos individuales.

Otro de los motivos de la implementación con NetBeans es que trabaja con un lenguaje multiplataforma, esto significa que todo el código que esté escrito en este programa en el lenguaje Java, podrá ser manejado desde cualquier otra plataforma, dejando así la posibilidad de ampliación del proyecto a algún futuro estudiante que quiera seguir implementando “The Code Game”.

3.2 XAMPP

XAMPP [10] es un intérprete gratuito que permite la habilitación de un servicio Apache o de un servidor MySql [11] en tu propio equipo sin la necesidad de la configuración ni instalación requerida por los mismos.

Es utilizado principalmente como ayuda a en la ejecución y prueba de plataformas web o programas que necesiten conectarse a una base de datos o a un Servicio Apache, en tu propio campo, en un ámbito local, sin necesidad de servidores externos ni contratación de servicios en la web.

Los servicios de MySql son mediados y tratados por el usuario mediante un navegador web, accediendo a su dirección local http://127.0.0.1. Esta dirección permite acceder a PHPMyAdmin, utilizado para la gestión de las bases de datos MySql.

Además de estos servicios, XAMPP ofrece un entorno de interpretación de código en PHP y Perl, así como servidores FTP muy comunes utilizados por gran cantidad de usuarios como puede ser ProFTPD o FileZilla Server. Todas estas ventajas las puedes obtener solo con la descompresión y ejecución del instalador de XAMPP, ofrecido de forma gratuita en su propia página web.

3.3 JSP

Siglas de Java Server Pager [12] [11], es un lenguaje de programación simular a PHP que permite a los desarrolladores de software crear entornos y páginas web dinámicas mediante el uso de HTML y XML, entre otras posibles extensiones.

Se puede utilizar desde diferentes IDEs de programación, como puede ser Eclipse o NetBeans si se les dota de un servicio intérprete. Los servicios más habituales suelen ser servicios Apache, servidores MySql y, como en mi caso, XAMPP.

Este lenguaje de programación trabaja principalmente con tres capas de integración, la capa de librerias, en mi caso las que nos permiten la conexión a la base de datos, la capa de comunicación JSP llamada Servlets, y la propia visualización en el navegador, la cual utiliza HTML.

La principal ventaja de JSP frete a otros lenguajes como PHP, Python o Haskell es la versatilidad y la facilidad que ofrece a la hora de poder crear capas para adecuarse al proyecto. Según la lógica de negocio que maneje cada proyecto y según el acceso a los datos que se necesite, JSP permite todo tipo de flexibilidades para adaptarse al propio proyecto.

Otra de las principales ventajas de JSP respecto a otros tipos de programación web, es que permite la ejecución dinámica del código, es decir, no es necesario que cada vez que se compile y ejecute el código se cargue de nuevo toda la ejecución y se compruebe e inicialice todo, como ocurre por ejemplo en PHP, si no que la tecnología que utiliza JSP es la misma que se utiliza en los Servlets Java, es decir, la primera vez que se ejecuta la aplicación sí que carga todo desde cero, pero el resto de veces que se active o ejecute, la aplicación no necesita ser cargada, se mantiene en caché a la espera de ser llamada a la ejecución de nuevo, cosa muy beneficiosa que facilita sobre todo el manejo de sesiones y la gestión de datos y conexiones a la base de datos.

4. Análisis del sistema

En esta parte del proyecto hablaremos de las características y las necesidades del sistema para satisfacer a los usuarios en el estado final del proyecto. Esto será posible mediante la obtención de requisitos, tanto funcionales como no funcionales, así como la obtención y el análisis de los casos de uso y los diagramas de secuencia de cada una de las acciones posibles en el sistema.

Todo este apartado es la fase previa al diseño de la aplicación, que nos puede ayudar a el desarrollo y a la buena implementación de nuestra aplicación.

Para finalizar se muestran un conjunto de prototipos, diseños básicos de como pretendemos que la aplicación se vea finalmente. Todos estos mockups pueden variar durante el desarrollo del proyecto con la obtención de nuevos requisitos, nuevas funcionalidades, nuevos casos de uso y nuevas especificaciones.

4.1 Definición del Sistema.

Pretendemos realizar un sistema para la ayuda y motivación a los alumnos a realizar ejercicios de programación. Todo esto desde una interfaz sencilla e intuitiva tanto para los alumnos como para los profesores.

En este sistema tendremos principalmente dos tipos de rol de usuario, los alumnos y los profesores. Los alumnos registrados inicialmente pueden consultar cuales son los usuarios del sistema, así como los puntos obtenidos por cada uno de ellos en la resolución de ejercicios y luego podrán acceder a los ejercicios visibles en ese momento.

Cuando un alumno accede a un ejercicio, se le asigna una solución disponible de las que tiene registradas este ejercicio, de tal manera que, si hay 10 soluciones disponibles para 10 alumnos distintos, cada uno de ellos tendrá asignada una solución diferente.

Por otra parte, los profesores tienen el poder de administradores del sistema, son los encargados de los CRUD[2] tanto de los ejercicios como de los usuarios.

A su vez, también son los encargados de elegir los ejercicios que pueden ser resueltos por el alumno, es decir, los ejercicios que el alumno ve en su lista de ejercicios.

[2] CRUD: siglas de Create, Read, Update, Delete, funciones básicas en bases de datos.

4.2 Especificación de Requisitos

En un proyecto software existen dos tipos de requisitos, los requisitos funcionales y los requisitos no funcionales.

En los requisitos funcionales se especifica lo que se espera que realice el producto software que estamos desarrollando, es decir, especifican lo que nuestro software es capaz de hacer.

En cambio, los requisitos no funcionales son los requisitos que nuestro software debe cumplir más allá de la funcionalidad pura. Algunos ejemplos de estos requisitos pueden ser, por ejemplo, el uso intuitivo, la eficiencia, la robustez, etc.

4.2.1 Tabla de requisitos

En base a las características del sistema(equipo pequeño, tipo de cliente, etc) y del entorno, en esta primera dase se planificaron secuencias pasa la definición de los requisitos. El resultado lo mostramos a continuación.

La tabla de requisitos y el modelo de requisitos que vamos a utilizar es el siguiente:

Identificador:
Nombre:
Necesidad:
Prioridad:
Descripción:
Modificaciones:

Tabla 1. Tabla de Requisitos

- **Identificador:** es el nombre único que le vamos a dar a cada uno de los requisitos, diferenciándolos entre funcionales y no funcionales, la nomenclatura elegida para este proyecto es la siguiente:
  - **RF_ID:** Requisito funcional
  - **RNF_ID:** Requisito No Funcional
  - **ID:** Identificador único de cada requisito
- **Nombre:** Nombre dado a cada requisito.
- **Necesidad:** Especifica la importancia de la inclusión de este requisito en el sistema.

- **Prioridad:** Indica el grado de importancia de desarrollo del requisito a la hora de la implementación del mismo en este proyecto. Vamos a diferenciarlo en tres niveles de prioridad:
  - **Alto**
  - **Medio**
  - **Bajo**
- **Descripción:** Breve explicación de en qué consiste este requisito
- **Modificaciones:** Explicación de las modificaciones sufridas por el requisito durante la vida activa del desarrollo del proyecto. Este campo puede estar vacío si no han ocurrido modificaciones.

4.2.2 Requisitos Funcionales.

Identificador: RF_01
Nombre: Log-in en el sistema con diferentes roles de usuario
Necesidad: Esencial
Prioridad: Alta
Descripción: Al tener varios roles de usuario, el sistema debe diferenciar desde una misma ventana de log-in el menú de inicio y presentación al que tiene que mandar a cada usuario según el rol asignado.
Modificaciones:

Tabla 2. Requisito Funcional 01

Identificador: RF_02
Nombre: Registro inicial solo para Alumnos
Necesidad: Esencial
Prioridad: Alta
Descripción: Los alumnos deben ser capaces de registrarse por sí mismos sin necesidad de que el profesor tenga que dar de alta a cada uno de los alumnos.
Modificaciones: --

Tabla 3. Requisito Funcional 02

Identificador: RF_03
Nombre: Registro de un nuevo Ejercicio con sus respectivas soluciones
Necesidad: Esencial
Prioridad: Alta
Descripción: Solo el profesor puede dar de alta un ejercicio. Como es lógico, cada uno de los ejercicios tiene sus respectivas soluciones (input, output), siendo el número de éstas elección del propio profesor, pudiendo insertar todas las soluciones que necesite en un mismo ejercicio
Modificaciones:

Tabla 4. Requisito Funcional 03

Identificador: RF_04
Nombre: Asignación de soluciones a alumno
Necesidad: No es realmente esencial para el funcionamiento del software, pero sí que se necesita para evitar la asignación masiva de una solución a un número grande de alumnos y sobre todo para que todas las soluciones consultadas tengan algún alumno asignado.
Prioridad: Alta
Descripción: El sistema puede tener varios alumnos registrados que quieran resolver el mismo ejercicio, por lo que para que cada alumno tenga siempre los mismos parámetros de entrada en los mismos ejercicios, se crean las asignaciones. De esta forma cada vez que un alumno acceda a un ejercicio, será el mismo input.
Modificaciones: En el registro de las soluciones especificado en el RF_03, las asignaciones a usuarios de las soluciones están vacías.

Tabla 5. Requisito Funcional 04

Identificador: RF_05
Nombre: Un usuario no puede resolver un ejercicio más de una vez
Necesidad: Esencial
Prioridad: Alta
Descripción: Se quiere evitar que un alumno obtenga puntos por la resolución de un ejercicio en bucle, es decir, si resuelve un ejercicio, el alumno puede resolverlo las veces que quiera, pero solo obtiene un punto de bonificación por la primera resolución del ejercicio.
Modificaciones: Esto lo controlamos mediante la implantación del RF_06

Tabla 6. Requisito Funcional 05

Identificador: RF_06
Nombre: Marcar Asignación como resuelta
Necesidad: Importante
Prioridad: Alta
Descripción: Para evitar que un alumno incumpla el RF_05, se marcan como solucionadas las asignaciones de un alumno que obtengan una respuesta correcta.
Modificaciones: Se implementa para la mejora y correcta ejecución del RF_05

Tabla 7. Requisito Funcional 06

Identificador: RF_07
Nombre: Editar visibilidad de los ejercicios
Necesidad: Esencial
Prioridad: Media
Descripción: El profesor puede preparar ejercicios, pero tenerlos en segundo plano e ir lanzando cuando él prefiera nuevos ejercicios. Cuando se muestra un ejercicio se permite el acceso a este mismo a todos los alumnos registrados en el sistema, al esconderse, los alumnos que no lo hayan solucionado, no podrán hacerlo hasta que el profesor decida volver a mostrar dicho ejercicio.
Modificaciones:

Tabla 8. Requisito Funcional 07

Identificador: RF_08
Nombre: Editar ejercicio
Necesidad: Esencial
Prioridad: Alta
Descripción: El profesor puede tener algún tipo de error en el registro de un ejercicio, así que tiene que poder editar el mismo para poder modificar, tanto errores como el propio enunciado del ejercicio
Modificaciones:

Tabla 9. Requisito Funcional 08

Identificador: RF_09
Nombre: Añadir Soluciones a un ejercicio
Necesidad: Esencial
Prioridad: Alta
Descripción: El profesor puede añadir soluciones a un ejercicio existente sin la necesidad de volver a crear otro nuevo. Estos ejercicios añadidos son creados con asignaciones vacías a la espera de que cuando llegue un alumno que no tenga ninguna asignación de ese ejercicio, obtenga este nueva solución.
Modificaciones: Esto se realiza desde el mismo menú de edición de ejercicio especificado en el RF_08

Tabla 10. Requisito Funcional 09

Identificador: RF_10
Nombre: Editar Soluciones de ejercicio
Necesidad: Esencial
Prioridad: Alta
Descripción: El profesor puede modificar las soluciones que ha registrado en los ejercicios que haya dado de alta, siendo estas actualizadas tanto en las asignaciones a los alumnos que estén asignadas en ese momento como las que todavía estén sin asignar.
Modificaciones: Esto se realiza desde el mismo menú de edición de ejercicio especificado en el RF_08

Tabla 11. Requisito Funcional 10

Identificador: RF_11
Nombre: Eliminar Soluciones a un ejercicio
Necesidad: Esencial
Prioridad: Alta
Descripción: El profesor puede eliminar las soluciones que necesite de un ejercicio en concreto, sin necesidad de crear uno nuevo sin esta solución. Si esta solución está asignada a un alumno, se elimina esta asignación y cuando el alumno acceda de nuevo al ejercicio se le dará otra solución que esté disponible.
Modificaciones: Esto se realiza desde el mismo menú de edición de ejercicio especificado en el RF_08

Tabla 12. Requisito Funcional 11

Identificador: RF_12
Nombre: Actualizar lista de asignaciones
Necesidad: Esencial
Prioridad: Alta
Descripción: Al hacer una modificación en las soluciones de los ejercicios, es necesario actualizar la lista de asignaciones, y sea para añadir o para borrar asignaciones de soluciones a los alumnos. Se actualizará de la misma manera cuando un profesor elimine a un usuario registrado.
Modificaciones: Esto se modifica con las acciones realizadas en los RF_09 y RF_10

Tabla 13. Requisito Funcional 12

4.2.3 Requisitos No Funcionales

Identificador: RNF_01
Nombre: Aplicación Web
Necesidad: Esencial
Prioridad: Alta
Descripción: Debe ser una plataforma web, a la que los usuarios tengan acceso mediante cualquier navegador web.
Modificaciones:

Tabla 14. Requisito No Funcional 01

Identificador: RNF_02
Nombre: Interfaz Intuitiva y sencilla
Necesidad: Esencial
Prioridad: Media
Descripción: La interfaz de la aplicación tiene que ser fácil de usar e intuitiva para personas que utilicen normalmente el ordenador.
Modificaciones:

Tabla 15. Requisito No Funcional 02

Identificador: RNF_03
Nombre: Control de errores
Necesidad: Esencial
Prioridad: Alta
Descripción: La aplicación comprobará que los usuarios no cometan ningún error a la hora hacer log-in, o la hora de actualizar los ejercicios y soluciones.
Modificaciones:

Tabla 16. Requisito No funcional 03

Identificador: RNF_04
Nombre: Encriptación de contraseñas
Necesidad: Esencial
Prioridad: Alta
Descripción: Cuando el usuario se registre, es necesario almacenar la contraseña de log-in en la BD, pero debemos tener en cuenta que no debe guardarse en lenguaje natural, debe ser guardada en MD5 con algún SALT de control de contraseñas.
Modificaciones:

Tabla 17. Requisito No Funcional 04

Identificador: RNF_05
Nombre: Compatibilidad con los principales navegadores
Necesidad: Esencial
Prioridad: Alta
Descripción: La aplicación debe ser compatible con los navegadores más famosos disponibles en este momento, como son Firefox y Chrome.
Modificaciones:

Tabla 18. Requisito No Funcional 05

Identificador: RNF_06
Nombre: La aplicación deberá ser Compatible con el Estándar Sakai
Necesidad: No Esencial
Prioridad: Media-Baja
Descripción: La aplicación deberá cumplir con el estándar de comunicación ofrecido por Sakai para poder ser integrado como un complemento en el PoliformaT, así permitir un acceso más fácil a los alumnos.
Modificaciones:

Tabla 19. Requisito No Funcional 06

4.3 Diagrama de Clases.

JSP es un lenguaje basado principalmente en JAVA, por lo que como este, JSP también es Orientado a Objetos y en el modelado de una aplicación con un lenguaje de este tipo no puede faltar el diagrama de clases.

Un diagrama de clases describe el diseño del sistema gráficamente mostrando, en lenguaje de modelado UML, la relación entre clases, y objetos que tiene.

4.3.1 UML

El lenguaje utilizado para el diseño del diagrama de clases se llama UML [13], siglas de "Unified Modeling Language". UML es una notación muy utilizada en ingeniería del software para la realización de todos estos tipos de diagramas y representaciones gráficas necesarias para la explicación del funcionamiento básico de un Sistema Software.

que esto es un objeto coche, ya que instancia y poner valores a los parámetros de un elemento de la clase coche.

4.3.2 Elementos de un Diagrama de Clases

Dentro de un diagrama de clases, podemos encontrar diferentes tipos de elementos:

- **Clases:** Elemento que representa un grupo de Objetos con una estructura y un funcionamiento similar.

Ilustración 1. Ejemplo de Clase

Para los elementos de las clases existe una nomenclatura especial para distinguir y aplicar reglas de visibilidad:

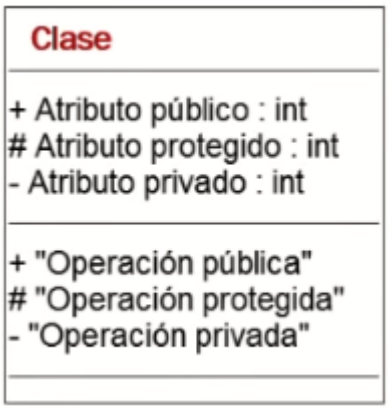

Ilustración 2. Visibilidad de Elementos en Clases

- **Relaciones:** elemento que une y relaciona dos clases, hay varios tipos de relaciones, los mas habituales y los utilizado por mí en este proyecto son:
    - **Asociación:** se representa con una línea continua entre dos clases, representa una unión bidireccional entre las clases, mostrando también la cardinalidad de la relación. Las cardinalidades más comunes son:
        - **1:** representa que para esa relación solo existe un elemento de ese tipo.
        - **0.. *:** representa que en esa relación puede no existir, o existir muchos elementos de ese tipo.
        - ***:** representa que al menos debe existir un elemento de este tipo en dicha relación

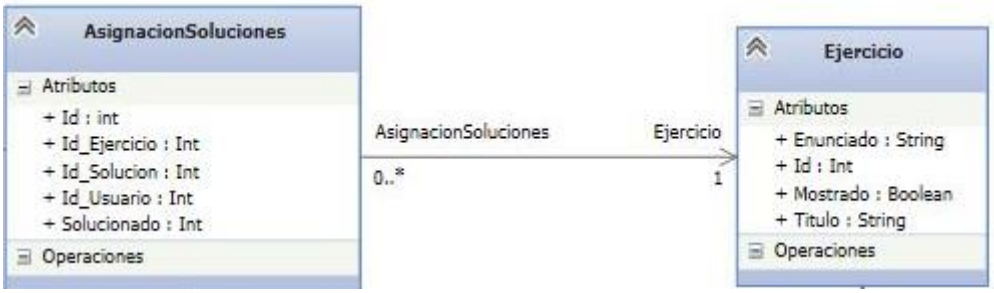

Ilustración 3. Ejemplo de Asignación

En la Ilustración 3 podemos ver un ejemplo de asociación 1 a muchos, explicando este ejemplo podríamos decir que un ejercicio puede tener ninguna o muchas asignaciones, pero una asignación siempre tiene que tener un Ejercicio.

- **Agregación:** en este caso la relación se representa con una línea recta, pero a un extremo podemos ver un rombo vacío. Esta relación puede leerse como: Un elemento contiene a otro elemento.

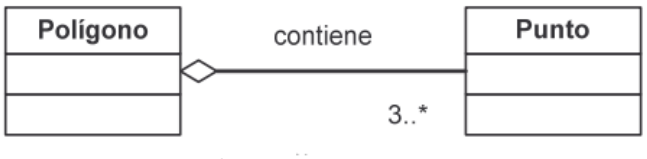

Ilustración 4. Ejemplo de Agregación

En la Ilustración 4 podemos ver un ejemplo de agregación, viendo esta relación podríamos decir que un Polígono contiene al menos 3 puntos.

- **Composición:** Se representa como una recta que une dos clases, pero tiene a un extremo un rombo pintado de negro. Podríamos leer una relación de este tipo como: Una clase está compuesta por otra clase.

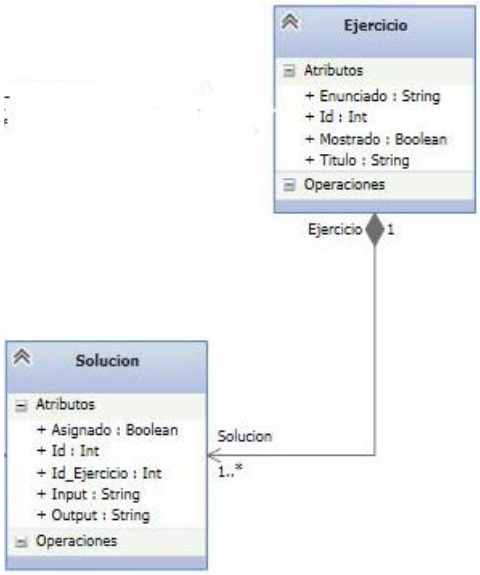

Ilustración 5. Ejemplo de Composición

En la Ilustración 5 podemos ver un ejemplo de composición. Podríamos leer esta relación como un ejercicio está compuesto por varias soluciones, es decir, que junto a la existencia de un ejercicio, hay al menos una Solución que lo compone.

- **Herencia:** Se representa con una flecha con la punta vacía en un extremo. Se utiliza para representar una clase como padre de otras, las cuales comparten atributos, pero queremos diferenciarlos en clases diferentes.

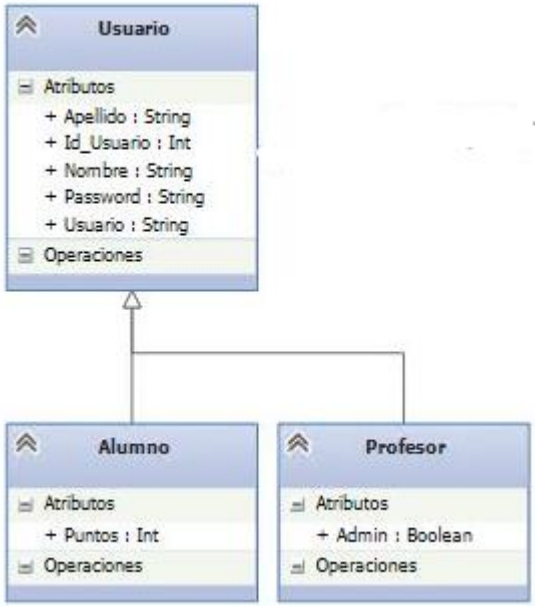

Ilustración 6. Ejemplo de Herencia

Como podemos ver en la Ilustración 6, tenemos un ejemplo de herencia, en el cual tenemos la clase usuario con sus atributos y unidas con una relación de herencia a las clases Alumno y Profesor. Esto lo podemos leer como que la clase Alumno, tiene los mismos atributos que la clase Usuario mas a parte los suyos propios, es decir, los atributos de un alumno son los del usuario más un indicador de puntos. Mientras que los atributos del Profesor son los mismos que los de Usuario más a parte un boolean llamado Admin.

4.3.3 Diagrama de Clases General

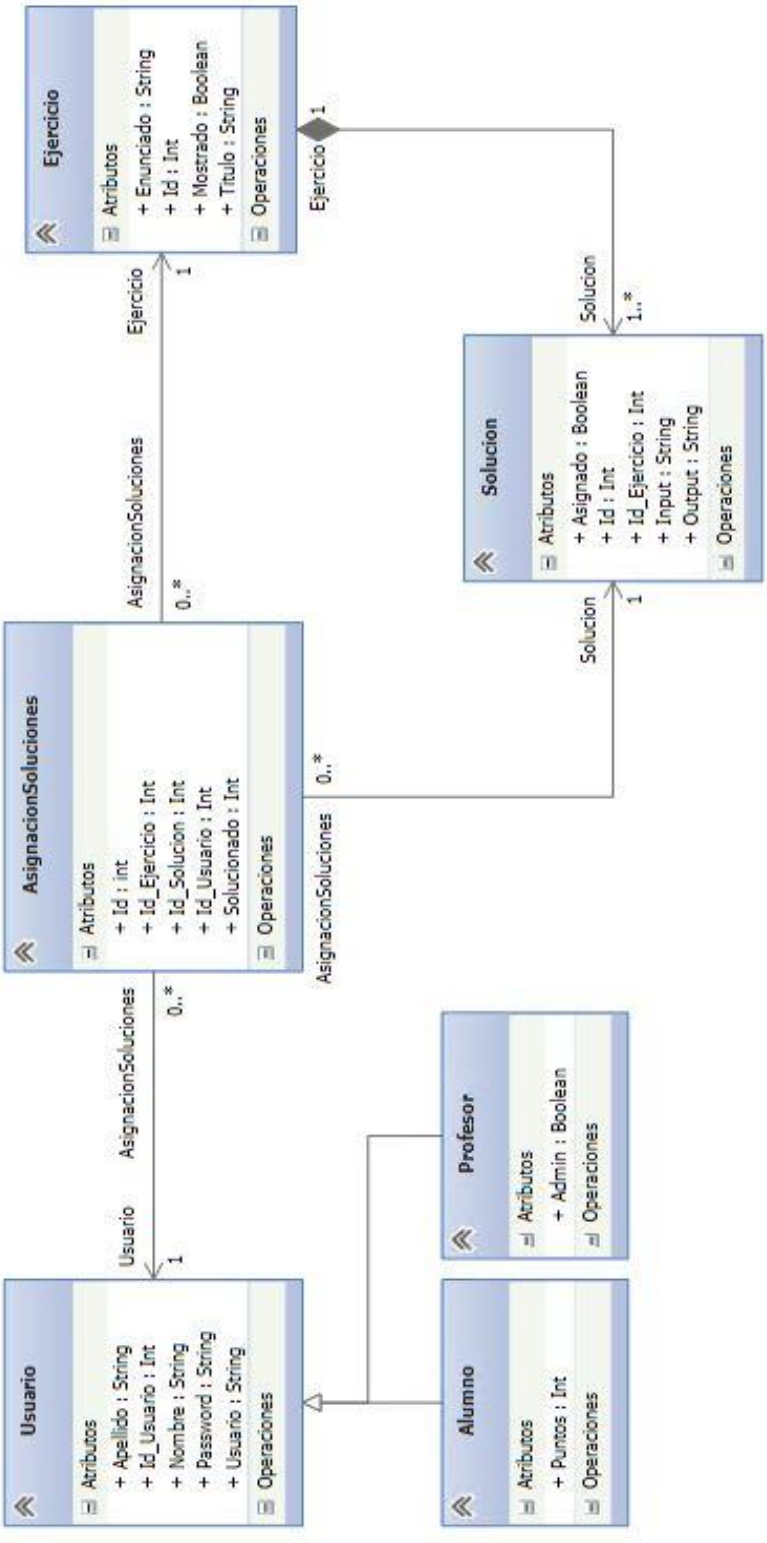

Ilustración 7. Diagrama de Clases "The Code Game"

Como podemos ver en la Ilustración 7, nuestra aplicación está compuesta por 6 clases distintas, cada una con sus propios atributos y relacionadas entre ellas.

Cada una de estas clases contienen elementos propios de cada uno de los elementos, tanto elementos utilizados para la identificación de cada Objeto que se cree, como son los ID de cada una de las tablas, así como de variables de control para el manejo de los datos dentro de la aplicación, como puede ser el elemento Mostrado dentro de la clase Ejercicio.

Como podemos ver tenemos dos tipos de usuarios, Alumno y Profesor, los cuales se diferencian porque el usuario Alumno tiene un parámetro de tipo Int, Puntos, que se utiliza para almacenar los puntos que tiene cada uno de los alumnos. Por otra parte, el usuario Profesor, tiene un elemento de tipo Boolean, Admin, que, en el caso de ser True, el usuario sería un Profesor y podría acceder al sistema con privilegios de dicho rol.

Por otra parte, podemos ver que la clase ejercicios tiene una composición con la clase Solución, lo que quiere decir que un ejercicio este compuesto por, una o muchas soluciones.

También podemos ver que la clase nexo entre las clases Usuario, Ejercicio y Solución es la clase AsignaciónSoluciones, la cual es utilizada para identificar, mediante los ID de cada uno de los elementos, la solución del ejercicio que ha sido asignada a un alumno.

Esta es la tabla que utilizamos para que el alumno siempre vea la misma solución dentro del mismo ejercicio.

4.4 Diagrama de casos de uso

El diagrama de casos de uso especifica las acciones que puede hacer cada uno de las personas o sistemas que interactúan con el sistema de nuestra aplicación.

Está diseñado en UML el mismo lenguaje que esta expresado anteriormente en el Diagrama de Clases.

4.4.1 Elementos del diagrama de casos de uso

Todos los diagramas que se realizan con UML son diagramas visuales, gráficos. Los elementos que podemos ver en este diagrama son:

- **Actor:** Persona o sistema capaz de realizar modificaciones directas a nuestro Sistema Software interactuando con los objetos y con la base de datos para conseguir un objetivo concreto.

  Podemos remarcar que NO es necesario que este actor sea una persona física, podemos considerar como un actor a un subsistema u otro sistema que interactúe con el nuestro.

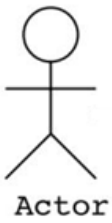

Ilustración 8. Actor Diagrama de Casos de Uso

- **Caso de uso:** es la operación realizada por el actor. Podríamos decir que es cada una de las acciones de modificación del sistema que realiza cada uno de ellos. Se representa con un óvalo, y esta unido al actor mediante una relación.

Ilustración 9. Caso de uso

- **Relaciones:** la relación es la unión entre un actor y su caso de uso, es unidireccional y puede ser de diferentes tipos:

    - **Relación de asociación:** es la relación básica y simple entre un caso de uso y su respectivo actor. Representa la invocación de dicho caso de uso por parte del actor. Se representa en el diagrama por una línea entre los dos elementos.

Ilustración 10. Relación de asociación

    - **Relación de inclusión:** relaciona dos casos de uso, indicando que el caso de uso origen no obtendrá su correcto resultado o no desarrollará su correcto funcionamiento sin la ejecución del caso de uso de destino. Se representa con una flecha con la línea discontinua, la cual incorpora una etiqueta “include”.

Ilustración 11. Relación de inclusión

- **Relación de extensión:** relaciona dos casos de uso, indicando que el caso de uso de origen no podría darse si no existe el caso de uso de destino, pero si al revés.

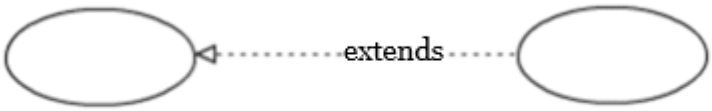

Ilustración 12. Relación de extensión

4.4.2 Diagramas de casos de uso del proyecto

La plantilla que vamos a utilizar para presentar los casos de uso de este proyecto es la siguiente:

Identificador	
Nombre	
Objetivo	
Actor	
Precondiciones	
Postcondiciones	
Escenario	

Tabla 20. Plantilla Casos de Uso

- **Identificador:**
  - CUA _ID. Siendo CU la etiqueta de Caso de Uso y ID el identificador propio y único de cada uno de ellos.
  - CUPR _ID. Siendo CU la etiqueta de Caso de Uso y ID el identificador propio y único de cada uno de ellos.
- **Nombre:** nombre del Caso de Uso.
- **Objetivo:** objetivo esperado con la aplicación de este Caso de Uso.
- **Actor:** actor que interacciona con el sistema para la realización de este Caso de Uso.

- **Precondiciones:** condiciones que deben cumplirse de forma previa a la realización de este Caso de Uso.
- **Postcondiciones:** estado del sistema después de la realización de este Caso de Uso.
- **Escenario:** descripción de los pasos a realizar para llevar a cabo la buena ejecución de este Caso de Uso.

4.4.2.1 Diagrama de Casos de Uso del alumno

Este es el diagrama de Casos de Uso del alumno en nuestro sistema:

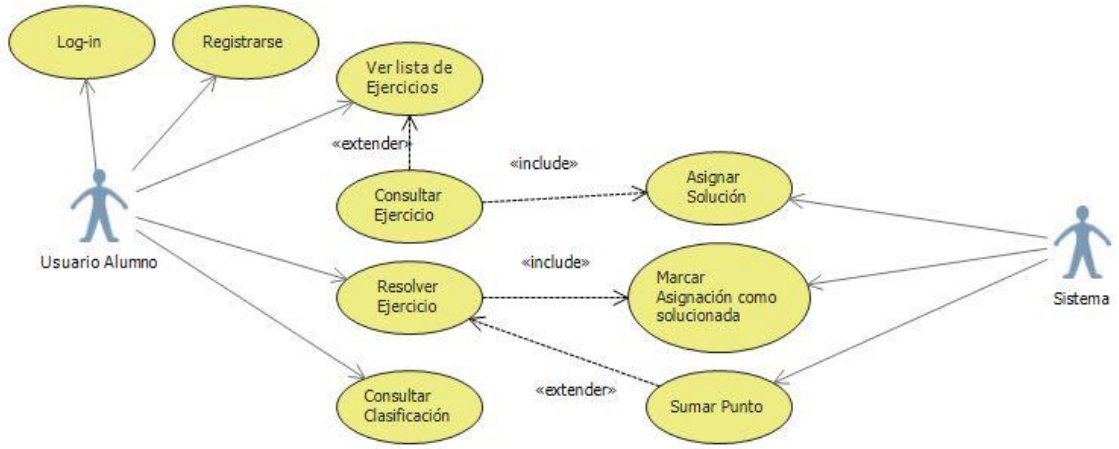

Ilustración 13. Diagrama de Caso de Uso del alumno

En la Ilustración 13 podemos ver todas las acciones disponibles que tendrá el alumno en el momento que tenga acceso a nuestro sistema. Como podemos observar, no solo necesita de la interacción de él mismo, sino que, en algunos casos, necesita de la interacción del propio sistema como un actor de él mismo.

Identificador	**CUA_02**
Nombre	Registro de usuario
Objetivo	Poder acceder al sistema
Actor	Usuario Alumno
Precondiciones	No estar registrado en el sistema
Postcondiciones	--
Escenario	1. Acceder a la aplicación 2. Pulsar en el enlace de registro 3. Introducir los datos solicitados 4. Pulsar botón Registrar

Tabla 21. Caso de Uso Alumno 01

Identificador	CUA_02
Nombre	Log-in
Objetivo	Acceder al sistema
Actor	Usuario Alumno
Precondiciones	Estar registrado en el sistema
Postcondiciones	--
Escenario	1. Introducir usuario 2. Introducir contraseña 3. Pulsar botón acceder

Tabla 22. Caso de Uso Alumno 02

Identificador	CUA_03
Nombre	Ver lista de ejercicios
Objetivo	Obtener la lista de ejercicios mostrados
Actor	Usuario Alumno
Precondiciones	Poder hacer log in en el sistema
Postcondiciones	--
Escenario	1. Acceder al sistema 2. Pulsar botón mostrar ejercicios

Tabla 23. Caso de Uso Alumno 03

Identificador	CUA_04
Nombre	Consultar ejercicio
Objetivo	Poder obtener el contenido de un ejercicio para poder resolverlo
Actor	Usuario alumno
Precondiciones	Poder hacer log in en el sistema Poder ver la lista de ejercicios
Postcondiciones	--
Escenario	1. Acceder al sistema 2. Pulsar botón mostrar ejercicios 3. Pulsar botón ir que aparece junto a cada ejercicio

Tabla 24. Caso de Uso Alumno 04

Identificador	CUA_05
Nombre	Asignar solución
Objetivo	Que al alumno se le asigne una solución disponible del ejercicio que consulta siendo ésta la misma cada vez que entre a consultar dicho ejercicio
Actor	Sistema
Precondiciones	Que el usuario alumno acceda a un ejercicio y no tenga ninguna solución asignada
Postcondiciones	--
Escenario	1. El alumno accede al ejercicio 2. Consulta si existe asignación 3. Si no existe asignación, consulta primera solución disponible 4. Se le asigna al alumno dicha solución

Tabla 25. Caso de Uso Alumno 05

Identificador	CUA_06
Nombre	Resolver ejercicio
Objetivo	Que el alumno proponga una solución al ejercicio
Actor	Usuario Alumno
Precondiciones	Acceder a un ejercicio
Postcondiciones	En caso de que la respuesta sea correcta, el sistema deberá cumplir con los CU_06 y CU_07
Escenario	1. Acceder al sistema 2. Acceder a la lista de ejercicios 3. Acceder a un ejercicio 4. Proponer una solución en el textbox habilitado para ello.

Tabla 26. Caso de Uso Alumno 06

Identificador	CUA_07
Nombre	Marcar asignación como solucionada
Objetivo	Marcar como solucionada la asignación de la solución al alumno para evitar que este pueda obtener un punto cada vez que resuelva este mismo ejercicio, es decir, que solo pueda obtener puntuación la primera vez que resuelva el ejercicio correctamente.
Actor	Sistema
Precondiciones	Que el alumno resuelva correctamente el ejercicio, CU_05
Postcondiciones	Poner un valor diferente de 0 en el parámetro “Solucionado” de la tabla “AsignacionSoluciones” de dicha asignación
Escenario	1. El alumno accede a un ejercicio 2. El alumno resuelve este ejercicio 3. Comprobación de la correcta solución del ejercicio

Tabla 27. Caso de Uso Alumno 07

Identificador	CUA_08
Nombre	Sumar punto
Objetivo	Que el alumno proponga una solución al ejercicio y ésta sea correcta
Actor	Usuario Alumno// Sistema
Precondiciones	Acceder a un ejercicio y proponer la solución correcta
Postcondiciones	El parámetro "Puntos" de la clase "Alumno" será incrementado en 1
Escenario	1. Acceder al sistema 2. Acceder a la lista de ejercicios 3. Acceder a un ejercicio 4. Proponer una solución en el textbox habilitado para ello. 5. Comprobación de la correcta solución del ejercicio

Tabla 28. Caso de Uso Alumno 08

Identificador	CUA_09
Nombre	Consultar clasificación
Objetivo	Que el alumno pueda consultar en todo momento el estado de la clasificación global del sistema
Actor	Usuario Alumno
Precondiciones	Acceder al sistema
Postcondiciones	--
Escenario	1. Acceder al sistema 2. Pulsar en el botón Clasificación

Tabla 29. Caso de Uso Alumno 09

4.4.2.2 Diagrama de Caso de Uso del Profesor

Este es el diagrama de Casos de Uso del profesor en nuestro sistema:

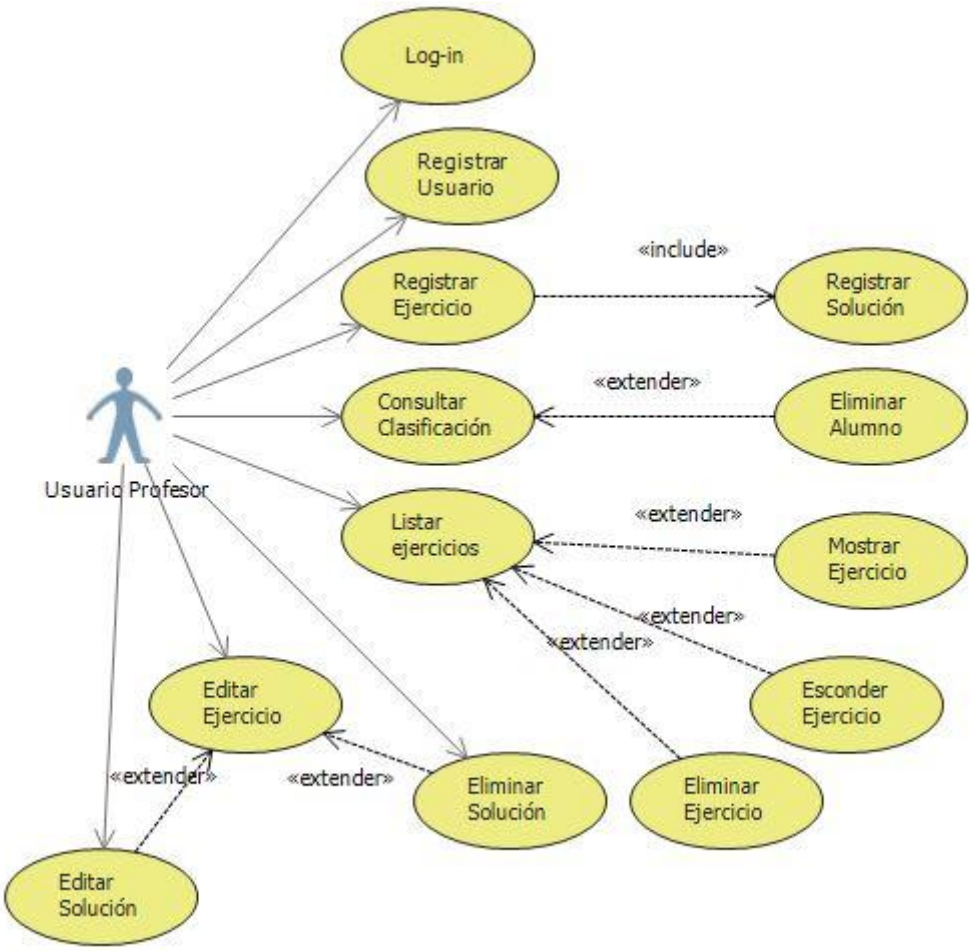

Ilustración 14. Diagrama de Casos de Uso de Profesor

En la Ilustración 14 podemos ver todas las acciones disponibles por un Usuario Profesor. Como podemos observar a diferencia del Usuario Alumno, el Actor Sistema no aparece en este diagrama, ya que los parámetros que se tienen que modificar que anteriormente los modificaba el Actor Sistema, ahora son modificados por el Profesor.

Identificador	**CUPR_01**
Nombre	Registrar Usuario
Objetivo	Registrar un usuario en el sistema
Actor	Usuario Profesor
Precondiciones	Estar registrado en el sistema y ser un Usuario Profesor, ya que solo los Usuarios Profesores pueden dar de alta a otros Usuario Profesores o pueden otorgar permisos de Profesor a un Usuario Alumno.

Postcondiciones	--
Escenario	1. Acceder al sistema 2. Pulsar en registrar Usuario 3. Introducir los datos solicitados 4. Si se desea que el usuario sea Profesor, marcar la casilla Administrador

Tabla 30. Caso de Uso Profesor 01

Identificador	**CUPR_02**
Nombre	Log-in
Objetivo	Acceder al sistema
Actor	Usuario Profesor
Precondiciones	Estar registrado en el sistema
Postcondiciones	--
Escenario	1. Introducir usuario 2. Introducir contraseña 3. Pulsar botón acceder

Tabla 31. Caso de Uso Profesor 02

Identificador	**CUPR_03**
Nombre	Registrar ejercicio// Registrar solución
Objetivo	Registrar un ejercicio en el sistema con sus respectivas soluciones
Actor	Usuario Profesor
Precondiciones	Estar registrado en el sistema y ser Usuario Profesor
Postcondiciones	La creación de un ejercicio con sus respectivas soluciones las cuales, desde un primer momento no tienen ningún tipo de asignación
Escenario	1. Acceder al sistema 2. Pulsar botón Registrar Ejercicio 3. Introducir Título y Enunciado 4. Pulsar el botón “+” tantas veces como soluciones se quieran introducir. 5. Introducir los inputs con sus respectivos outputs en los campos habilitados para ello. 6. Pulsar el botón Registrar

Tabla 32. Caso de Uso Profesor 03

Identificador	CUPR_04
Nombre	Consultar clasificación
Objetivo	Consultar el estado de la Clasificación global en ese momento
Actor	Usuario Profesor
Precondiciones	Estar registrado en el sistema
Postcondiciones	--
Escenario	1. Acceder al sistema 2. Pulsar el botón Clasificación

Tabla 33. Caso de Uso Profesor 04

Identificador	CUPR_05
Nombre	Eliminar Alumno
Objetivo	Eliminar el Usuario Alumno del sistema
Actor	Usuario Profesor
Precondiciones	Estar registrado en el sistema y ser Usuario Profesor
Postcondiciones	El Alumno debe ser eliminado del sistema, así como, las asignaciones activas que tenga en ese momento con los ejercicios a los que haya accedido.
Escenario	1. Realizar los pasos del CUPR_04 2. Seleccionar los Alumnos que se deseen eliminar 3. Pulsar el botón Eliminar

Tabla 34. Caso de Uso Profesor 05

Identificador	CUPR_06
Nombre	Listar Ejercicios
Objetivo	Ver todos los ejercicios que se han añadido al sistema
Actor	Usuario Profesor
Precondiciones	Estar registrado en el sistema y ser Usuario Profesor
Postcondiciones	--
Escenario	1. Acceder al sistema 2. Pulsar el botón Listar Ejercicios

Tabla 35. Caso de Uso Profesor 06

Identificador	CUPR_07
Nombre	Mostrar ejercicio
Objetivo	Permitir a los alumnos que puedan ver el ejercicio seleccionado
Actor	Usuario Profesor
Precondiciones	Estar registrado en el sistema y ser Usuario Profesor
Postcondiciones	El alumno puede ver y resolver el ejercicio que ha seleccionado el Profesor para ser visto.
Escenario	1. Realizar los pasos del CUPR _06 2. Marcar casilla mostrar en los ejercicios deseados 3. Pulsar botón Actualizar

Tabla 36. Caso de Uso Profesor 07

Identificador	CUPR_08
Nombre	Esconder ejercicio
Objetivo	Ocultar al alumno los ejercicios seleccionados
Actor	Usuario Profesor
Precondiciones	Estar registrado en el sistema y ser Usuario Profesor
Postcondiciones	Los ejercicios seleccionados ya no pueden ser solucionados ni vistos por el Usuarios Alumnos
Escenario	1. Realizar los pasos del CUPR _06 2. Marcar casilla Esconder en los ejercicios deseados. 3. Pulsar botón Actualizar

Tabla 37. Caso de Uso Profesor 08

Identificador	CUPR_09
Nombre	Eliminar Ejercicio
Objetivo	Eliminar un ejercicio del sistema
Actor	Usuario Profesor
Precondiciones	Estar registrado en el sistema y ser Usuario Profesor Que el ejercicio esté escondido
Postcondiciones	El Ejercicio debe ser eliminado del sistema, así como, las asignaciones activas que tenga en ese momento con los Alumnos que hayan accedido a él, así como eliminará las soluciones que dicho ejercicio tenga asignadas
Escenario	1. Realizar los pasos del CUPR_08. 2. Seleccionar el ejercicio que se desea eliminar 3. Pulsar el botón Eliminar

Tabla 38. Caso de Uso Profesor 09

Identificador	CUPR_10
Nombre	Editar Ejercicio
Objetivo	Editar un ejercicio del sistema
Actor	Usuario Profesor
Precondiciones	Estar registrado en el sistema y ser Usuario Profesor
Postcondiciones	--
Escenario	1. Acceder al sistema 2. Listar ejercicios 3. Pulsar botón Editar del ejercicio deseado 4. Modificar los parámetros deseados 5. Pulsar botón Actualizar

Tabla 39. Caso de Uso Profesor 10

Identificador	CUPR_11
Nombre	Eliminar solución
Objetivo	Eliminar solución de un ejercicio del sistema
Actor	Usuario Profesor
Precondiciones	Estar registrado en el sistema y ser Usuario Profesor
Postcondiciones	Si esta solución está asignada a algún Usuario, que éste deje de tener una solución asignada.
Escenario	1. Realizar los pasos del CUPR_10. 2. Seleccionar la solución que se desea eliminar 3. Pulsar el botón Actualizar

Tabla 40. Caso de Uso Profesor 11

Identificador	CUPR_12
Nombre	Editar solución
Objetivo	Editar una solución propuesta para un ejercicio
Actor	Usuario Profesor
Precondiciones	Estar registrado en el sistema y ser Usuario Profesor
Postcondiciones	--
Escenario	1. Realizar los pasos del CUPR_10. 2. Editar los parámetros de la solución que se deseen 3. Pulsar el botón Actualizar

Tabla 41. Caso de Uso Profesor 12

4.5 Diagramas de Secuencia

Antes hemos visto los diagramas de las interacciones entre las propias clases y entre los actores y el propio sistema, ahora, en el Diagrama de Secuencia, vamos a representar los movimientos internos de los datos y las peticiones de la propia aplicación para obtener y mejorar los datos y las acciones explicadas anteriormente.

El diagrama de Secuencia, como el resto de los diagramas con los que estamos trabajando, está representado en el lenguaje de modelado UML, que como ya hemos podido comprobar es el lenguaje por excelencia de diseño y representación de los diagramas necesarios en el proceso de Ingeniería del Software.

Este tipo de diagrama va directamente ligado con los Casos de Uso, ya que en el diagrama de secuencia representa gráficamente el camino seguido por un caso de uso para obtener su resultado óptimo.

4.5.1 Elementos del Diagrama de Secuencia

- **Línea de vida:** representa cada uno de los objetos principales que intervienen en el sistema. Se representa con un cuadrado que contiene el nombre del objeto del que sale una línea vertical discontinua hacia abajo.

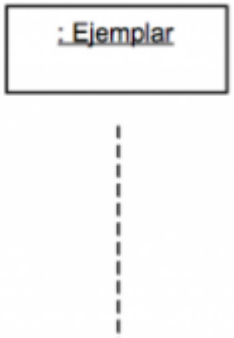

Ilustración 15. Ejemplo Linea de Vida

- **Activación:** es un rectángulo que se sitúa a lo largo de la línea de vida, que representa actividad en ese objeto durante un cierto periodo de tiempo, que puede ser hasta que mande la respuesta a otro objeto o hasta que se cierre la conexión.

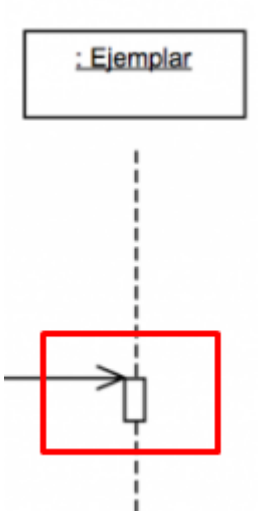

Ilustración 16. Ejemplo de Activación

- **Mensaje:** se representa con una flecha horizontal que va de una activación de una línea de vida a otra activación de otra línea de vida, e indica que existe una petición entre los objetos que representan estas dos.

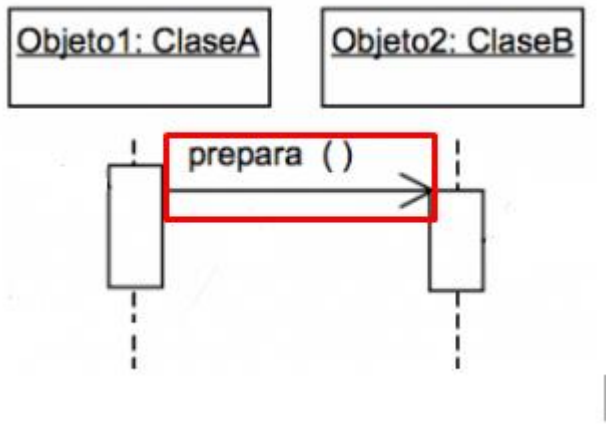

Ilustración 17. Ejemplo Mensaje

3.5.2 Diagramas de secuencia del sistema - Usuarios

A continuación, mostraremos los diferentes diagramas de secuencia junto a una breve explicación de su funcionamiento

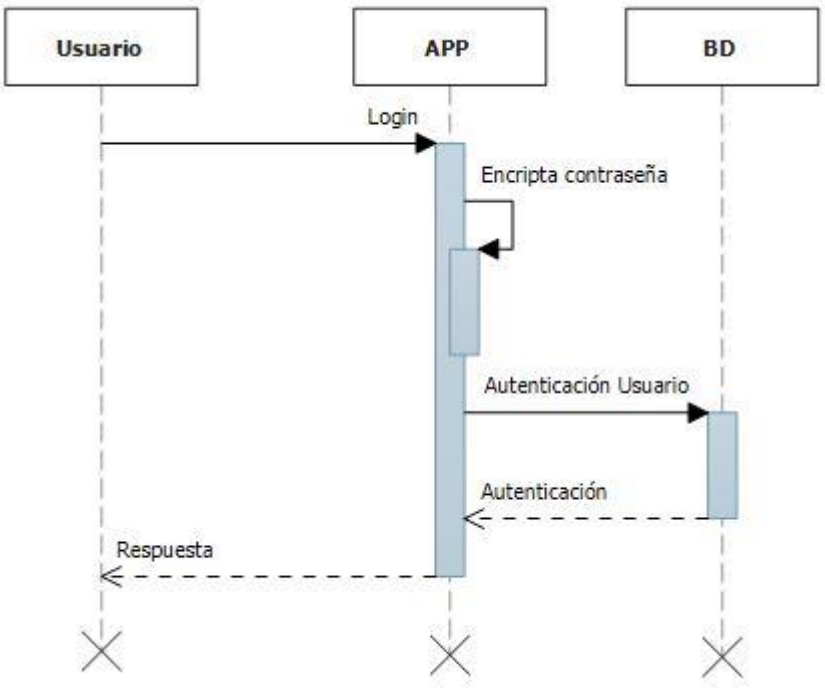

Ilustración 18. Diagrama de secuencia Usuario - Log in

Tal y como muestra la Ilustración 18, tenemos tres líneas de vida, Usuario, Aplicación y Base de Datos. Para el proceso de Log in al sistema, el Usuario introduce en la Aplicación los datos de registro. Ésta encripta la contraseña y le manda los datos a la Base de Datos para que los coteje con los existentes. La Base de Datos responde con la autenticación a la Aplicación la cual manda la respuesta al Usuario. Si es positiva, muestra el menú apropiado para el Usuario y si es negativa vuelve a solicitar los datos de acceso.

Este es el proceso que se realizará en los CUA_01 y CUPR_01.

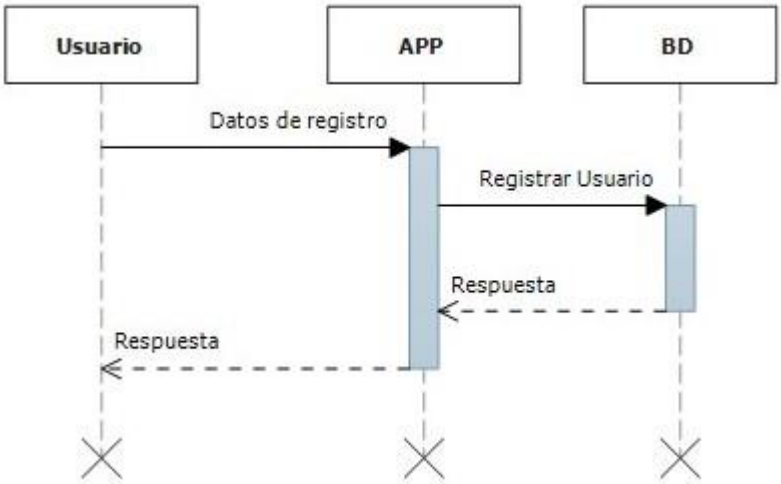

Ilustración 19. Diagrama de Secuencia Usuario - Registro

En el proceso de registro, el usuario manda a la Aplicación los datos de registro. La aplicación hace la petición de registro a la Base de Datos, ésta responde a la Aplicación y de la misma manera, ésta última responde al Usuario.

Este proceso se realizará en los CUA_02 y CUPR_02.

4.5.3 Diagramas de secuencia del sistema - Alumno

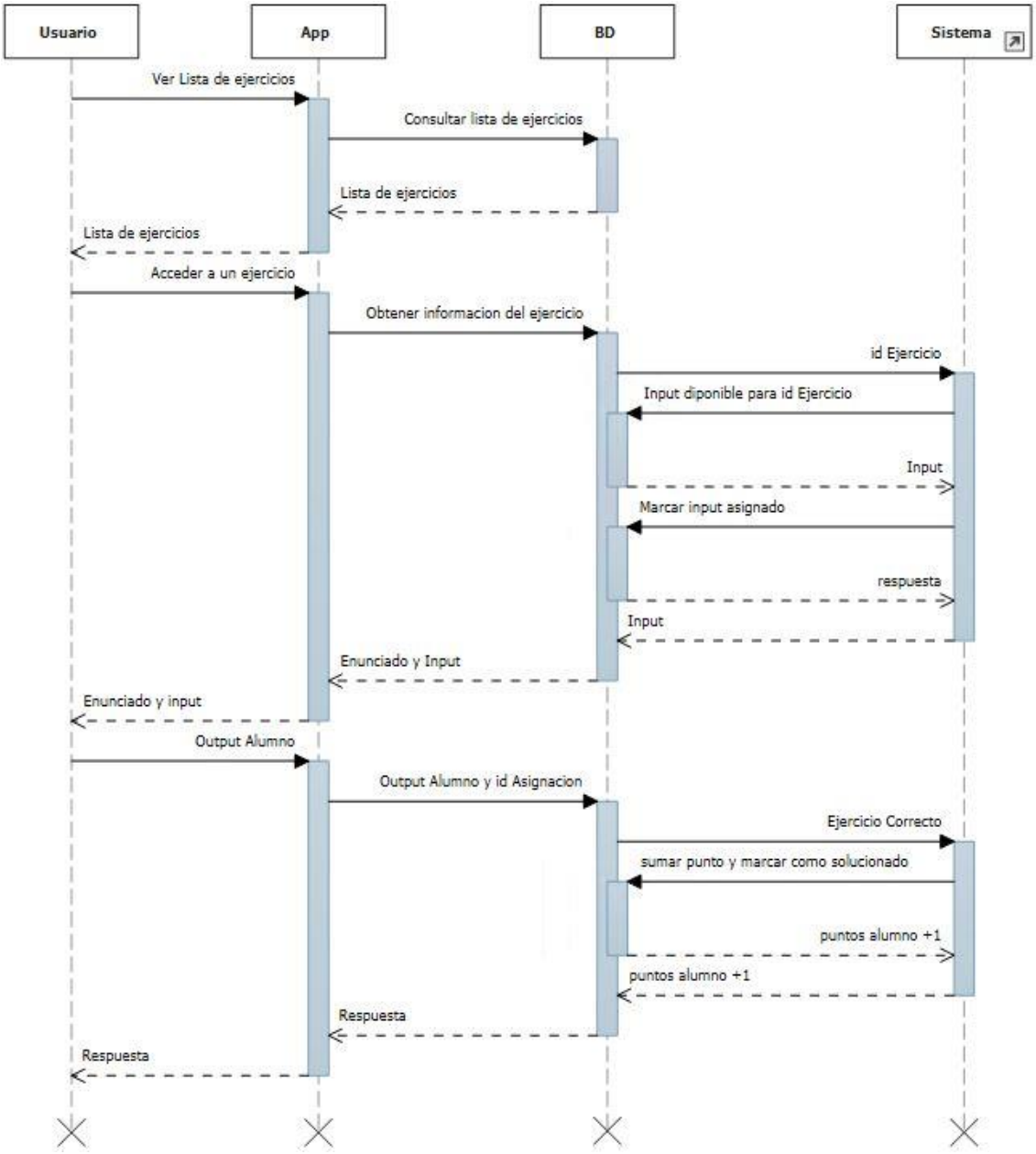

Ilustración 20. Diagrama de Secuencia Alumno - Gestión Ejercicio

Podemos dividir este diagrama en diferentes partes, es decir, varios Casos de Uso:

- Ver la lista de ejercicios CUA_03
- Consultar lista ejercicio CUA_04
- Asignar solución CUA_05
- Resolver ejercicio CUA_06
- Marcar asignación como solucionada CUA_07
- Sumar Punto CUA_08

En comparativa con los dos anteriores, podemos apreciar la aparición de una nueva línea de vida, el Sistema aparece como objeto de él mismo. La comunicación entre Usuario, Aplicación y Base de Datos es la misma que en los dos diagramas anteriores, pero se necesita la presencia del propio Sistema para la obtención y modificación de ciertos parámetros, como pueden ser la resolución de ciertas soluciones, la suma de puntos, y la obtención de asignaciones de ejercicios.

En este diagrama queda demostrado que las líneas de vida pueden trabajar de forma concurrente con más de una activación de forma simultánea. Un ejemplo lo tenemos en la solicitud del input disponible para un ejercicio, que se ejecuta a la misma vez que la activación de la consulta de un ejercicio realizada por el propio Alumno.

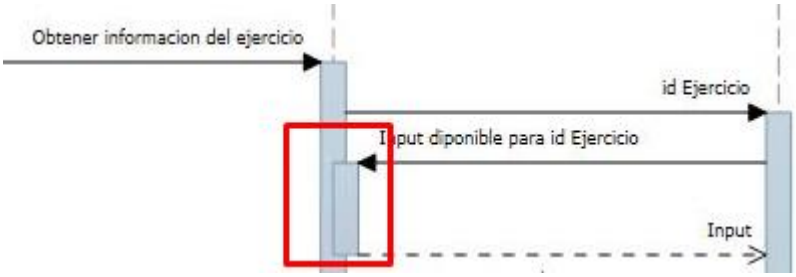

Ilustración 21. Ejemplo de concurrencia Diagrama de Secuencia

Para finalizar con los Diagramas de secuencia del Alumno, en la Ilustración 22 mostramos el diseño de la ejecución del CUA_09 correspondiente a consultar clasificación.

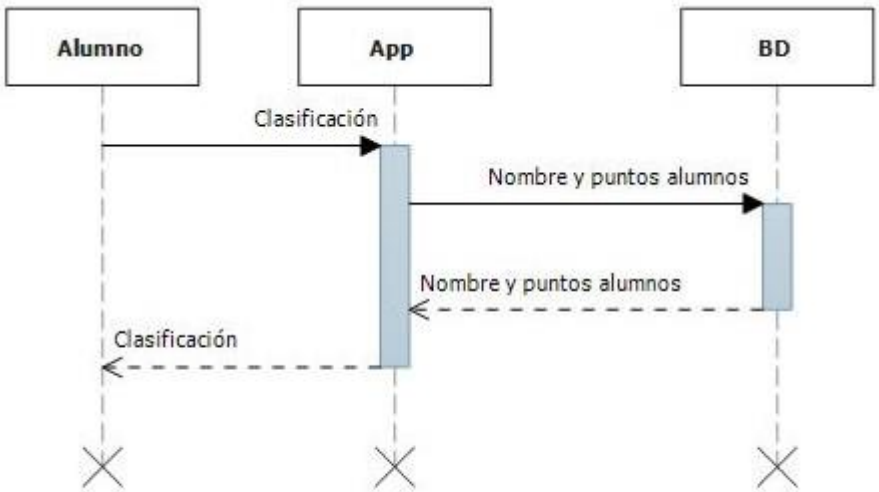

Ilustración 22. Diagrama de Secuencia Alumno – Clasificación

4.5.4 Diagramas de secuencia del sistema - Profesor

A partir del siguiente diagrama empezamos con los Diagramas exclusivos del Usuario Profesor

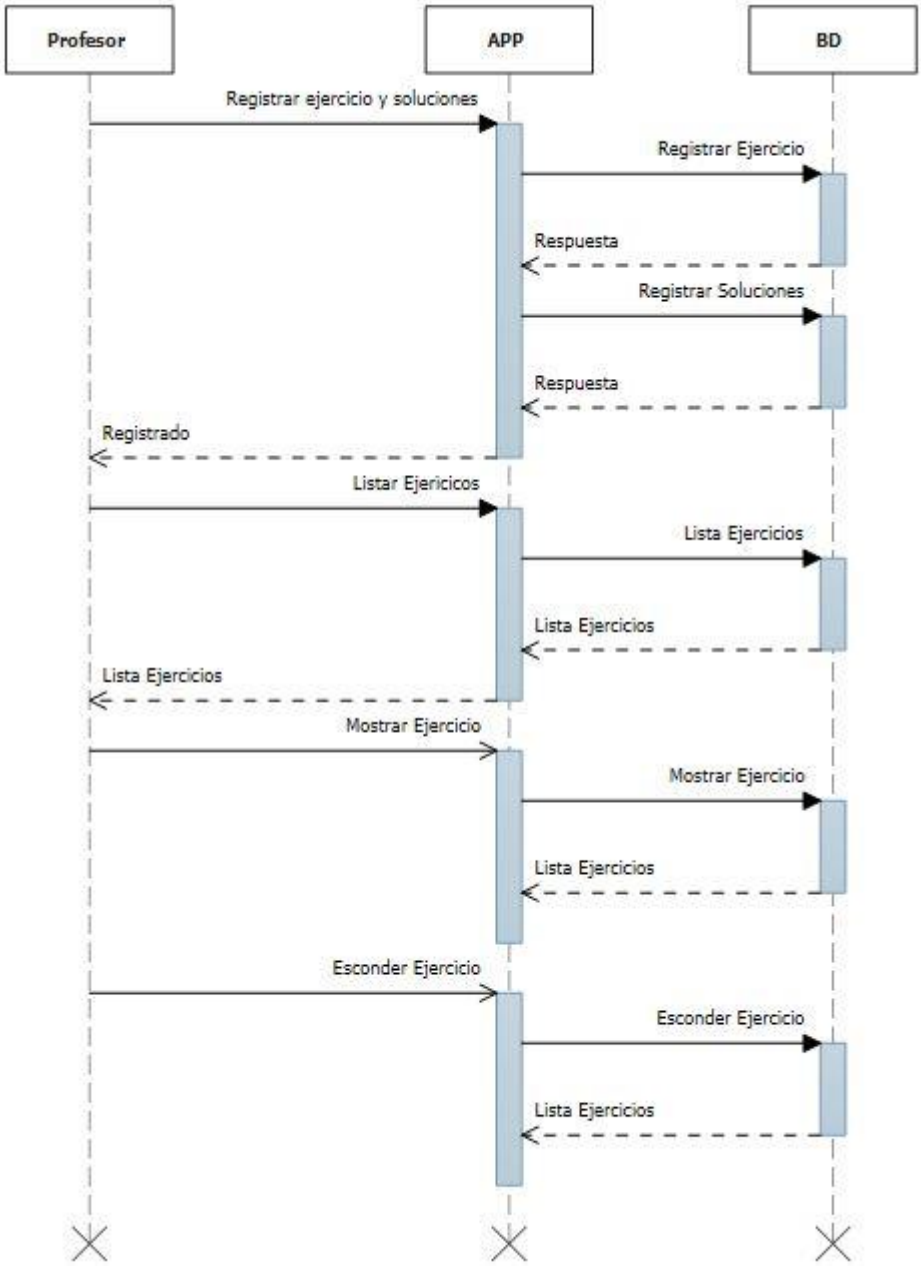

Ilustración 23. Diagrama de Secuencia Profesor - Gestión ejercicio

En la Ilustración 23 tenemos las cuatro principales acciones que puede realizar el profesor en la gestión de los ejercicios:

- Registrar ejercicio CUPR_03 (junto a Registrar solución)
- Listar ejercicios CUPR_06
- Mostrar ejercicios CUPR_07
- Esconder ejercicios CUPR_08

El funcionamiento de estas ejecuciones es prácticamente igual que las ejecuciones realizadas en los diagramas del Alumno. El Usuario manda datos a la aplicación, los

cuales tienen que ser cotejados, modificados o consultados a la Base de Datos, la cual responde a la Aplicación todas las peticiones solicitadas.

Lo más destacable en este diagrama es la doble consulta de registro (ejercicio y soluciones) que se realizan al solicitar a la Base de Datos que añada un ejercicio.

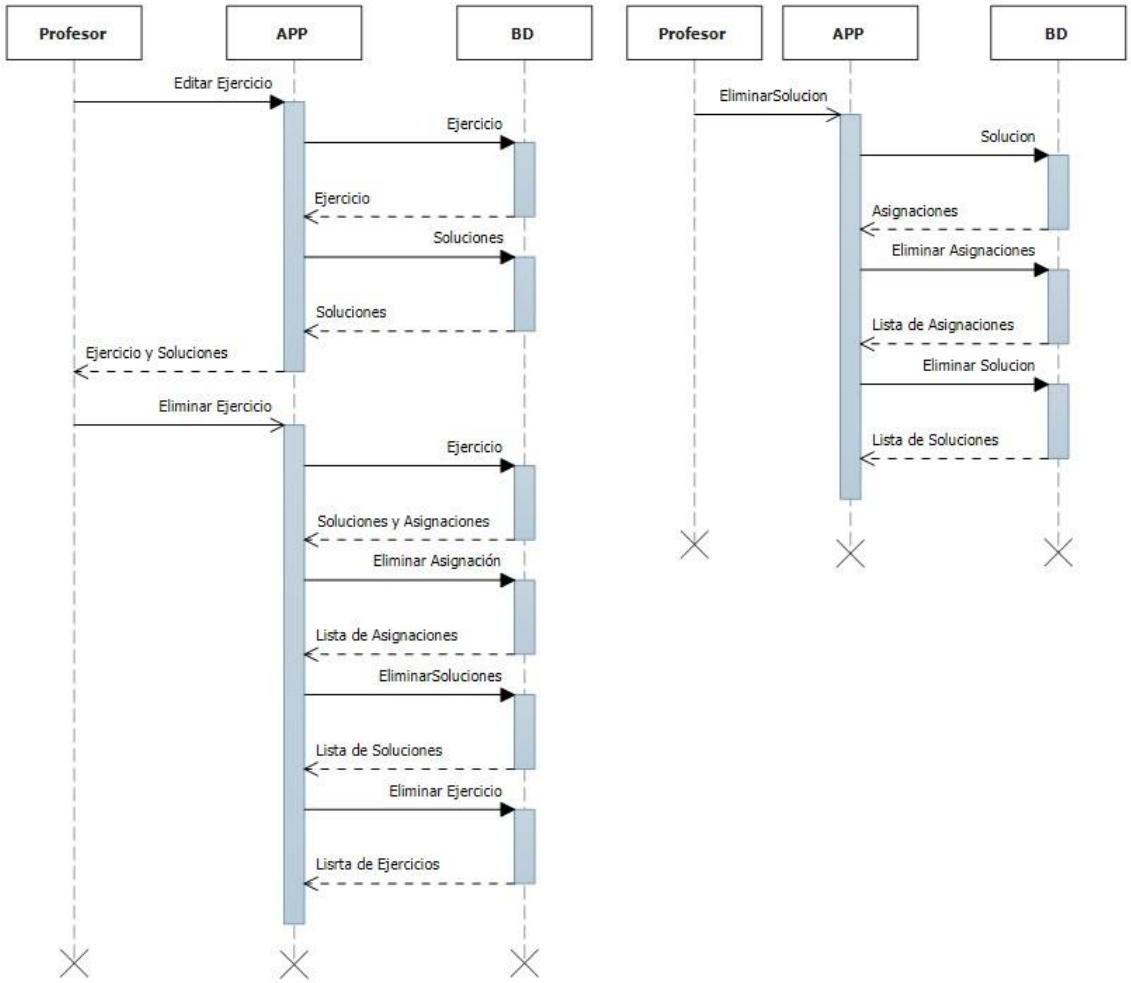

Ilustración 24. Diagrama de Secuencia Profesor - Edición Ejercicios

La Ilustración 24 muestra las principales secuencias de ejecución que puede realizar el profesor a la hora de editar un ejercicio:

- Eliminar Ejercicios CUPR_09
- Editar Ejercicios CUPR_10
- Eliminar Soluciones CUPR_11
- Editar Soluciones CUPR_12

Es estos diagramas podemos ver la diferencia entre las operaciones de Editar y de eliminar. Las acciones de editar son acciones síncronas, es decir, que esperan una

respuesta por parte del objeto demandado, en este caso la respuesta de la edición. En contraste, las acciones de eliminación son acciones asíncronas, es decir que no solicitan ningún tipo de respuesta por parte del objeto demandado.

Esto solo ocurre en las acciones de eliminación demandadas por el usuario. En el momento en el que la aplicación solicita la eliminación a la base de datos, dejan de ser comunicaciones asíncronas para convertirse de nuevo en comunicaciones síncronas, ya que la aplicación sí que necesita tener la lista actualizada con todos los parámetros en todo momento, (tanto asignaciones como ejercicios y soluciones).

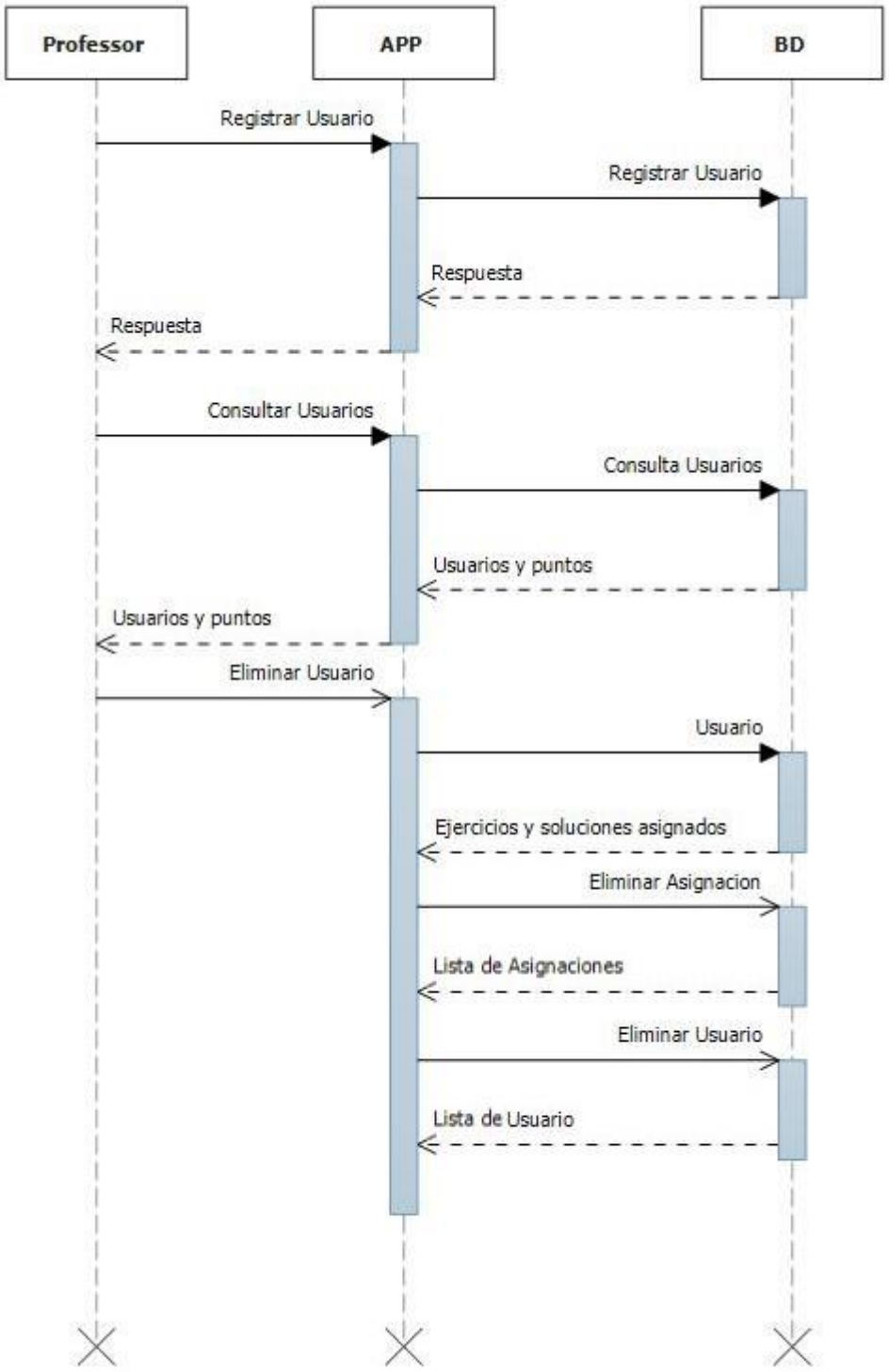

Ilustración 25. Diagrama de Secuencia Profesor - Gestión de Usuarios

En la Ilustración 25 tenemos representadas las últimas acciones que puede realizar un Profesor. En cuanto a las relacionadas con la gestión de los Usuarios:

- Registrar un Usuario CUPR_02
- Consultar Usuarios CUPR_04

- Eliminar un Usuario CUPR_05

El sistema utilizado para el registro y la eliminación es el mismo que se ha explicado en la Ilustración 24, relaciones sincrónicas para los registros, (esperando una respuesta de la Aplicación) mientras que las de eliminación, son asincrónicas por parte del Profesor y sincrónicas por parte de la Aplicación.

4.6 Prototipos del sistema

Los prototipos de un sistema son los diseños iniciales que se realizan en un primer momento para poder guiar a los diseñadores y poder ofrecer una idea de cómo va a ser la estructura gráfica del Sistema.

A continuación, presentaremos los diseños básicos de nuestro sistema:

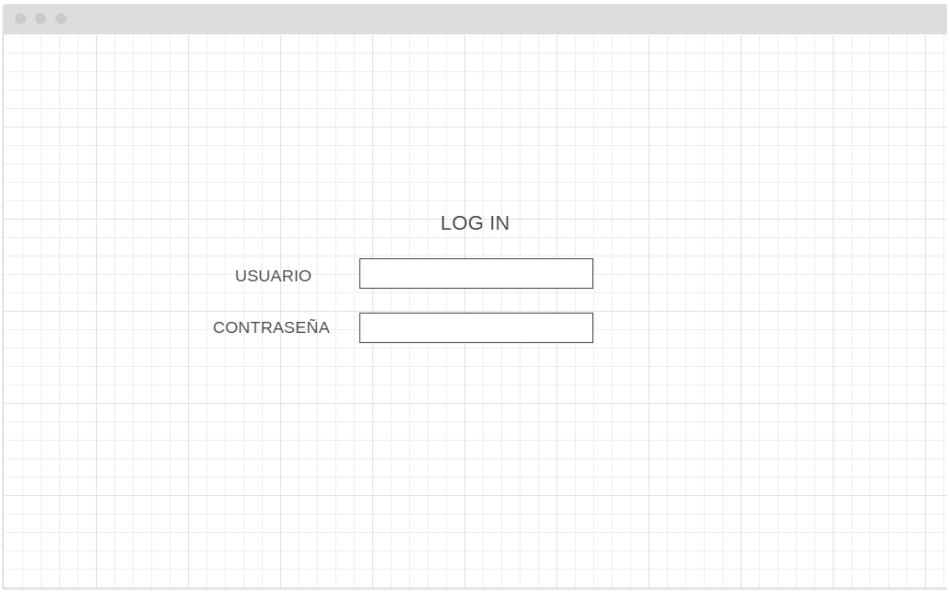

Ilustración 26. Prototipo Ventana Log-In

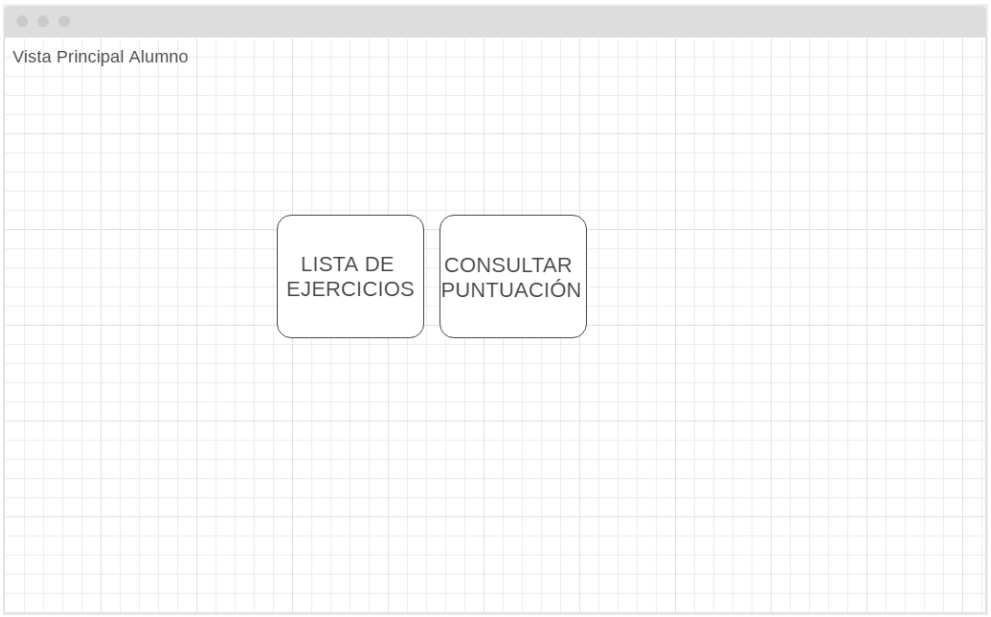

Ilustración 27. Prototipo Menu Alumno

Ilustración 28. Prototipo Menu Profesor

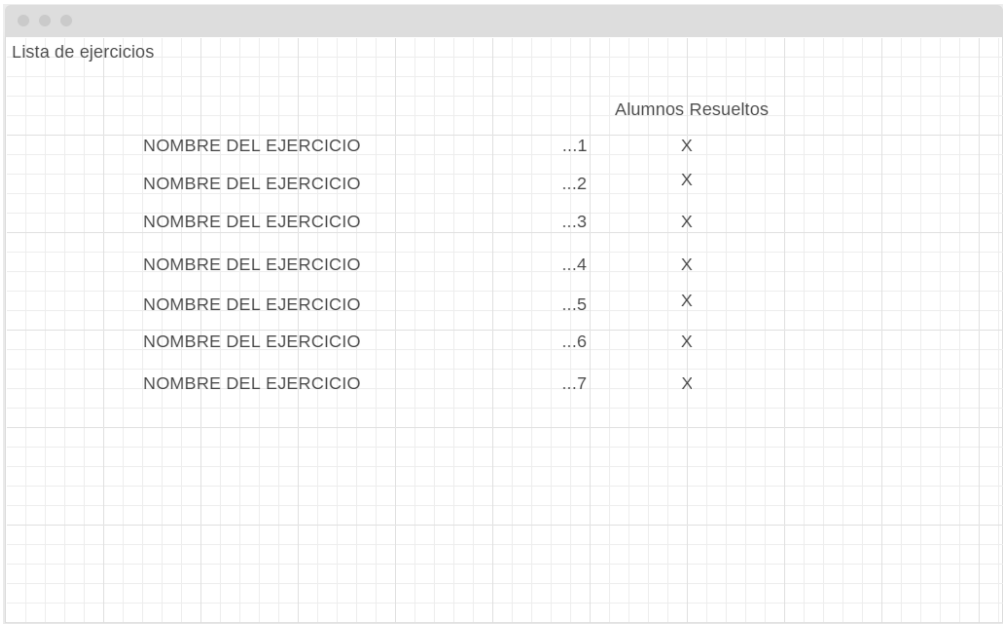

Ilustración 29. Prototipo Lista Ejercicios

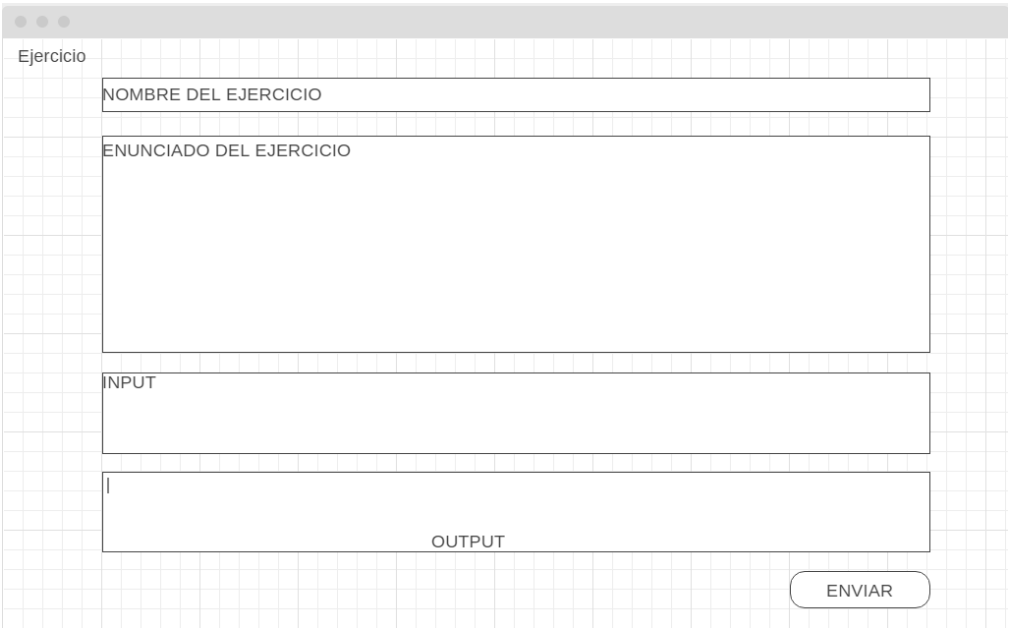

Ilustración 30. Prototipo Ejercicio Alumno

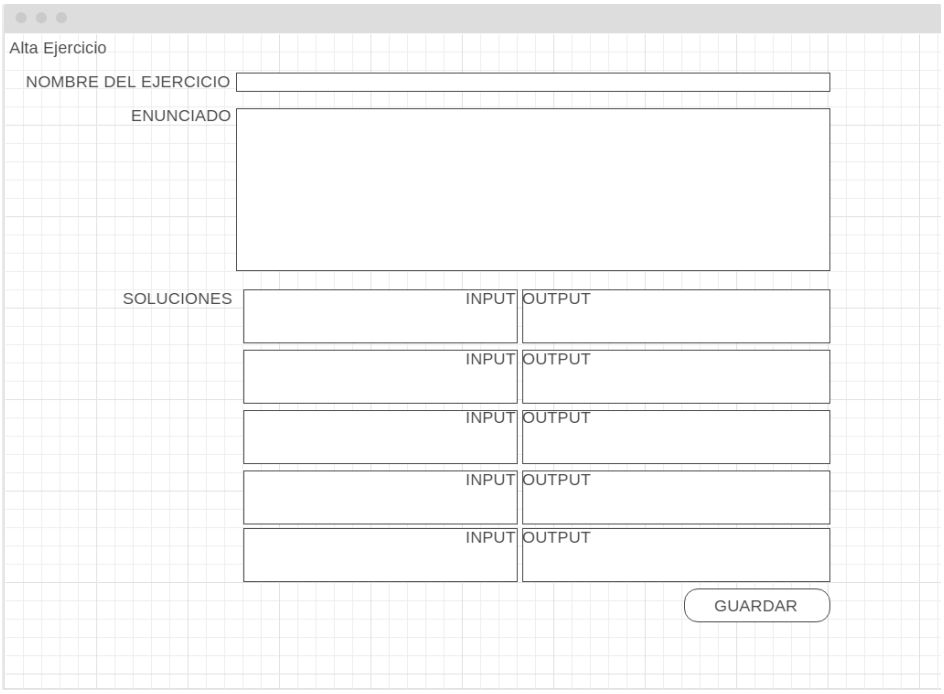

Ilustración 31. Prototipo Alta Ejercicio

Ilustración 32. Prototipo Lista Alumnos

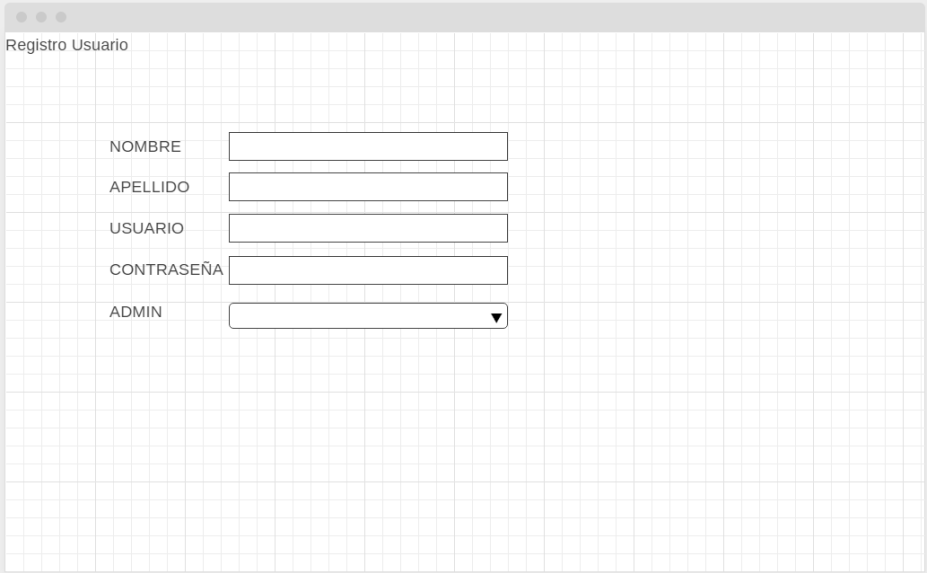

Ilustración 33. Prototipo Registro Usuario

5. Desarrollo del Sistema

En este capítulo vamos a hablar de la parte más técnica de este TFG. Hablaremos de la estructura utilizada en JSP para el desempeño de la aplicación, de las tres capas de programación que hemos utilizado y pondremos ejemplos de código sobre los elementos y estructuras más utilizados.

5.1 Estructura del Sistema

Para este trabajo hemos utilizado el lenguaje de programación JSP, un lenguaje basado en JAVA pero que funciona de forma diferente a la hora de la creación de capas de gestión.

Vamos a hablar de las capas de gestión utilizadas en “The Code Game” utilizando como base la imagen de la estructura final del proyecto.

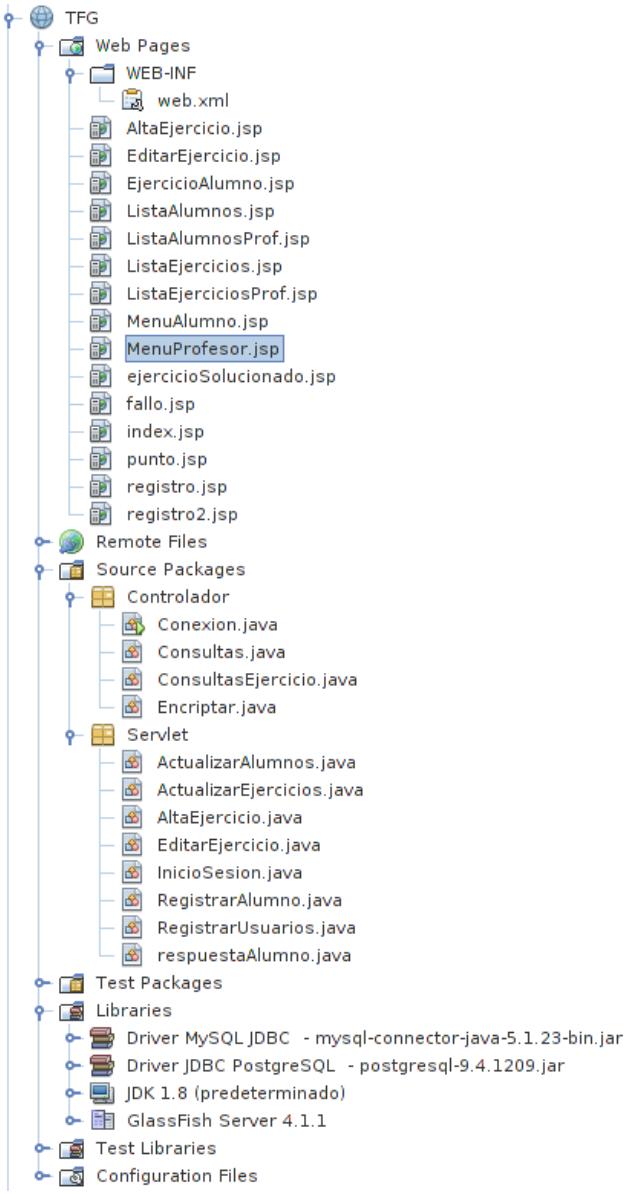

Ilustración 34. Estructura del Proyecto

Como podéis ver, cuando hablamos de código en NetBeans hablamos de proyecto. La Ilustración 34 muestra la estructura final en la cual podemos diferenciar claramente 3 capas:

- Web Pages
- Servlet
- Controlador

Cada una de estas capas tiene su función individual y su función global, por ejemplo, los web pages son los escenarios que se muestran en el navegador, dentro tienen todo lo necesario para la comunicación con la siguiente capa.

A continuación, vamos a hablar un poco mas en profundidad de cada una de ellas.

5.1.1 Web Pages

Los Web Pages son los archivos que contienen la vista gráfica de nuestro proyecto, cada uno de ellos, es una ventana diferente con una funcionalidad diferente.

Aunque la extensión del archivo sea .jsp, el código del interior es puramente código HTML, pero, en caso de ser necesario, se le pueden hacer insertos de código en cualquier tipo de lenguaje solo abriendo una etiqueta con el formato <% %> como podemos ver en el siguiente ejemplo.

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Registrarme</title>
    </head>
    <body>
        <%
        String idal= request.getParameter("idal");
%>
        <h1>Formulario de registro</h1>
        <form action="nuevousuario?idal=<%=idal%>" method="post">
            <label>Nombre</label>
            <input type="text" name="nombre"/></br>
            <label>Apellido</label>
            <input type="text" name="apellido"/></br>
            <label>Usuario</label>
            <input type="text" name="usuari"/></br>
            <label>Contraseña</label>
            <input type="password" name="contr"/></br>
            <input type="checkbox" name="isAdmin" value="Adminstrador"/> Administrador</br>
            <input type="submit" value="Registrar"/>
            <a href="MenuProfesor.jsp?idal=<%=idal%>"><input type="button" value="Atras"></a>
        </form>

    </body>
</html>
```

Ilustración 35. Ejemplo Código JSP

En este caso, la etiqueta insertada en la parte superior recoge un valor llamado idal, que se proporciona cuando se realiza una llamada a este archivo. Una vez recoge el dato, podemos realizar con las acciones que necesitemos y podemos mandarlo al sitio donde queramos. En este ejemplo lo utilizamos para mandarlo al servlet llamado nuevoregistro. Esto lo hacemos con la etiqueta "?idal=" junto al nombre del servlet. Posteriormente este dato será recogido para gestionar lo que sea necesario, en este caso para registrar al alumno.

No solo se pueden pasar datos entre jsp y servlet, al final de la imagen podemos ver un botón que llama al menú del profesor en el cual usamos el mismo método para pasar los datos.

Al tratarse de un entorno web, es muy probable que nos encontremos problemas que sea mucho más fácil resolver con JavaScript que con los propios métodos que nos aporta JSP.

Un ejemplo del caso lo podemos encontrar en el archivo de registro de ejercicio. Cuando accedes al navegador para registrar un ejercicio, a parte de los campos propios del ejercicio como son el enunciado y el título, aparece la parte de las soluciones asignadas al ejercicio. Como no sabemos la cantidad de soluciones que quiere proponer un profesor para cada ejercicio, decidimos implantar un botón "+" el cual al pulsarlo nos crea un campo de solución nuevo, de tal manera que el profesor puede elegir el número de soluciones que quiere registrar para cada ejercicio sin que exista un mínimo o un máximo.

```
            <button onclick="return duplicarCampo()">+</button>
            <div id="nuevoresultado">

                <label>INPUT</label>
                <input type="text" name="input0" value=""/>
                <label>OUTPUT</label>
                <input type="text" name="output0"/></br>
            </div>
            <button type="submit" name="Registrar"> REGISTRAR </button>
           <a href="MenuProfesor.jsp?idal=<%=idalum%>'><input type="button" value="Atras"></a>
        </form>

        <script>

        var cont=0;

function duplicarCampo(){

   cont++;
       $("#nuevoresultado").append("<label>INPUT</label><input type='text' name='input'+cont+''/><label>OUTPUT</label><input type='text' name='output'+cont+''/></br>");
       return false;


}
        </script>
```

Ilustración 36. Ejemplo JavaScript

Este botón decidimos implementarlo en JavaScript [14] (Ilustración 36) ya que con solo un par de líneas código podíamos tener solucionado el problema del registro de soluciones.

Al tratarse de código en HTML podemos introducir todo el código necesario dentro de las etiquetas <scrtipt> </script>.

5.1.2 Servlet

Los llamados servlets son clases java utilizadas para la comunicación entre la capa de web pages y la capa controlador.

Aunque tengan extensión .java, no son exactamente clases java ya que no hacen la misma función que pueden hacer este tipo de clases. Siguen su misma estructura, y utilizan sus mismos estilos (clases, tipos, …) pero se utilizan para recoger tanto datos que introduce el usuario como los propios datos internos de gestión del sistema.

La diferencia con una clase java es que puede recoger datos de las etiquetas de <input> utilizadas en HTML y guardarloe en variables de la tipología de java (String, Int, Boolean,...) para poder trabajar con ellos. A continuación mostramos un ejemplo:

```
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    request.setCharacterEncoding("UTF-8");
    PrintWriter out = response.getWriter();

    String respuesta = request.getParameter("respuesta");
    String id = request.getParameter("id");
    String idal = request.getParameter("idal");

    int idsol=Integer.parseInt(id);

    String output= Controlador.ConsultasEjercicio.getOutput(idsol);

    if(respuesta.equals(output)){
        if(Controlador.ConsultasEjercicio.estaSolucionado(idal,idsol)==true)
            response.sendRedirect("ejercicioSolucionado.jsp?idal="+idal);
        else{
        int puntos = Controlador.ConsultasEjercicio.getPuntos(idal);
        if(puntos>=0){
            Controlador.ConsultasEjercicio.sumarPunto(idal, puntos);
            Controlador.ConsultasEjercicio.solucionar(idal,idsol);
            response.sendRedirect("punto.jsp?idal="+idal);

            }
        }
```

Ilustración 37. Ejemplo Servlet

Como podemos ver en la Ilustración 37, un servlet tiene la estructura similar a cualquier clase java, pero vemos que dentro tiene dos elementos principales con los que trabajaremos para la obtención y el manejo de datos:

- **Response:** utilizado para la salida de datos. Como podemos ver en el ejemplo, utilizamos response para llamas a otro jsp, es decir, para poder acceder a otro sitio, lo utilizamos como puerta de salida a otros jsp incluso a otros servlets.
- **Request:** utilizado para recoger los datos que llegan de la capa de Web Pages. Al principio del Servlet vemos un grupo de parámetros que utilizar *servlet.getParameter("nombre del parámetro");* para la obtención de los datos mandados por el usuario o por el sistema con este nombre.

Una vez tenemos los datos necesarios pasamos a la gestión y a las operaciones con ellos, en la parte inferior tenemos un conjunto de llamadas a métodos de la clase Controlador. Esto significa que manda los datos para que sean gestionados desde allí.

Aquí podemos ver que esta capa se utiliza como nexo entre las otras dos.

5.1.3 Controlador

La clase controlador es la capa mas profunda de la estructura de JSP, es la que se encarga de la gestión y el manejo de los datos para que ya sean almacenados, eliminados, ... de la base de datos.

La estructura de la capa controlador de nuestro proyecto es muy simple. Consta de una clase padre, Conexión, que es la que se encara de abrir la conexión son la base de datos. Gestiona las conexiones y las cierra cuando es necesario. Todas las otras clases existentes en esta capa (a excepción de la clase encriptación) heredar de esta clase conexión, es decir, que todas son de tipo conexión.

```
public static Connection getDBConnection(){
    Connection con = null;
    try {
        Class.forName(CLASSNAME);
        con = DriverManager.getConnection(URL, USERNAME, PASSWORD);
        return con;
    } catch (ClassNotFoundException e) {
        System.err.println("ERROR "+ e);
    } catch(SQLException e){
        System.err.println("ERROR "+ e);
    }
    return con;
}
```

Ilustración 38. Clase Conexión

En esta capa está implementado el patrón singleton, el cual se encargada de restringir la creación de objetos pertenecientes a una misma clase a un objeto único, es decir, garantizar que una clase solo tenga una instancia que en este caso es la de conexión.

Las clases de tipo conexión contienen métodos de conexión a la base de datos, select, insert, update y delete. La estructura seguida para la generación se consultas es la siguiente:

```
public static Boolean registrado(String user){
PreparedStatement pst = null;
    Connection con = null;
    ResultSet rs = null;
    String sql = null;

    try {
         Class.forName("com.mysql.jdbc.Driver");
        con = getDBConnection();
        String consulta = "select * from usuarios where usuario like '"+user+"'";
        pst = con.prepareStatement(consulta);
        rs=pst.executeQuery();

        while(rs.next()){
           return true;
        }

    } catch (Exception e) {
        System.out.println("ERROR " + e);
    } finally{
        try {
             if (con != null )
                 con.close();
             pst.close();

        } catch (Exception e) {
          System.err.println("ERROR " + e);
        }
    }

    return false;
}
```

Ilustración 39. Ejemplo Consulta Base de Datos

Vemos en la Ilustración 38 el ejemplo de conexión para comprobar si un usuario está registrado en la base de datos. Si nos fijamos podemos ver que para la ejecución de la consulta generamos un preparedStatement que utilizaremos para lanzar la consulta a la base de datos, y un resultSet en el que guardaremos los datos obtenidos al realizar la dicha consulta.

Una vez obtenemos los datos y los tenemos en el resultSet, tenemos que comprobar que no esté vacío, operación que haremos con ayuda de un bucle while.

Una vez recogidos y guardados en los parámetros necesarios, estos datos son utilizados para, generar una respuesta al método. Esta respuesta será enviada de nuevo a la capa controladora para que siga con su ejecución dependiendo de las respuestas obtenidas.

Finalmente, se cierran el preparedStatemnt, el resultSet y la conexión con la base de datos para poder dejar paso a una nueva.

6. Trabajo Futuro

En este apartado hablaremos de las diferentes opciones y funcionalidades que podemos añadir a nuestro proyecto en futuras versiones.

Este apartado es necesario ya que esta aplicación está desarrollada desde cero sin ninguna otra aplicación con este mismo propósito como referencia. Es verdad que la idea está basada en la plataforma "www.adventofcode.com" y utiliza la idea principal para la implementación, la idead e la competición, pero actualmente en el estándar Sakai y en el grupo de Learning Tools Interoperability (LTI) no hay ninguna aplicación similar, por lo que esta versión inicial carece de muchas de las funcionalidades que se le pueden aportar.

6.1 Ideas de mejora

En las siguientes fases de implementación y en las siguientes ampliaciones, una de las ideas principales que tenemos es poder implementar y aplicar en nuestro sistema el estándar de comunicación de Sakai, para poder integrar la aplicación en el grupo de LTI y poder utilizar la aplicación desde el propio sistema de PoliformaT.

Esta idea es en la primera que nos enfocaremos, ya que al ser un proyecto encarado a la motivación y a la generación de iniciativa a la hora de programar, que mejor manera de implantarlo en las asignaturas que como un plug-in libre y gratuito integrado en la plataforma principal de la universidad.

Otra de las posibles ideas de mejora es la creación de grupos de estudiantes y la creación de ejercicios específicos para cada grupo. Esto sería beneficioso para los profesores que imparten más de una asignatura, ya que lo podrían utilizar indistintamente dentro de la cada una de ellas sin que haya mezcla de datos. Al ser un sistema que no compila código, se puede dar la libertad de utilizar el sistema para diferentes lenguajes de programación.

Otra idea que nos llamó la atención desde el principio es la creación de un apartado tanto para alumnos como para profesores donde pudieran añadir el código utilizado par la resolución de un ejercicio, de esta manera el profesor podría ver, corregir y opinar sobre el código que han utilizado sus alumnos.

También se habló en las primeras reuniones de la creación de una penalización cada vez que un alumno intente más de X veces la resolución de un ejercicio de forma errónea. Se habló de penalización en la puntuación o de bloqueo temporal del ejercicio para que no se pudiera resolver en un tiempo limitado.

Por último, se podría implementar una funcionalidad específica para profesores en la que se pudiera ordenar los ejercicios en base al criterio que el profesor desee, es decir, dar la opción al profesor de ordenar la lista de los ejercicios mostrados.

Esta aplicación actualmente es totalmente funcional y se pondrá a prueba en la asignatura de Lenguajes Teoremas y Paradigmas de la programación (LTP) impartida en el segundo curso de la carreara de Grado de Ingeniería Informática durante el desarrollo del curso que viene. Esta será probada por la profesora Alicia Villanueva, esperemos que con un resultado satisfactorio.

Al ser una aplicación que no se ha testeado en el entorno estudiantil, no podemos aportar pruebas de integración de resultados ni pruebas de testing.

7. Bibliografía

[1] «ADVENT OF CODE,» [En línea]. Available: https://adventofcode.com/. [Último acceso: 15 Abril 2019].

[2] «LEARNING TOOL INTEROPERABILITY,» [En línea]. Available: https://www.imsglobal.org/activity/learning-tools-interoperability. [Último acceso: 12 Junio 2019].

[3] «LEARNING MANAGEMENT SYSTEM,» [En línea]. Available: https://www.nubily.com/contenido/plataforma-lms/. [Último acceso: 12 JUNIO 2019].

[4] «DIGIEXAM,» [En línea]. Available: https://www.digiexam.com/. [Último acceso: 13 JUNIO 2019].

[5] «CIRCLEIN,» [En línea]. Available: https://www.circleinapp.com/. [Último acceso: 13 JUNIO 2019].

[6] «NETBEANS,» [En línea]. Available: https://netbeans.org/. [Último acceso: 2 MAYO 2019].

[7] «JAVA,» de *EMPEZAR A PROGRAMAR USANDO JAVA*, Valencia, UPV, 2012.

[8] «ECLIPSE,» [En línea]. Available: https://www.eclipse.org/ide/. [Último acceso: 2 MAYO 2019].

[9] «BLUEJ,» [En línea]. Available: https://www.bluej.org/. [Último acceso: 2 MAYO 2019].

[10] «APACHE XAMPP,» [En línea]. Available: https://www.apachefriends.org/es/index.html. [Último acceso: 30 ABRIL 2019].

[11] «MYSQL,» de *JSP, servlet and MySql*, Professional Minware, 2001.

[12] «JSP,» [En línea]. Available: https://www.oracle.com/technetwork/java/javaee/jsp/index.html. [Último acceso: 6 MAYO 2019].

[13] EL LENGUAJE UNIFICADO DE MODELADO (UML), VALENCIA: UPV.

[14] Web Design with HTML, CSS, JavaScript and jQuery Set, WILEY, 2014.