UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

Desarrollo de un IDE para el Desarrollo y la Corrección Automatizada de Ejercicios de Programación

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Juan Ramallón Martínez

Tutor: Josep Silva Galiana

Curso 2019-2020

A mi familia le debo todo,

por darme una buena educación,

por siempre estar a mi lado,

y por nunca dejarme caer,

Mil gracias.

Resumen

Este proyecto aborda el diseño y desarrollo de un entorno de desarrollo integrado (IDE) para la plataforma ASys. ASys es un sistema para la gestión y corrección de ejercicios de programación de la Universitat Politècnica de València. En este proyecto se plantea la parte consistente en el desarrollo de ejercicios, así como su corrección semi-automática.

La funcionalidad aportada por este proyecto incluye un módulo de autocorrección de ejercicios que tendrá una visión para el profesor y otra para el alumno. El sistema ofrece al profesorado herramientas que ahorran grandes cantidades de trabajo al detectar e incluso corregir automáticamente gran parte de los errores. Por otra parte, el alumnado dispone de un sistema de autoaprendizaje, donde el sistema detecta todos los errores y le ofrece pistas para corregirlos de manera guiada.

Todo el sistema desarrollado es accesible online a través de cualquier navegador. La parte cliente ha sido desarrollada con Vue, mientras que la parte servidora ha sido desarrollada con Spring.

Palabras clave: ASys, corrección semi-automática, autocorrección, autoaprendizaje, ejercicios.

Abstract

This project tackles the design and development of an *integrated development environment* (IDE) for the ASys platform. ASys is a system in development for the management and assessment of programming exercises at the Universitat Politècnica de València. This project addresses the part consisting in the development of exercises, as well as its semi-automatic correction and marking.

The functionality provided by this project includes a module for exercise self-assessment that provides a vision for the teacher and another for the student. Teachers have tools that will save large amounts of work by automatically detecting and even correcting most errors in the student's submission. On the other hand, students have a self-learning system, where the system can detect all errors and give them clues to correct them in a guided way.

The entire system is accessible online through any browser. The client part has been developed with Vue, while the server part has been developed with Spring.

Keywords: ASys, semi-automatic assessment, self-assessment, self-learning, exercises.

Tabla de contenidos

0.TABLA DE FIGURAS

1. INTRODUCCIÓN

1.1 Motivación

En un primer lugar, pensaba hacer el TFG mientras realizaba las prácticas de empresa, sin darle mayor relevancia al tema del proyecto. Mi preocupación por aquel entonces era simplemente finalizarlo cuanto antes. Un día me reuní con mi tutor, Josep Silva, y tras comentarle mis expectativas y objetivos, me propuso un abanico de posibilidades para realizar mi proyecto. Entre todos ellos me llamó la atención el sistema ASys. Un proyecto fruto de una investigación en mi propia universidad (la Universitat Politècnica de València), que él mismo coordinaba y cuyo futuro es más que prometedor.

El sistema ASys es en esencia una red social de estudiantes y profesores. Tiene como objetivo central la gestión del aprendizaje, pero incluye novedosas ideas como la corrección automática, la corrección cruzada, la gamificación, el uso de pistas para resolver ejercicios, etc.

Sentí que ASys estaba destinado a triunfar, que iba a renovar el futuro docente y supe que no podía rechazar la oportunidad de participar en ese proyecto.

1.2 Objetivos

El objetivo primordial del presente proyecto es la creación de un *entorno de desarrollo integrado* (IDE) para el desarrollo y la corrección automatizada de ejercicios de programación. Este entorno permitirá desde cualquier navegador realizar y gestionar todos los ejercicios asociados a un alumno o a un profesor. Y lo más importante, las soluciones asociadas a dichos ejercicios podrán ser corregidas tanto por los propios alumnos, como por los profesores, que verán reducido significativamente su trabajo en este ámbito.

Así bien, al finalizar este proyecto se pretende tener abordados los siguientes objetivos:

(1) Dar soporte a la resolución de ejercicios de programación desde cualquier navegador. Esto incluye la codificación, la compilación y la ejecución de casos de test.

(2) Dar soporte al envío de soluciones al sistema para su corrección por un profesor.

(3) Dar soporte a la corrección automática y semi-automática del código (tanto por profesores como por alumnos). Dicha corrección deberá realizarse en la máquina local del cliente para reducir la carga del servidor.

(4) Dar soporte a la puntuación de ejercicios por parte del profesor y a la introducción de anotaciones y correcciones.

1.3 Impacto esperado

Tras la realización de este trabajo se espera conseguir un gran avance en la docencia a nivel internacional, puesto que el sistema ASys será una plataforma docente accesible desde cualquier universidad del mundo. La implantación inicial se realizará en

la Escuela Técnica Superior de Ingeniería Informática (ETSINF) de la Universitat Politècnica de València (UPV). Se prevé conseguir los siguientes avances:

- En primer lugar, una mejora en el rendimiento académico. Los alumnos mejorarán su aprendizaje y sus resultados académicos, pues contarán con herramientas que les permitirán aprender con mayor facilidad.
- En segundo lugar, un aumento en la objetividad de la corrección. La corrección semi-automática de ASys elimina factores subjetivos de la corrección al aplicar los mismos criterios de corrección a todos los alumnos, independientemente del orden y el momento de la corrección.
- En tercer lugar, se aumentará la autonomía de los alumnos. El nuevo sistema permite al alumno detectar sus propios errores y corregirlos sin interacción del profesor.
- Por último, se pretende reducir considerablemente la cantidad de tiempo que los profesores han de invertir en la corrección de ejercicios y exámenes, gracias a la automatización del sistema corrector.

Cabe añadir, que todo esto será accesible desde el navegador, puesto que ASys no necesitará ningún tipo de instalación por parte de los usuarios y será accesible para todos aquellos con acceso a una red de internet.

1.4 Metodología

Para realizar este trabajo, se han diferenciado tres fases principales: una de **formación**, una de **autoaprendizaje** y otra de **implementación**. Teniendo en cuenta la siguiente cita de Steve Souders, Head Performance Engineer, Google, 2013:

"Good developers know how things work.
Great developers know why things work."

se pretende hacer hincapié en la importancia de saber por qué funciona el proyecto en el que vamos a trabajar; siempre hay que tener en cuenta la misión del proyecto.

La primera fase fue una fase de **formación**, donde absorbí todo el conocimiento que pude de mis tutores. En ella aprendí por qué se quería hacer el proyecto y sobre todo para quién iba dirigido, así como su ámbito de aplicación.

La segunda fase fue esencialmente una fase de **preparación o formación**, que se caracterizó por ser una toma de contacto con las tecnologías y los lenguajes usados en la implementación de ASys.

En esta segunda fase, me formé en *Vue (Front-end)* [1,2,3,4,11,12], el *framework* que se utiliza para los desarrollos de la parte cliente, como la construcción de las interfaces de usuario; y en *Spring (Back-end)*[5,6,13], el *framework* que se utiliza para los desarrollos de la parte servidora, y que incluye herramientas y utilidades como el sistema gestor de bases de datos, el servidor de correo electrónico, o el gestor de archivos.

Tras esta fase de toma de contacto con la tecnología que iba a utilizar, comenzó la tercera fase, la cual consistió en el propio **diseño e implementación** del IDE.

Finalmente, se terminó con una fase adicional de pruebas donde se validó el trabajo realizado, buscando errores y posibles defectos que pudiera tener el proyecto.

Por supuesto, la detección de defectos supuso una nueva iteración de la fase dos hasta alcanzar un producto final sin errores detectados.

La metodología de desarrollo seguida ha sido una metodología clásica en cascada, tal y como se detalla en la estructura del documento, que se explica a continuación.[8-9-10-14-15-16]

1.5 Estructura

"*Software processes are software, too*" escribió Lee Osterweil, Professor of Computer Science de la University of Massachusetts. Quería comenzar este apartado con esta cita porque el proceso software es un punto fundamental del proyecto.

La estructura del proyecto sigue un proceso iterativo e incremental que ha seguido las fases del ciclo de vida clásico [7,17]

- **Identificación de requisitos**. En esta fase se identifican las necesidades concretas de los usuarios.
- **Especificación de requisitos**. Definir los requisitos en detalle y siguiendo un estándar para poder trazarlos, evitando ambigüedades que dificulten su futuro desarrollo.
- **Validación de requisitos**. Comprobar que dichos requisitos son correctos.
- **Análisis del problema**. Se indica cómo abordar los requisitos ya definidos en la etapa anterior.
- **Diseño de la solución**. Se formaliza el trabajo a realizar y en nuestro caso, coincide con la creación del *Backlog* y de los diferentes *Sprints*.
- **Desarrollo de la solución**. Se comienzan a detallar las tareas a realizar (elementos del *Backlog*).
- **Programación**. Se abordan una a una todas las tareas contenidas en el *Backlog*, consiste en reducir un diseño a código.
- **Pruebas**. Consiste en comprobar que el software desarrollado realiza todas aquellas tareas indicadas en la fase de especificación de requisitos correctamente.
- **Documentación**. En esta fase se redacta todo aquello relacionado con el trabajo realizado, incluyendo todas aquellas referencias a los recursos utilizados.

1.6 Colaboraciones

El proyecto ha sido realizado gracias a la ayuda de mis tutores, Josep Silva Galiana, y Armando Maya Gomis, que me han aconsejado y asesorado durante todo el proyecto.

2. ESTADO DEL ARTE

En este apartado vamos a repasar las aplicaciones actuales que más se asemejan a nuestro producto. En particular, existen actualmente diversas aplicaciones web que nos permiten realizar ejercicios de programación. Vamos a describir las más destacadas en la comunidad y a ver las diferencias con ASys.

En primer lugar, tenemos a Solveet (http://ww1.solveet.com/) [18], una plataforma de retos de programación creada Rubén Bernárdez. En ella se resuelven retos, se desafía a otros usuarios y se discuten diferentes cuestiones. Es un lugar en el que cualquiera puede compartir su código demostrando sus razonamientos. Así bien, esta plataforma incentiva mucho a la participación y a la obtención de una multitud de soluciones válidas. ASys en cambio, apuesta por una única solución de referencia aportada por el creador del ejercicio. Se darán por válidas todas las soluciones que superen los casos de test asociados a la solución de referencia.

Figura 1: Pantalla principal del sistema Solveet

Otra aplicación es Topcoder (https://www.topcoder.com/challenges) [19], una plataforma, compuesta por casi un millón de usuarios, en la que se compite en decenas de retos de programación y diseño a nivel internacional. Topcoder, utiliza un sistema de corrección de ejercicios bastante similar a ASys; sin embargo, no es gratuita. El mismo caso sucede con Codeacademy (https://www.codecademy.com/) [20], tiene la misma base funcional que Topcoder y cuenta con una versión *free trial*, y en el caso de querer acceder a toda su funcionalidad podemos adquirir la versión de pago Codeacademy Pro (https://www.codecademy.com/pro/membership) [21].

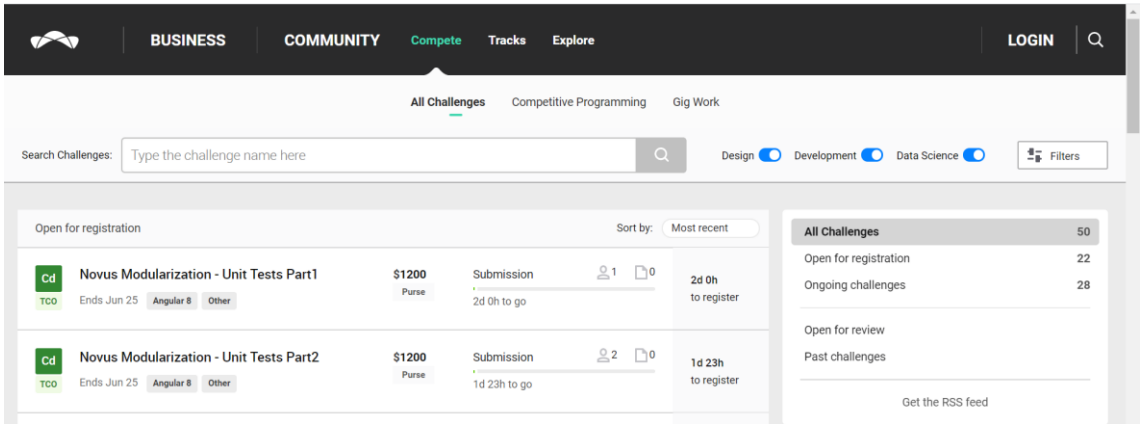

Figura 2: Página de desafíos del sistema Topcoder

HackerEarth (https://www.hackerearth.com/challenges/) [22], al igual que TopCoder, cuenta con numerosos problemas para poner a prueba las habilidades de programación de los usuarios. Uno de los puntos por los que se caracteriza esta página es que permite ejecutar y probar el código desde el propio *browser*. Esta plataforma es la más parecida a la utilización del IDE resultado de este proyecto, sin embargo, ASys nos irá ayudando progresivamente en la resolución del ejercicio, lo que será mucho más efectivo en términos de aprendizaje.

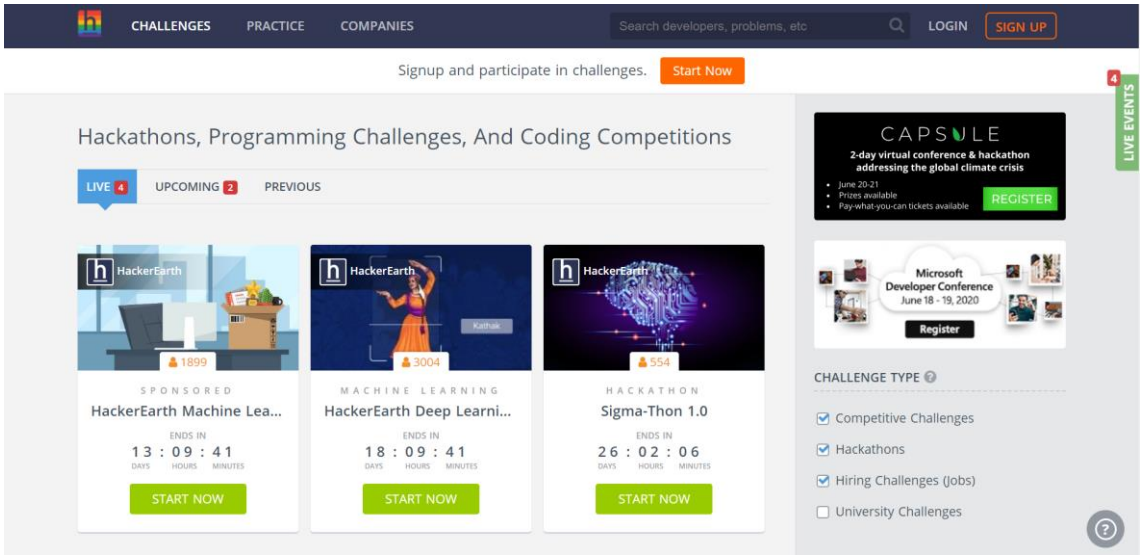

Figura 3: Página de desafíos del sistema HackerEarth

Además de estas plataformas de realización de ejercicios, también hay un gran número de plataformas dedicadas a la docencia web, PoliformaT (https://poliformat.upv.es/) [23] sería un ejemplo de ello. Esta plataforma de la Universidad Politécnica de València, utiliza un sistema que consiste en: la descarga del material por parte de los alumnos, seguido de una realización de dichas tareas en los propios ordenadores de los alumnos, tras la resolución se envían de vuelta las soluciones a la plataforma y, por último, los profesores corrigen manualmente una a una las soluciones. Como se puede observar, es un sistema considerablemente mejorable.

También cabe comentar la plataforma Code.org (https://code.org/) [24], que últimamente ha entrado en auge de popularidad entre la comunidad de resolutores de ejercicios de programación, especialmente entre los más jóvenes. Así bien, en Code.org

también podemos ver una amplísima selección de enlaces donde podemos encontrar tutoriales de todo tipo, desde lenguajes de programación hasta explicaciones sobre cómo crear páginas web e incluso aplicaciones para móviles. Como curiosidad el propio Steve Jobs, cofundador de Apple, dijo lo siguiente refiriéndose a esta plataforma: "Todo el mundo en este país debería aprender a programar un ordenador, porque lo que aprendes es a cómo pensar".[25]

Figura 4: Pantalla principal de la plataforma Code.org

Por último, se va a comentar el IDE de la aplicación web Repl (https://repl.it/) [26] pues ha sido una referencia para este trabajo. Repl es una potente herramienta que cuenta con un *In-browser* IDE, con colaboración en tiempo real con otros usuarios (amigos, compañeros de equipo…) mediante *Google-docs*, puedes programar en más de 50 lenguajes e integración con Git-hub, entre otras muchas funcionalidades. Es evidente que esta herramienta es increíblemente potente y es una referencia perfecta a la hora de enfocar este proyecto, pues hay grandes ideas y funcionalidades que pueden ser adaptadas para nuestro uso. En nuestro caso hemos desechado la idea de compartir el código en tiempo real y resolver estos ejercicios en equipo, pues nuestra aplicación está orientada a la mejoría personal del alumno en base a sus propios méritos, pues se basa en la visión más convencional de la enseñanza.

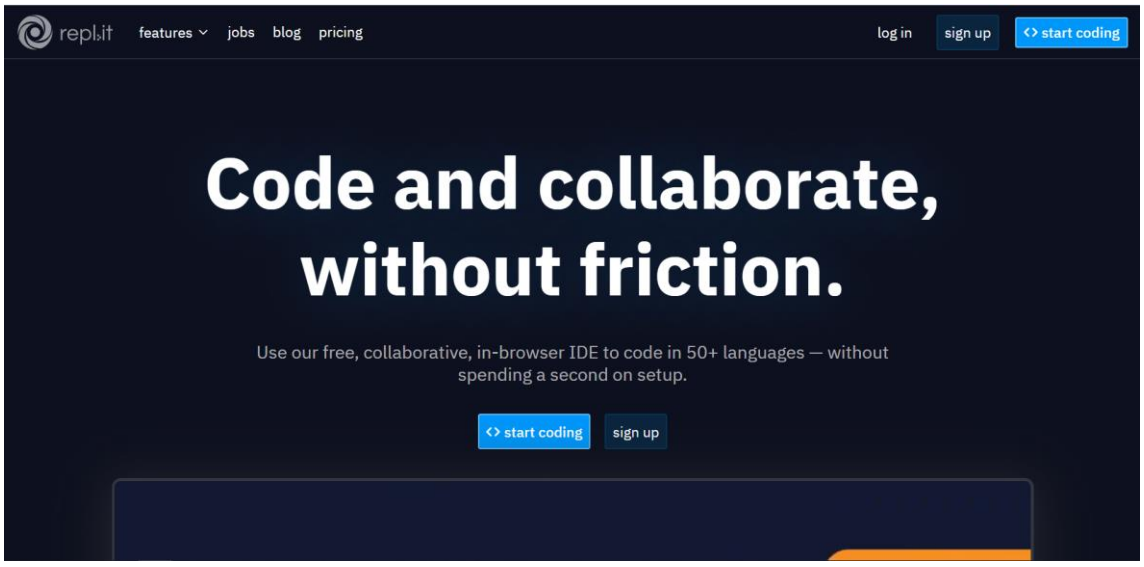

Figura 5: Pantalla principal del sistema Repl

2.1 Crítica al estado del arte

En resumen, en la actualidad ya se disponen de muchas plataformas online que nos permiten, entregar o realizar ciertos ejercicios (input) y nos devuelven una corrección o incluso una titulación (output), como seria en el caso de Udemy (https://www.udemy.com/) [27]. Pero la gran mayoría de estas plataformas son de pago, no tienen un IDE donde poder desarrollar el propio input, o bien, una vez realizada la entrega de dicho input, tenemos que esperar un tiempo considerable para obtener el output.

Por ejemplo, si quisiéramos realizar simplemente un ejercicio y obtener una nota resultado de una corrección. En primer lugar, tendremos que resolver el ejercicio en un IDE cualquiera, diferente a la plataforma que nos ofrece el ejercicio. Una vez resuelto, subiríamos la solución a la plataforma que, normalmente tras un proceso manual, será corregida. Todo esto implica, mayor tiempo en materia de corrección, poca escalabilidad, y poca usabilidad pues para una actividad tan sencilla se ven implicadas plataformas diferentes.

Existen plataformas que sí tienen un IDE propio, y que además son capaces de evaluar el input y retornar una nota o un error. Pero todas ellas están basadas en el uso de casos de prueba. Si el input pasa los casos de prueba, se considera correcto. Si el input no los pasa, entonces se retorna como error el propio caso de prueba como contraejemplo. No conocemos ninguna plataforma que sea capaz de retornar una evaluación no funcional, es decir, basada en la estructura y las propiedades del código fuente entregado como input. Tampoco existe ninguna plataforma que corrija el input.

2.2 Propuesta

Las carencias en el estado del arte comentadas en el punto anterior son el desencadenante que ha motivado la realización de este proyecto. Se ha marcado como objetivo desarrollar un IDE e integrarlo en ASys. Este IDE deberá permitir la resolución y corrección de los ejercicios en una misma plataforma. Además, permitirá la auto-corrección, la corrección asistida, y la corrección automática de propiedades, de la estructura del código y, por supuesto, funcionales, basadas en casos de prueba. Todo esto supondrá un avance significativo en el estado del arte y reportará una notable mejora en la experiencia del usuario.

3. ANÁLISIS DEL PROBLEMA

3.1 Especificación de requisitos

El propósito de este documento es analizar y definir las necesidades que debe satisfacer el sistema ASys. [28, 29, 30]

3.1.1 Visión

En este apartado se pretende aportar una perspectiva general de todo lo relacionado con el proyecto ASys, es decir, nos dará el alcance del proyecto. Así bien, se analizarán los actores que interactuarán con el sistema y los requisitos tanto funcionales como no funcionales que el sistema deberá cumplir.

3.1.1.1 Actores

Un actor es toda entidad que de alguna forma demanda funcionalidad al sistema. Así bien, dentro de ASys podemos destacar los siguientes actores:

- Usuario no autenticado. Persona que aún no ha sido introducida en la BD del sistema.
- Usuario autenticado. Persona que ya ha usado previamente el sistema y, por lo tanto, ya ha sido introducida en la BD de éste.
- Alumno. Es un tipo de usuario autenticado, cuyo caso de uso principal será la resolución de ejercicios y exámenes.
- Profesor. Es un tipo de usuario autenticado, cuyo caso de uso principal será la corrección de ejercicios y exámenes.
- Administrador. Es un tipo de usuario autenticado, cuya finalidad dentro del sistema será la de modificar la información de este.

3.1.1.2 Requisitos funcionales

Los requisitos funcionales son aquellos servicios que proveerá el sistema como resultado de recibir entradas particulares. A continuación, se señalan los requisitos funcionales del proyecto:

- Gestión de ejercicios. Se encuentran todas aquellas acciones relacionadas con los ejercicios y su resolución.
- Sistema de corrección:
    - Siempre se ha de corregir en modo automático y solamente cuándo el usuario lo desee, podrá modificar dicha corrección, y, en ese momento, será semi-automática.
    - El usuario debe poder visualizar la solución original del ejercicio.
    - El usuario debe ser capaz de poder rehacer la corrección.
    - El usuario siempre visualizará las soluciones en orden descendente filtrando por nota.

En psicología el orden de los factores altera el producto (de nuestra mente) y con esto se conseguirá evadir el efecto inconsciente que produce pasar de una solución excelente a una solución pésima, y viceversa.

Daniel Kahneman, catedrático en psicología, dice en su libro "pensar rápido, pensar despacio": **"Lo que leemos primero cambia nuestra percepción de lo que leemos a continuación",** este efecto sumado al efecto halo hacen que la corrección tradicional de los profesores sea bastante subjetiva, con ASys estos efectos son neutralizados [31].

3.1.1.3 Requisitos no funcionales

Los requisitos no funcionales, RNF, son aquellos requerimientos que hacen referencia a las propiedades del sistema, dejando de lado las funciones específicas que vienen incorporadas en el sistema. A continuación, se van a detallar los requisitos no funcionales del proyecto:

- Cumplimiento de la LOPD (Ley Oficial de Protección de Datos)[32].
- Tiempos de respuestas bajos. Los usuarios no deben de estar esperando a la respuesta del sistema durante un tiempo prolongado. Se establece como máximo de referencia 10 segundos. En ningún caso, ninguna página deberá tardar más de 10 segundos en cargar, debiendo ser el tiempo medio inferior a un segundo.
- Disponibilidad. El servicio debe continuar operativo a pesar de la aparición de algún fallo. Todas las excepciones deberán ser capturadas y tratadas, devolviendo al sistema a un estado consistente y operativo.
- Usabilidad.
    - Un usuario no deberá superar en 3 horas el tiempo de aprendizaje del sistema.
    - Los mensajes de error proporcionados por el sistema han de ser informativos y deben estar orientados al usuario final.
    - El diseño que debe poseer el sistema ha de ser sencillo.
- Fiabilidad. El usuario deberá estar informado en todo momento del estado del sistema.

3.1.1.4 Modelo de dominio

El modelo de dominio nos permite alcanzar principalmente 2 objetivos dentro del contexto del sistema a desarrollar. En primer lugar, identificar y nombrar conceptos importantes: entidades que se deben manipular o sobre los que se registra información y eventos que ocurren en el contexto del sistema. Y, en segundo lugar, identificar y nombrar las relaciones entre estos conceptos.

En el caso de este proyecto, puesto que no ha habido modificaciones en el diseño en el último año, el modelo de dominio se mantiene invariante con respecto a la versión anterior de ASys. El modelo de dominio de ASys se presenta en la Figura 6.

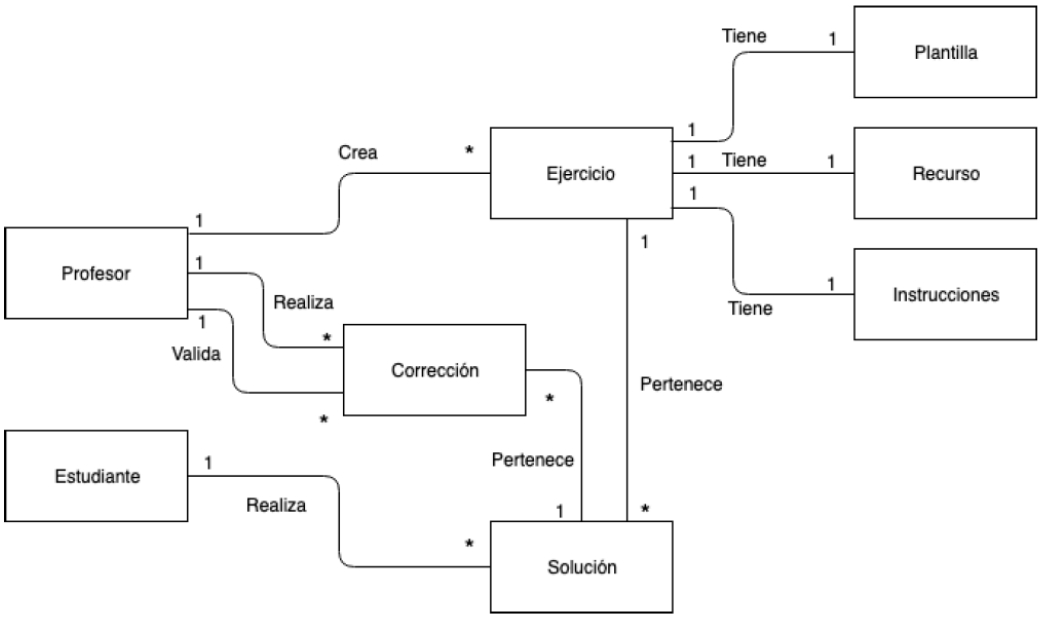

Figura 6: Modelo de dominio del sistema ASys

3.1.1.5 Modelo de contexto

En el modelo de contexto se describen aquellos procesos (actividades o tareas realizadas ya sean realizadas de forma manual o automatizada) que proveen valor a un actor. Sin embargo, no muestra secuencia de actividades, solo las actividades de alto nivel y su valor aportado a los actores. Al igual que en el apartado anterior, la Figura 7 muestra el modelo de contexto del sistema ASys, por permanecer invariable en el proyecto actual.

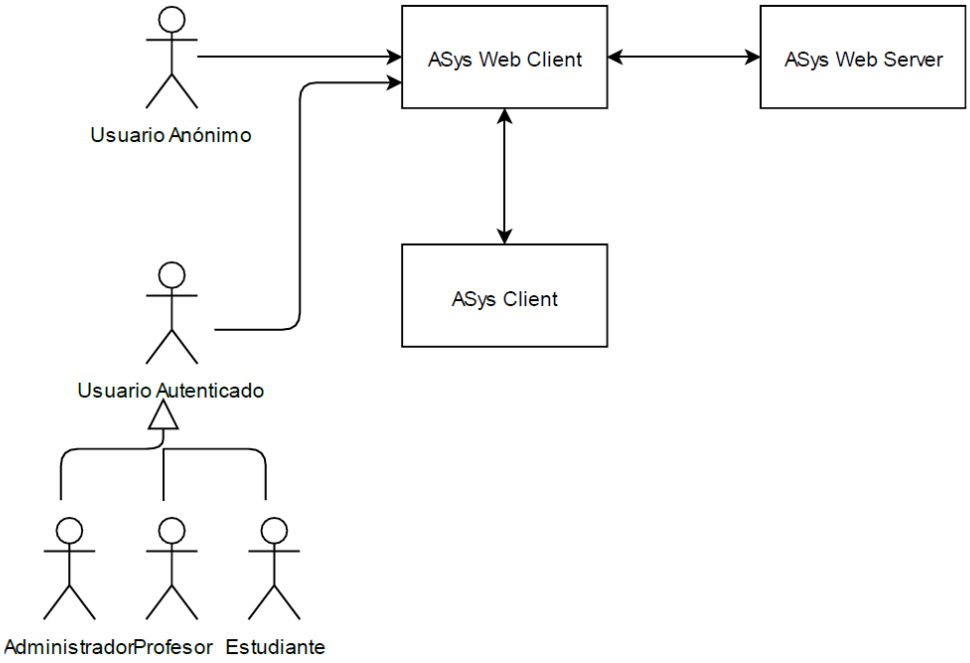

Figura 7: Modelo de contexto del sistema ASys

3.1.2 Casos de uso

Se utilizan para detallar el comportamiento que se considera deseado de un sistema. Su especificación define el conjunto de acciones que se han de llevar a cabo en el sistema para producir el resultado esperado para un actor, pero no describe cómo se llevan a cabo dichas acciones para obtener el resultado deseado. Además, ayuda a validar y verificar el sistema durante el desarrollo.

3.1.2.1 Gestión de ejercicios

En la Figura 8 se muestra un diagrama de casos de uso del subsistema de gestión de ejercicios.

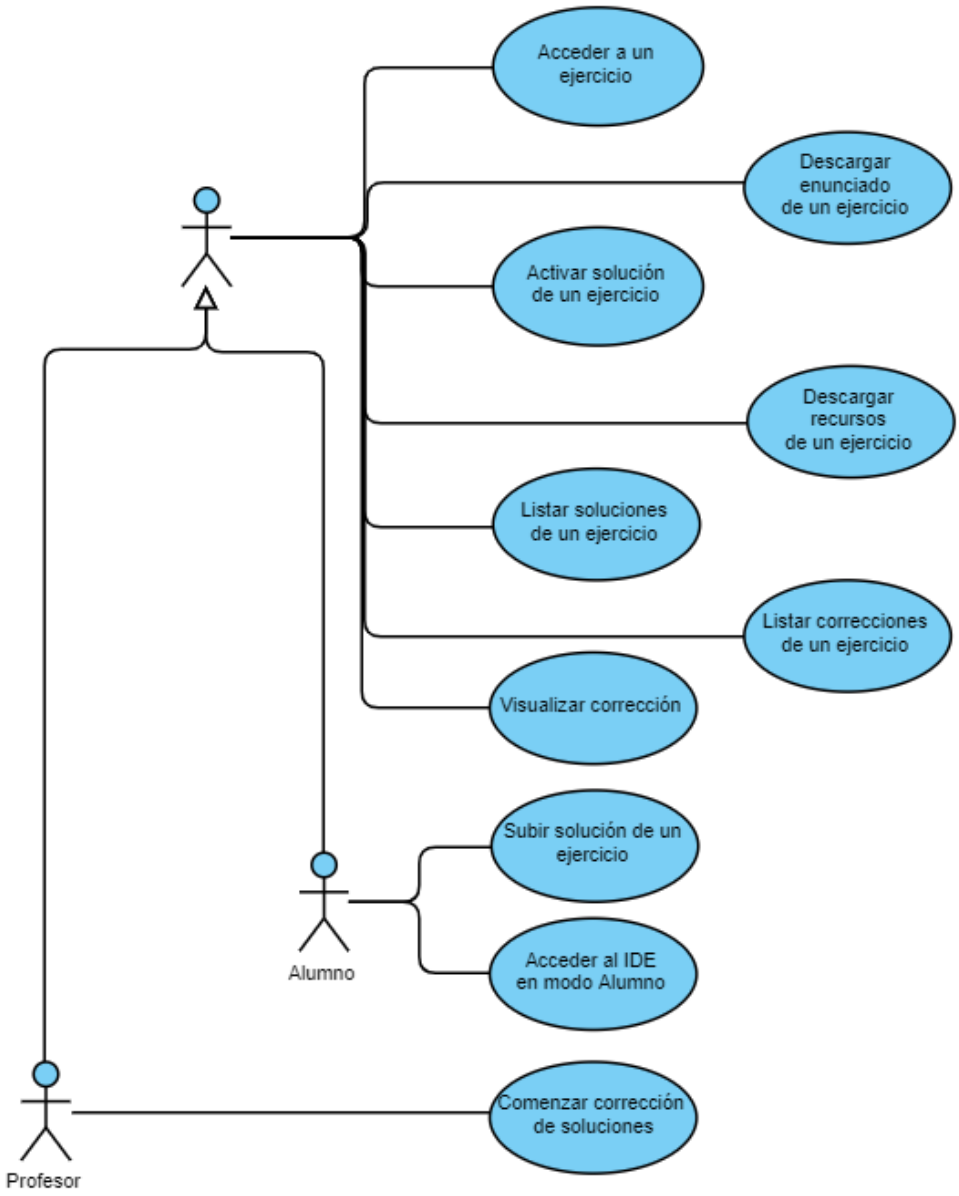

Figura 8: Casos de uso: subsistema de gestión de ejercicios

C.U.1. Acceder a un ejercicio

Descripción	Se accederá a la página del ejercicio, donde se podrán observar: los botones de acción y las soluciones—y correcciones referentes

	a estas si las hubiere—relacionadas al tipo de usuario autenticado.
Actor	Usuario autenticado.
Precondición	Estar situado en la página que se corresponde a la vista de los detalles del ejercicio.
Postcondición	Se visualiza la vista del ejercicio.

C.U.2. Descargar enunciado de un ejercicio

Descripción	El usuario antes de realizar un ejercicio debe poder leer su enunciado.
Actor	Usuario autenticado.
Precondición	Pulsar sobre el botón correspondiente a la función de descargar el enunciado.
Postcondición	Se descarga el enunciado.

C.U.3. Activar solución

Descripción	El usuario selecciona la solución y se debe poder visualizar dicha solución como marcada.
Actor	Usuario autenticado.
Precondición	Pulsar la solución desde el listado de soluciones.
Postcondición	Se activa la solución seleccionada.

C.U.4. Descargar recursos de un ejercicio

Descripción	El usuario antes de realizar un ejercicio debe tener acceso a los recursos que le suministra el creador del ejercicio para su resolución.
Actor	Usuario autenticado.
Precondición	Pulsar sobre el botón correspondiente a la función de descargar los recursos.
Postcondición	Se descargan los recursos.

C.U.5. Listar soluciones de un ejercicio

Descripción	El usuario debe poder visualizar en todo momento las soluciones que previamente han sido subidas en un listado.
Actor	Usuario autenticado.
Precondición	Tener mínimo una solución previamente subida.
Postcondición	Mostrar un listado en base al histórico.

C.U.6. Listar correcciones de un ejercicio

Descripción	El usuario debe poder visualizar en todo momento las correcciones realizadas respecto a la solución activa.
Actor	Usuario autenticado.
Precondición	Realizar previamente una corrección.
Postcondición	Mostrar un listado en base al histórico.

C.U.7. Visualizar corrección

Descripción	El usuario debe poder visualizar la corrección seleccionada.
Actor	Usuario autenticado.
Precondición	Tener una corrección previa y pulsar sobre ella en la lista de correcciones.

Postcondición	Se accede a la página correspondiente para visualizar la corrección.

C.U.8. Subir solución de un ejercicio

Descripción	El usuario desde el IDE debe poder subir la solución.
Actor	Alumno.
Precondición	Pulsar sobre el botón correspondiente a la función de subir la solución.
Postcondición	La solución debe aparecer en la lista de soluciones.

C.U.9. Acceder al IDE en modo alumno

Descripción	El usuario debe poder acceder al IDE para solucionar el ejercicio.
Actor	Alumno.
Precondición	Estar situado en la página que se corresponde a la vista de los detalles del ejercicio y posteriormente pulsar sobre el botón correspondiente a la función de acceder al IDE.
Postcondición	Se accede a la página correspondiente al IDE.

C.U.10. Comenzar corrección de soluciones

Descripción	El usuario debe poder acceder a la vista de corrección de soluciones.
Actor	Profesor.
Precondición	Estar situado en la página que se corresponde a la vista de los detalles del ejercicio y posteriormente pulsar sobre el botón correspondiente a la función de acceder al IDE.
Postcondición	Se accede a la página correspondiente al IDE.

3.1.2.2 Sistema de corrección

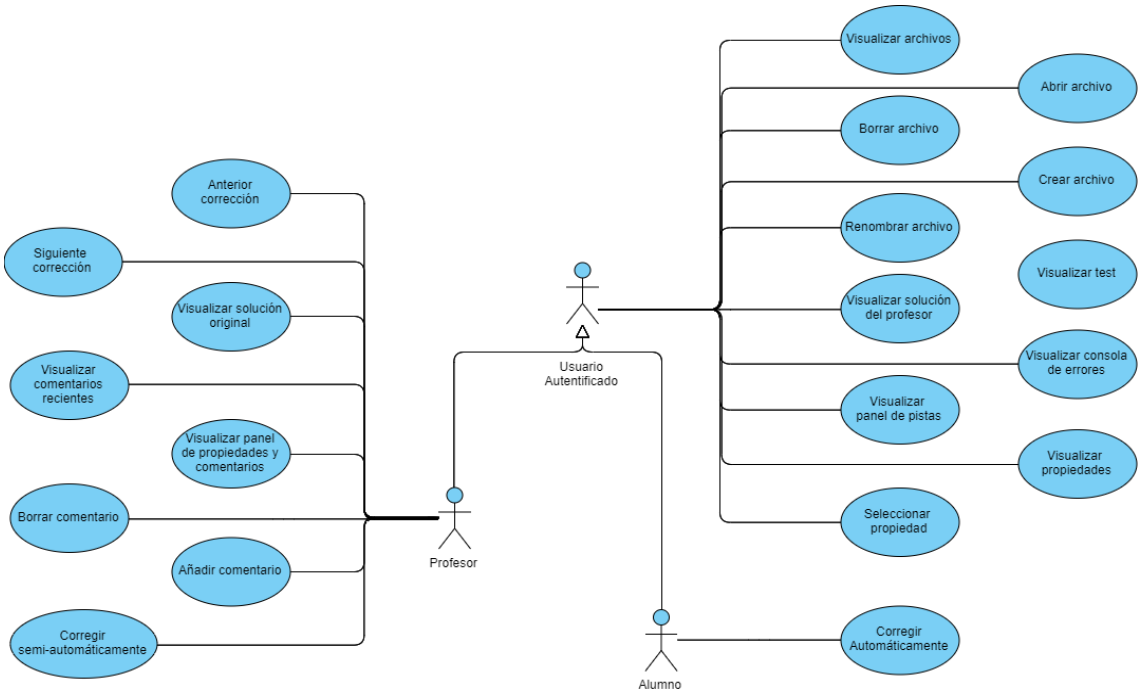

Figura 9: Casos de uso: sistema de corrección

C.U.11 Visualizar archivos

Descripción	El usuario debe poder visualizar los archivos referentes al zip de la solución.
Actor	Usuario autentificado.
Precondición	El usuario debe pulsar sobre el botón correspondiente a la función de visualizar los archivos.
Postcondición	Mostrar los archivos.

C.U.12 Abrir archivo

Descripción	El usuario debe poder visualizar el contenido de un archivo
Actor	Usuario autentificado.
Precondición	El usuario debe pulsar sobre el archivo que desea abrir.
Postcondición	Se muestra en el editor el contenido del archivo.

C.U.13 Borrar archivo

Descripción	El usuario debe poder eliminar un archivo
Actor	Usuario autentificado.
Precondición	El usuario debe pulsar click derecho sobre el archivo y seleccionar la función correspondiente al borrado.
Postcondición	Se elimina de la lista de archivos dicho archivo.

C.U.14 Crear archivo

Descripción	El usuario debe poder crear un archivo
Actor	Usuario autentificado.
Precondición	El usuario debe pulsar click derecho sobre el archivo y seleccionar la función correspondiente a la creación de un archivo.
Postcondición	Se muestra el nuevo archivo en la lista de archivos.

C.U.15 Renombrar archivo

Descripción	El usuario debe poder cambiar el nombre de un archivo.
Actor	Usuario autentificado.
Precondición	El usuario debe pulsar click derecho sobre el archivo y seleccionar la función correspondiente al cambio de nombre.
Postcondición	Se muestra el archivo con el nuevo nombre en la lista de archivos.

C.U.15 Visualizar test

Descripción	El usuario debe poder ver el test asociado al archivo que se encuentre abierto.
Actor	Usuario autentificado.
Precondición	El usuario debe pulsar sobre el botón correspondiente a la función de visualizar el test.
Postcondición	Se muestra el contenido del test asociado al archivo.

C.U.16 Visualizar solución del profesor

Descripción	El usuario debe ver la solución correcta asociada a cada archivo.
Actor	Usuario autentificado.
Precondición	El usuario debe pulsar sobre el botón correspondiente a la función de visualizar la solución del profesor.
Postcondición	Se muestra el contenido de la solución del profesor asociado al

	archivo.

C.U.17 Visualizar consola de errores

Descripción	El usuario debe poder ver la consola de errores.
Actor	Usuario autentificado.
Precondición	El usuario debe pulsar sobre el botón correspondiente a la función de visualizar la consola de errores.
Postcondición	Se muestra la consola de errores.

C.U.18 Visualizar panel de pistas

Descripción	El usuario debe ver la pista asociada a la propiedad que falla.
Actor	Usuario autentificado.
Precondición	El usuario debe pulsar sobre el botón correspondiente a la función de visualizar el panel de pistas y debe de estar seleccionada una propiedad que falle.
Postcondición	Se muestra la pista en el panel de pistas.

C.U.19 Visualizar propiedades

Descripción	El usuario debe ver el estado de las propiedades del ejercicio.
Actor	Usuario autentificado.
Precondición	El usuario debe pulsar sobre el botón correspondiente a la función de visualizar las propiedades.
Postcondición	Se muestra estado de las propiedades.

C.U.20 Seleccionar propiedad

Descripción	El usuario debe ha de tener seleccionada una propiedad en todo momento.
Actor	Usuario autentificado.
Precondición	Se ha de seleccionar una propiedad del panel de propiedades haciendo click en ella.
Postcondición	Se abre el archivo asociado a dicha propiedad y se muestra en el editor.

C.U.21 Anterior corrección

Descripción	El usuario debe poder volver a la corrección anterior.
Actor	Profesor.
Precondición	El usuario debe pulsar sobre el botón correspondiente a la función volver a la anterior corrección.
Postcondición	Se muestra en la misma vista la anterior corrección.

C.U.22 Siguiente corrección

Descripción	El usuario debe poder pasar a la siguiente corrección.
Actor	Profesor.
Precondición	El usuario debe pulsar sobre el botón correspondiente a la función de pasar a la siguiente corrección.
Postcondición	Se muestra en la misma vista la siguiente corrección.

C.U.23 Visualizar solución original

Descripción	El usuario debe poder visualizar el código original de la solución.

Actor	Profesor.
Precondición	El usuario debe pulsar sobre el botón correspondiente a la función de visualizar la solución original.
Postcondición	Se muestra el contenido de la solución original asociada al archivo.

C.U.24 Visualizar comentarios recientes

Descripción	El usuario debe poder visualizar los comentarios recientes previamente realizados sobre una propiedad.
Actor	Profesor.
Precondición	Hay al menos un comentario realizado sobre una propiedad y se encuentra abierto el panel de comentarios y propiedades.
Postcondición	Se muestra una lista con los comentarios recientes.

C.U.25 Visualizar panel de comentarios y propiedades

Descripción	El usuario debe poder visualizar el panel de comentarios y propiedades para poder realizar posteriormente cualquiera de las funcionalidades de dicho panel.
Actor	Profesor.
Precondición	El usuario debe pulsar sobre el botón correspondiente a la función de visualizar el panel de comentarios y propiedades.
Postcondición	Se muestra el panel de comentarios y propiedades.

C.U.26 Borrar comentario.

Descripción	El usuario debe poder eliminar un comentario.
Actor	Profesor.
Precondición	Se encuentra abierto el panel de comentarios y propiedades. Y en la lista de comentarios de la propiedad hay mínimo un comentario. Se pulsa el botón correspondiente al borrado del comentario.
Postcondición	Se elimina el comentario de la lista de comentarios.

C.U.27 Añadir comentario.

Descripción	El usuario debe poder añadir un comentario.
Actor	Profesor.
Precondición	Se encuentra abierto el panel de comentarios y propiedades. Y se encuentra una propiedad seleccionada. Se pulsa el botón correspondiente con la funcionalidad de añadir el comentario.
Postcondición	Se añade el comentario a la lista de comentarios.

C.U.28 Corregir semi-automáticamente.

Descripción	El usuario debe poder subir la corrección semi-automática.
Actor	Profesor
Precondición	Pulsar sobre el botón correspondiente a la corrección de la solución.
Postcondición	La corrección semi-automática aparece en la lista de correcciones corregida por el “nombre_profesor”

C.U.29 Corregir automáticamente.

Descripción	El usuario debe poder subir la corrección automática.

Actor	Alumno
Precondición	Pulsar sobre el botón correspondiente a la corrección de la solución.
Postcondición	La corrección automática aparece en la lista de correcciones corregida por "ASys".

3.2 Análisis de riesgos

El propósito de este apartado es plasmar en la memoria el estudio de las causas de las posibles amenazas, así como las repercusiones que éstas puedan producir en el proyecto. Para llevar a cabo dicho estudio, se ha seguido el siguiente procedimiento:

1. Identificar los riesgos potenciales.
2. Clasificar los riesgos en: de aceptación, de satisfacción y tecnológicos o de integración.
3. Definir el impacto que puede causar cada uno de los riesgos.
4. Definir las medidas a tomar para reducir el impacto producido.

3.2.1 Riesgos de aceptación

Título	El diseño de la interfaz del sistema de corrección no cumple con lo esperado con el *stakeholder*.
Impacto en el proyecto	Incremento en el tiempo de implementación. (Retrabajo)
Medidas reductoras de riesgo	Se validará el diseño de la interfaz periódicamente para asegurar que se cumple con las expectativas que tiene el cliente.

Título	El rendimiento del sistema de corrección es más tardío del esperado por el *stakeholder*.
Impacto en el proyecto	Buscar una alternativa para mejor la eficiencia del sistema. Incremento en el tiempo de implementación. (Retrabajo)
Medidas reductoras de riesgo	Se irán realizando pruebas periódicamente donde se evaluará el rendimiento del sistema, que deberá ser óptimo en todo momento. En el caso de detectar una bajada del rendimiento se analizará la situación.

3.2.2 Riesgos de satisfacción

Título	El usuario no está conforme con la fiabilidad del sistema
Impacto en el proyecto	Incremento en el tiempo de implementación. (Retrabajo)
Medidas reductoras de riesgo	Periódicamente se realizarán pruebas donde el cliente podrá comprobar la fiabilidad del sistema progresivamente durante su implementación.

Título	La interfaz del sistema es poco intuitiva y tiene un elevado coste en el tiempo de aprendizaje.
Impacto en el	Incremento en el tiempo de implementación. (Retrabajo)

proyecto	
Medidas reductoras de riesgo	Se realizarán *mock-ups* antes de comenzar con el diseño de la interfaz, para asegurar que el usuario final tiene un buen *feed-back* desde el primer momento.

3.2.3 Riesgos tecnológicos

En cuanto a mi parte del proyecto, la tecnología utilizada ya había sido elegida previamente y no participé en dicho análisis. Sin embargo, los riesgos del proyecto de ASys siguen siendo exactamente los mismos:

Título	Las herramientas elegidas dejan de tener soporte
Impacto en el proyecto	Las herramientas no recibirán actualizaciones. Posible aparición de errores.
Medidas reductoras de riesgo	La elección de las herramientas se realizará tras un estudio de las diversas herramientas del mercado, escogiendo aquellas que tengan mejor *feed-back* en cuanto a fiabilidad a largo plazo. Resolución de errores manual.

Título	El sistema carece de escalabilidad
Impacto en el proyecto	Impedirá la ampliación del proyecto en aquellas mejoras que requieran un aumento del uso de recursos del sistema.
Medidas reductoras de riesgo	El diseño y desarrollo del sistema se realizará teniendo en cuenta que el sistema debe ser escalable.

3.3 Identificación y análisis de soluciones posibles

En este apartado se van a explicar las tecnologías empleadas en la creación del IDE para el desarrollo y la corrección automatizada de ejercicios de programación. Y se explicará, tanto por qué se eligió dicha tecnología como si cumple con las expectativas que se tenían cuando se realizó el estudio.

Cabe destacar que el IDE creado se integra con la aplicación web, sin embargo, también entra en contacto con el servidor y tiene conexión directa con éste. Por lo tanto, se analizarán tanto la tecnología utilizada para la aplicación web como para el servidor.

3.3.1 Análisis de tecnologías de la aplicación web

En el presente proyecto se quería utilizar una aplicación de SPA (*Single Page Application*) con la finalidad de evitar un alto acoplamiento entre el servidor y la aplicación web. Así bien, esto se explica en el TFG de Armando Maya [33]: permite evitar: "*Problemas de disponibilidad en el despliegue*" debido a que "*al tener la aplicación web y el servidor en el mismo proyecto, cuando se realiza un despliegue por haber añadido cambios a uno de los dos proyectos, ambos dejan de funcionar*" y "*el uso de sesiones y páginas generadas en el servidor*", *esto último genera una dificultad añadida a la hora de depurar pues, como decía Armando Maya: "resulta muy complicado seguir la traza de ejecución ya que en cualquier punto del sistema se pueden obtener, añadir y modificar las variables utilizadas para cierta funcionalidad".*

En resumen, la aplicación web será independiente del servidor; y, por lo tanto, podrá funcionar autónomamente sin depender del servidor para la renderización de las páginas.

Para la elección del *framework* utilizado para desarrollar la aplicación web, se realizó un estudio entre los más populares o empleados por grandes compañías. Así bien se realizó un estudio de los siguientes *frameworks:* Angular, React.js, Polymer y Vue.js. Al terminar el estudio se propuso el uso de Vue.js principalmente por su gran rendimiento, documentación y aporte de la comunidad.

3.3.2 Valoración del *framework* elegido para la aplicación web

Personalmente, al ser un usuario que nunca había tratado con Vue.js, y debido a esto ha debido de iniciarse en la tecnología, voy a comparar las expectativas que se tenían durante la elección del *framework* con el *feed-back* que he tenido durante el desarrollo del proyecto.

En primer lugar, en el TFG de Armando Maya se expone [33]: "es una apuesta por su reciente creación, Vue.js se está convirtiendo en una tendencia para los nuevos desarrollos, debido a su orientación *open source* la cual garantiza un soporte infinito por parte de la comunidad. Además, su rendimiento es tremendamente bueno en comparación con sus competidores y el gran esfuerzo invertido en la documentación es un factor que se ha tenido muy en cuenta." Es decir, se eligió Vue.js por su gran rendimiento, documentación y aporte de la comunidad.

Como usuario estoy gratamente sorprendido por el rendimiento del *framework* elegido, sin embargo, aunque la documentación oficial es bastante buena no puedo decir lo mismo del aporte comunitario, a la hora de resolver errores que me han ido surgiendo durante el desarrollo me ha sido, en muchos casos, bastante complicado encontrar la solución. Pienso que es debido a su reciente creación. A pesar de todo, como desarrollador de ASys, estoy contento con la elección y creo que fue una buena apuesta.

3.3.3 Análisis de tecnologías del servidor

En el caso del servidor de ASys se tomaron en cuenta diversos factores para la elección de la tecnología. En primer lugar, se quería **conservar el lenguaje** empleado en su primera versión, Java, por dos motivos:

(1) Evitar costosos procesos de reimplementación en los componentes y sistemas asociados.

(2) El desarrollador emplearía el mismo lenguaje para todos los proyectos utilizados.

En segundo lugar, se buscaba un *framework* con un **amplio apoyo de la comunidad**, que proporcionara muchos recursos y ayuda a la hora del desarrollo.

Por último, se apostó por un *framework* **completo** frente a uno ligero, pues ofrecen una gran cantidad de librerías que simplifican el desarrollo de todos los servicios que van a ser necesarios en el servidor.

Valorando todos estos factores se optó por utilizar Spring *framework*, además de, como añadía Armando Maya en su TFG [33]: "es el *framework* más usado en Java, sumado a la capacidad para adaptarse al enfoque REST". Esto último nos permite separar el cliente del servidor.

3.3.5 Valoración del *framework* elegido para el servidor

Personalmente, en mi proyecto he tenido que interactuar muy poco con el servidor en Spring, en concreto para cambiar la estructura de la BD, añadir ciertas consultas que eran necesarias y para implementar algunos métodos adicionales en la capa de lógica. En ningún caso he tenido problemas de desarrollo con el entorno, solo recuerdo uno referente a la paginación a la hora de hacer consultas en Spring, pero gracias al soporte de la comunidad en seguida pude resolverlo. Desde mi perspectiva, Spring fue una excelente decisión resultado de un buen estudio.

4. DISEÑO DE LA SOLUCIÓN

En este apartado se va a explicar cómo se ha abordado el diseño de la solución del proyecto, desde la identificación de requisitos hasta su validación final. Cabe añadir que este es un proceso iterativo. Por lo tanto, se ha pasado por la especificación y validación de requisitos en numerosas ocasiones. Debido a la gran amplitud que tendría en esta memoria reflejar cada una de las veces que se ha pasado por cada una de las fases solamente se redactará una vez cada etapa, resumiendo en cada apartado los cambios más importantes que han sido identificados.

4.1 Identificación de requisitos

Como técnica de elicitación para la identificación de requisitos se ha optado por la técnica del prototipado. Una técnica muy utilizada para proveer un contexto a los usuarios y que puedan entender mejor el sistema a desarrollar, además nos permite visualizar una versión preliminar del sistema, lo que ayudará a la hora de anticipar errores de diseño.

En este proyecto se ha optado por los *mock-ups* como forma de prototipado [34, 35].

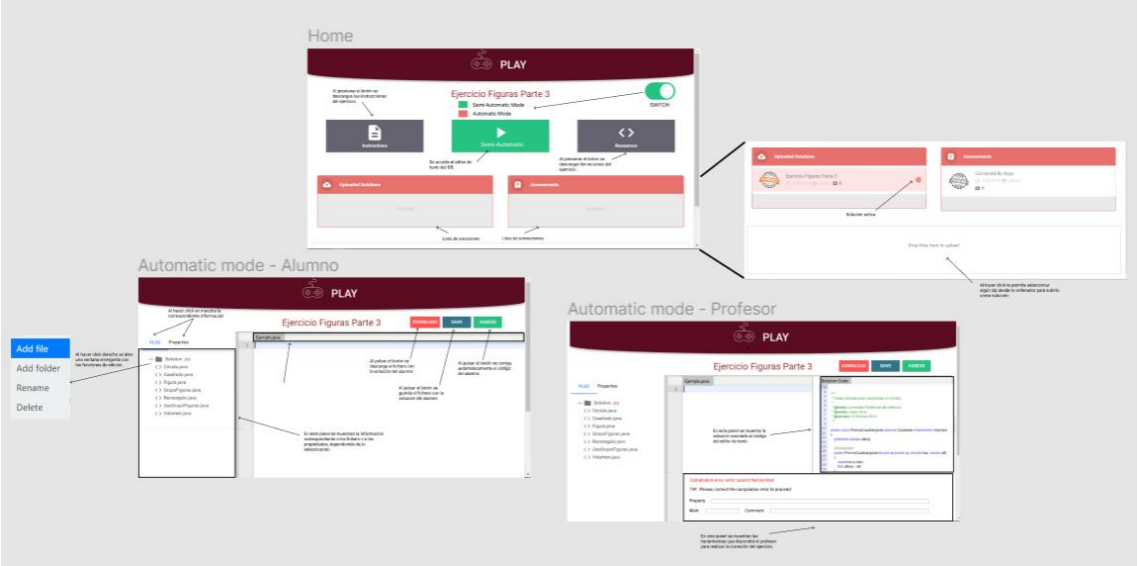

Figura 10: Mock-ups, visión general del subsistema

4.2 Validación de requisitos

Como se ha comentado brevemente en el apartado anterior, el usuario final, gracias a la visualización preliminar del sistema, puede detectar errores antes de la implementación de estos requisitos.

Así bien, algunos de estos errores fueron:

1. El modo de corrección (Automático/Semi-automático). El error consistía en que el desarrollador pensaba que el usuario debía ser el responsable de la elección del modo de corrección, sin embargo, esto es irrelevante para él. La corrección

de dicho error fue que internamente el sistema eligiera el modo más conveniente, haciendo la acción de elección transparente para el usuario.

2. Los colores y la posición de los botones de acción ("download", "save" y "assess"). Este era un error de diseño, pues los botones eran de un color muy llamativo y no concordaban con el estilo de la página web. Así bien, dicha corrección se realizó haciendo uso de colores pastel y cambiando el posicionamiento de los botones.
3. El botón "save" era innecesario pues cada vez que se ejecuta "assess" ya se guardaba el código de la solución. Esto es debido a que el botón "assess" tiene la funcionabilidad de corregir la solución, es decir, el código que se encuentra presente en la página web queda persistido en la base de datos y, por ende, al corregirlo ya queda guardado por defecto.
4. El alumno también debía de ser capaz de visualizar la consola y las pistas. En un primer lugar, el desarrollador pensó que solamente el usuario "profesor" debía poder visualizar la consola y las pistas. Esto se solucionó habilitando estas secciones al usuario "alumno", de forma que pudiera el propio alumno ir corrigiendo semi-automáticamente el código.
5. El profesor necesitaba algún medio para visualizar los tests y la solución original del alumno. El desarrollador a la hora de realizar el diseño de la web no tuvo en cuenta la visualización de los tests asociados al ejercicio, ni tampoco, que la corrección del ejercicio se llevara a cabo por partes (se inicia la corrección, pero no se acaba). La corrección de este error consistió en la creación de pestañas que permitieran navegar entre: la solución original del alumno, la solución del profesor y los tests asociados a la propiedad activada por el profesor.

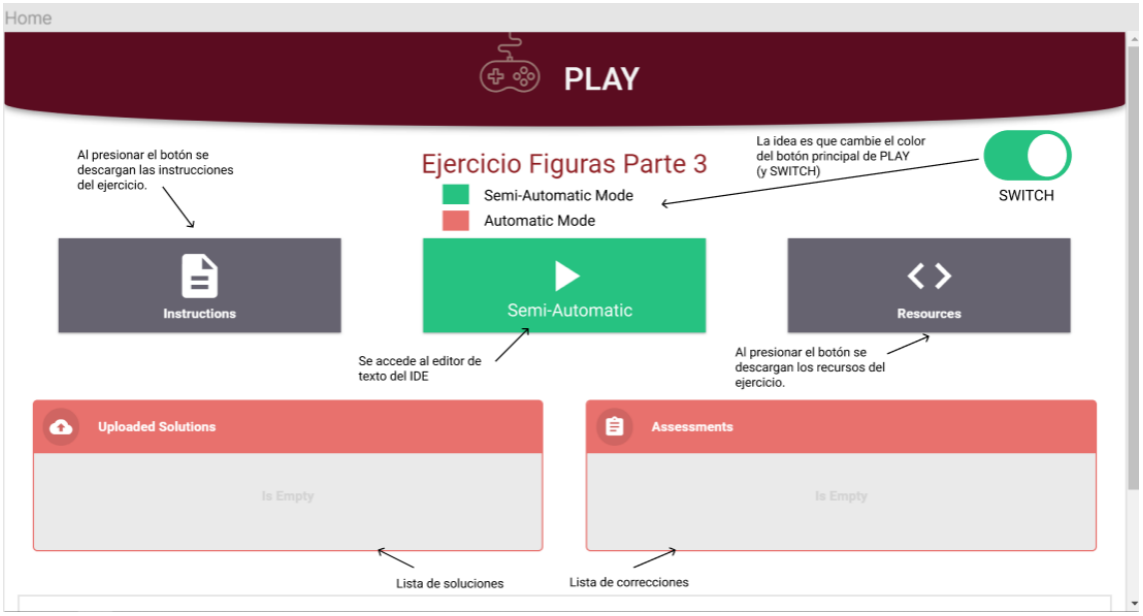

Figura 11: Mock-up, página principal del ejercicio

En un primer lugar, se optó por poder seleccionar el modo de corrección antes de acceder al editor de texto del IDE. Posteriormente, gracias a esta técnica de prototipado, el usuario final al ver esta versión preliminar desechó la idea, debido al siguiente

requisito: “siempre se ha de corregir en modo automático y solamente cuándo el usuario lo desee, podrá modificar dicha corrección, y, en ese momento, será semi-automática”. En este caso, se ha ahorrado bastante tiempo de retrabajo en el proyecto y ha ayudado al desarrollador a entender mucho mejor la perspectiva del usuario final.

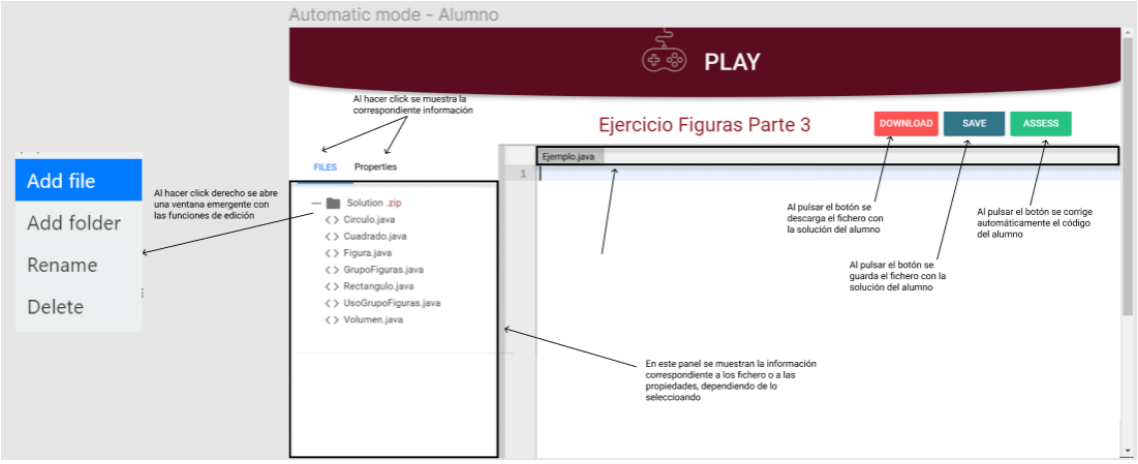

Figura 12: Mock-up, editor de texto – alumno

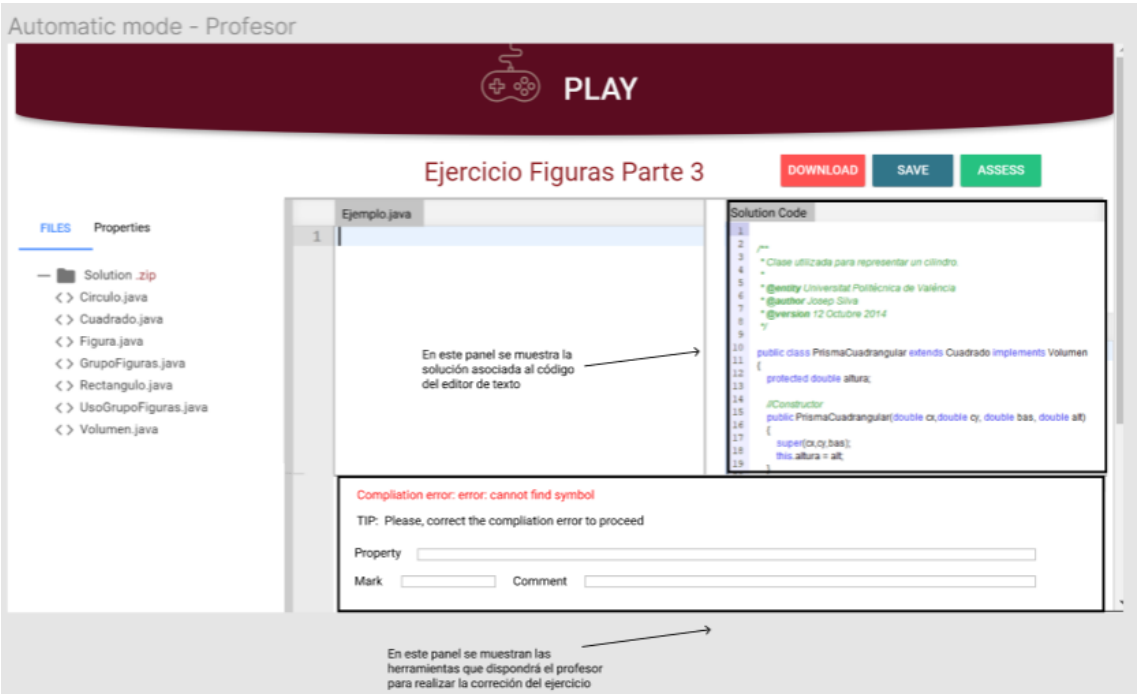

Figura 13: Mock-up, editor de texto – profesor

En este caso, tenemos dos posibles vistas del editor de texto del IDE: en modo profesor y en modo alumno. El modo en el que se accede a la vista depende del tipo de usuario que se encuentre registrado a la hora de pulsar el botón de acceso al editor de texto.

El modo alumno es una versión resumida del modo profesor, simplemente posee un conjunto de funcionalidades reducida pues el profesor tiene más competencias que el alumno.

5. DESARROLLO DE LA SOLUCIÓN

En esta sección se va a explicar cómo se ha efectuado la implementación de la solución, así como ciertas partes del proyecto base de ASys que han tenido que ser modificadas.

5.1 Desarrollo del servidor de ASys

Respecto a la base de datos de ASys el diseño se ha mantenido prácticamente intacto, el motivo de esto ha sido probablemente gracias a un buen estudio del modelo de dominio. Así bien el modelo relacional ha tenido una breve modificación para ajustarse a los nuevos requerimientos del sistema.

En la Figura 14 se va a mostrar un fragmento del modelo relacional diseñado para este proyecto, tras las ampliaciones marcadas en rojo.

En este caso se ha añadido una nueva tabla llamada "property_test". Esta tabla era necesaria puesto que cada una de las propiedades de un "*assessment"* debía de tener asociado un test, para así poder ayudar al profesor o corrector a identificar y solventar el error lo más rápidamente posible.

Una demostración visual de su necesidad se puede encontrar en la Figura 15.

Se puede observar cómo la propiedad "GrupoFigurasVolumenMethod" está fallando en un test (está de color azul), así bien, si queremos saber en que test está fallando simplemente bastará con abrir la pestaña "Test" en el editor de texto derecho para poder visualizarlo.

Por otro lado, también se ha añadido un atributo llamado "send_teacher" a la tabla "student_solution". Este atributo es de tipo booleano y nos permite filtrar las soluciones que ha de corregir el profesor. Anteriormente, antes de la existencia de este atributo, todas aquellas soluciones que los alumnos hubieran creado sobre dicho ejercicio eran visibles para el profesor, lo cual podía llegar a ser insostenible y un poco caótico para el profesor. Una forma de evitar este comportamiento es simplemente que el profesor visualice aquellas soluciones que los alumnos quieren que sean corregidas.

Por lo tanto, un alumno podrá enviar cada solución una única vez (Figura 16), y una vez este ejercicio sea enviado se le notificará como tal (Figura 17).

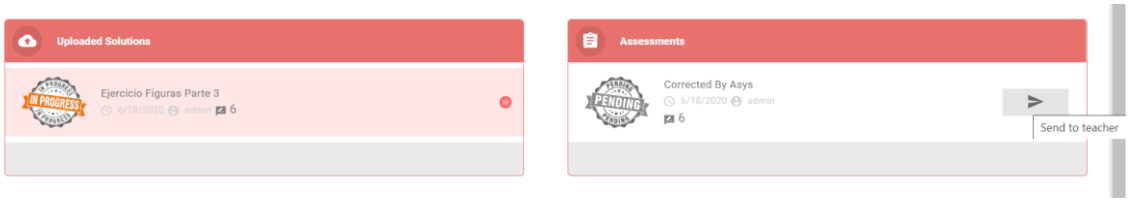

Figura 14: IDE de ASys, botón "send_teacher" habilitado

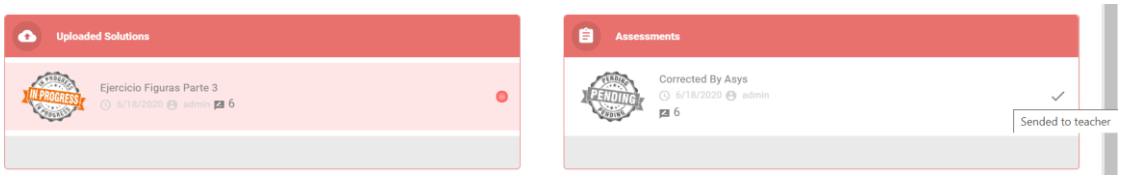

Figura 15: IDE de ASys, botón "send_teacher" deshabilitado

Para persistir estos cambios en el modelo relacional dentro del servidor de Spring ha sido necesario modificar la carpeta "model" del proyecto (Figura 18). Simplemente era necesario añadir una nueva clase.java (Figura 19) (Figura 20) y, dentro de dicha clase, poner la notación *@Entity*, la cual le indica a Spring que esta es una clase que deberá mapearse en la BD. Seguidamente, solo hay que añadir las propiedades. Una propiedad es cada una de las columnas correspondientes a la tabla de la base de datos. Para indicar comportamientos de dichas propiedades en la base de datos como *not null, id o una relación 1-N* Spring utiliza una notación muy intuitiva para declarar dichos comportamientos (Figura 20)

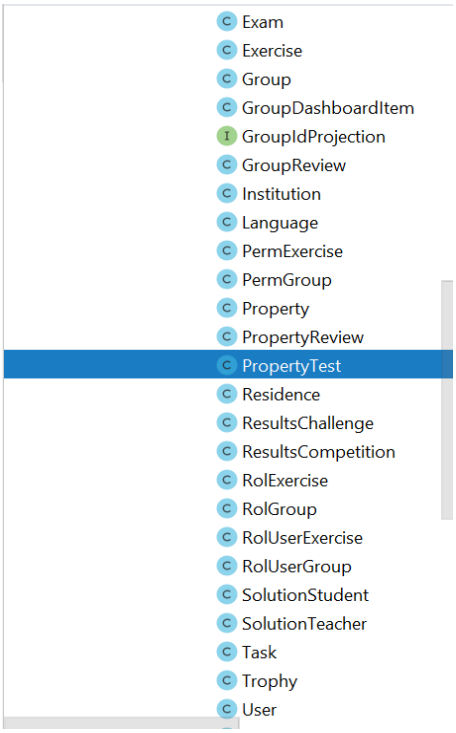

Figura 16: Estructura del servidor de ASys 1

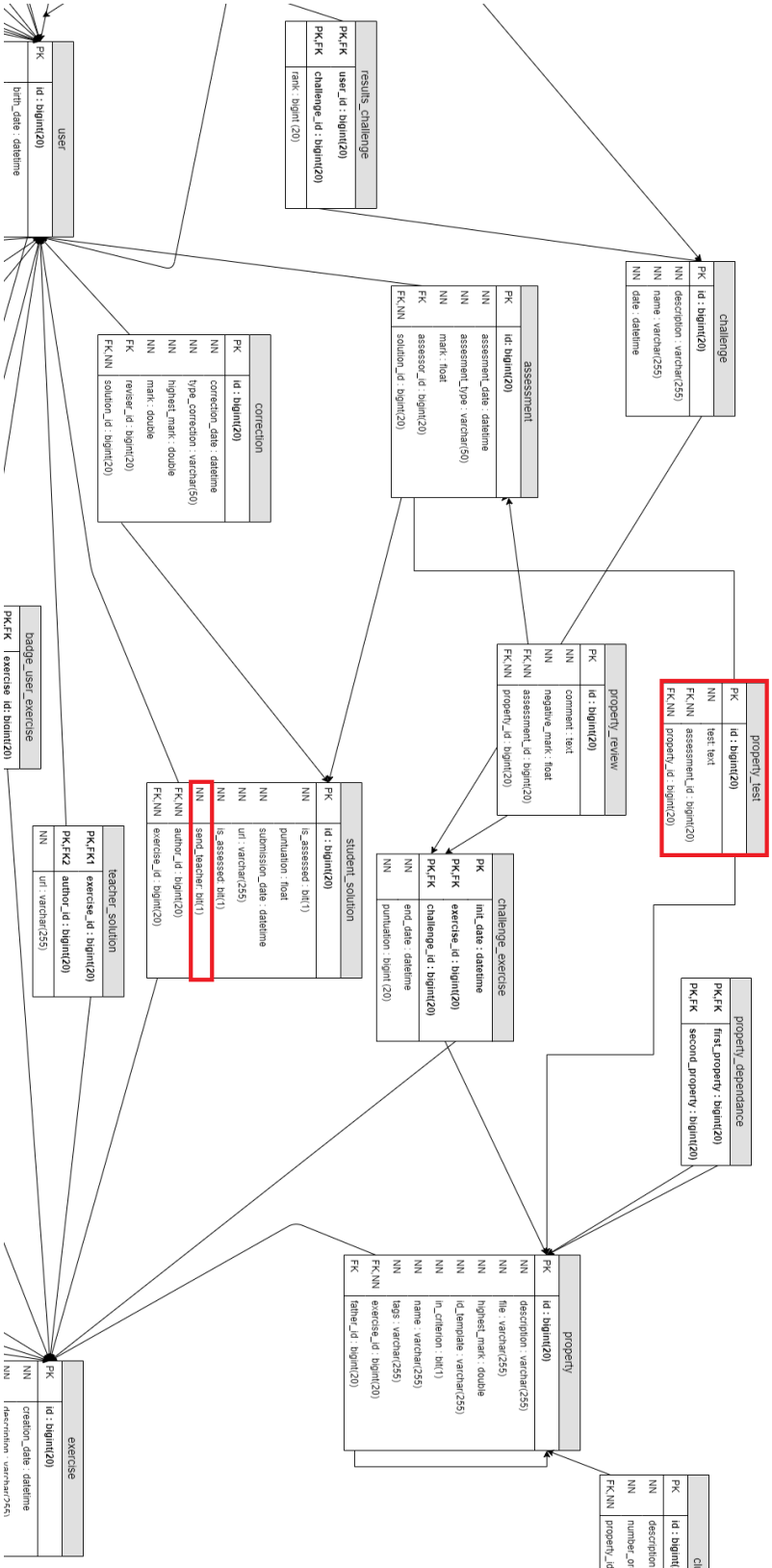

Figura 17: Fragmento del diagrama relacional de ASys

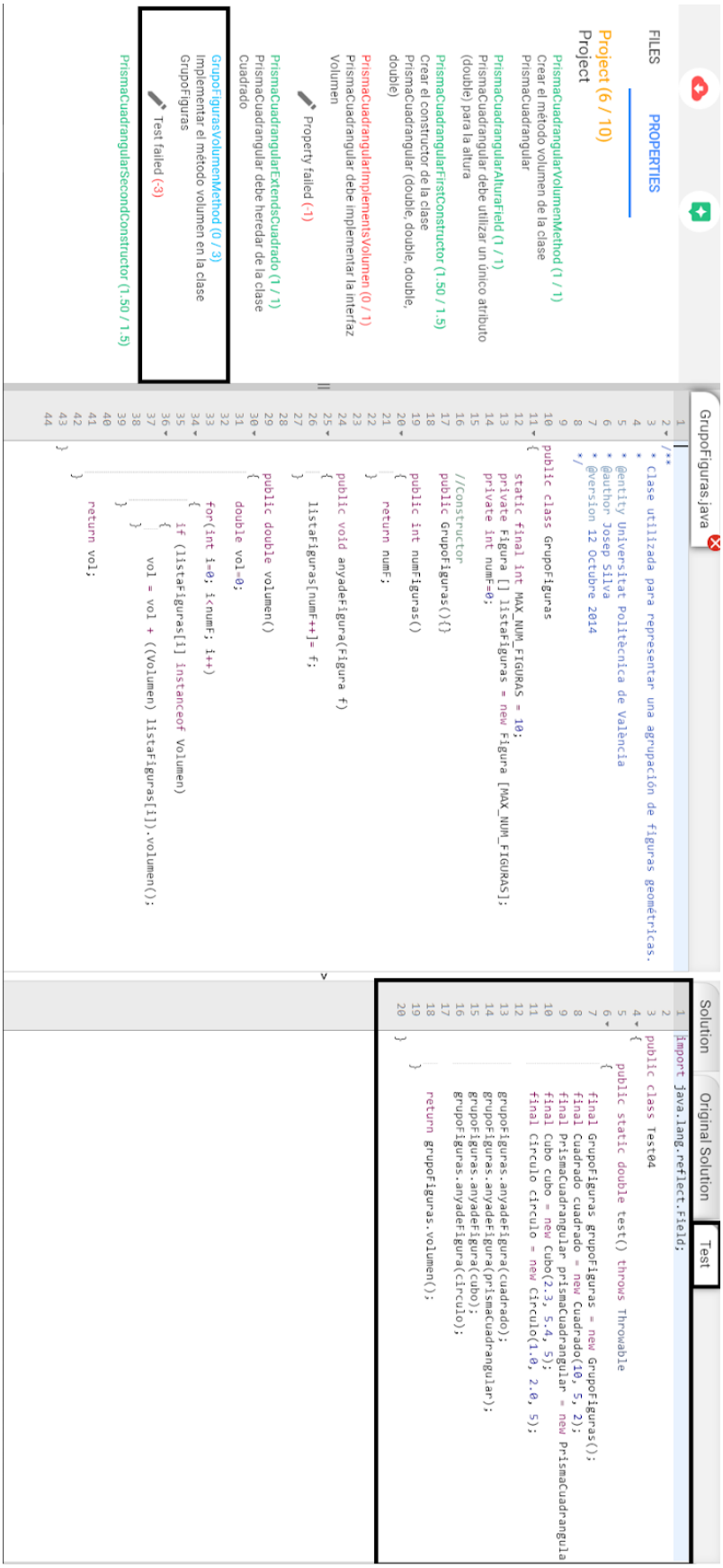

Figura 18: IDE de ASys, perspectiva profesor 1

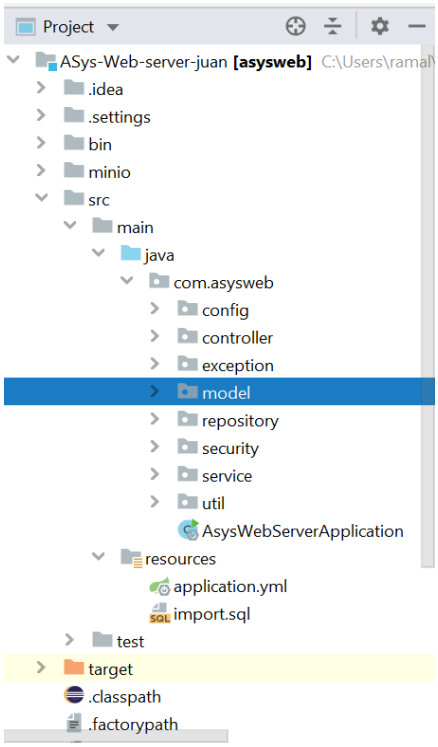

Figura 19: Estructura del servidor de ASys 2

```java
package com.asysweb.model;

import ...

@Entity(name = "PROPERTY_TEST")
public class PropertyTest {

    @Id
    @GeneratedValue
    private Long id;

    @NotNull
    @ManyToOne
    private Assessment assessment;

    @ManyToOne
    @NotNull
    private Property property;

    @NotNull
    private String test;

    public Long getId() { return id; }

```

Figura 20: Estructura del servidor de ASys 3

5.2 Desarrollo de la aplicación web de ASys

Esta sección es la más extensa de este proyecto pues, tanto la interfaz de la página donde reside el IDE como la gran parte de su funcionalidad han sido implementadas en Vue.js.

En primer lugar, vamos a recordar cómo funciona el framework de Vue. Una de las características más poderosas de Vue son los componentes. Un componente te permite extender elementos HTML básicos para encapsular código reutilizable, es decir, son elementos personalizados a los que el compilador de Vue les añade comportamiento.

5.2.1 Estructura de la aplicación web de ASys

Sabiendo esto, vamos a introducir brevemente la estructura del proyecto de la aplicación web de ASys (Figura 21). Principalmente se ha trabajado con los componentes y las vistas, estos dos elementos son los que tienen un mayor impacto en la interfaz y además también contienen, en gran parte, la funcionalidad de esta. Cabe mencionar que tanto la carpeta store como los servicios han tenido un papel importante en el desarrollo. La carpeta store nos permite añadir y obtener datos de forma organizada entre los componentes (tiene la función de ser un almacén de datos global). Los servicios permiten la conexión entre la aplicación web y los otros subsistemas del proyecto.

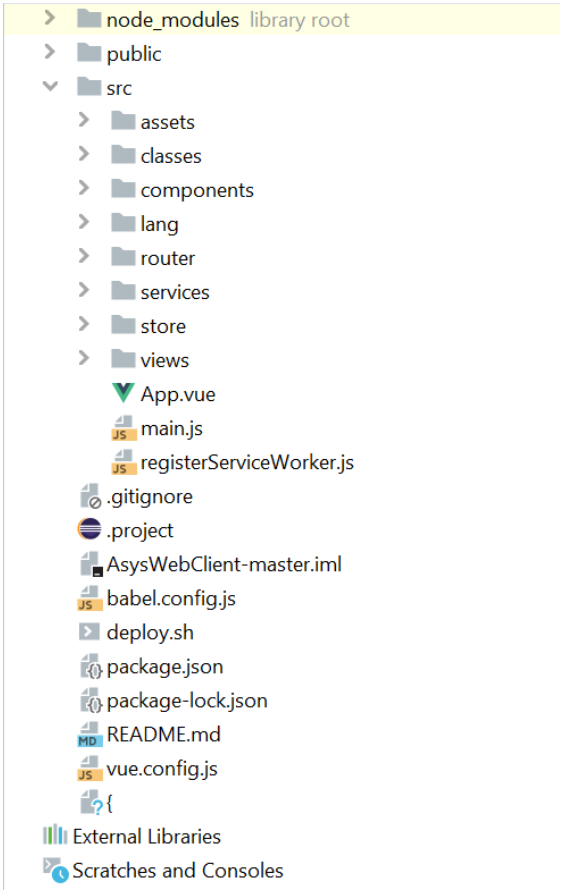

Figura 21: Estructura de carpetas

5.2.2 El IDE en la estructura de la aplicación web

Antes de comenzar a explicar la estructura del IDE cabe mencionar como se accede a este. Cualquier usuario debe de seleccionar un ejercicio y, posteriormente debe tener la posibilidad de, según su función en dicho ejercicio, resolverlo (alumno) o corregirlo (profesor). Esta situación donde el usuario se encuentra dentro de un ejercicio y tiene acceso a su respectiva función es la vista llamada "Information.vue" (Figura 22), si el usuario decide realizar la función entrará en la vista "Code.vue" (Figura 22).

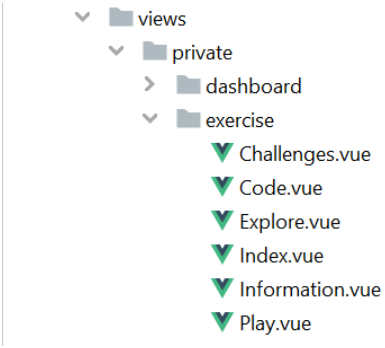

Figura 22: Estructura de la vista "ejercicio"

Es por esto último que el IDE se encuentra respectivamente en la vista Code.vue y dependiendo del rol del usuario registrado se mostrará una interfaz.

Por último hay que mencionar el componente "braceEditor". Este componente contiene prácticamente toda la estructura del IDE, utiliza la librería "brace" (https://www.npmjs.com/package/brace) [36]. Brace es una librería que contiene scripts que nos facilitan el uso de la librería Ace-editor [37]. Brace contiene un script que automáticamente compila y refactoriza las *ace build*[1]*s*, por este motivo para hacerlo funcionar simplemente hay que pasarle un modo y un tema con "setMode" y "setTheme". A continuación se mostrará cómo se ha hecho está implementación el proyecto.

```
<braceEditor ref="braceEditor"
  :fontsize="'12px'"
  :theme="'eclipse'"
  :mode="exerciseState.exercise.language.name.toLowerCase()"
  :codefolding="'markbegin'"
  :softwrap="'off'"
  :selectionstyle="'text'"
  :highlightline="true"
```

Figura 23: Implantación del componente "brace" en ASys

Desde Code.vue llamamos al componente Brace, al cual mediante propiedades le podemos pasar los valores necesarios para configurar el componente, en este caso a

[1] Ace es un editor de texto simple con las funciones más comunes como copiar/pegar y abrir/guardar. Llamamos *Ace-build* al proceso de convertir archivos del código fuente de Ace a *standalone software*, que se puede ejecutar en un ordenador.

parte de pasar el modo y el tema, también estamos pasándole otras propiedades como son el tamaño de la fuente.

Con esto conseguimos que al montarse la instancia del componente braceEditor, ésta disponga de todas las variables necesarias. Véase en la Figura 24 el código que se ejecuta al iniciarse el componente.

```
editor = brace.edit( el: 'vue-brace-editor')
editorTeacher = brace.edit( el: 'vue-brace-editor-teacher')
// Autocompletion
editor.setOptions({
  enableBasicAutocompletion: true,
  enableLiveAutocompletion: true,
  enableSnippets: true
  })

editorTeacher.setOptions({
  enableBasicAutocompletion: true,
  enableLiveAutocompletion: true,
  enableSnippets: true
})

// Editor options
this.setImportedMode()
this.setTheme()

editor.setFontSize(this.fontsize)
editor.setShowFoldWidgets(this.codefolding !== 'manual')
editor.setOption( optionName: 'wrap', this.softwrap)
editor.setOption( optionName: 'selectionStyle', this.selectionstyle)
editor.setHighlightActiveLine(this.highlightline)
editor.$blockScrolling = Infinity
editor.session.setFoldStyle(this.codefolding)

editorTeacher.setFontSize(this.fontsize)
editorTeacher.setShowFoldWidgets(this.codefolding !== 'manual')
editorTeacher.setOption( optionName: 'wrap', this.softwrap)
editorTeacher.setOption( optionName: 'selectionStyle', this.selectionstyle)
editorTeacher.setHighlightActiveLine(this.highlightline)
editorTeacher.$blockScrolling = Infinity
editorTeacher.session.setFoldStyle(this.codefolding)
```

Figura 24: Fragmento del código para montar el componente "brace": brace-options

Como se puede observar, disponemos de dos editores de texto: "editor" y "editorTeacher". Ambos se han configurado de la misma forma; la única diferencia entre ellos será que uno será editable y el otro simplemente se limitará a mostrar información. Gracias a las siguientes líneas lo hacemos editable (Figura 25).

```
// Listener and handlers
this.createHandlers()
editor.on( ev: 'change', this.updateCode)
window.addEventListener( type: 'resize', this.resizeEventHandler)
document.body.onmousedown = function(e) { // Avoid middle mouse button scroll
  if(e.button == 1)
    return false
}
```

Figura 25: Fragmento del código para montar el componente "brace": listeners y handlers

Básicamente emitimos un evento cada vez que el código cambia. Así, gracias a un *listener*, podemos ir refrescando el código de dicho editor.

5.2.3 El IDE perspectiva del profesor

El usuario que ha creado el ejercicio tiene acceso a éste como "profesor", es decir, puede ver todas las resoluciones de los ejercicios que los alumnos le han enviado. El IDE, como se ha mencionado anteriormente, cuenta con dos editores de texto: en el editor situado en la parte izquierda se muestra el código del alumno, que es a su vez editable por el profesor; por otro lado, en el editor de la parte derecha se encuentra un código inmutable, que sirve de mera orientación al usuario (Figura 26).

El profesor, en el editor derecho, puede visualizar 3 diferentes elementos:

1. La solución del ejercicio. Por lo tanto, al corregir el código del alumno siempre tendrá de referencia el código que funciona correctamente y que cumple con lo especificado en el ejercicio.
2. La solución original del alumno. Es posible que el profesor haya iniciado una corrección del código del alumno y haya tenido que parar por cualquier motivo, entonces cuando vuelva a continuar con la corrección es muy probable que no recuerde el estado inicial de la solución del alumno para poder asignarle una nota. O bien, que simplemente el alumno desee reclamar. De esta forma, el profesor siempre tendrá a mano el código original que el alumno le envió.
3. El test asociado a la propiedad. Muchas veces hay propiedades del ejercicio que no contienen errores sintácticos y el compilador no es capaz de localizarlos. Es por esto, que el corrector de ASys pasa una serie de tests para comprobar que el código devuelve el resultado esperado. Cuando se localiza uno de estos errores, es necesario que el usuario tenga esta información ya que puede ahorrar mucho tiempo en la búsqueda del error.

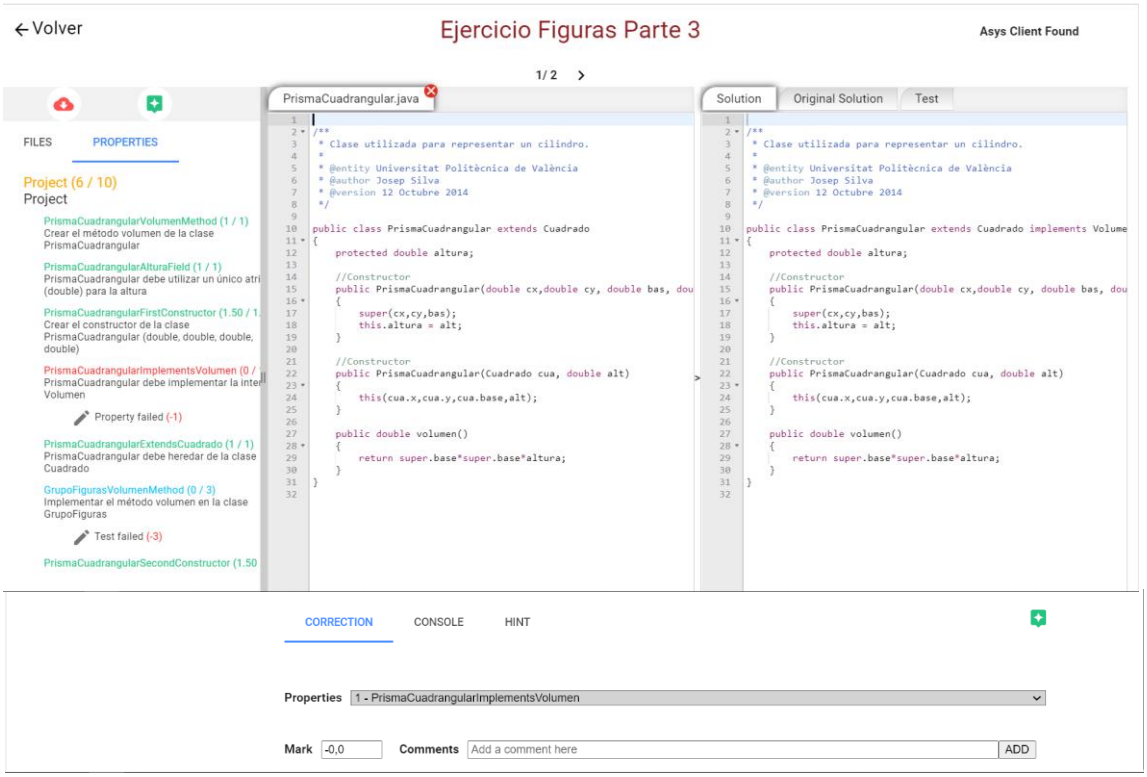

Figura 26: Vista "Code.vue" del rol del profesor

5.2.4 El IDE perspectiva del alumno

Respecto a la perspectiva que tiene el alumno del IDE, básicamente tiene una visión reducida de la visión que tiene el profesor (Figura 37). No tiene funcionalidades como ver la solución original o añadir comentarios a la corrección.

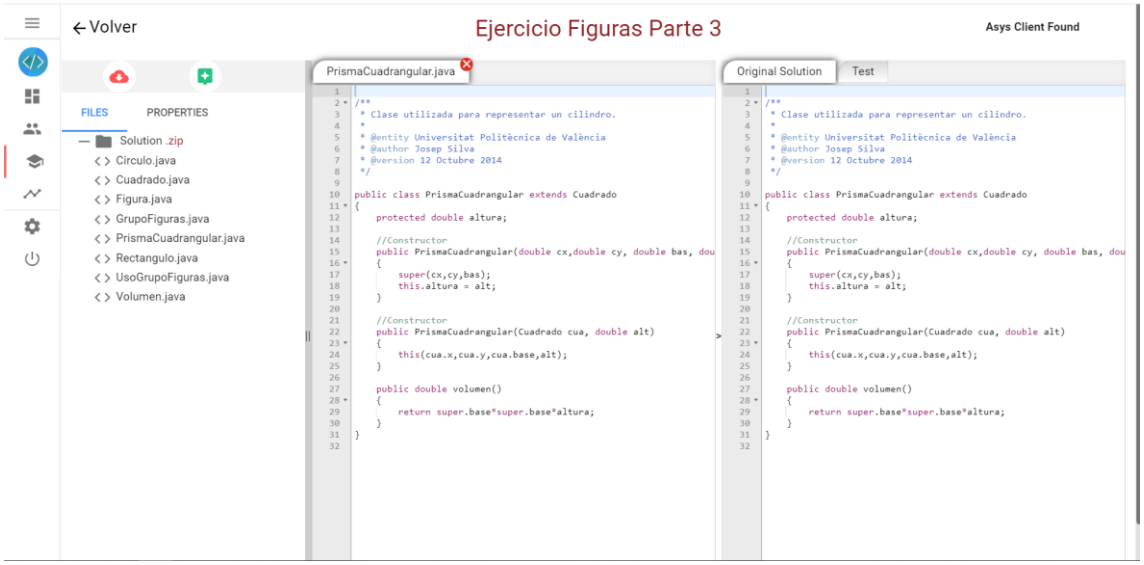

Figura 27: Vista "Code.vue" del rol del alumno

La implementación de ambas perspectivas ha sido sencilla gracias a la utilización de la renderización condicional de Vue. Consiste en utilizar la directiva "v-if" seguida de la condición que se debe cumplir; en nuestro caso lo que se utilizaba para renderizar o no una plantilla es el rol del usuario registrado.

6. IMPLANTACIÓN DE LA SOLUCIÓN

El objetivo de este apartado es explicar cómo implantar la solución propuesta en este proyecto en un entorno de explotación. Esto se abordará en dos diferentes apartados, en primer lugar, se explicarán las configuraciones previas a la instalación del proyecto y seguidamente se detallará cómo llevar a cabo dicha instalación.

6.1 Configuración

Para configurar y preparar el proyecto ASys se han de seguir los siguientes pasos:

1. Lo primero que debemos hacer es situarnos en el directorio del proyecto ASys y ejecutar el comando "npm install". Gracias a este comando obtendremos todas las dependencias necesarias para el proyecto.

2. A continuación, nos situaremos sobre el archivo "connection.js" situado en la ruta "src/services/connection.js". Una vez aquí, inicializamos la variable "baseURL" con el valor de la URL donde se sitúa el servidor web. Por ejemplo: "baseURL: 'http://localhost:8080/asysweb'".

3. Al igual que en el apartado anterior, nos situaremos sobre el archivo "index.js" situado en la ruta "src/store/index.js". Una vez aquí, inicializamos la variable "apiPath" con la misma que inicializamos "baseURL" en el apartado anterior.

4. Una vez obtenidas todas las dependencias y configuradas las URL mencionadas mediante el comando "npm run build" construiremos el proyecto. Tras la finalización de la creación del proyecto se creará una carpeta de distribución llamada *dist*.

5. Por último, moveremos dicha carpeta al servidor donde queramos depositar nuestra aplicación web.

Una vez configurado el cliente de ASys, pasaremos a configurar el servidor para su posterior desplegado.

1. En primer lugar, abriremos el archivo "application.yml". Desde este archivo se configurará la mayor parte del servidor. La ruta para su acceso es la siguiente: "src/main/resources/application.yml"

2. Configuraremos la base de datos mediante las propiedades situadas en "spring/datasource". Aquí insertaremos el usuario y la contraseña de la BD a la que queremos acceder.

3. Habilitaremos CORS [39] para que no nos bloquee el acceso de nuestro cliente web poniendo la URL en la propiedad "cors/path3". La URL ha de ser aquella donde hayamos decidido desplegar la aplicación web.

4. La propiedad ddl-auto la pondremos a “none”. Su ruta es “jpa/hibernate/ddl-auto”.

5. A continuación, configuraremos el servidor SMTP [40, 41] insertando sus credenciales o, por su defecto, si no disponemos de uno utilizaremos una cuenta de MailTrap[42]. Para la configuración de ambas credenciales hay que acceder al archivo “JavaEmailUtil.java” accediendo mediante “src/main/java/com/asysweb/útil/JavaEmailUtil.java”. [43]

6. Configuraremos el servidor de MinIO [44] (servidor de recursos) mediante el archivo “AmazonS3ServiceImpl.java”. La ruta para su acceso es “src/main/java/com/asysweb/service/AmazonS3ServiceImpl.java”.
   Inicializaremos las propiedades con las credenciales y la URL donde se haya desplegado el servidor.

7. Revisaremos que en el archivo pm.xml se encuentra la propiedad packaging en “war”.

8. Para concluir, construiremos el proyecto. El archivo resultante de esto será un war. Este war será utilizado más adelante en un servidor de explotación, es decir, se desplegará nuestra aplicación web. Cabe mencionar, que hemos de renombrar dicho archivo según el nombre mediante el cual deseemos acceder a través de la URL.

Tras realizar todos estos pasos en el orden correcto ya dispondremos de ASys preparado para ser desplegado en el entorno de explotación.

6.2 Instalación

No podemos desplegar ASys sin antes tener instalados los servidores necesarios. Así bien, en este apartado se explicará como instalar cada uno de los servidores.

Para que Asys funcione correctamente son necesarios los siguientes servidores:

1. MySQL [45]. Es un sistema de gestión de BD relacional.
2. MinIO [46]. Es un servidor de recursos.
3. Tomcat [47]. Es un servidor web y un contenedor de páginas Servlet / JavaServer.
4. SMTP o Mailtrap [48]. Es un servidor de correo.

Tras descargar todos los mencionados servidores pasaremos a explicar la instalación de aquellos que necesitan una configuración específica.

6.2.1 Instalación de MySQL

Una vez descargado el servidor MySQL, especificaremos las credenciales de acceso que previamente hemos introducido en el archivo de configuración del servidor de ASys (Figura 28).

```
datasource:
  # MYSQL
  url: jdbc:mysql://127.0.0.1:3308/asys-v2
  username: root
  password: password
  driver-class-name: com.mysql.jdbc.Driver
```

Figura 28: Configuración del servidor ASys

Por defecto, si no se pone ningún usuario, es *root*. Una vez hecho esto, crearemos una nueva base de datos (BD) utilizando el mismo nombre con el que configuramos el proyecto. Por último, importaremos el script que contiene la BD ya generada anteriormente.

Por ejemplo, para crear una BD llamada "alumnos" ejecutaríamos el siguiente comando desde la consola de MySQL:

```
CREATE DATABASE alumnos;
```

Como se puede ver en la siguiente imagen:

```
mysql> CREATE DATABASE alumnos;
Query OK, 1 row affected (0,00 sec)
```

Con el mensaje "*Query OK, 1 row affected*" ya sabremos que la BD se ha creado correctamente. Es importante no olvidar el ´;´ al final de las sentencias desde la consola MySQL.

6.2.2 Instalación de MinIO

Cabe destacar que para la descarga de MinIO es necesario descargar el binario correspondiente al sistema operativo (SO) en el que se vaya a desplegar ASys.

A continuación, extraeremos del proyecto del cliente de ASys la carpeta con su mismo nombre, y agregaremos en su interior el ejecutable previamente descargado. Seguidamente, ejecutaremos el binario correspondiente al SO en el que nos encontremos. En el binario se indicará el Puerto donde escuchará el servidor y el directorio donde almacenará los recursos. Algunos de los comandos dependiendo del SO pueden ser:

- En Linux: `./minio server ./ --address ":port"`
- En Windows: `minio.exe server ./ --address ":port"`

Como se puede observar el directorio no se indica explícitamente, pues según en la ruta donde nos encontremos a la hora de ejecutar el comando esta será la que tomará como almacén de datos. Por ejemplo, en mi caso:

```
C:\Users\ramal\Desktop\Cuarto\TFG\ASys-Web-server-juan\minio>minio.exe server ./ --address ":9001"

+------------------------------------------------------------+
| You are running an older version of MinIO released 3 months ago |
| Update: Run `mc admin update`                              |
+------------------------------------------------------------+
```

En el caso de ser la primera vez que se despliega el servidor se habrá creado una carpeta llamada ".minio.sys". Dicha carpeta contiene la configuración de las credenciales del servidor de recursos. Es necesario que se modifiquen las credenciales para que coincidan con las que se han creado el proyecto ASys. Para dicha modificación es necesario acceder a la carpeta "config.json" y cambiar las propiedades *acessKey* y *secretKey*. Para que estos cambios se apliquen es necesario reiniciar el servidor de MinIO.

6.2.3 Instalación de Tomcat

El servidor de Tomcat [49] es muy sencillo de configurar, lo único que debemos hacer es arrastrar los proyectos de distribución generados anteriormente en la carpeta "webapps". Los proyectos de distribución son respectivamente el "war" y el archivo "dist" de los que se ha hablado en el apartado 6.1 de esta memoría. En mi caso, a utilizar Wampstack [50], la carpeta no se llama "webapps" sino "htdocs", sin embargo, el prodecimiento sigue siendo exactamente el mismo. Por último, iniciaremos el servidor mediante un doble click en el archivo "startup", dicho archivo se encuentra situado en la carpeta "bin" del servidor Tomcat.

7. PRUEBAS

En este apartado se va a explicar cómo se ha realizado la comprobación del correcto funcionamiento del sistema tras la implantación del IDE para el desarrollo y la corrección automatizada de ejercicios de programación.

Se ha decidido evaluar el sistema mediante el uso de pruebas. Así bien, mediante esta técnica podemos comprobar si el sistema cumple con los requisitos especificados. En concreto se han empleado pruebas funcionales. Matías Fossati, en su libro: "Testing: conviértete en un experto probando software" se define prueba funcional como "es una prueba basada en la ejecución, revisión y retroalimentación de las funcionalidades previamente diseñadas para el software" [51]. Hay distintos tipos de pruebas funcionales según el nivel que deseemos probar [52,53]:

1. Pruebas unitarias. Se emplean para validar el correcto funcionamiento de una función o un método.
2. Pruebas de integración. Se realizan después de comprobar que los componentes individualmente funcionan correctamente, y se combinan en un entorno activo. La integración ha de planificarse para que al detectar un error se pueda conocer su origen. Se componen de dos subtipos de pruebas: Alpha y Beta.
3. Pruebas de sistema. Es el nivel más cercano a la experiencia cotidiana. El objetivo de estas pruebas es la comprobación de la integración del sistema en su totalidad, verificando el funcionamiento correcto de todos sus subsistemas.
4. Pruebas alpha. Se realizan en un entorno de desarrollo y tienen el objetivo de asegurar la utilidad y la funcionalidad solicitada por el cliente.
5. Pruebas beta. Se realizan cuando el sistema está teóricamente correcto y pasa a ejecutarse en un entorno real.
6. Pruebas de aceptación. Es el último nivel de pruebas se realiza previamente a la liberación del producto. Tienen la finalidad de precisar si se cumplen con las necesidades y/o requisitos de los *Stakeholders.*
7. Pruebas de regresión. Aquellas pruebas de software con el objeto de descubrir errores (bugs), siempre y cuando haya un cambio previo del software. Es decir, son las encargadas de comprobar que "no se ha roto nada que funcionaba bien en la versión anterior".

A continuación, se va a exponer brevemente cómo han influido en el proyecto cada uno de los mencionados tipos de prueba.

7.1 Pruebas unitarias

Las pruebas unitarias se han llevado a cabo con una frecuencia diaria, pues el propio desarrollador al mismo tiempo que iba implementando los métodos, iba comprobando que dicho método tenia la funcionalidad correspondiente. Véase en la Figura 29.

```
getPropertySelected() {
  const el = this.assessment.properties[0].children.find(e => e.id == this.selectedProperty)
  return el;
},
```

Figura 29: Método de la clase Code.vue 1

En la Figura 30 podemos observar un método de la clase Code.vue. Este método tiene la funcionalidad de retornar la propiedad seleccionada por el usuario. En una primera instancia siempre devolvía la constante 'el' sin comprobar si existía. Sin embargo, es posible que no se encuentre seleccionada ninguna propiedad.

```
getPropertySelected() {
  const el = this.assessment.properties[0].children.find(e => e.id == this.selectedProperty)
  if (el) {
    return el;
  }
},
```

Figura 30: Método de la clase Code.vue 2

En este punto el desarrollador comprueba que cuando no hay ninguna propiedad seleccionada el método devolvía *undefined* en lugar de *null*. Gracias a esta prueba de integración se consiguió el estado correcto del método "getPropertySelected()" (Figura 33).

7.2 Pruebas de integración

Las pruebas de integración en el caso de Vue.js, en la mayoría de los casos coinciden, con las pruebas de componentes. Uno de los componentes que más relevancia ha tenido en este proyecto, como se comentaba en el apartado 5.2, ha sido "Brace". En este caso, tras comprobar que todos los métodos de Brace.vue tienen un funcionamiento correcto por separado, gracias a las pruebas de integración podemos confirmar que todos ellos en su conjunto siguen funcionando sin errores.

Por ejemplo, en la Figura 31 podemos observar un fragmento del *template* de Brace.vue.

```
<div
        :class="{'code-tabs__tabs__tab-wrapper__tab': true, active: isTabActive(file)}"
        :title="file.name"
        @click="changeCurrentFile(file)">
  {{ tabName(file) }}
</div>
```

Figura 31: Fragmento de Template del componente Brace.vue

El desarrolador, partiendo de la base de saber que el método "changeCurrentFile(file)" funciona unitariamente como debe, ha de comprobar que dentro del Template continua funcionando. ¿Porqué no puede asumir su correcto

funcionamiento? No puede asumirse porque ahora se encuentra en contacto con otros métodos del componente y, por lo tanto, puede verse afectado por factores ajenos.

7.3 Pruebas de sistema

En nuestro caso las pruebas de sistema son realmente pruebas de subsistema, pues no se han probado todas las funcionalidades del sistema ASys, sino todo aquello que guarda una relación directa o indirecta con el IDE fruto de este trabajo.

Dicho esto, las pruebas de integración en este proyecto se han dividido en dos:

- Las pruebas alpha.
- Las pruebas beta.

7.3.1 Pruebas alpha

Para comprobar el estado del sistema se han probado diversos escenarios. En todos ellos siempre se parte con la BD vacía y con el sistema recién iniciado, esto se hace con el fin de evitar los estados ya preparados de la BD con los que se trabaja en producción que se utilizan para agilizar la implementación. Con todo esto conseguimos asemejarnos a un entorno más realista de cara al usuario.

El primer escenario consistió en lo siguiente:

1. Se crean tres usuarios en el Sistema ASys, a uno de ellos se le asigna el rol de **profesor** y a los otros dos el de alumno (los llamaremos: **alumno1** y **alumno2**).
2. Al **profesor** se le asigna un ejercicio, de tal forma que se simula la creación de este para que los alumnos lo puedan resolver.
3. En primer lugar, el **profesor** cuando entra en la vista del ejercicio le aparece vacío, no tiene nada para corregir.
4. A continuación, nos registramos con **alumno1** y accedemos a dicho ejercicio comprobando que todas las funcionalidades de la vista funcionan (descargar los recursos del ejercicio, descargar las instrucciones...)
5. Se realiza el ejercicio y se envía al **profesor**.
6. Ahora nos registramos con el **profesor** y comprobamos que puede visualizar la solución del ejercicio del **alumno1**, y también que podemos realizar una corrección.
7. En este momento, nos registramos con **alumno2** y realizamos los mismos pasos que **alumno1**, sin embargo, no enviamos la solución (simulando que el **alumno2** no la ha acabado y desea perfeccionarla).
8. Por último, al registrarnos con el **profesor** la solución de **alumno2** no es visible para este.

7.3.2 Pruebas beta

Las pruebas beta se realizan en un entorno real. Para su realización se ha puesto como servidor mi computadora, por lo tanto, mediante mi IP pública los usuarios podían acceder a la aplicación web de manera remota.

Mi director Josep, puesto que no ha participado en la implementación, se ha considerado usuario final.

La primera sesión de pruebas fue sin guion, es decir, el usuario final iba testeando la aplicación superficialmente, viendo que los requisitos y especificaciones definidas en el sistema se cumplían. Duró cerca de dos horas, se crearon dos usuarios y se probó la aplicación web con dos navegadores diferentes. El sistema se sometió a pruebas de estrés, introduciendo errores deliberadamente en los formularios para comprobar que el sistema los detectaba y no entraba en un estado inconsistente. Se anotaron todas aquellas mejoras que se vieron convenientes, así como errores del sistema (en esta primera revisión salieron cerca de 20 mejoras).

Cabe añadir que, por los escasos errores funcionales encontrados, se inició un proceso de búsqueda de *buggs*. Uno de los *buggs* más importantes fue un agujero de seguridad que había en el sistema y se describió de la siguiente forma:

Josep Silva (usuario final):

> *"Problema:*
> *Hay un agujero de seguridad que permite a un alumno eliminar la corrección del profesor.*
> *Solución propuesta:*
> *Cuando un alumno soluciona un ejercicio y tiene una fila en 'Assessments', esa fila debe tener un botón 'enviar al profesor'.*
> *Una vez enviado al profesor, ya no se podrán enviar más soluciones ni subir más soluciones (se debe inhabilitar el panel drop file)*
> *Cuando se haya enviado una solución al profesor, se indicará con un mensaje en la fila correspondiente de 'Assessments'"*

La resolución del agujero de seguridad del sistema se resolvió de la forma mencionada en el párrafo anterior. En la Figura 32 se puede observar el botón "Send to teacher" antes de ser presionado.

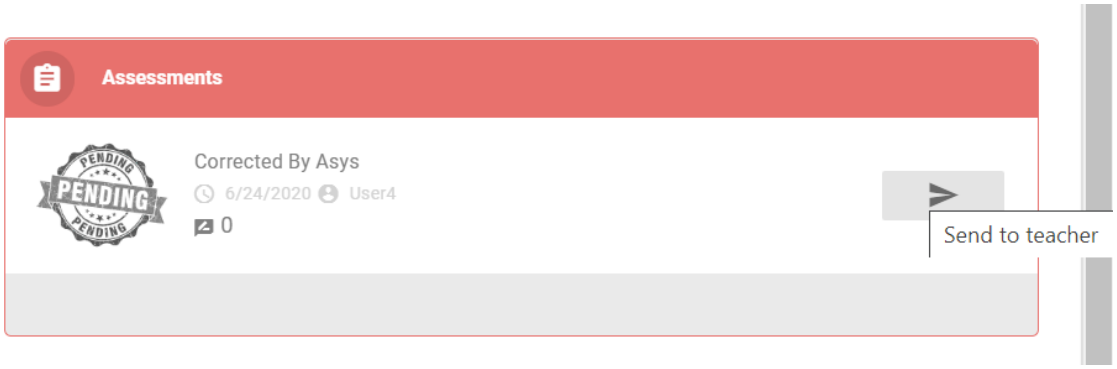

Figura 32: funcionalidad "Send to teacher" de ASys 1

Una vez presionado el botón, se indica que está enviado con un icono de "done" y aparece un mensaje informativo para el usuario (véanse las Figuras 33 y 34).

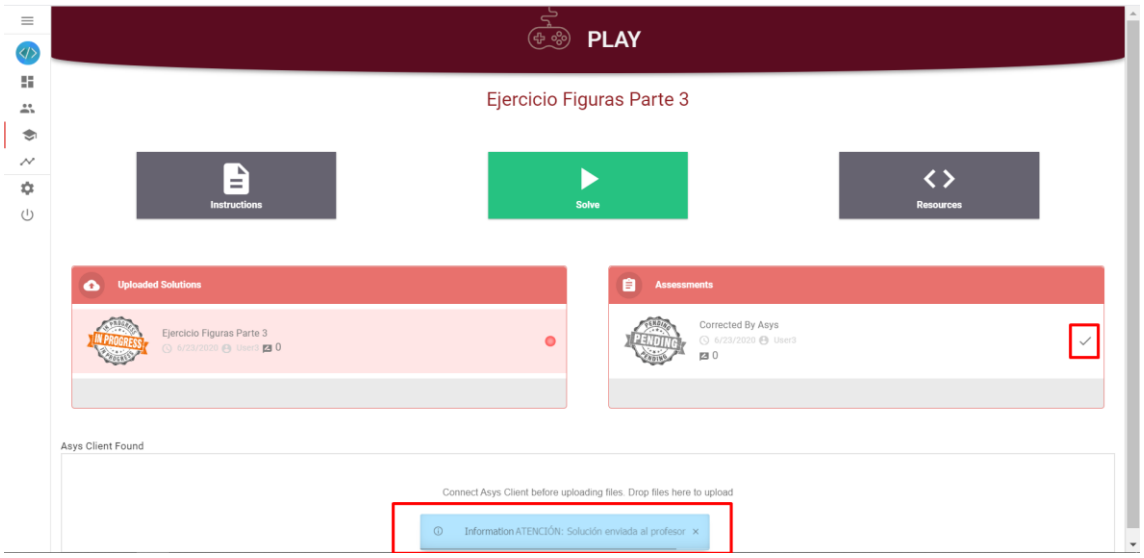

Figura 33: Funcionalidad "Send to teacher" de ASys 2

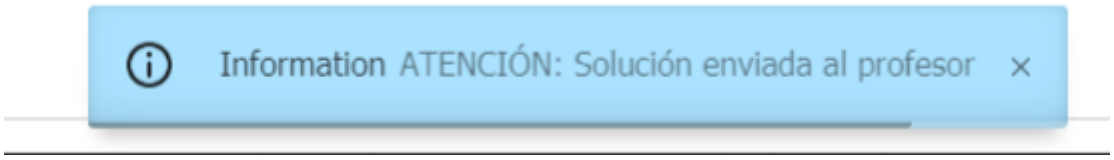

Figura 34: Funcionalidad "Send to teacher" de ASys 3

La segunda sesión de pruebas se ejecutó siguiendo el mismo patrón que en la primera sesión añadiendo a su vez pruebas de regresión con el fin de verificar que seguía funcionando lo ya comprobado previamente, es decir, que no se había "roto" nada. Duró 2 horas aproximadamente, se crearon tres usuarios y se probó la aplicación web con dos navegadores diferentes (uno de ellos nuevo respecto a la sesión anterior).

Resultado de esta sesión se obtuvo una batería de 12 mejoras y 4 errores funcionales que se habían "roto", pues en la primera sesión funcionaban correctamente. Una de las mejoras más importantes a nivel funcional de esta sesión se redactó de la siguiente manera:

Josep Silva (usuario final):

> *"Problema:*
> *Al subir la solución de un ejercicio a otro ejercicio, el sistema no se da cuenta y pone todos los ficheros vacíos.*
> *Solución: en el fichero info. además de estar el usuario, debería estar el ejercicio.*
> *Solución propuesta:*
> *El fichero "Resources.zip" debería contener un fichero info.*
> *El info tiene dos campos:*
> ***Ejercicio: "identificador del ejercicio"***
> ***Usuario: "nombre del usuario"***
> *"*

A partir de este momento los zip que los usuarios suben como soluciones a la plataforma ya no son simplemente los archivos .java, sino además contienen un archivo de texto donde se indica: el usuario responsable de la realización y el identificador del ejercicio. Además, esta solución está pensada para ser escalable, es decir, para que en un futuro se puedan añadir tantos atributos como se consideren necesarios.

Esta sesión fue muy importante en la fase de pruebas pues permitió acelerar mucho el proceso del software gracias a la detección de errores, que de haberse pasado por alto hubieran retrasado mucho el procedimiento de entrega final.

La última sesión fue la más costosa a nivel temporal, duró cerca de 3 horas, se realizó un *testing* de la aplicación web "a fondo". Se comprobaron todas y cada una de las funcionalidades implementadas en el sistema que son nombradas en esta memoria.
El sistema se sometió de nuevo a pruebas de estrés, introduciendo errores deliberadamente en los formularios para comprobar como respondía ante ellos. El sistema en ningún momento quedó en estado inconsistente, es decir, en todo momento trataba los errores y mantenía informado al usuario de su estado.

Se localizaron 4 mejoras posibles, sin embargo, no sé encontró ningún *bugg* ni error funcional, fruto de esto se dio por finalizada la fase de pruebas beta.

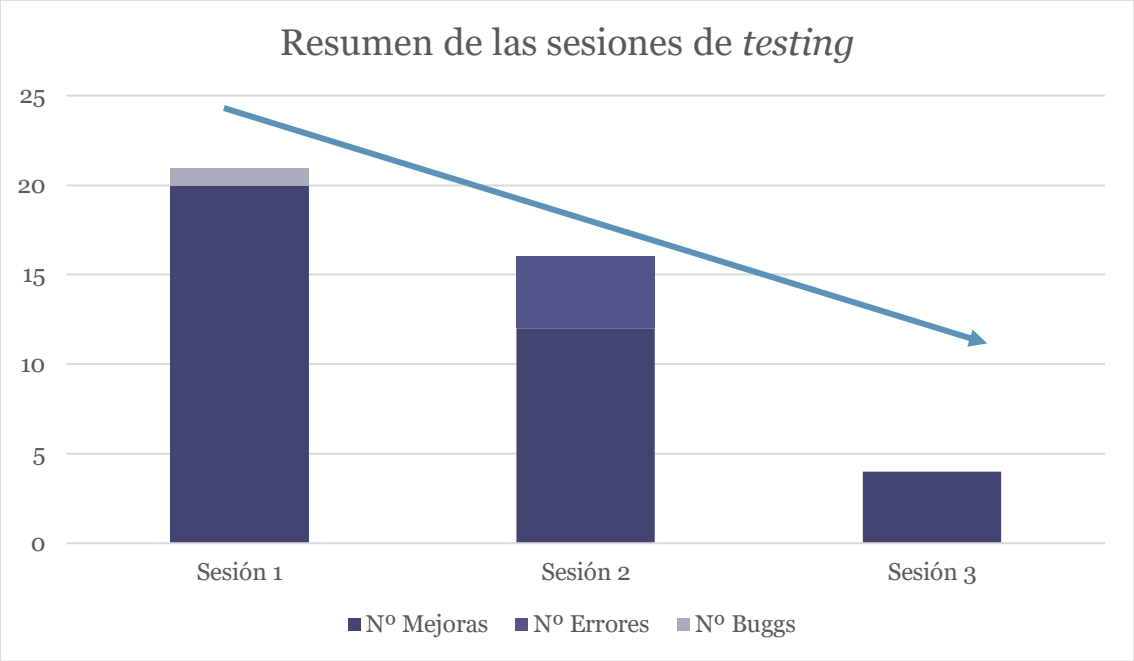

Figura 35: Resumen de las sesiones de testing

En el gráfico "Resumen de las sesiones de *testing*" se puede observar cómo ha habido una tendencia decreciente respecto a la aparición de *bugs*, mejoras y errores. Como resultado de dicha tendencia podemos deducir que la experiencia ganada a lo largo de los diversos *sprints* del proyecto ha sido decisiva a la hora de enfocar la implementación y el diseño de las unidades de trabajo, que se ha visto reflejada en un menor tiempo de retrabajo.

8. CONCLUSIÓN

Para concluir este proyecto lo primero que he de mencionar es la figura de mi tutor Josep Silva. Sin duda ha sido gracias a él por lo que he podido finalizar este proyecto en el periodo planeado. Me ha orientado siempre que lo he necesitado y me ha aconsejado en todas las decisiones de diseño y en la redacción de la memoria final. Ahora puedo verme hace unos meses, lo ingenuo que era, pensando que esto sería simplemente uno más de los muchos proyectos realizados en la carrera. Y, por encima de todo, deseo agradecerle la confianza depositada en mí para la participación en un proyecto en el que se han invertido años de dedicación y que tiene tantas posibilidades de trascender en el sistema educativo y redundar en la mejora de la enseñanza. También deseo agradecer a mi tutor Armando Maya su inestimable ayuda en la comprensión del sistema y en la implementación. Al principio del proyecto anduve muy perdido y de no ser por su ayuda posiblemente aún estaría indagando, tratando de solucionar alguno de los innumerables problemas que he tenido a lo largo del proyecto. Muchos de ellos eran tonterías provocadas por la simple inexperiencia en los lenguajes de programación donde tuve que iniciarme, pero en el ámbito de la informática estas tonterías son las que llevan horas, días o incluso semanas en poder solucionarse.

El desarrollo de este proyecto ha sido una gran experiencia para mí. Me ha aportado una ingente cantidad de conocimiento en muy pocos meses y, sobre todo, me ha abierto los ojos de cara a lo que soy capaz de hacer. En muy poco tiempo he aprendido nuevos lenguajes de programación incluidos en dos *frameworks* de desarrollo industrial; y he aprendido a instalar, configurar y mantener cuatro servidores diferentes y a hacerlos interactuar entre ellos. Todo esto ha posibilitado la integración de muchos conocimientos que había estudiado durante la carrera de manera aislada.

De todo lo aprendido he de destacar las tecnológicas necesarias para poder llevar a cabo el desarrollo del sistema ASys. Tecnologías de las que nunca había oído hablar, pues eran tan novedosas en el mundo de la programación que solo habían sido adoptadas por una comunidad muy reducida. Pero que, gracias a su potencial, han destacado entre sus competidores y se están convirtiendo en tecnologías de uso generalizado. El simple hecho de saber utilizarlas, saber cómo funcionan y saber cómo poder explotar este potencial me hace sentir, en estos momentos, muy realizado.

Como bien me dijo el día que se me propuso este proyecto mi tutor Josep Silva: “será un proceso duro de aprendizaje, pero los resultados serán exponenciales”. Y, a día de hoy puedo recordar aquellas palabras y decir con orgullo que tenía mucha razón.

En lo referente a los objetivos que se pretendían abordar al iniciar este proyecto (apartado 1.2) hemos de remitirnos a los hechos y al *feedback* aportado por los usuarios a los que esta aplicación web está orientada, es decir, alumnos y profesores. En estos momentos, el sistema ASys va a entrar en la fase de pruebas de aceptación, con usuarios reales trabajando en concurrencia. Todas las valoraciones realizadas por usuarios son positivas y muy satisfactorias.

Hay que tener en cuenta un aspecto muy importante que ya se tuvo presente a la hora de elaborar el proyecto, y es el siguiente: la mayoría de los profesores han utilizado durante décadas el proceso de corrección manual, un proceso tradicional y eficaz pero no eficiente. Y, por el miedo al cambio, es previsiblemente complicado que adopten ASys como herramienta de corrección. En muchas ocasiones similares se da por buena la locución "si funciona, no lo cambies"; y es cierto que funciona, pero ¿podría funcionar mejor? Este proyecto es una prueba de que sí podría funcionar mejor, y de que sí es posible mejorar los sistemas de aprendizaje actuales.

Para resumir, pienso que este proyecto será un éxito rotundo cuanto sea implantado en la universidad. Los alumnos dispondrán de una herramienta automatizada de auto-corrección. Dispondrán de baterías de ejercicios auto-corregibles. Minimizando así la necesidad de interacción con el profesor para resolver sus dudas y, por tanto, los tiempos de corrección. Los profesores tendrán una herramienta de corrección exhaustiva donde todos los errores serán detectados automáticamente. Además, el sistema guardará las correcciones, de tal forma que el mismo error no se penalizará de manera diferente a dos alumnos, ni siquiera si los que corrigen son profesores diferentes. Estas y muchas otras ventajas descritas en la memoria son el aporte del presente proyecto.

9.BIBLIOGRAFÍA

[1] Nirmal Hota, Tadit Dash, Dr. Vishal Jain, Learn Vue.js in 7 Days: Journey Through Vue.js, BPB Publications, 2019

[2] Agus Kurniawan, Vue.js Programming by Example, editor: PE PRESS, chapters 2-3-5-6

[3] Olga Filipova, Learning Vue.js 2, editor: Pakt Publishing Ltd, 2016, chapters 3-4-8

[4] Brett Nelson, Getting to Know Vue.js: Learn to Build Single Page Applications in Vue from Scratch, editor: Apress, 2018, chapters 4-5-6

[5] Sergio Rios, Web Project Spring Java Revolutions: J2EE Architecture with Spring, 2013, pp 4-11

[6] Dhrubojyoti Kayal, Pro Java EE Spring Patterns: Best Practices and Design Strategies Implementing Java EE Patterns with the Spring Framework, Apress , 2008, chapters 1-2-5

[7] Ivica Crnkovic, Volker Gruhn, Matthias Book (Eds.): Software Architecture 5th European Conference, ECSA 2011, Essen, Germany, Septemver 13-16, 2011, Springer-Verlag, pp. 26-35; 83-105.

[8] Niclkas Persson, Christopher Murphy: HTML and CSS Web Standards Solutions A Web Standardistas' Approach, 2009

[9] Marijn Haverbeke: Eloquent Javascript, 2nd edition, Introducción moderna a la programación, 19-02-2020. Disponible: https://www.pdf-manual.es/programacion-web/javascript/181-eloquent-javascript-en-espanol.html

[10] Kyle Simpson: You don't know JS yet - Get Started - 2nd edition, Front End Masters. 19-03-2020. Disponible: https://github.com/getify/You-Dont-Know-JS/tree/2nd-ed/get-started

[11] Cody Lindley: Front-end Developer Handbook 2019, ed. Front End Masters. 19-03-2020. Disponible: https://frontendmasters.com/books/front-end-handbook/2019/

[12] Vuejs Team, Vue.js, 13-03-2020. Disponible: https://vuejs.org/

[13] Pivotal Software Inc, Spring.io, 13-03-2020. Disponible: https://spring.io/

[14] Juan Carlos Moreno Pérez, Entornos de Desarrollo. Editorial Síntesis, S. A, pp 37-41. 12-03-2020. Disponible: https://www.sintesis.com/data/indices/9788491711612.pdf

[15] José Luis Comesaña Cabeza, Entornos de Desarrollo del curso de "Desarrollo de Aplicaciones Web", 2011-2012, pp 5-8. Disponible: https://www.sitiolibre.com/curso/pdf/ED2.pdf

[16] Paul Halliday: Vue.js Design Patterns and Best Practices, ed. Packt Publishing Ltd, 15-02-2020. Disponible: https://sites.google.com/site/didushiano1/1eKw2jRh1343, pp 32-40, 62-105, 131-164

[17] Patricio Orlando Letelier Torres: Modelos de proceso para desarrollo ágil. 30-03-2020. Disponible: http://agilismoatwork.blogspot.com/2014/10/modelos-de-proceso-para-desarrollo-agil.html
[18] Rubén Bernárdez: Solveet, Beta. 13-03-2020. Disponible: http://ww1.solveet.com/
[19] Topcoder Team: Topcoder, 13-03-2020. Disponible: https://www.topcoder.com/challenges
[20] Codeacademy Team: Codeacademy, 13-03-2020. Disponible: https://www.codecademy.com/
[21] Codeacademy Team: Codeacademy.pro, 13-03-2020. Disponible: https://www.codecademy.com/pro/membership
[22] HackerEarth Team: HackerEarth, 13-03-2020. Disponible: https://www.hackerearth.com/challenges/
[23] Universidad Politècnica de València: PoliformaT, 2003 . Disponible: https://poliformat.upv.es/
[24] Hadi y Ali Partovi: Code.org, 20-03-2020. Disponible: https://code.org/
[25] Alexis Marin: ¿Qué es Code.or?, 22-03-2020. Disponible: https://alexismarin.wordpress.com/2013/03/13/que-es-code-org/
[26] Replt Team: repl.it, 13-03-2020. Disponible: https://repl.it/
[27] Udemy Inc: Udemy Online Courses – Learn Anything, On your Schedule, 25-03-2020. Disponible: https://www.udemy.com/
[28] G. Kotonya and I. Sommerville, Requirements Engineering: Processes and Techniques, John Wiley & Sons, 2000, pp 141-154
[29] A. van Lamsweerde, Requirements Engineering: From System Goals to UML Models to Software Specifications. John Wiley & Sons, 2009, chapter 4 (Requirements Specification & Documentation)
[30] E. Gottesdiener, The Software Requirements Memory Jogger: A Pocket Guide to Help Software And Business Teams Develop And Manage Requirements. Goal Q P C Inc, 2005.
[31] Daniel Kaheman: Pensar rápido, pensar despacio Ed. 10, Psicología, editorial. Debate, pp 80-99
[32] Ley Orgánica 15/1999 , de 13 de diciembre, de Protección de datos de Carácter Personal. BOE, núm. 298, de 14/12/1999, pp 43088 - 43099
[33] Armando Maya Gomis: Trabajo Fin de Grado: "*Diseño y Desarrollo del sistema Asys con Spring y Vue.js*". 2018-2019
[34] Juan Ferrer Martínez, Desarrollo de Interfaces, Editoral: RA-MA, pp 51-62
[35] Kenneth E. Kendall, Análisis y diseño de sistemas, Editorial: Person Education, 2005, Capítulo 2
[36] Thlorentz: brace, versión: 0.11.1, 28-03-2020. Disponible: https://www.npmjs.com/package/brace
[37] Ajax.org B.V.:Ace. 02-04-2020. Disponible: https://github.com/ajaxorg/ace-builds
[38] Ilya Grigorik: High Performance Browser Networking, 1st Edition, ed. O'reilly,
[39] Mozilla Team: CORS. 03-04-2020. Disponible: https://developer.mozilla.org/es/docs/Web/HTTP/Access_control_CORS

[40] Mailrelay Team: SMTP, Simple Mail Transfer Protocol. 25-04-2020. Disponible: https://blog.mailrelay.com/es/2017/04/25/que-es-el-smtp
[41] 1&1 IONOS España S.L.U: SMTP server. 25-04-2020. Disponible: https://www.ionos.es/digitalguide/correo-electronico/cuestiones-tecnicas/servidor-smtp/
[42] Railsware Products Inc: Mailtrap. 12-03-2020. Disponible: https://mailtrap.io/
[43] Clemir Rondón: ¿Cómo enviar emails de prueba?, 25-04-2020. Disponible: https://styde.net/como-enviar-emails-de-prueba-con-mailtrap-io-en-laravel/
[44] MinIO Inc: MinIO. 15-04-2020. Disponible: https://min.io/
[45] Oracle Corporation and/or its affiliates: Download MySQL server. 15-04-2020. Disponible: https://downloads.mysql.com/archives/installer/
[46] MinIO Inc: Download MinIO server. 15-04-2020 . Disponible: https://min.io/download#/windows
[47] The Apache Software Foundation: Download Tomcat Apache. 15-04-2020. Disponible: https://tomcat.apache.org/download-80.cgi
[48] Tucows Team: Download SMTP. 15-04-2020. Disponible: http://www.tucows.com/preview/854208/SMTP-Server
[49] The Apache Software Foundation: Apache Tomcat. 15-04-2020. Disponible: http://tomcat.apache.org/
[50] Bitnami: Bitnami WAMP Stack. 20-04-2020. Disponible: https://bitnami.com/stack/wamp
[51] Matias Fossati: Testing, convertite en un experto probando software, ed. 1, 2016, pp 27
[52] Edgar Serna M.: Prueba functional del software, Un proceso de verificación constante, editorial ITM, Pp 30-46
[53] Collard, J. f, Burnstein, I., Practical Software Testing: A Process-Oriented Approach, Springer, 2003.