Máster Universitario en Tecnologías Web, Computación en la Nube y Aplicaciones Móviles

Trabajo Fin de Máster

Desarrollo de una plataforma web para la resolución de problemas aritmético-verbales

Autor: Javier Valero Cejudo

Tutor: Francisco Grimaldo Moreno

Septiembre 2021

VNIVERSITAT DE VALÈNCIA

Escola Tècnica Superior d'Enginyeria ETSE-UV

Máster Universitario en Tecnologías Web, Computación en la Nube y Aplicaciones Móviles

Trabajo Fin de Máster

Desarrollo de una plataforma web para la resolución de problemas aritmético-verbales

---

Autor: Javier Valero Cejudo

Tutor: Francisco Grimaldo Moreno

TRIBUNAL

PRESIDENTE/A:

VOCAL 1:

VOCAL 2:

FECHA DE DEFENSA:

CALIFICACIÓN:

Declaración de autoría:

Yo, Javier Valero Cejudo, declaro la autoría del Trabajo Fin de Máster titulado "Desarrollo de una plataforma web para la resolución de problemas aritmético-verbales" y que el citado trabajo no infringe las leyes en vigor sobre propiedad intelectual. El material no original que figura en este trabajo ha sido atribuido a sus legítimos autores.

Valencia, 23 de septiembre de 2021

Resumen:

En los últimos años la tecnología ha evolucionado rápidamente, provocando una transformación de los procesos analógicos a procesos digitales. Como consecuencia, surge la necesidad de informatizar las distintas herramientas que poseemos, como los tests de resolución de problemas aritmético-verbales.
Además, debido al hecho de que queremos que los procesos sean cada vez más automatizados, es necesario desarrollar un corrector que automatice la corrección de estos problemas.

Agradecimientos:

En primer lugar, me gustaría agradecer todo el apoyo recibido por parte de mis padres y mi hermana durante todo este año. Sobre todo, los meses de exámenes que han sido muy duros.
También agradecer a todos mis compañeros de máster por ayudarme en clase y por los buenos momentos.
Por último, debo agradecer a mi tutor, Francisco Grimaldo, por el apoyo prestado en la elaboración de este proyecto y su ayuda y disponibilidad siempre que ha podido.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1. Introducción

En los últimos años, con la evolución constante de la tecnología, estamos viviendo una transformación de procesos analógicos a procesos digitales, los cuales, a su vez, están cada vez más automatizados. Como consecuencia de esto, surge la necesidad de informatizar los tests de resolución de problemas aritmético-verbales.

Los problemas aritmético-verbales son problemas de matemáticas cuyos datos se presentan en forma de enunciado exclusivamente mediante el lenguaje verbal, y se resolverán usando las operaciones elementales: suma, resta, multiplicación y división.

Un ejemplo de este tipo de problema sería el siguiente: "Tenía 20 cromos y me regalaron 50 cromos ¿Cuántos cromos tengo ahora?". Para resolverlo sumamos 50 más 20 y el resultado sería 70. Si esta operación la ponemos en forma de árbol de manera que un nodo tenga la operación y su resultado y los hijos de este nodo sean los datos de esa operación, tendríamos la siguiente estructura:

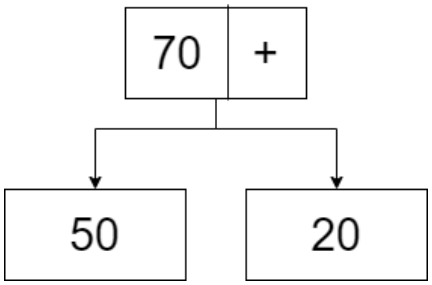


Figura 1.1: Ejemplo de resolución de un problema arimético-verbal

Por tanto, la respuesta a los problemas que plantean los tests que se quieren informatizar serían estas estructuras en forma de árbol.

Este proyecto se centrará en la implementación de un test informatizado para la resolución de problemas aritmético-verbales, es decir, un test que se pueda realizar mediante un navegador web (p.ej. Google Chrome, Mozilla Firefox, etc.) y que pueda automatizar la corrección de estos problemas mediante un corrector automático.

Dada la naturaleza de los problemas, y que estos además permiten evaluar el aprendizaje de los alumnos, el test está principalmente dirigido a los alumnos de primaria.

Además, está dirigido a este tipo de alumnos debido a que se puede evaluar, no solo

el nivel de matemáticas que poseen, sino también la capacidad lectora, ya que toda la información para resolver un problema viene dada por un enunciado y su comprensión afecta a la solución que plantee el alumno.

1.2. Motivación

La tecnología está en constante evolución, y por ello, los procesos analógicos se están transformando en digitales, siendo cada vez más automatizados. Es por esto que el motivo principal de este proyecto es informatizar los tests de problemas aritmético-verbales.

Además, otro de los motivos por el cual realizar este proyecto es el de automatizar procesos, como el de corregir estos problemas o el poder evaluar el tiempo que se tarda en realizar un problema.

1.3. Objetivos

El principal objetivo de este proyecto es desarrollar una plataforma web que permita a los usuarios resolver problemas aritmético-verbales. Para poder alcanzar este objetivo principal, se plantean los siguientes objetivos específicos:

El primer objetivo específico consiste en desarrollar la plataforma web teniendo en cuenta que los usuarios tendrán roles (administrador, profesor y alumno), los cuales determinarán las acciones que pueden realizar en la plataforma.

El segundo objetivo específico es permitir que los usuarios con el rol de profesor puedan definir los problemas aritmético-verbales y sus posibles soluciones para que los resuelvan los alumnos.

El tercer objetivo específico que se tratará de alcanzar es el de realizar un seguimiento durante la resolución de los problemas, permitiendo recoger las acciones que realicen los alumnos.

El último objetivo específico consistirá en tratar de generar una respuesta o feedback a los problemas aritmético-verbales resueltos, automatizando su corrección.

1.4. Organización de la memoria

La memoria de este proyecto se estructura de la siguiente forma:

- **Capítulo 1:** capítulo en el cual se introduce el tema del proyecto, la motivación para realizarlo y los objetivos que tiene.
- **Capítulo 2:** en este capítulo revisaremos las aplicaciones similares que se han desarrollado previamente y qué tecnologías existen actualmente para poder realizar este proyecto.
- **Capítulo 3:** en este capítulo se definirán los requisitos del proyecto, la planificación de los costes y su estimación y un estudio de los riesgos que pueden impedir la finalización del proyecto, así como un estudio de su viabilidad.

- **Capítulo 4:** en este capítulo se realizará un análisis del proyecto especificando sus casos de uso, así como los diagramas de actividades.
- **Capítulo 5:** en este capítulo se realizará tanto el diseño de la base de datos como el diseño del modelo de datos que usará la plataforma web.
- **Capítulo 6:** en este capítulo se incluirán detalles sobre la implementación de la plataforma y la descripción de las pruebas realizadas.
- **Capítulo 7:** en este capítulo se realizará una revisión de los costes definidos en el capítulo 3, se expondrán las conclusiones de este proyecto y cuál es su futuro.

Capítulo 2

Estado del arte

2.1. Análisis de aplicaciones similares

2.1.1. Read&Learn

Read&Learn (R&L) es un sofware educativo para diseñar y realizar experimentos sobre comprensión y aprendizaje de textos que está disponible en la web mediante el uso de un navegador web (Google Chrome, Mozilla Firefox, etc.).

R&L permite registrar las acciones que hace un alumno durante la realización del experimento y las transforma en variables (p.ej., tiempo de lectura o decisiones de lectura) que ayudan a comprender las estrategias de aprendizaje de los alumnos. Además, ofrece la posibilidad de descargar estas variables en un fichero mediante un exportador integrado.

2.1.2. SupLEC

SupLEC es otro software educativo para evaluar la capacidad lectora de los alumnos mediante la realización de un test de palabras y un test de textos, pudiendo elegir realizar uno de los dos o ambos, y también esta disponible en la web mediante el uso de un navegor web.

SupLEC también permite registrar las acciones que hace un alumno durante la realización de los tests y las transforma en variables (p.ej., número de palabras acertadas, tiempo de lectura de un texto, etc.) que ayudan a los docentes a determinar la capacidad lectora de sus alumnos e incidir en qué aspectos deben incidir para que mejoren.

Además posee un exportador integrado, no solo para descargar esas variables en un fichero, sino que también para descargar un informe que donde representamos estas variables mediante gráficos.

2.2. Tecnologías

2.2.1. HTML

HTML[1] (Hyper Text Markup Language) es un lenguaje de marcas, que se conocen como *etiquetas* o *tags*, y que mediante el formato ASCII se indica qué contenido y cómo

se visualiza un documento HTML.

Dado que es el único lenguaje que permite generar documentos hipertexto y que es la base de cualquier página web, se usen o no frameworks tanto de frontend como de backend, es razón suficiente para usar este lenguaje.

Además, HTML nos proporciona los elementos básicos como enlaces, botones, tablas, formularios, elementos para organizar el contenido, etc. que son la base para crear cualquier página web.

2.2.2. CSS

CSS[2] (Cascade Style Sheet u Hojas de Estilo en Cascada), nos proporciona un gran número de regrlas de formato que nos permiten controlar el aspecto de un documento HTML, que enriquecen el propio documento.

HTML proporciona mediante clases o el atributo *style* a los tags para dar formato a un documento HTML, como por ejemplo que un botón sea de color azul o rojo, o que el texto tenga un tamaño determinado. Pero es insuficiente, pues hace que sea repetitivo el uso de un mismo formato en múltiples páginas web y también hace que pueda llegar a ser inteligible el código HTML.

Por ello, CSS permite separar el formato del contenido en ficheros a parte e incorporarlos mediante la etiqueta `link`. Además permite definir un mismo formato solo una vez y usarlo múltiples veces.

2.2.3. Bootstrap

Bootstrap[3] es un framework opensource de CSS que contiene plantillas de diseño para todos los tags que nos proporciona el lenguaje HTML.

Además, nos proporciona un diseño base del cual partir y reduce la carga de trabajo a la hora de tener que dar formato a un documento HTML. También incorpora las clases `row` y `col` que nos permiten organizar el contenido de los documentos mediante un sistema de grid.

2.2.4. JavaScript

JavaScript[4] es un lenguaje de script orientado a la web que nos permite interactuar tanto con el DOM (Document Object Model) como con el BOM (Browser Object Model), es decir, permite modificar el documento HTML e interactuar con el propio navegador.

JavaScript también permite comunicar el navegador con el servidor de forma asíncrona mediante la tecnología AJAX (Asynchronous JavaScript And XML). De esta forma se puede modificar el DOM con datos del servidor sin tener que recargar la página web, lo que permite darle dinamismo a una página web.

2.2.5. jQuery

jQuery[5] es una librería de JavaScript que permite simplificar la manipulación de documentos HTML, el manejo de eventos, el uso de animaciones, el uso de la técnica

AJAX, etc.

Además, es una librería open source que tiene un doble licenciamiento bajo la Licencia MIT y la Licencia Pública General de GNU v2, y por tanto, permite usar esta librería tanto en proyectos libres como en privados.

Si quisiéramos realizar una petición `POST` mediante AJAX y JavaScript, siendo postData los datos a enviar a un servidor PHP, realizaríamos lo siguiente:

```
let xhttp = new XMLHttpRequest();
xhttp.open("POST", "ajaxfile.php", true);
xhttp.setRequestHeader("Content-Type", "application/json");
xhttp.onreadystatechange = function() {
   if (this.readyState == 4 && this.status == 200) {
    let response = this.responseText;
    console.log(response);
   }
};
xhttp.send(JSON.stringify(postData));
```

En cambio, con jQuery la misma petición se realizaría de la siguiente forma:

```
$.post("ajaxfile.php", postData)
    .done(data => {
        data = JSON.parse(data);
        console.log(data);
    });
```

Como se puede ver en estos ejemplos, al utilizar jQuery, se simplifica mucho la realización de una petición `POST` a un servidor PHP y facilita el desarrollo web.

2.2.6. Angular

Angular[6] es un framework de frontend basado en JavaScript para el desarrollo de aplicaciones web dinámicas. Además permite aplicar el patrón Modelo-Vista-Controloador (MVC) al desarrollo de aplicaciones web y también permite simplificar el desarrollo de aplicaciones de página única (SPA, Simple Page Applications).

Angular se basa en clases tipo "Componentes", cuyas propiedades son las usadas para hacer el binding de los datos. En dichas clases tenemos propiedades (variables) y métodos (funciones a llamar).

2.2.7. Java

JavaEE

Java Enterprise Edition (JavaEE)[7] es un conjunto de especificaciones para desarrollar aplicaciones en Java.

JavaEE proporciona APIs para servicios web, modelo de componentes, gestión y comunicación que lo convierten en un estándar a la hora de implementar aplicaciones web.

También proporciona una arquitectura multi-capa, formdas por la capa de cliente, constituida por el navegador web, las capas web, mediante las tecnologías Servlets, JSP

y JSF y las capas de negocio, mediante el uso de EJB, JMS o Web Services.

Todas estas capas se comunican con una capa de datos, mediante el uso de una base de datos o aplicaciones y sistemas legacy.

Spring

Spring[8] es un framework para el desarrollo de aplicaciones empresariales basadas en Java, aunque también tiene soporte para Kotlin y Groovy. Es además portable y lightweight y permite construir las aplicaciones con POJOs (Plain Old Java Objects).

También proporciona muchas características, como la inyección de dependencias o el uso de módulos, que permiten aumentar la productividad de los desarrolladores.

Aunque Spring provee muchos módulos, podemos destacar los módulos Spring Test, Spring MVC, Spring Data, Spring Rest y Spring Security entre otros.

2.2.8. PHP

PHP[9] (Hypertext Preprocesor) es un lenguaje de scripting de propósito general y de código abierto orientado exclusivamente al servidor.

También permite embeber código PHP en documentos HTML, permitiendo crear páginas dinámicas con datos del servidor, como las páginas JSP de Java, y con soporte tanto para bases de datos SQL como para NoSQL.

2.2.9. Python

Django

Django[10] es un framework basado en Python que promueve el desarrollo rápido y el diseño limpio y pragmático. Además permite crear aplicaciones web sin necesidad de reinventar la rueda, por ello sigue el principio Don't Repeat Yourself (DRY).

Django requiere el uso de una versión Python 2.5 o superior y proporciona soporte para bases de datos SQLite, MySQL, PostgreSQL y Oracle Database. También proporciona un servidor ligero embebido, un WSGI (Python Web Server Gateway Interface), que usando el módulo de Apache (mod_python), se puede desplegar la aplicación web en un servido Apache.

Django usa una variante del patrón MVC llamado MVT (Model-View-Template) que hace uso de Templates, Urls & Views y Models & Managers para implementar la Vista, el Controlador y el Modelo respectivamente.

Flask

Flask[11] es un microframework de Python que permite crear aplicaciones web de forma rápida y con pocas líneas de código. Está basado en la especificación WSGI de Werkzeug[12] y en el motor de templates Jinja2[13].

Al igual Django, Flask también permite usar el patrón MVC, proporciona un servidor web de desarrollo y es compatible con WSGI. Pero al ser un microframework no

proporciona un ORM o una forma de manejar a los usuarios, como sí tiene Djanjo.

Por ello, Flask usa plugins (extensiones), siendo los más comunes Flask-SQLAlchemy[14], que proporciona un ORM para bases de datos SQL, Flask-Login[15], para el manejo de las sesiones y los usuarios, Flask-WTF[16], para el uso de formularios desde el servidor, y Flask-Babel[17], para la internacionalización (i18n) del texto de los documentos HTML.

2.3. Propuesta

En resumen, tanto R&L como SupLEC comparten muchos aspectos en común, como el hecho de tener un exportador integrado, que se registren las acciones de los alumnos, etc. y son aspectos que también se busca que tenga este proyecto, por lo que son muy buenas aplicaciones en las cuales basarse.

Además, cabe destacar que R&L es un software que ya tiene varios años y ha sido probado en diferentes contextos y por lo tanto, podemos usar algunas de sus funcionalidades sin temor a que existan graves errores.

En cuanto a qué lenguaje de programación de los anteriores descrito se puede usar, realmente se pueden usar cualquiera, pero cabe destacar que tanto Spring, Angular o Django poseen una curva de aprendizaje mayor que el resto.

En cuanto al resto, poseen una curva de aprendizaje más leve y, a pesar de que tanto en R&L como en SupLEC se ha utilizado PHP, para este proyecto me he decantado por usar Flask, no sólo por su sencillez sino también por el hecho de utilizar un framework que tenga una curva de aprendizaje muy sencilla y que requiera poco tiempo de dominar.

Dado la elección de usar Flask, un microframework de backend, para el frontend, se va a utilizar HTML, CSS y JavaScript en combinación con Bootstrap y jQuery mediante el uso del motor Jinja que proporciona Flask.

En conclusión, se utilizará Flask, HTML, CSS y JavaScript en combinación con jQuery y Bootstrap para realizar una plataforma web para resolver problemas aritmético-verbales, registrar las acciones de los alumnos y posteriormente automatizar la corrección de éstos. Además, se reutilizarán algunas funcionalidades de R&L y SupLEC para su desarrollo.

Capítulo 3

Requisitos, coste, riesgos

3.1. Requisitos

3.1.1. Requisitos Funcionales

- Instanciar los experimentos en un grupo
- Clonar los experimentos y las soluciones de un problema
- Validar la solución de un alumno mediante un corrector automático
- Resolver los problemas mediante un grafo con forma de árbol compuesto por nodos
- Bloquear el acceso del alumno mientras haya terminado el experimento o no haya ningún experimento instanciado en su grupo
- Registrar datos (tiempo y solución) de los problemas
- Proporcionar una calculadora para resolver los problemas y registrar los cálculos realizados
- Registrar el enmascarado y desenmascarado de las proposiciones de un problema
- Generar variables a partir de los datos recogidos para usar en investigación

3.1.2. Requisitos No Funcionales

- Habrá tres tipos de usuarios: Administradores, Profesores y Alumnos
- Acceso exclusivo de los alumnos a resolver los problemas
- Los profesores únicamente tendrán acceso a sus grupos, experimentos y datos y resultados de sus alumnos
- Los administradores tendrán acceso tanto a los grupos, experimentos y datos y resultados de los alumnos como a los datos de todos los administradores y profesores
- Los alumnos deben rellenar un formulario antes de resolver los problemas

- El enunciado de un problema se dividirá en proposiciones que se marcarán mediante un resaltado
- La dificultad de un problema estará determinada por la información que contienen sus nodos en la paleta
- La interfaz del alumno estará en Castellano, Valenciano o Inglés y será determinado por el idioma establecido en el grupo al que pertenezca
- Los nodos estarán formado por el literal, una imagen, un campo que especifique el tipo de nodo (valor o ?), el dato del nodo y la operación que realiza

3.2. Costes

3.2.1. Planificación de Costes

Planificación del Trabajo

Análisis. Se analizará el proyecto mediante las siguientes actividades:

1. Se realizará un análisis de los requisitos de este proyecto para elaborar la mejor solución posible.
2. Se analizarán aplicaciones similares para poder reutilizar las funcionalidades que tengan en común.
3. Se analizarán las tecnologías web para encontrar la mejor tecnología con la cual poder realizar este proyecto.

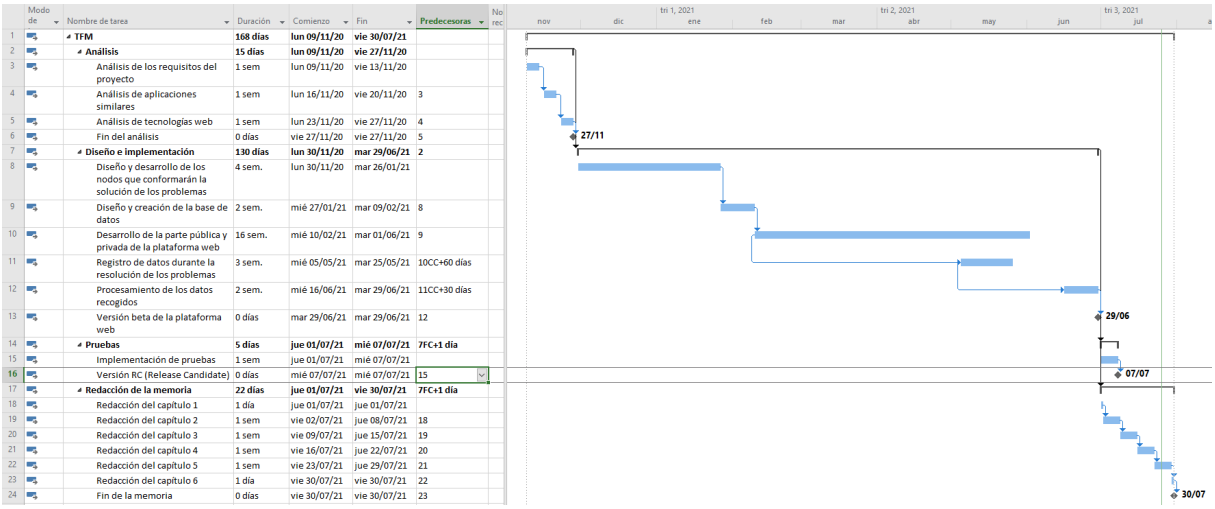
Diseño e Implementación. Se realizará un diseño preliminar de la plataforma y posteriormente se implementará mediante las siguientes actividades:

1. Se diseñarán y desarrollarán los nodos que conformarán la solución de los problemas que debe resolver el alumno.
2. Se diseñará y creará la base de datos que almacenará todos los datos de la plataforma.
3. Se desarrollará tanto la parte pública como la parte privada de la plataforma web.
4. Se implementará el registro de datos durante la realización de los experimentos.
5. Se realizará el procesamiento de los datos registrados para generar variables.

Pruebas. Se realizarán pruebas sobre el software para validar que sus funcionalidades son correctas.

Redacción de la memoria. Esta tarea recoge la realización de los documentos finales del proyecto.

Planificación temporal



	Modo de	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1		**TFM**	**168 días**	**lun 09/11/20**	**vie 30/07/21**	
2		**Análisis**	**15 días**	**lun 09/11/20**	**vie 27/11/20**	
3		Análisis de los requisitos del proyecto	1 sem	lun 09/11/20	vie 13/11/20	
4		Análisis de aplicaciones similares	1 sem	lun 16/11/20	vie 20/11/20	3
5		Análisis de tecnologías web	1 sem	lun 23/11/20	vie 27/11/20	4
6		Fin del análisis	0 días	vie 27/11/20	vie 27/11/20	5
7		**Diseño e implementación**	**130 días**	**lun 30/11/20**	**mar 29/06/21**	**2**
8		Diseño y desarrollo de los nodos que conformarán la solución de los problemas	4 sem.	lun 30/11/20	mar 26/01/21	
9		Diseño y creación de la base de datos	2 sem.	mié 27/01/21	mar 09/02/21	8
10		Desarrollo de la parte pública y privada de la plataforma web	16 sem.	mié 10/02/21	mar 01/06/21	9
11		Registro de datos durante la resolución de los problemas	3 sem.	mié 05/05/21	mar 25/05/21	10CC+60 días
12		Procesamiento de los datos recogidos	2 sem.	mié 16/06/21	mar 29/06/21	11CC+30 días
13		Versión beta de la plataforma web	0 días	mar 29/06/21	mar 29/06/21	12
14		**Pruebas**	**5 días**	**jue 01/07/21**	**mié 07/07/21**	**7FC+1 día**
15		Implementación de pruebas	1 sem	jue 01/07/21	mié 07/07/21	
16		Versión RC (Release Candidate)	0 días	mié 07/07/21	mié 07/07/21	15
17		**Redacción de la memoria**	**22 días**	**jue 01/07/21**	**vie 30/07/21**	**7FC+1 día**
18		Redacción del capítulo 1	1 día	jue 01/07/21	jue 01/07/21	
19		Redacción del capítulo 2	1 sem	vie 02/07/21	jue 08/07/21	18
20		Redacción del capítulo 3	1 sem	vie 09/07/21	jue 15/07/21	19
21		Redacción del capítulo 4	1 sem	vie 16/07/21	jue 22/07/21	20
22		Redacción del capítulo 5	1 sem	vie 23/07/21	jue 29/07/21	21
23		Redacción del capítulo 6	1 día	vie 30/07/21	vie 30/07/21	22
24		Fin de la memoria	0 días	vie 30/07/21	vie 30/07/21	23

Figura 3.1: Diagrama de Gantt

3.2.2. Estimación de Costes

Ahora que ya hemos realizado una planificación del proyecto, vamos a realizar una estimación económica.

Para ello, vamos a diferenciar entre recursos humanos y recursos materiales, que son las personas que trabajarán en el proyecto y los materiales requeridos para elaborarlo.

Recursos Humanos

Este proyecto será elaborado por una sola persona con el título de Ingeniería Multimedia, por tanto tendrá un coste de 12€/h trabajando un total de 720h.

Resumiendo:

Recursos Humanos	**Coste por hora (€/h)**	**Coste total**
Ingeniero Multimedia	12	8640
COSTE TOTAL RECURSOS HUMANOS		**8640**

Tabla 3.1: Coste de los Recursos Humanos

Recursos Materiales

En cuanto a los recursos materiales, podemos diferenciar entre hardware y software y consumibles.

Con respecto al hardware necesario para este proyecto:

Elemento	Coste (€)
Ordenador Personal	1200
Disco duro externo (2 TB)	200
COSTE TOTAL HARWARE	**1400**

Tabla 3.2: Coste del Hardware

Con respecto al software necesario, cabe destacar que el IDE utilizado es el editor de texto Visual Studio Code añadiéndole algunos plugin de soporte para Python. Visual Studio Code es una solución creada por Microsoft y es open source, por tanto su coste es de 0€.

Además, el sistema operativo usado en nuestro ordenador personal será Kubuntu 20.04.2LTS, una versión de Ubuntu que usa el software de KDE y su entorno de escritorio Plasma. Al ser un sistema operativo basado en Ubuntu, es un proyecto open source y, al igual que el IDE usado, su coste también será de 0€.

Por tanto, el coste de software será de 0€, pero aún nos que revisar el coste de los consumibles:

Elemento	Coste (€)
Electricidad consumida	100
Conexión a Internet Fibra óptica (50€/mes x 9 meses)	450
Material de oficina: papel, bolígrafos, etc.	50
COSTE TOTAL CONSUMIBLES	**600**

Tabla 3.3: Coste de los Consumibles

Resumen del Coste del Proyecto

Elemento	Coste (€)
Recursos Humanos	8640
Hardware	1400
Software	0
Consumibles	600
COSTE TOTAL PROYECTO	**10640**

Tabla 3.4: Coste Total del proyecto

Por tanto, el coste estimado del proyecto asciende a la cantidad de 10640€

3.3. Riesgos

Los riesgos que pueden tener los proyectos son situaciones que pueden impedir la elaboración de dicho proyecto y para evitar o reducir su impacto en la elaboración del proyecto, se desarrollan Planes de Mitigación y Contingencia.

A continuación, vamos a ver los riesgos que pueden afectar a este proyecto y qué medidas se han tomado para reducir su impacto.

3.3.1. Riesgo 1

Un retraso en una tarea produce retrasos en cascada en las tareas pendientes.

Este riesgo se produce cuando una tarea tarda en completarse más tiempo del previsto y otras tareas requieren que ésta esté completa para poder llevarlas a cabo. En este proyecto, por ejemplo, no se pueden registrar datos si no se ha desarrollado la interfaz de para la resolución del experimento.

Por ello, las medidas de mitigación que se han tomado son las siguientes:

1. Dejar un margen de tiempo en las tareas que más riesgo tienen de retrasarse.
2. Ampliar los plazos.

En caso de que las medidas de mitigación fallen, se harán horas extras para cumplir los plazos como medida de contingencia.

3.3.2. Riesgo 2

Los usuarios finales insisten en nuevos requisitos.

Este riesgo puede causar que el proyecto tenga una duración mayor de la prevista incluso pudiendo llegar a una duración infinita.

Las medidas de mitigación que se han tomado son las siguientes:

1. Hacer revisiones periódicas de los requisitos.
2. Marcar un tope de requisitos.
3. Ayudar al cliente a definir los requisitos.

Cuando estas medidas fallen, se tomarán las siguientes medidas de contingencia:

1. Admitir los requisitos ampliando el plazo del proyecto.
2. No admitir más requisitos y dejarlos para una posterior versión del proyecto.

3.3.3. Riesgo 3

La curva de aprendizaje para la nueva herramienta de desarrollo es más larga de lo esperado.

Este riesgo afecta al producto, ya que los problemas que puedan surgir al usar la herramienta que genera el producto puede retrasar la elaboración del proyecto.

Por ello, las medidas de mitigación que se han tomado son:

1. Leer el manual de desarrollo de la herramienta de desarrollo.
2. Realizar pruebas con la herramienta de desarrollo.
3. Crear copias de seguridad del proyecto o usar Git.

Si fallan estas medidas de mitigación, se aplicarán las siguientes medidas de contingencia:

1. Cambiar de herramienta de desarrollo.
2. Reinstalar la herramienta de desarrollo.

Capítulo 4

Análisis y Diseño

4.1. Análisis

4.1.1. Casos de Uso

A continuación se describirán los casos de uso indicando las interacciones entre el usuario y la aplicación mediante el uso de diagramas UML (Unified Modeling Language).

Además del diagrama, se adjuntará una tabla con la especificación de cada caso de uso, cuyo contenido será:

- **Identificador:** CU A00-F1
  - Siendo CU la abreviatura de Caso de uso, 00 el número de identificación del caso y F1 el flujo del caso. Cada número de identificación irá precedido de una letra que indicará el actor al cuál se refiere: 'A'(administrador), 'P'(profesor) y 'E'(estudiante o alumno).
- **Nombre:** Nombre del caso de uso.
- **Flujo:** Flujo del caso de uso a tratar.
- **Precondiciones:** Condiciones que se deben cumplir para que se realice el caso de uso.
- **Postcondiciones:** Condiciones que se cumplen posteriormente al caso de uso.
- **Secuencia:** Flujo de ejecución del caso de uso.

Los casos de uso CRUD comprenden las acciones 'Create', 'Read', 'Update' y 'Delete' y se pueden identificar mediante las palabras gestionar, administrar o cualquier otro sinónimo de estas. Cada una de estas acciones corresponderá a un flujo y, por tanto, usaremos la tabla descrita anteriormente para especificar cada acción.

Casos de Uso "Profesor"

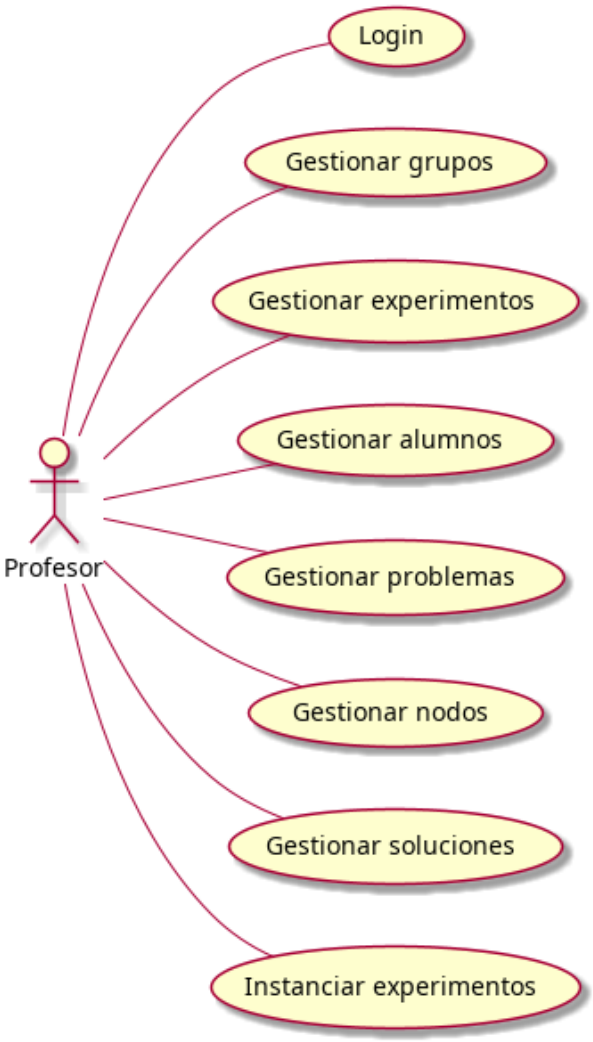


Figura 4.1: Casos de Uso Profesor

Identificador	CU P01-F1
Nombre	Login
Flujo	—
Precondiciones	—
Postcondiciones	—
Secuencia	1. El caso de uso comienza cuando el usuario hace clic en el botón "Acceso" y salta un pop-up con dos opciones de acceso y una de registro. 2. El usuario hace clic en el botón de "Acceso profesores". 3. El usuario introduce su email y contraseña. 4. El usuario hace clic en el botón de "Acceder". 5. El sistema verifica el usuario y la contraseña. 6. Se vuelve al paso 3 en caso de que el usuario y contraseña sean incorrectos o la cuenta no esté activada, en caso contrario el usuario accede a la aplicación. 7. El caso de uso termina.

Tabla 4.1: Descripción del Caso de Uso P01-F1

Identificador	CU P02-F1
Nombre	Gestionar Grupos
Flujo	Ver Grupo
Precondiciones	Login
Postcondiciones	—
Secuencia	1. El sistema presenta la lista de todos los grupos de la aplicación mediante una tabla. 2. El profesor clica el botón "Open" de la columna "Actions" del grupo deseado. 3. El sistema muestra un panel con la información del grupo y una lista con los alumnos de dicho grupo. 4. El caso de uso termina.

Tabla 4.2: Descripción del Caso de Uso P02-F1

Identificador	CU P02-F2
Nombre	Gestionar Grupos
Flujo	Editar Grupo
Precondiciones	Login, Ver grupo
Postcondiciones	—
Secuencia	1. El caso de uso comienza cuando el sistema muestra un panel con la información del grupo y una lista con los alumnos de dicho grupo y otra con las instancias de los experimento a realizar. 2. El profesor clica en el botón “Edit” del panel de información. 3. El sistema abre un pop-up con la información a modificar del grupo. 4. El profesor modifica la información deseada y le da al botón “Save” 5. El sistema modifica la información en la base de datos. 6. El caso de uso termina.

Tabla 4.3: Descripción del Caso de Uso P02-F2

Identificador	CU P02-F3
Nombre	Gestionar Grupos
Flujo	Eliminar Grupo
Precondiciones	Login, Ver grupo
Postcondiciones	—
Secuencia	1. El caso de uso comienza cuando el sistema muestra un panel con la información del grupo y una lista con los alumnos de dicho grupo y otra con las instancias de los experimento a realizar. 2. El profesor selecciona el botón “Delete” de la columna “Actions” del grupo deseado. 3. El sistema abre un pop-up para confirmar la eliminación de los datos de ese grupo. 4. El profesor confirma su eliminación. 5. El sistema elimina la información de ese grupo en la base de datos. 6. El caso de uso termina.

Tabla 4.4: Descripción del Caso de Uso P02-F3

Identificador	CU P02-F4
Nombre	Gestionar Grupos
Flujo	Crear Grupo
Precondiciones	Login
Postcondiciones	—
Secuencia	1. El caso de uso comienza cuando el sistema muestra un panel con la información del grupo y una lista con los alumnos de dicho grupo y otra con las instancias de los experimento a realizar. 2. El profesor clica en el botón “New group”. 3. El sistema abre un pop-up y muestra un formulario a rellenar con los datos a necesarios para crear el grupo. 4. El profesor rellena los datos con la información del grupo a crear y clica en el botón “Create”. 5. El sistema crea el grupo con los datos proporcionados. 6. El caso de uso termina.

Tabla 4.5: Descripción del Caso de Uso P02-F4

Identificador	CU P03-F1
Nombre	Instanciar experimento
Flujo	—
Precondiciones	Login, Ver experimento
Postcondiciones	—
Secuencia	1. El caso de uso comienza cuando el sistema muestra un panel con la información del grupo y una lista con los alumnos de dicho grupo y otra con las instancias de los experimento a realizar. 2. El profesor le da al botón “Instantiate experiment”. 3. El sistema abre un pop-up con una lista de los experimentos que se pueden instanciar. 4. El profesor selecciona el experimento que quiere instanciar. 5. El sistema crea una instancia del experimento seleccionado. 6. El caso de uso termina.

Tabla 4.6: Descripción del Caso de Uso P03-F1

Identificador	CU P04-F1
Nombre	Gestionar Alumnos
Flujo	Corregir experimento
Precondiciones	Login, Ver Grupo
Postcondiciones	—
Secuencia	1. El caso de uso comienza cuando el sistema muestra un panel con la información del grupo y una lista con los alumnos de dicho grupo y otra con las instancias de los experimento a realizar. 2. El profesor clica en el botón "Correct experiment" de la columna "Actions" del listado en el alumno deseado. 3. El sistema corrige todos los problemas de todos los experimentos que ha realizado el alumno y presenta el resultado. 4. El caso de uso termina.

Tabla 4.7: Descripción del Caso de Uso P04-F1

Identificador	CU P04-F2
Nombre	Gestionar Alumnos
Flujo	Eliminar Alumno
Precondiciones	Login, Ver Grupo
Postcondiciones	—
Secuencia	1. El caso de uso comienza cuando el sistema muestra un panel con la información del grupo y una lista con los alumnos de dicho grupo y otra con las instancias de los experimento a realizar. 2. El profesor clica en el botón "Delete" en la columna de "Acciones" del listado en el alumno deseado. 3. El sistema abre un pop-up para confirmar la eliminación del alumno. 4. El profesor confirma la eliminación del alumno. 5. El sistema elimina la información del alumno de la base de datos. 6. El caso de uso termina.

Tabla 4.8: Descripción del Caso de Uso P04-F2

Identificador	CU P04-F3
Nombre	Gestionar Alumnos
Flujo	Eliminar todos los alumnos
Precondiciones	Login, Ver Grupo
Postcondiciones	—
Secuencia	1. El caso de uso comienza cuando el sistema muestra un panel con la información del grupo y una lista con los alumnos de dicho grupo y otra con las instancias de los experimento a realizar. 2. El profesor clica en el botón “Delete all”. 3. El sistema abre un pop-up para confirmar la eliminación de todos los alumnos. 4. El profesor confirma la eliminación de todos los alumnos del grupo. 5. El sistema elimina de la base de datos la información de todos los alumnos del grupo. 6. El caso de uso termina.

Tabla 4.9: Descripción del Caso de Uso P04-F3

Identificador	CU P04-F4
Nombre	Gestionar Alumnos
Flujo	Crear Alumno
Precondiciones	Login, Ver Grupo
Postcondiciones	—
Secuencia	1. El caso de uso comienza cuando el sistema muestra un panel con la información del grupo y una lista con los alumnos de dicho grupo y otra con las instancias de los experimento a realizar. 2. El profesor clica en el botón “New student”. 3. El sistema muestra un formulario con el número de alumnos a crear y si deben tener la misma contraseña y usuario. 4. El profesor selecciona el número de alumnos a crear y si deben tener mismo usuario y contraseña o una generada de forma aleatoria. 5. El sistema crea el número de alumnos deseado. 6. El caso de uso termina.

Tabla 4.10: Descripción del Caso de Uso P04-F4

Identificador	CU P04-F5
Nombre	Gestionar alumnos
Flujo	Descargar datos de los alumnos
Precondiciones	Login, Ver Grupo
Postcondiciones	—
Secuencia	1. El caso de uso comienza cuando el sistema muestra un panel con la información del grupo y una lista con los alumnos de dicho grupo y otra con las instancias de los experimento a realizar. 2. El profesor clica en el botón “Student data”. 3. El sistema genera un archivo CSV con los datos de los alumnos del grupo. 4. El sistema descarga el archivo generado. 5. El profesor guarda el archivo CSV en su equipo. 6. El caso de uso termina.

Tabla 4.11: Descripción del Caso de Uso P04-F5

Identificador	CU P04-F6
Nombre	Gestionar alumnos
Flujo	Descargar resultados
Precondiciones	Login, Ver Grupo
Postcondiciones	—
Secuencia	1. El caso de uso comienza cuando el sistema muestra un panel con la información del grupo y una lista con los alumnos de dicho grupo y otra con las instancias de los experimento a realizar. 2. El profesor clica en el botón "Student results". 3. El sistema genera un archivo CSV con los resultados de los alumnos del grupo. 4. El sistema descarga el archivo generado. 5. El profesor guarda el archivo CSV en su equipo. 6. El caso de uso termina.

Tabla 4.12: Descripción del Caso de Uso P04-F6

Identificador	CU P05-F1
Nombre	Gestionar experimentos
Flujo	Ver experimento
Precondiciones	Login
Postcondiciones	—
Secuencia	1. El caso de uso comienza cuando el sistema presenta la lista de todos los experimentos del profesor mediante una tabla. 2. El profesor clica el botón "Open" de la columna "Actions" del experimento deseado. 3. El sistema muestra un panel con la información del experimento y una lista con los problemas de dicho experimento. 4. El caso de uso termina.

Tabla 4.13: Descripción del Caso de Uso P05-F1

Identificador	CU P05-F2
Nombre	Gestionar experimentos
Flujo	Editar experimento
Precondiciones	Login, Ver experimento
Postcondiciones	—
Secuencia	1. El caso de uso comienza cuando presenta un panel con la información del experimento y una lista con los problemas de dicho experimento. 2. El profesor clica en el botón “Edit” del panel de información. 3. El sistema abre un pop-up con la información a modificar del experimento. 4. El profesor modifica la información deseada y le da al botón “Save”. 5. El sistema modifica la información en la base de datos. 6. El caso de uso termina.

Tabla 4.14: Descripción del Caso de Uso P05-F2

Identificador	CU P05-F3
Nombre	Gestionar experimentos
Flujo	Crear experimento
Precondiciones	Login, Ver experimento
Postcondiciones	—
Secuencia	1. El caso de uso comienza cuando el sistema presenta la lista de todos los experimentos del profesor mediante una tabla. 2. El profesor clica en el botón “New experiment”. 3. El sistema abre un pop-up y muestra un formulario a rellenar con los datos a necesarios para crear el experimento. 4. El profesor rellena los datos con la información del experimento a crear y clica en el botón “Create”. 5. El sistema crea el experimento con los datos proporcionados. 6. El caso de uso termina.

Tabla 4.15: Descripción del Caso de Uso P05-F3

Identificador	CU P05-F4
Nombre	Gestionar experimentos
Flujo	Eliminar experimento
Precondiciones	Login
Postcondiciones	—
Secuencia	1. El caso de uso comienza cuando el sistema presenta la lista de todos los experimentos del profesor mediante una tabla. 2. El profesor clica en el botón "Delete" en la columna de "Actions" del listado en el experimento deseado. 3. El sistema abre un pop-up para confirmar la eliminación del experimento. 4. El profesor confirma la eliminación del experimento. 5. El sistema elimina la información del experimento de la base de datos. 6. El caso de uso termina.

Tabla 4.16: Descripción del Caso de Uso P05-F4

Identificador	CU P05-F5
Nombre	Gestionar experimentos
Flujo	Clonar experimento
Precondiciones	Login
Postcondiciones	—
Secuencia	1. El caso de uso comienza cuando el sistema presenta la lista de todos los experimentos del profesor mediante una tabla. 2. El profesor selecciona el botón "Clone" de la columna de "Actions" del experimento deseado. 3. El sistema realiza una copia de los datos de ese experimento. 4. El caso de uso termina.

Tabla 4.17: Descripción del Caso de Uso P05-F5

Identificador	CU P05-F6
Nombre	Gestionar experimentos
Flujo	Eliminar experimento
Precondiciones	Login
Postcondiciones	—
Secuencia	1. El caso de uso comienza cuando el sistema presenta la lista de todos los experimentos del profesor mediante una tabla. 2. El profesor selecciona el botón "Delete" de la columna de "Actions" del experimento deseado. 3. El sistema abre un pop-up para confirmar la eliminación de los datos de ese experimento. 4. El profesor confirma su eliminación. 5. El sistema elimina la información de ese experimento en la base de datos. 6. El caso de uso termina.

Tabla 4.18: Descripción del Caso de Uso P05-F6

Identificador	CU P06-F1
Nombre	Gestionar problemas
Flujo	Ver problema
Precondiciones	Login, Ver experimento
Postcondiciones	—
Secuencia	1. El caso de uso comienza cuando el sistema presenta la información de un experimento, mostrando una lista de los diferentes problemas mediante una tabla. 2. El profesor selecciona el botón "Open" de la columna de "Actions" del problema deseado. 3. El sistema presenta el enunciado del problema y la lista de nodos y soluciones que posee dicho problema. 4. El caso de uso termina.

Tabla 4.19: Descripción del Caso de Uso P06-F1

Identificador	CU P06-F2
Nombre	Gestionar problemas
Flujo	Editar enunciado
Precondiciones	Login, Ver problema
Postcondiciones	—
Secuencia	1. El caso de uso comienza cuando el sistema presenta el enunciado del problema y la lista de nodos y soluciones que posee dicho problema. 2. El profesor edita el enunciado del problema mediante el editor de texto y clica el botón "Save". 3. El sistema actualiza el enunciado en la base de datos. 4. El caso de uso termina.

Tabla 4.20: Descripción del Caso de Uso P06-F2

Identificador	CU P06-F3
Nombre	Gestionar problemas
Flujo	Crear problema
Precondiciones	Login, Ver experimento
Postcondiciones	—
Secuencia	1. El caso de uso comienza cuando el sistema presenta la información de un experimento, mostrando una lista de los diferentes problemas mediante una tabla. 2. El profesor clica en el botón "New problem". 3. El sistema crea un nuevo problema para el experimento. 4. El caso de uso termina.

Tabla 4.21: Descripción del Caso de Uso P06-F3

Identificador	CU P06-F4
Nombre	Gestionar problemas
Flujo	Eliminar problema
Precondiciones	Login, Ver experimento
Postcondiciones	—
Secuencia	1. El caso de uso comienza cuando el sistema presenta la información de un experimento, mostrando una lista de los diferentes problemas mediante una tabla. 2. El profesor selecciona el botón "Delete" de la columna de "Actions" del problema deseado. 3. El sistema abre un pop-up para confirmar la eliminación de los datos de ese problema. 4. El profesor confirma su eliminación. 5. El sistema elimina la información de ese problema en la base de datos. 6. El caso de uso termina.

Tabla 4.22: Descripción del Caso de Uso P06-F4

Identificador	CU P07-F1
Nombre	Gestionar nodos
Flujo	Crear nodo
Precondiciones	Login, Ver problema
Postcondiciones	—
Secuencia	1. El caso de uso comienza cuando el sistema presenta el enunciado del problema y la lista de nodos y soluciones que posee dicho problema. 2. El profesor selecciona el botón "New node". 3. El sistema abre un pop-up y muestra un formulario a rellenar con los datos del nodo. 4. El profesor rellena los datos con la información del nodo a crear y le da al botón "Save". 5. El sistema crea el nodo con los datos proporcionados. 6. El caso de uso termina.

Tabla 4.23: Descripción del Caso de Uso P07-F1

Identificador	CU P07-F2
Nombre	Gestionar nodos
Flujo	Editar nodo
Precondiciones	Login, Ver problema
Postcondiciones	—
Secuencia	1. El caso de uso comienza cuando el sistema presenta el enunciado del problema y la lista de nodos y soluciones que posee dicho problema. 2. El profesor clica en el botón “Edit” de la columna “Actions” de la tabla. 3. El sistema abre un pop-up con la información a modificar del experimento. 4. El profesor modifica la información deseada y le da al botón “Save”. 5. El sistema modifica la información en la base de datos. 6. El caso de uso termina.

Tabla 4.24: Descripción del Caso de Uso P07-F2

Identificador	CU P07-F3
Nombre	Gestionar nodos
Flujo	Eliminar nodo
Precondiciones	Login, Ver problema
Postcondiciones	—
Secuencia	1. El caso de uso comienza cuando el sistema presenta el enunciado del problema y la lista de nodos y soluciones que posee dicho problema. 2. El profesor selecciona el botón “Delete” de la columna de “Actions” del nodo deseado. 3. El sistema abre un pop-up para confirmar la eliminación de los datos de ese nodo. 4. El profesor confirma su eliminación. 5. El sistema elimina la información de ese nodo de la base de datos. 6. El caso de uso termina.

Tabla 4.25: Descripción del Caso de Uso P07-F3

Identificador	CU P08-F1
Nombre	Gestionar soluciones
Flujo	Crear solución
Precondiciones	Login, Ver problema
Postcondiciones	—
Secuencia	1. El caso de uso comienza cuando el sistema presenta el enunciado del problema y la lista de nodos y soluciones que posee dicho problema. 2. El profesor clica en el botón “New solution”. 3. El sistema carga una página con el enunciado del problema y los nodos que formarán la solución. 4. El profesor usa los nodos para formar la solución y le da al botón “Save solution”. 5. El sistema abre un pop-up con una caja de texto. 6. El profesor rellena la caja de texto con una descripción de la solución y le da al botón “Save”. 7. El sistema guarda la solución en la base de datos. 8. El caso de uso termina.

Tabla 4.26: Descripción del Caso de Uso P08-F1

Identificador	CU P08-F2
Nombre	Gestionar soluciones
Flujo	Editar solución
Precondiciones	Login, Ver problema
Postcondiciones	—
Secuencia	1. El caso de uso comienza cuando el sistema presenta el enunciado del problema y la lista de nodos y soluciones que posee dicho problema. 2. El profesor clica en el botón “Edit” de la columna “Actions” de la lista de soluciones. 3. El sistema carga una página con el enunciado del problema, los nodos que formarán la solución y la solución a editar. 4. El profesor usa los nodos para editar la solución y le da al botón “Save solution”. 5. El sistema abre un pop-up con una caja de texto. 6. El profesor edita el texto y le da al botón “Save”. 7. El sistema guarda la solución en la base de datos. 8. El caso de uso termina.

Tabla 4.27: Descripción del Caso de Uso P08-F2

Identificador	CU P08-F3
Nombre	Gestionar soluciones
Flujo	Eliminar solución
Precondiciones	Login, Ver problema
Postcondiciones	—
Secuencia	1. El caso de uso comienza cuando el sistema presenta el enunciado del problema y la lista de nodos y soluciones que posee dicho problema. 2. El profesor clica en el botón "Delete" de la columna "Actions" de la lista de soluciones. 3. El sistema abre un pop-up para confirmar la eliminación de la solución seleccionada. 4. El profesor confirma su eliminación. 5. El sistema elimina los datos de la solución de la base de datos. 6. El caso de uso termina.

Tabla 4.28: Descripción del Caso de Uso P08-F3

Identificador	CU P08-F4
Nombre	Gestionar soluciones
Flujo	Clonar solución
Precondiciones	Login, Ver problema
Postcondiciones	—
Secuencia	1. El caso de uso comienza cuando el sistema presenta el enunciado del problema y la lista de nodos y soluciones que posee dicho problema. 2. El profesor clica en el botón "Clone" de la columna "Actions" de la lista de soluciones. 3. El sistema clona la información de la solución. 4. El caso de uso termina.

Tabla 4.29: Descripción del Caso de Uso P08-F4

Casos de Uso "Administrador"

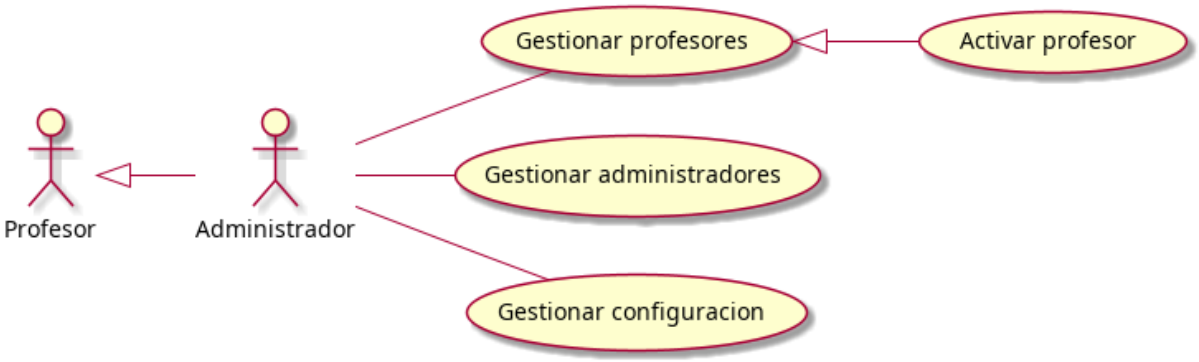


Figura 4.2: Casos de Uso Administrador

Identificador	CU A01-F1
Nombre	Gestionar profesores
Flujo	Crear profesor
Precondiciones	Login
Postcondiciones	—
Secuencia	1. El caso de uso comienza cuando el sistema presenta las lista de los profesores de la aplicación mediante una tabla. 2. El administrador clica en el botón "New manager". 3. El sistema abre un pop-up con un formulario a rellenar con la información del profesor a crear. 4. El administrador rellena el formulario con los datos del profesor y le da al botón "Create". 5. El sistema crea el profesor en la base de datos. 6. El caso de uso termina.

Tabla 4.30: Descripción del Caso de Uso A01-F1

Identificador	CU A01-F2
Nombre	Gestionar profesores
Flujo	Editar profesor
Precondiciones	Login
Postcondiciones	—
Secuencia	1. El caso de uso comienza cuando el sistema presenta las lista de los profesores de la aplicación mediante una tabla. 2. El administrador clica en el botón “Edit” de la columna “Actions” de la lista de profesores. 3. El sistema abre un pop-up con la información del profesor a modificar. 4. El administrador modifica los datos del profesor y le da al botón “Save”. 5. El sistema modifica el profesor en la base de datos. 6. El caso de uso termina.

Tabla 4.31: Descripción del Caso de Uso A01-F2

Identificador	CU A01-F3
Nombre	Gestionar profesores
Flujo	Eliminar profesor
Precondiciones	Login
Postcondiciones	—
Secuencia	1. El caso de uso comienza cuando el sistema presenta las lista de los profesores de la aplicación mediante una tabla. 2. El administrador clica en el botón “Delete” de la columna “Actions” de la lista de profesores. 3. El sistema abre un pop-up para confirmar la eliminación del profesor seleccionado. 4. El profesor confirma su eliminación. 5. El sistema elimina los datos del profesor de la base de datos. 6. El caso de uso termina.

Tabla 4.32: Descripción del Caso de Uso A01-F3

Identificador	CU A01-F4
Nombre	Gestionar profesores
Flujo	Activar profesor
Precondiciones	Login
Postcondiciones	—
Secuencia	1. El caso de uso comienza cuando el sistema presenta las lista de los profesores de la aplicación mediante una tabla. 2. El administrador clica en el botón “Activate” de la columna “Active” de la lista de profesores. 3. El sistema activa el profesor y guarda el estado en la base de datos. 4. El caso de uso termina.

Tabla 4.33: Descripción del Caso de Uso A01-F4

Identificador	CU A02-F1
Nombre	Gestionar administradores
Flujo	Crear administrador
Precondiciones	Login
Postcondiciones	—
Secuencia	1. El caso de uso comienza cuando el sistema presenta las lista de los administradores de la aplicación mediante una tabla. 2. El administrador clica en el botón “New manager”. 3. El sistema abre un pop-up con un formulario a rellenar con la información del administrador a crear. 4. El administrador rellena el formulario con los datos del administrador y le da al botón “Create”. 5. El sistema crea el administrador en la base de datos. 6. El caso de uso termina.

Tabla 4.34: Descripción del Caso de Uso A02-F1

Identificador	CU A02-F2
Nombre	Gestionar administradores
Flujo	Editar administrador
Precondiciones	Login
Postcondiciones	—
Secuencia	1. El caso de uso comienza cuando el sistema presenta las lista de los administradores de la aplicación mediante una tabla. 2. El administrador clica en el botón “Edit” de la columna “Actions” de la lista de administradores. 3. El sistema abre un pop-up con la información del administrador a modificar. 4. El administrador modifica los datos del administrador y le da al botón “Save”. 5. El sistema modifica el administrador en la base de datos. 6. El caso de uso termina.

Tabla 4.35: Descripción del Caso de Uso A02-F2

Identificador	CU A02-F3
Nombre	Gestionar administradores
Flujo	Eliminar administrador
Precondiciones	Login
Postcondiciones	—
Secuencia	1. El caso de uso comienza cuando el sistema presenta las lista de los administradores de la aplicación mediante una tabla. 2. El administrador clica en el botón “Delete” de la columna “Actions” de la lista de administradores. 3. El sistema abre un pop-up para confirmar la eliminación del administrador seleccionado. 4. El administrador confirma su eliminación. 5. El sistema elimina los datos del administrador de la base de datos. 6. El caso de uso termina.

Tabla 4.36: Descripción del Caso de Uso A02-F3

Identificador	CU A03-F1
Nombre	Gestionar configuración
Flujo	Crear configuración
Precondiciones	Login
Postcondiciones	—
Secuencia	1. El caso de uso comienza cuando el sistema presenta las lista de configuraciones de la aplicación mediante una tabla. 2. El administrador clica en el botón “New Key”. 3. El sistema abre un pop-up con un formulario a rellenar con la información de la configuración a crear. 4. El administrador rellena el formulario con los datos de la configuración y le da al botón “Create”. 5. El sistema crea la configuración en la base de datos. 6. El caso de uso termina.

Tabla 4.37: Descripción del Caso de Uso A03-F1

Identificador	CU A03-F2
Nombre	Gestionar configuración
Flujo	Editar configuración
Precondiciones	Login
Postcondiciones	—
Secuencia	1. El caso de uso comienza cuando el sistema presenta las lista de configuraciones de la aplicación mediante una tabla. 2. El administrador clica en el botón “Edit” de la columna “Actions” de la lista de configuraciones. 3. El sistema abre un pop-up con la información de la configuración a modificar. 4. El administrador modifica los datos de la configuración y le da al botón “Save”. 5. El sistema modifica la configuración en la base de datos. 6. El caso de uso termina.

Tabla 4.38: Descripción del Caso de Uso A03-F2

Identificador	CU A03-F3
Nombre	Gestionar configuración
Flujo	Eliminar configuración
Precondiciones	Login
Postcondiciones	—
Secuencia	1. El caso de uso comienza cuando el sistema presenta las lista de configuraciones de la aplicación mediante una tabla. 2. El administrador clica en el botón “Delete” de la columna “Actions” de la lista de configuraciones. 3. El sistema abre un pop-up para confirmar la eliminación de la configuración seleccionado. 4. El administrador confirma su eliminación. 5. El sistema elimina los datos de la configuración de la base de datos. 6. El caso de uso termina.

Tabla 4.39: Descripción del Caso de Uso A03-F3

Casos de Uso “Alumno”

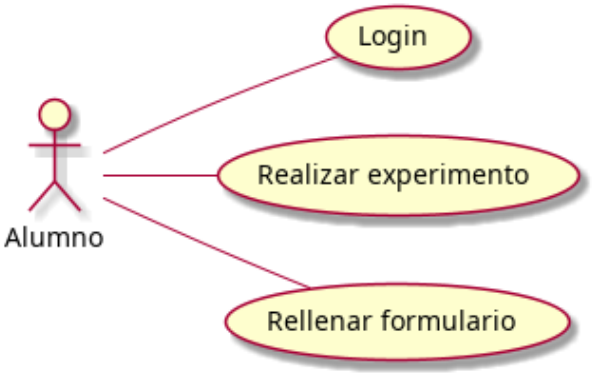


Figura 4.3: Casos de Uso Alumno

Identificador	CU E01-F1
Nombre	Login
Flujo	—
Precondiciones	—
Postcondiciones	—
Secuencia	1. El caso de uso comienza cuando el usuario hace clic en el botón “Acceso” y salta un pop-up con dos opciones de acceso y una de registro. 2. El usuario hace clic en el botón de “Acceso estudiantes”. 3. El usuario introduce su usuario y contraseña y hace clic en el botón de “Acceder”. 4. El sistema verifica el usuario y la contraseña. 5. Se vuelve al paso 3 en caso de que el usuario y contraseña sean incorrectos o la cuenta no esté activada, en caso contrario el sistema verifica si el alumno ya ha realizado el experimento activo de su grupo o si no hay ninguno activado. 6. Se vuelve al paso 3 en caso de que el alumno ya ha realizado el experimento activo de su grupo o si no hay ninguno activado, en caso contrario el sistema carga el experimento activo. 7. El caso de uso termina.

Tabla 4.40: Descripción del Caso de Uso E01-F1

Identificador	CU E02-F1
Nombre	Rellenar formulario
Flujo	—
Precondiciones	Login
Postcondiciones	—
Secuencia	1. El caso de uso comienza cuando el sistema muestra el formulario del experimento que debe realizar el alumno. 2. El alumno rellena el formulario y clica en el botón “Siguiente”. 3. El sistema guarda los datos del formulario en la base de datos. 4. El caso de uso termina.

Tabla 4.41: Descripción del Caso de Uso E02-F1

Identificador	CU E03-F1
Nombre	Realizar experimento
Flujo	—
Precondiciones	Login
Postcondiciones	—
Secuencia	1. El caso de uso comienza cuando el sistema muestra el botón "Empezar experimento". 2. El sistema abre una ventana y carga el primer problema a resolver del experimento. 3. El alumno resuelve el problema y clica en el botón "Enviar solución". 4. El sistema guarda la solución y comprueba que hayan más problemas a resolver en el experimento. 5. Si hay más problemas a resolver el sistema carga el siguiente problema y volvemos al paso 3, en caso contrario se cierra la venta y termina el experimento. 6. El caso de uso termina.

Tabla 4.42: Descripción del Caso de Uso E03-F1

4.1.2. Diagramas de Actividades

Un diagrama de actividades nos permite representar de forma gráfica un algoritmo o proceso indicando los pasos que se deben seguir. Gracias a este tipo de diagramas se describirán dos de los algoritmos o procesos más importantes que se han desarrollado para esta plataforma: realizar un experimento y corregirlo.

Realizar experimento

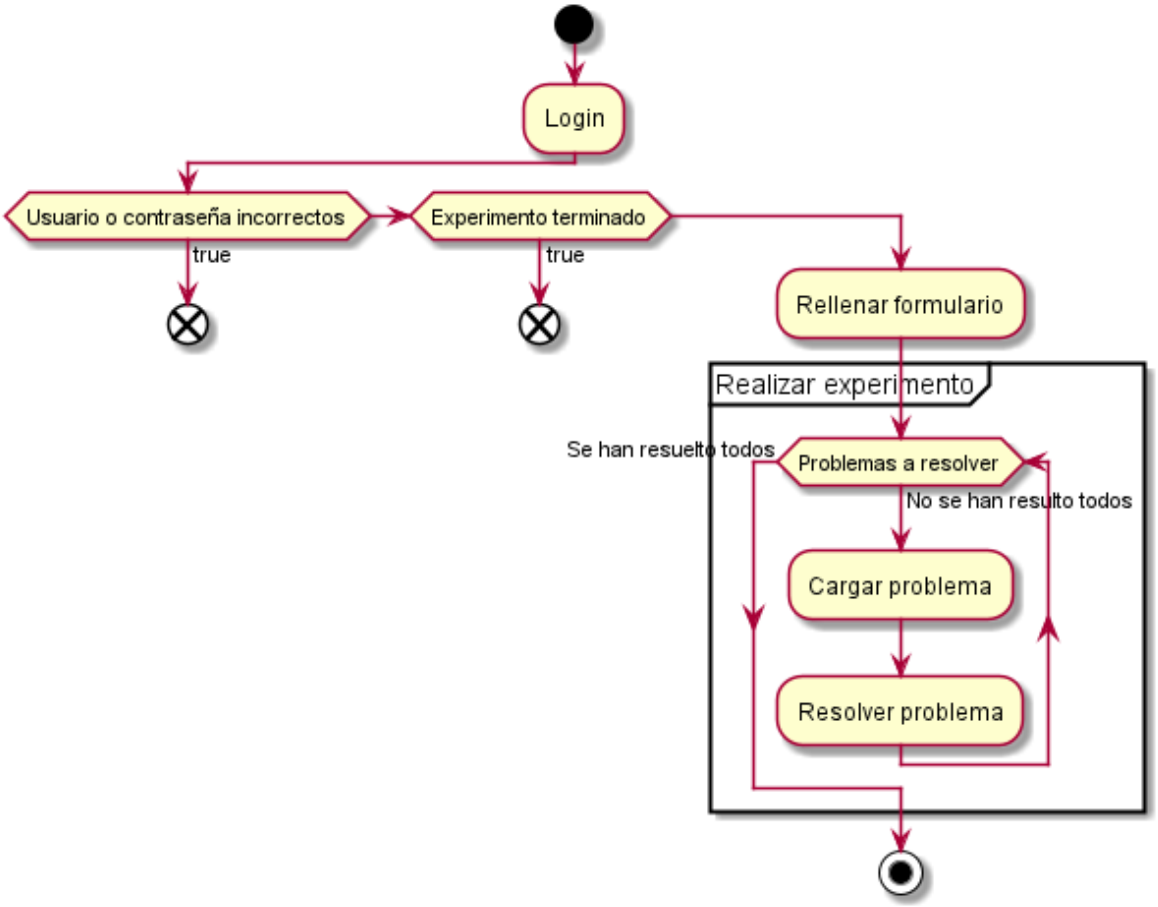


Figura 4.4: Diagrama de actividades de realizar experimento

Primero el estudiante inicia sesión en la plataforma con su usuario y contraseña y el sistema procede a verificar las credenciales. En caso de que las credenciales sean incorrectas el sistema no creará la sesión para el estudiante y redirigirá al usuario a la página inicial, mostrando el mensaje "Usuario o contraseña incorrectos".

En caso de que las credenciales sean correctas, el sistema verificará si el estudiante puede realizar el experimento. Si en el grupo al que pertenece el estudiante no hay ningún experimento instanciado, los que haya instanciados no están activos o ya ha realizado ese experimento, el sistema seguirá sin crear la sesión y al igual que en el caso anterior, redirigirá al usuario a la página inicial, pero mostrando el mensaje "Experimento terminado".

En caso de que aún no haya realizado el experimento, el sistema creará la sesión para el estudiante y mostrará un formulario que debe ser rellenado por el alumno. Una vez

rellenado el formulario, el sistema guardará esos datos en la base de datos y cargará los problemas del experimento que debe resolver el estudiante.

Cuando el sistema muestre el primer problema, el alumno podrá resolverlo. Cuando tenga la solución, el estudiante enviará la respuesta al sistema que la guardará en la base de datos y cargará el siguiente problema del experimento. Esto se repetirá hasta que el alumno resuelva el último problema y entonces terminará el experimento y se cerrará la sesión del estudiante.

Corregir experimento

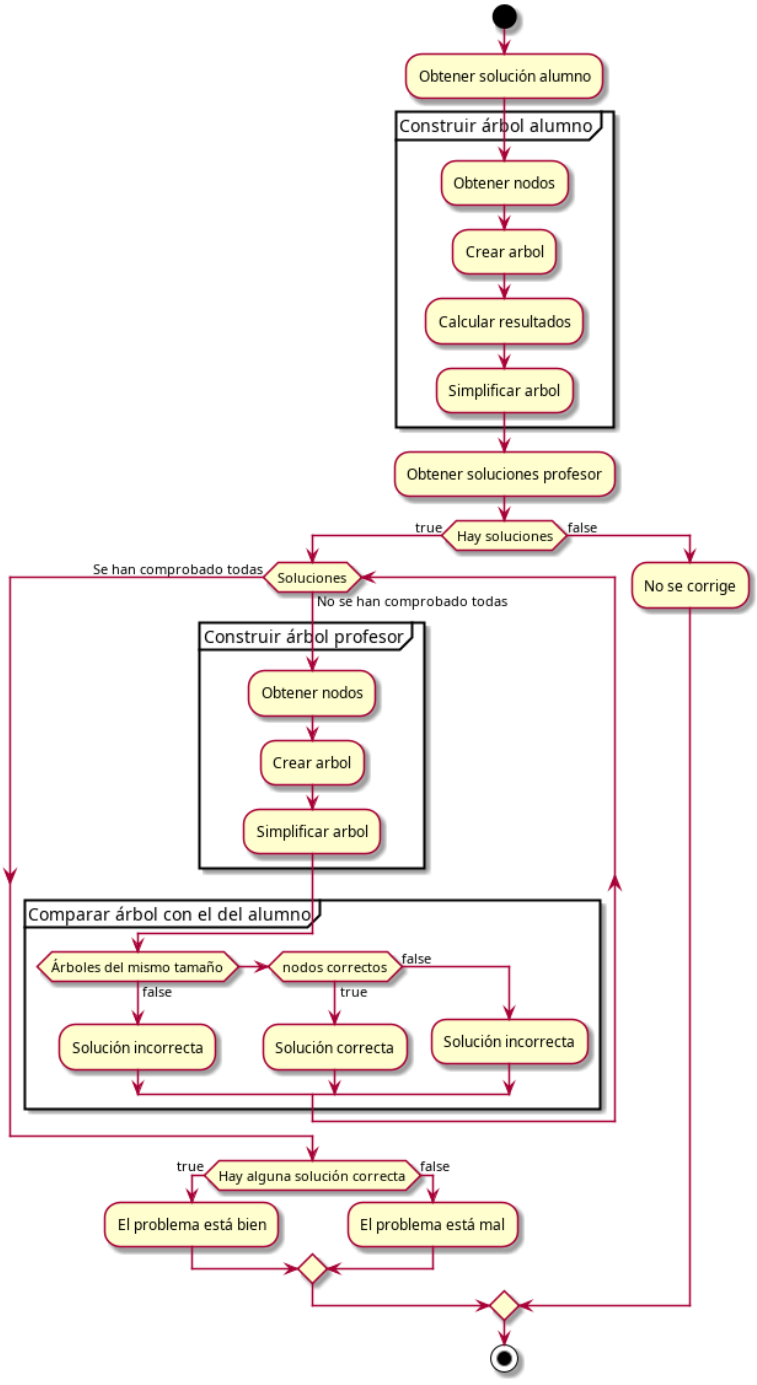


Figura 4.5: Diagrama de actividades de corregir problema

Para corregir un experimento, se deben corregir todos los problemas de ese experimento. El diagrama de la figura 4.5 nos muestra el proceso de corrección de un problema, y, si repetimos este proceso para todos los problemas de un experimento, tendremos el experimento corregido.

Por tanto, para corregir un problema, primero debemos obtener la solución que ha hecho el alumno y con ella reconstruiremos el árbol mediante la librería `treelib`[18].

Para construir el árbol, primero obtenemos los nodos que componen la solución y los procesaremos, identificando los nodos padres e hijos, creando así el árbol. Una vez tengamos el árbol, lo recorremos en anchura en orden inverso, es decir, lo recorreremos en anchura de hijos a padres, y calculamos el resultado de las operaciones. Por último, simplificamos el árbol, eliminando los nodos que se han utilizado para realizar los cálculos anteriores, es decir, los nodos de tipo "Valor".

Ahora que ya hemos obtenido el árbol realizado por el alumno, obtendremos, en una lista, las diferentes soluciones aportadas por el profesor. En caso de que la lista de soluciones del profesor esté vacía, es decir, en caso de que el profesor no haya aportado ninguna solución, el proceso de corrección terminará y el problema queda sin corregir.

En caso de que la lista de soluciones no esté vacía, para cada solución se construirá el árbol con la solución del profesor y se comparará con el árbol del alumno.

Para construir el árbol del profesor, se realiza de forma similar a la construcción del árbol del alumno. Se obtienen los diferentes nodos que componen el árbol y se construye con ellos. Una vez ya tenemos el árbol, lo simplificamos eliminando los nodos de tipo "Valor".

Cuando ya hemos creado el árbol con la solución del profesor, lo comparamos con el árbol del alumno. Primero comprobamos que los árboles tengan el mismo tamaño, en caso de que no tengan el mismo tamaño se almacenará en una lista de resultados que esta solución no es correcta. En caso contrario, compararemos uno a uno los nodos del árbol del alumno con los del profesor.

Si para cada nodo del árbol del alumno, el dato y el dato calculado coinciden y, además, el dato del nodo del árbol del alumno coincide con el dato del nodo del árbol del profesor, se considerará que la solución del alumno es correcta y se almacenará en la lista de resultados mencionada anteriormente como respuesta correcta. En caso de que haya algún nodo del alumno que no coincida con el del profesor, se considerará que la solución no es correcta y se almacenará en la lista de resultados.

Cuando se hayan comprobado todas las soluciones aportadas por el profesor, se comprobará si en esa lista de resultados hay alguna solución se que se haya marcado como correcta. Si hay una o más soluciones correctas, se determinará que el problema está bien y, en caso contrario, si no hay ninguna solución correcta, se determinará que el problema está mal.

Este proceso se realizará para todos los problemas de cada experimento a corregir.

4.2. Diseño

4.2.1. Diseño de la base de datos

Modelo Lógico

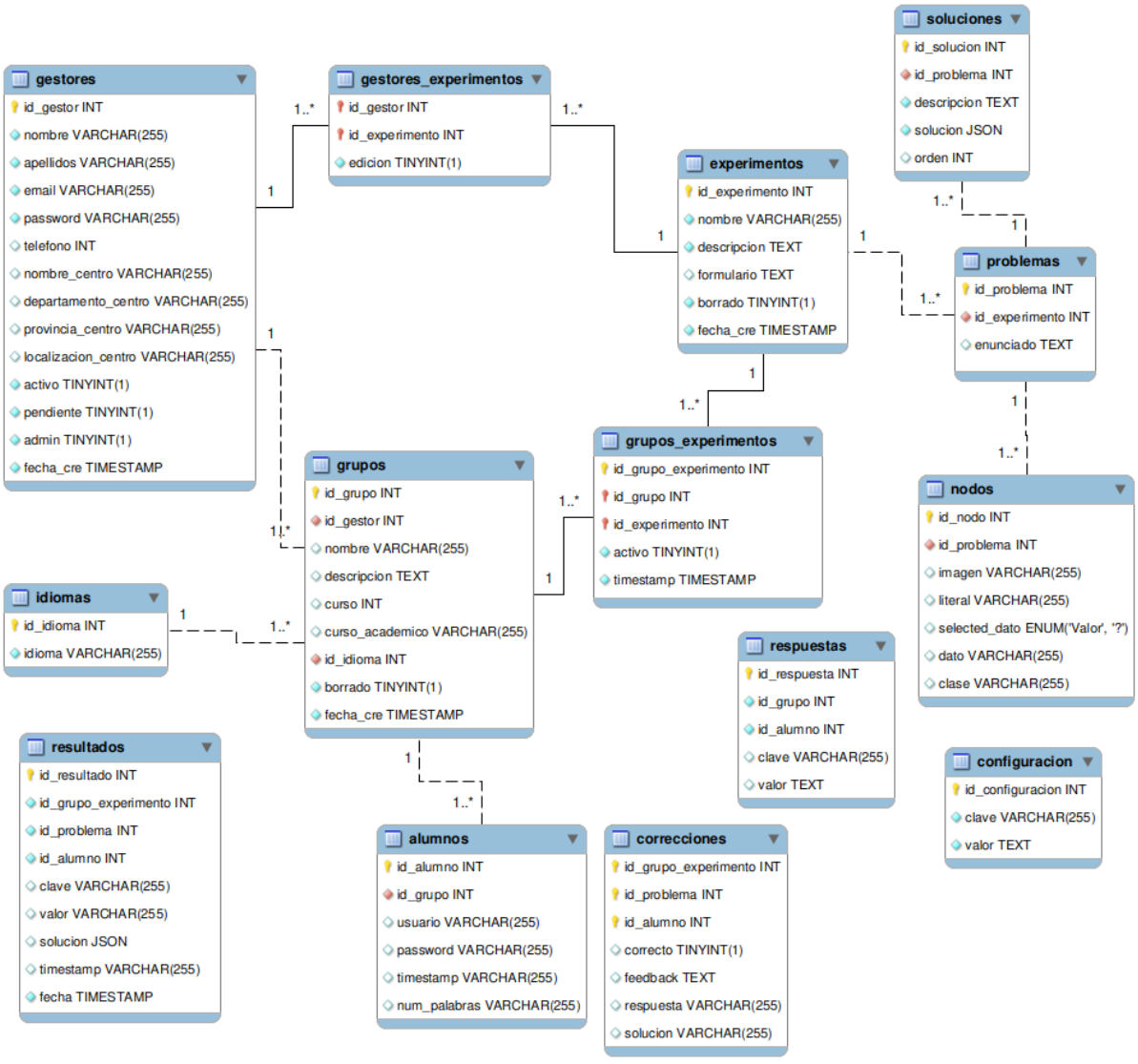


Figura 4.6: Modelo lógico de la Base de Datos

Descripción textual del modelo lógico

Gestores		
Describe la información de un gestor (profesor o administrador) de la aplicación		
Atributo	**Dominio**	**Descripción**
id_gestor	Entero	Índice de la tabla
nombre	Texto	Nombre del gestor
apellidos	Texto	Apellidos del gestor
email	Texto	Email del gestor
password	Texto	Contraseña del gestor
telefono	Entero	Teléfono del gestor
nombre_centro	Texto	Nombre del centro donde trabaja el gestor
departamento_centro	Texto	Departamento del centro donde trabaja el gestor
provincia_centro	Texto	Provincia del centro donde trabaja el gestor
localizacion_centro	Texto	Localización del centro donde trabaja el gestor
activo	Booleano	Indica si la cuenta del gestor está activada
pendiente	Booleano	Indica si la cuenta del gestor está pendiente de activar
admin	Booleano	Indica si el gestor es de tipo profesor o administrador
fecha_cre	Fecha	Fecha en la cual se crea el gestor
Relaciones		
Un gestor es propietario de distintos grupos y experimentos		

Tabla 4.43: Descripción textual de la tabla *gestores*

Idiomas		
Contiene los distintos idiomas de la plataforma		
Atributo	**Dominio**	**Descripción**
id_idioma	Entero	Índice de la tabla
idioma	Texto	Idioma de la plataforma
Relaciones		
El idioma está asociado a los grupos		

Tabla 4.44: Descripción textual de la tabla *idiomas*

Grupos		
Describe los distintos grupos creados por los gestores		
Atributo	**Dominio**	**Descripción**
id_grupo	Entero	Índice de la tabla
id_gestor	Entero	Gestor propietario del grupo
nombre	Texto	Nombre del grupo
descripción	Texto	Descripción del grupo
curso	Entero	Curso al que pertenece el grupo
curso_academico	Texto	Curso académico al que pertenece el grupo
id_idioma	Entero	Idioma del grupo
borrado	Booleano	Indica si el grupo se ha borrado
fecha_cre	Fecha	Fecha en la cual se creó el grupo
Relaciones		
El grupo pertenece a un gestor, contiene a los alumnos, decide en qué idioma realiza el alumno el experimento y crea las instancias de los experimentos		

Tabla 4.45: Descripción textual de la tabla *grupos*

Experimentos		
Describe los distintos experimentos que realizarán los alumnos y que son creados por los gestores		
Atributo	**Dominio**	**Descripción**
id_experimento	Entero	Índice de la tabla
nombre	Texto	Nombre del experimento
descripción	Texto	Descripción del experimento
formulario	Texto	Contiene el formulario que deben rellenar los alumnos
borrado	Booleano	Indica si el experimento se ha borrado
fecha_cre	Fecha	Fecha en la cual se creó el experimento
Relaciones		
Los experimentos son creados por los gestores y se crean instancias, que son las que usan los grupos		

Tabla 4.46: Descripción textual de la tabla *experimentos*

Gestores Experimentos		
Contiene los experimentos que pueden usar los gestores		
Atributo	**Dominio**	**Descripción**
id_gestor	Entero	Gestor del experimento
id_experimento	Entero	Experimento del gestor
edicion	Booleano	Indica si se puede editar el experimento
Relaciones		
Relaciona los gestores con los experimentos		

Tabla 4.47: Descripción textual de la tabla *gestores_ experimentos*

Grupos Experimentos		
Contiene las instancias de los experimentos		
Atributo	**Dominio**	**Descripción**
id_grupo_experimento	Entero	Asegura la unicidad de la tabla
id_grupo	Entero	Grupo que usa el experimento
id_experimento	Entero	Experimento a realizar
activo	Texto	Indica si la instancia está activa
timestamp	Fecha	Fecha de creación de la instancia
Relaciones		
Relaciona los grupos con los experimentos		

Tabla 4.48: Descripción textual de la tabla *grupos_ experimentos*

Alumnos		
Describe la información de un alumno		
Atributo	**Dominio**	**Descripción**
id_alumno	Entero	Índice de la tabla
id_grupo	Entero	Grupo al que pertenece el alumno
usuario	Texto	Usuario del alumno
password	Texto	Contraseña del alumno
timestamp	Texto	Tiempo de lectura de los experimentos en milisegundos
num_palabras	Texto	Número de palabras leídas
Relaciones		
Un alumno pertenece a un grupo		

Tabla 4.49: Descripción textual de la tabla *alumnos*

Problemas		
Describe la información de un problema		
Atributo	**Dominio**	**Descripción**
id_problema	Entero	Índice de la tabla
id_experimento	Entero	Experimento al que pertenece el problema
enunciado	Texto	Enunciado del problema
Relaciones		
Los problemas pertenecen a un experimento y cada problema tiene varias soluciones y varios nodos con los cuales crear las soluciones		

Tabla 4.50: Descripción textual de la tabla *problemas*

Soluciones		
Describe la información de una solución		
Atributo	**Dominio**	**Descripción**
id_solucion	Entero	Índice de la tabla
id_problema	Entero	Problema al que pertenece la solución
descripcion	Texto	Descripción de la solución
solucion	JSON	Solución del problema
orden	Entero	Orden de las soluciones
Relaciones		
Cada solución pertenece a un problema		

Tabla 4.51: Descripción textual de la tabla *soluciones*

Nodos		
Describe la información de un nodo		
Atributo	**Dominio**	**Descripción**
id_nodo	Entero	Índice de la tabla
id_problema	Entero	Problema al que pertenece el nodo
imagen	Texto	Nombre de la imagen del nodo
literal	Texto	Literal del nodo
selected_dato	Lista de Valores	Los posibles valores son 'Valor'y '?'
dato	Texto	Dato del nodo
clase	Texto	Nombre o tipo del nodo
Relaciones		
Cada nodo pertenece a un problema		

Tabla 4.52: Descripción textual de la tabla *nodos*

Resultados		
Contiene el registro de los datos del alumno		
Atributo	**Dominio**	**Descripción**
id_resultado	Entero	Índice de la tabla. Asegura la unicidad de los datos
id_grupo_experimento	Entero	Instancia del experimento a resolver
id_problema	Entero	Problema a resolver
id_alumno	Entero	Alumno que realiza el experimento
clave	Texto	Nombre del dato que se está registrando
valor	Texto	Valor del dato que se está registrando
solucion	JSON	Solución del problema hecha por el alumno
timestamp	Texto	Fecha de registro en milisegundos
fecha	Fecha	Fecha de registro
Relaciones		
Está relacionada de forma indirecta con las instancias, los problemas y los alumnos		

Tabla 4.53: Descripción textual de la tabla *resultados*

Respuestas		
Contiene las respuestas del formulario rellenado por los alumnos		
Atributo	**Dominio**	**Descripción**
id_respuesta	Entero	Índice de la tabla. Asegura la unicidad de los datos
id_grupo	Entero	Grupo al que pertenece el alumno que rellena los datos
id_alumno	Entero	Alumno que rellena los datos
clave	Texto	Dato del formulario que se almacena
valor	Texto	Valor del dato del formulario que se almacena
Relaciones		
Se relaciona de forma indirecta con los grupos y alumnos		

Tabla 4.54: Descripción textual de la tabla *respuestas*

Correcciones		
Contiene la corrección de los experimentos que realizan los alumnos		
Atributo	**Dominio**	**Descripción**
id_grupo_experimento	Entero	Instancia del experimento
id_problema	Entero	Problema que resuelto por el alumno
id_alumno	Entero	Alumno que resuelve el problema
correcto	Booleano	Indica si el problema corregido está bien o está mal
feedback	Texto	Anotaciones hechas por el profesor
respuesta	Texto	Valor del resultado final del problema
solucion	Texto	Indica con cual de las soluciones coincide
Relaciones		
Se relaciona de forma indirecta con las instancias de los experimentos, los problemas y los alumnos		

Tabla 4.55: Descripción textual de la tabla *correcciones*

Configuración		
Contiene los distintos parámetros de configuración de la aplicación		
Atributo	**Dominio**	**Descripción**
id_configuracion	Entero	Índice de la tabla
clave	Texto	Clave de configuración
valor	Texto	Valor de la configuración
Relaciones		
No está relacionada con ninguna tabla		

Tabla 4.56: Descripción textual de la tabla *configuracion*

4.2.2. Diseño del modelo de datos

Para el modelo de datos usaremos un ORM (Object Relational Model) que nos permitirá mapear clases (en este caso clases de python) con las tablas de la base de datos. Por tanto, al usar un ORM, podemos decir que el propio modelo lógico de la base de datos, representado en la figura 4.6, nos sirve como diagrama de clases para el modelo de datos.

Sin embargo, Flask, que es el microframework de python que se usará para este proyecto, no tiene un ORM propio, por lo que se usará SQLAlchemy mediante la extensión Flask-SQLAlchemy.

SQLAlchemy nos proporciona un ORM y un conjunto de herramientas para SQL usando python como lenguaje de programación. Entre las diferentes herramientas que nos proporciona se encuentra la conexión con la base de datos, el manejo de pools, la simplificación de las consultas SQL, etc.

A pesar de todo lo que nos proporciona SQLAlchemy, herramientas como el manejo del pool de conexiones son delicadas de manejar. Por ello, el plugin Flask-SQLAlchemy se centra en simplificar estas tareas relegándolas a que las maneje el propio microframework.

En resumen, el plugin Flask-SQLAlchemy nos proporciona las herramienta de SQLAlchemy, pero la conexión con la base de datos, la gestión del pool de conexiones y el manejo de las sesiones será llevado a cabo por Flask y libera al programador de realizar estas tareas que son tan delicadas.

En cuanto a cómo se crea el modelo de datos, según la documentación de Flask-SQLAlchemy se deberá crear un objeto de la clase "SQLAlchemy" y luego se incorporará la aplicación de Flask y en el parámetro de configuración "SQLALCHEMY_DATABASE_URI" se incluirá la URI de la base de datos de la siguiente forma:

```
app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'mysql+pymysql://user:password@IP/
    database?charset=utf8'
db = SQLAlchemy()
db.init_app(app=app)
```

Siendo user y password, el usuario y contraseña de acceso a la base de datos, IP, la dirección IP del servidor donde se encuentra la base de datos, y database, la base de datos a la cual se quiere acceder.

Una vez se ha creado el objecto "db", nos proporciona la clase "Model" que deberán heredar las clases del modelo para que se consideren parte del ORM, y, mediante el atributo "__tablename__" se indicará a qué tabla de la base de datos hace referencia cada clase.

En cuanto a las columnas de la base de datos, se referenciarán mediante la clase "Column" que nos proporciona SQLAlchemy, donde indicaremos la columna de la tabla a la cual se refiere, el tipo de dato de la columna y otros datos como si la columna puede ser nula, por ejemplo.

En cuanto al mapeo de las relaciones de la base de datos, se deben crear la referencia a todas las columnas de la tabla incluidas las columnas que son clave ajena. Las columnas que son clave ajena se indicará mediante la clase "ForeingKey" de SQLAlchemy a qué columna de qué tabla hace referencia.

Por ejemplo, la clase "Alumno" tiene relación con la clase "Grupo" y por ello, la columna "id_grupo" de la tabla *alumnos* es clave ajena de la tabla *grupos*. Esto se indica de la siguiente forma:

```
id_grupo = Column('id_grupo', Integer, ForeignKey('grupos.id_grupo', ondelete='
    CASCADE', onupdate='CASCADE'), nullable=False)
```

Una vez ya hemos indicado todas las claves ajenas en la clase, deberemos indicar, mediante la función "relationship" de SQLAlchemy, la relación entre las clases, siendo en el ejemplo anterior, la relación entre grupos y alumnos.

4.2.3. Diseño de los decoradores

Los decoradores son un patrón de diseño que permiten a una función (A) o clase (A) tomar otra función (B) como argumento para devolver una función (C). Es decir, nos

permiten reducir las líneas de código duplicadas, evitando implementar una funcionalidad múltiples veces.

Como ya se ha mencionado anteriormente, Flask únicamente incorpora lo esencial para crear aplicaciones web, y por ello, no incorpora el manejo de los usuarios, funcionalidad que es posible suplir con el plugin Flask-Login.

Aunque Flask-Login nos permite manejar las sesiones de los usuarios e incluso incorpora el decorador "login_required" que permite proteger las vistas para usuario no logeados, no incorpora una gestión de los roles de los distintos usuarios, que son los roles de profesor, administrador y alumno.

Por ello, se creará un decorador que sí maneje los roles, de forma que se proporcione una lista de los roles que se quieran comprobar y en caso de que el rol del usuario logueado no esté presente en esa lista el decorador devolverá el código 401, Unauthorized. En caso de que el rol sí que esté en la lista, sí que se permitirá al usuario acceder a la vista.

Por tanto, este decorador protegerá las vistas de los usuarios que no deban acceder y los usuarios solo podrán realizar las acciones que se le han especificado en sus casos de uso.

Además de este decorador tan importante, se usará otro decorador que sugiere la propia documentación de Flask y que lo llama "Templating decorator". Este decorador pretende simplificar la forma de renderizar una template.

Para renderizar una template se debe usar el método "render_template" de Flask y proporcionar el nombre de la template a renderizar y un diccionario con los datos que se deben utilizar en la template como argumento de la función.

Por tanto, la idea que propone este decorador es que se indique como argumento el nombre de la template y que la función de la vista devuelva únicamente el diccionario.

Para usar los dos decoradores descritos, se incluirán después del decorador "route" y antes de la función view. Por ejemplo:

```
@app.route('/')
@login_required
@roles_requires('admin','profesor')
@templated('home.html')
def index():
    return {
        'value': 42
    }
```

Capítulo 5

Implementación y pruebas

5.1. Implementación

5.1.1. Implementación del patrón MVC

El patrón MVC (Model-View-Controller) es uno de los patrones de diseño más utilizados ya que nos permite separar la lógica de negocio de la vista (cómo se muestran los datos), de forma que se pueda modificar la vista sin que afecte a la lógica de negocio y viceversa.

Por tanto la implementación de este proyecto seguirá el siguiente diagrama que seguirá el patrón MVC:

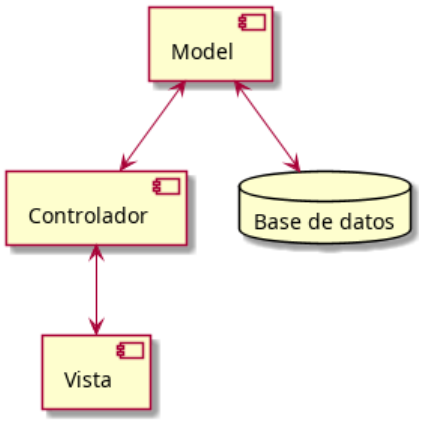


Figura 5.1: Patrón de diseño MVC

Según este diagrama, el modelo de datos se conectará a la base de datos para extraer la información, pero será el controlador el que demandará qué información debe ser extraída, cuál se deberá actualizar, eliminar, etc.

En cuanto a la vista, se corresponderán con las templates HTML que nos proporciona Flask mediante el motor de renderizado Jinja. Jinja nos permitirá injectar los datos que proporciona el controlador mediante las variables y tags.

Además, también nos permite tener una plantilla HTML que nos sirva como base para poder realizar todas las páginas HTML sin duplicar código, de esta forma mediante la

plantilla HTML podemos incluir todas las librerías JavaScript y todas las hojas de estilo CSS que necesitemos en la platilla y el resto de páginas heredarán de esta plantilla.

Para poder usar las páginas HTML heredadas, en la página HTML base debemos añadir el tag `{% block %}`, que definirán los bloques que se usarán en las páginas HTML hijas para escribir la información que corresponda en cada página.

Por último la vista se comunicará con el controlador mediante las URLs. Para generar las URLs, Flask nos proporciona el decorador `route`, donde indicaremos la URL y en el parámetro `methods`, una lista de los diferentes métodos HTTP (`GET`, `POST`, `PUT`, `PATCH` y `DELETE`) que queremos que acepte la URL. Además, en la URL puede haber variables que nos ayudarán a crear URLs dinámicas, por ejemplo:

```
@app.route('/index/', methods=['GET'])
def index():
    return 'index'

@app.route('/index2/<name>', methods=['POST'])
def index2(name):
    return name
```

En la primera función, mediante el decorador `route`, hemos indicado que su URL será `/index/` y solamente permitirá acceder a ella mediante el método `GET`.

En la segunda función, hemos indicado que su URL será `/index2/<name>`, donde `<name>` es un parámetro que recibe la función. Además solo podermos acceder a esta URL mediante el método `POST`.

A pesar de tener este decorador, en los ejemplos lo estamos usando sobre la aplicación principal, lo cual es insuficiente para grandes proyectos como este, donde tenemos que gestionar una parte de administración y otra parte pública, además de gestionar los errores (errores 404, 500, 403, etc.), peticiones AJAX y los datos que exportamos.

Por tanto, para poder dividir estas tareas y tener diversos controladores, Flask nos ofrece usar `Blueprints`, que nos permite registrar las URL de la misma forma que se ha hecho en los ejemplos. Un ejemplo de creación y uso de una blueprint sería el siguiente:

```
from flask import Flask, Blueprint

app = Flask(__name__)

controller = Blueprint('controller', __name__)

@controller.route('/', methods=['GET'])
def index():
    return 'index'

app.register_blueprint(controller)

if __name__ == '__main__':
    app.run(host='0.0.0.0')
```

5.1.2. Implementación del patrón Repository

El patrón repositorio (Repository Pattern) nos permite separar el acceso a la base de datos de la lógica de negocio de tal forma que, mediante una clase se realice una lógica de negocio sobre un objeto y otra clase sea la que modifique el ese objeto en la base de datos. Estos objetos que se modifican son los que forman el modelo de datos que es nuestro ORM. Por tanto, el patrón repository facilita la interacción del modelo de datos con la base de datos.

Al usar este patrón de diseño, el anterior diagrama 5.1 se actualizaría para incluir un nuevo componente repository. Además también incluiremos un componente de servicios (service) que es el que se encargará de la lógica de negocio:

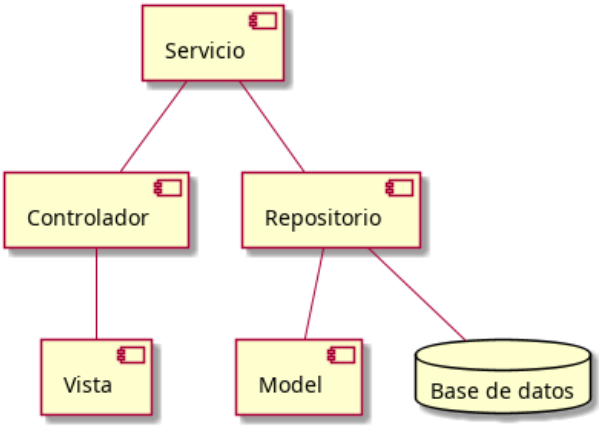


Figura 5.2: Patrón de diseño repository

Por tanto, ahora el repositorio es el que accede a los datos de la base de datos, el servicio se encarga de la lógica de negocio y el controlador accede a todos los servicios que sean necesarios para realizar las operaciones que correspondan y actualiza la vista.

Por ejemplo, si queremos editar la información de un alumno, el controlador recibe de la vista el identificador del alumno que quiere modifica y los datos que se quieren modificar. El controlador llama al servicio encargado de manejar la lógica de negocio de los alumnos y le manda la información que ha recibido de la vista.

Entonces, el servicio llama al repositorio del alumno y le dice que obtenga los datos del alumno cuyo identificador es el que le manda el controlador, luego modifica ese alumno con el resto de los datos que recibe del controlador y le vuelve a decir al repositorio que actualice el alumno en la base de datos.

Una vez el repositorio ha actualizado el alumno en la base de datos, el servicio le devuelve al controlador el alumno actualizado y éste actualiza la vista.

Sin embargo, dado que el ORM que usa Flask es el que nos proporciona SQLAlchemy, como ya se ha explicado en el apartado 4.2.2, y a pesar de que no nos proporciona ninguna clase que nos permita aplicar el patrón repository, sí que nos proporciona la manera de abstraer el lenguaje SQL para usar las clases del ORM.

```
a = Alumno()
a.usuario = 'user'
a.password = 'password'
```

```
db.session.add(a)
db.session.commit()

a = db.session.query(Alumno).get(1)

a.usuario = 'user1'
db.session.commit()

db.session.delete(a)
db.session.commit()
```

Partiendo de que existe una clase `Alumno` que forma parte de nuestro ORM, en las tres primeras líneas del ejemplo creamos un objeto de esta clase y le asignamos los atributos. En las dos siguientes líneas, teniendo en cuenta que `db` es un objeto de tipo `SQLAlchemy`, creamos un registro en la tabla de alumnos de la base de datos con los datos del objeto creado anteriormente.

En la siguiente línea recuperamos el alumno cuyo identificador de la tabla es 1. Después, modificamos el usuario y guardamos los cambios en la base de datos. Finalmente, eliminamos ese alumno de la base de datos.

Estas serías las cuatro operaciones básicas a realizar en una base de datos, pero SQLAlchemy nos proporciona también los métodos `filter_by` y `order_by`, a los que se les pasa como argumento un diccionario y una lista que representan las palabras clave SQL `WHERE` y `ORDER BY` respectivamente. Estos métodos se utilizan junto con los métodos `first` y `all` para obtener un resultado o una lista de resultados.

Gracias a todos estos métodos que nos ofrece SQLAlchemy, a pesar de no proporcionarnos una clase `Repository` como tal, nos permite crear una clase `Repository` propia.

En el apéndice A.1 se encuentra la clase `Repository` que se ha implementado usando los métodos que nos proporciona SQLAlchemy y nos permite realizar las operaciones CRUD (Create, Read, Update, Delete). Además se ha incluido el atributo `__model__` que representa la clase del ORM que se quiere usar, de forma que para acceder a los datos de la base de datos se deba crear una clase que herede esta.

```
from models import Alumno
from .repository import Repository

class AlumnoRepository(Repository):
    __model__ = Alumno
```

Como podemos ver en el código, la clase `AlumnoRepository` hereda la clase `Repository` y al indicar que el atributo `__model__` corresponde a la clase `Alumno`, todos .los métodos de la clase `Repository` actuarán sobre la clase `Alumno`.

Y, de esta forma, conseguimos implementar satisfactoriamente el patrón repository a pesar de que SQLAlchemy no nos proporciona una clase `Repository` como tal.

5.1.3. Registro de datos del experimento

Anteriormente, en el apartado 4.1.2 hablamos de la corrección del experimento y que para ello se debía obtener la solución de un problema que resolvía un alumno. Pues bien, para poder obtener este dato, realizaremos un registro o tracking del alumno durante la realización del experimento.

Además, este tracking nos permitirá registrar las acciones que realice el alumno, como el enmascaramiento de las proposiciones del enunciado o el uso de la calculadora que se les proporciona a los alumnos para poder realizar las operaciones matemáticas para resolver los problemas.

En cuanto al enmascaramiento de las proposiciones del enunciado, cuando el profesor redacte el enunciado, éstas se marcarán mediante un resaltado y por tanto cada proposición quedará marcada mediante un `span` con el atributo style y un color de fondo.

Por tanto, para poder generar el enmascarado usaremos el método de la clase descrito en el apéndice A.2 para eliminar el tag `style` y añadirle las clases `region`, `region-hide` y `region-p1`, siendo p1 la proposición 1.

Esta clase `region` de CSS hará que la proposición sea ilegible a menos que se haga clic sobre ella y se haga completamente legible. De esta forma conseguimos crear la funcionalidad del enmascaramiento y desenmascaramiento de las proposiciones del enunciado.

Ahora que ya sabemos cómo funciona el enmascaramiento de las proposiciones del problema, podemos registrar datos de esta funcionalidad y cada vez que se haga clic sobre una proposición se enviarán datos al servidor. Cuando se enmascare la proposición el servidor recibirá los datos del cliente mediante una petición AJAX.

Los datos que recibirá son datos de contexto como el identificador de la instancia del experimento que está realizando, el problema que está resolviendo y el alumno que lo resuelve. Además de los datos de contexto, recibirá una clave, que en este caso será `unmaskRegion` para en desenmascaramiento y `maskRegion` para el enmascarmiento de la proposición, un valor que corresponderá a la proposición en cuestión y el tiempo en el cual se envían los datos al servidor.

Para poder enviar estos datos, se van a utilizar las dos funciones JavaScript descritas en el apéndice A.3. La primera función obtiene el tiempo actual en milisegundos y la segunda envía al servidor datos mediante una petición `POST` usando la tecnología AJAX. Esta función es genérica y nos sirve para enviar todos los datos que se quieren registrar.

Además de registrar el enmascaramiento de las proposiciones, también queremos registrar las operaciones que realiza, por ello cada vez que realice un cálculo mediante la calculadora que se les proporciona a los alumnos, éste enviará datos al servidor mediante la función descrita anteriormente usando la clave `calculate` y como valor el resultado de la operación.

Por último, queremos registrar la solución del alumno y para ello, cuando el alumno le dé al botón de "Enviar solución", usando la función `sendData` del apéndice A.3, convertirá el grafo creado por el alumno en una variable JSON y se enviará al servidor junto con la clave `endProblem` y cada vez que el cliente cargue un problema se enviará la clave `startProblem`.

5.2. Pruebas funcionales

Python incorpora un framework de testing llamado `unittest`[19] que está inspirado en JUnit y es similar a los frameworks de testing de otros lenguajes.

Al usar este framework de testing, la estructura de nuestros tests y cómo crearlos será la siguiente:

1. Creamos una clase `Test<NombreDeLoQueSePrueba>` que hereda de `unittest.TestCase`
2. Definimos los tests que sean necesarios como métodos de la clase. El nombre de estos métodos tendrá `test_` como prefijo.
3. Cada test ejecuta las comprobaciones necesarias usando `assertEqual`, `assertTrue`, `assertFalse`, `assertIn` etc.

Por tanto, si, por ejemplo, tenemos la siguiente función:

```
def calcula_media(*args):
    return(sum(*args)/len(*args))
```

La clase que tendrá los tests para comprobar que esta función clacula la media de forma correcta será:

```
class TestCalculaMedia(unittest.TestCase):
    def test_1(self):
        resultado = calcula_media([10, 10, 10])
        self.assertEqual(resultado, 10)

    def test_2(self):
        resultado = calcula_media([5, 3, 4])
        self.assertEqual(resultado, 4)
```

Además, `TestCase` nos proporciona los métodos `setUp`, `tearDown`, `setUpClass` y `tearDownClass`. Estos métodos nos permiten crear variables para la clase del test que pueden ser usadas por todos los métodos.

Los métodos `setUp` y `tearDown` se ejecutan cada vez cada vez que se ejecuta un test de la clase, usando `setUp` para crear el contexto del test y `tearDown` para destruirlo.

En cuanto a los otros dos métodos, `setUpClass` y `tearDownClass`, funcionan de igual forma que los anteriores, salvo por el hecho de que el contexto se aplica a toda la clase, en vez de sólo a cada método.

Por tanto, al ejemplo anterior se le podrían añadir estas funciones de la siguiente forma:

```
class TestCalculaMedia(unittest.TestCase):
    def setUp(self):
        valores1 = [10, 10, 10]
        resultado1 = 10
        valores2 = [5, 3, 4]
        resultado2 = 4

    def test_1(self):
```

```
        resultado = calcula_media(self.valores1)
        self.assertEqual(resultado, self.resultado1)

    def test_2(self):
        resultado = calcula_media(self.valores2)
        self.assertEqual(resultado, self.resultado2)
```

Ahora que ya sabemos cómo crear las pruebas con este framework, lo usaremos para comprobar que el registro del enmascaramiento de las proposiciones del enunciado (enmascaramiento del texto o enmascaramiento, de ahora en adelante) se hace de forma correcta y también para comprobar que el corrector automático funciona.

5.2.1. Enmascaramiento del texto

Para comprobar que el enmascaramiento se registra de forma correcta, partimos del hecho de que se ha creado un experimento con un problema que tiene tres proposiciones. Además, también se ha creado un grupo con cinco alumnos y cada alumno (me referiré a cada uno por su usuario: 6_1, 6_2, 6_3, 6_4 y 6_5) tendrá un comportamiento distinto, antendiendo a cada una de las cinco pruebas que se van a realizar.

Prueba alumno 6_1

El alumno realizará el experimento y desenmascarará el texto de la siguiente forma:

1. El alumno hace clic en la proposición 1.
2. El alumno hace clic en la proposición 2.
3. El alumno hace clic en la proposición 3.

Estas acciones deberían registrarse en este orden con los valores 1, 2 y 3.

Prueba alumno 6_2

El alumno realizará en experimento y desenmascarará el texto de la siguiente forma:

1. El alumno hace clic en la proposición 1.
2. El alumno hace clic en la proposición 3.
3. El alumno hace clic en la proposición 1.
4. El alumno hace clic en la proposición 2.

Estas acciones deberían registrarse en este orden con los valores 1, 3, 1 y 2.

Prueba alumno 6_3

El alumno realizará el experimento y desenmascarará el texto de la siguiente forma:

1. El alumno hace clic en la proposición 1.
2. El alumno hace clic en la proposición 1.
3. El alumno hace clic en la proposición 3.
4. El alumno hace clic en la proposición 2.
5. El alumno hace clic en la proposición 1.
6. El alumno hace clic en la proposición 2.
7. El alumno hace clic en la proposición 2.

Estas acciones deberían registrarse en este orden con los valores 1, 3, 2, 1 y 2.

Prueba alumno 6_4

El alumno realizará el experimento y desenmascarará el texto, además de interactuar con el resto de la página web, de la siguiente forma:

1. El alumno hace clic en la proposición 1.
2. El alumno hace clic en la proposición 1.
3. El alumno hace clic en el resto de la página.
4. El alumno hace clic en la proposición 1.
5. El alumno hace clic en la proposición 2.
6. El alumno hace clic en el resto de la página.
7. El alumno hace clic en la proposición 2.
8. El alumno hace clic en la proposición 3.
9. El alumno hace clic en el resto de la página.

Estas acciones deberían registrarse de la siguiente forma:

Clave	**Valor**
unmaskRegion	1
maskRegion	
unmaskRegion	1
unmaskRegion	2
maskRegion	
unmaskRegion	2
unmaskRegion	3
maskRegion	

Tabla 5.1: Registro enmascarimento test alumno 6_4

Prueba alumno 6_5

El alumno realizará el experimento y desenmascarará el texto, además de interactuar con el resto de la página web y con el propio ordenador, de la siguiente forma:

1. El alumno hace clic en la proposición 1.
2. El alumno hace clic en la proposición 1.
3. El alumno hace clic en el resto de la página.
4. El alumno hace clic en la proposición 1.
5. El alumno minimiza la ventana del experimento.
6. El alumno abre una pestaña nueva en el navegador.
7. El alumno abre el explorador de archivos.
8. El alumno vuelve a la ventana del experimento.
9. El alumno hace clic en la proposición 2.
10. El alumno cambia de ventana.
11. El alumno vuelve a la venta del experimento y hace clic en la proposición 3.

Estas acciones deberían registrarse de la siguiente forma:

Clave	**Valor**
unmaskRegion	1
maskRegion	
unmaskRegion	1
maskRegion	
unmaskRegion	2
maskRegion	
unmaskRegion	3

Tabla 5.2: Registro enmascarimento test alumno 6_5

Conclusión

En el apéndice A.4 se implementa el test que comprueba que las acciones registradas son iguales a las acciones que se espera que se registren.

Al ejecutar los tests ninguno ha dado error, por lo tanto, podemos decir que el registro del enmascaramiento funciona de forma correcta.

5.2.2. Corrector automático

Para comprobar que el corrector automático de problemas funciona bien, partimos del hecho que se ha creado un experimento con seis problemas (me referiré a cada problema según su identificador: 17, 18, 19, 20, 21 y 22) y cada problema tiene una solución. Además, también se ha creado un grupo con un alumno que ha realizado el experimento y ha resulto todos los problemas y la solución aportada a cada problema es correcta.

Este test consta de dos partes, la creación de la solución aportada por el usuario que resuelve los problemas y la aplicación del algoritmo descrito en el diagrama 4.5 que es el que corrige los experimentos. A continuación, veremos el enunciado de cada problema, la solución que se ha aportado y cuál debería ser la respuesta del corrector.

Test problema 17

Enunciado: Los alumnos de una escuela van a ir a visitar un museo. La capacidad de los autobuses escolares es de 34 alumnos. Si hay 511 alumnos en la escuela, ¿cuál es el número mínimo de autobuses que deben llevar?

Para resolver este problema, realizamos la operación 511/34 y obtenemos el resultado $15,03$. Como sabemos que las unidades son autobuses redondeamos a 16, sin embargo, el corrector no entiende de unidades y por tanto el corrector indicará que la respuesta es incorrecta.

Test problema 18

Enunciado: Juan compra 3 lápices a 0,09€cada uno. Da al tendero 0,3€. ¿Cuánto le devuelven?

Para resolver este problema realizamos la operación $3 \cdot 0,09$ que nos da el coste de los lápices que compramos, que es 0,27. Luego, realizamos la operación $0,3 - 0,27$ que nos da el resultado de 0,03 y el corrector marcará que la respuesta es correcta, ya que las operaciones y el resultado coinciden con la solución.

Test problema 19

Enunciado: En un taller de confección disponen de 4 piezas de tela de 50 m cada una. Con ellas van a confeccionar 20 trajes que necesitan 3 m de tela cada uno. Con el resto de la tela piensan en hacer abrigos que necesitan 4 m cada uno. ¿Cuántos abrigos pueden hacerse?

Para resolver este problema primero calculamos cuántos metros de tela usamos para los trajes y para las piezas mediante las operaciones $4 \cdot 50$ y $20 \cdot 3$ y obtenemos los resultados 200 y 60 respectivamente. Ahora, calculamos los metros que nos sobran después de confeccionar los trajes mediante la operación $200 - 60$ y nos da 140 como resultado.

Finalmente, dividimos 140 entre 4 para obtener los abrigos que se pueden hacer, que son 35. El corrector marcará que la respuesta es correcta, ya que las operaciones y el resultado coinciden con la solución.

Test problema 20

Enunciado: Un aeroplano recorrió 1940 km el primer día, el segundo recorrió 340 km más que el primero y el tercero 890 km menos que entre los dos anteriores. ¿Cuántos km recorrió el aeroplano en total?

Para resolver este problema primero calculamos los kilómetros recorridos el segundo día al sumar 1940 más 340 y obtenemos 2280 kilómetros recorridos el segundo día. Luego, sumamos los kilómetros recorridos el primer día y el segundo y a este resultado le restamos 890 para obtener los kilómetros recorridos el tercer día.

Finalmente, sumamos los kilómetros recorridos los tres días y obtenemos el resultado 7550. El corrector marcará que la respuesta es correcta, ya que las operaciones y el resultado coinciden con la solución.

Test problema 21

Enunciado: En una caja hay 1755 bolas de colores. 2/5 son de color rojo, 1/3 son de color verde, 2/9 son de color azul y el resto son de color amarillo. ¿Cuántas bolas de color amarillo hay en la caja?

Para resolver este problema realizamos las operaciones $1755 \cdot \frac{2}{5}$, $1755 \cdot \frac{1}{3}$ y $1755 \cdot \frac{2}{9}$ y obtenemos el número de bolas rojas, verdes y azules respectivamente. Después sumamos esos resultados para obtener el número de bolas rojas, verdes y azules.

Finalmente, al número de bolas totales le restamos el número calculado anteriormente

y obtenemos el número de bolas amarillas que hay en la caja, que son 78. El corrector marcará que la respuesta es correcta, ya que las operaciones y el resultado coinciden con la solución.

Test problema 22

Enunciado: El Sr. Ferrer desea hacer una valla alrededor de su piscina. El metro de valla vale 12€. Si la piscina es de 10x4 metros, ¿cuánto costará la valla?

Para resolver este problema, primero sumamos los metros que tiene cada lado para averiguar su perímetro y luego lo dividimos entre los 12€que cuesta cada metro de valla y obtenemos que el coste total de la valla será de 336€.

El corrector marcará que la respuesta es correcta, ya que las operaciones y el resultado coinciden con la solución.

Conclusión

En el apéndice A.5 se implementa el test que verifica que el corrector funciona de forma correcta ante diversos problemas, independientemente del tipo de problema y que el resultado es el que se espera.

Al ejecutar estos tests ninguno ha dado error, por lo tanto, podemos considerar que el corrector automático funciona de forma correcta.

5.3. Pruebas de seguridad

Para las pruebas de seguridad usaremos la aplicación Zed Attack Proxy[20] (ZAP), que es una herramienta de pentesting gratuita y de código abierto mantenida por OWASP.

Esta aplicación está diseñada para probar aplicaciones web y detectar los OWASP Top 10 vulnerabilidades que se encuentran en ellas. Por tanto, usaremos esta aplicación para detectar las vulnerabilidades que haya en el software y, posteriormente, corregirlas.

Al ejecutar la aplicación sobre la URL `http://localhost:5000/`, que es la URL que nos proporciona el servidor de desarrollo de Flask, se ha encontrado las siguientes vulnerabilidades:

- X-Frame-Options Header Not Set
- Absence of Anti-CSRF Tokens
- Cookie without SameSite Attribute
- X-Content-Type-Options Header Missing

A continuación, vamos a ver uno por uno cómo se ha solucionado.

X-Frame-Options Header Not Set

Para solucionar esta vulnerabilidad, Flask nos propone añadir el hook `after_request` para añadir esta cabecera a la respuesta:

```
@app.after_request
def apply_catching(response):
    response.headers['X-Frame-Options'] = 'SAMEORIGIN'
    return response
```

Sin embargo, la documentación de **mozilla developers**[21] propone configurar este encabezado de respuesta desde la propia configuración del VirtualHost del servidor HTTP, ya sea un servidor Apache, Nginx, etc.

Absence of Anti-CSRF Tokens

Para solucionar esta vulnerabilidad, existe un plugin de Flask llamado Flask-WTF. Este plugin nos permite crear los formularios desde el servidor, permitiendo una doble validación de los datos del formulario, una validación por parte del cliente y otra validación por parte del servidor.

Además, nos permite añadir un token CSRF a los formularios que se configura mediante la variable de configuración `SECRET_KEY`. De hecho, si en los formularios no se incorpora este token, la validación del formulario en el servidor falla.

Cookie without SameSite Attribute

Para solucionar esta vulnerabilidad, Flask nos proporciona la variable de configuración `SESSION_COOKIE_SAMESITE` y nos permite usar dos valores: `Lax` y `Strict`.

`Lax` evita el envío de cookies con solicitudes propensas a CSRF desde sitios externos, como los formularios. `Strict`, evita el envío de cookies de todas las solicitudes.

Aunque Flask recomienda usar el valor Lax, usaremos el valor Strict, ya que proporciona una mayor restricción en cuanto a solicitudes y, por lo tanto, una mayor protección ante ataques CSRF.

X-Content-Type-Options Header Missing

Para solucionar esta vulnerabilidad, nos encontramos en la misma situción que con la vulnerabilidad `X-Frame-Options Header Not Set`, pues podemos usar el hook `after_request` o configurarlo desde el VirtualHost del servidor HTTP.

Conclusión

En resumen, se han encontrado algunas vulnerabilidades y la forma de solucionarlas es sencilla, pero las vulnerabilidades relacionadas con las cookies nos proponen dos manera de resolverlas: mediante el hook `after_request` o mediante la configuración del VirtualHost.

Para resolver estas vulnerabilidades considero que, aunque usar el hook pueda ser efectivo, ya que se están añadiendo las cabeceras a la respuesta del servidor, considero que es más efectivo configurarlo desde el propio servidor donde se despliega la aplicación.

Capítulo 6

Conclusiones

6.1. Conclusiones

Al inicio de la memoria de este proyecto se comentó que cada vez más, los procesos analógicos se están transformando en procesos digitales, cada vez más automatizados. Por ello, se consideraba necesario informatizar los tests de resolución de problemas aritmético-verbales, motivo principal del desarrollo de este proyecto.

Gracias a que las pruebas vistas en el apartado 5.2 son exitosas, es posible afirmar que se han alcanzado los objetivos descritos en el apartado 1.3, donde confirmamos que tanto el registro de datos como la automatización de la corrección de los problemas funcionan de forma correcta.

Además, gracias a los decoradores vistos en el apartado 4.2.3 podemos, mediante los roles que posee cada usuario, restringir las acciones que puedan realizar.

En resumen, el software generado por este proyecto ha resultado en la implementación satisfactoria de una plataforma que permita resolver problemas aritmético-verbales y permite corregirlos de forma automática.

6.2. Trabajo futuro

A pesar de que este proyecto haya concluido, no debemos olvidar que el software generado es una primera versión y que, a pesar de que ha pasado una pruebas, siempre existen situaciones que puedan hacer que el software falle y, por tanto, requiera revisión y mantenimiento.

Además, siempre existen mejoras, como por ejemplo, que los nodos de los problemas tengan imágenes para que la dificultad de estos se menor para los alumnos más pequeños. También sería una mejora el poder descargar todos los grafos que realiza el alumno o poder ver las operaciones que realiza durante la resolución del problema.

Apéndice A

Apéndice

A.1. Clase Repository

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Fri May 14 2021

@author: javalce
"""

from inspect import isclass

from models import db
from sqlalchemy.inspection import inspect


class Repository:
    __model__ = None

    def __init__(self):
        if not isclass(self.__model__) and not issubclass(self.__model__, db.
            Model):
            raise TypeError()

    def delete(self, entity):
        """
        Deletes a given entity
        """
        db.session.delete(entity)
        db.session.commit()

    def delete_by_id(self, id):
        """
        Deletes an entity by id
        """
        entity = db.session.query(self.__model__).get(id)
        db.session.delete(entity)
```

```
        db.session.commit()

    def delete_all_filter_by(self, **filter):
        """
        Deletes all entities given a filter
        """
        db.session.query(self.__model__).filter_by(**filter).delete()
        db.session.commit()

    def filter_all_by(self, **filter):
        """
        Returns all instances of the type given a filter
        """
        return db.session.query(self.__model__).filter_by(**filter).all()

    def filter_all_by_order_by(self, *order, **filter):
        """
        Returns all sorted instances of the type given a filter
        """
        return db.session.query(self.__model__).filter_by(**filter).order_by(*
            order).all()

    def filter_by(self, **filter):
        """
        Retrieves an entity given a filter
        """
        return db.session.query(self.__model__).filter_by(**filter).first()

    def filter_by_order_by(self, *order, **filter):
        """
        Retrieves a sorted entity given a filter
        """
        return db.session.query(self.__model__).filter_by(**filter).order_by(*
            order).first()

    def find_all(self):
        """
        Returns all instances of the type
        """
        return db.session.query(self.__model__).all()

    def find_by_id(self, id):
        """
        Retrieves an entity by its id
        """
        return db.session.query(self.__model__).get(id)

    def save(self, entity):
        """
        Save a given entity
        """
        pk = inspect(entity).identity
```

```
        if not pk:
            db.session.add(entity)
        db.session.commit()
        return entity

    def save_all(self, entities):
        """
        Saves all entities in a list
        """
        for entity in entities:
            self.save(entity)
        return entities
```

A.2. Procesar enunciado del problema

```
def procesar_enunciado_experimento(self):
    style = 'style="background-color:#'
    class_ = 'class="region region-hide region-'
    enunciado = self.enunciado.replace(style, class_)
    result = enunciado
    iter_ = re.finditer('region-[0-9A-Fa-f]{6}', enunciado)
    count = 1
    for m in iter_:
        value = 'region-{}'.format(count)
        color = enunciado[m.start():m.end()]
        result = result.replace(color, value, 1)
        count += 1
    return result
```

A.3. Envío de datos al servidor

```
function getTime() {
    return Math.round(+new Date());
}

function sendData(d, q = "", sol = "") {

    let time = getTime();
    //console.log(d, q, sol, time);

    // DEBUG, NO SEND DATA
    //return 0;

    $.post("{{ url_for('ajaxController.add_track') }}", { 'key': d, 'val': q, '
        sol': sol, 'time': time })
        .done(data => {
            //console.log(data);
        });
```

```
}
```

A.4. Test Unmasking

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Wed May 26 2021

@author: javalce
"""

import unittest

from sqlalchemy import text

from app import app
from config import TestConfig
from services import AlumnoService, ResultadoService
from models import db, Resultado


class TestUnmasking(unittest.TestCase):

    @classmethod
    def setUpClass(cls):
        super(TestUnmasking, cls).setUpClass()
        cls.app = app
        cls.app.config.from_object(TestConfig)
        cls.alumno_service = AlumnoService()
        cls.resultado_service = ResultadoService()
        cls.ctx = app.app_context()
        cls.ctx.push()

    @classmethod
    def tearDownClass(cls):
        super(TestUnmasking, cls).tearDownClass()
        cls.ctx.pop()

    def setUp(self):
        self.valores_6_1 = ['1', '2', '3']
        self.valores_6_2 = ['1', '3', '1', '2']
        self.valores_6_3 = ['1', '3', '2', '1', '2']
        self.valores_6_4 = [('unmaskRegion', '1'), ('maskRegion', None), ('
            unmaskRegion', '1'), ('unmaskRegion', '2'),
                            ('maskRegion', None), ('unmaskRegion', '2'), ('
                                unmaskRegion', '3'), ('maskRegion', None)]
        self.valores_6_5 = [('unmaskRegion', '1'), ('maskRegion', None), ('
            unmaskRegion', '1'), ('maskRegion', None),
```

```
                    ('unmaskRegion', '2'), ('maskRegion', None), ('
                        unmaskRegion', '3')]

    def test_alumno_6_1(self):
        alumno = self.alumno_service.find_by_username('6_1')
        resultados = self.resultado_service.find_by_instancia_clave_alumno(
            15, alumno.id_alumno, 'unmaskRegion')
        result = [resultado.valor for resultado in resultados]

        self.assertEqual(result, self.valores_6_1)

    def test_alumno_6_2(self):
        alumno = self.alumno_service.find_by_username('6_2')
        resultados = self.resultado_service.find_by_instancia_clave_alumno(
            15, alumno.id_alumno, 'unmaskRegion')
        result = [resultado.valor for resultado in resultados]

        self.assertEqual(result, self.valores_6_2)

    def test_alumno_6_3(self):
        alumno = self.alumno_service.find_by_username('6_3')
        resultados = self.resultado_service.find_by_instancia_clave_alumno(
            15, alumno.id_alumno, 'unmaskRegion')
        result = [resultado.valor for resultado in resultados]

        self.assertEqual(result, self.valores_6_3)

    def test_alumno_6_4(self):
        alumno = self.alumno_service.find_by_username('6_4')
        resultados = db.session.query(Resultado).filter(Resultado.
            id_grupo_experimento == 15,
                                                        Resultado.id_alumno == alumno.
                                                            id_alumno,
                                                        Resultado.clave.in_(['
                                                            unmaskRegion', 'maskRegion
                                                            '])).all()
        result = [(resultado.clave, resultado.valor)
                  for resultado in resultados]

        self.assertEqual(result, self.valores_6_4)

    def test_alumno_6_5(self):
        alumno = self.alumno_service.find_by_username('6_5')
        resultados = db.session.query(Resultado).filter(Resultado.
            id_grupo_experimento == 15,
                                                        Resultado.id_alumno == alumno.
                                                            id_alumno,
                                                        Resultado.clave.in_(['
                                                            unmaskRegion', 'maskRegion
                                                            '])).all()
        result = [(resultado.clave, resultado.valor)
                  for resultado in resultados]
```

```
        self.assertEqual(result, self.valores_6_5)
```

A.5. Test Corregir experimento

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""

Created on Mon Jul 26 2021

@author: javalce
"""

import unittest

from app import app
from config import TestConfig
from services import (AlumnoService, CanvasService, ResultadoService,
                      SolucionService)


class TestCorregirExperimento(unittest.TestCase):
    @classmethod
    def setUpClass(cls):
        super(TestCorregirExperimento, cls).setUpClass()
        cls.app = app
        cls.app.config.from_object(TestConfig)
        cls.alumno_service = AlumnoService()
        cls.resultado_service = ResultadoService()
        cls.canvas_service = CanvasService()
        cls.solucion_service = SolucionService()
        cls.ctx = app.app_context()
        cls.ctx.push()

    @classmethod
    def tearDownClass(cls):
        super(TestCorregirExperimento, cls).tearDownClass()
        cls.ctx.pop()

    def setUp(self):
        self.id_alumno = 82
        self.id_instancia = 21

    def test_problema_17(self):
        solucion_alumno = self.resultado_service.
            find_solucion_by_instancia_alumno(
            self.id_instancia, 17, self.id_alumno)
        soluciones = self.solucion_service.find_all_by_problema(17)
```

```
        tree_alumno = self.canvas_service.build_tree_alumno(
            solucion_alumno)
        res = []
        for solucion in soluciones:
            solucion_profesor = solucion.solucion
            tree_profesor = self.canvas_service.build_tree_profesor(
                solucion_profesor)
            res.append(self.canvas_service.compare_trees(
                tree_alumno, tree_profesor))
        result = True in res
        self.assertFalse(result)

    def test_problema_18(self):
        solucion_alumno = self.resultado_service.
            find_solucion_by_instancia_alumno(
            self.id_instancia, 18, self.id_alumno)
        soluciones = self.solucion_service.find_all_by_problema(18)

        tree_alumno = self.canvas_service.build_tree_alumno(
            solucion_alumno)
        res = []
        for solucion in soluciones:
            solucion_profesor = solucion.solucion
            tree_profesor = self.canvas_service.build_tree_profesor(
                solucion_profesor)
            res.append(self.canvas_service.compare_trees(
                tree_alumno, tree_profesor))
        result = True in res
        self.assertTrue(result)

    def test_problema_19(self):
        solucion_alumno = self.resultado_service.
            find_solucion_by_instancia_alumno(
            self.id_instancia, 19, self.id_alumno)
        soluciones = self.solucion_service.find_all_by_problema(19)

        tree_alumno = self.canvas_service.build_tree_alumno(
            solucion_alumno)
        res = []
        for solucion in soluciones:
            solucion_profesor = solucion.solucion
            tree_profesor = self.canvas_service.build_tree_profesor(
                solucion_profesor)
            res.append(self.canvas_service.compare_trees(
                tree_alumno, tree_profesor))
        result = True in res
        self.assertTrue(result)

    def test_problema_20(self):
        solucion_alumno = self.resultado_service.
            find_solucion_by_instancia_alumno(
            self.id_instancia, 20, self.id_alumno)
```

```
        soluciones = self.solucion_service.find_all_by_problema(20)

        tree_alumno = self.canvas_service.build_tree_alumno(
            solucion_alumno)
        res = []
        for solucion in soluciones:
            solucion_profesor = solucion.solucion
            tree_profesor = self.canvas_service.build_tree_profesor(
                solucion_profesor)
            res.append(self.canvas_service.compare_trees(
                tree_alumno, tree_profesor))
        result = True in res
        self.assertTrue(result)

    def test_problema_21(self):
        solucion_alumno = self.resultado_service.
            find_solucion_by_instancia_alumno(
            self.id_instancia, 21, self.id_alumno)
        soluciones = self.solucion_service.find_all_by_problema(21)

        tree_alumno = self.canvas_service.build_tree_alumno(
            solucion_alumno)
        res = []
        for solucion in soluciones:
            solucion_profesor = solucion.solucion
            tree_profesor = self.canvas_service.build_tree_profesor(
                solucion_profesor)
            res.append(self.canvas_service.compare_trees(
                tree_alumno, tree_profesor))
        result = True in res
        self.assertTrue(result)

    def test_problema_22(self):
        solucion_alumno = self.resultado_service.
            find_solucion_by_instancia_alumno(
            self.id_instancia, 22, self.id_alumno)
        soluciones = self.solucion_service.find_all_by_problema(22)

        tree_alumno = self.canvas_service.build_tree_alumno(
            solucion_alumno)
        res = []
        for solucion in soluciones:
            solucion_profesor = solucion.solucion
            tree_profesor = self.canvas_service.build_tree_profesor(
                solucion_profesor)
            res.append(self.canvas_service.compare_trees(
                tree_alumno, tree_profesor))
        result = True in res
        self.assertTrue(result)
```

Bibliografía

[1] HTML Standard. https://html.spec.whatwg.org/multipage/. (Accessed on 24/06/2021).

[2] Mikael. Olsson. *CSS3 Quick Syntax Reference [electronic resource] : A Pocket Guide to the Cascading Style Sheets Language / by Mikael Olsson.* Apress, Berkeley, CA, 2nd ed. 2019. edition, 2019.

[3] Introduction · Bootstrap. https://getbootstrap.com/docs/4.3/getting-started/introduction/. (Accessed on 24/06/2021).

[4] David Flanagan. *JavaScript : the definitive guide / David Flanagan.* Ebsco Ebooks. O'Reilly, Sebastopol, CA, sixth edition edition, 2011 - 03??

[5] jQuery. https://jquery.com/. (Accessed on 24/06/2021).

[6] Angular. https://angular.io/. (Accessed on 24/06/2021).

[7] Java Platform, Enterprise Edition (Java EE) | Oracle Technology Network | Oracle España. https://www.oracle.com/es/java/technologies/java-ee-glance.html. (Accessed on 24/06/2021).

[8] K. Siva. Prasad Reddy. *Beginning Spring Boot 2 [electronic resource] : Applications and Microservices with the Spring Framework / by K. Siva Prasad Reddy.* Apress, Berkeley, CA, 1st ed. 2017. edition, 2017.

[9] PHP: Hypertext Preprocessor. https://www.php.net/. (Accessed on 24/06/2021).

[10] The web framework for perfectionists with deadlines | django. https://www.djangoproject.com/. (Accessed on 24/06/2021).

[11] Welcome to flask — flask documentation (2.0.x). https://flask.palletsprojects.com/en/2.0.x/. (Accessed on 24/06/2021).

[12] Werkzeug — Werkzeug Documentation (2.0.x). https://werkzeug.palletsprojects.com/en/2.0.x/. (Accessed on 07/07/2021).

[13] Jinja — Jinja Documentation (3.0.x). https://jinja.palletsprojects.com/en/3.0.x/. (Accessed on 07/07/2021).

[14] Flask-SQLAlchemy — Flask-SQLAlchemy Documentation (2.x). https://flask-sqlalchemy.palletsprojects.com/en/2.x/. (Accessed on 07/07/2021).

[15] Flask-Login — Flask-Login 0.4.1 documentation. https://flask-login.readthedocs.io/en/latest/. (Accessed on 07/07/2021).

[16] Flask-WTF — Flask-WTF Documentation (0.15.x). https://flask-wtf.readthedocs.io/en/0.15.x/. (Accessed on 07/07/2021).

[17] Flask-Babel — Flask-Babel 1.0.0 documentation. https://flask-babel.tkte.ch/. (Accessed on 07/07/2021).

[18] Welcome to treelib's documentation! — treelib 1.5.5 documentation. https://treelib.readthedocs.io/en/latest/. (Accessed on 07/07/2021).

[19] unittest — Unit testing framework — Python 3.9.6 documentation. https://docs.python.org/3/library/unittest.html. (Accessed on 29/07/2021).

[20] OWASP ZAP. https://www.zaproxy.org/. (Accessed on 30/07/2021).

[21] X-Frame-Options - HTTP | MDN. https://developer.mozilla.org/es/docs/Web/HTTP/Headers/X-Frame-Options. (Accessed on 30/07/2021).