Capítulo 4

Metodología y resultados

Todo proyecto de desarrollo de software está basado en algún ciclo de vida del software. En este capítulo se va a describir el ciclo de vida empleado para el desarrollo mediante un diagrama de Gantt y detalles acerca de los requisitos, el diseño y desarrollo de la aplicación.

4.1.- PLANIFICACIÓN DEL PROYECTO

Este proyecto consiste en una aplicación destinada al Centro de Investigación Operativa (CIO) de la Universidad Miguel Hernández de Elche para suplir una necesidad que hay de tener una plataforma en la que se puedan crear cursos de manera sencilla e intuitiva. La planificación del proyecto ha partido con unos requisitos básicos y a raíz del proceso del desarrollo se han ido descartando, añadiendo o perfeccionando ciertas características del proyecto.

4.1.1.- Ciclo de vida

Para la realización del proyecto se ha seguido un ciclo de vida incremental, el cual se desarrolla mediante una serie de ciclos repetidos de fases que van añadiendo sucesivamente funcionalidad al proyecto. Cada fase habrá completado un conjunto de objetivos y las siguientes fases pueden mejorar los objetivos creados anteriormente o crear nuevos, por lo que el resultado del proyecto será la acumulación de funcionalidades construidas en las fases [60] (Ver figura 4.1).

Las primeras fases se trataría del proceso de estudio de los cuadernos Jupyter y posteriormente de los editores de código online, en el que se realizan numerosas implementaciones, se comprueba si funciona correctamente y después se mejora o se descarta dicha funcionalidad. Después se implementa la base de datos, la cual se va modificando conforme van cambiando los objetivos requeridos y el desarrollo del código de la aplicación. Todo ello se ha ido construyendo poco a poco siguiendo unos objetivos marcados de funcionalidad y además siguiendo un orden en el desarrollo para que sea más sencillo, como por ejemplo, empezar por las funcionalidades del Administrador, después del profesor y por último del alumno.

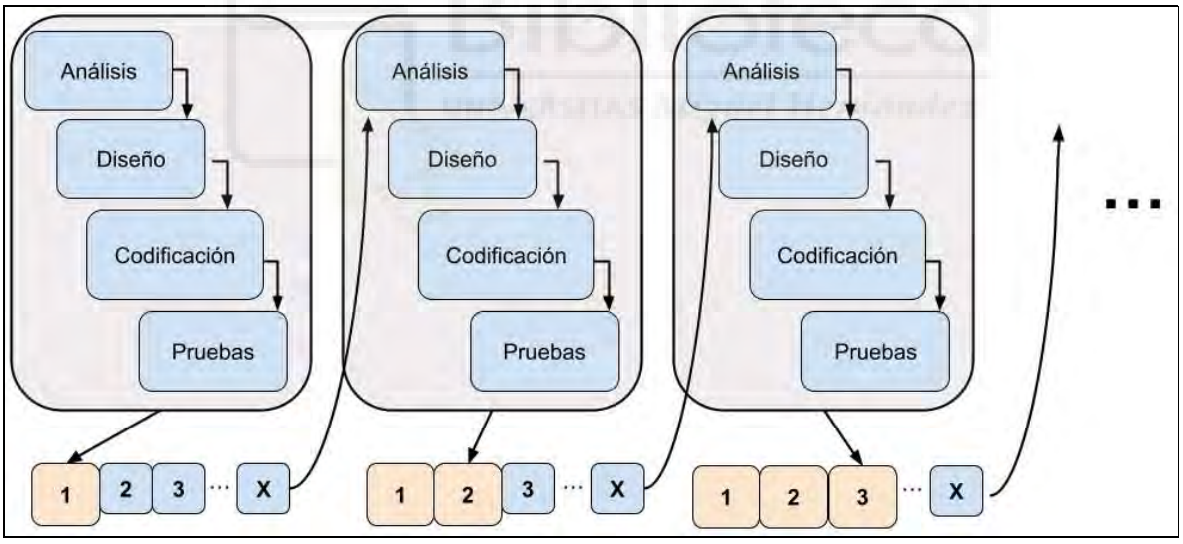

Figura 4.1.- Ciclo de vida incremental

4.1.2.- Etapas del proyecto. Diagrama de Gantt

El desarrollo del proyecto ha tenido una duración de 16 semanas, desde la tercera semana de febrero hasta la segunda semana de junio. En la figura 4.2 se puede observar las diferentes etapas que han habido en el proceso de desarrollo de la aplicación y la duración de cada una de ellas.

Figura 4.2.- Diagrama de Gantt

En la tercera semana de Febrero se tuvo la primera reunión donde se especificaron los requisitos que debía tener el proyecto y a partir de ese momento se comenzó con el proceso de investigación de las diferentes características que proporcionan otras plataformas, la implementación de un cuaderno Jupyter en el navegador y de los editores de código online.

En el estudio de la implementación del cuaderno Jupyter y los editores de código online se realizaron numerosas pruebas de integración y modificaciones para comprobar antes de comenzar con el desarrollo del código de la aplicación, que la edición de código en python funciona correctamente.

A medida que se sacaban nuevas conclusiones como resultado de las investigaciones, se reflejaban todos los resultados en la memoria. Cuando finalizó la fase de investigación se comenzó con el análisis, el diseño de la aplicación y con la programación del código. A medida que se iban desarrollando nuevas funcionalidades en la aplicación, se realizaba la corrección de errores, la comprobación de su correcto funcionamiento y mejoras en su optimización.

Las reuniones con el tutor se han realizado casi todas las semanas para tener un seguimiento de la evolución del desarrollo del TFG, debatir sobre la implementación de nuevas funcionalidades y resolver dudas que surgen acerca de los requisitos que se necesitan y a la hora de implementar ciertas funcionalidades en el código.

Por último, se resolvieron los últimos detalles pendientes en la aplicación y se completó la presentación para la defensa del TFG.

4.2.- CAPTURA DE REQUISITOS

La captura de requisitos tiene como objetivo descubrir y recoger todos y cada uno de los requisitos funcionales y no funcionales de forma clara y concisa de la aplicación., empezando por identificar los diferentes tipos de usuarios que la usarán, para, seguidamente, determinar qué funciones puede hacer cada uno de ellos

Esta aplicación dispone de 5 roles o tipos de usuario diferentes, siendo el administrador, el profesor, el profesor responsable, el alumno y el usuario no identificado (Ver figura 4.3).

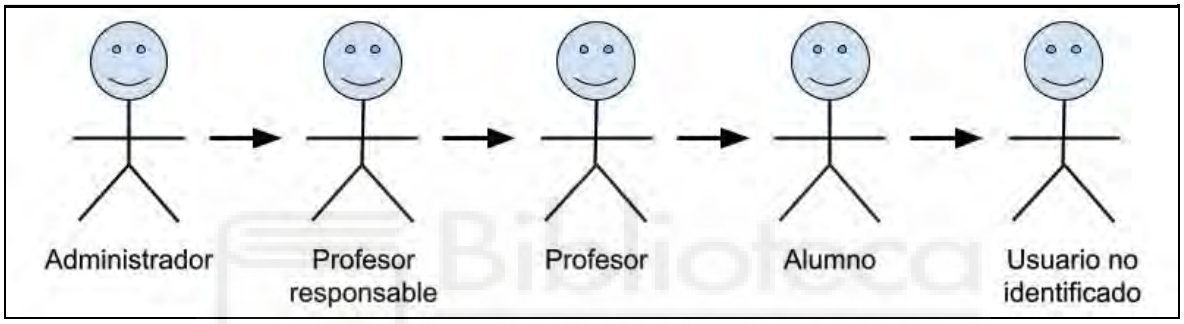

Figura 4.3.- Tipos de usuario y relación de herencia

Todos los usuarios tienen acceso a ciertas funcionalidades de la aplicación, desde las funcionalidades más básicas para el usuario no registrado, hasta acceso a todas las funcionalidades para el administrador. Todos los usuarios siguen una relación de herencia en el que cada usuario tiene sus funcionalidades específicas y además tienen acceso a las funcionalidades del usuario que hereda, aunque en ciertos casos no se cumple estrictamente la herencia entre el administrador y el profesor responsable. El administrador puede ser docente de una edición, pero en algunos casos será profesor responsable (se cumplirá la relación de herencia), y en otros será profesor no responsable, por lo que el administrador no tendrá las funcionalidades que tiene el profesor responsable, si no las funcionalidades que tiene el profesor no responsable.

Tabla 4.1.- Rol de usuario no identificado

Usuario	Usuario no identificado
Descripción	El usuario no identificado únicamente podrá visualizar los cursos que en ese momento están disponibles y una pequeña descripción de dichos cursos (Ver figura 4.4).
Casos de uso	C.U.1, C.U.2, C.U.3

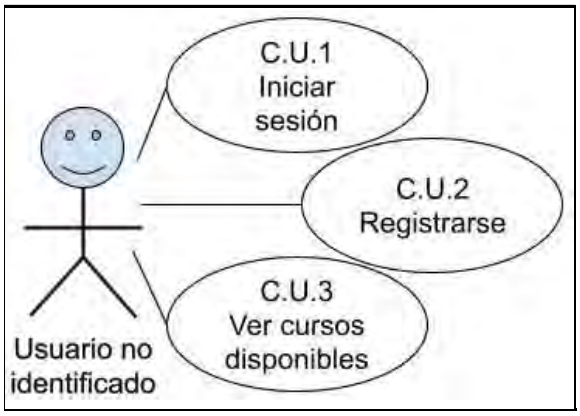

Figura 4.4.- Casos de uso del usuario no identificado

Tabla 4.2.- Rol de alumno

Usuario	Alumno
Descripción	El alumno puede hacer todo lo que hace el usuario no identificado y además puede inscribirse en cursos, ver y modificar sus datos y utilizar el editor de código dentro de un curso (Ver figura 4.5).
Casos de uso	En el caso que esté matriculado de algún curso: C.U.4, C.U.5, C.U.6, C.U.7, C.U.8, C.U.9, C.U.10, C.U.11, C.U.12, C.U.13, C.U.14, C.U.15, C.U.16, C.U.17, C.U.18, C.U.19, C.U.20. En el caso que no esté matriculado de ningún curso: C.U.4, C.U.5, C.U.6, C.U.7, C.U.8, C.U.9.

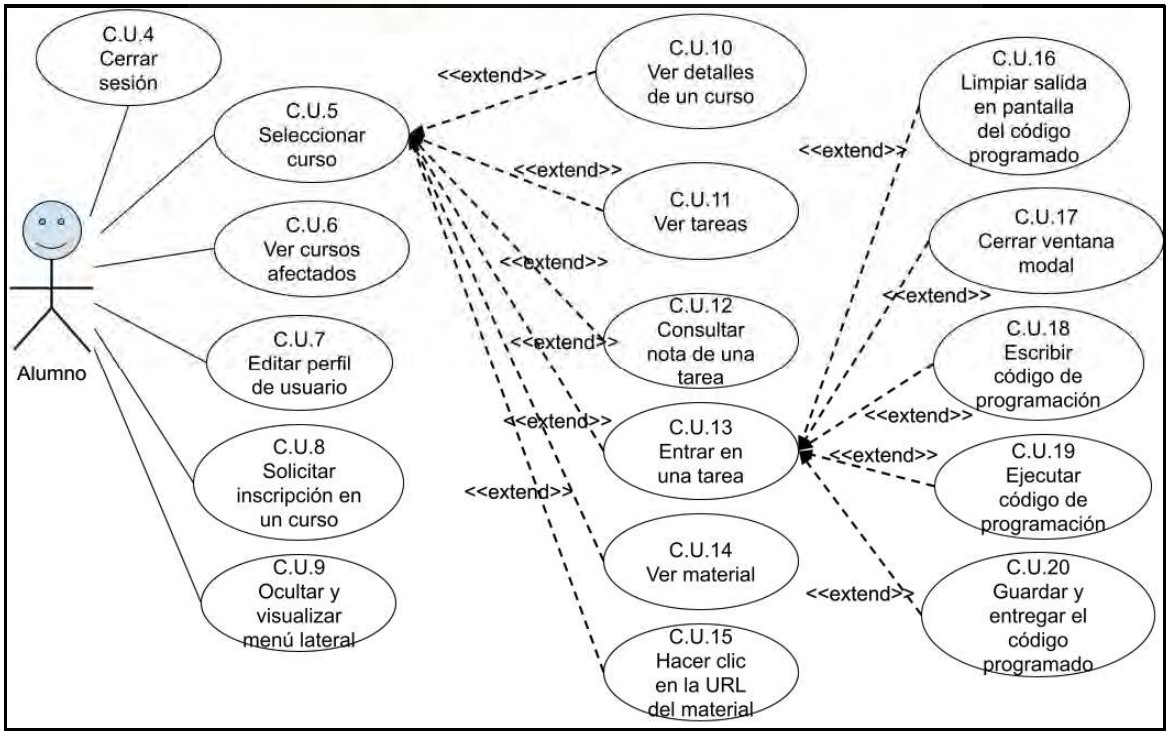

Figura 4.5.- Casos de uso del alumno

Tabla 4.3.- Rol de profesor

Usuario	Profesor
Descripción	El profesor puede hacer todo lo que hace el alumno, puesto que también puede ser alumno de otros cursos y además puede editar un curso y calificar a los alumnos (Ver figura 4.6).
Casos de uso	En el caso que esté matriculado en algún curso pero no imparta ninguno: C.U.4, C.U.5, C.U.6, C.U.7, C.U.8, C.U.9, C.U.10, C.U.11, C.U.12, C.U.13, C.U.14, C.U.15, C.U.16, C.U.17, C.U.18, C.U.19, C.U.20. En el caso que esté matriculado en algún curso e imparta alguno: C.U.4, C.U.5, C.U.6, C.U.7, C.U.8, C.U.9, C.U.10, C.U.11, C.U.12, C.U.13, C.U.14, C.U.15, C.U.16, C.U.17, C.U.18, C.U.19, C.U.20, C.U.21, C.U.22, C.U.23. En el caso que no esté matriculado en algún curso y no imparta ninguno: C.U.4, C.U.5, C.U.6, C.U.7, C.U.8, C.U.9. En el caso que no esté matriculado en algún curso pero imparta alguno: C.U.4, C.U.5, C.U.6, C.U.7, C.U.8, C.U.9, C.U.21, C.U.22, C.U.23.

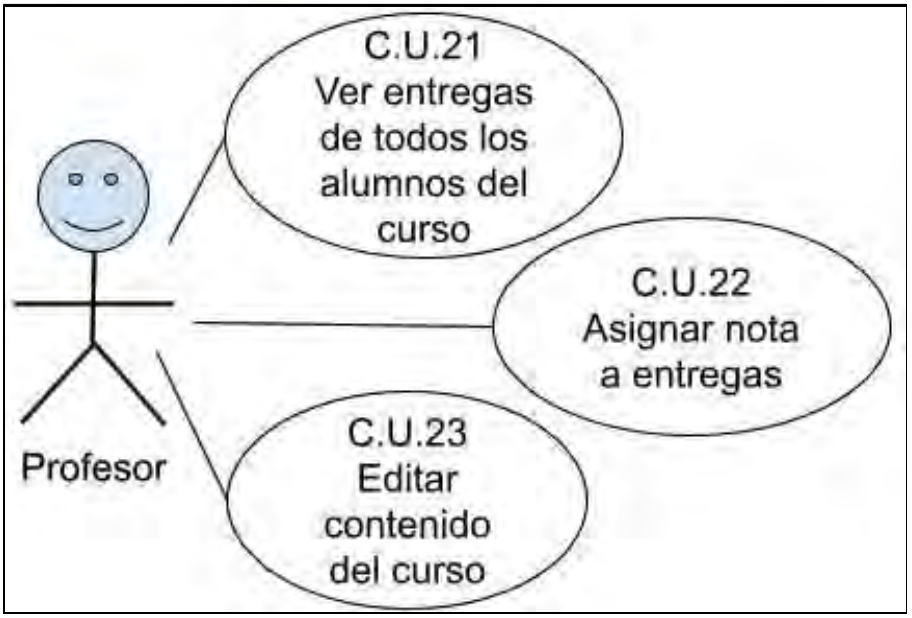

Figura 4.6.- Casos de uso del profesor

Tabla 4.4.- Rol de profesor responsable

Usuario	Profesor responsable
Descripción	El profesor responsable puede hacer todo lo que puede hacer el rol profesor y además puede gestionar los profesores y alumnos que hay dentro de un curso (Ver figura 4.7).
Casos de uso	En el caso que esté matriculado en algún curso pero no imparta ninguno: C.U.4, C.U.5, C.U.6, C.U.7, C.U.8, C.U.9, C.U.10, C.U.11, C.U.12, C.U.13, C.U.14, C.U.15, C.U.16, C.U.17, C.U.18, C.U.19, C.U.20. En el caso que esté matriculado en algún curso e imparta alguno: C.U.4, C.U.5, C.U.6, C.U.7, C.U.8, C.U.9, C.U.10, C.U.11, C.U.12, C.U.13, C.U.14, C.U.15, C.U.16, C.U.17, C.U.18, C.U.19, C.U.20, C.U.21, C.U.22, C.U.23, C.U.24, C.U.25, C.U.26, C.U.27, C.U.28, C.U.29. En el caso que no esté matriculado en algún curso y no imparta ninguno: C.U.4, C.U.5, C.U.6, C.U.7, C.U.8, C.U.9. En el caso que no esté matriculado en algún curso pero imparta alguno: C.U.4, C.U.5, C.U.6, C.U.7, C.U.8, C.U.9, C.U.21, C.U.22, C.U.23, C.U.24, C.U.25, C.U.26, C.U.27, C.U.28, C.U.29.

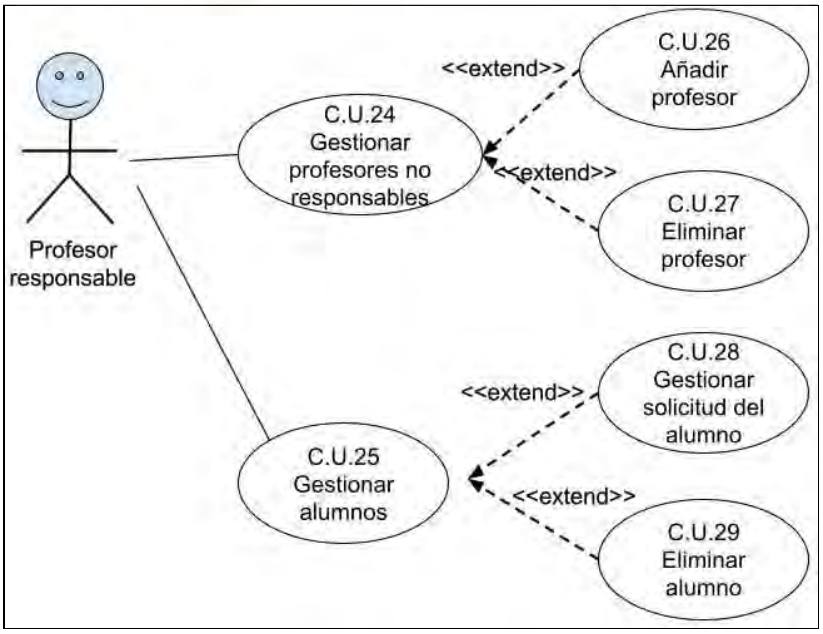

Figura 4.7.- Casos de uso del profesor responsable

Tabla 4.5.- Rol de administrador

Usuario	Administrador
Descripción	El administrador puede hacer todo lo que hace el profesor responsable, ya que también puede ser responsable de un curso y además puede gestionar los cursos, las ediciones de los cursos, los alumnos y los profesores en el caso que sea docente de un curso (Ver figura 4.8).
Casos de uso	En el caso que esté matriculado en algún curso pero no imparta ninguno: C.U.4, C.U.5, C.U.6, C.U.7, C.U.8, C.U.9, C.U.10, C.U.11, C.U.12, C.U.13, C.U.14, C.U.15, C.U.16, C.U.17, C.U.18, C.U.19, C.U.20, C.U.30, C.U.31, C.U.32, C.U.33, C.U.34, C.U.35, C.U.36, C.U.37, C.U.38, C.U.39, C.U.40, C.U.41. En el caso que esté matriculado en algún curso e imparta alguno como docente responsable: C.U.4, C.U.5, C.U.6, C.U.7, C.U.8, C.U.9, C.U.10, C.U.11, C.U.12, C.U.13, C.U.14, C.U.15, C.U.16, C.U.17, C.U.18, C.U.19, C.U.20, C.U.21, C.U.22, C.U.23, C.U.24, C.U.25, C.U.26, C.U.27, C.U.28, C.U.29, C.U.30, C.U.31, C.U.32, C.U.33, C.U.34, C.U.35, C.U.36, C.U.37, C.U.38, C.U.39, C.U.40, C.U.41. En el caso que esté matriculado en algún curso e imparta alguno como docente no responsable: C.U.4, C.U.5, C.U.6, C.U.7, C.U.8, C.U.9, C.U.10, C.U.11, C.U.12, C.U.13, C.U.14, C.U.15, C.U.16, C.U.17, C.U.18, C.U.19, C.U.20, C.U.21, C.U.22, C.U.23, C.U.30, C.U.31, C.U.32, C.U.33, C.U.34, C.U.35, C.U.36, C.U.37, C.U.38, C.U.39, C.U.40, C.U.41. En el caso que no esté matriculado en algún curso y no imparta ninguno: C.U.4, C.U.5, C.U.6, C.U.7, C.U.8, C.U.9, C.U.30, C.U.31, C.U.32, C.U.33, C.U.34, C.U.35, C.U.36, C.U.37, C.U.38, C.U.39, C.U.40, C.U.41. En el caso que no esté matriculado en algún curso pero imparta alguno como docente responsable: C.U.4, C.U.5, C.U.6, C.U.7, C.U.8, C.U.9, C.U.21, C.U.22, C.U.23, C.U.24, C.U.25, C.U.26, C.U.27, C.U.28, C.U.29, C.U.30, C.U.31, C.U.32, C.U.33, C.U.34, C.U.35, C.U.36, C.U.37, C.U.38, C.U.39, C.U.40, C.U.41. En el caso que no esté matriculado en algún curso pero imparta alguno como docente no responsable: C.U.4, C.U.5, C.U.6, C.U.7, C.U.8, C.U.9, C.U.21, C.U.22, C.U.23, C.U.30, C.U.31, C.U.32, C.U.33, C.U.34, C.U.35, C.U.36, C.U.37, C.U.38, C.U.39, C.U.40, C.U.41.

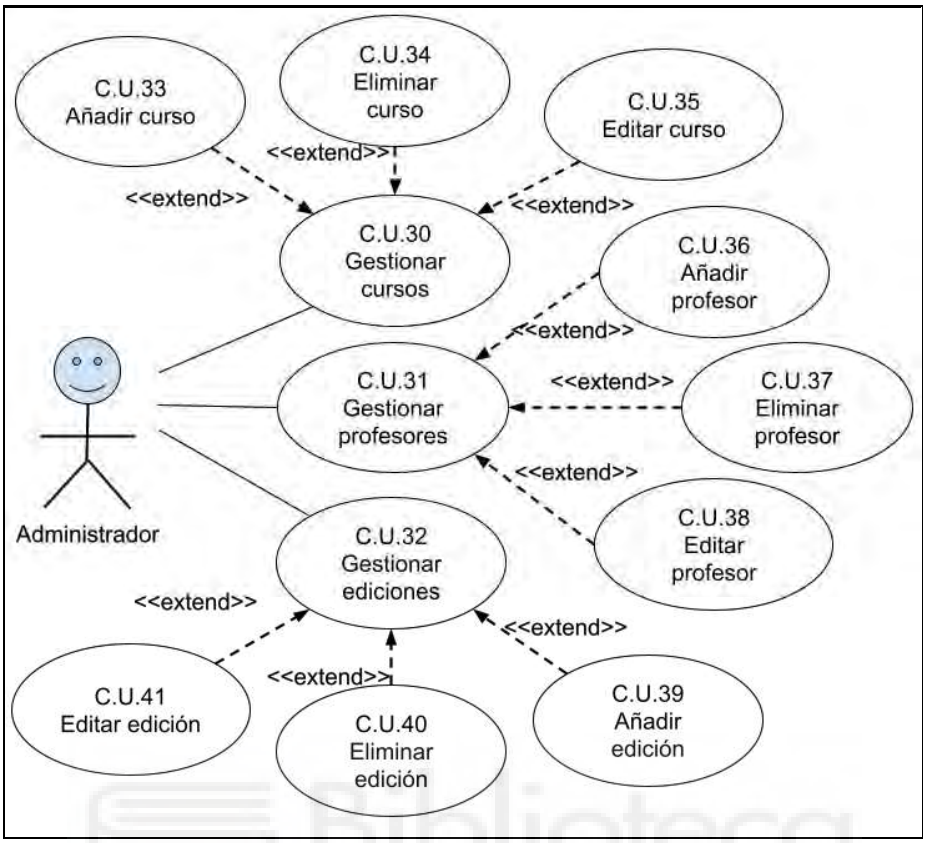

Figura 4.8.- Casos de uso del administrador

En el Anexo que hay al final de la presente memoria se encuentran las plantillas que describen en detalle cada uno de los casos de uso enumerados anteriormente.

4.3.- Diseño

En este punto se explicará de forma breve el diseño de la base de datos y algunas capturas del esbozo del diseño de la aplicación.

4.3.1.- Diagrama Entidad/Relación y Relacional

Para la realización de la base de datos de esta aplicación, se han creado 8 tablas (siendo 2 de ellas relaciones muchos a muchos) (Ver figura 4.9). Por un lado se tiene la tabla de "Usuarios" donde ahí estarán incluidos todos los tipos de usuario, es decir, independientemente del rol que tengan. La tabla "Curso" se trata del tipo de curso que se tiene y la tabla "Edición" corresponderá a una alteración del curso, es decir, si existe un curso llamado "Python desde cero", podrá haber varias ediciones de dicho curso para verano, para invierno y para primavera. Si el usuario tiene el rol de "Administrador" o "Profesor" tendrá la potestad de impartir una edición, editarla, si es responsable gestionar usuarios del curso y

además podrá matricularse de una edición, actuando así como alumno de esa edición. El rol de "Alumno" solo podrá matricularse en las ediciones, visualizar su contenido y entregar tareas, pero no podrá impartirla. Cabe destacar que la tabla "Entrega" tiene un campo llamado "Entrega" que se tratará del código de programación que haya guardado el usuario, en el que el usuario no tendrá permitido entregar el código cuando la fecha de entrega haya expirado.

Cuando el usuario con rol de "Alumno" quiere inscribirse en una edición, deberá enviar una solicitud de admisión y el profesor responsable de ese curso podrá aceptar o rechazar la solicitud, por lo que cuando se acepte la solicitud, el usuario ya tendrá acceso a la edición.

El administrador tendrá la potestad de gestionar los profesores, los cursos y las ediciones, por lo que en todas ellas se podrá modificar, añadir o eliminar registros.

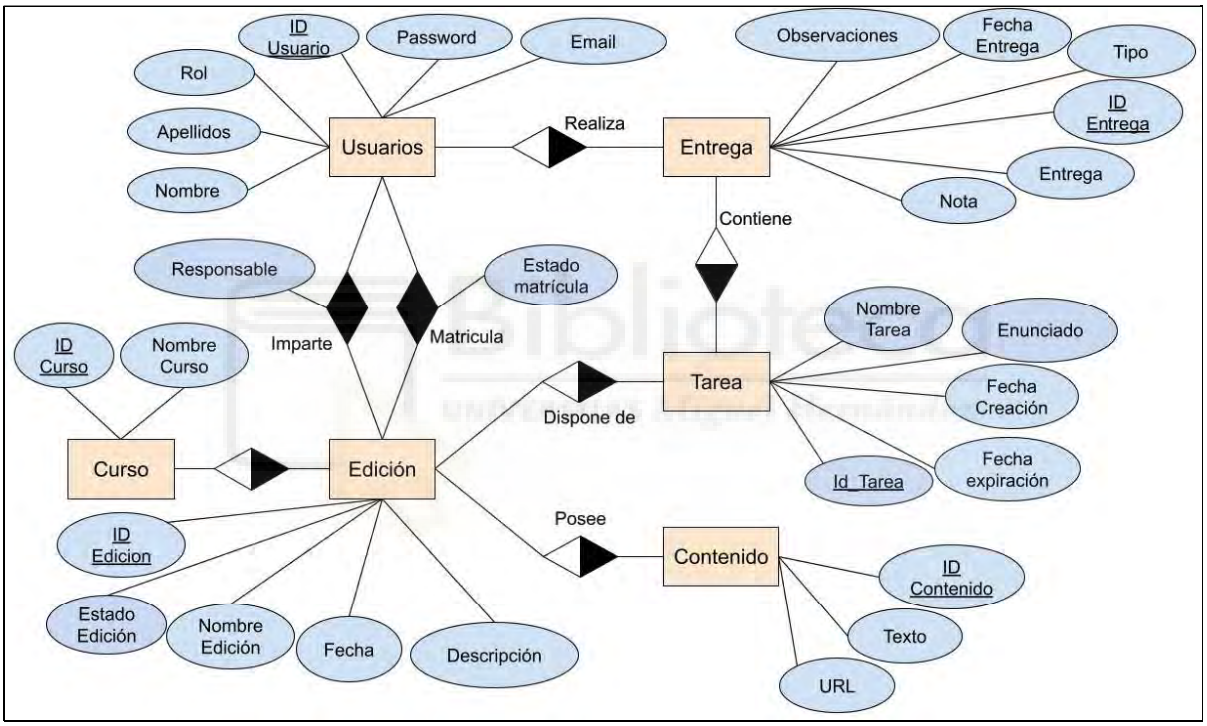

Figura 4.9.- Modelo Entidad/Relación de la base de datos

En la figura 4.10 se puede observar las relaciones, las claves principales y las claves foráneas de cada tabla. Cabe destacar que la tabla "imparte", "matricula" y "tarea" tienen una clave principal compuesta ya que ningún campo por si solo cumple la condición de ser clave:

- Tabla "imparte": Tiene clave principal compuesta en los campos del identificador del profesor y el identificador de la edición, puesto que un profesor puede impartir muchas ediciones diferentes y puede haber más de un profesor impartiendo una edición, pero en ningún caso un mismo profesor podrá impartir en el mismo momento una misma edición.

- Tabla “matricula”: Tiene clave principal compuesta en los campos del identificador del alumno y el identificador de la edición, ya que un alumno puede estar matriculado en el mismo momento de muchas ediciones, una edición puede tener muchos alumnos, pero en ningún caso un alumno puede estar matriculado dos veces en un mismo momento en una misma edición.

- Tabla “tarea”: Tiene como clave principal compuesta en los campos del identificador de la tarea y en el identificador de la edición, ya que una edición puede tener muchas tareas diferentes en un mismo momento, pero en ningún caso pueden haber dos tareas iguales (con el mismo identificador) en alguna edición en un mismo momento.

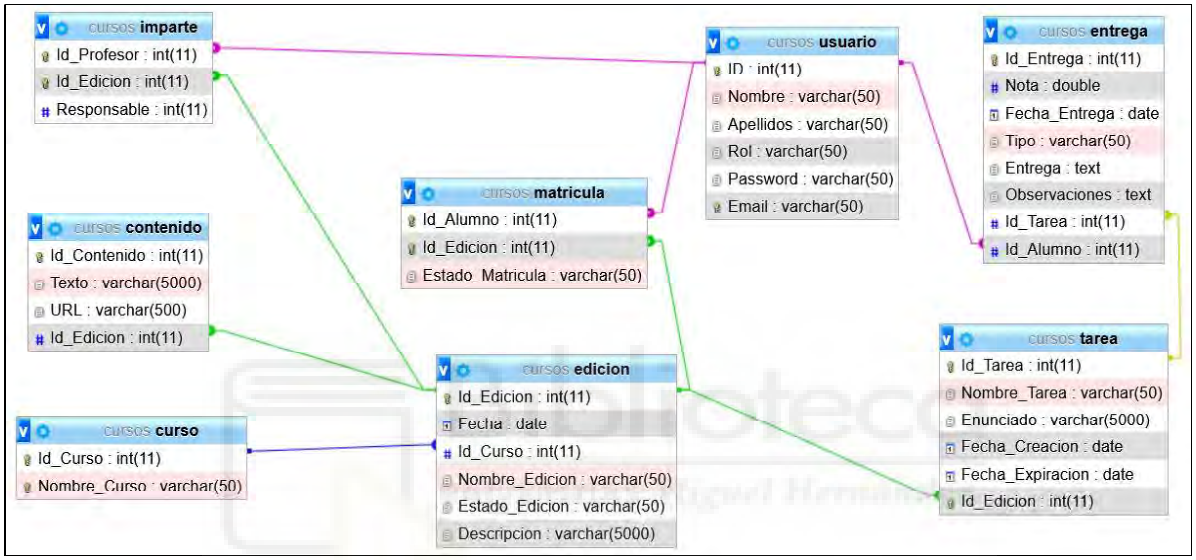

Figura 4.10.- Modelo relacional visto desde XAMPP

4.3.2.- Interfaz gráfica

En este apartado se van a mostrar algunos de los diseños de pantalla realizados. En la figura 4.11 se puede observar cómo será la aplicación cuando el usuario no está logueado (apariencia pública) y cuando el usuario ya se ha logueado (apariencia privada). En la apariencia pública aparecerán únicamente los cursos que hay disponibles en ese momento y en la apariencia privada aparecerá un menú superior con varios botones, entre los cuales uno de ellos es un botón redondo “hamburguesa” que permitirá mostrar y ocultar un menú lateral izquierdo con las diferentes opciones, dependiendo del rol del usuario logueado. El espacio que se encuentra en la parte central, se trata de el lugar donde estará situada toda la información según en qué apartado esté el usuario.

En la figura 4.12 se muestra el menú desplegable que habrá en la parte lateral izquierda dentro del acceso identificado.

Cuando el usuario tenga un rol de administrador, le aparecerá el botón de "Mis cursos" y cuando haga clic, se le mostrarán todas las ediciones a las que está matriculado. También se le mostrará el botón "Docencia" donde aparecerán todos los cursos que imparte y por último el botón "Administración" donde al hacer clic, aparecerá un desplegable con las opciones de gestión de cursos, ediciones y profesores. Cuando el usuario tenga rol de profesor, le aparecerán los desplegables de "Mis cursos" y "Docencia" ya que puede impartir una edición y también estar matriculado de otras ediciones. Cuando el usuario tenga rol de alumno, únicamente se le aparecerá el desplegable de "Mis cursos" ya que únicamente puede estar matriculado de los cursos.

Cuando no se esté impartiendo algún curso o no se esté matriculado de ninguna edición, se le podrá seguir haciendo clic al botón pero no aparecerá ningún desplegable, sino un mensaje avisando de que no está impartiendo o cursando ningúna edición ya que ese desplegable se abrirá con las diferentes ediciones que se están impartiendo o cursando.

Figura 4.11.- Diseño de la parte pública y privada de la aplicación

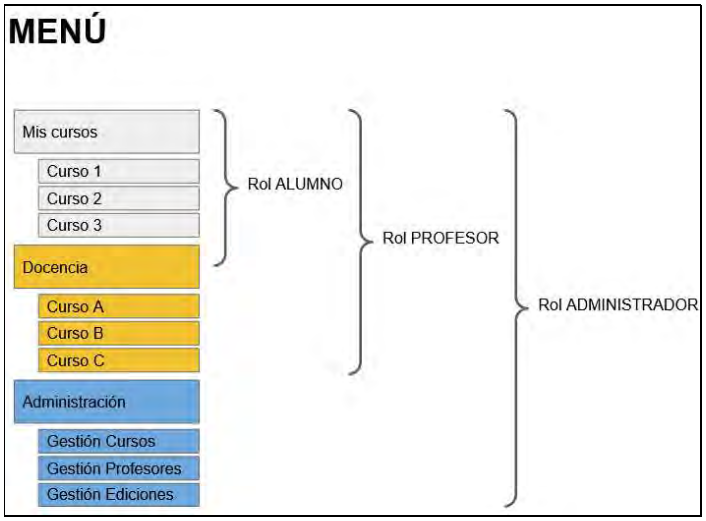

Figura 4.12.- Menú lateral izquierdo dependiendo del rol de usuario

En la figura 4.13 se puede observar un ejemplo de cómo visualizará el administrador una tabla con todos los datos de los profesores (a excepción de su contraseña) y con varios botones para poder añadir nuevos usuarios con rol de profesor, editar sus datos (a excepción de su ID, su rol de usuario y su email) y además poder eliminarlos. Cabe destacar que los colores que se han empleado en los diseños, no se han utilizado en la aplicación real.

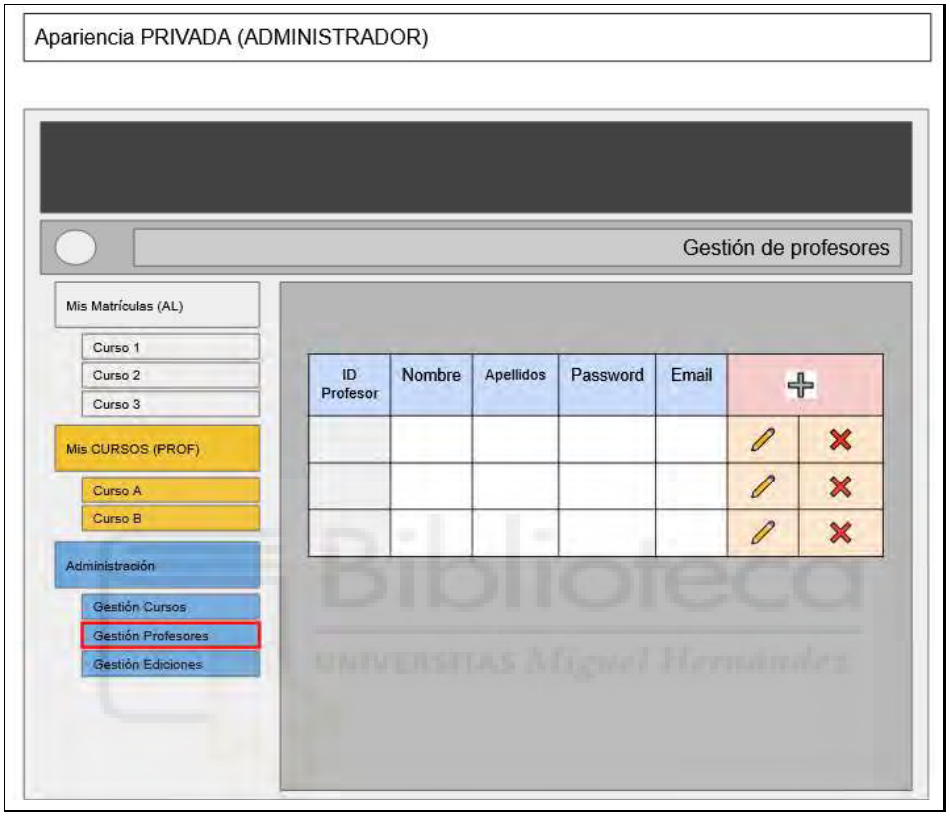

Figura 4.13.- Apariencia de la sección "Gestión Profesores" del administrador

4.4.- Implementación

Unos de los problemas más complejos que ha tenido el desarrollo de la aplicación ha sido la creación del editor de código online que permitiera la ejecución de código python en la parte del cliente. Por todo ello, se describirán las partes más relevantes de CodeMirror y Pyodide para permitir el correcto funcionamiento.

4.4.1.- CodeMirror

Para la implementación del editor de código online CodeMirror se han necesitado seguir algunos pasos para su implementación. Primero se deberá importar todo el código fuente de CodeMirror y del lenguaje de programación requerido para que se pinten las palabras clave

de ese lenguaje (Ver figura 4.14). La integridad de la etiqueta script se trata de una firma digital del contenido de manera que permita la integridad de los recursos, es decir, que no se ejecute si ha sufrido alguna modificación [61].

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/codemirror/5.63.3/codemirror.min.js"
        integrity="sha512-XMlgZzPyVXf1I/wbGnofk1Hfdx+zAWyZjh6c21yGo/k1zNC4Ve6xcQnTDTCHrjFGsOrVicJsBURLYktVEu/8vQ=="
        crossorigin="anonymous" referrerpolicy="no-referrer"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/codemirror/5.63.3/mode/python/python.min.js"
    integrity="sha512-/mavDpedrvPG/0Grj2Ughxte/fsm42ZmZWWpHz1jCbzd5ECv8CB7PomGtw0NAnhHmE/lkDFkRMupjoohbKNA1Q=="
    crossorigin="anonymous" referrerpolicy="no-referrer"></script>
```

Figura 4.14.- Importación del código fuente y de python en CodeMirror

Después se creará un elemento “textarea” con el id “code” donde se convertirá después en el editor de CodeMirror (Ver figura 4.15). Se le podrá añadir los lenguajes de programación que se quiera (descargando su correspondiente librería) y además se podrá desactivar y activar ciertas características como por ejemplo comentarios, el número de líneas, entre otros.

```
var editor = CodeMirror.fromTextArea(document.getElementById("code"), {
    mode: {
        name: "python",
        version: 3,
        singleLineStringErrors: false
    },
    lineNumbers: true,
    matchBrackets: true,
    afterCursor: true,
    closeBrackets: true,
    comment: true
});
```

Figura 4.15.- Transformación de textarea a editor de CodeMirror

4.4.1.- Pyodide

Para la implementación de Pyodide, se ha necesitado realizar un estudio riguroso acerca de cómo implementarlo de manera efectiva y práctica, puesto que actualmente están en la versión 19 y casi todos los ejemplos y explicaciones que están en la web son con la versión 17 (algo desactualizadas y realizado de manera distinta).

Para usar Pyodide, primero se debe importar la librería de Pyodide y después cargar Pyodide con la función “loadPyodide”. Para importar dicha librería se especificará la URL de los paquetes de Pyodide dentro de dicha función (Ver figura 4.16). Es importante que esta función sólo se ejecute 1 única vez a lo largo de toda la ejecución de la aplicación y para ello sólamente se cargará la primera vez que entre el usuario en el editor, independientemente si anteriormente se ha cerrado sesión sin refrescar página, puesto que sólo se puede cargar una copia de Pyodide en un ámbito global de JavaScript porque Pyodide usa variables globales

para cargar paquetes. Si se intenta cargar una segunda copia de Pyodide, la función "loadPyodide" mostrará un mensaje de error. Si se desea poder añadir posteriormente en el código importaciones de librerías, se deberá añadir anteriormente a la función "loadPackage". Se pueden añadir todos los paquetes que se deseen, pero también hay que tener en cuenta que cuantos más se añadan, más tardará Pyodide en cargarse.

Además se utiliza una función asíncrona para que cuando se llame a dicha función, mientras está cargando Pyodide, se pueda estar cargando otros elementos de la página y evitar que ésta no responda hasta que se haya completado la carga de Pyodide.

```
async function init() {

    loadPyodide({ indexURL: "https://cdn.jsdelivr.net/pyodide/v0.19.1/full/" }).then((pyodide) => {
        globalThis.pyodide = pyodide;
        pyodide.loadPackage(["numpy","matplotlib"]);
        console.log("Module loaded.");
    });
}
```

Figura 4.16.- Función para cargar Pyodide

Para poder ejecutar el código de programación del usuario, primero se debe controlar la salida y para ello hay que redireccionar temporalmente la variable "stdout" de Pyodide para posteriormente añadirle el valor del "textarea" que haya rellenado el usuario (Ver figura 4.17). Después se guarda en una variable el valor de "textarea" con la función "getValue" del editor CodeMirror y se ejecuta la función "execute_code_block". Mientras se va cargando la función "execute_code_block" se ejecuta también la función "flush()" de Pyodide para limpiar todos los resultados de ejecuciones anteriores (se puede ejecutar mientras que se está ejecutando la función para ejecutar el código porque es asíncrona). En la ejecución del código, si el resultado no ha dado un error, aparecerá el resultado correcto y en cualquier otro caso, aparecerá un mensaje de error de Pyodide.

```
function add_block() {

    pyodide.runPython(`
            import sys
            from io import StringIO
            sys.stdout = StringIO()
        `);

    let codeBlock = editor.getValue();

    console.log("EL VALOR RECOGIDO ES: ",codeBlock);

    execute_code_block(codeBlock, resultBlock);

    resultBlock.innerHTML = resultBlock;
    pyodide.runPython("sys.stdout.flush()");
    pyodide.runPython("sys.stdin.flush()");
}
```

```
async function execute_code_block(codeBlock, resultBlock) {

    console.log(codeBlock);
    let result = await pyodide.runPythonAsync(codeBlock)
    .then(output =>{

        let stdout = pyodide.runPython("sys.stdout.getvalue()");

        console.log("EL VALOR DE S ES: "+stdout);
        resultBlock.innerHTML = stdout;

        console.log("El valor de nuevo de stdout es: "+stdout);
    })
    .catch(err=>{
        console.log("RESULTADO ERRONEO: ",err);

        resultBlock.innerHTML = err;
    });
}
```

Figura 4.17.- Ejecución de código python con Pyodide

Es muy importante tener presente que Pyodide es una tecnología muy nueva, por lo que es posible que vayan surgiendo nuevas versiones en relativamente poco tiempo (actualmente la versión más reciente es la 19), por lo que que si se desea cambiar a versiones posteriores, es posible que algunos métodos o formas de cargar o ejecutar código Pyodide se realice de manera diferente.

Capítulo 5

Conclusiones y trabajo futuro

5.1.- CONCLUSIONES

Con respecto a la conclusión del proyecto, se puede decir que se ha cumplido con el objetivo principal de incorporar un editor de código online dentro de la aplicación, poder ejecutar el código, gestionar los cursos y permitir al usuario obtener conocimientos y poder aplicarlos a la hora de realizar tareas.

Con respecto a la tecnología utilizada para ejecutar código online, se ha realizado un estudio acerca de cuál sería la mejor solución para poder implementarlo de manera sencilla. Toda la investigación fue compleja debido a que la tecnología es muy nueva y está en fase beta, por lo que no había mucha información acerca de errores específicos y únicamente estaba la documentación oficial.

Al principio se intentó incorporar un cuaderno Jupyter que funcionaba completamente en el lado del cliente, pero debido a su complejidad, se optó finalmente por incorporar un editor de código online más sencillo pero manteniendo el uso de la tecnología (Pyodide) que permitía ejecutar código de programación en el lado del cliente.

Con respecto a las funcionalidades de la aplicación, se puede decir que se han superado los objetivos iniciales permitiendo mejorar la experiencia de los usuarios y facilitando en todo momento la usabilidad de la aplicación. Además se ha intentado evitar los posibles errores que pueda cometer el usuario realizando un control exhaustivo de errores y avisando en todo momento del tipo de error que se ha cometido para ayudar a solventar el problema.

El diseño de la aplicación es muy minimalista, sin ninguna complejidad utilizando únicamente CSS y permitiendo a un usuario sin conocimientos técnicos hacer uso de la aplicación sin ningún esfuerzo y de forma más cómoda.

Al comenzar el proyecto, el objetivo principal que se tenía era crear una aplicación que sea útil y funcional que supla ciertas necesidades en el futuro y pueda aumentar su funcionalidad, de manera que se pueda seguir desarrollando la aplicación para conseguir tener una aplicación robusta de ofertas de cursos. Este objetivo se ha conseguido, ya que la aplicación es funcional y permite empezar a ofertar cursos y matricular a alumnos para facilitar su aprendizaje.

5.2.- POSIBLES DESARROLLOS FUTUROS

En este punto, se explicarán algunos de los posibles desarrollos futuros que se puedan implantar en la aplicación:

- **Añadir un módulo para hacer tests online autocorregibles:** El docente de un curso tendrá la capacidad de crear pruebas con una batería de preguntas tipo test, de manera que cuando el alumno lo finalice, pueda obtener la nota de manera automática, sin la necesidad de que el docente tenga que corregir el test.

- **Tareas entregables mediante fichero:** El alumno podrá entregar la tarea mediante un fichero, el cual podrá ser descargado por el alumno para comprobar que todo está correcto y por el docente para corregirlo.

- **Utilización de ficheros CSV en el editor de código online:** El alumno podrá subir al editor de código online un fichero csv para poder trabajar con datos utilizando librerías propias de ciencia de datos.

- **Incorporación de ficheros en el material de un curso:** Además de poder incorporar enlaces en la tabla de material, también se podrá incrustar directamente ficheros PDF para facilitar al docente la prestación de material al alumno.

- **Añadir un módulo de espacio personal del usuario:** Se tratará de una sección en la que todos los usuarios tienen acceso, permitiendo guardar información que considere importante el alumno, ya sea ficheros o tareas guardadas (sin la necesidad de estar entregadas) en un espacio privado y único para cada usuario.

- **Añadir nuevos lenguajes de programación al editor de código online:** Actualmente únicamente se puede escribir código python en el editor, pero CodeMirror es compatible con hasta 60 lenguajes de programación diferentes, por lo que el usuario podrá elegir el lenguaje de programación que necesite usar dependiendo de los requerimientos de las tareas.

- **Creación de gráficas con Pyodide:** Además de codificar código en el editor, también se podrán codificar gráficas y éstas se podrán visualizar. Se pueden realizar utilizando la librería Matplotlib.

- **Creación de bandeja de entrada de mensajes para cada usuario:** Se tratará de una sección en la que pueda haber un historial de los mensajes enviados y recibidos entre los usuarios de la aplicación. Se podrá enviar un mensaje a un usuario introduciendo su correo electrónico. No se enviará un correo electrónico como tal, sino que quedará registrado un mensaje al usuario emisor y receptor del mensaje.

- **Avisos a los alumnos en un curso:** El docente de un curso podrá enviar un mensaje a todos los alumnos del curso.

- **Adición de librerías Python y propias:** El alumno podrá incluir en el código su propia librería y además se podrán añadir nuevas librerías de Python. Todas las librerías de python que se añadan en el código deberán haber estado añadidas anteriormente en el código de Pyodide (Sólo lo podrá hacer el programador), ya que se cargan todas antes de habilitar el editor y cuando se habilite, ya se podrán hacer los “includes”.

- **Resumen de calificaciones:** El alumno tendrá dentro de un curso una sección para visualizar el resumen de todas las notas que tiene en las tareas. Además se podría incluir un gráfico de notas de los alumnos con información relevante para el docente.

- **El docente podrá asignar nota a una tarea no entregada:** El profesor podrá asignarle una nota a una tarea que no ha sido entregada por el alumno y la fecha de expiración haya vencido. Actualmente sólo se puede asignar nota a las entregas de los alumnos. Normalmente la nota puesta a una entrega no realizada será de “0” o “No

presentado", actuando de la misma manera a la hora de ponderar la nota media del alumno (actuando como un 0).

- **Incorporación de algún algoritmo de ciencia de datos para predecir notas:** Se podrá añadir algún algoritmo de ciencia de datos para predecir si un alumno tiene más probabilidad de obtener calificaciones bajas y ponerlo en conocimiento del docente para proporcionar ayuda extra al alumno.

- **Bloqueo de cuenta de un usuario por parte del administrador:** El administrador tendrá la posibilidad de bloquear las cuentas de los usuarios, sin la necesidad de eliminarlas.

- **Recuperación de contraseña:** El usuario podrá de alguna forma recuperar su contraseña olvidada para poder volver a entrar a su acceso identificado.