
*Facultad*
*de*
*Ciencias*

# HERRAMIENTA DE APOYO PARA LA CONSULTA Y EVALUACIÓN DE TAREAS MOODLE A TRAVÉS DE WEBSERVICES (Support tool for querying and evaluation of Moodle tasks through WebServices)

Trabajo de Fin de Grado
para acceder al

## GRADO EN INGENIERÍA INFORMÁTICA

Autor: Jesús Campo López

Director: Héctor Pérez Tijero

Co-Director: Mario Aldea Rivas

Septiembre - 2022

## Resumen

Este proyecto consiste en el análisis, diseño y desarrollo de una herramienta de apoyo para la autoevaluación de tareas de Moodle mediante el uso de los WebServices que proporciona la propia plataforma. El objetivo es conseguir facilitar el proceso de autoevaluación de prácticas de programación para dos casos de uso concretos, tanto para prácticas de evaluación única (exámenes) como para prácticas de evaluación interactiva (prácticas de aula).

Mediante el uso de una librería previamente estudiada llamada *moodleteacher*, la cual hace uso de los WebServices, se va a construir una librería de utilidades que recoge diferentes funcionalidades específicas en cuanto al proceso general de autoevaluar tareas de Moodle. Esta serie de funcionalidades que abarca la librería que se va a implementar, servirá de apoyo para la construcción de las dos herramientas principales que se van a construir, para cubrir los dos casos de uso de evaluación (única o interactiva).



## Palabras Clave

*Moodle, WebServices, Evaluación única, Evaluación interactiva, Tarea, Moodleteacher*

# Abstract

This project consists of the analysis, design and development of a support tool for the self-assessment of Moodle tasks through the use of WebServices provided by the platform itself. The objective is to facilitate the self-assessment process of programming practices for two specific use cases, both for single assessment practices (exams) and for interactive assessment practices (classroom practices).

Through the use of a previously studied library called moodleteacher, which makes use of WebServices, a library of utilities will be built and it includes different specific functionalities regarding the general process of self-assessing Moodle tasks. This series of functionalities that includes the library that is going to be implemented, will support the construction of the two main tools that are going to be built, to cover the two evaluation use cases (single or interactive).



# Key Words

*Moodle, WebServices, Unique assessment, Interactive assessment, Task, Moodleteacher*

# *Índice*

# 1. Introducción

## 1.1 Contexto

Facilitar el desarrollo de tareas o acciones que suelen resultar rutinarias en el día a día es uno de los principales objetivos que se buscan en muchos ámbitos de la ciencia, sobre todo en el campo de la informática. Una de las acciones más recurrentes que afrontan tanto los profesores de las universidades como los docentes de muchos colegios o institutos es la de corregir o evaluar tareas. Hoy en día la mayor parte de las prácticas o trabajos se entregas y evalúan a través de diferentes plataformas de e-learning las cuales dan un soporte al centro educativo para facilitar este tipo de acciones. Existen muchos casos que hasta las pruebas y los exámenes de ciertas asignaturas se realizan vía estas plataformas. A raíz de la relevancia que estas han adquirido, la informática presenta innumerables formas de progreso de cara a optimizar diferentes procesos habituales en ellas. En este caso, en la Universidad de Cantabria se trabaja con la plataforma Moodle [1], en la cual los alumnos tienen acceso a los cursos en los que están matriculados. En estos cursos se pueden encontrar diversas tareas para entregar, cuestionarios o teoría sobre la asignatura entre otras muchas cosas. Por su parte, los profesores hacen uso de la plataforma para subir contenidos y evaluar diferentes tareas o cuestionarios a sus cursos.

De la evaluación de tareas proviene la principal motivación de este proyecto ya que resulta muy costoso para los profesores la acción de calificar a los alumnos vía Moodle, ya que tienen que descargar las diferentes entregas de todos los alumnos, calificarlas a través de sus respectivas herramientas, anotar las calificaciones y finalmente subirlas a la plataforma. Estos procesos resultan muy largos y costosos a los docentes.

En la actualidad se utiliza *evalcode* [2], un plugin de Moodle, que ha sido desarrollado en la Universidad de Cantabria como resultado de varios TFGs [3][4]. Este plugin realiza la autoevaluación de ejercicios de programación realizados por los alumnos en un servidor de Moodle paralelo al de la Universidad de Cantabria. El uso de este servidor paralelo genera gran cantidad de problemas, ya que requiere actualizarse con las nuevas versiones de Moodle.

Los profesores pueden necesitar la evaluación de prácticas para dos situaciones diferentes. Existe la evaluación única, que se realiza para todos los alumnos una vez se ha finalizado el plazo de entrega y las calificaciones son definitivas(p. ej. Examen). Y por otro lado, está lo que se va a denominar evaluación interactiva, en la que se van evaluando las entregas a medida que van siendo realizadas por los alumnos. En este tipo de evaluación, existe la posibilidad de que los alumnos puedan modificar sus entregas, por lo que se deberán volver a evaluar.

En este TFG, se va a implementar una herramienta de apoyo para el propio servidor de la Universidad capaz de automatizar estos procesos para los dos tipos de evaluación y conseguir que los profesores o los docentes que la utilicen puedan agilizar su trabajo.

## 1.2 Objetivos

El objetivo principal de este trabajo consiste en desarrollar una herramienta que facilite la evaluación automática de prácticas de programación a través de la plataforma de Moodle. Como se ha comentado, actualmente se dispone de una extensión de Moodle que facilita la autoevaluación de ejercicios de programación. Sin embargo, esta extensión requiere de un mantenimiento continuo y su integración con el servidor oficial de Moodle de la UC resulta complejo. Por ello, este TFG propone crear una herramienta de escritorio que se comunique con el servidor Moodle a través de la interfaz de WebServices para gestionar de forma sencilla las tareas de un curso de moodle y las entregas realizadas por los alumnos (p.ej., para obtener información sobre la misma tarea, descargar las entregas de una tarea o para asignar calificaciones y archivos de retroalimentación de forma automatizada).

Como punto de partida, existe una librería Python denominada *moodleteacher* [5] que implementa parte de la funcionalidad requerida. Este TFG se va a encargar de crear unas herramientas complementarias a esta librería para poder cumplir con funcionalidades más generales y finalmente poder automatizar el proceso total de autoevaluar prácticas de programación.

Para cumplir con este objetivo principal, se establece una lista de subobjetivos que nos permitan alcanzarlo:

- Desarrollar una librería de utilidades que proporciona funciones de alto nivel que extienden la funcionalidad dada por la librería *moodleteacher*.
- Construir una versión funcional de la herramienta, la cual permita la comunicación con el servidor de Moodle a través de la interfaz de los WebServices.
- Generar documentación para facilitar el uso y el desarrollo de futuras extensiones, así como para comprender las funcionalidades implementadas en la herramienta.
- Desarrollar las dos aplicaciones que satisfacen los dos tipos de evaluaciones mediante el uso de las funcionalidades implementadas por la librería de utilidades y de los WebServices.

## 1.3 Organización

En esta memoria se describe el proceso total de análisis, diseño e implementación de la herramienta de apoyo para la autoevaluación de prácticas de programación en Moodle.

En el capítulo 2 se detallan las herramientas que se han utilizado para la elaboración del proyecto, como lo son la plataforma Moodle, los WebServices que permiten la comunicación con esta y la librería de *moodleteacher* que hace uso de estos servicios y proporciona una base para la herramienta que se va a implementar.

En el capítulo 3 se explican los requisitos principales de la misma herramienta, destacando el autoevaluar ejercicios de programación en los casos de exámenes y en los casos de prácticas de aula.

En el capítulo 4 correspondiente con la fase de diseño, se presenta una arquitectura por capas que describe las diferentes partes de la implementación y la conexión entre las mismas. En la capa superior se encuentra la construcción de las aplicaciones finales para la evaluación de tareas de Moodle a través de los WebServices. Estas

aplicaciones hacen uso de una serie de funciones previamente implementadas, entre las que destacan funcionalidades como, por ejemplo, descargar las entregas de los alumnos o calificar la entrega de un alumno. Estas funciones se basan en las clases y funciones proporcionadas por la librería *moodleteacher* y en los WebServices que permiten la comunicación con un servidor de Moodle.

En el capítulo 5 se detallan algunas de las funciones más complejas, con detalles más concretos sobre la parte programática, junto a pseudocódigos de estas, en la fase de implementación.

En el capítulo 6, correspondiente con la fase de pruebas, se detalla el proceso de configuración de un servidor de pruebas de Moodle, el cual ha sido necesario. A su vez, se explican las diferentes pruebas unitarias que se han realizado para cada función implementada en la librería de utilidades así como las pruebas de integración sobre las aplicaciones finales con el objetivo de comprobar el correcto funcionamiento conjunto de todas las funcionalidades.

En el capítulo 7, finalmente se sacan conclusiones tanto del proyecto en sí como de lo que este puede dar lugar, ya que la herramienta deja un amplio margen de evolución y de expansión.

# 2. Herramientas

## 2.1 Moodle

La principal herramienta sobre la que se basa este trabajo es Moodle. Se trata de una plataforma de e-learning que conforma un sistema educativo que permite cubrir las necesidades de los profesores y estudiantes, diseñado para crear y gestionar espacios de aprendizaje adaptados. Tanto en la Universidad de Cantabria como en muchas otras, se utiliza esta plataforma para facilitar los procesos de envío de tareas, prácticas o apuntes entre alumnos y profesores. Un servidor de Moodle se conforma de diferentes cursos, cada uno correspondiente a una asignatura. Estos cursos están formados por uno o varios profesores, los cuales pueden manipular el mismo y añadir o eliminar contenido, y por una serie de alumnos que podrán acceder a todo el contenido proporcionado por el mismo.

Dentro de las posibilidades que tiene un profesor dentro de su curso, existen 7 módulos que se pueden crear dentro de este: Foro, Consulta, Glosario, Cuestionario, Recurso, Encuesta y *Tarea*. Este último módulo mencionado es con el que se va a trabajar durante este proyecto. Las tareas pueden ser tanto prácticas, ejercicios o exámenes, los cuales los alumnos del curso deben de realizar y entregar antes de un plazo fijado por el profesor. Los profesores pueden evaluar estas entregas en la propia plataforma, para lo cual se les permite añadir para cada entrega una calificación y, opcionalmente, un fichero de retroalimentación.

La plataforma contiene diferentes formas de acceso a la misma, siendo vía web la más común de todas, ya que admite la mayoría de los navegadores de Internet. También dispone de aplicación móvil tanto para Android como para iOS. A su vez, Moodle tiene una larga lista de API's (Conjunto de subrutinas, funciones y procedimientos ofrecidos para ser utilizados por otro software) donde entre las más utilizadas se pueden destacar: API de acceso, API de manipulación de datos, API de navegación o API de ficheros. Se pueden encontrar hasta más de 50 API's diferentes. En concreto, en este proyecto se va a utilizar la API de los Web Services.

## 2.2 API de WebServices de Moodle

Un servicio web, como su propio nombre indica, es una tecnología que utiliza una serie de protocolos y estándares que sirven como vía de comunicación y de intercambio de datos entre dos aplicaciones. Por lo tanto, cada servicio web facilita un servicio concreto a través de internet.

Como ya se mencionó, durante este proyecto se hace uso de la API de Web Services de la plataforma Moodle que, en pocas palabras, permite exponer diferentes funciones particulares, generalmente funciones externas, como servicios web. Contiene una lista innumerable de funciones, donde cada una realiza una labor concreta, desde crear un curso cualquiera hasta obtener una lista de los alumnos de un mismo curso. A lo largo de la implementación de este proyecto, se hacen uso de muchos de los diferentes Web Services que proporciona esta API de Moodle.

En vistas generales, algunos de los servicios más relevantes y utilizados durante la implementación de este proyecto son los siguientes:

- ***'mod_assign_get_submission_status'***: Proporciona datos acerca de la entrega de un alumno sobre una tarea.
- ***'mod_assign_save_grade'***: Permite subir la calificación de un alumno respecto a una tarea.
- ***'mod_assign_get_submissions'***: Retorna una lista de todas las entregas realizadas por los alumnos sobre una tarea.
- ***'core_user_get_users_by_field'***: Proporciona todos los datos sobre uno o más usuarios del servidor de Moodle.

## 2.3 Librería moodleteacher

### 2.3.1 Introducción

La librería de moodleteacher se encuentra escrita en Python y está enfocada y destinada hacia los profesores con cursos en Moodle para poder automatizar la gestión de estos mismos cursos o poder calificar entregas de estos gracias al uso de los 'Web Services' de Moodle.

En la propia documentación de la librería aparecen los pasos a seguir para instalar la librería junto a ejemplos muy básicos de como importar la librería desde una sesión interactiva de Python y establecer la primera conexión con el servidor Moodle para comenzar a interactuar con diferentes elementos de este.

La librería se estructura en varios ficheros Python, en concreto en catorce, incluyendo el de inicialización (__init__.py). Cada uno está enfocado en un elemento distinto de un servidor de Moodle, como pueden ser los cursos, los usuarios o las entregas, y cada uno de estos ficheros contiene una o más clases relacionadas con dicho elemento. El diagrama de clases de la librería *moodleteacher* es el mostrado en la Figura 1, donde se pueden ver las relaciones entre las clases de la librería.

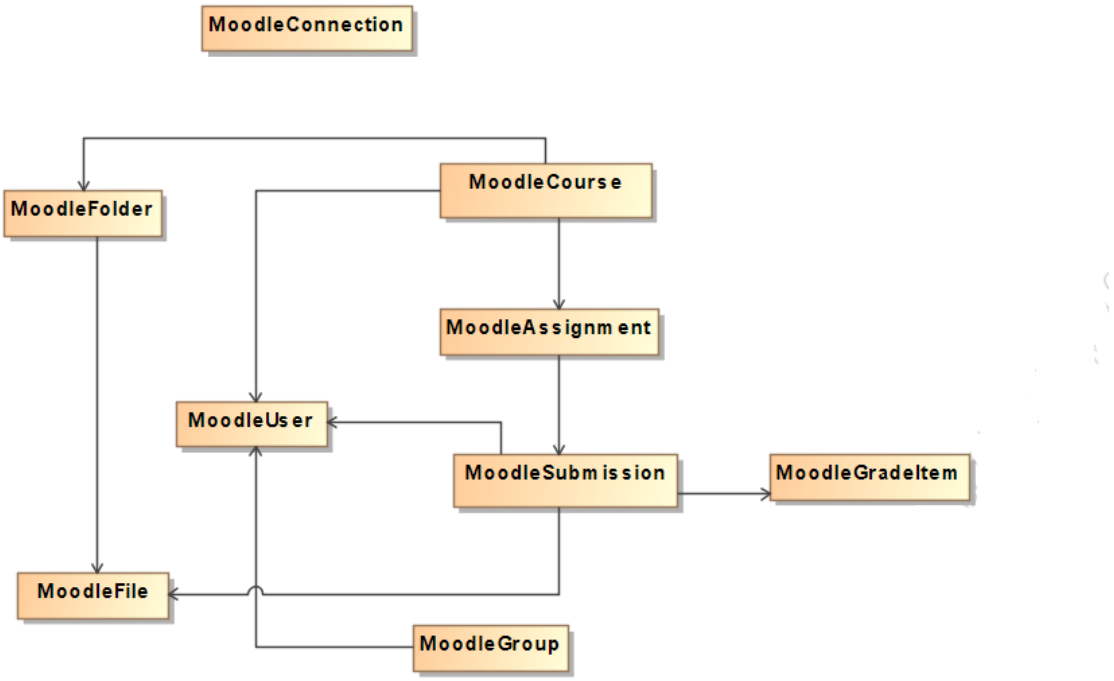


**Figura 1.** ***Diagrama de clases de la librería moodleteacher.***

### 2.3.2 Funciones

No todas las clases o funciones contenidas en las diferentes clases se utilizan directamente, por lo que se van a explicar las clases más utilizadas y las que tienen verdadera relevancia en el trabajo, junto con una breve presentación de sus funciones principales:

Clase '***MoodleConnection'***: Establece la conexión con el servidor de Moodle y se encarga de configurar dicha conexión. Contiene una única función principal para establecer la conexión, la cual recibe los siguientes parámetros:

- *'moodle_host'*: Url de la instalación del servidor de Moodle.
- '*token':* Token del cliente que se conecta al servidor.
- '*is_fake'*: Booleano. En caso de activarse crea una conexión falsa para realizar tests.
- '*interactive':* Booleano. En caso de activarse muestra un prompt interactivo para introducir los dos primeros parámetros mencionados.

Clase '***MoodleCourse'***: Corresponde con un curso de Moodle y permite interactuar y obtener información acerca del propio curso, de los módulos que lo componen y de los usuarios pertenecientes al mismo. Sus funciones principales son las siguientes:

- *'from_course_id'*: Obtiene un curso a partir de su identificador.
- '*get_user':* Proporciona un usuario del curso a partir de su identificador.
- *'get_group_members':* Proporciona un grupo de usuarios a partir del identificador de su grupo.
- *'assignments':* Devuelve una lista de todas las tareas que existan en el curso.

Clase ***'MoodleAssignment'****:* Corresponde con una tarea de un curso de Moodle y permite interactuar y obtener información de la propia tarea. Algunas de las funciones principales son las siguientes:

- '*from_assignment_id':* Devuelve una tarea a partir de su identificador.
- *'get_user_submission':* Proporciona la entrega de un usuario concreto respecto de la tarea.
- *'submissions':* Retorna una lista de todas las entregas realizadas por los alumnos sobre la propia tarea.

Clase *'**MoodleSubmission'***: Corresponde con una entrega de un alumno sobre una tarea determinada de un curso y permite obtener información, manipular la calificación y cargar retroalimentación asociada a la entrega del alumno. Sus funciones principales son las siguientes:

- '*load_feedback':* Devuelve la retroalimentación correspondiente a la entrega.
- *'save_feedback':* Añade retroalimentación en formato de texto o de fichero a la entrega.
- *'load_grade':* Devuelve la calificación correspondiente a la entrega.
- *'save_grade':* Añade una calificación a la entrega.
- *'is_empty':* Comprueba si la entrega está vacía.
- *'is_graded':* Comprueba si la entrega está calificada.

Clase ***'MoodleUser'***: Corresponde con un usuario del servidor de Moodle y permite obtener información sobre este, como los cursos en los que se encuentra o sus datos personales. Relacionada con esta clase, existe la clase ***'MoodleGroup'*** que se trata de un grupo de usuarios, con su propio nombre e identificador. La única función relevante de la clase '*MoodleUser'* es la siguiente:

- '*from_userid':* Devuelve un usuario a partir de su identificador.

Clase '***MoodleFile'***: Corresponde con un archivo de un curso de Moodle, desde temario del propio curso hasta un archivo entregado por un alumno en una tarea. Esta clase permite obtener y cargar estos ficheros desde el servidor y obtener información acerca de su tamaño, ubicación o formato. Sus funciones principales son las siguientes:

- '*from_url':* Proporciona un archivo a partir de una url dada.
- *'from_local_file':* Proporciona un archivo a partir de su ruta
- *'unpack_to':* En caso de que el archivo sea comprimido, lo descomprime en un directorio concreto dado, y en caso contrario, lo descarga en el directorio de trabajo.

# 3. Análisis de Requisitos

## 3.1 Funcionales

Previo al diseño de la estructura de las diferentes funcionalidades que se van a implementar, se parte de una base compuesta por dos requisitos funcionales principales.

- Realizar la evaluación de las entregas de una tarea una vez finalice el plazo establecido para su realización. Esta evaluación será definitiva y se llamará *evaluación única*
- Realizar la evaluación de las entregas de una tarea a medida que los alumnos las van realizando, y en caso de que los alumnos modifiquen sus entregas, deberán ser recalificadas. Recibirá el nombre de *evaluación interactiva.*

La *evaluación única* se puede dar en casos como los de un examen, los cuales requieren una única calificación que será definitiva. Una vez realizadas las entregas oportunas por parte de los alumnos sobre una tarea concreta, la aplicación ha de ser capaz de descargar todas las entregas de los alumnos, y sobre estas entregas se ejecutará una herramienta de autoevaluación diseñada por el profesor cuyo desarrollo queda fuera de los objetivos de este proyecto. Finalmente, la aplicación permitirá subir las calificaciones junto con los posibles documentos de retroalimentación generados.

La *evaluación interactiva* se da en casos como los de una práctica de aula, donde a medida que los alumnos van realizando la práctica, van subiendo sus entregas a la plataforma Moodle, de manera que cada vez que la suben reciben una calificación. Más en adelante, los alumnos podrán modificar sus entregas y subirlas de nuevo, de manera que estas nuevas entregas deban ser calificadas de nuevo. Por lo tanto, la aplicación necesita estar comprobando constantemente si se han realizado nuevas entregas o si alguna entrega ya existente ha sido modificada. En caso afirmativo, se descargará la entrega y se calificará con una herramienta de autoevaluación. Finalmente, la aplicación subirá su calificación y un posible fichero de retroalimentación.

## 3.2 No Funcionales

Para el diseño y desarrollo de este proyecto, existen un conjunto de requisitos impuestos por decisiones de los profesores promotores del mismo:

- Uso de la plataforma de Moodle. Se utiliza esta plataforma de e-learning debido a que es la utilizada por la Universidad de Cantabria y tanto los alumnos como los profesores del centro están altamente familiarizados con la misma.
- Utilización de la librería *moodleteacher.* Debido a que se cuenta con una cierta experiencia previa de uso y de familiarización con la librería, va a ser utilizada como base de implementación de las utilidades de este proyecto y como acceso a los WebServices de Moodle.
- Lenguaje Python. Se usa este lenguaje principalmente porque la librería *moodleteacher* se encuentra escrita en el mismo.

## 3.3 Diagramas de secuencia

Cada uno de los dos requisitos funcionales, corresponde con un caso de uso concreto. A continuación se muestran los diagramas de secuencias de los dos casos de uso, en los que se ven reflejados los procesos básicos que cada una de las aplicaciones de autoevaluación van a tener que realizar.

Como se puede observar en la Figura 2, se detalla el diagrama de secuencias de la aplicación que cubre el caso de uso de una evaluación única. Una vez se ejecuta la aplicación, se descargan todas las entregas realizadas por los alumnos sobre la tarea indicada, se evalúan bajo una herramienta de autoevaluación, y finalmente se suben las calificaciones y las retroalimentaciones concretas.

El diagrama de secuencias para el caso de uso de la evaluación interactiva es el mostrado en la Figura 3. Tras ejecutar la aplicación, se comienza a comprobar si se han realizado alguna nueva entrega o si alguna ya realizada ha sido modificada. Al detectar una entrega para evaluar, se descarga y se califica con una herramienta de autoevaluación. Finalmente se sube su calificación y su archivo de retroalimentación pertinente. Este proceso se realiza durante un bucle continuo.

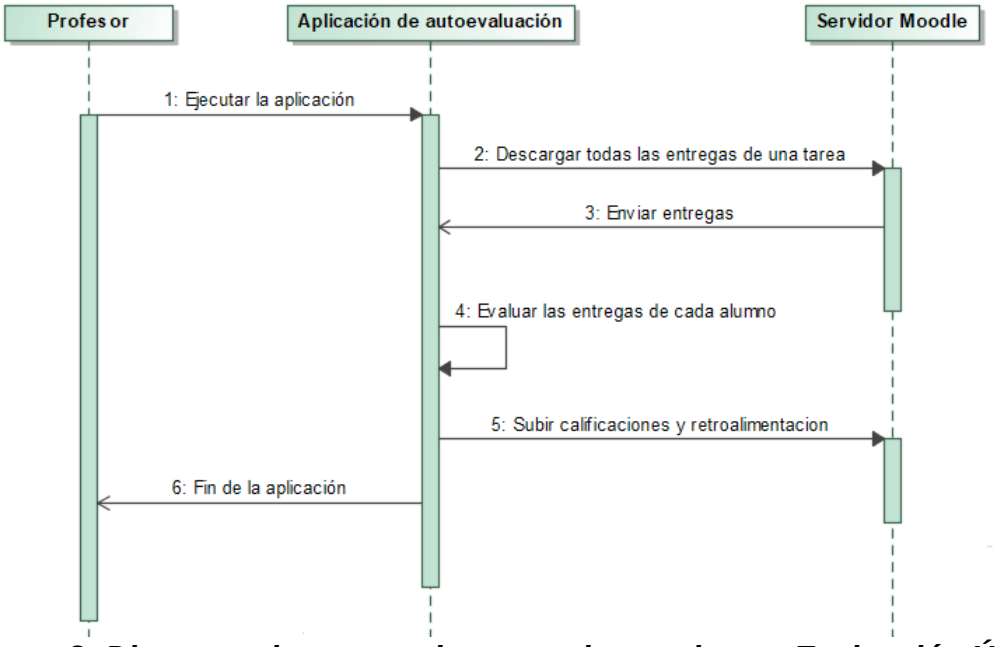


**Figura 2.** ***Diagrama de secuencias para el caso de uso Evaluación Única.***

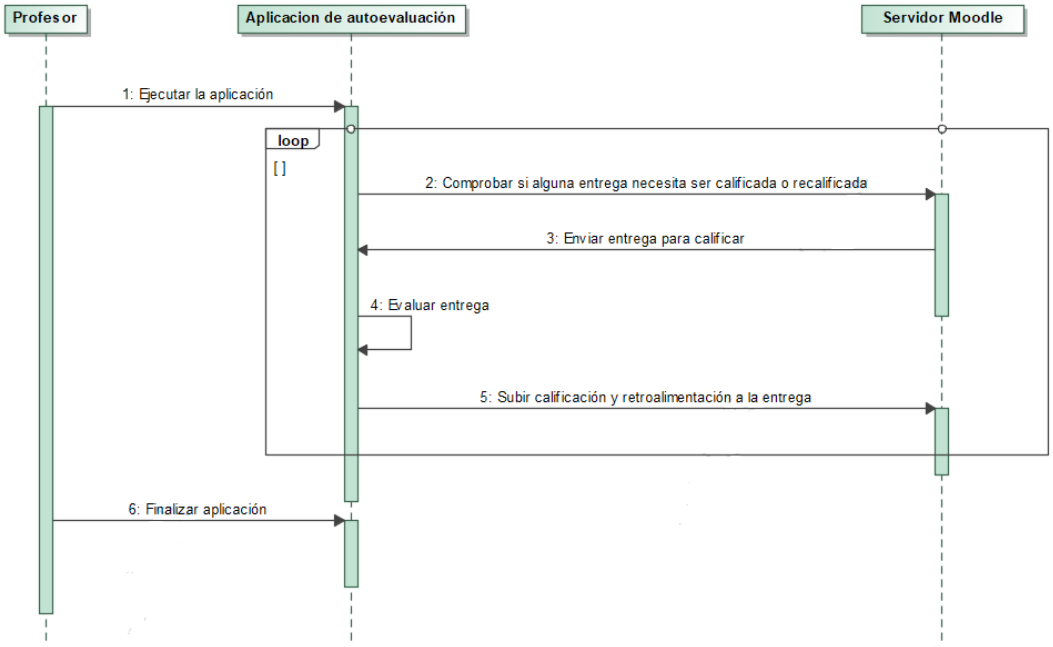


**Figura 3.** ***Diagrama de secuencias para el caso de uso Evaluación Continua.***

# 4. Diseño

## 4.1 Arquitectura en Capas

Este proyecto cuenta con diferentes capas entre las cuales se distribuyen las diferentes funcionalidades que van a ser implementadas y las ya existentes que van a ser utilizadas. La arquitectura por capas del proyecto es la mostrada en la Figura 4.

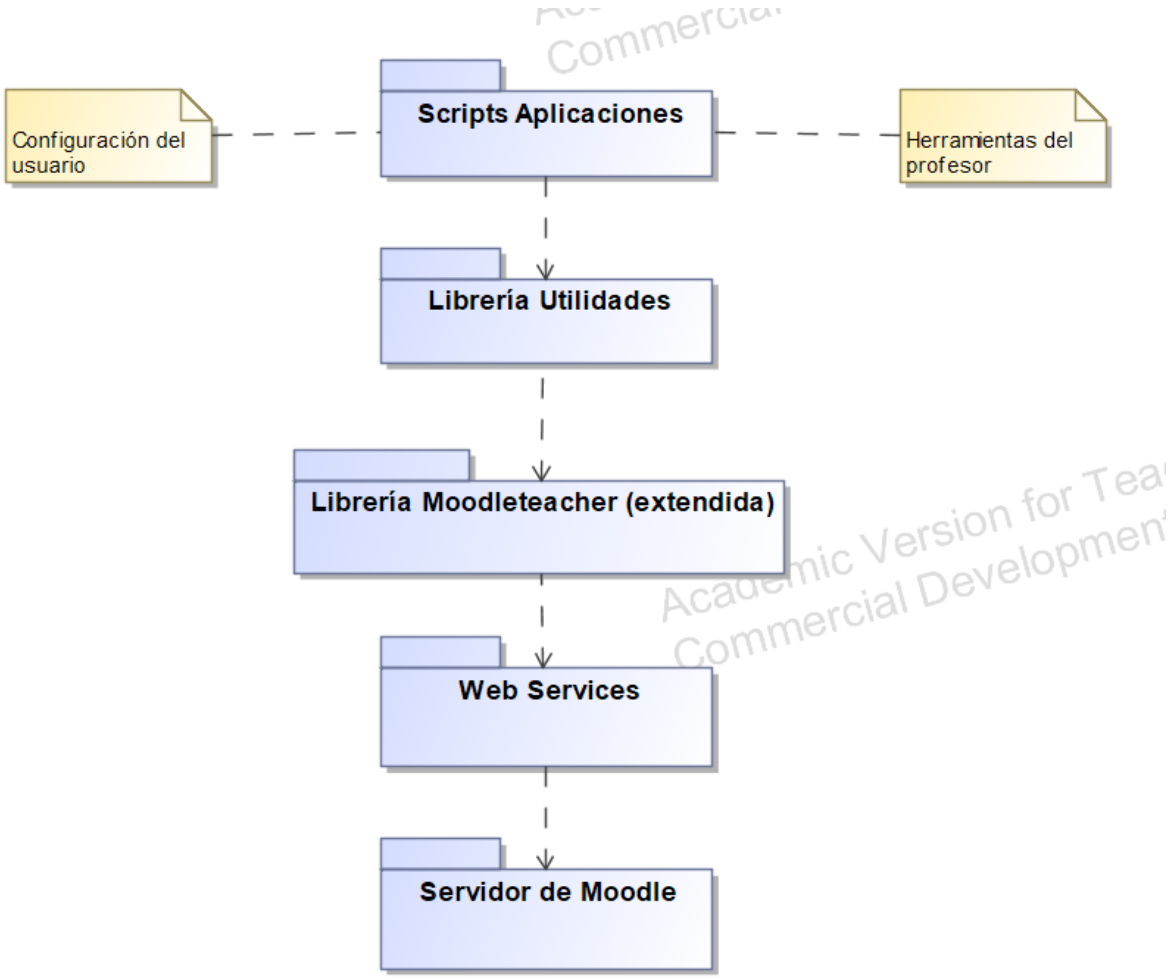


**Figura 4.** ***Arquitectura por capas de la herramienta***

Cada una de las capas, hace uso de la capa inferior. Los scripts de las aplicaciones, los cuales son dos como ya se comentó en la fase de requisitos, hacen uso de las funciones individuales, los cuales a su vez se basan en el uso de la librería moodleteacher, y así sucesivamente.

Con respecto a las capas, la parte de la implementación de este proyecto abarca las capas de Scripts de aplicaciones, de utilidades y la extensión de la librería *moodleteacher*. Esta última se trata de una pequeña extensión a la librería original,

donde se han añadido varias funciones para agilizar los procesos de las capas superiores.

Los scripts de aplicaciones, como aparece a sus lados, necesitan de una configuración del usuario y de una herramienta del profesor para autoevaluar prácticas.

## 4.2 Descripción de las Capas

**Moodle:** Sobre esta plataforma se van a realizar todas las manipulaciones y se va a interactuar para poder facilitar los procesos de evaluación a los diferentes profesores.

**Web Services:** Los servicios web harán de intermediarios entre un servidor de Moodle y las funciones implementadas por la librería de *moodleteacher*. Estos servicios web nos proporcionan diversas funciones para dar y recibir información del servidor de Moodle.

**Librería *moodleteacher*:** Esta librería proporciona una gran lista de clases referidas a módulos y objetos de un servidor de Moodle, cuyas funcionalidades hacen uso de los servicios web para poder interactuar con dicho servidor.

**Scripts Utilidades:** Esta capa abarca todas las funciones individuales que se van a implementar con el objetivo final de elaborar las dos aplicaciones de la capa superior, donde se realizará el proceso global de autoevaluar las prácticas de los alumnos. Estas funciones hacen uso directo de las clases y funciones proporcionadas por la librería *moodleteacher,* así como de los WebServices.

Para distribuir las diferentes funciones que se han querido implementar, se crean dos ficheros principales. El primer fichero será '*entregas.py*', que contendrá todas las funciones relacionadas con las entregas de los alumnos, como descargar las entregas o comprobar si existen modificaciones en una entrega. El segundo fichero será '*hojaCalificaciones.py*', que contendrá todas las funciones relacionadas con las hojas de calificaciones de una tarea, como descargar una hoja de calificaciones de una tarea de Moodle o subir calificaciones a las entregas de los alumnos a partir de una hoja de calificaciones. Con estos ficheros preestablecidos, las funciones que serán

implementadas con el objetivo de construir las aplicaciones finales se reparten siguiendo la siguiente estructura:

- ***'entregas.py':***
    - ***descargaTodas():*** *Descarga todas las entregas realizadas por los alumnos sobre una misma tarea.*
    - ***descargaPorUsuario():*** *Descarga la entrega de un alumno concreto sobre una tarea.*
    - ***existenModificaciones():*** *Comprueba si existen modificaciones en una entrega de un alumno sobre una tarea.*
- ***'hojaCalificaciones.py':***
    - ***subirNota():*** *Sube la calificación y retroalimentación a la entrega de un alumno sobre una tarea.*
    - ***descarga():*** *Descarga la hoja de calificaciones de una tarea concreta.*
    - ***subir():*** *Sube las calificaciones y retroalimentaciones a las entregas de los alumnos correspondientes en base a una hoja de calificaciones.*
    - ***apuntarNotas():*** *Anota las calificaciones correspondientes a los alumnos en la hoja de calificaciones de la tarea que se califica.*

**Scripts Aplicaciones:** Esta capa contiene las dos aplicaciones generales que van a ser implementadas en este proyecto, ya mencionadas en la fase de requisitos. Cada una de las dos aplicaciones constará de un script en Python.

El primer script recibe el nombre de '*EvaluacionUnica.py'.* Este será capaz de realizar el proceso global de autoevaluar todas las entregas realizadas por los alumnos sobre una misma tarea una única vez. Desde descargar todas las entregas, evaluarlas y anotar sus calificaciones en una hoja, y a partir de la misma subir dichas calificaciones junto a su retroalimentación al servidor de Moodle correspondiente.

El segundo script recibe el nombre de '*EvaluacionInteractiva.py'*. Este será capaz de realizar el proceso global de autoevaluar las entregas realizadas por los alumnos sobre una misma tarea todas las veces que se requerido durante un periodo de tiempo arbitrario, es decir, estará constantemente comprobando si las entregas de los alumnos han sufrido modificaciones, y en caso de que las hayan sufrido, se volverán

a autoevaluar y actualizar tanto su calificación como su retroalimentación. Este proceso se hará durante un periodo de tiempo indefinido.

Ambas aplicaciones necesitarán de una respectiva configuración de usuario previa, donde el usuario que vaya a hacer uso de estas aplicaciones, que generalmente será un profesor, necesita anotar en un fichero de texto los datos necesarios para establecer la conexión con el servidor de Moodle que se requiera y para realizar los procesos pertinentes. Estos datos serán la IP del servidor del Moodle, el *token* del usuario que va a usar la aplicación y el nombre completo del mismo usuario. El fichero que contendrá estos datos recibe el nombre de '*ConfiguracionUsuario.txt'*. En caso de que los datos necesiten ser cambiados, dichas modificaciones se harán directamente en el fichero nombrado.

A su vez, las dos aplicaciones necesitan de una herramienta de autoevaluación para realizar el proceso de calificar las entregas de los alumnos. Esta herramienta debe de ser un script encargado de evaluar una entrega, el cual se debe de adaptar a un formato que devuelva una calificación y genere un fichero de retroalimentación. La calificación deberá de ser un numero entero entre 0 y 100, y el fichero de retroalimentación generado deberá de recibir el nombre de *'Analysis.html'*, el cual se ubicará dentro de una carpeta que se llamará '*Output'*. La herramienta que vaya a ser empleada deberá de cumplir todas estas condiciones para poder ser complementaria a las dos aplicaciones implementadas en este proyecto.

# 5. Implementación

## 5.1 Pseudocódigos desarrollados de la librería de utilidades

Para la implementación de este trabajo y de las diversas funcionalidades explicadas en los apartados anteriores, al igual que en la librería *moodleteacher,* se ha utilizado el lenguaje de programación Python.

Se implementa una librería intermedia que incluye funciones de mayor nivel que las definidas en la librería *moodleteacher*. Cada función implementada hace uso de las clases y funcionalidades que contiene *moodleteacher* con el objetivo de realizar una funcionalidad avanzada con la que construir las aplicaciones. Por tanto, se han de implementar dichas funciones ya mencionadas en la fase de *Diseño.*

```
descargaTodas(conexión, curso, tarea):
    crear directorio para almacenar todas las entregas
    recorrer todas las entregas de la tarea:
        si hay algún fichero subido a la entrega:
            obtener el usuario que ha realizado la entrega
            crear subdirectorio para almacenar la entrega del usuario
            si hay comentarios en la entrega:
                mostrar comentarios por consola
            recorrer los archivos de la entrega:
                guardar archivo en el subdirectorio
    mostrar por consola el número de entregas realizadas
retornar los usuarios que han realizado entrega junto al nombre de los ficheros
```

**Pseudocódigo 1. *Función "descargaTodas".***

El *Pseudocódigo 1* corresponde con la función *descargaTodas*, la cual se encarga de descargar todas las entregas realizadas por los alumnos sobre una misma tarea.

La función recibe como parámetro un objeto de la clase *MoodleConnection* denominado *conn*, el cual contiene la conexión con el servidor de Moodle que queremos manipular e interactuar con él. Los otros dos parámetros son *course*, que se trata de un objeto de la clase *MoodleCourse* que hace referencia al curso donde

se encuentra ubicada la tarea, y *assignment*, que se trata de un objeto de la clase *MoodleAssignment* que hace referencia a la tarea de la cual se quieren descargar todas sus entregas.

Antes de comenzar con el proceso, cabe destacar la estructura de almacenamiento que se ha llevado a cabo a la hora de descargar las entregas. Primero se tiene un directorio general con el nombre de '*Entregas_nombredelatarea'*, el cual a su vez contendrá un número de subdirectorios igual al número de entregas que vayan a ser descargadas. Cada uno de estos subdirectorios adquirirá el nombre de *'nombredelatarea_nombredelalumno',* y contendrá todos los archivos relacionados con la entrega de ese alumno. En este mismo directorio, es donde la herramienta de autoevaluación deberá crear un subdirectorio con el nombre de *Output* dentro del cuál se generará el fichero de retroalimentación con el nombre de *Analysis.html* correspondiente con la entrega del alumno al que corresponde el directorio.

Durante este trabajo solo se han considerado tareas que requieran la entrega de un solo fichero. Partiendo de esta base, lo primero que hace la función es crear el directorio general con el nombre de '*Entregas_nombredelatarea'* en caso de que no exista. Se inicializa una lista que contendrá los usuarios que han realizado una entrega sobre la tarea, y otra lista que contendrá respectivamente el nombre del fichero que han entregado cada usuario.

Posteriormente se recorren todas las entregas de la tarea, las cuales se obtienen con el uso de la función *submissions* de la clase *MoodleAssignment,* en este caso para la tarea que se recibe como parámetro. Para cada entrega, se comprueba si contiene algún fichero, ya que puede darse el caso de una entrega vacía. Comprobado si la entrega no está vacía, con el identificador del usuario que ha realizado la entrega y mediante la función *from_userid* de la clase *MoodleUser*, se obtiene el usuario completo, el cual se almacena en la lista de usuarios. Luego, en caso de que no exista, se crea un subdirectorio para dicho usuario dentro del directorio general, bajo el nombre de *'nombredelatarea_nombredelalumno'.* Se comprueba si existe algún comentario del usuario en la entrega, y en caso de haberlo se muestra por consola para que pueda ser visualizado por el profesor durante el proceso. Finalmente, se descarga el fichero entregado por el alumno en el subdirectorio recientemente creado con la función *unpack_to* de la clase *MoodleFile,* y se almacena su nombre en la lista

de ficheros. Se muestra por consola el número de entregas realizadas en comparativa con el número de alumnos matriculados en el curso, y se retornan la lista de usuarios y la lista de ficheros.

Existen casos en los que solamente se quiera descargar una sola entrega de un usuario, como puede ser el caso de una evaluación continua en el que tan sólo un usuario haya realizado modificaciones en su entrega y se tenga que recalificar. Para este caso en el que solo se tenga que recalificar una entrega, es muy costoso tener que descargar de nuevo todas las entregas y volver a recalificarlas, ya que habrá casos en que se utilicen herramientas de calificación muy sofisticadas y complejas, las cuales lleven mucho tiempo. Por este motivo, se implementó la función encargada de descargar una sola entrega, bajo el nombre de *'descargaPorUsuario'*. Esta función tiene la diferencia respecto a la función *descargaTodas* de que recibe un parámetro extra, *userid*, haciendo referencia al identificador del usuario del que se quiere descargar su entrega. Realiza un proceso idéntico simplemente que al recorrer todas las entregas de la tarea pasada como parámetro, solamente descarga la entrega del usuario cuyo identificador se ha pasado como parámetro, y se retorna el usuario y el nombre del fichero que ha entregado.

```
subirNota(conexión, tarea, id usuario, calificación, texto feedback, archivo feedback):
    recorrer todas las entregas de la tarea:
        si el id de usuario de la entrega == id usuario:
            subir calificación y feedback para la entrega
```

**Pseudocódigo 2*. Función "subirNota".***

El *Pseudocódigo 2* corresponde con la función *subirNota*, la cual se encarga de subir la calificación y la retroalimentación correspondiente a un alumno sobre una tarea.

Se reciben como parámetros el objeto de la conexión al servidor de Moodle, la tarea y el identificador de usuario ya explicados en previas funciones. Además, se reciben la calificación en formato numérico flotante, feedback en formato de texto, y feedback en formato de archivo. La función lo primero que hace es recorrer todas las entregas de la tarea dada con el uso de la función *submissions* de la clase *MoodleAssignment*

hasta encontrar una entrega cuyo autor coincida con el identificador de usuario pasado como parámetro. Una vez encontrada la entrega que se busca, se procede a subir la calificación al servidor de Moodle, para lo que se utiliza la función *save_grade* de la clase *MoodleSubmission*.

La función *save_grade* hace uso del Web Service llamado '*mod_assign_save_grade'* que ya ha sido previamente explicado en el apartado de los Web Services. Este servicio, para poder subir un archivo de feedback a una entrega de un usuario, necesita un parámetro llamado *files_filemanager*, que se trata del identificador del *draft area* donde se ubica dicho fichero de feedback. En un servidor de Moodle, para cada usuario existe un área de archivos privados del usuario y un área de borradores del usuario, llamado *draft area*. Previo a realizar una llamada a la función *subirNota*, se debe subir el archivo de feedback a un *draft area,* y una vez subido, obtener su identificador. Este identificador es el que se recibe como parámetro en la función *subirNota* y es el requerido por el *Web Service* para poder cargar el archivo de retroalimentación a la entrega del usuario.

Por tanto, la función *save_grade,* recibe como parámetros la calificación, el feedback en texto y el identificador de la ubicación del archivo de feedback, y se encarga de subir la calificación y la retroalimentación correspondiente a la entrega del usuario que se busca.

```
existenModificaciones(conexión, tarea, id usuario):
    obtener entrega del usuario sobre la tarea
    si el usuario no ha realizado entrega en dicha tarea:
        retorno Falso
    si la entrega no ha sido aun calificada:
        retorno True
    sino:
        obtener fecha de última modificación
        obtener fecha de última calificación
        si fecha modificación > fecha calificación:
            retorno True
        sino:
            retorno False
```

**Pseudocódigo 3*. Función "existenModificaciones".***

El *Pseudocódigo 3* corresponde con la función *existenModificaciones*, la cual se encarga de comprobar si la entrega de un alumno sobre una tarea necesita ser calificada, para lo cual comprueba si ha habido modificaciones.

Primero, para obtener la entrega del usuario sobre la tarea, se realiza una llamada al *Web Service* llamado '*mod_assign_get_submission_status'*, al cual se le pasa como parámetro el identificador del usuario y el identificador de la tarea y retorna toda la información correspondiente a la entrega de dicho usuario en la tarea. Con esta información, uno de los datos que se reciben sobre la entrega es el *status* de esta, el cual puede ser *'new'*(el alumno no ha realizado entrega), '*not graded'*(el alumno ha realizado entrega pero no ha sido calificada) o *'graded*'(el alumno ha realizado entrega y ha sido calificada). En caso de ser *'new'*, se retorna False, ya que no se ha realizado ninguna entrega y no se puede calificar, en caso de ser *'not graded'*, se retorna True, ya que la entrega aún no ha sido calificada en ningún momento, y en caso de ser *'graded'*, se debe comprobar si la calificación está correctamente actualizada, ya que puede que después de haberse calificado, el alumno haya realizado modificaciones sobre la misma y necesite recalificarse .Por tanto, se obtiene la fecha de la última modificación y la fecha de la última calificación a partir de los datos obtenidos sobre la entrega. Se comparan ambas fechas, y en caso de que la fecha de la última modificación sea posterior a la fecha de la última calificación, indicará que la entrega debe ser recalificada, por lo que se retornará un valor True. De lo contrario, se retornará un valor False.

```
descarga(conn, course, assignment):
    crear hoja de calculo
    establecer campos de cabecera
    recorrer todos los alumnos del curso:
        obtener toda la información de la entrega del alumno sobre la tarea dada
        obtener toda la información sobre el alumno
        crear una nueva fila en la hoja para el alumno
        completar la fila con la información obtenida
```

**Pseudocódigo 4.** ***Función "descarga".***

El *Pseudocódigo 4* corresponde con la función *descarga* encargada de descargar la hoja de calificaciones de una tarea concreta.

Esta función recibe como parámetros el objeto de la conexión con el servidor de Moodle, la tarea y el curso al que pertenece dicha tarea. En primera instancia, se crea una hoja de cálculo vacía con el nombre de '*Calificaciones_nombredelatarea.csv'*, ya que todas las hojas de calificaciones que se utilizan en este proyecto deberán seguir estas pautas en sus nombres. Se establecen los campos de cabecera de la hoja, los cuales serán idénticos a los proporcionados por Moodle en sus hojas de calificaciones: *Identificador, Nombre completo, Dirección de correo, Estado, Calificación, Calificación máxima, La calificación puede ser cambiada, Última modificación (entrega), Última modificación (calificación), Comentarios de retroalimentación.*

Una vez establecidos, se recorren todos los alumnos del curso, y para cada uno de estos se completará una fila de la hoja de calificaciones con los datos respectivos a sus entregas sobre la tarea dada. Para la obtención de estos datos, se hace uso del Web Service llamado *mod_assign_get_submission_status* mencionado en el apartado de Web Services. Este servicio recibe como parámetros un usuario y una tarea y devuelve una lista con todos los datos relacionados con la entrega del usuario sobre dicha tarea. Además, es necesario obtener toda la información sobre el alumno, para lo que se hace uso de la función *from_userid* de la clase *MoodleUser.* Se crea una nueva fila en la hoja de calificaciones la cual contendrá toda la información correspondiente con la entrega del alumno. Cada casilla de la nueva fila será rellenada con los datos obtenidos sobre la entrega y sobre el alumno. Este proceso se realizará para todos los alumnos contenidos en el curso, aunque no hayan realizado ninguna entrega sobre la tarea. En este último caso sus campos permanecerán vacíos en la hoja de calificaciones. Al finalizar la ejecución de la función, se habrá creado una hoja de calificaciones respecto a la tarea correspondiente con los datos de todos los alumnos pertenecientes al curso.

## 5.2 Aplicaciones finales

Se desarrollan dos aplicaciones finales encargadas de satisfacer los dos casos de uso explicados en la fase de *Análisis de Requisitos*. Como ya se comentó en el apartado de *Diseño*, para cada caso de uso se crea un script encargado de realizar el proceso total de autoevaluar prácticas de alumnos en sus respectivas situaciones, uno para los casos en los que solamente se requiera calificar una vez las entregas de los alumnos y otro para los casos que soporten modificaciones en las entregas de los alumnos y estas deban de ir recalificándose a medida que se sufren dichas modificaciones.

El pseudocódigo del script ***'EvaluacionUnica.py'*** se muestra en el Pseudocódigo 5.

```
EvaluacionUnica(id curso, nombre tarea):
    establecer conexión con el servidor Moodle
    obtener curso
    obtener tarea
    descargar todas las entregas
    descargar hoja de calificaciones de la tarea
    recorrer todas las entregas:
        ejecutar herramienta de autoevaluación sobre la entrega
        almacenar la calificación de la entrega
    anotar las calificaciones en la hoja de calificaciones
    subir las calificaciones y los archivos de retroalimentación al servidor Moodle
```

**Pseudocódigo 5. *Script "EvaluacionUnica.py"***

En primer lugar se establece la conexión con el servidor de Moodle, para lo que se requiere tanto la IP del servidor de Moodle como del token del usuario que quiere establecer la conexión. En este caso, estos datos se obtienen del fichero *ConfiguracionUsuario.txt* explicado previamente. Tras establecer la conexión, a partir de los argumentos de entrada, se obtienen el curso y la tarea que se desea evaluar. Se comprueba si la tarea proporcionada existe en dicho curso. Una vez comprobado, con la función *descargaTodas* del fichero 'entregas.py', se descargan todas las entregas de la tarea dada, y se almacenan siguiendo la jerarquía de directorios ya

mencionada. Posteriormente se descarga la hoja de calificaciones correspondiente a dicha tarea, la cual contendrá los campos vacíos, ya que aún ninguna entrega ha sido calificada.

Tras las descargas, se recorren todas las entregas realizadas por los alumnos, y se recupera el fichero a evaluar para cada alumno con entrega disponible. Con la ruta del fichero que se desea evaluar, se ejecuta la herramienta de autoevaluación, que retornará calificación en un valor numérico del 0 al 100. A su vez, dentro del directorio donde se encuentra el fichero de entrega, se crea un directorio *output*, en el cual se generará un fichero de retroalimentación con el nombre de *analysis.html.* Tras haber ejecutado la herramienta de autoevaluación sobre todos los ficheros entregados por los alumnos y haber almacenado todas las calificaciones, se hará una llamada a la función *apuntarNotas*, la cual recibe como parámetro dichas calificaciones de todos los alumnos y se encargará de anotarlas en la hoja. Finalmente y con las calificaciones anotadas, se pasa a realizar una llamada a la función *subir* del fichero 'hojaCalificaciones.py', que se encargará de subir al servidor de Moodle tanto los ficheros de retroalimentación como las notas que aparecen en la hoja de calificaciones.

El pseudocódigo del script ***'EvaluacionInteractiva.py'*** se muestra en el Pseudocódigo 6.

```
EvaluacionInteractiva(id curso, nombre tarea):
    establecer conexión con el servidor Moodle
    obtener curso
    obtener tarea
    bucle infinito:
        recorrer todas las entregas:
            comprobar si la entrega necesita ser calificada
            si necesita ser calificada:
                descargar la entrega
                ejecutar herramienta de autoevaluación sobre la entrega
                cargar fichero de retroalimentación
                subir calificación y fichero de retroalimentación a Moodle
        esperar 30 segundos
```

**Pseudocódigo 6.** ***Script "EvaluacionInteractiva.py".***

El comienzo es idéntico al del script '*PruebaGeneral.py'*, donde se establece la conexión con el servidor de Moodle y se obtienen el curso y la tarea correspondientes. Tras comprobar si la tarea existe en el curso proporcionado, durante un ciclo infinito de tiempo, se recorren todas las entregas de la tarea proporcionada con la función *submissions* de la clase *MoodleAssignment*. Para cada entrega, se realizará una llamada al método *existenModificaciones* para comprobar si la entrega necesita ser calificada o recalificada. En caso afirmativo, se descargará esa única entrega mediante una llamada al método *descargaPorUsuario*, y tras descargarse se ejecutará la herramienta de autoevaluación para el fichero entregado de donde se obtendrá la calificación de la entrega del alumno. Una vez evaluada la entrega y se haya generado un fichero de retroalimentación, como ya se explicó, se cargará dicho fichero de retroalimentación a un *draft area* de Moodle y se obtendrá el identificador de la ubicación donde se ha cargado el fichero de retroalimentación, ya que posteriormente servirá para subirlo a la entrega del alumno.

Con el fichero de retroalimentación ya cargado y la calificación de la entrega obtenida, se realiza una llamada al método *subirNota*, encargado de subir la calificación de un único alumno junto a su fichero de retroalimentación correspondiente.

En caso de que no se haya producido ninguna modificación en la entrega, simplemente se pasa a comprobar la siguiente entrega. Al finalizar este bucle encargado de recorrer todas las entregas realizadas, mediante el método *time.sleep*, se duerme la ejecución durante un periodo de 30 segundos. Al concurrir dicho periodo de tiempo, se vuelve de nuevo a comprobar si ha habido modificaciones en las entregas, y así sucesivamente hasta que el profesor decida finalizar la clase o finalizar el tiempo estipulado para la realización de la tarea.

El script tan sólo se dejará de ejecutarse si mediante la línea de comandos de la consola se aborta su ejecución.

## 5.3 Extensión de la librería Moodleteacher

Por sencillez para los profesores o usuarios que vayan a hacer uso de las funcionalidades implementadas en este trabajo, ha sido necesario extender la librería *moodleteacher* con algunas funcionalidades adicionales que han simplificado el desarrollo de las aplicaciones de alto nivel.

En la librería *moodleteacher,* dentro de la clase '*MoodleAssignment'* contenida en el fichero '*Assignments.py',* tan sólo existe una forma de obtener una instancia de un objeto de dicha clase a partir de su identificador. Debido a que un mismo curso puede llegar a tener infinidad de tareas cada una con un identificador numérico diferente, es muy tedioso y complicado que un profesor tenga que memorizar o tener apuntado los identificadores de cada tarea de sus cursos, ya que un mismo profesor puede llegar a tener varios cursos. Por lo tanto, el profesor que use alguna de las aplicaciones implementadas, indicará el nombre completo de la tarea a evaluar ya que será más fácil para un profesor conocer el nombre de una tarea que su identificador. Con este dato, para poder obtener una instancia de la clase *MoodleAssignment* correspondiente con la tarea que se busca, se necesita una función capaz de obtenerlo, por lo tanto, se ha añadido un nuevo método a la clase *MoodleAssignment* llamado '*from_assignment_name'* el cual devolverá una tarea a partir de su nombre pasado como argumento de entrada. Esta función es similar a la función llamada *'from_assignment_id'* ya predefinida en la propia clase, ya que reciben como argumentos de entrada el objeto del curso que contiene la tarea y el nombre de la misma tarea. Con estos datos, se recorre la lista de *MoodleAssignment* contenidos en el curso hasta encontrar el *MoodleAssignment* cuyo nombre coincida con el recibido por parámetro.

Otra de las extensiones implementadas en la librería *moodleteacher* está relacionada con la clase *MoodleCourse*. Esta clase contiene una función para obtener un objeto de un curso a partir de su identificador, pero esta clase viene incompleta y por finalizar, por lo que se debe completar para poder ejecutarla, ya que originalmente esta función simplemente devuelve un objeto vacío de la clase *MoodleCourse*. En primer lugar, se realiza una llamada al servicio web llamado *'core_course_get_courses_by_field',* en el que se pasa como argumento el identificador del curso, y se obtienen todos los datos del curso como retorno. Estos datos, es decir, el nombre completo, el nombre

corto y el identificador, se almacenan como parámetros del propio objeto de la clase *MoodleCourse* y se retornan.

Estas extensiones o modificaciones a la librería *moodleteacher* se han añadido al repositorio creado para este trabajo para que puedan ser añadidas directamente a la librería original sin necesidad de modificarla. De esta forma, la librería *moodleteacher* junto a las nuevas modificaciones, compondrán una nueva librería capaz de servir de soporte para la realización de las diferentes funcionalidades implementadas en este trabajo. En el mismo repositorio aparecen instrucciones de cómo añadir las modificaciones a la propia librería.

# 6. Pruebas

## 6.1 Configuración del Servidor de Pruebas

Para la realización de todas las pruebas correspondientes con cada funcionalidad implementada durante el trabajo, se requiere de un servidor de Moodle de pruebas con el que interaccionar y crear cursos, alumnos y tareas para poder probar las distintas funciones.

En cuanto al sistema operativo utilizado para la realización de este trabajo, se ha utilizado una máquina virtual de *VirtualBox* conteniendo la versión de Ubuntu 20.04 de 64 bits. Una vez configurada e instalada la máquina virtual, el primer paso a realizar era el de configurar este nuevo servidor de Moodle donde se va a trabajar, cuya configuración se explica a continuación.

Antes que todo, a la hora de manipular con ficheros de la propia configuración del servidor de Moodle o incluso de la librería de *moodleteacher* se necesita de una herramienta para ello, por lo que en este caso se ha utilizado el editor *vim*, instalado con el comando:
*sudo apt-get install vim*

Prosiguiendo, se requiere la instalación del servidor y cliente de MySQL para la gestión de la base de datos del servidor de Moodle, así como la instalación de apache y de php. Para ello, simplemente se ejecuta la siguiente línea de comandos:
*sudo apt install apache2 mysql-client mysql-server php7.4 libapache2-mod-php*

Para establecer la contraseña del usuario *root* en el servidor de mysql, se ejecuta el siguiente comando, ya que dicha contraseña hará falta más adelante para acceder al servidor:
*sudo mysql_secure_installation*

A parte de estos módulos principales, el servidor de Moodle necesita la instalación de otros paquetes adicionales para los cuales se correrá el siguiente comando:
*sudo apt install graphviz aspell ghostscript clamav php7.4-pspell php7.4-curl php7.4-gd php7.4-intl php7.4-mysql php7.4-xml php7.4-xmlrpc php7.4-ldap php7.4-zip php7.4-soap php7.4-mbstring*

Una vez instalados estos módulos, se reinicia el apache de manera que estos sean cargados de forma correcta. Para finalizar los pasos previos a la instalación del Moodle, se requiere *Git* para instalar y actualizar la aplicación principal de Moodle, por lo que se usará el siguiente comando:
*sudo apt install git*

Una vez está todo listo, el siguiente paso será descargar Moodle. Primero, se establece el repositorio local, para lo que se utilizará el directorio /opt, y dentro del mismo, en el terminal ejecutamos el siguiente comando para descargar mediante *Git* el código de Moodle y su índice:
*sudo git clone git://git.moodle.org/Moodle.git*.

Una vez descargado, debemos copiar el repositorio local a nuestro *webroot,* es decir, al directorio /var/www/html/, para lo que ejecutaremos las siguientes líneas de comando:
*sudo cp -R /opt/moodle /var/www/html/* ***Copiamos el contenido al webroot***
*sudo mkdir /var/moodledata* ***Creamos el directorio moodledata***
*sudo chown -R www-data /var/moodledata* ***Cambiamos el propietario de moodledata***
*sudo chmod -R 777 /var/moodledata* ***Otorgamos permisos a todos los usuarios***
*sudo chmod -R 0755 /var/www/html/moodle* ***Damos permisos de ejecución solo al propietario del directorio moodle***

El siguiente paso será configurar el servidor MySQL para nuestra base de datos que dará soporte al servidor de Moodle. Primero que todo, cambiamos el almacenamiento predeterminado a *innodb* y el formato de archivo predeterminado a *Barracuda*. Estos cambios los realizaremos en el archivo *mysqld.cnf* contenido en el directorio de instalación del MySQL server en el directorio */etc/mysql/mysql.conf.d/.* Escribimos en una línea de dicho archivo las siguientes líneas:
*default_storage_engine = innodb*
*innodb_file_per_table = 1*
*innodb_file_format = Barracuda*

En este caso, establecer el formato de ficheros a Barracuda generó un error, en el que desconocía la variable establecida, por lo que, al usar la versión MySQL 8.0,

podemos prescindir de estos cambios, así que simplemente eliminamos esa línea del archivo *mysqld.cnf.*

Una vez realizados los cambios anteriormente mencionados, se reinicia el servidor de mysql para que se hagan efecto y se accede al mismo usando el usuario root y la contraseña previamente almacenada, y se crea la base de datos con el nombre de *moodle* para el servidor de Moodle, ejecutando la siguiente línea de comandos por la terminal de mysql:
*CREATE DATABASE moodle DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;*

Con la base de datos establecida, se crea lo que será el usuario principal y administrador del Moodle, que en este caso de ejemplo se han usado el usuario *'moodleadmin'* y la contraseña *'Moodleadminpass3!'.* Para crear dicho usuario y otorgarles todos los permisos, se ejecutan los siguientes comandos:
*CREATE USER 'moodleadmin'@'localhost' IDENTIFIED BY 'Moodleadminpass3!';*

*GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,CREATE TEMPORARY TABLES,DROP,INDEX,ALTER ON moodle.* TO 'moodleadmin'@'localhost';*

Con todo ya preparado, se pasa a completar la instalación del servidor de Moodle, para lo que se accede a nuestro navegador en la siguiente dirección: *http://127.0.0.1/moodle* , donde la dirección IP corresponde con el localhost. Una vez accedido, se mostrará una serie de formularios interactivos donde se rellenará con la información que se requiere. En dicha información a proporcionar, se indicará que el tipo de base de datos a usar es *mysqli* y se establecerá que el servidor host sea *localhost*, la base de datos sea *moodle,* el usuario sea *moodleadmin* y la contraseña previamente mencionada en la creación del usuario. A parte de esta información, es necesario rellenar más información extra como puede ser el nombre del servidor, que en este caso será *moodlejesus* y finalmente se confirma la instalación y se podrá comenzar a usar el nuevo servidor de Moodle.

Con el servidor ya listo, para poder comenzar a trabajar con la librería *moodleteacher* y con los *Web Services* que proporciona la plataforma de Moodle, es necesario activar estos últimos. Para ello, se accede al servidor de Moodle mediante el usuario y la contraseña de administrador previamente creados durante la instalación, y en la pestaña de opciones avanzadas dentro de la administración del sitio, se habilitan los

*Web Services*. Además, dentro del apartado *Servicios Web* en la pestaña de *Extensiones*, se accederá a la opción *Administrar protocolos* y se deberán habilitar aquellos protocolos que vayan a ser necesarios para los servicios web que se vayan a dar uso durante la fase de programación de las diferentes funcionalidades, por lo que en este caso, se habilitarán los protocolos *REST, SOAP* y *XML-RPC,* y también, se tendrá la oportunidad de activar la autogeneración de la documentación de los servicios web, la cual se activará para poder ver al detalle sus diferentes servicios y funciones y poder ponerlas en uso.

Una vez activados los servicios web y los protocolos necesarios, se creará un usuario que será destinado a utilizar los servicios web desde el rol del profesor de un curso de ejemplo. Después, se necesita crear un servicio externo. Un servicio externo se trata de un conjunto de funciones de los *Web Services*, las cuales podrán ser utilizadas por todos los usuarios o solo por usuarios autorizados. Para ello, se crea un servicio externo que en un principio va a contener aquellas funciones de los *Web Services* que se utilicen desde la librería *moodleteacher* de la cual se hará uso para este trabajo. Se añaden dichas funciones al nuevo servicio externo creado, con el nombre de *ServicioWebTFG* y se añade a la lista de usuarios autorizados a utilizar dicho servicio al usuario previamente creado como profesor de un curso de ejemplo. Este servicio externo se irá viendo ampliado o reducido durante el transcurso de la fase de implementación de las funcionalidades, ya que habrá algunos servicios web que no serán utilizados y estén incluidos, y viceversa. A su vez, al servicio externo hay que otorgarle una serie de permisos para poder utilizar los protocolos previamente habilitados, donde en la pestaña de los *Web Services* se selecciona el servicio *ServicioWebTFG* y añadimos los siguientes permisos: *webservice/rest:use, webservice/soap:use, webservice/xmlrpc:use.* A su vez, se añadirá el permiso *moodle/webservice:createtoken* para permitir generar un token de seguridad que se explicará a continuación.

Finalmente, una vez establecido el servicio externo, se creará un token para el usuario que hará uso de los servicios web del Moodle de manera segura desde fuera del propio sistema. En el caso del servidor Moodle recientemente creado, se utilizará el usuario creado con el rol de profesor, pero que en cualquier caso será aquel profesor que vaya a hacer uso de las funciones que se irán a implementar y que necesitarán los servicios web. En todo caso, se accede a la pestaña de *Servicios Web* y se

pinchará sobre la opción *Administrar fichas (tokens)*. Dentro, se selecciona el usuario y el servicio *ServicioWebTFG* previamente creados y se guardan los cambios. Se generará un token para dicho usuario, que será utilizada más adelante durante la fase de programación. Con el servidor de Moodle configurado y los servicios web activados, se podrán realizar las manipulaciones necesarias con el objetivo de probar todas las funcionalidades implementadas.

## 6.2 Pruebas Unitarias

Como se acaba de explicar, para este trabajo se ha creado un servidor de Moodle para realizar pruebas sobre las diferentes funcionalidades que se han ido implementando y para comprobar el correcto funcionamiento de los diferentes *Web Services*. El servidor en concreto recibe el nombre de *moodlejesus*, y para realizar las pruebas se ha creado un curso, llamado *Programación Web*. A su vez, han sido necesarios varios usuarios para simular un curso real, por lo que se ha creado un usuario el cual ha adquirido el rol de profesor del curso, y cinco usuarios que han adquirido el rol de alumnos de dicho curso. Dentro del propio curso, a lo largo del transcurso de las diferentes pruebas se han ido creando diversas tareas con características variadas y así poder comprobar situaciones distintas.

En este apartado se van a detallar las diferentes pruebas que se han realizado para comprobar el funcionamiento de las funciones individuales implementadas más complejas:

- ***descargaTodas****:* Para probar el correcto funcionamiento de dicha función se creó una tarea que admitiera un número indefinido de archivos en la entrega, ya sean comprimidos o ficheros únicos. Iniciando sesión con tres de los usuarios previamente creados, uno de ellos subió varios ficheros de texto, otro de ellos subió un archivo comprimido 'zip' y el último subió un archivo comprimido 'tar'. Con esto se quiere comprobar que la función es capaz de descargar y desempaquetar los archivos subidos en un directorio concreto para cada usuario (siguiendo la jerarquía de directorios explicada en la implementación). Se crea un script de prueba que implementa la función *descargaTodas,* y pide por consola el identificador del curso en el que se halla la tarea y el nombre de la propia tarea. Se ejecuta dicho script proporcionando el identificador del curso que fue creado, que en este caso es '2', y el nombre de la tarea que se ha creado, en este caso es '*Nuevaprueba*'. La figura 5 muestra el resultado por consola tras ejecutar el script.

```
jesuscampo@jesustfg:~/Documentos/PruebaDos$ ./pruebaDescarga.py -c 2 -a Nuevaprueba
Hay un total de 3 entregas realizadas de un total de 5 alumnos matriculados

Los ficheros han sido descargados correctamente
```

**Figura 5.** ***Consola tras probar la función "descargaTodas".***

La jerarquía de directorios y los ficheros que estos contienen son los mostrados en la Figura 6.

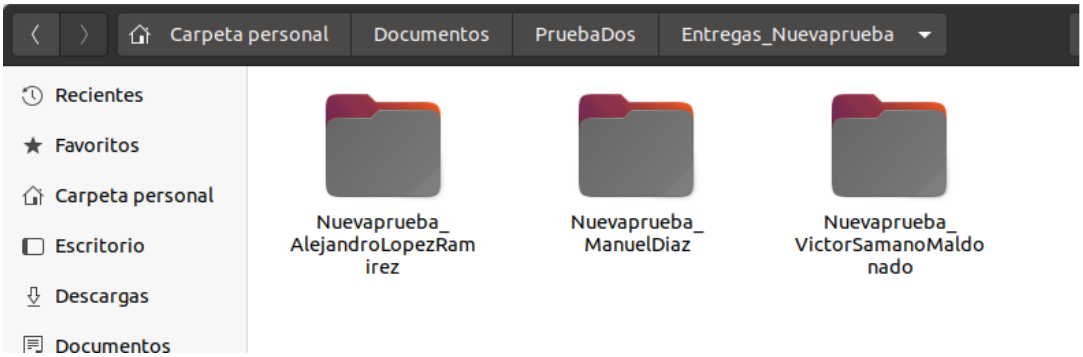


**Figura 6.** ***Jerarquía de directorios tras ejecutar la función "descargaTodas".***

Como se puede observar, se crea un directorio para cada usuario, donde dentro de cada uno de ellos se almacenan los archivos que han añadido a sus respectivas entregas, como se muestra en la Figura 7.

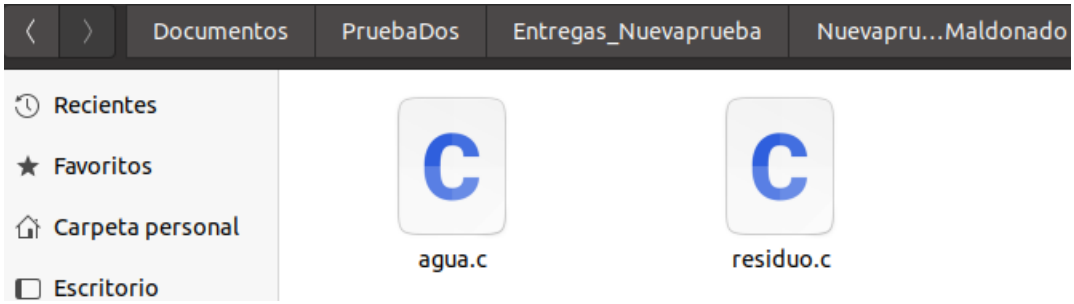


**Figura 7.** ***Contenido del directorio creado para el alumno Víctor Sámano Maldonado tras la ejecución de la función "descargaTodas".***

-***existenModificaciones:*** Para comprobar su funcionamiento se han probado los cuatro posibles casos que puede haber en una entrega, en función de si la entrega está vacía, calificada, o ha sido modificada tras la última calificación. A continuación, en la Tabla 1, se presenta una tabla con las características de las cuatro entregas que se han probado y los valores que retornan, que en el caso de necesitar calificarse de nuevo se devuelve un valor True y de lo contrario un valor False.

| ENTREGAS | Vacía | Calificada | Modificada desde la última calificación | Necesita calificarse |
|---|---|---|---|---|
| Entrega 1 | SÍ | - | - | NO |
| Entrega 2 | NO | NO | - | SÍ |
| Entrega 3 | NO | SÍ | NO | NO |
| Entrega 4 | NO | SÍ | SÍ | SÍ |

**Tabla 1.** ***Características de las pruebas sobre la función "existenModificaciones".***

***-descarga:*** Esta función se encarga de descargar la hoja de calificaciones de una tarea concreta y para realizar las pruebas convenientes, se utilizó la misma tarea ('*Nuevaprueba*') que en la prueba de la función *descargaTodas.* Como ya se habían subido entregas con tres alumnos distintos a dicha tarea, se inició sesión con el usuario profesor del curso, y se calificaron las tres entregas con una calificación diferente. Una vez evaluadas, se ejecutó un script de prueba encargado de realizar una llamada a la función *descarga*, el cual se encarga de crear una hoja de calificaciones con los datos de todos los alumnos matriculados, hayan realizado una entrega sobre dicha tarea o no. Tras la ejecución, se crea la hoja de calificaciones correspondiente con la entrega, conteniendo los mismos campos de cabecera que la hoja de calificaciones proporcionada por la plataforma Moodle.

La hoja de calificaciones queda distribuida tal y como aparece en la Figura 8, para el ejemplo dado en el que se han calificado las entregas de tres alumnos de cinco matriculados:

| | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| 1 | Identificador | Nombre completo | Dirección de correo | Estado | Calificación | Calificación máxima | La calificación puede ser cambiada |
| 2 | 4 | Alejandro Lopez Ramirez | alejandrolopez@gmail.com | Enviado para calificar - Calificado | 100 | 100 | Sí |
| 3 | 6 | Carlos Fernandez Acebo | carlosfernandez@gmail.com | Sin entrega | | 100 | Sí |
| 4 | 7 | Manuel Diaz | manueldiaz@gmail.com | Enviado para calificar - Calificado | 55 | 100 | Sí |
| 5 | 8 | Angel Sanchez Ramiro | angelsanchez@gmail.com | Sin entrega | | 100 | Sí |
| 6 | 9 | Victor Samano Maldonad | victorsamano@gmail.com | Enviado para calificar - Calificado | 100 | 100 | Sí |

| F | G | H | I | J |
|---|---|---|---|---|
| Calificación máxima | La calificación puede ser cambiada | Última modificación (entrega) | Última modificación (calificación) | Comentarios de retroalimentación |
| 100 | Sí | lunes, 1 de agosto de 2022, 13:26 | viernes, 5 de agosto de 2022, 20:46 | |
| 100 | Sí | - | - | |
| 100 | Sí | lunes, 1 de agosto de 2022, 13:26 | viernes, 5 de agosto de 2022, 20:46 | |
| 100 | Sí | - | - | |
| 100 | Sí | lunes, 1 de agosto de 2022, 13:26 | viernes, 5 de agosto de 2022, 20:46 | |

**Figura 8.** ***Hoja de calificaciones obtenida tras la ejecución de la función "descarga".***

***-subir:*** Para realizar las pruebas correspondientes sobre la función *subir* del fichero *hojaCalificaciones.py*, se creó una tarea para la cual tres de los cinco alumnos del curso subieron un fichero de ejemplo como entrega a dicha tarea. Una vez subidas las entregas, se ejecutó el script de prueba de la función *descarga*, el cual descargó tanto las entregas de los alumnos como la hoja de calificaciones correspondiente con dicha tarea.

La labor de generar tanto la calificación de una entrega como el fichero de retroalimentación dentro de un directorio con el nombre de *Output* la realiza la herramienta de autoevaluación que se vaya a utilizar, pero en este caso, para probar solamente la función *subir*, no es necesario utilizar dicha herramienta, por lo que estos datos se añadirán manualmente. En la hoja de calificaciones recientemente descargada, se introducen valores aleatorios en el campo de calificación de los tres alumnos que habían realizado la entrega. Además, dentro del subdirectorio de cada alumno donde se almacenan sus entregas, se creó un directorio con el nombre de *Output* y dentro se añadió un fichero de retroalimentación de ejemplo con el nombre de *Analysis.html*.

Teniendo la hoja de calificaciones preparada y los ficheros de retroalimentación ubicados en los directorios correctos, se ejecuta un script de prueba encargado de llamar a la función *subir*, pasándole como argumento de entrada la conexión establecida con el servidor de Moodle y la tarea a la cual se quiere subir las calificaciones. Como ya se comentó en la implementación, a la hora de definir el nombre de las hojas de calificaciones, se ha seguido un patrón concreto, y los nombres adaptan la estructura de *'Calificaciones_nombredelatarea.csv'*, por lo que no es necesario pasar como argumento el nombre de dicha hoja.

Con el script ya ejecutado, se suben tanto las calificaciones como los ficheros de retroalimentación correctamente a las respectivas entregas de los alumnos sobre la tarea.

## 6.3 Pruebas de Integración

Para la realización de estas pruebas de integración de las aplicaciones finales las cuales reúnen todas las funcionalidades testeadas durante las pruebas unitarias, hace falta el uso de una herramienta de autoevaluación, y en este caso, se ha utilizado una herramienta proporcionada por el director de este TFG, Héctor Pérez, la cual se encarga de evaluar ficheros individuales en formato *.c* en un lenguaje de programación C. También se han proporcionado dos ficheros de ejemplo para probar la herramienta, ambos bajo el nombre de '*agua.c*', pero cada uno de ellos con un contenido distinto, por lo que tendrán calificaciones diferentes. Esta herramienta está amoldada a las condiciones mencionadas en la fase de *Diseño*, ya que consiste en un script el cual tras evaluar un fichero de *c*, retorna una calificación como número entero entre 0 y 100, y genera un fichero de retroalimentación bajo el nombre de *Analysis.html* dentro de un directorio llamado *Output* creado dentro del subdirectorio correspondiente al alumno del cual se evalúa su fichero. El script recibe el nombre de *intro_sw_static_check.sh*, y recibe dos argumentos de entrada. El primero es la ruta en la que se encuentra el fichero a evaluar, y el segundo es la ruta de un fichero llamado *config.yaml* la cual contiene la configuración global necesaria para la evaluación.

Las dos aplicaciones finales implementadas en este proyecto, cumplen cada una con un requisito diferente. Por lo tanto, para cada uno de los dos casos de uso que se han planteado, habrá que generar escenarios amoldados a dicho caso de uso dentro de nuestro servidor de Moodle de pruebas.

### 6.3.1 Prueba de la aplicación EvaluacionUnica.py

Esta aplicación corresponde con el caso de uso de autoevaluar prácticas de programación las cuales necesiten de una única calificación, ya que los alumnos no pueden modificar sus entregas y sus calificaciones en estas serán definitivas. Estos casos se pueden dar en ejercicios de exámenes finales o parciales, donde una vez realizas la entrega final, no hay vuelta atrás.

Por lo tanto, se ha simulado un examen final. Dentro del curso, se crea una tarea que recibe el nombre de *ExamenFinal*, entre cuyas características se detalla que las entregas no son modificables y solo admiten la entrega de un fichero. Se inicia sesión

con los cinco alumnos matriculados en el curso, y se entrega para cada uno de ellos uno de los ficheros de prueba '*agua.c*'.

Tras añadir las entregas de todos los alumnos correctamente, hay que añadir al script de prueba la llamada a la herramienta de autoevaluación, añadiendo la siguiente línea de código:

```
# Ejecutamos la herramienta correspondiente para calificar la entrega
r = subprocess.call(["./intro_sw_static_check.sh",totalpath,"config.yaml"])
```

Donde la variable *totalpath* contendrá la ruta del fichero que se quiere evaluar, y en la variable *r* se almacena la calificación correspondiente. En caso de que se quisiera utilizar otra herramienta, simplemente en el código habría que cambiar el nombre del script que se quisiera ejecutar, y sus argumentos pertinentes.

Una vez se tiene tantos las entregas correctamente añadidas a la tarea *ExamenFinal* y el script preparado, se procede a su ejecución. Se pasan como argumentos de entrada el identificador del curso de prueba, que en este caso es 2, ya que el identificador 1 corresponde con el curso principal de Moodle. Además, se pasa el nombre de la tarea a evaluar, que como ya se ha mencionado es *ExamenFinal.* Tras la ejecución del script el cual contiene el llamamiento a gran parte de las funciones implementadas, el resultado por consola queda reflejado en la Figura 9.

```
jesuscampo@jesustfg:~/Documentos/PruebaDos$ ./PruebaGeneral.py -c 2 -a ExamenFinal
Hay un total de 5 entregas realizadas de un total de 5 alumnos matriculados

Los ficheros han sido descargados correctamente
FICHERO CREADO
100
55
55
55
100



Se ha evaluado correctamente la tarea ExamenFinal a el usuario Alejandro Lopez Ramirez



Se ha evaluado correctamente la tarea ExamenFinal a el usuario Carlos Fernandez Acebo



Se ha evaluado correctamente la tarea ExamenFinal a el usuario Manuel Diaz



Se ha evaluado correctamente la tarea ExamenFinal a el usuario Angel Sanchez Ramiro



Se ha evaluado correctamente la tarea ExamenFinal a el usuario Victor Samano Maldonado
Calificaciones subidas satisfactoriamente
```

**Figura 9.** ***Resultado por consola de la aplicación "EvaluaciónUnica.py"***

### 6.3.2 Prueba de la aplicación EvaluacionInteractiva.py

Para probar el correcto funcionamiento de esta aplicación, hace falta establecer un escenario distinto al creado para la anterior prueba. Este caso de uso se da cuando se quieren evaluar tareas en las cuales los alumnos pueden ir modificando sus entregas y recibiendo nuevas calificaciones y retroalimentación a medida que se realizan estas modificaciones. Esto se puede dar para prácticas de aula, las cuales suelen durar en torno a las dos horas, en las que los alumnos en una primera instancia suban su ejercicio, vean sus errores y los aspectos que pueden mejorar, vuelvan a realizar de nuevo el ejercicio, y cuando lo vuelvan a subir puedan recibir una nueva calificación y así sucesivamente hasta conseguir la máxima puntuación.

La aplicación está ideada para que se comience a ejecutar al comienzo de una clase, y se finalice su ejecución al final de esta.

Para ello, se establece un escenario simulando una práctica de aula. Se crea una tarea llamada *PracticaAula*, la cual admite modificaciones en sus entregas y tan solo admite la entrega de un fichero. A diferencia de la prueba anterior, no añadimos entregas con ninguno de los alumnos matriculados, ya que al comienzo de una práctica de aula ningún alumno va a haber realizado una entrega. Por tanto, se comienza a ejecutar el script pasando como argumentos el identificador del curso, *2*, y el nombre de la tarea, *PracticaAula*. En un primer ciclo, la consola no mostrará nada por pantalla, ya que no se han realizado ninguna entrega. Cada 30 segundos, irá comprobando si se ha realizado una nueva entrega, y en caso afirmativo, la calificará y generará el fichero de retroalimentación correspondiente.

Ahora se inicia sesión con uno de los alumnos matriculados, Ángel Sánchez Ramiro y se añade un fichero de prueba *agua.c* a la tarea y se guardan los cambios. La figura 10 muestra el resultado por consola cuando se cumpla un ciclo de 30 segundos en la ejecución de la aplicación y se comprueben las entregas realizadas.

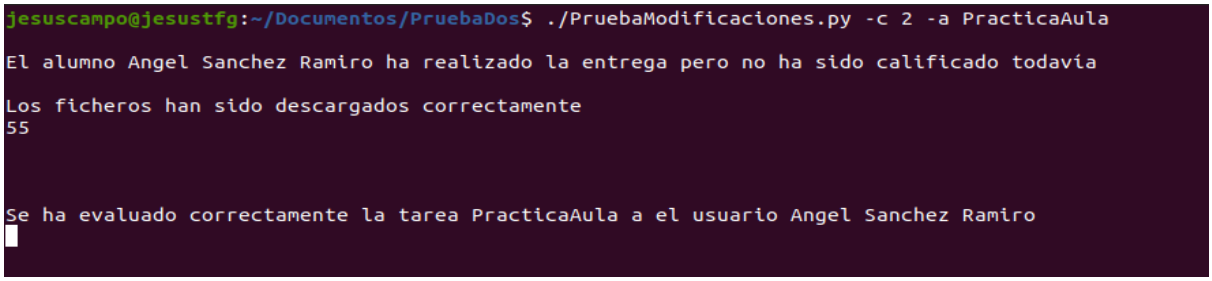


**Figura 10.** ***Resultado por consola de la aplicación "EvaluaciónInteractiva.py"***

En la figura 10 se puede observar que una vez que se detecta una nueva entrega, el script llama a la función *existenModificaciones* que comprueba si la entrega necesita calificarse o no, y en caso afirmativo, muestra por consola que el alumno correspondiente ha realizado una entrega que aún no ha sido calificada y procede a evaluarla y subir la calificación junto al fichero de retroalimentación al servidor de Moodle.

En los posteriores ciclos de la aplicación, esta va a ir comprobando si la entrega del alumno Ángel ha sufrido modificaciones y necesita ser recalificada. A medida que se van añadiendo futuras entregas por los alumnos, la aplicación va calificándolas y comprobando si sufren modificaciones cada 30 segundos.

Se realiza una modificación a la entrega de Ángel.

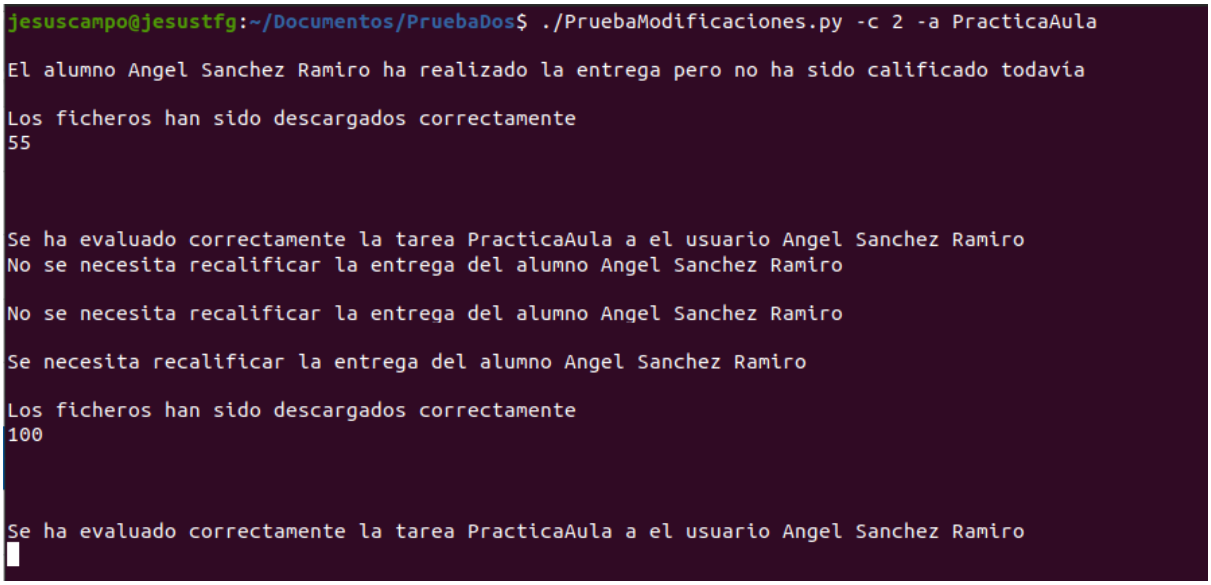


**Figura 11.** ***Resultado por consola de la aplicación "EvaluaciónUnica.py"***

En la figura 11 se observa que ha habido varios ciclos en los que la entrega de Ángel no ha necesitado recalificación, pero en cuanto se ha realizado una modificación a la entrega, la consola muestra que debe ser recalificada, y acto seguido vuelve a evaluarla y subirla a Moodle.

Por tanto, se comprueba el correcto funcionamiento de la aplicación.

# 7. Conclusiones y líneas futuras

En términos generales, este Trabajo de Fin de Grado ha consistido en la realización de una herramienta de autoevaluación de prácticas de programación para los dos casos de uso distintos que se le pueden presentar a un profesor. Con estas herramientas se automatiza la tarea de evaluar prácticas y la convierte en una actividad menos costosa para los docentes.

Las herramientas, la librería de utilidades y las extensiones realizadas sobre la librería de *moodleteacher*, se encuentran en el repositorio X junto a la documentación necesaria para su uso.

Desde un comienzo se ha partido de la base de una librería ya creada, *moodleteacher*, la cual mediante el uso de los WebServices nos abre una gran cantidad de oportunidades de cara a automatizar todo tipo de consultas y procesos posibles dentro de la plataforma Moodle. Se ha implementado una librería de utilidades con un conjunto de funciones de alto nivel con el uso de las clases y funciones contenidas en la librería *moodleteacher*. Sobre esta librería de utilidades, se han desarrollado las dos aplicaciones finales, una para tareas con evaluación única y otra para tareas con evaluación interactiva.

En este trabajo nos hemos centrado en la parte de autoevaluar prácticas de programación, pero durante la fase de diseño y de implementación me he dado cuenta de que a través de estas herramientas, fundamentalmente de los WebServices, existe un campo amplio por desarrollar para facilitar el uso de la plataforma Moodle tanto a los profesores como a los alumnos.

Durante el grado de Ingeniería Informática en la UC, en la mención de Software no se hace prácticamente nada de hincapié en el lenguaje de programación de Python, por lo que realizar una herramienta en este lenguaje ha servido de gran ayuda para aprenderlo. Esta fase de aprendizaje ha sido constante durante la totalidad del proyecto, desde que comencé a comprender la librería de Python de la cual partíamos hasta finalizar la implementación de la herramienta final.

Por otro lado, otro de los conocimientos adquiridos corresponde con el uso de los WebServices de Moodle, los cuales permiten la conexión e interacción con Moodle. En un primer momento fue complicado de entenderlos, pero tras estudiar la librería

de *moodleteacher* y la documentación de la API de los WebServices, conseguí comprender poco a poco su funcionamiento. Hasta tal punto, que durante alguna fase de la implementación se han hecho un uso directo de los WebServices sin necesidad de utilizar la librería *moodleteacher*.

Al tener que configurar y desarrollar un nuevo servidor de Moodle, la elaboración de este proyecto me ha permitido adquirir conocimientos acerca del funcionamiento interno de la plataforma desde el punto de vista de un administrador y de un profesor, y comprender todas las oportunidades que la propia plataforma brinda.

Este proyecto presenta una serie de mejoras y de posibles futuras ampliaciones que hagan de nuestra herramienta una más práctica, comprensible y con más funcionalidades. Por lo tanto, se presentan diferentes mejoras y desarrollos para un futuro:

- **Adaptar las aplicaciones finales para el uso de cualquier herramienta de autoevaluación.** Como se ha explicado durante la memoria, en este proyecto se han implementado dos aplicaciones las cuales autoevalúan prácticas de programación de Moodle. Estas aplicaciones se han enfocado en herramientas de programación las cuales evalúan un solo fichero y cumplen unos patrones concretos: generar un fichero de retroalimentación en un directorio determinado y generar una calificación entre 0 y 100. En líneas futuras, se pretende extender las aplicaciones finales para poder utilizar cualquier tipo de herramienta de autoevaluación que contenga sus propios patrones de generación de calificación y de retroalimentación y que sea capaz de evaluar más de un fichero.
- **Construir una interfaz gráfica para la configuración de usuario.** En nuestra herramienta, para proporcionar los datos necesarios para que el usuario pueda conectarse a su servidor de Moodle y pueda ejecutar nuestra aplicación, necesita añadir dichos datos a un fichero de configuración llamado '*ConfiguracionUsuario.txt*'. En un futuro se pretende implementar una interfaz gráfica que solicite estos datos y los almacene. De esta manera, la primera vez que un usuario ejecute la aplicación necesitará introducir los datos, pero las siguientes veces estos datos se cargarán directamente, a no ser que el usuario decida cambiarlos.

- **Realizar pruebas en el servidor de Moodle de la UC.** Tan sólo se han realizado pruebas en un servidor de prueba elaborado durante este proyecto, por lo que para poner en funcionamiento nuestra herramienta hace falta realizar pruebas en un servidor oficial de una universidad, en este caso de la Universidad de Cantabria. De resultar satisfactorias, se podría comenzar a expandir la herramienta a nuevos servidores y universidades.
- **Mantenibilidad de la herramienta.** Nuestro trabajo hace uso directo de la librería *moodleteacher* y de los WebServices, los cuales pueden sufrir modificaciones, ampliaciones o reducciones de funcionalidades. Por ello, se necesita estar pendiente de estas posibles modificaciones y poder adaptar nuestra herramienta a ellas de manera que siga funcionando de la misma forma.

# Referencias

[1] Moodle - Acerca de Moodle https://docs.moodle.org/all/es/Acerca_de_Moodle Revisado el 08/09/2022

[2] GitHub – Evalcode https://github.com/aldeam/evalcode/blob/master/README.md Revisado el 08/09/2022

[3] Martínez, E. (2020). *Nuevas Herramientas de apoyo para la evaluación automática de ejercicios de programación en Moodle a través de la extensión EvalCode*. Universidad de Cantabria, Santander, España. Obtenido el 08/09/2022 de https://repositorio.unican.es/xmlui/handle/10902/20546

[4] Quintana, G. (2019). *Mantenimiento y modularización de EvalCode para su puesta en marcha en plataformas Moodle*. Universidad de Cantabria, Santander, España. Obtenido el 08/09/2022 de https://repositorio.unican.es/xmlui/handle/10902/16895

[5] GitHub – Moodleteacher https://github.com/troeger/moodleteacher/blob/master/README.md

Revisado el 08/09/2022.