Capítulo 3

Lenguaje de programación

María de León Sigg
Juan Luis Villa Cisneros

https://doi.org/10.61728/AE24320030

Módulo de actividades / CodeRunner

Introducción

El sistema de gestión del aprendizaje Moodle facilita la creación de servicios educativos en línea y permite la incorporación de estrategias basadas en las tecnologías de la información para mejorar los procesos de enseñanza en áreas muy diversas, incluyendo aquellas que tienen como objetivo el desarrollo de habilidades de programación. Con el desarrollo de diferentes *plugins*, Moodle ha podido extender su funcionalidad, permitiendo que las actividades de enseñanza, aprendizaje y seguimiento a los estudiantes se diversifiquen y simplifiquen.

CodeRunner es un *plugin* que extiende las capacidades de Moodle para incluir preguntas de tipo adaptativo cuyas respuestas son instrucciones en un lenguaje de programación (Lobb y Harlow, 2016). Estas preguntas se añaden a través de un examen en la plataforma de Moodle y permite que los estudiantes escriban y verifiquen el código sin salir de él (Cardone et al., 2022), disminuyendo significativamente el esfuerzo del profesor para revisar y permitiendo a los estudiantes mejorar sus habilidades mediante la retroalimentación instantánea de su trabajo (Hatano, 2021).

En este capítulo se describe cómo se configura *CodeRunner* para añadir preguntas que esperan instrucciones en un lenguaje de programación. En la siguiente sección se resume cómo se instala y configura *CodeRunner,* se presentan ejemplos de su uso y se describe cómo se pueden extender las capacidades de las preguntas de *CodeRunner* con la adaptación de las plantillas de preguntas. Finalmente, se exponen algunas consideraciones a tener en cuenta cuando se usa el *plugin*.

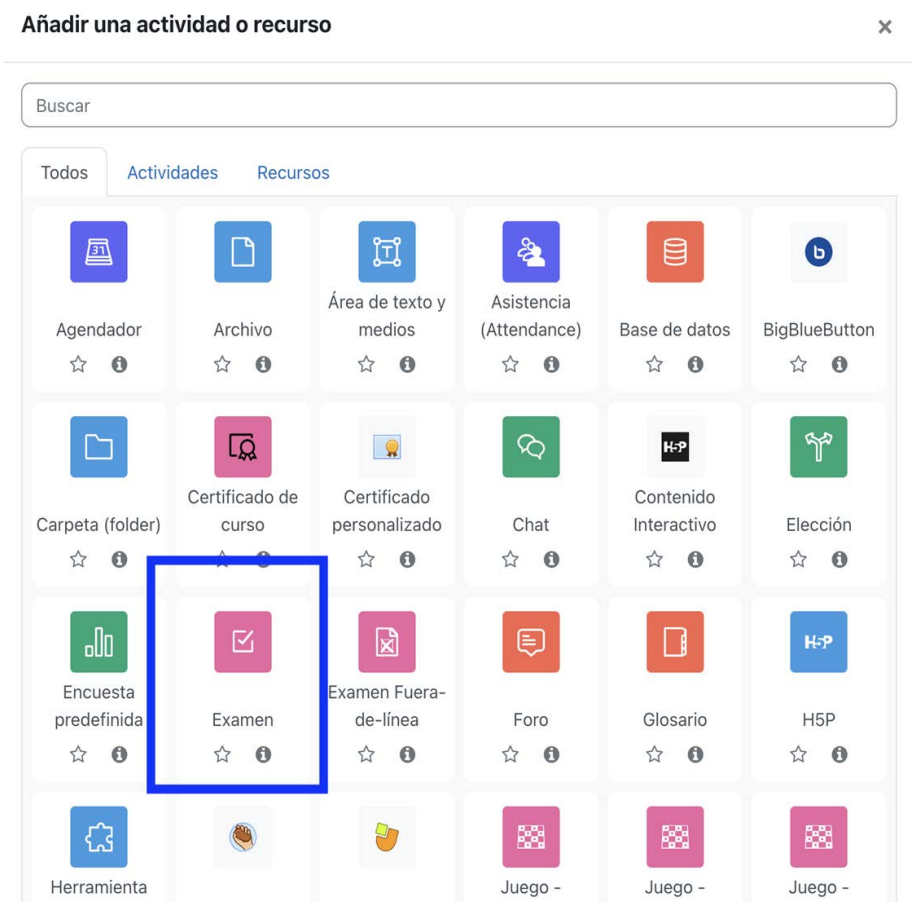
Instalación y uso de *CodeRunner*

Previo a su ejecución, *CodeRunner* debió ser instalado en Moodle. Para ello, se descargaron e instalaron dos *plugins*, uno para el tipo de preguntas y otro para configurar el modo adaptativo. Ambos se encuentran en el directorio de *plugins* de Moodle.

Una vez instalado, *CodeRunner* usa un servidor *Jobe* instalado en la Universidad de Canterbury para ejecutar el código que se va a revisar. Este es un servidor que da soporte a tareas pequeñas en diferentes idiomas y que fue desarrollado para usarse con *CodeRunner*. Sin embargo, se recomienda configurar un servidor propio con características similares, una vez que se utilice en el entorno de producción. La descripción de la instalación de este servidor propio se encuentra en https://github.com/trampgeek/jobe (Lobb y Hunt, 2023).

Cuando está instalado *CodeRunner,* se pueden añadir las preguntas de código a un examen, tal y como se puede ver en la figura 1. Si el servidor en el que se está ejecutando no ha sido cambiado, aparecerá un mensaje resaltado en amarillo indicando que se está utilizando el servidor de la Universidad de Canterbury.

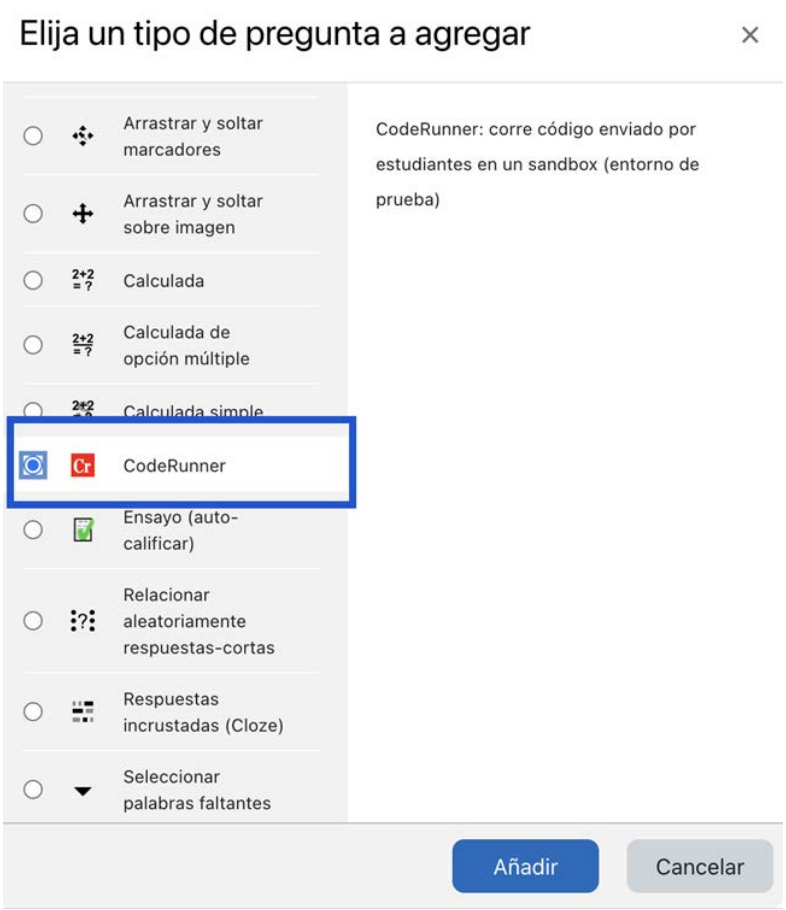
Figura 1
Crear un examen para usar CodeRunner



Dentro de las opciones para la configuración del examen en Moodle, se puede elegir el modo adaptativo. Sin embargo, es importante señalar que las preguntas *CodeRunner* siempre están en modo adaptativo, independientemente de la configuración del examen, ya que una de sus características es que los estudiantes pueden seleccionar la opción "Comprobar" para verificar si su código pasa las pruebas definidas en una pregunta, generalmente con una penalización, antes de enviarlo para su revisión (Lobb y Hunt, 2017).

Después de que se ha creado el examen en Moodle, el siguiente paso es añadir preguntas. Para ello, es necesario seleccionar las preguntas de tipo CodeRunner, como se puede observar en la figura 2.

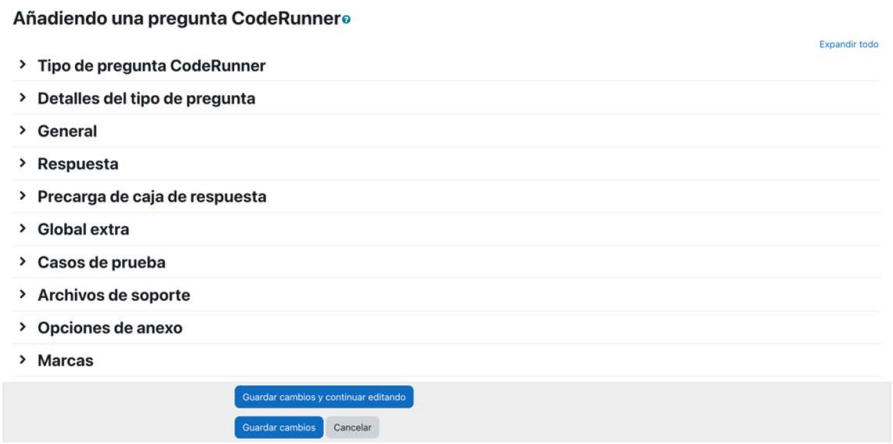
Figura 2
Insertar preguntas tipo CodeRunner



De manera general, las preguntas de *CodeRunner* se integran a un examen siguiendo este orden de pasos: primero se elige el lenguaje y tipo de preguntas que ya están integradas en *CodeRunner,* luego se nombra la pregunta y se redacta el enunciado del ejercicio que deberá resolver el estudiante. A continuación, se especifica la respuesta esperada del estudiante, después se

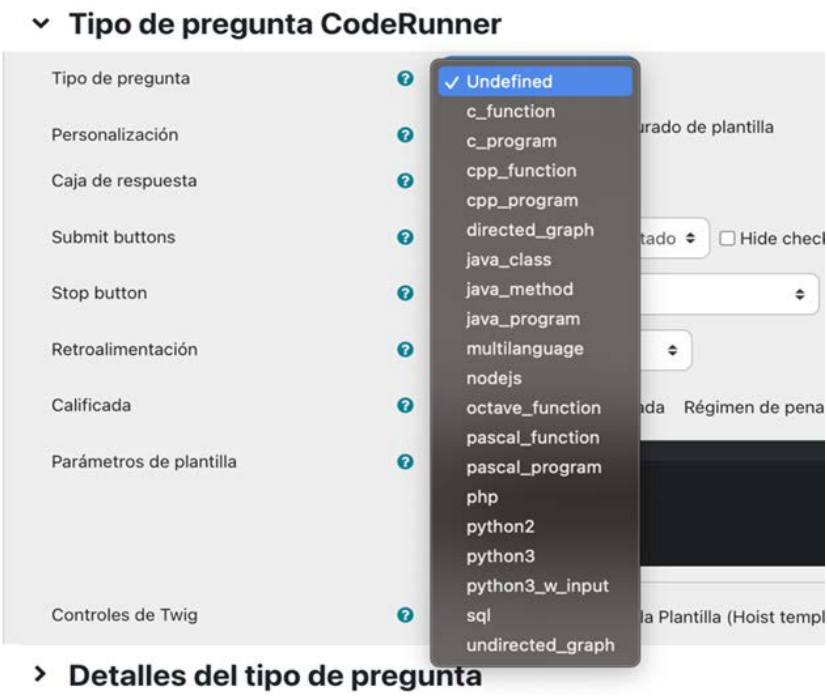
escriben uno o más casos de prueba, y, finalmente, al guardar los cambios, la pregunta queda lista en el examen. Lo anterior se configura en las secciones mostradas en la figura 3, que se muestran una vez que la pregunta se haya añadido al examen. En los siguientes párrafos se describe cada una de estas secciones.

Figura 3
Secciones de configuración de preguntas CodeRunner



La primera sección que aparece en pantalla permite seleccionar el tipo de pregunta y el lenguaje con el que trabajará el estudiante. Esto se muestra en la figura 4.

Figura 4
Opciones de la sección "Tipo de pregunta CodeRunner"



Los elementos a configurar en la sección "Tipo de pregunta *CodeRunner*" son los siguientes:

1. Tipo de pregunta. Permite elegir entre los tipos de preguntas integradas que son (Lobb y Hunt, 2023):

a) Funciones en C *(c_function)*: para funciones en lenguaje C, con pruebas en la forma *printf(format_string, func*(arg1, arg2, ..)).

b) Funciones en C++ *(cpp_function)*: para funciones en C++, con pruebas en la forma cout << func(arg1, arg2, ..).

c) Programas en C y C++ *(c_program y cpp_program)*: que requieren que el estudiante haga un programa completo. Para cada caso de prueba se ofrece un *stdin* y se especifica el *stdout* esperado. El programa se compila y en el modo de calificación "todo o nada", el código del estudiante debe producir la salida correcta para todos los casos de prueba para considerarse como correcto.

d) *Python3*: en donde para cada caso de prueba, el código del estudiante se ejecuta primero, seguido por el código de prueba.
e) *Python3_w_input*: que es una variante de las preguntas *Python3*, en el que la función input se redefine al principio del programa para que los caracteres de entrada ingresados se impriman conforme sean escritos en el teclado cuando son probados.
f) Python2: que soporta preguntas en la versión dos de *Python*.
g) Métodos en *Java*: en donde los casos de prueba hacen una llamada a los métodos escritos por el estudiante e imprimen los resultados.
h) Clases en *Java*: en donde los casos de prueba van incluidos en el método main de una clase test pública.
i) Programas en Java: donde se espera que el estudiante escriba un programa completo.
j) Funciones en *Octave*: que sirve para entregas tipo *Matlab*.
k) PHP: donde la respuesta esperada es un archivo PHP con el código contenido en etiquetas <?php ... ?> y la salida es el contenido HTML fuera de las etiquetas PHP.

Además, también pueden incluirse preguntas de tipo *nodejs* y programas y funciones en *Pascal*. Una descripción más completa de estos tipos de preguntas se puede encontrar en (Lobb y Hunt, 2023).

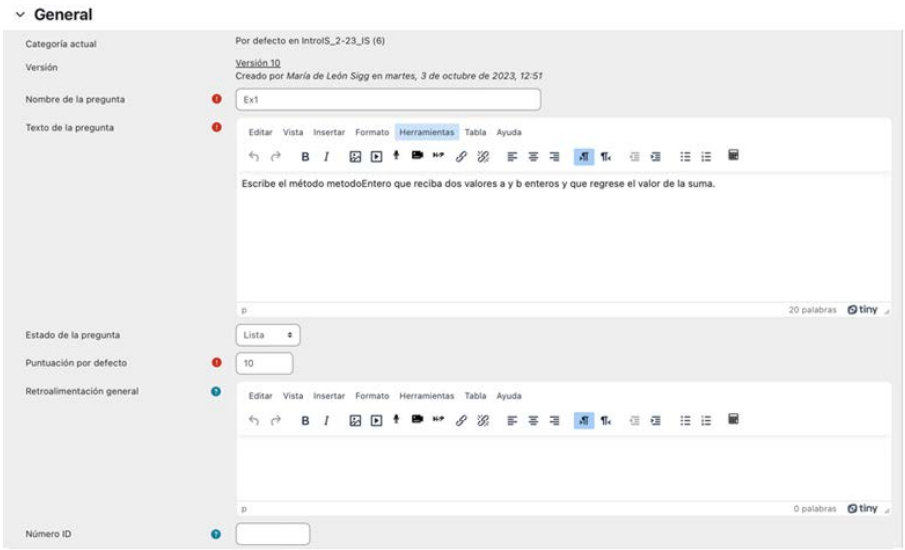
2. Personalización. Permite editar la plantilla de la pregunta de tal manera que se puede cambiar la configuración de las preguntas al seleccionar "Personalizar". Estos cambios se hacen directamente en la plantilla de la pregunta que está escrita en la sintaxis de *Twig*, que es un motor de plantillas escrito para PHP.
3. Caja de respuesta. Permite aumentar o disminuir la cantidad de filas que se esperan como parte de la respuesta.
4. Botón de entrega. Permite configurar opciones de entrega para los estudiantes. Desde aquí se puede determinar si se permitirá que los estudiantes revisen su código sin penalización antes de enviarlo, en el caso de que algunos de los casos de prueba se incluyan como ejemplos.
5. Botón de alto. Esta opción permite a los estudiantes detener la interacción con la pregunta y pasar a la retroalimentación.

6. Retroalimentación. Esta opción permite controlar la retroalimentación a la respuesta de los estudiantes para definir si se mantendrá la configuración que se haya hecho al crear el examen o si se forzará el mostrarla u ocultarla.
7. Calificada. Permite configurar cómo se calificará la respuesta de los estudiantes y el régimen de penalización a utilizar por cada intento que hagan para responder la pregunta.
8. Parámetros de plantilla. Permite modificar los parámetros de la plantilla de la pregunta para configurar cómo se presentará la pregunta a los estudiantes. Por ejemplo, se puede configurar para que se presenten versiones aleatorias de cada pregunta a diferentes estudiantes, restringir el uso de algunas funciones o la cantidad de líneas en las que se debe resolver el ejercicio. Un listado completo de los parámetros de la plantilla y cómo se pueden modificar está descrito en la información del *plugin* en *GitHub* (Lobb y Hunt, 2023).
9. Controles *Twig*. Permite configurar el preprocesador utilizado para las plantillas de las preguntas. Esta configuración depende de los parámetros que se hayan elegido modificar y que se describieron en el punto 8.

La sección "Detalles del tipo de pregunta" describe brevemente cómo *CodeRunner* interpreta al tipo de pregunta, ampliando la información mostrada en la sección anterior. Los detalles del tipo de pregunta permiten identificar, por ejemplo, cómo se deben escribir los casos de prueba o cómo se inserta el código del estudiante dentro de la plantilla para que pueda ser incluido en la plantilla de la pregunta. El contenido de esta sección varía en función del tipo de pregunta que se haya seleccionado, como se describió en el primer punto de la sección "Tipo de pregunta CodeRunner".

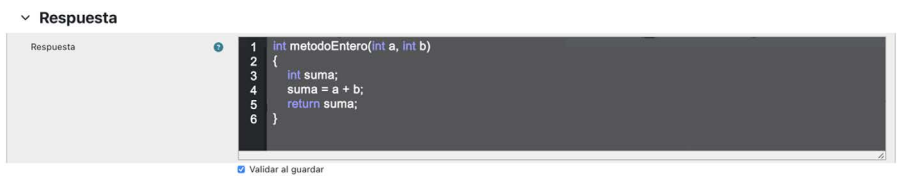
En la sección "General", mostrada en la figura 5 es donde se redacta la pregunta. Hay tres campos obligatorios: el nombre de la pregunta, su enunciado y la puntuación por defecto. También en esta sección se puede indicar si la pregunta ya está lista o todavía se va a editar, así como proporcionar la retroalimentación que recibirá el estudiante al responderla.

Figura 5
Identificación de la pregunta



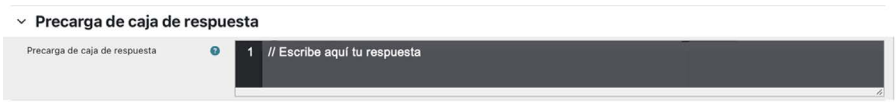
En la sección "Respuesta" se describe una respuesta de muestra esperada. Esta respuesta se valida si se selecciona el control correspondiente, como se muestra en la figura 6. Se recomienda seleccionar la casilla "Validar al guardar" para asegurar que el código de la respuesta es insertado adecuadamente en la plantilla de la pregunta.

Figura 6
Respuesta esperada del estudiante



En la sección "Precarga de caja de respuesta", mostrada en la figura 7, se puede incluir texto que ayude al estudiante a determinar qué respuesta es la que se espera.

Figura 7
Información en caja de respuesta



En la sección "Global extra" se puede incorporar un campo de texto de propósito general que funcione de manera global para todas las pruebas. Este campo se utiliza cuando se modifica la plantilla de preguntas.

En la sección "Casos de prueba" se pueden añadir tantos casos de prueba como se considere necesario para probar el código de estudiante. En la figura 8 se muestran los campos a llenar para cada caso de prueba.

Figura 8
Configuración de los casos de prueba

Estos casos de prueba deben documentarse de acuerdo con el tipo de pregunta realizada. Como mínimo, para cada caso de prueba, se debe determinar el escenario en el que se probaría el código y la salida esperada una vez ejecutado. Además, en esta sección se puede configurar la prueba para que funcione como ejemplo y ayude al estudiante a comprender mejor lo que se está solicitando en su respuesta. También se puede elegir mostrar, ocultar, ocultar si falla u ocultar si funciona el código del estudiante. Esta funcionalidad es útil para asegurar que el código del estudiante pase casos

de prueba que no necesariamente le sean visibles y evitar la posibilidad de construir código que solamente pase los casos expuestos. También se puede configurar el puntaje para cada caso de prueba cuando no se configuró la pregunta como "Todo o nada" en la sección "Tipo de pregunta *CodeRunner*" (ver figura 2). Finalmente, en esta sección también se puede configurar el orden en que se guardan los casos de prueba.

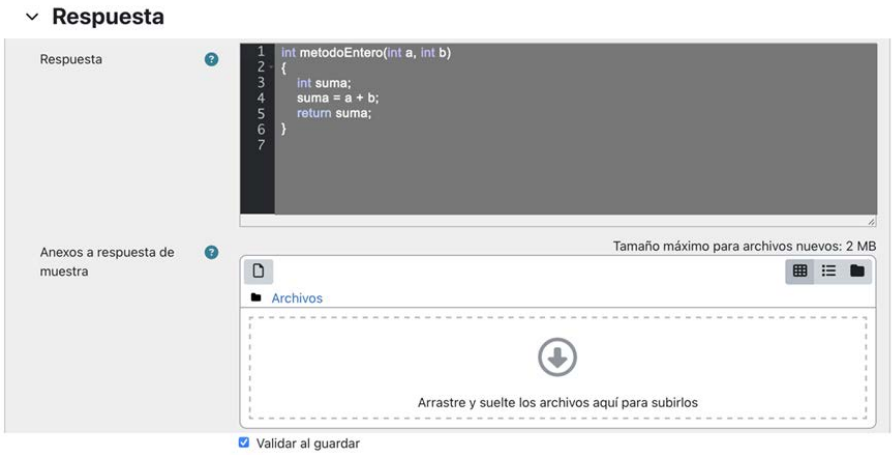
En la sección "Archivos de soporte", se pueden agregar archivos que contengan cantidades significativas de datos u otra información necesaria para preguntas específicas. Estos archivos de soporte se guardan en el mismo directorio de trabajo de Moodle, por lo que su tamaño está limitado por la configuración del curso para archivos subidos. El manejo de estos archivos depende de las instrucciones que se añadan o modifiquen a la plantilla de la pregunta *CodeRunner*.

Además, también es posible permitir la anexión de archivos para las respuestas de los estudiantes. Esto se configura en la sección "Opciones de Anexo". Las opciones para adjuntar anexos incluyen la opción de requerirlos de manera opcional u obligatoria, hasta un total de tres archivos adjuntos. También es posible configurar el nombre de los archivos esperados mediante expresiones regulares y el tamaño máximo de los archivos en un rango de 1 KB hasta 100 MB. Sin embargo, se recomienda tener precaución con estos archivos para evitar problemas de almacenamiento en el servidor Jobe. Estas dos últimas secciones se ilustran en la figura 9.

Figura 9
Sección para añadir archivos de soporte y sección para configurar los archivos anexados en las respuestas

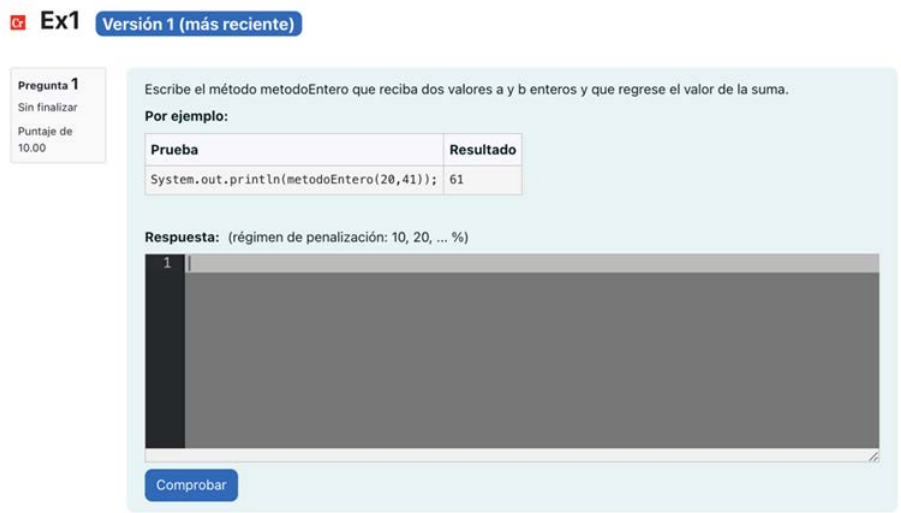
Cuando se solicita una opción de anexo, la sección de "Respuesta" cambia para indicar que la respuesta del estudiante debe incluir un archivo adjunto. Este cambio se muestra en la figura 10:

Figura 10
Actualización de la sección de Respuesta cuando se configura la anexión de archivos



Una vez que se ha terminado de configurar la pregunta, esta queda insertada en el examen. Un estudiante la verá tal como se muestra en la figura 11. En el caso de esta pregunta, uno de los casos de prueba se ha seleccionado como ejemplo.

Figura 11
Vista previa de una pregunta CodeRunner

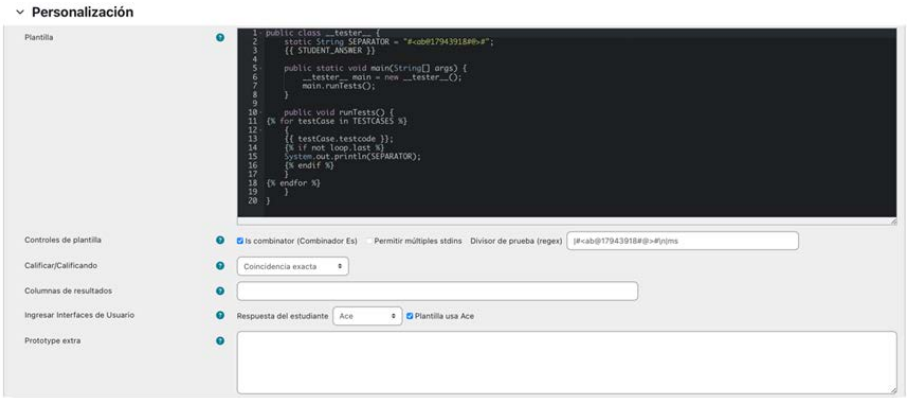


Si el código del estudiante contiene errores, *CodeRunner* los indica y señala dónde se detectaron, como se observa en la sección en rojo de la figura 12.

Figura 12
Retroalimentación de CodeRunner cuando hay errores en las respuestas

En el caso mostrado en la figura 12, donde la pregunta es del tipo "método de *Java*", el error se reporta en la clase *_tester_*, que forma parte de la plantilla del tipo de pregunta. Para poder ver la plantilla, se deben seleccionar las casillas "Personalizar" y "Depurado de plantilla" que aparecen en la sección "Tipo de pregunta *CodeRunner*", justo abajo del selector de tipo de pregunta. Al hacerlo, se hace visible la sección "Personalización", que se muestra en la figura 13.

Figura 13
Plantilla de pregunta CodeRunner



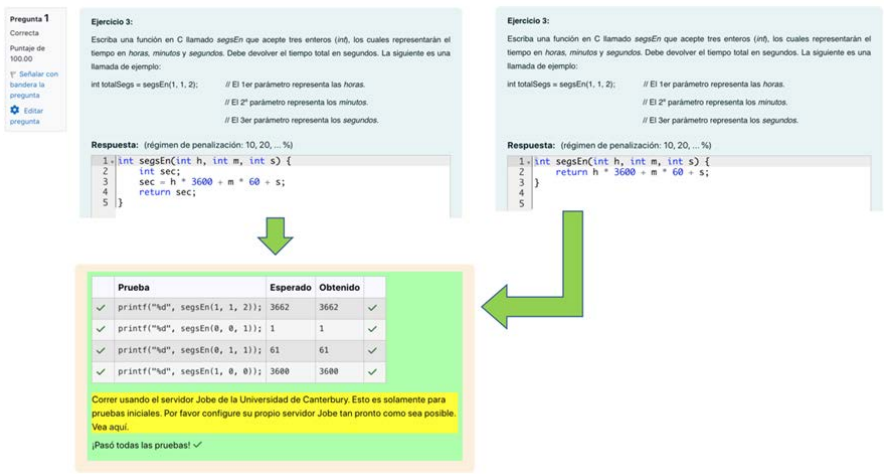
Si el código del estudiante es correcto, pasará todos los casos de prueba, incluyendo el que se marcó como ejemplo y aquellos que se hayan ocultado. En la figura 14 se muestra cómo vería el estudiante su pantalla luego de responder correctamente.

Figura 14
Vista del estudiante cuando respondió correctamente el ejercicio

Las plantillas, como la mostrada en la figura 13, muestran cómo se prueban las respuestas entregadas por los estudiantes. Cada uno de los tipos de preguntas mencionados en la figura 4 tiene su propia plantilla con espacios específicos para insertar las respuestas y el código de los casos de prueba.

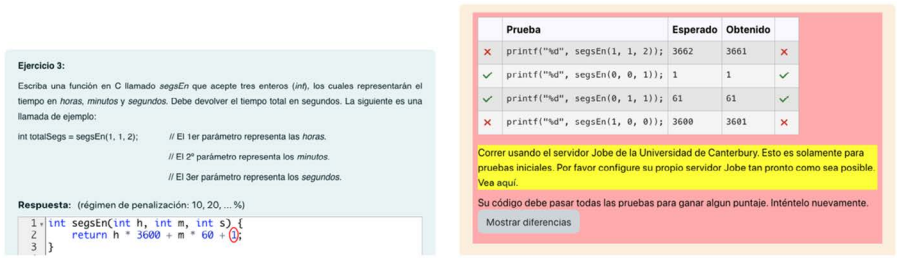
El código formado por la plantilla, las respuestas del estudiante y el código de prueba se envían a un entorno aislado, o *sandbox*, donde se compilan, cuando es necesario, y luego se ejecutan con la salida declarada en el caso de prueba. La salida obtenida en el *sandbox* se compara con la salida esperada. Este proceso se realiza mediante *Twig*, de tal manera que cuando se procesa la plantilla, el resultado es un programa completo, dependiendo de la respuesta del estudiante (Lobb y Hunt, 2023). Este procesamiento permite que *CodeRunner* detecte como correcto el código que tenga variaciones sencillas, como el código para una función en C que se muestra en la figura 15.

Figura 15
Variaciones en el código aceptadas



En caso de que el estudiante escriba un código que pase algunas pruebas, pero no otras, podría presentarse un caso como el que se muestra en la figura 15, donde el código suma 1 a la instrucción necesaria para resolver el ejercicio, haciendo que algunos casos de prueba satisfagan la instrucción y otros no. *CodeRunner* detecta esto y lo muestra al estudiante como casos de prueba no exitosos. En este caso todos los casos de prueba son visibles al estudiante. Por esta razón, se sugiere que para cada pregunta existan varios casos de prueba configurados de manera diferente, utilizando las opciones mostradas en la figura 8.

Figura 16
Comportamiento de CodeRunner cuando algunos casos de prueba no pasan



Las plantillas de las preguntas pueden ser modificadas para crear tipos de preguntas adicionales a las integradas en *CodeRunner*. Toda la información sobre cómo modificar las plantillas para cambiar los tipos de preguntas integradas y añadir variaciones a la forma en la que son calificadas se encuentra en detalle en el repositorio de GitHub de CodeRunner (Lobb y Hunt, 2023).

Algunas consideraciones para tener en cuenta

Aunque CodeRunner ofrece una amplia variedad de tipos de preguntas en diferentes lenguajes de programación, su funcionalidad se puede ampliar aún más con la adaptación de las plantillas de preguntas, incluso para aquellas que involucren el uso de gráficos. Sin embargo, para hacerlo adecuadamente, es necesario comprender su arquitectura y dedicar suficiente tiempo para utilizarlo al máximo, especialmente cuando las respuestas esperadas aumentan de complejidad y se busca mejorar el nivel de retroalimentación a los estudiantes.

Por lo tanto, su uso es más conveniente para tareas simples con especificaciones claras, como lo explican sus creadores, R. Lobb y J. Harlow (Lobb y Harlow, 2016). Otro aspecto importante es la necesidad de configurar adecuadamente el servidor *Jobe* antes de que el curso en el que estén las preguntas se utilice, para evitar saturar el espacio del servidor *Jobe* de la Universidad de Canterbury, así como controlar el tamaño máximo de los archivos de soporte y los archivos adjuntos esperados en las respuestas. El tamaño de estos archivos se limita desde la configuración del curso en

el que estará el examen con las preguntas *CodeRunner,* pero es importante tenerlo en cuenta.

Referencias

Cardone, F., Rabellino, S. y Roversi, L. (2022). Un assetto moodle per l'esame online di un corso di programmazione. In E. I. MediaTouch 2000 (Ed.), *MoodleMoot Italia 2021* (pp. 45-52). Torino: Università degli Studi di Torino.

Hatano, K. (2021). *Framework to analyze logs of coderunnder for improving programming education.* 8th International Conference on Educational Technologies 2021, ICEduTech 2021 and 17th International Conference on Mobile Learning 2021, ML 2021 (pp. 269-270). IADIS Press.

Lobb, R., y Harlow, J. (2016). Coderunner: A Tool for Assessing Computer Programming Skills. *ACM Inroads, 7*(1), 47-51. https://doi.org/10.1145/2810041

Lobb, R., y Hunt, T. (1 de febrero de 2017). CodeRunner. Retrieved Octubre 2023, from Moodle Plug-Ins: https://moodle.org/plugins/qtype_coderunner/3.1.2/13150

Lobb, R., y Hunt, T. (2023, 18 de septiembre). CodeRunner. Retrieved Octubre 2023, from GitHub: https://github.com/trampgeek/moodle-qtype_coderunner#coderunner