

Módulo de Simulaciones y Pruebas de Código con Inteligencia Artificial en Moodle para Retroalimentación Instantánea

MEMORIA DE RESULTADOS

EQUIPO:

André Filipe Sales Mendes (Coordinador)

Juan Francisco de Paz Santana

María Navarro Cáceres

Daniel Hernández de la Iglesia

Héctor Sánchez San Blas

Raúl García Ovejero

José Torreblanca González

Lydia Rozas Izquierdo

Diego Manuel Jiménez Bravo

Yanira Navarro Marrero

Rocío Galache Iglesias

Sergio García González

David Cruz García

Sergio Fernández Marcos

Contenido

- Datos del proyecto..... 3
- 1. Introducción..... 4
- 2. Estado del arte 5
- 3. Objetivos del proyecto 7
- 4. Arquitectura..... 9
- 5. Funcionamiento del módulo en el entorno Moodle 12
- 6. Evaluación de la plataforma 31
- 7. Repositorio del proyecto 35
- 8. Resultados y Conclusiones..... 36
- 9. Lineas futuras..... 37
- 10. Plan de difusión 38
- Agradecimientos..... 39
- Referencias 40

Datos del proyecto

Título: Módulo de Simulaciones y Pruebas de Código con Inteligencia Artificial en Moodle para Retroalimentación Instantánea

Referencia: ID2024/031

Coordinador del proyecto: André Filipe Sales Mendes

Organismo: Universidad de Salamanca

Centro: Facultad de Ciencias

Departamento: Informática y Automática

Investigadores que forman el equipo:

- Juan Francisco de Paz Santana
- María Navarro Cáceres
- Daniel Hernández de la Iglesia
- Héctor Sánchez San Blas
- Raúl García Ovejero
- José Torreblanca González
- Lydia Rozas Izquierdo
- Diego Manuel Jiménez Bravo
- Yanira Navarro Marrero
- Rocío Galache Iglesias
- Sergio García González
- David Cruz García
- Sergio Fernández Marcos

Duración: curso académico 2024-2025

1. Introducción

La enseñanza de la programación requiere no solo una sólida base teórica, sino también oportunidades continuas de práctica y experimentación. En este proceso, **la retroalimentación inmediata** juega un papel fundamental: permite al estudiante detectar errores, corregir conceptos y mejorar su comprensión en tiempo real. Sin embargo, los entornos tradicionales de enseñanza, incluso en plataformas virtuales, no siempre ofrecen esta capacidad de forma eficiente o escalable.

Con este reto en mente, nace el *plugin* que hemos denominado **QuickCodeIA**, un módulo de simulaciones y pruebas de código con inteligencia artificial integrado en la plataforma Moodle. El objetivo principal del proyecto es ofrecer un entorno interactivo donde el estudiante pueda programar, ejecutar y depurar código dentro de Moodle, recibiendo asistencia inmediata gracias a la integración de modelos de IA.

A diferencia de otros sistemas basados en IA que ofrecen soluciones completas o resuelven directamente el ejercicio, el *plugin* **QuickCodeIA ha sido diseñado con un enfoque pedagógico específico: la IA actúa como guía, no como sustituto del razonamiento del estudiante**. El sistema está configurado para sugerir pistas, interpretar errores comunes, ofrecer retroalimentación parcial y orientar al alumno hacia la solución, sin proporcionar directamente el código final.

Este enfoque fomenta un aprendizaje más profundo, basado en la resolución progresiva y el desarrollo de competencias, evitando la dependencia de respuestas automatizadas. El *plugin* representa así una herramienta innovadora para el aprendizaje autónomo, accesible desde cualquier dispositivo, y adaptable a múltiples asignaturas con componente práctico de programación.

2. Estado del arte

Importancia de la retroalimentación inmediata en programación

La enseñanza de la programación se beneficia enormemente de la retroalimentación inmediata. A diferencia de otras disciplinas, programar es un proceso iterativo en el que los estudiantes aprenden probando y corrigiendo sus códigos en tiempo real. La disponibilidad de comentarios instantáneos sobre la corrección o error de un programa permite al alumno identificar fallos rápidamente y evitar la acumulación de conceptos erróneos. La retroalimentación inmediata reduce la incertidumbre y ansiedad asociadas a no saber si una implementación es adecuada o no; incluso en exámenes, los alumnos aprecian salir sabiendo su calificación inmediatamente. En síntesis, disponer de un entorno educativo que brinde retroalimentación inmediata en programación favorece la autoevaluación continua y la autorregulación del aprendizaje.

Herramientas educativas con feedback inmediato

En los últimos años han proliferado herramientas diseñadas para la educación en programación que incorporan mecanismos de evaluación automática y respuesta en tiempo real. En Moodle, destaca CodeRunner, un *plugin* que permite a los profesores proponer preguntas cuya respuesta es un fragmento de código. CodeRunner ejecuta el código enviado por el estudiante contra casos de prueba predefinidos y devuelve al instante si la solución pasó o no las pruebas (Lobb & Harlow, 2016). Fuera de Moodle, Replit proporciona un IDE colaborativo basado en web con funcionalidades de autoevaluación. Otra herramienta muy utilizada es Jupyter Notebook, que permite una experiencia interactiva. Herramientas como nbgrader o UNCode permiten calificar cuadernos automáticamente y dar *feedback* formativo inmediato (González-Carrillo et al., 2021).

Auge de los modelos generativos: ChatGPT y GitHub Copilot

Modelos generativos como ChatGPT y GitHub Copilot actúan como asistentes de programación capaces de generar código, explicar funciones o sugerir correcciones. Estas herramientas ofrecen al estudiante una especie de tutor personal *on-demand*. Sin embargo, su uso sin restricciones puede derivar en una dependencia pasiva, comprometiendo la comprensión conceptual (Kasneji et al., 2023; Wermelinger, 2023). Algunos estudiantes han manifestado preocupación por “no entender cómo o por qué funcionan” las sugerencias generadas por IA, y se han señalado riesgos de integridad académica.

Riesgos de dependencia vs. beneficios de las pistas guiadas

Estudios en tecnología educativa muestran que proporcionar respuestas completas puede acelerar la tarea pero reducir la retención. Por el contrario, ofrecer pistas o retroalimentación parcial mantiene al estudiante involucrado en la solución del problema. Este tipo de retroalimentación guiada apoya el aprendizaje sin anularlo, combatiendo el riesgo de que la IA se convierta en una muleta. Tal como señalan Chi, Bassok, Lewis, Reimann y Glaser (1989), cuando los estudiantes reciben únicamente una ayuda parcial en lugar de soluciones completas, se ven impulsados a generar explicaciones propias, lo cual promueve un aprendizaje más profundo y duradero.

Valor pedagógico de los sistemas de tutoría inteligente

Los Sistemas de Tutoría Inteligente han demostrado mejoras significativas en el rendimiento académico al brindar tutoría adaptativa paso a paso. Estos sistemas responden inmediatamente a las acciones del alumno, simulando la atención de un profesor particular. Con el avance de la IA, los nuevos modelos generativos se integran como tutores conversacionales más flexibles, proporcionando experiencias de aprendizaje más interactivas y personalizadas (Carbonell Bernal & Hernández Prados, 2024).

3. Objetivos del proyecto

El proyecto tenía como finalidad principal desarrollar e implementar un módulo de simulaciones y pruebas de código con inteligencia artificial dentro de la plataforma Moodle, orientado a ofrecer **retroalimentación inmediata y formativa** en el aprendizaje de la programación. Esta herramienta fue concebida para fomentar la autonomía del estudiante, facilitar la práctica distribuida y mejorar la experiencia de aprendizaje en asignaturas con un fuerte componente práctico.

Los objetivos del proyecto se estructuraron en dos niveles: generales y específicos.

Objetivo general

- Desarrollar un entorno interactivo en Moodle que permitiera a los estudiantes resolver ejercicios de programación breves, con evaluación automática y asistencia contextualizada mediante inteligencia artificial, promoviendo así la práctica autónoma y la adquisición progresiva de competencias en programación.

Objetivos específicos

- **Integración técnica:** desplegar un entorno de codificación accesible desde Moodle que incluyera editor, terminal de ejecución y sistema de evaluación automática, con soporte para lenguajes como Python, C y Verilog.
- **Asistencia con IA pedagógica:** configurar un asistente conversacional basado en IA, diseñado para interpretar errores, ofrecer pistas y guiar al estudiante sin proporcionar directamente la solución completa.
- **Diseño de ejercicios breves:** crear y validar un banco inicial de más de 20 ejercicios de programación.
- **Adaptación curricular:** aplicar el módulo en distintas asignaturas de grados del ámbito de la ingeniería y las ciencias, asegurando su utilidad en contextos curriculares diversos.
- **Evaluación del impacto:** recoger evidencias mediante encuestas, análisis de uso y evaluación cualitativa con el fin de valorar la utilidad del sistema desde la perspectiva del alumnado y del profesorado.

-
- **Difusión y replicabilidad:** documentar el desarrollo del módulo, generar materiales de uso y presentar los resultados en al menos una publicación o congreso de innovación docente antes de diciembre de 2025.

Estos objetivos respondieron a la necesidad de ofrecer entornos accesibles, escalables y centrados en el estudiante, alineándose con las tendencias actuales en educación digital y aprendizaje asistido por inteligencia artificial.

4. Arquitectura

El *plugin* se estructuró mediante una arquitectura modular compuesta por dos bloques principales: la plataforma Moodle, donde reside el *plugin* desarrollado, y los servicios externos del *plugin* encargados de gestionar la ejecución de código y el soporte con inteligencia artificial. La comunicación entre ambos bloques se realiza a través de un servidor proxy, que permite el intercambio seguro de datos y el aislamiento de procesos críticos como la ejecución de código o el acceso a modelos de IA.

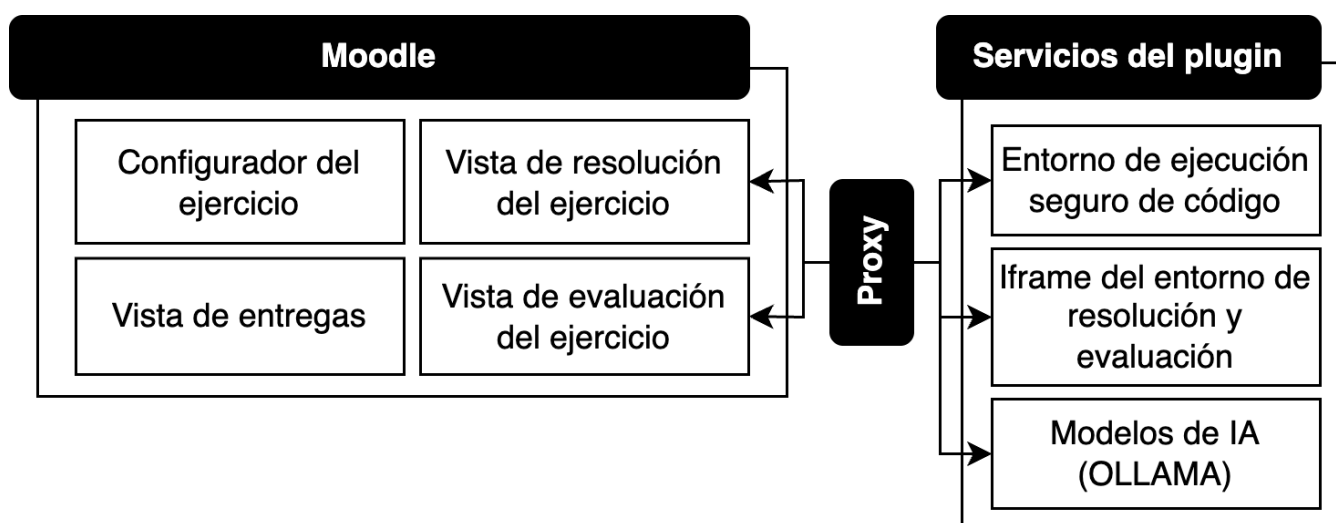


Figura 1. Arquitectura del sistema

A. Módulos en Moodle

1. **Configurador del ejercicio:** esta vista está diseñada para que el profesorado cree nuevos ejercicios de programación de forma sencilla. Permite definir el enunciado, el código de solución (no visible para el estudiante), el número de pistas disponibles y la activación o no del asistente de IA. Toda esta configuración se almacena como parte del recurso en Moodle, lo que garantiza su trazabilidad y reutilización.
2. **Vista de resolución del ejercicio:** desde esta interfaz, el estudiante accede al entorno práctico. **Esta vista embebe un *iframe*** que carga el entorno externo de edición y ejecución. El *plugin* envía al *iframe* los datos del ejercicio (enunciado, parámetros de IA, etc.), y a su vez recibe eventos como ejecuciones realizadas, pistas solicitadas o resultados obtenidos. Esta comunicación bidireccional permite construir una experiencia interactiva con retroalimentación inmediata, manteniendo la integración total dentro de Moodle.

-
3. **Vista de entregas:** esta sección del módulo permite al profesorado consultar el histórico de interacciones del alumnado con cada ejercicio. Se pueden visualizar los códigos enviados, el número de ejecuciones realizadas, las pistas utilizadas y el resultado final de cada intento. Esta información es útil para detectar dificultades comunes, valorar el proceso seguido por los estudiantes y facilitar una evaluación formativa.
 4. **Vista de evaluación del ejercicio:** desde esta vista, el profesorado puede calificar manual o automáticamente las entregas, según la configuración del ejercicio. Además, se pueden añadir comentarios personalizados y revisar las respuestas. Esta integración dentro de Moodle permite que las calificaciones se reflejen automáticamente en el libro de calificaciones del curso.

B. Servicios del *plugin*

Los servicios del *plugin* se encargan de proporcionar las funcionalidades necesarias para la ejecución del código, la interacción con inteligencia artificial y la presentación del entorno práctico embebido. Estos servicios se ejecutan fuera del núcleo de Moodle y se comunican con él a través de un **proxy**, lo que permite escalar el sistema y mantener la seguridad del entorno educativo.

A continuación, se describen sus tres componentes principales:

1. **Entorno de ejecución seguro de código:** este módulo se encarga de ejecutar el código que el estudiante introduce desde la interfaz integrada en Moodle. Para garantizar la seguridad y el aislamiento, el sistema emplea contenedores de Docker configurados que evitan la ejecución de código potencialmente malicioso. Cada ejecución se realiza de forma controlada, y los resultados (salida estándar, errores, etc.) se devuelven al entorno visual del estudiante en tiempo real. Este servicio soporta múltiples lenguajes de programación como Python, C o Verilog, y está diseñado de forma modular, lo que permite incluir nuevos lenguajes de manera sencilla y rápida, adaptándose fácilmente a las necesidades de distintas asignaturas.
2. **Iframe del entorno de resolución y evaluación:** este componente representa la **interfaz interactiva que el estudiante visualiza dentro de Moodle**. Está desarrollada con tecnologías web y se encarga de presentar:
 - El editor de código.
 - La consola de salida.
 - El acceso al asistente con IA.

-
- Las herramientas para ejecutar el código, solicitar pistas y consultar resultados.

El *iframe* se comunica con Moodle mediante parámetros URL y eventos *postMessage*, lo que permite una interacción fluida sin necesidad de recargar la página. Debido a esto, es posible mantener el flujo de trabajo del estudiante dentro del propio campus virtual, pero con una experiencia mucho más rica que la que ofrecen los formularios nativos de Moodle.

3. **Modelos de IA:** el sistema de asistencia con inteligencia artificial se basa en modelos de lenguaje desplegados localmente mediante la plataforma OLLAMA, que permite utilizar modelos como LLaMA, Mistral o similares sin depender de servicios externos ni enviar datos sensibles fuera del entorno universitario.

Estos modelos a través de **ingeniería de *promts*** se pueden configurar para responder de la forma más adecuada a las necesidades docentes. La IA analiza el código actual, identifica errores comunes, **ofrece pistas progresivas** y responde preguntas relacionadas con el enunciado o la lógica del ejercicio, promoviendo así un aprendizaje activo y reflexivo.

5. Funcionamiento del módulo en el entorno Moodle

Una vez instalado, el módulo **QuickCodeIA** se integra completamente en Moodle, permitiendo su uso como cualquier otra actividad del curso. Tal como se muestra en la figura, aparece en el listado de actividades disponibles al seleccionar "Añadir una actividad o un recurso", junto a herramientas clásicas como Tarea, Cuestionario o H5P. Esta integración facilita su adopción por parte del profesorado, sin requerir conocimientos técnicos adicionales ni configuraciones externas complejas.

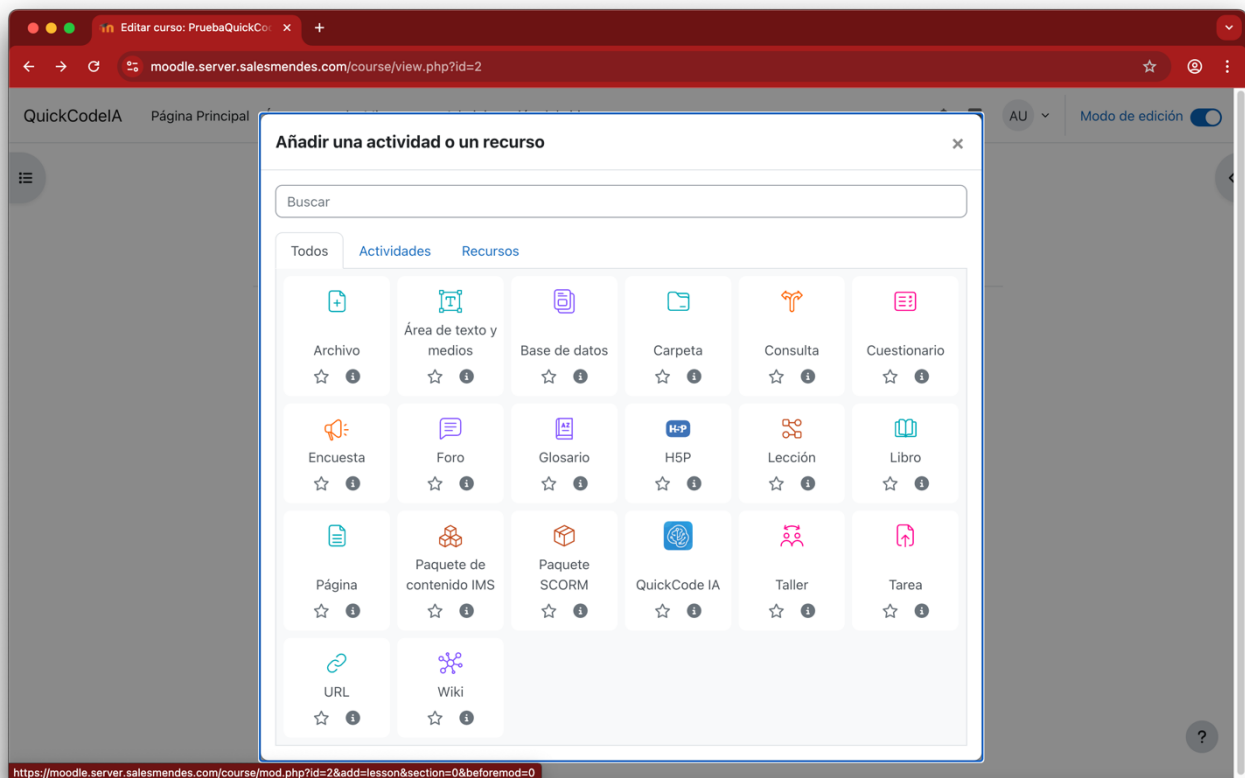


Figura 2. Integración del módulo QuickCodeIA como actividad nativa en Moodle. (Rol de profesor)

Para crear un nuevo ejercicio en el profesorado accede al formulario de configuración integrado dentro del entorno de edición de Moodle. Esta interfaz, accesible desde el listado de actividades del curso, permite definir todos los parámetros necesarios del ejercicio, incluyendo el enunciado, el lenguaje de programación, el número de pistas disponibles y el tipo de asistencia con inteligencia artificial. El formulario se divide en secciones desplegables que agrupan la configuración general, los ajustes del ejercicio y las restricciones de acceso, entre otros.

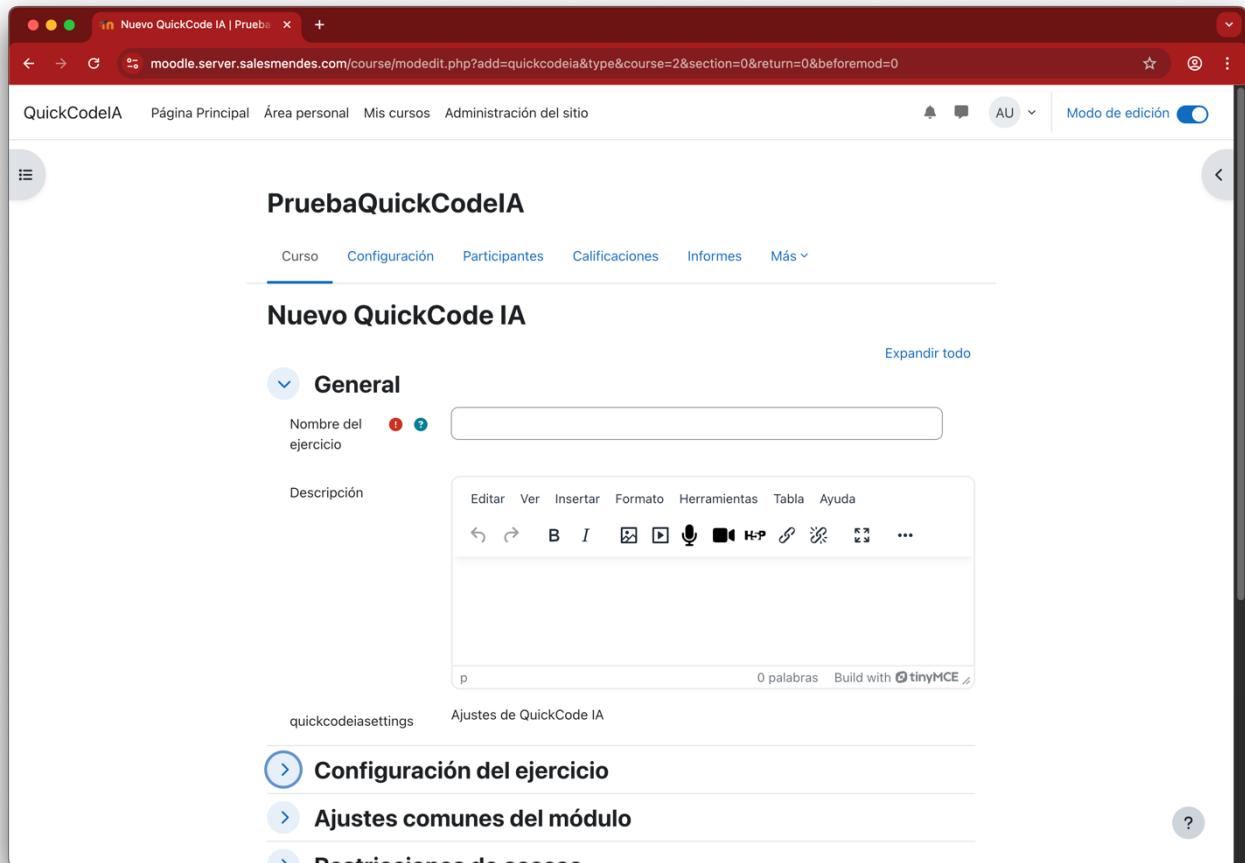


Figura 3. Interfaz de creación de una nueva actividad QuickCodeIA dentro de Moodle. (Rol de profesor)

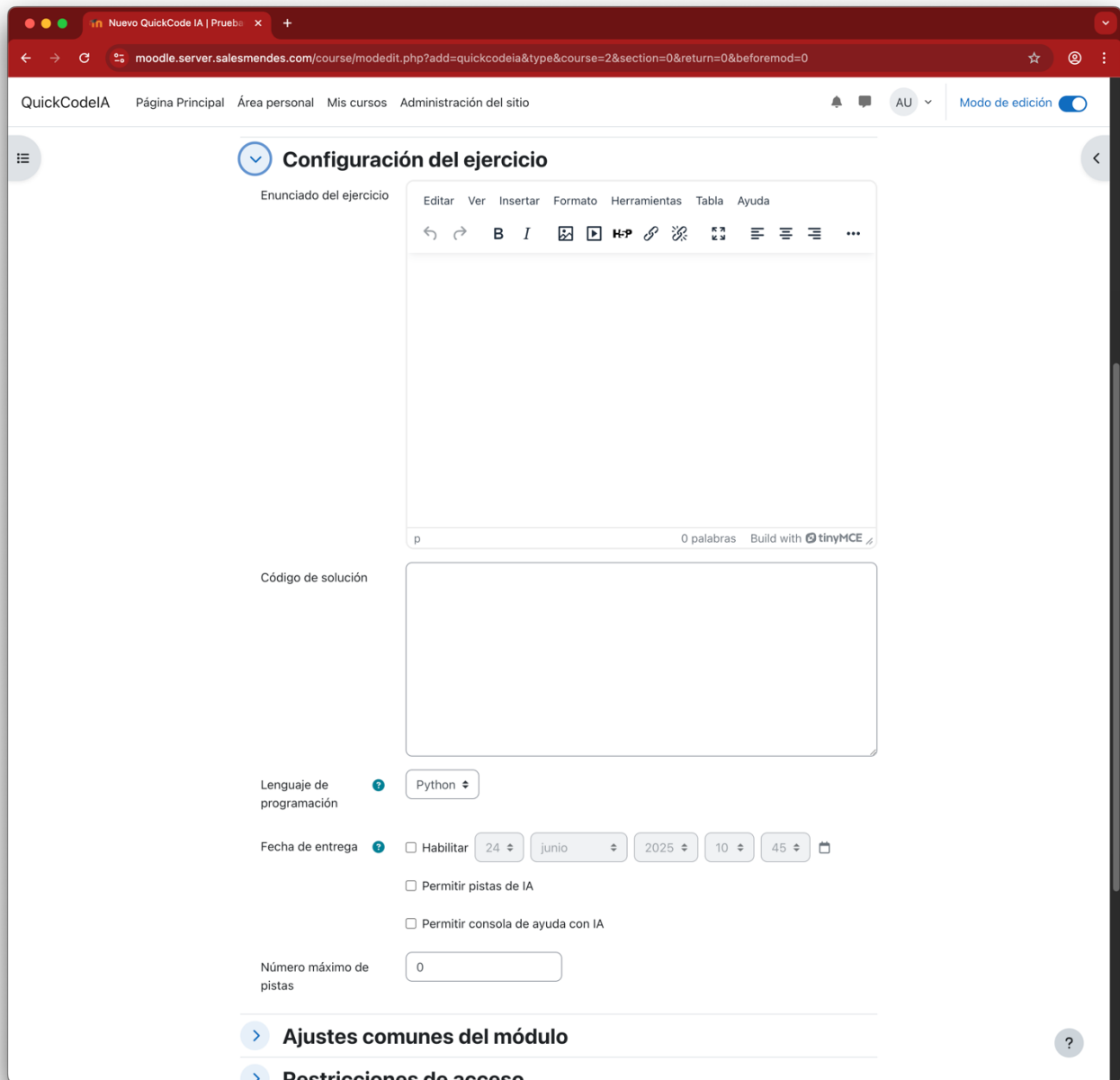


Figura 4. Panel de configuración del ejercicio en QuickCodeIA. Se permite definir el enunciado, incluir un código de solución oculto, seleccionar el lenguaje de programación, establecer una fecha de entrega y activar las opciones de ayuda con IA y número de pistas disponibles. (Rol de profesor)

Además de las configuraciones específicas del ejercicio y de la asistencia con IA, el módulo mantiene las funcionalidades estándar de cualquier actividad en Moodle. El profesorado puede gestionar parámetros habituales como la visibilidad en el curso, la asignación de un identificador, el idioma, las restricciones de acceso por fecha o condición, así como los criterios de finalización, competencias y marcas. Esto permite una integración completa con el flujo de trabajo y la estructura pedagógica de los cursos existentes.

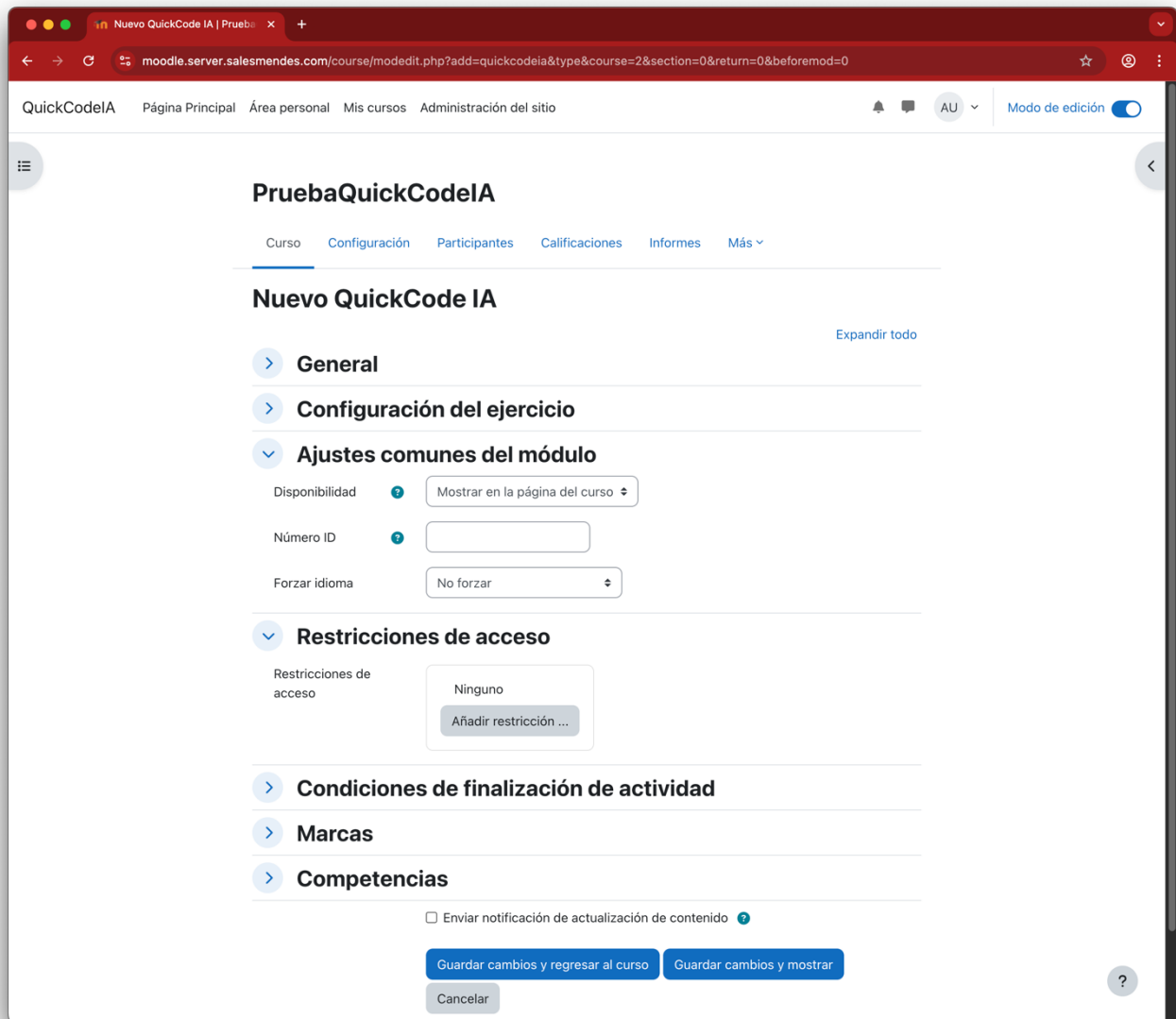


Figura 5. Opciones comunes de configuración de una actividad en Moodle. El módulo conserva los ajustes estándar del sistema como la disponibilidad, restricciones de acceso, condiciones de finalización o competencias asociadas. (Rol de profesor)

En la siguiente imagen se muestra un ejemplo real de configuración de un ejercicio en el módulo. El profesorado puede definir el enunciado (utilizando formato enriquecido), el código de solución esperado y el lenguaje de programación. Además, es posible establecer la fecha de entrega y activar las opciones de ayuda con inteligencia artificial, incluyendo el número máximo de pistas disponibles.

The screenshot shows the Moodle QuickCode IA configuration page. The browser address bar indicates the URL: moodle.server.salesmendes.com/course/modedit.php?update=3&return=1. The page title is "QuickCode IA" and the user is logged in as "AU".

The configuration is divided into two main sections:

- General:**
 - Nombre del ejercicio: Operaciones con Matriz de Tamaño Fijo
 - Descripción: A rich text editor with 0 palabras.
- Configuración del ejercicio:**
 - Enunciado del ejercicio: A rich text editor containing three numbered steps:
 - Declarar y rellenar una matriz de enteros de tamaño 3x3 con valores predefinidos.
 - Calcular la **transpuesta** de la matriz usando una función.
 - Mostrar por pantalla:
 - La matriz original.
 - La matriz transpuesta.
 - La suma de los elementos de la diagonal principal.
 - Un mensaje indicando si la matriz es **simétrica** (es decir, si es igual a su transpuesta).
 - Código de solución: A code editor containing the following C code:

```
bool esSimetrica(int matriz[SIZE][SIZE], int transpuesta[SIZE][SIZE]) {
    for (int i = 0; i < SIZE; i++) {
        for (int j = 0; j < SIZE; j++) {
            if (matriz[i][j] != transpuesta[j][i]) {
                return false;
            }
        }
    }
    return true;
}
```
 - Lenguaje de programación: C
 - Fecha de entrega: Habilitar, 24 junio 2025, 10:54. Permitir pistas de IA. Permitir consola de ayuda con IA.
 - Número máximo de: 10

Figura 6. Ejemplo de ejercicio configurado en QuickCode IA. (Rol de profesor)

Desde el punto de vista del alumnado, los ejercicios creados con *QuickCodeIA* se presentan como una actividad más dentro del curso de Moodle. Tal y como se muestra en la siguiente imagen, el estudiante puede acceder al ejercicio de forma directa desde la interfaz habitual del aula virtual, sin necesidad de instalar herramientas adicionales ni salir del entorno educativo.

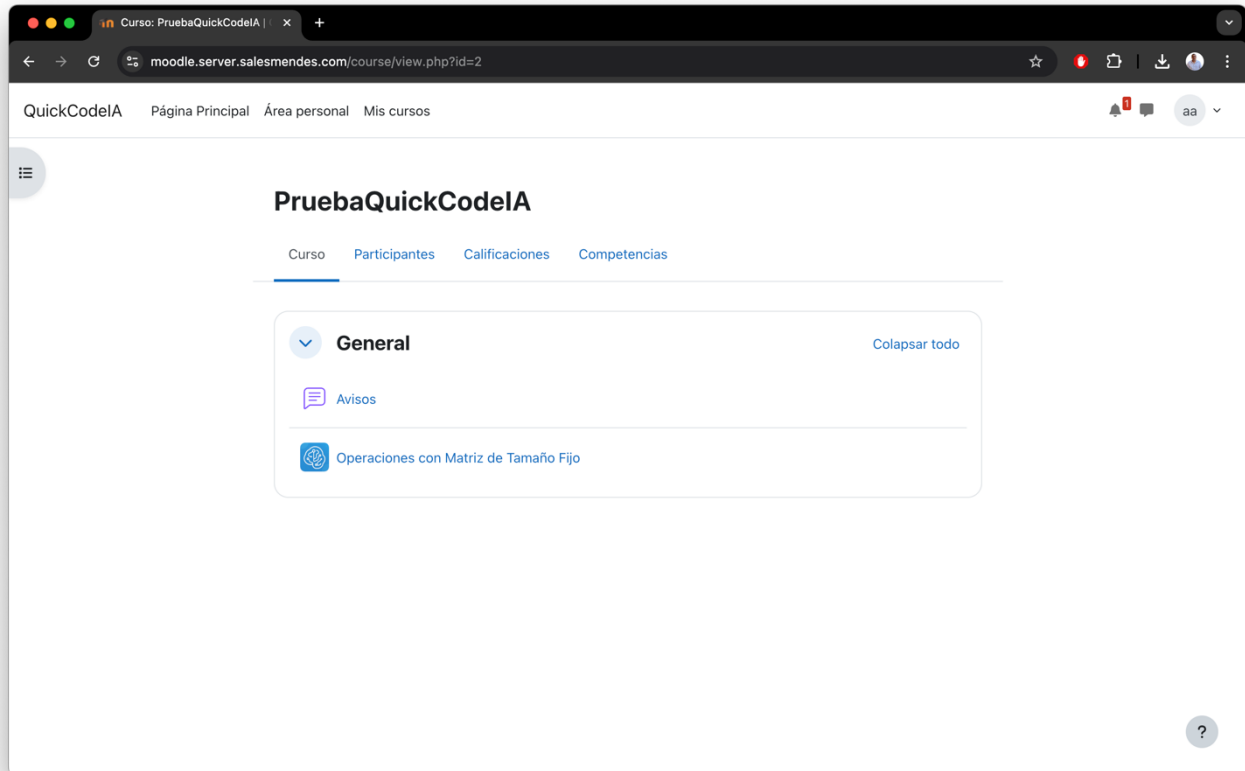


Figura 7. Vista del estudiante en Moodle. La actividad QuickCodeIA aparece integrada en la sección del curso junto con otros recursos. (Rol de estudiante)

Al acceder a un ejercicio, el alumnado visualiza una pantalla inicial con el enunciado detallado y un botón para comenzar. Una vez se pulsa “Empezar ejercicio”, el entorno de trabajo se expande a pantalla completa para aprovechar al máximo el espacio disponible, facilitando así una experiencia más inmersiva y centrada en la resolución del problema, sin distracciones del resto de la interfaz de Moodle.

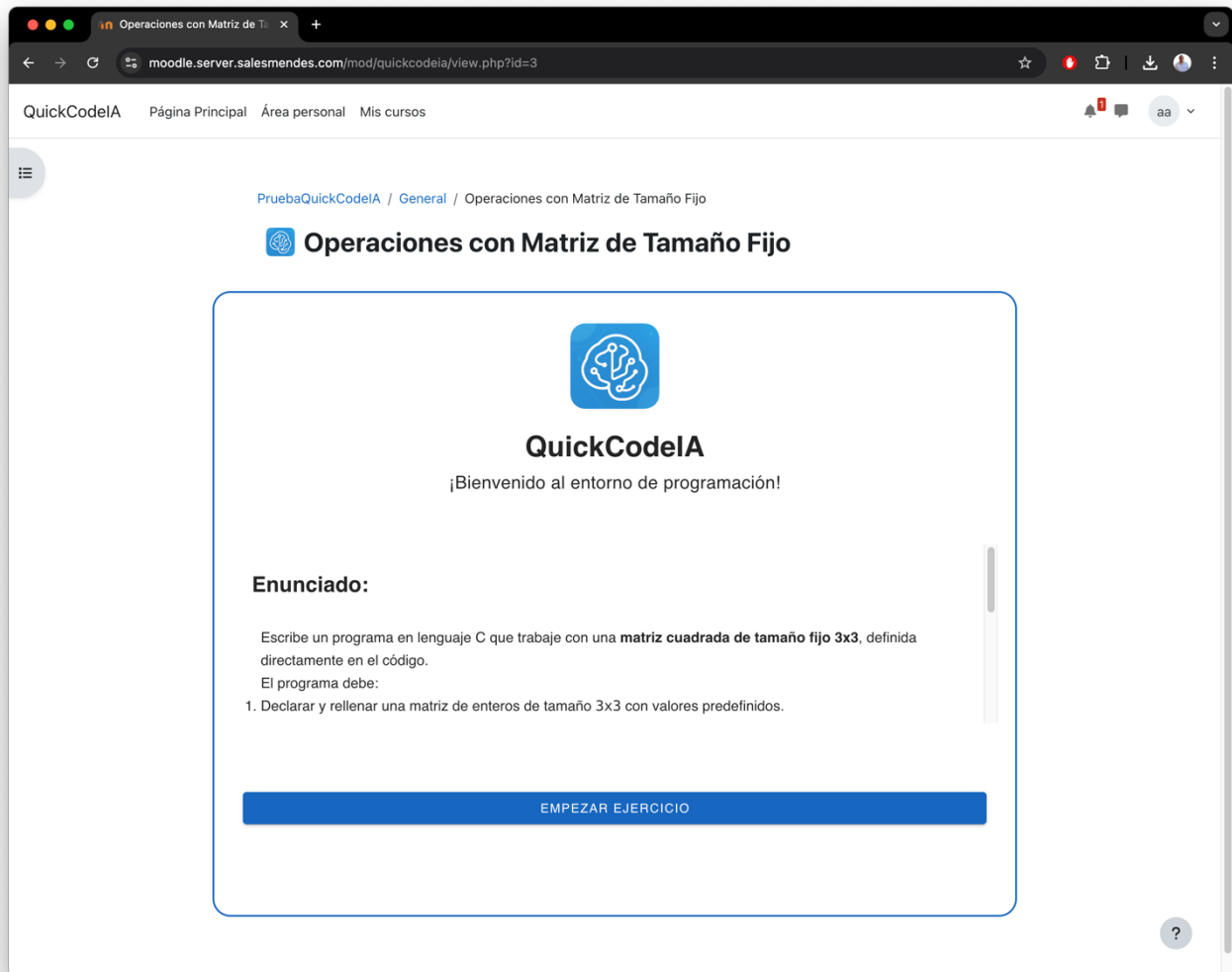


Figura 8. Pantalla de inicio del ejercicio desde la perspectiva del estudiante. El enunciado se muestra antes de acceder al entorno de resolución, que posteriormente se despliega a pantalla completa para una experiencia más inmersiva. (Rol de estudiante)

Una vez iniciado el ejercicio, el entorno se presenta en modo pantalla completa para maximizar la concentración del estudiante. A la izquierda se dispone un editor de código con resaltado de sintaxis y una terminal de salida integrada que permite compilar el programa directamente desde el entorno. En la parte derecha se mantiene visible el enunciado del ejercicio, permitiendo su consulta sin necesidad de cambiar de vista. Además, el estudiante cuenta con un botón para solicitar ayuda con la terminal, mejorando la autonomía en el proceso de depuración. Este diseño busca simular un entorno real de desarrollo, pero adaptado al contexto educativo y con herramientas de asistencia inteligentes integradas.

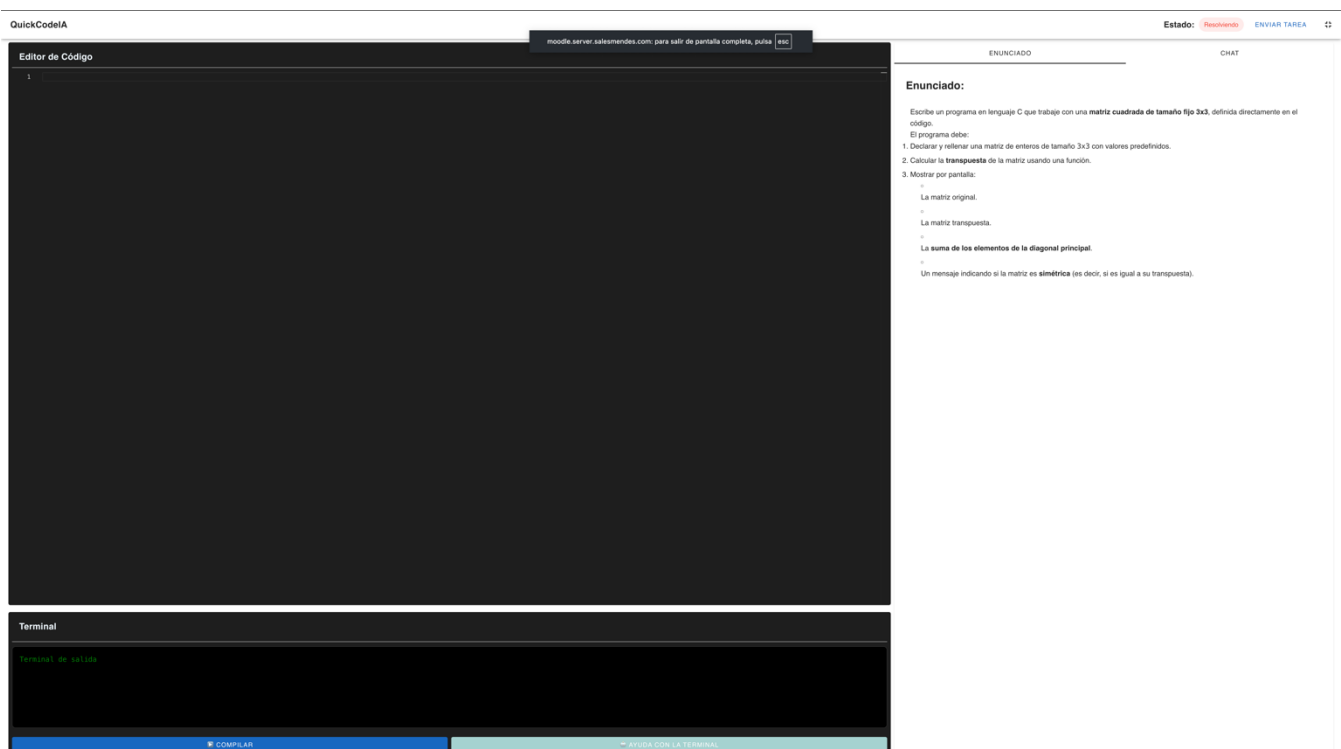


Figura 9. Vista del entorno de resolución en pantalla completa. El editor de código y la terminal están integrados junto al enunciado, favoreciendo una experiencia inmersiva y una práctica autónoma del estudiante. (Rol de estudiante)

En la siguiente imagen se muestra un ejemplo del entorno de resolución de en el que el estudiante ha introducido un código con un error de compilación.

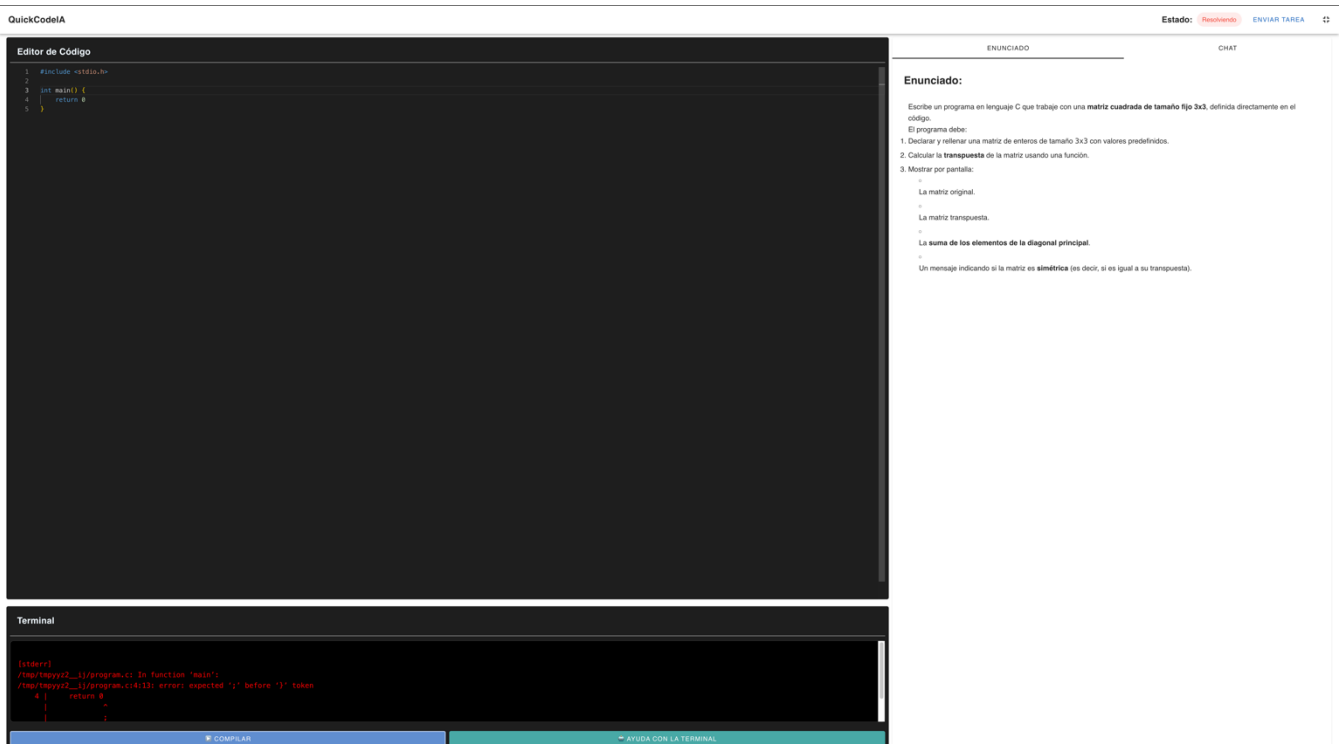


Figura 10. Ejemplo de ejecución con error en el entorno QuickCodeIA, mostrando el mensaje de compilación en la terminal. (Rol de estudiante)

En la siguiente imagen se ilustra la funcionalidad de asistencia contextual con inteligencia artificial. Tras obtener un error en la compilación del código, el estudiante ha utilizado el botón "Ayuda con la terminal" para solicitar una explicación sobre el mensaje de error mostrado. A la derecha, el asistente virtual responde con una explicación clara y detallada sobre el origen del problema y ofrece una solución concreta. Este tipo de ayuda inmediata y adaptativa permite al alumnado comprender mejor los errores y avanzar de manera más autónoma, fomentando así un aprendizaje activo, autorregulado y profundamente orientado a la resolución de problemas.

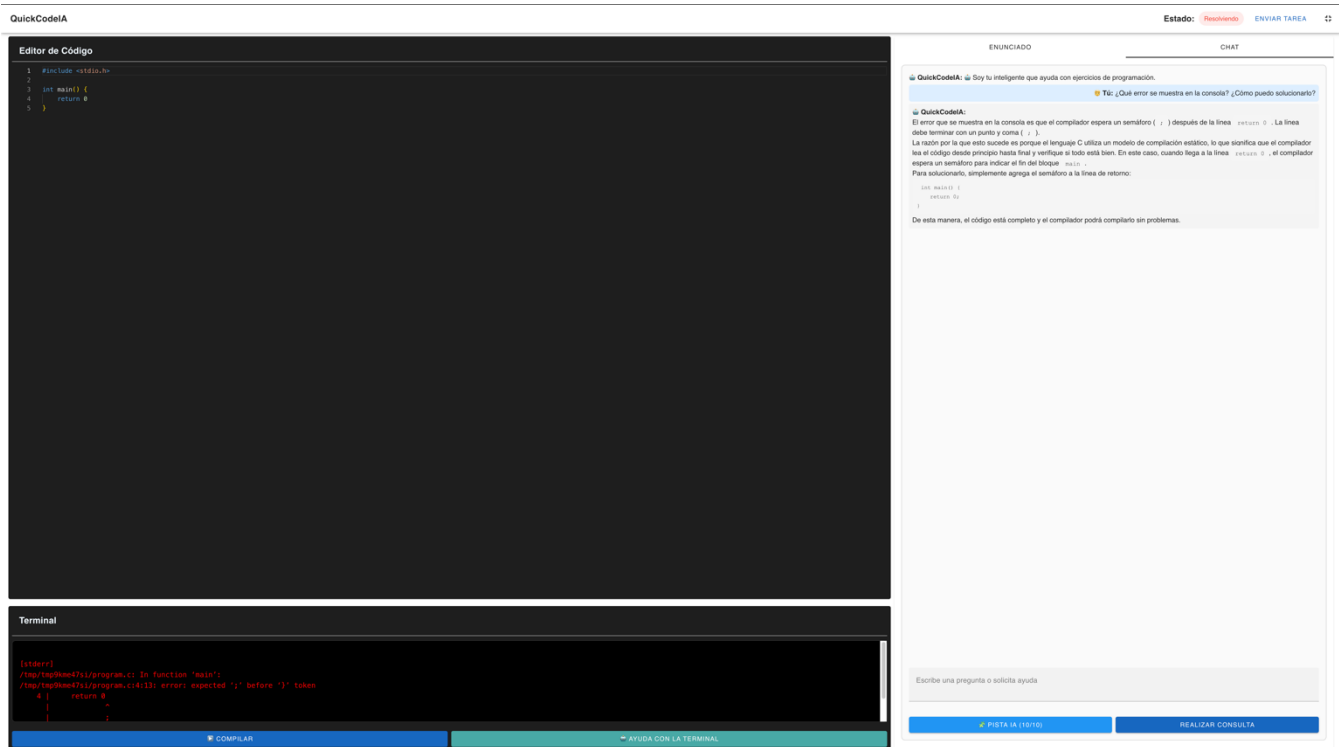


Figura 11. Solicitud de ayuda sobre el mensaje de error en la terminal, con respuesta contextual generada por la inteligencia artificial integrada. (Rol de estudiante)

En esta imagen se ejemplifica el uso del sistema de asistencia por pistas. El estudiante ha solicitado tanto una aclaración sobre el mensaje de error como una pista para avanzar en la resolución del ejercicio. A la derecha, el asistente inteligente responde primero con una explicación detallada del error de compilación, y después con una serie de recomendaciones para abordar la tarea planteada. Este tipo de retroalimentación progresiva y contextualizada permite al alumnado enfrentarse al problema desde distintas perspectivas, reforzando su comprensión y autonomía en el proceso de aprendizaje.

The screenshot displays the QuickCodeIA interface. On the left, there is a code editor titled 'Editor de Código' with the following C++ code:

```
1 #include <stdio.h>
2
3 int main() {
4     return 0;
5 }
```

Below the editor is a terminal window titled 'Terminal' with the text 'Terminal de salida'. At the bottom of the editor area are two buttons: 'COMPIRAR' and 'AYUDA CON LA TERMINAL'.

On the right, there is a chat window titled 'ENUNCIADO' and 'CHAT'. The chat contains several messages from 'QuickCodeIA' and 'Tú' (the student):

- QuickCodeIA: Soy tu inteligente que ayuda con ejercicios de programación.
- Tú: ¿Qué error se muestra en la consola? ¿Cómo puedo solucionarlo?
- QuickCodeIA: El error que se muestra en la consola es un mensaje de advertencia sobre la sintaxis incorrecta del código. La línea 4 `return 0;` está mal escrita, ya que no tiene un semáforo `;` después de `return`. En C, el bloque principal del programa debe terminar con un semáforo. Para solucionarlo, debes agregar un semáforo después de la función `main`.
- Tú: como defino una matriz?
- QuickCodeIA: Para definir una matriz en C, debes especificar el tamaño de la matriz y luego los valores del elemento que va a ocupar cada posición de la matriz. Por ejemplo:

```
int matriz[3][3];
```

Esto define una matriz 3x3 con todos sus elementos como cero. Para asignar un valor específico a una posición, puedes hacerlo así:

```
matriz[0][0] = 1;
matriz[1][1] = 2;
```

Recuerda que en C, los índices de matriz comienzan en 0.
- Tú: Me das una pista para resolver el ejercicio.
- QuickCodeIA: Pistas para resolver el ejercicio: Asegúrate de agregar un semáforo después del bloque principal `main` en la línea 4. Debes declarar la matriz antes de utilizarla. Intenta colocar la declaración de la matriz justo después de la declaración de `main`. La función que calculates la transpuesta de la matriz debe tomar la matriz como parámetro y devolver su transpuesta. ¿Cómo podrías pasar la matriz a esa función? La función que muestra por pantalla los elementos del programa necesita leer las entradas desde el teclado, ¿cómo lo harías?

At the bottom of the chat window are two buttons: 'PISTA IA (8/16)' and 'REALIZAR CONSULTA'.

Figura 12. Solicitud de pistas y ayuda contextual durante la resolución del ejercicio, generadas por el asistente de inteligencia artificial integrado. (Rol de estudiante)

En la siguiente imagen se observa la fase final del proceso de resolución de un ejercicio. En este punto, al pulsar el botón "Enviar tarea", aparece un cuadro de confirmación que informa de que, una vez enviada, la actividad no podrá ser modificada. Esta medida garantiza la integridad del proceso de evaluación.

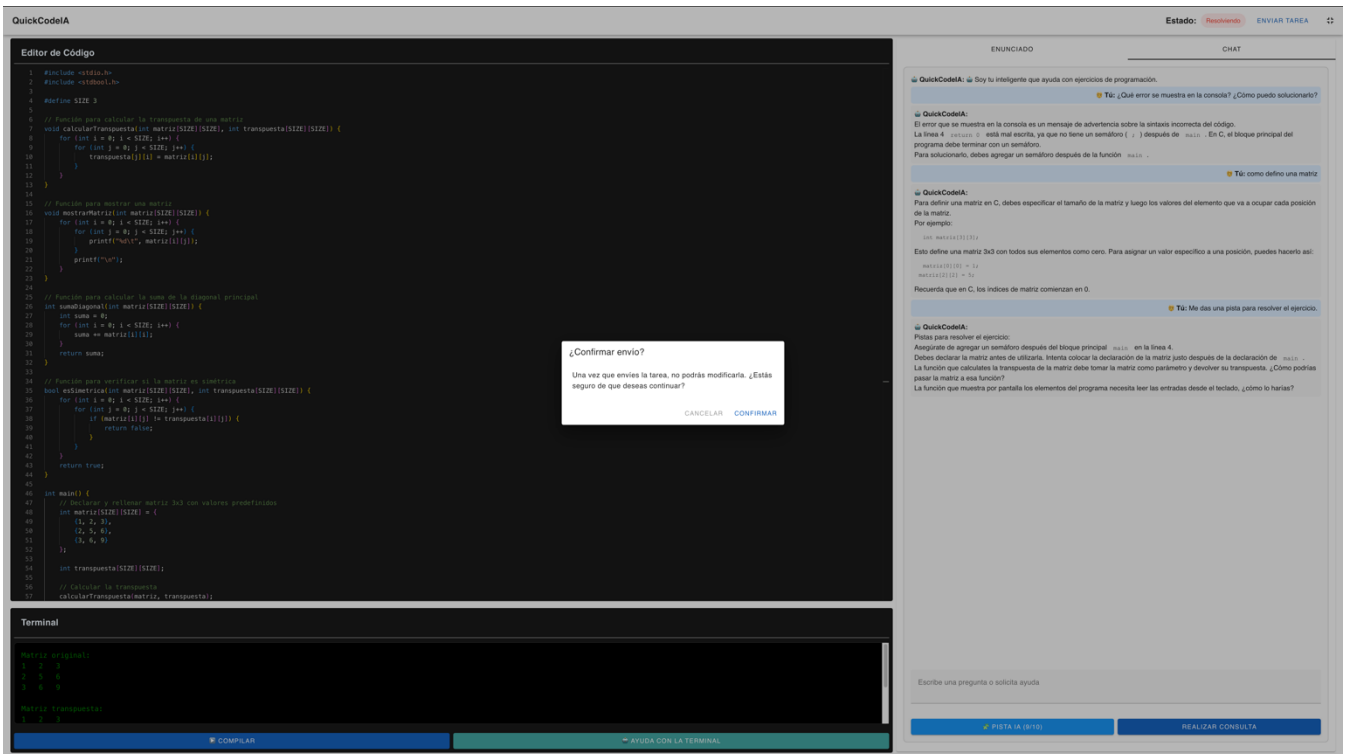


Figura 13. Confirmación de envío tras la correcta ejecución del código. El sistema solicita al estudiante validar su decisión antes de cerrar el ejercicio y remitirlo para su corrección. (Rol de estudiante)

Una vez que el alumnado ha enviado su solución, si vuelve a acceder a la vista del ejercicio, podrá revisar el código que entregó, compilarlo nuevamente y ver los resultados de ejecución en la terminal. Sin embargo, la interfaz se encuentra en modo de solo lectura, por lo que no es posible editar el código ni enviar una nueva versión. Esta funcionalidad permite repasar el trabajo realizado y analizar posibles mejoras, pero garantiza la integridad de la evaluación tras el envío.

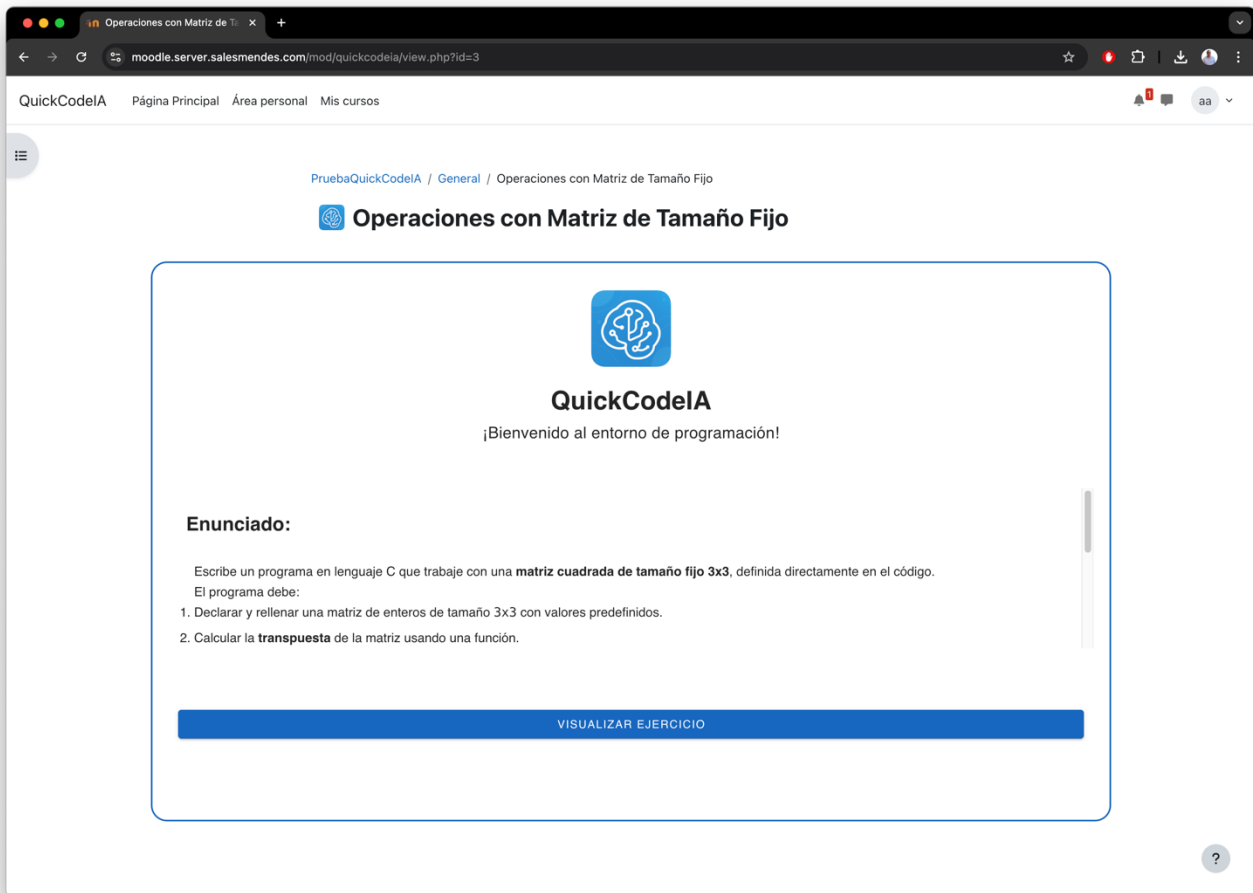


Figura 14. Vista general del ejercicio después de ser enviado. (Rol de estudiante)

evaluación.

QuickCodeA

Estado: [Enviado](#)

Editor de Código

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 #define SIZE 3
5
6 // Función para calcular la transpuesta de una matriz
7 void calcularTranspuesta(int matriz[SIZE][SIZE], int transpuesta[SIZE][SIZE]) {
8     for (int i = 0; i < SIZE; i++) {
9         for (int j = 0; j < SIZE; j++) {
10             transpuesta[j][i] = matriz[i][j];
11         }
12     }
13 }
14
15 // Función para mostrar una matriz
16 void mostrarMatriz(int matriz[SIZE][SIZE]) {
17     for (int i = 0; i < SIZE; i++) {
18         for (int j = 0; j < SIZE; j++) {
19             printf("%d\t", matriz[i][j]);
20         }
21         printf("\n");
22     }
23 }
24
25 // Función para calcular la suma de la diagonal principal
26 int sumDiagonal(int matriz[SIZE][SIZE]) {
27     int suma = 0;
28     for (int i = 0; i < SIZE; i++) {
29         suma += matriz[i][i];
30     }
31     return suma;
32 }
33
34 // Función para verificar si la matriz es simétrica
35 bool esSimetrica(int matriz[SIZE][SIZE], int transpuesta[SIZE][SIZE]) {
36     for (int i = 0; i < SIZE; i++) {
37         for (int j = 0; j < SIZE; j++) {
38             if (matriz[i][j] != transpuesta[j][i]) {
39                 return false;
40             }
41         }
42     }
43     return true;
44 }
45
46 int main() {
47     // Declaración y rellenar matriz 3x3 con valores predefinidos
48     int matriz[SIZE][SIZE] = {
49         {1, 2, 3},
50         {2, 5, 6},
51         {3, 6, 9}
52     };
53
54     int transpuesta[SIZE][SIZE];
55
56     // Calcular la transpuesta
57     calcularTranspuesta(matriz, transpuesta);
```

Terminal

```
Matriz original:
1 2 3
2 5 6
3 6 9
Matriz transpuesta:
1 2 3
```

[COMPILAR](#) [AYUDA CON LA TERMINAL](#)

ENUNCIADO CHAT RETROALIMENTACIÓN

Estado: [Enviado](#)

QuickCodeA: Soy tu inteligente que ayuda con ejercicios de programación.

Tú: ¿Qué error se muestra en la consola? ¿Cómo puedo solucionarlo?

QuickCodeA: El error que se muestra en la consola es un mensaje de advertencia sobre la sintaxis incorrecta del código. La línea 4 `return 0;` está mal escrita, ya que no tiene un semáforo (`return`) después de `return`. En C, el bloque principal del programa debe terminar con un semáforo. Para solucionarlo, debes agregar un semáforo después de la función `main`.

Tú: ¿Cómo defino una matriz?

QuickCodeA: Para definir una matriz en C, debes especificar el tamaño de la matriz y luego los valores del elemento que va a ocupar cada posición de la matriz. Por ejemplo:

```
int matriz[3][3];
```

 Esto define una matriz 3x3 con todos sus elementos como cero. Para asignar un valor específico a una posición, puedes hacerlo así:

```
matriz[0][0] = 1;
matriz[0][1] = 2;
```

 Recuerda que en C, los índices de matriz comienzan en 0.

Tú: Me das una pista para resolver el ejercicio.

QuickCodeA: Pistas para resolver el ejercicio: Asegúrate de agregar un semáforo después del bloque principal `main` en la línea 4. Debes declarar la matriz antes de utilizarla. Intenta colocar la declaración de la matriz justo después de la declaración de `main`. La función que calcula la transpuesta de la matriz debe tomar la matriz como parámetro y devolver su transpuesta. ¿Cómo podrías pasar la matriz a esa función? La función que muestra por pantalla los elementos del programa necesita leer las entradas desde el teclado, ¿cómo lo harías?

Escribe una pregunta o solicita ayuda

[PISTA IA \(BETA\)](#) [REALIZAR CONSULTA](#)

Figura 15. Vista del ejercicio después de ser enviado: el código no es editable, pero se puede volver a compilar y consultar la salida. (Rol de estudiante)

Desde la vista de profesor, el profesorado puede acceder al listado de entregas realizadas por el alumnado. En esta tabla se muestran los datos clave como el nombre del estudiante, la fecha y hora de la última modificación, el estado de la entrega (enviado/no enviado), y accesos directos para visualizar el contenido de la entrega o eliminarla si fuera necesario.

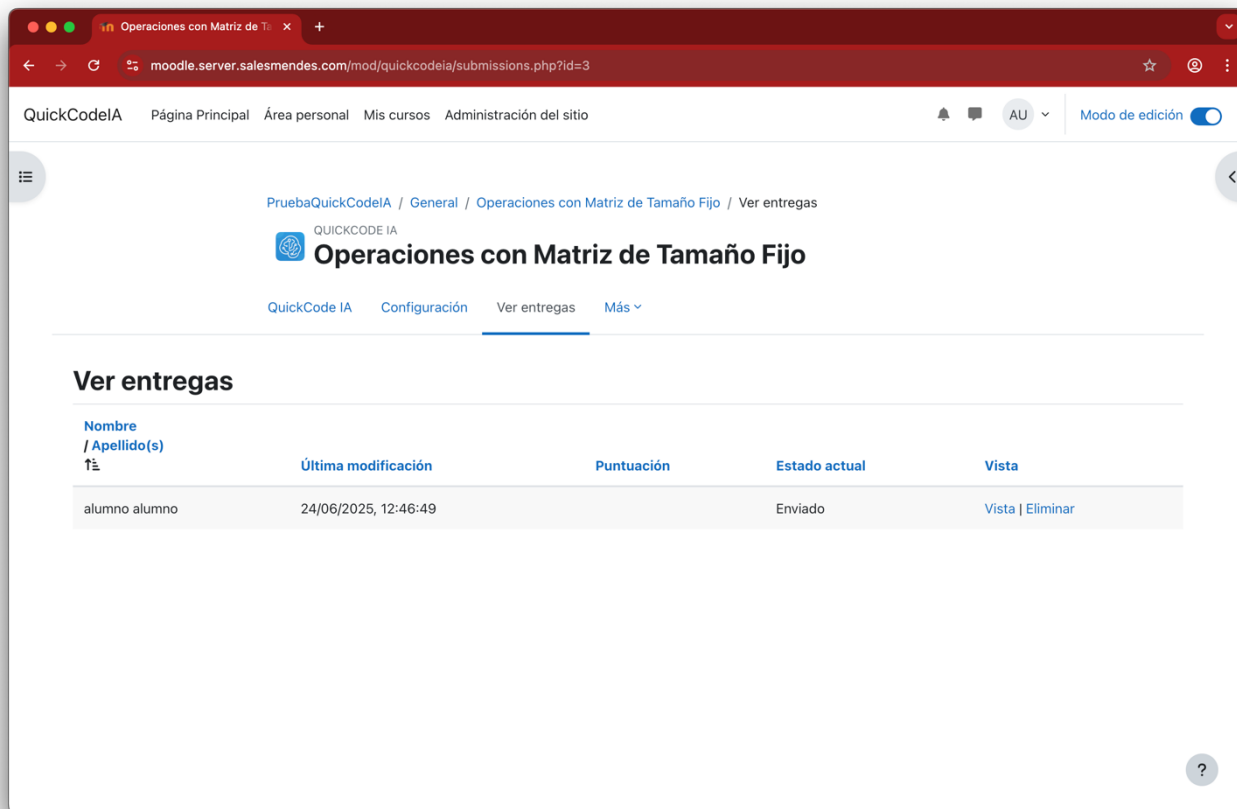


Figura 16. Vista del listado de entregas disponibles para el profesorado, incluyendo nombre del alumno, estado, y acciones. (Rol de profesor)

Al acceder al detalle de una entrega específica, el profesorado puede revisar el ejercicio tal y como fue presentado por el alumnado. Desde esta vista es posible volver a ejecutar el código entregado y analizar tanto el código fuente como los mensajes generados en la terminal o las interacciones con el asistente de IA. Esta funcionalidad permite realizar una evaluación precisa del trabajo realizado, sin necesidad de modificar el contenido original.

The screenshot displays the QuickCodeIA interface, which is divided into three main sections:

- Editor de Código:** A code editor showing C++ code for matrix operations. The code includes functions for calculating the transpose of a matrix, displaying a matrix, calculating the sum of the main diagonal, and checking if a matrix is symmetric. The main function initializes a 3x3 matrix and calculates its transpose.
- Terminal:** A terminal window showing the output of the program. It displays the original matrix and its transpose.


```

      Matriz original:
      1 2 3
      4 5 6
      7 8 9

      Matriz transpuesta:
      1 4 7
      2 5 8
      3 6 9
      
```
- Chat:** A chat interface with the AI assistant, QuickCodeIA. The chat history shows several interactions:
 - ENUNCIADO:** The user asks for help with a programming exercise.
 - CHAT:** The AI provides advice on semicolons and array syntax.
 - RETROALIMENTACIÓN:** The AI explains the error in the code, pointing to the missing semicolon in the array declaration and the incorrect loop range.
 - CHAT:** The user asks how to define a matrix.
 - RETROALIMENTACIÓN:** The AI provides instructions on how to declare and initialize a matrix in C++.
 - CHAT:** The user asks for a hint to solve the exercise.
 - RETROALIMENTACIÓN:** The AI provides a hint to add a semicolon after the main block and to declare the matrix before using it.

Figura 17. Vista del ejercicio enviado por un alumno, accesible desde la revisión de entregas. (Rol de profesor)

Desde la pestaña "Retroalimentación", es posible visualizar el código entregado junto con su resultado en la terminal y emitir una calificación acompañada de comentarios personalizados. Esta funcionalidad permite una evaluación cualitativa y cuantitativa centrada en el proceso de aprendizaje del alumnado.

The screenshot displays the QuickCodeLA interface. On the left, there is a code editor titled "Editor de Código" containing C++ code for matrix operations. Below the editor is a terminal window showing the output of the program. On the right, there is a navigation bar with tabs for "ENUNCIADO", "CHAT", and "RETROALIMENTACIÓN". The "RETROALIMENTACIÓN" tab is active, showing a section for "Nota por el profesor:" with a text input field, and a "Retroalimentación:" section with a larger text area for comments. A "GUARDAR FEEDBACK" button is located at the bottom of the feedback section.

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 #define SIZE 3
5
6 // Función para calcular la transpuesta de una matriz
7 void calcularTranspuesta(int matriz[SIZE][SIZE], int transpuesta[SIZE][SIZE]) {
8     for (int i = 0; i < SIZE; i++) {
9         for (int j = 0; j < SIZE; j++) {
10            transpuesta[j][i] = matriz[i][j];
11        }
12    }
13 }
14
15 // Función para mostrar una matriz
16 void mostrarMatriz(int matriz[SIZE][SIZE]) {
17     for (int i = 0; i < SIZE; i++) {
18         for (int j = 0; j < SIZE; j++) {
19             printf("%d ", matriz[i][j]);
20         }
21         printf("\n");
22     }
23 }
24
25 // Función para calcular la suma de la diagonal principal
26 int sumaDiagonal(int matriz[SIZE][SIZE]) {
27     int suma = 0;
28     for (int i = 0; i < SIZE; i++) {
29         suma += matriz[i][i];
30     }
31     return suma;
32 }
33
34 // Función para verificar si la matriz es simétrica
35 bool esSimetrica(int matriz[SIZE][SIZE], int transpuesta[SIZE][SIZE]) {
36     for (int i = 0; i < SIZE; i++) {
37         for (int j = 0; j < SIZE; j++) {
38             if (matriz[i][j] != transpuesta[j][i]) {
39                 return false;
40             }
41         }
42     }
43     return true;
44 }
45
46 int main() {
47     // Declarar y rellenar matriz 3x3 con valores predefinidos
48     int matriz[SIZE][SIZE] = {
49         {1, 2, 3},
50         {2, 5, 6},
51         {7, 8, 9}
52     };
53
54     int transpuesta[SIZE][SIZE];
55
56     // Calcular la transpuesta
57     calcularTranspuesta(matriz, transpuesta);
```

Terminal output:

```
Matriz original:
1 2 3
2 5 6
7 8 9

Matriz transpuesta:
1 2 7
2 5 8
3 6 9
```

Figura 18. Vista de evaluación docente con campo para nota y retroalimentación sobre la entrega del ejercicio. (Rol de profesor)

La plataforma Moodle permite al profesorado consultar de forma centralizada las calificaciones de todas las actividades evaluables del curso mediante el informe del calificador.

Gracias a la integración del *plugin* con el libro de calificaciones, las notas obtenidas en los ejercicios de programación aparecen reflejadas automáticamente en este informe, facilitando el seguimiento del rendimiento académico del alumnado.

The screenshot shows the Moodle gradebook interface for the activity 'PruebaQuickCodeIA'. The browser address bar indicates the URL 'moodle.server.salesmendes.com/grade/report/grader/index.php?id=2'. The page title is 'PruebaQuickCodeIA' and the user is logged in as 'AU'. A navigation menu includes 'Curso', 'Configuración', 'Participantes', 'Calificaciones', 'Informes', and 'Más'. A warning message states: 'Advertencia: ¡Borrado de actividad en curso! Algunas calificaciones están a punto de ser eliminadas.' Below this, there are search and filter options: 'Informe del calificador', 'Buscar usuarios', 'Filtrar por nombre', and 'Columnas colapsadas'. The main table displays student names, email addresses, and scores for the activity. The 'Operaciones con Matriz de Tamaño Fijo' column shows scores of 10,00 for the first student and empty boxes for the others. The 'Total del curso' column shows a score of 10,00 for the first student and empty boxes for the others. A 'Promedio general' row at the bottom shows a score of 10,00. The interface also includes a 'Mostrar' dropdown set to 20 and a 'Guardar cambios' button.

| | | PruebaQuickCodeIA | | |
|----------------------------------|----------------------------------|-------------------|---------------------------------------|-----------------|
| Nombre / Apellido(s) | Dirección de correo | | Operaciones con Matriz de Tamaño Fijo | Total del curso |
| aa alumno alumno | alumno@usal.es | | 10,00 | 10,00 |
| DC David Cruz García | david.cruz.garcia_alumno@usal.es | | | |
| Jd Juan Francisco de Paz Santana | fcofds_alumno@usal.es | | | |
| Promedio general | | | 10,00 | 10,00 |

Figura 19. Vista general del informe de calificador en Moodle, mostrando las notas asignadas a cada estudiante en el ejercicio práctico desarrollado. (Rol de profesor)

Una vez que el ejercicio ha sido corregido por el profesor, el alumno puede visualizar la nota obtenida junto con la retroalimentación correspondiente.

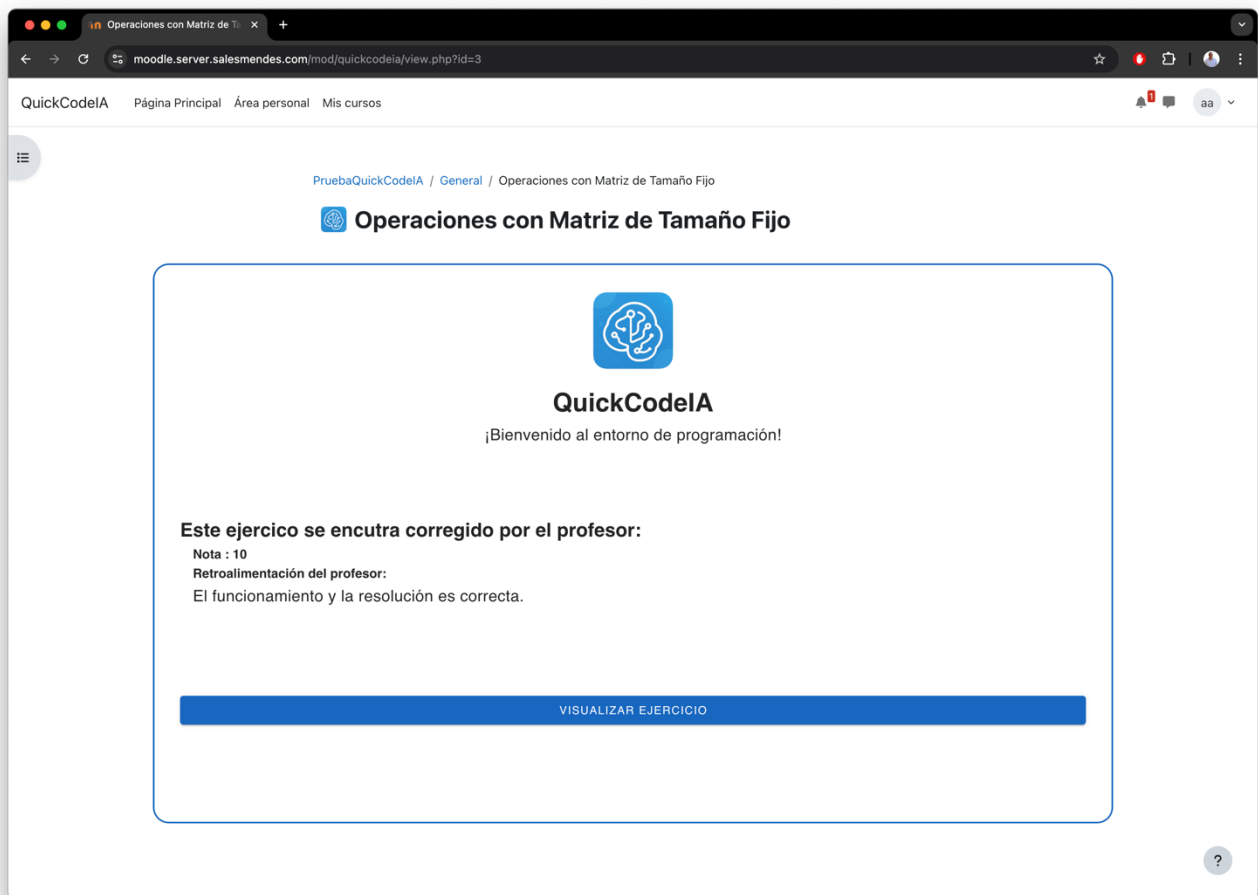


Figura 20. Vista del alumno tras la corrección del ejercicio. Se muestra la nota y la retroalimentación del profesor. Si se requiere, el ejercicio puede reabrirse para su mejora. (Rol de estudiante)

6. Evaluación de la plataforma

Para evaluar la plataforma QuickCodeIA desde la perspectiva del estudiante, se ha realizado una encuesta a los alumnos que han interactuado con la actividad. El objetivo ha sido valorar aspectos clave como la facilidad de uso, el soporte de IA integrado, la utilidad del editor con terminal, y la percepción general del entorno como herramienta de apoyo al aprendizaje de programación.

La encuesta estuvo estructurada de la siguiente manera:

Información general:

Nombre del curso:

Fecha:

Semestre/Periodo:

Usabilidad de la Plataforma:

¿Cómo calificaría la facilidad de uso del entorno de programación QuickCodeIA?

- Muy fácil
- Fácil
- Neutral
- Difícil
- Muy difícil

¿Encontró algún problema técnico al utilizar la plataforma?

- Sí (por favor, especifique)
- No

Interacción con el Chat IA y la Terminal:

¿El chat de ayuda con IA le fue útil para resolver dudas o errores en su código?

- Totalmente de acuerdo
- De acuerdo
- Neutral
- En desacuerdo
- Totalmente en desacuerdo

¿Considera útil la funcionalidad de solicitar pistas paso a paso?

- Muy útil
- Útil
- Poco útil
- Innecesaria

Experiencia de Aprendizaje:

¿La plataforma ha facilitado su aprendizaje en programación en cuestión?

- Totalmente de acuerdo
- De acuerdo
- Neutral
- En desacuerdo
- Totalmente en desacuerdo

¿Qué aspecto del entorno consideró más beneficioso para su aprendizaje?

(Respuesta abierta)

Evaluación de Funcionalidades:

¿Considera que el editor, la terminal y el chat integrados cubren sus necesidades básicas para trabajar los ejercicios?

- Superaron mis expectativas
- Cumplieron mis expectativas
- Fueron por debajo de mis expectativas

¿Qué herramientas o mejoras cree que podrían añadirse al entorno?

(Respuesta abierta)

Sugerencias y Comentarios:

Por favor, proporcione cualquier comentario adicional o sugerencia para mejorar el entorno QuickCodeIA.

(Respuesta abierta)

La recopilación y análisis de estas encuestas permite conocer la percepción de los usuarios finales y detectar áreas de mejora. Se valora tanto el componente cuantitativo (respuestas cerradas) como cualitativo (comentarios abiertos), lo que proporciona una visión holística de la experiencia.

Los resultados tras haber realizado las encuestas a 25 estudiantes se pueden ver en las siguientes tablas:

| <i>Facilidad de Uso</i> | <i>Número de Alumnos</i> |
|-------------------------|--------------------------|
| <i>Muy fácil</i> | 11 (44%) |
| <i>Fácil</i> | 10 (40%) |
| <i>Neutral</i> | 3 (12%) |
| <i>Difícil</i> | 1 (4%) |
| <i>Muy difícil</i> | 0 (0%) |

Tabla 1. Facilidad de uso de QuickCodeIA

| <i>Problemas Técnicos</i> | <i>Número de Alumnos</i> |
|---------------------------|--------------------------|
| <i>Sí (Especificar)</i> | 4 (16%) |
| <i>No</i> | 21 (84%) |

Tabla 2. Problemas técnicos encontrados

| <i>El chat con IA fue útil</i> | <i>Número de Alumnos</i> |
|---------------------------------|--------------------------|
| <i>Totalmente de acuerdo</i> | 9 (36%) |
| <i>De acuerdo</i> | 12 (48%) |
| <i>Neutral</i> | 3 (12%) |
| <i>En desacuerdo</i> | 1 (4%) |
| <i>Totalmente en desacuerdo</i> | 0 (0%) |

Tabla 3. Valoración del chat de ayuda

| <i>Cumplimiento de expectativas</i> | <i>Número de Alumnos</i> |
|---------------------------------------|--------------------------|
| <i>Superaron mis expectativas</i> | 6 (24%) |
| <i>Cumplieron mis expectativas</i> | 17 (68%) |
| <i>Por debajo de mis expectativas</i> | 2 (8%) |

Tabla 4. Evaluación general del entorno QuickCodeIA

Comentarios destacados de los estudiantes:

- “La posibilidad de interactuar con el código, la consola y un chat de ayuda todo en uno es muy cómodo.”
- “El sistema de pistas me ayudó a resolver un ejercicio que no entendía al principio.”
- “Sería interesante poder guardar versiones del código a lo largo del ejercicio.”

Problemas técnicos reportados:

- “Hubo una ocasión en la que el terminal no mostró nada, pero al recargar se solucionó.”
- “Al compilar a veces tarda un poco, pero funciona bien.”

Sugerencias de mejora:

- “Agregar una función para exportar el código final en PDF.”
- “Un modo oscuro sería útil para trabajar durante más tiempo.”

7. Repositorio del proyecto

Tanto el *plugin* desarrollado para Moodle como los servicios externos que conforman el entorno de ejecución y asistencia por inteligencia artificial han sido publicados en abierto en el repositorio de GitHub del proyecto, disponible en: <https://github.com/salesmendesandre/QuickCodeIA>. Esta publicación tiene como objetivo fomentar la transparencia, facilitar la replicabilidad del sistema en otros contextos educativos y promover su mejora colaborativa por parte de la comunidad docente y técnica.

8. Resultados y Conclusiones

La implementación de la plataforma QuickCodeIA dentro del entorno Moodle ha demostrado ser una solución eficaz para el desarrollo de ejercicios prácticos de programación. A lo largo de la experiencia piloto, se ha observado un alto nivel de participación por parte del alumnado, que no solo completó las tareas asignadas, sino que también interactuó de forma activa con las herramientas de ayuda integradas, como la terminal y el chat con inteligencia artificial.

Los resultados de las encuestas muestran que la mayoría de los estudiantes consideró la plataforma fácil de usar y útil para el aprendizaje autónomo. El sistema fue valorado positivamente tanto por su estabilidad técnica como por la claridad de su interfaz, lo cual facilitó la navegación y el trabajo con el editor de código. Destacó especialmente la utilidad del asistente inteligente, que permitió a los alumnos recibir ayuda contextualizada en tiempo real, identificar errores comunes y obtener pistas graduales sin necesidad de abandonar el entorno.

Desde el punto de vista del profesorado, el panel de gestión resultó funcional para supervisar las entregas, revisar el código y proporcionar retroalimentación.

En conjunto, se concluye que QuickCodeIA contribuye a mejorar la experiencia de aprendizaje en programación, al combinar un entorno técnico controlado con mecanismos de asistencia pedagógica. Su uso promueve la autonomía, facilita el seguimiento del progreso y permite integrar la evaluación dentro del propio proceso de resolución de problemas.

9. Líneas futuras

Como líneas futuras, el proyecto plantea diversas mejoras orientadas a ampliar su funcionalidad y aplicabilidad dentro del aula. Entre las propuestas destacadas se encuentra la incorporación de ejercicios para otros lenguajes de programación, permitiendo así su adaptación a distintos niveles y asignaturas.

También se contempla el desarrollo de estadísticas avanzadas de rendimiento, con informes automáticos que ayuden al profesorado a identificar patrones de dificultad, progresos individuales y necesidades específicas de los estudiantes. Asimismo, se pretende enriquecer el asistente con IA mediante el entrenamiento de modelos más especializados que mejoren su capacidad de dar pistas pedagógicas y detectar errores lógicos comunes.

10. Plan de difusión

Se tiene previsto enviar los resultados del proyecto de innovación docente a congresos especializados en docencia e innovación educativa, incluyendo el *I Encuentro de Buenas Prácticas Docentes con Inteligencia Artificial* y futuras ediciones del congreso *NODOS DEL CONOCIMIENTO*, dentro del área temática de *Innovación Docente y Transferencia del Conocimiento*, especialmente en el nodo de *Innovación Docente en Áreas STEM*.

Agradecimientos

Todos los miembros del equipo de trabajo de este PID quieren agradecer al alumnado que ha participado en la realización del proyecto.

Referencias

- Carbonell Bernal, N., & Hernández Prados, M. Á. (2024). *Impacto de los sistemas de tutoría inteligente: Una revisión sistemática*. *Edutec. Revista Electrónica de Tecnología Educativa*, (89), 121-134. <https://doi.org/10.21556/edutec.2024.89.3025>
- González-Carrillo, C. D., Restrepo-Calle, F., Ramírez-Echeverry, J. J., & González, F. A. (2021). *Automatic grading tool for Jupyter notebooks in artificial intelligence courses*. *Sustainability*, 13(21), 12050. <https://doi.org/10.3390/su132112050>
- Kasneci, E., Sessler, K., Küchemann, S., Bannert, M., et al. (2023). *ChatGPT for good? On opportunities and challenges of large language models for education*. *Learning and Individual Differences*, 103, 102274. <https://doi.org/10.1016/j.lindif.2023.102274>
- Lobb, R., & Harlow, J. (2016). *CodeRunner: A tool for assessing computer programming skills*. *ACM Inroads*, 7(1), 47-51. <https://doi.org/10.1145/2810041>
- Wermelinger, M. (2023). *Using GitHub Copilot to solve simple programming problems*. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education (SIGCSE'23)* (pp. 172-178). ACM.
- Chi, M. T. H., Bassok, M., Lewis, M. W., Reimann, P., & Glaser, R. (1989). *Self-explanations: How students study and use examples in learning to solve problems*. *Cognitive Science*, 13(2), 145–182. https://doi.org/10.1207/s15516709cog1302_1