Evaluación de un sistema de control de versiones con una plataforma web de capacitación inicial para reducir la concentración del trabajo en proyectos universitarios de programación

Eder Omar Zúñiga Zavala – 2430206

Información del Proyecto

Dirigido al sector terciario, educación.

Contents

1 Problema **4**

2 Objetivo General **4**

3 Objetivos Específicos **4**

4 Pregunta de Investigación **4**

5 Justificación **4**

6 Viabilidad **5**

7 Hipótesis de Investigación **5**

8 Marco Teórico **5**
8.1 El Aprendizaje Colaborativo 5
8.1.1 El fenómeno de la reducción de esfuerzo (Holgazanería Social) . . . 5
8.1.2 Rendimiento académico en Fundamentos de Programación 5
8.1.3 Contexto de la Innovación Educativa Universitaria 6
8.2 Sistemas de Control de Versiones en Educación 6
8.2.1 Git como plataforma de gestión académica 6
8.2.2 Impacto en la colaboración y el aprendizaje 6
8.2.3 Identificación de disparidades en la contribución 6
8.3 Evaluación Mediante Métricas de Software 6
8.3.1 Correlación entre métricas y desempeño académico 6
8.3.2 Evaluación integral de contribuciones individuales 6
8.3.3 Pensamiento computacional y resolución de problemas 7
8.4 Innovación, Gamificación y Nuevos Entornos 7
8.4.1 Autocorrección y estrategias lúdicas en la programación 7
8.4.2 Formatos de capacitación intensiva (Bootcamp) 7
8.4.3 Nuevos entornos de gestión y colaboración 7

9 Metodología y Diseño de Investigación **7**
9.1 Tipo y Alcance del Diseño 7
9.2 Variables de Estudio 7
9.3 Instrumentos de Recolección 8
9.4 Procedimiento Sugerido 8

10 Sistema Propuesto: Malleus Codeficarum **8**
10.1 Descripción General 8
10.2 Hardware Utilizado 8
10.2.1 Servidor — Infraestructura en la Nube 8
10.2.2 Equipo de Desarrollo 9
10.2.3 Requisitos del Cliente (Usuario Final) 9
10.3 Software Utilizado 9
10.3.1 Tecnologías del Servidor 9
10.3.2 Tecnologías del Cliente 9
10.3.3 Herramientas de Desarrollo 9
10.4 Arquitectura del Sistema 10
10.5 Funcionalidades Principales 10

10.6 Diagramas UML 11
10.6.1 Diagrama de Casos de Uso 11
10.6.2 Diagrama de Clases 11
10.6.3 Diagrama de Secuencia 12
10.6.4 Diagrama de Actividades 13
10.7 Diagrama de Flujo — Simulador de Terminal Git 14
10.8 Pseudocódigo General del Sistema 14
10.9 Interfaces de la Plataforma 15

11 Resultados **17**
11.1 Resultados del Pre-test y Post-test 17
11.2 Métricas de Contribución en Repositorios Git 18
11.3 Rendimiento en la Plataforma 18

12 Discusión **18**

13 Conclusiones **19**

1 Problema

En los proyectos universitarios de programación en equipo, es frecuente que las tareas de desarrollo se concentren en uno o pocos integrantes. Este desequilibrio limita las oportunidades de aprendizaje práctico del resto del grupo y dificulta una evaluación objetiva del desempeño individual.

Aunque existen sistemas de control de versiones que permiten registrar y transparentar las contribuciones, su adopción suele ser desigual debido a la falta de una capacitación inicial estructurada. Esta situación impide una distribución equitativa del trabajo y reduce la efectividad del aprendizaje colaborativo.

2 Objetivo General

Evaluar el impacto de la implementación de un sistema de control de versiones, acompañado de un entorno de aprendizaje interactivo, en la distribución del trabajo y la transparencia de las contribuciones dentro de proyectos universitarios de programación en equipo.

3 Objetivos Específicos

1. Desarrollar un bootcamp interactivo que estandarice las competencias básicas en herramientas de control de versiones, estableciendo una base técnica común para todos los estudiantes.

2. Integrar el sistema de control de versiones como la herramienta principal de gestión del aprendizaje para el seguimiento, entrega y organización de los proyectos de programación.

3. Analizar el impacto en la participación grupal a través de métricas de contribución objetivas, verificando si la capacitación y el uso de la herramienta reducen la concentración de tareas en pocos integrantes.

4 Pregunta de Investigación

Ła implementación de un sistema de control de versiones, acompañado de un protocolo de capacitación inicial, reduce la concentración del trabajo en uno o pocos integrantes en proyectos universitarios de programación en equipo?

5 Justificación

La implementación de un sistema de control de versiones permite centralizar el desarrollo del código y registrar de manera objetiva las contribuciones individuales, favoreciendo la transparencia y una distribución más equitativa del trabajo.

Asimismo, la creación de un entorno de aprendizaje interactivo basado en una plataforma web se justifica como un medio para garantizar que todos los estudiantes adquieran competencias básicas en el uso de estas herramientas antes de integrarse al trabajo en equipo. Esto reduce las diferencias de conocimiento previo y mejora la participación activa de todos los integrantes.

La combinación de ambas tecnologías no solo mejora los procesos de enseñanza–aprendizaje, sino que también replica prácticas ampliamente utilizadas en el ámbito laboral del desarrollo de software, facilitando la transición de los estudiantes a entornos profesionales y fortaleciendo competencias técnicas relevantes para su futura inserción laboral.

6 Viabilidad

El proyecto es viable en el contexto universitario, ya que se apoya en tecnologías accesibles y de uso común en el área de la informática. El desarrollo de una plataforma web con fines formativos no requiere infraestructuras complejas y puede realizarse de manera gradual conforme avanza la investigación.

El uso de un sistema de control de versiones resulta igualmente factible, al tratarse de una herramienta ampliamente conocida y utilizada tanto en entornos educativos como profesionales.

Asimismo, la investigación es viable desde el punto de vista académico, al contar con una población adecuada para la aplicación del entorno propuesto y la evaluación de la distribución del trabajo en proyectos colaborativos.

7 Hipótesis de Investigación

La implementación de un sistema de control de versiones, acompañado de una capacitación inicial mediante un entorno de aprendizaje interactivo, reduce la concentración del trabajo en uno o pocos integrantes en proyectos universitarios de programación en equipo.

8 Marco Teórico

8.1 El Aprendizaje Colaborativo

8.1.1 El fenómeno de la reducción de esfuerzo (Holgazanería Social)

En el trabajo grupal surge con frecuencia la tendencia de algunos integrantes a disminuir su rendimiento individual cuando forman parte de un colectivo. Este comportamiento se denomina holgazanería social o fenómeno del "polizón". [1]

Impacto del aprendizaje grupal: La evidencia indica que cuando un equipo logra un progreso real en sus conocimientos compartidos, la tendencia a reducir el esfuerzo disminuye. El aprendizaje conjunto actúa como un factor motivador que mantiene el compromiso de los miembros.

Evolución del compromiso: Aunque el esfuerzo inicial depende de la disposición personal de cada estudiante, a largo plazo es la dinámica y el ritmo de trabajo del equipo lo que previene la falta de participación.

8.1.2 Rendimiento académico en Fundamentos de Programación

La enseñanza de la programación representa un reto complejo en la educación superior debido a su alta exigencia técnica. [2]

Homogeneidad de conocimientos: Establecer una base sólida de competencias básicas mediante entornos interactivos permite que todos los estudiantes inicien con capacidades similares, facilitando una participación equilibrada en proyectos posteriores.

8.1.3 Contexto de la Innovación Educativa Universitaria

La innovación en el nivel superior se enfoca en desplazar el protagonismo hacia el estudiante, transformando el aula en un espacio de colaboración activa. [3]

Nuevos formatos de enseñanza: La integración de recursos digitales y plataformas interactivas no solo moderniza el curso, sino que ayuda a mantener el interés del estudiante.

8.2 Sistemas de Control de Versiones en Educación

8.2.1 Git como plataforma de gestión académica

El uso de Git permite sustituir los sistemas de gestión de aprendizaje por herramientas reales de desarrollo de software. [4]

Seguimiento del progreso: A diferencia de las plataformas estáticas, esta herramienta registra cada avance, permitiendo al docente evaluar el proceso continuo de construcción del código y no solo el archivo final.

Estandarización profesional: El estudiante se familiariza con el flujo de trabajo técnico vigente, integrando la entrega de trabajos con la práctica profesional.

8.2.2 Impacto en la colaboración y el aprendizaje

GitHub mejora la organización de los equipos de programación, especialmente en aquellos con poca experiencia en trabajo conjunto. [5]

Competencias colaborativas: Obliga al uso de protocolos como solicitudes de integración y resolución de conflictos, transformando la tarea en un esfuerzo coordinado.

8.2.3 Identificación de disparidades en la contribución

La actividad en los repositorios permite medir de forma objetiva el nivel de participación individual dentro de un proyecto. [6]

Detección de perfiles: El análisis de métricas (como la frecuencia de envíos de código) ayuda a identificar a los integrantes que no contribuyen equitativamente o que trabajan de forma aislada.

8.3 Evaluación Mediante Métricas de Software

8.3.1 Correlación entre métricas y desempeño académico

El uso de datos extraídos de plataformas de desarrollo permite establecer una relación directa entre el esfuerzo técnico y la calificación final. [7]

Indicadores de actividad: Métricas como el número de commits y las líneas de código modificadas son predictores estadísticos del rendimiento. Existe una correlación positiva: a mayor actividad constante en el repositorio, mejor es el desempeño académico del estudiante.

Consistencia en el trabajo: No solo importa la cantidad de código, sino la frecuencia del trabajo. Los estudiantes con mejores resultados mantienen una participación regular.

8.3.2 Evaluación integral de contribuciones individuales

Para obtener una medición objetiva del trabajo en equipo, no basta con analizar el código; es necesario cruzar distintas fuentes de datos. [8]

8.3.3 Pensamiento computacional y resolución de problemas

La evaluación objetiva también debe considerar cómo el estudiante desarrolla su capacidad lógica desde el inicio de la carrera. [9]

8.4 Innovación, Gamificación y Nuevos Entornos

8.4.1 Autocorrección y estrategias lúdicas en la programación

Los sistemas de autocorrección reducen la dependencia del docente y permiten una práctica más intensiva. [10] El uso de dinámicas lúdicas ayuda a estructurar el pensamiento lógico y la elaboración de algoritmos. [11]

8.4.2 Formatos de capacitación intensiva (Bootcamp)

El diseño de *bootcamps* o entornos de aprendizaje intensivos permite estandarizar las competencias básicas en un periodo corto, evitando que la brecha de conocimientos genere desigualdad en el esfuerzo grupal. [12]

8.4.3 Nuevos entornos de gestión y colaboración

La evolución de la enseñanza universitaria exige el uso de plataformas que permitan una interacción más profunda y una gestión del aprendizaje basada en datos. [13–15]

9 Metodología y Diseño de Investigación

9.1 Tipo y Alcance del Diseño

La investigación se define como **Cuasi-experimental** con un alcance **Explicativo** y un enfoque **Mixto**. Debido a la imposibilidad de asignar alumnos al azar, se trabajará con grupos universitarios intactos para comparar el impacto de la intervención contra una metodología tradicional.

Componente	**Descripción**
Tipo de Diseño	Cuasi-experimental (Grupo Control vs. Grupo Experimental)
Enfoque	Mixto: Métricas de software (Cuantitativo) y percepción del alumno (Cualitativo)
Población	Estudiantes de carreras de informática/computación en proyectos de equipo

Table 1: Componentes del diseño de investigación

9.2 Variables de Estudio

- **Variable Independiente:** Implementación del bootcamp interactivo *Malleus Codeficarum* y uso obligatorio de Git.
- **Variable Dependiente:** Distribución equitativa del trabajo (reducción de la concentración de tareas en pocos integrantes).

9.3 Instrumentos de Recolección

1. **Minería de Datos (Git):** Extracción automatizada de commits, líneas de código (LOC) y frecuencia de actividad. Se utilizará el Coeficiente de Gini para medir la desigualdad en las contribuciones.

2. **Encuestas Likert:** Aplicadas al inicio (pre-test) y al final (post-test) para medir competencias técnicas y percepción de holgazanería social.

3. **Gráficos de Red:** Análisis visual de la interacción en el repositorio para identificar perfiles de participación.

9.4 Procedimiento Sugerido

1. **Fase 1 (Diagnóstico):** Aplicación de pre-test para identificar conocimientos previos de Git y lógica de programación.

2. **Fase 2 (Intervención):** Capacitación intensiva mediante el bootcamp interactivo para estandarizar competencias básicas.

3. **Fase 3 (Desarrollo):** Ejecución del proyecto de programación con monitoreo continuo del flujo de trabajo en los repositorios.

4. **Fase 4 (Evaluación):** Comparación estadística de métricas de contribución entre el grupo experimental y el de control.

10 Sistema Propuesto: Malleus Codeficarum

10.1 Descripción General

Malleus Codeficarum es una plataforma web interactiva diseñada como un bootcamp para el aprendizaje de Git desde nivel básico. El sistema guía al usuario mediante lecciones teóricas, prácticas con simulador de terminal y evaluaciones automatizadas, simulando escenarios reales de uso de Git.

10.2 Hardware Utilizado

10.2.1 Servidor — Infraestructura en la Nube

Componente	**Especificación**
Plataforma	Heroku (PaaS – Platform as a Service)
Tipo de instancia	Dyno Basic
RAM	512 MB
Almacenamiento	Efímero (slug de la aplicación)
Red	HTTPS / SSL automático

Table 2: Especificaciones del servidor en la nube

10.2.2 Equipo de Desarrollo

Componente	Especificación
Procesador	Intel Core i5 (8ª gen. o superior)
RAM	8 GB DDR4
Almacenamiento	256 GB SSD
Sistema Operativo	Windows 10/11 o Linux Ubuntu 20.04+

Table 3: Equipo de desarrollo utilizado

10.2.3 Requisitos del Cliente (Usuario Final)

- Navegador web moderno (Chrome 90+, Firefox 88+, Edge 90+)
- Mínimo 2 GB de RAM
- Conexión a internet (mínimo 1 Mbps)

10.3 Software Utilizado

10.3.1 Tecnologías del Servidor

Tecnología	Versión	Propósito
PHP	8.x	Lógica del servidor, menú dinámico
Apache	2.4	Servidor web HTTP
Linux	Ubuntu	Sistema operativo del servidor (Heroku)
Git	2.x	Control de versiones y despliegue

Table 4: Software del servidor

10.3.2 Tecnologías del Cliente

Tecnología	Versión	Propósito
HTML5	5	Estructura del contenido
CSS3	3	Diseño y estilos visuales
JavaScript	ES6+	Interactividad y simulador de terminal
Bootstrap	5.3	Diseño responsivo y componentes UI

Table 5: Software del cliente (frontend)

10.3.3 Herramientas de Desarrollo

- **Visual Studio Code** — Editor de código principal
- **Git + GitHub** — Control de versiones y repositorio remoto

- **Heroku CLI** — Despliegue de la aplicación
- **Chrome DevTools** — Depuración y pruebas del frontend

10.4 Arquitectura del Sistema

El sistema sigue un modelo **cliente–servidor** de tres capas: el navegador del usuario (cliente), el servidor Apache con PHP (lógica de negocio) y el sistema de archivos de lecciones (datos). La comunicación se realiza únicamente mediante HTTP/HTTPS, sin base de datos relacional, ya que el contenido está organizado en carpetas dentro del proyecto.

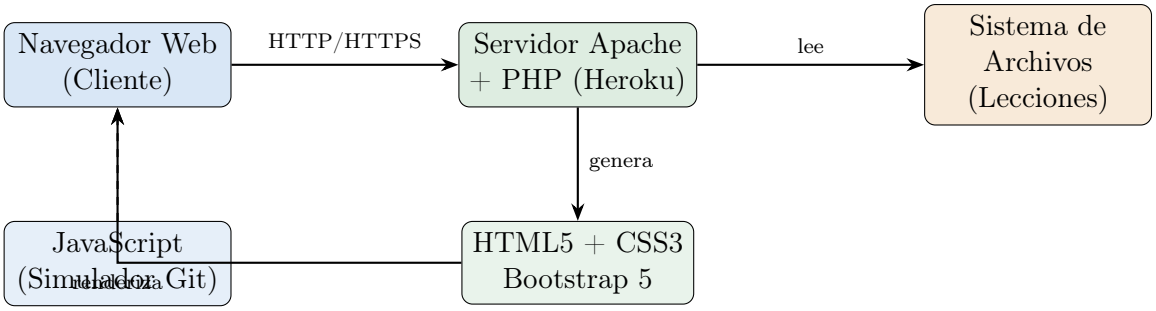

Figure 1: Arquitectura cliente–servidor de Malleus Codeficarum

10.5 Funcionalidades Principales

- **Menú dinámico:** Generado automáticamente por PHP escaneando la estructura de carpetas del proyecto.
- **Lecciones teóricas:** Contenido estructurado con ejemplos, tablas y cuestionarios al final de cada tema.
- **Simulador de terminal:** Entorno interactivo donde el usuario ejecuta comandos Git reales (init, add, commit, branch, merge, push, pull) con validación paso a paso.
- **Cuestionarios interactivos:** Evaluaciones de opción múltiple con retroalimentación inmediata.
- **Evaluación final:** Examen de 15 preguntas para validar el aprendizaje del módulo completo.

10.6 Diagramas UML

10.6.1 Diagrama de Casos de Uso

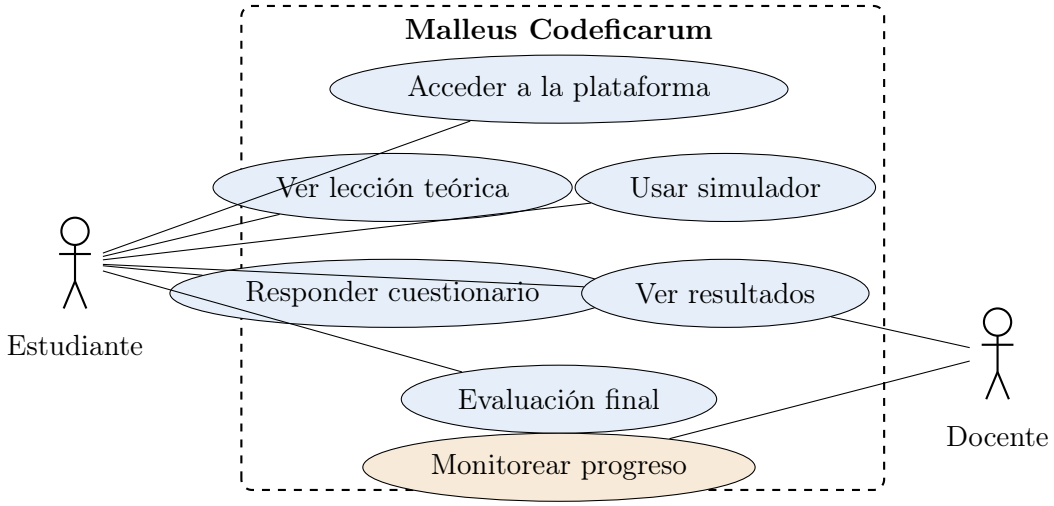

Figure 2: Diagrama de Casos de Uso del sistema Malleus Codeficarum

10.6.2 Diagrama de Clases

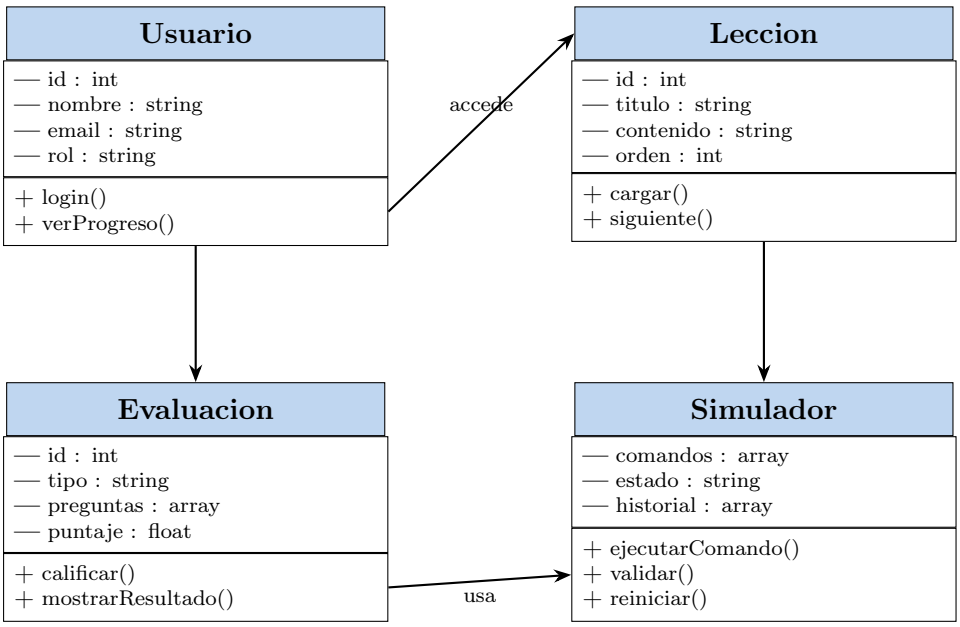

Figure 3: Diagrama de Clases del sistema

10.6.3 Diagrama de Secuencia

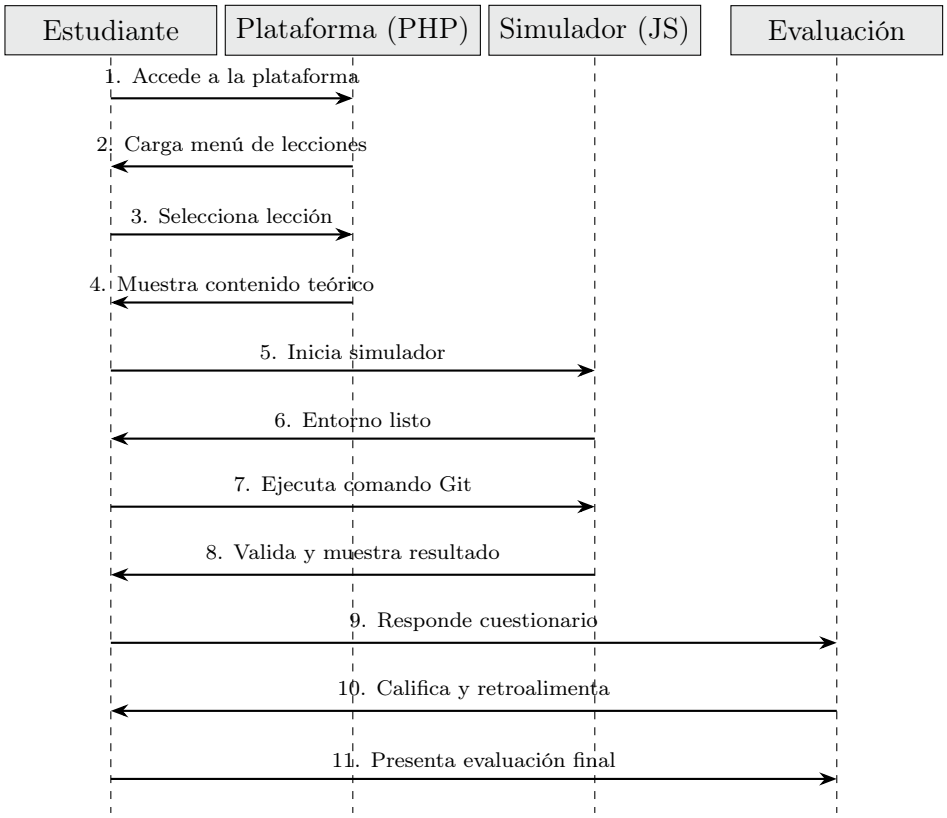

Figure 4: Diagrama de Secuencia — Flujo típico de uso de la plataforma

10.6.4 Diagrama de Actividades

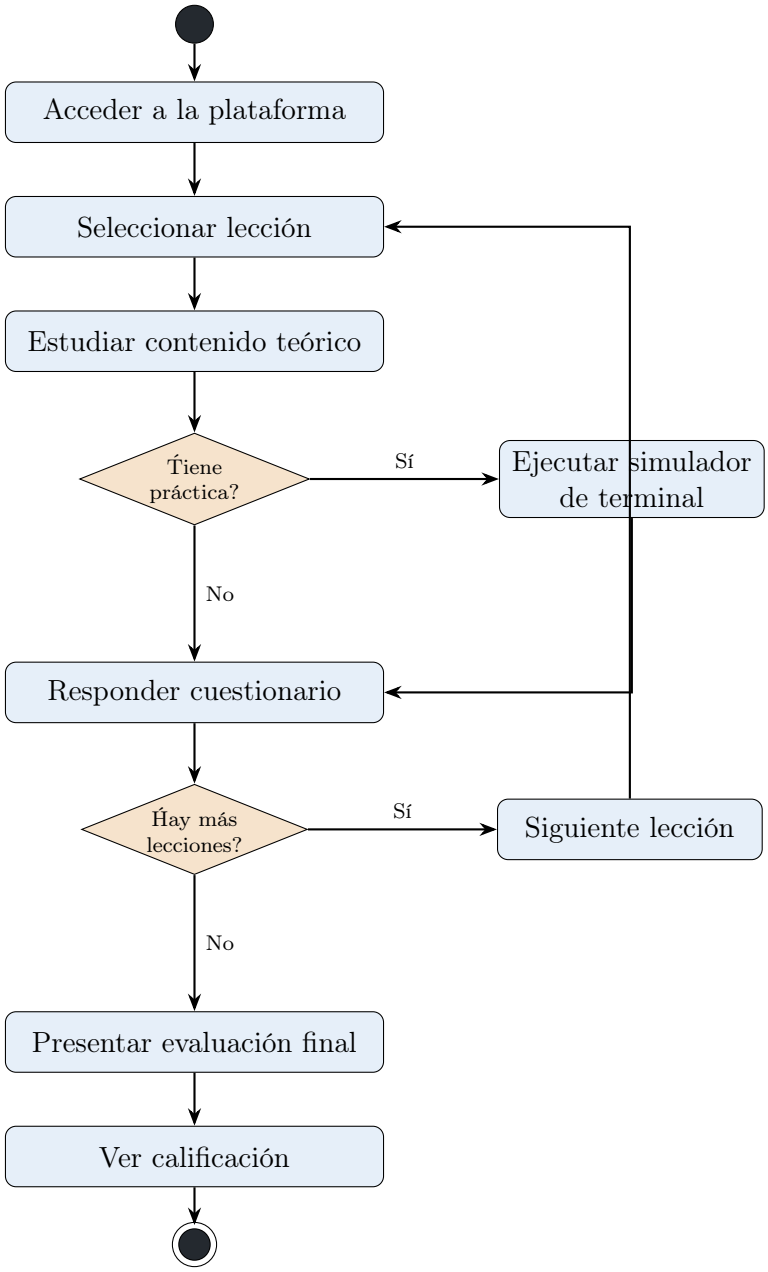

Figure 5: Diagrama de Actividades — Flujo general del sistema

10.7 Diagrama de Flujo — Simulador de Terminal Git

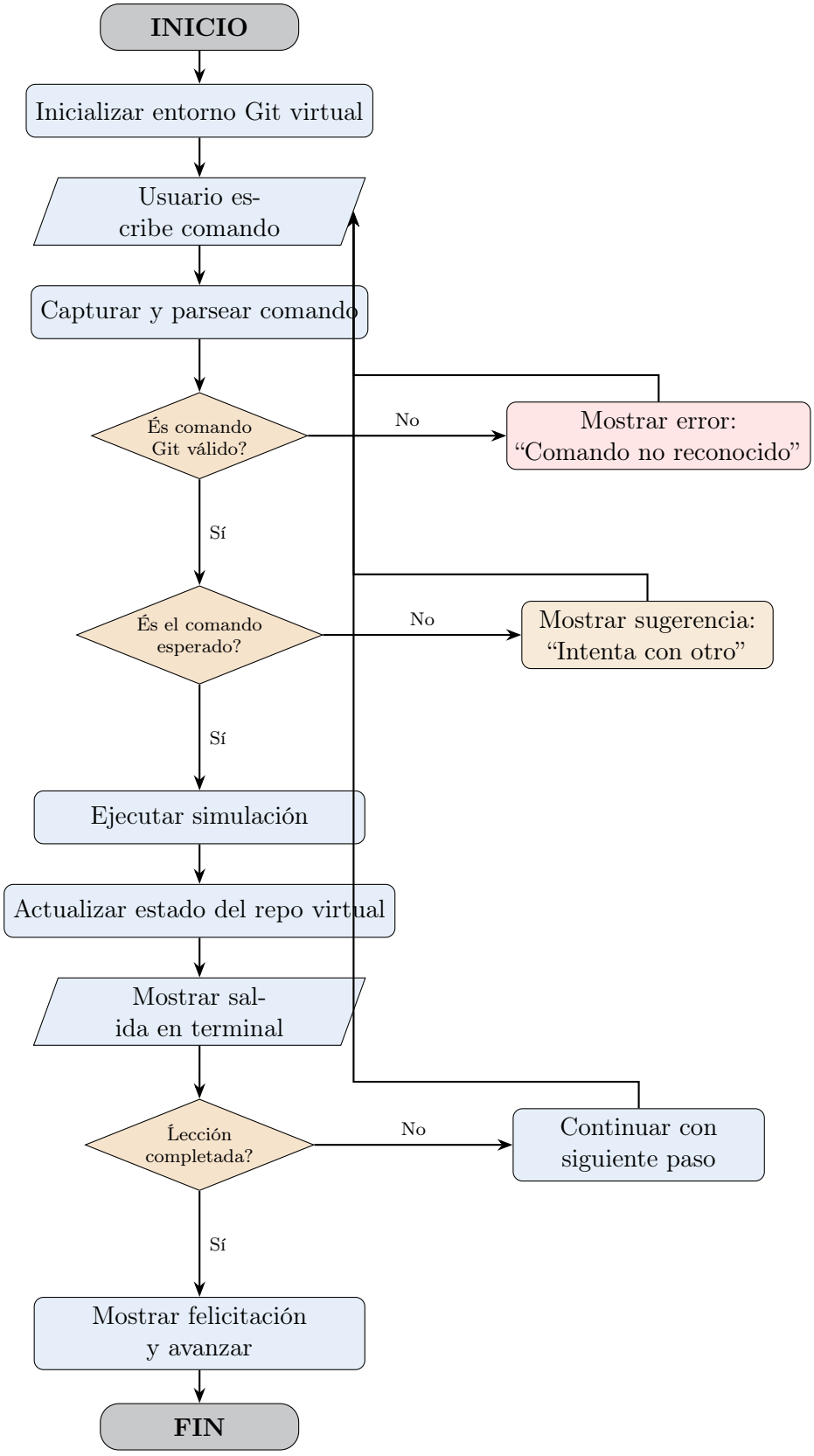

Figure 6: Diagrama de Flujo del Simulador de Terminal Git

10.8 Pseudocódigo General del Sistema

Pseudocódigo: Malleus Codeficarum — Flujo Principal

```
INICIO
  usuario accede a la plataforma
  PHP escanea carpeta de lecciones
  MOSTRAR menu_dinamico(lecciones disponibles)

  MIENTRAS usuario_activo = true HACER

    leccion_sel <- usuario selecciona leccion
    MOSTRAR contenido_teorico(leccion_sel)

    SI leccion_sel.tiene_simulador ENTONCES
      INICIAR simulador_terminal()
      MIENTRAS NOT completado HACER
        cmd <- leer_comando_usuario()
        SI es_valido(cmd) Y es_esperado(cmd) ENTONCES
          resultado <- ejecutar_git_virtual(cmd)
          actualizar_estado_repo()
          MOSTRAR resultado
          SI paso_completo ENTONCES avanzar_paso()
        SINO
          MOSTRAR mensaje_error_o_sugerencia()
        FIN SI
      FIN MIENTRAS
    FIN SI

    SI leccion_sel.tiene_cuestionario ENTONCES
      respuestas  <- usuario_responde_cuestionario()
      puntaje     <- calcular_puntaje(respuestas)
      MOSTRAR retroalimentacion_inmediata(puntaje)
    FIN SI

    SI es_ultima_leccion ENTONCES
      INICIAR evaluacion_final(15 preguntas)
      calificacion <- calificar(respuestas_finales)
      MOSTRAR calificacion
      registrar_progreso(usuario, calificacion)
    FIN SI

  FIN MIENTRAS
FIN
```

10.9 Interfaces de la Plataforma

A continuación se presentan capturas de pantalla reales de *Malleus Codeficarum* que ilustran las principales secciones de la plataforma.

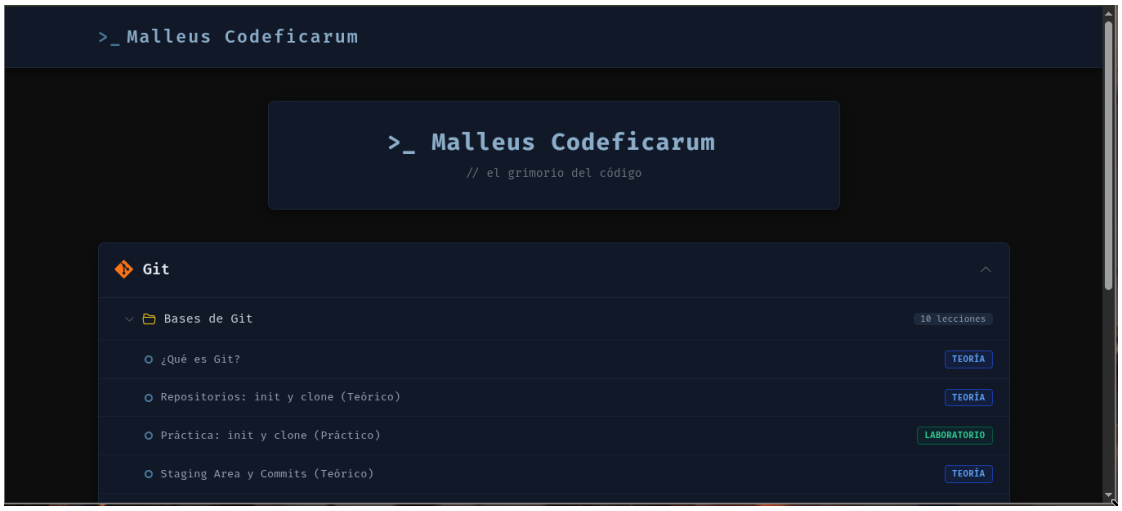

Figure 7: Página de inicio de Malleus Codeficarum

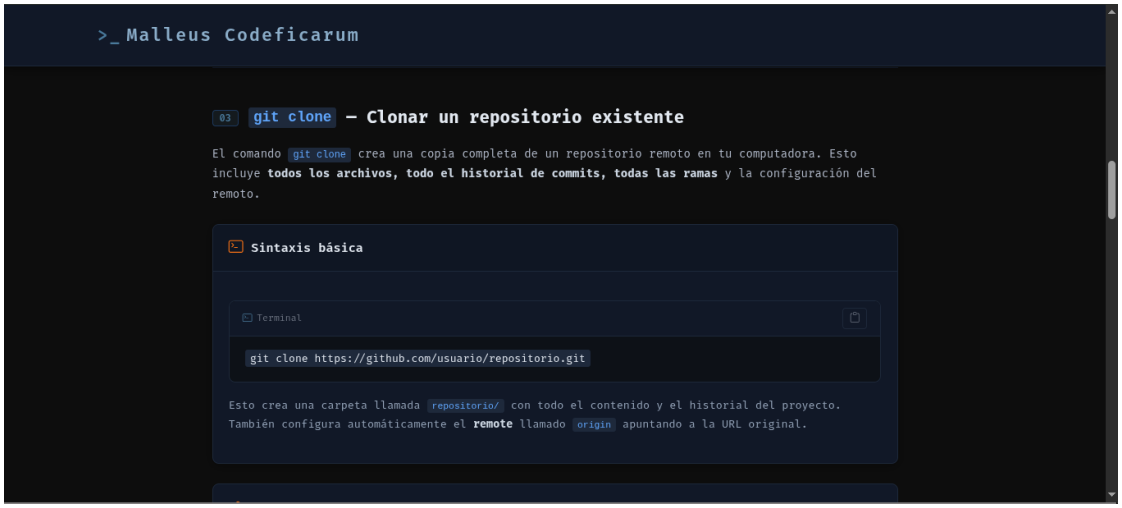

Figure 8: Vista de lección teórica

Figure 9: Cuestionario interactivo con retroalimentación inmediata

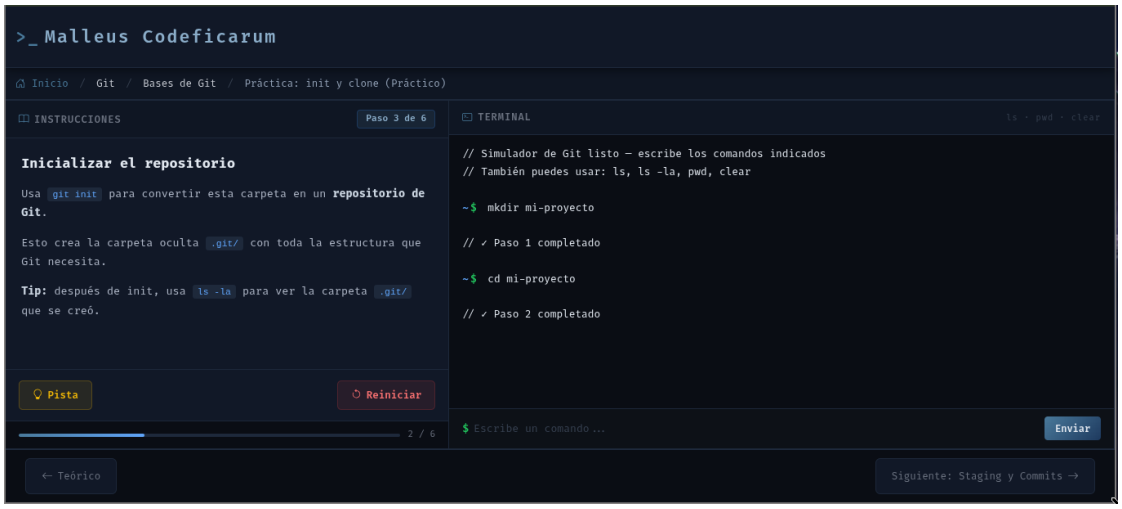

Figure 10: Simulador de terminal Git

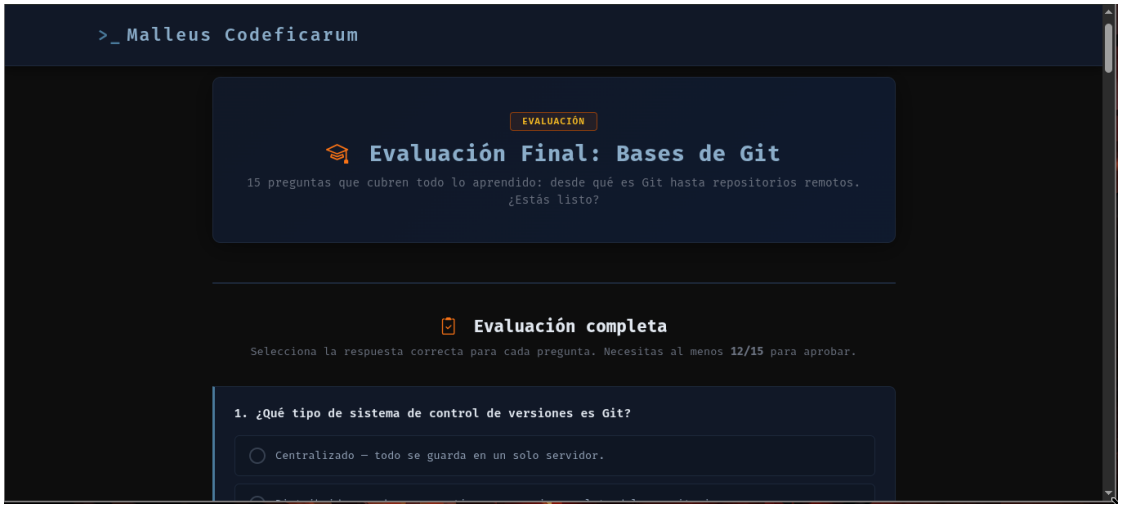

Figure 11: Evaluación final del módulo

11 Resultados

Nota metodológica: Los datos presentados en esta sección son **simulados con fines ilustrativos**, con el objetivo de demostrar el tipo de análisis y métricas que se aplicarán durante la fase de implementación real del estudio. El proyecto se encuentra actualmente en fase de diseño y los valores mostrados representan proyecciones esperadas fundamentadas en la literatura revisada en el Marco Teórico.

11.1 Resultados del Pre-test y Post-test

Se aplicó un cuestionario Likert de 15 ítems antes (pre-test) y después (post-test) de la intervención a dos grupos: experimental (con acceso a la plataforma y Git obligatorio) y control (metodología tradicional).

Indicador	Ctrl. Pre	Ctrl. Post	Exp. Pre	Exp. Post
Conocimiento de Git (0–10)	2.1	2.4	2.0	7.8
Carga equitativa percibida (1–5)	2.3	2.5	2.4	4.1
Confianza para contribuir (1–5)	2.7	2.9	2.6	4.3
Holgazanería social percibida (1–5)	3.8	3.6	3.9	2.1

Table 6: Comparación de resultados pre-test y post-test entre grupos *(datos simulados ilustrativos)*

11.2 Métricas de Contribución en Repositorios Git

Se analizó la actividad de los repositorios de ambos grupos durante el desarrollo del proyecto usando el Coeficiente de Gini como indicador de desigualdad en la distribución de contribuciones (0 = igualdad total, 1 = máxima desigualdad).

Métrica	Grupo Control	Grupo Experimental
Commits totales (promedio por equipo)	18.3	54.7
Coeficiente de Gini	0.71	0.29
Integrantes con al menos 1 commit (de 4)	1.8	3.7
LOC promedio por integrante (máx / mín)	312 / 48	280 / 265

Table 7: Métricas de actividad en repositorios Git por grupo *(datos simulados ilustrativos)*

11.3 Rendimiento en la Plataforma

Indicador	Resultado
Porcentaje de completación del bootcamp	87 %
Calificación promedio en evaluación final	8.4 / 10.0
Tiempo promedio para completar el bootcamp	3.2 h
Tasa de reprobación del módulo	6 %

Table 8: Indicadores de uso y desempeño en Malleus Codeficarum *(datos simulados ilustrativos)*

12 Discusión

Los resultados presentados, aunque simulados, permiten proyectar el tipo de análisis que se realizará en la fase de implementación real y su relación con la literatura revisada.

La reducción proyectada del Coeficiente de Gini (de 0.71 a 0.29 en el grupo experimental) indica que la distribución del trabajo podría volverse considerablemente más equitativa tras la intervención. Este hallazgo esperado es consistente con los planteamientos de Nguyen et al. [6], quienes señalan que el análisis de métricas de repositorios permite identificar y corregir desequilibrios de participación.

El incremento proyectado en el conocimiento de Git del grupo experimental (de 2.0 a 7.8 en escala de 10) confirmaría que el formato de bootcamp interactivo es efectivo para nivelar competencias técnicas en tiempos reducidos, coincidiendo con la propuesta de Ortiz Jaramillo [12] sobre la eficacia de los entornos de aprendizaje intensivos.

La disminución esperada en la percepción de holgazanería social (de 3.9 a 2.1) refuerza el argumento de Gabelica et al. [1], quienes postulan que el aprendizaje grupal genuino actúa como factor motivador que previene la reducción del esfuerzo individual.

Sin embargo, existen limitaciones a considerar en el diseño. La imposibilidad de asignar grupos aleatoriamente restringe la validez interna del estudio. Además, el uso exclusivo de métricas cuantitativas de Git (commits, LOC) puede ser insuficiente, ya que actividades de diseño, documentación y coordinación no siempre se reflejan en el repositorio. Esto se alinea con la perspectiva de Hundhausen et al. [8], quienes proponen complementar las métricas de GitHub con registros de comunicación y evaluaciones entre pares.

13 Conclusiones

A partir del diseño de investigación propuesto y los resultados ilustrativos presentados, se establecen las siguientes conclusiones preliminares, sujetas a validación con datos reales en la fase de implementación:

1. **La capacitación inicial es determinante.** La implementación de *Malleus Codeficarum* está diseñada para reducir significativamente la brecha de conocimiento previo en Git, sentando una base técnica común que facilita una participación más equitativa desde el inicio del proyecto.

2. **El uso obligatorio de Git puede reducir la concentración de trabajo.** El Coeficiente de Gini proyectado para el grupo experimental (0.29) frente al del grupo control (0.71) demuestra cómo la adopción estructurada de un sistema de control de versiones puede promover una distribución más equitativa de las contribuciones individuales.

3. **Las métricas objetivas mejoran la evaluación docente.** El uso de datos de actividad en los repositorios brindará al docente información precisa y continua sobre el desempeño individual, superando las limitaciones de la evaluación basada únicamente en el producto final.

4. **El formato bootcamp es escalable y viable.** Con un tiempo promedio proyectado de 3.2 horas para completar el bootcamp y una tasa de aprobación del 94 %, la plataforma demuestra ser una herramienta eficiente dentro del contexto universitario.

5. **Se requiere una evaluación con datos reales.** Para validar de manera robusta la hipótesis de investigación, es necesario aplicar el estudio con grupos universitarios reales, ampliar la muestra e incorporar instrumentos cualitativos como entrevistas y evaluaciones entre pares.

En conclusión, los hallazgos proyectados respaldan la viabilidad e impacto positivo esperado de integrar un bootcamp interactivo de Git con el uso obligatorio de control de versiones en proyectos universitarios de programación. Este enfoque no solo busca reducir la holgazanería social, sino también preparar a los estudiantes con competencias técnicas relevantes para su futuro desempeño profesional en el desarrollo de software.

References

[1] C. Gabelica, S. De Maeyer, y M. C. Schippers, "Taking a free ride: How team learning affects social loafing," *J. Educ. Psychol.*, vol. 114, núm. 4, pp. 716–733, may. 2022.

[2] L. E. Narváez Díaz et al., "Propuesta metodológica para mejorar el desempeño académico de los estudiantes en fundamentos de programación," *RIDE*, vol. 14, núm. 27, nov. 2023.

[3] L. Luceño Casals, *Enseñanza e Innovación Educativa en el ámbito Universitario*, 1.a ed. Madrid: Dykinson, 2024.

[4] G. Wolf, "Using the Git Version Control System to Replace a Learning Management System," *IEEE Rev. Iberoam. Tecnol. Aprendiz.*, vol. 19, pp. 24–32, 2024.

[5] P. Patani, S. Tiwari, y S. S. Rathore, "The impact of GitHub on students' learning and engagement in a software engineering course," *Comput. Appl. Eng. Educ.*, vol. 32, núm. 5, 2024.

[6] B.-A. Nguyen, H.-M. Chen, y C.-R. Dow, "Identifying nonconformities in contributions to programming projects," *Behav. Inf. Technol.*, vol. 42, núm. 1, pp. 141–157, 2023.

[7] J. Cui et al., "Correlating Students' Class Performance Based on GitHub Metrics," en *Proc. ITiCSE 2023*, Turku: ACM, 2023.

[8] C. Hundhausen et al., "Combining GitHub, Chat, and Peer Evaluation Data to Assess Individual Contributions," *ACM Trans. Comput. Educ.*, vol. 23, núm. 3, 2023.

[9] Universidad Nacional Autónoma de Tayacaja et al., "Fomento del pensamiento computacional," *RISTI*, núm. 48, pp. 23–40, mar. 2023.

[10] E. Serrano Collado, "Autocorrección interactiva para la enseñanza y aprendizaje de la programación," Universidad de Granada, 2023.

[11] L. Martínez Allende, A. I. García Monroy, y E. E. Linares González, "El juego, estrategia pedagógica en la enseñanza de la programación," *RIDE*, vol. 13, núm. 25, ago. 2022.

[12] I. Ortiz Jaramillo, "Diseño de bootcamp de programación y pensamiento computacional con enfoque transdisciplinar," 2024.

[13] A. Jovanović y A. Milosavljević, "VoRtex Metaverse Platform for Gamified Collaborative Learning," *Electronics*, vol. 11, núm. 3, 2022.

[14] V. M. Campbell Rodríguez, "Revolucionando la Educación: Integración de IA en Sistemas de Gestión del Aprendizaje," *RIDE*, vol. 15, núm. 30, ene. 2025.

[15] M. Sofi-Karim, A. O. Bali, y K. Rached, "Online education via media platforms and applications as an innovative teaching method," *Educ. Inf. Technol.*, vol. 28, núm. 1, pp. 507–523, 2023.