Del error al aprendizaje: *Dojo*, una plataforma para mejorar la enseñanza de la programación

Lucas Videla, Renata Guatelli y Verónica Aubín

Universidad Nacional de La Matanza, Buenos Aires, Argentina,
{lvidela, rguatelli, vaubin}@unlam.edu.ar

Resumen La enseñanza de la programación enfrenta desafíos pedagógicos particulares, como la necesidad de retroalimentación constante, la práctica sostenida y la comprensión progresiva de errores. En contextos educativos heterogéneos numerosos, los métodos tradicionales de corrección manual resultan insuficientes y poco escalables.

Este artículo presenta *Dojo*, una plataforma virtual de código abierto diseñada para acompañar la enseñanza de la programación. Este sistema integra mecanismos de retroalimentación automatizada mediante herramientas profesionales de análisis de código, que permiten detectar errores funcionales y de estilo en la solución de los estudiantes. *Dojo* no solo identifica problemas, sino que genera explicaciones contextualizadas y proporciona recursos de aprendizaje específicos, permitiendo a los estudiantes recibir correcciones precisas de manera temprana. Un repositorio pedagógico especializado complementa estos análisis técnicos, traduciendo los mensajes de error a un lenguaje accesible. Desde el punto de vista didáctico, la visualización directa de errores y su corrección contribuyen a la autorregulación del aprendizaje, facilitando la adquisición progresiva de habilidades de depuración y mejora del código, fomentando una interpretación constructiva del error.

Los resultados preliminares obtenidos con la implementación de la plataforma han sido alentadores, mostrando beneficios en términos de agilidad en las devoluciones, dando mayor autonomía a los estudiantes y potenciando el rol docente, permitiéndole enfocarse en aspectos conceptuales avanzados.

Su arquitectura modular permite futuras expansiones, consolidando su potencial como recurso educativo transformador, convirtiendo el error en una oportunidad de aprendizaje. *Dojo* representa un puente entre la formación académica y las prácticas profesionales, ofreciendo un modelo escalable para la enseñanza de programación.

Keywords: enseñanza de programación, retroalimentación automatizada herramientas educativas, aula virtual

Abstract The teaching of programming faces particular pedagogical challenges, such as the need for continuous feedback, consistent practice, and a gradual understanding of errors. In numerous heterogeneous educational contexts, traditional manual correction methods are insufficient and not scalable.

To address these challenges, this article presents *Dojo*, an open source virtual platform designed to accompany the teaching of programming. This system integrates automated feedback mechanisms through industry-grade code analysis tools, which enable the detection of functional and stylistic errors in the students' solution. *Dojo* not only identifies problems, but also generates contextualized explanations and provides specific learning resources, allowing students to receive accurate corrections early on. A specialized pedagogical repository complements these technical analyses, translating error messages into student-friendly language. From a didactic point of view, the direct visualization of errors and their correction contribute to the self-regulation of learning, facilitating the progressive acquisition of debugging and code improvement skills, encouraging a constructive interpretation of the error.

Preliminary results from the platform's implementation have been encouraging, showing benefits in terms of faster feedback cycles, giving greater autonomy to students and enhancing the educator's role, allowing them to focus on advanced conceptual aspects.

Its modular architecture allows for future expansions, consolidating its potential as a transformative educational resource, transforming errors into valuable learning opportunities. *Dojo* represents a bridge between academic training and professional practices, offering a scalable model for teaching programming.

Keywords: computer programming instruction, automated feedback, educational technologies, virtual learning environment

1. Introducción

El aprendizaje de la programación constituye, en los entornos educativos actuales, uno de los desafíos más complejos tanto para estudiantes como para docentes. A diferencia de otros saberes, su adquisición requiere una práctica constante, un alto nivel de abstracción y una comprensión progresiva de los errores para avanzar en la construcción de conocimiento. En este marco, la retroalimentación oportuna y significativa se vuelve central para orientar el proceso de aprendizaje, especialmente en niveles iniciales, donde la desmotivación o la falta de comprensión pueden derivar en el abandono (Almeida et al., 2018).

Las dinámicas tradicionales de enseñanza de la programación presentan limitaciones que dificultan un acompañamiento pedagógico eficaz. La corrección manual del código, la necesidad de personalizar la devolución para grupos heterogéneos y la escasez de tiempo docente se convierten en obstáculos recurrentes

(Hollingsworth, 1960). En este enfoque, el docente actúa como evaluador, revisando cada entrega de código en persona o mediante comentarios escritos, lo que genera un ciclo continuo de entrega, corrección y devolución que puede variar su calidad y profundidad en función del tiempo y los recursos disponibles (Ureel Ii y Wallace, 2019). La necesidad de adaptar las devoluciones a estudiantes con distintos niveles de conocimiento, exigen devoluciones personalizadas, lo que no siempre es viable en escenarios con alta carga docente o muchos alumnos.

La expansión de la educación mediada por tecnologías y el acceso ubicuo a dispositivos digitales han abierto nuevas posibilidades para repensar estrategias pedagógicas más eficientes, interactivas y sostenibles.

Frente a este escenario, en este trabajo se presenta *Dojo* (Dojo Team, 2025), una plataforma virtual desarrollada con el objetivo de fortalecer la enseñanza y el aprendizaje de la programación. Su principal innovación radica en la incorporación de un sistema de retroalimentación automatizada, que no solo detecta errores en el código de los estudiantes, sino que ofrece explicaciones contextualizadas y enlaces a recursos formativos que promueven la comprensión y la mejora continua. Este enfoque se basa en la premisa de que el error, más que un indicador de fracaso, debe ser interpretado como un recurso didáctico esencial para la construcción de aprendizajes significativos.

Inspirada en prácticas profesionales de calidad de software, *Dojo* no pretende reemplazar la figura del docente, sino potenciar su rol mediante la liberación de tareas repetitivas, permitiendo así una dedicación más profunda a aspectos conceptuales y formativos (Edwards, 2003). Esta propuesta se enmarca en un paradigma pedagógico que promueve la autonomía del estudiante mediante la exploración, la autorregulación y el desarrollo del pensamiento computacional.

En las secciones siguientes se describe el diseño, los fundamentos y las funcionalidades de *Dojo*, así como su potencial impacto en contextos educativos donde la práctica constante, la retroalimentación eficaz y la retención del estudiantado resultan desafíos prioritarios.

2. Marco teórico

2.1. El error como motivador de aprendizaje

En el ámbito educativo, el error ha dejado de concebirse como un indicador de fracaso para ser reconocido como una instancia valiosa en el proceso de aprendizaje. Desde una perspectiva constructivista, constituye una oportunidad para que el estudiante confronte sus ideas previas, detecte inconsistencias y reestructure su conocimiento, favoreciendo así una comprensión más profunda y duradera. Tal como plantea Astolfi, el error no es un accidente, sino un indicador de los procesos intelectuales del alumno, y su análisis permite identificar obstáculos conceptuales que pueden resignificarse didácticamente (Astolfi, 2003). En el contexto particular de la enseñanza de la programación, esta mirada resignificada transforma al aula en un espacio donde los errores no son barreras, sino herramientas fundamentales para el desarrollo de un pensamiento computacional robusto y adaptable. En este sentido, aprender implica necesariamente

arriesgarse a errar; y el rol del docente consiste en acompañar ese proceso con una mirada comprensiva y reflexiva, capaz de transformar el error en un punto de partida para el aprendizaje genuino.

2.2. Herramientas de la Industria del Software.

Los profesionales del software apoyan su labor diaria en el uso de herramientas automatizadas que verifican aspectos comunes del trabajo que realizan. De este modo pueden garantizar la validez y calidad del código desarrollado. Asimismo, se puede aprovechar esta tendencia del mercado para obtener resultados similares en los estudiantes de programación. La confección de herramientas *ad-hoc* que sirvan para la validación del código suministrado resultaría en la reinvención de muchas preexistentes. Es por ello que se ha potenciado *Dojo*, incorporando *dojobot* y un conjunto inicial de herramientas enfocadas en pruebas unitarias y comprobación de estilo. Estas tecnologías no solo automatizan la detección de errores, sino que los transforman en oportunidades pedagógicas: las pruebas unitarias convierten los fallos en casos de estudio que revelan problemas de lógica o diseño, mientras que los chequeos de estilo identifican malas prácticas antes de que se arraiguen en el proceso de aprendizaje.

Pruebas unitarias con JUnit. Este *framework* de pruebas unitarias es utilizado en forma masiva para tecnologías Java. Es una herramienta que permite evaluar la corrección del código mediante la ejecución de casos de prueba predefinidos, arrojando un resultado binario: el código pasa satisfactoriamente, o falla, en cada una de las pruebas (Mekterovic et al., 2020).

Verificación de estilo con Checkstyle. Esta herramienta analiza el código en busca de malas prácticas de programación e incumplimientos de convenciones de estilo. Si bien está pensado originalmente para ser configurado según las preferencias particulares de cada grupo de trabajo, existen ciertos estándares en la escritura de código Java que suelen ser la norma dentro de la comunidad de dicho lenguaje (Leinonen et al., 2022).

3. Desarrollo

En el contexto de la enseñanza de la programación, *Dojo* se presenta como una plataforma de código abierto orientada a optimizar el proceso de enseñanza-aprendizaje en entornos híbridos. Su diseño contempla un aula virtual integral, accesible desde cualquier dispositivo con navegador web, que permite gestionar contenidos, consignas, entregas y retroalimentación de manera unificada. La versión 2.0, objeto de este trabajo, incorpora un sistema de retroalimentación automatizada mediante agentes de corrección, lo que habilita un ciclo continuo de evaluación formativa.

Esta funcionalidad no solo agiliza el seguimiento del progreso estudiantil, sino que integra un repositorio pedagógico que contextualiza técnicamente los errores

detectados. De este modo, se combinan herramientas propias de la industria del software con explicaciones formativas adaptadas al nivel de los estudiantes, promoviendo una comprensión profunda de los errores y facilitando la mejora iterativa.

El desarrollo de esta nueva versión se apoyó en la experiencia obtenida con la versión inicial de la plataforma, utilizada durante el año 2024. A lo largo de ese ciclo, se recolectó un corpus considerable de resoluciones entregadas por los estudiantes, que sirvió como base para entrenar el sistema. Las entregas fueron procesadas por *dojobot*, el agente de corrección automatizada, lo que permitió validar la generación de informes, identificar patrones de errores frecuentes y ajustar las respuestas pedagógicas del sistema.

Este enfoque permitió avanzar en el desarrollo sin depender de una cohorte activa, evitando las restricciones asociadas al calendario académico. Los primeros resultados evidenciaron la capacidad de *dojobot* para detectar tanto errores funcionales como problemas de estilo, a partir de los cuales se construyeron las primeras explicaciones pedagógicas. Así se dio forma inicial al sistema de retroalimentación enriquecida que distingue a esta nueva versión de la plataforma.

3.1. Funcionalidades

Las siguientes funcionalidades son interesantes a los efectos de este trabajo:

Extensión digital del aula. Se implementó un sistema de aula virtual, desde donde se facilita la gestión de un curso, con acceso a material teórico, calendario académico y otros elementos necesarios en todo curso universitario, como puede verse en la Figura 1. La virtualización del entorno educativo permite la gestión de múltiples comisiones, formación de grupos de trabajo y el registro de asistencia.

Consignas de programación. Dojo incorpora un sistema de difusión de consignas que permite a los estudiantes, ya sea de forma individual o grupal, presentar sus soluciones de programación directamente en la plataforma mediante un editor de texto especializado en el manejo de código fuente. Esta funcionalidad facilita la recolección ordenada de entregas y permite el seguimiento automatizado de fechas límite y estados de cada actividad.

Tablero de control. Cada estudiante cuenta con un tablero de control personal desde el cual puede consultar todas las tareas asignadas junto con sus respectivas fechas de entrega. Este tablero les permite visualizar su progreso general en el curso, identificar actividades pendientes, revisar devoluciones recibidas y acceder a un resumen del desempeño académico. De esta forma se promueve la autonomía y la autorregulación del aprendizaje. El tablero también ofrece a los docentes una visión consolidada del avance de cada estudiante, facilitando el seguimiento y la toma de decisiones pedagógicas informadas.

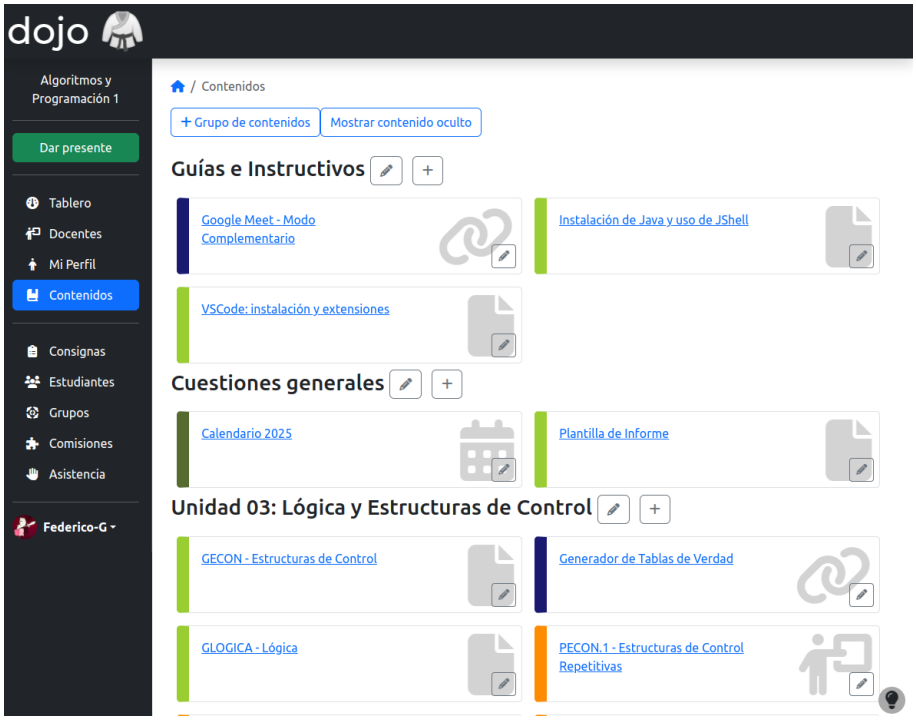
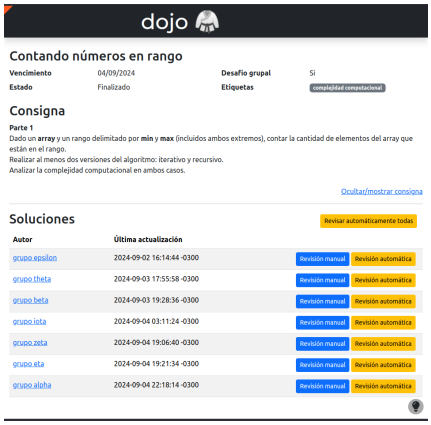


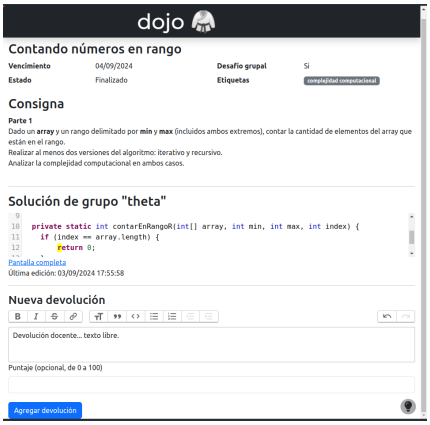
Figura 1: Pantalla de ejemplo del aula virtual de Dojo

Retroalimentación docente. Se brindan herramientas a los docentes para realizar devoluciones personalizadas sobre las soluciones presentadas por los estudiantes. Como se puede observar en la Figura 2a, hay una lista de soluciones entregadas por los alumnos con un botón azul para realizar la devolución manual. En la Figura 2b se puede observar que debajo de la consigna y la solución entregada por el estudiante o grupo se abre el espacio del editor de texto para escribir la devolución. Ésta puede hacerse mediante texto libre, o en forma estructurada a través de una rúbrica tabulada con escalas predefinidas (por ejemplo, escala del 1 al 5, opciones sí/no, escala de Likert). Esto permite una retroalimentación estructurada y objetiva, facilitando al alumno la comprensión de su desempeño y sus posibilidades de mejora.

Retroalimentación automática. Una vez realizada una entrega, el sistema permite al docente ejecutar, mediante la presión de un botón, las pruebas previamente configuradas para la actividad, incluyendo tanto los casos funcionales como las verificaciones de estilo de código establecidas de antemano. Esta funcionalidad debe configurarse en forma alineada a los objetivos pedagógicos del curso y puede adaptarse al momento del cuatrimestre en el que se utilice. Los resultados se presentan mediante reportes contextualizados, que quedan visibles junto a la entrega del estudiante, proporcionando una retroalimentación inmediata y



(a) Listado de soluciones grupales enviadas a una consigna



(b) Ejemplo de devolución manual a una resolución del grupo “theta”

Figura 2: Vista docente de las resoluciones a una consigna

coherente con los criterios de evaluación. Como puede verse en la Figura 2a existe un botón naranja en la parte superior derecha que realizaría la corrección automática del curso completo y un botón individual para cada solución que permitiría la corrección automática de un alumno.

3.2. Hacia una retroalimentación escalable y automatizada

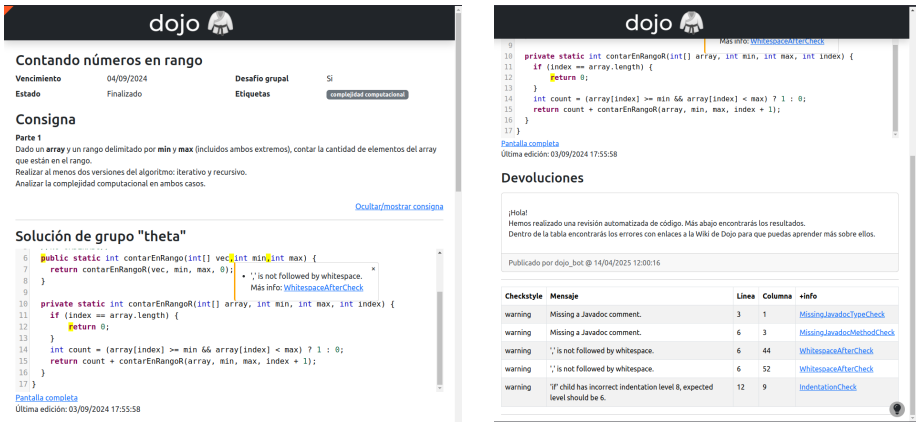
El proceso de retroalimentación convencional reviste una naturaleza manual, ya sea que se utilicen rúbricas o guías para hacerlo, o bien se realice en forma completamente libre. Es así que también este proceso reviste falencias, como ser la falta de consistencia o incluso la pluralidad de perspectivas, dependiendo de quién sea el docente encargado de realizar las devoluciones.

La plataforma permitió, desde su concepción, la existencia de múltiples revisiones por cada consigna. Esto facilitó el aporte de distintos puntos de vista al momento de la corrección, lo cual resulta de gran valor para comprender diversas perspectivas profesionales.

Como se ha establecido anteriormente, este proceso carece de una estructura que permita la objetividad y consistencia. Debe considerarse, también, el tiempo que insume realizar una revisión profunda y consistente de las soluciones de los estudiantes de un curso de programación (Sarsa et al., 2022). Es por ello que se procedió al desarrollo e integración de un agente de corrección automatizada denominado *dojobot*. El mismo se encarga de tomar el código fuente proporcionado por los estudiantes, y ejecutar un conjunto de pruebas con el soporte de diversas herramientas de validación de código que analizan aspectos tanto estáticos como dinámicos del software.

La evaluación automática de programas puede realizarse de forma dinámica o estática, y ambas son complementarias (Ala-Mutka, 2005). La evaluación

dinámica consiste en ejecutar el código del estudiante para verificar su comportamiento en distintos casos. Esto permite comprobar si el programa funciona correctamente. Por otro lado, la evaluación estática analiza el código sin ejecutarlo. Permite revisar aspectos como el estilo, la estructura del programa, el uso correcto del lenguaje o incluso detectar errores potenciales. Una buena solución no se define sólo por su funcionamiento, sino también por su calidad estructural, y es por ello que se escogieron herramientas de ambos tipos.



(a) Errores marcados a lo largo del código suministrado, con carteles destacados

(b) Listado de errores tabulados para un seguimiento complementario

Figura 3: Vista de estudiante, de las correcciones automatizadas

Este nuevo enfoque permite detectar errores comunes de manera rápida y objetiva, reduciendo la carga de corrección de los docentes, eliminando inconsistencias y fomentando el aprendizaje basado en la mejora iterativa.

Verificación del comportamiento esperado Se integró JUnit (JUnit Team, 2025) para verificar los requisitos funcionales de las entregas suministradas por los estudiantes. Es de primordial importancia verificar, antes que nada, que la entrega realice aquello que se ha solicitado. Como complemento a las pruebas básicas, se pueden agregar casos que los estudiantes probablemente no hubieran pensado con antelación, y de tal modo alertarlos sobre las alteraciones en los datos de entrada que pueden generar que su código se comporte de formas inesperadas.

Revisión de convenciones de codificación Mediante la integración de Checkstyle (Checkstyle Team, 2025), los estudiantes reciben un informe con las incidencias detectadas en el código enviado, que puede señalarse dentro del mismo código

(Figura 3a), o mediante una tabla detallada (Figura 3b). Estas incidencias abarcan desde aspectos simples, como el uso correcto de espacios, sangrías y nombres de variables, hasta cuestiones más profundas, como la falta de documentación, estructuras innecesariamente complejas o uso inadecuado de construcciones del lenguaje. De este modo se fomenta la escritura de código limpio, legible y mantenible, alineado a las prácticas profesionales.

Otras extensiones. *Dojo* integra actualmente dos herramientas ampliamente reconocidas en el entorno profesional del desarrollo en Java. No obstante, desde su diseño, *dojobot* fue concebido como un sistema flexible y extensible, capaz de incorporar nuevas herramientas de análisis y soportar otros lenguajes de programación. La arquitectura modular de su sistema permite programar integraciones adicionales sin reestructurar su funcionamiento central, abriendo el camino hacia nuevos contextos educativos y tecnológicos.

Repositorio pedagógico. Las herramientas utilizadas en la industria del software suelen brindar retroalimentación técnica que muchas veces resulta poco accesible para las personas que están dando sus primeros pasos en la programación. Para complementar la retroalimentación automatizada proporcionada por las herramientas anteriormente mencionadas, se desarrolló una wiki con explicaciones detalladas sobre los errores detectados (Parker y Chao, 2007). Esta wiki no sólamente traduce los mensajes técnicos a un lenguaje accesible, sino que también ofrece ejemplos y recomendaciones concretas para corregirlos y evitarlos a futuro (Figura 4). De este modo se ayuda al estudiante a comprender la causa de sus errores y cómo mejorar progresivamente sus futuras entregas de código.

4. Evaluación en contexto real

Durante el presente ciclo lectivo, la plataforma *Dojo* fue incorporada en siete comisiones, alcanzando a más de 250 estudiantes. Esta primera implementación en condiciones reales permitió comenzar a observar el funcionamiento del sistema de retroalimentación automatizada dentro de un entorno educativo activo. Si bien el período de uso aún es breve para realizar evaluaciones concluyentes, se registraron algunas mejoras iniciales, como mayor agilidad en la devolución de entregas y una reducción parcial en las tareas repetitivas que realiza el cuerpo docente. Asimismo, se comenzó a recolectar información sobre errores frecuentes, tiempos de entrega y patrones de corrección, lo que permitirá orientar futuras mejoras. Desde la percepción docente, se observa una disposición positiva de los estudiantes hacia la herramienta, especialmente por la posibilidad de visualizar su avance y recibir devoluciones tempranas. Algunos docentes también señalaron un incremento en la participación y el compromiso con las actividades prácticas. Si bien los resultados son preliminares, esta implementación constituye un paso relevante hacia modelos de enseñanza más escalables y sostenidos.

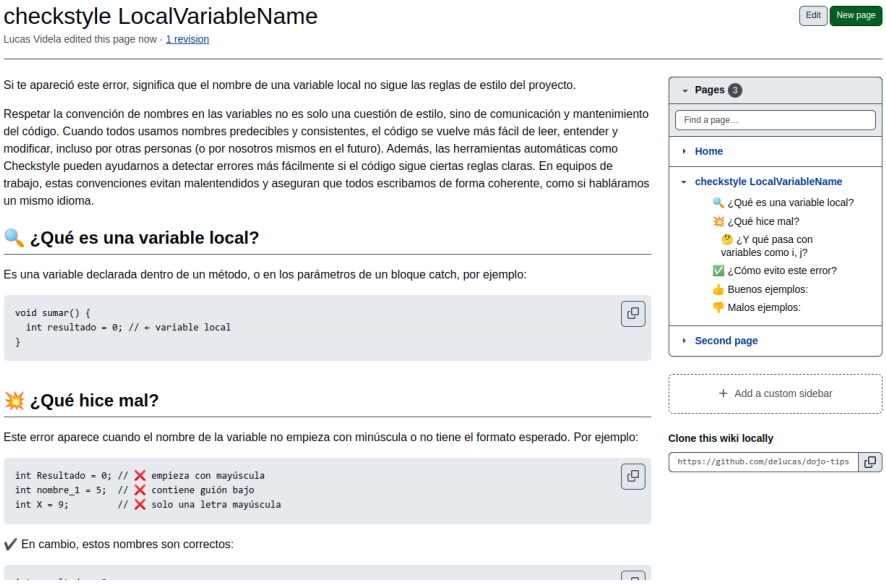


Figura 4: Ejemplo de página del repositorio pedagógico

5. Conclusión

El desarrollo de competencias en programación plantea desafíos pedagógicos que requieren enfoques centrados en el estudiante, alineados con prácticas profesionales y orientados a la autonomía, la reflexión y la mejora continua. En este contexto, *Dojo* se presenta como una solución concreta y adaptable, capaz de ofrecer retroalimentación oportuna y formativa en contextos educativos masivos y heterogéneos.

Integrando herramientas como JUnit y Checkstyle con un repositorio pedagógico accesible, la plataforma garantiza devoluciones rápidas, coherentes y escalables. Los primeros resultados, obtenidos con más de 250 estudiantes, muestran una mejora en los tiempos de corrección y en la calidad del proceso formativo. El estudiante puede iterar sobre sus errores, dando lugar a un ciclo virtuoso de detección, corrección y reflexión, donde el error deja de ser un obstáculo para convertirse en un recurso de aprendizaje.

Desde la perspectiva docente, *Dojo* no reemplaza su rol, sino que lo potencia. Al automatizar tareas simples y repetitivas, libera tiempo para tutorías personalizadas y el abordaje de aspectos conceptuales más profundos. Asimismo, su capacidad de escalar sin comprometer la calidad pedagógica la convierte en una herramienta valiosa para entornos con alta matrícula.

La incorporación de una wiki explicativa permite abordar errores frecuentes con claridad y contextualización, fomentando la autonomía del estudiante y reduciendo la dependencia de respuestas individuales. La sistematización de

patrones de error habilita la creación de recursos didácticos más eficaces e intervenciones pedagógicas más informadas.

El éxito de *Dojo* reside en asumir que aprender a programar implica equivocarse, pero también en diseñar entornos donde esos errores se transformen en oportunidades concretas para construir una comprensión más profunda y robusta.

6. Próximos trabajos

En adelante, se continuará utilizando y refinando la herramienta en nuevas cohortes, incorporando mejoras a partir del uso en condiciones reales. Se prevé ampliar la base de usuarios, extendiendo su aplicación a más comisiones o materias, lo que permitirá detectar nuevos patrones y ajustar funcionalidades.

Entre las próximas incorporaciones se contempla el desarrollo de herramientas de visualización del progreso, que faciliten a estudiantes y docentes el seguimiento del aprendizaje y las mejoras realizadas a lo largo del curso. También se evaluará la integración de nuevas herramientas de la industria del software, la expansión a otros lenguajes de programación y la incorporación de funcionalidades para el trabajo colaborativo entre estudiantes. Estas líneas de trabajo buscan consolidar a *Dojo* como una plataforma escalable, versátil y alineada con las necesidades pedagógicas y tecnológicas de la enseñanza de la programación.

7. Agradecimientos

Este trabajo se enmarca en el proyecto C2-ING-125, titulado “Retroalimentación en la Era Digital: Diseño y Simulación de una Herramienta para Evaluación de Código Fuente” del DIIT de la Universidad Nacional de La Matanza. En el marco de esta línea de investigación, se desarrolla actualmente la tesis de Maestría en Tecnología Informática del Ing. Lucas Videla.

Referencias

Ala-Mutka, K. M. (2005). A Survey of Automated Assessment Approaches for Programming Assignments. *Computer Science Education*, *15*(2), 83-102. https://doi.org/10.1080/08993400500150747

Almeida, J. B., Cunha, A., Macedo, N., Pacheco, H., & Proença, J. (2018). Teaching how to program using automated assessment and functional glossy games (experience report). *Proceedings of the ACM on Programming Languages*, *2*(ICFP), 1-17. https://doi.org/10.1145/3236777

Astolfi, J. P. (2003). *El 'error' un medio para enseñar: Gaston Bachelard, Jean Piaget* (2a. ed). Díada.
OCLC: 630238849.

Checkstyle Team. (2025). *Checkstyle 10.23.0* [Accedido el 16/04/2025]. https://checkstyle.org/

Dojo Team. (2025). *Dojo* [Accedido el 16/04/2025]. https://dojo.sipofcode.com

Edwards, S. (2003). Using Test-Driven Development in the Classroom : Providing Students with Automatic, Concrete Feedback on Performance.

Hollingsworth, J. (1960). Automatic graders for programming classes. *Communications of the ACM*, *3*(10), 528-529. https://doi.org/10.1145/367415.367422

JUnit Team. (2025). *JUnit 5* [Accedido el 16/04/2025]. https://junit.org/junit5/

Leinonen, J., Denny, P., & Whalley, J. (2022). A Comparison of Immediate and Scheduled Feedback in Introductory Programming Projects. *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education*, 885-891. https://doi.org/10.1145/3478431.3499372

Mekterovic, I., Brkic, L., Milasinovic, B., & Baranovic, M. (2020). Building a Comprehensive Automated Programming Assessment System. *IEEE Access*, *8*, 81154-81172. https://doi.org/10.1109/ACCESS.2020.2990980

Parker, K., & Chao, J. (2007). Wiki as a Teaching Tool. *Interdisciplinary Journal of E-Learning and Learning Objects*, *3*(1), 57-72.

Sarsa, S., Leinonen, J., Koutcheme, C., & Hellas, A. (2022). Speeding Up Automated Assessment of Programming Exercises. *The United Kingdom and Ireland Computing Education Research (UKICER) Conference*, 1-7. https://doi.org/10.1145/3555009.3555013

Ureel Ii, L. C., & Wallace, C. (2019). Automated Critique of Early Programming Antipatterns. *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, 738-744. https://doi.org/10.1145/3287324.3287463