
# La gamificación aplicada al aprendizaje de los principios de la programación orientada a objetos en los alumnos de ingeniería de software y tecnologías de la información

Ortiz Adame, Nicolás Esteban

Departamento de Ciencias de la Computación

Carrera de Ingeniería de Sistemas e Informática

Trabajo de titulación, precio a la obtención del título de Ingeniero en Sistemas e Informática

PhD. Lascano, Jorge Edison

22 de febrero de 2023

**ANALYSIS SUMMARY**
*magister*

# 20230206OrtizNicolasProyectoTitulacion

**5%** Similarities

2% Text between quotes
0% similarities between quotation marks
0% Language not recognised

**Document**
**name:** 20230206OrtizNicolasProyectoTitulacion.pdf
**Document**
**ID:** 27c9ce73e7eae4ae5ba141e3080c571c5c2560b6
**Original document size:** 1.94 Mo

**Submitter:** JORGE EDISON LASCANO
**Submission date:** 2/7/2023
**Upload type:** interface
**analysis end date:** 2/7/2023

**Number of words:** 28,600
**Number of characters:** 208,660

Location of similarities in the document:

## Main sources detected

| No. | Description | Similarities | Locations | Additional information |
|---|---|---|---|---|
| 1 | **repositorio.espe.edu.ec** \| Estudio comparativo entre paradigmas de programación y ...<br>http://repositorio.espe.edu.ec:8080//bitstream/21000/22123/5/T-ESPE-043669.pdf.txt<br>7 similar sources | 3% | | Identical words : 3% (864 words) |
| 2 | **www.iste.org** \| ISTE<br>https://www.iste.org/es/explore/In-the-classroom/5-ways-to-gamify-your-classroom<br>1 similar source | < 1% | | Identical words : < 1% (264 words) |
| 3 | **yachachiq.pe** \| 5 formas de gamificar tu aula - YACHACHIQ.PE<br>https://yachachiq.pe/5-formas-de-gamificar-tu-aula/ | < 1% | | Identical words : < 1% (180 words) |
| 4 | **tesis.pucp.edu.pe**<br>https://tesis.pucp.edu.pe/repositorio/bitstream/handle/20.500.12404/17477/SANTILLANA_VALDIVIA_M... | < 1% | | Identical words : < 1% (70 words) |
| 5 | **hdl.handle.net** \| Análisis de uso de la Gamificación en la Enseñanza de la Informática<br>http://hdl.handle.net/10045/125759<br>2 similar sources | < 1% | | Identical words : < 1% (67 words) |

## Sources with incidental similarities

| No. | Description | Similarities | Locations | Additional information |
|---|---|---|---|---|
| 1 | **umg-desarrollo-software.blogspot.com** \| Metodologías tradicionales<br>https://umg-desarrollo-software.blogspot.com/2022/02/metodologias-tradicionales.html | < 1% | | Identical words : < 1% (40 words) |
| 2 | **aprendejugandoeym.blogspot.com** \| ¡Aprende Jugando¡: Beneficios del Juego<br>https://aprendejugandoeym.blogspot.com/2018/08/el-juego-es-un-invento-poderoso-de-la.html | < 1% | | Identical words : < 1% (36 words) |
| 3 | **redalyc.org** \| "Gamificación" de la enseñanza para ciencia, tecnología, ingeniería y m...<br>https://redalyc.org/journal/998/99863569004/html/ | < 1% | | Identical words : < 1% (20 words) |
| 4 | **Document from another user** #af9b00<br>The document is from another group | < 1% | | Identical words : < 1% (17 words) |
| 5 | **profile.es** \| SOLID: los 5 principios que te ayudarán a desarrollar software de calidad<br>https://profile.es/blog/principios-solid-desarrollo-software-calidad/ | < 1% | | Identical words : < 1% (20 words) |

## Referenced sources (without similarities detected)

These sources were cited in the paper without finding any similarities.

1 https://github.com/neortiz1/Playbox
2 https://github.com/neortiz1/Geometry
3 https://github.com/neortiz1/Chess
4 https://github.com/neortiz1/SRP-OCP-LSK
5 https://doi.org/10.1109/ISECon.2018.8340460

Firmado electrónicamente por:
JORGE EDISON
LASCANO .


**Departamento de Ciencias de la Computación**

**Carrera de Ingeniería de Sistemas e Informática**

**Certificación**

Certifico que el trabajo de titulación: "**La gamificación aplicada al aprendizaje de los principios de la programación orientada a objetos en los alumnos de ingeniería de software y tecnologías de la información**" fue realizado por el señor **Ortiz Adame, Nicolás Esteban;** el cual ha sido revisado y analizado en su totalidad por la herramienta de verificación de similitud de contenido; por lo tanto cumple con los requisitos legales, teóricos y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, razón por la cual me permito acreditar y autorizar para que lo sustente públicamente.

Sangolquí, 22 de febrero de 2023

Firma:


.....

**Ing. Jorge Edison Lascano, PhD.**

C. C. 1710893114


**Departamento de Ciencias de la Computación**

**Carrera de Ingeniería de Sistemas e Informática**

**Responsabilidad de Autoría**

Yo, **Ortiz Adame, Nicolás Esteban**, con cédula de ciudadanía N°1716088867, declaro que el contenido, ideas y criterios del trabajo de titulación: **La gamificación aplicada al aprendizaje de los principios de la programación orientada a objetos en los alumnos de ingeniería de software y tecnologías de la información** es de mi autoría y responsabilidad, cumpliendo con los requisitos legales, teóricos, científicos, técnicos, y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, respetando los derechos intelectuales de terceros y referenciando las citas bibliográficas.

Sangolquí, 22 de febrero de 2023

.....

**Ortiz Adame, Nicolás Esteban**

C.C.: 1716088867

**Departamento de Ciencias de la Computación**

**Carrera de Ingeniería de Sistemas e Informática**

**Autorización de Publicación**

Yo **Ortiz Adame, Nicolás Esteban**, con cédula de ciudadanía N°1716088867, autorizo a la Universidad de las Fuerzas Armadas ESPE publicar el trabajo de titulación: **La gamificación aplicada al aprendizaje de los principios de la programación orientada a objetos en los alumnos de ingeniería de software y tecnologías de la información** en el Repositorio Institucional, cuyo contenido, ideas y criterios son de mi/nuestra responsabilidad.

Sangolquí, 22 de febrero de 2023

.....

**Ortiz Adame, Nicolás Esteban**

C.C.: 1716088867

**Dedicatoria**

Este trabajo dedico primero a Dios por guiar el camino de mi vida, a mis padres por sus desvelos, por su apoyo incondicional, por creer en mí, por sus sabios consejos, sus valores, pero sobre todo por su infinito amor y fe en el Señor, a mi hermana por su sabiduría, amor y bendiciones, han hecho de mí una persona de bien y a mi novia por estar siempre pendiente de mí con su paciencia, amor y apoyo que me ayudaron a lograr este sueño.

Ortiz Adame, Nicolás Esteban

**Agradecimiento**

Agradezco a mi familia y a mi novia por su respaldo y apoyo incondicional.

A mi director de tesis, Ing. Jorge Edison Lascano, PhD. Por su calidad humana, por todo el tiempo brindado, en la guía para culminar con éxito y responsabilidad este proyecto de titulación.

A mi amigo Santiago Posada por brindarme su ayuda en todos los momentos difíciles en mi vida laboral y profesional.

**Índice de contenidos**

**Índice de tablas**

## Índice de figuras

**Resumen**

El conocimiento teórico en programación orientada a objetos puede ser abstracto y tedioso, haciendo que los estudiantes pierdan fácilmente el interés en cursos de informática que son de vital importancia en sus estudios y en su carrera profesional. Para dar solución a este problema, introducimos el concepto de gamificación en el curso de programación orientada a objetos en las carreras de Ingeniería de Software e Ingeniería en Tecnologías de la Información, con materiales didácticos y diseños de juegos como el juego Tetris, donde se encuentran los cuatro pilares de la programación orientada a objetos aplicados, es decir, encapsulamiento, abstracción, herencia y polimorfismo. Además, utilizamos otro juego para presentar a los estudiantes los tres primeros principios de SOLID: Single Responsibility, Open/Closed and Liskov Substitution (Los principios SOLid) con el uso del juego Ajedrez. Los estudiantes al programar con juegos piensan en la forma más rápida de codificarlos e interactuar con ellos, aprendiendo los principios de programación orientada a objetos. Vimos que hay un incremento en el conocimiento, las habilidades de codificación y la motivación, cuando los estudiantes aprenden los cuatro pilares de la programación orientada a objetos y Principios SOLid. Cuando se aplica la gamificación en los cursos de programación orientada a objetos, los estudiantes en su mayoría están más motivados y son más hábiles. Sin embargo, los hallazgos mostraron que no hubo diferencias significativas en los resultados generales del desempeño de los estudiantes. Se necesitan estudios futuros que incluyan una muestra más grande de estudiantes para obtener resultados más concluyentes.

*Palabras clave: conocimiento, gamificación, programación, habilidad, motivación.*

**Abstract**

Theoretical knowledge in object-oriented programming can be abstract and tiresome, easily making students lose interest in computer science courses that are of vital importance in their studies and in their professional career. To solve this problem, we introduce the concept of gamification in the object-oriented programming course in Software Engineering and Information Technology Engineering careers, with didactic materials and game designs such as the game Tetris, where the four pillars of object-oriented programming are applied, i.e., encapsulation, abstraction, inheritance, and polymorphism. Additionally, we used other game to introduce students to the three first principles of SOLID: Single Responsibility, Open/Closed and Liskov Substitution (SOLid Principles) with the use of the Chess game. When programming/playing the games, the students think about the fastest way to code the programs and interact with them by analyzing all the elements of the game's environments, at the same time, the students are learning the OOP principles. We saw that, there is an increment in knowledge, coding skills and motivation, when students learn the four pillars of object-oriented programming and the SOLid Principles. When gamification is applied in object-oriented programming courses, students are mostly more motivated and skillful. Nevertheless, the findings showed that there were not significant differences in the overall results of student's performance. Future studies that include a bigger sample of students is needed to get more conclusive results.

*Keywords: knowledge, gamification, programming, skill, motivation.*

# Capítulo I

## Introducción

En el capítulo I del presente proyecto de titulación se abordan los antecedentes, la problemática, justificación, objetivos (general y específicos), alcance, hipótesis y la estructura del proyecto.

## Antecedentes

Las disciplinas de la informática y la tecnología de la información se están popularizando día a día y la vida de la población depende cada vez más de las aplicaciones de software. El software actual tiene un efecto profundo en las actividades diarias, encontrándose en los hogares, oficinas, compras, comercio, además en automóviles, aviones, refrigeradores, tarjetas inteligentes, televisores; que dependen en gran medida del software.

Debido a la dependencia que existe de la tecnología en el mercado, los fundamentos de programación han sido incluidos como un curso básico, de acuerdo con el programa de educación superior, es decir, los estudiantes deben tomar este curso independiente de su especialidad. Hay que tomar en cuenta que, según las estadísticas, la tasa de empleo de desarrolladores de software ha incrementado en una tasa del 24% entre 2016 con previsión al 2026, sin embargo, en los institutos educativos, la tasa de abandono de los estudiantes de ciencias de la computación es muy alta, pero por otra parte la demanda de desarrolladores de software aumenta día tras día en la industria global y en particular en Sudamérica y en el Ecuador (Mora et al., 2015; Vahldick et al., 2014).

Algunas de las razones identificadas para la alta tasa de abandono son que los estudiantes sienten dificultad para comprender los conceptos de los fundamentos de programación, la falta de práctica, actividades de laboratorio mal diseñadas, la falta de pasión, esfuerzo intelectual y razonamiento lógico. También cabe mencionar que la falta de habilidades

cognitivas y de resolución de problemas entre los estudiantes dificultan el afrontamiento de temas desafiantes de la programación. Debido a esto, la mayoría de estudiantes abandonaron los estudios en el primer año de su carrera (Carreño-León et al., 2018).

Comprender los conceptos de programación siempre ha sido difícil para los estudiantes, especialmente para los programadores novatos, por lo tanto, una investigación planteó un método semántico, donde independientemente de un lenguaje de programación y una sintaxis en particular, establece que se debe incorporar el entorno de juego, que se centrará en las construcciones de programación y los conceptos abstractos de la programación, lo cual pueda mejorar el proceso de aprendizaje (Zapušek & Rugelj, 2013).

Asimismo, un estudio realizado en la India, determinó que la falta de pasión, el esfuerzo intelectual, el razonamiento lógico y la falta de habilidades cognitivas, hacían que sea difícil lidiar con temas desafiantes como la programación. Como consecuencia, la mayoría de alumnos abandonaron la carrera en el primer año de estudios (Kumar & Sharma, 2018).

La idea de aprender a través de juegos en un contexto educativo se inició en 1900, desde entonces ha evolucionado el uso de juegos en el aprendizaje académico a nivel de primaria, secundaria y educación superior. En 2014, (Vahldick et al., 2014), muestran evidencias sobre el aprendizaje basado en juegos, en disciplinas orientadas en la programación de software, sin embargo este método no es muy utilizado en el proceso de enseñanza. Posteriormente (Simões Gomes et al., 2018) comenzaron a trabajar en juegos formativos con el objetivo de involucrar y motivar a los estudiantes en el aprendizaje. Surge de esta manera, el concepto de juegos serios, que son un tipo de juegos formativos, los que se definen como aquellos juegos en los que el objetivo principal es el aprendizaje en lugar del entretenimiento, el regocijo o la diversión (Abdellatif et al., 2018).

A los juegos serios se los distingue de otros por tener el propósito formativo más que la distracción, por otro lado, el aprendizaje basado en la gamificación se refiere a la perspectiva educativa del uso de juegos, pero no con fines de entretenimiento. La gamificación se centra en aprender con diversión, consiste en utilizar una metodología de enseñanza de carácter lúdico que involucra datos educativos, enfocado en mejorar el proceso de aprendizaje mediante el uso de técnicas simples (Azmi et al., 2015). En la actualidad, seguir las antiguas y tradicionales metodologías de aprendizaje, resulta ser menos efectivo. (Layth Khaleel et al., 2019; Soflano et al., 2015) sugieren que la enseñanza basada en juegos aumenta la experiencia de aprendizaje y el interés de los estudiantes en comparación con los métodos tradicionales de aprendizaje.

En consecuencia, se puede decir que los métodos tradicionales de enseñanza son la razón principal de la falta de motivación e interés en los alumnos que comienzan en la programación. Por lo tanto, para superar esto, se sugiere que el enfoque de gamificación sea integrado en los cursos de programación, con el fin de impulsar la atención y la motivación. Varios investigadores han propuesto a la gamificación como una solución que puede mejorar el comportamiento y el compromiso de los alumnos para impartir/recibir habilidades (Díaz et al., 2018; Mi et al., 2018; Tsai, 2019).

**Planteamiento del problema**

Actualmente en Ecuador, un problema que se enfrenta de forma permanente en el sistema de educación superior es la alta tasa de abandono estudiantil. Muchos estudiantes no completan sus estudios, especialmente durante los primeros semestres de su programa de estudios, lo que reduce el nivel educativo y profesional del país. Esto es una preocupación importante, ya que el abandono estudiantil tiene consecuencias negativas tanto para los individuos como para la sociedad en su conjunto.

Se ha estimado que el 37,50% de los estudiantes que ingresan a la carrera de Ingeniería en Sistemas en el Ecuador desertan (Verdesoto et al., 2018). Esto es un porcentaje alarmantemente alto, y demuestra la necesidad de encontrar soluciones para este problema. Uno de los factores que inciden en la deserción es el poco interés por los estudios por la carrera y por la institución en la que estudian. Esto puede deberse a una falta de motivación o a dificultades académicas o personales que dificultan el proceso de aprendizaje.

La programación es un curso comúnmente incluido en la malla curricular de carreras relacionadas con las Tecnologías de la Información o las Ciencias de la Computación. Aunque el nombre puede variar según la institución, en esencia es el mismo objetivo, aprender a programar. Muchos estudiantes encuentran este curso complicado, aburrido y difícil de aprender, lo que puede afectar su motivación y rendimiento. Además, los profesores también pueden tener dificultades para enseñar los contenidos del curso y como consecuencia el hecho de que los estudiantes realicen tareas y ejercicios en clase también se dificulta (Zatarain Cabada & Zatarain Cabada, 2018).

Como resultado, muchos estudiantes tienen dificultades para comprender los contenidos de las asignaturas relacionadas con la programación y fracasan en el curso debido a su dificultad. Esto puede tener consecuencias negativas tanto para los individuos como para la sociedad en su conjunto, ya que la programación es una habilidad clave en muchas carreras y ocupaciones. Según investigaciones, los problemas que los estudiantes tienen para comprender los conceptos del curso de programación pueden deberse a que les resultan poco interesantes, lo que les desmotiva y dificulta el aprendizaje efectivo (Funabiki et al., 2013).

(Jenkins, 2001) asegura que los estudiantes deben estar motivados para que puedan participar adecuadamente en una actividad de aprendizaje, es así que con la motivación adecuada, su actitud cambiará a medida que avance el curso. Algunos estudiantes pueden

descubrir un gran interés en la informática, luego en la programación y desarrollar la motivación intrínseca que realmente están buscando. Por otro lado, otras personas pueden valorar el aprendizaje de la programación por razones básicamente externas, por ejemplo, tienen buenas perspectivas financieras una vez que comienzan a trabajar.

Por tal razón, la motivación es el determinante en el aprendizaje, de lo que se deriva la importancia de profundizar su estudio. Desde este aspecto, es importante comprender las expectativas de los estudiantes y entender cuáles son sus motivaciones, de esta manera, considerar la interacción entre las motivaciones internas, externas y las motivaciones negativas relacionadas con diversos factores que inciden en el desempeño estudiantil (Steinmann et al., 2013).

La presente investigación se centra en los estudiantes de Ingeniería de Software y Tecnologías de la Información de la Universidad de la Fuerzas Armadas ESPE y se enfoca en las dificultades que tienen los alumnos al aprender, desarrollar y practicar temas y actividades relacionados con los principios de la programación orientada a objetos. Se busca encontrar una solución para mejorar la práctica de programación y el aprendizaje de principios orientados a objetos, motivando al alumno y reduciendo la dificultad en el proceso de aprendizaje.

**Justificación**

La gamificación es la metodología de formación preferida, porque aumenta el atractivo de los procesos de aprendizaje, la innovación, la diversión, la productividad y, la capacidad de retener conocimientos y adquirir nuevas habilidades. Esta técnica consiste en utilizar técnicas de pensamiento y juego en entornos que no son de entretenimiento, como la educación o el trabajo.

Para enseñar, motivar e involucrar a los estudiantes en el aprendizaje de conceptos de programación, muchos investigadores utilizan la mecánica del juego a través de una historia.

Esta técnica es útil para los instructores, ya que les ayuda a lograr un mayor compromiso, motivación, colaboración, diversión y eficacia en el proceso de enseñanza.

Los juegos son herramientas que no son útiles, a menos que la gente los juegue, por lo tanto, los creadores de juegos deben estructurarlos con emoción a través de la dinámica y la mecánica del juego, para alentar a los jugadores a continuar jugando. Se utilizan cuatro estrategias en el diseño de juegos para desbloquear las emociones de los jugadores: 1) brindar oportunidades para el desafío, la estrategia y la resolución de problemas (diversión dura); 2) introducir elementos que fomenten el misterio, la intriga y la curiosidad (diversión fácil); 3) llevar a los jugadores a estados de ánimo de excitación o alivio (estados alterados); y 4) promover la competencia y el trabajo en equipo (diversión de la gente) (Lazzaro, s. f.).

En consecuencia, considerando la afirmación subyacente es que los métodos tradicionales de enseñanza son la razón principal de la falta de motivación e interés entre los programadores novatos, se sugiere que, para superar esto, el enfoque de gamificación debe estar integrado en los cursos de programación con el fin de impulsar la atención, la motivación y el compromiso de los estudiantes. Anteriormente, los investigadores habían sugerido que la gamificación puede ser una solución para mejorar el comportamiento y el compromiso de los estudiantes al impartir habilidades de programación (Tsai, 2019). A pesar de que la gamificación es popular y solicitada, hasta el momento los juegos no han logrado el resultado deseado como método de aprendizaje. Independientemente de los muchos juegos serios disponibles, se carece de un buen juego que transmita el concepto principal a los estudiantes. Según (Marti-Parreño et al., 2016), solo el 11,30% de los docentes han incorporado la metodología de gamificación en sus cursos mientras que la mayoría aprecia la noción, pero no la integra. Uno de los factores para esta falta de integración, es que los juegos no están disponibles en línea o sus versiones gratuitas no están disponibles.

Varios estudios han demostrado que el aprendizaje basado en juegos aumenta la experiencia de aprendizaje y su interés en relación con el aprendizaje tradicional. Esto no solo se aplica al campo de la informática, sino que también se ha demostrado que el aprendizaje basado en juegos es motivador y alentador para los estudiantes de otros campos (Shahid et al., 2019).

Bajo este contexto, para fortalecer el aprendizaje de la programación, se puede pensar en recurrir a la gamificación, que, mediante las dinámicas de juegos, ha mostrado ser una herramienta útil en diferentes áreas. En este trabajo se ha decidido desarrollar una propuesta basada en la gamificación para mejorar el aprendizaje de principios de la programación orientada a objetos y aumentar la motivación en los estudiantes, permitiendo que aprendan de manera agradable y minuciosa. Se espera que la técnica propuesta capture la atención de los estudiantes y les proporcione un entorno en el que se sientan cómodos, divertidos y motivados. El juego dinamiza la clase, despierta el interés previamente y lo mantiene durante todo el desarrollo de la actividad de aprendizaje. Además, permite al profesor estar al tanto de los avances y decisiones de los estudiantes, para detectar sus fortalezas y debilidades en relación con la asignatura y comprobar su nivel de comprensión de los conocimientos.

Gracias a este método de aprendizaje, los estudiantes desarrollarán mejor sus habilidades de razonamiento y se asume que serán más autónomos, ya que se encuentran en situaciones de reflexión, toma de decisiones y solución de problemas y errores en un entorno seguro. Con este método de aprendizaje, no solo asimilan conceptos de la asignatura o del tema en el que se centra la gamificación, sino que también desarrollan capacidades cognitivas a través del pensamiento crítico, la resolución de problemas y el análisis de la realidad.

Teniendo en cuenta la importancia que tiene esta investigación en relación a motivación aprendizaje y el fomentar un ambiente de clases más dinámico para los alumnos, se prevé que

los docentes puedan disponer de esta estrategia como modelo y lo puedan aplicar en sus sesiones de aprendizaje.

## Objetivos

### *Objetivo General*

Desarrollar una herramienta de apoyo basada en la gamificación, para aumentar la motivación y aprendizaje en los cuatro pilares de programación orientada a objetos y los 3 primeros principios SOLID, en los estudiantes de Ingeniería de Software y de Ingeniería de Tecnologías de la Información de la Universidad de las Fuerzas Armadas ESPE.

### *Objetivos Específicos*

- Realizar una revisión preliminar de la información de los contenidos de aprendizaje de la materia de programación orientada a objetos y sus respectivas unidades para llevar a cabo la gamificación a través de estrategias lúdicas.
- Diseñar un experimento sobre el desarrollo de una herramienta gamificada, para comparar el aumento de motivación y aprendizaje entre los grupos de estudiantes que utilicen gamificación y sin gamificación.
- Ejecutar el experimento diseñado, siguiendo los parámetros establecidos para conseguir datos sobre la investigación y analizar los resultados de los sujetos de estudio.
- Evaluar e interpretar los resultados de la propuesta, a partir de los datos recolectados y medir un aumento de aprendizaje y motivación en los estudiantes de programación orientada a objetos.

## Alcance

El alcance del proyecto será desarrollar una herramienta de apoyo basado en los principios de gamificación, con el objetivo motivar el proceso de aprendizaje en la materia de programación orientada a objetos, enfocado en los siguientes principios: encapsulamiento,

polimorfismo, abstracción, herencia, y en los tres primeros principios SOLID que de aquí en adelante lo representaremos de la siguiente forma SOLid, para los alumnos de Ingeniería de Software y Tecnologías de la Información en la Universidad de las Fuerzas Armadas ESPE, además entregar a los docentes información de la participación y contribución de cado uno de sus estudiantes. Por otra parte, se realizará una evaluación e interpretación de los resultados obtenidos, bajo parámetros técnicos estadísticos y así poder medir el aumento en los niveles de motivación y aprendizaje, para de esta manera verificar la eficacia de la implementación aplicada.

Para determinar de manera correcta el alcance de la investigación planteada, se muestra la siguiente Tabla 1.

**Tabla 1**

*Preguntas de investigación*

| Objetivo específico | Pregunta de investigación |
|---|---|
| I. Realizar una revisión preliminar de la información de los contenidos de aprendizaje de la materia de programación orientada a objetos y sus respectivas unidades para llevar a cabo la gamificación a través de estrategias lúdicas.<br><br>II. Diseñar un experimento sobre el desarrollo de una herramienta gamificada, para comparar el aumento de motivación y aprendizaje entre los grupos de estudiantes que utilicen gamificación y sin gamificación. | ¿Cómo mejora la motivación los estudiantes que aprenden programación utilizando herramientas gamificadas?<br>¿Cómo influye la tecnología en aplicaciones con gamificación en la asignatura de programación?<br>¿Por qué involucra a los estudiantes de pregrado en Ingeniería de Software e Ingeniería de Tecnologías de la Información de las Universidad de las Fuerzas Armadas ESPE, es la mejor opción para este tipo de investigación? |

| Objetivo específico | Pregunta de investigación |
|---|---|
| III. Ejecutar el experimento diseñado, siguiendo los parámetros establecidos para conseguir datos sobre la investigación y analizar los resultados de los sujetos de estudio. | ¿Cuáles son las principales amenazas a la validez que se pueden producir al realizar el experimento?<br>¿Es posible generar contingencias adecuadas para estas amenazas a la validez? |
| IV. Evaluar e interpretar los resultados de la propuesta, a partir de los datos recolectados y medir un aumento de aprendizaje y motivación en los estudiantes de programación orientada a objetos. | Durante la aplicación de gamificación en la enseñanza de la materia de programación orientada a objetos. ¿Qué mecanismos han sido utilizados para evaluar el aprendizaje de los estudiantes? |

**Hipótesis**

Se ha planteado la siguiente hipótesis de investigación:

Hipótesis de investigación: El uso de la gamificación como método de enseñanza y aprendizaje para comprender los cuatro pilares de programación orientada a objetos y los 3 primeros principios SOLID, mejora la motivación y el resultado de aprendizaje en estudiantes de carreras de Ciencias de Computación y afines.

**Estructura del trabajo**

El presente proyecto de titulación está estructurado por 5 capítulos los cuales abordan las siguientes temáticas: Capítulo II es el estado del arte relacionado a la gamificación y a las estrategias y metodologías de aprendizaje para los Principios de la Programación Orientada a Objetos, se lo realiza a través de una revisión de literatura con sus diferentes fases. El capítulo III presenta el modelo propuesto que detalla lineamientos del desarrollo de las herramientas con gamificación, diseño de la aplicación, herramientas que se utilizarán y el desarrollo del prototipo de la aplicación “Tetris” y “Chess”. El capítulo IV es el análisis e interpretación de los resultados del experimento, donde se compara el aumento de motivación y aprendizaje entre los grupos de

estudiantes que utilicen gamificación y sin gamificación. Finalmente, el capítulo V muestra las conclusiones, recomendaciones y los trabajos futuros que se pueden realizar a partir del presente proyecto.

**Desarrollo de la propuesta**

Una metodología de diseño de software puede estructurarse comprendiendo el componente del proceso de diseño de software y la representación del diseño de software o el componente de diagrama. El componente de proceso se basa en los principios básicos establecidos en la metodología, mientras que el componente de representación es el "plano" a partir del cual se construirá el código para el software. Cabe señalar que, en la práctica, la metodología de diseño a menudo se ve limitada por la configuración de hardware existente, el lenguaje de implementación, las estructuras de datos y archivos existentes y las prácticas existentes de la empresa, todo lo cual limitaría el espacio de solución disponible para desarrollar el software. La evolución de cada diseño de software debe registrarse o diagramarse meticulosamente, incluida la base de las elecciones realizadas, para futuros recorridos y mantenimiento.

# Capítulo II

## Estado del arte y marco teórico

### Introducción

En el presente trabajo, la revisión de literatura acerca de la gamificación se centra en la aplicación de elementos de juego a contextos educativos con el objetivo respaldar aquellos aspectos que permitan aumentar la participación de los estudiantes e inspirarlos a continuar aprendiendo.

Para el marco teórico es preciso establecer un grupo de definiciones y conceptos que permitan tener una perspectiva teórica que servirá como sustento para el constructo cognitivo del presente estudio. Se abordarán los temas relacionados con las variables dependientes e independientes de la hipótesis de investigación, que son: los principios de la programación orientada a objetos y la participación en el aprendizaje académico de la programación en contextos universitarios.

### Planteamiento de la revisión de literatura

De acuerdo al problema de investigación planteado, fue posible proporcionar un contexto para llevar a cabo la búsqueda de estudios científicos, definir un objetivo de búsqueda y formular preguntas de investigación. Para la revisión de literatura, se ha definido la búsqueda, selección, análisis e interpretación de artículos, investigaciones y estudios divulgados en bases de datos académicos y científicos considerados relevantes para la investigación. Estos son aquellos que consideran aspectos generales sobre la metodología aplicada en la gamificación y su éxito en evaluar y demostrar los resultados obtenidos. El grupo de control seleccionado y las palabras clave relevantes se muestran en la Tabla 2. Después de analizar los estudios que conforman el grupo de control, se seleccionaron las palabras más relevantes para construir la cadena de búsqueda.

**Tabla 2**

*Grupo de Control*

| Título | Cita | Palabras clave |
|---|---|---|
| Gamificación technique programming | (Carreño-León et al., 2018) | Teaching programming, teaching, challenges, motivation. |
| Gamification of Software Engineering Curriculum | (Uskov & Sekar, 2014) | Gamification, serious games, software engineering, curriculum. |
| Gamification in Software Engineering Education | (Ivanova et al., 2019) | Gamification, game-based learning, education, software engineering, higher education. |

**Construcción de la cadena de búsqueda**

A partir de las palabras clave que se obtuvieron de los artículos científicos del grupo de control, y después de realizar varias pruebas con diferentes combinaciones de las mismas, se formaron las siguientes cadenas de búsqueda utilizando operadores booleanos: (GAMIFICATION) AND (SOFTWARE ENGINEERING) AND ((TEACHING PROGRAMMING) OR (CURRICULUM)) y (GAMIFICATION) AND (GAME BASED LEARNING) AND (EDUCATION) AND (MOTIVATION).

La primera cadena estuvo enfocada a las principales causas de éxito de los proyectos de software y la segunda en los métodos empleados para aplicar la gamificación en materias de programación.

**Selección de estudios**

Para buscar información relevante sobre el uso de la gamificación en la enseñanza de la programación orientada a objetos, se realizó una búsqueda en la base de datos IEEE Xplore. Los resultados de la búsqueda incluyeron alrededor de 2.000 artículos relacionados, pero luego se aplicaron filtros para reducir el número de documentos a un tamaño manejable.

Los filtros utilizados incluyeron tipos de estudios como technical report, conference paper y journal paper, y un límite de fecha de publicación a partir de 2014. Con estos filtros, se encontraron alrededor de 75 artículos científicos relevantes que fueron revisados por los investigadores para elegir 7 estudios primarios para analizar el estado actual del arte, estos estudios se muestran en la Tabla 3.

**Tabla 3**

*Estudios primarios seleccionados*

| Código | Título | Cita |
|---|---|---|
| **EP1** | Tutorial on a Gamification Toolset for Improving Engagement of Students in Software Engineering Courses | (Vos et al., 2020) |
| **EP2** | Gamification Activities for Learning Visual Object-Oriented Programming | (Soepriyanto & Kuswandi, 2021) |
| **EP3** | A Gamification Technique for Motivating Students to Learn Code Readability in Software Engineering | (Mi et al., 2018) |
| **EP4** | Engaging students with a mobile game-based learning system in university education | (Bartel & Hagel, 2014) |
| **EP5** | Gamification in Software Engineering Education | (Ivanova et al., 2019) |
| **EP6** | A Gamification Approach to Improve Motivation on an Initial Programming Course | (Díaz et al., 2018) |
| **EP7** | Universidad de las Fuerzas Armadas – ESPE Carrera de Software ESPE Sede Matriz y Sede Latacunga | (*Ingeniería de Software*, 2020) |

A continuación, se presenta un análisis de cada uno de los estudios primarios que fueron seleccionados, para verificar el aporte que tuvo la aplicación de este método de aprendizaje.

EP1 (Vos et al., 2020): Tutorial on a Gamification Toolset for Improving Engagement of Students in Software Engineering Courses. El proyecto europeo Erasmus IMPRESS se creó para

explorar el uso de la gamificación en la enseñanza de la ingeniería de software a nivel universitario. El objetivo es desarrollar un conjunto de herramientas que ayuden a mejorar la participación de los estudiantes y por ende su apreciación de los temas que se enseñan, como exámenes y tutorías con relación a la tecnología de software.

EP2 (Soepriyanto & Kuswandi, 2021): Gamification Activities for Learning Visual Object-Oriented Programming. Soepriyanto analiza las dificultades que los estudiantes pueden encontrar durante el desarrollo programático de la unidad curricular académica de programación y ofrece una nueva forma de enseñar a los programadores que son nuevos en los juegos. Deliberó en experiencias para hacer que el módulo de programación sea fácil de usar y mejorar la tasa de éxito de los nuevos programadores. Este estudio proporciona aspectos teóricos del aprendizaje, métodos de enseñanza y programas de apoyo en detalle técnico.

EP3 (Mi et al., 2018): A Gamification Technique for Motivating Students to Learn Code Readability in Software Engineering. El estudio proporciona un modelo integral de incentivos y recompensas, así como un conjunto de motivaciones intrínsecas y extrínsecas específicas. Para garantizar su eficacia dinámica, se realizó una prueba de campo para comparar GamiCRS con sus contrapartes no gamificadas y evaluar los resultados del aprendizaje. Los resultados experimentales muestran un efecto positivo en la aplicación de GamiCRS en el entorno del aula. Dado que muchas actividades de aprendizaje de ingeniería de software a menudo son desafiantes y rara vez se disfrutan, la gamificación se puede utilizar como un complemento atractivo para respaldar una amplia variedad de estrategias de enseñanza. GamiCRS es un novedoso sistema de gamificación que tiene como objetivo ayudar a los estudiantes a comprender la legibilidad del código.

EP4 (Bartel & Hagel, 2014): Engaging students with a mobile game-based learning system in university education. Este referente teórico contribuye introduciendo el concepto de

aprendizaje basado en juegos, haciendo uso de dispositivos móviles. Se centra en la agradable estabilidad del conocimiento y la participación de los estudiantes, mediante el método de juego y la mecánica del juego. Los hallazgos anteriores sobre cómo mejorar la motivación de los estudiantes se adaptan y discuten en el contexto de la movilidad. Se describen las etapas de realización del concepto en la universidad.

EP5 (Ivanova et al., 2019): Gamification in Software Engineering Education. Este artículo describe los resultados empíricos de la aplicación de juegos de métodos de ingeniería de software y en la educación para involucrar y mejorar la motivación y participación de los estudiantes en el aula. Se describe un enfoque para incorporar juegos, para enseñar conceptos y métodos de ingeniería de software. Se prueban varios estilos de juegos a este respecto y se crean algunos más adecuados para los estudiantes. El campo de estudio está integrado en el curso de Ingeniería de Software. A través de ellos se muestran descripciones de los juegos y el progreso esperado del estudiante. Se implementa y describe un enfoque de equipo colaborativo y ágil. Se seleccionaron algunos grupos de estudiantes adecuados y se presentaron los resultados del experimento que muestran el enfoque positivo que adoptan los estudiantes para utilizar la gamificación. La estrategia de juego utilizada en el proceso de aprendizaje muestra que este método es eficaz, ya que se utiliza por tercer año consecutivo. En la encuesta de 2018, la mayoría de los estudiantes mostró que aprueban los juegos como una estrategia viable para mejorar el aprendizaje y prefieren estudiar mientras juegan.

EP6 (Díaz et al., 2018): A Gamification Approach to Improve Motivation on Programming Course. Estos autores describen una propuesta basada en juegos llamada Marketplace y su implementación en un curso formal de programación de educación en ingeniería en la Universidad de La Frontera. Usaron los dos primeros niveles del modelo de Kirkpatrick para evaluar la influencia y motivación de los estudiantes. Los resultados reafirman la idea de que el

uso de diferentes estrategias es valioso por ser beneficioso para el proceso de enseñanza y aprendizaje.

EP7 (*Ingeniería de Software*, 2020): Universidad de las Fuerzas Armadas – ESPE Carrera de Software ESPE Sede Matriz y Sede Latacunga. La asignatura de programación orientada a objetos con enfoque en la formación profesional caracteriza el aporte a la formación de elementos competenciales y el fortalecimiento de unidades en este sentido en el análisis, diseño y construcción de aplicaciones. El software basado en objetos, su base y principios orientados a modelos, como encapsulación, abstracción, herencia, polimorfismo, resuelven problemas complejos del mundo real y crean aplicaciones de alta calidad, haciendo uso de principios y prácticas, como pruebas de integridad y patrones de diseño.

**Carreras de Ingenierías de software y tecnologías de la información**

La Universidad de las Fuerzas Armadas ESPE, a través de su Departamento de Ciencias de la Computación, ofrece en su campus matriz en Sangolquí y en la provincia de Santo Domingo de los Tsáchilas y en otras áreas cercanas la carrera de Ingeniería en Tecnologías de la Información (TI). Esta carrera tiene una duración de ocho semestres y forma a Ingenieros en TI capaces de elegir, crear, aplicar, integrar y administrar eficientemente las tecnologías de la información con el objetivo de satisfacer las necesidades de los usuarios en un entorno social, organizacional y humanista (ACM, 2008).

La carrera tiene como objetivo transformar el modelo de desarrollo económico y social del país a través de la selección, creación, aplicación, integración y administración de tecnologías computacionales innovadoras, que contribuyan a la creación de una sociedad del conocimiento y a reducir la brecha digital al promover el acceso universal a la información. El currículo del Ingeniero en TI incluye itinerarios teóricos, metodológicos y prácticos que cubren todo el entorno de formación del profesional. Los itinerarios teóricos incluyen contenidos básicos y

profesionales que proporcionan al estudiante las herramientas teóricas básicas para comprender la base ingenieril de las TI desde la perspectiva de las matemáticas. Los itinerarios metodológicos y prácticos están compuestos por el resto de asignaturas, que están orientadas al desarrollo de las habilidades cognitivas del estudiante, basadas principalmente en el método científico y en la validación tecnológica a través de proyectos de investigación y vinculación con la sociedad.

En la unidad básica, se busca comprender los fundamentos de la investigación científica que permitan abordar los problemas de la ingeniería en tecnologías de la información a partir del diagnóstico de las realidades en su campo de actuación. En la unidad profesional, se busca analizar, desde el paradigma de la investigación-acción, fenómenos que ocurren en el campo de las tecnologías de la información para encontrar soluciones y mejores prácticas aplicables a diferentes contextos organizacionales. En la unidad de titulación, se busca vincular los aprendizajes relacionados con el perfil profesional, permitiendo demostrar el dominio de conocimientos y lecciones aprendidas a través de la aplicación de herramientas teórico-prácticas de la investigación científica, innovación, emprendimiento y desarrollo tecnológico dentro de su campo de estudio. También se espera que los estudiantes demuestren capacidades como solidaridad, cooperación, responsabilidad, autonomía, alta conciencia ciudadana y espíritu de innovación y creatividad

La Carrera de Software que es impartida en la matriz (Sangolquí) y en la sede Latacunga, tiene un programa de 8 semestres, que inicia en las Unidades Básicas con materias como: Cálculo Diferencial e Integral, Química, Fundamentos de la Programación y de la Ingeniería de Software; posteriormente con la profundización de materias de Cálculo Vectorial, Física, Programación Orientada a Objetos y Estructura de Datos. Adicionalmente, se integran algunas materias como: Cultura Ambiental, Realidad Nacional. A partir de la Unidad Profesional en el tercer semestre, se integran materias más avanzadas de Computación Digital, Programación

Web, Modelos de Desarrollo de Software, Bases de Datos, Computación Paralela. Las materias más avanzadas ven en los siguientes semestres. Para el último año se integran Aseguramiento de la Calidad de Software, Aplicación Distribuida, Arquitectura de Software, Gestión de Proyectos de Software (ESPE, 2021).

De esta manera, el profesional formado como ingeniero de software en la ESPE, ha estudiado el proceso completo de desarrollo para su aplicación práctica con diferentes enfoques y buenas prácticas, haciendo uso de los lenguajes de programación, las herramientas, los estándares, que sean aplicables a los sectores que así lo requieran (ESPE, 2021).

De acuerdo con la Universidad Madero Puebla -UMAD (2021), la ingeniería de software es una de las carreras altamente valoradas en el campo laboral, esto debido a la alta demanda existente en la actualidad y de los beneficios económicos percibidos para los profesionales en esta área. Especialmente para quienes posean altas capacidades de análisis, pensamiento lógico y matemático. Esta carrera permite hacer uso de las tecnologías de la información para el diseño, implementación, mantenimiento, de software en las organizaciones, incluyendo la posibilidad de ser consultores, trabajar de forma independiente, y desarrollar innovación.

Es importante dentro de este campo entender la importancia que representan las tecnologías de la información y comunicación, Estas tecnologías se pueden entender como el conjunto de innovaciones tecnológicas desarrolladas para la comunicación e interconexión instantánea, mediante sistemas de información computarizados que permiten el almacenamiento, recuperación, manipulación, interacción y transmisión de información. Las TIC se han desarrollado a la par del avance científico, siendo el computador el elemento más representativo de este ámbito y el internet como herramienta de comunicación, Las TIC favorecen varios ámbitos profesionales, prácticos y especialmente en la educación (Sánchez, 2017), por ello la importancia en el presente estudio.

## Características del estado del arte

Los juegos ofrecen un entorno atractivo para que los estudiantes aprendan de manera eficiente y mejoren su aprendizaje mientras superan los desafíos planteados en el juego. Diferentes investigadores han realizado estudios sobre diversas perspectivas en el aprendizaje basado en juegos y han señalado los problemas que encuentran los estudiantes en el aprendizaje de los cursos de programación tradicionales. Uno de sus principales hallazgos es la identificación de elementos de diseño de juegos y la participación de los estudiantes que deberían incluirse en el aprendizaje colaborativo en línea. Consideraron el trabajo en equipo, el apoyo del instructor y el compromiso personal de los estudiantes como elementos de participación.

Para la identificación de los elementos de diseño del juego, los autores concluyeron que existen factores que afectan el compromiso del estudiante: la demografía, los rasgos de personalidad, la asociación de tareas, el contexto temporal y espacial. Los estudios recomiendan que los juegos deberían tener un diseño controlado para medir el resultado, y deberían emplear más elementos sociales y de inmersión en sus soluciones. La importancia de la gamificación radica en su aplicación como una técnica de aprendizaje, que busca el desarrollo educativo-profesional a través de la mecánica de los juegos.

## La gamificación

### *Concepto de gamificación*

La gamificación es la aplicación de elementos de diseño de juegos y principios de juegos en contextos diferentes a estos. También se puede definir como un conjunto de actividades y procesos para resolver problemas utilizando o aplicando las características de los elementos lúdicos (King, Greaves, & Exeter, 2015). Los juegos y los elementos similares a estos se han utilizado para educar, entretener y participar durante miles de años. Algunos elementos clásicos

del juego son los puntos, insignias y tablas de clasificación. En los últimos años, la industria ha logrado considerables inversiones en estrategias para comprender qué motiva a la audiencia objetivo, así como la fidelidad de sus productos. Basado en esos resultados y la consecuente popularidad de los juegos, la idea de utilizar la estructura de los juegos (principios y mecánica) para motivar e involucrar a personas en diferentes áreas ha sido aceptada globalmente. Este proceso fue bautizado con el término “gamificación” (Ferreira, 2017).

La gamificación se ha convertido en un fenómeno emergente, demostrando ser una poderosa alternativa a los enfoques tradicionales en varios sectores, tales como salud, educación, políticas públicas y sociales, deportes; convirtiendo tareas repetitivas de la vida diaria en experiencias divertidas, ayudando de esta manera a las personas en el proceso de asimilación de nuevos conceptos y tecnologías y sobre todo, animando a las personas a adoptar nuevos comportamientos.

Entre los principales objetivos que tiene la gamificación se puede destacar la posibilidad de alentar a los usuarios a participar en actividades de poco interés; hacer la tecnología más atractiva; mostrar un camino más fascinante para el aprendizaje, favoreciendo de esta manera la solución de problemas sin distracciones, y aprovechando la predisposición psicológica que se obtiene a través de la participación en juegos (Buck, 2017).

En este orden de ideas, conceptualmente la gamificación se define como: "(...) el uso del pensamiento, la mecánica y el juego estética para involucrar a las personas, motivar acciones, promover el aprendizaje y la resolución problemas. " (Dicheva, Agre, & Angelova, 2015, pág. 78). La gamificación es el uso de principios y mecánicas de juegos para hacer actividades o rutinas poco atractivas más interesantes y divertidas. Además, la mecánica de los juegos incluye elementos que ayudan a lograr un equilibrio emocional y a aprovechar al máximo el tiempo, la energía y el intelecto de la persona (Aranda & Sánchez, 2015).

Entre los principales elementos considerados al diseñar juegos están: objetivos claros, reglas, narrativas, estructura recompensada, reconocimiento de niveles, y entornos de competencia y colaboración. El éxito de la gamificación radica en el resultado de la combinación efectiva de esos diferentes elementos (Aranda & Sánchez, 2015).

***Diseño de juegos***

Internet ha abierto un amplio portafolio de oportunidades para el aprendizaje, logrando el objetivo de aprender algo, de forma amena y motivada, a través de juegos didácticos virtuales. Entre los principales ámbitos que promueven este tipo de actividades están: el proceso de enseñanza-aprendizaje de habilidades, la autonomía, el pensamiento inductivo, la creatividad y el conocimiento especializado para que el alumno pueda aprender e investigar. Además, este tipo de actividad permite también autoevaluarse a su propio ritmo.

En este mismo orden de ideas, la propuesta de Werbach y Hunter (2012) sugiere que, para crear un juego exitoso, se deben introducir tres elementos básicos: 1) composición del juego, 2) mecánica y 3) dinámica. Sin embargo, antes de elegir estos elementos, se debe seguir seis pasos: 1) Definir el objetivo comercial, 2) Identificar los comportamientos deseados en el mercado objetivo, 3) Describir el tipo de jugador al que se dirigen, 4) Elegir las actividades a realizar, 5) Incluir diversión en las actividades y 6) Desarrollar las herramientas. Gracias a su estructura, esta metodología flexible se puede incorporar en la mayor parte de contextos, incluida la investigación, y el diseño de sistemas educativos de gamificación.

El desarrollo de juegos educativos para estudiantes universitarios es todavía un campo nuevo nuevo y tiene un largo camino por recorrer. La actividad lúdica no pretende ser una panacea en la enseñanza y el principal "agente" en este tipo de enseñanza es el docente o quien forma la herramienta de apoyo y acompañamiento académico, identificando y visualizando el contexto en el que el juego es pedagógicamente relevante. En un futuro próximo, se tendrá la

experiencia suficiente con estudios y datos reales, permitiendo determinar cómo utilizarlos y en qué condiciones se obtienen los mejores resultados, minimizando los diferentes problemas existentes. De ello pueden surgir técnicas que ayudarán a popularizarlos con la rigurosidad académica requerida para utilizar este tipo de herramientas en el contexto universitario.

***Mecánica del juego***

Las mecánicas del juego son las reglas y acciones permitidas o requeridas para que los jugadores interactúen. La mecánica incluye:

- Selección de turnos.
- Puntajes.
- Elementos del intercambio (por ejemplo, subasta y licitación).
- Lanzamiento de dados.
- Movimiento de piezas de juego sobre terreno virtual.
- Colocación de las piezas de juego) (Alsawaier, 2018).

Las mecánicas de juego se utilizan cada vez más en la educación para incentivar el aprendizaje, hacer que las lecciones sean divertidas y transmitir información de una manera que atraiga a los involucrados, por ejemplo, a los milenials (conocedores de la tecnología). Las aplicaciones educativas utilizan elementos de competencia, como por ejemplo la rivalidad con amigos para ganar puntos, esto fomenta el compromiso de los alumnos con el dominio de los materiales (Tsay & Kofinas, 2018). Existen numerosas ventajas en el uso de la mecánica del juego en la educación, por ejemplo:

- Progresión de objetivos no lineal: Los alumnos tienen varias formas de trazar un camino hasta el final de una lección, por lo que el contenido se siente atractivo en lugar de aburrido.

- Recompensar al esfuerzo y no solo al éxito: Dar a los alumnos de todos los niveles el incentivo para perseverar.
- Motivación entre compañeros: Los alumnos se animan entre sí para alcanzar los objetivos de aprendizaje (en un entorno de aprendizaje lúdico y basado en equipos) (Dicheva, Agre, & Angelova, 2015).

***Enfoques de participación gamificada***

En el mundo de los videojuegos y de las herramientas de gamificación, los diseñadores persuaden a los jugadores o participantes a regresar para cumplir y conquistar nuevos objetivos, fomentando el compromiso al uso de todas las opciones que el juego provee. Las características que se pueden utilizar para aumentar la participación en el aula gamificada incluyen:

- Recompensas desbloqueables.
- Incentivos sociales (como formar una liga y ascender a través de una tabla de líderes).
- Beneficios y recompensas vinculados a "subir de nivel" (alcanzar un nivel más alto de dominio) (Rodríguez & Santiago , 2015).

En este orden de ideas, se pueden incorporar estrategias de participación lúdicas en el aula, creando incentivos sociales a través del trabajo en grupo. También es posible estructurar el contenido con mayor grado de entretenimiento para los alumnos, a medida que alcancen niveles específicos de progreso.

***Contenido modificable***

En la industria del juego, se han desarrollado grandes comunidades en torno a juegos originales, comunidades que producen su propio contenido derivado. Esta cultura de "modding" de contenido se puede incorporar en el aula a medida que se brinda a los alumnos la oportunidad de crear sus propios diseños interactivos, que convierten los conceptos más recientes de las lecciones en juegos (Buck, 2017).

¡Un ejemplo de este enfoque es la aplicación Kahoot!, donde los usuarios pueden combinar preguntas de elección multipolar y agregar videos, imágenes y diagramas para ampliar la participación de los jugadores. Puede diseñarse su propio desafío o (si trabaja con alumnos mayores), hacer que los alumnos colaboren en la creación de sus propias lecciones utilizando los temas que se est en debate (Guayara, Cortés , & González , 2018).

***Afirmar el seguimiento del progreso***

En los modelos educativos tradicionales, la responsabilidad recae en el maestro para realizar un seguimiento e informar sobre el progreso de los estudiantes. Sin embargo, una de las grandes características de la gamificación es que los jugadores pueden rastrear y reforzar los sentimientos positivos sobre su propio progreso. Goalbook, por ejemplo, es una herramienta colaborativa de seguimiento del progreso, donde los profesores establecen tareas para los estudiantes y realizan un seguimiento de su progreso, al tiempo que les dan acceso a los recursos. Esto agiliza el seguimiento y la presentación de informes de los estudiantes al mismo tiempo que incentiva el aprendizaje (Rodríguez & Santiago, 2015).

El enfoque gamificado puede, por lo tanto, transformar los métodos de instrucción con un enfoque lúdico que aliente a los alumnos a colaborar, perseverar y enorgullecerse de su aporte, lo que fomentará su compromiso y a su vez mejorará el rendimiento de los estudiantes (Caffe, 2019).

***Tipos de Gamificación***

Un administrador de canales tiene la opción de gamificar sus programas a través de la gamificación estructural o de la gamificación de contenido para mejorar sus resultados. De acuerdo a Karl Kapp, profesor de tecnología educativa en la Universidad de Bloomsburg, estos dos tipos de gamificación son la clave para una mayor productividad y rentabilidad (Rodríguez & Santiago , 2015).

La *gamificación estructural* acontece cuando un programa tiene elementos de juego incluidos en la estructura del contenido, pero los dos no están relacionados. En otras palabras, la jugabilidad y el contenido de comunicación no están relacionados, esto permite una mayor flexibilidad del programa, una mayor eficiencia y menores costos, lo que incluye sus insignias, niveles más altos y medidores de progreso.

Un ejemplo, es el perfil de LinkedIn, que está ideado para que los usuarios se motiven para lograr completar el 100% del perfil, simplemente agregando elementos como un medidor de progreso y niveles, que son utilizados comúnmente para impulsar el compromiso y la motivación dentro de un programa (Alsawaier, 2018).

La *gamificación de contenido* sucede cuando los elementos reales del juego "divertidos" del programa educativo están relacionados con la aplicación. Estos también se conocen como "juegos serios" y, debido a la conexión de los objetivos de su programa con el resultado final del juego, son más especializados, requieren un mayor capital intelectual (para jugar y crear), por lo tanto, son menos flexibles y tienden a ser más costosos. Ejemplos comunes de estos incluyen: Farmville, Duolingo, que se usan comúnmente como un método de capacitación y/o aprendizaje como una forma de hacerlo más atractivo y divertido para el usuario como se muestra en la Figura 1 (Dicheva, Agre, & Angelova, 2015).

**Figura 1**

*Spin to Win*


*Nota.* Adaptado de "Gamification technique for teaching programming", por Carreño-León et al., 2018, *IEEE Global Engineering Education Conference (EDUCON)*, 2009-2014.

Por consiguiente, la gamificación de sus productos (promociones) solo funciona si el programa de su canal o los objetivos comerciales se identifican y se alinean con sus elementos de juego, el primer paso es identificar sus objetivos promocionales. A continuación, se presentan cuatro reglas a seguir para aprovechar de manera efectiva la gamificación en un canal de ventas:

Se debe proporcionar a sus participantes múltiples formas de ganar. Esto significa que debe incluir tanto las ganancias pequeñas / fáciles como las aspiracionales para que los usuarios mantengan su participación. Ganar con más frecuencia minimiza el esfuerzo cognitivo por parte del jugador, lo que hace que sea más probable que un usuario se involucre con su promoción. Un ejemplo de ello es WorkStride Spin-to-Win Figura 1, donde hay un gran premio y, también premios más pequeños frecuentes cada vez que un usuario elegible hace girar la rueda.

Se debe permitir que la mayoría de sus participantes ganen al menos una vez. Esto no significa que todo el mundo deba ganar un gran premio porque sería costoso e ineficiente. En su lugar, configure su juego de modo que los premios se distribuyan de acuerdo con el valor; por ejemplo, ciertas porciones pueden costar $ 5, $ 10 o $ 15, mientras que solo una porción vale $

100. Esto mantiene a los participantes del programa motivados y asegura que participen permanentemente, ya que saben que es probable que se les otorgue algún tipo de recompensa o premio.

Se debe garantizar que las reglas del juego se comuniquen y comprendan por el bien de todas las partes. La intención del programa debe comunicarse y ser fácilmente accesible dentro de sus promociones. Los malentendidos frustran y por lo general desvinculan y desmotivan a los jugadores. Esto se puede hacer a través de comunicaciones por correo electrónico, acceso a archivos PDF o páginas que describen los juegos, o dentro de los detalles del programa. El kit de herramientas de administración de contenido Figura 2 es un gran ejemplo de esto, porque permite a los administradores del programa agregar contenido que sea fácilmente accesible para todos los participantes del canal, lo que facilita la transparencia.

**Figura 2**

*Kit de herramientas para la gestión de contenidos*


*Nota*. Adaptado de "Tutorial on a Gamification Toolset for Improving Engagement of Students in Software Engineering Courses", por Vos et al., 2020, *IEEE 32nd Conference on Software Engineering Education and Training (CSEE&T)*, 1-3.

Se debe aprovechar las alianzas estratégicas para compensar los costos y aumentar los ingresos. Las asociaciones de marca compartida ofrecen la posibilidad de que organizaciones de diferentes tamaños trabajen juntas para lograr mayores resultados. Por ejemplo, considere un Spin-to-Win en el que cada tragamonedas de premios tiene diferentes recompensas valoradas. Es el caso por ejemplo que un giro puede ofrecer café en la cafetería local, una compañía diferente puede ofrecer una tarjeta de regalo de $ 100 como recompensa (Ortiz, 2019).

Aunque se pueden crear muchas mecánicas únicas para la gamificación, cómo se implementan muestra el enfoque adoptado. En la industria se reconocen dos enfoques clave para la gamificación: estructural y basado en contenido (Caffe, 2019).

***Gamificación superficial o de contenido***

La gamificación de contenido tiene como objetivo transformar el contenido de aprendizaje o trabajo existente y hacerlo más atractivo. Todo se convierte en parte de un juego, aunque el usuario sigue adquiriendo habilidades e información importantes y realiza tareas importantes de una manera no tradicional (Csikszentmihalyi, 2015).

Por ejemplo, un trabajador en una línea de montaje que debe documentar su productividad en una aplicación podría ver una interfaz divertida con cada unidad atendida representada por cocos o algo divertido en un juego profesional que imita el proceso de montaje (Buck, 2017). Entre las características típicas de la gamificación de contenidos están:

Texto adaptado: Una cosa que notará en la mayoría de las aplicaciones de contenido gamificado es que los textos son divertidos, breves, atractivos y a veces, incluso llenos de humor. Esto se logra al obtener los puntos clave del tema e insertándolos en un contexto y diseño completamente nuevos (Ferreira, 2017).

Nuevas estrategias: Después de la gamificación, se puede pedir a los usuarios que hagan cosas muy diferentes a las que hacían originalmente, incluida la participación en un juego (casual

o de aventura). En este nuevo entorno, el usuario aún aprende cosas y aplica sus conocimientos. Pero el proceso es más espontáneo e impredecible, por lo que debe aplicar nuevas estrategias y formas de pensar (Guayara, Cortés , & González , 2018).

Riesgo y aleatorización: Para aumentar la dinámica de su aplicación, muchos propietarios de negocios agregan elementos de aleatorización (como una lotería de recompensas), o corren el riesgo de no mantener a los jugadores alerta. Esto los entusiasma para seguir en la interacción con el software (Rodríguez & Santiago , 2015).

Diseño creativo: Normalmente se agrega una gran cantidad de contenido con este enfoque, lo que abre la puerta a un gran diseño creativo, incluso en cosas simples como la gamificación de perfiles. Por ejemplo, toda la aplicación puede convertirse en un juego de aventuras o puede que vea tareas serias entremezcladas con minijuegos, animaciones y demás (Rodríguez & Santiago , 2015).

Paso acelerado: A diferencia del temporizador del enfoque estructural, este está más centrado en un juego de ritmo rápido. Al poner a los usuarios en una situación en la que tienen que pensar rápido para tener éxito. Los desarrolladores también aumentan la probabilidad de que los usuarios aprendan las cosas "serias" que agregaron (Hoe, 2015).

Elementos de la historia: Una de las mejores formas en que se reinventan los materiales complejos y aburridos es a través de la narración. Esto los convierte en parte de una historia más amplia en la que encaja mejor y tienen un papel más claro (Guayara, Cortés , & González , 2018).

***Gamificación estructural o profunda***

La gamificación estructural busca añadir elementos dinámicos y novedosos a materiales/sistemas/procesos existentes. El contenido del juego se presenta de forma independiente del contenido serio, sin alterar los textos y la mecánica ya presentes en la aplicación (Hoe, 2015).

Por ejemplo, una aplicación de aprendizaje puede mantener los textos educativos, pero añadir una barra de progreso, minijuegos entre unidades y recompensas mientras el usuario avanza. Esto hace que el proceso de aprendizaje de los mismos materiales sea más atractivo (Ortiz, 2019). En este sentido se hace necesario conocer las siguientes características típicas de la gamificación estructural:

- Metas pasivas: Muchas aplicaciones que utilizan este enfoque tienen barras de progreso, contadores y otros indicadores que muestran el avance del usuario a través del material. Esto los mantiene enfocados en completar el proceso (Dicheva, Agre, & Angelova, 2015).
- Sistema de recompensas: Las insignias y logros suelen incluirse en el software para proporcionar una sensación de satisfacción y orgullo a los usuarios al completar una unidad, tarea u otras tareas (Caffe, 2019).
- Competencia entre usuarios: La competencia es una gran fuente de motivación, por lo que muchas aplicaciones estructurales permiten a los usuarios comparar su rendimiento con el de los demás, como la creación de una tabla de clasificación (Ferreira, 2017).
- Desbloqueo de contenido: Con el enfoque estructural, el contenido se puede dividir en niveles que solo se desbloquean cuando el usuario cumple con los objetivos. Esto añade un elemento de misterio sobre lo que viene a continuación (Guayara, Cortés , & González , 2018).
- Refuerzo positivo: No es raro que las aplicaciones incluyan texto motivacional e incluso personajes que las guíen a través del proceso de aprendizaje o trabajo (Ferreira, 2017).
- Límites de tiempo: Además de las mecánicas que hemos enumerado, algunos proveedores también agregan un temporizador a las unidades o secciones de su aplicación, lo que mantiene a los jugadores enfocados en su tarea y motivados para superar sus tiempos pasados (Rodríguez & Santiago, 2015).

De esta manera, con las diferentes estrategias utilizadas para gamificar la educación se pueden ir integrando temas serios y gran contenido con la facilidad que permite esta forma de aprendizaje, la cual se es aplicable al ámbito universitario, como se desarrolla en el siguiente apartado.

**La gamificación como estrategia educativa en el ámbito universitario**

La búsqueda constante de técnicas que puedan impulsar la motivación y el compromiso de los estudiantes en las diferentes modalidades académicas ha provocado la necesidad de innovaciones en el ámbito educativo. Para Buck (2017), el uso de elementos de juego en este contexto es el concepto conocido como gamificación, mismo que aparece como una alternativa para resolver este problema. La gamificación es la aplicación de elementos de juegos, como estrategias, rompecabezas y racionalización, fuera de su contexto lúdico, dentro de un escenario en el que se discute la centralidad del papel del docente. En el proceso de aprendizaje la gamificación en entornos educativos ha logrado la atención de diferentes autores.

Los juegos proporcionan complejos sistemas de reglas para que los jugadores exploren a través de la experimentación activa y el descubrimiento. En términos más generales, los juegos guían a los jugadores a través del proceso de dominio y los mantienen comprometidos con tareas potencialmente difíciles. Una técnica de diseño de juego fundamental es ofrecer desafíos concretos que se adapten perfectamente al nivel de habilidad del jugador, con un aumento de la dificultad a medida que la habilidad se expande. Los objetivos inmediatos, específicos, moderadamente difíciles son motivadores para los estudiantes y estos son precisamente los que proporcionan los juegos. En consecuencia, estos también brindan múltiples rutas hacia el éxito, lo que permite a los estudiantes elegir sus propios subobjetivos dentro de la tarea más amplia. Esto también apoya la motivación y el compromiso (Tsay & Kofinas, 2018).

**La gamificación en el aprendizaje de la programación en la ingeniería de software**

En los últimos años, la gamificación ha despertado el interés de académicos y profesionales en múltiples campos, especialmente en las tecnologías de la información, donde se ha utilizado en el desarrollo de sistemas y aplicaciones en diferentes campos y ha obtenido resultados positivos. De hecho, la experiencia práctica en el mercado laboral y las investigaciones reportadas en la literatura han confirmado que la gamificación es muy útil en sectores heterogéneos, especialmente en actividades colaborativas, en educación, en transporte urbano y en ingeniería de software (Soepriyanto & Kuswandi, 2021).

En las diferentes áreas mencionadas, un aspecto interesante relacionado con el juego radica en comprender los mecanismos que conducen al usuario a volver a la dinámica del sistema y a utilizar ciertos constructos (relevantes para el dominio de la aplicación) en el entorno, con la intención de impulsar el compromiso.

Por ello, surgen desafíos sobre cómo se aplica la gamificación en los planes de estudio universitarios. La base de la gamificación a la que se pueden aplicar diferentes mecanismos y dinámicas es la distribución de puntos. Existen herramientas donde los profesores a través del teléfono móvil pueden premiar a los alumnos en clase cuando realizan determinadas acciones, como participar en clase, ayudar a un compañero o dar otro punto de vista.

Los elementos comúnmente incorporados en escenarios educativos basados en juegos incluyen: insignias, puntajes, narración de historias, logros, niveles, barras de progreso y comentarios. Las tablas y los métodos de clasificación también se utilizan comúnmente para realizar un seguimiento del progreso del aprendizaje, así como un componente social donde los estudiantes o participantes pueden compartir sus logros con otros participantes (Kučak et al., 2021).

**Clase gamificada**

Gamificar una clase implica convertir el ambiente del aula y las actividades cotidianas en un juego. Esto requiere creatividad, colaboración y diversión. Hay muchas formas de llevar juegos al salón de clase para promover el aprendizaje y mejorar la comprensión de los estudiantes sobre la materia (Dicheva, Agre, & Angelova, 2015). Aquí hay cinco maneras de gamificar el aula para mejorar el compromiso, la colaboración y el aprendizaje en entornos de aprendizaje remoto, híbrido y en persona:

- Adaptar juegos tradicionales para su uso en el aula: La búsqueda del tesoro, el bingo, los juegos de dados, Connect Four y Scrabble han existido durante décadas y se pueden adaptar para el aprendizaje en el aula. Por ejemplo, se pueden colocar palabras del vocabulario en cartones de bingo y ver si los estudiantes pueden relacionar las palabras después de escuchar las definiciones. En el trabajo en equipo, los estudiantes pueden jugar Scrabble para deletrear respuestas a preguntas específicas de contenido. En un entorno de aprendizaje a distancia, se puede considerar una búsqueda del tesoro proporcionando a los estudiantes una lista de elementos para encontrar en su hogar y luego compartirlos en Zoom o durante una reunión de Google Meet. En un entorno híbrido, se pueden poner a los estudiantes en equipos usando Goose Chase, una aplicación dirigida a estudiantes y profesores de primaria a bachillerato que integra pruebas y actividades para encontrar el tesoro a través del teléfono celular. También se pueden crear búsquedas del tesoro digitales enviando a los estudiantes a tomar fotos en el área donde recibe clases, o creando un video o buscando una respuesta en línea relacionada con un tema específico.
- Jugar juegos digitales: A los estudiantes les encanta jugar en plataformas como Kahoot, Quizizz, Quizlet Live, Gimkit y juegos de trivia online más nuevos como Blooket. Estas plataformas gratuitas permiten a los profesores crear preguntas de opción múltiple que

los jugadores responden en sus propios dispositivos, además los profesores pueden elegir entre miles de cuestionarios ya compartidos en estos sitios o crear preguntas de contenido específico para usar como evaluaciones previas, cuestionarios o tickets de salida. Breakout EDU también tiene una colección de juegos digitales, acertijos y criptogramas que promueven el pensamiento crítico en el aprendizaje en línea.

- Crear una misión: Un juego de búsqueda es una actividad que tiene como objetivo alcanzar un destino claro o completar una tarea. Por ejemplo, los estudiantes pueden participar en una búsqueda de aventuras basada en la lectura de noticias actuales de la semana. Los estudiantes que respondan correctamente a preguntas específicas relacionadas con el texto reciben puntos. El estudiante con más puntos al final de seis semanas gana un premio. Además, se publican preguntas adicionales en plataformas como Remind y Twitter para permitir que los estudiantes obtengan puntos extra. Las misiones también pueden ser proyectos o actividades individuales para los estudiantes que han completado su trabajo principal.
- Lucha contra un jefe: En los juegos, un "jefe" es un antagonista al que el protagonista debe derrotar para lograr el éxito. Por ejemplo, en el juego Super Mario Bros, el héroe debe derrotar a un monstruo al final de cada nivel antes de avanzar al siguiente. En la plataforma de gamificación Classcraft, los profesores pueden crear sus propias batallas y misiones contra jefes utilizando preguntas de los contenidos de clase. Los profesores también pueden crear batallas contra jefes mediante el uso de herramientas como Google Forms o Google Slides, creando así su propio jefe ficticio único.
- Gana una insignia por dominio: Las insignias son reconocimientos que se otorgan a individuos que han demostrado dominio o logro en una habilidad o conocimiento específico. Las Girl Scouts y los Boy Scouts utilizan insignias para reconocer el dominio y los logros de sus miembros. De manera similar, los maestros pueden recompensar a los

> estudiantes por sus logros y dominio de un concepto, estándar o habilidad con insignias que van más allá de las calificaciones y que representan más que solo logros académicos. Los estudiantes pueden trabajar para completar diferentes insignias para demostrar su dominio de una habilidad o conocimiento. Estas insignias se pueden presentar digitalmente a través de Classbadges o se pueden mostrar para que todos las vean una vez que los estudiantes hayan obtenido una insignia específica (Hoe, 2015).

La gamificación consiste en transformar el entorno del aula y las actividades habituales en un juego. Requiere creatividad, colaboración y entretenimiento. Existen numerosas formas de llevar juegos al salón de clases para promover el aprendizaje y profundizar la comprensión de los estudiantes sobre la materia. Ya sea que los maestros estén en búsqueda de incorporar algún aspecto de los juegos a su clase o usar una plataforma de juegos en todo el plan de estudios, pueden usar elementos de gamificación para mejorar el aprendizaje y la participación de los estudiantes (Chan, Tan, & Hew, 2017). Es así que una vez entendido en tema es necesario adentrarse en las estrategias lúdicas de aprendizaje, mismo que se presenta a continuación.

**Estrategias lúdicas de aprendizaje**

Las estrategias de juego son actividades que incluyen juegos educativos, dinámicas de grupo, uso de juegos o juegos de mesa. Los maestros utilizan estas herramientas para reforzar el aprendizaje, el conocimiento y las habilidades de los estudiantes dentro o fuera del aula.

El juego es una forma de vida cotidiana, uno de sus objetivos es que el jugador sienta alegría y aprecie lo que está sucediendo. Al tratarse de un acto de gratificación física, mental o espiritual, las actividades lúdicas fomentan el desarrollo de habilidades, relaciones y sentido del humor.

De acuerdo a los fundamentos del constructivismo, los estudiantes no se limitan a copiar el conocimiento, sino que lo construyen a partir de factores personales, experiencias e ideas

potenciales, previamente ocultas, para estipular significado y representar nuevos conocimientos con significado adquirido. Como resultado, el rol del docente pasa de brindar conocimientos, a participar en la construcción del conocimiento con los estudiantes o como facilitador, por lo que es un conocimiento que se construye y de esta manera, se afianza con más firmeza en la mente del estudiante. (Zapata, 2015).

El constructivismo conduce al aprendizaje experiencial, con estudiantes y maestros involucrados en el proceso. Sin embargo, la relación que se crea con el juego es el experimento proporcionado y a partir del que se busca construir un proceso de aprendizaje. Por lo que el constructivismo es la teoría que mayor relación tiene con el juego (Zapata, 2015). De esta manera, el constructivismo se convierte en un actor primordial en estas nuevas experiencias. El alumno pasa de ser el receptor del acto de enseñar a convertirse en el protagonista, mientras que el docente actúa como mediador.

**La gamificación como estrategia metodológica de la acción del docente en el aula de educación universitaria**

La innovación educativa es una realidad que está presente en los espacios de aprendizaje. El uso de metodologías emergentes como la gamificación y el aprendizaje invertido han mostrado un gran potencial para mejorar el proceso de enseñanza y aprendizaje.

El cambio en la actualidad, se producen a una velocidad extrema, algo inusual en la historia de la humanidad que se evidencia en muchos niveles, con efecto en la mayoría de ellos en toda la sociedad. Entre estos avances, cabe destacar la irrupción de las tecnologías de la información y la comunicación (TIC), que se han convertido en parte de la evolución cotidiana de la sociedad. La educación, como valor fundamental del desarrollo humano, también se ve favorecida por el progreso e inclusión de estas herramientas, tanto para la docencia como para el aprendizaje de los estudiantes del siglo XXI. Estas acciones se llevan a cabo para actualizar los

procesos didácticos y convertirlos en innovadores, y adaptarlos a la vida habitual de los estudiantes (Pozo & López, 2020).

Las escuelas, en los últimos años, han desarrollado una transformación en la enseñanza, liderada por el uso de tecnología que está disponible para todos los miembros de la comunidad escolar. Por ello, se ha producido una mejora en la calidad de las acciones docentes, lo que se ha traducido en un incremento de valores como la motivación y, además, poniendo a disposición de todos, un amplio portafolio de recursos tecnológicos al servicio de la acción docente. Los estudiantes también ven incrementado su interés por la acción educativa, siempre que las tecnologías de la información y la comunicación (TIC) estén disponibles y se provoque su uso, lo que conduce a un mejor acceso a la formación y los contenidos (Garrote & Arenas, 2018).

Por tanto, se puede decir que las TIC son fundamentales en el desarrollo educativo en la actualidad. Concretamente en los procesos de enseñanza y aprendizaje que quieren adaptarse a los cambios de la sociedad actual. Además, no solo cambian los procesos, sino también los espacios dedicados al aprendizaje, lo que genera nuevas experiencias en torno a este tema (Khine & Ali, 2017). El objetivo de todos estos cambios tiene como eje fundamental la calidad, entendida como la adaptación de la educación a la era digital. Además, la aplicación de estas tecnologías da lugar a la aparición de nuevas metodologías y técnicas de enseñanza. Existen un sinfín de herramientas, tecnologías, estrategias, modelos, que se ajustan a lo que la sociedad, y en particular los estudiantes, demandan en estos momentos (Cabero & Barroso, 2018). Surge así, el aprendizaje volteado, que consiste en poner los contenidos a disposición de los estudiantes de forma audiovisual, para que puedan acceder a ellos y personalizar su aprendizaje en otros espacios fuera del ámbito escolar y antes de la sesión presencial, donde se trabajarán los contenidos desde una perspectiva más práctica (Pereira & Fillol, 2019).

**Beneficios de la gamificación en los estudiantes**

La gamificación implica el uso de elementos y técnicas propias de los juegos para motivar y fomentar el aprendizaje y la solución de problemas en las personas. Esto se logra a través de la inclusión de elementos como recompensas, desafíos y metas en actividades o tareas que de otra manera podrían considerarse poco atractivas. La gamificación puede ser una herramienta efectiva para involucrar y motivar a las personas y promover el aprendizaje y la resolución de problemas (Ferreira, 2017).

El desarrollo adecuado de la gamificación puede ayudar a los estudiantes a adquirir habilidades y conocimientos en cortos períodos de tiempo, haciendo que la tasa de retención de contenido sea efectiva (Rodríguez & Santiago , 2015). En este sentido, es un enfoque serio para acelerar la experiencia de la persona, favoreciendo el aprendizaje de contenidos y sistemas complejos.

Se reconoce que la gamificación se basa en teorías psicológicas que usan modelos motivacionales. Según Guayara, Cortés y González (2018) consideran que la estimulación en el acto de jugar cubre las siguientes áreas:

El área cognitiva denota autonomía en el individuo, una vez que este la usa con habilidad en diversas actividades. Por tanto, el acto de jugar involucra el área cognitiva del estudiante, mientras que su sistema establece un conjunto complejo de reglas orientadas desde pasos y tareas, desarrollándose como ciclos compuestos por experiencias cortas y rápidas. Este proceso basado en ensayo y error permite aumentar el nivel de habilidades en la persona, lo que facilita la solución de un problema en particular. Esto es posible, cuando el individuo hace lo que más le gusta, pues se encuentra en un estado de armonía entre su mente y cuerpo, con esto se alcanza, la motivación, el buen desempeño, la concentración, la participación y la energía necesaria al ejecutar una tarea (Csikszentmihalyi, 2015).

El área emocional denota la competencia individual y los enfoques principalmente sobre el concepto de éxito y fracaso. En tal sentido que, si el estudiante se encuentra en un entorno que fomenta el pensamiento como los juegos, estos poseen un complejo sistema de reglas que los individuos dominan y esta experiencia depende de la autonomía de él para lograrlo. Esto se puede apreciar cuando la tarea se completa de la manera esperada por el individuo, lo que resulta en una emoción positiva para él por el simple hecho de que el individuo ha superado una cierta dificultad. Sobre ello, Lanz (2010) manifiesta que el estudiante debe ser reconocido para que aumente los sentimientos positivos de éxito en la realización de tareas.

El área social es la estrategia que se utiliza, es constructiva y consiste en experiencias divertidas y estructuradas, permite elevar las relaciones positivas interpersonales de los participantes, de tal manera que el área social denota la relación, es decir, la interacción entre estudiantes mientras usan el juego con el alcance de la socialización como la colaboración y la competencia (Aranda & Sánchez, 2015). Desde un punto de vista social, estimula la competencia, lo que concibe resultados constructivos y destructivos.

En este orden de ideas, Rodríguez y Santiago (2015) señalan que como los límites entre estas áreas no están definidos, en general, la mecánica y la dinámica utilizadas en el proceso de gamificación abarca todo al mismo tiempo como, por ejemplo: Dentro de un juego, mucho de lo que logra el jugador es clave para nuevos comienzos, para ciclos con mayor dificultad y con una mayor complejidad. Esto impacta emocional y cognitivamente al individuo. Cuando una tarea requiere cooperación entre jugadores para la solución de un problema, este se ve afectado en el área social, pero siempre relacionado con el área cognitiva o emocional, especialmente cuando la recompensa se refiere al estado social del jugador (estudiante).

**Principios que desarrolla la gamificación en los estudiantes**

Según Hoe (2015), una actividad de aprendizaje gamificada permite a los estudiantes adquirir conocimientos, perfeccionar habilidades y fomentar rasgos positivos a través del juego construido específicamente con el propósito de aprender. En otras palabras, la gamificación en las actividades de aprendizaje es un proceso centrado en el estudiante (Dicheva, Agre, & Angelova, 2015). Esta estrategia se ha aplicado en todos los niveles de educación, desde el nivel escolar hasta el nivel de educación superior, aunque a menudo a los juegos en general se les concede ciertas connotaciones negativas por la sociedad, especialmente por parte de los padres de los estudiantes. Sin embargo, la reputación de la gamificación y el aprendizaje basado en juegos permanece intacta (Hishamuddin & Rahman, 2018).

En consecuencia, no es de extrañar que la gamificación sea más adecuada para ser practicada en el escenario de aprendizaje actual, que ya no está centrado en el maestro sino más bien pone énfasis en las actividades dinámicas de los estudiantes (Tsay & Kofinas, 2018). Por lo tanto, aquí es donde la ventaja de la gamificación es prominente, porque este método puede proporcionar diversas formas de estímulos a los estudiantes, especialmente para involucrarlos y motivarlos a mantenerse activos mientras estudian. Como explica Alsawaier (2018), la gamificación tiene una relación significativa con la motivación y el compromiso de los estudiantes en el aprendizaje, por tanto, también tiene un impacto positivo en el su rendimiento, lo que también está respaldado por el autor Chan et al. (2017). Esto se debe a que la gamificación contiene tipos de elementos que intentan dar un impulso a los estudiantes, por ejemplo, desafíos, recompensas, puntos, niveles, opciones e insignias (Hoe, 2015).

**Dificultades de aprendizaje de programación**

La alta tasa de deserción y la tasa de fallas en las asignaturas de programación han llamado seriamente la atención de los investigadores para averiguar las causas y soluciones.

Moser (1997) mencionó que la programación es una habilidad de múltiples capas que es aburrida, intimidante y no está relacionada con gran parte de la experiencia del día a día, donde los estudiantes solo aprenden en un contexto único. Los principiantes comienzan a aprender a programar en un solo contexto antes de aprender la estructura y el estilo. Esto puede conducir a un hábito de programación negativo que puede afectar la flexibilidad de aprender otro lenguaje de programación en un contexto diferente (Phit Huan & Ting Choo, 2015).

La percepción de los estudiantes universitarios sobre los problemas de aprendizaje y las sugerencias para mejorar el método de enseñanza, podrían ser útiles para identificar un enfoque alternativo. Para ello, es importante resaltar las causas que llevan a los estudiantes a tener un desempeño deficiente en el aprendizaje de la programación. Se han realizado varios estudios, algunos de los más relevantes, se presentan a continuación.

Un estudio de (Milne & Rowe, 2002) reveló que el aprendizaje de la programación ha sido difícil debido a la falta de comprensión de los estudiantes sobre cómo se ejecuta un programa. Como tal, se sugiere que una herramienta de visualización de programas sea la solución para ayudar y mejorar la comprensión del programador de lo que sucede en la memoria a medida que se ejecuta su programa. Por otro lado, (Jenkins, 2001), (Pozo & López, 2020) y (Lahtinen et al., 2005) consideran que una de las soluciones efectivas para ayudar en la enseñanza y el aprendizaje de asignaturas de programación es a través de la práctica.

El juego puede ser otro método de enseñanza convincente, que tiene un entorno de aprendizaje muy propicio. Este entorno permite la práctica de la programación; y proporciona varios ejemplos, comentarios, sugerencias, interacción, y progreso académico en materias de programación de enseñanza y aprendizaje (Alsawaier, 2018). Se realizó un estudio empírico haciendo uso de un juego educativo conocido como CeeBot-4, para determinar los componentes

del diseño del juego que contribuyen a promover el interés, la motivación y la diversión entre los estudiantes universitarios para aprender a programar.

Para Gomes y Mendes (2021), la programación es una asignatura muy compleja que requiere esfuerzo y un enfoque especial en la forma en que se aprende y se enseña. Para convertirse en un buen programador, el alumno debe adquirir una serie de habilidades que van mucho más allá de conocer la sintaxis de algún lenguaje de programación. Se han propuesto varios enfoques y herramientas con el objetivo de apoyar el aprendizaje de la programación de diferentes maneras. Aunque se encuentran informes de resultados positivos como resultado del uso de algunas herramientas, ninguna de ellas tiene un uso generalizado. De hecho, el problema permanece relativamente sin cambios a medida que se continúa con la realización de informes sobre las dificultades que experimentan muchos estudiantes al aprender programación básica.

La experiencia generalizada muestra que el problema comienza para muchos estudiantes en la fase inicial de aprendizaje, cuando tienen que comprender y aplicar conceptos abstractos de programación, como estructuras de control para crear algoritmos que resuelvan problemas concretos. Es necesaria una especial atención en esta etapa inicial, no solo en el desarrollo de habilidades específicas de programación, sino también, y quizás sobre todo en la mejora y/o consolidación de conocimientos y habilidades que deberían haber sido adquiridas en años anteriores, estos incluyen habilidades genéricas de resolución de problemas y de razonamiento lógico (Delgado & Xexeo, 2016).

**Análisis de deserción estudiantil**

En Ecuador, una de las principales preocupaciones del sistema educativo superior es la tasa de abandono estudiantil. Un número considerable de estudiantes deja de asistir a clases y no completan sus estudios, especialmente durante los primeros semestres de los programas académicos. Esto tiene un impacto negativo en el nivel educativo y profesional del país. Se

estima que el 37,50 % de los alumnos que ingresan a las universidades en la carrera de Ingeniería en Sistemas termina en deserción (Verdesoto et al., 2018), además, identifica que uno de los factores que inciden en la deserción, es el poco interés por los estudios en general, por la carrera y por la institución en la que estudia. Es fundamental reducir estos índices mediante la implementación de técnicas que permitan aumentar el interés de los estudiantes en la continuidad de sus estudios en ingeniería, especialmente en las disciplinas vinculadas a la informática.

La deserción universitaria es un fenómeno importante, por lo que se ha convertido en un problema social, poniendo límite a la capacidad de desarrollo humano, relacionado con aspectos económicos, personales, institucionales y sociales. Entre los posibles factores de riesgo para la deserción universitaria, se destacan tres aspectos (Verdesoto et al., 2018):

- Académico, relacionado con las variables: malos resultados, no cumplir con los requisitos laborales, pedagogía desfavorable de los docentes, dificultad para cubrir las necesidades de aprendizaje.
- Socioeconómicos, asociado a bajos ingresos familiares y situación laboral desfavorable.
- Institucional, dada la baja disponibilidad de becas y créditos universitarios.

Uno de los cursos que siempre está presente en la malla curricular de las carreras de las áreas de Tecnologías de Información, de Ingeniería de Software o de Ciencias de la Computación, es la programación. En general el aprendizaje de este curso o similares ha sido catalogado desde la percepción de los alumnos como complicado, tedioso y de difícil aprendizaje. Desde el punto de vista del docente, los alumnos tienen dificultades para asimilar los contenidos del curso y para realizar las tareas o los ejercicios que se desarrollan en clases.

**Métodos de enseñanza efectivos**

El aula es un entorno dinámico, que reúne a estudiantes de diferentes orígenes, con diversas habilidades y personalidades, por lo tanto, ser un maestro eficaz requiere la implementación de estrategias de enseñanza creativas e innovadoras para satisfacer las necesidades individuales de los estudiantes (Cuevas & Feliciano, 2019).

Es importante tener en cuenta que, independientemente del tiempo de experiencia de un maestro en el proceso enseñanza-aprendizaje, encontrar la mejor manera de motivar a sus estudiantes siempre es una necesidad y un desafío. Como profesor, es importante tener en cuenta que no hay una respuesta única para todos, por lo que es útil considerar diferentes estrategias de enseñanza efectivas que se puedan utilizar para fomentar el aprendizaje en el aula (Bergmann & Sams, 2016).

**Estrategias de enseñanza eficaces para el aula**

Visualización: Es importante enriquecer la educación de los estudiantes con actividades visuales y prácticas que hagan más interesantes los conceptos académicos que pueden ser aburridos. De esta manera, se puede ayudar a los estudiantes a comprender cómo se relaciona su educación con la realidad y cómo pueden aplicarla en el mundo real (Delgado & Xexeo, 2016). Los ejemplos incluyen el uso de la pizarra digital interactiva para mostrar fotos, clips de audio y videos, así como alentar a sus estudiantes a levantarse de sus asientos con experimentos en el aula y excursiones locales.

Aprendizaje cooperativo: Es importante motivar a los estudiantes con habilidades mixtas a colaborar, ya sea en grupos pequeños o en actividades de clase completa, para mejorar su comprensión y aprendizaje (Csikszentmihalyi, 2015). Al participar en la discusión de ideas y responder a los demás, los estudiantes pueden aumentar su confianza en sí mismos y mejorar sus habilidades de comunicación y pensamiento crítico’; habilidades esenciales a lo largo de toda

la vida. Es importante involucrar a los estudiantes en actividades que fomenten el aprendizaje cooperativo, como resolver problemas matemáticos, realizar experimentos científicos o representar breves obras teatrales. Estos ejemplos solo son algunas formas en que se puede integrar el aprendizaje cooperativo en las lecciones en el aula.

Instrucción basada en consultas: Es importante animar a los estudiantes a cuestionar y explorar sus propias ideas, ya que esto puede ayudarles a mejorar sus habilidades de resolución de problemas y obtener una comprensión más profunda de los conceptos académicos. Estas habilidades son vitales para el éxito, no únicamente profesional, sino también en la vida.

Diferenciación: Es importante adaptar la enseñanza a las necesidades y habilidades individuales de los estudiantes, para asegurar que todos avancen y nadie se quede atrás. Esto puede incluir la asignación de tareas y actividades que varíen en complejidad de acuerdo con las necesidades de aprendizaje de cada estudiante. Por ejemplo, se podría entregar hojas de trabajo de diferentes niveles de dificultad a diferentes grupos o crear estaciones de trabajo con una variedad de tareas para que los estudiantes elijan (Hoe, 2015). También se pueden utilizar herramientas educativas como Quizalize, que agrupan automáticamente a los estudiantes para que puedan identificar fácilmente las brechas de aprendizaje individuales y de toda la clase (Bergmann & Sams, 2016). Esto puede ayudar a ahorrar tiempo y esfuerzo en la identificación de necesidades individuales de aprendizaje.

La incorporación de tecnología en la enseñanza es una excelente manera de involucrar a los estudiantes de manera activa, especialmente en un mundo donde los medios digitales están presentes en la vida de los jóvenes (Cuevas & Feliciano, 2019). Se pueden utilizar pizarras interactivas o dispositivos móviles para mostrar imágenes y videos que ayuden a los estudiantes a visualizar nuevos conceptos académicos. El aprendizaje puede ser más interactivo y participativo cuando se usa la tecnología, ya que los estudiantes pueden investigar sus ideas de

manera inmediata y desarrollar su autonomía (Delgado & Xexeo, 2016). Los dispositivos móviles, como iPads y tabletas, pueden usarse en el aula para que los estudiantes registren resultados, tomen fotos o videos, o incluso como una herramienta de gestión del comportamiento. Además, incorporar programas educativos como Quizalize en los planes de lección también puede hacer que las evaluaciones formativas sean divertidas y atractivas.

Manejo de la conducta: Es esencial implementar una estrategia efectiva de gestión del comportamiento para ganarse el respeto de los estudiantes y asegurar que tengan las mismas oportunidades de alcanzar su máximo potencial (Rodríguez & Santiago, 2015). Un ambiente de clase ruidoso y perturbador no promueve un entorno de aprendizaje productivo. Crear una atmósfera de respeto mutuo a través de una combinación de disciplina y recompensa puede ser beneficioso tanto para el profesor como para los estudiantes. Algunas opciones pueden incluir tablas de recompensas interactivas y divertidas para estudiantes más jóvenes, donde los estudiantes suben o bajan según su comportamiento y el mejor estudiante recibe un premio al final de la semana. El "tiempo de oro" también puede ser efectivo para estudiantes de todas las edades, donde se les ofrece la opción de elegir entre varias actividades como recompensa por su arduo trabajo, estas actividades podrían ser juegos o inclusive se puede eliminar las tareas de casa (Tsay & Kofinas, 2018).

Desarrollo profesional: Participar en programas regulares de desarrollo profesional es una excelente manera de mejorar la enseñanza y el aprendizaje en su salón de clases (Cabero & Barroso, 2018). Con las políticas educativas en constante cambio, es extremadamente útil asistir a eventos en los que puede inspirarse en otros profesores y académicos. Las sesiones pueden incluir aprendizaje sobre nuevas tecnologías educativas, capacitación en seguridad en línea, consejos sobre cómo utilizar a los asistentes de enseñanza y mucho más.

Ser un maestro eficaz es un desafío porque cada estudiante es único; sin embargo, al usar una combinación de estrategias de enseñanza, se puede abordar los diferentes estilos de aprendizaje y las capacidades académicas de los estudiantes, así como hacer de su salón de clases un entorno dinámico y motivador para los estudiantes (Bergmann & Sams, 2016).

## Metodología de investigación

Al realizar una investigación, es importante definir claramente sobre qué base se pretende abordar el problema y obtener los resultados. Como tal, la investigación es un conjunto de métodos que se aplican para obtener un conocimiento profundo de una pregunta planteada o un problema determinado para generar nuevos conocimientos en el campo al que se aplica. Existen varios tipos de investigación que se clasifican según el objetivo, la profundidad, los datos analizados, el tiempo necesario para estudiar el fenómeno y otros factores.

### *Método deductivo*

Este aspecto metodológico está relacionado con extraer conclusiones lógicas y validas a partir de un conjunto de premisas o una serie de proposiciones verdaderas, va de lo general a lo particular. Es decir, el enfoque parte de una declaración general de la que se despliegan partes o elementos específicos.

### *Método inductivo*

A diferencia de la inferencia, este enfoque va de casos especiales a generalizaciones. En tal sentido, la información parte de datos o elementos individuales que, por similitudes, se agregan y llegan a un enunciado común que explica y comprende estos casos particulares.

### *Método histórico*

Este método tiene como objetivo pasar del pasado al presente para proyectarse hacia el futuro. Es decir, permite examinar los hechos del pasado con el objetivo de encontrar

explicaciones causales para las manifestaciones actuales de las sociedades o de reconstruir el pasado de la manera más precisa y objetiva posible.

***Método descriptivo***

Se refiere a una orientación que se centra en responder a la pregunta de cómo es una determinada parte de la realidad que se estudia.

***Método explicativo***

Siguiendo este método, se intenta, además de considerar la respuesta a "¿cómo?" se centró en responder la pregunta de "¿por qué la realidad es así?" ", O" ¿Cuál es la causa? ". Esto implica tanto la propuesta de hipótesis explicativas como una estimación explicativa.

***Método experimental***

Es una orientación, basada en lo descrito y explicado, enfocada a predecir lo que sucederá en el futuro si, en esta situación real, se realiza algún cambio basándose en la respuesta a "¿cómo?" "¿Y por qué?, por lo que, como premisa se afirma que, si se hacen cambios, sucederá lo mismo.

**Metodologías de desarrollo de software**

Es conocido que muchos proyectos de desarrollo de software incurren en errores de diseño al inicio del proceso de desarrollo, que resultan muy extensos y costosos (Delgado & Xexeo, 2016). Esta realidad, resalta la necesidad de una mejor definición de requisitos y metodología de diseño de software, puesto que esta es una actividad importante que determina cómo se desarrollaría todo este proceso incluido el mantenimiento del sistema.

El diseño de software es esencialmente una práctica ejecutada por profesionales del área que generalmente requiere una estructura metodológica para esta tarea. Una metodología se puede definir como el conjunto de principios y reglas subyacentes que gobiernan un sistema, mientras que un método puede definirse como un procedimiento sistemático para un conjunto

de actividades. Diferentes metodologías pueden apoyar el trabajo en diversas fases del ciclo de vida del sistema, por ejemplo, planificación, análisis, diseño, programación, prueba e implementación. A este respecto Svoboda (1990) desarrolló la idea de que una metodología debería considerar al menos cuatro componentes:

- Un modelo conceptual de constructos esenciales para el problema
- Un conjunto de procedimientos que sugieren la dirección y el orden a seguir
- Una serie de pautas que identifican las cosas que deben evitarse
- Una colección de criterios de evaluación para evaluar la calidad del producto (Caffe, 2019).

El modelo conceptual es necesario para dirigir o guiar a los diseñadores hacia los aspectos relevantes del sistema, por ello, todos los procedimientos proporcionan al diseñador un conjunto sistemático y lógico de actividades para comenzar el proceso del diseño.

Una metodología de diseño de software puede estructurarse comprendiendo el componente del proceso de su diseño y la representación de este o el componente de diagrama. El componente de proceso se basa en los principios básicos establecidos en la metodología, mientras que el componente de representación es el "plano" a partir del que se construirá el código para el software. Cabe señalar que, en la práctica, la metodología de diseño a menudo se ve limitada por la configuración de hardware existente, el lenguaje de implementación, las estructuras de datos, los archivos existentes y las prácticas propias de la empresa, lo que limitaría el espacio de solución disponible para desarrollar el software, por tanto, la evolución de cada diseño de este debe registrarse o diagramarse meticulosamente, incluida la base de las elecciones realizadas, para futuros recorridos y mantenimiento (Bergmann & Sams, 2016).

Con el tiempo, las metodologías de desarrollo de software han evolucionado desde un proceso organizacional simple hasta convertirse en una base fundamental para el desarrollo de

software eficiente y productivo. En las últimas décadas, las metodologías de desarrollo de software ágil predominan sobre otros métodos tradicionales, por ejemplo, el último Project Management Institute (PMI) muestra que, en Francia el 71% de las empresas de desarrollo utilizan métodos ágiles de ingeniería de software. Por tanto, para dedicarse a una carrera profesional en el desarrollo de software, es necesario conocer cómo funcionan los métodos ágiles.

Es importante mencionar que algunas empresas todavía utilizan metodologías tradicionales de desarrollo de software, que pueden ser adecuadas según el proyecto y la empresa. En cualquier caso, el uso de metodologías de desarrollo es esencial en la programación y otros campos para crear productos de software de alta calidad.

Cuando se trata de desarrollar un producto o solución para un cliente o mercado específico, hay que tener en cuenta factores como el costo, la planificación, la dificultad, el personal disponible y los idiomas. El uso de una metodología de desarrollo de software junto con estos factores ayuda a organizar el trabajo de la manera más ordenada posible. El desarrollo de software puede ser una industria particularmente compleja, especialmente cuando se trata de grandes equipos y aplicaciones. Por tanto, iniciar un proyecto de desarrollo sin una metodología clara puede llevar a un proceso más complicado, con problemas, retrasos, errores y, en última instancia, un resultado inferior. Trabajar con una metodología de desarrollo de software reduce el nivel de dificultad, organiza las tareas, agiliza los procesos y mejora el resultado final de las aplicaciones a desarrollar.

Las metodologías tradicionales de desarrollo se caracterizan por una definición completa y rígida de los requisitos al inicio de los proyectos de ingeniería de software. Los ciclos de desarrollo son inflexibles y es difícil manejar los cambios, a diferencia de las metodologías ágiles.

La organización del trabajo en las metodologías tradicionales es lineal, lo que significa que algunas fases no pueden comenzar hasta que se haya completado la anterior. Regresar a un proceso o etapa anterior es muy costoso. Estas metodologías no se adaptan bien al cambio, que es constante en el mundo actual. Según, Maida & Pacienzia (2015) las principales metodologías tradicionales o clásicas son:

- Waterfall
- Prototipado
- Espiral
- Incremental
- Diseño rápido de aplicaciones (RAD)

Las metodologías ágiles se basan en un enfoque incremental que implica agregar nuevas funcionalidades a la aplicación al final de cada ciclo de desarrollo más corto y rápido. Este método permite formar equipos de trabajo autónomo e independiente que se reúnen regularmente para compartir información y experiencias. Poco a poco se va construyendo y perfeccionando el producto final, y el cliente puede revisar el software mientras el equipo de desarrollo ha implementado un requerimiento determinado. Además, el cliente puede verificar el progreso del proyecto en tiempo real y solicitar correcciones o validar y mejorar los requerimientos si es necesario. Las principales metodologías ágiles son:

- Scrum
- Kanban
- Lean
- Programación extrema (XP)

### *Metodología SCRUM*

Al ser una metodología Ágil, el proceso SCRUM emplea un conjunto de roles y prácticas que aportan al desarrollo del software e intervienen de forma eficiente en los procesos. En este sentido, la metodología es flexible por que trabaja en la aplicación de los 12 principios agiles.

Para la ejecución de SCRUM, se trabaja con bloques temporales, cortos y periódicos más conocidos como Sprints. Los Sprints son entidades que permiten un resultado completo en plazos que oscilan entre 2 a 4 semanas, tiempo en que se contribuye a la retroalimentación y la reflexión esenciales para modificaciones o cambios en el producto final que se entrega a los clientes.

Emplear esta metodología es complejo, requiere del conocimiento y de la experiencia. Los procesos deben estar alineados a un punto de partida que no es más que la lista de objetivos que serán esenciales para el plan del proyecto, para lo cual se considera el equilibrio entre su valor y su financiamiento. Facilitando el desarrollo de las interacciones y entregas (Subra & Vannieuwenhuyse, 2018).

### *Concepto Scrum*

El método Scrum se fundamentó en el desarrollo de ciclos cortos e iterativos que permiten cambios en los requisitos, dándole relevancia e interés a la producción de valor y al mejoramiento continuo que cada vez se adapta a las necesidades de los clientes. Las interacciones son funcionales, facilitando el manejo de la información (Sutherland, 2016).

La ventaja de utilizar este método es que permite detectar con mayor facilidad errores en la aplicación y en la ejecución de tareas. Situación que requiere de constantes cambios y modificaciones que van acorde a los problemas que se presentan (Heredero, López, Martín-Romo, & Medina, 2019).

***Diseño orientado a objetos***

Scrum es uno de los métodos Agiles que aporta al modelado de objetos, trabaja en el diseño de la interfaz de usuario y el software al igual que el comportamiento de sistemas (DAMA International, 2017). En este sentido, se establece un control orientado a procesos empíricos para lograr el desarrollo iterativo e incremental, el cual permite un mejoramiento continuo a los productos, logrando un trabajo optimizado y flexible.

El diseño orientado a objetos está en constante evolución. La Orientación a Objetos es una de las técnicas más utilizadas en el modelado de software y se debe dar la importancia y relevancia a sus principios, los cuales proveen los pilares de la programación orientada objetos, dado que su finalidad es proporcionar con soluciones concretas (Boulet, 2017).

La orientación a objetos es una técnica de modelado de software a gran escala, que sugiere tratar la complejidad del sistema software de manera análoga a la realidad, a como los seres humanos tratamos la complejidad: descomponemos el problema en pequeñas piezas con un comportamiento propio que sirven para resolver el problema. La orientación a objetos centra su atención en el problema, en lugar de la solución. El análisis del problema es el que conduce a una solución (Jiménez, 2008, pág. 25).

De esta manera, la estructuración de los sistemas y el desarrollo de alternativas para un determinado servicio se adaptarán a las necesidades de los humanos para brindar una solución a sus problemas, “cada objeto proporciona un conjunto de servicios o ejecuta acciones que los otros miembros de la comunidad usan” (Becker, 2001, pág. 50), es decir, se proyectan características humanas a los objetos para proveer soluciones a los problemas mediante tareas. Los objetos se comunican entre ellos y saben cómo comunicarse a través de la programación.

***Principios de la Programación Orientada a Objetos (POO): encapsulamiento, polimorfismo, abstracción, herencia, SOLID (5)***

La programación orientada a objetos se centra en los objetos que el programador desea manipular en lugar de la lógica necesaria para manipularlos. Este paradigma de programación es apropiado para programas grandes y complejos, así como para actualizaciones y mantenimiento (Sarcar, 2021) .

El hecho de que un programa orientado a objetos esté organizado también hace que este enfoque sea beneficioso para el desarrollo en equipo, donde varios grupos trabajan en proyectos conjuntos.

Entre los beneficios de la programación orientada a objetos se encuentran la reutilización, la escalabilidad y la eficiencia del código. Incluso cuando se utilizan microservicios, los desarrolladores pueden aplicar los principios de la programación orientada a objetos, los cuales se presentan a continuación:

- Encapsulamiento: Los objetos mantienen su implementación y estado dentro de una clase determinada, que es inaccesible a otros objetos. Sin embargo, estos objetos externos pueden invocar funciones o métodos públicos (abstracción). Esta ocultación de datos brinda mayor seguridad al programa y evita la corrupción involuntaria de los datos.
- Abstracción: Los objetos muestran solo los mecanismos internos necesarios para su uso por otros, ocultando el código de implementación. Este enfoque facilita a los desarrolladores la modificación y el agregado de características a lo largo del tiempo.
- Herencia: Es posible establecer relaciones y subclases entre objetos, lo que permite a los desarrolladores reutilizar la lógica y atributos comunes mientras mantienen una jerarquía única. Esta característica de la POO requiere un análisis de datos más

detallado, lo que reduce el tiempo de desarrollo y garantiza un mayor grado de precisión.

- Polimorfismo: Los objetos pueden adoptar diferentes formas según el contexto. El programa determina el significado o uso necesario para cada ejecución del paradigma, lo que reduce la necesidad de tener código duplicado.
- SOLID (5): Si se habla de diseño y desarrollo de aplicaciones, los principios SOLID son algunos de los términos que se deben conocer como una de las bases de la arquitectura y el desarrollo de software orientado a objetos.

  SOLID es un acrónimo acuñado por Michael Feathers, basado en los principios de la POO que Robert C. Martin compiló en 2000 en el artículo "Principios de diseño y patrones de diseño". Los principios se resumen en la Tabla 4.

**Tabla 4**

*Los 5 principios SOLID de diseño de aplicaciones de software*

| **Principios SOLID** | | |
|---|---|---|
| **SRP** | El Principio de Responsabilidad Única | Este principio establece que cada clase debe tener una única responsabilidad dentro de nuestro software, y esta responsabilidad debe estar definida y ser concreta. |
| **OCP** | El Principio Abierto/Cerrado | Debe ser capaz de extender un comportamiento de clases, sin modificarlo. |
| **LSP** | El Principio de Sustitución de Liskov | Las clases derivadas deben ser sustituibles por sus clases base. |

| Principios SOLID | | |
|---|---|---|
| **ISP** | El Principio de Segregación de Interfaces | Hacer interfaces de grano fino que son específicos del cliente. |
| **DIP** | El Principio de Inversión de Dependencia. | Depende de abstracciones, no de concreciones. |

El campo del desarrollo de software es un área en constante debate y SOLID no es una excepción. Aunque estos cinco principios se consideran ampliamente como la base de un buen desarrollo o al menos una pauta a tener en cuenta, muchos expertos critican los principios SOLID, acusándolos de ser imprecisos, confusos, de complicar el código, retrasar el desarrollo e incluso llamarlos completamente falsos e innecesarios. Sin embargo, el principal beneficio de seguir los Principios SOLID en el desarrollo es la obtención de mejor código para legibilidad, mantenimiento y testeo. Un mejor código facilita la colaboración entre desarrolladores quienes pueden ampliar, modificar y probar sus programas con menos complicaciones (Sarcar, 2021).

***Ejemplos de código y diseño***

El ejemplo estará orientado a la POO en Java para agregar un lenguaje a la síntesis y semántica que se va a emplear para los cuatro conceptos principales: abstracción, encapsulamiento, herencia y polimorfismo. Primero, se diseña el modelo de clases que emplea el ejercicio y a continuación se detalla las clases y su contenido.

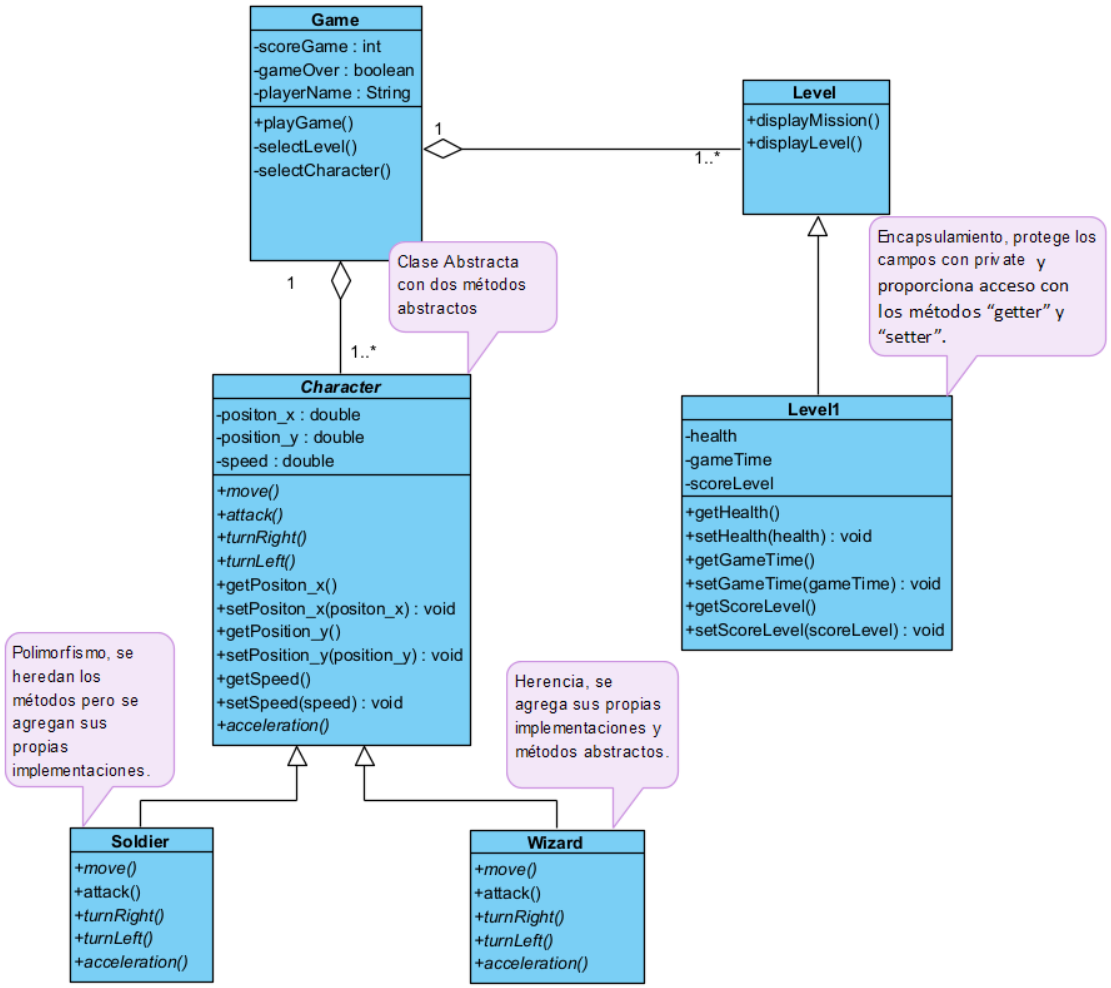
El ejercicio propuesto es el siguiente: Se debe construir un juego llamado “World War Soldiers”, el juego contará con un nivel. Dentro de este nivel, el jugador debe tener la opción de visualizar el tiempo de juego y el puntaje. Para administrar este nivel se debe crear una clase llamada “Level”, que utilice las funciones: seleccionar nivel y mostrar la misión. Se debe tener en cuenta que a futuro se puede agregar más niveles, adicionalmente es necesario una clase “Game” para comenzar o terminar el juego. Contará con dos personajes. El primer personaje se

llama “Soldier” y el segundo personaje se llama “Wizard”. Estos personajes tendrán la opción de atacar y moverse, pero sus ataques y movimientos serán distintos.

En la siguiente Figura 3 se representa la solución para el problema propuesto, se pueden observar los cuatro conceptos principales de la POO en el diagrama de clases.

**Figura 3**

*Diagrama de clases “World War Soldiers”*




En la Tabla 5 se muestra el ejercicio resuelto llamado “World War Soldiers” donde se pone en práctica los 4 pilares fundamentales de la programación orientada a objetos.

**Tabla 5**

*Ejemplo del ejercicio "World War Soldiers"*

| Descripción | Ejemplo de sintaxis de código |
|---|---|
| Clase abstracta llamada "*Character*" con dos métodos abstractos. | abstract class Character {<br>// Abstract Methods<br>abstract void attack();<br>abstract void move();<br>} |
| Dos clases secundarias con sus propias implementaciones y métodos abstractos | class Soldier extends Character {<br>void attack() {<br>System.out.println("Attack with a machine gun.");<br>}<br>void move() {<br>System.out.println("Soldier moves 3 spaces.");<br>}<br>}<br>class Wizard extends Character {<br>void attack () {<br>System.out.println("Attack with a spell.");<br>}<br>void move () {<br>System.out.println("Wizard moves 5 spaces.");<br>}<br>} |
| Para realizar las pruebas se creó una clase TestSoldier y TestWizard ambos inicializan con un objeto (soldier1 y wizard1). | class TestSoldier {<br>public static void main(String[] args) {<br>Soldier soldier1 = new Soldier();<br>soldier1.attack();<br>soldier1.move(); |

| Descripción | Ejemplo de sintaxis de código |
|---|---|
| | } } |
| | [Console output of TestSoldier]<br>Attack with a machine gun.<br>Soldier moves 3 spaces.<br>class TestWizard {<br>public static void main(String[] args) {<br>Wizard wizard1 = new Wizard();<br>wizard1.attack();<br>wizard1.move();<br>} }<br>[Console output of TestWizard ]<br>Attack with a spell.<br>Wizard moves 5 spaces.<br>class Level1 {<br>private String gameTime;<br>private int score;<br>// Getter methods<br>public String getGameTime() {<br>return gameTime; |
| Encapsulamiento, se puede proteger los campos de una clase se debe declarar los campos como private como nombre reservado por la sintaxis de programación y proporcionar accesos a los métodos con “getter” y “setter”. | }<br>public int getScore() {<br>return score;<br>}<br>// Setter methods<br>public void setGameTime(String gameTime) {<br>this.nombre = gameTime;<br>}<br>public void setScore (String score) {<br>this.nombre = score;<br>}<br>} |

| Descripción | Ejemplo de sintaxis de código |
|---|---|
| Para realizar las pruebas se creó una clase TestLevel con un nuevo objeto, se define un valor para cada campo con los métodos setter y finalmente se muestra los valores usando los métodos getter. | class TestLevel {<br>public static void main(String[] args) {<br>Level1 ObjLevel = new Level1 ();<br>ObjLevel . setGameTime ("5 minutes");<br>ObjLevel . setScore (80);<br>System.out.println("Game Time: " + ObjLevel .getGameTime());<br>System.out.println("Score: " + ObjLevel . getScore () + "Points");<br>}<br>}<br>[Console output of TestLevel ]<br>Max Time: 5 minutes<br>Score: 80 Points |
| La herencia permite extender una clase con una o más clases secundarias que heredan los campo y métodos de la clase principal. En java, se debe utilizar la palabra clave “extends” para crear una clase secundaria. | class Level {<br>public void selectLevel1() {<br>System.out.println("Begin the first level …");<br>}<br>public void displayMission() {<br>System.out.println("The objective of the mission is: ");<br>}<br>}<br>class Level1 extends Level {<br>public void displayMission () {<br>super.displayMission ();<br>System.out.println("World war between soldiers and wizards. Destroy all the wizards you find");<br>}<br>} |

| Descripción | Ejemplo de sintaxis de código |
|---|---|
| Para realizar las pruebas se creó una clase TestLevel1 con un nuevo objeto para mostrar toda la información de los campos y métodos heredados. | class TestLevel1 {<br>public static void main(String[] args) {<br>Level1 obj = new Level1 ();<br>obj . selectLevel1();<br>obj .displayMission();<br>}<br>}<br>[Console output of TestLevel1]<br>Begin the first level …<br>The objective of the mission is:<br>World war between soldiers and wizards. Destroy all the wizards you find |

**Lenguajes y frameworks**

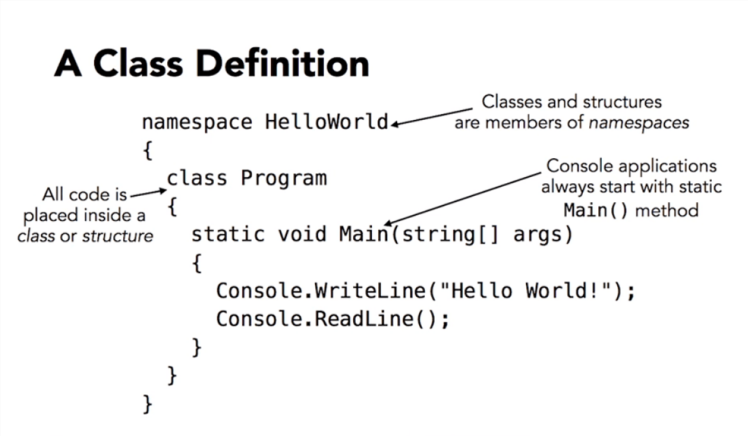
Un lenguaje de programación es el conjunto de instrucciones que utilizan los humanos para interactuar con las computadoras. Permite comunicarse con las computadoras a través de algoritmos e instrucciones escritas en una sintaxis que la computadora entiende e transforma a lenguaje de máquina. Algunos lenguajes de programación, entre los más populares en la actualidad (2022) son: C, C++, C#, Java, Python, JavaScript, PHP,Ruby.

Para facilitar el desarrollo, además de los lenguajes de programación, existen varios tipos de frameworks. Genéricamente un framework es un marco de trabajo que tiene como objetivo proporcionar la solución de problemas que pueden surgir al programar, aceleran el proceso de programación facilitando tareas como la organización del código, funciones preestablecidas, el trabajo en equipo dentro de un proyecto. Por lo general, los frameworks están relacionados con un lenguaje de programación específico, por ejemplo, Laravel para PHP, Ruby on Rails para Ruby, Visual Studio para C#.

C# es un lenguaje de programación el cual “es un lenguaje de programación simple pero eficaz, diseñado para escribir aplicaciones empresariales.” (adegeo, s. f.). Por otro lado, se debe gestionar el código de los proyectos desarrollados, utilizando una herramienta CVS (Control Version System) Git, donde se permita la colaboración entre diferentes usuarios, para el presente experimento se utilizará GitHub, por su alta disponibilidad y su libre acceso a la comunidad académica. Para definir una clase en C# se utiliza la palabra reservada “class” En la Figura 4, se visualiza los elementos principales que componen una clase.

**Figura 4**

*Elementos principales que componen una clase*




- WPF es una herramienta framework .NET, está destinada a proporcionar una API para crear interfaces de usuario robustas y sofisticadas para Windows, que permite desarrollar aplicaciones visualmente atractivas, con facilidades de interacción, que incluyen animación, vídeo, audio, documentos, navegación o gráficos 3D. Separa, con el lenguaje declarativo XAML y los lenguajes de programación de .NET, la interfaz de interacción con el usuario de la lógica del negocio, propiciando una arquitectura “Modelo Vista – Modelo de Vista” para el desarrollo de aplicaciones (adegeo, s. f.).

- GitHub registra el desarrollo de los proyectos de manera remota, permite compartir proyectos entre distintos usuarios y proporciona la seguridad de la nube entre otras funciones. Cuando se trabaja en proyectos colaborativos, la base de la interacción entre los colaboradores de un proyecto y GitHub se basa que todos tienen una copia del proyecto, es decir tanto los desarrolladores como GitHub. No obstante, para beneficio de los desarrolladores GitHub contiene una copia centralizada del control de versiones de los repositorios. (Astigarraga & Cruz-Alonso, 2022)

**Experimentos en Ingeniería de Software**

En la actualidad, las tecnologías utilizadas en el desarrollo de software no cuentan con pruebas de su adecuación, límites, cualidades, costos y riesgos (Boulet, 2017). La única manera de verificar creencias y opiniones y convertirlas en hechos es a través de la experimentación.

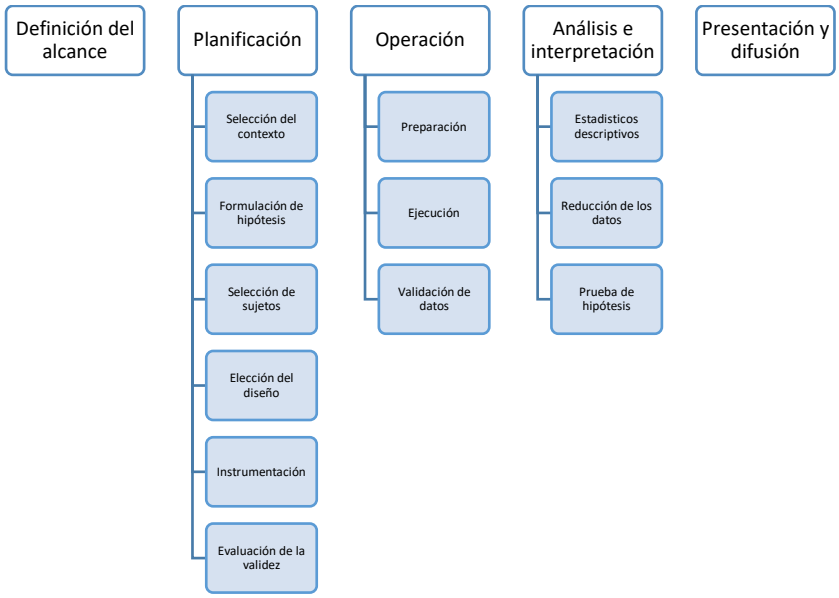
En general, el objetivo de la experimentación es determinar las causas de los resultados obtenidos. Al utilizar la ingeniería de software, la experimentación ayuda a identificar y comprender diferentes aspectos y conexiones involucradas en el desarrollo y mantenimiento de productos de software. Esta identificación de aspectos y conexiones que permite validar o refutar con hechos las creencias y prácticas que utilizan en el desarrollo y mantenimiento de productos software (Genero, Cruz-Lemus, & Piattini, 2014).

La experimentación en ingeniería de software permite comprender e identificar las variables involucradas en la creación de aplicaciones software. Experimentar con la construcción de software aumenta la comprensión de lo que hace el software y de cómo crear software de calidad. El objetivo de la experimentación en ingeniería de software es hacer que el desarrollo de software sea una actividad predecible científicamente al conocer las relaciones entre los procesos de producción de software y los productos obtenidos (Genero, Cruz-Lemus, & Piattini, 2014).

Proceso experimental: Para llevar a cabo un experimento es necesario seguir un proceso consistente que permita definir claramente su alcance, objetivo y desarrollo. En la Figura 5, se muestran los elementos principales del proceso experimental.

**Figura 5**

Proceso experimental




*Nota.* Adaptado de *Métodos de Investigación en Ingeniería del Software*, por (Genero, Cruz-Lemus, & Piattini, 2014), Ediciones de la U LTDA.

- **Definición del alcance:** Esta actividad consiste en especificar de manera general los aspectos principales del experimento a realizar y en definir las hipótesis de trabajo. En esta fase se puede utilizar el método GQM (Goal-Question-Metric) para definir el objeto de estudio del experimento, el propósito, el enfoque de calidad, la perspectiva y el contexto en el que se realiza el experimento.
- **Planificación:** En la etapa de planificación se llevan a cabo una serie de tareas que ayudan a tener una idea clara de cómo se realizará el experimento, las tareas son:

- Selección del contexto: Se determina el entorno en el que se llevará a cabo el experimento, lo que permite decidir en qué situaciones son válidos los resultados del experimento. Por lo general, el contexto de un experimento está determinado por cuatro dimensiones, según (Genero, Cruz-Lemus, & Piattini, 2014) estas son:
    - Online vs Offline
    - Profesionales vs Estudiantes
    - Problemas “de juguete” o Proyectos reales
    - Específico o General
- Formulación de la hipótesis: En un experimento, es necesario definir lo que se quiere probar, lo que se puede expresar mediante una hipótesis. Por lo general, se parte de una o varias hipótesis nulas, que son las suposiciones que se rechazan mediante la ejecución del experimento
- Selección de variables: Hay varios tipos de variables que deben tenerse en cuenta durante un experimento, tales como:
    - Variables independientes: se modifican para estudiar el efecto de cambios producidos en base a la hipótesis.
    - Variables dependientes: se observan para determinar los cambios producidos por las variables independientes.
    - Variables controladas: variables independientes que se pueden controlar de manera fija
    - Variables enmascaradas: variables que no se pueden controlar y también pueden producir cambios en las variables dependientes.
    - Variables aleatorias: variables no controladas que se pueden tratar como errores aleatorios.

- Elección del diseño: La elección del diseño experimental es un paso crucial durante la planificación de un experimento, ya que es la base para la replicación y confirmación del experimento (Genero, Cruz-Lemus, & Piattini, 2014):
  - Aleatorización: Se refiere a la distribución aleatoria de los tratamientos y los sujetos que participarán en el experimento, así como a la selección aleatoria de sujetos dentro de una muestra.
  - Bloqueo: Permite eliminar el efecto no deseado de un factor dentro del experimento, lo que evita sesgos al procesar los datos obtenidos.
  - Balanceo o equilibrado: Es útil al asignar tratamientos y se cumple cuando a cada tratamiento se le asigna un número igual de sujetos.
- Selección de sujetos: Un sujeto es el individuo en el que se realiza el experimento, por lo tanto, la selección de sujetos tiene un efecto significativo en los resultados que se obtendrán durante la ejecución del experimento (Genero, Cruz-Lemus, & Piattini, 2014).
- Instrumentación: La instrumentación busca establecer una metodología para llevar a cabo un experimento sin afectar el grado de control que se mantiene sobre él. La validez de un experimento depende en gran medida de cómo se haya instrumentado, los resultados deben ser los mismos independientemente de cómo se haya instrumentado. Esto significa que si los instrumentos afectan el resultado del experimento, este no será válido (Genero, Cruz-Lemus, & Piattini, 2014).
- Evaluación de la validez: Antes de realizar cualquier experimento, es importante estimar cuán válidos serán los resultados del experimento, es decir, considerar los posibles factores que pueden amenazar la validez y planificar cómo mitigarlos.

- A la validez interna: Esto afecta el grado de confianza en la relación causa-efecto entre las variables independiente y dependiente. Por ejemplo, la selección y agrupación de sujetos, su tratamiento durante el experimento, los materiales utilizados y cualquier evento inesperado que ocurra durante la ejecución.
- A la validez externa: Esto afecta el grado en el que se pueden generalizar los resultados a la población seleccionada para el experimento.
- A la validez de constructo: La principal amenaza a la validez de constructo es la falta de pruebas teóricas que afirman que las variables dependientes e independientes realmente miden los conceptos especificados en el experimento.
- A la validez de la conclusión: Se determina por la significación estadística de los resultados del experimento.

- **Operación**
  - Preparación: En esta etapa, se preparan a los participantes del experimento para llevar a cabo los tratamientos especificados. Hay varias consideraciones éticas que deben tenerse en cuenta ya que, en la mayoría de los casos, un experimento de ingeniería de software involucra a seres humanos:
    - Obtener el consentimiento de los sujetos.
    - Mantener la confidencialidad del rendimiento de cada sujeto
    - Ofrecer un incentivo por participar en el experimento.
    - Comunicar todos los aspectos del experimento de manera equitativa, siempre y cuando esto no afecte los resultados.

Además, durante la fase de preparación es importante entrenar a los sujetos en el tema de estudio para que puedan realizar los tratamientos asignados. También se recomienda

pedir a los sujetos que completen una encuesta sobre sus datos personales, experiencia y conocimientos previos con el fin de obtener resultados más precisos (Genero, Cruz-Lemus, & Piattini, 2014).

- Ejecución: Una vez que se ha definido el experimento y se ha seleccionado el diseño, es el momento de llevar a cabo el experimento. En esta etapa, se tienen todos los materiales y equipos necesarios, así como el espacio donde se llevará a cabo el experimento. Antes de comenzar la sesión del experimento, los sujetos reciben instrucciones y los materiales con los que trabajarán durante la sesión.
- Validación de los datos: Después de recopilar los datos, es importante considerar cualquier anormalidad detectada durante la ejecución del experimento para facilitar la extracción de datos.

- **Análisis e interpretación**

Para obtener resultados precisos en el experimento, es importante llevar a cabo un análisis estadístico cuantitativo utilizando técnicas como estadísticos descriptivos, reducción de datos y contraste de hipótesis. La Tabla 6, muestra los test estadísticos más apropiados para comparar hipótesis en función del tipo de diseño experimental elegido y la distribución de los datos. Si la distribución de los datos es normal, se deben utilizar test paramétricos; de lo contrario, se deben utilizar test no paramétricos. Una vez realizado el análisis cuantitativo, es posible interpretar los resultados y presentar las conclusiones del experimento.

**Tabla 6**

*Test estadísticos según el diseño experimental*

| Tipo de diseño | Test paramétricos | Test no paramétricos |
|---|---|---|
| Un factor, un tratamiento | | • Test binomial<br>• Chi 2 |
| Un factor, dos | • Test T | • Mann-Whitney |

| Tipo de diseño | Test paramétricos | Test no paramétricos |
|---|---|---|
| tratamientos | • Test F<br>• Test T emparejado | • Chi 2<br>• Wilcoxon |
| Un factor, más de dos tratamientos | • ANOVA | • Kruskal-Wallis<br>• Chi 2 |
| Más de un factor | • ANOVA | |

Test estadístico: Son utilizados para ayudar a la toma de decisión de si una suposición sobre una población se puede confirmar o rechazar, mediante la observación de una muestra. El test estadístico consiste en plantear una hipótesis nula sobre la población y aplicar el test adecuado, en contraste con la hipótesis, sobre la muestra observada (Alsawaier, 2018).

La probabilidad de encontrar diferencias significativas o equivalentes entre los resultados observados y los teóricos del estudio debe ser muy pequeña, por lo que se debe fijar un nivel de significancia tal que los sucesos con probabilidad menor lleven a rechazar la hipótesis nula.

A dicha probabilidad se le denomina p-valor, por lo que, si el p-valor es menor que el nivel de significancia escogido, la diferencia entre la observación de la muestra y lo esperado por la hipótesis nula será estadísticamente significativa, rechazando la hipótesis del estudio.

Previo a ejecutar un test estadístico sobre una muestra, es necesario comprobar si los datos siguen una distribución normal o no, para así, aplicar el test estadístico apropiado (Alsawaier, 2018).

En el contexto del presente estudio se utilizará el test no paramétrico de Mann–Whitney cuando los datos de la muestra no sigan una distribución normal, caso contrario se utilizará el test paramétrico Test-T.

Para ejemplificar el proceso experimental aquí expuesto, se describe el siguiente estudio que tiene como objetivo principal, investigar la evidencia existente sobre la influencia del uso de aplicaciones gamificadas en la mejora del rendimiento de los estudiantes en la enseñanza de la Informática. Los resultados muestran que el uso de la gamificación ejerce una influencia significativa en la mejora del rendimiento de los estudiantes. Sin embargo, estos resultados deben ser considerados como preliminares, ya que la evidencia empírica es aún escasa (Marín et al., s. f.).

Ejemplo de proceso experimental: Una investigación empírica sobre los beneficios de la gamificación en los cursos de programación (Marín et al., s. f.).

- **Objetivo:** El objetivo principal es obtener evidencia empírica sobre la mejora del rendimiento de aprendizaje, cuando se utiliza una plataforma gamificada llamada "UDPiler" en comparación con un compilador no gamificado.
- **Método de investigación:** El método de investigación elegido fue un cuasi-experimento (observational study), no fue posible asignar aleatoriamente los tratamientos a los sujetos. El cuasi-experimento, se realizó con dos grupos de estudiantes de primer año de ingeniería de la Universidad Diego Portales en Chile, utilizando un compilador no gamificado y una plataforma gamificada. Se utilizó la plantilla Goal-Question-Metric, un método comúnmente utilizado para la definición de objetivos experimentales, en el que se establece claramente el objeto de estudio, el propósito, el enfoque y el punto de vista de la medición.
- **Variables:** La variable independiente fue el método de enseñanza de la programación en C, al que denominaron Prog - Method. Esta es una variable nominal con dos valores: Gamificado (Gam) y No Gamificado (Non-Gam). Y la variable dependiente fue el

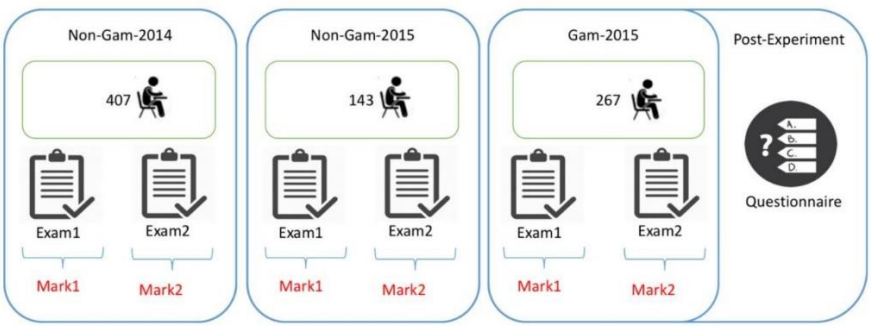
rendimiento de aprendizaje de los estudiantes. Midieron la variable dependiente de la siguiente manera:

  - Calificación 1: Examen realizado a mitad de semestre, denominado Examen 1.
  - Calificación 2: Examen realizado al final del semestre, denominado Examen 2.

- **Hipótesis:** En base al objetivo planteado y la definición de las medidas definidas para la variable dependiente se formularon las siguientes hipótesis:
  - H0: Los alumnos que utilicen la plataforma gamificada no obtendrán notas diferentes a las obtenidas por los alumnos que utilicen la plataforma no gamificada en los exámenes de medio semestre (Calificación 1) y fin de semestre (Calificación 2).
  - H1: ¬H0.
- **Diseño:** Realizaron un diseño entre sujetos. Esto significa que hubo dos grupos diferentes de estudiantes, cada uno de los cuales realizó tareas similares utilizando una de las dos técnicas de enseñanza. La Figura 6 muestra un diagrama esquemático del diseño del cuasi-experimento.

**Figura 6**

*Diseño del cuasi-experimento*




*Nota.* Adaptado de "An Empirical Investigation on the Benefits of Gamification in Programming Courses", por Marín et al., s. f., *ACM Transactions on Computing Education*, *19*(1), 22.

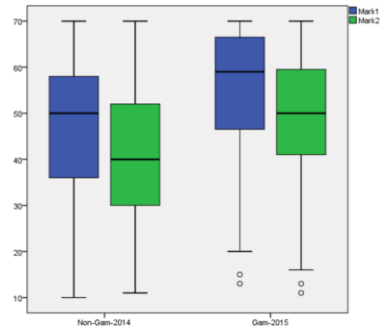
- **Procedimiento de análisis:** El análisis de datos se llevó a cabo en los siguientes pasos:
    - Primero se realizó un estudio descriptivo de las medidas de la variable dependiente, es decir, Mark 1 y Mark 2, para obtener una visión general de los resultados.
    - Realizaron las pruebas estadísticas de normalidad y homogeneidad de uso común, es decir, las pruebas de Kolmogorov-Smirnov y Levene, para determinar la normalidad de la homogeneidad de las varianzas. Estos análisis fueron útiles para determinar qué prueba sería adecuada en las pruebas de contraste de hipótesis.
    - Como los datos no tenían una distribución normal, utilizaron la prueba U de Mann-Whitney para contrastar las hipótesis experimentales. En todas las pruebas estadísticas decidieron aceptar una probabilidad del 5% de cometer un error.
- **Resultados experimentales:** Los resultados revelaron que los estudiantes obtuvieron mejores calificaciones cuando se utilizó la plataforma gamificada para aprender programación en C. Además, existe datos estadísticos a favor de un efecto positivo en el rendimiento de aprendizaje, de aquellos alumnos que utilizaron la plataforma gamificada.
- **Estadísticas:** Las estadísticas descriptivas relacionadas con la Calificación 1 y la Calificación 2 que se observan en la Figura 7, revelan que las calificaciones obtenidas al usar la plataforma gamificada (Gam-2015) fueron más altas que en el otro caso. Los resultados obtenidos por el grupo Non-Gam-2014 y el grupo Gam-2015, se muestran en la Figura 8, el desempeño de los estudiantes en términos de las calificaciones obtenidas. En ambos casos, Calificación 1 y Calificación 2, se puede observar una mejora en el rendimiento de aprendizaje del grupo Gam-2015 con respecto al grupo No-Gam-2014, estos resultados se analizaron estadísticamente con SPSS v.24. En el mismo año (Gam-

2015 vs. Non-Gam-2015), esta doble verificación de las hipótesis se realizó para fortalecer los beneficios de la gamificación. Para comprobar la normalidad de los datos (Kolmogorov-Smirnov), se vio que los valores no se distribuían normalmente. En consecuencia, se utilizó la prueba U de Mann-Whitney para probar las hipótesis. Al final se compararon los resultados de los grupos Non-Gam-2014 y Gam-2015, En la Figura 8 se muestra que el nivel de significación de la prueba es, para ambas variables, superior al 99,9%, lo que significa que la hipótesis nula H0, que declaró que los estudiantes que utilizan la plataforma gamificada no obtendría calificaciones diferentes a los estudiantes que usan la plataforma no gamificada, puede ser rechazada como el nivel de significación está por debajo del valor alfa, que se estableció en 0,05.

**Figura 7**

*Comparación de las Calificaciones (No-Gam-2014 vs Gam-2015)*

| | Prog-Method | N | Mean | Std. Dev. |
|---|---|---|---|---|
| Mark 1 | Non-Gam-2014 | 407 | 46.563 | 15.6990 |
| | Gam-2015 | 267 | **55.555** | 12.7114 |
| | Non-Gam-2015 | 143 | 50.406 | 14.6956 |
| Mark 2 | Non-Gam-2014 | 407 | 40.378 | 14.7188 |
| | Gam-2015 | 267 | **49.109** | 12.6238 |
| | Non-Gam-2015 | 143 | 48.175 | 14.3950 |




| Variable | U | sig. |
|---|---|---|
| Mark 1 | 37818.0 | **<0.001** |
| Mark 2 | 39705.5 | **<0.001** |

*Nota.* Adaptado de "An Empirical Investigation on the Benefits of Gamification in Programming Courses", por Marín et al., s. f., *ACM Transactions on Computing Education*, *19*(1), 22.

Sin embargo, en la Figura 8 se muestra que al comparar los resultados obtenidos por el grupo Gam-2015 con los del grupo Non-Gam-2015, el nivel de significancia de la prueba es superior al 99,9% solo para la Calificación 1. Por lo tanto, la hipótesis nula H0 se rechaza en este caso.

**Figura 8**

*Comparación de la prueba t (Gam-2015 vs No-Gam-2015)*

| Variable | U | sig. |
|---|---|---|
| Mark 1 | 15963.5 | **<0.001** |
| Mark 2 | 19685.5 | 0.403 |

*Nota.* Adaptado de "An Empirical Investigation on the Benefits of Gamification in Programming Courses", por Marín et al., s. f., *ACM Transactions on Computing Education*, *19*(1), 22.

- **Conclusiones:** La gamificación es un enfoque alentador para enseñar programación en C, descubrimiento que se encuentra alineado con estudios empíricos previos sobre gamificación en cursos de programación realizados en contextos académicos. Sin embargo, también se requiere una mayor validación para corroborar y fortalecer los hallazgos obtenidos e investigar si el tipo de elementos gamificados que se utilizó, tienen alguna influencia en el rendimiento de los estudiantes, entre otras cuestiones que merecen mayor atención (Marín et al., s. f.).

# Capítulo III

## Diseño y planificación del experimento

### Definición del alcance

El presente experimento tiene que como objetivo desarrollar una herramienta de apoyo basada en la gamificación, para aumentar la motivación y mejorar el nivel de aprendizaje durante el proceso de enseñanza de los principios de la programación orientada a objetos, en alumnos de Ingeniería de Software y de Ingeniería de Tecnologías de la Información de la Universidad de las Fuerzas Armadas ESPE.

#### *Selección del contexto*

- Objetos experimentales: Para estudiar el presente contexto, se han elegido dos elementos experimentales: una interfaz de usuario web (frontend) que no se modificará durante el experimento y un conjunto de servicios REST (backend). Ambas se programarán utilizando el paradigma de programación orientada a objetos y los lenguajes de programación C# y Java.
- Sujetos experimentales: Para llevar a cabo el estudio, se cuenta con 123 estudiantes que cursan la asignatura de Programación Orientada a Objetos (POO) en distintas carreras del Departamento de Ciencias de la Computación de la Universidad de las Fuerzas Armadas ESPE. La siguiente es la distribución de los participantes:
    - 59 estudiantes de segundo semestre de Tecnologías de la información TI de la asignatura con el código de la materia. COMPA0J07 (POO para Tecnologías de la Información (TI) 6 horas (3 sesiones)) (Malla Curricular, s. f.-b).
    - 64 estudiantes de segundo semestre de Software de la asignatura con el código de la materia. COMPA0J08 (POO de Software 8 horas (4 sesiones)) («Malla Curricular», s. f.-a).

Los grupos de sujetos experimentales se han elegido porque cuentan con los conocimientos necesarios en programación, algoritmos, resolución de problemas y fundamentos de ingeniería de software para realizar correctamente las tareas del experimento.

Los sujetos del experimento participan de forma voluntaria. Para preservar la validez del experimento y evitar ideas equivocadas en los sujetos, se siguen algunas medidas sugeridas. Entre ellas, se encuentra informar a los estudiantes que su rendimiento en el experimento no será evaluado y que solo realizarán una actividad práctica sobre temas relacionados con el experimento.

Por otro lado, para asegurar que todos los sujetos cuenten con una base de conocimientos común, se impartirá una capacitación básica sobre los lenguajes de programación C# y Java, los fundamentos de la POO relacionados con el experimento y la herramienta gamificada (programa ejemplo) que se utilizará para el experimento. Para este experimento se ha decidido trabajar con estudiantes debido a que:

- Las tareas propuestas no requieren experiencia profesional.
- Los grupos de estudiantes tienen un nivel de conocimientos homogéneo.
- Es factible monitorear el experimento con estudiantes.
- Hay una buena cantidad de sujetos experimentales disponibles.
- El objetivo principal de este trabajo es mejorar y motivar el aprendizaje de programación orientada a objetos.

**Selección de variables**

***Variable independiente***

Método de enseñanza de los pilares de programación orientada a objetos (Abstracción, Herencia, Encapsulamiento, Polimorfismo y SOLID Principles), a los que se denominará de aquí de aquí en adelante como ***PrinciplesOOP.*** Para la variable nominal se escogió dos valores:

Gamificado (***Gam***) y No Gamificado (***NonGam***).

***Variables dependientes***

Aumento de motivación y aprendizaje de los estudiantes. Para medir las variables dependientes se realizará una evaluación y una encuesta al finalizar el proceso experimental, utilizando las siguientes métricas.

- ***Aumento de aprendizaje (Auapr):*** Esta métrica busca medir el aumento de aprendizaje de un sujeto en la relación a ***PrinciplesOOP*** a partir de la resolución de problemas mediante la programación de computadoras utilizando la gamificación como metodología de enseñanza. La métrica es calculada con una evaluación.
- ***Aumento de motivación (Aumot):*** Esta métrica está relacionada con el incremento de la motivación para estudiar programación y realizar las actividades planificadas, para adquirir el mayor nivel de conocimiento en aprender ***PrinciplesOOP*** y es calculada mediante una encuesta.

**Formulación de hipótesis**

En la Tabla *7*, se presentan las hipótesis nulas y alternativas que evalúan la relación entre la variable dependiente e independiente. El propósito del análisis estadístico es rechazar las hipótesis nulas y, en su lugar, determinar que existe la posibilidad de aceptar algunas hipótesis alternativas. Siempre y cuando, exista suficiente evidencia. Las hipótesis nulas se formularon de la siguiente manera:

- **$H_{0,1}$:** Los estudiantes que utilizan la plataforma gamificada obtendrán igual aprendizaje en conocimientos (teórico) a los estudiantes que no utilicen la plataforma no gamificada en la evaluación final del experimento.

- **$H_{0,2}$:** Los estudiantes que utilizan la plataforma gamificada obtendrán igual aprendizaje en habilidades (práctico) a los estudiantes que no utilicen la plataforma no gamificada en la evaluación final del experimento.
- **$H_{0,3}$:** Los estudiantes que utilizan la plataforma gamificada obtendrán igual motivación a los estudiantes que no utilicen la plataforma no gamificada en la encuesta final del experimento

**Tabla 7**

*Hipótesis nulas y alternativas del experimento*

| Hipótesis Nula | Hipótesis Alternativas |
|---|---|
| **$H_{0,1}$: $Auapc_{gam} = Auapc_{nongam}$** | $H_{1,1,1}$: $Auapc_{gam} \neq Auapc_{nongam}$ |
| | $H_{1,1,2}$: $Auapc_{gam} > Auapc_{nongam}$ |
| | $H_{1,1,3}$: $Auapc_{gam} < Auapc_{nongam}$ |
| **$H_{0,2}$: $Aumoh_{gam} = Aumoth_{nongam}$** | $H_{1,2,1}$: $Aumoh_{gam} \neq Aumoh_{nongam}$ |
| | $H_{1,2,2}$: $Aumoh_{gam} > Aumoh_{nongam}$ |
| | $H_{1,2,3}$: $Aumoh_{gam} < Aumoh_{nongam}$ |
| **$H_{0,3}$: $Aumot_{gam} = Aumot_{nongam}$** | $H_{1,3,1}$: $Aumot_{gam} \neq Aumot_{nongam}$ |
| | $H_{1,3,2}$: $Aumot_{gam} > Aumot_{nongam}$ |
| | $H_{1,3,3}$: $Aumot_{gam} < Aumot_{nongam}$ |

**Elección del diseño**

Se realizará un diseño entre sujetos. Esto significa que habrá dos grupos diferentes de estudiantes, cada uno de los grupos tendrá tareas similares y un mismo objetivo en común, utilizando una de las dos técnicas de enseñanza respectivamente.

- Tratamiento 1: Ejecutar el método de enseñanza (***PrinciplesOOP***) sin la técnica de gamificación (NonGam).

- Tratamiento 2: Ejecutar el método de enseñanza (***PrinciplesOOP***) con la técnica de gamificación (Gam).

Para los diferentes grupos de estudiantes, se aplicará un diseño intra-sujetos descrito por (Gómez et al., 2018), en donde menciona que cada grupo de sujetos se divide en dos subgrupos con el mismo número de sujetos seleccionados al azar. Cada subgrupo recibe un orden específico en el que se llevará a cabo el experimento.

El orden en que se llevarán a cabo las tareas gamificadas se detalla en el instructivo de desarrollo *Apéndice A*. La Tabla *8*, muestra la distribución de los sujetos y los tratamientos para el experimento.

**Tabla 8**

*Distribución de sujetos y tratamientos para el experimento*

| **Día 1–2** | **Día 3-4** | **Día 5-6** |
|---|---|---|
| G1-Gam-T1-T2-$SOFT_{OOP}$ | G1-Gam- T3-T4-$SOFT_{OOP}$ | G1-Gam- T5- T6- T7-$SOFT_{OOP}$ |
| G2-NonGam- T1-T2-$SOFT_{OOP}$ | G2-NonGam- T3-T4 -$SOFT_{OOP}$ | G2-NonGam- T5- T6- T7 -$SOFT_{OOP}$ |
| G1-Gam- T1-T2 -$TI_{OOP}$ | G1-Gam- T3-T4 -$TI_{OOP}$ | G1-Gam- T5- T6- T7 -$TI_{OOP}$ |
| G2-NonGam- T1-T2- $TI_{OOP}$ | G2-NonGam- T3-T4 -$TI_{OOP}$ | G2-NonGam- T5- T6- T7 - $TI_{OOP}$ |

Donde:

- $SOFT_{OOP}$: Curso de Programación Orientada a Objetos en la carrera de Ingeniería de Software.

- $TI_{OOP}$: Curso de Programación Orientada a Objetos en la carrera de Ingeniería en Tecnologías de la Información.
- G1 y G2: Grupo de sujetos seleccionados aleatoriamente.
- Gam: Técnica de aprendizaje gamificación
- NonGam: No incluye la técnica de aprendizaje gamificación.
- T1: Tema de abstracción
- T2: Tema de encapsulamiento
- T3: Tema de herencia
- T4: Tema de polimorfismo
- T5: Tema de SOLID (Responsabilidad única)
- T6: Tema de SOLID (Abierto/Cerrado)
- T7: Tema de SOLID (Substitución de Liskov)

**Instrumentación**

Para evaluar las métricas de motivación y aprendizaje, se utilizan los siguientes instrumentos:

- Un backend en dos versiones, utilizando el paradigma de programación Orientada a Objetos en (C#), para los 4 pilares fundamentales de programación.
- Un frontend desarrollado en WPF (Windows Presentation Foundation), que permite a los sujetos probar el sistema mediante una interfaz amigable.
- Un backend en dos versiones, utilizando el paradigma de programación Orientada a Objetos en (JAVA), para los principios SOLID.
- Un frontend desarrollado en SWING (Biblioteca de clases que permite crear interfaces gráficas en Java), que permite a los sujetos probar el sistema mediante una interfaz amigable.

Para cada versión de backend se desarrolló una guía práctica de laboratorio, una evaluación y una encuesta. Uno por cada tema propuesto en el experimento, las cuales permitieron comprobar si hubo un aumento de aprendizaje y motivación. Para calcular el porcentaje de cada uno de las métricas del experimento.

Se consideró una guía práctica de laboratorio que tiene como único objetivo evidenciar el aprendizaje y motivación de los estudiantes, detallando la duración, objetivo, descripción, actividades, planteamiento del problema, y desarrollo de las siguientes tareas:

- Crear las clases necesarias para resolver el problema planteado.
- Utilizar el paradigma de programación orientada a objetos manejando los 4 pilares fundamentales y los principios SOLID.
- Implementar la funcionalidad del juego que se describe en la práctica.
- Seguir las instrucciones de la práctica guiada por la persona a cargo.

Cada tarea de la práctica tiene como objetivo modificar el código del backend y se detalla de la siguiente manera:

- Título de la tarea, principio fundamental de POO
- Descripción sobre el planteamiento del problema, requerimiento
- Pasos para reproducir la solución reutilizando el frontend y programando el backend, salida esperada y tiempo estimado para completar el experimento

Para cada grupo de estudiantes ($SOFT_{OOP}$ y $TI_{OOP}$) se configurarán dos repositorios en GitHub: OOP-Gam-Experiment-TI[1] y OOP-Gam-Experiment-Soft[2]. Cada sujeto manejará su propia rama (branch) en el repositorio, que incluye las guías prácticas de laboratorio y el código fuente

[1] https://github.com/neortiz1/OOP-Gam-Expreiment-TI
[2] https://github.com/neortiz1/OOP-Gam-Experiment-Soft

del frontend y las versiones de backend para cada principio fundamental de POO. Esto permitirá revisar de manera eficiente los cambios realizados durante el experimento.

Finalmente, como parte del experimento, se diseñaron ocho evaluaciones y siete encuestas para cada tema planteado:

- Evaluación previa al experimento, evidenciar conocimientos previos.
- Evaluación post experimento, abstracción
- Evaluación post experimento, encapsulamiento
- Evaluación post experimento, herencia
- Evaluación post experimento, polimorfismo
- Evaluación post experimento, SOLID (Responsabilidad única), (Abierto/Cerrado) y (Substitución de Liskov).
- Encuesta post experimento por cada tema mencionado anteriormente.

Se harán preguntas sencillas en la encuesta para medir la percepción del aprendizaje de los participantes. Se utilizará la escala de Likert de cinco puntos. Esta información será útil para interpretar y explicar los resultados obtenidos en relación a la facilidad de aprendizaje.

El modelo de la guía práctica de laboratorio y los enlaces a los repositorios de GitHub se encuentran en el *Apéndice A*, la evaluación pre experimento en el *Apéndice B*, la evaluación post experimento para abstracción en el *Apéndice C*, la evaluación post experimento para encapsulamiento en el *Apéndice D*, la evaluación post experimento para herencia en el *Apéndice E*, la evaluación post experimento para polimorfismo en el *Apéndice F*, la evaluación post experimento para SOLID (Responsabilidad única, Abierto/Cerrado y Subtitución de Liskov ) , y la encuesta post experimento que será única para cada tema se encuentra en el *Apéndice G*.

**Operación**

**Preparación**

Antes de comenzar el experimento, se realizará las siguientes actividades:

- Se asignarán los sujetos a los grupos de acuerdo a sus horarios de clase, como se muestra en la Tabla 9, Tabla 10 y Tabla 11.
- **$TI_{OOP}$:** Curso de Programación Orientada a Objetos en la carrera de Ingeniería en Tecnologías de la Información.
- **G1 y G2:** Grupo de sujetos seleccionados aleatoriamente.
- **Gam:** Técnica de aprendizaje gamificación.
- **NonGam:** No incluye la técnica de aprendizaje gamificación.
- **I1:** Inducción al Experimento.
- **T1:** Tema de abstracción.
- **T2:** Tema de encapsulamiento.
- **T3:** Tema de herencia.
- **T4:** Tema de polimorfismo.
- **T5:** Tema de SOLid (Responsabilidad única, Abierto/Cerrado y Substitución de Liskov).

El tiempo establecido a cada entrenamiento se ajusta al nivel de conocimientos generales de cada grupo (cada semana representa de 4 a 6 horas de clase). Se pide a los sujetos que respondan a una encuesta de conocimientos previos. Se llevará a cabo un entrenamiento, dos semanas antes del experimento, en el que se capacitará a los sujetos en conceptos generales y herramientas necesarias.

**Tabla 9**

*Distribución de sujetos y tratamientos para el entrenamiento, semana 1*

| Día 1 | Día 2 |
|---|---|
| G1-Gam-I1-SOFT$_{OOP}$ | G1-Gam- T1-SOFT$_{OOP}$ |
| G2-NonGam- I1-SOFT$_{OOP}$ | G2-NonGam- T1 -SOFT$_{OOP}$ |
| G1-Gam- I1 -TI$_{OOP}$ | G1-Gam- T1 -TI$_{OOP}$ |
| G2-NonGam- I1- TI$_{OOP}$ | G2-NonGam- T1 - TI$_{OOP}$ |

**Tabla 10**

*Distribución de sujetos y tratamientos para el entrenamiento, semana 2*

| Día 3 | Día 4 |
|---|---|
| G1-Gam-T2-T3-SOFT$_{OOP}$ | G1-Gam- T4-SOFT$_{OOP}$ |
| G2-NonGam- T2-T3-SOFT$_{OOP}$ | G2-NonGam- T4 -SOFT$_{OOP}$ |
| G1-Gam- T2-T3 -TI$_{OOP}$ | G1-Gam- T4 -TI$_{OOP}$ |
| G2-NonGam- T2-T3- TI$_{OOP}$ | G2-NonGam- T4 - TI$_{OOP}$ |

**Tabla 11**

*Distribución de sujetos y tratamientos para el entrenamiento, semana 3*

| Día 5 | Día 6 |
|---|---|
| G1-Gam-T5-SOFT$_{OOP}$ | G1-Gam-T5-SOFT$_{OOP}$ |
| G2-NonGam- T5-SOFT$_{OOP}$ | G2-NonGam- T5 -SOFT$_{OOP}$ |
| G1-Gam- T5 -TI$_{OOP}$ | G1-Gam- T5 -TI$_{OOP}$ |
| G2-NonGam- T5- TI$_{OOP}$ | G2-NonGam- T5 - TI$_{OOP}$ |

**Ejecución**

Se divide a los sujetos de acuerdo al esquema presentado en la Tabla 8 se proporcionará el material necesario para llevar a cabo las tareas descritas en las instrucciones y se brindará las indicaciones sobre cómo llevar a cabo el laboratorio/taller que implemente los fundamentos de la POO:

a) Antes de comenzar el entrenamiento los sujetos completarán un formulario de consentimiento donde aceptan los términos y condiciones del experimento.
b) Los sujetos participarán en una fase de entrenamiento para comprender los conceptos básicos y configurar las herramientas necesarias para implementar los requerimientos solicitados.
c) Durante el experimento, no se permitirá la comunicación entre los sujetos.
d) Solo la persona encargada podrá dar la orden de iniciar la aplicación web.
e) Una vez que todos los sujetos hayan iniciado la aplicación, se registrará la hora de inicio y podrán comenzar a completar las tareas del instructivo en el orden especificado.
f) Cada vez que un sujeto complete una tarea, deberá informarlo a las personas encargadas para registrarlo en la rúbrica.
g) Al finalizar el tiempo estimado para el experimento, los sujetos deberán subir sus cambios en el código a su respectiva rama del repositorio GitHub.
h) Para evitar posibles sesgos en los resultados, no se comunicaron a los sujetos las hipótesis del estudio, solo se les informará que el presente experimento forma parte de una investigación sobre las técnicas de enseñanza en el ámbito académico.
i) Cualquier duda debe ser consultada con las personas encargadas.

**Validación de datos**

Una vez finalice el experimento, se recopilan los datos de la rúbrica y los elementos necesarios para calcular cada una de las métricas especificadas en la variable dependiente. Las tareas incorrectas no se puntúan negativamente. Si se detectan valores irregulares, estos serán excluidos.

**Análisis e Interpretación**

Para obtener los resultados del experimento, es necesario llevar a cabo un análisis estadístico (cuantitativo) que incluya estadísticos descriptivos, reducción de datos y contraste de hipótesis.

En la Tabla 12 se muestra los test estadísticos más adecuados para el contraste de la hipótesis en función del tipo de diseño experimental seleccionado y la distribución de los datos. Si la distribución de los datos es normal, se utilizará un test paramétrico, en caso contrario se utilizará uno no paramétrico. Una vez realizado el análisis cuantitativo se pueden interpretar los resultados y presentar las conclusiones del experimento.

**Tabla 12**

*Test estadísticos según el diseño experimental*

| Tipo de diseño | Test paramétricos | Test no paramétricos |
|---|---|---|
| Un factor, dos tratamientos | • Test T<br>• Test F<br>• Test T emparejado | • Mann-Whitney<br>• Chi 2<br>• Wilcoxon |
| Un factor, más de dos tratamientos | ANOVA | • Kruskal-Wallis<br>• Chi 2 |

# Capítulo IV

## Análisis e interpretación de resultados

### Análisis del consentimiento informado

El formulario de consentimiento informado fue necesario para que la investigación se ejecutará éticamente y que los sujetos entiendan todos los puntos descritos en el *Apéndice H,* como por ejemplo el propósito de la investigación, los beneficios y el acuerdo de confidencialidad. En la Figura 9 se muestra que el cien por ciento de los sujetos aceptaron los términos y condiciones del consentimiento informado.

**Figura 9**

*Resultados del análisis del consentimiento informado*


### Análisis descriptivo

El presente análisis descriptivo tiene como fin presentar la información de los resultados obtenidos en las pruebas y encuestas de los 4 cursos de Programación Orientada a Objectos, enfocado en los siguientes parámetros:

- Evaluar los conocimientos teóricos (Knowledge) de los 4 pilares de la programación orientada a objetos y los tres primeros principios SOLID (Single Responsibility / Open-Close / Liskov Sustitution), se verificó las calificaciones obtenidas del aprendizaje de los

dos grupos, el primero que recibieron instrucción con gamificación y el segundo que continuó con las clases convencionales.

- Evaluar el nivel de motivación obtenido mediante una encuesta cualitativa.
- Evaluar las habilidades obtenidas (Skills), aplicando una prueba práctica de los conocimientos adquiridos durante el experimento.

***Evaluación de conocimientos / Knowledge***

En la Tabla 13, se presentan los promedios de las calificaciones de cada uno de los pilares de programación orientada a objetos y los tres principios SOLID, para los 4 cursos en el presente estudio:

**Tabla 13**

*Resultados descriptivos de los promedios obtenidos de las pruebas de conocimiento*

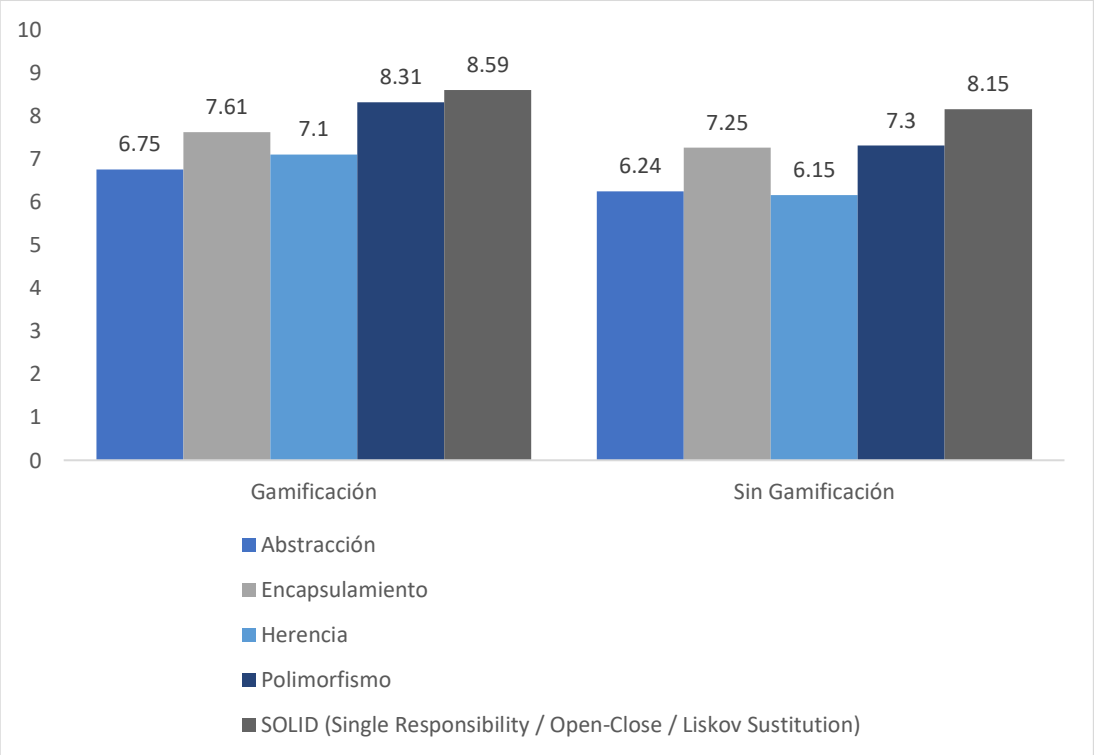
| Conocimientos / Knowledge | Experimentos | |
|---|---|---|
| | Gamificación | No Gamificación |
| **Abstracción** | 6.75 | 6.24 |
| **Encapsulamiento** | 7.61 | 7.25 |
| **Herencia** | 7.10 | 6.15 |
| **Polimorfismo** | 8.31 | 7.30 |
| **SOLid (Single Responsibility / Open-Close / Liskov Sustitution)** | 8.59 | 8.15 |

Se debe tomar en cuenta que la calificación máxima es diez y la mínima es cero. Para tener mayor claridad de los resultados se presenta el gráfico de los datos obtenidos de las calificaciones: Como se puede observar en la Figura 10, en todos los casos la calificación presentada en la evaluación teórica de los cuatro pilares de la programación orientada a objetos

y los principios SOLID para los estudiantes que no aprendieron mediante gamificación es menor. La diferencia no es notable, pues en el caso de la Abstracción, los promedios para gamificación y no gamificación fueron 6,75 y 6,24 respectivamente, sin embargo, esta diferencia implica los resultados globales de todos los 59 estudiantes evaluados con gamificación y 54 evaluados sin gamificación, es decir más de cien estudiantes. Posteriormente se analizará si esta diferencia en cada uno de los caos es significativa estadísticamente y permitirá determinar si la gamificación obtuvo mejores resultados.

**Figura 10**

*Resultados de los conocimientos teóricos*




***Evaluación de habilidades / Skill***

En la Tabla 14, se observan los resultados de la medición de las habilidades para los estudiantes que aprendieron con gamificación y sin gamificación. Se puede evidenciar, que tanto

en el aprendizaje de los 4 pilares de la programación orientada a objetos, como de los principios SOLid, los estudiantes con gamificación tuvieron mejores resultados.

**Tabla 14**

Resultados descriptivos de los promedios obtenidos de las pruebas de habilidades

| Habilidades / Skill | Experimentos | |
|---|---|---|
| | Gamificación | No Gamificación |
| **Cuatro pilares** | 7.84 | 7.56 |
| **Principios SOLID** | 8.04 | 7.62 |

***Encuesta de motivación***

La encuesta se enfocó en los cuatro pilares de la programación orientada a objetos y en los principios SOLid y permitió obtener los resultados que se encuentran en la Tabla 15:

**Tabla 15**

*Resultados descriptivos de los promedios obtenidos de las encuestas de motivación*

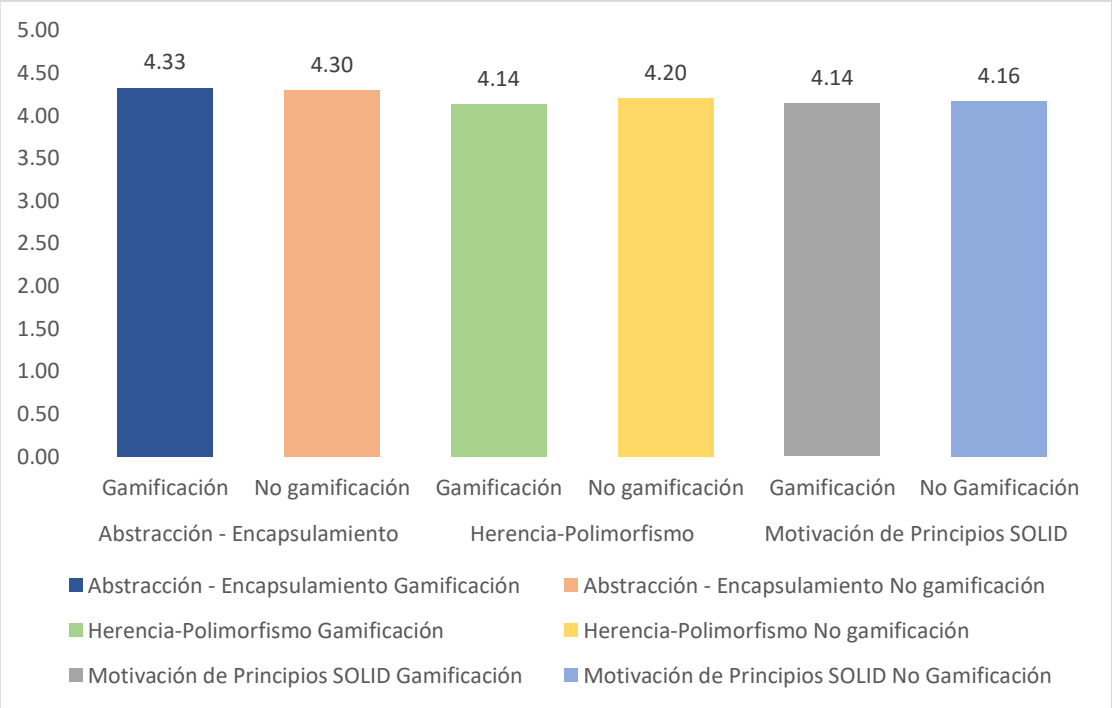
| | Conocimientos | Gamificación | No gamificación |
|---|---|---|---|
| **Motivación** | **Abstracción / Encapsulamiento** | 4.33 | 4.30 |
| | **Herencia / Polimorfismo** | 4.14 | 4.20 |
| | **Principios SOLID** | 4.14 | 4.17 |

Para este caso se puede observar en la Figura 11, que la motivación obtenida para los casos de Abstracción y Encapsulamiento tuvo un mayor valor para los estudiantes que tuvieron gamificación, sin embargo, para Herencia, Polimorfismo, y Principios SOLid, el resultado fue diferente, obteniendo una calificación menor para los estudiantes que recibieron gamificación. Cabe recalcar que las diferencias en los dos casos no son notables, se podría indicar que la

motivación no tuvo importantes diferencias para los dos grupos, pero posteriormente la estadística permitirá demostrar si existen o no diferencias significativas.

**Figura 11**

*Resultados de los conocimientos de motivación*




**Análisis comparativo**

***Análisis comparativo entre los resultados de conocimientos (Knowledge)***

En la Tabla 16, se encuentran los estadísticos descriptivos de los resultados de las evaluaciones teóricas por cada tema: abstracción, encapsulamiento, herencia, polimorfismo y de forma general los principios SOLid.

El grupo de conocimiento con la media más elevada es para gamificación con los principios SOLid, seguido de polimorfismo y el resto de pilares de programación orientada a objetos.

**Tabla 16**

*Estadísticos descriptivos por conocimiento*

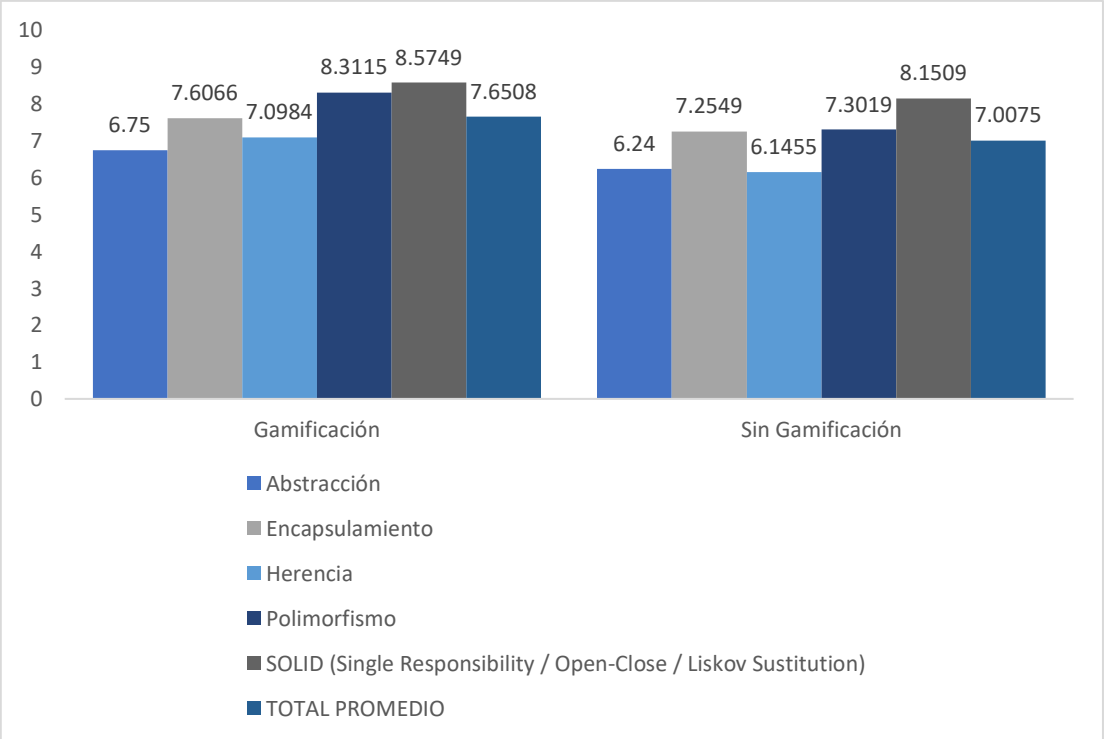
| Grupos por Conocimiento (Knowledge) | | Experimento | |
|---|---|---|---|
| | | **Gamificación** | **Sin Gamificación** |
| **Abstracción** | Media | 6.75 | 6.24 |
| | Cantidad | 59 | 54 |
| | Desviación estándar | 1.87 | 2.08 |
| | Error estándar | 0.2436 | 0.28340 |
| **Encapsulamiento** | Media | 7.6066 | 7.2549 |
| | Cantidad | 61 | 51 |
| | Desviación estándar | 2.0839 | 1.9062 |
| | Error estándar | 0.2668 | 0.2669 |
| **Herencia** | Media | 7.0984 | 6.1455 |
| | Cantidad | 61 | 55 |
| | Desviación estándar | 2.0712 | 2.2395 |
| | Error estándar | 0.2652 | 0.3019 |
| **Polimorfismo** | Media | 8.3115 | 7.3019 |
| | Cantidad | 61 | 53 |
| | Desviación estándar | 1.2320 | 1.5882 |
| | Error estándar | 0.1577 | 0.2181 |
| **SOLid (Single Responsibility / Open-Close / Liskov Sustitution)** | Media | 8.5849 | 8.1509 |
| | Cantidad | 53 | 53 |
| | Desviación estándar | 1.4201 | 1.7801 |
| | Error estándar | 0.1950 | 0.2445 |
| **PROMEDIO TOTAL** | Media | 7.6508 | 7.0075 |
| | Cantidad | 295 | 266 |

| Grupos por Conocimiento (Knowledge) | Experimento | |
|---|---|---|
| | **Gamificación** | **Sin Gamificación** |
| Desviación estándar | 1.8961 | 2.0631 |
| Error estándar | 0.1104 | 0.1265 |

En la Figura 12 se puede observar que las medias de gamificación para los temas de abstracción, encapsulamiento, herencia, polimorfismo y los 3 primeros principios SOLID son superiores al grupo de experimento sin gamificación en todos los casos.

**Figura 12**

*Medias agrupadas por experimento en la evaluación de conocimiento*




***Análisis comparativo entre los resultados de habilidades (Skills)***

En la Tabla 17 se encuentran los estadísticos descriptivos agrupados por experimento y habilidades, Al analizar los datos, es posible notar que los resultados obtenidos con gamificación

en promedio total son mejores que los resultados obtenidos por los estudiantes sin gamificación.

**Tabla 17**

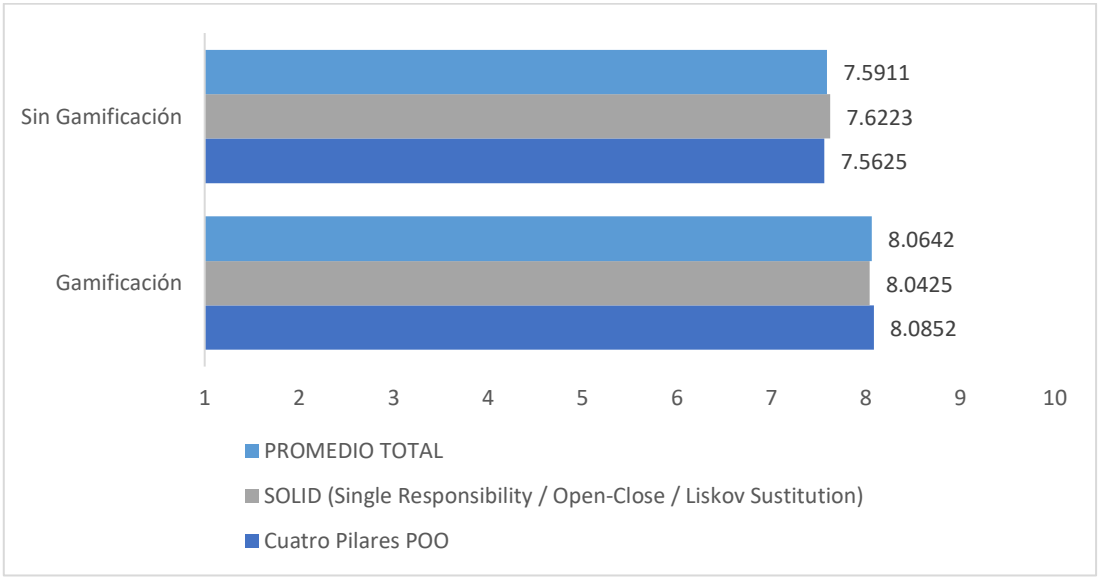
*Estadísticos descriptivos por habilidades*

| Grupos por Habilidades (Skill) | | Experimento | |
|---|---|---|---|
| | | **Gamificación** | **Sin Gamificación** |
| **Cuatro pilares POO** | Media | 8.0852 | 7.5625 |
| | Cantidad | 55 | 50 |
| | Desviación estándar | 1.5321 | 0.2066 |
| | Error estándar | 0.2066 | 0.2869 |
| **SOLid (Single Responsibility / Open-Close / Liskov Sustitution)** | Media | 8.0425 | 7.6223 |
| | Cantidad | 53 | 46 |
| | Desviación estándar | 2.0694 | 1.8569 |
| | Error estándar | 0.2842 | 0.2737 |
| **PROMEDIO TOTAL** | Media | 8.0642 | 7.5911 |
| | Cantidad | 108 | 96 |
| | Desviación estándar | 1.8073 | 1.9385 |
| | Error estándar | 0.1739 | 0.1978 |

Respecto a la diferencia entre experimentos en la evaluación de habilidades, se puede observar en la Figura 13 que tanto los Cuatro Pilares POO como SOLid, presentan mejores resultados con gamificación que los estudiantes sin gamificación.

**Figura 13**

*Medias agrupadas por experimento en la evaluación de habilidades*




***Análisis comparativo entre los resultados de motivación***

En la Tabla 18 se encuentra los estadísticos descriptivos distribuidos para los tres grupos de motivación y son los siguientes: Encapsulamiento / Abstracción, Herencia / Polimorfismo y SOLid (Single Responsibility / Open-Close / Liskov Sustitution). El grupo de motivación con la media más elevada con gamificación es Encapsulamiento / Abstracción, sin embargo, los resultados fueron diferentes para los otros casos. Cabe mencionar que las diferencias entre ambos experimentos no son notables.

**Tabla 18**

*Estadísticos descriptivos por motivación*

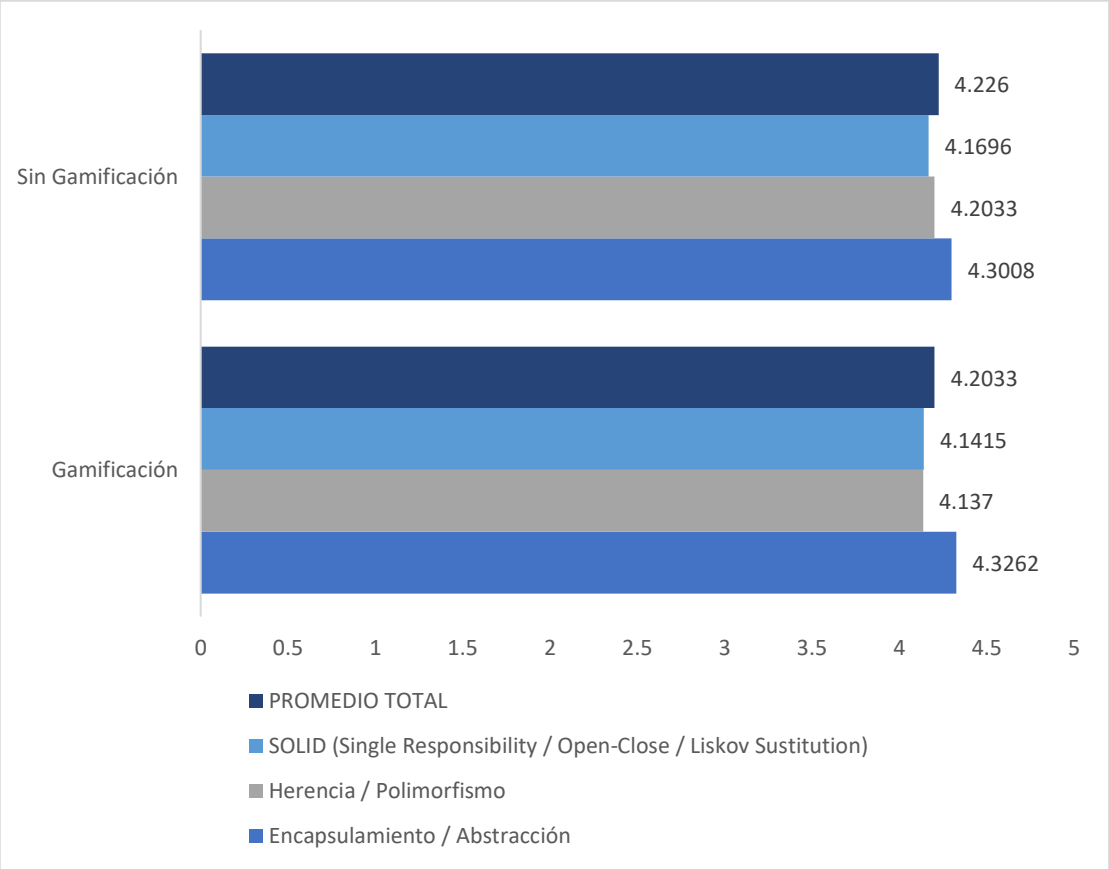
| **Grupos por Motivación** | | **Experimento** | |
|---|---|---|---|
| | | **Gamificación** | **Sin Gamificación** |
| **Encapsulamiento / Abstracción** | Media | 4.3262 | 4.3008 |
| | Cantidad | 60 | 52 |

| Grupos por Motivación | | Experimento | |
|---|---|---|---|
| | | **Gamificación** | **Sin Gamificación** |
| **Herencia / Polimorfismo** | Desviación estándar | 0.3272 | 0.4177 |
| | Error estándar | 0.0422 | 0.0579 |
| | Media | 4.1370 | 4.2033 |
| | Cantidad | 61 | 52 |
| **SOLID (Single Responsibility / Open-Close / Liskov Sustitution)** | Desviación estándar | 0.4553 | 0.4953 |
| | Error estándar | 0.0283 | 0.0686 |
| | Media | 4.1415 | 4.1696 |
| | Cantidad | 54 | 48 |
| **PROMEDIO TOTAL** | Desviación estándar | 0.4853 | 0.5187 |
| | Error estándar | 0.0660 | 0.0748 |
| | Media | 4.2033 | 4.2260 |
| | Cantidad | 175 | 152 |
| | Desviación estándar | 0.4330 | 0.4782 |
| | Error estándar | 0.0327 | 0.0387 |

En la Figura 14 se puede observar que las medias de gamificación para el grupo de motivación con Encapsulamiento / Abstracción son superiores a los estudiantes sin gamificación, sin embargo, el grupo de Herencia / Polimorfismo y SOLid (Single Responsibility / Open-Close / Liskov Sustitution) presenta resultados contrarios, donde los estudiantes sin gamificación presentan mejores resultados que con gamificación.

**Figura 14**

*Medias agrupadas por experimento en la encuesta de motivación*




**Verificación de hipótesis**

La Tabla 19 muestra los resultados previamente analizados y de los que se ha realizado la prueba de hipótesis presentando el valor t obtenido, la cantidad de datos para cada grupo, el promedio y el valor p que finalmente permite establecer si existe o no diferencia significativa bajo un nivel de significancia de 0.05.

***Prueba de hipótesis relacionada con el aprendizaje de conocimientos***

De acuerdo a los resultados obtenidos mediante la prueba T de Student para la comparación de medias entre el aprendizaje de conocimientos, habilidades y motivación se detalla en la Tabla 19, es posible rechazar la hipótesis nula $H_{0,1}$: $Auapc_{gam} = Auapc_{nongam}$, debido a

que para un intervalo de confianza del 95% se obtuvo un p-valor menor a 0.05, además de acepta la hipótesis alternativa $H_{1,1,2}$: $Auapr_{gam} > Auapr_{nongam}$ ya que el valor del estadístico de contrate t es mayor que 0.

**Tabla 19**

*Pruebas de muestras emparejadas por conocimiento*

| Comparación | Valor t | Recuento | Promedio | P-valor | Diferencia | SI / NO diferencia estadísticamente significativa (valor p<0.05) |
|---|---|---|---|---|---|---|
| **Abstracción gamificación vs no gamificación** | 1.358 | 59<br>54 | 6.7458<br>6.2407 | 0.177 | 0.5051 | No |
| **Encapsulamiento gamificación vs no gamificación** | 0.924 | 61<br>51 | 7.6066<br>7.2549 | 0.357 | 0.3517 | No |
| **Herencia gamificación vs no gamificación** | 2.381 | 61<br>55 | 7.0984<br>6.1455 | 0.019 | 0.9529 | Sí |
| **Polimorfismo gamificación vs no gamificación** | 3,750 | 61<br>53 | 8.3115<br>7.3019 | 0.000 | 1.0096 | Sí |
| **Principio SOLID gamificación vs no gamificación** | 1,386 | 53<br>53 | 8.5849<br>8.1509 | 0.168 | 0,434 | No |
| **Resultado total de las pruebas teóricas** | 1,386 | 295<br>266 | 7.6508<br>7.0075 | 0.000 | 0.64333 | Sí |

***Prueba de hipótesis relacionada con el aprendizaje de habilidades***

En la Tabla 20 se detallan los resultados de la comparación de medias mediante la prueba T para las medidas de motivación de los estudiantes donde se evidencia la posibilidad de aceptar la hipótesis nula $Aumoh_{gam} = Aumoth_{nongam}$ ya que no existe diferencia significativa que compruebe si los estudiantes con gamificación tienen una mejor de percepción conocimientos que el grupo sin gamificación, pero existe un valor promedio a favor de los estudiantes con gamificación.

**Tabla 20**

*Pruebas de muestras emparejadas por habilidades*

| Comparación | Valor t | Recuento | Promedio | P-valor | Diferencia | SI / NO diferencia estadísticamente significativa (valor p<0.05) |
|---|---|---|---|---|---|---|
| **Cuatro pilares GAM vs Cuatro Pilares NO GAM** | 1.498 | 55<br>50 | 8.0852<br>7.5625 | 0.137 | 0.52273 | No |
| **Principio SOL GAM vs Principio SOL NO GAM** | 1,056 | 53<br>46 | 8.0425<br>7.6223 | 0.693 | 0.42017 | No |
| **Resultados totales GAM vs Resultados totales NO GAM** | 1.803 | 108<br>96 | 8.0642<br>7.5911 | 0.073 | 0.47309 | No |

***Prueba de hipótesis relacionada con la motivación***

En la Tabla 21 se detallan los resultados de la comparación de medias mediante la prueba T para las medidas de motivación de los estudiantes donde se evidencia la posibilidad de aceptar la hipótesis nula $Aumot_{gam} = Aumot_{nongam}$ ya que no existe diferencia significativa que compruebe si los estudiantes con gamificación tengan una mejor motivación que el grupo sin gamificación, pero existe un valor promedio a favor de los estudiantes sin gamificación.

**Tabla 21**

*Pruebas de muestras emparejadas por motivación*

| Comparación | Valor t | Recuento | Promedio | P-valor | Diferencia | SI / NO diferencia estadísticamente significativa (valor p<0.05) |
|---|---|---|---|---|---|---|
| **Encapsulamiento / Abstracción gamificación vs no gamificación** | 0.36 | 60<br>52 | 4.3262<br>4.3008 | 0.72 | 0.02537 | No |
| **Herencia/Polimorfismo gamificación vs no gamificación** | -0.741 | 61<br>52 | 4.137<br>4.2033 | 0.46 | -0.06629 | No |
| **Principio SOLID gamificación vs no gamificación** | -0.283 | 54<br>48 | 4.1415<br>4.1696 | 0.778 | -0.02811 | No |
| **Resultado total de las pruebas prácticas** | 1.803 | 175<br>152 | 4.2033<br>4.226 | 0.073 | 0.47309 | No |

**Amenazas a la validez**

***Validez externa***

La validez está relacionada a la posibilidad de generalizar los resultados. Existen varias amenazas, mismas que se detallan a continuación:

- El uso de estudiantes como sujetos experimentales, que puedan no ser profesionales representativos de la industria de software. Sin embargo, las tareas de desarrollo no requieren experiencia profesional.
- Es importante que los experimentos incluyan una variedad de participantes con diferentes niveles de experiencia y habilidades en programación, y que se realice en diferentes contextos y entornos.

***Validez interna***

La validez interna se refiere al diseño y la implementación del experimento que sean apropiados. Se logra asegurándose de que los instrumentos utilizados para medir el aprendizaje de los estudiantes de programación sean válidos y fiables, que las condiciones del experimento sean controladas y equitativas, y que los procedimientos de análisis de datos sean apropiados y rigurosos.

Las amenazas a la validez interna fueron mitigadas, dentro de lo posible, mediante el diseño del experimento. Se utilizó grupos de sujetos balanceados y seleccionados aleatoriamente. La evaluación post experimento tanto teórica como practica revela que no existió dificultades para comprender los temas de programación orientada a objetos. Además, permitió confirma que los estudiantes percibieron de forma satisfactoria ambos experimentos.

***Validez de constructo***

Una forma de aplicar la validez del constructo es realizar un estudio que investigue la efectividad del programa de enseñanza en el aprendizaje de programación orientada a objetos.

En este caso, la validez de constructo se refiere a la medida en que las pruebas utilizadas para medir el conocimiento de programación realmente reflejan el conocimiento que se está requiriendo y no algún otro constructo. Para asegurar la validez de constructo en este estudio, los investigadores aplicaron lo siguiente:

- Consultar expertos en programación para confirmar que las pruebas utilizadas en el estudio midan realmente el conocimiento de programación.
- Asegurarse de que las pruebas incluyan tareas que requieran la aplicación del conocimiento de programación, en lugar de simplemente preguntar sobre los conceptos básicos.
- Utilizar métodos estadísticos rigurosos para analizar los resultados y asegurarse de que las pruebas sean fiables y válidas.

***Validez de la conclusión***

Para asegurar la validez de la conclusión en este experimento se utilizó un análisis comparativo de medias a través de la distribución T de Student, los investigares aplicaron lo siguiente:

- Utilizar un tamaño de muestras adecuado para garantizar que los resultados sean precisos y que se puedan generalizar a poblaciones más amplias.
- Verificar la homogeneidad de los sujetos comparados antes de realizar el análisis comparativo, para asegurarse de que los resultados sean justos y significativos.
- Realizar análisis adicionales, como análisis de regresión o análisis de varianza, para asegurarse de que las conclusiones sean precisas y que se haya controlado por otros factores que puedan afectar los resultados.
- Utilizar técnicas de visualización de datos como histogramas o gráficos, para detectar valores atípicos o patrones inusuales en los datos.

## Conclusiones

Mediante una revisión de contenidos y temas de la materia de programación orientada a objetos, se puede concluir que antes de aplicar la gamificación se debe seleccionar una estrategia lúdica, esto permitirá determinar si los temas y conceptos se pueden gamificar para mejorar la motivación y aprendizaje de los estudiantes. Además, es esencial asegurar que la gamificación sea apropiada y complementaria a los objetivos de aprendizaje de la materia, sin distraer o afectar la enseñanza de los contenidos fundamentales.

Se diseño un experimento de manera rigurosa con grupos de control medidas objetivas para evaluar el conocimiento, habilidad y motivación mediante una herramienta gamificada, así mismo fue importante considerar las variables individuales de los estudiantes, como la edad, carrera universitaria y el nivel de destrezas para minimizar algún posible sesgo y obtener resultados precisos. En general, el experimento proporcionó información valiosa sobre los beneficios de la gamificación y su potencial para percibir mejor el aprendizaje y la motivación de los programadores.

Al ejecutar el experimento gamificado con los parámetros apropiados se obtuvo datos precisos y verídicos, siguiendo los parámetros adecuados se logró obtener datos confiables que permitieron analizar los resultados y determinar el beneficio de la gamificación como estrategia pedagógica. Un experimento bien ejecutado es la base para tomar decisiones sobre la implementación de otros métodos técnicos de educación como la gamificación en el aprendizaje y la motivación de los estudiantes de programación orientada a objetos.

Se obtuvieron resultados favorables en la evaluación de conocimientos teóricos mediante el aprendizaje con gamificación, cumpliendo la hipótesis de que los estudiantes que utilicen la herramienta gamificada tendrán una mejor percepción de conocimientos teóricos de programación orientada a objetos, sin embargo, la motivación en ambos casos fue alta, sin obtenerse una diferencia significativa.

El presente experimento no permitió demostrar la hipótesis que los estudiantes se encuentran más motivados con la gamificación.

A lo largo de la presente investigación y en cumplimiento de los objetivos, se ha desarrollado una herramienta de apoyo basada en la gamificación, con el fin de mejorar la motivación y el aprendizaje de los principios de la programación orientada a objetos, en los alumnos de Ingeniería de Software y de Ingeniería de Tecnologías de la Información de la Universidad de las Fuerzas Armadas ESPE.

## Recomendaciones

Para revisar los conceptos y temas en la materia de programación orientada a objetos, se recomienda aprender sobre los principios básicos de la programación orientada a objetos y los conceptos clave de la gamificación.

Investigar ejemplos de gamificación en el aprendizaje de programación y cómo puede mejorar la experiencia de los estudiantes, la retención de conocimientos y aprender sobre elementos de juego como puntos, logros y desafíos.

La gamificación es una técnica poderosa, pero debe aplicarse de manera estratégica para ser efectiva y no descuidar los objetivos de aprendizaje.

Para diseñar experimentos con gamificación que aumenten el aprendizaje y motivación de los estudiantes de programación orientada a objetos, se recomienda definir objetivos claros, seleccionar elementos de juego relevantes, establecer recompensas claras y significativas, evaluar impacto mediante métricas y retroalimentación, y mejorar el diseño en base a los resultados.

Para ejecutar experimentos con gamificación, se recomienda desarrollar una herramienta que incluya actividades de juego desafiantes que integre recompensas y retroalimentación para mantener a los estudiantes motivados y enfocados en sus objetivos de

aprendizaje. Asegurarse que los desafíos estén alineados con los objetivos de enseñanza. Se recomienda utilizar la gamificación para que la programación sea divertida y retadora.

Se recomienda utilizar un enfoque estadístico riguroso. Una buena manera es realizar un estudio comparativo en el que se divida a los estudiantes en dos grupos: un grupo experimental con la gamificación y el otro grupo que siga un enfoque de enseñanza tradicional. Se recomienda utilizar otras técnicas estadísticas, por ejemplo, el análisis de varianza. Sin embargo, es importante considerar que hay muchos factores que pueden influir en el aprendizaje y la motivación, por lo que es importante tener un enfoque crítico al interpretar los resultados y considerar otros factores que puedan haber afectado el aprendizaje y la motivación.

La gamificación puede ser una forma excelente de fomentar la colaboración entre los estudiantes, lo que les permitirá aprender unos de otros. Esto puede lograr que los estudiantes trabajen juntos en equipo o compartan soluciones y estrategias. Se recomienda analizar y utilizar este tipo de estrategias educativas en las materias de programación de carreras relacionadas a ciencias la computación.

**Bibliografía**

Abdellatif, A. J., McCollum, B., & McMullan, P. (2018). Serious games: Quality characteristics evaluation framework and case study. *2018 IEEE Integrated STEM Education Conference (ISEC)*, 112-119. https://doi.org/10.1109/ISECon.2018.8340460

Azmi, S., Iahad, N., & Ahmad, N. (2015). *Gamification in online collaborative learning for programming courses: A literature review*. *10*, 18087-18094.

Bartel, A., & Hagel, G. (2014). Engaging students with a mobile game-based learning system in university education. *2014 IEEE Global Engineering Education Conference (EDUCON)*, 957-960. https://doi.org/10.1109/EDUCON.2014.6826215

Carreño-León, M., Sandoval-Bringas, A., Álvarez-Rodríguez, F., & Camacho-González, Y. (2018). Gamification technique for teaching programming. *2018 IEEE Global Engineering Education Conference (EDUCON)*, 2009-2014. https://doi.org/10.1109/EDUCON.2018.8363482

Díaz, S., Díaz, J., & Ahumada, D. (2018). A Gamification Approach to Improve Motivation on an Initial Programming Course. *2018 IEEE International Conference on Automation/XXIII Congress of the Chilean Association of Automatic Control (ICA-ACCA)*, 1-6. https://doi.org/10.1109/ICA-ACCA.2018.8609701

Funabiki, N., Korenaga, Y., Nakanishi, T., & Watanabe, K. (2013). An extension of fill-in-the-blank problem function in Java programming learning assistant system. *2013 IEEE Region 10 Humanitarian Technology Conference*, 85-90. https://doi.org/10.1109/R10-HTC.2013.6669019

Genero Bocco, M., Cruz Lemus, J. A., & Piattini Velthuis, M. (2015). *Métodos de investigación en ingeniería del software*. Ediciones de la U LTDA. http://190.57.147.202:90/xmlui/handle/123456789/2525

Gómez, O. S., Aguilar Vera, R., & Ucán Pech, J. (2018). *Experimentación en Ingeniería de Software* (p. 146).

*Ingeniería de Software*. (2020, mayo 5). ESPE. https://www.espe.edu.ec/carreras/ingenieria-de-software/

Ivanova, G., Kozov, V., & Zlatarov, P. (2019). Gamification in Software Engineering Education. *2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 1445-1450. https://doi.org/10.23919/MIPRO.2019.8757200

Jenkins, T. (2001). *The motivation of students of programming*. *33*, 53-56. https://doi.org/10.1145/377435.377472

Kučak, D., Bele, D., & Pašić, Đ. (2021). Climbing up the Leaderboard: An Empirical Study of Improving Student Outcome by Applying Gamification Principles to an Object-Oriented Programming Course on a University Level. *2021 44th International Convention on Information, Communication and Electronic Technology (MIPRO)*, 527-531. https://doi.org/10.23919/MIPRO52101.2021.9596709

Kumar, B., & Sharma, K. (2018). A Gamified Approach to Achieve Excellence in Programming. *2018 4th International Conference on Computing Sciences (ICCS)*, 107-114. https://doi.org/10.1109/ICCS.2018.00026

Lahtinen, E., Ala-Mutka, K., & Järvinen, H.-M. (2005). A study of the difficulties of novice programmers. *ACM SIGCSE Bulletin*, *37*(3), 14-18. https://doi.org/10.1145/1151954.1067453

Layth Khaleel, F., Ashaari, N., & Tengku Wook, T. S. M. T. W. (2019). An empirical study on gamification for learning programming language website. *Jurnal Teknologi*, *81*. https://doi.org/10.11113/jt.v81.11133

Lazzaro, N. (s. f.). *Why We Play Games: Four Keys to More Emotion in Player Experiences*. 46.

Marín, B., Frez, J., Portales, U. D., Cruz-Lemus, J., & Genero, M. (s. f.). An Empirical Investigation on the Benefits of Gamification in Programming Courses. *ACM Transactions on Computing Education*, *19*(1), 22.

Marti-Parreño, J., Seguí Mas, D., & Seguí-Mas, E. (2016). Teachers' Attitude towards and Actual Use of Gamification. *Procedia - Social and Behavioral Sciences*, *228*, 682-688. https://doi.org/10.1016/j.sbspro.2016.07.104

Mi, Q., Keung, J., Mei, X., Xiao, Y., & Chan, W. K. (2018). A Gamification Technique for Motivating Students to Learn Code Readability in Software Engineering. *2018 International Symposium on Educational Technology (ISET)*, 250-254. https://doi.org/10.1109/ISET.2018.00062

Milne, I., & Rowe, G. (2002). Difficulties in Learning and Teaching Programming—Views of Students and Tutors. *Education and Information Technologies*, *7*, 55-66. https://doi.org/10.1023/A:1015362608943

Mora, A., Riera, D., Gonzalez, C., & Arnedo-Moreno, J. (2015). A Literature Review of Gamification Design Frameworks. *2015 7th International Conference on Games and Virtual Worlds for Serious Applications (VS-Games)*, 1-8. https://doi.org/10.1109/VS-GAMES.2015.7295760

Sarcar, V. (2021). Know SOLID Principles. En V. Sarcar (Ed.), *Simple and Efficient Programming with C#: Skills to Build Applications with Visual Studio and .NET* (pp. 49-108). Apress. https://doi.org/10.1007/978-1-4842-7322-7_4

Simões Gomes, T. C., Pontual Falcão, T., & Cabral de Azevedo Restelli Tedesco, P. (2018). Exploring an approach based on digital games for teaching programming concepts to young children. *International Journal of Child-Computer Interaction*, *16*, 77-84. https://doi.org/10.1016/j.ijcci.2017.12.005

Soepriyanto, Y., & Kuswandi, D. (2021). Gamification Activities for Learning Visual Object-Oriented Programming. *2021 7th International Conference on Education and Technology (ICET)*, 209-213. https://doi.org/10.1109/ICET53279.2021.9575076

Soflano, M., Connolly, T. M., & Hainey, T. (2015). An application of adaptive games-based learning based on learning style to teach SQL. *Computers & Education*, *86*, 192-211. https://doi.org/10.1016/j.compedu.2015.03.015

Steinmann, A., Bosch, B., & Aiassa, D. (2013). Motivación y expectativas de los estudiantes por aprender ciencias en la universidad: Un estudio exploratorio. *Revista mexicana de investigación educativa*, *18*, 585-598.

Tsai, C.-Y. (2019). Improving students' understanding of basic programming concepts through visual programming language: The role of self-efficacy. *Computers in Human Behavior*, *95*, 224-232. https://doi.org/10.1016/j.chb.2018.11.038

Uskov, V., & Sekar, B. (2014). Gamification of software engineering curriculum. *2014 IEEE Frontiers in Education Conference (FIE) Proceedings*, 1-8. https://doi.org/10.1109/FIE.2014.7044098

Vahldick, A., Mendes, A. J., & Marcelino, M. J. (2014). A review of games designed to improve introductory computer programming competencies. *2014 IEEE Frontiers in Education Conference (FIE) Proceedings*, 1-7. https://doi.org/10.1109/FIE.2014.7044114

Verdesoto, G. J. Z., Mora, K. G. R., & Torres, L. H. G. (2018). Análisis de la deserción estudiantil en las universidades del ecuador y américa latina. *Revista Pertinencia Académica. ISSN 2588-1019*, *8*, Art. 8.

Vos, T. E. J., Fraser, G., Martinez-Ortiz, I., Prada, R., Silva, A. R., & Prasetya, I. S. W. B. (2020). Tutorial on a Gamification Toolset for Improving Engagement of Students in Software Engineering Courses. *2020 IEEE 32nd Conference on Software Engineering Education and Training (CSEE&T)*, 1-3. https://doi.org/10.1109/CSEET49119.2020.9206212

Zapušek, M., & Rugelj, J. (2013). Learning programming with serious games. *EAI Endorsed Transactions on Game Based Learning*, *13*(01), 8.

Zatarain Cabada, R., & Zatarain Cabada, R. (2018). Reconocimiento afectivo y gamificación aplicados al aprendizaje de Lógica algorítmica y programación. *Revista electrónica de investigación educativa*, *20*(3), 115-125. https://doi.org/10.24320/redie.2018.20.3.1636

# Apéndice