UNIVERSIDAD SEÑOR DE SIPÁN

ESCUELA DE POSGRADO

TESIS

ESTRATEGIA DE AUTOAPRENDIZAJE DE LA PROGRAMACION, SUSTENTADA EN UN MODELO DE GAMIFICACION INTEGRAL CONTEXTUALIZADO

PARA OPTAR EL GRADO ACADÉMICO DE DOCTOR EN EDUCACION

Autor:

Vives Garnique Luis Alberto

ORCID

https://orcid.org/0000-0003-0280-2990

Asesor:

Dr. Callejas Torres Juan Carlos

ORCID

Orcid.org/0000-0001-8919-1322

Línea de Investigación:

Educación y Calidad

Pimentel – Perú

2021

UNIVERSIDAD SEÑOR DE SIPÁN
ESCUELA DE POSGRADO

DOCTORADO EN EDUCACION

ESTRATEGIA DE AUTOAPRENDIZAJE DE PROGRAMACION, SUSTENTADA EN UN MODELO DE GAMIFICACION INTEGRAL CONTEXTUALIZADO

Autor:

Vives Garnique Luis Alberto

Pimentel – Perú

2021

ESTRATEGIA DE AUTOAPRENDIZAJE DE PROGRAMACION, SUSTENTADA EN UN MODELO DE GAMIFICACION INTEGRAL CONTEXTUALIZADO

APROBACIÓN DE LA TESIS

_____

Asesor Metodológico

_____

Presidente del jurado de tesis

_____ _____

Secretaria del jurado de tesis **Vocal del jurado de tesis**

_____ _____

Secretaria del jurado de tesis **Vocal del jurado de tesis**

Dedicatorias

A mis padres Haydee y Juan por haberme forjado como la persona que soy en la actualidad; muchos de mis logros se los debo a ustedes. Me formaron y motivaron para alcanzar mis anhelos.

A mis hermanos Mary, Juan, Julio, Carlos y mis sobrinos por estar siempre presentes en todas mis etapas académicas y por ser los referentes por seguir siempre.

A mi esposa Carla, y mis hijos Julián Alessandro y Liam Mauricio por ser el impulso motivador en cada caída, los que me sostuvieron en los momentos difíciles y me alentaron a continuar con este logró.

Agradecimientos

Un agradecimiento muy especial a mi maestro Juan Carlos Callejas por creer siempre en mi y por estar pendiente en mi evolución académica. Gracias por los consejos para mejorar no solamente en el ámbito académico, sino también en lo personal y profesional.

Resumen

El proceso formativo de un Ingeniero de software se base en la habilidad y conocimiento para codificar software de calidad bajo estándares internacionales. Sin embargo, se evidencia una brecha entre el conocimiento teórico - práctico brindado en las Universidades que traer como consecuencia la deserción y la desmotivación de los estudiantes. En esta investigación abordamos el problema de la insuficiencia apropiación de contenidos de programación que limitan la codificación en el desarrollo de software. Se diseño un modelo de autoaprendizaje de la programación en la formación de ingenieros de software haciendo uso de técnicas de gamificación para la codificación en el desarrollo de software y se elaboró una estrategia de autoaprendizaje de la programación estructurada en 2 etapas 6 fases, basada en el modelo diseñado. Para la validación de la investigación se usó el método de preexperimental con 120 estudiantes. Los resultados obtenidos posterior a la aplicación del post test permiten concluir que el 87.5% en promedio comprenden e identifican la importancia de cada etapa y sus fases propuestas en la estrategia de autoaprendizaje de la programación.

Palabras Clave: Autoaprendizaje de la programación, ingeniería de software, programación, gamificación

Abstract

The training process of a Software Engineer is based on the ability and knowledge to code quality software under international standards. However, there is evidence of a gap between the theoretical - practical knowledge provided in the Universities that results in the desertion and demotivation of the students. In this research we address the problem of insufficient appropriation of programming content that limits coding in software development. A self-learning programming model was designed in the training of software engineers making use of gamification techniques for coding in software development and a self-learning strategy of structured programming was developed in 2 stages 6 phases, based on the designed model. For the validation of the research, the pre-experimental method was used with 120 students. The results obtained after the application of the post test allow us to conclude that 87.5% on average understand and identify the importance of each stage and its proposed phases in the self-learning strategy of programming.

Keywords: Programming self-learning, software engineering, programming, gamification

Índice

I. INTRODUCCIÓN

1.1. Realidad Problemática.

La educación universitaria en el territorio peruano busca mejorar contantemente la calidad de sus procesos con el objetivo de ser competitivas y brindar un producto de calidad (profesionales) para el beneficio de la comunidad empresarial, científica y académica.

En tal sentido la educación universitaria peruana pasa por el desafío actual de preparar a un individuo, de manera que se confirme lo teórico - práctico esencialmente necesarios para generar capacidades cognitivas, creativas e innovadoras, con el objetivo de dotar a los profesionales con pensamiento crítico, productivo, creativo, científico e innovador.

El paradigma constructivista-cognitivo de enseñanza-aprendizaje predomina en la educación superior, donde la dinámica del proceso de enseñanza-aprendizaje se controla como un proceso colaborativo e interactivo de la asignatura, con el objetivo de que el sujeto que la aprende se apropie y construya el conocimiento (Calle, 2021; Pinillos & Santa Cruz, 2021). Como consecuencia tenemos un aprendizaje significativo que perdura en el tiempo. El aprendizaje consiste en que los individuos construyan su propio conocimiento con el objetivo de aprender a comprender diversas alternativas de solución basadas en el conocimiento previamente adquirido, se identifican a los docentes, técnicas pedagógicas, herramientas pedagógicas y el dialogo como medios de transferencia de conocimiento (C. J. S. Moreira et al., 2021). El conocimiento evoluciona y no permanece estático, su construcción se basa en la experiencia que es adquirida por el individuo (Araceli & Barajas, n.d.). El aprendizaje significativo en un individuo se alcanza cuando el sujeto toma importancia en

el conocimiento adquirido y lo aplica de acuerdo con una necesidad personal o social. Es por eso, que los docentes juegan un papel importante que conlleva a una responsabilidad permanente para desarrollar habilidades y capacidades evolutivas que permitan guiar el conocimiento y desempeño trasmitido a sus estudiantes (Ortega-ruipérez, 2021). Así mismo, existen estrategias que permiten dar apoyo y sustento al aprendizaje: (i) el aprendizaje basado en problemas o proyectos, (ii) el aprendizaje colaborativo y, (iii) el aprendizaje basado en componentes (Davidson & Major, 2014).

En esta Actualidad, las Tecnologías de la Información y de las Comunicaciones – TICs, se consideran importantes porque permiten desarrollar entornos de enseñanza-aprendizaje con el objetivo de dar soporte a las diferentes estrategias de aprendizaje que existen para guiar al sujeto su propio descubriendo del conocimiento (Tang Qin et al., 2021).

La Universidad Señor de Sipan y su carrera de Ingeniería de Sistemas, forman profesionales capaces de analizar, diseñar, desarrollar e implementar software, haciendo uso de estándares internacionales. La programación cumple un rol fundamental y es la base de la formación de ingenieros de sistemas, en tal sentido constituye una disciplina de corte académico, siendo el sustento fundamental para otros cursos en la línea o área de programación, en los estudiantes es una herramienta para aprender los conceptos, reglas, técnicas, métodos y estrategias que permitan la solución de problemas reales que se presentan en el campo profesional, lo que requiere de una alta actividad cognitiva, innovadora y creativa de los estudiantes.

La programación como área académica abarca asignaturas básicas como subsistema, tales como programación 1, programación 2, algoritmos y estructura de datos, complejidad algorítmica. Se realizó un estudio empírico que permitió corroborar que en el area de programación los estudiantes presentan mayores dificultades de aprendizaje. Se analizaron los rendimientos académicos de los estudiantes del curso de programación I, de los 2017, 2018, y 2019, los cuales fueron 65%, 73%, y 55%, de aprobados respectivamente, en contraposición, en la mayoría de los cursos los estudiantes presentan un mejor rendimiento que se evidencia en mejores resultados cuantitativos y se evidencia una motivación hacia la carrera.

Del análisis anterior se observa una serie de insuficiencias que este trabajo trata de paliar y que da lugar a un diagnóstico factico:

- Los estudiantes atraviesan dificultades al momento de transformar un problema de lenguaje natural a lenguaje de programación.
- Se evidencia la falta de logica de programación debido a la falta de aplicación de conceptos en la solución de problemas en contextos reales.
- Constantemente se evidencia la carencia de análisis de casos para la construcción de algoritmos y programación apropiados para la programación y solución de un problema dado.
- Se aprecia la falta de sistematización en el análisis, diseño y desarrollo de software educativo, siendo la fase del desarrollo la que presenta mayor dificultad.

Se aprecia que los elementos anteriormente descritos permiten declarar el **problema científico**, insuficiencias en la apropiación de los contenidos de programación, limitan la codificación en el desarrollo y mantenimiento de

software, expresión científica de la **contradicción epistemológica inicial** reflejada entre la apropiación de contenidos de programación y los requerimientos formativos de futuros ingenieros de software, la apropiación de contenidos de programación en el desarrollo de software es el resultado que debe lograrse en el proceso de autoaprendizaje del ingeniero de software, mientras que la demanda de futuros profesionales de Ingeniería de Sistemas está vinculadas a la necesidad de que se de acuerdo con las demandas de una sociedad que siempre se está desarrollando científica y tecnológicamente.

En la actualidad se manifiesta una incoherencia entre la actual el proceso de enseñanza que limita al proceso de autoaprendizaje y se refleja en las exigencias formativas de ingeniero de software.

Para comprobar las causas que dan origen a la problemática se realizaron cuestionarios de encuestas y entrevistas a los estudiantes y profesores de la USS, específicamente de la carrera de Ingeniería de Sistemas, y se evidencio que los docentes no realizan un diagnóstico previo de los alumnos, basado en conocimientos, habilidades y destrezas que poseen los estudiantes para apropiarse de los nuevos contenidos académicos, aunque se haya realizado dicho diagnóstico, las variantes utilizadas, en cuanto a métodos o medios, no garantizan que todos los estudiantes logren la competencia académica.

Otra causa es producida por las concepciones teóricas y prácticas del proceso de autoaprendizaje de la programación que no consideran la sistematización como vía fundamental en el logro de la apropiación de los contenidos en el contexto.

Así mismo se atribuye como causa del problema a la orientación didáctico-metodológica del trabajo independiente para la apropiación de los contenidos de programación, la cual, carece de un soporte tecnológico que se constituye en mediador didáctico capaz de representar procesos simulados que estimulen el aprendizaje.

Por su parte la concepción teórico-práctico de los mediadores didácticos para el estudio independiente de la programación presenta limitaciones relacionados con la interactividad, retroalimentación y visualización de procesos, lo cual no reúne un sustento metodológico educacional para su desarrollo.

Por lo tanto, es necesario profundizar en el proceso de autoaprendizaje de la programación, **objeto de estudio** de la presente investigación.

De esta manera el análisis praxeológico y epistemológico del objeto y campo de la investigación permite descubrir, como **fisura epistemológica:** insuficientes aportes teóricos y metodológicos vinculados con la sistematización formativa interactiva y la lógica del autoaprendizaje del contenido de la programación precedente en el proceso de autoaprendizaje del ingeniero de sistemas, debido a la falta de un enfoque de integración entre los contenidos brindados y una plataforma de autoformación virtual.

El objetivo de esta investigación es elaborar una estrategia de autoaprendizaje de la programación para la formación del ingeniero de sistemas, sustentado en un modelo de gamificación contextualizado.

En la fundamentación metodológica y epistemológica del campo de acción y objeto de estudio se establece la brecha epistemológica creada por la

contradicción dialéctica entre la lógica formal interactiva de esta disciplina y la lógica del autoaprendizaje de nuevos contenidos.

Teniendo en cuenta lo anterior se sustenta como hipótesis de investigación:

Si se elabora una estrategia de autoaprendizaje de la programación en la formación del ingeniero de sistemas, mediante el empleo de estrategias de gamificación, que tenga en cuenta la relación entre la lógica de la sistematización autoguiada – interactiva y la lógica la autoformación gamificada de contenidos de programación en el desarrollo de software, entonces se contribuye la apropiación de los contenidos de programación.

Por lo que es necesario y relevante profundizar en la dinámica del proceso de autoaprendizaje de la programación **campo de acción** de la presente investigación.

1.2. Trabajos previos

a) Antecedentes del estudio

El desarrollo de aplicaciones tecnológicas utilizando la computadora se fundamenta en la enseñanza y el aprendizaje de la programación. Aprender y enseñar programación es una tarea compleja tanto para estudiantes como docente. En (Lister, 2020) indican que el principal fracaso en la introducción a la programación a estudiantes del nivel 1, es suponer que el estudiante domina el razonamiento abstracto de llevar un problema de la vida real a codificar líneas de códigos para resolverlo.

Según (Abbasi et al., 2021) para tener éxito en el primer curso de programación (programación 1, fundamentos de programación o

introducción a la programación), los estudiantes deben ser capaces de interpretar una solución específica para un problema usando el razonamiento abstracto que le permita codificar una solución, poniendo en práctica aspectos teóricos de la programación tales como sintaxis básicas, funciones, estructuras de control, estructuras repetitivas, arreglos unidimensionales y bidimensionales. Sin embargo, los estudiantes encuentran una limitación entre lo aprendido teóricamente y la aplicación práctica en la resolución de ejercicios que se evidencia generalmente al querer comprobar sus resultados o querer seguir practicando nuevos ejercicios que involucren retos en diferentes niveles.

Por su parte (Sobral, 2021b) realiza un mapeo sistemático acerca de las estrategias de enseñanza de la programación encontrando 95 estudios primarios donde se destacan las estrategias de Aprendizaje basado en proyectos (Robberts & Employability, 2019), gamificación (Manzano-León et al., 2021), aprendizaje activo (F. Moreira et al., 2021), programación por parejas (Sobral, 2021a), aula invertida (Calle, 2021; Campbell et al., 2014).

Según (Gomes et al., 2019) determina que los estudiantes de los cursos de programación poseen una motivación de logro personal, seguido de una motivación extrínseca, sin embargo la mejor motivación que ellos experimentan es el acompañamiento en todo el proceso de aprendizaje. Sienten seguridad cuando son acompañados en el logro de la solución de un problema o en verificar los errores que van cometiendo. Sin embargo, el docente no puede estar permanentemente con el estudiante generando desmotivación en el curso.

La programación es una competencia que se adquiere en los cursos de las carreras afines a computación, es decir Ingeniería de Software, y Ciencias de la Computación, Ingeniería de Sistemas de Información, quienes sustentan su plan curricular en la Computing Curricula 2001 (JTS-CC2005, 2006), informe actualizado y elaborado por la comisión de trabajo integrada por la Association for Computing Machinery (ACM) y la Sociedad de Computación del Institute of Electrical and Electronics Engineer (IEEE-CS). En (JTS-CC2005, 2006) recomienda 5 planes de estudios: i) Ingeniería de Computadoras (CE de Computer Engineering), ii) Ciencias de la Computación (CS de Computer Science), iii) Sistemas de Información (IS de Information System), iv) Tecnologías de la Información (IT de Information Technology); e, v) Ingeniería de Software (SE de Software Engineering). Posteriormente se genera una versión actualizada denominada Computing Curricula 2020 (Impagliazzo et al., 2018; Takada et al., 2020) donde se proponen 7: Ingeniería de Computadoras (CE), Ciencias de la Computación (CS), Ciberseguridad (CSEC), Ciencia de Datos (DS de Data Science), Sistemas de Información (IS), Tecnología de la Información (IT) e Ingeniería de Software (SE).

En el Perú no se evidencia un interés por mejorar o superar las barreras de aprendizaje en la programación, así mismo en la institución los docentes realizan el mayor esfuerzo por brindar conocimiento y motivar a los estudiantes de una manera empírica y no sistematizada.

b) Investigaciones previas

En (Malik et al., 2021) se encontró que los que los estudiantes del curso de programación de la Business and Technology University, aprenden con el método tradicional, resuelven ejercicios propuestos, sin embargo no resuelven problemas reales basado en proyectos, como consecuencia los problemas aprendidos no son aplicables en un futuro campo laboral. Por otro lado, se identificó que los conceptos de programación son trasmitidos con estrategias de memorización y repetición lo cual dificulta la manera de pensar del estudiante al momento de codificar problemas. Para lo cual propusieron una metodología de enseñanza centrada en el objeto (alumno), donde un influyente actúa como monitor del desempeño del objeto, llamado sujeto (docente) y la aplicabilidad en un entorno real, universidad (entorno). Aplicaron su estudio en 12 aulas con 25 alumno y lograron comprobar que los estudiantes obtuvieron entre 10 - 15 % de mejora en rendimiento y conocimiento adquirido en el curso de programación.

Por su parte, en (Kesler et al., 2021) buscaron identificar el rol de la tecnología en la evolución de la pedagogía tradicional. Se centra en la brecha que existe entre los conceptos aprendidos y la acción de codificar, donde el estudiante no interioriza los conceptos y como consecuencia se enfrenta a problemas distantes de su ámbito de conocimiento. Se propuso relacionar las perspectivas constructivista e instruccionista y su relación con la tecnología. Para lo cual, realizaron un caso de estudio y entrevista a 98 docentes y 96 estudiantes, y evidenciaron una mayor motivación de

los estudiantes por aprender a programar usando scratch como herramienta tecnológica bajo la perspectiva constructivista. Se destaca como trabajo futuro el uso de metodológicas de aprendizaje como pensamiento visual, aprendizaje basado en proyectos.

Por otra parte, en (Franco-González et al., 2020) se encontró que las herramientas tecnológicas rara vez son usadas en la enseñanza de los contenidos del curso de programación debido a la inconsistencia curricular de los temas a tratar. Se propuso usar scratch como herramienta para la enseñanza y motivación de la programación, aplicando 2 pre-test y 2 post-test distribuidos en 58 estudiantes y concluyeron que la tendencia va en una enseñanza aprendizaje centrada en el estudiante, en trabajo colaborativo, aprendizaje y monitoreo de lo aprendido y se debe disminuye el uso de programación estructurada para pasar a programación orientada a objetos.

Mientras que, en (Becker & Quille, 2019) realizaron una revisión de la evolución del curso de introducción a la programación por los 50 años de la existencia del curso a nivel mundial. Se destaca lo siguiente: i) Los paradigmas de programación han variado y actualmente se usa la programación orientada a objetos en los cursos de nivel 1, ii) Los lenguajes de programación ha evolucionado y se destaca el uso de C++, java, Python para la enseñanza de la programación, ii) Aún persiste el problema de aprender conceptos y evaluarlos cognitivamente, pero se aprecia un crecimiento en el uso de estilos de aprendizaje, tutoriales y evaluaciones automáticas, exámenes en línea, iii) Enseñar por medio del uso de hardware (robots, raspberry, arduinos) es muy utilizado en escuelas

primarias y secundarias. Así mismo crece el interés por nuevas estrategias de enseñanza como aula invertida, aprendizaje basado en proyectos, gamificación. Una de las conclusiones resaltantes es que Scratch es una herramienta que se apoya en la enseñanza lúdica, para aprender conceptos de programación, y es muy recomendada para estudiantes de colegio. En entornos universitarios has sido poco usada.

Asimismo, en (Luxton-Reilly et al., 2018) realizó una revisión sistemática de la literatura acerca de los cursos de introducción a la programación, se concluye que los cursos de programación proponen: (i) el uso de muchas herramientas tecnológicas, sin embargo, no se ha logrado comprobar técnicamente el uso de cada uno de ellas, (ii) se continúa debatiendo sobre que enseñar en el curso de programación (orientado a objetos o estructurado), (iii) se evidencia la ansiedad y frustración de los estudiantes del curso de programación y, (iv) se propone documentar las mejores experiencias en la enseñanza aprendizaje de la programación y compartirla con la comunidad académica.

Finalmente, en (Luxton-Reilly, 2016) encontraron que existe una ruptura entre lo que se enseña en el curso de programación, como se enseña y el objetivo del curso. Encontraron que el estudiante tiene un concepto errado de que la programación es difícil y que se trasmite de generación en generación. La cantidad de contenidos y el poco tiempo que el estudiante tiene para aprenderlos genera frustración en los estudiantes, que concluye en altas tasas de deserción, siendo el mayor impacto en las mujeres quienes rápidamente dejan la carrera debido a las frustraciones y estrés que genera la enseñanza del curso. Según el estudio de (Luxton-Reilly,

2016) el enfoque o perspectiva instruccionista busca centrar el conocimiento en solo una respuesta correcta a cada pregunta. Mientras que el enfoque constructivista permite tener muchas respuesta validas y comprobadas para una pregunta. Se concluyó que el constructivismo es el enfoque que mejor se adapta al uso de herramientas tecnológicas en la enseñanza de la programación porque permite al docente liderar el uso de la tecnología y brindarle al estudiante una herramienta que le permita construir su propio conocimiento.

1.3. Teorías relacionadas al tema.

1.3.1. Proceso de autoaprendizaje de la programación y su dinámica

Según Volet, citado en (Nurulain Mohd Rum & Zolkepli, 2018) define al proceso de autoaprendizaje como el proceso de aprender a aprender basado en el razonamiento, percepción y memorización del objeto de estudio y que se apoya en actividades de verificación, planificación, selección, seguimiento, evaluación y revisión de conceptos teóricos prácticos que reafirmen el objetivo de aprendizaje.

Del mismo modo, según (Holz-Clause et al., 2015) el proceso de autoaprendizaje se centra en el proceso de evaluar las necesidades de aprendizaje de un estudiante, con el objetivo de establecer estrategias, planes de acciones y obtener resultados al finalizar el proceso que posteriormente sirvan para retroalimentar el conocimiento adquirido.

Por su parte, según (Abreu Alvarado et al., 2018) considera que el estudiante es el centro del proceso de autoaprendizaje, teniendo como facilitador al docente, durante todo el proceso de construcción de conocimiento, siendo la interacción con el entorno, compañeros y otros

docentes, lo que le da la capacidad de experimentar y reflexionar acerca de lo que va aprendiendo.

Así mismo, según (Barcia & Carvajal, 2015) en el proceso de autoaprendizaje donde participan el docente y los estudiante es fundamental una adecuada planificación teniendo en cuenta los componentes tales como objetivo, contenido, métodos, medios de enseñanza y la evaluación del aprendizaje. Cada componente contribuye al proceso de enseñanza aprendizaje del estudiante.

Por otro lado, según (D. Lescay et al., 2015) el proceso de autoaprendizaje debe tomar en cuenta las actividades vivenciales y sociales del estudiante con el propósito de conceptualizar y aplicar lo aprendido en su mismo entorno, de esta manera el estudiante toma conciencia de la relación directa que posee los conceptos teóricos y su aplicabilidad mediante la práctica. En ese sentido, el modelo Holística-Configuracional (Molina & Mejia, 2013) brinda un enfoque que permite enlazar dimensiones propias del proceso de enseñanza aprendizaje (diseñar, construir, producir teorías, conceptualizar y evaluar) resultantes de las acciones o actividades vivenciales y sociales, como resultado se obtiene un proceso cíclico que permite descubrir problemáticas recurrentes y propuestas de acción para mejorar el proceso de autoaprendizaje.

1.3.2. Proceso de autoaprendizaje de la programación en la formación del ingeniero de software, mediante el empleo de herramientas tecnológicas y técnicas de gamificación.

El objetivo de enseñar a programar cualquier dispositivo tecnológico es el núcleo de la formación de un ingeniero de software. Programar implica adquirir conocimiento y habilidades abstractas que permiten trasladar un problema real a un mundo abstracto donde la solución se representa en códigos binarios de ceros y unos. Según (Malik et al., 2021) en su investigación encontró que los estudiantes del curso de programación de la Business and Technology University, aprenden con el método tradicional, resuelven ejercicios propuestos, sin embargo generan problemas reales basado en proyectos, como consecuencia los problemas aprendidos no son aplicables en un futuro campo laboral. Como propuesta de solución (Malik et al., 2021) propusieron una metodología enseñanza centrada en el objeto (alumno), un influyente como monitor del desempeño del objeto, llamado sujeto (docente) y la aplicabilidad en un entorno real, universidad (entorno), como resultado obtuvieron una mejora entre el 10 – 15% en el rendimiento y conocimiento adquirido en el curso de programación aplicado a 300 estudiantes.

Asimismo, según (Rojas-López & García-Peñalvo, 2020) buscaron establecer una relación entre el contenido del curso de programación y las habilidades del pensamiento computacional (abstracción, generalización, descomposición, diseño de algoritmos y evaluación), para lo cual distribuyeron 255 estudiantes en 5 semestres académicos y se evidencio que 72% de estudiantes logran la habilidad de diseño algorítmico, 39% las

habilidades de descomposición, 35.54% logran la habilidad de generalización, la habilidad de abstracción alcanza un entendimiento el 64.7%. Concluyeron que hay que reforzar conceptos basados en expresiones aritméticas, lógicas relacionales para lograr la habilidad de generalización y reforzar lo aprendido con problemas complejos de entorno real para lograr la habilidad de descomposición.

Además, según (Soler Pellicer et al., 2020) encontraron que los problemas pedagógicos pueden tener la raíz en la falta de compresión de sabes previos que dificulta entender conceptos nuevos, así como también la nueva forma de iniciar a escribir Código desde el primer día de clase sin las bases necesarias para planificar y organizar el problema dado. Por su parte los problemas psicológicos productos de la frustración al interpretar el problema del mundo real basado en la abstracción de su solución, es un reto para todo estudiante. Así mismo, la falta visualización del Código escrito según su funcionamiento en casos reales también genera un problema de frustración. Proponen una herramienta tecnológica de aprendizaje llamada (Ambiente Visual Integrado de Estructura de Datos), la cual fue aplicada y utilizada por 41 como con entrenamiento en VIA-ED y 47 como grupo sin entrenamiento VIA-ED. Lograron obtener que el 86.6% de los estudiantes lograron mejoras en el aprendizaje de los conceptos de programación, permitiéndoles comprender los conceptos relacionados y seguir la guía o pistas de la ejecución y visualización del programa generado.

Igualmente, según (Selby, 2015) desarrollaron una taxonomía de pensamiento computacional basada en un modelo que integra , los

procesos cognitivos, la pedagogía de la programación y los niveles del pensamiento computacional. Buscaron establecer la relación entre habilidades computacionales (abstracción, generalización, descomposición, diseño algorítmico y evaluación) y los niveles cognitivos descritos por Bloom (aplicación, análisis, síntesis y evaluación). Concluyeron que la evaluación se asigna al nivel de evaluación de Bloom; el diseño de algoritmos se asigna al nivel de síntesis de Bloom; la abstracción y la descomposición se asignan al nivel de análisis de Bloom; la generalización se asigna al nivel de aplicación de Bloom. Concluyeron que descomposición también se percibe como la habilidad de programación más difícil de dominar. Estos incluyen falta de experiencia, comprensión incompleta del problema a resolver y el orden de enseñanza de la programación.

Mientras que, según (Mehmood et al., 2020) realizaron una revisión sistemática acerca del curricular, los métodos pedagógicos, herramientas y diseño de evaluación de los cursos de programación 1 o introducción a la programación, donde se encontraron 60 artículos relacionados a los criterios anteriormente mencionados. Los hallazgos fueron: (i) Se ha prestado poca atención al diseño de planes curriculares existiendo una brecha entre lo que se enseña y se debe enseñar. Algunos programas optan por enseñar una programación estructurada, otros enseñan programación orientada a objetos. Algunos enseñan a codificar, otros se centran en el diseño algorítmico, (ii) en cuanto a los métodos pedagógicos se centran en la enseñanza como técnicas, herramientas, enfoques y métodos desarrollados para la enseñanza de la programación con el objetivo de

desarrollar actividades para mejorar el curso de programación, la percepción del del aprendizaje y la autoeficacia de la programación. pedagógica. Y el aprendizaje según los estilos de aprendizaje tales como VARK, índice de estilos de aprendizaje propuesto por Solomar-Felder, estilo de aprendizaje descrito por George (GSD), (iii) Se concluye que la tendencia en herramientas se basa en incorporar juegos serios y gamificación para motivar a los estudiantes, agregándole estrategias basada en problemas reales., (iv) Por su parte se evidencia poca atención a diseño de evaluaciones lo que dificulta conocer el estado real del aprendizaje de un estudiante.

Por su parte, según (Chiung-Sui Chang et al., 2020) propone la enseñanza basada en juegos en curso de programación. Siendo la estrategia del juego el aprendizaje basado en problemas. Los resultados mostraron que el estudiante logra positivamente los niveles de comprensión y aplicación según la taxonomía de Bloom. Sin embargo, la investigación no presenta comparativas basada en pre-test y post-test por lo que es difícil tener una conclusión segura del logro de los resultados.

1.3.3. Tendencias del proceso de autoaprendizaje de la programación y su dinámica a través del tiempo

En el proceso de autoaprendizaje de la programación, ha habido varias tendencias históricas, las cuales se dinamizaron 3 compontes fundamentales: plan curricular, gamificación como estrategia de aprendizaje, los lenguajes de programación y su codificación:

Etapa del inicio de las Computadoras (1946-1957)

En esta etapa se dio el inicio a las primeras computadoras con fines científicos y militares, las cuales usaban tubos al vació y ocupaban entre 20 - 25 metros cuadrados para ser instaladas. Assembly surgen como uno de los primeros lenguajes de programación. Assembly - lenguaje Maquina (principios 50) se caracterizó por ser lento, poco entendible dado el uso de uso de memorización de abreviaturas nemotécnicas. No se evidencia la existencia de un plan curricular la enseñanza aprendizaje de los contenidos de programación. En esta etapa tanto como las computadoras y la enseñanza aprendizaje de programación son nulas.

Etapa de la formación y mejoras de las computadoras (1958-1964)

Para mejorar el rendimiento y uso de la computadora se usaron transistores que permitieron disminuir el tamaño de las computadoras. Se crearon diversos lenguajes de programación con el objetivo de ser usados para crear nuevos programas con fines comerciales. En 1958, surge FORTRAN y es añadido a IBM 704. FOTRAN del abreviado **FOR**mula **TRAN**slator fue creado con un manual de instrucción. FORTRAN y su manual son una guía base para iniciar la enseña de la programación en Universidades, sin embargo, carecía de ejemplos prácticos, por lo que la enseñanza se volvió autodidactica para cada estudiante. En 1959 se crea COBOL (Common Business Oriented Lenguage) a cargo del departamento de defensa de Estados Unidos, con el objetivo de mejorar la flexibilidad en la lectura - escritura y portabilidad de los programas en diferentes ordenadores. COBOL se convierte en el primer lenguaje de programación orientado a funciones administrativas. En 1964 Jon Kemeny y Thomas Kurstz,

abordaron el problema de la enseña a través de un lenguaje de programación, es así como surge BASIC un lenguaje de programación sencillo de aprender, con menor uso de memoria, pero no era el mejor lenguaje de programación en cuanto a seguridad. Dadas las características de sencillez y mejor uso de memoria, BASIC fue adaptado por Microsoft para sus primeras computadoras personales. El docente juega cumple el rol de instruccionista del aprendizaje. No existe un plan curricular para la enseñanza de la programación, no existen estilos de aprendizaje, tampoco estrategias de aprendizaje.

Etapa de nuevos conceptos y la innovación en las computadoras (1965-1970)

Con el objetivo de hacer masivo el uso de la computadora, abaratando costos y mejorando el rendimiento, se usaron circuitos integrados. En 1966, en las instalaciones del MIT, fue creado LOGO lenguaje de programación a cargo de Seymour Papert e influenciado por Jean Piaget, LOGO se convertiría en el primer lenguaje de programación orientado para niños de menor edad. Con la divulgación de su libro Mindstorm: Children Computer and Powerful ideas. LOGO tendría una dificultad al momento de ser enseñado en los colegios, debido a que no existe parte gráfica y sus componentes basados en subrutinas evitaban ser comprobados de manera grafica. Surgen los primeros libros de computación y como consecuencia se tomaron los conceptos vertidos en el libro como Unidades pedagógicas para la enseñanza. Inicia la carrera de Ingeniería de Sistemasen 1968. En 1970, en búsqueda de un lenguaje de programación entendible y estructurado, nace PASCAL que busca un

mejor uso de la memoria a partir del manejo de punteros, concepto que se genera como un reto en la enseñanza aprendizaje por la forma abstracta de manejar la memoria. Se considera al docente como el principal gestor de la enseñanza aprendizaje, y se considera al estudiante como un ente receptor de conocimiento.

Etapa del desarrollo de habilidades y la innovación en las computadoras (1971 - Actualidad)

En esta etapa se buscó disminuir el tamaño de las computadoras y seguir con el alto rendimiento computacional, surge el uso de microprocesadores y sus núcleos como varias unidades mínimas de procesamiento dentro de una computadora para similares tareas en paralelo. En 1972 surge C, como una alternativa a la portabilidad, C se convertía en el lenguaje de programación más usado por diversos programadores, educadores y fue la base para otros lenguajes de programación. En el 2001 se consolida una primera guía curricular a partir del esfuerzo de ACM - IEEE, que posteriormente da lugar actualizaciones (2004, 2005, 2015, 2021). Todas las actualizaciones tienen como objetivo establecer diferencias entre las diferentes carreras que surgieron sobre computación, así mismo se buscó establecer un orden estandarizado en los conceptos que deben ser enseñados y personalizados por cada carrera a fin a la computación. El docente juega un papel importante en la enseñanza aprendizaje de la programación, dándole poder a los estudiantes de construir su propio conocimiento. Iniciativas como La hora del Código (Code.org, 2021) y Scratch (MIT, 2021) tiene como objetivo enseñar a programar a niños usando estrategias de gamificación. Sin embargo, no se evidencias

herramientas estandarizadas en la enseñanza aprendizaje de los conceptos de programación a nivel de universidad.

Etapa de la mejora continua (Actualidad- tendencia)

La próxima etapa de las computadoras está muy cerca y se trata de las computadoras cuánticas, que busca un alto rendimiento para procesar grandes cantidades de datos que permitan obtener resultados eficientes a partir de los modelos de machine learning y Deep learning. Según ACM (Becker & Quille, 2019) las nuevas carreras serán Ciencias de Datos, Seguridad informática.

Los avances descritos hasta la fecha en el desarrollo del proceso autoaprendizaje de la programación, aun son insuficientes en cuanto a la dinamización de sus componentes plan curricular, estudiante, estrategias de aprendizaje que permitan una sistematización y apropiación de los contenidos de programación por parte del estudiante con la finalidad de escribir códigos de software.

1.3.4. Marco Conceptual

Se presentan los conceptos relevantes en esta investigación:

- **Aprendizaje basado en Proyectos:** Estrategia de enseñanza aprendizaje que permite al estudiante analizar, diseñar, desarrollar una solución a un problema real, donde pone en práctica los conocimientos aprendidos durante el curso. Los avances son supervisados como avances del proyecto con la finalidad de que el estudiante relacione los conceptos y su aplicabilidad en el proyecto.
- **Aula invertida:** Estrategia de enseñanza aprendizaje donde el estudiante primero descubre los conceptos e intentan resolver ejercicios o problemas

con los conceptos estudiados, y posteriormente el docente refuerza esos conocimientos y soluciona los problemas, con el objetivo de que el estudiante compare las soluciones realizadas.

- **Autoaprendizaje de la programación:** Proceso mediante el cual un estudiante aprende por sí mismo los conceptos de programación. El autoaprendizaje puede ser logrado con la practica constante apoyado herramientas tecnológicas o sin herramientas tecnológicas. El autoaprendizaje tiene una motivación intrínseca que permiten al estudiante reconocer que si puede lograr sus objetivos y extrínseca que permite al estudiante motivarse de acuerdo con el reconocimiento externo.
- **Codificación:** Etapa de la programación que le permite al estudiante escribir códigos mediante el aprendizaje de la lógica de programación y de sintaxis propias de un lenguaje de programación**.**
- **Comprensión de la programación:** Es el proceso mediante el cual el estudiante interioriza los conceptos de programación y le permite crear sus propios códigos para resolver problemas o desarrollar soluciones innovadoras.
- **Estilos de aprendizaje de la programación:** Se definen como el conjunto combinado de aspectos cognitivos, afectivos, fisiológicos que permiten al sujeto aprender de acuerdo con sus actitudes y comportamientos que evolucionan con el tiempo. El modelo de estilos de aprendizaje de los estudiantes de programación más estudiado es el modelo propuesto por Neil Fleming y Collen Mills, llamado VARK, por sus siglas de acuerdo con los sentidos del ser humano que permite captar el aprendizaje: Visual, Auditivo, Lectura - escritura, Kinestésico. Por su parte Felder-Silverman

propusieron 4 dimensiones que permiten abordar el estilo de aprendizaje de acuerdo con: i) Trato de la información, ii) Percepción de la información, iii) Recepción de la información y, iv) Entendimiento de la información.

- **Estrategias de enseñanza aprendizaje:** Se refiere a las técnicas innovadoras utilizada para transferir conocimiento cuyo objetivo es motivar al estudiante de manera intrínseca y extrínseca. se destacan las estrategias de Aprendizaje basado en proyectos (Robberts & Employability, 2019), gamificación (Manzano-León et al., 2021), aprendizaje activo (F. Moreira et al., 2021), programación por parejas (Sobral, 2021a), aula invertida (Calle, 2021; Campbell et al., 2014).
- **Gamificación:** Es la actividad de combinar aspectos lúdicos, reconocimiento mediante recompensas y contenidos académicos que dan como origen una estrategia de aprendizaje como herramienta de apoyo a la enseñanza.
- **Lenguajes de programación:** Lenguaje utilizado por los programadores para escribir códigos que posteriormente se transforman en programas o software.
- **Proceso de autoaprendizaje de la programación:** Se consideran al conjunto de pasos que permiten al individuo seleccionar y construir sus propias herramientas, técnicas, métodos o estrategias que apoyan a la apropiación de sus conocimientos.
- **Programación de computadoras:** La programación es el conjunto de actividades que permiten la creación de software. Es fundamental entender que la programación permite trasladar un problema real a ser resuelto

mediante la computadora, dicha solución conlleva a analizar el problema, diseñar la solución, codificar la solución, probar la solución.

1.4. Formulación del Problema.

Insuficiencias en la apropiación de los contenidos de programación, limitan la codificación en el desarrollo y mantenimiento de software

Justificación e importancia del estudio.

Proceso de autoaprendizaje de la programación en ingenieros de software, mediante el empleo de estrategias de gamificación contextualizado debe ser diseñado y direccionarse de acuerdo con la situación actual, por lo tanto, se debe mantener y/o elevar el nivel de los contenidos en las universidades. Desde el plan curricular y específicamente en el curso de programación I se deben establecer las estrategias que permitan al estudiante organizar estructuras cognitivas con el objetivo de apropiarse de nuevos contenidos de la programación en correspondencia al desarrollo de un razonamiento lógico de programación.

Desde la perspectiva social, permitirá contribuir a la comunidad científica con una estrategia de autoaprendizaje que garantice que los estudiantes, futuros ingenieros de sistemas están aptos para proponer soluciones informativas a problemas reales.

Desde la perspectiva académica, se construye una base fundamental para futuras investigaciones basadas en nuevas estrategias de autoaprendizaje de la programación, adaptando nuevas perspectivas, herramientas y metodologías en beneficio de la enseñanza aprendizaje de futuros ingenieros de sistemas.

Aporte teórico: Se modela la dinámica del proceso de autoaprendizaje de la programación a partir de la sistematización autoguiada-interactiva de contenidos de programación considerando los fundamentos epistemológicos, didácticos, psicológicos y sociológicos que sustentan su desarrollo teórico.

La **novedad científica** de la investigación se basa en evidenciar la lógica integradora que se establece entre la apropiación de contenidos mediante el autoaprendizaje y sistematización formativa interactiva del contenido de programación como expresión del desarrollo de una lógica de programación

Aporte práctico: se basa en estrategias de gamificación integral contextualizado, cuyo objetivo es desarrollar un proceso de autoaprendizaje de la programación del ingeniero de software, se sustenta en dos dimensiones: La dimensión de autoformación de la programación y la dimensión de la aplicación autoguiada de contendidos gamificados.

1.4.1. Hipótesis

Si se elabora una estrategia de autoaprendizaje de la programación, sustentada en el modelo de gamificación integral contextualizado, que tenga en cuenta la relación entre la lógica de la sistematización formativa interactiva y la lógica del autoaprendizaje, entonces se contribuye a la codificación en el desarrollo y mantenimiento de software.

1.4.2. Variables, Operacionalización.

VARIABLE INDEPENDIENTE:

Estrategia de autoaprendizaje de la programación basada en un modelo de gamificación integral contextualizado.

VARIABLE DEPENDIENTE:

Codificación en el desarrollo y manteniendo de software

Objetivos

Objetivos General

Aplicar una estrategia autoaprendizaje de la programación, sustentada en un modelo de gamificación integral contextualizado para la codificación en el desarrollo y mantenimiento de software

Objetivos Específicos

a) Caracterizar el proceso del autoaprendizaje de la programación y su dinámica.
b) Determinar las tendencias históricas del proceso de autoaprendizaje de la programación y su dinámica.
c) Diagnosticar el estado actual de la dinámica del proceso de autoaprendizaje de la programación.
d) Elaborar el modelo teórico del proceso de autoaprendizaje de la programación del ingeniero de software, mediate el empleo de estrategias de gamificación.
e) Elaborar la estrategia didáctica interactiva contextualizada del proceso de autoaprendizaje de la programación del ingeniero de sistemas, mediate el empleo de estrategias de gamificación.
f) Validar y corroborar los resultados de la investigación.

II. MÉTODO

2.1. Tipo y Diseño de Investigación.

La investigación aporta al campo de la educación y se basa en una investigación aplicada, de enfoque mixto (Sampieri & P., 2018): (i) Cualitativa que permite describir el proceso de autoaprendizaje de la programación, sus competencias y las estrategias de aprendizaje mediante entrevistas a los actores del proceso (estudiantes y docentes), (ii) Cuantitativa basada en una investigación de tipo experimental que permite accionar con respecto a la variable independiente, con situaciones controladas y que tiene el objetivo de describir las causas que producen sobre la variable dependiente (Guevara et al., 2020). Así mismo se diseña una investigación preexperimental basada en pre-test y post-test.

2.2. Población y muestra.

Se ha considera a la población conformada por 240 estudiantes distribuidos en 8 secciones, y dos docentes a cargo de dos secciones cada uno, pertenecientes al semestre académico 2020-1.

El muestreo se realizó por selección intencional, considerando a toda la población, siendo la muestra representada por el 100% de la población.

Para la pre-experimentación se tomó en cuenta a toda la población estudiantil, 120 estudiantes para el pre-test y a los mismos estudiantes conforman el grupo post-test luego de aplicar la estrategia de autoaprendizaje de la programación

Así mismo se usó una validación adicional que consta de:

El grupo experimental, se basa en 120 estudiantes del curso de programación 1 que aprenden con las estrategias de autoaprendizaje de la programación basado en un modelo de gamificación.

El grupo control 1, se basa en 120 estudiantes del curso de programación 1 que aprenden de forma tradicional.

2.3. Técnicas e instrumentos de recolección de datos, validez y confiabilidad.

Los métodos teóricos, estadísticos y empíricos empleados en la presente investigación son:

- Inducción - deducción: permite determinar los razonamientos especifico y general del objeto de estudio y el campo de acción de la investigación

- Dialectico - Holístic: permite modelar la dinámica del proceso de autoaprendizaje de la programación.

- Análisis - síntesis: permite guiar la investigación mediante el análisis y la síntesis de cada proceso descrito.

- Histórico - lógico: permite establecer las características históricas del proceso autoaprendizaje de la programación.

- Encuestas y entrevista: permite establecer un diagnóstico del objeto de estudio y su campo de acción y posteriormente permite comprobar la estrategia aplicada.

- Técnicas estadísticas: permite establecer las técnicas estadísticas utilizadas.

2.4. Procedimientos de análisis de datos.

Esta investigación, se basó en encuestas y entrevistas para la recolección de datos, que permitieron medir cuantitativa y cualitativamente el estado actual del objeto y campo de la investigación.

Encuestas: aplicada a los estudiantes del curso de programación para obtener información del nivel actual de programación que poseen en el curso de programación de la Universidad Señor de Sipan. Del semestre 2020-1.

Análisis documental: Información recolectada de artículos, revistas, tesis doctorales, etc durante el proceso de investigación.

2.5. Criterios éticos

En la presente investigación fueron utilizados artículos, revistas, tesis doctorales citados de acuerdo con la norma APA:

Se ha procurado mantener los estándares de calidad en correspondencia a las exigencias de la Universidad.

Se mantiene la información objetiva en correspondencia a las exigencias de la Universidad.

Se establecen criterios de privacidad para los encuestados y entrevistados.

2.6. Criterios de Rigor científico.

Todos los datos plasmados en esta investigación son reales, sin edición o falsedad, producto de las entrevistas y encuestas a los actores involucrados.

Aplicabilidad; Se establece el análisis e interpretación de la estrategia de autoaprendizaje de la programación basada en técnicas de gamificación (V. dependiente) y la apropiación de contenidos de programación (V.

Independiente) y en el futuro ser tomados como bases para otras investigaciones.

III. RESULTADOS

3.1. Resultados en Tablas y Figuras

Para comprobar la problemática de la investigación se procedió a realizar encuestas a 120 estudiantes de la carrera de Ingeniería de Sistemas. La encuesta para los estudiantes fue elaborada y validada por expertos y conto con 27 preguntas distribuidas en 2 dimensiones y 5 indicadores. Se presentan los resultados previos a la aplicación de la estrategia propuesta:

Se ha considerado a 120 estudiantes los cuales respondieron la encuesta que constó de 27 preguntas distribuidas en dimensiones e indicadores:

a. **Dimensión: Lógica contextual gamificada del autoaprendizaje en programación.**

i. **Reconocimiento contextual formativo del autoaprendizaje de la programación**

La Fig. 1 muestra los resultados a partir de la encuesta a los estudiantes donde se evidencia que el 67% y el 25% consideran que nunca y casi nunca se toma en cuenta el reconocimiento contextual formativo del autoaprendizaje de la programación, sin embargo, el 8% indica que a veces se ha contextualizado la formación del autoaprendizaje de la programación que implica un diagnóstico previo de cada estudiante, conocer el objetivo y competencias del curso, identificar las fortalezas y debilidades del grupo de estudiantes.

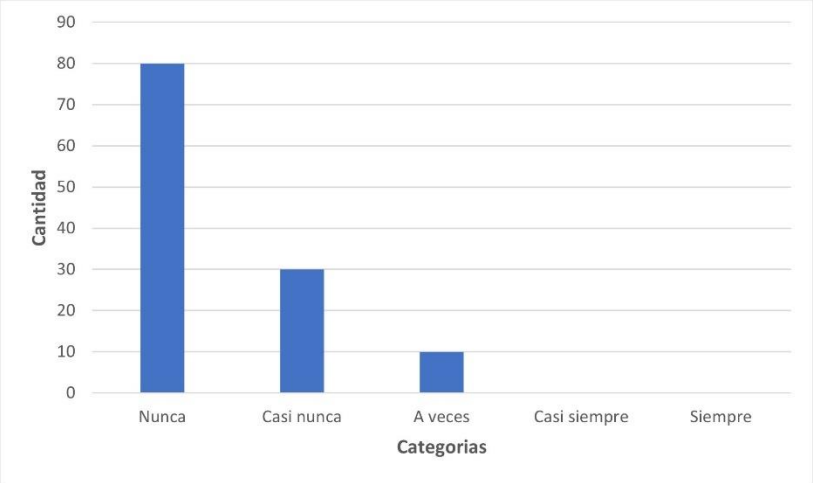


Fig. 1 Reconocimiento contextual formativo del autoaprendizaje de la programación

ii. **Interpretación teórica-personalizada de la programación**

La

Fig. *2* muestra los resultados obtenidos para el segundo indicador perteneciente a la dimensión de la lógica contextual gamificada del autoaprendizaje en programación, donde los estudiantes manifiestan que el 63% nunca y el 29% casi nunca se toma en cuenta la interpretación teórica-personalizada de la

programación es decir las actividades teóricas y prácticas no tiene relación con los problemas de la vida real, así mismo no existen actividades motivacionales intrínseca y extrínsecas que permitan relacionar la teoría con la práctica, no se generan mecanismos de retroalimentación en la clase. Son pocas las actividades grupales que permitan interpretar de forma compartida los resultados.

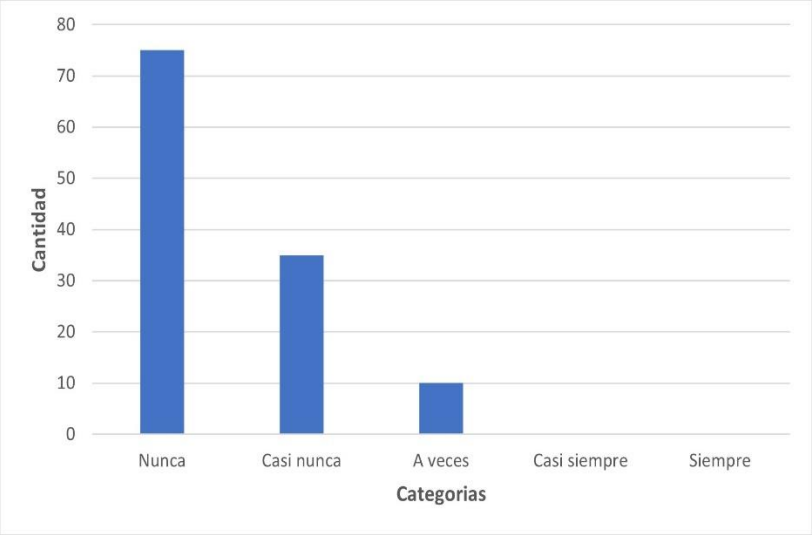


Fig. 2 Interpretación teórica-personalizada de la programación

iii. Sistematización autoguiada-interactiva de contenidos de programación

La Fig. 3 muestra como resultado que el 63% nunca se ha realizado una sistematización autoguiada interactiva de los contenidos de programación, el 25% manifestó que casi nunca y el 13% que a veces se toma en cuenta dicha sistematización. Los estudiantes declaran que no se usa un software o aplicativo que los motive a seguir desarrollando o practicando ejercicios de programación con los temas tratados en clase. Así mismo, manifiestan que son pocas las actividades propuestas de naturaleza cooperativa con materiales atractivos para motivarlos a seguir practicando y aprendiendo los conceptos de programación.

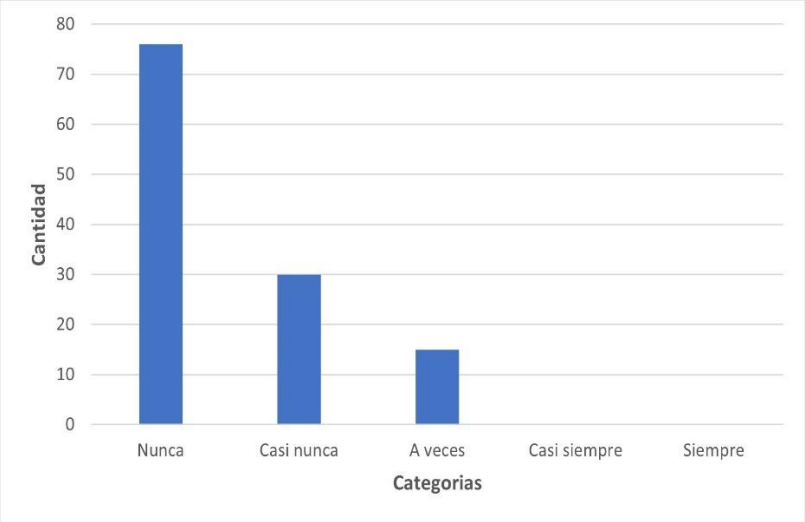


Fig. 3 Sistematización autoguiada-interactiva de contenidos de programación

b. Sistematización autoguiada - interactiva de contenidos gamificados de la programación

i. Diseño de actividades de gamificación en la programación

En la Fig. 4 se aprecian los resultados de la dimensión basada en el diseño de actividades de gamificación en la programación. Se puede ver que el 75% de los estudiantes manifiestan que no existen actividades de gamificación diseñadas para la apropiación o reforzamiento de los contenidos de programación. El 25% de estudiantes manifiesta que casi nunca se hacen actividades de gamificación para aprender los conceptos o practicas a partir de actividades lúdicas.

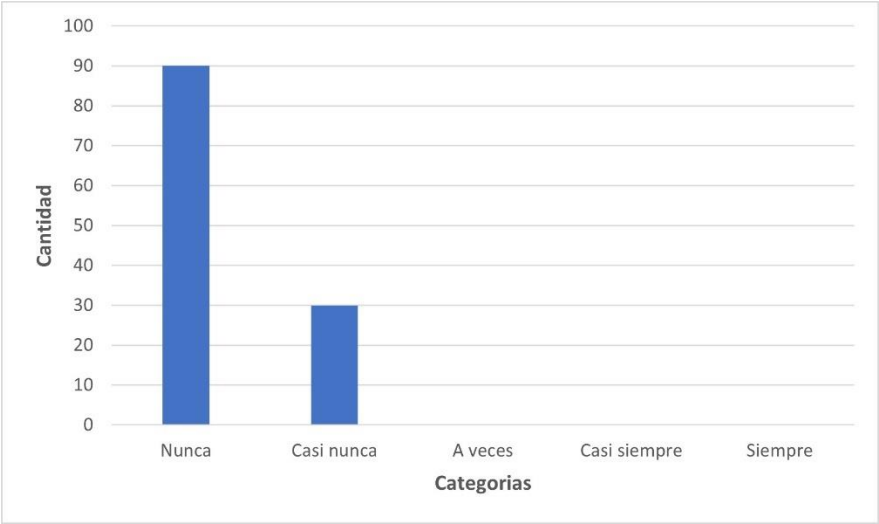


Fig. 4 Diseño de actividades de gamificación en la programación

ii. Gestión informatizada del conocimiento de la programación

En la

Fig. 5 muestra que el 83% de estudiantes considera que nunca se han realizado actividades informatizadas que refuercen sus conocimientos de la programación. Así mismo el 17% indica que casi nunca o pocas veces se realizan actividades informatizadas para reforzar la apropiación de conocimientos de programación. Es decir, los estudiantes consideran que no alcanzan la confianza necesaria para resolver los problemas computacionales propuestos en clase.

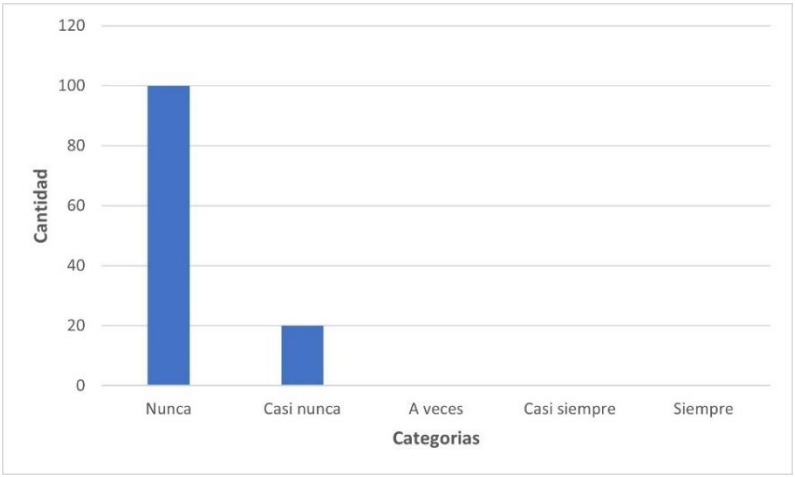


Fig. 5 Gestión informatizada del conocimiento de la programación

iii. Generalización del contenido de programación para el desarrollo de software.

El indicador de la generalización del contenido de programación para el desarrollo de software no es considerado en la apropiación del aprendizaje de los contenidos de programación, tal como lo manifiesta el 79% de estudiantes que consideran que nunca se generalización los contenidos de la programación y el 21% manifiesta que casi nunca se toma en cuenta dicho indicador. Así como se aprecia en la Fig. 6.

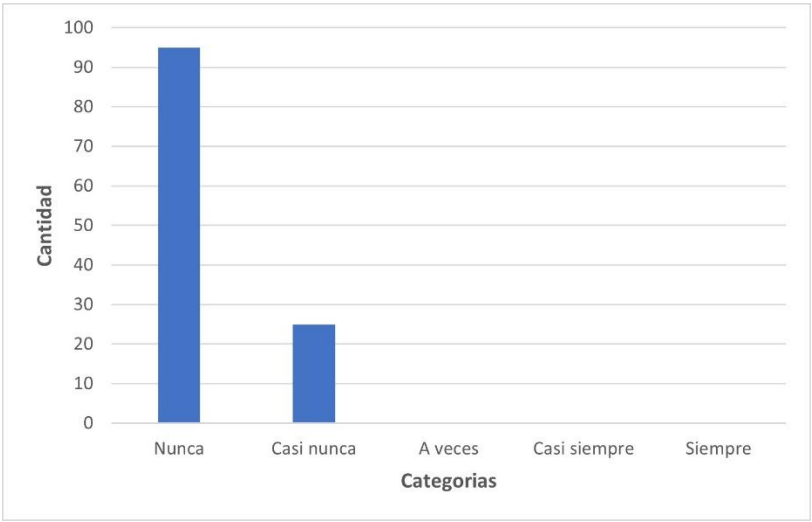


Fig. 6 Generalización del contenido de programación para el desarrollo de software.

3.2. Discusión de resultados

Luego de la ejecución de la encuesta, se puedo concluir en lo siguiente:

Que el 94% de los estudiantes manifiestan que nunca y casi nunca se contextualiza la lógica de gamificación para el autoaprendizaje de la programación y no se aplica contenidos autoguiados y gamificados en la programación, lo que imposibilita la apropiación de los contenidos de programación.

La dimensión lógica contextual gamificada del autoaprendizaje en programación permito conocer que el 64% y el 26% de estudiantes nunca y casi nunca reconocen contextualmente la formación del autoaprendizaje, además manifiestan que no reconocen la interpretación teórica-personalizada, como consecuencia, nunca o casi nunca se sistematiza autoguiada e interactivamente en los contenidos de programación, y finalmente, no se considera una generalización de los contenidos de programación para el desarrollo de software.

La dimensión de la sistematización autoguiada-interactiva de contenidos gamificados de la programación presenta resultados poco alentadores. El 79% de estudiantes manifiestan que nunca se diseñan actividades de gamificación en la programación y tampoco se realiza una gestión informatizada del conocimiento de la programación.

3.3. Aporte teórico

3.3.1. Construcción teórica del proceso de autoaprendizaje de la programación de la programación, sustentada en un modelo de gamificación integral contextualizado.

Se modela la dinámica del autoaprendizaje a partir de la sistematización autoguiada-interactiva de materiales de programación, teniendo en cuenta los fundamentos sociales, psicológicos, epistemológicos y pedagógicos que amparan su diseño teórico. El modelo parte de la contradicción fundamental que se manifiesta entre el reconocimiento contextual formativo del autoaprendizaje de la programación y la interpretación personalizada de la programación. Así mismo se manifiesta de la contracción inicial entre el diseño de actividades de gamificación en la programación y la gestión informatizada del conocimiento de

la programación. Relaciones dialécticas que del dan coherencia e integración al proceso y a la vez dinamizan sus relaciones.

A partir de las interacciones regulares del proceso se logra elaborar una estrategia, que permite descubrir una relación entre la sistematización autoguiada-interactiva de contenidos de programación y la generalización de los contenidos de programación, lo que permitió proyectar dos niveles.

El modelo actual se configura coherentemente para revelar dos dimensiones fundamentales de acuerdo con la Teoría Holístico Configuracional, con el objetivo de construir el sustento teórico para la elaboración de una estrategia de autoaprendizaje de la programación, mediante un modelo de gamificación.

3.3.2. Fundamentación del aporte teórico

La Concepción Científica Holística De las Fuentes de H. (2009), es fundamental para la sustentación de este modelo debido a que permite desarrollar habilidades de lectura, aspecto fundamental para establecer relaciones dialécticas desde las visiones metodológicas y epistemológicas. En ese sentido, el modelo Holística-Configuracional (Molina & Mejia, 2013) brinda un enfoque que permite enlazar dimensiones propias del proceso de autoaprendizaje (diseñar, construir, producir teorías, conceptualizar y evaluar) resultantes de las acciones o actividades vivenciales y sociales, como resultado se obtiene un proceso cíclico que permite descubrir problemáticas recurrentes y propuestas de acción para mejorar el proceso de autoaprendizaje.

3.3.3. Descripción argumentativa del aporte teórico

Los vínculos entre configuraciones y dimensiones se revelan en el modelo actual como una expresión de la totalidad del proceso modelado. Se han revelado las siguientes dimensiones:

- Dimensión lógica contextual gamificada del autoaprendizaje de la programación

- Dimensión sistematización autoguiada - interactiva de contenidos gamificados de la programación.

El carácter básico de las interacciones que surgen dentro de cada una de ellas a partir de sus configuraciones caracteriza a este modelo, que tiene como centro la sistematización autoguiada-interactiva de los contenidos de programación, configuración que rige la lógica dinámica del proceso.

La dimensión Lógica contextual gamificada del autoaprendizaje de la programación se constituye a partir de las configuraciones: i) Autoaprendizaje gamificado para la comprensión de la programación, ii) Reconocimiento contextual formativo del autoaprendizaje de la programación, iii) Interpretación teórica-personalizada de la programación, y iv) Sistematización autoguiada-interactiva de contenidos de programación, como se aprecia en la

Fig. 7.

Fig. 7 Lógica contextual gamificada del autoaprendizaje en programación

Autoaprendizaje gamificado para la comprensión de la programación Se basa en los factores motivacionales, cognitivos y la didáctica gamificadora, como elementos que favorecen las condiciones para el autoaprendizaje de la programación:

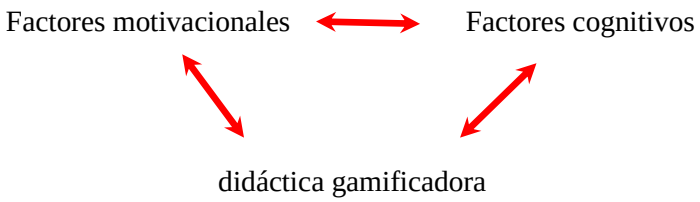


Fig. 8 Autoaprendizaje gamificado para la comprensión de la programación

El arduo trabajo que tienen los docentes de la carrera de Ingeniería de Sistemas para comprometer a los estudiantes en el autoaprendizaje de la programación como materia del primer año de su formación profesional involucra el esfuerzo de mantenerlos motivados no solo en cada sesión de clase sino también fuera de ellas.

Como objeto de estudio, el “Proceso de autoaprendizaje de la programación”, no se puede imponer académicamente, especialmente porque esta capacidad deductiva

requiere un alto nivel de abstracción por parte de los alumnos para que puedan comprender. Como resultado, se deben introducir nuevas herramientas informáticas que incluyan características didácticas gamificantes y que tomen en cuenta actividades educativas bien diseñadas que permitan generalizar el contenido de la programación.

Se debe alentar a los estudiantes a ser creativos, y se debe poner a su alcance una serie de recursos que les permita comprender de manera más amplia la aplicabilidad de los conceptos de programación que se les enseñan en su formación universitaria y que sirva para reafirmar su motivación profesional. Lo anterior, requiere la creación y organización de todas las asignaturas que incluyan aspectos informáticos que sirvan de motivadores gamificando factores didácticos.

Es fundamental enseñarles cómo usar las técnicas que han aprendido en un entorno del mundo real. La programación está presente en todas las acciones humanas, mientras que parece que algunos intentan separarla de la realidad hasta el punto en que los estudiantes son incapaces de reconocer su presencia y aplicabilidad.

Muchos de los problemas de programación que sufren muchos estudiantes provienen de la falta de deseo de su propia capacidad en esta disciplina, que a menudo es causada por la débil introducción de sus profesores a los contenidos.

Debido a que la motivación es un componente del proceso emocional, la motivación debe presentarse en unidad dialéctica. Sin embargo, es necesario examinar el escenario de antemano para que el estudiante experimente la emoción y viceversa. Es decir, es vital subrayar la importancia de su estado emocional actual para que pueda comprender el escenario del problema computacional. Para que surja una emoción, se deben considerar aspectos cognitivos, pero también se debe considerar la influencia reguladora del estado afectivo. El desarrollo de esta relación fomenta

la motivación, así como la apertura y la voluntad de aprender. En programación no es suficiente la motivación para que se genere la apropiación de conocimiento y de la lógica computacional, además, se requiere identificar las necesidades de qué aprender y cómo aprenderlo lo cual permitirá personalizar los conocimientos de programación en los estudiantes.

El **reconocimiento contextual formativo del autoaprendizaje** de la programación se fundamenta en las actividades cognitivas basada en las formas lógicas del pensamiento, las condiciones necesarias de aprendizaje y los procesos lógicos que permiten el autoaprendizaje de la programación, como se aprecia en la Fig. 9.

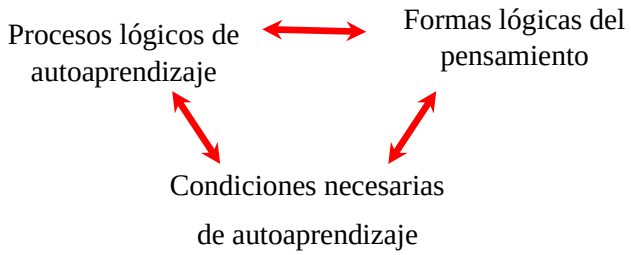


Fig. 9 Reconocimiento contextual formativo del autoaprendizaje de la programación

Los procesos lógicos de autoaprendizaje se basan en las condiciones necesarias de autoaprendizaje, que a su vez contienen acciones a realizar, tales como: reconocer, comparar, criticar, relacionar conceptos de programación con el objetivo de realizar formas lógicas de pensamiento que permiten criticar la veracidad o falsedad del concepto adquirido.

Estos métodos, lógicos o específicos dentro de la asignatura de programación, deben considerarse como un "objeto de instrucción", y se pueden enseñar durante el proceso de autoaprendizaje.

Hoy en día se busca que los estudiantes piensen de manera integral, es decir reconociendo conceptos y los juicios que lo conforman, lo cual permite crear un razonamiento lógico de los conceptos adquiridos.

El autoaprendizaje gamificado para la compresión de la programación y el reconocimiento contextual formativo del autoaprendizaje de la programación están contrapuestos dialécticamente porque el primero de ellas permite el desarrollo del proceso emocional, mientras que la segunda determina la conformación de estructuras cognitivas. A pesar de esto, también se dan en unidad dialéctica, porque ambas constituyen sistemas esenciales para la apropiación de una cultura de programación. El autoaprendizaje gamificado para la compresión de la programación, como síntesis de lo cognitivo y lo afectivo está condicionado por el desarrollo de procesos y procedimientos lógicos relacionados con la orientación lógica de la programación, pero esta a su vez se logra si los estudiantes están motivados por los nuevos contenidos.

Con una fuerte tendencia hacia la deshumanización de la ciencia y la lógica en el entorno actual, es primordial la eficacia entre la relación dialéctica del autoaprendizaje gamificado para la compresión de la programación y el reconocimiento contextual formativo del autoaprendizaje de la programación.

La **interpretación teórica-personalizada de la programación** es la capacidad de comprender conceptos abstractos y sus interacciones con los estilos de aprendizaje y objetivos educativos de cada estudiante se entiende como la dinámica del proceso de autoaprendizaje de la programación, como se muestra en la Fig. 10.

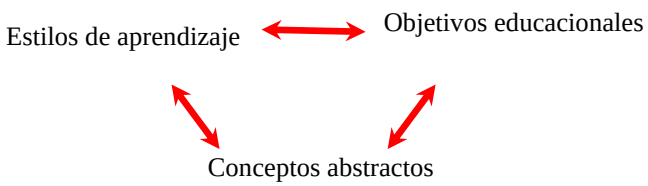
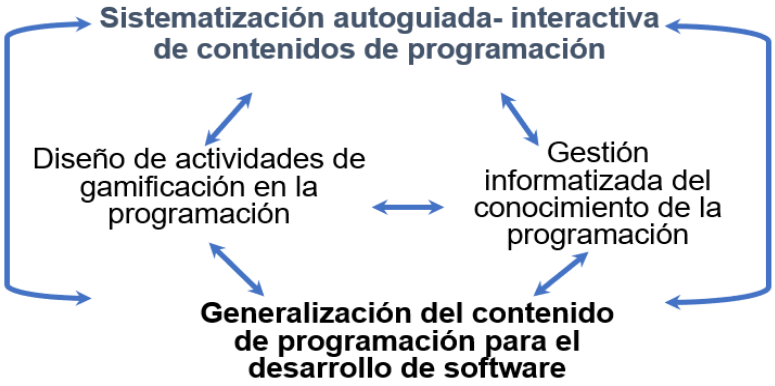


Fig. 10 Interpretación teórica-personalizada de la programación

Ahora bien, siendo la **sistematización autoguiada- interactiva de contenidos de programación** el objetivo del modelo propuesto, que se fundamenta de los procesos de reconocimiento contextual formativo del autoaprendizaje de la programación e interpretación teórica-personalizada de la programación, es la interacción de estas configuraciones, requeridas para lograr el proceso de autoaprendizaje de la programación basada un modelo gamificado para la compresión de la programación, relación holística que se constituyen de la dimensión **Lógica contextual gamificada del autoaprendizaje en programación.**

La **generalización del contenido de programación para el desarrollo de software** es la configuración resultante de la relación entre el **diseño de actividades de gamificación en la programación** y la **gestión informatizada del conocimiento de la programación**, que se constituyen en el fin de la **sistematización autoguiada – interactiva de contenidos de programación**, cuya sinergia conforman la **dimensión de la sistematización autoguiada-interactiva de contenidos gamificados**. Como se muestra en



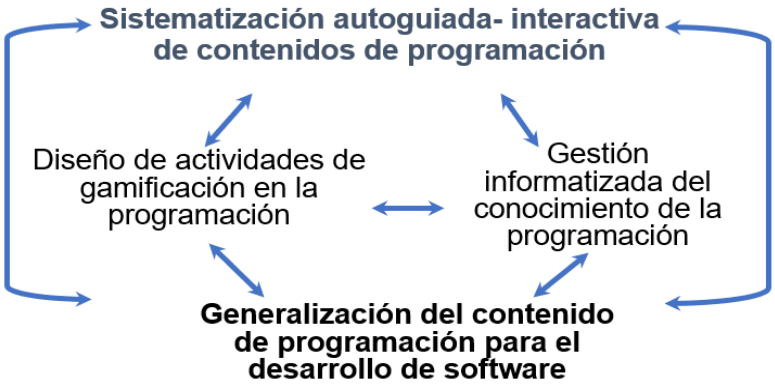


Fig. 11

Fig. 11 Aplicación autoguiada de contenidos gamificados de la programación

El diseño de actividades de gamificación en la programación se entiende como el conjunto de conocimiento, destrezas y valores que componen el sistema de contenidos de programación, adicionando elementos lúdicos e interactivos, necesarios para adquirir nuevos conocimientos a través de la gestión informatizada del conocimiento de la programación, evidenciando una lógica de razonamiento lógico que generan el enriquecimiento de las estructuras cognitivas.

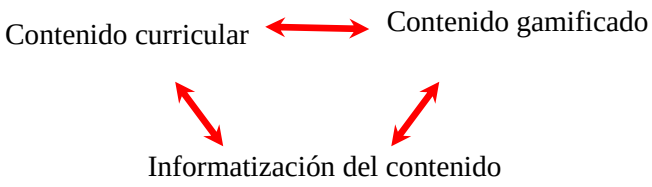


Fig. 12 Diseño de actividades de gamificación en la programación

Se basa en establecer una relación entre el contenido actual y el contenido gamificado que permita informatizar dicho contenido.

diseño de actividades de gamificación en la programación se desarrolla en contradicción dialéctica con la gestión informatizada del conocimiento de la programación, dado que esta es posible solamente si existe dominio en el diseño de actividades de gamificación y se genera la relación entre ambos, es decir la

existencia de la gestión informatizada depende de las actividades de gamificación y viceversa.

La **Gestión informatizada del conocimiento de la programación** es la interacción dinámica y sintetizada que existe entre las **actividades de gamificación de la programación** y la **Sistematización autoguiada- interactiva de contenidos de programación,** esta configuración se define como la síntesis de nuevos conocimientos, habilidades y valores adquiridos a través de una interacción sistematizada y autoguiada de actividades gamificadas que resulta en el enriquecimiento de las estructuras cognitivas.

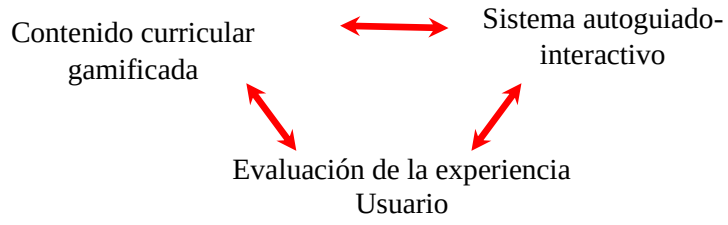


Figura 13: Gestión informatizada del conocimiento de la programación.

La relació autoguiado-interactivo permite ser evaluado por los estudiantes por medio de su experiencia en el uso, manejo y sobre todo en el conocimiento adquirido, dando muestras de motivación, aprendizaje y valoración de la herramienta informática integral, teniendo en cuenta: i) La experiencia previa del estudiante para comprender el nuevo contenido. ii) Las habilidades e interpretaciones basado en la experiencia determinan la forma de ejecutar las actividades de cada estudiante.

La **generalización del contenido de programación para el desarrollo de software** debe entenderse como la secuencia lógica y sistémica que posibilita que el estudiante valore el conocimiento de programación informatizado que se encuentra basado en el diseño de actividades de gamificación de la programación.

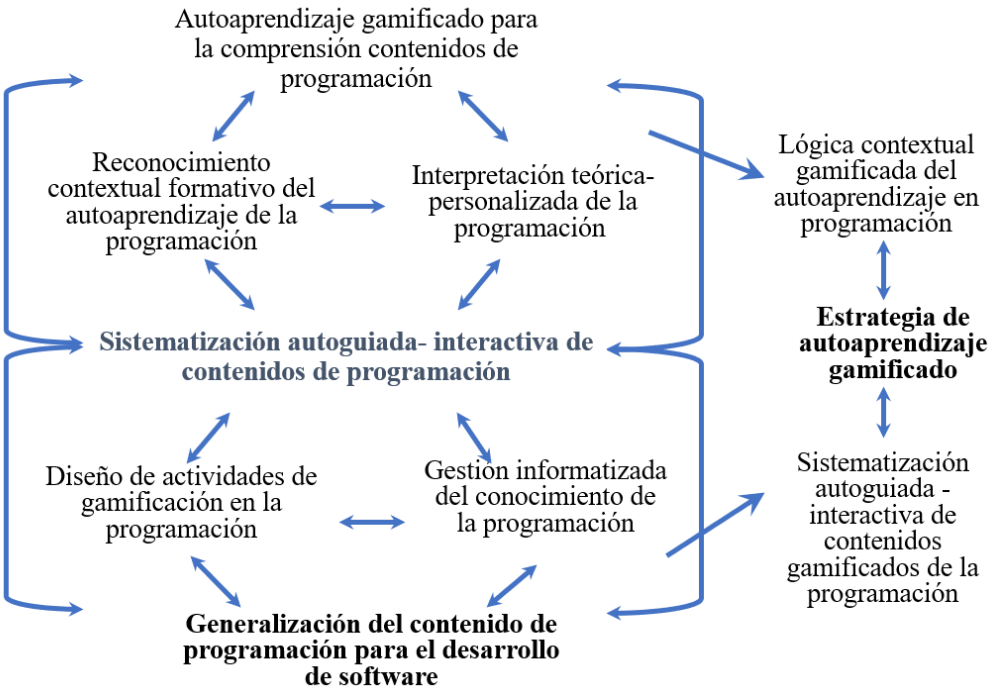


Fig. 13 Modelo de Autoaprendizaje de la programación

3.4. Aporte práctico

En esta sección se considera los aspectos que fundamentan el proceso de autoaprendizaje de la programación como aporte práctico de la investigación. El proceso de autoaprendizaje de la programación del ingeniero de software se sustenta en el modelo lógico contextualizado gamificado basado en estrategias de gamificación que revela la dinámica "autoguiada e interactiva del proceso de autoaprendizaje" en el área de la programación a partir de la sistematización lógica de los contenidos. EL modelo parte de las contradicciones manifestadas al comparar o interpretar la lógica de los contenidos precedentes y la comprensión del nuevo conocimiento mediante el uso autoguiado de contenidos gamificados, relación dialéctica que dinamiza en todo momento el modelo y a la vez le da coherencia e integración.

El proceso de autoaprendizaje tiene como objetivo revelar una relación entre la apropiación de la **sistematización lógica del contenido autoguiado y gamificado**

de la programación y la generalización de contenidos de programación, lo que determina la proyección de dos niveles que se dan constantemente en contraposición y en unidad dialéctica.

3.4.1. Fundamentación del aporte práctico

La estrategia de autoaprendizaje de la programación basada en un modelo de gamificación contextualizado se sustenta en dos dimensiones: La dimensión de la lógica contextual gamificada del autoaprendizaje de la programación y la dimensión de la aplicación autoguiada de contendidos.

Las fases de la primera etapa de este proceso derivan de las configuraciones relacionadas al reconocimiento contextual formativo del autoaprendizaje de la programación, la interpretación teórica-personalizada de la programación, la sistematización autoguiada-interactiva de contenidos de programación y el autoaprendizaje gamificada para la comprensión de la programación, actividades que permiten apropiarse de las fundamentaciones mínimas exigible.

Las fases de la segunda etapa derivan de las configuraciones relacionadas al diseño de actividades de gamificación en la programación, la gestión informatizada del conocimiento de la programación que conlleva a la generalización de conceptos y contenidos de programación, necesarios para el desarrollo de software y la sistematización autoguiada-interactiva de contenidos de programación, etapa que permite unir las dos dimensiones.

De las configuraciones del modelo lógico contextualizado formativo basado en estrategias de gamificación con un enfoque holístico configuracional surge la fundamentación y explicación del proceso de autoaprendizaje de la programación en ingenieros de sistemas, considerando la relación entre la identificación de

necesidades de aprendizaje de la programación, como contenidos precedentes, y la comprensión de nuevos contenidos. La necesidad de la generalización del contenido de la programación necesarios para el desarrollo de software y su gestión informatizada. La relación entre la autoformación gamificada para el desarrollo de software y la sistematización autoguiada-interactiva de contenidos de programación como elementos esenciales para la apropiación de la lógica de programación.

En general la estructura sistémica del proceso parte de un diagnóstico en el que se evidencia un problema y sus causas fundamentales, se plantean acciones organizadas en etapas que permiten alcanzar de forma incremental el objetivo propuesto, posteriormente se establecen indicadores de control, evaluación y retroalimentación del proceso desarrollado.

El proceso de autoaprendizaje de la programación del ingeniero de software se fundamenta en la estrategia de gamificación que ubica al estudiante al centro del proceso como un receptor - actuador del conocimiento, en este contexto el docente cumple el rol de facilitador de contenidos que domina el conocimiento de la programación, estrategias lúdicas de gamificación y las tecnologías, pilares fundamentales para la transferencia de conocimiento hacia el estudiante.

3.4.2. Relación entre aporte teórico y práctico

La investigación se fundamenta en el aporte de un modelo de autoaprendizaje de la programación, el cual se debe concretar con el planteamiento de una estrategia de autoaprendizaje de la programación, que implica acciones gamificadas que colaboran con la transformación socio – cognitiva del estudiante de ingeniería de software.

La estrategia de autoaprendizaje tiene como base la sistematización formativa integral en competencias, que se establece en la intencionalidad investigativa de desarrollar la capacidad resolutiva de problemas computacionales basados en estrategias gamificadas en entornos digitales y teniendo como principal actuador al estudiante.

3.4.3. Construcción del aporte práctico

El autoaprendizaje de la programación como estrategia, tiene como clave la sistematización lógica contextualizada gamificada de los nuevos conocimientos, que se expresa como el conjunto de acciones lúdicas que contribuyen al logro de competencias de programación en estudiantes de Ingeniería de Sistemas, basada en estrategias gamificadas como complemento a los conocimientos transferidos por el docente hacia los estudiantes.

Se define al proceso de autoaprendizaje de la programación como un conjunto de acciones donde el estudiante refuerza los conocimientos de forma independiente o en equipo para lograr los objetivos de aprendizaje asignados en una competencia académica. Según (Yulianto et al., 2016) los contenidos en un proceso de autoaprendizaje contiene desafíos como las características del estudiante, las pautas del diseño, los requisitos tecnológicos, la disponibilidad y la calidad de los contenidos.

En tal sentido, el modelo lógico contextualizado gamificado para fundamentar el proceso de autoaprendizaje de la programación se basa en diseñar estrategias didácticas de autoaprendizaje gamificadas que permitan la motivación intrínseca y extrínseca del estudiante. El estudiante está motivado intrínsicamente por su interés en lograr los objetivos y encuentra placer en alcanzar su meta; así mismo

extrínsecamente buscan un reconocimiento, valoración de su esfuerzo en el logro de los objetivos. Para (Banfield & Wilkerson, 2014) la gamificación como un tipo de pedagogía puede lograr ambas motivaciones. La lógica de este modelo en el proceso de investigación se desarrolla a través de los eslabones, mediante los procedimientos del método, como la caracterización epistemológica y praxeológica del proceso (objeto), la formulación del objeto transformado considerando el instrumento y la aplicación del instrumento.

Además, se considera necesario ver al proceso de autoaprendizaje de la programación desde una gestión formativa integral universal contextualizada, que exige una intencionalidad de formación basada en competencias profesionales y debe ser configurada congruentemente para que contribuya armónicamente con la cadena formativa integral del estudiante.

De esta manera la estructura general del proceso de autoaprendizaje de la programación tiene en cuenta:

- Diagnóstico de la problemática
- Premisas
- Requisitos
- Objetivo estratégico
- Precisiones metodológicas para la implementación del proceso
- Etapas, momentos, objetivos específicos y acciones propuestas.
- Evaluación del proceso de autoaprendizaje
- Control y retroalimentación de la ejecución de del proceso de autoaprendizaje

Diagnóstico del estado actual del del proceso de autoaprendizaje de la programación

Es fundamental identificar los elementos actuales relacionados al proceso de autoaprendizaje de la programación existentes en la formación de profesionales en Ingeniería de Software, a partir de las manifestaciones de sus actores directos: docentes y estudiantes. En esta investigación se empleó una encuesta para docentes del área implicada en la investigación y otra encuesta para estudiantes del ciclo I de la carrera de Ingeniería de Sistemas de la Universidad Señor de Sipan, además del análisis documentario, que permitió recopilar información acerca de las competencias de programación que se buscan en el curso, los contenidos y el material utilizado por los docentes. De esta manera el diagnóstico permite evidenciar cada una de las fases contempladas en el proceso de la presente investigación:

Reconocimiento de la lógica contextual gamificada del autoaprendizaje en programación

- No siempre se diagnostica el estilo de aprendizaje del estudiante para apropiación de conceptos de programación
- Parcialmente se identifican los componentes motivacionales intrínsecos y extrínsecos que generan sentimientos positivos en los estudiantes.
- Limitado diagnóstico de los saberes previos de los estudiantes como línea base para la adquisición de nuevos conocimientos
- Carencia de identificación la relación los contenidos del curso de programación con las competencias específicas y generales del area de programación y su relación con los estándares internacionales

Interpretación teórica-personalizada de la programación

- Parcialmente se diagnostican los conceptos importantes y de mayor dificultad en el aprendizaje de la programación
- Carencia de actividades de motivación previa en cada sesión de aprendizaje teórica
- Limitadas sesiones de clase con participación constante del estudiante
- Escasa retroalimentación de los aprendido.
- Limitadas actividades extras que generen motivación extrínseca (recompensa, puntos, reconocimiento)

Sistematización autoguiada-interactiva de contenidos de programación

- Carencia de identificación de los conceptos que deben ser sistematizados para generar contenidos de autoaprendizaje
- Falta de Clasificación los contenidos por estilos de aprendizaje
- Limitados elementos motivacionales extrínsecos tales como (recompensas, puntos, insignias)

Diseño de actividades de gamificación en la programación.

- Carencia de recopilación de información de los estilos de aprendizaje
- Limitada recolección de información de los conceptos relevantes y que generan dificultad para la apropiación de conocimiento de programación
- Carencia de diseño de sesiones de clase usando conceptos de gamificación

Gestión informatizada del conocimiento de la programación.

- Carencia de una aplicación con conceptos de gamificación teniendo en cuenta estilos de aprendizaje y conceptos motivacionales.
- Carencia de generación de reportes de participación y logros de los estudiantes.

Generalización del contenido de programación para el desarrollo de software.

- Carencia de generación de retroalimentación de la aplicación por parte de los estudiantes.
- Carencia de generación de retroalimentación de la aplicación por parte de los docentes.

Premisas

En la estructuración de la estrategia se considera las siguientes premisas:

- Comprensión teórica de la autoformación gamificada para el desarrollo de software
- Diagnóstico contextual gamificado de competencias de programación
- Sistematización autoguiada-interactiva de contenidos de programación.
- Diseño de actividades de gamificación en la programación.
- Gestión informatizada del conocimiento de la programación.
- Generalización del contenido de programación para el desarrollo de software.

Requisitos

Se han considerado los siguientes requisitos en relación con los docentes:

- Capacidad investigativa para proponer actividades innovadoras que permitan al estudiante apropiarse del conocimiento.

- Intencionalidad y compromiso para la búsqueda del desarrollo de la capacidad resolutiva de problemas de programación.
- Identificación con la labor docente en la búsqueda para mejorar los procesos educativos.
- Uso adecuado de las herramientas tecnológicas como soporte para la enseñanza-aprendizaje de la programación
- Apropiación de contenidos de programación de acuerdo con las competencias del curso para generar y proponer actividades lúdicas que permitan al estudiante apropiarse de los conocimientos.
- Preparación continua en el manejo de nuevas herramientas tecnológicas y retroalimentación de las actividades propuestas.
- Valoración del procesos y relaciones con las etapas y sus fases.

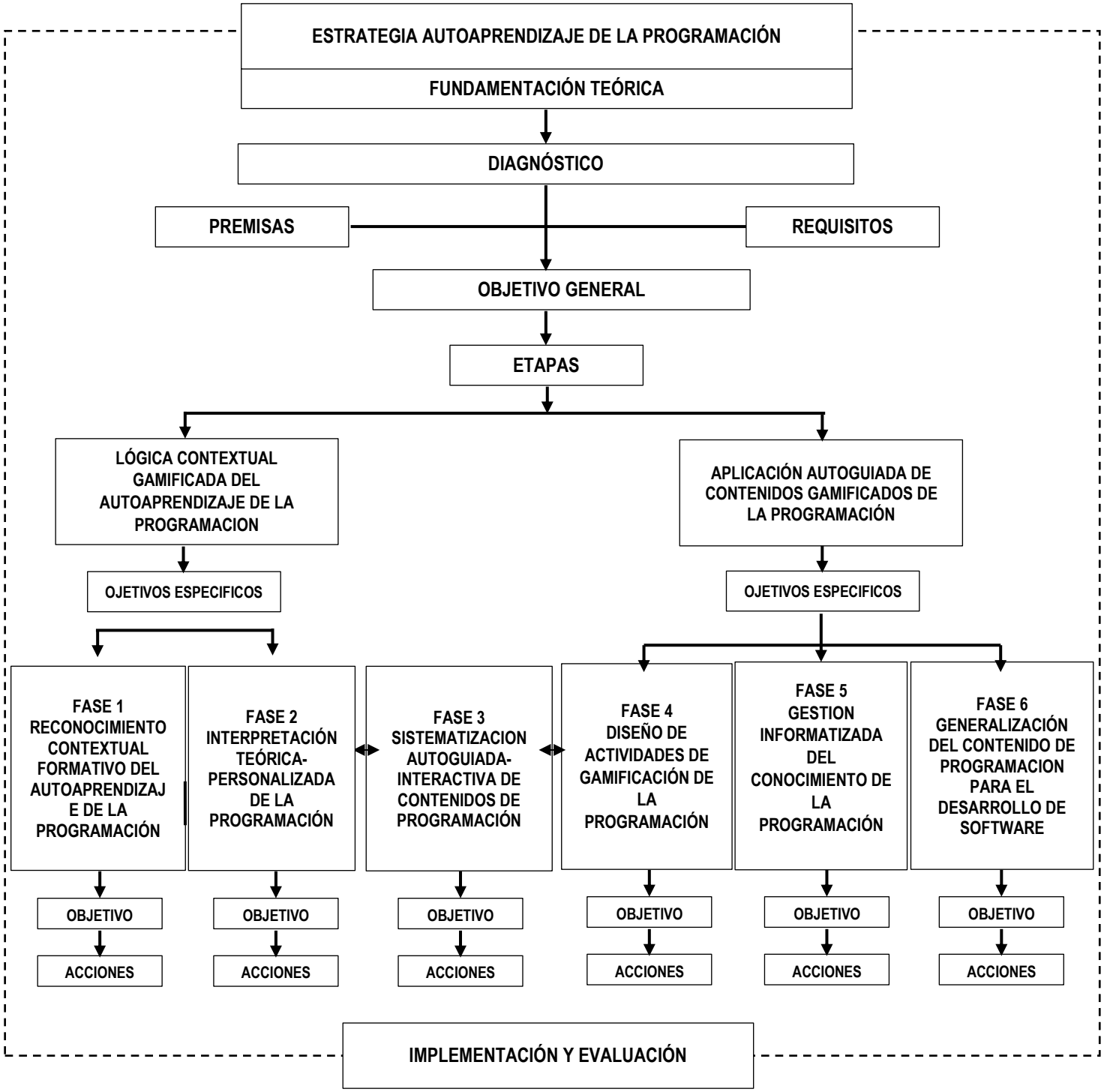
En referencia a los requisitos de la estrategia, en estudiantes:

- Concientización de su rol activo dentro del proceso.
- Motivación para efectivizar su proceso de aprendizaje.
- Responsabilidad para ejecutar las acciones de cada fase del proceso de autoaprendizaje.
- Comunicación de la comprensión de los procesos que va realizando.
- Atención a las orientaciones del docente.
- Capacidad para trabajar colaborativamente.

Objetivo general de la estrategia

Sistematizar el proceso de autoaprendizaje de la programación a través del diagnóstico, la sistematización de los contenidos, la comprensión teórica el diseño de actividades, la gestión del conocimiento y la generalización mediante técnicas de gamificación en los cursos del area de programación de la Universidad Señor de Sipan.

APORTE PRÁCTICO DE LA INVESTIGACIÓN



5.2. Planeación Estratégica

Para la elaboración de la estrategia se consideró distribuir los objetivos y fases en dos etapas que permitan lograr el objetivo general:

- Etapa lógica contextual gamificada del autoaprendizaje en programación
- Etapa de la sistematización autoguiada - interactiva de contenidos gamificados de la programación

Etapa 1: Lógica contextual gamificada de la autoformación en programación

Objetivo Especifico: Contextualizar el proceso del autoaprendizaje de la programación, considerando el reconocimiento contextual formativo del autoaprendizaje de la programación, la interpretación teórica-personalizada de la programación, la sistematización autoguiada-interactiva de contenidos de programación.

Etapa	Fase	Objetivo	Acciones	Recursos
Lógica contextual gamificada del autoaprendizaje de la programación	Reconocimiento de la lógica contextual gamificada del autoaprendizaje en programación	Identificar la situación actual de los aspectos cognitivos relacionados a la enseñanza aprendizaje de la programación.	Diagnosticar el estilo de aprendizaje del estudiante para apropiación de conceptos de programación	Aplicación de encuesta
			Identificar los componentes motivacionales intrínsecos e extrínsecos que generan sentimientos positivos en los estudiantes.	Aplicación de encuesta
			Diagnosticar los saberes previos de los estudiantes como línea base para la adquisición de nuevos conocimientos	Aplicación de encuesta
			Identificar la relación los contenidos del curso de programación con las competencias específicas y generales del área de programación y su relación con los estándares internacionales	Revisión documentaria
	Interpretación teórica-personalizada de la programación	Interpretar los componentes teóricos personalizables de la programación según el estilo de aprendizaje del estudiante y sus motivaciones	Diagnosticar los conceptos importantes y de mayor dificultad en el aprendizaje de la programación	Revisión documentaria
			Generar actividades de motivación previa en cada sesión de aprendizaje teórica	Presentaciones con componentes visuales
			Desarrollar sesiones de clase con participación constante del estudiante	Presentaciones con componentes visuales
			Generar retroalimentación de los aprendido.	Presentaciones con componentes visuales
			Proponer actividades extras que generen motivación extrínseca (recompensa, puntos, reconocimiento)	Uso de aplicaciones: kahoot, trello, socrates, scratch, tinkercard.

	Sistematización autoguiada-interactiva de contenidos de programación	Sistematizar conceptos de programación para el autoaprendizaje de sus contenidos	Identificar los conceptos que deben ser sistematizados para generar contenidos de autoaprendizaje	Revisión documentaria
			Clasificar los contenidos por estilos de aprendizaje	Revisión documentaria
			Identificar elementos motivacionales extrínsecos tales como (recompensas, puntos, insignias)	Revisión documentaria

Etapa 2: Aplicación autoguiada de contenidos gamificados de la programación

Objetivo Especifico: Proponer la sistematización de conceptos de programación mediante el diseño de actividades gamificadas y la gestión informatizada del conocimiento de programación que conlleva a la generalización y apropiación de conceptos y contenidos de la programación.

Etapa	Fase	Objetivo	Acciones	Recursos
Sistematización autoguiada - interactiva de contenidos gamificados de la programación	Diseño de actividades de gamificación en la programación	Diseñar la sistematización de actividades gamificadas de la programación.	Recopilar la información de los estilos de aprendizaje	Revisión documentaria
			Recopilar información de los conceptos relevantes y que generan dificultad para la apropiación de conocimiento de programación	Revisión documentaria
			Diseñar sesiones de clase usando conceptos de gamificación	Técnicas de UX, mockups
	Gestión informatizada del conocimiento de la programación	Gestionar aplicaciones tecnológicas el conocimiento de la programación y los conceptos de gamificación	Desarrollar una aplicación con conceptos de gamificación teniendo en cuenta estilos de aprendizaje y conceptos motivacionales	IDE de desarrollo de aplicaciones web con componentes de realidad aumentada
			Generar reportes de participación y logros de los estudiantes	Reportes

	Generalización del contenido de programación para el desarrollo de software.	Generalizar los contenidos de la programación y sus resultados para una mejora continua	Generar la retroalimentación de la aplicación por parte de los estudiantes	Encuesta
			Generar la retroalimentación de la aplicación por parte de los docentes	Juicio de expertos

5.2.1 Instrumentación

El proceso se aplica durante las sesiones de aprendizaje en el curso de programación 1 de la carrera de Ingeniería de Sistemas de la USS. En 4 secciones de 30 estudiantes, distribuidas en 1 grupo experimental y 1 grupo control. Se involucra en el proceso a 4 docentes del curso de programación 1. Se aplica una prueba diagnóstica de conocimientos previos y se compara los resultados luego de aplicar el proceso de autoaprendizaje de la programación.

El grupo control 1, se basa en estudiantes del curso de programación 1 que aprenden con las estrategias de autoaprendizaje de la programación basado en un modelo de gamificación.

El grupo experimental 1, se basa en estudiantes del curso de programación 1 que aprenden de forma tradicional.

5.2.2 Evaluación

Durante la aplicación del proceso de autoaprendizaje de la programación se debe cumplir las acciones detalladas en cada fase, de esa manera el estudiante supera las limitaciones presentadas en el diagnóstico.

La evaluación es de proceso por lo que cada sesión se debe identificar a través de indicadores, el progreso de los estudiantes, para ello se debe trabajar instrumentos de evaluación que sean registros del progreso de los estudiantes.

Cada fase debe evidenciar logros que permitan ir valorando las acciones de la estrategia de acuerdo con los objetivos previstos.

Aspectos para evaluar

- Estilos de aprendizaje del estudiante
- Componentes motivacionales basados en gamificación
- Diagnóstico de conceptos previos de los estudiantes.
- Reportes de participantes según logros luego de la aplicación de la gestión informatizada de la programación
- Reporte de retroalimentación por parte de alumnos
- Reporte de retroalimentación por parte de los docentes.

Evaluación General

Al finalizar la aplicación de la estrategia se valoran las fases de cada etapa detectando logros y dificultades superadas, así como reformulación de limitaciones en algunas fases para que a través de valoraciones se rectifique y se mejore.

Fuentes de evaluación

- Observación de los estudiantes en el momento que desarrollen las acciones de cada fase.
- Registro de evidencias.
- Exposiciones de los trabajos realizados durante las sesiones.
- Participaciones con opiniones, críticas y reflexiones del proceso.
- Listas de cotejo o rúbricas con indicadores que señalen los procesos que deben incluirse en sus actuaciones.
- Resultado de las evaluaciones.

3.5. Valoración y corroboración de los resultados de la estrategia de autoaprendizaje de la programación

3.5.1. Valoración de los resultados

Para la validación y corrobación de los resultados se aplicó un pre test y post test de la aplicación del aporte práctico. Se tomó en consideración a los 120 estudiantes para todas las etapas y fases de la ejemplificación del aporte práctico y se les considero como grupo experimental. El grupo control estuvo conformado por 120 estudiantes que no tuvieron contacto con la estrategia de autoaprendizaje de la programación y aprenden los conceptos de forma tradicional.

Etapa 1: Lógica contextual gamificada del autoaprendizaje de la programación

- **Fase 1: Reconocimiento de la lógica contextual gamificada del autoaprendizaje en programación**

En esta fase se pudo diagnosticar el estilo de aprendizaje de los estudiantes que permitan apropiarse de los conceptos de programación, para lo cual nos basados en los estilos de aprendizaje propuestos por Solomar-Felder, cuyas siglas son VARK (Visual, Auditivo, Lectura-Escritura, kinestésico), como se muestra en la tabla Fig. 14 el estilo de aprendizaje que predomina entre los estudiantes de Ingeniería de Sistemas es el Multimodal (30 estudiante) dado que los estudiantes pueden aprender con una mezcla de estilos. Seguido del estilo de aprendizaje visual (26 estudiantes), el estilo lector/Escritor (24) indica que estudiante lee y escribe para poder aprender, del mismo modo el estilo de aprendizaje kinestésico (24 estudiante) permite entender que los estudiantes usan todo su cuerpo para poder aprender, como ultimo estilo de aprendizaje, los estudiantes de Ingeniería

de Sistemas usan el oído (16) posiblemente sea referenciado para la escucha de clases e indicaciones.

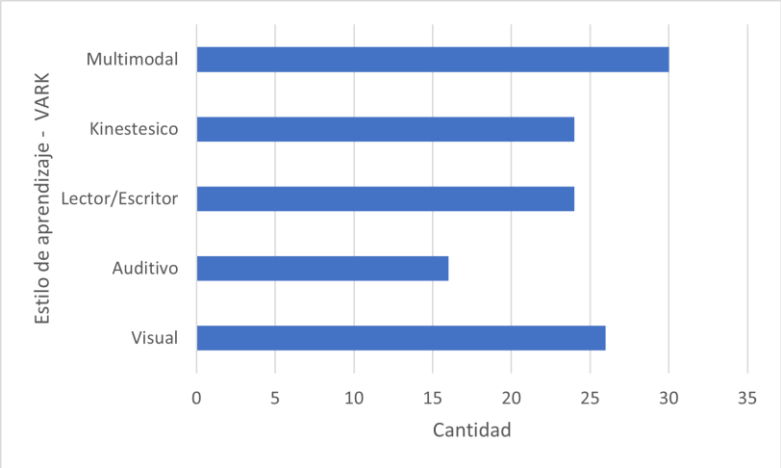


Fig. 14 Estilo de Aprendizaje de los estudiantes del curso de programación

Así mismo, para esta etapa se establecen las motivaciones intrínsecas y extrínsecas que generan sentimientos positivos en los estudiantes**.**

La Fig. 15 muestra que los estudiantes se motivan intrínsecamente cuando valoran lograr algo que les permita aprender y sienten una motivación de superación personal (120 estudiantes), así mismo indican poseer un sentido de responsabilidad y satisfacción por lo logrado y un reto personal de lograr ser profesionales (100), finalmente consideran que son los principales responsables de su aprendizaje.

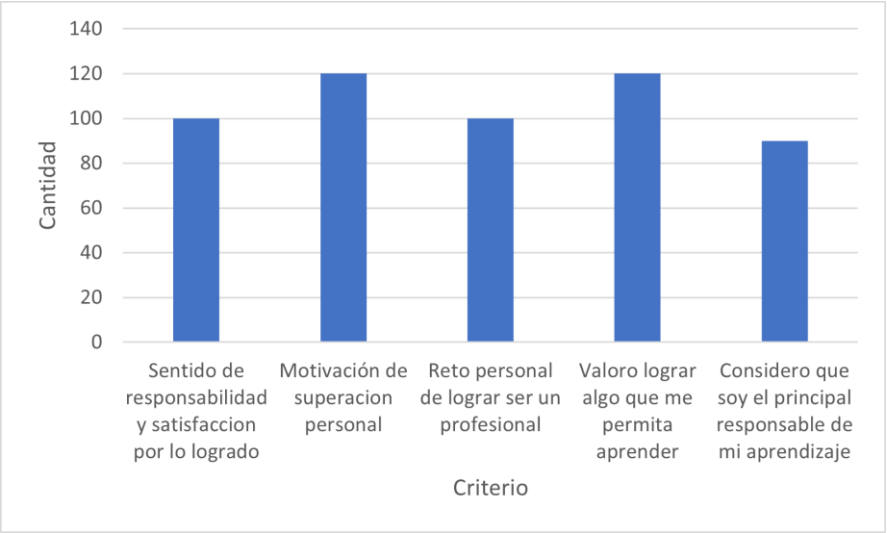
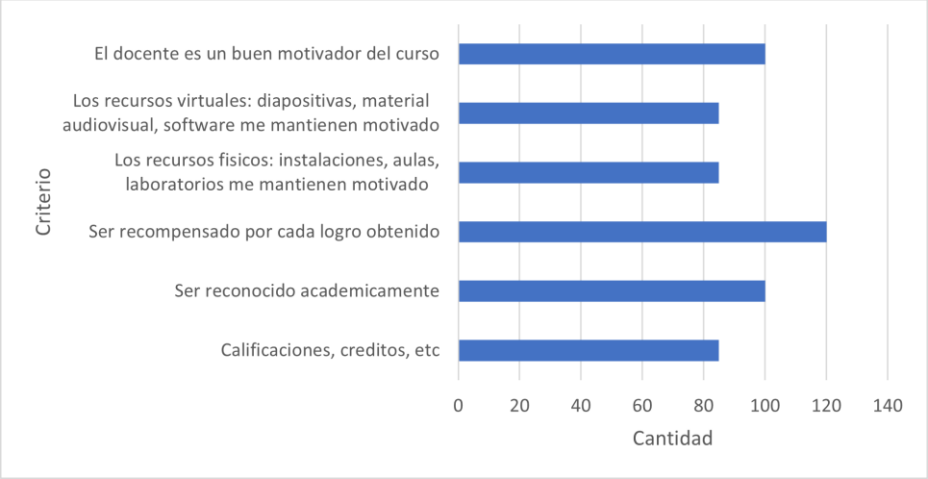


Fig. 15 Motivación intrínseca de los estudiantes del curso de programación



En la Fig. se destaca que los estudiantes extrínsecamente están motivados por ser recompensados por cada logro obtenido (120), así como también por ser reconocidos académicamente y los motiva tener un buen docente como guía (120),

consideran que los recursos físicos y virtuales son necesarios, pero no indispensables para lograr buenos resultados en sus cursos.

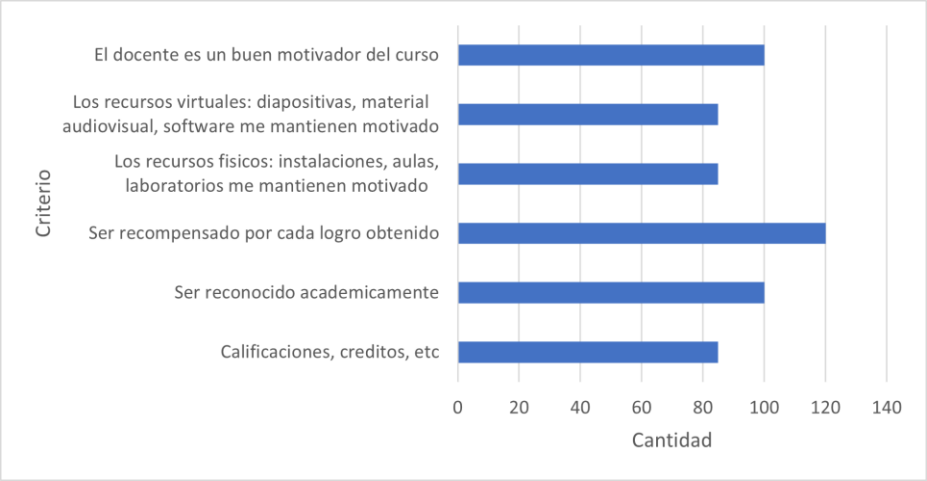


Fig. 16 Motivación extrínseca de los estudiantes de Ingeniería de Software

Posteriormente se evaluó los saberes previos de los estudiantes como línea base para la adquisición de nuevos conocimientos. Como se aprecia en la Tabla 1 los estudiantes tienen dificultad para resolver problemas de lógica matemática y algorítmica, ambos conocimientos son necesarios para poder iniciar adecuadamente el curso. Sin embargo, los estudiantes en promedio conocen temas de actualidad basados en el uso de tecnología.

Secciones de evaluación del examen de entrada	**Nota Promedio**
Resolución de problemas usando lógica matérica	12.5
Resolución de problemas usando lógica algorítmica	5.5
Resolución de problemas basados en actualidad tecnológica	16

Tabla 1 Sabes previos de los estudiantes de Ingeniería de Software

Finalmente pudimos comprobar que existe una coherencia entre la competencia general y especifica del curso y los estándares nacionales, debido a que la Escuela de Ingeniería de Sistemas ha recibido su acreditación nacional SINEACE.

- **Fase 2: Interpretación teórica-personalizada de la programación**

En esta fase se busca identificar los conceptos previos y de mayor dificultad en el curso, para lo cual se realizó una encuesta a 50 estudiantes que ya llevaron el curso de programación. Los resultados se aprecian en la Fig. 17, se coincide que los conceptos a reforzar deben ir acompañado de ejercicios con enunciados a problemas del mundo real, los conceptos de mayor dificultad están relacionados a los ciclos repetitivos while, do-while, for aplicados a figuras y matrices aplicadas al desarrollo de mapas en un juego.

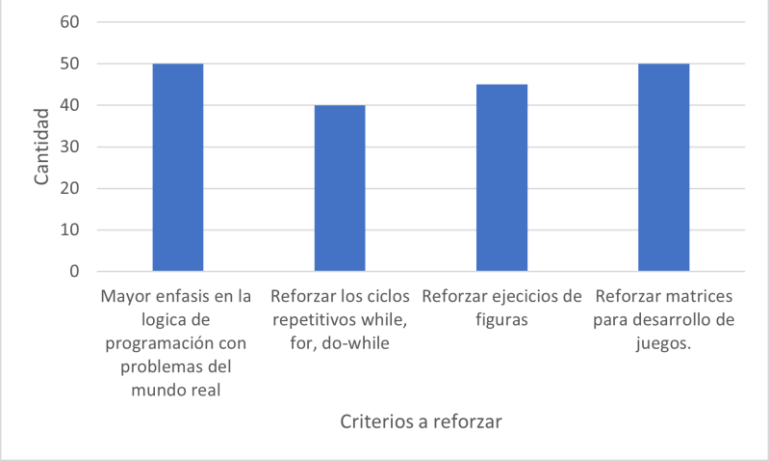


Fig. 17 Criterios o conocimientos a reforzar

Para generar actividades motivacionales se hizo uso de varias herramientas tales com: hakoot que permite motivar al estudiante basado en recompensas y

reconocimiento, Trello que permite al estudiante realizar trabajos colaborativos y scratch que permite al estudiante desarrolla actividades visualmente interactivas. En la Fig. 18 se muestra una de las actividades diseñadas para ser usada con los estudiantes con el objetivo de reforzar los conocimientos teóricos.

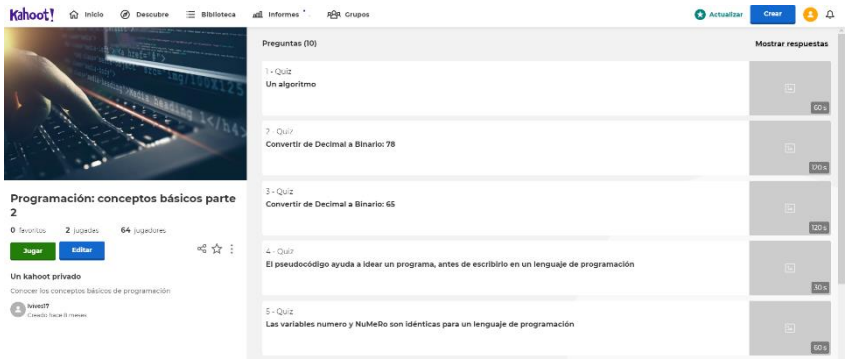


Fig. 18 Enseñanza con Hakoot

En la Fig. 19 se apreciar que scratch permite a los estudiantes aprender conceptos de programación basado en el desarrollo de un juego, ideal para aprender conceptos básicos de programación.

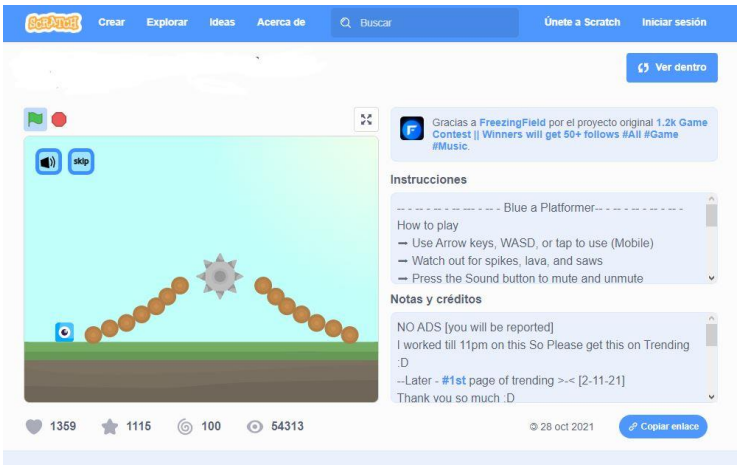


Fig. 19 Enseñanza con Scratch

- **Fase 3: Sistematización autoguiada-interactiva de contenidos de programación.**

En el desarrollo de la fase 3, se identificó los contenidos que son candidatos para sistematizarse, tomando en cuenta el estilo de aprendizaje y los factores de motivación. La Tabla 2 muestra los contenidos del sílabo, cabe mencionar que se agregaron los contenidos que generan mayor dificultad de aprendizaje para los estudiantes, tales como figuras con estructuras repetitivas y juegos.

Contenidos	**Contenido para Sistematizar**		**Estilo de Aprendizaje para la sistematización de contenidos**					**Motivación**		
	Si	**No**	**V**	**A**	**R**	**K**	**M**	**estrellas**	**Corazones**	**aplausos**
Conceptos de básicos de Programación	x		x					x	x	x
Estructura if - else	x				x			x	x	x
Estructura switch										
Estructura repetitiva while		x			x		x	x	x	x
Estructura repetitiva Do-While		x			x		x	x	x	x
Estructura repetitiva for		x			x		x	x	x	x
Figuras con estructuras repetitivas		x			x		x	x	x	x
Vectores	x		x		x			x	x	x
Estructuras										
Matrices										
Matrices para juegos	x		x		x		x	x	x	x

Tabla 2 Contenidos según el Syllabus del curso, que serán sistematizados

Etapa 2: Sistematización autoguiada - interactiva de contenidos gamificados de la programación

- **Fase 4: Diseño de actividades de gamificación en la programación**

En esta fase se diseña el diagrama lógico del modelo de autoaprendizaje de la programación basado en información sobre estilos de aprendizaje, prueba diagnóstica,

habilidades cognitivas (Competencias del curso), conceptos relevantes o con dificultad para aprender, se consideran los diseños de sesiones basado en estrategias de gamificación. En la Fig. 20 se muestra el diagrama de la arquitectura lógica del modelo de aprendizaje de la programación, que tiene como datos de entrada el test de estilos de aprendizaje, prueba diagnóstica, el proceso consiste en brindar material autoguiado interactivo para luego resolver ejercicios basado en técnicas de gamificación, el alumno alcanza el puntaje que le permita avanzar de niveles, caso contrario regresará a revisar conceptos. El sistema concluye y tiene como salida los reportes respectivos.

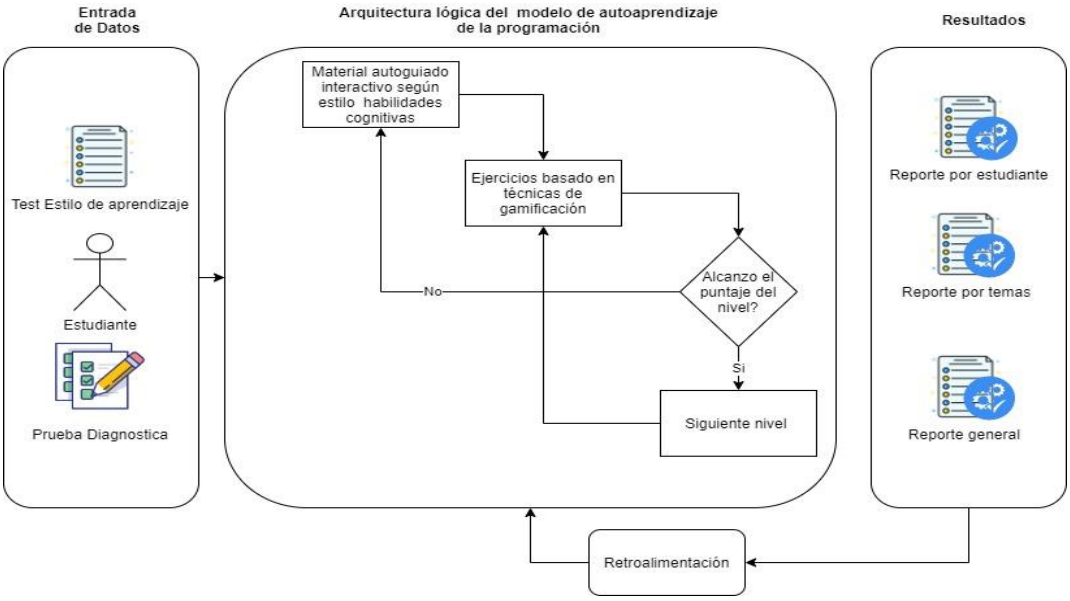


Fig. 20 Arquitectura lógica del modelo de autoaprendizaje de la programación

- **Fase 5: Gestión informatizada del conocimiento de la programación**

En la fase 5 se consideran actividades que permitan obtener información en base a reportes. En la Fig. 20 se muestra como salida la obtención de reportes por estudiante, por sección por tema tratado, reporte general con la finalidad de tomar decisiones o hacer las correcciones respectivas durante las sesiones de clase presenciales o virtuales.

- **Fase 6: Generalización del contenido de programación para el desarrollo de software.**

La retroalimentación es un factor importante para el éxito del modelo es por eso sé que ha incorporado una estrategia de obtención del proceso de retroalimentación que va desde los reportes hacia el modelo

3.5.2. Corroboración estadística de las transformaciones logradas

Luego de aplicar la estrategia de autoaprendizaje de la programación, basada en sus dimensiones Lógica contextual gamificada del autoaprendizaje en programación y la Sistematización autoguiada - interactiva de contenidos gamificados de la programación, se logró comprobar que los estudiantes se apropian de los conocimientos debido a que realizan actividades adicionales con técnicas de gamificación que le permiten afianzar sus conocimientos y poner en práctica todo lo aprendido en el curso y en su vida diaria.

La Tabla 3 presenta los resultados de la pre test y post test, se aprecia que existen un cambio de pensamiento desde el aspecto negativo (nunca - casi nunca), hacia el aspecto positivo (siempre - casi siempre). La dimensión Lógica contextual gamificada del autoaprendizaje en programación y sus fases: i) Reconocimiento contextual formativo del autoaprendizaje de la programación, ii) Interpretación teórica-personalizada de la programación, y iii) Sistematización autoguiada-interactiva de contenidos de programación. Lograron que un 63% y 21% de estudiantes comprendan la importancia de identificar sus estilos de aprendizaje, diagnosticar sus conocimientos previos para personalizar los contenidos de la programación, comprender la coherencia entre lo teórico – practico y la realidad. Así mismo la dimensión Sistematización autoguiada - interactiva

de contenidos gamificados de la programación y sus fases: i) Diseño de actividades de gamificación en la programación, ii) Gestión informatizada del conocimiento de la programación, y iii) Generalización del contenido de programación para el desarrollo de software. Permitieron que un 92% de estudiantes valore aprender conocimientos de programación de forma interactiva y autoguiada con técnicas de gamificación y recompensas extrínsecas e intrínsecas para el logro de sus objetivos.

VARIABLE DEPENDIENTE	**Dimensiones**	**Indicadores**		**Técnicas de Investigación**			
				Pre-Test		**Post Test**	
			categorías	**N**	**%**	**N**	**%**
Codificación en el desarrollo y manteniendo de software	Lógica contextual gamificada del autoaprendizaje en programación	Reconocimiento contextual formativo del autoaprendizaje de la programación	Nunca	80	67%	5	4%
			Casi nunca	30	25%	5	4%
			A veces	10	8%	10	8%
			Casi siempre	0	0%	25	21%
			Siempre	0	0%	75	63%
		Interpretación teórica-personalizada de la programación	Nunca	75	63%	5	4%
			Casi nunca	35	29%	5	4%
			A veces	10	8%	10	8%
			Casi siempre	0	0%	25	21%
			Siempre	0	0%	75	63%
		Sistematización autoguiada-interactiva de contenidos de programación	Nunca	76	63%	5	4%
			Casi nunca	30	25%	5	4%
			A veces	15	13%	10	8%
			Casi siempre	0	0%	25	21%
			Siempre	0	0%	75	63%
	Sistematización autoguiada - interactiva de contenidos gamificados de la programación	Diseño de actividades de gamificación en la programación	Nunca	90	75%	2	2%
			Casi nunca	30	25%	3	3%
			A veces	0	0%	5	4%
			Casi siempre	0	0%	35	29%
			Siempre	0	0%	75	63%
		Gestión informatizada del conocimiento de la programación	Nunca	100	83%	2	2%
			Casi nunca	20	17%	3	3%
			A veces	0	0%	5	4%
			Casi siempre	0	0%	35	29%

			Siempre	0	0%	75	63%
		Generalización del contenido de programación para el desarrollo de software.	Nunca	100	83%	2	2%
			Casi nunca	20	17%	3	3%
			A veces	0	0%	5	4%
			Casi siempre	0	0%	35	29%
			Siempre	0	0%	75	63%

Tabla 3 Resumen comparativo de pre-prueba y post-prueba luego de aplicar la estrategia de auto aprendizaje de la programación y sus dimensiones

Así mismo, en la Fig. 21 se muestra el comparativo de notas alcanzadas en las primeras 8 semanas del curso de programación 1. Las secciones A y B fueron consideradas como grupo control 1 (30 estudiantes cada sección), estudiantes donde se aplicó la estrategia de autoaprendizaje de la programación y que forman parte del grupo de investigación descrito líneas arriba. Las secciones C y D fueron consideradas como grupo no controlado (30 estudiantes cada sección), siguieron la enseñanza tradicional. Se evidencia que el grupo control 1 logra un 65% en promedio de aprobados en la evaluación parcial, 67% de aprobados en la práctica calificada 1 y un 100% en el trabajo parcial. Sin embargo, las secciones C y D solo lograron 50% de aprobados en la evaluación parcial, 58% en la práctica calificada 1 y 85% en el trabajo parcial. Se evidencian mejores resultados de los estudiantes que aprenden con la estrategia de autoaprendizaje de la programación basada en un modelo de gamificación.

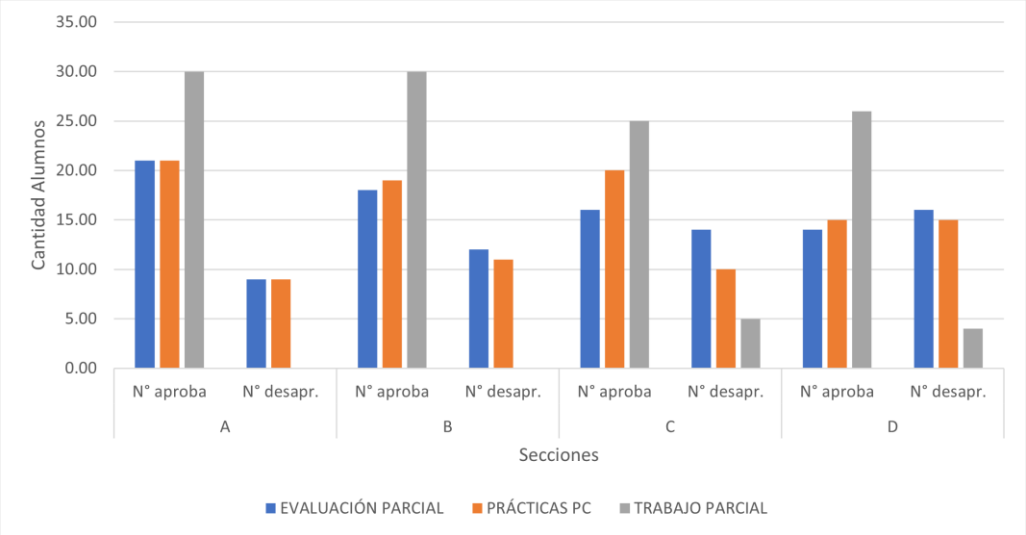


Fig. 21 Pre test y Post test según tipo de nota aplicada a los estudiantes

IV. CONCLUSIONES

- Se logró caracterizar el proceso de autoaprendizaje de la programación y su dinámica donde se manifiesta la inconsistencia entre las concepciones teóricas y prácticas que limitan la sistematización del autoaprendizaje para el logro de la apropiación de contenidos en el desarrollo de software en estudiantes del curso de programación. En tal sentido, como aporte teórico del modelo de autoaprendizaje de la programación, revelamos 2 dimensiones: Lógica contextual gamificada del autoaprendizaje en programación y Sistematización autoguiada - interactiva de contenidos gamificados de la programación. Además, consideramos que el eje fundamental que une las dimensiones es la ssistematización autoguiada-interactiva

de contenidos de programación y se establece la importancia de las técnicas de gamificación para que el estudiante se apropie de sus conocimientos.

- Se determino las tendencias históricas del proceso de autoaprendizaje de la programación y su dinámica, estructurándolo en 5 etapas, sin embargo, concluimos que los avances descritos hasta la fecha en el desarrollo del proceso autoaprendizaje de la programación, aun son insuficientes en cuanto a la dinamización de sus componentes plan curricular, estudiante, estrategias de aprendizaje que permitan una sistematización y apropiación de los contenidos por parte del estudiante con la finalidad de escribir códigos de software.

- Se diagnosticó el estado actual de la dinámica del proceso de autoaprendizaje de la programación y se concluyó que los estudiantes no toman en cuenta el reconocimiento contextual formativo del autoaprendizaje de la programación. Carecen de una interpretación teórica-personalizada de la programación. Se identifica que el curso tiene una limitada sistematización autoguiada-interactiva de contenidos de programación. No existe un diseño de actividades de gamificación en la programación. No evidencian una adecuada gestión informatizada del conocimiento de la programación. Y no comprueban una adecuada generalización del contenido de programación para el desarrollo de software.

- Se aplico el modelo teórico del proceso de autoaprendizaje de la programación del ingeniero de software a 120 estudiantes. Se determinó que el estilo de aprendizaje más usado por los estudiantes es el visual y multimodal. Se identifico que la motivación intrínseca de los estudiantes se basa en una motivación de superación

personal y la valoración por lograr algo que les permita aprender, así mismo el estudiante busca ser recompensado por cada logro obtenido como motivación extrínseca. Posteriormente los resultados de una prueba diagnostican indican debilidades en la apropiación de conceptos sobre lógica algorítmica. Se reconocen conceptos como estructuras repetitivas, matrices para elaboración de juegos como temas difíciles de aprender, seguido de aprender con ejercicios del mundo real. Finalmente se construye una arquitectura lógica del modelo de autoaprendizaje de la programación que tiene como entrada una prueba diagnóstica, los estilos de aprendizaje y las competencias del curso, el proceso permite una constante retroalimentación si no se llega al puntaje adecuado para subir de nivel y como salida tenemos los reportes de los logros alcanzados por el estudiante.

- Se validaron los resultados atrás ves de un pre test y un post test. Se lograron los siguientes resultados: El 75% de estudiantes toman en cuenta el reconocimiento contextual formativo del autoaprendizaje de la programación. el 75% de estudiantes considera necesaria la interpretación teórica-personalizada de la programación. El 75% de estudiantes identifican que el curso tiene una adecuada sistematización autoguiada-interactiva de contenidos de programación. El 75 % de estudiantes considera que existe un diseño de actividades de gamificación en la programación. El 75% de estudiantes evidencia una adecuada gestión informatizada del conocimiento de la programación. El 75% de estudiantes comprueban una adecuada generalización del contenido de programación para el desarrollo de software.

IV. RECOMENDACIONES

Se recomienda lo siguiente:

- Profundizar en la estrategia de autoaprendizaje de la programación agregando componentes tecnológicos como machine learning que permita automatizar los contenidos a ser personalizados para cada estudiante.
- Profundizar en los procesos de retroalimentación, actualizando constantemente los contenidos interactivos del curso.
- Establecer una unidad conformada por docentes, que permita monitorear la estrategia de autoaprendizaje de la programación.

V. REFERENCIAS

Abbasi, S., Tabbassum, K., Kazi, H., Tunio, S., & Qureshi, S. (2021). Investigating Student ‘ s Obstacles While Learning Object Orientation. *INTERNATIONAL JOURNAL OF SCIENTIFIC & TECHNOLOGY RESEARCH*, *06*, 7–11.

Abreu Alvarado, Y., Barrera Jiménez, A. D., Worosz, T. B., & Vichot, I. B. (2018). El proceso de enseñanza-aprendizaje de los Estudios Lingüísticos: su impacto en la motivación hacia el estudio de la lengua. *Mendive. Revista de Educación*, *16*(4), 610–623.

Araceli, J., & Barajas, C. (n.d.). *Detección de los niveles de madurez de la gestión del conocimiento en el aprendizaje de un curso de programación de ordenadores*.

Barcia, J., & Carvajal, B. (2015). EL PROCESO DE ENSEÑANZA APRENDIZAJE

EN LA EDUCACIÓN SUPERIOR ISSN 1390-9010. *Revista Electrónica Formación y Calidad Educativa (REFCalE)*, 59–72.

Becker, B. A., & Quille, K. (2019). 50 years of CS1 at SIGCSE: A review of the evolution of introductory programming education research. *SIGCSE 2019 - Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, *1*, 338–344. https://doi.org/10.1145/3287324.3287432

Calle, J. C. C. (2021). Inverted Classroom to Develop Self-regulated Learning in University Students in Times of Pandemic. *Revista Gestão Inovação e Tecnologias*, *11*(3), 727–740. https://doi.org/10.47059/revistageintec.v11i3.1971

Campbell, J., Horton, D., Craig, M., & Gries, P. (2014). Evaluating an inverted CS1. *SIGCSE 2014 - Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, 307–312. https://doi.org/10.1145/2538862.2538943

Chiung-Sui Chang, Chung, C., & Chang, J. A. (2020). Influence of problem-based learning games on effective computer programming learning in higher education. *Education Tech Research Dev*, *68*, 2615–2634. https://doi.org/10.1007/s11423-020-09784-3

Code.org. (2021). *La hora del Código*. https://hourofcode.com/es

D. Lescay, Romero, E., & Mestre, U. (2015). Modelo holístico-configuracional de la Educación Antialcohólica de estudiantes de carreras pedagógicas Durante el proceso de formación de los estudiantes de carreras pedagógicas , para la Salud , por lo que constituyen las vías para el desarrollo de una d. *Didáctica y Educación.*, *6*(3), 163–174.

Davidson, N., & Major, C. H. (2014). Boundary Crossings: Cooperative Learning, Collaborative Learning, and Problem-Based Learning. *Journal on Excellence in College Teaching*, *25*(3&4), 7–55.

Franco-González, D., García-Herrera, D. G., Guevara-Vizcaíno, C. F., & Erazo-Álvarez, J. C. (2020). Scratch para la enseñanza de Lenguaje de Programación en Primero de Bachillerato. *Revista Arbitrada Interdisciplinaria Koinonía*, *5*(5), 398. https://doi.org/10.35381/r.k.v5i5.1050

Gomes, A., Ke, W., Lam, C. T., Marcelino, M. J., & Mendes, A. (2019). Student motivation towards learning to program. *Proceedings - Frontiers in Education Conference, FIE*, *2018-Octob*. https://doi.org/10.1109/FIE.2018.8659134

Guevara, G., Verdesoto, A., & Castro, N. (2020). Metodologías de investigación educativa (descriptivas, experimentales, participativas, y de investigación-acción). *Recimundo*, *4*(3), 163–173. https://doi.org/10.26820/recimundo/4.(3).julio.2020.163-173

Holz-Clause, M., Guntuku, D., Koundinya, V., Clause, R., & Singh, K. (2015). Current and future trends in higher education learning: Implications for curriculum design and delivery. *Handbook of Research on Enhancing Teacher Education with Advanced Instructional Technologies*, *January*, 277–292. https://doi.org/10.4018/978-1-4666-8162-0.ch015

Impagliazzo, J., Clear, A., & Alrumaih, H. (2018). Developing an overview of computing/engineering curricula via the CC2020 project. *IEEE World Engineering Education Conference - EDUNINE-2018*, 1–4. https://doi.org/10.1109/EDUNINE.2018.8450965

Kesler, A., Shamir-Inbal, T., & Blau, I. (2021). Active Learning by Visual Programming: Pedagogical Perspectives of Instructivist and Constructivist Code Teachers and Their Implications on Actual Teaching Strategies and Students' Programming Artifacts. *Journal of Educational Computing Research*. https://doi.org/10.1177/07356331211017793

Lister, R. (2020). On the cognitive development of the novice programmer: And the development of a computing education researcher. *ACM International Conference Proceeding Series*. https://doi.org/10.1145/3442481.3442498

Luxton-Reilly, A. (2016). Learning to program is easy. *Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE, 11-13-July*, 284–289. https://doi.org/10.1145/2899415.2899432

Luxton-Reilly, A., Simon, Albluwi, I., Becker, B. A., Giannakos, M., Kumar, A. N., Ott, L., Paterson, J., Scott, M. J., Sheard, J., & Szabo, C. (2018). Introductory programming: A systematic literature review. In *Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE*. https://doi.org/10.1145/3293881.3295779

Malik, S. I., Tawafak, R. M., & Shakir, M. (2021). Aligning and assessing teaching approaches with solo taxonomy in a computer programming course. *International Journal of Information and Communication Technology Education*, *17*(4), 1–15. https://doi.org/10.4018/IJICTE.20211001.oa5

Manzano-León, A., Camacho-Lazarraga, P., Guerrero, M. A., Guerrero-Puerta, L., Aguilar-Parra, J. M., Trigueros, R., & Alias, A. (2021). Between level up and game over: A systematic literature review of gamification in education. *Sustainability (Switzerland)*, *13*(4), 1–14. https://doi.org/10.3390/su13042247

Mehmood, E., Abid, A., Farooq, M. S., & Nawaz, N. A. (2020). Curriculum, Teaching and Learning, and Assessments for Introductory Programming Course. *IEEE Access*, *8*, 125961–125981. https://doi.org/10.1109/ACCESS.2020.3008321

MIT. (2021). *SCRATCH*. https://scratch.mit.edu/

Molina, A., & Mejia, H. (2013). *La investigación acción y el enfoque holístico configuracional en una pedagogía inclusiva, lúdica y creativa para la estimulación de las competencias afectiva* (Vol. 6, Issue 1). https://doi.org/10.21676/23897856.83

Moreira, C. J. S., Beltron, C. R. A., & Beltrón, C. V. C. (2021). Aprendizaje significativo una alternativa para transformar la educación. *Dominio de Las Ciencias*, *7*(2), 915–924. https://dominiodelasciencias.com/ojs/index.php/es/article/view/1835/3708

Moreira, F., Ferreira, M. J., Pereira, C. S., Escudero, D. F., Collazos, C., & Gomes, A. (2021). Higher Education Teachers Training (HET2) Model: Active Learning in Higher Education Environment. *Advances in Intelligent Systems and Computing*, *1367 AISC*, 103–112. https://doi.org/10.1007/978-3-030-72660-7_11

Nurulain Mohd Rum, S., & Zolkepli, M. (2018). Metacognitive Strategies in Teaching and Learning Computer Programming. *International Journal of Engineering & Technology*, *7*(4.38), 788. https://doi.org/10.14419/ijet.v7i4.38.27546

Ortega-ruipérez, B. (2021). COMPETENCIA EN CREACIÓN DE CONTENIDOS DIGITALES CON GAMIFICACIÓN. *Revista Educación y Tecnología*, *14*, 1–22.

Pinillos, J. A. H., & Santa Cruz, F. F. (2021). Metodologías constructivistas en educación superior : impulsoras del pensamiento divergente Constructivist methodologies in higher education : divergent thinking drivers. *Revista de Educación*, *23*, 241–261.

Robberts, A. S., & Employability, A. (2019). Work in Progress: Enabling Learning Environments for Underprepared Engineering Students: Blending Game-Based and Project-Oriented Methodologies. *IEEE Global Engineering Education Conference (EDUCON)*, 722–726. https://doi.org/10.1109/EDUCON.2019.8725040

Rojas-López, A., & García-Peñalvo, F. J. (2020). Assessment of computational thinking

skills to predict student learning and retention in the subject programming computer in higher education. *Revista de Educacion a Distancia*, *20*(63). https://doi.org/10.6018/RED.409991

Sampieri, R. H., & P., M. T. C. (2018). *Metodología de la investigación: las rutas cuantitativa, cualitativa y mixta*. McGraw Hill México.

Selby, C. C. (2015). Relationships: Computational thinking, Pedagogy of programming, And bloom's taxonomy. *ACM International Conference Proceeding Series*, *09-11-Nove*, 80–87. https://doi.org/10.1145/2818314.2818315

Sobral, S. R. (2021a). Pair Programming and the Level of Knowledge in the Formation of Pairs. *Advances in Intelligent Systems and Computing*, *1367 AISC*, 212–221. https://doi.org/10.1007/978-3-030-72660-7_21

Sobral, S. R. (2021b). Strategies on Teaching Introducing to Programming in Higher Education. *Advances in Intelligent Systems and Computing*, *1367 AISC*, 133–150. https://doi.org/10.1007/978-3-030-72660-7_14

Soler Pellicer, Y., Cárdenas Zea, M., Aguirre Pérez, R., Castro Blanco, Y., & Lezcano Brito, M. G. (2020). Visualización Dinámica, Una Opción Para La Enseñanza-Aprendizaje De La Programación De Computadoras. *Holos*, *2*, 1–20. https://doi.org/10.15628/holos.2020.4241

Takada, S., Cuadros-Vargas, E., Impagliazzo, J., Gordon, S., Marshall, L., Topi, H., van der Veer, G., & Waguespack, L. (2020). Toward the visual understanding of computing curricula. *Education and Information Technologies*, *25*(5), 4231–4270. https://doi.org/10.1007/s10639-020-10127-1

Tang Qin, Parthasarathy Poovendran, & S. BalaMurugan. (2021). Student-Centered Learning Environments Based on Multimedia Big Data Analytics Tang. *Arabian Journal for Science and Engineering*. https://doi.org/10.1007/s13369-021-05962-4 RESEARCH

Anexos

Anexo 01: Matriz de consistencia

Anexo 02: Operacionalización de las variables.

Anexo 03: Instrumentos

Anexo 04: Validación de instrumentos por juicio de expertos

Anexo 05: Validación de los aportes de la investigación

Anexo 06: Consentimiento Informado

Anexo 07: Aprobación del Informe de Tesis

USS | UNIVERSIDAD SEÑOR DE SIPÁN

ANEXO N° 1 MATRIZ DE CONSISTENCIA

Manifestaciones del problema	Los estudiantes atraviesan dificultades al momento de transformar un problema de lenguaje natural a lenguaje de programación. Se evidencia la falta de aplicación de los conceptos de la lógica de programación en la solución de problemas que se puedan presentar en el quehacer profesional. Constantemente se evidencia la carencia de análisis de casos para la construcción de algoritmos y

	programación apropiados para la programación y solución de un problema dado. Se aprecia la falta de sistematización en el diseño de software educativo, siendo esta fase del desarrollo del software la más crítica, según la ingeniería del software.
Problema	Insuficiencias en la apropiación de los contenidos de programación que limitan la codificación en el desarrollo de software
Causas que originan el Problema	Al iniciar un tema de estudio no siempre se diagnostica, el estado de las estructuras cognitivas de los estudiantes, para la apropiación de los nuevos contenidos. No se tienen presente las limitaciones existentes para el desarrollo de la asignatura. Aunque se haya realizado un diagnóstico, las variantes utilizadas, en cuanto a métodos o medios, no posibilitan a todos vencer las dificultades precedentes. Las concepciones teóricas y prácticas del proceso de enseñanza aprendizaje de la programación no toman en cuenta la sistematización como vía fundamental en el logro de la apropiación de los contenidos en el contexto
Objeto de la Investigación	Proceso de autoaprendizaje de la programación

Objetivo General de la Investigación	Aplicar una estrategia autoaprendizaje de la programación, sustentada en un modelo de gamificación integral contextualizado para la codificación en el desarrollo y mantenimiento de software
Objetivos específicos	g) Caracterizar el proceso del autoaprendizaje de la programación y su dinámica. h) Determinar las tendencias históricas del proceso de autoaprendizaje de la programación y su dinámica. i) Diagnosticar el estado actual de la dinámica del proceso de autoaprendizaje de la programación. j) Elaborar el modelo teórico del proceso de autoaprendizaje de la programación del ingeniero de software, mediate el empleo de estrategias de gamificación. k) Elaborar la estrategia didáctica interactiva contextualizada del proceso de autoaprendizaje de la programación del ingeniero de software, mediate el empleo de estrategias de gamificación. l) Validar y corroborar los resultados de la investigación.

Campo de la investigación	Dinámica del proceso de autoaprendizaje de la programación en la formación del ingeniero de software.
Título de la Investigación	Estrategia de autoaprendizaje de programación, sustentada en un modelo de gamificación integral contextualizado
Hipótesis	Si se elabora una estrategia de autoaprendizaje de la programación, sustentada en el modelo de gamificación integral contextualizado, que tenga en cuenta la relación entre la lógica de la sistematización formativa interactiva y la lógica del autoaprendizaje, entonces se contribuye a la codificación en el desarrollo y mantenimiento de software
Variables	VARIABLE INDEPENDIENTE: Estrategia de autoaprendizaje de la programación basada en técnicas de gamificación VARIABLE DEPENDIENTE: Codificación en el desarrollo y manteniendo de software.

UNIVERSIDAD SEÑOR DE SIPÁN

ANEXO N° 2 OPERACIONALIZACIÓN DE VARIABLES

VARIABLE DEPENDIENTE	Dimensiones	Indicadores	Técnicas e instrumentos	Fuente de verificación
Codificación en el desarrollo y manteniendo de software	Lógica contextual gamificada del autoaprendizaje en programación	Reconocimiento contextual formativo del autoaprendizaje de la programación	Revisión documentaria encuestas	Estudiantes, Docentes
		Interpretación teórica-personalizada de la programación		
		Sistematización autoguiada-interactiva de contenidos de programación		

	Sistematización autoguiada - interactiva de contenidos gamificados de la programación	Diseño de actividades de gamificacion en la programación		
		Gestión informatizada del conocimiento de la programación		

UNIVERSIDAD SEÑOR DE SIPÁN

VARIABLE INDEPENDIENTE	Dimensiones	Descripción
Estrategia de autoaprendizaje de la programación basada en un modelo de gamificación integral contextualizado	Introducción- Fundamentación.	Se establece el contexto y ubicación de la problemática a resolver. Ideas y puntos de partida que fundamentan la estrategia. Se indica la teoría en que se fundamenta el aporte propuesto.
	Diagnóstico-	Indica el estado real del objeto y evidencia el problema en torno al cual gira y se desarrolla la estrategia, protocolo, o programa, según el aporte practico a desarrollar.
	Planteamiento del objetivo general	Se desarrolla el objetivo general del aporte práctico. Se debe tener en cuenta que no es el de la investigación.

	Planeación estratégica	Se definen metas u objetivos a corto y mediano plazo que permiten la transformación del objeto desde su estado real hasta el estado deseado. Planificación por etapas de las acciones, recursos, medios y métodos que corresponden a estos objetivos. ETAPA 1: Lógica contextual gamificada del autoaprendizaje en programación ETAPA II: Sistematización autoguiada - interactiva de contenidos gamificados de la programación
	Instrumentación	Explicar cómo se aplicará, en qué condiciones, durante qué tiempo, responsables, participantes.
	Evaluación	Definición de los logros obstáculos que se han ido venciendo, valoración de la aproximación lograda al estado deseado

USS | UNIVERSIDAD SEÑOR DE SIPÁN

ANEXO N° 3 INSTRUMENTO

Estimado (a) docente:

Con la presente encuesta se pretende diagnosticar el estado actual del proceso de autoaprendizaje de la programación en la Escuela de Ingeniería de Sistemas de la Universidad Señor de Sipan y tiene como objetivo obtener información sobre determinados aspectos de la codificación de software. Por ello, le pido conteste con toda objetividad y claridad a las siguientes preguntas.

Agradezco de antemano su valioso aporte que tiene carácter de anónimo.

INSTRUCCIONES: - Lea detenidamente cada pregunta, antes de contestarla, así como sus posibles respuestas. - Para responder debe utilizar el número correspondiente de la escala que se le ofrece. Se marcará con una "X" su valoración sobre los siguientes aspectos, teniendo en cuenta la escala Likert:

Escala de Evaluación				
Nunca	**Casi nunca**	**A veces**	**Casi Siempre**	**Siempre**
1	**2**	**3**	**4**	**5**

N°	Dimensión / Criterio	Valores				
Reconocimiento contextual formativo del aprendizaje de la programación		**1**	**2**	**3**	**4**	**5**
1	Al iniciar las clases realizas un prediagnóstico del estilo de aprendizaje que posee cada uno de sus estudiantes					
2	Al iniciar las clases realizas un prediagnóstico de los saberes previos que posee el estudiante sobre lógica proposicional o lógica matemática.					
3	Identifica los objetivos y competencias del curso de programación					
4	Las actividades propuestas permiten al estudiante motivaciones intrínsecas					
5	Las actividades propuestas permiten al estudiante motivaciones extrínsecas					
6	Existe relación entre los contenidos y las competencias del curso en cuanto horas, cantidad de contenido y logros en el curso.					
Interpretación teórica-personalizada de la programación		**1**	**2**	**3**	**4**	**5**
7	Las actividades propuestas se centran en ejercicios reales que ayudan a tus estudiantes a la interpretación teórica y práctica de la programación					
8	Las actividades propuestas incluyen ejercicios grupales, para su interpretación compartida					
9	Reconoce la importancia de los temas y los que generan mayor dificultad de aprendizaje					
10	Propone actividades motivacionales precias a cada tema propuesto					
11	Establece mecanismos de retroalimentación de lo aprendido con actividades motivacionales extrínsecas					
Diseño de actividades de gamificación en la programación		**1**	**2**	**3**	**4**	**5**
11	Recopila información acerca del estilo de aprendizaje que posee el estudiante					
12	Recopila información acerca de los conceptos relevantes del curso					
13	Recopila información acerca de los conceptos que generan mayor dificultad de comprensión de los contenidos de programación					

14	Emplea aplicaciones o software que permitan la motivación extrínseca de los estudiantes					
15	Propone actividades usando conceptos de gamificación					
16	Propone actividades con casos reales					
17	Las actividades propuestas en clase son efectivamente formativas - integrales.					
18	Las actividades propuestas en clase son de naturaleza cooperativa.					
19	Promueves la comunicación entre los estudiantes, con actividades interactivas					
20	Eres consistente a lo largo de la sesión en crear un ambiente adecuado, desde el saludo inicial					
21	Interactúas con los estudiantes de una manera empática.					
22	Los materiales presentados en clase son atractivos.					
Gestión informatizada del conocimiento de la programación		**1**	**2**	**3**	**4**	**5**
23	Propone actividades usando alguna aplicación que reúna estilo de aprendizaje, estrategia de aprendizaje y motivación					
24	Genera reportes acerca del avance del estudiante según sus logros de aprendizaje					
25	Genera retroalimentación de los conceptos difíciles de aprender					
26	Genera retroalimentación de las actividades propuestas					
Sistematización autoguiada-interactiva de contenidos de programación		**1**	**2**	**3**	**4**	**5**
27	Tus estudiantes alcanzan confianza para resolver problemas computacionales					
28	Tus estudiantes, en su formación integral, aumentan sus conocimientos de programación					
29	Tus estudiantes, en su formación integral, desarrollan sus capacidades en la resolución de problemas computaciones integrándolas con a su entorno social.					
30	Tus estudiantes comunican adecuadamente su solución haciendo uso de un lenguaje técnico y formal.					

UNIVERSIDAD SEÑOR DE SIPÁN

Estimado (a) estudiante:

Con la presente encuesta se pretende diagnosticar el estado actual del proceso de autoaprendizaje a la programación en la Escuela de Ingeniería de Sistemasde la Universidad Señor de Sipan y tiene como objetivo obtener información sobre determinados aspectos de la codificación de software. Por ello, le pido conteste con toda objetividad y claridad a las siguientes preguntas.

Agradezco de antemano su valioso aporte que tiene carácter de anónimo.

INSTRUCCIONES: - Lea detenidamente cada pregunta, antes de contestarla, así como sus posibles respuestas. - Para responder debe utilizar el número correspondiente de la escala que se le ofrece. Se marcará con una “X” su valoración sobre los siguientes aspectos, teniendo en cuenta la escala Likert:

Escala de Evaluación				
Nunca	**Casi nunca**	**A veces**	**Casi Siempre**	**Siempre**
1	**2**	**3**	**4**	**5**

N°	Dimensión / Criterio	Valores				
Reconocimiento contextual formativo del aprendizaje de la programación		1	2	3	4	5
1	Al iniciar las clases conoces los temas generales que trataran a lo largo del curso					
2	Tus saberes previos te permiten resolver problemas sobre lógica proposicional o lógica matemática.					
3	Identificas los objetivos y competencias del curso de programación					
4	Las actividades propuestas te permiten estar motivado voluntariamente					
5	Las actividades propuestas permiten estar motivado por alguna recompensa en particular					
6	Existe relación entre los contenidos y las competencias del curso en cuanto horas, cantidad de contenido y logros en el curso.					
Interpretación teórica-personalizada de la programación						
7	Las actividades propuestas se centran en ejercicios reales que ayudan comprender los conceptos teóricos y prácticos de la programación					
8	Las actividades propuestas incluyen ejercicios grupales, para su interpretación compartida					
9	Reconoce la importancia de los temas y los que generan mayor dificultad de aprendizaje					
10	Resuelve actividades motivacionales precias a cada tema propuesto					
11	El docente establece mecanismos de retroalimentación de lo aprendido con actividades motivacionales extrínsecas					

Diseño de actividades de gamificación en la programación						
12	Las actividades de cada sesión de clase le permiten aprender los conceptos de programación					
13	Las actividades de cada sesión de clase le permiten estar motivado					
14	Reconoce que conceptos le generan mayor dificultad de comprensión de los contenidos de programación					
15	El docente usa algún aplicativo que le permita afianzar sus conocimientos.					
16	Las actividades propuestas tienen componentes lúdicos que lo mantengan motivado					
17	Propone actividades con casos reales					
18	Las actividades propuestas en clase son efectivamente formativas e integrales					
19	Las actividades propuestas en clase son de naturaleza cooperativa.					
20	Promueves la comunicación entre los estudiantes, con actividades interactivas					
21	Eres consistente a lo largo de la sesión en crear un ambiente adecuado, desde el saludo inicial					
22	Interactúas con tus compañeros de una manera empática.					
23	Los materiales presentados en clase son atractivos.					
Gestión informatizada del conocimiento de la programación						
24	las actividades propuestas muestran un nivel de dificultad de acuerdo con tu aprendizaje, estrategia de aprendizaje y motivación					
25	Se genera retroalimentación de los conceptos difíciles de aprender					
26	Se genera retroalimentación de las actividades propuestas					
Sistematización autoguiada-interactiva de contenidos de programación						
27	Alcanzas la confianza necesaria para resolver problemas computacionales					

28	logras aumentar tus conocimientos de programación con lo aprendido en clases.					

UNIVERSIDAD SEÑOR DE SIPÁN

ANEXO N° 4 INSTRUMENTO DE VALIDACION NO EXPERIMENTAL POR JUICIO DE EXPERTOS

1. NOMBRE DEL JUEZ		Henry Mendoza Puertas
2.	PROFESIÓN	Ingeniero de Sistemas
	ESPECIALIDAD	Ingeniería de Software
	GRADO ACADÉMICO	Magister

	EXPERIENCIA PROFESIONAL (AÑOS)	10 años
	CARGO	Docente Universitario
Título de la Investigación: Estrategia de autoaprendizaje de programación, sustentada en un modelo de gamificación integral contextualizado		
3. DATOS DEL TESISTA		
3.1	NOMBRES Y APELLIDOS	Luis Alberto Vives Garnique
3.2	PROGRAMA DE POSTGRADO	Doctorado en Ciencias de la Educación
4. INSTRUMENTO EVALUADO		1. Entrevista ( ) 2. Cuestionario (X) 3. Lista de Cotejo ( ) 4. Diario de campo ( )
5. OBJETIVOS DEL INSTRUMENTO		<u>GENERAL</u> Aplicar una estrategia autoaprendizaje de la programación, sustentada en un modelo de gamificación integral contextualizado para la codificación en el desarrollo y mantenimiento de software
		Caracterizar el proceso del autoaprendizaje de la programación y su dinámica. Determinar las tendencias históricas del proceso de autoaprendizaje de la programación y su dinámica. Diagnosticar el estado actual de la dinámica del proceso de autoaprendizaje de la programación. Elaborar el modelo teórico del proceso de autoaprendizaje de la programación del ingeniero de

	software, mediate el empleo de estrategias de gamificación. Elaborar la estrategia didáctica interactiva contextualizada del proceso de autoaprendizaje de la programación del ingeniero de sistemas, mediate el empleo de estrategias de gamificación. Validar y corroborar los resultados de la investigación.

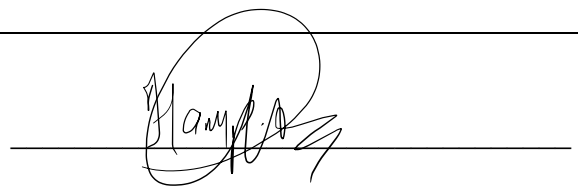
A continuación, se le presentan los indicadores en forma de preguntas o propuestas para que Ud. los evalúe marcando con un aspa (x) en "A" si está de ACUERDO o en "D" si está en DESACUERDO, SI ESTÁ EN DESACUERDO POR FAVOR ESPECIFIQUE SUS SUGERENCIAS

N	**6.** DETALLE DE LOS ITEMS DEL INSTRUMENTO	
1	Al iniciar las clases realizas un prediagnóstico del estilo de aprendizaje que posee cada uno de sus estudiantes Escala de medición: Escala de Likert	A (X) D ( ): Sugerencias:
2	Al iniciar las clases realizas un prediagnóstico de los saberes previos que posee el estudiante sobre lógica proposicional o lógica matemática. Escala de medición: Escala de Likert	A (X) D ( ): Sugerencias:
3	Identifica los objetivos y competencias del curso de programación Escala de medición: Escala de Likert	A (X) D ( ): Sugerencias:
4	Las actividades propuestas permiten al estudiante motivaciones intrínsecas Escala de medición: Escala de Likert	A(X) D ( ): Sugerencias:

5	Las actividades propuestas permiten al estudiante motivaciones extrínsecas Escala de medición: Escala de Likert	A(X) D ( ): Sugerencias:
6	Existe relación entre los contenidos y las competencias del curso en cuanto horas, cantidad de contenido y logros en el curso. Escala de medición: Escala de Likert	A(X) D ( ): Sugerencias:
7	Las actividades propuestas se centran en ejercicios reales que ayudan a tus estudiantes a la interpretación teórica y práctica de la programación Escala de medición: Escala de Likert	A(X) D ( ): Sugerencias:
8	Las actividades propuestas incluyen ejercicios grupales, para su interpretación compartida Escala de medición: Escala de Likert	A(X) D ( ): Sugerencias:
9	Reconoce la importancia de los temas y los que generan mayor dificultad de aprendizaje Escala de medición: Escala de Likert	A(X) D ( ): Sugerencias:
10	Propone actividades motivacionales precias a cada tema propuesto Escala de medición: Escala de Likert	A(X) D ( ): Sugerencias:
11	Establece mecanismos de retroalimentación de lo aprendido con actividades motivacionales extrínsecas Escala de medición: Escala de Likert	A(X) D ( ): Sugerencias:
11	Recopila información acerca del estilo de aprendizaje que posee el estudiante Escala de medición: Escala de Likert	A(X) D ( ): Sugerencias:
12	Recopila información acerca de los conceptos relevantes del curso Escala de medición: Escala de Likert	A(X) D ( ): Sugerencias:
13	Recopila información acerca de los conceptos que generan mayor dificultad de comprensión de los contenidos de programación. Escala de medición: Escala de Likert	A(X) D ( ): Sugerencias:

14	Emplea aplicaciones o software que permitan la motivación extrínseca de los estudiantes. Escala de medición: Escala de Likert	A(X) D ( ): Sugerencias:
15	Propone actividades usando conceptos de gamificación. Escala de medición: Escala de Likert	A(X) D ( ): Sugerencias:
16	Propone actividades con casos reales. Escala de medición: Escala de Likert	A(X) D ( ): Sugerencias:
17	Las actividades propuestas en clase son efectivamente formativas - integrales. Escala de medición: Escala de Likert	A(X) D ( ): Sugerencias:
18	Las actividades propuestas en clase son de naturaleza cooperativa. Escala de medición: Escala de Likert	A(X) D ( ): Sugerencias:
19	Promueves la comunicación entre los estudiantes, con actividades interactivas. Escala de medición: Escala de Likert	A(X) D ( ): Sugerencias:
20	Eres consistente a lo largo de la sesión en crear un ambiente adecuado, desde el saludo inicial. Escala de medición: Escala de Likert	A(X) D ( ): Sugerencias:
21	Interactúas con los estudiantes de una manera empática. Escala de medición: Escala de Likert	A(X) D ( ): Sugerencias:
22	Los materiales presentados en clase son atractivos. Escala de medición: Escala de Likert	A(X) D ( ): Sugerencias:
23	Propone actividades usando alguna aplicación que reúna estilo de aprendizaje, estrategia de aprendizaje y motivación. Escala de medición: Escala de Likert	A(X) D ( ): Sugerencias:

24	Genera reportes acerca del avance del estudiante según sus logros de aprendizaje Escala de medición: Escala de Likert	A(X)	D ( ): Sugerencias:
25	Genera retroalimentación de los conceptos difíciles de aprender Escala de medición: Escala de Likert	A(X)	D ( ): Sugerencias:
26	Genera retroalimentación de las actividades propuestas Escala de medición: Escala de Likert	A(X)	D ( ): Sugerencias:
27	Tus estudiantes alcanzan confianza para resolver problemas computacionales Escala de medición: Escala de Likert	A(X)	D ( ): Sugerencias:
28	Tus estudiantes, en su formación integral, aumentan sus conocimientos de programación Escala de medición: Escala de Likert	A(X)	D ( ): Sugerencias:
29	Tus estudiantes, en su formación integral, desarrollan sus capacidades en la resolución de problemas computaciones integrándolas con a su entorno social. Escala de medición: Escala de Likert	A(X)	D ( ): Sugerencias:
30	Tus estudiantes comunican adecuadamente su solución haciendo uso de un lenguaje técnico y formal. Escala de medición: Escala de Likert	A(X)	D ( ): Sugerencias:
	PROMEDIO OBTENIDO:	A(X)	D ( ):
6 COMENTARIOS GENERALES Instrumento válido en lo concerniente a contenido, criterio y constructo			
7 OBSERVACIONES			



Juez Experto

DNI 41452777

UNIVERSIDAD SEÑOR DE SIPÁN

ANEXO N° 4 INSTRUMENTO DE VALIDACION NO EXPERIMENTAL POR JUICIO DE EXPERTOS

1. NOMBRE DEL JUEZ		Eduardo Diaz Suarez
2.	PROFESIÓN	Ingeniero de Sistemas
	ESPECIALIDAD	Ingeniería de Software
	GRADO ACADÉMICO	Doctor

	EXPERIENCIA PROFESIONAL (AÑOS)	10 años
	CARGO	Docente Universitario
Título de la Investigación: Estrategia de autoaprendizaje de programación, sustentada en un modelo de gamificación integral contextualizado		
3. DATOS DEL TESISTA		
3.1	NOMBRES Y APELLIDOS	Luis Alberto Vives Garnique
3.2	PROGRAMA DE POSTGRADO	Doctorado en Ciencias de la Educación
4. INSTRUMENTO EVALUADO		5. Entrevista ( ) 6. Cuestionario (X) 7. Lista de Cotejo ( ) 8. Diario de campo ( )
5. OBJETIVOS DEL INSTRUMENTO		<u>GENERAL</u> Aplicar una estrategia autoaprendizaje de la programación, sustentada en un modelo de gamificación integral contextualizado para la codificación en el desarrollo y mantenimiento de software
		<u>ESPECÍFICOS</u> Caracterizar el proceso del autoaprendizaje de la programación y su dinámica. Determinar las tendencias históricas del proceso de autoaprendizaje de la programación y su dinámica. Diagnosticar el estado actual de la dinámica del proceso de autoaprendizaje de la programación. Elaborar el modelo teórico del proceso de autoaprendizaje de la programación del ingeniero de

	software, mediate el empleo de estrategias de gamificación. Elaborar la estrategia didáctica interactiva contextualizada del proceso de autoaprendizaje de la programación del ingeniero de sistemas, mediate el empleo de estrategias de gamificación. Validar y corroborar los resultados de la investigación.

A continuación, se le presentan los indicadores en forma de preguntas o propuestas para que Ud. los evalúe marcando con un aspa (x) en "A" si está de ACUERDO o en "D" si está en DESACUERDO, SI ESTÁ EN DESACUERDO POR FAVOR ESPECIFIQUE SUS SUGERENCIAS

N	**7.** DETALLE DE LOS ITEMS DEL INSTRUMENTO	
1	Al iniciar las clases conoces los temas generales que trataran a lo largo del curso	A (X) D ( ): Sugerencias:
2	Tus saberes previos te permiten resolver problemas sobre lógica proposicional o lógica matemática.	A (X) D ( ): Sugerencias:
3	Identificas los objetivos y competencias del curso de programación	A (X) D ( ): Sugerencias:
4	Las actividades propuestas te permiten estar motivado voluntariamente	A(X) D ( ): Sugerencias:
5	Las actividades propuestas permiten estar motivado por alguna recompensa en particular	A(X) D ( ): Sugerencias:

6	Existe relación entre los contenidos y las competencias del curso en cuanto horas, cantidad de contenido y logros en el curso.	A(X) D ( ): Sugerencias:
7	Las actividades propuestas se centran en ejercicios reales que ayudan comprender los conceptos teóricos y prácticos de la programación	A(X) D ( ): Sugerencias:
8	Las actividades propuestas incluyen ejercicios grupales, para su interpretación compartida	A(X) D ( ): Sugerencias:
9	Reconoce la importancia de los temas y los que generan mayor dificultad de aprendizaje	A(X) D ( ): Sugerencias:
10	Resuelve actividades motivacionales precias a cada tema propuesto	A(X) D ( ): Sugerencias:
11	El docente establece mecanismos de retroalimentación de lo aprendido con actividades motivacionales extrínsecas	A(X) D ( ): Sugerencias:
11	Las actividades de cada sesión de clase le permiten aprender los conceptos de programación	A(X) D ( ): Sugerencias:
12	Las actividades de cada sesión de clase le permiten estar motivado	A(X) D ( ): Sugerencias:
13	Reconoce que conceptos le generan mayor dificultad de comprensión de los contenidos de programación	A(X) D ( ): Sugerencias:
14	El docente usa algún aplicativo que le permita afianzar sus conocimientos.	A(X) D ( ): Sugerencias:
15	Las actividades propuestas tienen componentes lúdicos que lo mantengan motivado	A(X) D ( ): Sugerencias:

16	Propone actividades con casos reales	A(X) D ( ): Sugerencias:
17	Las actividades propuestas en clase son efectivamente formativas e integrales	A(X) D ( ): Sugerencias:
18	Las actividades propuestas en clase son de naturaleza cooperativa.	A(X) D ( ): Sugerencias:
19	Promueves la comunicación entre los estudiantes, con actividades interactivas	A(X) D ( ): Sugerencias:
20	Eres consistente a lo largo de la sesión en crear un ambiente adecuado, desde el saludo inicial	A(X) D ( ): Sugerencias:
21	Interactúas con tus compañeros de una manera empática.	A(X) D ( ): Sugerencias:
22	Los materiales presentados en clase son atractivos.	A(X) D ( ): Sugerencias:
23	las actividades propuestas muestran un nivel de dificultad de acuerdo a tu aprendizaje, estrategia de aprendizaje y motivación	A(X) D ( ): Sugerencias:
24	Se genera retroalimentación de los conceptos difíciles de aprender	A(X) D ( ): Sugerencias:
25	Se genera retroalimentación de las actividades propuestas	A(X) D ( ): Sugerencias:
26	Alcanzas la confianza necesaria para resolver problemas computacionales	A(X) D ( ): Sugerencias:

27	Logras aumentar tus conocimientos de programación con lo aprendido en clases.	A(X) D ( ): Sugerencias:
28	Al iniciar las clases conoces los temas generales que trataran a lo largo del curso	A(X) D ( ): Sugerencias:
	PROMEDIO OBTENIDO:	A(X) D ( ):
8 COMENTARIOS GENERALES Instrumento válido en lo concerniente a contenido, criterio y constructo		
9 OBSERVACIONES		

Juez Experto
DNI 42461628

UNIVERSIDAD SEÑOR DE SIPÁN

ANEXOS N° 5 CONSENTIMIENTO INFORMADO

Institución: Universidad Señor de Sipan

Investigador: Luis Alberto Vives Garnique

Título: ESTRATEGIA DE AUTOAPRENDIZAJE DE PROGRAMACIÓN, SUSTENTADA EN UN MODELO DE GAMIFICACIÓN INTEGRAL CONTEXTUALIZADO

Yo, Víctor Tuesta Monteza, DECLARO:

Haber sido informado (a) de forma clara, precisa y suficiente sobre los fines y objetivos que busca la presente investigación **"**ESTRATEGIA DE AUTOAPRENDIZAJE DE PROGRAMACIÓN, SUSTENTADA EN UN MODELO DE GAMIFICACIÓN INTEGRAL CONTEXTUALIZADO**"**, así como en qué consiste mi participación.

Estos datos que yo otorgue serán tratados y custodiados con respeto a la intimidad, manteniendo el anonimato de la información y la protección de datos desde los principios éticos de la investigación científica. Sobre estos datos se asisten los derechos de acceso, rectificación o cancelación que podré ejercitar mediante solicitud ante el investigador responsable. Al término de la investigación, seré informado de los resultados que se obtengan.

Por lo expuesto otorgo MI CONSENTIMIENTO para que se realice la Entrevista/Encuesta que permita contribuir con los objetivos de la investigación:

Objetivo general de la investigación:

Aplicar una estrategia autoaprendizaje de la programación, sustentada en un modelo de gamificación integral contextualizado para la codificación en el desarrollo y mantenimiento de software

Objetivos específicos:

Caracterizar el proceso del autoaprendizaje de la programación y su dinámica.

Determinar las tendencias históricas del proceso de enseñanza aprendizaje de la programación y su dinámica

Diagnosticar el estado actual de la dinámica del proceso de enseñanza aprendizaje de la programación en los estudiantes de Ingeniería de Sistemasde la Universidad Señor de Sipan.

Diseñar el modelo teórico del proceso de autoaprendizaje de la programación del ingeniero de software, mediate el empleo de estrategias Construir la estrategia didáctica interactiva contextualizada del proceso de autoaprendizaje de la programación del ingeniero de software, mediate el empleo de estrategias de gamificación.

Validar los resultados de la investigación.

Chiclayo, 02 de Julio del 2001

Mg. Victor Tuesta Monteza
Decano de la Facultad de Ingeniería, Arquitectura y Urbanismo
Universidad Señor de Sipan

ANEXOS N° 6 APROBACIÓN DEL INFORME DE TESIS

El Docente:

Dr. Juan Carlos Callejas Torres

De la Asignatura:

SEMINARIO DE INVESTIGACIÓN VI: INFORME DE TESIS

APRUEBA:

El Informe de Tesis: “ESTRATEGIA DE AUTOAPRENDIZAJE DE PROGRAMACIÓN, SUSTENTADA EN UN MODELO DE GAMIFICACIÓN INTEGRAL CONTEXTUALIZADO”

Presentado por:

Luis Alberto Vives Garnique.

Chiclayo,18 de noviembre del 2021

Dr. Juan Carlos Callejas Torres

DOCENTE