MODALIDAD A.1, A.2 Y A.3 PROYECTO DE INVESTIGACIÓN CIENTÍFICA, TECNOLÓGICA O EDUCATIVA.

DOCUMENTO(S) QUE DEBE(N) INTEGRAR LA PROPUESTA DE TRABAJO.

a) Formato para la propuesta de trabajo para proyecto de investigación científica, tecnológica o educativa.

FECHA DE ELABORACIÓN | 7 (DÍA) | 06 (MES) | 2023 (AÑO)

NOMBRE DEL INSTITUTO, UNIDAD O CENTRO: Instituto Tecnológico de San Luis Potosí	
Responsable del Proyecto: Dr. Martín Guerrero Posadas Correo electrónico: martin.gp@slp.tecnm.mx	Título del Proyecto: Relación de la inteligencia fluida y los estilos de aprendizaje con el desarrollo de las competencias de programación en estudiantes de la carrera de sistemas computacionales
Tipo de investigación: Científica ( ) Tecnológica ( ) Educativa (X)	

2. DESCRIPCIÓN DEL PROYECTO

Nombre y clave del programa educativo al que impacta el proyecto: **Programa: Ingeniería en Sistemas Computacionales Clave: ISIC-2010-224**
Nombre y clave de la línea de investigación o de trabajo registrada ante el TecNM: **Docencia y Aprendizaje ITF-SLP-LIE-2016-092**

Especifique el área (marque sólo una)

Ciencias Químicas	( )	Ingeniería Industrial, Administrativa y Desarrollo Regional	( )
Ciencias Biológicas	( )	Ciencias de la Educación	( )
Ciencias de la Tierra y del Medio Ambiente	( )	Ciencias del Mar	( )
Ingeniería Eléctrica, Electrónica	( )	Ciencias Agrícolas y Forestales	( )
Ingeniería Química, Bioquímica, Alimentos, Biotecnología	( )	Ingeniería Mecánica, Mecatrónica	( )

Ciencias de los Materiales, Polímeros	( )	Ciencias de la Computación, Sistemas Computacionales, Informática	(X)

1.1. Justificación

La evidencia existente en la literatura científica presenta que los estudiantes de las carreras de sistemas computacionales presentan dificultades en el aprendizaje de las competencias en el diseño de algoritmos y aplicación del lenguaje de programación. De igual manera los docentes del área de sistemas y computación y empleadores de la zona industrial han mostrado su preocupación por la ausencia de competencias en el diseño algorítmico y los fundamentos de programación de los estudiantes y egresados de la carrera de sistemas computacionales. Por lo que esta investigación pretende determinar el efecto de la inteligencia fluida y los estilos de aprendizaje en las competencias de diseño de algoritmos y aplicación del lenguaje de programación.

Una de las carreras que ofrece el ITSLP, es la carrera de ingeniería en sistemas computacionales, con un total de 470 estudiantes. El perfil de egreso de los ingenieros en sistemas computacionales establece las competencias que debe poseer cada egresado. Las competencias relacionadas con el desarrollo de software que debe demostrar el ingeniero en sistemas computacionales son diseñar, programar y dar mantenimiento a sistemas informáticos y aplicaciones móviles. Por lo que es fundamental que posea las competencias en el diseño de algoritmos y aplicación del lenguaje de programación.

Como se planteó en los párrafos anteriores, se ha mostrado que los estudiantes y egresados de la carrera de sistemas computacionales carecen de las competencias en el diseño de algoritmos y aplicación del lenguaje de programación, por lo que esta investigación pretende determinar el efecto de la inteligencia fluida y los estilos de aprendizaje en las competencias de diseño de algoritmos y aplicación del lenguaje de programación.

El principal beneficio que se tendría con el desarrollo de esta investigación es conocer la dificultad que representa para los estudiantes el diseño de algoritmos y aplicación del lenguaje de programación tomando como base la inteligencia fluida y los estilos de aprendizaje.

Otro beneficio que se lograría con la investigación es lograr que cada uno de los estudiantes desarrolle plenamente las competencias para diseñar algoritmos y aplicar el lenguaje de programación correctamente. Lo anterior se ha establecido como una prioridad para la academia de sistemas computacionales.

Otro punto en el que abona la investigación es proporcionar a través de la literatura científica mayor evidencia sobre la complejidad en el aprendizaje de la programación, también pretende aportar información sobre sí la capacidad cognitiva es un factor determinante para el aprendizaje de la programación. Para fundamentar la falta de literatura científica relacionada entre inteligencia fluida y estilos de aprendizaje con el desarrollo de las competencias en programación se

desarrolló una búsqueda en las bases de datos de WoS (Web of Science), Scopus, y Google Scholar. Las palabras clave de búsqueda en español e inglés en las tres bases de datos fueron: aprendizaje e la progrmación, inteligencia fluida, programación, learning of programming, fluid intelligence y coding.

Después de buscar en cada una de las bases de datos se encontraron en total 4 artículos científicos diferentes relacionados con las dos variables de búsqueda. De los cuales tres se encuentran desarrollados en el nivel superior. Además, se encontró una tesis doctoral. Con estas búsquedas, se evidencia la falta de literatura sobre el tema que trata esta investigación.

El impacto que tiene la presente investigación beneficia al Tecnológico de San Luis Potosí con respecto a la línea de investigación educativa "Docencia y Aprendizaje", porque va a permitir conocer factores que influyen el proceso de aprendizaje de los estudiantes.

Los resultados de la investigación se pretenden tomar como base para establecer estrategias de enseñanza y aprendizaje para mejorar las competencias en el diseño de algoritmos y aplicación del lenguaje de programación. Con lo que se espera que los estudiantes mejoren sus competencias de egreso y puedan colocarse en empresas o departamentos relacionados con el área.

El desarrollo del proyecto no involucra costos, ya que el investigador lo desarrollará dentro del área de sistemas computacionales, por lo que es viable económicamente. Tampoco se requieren de recursos humanos adicionales, ya que el investigador realizará completamente el trabajo necesario con sus propios recursos para el logro del objetivo.

1.2. Resumen:

La evidencia existente en la literatura científica ha mostrado que los estudiantes de las áreas de sistemas computacionales presentan dificultades en el aprendizaje de las competencias en el diseño de algoritmos y aplicación del lenguaje de programación, por lo que esta investigación pretende determinar el efecto de la inteligencia fluida y los estilos de aprendizaje en las competencias de diseño de algoritmos y aplicación del lenguaje de programación.

El diseño de esta investigación será observacional, prospectivo, transversal y descriptivo. El trabajo es observacional, porque no se tendrá ninguna intervención en grupo alguno para comprobar sus efectos en los sujetos analizados. Es transversal, porque se tomará una sola medición en un momento concreto. Será descriptivo, porque analizaremos de forma descriptiva las variables seleccionadas de los instrumentos utilizados.

Para medir la inteligencia fluida se aplicará por medio de computadora el Test de Matrices Progresivas de Raven, este test fue publicado por primera vez en el año 1938 por su autor, John C. Raven. Se trata de un test no verbal, de capacidad intelectual, de habilidad mental general. El modelo que se utilizará para determinar los estilos de aprendizaje será el Modelo VARK desarrollado por el pedagogo Neil Fleming, el modelo hace referencia a las preferencias sensoriales de las personas (Rosas-

Prado et al., 2019). Por último, se diseñará, validará y aplicará un instrumento para medir competencias en el diseño de algoritmos y aplicación del lenguaje de programación.

Los resultados de la investigación se pretenden tomar como base para establecer estrategias de enseñanza y aprendizaje para mejorar las competencias en el diseño de algoritmos y aplicación del lenguaje de programación de los estudiantes de la carrera de sistemas computacionales del Tecnológico de San Luis Potosí y de ser posible, de todo el sistema del TecNM. Con lo que se espera que los estudiantes mejoren sus competencias de egreso.

1.3. Introducción:

Las computadoras y la forma de programarlas han tenido cambios muy significativos a través del tiempo. Es esencial, para desarrollar un programa de computadora, dominar las competencias del pensamiento algorítmico y aplicar correctamente el lenguaje de programación.

El pensamiento algorítmico es un conjunto de acciones, técnicas y formas mentales, donde el medio, objeto y resultado del trabajo mental son algoritmos (Kovalchuk, 2018). Byrka et al. (2021) definen el pensamiento algorítmico como un conjunto de acciones mentales y técnicas encaminadas a resolver un problema específico, como resultado de esto se crea el algoritmo correspondiente. También, consideraron que las principales características del pensamiento algorítmico son construir un modelo del proceso de resolución del problema, identificar las principales acciones necesarias para resolver el problema, organizar las acciones necesarias para solucionar el problema, correlacionar los resultados obtenidos con los esperados. Además, escribir el algoritmo, analizar el algoritmo compilado y optimizar el algoritmo propuesto. Posteriormente, ya que el algoritmo ha sido diseñado y evaluado, se traduce a un programa informático basado en un lenguaje de programación, lo que conoce como programación de computadoras.

Balanskat y Engelhardt (2015) definen la programación de computadoras como "el proceso de desarrollar e implementar varios conjuntos de instrucciones para que una computadora realice una determinada tarea, resuelva problemas y brinde interactividad humana". De manera específica, Juganaru (2014) consideró que un programa es una colección de instrucciones, que al ejecutarse efectúa actividades específicas, para desarrollarlo se requiere de un lenguaje de programación con su sintaxis, especificando las normas de codificación y una semántica que le permite plasmar sus objetivos en un entorno formal.

Planteamiento del problema

La programación no es una competencia que se desarrolla fácilmente. Se reconoce que aprender a programar es muy difícil, dado que se espera que los estudiantes tengan la comprensión abstracta correcta de un concepto y sean capaces de implementarlo de manera concreta usando estrategias apropiadas (Robins et al., 2003), lo que requiere una cantidad sustancial de experiencia práctica en programación (Yeomans et al., 2019). Figueiredo y García-Peñalvo (2020) encontraron que los programadores novatos eran menos propensos a captar la interrelación entre las partes de un programa y el programa como un todo. Además, los alumnos tienen que conocer la sintaxis, la semántica y la estructura de

un nuevo lenguaje no natural, en un tiempo corto. En este mismo sentido, la programación plantea múltiples dificultades a los alumnos que se inician en su aprendizaje: uso de variables, la comprensión de estructuras de control, la modularización de código a través de funciones, el manejo de listas o la corrección de errores sintácticos, entre otros (Bosse y Gerosa, 2017). Se ha observado en algunos cursos sobre el aprendizaje de la programación más del 60% de los alumnos suelen reprobarlos (Celepkolu y Boyer, 2018), es decir, su desempeño fue muy por debajo de las expectativas (Yeomans et al., 2019).

El aprendizaje de la programación es complejo porque se tiene una inadecuada comprensión de cómo trabaja un modelo computacional, no se domina la lectura, rastreo y escritura de código, además, existen dificultades para resolver problemas complejos relacionados con la programación y que los estudiantes carecen de habilidades para resolver problemas (Cheah, 2020). Otro de los resultados que han arrojado las investigaciones sobre el aprendizaje de la programación es que los programadores que inician piensan que la programación es difícil y desalentadora debido a su incapacidad para comprender conceptos (Gökoğlu y Kilic, 2022).

Con base en las ideas planteadas en los párrafos anteriores y para contextualizar el problema que se estudia en esta investigación, la academia de sistemas y computación, específicamente, los docentes que pertenecen a la especialidad de desarrollo de software, han observado y manifestado que los estudiantes carecen de las competencias para la programación de sistemas, ya sea de escritorio, móvil o web. De igual forma, los empleadores (directores, gerentes o jefes de tecnologías de la información) de empresas de la zona industrial de la ciudad donde se desarrolla la investigación han comunicado a la academia de sistemas su preocupación debido a que en las entrevistas de ingreso han observado que los estudiantes de la carrera de ingeniería en sistemas computacionales carecen de las competencias en el diseño de algoritmos y aplicación del lenguaje de programación, o sea el desarrollo de sistemas.

1.4. Antecedentes:

En esta sección se presentan investigaciones relacionados con el tema del aprendizaje de del pensamiento algorítmico y de la programación de computadoras, para confeccionar un estado de la cuestión que permita brindar el panorama actual del tema de la investigación. Los trabajos fueron seleccionados de las bases de datos WoS (Web of Science), Scopus, y Google Scholar.

Una de las investigaciones es la que desarrollaron Özcan et al. (2021), que tuvo como objetivo evaluar el impacto de un programa para aprender a codificar en tres habilidades cognitivas en niños: pensamiento computacional, inteligencia fluida y orientación espacial, mediante un ensayo aleatorio. La muestra fue de 174 niños de cuarto grado, fueron asignados al azar a uno de los tres programas de aprendizaje de 10 semanas: aprender a codificar (tratamiento de interés), matemáticas (otro tratamiento de comparación relacionado con STEM) y lectura (control). Los resultados mostraron que las puntuaciones de pensamiento computacional de los niños aumentaron significativamente solo en la condición de aprender a codificar. La inteligencia fluida aumentó

significativamente en todas las condiciones y la orientación espacial no mejoró en ninguna de las condiciones.

Otro ejemplo es la investigación de Wagner y Wyrich (2022), quienes realizaron un estudio en el nivel superior, para investigar la correlación de la inteligencia y la personalidad con el desempeño en la comprensión de código. En la inteligencia encontraron efectos positivos significativos pequeños a moderados en el rendimiento de comprensión de código para tres de los cuatro factores medidos, es decir, inteligencia fluida, percepción visual y velocidad cognitiva. Los investigadores hacen un llamado a la comunidad científica a desarrollar más estudios sobre estos temas.

Dirzyte et al. (2021) identificaron los rasgos de personalidad de los e-learning de programación informática, las habilidades cognitivas y los factores de motivación del aprendizaje en comparación con otros e-learning. Los investigadores aplicaron un cuestionario de factores motivadores de aprendizaje, el Big Five Inventory—2 y los instrumentos SRMCA. La muestra consistió en 444 recursos de e-learning, incluidos 189 recursos de e-learning de programación informática. Se encontró que los e-learning de programación informática demostraron puntajes significativamente más bajos de extroversión y puntajes significativamente más bajos de factores motivadores de actitud y expectativa individual, recompensa y reconocimiento, y castigo. No se encontraron diferencias significativas en las puntuaciones de las capacidades cognitivas entre los grupos. Los resultaron demostraron que la inteligencia fluida y la comprensión-conocimiento fueron predictores significativos de metas desafiantes; comprensión-conocimiento fue un predictor significativo de dirección clara; y el procesamiento visual fue un predictor significativo de la presión social y la competencia.

La investigación de Rodríguez (2021) analizó de forma experimental el impacto que tienen las actividades de programación de computadores como pueden ser la robótica educativa, la computación creativa o la programación de dispositivos móviles en escuelas e institutos sobre las funciones ejecutivas de los estudiantes. Los resultados demostraron la existencia de una relación directa entre la evolución de las funciones ejecutivas y las acciones formativas relativas a la computación creativa. De esta forma se abre la puerta tanto a la replicación del propio estudio con objetivos de evaluación de acciones formativas similares, como a la posibilidad de exploración de la aplicación de este tipo de acciones con el objetivo de mejorar los resultados cognitivos.

La enseñanza y aprendizaje del pensamiento algorítmico y de la programación de computadoras son temas que han sido ampliamente estudiados. Algunos ejemplos de investigaciones sobre el pensamiento algorítmico son: el problema de la formación del pensamiento algorítmico (Tadevosyan y Shevchuk, 2014), aspectos semánticos del pensamiento algorítmico (Kovalchuk, 2018), el pensamiento algorítmico como componente de la competencia TIC y la formación del pensamiento algorítmico en la enseñanza de la programación de juegos (Byrka et al., 2021).

De igual forma, la enseñanza y el aprendizaje de la programación de computadoras han sido tratados ampliamente: explorar la eficacia de los mensajes de error para aprender a programar (Zhou et al., 2021),

perspectivas de estudiantes y programadores profesionales sobre conceptos difíciles en programación (Yeomans et al., 2019), evaluación del pensamiento computacional para el aprendizaje de programación de computadoras en educación superior (Rojas-López y García-Peñalvo, 2020), motivación de los estudiantes hacia el aprendizaje de la programación (Latih et al., 2020) y problemas de enseñanza y aprendizaje de los fundamentos de programación (Insuasti, 2016).

1.5. Marco teórico:

En este apartado se considera el fundamento teórico de los principales temas que le dan sustento al proyecto de investigación. Además, se tomaron en cuenta las teorías que han sido base en el desarrollo de los temas, por lo que algunas de las referencias datan de hace ya más de 5 años.

Pensamiento computacional

El pensamiento computacional es considerado una habilidad y una actitud de aplicación universal para todas las personas, Wing (2017) lo definió como el proceso de pensamiento involucrado en la formulación de un problema y la expresión de su solución de tal manera que una computadora, pueda llevarla a cabo efectivamente. Zabala et al. (2016) señalaron que el aprendizaje de lenguajes de programación y la creación de algoritmos permite desarrollar el pensamiento computacional (Byrka et al., 2021). El objetivo del pensamiento computacional es desarrollar sistemáticamente las habilidades de pensamiento crítico y resolución de problemas con base en los conceptos de la computación, o sea por medio de la programación de computadoras (Astudillo, 2019).

El pensamiento computacional es un proceso cognitivo que permite la generación de soluciones a problemas a través del uso de habilidades específicas, tales como abstracción, descomposición, generalización, evaluación y diseño algorítmico (Selby, 2015). El pensamiento computacional es fundamental en la resolución de problemas, es una competencia útil para las personas independientemente de su perfil profesional, además, se considera un elemento clave para el éxito de estudiantes en las ciencias de la computación.

Pensamiento Algorítmico

La importancia del pensamiento algorítmico se basa en las necesidades de la persona para planificar asuntos, describir en detalle las acciones que se llevarán a cabo para lograr el objetivo y determinar su secuencia. El pensamiento algorítmico permite a los futuros profesionales dividir la tarea general en subtareas, planificar las etapas y el tiempo de su realización, evaluar la eficacia de las actividades, buscar, procesar y percibir nueva información (Vinichenko et al., 2018).

Para Zashchirinskaia (2020), el pensamiento algorítmico es considerado como una de las principales habilidades informáticas que los estudiantes deben lograr en su proceso de formación escolar. El pensamiento algorítmico es un conjunto de competencias que se encuentran conectadas para construir y entender algoritmos: (a) para analizar problemas dados, (b) para especificar exactamente un problema, (c) para encontrar las acciones básicas que son adecuadas para el problema dado, (d) para construir un algoritmo correctamente a un problema dado usando las

acciones básicas, (e) para pensar acerca de todos los casos normales y especiales de un problema y (f) para evaluar los algoritmos.

Mingus y Grassl (1998) determinaron que el pensamiento algorítmico es un método de pensamiento y guía de los procesos de reflexión que utiliza procedimientos paso a paso, precisa entradas y produce salidas de datos, requiere decisiones sobre la calidad y adecuación de la información que llega y de la información que sale, y controla los procesos de reflexión como medio para vigilar y dirigir el proceso de pensamiento. En esencia, el pensamiento algorítmico es simultáneamente un método de pensamiento y un medio para pensar sobre el pensamiento de uno (Khalimon et al., 2019). Una de las principales competencias que se deben de tener para el dominio del pensamiento algorítmico es saber elaborar correctamente algoritmos. Los algoritmos son definidos de forma diferente en la literatura.

Según Deitel y Deitel (2008) un algoritmo es un procedimiento para resolver un problema en términos de las acciones a ejecutar y el orden en el que se ejecutan estas acciones. La resolución de un problema se divide en el análisis del problema, diseño del algoritmo y la programación del algoritmo.

En el primer paso el problema es definido y comprendido claramente para que pueda ser analizado con todo detalle. Una vez analizado el problema se debe desarrollar el algoritmo, paso a paso. Por último, para traducir el algoritmo y que sea comprendido por una computadora se necesita de un lenguaje de programación, es decir, convertir el algoritmo en programa, ejecutarlo y comprobar que resuelve el problema que se había planteado (Eslava, 2013).

Programación de computadoras

La programación de computadoras es el proceso de desarrollar e implementar varios conjuntos de instrucciones en un lenguaje de programación para que una computadora realice una determinada tarea, resuelva problemas y proporcione interactividad (Balanskat & Engelhardt, 2015). La programación no es únicamente escribir código en un lenguaje de programación, un programa de computadora es una colección de instrucciones, que al ejecutarse efectúa actividades específicas, a través de un sistema de cómputo. Para su escritura y ejecución, necesita un lenguaje de programación que tiene una sintaxis, con la cual fija las normas de codificación y una semántica que le permite plasmar sus objetivos en un entorno formal (Juganaru, 2014).

Para poder realizar programas de computadoras, se deben tener conocimientos de lenguajes de programación, experiencia en temas relacionados con el desarrollo de lógica y algoritmos especializados, y la habilidad para analizar, comprender y resolver problemas en un proceso iterativo (Forsström & Kaufmann, 2018). Por lo tanto, los procesos involucrados en la programación son muy similares a los involucrados en la resolución de problemas, como la descomposición de problemas, la aplicación de algoritmos, la abstracción y la automatización (Shute et al., 2017).

Inteligencia Fluida

La inteligencia fluida es una habilidad para adaptarse y resolver problemas novedosos sin el uso de experiencia o conocimientos previos (Carpenter et al., 1990). La inteligencia fluida se cuenta como uno de

los predictores críticos del aprendizaje y el éxito académico (Lynn et al., 2007). La inteligencia fluida ejemplifica la habilidad para resolver problemas nuevos o profundos y se cree que tiene una base psicológica. La inteligencia fluida, que es genuina, ha sido confirmada por la capacidad de la memoria de trabajo, de centralizarse en la corteza prefrontal (Dehn, 2010).

La inteligencia fluida se refiere a la cognición deductiva, inductiva y cuantitativa con materiales y procedimientos que son novedosos para las personas que hacen la razón. Las capacidades fluidas permiten que un individuo reflexione y actúe rápidamente, descubra nuevos problemas y codifique recuerdos breves (Willis et al., 2011). La relación entre inteligencia y resolución de problemas ha sido descrita en el sentido de que los problemas particularmente misteriosos y complicados necesitan creatividad e inteligencia para ser resueltos frente a problemas claramente realizados y construidos (HoBbach, 2019).

Las medidas más reconocidas relacionadas con la inteligencia fluida involucran la prueba de inteligencia cultural justa (CFIT) y las matrices progresivas de Raven (RPM). Ambos miden las diferencias individuales en la capacidad de especificar un patrón entre una combinación de imágenes figurativas. Además, generalmente se asume que el CFIT y el de Raven son pruebas de capacidad cognitiva reducidas por la cultura, ya que se ven menos claramente afectadas por la experiencia y las oportunidades educativas (Gignac, 2018).

Estilos de Aprendizaje

El estilo de aprendizaje se define ampliamente como las preferencias inherentes de los individuos en cuanto a cómo se involucran en el proceso de aprendizaje (Ehrman y Oxford, 1990), es decir, cómo aprenden, por lo que los "rasgos cognitivos, afectivos y fisiológicos" de los estudiantes han recibido especial atención (Reid, 1987). Hasta la fecha, ha habido una proliferación de definiciones de estilo de aprendizaje propuestas para explicar las preferencias de aprendizaje de las personas, cada una de las cuales se centra en diferentes aspectos. Los esfuerzos por diseccionar el estilo de aprendizaje han sido cuestionados, y algunos destacan el proceso dinámico de la interacción del alumno con el entorno de aprendizaje y otros subrayan las formas individualizadas de procesamiento de la información.

Estos modelos de estilo de aprendizaje han proporcionado información útil para analizar los estilos de aprendizaje de los estudiantes. El estilo de aprendizaje es un factor cognitivo compuesto, afectivo y psicológico que actúa como un indicador de cómo los individuos interactúan y responden al entorno de aprendizaje (Duff, 2000). Un individuo puede tener más de un estilo de aprendizaje, lo cual representa una ventaja, porque esos individuos poseen puntos de vista más flexibles y una aceptación de su entorno de aprendizaje. Un modelo para conocer los estilos de aprendizaje de las personas, es el Modelo VARK desarrollado por el pedagogo Neil Fleming, el modelo hace referencia a las preferencias sensoriales de las personas (Rosas-Prado et al., 2019).

Las siglas VARK provienen de las siglas en inglés: Visual, Auditory, Reader/ Writer y Kinesthesic. Las preferencias sensoriales se caracterizan por lo siguiente:

Los visuales adquieren sus conocimientos a través de gráficas o ilustraciones que les permiten observar los conceptos de forma significativa. Los auditivos prefieren escuchar la información. Mientras que los lectores/escritores prefieren la información impresa y los kinestésicos reciben mejor la información por medio de la experiencia y la práctica, teniendo en cuenta la realidad de su contexto.

Otro de los modelos de estilos de aprendizaje es el modelo de Kolb, se centra en los procesos de pensamiento de los alumnos e identifica cuatro tipos de aprendizaje, a saber, divergente, asimilador, convergente y acomodativo. El modelo de indicador de tipo de Myers-Briggs clasifica a los alumnos en tipos de extroversión e introversión, y los primeros prefieren aprender de la comunicación interpersonal y los segundos se inclinan por beneficiarse de la experiencia personal.

1.6. Objetivos:

Objetivo general

Evaluar el efecto de la inteligencia fluida y los estilos de aprendizaje en las competencias de diseño de algoritmos y aplicación del lenguaje de programación.

Objetivos específicos

- Diseñar, validar y aplicar un instrumento para medir competencias en el diseño de algoritmos y aplicación del lenguaje de programación.
- Determinar la inteligencia fluida y los estilos de aprendizaje de los estudiantes de la carrera de sistemas computacionales
- Determinar el nivel de las competencias en el diseño de algoritmos y aplicación del lenguaje de programación.
- Analizar el efecto de la inteligencia fluida y los estilos de aprendizaje en las competencias de diseño de algoritmos y aplicación del lenguaje de programación.

1.7. Metas:

El objetivo general de este proyecto es evaluar el efecto de la inteligencia fluida y los estilos de aprendizaje en las competencias de diseño de algoritmos y aplicación del lenguaje de programación. Para lograr este objetivo, es necesario:

a. Diseñar, validar y aplicar un instrumento para medir competencias en el diseño de algoritmos y aplicación del lenguaje de programación.
b. Determinar la inteligencia fluida y los estilos de aprendizaje de los estudiantes de la carrera de sistemas computacionales
c. Determinar el nivel de las competencias en el diseño de algoritmos y aplicación del lenguaje de programación.
d. Analizar el efecto de la inteligencia fluida y los estilos de aprendizaje en las competencias de diseño de algoritmos y aplicación del lenguaje de programación.

Una de las metas principales del proyecto es establecer el nivel de inteligencia fluida y los estilos de aprendizaje para tomarlo como base para desarrollar estrategias de enseñanza y aprendizaje para mejorar las competencias en el diseño de algoritmos y aplicación del lenguaje de programación. Con estas estrategias se espera que los estudiantes

mejoren sus competencias de egreso y puedan colocarse en empresas o departamentos relacionados con el área.

Otro meta es presentar el reporte a la academia de sistemas computacionales para que con base en los resultados se desarrollen las estrategias de enseñanza aprendizaje para el desarrollo del diseño de algoritmos y lenguajes de programación. De igual forma, se pretende aportar información sobre sí la capacidad cognitiva es un factor determinante para el aprendizaje de la programación y proporcionar mayor evidencia sobre la complejidad en el aprendizaje de la programación.

También se establece como meta abonar al campo científico a través de la publicación de un artículo científicos en revistas arbitradas y la presentación en un congreso. Por último, se pretende tomar como base esta investigación para desarrollar otras investigaciones relacionadas con la programación orientada a objetos dentro de la misma área de sistemas computacionales.

1.8. Metodología:

Tipo de estudio

El diseño de esta investigación es observacional, prospectivo, transversal y descriptivo. El trabajo es observacional, porque no se tendrá ninguna intervención en grupo alguno para comprobar sus efectos en los sujetos analizados. Es transversal, porque se tomará una sola medición en un momento concreto. Será descriptivo, porque analizaremos de forma descriptiva las variables seleccionadas de los instrumentos utilizados.

Muestra

La población delimitada estará conformada por estudiantes de la carrera de sistemas computacionales que ya hayan cursado y aprobado la materia de fundamentos de programación, por lo que en su mayoría serán estudiantes de segundo semestre a noveno. El tipo de muestreo que se utilizará será no probabilístico por conveniencia.

Instrumentos

Test de Matrices Progresivas de Raven. Para llevar a cabo esta investigación, primero se medirá la capacidad cognitiva no verbal de los adolescentes con el Test de Matrices Progresivas de Raven (Raven et al.,1998). El test fue publicado por primera vez en el año 1938 por su autor, John C. Raven. Se trata de un test no verbal, de capacidad intelectual, de habilidad mental general.

El Test de Matrices Progresivas de Raven examina el factor G, que surge de la Teoría Ecléctica de los dos Factores, propuesta por Charles Spearman en 1904. Para esto coloca en juego procesos de educción de relaciones y correlaciones sobre un material en el que las variables no son obvias, es decir, que se deben extraer nuevas comprensiones a partir de la información dada (Raven et al., 2005). Algunas investigaciones afirman que los puntajes del test reflejan el Factor G y otros factores como la capacidad visoespacial, la motivación y las estrategias de resolución de problemas (Hayes et al., 2015), dentro de los tests de capacidad intelectual es el test con mayor saturación de Factor G (Gignac, 2015).

Este test presenta a los participantes 60 rompecabezas visuales en 5 series de 12. Cada rompecabezas está formado por una matriz (generalmente 2x2 o 3x3) que muestra el cambio a lo largo de los ejes x e y. Falta una pieza de cada matriz y debe identificarse por selección múltiple entre seis opciones (conjuntos A-B) u ocho opciones (conjuntos C-E). La dificultad aumenta progresivamente dentro de cada conjunto y de un conjunto a otro. Las puntuaciones van desde 0-60. El Test de Matrices Progresivas de Raven tiene una buena confiabilidad de prueba y re-expresión durante períodos de hasta un año en una variedad de culturas, y muy buena validez concurrente y de construcción (Raven, 2000).

Cuestionario para medir los estilos de aprendizaje. Para Castro y Guzmán (2005) el estilo de aprendizaje es una característica cognitiva que responde a la manera de aprender de los individuos, por medio del cual los estudiantes pueden mejorar y promover su aprendizaje, es decir, la manera en que los estudiantes perciben y procesan la información para construir su propio conocimiento. El estilo de aprendizaje es un factor cognitivo compuesto, afectivo y psicológico que actúa como un indicador de cómo los individuos interactúan y responden al entorno de aprendizaje (Duff, 2000). Un individuo puede tener más de un estilo de aprendizaje, lo cual representa una ventaja, porque esos individuos poseen puntos de vista más flexibles y una aceptación de su entorno de aprendizaje. Un modelo para conocer los estilos de aprendizaje de las personas, es el Modelo VARK desarrollado por el pedagogo Neil Fleming, el modelo hace referencia a las preferencias sensoriales de las personas (Rosas-Prado et al., 2019).

Las siglas VARK provienen de las siglas en inglés: Visual, Auditory, Reader/ Writer y Kinesthesic. Las preferencias sensoriales se caracterizan por lo siguiente:

- Visuales: en este estilo los estudiantes adquieren sus conocimientos a través de gráficas o ilustraciones que les permiten observar los conceptos de forma significativa.
- Auditivos: en este grupo los prefieren escuchar la información.
- Lectores/escritores: en este estilo los estudiantes prefieren la información impresa.
- Kinestésicos: en este estilo los estudiantes reciben mejor la información por medio de la experiencia y la práctica, teniendo en cuenta la realidad de su contexto.

El instrumento de medición de modelo VARK consta de 16 preguntas, clasifica a los estudiantes en cuatro grupos de acuerdo a su particular manera de aprender. Si el alumno conoce su estilo de aprendizaje, podrá reconocer las estrategias de aprendizaje que mejor se acomoden a sus preferencias y es probable que se transformen en estudiantes exitosos.

Instrumento para medir las competencias de los fundamentos programación. La materia de fundamentos de programación se imparte a los estudiantes de la carrera de sistemas computacionales. Los temas principales que cubre la materia son: diseño algorítmico, introducción a la programación, control de flujo, arreglos y modularidad. De acuerdo con el programa de la materia, el estudiante debe adquirir las competencias para dar soluciones a problemas del mundo real, es decir, aplicar

correctamente el diseño algorítmico y el lenguaje de programación para el desarrollo programas informáticos.

Para brindar la solución a un problema informático, en primer lugar, se debe identificar el problema, después analizar el problema y dividir el problema general en problemas más pequeños. De igual manera, cada problema debe ser descompuesto en problemas más pequeños hasta que cada uno pueda ser tratado como una tarea única. Por último, para cada tarea se debe diseñar un algoritmo y después codificarlo en un lenguaje de programación. Para poder realizar lo anterior, se tienen que aplicar diferentes conocimientos y competencias para la elaboración de un programa.

Por lo tanto, los conocimientos y competencias que debe desarrollar el futuro ingeniero en sistemas computacionales en la materia de fundamentos de programación para la elaboración de un programa son varias, que van desde la definición de variables, utilización de estructuras condicionales, estructuras de repetición, manejo de estructuras de datos como los arreglos y la capacidad para dividir el programa en módulos más pequeños.

Por lo anterior se propone diseñar y validar a través del análisis psicométrico un instrumento dirigido a medir la valoración de los estudiantes en torno a sus conocimientos y competencias en los temas de la materia, es decir, en diseño algorítmico, introducción a la programación, control de flujo, arreglos y modularidad.
Para el diseño del instrumento se hará una revisión en las bases de datos WoS (Web of Science), Scopus y Google Scholar para conocer a profundidad acerca de los trabajos desarrollados sobre el tema. Posteriormente, desarrollar un listado de ítems que permitan evaluar las competencias de cada tema que trata la materia. Para validar el instrumento se utilizará el juicio de expertos. Una vez que haya sido validado por los expertos, el instrumento se probará y analizará con una muestra significativa de estudiantes, lo que comúnmente conocemos como prueba piloto. Con los datos obtenidos se analizará la confiabilidad del instrumento con el coeficiente alfa de Cronbach. En caso de no tener la consistencia interna deseada, se realizarán ajustes a los ítems que no hayan tenido el alfa permitido. Otra técnica que será utilizada para verificar la validez de constructo es el análisis factorial exploratorio. El análisis factorial exploratorio es un método de agrupación de reactivos en determinadas dimensiones o factores que están altamente correlacionados entre sí y que explican las respuestas a los ítems de un test.

Análisis de resultados

Para conocer el objetivo de la investigación se identificó como variable dependiente el desempeño en el diseño algorítmico y utilización correcta del lenguaje de programación y como variables independientes la capacidad cognitiva y los estilos de aprendizaje. Por lo que, para determinar la relación existente entre la inteligencia y estilos de aprendizaje con el desarrollo de las competencias en el diseño de algoritmos y aplicación del lenguaje de programación, se utilizará el análisis de regresión lineal múltiple utilizando las variables señaladas anteriormente. El análisis se realizará en el software estadístico SPSS versión 25. Posteriormente, con base en el análisis, se realizarán la discusión y las conclusiones del estudio.

1.9. Cronograma

	MESES											
ACTIVIDAD	1	2	3	4	5	6	7	8	9	10	11	12
Diseño, validación del instrumento para medir las competencias en el diseño de algoritmos y aplicación del lenguaje de programación.												
Aplicación del instrumento para medir las competencias en el diseño de algoritmos y aplicación del lenguaje de programación.												
ENTREGA DEL REPORTE INTERMEDIO 50%, seleccione el periodo	**( ) Semestre Sabático**: 1° febrero al 30 de abril de 2023 **(X) Año Sabático**: 1° septiembre al 28 de febrero de 2023											
Análisis de resultados												
Discusión y Conclusiones												
Elaboración de reporte y artículos científicos												
ENTREGA DEL REPORTE FINAL 100%, seleccione el periodo	**( ) Semestre Sabático**: 1° febrero al 31 de julio de 2023 **(X) Año Sabático**: 1° marzo de 2023 al 31 de agosto de 2024											

PRODUCTOS ENTREGABLES Especificar cuantitativamente los productos que permitan medir el impacto del proyecto considerando su duración y lo declarado en la sección correspondiente			
Contribución en la formación de recursos humanos	**Cant.**	**Productividad académica**	**Cant.**

Incorporación de estudiantes de licenciatura al proyecto, créditos complementarios:		Artículos científicos en revistas indizadas enviados:	
Incorporación de estudiantes de licenciatura al proyecto, servicio social:		Artículos científicos en revistas arbitradas enviados:	**1**
Estudiantes con residencia concluida:		Artículos de divulgación enviados:	
Tesis en desarrollo de Licenciatura:		Artículos científicos en revistas indizadas publicados:	
Tesis concluida de Licenciatura:		Artículos científicos en revistas arbitradas publicados:	
Tesis en desarrollo de Maestría:		Artículos de divulgación publicados:	
Tesis concluida de Maestría:		Memorias en extenso en congresos:	**1**
Tesis en desarrollo de Doctorado:		Capítulos de libros enviados para revisión:	
Tesis concluida de Doctorado:		Libros enviados para revisión:	
		Libros editados y publicados:	

Productos de transferencia tecnológica	**Cant.**
Registro de Patente (IMPI):	
Registro de Modelo de Utilidad (IMPI):	
Registro de Signos Distintivos (IMPI):	
Registro de Diseño Industrial (IMPI):	
Registro de Esquemas de Trazado de Circuitos Integrados (IMPI):	
Registro de Obra (INDAUTOR):	
Carta de Usuario (Empresa):	

1.10. Vinculación con el Sector Productivo.

1.11. Carta de aceptación de la empresa o institución en hoja membretada con sello y RFC.

1.12. Referencias

Balanskat, A., y Engelhardt, K. (2015). *Computing our future. Computer programming and coding: Priorities, school curricula and initiatives across europe*. Brussels: European schoolnet. Recuperado de https://www.researchgate.net/publication/284139559_Computing_our_future_Computer_programming_and_coding_ -_Priorities_school_curricula_and_initiatives_across_Europe

Bosse, Y. y Gerosa. (2017). M.A. Why is programming so difficult to learn?: Patterns of Difficulties Related to Programming Learning Mid-Stage. *ACM SIGSOFT Software Engineering Notes*, 41(605). pp 1-6
https://doi.org/10.1145/3011286.3011301

Byrka, M.F., Sushchenko, A.V ., Svatiev, A.V . Mazin, V.M., y Veritov, O.I. (2021). A New Dimension of Learning in Higher Education: Algorithmic Thinking. *Propósitos y Representaciones,* 9(SPE2), e990.
Doi: http://dx.doi.org/10.20511/pyr2021.v9nSPE2.990

Cheah, C.S. (2020) Factors contributing to the difficulties in teaching and learning of computer programming: A literature review. *Contemporary Educational Technology*, 12(2),ep272. DOI: 10.30935/cedtech/8247.

Carpenter, P. A., Just, M. A., & Shell, P. (1990). What one intelligence test measures: A theoretical account of the processing in the raven progressive matrices test. *Psychological Review*, 97, 404– 431. https://doi.org/10.1037/0033-295X.97.3.404

Castro, S., y Guzmán, B. (2005). Los estilos de aprendizaje en la enseñanza y el aprendizaje: una propuesta para su implementación. *Revista de Investigación*, (58), pp. 83-102
https://www.redalyc.org/pdf/3761/376140372005.pdf.

Celepkolu, M., y Boyer, K. E. (2018). The importance of producing shared code through pair programming. SIGCSE 2018 - *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, 765–770. https://doi.org/10.1145/3159450.3159516

Deitel, P. J. y Deitel, H.M. (2008). *Como programar en Java*. México: Pearson Educación.

Dehn, M.J. (2010). *Long-term memory problems in children and adolescent assessment, intervention and effective instruction*. New Jersey, NJ: John Wiley & Sons.

Duff, A. (2000). Learning style of UK higher education students: Four studies of the reliability and replicability of the learning style questionnaire (LSQ). *Bristol Business School Teaching and Research Review*, 14 (3), 131-177.

Dirzyte, A., Vijaikis, A., Perminas, A., Rimasiute-Knabikiene, R., Kaminskis, L. y Zebrauskas, G. (2021). Computer Programming E-Learners' Personality Traits, Self-Reported Cognitive Abilities, and Learning Motivating Factors. *Brain Sci*. 11. Pp. 1-24. https://doi.org/10.3390/

Ehrman, M. y Oxford, R. (1990). Adult language learning styles and strategies in an intensive training set- ting. *The Modern Language Journal*, 74. Pp. 311–327. https://doi.org/10.1111/j.1540-4781.1990. tb01069.x

Eslava, V. J. (2013). Aprendiendo a programar paso a paso con C. Madrid : Bubok Publishing S.L.

Figueiredo, J. y García-Peñalvo, F. J. (2020). Increasing student motivation in computer programming with gamification. *IEEE Global Engineering Education Conference (EDUCON)*. Pp. 997-1000. doi: 10.1109/EDUCON45650.2020.9125283.

Forsström, S. E., & Kaufmann, O. T. (2018). A literature review exploring the use of programming in mathematics education. *International Journal of Learning, Teaching and Educational Research*, 17(12), 18-32. https://doi.org/10.26803/ijlter.17.12.2.

Gignac, G. E. (2015). Raven's is not a pure measure of general intelligence: Implications for g factor theory and the brief measurement of g. *Intelligence*, 52(2), 71-79. https://doi.org/10.1016/j.intell.2015.07.006

Gignac, G.E. (2018). Conceptualizing and measuring intelligence. In V. Zeigler-Hill and T.K. Shockelford (Ed.), *The SAGE handbook of personality and individual differences: The Science of Personality and individual differences* (pp. 439-464). London: SAGE.

Gökoğlu, S. y Kilic, S. (2022). Programming learning and teaching of pre-service computer science teachers: Challenges, concerns, and solutions. *E-Learning and Digital Media*, 0(0). https://doi.org/10.1177/20427530221117331

Hayes, T. R., Petrov, A. A. y Sederberg, P. B. (2015). Do we relaly become smarter when our fluid-intelligence test score improve? *Intelligence*, 48(2), pp. 1-14. doi: 10.1016/j.intell.2014.10.005

HoBbach, C. (2019). *Organizational climate for creativity: Exploring the influence of distinct type of individual differences.* Halle, Germany: Springer Gabler.

Insuasti, J. (2016). Problemas de enseñanza y aprendizaje de los fundamentos de programación. *Revista Educación Y Desarrollo Social*, 10(2), 234–246. https://doi.org/10.18359/reds.1966

Juganaru, M. (2014). *Introducción a la programación*, 1st ed. México: Patria S.A. de C.V.

Khalimon, E.A., Guseva, M.N., Kogotkova, I.Z., Brikoshina, I.S. (2019). Digitalization of the Russian economy: first results. In: *International Scientific Conference on Global Challenges and Prospects of the Modern Economic Development (GCPMED)*, pp. 199-213. Samara: Samara State Univ Econ.

Kovalchuk, M.B. (2018). Semantic aspects of algorithmic thinking. *Physical and Mathematical Education, 3*(17), 61-66. Doi:10.31110/2413-1571-2018-017-3-011

Latih, R, Peremol, A. y Jailani, N. (2020). Understanding Students' Perceptions and Motivations Towards the Learning Of Programming in Malaysian High Schools. *Journal of Theoretical and Applied Information Technology, 98*(24), pp. 4082-4091. http://www.jatit.org/volumes/Vol98No24/12Vol98No24.pdf

Lynn, R., Meisenberg, G., Mikk, J., y Williams, A. (2007). National IQs predict differences in scholastic achievement in 67 countries. *Journal of Biosocial Science, 39*, 861–874. https://doi.org/10.1017/S0021932007001964

Mingus, T. T. y Grassl, R. M. (1998). *Algorithmic and recursive thinking: Current beliefs and their implications for the future. The teaching and learning of algorithms in school mathematics.* Yearbook of the National Council of Teachers of Mathematics. Reston, VA: National Council of Teachers of Mathematics.

Özcan, M.Ş., Çetinkaya, E., Göksun, T., y Kisbu-Sakarya, Y. (2021). Does learning to code influence cognitive skills of elementary school children? Findings from a randomized experiment. *The British journal of educational psychology*, e12429. DOI:10.1111/bjep.12429

Raven, J.C., Styles, I., y Raven, M. (1998). *Ravens progressive matrices: SPM plus test booklet*. San Antonio, TX: Harcourt Assessment.

Raven J. C., Raven, J. y Court, J. H. (2005). *Test de Matrices Progresivas: Escala General*. Buenos Aires: Paidós.

Raven, J. (2000). The Ravens Progressive Matrices: Change and stability over culture and time. *Cognitive Psychology, 41*, 1-48. doi: 10.1006/cogp.1999.0735

Reid, J. M. (1987). The learning style preferences of ESL students. *TESOL Quarterly, 21*(1). Pp. 87-110. https://doi.org/10.2307/3586356

Robins, A, Rountree, J. y Rountree, N. (2003). Learning and Teaching Programming:A Reviewand Discussion. *Computer Science Education, 13*(2), 137–172. DOI:hp://dx.doi.org/10.1076/csed.13.2.137.14200

Rodríguez, M. A. (2021). *Estudio experimental sobre el impacto del uso de la computación creativa en las funciones ejecutivas*. Tesis Doctoral. Universidad de Barcelona.

Rojas-López, A., y García-Peñalvo, F. J. (2020). Evaluación del pensamiento computacional para el aprendizaje de programación de computadoras en educación superior. *Revista de Educación a Distancia (RED), 20*(63). https://doi.org/10.6018/red.409991

Rosas Prado, C. E., Zuloeta Salazar, J. F., Urbina Rosas, C. M., y Zuñe Chero, L. (2019). Relación entre los factores de la personalidad y los estilos de aprendizaje en estudiantes universitarios peruanos. UCV-HACER. *Revista de Investigación y Cultura, 8*(4), 41-55.

Selby, C. C. (2015). Relationships: computational thinking, pedagogy of programming, and Bloom's Taxonomy. *WiPSCE '15 Proceedings of the Workshop in Primary and Secondary Computing Education* (págs. 80-87). London, United Kingdom: ACM New York, NY, USA. doi:10.1145/2818314.2818315

Shute, V. J., Sun, C., y Asbell-Clarke, J. (2017). Demystifying computational thinking. Educational *Research Review, 22*, 142158. https://doi.org/10.1016/j.edurev.2017.09.003

Tadevosyan, R.G., Shevchuk, O.F. (2014). *About the problem of formation of algorithmic thinking. Scientific Notes of Vinnytsia State Pedagogical University named after Mykhailo Kotsyubynsky*. Series: Pedagogy and Psychology, 41, 93-101.

Vinichenko, M.V., Melnichuk, A.V., Makushkin, S.A. (2018). Implementation of game methods in the preparation of management personnel. In: *4th international conference on higher education advances (head'18)*, pp. 373-380. Valencia: Universitat Politécnica de Valencia.

Wagner, S. y Wyrich, M. (2022). Code Comprehension Confounders: A Study of Intelligence and Personality. *IEEE Transactions on Software Engineering, 48*(12), pp. 4789-4801, 1 doi: 10.1109/TSE.2021.3127131.

Willis, J.O., Dumont, R. y Kaufman, A.S. (2011). Factor – analytic models of intelligence. In R.J. Sternberg & S.B. Kaufman (Eds.),

The Cambridge handbook of intelligence (pp. 39-57). New York, NY: Cambridge University Press.

Wing, J. (2017). Computational Thinking's Influence on Research and Education for all. *Italian Journal on Educational Technology, 25*(2), 7-14.

Yeomans, L., Zschaler, S. y Coate, K. (2019) Transformative and troublesome? Students' and professional programmers' perspectives on difficult concepts in programming. *ACM Transactions on Computing Education, 19* (3). 23 1-27. https://doi.org/10.1145/3283071

Zabala, G., Cerrato, L. P., Blanco, S., Morán, R., y Teragni, M. (2016). Minecraft Programable: Una herramienta para aprender programación en nivel medio. *Virtualidad, Educación y Ciencia,* 7(12), 113-124.

Zashchirinskaia, O.V. (2020). Specific features of the comprehension of texts and story pictures by adolescents with intellectual disturbances. *Acta Neuropsychologica, 18(*2), 221-231.

Zhou, Z., Wang, S. y Qian, Y. (2021). Learning From Errors: Exploring the Effectiveness of Enhanced Error Messages in Learning to Program. *Front. Psychol. 12(*768962). doi: 10.3389/fpsyg.2021.768962

2. LUGAR(ES) EN DONDE SE VA A DESARROLLAR EL PROYECTO

Especifique el nombre de la Sección, Departamento, Taller o Laboratorio en que se realizará el proyecto, mencionando la dirección exacta del lugar. Si el proyecto requiere de pruebas de campo, indique: estado, región, zona y municipio, así como la distancia en Km. con respecto al Instituto, Unidad o Centro.

El proyecto se desarrollará dentro del mismo Tecnológico de San Luis Potosí, específicamente con los estudiantes de la carrera de sistemas computacionales.

3. INFRAESTRUCTURA

Mencione la infraestructura disponible en el plantel para el desarrollo del proyecto. Indique si va a hacer uso de las instalaciones en otras instituciones o dependencias.

No se requiere infraestructura adicional, solo el equipo del investigador, internet, software de trabajo (Word, Excel, SPSS, etc).

Subdirector Académico	Docente
Dra. Dubelza B. Oliva Garza Nombre y Firma	Dr. Martín Guerrero Posadas Nombre y Firma

b) Incluir reporte de software especializado en el que se muestre que no se tiene un porcentaje mayor al 30% de similitud, con el visto bueno de la Subdirección Académica.

Esta documentación deberá ser escaneada en el orden que se presenta, en un solo archivo en formato PDF, este no deberá ser mayor a 2 MB.