Universidad Inca Garcilaso de la Vega

Facultad de Ingeniería de Sistemas, Cómputo y Telecomunicaciones

ALGORITMOS IMPERATIVOS COMO INSUMO PARA GENERAR CÓDIGO DE PROGRAMACIÓN (Generative AI) APLICANDO TÉCNICAS DE "INGENIERÍA RÁPIDA (Prompt Engineering AI) EN INTELIGENCIA ARTIFICIAL (AI)"

Trabajo de Suficiencia Profesional para optar al Título de Ingeniero de Sistemas y Cómputo.

Autor

Medianero Acosta, Jorge

(https://orcid.org/0009-0004-9425-3742)

Asesor

Dr.**Hilario Falcon, Francisco** Manuel

(https://orcid.org/0000-0003-3153-9343)

Lima – Perú

2024

Trabajo Suficiencia Profesional - Jorge Medianero Acosta - 02-04-2024-COMPLETO.docx

INFORME DE ORIGINALIDAD

15%	14%	2%	4%
INDICE DE SIMILITUD	FUENTES DE INTERNET	PUBLICACIONES	TRABAJOS DEL ESTUDIANTE

FUENTES PRIMARIAS

1	Submitted to Universidad Inca Garcilaso de la Vega Trabajo del estudiante	2%
2	1library.co Fuente de Internet	1%
3	documentop.com Fuente de Internet	1%
4	intra.uigv.edu.pe Fuente de Internet	1%
5	www.ocpla.uni.edu.pe Fuente de Internet	1%
6	www.mediummultimedia.com Fuente de Internet	<1%
7	repositorioacademico.upc.edu.pe Fuente de Internet	<1%
8	www.dykinson.com Fuente de Internet	<1%

DEDICATORIA

Quiero dedicar este proyecto tecnológico a mi esposa Margarita, amados hijos Daniel, Raquel y querido nieto Nicolas, quienes dan sentido a mi vida y son la fuente de inspiración de nuevos retos personales y superación académica. También, "*Quiero honrar a mi madre política, Irene, quien en vida se desempeñó como docente y nos brindó un cariño incondicional, transmitiéndonos valores positivos en todo momento. Quiero con esto reconocer su gran labor y legado en nuestra familia.*"

Espero retribuir y afirmar el amor por ellos en cada logro académico.

AGRADECIMIENTOS

Un reconocimiento a la Universidad, a los integrantes de la promoción 2014-2019 y a nuestro asesor Francisco Hilario, por impulsar la realización de esta Tesis de investigación.

RESUMEN

En el panorama actual de desarrollo de software, enfrentamos el desafío de agilizar y optimizar el proceso de codificación en el ámbito del modelo imperativo. La incorporación de la Ingeniería Rápida (Prompt Engineering) de la Inteligencia Artificial (AI) emerge como una solución revolucionaria para la autogeneración de código de programación (Generative AI) con la intervención de los Modelos LLM (Large Language Models), permitiendo una mayor eficiencia en el desarrollo, mejora en la formación efectiva de profesionales en Ingeniería de software y, en última instancia, una reducción de tiempo, recursos y costos de producción. Al integrar algoritmos imperativos como fuente e input sintácticamente escritos, estamos allanando el camino hacia un futuro escenario de productividad, donde la ingeniería rápida (Prompt Engineering) respaldada por la inteligencia artificial (AI) redefine la forma en que construimos software profesional implementado en cualquiera de los lenguajes de programación imperativos vigentes.

Palabras clave:

Modelo Imperativo, Generative AI (GenAI), Ingenieria Rápida (Prompt Engineering AI), Inteligencia Artificial (AI), Modelo LLM (Large Language Models), Algoritmo Imperativo.

ABSTRACT and KEYWORDS

In today's software development landscape, we face the challenge of streamlining and optimizing the coding process within the scope of the imperative model. The incorporation of Prompt Engineering of Artificial Intelligence (AI) emerges as a revolutionary solution for the self-generation of programming code (Generative AI) with the intervention of LLM (Large Language Models), enabling greater efficiency in development, improvement in the effective training of software engineering professionals and, ultimately, a reduction in time, resources and production costs. By integrating imperative algorithms as syntactically typed source and input, we are paving the way to a future productivity scenario, where Prompt Engineering supported by Artificial Intelligence (AI) redefines the way we build professional software implemented in any of the current imperative programming languages.

Keywords:

Imperative Model, Generative AI (GenAI), Prompt Engineering AI, Artificial Intelligence (AI), Large Language Models (LLM), Imperative Algorithm.

ÍNDICE GENERAL

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

En el dinámico paisaje del desarrollo de software contemporáneo, nos encontramos ante un desafío crucial: la agilización y optimización del proceso de codificación dentro del ámbito del modelo imperativo. En respuesta a esta necesidad imperiosa, emerge la Ingeniería Rápida respaldada por la Inteligencia Artificial (IA), como una solución innovadora para la autogeneración de código de programación mediante el uso de Modelos LLM (Large Language Models). Este enfoque revolucionario no solo promete una mayor eficiencia en el desarrollo de software, sino que también ofrece mejoras sustanciales en la formación de profesionales en Ingeniería de Software y, en última instancia, una reducción significativa en los tiempos, recursos y costos de producción.

Al integrar algoritmos imperativos como fuente e input sintácticamente escritos, estamos pavimentando el camino hacia un futuro prometedor en términos de productividad. La combinación de Ingeniería Rápida y la inteligencia artificial redefine fundamentalmente la manera en que concebimos, desarrollamos y desplegamos software profesional en cualquier lenguaje de programación imperativo actualmente vigente.

Este avance tecnológico representa un hito trascendental en la evolución del desarrollo de software, ofreciendo no solo una aceleración en los ciclos de desarrollo, sino también una mayor capacidad para abordar proyectos complejos con mayor eficacia y precisión. En última instancia, la fusión entre la ingeniería rápida y la IA no solo optimiza el proceso de codificación, sino que también promete transformar la industria del software en su totalidad, impulsando la innovación y la eficiencia a nuevas alturas.

CAPÍTULO I: ASPECTOS GENERALES

1.1. DESCRIPCIÓN DE LA EMPRESA O INSTITUCIÓN.

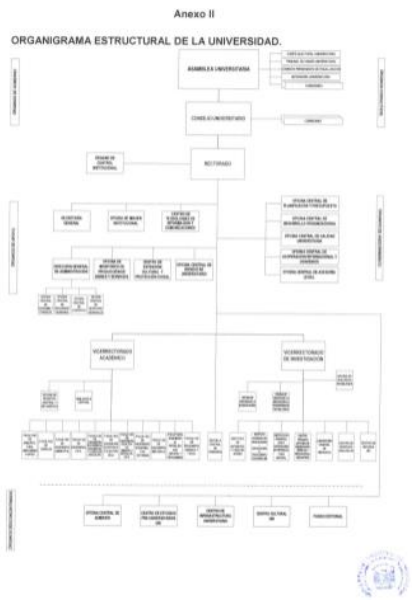
Universidad Nacional de Ingeniería

Institución: Es una universidad pública peruana ubicada en la ciudad de Lima.

Dirección: Av. Túpac Amaru 210, Rímac 15333. Lima-Perú.

Rector: Pablo Alfonso López-Chau Nava PhD

1.2. DESCRIPCIÓN DEL PRODUCTO O SERVICIO.



Universidad Nacional de Ingeniería

Información institucional: ¿Qué hacemos?

Entrenar individuos destacados en áreas como ciencias, ingeniería y arquitectura con un enfoque humanitario y un compromiso firme con la investigación científica, así como la innovación y desarrollo tecnológico. Este compromiso incluye una dedicación constante a mejorar la calidad educativa y a fomentar la responsabilidad social, todo ello con el propósito de contribuir al progreso sostenible de la nación.

La Universidad Nacional de Ingeniería (UNI) es una entidad adscrita al Ministerio de Educación.

Figura 1. Organigrama de Estructura Organizacional

URL: [Universidad Nacional de Ingeniería - Organigrama de Estructura Organizacional]

1.3. UBICACIÓN GEOGRÁFICA Y CONTEXTO SOCIOECONÓMICO

Contexto socioeconómico.

En el Plan Estratégico Institucional PEI UNI 2020 – 2025 Ampliado, se expone que:

> La UNI, fundada en 1876 por el ingeniero polaco Eduardo de Habich bajo el nombre de Escuela Especial de Construcciones Civiles y de Minas del Perú, más tarde conocida como Escuela de Ingenieros, se convirtió en Universidad Nacional de Ingeniería en 1955. Hoy en día, es la principal institución educativa para la formación de ingenieros, arquitectos y científicos en el país (Abril 2022, pág. 7).

Figura 2. Panorámica del frente principal de la UNI. (Extraído de: Google Imágenes)

Sigla: UNI

Fundación: 18 de marzo de 1876

Fundador: Eduardo de Habich

Colores académicos: Granate Crema

Campus: Distrito del Rímac (66 ha)

Figura 3. Ubicación Geográfica. (Extraído de: Google Maps)

Coordenadas:

12°01′11″S 77°02′55″O / (-12.01972222, -77.04861111)

URL: https://www.google.com/maps/place/Universidad+Nacional+de+Ingenier%C3%ADa/@-12.0213263,-77.0519572,15.31z/data=!4m14!1m7!3m6!1s0x9105cf230af7b5e7:0x43ae74b7b59a27a0!2sUniversidad+Nacional+de+Ingenier%C3%ADa!8m2!3d-12.0238022!4d-77.0484059!16zL20vMDY3dHFo!3m5!1s0x9105cf230af7b5e7:0x43ae74b7b59a27a0!8m2!3d-12.0238022!4d-77.0484059!16zL20vMDY3dHFo?hl=es-419&entry=ttu)

1.4. ACTIVIDAD GENERAL O ÁREA DE DESEMPEÑO

DECLARACIÓN DE POLÍTICA INSTITUCIONAL

En el Plan Estratégico Institucional PEI UNI 2020 – 2025 Ampliado, se declara.

> El Estatuto de la Universidad Nacional de Ingeniería (UNI), creado en concordancia con la Ley Universitaria N° 30220, es el documento que regula la vida institucional para el logro de sus objetivos. En este sentido, la UNI promueve la mejora continua de la calidad de sus programas educativos y servicios, implementando un proceso de acreditación obligatorio y constante. Asimismo, la investigación es considerada una función fundamental y obligatoria de la Universidad, la cual promueve y realiza, contribuyendo al avance del conocimiento y el desarrollo en consonancia con los avances científicos, tecnológicos e innovadores, orientados a las necesidades de la sociedad (abril 2022, pág. 6).

La UNI, según su Estatuto PEI UNI 2020 – 2025 Ampliado, declara que se rige por los siguientes principios:

> Búsqueda y difusión de la verdad;
> Calidad académica; Autonomía;
> Libertad de cátedra, libertad de asistencia del estudiante y libertad de pensamiento;
> Espíritu crítico y de investigación;
> Democracia y cogobierno institucional; Meritocracia;
> Pluralismo, tolerancia, diálogo intercultural e inclusión;
> Pertinencia y compromiso con el desarrollo del país;
> Afirmación de la vida y dignidad humana;
> Mejoramiento continuo de la calidad académica; Creatividad e innovación; Internacionalización;
> Pertinencia de la enseñanza e investigación con la realidad social;
> Ética pública y profesional (abril 2022, pág. 6).

1.5. MISIÓN Y VISIÓN

1.5.1. MISIÓN. [Plan Estratégico Institucional 2020 - 2025 Ampliado - Aprobado con Resolución Rectoral N° 1201-2022 del 21-06-2022]

Capacitar a individuos destacados en áreas como ciencias, ingeniería y arquitectura, enfatizando una perspectiva humanista y otorgando importancia primordial a la investigación científica, así como a la innovación y al avance tecnológico. Nos comprometemos con la mejora continua de la excelencia y la responsabilidad social, contribuyendo al desarrollo sostenible de la nación (PEI UNI 2020 – 2025, pág. 7).

1.5.2. VISIÓN

Convertirse en una institución educativa con programas académicos altamente reputados, reconocidos a nivel internacional por su destacada labor en investigación científica y progreso tecnológico. Nos dedicamos a promover el espíritu emprendedor y a cultivar un talento humano competitivo tanto en el sector público como en el privado, manteniendo un compromiso sólido con la responsabilidad social, el desarrollo sostenible y el bienestar nacional (PEI UNI 2020 – 2025).

LEMA: "UNI, Ciencia y Tecnología al servicio del País"

NUESTROS VALORES INSTITUCIONALES.

La comunidad universitaria promueve los siguientes valores:

Excelencia: Práctica continua de alta calidad y mejora de la gestión del conocimiento, orientada a la formación integral de líderes profesionales.

Compromiso: Una firme dedicación a los valores, la tradición, la cultura y la educación en ciencia y tecnología, con el propósito de satisfacer las demandas de la sociedad y la nación.

Meritocracia: Promueve la integridad de capacidades y competencias, y el reconocimiento del talento a través de evaluación continua y resultados tangibles con transparencia.

Innovación: Impulsa la innovación tecnológica creativa en todos sus niveles con la firme consigna de contribuir al desarrollo tecnológico, social y económico del país.

CAPÍTULO II: DESCRIPCIÓN GENERAL DE LA EXPERIENCIA

2.1. ACTIVIDAD PROFESIONAL DESARROLLADA.

- Docente Facultad de Ingeniería Industrial y Sistemas. UNI-FIIS 2016
- Instructor Convenio UNI-CITEN "Instituto Tecnológico Naval" Marina de Guerra del Perú. ITN-MGP / UNI-FIIS 2018
- Docente Unidad de Postgrado y Extensión - Sistemasuni (2024 -2025).

2.2. PROPÓSITO DEL PUESTO Y FUNCIONES ASIGNADAS.

Docente del área Académica de Tecnología

Asignaturas:

- Modelamiento de Datos y Desarrollo de Software
- Implementación de Lenguajes de Programación
- Administración de Servicios de Datos Relacionales DBA

2.3. APLICACIÓN DE LA TEORÍA EN LA PRÁCTICA DEL DESEMPEÑO PROFESIONAL

Desempeño profesional periodo 2016 - 2024

1. Investigación continua en Metodología de la Programación.

   Principios y Teoremas de los Modelos de Programación
   - Modelo Imperativo
   - Modelo Funcional
   - Modelo Lógico
2. Investigación continua en Técnicas de Modelado Orientada a Objetos

   Estándar Industrial UML / OMG:
   - Modelado de Software de Aplicación.
   - Modelado de Software de Persistencia.

   Proceso de Desarrollo de Software UP / OMG
   - Modelo BPN
   - Modelo Use Case
   - Modelo Dominio
   - Modelo Diseño
   - Modelo de Implementación
   - Modelo de Componentes
   - Modelos de Despliegue
   - Modelo de Pruebas

CAPÍTULO III: FUNDAMENTACIÓN DEL TEMA ELEGIDO.

3.1. DESCRIPCIÓN DE LA PROBLEMÁTICA.

La inclusión de los LLM en la construcción de software puede llevar a una mejora general en la calidad, eficiencia y usabilidad del software, lo que a su vez lleva a una mayor satisfacción del cliente y un mejor rendimiento comercial. Los LLM pueden ser especialmente útiles en aplicaciones de AI como el procesamiento de lenguaje natural, el razonamiento automatizado y la toma de decisiones (Brown, T. B. et.al., 2020).

3.1.1. Los Modelos de Lenguaje y Lógica (LLM) son una representación formal de conocimientos y reglas que pueden ser utilizados para resolver problemas y tomar decisiones en el ámbito de la Inteligencia Artificial (AI). Algunas de las principales características de los LLM incluyen:

a) Lógica formal: Los LLM utilizan la lógica formal para representar y razonar sobre los conocimientos y comprensión de las reglas del dominio del problema, lo que garantiza una consistencia clara y precisión en el razonamiento.

b) Automatización de tareas: Los LLM pueden automatizar tareas repetitivas y tediosas en el proceso de desarrollo de software, lo que aumenta la eficiencia y productividad.

c) Mejora de la usabilidad: Los LLM pueden ayudar a crear interfaces de usuario más intuitivas y fáciles de usar, lo que mejora la experiencia del usuario final.

d) Reducción de costos: Al automatizar tareas y mejorar la eficiencia, los LLM pueden ayudar a reducir los costos de construcción y mantenimiento de software.

e) Flexibilidad: Los LLM son flexibles y pueden ser utilizados en una variedad de aplicaciones y dominios, lo que los hace una herramienta útil en una amplia gama de escenarios de la vida real.

En suma, los modelos LLM son una representación formal y precisa de conocimientos y reglas de dominio, que pueden ser utilizados para resolver problemas y tomar decisiones en el ámbito de la AI. Los LLM utilizan la lógica formal para representar y razonar sobre los conocimientos y reglas del dominio del problema, y pueden automatizar tareas y mejorar la eficiencia y usabilidad del software.

Al aplicar herramientas GenAI basados en los Modelos LLM, y específicamente técnicas de Ingeniería Rápida (Prompt Engineering AI) sobre algoritmos del modelo imperativo permite automatizar tareas y elevar la productividad de los profesionales de la Ingeniería de software.

3.1.2. Implementación de AI en la construcción de software.

Objetivos generales.

El título propuesto aborda la integración de algoritmos del modelo imperativo, la autogeneración de código de lenguaje de programación y la aplicación de la ingeniería rápida en el ámbito de la inteligencia artificial. En la educación universitaria y en especial en la formación de ingenieros de software, implica la incorporación de conceptos avanzados que podrían transformar la forma en que se enseña y se desarrolla software. La relevancia radica en la preparación de los estudiantes para adoptar enfoques más eficientes y avanzados, alineados con las tendencias actuales en inteligencia artificial, lo que podría impactar positivamente en la productividad y la calidad de la construcción de software (Nguyen-Duc. et.al., 2023).

Escribir código es sin duda un proceso arduo y repetitivo, consideramos que si la inteligencia artificial se hace cargo de esta tarea y los desarrolladores de código quedarían relevados para realizar otras tareas mucho más productivas en la Ingeniería de software. A decir:

a) Diseño arquitectónico:

   Los desarrolladores pueden dedicar más tiempo a la planificación y diseño de la arquitectura de software. Esto implica tomar decisiones fundamentales sobre la estructura del sistema, la elección de tecnologías y la definición de patrones de diseño.

b) Optimización de rendimiento:

   Los desarrolladores pueden concentrarse en mejorar el rendimiento del software, identificando y solucionando cuellos de botella, ajustando algoritmos y optimizando el código existente.

c) Resolución de problemas complejos:

   Los profesionales se pueden enfocar más en la resolución de problemas complejos y desafíos únicos que requieren un pensamiento creativo y estratégico. Esto incluye abordar problemas de lógica, algoritmos avanzados y situaciones no triviales.

d) Colaboración y comunicación:

   Los desarrolladores pueden dedicar más tiempo a la colaboración con otros miembros del equipo, participando en discusiones de diseño, revisiones de código y facilitando una comunicación más efectiva dentro del equipo de desarrollo.

e) Mejora continua:

   Los ingenieros de software se pueden enfocar en la mejora continua de los procesos de desarrollo de software, adoptando prácticas ágiles, incorporando retroalimentación de los usuarios y aplicando metodologías que impulsen la calidad del software.

Objetivos específicos.

Es importante destacar que, aunque la inteligencia artificial puede asistir en la generación de código, la participación humana sigue siendo esencial para la toma de decisiones estratégicas, la comprensión del contexto empresarial y la garantía de la calidad del software. Además, los desarrolladores pueden encontrar nuevas oportunidades para expandir sus habilidades y contribuir de manera más significativa al proceso de desarrollo de software en su conjunto.

¿Cuál sería entonces el impacto del uso de la Ingeniería rápida (Prompt Engineering) en la formación universitaria de Ingenieros de Software?

Ventajas:

a) Adopción de Tecnologías Avanzadas: La introducción de la Ingeniería Rápida en la formación permitirá a los estudiantes familiarizarse con tecnologías de vanguardia y técnicas avanzadas de inteligencia artificial.

b) Eficiencia en el Desarrollo: Los ingenieros formados en Ingeniería Rápida pueden ser más eficientes en el desarrollo de software, ya que aprenderán a aprovechar algoritmos de generación de código para tareas rutinarias.

c) Enfoque en Problemas Complejos: Los estudiantes podrán dedicar más tiempo a abordar problemas complejos y desafíos estratégicos en lugar de centrarse exclusivamente en la implementación detallada del código.

d) Preparación para el Futuro: La formación en estas metodologías prepara a los ingenieros de software para un entorno laboral futuro en el que la inteligencia artificial desempeñará un papel más destacado en el desarrollo de software.

e) Mejora de la productividad: La automatización de ciertas tareas puede aumentar la productividad general, permitiendo a los ingenieros concentrarse en aspectos más críticos y creativos del desarrollo.

Desventajas:

a) Pérdida de Conocimientos Fundamentales: Dependiendo del grado de automatización, existe el riesgo de que los ingenieros pierdan la comprensión profunda de los fundamentos de programación y algoritmos, ya que la generación de código podría ocultar estos detalles.

b) Dependencia de Herramientas Específicas: Si la formación se centra demasiado en herramientas específicas de Ingeniería Rápida, los ingenieros pueden volverse dependientes de esas herramientas y carecer de habilidades transferibles.

c) Limitación en la Creatividad: La automatización puede limitar la creatividad en el proceso de desarrollo, ya que las soluciones generadas automáticamente pueden carecer de la originalidad y el pensamiento innovador de los desarrolladores humanos.

3.2. TEORÍA SOBRE LA PROBLEMÁTICA.

3.2.1. Descripción de los Algoritmos del Modelo Imperativo:

Los Algoritmos del Modelo Imperativo son una serie de instrucciones específicas y ordenadas que se ejecutan en un entorno computacional, con el objetivo de resolver un problema o lograr un resultado deseado. A diferencia de los modelos declarativos, en el modelo imperativo se especifican las normativas y reglas sintácticas a seguir de manera explícita.

a) Concepto: Se refiere a los procedimientos o secuencias de instrucciones detalladas que representan el modelo imperativo en la programación. Este modelo se centra en indicar "cómo" realizar una tarea, definiendo explícitamente los pasos a seguir.

b) Relevancia en la Educación Universitaria: Los algoritmos imperativos son fundamentales en la formación de ingenieros de software, ya que proporcionan una base sólida para el diseño y la implementación de soluciones computacionales.

c) Desde el punto de vista de la Teoría de los Lenguajes de programación, los algoritmos imperativos suelen utilizar una variedad de métodos para lograr su objetivo. Estos métodos incluyen:

- Método Estructurado (Bohm y Jacopini - 1960)
- Método Procedimental o Modular (Edsger W. Dijkstra - 1960 y Niklaus Wirth - 1988)
- Método Orientado a Objetos (Ole-Johan Dahl y Kristen Nygaard -1962)

Estos métodos permiten a los algoritmos imperativos realizar tareas complejas y flexibles. Los algoritmos imperativos bajo estas reglas y normas metodológicas permiten el diseño de soluciones informáticas que cumplen con el principio de "Bajo Acoplamiento y Alta Cohesión".

3.2.2. Análisis teórico de los Algoritmos:

> Según Donald Knuth, el análisis teórico de algoritmos se divide en dos tipos: Tipo A. Analizar un algoritmo particular desde lo cuantitativo. Este punto de vista, es útil para ver qué tan bueno es en el uso de recursos de ejecución.
>
> Por ejemplo, es posible predecir el tiempo de ejecución para clasificar (sort) de varios algoritmos.
>
> Tipo B. Complejidad: Realiza un estudio de clases de algoritmos, ante una tarea particular, buscando la mejor forma posible de hacer un proceso (s).
>
> Por ejemplo, bajo ciertos supuestos se puede examinar el problema de clasificación (sort) y tomar decisiones sobre el mejor. Analizando la complejidad se crearon los mejores algoritmos de ordenamiento posibles (1970, pág. 51).

La definición de Algoritmos del Modelo Imperativo es una descripción general del concepto y no está asociada específicamente con ningún científico computacional en particular. Sin embargo, muchos científicos de la computación han contribuido al desarrollo y estudio de algoritmos en el contexto del modelo imperativo, incluyendo a Donald Knuth (The analysis of algorithms, 1969), Alan Turing (The Machine of Tuting, 1995), y Edsger Dijkstra (The humble programmer, 1972), entre otros.

3.2.3. Autogenerar Código de Lenguaje de Programación: Generative AI (GenAI).

Un modelo de IA generativa es un tipo de arquitectura de aprendizaje automático que utiliza algoritmos de IA para crear instancias de datos novedosas, basándose en los patrones y relaciones observados en los datos de entrenamiento (Gerhard et.al., 2022, July).

a) Concepto: Implica la capacidad de crear automáticamente código fuente de programas informáticos sin intervención humana directa. En este contexto, se busca utilizar algoritmos computacionales para generar código de lenguajes de programación.

b) Relevancia en la Educación Universitaria: Este concepto plantea nuevas posibilidades y desafíos en la enseñanza de la ingeniería de software al introducir herramientas y técnicas avanzadas de AI, que podrán transformar la forma en que se desarrollan aplicaciones.

c) Algoritmos Imperativos como suministro en modelos GenAI.

El investigador Gutiérrez-Jiménez, D. A., indica que:

> "Un modelo de IA generativo es de naturaleza críticamente central pero incompleta, ya que requiere un mayor ajuste de tareas específicas a través de sistemas y aplicaciones".
>
> Según esto, al suministrar algoritmos semántica y sintácticamente correctos ajustaremos las tareas específicas resolviendo preguntas acotadas para generar respuestas coherentes y confiables.
>
> Los algoritmos imperativos son estructuras pre construidas en base a Métodos, patrones y técnicas de una de las disciplinas de la ciencia Computacional: Ciencia de los Algoritmos (2024. pág. 4. párr. 2.a.).

3.2.4. Los Modelos LLM (Large Language Models) de la AI en la Ingeniería de Software:

a) Concepto: Los Modelos de Lenguaje y Lógica (LLM, Language and Logic Models) son una parte importante de la Ingeniería de Software, ya que pueden mejorar la precisión y eficiencia de los sistemas de software.

b) Relevancia en la formación de Ingenieros de Software: Al incorporar los conocimientos y técnicas de la Inteligencia Artificial (AI) en el proceso de construcción de software, se pueden lograr los siguientes beneficios:

- Mejora de la precisión: Los LLM pueden mejorar la precisión de los sistemas de software al proporcionar una representación formal y precisa de los conocimientos y reglas del dominio del problema.
- Mejora de la calidad del software: Los LLM pueden ayudar a identificar y corregir errores y fallas en el software, lo que lleva a un producto final más confiable y robusto.
- Automatización de tareas: Los LLM pueden automatizar tareas repetitivas y tediosas en el proceso de desarrollo de software, lo que aumenta la eficiencia y productividad.
- Mejora de la usabilidad: Los LLM pueden ayudar a crear interfaces de usuario más intuitivas y fáciles de usar, lo que mejora la experiencia del usuario final.
- Reducción de costos: Al automatizar tareas y mejorar la eficiencia, los LLM pueden ayudar a reducir los costos de construcción y actualización de software.

En general, la inclusión de los LLM en la Ingeniería de Software puede llevar a una mejora general en la calidad, eficiencia y usabilidad del software, lo que a su vez lleva a una mayor satisfacción del cliente y un mejor rendimiento comercial.

Los Modelos de Lenguaje y Lógica (LLM) pueden ser utilizados para resolver problemas y tomar decisiones en el ámbito de la Inteligencia Artificial (AI). Los LLM se basan en la representación de conocimientos y la lógica formal para representar y razonar sobre diferentes aspectos del mundo real.

Ingeniería Rápida (Prompt Engineering AI) de Inteligencia Artificial (AI):

a) Concepto: Se refiere a la aplicación de técnicas de inteligencia artificial, específicamente la ingeniería rápida, para acelerar y mejorar el proceso de construcción de software. El término "prompt engineering" sugiere la manipulación estratégica de estímulos para guiar el comportamiento de la inteligencia artificial.

b) Casos de uso en asistencia automática: La Técnica de la Ingeniería Rápida (Prompt Engineering AI) en Inteligencia Artificial se refiere al proceso de crear y optimizar preguntas o instrucciones específicas para obtener las respuestas deseadas de modelos de lenguaje. Esto incluye el uso de palabras clave, frases de activación y estructuras sintácticas adecuadas. El objetivo es lograr una comunicación efectiva y precisa entre humanos y sistemas de Inteligencia Artificial (Caines. et.al. 2023).

3.2.5. La ingeniería rápida (Prompt Engineering AI), es la clave de acceso al mundo de la IA.

La ingeniería del prompt facilitará la confiabilidad y la accesibilidad de los generadores de código automatizados (por ejemplo, ChatBots Generative AI).

En COMPUTERWEEKLE.ES Preinfalk Alejandro (2023), argumenta que:

> Interactuar con la inteligencia artificial generativa es similar a conversar con alguien que tiene un amplio conocimiento pero que a veces carece de discernimiento; es esencial comprender cómo y por qué responde de cierta manera para obtener el resultado deseado. Por lo tanto, el concepto de "prompt engineering" será fundamental en el futuro, ya que ampliará las capacidades de la inteligencia artificial al generar respuestas coherentes y consistentes que serán aplicables tanto en entornos laborales como sociales.
>
> Este mismo principio se aplicará en el ámbito industrial, donde la inteligencia artificial ha revolucionado los procesos de trabajo en las plantas de producción y ha establecido nuevos estándares de competitividad en el mercado. Por ejemplo, ha permitido a las instalaciones de fabricación expandir sus modelos de producción sin comprometer la calidad de los procesos, mediante la implementación de herramientas como el aprendizaje automático, que posibilita la adopción de sistemas de mantenimiento predictivo o la creación de algoritmos que sustentan el concepto de "Smart Manufacturing". A través de este enfoque, se analizan los datos de producción y se ajustan los procesos de manera automática para adaptarse a los cambios (Pág. 1, párr. 3, 5).

En el post de Siemens Digital Industries Software PR Team, se anunció que:

> Durante la Hannover Messe, la feria industrial más grande del mundo, Siemens y Microsoft revelaron una colaboración destinada a facilitar a los desarrolladores de software e ingenieros de automatización la creación más rápida de código para controladores lógicos programables (PLC), dispositivos computacionales industriales que supervisan la mayoría de las operaciones en fábricas a nivel global. En este contexto, se presentó cómo los equipos de ingeniería pueden reducir de manera significativa el tiempo y los posibles errores al generar código PLC mediante la utilización de entradas en lenguaje natural, como las proporcionadas por ChatGPT (2023, Pág. 1, párr. 1).

Relevancia en la formación y Educación Universitaria:

a) Relevancia en la Educación Universitaria: Introducir estas técnicas en la formación de ingenieros de software puede preparar a los estudiantes para adoptar enfoques más eficientes y avanzados en el desarrollo de software, alineados con las tendencias actuales en inteligencia artificial.

b) La introducción de la Ingeniería Rápida en la formación universitaria de Ingenieros de Software ofrece varias ventajas, como la adopción de tecnologías avanzadas, eficiencia en el desarrollo y preparación para el futuro. Sin embargo, es esencial abordar desafíos potenciales, como la posible pérdida de conocimientos fundamentales y la dependencia excesiva de herramientas específicas. En general, el impacto dependerá de cómo se equilibre la automatización con la necesidad de mantener una comprensión profunda de los principios de la ingeniería de software.

En la era de la expansión de la inteligencia artificial (AI), con cada vez más recursos educativos disponibles en internet, surgen modos muy originales de adquirir conocimiento, generando también patrones más flexibles y sistemas o modelos multivariantes en la enseñanza-aprendizaje.

Se producirán definitivamente drásticos cambios en la educación y formación de profesionales, La IA, solventará la adquisición inmediata de conocimientos concretos definidos y contextualizados. La IA impactará definitivamente la educación, mejorará las capacidades, habilidades, competencia y productividad de los estudiantes en el futuro inmediato en todos sus niveles (Wang, et.al., 2018).

3.3. ANÁLISIS DE LA PROBLEMÁTICA.

3.3.1. Construcción de código con Generative AI (GenAI).

La problemática planteada implica la generación automática (GenAI: Generative Artificial Inteligente) de código de lenguajes de programación a partir de algoritmos del modelo imperativo. Esto conlleva desafíos en la traducción precisa de conceptos algorítmicos a código ejecutable, para asegurar esto último es necesario implementar una "Especificación de Algoritmos y Notación de Tipos" basada en las tres metodologías vigentes del Paradigma Imperativo, a saber: Programación Estructurada, Programación Procedimental y Programación Orientada a Objetos. La complejidad de la programación imperativa y la demanda creciente de software profesional hacen esencial explorar soluciones innovadoras como la Inteligencia Artificial (AI: Artificial Intelligence) como herramienta de construcción de aplicaciones en la Ingeniería de Software (ES: Engineering Software).

3.3.2. Análisis y Propuesta de solución: Aplicar técnicas de Ingeniería Rápida (Prompt Engineering) de la Inteligencia Artificial (AI) en la construcción de software.

Muchos expertos e Ingenieros de Software, sostienen que:

> La técnica de Prompt Engineering de la Inteligencia Artificial (Prompt Engineering AI), se revela como un catalizador crucial en este contexto. Permite entrenar modelos de IA específicamente para comprender y generar código a partir de instrucciones en lenguaje natural. Integrar algoritmos imperativos como suministro para autogenerar código optimiza la eficiencia y precisión del proceso de desarrollo (Ortolan, P., 2023, pág. 33).

Varios profesionales de la ciencia computacional, indican que:

> La inteligencia artificial multimodal (IA multimodal) es un enfoque de inteligencia artificial que analiza y fusiona datos provenientes de distintas fuentes o modalidades, tales como texto escrito, así como medios visuales y auditivos, como imágenes, audio y video.
>
> Este tipo de inteligencia artificial se fundamenta y se inspira en la manera en que los seres humanos utilizan múltiples sentidos, como la vista, el oído, el tacto, el olfato y el gusto, para percibir y relacionarse con el entorno. En consecuencia, la IA multimodal ofrece una vía más natural y comprensible para interactuar con el mundo tecnológico y artificial (Arrendajo, 2023, párr. 6).

Los investigadores de Modelos de Lenguaje AI, definen los LLM como:

> Los Modelos LLM (Large Language Models) o Modelos de lenguaje basados en la Lingüística (estudio del lenguaje) en especial de la morfología, sintaxis (estructura de sentencias) y semántica computacional tienen la capacidad de comprender y crear texto similar al de los humanos. El Modelo LLM, es un programa informático inteligente que aprende de una amplia colección de texto escrito proveniente de una fuente inagotable como Internet. (Gerhard, 2022. pp. 206-215).

3.3.3. Análisis y Objetivos de Productividad: Reentrenamiento en la formación de profesionales de construcción de software.

a) Eficiencia en Desarrollo: La autogeneración de código con intervención de la AI a partir de algoritmos imperativos sintácticamente elaborados reduce significativamente el tiempo de desarrollo, permitiendo a los profesionales e ingenieros de Software centrarse en otras tareas más relevantes como el análisis de procesos, especificación de reglas de negocio, la lógica y la arquitectura del software.

b) Mejora en la optimización y precisión: La implementación de la Ingenieria Rápida (Prompt Engineering) en la construcción de software contribuye a la generación de código más preciso y alineado con las directivas del desarrollador entrenado en Metodologías de Ingeniería de Software, optimizando recursos computacionales, minimizando errores y facilitando el mantenimiento del ciclo de versiones de software.

c) Escalabilidad y Adaptabilidad: La técnica Prompt Engineering, permite adaptarse a diferentes paradigmas de programación y entornos, ofreciendo flexibilidad para trabajar con toda gama de lenguajes de programación disponibles en la actualidad sean Compilados o Interpretados.

d) Formación Efectiva: Los modelos LLM y la Ingeniería Rápida (Prompt Engineering), facilita la formación de profesionales especializados en el desarrollo de software, al permitirles enfocarse en el análisis y resolución de problemas, liberándose de tareas agotadoras, repetitivas y rutinarias de codificación.

e) Reducción de Costos: La automatización de la generación de código con la intervención de la AI contribuye a la reducción de costos asociados al desarrollo, aliviando la carga de trabajo y permitiendo la asignación de recursos de manera más eficiente.

f) Formación innovadora: Los avances de la tecnología con la intervención de la IA generativa, abre grandes posibilidades en el futuro inmediato. Los profesionales que decidan especializarse en la IA

generativa desde sus tempranas etapas, van a adquirir capacidades, habilidades y los conocimientos necesarios para dar forma a su futuro profesional y laboral.

3.3.4. Análisis de la productividad y eficiencia en la Ingeniería de Software.

Pregunta: ¿Estarías de acuerdo en que lo importante en la construcción de software es atender el análisis de requerimientos y el diseño de Algoritmos más que la implementación o escritura mecánica de código?

Aquí hay algunas razones para respaldar esta afirmación:

a) Claridad en los Requerimientos:

   Un análisis de requerimientos sólido garantiza una comprensión clara y completa de lo que se espera del software. Esto es crucial para evitar malentendidos y para proporcionar una base sólida para el diseño y la implementación.

b) Diseño Efectivo:

   Un buen diseño establece la estructura y la arquitectura del software. Define cómo los diferentes componentes interactúan entre sí y cómo se abordan los desafíos específicos. Un diseño bien pensado facilita la implementación y mantenimiento futuros.

c) Eficiencia y Rendimiento:

   La elección de algoritmos adecuados en la etapa de diseño puede tener un impacto significativo en la eficiencia y el rendimiento del software. Un diseño cuidadoso considera aspectos como la complejidad algorítmica y la optimización.

d) Identificación de Problemas:

   El análisis y diseño exhaustivos pueden ayudar a identificar posibles problemas y desafíos antes de llegar a la etapa de implementación. Esto permite abordar problemas en una fase temprana, cuando son más fáciles y menos costosos de corregir.

e) Facilita la Colaboración:

   Un buen análisis de requerimientos y diseño proporciona una base clara para la colaboración entre miembros del equipo. Facilita la comprensión compartida y la comunicación efectiva en todo el ciclo de desarrollo.

Si bien la implementación es esencial, no se puede subestimar la importancia de un análisis detallado y un diseño cuidadoso. La implementación eficiente y exitosa se basa en gran medida en la calidad de las decisiones tomadas en las etapas de análisis y diseño. En un contexto en el que la ingeniería rápida y la

inteligencia artificial pueden ocuparse de la implementación, la capacidad humana para realizar un análisis profundo y diseñar soluciones efectivas se vuelve aún más crucial.

3.3.5. Algoritmos Imperativos como insumo para GenAI aplicando técnicas de Ingeniería Rápida.

Como Ingeniero de Sistemas pretendo implementar una Especificación de diseño de algoritmos basado en normativas del modelo Imperativo que incluye: Método Estructurado, Método Procedimental y el Método Orientado a Objetos. Esta especificación de diseño se denomina: "Algorithmic Specification & Type Hints Notations for AI".

Aspectos Positivos:

a) Enfoque Metódico:

   La inclusión de Métodos, Patrones y Técnicas del Modelo Imperativo refleja un enfoque metódico en el diseño de algoritmos. Esto puede proporcionar una base sólida y estructurada para el desarrollo de software.

b) Consideración de Paradigmas de Programación:

   La inclusión de la Programación Estructurada, Procedimental y Orientada a Objetos sugiere una comprensión de diferentes paradigmas de programación, permitiendo adaptarse a diversos contextos y requisitos del proyecto.

c) Type Hints Notations:

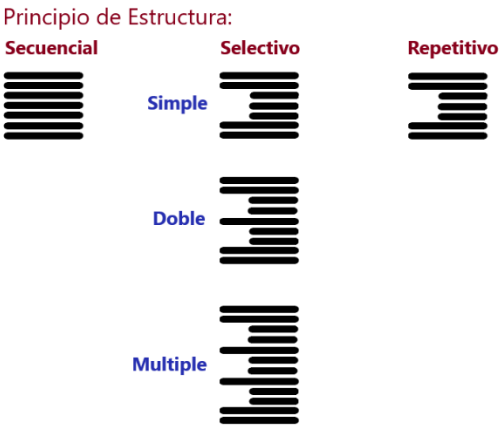
   La incorporación de Type Hints es una práctica beneficiosa para mejorar la claridad y la robustez del código. Ayuda a los desarrolladores a comprender mejor la intención del código y facilita la detección de errores antes de la ejecución.

d) Abordaje Integral:

   La combinación de elementos como métodos, patrones, técnicas y type hints sugiere un enfoque integral para la especificación de diseño, abordando diferentes aspectos de la programación.

"Algorithmic Specification & Type Hints Notations for AI".

Algoritmos Imperativos	Patrones	Investigadores - Meteorologistas
Método Estructurado	Secuence, Selection, Repeat	(Bohm y Jacopini - 1960)
Método Procedimental	Procedure - Function	(Edsger W. Dijkstra - 1960 y Niklaus Wirth - 1988)
Método Orientado a Objetos	Class - Object	(Ole-Johan Dahl y Kristen Nygaard -1962)




```
Algoritmo Circulo v2
var
        real: radio, area, longitud
inicio
        //entrada:  invocacion sub-rutina
        radio <- leerRadio()

        //invocar proceso 1
        area <- calcularArea(radio)

        //invocar proceso 2
        longitud  <- circunferencia(radio)

        //salida: invocar sub-rutina
        llamar_a  escribirResultados(area, longitud)
fin
```

```
//Sub-rutinas: Definicion de prototipos
real funcion leerRadio()
var
    real: r
inicio
    repetir
                escribir("Ingrese radio: ")
                leer(r)
    hasta_que(r > 0)
    devolver (r)
fin_funcion

real funcion calcularArea(E real: r)
var
    real: s
    const PI <- 3.141592
inicio
    s <- PI * r * r
    devolver (s)
fin_funcion

real funcion circunferencia(E real: r)
var
    real: p
    const PI <- 3.141592
inicio
    p <- 2 * PI * r
    devolver (p)
fin_funcion

procedimiento escribirResultados(E real: s, p)
var
inicio
        escribir("Area del circulo: ", s)
        escribir("Y circunferencia: ", p)
fin_procedimiento
```

Sugerencias para Mejorar:

a) Documentación Clara:

   Asegúrate de proporcionar una documentación clara que explique la metodología detrás de la especificación. Esto ayudará a los desarrolladores a comprender cómo aplicarla efectivamente.

b) Ejemplos Prácticos:

   Incluye ejemplos prácticos que demuestran cómo implementar la especificación en la práctica. Esto facilitará la comprensión y la adopción por parte de los desarrolladores.

c) Flexibilidad:

   Asegúrate de que la especificación sea lo suficientemente flexible para adaptarse a diferentes tipos de proyectos y requisitos. La flexibilidad es clave en el desarrollo de software.

d) Actualización Continua:

   Considera la posibilidad de mantener la especificación actualizada a medida que evolucionan las prácticas y tecnologías en el campo de la ingeniería de software.

e) Colaboración y Retroalimentación:

   Fomenta la colaboración y busca retroalimentación de otros profesionales de la ingeniería de software. La mejora continua basada en experiencias prácticas es esencial.

Por lo expuesto, la idea de crear una especificación de diseño de algoritmos con un enfoque en Métodos, Patrones y Técnicas del Modelo Imperativo, complementada con una especificación y notación Type Hints, es positiva. La clave será proporcionar una guía clara y práctica que beneficie a los desarrolladores en su formación continua y trabajo diario.

.

CAPÍTULO IV: PRINCIPALES CONTRIBUCIONES

4.1. DESCRIPCIÓN DE ALTERNATIVAS DE SOLUCIÓN

4.1.1 Alternativas disponibles de los Modelos LLM.

a) ¿Qué posibilidades ofrece la API del modelo LLM de OpenAI, Gemini, y otros proveedores AI?

El modelo LLM tiene características que permiten crear contenidos multimodales como sonido, video, escritura y conversaciones que suenan mucho más naturales y próximos al humano. Los modelos LLM han sido alimentados con una extensa gama de información de diversas fuentes provenientes de documentos técnicos y de investigaciones, libros de distintas especialidades, papers profesionales, blogs, redes sociales y la inmensa data que subyace en Internet. Los LLM de inteligencia artificial están diseñados y entrenados para responder a consultas desde las más triviales hasta las más especializadas y complejas (Raffel, C. et.al., 2019).

b) ¿Qué es y para qué sirve la API de Modelos LLM como Gemini, Open AI, etc. ?

Según, Terol, Moncho en ThinkBig - Telefónica, describe una API de inteligencia Artificial como:

> Funcionalmente, una API implementada en los Modelos LLM de inteligencia artificial, es un conjunto de especificaciones definidas por parámetros, que rigen la comunicación entre componentes de software. El objetivo de una API es facilitar la integración y comunicación de sistemas y garantizar la seguridad e integridad en el intercambio de datos. Su propósito final es solventar la conexión de sistemas de distintos orígenes (marzo 2023, párr. 6).

4.1.2. Proveedores Representativos de Modelos LLM de Inteligencia Artificial (AI).

Algunos proveedores que compiten en el campo de la inteligencia artificial y la creación de modelos de lenguaje:

OpenAI - IBM Watson - Google Genini AI

a) Google LLC:

- Google es una compañía multinacional de tecnología conocida por su amplio conjunto de herramientas de inteligencia artificial, incluyendo TensorFlow y BERT. Ofrece servicios como Google Cloud AI Platform que permiten a los desarrolladores construir y desplegar modelos de lenguaje y otras aplicaciones de IA (Demis H, Sundar P., Dec 06, 2023).

- Google Gemini AI es el primer modelo que sobrepasa a los expertos humanos en MMLU (Massive Multitask Language Understanding: Entendimiento de Lenguaje Multitarea Masiva),

una de las técnicas más comunes para evaluar el conocimiento y la habilidad para resolver problemas de los modelos de inteligencia artificial.

- Los modelos Gemini son multimodales de forma nativa, lo que le brinda el potencial de transformar cualquier tipo de entrada en cualquier tipo de salida:
    - Los modelos Gemini pueden generar código en función de las diferentes entradas que usted les proporcione.
    - Los modelos Gemini pueden generar texto e imágenes combinados.
    - Los modelos de Géminis pueden razonar visualmente en distintos idiomas.

b) IBM:

- IBM Watson es una plataforma de inteligencia artificial desarrollada por IBM. Está diseñada para entender y procesar lenguaje natural, analizar imágenes y realizar análisis avanzados. Watson puede potenciar soluciones de asistentes virtuales y otras aplicaciones de IA (IBM Education, September 19, 2023).

- IBM Watson ofrece una amplia gama de servicios, incluyendo análisis de lenguaje natural, asistentes virtuales y herramientas de procesamiento del habla. IBM también ha desarrollado modelos de lenguaje como Project Debater.

- El Asistente de Código de IBM Watson ayuda a los programadores a crear código mediante sugerencias generadas por inteligencia artificial, independientemente de su nivel de habilidad. Los desarrolladores pueden hacer solicitudes en un lenguaje simple o utilizar el código existente para generar código destinado a casos de uso particulares. El Asistente de Código de Watson proporciona modelos pre entrenados adaptados a lenguajes de programación específicos, asegurando confiabilidad en generación precisa de código.

c) OpenAI:

- OpenAI es una compañía de investigación en inteligencia artificial ubicada en San Francisco, California. Su creación se dio con el propósito de avanzar en el desarrollo de inteligencia artificial amigable y beneficiosa para la humanidad. OpenAI ha sido líder en la creación de modelos de lenguaje de gran envergadura, como GPT (Transformador Pre-entrenado Generativo), los cuales tienen la capacidad de producir texto de manera coherente y pertinente. Estos modelos se han aplicado en diversas áreas, desde la generación de texto hasta la traducción automática y la creación de chatbots inteligentes (OpenAI, Blog November 6, 2023).

4.2. EVALUACIÓN DE ALTERNATIVAS DE SOLUCIÓN.

4.2.1. ¿Qué es una API de Modelo LLM de inteligencia artificial?

Desde el punto de vista de la Ingeniería de Software, una API de Modelo LLM (Modelo de Lenguaje de Inteligencia Artificial) es una interfaz de programación de aplicaciones diseñada para permitir a los desarrolladores interactuar con modelos de lenguaje de inteligencia artificial de manera programática. Esta API proporciona métodos y funciones que permiten enviar solicitudes de texto al modelo de lenguaje y recibir respuestas generadas por el modelo (Liu, Y., et.al., Jul. 2019). Aquí hay algunas características tecnológicas clave de una API de Modelo LLM:

1. Interfaz de comunicación estandarizada: La API define una interfaz de comunicación estandarizada que especifica cómo los desarrolladores pueden interactuar con el modelo de lenguaje, incluyendo los métodos disponibles, los parámetros de entrada y los formatos de salida.
2. Capacidad de procesamiento de lenguaje natural (NLP): La API está diseñada para procesar texto natural de manera efectiva, lo que permite al modelo de lenguaje comprender y generar texto coherente y relevante.
3. Escalabilidad y rendimiento: La API está optimizada para admitir un alto volumen de solicitudes y proporcionar respuestas rápidas, lo que garantiza un rendimiento eficiente incluso en entornos de producción con grandes cargas de trabajo.
4. Flexibilidad y adaptabilidad: La API es flexible y puede adaptarse a diferentes casos de uso y requisitos específicos del usuario. Puede admitir una variedad de modelos de lenguaje y configuraciones personalizadas según las necesidades del desarrollador.
5. Seguridad y privacidad: La API incorpora medidas de seguridad para proteger los datos confidenciales y garantizar la privacidad del usuario. Esto puede incluir autenticación de usuarios, cifrado de datos y cumplimiento de regulaciones de privacidad, como GDPR.

Estas características tecnológicas son fundamentales para el diseño y la implementación exitosa de una API de Modelo LLM de inteligencia artificial, ya que garantizan su funcionalidad, rendimiento, seguridad y facilidad de uso para los desarrolladores que la utilizan en sus aplicaciones y sistemas informáticos.

4.2.2. Utilidad de las API de los Modelos LLM.

La exposición abierta de la API de los modelos LLM permite acceder y tener control muy específico sobre los contenidos especializados con los que han sido entrenados estos modelos LLM inteligentes. También posibilita la creación de chatbots que pueden responder en tiempo real tanto a mensajes de voz como de texto, entre otras utilidades. Es una API que permite implementar en sitios Web y aplicaciones

Móvil embeber e integrar chatbots configurados en tareas de procesamiento de lenguaje natural. La disponibilidad de una API de inteligencia artificial permite utilizar para construir diálogos personalizados, generar respuestas especializadas en tiempo real a preguntas específicas, entre otras utilidades (Pérez, L., Junio 2023).

¿Qué cosas se puede hacer con las API de los Modelos LLM como OpenAI y Google?

Las API de los Modelos de Lenguaje de Gran Escala (LLM) como OpenAI y Google ofrecen una variedad de utilidades y posibilidades en diferentes campos. Aquí tienes algunas utilidades que se pueden hacer con estas API:

1. Generación de texto: Esto puede ser útil para completar frases, generar ideas para contenido creativo, escribir artículos, entre otros.
2. Traducción automática: Aunque no están diseñadas específicamente para la traducción, pueden generar traducciones razonablemente precisas en muchos casos.
3. Resumen de texto: Esto es útil para procesar grandes volúmenes de información de manera eficiente.
4. Chatbots inteligentes: Con las API de LLM, se pueden construir chatbots inteligentes capaces de interactuar con usuarios de manera natural y proporcionar respuestas útiles a sus consultas o realizar tareas específicas.
5. Análisis de sentimientos: Esto es útil para realizar análisis de opiniones en redes sociales, comentarios de clientes, etc.
6. Generación de código: Esto puede ser útil para automatizar tareas de desarrollo de software o generar prototipos rápidos.
7. Creación de contenido: Esto puede ayudar a agilizar el proceso de creación de contenido y proporcionar ideas creativas para nuevas publicaciones.

Estas son solo algunas de las muchas aplicaciones posibles de las API de Modelos de Lenguaje de Gran Escala como OpenAI y Google. Su flexibilidad y capacidad para comprender y generar texto de manera efectiva las hacen útiles en una amplia gama de campos y aplicaciones.

4.3. IMPLEMENTACIÓN DE ALTERNATIVA SELECCIONADA ACTIVIDADES Y PROCEDIMIENTOS

4.3.1 Asistencia académica de chatbots en metodología de la programación

El siguiente paso natural a evaluar como alternativa de solución es justamente la construcción de Chatbots inteligentes de asistencia en el ámbito de la metodología de la programación Imperativa.

La API de un Modelo LLM permite personalizar las respuestas de un chatbot con gran flexibilidad, logrando imprimir la personalización deseada en cada interacción con el chatbot. Esto se logra ajustando los parámetros de los modelos de lenguaje, el contexto, tema específico, el tono y el estilo de las respuestas (Diaz, S., febrero de 2024).

Este chatbot, puede tener tres tareas específicas de asistencia:

1. Asistencia en el aprendizaje de la Metodología Imperativa en sus tres ámbitos:
    a. Principios y teoremas de la Programación Estructurada.
    b. Principios y Teoremas de la Programación Procedimental (o Modular) y los
    c. Principios y teoremas de la Programación Orientada a Objetos
2. Asistencia en el Diseño de algoritmos basados en la especificación: "Algorithmic Specification & Type Hints Notations for AI".
3. Asistencia parametrizada en la Generación de código (GenAI) para implementar algoritmos imperativos en distintos lenguajes de programación vigentes en el mercado (Compilados e Interpretados)

4.3.2. Análisis, diseño e implementación de chatbot de asistencia académica en Metodología de la programación Imperativa.

1. Definición de objetivos y alcance:

    Identificar el propósito del chatbot y los objetivos que se espera que cumpla. ¿Qué tipo de asistencia académica proporcionará? ¿Cuál será su alcance en términos de temas cubiertos y nivel de detalle?

2. Selección de la plataforma y tecnología:

    Elegir la plataforma o herramienta de desarrollo de chatbots que mejor se adapte a tus necesidades y habilidades técnicas. Puedes considerar opciones como Dialogflow, IBM Watson Assistant, Microsoft Bot Framework, entre otros. Además, decide qué tecnologías utilizarás para la implementación, como lenguajes de programación (por ejemplo, Python), frameworks y bibliotecas de IA.

3. Diseño de la conversación:

   Diseña el flujo de conversación del chatbot, definiendo los diferentes tipos de preguntas que los usuarios podrían hacer y las respuestas correspondientes del chatbot. Considera cómo manejar diferentes escenarios y consultas específicas relacionadas con la metodología de la programación imperativa.

4. Desarrollo del chatbot:

   Implementar el chatbot utilizando la plataforma y tecnologías seleccionadas. Esto implica la creación de intents (prototipado de formas GUIs), entidades, diálogos y respuestas del chatbot. Utiliza bibliotecas y herramientas de procesamiento del lenguaje natural (NLP) para comprender y responder a las consultas de los usuarios de manera efectiva.

5. Pruebas y refinamiento:

   Realizar pruebas exhaustivas del chatbot para asegurarte de que funcione correctamente y proporcione respuestas precisas y útiles al contexto. Recopilar comentarios de los usuarios y realizar ajustes según sea necesario para mejorar la experiencia del usuario y la efectividad del chatbot.

6. Despliegue y mantenimiento:

   Desplegar el chatbot en la plataforma seleccionada y asegurar de que esté disponible para su uso. Además, establecer un plan de mantenimiento para garantizar que el chatbot siga funcionando correctamente y realizar actualizaciones según sea necesario para mantenerlo relevante y eficaz.

7. Evaluación de resultados:

   Examinar de forma periódica el desempeño del chatbot implica analizar indicadores como la eficacia para resolver consultas, el nivel de satisfacción del usuario y la precisión en las respuestas. Estos datos se emplean para detectar áreas de posible mejora y para realizar ajustes continuos destinados a optimizar el funcionamiento del chatbot.

Siguiendo estas pautas como guía de desarrollo, podrás construir un Chatbot de Asistencia Académica en Metodología de la Programación Imperativa efectivo y útil para los estudiantes. Recuerda que el proceso puede requerir iteraciones y ajustes a medida que obtienes retroalimentación y aprender más sobre las necesidades de los usuarios.

4.3.3. ChatBot para entorno Web & Móvil.

Detalles técnicos sobre el desarrollo del chatbot, tanto para el entorno web como para aplicaciones móviles smartphone.

a) Desarrollo del Chatbot para Entorno Web.

1. Elección de la plataforma de desarrollo:

   Se necesita utilizar frameworks y bibliotecas populares como Flask o Django si se está utilizando Python como lenguaje de programación. Para la parte del frontend, se usan las tecnologías convencionales como HTML, CSS y JavaScript, o frameworks como React o Angular.

2. Implementación del backend:

   Crear un servidor web utilizando el framework elegido y definir correctamente los endpoints para manejar las solicitudes del chatbot. Utilizar una biblioteca de procesamiento del lenguaje natural (NLP) como spaCy o nltk para comprender y procesar las consultas de los usuarios.

3. Integración de la lógica del chatbot:

   Implementar la lógica del chatbot, incluyendo la interpretación de las consultas de los usuarios, la selección de respuestas adecuadas y la generación de respuestas basadas en el contexto de la conversación. Se recomienda utilizar bibliotecas de chatbot como Rasa o ChatterBot para facilitar este proceso.

4. Desarrollo del frontend:

   Diseñar y desarrollar la interfaz de usuario del chatbot en el frontend utilizando HTML, CSS y JavaScript. Esto puede incluir la creación de una interfaz de chat interactiva donde los usuarios puedan escribir sus consultas y ver las respuestas del chatbot.

5. Pruebas y depuración:

   Realizar pruebas exhaustivas del chatbot en el entorno web para asegurarte de que funcione correctamente en diferentes navegadores y dispositivos responsivos. Realizar ajustes y correcciones según sea necesario para mejorar la experiencia del usuario y la funcionalidad del chatbot.

b) Desarrollo del Chatbot para Aplicaciones Móviles Smartphone:

1. Elección del framework de desarrollo:

   Es recomendable utilizar frameworks de desarrollo de aplicaciones móviles multiplataforma como React Native o Flutter para construir el chatbot de manera eficiente y mantener la compatibilidad con múltiples plataformas.

2. Implementación del backend:

   Crear un backend similar al utilizado para el chatbot web, con endpoints definidos claramente para manejar las solicitudes del chatbot y la lógica de procesamiento de texto. Es recomendable reutilizar parte del código del backend web si es compatible con la arquitectura de tu aplicación móvil.

3. Desarrollo del frontend móvil:

   Diseña y desarrolla la interfaz de usuario del chatbot en la aplicación móvil utilizando los componentes y herramientas proporcionados por el framework elegido. Asegúrate de optimizar la experiencia del usuario para pantallas móviles y dispositivos táctiles.

4. Integración con plataformas de mensajería:

   Es deseable que el chatbot sea accesible a través de aplicaciones de mensajería como Telegram, Facebook Messenger o WhatsApp, integrando la funcionalidad del chatbot con las API proporcionadas por estas plataformas.

5. Pruebas y depuración:

   Lleva a cabo pruebas detalladas del chatbot en dispositivos móviles para garantizar su correcto funcionamiento en diversas versiones de sistemas operativos y tipos de dispositivos. Realiza modificaciones y correcciones según sea requerido para mejorar tanto la experiencia del usuario como la funcionalidad del chatbot en entornos móviles.

Siguiendo este procedimiento de desarrollo, se logra construir un chatbot efectivo tanto para entornos web como para aplicaciones móviles smartphone. Recordar y tener en cuenta las mejores prácticas de desarrollo y optimización para garantizar una experiencia de usuario fluida y satisfactoria.

4.4. COSTO DE IMPLEMENTACIÓN.

Costos asociados con las tareas de generación de código con Prompt Engineering AI.

1. Costos de tareas de procesamiento del lenguaje natural (NLP) con aprendizaje supervisado y Prompt Engineering:
   Implementación de modelos de NLP: Los costos pueden incluir la adquisición de Herramientas AI, Chatbots y planes de servicio AI, para entrenar y ejecutar modelos de NLP, así como la contratación de profesionales en inteligencia artificial.
   Ingeniería de prompts: Los costos de la ingeniería de prompts incluyen el diseño y la creación de preguntas y respuestas de muestra para entrenar modelos de manera efectiva, lo que puede requerir la participación de expertos en el dominio y en NLP.
   Adquisición de datos etiquetados: recopilación de datos etiquetados para el entrenamiento supervisado puede requerir costos asociados con la obtención, análisis de prompts y anotación de datos.
2. Costos de interacción y uso continuo de Chatbots AI (Gemini Ultra, ChatGPT 4) aplicando técnicas de Prompt Engineering en la generación de código para lenguajes imperativos:
   Desarrollo del chatbot: Los costos incluyen el diseño, desarrollo e implementación del chatbot utilizando la API de modelos LLM como Google Gemini, OpenAI, así como la integración con el entorno de desarrollo de software.
   Mantenimiento y actualización: Los costos asociados con el mantenimiento continuo del chatbot, incluida la solución de problemas, la optimización del rendimiento y la implementación de nuevas características.
   Consumo de recursos de la API: Dependiendo del plan de precios de la API de ChatGPT, los costos pueden aumentar con el uso continuo y la interacción del chatbot, especialmente en entornos de alta demanda.

3. Costos de rendimiento en la calidad de pregunta-respuesta mediante Prompt Engineering:
   Análisis de datos y retroalimentación: Los costos pueden incluir la recopilación y análisis de datos de interacción del chatbot para evaluar la calidad de las respuestas e identificar áreas de mejora.
   Optimización continua: Los costos asociados con la optimización continua del chatbot, incluida la modificación de prompts, el reentrenamiento de modelos y la implementación de nuevas estrategias de respuesta.
   Retroalimentación de expertos: Involucrar a expertos en el dominio para revisar y proporcionar retroalimentación sobre la calidad de las respuestas del chatbot puede requerir costos adicionales.

En resumen, los costos asociados con las tareas de inteligencia artificial mencionadas incluyen recursos de desarrollo, adquisición de datos, consumo de recursos de API y análisis continuo del rendimiento del

sistema. Es importante considerar estos costos al planificar e implementar soluciones de inteligencia artificial para garantizar una inversión efectiva y un rendimiento óptimo del sistema.

Tabla 1. Costos de implementación.

Costos asociados con las tareas de inteligencia artificial y Prompt Engineering.

Tarea	**Componentes de Costo**
Procesamiento del Lenguaje Natural (NLP)	- Implementación de modelos de NLP (Tools AI, Planes de servicio AI)
	- Adquisición de datos etiquetados
	- Ingeniería de prompts (Diseño y Análisis de Prompts)
Interacción y uso continuo de Chatbots AI	- Desarrollo del chatbot (diseño, desarrollo, implementación)
	- Mantenimiento y actualización
	- Consumo de recursos de la API de modelos LLM (Google Gemini, OpenAI)
Rendimiento en la calidad de pregunta-respuesta	- Análisis de datos y retroalimentación para evaluar la calidad de las respuestas e identificar áreas de mejora.
Mediante Prompt Engineering	- Optimización continua por modificación de Prompts
	- Retroalimentación de expertos

Nota. Esta tabla proporciona una visión general de los componentes de costos asociados con cada tarea de inteligencia artificial y Prompt Engineering. Cada tarea tiene sus propios costos específicos que deben tenerse en cuenta al planificar e implementar soluciones de inteligencia artificial.

4.5. EVALUACIÓN DE FACTIBILIDAD DE LA IMPLEMENTACIÓN.

Factibilidad asociados a las tareas de generación de código con Prompt Engineering AI.

4.5.1. Evaluación de la factibilidad de incursionar e impulsar el uso de modelos inteligentes en la construcción de software y la formación de profesionales desarrolladores:

1. Factibilidad técnica:
   Recursos computacionales: Se requerirá hardware adecuado para entrenar y ejecutar modelos inteligentes, lo que puede ser costoso.
   Herramientas y plataformas: Es fundamental contar con herramientas y plataformas de desarrollo de inteligencia artificial accesibles y fáciles de usar.
   Experiencia y conocimientos: Se necesitará un equipo de profesionales con experiencia en inteligencia artificial y metodologías de desarrollo de software.
2. Factibilidad económica:
   Costos de formación: La capacitación de profesionales en inteligencia artificial puede ser costosa y requerir una inversión significativa.
   Costos de adquisición de herramientas: Adquirir herramientas y plataformas de desarrollo de inteligencia artificial puede tener un costo inicial considerable.
3. Factibilidad organizativa:
   Cambio de cultura organizativa: Puede requerirse un cambio en la cultura organizativa para adoptar y aprovechar los modelos inteligentes en el proceso de desarrollo de software.

4.5.2. Evaluación de la factibilidad de los pros y contras de los Modelos de Inteligencia Artificial entrenados con datos de Internet:

1. Factibilidad técnica:
   Desafíos de calidad de datos: Los datos de Internet pueden ser ruidosos y no estar etiquetados, lo que puede afectar la calidad y precisión del modelo entrenado.
2. Factibilidad económica:
   Costos de adquisición de datos: El suministro de datos de Internet puede ser costoso, especialmente si se requieren conjuntos de datos específicos y de alta calidad.
3. Factibilidad operativa:
   Proceso de evaluación regular: Es factible establecer un proceso de evaluación regular del rendimiento del chatbot, lo que permite identificar áreas de mejora y optimización continua.

En resumen, las tareas mencionadas son factibles desde el punto de vista técnico, económico y operativo, siempre y cuando se asignen los recursos adecuados y se aborden los desafíos asociados. Es importante realizar un análisis exhaustivo de la factibilidad en cada etapa del proceso de implementación de inteligencia artificial para garantizar el éxito del proyecto.

Tabla 2. Factibilidad

Evaluación de la factibilidad asociada a las tareas de inteligencia artificial mencionadas:

Aspecto	Incursionar en Modelos Inteligentes en la Construcción de Software	Modelos de IA Entrenados con Datos de Internet	Evaluación Regular del Rendimiento de Chatbot-AI
Factibilidad Técnica	- Recursos computacionales adecuados - Herramientas y plataformas accesibles - Experiencia y conocimientos en IA	- Disponibilidad de datos de Internet - Desafíos de calidad de datos	- Disponibilidad de métricas de rendimiento - Herramientas de análisis de datos
Factibilidad Económica	- Costos de formación y adquisición de herramientas - Inversión inicial significativa	- Costos de adquisición de datos - Recursos financieros para entrenamiento	- Costos de análisis de datos y recursos adicionales
Factibilidad Operativa	- Cambio de cultura organizativa - Integración con procesos existentes	- Factible con la supervisión adecuada - Gestión de desafíos de calidad de datos	- Establecimiento de un proceso de evaluación regular - Identificación de áreas de mejora

Nota. Esta tabla proporciona una comparación estructurada de la factibilidad asociada a las tareas de inteligencia artificial mencionadas, destacando los aspectos técnicos, económicos y operativos asociados con cada uno de ellos.

CONCLUSIONES

El enfoque del presente trabajo de investigación es la integración de algoritmos del modelo imperativo, la autogeneración de código (GenAI) en cualquier lenguaje de programación vigente y la aplicación de la ingeniería rápida (Prompt Engineering AI) de la inteligencia artificial. El objetivo principal es transformar los hábitos y el enfoque tradicional en la construcción de software, así como modificar el modelo educativo para la formación de profesionales en este campo. Se busca liberar a los profesionales tecnológicos de las tareas rutinarias y abrumadoras que consumen su tiempo y atención.

La implementación de la inteligencia artificial en los planes de estudio de instituciones de educación superior formará profesionales en una realidad cada vez más cercana a los principios fundamentales de la Ingeniería: simplificar procesos, optimizar el uso de recursos y reducir los tiempos de desarrollo. Sin embargo, para lograr este objetivo, es necesario reformular los contenidos curriculares de institutos tecnológicos y universidades que ofrecen programas relacionados con la Ciencia Computacional y, específicamente, la Ingeniería de Software.

En la educación universitaria de ingenieros de software, esto implica la incorporación de conceptos avanzados que podrían transformar la forma en que se enseña y se desarrolla software. La relevancia radica en la preparación de los estudiantes para adoptar enfoques más eficientes y avanzados, alineados con las tendencias actuales en inteligencia artificial, lo que podría impactar positivamente en la productividad y la calidad del desarrollo de software.

Adoptar la automatización mediante inteligencia artificial (IA) e incluir técnicas de “Ingeniería Rápida” en el proceso de escritura de código tiene el potencial de liberar a los desarrolladores de tareas repetitivas y rutinarias. Esto puede permitir que los profesionales de la programación se centren en tareas más estratégicas y creativas en el desarrollo de software.

Si bien la implementación es esencial, no se puede subestimar la importancia de un análisis detallado y un diseño cuidadoso. La implementación eficiente y exitosa se basa en gran medida en la calidad de las decisiones tomadas en las etapas de análisis y diseño. En un contexto en el que la ingeniería rápida y la inteligencia artificial pueden ocuparse de la rutina de implementación, y dejar al profesional la capacidad humana para realizar un análisis profundo y diseñar soluciones efectivas, se vuelve aún más crucial.

RECOMENDACIONES

Como profesional en Ingeniería de Sistemas y Cómputo, recomiendo implementar una Especificación de diseño de algoritmos basado en Métodos, Patrones y Técnicas del Modelo Imperativo como insumo de aprendizaje automático en los Modelos LLM (Large Linguistic Model) basados en AI (Artificial Intelligence).

Aporte profesional.

> Especificación y Notación para el Análisis y Diseño: "Algorithmic Specification & Type Hints Notation for AI".
>
> Implementación: La idea de implementar una especificación de diseño de algoritmos con un enfoque Metódico sustentado en Patrones y Técnicas del Modelo Imperativo, y complementada con Type Hints Notation, es pertinente, necesaria y productiva.
>
> Productividad: La clave será proporcionar una especificación con reglas claras y prácticas que beneficie a los desarrolladores en su formación continua y trabajo diario, delegando la agotadora tarea de la escritura de código a la Inteligencia Artificial.

Es importante destacar que, aunque la inteligencia artificial puede asistir en la generación de código, la participación humana sigue siendo esencial para la toma de decisiones estratégicas, la comprensión del contexto empresarial y la garantía de la calidad del software. Además, los desarrolladores pueden encontrar nuevas oportunidades para expandir sus habilidades y contribuir de manera más significativa al proceso de desarrollo de software en su conjunto.

La introducción de la Ingeniería Rápida en la formación universitaria de Ingenieros de Software ofrece varias ventajas, como la adopción de tecnologías avanzadas, eficiencia en el desarrollo y preparación para el futuro. Sin embargo, es esencial abordar desafíos potenciales, como la posible pérdida de conocimientos fundamentales y la dependencia excesiva de herramientas específicas. En general, el impacto dependerá de cómo se equilibre la automatización con la necesidad de mantener una comprensión profunda de los principios de la ingeniería de software.

"*Lo importante en la construcción de software es atender el análisis de requerimientos y el diseño de Algoritmos, más que la implementación*".

El análisis de requerimientos y el diseño de algoritmos son etapas fundamentales en el proceso de construcción de software eficaz y, en muchos casos, son incluso más críticos que la implementación en sí misma.

Sugerencias para Mejorar:

Documentación Clara:

> Asegúrate de proporcionar una documentación clara que explique la metodología detrás de la especificación. Una normativa con reglas explícitas ayudará a los desarrolladores a comprender cómo aplicarla efectivamente.

Ejemplos Prácticos:

> Incluye ejemplos prácticos que demuestran cómo implementar la especificación en la práctica. Esto facilitará la comprensión y la adopción por parte de los desarrolladores.

Flexibilidad:

> Asegúrate de que la especificación sea lo suficientemente flexible para adaptarse a diferentes tipos de proyectos y requisitos. La flexibilidad es clave en el desarrollo de software.

Actualización Continua:

> Considera la posibilidad de mantener la especificación actualizada a medida que evolucionan las prácticas y tecnologías en el campo de la ingeniería de software.

Colaboración y Retroalimentación:

> Fomenta la colaboración y busca retroalimentación de otros profesionales de la ingeniería de software. La mejora continua basada en experiencias prácticas es esencial.

En resumen, la inclusión de los modelos LLM de inteligencia artificial en la Ingeniería de Software puede llevar a una mejora general en la calidad, eficiencia y usabilidad del software, lo que a su vez lleva a una mayor satisfacción del cliente y un mejor rendimiento comercial.

REFERENCIAS BIBLIOGRÁFICAS.

Alejandro Preinfalk. “La ingeniería rápida, el ingrediente principal en el mundo de la IA”. Publicado: 05 julio 2023. COMPUTERWEEKLE.ES - TechTarget, S.A de C.V 2013 - 2024. https://www.computerweekly.com/es/opinion/La-ingenieria-rapida-el-ingrediente-principal-en-el-mundo-de-la-IA.

Arrendajo. (18 de noviembre 2023) Comprender la IA multimodal. HashDork - Publicaciones Squeeze Growth® LLP| 2020 – 2024. https://hashdork.com/es/ia-multimodal/

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... & Amodei, D. (2020). "Language Models are Few-Shot Learners." arXiv preprint arXiv:2005.14165. https://arxiv.org/abs/2005.14165

Caines, A., Benedetto, L., Taslimipoor, S., Davis, C., Gao, Y., Andersen, O., ... & Buttery, P. (2023). On the application of large language models for language teaching and assessment technology. arXiv preprint arXiv:2307.08393. https://arxiv.org/pdf/2307.08393.pdf

Demis Hassabis, Sundar Pichai. AI: Introducing Gemini: our largest and most capable AI model. Google Dec 06, 2023. https://blog.google/technology/ai/google-gemini-ai/#sundar-note

Díaz, Sara. Crea un chatbot realmente inteligente con la API de OpenAI: Guía práctica. OpenWebinars.net - Inteligencia Artificial Publicado el 16 de Febrero de 2024. https://openwebinars.net/blog/chatbot-api-openai/#:~:text=La%20API%20de%20OpenAI%20ofrece,el%20estilo%20de%20las%20respuestas

Dijkstra, E. W. (1972). The humble programmer. Communications of the ACM, 15(10), 859-866. https://dl.acm.org/doi/pdf/10.1145/355604.361591

Gerhard, D., Köring, T., & Neges, M. (2022, July). Generative Engineering and Design–A Comparison of Different Approaches to Utilize Artificial Intelligence in CAD Software Tools. In IFIP International Conference on Product Lifecycle Management (pp. 206-215). Cham: Springer Nature Switzerland. https://link.springer.com/chapter/10.1007/978-3-031-25182-5_21

Gutiérrez-Jiménez, D. A., Salazar-Colores, S., Rivera, F. F., & López-Maldonado, J. T. (2024). Resultados Preliminares de la Creación de una Base de Datos de Flujo Óptico en Reactores Electroquímicos: Una Herramienta para el Desarrollo y Evaluación de Algoritmos de Deep Learning. JÓVENES EN LA CIENCIA, 25, 1-11. https://www.jovenesenlaciencia.ugto.mx/index.php/jovenesenlaciencia/article/view/4211

Hannover Messe. Siemens and Microsoft drive industrial productivity with generative artificial intelligence. April 12, 2023. SIEMENS DIGITAL INDUSTRIES SOFTWARE NEWSROOM. Germany. https://newsroom.sw.siemens.com/en-US/siemens-microsoft-generative-ai/

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., ... & Stoyanov, V. (Jul. 2019). Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692. https://arxiv.org/abs/1907.11692

IBM Education. AI code-generation software: What it is and how it works. Blog-Artificial intelligence. September 19, 2023. https://www.ibm.com/blog/ai-code-generation/

Knuth, D. E. (1970, September). The analysis of algorithms. In Actes du Congres International des Mathématiciens (Nice, 1970) (Vol. 3, pp. 269-274). http://homepages.cs.ncl.ac.uk/brian.randell/Seminars/139.pdf

Moncho, Terol. Inteligencia Artificial: API de OpenAI: revolucionando la industria de chatbots. ThinkBig-Telefónica. Marzo 2023. https://blogthinkbig.com/api-de-openai/

Nguyen-Duc, A., Cabrero-Daniel, B., Przybylek, A., Arora, C., Khanna, D., Herda, T., ... & Abrahamsson, P. (2023). Generative Artificial Intelligence for Software Engineering--A Research Agenda. arXiv preprint arXiv:2310.18648. URL: https://arxiv.org/abs/2310.18648

OpenAI. Introducing GPT - Generative Pre-trained Transformer. Blog - November 6, 2023. https://openai.com/blog/introducing-gpts#OpenAI

Ortolan, P. (2023). Optimizing Prompt Engineering for Improved Generative AI Content. https://repositorio.comillas.edu/xmlui/bitstream/handle/11531/80629/TFM-ORTOLAN-PABLO.pdf?sequence=1

Pérez, Luz. Descubre la API ChatGPT: Convierte tus Conversaciones. junio 7, 2023. https://neuroflash.com/es/blog/descubre-la-api-chatgpt-convierte-tus-conversaciones-en-arteificial-inteligencia/#:~:text=S%C3%AD%2C%20ChatGPT%20ofrece%20una%20API,a%20preguntas%20y%20mucho%20m%C3%A1s

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., ... & Liu, P. J. (2019). "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer." arXiv preprint arXiv:1910.10683. https://arxiv.org/abs/1910.10683

Turing, A. M., Girard, J. Y., Basch, J., & Blanchard, P. (1995). La machine de Turing (pp. 47-102). Editions du seuil. https://bfourlegnie.com/nsi_2019/cours/CHAP6/cours_architecture_turing.pdf

UNI. Órgano de Planeamiento Estratégico (OPE) - Oficina Central de Planificación y Presupuesto (OCPLA). “Plan Estratégico Institucional PEI UNI 2020 – 2025 Ampliado”. UNI, Abril 2022. https://drive.google.com/file/d/1e-ZKIQ-Sx7aSof2HiNVtxY5ybgvYxOQB/view

Wang, B., Liu, H., An, P., Li, Q., Li, K., Chen, L., ... & Gu, S. (2018). Artificial intelligence and education (pp. 129-161). Springer Singapore. https://link.springer.com/chapter/10.1007/978-981-13-2209-9_5

Bhutoria, A. (2022). Personalized education and artificial intelligence in the United States, China, and India: A systematic review using a human-in-the-loop model. Computers and Education: Artificial Intelligence, 3, 100068. https://www.sciencedirect.com/science/article/pii/S2666920X22000236

Desouza, K. C., Dawson, G. S., & Chenok, D. (2020). Designing, developing, and deploying artificial intelligence systems: Lessons from and for the public sector. Business Horizons, 63(2), 205-213. https://www.sciencedirect.com/science/article/abs/pii/S0007681319301582

Ebert, C., & Louridas, P. (2023). Generative AI for software practitioners. IEEE Software, 40(4), 30-38. https://ieeexplore.ieee.org/abstract/document/10176168

Joshi, S. C., & Joshi, Y. (2022). Prospects and Future of Artificial Intelligence (AI) in Business Strategies. Decision Intelligence Analytics and the Implementation of Strategic Business Management, 53-67. https://link.springer.com/chapter/10.1007/978-3-030-82763-2_5

Joskowicz, J., & Slomovitz, D. (2023). Engineers' Perspectives on the Use of Generative Artificial Intelligence Tools in the Workplace. IEEE Engineering Management Review. https://ieeexplore.ieee.org/abstract/document/10319661

Kankanhalli, A. (2020). Artificial intelligence and the role of researchers: Can it replace us?. Drying Technology, 38(12), 1539-1541. https://www.tandfonline.com/doi/epdf/10.1080/07373937.2020.1801562?needAccess=true

Petrovska, O., Clift, L., Moller, F., & Pearsall, R. (2024, January). Incorporating Generative AI into Software Development Education. In Proceedings of the 8th Conference on Computing Education Practice (pp. 37-40). https://dl.acm.org/doi/abs/10.1145/3633053.3633057

Raj, R., & Kos, A. (2023). Artificial Intelligence: Evolution, Developments, Applications, and Future Scope. Przeglad Elektrotechniczny, 99(2). https://openurl.ebsco.com/EPDB%3Agcd%3A9%3A6231065/detailv2?sid=ebsco%3Aplink%3Ascholar&id=ebsco%3Agcd%3A161782056&crl=c

Sun, J., Liao, Q. V., Muller, M., Agarwal, M., Houde, S., Talamadupula, K., & Weisz, J. D. (2022, March). Investigating explainability of generative AI for code through scenario-based design. In 27th International Conference on Intelligent User Interfaces (pp. 212-228). https://dl.acm.org/doi/abs/10.1145/3490099.3511119

ANEXOS.

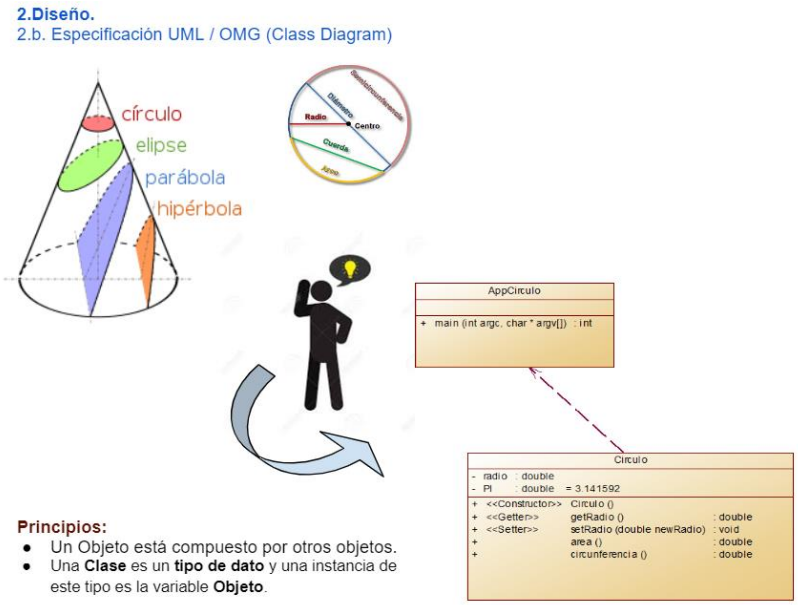
"Algorithmic Specification & Type Hints Notations for AI".

Algoritmos Imperativos	Patrones	Investigadores - Meteorologistas
Método Estructurado	Secuence, Selection, Repeat	(Bohm y Jacopini - 1960)
Método Procedimental	Procedure - Function	(Edsger W. Dijkstra - 1960 y Niklaus Wirth - 1988)
Método Orientado a Objetos	Class - Object	(Ole-Johan Dahl y Kristen Nygaard -1962)

1. Diseño de Algoritmo & Type Hints

```
Algoritmo. Ordenamiento v2
        const N <- 5
tipo
        array [1..N] de entero: vector
var
        vector: v <- {5, 4, 2, 3, 1}
        entero: i, j
        entero: aux
inicio
i <- 1
mientras(hasta N-1)hacer
j <- 1
mientras(hasta N-1)hacer
        //iteracion: comparacion
llamar_a intercambio (v[j], v[j+1])

j <- j + 1
fin_mientras
i <- i + 1
fin_mientras

//desplegar vector ordenado
llamar_a desplegarVector(v)
fin
//definicion de prototipos
procedimiento intercambio (E/S entero: a, b)
var
        entero: aux
inicio
        si(a > b)entonces
        //intercambio
        aux <- a
        a <- b
        b <- aux
        fin_si
fin_procedimiento

procedimiento desplegarVector(E vector: vec)
var
        entero: i
inicio
        i <- 1
        mientras( hasta N)hacer
        //iteracion
        escribir(vec[i])

        i <- i + 1
        fin_mientras
fin_procedimiento
```

2. Diseño Técnica UML.



3. Diseño: Orientación a Objetos.

```
Algoritmo Circulo v32
tipo
        clase Circulo
                privado: //miembros privados de datos
                real: radio
                const real: PI <- 3.141592

                publico:          //miembros publicos
                //constructor por omision
                procedimiento Circulo()
                var
                inicio
                        radio := 0
                fin_procedimiento

                //sobrecarga de constructores
                procedimiento Circulo(E real: radio)
                var
                inicio
                        setRadio(radio)
                fin_procedimiento

                procedimiento Circulo(E Circulo: c)
                var
                inicio
                        radio := c.getRadio()
                fin_procedimiento
```

```
Algoritmo AppCirculo v32
var
        //declaración de variables objeto
        Circulo: objCirculo1, objCirculo2
        real: r
inicio
        //entrada
         repetir
                escribir("Ingrese radio: ")
                leer(r)
         hasta_que(r > 0)
        //crear instancia objeto - constructor default
        objCirculo1 = Circulo()
```