INFORME PROGRAMACIÓN

Alberto Arranz Roldán.
Grado de Ciberseguridad.

Noviembre 2023

ÍNDICE

LAS REFERENCIAS IRÁN MARCADAS EN CADA PAGINA SIGUIENDO LOS ESTANDARES ESTIPULADOS

1.-INTRODUCCIÓN.

1.1.-Sobre mi trabajo.

Mi trabajo conlleva un diseño de un informe sobre la programación, sus tipos de lenguajes, paradigmas dentro de este mismo y los estándares estipulados a la hora de poder trabajar con estos criterios. Estos temas a tratar, se han tratado en clase y además tenemos información sobre ello, gracias a las clases que hemos tenido sobre este tema, podemos organizar una buena estructura para este trabajo y su posterior desarrollo y como no, como en todos los trabajos valga la redundancia, vamos a explicar todos los conceptos del índice con una buena conclusión al final del trabajo para así poder exponer nuestro razonamiento e ideas tenidas a lo largo de este trabajo.

2.-INTRODUCCIÓN.

El primer paso a la hora de la creación de un buen informe es una introducción, un breve sumario para poder dar pie a los demás temas, para poder entender mejor el trabajo y a su misma vez, poderlo llevar a cabo de una forma eficiente.

2.1.-Descripción sobre la importancia de los lenguajes, paradigmas y estándares dentro del mundo de la programación.

Programación: La programación es el desarrollo para su posterior creación de un conjunto de instrucciones que le transmitimos a un PC para realizar una tarea en específico. Pero no es solo eso, además incluye, todas las tareas necesarias para que el script/código cumpla los entandares y funcione correctamente.

Lenguajes: Los lenguajes de programación son una herramienta muy importante a la hora de crear aplicaciones[1], aplicaciones web[1], sistemas de software[1] y sistemas informáticos[1]. Permite a los programadores expresar sus ideas y soluciones en un lenguaje que el PC pueda interpretar.

Paradigmas: Los paradigmas de programación te ayudan a hacer un mejor "*development"*, con códigos más legibles y organizados. También, te ofrecen técnicas adecuadas para aumentar la productividad de los ejemplos[1] que he puesto en los lenguajes.

Estándares: También llamados estándares de código, guías de estilo o convención de código, son términos que te describen un convenio para escribir código fuente en ciertos lenguajes. Los estándares son ciertas pautas que se cogieron para obtener una uniformidad en los PCs y sistemas/equipos informáticos, facilitando le trabajo de mantenerlo *"updated"* y la reparación del *"hardware"*

3.-TIPOS DE LENGUAJES DE PROGRAMACIÓN

La evolución de los PCs ha llevado a cabo un gran crecimiento y concienciación de cientos y cientos de lenguajes para un desarrollo sostenible de estos mismos, ya sea para PCs, software, aplicaciones web, videojuegos…. La programación en si misma es muy grande, con lo cual el lenguaje a seguir será marcado dependiendo de los intereses que tenga cada uno y las funciones que le quiera dar.

3.1.-Descripción de los lenguajes de bajo, medio y alto nivel, lenguaje maquina y ensamblador.

Los lenguajes de programación se pueden clasificar principalmente como lenguajes de alto, medio y bajo nivel. Pero son muy simples comparándolos con lenguajes humanos. Los lenguajes de alto nivel son más difíciles que los medio y bajo nivel, Pero a su misma vez los de alto nivel tienen una mejor comprensión que los de bajo nivel[2], para

entender estos mismos[2] se necesita un buen conocimiento de las arquitecturas de informática.

- **Bajo nivel:** Este lenguaje es el mas cercano a el lenguaje máquina, con lo cual no proporciona ningún tipo de abstracción a la arquitectura de los PCs. Por esto mismo los lenguajes de bajo nivel su notación es cercana a el *"hardware"* suelen estar condicionados por la estructura física de los PCs, el hecho de que se llame bajo nivel no tiene nada que ver con que no sea efectivo como los demás lenguajes. Ejemplos: ("Código Binario, Lenguaje de maquina y lenguaje ensamblador.")
- **Medio nivel:** Este lenguaje su objetivo principal es una orientación a procedimientos, que estos se componen a su misma vez en procesos. Permiten el uso de funciones a nivel aritmético, pero sin embargo a nivel lógico dependen de los ensambladores. Ejemplos: (C, Basic, C++, Rust, Go….
- **Alto nivel:** Este lenguaje es el más cercano al razonamiento y comprensión de un ser humano, en lugar de un lenguaje de mas bajo nivel que va mas al ras de la arquitectura de los PCs. Estos lenguajes tienen una mayor flexibilidad. Por lo general están mas orientados a objetos, eventos o a funciones. A su misma vez estos lenguajes pueden ser compilados o interpretados. Ejemplos: ("Java, PHP, Python, JS y C#)
- **Maquina:** Este lenguaje es obtenido mediante códigos directamente interpretables por un circuito microprogramable, este lenguaje está compuesto por un conjunto de instrucciones aplicadas directamente a la máquina. Estos lenguajes, se simbolizan con 0 y 1, por eso este lenguaje sol utiliza estos signos. Esto permite las teorías booleanas y el sistema binario para su futuro desarrollo.
- **Ensamblador(asm):** Este lenguaje de programación es usualmente definido por el fabricante del ("hardware"), y estas se basan en los pasos de procesamiento, los logs del procesador y las posiciones de memoria. Su mayor trabajo es traducir ciertas sentencias del lenguaje ensamblador y transformarlo en lenguaje máquina. Fue usado en los inicios en el desarrollo del software cuando había pocos lenguajes de programación o poco desarrollados y con los recursos limitados. Ahora mismo su uso más común es usado con frecuencia en entornos académicos y de investigación.

3.2.-Ejemplos representativos de cada uno y sus usos más comunes.

En este apartado vamos a hacer una (“review”) de los ejemplos representativos de cada lenguaje de programación para poder entenderlos un poco mas y a su misma vez pondremos los usos mas comunes de cada uno de ellos para que podamos comparar con claridad y coherencia estos mismos.

BAJO NIVEL.

- **Código binario:** El código binario es una forma de representar datos o instrucciones utilizando sólo dos valores “0” y “1”, se utiliza para mostrar cualquier cosa. Hoy en día el código binario es la forma mas común de representar datos e instrucciones ya bien sea en PCs o en algún aparato tecnológico.
- Los lenguajes de bajo nivel también son, el lenguaje maquina y el ensamblador, pero no los voy a explicar ahora, los explicaré más adelante.

MEDIO NIVEL

- **C:** En sus orígenes estaba destinado a la implementación de los Sistemas como **Unix,** para esas fechas era necesario disponer de un lenguaje de programación muy eficiente. Es el lenguaje mas popular a la hora de crear software de sistemas. Es muy fácil gracias a este lenguaje de programación el dar instrucciones a la máquina, porque va a bajo nivel y se inserta directamente sin casi tener que traducirlo a Binario u otros lenguajes máquina.
- **Basic (Beginners all-purpouse symbolic instructión code):** Es un lenguaje fácil de usar para los principiantes, También es un lenguaje de propósito general, Puedes meterle adicionalmente características avanzadas, conservando siempre el lenguaje fácil para los principiantes. También gracias a su interfaz puede llegara a proporcionar claramente los errores cometidos, y por último no requiere de un conocimiento exhaustivo del (“hardware”) del PC. Este lenguaje es utilizado para crear juegos, aplicaciones para móvil o para una red que tengan protocolos HTTP-DNS-FTP-SMTP adicionalmente se puede también trabajar en bases de datos tales como MySQL ó SQLite.
- **C++:** El código fuente en C puede compilarse como C++, al principio esto se veía como una virtud, pero según pasó el tiempo fue una debilidad, porque para seguir siendo similares ha mantenido algunos errores graves del lenguaje C. Los

ejemplos más representativos, serían más o menos los mimos que C, Sistemas Operativos como Windows, Mac OS, Linux, navegadores Google….

- **Rust:** Es un lenguaje de programación compilado, general y multiparadigma y que además soporta programación funcional pura, orientada a objetos…. En 2020 fue uno de los lenguajes mas utilizados en las criptomonedas y la creación de nodos a la hora de minar criptoactivos. Es utilizado también para el back end.

ALTO NIVEL

- **Java:** Java ha marcado un hito en el desarrollo del software, gracias a que una vez esta desarrollado puede instalarse y ejecutarse en cualquier dispositivo, dejando de lado el sistema operativo, esto lo ha convertido en una herramienta esencial para todo tipo de programadores. Esta posicionado como uno de los lenguajes más populares y versátiles para la gama de aplicaciones. Sus usos mas comunes son, Videojuegos, Herramientas Didácticas, Comunicación, Aplicaciones móviles y mucho más.
- **PHP:** PHP es un lenguaje muy completo y además se popularizó muchísimo debido a que es de código abierto. Esto significa que se pueden hacer cambios dentro de el significativamente. Además de que está en constante perfeccionamiento, gracias a una comunidad de desarrolladores proactiva. Esta definido como lenguaje del lado del servidor sobre todo en aplicaciones y en sitios web. Es usado en Aplicaciones y sitios web como antes he expresado, Plugins de WordPress, Ecommerce entro muchos otros.
- **Python:** Python como PHP es un lenguaje de código abierto, orientado a objetos, fácil de interpretar y con una sintaxis que permite leerlo de manera parecida a la del inglés. Con el paso del tiempo ha ido creciendo y a su misma vez se ha ido haciendo cada vez mas flexible, gracias a ello lo adecua para las necesidades de muchas industrias. Sus usos mas comunes son, Scripting y automatización, desarrollo del software, análisis de datos, ciencias de blockchain y muchos más.

MAQUINA

- **Maquina:** El lenguaje maquina es el lenguaje principal que poseen todos los ordenadores, y es un conjunto se instrucciones codificadas normalmente en código binario las cuales generan acciones que hace la máquina. Estas combinaciones de 0 y 1 dan una instrucción que se interpreta y este proceso se

llama compilación, es leído por medio de la CPU. Se compone como he dicho antes de una secuencia de 1 y 0, es el único lenguaje leído y entendido por le "hardware", se interpreta de forma directa. Con los sistemas nuevos y debido al continuo desarrollo hemos podido descifrar este lenguaje. Su característica principal es el llevar o realizar procesos para los cuales anteriormente haya sido programado.

ENSAMBLADOR

- **Ensamblador:** Este lenguaje de programación es usualmente definido por el fabricante del ("hardware"), y estas se basan en los pasos de procesamiento, los logs del procesador y las posiciones de memoria. Su mayor trabajo es traducir ciertas sentencias del lenguaje ensamblador y transformarlo en lenguaje máquina. Fue usado en los inicios en el desarrollo del software cuando había pocos lenguajes de programación o poco desarrollados y con los recursos limitados. Ahora mismo su uso más común es usado con frecuencia en entornos académicos y de investigación.

4.-PARADIGMAS DE PROGRAMACIÓN

"Se trata de un conjunto de métodos sistemáticos aplicables en todos los niveles del diseño de programas para resolver problemas computacionales.
Los lenguajes de programación adoptan uno o varios paradigmas en función del tipo de órdenes que permiten implementar como, por ejemplo, Python o JavaScript, que son multiparadigmas."

4.1.-Descripción detallada de los principales paradigmas.

En este apartado vamos a hablar de los principales paradigmas de programación, y eso viene a ser una manera o estilo de programación de software, es un conjunto de métodos aplicables en el diseño de programas para resolver una o varias anomalías computacionales claramente delimitadas. Los lenguajes de programación pueden tener uno o muchos paradigmas dependiendo de el tipo de orden que le des, por ejemplo; Python o JS, son multiparadigmas.

1. **Paradigma Imperativo:** Este paradigma, como bien dice su nombre manda, *Imperar=Mandar*, claramente ordena a la computadora lo que el programador quiera realizar. Un cómputo consiste en una serie de sentencias, ejecutadas según un control de flujo explícito, y modifican el estado del programa. Las

variables son “cárceles” de memoria, que contienen datos y pueden ser modificadas, representan el estado del programa. La sentencia principal es la asignación. Los programas imperativos, realizan normalmente una tarea ejecutando un loop. Ejemplos de lenguajes de programación imperativa; COBOL, Fortran-77, Pascal y C.

2. **Paradigma Declarativo:** No define algoritmos, porque describe el problema, o sea que por ende tampoco encuentra una solución como tal. Este paradigma utiliza el razonamiento lógico para poder expresarse de manera lógica a las preguntas consultadas. Según diversas fuentes de información se puede dividir en 2:
    a. **Programación Lógica:** Prolog.
    b. **Programación Funcional:** Lisp, Scala, Java y Kotlin.

3. **Paradigma Orientado a objetos:** En sí, el paradigma orientado a objetos es una representación literal de la realidad. Un conjunto de elementos que trabajan todos juntos para la resolución de una errata. Es un enfoque diferente al momento de obtener respuestas computacionales. El lenguaje de programación orientada a objetos es robusto, flexible y fácil de mantener. Gracias a este paradigma y en si a POO nos descentramos de la lógica pura y empezamos a pensar en objetos y nos ayuda mucho en sistemas de gran volumen. Ejemplos de POO; Java, C++, Python, C#.
4. **Paradigma Funcional:** Este paradigma, nos ayuda a crear un software mas limpio y mantenible. Mas concretamente son una serie de enfoques que suelen describirse como un paradigma de programación. La programación funcional a veces es lo contrario que la POO y la procedimental. Es una de las tendencias actuales del software. Es muy útil y potente. Ejemplos del paradigma funcional puros; HasKell y Miranda.
5. **Paradigma Lógico:** Su creación y desarrollo es debido a la necesidad de las nuevas tecnologías en concreto a la de la inteligencia artificial. Utiliza la lógica matemática debido a que es lo mas cercano a el ser humano, de realizar ciertas tareas ya sean problemas y su correspondiente resolución. Este paradigma esta indirectamente conectado con la programación Prolog. Ejemplos de lenguajes que hacen uso del paradigma lógico; Prolog, Mercury y ALF.

6. **Paradigma orientado a eventos:** La POE (Programación orientada a eventos), es muy fácil de usar sobre todo para personas con un conocimiento poco extenso o nulo de la programación. Gracias a estos lenguajes se pueden hacer aplicaciones sencillas y con mucha funcionalidad. Evento: Al iniciarse un evento se da inicio a un conjunto de acciones preprogramadas por el usuario para ese evento. Ejemplos del paradigma orientado a eventos: Editor de texto, Sistemas de gestión, Servidor web, Detección de fraude.

4.2.-Lenguajes representativos de cada paradigma y sus características distintivas.

En este apartado hablaremos de los ejemplos representativos explicados arriba, y sus características mas distintivas, estaremos de una manera indirecta contrastando información para hacer una alegación sobre que lenguaje seria mejor para usar en cada paradigma. Hablaremos de las diferencias de unos a otros y detallaremos que usos se les puede dar a cada uno.

PARADIGMA IMPERATIVO

I. **COBOL (Common Bussines-Oriented Language):** Se dice que este lenguaje de programación gobierna el mundo porque está detrás de las acciones de un cajero automático, gestiona los seguros y también muy interesante el cálculo de las pensiones. Lo malo es que no es muy atractivo y hay pocos informáticos que le saquen el rendimiento al 100%. Está muy basado en el idioma (Ingles), y ha ayudado a muchas entidades para construir aplicaciones y negocios. Es un lenguaje “relativamente” fácil de adquirir y se utiliza en la Seguridad Social a nivel español.

II. **FORTRAN (Formula Translating System):** Es un lenguaje de alto nivel e imperativo y esta adaptado al calculo numérico. Fue desarrollado por IBM y es uno de los lenguajes mas populares para la computación de alto rendimiento. Su sintaxis fue orientada para el uso de proyectos matemáticos. Por estas razones Fortran, no se utiliza fuera del ámbito matemático. Según avanzó fue mas portable y con una sintaxis más intuitiva.

III. **PASCAL (.pas\.pp):** Al principio se quería utilizar como un lenguaje a nivel didáctico sin embargo con el tiempo se fue extendiendo en una herramienta para creación de herramientas de todo tipo. Es un lenguaje de fuerte tipado, significa que es muy legible. Una característica importante es que el tipo de

variable se fija en su definición. Esto previene erratas donde las variables son usadas incorrectamente.

IV. **C:** Sus principales características son una programación clara y estructurada, producir el código altamente optimizado, codifica en alto y bajo nivel simultáneamente, no está orientado a ningún área en especial y a su misma vez es de un "fácil" aprendizaje. Es un lenguaje muy eficaz y a su misma vez muy efectivo, y esta muy bien organizado en cuanto a tamaño para una mayor ejecución.

PARADIGMA DECLARATIVO.

I. **LISP (List Processor):** Es un lenguaje de programación multiparadigma, en un inicio fue creado como una notación matemática, pero en la actualidad se utiliza para la investigación de las IA, su código fuente está hecho por listas como bien indica su nombre. Es uno de los lenguajes de programación mas "simples" que existen en 7-8 conceptos ya abracas todo el lenguaje.

II. **JAVA:** Es un lenguaje de medio-bajo aprendizaje no es muy difícil pero tampoco es fácil. Esta orientado a objetos, con un fuerte tipado estáticamente y muy robusto. Al parecer es también bastante seguro y tiene una arquitectura neutral, también es portable, dinámico y de alto rendimiento.

III. **KOTLIN(.kt\.kts\.ktm):** Es un lenguaje multiplataforma, y de nivel alto. Esta diseñado para ser totalmente interoperable con Java. Y tiene una sintaxis mucho mas sencilla que la de java. Tiene una curva de aprendizaje sencilla con un menos tiempo de programación que muchos leguajes. Puede estar orientado a objetos y programación funcional, también tiene desarrollo multiplataforma con una alta flexibilidad.

PARADIGMA ORIENTADO A OBJETOS.

I. **JAVA:** Es un lenguaje de medio-bajo aprendizaje no es muy difícil pero tampoco es fácil. Y orientado a objetos, con un fuerte tipado estáticamente y muy robusto. Al parecer es también bastante seguro y tiene una arquitectura neutral, también es portable, dinámico y de alto rendimiento.

II. **Python:** Es un lenguaje de alto nivel, y es de fácil legibilidad, se utiliza para desarrollo web como Spotify…. A su misma vez es un lenguaje multiparadigma. Es un lenguaje interpretado, traduce nuestro código a raíz de nuestras necesidades. Es también de tipado dinámico y por último es un lenguaje muy bueno debido a que es multiplataforma, eso significa que se puede ejecutar en cualquier OS.

III. **C#:** Es una evolución de los lenguajes nombrados anteriormente C y C++, y añadiendo más funcionalidades hasta alguna de java, aunque se orientó a objetos y con el paso del tiempo se adaptó poco a poco a todas las facilidades que te daban otros lenguajes, viene a ser un lenguaje reciclado de muchos otros lenguajes para poder lograr un desempeño y una buena funcionalidad. Tiene muy buenas características debido a su sencillez, modernidad, seguridad, extensibilidad como ya he nombrado antes es muy compatible y de los más eficientes y por último es muy seguro debido a su control de acceso a tipos de datos.

PARADIGMAS FUNCIONALES.

I. **HassKell^3:** Es un lenguaje funcionalmente puro, es polimórfico y de tipado estático, está diseñado para generar una amplia gama de aplicaciones, es también declarativo. Tiene muy buena sintaxis y con una buena variedad de tipos primitivos Int\Char\Bool\Float. Es utilizado en programas concisos con buen sistema de tipos y muy concurrente, se caracteriza sobre todo porque se pasan las funciones sin evaluar (lazy evaluation),.

IV. **Miranda:** El objetivo de Miranda es el desarrollo de una versión comercial de un lenguaje funcional. Es muy parecido a HassKell^3, se caracteriza sobre todo porque se pasan las funciones sin evaluar (lazy evaluation), tiene también una buena recursión (loop). Este lenguaje suele ser corrrido por el sistema operativo UNIX. El usuario define los tipos de los datos y no hay ninguna declaración del tipo obligatoria.

PARADIGMA LÓGICO.

I. **Mercury:** Es un lenguaje de programación lógico-funcional, también es un lenguaje puramente declarativo porque ni tiene declaraciones y permite optimizar mejor el programa. Se parece a HassKell[3] pero soporta tipificación dinámica, a diferencia de Prolog\Hasskell tiene una fase de compilación separada, en lugar de ser interpretado directamente. Como dato de interés, los programadores de Mercury afirman que es el lenguaje lógico más rápido del mundo con ventaja a los demás.

II. **ALF (Another Logical Framework):** Es in lenguaje de alto nivel, tiene una sintaxis "sencilla", similar a la inglesa, y eso facilita la recitación y redacción del código. Puede incorporar varios tipos de datos y a su misma vez los definidos por el usuario.

5.-ESTÁNDARES DE PROGRAMACIÓN.

Cada lenguaje de programación tiene sus propios estándares, son formas de programar en determinados lenguajes. Los estándares son parte del lenguaje en sí. Son los comandos básicos utilizados para poder programar, variables para poder hacer funciones, condiciones…. Están presentes en muchos de los lenguajes y son utilizados por estos mismos.

5.1.-Introducción a la importancia de seguir estándares.

Los estándares están muy marcados hasta en nuestra vida cotidiana, son importantes para mantener un cierto orden en las cosas, hay ciertas técnicas muy efectivas de manejo y control de proyectos combinados con una participación activa de los usuarios para minimizar el riesgo de incumplimiento de fechas de actividades importantes, de gastos excesivos e insatisfacción de los usuarios de los sistemas. No todas las empresas utilizan las mismas herramientas en sus desarrollos y es muy difícil encontrar un estándar idead adaptable a las necesidades específicas de cada empresa, con lo cual es recomendable establecer un grupo de estándares propios para poder lograr un buen código de calidad.

5.2.-Descripción de algunos estándares de programación populares y sus características principales.

Los estándares como ya he hablado antes son una parte muy importante a la hora de hacer un código o utilizar un concreto lenguaje de programación, y estas podrían ser algunas de las razones.

1.- El 70-80% del coste de un código es debido a su mantenimiento.

2.- Si se extiende el código fuente de un producto tenemos que concienciarnos de que está bien hecho.

3.- Cada participante de un proyecto piensa de forma distinta o sea que, fijas estándares te ayudan al entendimiento mutuo.

Debido a que estamos en un corso de ciberseguridad, voy a explicar algunos de los estándares de seguridad y protección populares en estos momentos que son:

- **MISRA[4] (Motor-Industry-Software-Reliability-Association) C/C++:** Es un subconjunto de C y C++, su uso genera una mejor seguridad de la aplicación, aunque anteriormente se utilizaba para aplicaciones automotrices, se usa en todas las industrias y particularmente para aplicaciones críticas de seguridad.
- **SEI/SANS CERT:** Estas instituciones se centran como el **MISRA[4],** en la seguridad e integridad de las aplicaciones. Estas pautas se forman en, la importancia de las “reglas”, “recomendaciones” y son muy completas y amplias e incluyen hasta metadatos de riesgo.
- **OWASP Top 10(Open Web Application Security Project):** Es una organización que le proporcionan seguridad a a las aplicaciones web. Este proyecto proporciona una lista de vulnerabilidades de seguridad de aplicaciones mas comunes y conocidas.
- **Código Limpio:** Una buena estrategia es buscar que todo el equipo utilice ciertos principios para mantener el código más limpio y a su misma vez que se aclaren entre ellos y puedan hacer consultas de una manera óptima. Ejemplos de estos mismo serían: DRY, YAGNI, KISS, SOLID….

5.3.-Beneficio de adherirse a estándares y consecuencias de no hacerlo.

El uso de los estándares de programación, proporcionan, muchos beneficios, es mucho mejor tener ciertos estándares establecidos para lograr un trabajo mas limpio y con muchas mas facilidades a la hora de tener que corregirlo o reciclarlo para otras funciones, algunas de las razones porque deberíamos adherirnos son:

1) **Detección temprana de fallas:** Gracias a cumplirlos podemos detectar errores revisando el código evitando de que esos errores lleguen a la producción, y así poder corregirlos.
2) **Reducción de la complejidad:** Cumpliendo estas reglas, ayudas a construir código más limpio y claro, permitiendo encontrar fácilmente oportunidades para simplificar.
3) **Código de fácil lectura:** Respetar los estándares en un proyecto le permite a nuevos miembros del equipo acoplarse más fácilmente a un trabajo eficiente y a entender mejor el código de los proyectos.
4) **Código reusable:** Contar con partes del código que pueden ser utilizados por más de un servicio, gracias al uso de buenas prácticas, hace que haya menos repetición de código.

6.- CONCLUSIÓN.

6.- Reflexión sobre la importancia de entender los diferentes lenguajes, paradigmas y estándares y cómo estos influyen en el desarrollo de software.

Mi reflexión sobre la importancia de los lenguajes de programación es, que los lenguajes de programación nos ofrecen unas cualidades específicas para cada trabajo que queramos realizar, además que nos permite usar diferentes lenguajes más al ras del nivel computacional o mas al el nivel de lenguaje del ser humano, creo que es muy importante conocer los tipos de lenguajes adyacentes ya que dependiendo de las instituciones, tipo de trabajo realizado u otros aspectos, necesitamos tener una idea clara de estos mismos para poderlos llegar a entender. Mi reflexión sobre los paradigmas de programación, son que no podríamos resolver estos problemas para poder llevarlos a un mejor uso, dependiendo de los lenguajes, podríamos decir que tienen un paradigma o pueden llegar a ser multiparadigmas que es muy útil a la hora de realizar cualquier actividad, gracias a que serían multifacéticos.

Me reflexión sobre los estándares de programación y aun mas los nominados en este trabajo son que gracias a ellos podremos tener una buena seguridad y una buena práctica de estos mismos. Gracias a todos estos elementos podemos hacer un desarrollo de software de manera productiva, deductiva y a su misma vez segura, creo que es muy importante la implementación de todos estos lenguajes para poder llevar a cabo todas las actividades relacionadas con el software creo que debería de ser obligatorio el uso de todo lo nombrado anterior y que se pusiese en practica en todos los ámbitos relacionados con el software y creación de código.

7.-REFERÉNCIAS.

Referencia.

Profile.es, 2020. *¿Que son los paradigmas de programación?,* s.l.: s.n.

Unknown, 2020. *profile.es.* [En línea]
Available at: https://profile.es/blog/que-son-los-paradigmas-de-programacion/
[Último acceso: 29 11 2023].