TRABAJO DE INVESTIGACIÓN CIENCIA DE DATOS

Juan Pablo Oviedo

Eddie Alejandro Arenas

Esneider Flórez

Jhon Jairo Moguea

ADSO - 2828523

**1. Entorno Personal de Aprendizaje (EPA) o Personal Learning Environment (** **PLE)**

Es el conjunto de herramientas, recursos, estrategias y conexiones que una persona usa para gestionar su aprendizaje de forma autónoma y continua.

**Características:**

**Personalizado:** Se adapta a los intereses y necesidades del usuario.

**Autónomo:** El aprendiz decide qué, cómo y cuándo aprender.

**Tecnológico:** Utiliza herramientas digitales como redes sociales, blogs y plataformas educativas.

**Colaborativo:** Fomenta la interacción con expertos y comunidades de aprendizaje.

Flexible y dinámico: Evoluciona con el tiempo según los objetivos personales.

**¿Cómo diseñar un EPA?**

**Definir objetivos de aprendizaje:** Identificar qué temas o habilidades se quieren desarrollar.

Seleccionar fuentes de información: Blogs, libros, podcasts, videos, cursos en línea, etc.

**Elegir herramientas digitales:** Plataformas como YouTube, Coursera, Udemy, Google Drive, Trello o Notion.

**Construir una red de contactos:** Seguir expertos en LinkedIn, participar en foros y comunidades en redes sociales.

**Organizar y reflexionar:** Usar herramientas de gestión (como Evernote o Notion) y evaluar constantemente el progreso. Aplicar el conocimiento: Practicar lo aprendido mediante proyectos, blogs o participación en comunidades. Ejemplos de EPA en la práctica:

**Estudiante de desarrollo de software:** Usa YouTube para tutoriales, GitHub para practicar código y Stack Overflow para resolver dudas.

**Emprendedor:** Aprende sobre marketing digital en Udemy, sigue a expertos en Twitter y organiza su aprendizaje en Notion.

**Docente**: Crea un blog para compartir conocimientos, usa Moodle para gestionar cursos y asiste a webinars educativos.

1. **Métodos de aprendizaje.**

Los métodos de aprendizaje son enfoques o técnicas utilizadas para facilitar la adquisición y comprensión de conocimientos, habilidades, actitudes y valores. Cada método de aprendizaje se basa en diferentes principios y teorías educativas, y se adapta a las necesidades, preferencias y estilos de aprendizaje de los individuos.

**Aprendizaje Visual**

Se basa en el uso de imágenes, gráficos, diagramas y otras herramientas visuales para facilitar la comprensión.

Ejemplo: Un estudiante que usa mapas conceptuales y diagramas para estudiar temas de biología.

**Aprendizaje Auditivo**

Se enfoca en la escucha activa. Los estudiantes aprenden mejor a través de conferencias, discusiones y grabaciones de audio.

Ejemplo: Un estudiante que escucha podcasts educativos o graba y repasa las clases para estudiar historia.

**Aprendizaje Kinestésico**

Implica el uso del cuerpo y el movimiento. Los estudiantes aprenden mejor a través de actividades prácticas y experimentación.

Ejemplo: Un estudiante de química que participa en experimentos de laboratorio para entender las reacciones químicas.

**Aprendizaje Lector/Escritor**

Se basa en la lectura y la escritura. Los estudiantes aprenden mejor mediante la toma de notas y la lectura de libros y artículos.

Ejemplo: Un estudiante que escribe resúmenes y lee libros de texto para preparar sus exámenes de literatura.

**Aprendizaje Cooperativo**

Involucra el trabajo en grupo y la colaboración. Los estudiantes aprenden mejor al interactuar con sus compañeros.

Ejemplo: Un grupo de estudiantes que trabaja en un proyecto de investigación en equipo, compartiendo ideas y dividiendo tareas.

**Aprendizaje Autónomo**

Se centra en la auto-motivación y la autodisciplina. Los estudiantes establecen sus propios objetivos y gestionan su propio tiempo de estudio.

Ejemplo: Un estudiante que usa aplicaciones de aprendizaje en línea para aprender un nuevo idioma por su cuenta.

**Aprendizaje Basado en Problemas (ABP)**

Los estudiantes aprenden resolviendo problemas reales y aplicando conocimientos teóricos a situaciones prácticas.

Ejemplo: Un estudiante de ingeniería que trabaja en un proyecto para diseñar una solución a un problema técnico en su comunidad.

1. **Tipos de aprendizaje.**

**Aprendizaje Implícito**

Es un tipo de aprendizaje no consciente que ocurre sin intención directa, a menudo a través de la repetición y la exposición.

Ejemplo: Un niño aprende a hablar su idioma materno imitando y escuchando a sus padres y otras personas en su entorno.

**Aprendizaje Explícito**

Involucra una intención consciente de aprender, con atención y esfuerzo dirigido hacia la adquisición de conocimientos específicos.

Ejemplo: Un estudiante que estudia y memoriza fechas importantes de la historia para un examen.

**Aprendizaje Asociativo**

Se basa en la formación de asociaciones entre estímulos y respuestas, comúnmente conocido a través de los principios del condicionamiento clásico y operante.

Ejemplo: Un perro que aprende a salivar cuando escucha la campana porque asocia el sonido con la comida (experimento de Pavlov).

**Aprendizaje No Asociativo**

Ocurre cuando la respuesta a un estímulo cambia sin que se forme una asociación entre dos estímulos o entre un estímulo y una respuesta.

Ejemplo: Una persona que se acostumbra a un ruido constante de fondo, como el sonido del tráfico, y deja de notarlo (habituación).

**Aprendizaje Cooperativo**

Implica trabajar juntos en grupo para alcanzar objetivos comunes, aprovechando las fortalezas individuales de cada miembro.

Ejemplo: Un grupo de estudiantes que colabora en un proyecto de ciencias, repartiendo tareas y compartiendo conocimientos para completar el trabajo.

**Aprendizaje Colaborativo**

Similar al aprendizaje cooperativo, pero se enfoca más en la construcción conjunta de conocimiento, con un énfasis en el diálogo y la interacción entre los participantes.

Ejemplo: Estudiantes que trabajan juntos en una discusión en línea, intercambiando ideas y desarrollando una comprensión colectiva de un tema.

**Aprendizaje Emocional**

Involucra la comprensión y gestión de las emociones, tanto propias como ajenas, y su impacto en el aprendizaje y la interacción social.

Ejemplo: Un niño que aprende a identificar y regular sus emociones a través de actividades de inteligencia emocional en la escuela.

**Aprendizaje Experiencial**

Se basa en aprender a través de la experiencia directa y la reflexión sobre esas experiencias.

Ejemplo: Un estudiante de biología que participa en una excursión al campo para estudiar ecosistemas y reflexiona sobre lo aprendido al regresar.

1. **Qué es una base de datos.**

Una base de datos es una recopilación de información que se almacena digitalmente en una computadora, en un servidor o en la nube. Las bases de datos pueden ser muy simples o bastante complicadas, dependiendo de la estructura y organización de los datos. El sistema de bases de datos generalmente se compone de la información en sí y un sistema de gestión de bases de datos (DBMS), lo que permite a los usuarios acceder, actualizar, analizar y gestionar fácilmente la información.

1. **Tipos de bases de datos.**

**Bases de datos relacionales:**

Es el tipo más común y tradicional de base de datos. organiza los datos en tablas relacionales formadas por filas y columnas. Los datos pueden incluir listas de clientes con información de contacto, registros de inventario, registros de ventas, información financiera y mucho más. Las bases de datos relacionales se utilizan para dar soporte a muchos tipos de aplicaciones, incluyendo el comercio electrónico, seguimiento de inventario, aplicaciones de gestión de relaciones con los clientes y muchas más.

Prácticamente todas las bases de datos relacionales utilizan lenguaje de consulta estructurado (SQL) para agregar, actualizar, consultar y eliminar datos almacenados en una base de datos relacional.

**Bases de datos no relacionales o NoSQL**

Las bases de datos no relacionales se diferencian de las bases de datos relacionales en que organizan los datos en un formato no tabular y utilizan métodos distintos de SQL para manipular los datos, lo que las hace propicias para una amplia variedad de casos de uso. Sin embargo, existen inconvenientes en las bases de datos no relacionales, incluyendo el hecho de que algunas carecen de soporte para transacciones ACID.

**7. Marcas y fabricantes de gestores de bases de datos.**

Existen varias marcas y fabricantes que desarrollan gestores de bases de datos (DBMS), cada uno con diferentes características, ventajas y aplicaciones.

* **ORACLE DATABASE**

**Fabricante:** Oracle Corporation.

**Tipo:** Relacional (RDBMS).

**Detalles:** Oracle es uno de los líderes más conocidos en el mercado de bases de datos relacionales. Su DBMS está orientado a grandes empresas, ofreciendo alta escalabilidad, rendimiento y seguridad. Es ampliamente utilizado en aplicaciones empresariales críticas. Oracle Database soporta SQL y PL/SQL (su propio lenguaje de programación).

**Características:**

* + Alta disponibilidad y recuperación ante desastres.
  + Soporte para bases de datos en la nube.
  + Seguridad avanzada (encriptación y auditoría).
  + Soporte de bases de datos híbridas (relacionales y no relacionales).
* **MICROSOFT SQL SERVER**

**Fabricante:** Microsoft.

**Tipo:** Relacional (RDBMS).

**Detalles:** SQL Server es un RDBMS ampliamente utilizado en entornos corporativos y comerciales. Ofrece herramientas integradas de análisis, inteligencia empresarial, integración de datos y desarrollo. Compatible con SQL, proporciona una plataforma robusta para gestionar grandes volúmenes de datos.

**Características:**

* + Soporta almacenamiento de datos estructurados y no estructurados.
  + Alta integración con otras tecnologías Microsoft (como Azure y Power BI).
  + Funciones avanzadas de seguridad, como encriptación y autenticación multifactor.
* **MySQL**

**Fabricante:** Oracle Corporation (anteriormente Sun Microsystems).

**Tipo:** Relacional (RDBMS).

**Detalles:** MySQL es uno de los DBMS más populares del mundo, especialmente en aplicaciones web. Es una opción de código abierto con un enfoque en la simplicidad y la alta velocidad. Utiliza SQL para la gestión de bases de datos y es conocido por su alta eficiencia y baja carga en el sistema.

**Características:**

* + Open source.
  + Soporte para transacciones ACID.
  + Compatible con muchas plataformas.
  + Integración con aplicaciones web como WordPress, Joomla y Drupal.
* **POSTGRE SQL**

**Fabricante:** Comunidad de PostgreSQL.

**Tipo:** Relacional (RDBMS).

**Detalles:** PostgreSQL es una base de datos objeto-relacional, también open source, que soporta una amplia variedad de tipos de datos y operaciones. Es conocido por su conformidad con los estándares SQL y su capacidad para manejar grandes volúmenes de datos. Es una excelente opción para aplicaciones que requieren un modelo de datos complejo.

**Características:**

* + Soporta JSON y consultas complejas.
  + Extensible y personalizable.
  + Potente en términos de integridad referencial y transacciones.
* **MONGO DB**

**Fabricante:** MongoDB, Inc.

**Tipo:** No Relacional (NoSQL).

**Detalles:** MongoDB es una de las bases de datos NoSQL más populares y es conocida por su flexibilidad y escalabilidad. A diferencia de las bases de datos relacionales, utiliza un modelo de documentos (basado en BSON), lo que permite almacenar datos de manera más flexible.

**Características:**

* + Base de datos orientada a documentos.
  + Escalabilidad horizontal (sharding).
  + Ideal para aplicaciones que manejan grandes volúmenes de datos no estructurados.
  + Soporta consultas complejas con filtros avanzados.
* **CASSANDRA**

**Fabricante:** Apache Software Foundation.

**Tipo:** No Relacional (NoSQL).

**Detalles:** Cassandra es una base de datos distribuida NoSQL diseñada para manejar grandes cantidades de datos en una arquitectura distribuida sin un punto único de fallo. Es ideal para aplicaciones que necesitan disponibilidad y escalabilidad global.

**Características:**

* + Escalabilidad lineal y alta disponibilidad.
  + Almacenamiento distribuido y tolerancia a fallos.
  + Ideal para grandes volúmenes de datos en tiempo real.
* **SQ LITE**

**Fabricante:** D. Richard Hipp (creador original).

**Tipo:** Relacional (RDBMS).

**Detalles:** SQLite es una base de datos ligera que se utiliza principalmente en aplicaciones móviles, embebidas o de escritorio. A diferencia de otros DBMS, no tiene un servidor independiente, sino que se integra directamente en las aplicaciones. Ideal para proyectos más pequeños o con recursos limitados.

**Características:**

* + Embebida directamente en aplicaciones.
  + Muy ligera y fácil de integrar.
  + Base de datos transaccional, soporta ACID.
  + No requiere configuración ni administración.
* **REDIS**

**Fabricante:** Redis Labs.

**Tipo:** No Relacional (NoSQL).

**Detalles:** Redis es una base de datos en memoria que se utiliza para almacenamiento de estructuras de datos avanzadas como listas, conjuntos, mapas y otros tipos. Es popular para cachés de alto rendimiento y almacenamiento de datos temporales.

**Características:**

* + Almacenamiento en memoria, extremadamente rápido.
  + Soporta estructuras de datos complejas.
  + Ideal para sistemas de caché y mensajería en tiempo real.
  + Persistencia opcional (puede ser configurada).
* **AMAZON AURORA**

**Fabricante:** Amazon Web Services (AWS).

**Tipo:** Relacional (RDBMS).

**Detalles:** Amazon Aurora es una base de datos relacional en la nube que es compatible con MySQL y PostgreSQL, pero ofrece un rendimiento y escalabilidad significativamente mayores. Está diseñada para aplicaciones en la nube que necesitan alta disponibilidad y recuperación ante desastres.

**Características:**

* + Totalmente administrada por AWS.
  + Compatible con MySQL y PostgreSQL.
  + Alta disponibilidad, escalabilidad automática.
  + Integración con otros servicios de AWS.
* **ELASTICSEARCH**

**Fabricante**: Elastic NV.

**Tipo:** No Relacional (NoSQL, motor de búsqueda).

**Detalles:** Elasticsearch es un motor de búsqueda y análisis de texto, ampliamente utilizado para manejar y explorar grandes volúmenes de datos no estructurados. Es muy popular en aplicaciones de búsqueda y análisis en tiempo real.

**Características:**

* + Índices invertidos para búsquedas rápidas.
  + Escalabilidad y distribución automática.
  + Ideal para log analytics y análisis de grandes volúmenes de datos.
* **IBM DB2**

**Fabricante:** IBM.

**Tipo:** Relacional (RDBMS).

**Detalles:** Db2 es una base de datos relacional de IBM que se utiliza en entornos de misión crítica. Ofrece capacidades avanzadas para procesar grandes volúmenes de datos tanto en sistemas locales como en la nube.

**Características:**

* + Alta disponibilidad y recuperación ante desastres.
  + Integración con herramientas de análisis e inteligencia artificial.
  + Compatible con SQL y JSON.
* **COUCHBASE**

**Fabricante:** Couchbase, Inc.

**Tipo:** No Relacional (NoSQL).

**Detalles:** Couchbase es una base de datos NoSQL que ofrece un modelo de documento flexible y está diseñada para aplicaciones web y móviles de alta disponibilidad y baja latencia.

**Características:**

* + Soporta almacenamiento en memoria y persistente.
  + Escalabilidad horizontal.
  + Ideal para aplicaciones de alto rendimiento y en tiempo real.

1. **Que es Python, historia y documentación. Historia de Python**

Python fue creado por Guido van Rossum, un programador holandés, en la década de 1980. Su desarrollo comenzó en diciembre de 1989 como un proyecto personal mientras Guido trabajaba en el Centro de Investigación de Matemáticas y Computación de los Países Bajos (CWI). A continuación, te doy un resumen de los momentos clave en la historia de Python:

* 1989 - Creación de Python Guido van Rossum comenzó a trabajar en Python en diciembre de 1989, inspirado en su necesidad de un lenguaje de programación interpretado que fuera fácil de entender y con una sintaxis legible.
* Python fue diseñado como una mejora sobre ABC, otro lenguaje de programación creado en el CWI. Aunque ABC fue innovador en su época, no tuvo mucho éxito debido a su enfoque en una interfaz de usuario sencilla pero limitada. Python se inspiró en ABC, pero intentó corregir sus deficiencias y ofrecer más flexibilidad.
* 1991 - Primer Lanzamiento (Python 0.9.0) Guido van Rossum lanzó la versión 0.9.0 de Python en febrero de 1991. Esta versión ya incluía características que se encuentran en Python hoy en día, como manejo de excepciones, funciones, y el sistema de módulos.
* 1994 - Python 1.0
* En enero de 1994, se lanzó Python 1.0. Esta versión ya contenía muchas características clave que definirían al lenguaje: Manejo de excepciones (con try y except). Módulos: Un sistema que permite la reutilización de código. Tipos de datos como listas, diccionarios y cadenas.
* 2000 - Python 2.0
* En octubre de 2000, Python 2.0 fue lanzado. Esta versión introdujo mejoras significativas como: List comprehensions: Una forma más concisa de crear listas. Nuevo sistema de recolección de basura (garbage collection) para la gestión automática de memoria. Módulos adicionales y nuevas características para la creación de aplicaciones más complejas.
* Aunque Python 2.x continuó siendo ampliamente utilizado, las versiones de Python 2 llegaron a su fin con Python 2.7 en 2010.
* 2008 - Python 3.0
* En diciembre de 2008, Python 3.0 fue lanzado. Esta nueva versión fue incompatible con versiones anteriores, lo que significaba que los programas escritos en Python 2 no funcionaban directamente en Python 3 sin modificaciones. Algunas mejoras clave de Python 3 fueron: Mejoras en la sintaxis: Por ejemplo, print pasó de ser una declaración a una función (print()). Mejor manejo de cadenas de texto: Python 3 hace una distinción más clara entre cadenas de texto en ASCII y Unicode. Funciones como iteradores: En lugar de devolver listas en algunas funciones estándar, Python 3 devuelve iteradores, lo que mejora la eficiencia y el uso de memoria.
* 2010 - Fin del soporte para Python 2.x Python 2.7, la última versión de la serie Python 2, recibió soporte hasta 2020. Esto significó que la comunidad empezó a centrarse en Python 3, y la migración de proyectos de Python 2 a Python 3 fue una prioridad.
* 2020 - Python 3 en su madurez
* Con el fin del soporte de Python 2 en 2020, Python 3 se consolidó como el estándar. Esta versión es ahora la recomendada para el desarrollo de proyectos nuevos, y las principales bibliotecas y marcos de desarrollo ya están completamente adaptados a Python 3.
* Python 3.x ha continuado evolucionando, con versiones mejoradas como Python 3.7, 3.8, 3.9 y 3.10 que han introducido nuevas características y mejoras en el rendimiento, la seguridad y la productividad.

La documentación de Python es un conjunto de recursos que describen cómo usar el lenguaje de programación Python, sus bibliotecas estándar, su sintaxis, sus funcionalidades y su comportamiento. Está diseñada para ayudar a los programadores a comprender cómo trabajar con Python de manera efectiva, resolver problemas y aprender nuevas características del lenguaje.

* Tipos de documentación de Python:

Documentación oficial de Python: Es el recurso principal y más detallado sobre Python. Está organizada de manera que cubre todos los aspectos del lenguaje y sus bibliotecas estándar.

Incluye: Tutoriales: Explican conceptos básicos y avanzados del lenguaje, ideales para principiantes. Guías de referencia: Descripciones detalladas de cada aspecto del lenguaje, como tipos de datos, operadores, control de flujo, etc. Biblioteca estándar: Detalles sobre los módulos y paquetes que vienen con Python, como os, sys, math, etc. \*Guías de estilo: Como la famosa PEP 8, que establece buenas prácticas de codificación en Python. Acceso: [Documentación oficial de Python](https://docs.python.org/3/).

PEPs (Python Enhancement Proposals): Son documentos que proponen nuevas características o mejoras para Python. Las PEPs describen tanto las ideas como los procedimientos técnicos para implementar esas mejoras.

PEP 8 (guía de estilo de Python) y PEP 20 (Zen de Python) son ejemplos populares de PEPs.

Acceso: [PEPs en Python](https://peps.python.org/).

Tutoriales y ejemplos de uso: A menudo, la documentación incluye ejemplos prácticos de cómo utilizar las diferentes funcionalidades de Python en proyectos reales. Estos ejemplos ayudan a los programadores a comprender cómo aplicar el lenguaje en escenarios específicos.

Notas de la versión: Documentación sobre las nuevas características, correcciones de errores y mejoras que se introducen en cada versión de Python. Las notas de la versión son esenciales para mantenerse actualizado con los cambios que ocurren en el lenguaje.

Documentación de módulos de terceros: Python tiene una vasta comunidad que crea bibliotecas y paquetes adicionales. Cada uno de estos módulos tiene su propia documentación, generalmente accesible en sus respectivos repositorios (por ejemplo, en GitHub) o en sitios como ReadTheDocs.

En resumen, la documentación de Python está compuesta por una combinación de recursos oficiales (guías, tutoriales, PEPs) y comunitarios (documentación de bibliotecas de terceros). Todo ello está diseñado para ayudar a los desarrolladores a comprender y utilizar Python de manera efectiva en sus proyectos.

1. **Que es Replit y GitHub, y para qué son.**

Replit y GitHub son dos herramientas muy populares en el mundo de la programación, pero cada una tiene una función distinta.

**REPLIT**

¿Qué es? Replit es una plataforma en línea que proporciona un entorno de desarrollo integrado (IDE) para escribir, ejecutar y compartir código. No requiere instalación de software en tu computadora, ya que todo se realiza desde el navegador. Soporta más de 50 lenguajes de programación.

¿Para qué sirve? Replit es útil para programadores que quieren escribir y probar código rápidamente sin configurar un entorno local. Es ideal para aprender a programar, crear prototipos de proyectos pequeños o colaborar con otros en tiempo real. Permite ejecutar código inmediatamente y colaborar en proyectos con otros usuarios sin necesidad de tener conocimientos avanzados de configuración de herramientas.

**GITHUB**

¿Qué es? GitHub es una plataforma de alojamiento de código basada en la web que utiliza Git, un sistema de control de versiones. Permite a los desarrolladores almacenar y gestionar proyectos de software, hacer un seguimiento de los cambios en el código y colaborar en equipos.

¿Para qué sirve? GitHub se usa para gestionar el código fuente de proyectos a lo largo del tiempo. Permite controlar versiones, hacer colaboraciones entre múltiples desarrolladores y compartir proyectos con la comunidad, especialmente en el caso de proyectos de código abierto. También es útil para llevar un registro de los cambios realizados en el código y facilitar la colaboración en equipos grandes.

1. **¿Qué es la ciencia de datos?**

La ciencia de datos es una disciplina que se centra en la extracción de conocimiento y patrones a partir de grandes volúmenes de datos. Combina diversas áreas del conocimiento, como estadística, matemáticas, informática y conocimiento específico del dominio en cuestión, para transformar datos crudos en información útil para la toma de decisiones.

La ciencia de datos involucra varios procesos clave:

* Recolección y almacenamiento de datos: Implica la obtención y organización de grandes cantidades de datos, tanto estructurados (como bases de datos) como no estructurados (como texto, imágenes o videos).
* Limpieza y preparación de datos: Los datos suelen estar desordenados o incompletos, por lo que se requiere un proceso de depuración para prepararlos para su análisis.
* Análisis exploratorio de datos: Consiste en examinar los datos para identificar patrones, relaciones y características relevantes.
* Modelado y algoritmos: Utiliza técnicas estadísticas y de aprendizaje automático (machine learning) para construir modelos predictivos o descriptivos. Esto permite hacer predicciones sobre eventos futuros o extraer información que no es inmediatamente obvia.
* Visualización de datos: Una vez analizados, los resultados se presentan de manera visual (gráficos, mapas, etc.) para facilitar la comprensión de los hallazgos.

La ciencia de datos no solo se limita a analizar datos, sino que también crea modelos predictivos para prever comportamientos o tendencias futuras. Por ejemplo, puede usarse para predecir el comportamiento de los consumidores, los precios de las acciones en los mercados financieros o el diagnóstico de enfermedades a partir de datos médicos.

Es una disciplina esencial en diversos sectores, como el negocio, salud, finanzas, marketing, entretenimiento, y muchos otros, donde se utiliza para mejorar decisiones estratégicas, optimizar procesos y descubrir oportunidades ocultas en los datos.