Trabajo Práctico de Prácticas Profesionalizantes 1

Germán bisutti Moscatelli

1.Armar un glosario de términos y definiciones:

Software de sistema. Software de programación. Software de aplicación. Software incrustado. Programador. Ingeniero de software. Desarrollador de software. Desarrollo ágil. DevOps. Metodología de cascada. Inteligencia artificial (IA). Desarrollo nativo de la nube. Blockchain. Código bajo. Pueden agregar otros que hayan identificado

-**Software de sistema**: es el conjunto de instrucciones o programas que indican a una computadora lo que debe hacer para proporcionar funciones básicas como sistemas operativos, gestión de discos, utilidades, gestión de hardware y otras necesidades operativas.

-**Software de programación**: es el conjunto de instrucciones o programas que indican a una computadora lo que debe hacer para proporcionar a los programadores herramientas como editores de texto, compiladores, enlazadores, depuradores y otras herramientas para crear código.

-**Software de aplicación**: es el conjunto de instrucciones o programas que indican a una computadora lo que debe hacer para ayudar a los usuarios a realizar tareas. Las suites de productividad de oficina, el software de gestión de datos, los reproductores multimedia y los programas de seguridad son algunos ejemplos. Las aplicaciones también se refieren a aplicaciones web y móviles como las que se utilizan para comprar en Amazon.com, socializar en Facebook o publicar fotos en Instagram.

-**Software incrustado**: El software de sistemas incrustados se utiliza para controlar máquinas y dispositivos que no suelen considerarse computadoras: redes de telecomunicaciones, automóviles, robots industriales, entre otros. Estos dispositivos, y su software, pueden conectarse como parte del Internet de las cosas.

-**Programador**: Los programadores, o codificadores, escriben el código fuente para programar computadoras para tareas específicas como la fusión de bases de datos, el procesamiento de pedidos en línea, el enrutamiento de las comunicaciones, la realización de búsquedas o la visualización de texto y gráficos. Los programadores suelen interpretar las instrucciones de los desarrolladores e ingenieros de software y utilizan lenguajes de programación como C++ o Java para llevarlas a cabo.

-**Ingeniero de software**:Los ingenieros de software aplican principios de ingeniería para crear software y sistemas que resuelvan problemas. Utilizan un lenguaje de modelado y otras herramientas para idear soluciones que a menudo pueden aplicarse a los problemas de forma general, en lugar de limitarse a resolver un caso o cliente específico. Las soluciones de ingeniería de software se adhieren al método científico y deben funcionar en el mundo real, como ocurre con los puentes o los ascensores.

-**Desarrollador de software**: Los desarrolladores de software juegan un papel menos formal que los ingenieros y pueden participar activamente en áreas específicas del proyecto (por ejemplo, la escritura de código). Al mismo tiempo, generan todo el ciclo de vida del desarrollo del software, incluyendo el trabajo en equipos funcionales para transformar los requisitos en características, gestionar los equipos y procesos de desarrollo y realizar pruebas y mantenimiento del software.

-**Desarrollo ágil**:El desarrollo ágil divide los requisitos en funciones que se pueden incorporar y ofrece rápidamente esas funciones mediante un desarrollo incremental. Un bucle de retroalimentación ayuda a identificar y corregir los defectos a medida que la funcionalidad continúa implementándose.

-**Devops**: una combinación de desarrollo y operaciones, es un enfoque ágil que reúne el desarrollo de software y las operaciones de TI en el diseño, desarrollo, implementación y soporte de software.

-**Metodología de cascada**:La metodología de cascada, a menudo considerada la metodología tradicional de desarrollo de software, es un conjunto de pasos lineales en cascada que van desde la planificación y la recopilación de requisitos hasta el despliegue y el mantenimiento.

-**Inteligencia artificial (IA)**:La IA permite que el software emule la toma de decisiones y el aprendizaje humanos. Las redes neuronales, el aprendizaje automático, el procesamiento del lenguaje natural y las capacidades cognitivas brindan a los desarrolladores y a las empresas la oportunidad de ofrecer productos y servicios que desarticulan los mercados y se adelantan a la competencia.

-**Desarrollo nativo de la nube**: El desarrollo nativo de la nube es una forma de crear aplicaciones para utilizar en entornos de nube. Una aplicación nativa de la nube consta de componentes discretos y reutilizables que se conocen como microservicios y que están diseñados para integrarse en cualquier entorno de nube.

Estos microservicios actúan como bloques de construcción y a menudo se empaquetan en contenedores. Gracias a esta arquitectura, las aplicaciones nativas de la nube pueden utilizar entornos de nube para mejorar el rendimiento, la flexibilidad y la extensibilidad de la aplicación.

-Blockchain: Blockchain es un libro de contabilidad seguro y vinculado digitalmente que elimina el costo y la vulnerabilidad que introducen partes como bancos, organismos reguladores y otros intermediarios. Está transformando los negocios al liberar capital, acelerar los procesos, reducir los costos de las transacciones y mucho más.

Blockchain presenta una enorme oportunidad para el desarrollo de software. Los desarrolladores están trabajando con los libros de contabilidad distribuidos y la tecnología de código abierto Hyperledger (enlace externo a ibm.com) para cambiar el funcionamiento de las empresas.

-**Código bajo**: Forrester define el concepto de código bajo como: "Productos y/o servicios en la nube para el desarrollo de aplicaciones que emplean técnicas visuales y declarativas en lugar de programación y que están disponibles para los clientes a bajo costo o sin costo en dinero ni en capacitación ..." 4 En resumen, es una práctica de desarrollo que reduce la necesidad de programación y permite a quienes no son programadores o desarrolladores crear o ayudar a crear aplicaciones con rapidez y a un costo menor.

2. Roles Profesionales:

A)Elabora un cuadro comparativo que describa las principales diferencias y similitudes entre los roles de programador, ingeniero de software y desarrollador de software.

B)Investiga y menciona ejemplos de herramientas o tecnologías que cada uno de estos profesionales podría utilizar en su trabajo diario.

Características	Programador	Desarrollador de Software	Ingeniero de Software
Definición	Se enfoca en escribir código y solucionar problemas específicos mediante la programación.	Diseña, desarrolla y mantiene software, participando en todo el ciclo de vida del desarrollo.	Aplica principios de ingeniería para diseñar sistemas de software robustos y escalables.
Enfoque	Principalmente en la implementación del código.	En el desarrollo de software como un producto completo.	En la arquitectura, escalabilidad y eficiencia del software.
Tareas principales	Escribir código, depurar errores, probar funciones específicas.	Diseñar, programar, probar y mantener aplicaciones.	Diseñar arquitecturas, gestionar sistemas complejos, optimizar procesos.
Conocimientos clave	Lenguajes de programación, estructuras de datos, algoritmos.	Desarrollo full-stack, bases de datos, control de versiones, metodologías ágiles.	Patrones de diseño, arquitectura de software, ingeniería de requisitos, optimización de sistemas.
Grado de responsabilidad	Bajo, se enfoca en tareas específicas.	Medio, tiene más control sobre el desarrollo del software.	Alto, toma decisiones sobre diseño, arquitectura y escalabilidad.
Necesidad de formación	Puede aprender de forma autodidacta o con cursos.	Generalmente requiere conocimientos más	Suele requerir formación académica en

		amplios en tecnologías y buenas prácticas.	Ingeniería de Software o Ciencias de la Computación.
Participación en el ciclo de desarrollo	Principalmente en la fase de codificación.	Participa en todo el proceso de desarrollo.	Involucrado desde la planificación hasta la implementación y mantenimiento.
Ejemplo de tarea	Implementar una nueva función en una aplicación.	Desarrollar una aplicación desde cero, integrando múltiples componentes.	Diseñar la arquitectura de un sistema para soportar miles de usuarios concurrentes.

Programador:

1. **Visual Studio Code** (Editor de código)
2. **Git** (Control de versiones)
3. **GitHub** / **GitLab** (Plataformas de colaboración y control de versiones)
4. **Node.js** (Entorno de ejecución para JavaScript)
5. **JQuery** (Biblioteca de JavaScript para simplificar la manipulación del DOM)

Desarrollador de Software:

1. **React** / **Vue.js** / **Angular** (Frameworks para el desarrollo de aplicaciones web interactivas)
2. **Django** / **Flask** (Frameworks de Python para desarrollo web)
3. **Node.js** (Entorno de ejecución JavaScript en el servidor)
4. **MySQL** / **PostgreSQL** / **MongoDB** (Bases de datos)
5. **Docker** (Contenerización de aplicaciones para facilitar el despliegue)
6. **AWS** / **Azure** (Plataformas en la nube para despliegue y gestión de aplicaciones)

Ingeniero de Software:

1. **Jira** (Gestión de proyectos y tareas)
2. **UML** (Lenguaje de modelado para diseñar sistemas)

3. **Enterprise Architect** (Herramienta para modelado y diseño de sistemas complejos)
4. **Selenium** (Herramienta de pruebas automatizadas)
5. **JUnit** / **TestNG** (Frameworks para pruebas unitarias)
6. **AWS** / **Azure** (Plataformas de nube para infraestructura de software)
7. **Terraform** (Herramienta para infraestructura como código)

3- Enumera y describe brevemente (con sus palabras) los pasos del proceso de desarrollo de software mencionados en el texto

El proceso de desarrollo de software es un enfoque que se compone de diversas fases relacionadas entre sí.

1. Seleccionar una metodología: Escoger un marco de trabajo que guiará el desarrollo, como Agile, DevOps, metodología en cascada u otros.

2. Recopilar requisitos: Identificar y documentar las necesidades del usuario para entender lo que se debe construir.

3. Elegir o crear una arquitectura: Definir el formato técnico del software, incluyendo tecnologías y características a utilizar.

4. Desarrollar un diseño: Crear un plan detallado para ver cómo se podrán resolver los planteamientos, incluyendo diagramas y modelos.

5. Crear un modelo: Utilizar herramientas de modelado (como UML o SysML) para validar y simular el diseño antes de implementarlo.

6. Construir código: Escribir el software en el lenguaje de programación elegido, asegurando la calidad a través de revisiones y colaboración en equipo.

7. Pruebas: Ejecutar pruebas previas basadas en los escenarios planificados para garantizar que el software funcione correctamente y cumpla con los requisitos.

8. Gestionar la configuración y los defectos: Mantener el control sobre todas las distintas versiones y equipos del software, analizando y tomando nota de los defectos.

9. Implementar: Liberar y Ejecutar el software para su uso y resolver cualquier problema que surja en dicho proceso.

10. Migrar datos: Transferir datos de las aplicaciones o fuentes existentes al nuevo sistema, si es necesario.

11. Gestionar y medir el proyecto: Monitorear el progreso y la calidad del mismo a lo largo del ciclo de vida del software, utilizando métricas y modelos apropiados.

El proceso se integra en la gestión del ciclo de vida de las aplicaciones (ALM), que permite un enfoque continuo de mejora, donde los problemas identificados en la fase de mantenimiento pueden retroalimentar el ciclo de desarrollo futuro.

4. Cada grupo deberá investigar y analizar cómo estas tecnologías están influyendo en el desarrollo de software, identificando ejemplos y casos de uso relevantes.

Áreas de enfoque:
Grupo 3 = Área 3

AREA3: Blockchain y Desarrollo de Software:

- Desarrollo de aplicaciones descentralizadas (dApps).
- Seguridad y transparencia en el desarrollo de software.
- Impacto en la gestión de datos y la colaboración.

Blockchain y Desarrollo de Software

1. Desarrollo de Aplicaciones Descentralizadas (dApps)

Las dApps (aplicaciones descentralizadas) son aplicaciones que funcionan sobre una red blockchain en lugar de depender de servidores centralizados. Se caracterizan por su resistencia a la censura, mayor seguridad y descentralización en la gestión de datos.

Ejemplo: Ethereum y los Smart Contracts

Ethereum es una plataforma blockchain que permite el desarrollo de dApps mediante contratos inteligentes, que son programas autoejecutables con reglas predefinidas.

- Uniswap: Plataforma de intercambio de criptomonedas sin intermediarios.
- OpenSea: Mercado de NFTs basado en Ethereum.
- Axie Infinity: Juego basado en blockchain con economía digital propia.

Ventajas de las dApps en el desarrollo de software

- No requieren servidores centralizados, reduciendo costos de mantenimiento.
- Mayor transparencia gracias a la inmutabilidad de la blockchain.
- Eliminación de intermediarios en transacciones y procesos.

2. Seguridad y Transparencia en el Desarrollo de Software

Blockchain mejora la seguridad del software al proporcionar un registro inmutable de transacciones y eventos dentro de una red descentralizada.

Casos de uso en seguridad

- Control de versiones y auditoría: Plataformas como Hyperledger Fabric permiten registrar cambios en el código fuente de software, garantizando integridad y trazabilidad.
- Gestión de identidades digitales: Blockchain evita fraudes y robos de identidad mediante sistemas como Self-Sovereign Identity (SSI), donde los usuarios tienen control total sobre sus datos.

- Ciberseguridad y detección de amenazas: Empresas como Guardtime usan blockchain para proteger datos y detectar ataques informáticos en tiempo real.

Beneficios

- Protección contra manipulaciones y alteraciones de datos.
- Verificación de autenticidad del software sin necesidad de terceros.
- Transparencia en procesos de desarrollo y auditoría.

3. Impacto en la Gestión de Datos y la Colaboración

Blockchain permite nuevas formas de almacenar, compartir y gestionar datos en el desarrollo de software.

Casos de uso en gestión de datos

- Almacenamiento distribuido: Plataformas como IPFS (InterPlanetary File System) y Storj reemplazan servidores tradicionales con almacenamiento descentralizado.
- Colaboración en desarrollo de software: Proyectos como Gitcoin permiten a desarrolladores colaborar en código abierto y recibir recompensas en criptomonedas.
- Bases de datos inmutables: Tecnologías como BigchainDB combinan blockchain con bases de datos escalables para mejorar la seguridad en registros de datos.

Beneficios

- Eliminación de puntos únicos de fallo en almacenamiento de datos.
- Mayor eficiencia en colaboración y financiamiento de proyectos de software.
- Transparencia en contribuciones y licencias de código abierto