UML

NICOLAS ANDRES ROJAS GOMEZ

CENTRO TECNOLOGICO DE LA MANOFACTURA AVANZADA

BASES DE DATOS SQL Y NOSQL

FICHA:3169892

2025

Preguntas de Introducción:

• ¿Qué es un diagrama?

Un **diagrama** es una representación visual que organiza y muestra información de manera estructurada. Se utiliza para explicar conceptos, ilustrar relaciones entre elementos, visualizar datos y facilitar la comprensión de temas complejos.

• ¿Qué tipos de diagramas conoce o ha utilizado?

- **Diagramas de flujo**: Ideales para representar procesos paso a paso, como algoritmos en programación o flujos de trabajo.
- **Diagramas de Venn**: Excelentes para comparar y contrastar conceptos en matemáticas o lógica.

- **Diagramas UML** (como diagramas de clases y de casos de uso): Son clave en ingeniería de software para modelar estructuras y funcionalidades de sistemas.
- **Diagramas de árbol**: Útiles en estructuras jerárquicas y en inteligencia artificial para representar decisiones.

• ¿Cómo se estructura un diagrama?

La estructura de un diagrama depende de su propósito y tipo, pero en general sigue estos principios básicos:

1. **Elementos principales**: Son los componentes clave que representan información. Pueden ser figuras como rectángulos, círculos, nodos o líneas.
2. **Conexiones**: Indican relaciones entre elementos mediante líneas, flechas o vínculos.
3. **Etiquetas o textos**: Describen cada elemento para mejorar la comprensión.
4. **Jerarquía y organización**: Se distribuyen los elementos de forma lógica, ya sea de arriba hacia abajo, de izquierda a derecha o en una disposición radial.
5. **Simplicidad y claridad**: Debe ser fácil de interpretar y evitar sobrecarga visual.

• ¿Cómo se construye un diagrama?

1. **Definir el propósito del diagrama**

Antes de empezar, identifica qué información deseas representar. ¿Es un proceso? ¿Una estructura jerárquica? ¿Una relación entre elementos?

2. **Seleccionar el tipo de diagrama adecuado**

Dependiendo del propósito, elige el tipo más apropiado:

- Para procesos: **Diagrama de flujo**.
- Para bases de datos: **Diagrama entidad-relación**.

- Para comparación de conceptos: **Diagrama de Venn**.
- Para proyectos: **Diagrama de Gantt**.

3. **Recopilar los elementos clave**

Enumera los conceptos, objetos o datos que formarán parte del diagrama. Cada elemento debe estar claramente definido.

4. **Determinar relaciones y estructura**

- Organiza los elementos de forma lógica.
- Usa líneas, flechas o conexiones para mostrar relaciones entre ellos.
- Aplica jerarquías si es necesario.

5. **Diseñar el esquema inicial**

- Dibuja un boceto preliminar.
- Ajusta posiciones y conexiones para mayor claridad.

6. **Utilizar herramientas digitales o materiales físicos**

Puedes construir el diagrama manualmente en papel o con software como:

- **Microsoft Visio**, **Lucidchart**, **Draw.io** para diagramas técnicos.
- **Excel** o **PowerPoint** para esquemas sencillos.

7. **Revisar y mejorar**

- Verifica que sea claro y fácil de interpretar.
- Asegúrate de que las conexiones y etiquetas sean precisas.
- Ajusta el diseño para mejorar la presentación.

• ¿Cuál es el objetivo de usar diagramas?

El objetivo de usar **diagramas** es representar información de manera visual y estructurada para facilitar la comprensión, el análisis y la comunicación de ideas complejas.

• ¿Por qué consideras que deben usarse diagramas en el diseño de soluciones de software?

- Facilitan la representación de la estructura del software antes de la implementación.

- Ayudan a visualizar cómo interactúan los componentes, como módulos, bases de datos y usuarios.

- Un diagrama bien hecho permite que programadores, analistas y clientes entiendan el diseño sin necesidad de código.

- Reduce malentendidos en los requerimientos del sistema.

- Con un esquema visual, es más fácil identificar problemas y mejorar la estructura del software antes de escribir código.

- Agiliza el trabajo de los equipos al ofrecer una visión clara del flujo de datos y procesos.
- Un software bien documentado con diagramas es más fácil de modificar o mejorar en el futuro.

Investigue:

- **¿Cuáles son los tipos de diagrama más relevantes en UML? – Describa al menos 4 de ellos.**

UML (**Unified Modeling Language**) es un lenguaje de modelado utilizado en ingeniería de software para diseñar y visualizar sistemas antes de su implementación. Dentro de UML, existen varios tipos de diagramas esenciales

1. Diagrama de casos de uso

- Representa **interacciones** entre usuarios y el sistema.
- Muestra **actores** (usuarios, sistemas externos) y **casos de uso** (funcionalidades principales del sistema).
- Útil para definir **requerimientos funcionales** y alcance del sistema.
- Ejemplo: Un usuario que inicia sesión en una aplicación y accede a su perfil.

2. Diagrama de clases

- Modela la **estructura de un sistema orientado a objetos**.
- Define **clases**, atributos, métodos y relaciones entre ellas.
- Es clave para diseñar la **arquitectura de software**, especialmente en lenguajes como Java o C++.
- Ejemplo: Una clase "Usuario" con atributos como "nombre" y "email", y métodos como "registrar()" y "actualizarDatos()".

3. Diagrama de secuencia

- Representa el **orden en el que ocurren las interacciones** entre objetos.
- Muestra mensajes intercambiados en **líneas de tiempo**, útil para visualizar flujos de procesos.
- Facilita la comprensión de lógica interna en módulos de software.
- Ejemplo: El flujo de autenticación en una aplicación con validación de contraseña.

4. Diagrama de actividad

- Describe **procesos secuenciales** o flujos de trabajo dentro del sistema.
- Usa **acciones, decisiones y bifurcaciones** para representar estados y transiciones.
- Ayuda en el modelado de **algoritmos y flujos de negocio**.
- Ejemplo: El proceso de compra en un comercio electrónico desde la selección de productos hasta el pago.

- **¿Cómo representar instancias de las clases en UML? -Elabore un ejemplo de una clase. Con sus atributos y características.**

En UML, una **instancia de clase** se representa mediante un **diagrama de objetos**, que muestra objetos específicos creados a partir de una clase y sus valores concretos. Estos diagramas ayudan a visualizar cómo las clases se comportan con datos reales en un sistema.

- **¿ UML permite incluir notas, cómo, para qué?**

UML permite **incluir notas** en los diagramas para agregar información adicional que ayude a la comprensión del modelo. Estas notas no afectan la estructura del sistema, pero proporcionan contexto, aclaraciones o explicaciones sobre elementos específicos.

Cómo se incluyen notas en UML

- Se representan como **rectángulos con esquinas dobladas**, simulando una hoja de papel.
- Se conectan al elemento relevante mediante una **línea discontinua**.
- Pueden contener **descripciones, justificaciones o detalles técnicos**.

¿Para qué sirven las notas en UML?

Las notas tienen múltiples usos, entre ellos:

1. **Explicar detalles técnicos**: Se pueden usar para clarificar reglas de negocio o restricciones en el sistema.
2. **Agregar comentarios**: Permiten documentar decisiones de diseño.
3. **Ejemplificar atributos o métodos**: Se pueden incluir valores de ejemplo para mejorar la comprensión.
4. **Indicar cambios o futuras mejoras**: Útil en diagramas colaborativos para señalar ajustes.
5. **Aclarar relaciones complejas**: Cuando las conexiones entre clases u objetos no son evidentes, una nota ayuda a describir la lógica.

- ## ¿Qué es un método constructor y para qué se utiliza?

Un **método constructor** es una función especial dentro de una clase en programación orientada a objetos (POO) que se utiliza para **crear e inicializar** objetos de esa clase. Su propósito es asignar valores iniciales a los atributos de la clase y preparar el objeto para su uso.

¿Para qué se utiliza?

Inicializar atributos: Asigna valores a los atributos cuando se crea un objeto.

Automatizar la creación de objetos: Evita que el programador tenga que establecer manualmente cada atributo después de instanciar un objeto.

Garantizar reglas de negocio: Puede validar datos al momento de la creación de un objeto, evitando valores incorrectos.

- ## ¿Qué tipos de Relaciones pueden especificarse entre clases?

En UML, las **relaciones entre clases** son fundamentales para definir la estructura y el comportamiento del sistema. Aquí están los tipos más comunes:

1 Asociación

- Representa una **relación estructural** entre dos clases.
- Se indica con una **línea** entre las clases y puede tener cardinalidad (`1`, `0..*`, etc.).
- Ejemplo: Un `Cliente` puede estar asociado con varios `Pedidos`.

2 Agregación

- Es un tipo de asociación, pero con una **relación de "todo-parte"**.

- La clase agregadora **contiene** objetos de la otra clase, pero estos pueden existir de forma independiente.
- Se indica con un **rombo vacío** en la clase principal.
- Ejemplo: Un `Departamento` tiene varios `Empleados`, pero los empleados pueden existir sin el departamento.

3 Composición

- Similar a la agregación, pero con una **relación de dependencia fuerte**.
- Si el objeto principal se elimina, sus partes también lo hacen.
- Se indica con un **rombo lleno** en la clase principal.
- Ejemplo: Un `Auto` tiene `Motor`; si el auto se destruye, el motor también.

4 Herencia (Generalización)

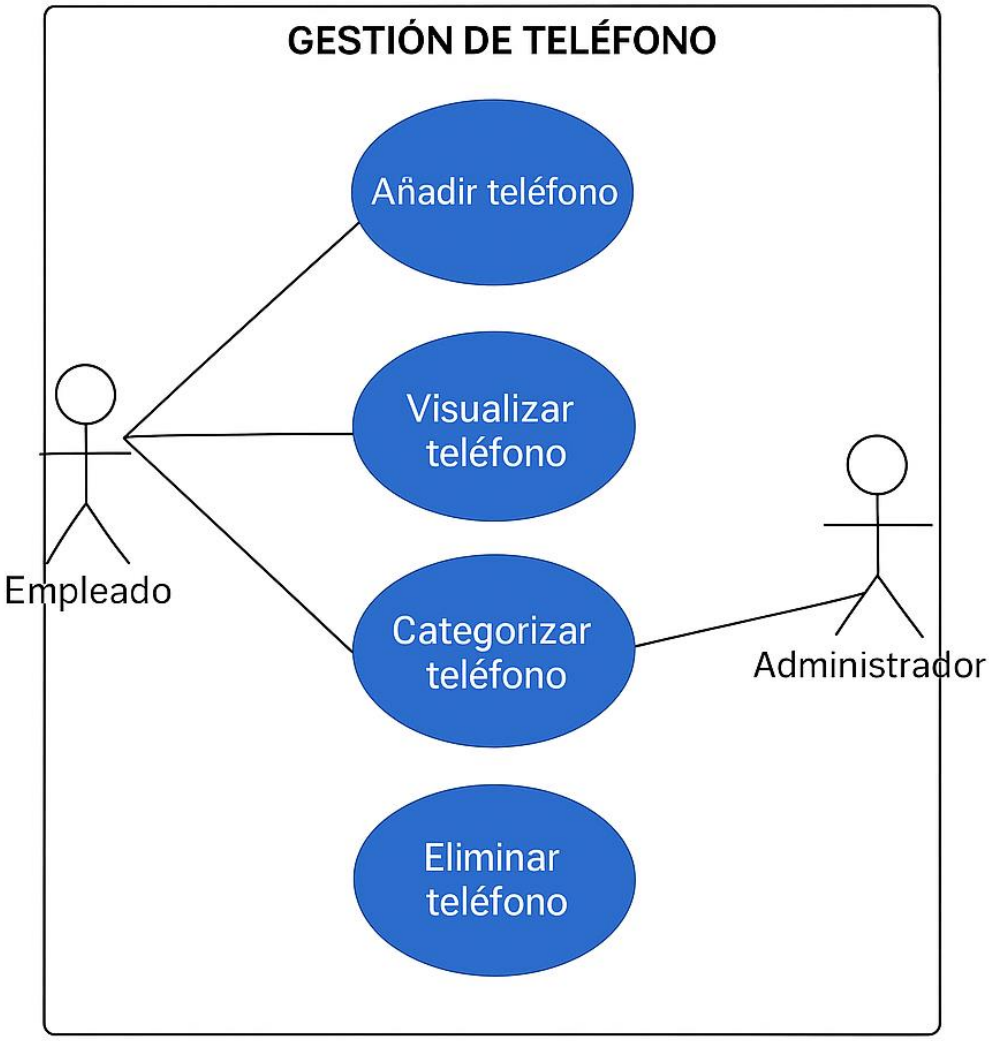
- Representa una relación **padre-hijo**.
- Una subclase **hereda** atributos y métodos de la superclase.
- Se indica con una **línea con triángulo** apuntando a la clase padre.
- Ejemplo: `Estudiante` hereda de `Persona`.

5 Dependencia

- Indica que una clase **usa** a otra de manera temporal.
- Se representa con una **línea discontinua con flecha**.
- Ejemplo: `Usuario` depende de `ServicioAutenticación` para iniciar sesión.

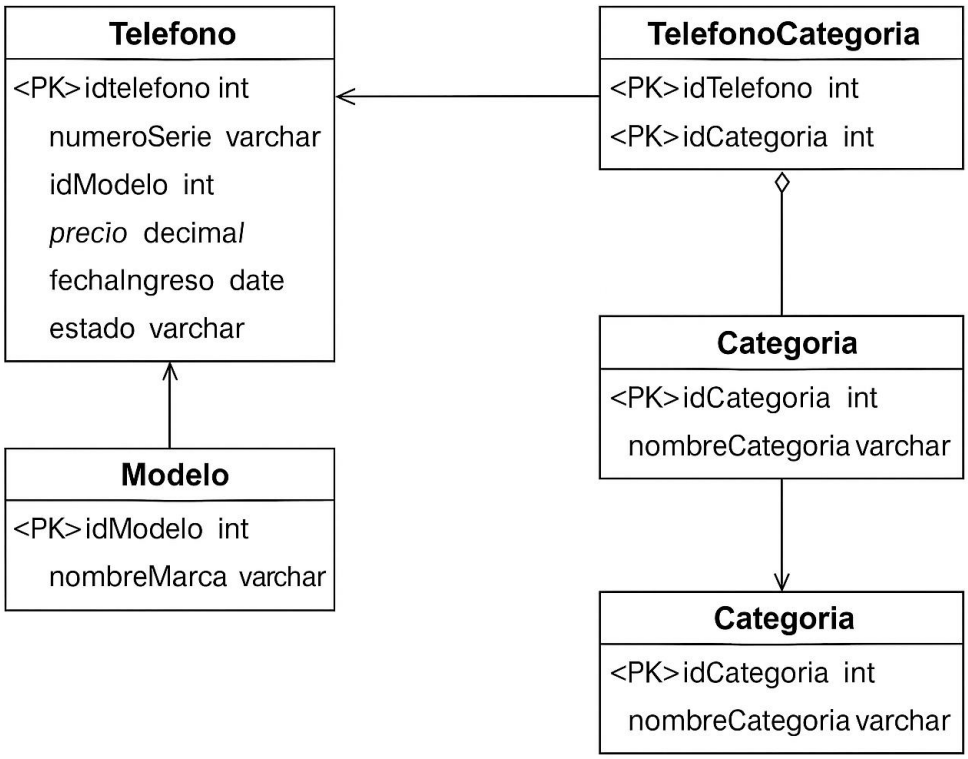
ACTIVIDAD 1: DIAGRAMA DE CASOS DE USO EN UML

Consulte cómo se crea un Diagrama de Casos de Uso en UML y cuáles son los elementos que lo componen, realice un diagrama para describir un proceso que identifique en su contexto (por ejemplo: el proceso para ir de compras al supermercado, un proceso de matricula, el pro ceso para hacer un saque en un partido de tenis, etc).


GESTIÓN DE TELÉFONO
Añadir teléfono
Visualizar teléfono
Categorizar teléfono
Eliminar teléfono
Empleado
Administrador

Actividad 2: Diagrama de Clases en UMl:

Los diagramas de clase permiten modelar las diferentes entidades que hacen parte de un sistema de información; observe el ejemplo de la imagen 3 para comprender mejor lo que es un diagrama de Clases en UML:



Glosario de Términos de Programación y UML

Término	Definición
UML (Lenguaje Unificado de Modelado)	Un lenguaje gráfico utilizado para visualizar, especificar, construir y documentar sistemas de software. Incluye diagramas como de clases, casos de uso, secuencia, etc.
POO (Programación Orientada a Objetos)	Paradigma de programación basado en el concepto de "objetos", que encapsulan datos (atributos) y comportamientos (métodos o funciones).
Objeto	Instancia de una clase que contiene atributos y métodos definidos por dicha clase. Representa una entidad con identidad, estado y comportamiento.
Clase	Plantilla o modelo a partir del cual se crean objetos. Define qué atributos y métodos tendrán esos objetos.
Modelo	Representación simplificada de un sistema o parte del mismo. En UML, los modelos ayudan a comprender, diseñar y comunicar ideas sobre el sistema.
Atributo	Propiedad o característica que describe el estado de un objeto (por ejemplo, nombre, edad, color).
Alcance (Scope)	Define la visibilidad de una variable o miembro (por ejemplo, público, privado o protegido). Controla desde dónde puede ser accedido.
Parámetro	Variable que se pasa a una función o método para que actúe sobre ella. Puede ser entrada, salida o ambos.
Agregación	Relación entre clases donde una es parte de otra, pero puede existir independientemente. Es una relación "tiene un".
Herencia	Mecanismo que permite que una clase hija herede atributos y métodos de una clase padre, promoviendo la reutilización de código.
Polimorfismo	Capacidad de los objetos de diferentes clases de responder de forma diferente al mismo mensaje o método.
Extensión	En UML, permite a un caso de uso ampliar el comportamiento de otro de forma condicional o adicional.
Función (o Método)	Conjunto de instrucciones que realizan una tarea específica. Puede recibir parámetros y devolver un resultado.

Término	Definición
Relación	Conexión entre dos clases u objetos que indica una asociación, dependencia, herencia u otro vínculo lógico.
Abstracción	Principio que permite centrarse en los aspectos esenciales de un objeto, ignorando los detalles irrelevantes. Facilita simplificar sistemas complejos.
Sobrecarga	Técnica que permite definir múltiples funciones o métodos con el mismo nombre pero diferentes parámetros o tipos.