MANUAL DE DIAGRAMAS UML

1. Introducción a UML

UML (Lenguaje de Modelado Unificado) es un lenguaje gráfico que permite representar, visualizar, construir y documentar los elementos de un sistema software. Se usa ampliamente en el desarrollo orientado a objetos.

Los dos diagramas tratados en este manual —de clases y de secuencia— son esenciales para representar la estructura estática (clases) y dinámica (interacción entre objetos en el tiempo) de un sistema.

2. Diagramas de Clases

2.1 ¿Qué es un diagrama de clases?

Un diagrama de clases muestra la estructura estática de un sistema: clases, atributos, métodos y relaciones. Es fundamental en el análisis y diseño orientado a objetos, pues permite planificar la arquitectura del software antes de su implementación.

2.2 Elementos básicos

Clase: Rectángulo dividido en tres partes: nombre, atributos y métodos.

+ público, - privado, # protegido

Atributos: Variables internas de la clase, ej. -nombre: String

Métodos: Funciones que actúan sobre los atributos, ej. +getNombre(): String

2.3 Relaciones entre clases

Asociación: ClaseA ——— ClaseB

Agregación: ClaseA ◇——— ClaseB

Composición: ClaseA ◆——— ClaseB

Herencia: ClaseHija ———▷ ClasePadre

Implementación: Clase – –▷ Interfaz

Multiplicidad: Libro 1 —— * Ejemplar

2.4 Buenas prácticas

Usa nombres claros y en singular. Agrupa por función. Añade multiplicidad. Evita métodos innecesarios. No mezcles lógica de negocio con detalles técnicos.

2.5 Herramientas recomendadas

Online: Lucidchart, Draw.io, Creately, GenMyModel

Escritorio: StarUML, Visual Paradigm, Enterprise Architect, Astah

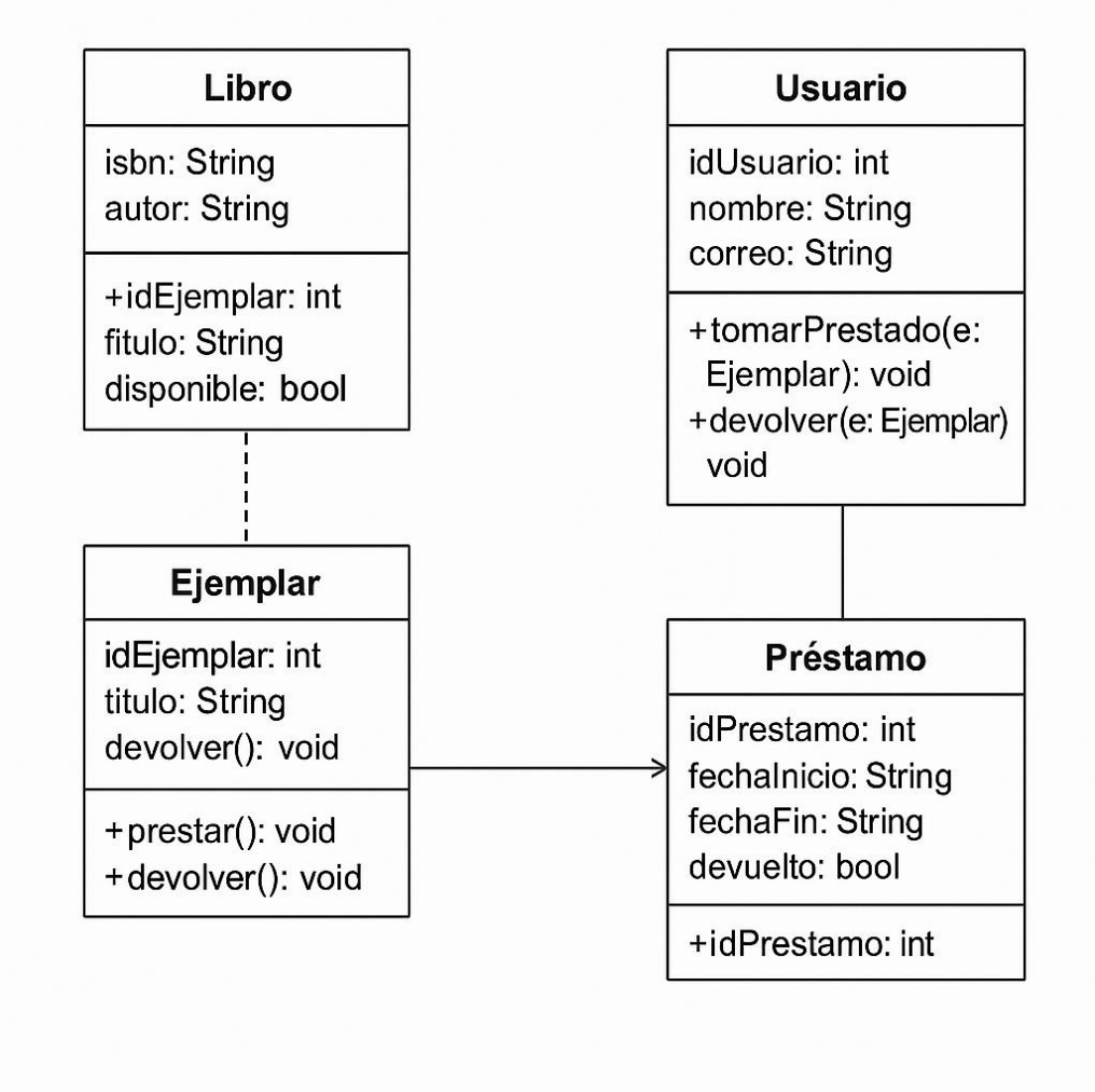
2.6 Ejemplo

Estudiante 1 ——— * Matricula

3. Diagrama de Clases Aplicado: Biblioteca

Clases: Libro, Ejemplar, Usuario, Préstamo. Con atributos, métodos y relaciones correspondientes.

Relaciones: Usuario ←→ Préstamo, Ejemplar ←→ Préstamo, Libro ◇——— Ejemplar



4. Diagramas de Secuencia

4.1 ¿Qué es?

El diagrama de secuencia muestra cómo interactúan los objetos del sistema en el tiempo. Se centra en el orden temporal de los mensajes.

4.2 Relación con Casos de Uso

Los casos de uso definen qué hace el sistema. Los diagramas de secuencia muestran cómo se lleva a cabo esa funcionalidad.

4.3 ¿Cuándo usarlo?

Para entender el comportamiento del sistema, modelar casos de uso, identificar errores lógicos, documentar o mejorar el diseño.

5. Elementos del Diagrama de Secuencia

Objetos y Actores: Rectángulos con nombre:Clase

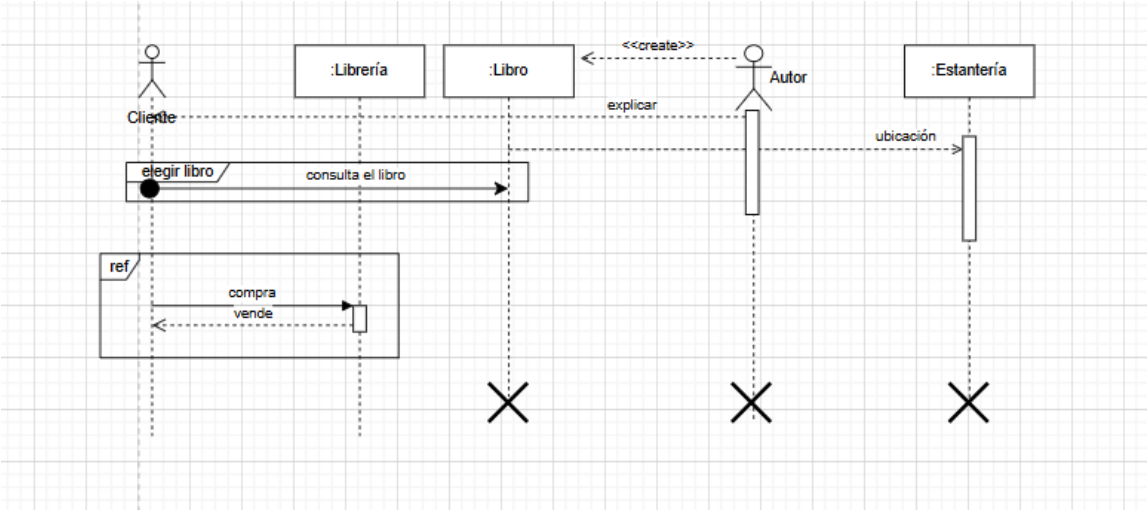
Línea de vida: línea vertical desde cada objeto

Cuadro de activación: rectángulo sobre la línea de vida

Tipos de mensajes: Síncrono, Asíncrono, Retorno, Creación, Eliminación, Recursivo, Perdido/Encontrado

6. Ejemplo aplicado

Escenario: Préstamo de libro en biblioteca. Flujo: Solicitud → Verificación → Registro → Cambio de estado → Devolución.



7. Diagramas de Casos de Uso

7.1 ¿Qué es un diagrama de casos de uso?

Un diagrama de casos de uso es una representación visual de las funciones de un programa, indicando qué acciones se realizan y quién las realiza, ya sea el sistema o los usuarios. Cada usuario u otro sistema involucrado es denominado un actor.

Estos diagramas cumplen dos propósitos fundamentales:

- Capturar los requisitos funcionales del sistema.
- Simplificar la construcción de modelos de objetos.

Cada acción realizada por un actor se denomina caso de uso. Aunque algunos casos puedan parecer similares, no deben considerarse idénticos, ya que responden a distintas interacciones o condiciones.

7.2 ¿Cómo hacer un diagrama de casos de uso?

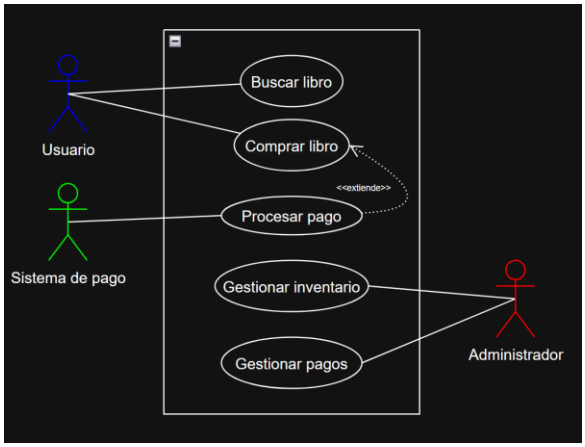
- Identificar los actores involucrados en el sistema.
- Definir los casos de uso que describen las acciones posibles.
- Dibujar el diagrama, incluyendo todos los actores y casos definidos.
- Conectar actores y casos mediante líneas que reflejen su relación.
- Revisar y refinar el diagrama para garantizar claridad y ausencia de errores.

7.3 Ejemplo aplicado: Librería

El usuario puede buscar libros y comprar aquellos disponibles en el inventario.

El administrador gestiona tanto los pagos como el inventario de libros.

El sistema de pago procesa las transacciones. Este caso de uso extiende al de 'comprar libro', ya que la compra depende de un pago exitoso.



8. Entornos de desarrollo utilizados

Para trabajar de forma ágil, colaborativa y versionable en la creación de diagramas UML, proponemos el siguiente flujo basado en **Visual Studio Code**, **PlantUML** y **Draw.io Integration**:

8.1 Visual Studio Code como editor principal

- **Ligero y gratuito**: VS Code arranca rápido y funciona en Windows, macOS y Linux.
- **Ecosistema de extensiones**: permite integrar herramientas de diagramado, control de versiones, previsualización de Markdown y más.
- **Workspaces**: configurar un espacio de trabajo específico para el proyecto (con archivos .code-workspace) facilita tener ajustados los ajustes de UML, rutas de salida de imágenes y snippets personalizados.

8.2 Draw.io para diagramas gráficos

- A pesar de la potencia de PlantUML, hay casos (diagramas muy complejos o prototipos rápidos) donde el diseño gráfico directo es más ágil.
- Draw.io Integration permite editar archivos .drawio dentro de VS Code y, al igual que PlantUML, versionarlos en Git:
    1. Instalar la extensión Draw.io Integration desde el marketplace de VS Code.
    2. Crear un archivo CasosDeUso.drawio en la carpeta diagrams/.
    3. Abrirlo con el editor incorporado, dibujar con la interfaz habitual de Draw.io y guardar.
    4. Exportar desde el mismo plugin a .png o .svg para incrustar en la documentación Markdown.
- **Ventaja clave**: mantiene todos los artefactos de diagrama (texto y gráficos) bajo el mismo control de versiones y en el mismo IDE, sin necesidad de salir a la web.

9. Organización del equipo y partes del trabajo

La organización para este trabajo ha sido relativamente básica. Dividimos el trabajo en tres partes principales:

- Diagramas de caso de uso
- Diagramas de clases
- Diagramas de actividades o de secuencia

A partir de ahí cada uno hizo su parte de investigación y escribió el manual correspondiente realizando los gráficos requeridos. La presentación también fue consensuada.