Modelado de sistemas con UML

CÓRDOBA CREA FUTURO

Bienvenidos al curso avanzado de programación. Este curso incluye apuntes en formato de lecturas, cuya complejidad se incrementa de manera progresiva y ha sido especialmente diseñada para favorecer un aprendizaje cómodo, siguiendo un enfoque pedagógico basado en competencias y en el modelo de aula invertida. Fue desarrollado por el magíster Walter F. Agüero, docente de diversas universidades argentinas, quien aporta conceptos propios en todo el material, a modo de síntesis de múltiples bibliografías leídas por el autor. Cuando sea necesario incluir una cita bibliográfica, esta se presentará en formato APA para que los estudiantes puedan ampliar la lectura.

1. Diagrama UML

1. Diagrama UML

Un diagrama UML es una herramienta de modelado visual que utiliza un lenguaje estándar (lenguaje unificado de modelado) para representar, diseñar y documentar sistemas de software complejos. Sirve para describir la estructura y el comportamiento de un sistema, facilitando la comunicación entre desarrolladores y otros profesionales. Es decir, los diagramas son representaciones gráficas utilizadas para expresar información de manera estructurada. Se emplean en diversos campos, como la educación, la informática, la ingeniería y la administración, ya que permiten organizar datos y facilitar su interpretación.

Los diagramas UML se dividen en dos categorías: los estructurales, que muestran la arquitectura estática de un sistema (como los diagramas de clases), y los de comportamiento, que describen cómo interactúa el sistema

con el tiempo (como los diagramas de casos de uso o de secuencias).

En el primer módulo del curso número 1, de carácter introductorio, iniciábamos los primeros pasos en programación describiendo los diagramas de flujo. Como puede apreciarse en este curso avanzado, hacemos algo similar, aunque con un enfoque y un uso más profesional.

Funciones principales de los diagramas UML

Los diagramas UML cumplen diversas funciones que contribuyen al desarrollo y la comprensión de sistemas de software. Entre las más destacadas, se encuentran las siguientes:

- **Visualizar la estructura y el comportamiento**. Permiten representar la estructura estática del sistema, como sus clases y relaciones, así como su comportamiento dinámico, como los flujos de proceso y las interacciones entre objetos.

- **Facilitar la comunicación**: proporcionan un lenguaje común y estandarizado para que los equipos de desarrollo, los analistas de negocio y otros actores clave puedan comprender el sistema con claridad.

- **Documentar sistemas**: funcionan como documentación de referencia sobre el diseño, la arquitectura y el funcionamiento de un sistema de *software*.

- **Diseñar y analizar sistemas**: resultan útiles en las fases de diseño, ya que permiten explorar diferentes opciones, refinar la arquitectura y verificar la coherencia del diseño.

Tipos comunes de diagramas UML

A continuación, se presentan los diagramas UML más comunes:

- **Diagramas de clases**. Muestran las clases del sistema, sus atributos, operaciones y las relaciones entre ellas.

- **Diagramas de casos de uso**: representan las interacciones entre los actores (usuarios o

sistemas externos) y el sistema.

- **Diagramas de secuencia**: muestran el orden de los mensajes que se intercambian los objetos a lo largo del tiempo.

- **Diagramas de actividades**: descomponen un proceso en pasos o actividades más pequeñas para ilustrar el flujo de trabajo.

- **Diagramas de componentes**: muestran cómo los componentes de *software* se organizan en módulos y las dependencias entre ellos.

- **Diagramas de despliegue**: representan la disposición física del *hardware* y el *software*, mostrando cómo se ejecutan los componentes en diferentes nodos

Explicación de los diagramas de clases UML

Cuando se construye una casa o un edificio, los arquitectos e ingenieros elaboran un conjunto de planos para visualizar todos los aspectos de la construcción antes de iniciar

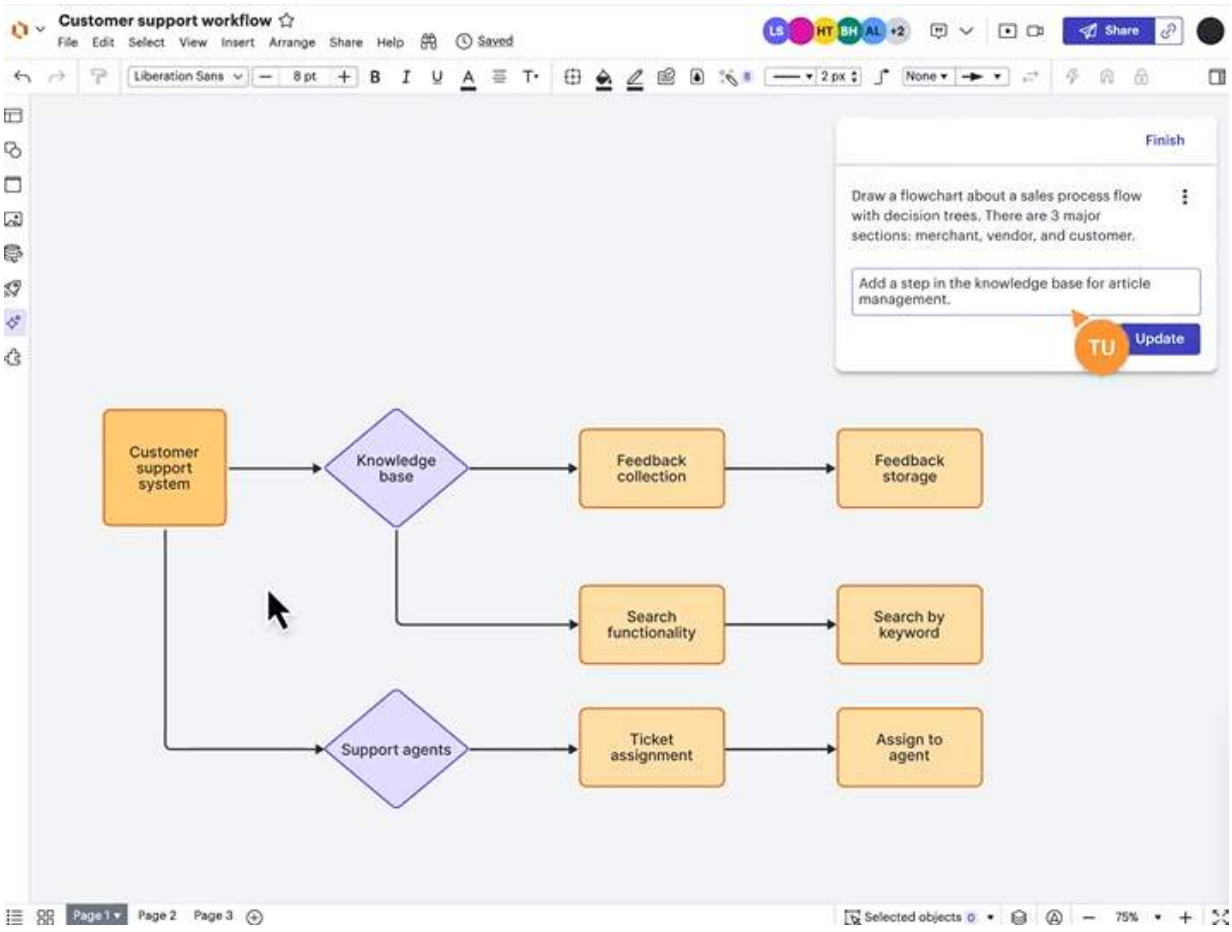
cualquier trabajo. Los diagramas de clases del lenguaje unificado de modelado (UML, por sus siglas en inglés)

cumplen una función similar a la de un plano, pero aplicada al desarrollo de *software*.

Siempre es recomendable construir un sistema con diagramas UML, especialmente durante las etapas iniciales y en los procesos de actualización. Los diagramas de clases forman parte de un conjunto más amplio de diagramas UML que representan distintos aspectos del desarrollo, todos ellos utilizando el mismo lenguaje para facilitar su interpretación.

Para construir un diagrama de clases UML, se puede utilizar la herramienta en línea Lucidchart, que cuenta con una versión gratuita y será explicada más adelante en este apunte o lectura (https://www.lucidchart.com).

Figura 1. Sitio web para armar diagramas de clases



Fuente: captura de pantalla de Lucidchart (https://www.lucidchart.com).

¿Qué es un diagrama de clases en UML?

Los diagramas de clases son uno de los diversos tipos de diagramas de estructura de UML. Estos diagramas muestran la estructura estática de un sistema, en lugar de representar cómo cambian los objetos con el tiempo.

En particular, los diagramas de clases visualizan las clases de un sistema y las relaciones entre ellas. En el diseño orientado a objetos, las clases crean y operan sobre objetos, los cuales

son instancias de clases. Por ello, las clases son componentes fundamentales de alto nivel en un sistema. Se derivan durante el proceso de diseño y se utilizan para comunicar el diseño o sus posibles modificaciones.

En un diagrama de clases, los nombres de las clases coinciden con los de los objetos, ya que la función de una clase es definir los atributos y operaciones de cada instancia de objeto del sistema. Así, una clase representa el plano de un objeto, y el diagrama de clases, el plano estático del sistema.

Durante el desarrollo de *software*, los diagramas de clases de UML no existen de manera aislada. Están vinculados a los diagramas de casos de uso y se relacionan estrechamente con los diagramas de objetos y de comunicación. En conjunto, todos los diagramas de UML permiten modelar, conceptualizar y documentar el funcionamiento de un sistema antes, durante y después de su implementación.

Para ampliar la información sobre los diagramas de clases en UML, se puede consultar el siguiente videotutorial:

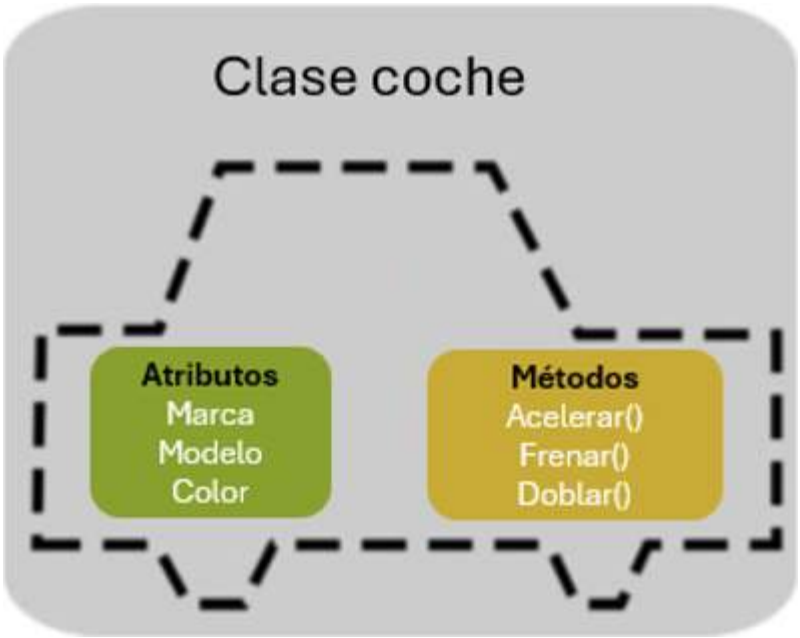
Fuente: **Lucid Software Español** [Lucid Software Español], (s.f.). *Tutorial - Diagrama de Clases UML* [video]. YouTube. https://www.youtube.com/watch?v=Z0yLerU0g-Q

¿Qué es una clase en UML?

Una clase es un elemento de modelado que define las características del objeto que representa, incluidos sus atributos y comportamientos.

Como ejemplo, puede considerarse la clase «Coche». Esta tiene un conjunto de atributos estáticos, como marca, modelo y color. La clase «Coche» también incluye métodos —acciones que puede realizar el objeto—, como acelerar, frenar y doblar.

Figura 2. Definición de clase en UML



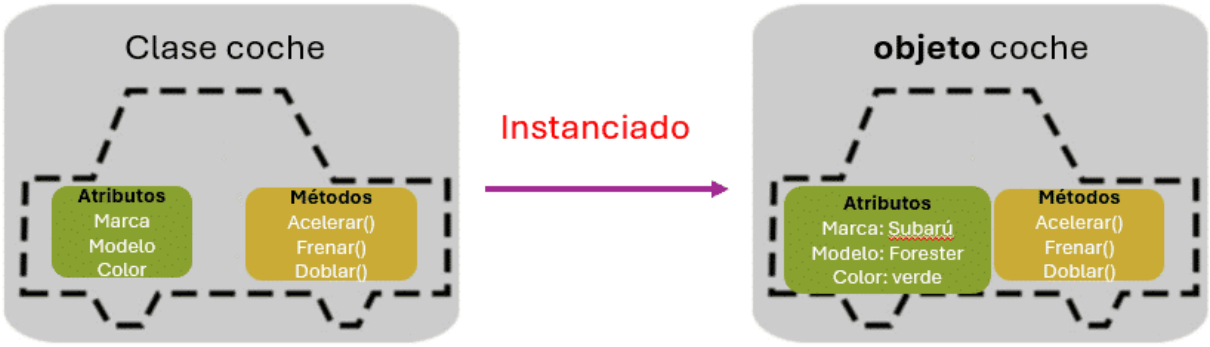
Fuente: elaboración propia.

Si la figura 2 se convirtiera en un código fuente en un lenguaje de programación orientado a objetos, en particular en lenguaje Java, se obtendría algo como lo siguiente:

```
public class Coche {

  // Atributos (variables de instancia)

  String color;

  String modelo;

  int ruedas;

   // Métodos

  public void acelerar() { };

  public void frenar() { };

  public void doblar() { }

}
```

Puesto que un objeto es una instancia de la clase, una instancia de ese objeto podría ser: Subaru, Forester, verde. En este ejemplo, los atributos del coche son marca, modelo y color.

Figura 3. Objeto coche (instanciada de la clase)



Fuente: elaboración propia.

¿Cuál es la finalidad de los diagramas de clases UML?

Un diagrama de clases UML cumple dos finalidades principales como modelo estático de un sistema orientado a objetos. En primer lugar, permite visualizar las clases de un sistema y sus propiedades. En segundo lugar, posibilita mostrar y analizar las relaciones entre las clases.

Además, los diagramas de clases UML constituyen la base de los diagramas de componentes y de despliegue, los cuales representan los aspectos de hardware y software de un sistema.

Figura 4. Clase en UML

Fuente: elaboración propia.

Los diagramas de clases UML son herramientas prácticas de modelado para construir una arquitectura de software. Junto con otros diagramas UML, los desarrolladores y los *stakeholders* (personas, grupos u organizaciones que tienen un interés directo o indirecto en un proyecto, empresa o decisión corporativa, ya que pueden verse afectados por sus acciones o, a su vez, influir en ellas) visualizan diferentes aspectos de un sistema. Estos diagramas permiten comprender cómo funciona el sistema, cómo se comporta y cómo se relacionan sus partes.

Crear diagramas de clases durante el diseño facilita el proceso de desarrollo al mostrar claramente las clases, sus atributos y sus métodos. También permiten observar cómo se relacionan las clases entre sí. Contar con una construcción conceptual del sistema antes de escribir el código favorece la comunicación entre desarrolladores y con otras partes interesadas.

Los diagramas de clases también permiten gestionar cambios en el sistema, ya que muestran un esquema de toda una aplicación. Aplicar modificaciones o adiciones a un

diagrama de clases UML proporciona una vista previa de las nuevas clases y de cómo el cambio afecta a las clases existentes.

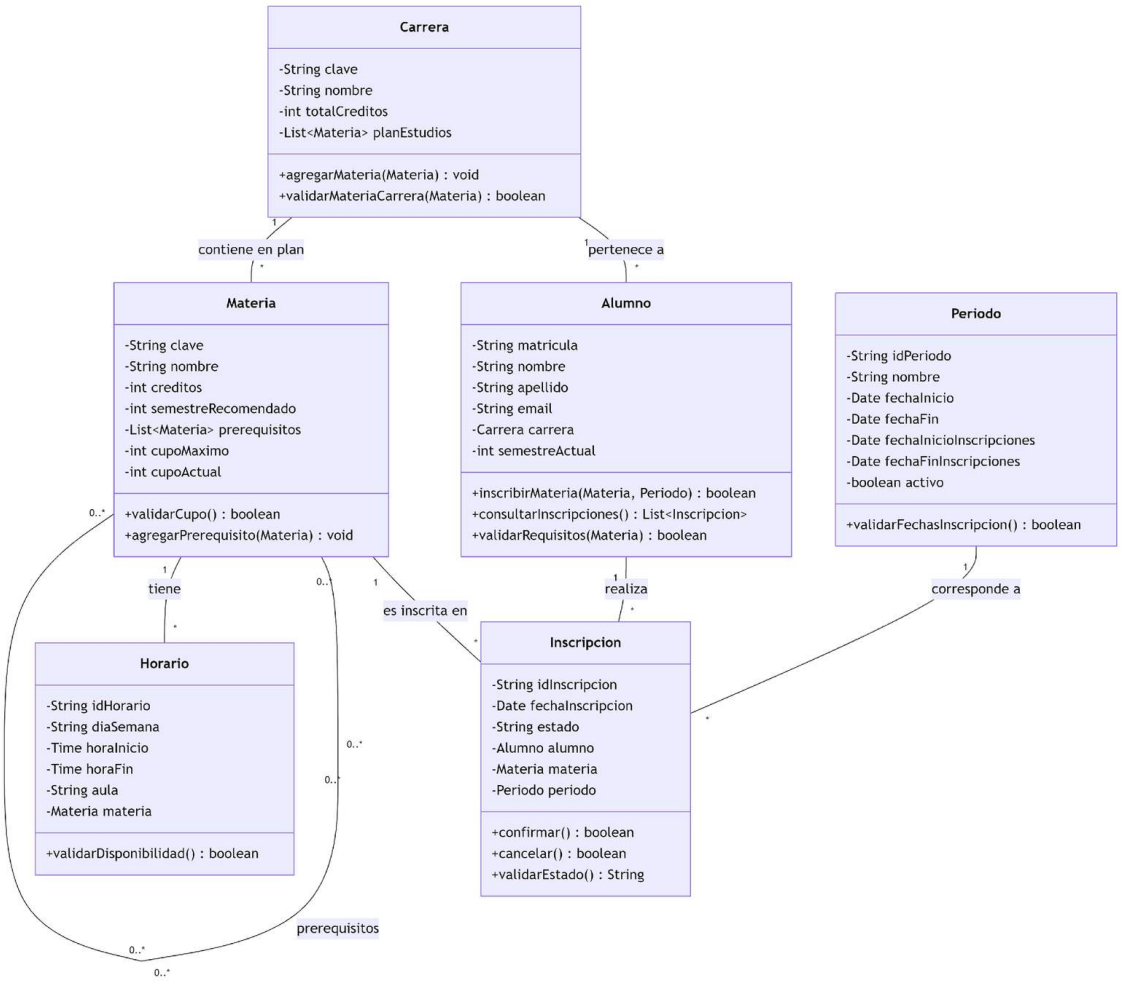
En el siguiente ejemplo, se puede observar que, usando el *prompt* de DeepSeek (empresa china de inteligencia artificial que desarrolla y publica modelos de lenguaje grandes, *large language models*, de código abierto, https://chat.deepseek.com/), se le pregunta:

Figura 5. Ejemplo de consulta en DeepSeek

Fuente: captura de pantalla de DeepSeek (High-Flyer, 2025)

También se emplearán modelos de ChatGPT y Copilot, permitiendo ver cómo las distintas opciones de estas herramientas resultan de gran utilidad. Como se observa en la figura 6, el resultado es el siguiente:

Figura 6. Imagen generada en DeepSeek



Fuente: captura de pantalla de DeepSeek (High-Flyer, 2025)

Es preciso aclarar que las herramientas de IA representan una gran ayuda y, desde esta cátedra, se alienta su uso para lograr un aprendizaje más rápido y práctico. El alumno debe esmerarse en aprovechar estas tecnologías, teniendo siempre presente que son herramientas que requieren de un criterio experto para producir resultados de mayor calidad.

Componentes clave de los diagramas de clases UML

Los diagramas de clases UML contienen clases y sus interacciones. Cada clase se representa en un rectángulo que incluye, de arriba abajo: el nombre de la clase, sus atributos y sus métodos. Solo es necesario el nombre de la clase; el nivel de detalle requerido determina si se muestran los atributos y los métodos.

Las interacciones en un diagrama de clases UML expresan las relaciones entre las clases mediante líneas y puntas de flecha. Estas tienen significados específicos para diferenciar los tipos de vínculos, los cuales incluyen la herencia, así como las relaciones bidireccionales y unidireccionales. Asimismo, las clases pueden agruparse en paquetes de clases estrechamente relacionadas.

Anotaciones del diagrama de clases UML

«La notación básica en un diagrama de clases UML incluye las etiquetas de los miembros de la clase, su visibilidad y los posibles paquetes. Por lo general, estos diagramas siguen convenciones de estilo como «camelCase», en el que la primera letra de la segunda palabra y de las siguientes se escribe en mayúscula, sin espacios entre ellas».

Clases

El nombre de una clase refleja la denominación de un objeto o entidad. Este se ubica en el centro de la sección superior, en negrita y con la inicial en mayúscula. En el modelo, la sección del nombre es la única obligatoria; las otras dos son opcionales, según el objetivo y el punto de vista del diagrama.

Figura 7. Representación de Clases

Fuente: elaboración propia.

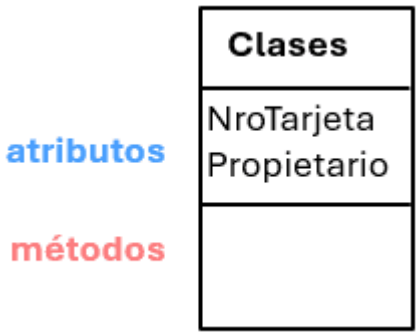
Atributos

La sección central del modelo incluye los atributos que describen las características estáticas de una clase, las cuales se aplicarán a las instancias de los objetos definidos por ella.

Por ejemplo, en un diagrama de clases diseñado para visualizar el funcionamiento de

un cajero automático, los atributos de la clase «Tarjeta de débito» incluirían el número de tarjeta y el propietario; así, cada instancia contará con sus propios datos identificativos.

Figura 8. Atributos de la clase

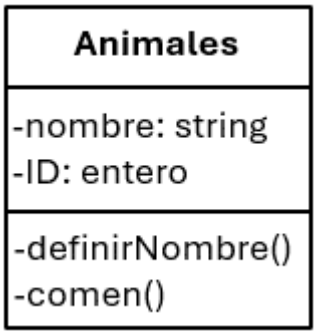


Fuente: elaboración propia.

Métodos (operaciones y comportamientos)

La tercera sección del modelo representa lo que la clase «hace»; es decir, las acciones que podrá realizar la instancia del objeto. Por ejemplo, la clase «Animal», que posee los atributos «nombre» e «ID», tendría como métodos «definirNombre» y «comer».

Figura 9. Métodos de la clase



Fuente: elaboración propia.

Visibilidad

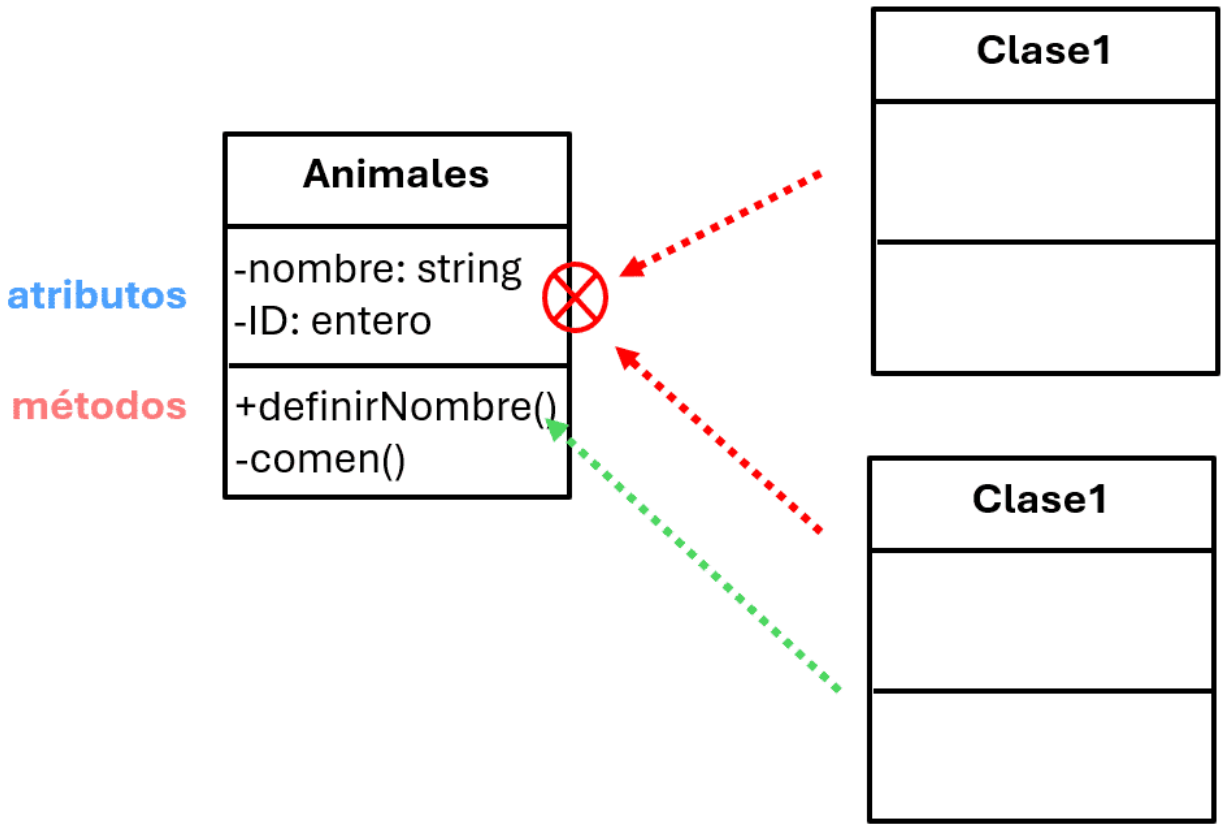
Los signos más (+) y menos (−) representan si los atributos y operaciones de una clase son públicos (visibles desde cualquier parte del sistema) o privados (visibles solo dentro de la clase). La almohadilla o signo de numeral (#) indica visibilidad protegida. La visibilidad define la accesibilidad de dicho atributo o método. En resumen, esta queda definida por el símbolo inicial, como se muestra en la siguiente figura:

Figura 10. Visibilidad

Fuente: elaboración propia.

En la figura 11, se puede observar cómo los atributos definidos como privados (−) no son accesibles desde otros niveles, mientras que el único elemento que sí lo es resulta ser el método «definirNombre», el cual es público (+).

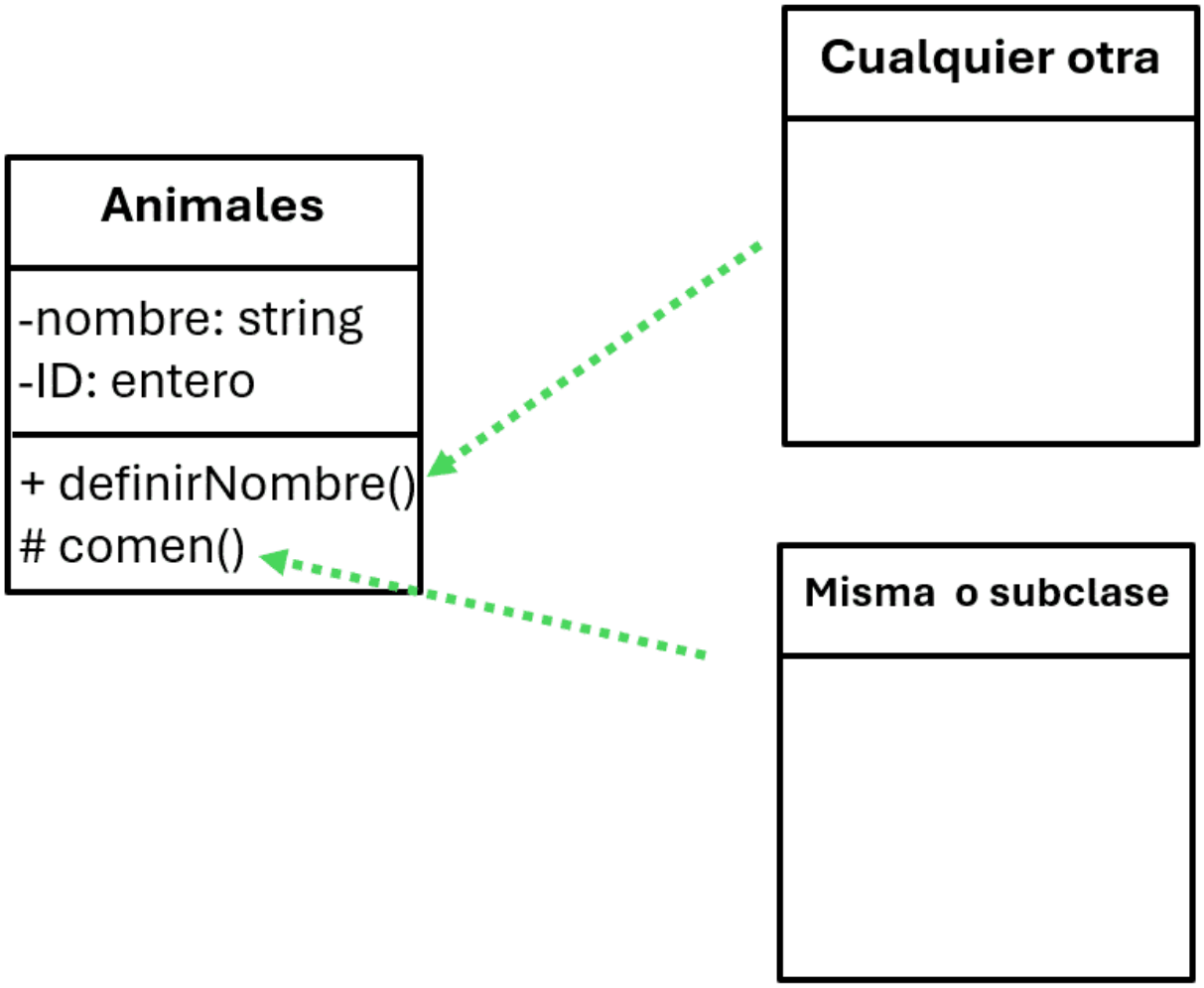
Figura 11. Ejemplo de visibilidad



Fuente: elaboración propia.

Cuando se habla de protegido (#) el método o atributo puede ser accedido desde la misma clase o una subclase. En el ejemplo de la figura 12 se aprecia el acceso al método protegido definirNombre() en la misma clase (Animales) o desde una subclase que se expanda o herede de Animales, también se muestra el acceso de cualquier otra clase al método comen().

Figura 12. Ejemplo de visibilidad protegida



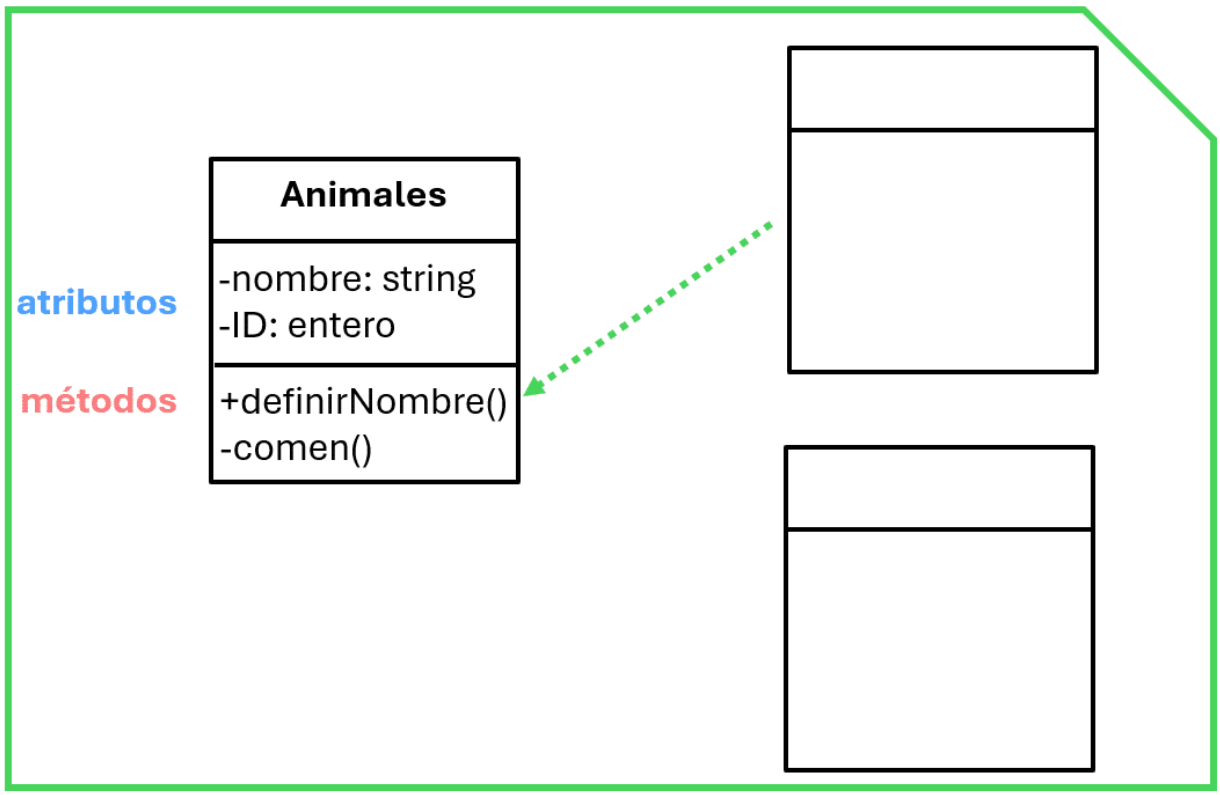
Fuente: elaboración propia.

Paquete

Cuando se trata de la visibilidad protegida (#), se puede acceder al método o atributo desde la misma clase o desde una subclase. En el ejemplo de la figura 13, se aprecia el acceso al método protegido «definirNombre()» en la misma clase («Animal») o desde una subclase que herede de esta;

también se muestra el acceso de cualquier otra clase al método público «comer()».

Figura 13. Paquete en UML



Fuente: elaboración propia.

Relaciones en los diagramas de clases UML

Los modelos de diagramas de clases UML se conectan visualmente mediante líneas, anotaciones textuales y números de multiplicidad. La siguiente lista incluye las

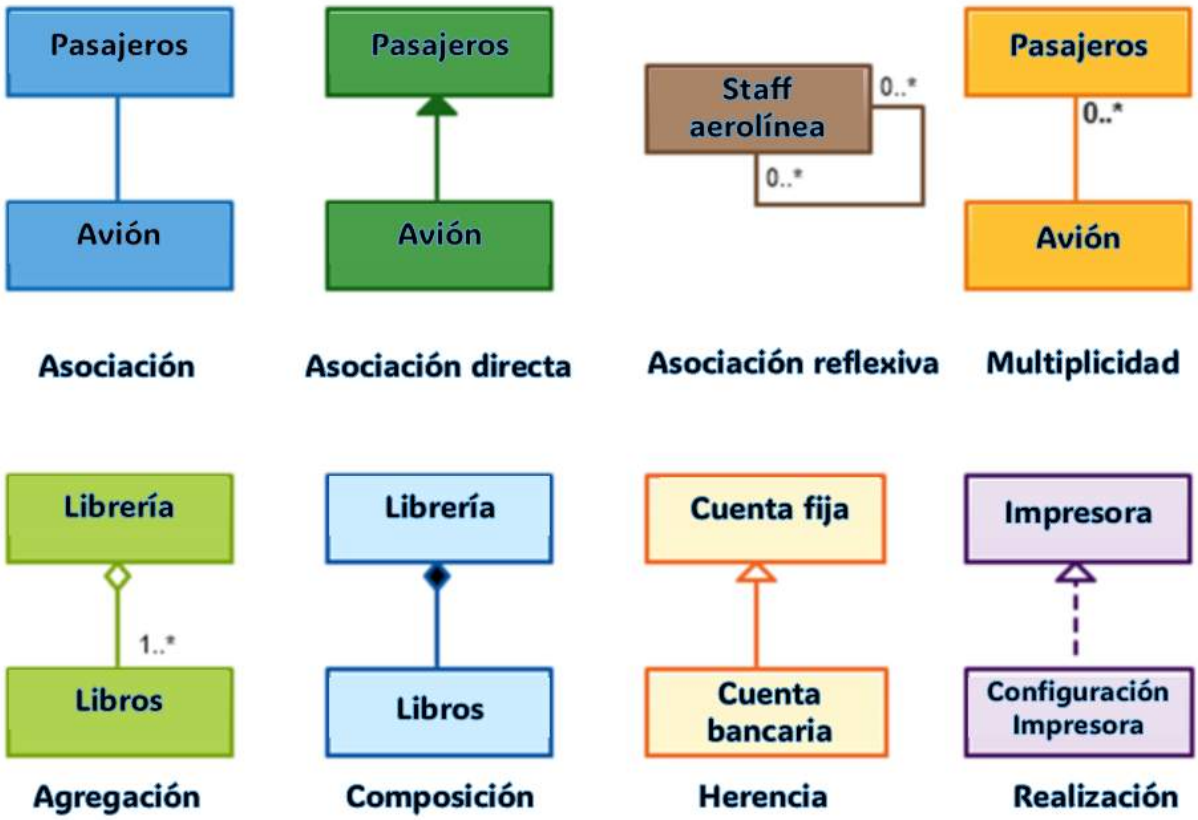
notaciones básicas de relación necesarias para crear un diagrama de clases de un sistema.

Las clases se interrelacionan de manera específica a través de diferentes tipos de conexiones lógicas. En UML, las relaciones posibles son las siguientes:

- Asociación
- Asociación dirigida
- Asociación reflexiva
- Multiplicidad
- Agregación
- Composición
- Herencia o generalización
- Realización

La siguiente imagen resume las relaciones mencionadas:

Figura 14. Las relaciones en los diagramas de clases



Fuente: elaboración propia.

Todas las notaciones de relación, a excepción de la de «Realización», implican un nombre escrito junto a la línea y al lado de la clase a la que pertenecen. Mantén todos los tamaños de texto equilibrados y las líneas ordenadas para conseguir un diagrama fácil de leer.

Asociación

Una asociación vincula dos clases mediante una conexión lógica; es la notación de relación más utilizada y se representa con una línea recta sólida. Las asociaciones son intrínsecamente bilaterales, lo que implica que ambas clases conocen la existencia de la otra. En UML, la asociación es una relación entre dos o más clasificadores (como las clases) que establece una conexión estructural. Esta relación indica que los objetos de una clase interactúan con los de otra, y se representa gráficamente mediante una línea continua que une ambos elementos.

Figura 15. Asociación

Fuente: elaboración propia.

La asociación n-aria define aquellas relaciones que vinculan a más de dos clases. La variante que sigue a la binaria es la asociación ternaria. Gráficamente, estas notaciones se representan mediante líneas que convergen en un rombo vacío situado en el punto de conjunción.

Asociación dirigida

Se refiere a una relación unidireccional representada por una línea con una punta de flecha. Esta indica el sentido del flujo o la navegabilidad entre las clases. Por ejemplo, un pasajero y una aerolínea pueden estar vinculados de esta forma, tal como se muestra a continuación:

Figura 16. Asociación dirigida

Fuente: elaboración propia.

Asociación reflexiva

La asociación reflexiva es un tipo de relación unilateral en los diagramas de clase. Se representa mediante una línea que forma un bucle sobre la misma clase. Su finalidad es visualizar una relación entre una instancia y otra instancia de esa misma clase.

Esto ocurre cuando una clase puede asumir múltiples funciones o responsabilidades. Por ejemplo, en el personal de un aeropuerto, un empleado podría ser piloto, ingeniero,

despachador o técnico de mantenimiento. Si el técnico es supervisado por el ingeniero, y ambos pertenecen a la clase «Personal», existiría una relación de gestión entre dos instancias de una misma clase.

Figura 17. Asociación reflexiva

Fuente: elaboración propia.

Multiplicidad

La multiplicidad es un indicador numérico situado en los extremos de una línea de relación. Esta define cuántas instancias de una clase participan en una relación con una instancia de otra clase. Es la notación utilizada para representar la cardinalidad dentro del modelo. Por ejemplo, una flota puede incluir varios aviones, mientras que un avión comercial puede contener de cero a muchos pasajeros. En un diagrama, la notación «0...*» significa «de cero a muchos». Los valores de multiplicidad más comunes se denotan de la siguiente manera:

Tabla 1. Multiplicidad

0...1 = Cero o uno 1 = Solo uno 0... = Cero o más 1...* = Uno o más 3 = Solo tres 0...5 = De cero a cinco 5...15 = De cinco a quince	

Fuente: elaboración propia.

Agregación y composición

La agregación y la composición son tipos específicos de relación de asociación.

La **agregación** vincula las clases mediante una línea recta y un rombo (o diamante) vacío junto a la clase «padre» o contenedora. En este tipo de relación, las clases «hijas» existen de forma independiente: si se elimina la clase padre, la clase hija permanece.

Se refiere a la formación de una clase como resultado de una colección de otras. Por ejemplo, una «biblioteca» está formada por libros, entre otros materiales; sin embargo, los libros seguirán existiendo aunque la biblioteca desaparezca. Por tanto, en la agregación, las clases contenidas no dependen del ciclo de vida del contenedor. Para representarla en un diagrama, se debe trazar una línea desde la clase padre hasta la clase hija con un rombo vacío cerca de la primera.

Figura 18. Agregación

Fuente: elaboración propia.

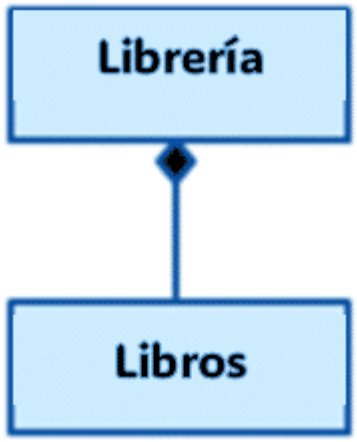
Composición

La composición representa la relación inversa a la agregación. Cuando dos clases se conectan mediante una línea recta con un rombo relleno, ambas dejan de ser independientes: si se elimina la clase «padre», también desaparece la clase «hija».

Esta relación es muy similar a la agregación, con la diferencia clave de que enfatiza la dependencia total de la clase contenida respecto al ciclo de vida de la contenedora; es decir, la clase contenida se elimina cuando la contenedora es destruida. Por ejemplo, el bolsillo lateral de una «bandolera» dejará de existir una vez que esta última se destruya. Para

representar esta relación en un diagrama UML, se utiliza una línea que conecta ambas clases, con un rombo relleno adyacente a la clase contenedora.

Figura 19. Composición



Fuente: elaboración propia.

Herencia y generalización

La notación de herencia o generalización visualiza una relación entre clases que comparten propiedades mediante herencia. Esta relación se representa con una línea recta y un triángulo vacío que apunta a la clase base o independiente.

Se refiere a un tipo de vínculo en el que una clase (hija) hereda funcionalidades de otra (padre). En otras palabras, la

clase «hija» es un tipo específico de la clase «padre». Para mostrar la herencia en un diagrama UML, se traza una línea sólida desde la subclase (hija) hasta la superclase (padre) utilizando una punta de flecha sin rellenar.

Figura 20. Herencia

Fuente: elaboración propia.

Realización

Una relación de realización vincula una clase que implementa el comportamiento definido por otra. En este modelo, los elementos se denominan «proveedor» (la clase que realiza la implementación) y «cliente» (el elemento que especifica el comportamiento). La notación de realización se

representa mediante una línea discontinua con un triángulo vacío en su extremo.

Esta relación denota la ejecución de una funcionalidad definida previamente. Para mostrarla en UML, se traza una línea discontinua con una punta de flecha vacía desde la clase que implementa la función hacia la interfaz o clase que la define. Por ejemplo, las preferencias de impresión establecidas en una interfaz de configuración son implementadas físicamente por la impresora.

Figura 21. Realización

Fuente: elaboración propia.

Herramienta *online* de UML

Existen diversas herramientas que permiten desarrollar diagramas de clases **UML**; una de ellas es **Lucidchart**, que

ofrece una versión gratuita (https://www.lucidchart.com). El alumno puede elegir cualquier aplicación para practicar; la que se presenta aquí es solo una de las opciones disponibles.

Lucidchart es una aplicación de diagramación basada en la web que permite crear y colaborar en tiempo real en documentos visuales como diagramas de flujo, organigramas, mapas mentales y modelos **UML**. Al ser una herramienta en la nube, facilita la comunicación

visual y el trabajo en equipo, independientemente de la ubicación de sus integrantes. Sus principales características son las siguientes:

- **Colaboración en tiempo real**. Permite que múltiples usuarios trabajen simultáneamente en el mismo documento y visualicen los cursores de los demás.
- **Versatilidad de diagramas**: se utiliza para crear una amplia variedad de gráficos, incluyendo mapas de procesos, diagramas técnicos y maquetas (*wireframes*).
- **Acceso multiplataforma**: al funcionar a través del navegador, permite el acceso desde

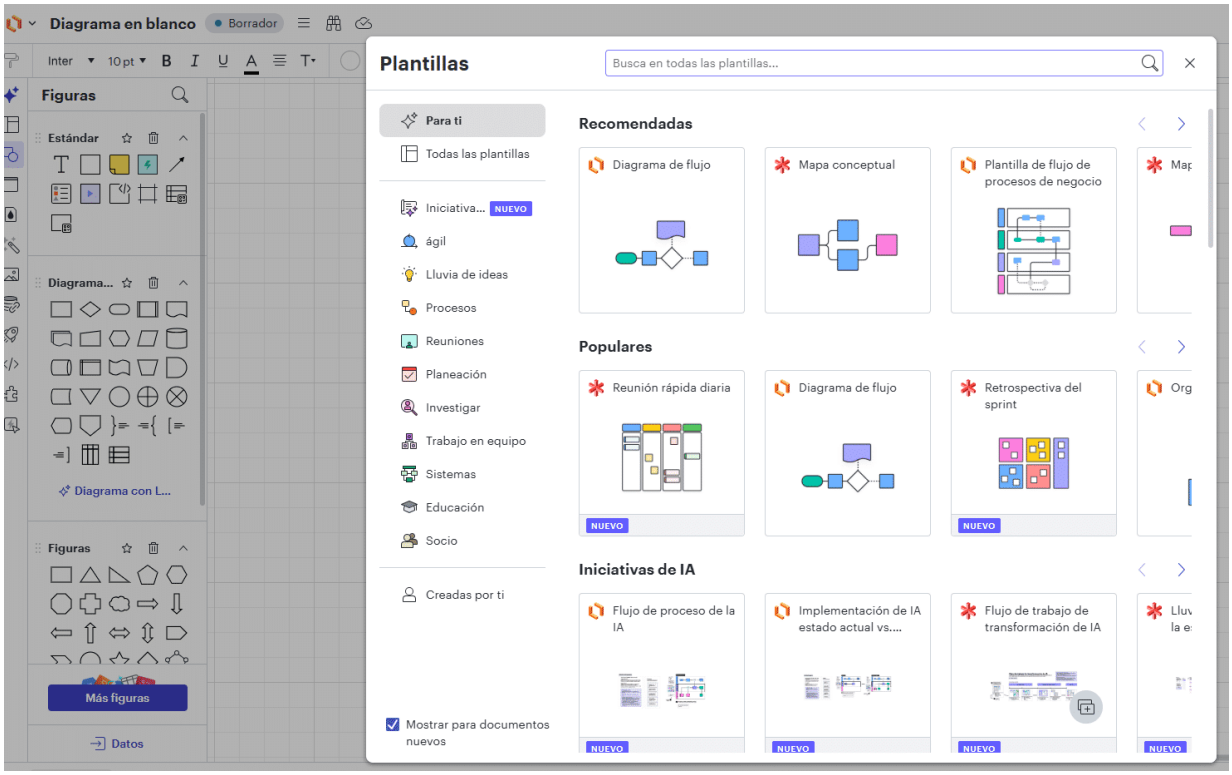
diferentes sistemas operativos sin necesidad de instalación.

- **Funciones avanzadas**: ofrece la importación de datos para generar diagramas automáticamente (por ejemplo, desde hojas de cálculo).

- **Uso empresarial y educativo**: es empleada por empresas para visualizar infraestructuras y procesos; asimismo, ofrece cuentas gratuitas para estudiantes y educadores.

La siguiente figura muestra el formato de presentación de la interfaz una vez realizado el registro:

Figura 22. Lucidchart para desarrollar diagramas de clases UML



Fuente: captura de pantalla de Lucidchart (https://www.lucidchart.com).

Con el fin de profundizar en el manejo de esta herramienta, se sugiere consultar, analizar y estudiar diversos tutoriales. Como ejemplo, se proponen los siguientes (aunque el alumno puede optar por cualquier otra fuente de aprendizaje):

Lucidchart, (s.f.). *Tutorial de diagrama de clases UML*. https://www.lucidchart.com/pages/es/tutorial-de-diagrama-de-clases-uml

Lucid Software Español [Lucid Software Español], (s.f.). *Cómo usar Lucidchart (en 3 minutos)* [video]. YouTube. https://www.youtube.com/watch?v=8VqlzBxjGJw

EducaTIC [EducaTIC], (s.f.). *Tutorial Lucidchart 2021 | Español | Crea diagramas, Organigramas, mapas mentales, diseños UML...*[video]. YouTube. https://www.youtube.com/watch?v=-wgG0oH8ml0

Calidad Académica Posgrado ESAN [Calidad Académica Posgrado ESAN], (s.f.). *Video tutorial Lucidchart* [video]. YouTube. https://www.youtube.com/watch?v=F5bdEFG5TG0

CONTINUAR

Referencias

Lucid Software Español [Lucid Software Español], (s.f.). *Tutorial - Diagrama de Clases UML* [video]. YouTube. https://www.youtube.com/watch?v=Z0yLerU0g-Q

Lucidchart, (s.f.). *Tutorial de diagrama de clases UML*. https://www.lucidchart.com/pages/es/tutorial-de-diagrama-de-clases-uml

Lucid Software Español [Lucid Software Español], (s.f.). *Cómo usar Lucidchart (en 3 minutos)* [video]. YouTube. https://www.youtube.com/watch?v=8VqlzBxjGJw

EducaTIC [EducaTIC], (s.f.). *Tutorial Lucidchart 2021 | Español | Crea diagramas, Organigramas, mapas mentales, diseños UML...*[video]. YouTube. https://www.youtube.com/watch?v=-wgG0oH8ml0

Calidad Académica Posgrado ESAN [Calidad Académica Posgrado ESAN], (s.f.). *Video tutorial Lucidchart* [video]. YouTube. https://www.youtube.com/watch?v=F5bdEFG5TG0

CONTINUAR