UD4.3. Herramientas para generar diagramas de clase, generación de código e ingeniería inversa

1. Herramientas para la generación de Diagramas de Clases

En el desarrollo de software, los diagramas de clases son una herramienta fundamental para modelar y visualizar la estructura de un sistema. Estos diagramas permiten definir las clases que compondrán el software, sus atributos, métodos y relaciones. Para crearlos, existen diversas **herramientas** en el mercado, que van desde soluciones simples y **genéricas** hasta herramientas **altamente especializadas** para diagramación y modelado en UML (Lenguaje Unificado de Modelado). En este apartado exploraremos algunas de estas herramientas, sus características principales y sus aplicaciones en distintos contextos.

Las herramientas para diagramas de clases pueden dividirse en dos grandes categorías: las genéricas, que no están diseñadas específicamente para UML, y las especializadas, que ofrecen un soporte más completo para este tipo de modelado.

1.1. Herramientas Genéricas

Una de las herramientas genéricas más populares es **Draw.io**, una plataforma en línea gratuita que permite la creación de una amplia variedad de diagramas, incluyendo UML. Al ser accesible desde cualquier navegador web, Draw.io es una opción ideal para quienes buscan facilidad de uso y no necesitan funcionalidades avanzadas como la generación de código. Sin embargo, su biblioteca de símbolos UML es limitada, lo que puede dificultar la creación de diagramas muy complejos. Esta herramienta destaca por su integración con servicios en la nube como Google Drive y GitHub, facilitando el trabajo colaborativo.

Otra herramienta de uso general es **Microsoft Visio**, ampliamente conocida en el ámbito empresarial. Visio cuenta con una extensa biblioteca de formas y diagramas que la convierten en una herramienta versátil, aunque su soporte para UML es menos robusto. A pesar de esto, su popularidad radica en la familiaridad que ofrece a los usuarios de Microsoft Office y su capacidad para integrarse con otros productos de esta suite. Sin embargo, el alto coste de esta herramienta puede ser un obstáculo para estudiantes y pequeñas empresas.

1.2. Herramientas Especializadas en UML

En el campo de las herramientas diseñadas específicamente para UML, **Visual Paradigm** se presenta como una de las opciones más potentes y completas. Esta herramienta no solo

permite la creación de diagramas UML, sino que también ofrece funcionalidades avanzadas como la generación de código en múltiples lenguajes de programación y la ingeniería inversa para generar diagramas a partir de código existente. Es particularmente adecuada para proyectos de gran escala, aunque su curva de aprendizaje y su coste pueden desanimar a los principiantes o a quienes trabajan en proyectos más pequeños.

StarUML, en cambio, es una solución más ligera que combina un enfoque minimalista con herramientas esenciales para diagramas UML. Aunque no cuenta con todas las funcionalidades avanzadas de Visual Paradigm, su interfaz intuitiva y su capacidad de generar código en lenguajes como Java, C++ y Python la convierten en una opción atractiva para estudiantes y desarrolladores que trabajan en proyectos de mediana envergadura.

1.3. Herramientas Open Source y Gratuitas

En el ámbito del software libre, herramientas como **PlantUML** han ganado popularidad. PlantUML se distingue por su enfoque basado en texto: los diagramas se generan escribiendo una sintaxis específica que describe las clases, relaciones y otros elementos. Este enfoque es ideal para desarrolladores acostumbrados a trabajar con código, ya que permite integrar fácilmente los diagramas en IDEs como IntelliJ IDEA, Visual Studio Code y Eclipse. Sin embargo, este método puede resultar menos intuitivo para quienes prefieren un enfoque visual y directo.

De manera similar, **Mermaid** es otra herramienta basada en texto que se utiliza ampliamente para generar diagramas directamente en Markdown, lo que la hace especialmente útil en plataformas de documentación como GitHub y GitLab. Aunque no está tan enfocada en UML como PlantUML, su simplicidad y su capacidad de integrarse en flujos de trabajo modernos la convierten en una herramienta valiosa para la creación rápida de diagramas.

1.4. Herramientas Online y Colaborativas

En un mundo cada vez más colaborativo, las herramientas en línea como **Lucidchart** ofrecen una experiencia integral para la creación de diagramas. Lucidchart combina una interfaz intuitiva con opciones avanzadas para trabajo en equipo, lo que permite a varios usuarios trabajar simultáneamente en un diagrama. Su modelo de suscripción puede ser un inconveniente, pero su capacidad de integración con herramientas de productividad como Slack y Google Workspace la hacen especialmente útil para equipos distribuidos.

1.5. Tabla resumen con las distintas herramientas de modelado UML

Herramienta	Categoría	Características	Ventajas	Desventajas	Precio
Draw.io	Genérica	Herramienta online para diagramas diversos, incluyendo UML.	Gratuita, fácil de usar, colaboración en la nube.	Biblioteca UML limitada, sin generación de código.	Gratis
Microsoft Visio	Genérica	Herramienta de diagramación empresarial con soporte básico para UML.	Familiar para usuarios de Office, amplia biblioteca de diagramas.	Costosa, soporte UML limitado.	Pago
Visual Paradigm	Especializada en UML	Creación de diagramas UML, generación de código, ingeniería inversa, y trabajo colaborativo.	Funcionalidades completas, ideal para proyectos grandes.	Costosa, curva de aprendizaje alta.	Freemium/Pago
StarUML	Especializada en UML	Herramienta ligera con soporte para UML y generación de código en múltiples lenguajes.	Interfaz intuitiva, adecuada para estudiantes y proyectos medianos.	Menos funciones avanzadas en comparación con Visual Paradigm.	Pago
PlantUML	Open Source/Gratuita	Generación de diagramas mediante sintaxis de texto.	Integración con IDEs, ideal para desarrolladores acostumbrados al código.	Menos intuitiva para principiantes, no visual.	Gratis/Open Source
Mermaid	Open Source/Gratuita	Creación de diagramas mediante texto, especialmente útil en	Fácil integración en Markdown, rápida creación de diagramas básicos.	Enfoque más limitado en UML que PlantUML.	Gratis/Open Source

Herramienta	Categoría	Características	Ventajas	Desventajas	Precio
		plataformas de documentación.			
Lucidchart	Online/Colaborativa	Plataforma online con opciones avanzadas para colaboración y diagramas UML.	Trabajo colaborativo, integración con herramientas como Google Workspace.	Requiere suscripción, dependiente de internet.	Freemium/Pago

1.6. Elección de la Herramienta Adecuada

La elección de una herramienta para diagramas de clases depende de varios factores, como el alcance del proyecto, el nivel de experiencia del usuario y las funcionalidades requeridas. Mientras que herramientas genéricas como Draw.io o Visio pueden ser suficientes para diagramas básicos, proyectos más complejos pueden beneficiarse de soluciones especializadas como Visual Paradigm o StarUML. Por otro lado, desarrolladores interesados en automatizar y documentar sus proyectos pueden encontrar en PlantUML y Mermaid herramientas que se ajusten a sus necesidades.

En cualquier caso, conocer las fortalezas y limitaciones de cada herramienta es clave para seleccionar la que mejor se adapte a las necesidades del proyecto y del equipo.

2. Algunos tips interesantes

Es posible importar un diagrama de Mermaid en draw.io. Échale un ojo a este artículo.

En este artículo explica, muy simplificadamente, como crear un diagrama de clases para diferentes lenguajes (Java, C# y VB) en Visual Paradigm.

3. Generación de diagramas de clases con draw.io

Draw.io (ahora conocido como diagrams.net) es una herramienta de diagramación versátil y gratuita que permite crear una amplia variedad de diagramas, desde diagramas de flujo y mapas mentales hasta diagramas UML y diagramas de clases. Su interfaz intuitiva y su compatibilidad con aplicaciones en la nube como Google Drive y OneDrive lo hacen ideal tanto para principiantes como para usuarios avanzados.

3.1. Creación de un nuevo diagrama

3.1.1. Desde cero

1. Abre Draw.io en su versión web o la aplicación de escritorio.
2. Selecciona **"Crear nuevo diagrama"**.
3. Elige una plantilla o selecciona **"Blanco"** para empezar desde cero.
4. Guarda el archivo en tu ubicación preferida (en la nube, en servicios cono oneDrive o Google Drive, o local).

3.1.2. Desde un archivo existente

1. Ve a **Archivo > Abrir desde**.
2. Selecciona el origen del archivo: Google Drive, OneDrive, local o URL.
3. Navega hasta el archivo que deseas abrir y cárgalo en Draw.io.

3.1.3. Desde una plantilla

1. En el menú principal, selecciona **"Crear nuevo diagrama"**.
2. Explora las plantillas disponibles en las categorías como "Flujo de trabajo", "Red", "UML", entre otras.
3. Selecciona la plantilla que mejor se adapte a tus necesidades y personalízala.

3.2. Operaciones básicas

3.2.1. Añadir figuras

1. En el panel izquierdo, selecciona una figura de las bibliotecas disponibles (rectángulos, óvalos, etc.).
2. Arrástrala al lienzo y suéltala en la ubicación deseada.
3. Puedes utilizar la barra de búsqueda para encontrar figuras específicas.

Para crear los diagramas de clases, navega hasta la sección UML, despliegala, y encontrarás los bloques básicos necesarios (clases con atributos y métodos, clases vacías, interfaces, y atributos adicionales).

Pulsa el botón "+ Más formas" en la parte inferior izquierda de la pantalla y explora todos los bloques que, de manera nativa, proporciona draw.io.

3.2.2. Modificar propiedades de las figuras

1. Haz clic en la figura para seleccionarla.
2. Usa el panel derecho para:
    - Cambiar el color de relleno, borde y sombra.
    - Ajustar el tamaño y la posición.
    - Modificar el texto dentro de la figura.

3.2.3. Añadir flechas

Puedes añadir flechas desde el panel izquierdo (categoría "General"), o, directamente, ubicando el cursor en los extremos del bloque (laterales, parte superior, o parte inferior); aparecerán unas flechas, y al hacer click en ellas, aparecerá una flecha conectada al bloque y que podrás conectar a otro bloque o dejar sin conectar.

3.2.4. Modificar flechas y conexiones

1. Selecciona la flecha para:
    - Cambiar su estilo (continuo, punteado, etc.).
    - Cambiar su forma (recta, angulosa, curva...)
    - Ajustar el tipo de punta (flecha, rombo, triángulo, etc.).
    - Añadir texto descriptivo sobre la flecha. Haz doble click sobre ella y podrás introducir un texto.

3.2.5. Agrupar y alinear

- **Agrupar:** Selecciona varias figuras, haz clic derecho y elige **Agrupar**. Esto creará un único "ente" que podrás mover sin que los bloques se desalineen.

- **Alinear:** Usa las guías automáticas o el menú de alineación para distribuir uniformemente los elementos.

3.2.6. Creación de clases

1. Usa el bloque de **Clase** disponible en la biblioteca UML.
2. Define los atributos y métodos dentro del bloque de la clase. Puedes usar el formato:
    - **Atributos:** `- nombre: tipo`
    - **Métodos:** `+ nombreMétodo(parámetros): tipoRetorno`

3.2.7. Representación de relaciones

Como recordatorio, debemos representar las relaciones en los diagramas de clases de la siguiente manera:

1. **Asociación:** Usa una línea simple entre clases.
2. **Agregación:** Usa una línea con un rombo blanco en un extremo.
3. **Composición:** Usa una línea con un rombo negro en un extremo.
4. **Herencia:** Usa una línea con una punta de flecha blanca.
5. **Realización:** Usa una línea discontinua con una punta de flecha blanca.

Ejemplo:

Puedes descargar e importar este ejemplo de archivo drawio con diferentes tipos de clases, interfaces y relaciones entre ellas:

3.3. Exportar diagramas

Draw.io permite exportar los diagramas a varios formatos:

3.3.1. Formatos comunes

1. **PNG o JPEG:** Ideal para presentaciones rápidas.
2. **SVG:** Mantiene la calidad vectorial para ajustes posteriores.
3. **PDF:** Para compartir diagramas en documentos.
4. **XML:** Para guardar el diagrama y abrirlo posteriormente en Draw.io. (como el ejemplo mostrado previamente)

3.3.2. Proceso de exportación

1. Ve a **Archivo > Exportar como**.
2. Selecciona el formato deseado.
3. Configura opciones adicionales (resolución, fondo transparente, etc.).
4. Guarda el archivo.

3.4. Uso de Draw.io sin conexión a internet

3.4.1. Aplicación de escritorio

1. Descarga la aplicación desde la página oficial.
2. Instálala en tu sistema operativo.
3. La interfaz es idéntica a la versión web, con todas las funcionalidades offline.

3.4.2. Extensión de VSCode

1. Instala la extensión Draw.io Integration desde el marketplace de Visual Studio Code.
2. Crea o abre archivos con extensión `.drawio` o `.xml`.
3. Usa la extensión para editar diagramas directamente desde el editor.

4. Cómo Generar Código a partir de un Diagrama de Clases

Generar código a partir de un diagrama de clases es un proceso que traduce la representación conceptual y visual de un sistema a un código fuente funcional en un lenguaje de programación. Este enfoque ayuda a mantener una conexión directa entre el diseño y la implementación, lo que reduce errores y mejora la consistencia del desarrollo del software.

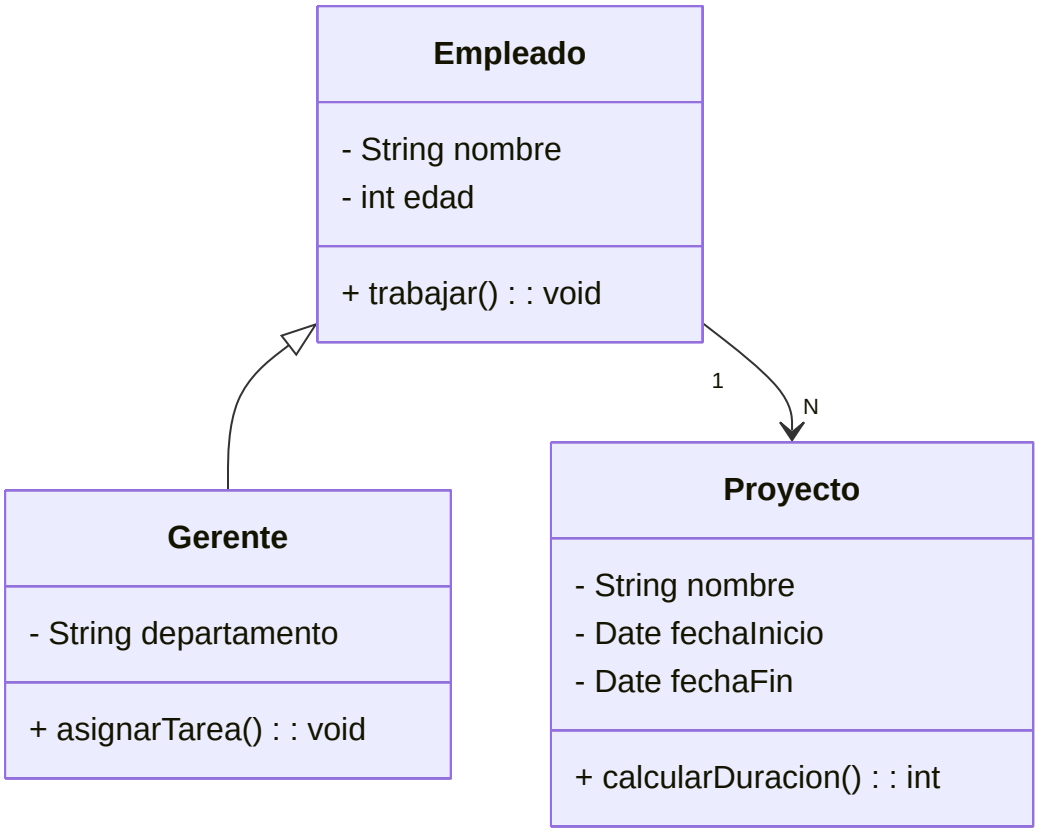
4.1. Comprensión del Proceso de Generación de Código

TBD

4.2. Ejemplo Práctico de Generación de Código

Diagrama de Clases:

Aquí tienes un diagrama UML simple que describe un sistema de gestión de empleados:



Código Generado (Java):

El código que se puede generar a partir del diagrama anterior es:

```java
// Clase Empleado
public class Empleado {
    private String nombre;
    private int edad;
    public void trabajar() {
        // Implementación
    }
}

// Clase Gerente
public class Gerente extends Empleado {
    private String departamento;
    public void asignarTarea() {
        // Implementación
    }
}

// Clase Proyecto
import java.util.Date;
public class Proyecto {
    private String nombre;
    private Date fechaInicio;
    private Date fechaFin;
    public int calcularDuracion() {
       // Implementación
       return 0;
    }
}
```

4.3. Beneficios de Generar Código desde UML

1. **Reducción de Errores**: Evita errores manuales en la transcripción del diseño al código.
2. **Consistencia**: Mantiene la alineación entre el modelo conceptual y la implementación.
3. **Ahorro de Tiempo**: Automatiza tareas repetitivas como la creación de clases y métodos.

5. Proceso de Ingeniería Inversa en IntelliJ IDEA

La **ingeniería inversa** en el contexto del desarrollo de software es el proceso de generar un diagrama UML, como el diagrama de clases, a partir de código fuente existente. Esta práctica

es útil para comprender sistemas complejos, documentar código o analizar la arquitectura del software.

IntelliJ IDEA, uno de los IDEs más avanzados, soporta esta funcionalidad a través de herramientas integradas o complementos.

Las herramientas relacionadas con la generación de diagramas de clases desde Intellij IDEA son parte de su versión ultimate, por lo que no podremos aplicarlas desde la versión community, pero, como estudiantes, se puede obtener una licencia gratuíta siguiendo los pasos descritos aquí.

A continuación, explicamos el proceso de generación de un diagrama de clases desde código Java paso a paso.

5.1. Preparación del Entorno

Antes de comenzar con la ingeniería inversa, asegúrate de cumplir con los siguientes requisitos:

1. **Instalación de IntelliJ IDEA**: Descarga e instala la versión Ultimate, ya que esta ofrece un soporte más completo para UML.
2. **Cargar el Proyecto**: Abre el proyecto Java o Kotlin cuyo diagrama deseas generar.
3. **Complementos Requeridos**:
    - Asegúrate de que el complemento **UML Support** está activado. Para verificar:
        - Ve a `File > Settings > Plugins`.
        - Busca "UML Support" y actívalo si no está instalado.

5.2. Generación del Diagrama UML

1. **Abrir el Proyecto**
   Abre tu proyecto en IntelliJ IDEA y asegúrate de que se compila correctamente.
2. **Acceso a la Funcionalidad de UML**
    - Navega al paquete o clase del proyecto del cual deseas generar el diagrama.
    - Haz clic derecho en el paquete o clase y selecciona:
      `Diagrams > Show Diagram`.
3. **Selección del Tipo de Diagrama**
    - IntelliJ IDEA generará un **diagrama UML de clases** para la selección actual (paquete, módulo o clase).

- Puedes personalizar el nivel de detalle del diagrama en: `View Options > Show/Hide Dependencies`, `Show/Hide Attributes`, etc.

4. **Explorar y Personalizar el Diagrama**
   - El diagrama generado incluirá clases, interfaces y relaciones como herencias, asociaciones y dependencias.
   - Puedes mover, organizar y ocultar elementos según sea necesario.
5. **Exportar el Diagrama**

   Una vez satisfecho con el diagrama, puedes exportarlo:
   - Ve a `File > Export Diagram`.
   - Selecciona el formato deseado, como PNG, SVG o PDF.

5.3. Ejemplo Práctico

Código de Ejemplo:

Supongamos que tienes el siguiente código en tu proyecto:

```java
// Clase Empleado
public class Empleado {
    private String nombre;
    private int edad;
    public void trabajar() {
        System.out.println("Trabajando...");
    }
}

// Clase Gerente
public class Gerente extends Empleado {
    private String departamento;
    public void asignarTarea() {
        System.out.println("Asignando tarea...");
    }
}

// Clase Proyecto
import java.util.Date;
public class Proyecto {
    private String nombre;
    private Date fechaInicio;
    private Date fechaFin;
    public int calcularDuracion() {
        return 0;
    }

}
```

Generación del Diagrama:

1. Haz clic derecho en el paquete que contiene las clases.
2. Selecciona `Diagrams > Show Diagram`.
3. Ajusta el nivel de detalle para mostrar atributos y métodos.

El diagrama generado incluirá las clases **Empleado**, **Gerente** y **Proyecto**, con relaciones entre ellas.

5.4. Ventajas de la Ingeniería Inversa con IntelliJ IDEA

1. **Documentación Automática**: Permite generar documentación visual de sistemas complejos de manera rápida.

2. **Comprensión de Sistemas Complejos**: Es ideal para analizar sistemas legados o proyectos desarrollados por otros equipos y permite visualizar rápidamente la estructura y relaciones entre clases.
3. **Análisis de Dependencias**: Facilita la identificación de relaciones y dependencias entre clases.
4. **Personalización**: Ofrece múltiples opciones para ajustar el nivel de detalle según las necesidades del proyecto.
5. **Compatibilidad**: Admite proyectos grandes y lenguajes como Java, Kotlin y más.

5.5. Limitaciones y Consideraciones

- **Nivel de Detalle**: El diagrama puede ser difícil de interpretar si el proyecto tiene demasiadas clases o relaciones.
- **Lógica de Negocio**: La ingeniería inversa no incluye detalles sobre la implementación de métodos o lógica interna.
- **Complejidad**: Proyectos mal estructurados generan diagramas desorganizados, lo que puede requerir ajustes manuales.

6. ¿Es buena práctica hacer uso de la ingeniería inversa?

6.1. Limitaciones y Riesgos

1. **Falta de Abstracción**: Los diagramas generados automáticamente reflejan el estado exacto del código. Esto puede incluir demasiados detalles irrelevantes y carecer de la abstracción necesaria para un diseño comprensible.
2. **Complejidad Visual**: En sistemas grandes, el diagrama resultante puede ser difícil de interpretar debido al exceso de elementos y relaciones.
3. **Diseño No Intencional**:
    - Generar un diagrama después de escribir el código no garantiza que siga principios de diseño sólido como SOLID o GRASP.
    - El diagrama refleja cómo está diseñado el sistema, no cómo debería estar diseñado.
4. **Dependencia de Herramientas**: El diagrama depende de la precisión de la herramienta utilizada para generarlo. Algunas relaciones o elementos clave podrían omitirse.

6.2. ¿Cuándo es una buena práctica?

- **Análisis de Código Existente**: Útil para comprender proyectos heredados o sistemas sin documentación previa.
- **Refactorización**: Ayuda a identificar dependencias o clases que pueden ser refactorizadas.
- **Documentación Complementaria**: Sirve como apoyo visual en proyectos que ya tienen un diseño establecido.

6.3. ¿Cuándo no es recomendable?

- **Diseño de Nuevos Sistemas**: El diseño debería preceder al código, y los diagramas deberían reflejar una arquitectura planificada. Generar diagramas después del código puede dar una falsa sensación de diseño.
- **Documentación Oficial**: Los diagramas generados automáticamente no siempre son claros y pueden no cumplir con los estándares de documentación esperados.
- **Sistemas Complejos y Desordenados**: En proyectos grandes o mal diseñados, los diagramas resultantes pueden ser caóticos e inútiles.

6.4. Buenas prácticas al usar ingeniería inversa

1. **Refina el Diagrama**: Elimina detalles innecesarios y ajusta el nivel de abstracción para que sea comprensible y útil.
2. **Combínalo con Documentación Manual**: Usa el diagrama como punto de partida, pero complementa con información de alto nivel, como patrones de diseño y decisiones arquitectónicas.
3. **Utiliza Herramientas de Calidad**: Herramientas como Visual Paradigm, StarUML o IntelliJ IDEA pueden generar diagramas más precisos y personalizables.
4. **Fomenta el Diseño Previo**: Utiliza ingeniería inversa solo como apoyo, no como un sustituto de un buen diseño previo al desarrollo.

Otros editores de interes

- Excalidraw: Herramienta de diagramación colaborativa y sencilla.
- Terrastruct: Terrastruct proporciona un lenguaje específico de dominio para diagramación, llamado D2