# **CASOS DE USO**

***Actividades***

1. *Tareas:* Las presentes tareas deben realizarse en orden -o no- prestando atención a la relación entre las lecturas realizadas.

1. Leer el Prefacio de UML Gota a Gota
2. Leer la sección “Cómo llegamos hasta aquí” de UML Gota a Gota y crea un documento que expanda lo escrito hasta nuestros días.
3. Aprende UML utilizando Visual Paradigm <https://www.visual-paradigm.com/guide/> y crea un resumen conciso de los conceptos más relevantes trabajados en clase. Explora los tutoriales y guías disponibles en el sitio web, enfocándote en los temas que se relacionan con los conceptos discutidos en clase. A medida que avances, toma notas y organiza un resumen estructurado de los conceptos clave de UML. Finalmente, comparte y discute tus hallazgos con tus compañeros de clase para mejorar la comprensión de UML y su aplicación en proyectos futuros.

2. Considere el ejercicio 2 del teórico 2.1:

*Se pide:*

c) Identifique los casos de uso del sistema y elabore un diagrama de casos de uso.

d) Escriba el escenario principal correspondiente a cada uno de ellos.

e) Escriba el desarrollo expandido para cada caso de uso del sistema (o sea, para cada uno de ellos, escriba una ficha que detalle actores, puntos de extensión y de inclusión, precondiciones, poscondiciones, flujo principal y flujos alternativos).

**🔹 Punto a) Prefacio de *UML Gota a Gota***

En el **prefacio** se explica que UML (*Unified Modeling Language*) nació como un esfuerzo de estandarización en los años 90, cuando existían muchos lenguajes y notaciones distintas para modelar software (Booch, OMT, OOSE, etc.). UML se planteó como un **lenguaje visual unificado** que sirviera a:

* Analistas, diseñadores y desarrolladores para comunicarse con un mismo “idioma gráfico”.
* Representar de manera clara sistemas complejos.
* Brindar diagramas que abarcan distintos puntos de vista (estructural, de comportamiento, de interacción).

El texto destaca que UML no es un **método de desarrollo**, sino un **lenguaje de modelado** que puede usarse con diferentes metodologías (ágiles, tradicionales, etc.).

En resumen: **El prefacio justifica la necesidad de UML como un estándar universal de modelado que facilite comunicación y documentación en proyectos de software.**

**🔹 Punto b) Sección *“Cómo llegamos hasta aquí”* y expandir hasta nuestros días**

En esa sección se relata la **evolución histórica**:

* Años 80–90: coexistían varios métodos de modelado orientado a objetos (Booch, OMT de Rumbaugh, OOSE de Jacobson).
* Mediados de los 90: Grady Booch, James Rumbaugh e Ivar Jacobson (“los tres amigos”) unifican sus notaciones bajo UML.
* 1997: UML es adoptado como estándar por la OMG (Object Management Group).

**Expansión hasta la actualidad (2025):**

* UML ha evolucionado en varias versiones: UML 1.x (1997–2003), UML 2.x (2005–actual).
* Hoy sigue siendo estándar de la OMG, con 14 tipos de diagramas (estructurales y de comportamiento).
* UML se combina con otros lenguajes/modelos:
  + **SysML** (Systems Modeling Language) para ingeniería de sistemas.
  + **BPMN** (Business Process Model and Notation) para procesos de negocio.
  + **ArchiMate** para arquitectura empresarial.
* En metodologías ágiles, UML se usa de manera **ligera**, sobre todo con **diagramas de casos de uso, de clases y de secuencia**, evitando la sobrecarga documental.
* Herramientas modernas (Visual Paradigm, Enterprise Architect, StarUML, PlantUML) permiten integrar UML con pipelines DevOps, generar código automáticamente o enlazar con historias de usuario.
* Aunque han surgido enfoques más ágiles y ligeros (como C4 Model o diagramas ad-hoc), UML sigue siendo una **referencia universal** en proyectos que requieren modelado formal o documentación estándar.

En resumen: **“Cómo llegamos hasta aquí” explica la unificación histórica de lenguajes; la expansión muestra que UML sigue vigente, pero adaptado a entornos ágiles y coexistiendo con nuevos lenguajes de modelado y notaciones modernas.**

## 🔹 Punto c) Resumen de UML a partir de Visual Paradigm

## 1. Introducción

El **Lenguaje Unificado de Modelado (UML, Unified Modeling Language)** es un estándar creado para representar de forma gráfica los sistemas de software y su comportamiento. No es una metodología de desarrollo, sino un **lenguaje visual** que facilita la comunicación entre analistas, diseñadores, programadores y clientes.

Su objetivo principal es **unificar** las distintas notaciones de modelado que existían en los años 90 y ofrecer un marco común para describir sistemas desde diferentes perspectivas.

## 2. Conceptos principales aprendidos en clase y reforzados con Visual Paradigm

### **Actores**

* Representan **roles externos** (usuarios, otros sistemas) que interactúan con el sistema.
* No son personas concretas, sino funciones o roles.

### **Casos de uso**

* Son los **objetivos o funcionalidades** que el sistema ofrece a los actores.
* Se representan con óvalos dentro del rectángulo del sistema.
* Deben escribirse con un nombre claro: verbo + sustantivo (ej. “Realizar pago”).

### **Relaciones**

* **Asociación**: línea simple entre actor y caso de uso.
* **Include (<<include>>)**: cuando un caso de uso **siempre incluye** la ejecución de otro (ej. “Realizar pago” incluye “Procesar pago”).
* **Extend (<<extend>>)**: cuando un caso de uso **puede extenderse opcionalmente** con otro (ej. “Agregar al carrito” extiende “Realizar pago”).
* **Generalización**: herencia entre actores o casos de uso (ej. Usuario y Administrador).

### **Diagramas UML más relevantes**

1. **Diagrama de Casos de Uso**: describe qué hace el sistema desde la perspectiva del usuario.
2. **Diagrama de Clases**: muestra la estructura estática (atributos, métodos y relaciones entre clases).
3. **Diagrama de Secuencia**: representa la interacción entre objetos en el tiempo, mostrando mensajes enviados y respuestas.
4. **Diagrama de Actividades**: describe flujos de procesos, útil para modelar lógica de negocio.

## 3. Buenas prácticas señaladas por Visual Paradigm

* Mantener los diagramas **simples y claros** (no sobrecargarlos).
* Usar nombres consistentes y fáciles de entender.
* Documentar cada caso de uso con escenarios principales y alternativos.
* Organizar diagramas de manera que los actores estén siempre **fuera** y los casos de uso **dentro** del límite del sistema.
* En proyectos ágiles, usar UML de forma **ligera**: centrarse en casos de uso y diagramas de secuencia para complementar historias de usuario.

## 4. Utilidad práctica de UML hoy

* Permite **comunicación efectiva** entre distintas partes de un proyecto.
* Ayuda a **documentar y analizar requerimientos**.
* Es útil tanto en proyectos tradicionales como ágiles (aunque se usa de manera más resumida en estos últimos).
* UML se integra con herramientas modernas como Visual Paradigm, PlantUML o StarUML, lo que facilita la generación de diagramas y su integración con procesos de desarrollo.

## 5. Conclusión

Aprender UML con Visual Paradigm permite reforzar la teoría con práctica visual e interactiva. El lenguaje sigue siendo relevante porque ofrece una **visión compartida y estandarizada** de sistemas complejos. Su aplicación en proyectos reales ayuda a reducir malentendidos, mejorar la documentación y dar soporte tanto a fases de análisis como de diseño y mantenimiento.

# **Resumen UML – Visual Paradigm (Apuntes rápidos)**

| **Tema** | **Conceptos clave** |
| --- | --- |
| **UML** | Lenguaje estándar de modelado visual, no es una metodología. Facilita comunicación y documentación. |
| **Actores** | Roles externos (usuarios, sistemas). Siempre fuera del límite del sistema. |
| **Casos de uso** | Funcionalidades/objetivos del sistema. Óvalos dentro del sistema. Nombre: verbo + sustantivo. |
| **Relaciones** | - **Asociación**: línea simple actor ↔ caso de uso. - **<<include>>**: caso de uso obligatorio. - **<<extend>>**: caso de uso opcional. - **Generalización**: herencia entre actores o casos. |
| **Diagramas UML usados** | - Casos de uso (requerimientos). - Clases (estructura estática). - Secuencia (interacción en el tiempo). - Actividades (flujos de procesos). |
| **Buenas prácticas** | - Diagramas simples y legibles. - Actores fuera, casos dentro. - Documentar escenarios. - Usar UML de forma ligera en proyectos ágiles. |
| **Aplicación actual** | - Complementa historias de usuario. - Documenta sistemas complejos. - Compatible con herramientas modernas (Visual Paradigm, PlantUML). |

## 2. Considere el ejercicio 2 del teórico 2.1:

*Se pide:*

c) Identifique los casos de uso del sistema y elabore un diagrama de casos de uso.

d) Escriba el escenario principal correspondiente a cada uno de ellos.

* **Buscar libro**: El cliente ingresa criterio de búsqueda → el sistema muestra lista de libros.
* **Consultar disponibilidad**: Cliente selecciona libro → el sistema muestra stock real.
* **Realizar compra**: Cliente selecciona libro → agrega dirección → confirma pago → el sistema genera orden.
* **Guardar dirección de envío**: Cliente guarda direcciones frecuentes → Sistema la asocia a su perfil, para próximas compras.
* **Dejar opinión**: Cliente califica libro comprado → sistema almacena reseña.
* **Recibir recomendaciones**: Sistema analiza historial del cliente → muestra sugerencias de libros.
* **Generar estadísticas**: Marketing consulta sistema → se generan reportes sobre búsquedas y carritos.
* **Cargar libro nuevo**: Administrador ingresa datos de un libro → sistema lo agrega al catálogo.
* **Editar libro**: Administrador modifica información → sistema actualiza.
* **Emitir reporte de ventas**: Administrador/Marketing consulta → sistema muestra informe (entrega reporte).

e) Escriba el desarrollo expandido para cada caso de uso del sistema (o sea, para cada uno de ellos, escriba una ficha que detalle actores, puntos de extensión y de inclusión, precondiciones, poscondiciones, flujo principal y flujos alternativos).

Desarrollo Expandido de Casos de Uso - Sistema Librería Online

# Caso de Uso: Buscar libro

|  |  |
| --- | --- |
| Actor principal | Cliente/Lector |
| Actores secundarios | - |
| Precondiciones | El sistema está disponible. |
| Postcondiciones | Se muestra lista de libros coincidentes. |
| Puntos de inclusión | - |
| Puntos de extensión | Consultar disponibilidad, Recibir recomendaciones |
| Flujo principal | El Cliente ingresa criterios de búsqueda (título, autor, género). El sistema procesa y muestra resultados. |
| Flujos alternativos | No se encuentran resultados → se muestra mensaje. |

# Caso de Uso: Consultar disponibilidad

|  |  |
| --- | --- |
| Actor principal | Cliente/Lector |
| Actores secundarios | Encargado de Atención |
| Precondiciones | El libro está en el catálogo. |
| Postcondiciones | El sistema muestra stock actualizado. |
| Puntos de inclusión | Incluido en 'Realizar compra'. |
| Puntos de extensión | Extiende 'Buscar libro'. |
| Flujo principal | El Cliente selecciona un libro. El sistema muestra disponibilidad. |
| Flujos alternativos | Stock = 0 → sistema ofrece notificación de falta. |

# Caso de Uso: Registrar/Iniciar sesión

|  |  |
| --- | --- |
| Actor principal | Cliente/Lector |
| Actores secundarios | - |
| Precondiciones | El cliente no ha iniciado sesión. |
| Postcondiciones | El cliente queda autenticado. |
| Puntos de inclusión | Incluido en 'Realizar compra'. |
| Puntos de extensión | - |
| Flujo principal | Cliente completa datos de registro o login. El sistema valida credenciales. El sistema confirma autenticación. |
| Flujos alternativos | Datos incorrectos → se muestra error. |

# Caso de Uso: Realizar compra

|  |  |
| --- | --- |
| Actor principal | Cliente/Lector |
| Actores secundarios | Sistema de Pago |
| Precondiciones | Cliente registrado, libro disponible. |
| Postcondiciones | Orden de compra confirmada, stock actualizado. |
| Puntos de inclusión | Registrar/Iniciar sesión, Consultar disponibilidad, Guardar dirección de envío, Confirmar compra |
| Puntos de extensión | Dejar opinión |
| Flujo principal | Cliente selecciona un libro y lo agrega al carrito. Cliente indica dirección de envío. Cliente confirma la compra. El sistema se comunica con el sistema de pago. El sistema registra la orden y muestra confirmación. |
| Flujos alternativos | Pago rechazado → se informa y se permite reintentar. Cliente cancela compra → se vacía el carrito. |

# Caso de Uso: Guardar dirección de envío

|  |  |
| --- | --- |
| Actor principal | Cliente/Lector |
| Actores secundarios | - |
| Precondiciones | Cliente registrado. |
| Postcondiciones | Dirección almacenada en el perfil. |
| Puntos de inclusión | Incluido en 'Realizar compra'. |
| Puntos de extensión | - |
| Flujo principal | Cliente ingresa dirección. El sistema la guarda en el perfil. |
| Flujos alternativos | - |

# Caso de Uso: Confirmar compra

|  |  |
| --- | --- |
| Actor principal | Cliente/Lector |
| Actores secundarios | Sistema de Pago |
| Precondiciones | Carrito con productos, dirección válida. |
| Postcondiciones | Compra registrada. |
| Puntos de inclusión | Incluido en 'Realizar compra'. |
| Puntos de extensión | - |
| Flujo principal | Cliente revisa los productos en el carrito. Cliente confirma compra. Sistema confirma pago y finaliza la transacción. |
| Flujos alternativos | - |

# Caso de Uso: Dejar opinión

|  |  |
| --- | --- |
| Actor principal | Cliente/Lector |
| Actores secundarios | - |
| Precondiciones | El libro ha sido comprado previamente. |
| Postcondiciones | Opinión registrada en el sistema. |
| Puntos de inclusión | - |
| Puntos de extensión | Extiende 'Realizar compra'. |
| Flujo principal | Cliente selecciona un libro comprado. Cliente escribe comentario y calificación. El sistema guarda la reseña. |
| Flujos alternativos | - |

# Caso de Uso: Recibir recomendaciones

|  |  |
| --- | --- |
| Actor principal | Cliente/Lector |
| Actores secundarios | - |
| Precondiciones | Cliente registrado o con historial de búsquedas. |
| Postcondiciones | Sistema muestra sugerencias personalizadas. |
| Puntos de inclusión | - |
| Puntos de extensión | Extiende 'Buscar libro'. |
| Flujo principal | El sistema analiza búsquedas y compras anteriores. El sistema muestra libros sugeridos. |
| Flujos alternativos | - |

# Caso de Uso: Registrar consulta/reclamo

|  |  |
| --- | --- |
| Actor principal | Cliente/Lector |
| Actores secundarios | Encargado de Atención |
| Precondiciones | Cliente registrado. |
| Postcondiciones | Consulta almacenada y derivada a atención. |
| Puntos de inclusión | - |
| Puntos de extensión | - |
| Flujo principal | Cliente redacta su consulta/reclamo. El sistema registra el caso. El Encargado de Atención lo recibe y responde. |
| Flujos alternativos | - |

# Caso de Uso: Generar estadísticas

|  |  |
| --- | --- |
| Actor principal | Responsable de Marketing |
| Actores secundarios | - |
| Precondiciones | Acceso autorizado. |
| Postcondiciones | Estadísticas generadas. |
| Puntos de inclusión | - |
| Puntos de extensión | - |
| Flujo principal | Marketing solicita informe. El sistema genera métricas de búsquedas y ventas. |
| Flujos alternativos | - |

# Caso de Uso: Cargar libro nuevo

|  |  |
| --- | --- |
| Actor principal | Administrador |
| Actores secundarios | - |
| Precondiciones | Acceso autorizado. |
| Postcondiciones | Nuevo libro disponible en catálogo. |
| Puntos de inclusión | - |
| Puntos de extensión | - |
| Flujo principal | Administrador ingresa datos del libro. El sistema lo registra en el catálogo. |
| Flujos alternativos | - |

# Caso de Uso: Editar información de libro

|  |  |
| --- | --- |
| Actor principal | Administrador |
| Actores secundarios | - |
| Precondiciones | El libro ya existe en el catálogo. |
| Postcondiciones | Datos actualizados. |
| Puntos de inclusión | - |
| Puntos de extensión | - |
| Flujo principal | Administrador selecciona libro. Edita la información. El sistema guarda cambios. |
| Flujos alternativos | - |

# Caso de Uso: Emitir reporte de ventas

|  |  |
| --- | --- |
| Actor principal | Administrador / Marketing |
| Actores secundarios | - |
| Precondiciones | Acceso autorizado. |
| Postcondiciones | Reporte generado. |
| Puntos de inclusión | - |
| Puntos de extensión | - |
| Flujo principal | El actor solicita reporte. El sistema calcula ventas. El sistema genera reporte. |
| Flujos alternativos | - |