

LENGUAJE UNIFICADO DE MODELADO



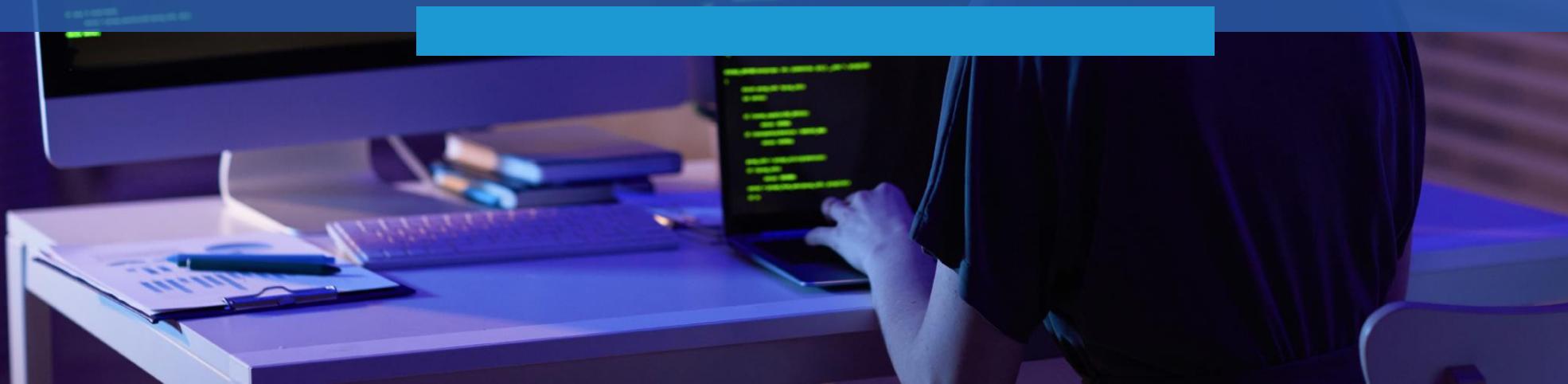
BIENVENIDA

Bienvenido(a) a la asignatura *Lenguaje Unificado de Modelado*, con la cual identificarás los elementos del metamodelo UML, así como sus reglas, mecanismos, las vistas arquitecturales definidas y conceptualizadas ante este lenguaje estándar para el modelado de software.

Al final, serás capaz de desarrollar modelos de diagramas estructurales, de comportamiento e interacción. Por último, identificarás las herramientas para el modelado de diagramas, reconociendo el valor intrínseco de cada uno.

UNIDAD 1

ELEMENTOS ESTRUCTURALES





TEMARIO

U1

1.1



Elementos del Metamodelo
de UML

1.2



Vistas Arquitecturales

INTRODUCCIÓN

La asignatura *Lenguaje Unificado de Modelado* tiene como objetivo formar al alumno en el uso de técnicas para el desarrollo de sistemas a través del Lenguaje Unificado de Modelado. Comprendido, principalmente, a través de sus diversos tipos de diagramas.

En esta primera unidad, aprenderás a identificar los elementos del metamodelo UML, así como sus reglas, mecanismos y las vistas arquitecturales para el desarrollo de sistemas.

A close-up photograph of a computer monitor. The screen displays two windows: one showing Python code and another showing terminal output. The code window contains several lines of Python code, likely related to image processing or machine learning, with syntax highlighting. The terminal window shows command-line interactions, including a command to run 'train.py' and its output to 'results.txt'. The monitor is a Dell model, as indicated by the logo at the bottom.

COMPETENCIAS A DESARROLLAR



1.

El alumno será capaz de reconocer los elementos, reglas y mecanismos del metamodelo UML, para el desarrollo de sistemas.

2.

El alumno será capaz de organizar los conceptos de alto nivel de UML en un conjunto de vistas y diagramas que presenta la arquitectura del desarrollo de sistemas.

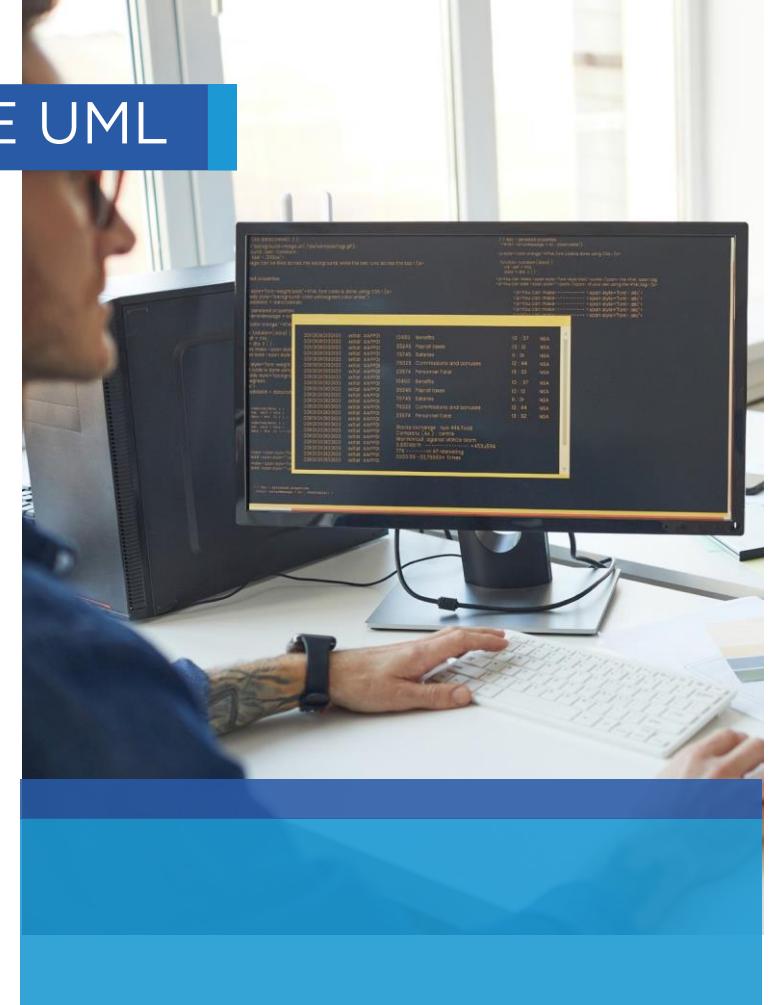
ELEMENTOS DEL METAMODELO DE UML

“El Lenguaje de Modelado Unificado (UML) es un lenguaje cuyo vocabulario y reglas se centran en la representación conceptual y física de un sistema”.

Booch, Jacobson y Rumbaugh.

Lenguaje = Notación + Reglas (sintácticas, semánticas)

En este sentido, UML ofrece vocabulario y reglas para crear y leer modelos bien formados que constituyen los planos de un sistema de software.





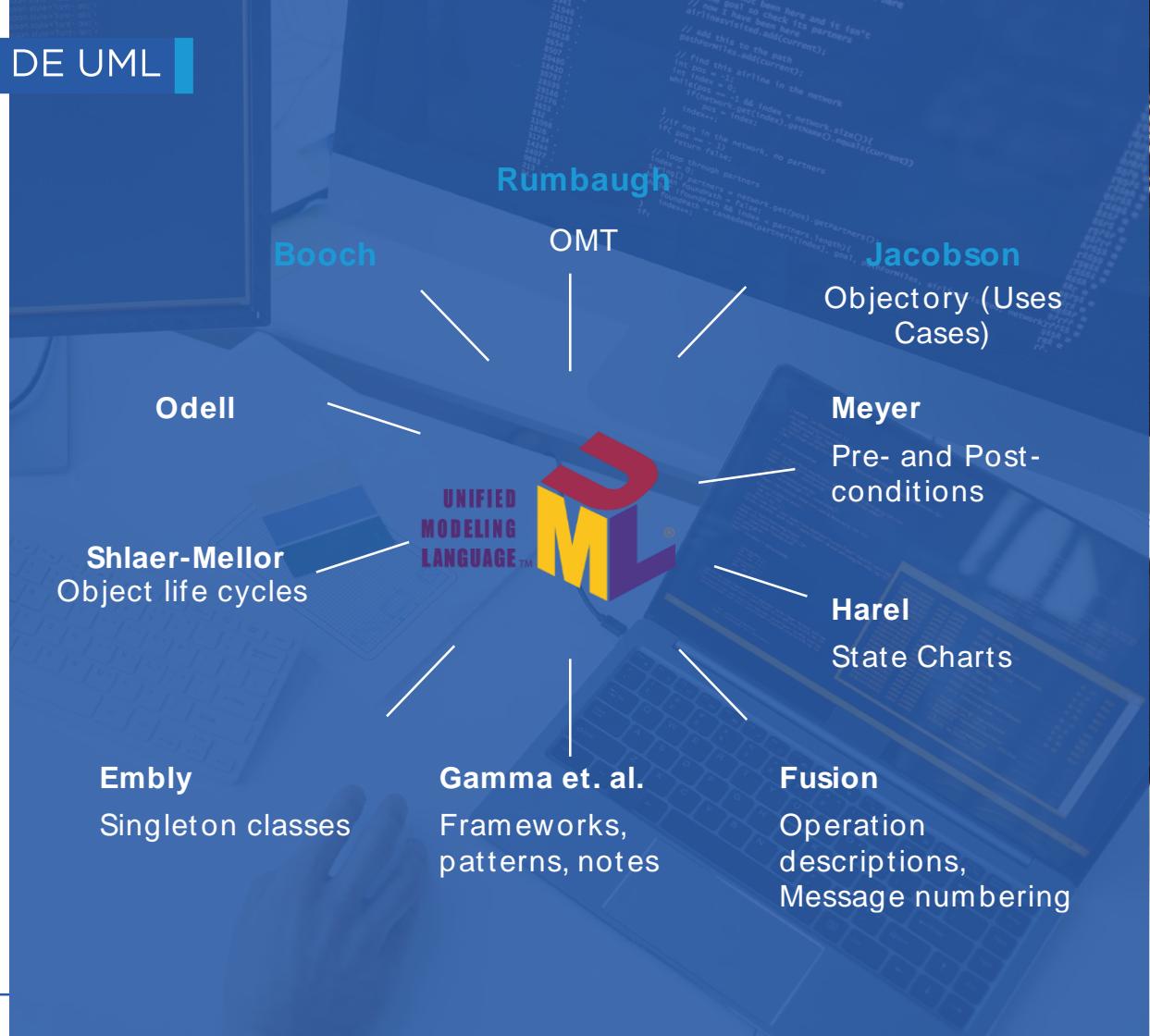
ELEMENTOS DEL METAMODELO DE UML

El UML se ha convertido en el estándar de facto de la industria. Esto se debe, principalmente, a que ha sido concebido por los autores de los tres métodos más usados de orientación a objetos.

La especificación de UML está definida y ha sido publicada por el OMG.

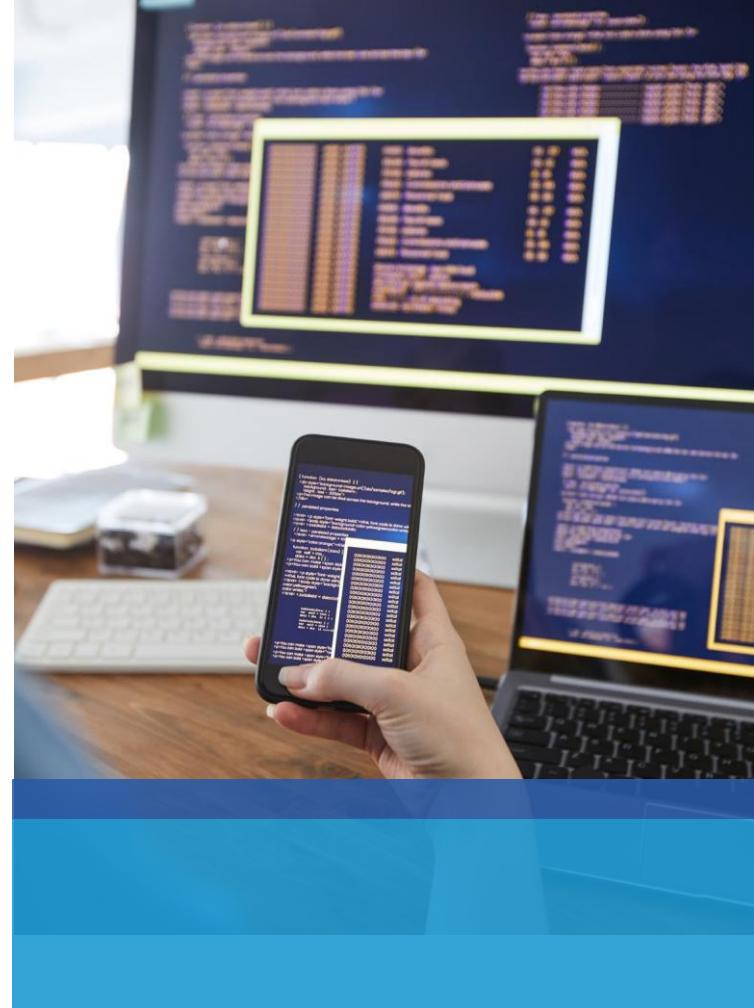
Las raíces técnicas de UML son:

- OMT, *Object Modeling Technique* (Rumbaugh).
- Método-Booch (G. Booch).
- OOSE, *Object-Oriented Software Engineering* (I. Jacobson).



UML es independiente de las metodologías de análisis y diseño, así como de los lenguajes de programación que se utilicen en la construcción de los sistemas software. Es importante destacar que se basa en el paradigma de la orientación a objetos.

Por lo tanto, cualquier sistema de software se construye desde la perspectiva de la orientación a objetos.



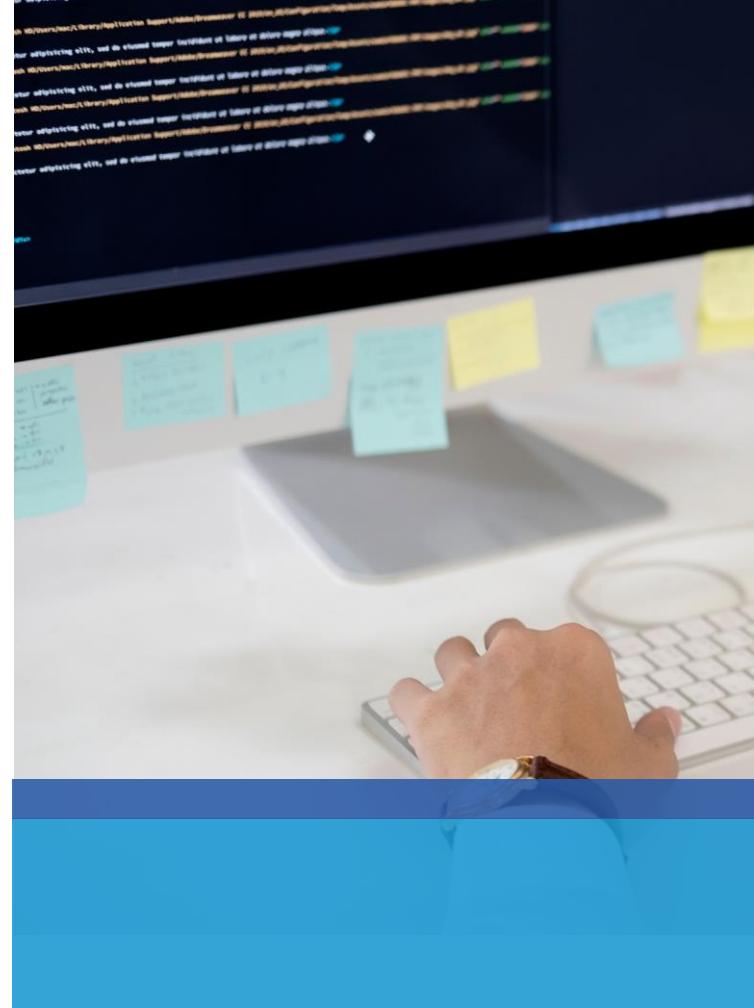


Entre los objetivos de UML se encuentran:

- Visualizar; es decir, expresar de forma gráfica.
- Especificar las características de un sistema.
- Construir a partir de modelos especificados.
- Documentar. En este sentido, los propios elementos gráficos sirven de documentación.

UML permite:

- Modelación de los **componentes estáticos** de una aplicación de software (diagramas de casos de uso, diagramas de clases).
- Modelación de **comportamiento dinámico** de sus principales elementos durante su funcionamiento (entre ellos, diagramas de estados y diagramas de secuencias).



ELEMENTOS DEL METAMODELO DE UML



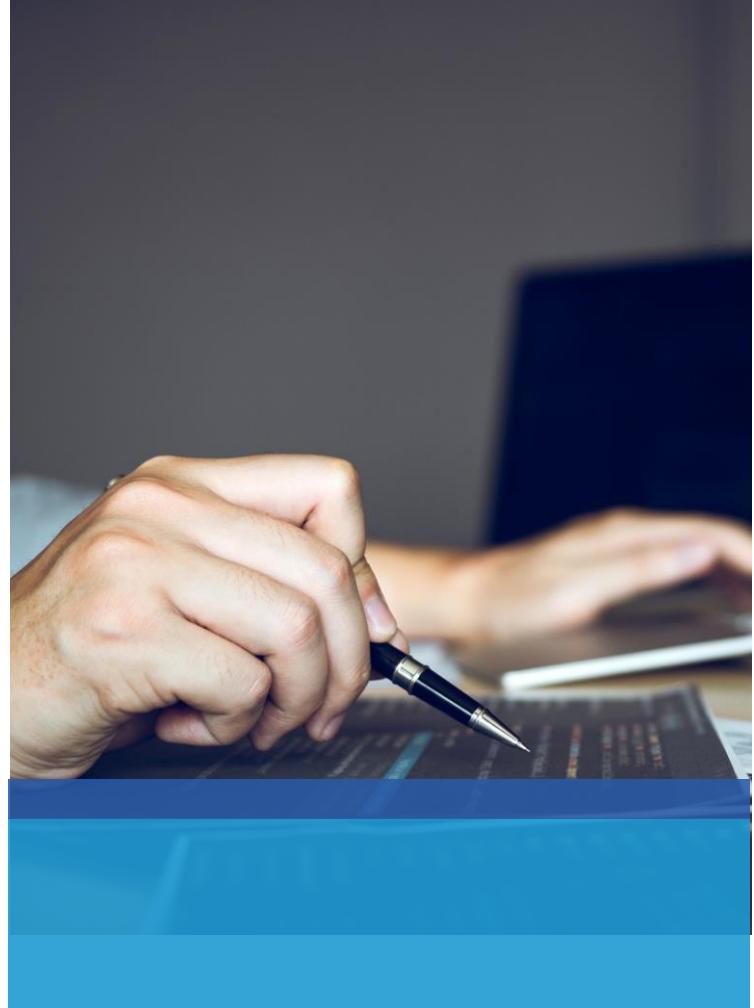
UML incorpora toda una serie de diagramas y notaciones gráficas y textuales destinadas a mostrar el sistema desde las diferentes perspectivas. Estas pueden utilizarse en las diferentes fases del ciclo de desarrollo del software.

Un diagrama de clases UML es una vista del modelo estático del sistema en cuestión. Esto quiere decir que una misma clase puede aparecer en varios diagramas y que podemos crear más de un diagrama.



Ventajas de UML:

- Es estándar.
- Facilita la comunicación.
- Está basado en un metamodelo con una semántica bien definida.
- Se basa en una notación gráfica concisa y fácil de aprender y utilizar.
- Es fácilmente extensible.
- Se puede utilizar para modelar sistemas de software en diversos dominios.





Inconvenientes de UML:

- No es una metodología. Además de UML, hace falta una metodología OO.
- No cubre todas las necesidades de especificación de un proyecto de software.
- No define los documentos textuales o el diseño de interfaces de usuario.
- Faltan ejemplos elaborados en la documentación.
- Puede resultar complejo alcanzar un conocimiento completo del lenguaje.

ELEMENTOS DEL METAMODELO DE UML

El metamodelo de UML está dividido en tres paquetes. Estos contienen todos los elementos de modelado, así como los tipos de diagramas que conforman el lenguaje unificado.

Elementos de comportamiento

Manejo de modelos

Fundamentación

Paquete de Elementos de Comportamiento (Behavioral Elements Package)

Define los elementos necesarios para representar la dinámica de un sistema, como son los casos de utilización (*use cases*), los diagramas de interacción, los diagramas de actividades y los diagramas de estado.

Elementos de comportamiento

Paquete de Manejo de Modelos (Model Management Package)

Especifica cómo organizar los elementos de UML en modelos, paquetes y sistemas.

```
// now go to its partners  
if(!partnerVisited.add(current))  
    // add this to the path  
    partnerPath.add(current);  
  
// Find this airline in the network  
int pos = -1;  
while(pos < 0;  
    pos = -1; doIndex < network.size();  
    while(partnerIndex < partners.length);  
        if(partnerIndex == index){  
            if(partner == current){  
                partnerIndex++;  
                pos = index;  
                break;  
            }  
        }  
    }  
    if(pos < 0)  
        return false;  
    index++;  
}  
  
// loop through partners  
for(int i = 0; i < partners.length; i++)  
    if(partners[i] == current){  
        partners[i].setVisited(true);  
        break;  
    }  
}  
return true;
```

Manejo de
modelos

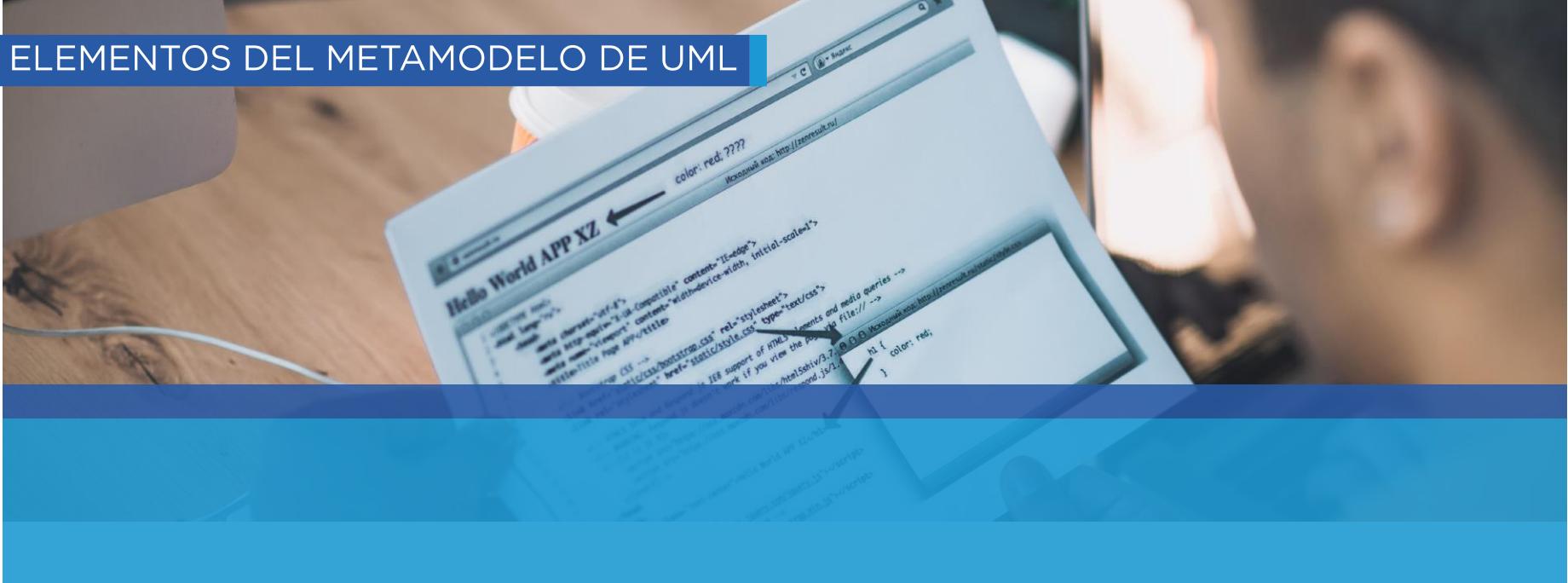
Paquete de Fundamentación (Foundation Package)

A este paquete pertenecen, por ejemplo, los elementos que componen los diagramas de clases, mismos que son utilizados para representar la estructura de un sistema; es decir, la infraestructura de UML.

A blue-tinted background image showing a person's hands on a keyboard and a laptop screen displaying code. A semi-transparent blue rectangle is overlaid on the image, containing the text "Fundamentación".

Fundamentación

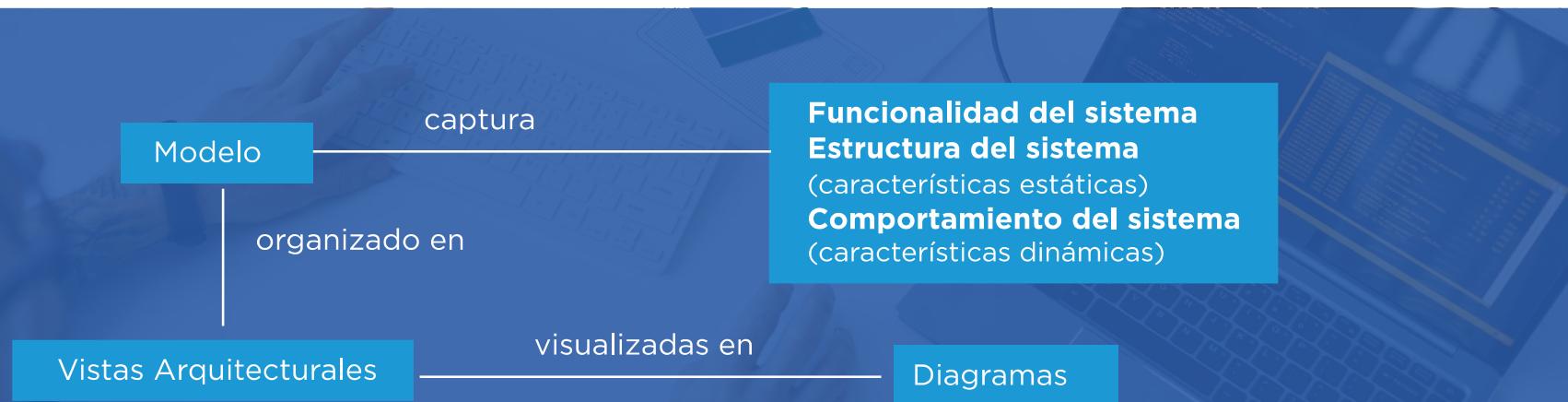
ELEMENTOS DEL METAMODELO DE UML



El modelo UML de un sistema consiste en un conjunto de elementos de modelado que definen la estructura, el comportamiento y la funcionalidad del sistema. Estos conceptos se agrupan en una base de datos única.

La presentación de esos conceptos se realiza a través de múltiples diagramas. Esto con el fin de introducirlos, editarlos y hacerlos comprensibles.

Los diagramas pueden agruparse en vistas; cada una enfocada a un aspecto particular del sistema.



La gestión de un modelo UML requiere una herramienta específica que mantenga la consistencia del modelo.

Aplicación Eficaz de UML => Conocer y comprender su metamodelo

Metamodelo = Modelo conceptual del lenguaje

- ¿Qué elementos nos ofrece UML para modelar un sistema?
- ¿Qué representa cada uno?
- ¿Para qué se usa?



VIDEO

Te invitamos a ver el siguiente video:



ELEMENTOS DEL METAMODELO DE UML

El metamodelo de UML incluye tres tipos de **elementos principales**:

1. Bloques de construcción
2. Reglas
3. Mecanismos

Metamodelo UML



Bloques de
construcción

Reglas

Mecanismos
comunes

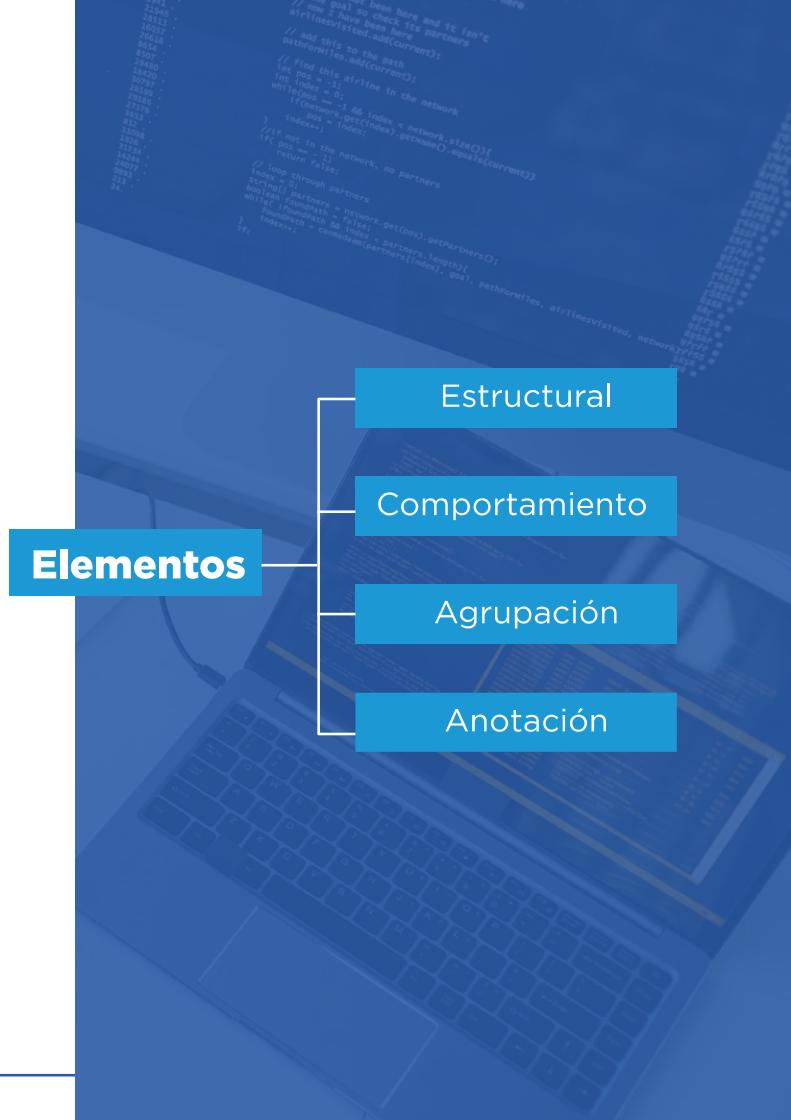
1. Bloques de construcción

Componentes:

- Elementos
- Relaciones
- Diagramas

Elementos. Son los bloques básicos de construcción de un sistema orientado a objetos. Se trata de abstracciones que constituyen los ciudadanos de primera clase en un modelo.

Se utilizan para construir modelos bien formados.



Relaciones. Una relación es una conexión entre elementos estructurales.

Existen cuatro tipos de relaciones y dos tipos especiales de asociación.

Relación

Dependencia

Asociación

Generalización

Realización

Agregación

Composición

ELEMENTOS DEL METAMODELO DE UML

Diagramas. Sirven para visualizar un sistema desde diferentes perspectivas. Se trata de una vista resumida de los elementos que constituyen el sistema.

Un diagrama puede contener cualquier combinación de elementos y relaciones. Surgen así, los tipos de diagramas de UML 2.

Diagramas

Estructurales (Estática)

Clases

Objetos

Componentes

Despliegue

Paquetes

Estructura Compuesta

Casos de uso

Estados

Actividades

Interacción

Comportamiento (Dinámica)

Secuencia

Comunicación

Tiempos

Revisión de Interacción



2. Reglas

Estas se encargan de dictar cómo pueden combinarse los bloques.

3. Mecanismos

Estos son comunes y se aplican a lo largo del lenguaje. Se dividen en:

- Especificaciones
- Adornos
- Divisiones comunes
- Mecanismos de extensibilidad

Los mecanismos dan cohesión y simplifican el modelo. Se aplican a todos los bloques de construcción del modelo.



VIDEO

Te invitamos a ver el siguiente video:



VISTAS ARQUITECTURALES

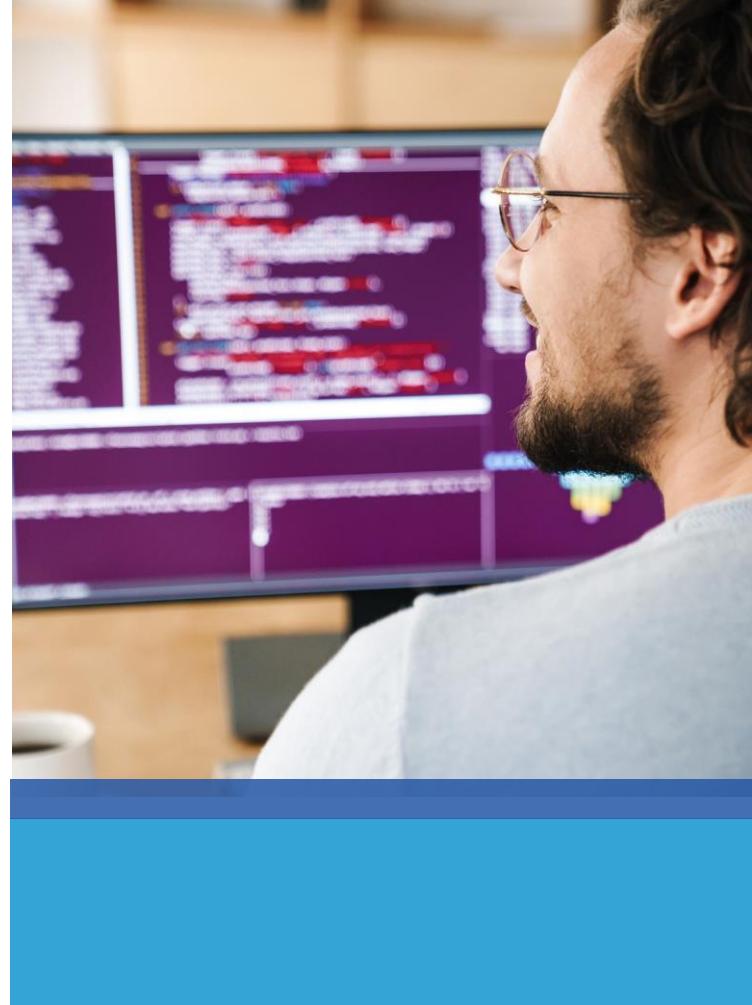
UML cubre las diferentes vistas de la arquitectura de un sistema mientras evoluciona a través del ciclo de vida del desarrollo de software.

Vistas de software (estáticas, dinámicas, etc.).



Una vista es, simplemente, un **subconjunto de UML** que modela construcciones que representan un aspecto de un sistema.

La división en diversas vistas es algo arbitraria, pero intuitiva. Una o dos clases de diagramas proporcionan una notación visual para los conceptos de cada vista.



VISTAS ARQUITECTURALES



Diferentes usuarios miran el sistema de formas diferentes, en momentos distintos.

Por lo anterior, la arquitectura del sistema es clave para poder manejar estos puntos de vista diferentes:

- Se organiza mejor a través de vistas arquitecturales interrelacionadas.
- Las proyecciones del modelo del sistema están centradas en un aspecto particular.

En el **nivel superior**, las vistas se pueden dividir en 3 áreas:

- Clasificación estructural,
- De comportamiento dinámico y
- Gestión del modelo.

Área	Vista	Diagramas
Estática	Vista Estática	D. de Clase
	Vista de casos de uso	D. de Casos de uso
	Vista de implementación	D. de Componentes
	Vista de despliegue	D. de Despliegue
Dinámica	Vista de máquinas de estados	D. de Estados
	Vista de actividad	D. de Actividad
	Vista de interacción	D. de Secuencia D. de Colaboración
Gestión de modelo	Vista de gestión de modelo	D. de Clases
Estensión de UML	Todas	Todos

Clasificación estructural. Describe los elementos del sistema y sus relaciones con otros elementos.

Los clasificadores incluyen clases, casos de uso, componentes y nodos, así como elementos que proporcionan la base sobre la cual se construye el comportamiento dinámico.

La clasificación de las vistas incluye la vista estática, la vista de casos de uso y la vista de implementación.



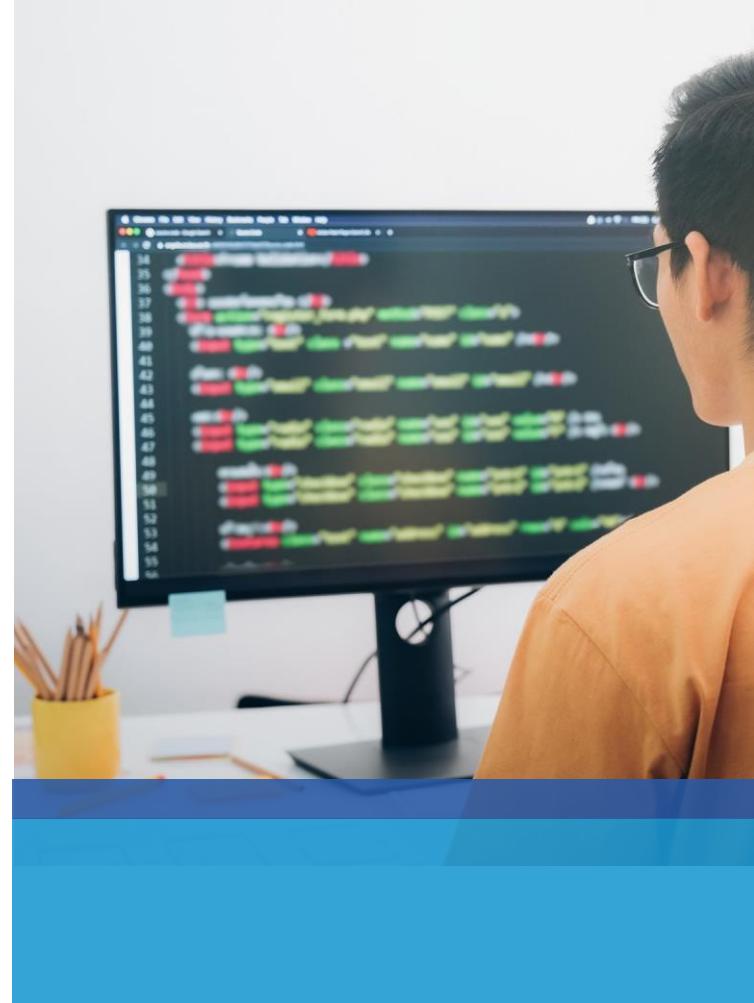
Comportamiento dinámico. Describe el comportamiento de un sistema en el tiempo.

El comportamiento se puede describir como una serie de cambios a las fotos del sistema dibujadas a partir de la visión estática. Las vistas de comportamiento dinámico incluyen la vista de la máquina de estados, la vista de actividad y la vista de interacción.

Gestión del modelo. Describe la organización de los propios modelos en unidades jerárquicas.

El **paquete** es la unidad genérica de organización para los modelos.

Los paquetes especiales incluyen a los modelos y a los subsistemas. La vista de gestión del modelo cruza las otras vistas y las organiza para el trabajo de desarrollo y el control de configuración.





Vista estática

Modela conceptos del dominio de la aplicación, así como conceptos internos inventados como parte de la implementación de una aplicación. Esta vista es estática porque no describe el comportamiento dependiente del tiempo del sistema.

Los principales componentes de la vista estática son las clases y sus relaciones: asociación, generalización y varios tipos de dependencia, como la realización y el uso.

VISTAS ARQUITECTURALES

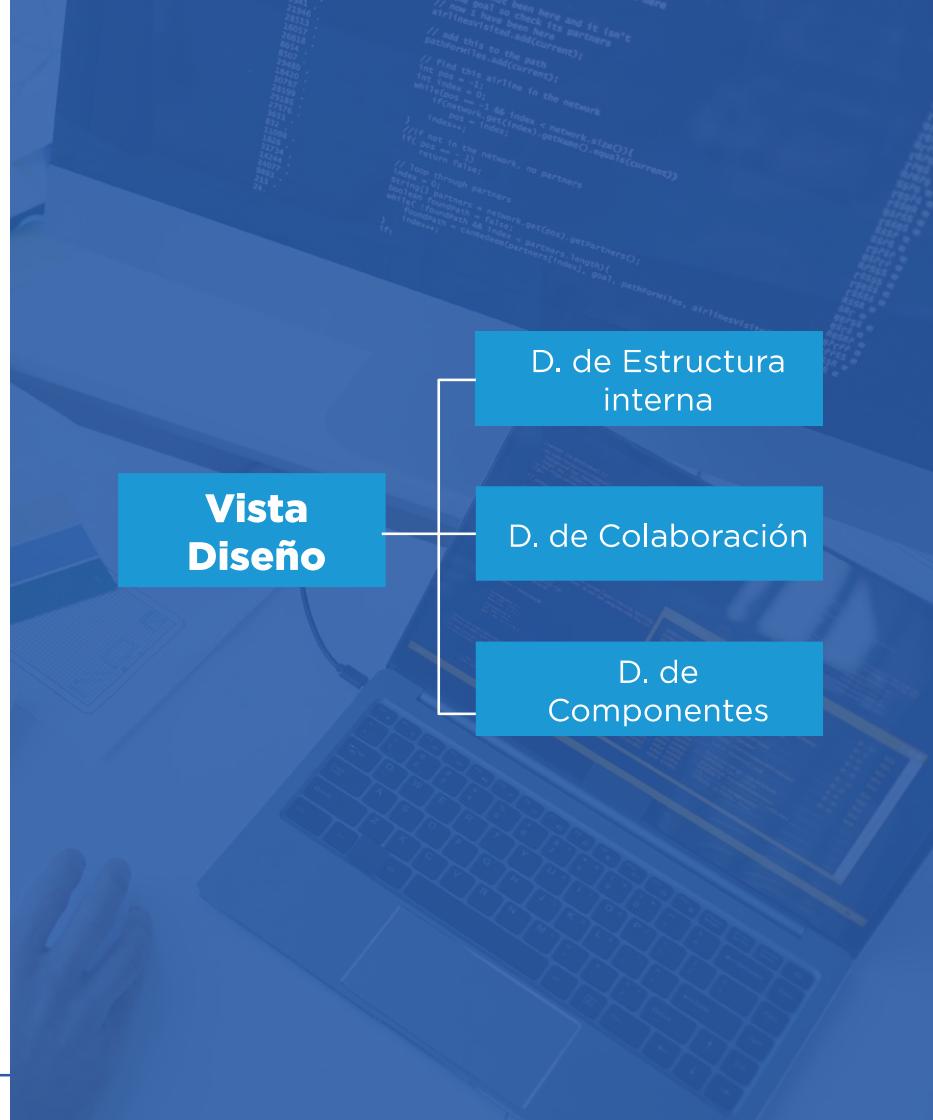
Diagrama de clases



Vistas de Diseño

Las vistas previas modelan los conceptos de la aplicación desde un punto de vista lógico.

Las vistas de diseño modelan la estructura de diseño de la propia aplicación, como su expansión en clasificadores estructurados, las colaboraciones que proporcionan funcionalidad y su ensamblado a partir de componentes con interfaces bien definidas.



Vistas de Casos de Uso

Modela la funcionalidad de un sistema tal como lo perciben los agentes externos, denominados actores. Estos últimos interactúan con el sistema desde un punto de vista particular.

Su propósito es enumerar los actores y casos de uso. Además, muestra qué actores participan en cada caso de uso.



Vistas de Máquina de Estados

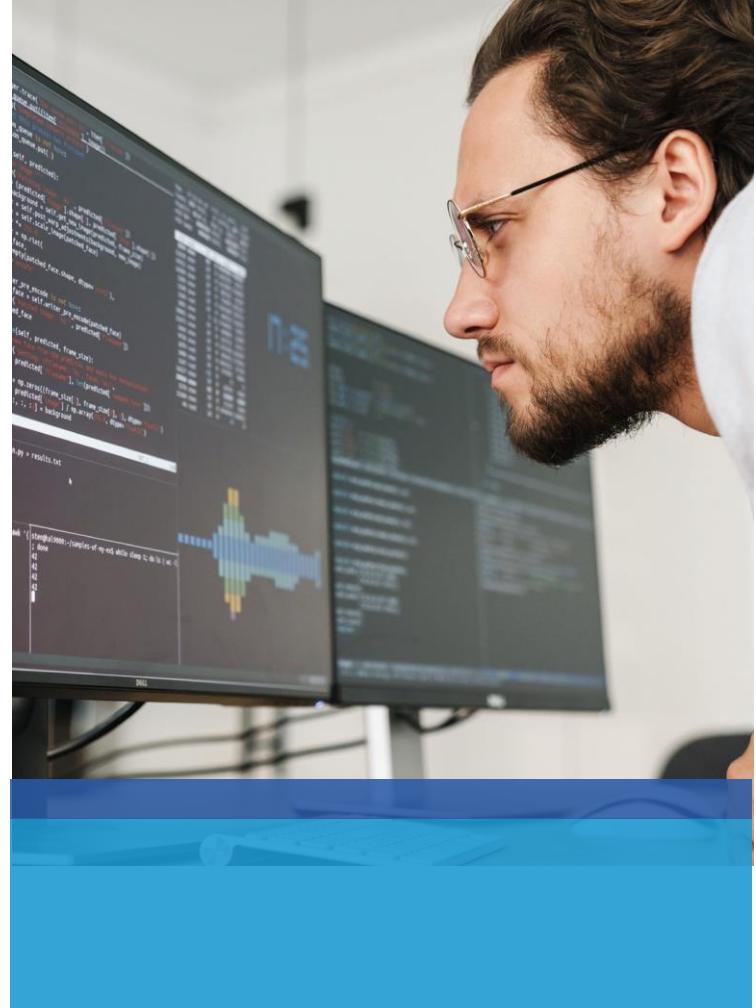
Modela las posibles historias de vida de un objeto de una clase. Una máquina de estados contiene estados conectados por transiciones. Cada estado modela un periodo de tiempo. Cuando ocurre un evento, se puede desencadenar una transición. Cuando se dispara una transición, se ejecuta un efecto.



Vista de Actividad

Una actividad muestra el flujo de control entre las actividades computacionales involucradas en la realización de un cálculo o un flujo de trabajo.

Una acción es un paso computacional primitivo. Por su parte, un nodo de actividad es un grupo de acciones o subactividades. Una actividad describe, tanto el cómputo secuencial como el concurrente. Las actividades se muestran en los diagramas de actividad.



Vista de Interacción

Describe el intercambio de secuencias de mensajes entre las partes de un sistema. Proporciona una visión integral del comportamiento de un sistema. Además, muestra el flujo de control a través de varios objetos.

Se muestra mediante dos diagramas:

- De secuencia
- De comunicación

Vista de Despliegue

Esta vista permite valorar las consecuencias de la distribución y la asignación de recursos.

Un diagrama de despliegue representa el despliegue de artefactos (unidad de implementación física) de tiempo de ejecución sobre nodos (recurso de tiempo de ejecución, ejemplo: una computadora).



Vista de Gestión del Modelo

Modela la organización del modelo en sí mismo. Un modelo abarca un conjunto de paquetes que contiene los elementos del modelo, tales como las clases, máquinas de estados y casos de uso.

Los paquetes pueden contener otros paquetes: por lo tanto, un modelo comienza con un paquete raíz que indirectamente alberga todos los contenidos del modelo.

FORO. Entorno de trabajo

Participa en el foro enviando imágenes que demuestren que ya tienes acceso a las siguientes herramientas en su versión de prueba:

- ArgoUML
- GitMind
- UMLet

Presiona el botón para participar en el foro.



LECTURAS PARA REFORZAR LA UNIDAD

- Dennis, A., Wixom, B. H., & Tegarden, D. (2015). *Systems Analysis and Design: An Object-Oriented Approach with UML*. Wiley.

<https://www.pdfdrive.com/systems-analysis-and-design-an-object-oriented-approach-with-uml-d185445863.html>

- Rumpe, B. (2016). *Modeling with UML: Language, Concepts, Methods* (2016 ed.). Springer.

<https://doi.org/10.1007/978-3-319-33933-7>

CONCLUSIÓN



Como pudimos ver a lo largo de esta unidad, UML es un lenguaje de modelado orientado a objetos que permite representar gráficamente los elementos estáticos y dinámicos de una aplicación de software. Además, facilita la construcción de una serie de modelos, a través de diagramas de diferentes visiones de un proyecto.

Por su parte, una vista es simplemente un subconjunto de UML. Esta se encarga de modelar construcciones que representan un aspecto de un sistema. En el nivel superior, las vistas pueden ser divididas en las siguientes áreas: clasificación estructural, comportamiento dinámico, diseño físico y gestión del modelo.

¡FELICIDADES!



Acabas de concluir la primera unidad de tu curso *Lenguaje Unificado de Modelado*. Te invitamos a finalizar este esfuerzo realizando el examen parcial correspondiente. Para ello, debes regresar a la pantalla principal y dar clic en *Presentar examen*.



LENGUAJE UNIFICADO DE MODELADO

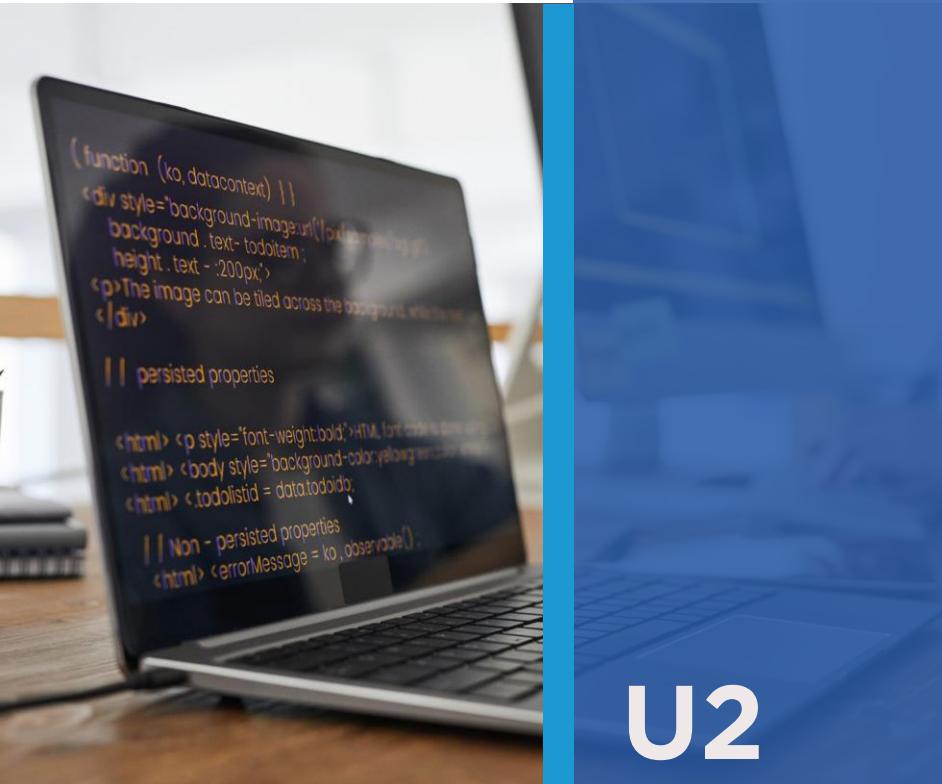


UNIDAD 2

DIAGRAMAS DE COMPORTAMIENTO E INTERACCIÓN

TEMARIO

U2



2.1



Diagramas de Comportamiento

2.2



Diagramas de Interacción

A photograph showing a person's hands typing on a silver laptop keyboard. The person is wearing a red and black plaid long-sleeved shirt. The background is slightly blurred, showing a coffee cup and a computer monitor.

INTRODUCCIÓN

En esta segunda unidad reconoceremos los tipos de diagramas de comportamiento e interacción que componen el estándar UML. Además, conoceremos sus elementos básicos, procesos de construcción y funciones principales para el desarrollo de un sistema.

COMPETENCIAS A DESARROLLAR



1.

El alumno será capaz de identificar y diseñar los diagramas de comportamiento para el modelado y diseño de sistemas de acuerdo al Lenguaje Unificado de Modelado.

2.

El alumno será capaz de identificar y diseñar los diagramas de interacción para el modelado y diseño de sistemas de acuerdo al Lenguaje Unificado de Modelado.

DIAGRAMAS DE COMPORTAMIENTO

A continuación se muestra la jerarquía de diagramas en UML 2:



DIAGRAMAS DE COMPORTAMIENTO

Los diagramas de comportamiento de UML 2 sirven para visualizar, especificar, construir y documentar los aspectos dinámicos de un sistema.

Se organizan con base en las formas en que se puede modelar la dinámica de un sistema.

Tipo de diagrama	Forma de modelar la dinámica
Casos de uso	Especificación del comportamiento externo.
Secuencia	Ordenamiento temporal de los mensajes.
Comunicación	Organización estructural de los objetos; envían y reciben mensajes.
Estado	Estados cambiante de objetos dirigidos por eventos.
Actividades	Flujo de control de actividades.
Tiempos	Tiempo real de los mensajes y estados.
Revisión de interacciones	Vista general de las interacciones.

DIAGRAMAS DE COMPORTAMIENTO

Diagrama de Casos de Uso:

- Este tipo de diagrama implica una técnica para capturar información respecto de los servicios que un sistema proporciona a su entorno (captura y especificación de requisitos).
- Muestra un conjunto de casos de uso y actores (tipo especial de clase), así como sus relaciones.
- Cubre la vista de casos de uso estático de un sistema.



DIAGRAMAS DE COMPORTAMIENTO

- Es importante en el modelado y organización del comportamiento de un sistema.
- Modela el comportamiento del sistema desde el punto de vista externo.
- Juega un papel clave en metodologías muy usadas (desarrollo dirigido por casos de uso).

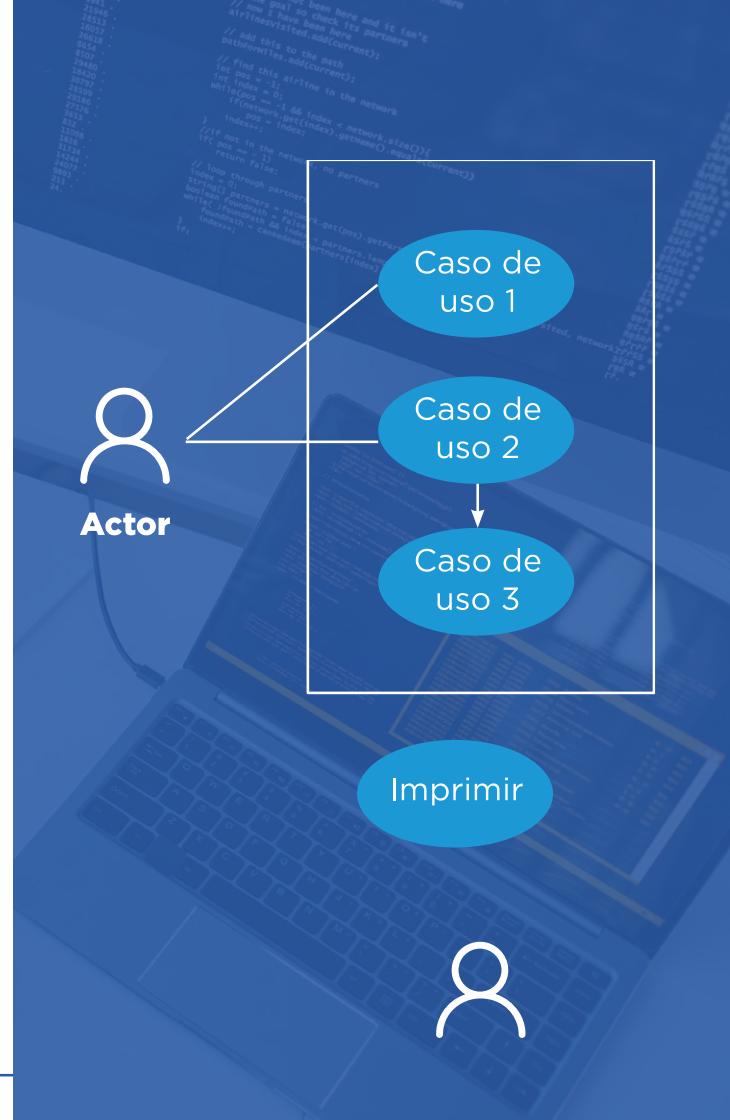


Elementos de Diagrama de Comportamiento:

■ **Sistema.** El rectángulo representa los límites del sistema que contienen los casos de uso. Los actores se ubican fuera de los límites del sistema y se identifican por estructura de una persona.

■ **Casos de Uso.** Se representan con óvalos. La etiqueta en el óvalo indica la función del sistema.

■ **Actores.** Son los usuarios de un sistema.



DIAGRAMAS DE COMPORTAMIENTO

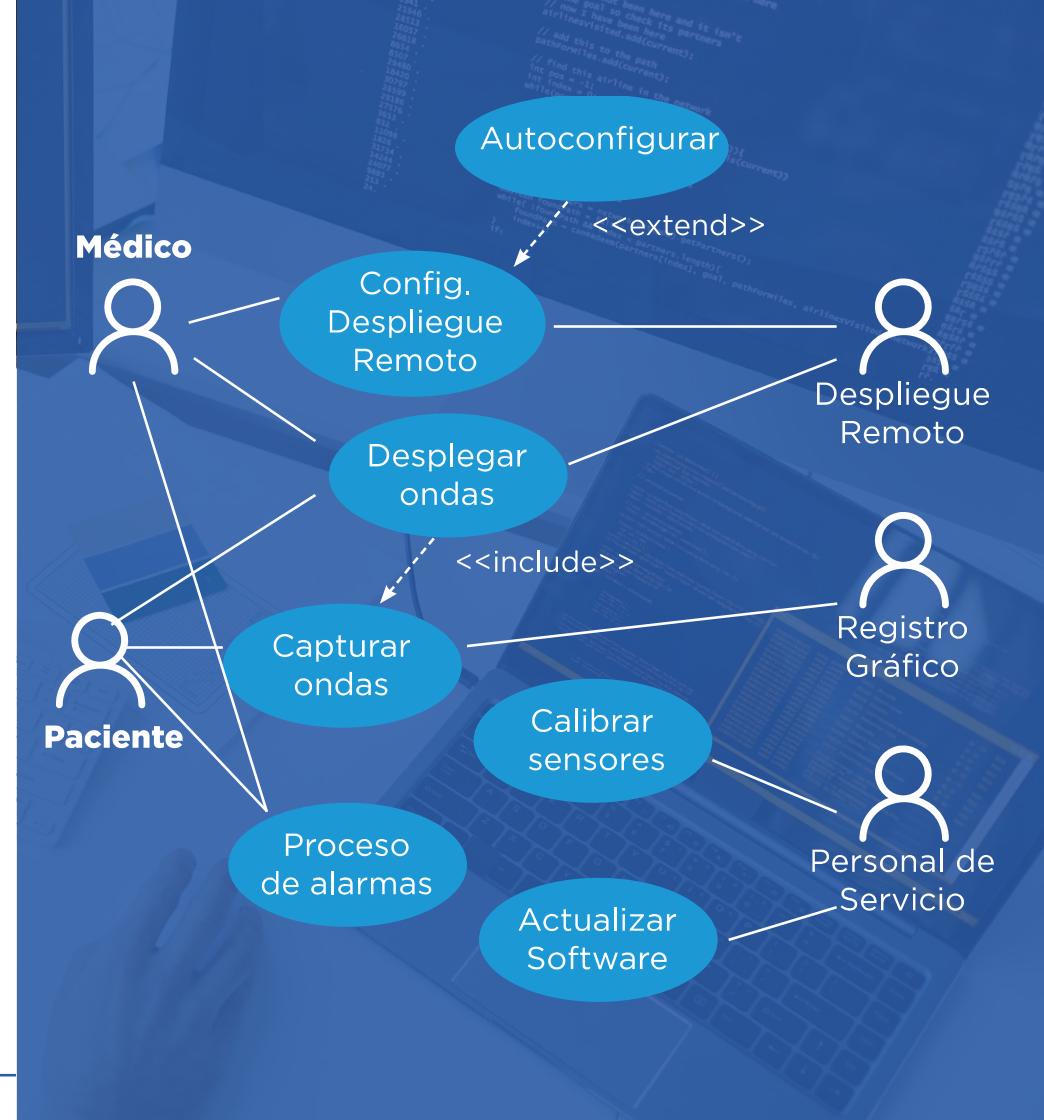
■ **Relaciones.** Las relaciones entre un actor y un caso de uso se dibujan con una línea simple. Para relaciones entre casos de uso, se utilizan flechas etiquetadas como “incluir” o “extender.” Una relación “incluir” indica que un caso de uso es necesitado por otro para poder cumplir una tarea. Una relación “extender” indica opciones alternativas para un cierto caso de uso.



DIAGRAMAS DE COMPORTAMIENTO

Creación de un Diagrama de Caso de Uso:

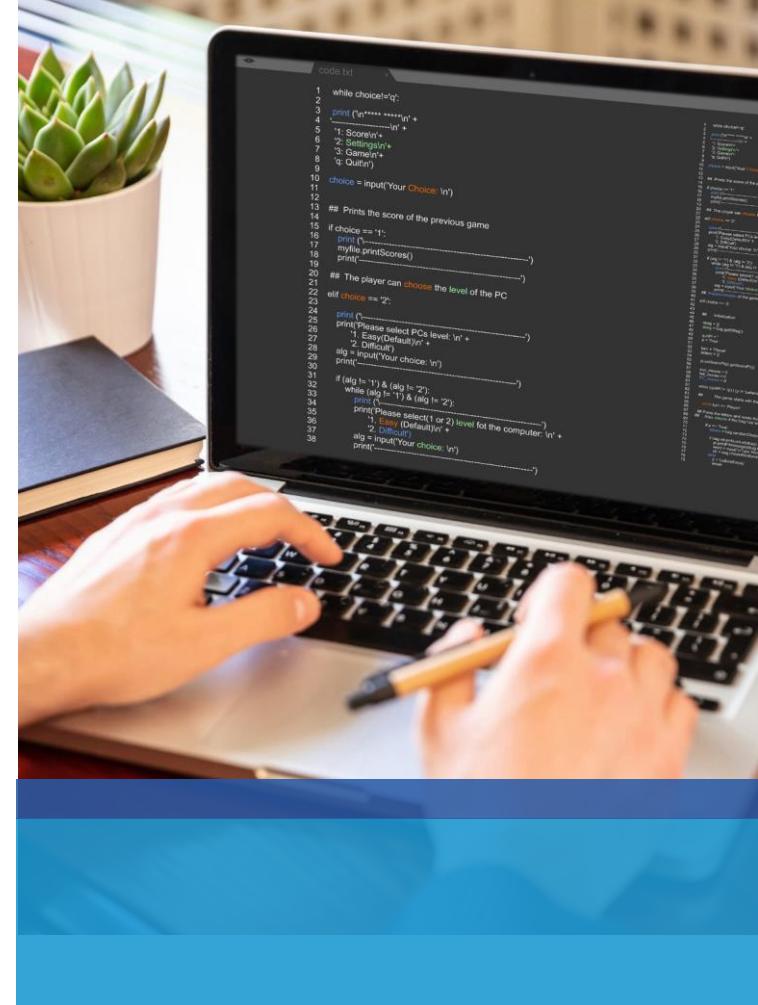
1. Revisa las especificaciones e identifica los actores en el dominio del problema.
2. Identifica los eventos de alto nivel y desarrolla los casos de uso principales que describen dichos eventos, y cómo los inician los actores.



DIAGRAMAS DE COMPORTAMIENTO

- 3.** Examina cuidadosamente los papeles jugados por los actores para identificar casos de uso iniciados por cada uno de ellos.
- 4.** Revisa cada caso de uso principal para determinar sus posibles variaciones. Con este análisis, establece las rutas alternativas.

Debido a que el flujo de eventos normalmente es diferente en cada caso, busca actividades que podrían tener éxito o fallar.



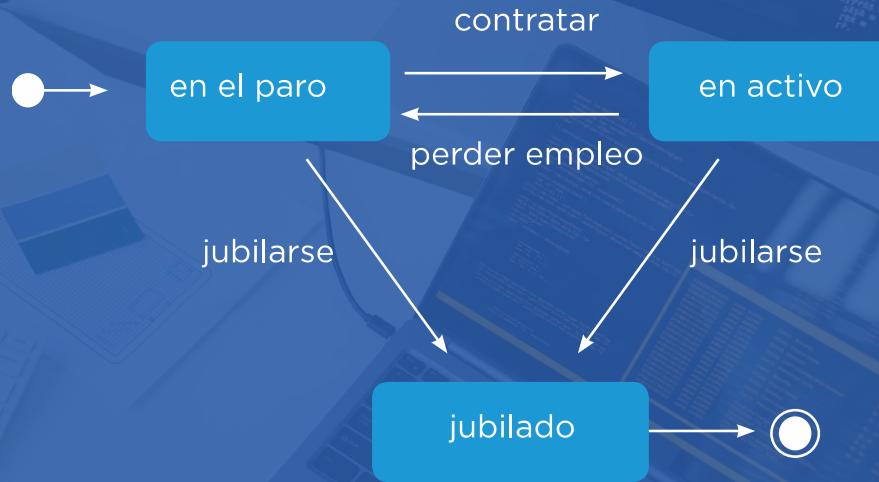
DIAGRAMAS DE COMPORTAMIENTO



DIAGRAMAS DE COMPORTAMIENTO

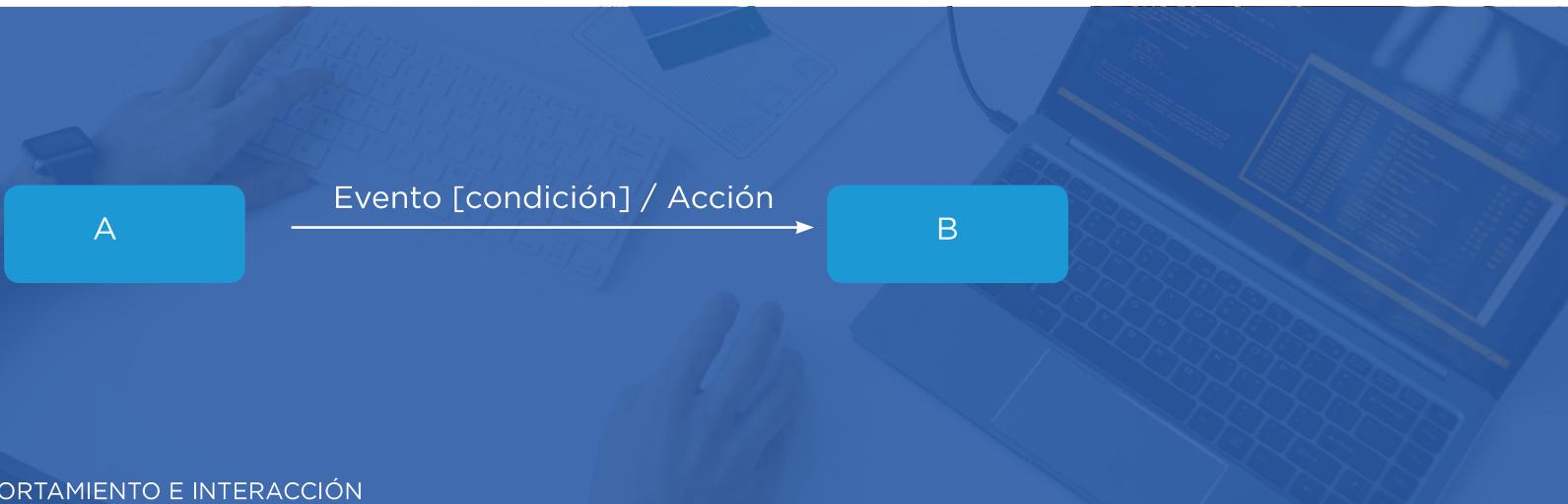
Diagrama de Estados:

- Muestra el conjunto de estados por los cuales pasa un objeto durante su vida en una aplicación, en respuesta a eventos, junto con sus respuestas y acciones.
- También ilustra qué eventos pueden cambiar el estado de los objetos de la clase.



DIAGRAMAS DE COMPORTAMIENTO

- Normalmente contiene máquinas de estados. Estas constan de: estados, transiciones, eventos y acciones.



DIAGRAMAS DE COMPORTAMIENTO

Elementos para la Creación de un Diagrama de Estado:

- **Estado.** Es una condición o situación en la vida de un objeto durante la cual satisface una condición, realiza alguna actividad o espera algún evento.

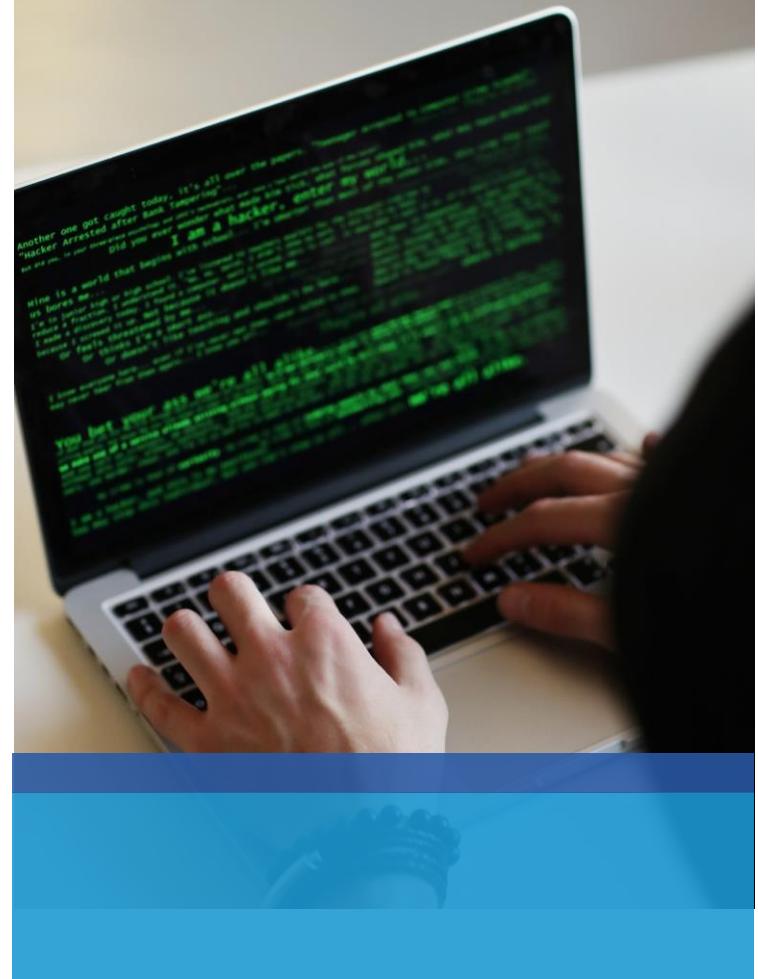


DIAGRAMAS DE COMPORTAMIENTO



- **Evento.** Es la especificación de un acontecimiento significativo que ocupa un lugar en el tiempo y en el espacio. Es la aparición de un estímulo que puede (o no) activar una transición de estado.

- **Transición.** Es una relación entre dos estados que indica que un objeto que esté en el primer estado realizará ciertas acciones, y entrará en el segundo estado cuando ocurra un evento especificado y se satisfagan unas condiciones especificadas.



DIAGRAMAS DE COMPORTAMIENTO



DIAGRAMAS DE COMPORTAMIENTO

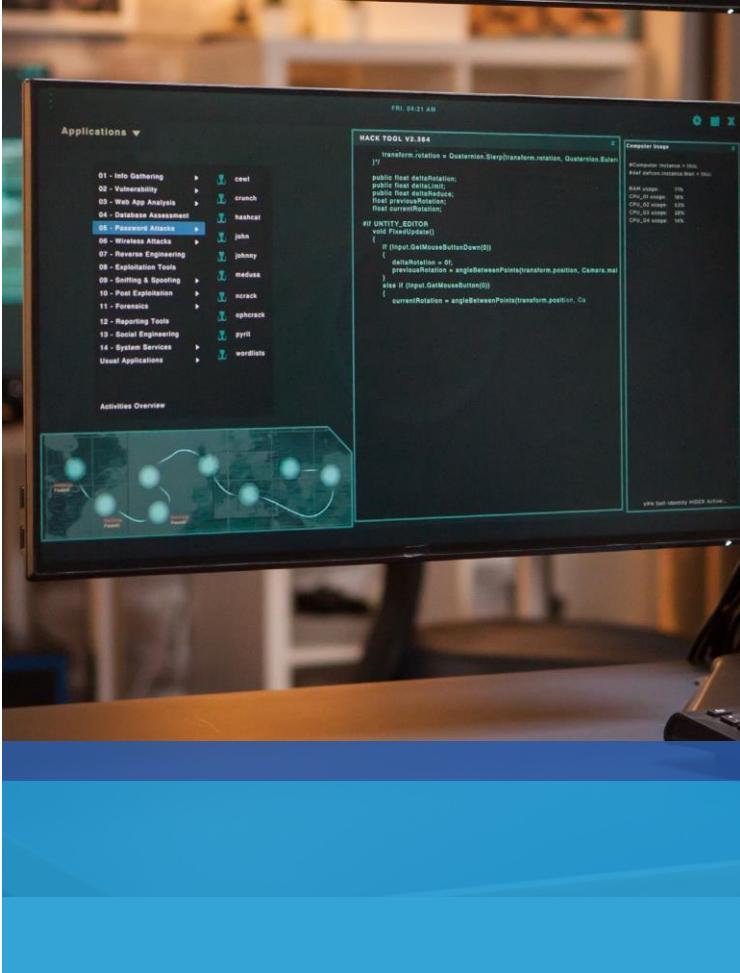


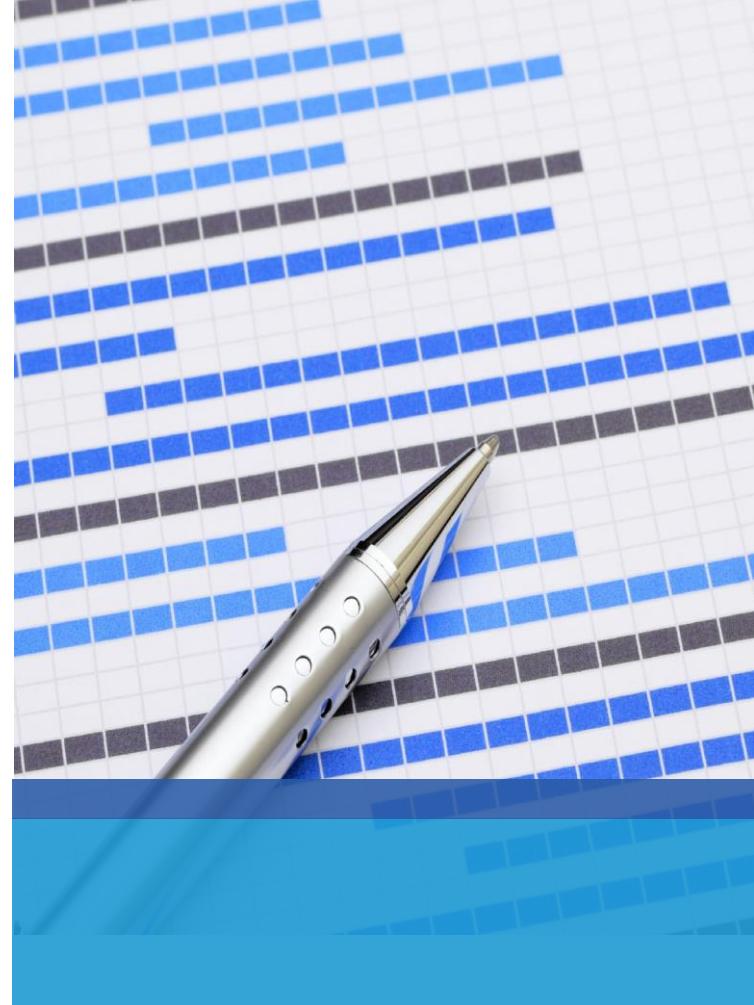
Diagrama de Actividad:

- No es más que un caso especial de un diagrama de estados. En este, todos los estados (o la gran mayoría) son actividades.
- Muestra una serie de acciones o tareas que se ejecutan en cierto orden (y otros elementos adicionales).
- Cubre la vista dinámica de un sistema.

- Es el equivalente en OO a los diagramas de flujo y DFD.

Se emplean para especificar:

- Una operación compleja.
- Un proceso de negocio (*business process*) o flujo de trabajo (*workflow*).
- El proceso de negocio asociado a un caso de uso.





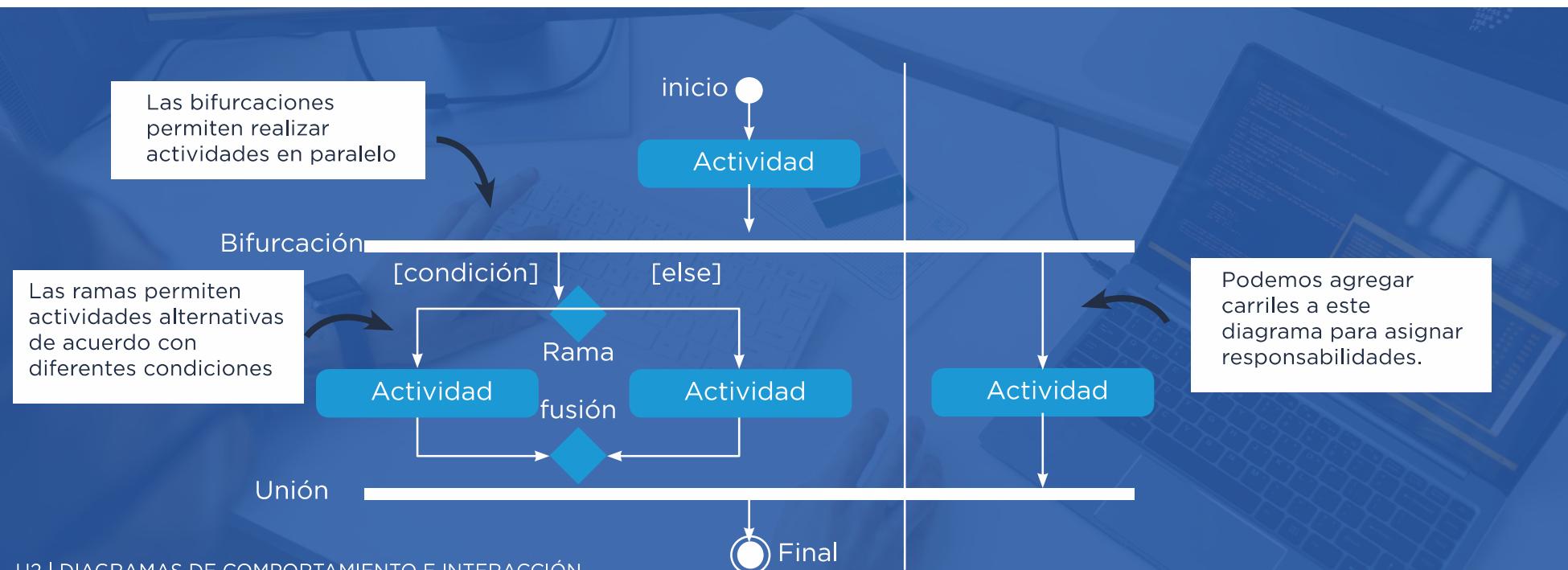
Los Diagramas de Actividades Expresan:

- Conjuntos de actividades
- ¿Qué hacen las actividades?
- ¿En qué orden se ejecutan?
- ¿Dónde ocurren?
- ¿Quién las ejecuta?
- ¿Qué insumos requieren?
- ¿Qué productos generan?

DIAGRAMAS DE COMPORTAMIENTO

Las actividades se enlazan por transiciones automáticas.

Cuando una actividad termina, se desencadena el paso a la siguiente.

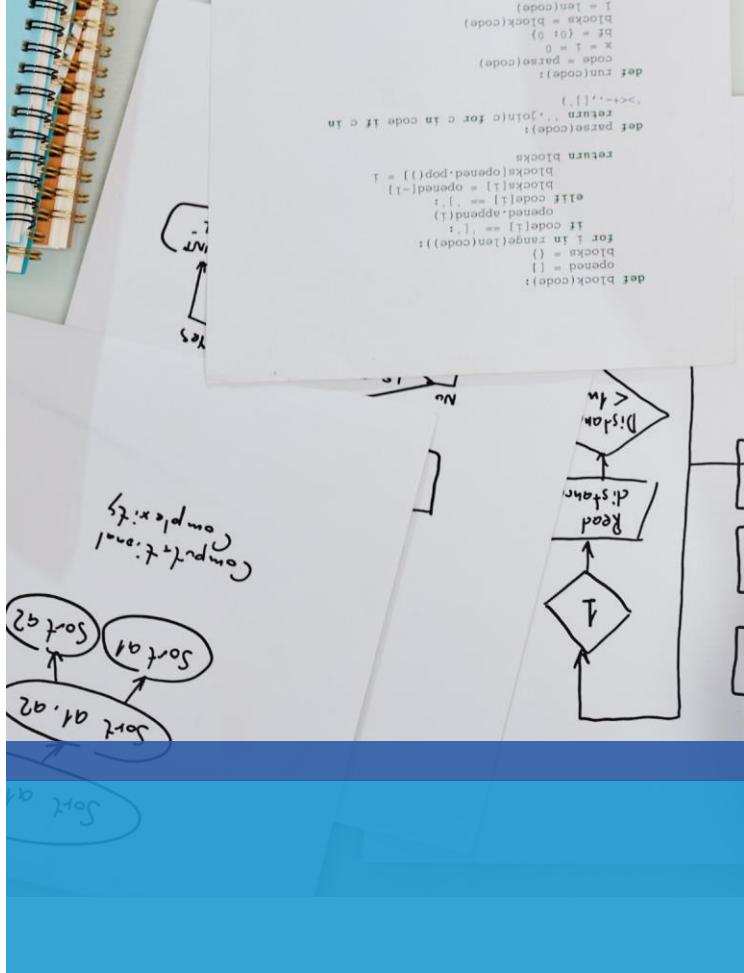


Elementos:

- **Estados de Acción.** Acciones no interrumpidas de los objetos.
- **Flujo de la Acción.** Flechas que ilustran las relaciones entre los estados de acción.
- **Flujo de Objetos.** La creación y modificación de objetos por parte de las actividades.



DIAGRAMAS DE COMPORTAMIENTO

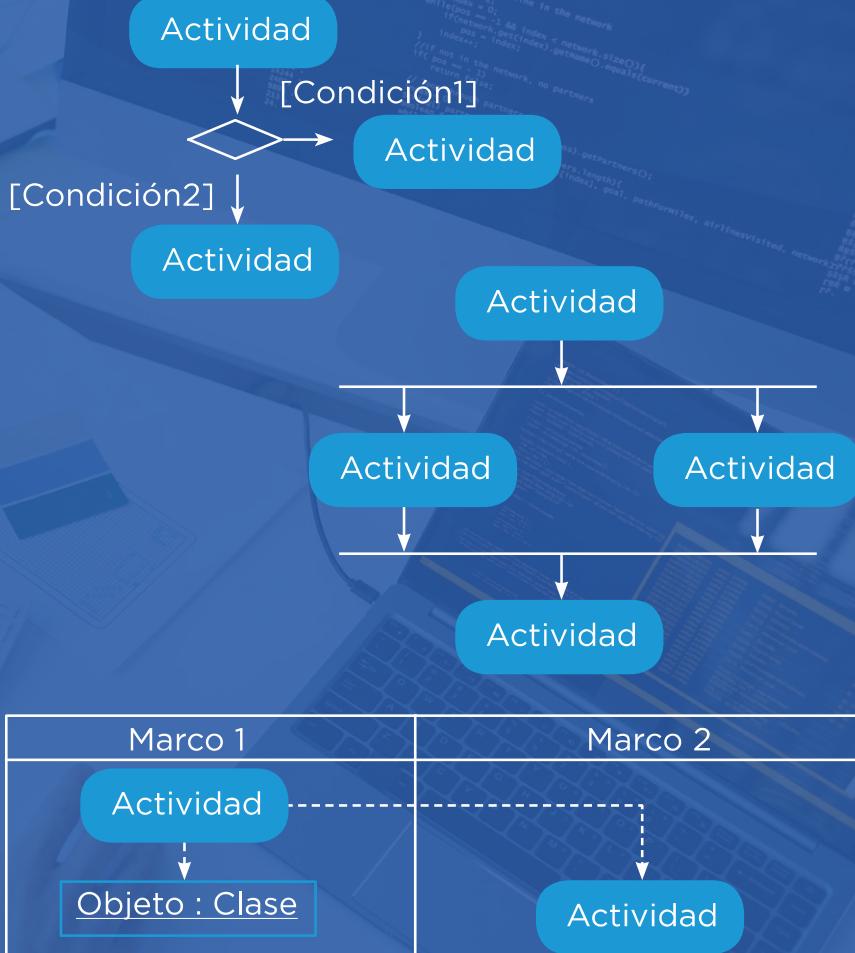


Una flecha desde una acción a un objeto significa que la acción está creando o influyendo sobre dicho objeto.

Una flecha desde un objeto a una acción significa que el estado de acción utiliza dicho objeto.

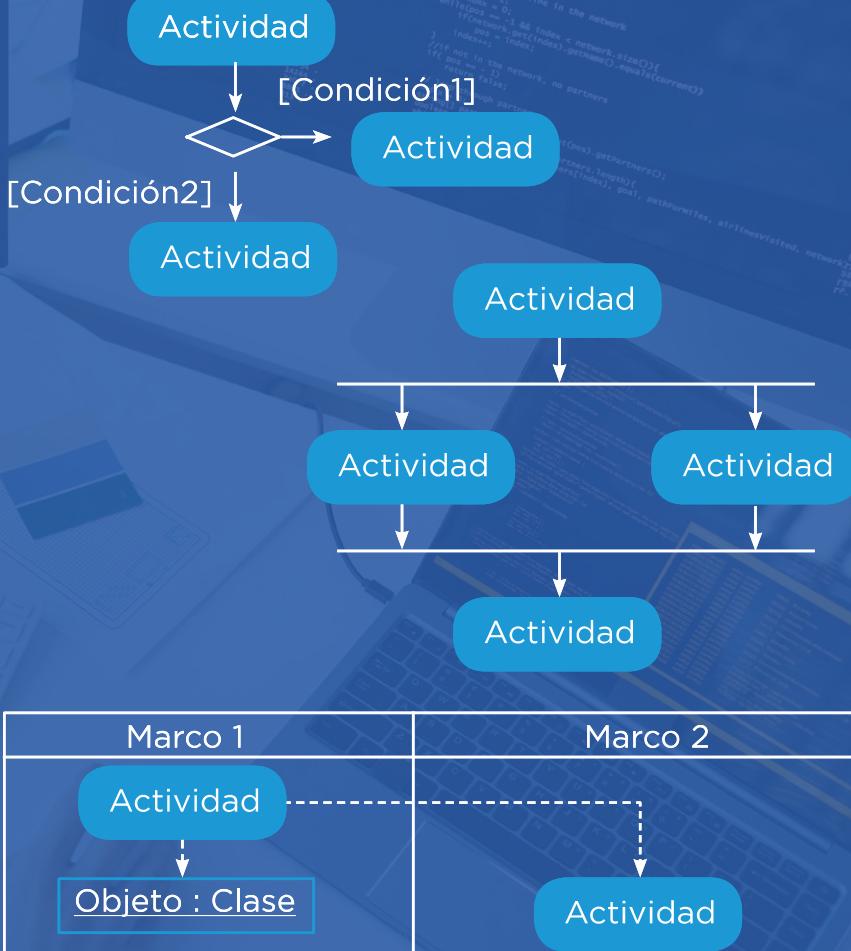
DIAGRAMAS DE COMPORTAMIENTO

- **Estado Inicial.** Se trata del primer estado de un estado de acción.
- **Estado final.** Se trata del último estado de un estado de acción.
- **Ramificación.** Un rombo representa una decisión con caminos alternativos. Las salidas alternativas deben estar etiquetadas con una condición.



DIAGRAMAS DE COMPORTAMIENTO

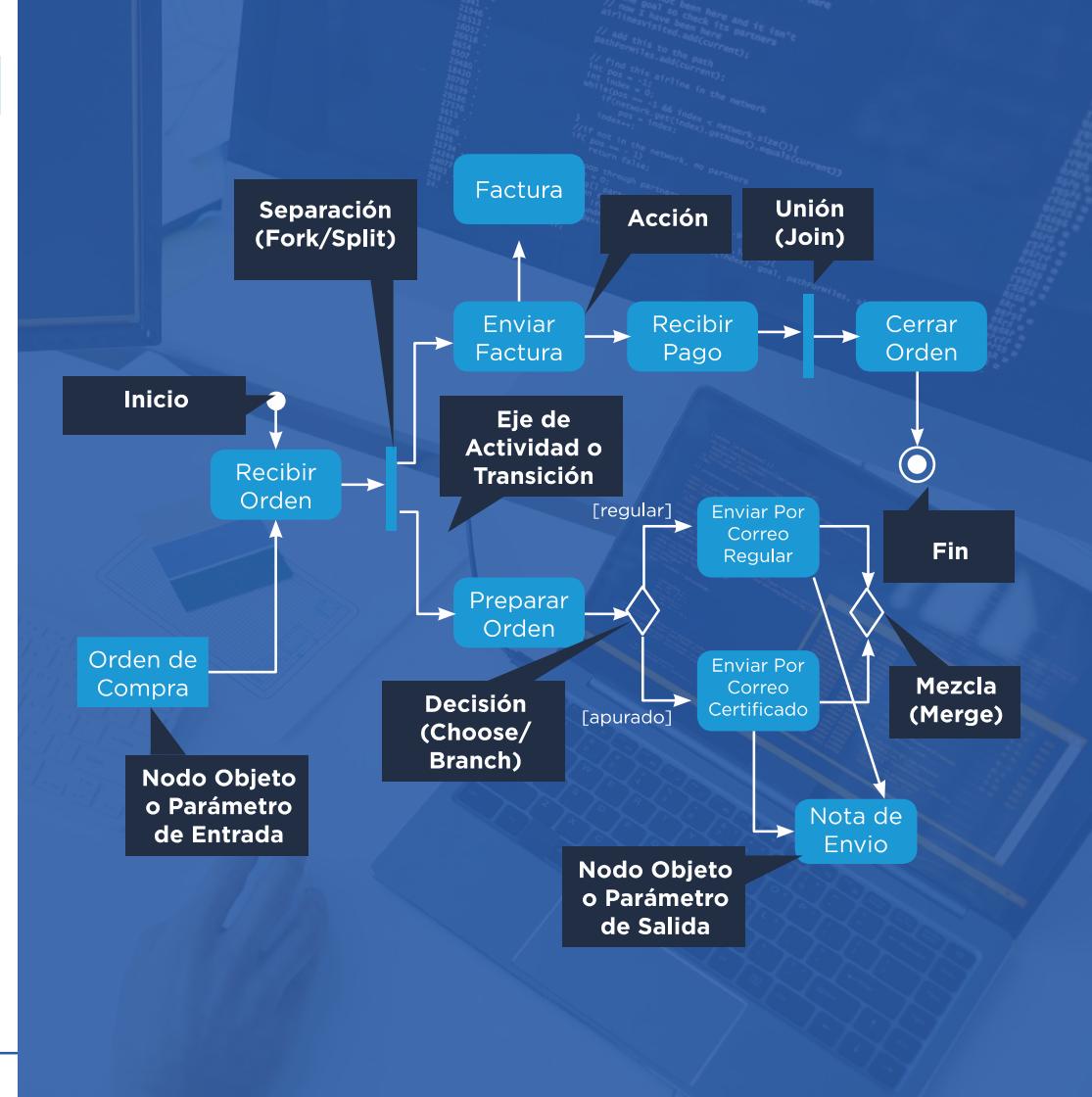
- **Sincronización.** Ayuda a ilustrar la ocurrencia de transiciones paralelas, es decir, las acciones concurrentes.
- **Marcos de Responsabilidad.** Agrupan a las actividades relacionadas en una misma columna.



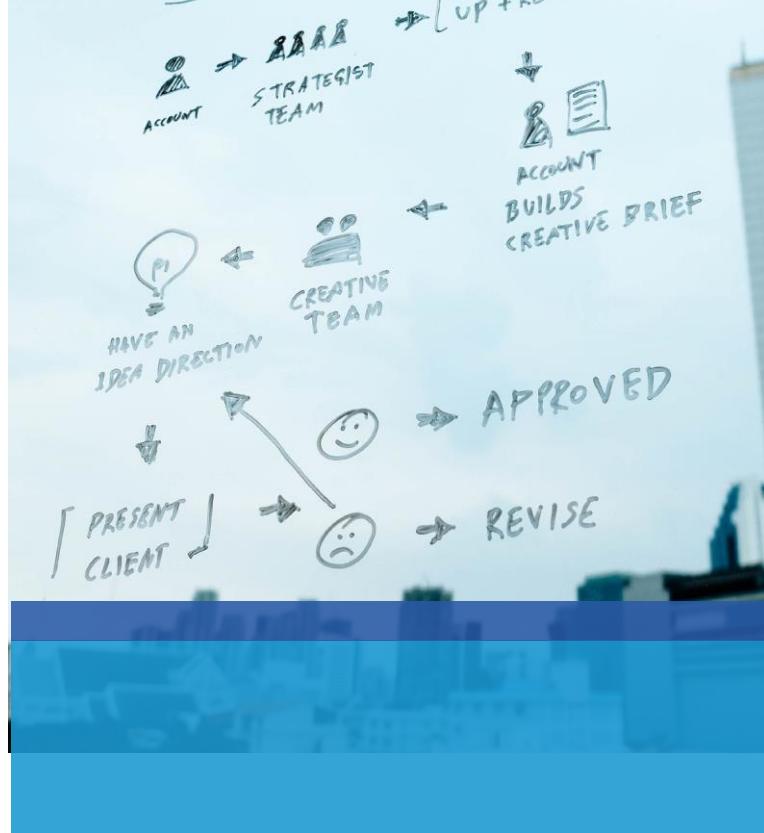
DIAGRAMAS DE COMPORTAMIENTO

Creación de un Diagrama de Actividad:

- Se crea preguntando qué pasa en primer lugar, qué pasa en segundo lugar, y así sucesivamente.
- Se debe determinar si las actividades se realizan en secuencia o en paralelo.
- Si se han creado diagramas de flujo de datos físicos, se podrían usar para determinar la secuencia de actividades.



DIAGRAMAS DE INTERACCIÓN

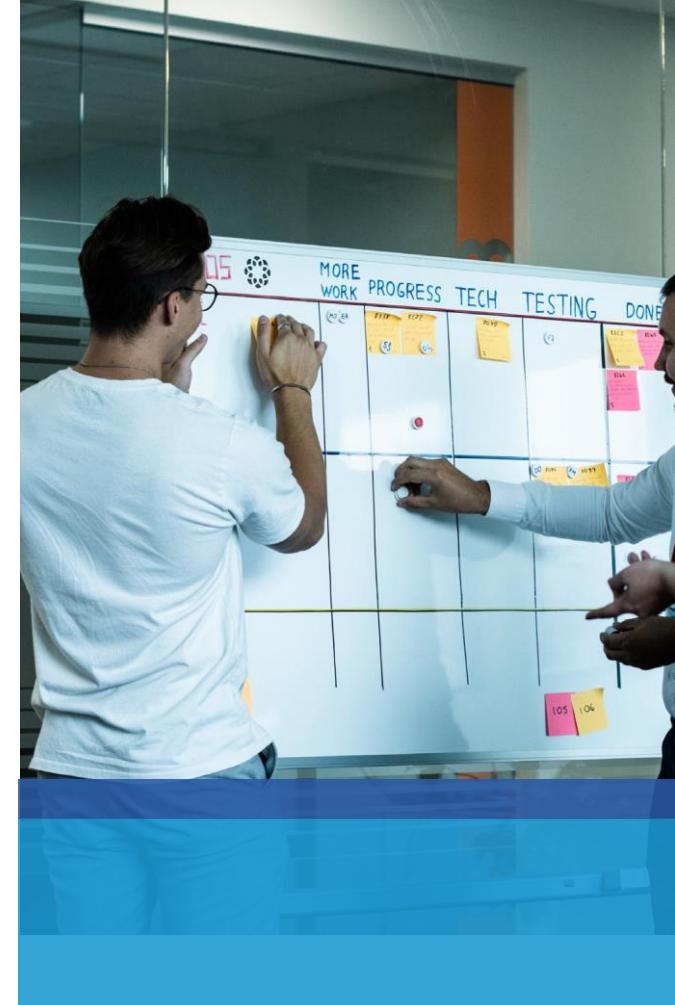


Los diagramas de interacción son un grupo especial de diagramas de comportamiento que muestran una **interacción**:

- Conjunto de objetos o roles y mensajes que pueden ser enviados entre ellos.
- Cubren la vista dinámica de un sistema.
- Los objetos interactúan para realizar colectivamente los servicios ofrecidos por las aplicaciones.

Un diagrama de interacción bien estructurado:

- Se encarga de modelar un aspecto de la dinámica de un sistema.
- Contiene solo aquellos elementos esenciales para comprender ese aspecto.
- Proporciona detalles de forma consistente con su nivel de abstracción. Es decir, solo debe mostrar aquellos adornos que son esenciales para su compresión.
- Debe proveer información suficiente para entender los aspectos importantes de la semántica de la interacción.



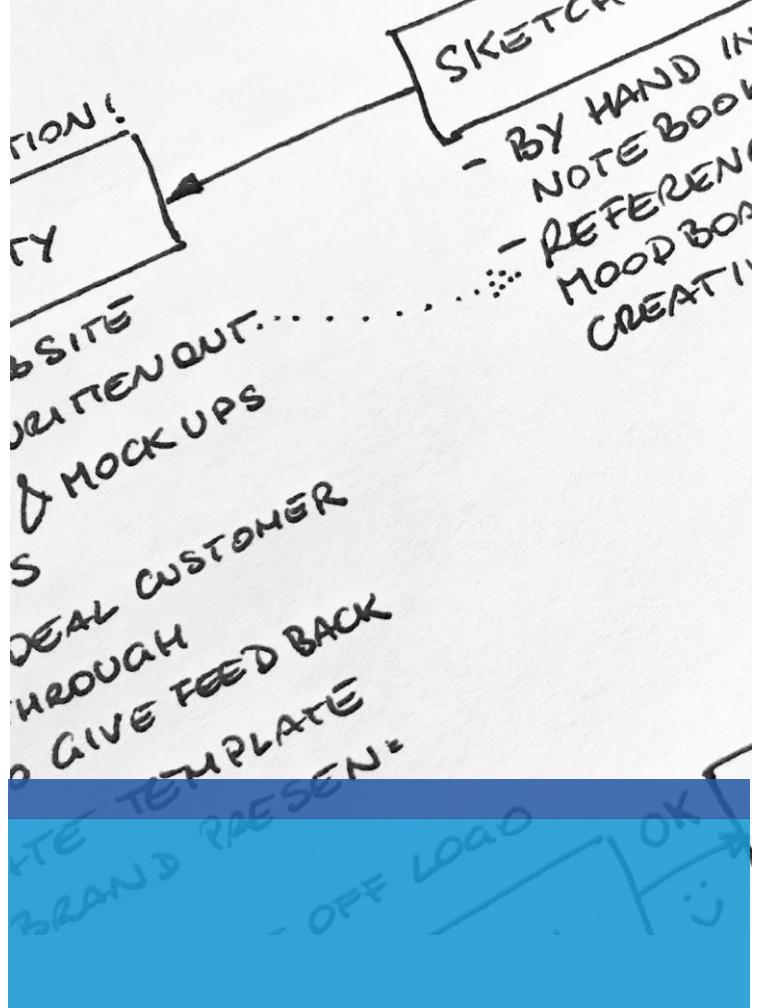
Al dibujar una interacción en UML se debe elegir su aspecto a destacar.

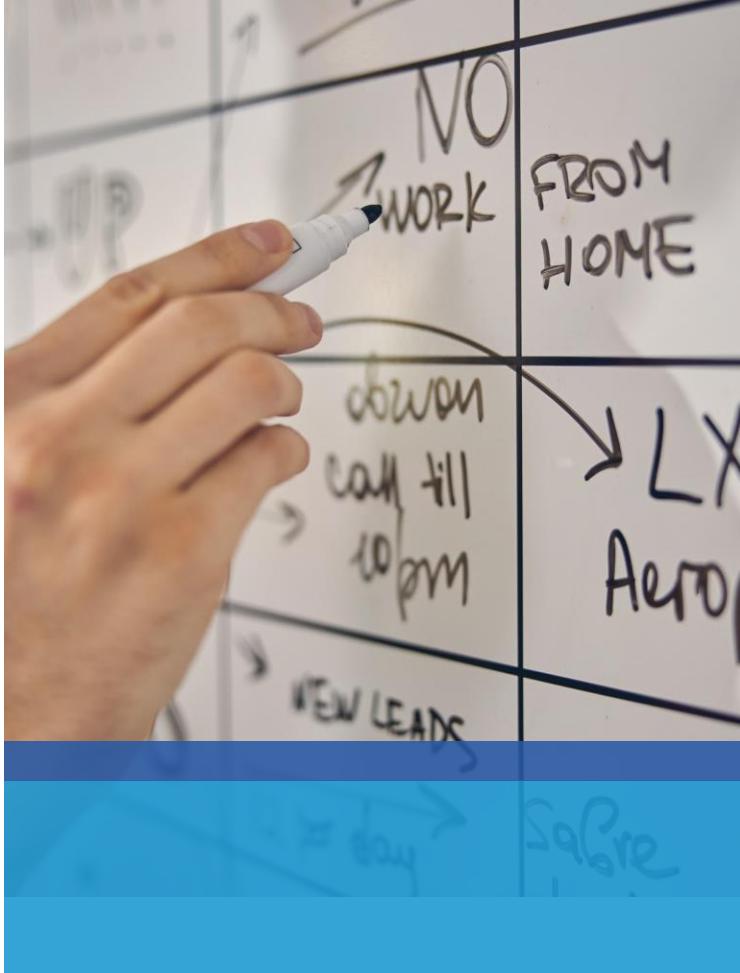
Ordenación temporal de los mensajes:

- Diagrama de Secuencia. Es la organización de los objetos implicados.
- Diagrama de Comunicación.

DIAGRAMAS DE INTERACCIÓN

- Los eventos en subsecuencias separadas solo están parcialmente ordenados.
- Se deben mostrar solo las propiedades importantes para comprender la interacción en su contexto:
 - Objetos: valores de atributos, rol y estado.
 - Mensajes: parámetros, semántica de ocurrencia, valor de retorno.



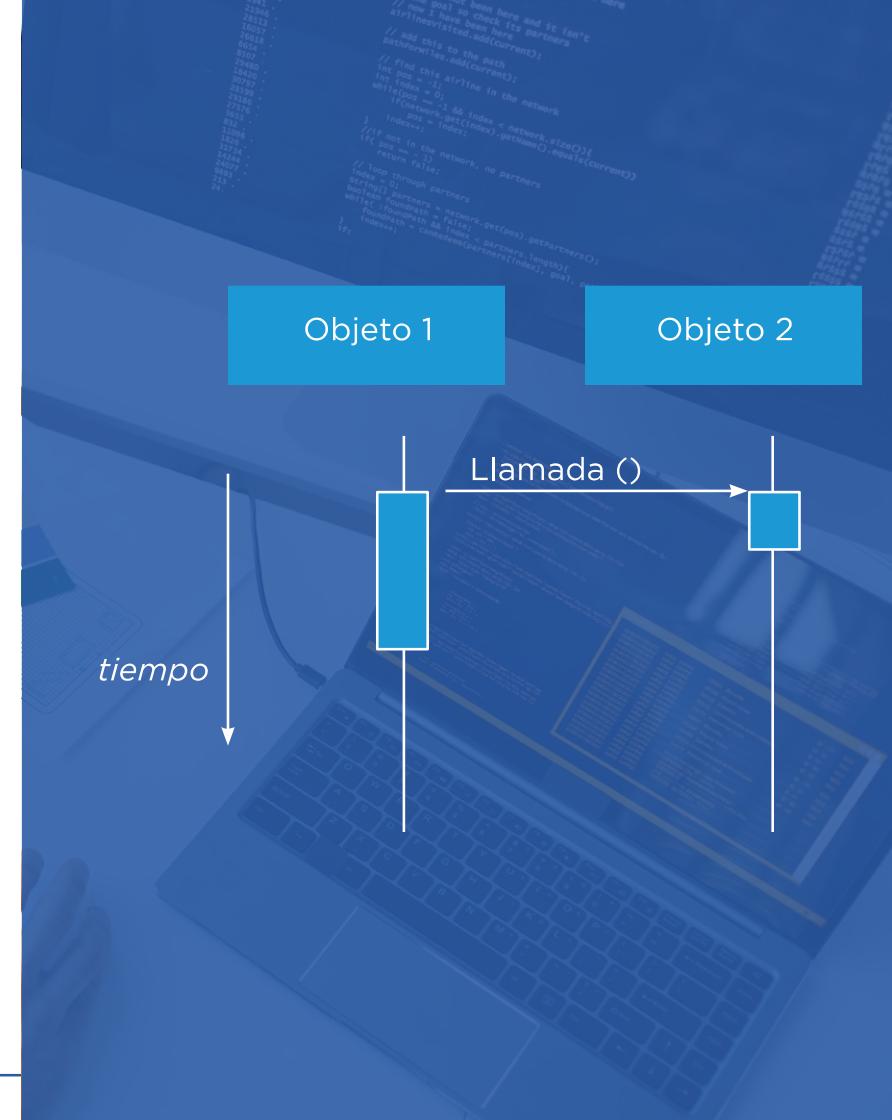


UML 2 incluye los siguientes diagramas en su clasificación de **diagramas de interacción**:

- Diagrama de Secuencias
- Diagrama de Comunicación
- Diagrama de Tiempos
- Diagrama de Revisión de las Interacciones

Diagrama de Secuencias:

- Es un diagrama de interacción que resalta la ordenación temporal de los mensajes intercambiados durante la interacción.
- Presenta un conjunto de roles, así como los mensajes enviados y recibidos por las instancias que interpretan dichos roles.
- Habitualmente, sirven para mostrar cómo interactúan unos objetos con otros en un caso de uso o un escenario de ejecución.



Tiene forma de tabla con objetos dispuestos en horizontal y mensajes en vertical, ordenados temporalmente.

En la siguiente imagen se presenta un ejemplo del diagrama de secuencias:



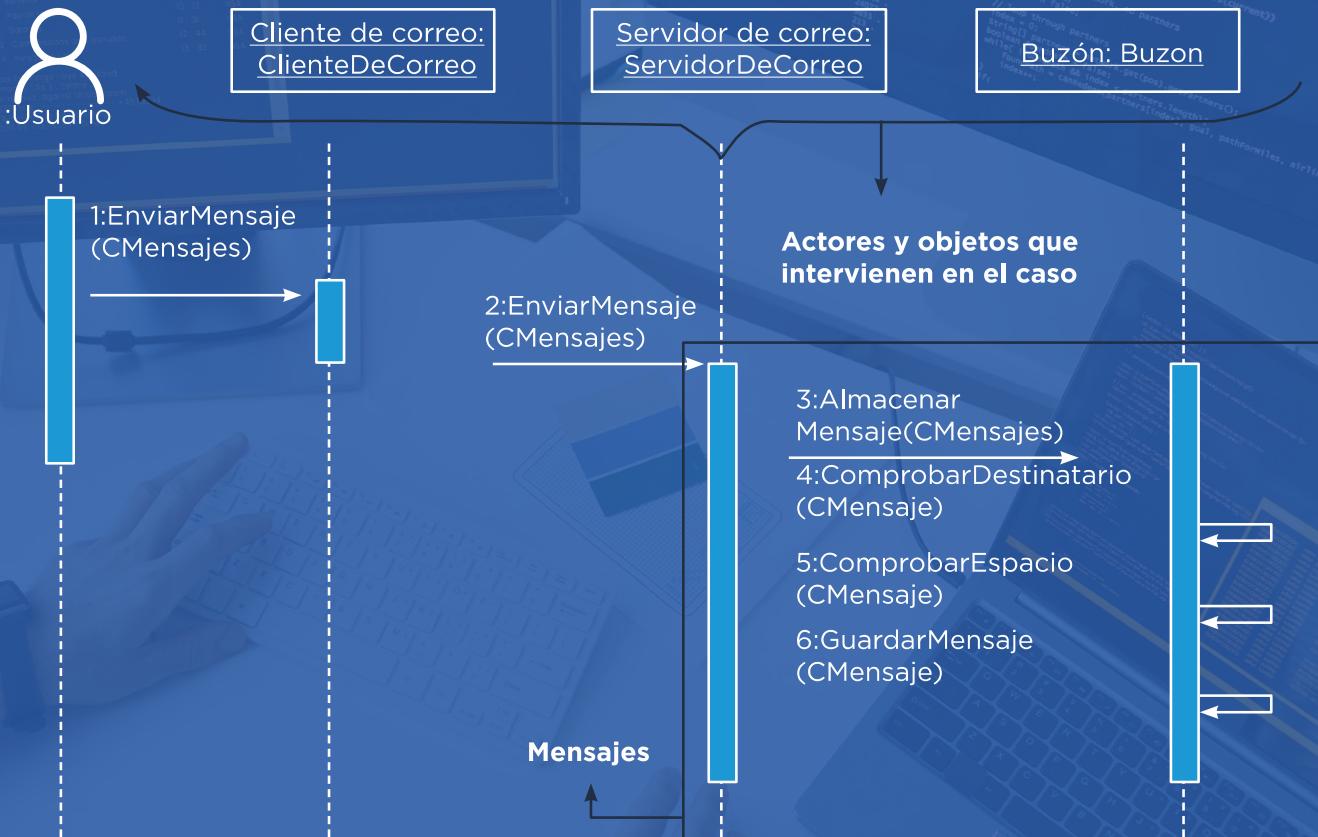


Creación de un Diagrama de Secuencias:

1. Los objetos que participan en la interacción deben ir en la parte superior del diagrama (eje “X”). Normalmente, el objeto que inicia la interacción va a la izquierda; y los objetos subordinados a la derecha.
2. Se colocan los mensajes que estos objetos envían y reciben a lo largo del eje “Y”. Deben colocarse en su orden de sucesión en el tiempo; desde arriba hasta abajo.

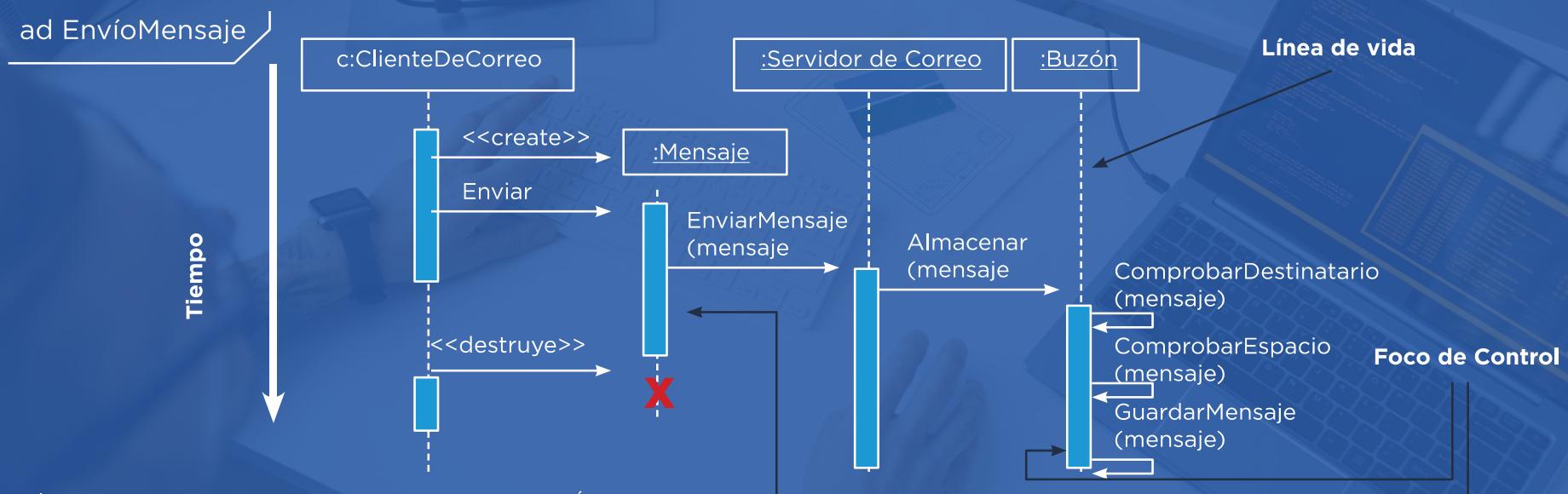
DIAGRAMAS DE INTERACCIÓN

Escenario normal (diagrama de secuencia): el mensaje llega a su destino



Los diagramas de secuencias cuentan con dos características que los distinguen de los de comunicación. A saber, la línea de vida y el foco de control o barra de activación:

La Línea de Vida. Representa la existencia de un objeto en un período de tiempo.



DIAGRAMAS DE INTERACCIÓN



El Foco de Control o barra de activación.

Representa el período de tiempo en el cual un objeto ejecuta una acción. Se muestra como un rectángulo en la línea de vida. La parte superior se alinea con el comienzo, y la inferior con su terminación. Además, puede tener un mensaje de retorno.

VIDEO

Te invitamos a ver el siguiente video:

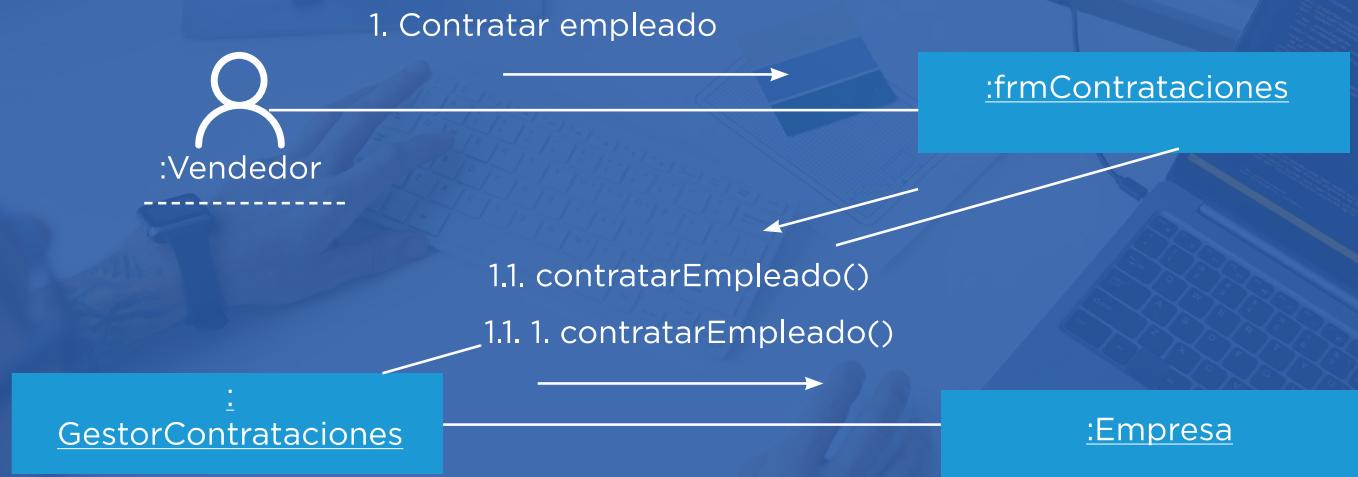


Diagrama de Comunicación:

- Es un diagrama de interacción que resalta la organización estructural de los objetos o roles que envían y reciben mensajes en la interacción.
- Muestra un conjunto de objetos, enlaces entre ellos, así como los mensajes enviados y recibidos.
- Se usan para modelar el comportamiento dinámico de un caso de uso.

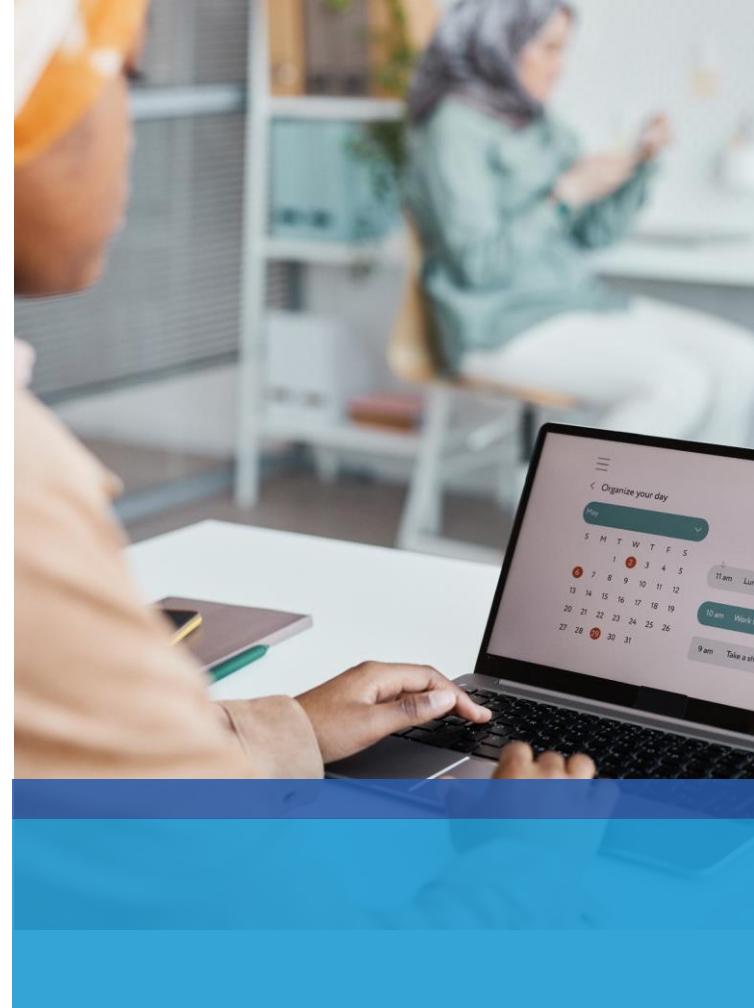


En la siguiente imagen se presenta un ejemplo de diagrama de comunicación:



Creación de un diagrama de comunicación:

- Colocar los objetos que participan en la interacción como nodos del grafo.
- Representar los enlaces que conectan esos objetos como arcos del grafo.
- Los enlaces se adornan con los mensajes que envían y reciben los objetos.
- Anotar las creaciones y destrucciones.

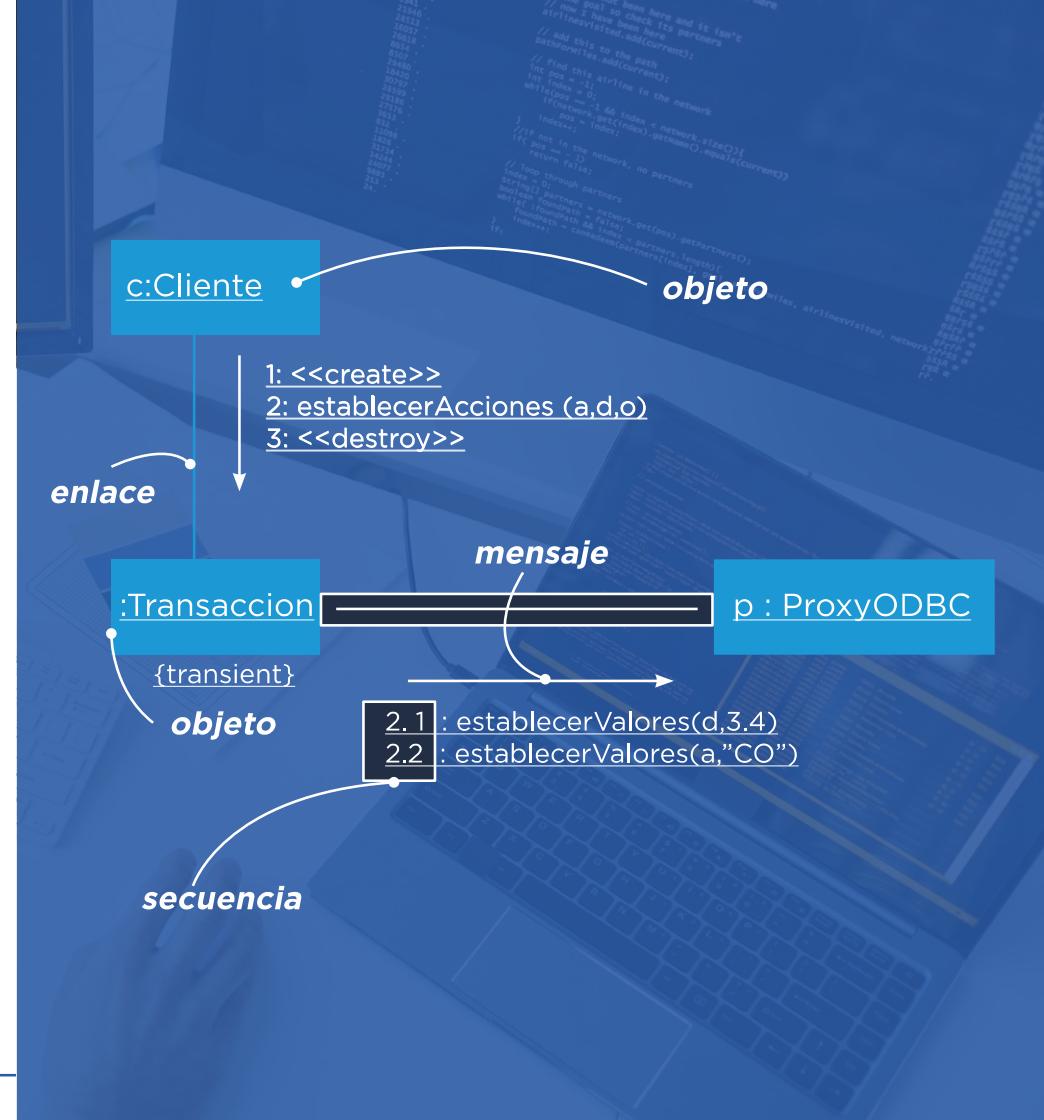


DIAGRAMAS DE INTERACCIÓN

Los diagramas de comunicación muestran de forma clara el flujo de control en el contexto de la organización estructural de los objetos que interaccionan.

Tienen dos **características** que los distinguen de los de secuencia:

- **Camino.** Indica cómo se enlaza un objeto a otro.
- **Número de secuencia.** Indica la ordenación temporal de los mensajes.



Existe **equivalencia semántica** entre los diagramas de secuencia y los de comunicación.

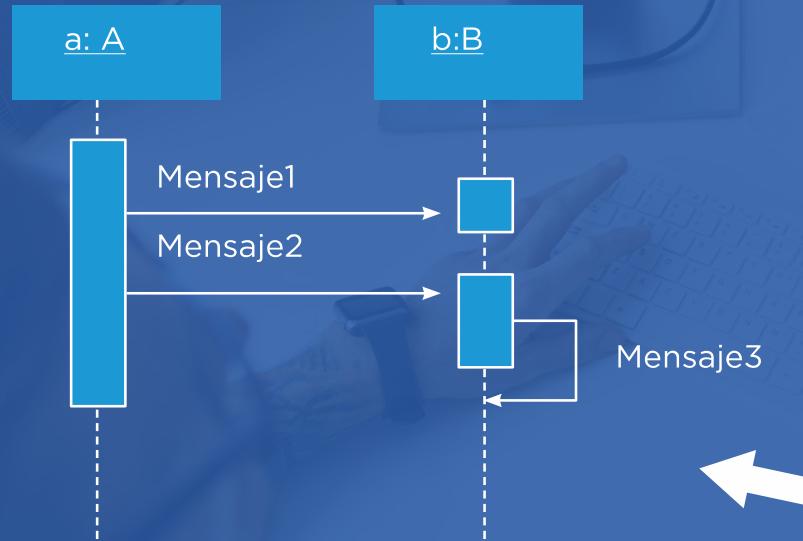


Diagrama de Secuencia

Se puede obtener automáticamente uno a partir del otro

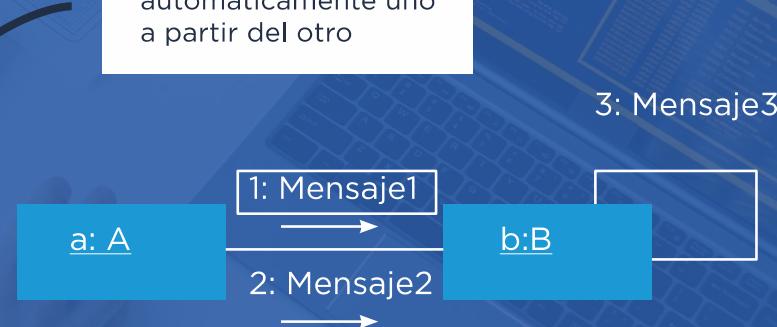
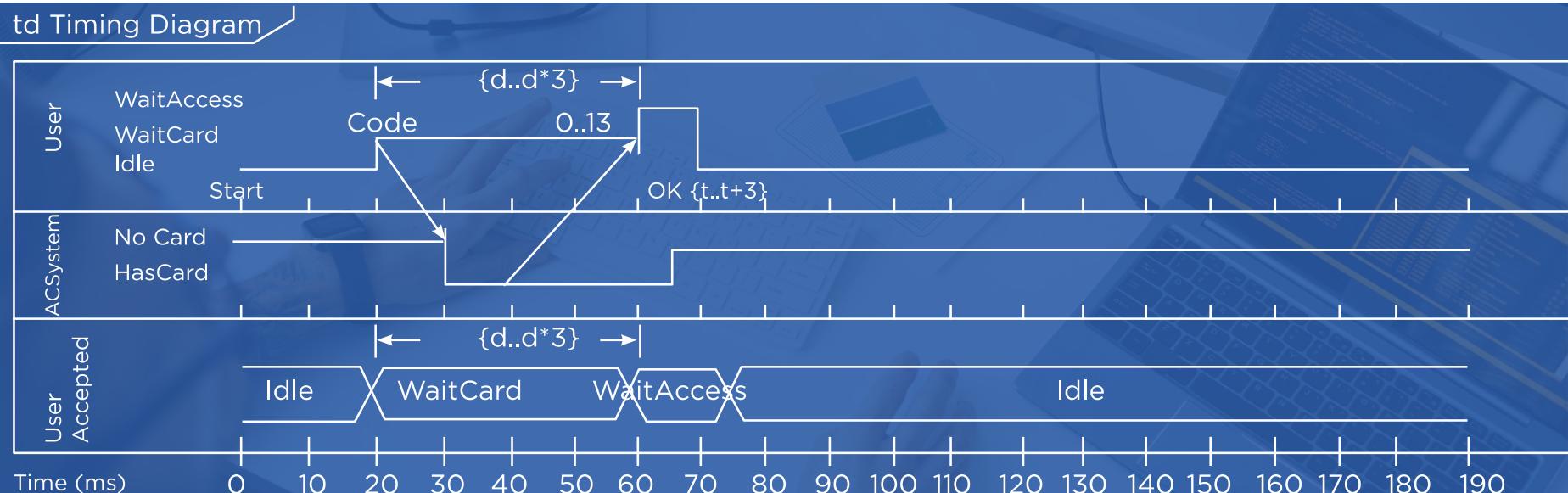


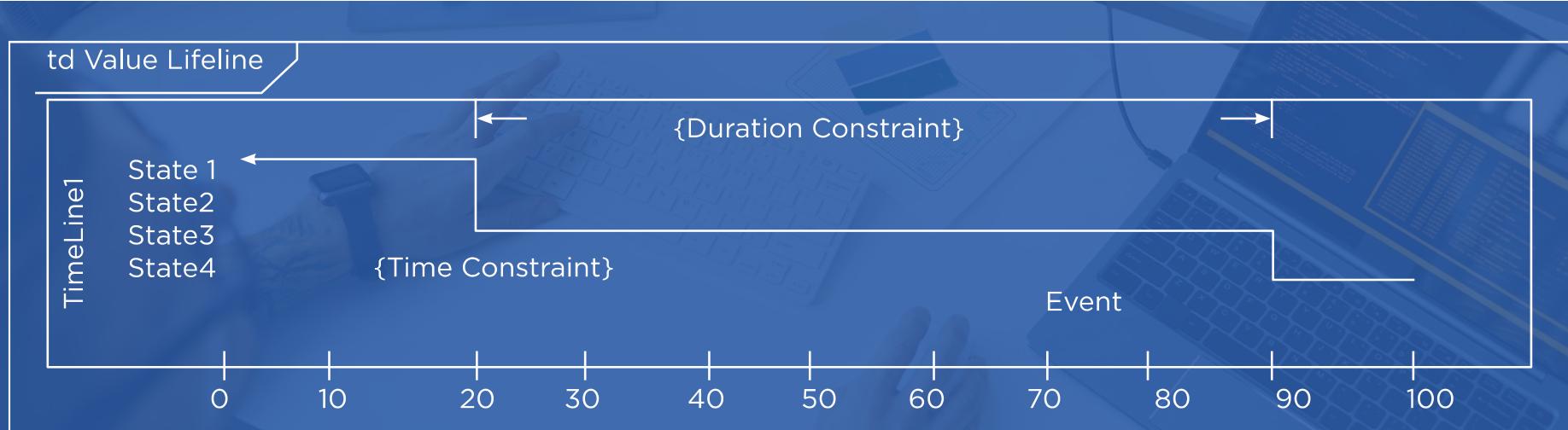
Diagrama de Tiempos:

- Muestra los tiempos reales en la interacción entre diferentes objetos o roles.
- Se observa el comportamiento de los objetos en un periodo determinado de tiempo.
- Es una forma especial de diagrama de secuencia.



Elementos del Diagrama de Tiempos:

- **Línea de vida del estado.** Muestra el cambio de estado de ítem en el tiempo. El eje “X” muestra el tiempo transcurrido en cualquier unidad; mientras que el eje “Y” se nombra con una lista de estados proporcionados.



- **Línea de vida del valor.** Muestra el cambio del valor de un ítem en el tiempo. El eje “X” muestra el tiempo transcurrido en cualquier unidad. El valor, por su parte, se muestra entre el par de líneas horizontales que se cruzan en cada cambio del valor.

Ambos gráficos (línea de vida del estado y línea de vida del valor) pueden combinarse para brindar una mejor comprensión e información complementada.

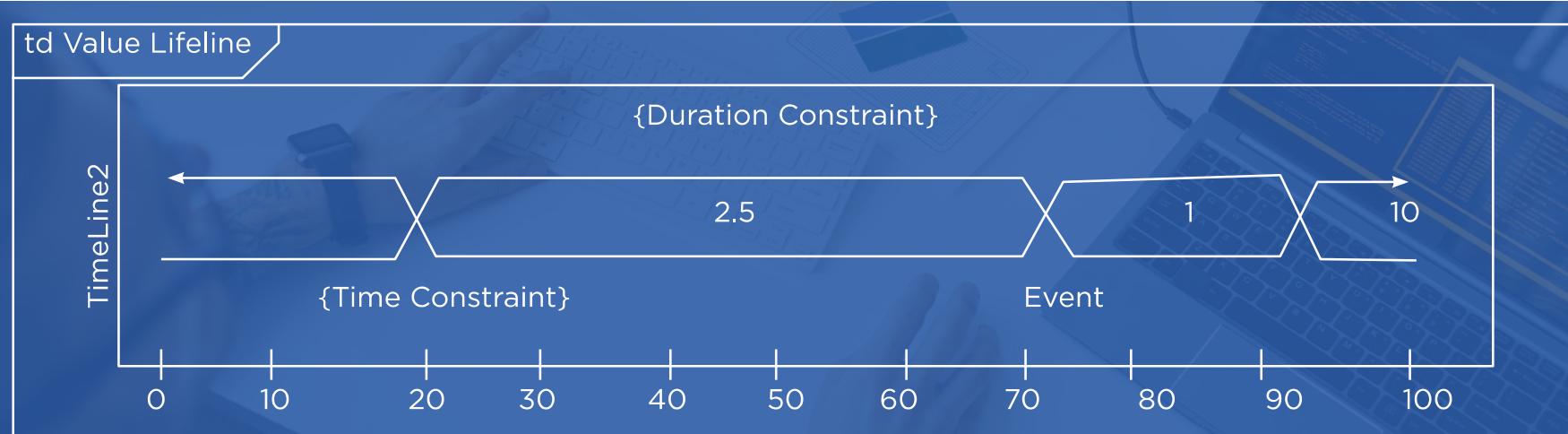
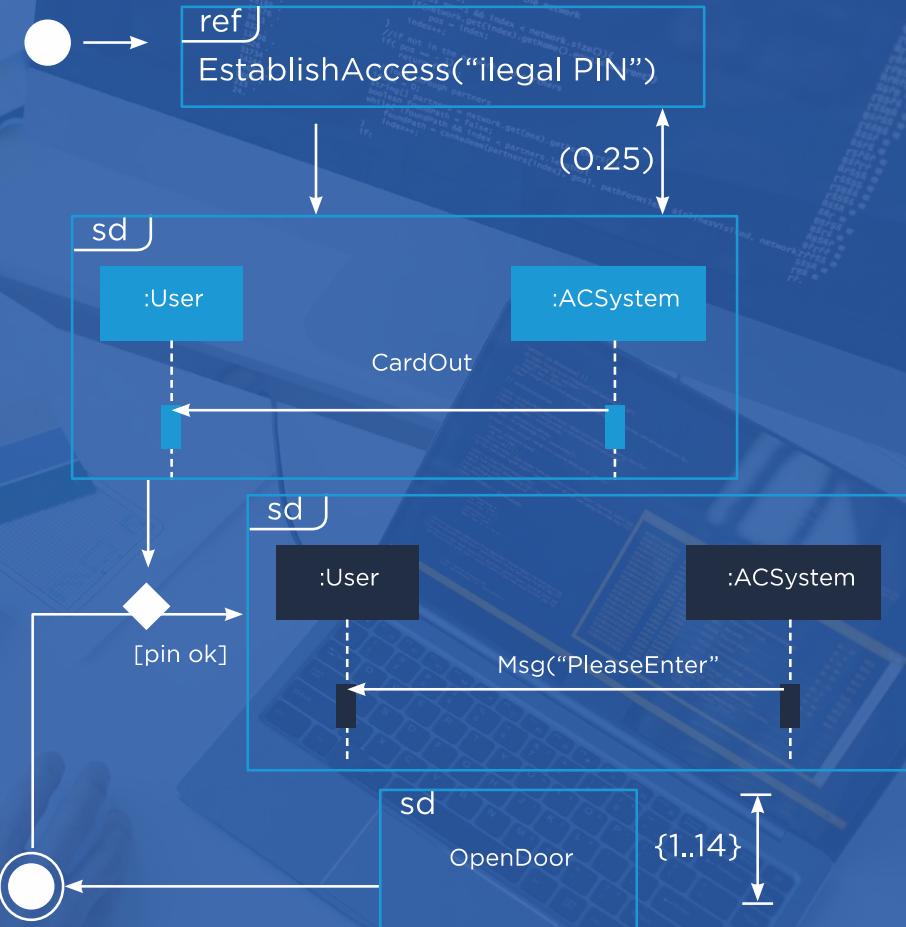


Diagrama de Revisión de Interacciones:

- Aportan una visión general del flujo de control de las interacciones.
- Es un híbrido entre un diagrama de actividad y un diagrama de secuencia.
- También llamado: **visión global de interacciones**.



Los diagramas de interacción muestran una secuencia de diagramas de interacción.

En términos simples, estos pueden ser conocidos como una colección de diagramas de interacción y el orden en que suceden.

ACTIVIDAD 1 - Unidad 2.

Modelado de casos de uso

Te invitamos a realizar la siguiente actividad:

Presiona el botón para descargar la actividad:

Presiona el botón para entregar la actividad:



LECTURAS PARA REFORZAR LA UNIDAD

- Debrauwer, L., van der Heyde, F., & van der Heyde, F. (2016). *UML* 2.5. Cap. 4,5. Ediciones ENI. ISBN:9782409003721

https://books.google.com.pe/books?id=sCU_bpElECAC&printsec=frontcover#v=onepage&q&f=false

- Seidl, M., Scholz, M., Huemer, C., & Kappel, G. (2015). *UML @ Classroom*. Springer Publishing.

https://petcomputacao.ufsc.br/wp-content/uploads/2020/06/2015_Book_UMLClassroom.pdf

- UML - *Diagrama de Interacción*. (2019).

[Vídeo]. YouTube.

<https://www.youtube.com/watch?v=srn6e0nz2b4>



CONCLUSIÓN



Dentro de los diagramas de comportamiento existen diferentes variantes. Es importante comprender la importancia de estos diagramas a la hora de diseñar los diferentes procesos que conforman un sistema. Antes que nada, se debe conocer la definición del sistema a alto nivel, para posteriormente profundizar en diagramas más detallados. Estos diagramas proporcionan una visión clara y detallada de los procesos a implementar, con los diferentes detonadores de cambio de estado de cada objeto.

Al dibujar un diagrama de interacción en UML es importante asignarle un nombre que comunique su propósito. De la misma manera, es importante elegir un tipo de diagrama acorde a la necesidad. A saber, el diagrama de secuencia enfatiza mensajes; por su parte, el de comunicación prima los objetos y distribuye sus elementos evitando los cruces de líneas. No es necesario que exista en un diseño todo tipo de diagramas. Lo que sí es importante, es que existan para entender el desarrollo del sistema, así como la documentación de los procesos correspondientes.



¡FELICIDADES!



Acabas de concluir la segunda unidad de tu curso *Lenguaje Unificado de Modelado*. Te invitamos a finalizar este esfuerzo realizando el examen parcial correspondiente. Para ello, debes regresar a la pantalla principal y dar clic en *Presentar examen*.

LENGUAJE UNIFICADO DE MODELADO



UNIDAD 3

DIAGRAMAS ESTRUCTURALES



TEMARIO

U3

3.1



Diagramas Particulares

3.2



Diagramas Globales

A photograph showing a man with curly hair and glasses, wearing a red sweater over a checkered shirt, standing at a whiteboard and writing with a marker. A woman with long blonde hair is visible from behind, looking at the board. The background is a bright office environment.

INTRODUCCIÓN

En esta tercera unidad conocerás sobre los conceptos fundamentales de los diagramas estructurales. Estos están clasificados en los modelos particulares y globales, de acuerdo al estándar de modelado unificado.

COMPETENCIAS A DESARROLLAR



1.

El alumno será capaz de identificar y diseñar los diagramas estructurales particulares para el modelado y diseño de sistemas de acuerdo al Lenguaje de Modelado Unificado.

2.

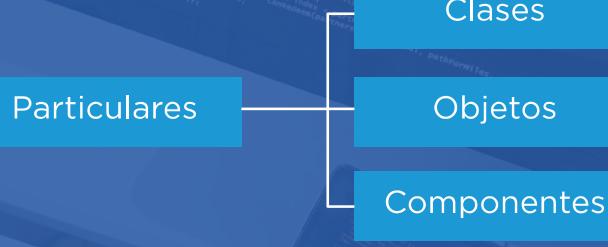
El alumno será capaz de identificar y diseñar los diagramas estructurales globales para el modelado y diseño de sistemas de acuerdo al Lenguaje de Modelado Unificado.

DIAGRAMAS PARTICULARES

Los diagramas estructurales representan la estructura estática de un sistema. Muestran diferentes niveles de abstracción e implementación, así como la jerarquía de componentes o módulos y cómo se conectan e interactúan entre sí.

Diagramas estructurales

Particulares



Globales



DIAGRAMAS PARTICULARES



Ofrecen orientación y garantizan que todas las partes de un sistema funcionen según lo previsto en relación con todas las demás partes.

Se organizan con base en los principales grupos de elementos que aparecen al modelar (durante las diferentes fases del proceso de desarrollo).

Tipo de diagrama	Elementos centrales
Clases	Clases, interfaces, colaboraciones
Componentes	Componentes
Estructura compuesta	Estructura interna de clase o componentes
Objetos	Objetos
Paquetes	Paquetes
Despliegue	Nodos, artefactos



Diferencias entre los diagramas estructurales y los de comportamiento:

Los diagramas estructurales muestran la estructura estática del sistema y sus partes en diferentes niveles de abstracción. Por su parte, los diagramas de comportamiento muestran cómo se comporta un sistema de información de forma dinámica. Es decir, describen los cambios que sufre un sistema a través del tiempo cuando están en ejecución.

VIDEO

Te invitamos a ver el siguiente video:



DIAGRAMAS PARTICULARES

Diagrama de Clases:

Muestra la estructura del sistema, subsistema o componente, utilizando clases con sus características.

Este diagrama (el más común en el desarrollo de software) tiene un aspecto similar al del diagrama de flujo, porque las clases se representan con cuadros.

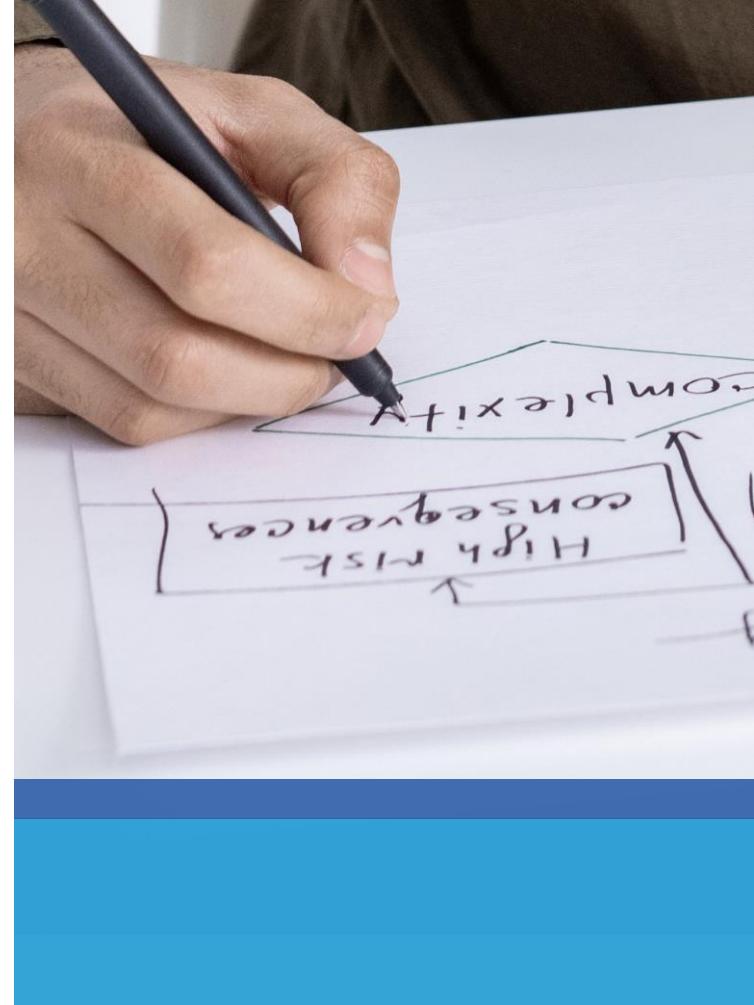


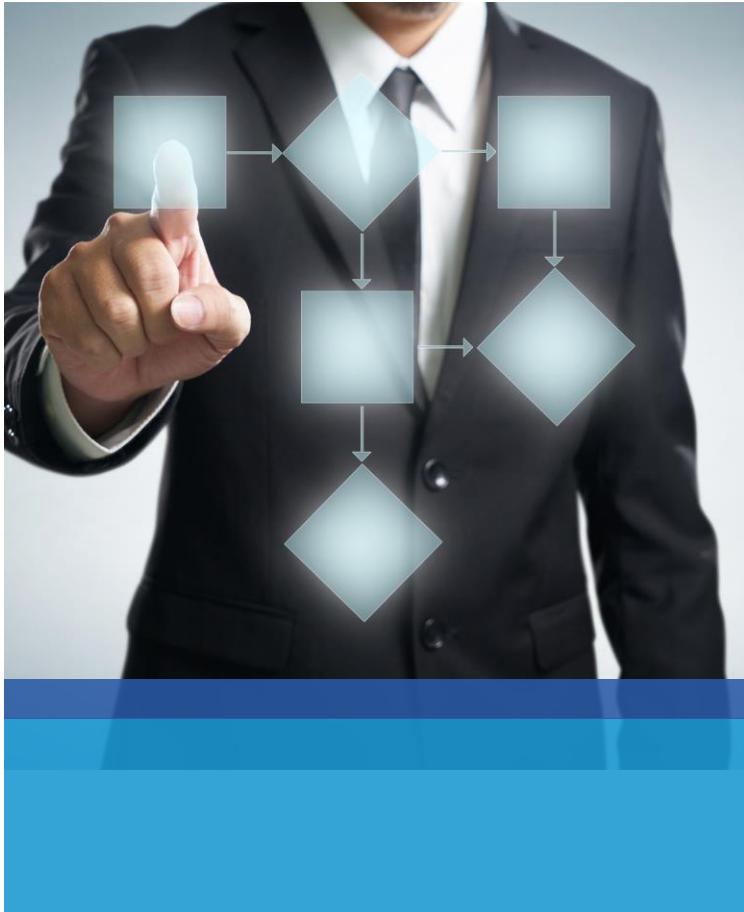
DIAGRAMAS PARTICULARES

El diagrama de clases abarca la vista de diseño estática de un sistema. La notación de este tipo de diagrama es la base para la mayor parte del resto de diagramas estructurales.

Principalmente, un **diagrama de clases** contiene:

- Clases (con atributos, operaciones y visibilidad).
- Relaciones: de dependencia, de generalización, de asociación, de agregación y de composición.





DIAGRAMAS PARTICULARES

Clases. Un rectángulo es el símbolo que representa a la clase. Un diagrama de clases está formado por varios rectángulos de este tipo, los cuales están conectados por líneas que representan las asociaciones o maneras en que las clases se relacionan entre sí.

DIAGRAMAS PARTICULARES

Las clases se representan con rectángulos.

Se dividen en **tres áreas**:

- Sección superior: nombre de clase.
- Sección central: atributos de clase.
- Sección inferior: métodos u operaciones de clase.

```
21944 // now has been here and it's not  
21933 // already visited, add current  
16037 // add shift to the path  
26013 partnersShift.addCurrent();  
8634 //  
8586 //  
23440 // find this airline in the network  
18420 int pos = -1;  
30315 while(pos < 0){  
23199 if(index < network.size() & network.get(index).getAirlines().contains(current))  
23186 index++;  
27513 }  
8731 //  
8586 //  
312098 //  
312144 //  
312144 //  
24001 //  
24001 //  
24 //  
21944 // now has been here and it's not  
21933 // already visited, add current  
16037 // add shift to the path  
26013 partnersShift.addCurrent();  
8634 //  
8586 //  
23440 // find this airline in the network  
18420 int pos = -1;  
30315 while(pos < 0){  
23199 if(index < network.size() & network.get(index).getAirlines().contains(current))  
23186 index++;  
27513 }  
8731 //  
8586 //  
312098 //  
312144 //  
312144 //  
24001 //  
24001 //  
24 //  
21944 // now has been here and it's not  
21933 // already visited, add current  
16037 // add shift to the path  
26013 partnersShift.addCurrent();  
8634 //  
8586 //  
23440 // find this airline in the network  
18420 int pos = -1;  
30315 while(pos < 0){  
23199 if(index < network.size() & network.get(index).getAirlines().contains(current))  
23186 index++;  
27513 }  
8731 //  
8586 //  
312098 //  
312144 //  
312144 //  
24001 //  
24001 //  
24 //
```

Nombre de Clase

atributo: Tipo
/atributo Derivado

operación ()

DIAGRAMAS PARTICULARES

El diagrama de clases está indicado sobre todo como ejemplo para un diagrama de estructura.

Otros diagramas de esta categoría utilizan componentes modificados del diagrama de clases para su notación.

```
21944 // now have been here and is there
22813 // if not visited add current
23027 // add shift to the path
26042 // find this airline in the network
26541 // if not < 0
26542 // if index < network.size()
29480 // if no partners
29481 // if no partners
30320 // if no partners
30321 // if no partners
32189 // if no partners
32190 // if no partners
27103 // if no partners
28123 // if no partners
23208 // if no partners
23213 // if no partners
23214 // if no partners
24001 // if no partners
24002 // if no partners
24 // if no partners
```

Aviones

modelo de avión
cantidad de motores
velocidad de crucero
carga útil

acelerar ()
elevarse ()
girar ()
descender ()
desacelerar ()

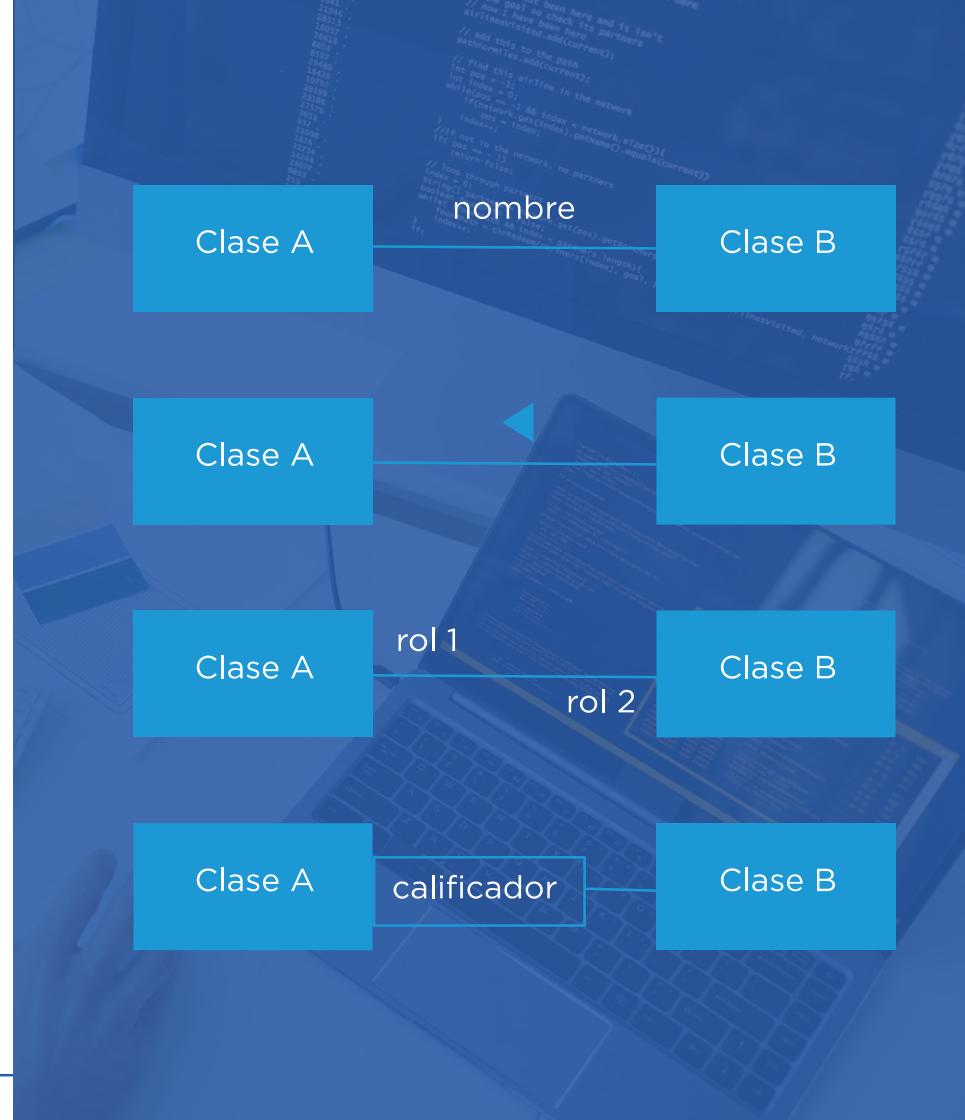
VIDEO

Te invitamos a ver el siguiente video:



DIAGRAMAS PARTICULARES

Relaciones. Las asociaciones son las que representan a las relaciones estáticas entre las clases. El nombre de la asociación va sobre o por debajo de la línea que la representa. Una flecha rellena indica la dirección de la relación. Los roles se ubican cerca del final de una asociación. Estos representan la manera en que dos clases se ven entre ellas.



DIAGRAMAS PARTICULARES

Multiplicidad. Las notaciones utilizadas para señalar la multiplicidad se colocan cerca del final de una asociación.

Estos símbolos indican el número de instancias de una clase vinculadas a una de las instancias de la otra clase.

1 no mas de uno

0..1 cero o uno

*

muchos

0..* cero o muchos

1..* uno o muchos

Empresa

1

1..*

Empleado

DIAGRAMAS PARTICULARES

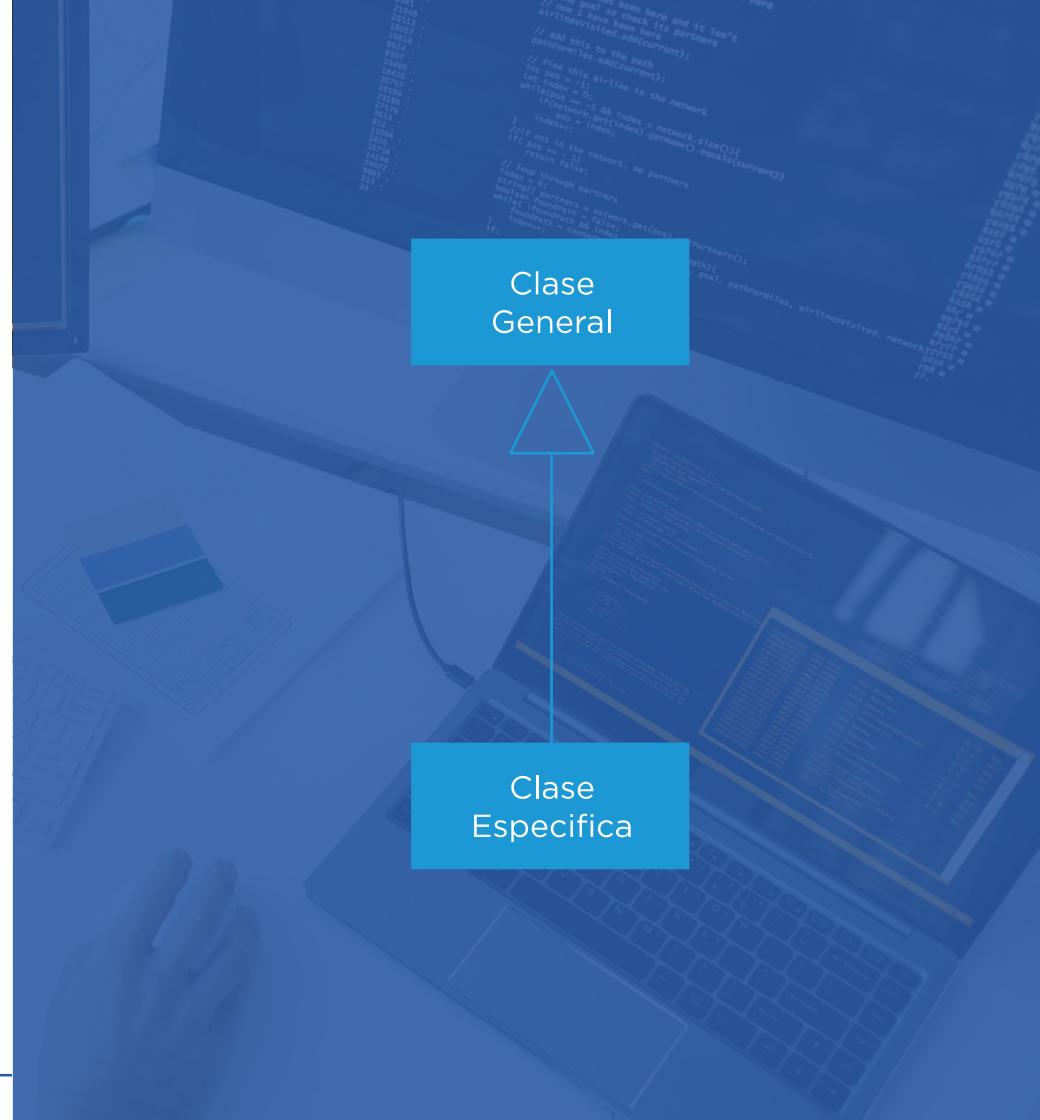
Composición. Es un tipo especial de agregación que denota una fuerte posesión de la clase *Todo* a la Clase *Parte*. Se grafica con un rombo diamante relleno contra la clase que representa el todo.

Agregación. Es una relación en la que la clase *Todo* juega un rol más importante que la Clase *Parte*. Las dos clases no son dependientes una de otra. Se grafica con un rombo diamante vacío contra la clase *Todo*.



DIAGRAMAS PARTICULARES

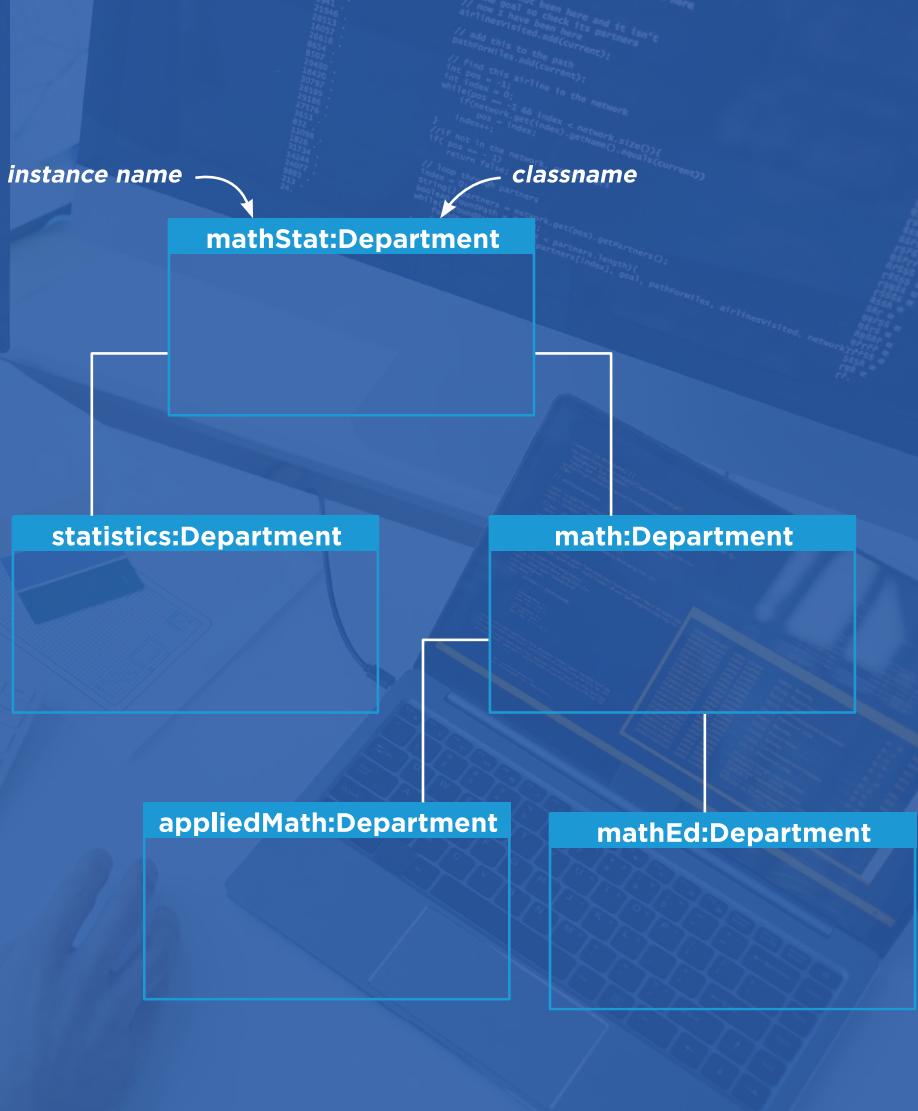
Generalización. Es otro nombre para herencia. Se refiere a una relación entre dos clases en donde una clase *Específica* es una versión especializada de la otra, o clase *General*.



DIAGRAMAS PARTICULARES

Diagrama de Objetos:

- Muestra los objetos de un sistema y sus relaciones. Además, ofrece una mejor visión de los potenciales defectos de diseño que necesitan reparación.
- Se usa como una forma de comprobar la revisión de un diagrama de clases para fines de precisión.
- Describe la vista de diseño estática pero desde el punto de vista de casos reales



DIAGRAMAS PARTICULARES



Los diagramas de objetos están vinculados con los diagramas de clases.

Un objeto es una instancia de una clase, por lo que un diagrama de objetos puede ser visto como una instancia de un diagrama de clases.

Los diagramas de objetos describen la estructura estática de un sistema en un momento particular. Son utilizados para probar la precisión de los diagramas de clases.

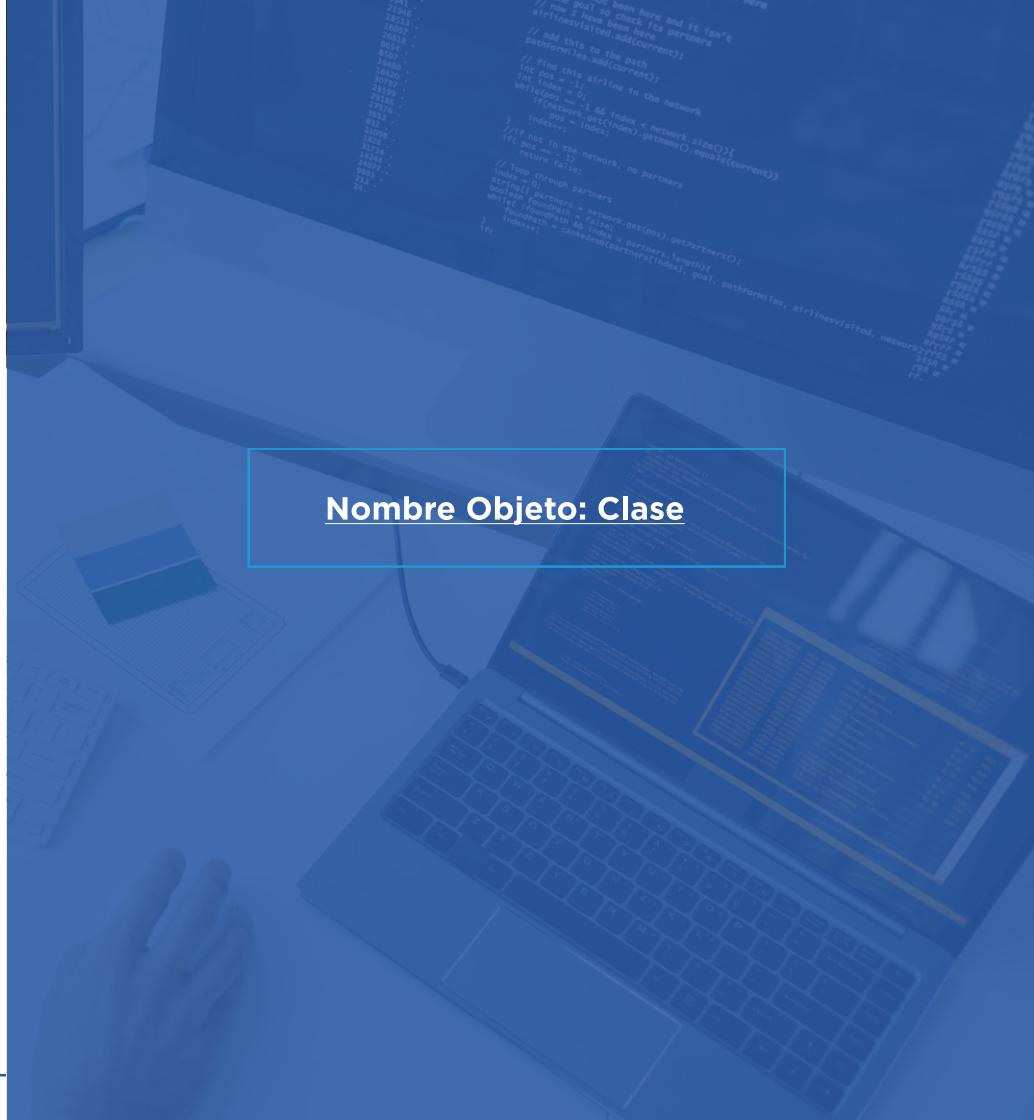
VIDEO

Te invitamos a ver el siguiente video:



DIAGRAMAS PARTICULARES

Elementos. Es el nombre que reciben los objetos. Cada objeto es representado como un rectángulo, que contiene el nombre del objeto y su clase subrayadas y separadas por dos puntos.



DIAGRAMAS PARTICULARES

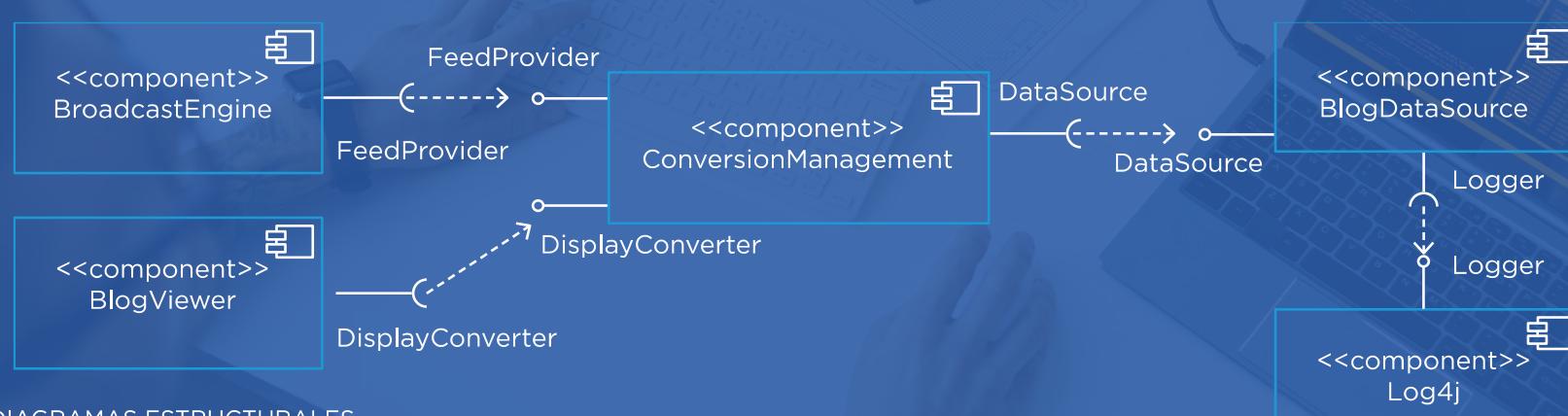
Atributos. Como con las clases, los atributos se listan en un área inferior. Sin embargo, los atributos de los objetos deben tener un valor asignado.

Nombre Objeto: Clase

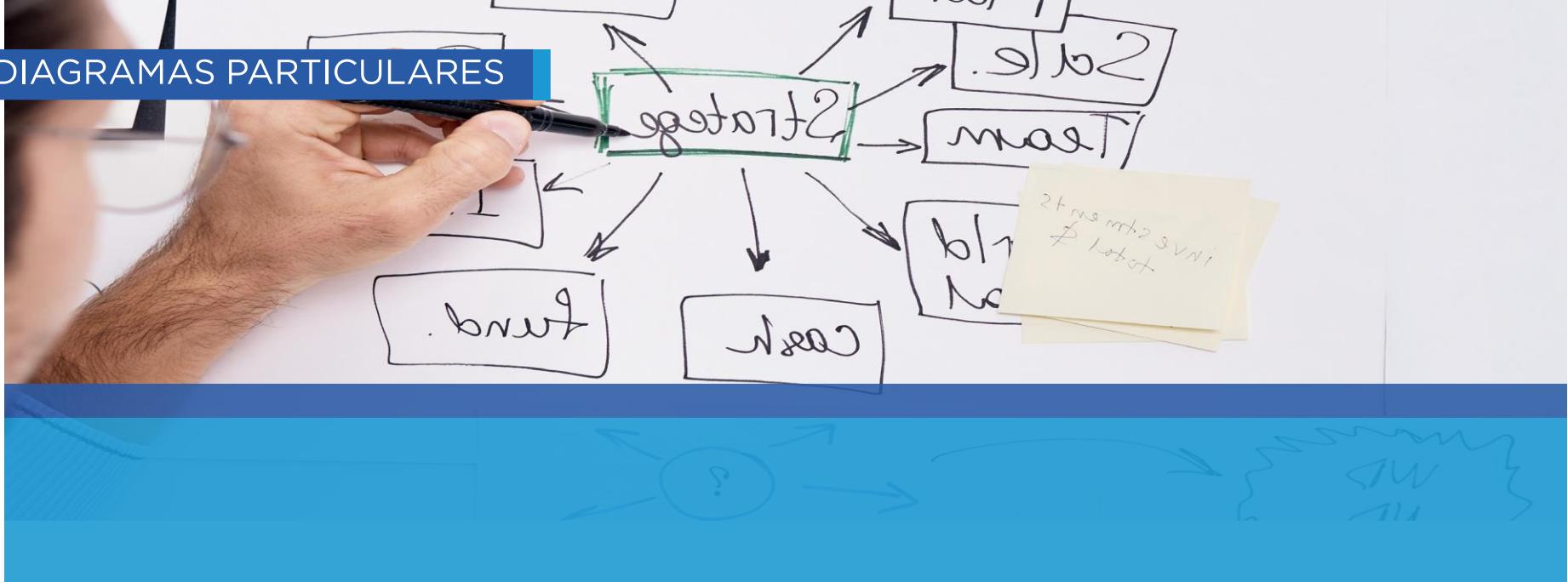
Atributo tipo = ‘Valor’
Atributo tipo = ‘Valor’
Atributo tipo = ‘Valor’
Atributo tipo = ‘Valor’

Diagrama de Componentes:

- Describe la organización de los componentes físicos de un sistema.
- Ofrece una vista más simplificada de un sistema complejo al desglosarlo en componentes más pequeños.
- Muestra componentes y dependencias entre ellos.
- Cubre la vista de la implementación estática del diseño de un sistema.



DIAGRAMAS PARTICULARES



Cada una de las piezas se muestra como una caja rectangular, que tiene su nombre escrito dentro.

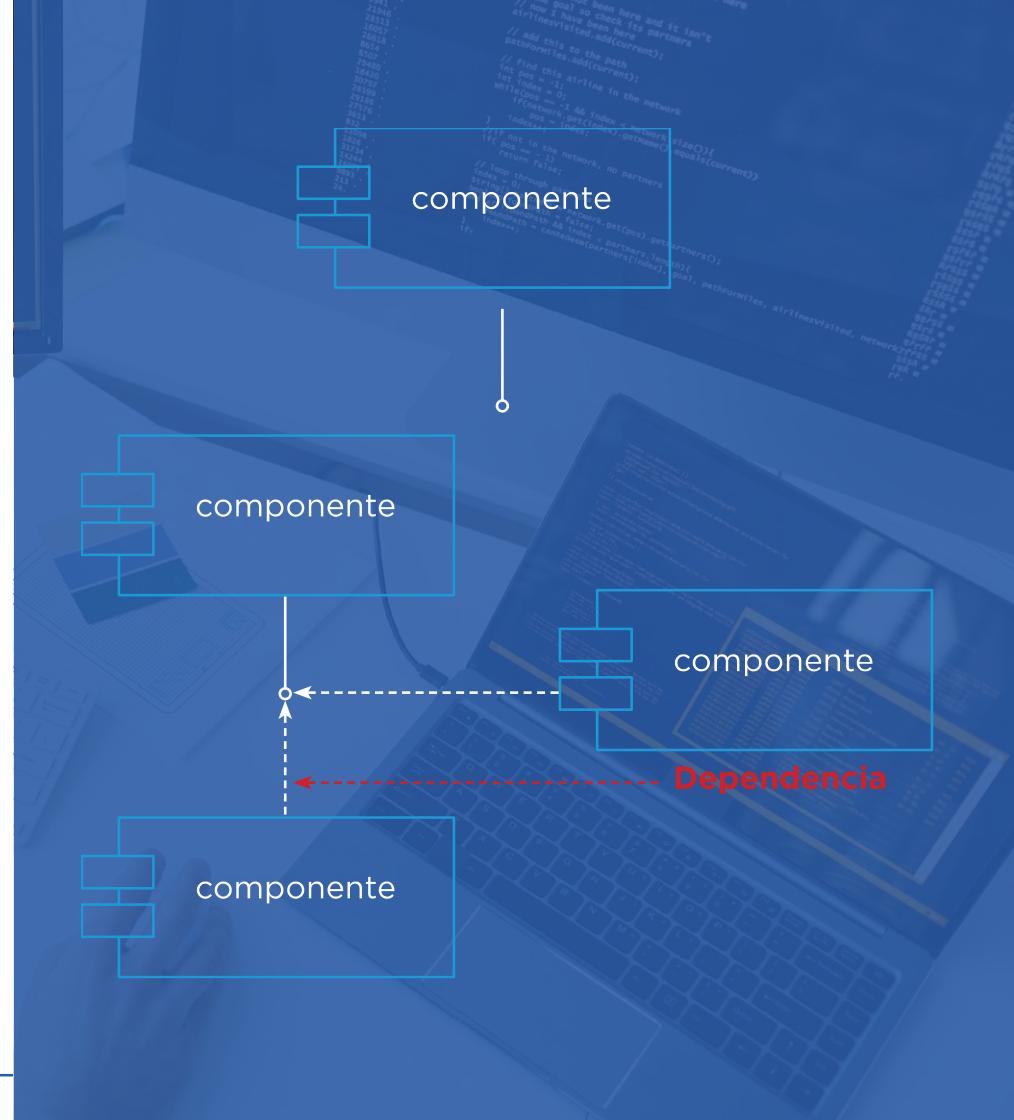
Los conectores definen la relación y las dependencias entre los diferentes componentes.

DIAGRAMAS PARTICULARES

Componente. Es un bloque de construcción física del sistema.

Interfaz. Describe a un grupo de operaciones usadas o creadas por componentes.

Dependencias. Estas se grafican usando flechas de puntos.



DIAGRAMAS GLOBALES

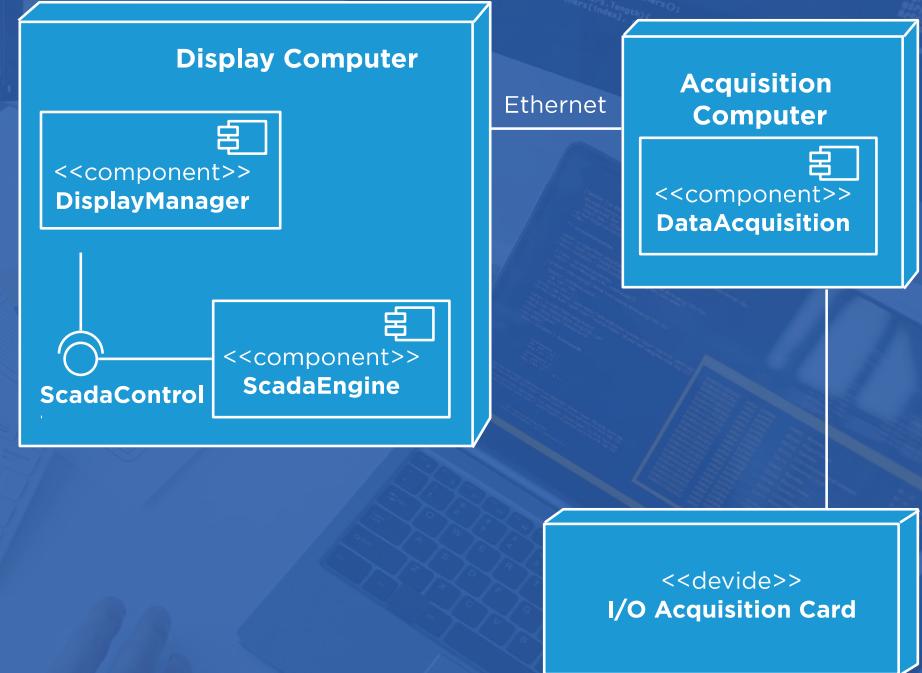


Diagrama de Despliegue:

- Este diagrama muestra los componentes de hardware (nodos) y software (artefactos) y sus relaciones.
- Ofrece una representación visual exacta del lugar donde se implementa cada componente de software.
- Muestra la arquitectura del sistema como despliegue (distribución) de artefactos de software.

DIAGRAMAS GLOBALES

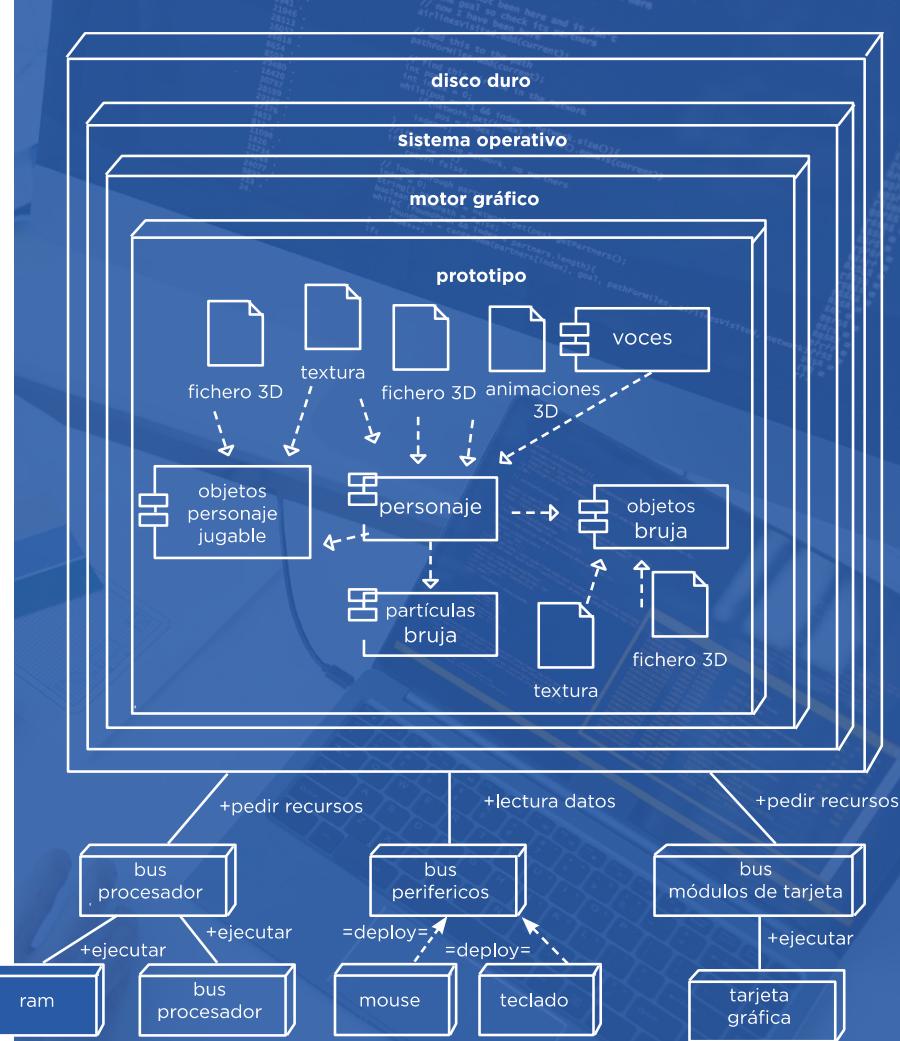
- Describe la vista de despliegue estática de una arquitectura.
- Cada nodo (hardware) suele albergar uno o más componentes.
- Muestran el hardware, el software y el *middleware*.



Funcionamiento:

- La vista de despliegue representa el despliegue de artefactos de tiempo de ejecución sobre nodos.
- Los diagramas de despliegue, junto con los diagramas de componentes, forman parte de la arquitectura física.

“La arquitectura física es una descripción detallada del sistema que muestra la asignación de artefactos de software a nodos físicos.” (García, Moreno, & García, 2018).



VIDEO

Te invitamos a ver el siguiente video:



Elementos:

Nodo. Es la representación de un sistema dirigido al usuario o autónomo y más frecuentemente de un hardware. Está representado como una caja a la cual se le agrega el nombre.



DIAGRAMAS GLOBALES

Mediante iconos especializados se puede precisar la naturaleza de los nodos como constituyentes físicos (dispositivos, archivos, bases de datos, etc.) de un sistema.



Desplegar. Se trata de una indicación, la cual da a entender que un nodo usa, de una manera dependiente y no aclarada, a otro nodo.

Comunicador. Por lo general, se usa cuando se explica de una manera explícita cómo es que usa un nodo a otro, y con qué cantidad.

```
21944 // now goal has been here and it's been visited, so check its partners
21945 // if partner not visited, add it to the path
21946 // add shift to the path
21947 partners[i].addCurrent();
21948 // Find the airline in the network
21949 for (int pos = 0; pos < network.size(); pos++) {
21950     if (pos == -1 || index < network[pos].size() &&
21951         network[pos].get(index).getname() == current) {
21952         if (index < partners[i].size()) {
21953             partners[i].add(index);
21954         }
21955     }
21956 }
21957 if (pos >= 0) {
21958     if (partners[i].size() > 0) {
21959         partners[i].add(pos);
21960     }
21961 }
21962 }
21963 if (pos >= 0) {
21964     if (partners[i].size() > 0) {
21965         partners[i].add(pos);
21966     }
21967 }
21968 }
21969 if (pos >= 0) {
21970     if (partners[i].size() > 0) {
21971         partners[i].add(pos);
21972     }
21973 }
21974 }
21975 if (pos >= 0) {
21976     if (partners[i].size() > 0) {
21977         partners[i].add(pos);
21978     }
21979 }
21980 }
21981 if (pos >= 0) {
21982     if (partners[i].size() > 0) {
21983         partners[i].add(pos);
21984     }
21985 }
21986 }
21987 if (pos >= 0) {
21988     if (partners[i].size() > 0) {
21989         partners[i].add(pos);
21990     }
21991 }
21992 }
```

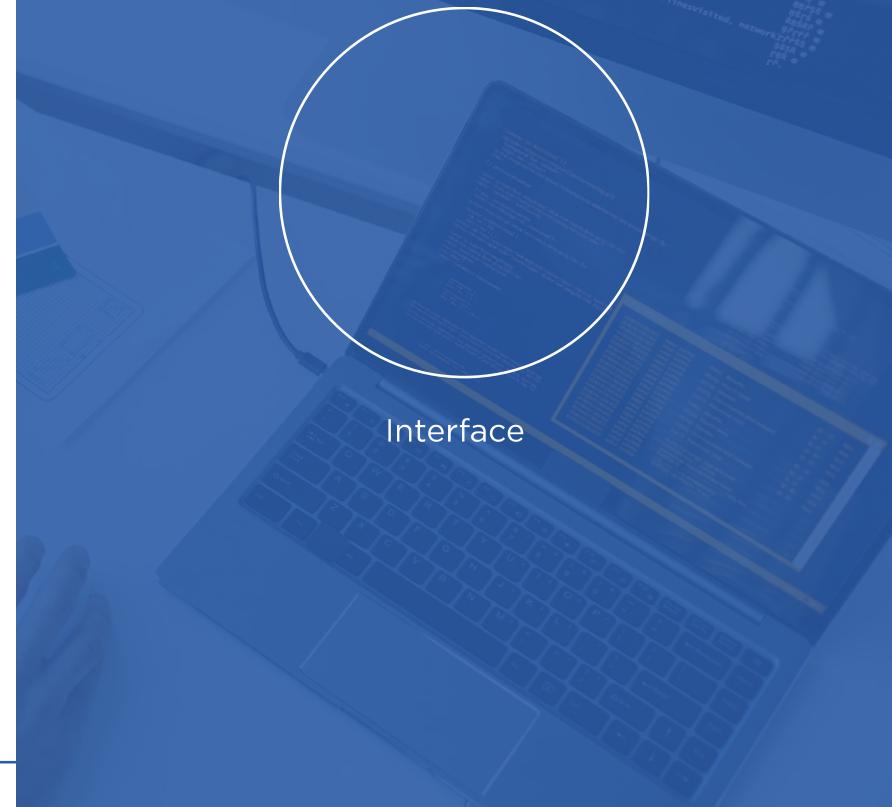
<<deploy>>



DIAGRAMAS GLOBALES

Interfaz. Hace parte del módulo con la que un componente se comunica con otros iguales, y se pone dentro de un nodo.

```
21944; // now have been here and is in the
23313; // airtimevisited.add(current);
23037; // add shift to the path
20632; // find this airline in the network
8854; if (airline == -1 && index < network.size()) {
8590; partners[index].partners();
85420; if (airline == -1 &amp; partners[index].size() <
82940; network[0].size()) {
82940; partners[index].partners();
30321; if (airline == -1 && partners[index].size() <
30321; network[0].size()) {
28199; partners[index].partners();
28199; if (airline == -1 && partners[index].size() <
27153; network[0].size()) {
27153; partners[index].partners();
8313; if (airline == -1 && partners[index].size() <
8313; network[0].size()) {
83098; partners[index].partners();
32124; if (airline == -1 && partners[index].size() <
32124; network[0].size()) {
32124; partners[index].partners();
24077; if (airline == -1 && partners[index].size() <
24077; network[0].size()) {
24077; partners[index].partners();
24;
```



Artefacto. Es la representación de los ficheros con los que se trabaja. Estos deben tener una connotación un tanto externa, como una factura o un archivo de texto, los cuales se colocan dentro del nodo.

```
21944 // now have been here and is in the
28313 // networkvisited.add(current);
16037 // add shift to the path
20612 // partners);
8524 // add shift to the path
8506 // partners);
29440 // find this airline in the network
18420 // int pos = -1;
30121 // int index = 0;
28199 // while(index < network.size()){
28188 //     if(network.get(index).getname().equals(current))
27153 //         if(index == -1)
28313 //             pos = index;
28308 //         else
23208 //             index++;
32734 //         if(index >= network.size())
32734 //             return false;
24077 //         else
24077 //             index++;
24 //     }
24 // }
```



Artefacto

Componente. La mayoría de las veces es representado como una especie de clase o subclase. Esta demuestra, por lo general, un conjunto de objetos dentro. Este se coloca dentro del nodo.

Dependencia. Demuestra que un componente tiene una existencia dependiente del componente al que señala. Nunca debe de salir del nodo.



DIAGRAMAS GLOBALES

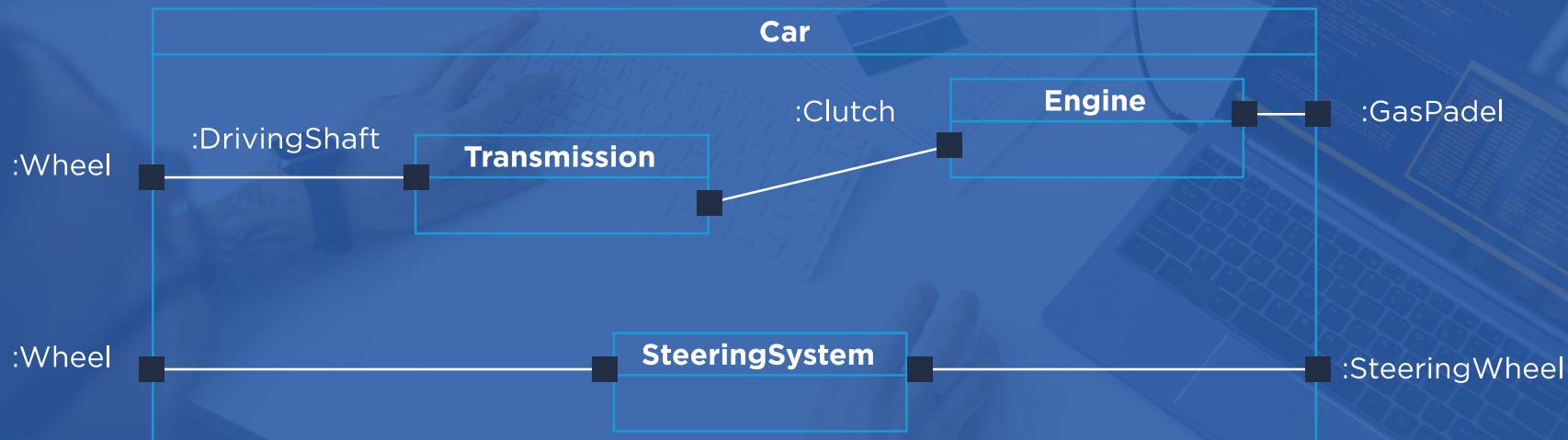
Realización. Demuestra que una interfaz, componente o artefacto realiza acciones no específicas con otro.

```
21944 // now have been here and is it's
28313 // partners
28627 // add shift to the path
30657 // partners
3854 // find this airline in the network
3855 // int pos = -1;
3856 // int index = 0;
3857 // while(index < network.size()){
3858 //     if(network.get(index).getnameO.equals(currentO){
3859 //         if(index == -1) {
3860 //             index = index + 1;
3861 //         }
3862 //         if(index == 0) {
3863 //             index = index + 1;
3864 //         }
3865 //         if(index == network.size() - 1) {
3866 //             index = index - 1;
3867 //         }
3868 //         if(index == -2) {
3869 //             index = index + 1;
3870 //         }
3871 //         if(index == -3) {
3872 //             index = index + 1;
3873 //         }
3874 //         if(index == -4) {
3875 //             index = index + 1;
3876 //         }
3877 //         if(index == -5) {
3878 //             index = index + 1;
3879 //         }
3880 //         if(index == -6) {
3881 //             index = index + 1;
3882 //         }
3883 //         if(index == -7) {
3884 //             index = index + 1;
3885 //         }
3886 //         if(index == -8) {
3887 //             index = index + 1;
3888 //         }
3889 //         if(index == -9) {
3890 //             index = index + 1;
3891 //         }
3892 //         if(index == -10) {
3893 //             index = index + 1;
3894 //         }
3895 //         if(index == -11) {
3896 //             index = index + 1;
3897 //         }
3898 //         if(index == -12) {
3899 //             index = index + 1;
3900 //         }
3901 //         if(index == -13) {
3902 //             index = index + 1;
3903 //         }
3904 //         if(index == -14) {
3905 //             index = index + 1;
3906 //         }
3907 //         if(index == -15) {
3908 //             index = index + 1;
3909 //         }
3910 //         if(index == -16) {
3911 //             index = index + 1;
3912 //         }
3913 //         if(index == -17) {
3914 //             index = index + 1;
3915 //         }
3916 //         if(index == -18) {
3917 //             index = index + 1;
3918 //         }
3919 //         if(index == -19) {
3920 //             index = index + 1;
3921 //         }
3922 //         if(index == -20) {
3923 //             index = index + 1;
3924 //         }
3925 //         if(index == -21) {
3926 //             index = index + 1;
3927 //         }
3928 //         if(index == -22) {
3929 //             index = index + 1;
3930 //         }
3931 //         if(index == -23) {
3932 //             index = index + 1;
3933 //         }
3934 //         if(index == -24) {
3935 //             index = index + 1;
3936 //         }
3937 //         if(index == -25) {
3938 //             index = index + 1;
3939 //         }
3940 //         if(index == -26) {
3941 //             index = index + 1;
3942 //         }
3943 //         if(index == -27) {
3944 //             index = index + 1;
3945 //         }
3946 //         if(index == -28) {
3947 //             index = index + 1;
3948 //         }
3949 //         if(index == -29) {
3950 //             index = index + 1;
3951 //         }
3952 //         if(index == -30) {
3953 //             index = index + 1;
3954 //         }
3955 //         if(index == -31) {
3956 //             index = index + 1;
3957 //         }
3958 //         if(index == -32) {
3959 //             index = index + 1;
3960 //         }
3961 //         if(index == -33) {
3962 //             index = index + 1;
3963 //         }
3964 //         if(index == -34) {
3965 //             index = index + 1;
3966 //         }
3967 //         if(index == -35) {
3968 //             index = index + 1;
3969 //         }
3970 //         if(index == -36) {
3971 //             index = index + 1;
3972 //         }
3973 //         if(index == -37) {
3974 //             index = index + 1;
3975 //         }
3976 //         if(index == -38) {
3977 //             index = index + 1;
3978 //         }
3979 //         if(index == -39) {
3980 //             index = index + 1;
3981 //         }
3982 //         if(index == -40) {
3983 //             index = index + 1;
3984 //         }
3985 //         if(index == -41) {
3986 //             index = index + 1;
3987 //         }
3988 //         if(index == -42) {
3989 //             index = index + 1;
3990 //         }
3991 //         if(index == -43) {
3992 //             index = index + 1;
3993 //         }
3994 //         if(index == -44) {
3995 //             index = index + 1;
3996 //         }
3997 //         if(index == -45) {
3998 //             index = index + 1;
3999 //         }
3999 }
```

Diagrama Integrado de Estructura:

- Es utilizado rara vez por personas externas al campo de desarrollo de software.
- Es similar a un diagrama de clases, pero adopta un enfoque más profundo. Este describe la estructura interna de múltiples clases y muestra las interacciones entre ellas.

- Genera una vista superior destinada al entendimiento de desarrolladores.
- Muestra la estructura interna (incluyendo partes y conectores) de un clasificador estructurado o una colaboración.



Funcionamiento

Está compuesto por partes y conectores de un clasificador estructurado o una colaboración, los cuales son muy parecidos a los de colaboración, además de los de clase.

Se usa para expresar arquitecturas en tiempo de ejecución, patrones de uso, así como las relaciones de los elementos participantes. Estos pueden no estar reflejados por diagramas estáticos



Elementos:

Clase. Dentro del gráfico demuestra una clase que debería de tener el sistema.

Función. Aparece dentro y representa una función real de la clase.

Atributo. Las funciones podrán tener objetos o variables con las cuales se desarrollen.

clase

<<dataType>>
funcion

<<dataType>>
funcion

+variable
+objeto

DIAGRAMAS GLOBALES

Operación. Las funciones podrán tener actividades con las cuales se demuestra la ejecución.

Puerto. Muestra la salida o entrada lógica de las clases que deben llevar a una función en otra clase.

```
21944 // now have been here and it hasn't  
23333 // arrived yet, add current  
23372 // add shift to the path  
23632 // partners[i].add(current);  
23634 //  
23635 // find this airline in the network  
23640 //  
23641 //  
23642 //  
23643 //  
23644 //  
23645 //  
23646 //  
23647 //  
23648 //  
23649 //  
23650 //  
23651 //  
23652 //  
23653 //  
23654 //  
23655 //  
23656 //  
23657 //  
23658 //  
23659 //  
23660 //  
23661 //  
23662 //  
23663 //  
23664 //  
23665 //  
23666 //  
23667 //  
23668 //  
23669 //  
23670 //  
23671 //  
23672 //  
23673 //  
23674 //  
23675 //  
23676 //  
23677 //  
23678 //  
23679 //  
23680 //  
23681 //  
23682 //  
23683 //  
23684 //  
23685 //  
23686 //  
23687 //  
23688 //  
23689 //  
23690 //  
23691 //  
23692 //  
23693 //  
23694 //  
23695 //  
23696 //  
23697 //  
23698 //  
23699 //  
23700 //  
23701 //  
23702 //  
23703 //  
23704 //  
23705 //  
23706 //  
23707 //  
23708 //  
23709 //  
23710 //  
23711 //  
23712 //  
23713 //  
23714 //  
23715 //  
23716 //  
23717 //  
23718 //  
23719 //  
23720 //  
23721 //  
23722 //  
23723 //  
23724 //  
23725 //  
23726 //  
23727 //  
23728 //  
23729 //  
23730 //  
23731 //  
23732 //  
23733 //  
23734 //  
23735 //  
23736 //  
23737 //  
23738 //  
23739 //  
23740 //  
23741 //  
23742 //  
23743 //  
23744 //  
23745 //  
23746 //  
23747 //  
23748 //  
23749 //  
23750 //  
23751 //  
23752 //  
23753 //  
23754 //  
23755 //  
23756 //  
23757 //  
23758 //  
23759 //  
23760 //  
23761 //  
23762 //  
23763 //  
23764 //  
23765 //  
23766 //  
23767 //  
23768 //  
23769 //  
23770 //  
23771 //  
23772 //  
23773 //  
23774 //  
23775 //  
23776 //  
23777 //  
23778 //  
23779 //  
23780 //  
23781 //  
23782 //  
23783 //  
23784 //  
23785 //  
23786 //  
23787 //  
23788 //  
23789 //  
23790 //  
23791 //  
23792 //  
23793 //  
23794 //  
23795 //  
23796 //  
23797 //  
23798 //  
23799 //  
23800 //  
23801 //  
23802 //  
23803 //  
23804 //  
23805 //  
23806 //  
23807 //  
23808 //  
23809 //  
23810 //  
23811 //  
23812 //  
23813 //  
23814 //  
23815 //  
23816 //  
23817 //  
23818 //  
23819 //  
23820 //  
23821 //  
23822 //  
23823 //  
23824 //  
23825 //  
23826 //  
23827 //  
23828 //  
23829 //  
23830 //  
23831 //  
23832 //  
23833 //  
23834 //  
23835 //  
23836 //  
23837 //  
23838 //  
23839 //  
23840 //  
23841 //  
23842 //  
23843 //  
23844 //  
23845 //  
23846 //  
23847 //  
23848 //  
23849 //  
23850 //  
23851 //  
23852 //  
23853 //  
23854 //  
23855 //  
23856 //  
23857 //  
23858 //  
23859 //  
23860 //  
23861 //  
23862 //  
23863 //  
23864 //  
23865 //  
23866 //  
23867 //  
23868 //  
23869 //  
23870 //  
23871 //  
23872 //  
23873 //  
23874 //  
23875 //  
23876 //  
23877 //  
23878 //  
23879 //  
23880 //  
23881 //  
23882 //  
23883 //  
23884 //  
23885 //  
23886 //  
23887 //  
23888 //  
23889 //  
23890 //  
23891 //  
23892 //  
23893 //  
23894 //  
23895 //  
23896 //  
23897 //  
23898 //  
23899 //  
23900 //  
23901 //  
23902 //  
23903 //  
23904 //  
23905 //  
23906 //  
23907 //  
23908 //  
23909 //  
23910 //  
23911 //  
23912 //  
23913 //  
23914 //  
23915 //  
23916 //  
23917 //  
23918 //  
23919 //  
23920 //  
23921 //  
23922 //  
23923 //  
23924 //  
23925 //  
23926 //  
23927 //  
23928 //  
23929 //  
23930 //  
23931 //  
23932 //  
23933 //  
23934 //  
23935 //  
23936 //  
23937 //  
23938 //  
23939 //  
23940 //  
23941 //  
23942 //  
23943 //  
23944 //  
23945 //  
23946 //  
23947 //  
23948 //  
23949 //  
23950 //  
23951 //  
23952 //  
23953 //  
23954 //  
23955 //  
23956 //  
23957 //  
23958 //  
23959 //  
23960 //  
23961 //  
23962 //  
23963 //  
23964 //  
23965 //  
23966 //  
23967 //  
23968 //  
23969 //  
23970 //  
23971 //  
23972 //  
23973 //  
23974 //  
23975 //  
23976 //  
23977 //  
23978 //  
23979 //  
23980 //  
23981 //  
23982 //  
23983 //  
23984 //  
23985 //  
23986 //  
23987 //  
23988 //  
23989 //  
23990 //  
23991 //  
23992 //  
23993 //  
23994 //  
23995 //  
23996 //  
23997 //  
23998 //  
23999 //  
23900 //  
23901 //  
23902 //  
23903 //  
23904 //  
23905 //  
23906 //  
23907 //  
23908 //  
23909 //  
23910 //  
23911 //  
23912 //  
23913 //  
23914 //  
23915 //  
23916 //  
23917 //  
23918 //  
23919 //  
23920 //  
23921 //  
23922 //  
23923 //  
23924 //  
23925 //  
23926 //  
23927 //  
23928 //  
23929 //  
23930 //  
23931 //  
23932 //  
23933 //  
23934 //  
23935 //  
23936 //  
23937 //  
23938 //  
23939 //  
23940 //  
23941 //  
23942 //  
23943 //  
23944 //  
23945 //  
23946 //  
23947 //  
23948 //  
23949 //  
23950 //  
23951 //  
23952 //  
23953 //  
23954 //  
23955 //  
23956 //  
23957 //  
23958 //  
23959 //  
23960 //  
23961 //  
23962 //  
23963 //  
23964 //  
23965 //  
23966 //  
23967 //  
23968 //  
23969 //  
23970 //  
23971 //  
23972 //  
23973 //  
23974 //  
23975 //  
23976 //  
23977 //  
23978 //  
23979 //  
23980 //  
23981 //  
23982 //  
23983 //  
23984 //  
23985 //  
23986 //  
23987 //  
23988 //  
23989 //  
23990 //  
23991 //  
23992 //  
23993 //  
23994 //  
23995 //  
23996 //  
23997 //  
23998 //  
23999 //  
23900 //  
23901 //  
23902 //  
23903 //  
23904 //  
23905 //  
23906 //  
23907 //  
23908 //  
23909 //  
23910 //  
23911 //  
23912 //  
23913 //  
23914 //  
23915 //  
23916 //  
23917 //  
23918 //  
23919 //  
23920 //  
23921 //  
23922 //  
23923 //  
23924 //  
23925 //  
23926 //  
23927 //  
23928 //  
23929 //  
23930 //  
23931 //  
23932 //  
23933 //  
23934 //  
23935 //  
23936 //  
23937 //  
23938 //  
23939 //  
23940 //  
23941 //  
23942 //  
23943 //  
23944 //  
23945 //  
23946 //  
23947 //  
23948 //  
23949 //  
23950 //  
23951 //  
23952 //  
23953 //  
23954 //  
23955 //  
23956 //  
23957 //  
23958 //  
23959 //  
23960 //  
23961 //  
23962 //  
23963 //  
23964 //  
23965 //  
23966 //  
23967 //  
23968 //  
23969 //  
23970 //  
23971 //  
23972 //  
23973 //  
23974 //  
23975 //  
23976 //  
23977 //  
23978 //  
23979 //  
23980 //  
23981 //  
23982 //  
23983 //  
23984 //  
23985 //  
23986 //  
23987 //  
23988 //  
23989 //  
23990 //  
23991 //  
23992 //  
23993 //  
23994 //  
23995 //  
23996 //  
23997 //  
23998 //  
23999 //
```

<<dataType>>
función

+operacion()

Port1

DIAGRAMAS GLOBALES

Realización. Demuestra la comunicación de una o varias funciones con otras clases o con otras funciones; de igual modo con los puertos.

Conector. Demuestra la conexión entre dos puertos de diferentes clases, sin aclarar de qué forma se transmiten los datos.



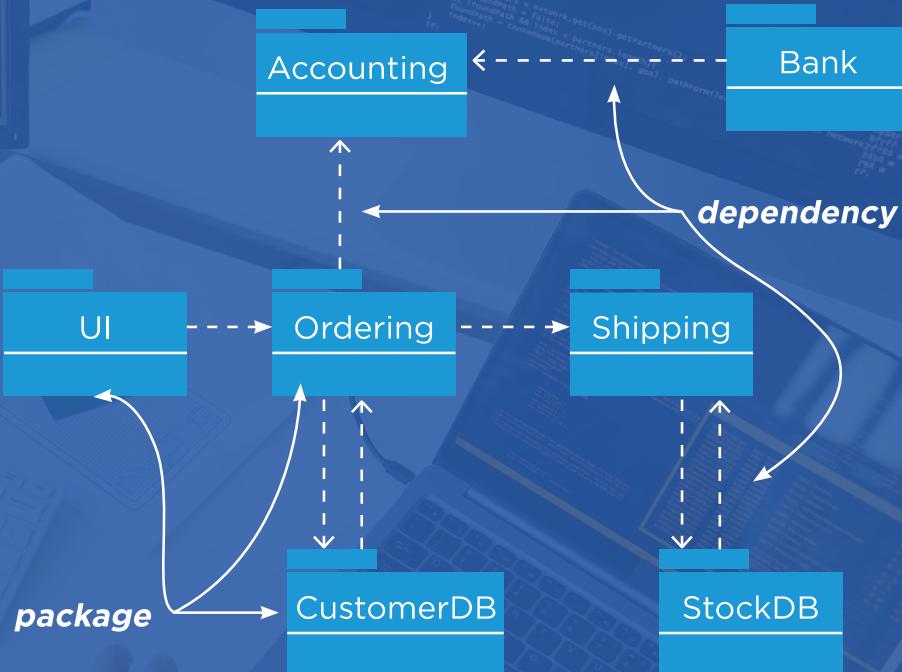


Diagrama de Paquetes:

- Este se utiliza para representar las dependencias entre los paquetes que componen un modelo.
- Su objetivo principal es mostrar la relación entre los diversos componentes grandes que forman un sistema complejo.

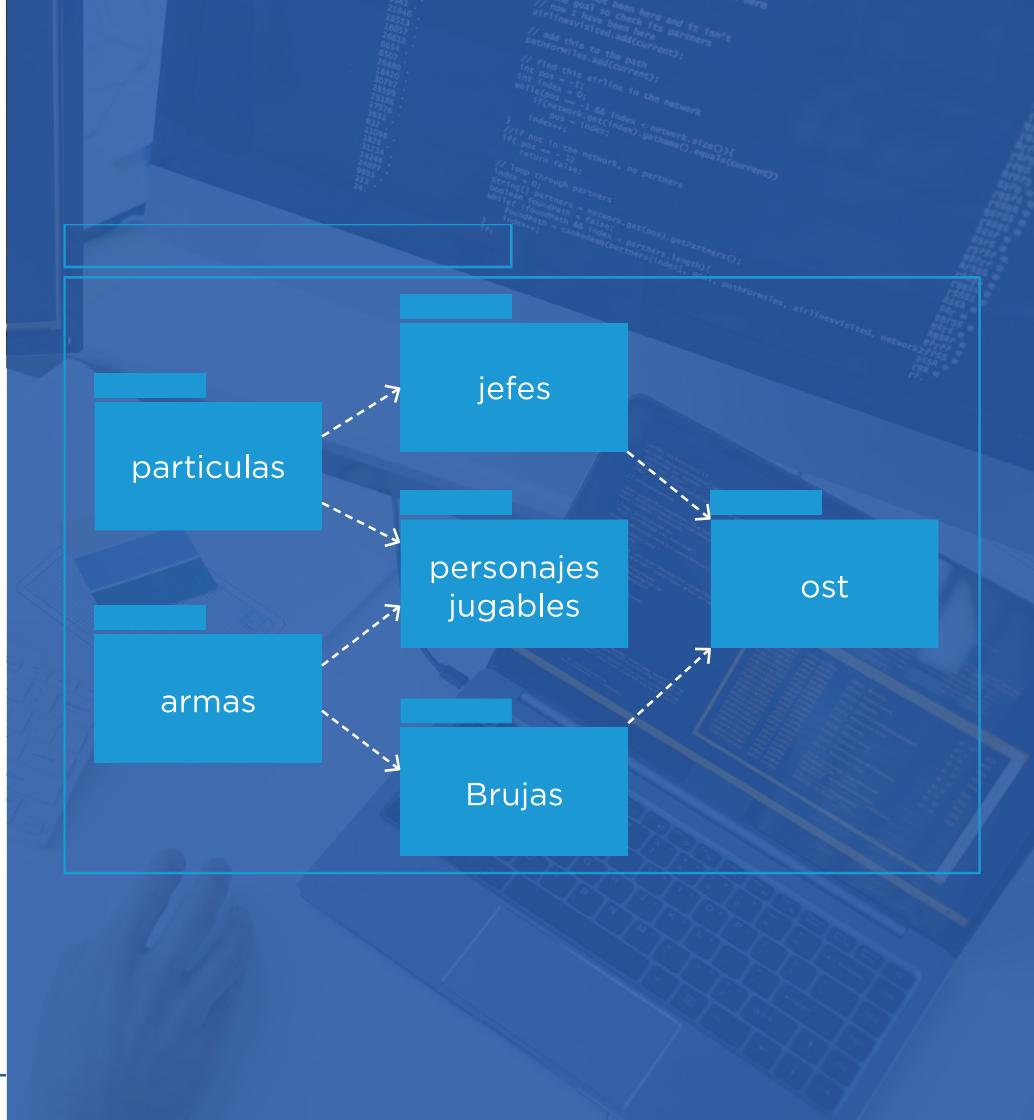
DIAGRAMAS GLOBALES

- Muestra la descomposición del propio modelo en unidades organizativas (paquetes) y sus dependencias.
- Simplifica los diagramas de clases complejos, permitiendo el agrupamiento de los clasificadores en paquetes.



DIAGRAMAS GLOBALES

- Los paquetes constituyen un mecanismo de agrupación para organizar elementos UML.
- Proporciona un espacio de nombres a los elementos agrupados.



Project

- Los paquetes se organizan jerárquicamente. De estos, el paquete raíz es el que contiene todo el sistema.
- Un paquete se representa como un rectángulo grande con un rectángulo pequeño sobre su esquina superior izquierda.

Elementos:

Carreta. Este elemento demuestra una carpeta, en la cual se albergarán objetos y clases.

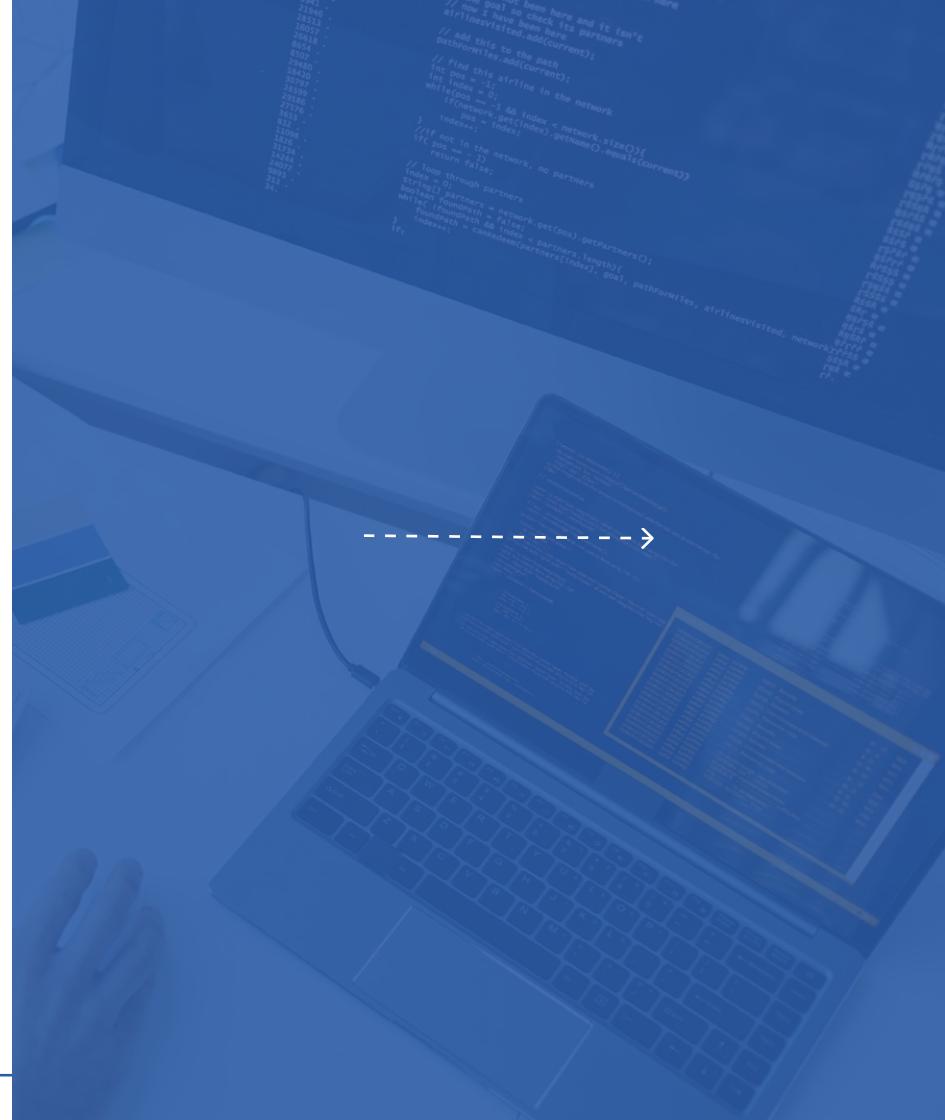


DIAGRAMAS GLOBALES

Contiene. Esta relata que la carpeta contiene dentro de ella a aquella que se encuentra del lado abierto de la línea.



Dependencia. Esta muestra que una carpeta depende de los objetos o clases a la quien se le señala.



ACTIVIDAD 2 - Unidad 3.

Diagramas de clases y objetos

Te invitamos a realizar la siguiente actividad:

Presiona el botón para descargar la actividad:

Presiona el botón para entregar la actividad:



LECTURAS PARA REFORZAR LA UNIDAD

- García, F. Moreno, M. García, A. (2018). *Ingeniería de Software I. Tema 8. UML-Unified Modeling Language*. Salamanca, España. Universidad de Salamanca.
https://repositorio.grial.eu/bitstream/grial/1949/1/IS_I%20Tema%208%20-%20UML.pdf
- Ramírez, C. (2020). *Diagramas Esenciales del Lenguaje Unificado de Modelados para los Requisitos Ágiles en el Desarrollo de Software*. Universidad Nacional Abierta y a Distancia. Escuela de Ciencias Básicas, Tecnología e Ingeniería Tecnología en Desarrollo de Software Bogotá D.C.
<https://repository.unad.edu.co/bitstream/handle/10596/38052/cvramirezc.pdf?sequence=1&isAllowed=y>



CONCLUSIÓN



Los diagramas de modelado estructurado son diagramas UML que describen las características estáticas de un sistema. Estos representan la disposición generalizada de un sistema para el modelado de estos aspectos. Entre ellos, se presentan los diagramas de clases y los de objetos. También, son útiles los diagramas de estructura compuesta de clases.

El modelado es vital en todo tipo de proyectos, pero cobra especial importancia a medida que el proyecto crece en tamaño. Al utilizar diagramas UML se consigue visualizar y verificar los diseños antes que la implementación. El modelado por diagramas particulares brinda el diagrama de clases, el cual es uno de los más comunes para el proceso de desarrollo.



¡FELICIDADES!



Acabas de concluir la tercera unidad de tu curso *Lenguaje Unificado de Modelado*. Te invitamos a finalizar este esfuerzo realizando el examen parcial correspondiente. Para ello, debes regresar a la pantalla principal y dar clic en *Presentar examen*.

LENGUAJE UNIFICADO DE MODELADO



UNIDAD 4

HERRAMIENTAS PARA MODELADO DE DIAGRAMAS UML





TEMARIO

U4

4.1



Herramientas Libres

4.2



Herramientas Privativas

INTRODUCCIÓN

En esta cuarta unidad conocerás las herramientas digitales, tanto privativas como libres, que son útiles para el modelado de diagramas UML.



COMPETENCIAS A DESARROLLAR



1.

El alumno será capaz de conocer, seleccionar y realizar los diagramas UML empleando herramientas digitales libres para el modelado.

2.

El alumno será capaz de conocer, seleccionar y realizar los diagramas UML empleando herramientas digitales privativas para el modelado.

HERRAMIENTAS LIBRES



Las herramientas CASE son un conjunto de aplicaciones informáticas. Estas son utilizadas para automatizar actividades del ciclo de vida de desarrollo de sistemas (SDLC).

HERRAMIENTAS LIBRES

Hay un gran número de herramientas CASE, disponibles para simplificar varias etapas en el desarrollo del ciclo vital del software. Existen, por ejemplo, herramientas de análisis, diseño de herramientas, gestión de proyectos de herramientas, proyectos de gestión de herramientas de bases de datos, gestión de herramientas de bases de datos. Por otra parte, deben nombrarse también algunas herramientas de documentación.



HERRAMIENTAS LIBRES

El uso de herramientas CASE acelera el desarrollo del proyecto en aras de producir los resultados deseados. Además, ayuda a encontrar imperfecciones antes de proseguir con la siguiente etapa del desarrollo de software.



No existe una clasificación estándar de herramientas CASE.

De acuerdo a su funcionalidad, las herramientas pueden ser clasificadas tal como se muestra en la siguiente tabla:

Nombre	Descripción
Herramientas integradas I - CASE	<i>Integrated CASE o CASE integrado.</i> Abarcan todo el ciclo de vida del desarrollo de software. También son conocidas como CASE <i>workbench</i> .
Herramientas de alto nivel U - CASE	<i>Upper CASE o CASE superior.</i> Son herramientas orientadas a automatizar y dar soporte a las actividades que se desarrollan durante el análisis y el diseño.
Herramientas de bajo nivel L - CASE	<i>Lower CASE o CASE inferior.</i> Son herramientas que están orientadas a automatizar las fases de construcción e implantación.

ArgoUML

Es un editor UML que tiene compatibilidad con el estándar UML1.4. Permite la exportación a varios formatos gráficos, y tiene la disponibilidad de perfiles para varios lenguajes de programación.

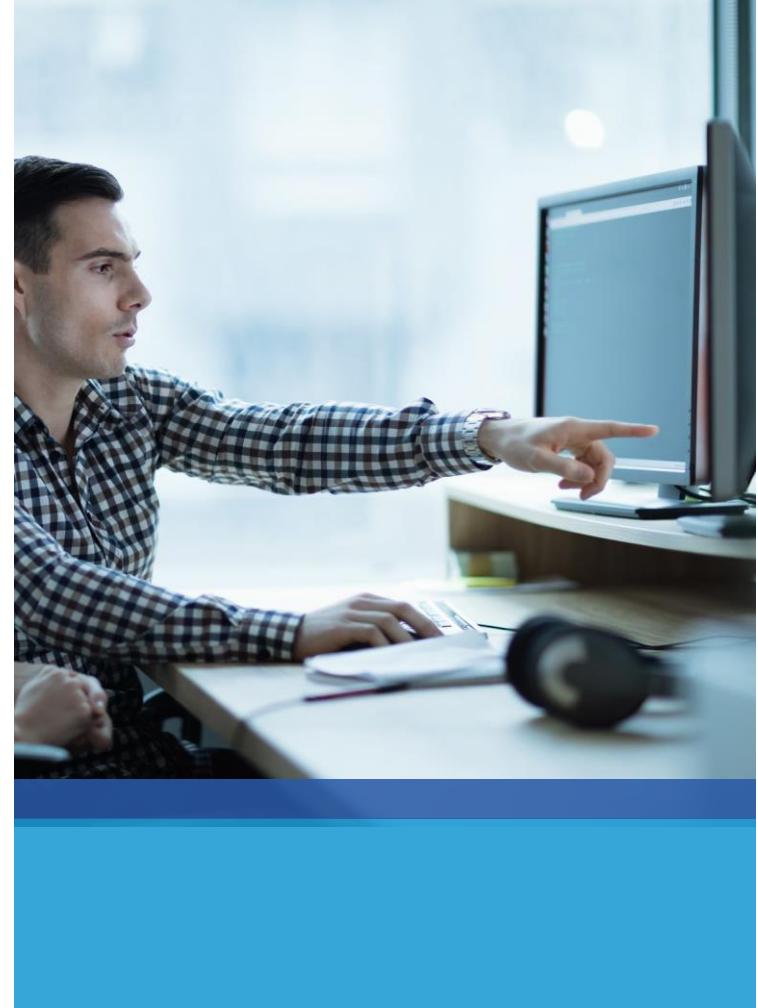
Acceso:

<https://www.ecured.cu/ArgoUML>



Características:

- Puede generar casi todos los tipos de diagramas UML conocidos.
- Cuenta con una interfaz simple y atractiva para el usuario.





HERRAMIENTAS LIBRES

- Permite la exportación de diagramas a diferentes formatos.
- Permite generación de código.
- Cuenta con soporte para bases de datos.
- Soporte cognitivo:
 - Críticas de diseño creado, listas de cosas por hacer (*To Do Lists*), correcciones automáticas, entre otros.
 - Comprensión y solución del problema.

HERRAMIENTAS LIBRES



- Su mayor limitante es que, al guardar, solo lo realiza con las extensiones *.zargo*, *.uml* y *.xmi*.

HERRAMIENTAS LIBRES

GitMind

Es una excelente plataforma en línea donde puedes crear diferentes diagramas. Se incluye UML de forma gratuita.

Acceso:

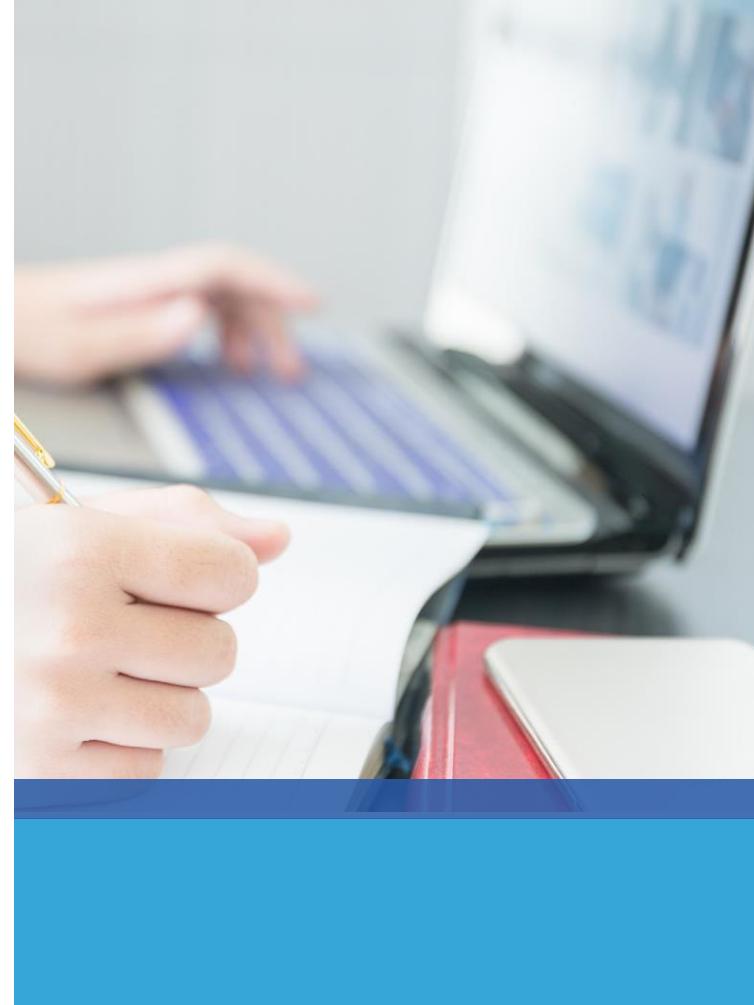
<https://gitmind.com/>

```
21546 // now have been here and if isn't  
25813 // visited.add(current)  
15657 // add this to the path  
26027 partners.add(current)  
8534 // find this airline in the network  
8535 int pos = 0;  
25940 while(pos < 1 && index < network.size()){
15820 pos = partners.get(index).partnerName.equals(current)  
30292     )  
28109     index++;  
27586 }  
27523 // if not in the network, no partners  
8112 return false;  
21209 // long through partners  
31714 // 2nd partners = network.get(pos).getPartners();  
24954 // boister. (boister = false);  
24901 // (boister = !boister); (partners.length() - 1), partners.get(index).partnerName,  
24 // atTimesVisited, network.size()  
    }  
}
```



Características:

- Brinda plantillas personalizables para la creación de los diagramas UML.
- Permite la creación de diagramas de manera rápida y sencilla.
- Cuenta con una interfaz de usuario amigable, que muestra funciones de edición para una experiencia sin problemas.



Gliffy

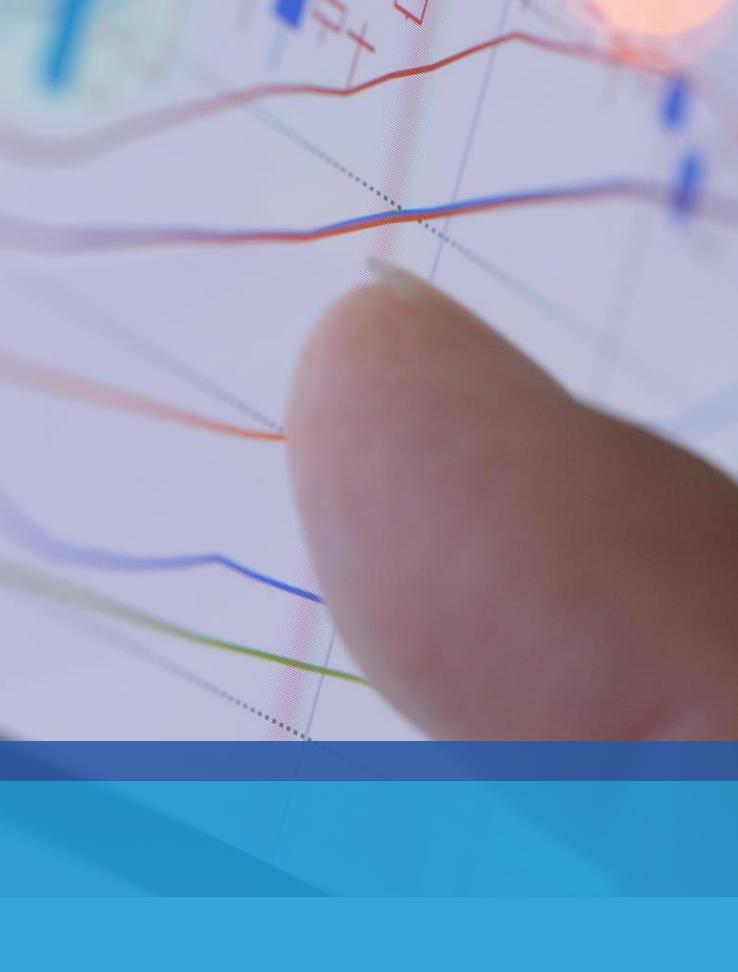
Facilita la creación, el intercambio y la colaboración en una amplia gama de diagramas. Funciona tanto *online* como *offline*.

Permite crear diagramas y organigramas profesionales de manera fácil y rápida.

Acceso:

<https://www.gliffy.com/blog>





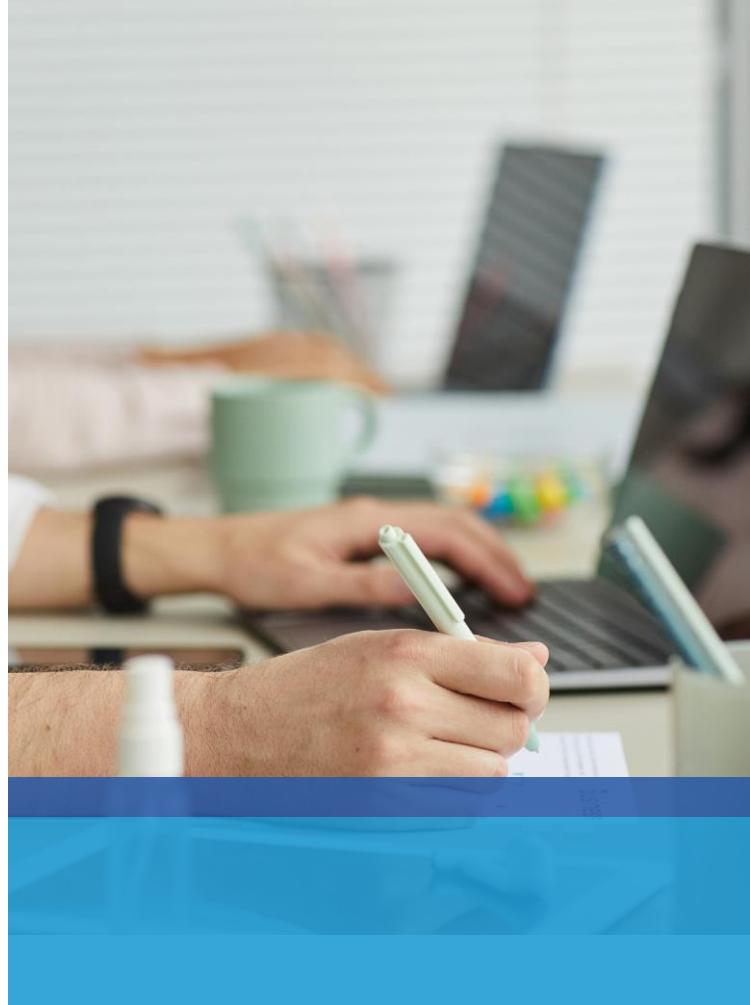
HERRAMIENTAS LIBRES

Características:

- Tiene énfasis en los aspectos de control colaborativo y de revisión.
- Admite todos los diagramas UML junto con una variedad de otros tipos de diagramas, incluido un soporte sólido para los modelos de procesos.
- Complementos para integrarse con otras herramientas.

Principales ventajas:

- Organizar información.
- Permite trabajar colaborativamente.
- Permite compartir información.
- Cuenta con un buen diseño de los gráficos.
- Es gratuita.
- Los cambios realizados son visibles para todos los usuarios con los que se comparten los proyectos.
- Presenta un gran número de herramientas.



Principales desventajas:

- Para su uso es necesaria una conexión a Internet.
- Los demás usuarios no reciben notificaciones sobre otros usuarios que están trabajando en el mismo diagrama.

HERRAMIENTAS LIBRES

MagicDraw

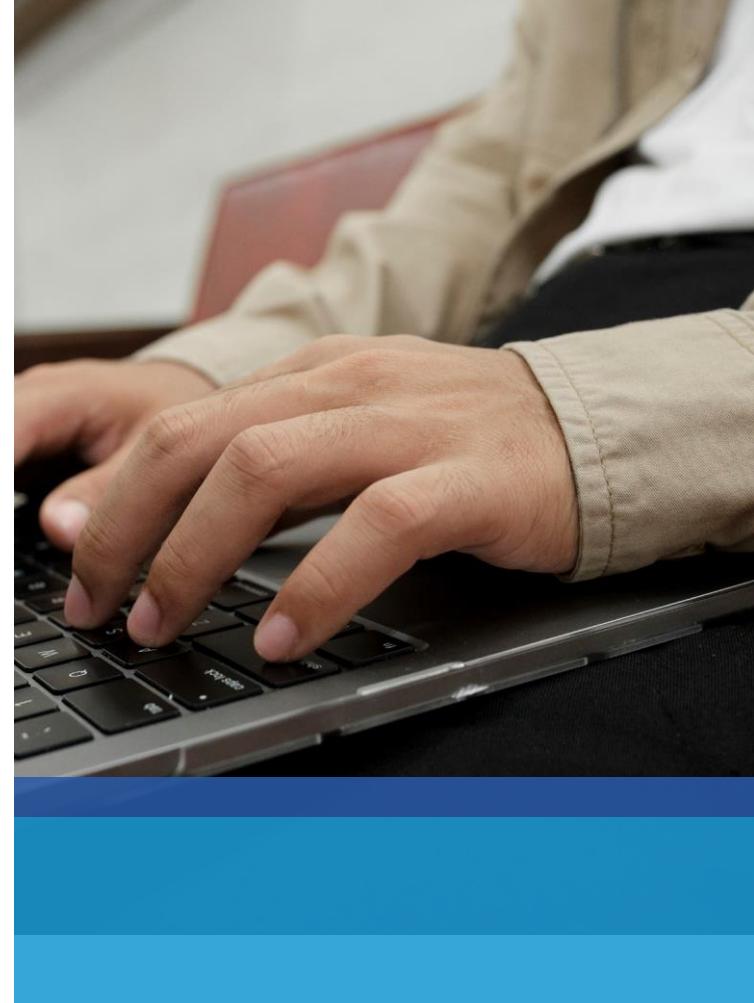
Diseñada para trabajos individuales y en equipo. Está creada para un sector específico de profesionales, como ingenieros, arquitectos, analistas, programadores y redactores de documentación.

Su descarga es gratuita y es compatible con cualquier sistema operativo.



Características:

- Estructura versátil y dinámica que facilita el análisis, el diseño de sistemas y las bases de datos orientadas a los objetos.
- También, posee las habilidades para el modelado de esquemas de base de datos, la generación de DDL y las instalaciones de ingeniería inversa.
- Cuenta con un motor de ejecución increíble para los modelos UML.





HERRAMIENTAS LIBRES

- Tiene una interfaz intuitiva.
- Crea diagramas rápidamente.
- Genera automáticamente informes.
- Su descarga es gratuita.
- Su uso es *offline*.
- Genera códigos para Java, EJB, C #, C ++, CORBA IDL, DDL, WSDL y esquema XML

HERRAMIENTAS LIBRES

- Facilidad y rapidez para el cambio del dominio del modelado.
- Cuenta con un generador automático de informes.
- Permite el desarrollo colaborativo.
- Disponible para un gran número de plataformas y sistemas operativos.



UMLet

Se trata de un herramienta muy práctica para el dibujo de diagramas. Además, es gratuita, intuitiva y muy sencilla de usar. Recomendada para iniciar el uso de UML. Puedes generar secuencias, diagramas de texto sin formato y exportarlos en diversos formatos como PDF, JPG, PNG, SVG y portapapeles.

Acceso:

<https://www.umlet.com/>





HERRAMIENTAS LIBRES

Características:

- Los elementos de UMLet se manejan mediante cuadros de textos que pueden ser modificados adaptando las necesidades acorde al modelo.
- Los diagramas varían según su motivo de uso. Además, cuenta con ejemplos para ayudar a generar ideas de diseños.

HERRAMIENTAS LIBRES

- Herramienta de descarga gratuita.
- Posibilidad de realizar diagramas rápidos.
- Interfaz sencilla.
- Compatible con cualquier software.
- Herramientas de extensión accesible al usuario
- Se puede trabajar *online* u *offline*.
- Se puede ejecutar de forma independiente o como complemento de Eclipse.



HERRAMIENTAS PRIVATIVAS

Edraw UML Diagram

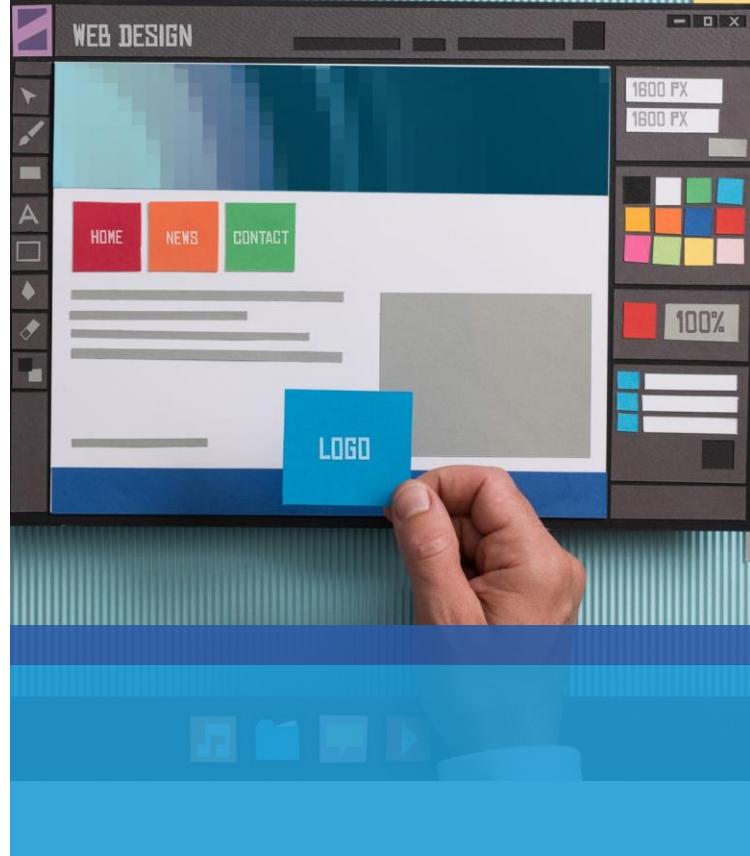
Permite la creación de diagramas todo en uno, lo que beneficia a las empresas para diseñar distintos diagramas. Diseñada con el objetivo de facilitar que las empresas creen diagramas y gráficos profesionales con su marca. Esta solución proporciona herramientas de personalización y ofrece un control total sobre diseños, colores, texto y otros detalles.

Acceso:

<https://www.edrawsoft.com/es/edraw-max/>



HERRAMIENTAS PRIVATIVAS



Características:

- Presenta una apariencia similar a la del programa Microsoft Office.
- Diversos tipos de diagramas.
- Permite crear diagramas claros y completos con los ejemplos prediseñados.
- Robusta compatibilidad de archivos.
- Comparte y colabora en tu trabajo en cualquier formato de archivo, y en cualquier canal que desees.

HERRAMIENTAS PRIVATIVAS

Visio

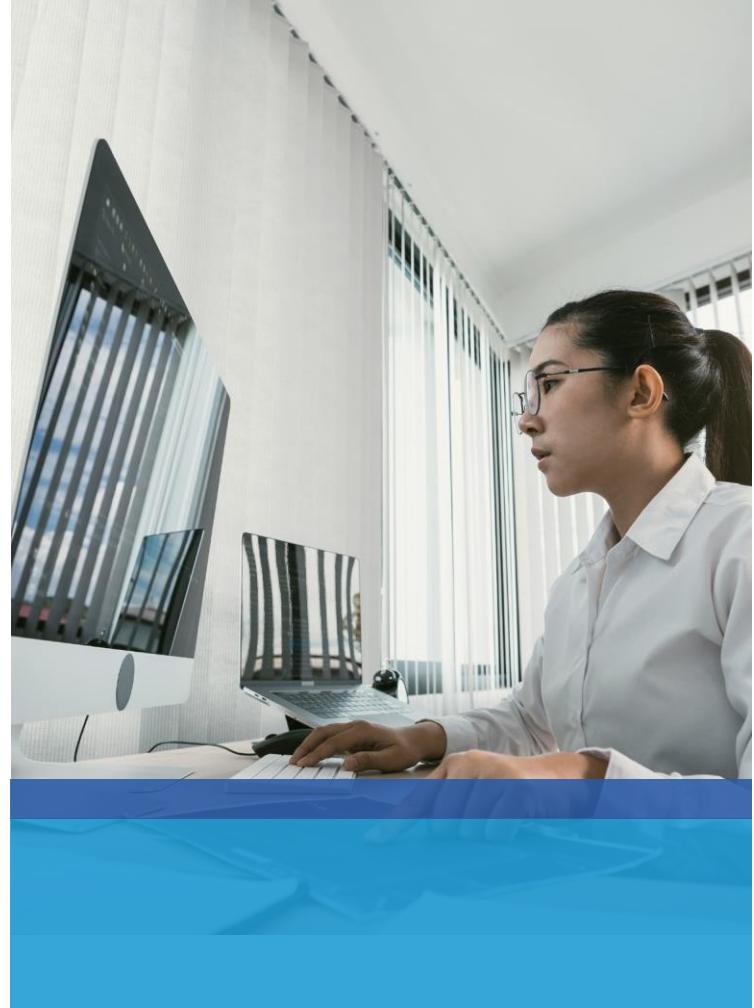
Sirve básicamente para diseñar diagramas de flujo y de procesos, así como mapas conceptuales, líneas de tiempo y organigramas con gran facilidad. Incluye, también, la opción de crear diagramas UML a partir de bases de datos.

Microsoft Visio puede ser usado para crear diagramas simples o complicados.



Características:

- Con temas fáciles de aplicar para la creación de diagramas modernos de forma rápida.
- Importación de archivos DWG a Visio.
- Galerías de símbolos.
- Actualización de formas.
- Soporte para varias reglas de negocio y estándares de proceso
- Los diagramas de comunicación, de componentes y de despliegue UML son realizables en Visio 2019.



Visio 2016

- Integración a Office 365.
- Función *Tell me*.
- Mensajería instantánea.
- Seguridad de archivos mejorada.
- Mejor compatibilidad con Excel.

Visio 2019

- Diagramas de Modelo de Base.
- *Wireframe*.
- Mejor integración con AutoCAD.
- Lenguaje de Modelado Unificado.

Ventajas:

- Amplia variedad de formas para diagramas.
- Incluye variaciones por edición (*Standard, Professional, Pro*) con variación en precios y opciones.
- Ofrece integración en la nube.
- Pueden usarse durante mucho tiempo, mientras sean compatibles con el dispositivo.
- Al ser el software líder en el mercado, muchos profesionales ya están familiarizados con Visio.



Desventajas:

- El software es costoso.
- Existen quejas del servicio al usuario de Microsoft.
- No es compatible con otros sistemas operativos.
- Es especialmente costoso al compararlo con las alternativas que ofrece el mercado.

HERRAMIENTAS PRIVATIVAS

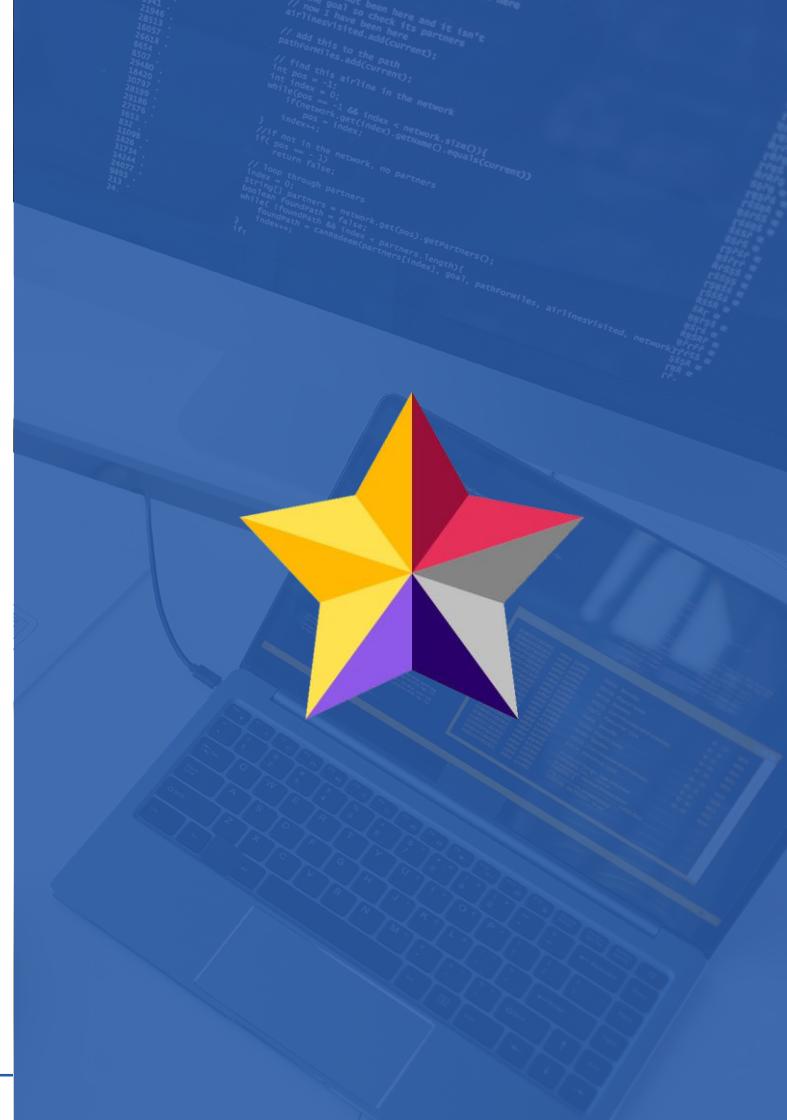
StarUML 3

Es una herramienta de fácil uso que ayuda a generar diagramas compatibles con la suite de ofimática de Microsoft Office. Permite el código, el cual es compatible con C++ y Java.

Se puede empezar a dibujar manualmente o hacer uso de plantillas que contienen archivos de instalación de modelamiento UML.

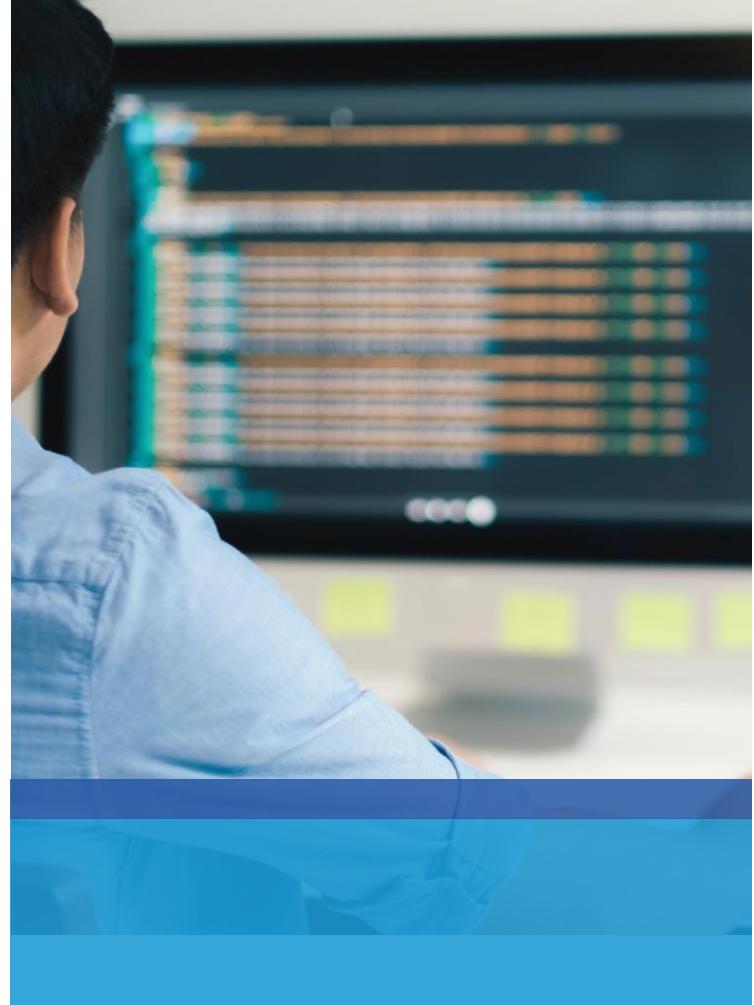
Acceso:

<https://staruml.io/>



Características:

- Fácil de usar, intuitiva, rápida.
- Podrás trabajar cómodamente obteniendo grandes resultados.
- Soporte de retina *display*, textos e iconos nítidos. Así como la ventaja de exportar imágenes en JPG o PNG.
- Permite instalar extensiones de programas de terceros de código abierto para ampliar las posibilidades de trabajo.



HERRAMIENTAS PRIVATIVAS

- Descarga gratuita.
- Compatible con Windows, Mac y Linux.
- Publicar documentos HTML.
- Posibilidad de exportar en PDF.
- Soporte para crear diagramas de relación de entidad (ERD).
- Permite los diagramas de flujo de datos (DFD).

HERRAMIENTAS PRIVATIVAS

Diagrama ConceptDraw 12

Ofrece un conjunto profesional de herramientas de dibujo, plantillas y bibliotecas de objetos. Es una alternativa viable para personas y organizaciones que buscan un software empresarial profesional.

Acceso:

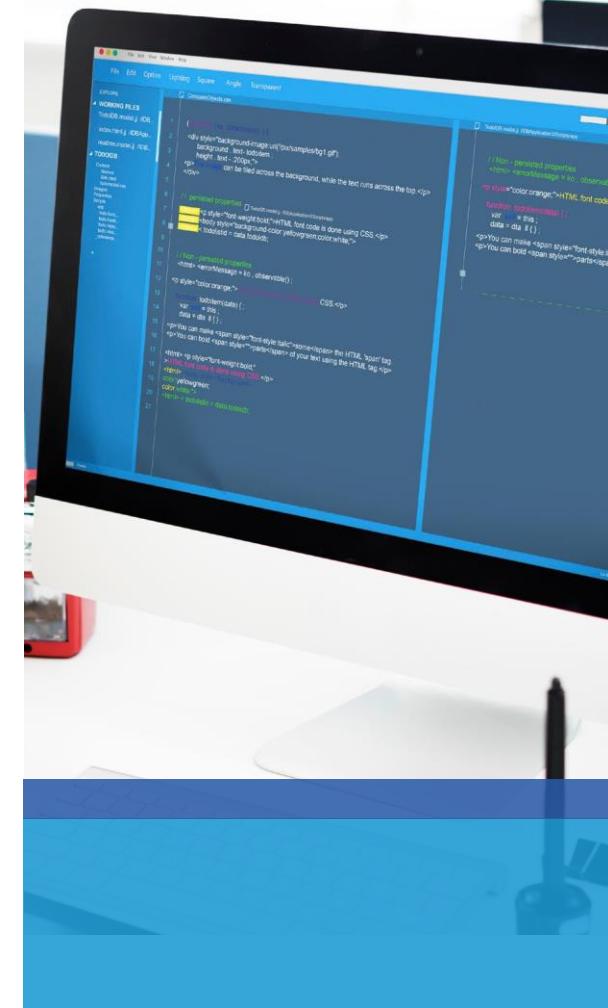
<https://www.conceptdraw.com/>



HERRAMIENTAS PRIVATIVAS

Características:

- Presentaciones con pantallas dinámicas de datos vinculados.
- Permite la importación y exportación de archivos en formato MS Visio.
- Compatible con los formatos de archivo más populares.
- Herramientas de dibujo potentes e integrales.
- Plantillas personalizadas.
- Paneles laterales mejorados para organizar y formatear cualquier tabla compleja.
- Tecnología de diagramas de flujo rápida.



LECTURAS PARA REFORZAR LA UNIDAD

- García, T. S. R. (2021). *UML Introducción al UML, modelando con UML, utilidad del UML, conceptos de USE CASE, objetos, clases y atributos, operaciones, Aplicaciones.* Repositorio UNE.
<https://repositorio.une.edu.pe/handle/UNE/5005>

- Rivero, M. Pérez, P. (2017). *Análisis de Herramientas de Modelado de Procesos de Negocio.* Dep. de Organización Industrial y Gestión de Empresas I Escuela Técnica Superior de Ingeniería Universidad de Sevilla.
http://bibing.us.es/proyectos/abreproj/91303/fichero/TFGMariarRiveroPinoGIOIVO.1_paz.pdf

CONCLUSIÓN



Al iniciar un proyecto es imprescindible determinar cuál de las herramientas disponibles en el mercado suple las necesidades que este plantea. Esta unidad presentó las características de algunas herramientas de modelado UML, teniendo en cuenta los aspectos y ventajas que ofrece cada una de ellas.

Tú deberás seleccionar la herramienta adecuada conforme a los requerimientos y necesidades particulares de tu proyecto y experiencia. A medida que tu nivel práctico y de conocimiento avance, podrás elegir una herramienta con mayor nivel de dificultad. Es importante entender que hay herramientas muy capaces, pero esto no necesariamente significa que serán las que debas usar. Se recomienda realizar el análisis y diseño con una herramienta que ofrezca un entorno amigable, interactivo e intuitivo, de manera que el proceso no sea demasiado complejo.



¡FELICIDADES!



Acabas de concluir la cuarta unidad de tu curso *Lenguaje Unificado de Modelado*. Te invitamos a finalizar este esfuerzo realizando el examen parcial correspondiente. Para ello, debes regresar a la pantalla principal y dar clic en *Presentar examen*.

A photograph showing four people in a professional setting, likely an office, working together on a project. One person in the foreground is pointing at a chart on a table, while others look on. The table is covered with various documents, charts, and a laptop. The scene is overlaid with a large blue rectangular area containing the text.

PROYECTO FINAL

PROYECTO FINAL.

Te invitamos a realizar la siguiente actividad:

Presiona el botón para descargar la actividad:

Presiona el botón para entregar la actividad:

