UT3 - Diseño y UML

Type	Category 1
Status	Archived
URL	
Updated	@May 11, 2023 10:13 AM
Created	@May 2, 2023 6:42 PM

Action Items

☐

Notes

Ways of using the UML

UML as sketch:

Forward engineering: dibuja un diagrama UML antes de escribir codigo.

Reverse engineering: crea un diagrama UML de código existente para ayudar a entenderlo mejor. Se utilizan los sketches para explicar como funciona alguna parte del sistema.

UML as blueprint:

UML as blueprint es sobre completitud.

Los blueprint son desarrollados por diseñadores cuyo trabajo es crear un diseño detallado para que un programador codifique.

Reverse engineering: blueprints intentan transmitir información detallada sobre el codigo. Pueden mostrar todos los detalles sobre una clase.

Herramientas de reverse-engineering leeb codigo funte y lo interpretan, generando diagrams.

UML as programming language:

desarrolladores dibujan diagramas UML que compilan directo a codigo ejecutable, y el UML se convierte en el codigo fuente.

development process

WaterFall process: descompone un proyecto basandose en actividades.

Iterative process: descompone un proyecto por subconjuntos de funcionalidad.

Diseño

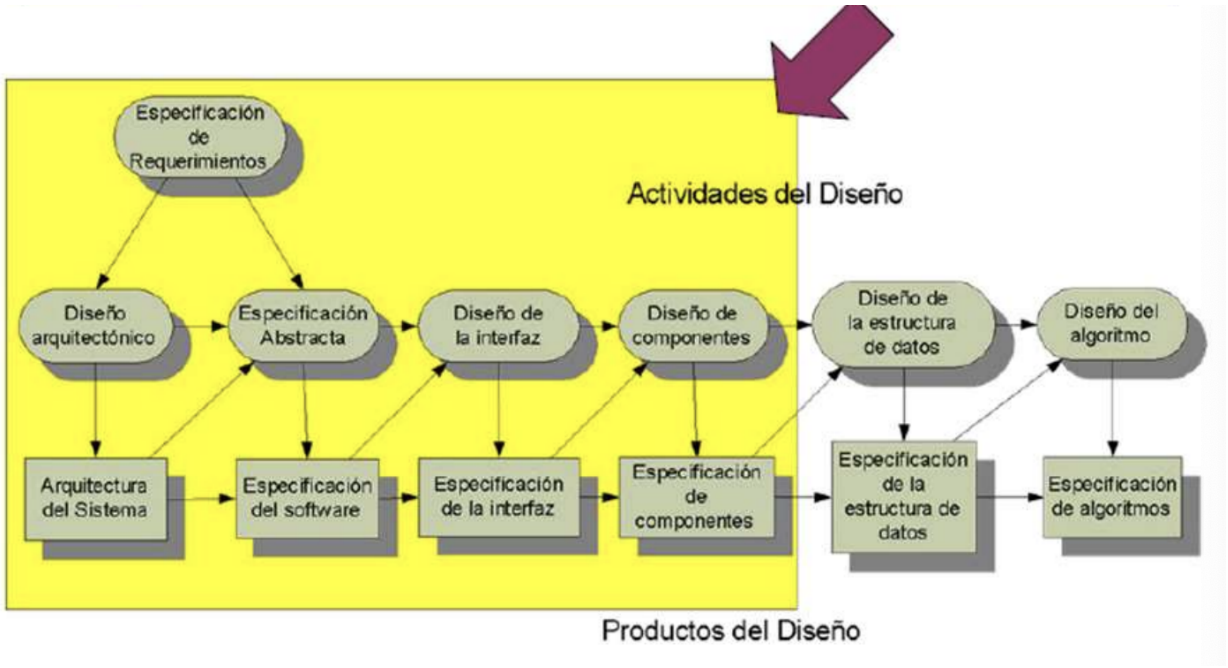
- El diseño es tanto el proceso de definir:
  - Arquitectura
  - Componentes
  - Interfases
  - Otras caracteristicas del sistema/componente
- Y el resultado de ese proceso.

Definiciónes:

- El sistema:

    - Es una **entidad lógica**, que tiene un conjunto de responsabilidades y objetivos, y consiste de hardware, software o ambos.
- Subsistema:
    - Es un sistema que es parte de un sistema mayor.
    - Tiene una interface bien definida.
- Componente:
    - **Cualquier pieza de software o hardware** que tenga definido un rol claro.
    - Un componente puede ser aislado, permitiendo que se lo reemplace con un componente diferente que tenga una funcionalidad equivalente.
    - Muchos componente están diseñasdos para ser reutilizables.
    - Recíprocamente, otros realizan funciones con unu propósito especial.
- Módulo:
    - Un componente que es definido a nivel de lenguaje de programación.
    - Package/Namespace

El Proceso de diseño del Software



Diseño arquitectonico:

- Se define la arquitectura (generalmente se utiliza un patrón de arquitectura capas, mvc, monolítica, microservicios, SOA, etc).

Especificación abstracta:

Se especifican los subsistemas. Cada subsistema realiza un servicio importante.

- Contiene objetos altamente acoplados.
- Relativamente independiente de otro subsistemas.
- Generalmente se descompone de módulos aunque se puede descomponer en subsistemas mas pequeños.

Diseño de la interfaz:

Se describen las interfaces de los subsistemas.

Diseño de los componentes:

Se descompone cada subsistema en componenetes:

ejemplo ——

Principios del Diseño:

- **Dividir y conquistar:** tratar algo grande de entrada es normalmente mucho nas dificil que si se hace cosas mas pequeñas.
- **Incrementar la cohesión:** Todo lo que sea posible. Un susbsistema o módulo tiene alto grado de cohesión si mantiene juntas las cosas que están relacionadas y afuera las restantes.
- **Reducir el acoplamiento:** Todo lo que sea posible. El acoplamiento ocurre cuando existe interdependencia entre un módulo y otro.
- Mantener el **nivel de abstracción tan alto** como sea posible. Asegúrese que el diseño oculte o difiera consideraciones de dettalle, reduciendo por tanto la complejidad.
- **Incrementar la Reusabilidad:** donde sea posible. Diseñe los variado aspectos de su sistema de tal forma que pueda ser usado nuevamente en otros contextos. Reutilización de los diseños y códigos existentes cuando sea posible.
- **Diseñe para ser flexible:** Activamente anticipe los cambios que el diseño pueda tener en el futuro, y prepararse para ello.
- **Anticipe la obsolescencia:** Planifique los cambios en la tecnología o el entorno de tal manera que el software continúe ejecutándose o sea fácilmente cambiado. Por ejemplo diseñar panatallas responisve.

Cohesión

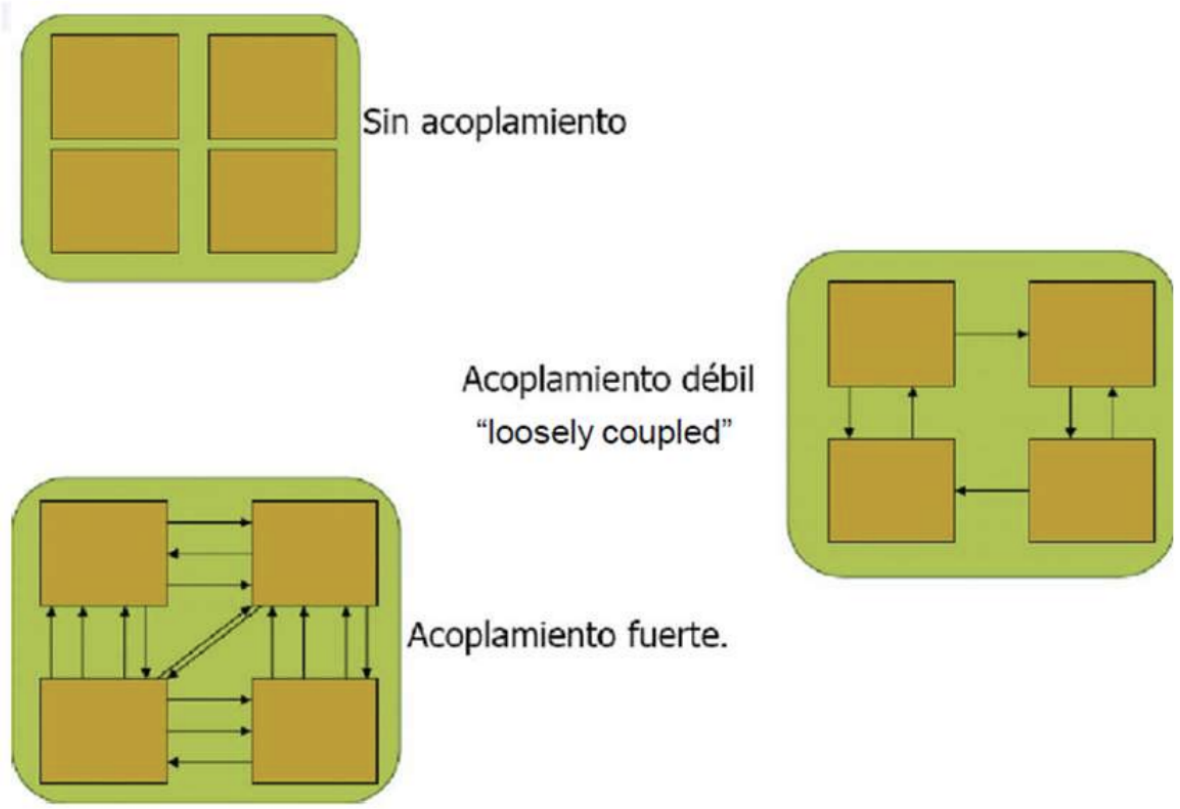
- Un subsistema o módulo tiene alto grado de cohesión si mantiene “unidas” cosas que están relacionadas entre ellas y mantien fuer el resto:
  - Un módulo cohesivo lleva a cabo una sola tarea dentro de un procedimiento de software.
- Objetivo:
  - Diseñar servicios robustos y altamente cohesionados cuyos elementos estén fuerte y genuinamente relacionados entre sí.

- Ventajas:
    - Favorece la comprensión y el cambio de los sistemas.
- El objetivo es entonces maximizar la cohesión.

Acomplamiento:

“Acoplamiento es la medida de la fortaleza de la asociación establecida por una conexión entre módulo dentro de una estructura de software”

- Depende de la complejidad de interconexión entre los módulos, el punto donde se realiza una entrada o referencia a un módulo y los datos que se padan a través de la interfaz.
- Es una medida de interconexión entre módulos dentro de una estrucutra del software.
- Un bajo acomplamiento indica un sistema bien dividios y puede conseguirse mediante la eliminación o reducción de relaciones innecesarias.
- Se produce una situación de acomplamineto cuando un elemento de diseño depende de alguna forma de otro elemento del diseño.
    - El objetivo es conseguir un acoplamiento lo más bajo posible.
    - Conseguir que cada componente sea tan independiente como sea posible.



Dependencia entre componentes:

- El acoplomaniento depende de varios factores:
  - Referencias hechas de un componente a otro (incvocaciones)
  - Cantidad de datos pasado de un componente a otro (parámetros)
  - El grado de control de un componente tiene sobre el otro (ej: Composición entre clases)

Clases de Acoplamiento

- Fuertemente acoplados:
  - Si los módulos utilizan variables compartidas o intercambian información de control.
- Débilmente acoplados:
  - Garantizar que los detales de la representación de datos están dentro de un componente.
  - Interfaz con otros componentes mediante una lista de parámetros.
  - Información compartida limitada a aquellos componentes que necesitan acceder a la información.
  - Evitar compartir información de forma global.

Jorney Map

Descubrimiento(awareness)

- Esta primera fase es cuando **un consumidor descubre el producto**. En este estado, las caracterírticas del producto o servicio no son tan importantes.

- Estamos en un a fase de información del usuario, y no se intenta inicitar al usuario a comprar, **simplemente se le informa** de que hay un producto para una necesidad que este puede tener.

Consideración (consideration)

- La fase de consideración es el momento en el que el **consumidor quiere realizar una compra,** y considera diferentes opciones para llevarla a cabo. Esta es la fase de valoración.
- En esta fase, es cuando el usuario debe reconocer la marca y ser consciente de la existencia de esta para tenerla en cuenta para realizar una posible compra. Para ello, aquí sí que hay que **informar a los consumidores de las características del producto, así como los puntos fuertes.**
- A diferencia de la publicidad informativa de la fase de descubrimiento, en este momento hay que lanzar un mensjae más directo sobre las ventajas de nuestro producto con el objetivo de incentivarlo a comprar.

Compra (Purchase)

- La fase de compra es cuando **el usuario ya ha tomado la decisión de compra y decide llevarla a cabo.** En este proces, tener un canal online que no ralentice o frene este proceso es importante, así como un personal en tienda cualificado para que la experiencia de compra en tienda sea buena.

Retención (Retention)

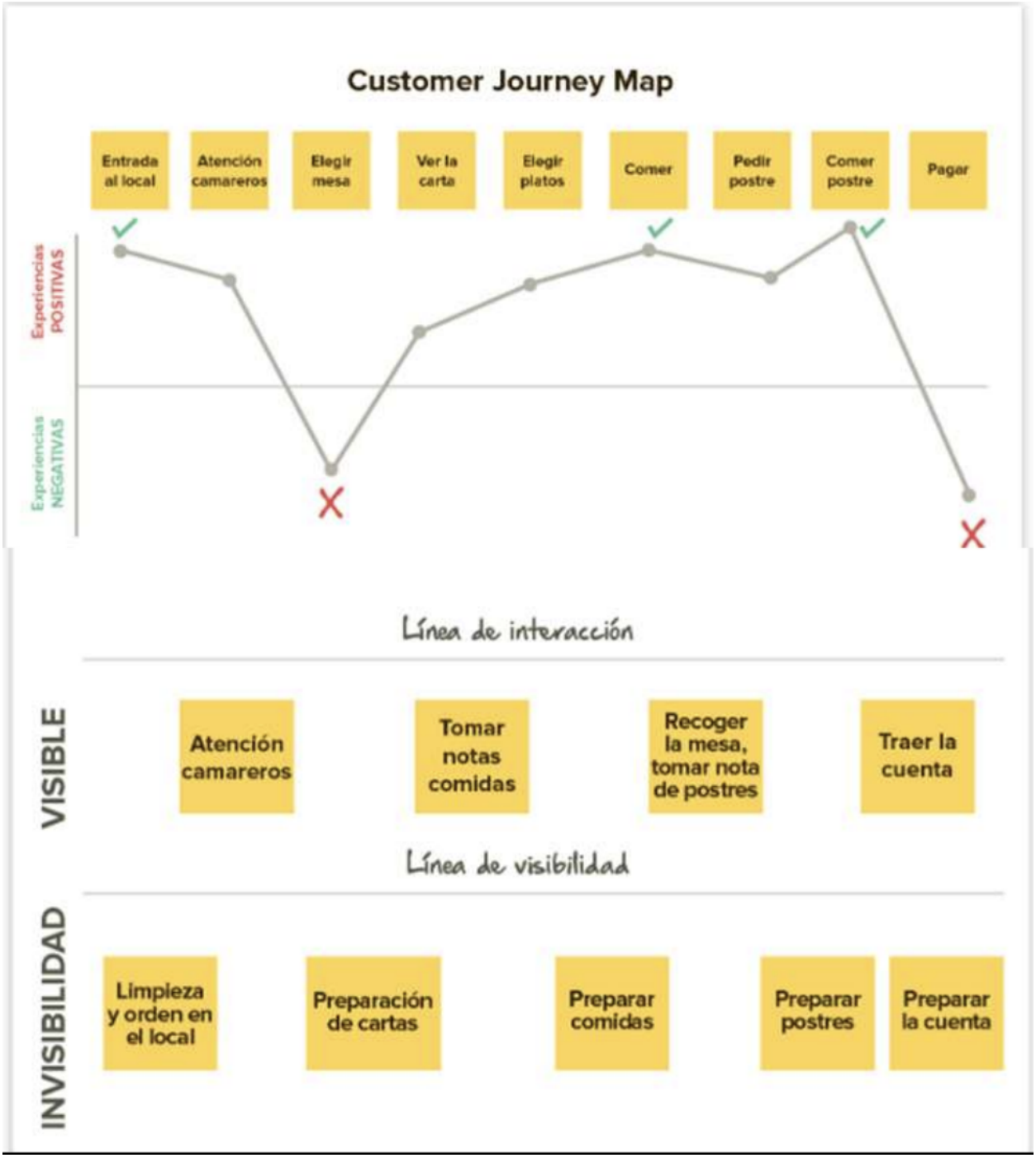
- Esta es la primera fase dentro del servicio post-veta donde se busca **mantener la satisfacción del cliente.**
- El objetivo de esta fase es mantener la relación con el cliente y proporcionar que este repite compras y se fidelice.

Recompendación (Advocacy)

- Tras una experiencia de compra satisfactoria, es posible que los **clientes nos ayuden a mejorar la propia imagen** de marca de la empresa e impactar en fases anteriores para otros usuarios.
- Las redes sociales, **las valoraciones y el boca a boca** son las partes más importantes de la fase de recomendación.

Construcción

- En primer lugar, se dibuja un gráfico en el que el Eje X muestra las **fases por la que pasa el cliente** a lo largo del tiempo y en elEje Y, se define **cómo siente las experiencias,** desde la más negativa, en orjo, hasta la mas positiva, en verde.
  - **Puntos positivos:** Entrada al loca, comida y postre.
  - **Puntos negativos:** Elección de la mesa y pagar.
  - **Stops o puntos críticos:** El primer visitazo al loca, la carta y la comida.


Customer Journey Map
Entrada al local
Atención camareros
Elegir mesa
Ver la carta
Elegir platos
Comer
Pedir postre
Comer postre
Pagar
Experiencias POSITIVAS
Experiencias NEGATIVAS
Línea de interacción
VISIBLE
Atención camareros
Tomar notas comidas
Recoger la mesa, tomar nota de postres
Traer la cuenta
Línea de visibilidad
INVISIBILIDAD
Limpieza y orden en el local
Preparación de cartas
Preparar comidas
Preparar postres
Preparar la cuenta

Contexto

El **Lenguaje Unificado de Modelado (UML)** es una familia de notaciones gráficas respaldada por un único metamodelo que ayuda a describir y diseñar sistemas de software, en particualr aquellos construidos utilizando el estido orientado a aobjetos (OO).

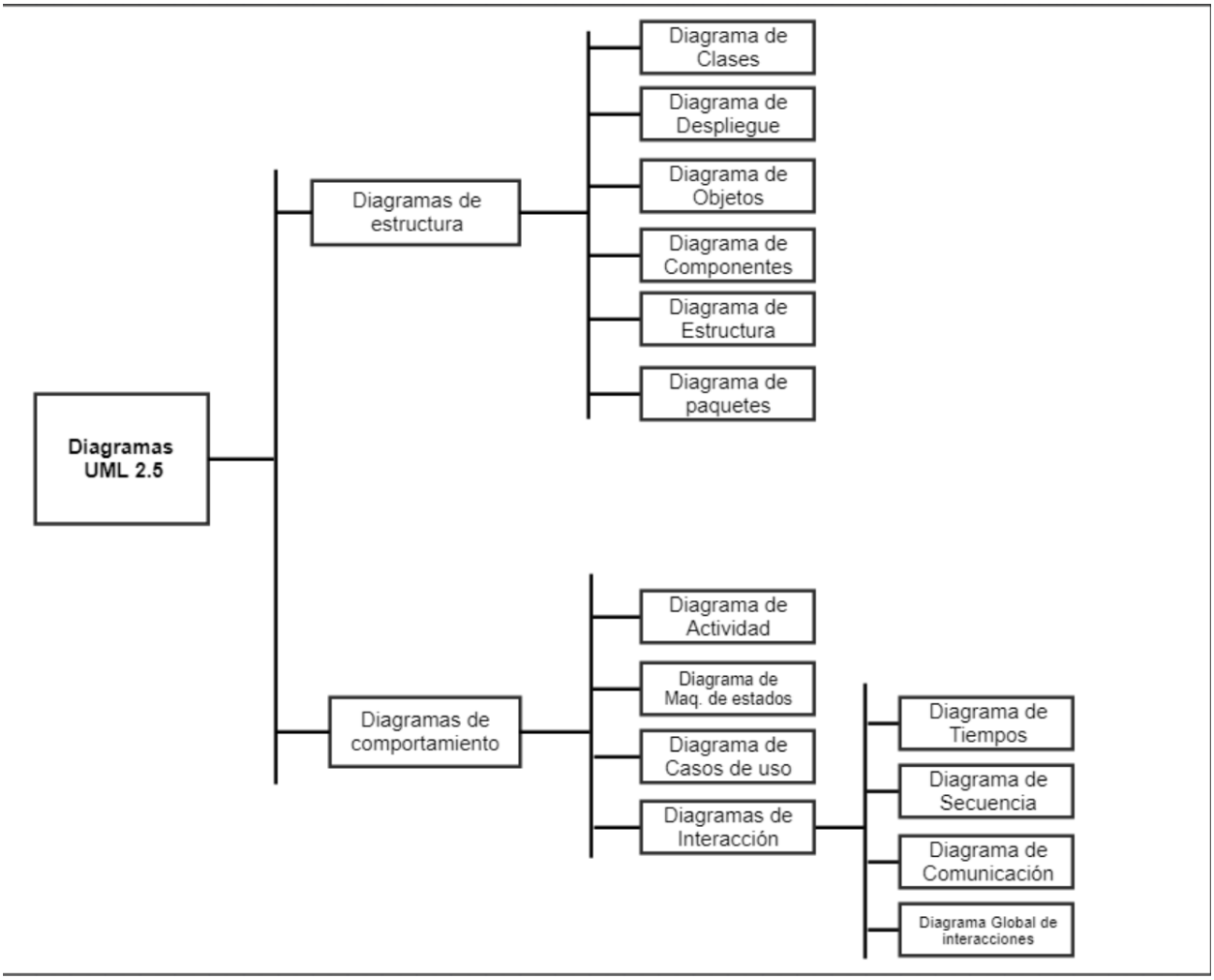
El UML es un término que puede signifcar cosa diferentes para diferentes peronas, dependiendo de su experiencia y percecpción de lo que hace un proceso de ingeniería de software sea efectivo.

A pesar de que los lenguajes de modelado gráfico han existido en laindustria del software por mucho tiempo, hay mucho debate sobre su papel y el papel del UML en particular.

EL UML es un estándar relativamente abierto controlado por el Object Management Group (OMG), un consorcio abierto de empresas que se formó para construir estándares que apoyaran la interoperabilidad de los sistemas orientado a objetos. Desde su apracicion en 1997, el UML ha unficiado muchso lenguajes de modelado gráfico orientado a objetos y ha sido de gra ayuda para los desarrolladores de software.

Tipos de diagrama

- De estructura
- De comportamiento:



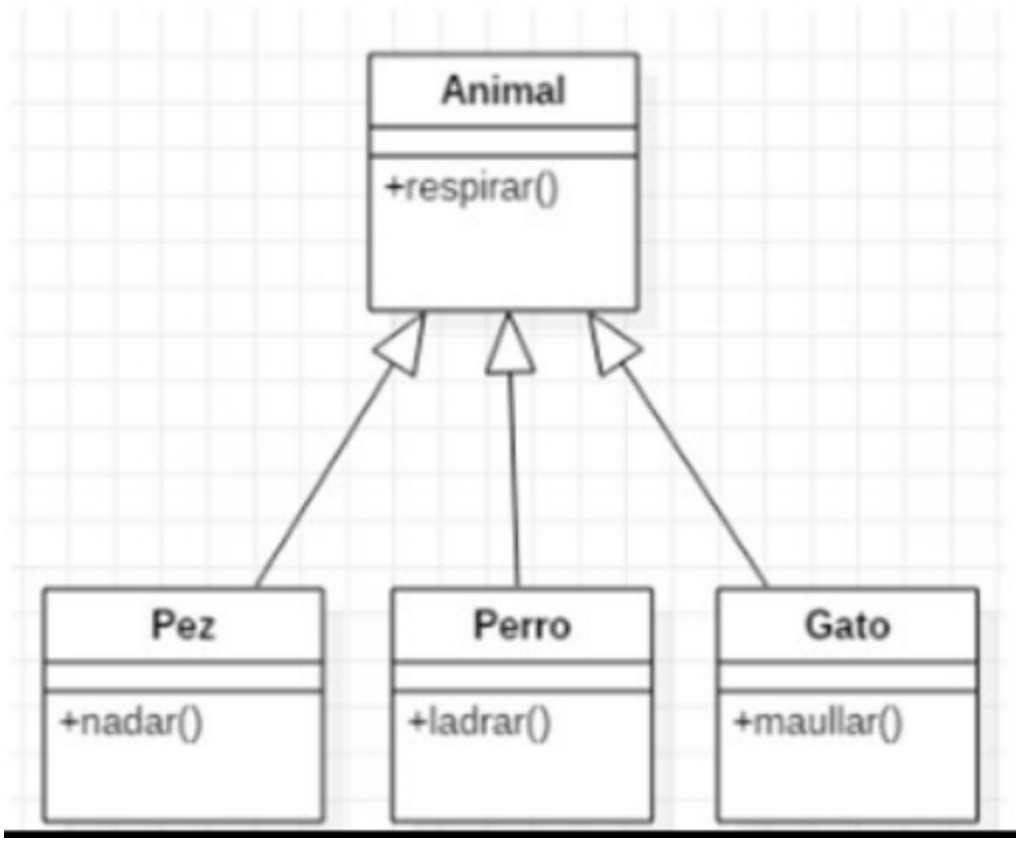
Diagramas de caso de uso

- Los casos de uso son una tecnica para capturar los reqs funcionales de un sistema.
- DDEscriben las interacciones tipicas entre los uusarios del sist
- Los casos de uso proporcionan una narrativa de cómo los actores utilizan el sistema.
    - **Representar los requisitos funcionales.**
    - Representar los **actores** que se comunican con el sistema. Normalmente los acotres del sistema son los **usuarios y otros sistemas externos** que se relacionan con el sistema. En el caso de lo usuarios hay que entender el actor como un “perfil”, pudiendo exisitr varios usuarios que actúan como el mismo actor.

- **Representar las relaciones** entre requisitos funcionales y actores.

Diagrama de clases:

- Describe los tipos de objetos en el sistema y los diversos tipos de relaciones estatitcas que exiten entre ellos.
- También muestra las propiedades y operaciones de una clase y las restricciones que se aplican a ala forma en los
- Los diagrmas de clases UML son ampliamente utilizados y describen los **tipos de objetos** en un sistema, así como las **relaciones** estáticas entre ellos.
- Inlcuyen **propiedades, operaciones y restricciones** aplicables a las conexiones entre objetos.
- Los diagramas de clase **pueden tender a volverse incoherentes** a medida que se xpanden y crecen. Es mejor eviar la creacion de diagramas grandes y **dividirlos** en otros más pequeños que se puedan vincular entre sí mas adelante.
- Usando la notación de clase simple, puedes crear rápidamente **una visión general de alto nivel** de su sistema.
- Cuantas más líneas s esuperpongan en sus diagramas de clase, mas abarrotado s evuelve..
- Usa **colores** para agrupar módulos comunes. Diferentes colos en diferentes clases ayudan al lector a diferencias entre los diversos grupos



Generalizacion:

- Las dimilitudes pueden colocarse en una clase generale (supertio), una relacion de tipo generalizacion a sus subtipos.
- Aunque la herencia es un mecanismo podernisi, rmb trae mucho peso que nos simpre se necesita.
- Problema: los lenguajes en general no permiten herencia multiple.

Asociaciones:

- Este ropo de relacopm es eñ ,as cpmun y utiliza para representar dependencia demantica. Se represetna con una simple linea contine que une las clases que estan incluidas en la asociacion.
- Clase de asociciacion:
    - Durante el proceso de diseño, puede surgir comportamiento u otros atributos que no tienen un responsable claro en la asociacion.

    - En esos casos, surhe la necesidad de tener una nueva clase con al ainfo de asociacion

Agregacio:

- Es parte de

composicion:

- Similar a la aagrefacion , per con una relacion jerarquica mas fuerte entre el objeto y las partes que lo componen
- Los elementos que forman parte no tienen sentido de existencia si no es dentro el elemento que lo compone
- Tienen el mismo tiempo de vida, cuando el objeto padre muerte todos los componentes también.

Dependencia

- se utiliza para reflejar relaciones

Atributos:

- visibilidad nombre: tipo mulitplicidad: valor predeterminado

Notas y comentarios:

- Pueden aparecer en cualquier tipo de diagrama

Enumerados:

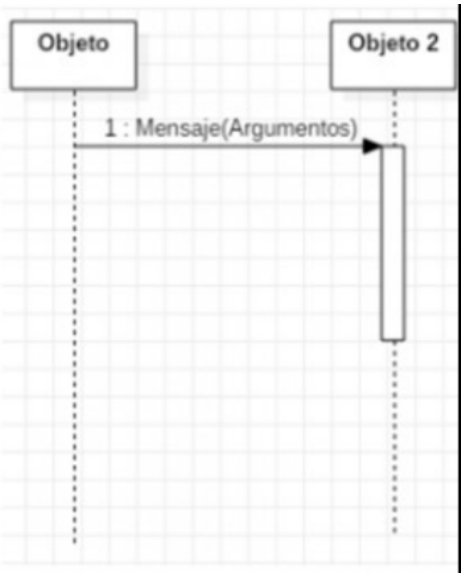
- Set fijo de valoes que no tienen comportamiento

Propiedades derivadas:

- Atributos que el valor Se calcula en base a otros atributos.

Diagrama de secuencia:

- Su objetivo es representar **el intercambio de mensajes entre los distintos XXX** del sisitema para cumplir con una funcionalidad. Define, por tanto, el comportamiento dinámico del sistema de información.
- También se suele construir para comprender mejor el diagrama de clases, ya que el diagrama de secuencia muestra como objetos de esas clases interactúan haciendo intercambio de mensajes.



Construcción:

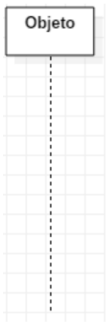
El diagrama de secuencia está construido a partir de dos dimensiones:

- Horizontal: Representa los objetos que participan en la secuencia.
- Vertical: Representa la línea de tiempo sobre la que los elementos actúan. Va de arriba (menor tiempo) hacia abajo (mayor tiempo ???)

Esta compuesto por dos elementos: OBJETOS Y MENSAJES

Objeto

- Un objeto representa a un participante en la interacción. Puede ser una instancia de una clase, un módulo, un grupo de clases, un objeto es un componente software que tiene una funcionalidad especifica.
- A diferencia de otros diagramas, cada uno de los objetos añadidos al diagrama representa solamente una instancia de ese objeto y no varias. En caso de ser un elemento multivaluado, debe dejar constancia de cual es el elemento que. está trabajando.

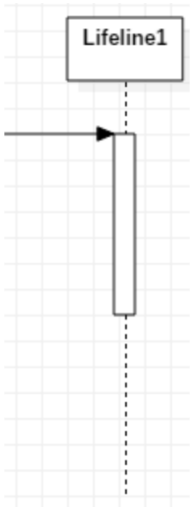


Notación de un objeto

La linea vertical, se llama linea de vida y representa el tiempo en el que el objeto está presente.

Los objetos que existen previamente al comienzo de la interacción se situan en el eje horizontal mientras que los objetos que se crean durante el transcuros de l amisma se situan en el momento de la creacion.

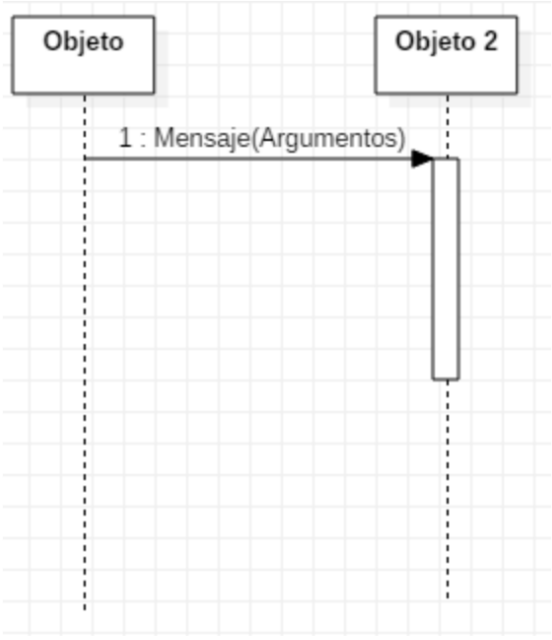
Los objetos contienen el denominado foco de control que no es más que el tiempo en el que tal objeto está llevando a cabo algún trabajo



Mensaje

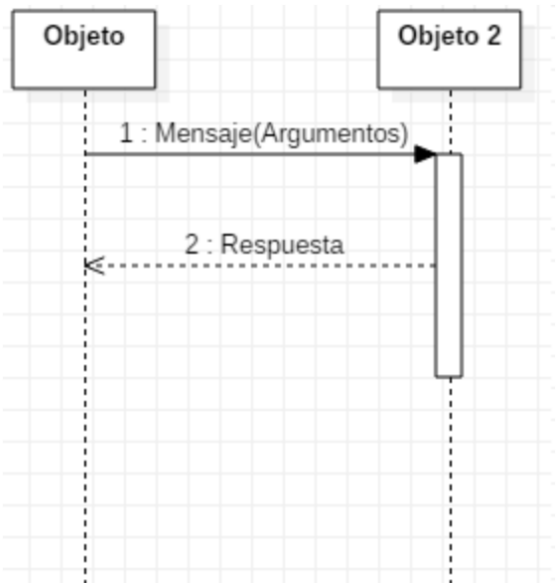
Se utiliza un mensaje para represetnar el paso de un mensaje entre dos objetos o entre un objeto y sí mismo.

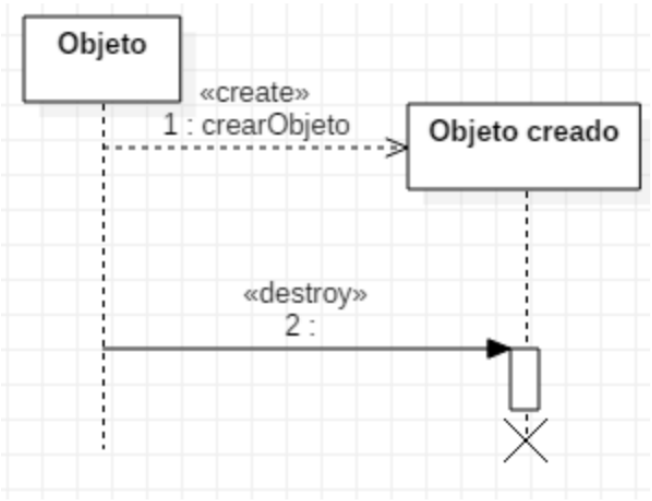
Se representan con una flecha que incluye el nombre del mensaje y los argumentos que incluye y que va desde el objeto que envía hasta el que lo recibe.



Notación de un mensaje

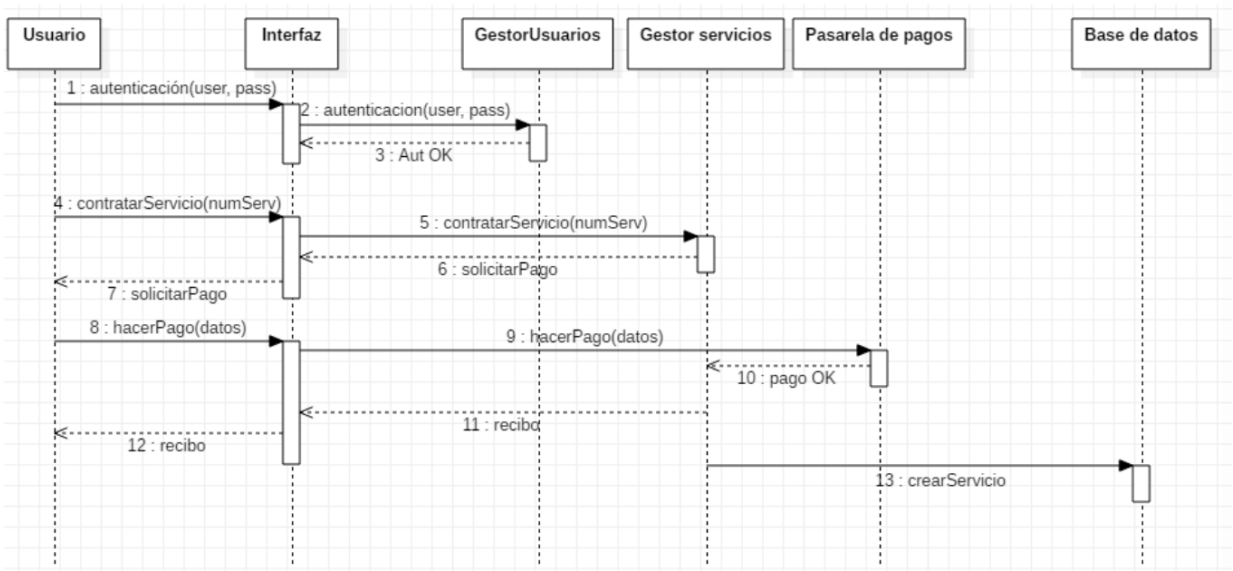
Cuando el objeto que recibe el mensaje envía una respuesta:





Notación create y destroy

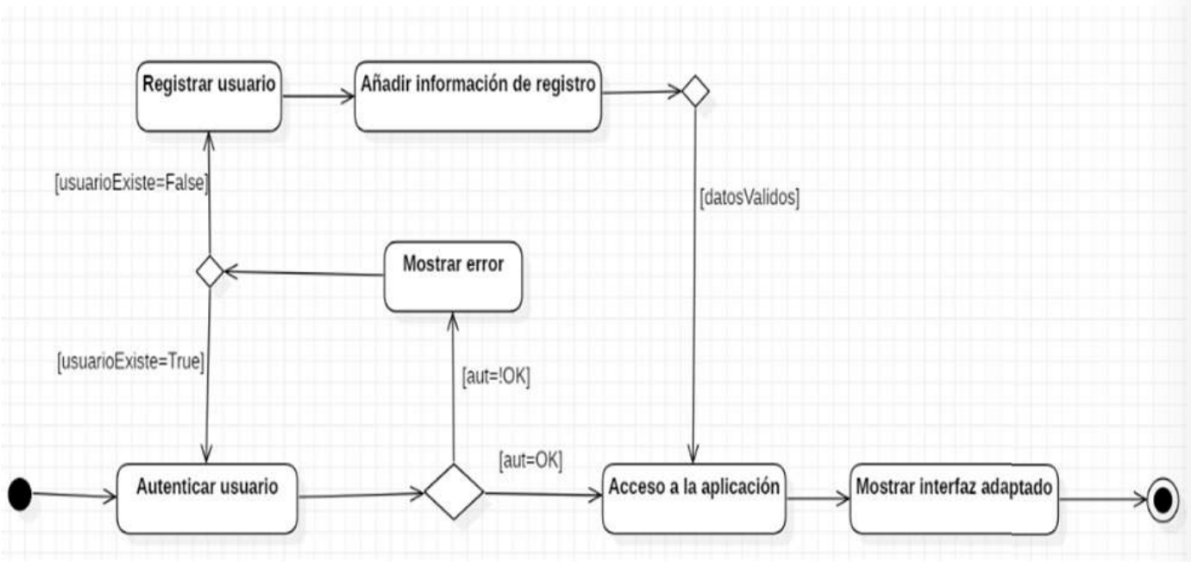
Ejemplo:



Ejemplo de diagrama de secuencia

Diagrama de actividad

- Los diagramas de actividades muestras una secuencia de acciones, **un flujo de trabajo queue va desde un punto inicial hasta un punto final.**
- La finalidad de este diagrama es modelar el **workflow** de una actividad a otra, pero sin tener en cuenta el paso de mensajes entre ellas.
- También es utilizado para modelar las actividades, que podemos asemejar a requisitos funcionales de negocio, por lo que este diagrama tendrá una influencia mayor a la hora de comprender el negocio o sus funcionalidades que en la propia implementación. Hay que tener en cuenta que este diagrama ofrece una visión a alto nivel.
    - Documentar los **requisitos** de negocio.
    - Modelar el **flujo de trabajo** entre actividades y/o entre subsistemas.
    - Comprender a **alto nivel** las funcionalidades del sistema de información.



Construcción

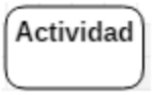
Esta compuesto por: Actividades, flujos de control, nodo inicial y nodo final.

Actividad

La actividad es una conducta parametrizada representada como flujo coordinado de acciones.

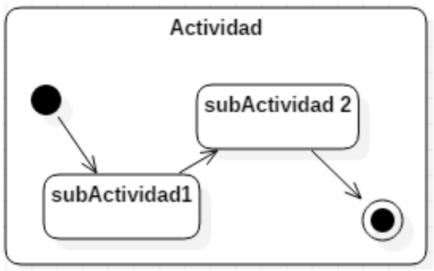
El flujo de ejecución que representa la funcionalidad deseada se modela utilizando nodos de actividad concectados por flujos de control.

Un nodo puede ser: un comportamiento subordinado, un cálculo artimético, una llamada a una operación o la manipulación del contenido del objeto. También incluyen flujo de construcciones de control, como sincronización, decisión y control de concurrencia.



Notación de una actividad

Se puede representar una actividad que incluye un subdiagrama de actividades:



Notación de una actividad compuesta

Las actividades que no tienen mas descomposición se denominan acciones.

Las activiades son nombradas utilizando verbos del modelo de negocio (Buscar, Actualizar, Hacer, etc).

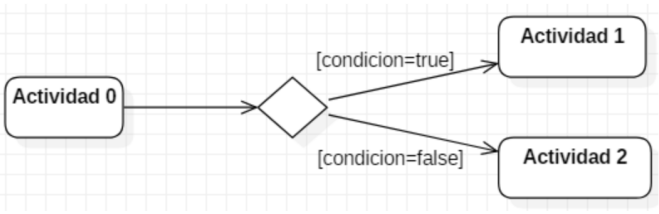
Flujo entre Actividades

Es una clase abstracta para las conexciones dirigidas a lo largo de las cuales los tokens u objetos de datos fluyen entre los nodos de actividad.

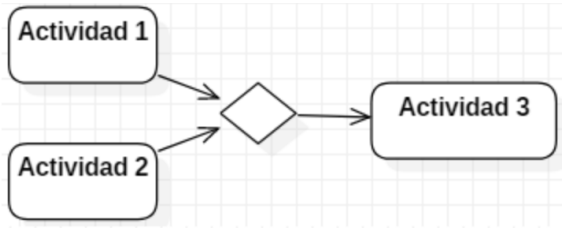
Incluye flujos de control y de objetos.

Notación de un flujo de actividad

Pueden usar condiciones, recibe el nombre de **nodo de decisión**:



El caso contrario es la llamada **nodo de fusión**, que recibe 2 o mas flujos y emite 1:



Notación nodo de fusión

Pueden ser combinads, haciendo que un nodo reciba varios flujos y a su vez emita también varios flujos.

Nodo Inicial y Nodo Final

El Nodo Inicial es un nodo de control en el que se inicia el flujo cuando se invoca la actividad. Solo hay uno por diagrama.

Las actividades pueden tener mas de un nodo inicial. En este caso, al invocar la activivdad se inician varios flujos, uno en cada nodo inicial.

Los flujos pueden comenzar en otros nodos.

Notación de un nodo inicial

Una actividad puede tener mas de un nodo final. El primer alcanzado detiene todos los flujos en la actividad. Un token que llega a un nodo final de actividad finaliza la actividad.

Notación de un nodo final

Diagrama de paquetes

- Los diagrmas de clases sirven para representar la estrucutra más básica de un sistema.
- A medida que estos empiezan a crecer, necesitamos una forma más práctica de visualizar las relaciones y dependencias.
- Los diagrmas de paquetes nos permiten agrupar por algún concepto con un mayor nivel de abstracción

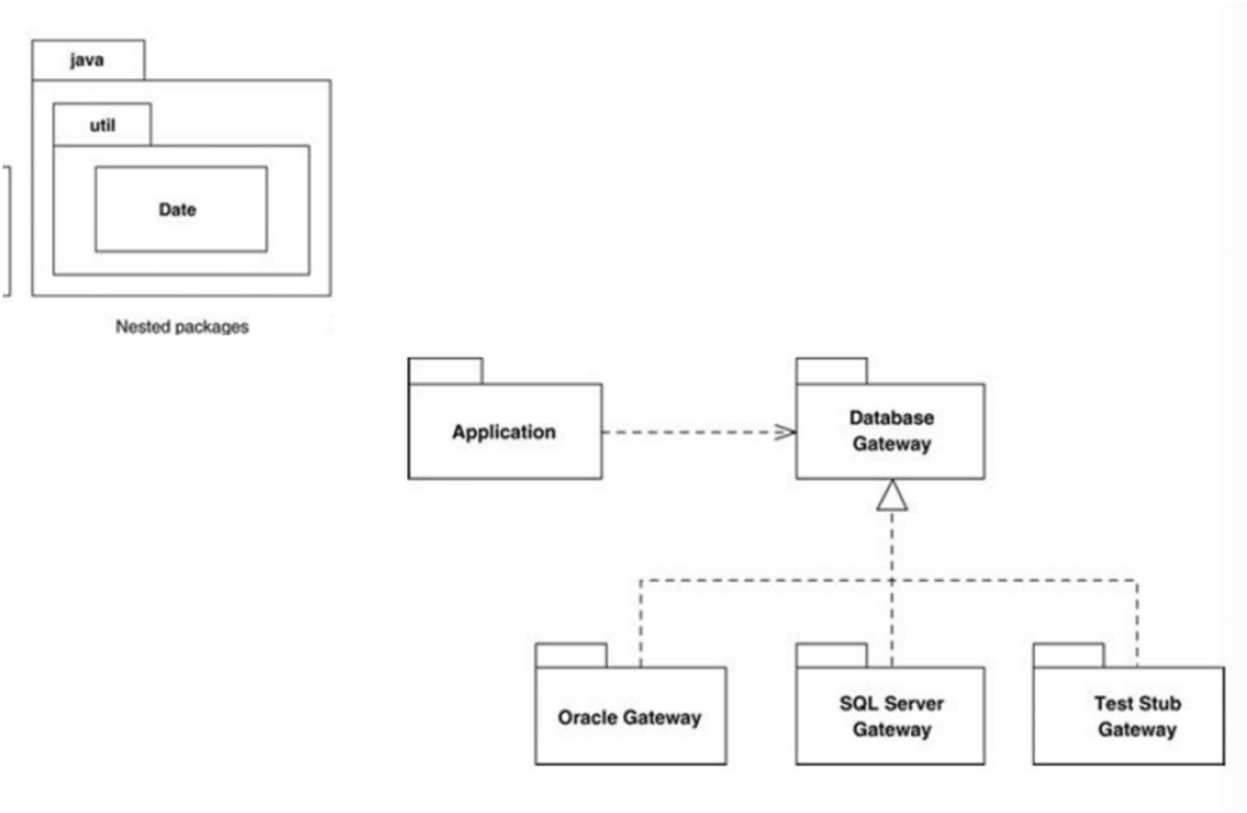


Diagrama de componentes

- Podemos verlo como una agrupación de módulos, por algún criterio.
- Sirve para documentar las relaciones entre el sistema a través de las interfaces

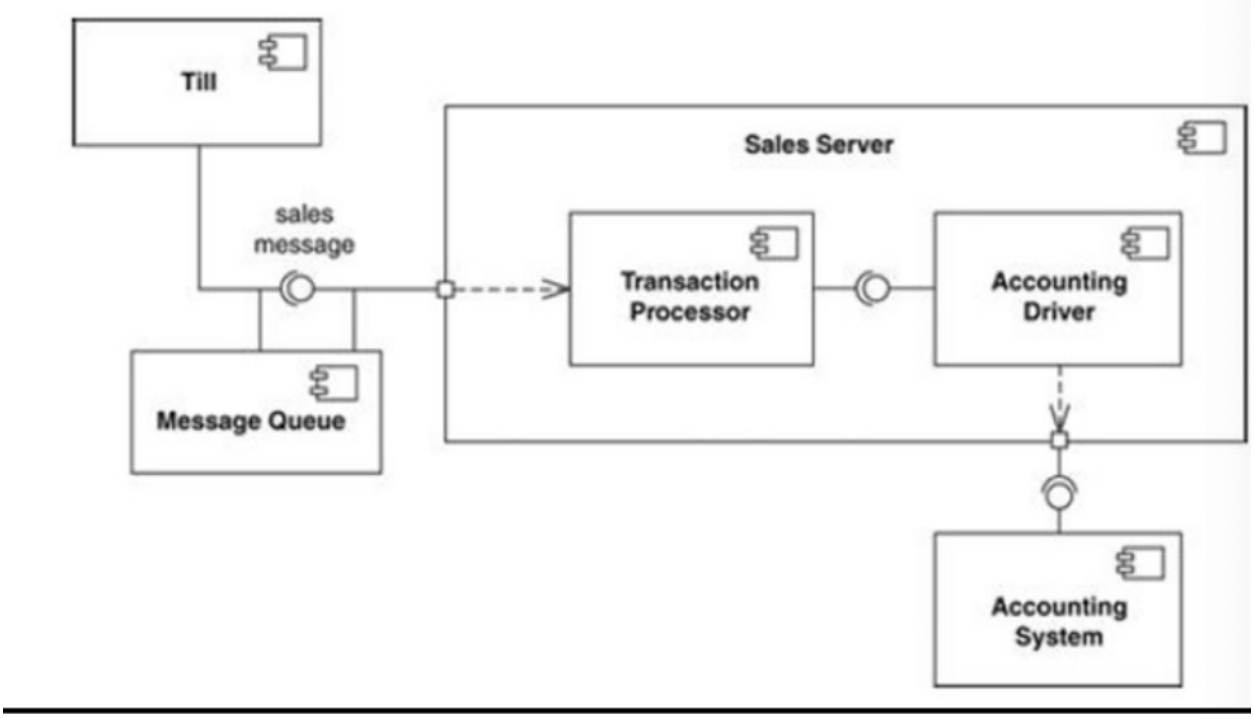
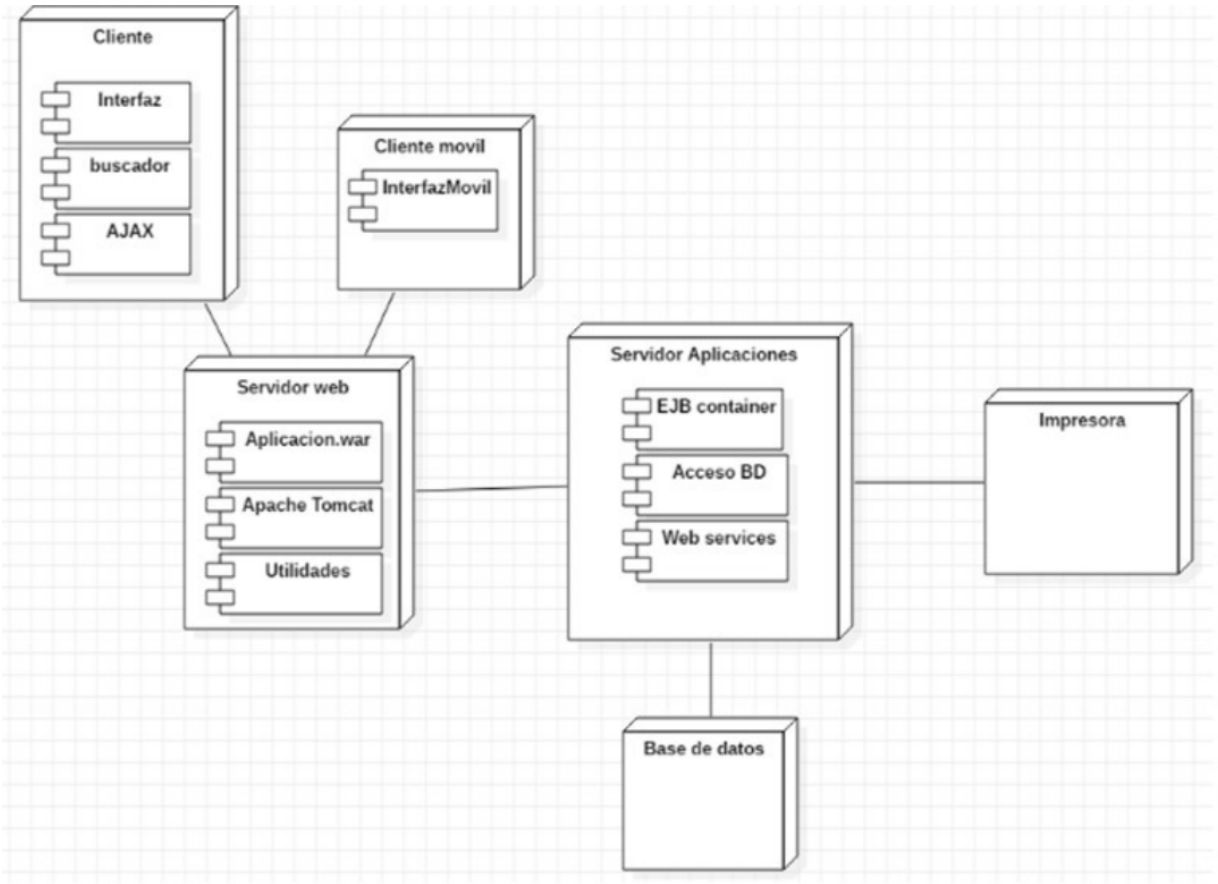


Diagrama de deploy

- Suele se utilizado junto con el diagrama de compoentens (incluso a veces con el de paquetes) de forma que, juntos, dan una visión general de como estará desplegado el sistema de información.
- El diagrama de componentes muestra que componentes existen y como se relacionan mientras que el diagrama de despliegue es **utilizado para ver como se sitúan estos componentes lógico en los distintos nodos físicos.**
- Como prácticamente todos los diagramas de UML, puede ser utilizado para represetnar aspectos generales o muy específicos, siendo utilizado de forma más común para aspectos generales.


Cliente
Interfaz
buscador
AJAX
Cliente movil
InterfazMovil
Servidor web
Aplicacion.war
Apache Tomcat
Utilidades
Servidor Aplicaciones
EJB container
Acceso BD
Web services
Impresora
Base de datos