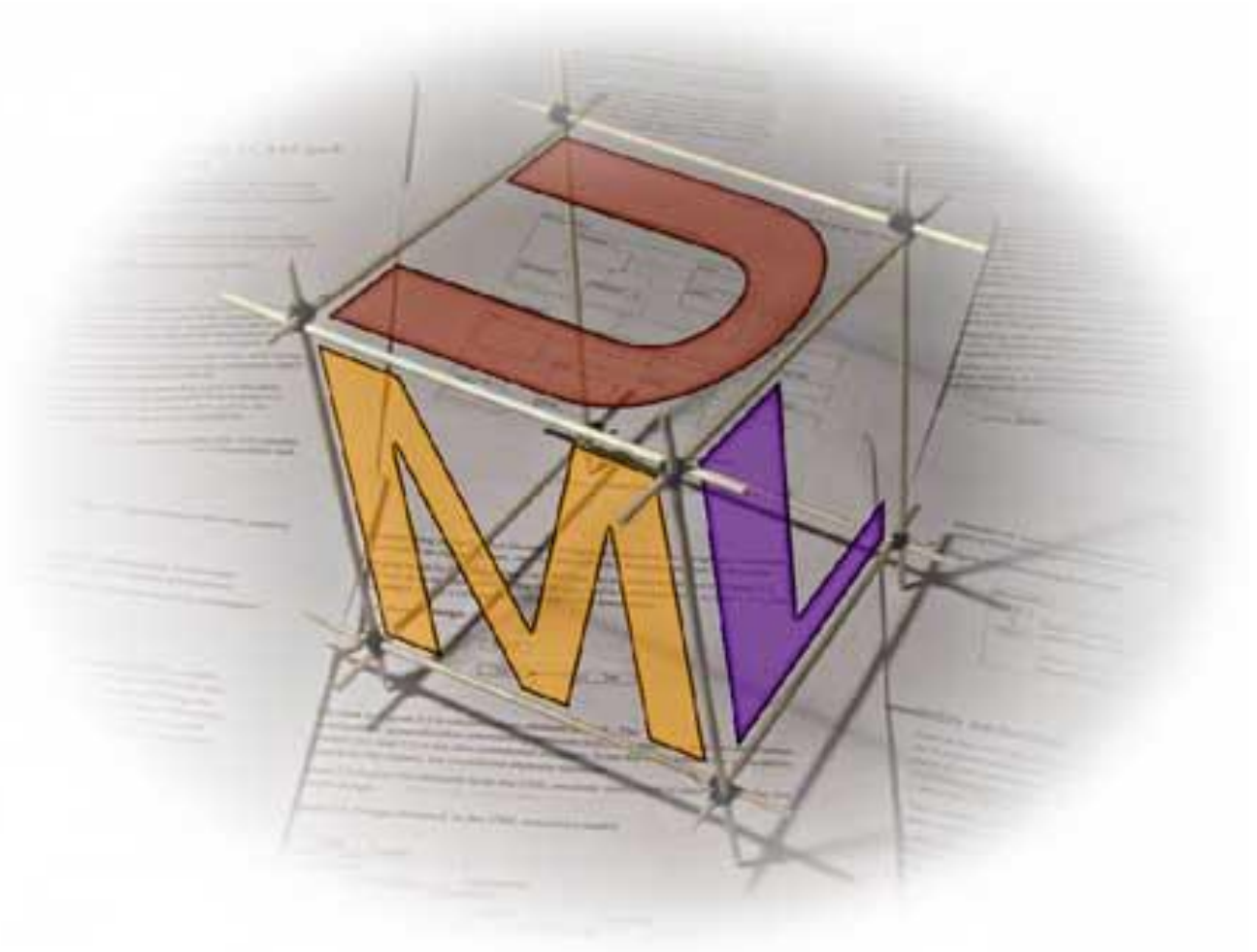

UML

Unified Modeling Language



CONCEPTES I REFERÈNCIES PRELIMINARS

OMG (Object Management Group)

És un organisme sense ànim de lucre creat el 1989 destinat a fomentar i promocionar les tecnologies de desenvolupament orientat a objectes.

<https://www.omg.org>

UML (Unified Modeling Language)

Tot i que l'UML es pot fer servir per a modelar (dissenyar) **qualsevol sistema** la utilització que se'n fa s'orienta als **sistemes de software**. De les seves especificacions se n'encarrega l'**OMG**.

L'UML es compon d'un ampli conjunt de **diagrames** que tenen com objectiu descriure principalment:

- quins són els **actors** que utilitzaran el sistema i quina funcionalitat n'obté cada actor
- quins són els **elements** que componen el sistema de software
- quins són els **processos** i **relacions** que s'estableixen entre els elements del sistema de software
- quins són els diferents **estats** en que cada un dels elements es pot trobar
- com es produeixen les **transicions** d'un estat a un altre

De diagrames n'hi ha molts, els podem classificar en dos grups:

- Els d'**estructura**.
- Els de **comportament**. Dins d'aquest grup es pot parlar d'un subgrup que són els diagrames d'**interacció**.

Algunes classificacions parlen dels diagrames d'estructura com **estàtics** i dels de comportament i interacció com a **dinàmics**.

El disseny UML **no està condicionat pel llenguatge de programació** que es farà servir per a implementar el sistema de software sinó que permet tenir una perspectiva global del sistema de la mateixa manera que els plànols d'una casa ens permeten tenir-ne una visió global sense

necessitat de construir-la. Tot i la independència respecte el llenguatge de programació no seria massa normal utilitzar un llenguatge de programació que no fos orientat a objectes.

Història de l'UML (Unified Modeling Language).

L'UML neix a mitjans dels anys 90 partir del treball de [Grady Booch](#), [Ivar Jacobson](#) i [James Rumbaugh](#) que treballaven per a l'empresa [Rational Software](#).

L'objectiu del seu treball era acabar amb la [diversitat](#) de tècniques i metodologies que s'utilitzaven en el disseny de software i establir un [model unificat](#) que en permetés l'estandardització. Com que en aquell temps la tendència principal en Programació ja era la [Programació Orientada a Objectes](#), l'estructura de l'UML segueix també aquesta perspectiva.

A mitjans de [1996](#) es va publicar la versió [UML 1.0](#) i es van formar grups de treball compostats per diversos investigadors i desenvolupadors per a que el milloressin. El resultat d'aquesta feina es va convertir en la versió [UML 1.1](#) que va ser adoptada com estàndard per l'OMG a finals de [1997](#). D'aquesta primera versió se'n van anar fent modificacions per a resoldre errors o mancances

versió	any
1.2	1999
1.3	2000
1.4	2001
1.5	2003

Els avenços tecnològics i l'aparició de noves metodologies en l'àmbit del desenvolupament de software van comportar la necessitat de revisar els diagrames i fer algunes ampliacions. D'aquesta manera va néixer a mitjans de [2005](#) l'[UML 2.0](#), que també s'ha anat ampliant fins arribar a la [versió 2.5.1](#) que va sortir a finals de [2017](#).

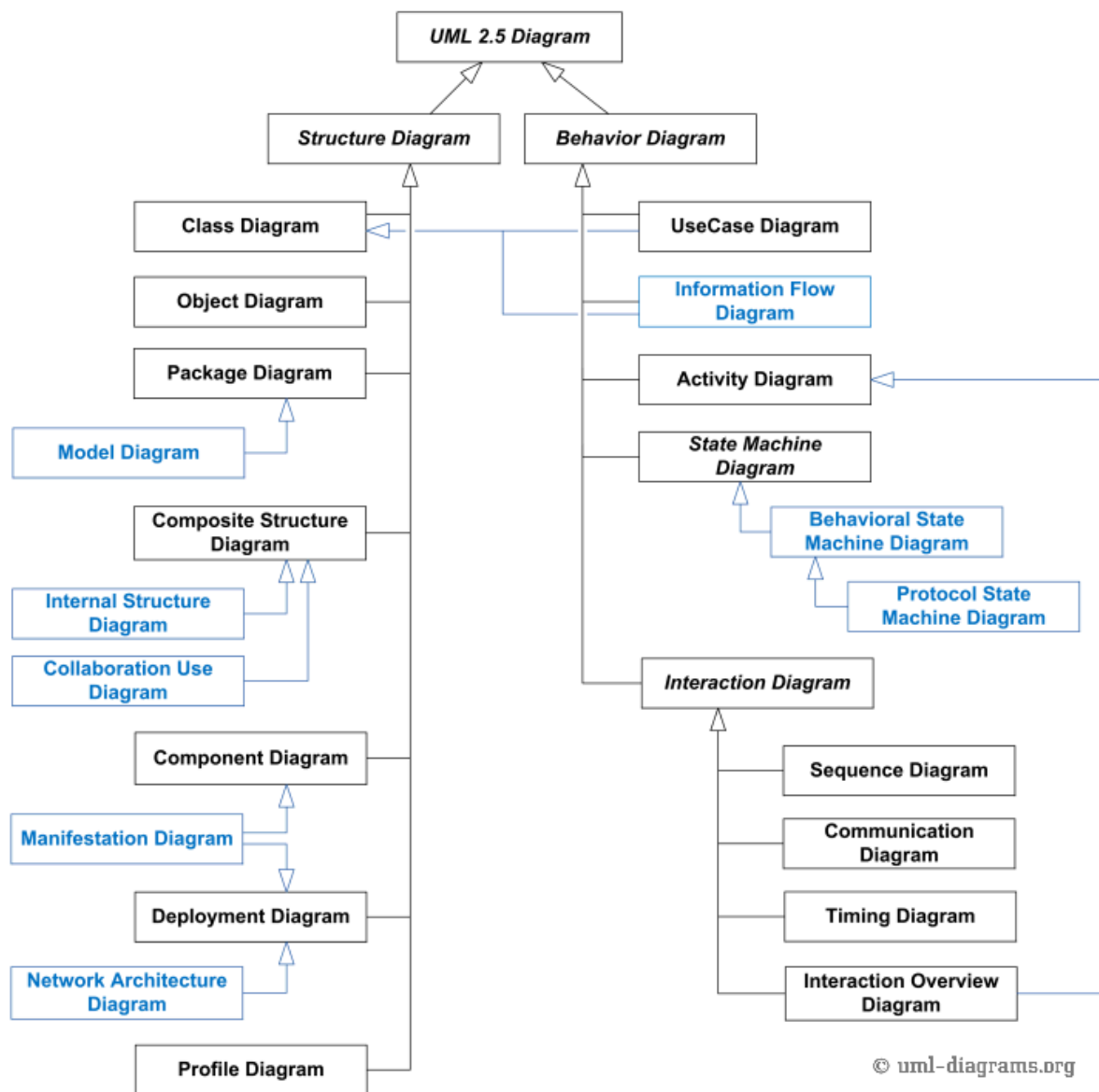
<https://www.omg.org/spec/UML/>

DIAGRAMES UML

Diagrames de la versió UML 2.5

<https://www.uml-diagrams.org/>

Els diagrames que estan en negre són els que corresponen realment a l'especificació oficial UML 2.5 mentre que els que estan en blau formen part d'ampliacions que s'hi poden integrar.



Exemples de diagrames UML

<https://www.uml-diagrams.org/index-examples.html>

DIAGRAMA DE CASOS D'ÚS

Definició i components

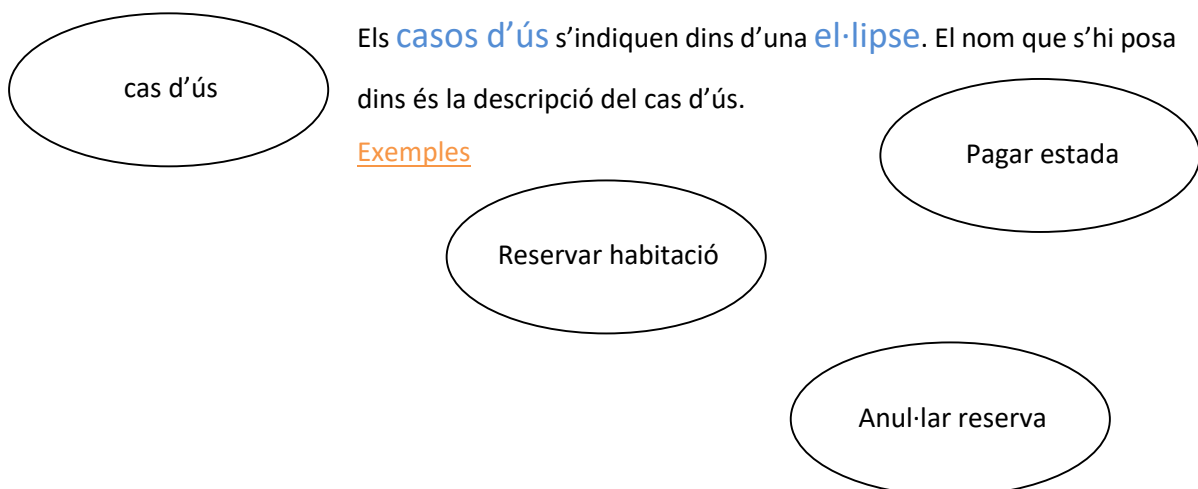
En el primer capítol d'aquest dossier hem parlat que hi ha diagrames d'estructura i diagrames de comportament. El diagrama de classes és clarament un dels diagrames d'estructura i ara parlarem del **diagrama de casos d'ús** que és un dels principals diagrames de comportament.

El diagrama de casos d'ús especifica la funcionalitat del sistema utilitzant **actors** i **casos d'ús**. És útil, entre d'altres coses, per a facilitar la comunicació entre el desenvolupador i l'usuari final.

Casos d'ús. Són **conjunts d'accions, serveis o funcions** que el sistema ha d'executar o oferir. Encara que es pot fer un diagrama tan detallat com es vulgui, normalment aquests diagrames mostren una descripció global de la funcionalitat del sistema i per això els casos d'ús són força generals. Per exemple, en l'exemple de l'hotel, posarem el cas «Reservar habitació» englobant totes les operacions necessàries per a fer-ho (mirar disponibilitat, assignar habitació a client, etc.).

Actors. Els actors són **aquells que interactuen** amb el sistema. Per a cada actor el sistema oferirà un conjunt de casos d'ús, és a dir, una funcionalitat. En l'exemple de l'hotel els actors podrien ser el client, el gestor de l'hotel però també podríem tenir altres actors com els empleats de neteja, el director/a, etc. Els actors poden ser usuaris però també poden ser organitzacions, dispositius com per exemple sensors, sistemes externs, el propi sistema informàtic,...

Notació





Els **actors** s'indiquen amb un ninot. [Exemples](#)



Client

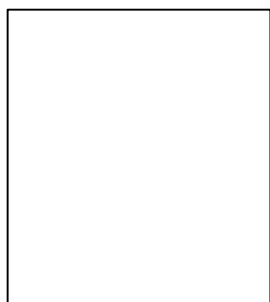


Empleat

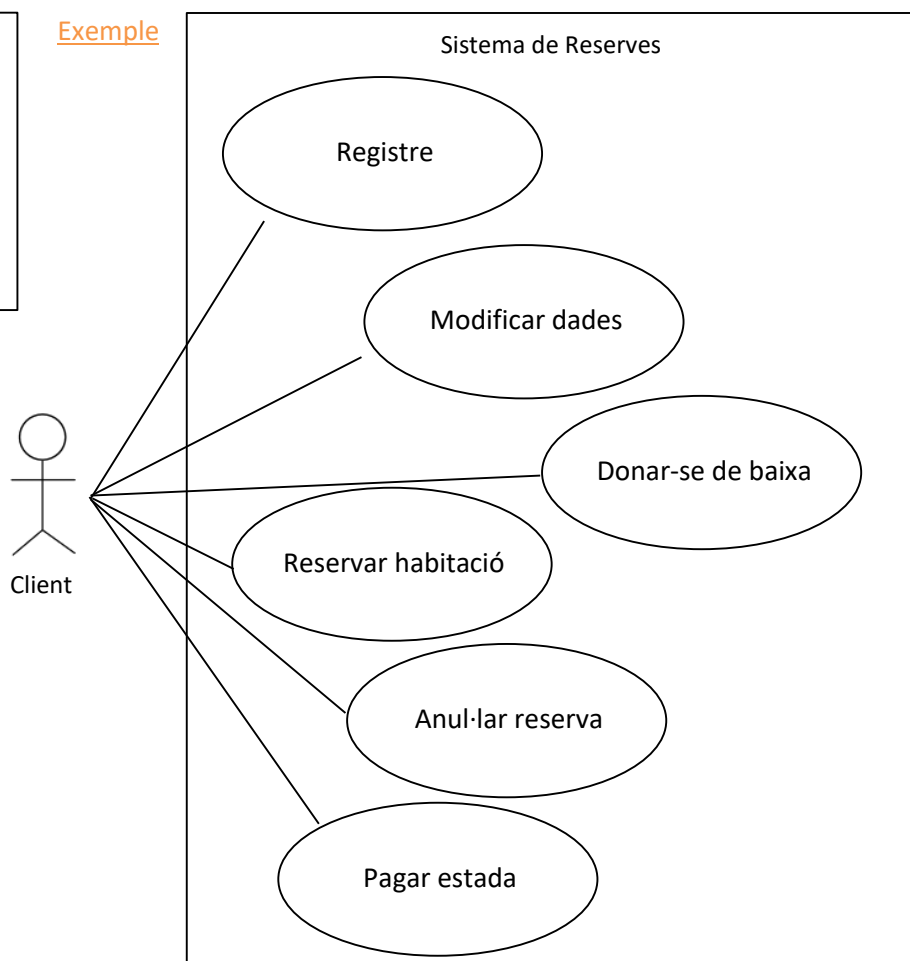


WebServer

Sistema. Un sistema es defineix a partir d'un conjunt de casos d'ús i es representa amb un rectangle que els agrupa.



[Exemple](#)

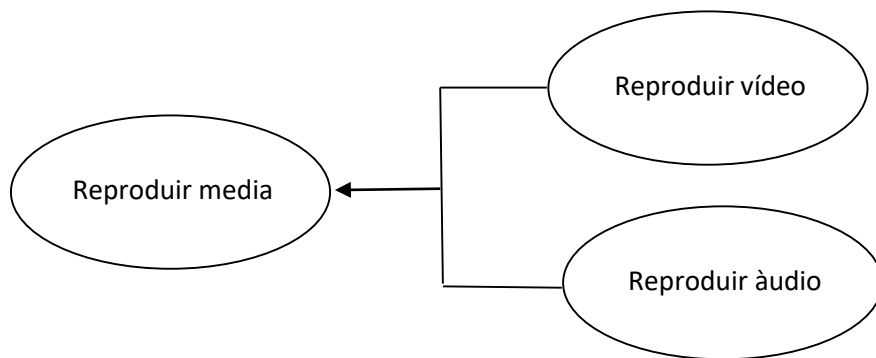


Associació. Una associació és una relació que s'estableix entre un actor i un cas d'ús. Dit d'una altra manera, un actor té associat un cas d'ús si «hi ha un camí» que li porta.

Representem les associacions amb línies que uneixen l'actor amb el cas. Ho podem veure en l'exemple anterior.

Relacions entre casos d'ús.

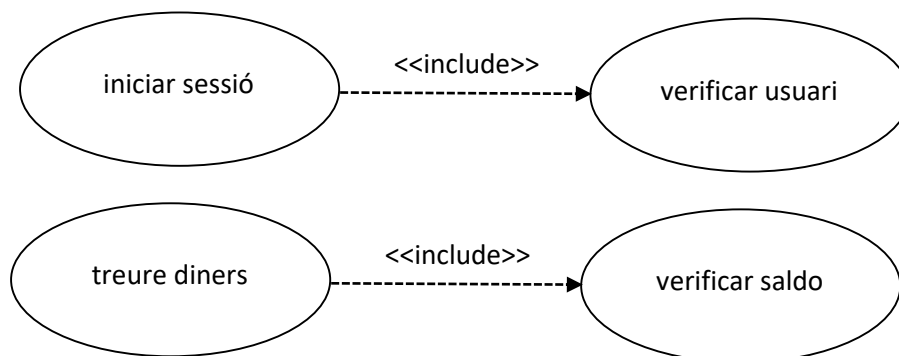
Generalització. Alguns casos d'ús són modificacions o ampliacions d'un altre, en aquest cas parlem de **generalització** o **especialització**. Per exemple, si estem en un sistema de reproducció multimèdia podem tenir un cas que sigui «**Reproduir media**» que es generalitzi en altres com «**Reproduir vídeo**» i un altre que sigui «**Reproduir àudio**». Ho representarem amb una fletxa que va cap al cas general.



Inclusió. Alguns casos d'ús n'inclouen d'altres. Per exemple, en un sistema en que tenim un cas d'ús «**Iniciar sessió**» aquest en pot **incloure** un altre que sigui «**Verificar usuari**».

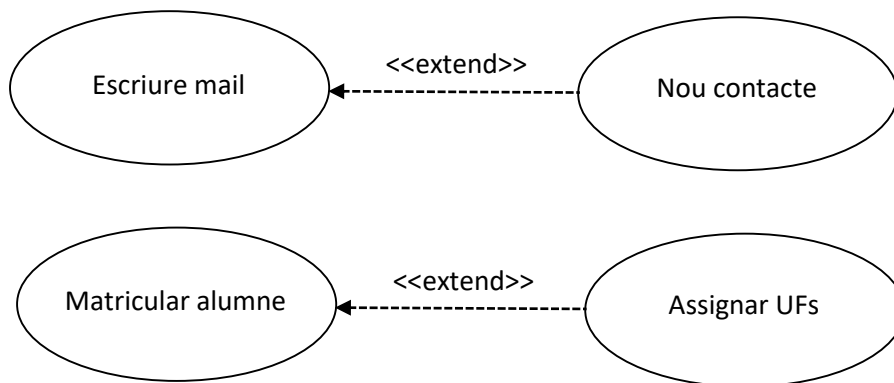
Es representa amb una fletxa discontinua amb l'etiqueta **<<include>>** que va cap al cas inclòs.

Sovint, en els diagrames de casos d'ús no s'especifiquen els casos inclosos per a facilitar-ne la lectura i perquè funcions com les d'autenticació d'usuaris, validació de dades, ... ja es donen per implícites.



Extensió. Alguns casos d'ús poden de manera opcional accedir a altres casos d'ús amb l'objectiu d'ampliar la funcionalitat que ofereixen. Per exemple, en un sistema en que tenim un cas d'ús «Escriure mail» aquest pot **estendre's** a un altre que sigui «Crear contacte» el qual pot ser cridat o no depenent de si l'usuari està escrivint un mail per a un destinatari que no té en la llibreta de contactes i li vol afegir.

Es representa amb una fletxa discontinua amb l'etiqueta **<<extend>>** que va de l'extensió al cas estès.



Sovint als casos estesos també estan associats amb l'actor, és a dir, que hi pot accedir directament. En els exemples equivaldria a dir que pot fer un contacte nou sense escriure un mail o pot assignar UFs sense fer la matriculació (si ja l'ha feta anteriorment).

EXERCICI. Fes el diagrama de casos d'ús d'un negoci semblant a Wallapop

EXERCICI. Fes el diagrama de casos d'ús d'una aplicació per a jugar al Tetris

EXERCICI. T'han demanat que desenvolupis una aplicació web per al Torneig de 24 hores de Bàsquet de la Festa Major del teu poble. L'aplicació ha d'estar pensada per 2 tipus d'usuari: l'administrador que és qui gestionarà totes les dades i un usuari genèric que podrà consultar els equips, els partits, els resultats i la classificació.

L'administrador haurà d'iniciar sessió per a poder entrar a l'aplicació i gestionar tot el que ha de gestionar que són els equips, els jugadors, els partits i els resultats. A mesura que s'introdueixin resultats s'anirà actualitzant automàticament la classificació.

Com a mesura de seguretat l'administrador també ha de poder canviar el password.

L'usuari genèric no ha d'iniciar sessió ja que les dades que es poden consultar es vol que siguin públiques i que qualsevol persona amb accés a l'aplicació les pugui veure.

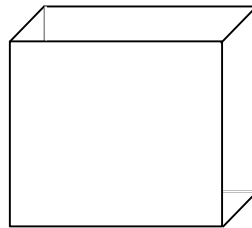
EXERCICI. Fes el diagrama de casos d'ús d'un sistema semblant a Slack (versió gratuïta)

DIAGRAMA DE DESPLEGAMENT / D'ARQUITECTURA

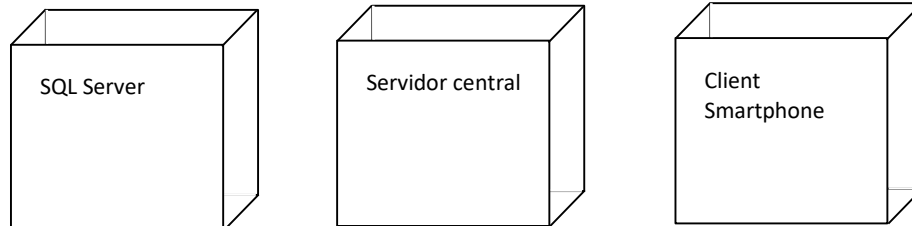
Definició i components

El **diagrama de desplegament** representa l'arquitectura del sistema en temps d'execució incloent els elements de hardware i els de software i les interconnexions físiques o virtuals entre aquests elements. En alguns documents també se'n diu de **distribució**.

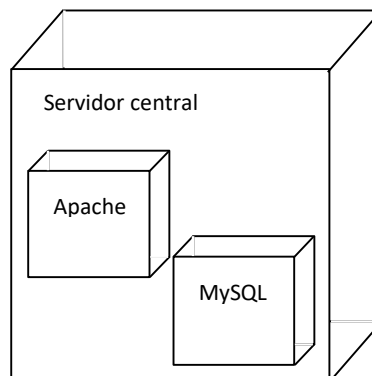
Nodes. Els nodes representen elements computacionals ja siguin elements de hardware o de software. Per exemple, un servidor web, un SGBD, una impressora, un smartphone, un ordinador, ... Es representen amb un prisma



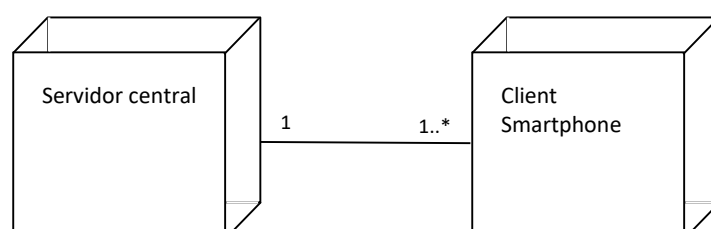
Exemples



Dins d'un node n'hi poden haver d'altres



Associacions. Les associacions permeten establir una relació de cardinalitat entre els nodes, per exemple, suposem un cas en que un servidor es pot relacionar amb N clients però un client només es pot relacionar amb 1 servidor.



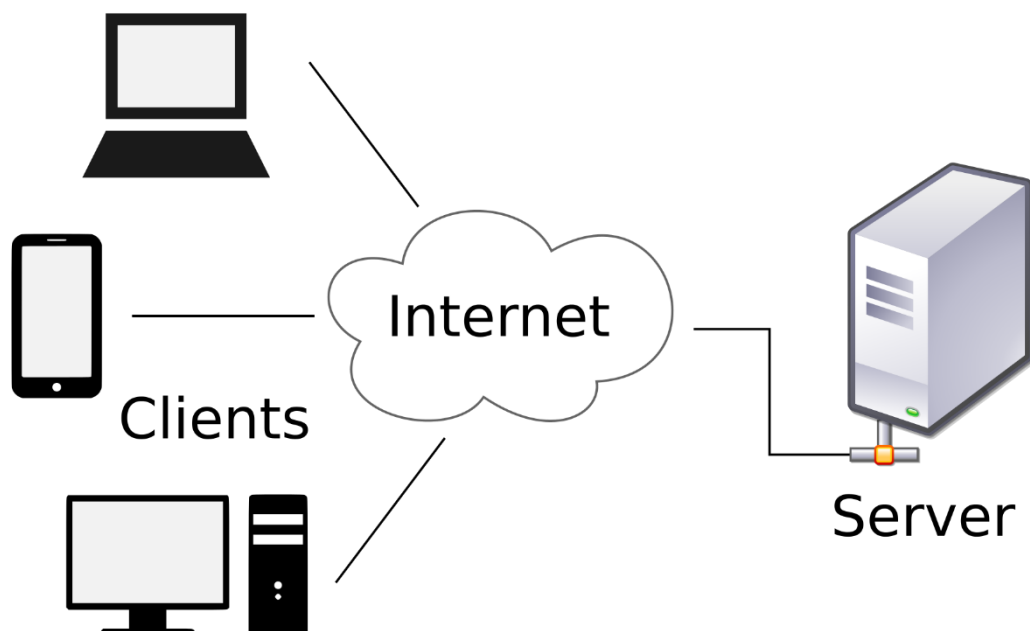
Un altre element del diagrama de desplegament és l'**artefacte** que representa un element que es va generant durant l'execució i s'inclou dins d'un node. Per exemple, un artefacte podria ser l'arxiu de còpies de seguretat de la BD i estar inclòs dins el node del servidor central. Un altre artefacte podria ser un arxiu de configuració, un document estadístic, etc.

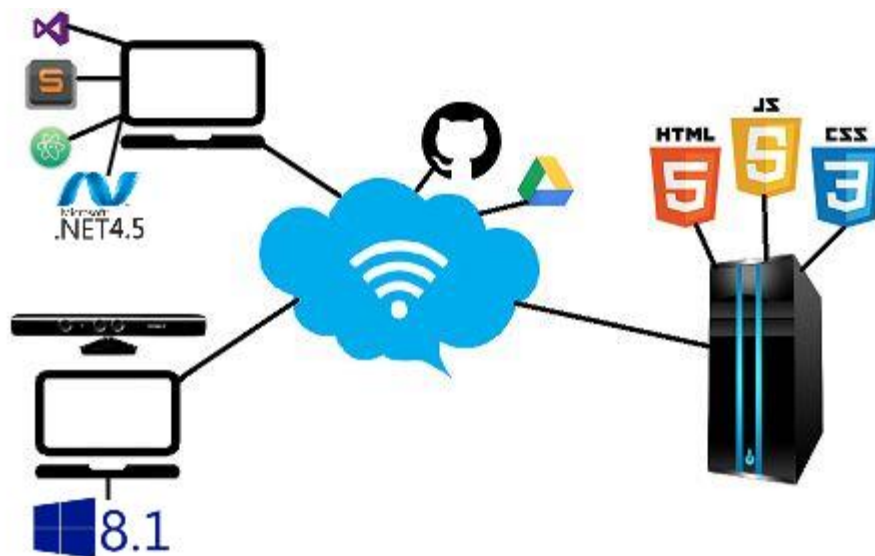
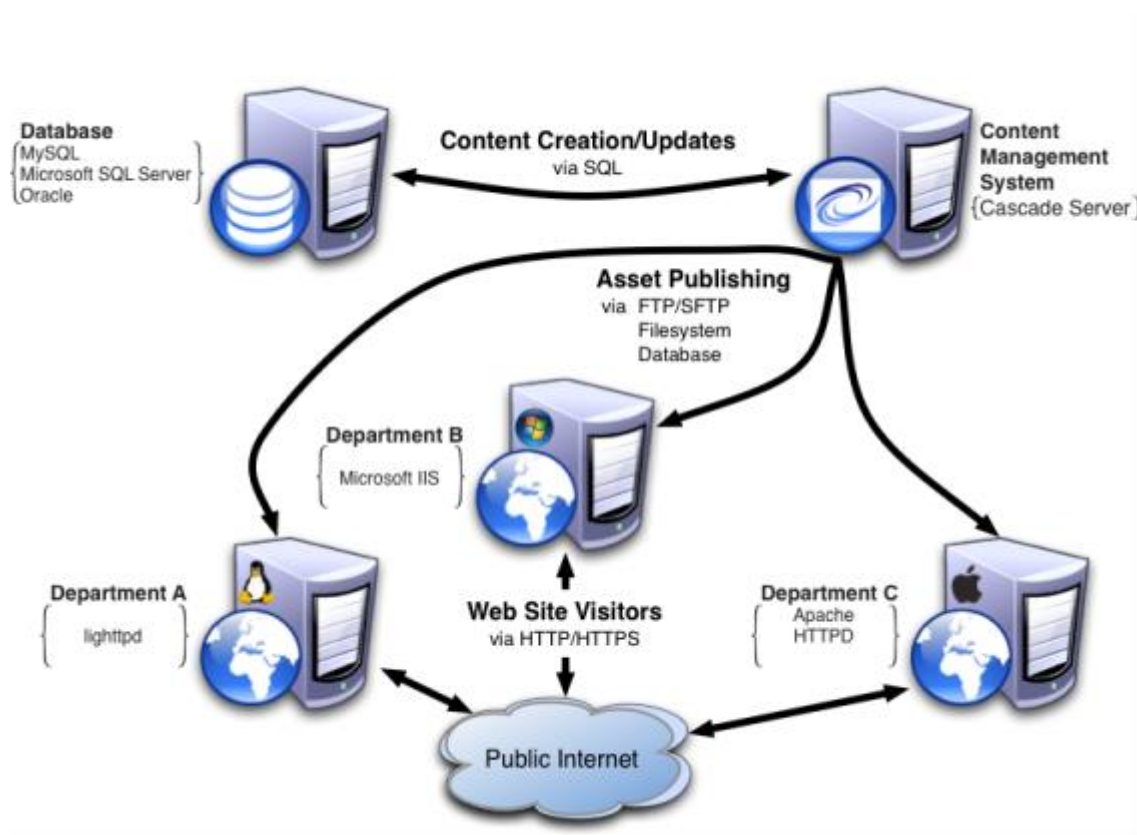
I el darrer element del diagrama de desplegament que considerarem és el **component** que representa els executables, llibreries i altres mòduls d'aquesta mena.

En aquesta UF no utilitzarem els diagrames de desplegament sinó que treballarem amb **diagrames d'arquitectura** que no formen part de l'especificació UML però ens permetran treballar d'una manera més intuïtiva i visual per a descriure els elements de hardware i software que intervenen en el sistema.

Per a fer el diagrama d'arquitectura utilitzarem logos i pictogrames que ens permetran, amb un cop d'ull, conèixer la composició i distribució del sistema.

Exemples





EXERCICI. Fes el diagrama d'arquitectura del sistema informàtic que fas servir a l'escola

EXERCICI. Fes el diagrama d'arquitectura d'una app per a intercanviar llibres

EXERCICI. Fes un diagrama d'arquitectura que sigui vàlid per a un sistema de compra d'entrades online

EXERCICI. Volem desenvolupar un sistema per a controlar un hort domèstic. No es tracta d'un hort urbà d'aquells que podem tenir en un balcó sinó d'un hort per a un jardí d'una superfície entre 20 i 50 m².

El sistema ha de permetre que a diferents zones del jardí puguem posar un dispositiu amb sensors de temperatura, d'humitat i d'intensitat lumínica. Si el terreny està molt sec i la temperatura no és massa freda s'ha d'engegar automàticament el sistema de reg. El reg s'aturarà quan la humitat torni a un llindar que ha fixat l'usuari.

Si la intensitat lumínica és alta entendrem que fa molt de sol i s'ha d'estendre un tendal de forma automàtica. Quan la intensitat lumínica torni a estabilitzar-se es retirarà el tendal també de forma automàtica.

Tots els paràmetres, els llindars mínims i màxims i les accions que cal realitzar en cada cas seran programables per l'usuari a través del seu smartphone o des de l'ordinador de sobretaula que té a la feina.

Cada vegada que el sistema automàtic activa una acció (regar, aturar el reg, estendre el tendal, recollir el tendal,...) aquesta acció ha de quedar enregistrada en un log que després es podrà consultar des de l'smartphone o l'ordinador.

També es vol que cada vegada que s'activa una acció l'usuari rebi una notificació al seu correu electrònic.

Hi ha d'haver un sistema de log in que protegeixi l'accés a la gestió.

DIAGRAMA DE CLASSES

Conceptes

Classe. Una classe és la representació abstracta d'un tipus d'entitat. Poden ser **tangibles** (coses, éssers vius, ...) **no tangibles** (conceptes, idees,...).

Món real: Persones, Cotxes, Agendes, Mobles, Aficions, Desitjos,...

Món informàtic: Icones, Menús, Finestres, Processos, Representació informàtica d'una persona, Representació informàtica d'un cotxe,...

Per a fer un bon desenvolupament és imprescindible identificar i representar correctament les diferents classes que intervenen en el projecte

Una classe es defineix a partir de:

- **Propietats** o **atributs**. Caracteritzen la classe. Del conjunt de valors que tenen les propietats en un moment determinat en diem **estat**.
- **Mètodes**. Accions que pot realitzar. Defineixen el comportament, la funcionalitat i les interaccions de la classe.
- **Events**. Situacions externes o internes davant les quals la classe reacciona. Són els esdeveniments que la classe és capaç de reconèixer quan passen. La major part dels events **es produeixen asíncronament** com els clics amb el ratolí però també poden ser programats (síncrons) com els ticks d'un rellotge (timer).

Objecte. Un objecte és una **instància** d'una classe. Un objecte té:

- **Estat**. L'estat d'un objecte el determina el conjunt de valors que tenen les seves propietats en un instant determinat.
- **Comportament**. Les accions que realitza.

- **Identitat.** Cada objecte té un identificador únic.

Els objectes **interactuen** amb:

- **Altres objectes.**
- **L'usuari.**
- **Processos locals** (Aplicacions o serveis locals).
- **Processos remots** (Client/Servidor, P2P via sockets,...).
- **El sistema operatiu.**

i ho fan enviant-se **missatges** els uns als altres. L'objecte s'assabenta que li arriba un missatge perquè **es «dispara» un dels events** que reconeix.

EXERCICI. Fes una classe per a representar habitacions d'hotel

- Enumera les propietats que cal considerar de cada habitació. Per a cada propietat especifica quins tipus de valors pot tenir aquesta propietat (numèric, text, data, ...)
- Enumera els mètodes que es podran fer amb cada habitació i quina és la informació que caldrà passar per a que l'acció es pugui dur a terme

Representació d'una classe en UML

nom de la classe
propietats
mètodes
events

La classe es representa amb un quadrat dividit en 3 seccions:

- Nom
- Propietats
- Mètodes
- Events (a Modelio són els Signal)

Entre els mètodes són necessaris almenys un **constructor** i un **destructor**.

Quan especifiquem tant les propietats com els mètodes cal indicar-ne la **visibilitat**, és a dir, qui pot accedir-hi i qui no. S'indica amb un signe que es posa al davant del nom de la propietat o mètode

+ (public). Tothom hi pot accedir.

- (private). Només s'hi pot accedir des dins de la pròpia classe.

(protected). Només s'hi pot accedir des de dins de la pròpia classe o des d'altres classes que s'hagin construït com a **subclasses** d'aquesta. *Per exemple, una classe Mamífers i una classe Rèptils serien subclasses d'una superclasse Animals.*

~ (packed). Per tal de fer els desenvolupaments més modulars és interessant crear paquets de classes que es puguin reutilitzar en projectes diferents. Les propietats i mètodes que definim com a **packed** son aquelles que volem que siguin visibles per a qualsevol classe continguda en el paquet però que no volem que siguin públiques.

EXERCICI - Ordena els diferents tipus de visibilitat de més visible a menys visible.

EXERCICI – Afegeix la visibilitat a les propietats i mètodes de la classe Habitacions que has fet abans.

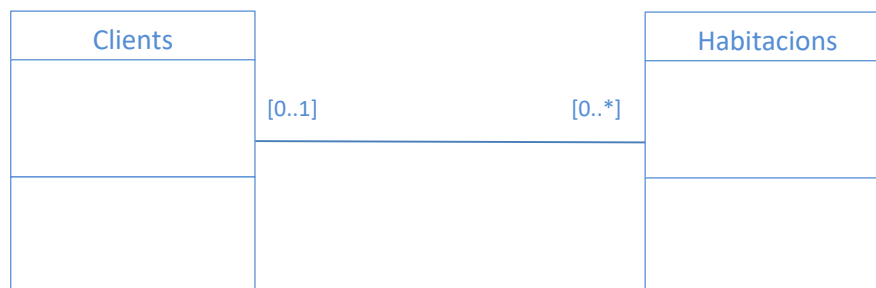
IMPORTANT!!!! Si som estrictes seguint els preceptes de la Programació Orientada a Objectes totes les propietats haurien de ser privades i accedir-hi a través de mètodes Get i Set.

EXERCICI. Fes una classe per a representar els clients d'un hotel

Relacions entre classes

Associació. Una associació entre 2 classes indica que els objectes d'una classe tenen relació amb els de l'altre. En el nostre cas Clients i Habitacions s'associen.

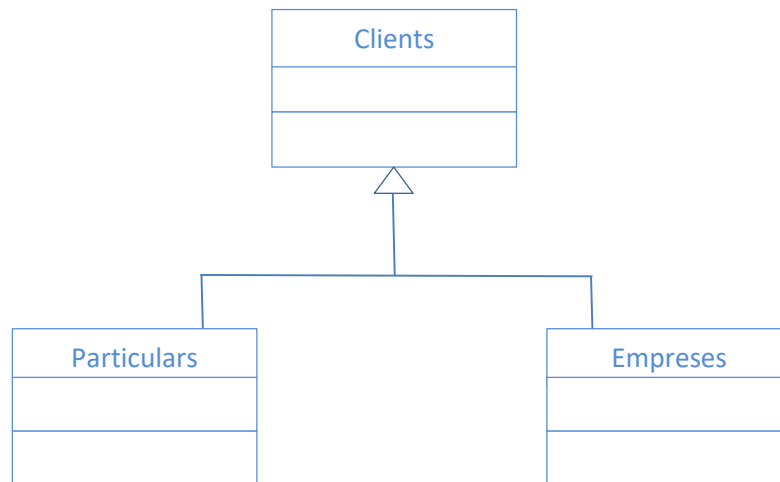
Gràficament s'indica amb **una línia contínua** que uneix les classes. També s'hi afegeix la **multiplicitat**.



Les associacions poden ser binàries, ternàries,...

Generalització. Les relacions de generalització es donen entre classes que tenen una relació d'**herència**, és a dir, que una inclou l'altra. S'indica amb una fletxa des de la classe que hereta (**subclasse**) a la classe global (**superclasse**).

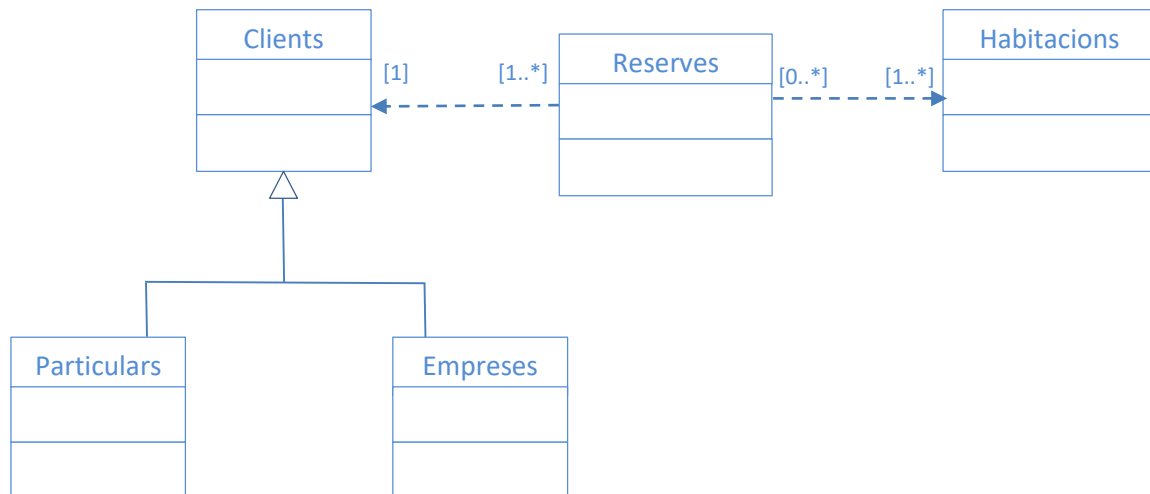
Agafem l'exercici que estem fent de l'hotel i considerarem que els clients poden ser particulars o empreses.



EXERCICI. Fes les tres classes de clients indicades en el diagrama anterior

Dependència. Les relacions de dependència es donen quan un objecte d'una classe necessita un objecte d'una altra. Es representa amb **una fletxa discontinua** que surt de la classe dependent i arriba a la classe de la que es depèn.

Agafem l'exercici que estem fent de l'hotel i afegim una classe **Reserves**. Aquesta classe depèn dels Clients i de les Habitacions perquè no entendríem una reserva sense client ni habitació.



EXERCICI. Fes la classe Reserves

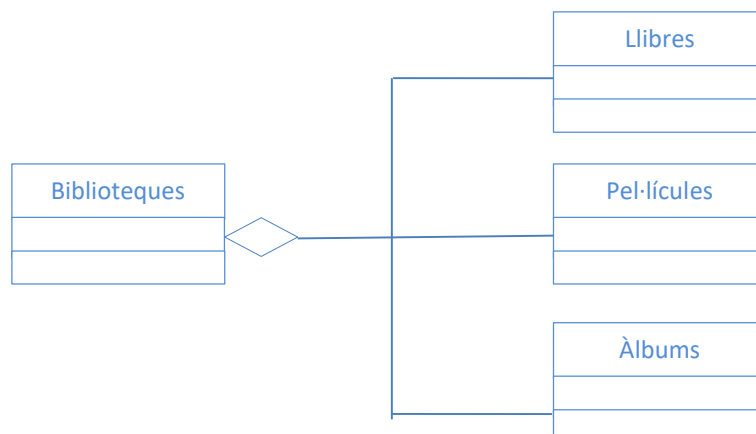
Agregació. Una relació d'agregació és un tipus especial d'associació que s'estableix quan els objectes d'una classe es fan ajuntant objectes d'altres classes però aquests components poden existir sense que existeixi l'objecte que els agrupa.

En el diagrama s'indica amb un rombe de color blanc

Exemples



Un determinat contacte pot existir sense necessitat d'estar associat a una llibreta d'adreces

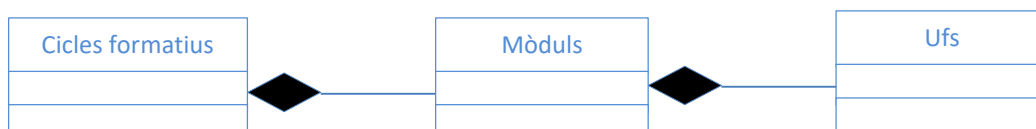


Un determinat llibre, pel·lícula o àlbum de música pot existir sense necessitat d'estar associat a una biblioteca. En canvi una biblioteca no existeix si no té associat almenys un llibre, una pel·lícula o àlbum.

Composició. Una relació de composició és un tipus especial d'associació que s'estableix quan els objectes d'una classe es fan ajuntant objectes d'altres classes però, en aquest cas, si s'elimina l'objecte agregat també desapareixen els objectes que el componen.

En el diagrama s'indica amb un rombe de color negre

Exemple



Una determinada UF no existeix si no està associada a un mòdul, si s'elimina un mòdul també en desapareixen les UFs associades. El mateix passa amb el mòduls respecte els cicles.

Representació dels events. En el diagrama de classes els events els posarem en una tercera secció que afegirem a les propietats i els mètodes. L'UML parla de **signals** incloent-hi tant els **events** com les **excepcions**. Nosaltres per a simplificar parlarem **només d'events**.

EXERCICI. Fes el diagrama de classes per a dissenyar el sistema de control i termòstat d'una calefacció o aire condicionat.

EXERCICI. Fes el diagrama de classes per a dissenyar una aplicació de tipus Alarma com les de bona part dels smartphones.

EXERCICI. Fes el diagrama de classes corresponent als diferents usuaris del sistema informàtic d'una escola. Has de tenir en compte que hi pot haver:

- directius
- professors
- personal d'administració i serveis
- administradors i tècnics del sistema informàtic
- alumnes

EXERCICI. Fes el diagrama de classes per a dissenyar una aplicació de missatgeria semblant a whatsapp.

EXERCICI. Fes el diagrama de classes per a dissenyar una aplicació amb la que es puguin gestionar les matrícules i resultats acadèmics dels alumnes de Cicles Formatius d'una escola.

EXERCICI. Fes el diagrama de classes per a dissenyar una aplicació de gestió d'una biblioteca pública

DIAGRAMA D'ACTIVITATS

Definició i components

El **diagrama d'activitats** representa la descomposició dels casos d'ús en un seguit d'activitats que ben combinades entre elles oferiran la funcionalitat esperada.

Per exemple, un cas d'ús «**Pagar comanda**» que s'ofereix a un actor «**Client**» es pot destriar dins el sistema en les activitats següents:

- Triar forma de pagament (PayPal o Targeta)

Si es tria PayPal

- Iniciar sessió a PayPal
- Verificar compte PayPal
- Realitzar cobrament
- Confirmar o rebutjar operació

Si es tria Targeta

- Introduir dades targeta
- Verificar dades targeta
- Generar i enviar codi de confirmació
- Verificar codi de confirmació
- Realitzar cobrament
- Confirmar o rebutjar operació

Inici. És el punt on s'inicien les activitats. Es representa amb un cercle pintat de color negre



Final. És el punt on finalitzen les activitats. Es representa com l'inicial però amb una corona que l'envolta

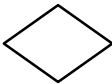


Activitat. Es representa amb un rectangle de cantonades arrodonides i a dins s'hi posa una etiqueta que identifica l'activitat

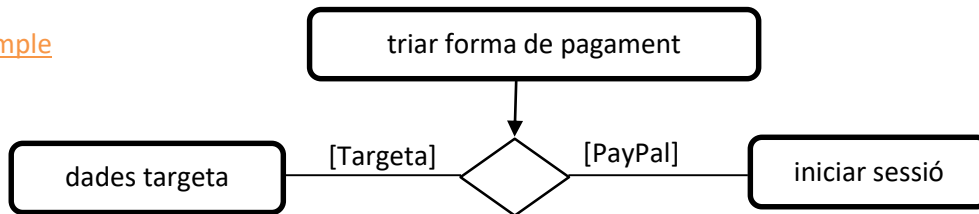


Les **transicions** d'una activitat a una altra s'indiquen amb una fletxa

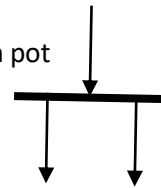


Decisió. Es representa amb un rombe  del que surten diverses opcions. A cada opció se li assigna una **condició** que es representa amb una etiqueta dins de claudàtors [etiqueta]

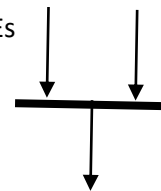
Exemple



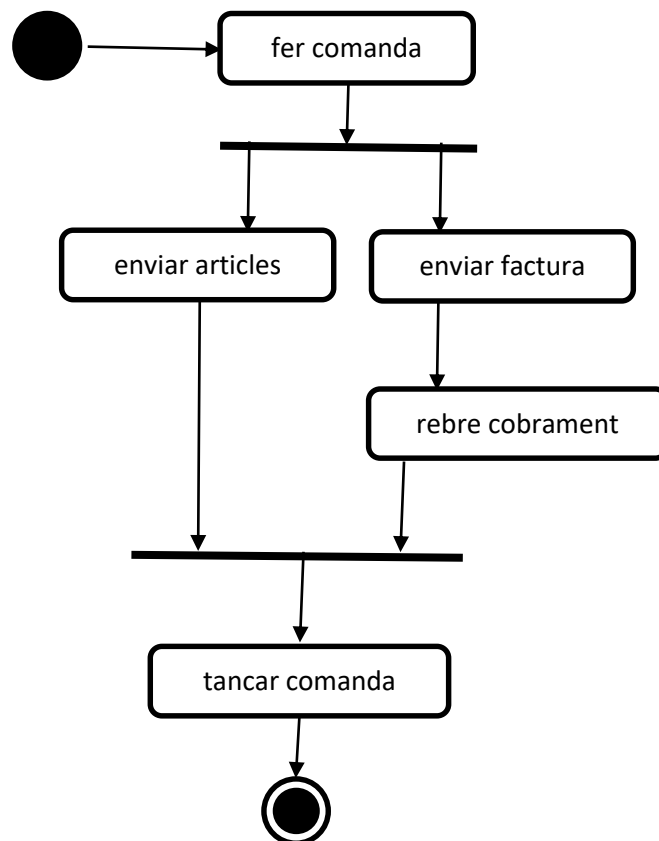
Fork (bifurcació). Representa la possibilitat de triar entre diverses activitats. Se'n pot triar més d'una i executar-les en paral·lel. Es representa amb una fletxa que arriba a una barra horizontal i d'aquesta en surten tantes fletxes com possibilitats.



Join (unió). Representa la confluència de diverses activitats en una única. Es representa de manera inversa al **fork**.



Exemple



EXERCICI. Fes el diagrama d'activitats d'un sistema de compra online d'entrades de teatre

EXERCICI. Fes el diagrama d'activitats del procés de matriculació d'un alumne de DAM tenint en compte el sistema de preinscripcions

EXERCICI. Fes el diagrama d'activitats del sistema de préstec d'una biblioteca pública en que el temps màxim de préstec és 10 dies i el màxim de préstecs simultanis és 3. Si se supera el temps màxim hi ha una penalització i durant un mes no es poden fer préstecs.

EXERCICI. Fes el diagrama d'activitats del procés d'una comanda d'Amazon des del punt de vista del client

EXERCICI. Fes el diagrama d'activitats del procés d'una comanda d'Amazon des del punt de vista del venedor

EXERCICI. Fes el diagrama d'activitats d'una màquina de zona blava d'aparcament


DIAGRAMA D'ESTATS

Definició i components

El **diagrama d'estats** mostra les diferents situacions en que es pot trobar un objecte durant el seu cicle de vida i quines són les accions o esdeveniments que provoquen la transició d'un estat a un altre.

Estat inicial. És l'estat en que es troba l'objecte quan se'n crea una instància, és a dir, en el moment que «neix». Es representa amb un cercle pintat de color negre ●

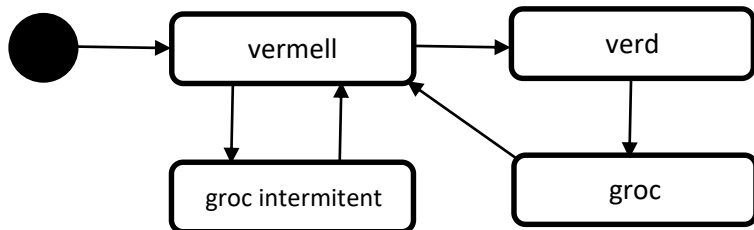
Estat final. És el darrer estat en que es troba l'objecte quan arriba al final del seu cicle de vida. Es representa com l'inicial però amb una corona que l'envolta ○

Estat. És qualsevol altre estat intermedi. Es representa amb un rectangle de cantonades arrodonides i a dins s'hi posa una etiqueta que identifica l'estat 

Les **transicions** d'un estat a un altre s'indiquen amb una fletxa →

Exemples

Semàfor de cotxes



En aquest cas podem considerar que no hi ha estat final si pensem que sempre està en marxa. Això no obstant podríem pensar que l'estat final és si el desactiven o s'espalla i en aquest cas hauríem de posar fletxes de transició des de qualsevol estat a l'estat final ja que la desactivació o avaria es pot produir en qualsevol moment.

EXERCICI. Fes el diagrama d'estat d'un semàfor de vianants

EXERCICI. Fes el diagrama d'estat d'un alumne de DAM tenint en compte el sistema de preinscripcions i matriculacions

EXERCICI. Fes el diagrama d'estat del sistema operatiu del teu ordinador portàtil

EXERCICI. Fes el diagrama d'estat d'un smartphone

EXERCICI. Fes el diagrama d'estat d'una comanda d'Amazon

EXERCICI. Fes el diagrama d'estat d'un llibre d'una biblioteca pública

EXERCICI. Fes el diagrama d'estat d'un jugador d'handbol durant un partit

DIAGRAMA DE SEQÜÈNCIA

Definició i components

El **diagrama de seqüència** descriu per a un cas d'ús el cicle de vida dels objectes, les interaccions entre els diferents objectes i com es produeixen en el temps.

Actor. És un dels actors del diagrama de casos d'ús



Objecte. Representa una instància d'una de les classes del diagrama de classes i es representa dins d'un rectangle.

nom classe

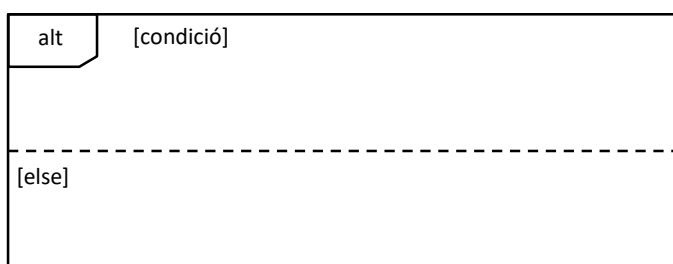
Línia de vida. És una línia discontinua vertical que va baixant des de l'actor o objecte indicant el seu temps de vida.

Activació. L'activació mostra el temps que necessita un actor o objecte per a realitzar una tasca determinada. Es representa amb un rectangle vertical que se sobreposa a la línia de vida.

Missatges. Els missatges indiquen la comunicació entre els objectes o entre l'actor i els objectes. Es representen amb fletxes de l'emissor al receptor. La forma de la fletxa i si la línia és contínua o discontinua determina el tipus de missatge.

----->	Missatge de creació d'objecte
—————>	Missatge síncron. L'execució espera resposta.
—————>	Missatge asíncron. L'execució segueix i la resposta ja arribarà en algun moment que no es pot preveure.
----->	Missatge de retorn

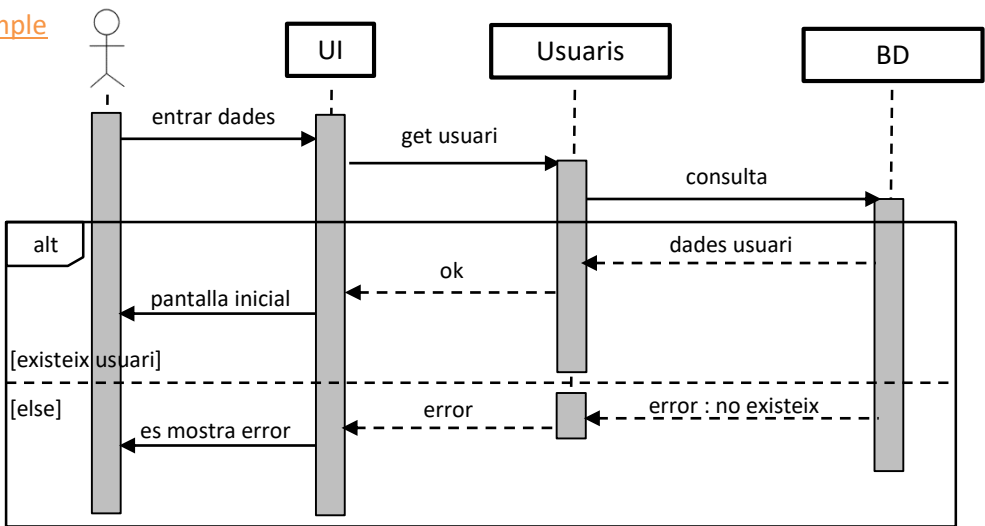
Alternativa. Mostra les situacions en que hi ha diferents alternatives i el cicle de vida dels objectes poden anar per una o per l'altra. Es representa posant dins d'un rectangle totes les alternatives separades per una línia discontinua. A la cantonada superior esquerra hi ha l'etiqueta **alt**.



Loop. Representa els processos iteratius.



Exemple



EXERCICI. Fes el diagrama de seqüència del cas d'ús «log in» en que la interfície primer demana el nom d'usuari i, si existeix, demana el password
EXERCICI. Fes el diagrama de seqüència d'un cas «Registre d'usuari»
EXERCICI. Fes el diagrama de seqüència d'un cas «Donar-se de baixa com a usuari enregistrat»
EXERCICI. Fes el diagrama de seqüència del cas «Reservar habitació d'hotel» en un web semblant a Kayak

REFERÈNCIES

<https://www.youtube.com/watch?v=1I979cB4QWQ>

<https://www.youtube.com/watch?v=JioEGJllg88>

<https://www.youtube.com/watch?v=orvAkFFWo5o>

<https://www.youtube.com/watch?v=Q1kH7XKxK5I>

https://www.youtube.com/watch?v=jN8cn_5y_xE

<https://www.youtube.com/watch?v=y2eCjadS8x8>

<https://www.youtube.com/playlist?list=PLUoebdZqEHTxNC7hWPPwLsBmWl0KEhZOd>

<https://www.youtube.com/user/Udacity/search?query=uml>