0. Control de Cambios:

V1.
Tema 1. Resumen + poner los enunciados con soluciones ejercicios 1-4.

V2.
Puntualizado el ejercicio del hospital
Añadido el capítulo 2. Diagramas de Transición de Estados.

V3.
Comentarios en el ejercicio 4. DTE
Añadido el capítulo 3. Diagramas de Flujo de Datos.

V4.
Capitulo DEst y DO.

V5.
Creación tema pruebas y ejercicios caja negra y transparente.
Creación de tema UML.

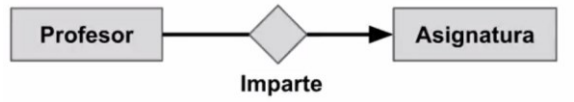
V6.
Finalización teoría de tema UML.

1. Entidad Relación (DER/ERD)

DER→Diagrama de Entidad/Relación | ERD→Entity Relation Diagram

Resumen Teoría

Tipo de modelado **estático** que nos permite establecer estructuras de información.



El profesor imparte asignatura

Una canción pertenece a un álbum

- Multiplicidad (Cardinalidad)

A: ¿Puede existir alguna "Entidad2" que no esté asociado a ninguna "Entidad1"?

→SI: A=0 →No: ¿Hay mínimo? NO: A=1 SI: A=min.

B: ¿Puede existir alguna "Entidad2" que esté asociada a varias "Entidad1"?

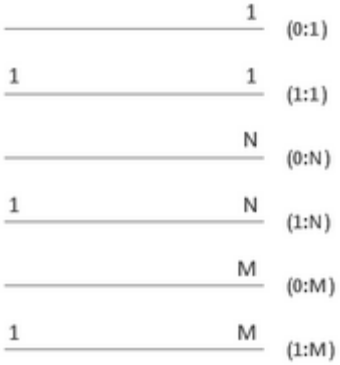
→SI: B=1 →No: ¿Hay máximo? NO: B=N SI: B=máx.

C: ¿Puede existir alguna "Entidad1" que no esté asociado a ninguna "Entidad2"?

→SI: C=0 →No: ¿Hay mínimo? NO: C=1 SI: C=min.

D: ¿Puede existir alguna "Entidad1" que esté asociada a varias "Entidad2"?

→SI: D=1 →No: ¿Hay máximo? NO: D=N SI: D=máx.



Con cardinalidades 0:x se puede omitir el 0: y dejar sólo x.

Equivalencia de cardinalidades

En situaciones en vez de ver la relación con número se ve con símbolos; las equivalencias son:

Cero o más (**0:N**)

Uno o más (**1:N**)

Uno y sólo uno (**1**)

Cero o uno (**0 ~ 1**)

- Diccionario de datos (DD)

  Nombre: xxxxxxxx

  Estructura: algo parecido a reglas BNF

Lista de Elementos →{XXXX}num
Nombre: **Libro**
Estructura**: {Capítulos}**

Lista de Elementos →{XXXX}num
Nombre: **Matrimonio**
Estructura**: {Personas}²**

Siempre en Singular

Composición → XXX+XXX
Nombre: **Mesa**
Estructura**: Superficie+{Patas}**

Opción → [XXX | XXX]
Nombre: **Empleado**
Estructura**: [Empleado | Jefe]**

Lista de Elementos →{XXXX}num
Composición → XXX+XXX
Opción → [XXX | XXX]

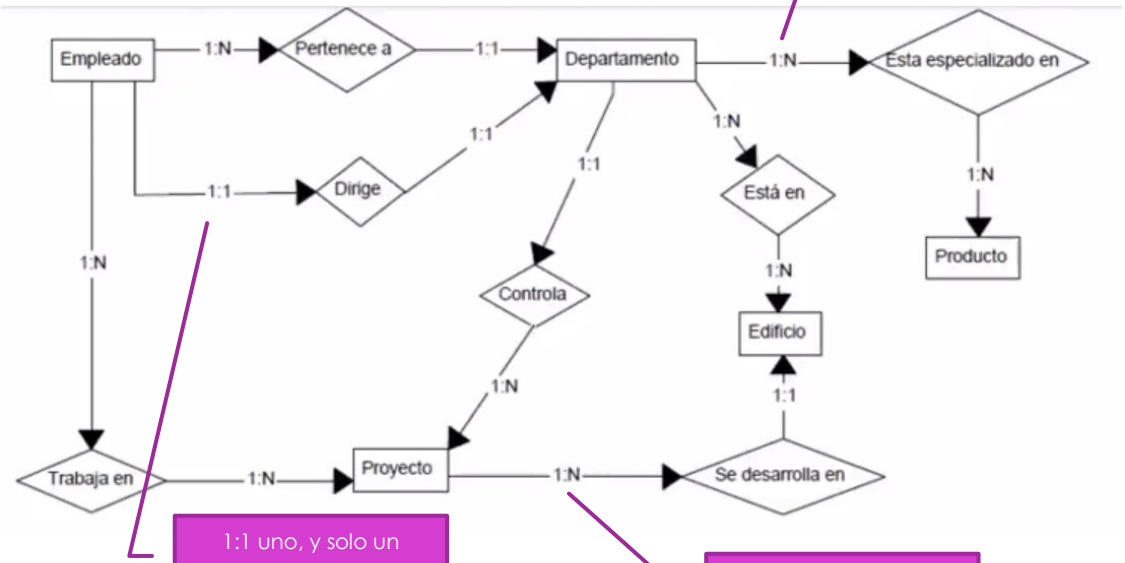
Ejercicio1. DER. Empresa

ENUNCIADO

Una empresa se organiza en departamentos. Cada departamento dispone de cierto número de empleados y de un director. Los departamentos se especializan en uno o varios productos, aunque puede darse la situación de que más de un departamento esté cualificado para construir un determinado producto. Por otro lado, cada departamento controla cierto número de proyectos. Un empleado esté asignado a un departamento, aunque puede trabajar en varios proyectos controlados por otros departamentos. Por último, la empresa dispone de varias sedes. Los departamentos pueden estar repartidos en distintos edificios. Sin embargo, los proyectos se desarrollan exclusivamente en una sede.

SOLUCIÓN

1:N. | 1 = no hay mínimo en productos en 1 departamento. N = varios departamentos pueden colaborar para 1 producto.



1:1 uno, y solo un empleado puede dirigir (entenderíamos que se refiere a que un solo empleado puede ser director)

1:N En un edificio, pueden haber varios proyectos.

Entidades: Empleado, Departamento, Proyecto, Edificio y Producto.

No son entidades: Director no puede ser entidad al ya de por sí ser empleado.
Sede, como es un edificio, se sobreentiende que ya está incluido.

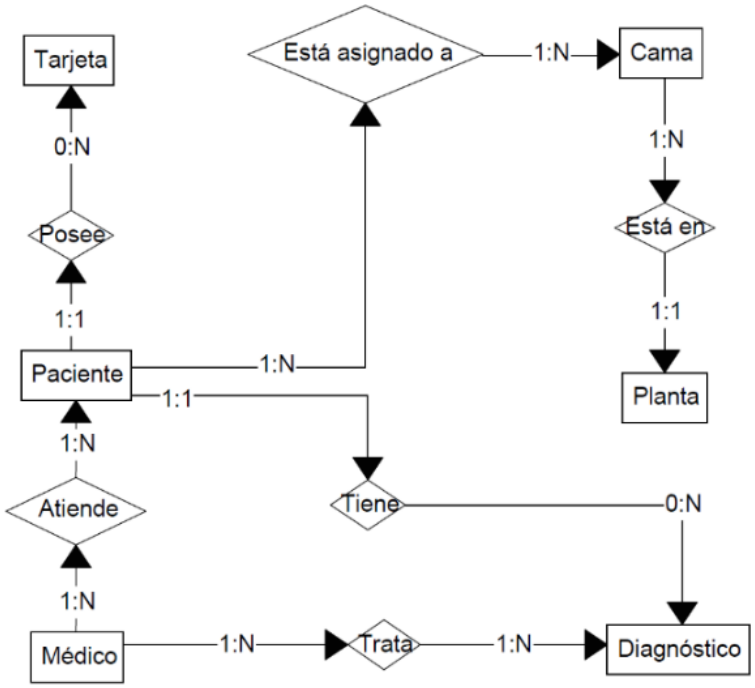
Ejercicio2. DER. Hospital

ENUNCIADO

- En un centro hospitalario se desea informatizar parte de la gestión relativa a sus pacientes. El sistema a construir deberá contemplar las siguientes cuestiones:
  - Un paciente estará asignado a una cama determinada de una planta del hospital, pudiendo estar a lo largo del tiempo de ingreso en diferentes camas y plantas.
  - Para cada paciente se entregarán hasta un máximo de 4 tarjetas de visita. Estas tarjetas servirán para que familiares y amigos del paciente le visiten durante su convalecencia.
  - A un paciente le pueden atender diferentes médicos.
  - Un paciente puede tener distintos diagnósticos de enfermedad.
  - Un médico puede tratar diferentes diagnósticos y viceversa.
- **Analice el sistema mediante la notación DER (Diagrama Entidad Relación).**

SOLUCION

1. Un paciente estará asignado a una cama determinada de una planta del hospital, pudiendo estar a lo largo del tiempo de ingreso en diferentes camas y plantas.
2. Para cada paciente se entregarán hasta un máximo de 4 tarjetas de visita. Estas tarjetas servirán para que familiares y amigos del paciente le visiten durante su convalecencia.
3. A un paciente le pueden atender diferentes médicos.
4. Un paciente puede tener distintos diagnósticos de enfermedad.
5. Un médico puede tratar diferentes diagnósticos y viceversa.



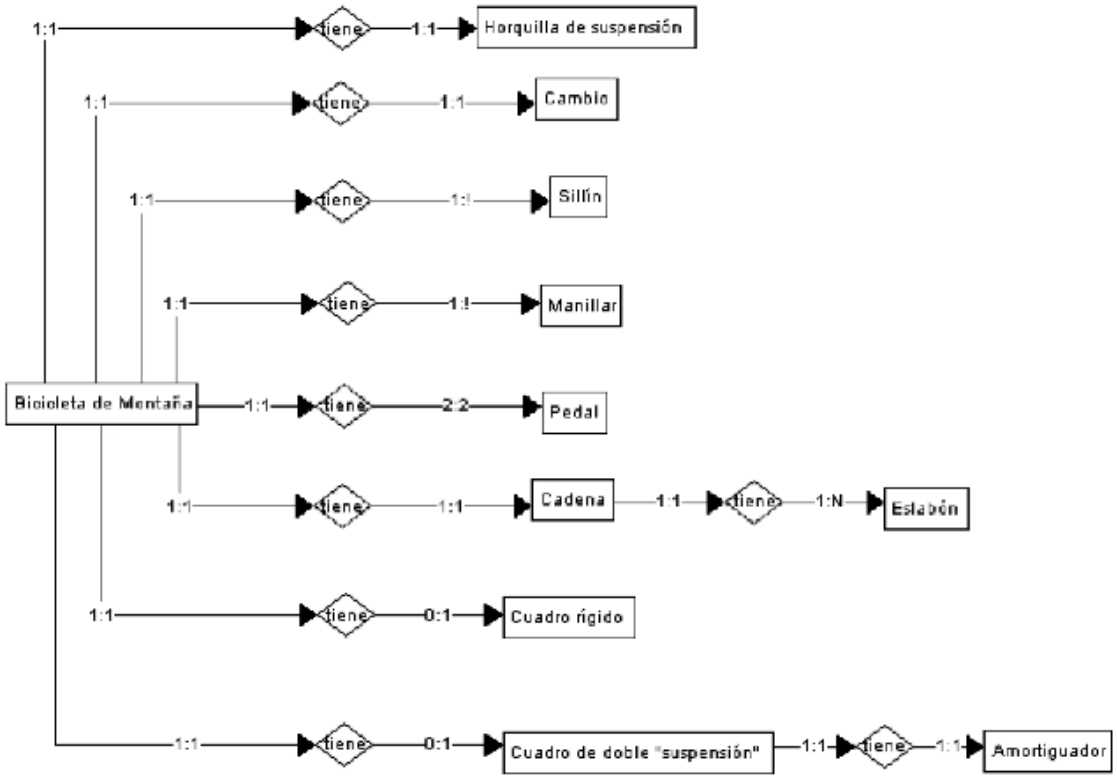
Analizamos el ejercicio (ver la realidad) de forma no ambigua.

Ejercicio3. DER. Bicicleta

ENUNCIADO

- Actualmente, las bicicletas de montaña están compuestas, entre otras piezas, por un manillar, un sistema de cambio, un sillín, una horquilla de suspensión, dos pedales, una cadena y un cuadro. La cadena está formada por un conjunto de eslabones y el cuadro, dependiendo de si la bicicleta es de "doble suspensión" o no, dispone de un amortiguador.
- En la asignatura se estudian dos notaciones para modelar datos, que en ocasiones pueden considerarse equivalentes: los diccionarios de datos y los diagramas entidad relación. **Modele el enunciado anterior utilizando ambas notaciones. En el diccionario de datos omita el campo "Utilidad" y las descripciones de los componentes de la bicicleta (limítese a cumplimentar los campos "Nombre" y "Estructura").**

SOLUCION



Nombre: Bicicleta de montaña
Estructura: Manillar + Cambio + Sillín + Horquilla de suspensión + {Pedal}^{2} + Cadena + [Cuadro rígido | Cuadro con suspensión]

Nombre: Cadena
Estructura: {Eslabón}

Nombre: Cuadro con suspensión
Estructura: Amortiguador

Ejercicio4. DD. Varios

ENUNCIADO

- Un libro tiene varios autores, varios capítulos y un título.
- Una persona tiene dos brazos, dos piernas y una cabeza.
- Un animal puede ser macho, hembra o hermafrodita.

SOLUCION

Un libro tiene varios autores , varios capitulos y un titulo → Nombre: **Libro**
Estructura: **{Capitulo} + {Autor} + Titulo**

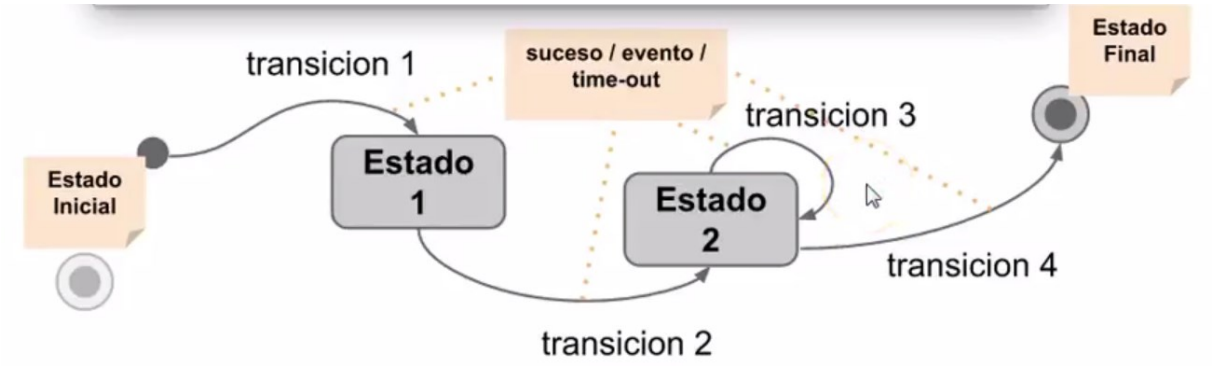
Una persona tiene dos brazos, dos piernas y una cabeza → Nombre: **Persona**
Estructura: **$\{Brazo\}^2 + \{Pierna\}^2 + Cabeza$**

Una animal puede ser macho, hembra o hermafrodita → Nombre: **Animal**
Estructura: **[Macho | Hembra | Hermafrodita]**

2. Diagramas de Transición de Estados (DTE)

DER→Diagrama de Entidad/Relación | ERD→Entity Relation

Resumen Teoría



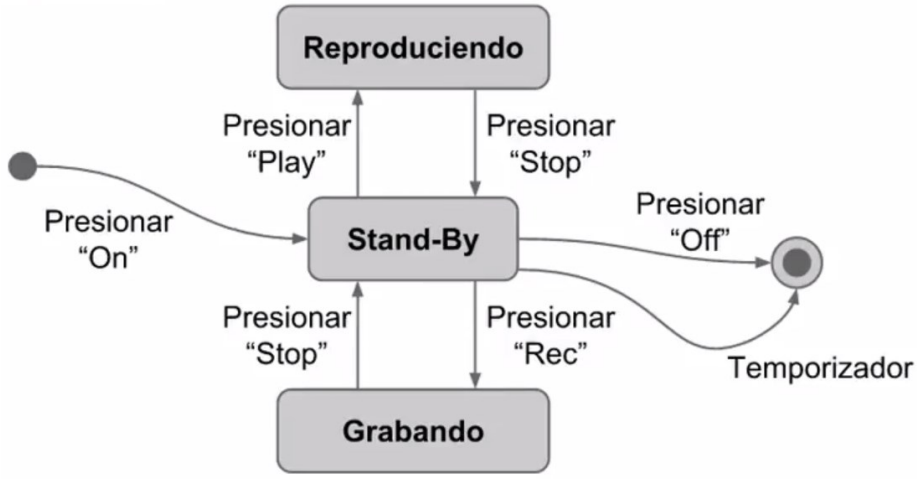
Tipo de modelado **dinámico** que se suele usar en la parte de análisis.
De todos sus estados posibles, sólo es importante resaltar aquellos que tienen cierta transcendencia desde un punto de vista funcional.
Las flechas mejor usando rectas que con "curvitas". El estado, un rectángulo.
Los inicios y fin suelen estar más relacionado con el inicio/apagado de sistema.
Mejor dividir estados que crear un estado con varios estados que se queda siempre en bucle consigo mismo.
Transición: Algo que le sucede al sistema (que pase algo).
Estados: Descripciones de cómo está el sistema.
Para los **estados**: Mejor usar gerundio.

Ejercicio1. DTE. Radio

ENUNCIADO

Hacer un diagrama de transición de estados para modelar el funcionamiento de una grabadora / reproductora de sonido que tiene los botones "On", "Off", "Play", "Rec" y "Stop", con temporizador de auto-apagado.

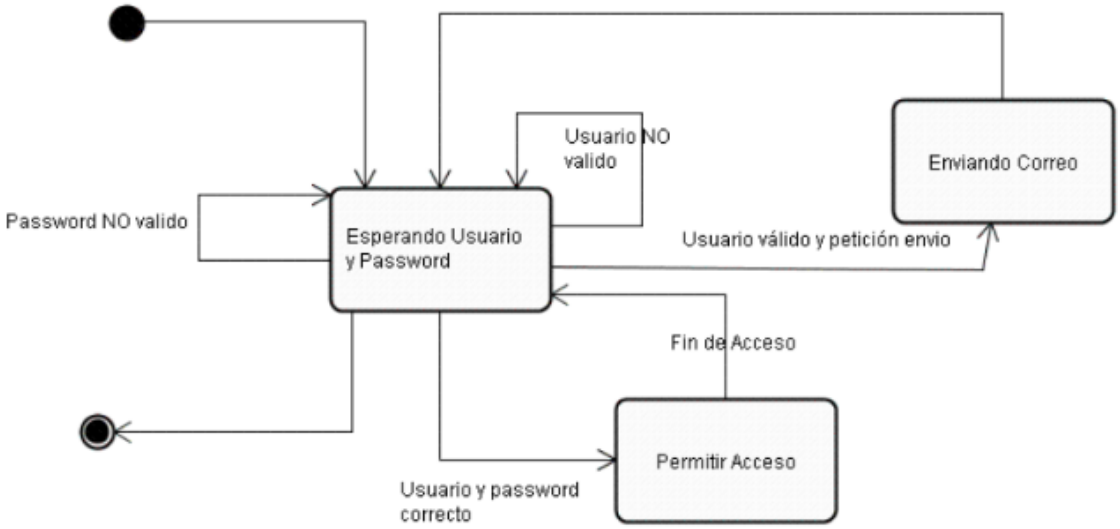
SOLUCIÓN



Ejercicio2. DTE. Usuario/Contraseña

ENUNCIADO

Modele, mediante un diagrama de transición de estados (DTE), el módulo de acceso a un sistema mediante usuario y contraseña. Contemple la posibilidad de enviar por correo electrónico la contraseña a los usuarios válidos que así lo soliciten en caso de olvido.



SOLUCIÓN

El nombre del estado suele ser una descripción de cómo se queda el sistema.

El estado Permitir Acceso sería más correcto como "autenticado".

Los inicios y fin suelen estar más relacionado con el inicio/apagado de sistema.

Enviando el correo… ñeeeee…. sería mejor una acción que permanezca activa hasta que suceda algo que le haga transicionar.

Ejercicio3. DTE. Autocobro supermercado

ENUNCIADO

- Un supermercado desea implantar un sistema de cobro automático, de forma que sean los propios clientes quienes pasen los productos por el lector de código de barras y paguen introduciendo su tarjeta de crédito en una ranura, tras lo que recibirán el comprobante de la compra. El cliente podrá cancelar el proceso en cualquier momento, pero no una vez aceptado el pago.
- **Se pide:**
- **Proponga, mediante lenguaje natural un sistema sencillo que resuelva el sistema deseado.**
  (Intentad primero y después ver la solución antes de seguir con el 2)
- **Realice el diagrama de transición de estados del sistema propuesto.**

SOLUCIÓN

En cada puesto de auto-pago habrá un terminal de punto de venta (TPV) con los siguientes elementos:

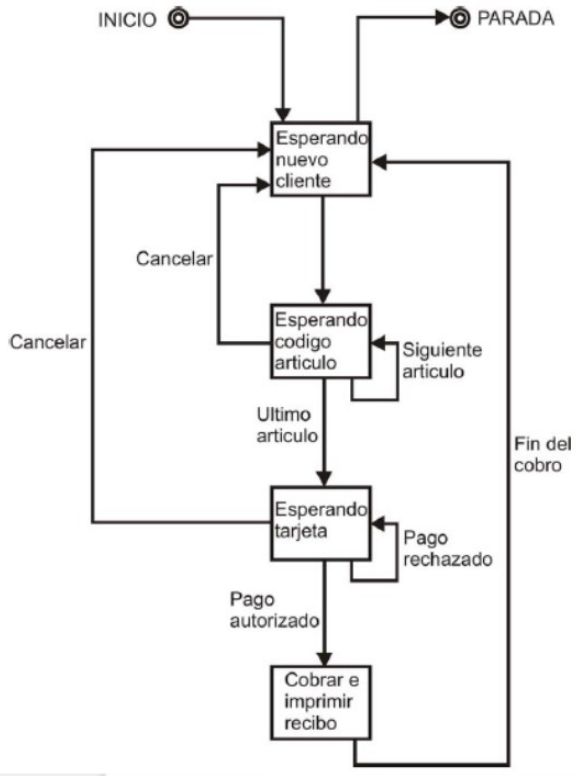
- Pantalla táctil
- Lector de código de barras
- Lector de tarjetas de banda magnética
- Impresora de recibos

Cada TPV estará conectado al sistema que gestiona la base de datos de existencias, referencias y precios de los productos, y gestionará la autorización para el pago con tarjeta de crédito.

Cuando llegue una nuevo cliente deberá pulsar en "cliente nuevo" en la pantalla. A continuación se le indicará que comience a pasar los productos por el lector de códigos de barras. En la pantalla se presentará la lista de productos leídos con su referencia y precio. En cualquier momento el cliente podrá pulsar "cancelar" o "último artículo".

"Cancelar" anula todo el proceso de pago; no se puede cancelar el último artículo que se ha detectado sino que se deberá empezar de nuevo a pasar todos los productos por el lector.

Tras pulsar "último artículo" la pantalla mostrará el mensaje "introduzca tarjeta de crédito". Si el pago es autorizado se imprimirá el recibo y el proceso habrá finalizado. Si no se autoriza el pago con la tarjeta introducida, se mostrará el mensaje "tarjeta no válida", pudiendo el cliente introducir otra tarjeta o cancelar el proceso.



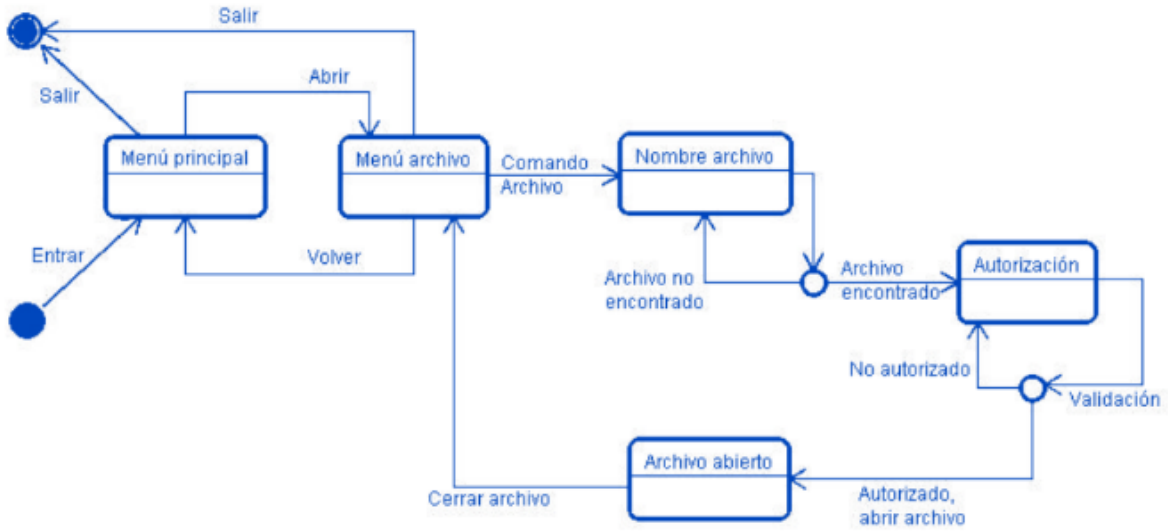
Ejercicio4. DTE. Menú

ENUNCIADO

Una interfaz gráfica tiene un "Menú principal" con un submenú de "Archivo" que, a su vez, tiene el comando de "Abrir archivo". Cada menú tiene una opción para "Salir". El comando "Abrir archivo" tiene un cuadro de texto en el que se puede escribir el nombre (y opcionalmente la ruta) del archivo que se quiere abrir. Supóngase que sólo se puede abrir un archivo cada vez. Si el nombre del archivo es correcto, se pedirá que se escriba una clave para autorizar su apertura y, si es incorrecta, la interfaz vuelve al submenú "Archivo".

Construya un modelo del comportamiento (para el análisis) utilizando un diagrama de transición de estados.

SOLUCIÓN



Que acciones hace el usuario con el sistema; que hace el sistema con lo que hace el usuario.
En los estados; poner la línea en medio…no es incorrecto, no recomendable hacer.
Los microestados (circulito) sin nombre "de transición" no recomendable usar.
Cada "click" un estado.
El ejercicio no tiene mucha coherencia entre resultado-solución.
Lo mejor es, tras hacer nuestro diagrama, explicar con palabras el porqué de nuestras decisiones.

3. Diagramas de Flujo de Datos (DFD)

DER→Diagrama de Entidad/Relación | ERD→Entity Relation

Resumen Teoría

Connotación usada:

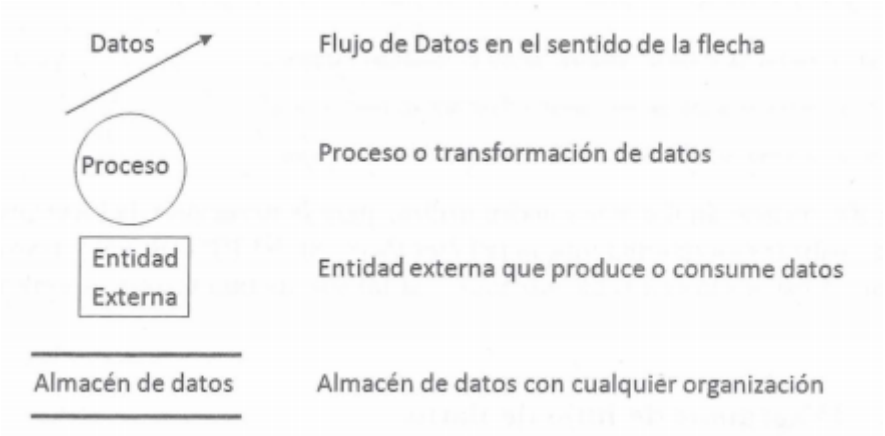
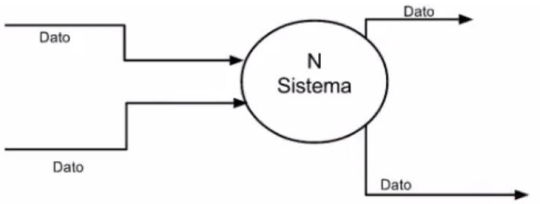
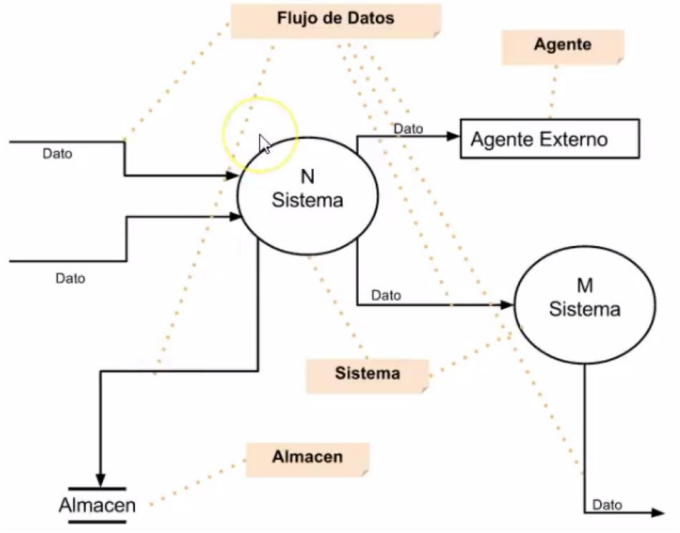


Diagrama **híbrido** (estático y dinámico). Representa sistemas (pelotitas) y el flujo de información que hay entre los sistemas(flechas).



Agente Externo y Almacenes de información no son tan habituales.

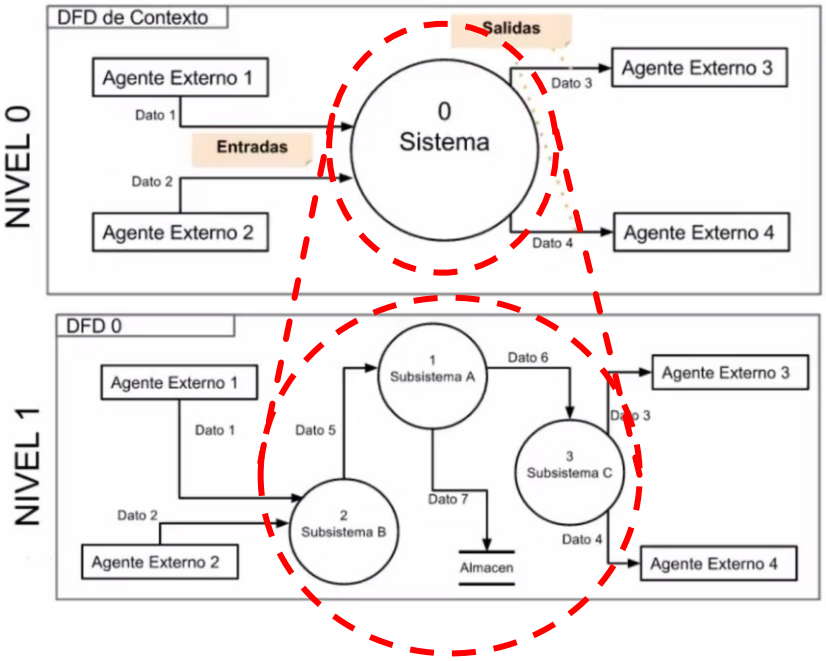


Almacén → sitio donde persiste información. Envía/recibe información.

Sistema → algo con funcionalidad, el que proceso o hace algo con la información

Agentes Externos → No pertenece al sistema, envía/recibe información.

- Se modela jerárquicamente, por niveles.

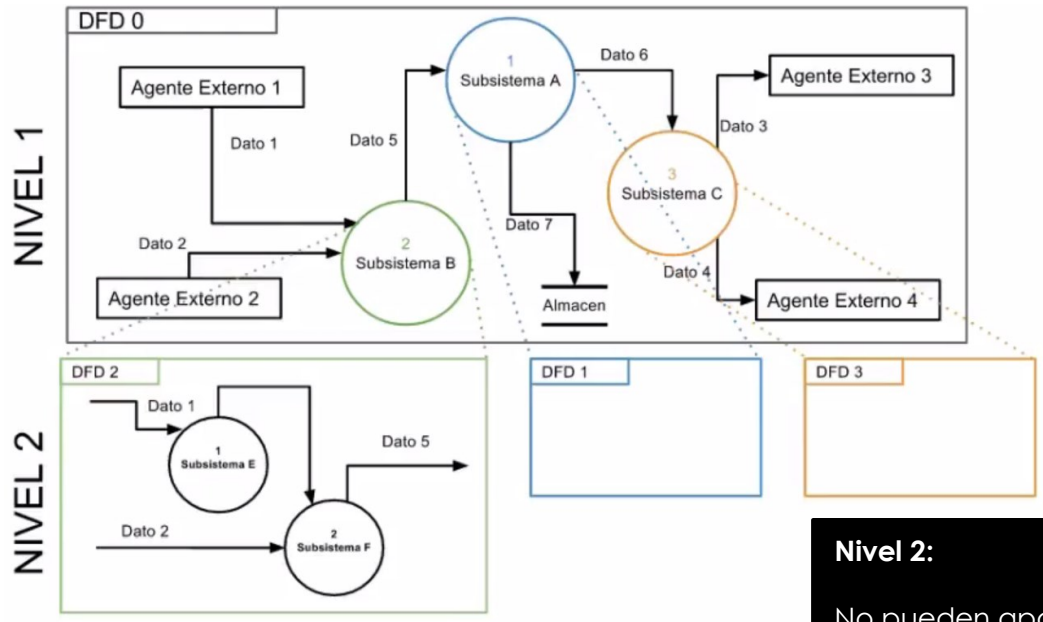


Nivel 0 o DFD de Contexto:

Sólo una pelotita.
No puede aparecer almacenes.

Nivel 1 o DFD 0:

La pelotita del nivel 0
Mismas entradas y salidas que en el nivel 0.
Aparecen los almacenes.
Cada subsistema tendrá un nombre y un numero



Nivel 2:

No pueden aparecer Aaentes Externos.

La parte estática → La estructura de los sistemas
La parte dinámica → La secuencia entre subsistemas

¡Cuanto más se divida, mejor!

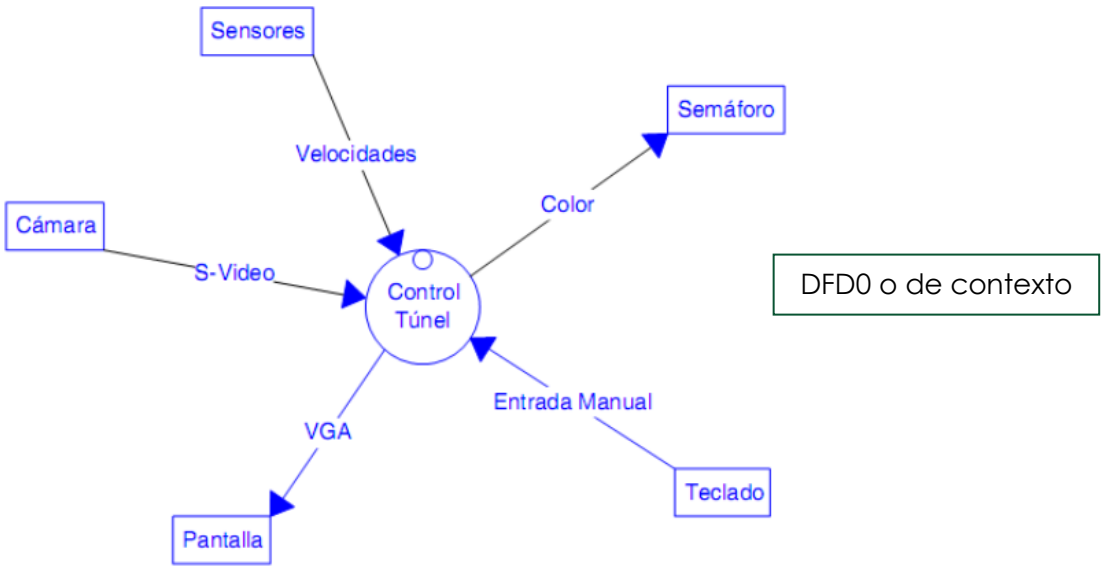
Ejercicio1. DFD. Radio

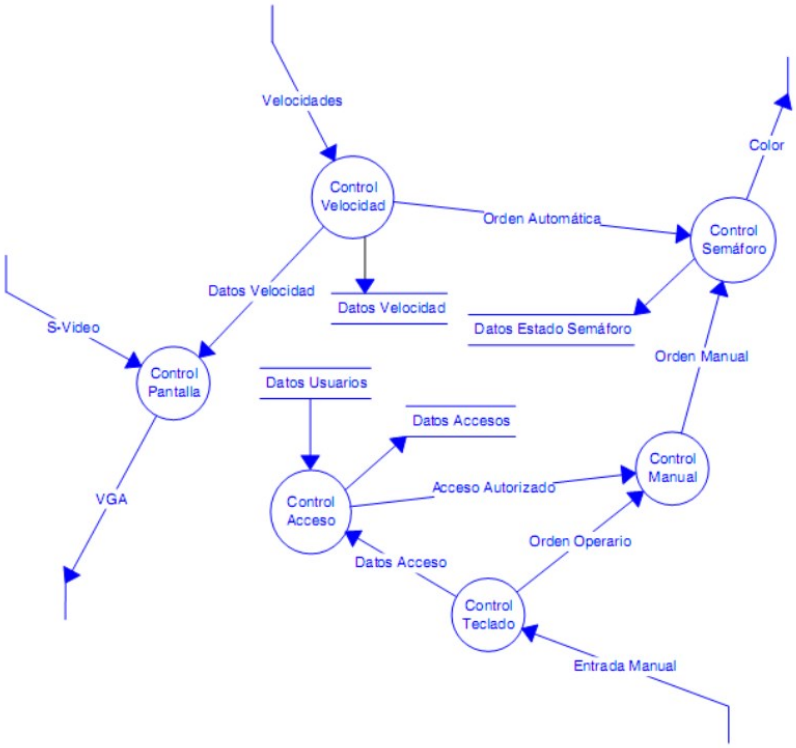
ENUNCIADO

Se desea realizar una aplicación informática para evitar congestiones de automóviles en el interior de un túnel. El sistema recibe datos de la velocidad de los automóviles provenientes de un conjunto de sensores distribuidos en el interior del túnel. Los datos de todos los sensores se reciben en paralelo y de forma periódica. El sistema deberá actuar, en función de dichos datos, sobre un semáforo situado a la entrada del túnel, de forma que cuando se detecte un coche circulando a una velocidad inferior a una estipulada, el semáforo se pondrá en rojo y cuando las velocidades detectadas por los sensores sean superiores a otro valor estipulado, se volverá a poner en verde. Existirá un centro de control con una pantalla que muestre el estado del semáforo y las velocidades instantáneas captadas por el conjunto de sensores. El sistema podrá funcionar en modo manual, para lo cual el operario deberá introducir por teclado su identificador y contraseña cada vez que quiera pasar de automático a manual, o viceversa, y/o cambiar el estado del semáforo. Una o varias cámaras de video enviarán imágenes del interior y exterior del túnel, que se mostrarán en la pantalla y servirán para ayudar al operario a tomar decisiones cuando realice el control manual. El sistema almacenará todos los datos sobre velocidades proporcionadas por los sensores, el estado del semáforo, los cambios de estado que se produzcan en él y su origen (automático o manual) y los accesos de los usuarios, junto con la fecha y hora de cada evento. "

Modele el sistema empleando un diagrama de flujo de datos (niveles 0 y 1).

SOLUCIÓN





DFD1

1º Saber el modelo y buscar los elementos y las partes importantes del enunciado acordes al diagrama.
2º En un DFD buscar, almacenes, agentes externos, flujos de datos s sistemas/subsistemas acordes a cada nivel.

Ejercicio2. DFD. Rutas Verdes

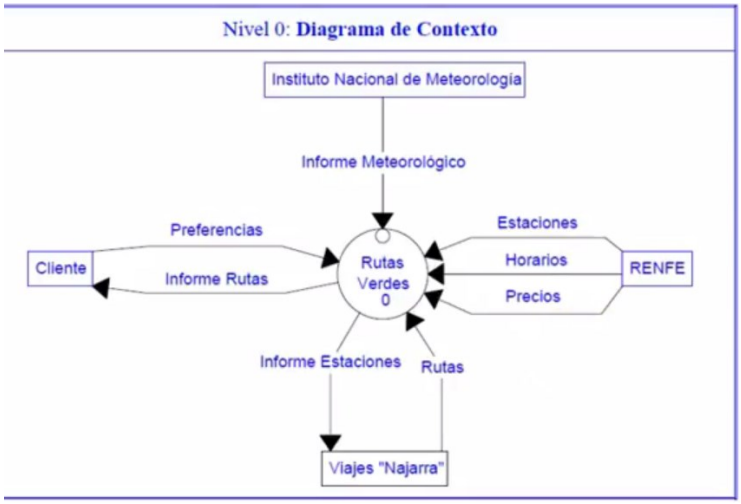
ENUNCIADO

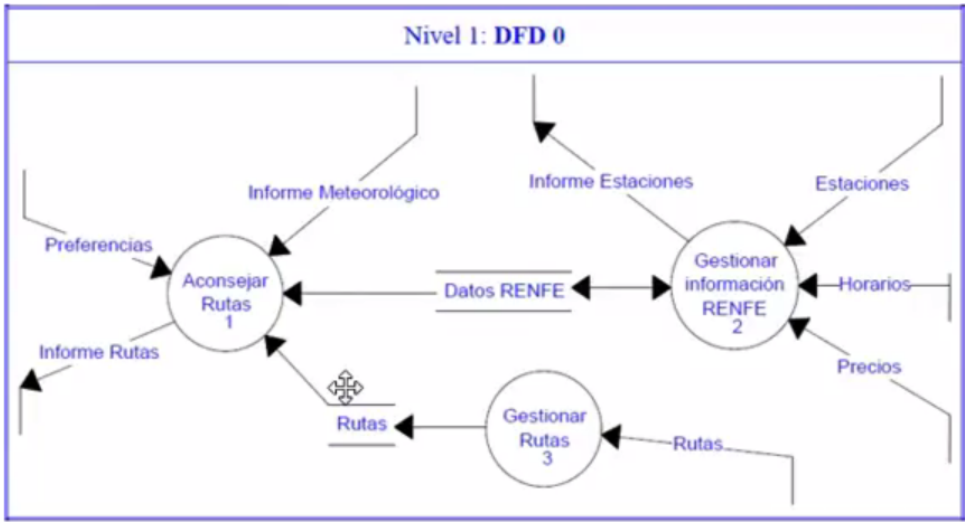
- Con el fin de promocionar el uso del transporte público y el ocio al aire libre, RENFE ha decidido encargar la construcción de un sistema informático que asesore a sus clientes acerca de 'rutas verdes' para hacer a pie a partir de sus estaciones de tren. El sistema recibirá periódicamente la siguiente información:
  - Un informe meteorológico del Instituto Nacional de Meteorología que contendrá las previsiones climáticas para los próximos días.
  - Datos referentes a las estaciones de tren, horarios y precios de billetes. Esta información será suministrada por RENFE.
  - Se ha encargado a la empresa "Viajes Najarra" la elaboración e introducción en el sistema de las rutas verdes. Para ello, la empresa podrá solicitar del sistema un informe de las estaciones de RENFE existentes.

  Los clientes introducirán en el sistema sus preferencias. A partir de estas y los datos antes descritos, se construirá un informe con las rutas aconsejadas.

  Analice el sistema mediante DFDs (Diagramas de Flujo de Datos), desarrollando exclusivamente los DFDs de nivel 0 (contexto) y nivel 1.

SOLUCIÓN





Ejercicio3. DFD. Mail Secure

ENUNCIADO

El sistema Mail Secure sirve para enviar correos electrónicos firmados y/o cifrados. El funcionamiento básico (simplificado) es el siguiente:

> NOTA: Cada usuario tiene 2 claves, una privada accesible por él y otra pública compartida por el resto de los usuarios.

Veamos con un ejemplo como Alicia envía un mensaje firmado y luego cifrado (oculto) a Bernardo, su amigo.

1) Alicia pasa su mensaje por una función hash que lo comprime a modo de resumen.
2) Este mensaje hash o resumen se codifica con la clave privada de Alicia. El mensaje resultante se conoce como firma digital.
3) El mensaje original queda firmado al usuario con su firma.
4) Alicia puede enviar el mensaje firmado (original + firma) a Bernardo.
5) Pero si Alicia quisiera además firmar (codificar) su mensaje antes de enviarlo, utilizaría la clave pública de Bernardo para cifrar el mensaje antes de firmarlo obteniendo así un mensaje firmado y cifrado (encriptado).
6) Ahora Alicia envía a Bernardo su mensaje firmado y cifrado (encriptado) a salvo de cualquier mirada y además llevando la marca de autenticidad (su firma).

Veamos ahora como procede Bernardo para descifrar el mensaje y comprobar su autoría.

1) Cuando Bernardo recibe el mensaje cifrado y firmado de Alicia pasa a descifrarlo usando su clave privada, obteniendo el mensaje original de Alicia y su firma.
2) Para comprobar que es Alicia quien lo ha firmado, descifra la firma con la clave pública de Alicia. Por otra parte, aplica la función hash al mensaje descifrado de Alicia. Obteniendo así dos mensajes resúmenes,
3) Si coinciden, se puede garantizar que la firma es de Alicia, si no, se rechaza el mensaje.

Modele el sistema anterior mediante dos DFD's: uno para leer la parte emisora (Alicia) y otro para la parte receptora (Bernardo) para el caso que se quiera enviar un mensaje cifrado y firmado.

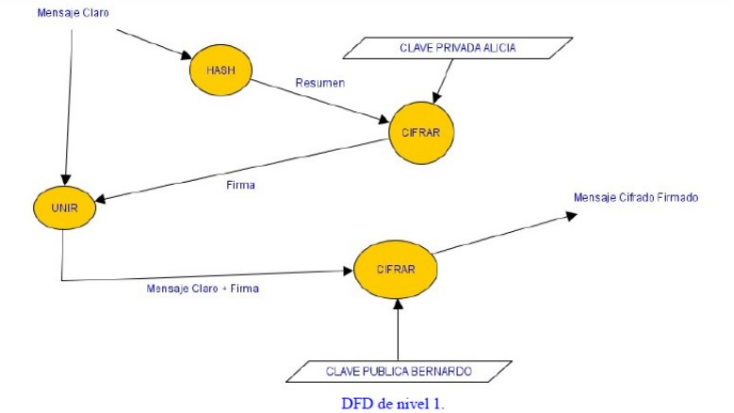
NOTA: Codificar, cifrar o encriptar es aplicar una transformación a un texto utilizando un algoritmo que necesita un parámetro que se llama clave. La finalidad es ocultar el texto original obteniendo otro texto ilegible. Descifrar o desencriptar es la operación inversa.

SOLUCIÓN

DFD Contexto:

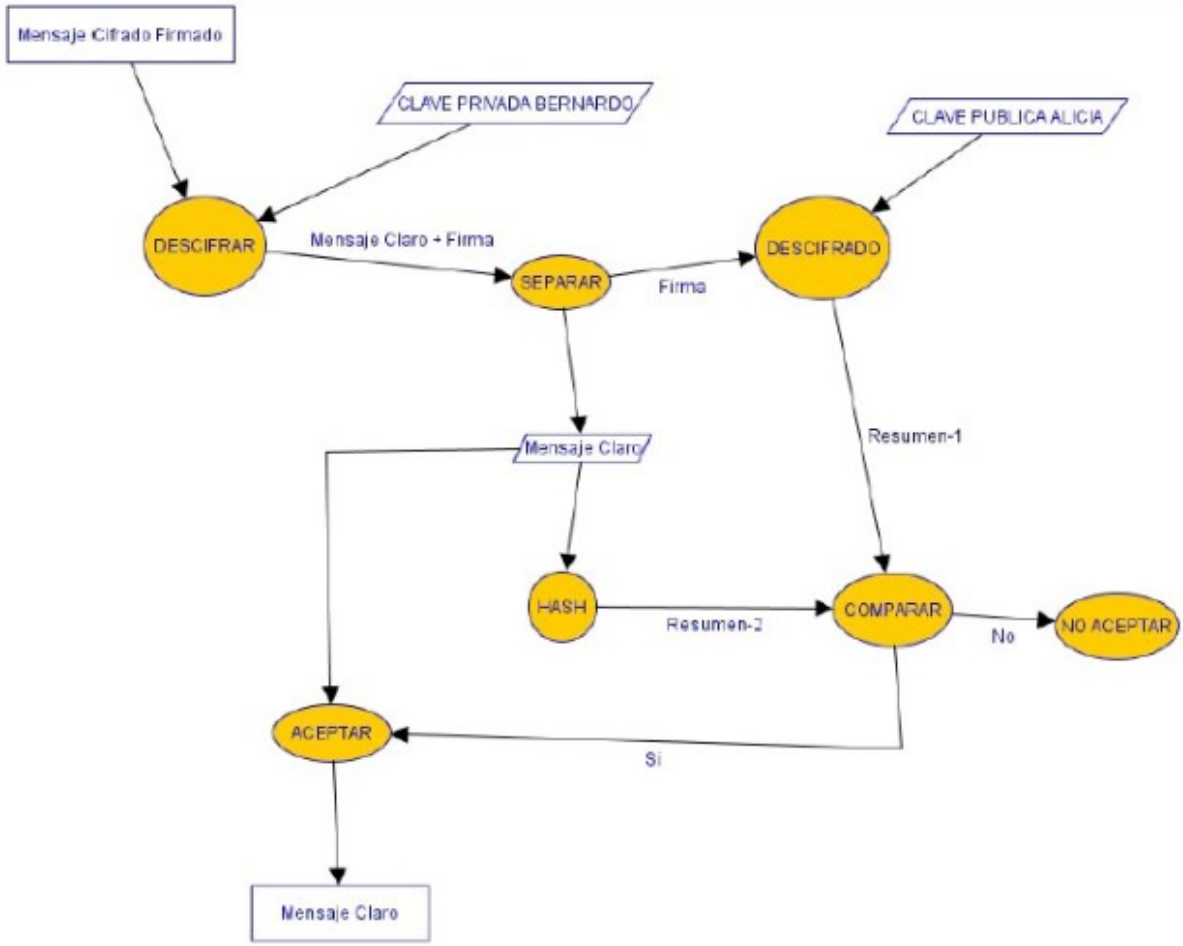
- **Emisor (Alicia)**

DFD de contexto. Nivel 0



DFD de nivel 1.

- **Receptor (Bernardo)**

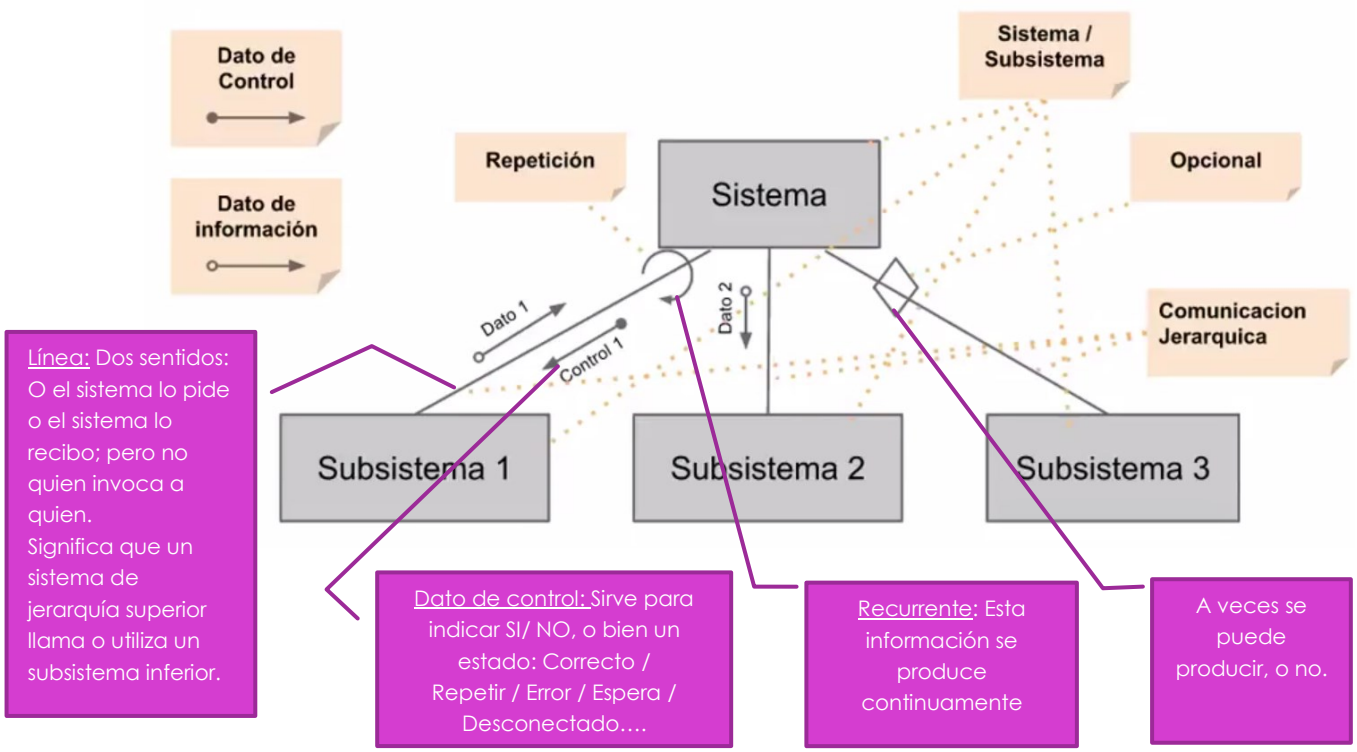


4. Diagramas de Estructura (DEst)

DER→Diagrama de Entidad/Relación | ERD→Entity Relation

Resumen Teoría

Tipo de modelado **estático** que también tiene flujo de información.
Comunicación jerarquía de sistema/subsistemas.
Datos de Información → son datos de información pura, el Dato…usuario, factura, …
Datos de control → sirven para hacer actuación en el sistema, como evento. (avanza, lectura, guarda de datos, ACK[1]).
DEst no establece ninguna secuencia concreta de utilización de los módulos y tan sólo refleja una organización estática de los mismos.



[1] ACK → Acknowlegment (Recibido)

Ejercicio1. DFD/DEst. Compilador

ENUNCIADO

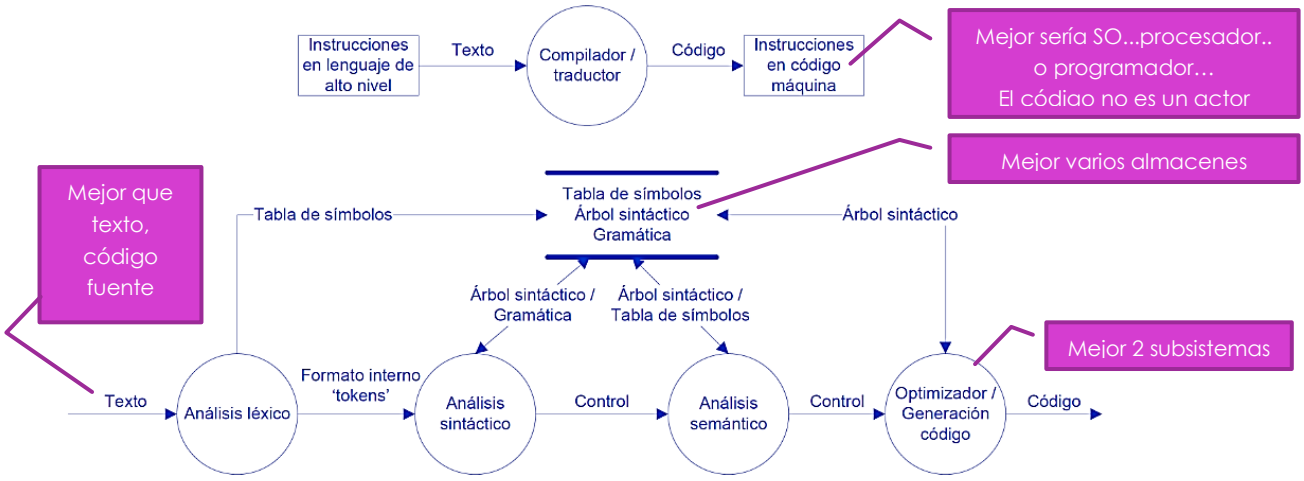
Los sistemas de procesamiento de lenguaje aceptan como entrada sentencias en algún lenguaje y generan como salida alguna otra representación del lenguaje de entrada. Un caso particular son los compiladores, que traducen un lenguaje de programación artificial de alto nivel a código máquina (abstracta o de un procesador real). Los principales componentes de un traductor (sea para un compilador u otro procesador de lenguaje) son:

1. Un analizador léxico, que toma como entrada los elementos del lenguaje, identifica sus símbolos y los convierte a un formato interno.
2. Una tabla de símbolos, que almacena información sobre los nombres de las entidades (variables, nombres de clases, de objetos, etc.) usadas en el texto que se está traduciendo.
3. Un analizador sintáctico, que comprueba la sintaxis del lenguaje que se está traduciendo. Utiliza una gramática definida y construye un árbol sintáctico.
4. Un árbol sintáctico, que es una estructura interna que representa el texto que se está traduciendo o el programa que se está compilando.
5. Un analizador semántico, que utiliza la información del árbol sintáctico y de la tabla de símbolos para comprobar la corrección semántica del texto en el lenguaje de entrada.
6. Un optimizador, que transforma el árbol sintáctico para mejorar la eficiencia y eliminar redundancias en el código máquina que se genere.
7. Un generador de código, que «recorre» el árbol sintáctico y genera código máquina (traducido).

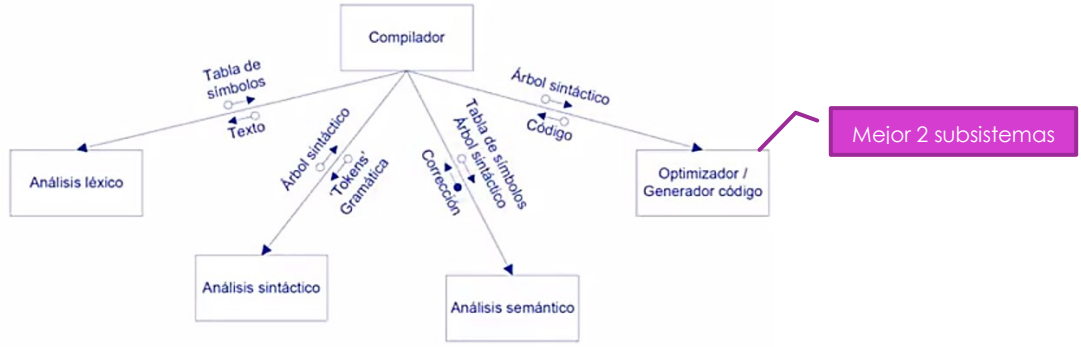
Se pide:

1. **Represente la arquitectura genérica de un compilador mediante un modelo de flujo de datos (haga el DFD, el correspondiente análisis de flujo de datos**
2. **Represente la arquitectura con un diagrama de estructura).**

SOLUCIÓN punto 1



SOLUCIÓN punto 2

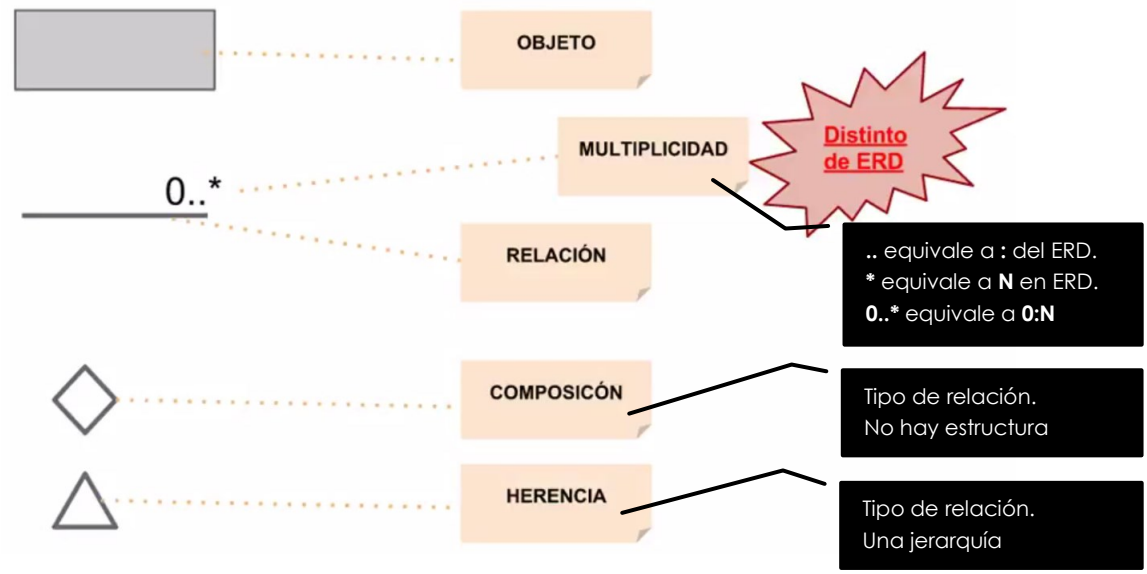


5. Diagramas de Objetos (DO)

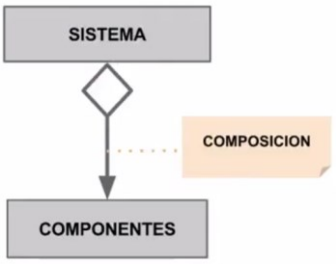
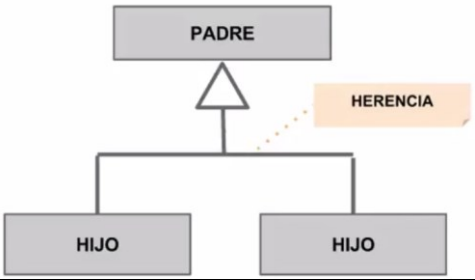
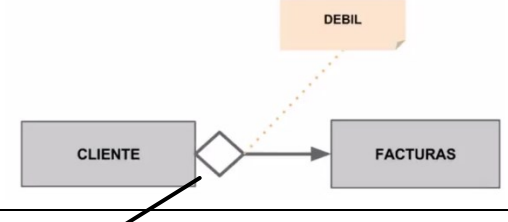
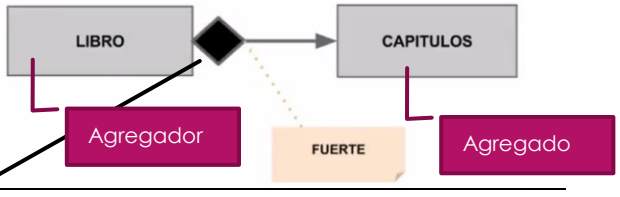
DER→Diagrama de Entidad/Relación | ERD→Entity Relation

Resumen Teoría

Tipo de modelado **estático** que también tiene flujo de información.



Cuando pasas de un diagrama de Objetos a Diagramas de Relación, todos los objetos de DO tienen su concepto en la relación para adaptarlo a un DER.

Representación Composición	Representación Herencia
	
Composición o Agregación. Siempre que un objeto esté compuesto de uno o varios tipos de componentes.	Concepto general que se puede clasificar en aspectos concretos.
¿Cuando unar una u otra represnetación? En el enunciadocuando veamos ciertas pistas usaremos uno u otro:	
… está **compuesto** de… … está **formado** por… … **tiene**… …está **dividido** en…	… **tipos** de… … es un tipo de… … se **clasifican** en… … es una **categoría** de…
	
Los dos conceptos que unen, pueden existir independientemente	No tiene sentido que sentido que exista agregado si no existe el agregador

Ejercicio1. DO. Metales

ENUNCIADO

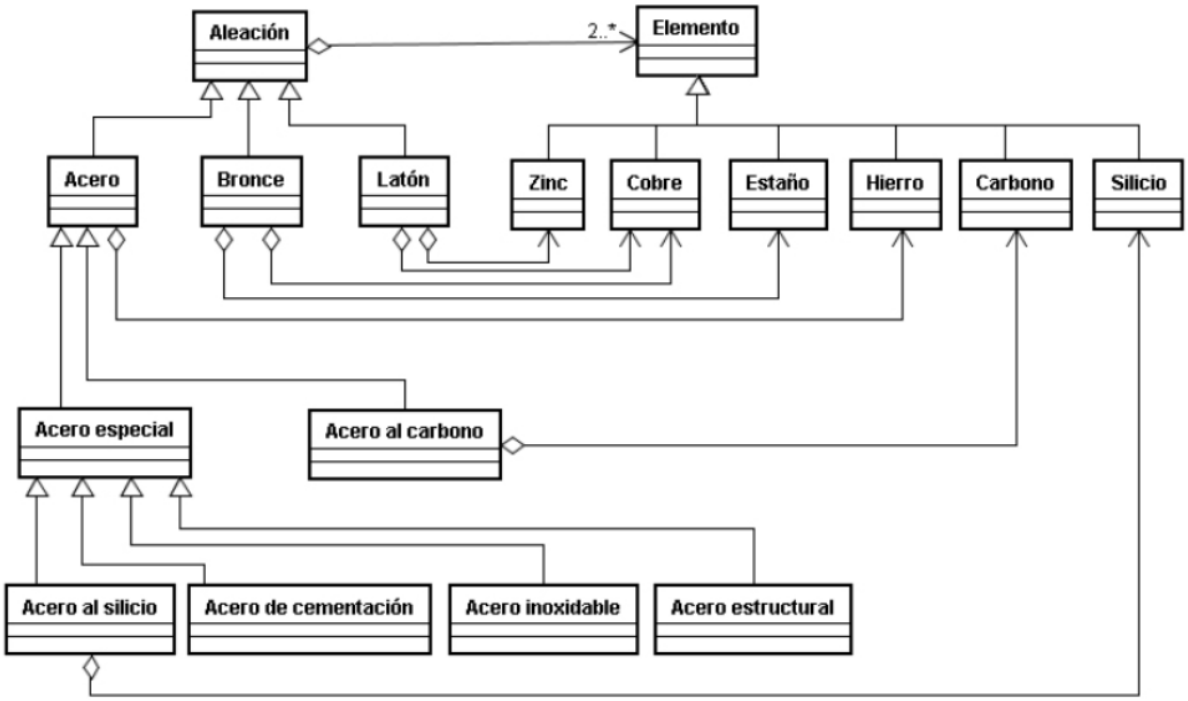
Modele con un diagrama de objetos lo que se indica en el siguiente enunciado:

Por aleación se entiende la unión homogénea de dos o más elementos. Generalmente, se dice que el acero es una aleación de hierro y carbono. Sin embargo, esta definición se restringe a los "aceros al carbono", ya que existen otros tipos de acero con composiciones muy diversas que reciben denominaciones específicas en virtud:

- de los elementos que, además del hierro, predominan en su composición (por ejemplo, aceros al silicio).
- de su susceptibilidad a ciertos tratamientos (por ejemplo, aceros de cementación)
- de alguna característica potenciada (por ejemplo, aceros inoxidables)
- en función de su uso (por ejemplo, aceros estructurales).

Usualmente, estas otras aleaciones de hierro se engloban bajo la denominación genérica de "aceros especiales". Además del acero, otras aleaciones muy comunes son el bronce, formado por cobre y estaño, y el latón, compuesto por cobre y zinc.

SOLUCIÓN



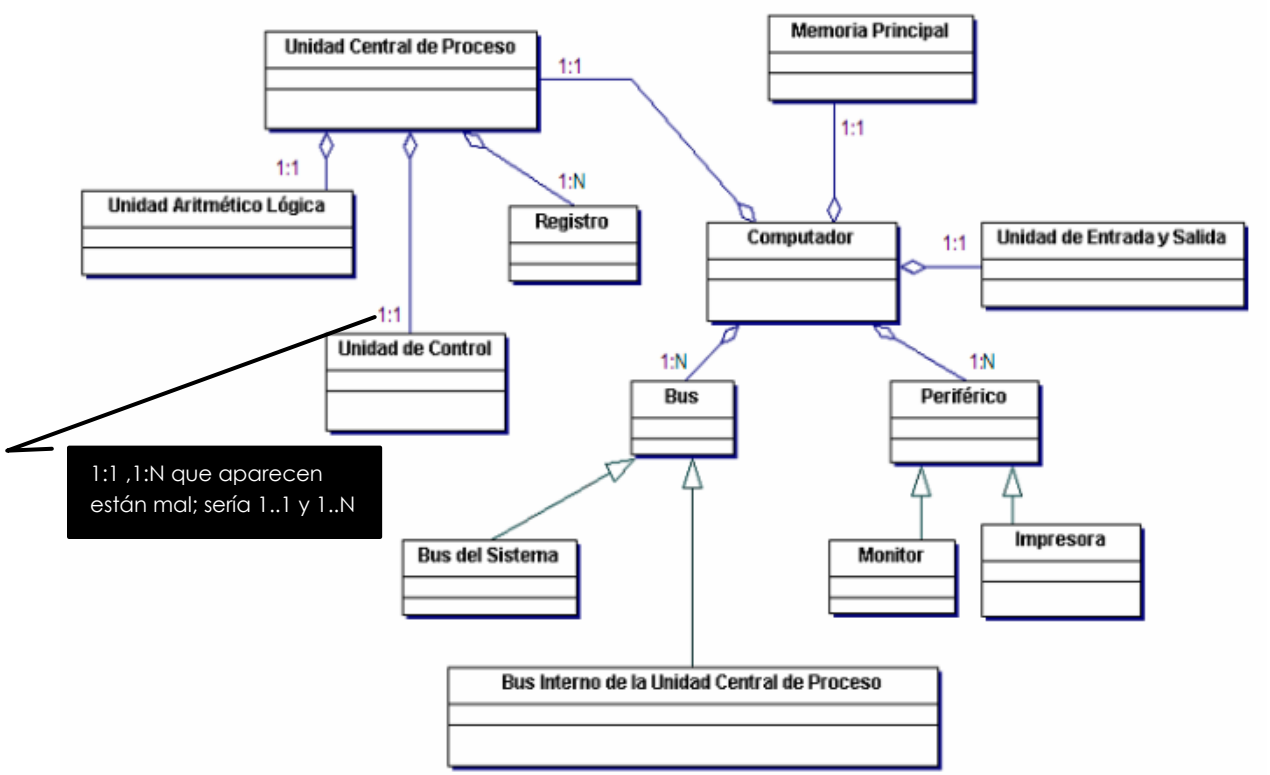
Ejercicio2. DO. Computador

ENUNCIADO

Según la arquitectura Von Neumann, un computador está compuesto por:

1. Unidad Central de Proceso: controla el funcionamiento del computador y lleva a cabo sus funciones de procesamiento de datos. A su vez, consta de:
   a. Unidad Aritmético Lógica: se encarga de realizar operaciones elementales como la suma, resta, AND…
   b. Unidad de Control: se encarga de leer y ejecutar las instrucciones máquina almacenadas en la memoria principal.
   c. Registros: proporcionan almacenamiento interno a la Unidad Central de Proceso.
2. Memoria Principal.
3. Unidad de Entrada y Salida: transfiere datos entre el computador y los periféricos.
4. Periféricos: los hay de distinto tipo, por ejemplo,
   a. Monitor
   b. Impresora
5. Elementos de interconexión o buses: los hay de distinto tipo, por ejemplo,
   a. Bus del Sistema: conecta Unidad Central de Proceso, Memoria Principal y Unidad de E/S
   b. Bus Interno de la Unidad Central de Proceso: conecta Unidad Aritmético Lógica, Unidad de Control y Registros.

SOLUCIÓN



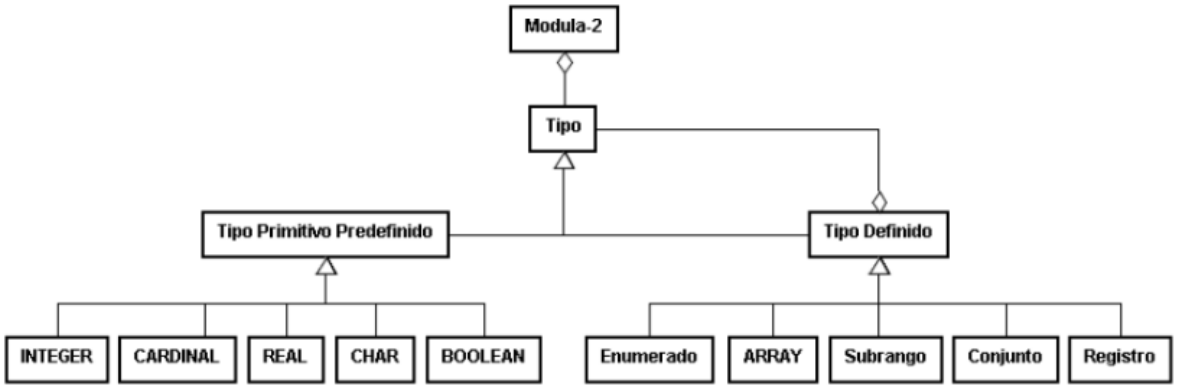
Ejercicio3. DO. Modula2

ENUNCIADO

Modele la siguiente especificación con un Diagrama de Objetos:

"El lenguaje de programación Modula-2 dispone de los siguientes tipos primitivos predefinidos: INTEGER, CARDINAL, REAL, CHAR y BOOLEAN. Además, soporta la definición por parte del programador de nuevos tipos Enumerado, Subrango, ARRAY, Conjunto y Registro. Los nuevos tipos se crean a partir de los tipos primitivos y/o cualquier otro tipo definido previamente por un programador."

SOLUCIÓN



6. Pruebas

Resumen Teoría

Objetivo de las pruebas: Hacer que falle, no que funciona.

Pruebas Caja Negra → lo que conocemos es como se debe de comportar según sus entradas/salidas sin saber nada del código.

Se crean las pruebas acordes a los requisitos del programa.

Partición en clases de equivalencia: dividir el espacio de ejecución del programa en varios subespacios. Cada subespacio o clase equivalente agrupa a todos aquellos datos de entrada al módulo que resultan equivalentes desde el punto de vista de la prueba de la caja negra.

Análisis de valores límite: hace hincapié en las zonas del espacio de ejecución que están próximas al borde. Los errores en los límites pueden ser graves al solicitar recursos que no se habían preparado. Se deben proponer casos válidos y casos inválidos.

Comparación de versiones: se hacen diferentes versiones del mismo modulo hechas por diferentes programadores y se someten al mismo juego de pruebas. Se comparan y evalúan los resultados.

Empleo de la intuición: Una persona ajena al desarrollo del proyecto crea una batería de pruebas basándose en su experiencia.

Pruebas Caja Transparente → necesitamos saber el código. Las pruebas se basan en él y se probará todo el código.

Cubrimiento lógico: tipo de modelo de diagramas en el que haces un diagrama de flujo poniendo los diferentes de caminos que puede recorrer el programa.

Nº máximo de caminos = Nº de predicados + 1.

- 3 Niveles:
  - Nivel1: creación de pruebas para que se ejecuten **todos** los caminos una vez mínimo.
  - Nivel2: creación de pruebas para que se ejecuten toda combinación de camino por parejas.
  - Nivel3: creación de pruebas para que se ejecuten un número significativo de combinaciones (todas sería inviable).

Bucles: se crearán pruebas para verificar los bucles.

- Número no acotado. Repetir el bucle 0, 1, 2, algunas veces, muchas veces.
- Ejecuciones máximas(M). Ejecutar el bucle M+1 de lo necesario.
- Anidados. Ejecutar el bucle externo el número mínimo (ir hacia dentro ejecutando en cada bucle el mínimo).
- Concatenados. Si son independientes se probarán en el número no acotado y en ejecuciones máximas. Si la salida de uno es la entrada de otro, se probará como anidado.

Lo importante de un juego de pruebas es que tenga las entradas, las salidas y algún comentario (opcional).

Ejercicio1. Pruebas Caja Negra. Hotel

ENUNCIADO

Se desea comprobar la corrección de un programa que calcula el precio de la estancia en un hotel. Los datos de entrada al programa son el día y el mes de la primera noche de estancia y el número de noches. El precio por noche es de:

- 100 € del 7 de enero al 30 de junio y del 1 de septiembre al 23 de diciembre.
- 140 € del 1 de julio al 31 de agosto.

El hotel permanece cerrado del 24 de diciembre al 6 de enero. La salida que genera el programa es el precio en euros, siempre que se le hayan proporcionado valores correctos a la entrada. Si los datos de entrada no son adecuados, el programa devuelve el texto "datos no válidos".

Se pide desarrollar un juego de pruebas de error del programa, justificando la elección de los casos escogidos.

SOLUCIÓN

Como sólo se conocen la especificación de entradas/salidas del programa, el juego de pruebas será caja negra.
Para ello emplearemos los métodos de "partición en clases de equivalencia" y "análisis de valores límite".
Podemos dividir el espacio de ejecución en 3 subespacios:

- Temporada baja (2 periodos → 07/01 al 30/06 y 01/09 al 23/12)
- Temporada alta (01/07 al 31/08)
- Temporada de cierre (21/12 al 07/01)

Y podemos estableces como valores límite las fechas de transición entre estas clases de equivalencia.

06/01	30/06	31/08	23/12
07/01	01/07	01/09	24/12

Se crea una matriz de caso válidos/no válidos con ciertos valores, entre ellos los límites.

CASOS VÁLIDOS		
ENTRADA	COMENTARIO	SALIDA
07/05, 5 noches	Valor límite temporada baja	500 euros (5 x 100)
23/03, 11 noches	1er periodo temporada baja	1100 euros (11 x 100)
28/06, 3 noches	Valor límite temporada baja	300 euros (3 x 100)
25/06, 7 noches	Transición baja/alta. Valor límite alta	740 euros (6 x 100 – 1 x 140)
01/07, 2 noches	Valor límite temporada alta	280 euros (2 x 140)
12/07, 20 noches	Temporada alta	2800 (20 x 140)
20/08, 12 noches	Valor límite temporada alta	1680 euros (12 x 140)
31/08, 2 noches	Transición baja/alta. Valores límite	240 euros (100 + 140)
15/11, 6 noches	2º periodo temporada baja	600 euros (6 x 100)
21/12, 3 noches	Valor límite temporada baja	300 euros (3 x 100)
28/06, 70 noches	Transición baja/alta.	9480 euros (8 x 100 + 62 x 140)
CASOS NO VÁLIDOS		
ENTRADA	COMENTARIO	SALIDA
03/01, 7 noches	Temporada cierre	"datos no válidos"
06/01, 2 noches	Valor límite temporada de cierre	"datos no válidos"
19/12, 11 noches	Temporada cierre	"datos no válidos"
24/12, 4 noches	Valor límite temporada de cierre	"datos no válidos"

Ejercicio2. Pruebas Caja Negra. Hotel

ENUNCIADO

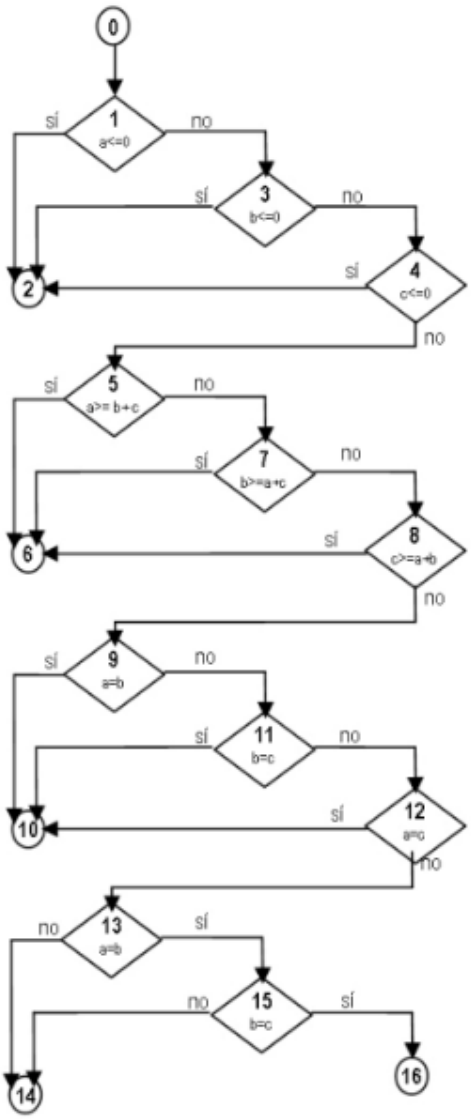
Se ha codificado con Modula-2 el siguiente subprograma que distingue, por el tamaño de sus lados si un triángulo es isósceles (2 lados iguales), equilátero (todos los lados iguales) o escaleno (ningún lado igual):

```
PROCEDURE TipoDeTriangulo(a, b, c: INTEGER);
BEGIN
  IF (a<=0) OR (b<=0) OR (c<=0) THEN
    WriteString("Error: alguno de los parámetros no es un número natural");
  ELSIF (a>=b+c) OR (b>=a+c) OR (c>=a+b) THEN
    WriteString("Error: no es posible cerrar el triángulo");
  ELSIF (a=b) OR (b=c) OR (a=c) THEN
    WriteString("El triángulo es isósceles");
  ELSIF (a=b) AND (b=c) THEN
    WriteString("El triángulo es equilátero");
  ELSE
    WriteString("El triángulo es escaleno");
  END;
END TipoDeTriangulo;
```

Verifique el subprograma con pruebas de caja transparente, realizando el cubrimiento lógico correspondiente.

SOLUCIÓN

Nº predicados = 11 | Nº máximos de caminos = Nº predicados+1 = 11+1 = 12



	Camino	Resultado esperado	Juego de prueba
1	0-1-2	Parámetros erróneos: valores no naturales	a=0, b=1, c=1
2	0-1-3-2		a=1, b=-3, c=1
3	0-1-3-4-2		a=3, b=1, c=0
4	0-1-3-4-5-6	Parámetros erróneos: imposible cerrar el triángulo	a=5, b=1, c=1
5	0-1-3-4-5-7-6		a=2, b=4, c=2
6	0-1-3-4-5-7-8-6		a=3, b=1, c=5
7	0-1-3-4-5-7-8-9-10	Triángulo isósceles	a=2, b=2, c=1
8	0-1-3-4-5-7-8-9-11-10		a=2, b=3, c=3
9	0-1-3-4-5-7-8-9-12-10		a=4, b=3, c=4
10	0-1-3-4-5-7-8-9-12-13-14	Triángulo escaleno	a=3, b=4, c=5
11	0-1-3-4-5-7-8-9-12-13-15-14		Es imposible crear un juego de prueba
12	0-1-3-4-5-7-8-9-12-13-15-16	Triángulo equilátero	

(2) Parámetros erróneos: valores no naturales

(6) Parámetros erróneos: imposible cerrar el triángulo

(10) Triángulo isósceles

(14) Triángulo escaleno

(16) Triángulo equilátero

7. UML

Resumen Teoría

UML: Lenguaje universal de modelado.
Lenguaje gráfico. Da información de la estructura estática y el comportamiento dinámico de un sistema.

7.1 ESTRUCTURA UML

Estructura estática, definir:

- Conceptos clave de la aplicación
- Propiedades internas
- Relaciones entre cada una de ellas.

Los conceptos de la aplicación son modelados como clases, la información almacenada se modela atributos.

Comportamiento dinámico, unifica:

- La estructura de datos
- Control de flujo
- Flujo de datos

Interacción entre objetos con flujo de mensajes y enlaces entre ellos.

Construcciones de implementación:
Los modelos UML tienen significado para el análisis lógico y para la implementación física.

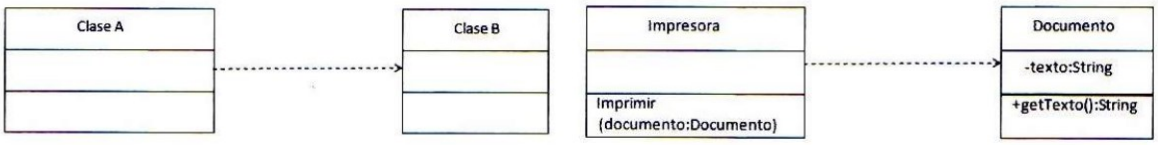
Mecanismos de extensión:
UML tiene una limitada capacidad de extensión.

Organización del modelo:
La información del modelo debe ser dividida lo más pequeño posible.

Anotaciones:
Partes explicativas del UML. Comentarios para ayudar a describir cualquier elemento del modelo.

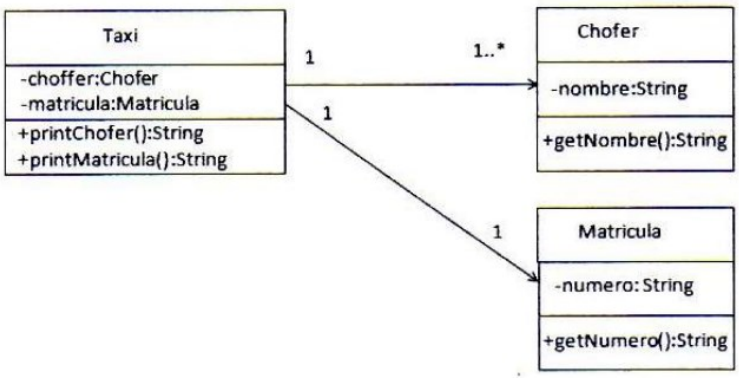
Relaciones:

- **Dependencia**. El cambio en una clase afecta a otra. La clase A usa la B.

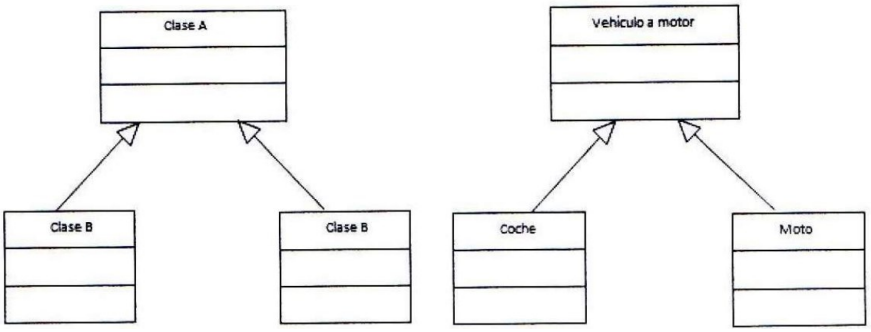


- **Asociación**. Relación que describe los objetos que intervienen usando multiplicidad.

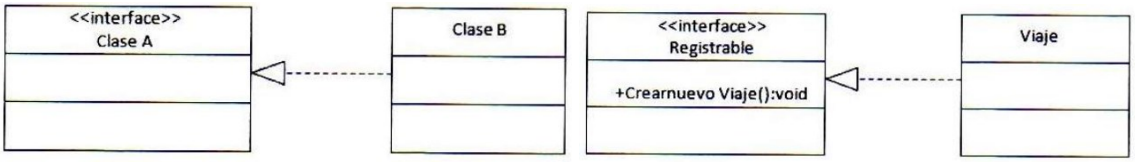
Multiplicidad	Significado
1	Uno y sólo uno
0..1	Cero o uno
N..M	Desde N hasta M
*	Cero o varios
0..*	Cero o varios
1..*	Mínimo uno, o varios



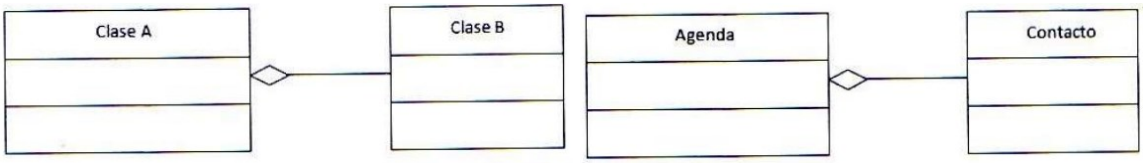
- **Generalización**. La herencia de POO. Lo del padre lo hereda el hijo.



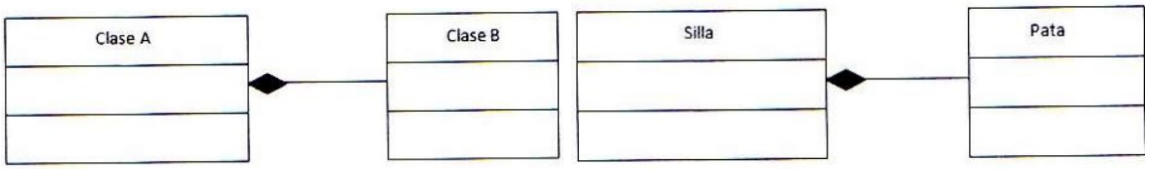
- **Realización**. En POO, los interfaces y las clases y componentes que las realizan.



- **Agregación**. Tiene multiplicidad preestablecida de "uno a muchos". Clase A puede *vivir* sin clase B.



- **Composición**. Tiene multiplicidad preestablecida de "uno a muchos". Clase A **NO** puede *vivir* sin clase B.



7.2 Diagramas. Caso de uso:

Describen las relaciones y dependencias entre un grupo de casos de uso y los actores del proceso.

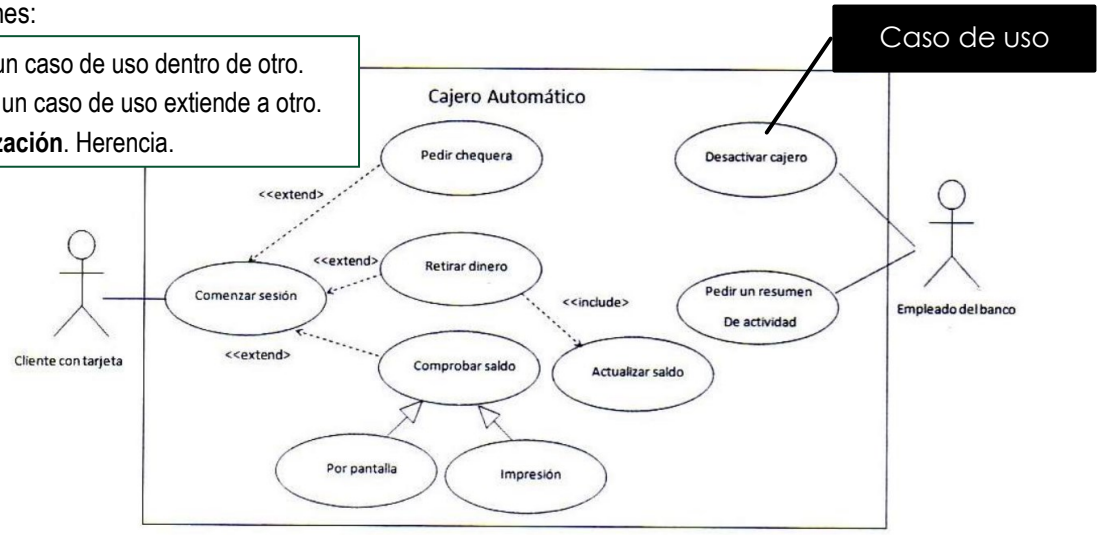
Describe que es lo que debe hacer el sistema, pero no describe el cómo.

Actor. Es el descriptor entre el usuario y el sistema. Representa el interfaz que verá el usuario.

Cada caso de uso está relacionado con un actor como mínimo.

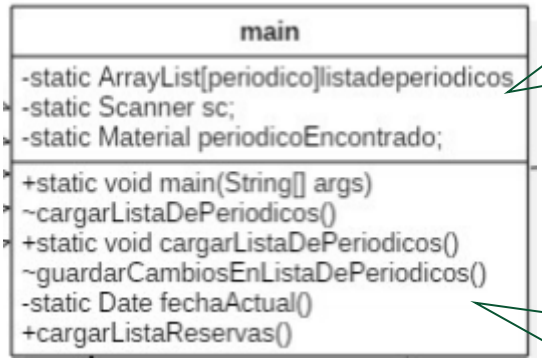
Tipos de relaciones:

- **Include**: un caso de uso dentro de otro.
- **Extends**: un caso de uso extiende a otro.
- **Generalización**. Herencia.



7.3 Diagramas de clases:

Muestra las **clases** del sistema y como se relacionan. Son diagramas estáticos.



Atributos (mostrar mínimo el nombre).
Tipos de acceso:
+ Atributo público.
Atributo protegido.
- Atributo privado.

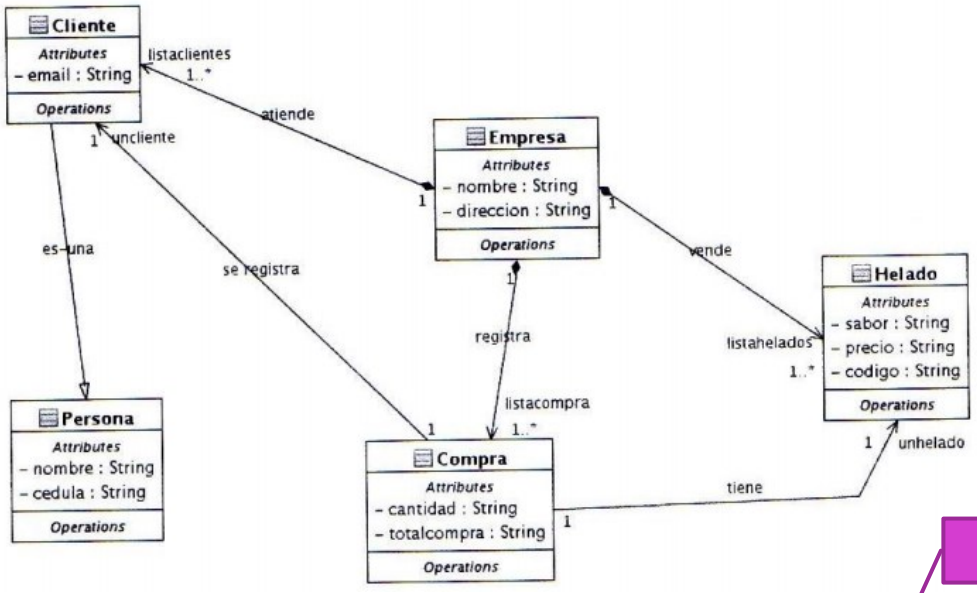
Operaciones (mostrar mínimo el nombre).
Tipos de acceso:
+ Atributo público.
Atributo protegido.

Posibilidad de relación entre clases:

- Generalización, asociación, realización, agregación y composición.

Los diagramas de clases pueden tener también los siguientes componentes:

- **Interface**. Clase abstracta. Puede tener operaciones, no tiene atributos. Solo sirve para que hereden de ella.
- **Tipo de datos**. Datos primitivos. No puede tener relación con clases, pero las clases con ellas sí.
- **Enumeraciones**. Son lista de valores. No puede tener relación con clases, pero las clases con ellas sí.
- **Paquetes**. Paquete que contiene más de una clase.

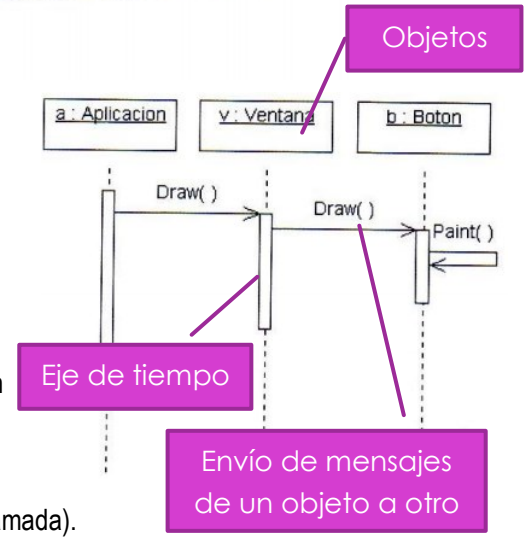


7.4 Diagramas de secuencia:

Muestra el **flujo de mensajes** entre objetos.

El orden y momento es importante. Puede tener mayor o menor detalle. Tiene 2 dimensiones:

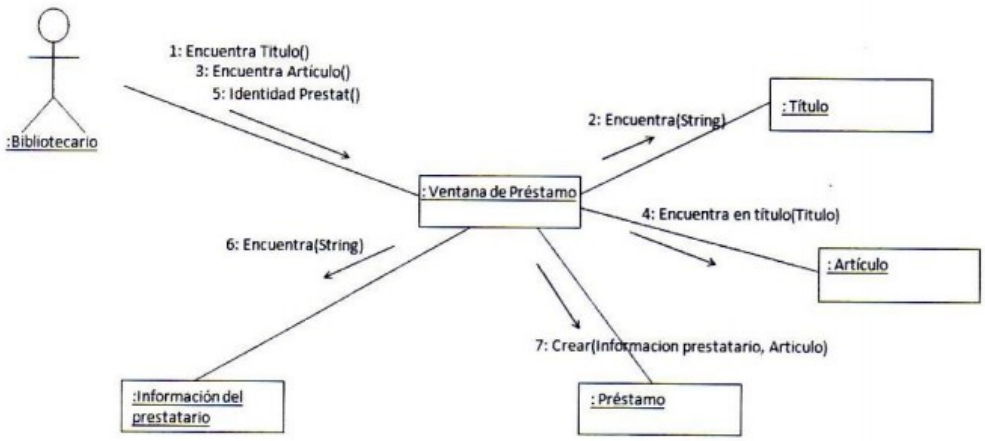
- Vertical: muestra secuencia de llamadas ordenadas en el tiempo.
- Horizontal: Muestra las instancias de objetos a las que son enviadas las llamadas.



Los mensajes pueden ser **síncronos** (pasa el mensaje cuando un método acaba) o **asíncronos** (el control va al objeto que hace la llamada).

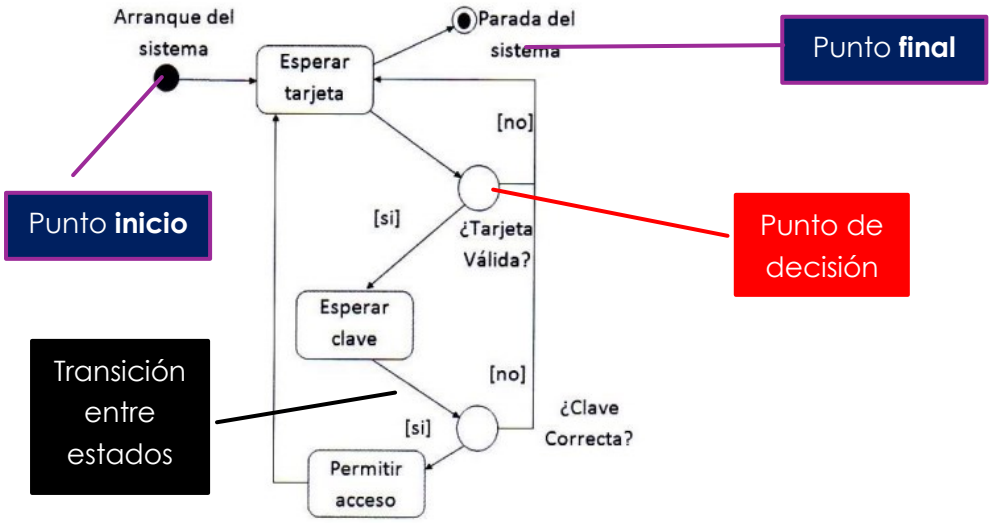
7.5 Diagramas de colaboración:

Muestra las iteraciones que ocurren entre objetos en una situación en concreto. Los mensajes entre objetos se representan con flechas.



7.6 Diagramas de estado:

Muestra los estados de un objeto durante su vida y que hace que cambie de estado. → rollo autómatas finitos. Se representan objetos con 3 o más estados; con menos pasar del tema.



7.6 Diagramas de actividad:

Sólo contiene actividades del sistema. Rollo diagramas de flujo, pero con actividades unidas a objetos.