**UNIDAD 1. DESARROLLO DE SOFTWARE**

El ordenador se compone de dos partes: el hardware y el software. El hardware los forman los componentes físicos que se pueden ver y tocar.El software lo forma la parte lógica del ordenador.

**SOFTWARE DEL ORDENADOR**

El software es todo aquello que se refiere a los programas y datos almacenados en un ordenador.

El software se divide en dos categorías: Según el **tipo de tarea que realiza** y según **el método de distribución.**

1. **SOFTWARE BASADO EN EL TIPO DE TAREA QUE REALIZA.**
   1. Software de sistema: Es aquel que permite que el hardware funcione. Lo forman los programas que permiten la administración de la parte física. Por ejemplo, los sistemas operativos, los controladores de dominio, etc.
   2. Software de aplicación: Lo forman los programas que nos ayudan a realizar tareas específicas
   3. Software de programación o desarrollo: Es el que proporciona al programador herramientas para ayudarle a escribir programas.
2. **SOFTWARE BASADO EN EL MÉTODO DE DISTRIBUCIÓN.**
   1. Shareware. Es una modalidad de distribución de software que el usuario puede evaluar de forma gratuita. Para adquirir una licencia que permita usar el software de forma completa se requiere un pago.
   2. Freeware. Es un software gratis. A veces se incluye el código fuente pero no es lo usual. Suele incluir su distribución, pero con muchas restricciones. Los programas de software libres no necesariamente son gratis.
   3. Adware.Suelen ser programas Shareware que de forma automática descargan publicidad.
   4. Software multimedia. Programas utilizados para presentar textos, gráficos,
   5. Software de uso específico. Es el que se desarrolla para resolver un problema determinado.
3. **LICENCIAS DE SOFWARE.SOFTWARE LIBRE Y PROPIETARIO.**

Una licencia de software es un contrato que se establece entre el desarrollador y el usuario en el que se definen los derechos y obligaciones de ambos.

1. Software libre: Es aquel en el que el autor cede una serie de libertades
   1. Libertad de utilizar el programa con cualquier fin
   2. Libertad de estudiar cómo funciona el programa y adaptar su código.
   3. Libertad de distribuir copias.
   4. Libertad para mejorar el programa.
2. Software propietario. Es el que se distribuye en formato binario, sin posibilidad de acceso al código fuente,
3. El software de dominio público es el que carece de licencia y no hay forma de determinarla porque se desconoce el autor.

La licencia más utilizada en los productos desarrollados de software libre es GPL (General Public License)

**CICLO DE VIDA DEL SOFTWARE.**

El proceso de desarrollo de software implica un conjunto de actividades que se tienen que planificar y gestionar de manera que aseguren un producto final que dé solución al problema.

Se consideran las siguientes etapas: especificación y análisis de requisitos, diseño del sistema, implementación del software, aplicación y pruebas, entrega y mantenimiento.

1. Análisis: Construye un modelo de requisitos. Se debe entender de forma detallada el problema a resolver.
2. Diseño. Ya sabemos que hay que hacer, ahora hay que decidir cómo se va a resolver el problema. Se deciden la estructura de datos, la arquitectura de software, la interfaz de usuario y los procedimientos.
3. Codificación. Se traduce lo descrito en el diseño a una forma legible por la máquina.
4. Pruebas. Se comprueba que se cumplen los criterios de corrección y calidad. Las pruebas deben garantizar el correcto funcionamiento del sistema.
5. Mantenimiento. Hay que asegurar que el sistema pueda adaptarse a los cambios.

* Modelos de ciclo de vida. Los modelos más importantes son: Cascada Incremental y Evolutivo.

**CICLO DE VIDA EN CASCADA.**

Las etapas para el desarrollo de software tienen un orden de forma que para empezar una etapa hay que finalizar la anterior. Después de cada etapa se realiza una revisión para poder pasar a las siguiente.

Este modelo permite hacer iteraciones, por ejemplo, durante la etapa del mantenimiento del producto el cliente requiere una mejora, esto implica que debe modificarse algo el diseño lo que implica hacer cambios en la codificación y se tendrán que realizar de nuevo las pruebas.



Modelo la cascada con realimentación

Tiene varias variantes una de ellas es la que produce una realimentación. Por ejemplo, supongamos que la etapa de Análisis. ha finalizado y se puede pasar a la de diseño. Durante el desarrollo de esta etapa se detectan fallos, entonces habrá que retornar a la etapa anterior, realizar los ajustes pertinentes y continuar de nuevo con el diseño.

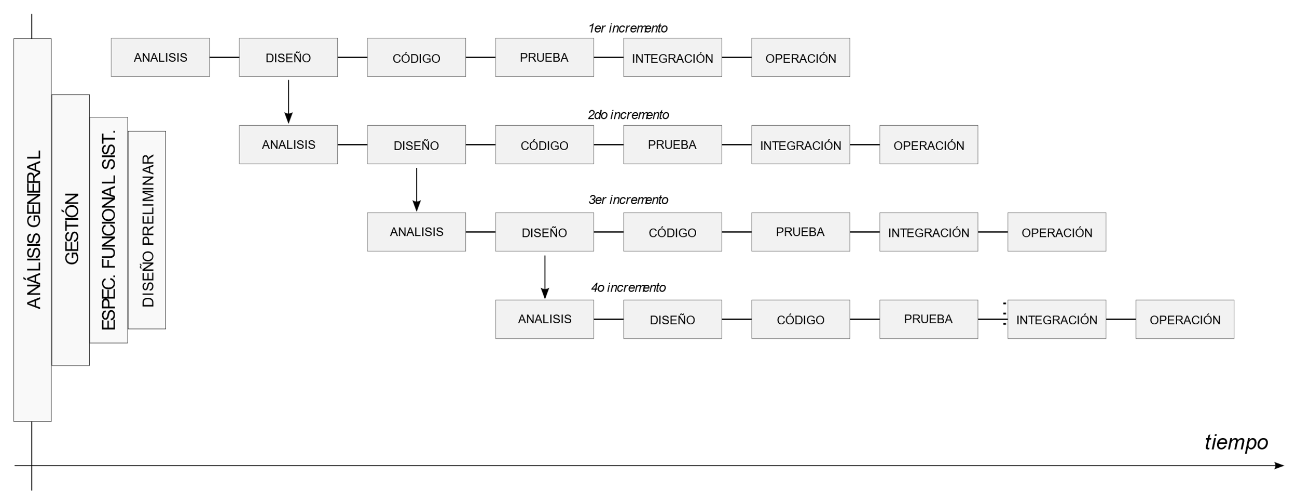
**MODELOS EVOLUTIVOS.**

El software evoluciona con el tiempo, es normal que también cambien los requisitos El modelo en cascada asume que se va a entregar un producto completo en cambio los modelos evolutivos permiten desarrollar versiones cada vez más completas

Los modelos evolutivos más conocidos son: el iterativo incremental y el Espiral

**MODELO ITERATIVO INCREMENTAL**

Está basado en varios ciclos cascada realimentados, aplicados repetidamente. Entrega el software en pequeñas partes, pero utilizables, llamadas incrementos. Cada incremento se construye sobre aquel que ya ha sido entregado.



Ejemplo un procesador de texto. En el primer incremento se desarrollan funciones básicas de gestión de archivos; en el segundo incremento se desarrollan funciones gramaticales y de corrección ortográfica en el tercer incremento funciones avanzadas de paginación y así sucesivamente.

**MODELO EN ESPIRAL**

Combina el modelo en cascada con el modelo iterativo. De construcción de prototipos. El proceso de desarrollo se representa como una espiral donde en cada ciclo se desarrolla una parte del mismo. Cada ciclo está formado por cuatro fases.

1. Determinar objetivos. Se identifican los objetivos las alternativas para alcanzar los objetivos y las restricciones impuestas.
2. Análisis de riesgos. Evalúa las alternativas en relación con los objetivos y limitaciones. Con frecuencia en esta fase se identifican los riesgos involucrados y si es posible la manera de resolverlos. Un riesgo puede ser cualquier cosa: requisito no comprendido, mal diseño, errores de implementación.
3. Desarrollar y probar. Desarrollar la solución a este problema y verificar que es aceptable.
4. Planificación. Revisar y evaluar todo lo que se ha hecho.

**Ventajas**

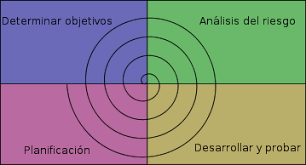
* No requiere una definición completa.
* Análisis del riesgo en todas las etapas.
* Reduce riesgos del proyecto.

**Inconvenientes**.

* Es difícil evaluar los riesgos.
* El costo del proyecto aumenta a medida que la espiral pasa por sucesivas etapas.

**Se recomienda para.**

* Proyectos de gran tamaño
* Proyectos donde sea importante el factor riesgo.



FASES DEL DESARROLLO DE UNA APLICACIÓN

Antes de desarrollar un proyecto software hay que elegir un modelo de ciclo de vida. Será necesario examinar las características del proyecto para elegir un modelo u otro. Independientemente del modelo elegido hay una serie de etapas que se deben seguir para construir un proyecto de calidad. Estas etapas son:

1. Análisis
2. Diseño
3. Codificación
4. Pruebas
5. Documentación.

**ANÁLISIS**

Lo más importante del éxito de un proyecto software es entender y comprender el problema que se necesita resolver y una vez comprendido darle solución. En esta fase se analizan y especifican los requisitos o capacidades que el sistema debe tener.

La obtención de requisitos no es tarea fácil ya que el cliente puede no tenerlos claros, pueden surgir nuevos requisitos pueden existir malentendidos, el cliente puede no expresarse de forma clara etc. Para facilitar esta comunicación se utilizan varias técnicas algunas de ellas son las siguientes.

* Entrevistas
* Desarrollo conjunto de aplicaciones (JAD)
* Planificación conjunta de requisitos (JRP)
* Brainstorming
* Prototipos
* Casos de uso

***ENTREVISTAS***

Consiste en hablar con el cliente

***DESARROLLO DE APLICACIONES (JAD)***

Es un tipo de de entrevistas muy estructurada aplicable a grupos de personas (usuarios, administradores, analistas, desarrolladores.). Cada persona juega un rol determinado y todo lo que se hace está reglamentado.

***PLANIFICACIÓN CONJUNTA DE REQUISITOS (JRP)***

Es un conjunto de JAD, se caracterizan por estar dirigidas a la alta dirección y los productos resultantes son los requisitos de alto nivel.

***BRAINSTORMING***

Tipo de reuniones en grupo cuyo objetivo es generar ideas desde puntos de vista diferentes puntos de vista para la resolución de un problema. Se emplea al principio del proyecto pues permite explorar un problema desde muchos puntos de vista.

***PROTOTIPOS***

Versión inicial del sistema, se utiliza para clarificar algunos puntos y demostrar los conceptos. Después se tira o se usa como base para añadir cosas.

***CASOS DE USO***

Se basa en la representación de escenarios que describen el comportamiento deseado del sistema. Representa requisitos funcionales. Es importante resaltar que describen qué hacen. No como lo hacen.

Se especifican dos tipos de requisitos.

* Requisitos funcionales. Describen la función que realiza el sistema, como reacciona ante determinadas entradas, como se comporta en situaciones particulares, etc.
* Requisitos no funcionales. Tratan sobre las características del sistema, como pueden ser la mantenibilidad, sistemas operativos, plataforma hardware, restricciones limitaciones, etc.

La siguiente tabla muestra ejemplos de requisitos funcionales y no funcionales.

|  |  |
| --- | --- |
| **Requisitos funcionales** | **Requisitos no funcionales** |
| El usuario puede agregar un nuevo contacto | La aplicación debe funcionar en sistemas operativos Linux y Windows |
| El usuario puede ver una lista con todos los contactos | El tiempo de respuesta a consultas altas, bajas y modificaciones ha de ser inferior a 5 segundos. |
| A partir de la lista de contactos el usuario puede acceder a un contacto. | Utiliza un sistema gestor de bases de datos para almacenar los datos. |
| El usuario puede eliminar un contacto o varios de la lista | Utiliza un lenguaje multiplataforma para el desarrollo de la aplicación- |
| El usuario puede modificar los datos de un contacto | La interfaz de usuario es a través de ventanas |
| El usuario puede imprimir la lista de contactos | El manejo de la aplicación se realiza con el teclado y el ratón. |
| El usuario puede seleccionar determinados contactos | Espacio libre en disco, mínimo 1GB  Máxima capacidad de memoria 2GB |

**---------------------------------------------------------------------------------------------------------**

**PRÁCTICA** 1: Realiza una tabla similar a la anterior en la que se recojan los requisitos funcionales y no funcionales correspondientes al problema que te describo en las siguientes líneas. Esta práctica es evaluable y debe subirla a la plataforma en los plazos y forma que tu profesor te indique.

El objetivo del sistema es facilitar la gestión de una biblioteca mediana, en lo referente a la atención directa a los usuarios. Esto incluye, fundamentalmente, el préstamo de libros, así como la consulta bibliográfica

El sistema a desarrollar se denominará BIBLIO-1. Y consistirá en un único programa que realizará todas las funciones necesarias. En particular, deberá realizar las siguientes funciones.

* Gestión del préstamo y devolución de libros
* Consulta bibliográfica por título, autor o materia.

El sistema no ha de soportar la gestión económica de la biblioteca, ni el control de adquisición de libros, ni otras funciones no relacionadas directamente con la atención al usuario.

El sistema facilita la atención al público por parte de un único bibliotecario. Que podrá atender a todas las funciones.

La biblioteca dispone de una colección de libros a disposición del público. Cualquier persona puede consultar los libros en la sala de lectura, sin necesidad de realizar ningún registro de esta actividad.

Los libros pueden ser también sacados en préstamo por un plazo limitado. Fijado por la organización de la biblioteca y siempre el mismo. En este caso es necesario mantener anotados los libros prestados y qué lector los tiene en su poder. Para que un lector pueda sacar un libro en préstamo debe disponer previamente de una ficha de lector con sus datos, y en particular con indicación de su teléfono, para facilitar la reclamación del libro en caso de demora en su devolución

Un lector puede tener a la vez varios libros en préstamo, hasta un máximo fijado por la organización de la biblioteca, igual para todos los usuarios.

Los usuarios deben disponer de algunas facilidades para localizar el libro que desea, tanto si es para consultarlos en la sala como si es para sacarlos en préstamo. Se considera razonable poder localizar un libro por su autor, su título ( o parte de él) o por la materia de que trata. Para la búsqueda por materias, existirá una lista de materias establecidas por el bibliotecario. Cada libro podrá tratar de una o varias materias.

El objetivo del sistema es facilitar las funciones más directamente relacionadas con la atención directa a los usuarios de la biblioteca. Las funciones principales serán.

* Anotar los préstamos y devoluciones
* Indicar los préstamos que hayan sobrepasado el plazo establecido.
* Búsqueda por autor, título o materia

Como complemento serán necesarias otras funciones, en concreto.

* Mantener registro de los libros
* Mantener un registro de los usuarios (lectores)

El programa deberá poder ejecutarse en máquinas de poca capacidad y con sistemas operativos sencillos, sin soporte de multitarea, tales como MS-DOS.

***REPRESENTACIÓN DE REQUISITOS.***

Para representar los requisitos se emplean varias técnicas.

1. **Diagramas de flujo de datos. DFD**. Es un diagrama que representa el flujo de datos entre los distintos procesos, entidades externas y almacenes que forman el sistema. Los procesos identifican funciones dentro del sistema, se representan mediante burbujas ovaladas o circulares. Las entidades externas representan componentes que no forman parte del sistema (por ejemplo una persona, un departamento, etc) pero proporcionan datos al sistema o los reciben de él. Las entidades externas se representan mediante rectángulos. Los almacenes representan los datos procesados o desde donde se recuperan. Se representan mediante dos líneas horizontales y paralelas.







Información Entrada Datos Intermedias Datos Intermedios

Informac. Salida



Datos Intermedios

Salida del Almac.

Información de entrada

Entrada en Almac.



ALMACENAMIENT0

2**.- Diagramas de** transición de estados, DTE. Representa cómo se comporta el sistema como consecuencia de sucesos externos.

3.-**Diagrama Entidad /Relación DER**. Se emplea para representar los datos y la forma en que se relacionan entre ellos.



(1,1)

(2,M) (0,1) (1,M)

(1,M) 

3.-**Diccionario de datos DD.** Es una descripción detallada de los datos usados por el sistema que gráficamente se representan por los diagramas de flujo de datos y almacenes presentes en el DFD

## Todo lo realizado en esta fase debe quedar reflejado en el documento de Especificación de Requisitos de Software (ERS). Este documento no debe tener ambigüedades, debe ser completo, consistente y fácil de verificar y modificar.

ANALISIS ESTRUCTURADO. Diagramas de flujo de datos

Un DFD es un diagrama que representa los flujos de datos y las transformaciones que se aplican sobre ellos al moverse desde la entrada hasta la salida. Se utiliza para modelar las funciones y los datos que fluyen entre ellas. El sistema se modeliza mediante un conjunto de DFD nivelados en el que los niveles superiores definen las funciones del sistema de forma general y los niveles inferiores definen esas funciones en niveles más detallados.

**Componentes de un DFD**

Los componentes de un DFD son.

* Procesos. Son las funciones del sistema.
* Almacenes. Representan datos almacenados
* Entidades externas. Fuente y/o destino de la información.
* Flujo de datos. Datos que fluyen entre las funciones.

PROCESOS

El proceso representa una función que transforma los datos de entrada en uno o varios flujos de datos de salida El procesos debe ser capaz de generar los flujos de datos de salida a partir de los flujos de datos de entrada más una información local (regla de conservación de datos).

La representación se realiza mediante un círculo en cuyo interior colocamos un número y un nombre.

ALMACENES DE DATOS

Es un almacén lógico de almacenamiento y puede representar cualquier dato temporalmente almacenado. Se representa por dos líneas paralelas.

* Todos los almacenes deberán llevar un nombre.
* Un almacén se puede representar varias veces es un DFD.
* Dentro de un conjunto de DFD nivelados el almacén se situará en el nivel más alto

ENTIDADES EXTERNAS.

Es un componente que representa un generador o consumidor de información y que no pertenece al mismo.

FLUJOS DE DATOS

Es un camino a través del cual viajan datos de una parte del sistema a otra.

* Flujo de consulta. Muestra la utilización de la información del almacén
* Flujo de actualización Indica que el proceso va a alterar la información.
* Flujo de diálogo. Representa como mínimo un flujo de consulta y uno de actualización.







F. de consulta F.Actualización F.Dialogo.



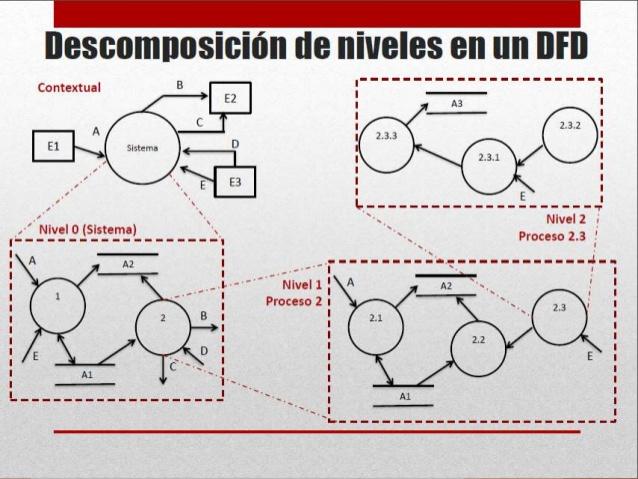
DESCOMPOSICIÓN EN NIVELES .

Como el DFD puede ser grande no se puede representar en una sola página. Se representa por capas y cada capa queda definida mediante un DFD. Se sigue una aproximación descendente en la que cada nivel proporciona una visión más detallada.

Se comienza por el nivel más alto mediante un DFD denominado diagrama de contexto. En este diagrama sólo hay un proceso que representa el sistema completo.. A continuación este proceso se descompone en otro DFD que se denomina diagrama del sistema en el que se representan las funciones principales del sistema. A continuación se descompone cada uno de los procesos en nuevos diagramas que representan funciones más simples.

Por tanto un DFD queda definido por.

* Diagrama de contexto. Único y en la parte superior
* Niveles medios.formado por el resto de diagramas.
* Funciones primitivas. Presentes tanto en los niveles intermedios como en los últimos niveles de la jerarquía y que se corresponden con procesos que no se explotan en nuevos DFD.



**PRACTICA 2**.-Realizar la descomposición de DFD de la siguiente receta de cocina.

Ingredientes.

1 Kg de costillas de ternera, 200 gr de champiñones, 2 cucharadas de zumo de limón, 75 gr de mantequilla, 3 zanahorias, 2 cebollas. I rama de apio, 1 ramito de hierbas, 250 gr de cebolletas, 1 cucharada de harina, 3 yemas , 6 cucharadas de nata líquida, sal , pimienta, nuez moscada, perejil picado.

Preparación de blanqueta de ternera

1. Retirar los pies terrosos de los champiñones, lavarlos cuidadosamente, cortarlos en lonchas. Se cuecen 1 minuto con una cucharada de zumo de limón, 2 de agua, 1 nuez de mantequilla, sal y pimienta. Se cuelan los champiñones y se reserva el líquido
2. Poner la carne cortada en dados en una cazuela, cubrirla con agua fría y añadir el líquido de cocer champiñones. Cuando hierva, espumar la superficie y añadir las cebollas peladas y troceadas, las zanahorias en rodajas, el apio en tallos y el ramito de hierbas. Cocer hasta que la carne esté tierna (algo más de una hora) y sazonar con sal y pimienta.
3. Cuando falte ½ horas, pelar las cebolletas, hacer unas incisiones en ellas y cocerlas 20 minutos en poca agua y una cucharada de mantequilla. Retirar la carne de la cazuela y ponerla en un recipiente aparte con los champiñones y las cebollas.

Dorar la harina a fuego suave en el resto de la mantequilla e incorporar a la salsa de la carne. En un cuenco aparte mezclar las yemas con la nata, el zumo de limón y la punta de un cuchillo de nuez moscada rallada. Devolver la carne con las cebolletas y champiñones a la cazuela y verter la salsa por encima . Por último salpicar la superficie con perejil picado..

**PRÁCTICA 3**.-Realizar la descomposición mediante DFD del siguiente caso.

Petición de libros.

Un usuario puede realizar una petición de uno o más libros de la biblioteca. Para ello es necesario presentar el carnet de usuario de la biblioteca y una ficha en la que se detallan los libros pedidos. Puede haber varios tipos de préstamo (préstamo de sala, colaborador, proyecto fin de carrera, doctorado) en función de los cuales el usuario puede disponer de los ejemplares un periodo de tiempo específico, como se indica en la siguiente tabla.

|  |  |
| --- | --- |
| **SALA** | **El día de la petición** |
| **colaborador** | **una semana** |
| **proyecto fin de carrera** | **quince días** |
| **doctorado** | **un mes** |

**una vez entregados el carnet y la FICHA, el sistema comprobará y aceptará la petición de los libros solicitados siempre que pueda satisfacer la petición, es decir cuando haya libros disponibles. Si se acepta la petición se actualiza el número de unidades de los libros de la biblioteca y se guarda la ficha de préstamo.**

Devolución de libros.

Un usuario no puede realizar más peticiones hasta que haya efectuado todas las devoluciones de la petición anterior ya que para hacer la nueva petición, necesita el carnet, que no se le entrega hasta que no haya devuelto todos los libros. Cuando un usuario realice una devolución, el sistema actualiza el stock de libros y comprobará la fecha de devolución de cada ejemplar para estudiar, en el caso de que la devolución se realice fuera de tiempo, la imposición de una sanción que tiene un coste de X unidades monetarias por cada ejemplar y días de retraso en la devolución. En este caso la sanción se emite cuando el usuario entrega el último ejemplar.

El bibliotecario se encarga de las altas y bajas de los libros de la biblioteca.

ANALISIS ESTRUCTURADO. DICCIONARIO DE DATOS

Podemos definir un diccionario de datos (DD) como una lista organizada de los datos utilizados por el sistema que gráficamente se encuentran representados por los flujos de datos y almacenes.

El DD se crea a la vez que los DFD durante el análisis. Las entradas son realizadas cada vez que se identifica un elemento y pueden ser de tres tipos: flujos de datos, almacenes y datos elementales.

Las componentes son definidas mediante componentes más detallados. Se incluyen los siguientes símbolos

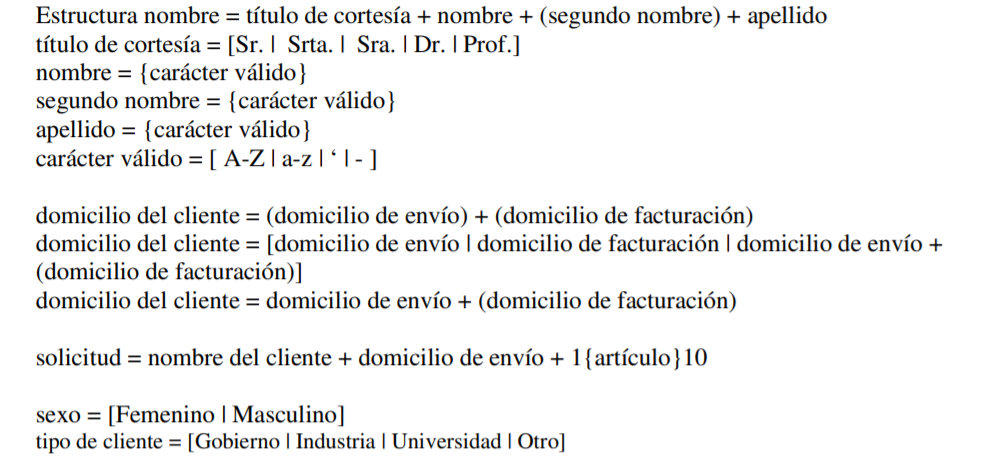
|  |  |
| --- | --- |
| SÍMBOLO | SIGNIFICADO |
| = | Composición: Está compuesto de |
| + | Y |
| [] | Selección. Selecciona una de las opciones encerradas entre corchetes. |
| () | Opción. El componentes encerrado es opcional |
| \*texto\* | Comentario. El texto entre asteriscos es un comentario |
| {} | Iteración. Iteran los componentes encerrados entre las llaves |
| @ | Identificado. Señala un campo o conjunto de campos que identifica cada ocurrencia de un almacén. |

EJEMPLO: Supongamos que tenemos una tabla que almacena datos con los pedidos que realiza un cliente podemos recogerlo en el diccionario de datos de la siguiente forma.

Idpedido + idcliente + {línea pedido]+(dirección entrega)+gastos envío+formapago+fecha pedido+fecha entrega.

Observar que línea de pedido puede ser múltiple, si el cliente realiza más de un pedido y la dirección de entrega es opcional dado que puede estar en otra tabla con los datos del cliente.

EJEMPLOS.



**PRÁCTICA 4.**Realiza el DD de las siguientes entidades.

Entidad Libro. Contiene los datos de identificación del libro.

Entidad Lector. Contiene datos personales del lector.

Entidad Materia. Contiene el término clave que contiene la materia.

Relación Prestado a . Enlaza un libro con el lector que lo ha sacado en préstamo. Contiene la fecha del préstamo.

Relación trata de. Enlaza un libro con cada materia de la que trata.

ANALISIS ESTRUCTURADO .ENTIDAD/RELACIÓN.

**Elementos del modelo Entidad/Relación**

Los elementos básicos del modelo E/R son cuatro: entidades, relaciones, atributos y dominios.

* Las entidades representan conjuntos de elementos con existencia propia que se caracterizan por las mínimas propiedades. Generalmente son personas, cosas, lugares,..
* Los atributos se utilizan para detallar las entidades dándoles propiedades descriptivas tales como nombre, color y peso. Existen dos tipos de atributos: identificadores y descriptores. Los primeros se utilizan para distinguir de manera única las ocurrencias de una entidad mientras que los descriptores se utilizan para describir una ocurrencia de entidad.
* Las relaciones representan asociaciones del mundo real entre una o más entidades. Las entidades se caracterizan por su nombre. el grado (número de entidades que participan en la relación) el tipo de correspondencia (número máximo de ejemplares de las otras entidades en la relación)
* Las jerarquías nos proporcionan un mecanismo de abstracción que permite descomponer una entidad en subtipos.

Ejemplo. Supongamos que disponemos de las entidades profesor, aula y alumno y queremos saber cuándo y en qué aula un profesor da clases a un alumno.





(1,n)

(1,n)

(1,n)

(1,n)

**PRÁCTICA 5.**

Se desea diseñar una base de datos para una universidad que contenga información sobre los alumnos, las asignaturas y las carreras que se pueden estudiar. Construir un modelo E-R teniendo en cuenta las siguientes restricciones.

* Un alumno puede estar matriculado en muchas asignaturas
* Una asignatura sólo puede pertenecer a una sola carrera
* Una carrera puede tener muchas asignaturas.

**PRÁCTICA** **6.**

Se desea diseñar una base de datos para una universidad que contenga información sobre los alumnos. Las asignaturas y los profesores. Tengan en cuenta las siguientes restricciones.

* Una asignatura puede estar impartida por muchos profesores (no a la vez) que pueden existir grupos
* Un profesor puede dar clase en muchas asignaturas.
* Un alumno puede estar matriculado en muchas asignaturas
* Se necesita tener constancia de las asignaturas en las que está matriculado un alumno, la nota obtenida y el profesor que las ha calificado
* También es necesario tener constancia de las asignaturas que imparten todos los profesores (independientemente de si tiene algún alumno matriculado en su grupo).
* No existen asignaturas con el mismo nombre
* Un alumno no puede estar matriculado en la misma asignatura con dos profesores distintos.

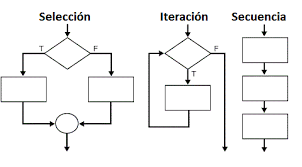
**DISEÑO**

Una vez identificados los requisitos es necesario componer la forma en que se solucionara el problema. En esta etapa se traducen los requisitos en una representación del software.

Hay dos tipos de diseño: diseño estructurado que está basado en el flujo de datos a través del sistema y el diseño orientado a objetos donde el sistema se entiende como un conjunto de objetos que tiene propiedades y comportamientos, además de eventos que activan operaciones que modifican el estado de los objetos.

Uno de los productos del diseño estructurado es diseño a nivel de componente o diseño procedimental. El resultado de esta tarea es el diseño de cada componente software con un nivel de detalle suficiente.

Se emplea un conjunto de construcciones lógicas: secuencial, condicional y repetitiva.



**PSEUDOCÓDIGO.**

**El** pseudocódigo utiliza texto descriptivo para realizar el diseño de un algoritmo.

---------------------------------------------------------------------------------------------

SECUENCIAL

Instrucción 1

Instrucción 2

----

Instrucción n

CONDICIONAL

SI <condición> ENTONCES

<Instrucciones>

SÍ NO

<instrucciones>

FIN SI

CONDICIONAL MÚLTIPLE

SEGÚN SEA <variable> HACER

CASO valor 1:

<instrucciones>

CASO valor 2:

<instrucciones>

OTRO CASO

<instrucciones>

FIN SEGÚN

FIN CASO

REPETIR – HASTA

REPETIR

<instrucciones>

HASTA QUE <condición>

HACER – MIENTRAS

MIENTRAS <condición> HACER

<instrucción>

FIN MIENTRAS

El siguiente ejemplo muestra el proceso de lectura y tratamiento de archivos secuenciales

INICIO

Abrir Fichero

Leer Registro del Fichero

MIENTRAS no sea Fin de Fichero HACER

Procesar Registro leído

Leer Registro del fichero

FIN MIENTRAS

Cerrar Fichero

FIN

**PRÁCTICA 6**

1. Escribe un programa en pseudocódigo que lea un número y si es par muestre el mensaje “ES PAR” y si es impar el mensaje “ES IMPAR”
2. Escribe un programa que lea un número del 1 al 7 e imprima una de las palabras, según el número leído, el día de la semana.
3. Escribe un programa que lea un número y diga si es múltiplo de 3
4. Escribe un programa que lea un número y calcule su factorial

**Diseño orientado a objetos.**

El diseño orientado a objetos requiere que en el análisis se hayan definido las clases que son importantes para el problema así como las operaciones, atributos asociados, relaciones y comportamiento asociado.

**CODIFICACIÓN**

Una vez realizado el diseño se realiza el proceso de codificación. En esta etapa el programador recibe las especificaciones del diseño y las transforma en un conjunto de instrucciones escritas en un lenguaje de programación. Este conjunto de instrucciones se llama código fuente.

El programador debe conocer la sintaxis del lenguaje a emplear para la codificación. Como ejemplo de esta fase realizaremos algunos programas sencillos en PHP. En el campus virtual tienes un resumen de la sintaxis de PHP.

**PRÁCTICA 7.**

1. Escribe tu nombre y apellidos.
2. Leer un número del 1 al 7 y escribir los días de la semana.
3. Lee un número y si es 1 calcula si es par, si es 2 calcula si es múltiplo de 3.
4. Escribe un programa que conozca las horas trabajadas y el precio de la hora calcule el salario semanal.

NOTA. Para realizar estos programas debes pasar los valores que se requieren variables dado que no sabemos leer de un formulario los datos que introduce el usuario. (No es el objetivo de este curso)

**PRÁCTICA 8.**

1. Realiza los ejercicios c) y d) con funciones.

**PRUEBAS**

En esta etapa ya se dispone del software y se trata de encontrar errores, no solo de codificación sino también de diseño.

Durante esta etapa se realizan pruebas de validación y verificación

1. Verificación: Actividades que tratan de comprobar si se está construyendo el producto correctamente. Es decir, si el software implementa correctamente una función dada.
2. Validación: Actividades que comprueban si el producto s e ajusta a los requisitos

La verificación comprueba si el software está bien construido y la validación si hace lo que debe o se espera de él.

El objetivo de esta etapa es planificar y diseñar las pruebas que saquen a la luz diferentes tipos de errores.

Las recomendaciones para las pruebas son.

* Cada prueba debe definir los resultados de salida esperados
* Un programador debe evitar probar sus propios programas
* Es necesario revisar los resultados de cada prueba en profundidad.
* Las pruebas deben incluir tanto datos de entrada válidos y esperados como inesperados y no válidos.
* Evitar hacer pruebas que no estén documentadas
* La probabilidad de encontrar un error en una parte del software es proporcional al número de errores encontrados.

Se siguen las siguientes etapas.

1. Generar un plan de pruebas
2. Diseñar las pruebas
3. Generar los casos de prueba
4. Definición de los procedimientos de prueba. Hay que especificar cómo se van a llevar a cabo las pruebas, quien lo va a realizar,..
5. Ejecución de las pruebas.
6. Evaluación. Se identifican los posibles errores producidos al comparar los resultados obtenidos con los esperados.
7. Depuración Trata de localizar y corregir los errores
8. Análisis de errores. Permite mejorar los procesos de desarrollo.

Para diseñar las pruebas se utilizan dos técnicas.

1. Prueba de caja blanca. Se centran en validar la estructura interna del programa. Se necesita conocer los detalles del código.
2. Pruebas de caja negra. Validan los requisitos funcionales sin fijarse en el funcionamiento interno.

**DOCUMENTACIÓN**

Todas las etapas de desarrollo deben estar perfectamente documentadas. La documentación se divide en dos clases.

1. La documentación del proceso. Registra el proceso de desarrollo y mantenimiento.
2. La documentación del producto. Describe el producto desarrollado. Se puede dividir el
   1. Documentación del sistema que describe el producto desde un punto de vista técnico y la documentación del usuario.

**EXPLOTACIÓN**

Después de realizar todas las pruebas se pone en marcha la explotación del sistema.

**MANTENIMIENTO**

Se define como la modificación de un producto después de la entrega para corregir los fallos, para mejorar el rendimiento o para adaptar el producto.

Existen 4 tipos de mantenimiento.

1. Mantenimiento adaptativo. Tiene como objetivo la modificación del producto por los cambios que se produzcan tanto en el hardware como el software.
2. Mantenimiento correctivo. Se corrigen los fallos.
3. Mantenimiento perfectivo. Modificación del producto para incorporar nuevas funciones.
4. Mantenimiento preventivo. Modificación del producto sin modificar las especificaciones para facilitar las tareas de mantenimiento.