Diseño de Proyectos de software en código abierto

José Manuel Godoy Giménez

email:josemanuel.godoy@hispalinux.es

1. 1 Noviembre 2002

Integrado en el proyecto de Gestión Libre

Indice

1. INTRODUCCIÓN

1.1 Histórico

Este documento nace como consecuencia del proyecto de *Gestión Libre* bajo el auspicio de *Hispalinux*. El objetivo del mismo es dar unas reglas generales para el desarrollo de software en el mundo del código abierto (open-source).

Si te has bajado este documento de un lugar distinto de hispalinux sería conveniente que compruebes que se trata de la última versión.

Esta es la versión 1.1 de Noviembre del 2002.

Autor: José Manuel Godoy Giménez (josemanuel.godoy@hispalinux.es)

1.2 Licencia

1.3 Marcas registradas

1.4 Obtención de este documento

Se puede obtener el presente documento a través de Hispalinux,

http://www.hispalinux.es

o bien en la página personal del autor:

http://www.fut.es/~jgodoy

1.5 Motivaciones

El presente documento intenta (no se si lo conseguirá), permitir al mundo del *código abierto* la posibilidad de embarcarse en proyectos de envergadura que requieran un estudio serio.

Debemos de tener en cuenta que el código abierto es extremadamente seguro y robusto, y que cuando aparece algún error (odio decir 'bug') se soluciona rápidamente, incluso en horas, sin embargo, tiene fama de poco serio. Es en estas ocasiones cuando se debe de tener en cuenta la validez de la frase "*La mujer del Cesar no sólo debe ser honrada sino que tiene que parecerlo*".

Con la mente puesta en esta frase, y dado la tendencia actual de valorarlo todo en base a la calidad debemos de tener en cuenta las etapas básicas en la planificación de la calidad, para poder aplicarlas a nuestra aplicación software:

1. Determinar los clientes
2. Determinar sus necesidades, expectativas e intereses
3. Transformar las necesidades en requerimientos del producto
4. Establecer los procedimientos para generar los productos que cumplan las especificaciones
5. Implantar los procedimientos

6. Obtención de los productos previstos.

Estas etapas son las que se tratará de definir e identificar a lo largo de este documento, el cual espero que no se os atragante .

1.6 Sugerencias críticas y aportaciones

Las preguntas, comentarios correcciones y sugerencias serán siempre bien recibidas, sobre todo en las primeras versiones de este documento, estas pueden ser dirigidas a josemanuel.godoy@hispalinux.es

1.7 Situación de este documento

La versión que estas leyendo es la versión 1.1 sin embargo no es el documento final. Le faltan por desarrollar los ejemplos de UML que ayudarán a comprender este lenguaje de modelado.

También es idea del autor realizar una serie de apéndices que permitan localizar rápidamente los documentos , gráficos y técnicas de modelado que se utilizan en este documento.

Si te animas puedes colaborar, no te cortes :-) .

Evolución del documento

1. Versión 0.0 Análisis Estructural.
2. Versión 1.0 Ejemplo de Análisis Estructural Gegaco
3. Versión 1.1 Se incluye descripción teórica del lenguaje de modelado UML

1.8 Agradecimientos

Agradezco los apoyos prestados por Fernando Acero, coordinador del proyecto Gestión Libre de Hispalinux, así como los comentarios y rollos que me ha prestado Carlos Moreno (Neurostar).

Así como a todos los que han colaborado con correcciones, sugerencias y enumeración de puntos oscuros de la primera versión de este documento.

2. PROYECTOS SOFTWARE

2.1 Comienzo de un proyecto software

Antes de poder empezar a planificar un proyecto, deben establecerse los ámbitos y los objetivos, considerar soluciones alternativas e identificar las restricciones técnicas y de gestión. Sin tener esta información clara, es imposible obtener un identificación realista de las tareas del proyecto o un plan de trabajo adecuado que proporcione una indicación significativa del progreso.

Los objetivos identificarán los fines globales sin considerar cómo se llegará a esos fines. El ámbito identificará las funciones primordiales que deben llevar a cabo el software, intentando limitarlas de forma cuantitativa, es decir, que debe hacer y que NO debe hacer el sistema software.

Tras comprender los objetivos y el ámbito del proyecto, se han de considerar las soluciones alternativas, ya que ésto permitirá al desarrollador seleccionar el 'mejor' enfoque, dadas las restricciones de fechas, presupuestos (recordar que software libre # gratis), colaboradores que programan, etc.

2.2 Métrica del software orientada a la función

El diseño de una aplicación software es un desafio técnico, y como tal, el hecho de medir sus características y funcionalidades ayudará a la comprensión del proceso que se utiliza para desarrollar un producto.

Dado que frecuentemente la medición conlleva controversia y discusión, en este documento se adoptará una métrica orientada a la función. Las métricas de software orientadas a la función son medidas indirectas del software y del proceso por el cual se desarrolla y se centran en la funcionalidad o utilidad del programa.

Para conseguir evaluar la productividad del programa se utilizan puntos de función, que son una serie de valoraciones subjetivas de la complejidad del software.

Los valores de la información se definen:

- *Número de entradas de usuario:* Se cuenta cada entrada de usuario que proporciona al software diferentes datos orientados a la función.
- *Número de salidas de usuario*: Se refiere a informes, pantallas, mensajes de error, etc. Los elementos dentro de un informe no se cuentan por separado.
- *Número de peticiones al usuario:* Se consideran como las entradas que debe realizar el usuario, generadas como respuesta a una salida del programa. Cada petición se contará por separado.
- *Número de archivos*
- *Número de interfaces externas:* Interfaces legibles por la máquina (archivos de datos, discos, cintas, scaner...) utilizados para transmitir información a otro sistema.

Un documento tipo que recoja este estudio sería:

Parámetro de medida	Valor	Factor de peso			Total
		Simple	Medio	Complejo	
Entradas de usuario		3	4	6	
Salidas usuario		4	5	7	
Peticiones al usuario		3	4	6	
Nº de archivos		7	10	15	
Nº de interfaces		5	7	10	
				Total	

2.3Reusabilidad

No hay un estudio de recursos software si no se considera la reusabilidad, que es la creación y reutilización de bloques constructivos de software. Esto es especialmente cierto centrándonos en el caso del código abierto (open source o Free software). Estos bloques constructivos deberían catalogarse para una fácil referencia, estandarizados para una fácil aplicación y validados para facilitar la integración. Esto nos da dos máximas que debemos seguir todos los desarrolladores de código abierto:

1. Si el software existente satisface los requisitos lo usaremos.
2. Si el software existente requiere alguna modificación para su integración en el sistema, esta se hará con un estudio previo ya que puede ser más económico desarrollar un software nuevo.

2.3 Plan del proyecto

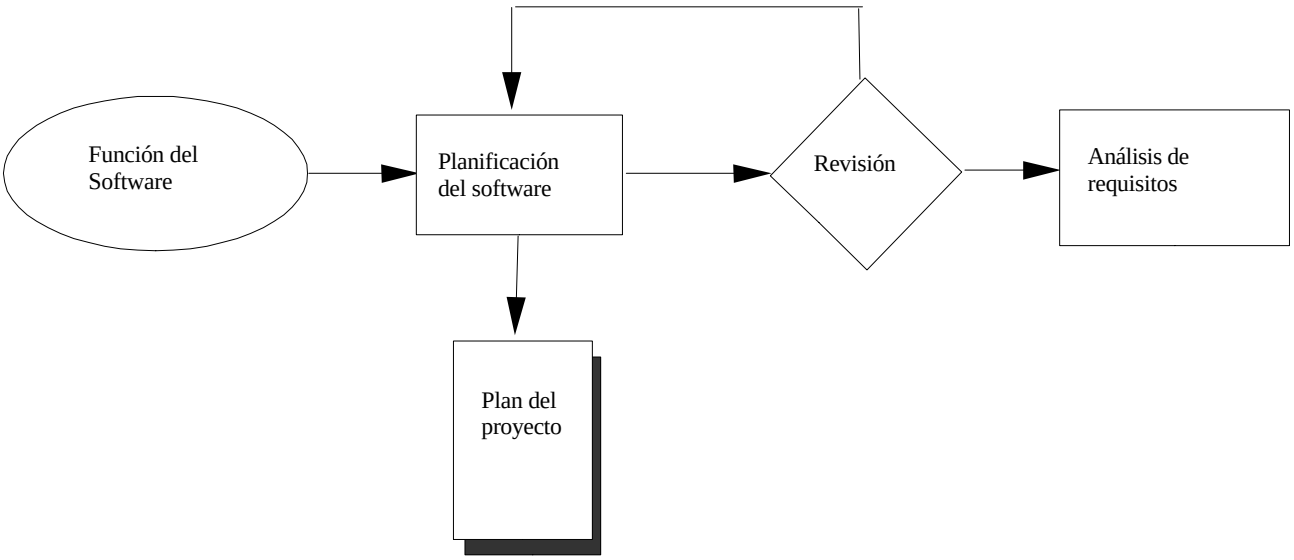
En todo proyecto software debe existir documentación que permita evaluar y trabajar de forma paralela al mismo, y que pueda servir de base para los pasos posteriores. El plan de proyecto de software es la culminación de la etapa de planificación. Proporciona una línea de base con información de costes y de agenda.

El plan de proyecto de software es un documento relativamente breve, dirigido a una audiencia diversa y contendrá los siguientes puntos:

- Ámbito y recursos
- Riesgos (económicos, temporales, tecnológicos).
- Agenda de trabajo.
- Recursos del proyecto
- Organización del personal (estructura de los equipos, si existen).
- Mecanismos de seguridad y control.

3. ANÁLISIS DEL PROYECTO

3.1 ¿Dónde estamos?



3.2 Análisis del sistema

En un sistema por computadora se distinguen los siguientes elementos:

- Software: Programas, estructura de datos y documentación asociada.
- Hardware: Dispositivos electrónicos de computación y los de interacción con el mundo externo (impresoras, escaner, etc).
- Personal: Individuos operadores y usuarios del software y del hardware.
- Bases de datos: Colección de información accedida desde el software y que es parte integral del funcionamiento del sistema.
- Documentación: Manuales impresos y demás información descriptiva que explica el uso y/o la operación del sistema.
- Procedimientos: Los pasos que definen el uso específico de cada elemento del sistema o del contexto del mismo.

3.2.1 Identificación de necesidades

Es el primer paso en el análisis del sistema y el punto de partida; la información que se debe manejar en éste punto es:

- Funcionamiento y rendimiento requeridos
- Aspectos de fiabilidad y calidad
- Fines generales del sistema
- Limitaciones de coste/agenda
- Requisitos de fabricación
- Tecnología disponible
- Ampliaciones futuras

Toda esta información se recoge en un documento de conceptos del sistema

3.2.2 Estudio de viabilidad

Todo proyecto es realizable, siempre y cuando los recursos sean ilimitados y el tiempo infinito. Como estas circunstancias difícilmente se dan en la vida real se debe (con prudencia) evaluar la viabilidad del proyecto, éste estudio se centra en cuatro áreas:

- Viabilidad económica
- Viabilidad técnica (disponibilidad de recursos)
- Viabilidad legal
- Alternativas al desarrollo del sistema

El documento de viabilidad debería tener un esquema como el siguiente:

1. Introducción
   - Declaración del problema
   - Entorno de implementación
   - Restricciones
2. Recomendaciones de gestión e impacto
3. Alternativas
4. Descripción del problema
   - Resumen del ámbito
   - Viabilidad de los elementos
5. Análisis de costes
6. Evaluación del riesgo técnico.
7. Consideraciones legales
8. Otros aspectos específicos del sistema

3.2.3 Análisis técnico

En este periodo se analizan los méritos técnicos del concepto de sistema, a la vez que se recoge información adicional sobre el rendimiento, fiabilidad, facilidad de mantenimiento y posibilidad de producción. Normalmente incluyen una capacidad *limitada* de investigación y de diseño.

El análisis técnico comienza con la definición de la viabilidad técnica del sistema propuesto:

- Tecnologías requeridas para conseguir la funcionalidad y el rendimiento del sistema
- Estudio de nuevos materiales, métodos, algoritmos y procesos; debe incluir un estudio de riesgos de su desarrollo
- Afectación de los nuevos elementos al coste

Como conjunto de criterios para el uso de modelos durante el análisis técnico de sistemas tenemos (Blanckard y Fabrycky):

1. El modelo debe ser simple de comprender y manipular, pero debe estar cerca de la realidad operativa
2. El modelo debe realzar los factores relevantes del problema y suprimir los que no lo sean. Debe ser fiable en cuanto a repetición de resultados.
3. El diseño debe permitir modificarlo y/o expandirlo fácilmente y permitir la evaluación de factores adicionales si se requieren.

Las herramientas CASE de simulación y creación de prototipos pueden ayudar en el análisis técnico. Los resultados del análisis técnico son la base de la decisión de seguir o no con el sistema o con el planteamiento sobre el mismo.

3.3 Especificaciones del sistema

La especificación del sistema es un documento que sirve como base para el desarrollo de cualquier sistema software, incluyendo el hardware y la necesidad de bases de datos, describe la función y el rendimiento de un sistema basado en la computadora y las restricciones que gobiernan su desarrollo. También describe la información (control y datos) que sirve de entrada y salida del sistema

3.3.1 Esquema de especificación del sistema

Un esquema recomendado para la especificación sería:

1. Introducción
   A- Ambito
   B- Visión general
      I. Objetivos
      II. Restricciones

2. Descripción funcional y de datos
   A- Arquitectura del sistema
      I Diagrama de contexto de arquitectura (DCA)
      II Descripción del DCA
3. Descripción de los subsistemas
   A- Especificación del diagrama de estructura
      - Diagrama de flujo de la arquitectura
      - Narrativa del módulo del sistema
      - Rendimiento
      - Restricción de diseño
      - Asignación de comportamiento del sistema
   B- Diccionario de datos
   C- Diagramas y descripción de la interconexión de la arquitectura
4. Resultados de la modelización y simulación del sistema (si hay lugar).
   A- Modelo del sistema usado para la simulación
   B- Resultados de la simulación
   C- Aspectos especiales del rendimiento
5. Aspectos del proyecto
   A- Costes del proyecto
   B- Agenda
6. Apéndices

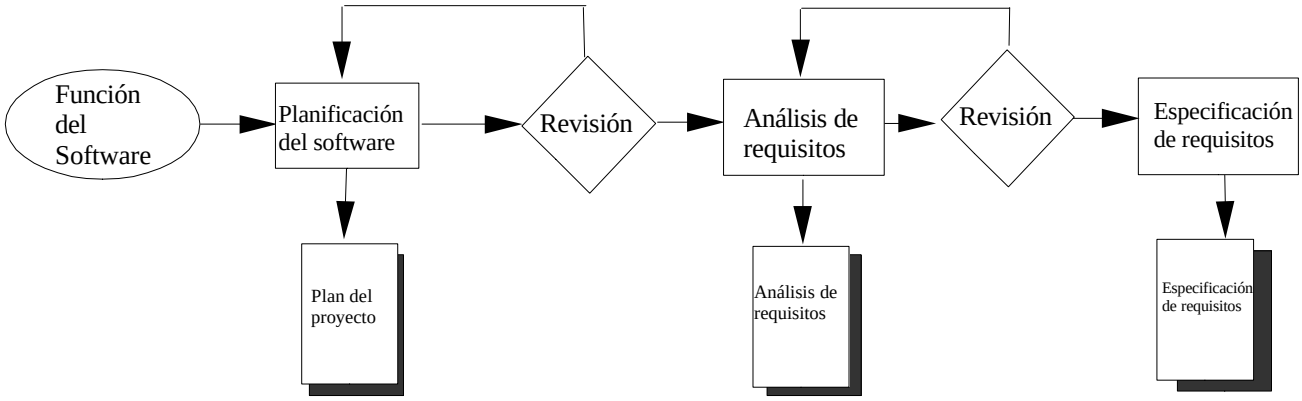
3.3.2 Diagrama de arquitectura

Como todas las técnicas de modelización utilizadas en la ingeniería del software y de sistemas, la plantilla de arquitectura permite crear una jerarquía de detalles. En el nivel superior está el diagrama de contexto de la arquitectura (DCA).

El DCA establece los límites de información entre los que se está implementando el sistema y el entorno en que va a funcionar el sistema, es decir, productos y consumidores de información del sistema y todas las entidades que se comunican a través de la interfaz.

4. ESPECIFICACIÓN DE REQUISITOS

4.1 ¿Dónde estamos?



4.2 Fundamentos

La especificación de requisitos es la tarea que establece un puente entre la asignación de software a nivel de sistema y el diseño software. Los princios de toda especificación son:

- Separar funcionalidad de implementación: Una especificación es una descripción de lo que se desea realizar, no de cómo se va a realizar.
- Una especificación debe describir un sistema tal y como es percibido por la comunidad de usuarios.
- Debe ser operativa: Para ello debe ser completa y lo suficientemente formal para que pueda determinar si se satisface o no con algunos casos de prueba amplios.
- Debe ser tolerante a la ampliación.

4.3 Notación básica

Cuando se trabaja con un sistema basado en computadora, la información fluye y se transforma. El sistema admite entradas, aplica los elementos software y humanos para transformar la entrada en salida. Por ello puede considerarse como natural especificar un sistema como un flujo de información, mediante los diagramas de flujo de datos.

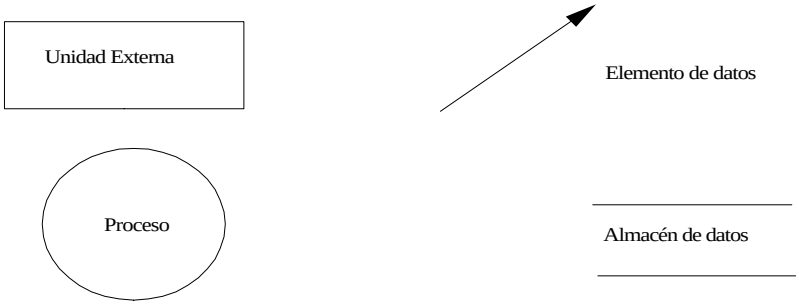
4.3.1 Diagrama de Flujo de Datos (DFD).

El DFD es una técnica gráfica que representa el flujo de información y las transformaciones que se aplican a los datos al moverse desde la entrada hasta la salida, también se conoce como diagrama de burbujas.

El DFD comienza por el nivel 0, modelo fundamental del sistema o modelo de contexto, y representa el elemento de software completo como una sola burbuja con los datos de entrada y salida representados por flechas de e/s.

Notación básica:

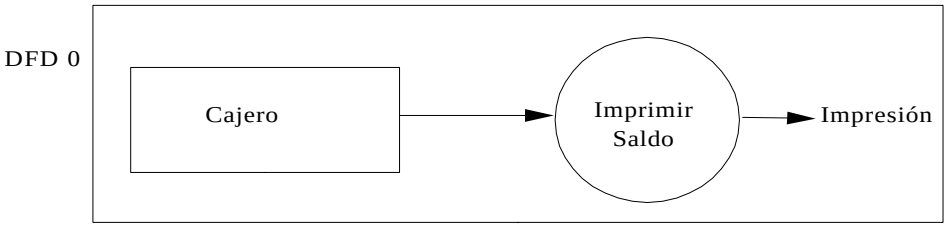
En un DFD un rectángulo representa una entidad externa, es decir, un elemento del sistema, u otro sistema que produce información. Un circulo representa un proceso o transformación que se aplica a los datos y los cambia. Las flechas indican el flujo de información y deben estar etiquetadas. La línea doble representa un almacén de información.



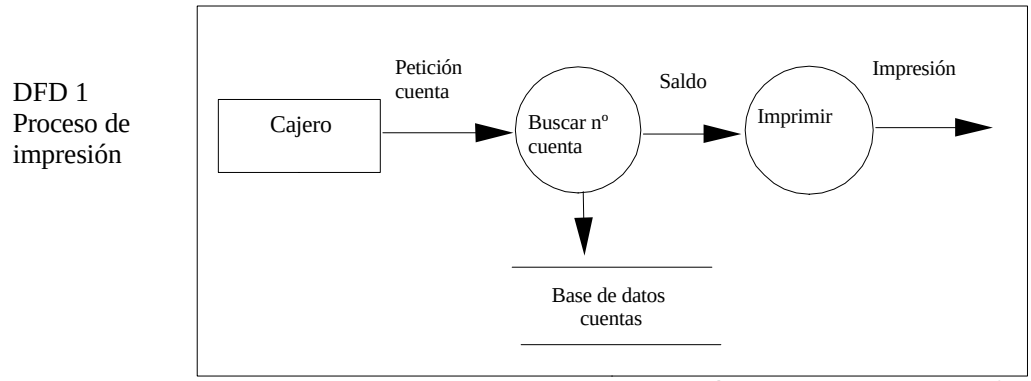
Utilización

Partiendo de una definición de la función genérica del sistema software, *nivel 0*, se van expandiendo las distintas burbujas en distintos niveles para mostrar un mayor detalle, hasta llegar a niveles de función. Es importante mantener la continuidad del flujo de información, es decir, que la entrada y la salida de cada refinamiento debe ser la misma.

Así por ejemplo en un diagrama de flujo de datos de un cajero, que indique la impresión de un resguardo si tenemos un DFD de nivel 0 cuya entrada es Cajero y la salida es Impresión cuando hagamos la explosión al nivel 1 se debe de mantener la correspondencia.



El DFD de nivel 1 que sería la explosión del proceso de impresión sería:



Como se puede observar el DFD de nivel 0, tiene una entrada de petición de cuenta y una salida en forma de impresión. Cuando explosionamos el proceso imprimir saldo en el DFD de nivel 1 mantenemos el mismo flujo de entradas y salidas.

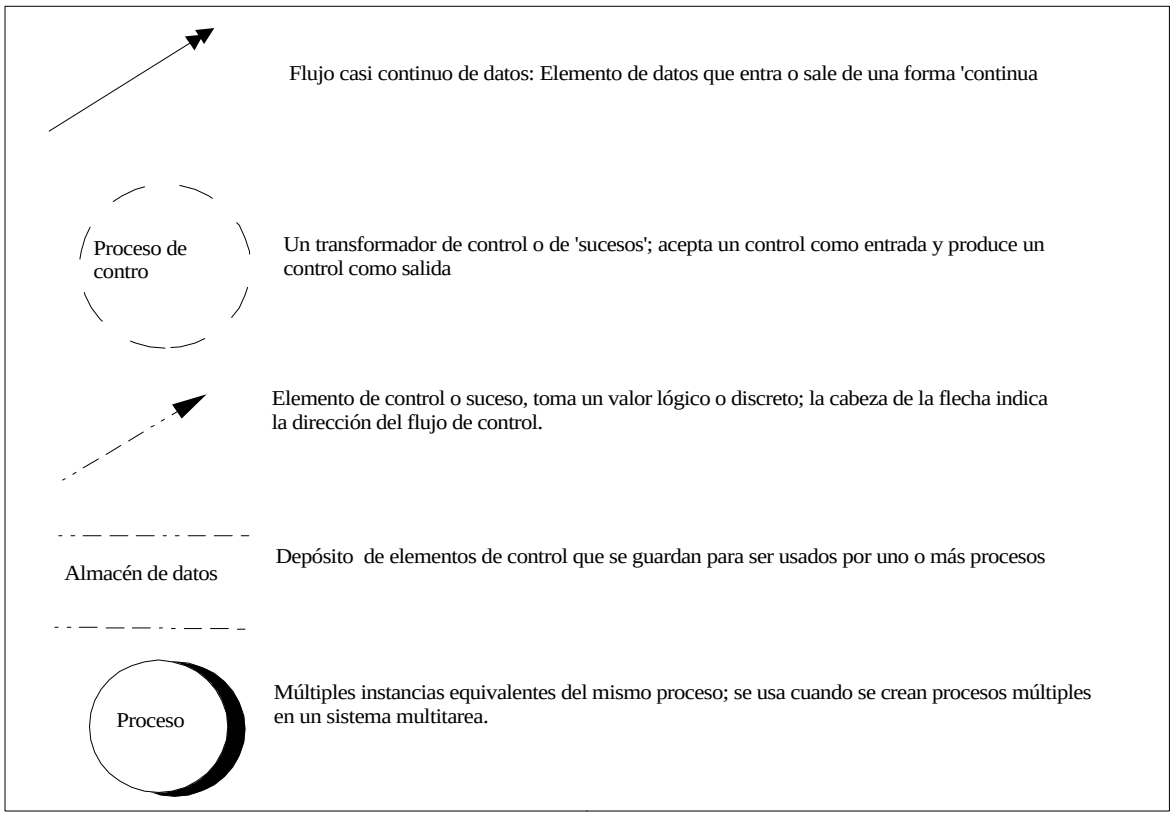
Para crear un DFD se pueden seguir estas directrices:

1. El DFD de nivel 0 debe reflejar el software/sistema como una sola burbuja.
2. Se deben de anotar cuidadosamente la entrada y salida principal.
3. Comenzar el refinamiento aislando los procesos, los objetos de datos y los almacenes de datos que sean candidatos a ser representados en el siguiente nivel.
4. Todas las flechas y burbujas se rotularán con nombres significativos.
5. Entre sucesivos niveles se debe mantener la continuidad del *flujo de información*.
6. Refinar las burbujas de una en una.

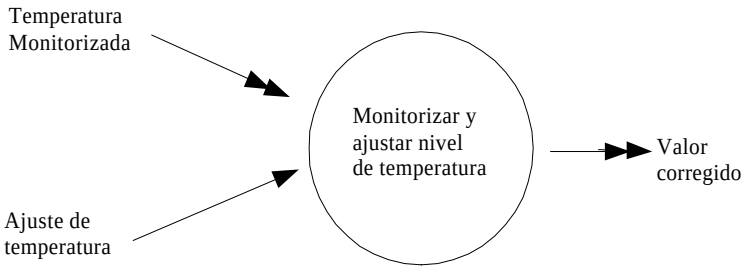
4.3.2 Diagrama de Flujo de Control

Toda aplicación está 'conducida' además de por los datos, por sucesos que complementan la visualización de la información. Los diagramas de flujo no premiten que se representen expecíficamente los flujos de control, esta exclusión es muy restrictiva (sobre todo para sistemas en tiempo real), por este motivo se ha desarrollado una notación especializada para la representación del flujo de sucesos y del procesamiento de control, esta notación y su funcionamiento es muy similar a los DFD.

- Una flecha con línea continua representa el flujo de datos.
- Una flecha con línea discontinua representa el flujo de control.
- Una burbuja con línea discontinua representa procesos que sólo manejan flujo de control.



Así por ejemplo, una estación de ajuste de temperatura tendría el siguiente DFC:



4.3.3 Modelo Entidad-Relación

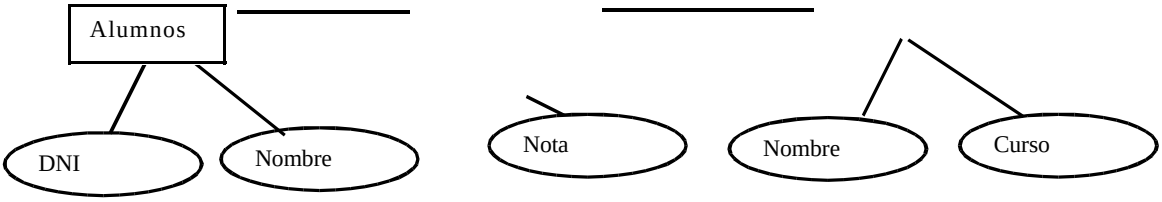
La técnica que se expone a continuación se centra únicamente en los datos, no en las acciones que se realizan en ellos, representando una 'red de datos', esta modelización es imprescindible para las aplicaciones en las que los datos y las relaciones que gobiernan los datos son complejos, y útil en cualquier caso. Esta modelización se usa ampliamente en aplicaciones de bases de datos (en consecuencia en el 99'99% de las aplicaciones de gestión), proporcionando una amplia visión de los datos y las relaciones que gobiernan los datos.

La notación principal de la modelización de los datos es el diagrama entidad relación (E-R), estos diagramas están compuestos de objetos de datos, atributos, relaciones e indicadores de tipo, y su principal propósito es representar los objetos de datos y sus relaciones.

4.3.3.1Diagramas entidad relación

La estructura lógica global de una base de datos puede representarse gráficamente por medio de un Diagrama Entidad-Relación en cual consta de los siguientes elementos:

- Rectángulos: Representan un conjunto de entidades (tablas)
- Elipses: Representan atributos (campos de las tablas)
- Rombos: Representan conjuntos de relaciones (tablas que resultarán de operar con otras tablas [vistas]).
- Líneas: Enlazan atributos a conjuntos de entidades, y conjuntos de entidades a conjuntos de relaciones.



Y las tablas serían:

ALUMNOS(Dni, Nombre)

ASIGNATURAS(Nombre, Curso)

ALUM_ASIG(Dni, NomAsig, nota)

En subrayado la clave primaria de cada tabla.

4.3.3.2 Formas normales

Almacenar los datos no es tan sencillo como parece, y cuando trabajamos con decenas de miles de datos debemos protegernos de inconsistencias entre los datos, redundancia de algún dato y debemos optimizar las busquedas de los datos. Esto se consigue mediante un diseño E-R basado en tercera forma normal (3NF), que consta del siguiente análisis.

Primera forma normal: Una relación se encuentra en primera forma normal (1NF) si y sólo si los valores en los dominios son atómicos, que en cristiano quiere decir que dentro de cada fila-columna de una relación siempre hay un y sólo un valor, nunca un conjunto de múltiples valores.

Por ejemplo en una tabla que almacenemos clientes con el DNI, nombre, primerapellido y segundo apellido, es incorrecto y poco eficiente hacer un campo que contenga el primero y el segundo apellido.

Segunda forma normal: Una relación se encuentra en segunda forma normal (2FN) si y sólo sí está en primera forma normal y los atributos no clave dependen por completo de la clave primaria. La clave primaria es aquella que identifica inequívocamente un datos (en el ejemplo anterior el DNI). Es decir, que todas las filas se deben de poder recuperar de forma única por la clave primaria (que evidentemente será única)

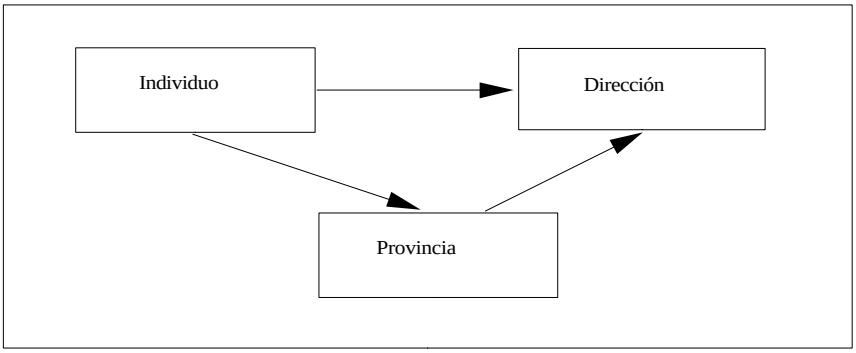
Como se ve estas dos formas normales son de cajón.

Tercera forma normal: Una relación está en tercera forma normal (3FN) si está en segunda forma normal y todos sus atributos no clave dependen de forma no transitiva de la clave primaria. Esto es un poco más difícil de ver, pero con un ejemplo no supone problema.

Para la representación del nombre y dirección del individuo podíamos tener la tentación de definirla tabla como sigue:

Tabla Individuo(Dni, nombre, apellido, Sapellido, Codigo_Postal, Provincia)

Bien si estudiamos gráficamente esta definición y prestamos un poco de antención vemos que tiene la siguiente forma:



Es decir, el código postal identifica inequívocamente la provincia de residencia y no es algo que dependa del idenficador del individuo. Esto implica que una definición así supone un aumento indecesario del tamaño de la base de datos y una fuente de inconsistencias de la misma. La solución en estos casos es siempre la descomposición de la tabla que sufre el problema en varias tablas, tantas como haga falta para solucionarlo.

En este caso sólo es necesario una descomposición de la tabla individuos en dos tablas:

Tabla Individuo(Dni, nombre, apellido, Sapellido, Codigo_Postal)

Tabla Provincia (Codigo_postal, Provincia)

Además el código postal lo consideraremos en la tabla Individuo como una clave ajena de la tabla Provincias, de forma que aseguraremos que los valores introducidos para los individuos existan realmente como provincias.

Para más información os remito a cualquier libro de diseño de bases de datos, ya que este tema se escapa del ámbito del documento, aquí

nos quedamos con que para presentar un diseño necesitamos que las tablas estén en tercera forma normal y la forma de representación.

4.4 Análisis orientado a los objetos

El análisis de los flujos de datos es inútil cuando lo que pretendemos es abordar la implementación del sistema a partir de un lenguaje que maneje objetos (C++, Pyton ... etc) y deseemos explotar estas características, debemos entonces de orientar el análisis hacia los objetos.

4.4.1 Identificación de objetos

Para identificar los objetos se estudia la narrativa de procesamiento del sistema a construir (descripción del programa), seleccionando los nombres o claúsulas nominales. Serán los posibles objetos, que pueden ser:

- *Entidades externas*: Producen o consumen información, pueden ser otros sistemas, dispositivos o personas.
- Cosas:Son parte del dominio de información del problema (informes, señales)
- Ocurrencias: Ocurren en el contexto de operación del sistema (transferencia de propiedades, terminación de una acción..).
- Paleles que juegan personas que interactúan con el sistema.
- Unidades organizativas relevantes para la aplicación (grupos, equipos).
- Lugares: Establecen el contexto del problema y del funcionamiento general del sistema.
- Estructura: Clases de objetos (sensores, computadoras).

También es importante determinar lo que no son objetos con el fin de evitar errores en el análisis del sistema.

En este momento tenemos una serie de objetos potenciales, se deben de estudiar cada uno de ellos para poder tenerlos como objetos reales. Se pueden ver seis características selectivas para incluir o no un objeto potencial en el modelo de análisis (Coad y Yourdon).

1. *Información retenida*: La información sobre el objeto debe ser recordada para que el sistema pueda funcionar.
2. *Servicios necesarios*: El objeto debe de tener un conjunto de operaciones identificables que puedan cambiar de alguna forma el valor de sus atributos.
3. *Múltiples atributos*: En el análisis, los objetos con un sólo atributo pueden presentar problemas de representación (se debe generalizar)
4. *Atributos comunes*: Se pueden definir un conjunto de atributos para los objetos.
5. *Operaciones* comunes: Se pueden definir un conjunto de operaciones para los objetos.
6. *Registros esenciales*: Las entidades externas que consumen o producen información casi siempre se definen como objetos.

4.4.2 Especificación de atributos

Los atributos de un objeto son evidentemente dependientes de la aplicación, así un objeto persona tiene atributos si hacemos una aplicación fiscal, si la aplicación es del padrón tendrá unos atributos comunes con la otra aplicación pero tendrá otros totalmente distintos. Por ello es importante revisar la narrativa de la aplicación y contestar a la siguiente pregunta: ¿Qué elementos de datos definen completamente al objeto en el contexto del problema actual?.

4.4.3 Definición de las operaciones

Una operación cambia un objeto en alguna forma, por ello las operaciones deben ser implementadas de forma que puedan manipular las estructuras de datos que se hayan derivado de los atributos.

Las operaciones se pueden dividir en tres grandes categorías, las que manipulan los datos de alguna forma, las que realizan algún cálculo y las que monitorizan un objeto frente a la ocurrencia de algún suceso de control.

Para obtener el conjunto de operaciones de los objetos se debe de estudiar la narrativa del problema y seleccionar las operaciones que se correspondan, para ello sirve de guía realizar un estudio sobre los verbos, ya que indican las operaciones legítimas y permite una conexión directa con un objeto específico. Además del análisis de los verbos, otra forma de comprender mejor las operaciones es considerando la comunicación entre objetos.

4.4.4 Comunicación

La definición de los objetos sirve de base para el diseño, sin embargo se debe de añadir algo más para que se pueda construir el sistema, se debe de establecer un mecanismo para la comunicación entre los objetos. Este mecanismo se denomina mensaje, tiene la forma siguiente:

***mensaje** : (destino, operación, argumentos)*

Donde destino define el objeto que recibe el mensaje, operación se refiere a la operación que va a recibir el mensaje y argumentos proporciona información para llevar a cabo la operación Por ejemplo el objeto A puede enviar un mensaje el objeto C de la forma

mensaje : (C, op_8, <datos>)

Entonces C busca 'op_8', la realiza con los datos y le devuelve el control a A.

4.4.5 Especificación de objetos

Finalmente se reune toda la información sobre los objetos en un documento de *Especificación de objetos* cuyo formato será el siguiente:

I. Nombre del objeto
II. Descripción de atributos
    A. Nombre del atributo
    B. Contenido del atributo
    C: Tipo/estructura de datos del atributo
III. Entrada externa al objeto
IV. Salida externa del objeto
V. Operaciones
    A. Nombre de la operación
    B. Descripción de la interfaz de la operación
    C. Descripción del procesamiento de la operación
    D. Aspectos de rendimiento
    E. Restricciones y limitaciones.
VI. Conexiones de mensaje

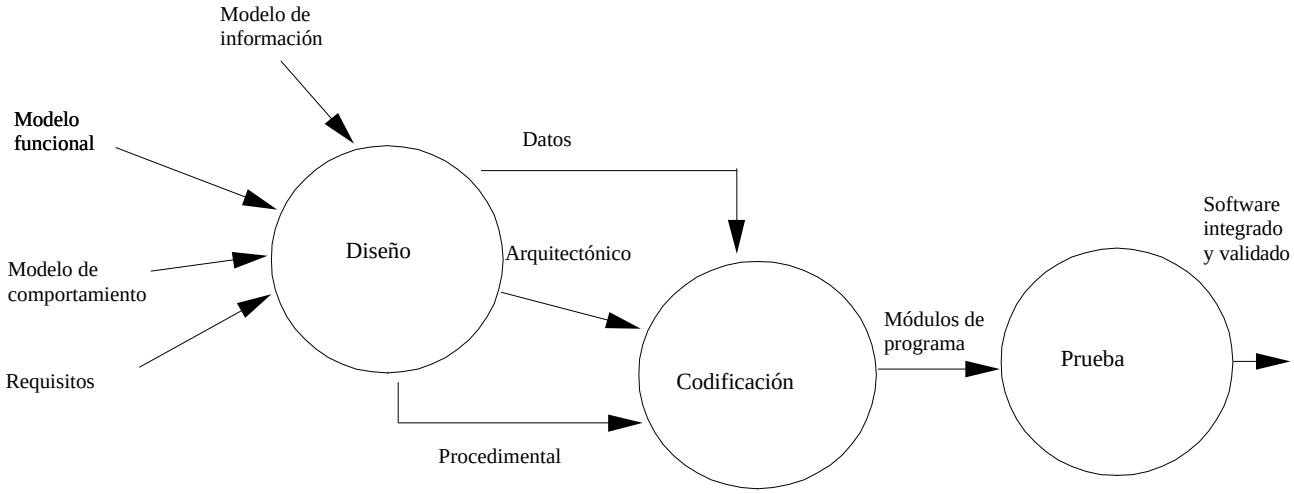
. DISE O DEL SOFTWARE

5.1¿Dónde estamos?

Tras analizar los requisitos del sistema, y habiendo implementado los modelos de información, funcional y de comportamiento estamos preparados para abordar el diseño.

La etapa de diseño, previa a la codificación (por fin código...) está dividida en tres tipos, diseño de datos, arquitectónico y procedimental.

Aunque parezca lo contrario, por todo el rollo que llevamos expuesto, la fase de diseño, codificación y prueba absorben el 75% del desarrollo del software (excluyendo el mantenimiento). La importancia del diseño (para frenar las ganas de crear código rápidamente) se resumen en una palabra: calidad.



El diseño del software es el proceso que permite traducir los requisitos analizados de un sistema en una representación del software, que incicialmente da una visión del mismo y tras posteriores refinamientos nos conducirá a una representación de diseño muy cercana al código fuente. Actualmente además del diseño de datos, arquitectónico y procedimental se debe de prestar especial atención al diseño de la interfaz.

5.2 Fundamentos de diseño

'*El comienzo de la sabiduría de un programador está en reconocer la diferencia entre obtener un programa que funcione y uno que funcione correctamente*' (M.A. Jackson megagurú de programación). Ante esta afirmación lo más que se puede decir, quizás, es que los conceptos fundamentales del diseño de software nos darán la base necesaria para que 'funcione correctamente'.

5.2.1 Abstracción

Mediante la abstracción nos podemos concentrar en un problema con indepencia de lo que haya en niveles más bajos, en los niveles superiores de abstracción utilizaremos el lenguaje del problema para establecer la solución en términos amplios; en niveles inferiores toma una orientación más procedimental.

> *En palabras llanas, y fuera del ámbito de la programación y la informática, si estamos buscando una solución para colocar nuestra televisión en el comedor de casa, en un primer lugar nos planteamos el lugar físico, obviamos el mueble y la forma de construirlo. Cuando tenemos elegido el lugar, procedemos a diseñar nuestro mueble (nos da lo mismo como lo vamos a hacer y en principio los materiales que usaremos), estamos en un nivel inferior de abstracción.*
>
> *Finalmente entramos en el nivel de abstracción procedimental, elegimos los materiales y la forma de construcción (sólo cómo encajaremos las piezas, el diseño ya está hecho). Si lo miramos sin complejos es una forma natural de proceder, se trata simplemente de no empezar la casa por el tejado, algo que a los programadores nos gusta mucho, lanzarnos en plancha a realizar código del programa al alcanzar un mínimo de diseño del sistema.*

Dejando de lado el martillo y el taladro, volviendo al mundo del software, conforme nos movemos por diferentes niveles de abstracción, crearemos abstracciones de datos y de procedimientos. Una abstracción de datos es una colección de información que describe un objeto, un ejemplo sería un documento de empadronamiento, en realidad está formado por varios trozos de información, pero nos podemos referir a todos mediante su nombre. Una abstracción procedimental es una determinada secuencia de instrucciones que tienen una función limitada y

específica. Una tercera forma de abstraccion sería la abstracción de control, que implica un mecanismo de control del programa sin especificar los detalles internos.

En la abstracción, definimos a grandes rasgos lo que hace el programa, los datos que utiliza y como los controlamos, aquí aparecen los primeros términos orientados al software como 'repetir hasta'.

5.2.2 Refinamiento

El refinamiento sucesivo nos permite, como estrategia de diseño, partir de una declaración generalista de una función hasta llegar a las sentencias del lenguaje (Nirklaus Wirth). En cada paso del refinamiento, una o varias instrucciones del programa se descomponen en instrucciones más detalladas, esta descomposición termina cuando todas las instrucciones están expresadas en términos de la computadora usada.

Por fin llegamos al código, ya que el refinamiento es realmente un proceso de elaboración, comenzamos con la declaración de una función, que hemos definido en el nivel superior de abstracción (que no proporciona información sobre su funcionamiento y/o estructura), el refinamiento permite ampliar la declaración dando cada vez más detalles.

5.2.3 Modularidad

El concepto de modularidad para el software se tiene en cuenta practicamente desde los años 50, el software se divide en componentes con nombres y ubicaciones determinados, que se denominan módulos y que se integran para satisfacer los requisitos del problema.

El software monolítico, un programa compuesto por un único módulo, es inabarcable y prácticamente irrealizable dado que el número de caminos de control, expansión de las referencias, número de variables y complejidad global, pueden hacer imposible su correcta comprensión.

La solución, como muchos problemas en el mundo de la programación, se encuentra por el axioma de 'divide y vencerás', es más fácil resolver un problema complejo cuando se divide en trozos manejables.

El esfuerzo de desarrollo de un módulo individual disminuye conforme aumenta el número total de módulos; sin embargo, al crecer el número de módulos, el esfuerzo de realizar las interfaces de conexión entre los módulos también crece. Esto quiere decir que debemos evitar tanto una excesiva modularización como una muy pobre.

5.3.4 Diseño modular efectivo

Un diseño modular reduce la complejidad, facilita los cambios (aspecto crítico en la facilidad del mantenimiento) y produce como resultado una implementación más sencilla, permitiendo el desarrollo paralelo de las diferentes partes de un sistema; vamos que es ideal para el mundo del código abierto donde trabajan varias personas en un proyecto.

Un principio que ayuda en el proceso de descomposición modular es el de *ocultamiento de la información*, el cual sugiere que los módulos se han de 'caracterizar por decisiones de diseño que los oculten unos a otros'; es decir, los módulos deben especificarse y diseñarse de forma que la información (procedimientos y datos) contenida dentro de un módulo sea inaccesible a otros módulos que no necesitan tal información. El beneficio de la ocultación se nota sobre todo a la hora de hacer modificaciones durante las pruebas o el mantenimiento del software; cómo la mayoría de datos y procedimientos estarán ocultos a otras partes del software, será menos probable que los errores introducidos (evidentemente sin querer) durante la modificación se propagen a otros lugares del software.

Un diseño modular efectivo debe asegurar la indepencia funcional de los mismos, la indepencia funcional nace directamente de la modularidad, de la abstracción y ocultamiento de la información, esto se puede afirmar que se adquiere desarrollando módulos con una clara función y una adversión a la excesiva interacción con otros módulos, se trata pues de diseñar software de forma que cada módulo se centre en una subfunción específica de los requisitos y tenga una interfaz sencilla, cuando se ve desde otras partes de la estructura software.

El software con indepencia funcional es fácil de desarrollar porque su función puede ser partida y se simplifican las interfaces (con las implicaciones que conlleva cuando el desarrollo es realizado por un equipo, como es nuestro caso). Los módulos independientes son fáciles de mantener y de probar ya que limitan los efectos secundarios, reduce la propagación de errores y fomenta la reutilización de código. Prácticamente la indepencia funcional se mide con dos parámetros: la cohesión y el acoplamiento.

Cohesión

Se puede definir como una medida de la fortaleza funcional relativa de un módulo; es una extensión del concepto de ocultamiento de información. Un módulo sólo debe hacer (idealmente) una cosa, siendo deseable una gran cohesión, lo normal es situarse en la parte media del espectro, básicamente se trata de evitar que los módulos sean una agrupación de líneas de código y debe de tender a que los elementos de procedimientos del módulo se concentren en el mismo momento, sobre el área de una estructura de datos y una función.

Como criterios para establecer el grado de cohesión tenemos (Stevens):

Escribir una frase que describa el propósito del módulo y examinarlo; si la frase no contiene un objeto específico sencillo a continuación del verbo lo más normal es que estemos en la banda baja de cohesión(p. Ejemplo editar todos los datos).

Acoplamiento

Se puede definir como una medida de la interdependencia relativa entre módulos, depende de la complejidad de las interfaces entre los módulos, del punto en el que se hace una entrada o referencia a un módulo y de los datos que pasan a través del interfaz. Nuestro objetivo es una interconectividad sencilla, es decir, acoplamiento bajo, más fácil de comprender y menos propenso a un efecto avalancha de los errores a lo largo del sistema.

Lo normal es conseguir un acoplamiento de control, es decir, un módulo le pasa un 'indicador' sobre el que se toman decisiones en un módulo subordinado. Se debe de evitar que se modifiquen los datos de un módulo en otro, es decir, limitar al máximo el uso de variables globales.

5.2.5 Jerarquía de control.

La jerarquía de control, también denominada estructura del programa, representa la organización de las componentes del programa. Una notación común es un diagrama en forma de árbol, donde cada hoja o nodo es un módulo y representa gráficamente dos características importantes la *visibilidad* y la *conectividad*.

La visibilidad indica el conjunto de componentes del programa que pueden ser invocados o utilizados (sus datos) por un componente dado. La conectividad indica el conjunto de componentes a los que directamente se invoca o se utilizan sus datos en un determinado módulo (por ejemplo un módulo que puede provocar la ejecución de otro módulo.

5.3Diseño de datos

El impacto de los datos sobre la estructura del programa y la complejidad procedimental, hace que el diseño de datos tenga una gran influencia en la calidad del software. Los conceptos de ocultamiento de la información y de abstracción de datos conforman la base de los métodos de diseño de datos.

Se consideran los siguientes principios para la especificación de datos:

- Se deben desarrollar y revisar las representaciones del flujo y contenido de datos, identificando los objetos y buscando alternativas.
- Identificar las estructuras de datos y operaciones que se deben realizar sobre ellas
- Establecer y realizar un diccionario de datos[1] para definir el diseño de los datos y del programa.
- El diseño de datos debe ser descendente, desde referencias generales especificándose en detalle.
- La representación de una estructura de datos sólo debe ser conocida por los módulos que hagan un uso directo de los datos contenidos en la estructura (ocultamiento de información y acoplamiento de datos).
- El uso de una biblioteca de plantillas de estructuras de datos (tipos abstractos de datos) pueden reducir el trabajo de especificación y diseño de datos.
- Hay que pensar en que el lenguaje de programación soporte la estructura de datos elegida.

5.4Diseño arquitectónico

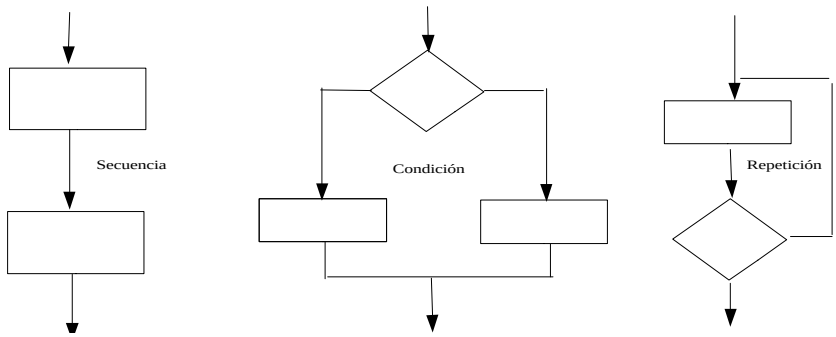
El objetivo del diseño arquitectonico es desarrollar una estructura de programa modular y representar las relaciones de control entre módulos. El diseño arquitectónico mezcla la estructura de programas y la de datos definiendo las interfaces que facilitan el flujo de los datos a lo largo del programa.

5.5Diseño procedimental

Tras establecer la estructura del programa (diseño arquitectónico) y de datos (diseño de datos) se realiza el diseño procedimental que, define los detalles algorítmicos. Esta especificación no puede ser ambigua por ello se realizará mediante una notación de programación estructurada, que tiene tres construcciones básicas: la secuencia, la condición y la repetición.

Cualquier programa con indepencia del área de aplicación y de la complejidad técnica, puede diseñarse e implementarse usando sólo estas tres construcciones a nivel de diseño.

El apoyo en una notación gráfica también es muy importante, siendo el más adecuado el diagrama de flujo, cuyos símbolos son:



1 Un diccionario de datos representa las relaciones entre los datos y las restricciones sobre los elementos de una estructura de d simplifica la definición de algorítmos.

MODELOS DE DOCUMENTACIÓN

6.1 Introducción

Existen numerosas propuestas sobre la organización de la documentación para el diseño software y recoger los requisitos, esta documentación debe ser revisada con frecuencia a lo largo del desarrollo de la aplicación, por lo que es muy conveniente que se redacte de una forma fácil de modificar. Por otro lado también debe facilitar la labor de verificación del cumplimiento de las especificaciones. Esto hace que la mejor manera de redactar este documento sea en forma de un contrato con distintas cláusulas organizadas y agrupadas según el caracter de los requisitos.

En este documento se seguirán dos modelos distintos el de la Agencia Especial Europea, que está orientado al diseño procedimental y el de UML. En cualquier caso ambos son utilizables si estamos interesados en diseñar productos de gestión o para la la Admisnistración pública ya que su trasvase a un modelo de Métrica 3 es directo.

6.2 Documento de Requisitos del Software

Este documento tiene un caracter general y en algunos casos no será necesario cumplimentar todos sus apartados. El índice de este documento es el siguiente:

1. Introducción
   1.1 Objetivo
   1.2 Ambito
   1.3 Definiciones, siglas y abreviaturas
   1.4 Referencias
2. Descripción general
3. Requisitos específicos
   3.1 Requisitos funcionales
   3.2 Requisitos de capacidad
   3.3 Requisitos de interface
   3.4 Requisitos de operación
   3.5 Requisitos de recursos
   3.6 Requisitos de verificación
   3.7 Requisitos de pruebas de aceptación
   3.8 Requisitos de documentación
   3.9 Requisitos de seguridad
   3.10 Requisitos de transportabilidad
   3.11 Requisitos de calidad
   3.12 Requisitos de fiabilidad
   3.13 Requisitos de mantenibilidad
   3.14 Requisitos de salvaguarda
4. Apéndices

6.3 Documento de Diseño del Software

Tambien tiene un caracter general y en algunos casos no será necesario cumplimentar todos sus apartados. El índice de este documento es el siguiente:

1. Introducción
   1.1 Objetivo
   1.2 Ámbito
   1.3 Definiciones, siglas y abreviaturas
2. Panorámica del sistema
3. Contexto del sistema
4. Diseño del sistema
   4.1 Metodología de diseño de alto nivel
   4.2 Descomposición del sistema
5. Descripción de componentes
6. Viabilidad y recursos estimados
7. Matriz Requisitos/Componentes

UNTÁNDOLO TODO E E PLOS .

7.1 Gegaco

Introducción

La empresa 'Gestransa' dedicada a los servicios decide contratar un producto software que le permita realizar una serie de operaciones sobre sus cuentas y presupuestos. Al conocer el proyecto de Hispalinux denominado 'Gestión Libre' decide apostar por este modelo de desarrollo de software para realizar la aplicación en la cual está interesada.

Puesta en contacto con un grupo de desarrollo de código abierto mantiene una reunión para determinar la forma de funcionamiento de la aplicación y los conceptos que cree que debe de cumplir la misma, de esta reunión nace el “Documento de Conceptos del Sistema”.

Documento de Conceptos del Sistema

'Gestransa' como empresa de servicios tiene dos tipos diferenciados de clientes, los constituidos como empresa y aquellos que acuden de forma `particular'. El servicio que presta la empresa es el de conseguir productos compuestos de las aportaciones de distintas empresas que contrata directamente 'Gestransa'. Es pues un producto personalizado a las necesidades del cliente.

El sistema informático que pretende poner en marcha Gestransa consiste en un programa que debe ser capaz de llevar a cabo el presupuesto de abonos de la empresa a 30 días vista, para ello tendrá en cuenta que se conocen perfectamente una serie de gastos, así como su periodicidad.

El sistema tendrá los datos de los clientes de Gestransa así como de los proveedores habituales con una relación de los productos que cada uno de ellos suministra.

Se desea además que se puedan desarrollar estudios gráficos de la repercusión económica de cada uno de los clientes, proveedores y productos con los que trabaja Gestransa. Así mismo se debe de tener controlado en cada momento el estado contable de la empresa, esto es el activo disponible, así mismo dado el mercado variable en el que se mueve el negocio de la empresa, se desea que sea capaz de realizar presupuestos de compra de los productos que se especifíquen a cada uno de los proveedores, para el cálculo de coste de los distintos productos, tendrá en cuenta la fluctuabilidad del mercado.

D R S

Proyecto: Sistema de Gestión de Gastos y Control Económico.

Autores: Gruceca

Fecha : Julio 2002

Documento : Gegaco- SRD -2002

Contenido:

I

Objetivo

El objetivo del sistema es facilitar la gestión de la empresa Gestransa en lo referente a los diversos abonos e ingresos que se le presentan en el siguiente mes vista tanto de abonos como de recaudación, de forma que pueda presupuestar ambos conceptos teniendo siempre en cuenta los activos reales de los que dispone la empresa.

Ambito

El sistema a desarrollar se denominará GEGACO-1.0 y consistirá en un conjunto de funciones que actuarán sobre una base de datos. En particular, deberá facilitar las siguientes:

- Gestión de activos
- Previsión de ingresos y abonos
- Previsión coste de compras

El sistema no ha de soportar sin embargo la gestión económica y documental de la propia empresa, aunque si se figurarán dichos conceptos en la gestión y previsión de abonos.

Tampoco supondrá una herramienta que controle de forma alguna la calidad del servicio que reciben los clientes, ya que para ello Gestransa está en periodo de certificación ISO 9004.

El sistema tampoco debe de soportar la emisión de facturas.

Definiciones, siglas y abreviaturas

Ninguna

Referencias

Ninguna

Panorámica del documento

El resto del documento contiene una descripción del modelo del sistema, en la Sección 2, y la lista de requisitos específicos en la Sección 3.

Para confeccionar el modelo se ha aplicado la metodología de ANALISIS ESTRUCTURADO. Los diagramas de flujo de datos se han incluido en la sección 2, así como el diccionario de datos y el diagrama Entidad-Relación correspondiente al modelo de datos. Las especificaciones de procesos se incluyen en el apartado 3.1 Requisitos Funcionales

D

Relación con otros proyectos

Se incluye dentro del proyecto Gestión Libre de Hispalinux, por ello toda la documentación del sistema deberá ser editada bajo licencia GPLF. El código resultante del presente documento de Requisitos del Software deberá ser liberdado bajo licencia GPL.

Relación con proyectos anteriores y posteriores

Si el sistema llega a ser totalmente operativo en plazo y forma, este proyecto podría ser ampliado con la gestión de nóminas de los empleados de Gestransa así como con la elaboración de documentación de la Seguridad Social (modelos Tc1 y Tc2 de la tesorería de la Seguridad Social) y de los cálculos pertinentes del IVA trimestral y su posterior declaración impositiva. A pesar de que estos apartados están en negociación se tendrá en cuenta esta posible ampliación.

Relación con proyectos anteriores y posteriores

Ninguna

Objetivo y funciones

La empresa Gestransa recibe sus ingresos a partir de los distintos servicios que contrata con sus clientes, el caracter de dichos servicios es

variable y los subcontrata a distintas empresas proveedoras de los productos necesarios. El sistema debe pues tener un control sobre los clientes y productos o servicios contratados, el periodo de pago de los mismos, los proveedores que tiene en un momento dado y los productos que cada uno de ellos suministra, así como el precio que tiene cada uno de ellos. Un determinado producto puede ser proporcionado por uno o varios proveedores, por lo que el sistema debe de tener capacidad para distinguirlos.

El sistema debe de tener control sobre el estado contable de la empresa, conociendo en todo momento el saldo corriente, así como capacidad de realizar presupuestos de ingresos y de los abonos que debe de soportar la empresa en un plazo de 30 días. También debe ser capaz de estimar el precio para la compra de productos basándose en los precios anteriores. El sistema permitirá realizar una serie de estudios estadísticos en forma de gráficos. El objetivo del sistema es pues el control económico sobre la actividad económica de Gestransa y sus clientes, para ello debe de proporcionar las funciones más directamente relacionadas con la gestión de clientes y proveedores. Así mismo proporcionará las funciones necesarias para el desarrollo de los estudios gráficos. Las funciones principales serán:

- Anotar los ingresos y los abonos realizados
- Indicar el estado contable de la empresa
- Gestionar los servicios contratados por los clientes
- Control del precio de los productos
- Gestión de proveedores
- Presupuestos
- Estudios gráficos

Como complemento serán necesarias otras funciones, en concreto:

- Mantener un registro de clientes
- Mantener un registro de proveedores
- Mantener un registro de productos
- Mantener un registro de servicios contratados

Consideraciones de entorno

Ninguna

Relación con otros sistemas

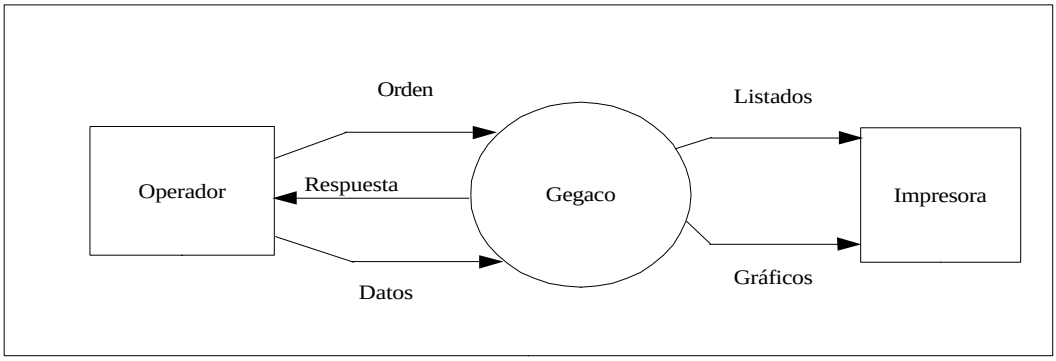
Este sistema se incluye dentro del proyecto Gestión Libre de Hispalinux que en un intento de unificar las herramientas de programación, aboga por el uso de PostgreSQL como SGBD (opcionalmente se puede utilizar otro como MySql).

Restricciones generales

- El sistema se desarrollará y documentará empleando la metodología de análisis y diseño estructurado.
- La implementación se hara sobre el soporte de una base de datos relacional
- El programa se ejecutará en máquinas de capacidad media (PII o superior) y con un sistema operativo con soporte multitarea GNU-LINUX, y en un entorno gráfico Xwindows (GNOME o KDE).

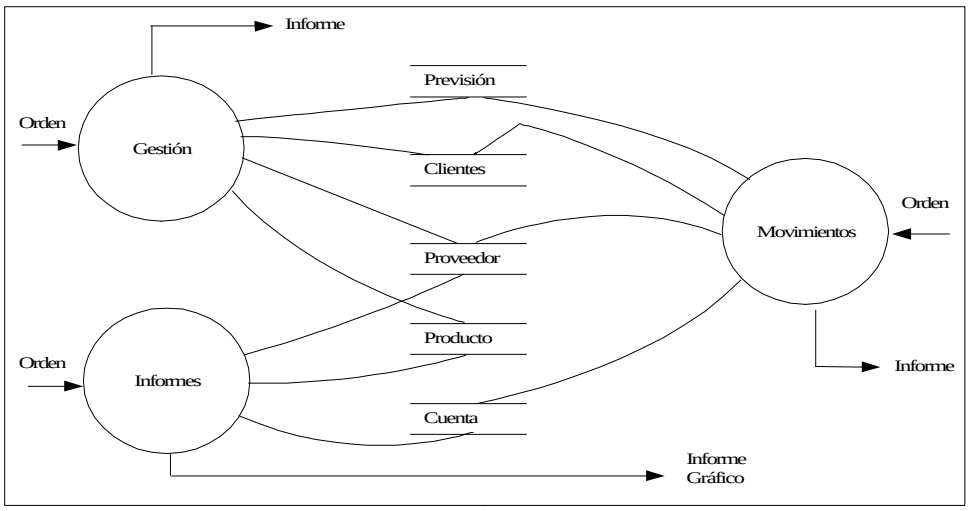
Descripción del modelo.

El sistema de Gestión será utilizado por una única persona, que dispondrá de un terminal con pantalla y teclado. También existirá una impresora por la que podrán obtenerse los listados y fichas de los clientes, proveedores y los gráficos pertinentes. Esta organización se refleja en el Diagrama de Contexto DC.



La operación del sistema se hará mediante un sistema de menús para seleccionar la operación deseada, y con formularios en pantalla para la entrada y presentación de datos. Las funciones a realizar se pueden organizar en varios grupos principales tal y como se indica en el diagrama de flujo de datos DFD.0 de la figura. Estos grupos de funciones se describen a continuación. La descripción precisa de cada

función se incluye en la Sección 3: Requisitos Específicos.

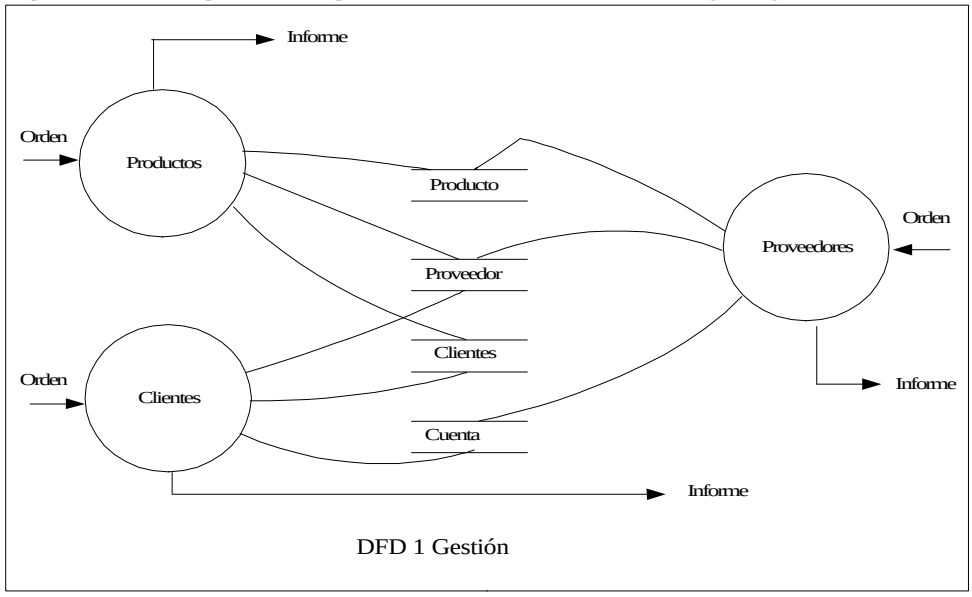


Gestión

Esta función es la encargada de realizar el mantenimiento de los datos de los clientes, proveedores, productos y previsiones del sistema.

Los clientes se identificarán con su NIF o CIF según el caso (Particulares o Empresas respectivamente), nombre y apellidos, dirección. Los proveedores se identificarán con el CIF, dirección y productos que suministran. Los productos se identificarán con un identificador del producto, una descripción del mismo y el precio según proveedor.

Las funciones que realizan estas operaciones se pueden ver en el **DFD1 Gestión** de la figura siguiente:



DFD 1 Gestión

Las funciones de Gestión son las siguientes:

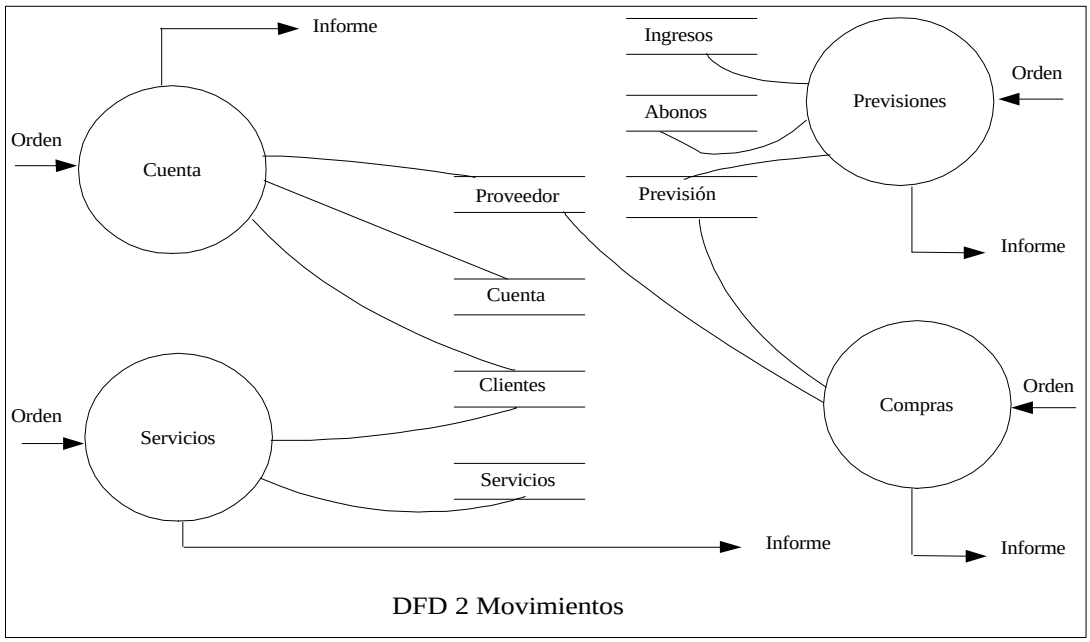
- Productos: Da de alta o de baja los productos.
- Proveedores: Da de alta o de baja proveedores, se debe de tener en cuenta que cuando se da de baja un proveedor también se dará de baja en cascada todos aquellos productos que éste suministre
- Clientes: Alta y baja de Clientes.

Movimientos

La función de movimientos se encarga de actualizar los estados de la cuenta, de los servicios, previsiones y compras que realiza Gestransa.

Las funciones que realizan estas operaciones se pueden ver en el **DFD2 Movimientos** de la figura siguiente:

Las funciones de Movimientos son:



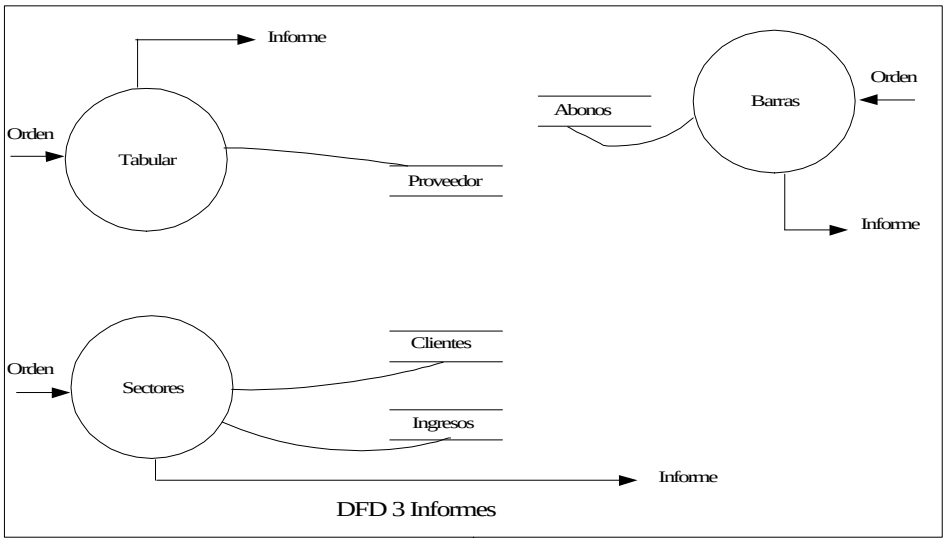
DFD 2 Movimientos

- Cuenta: Actualiza el estado contable de Gestransa de acuerdo a los ingresos y abonos.
- Previsiones: Realiza la previsión de abonos e ingresos en un plazo de 30 días naturales.
- Servicios: Indica los servicios contratados por los clientes.
- Compras: Actualización de las previsiones según los últimos pagos realizados.

Informes

La función Informes se encarga de realizar los informes impresos en tres formatos especificados barras, sectores y tabulares.

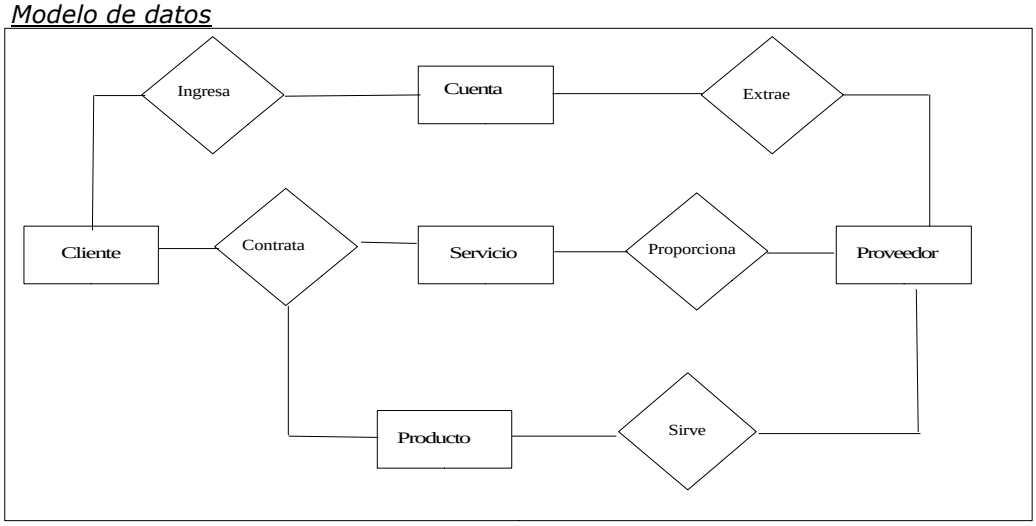
Las funciones de Informes son:



DFD 3 Informes

- Tabular: Realiza un gráfico tabular sobre los proveedores.
- Sectores: Gráfico de sectores de la categoría de ingresos de Gestransa.
- Barras: Gráfico de los abonos mensuales realizados en el último año.

Modelo de datos



Entidades:

Proveedor: Datos de los proveedores de productos a Gestransa.

Clientes : Datos de los clientes.

Cuenta: Datos del estado contable de Gestransa, incluye la fecha de los movimientos.

Servicio: Identificación de los distintos servicios contratados por los clientes.

Producto: Identificación y descripción de los productos

Relaciones:

Ingresa: Movimientos que suponen un incremento del saldo, incluyen fecha, motivo y cuantia.

Extrae: Movimientos que suponen un decremento del saldo, incluyen fecha, motivo y cuantia.

Contrata: Identificación de los servicios contratados por un determinado cliente, incluye los productos, una descripción del servicio y la cantidad.

Proporciona: Identificación de los servicios que puede proporcionar un proveedor

Sirve: Listado de productos que puede proporcionar un proveedor.

Diccionario de Datos:

PROVEEDOR {FICHA-PROVEEDOR}

CLIENTE	{FICHA-CLIENTE}
PRODUCTO	{FICHA-PRODUCTO}
CUENTA	{MOVIMIENTO-CUENTA}
SERVICIO	{FICHA-SERVICIO}
PREVISIÓN	{FICHA-PREVISION}
FICHA-PROVEEDOR	idproveedor + FICHA + DIRECCION
FICHA-CLIENTE	idcliente + FICHA + DIRECCION
FICHA-PRODUCTO	idproducto + descripción + precio
DIRECCION	idcliente | idproveedor + calle + número + localidad + municipio + codigo_postal
FICHA	Nombre + apellidos
FICHA-SERVICIO	idservicio + idcliente + DESCRIPCION
DESCRIPCION	descripcion + cantidad
MOVIMIENTO-CUENTA	cantidad + fecha + motivo + saldo
FICHA-PREVISIÓN	cantidad + periodo + motivo + identificación

Requisitos específicos

Requisitos funcionales.

Almacenamiento de datos.

R.1.1 El sistema mantendrá almacenados en forma permanente los datos indicados en PROVEEDORES, CLIENTES, PRODUCTOS, CUENTA y SERVICIOS, del diccionario de datos de la sección anterior.

Funciones principales

R.1.2 El sistema realizará al menos las siguientes funciones a petición del operador

1. Gestión

Función 1.1 **Alta de productos**: da de alta un nuevo producto

Entrada: PRODUCTO

Salida: FICHA-PRODUCTO

Actualiza: PRODUCTOS

Usa: PROVEEDOR

Efecto: Da de alta un nuevo producto que estará disponible para ser contratado en algún servicio. Todo producto ha de ser proporcionado por un Proveedor

Excepciones: Si existe un producto con ese identificador dará un aviso.

Si se intenta dar de alta un producto sin tener un proveedor asociado se dará un aviso

Función 1.2 **Alta de clientes**: Alta de nuevos clientes

Entrada: CLIENTE

Salida: FICHA-CLIENTE

Actualiza: CLIENTES

Usa:

Efecto: Da de alta un nuevo cliente, es un paso previo a la contratación de servicios

Excepciones: Si existe un cliente con ese NIF, CIF dará un aviso.

Función 1.3 **Alta de proveedores**: Da de alta un nuevo proveedor

Entrada: PROVEEDORES

Salida: FICHA-PROVEEDORES

Actualiza: PROVEEDOR

Usa:

Efecto: Da de alta un nuevo proveedor, paso previo a dar de alta algún producto

Excepciones: Si existe un proveedor con ese CIF dará un aviso.

Función 1.4 **Alta de previsiones**

Entrada: PREVISIÓN
Salida: FICHA-PREVISIÓN
Actualiza: PREVISIÓN
Usa: CLIENTES, PROVEEDORES, SERVICIO
Efecto: Da de alta un abono o ingreso, que tendrá una preriodicidad.
Excepciones:

Función 1.5 **Alta de servicios**:
Entrada: SERVICIO
Salida: FICHA-SERVICIO
Actualiza: SERVICIO
Usa: PRODUCTO-CLIENTES
Efecto: Da de alta un servicio contratado por un cliente
Excepciones

Función 1.6 **Baja de productos**: Da de baja un producto existente de un proveedor determinado
Entrada: FICHA-PRODUCTO
Salida:
Actualiza: PRODUCTO
Usa: PROVEEDORES
Efecto: Lee la identificación entrada por teclado o ratón y da de baja el producto especificado
Excepciones: Si el producto no existe dará un aviso

Función 1.7 **Baja de clientes:**
Entrada: FICHA_CLIENTES
Salida:
Actualiza: CLIENTES
Usa: SERVICIOS
Efecto: Da de baja un cliente
Excepciones: Si el cliente no existe dará un aviso, si tiene contratado un servicio no causará baja.

Función 1.8 **Baja de proveedores:**
Entrada: FICHA-PROVEEDOR
Salida
Actualiza: PROVEEDORES
Usa: PRODUCTOS
Efecto: Da de baja uno de los proveedores de productos, dará de baja los productos que sirva dicho proveedor, marcará los servicios que se vean afectados por dicha baja para su modificación.
Excepciones: Si el proveedor no existe se dará un aviso

Función 1.9 **Baja de previsiones**:
Entrada: FICHA-PREVISION
Salida
Actualiza: PREVISIONES
Usa
Efecto: Da de baja uno de los movimientos de las previsiones
Excepciones

Función 1.10 **Baja de servicios:**
Entrada: FICHA-SERVICIO
Salida:
Actualiza: SERVICIO
Usa:
Efecto: Da de baja un determinado servicio
Excepciones: Si el servicio no existe dará un mesaje de error

Función 1.11 **Modificación de productos**:

Entrada: PRODUCTO
Salida: FICHA-PRODUCTO
Actualiza: PRODUCTO
Usa: PROVEEDOR
Efecto: Modifica alguna de las características de un producto determinado
Excepciones Si el producto no existe dará un mensaje de error

Función 1.12 **Modificación de clientes:**

Entrada: CLIENTE
Salida: FICHA-CLIENTE
Actualiza: CLIENTES
Usa
Efecto: Modifica alguna de los datos del cliente
Excepciones: Si el cliente no existe dará un mensaje de error.

Función 1.13 **Modificación de previsiones:**

Entrada: PREVISION
Salida: FICHA-PREVISION
Actualiza: PREVISION
Usa
Efecto: modificación de los datos de una previsión
Excepciones: La previsión debe existir

Función 1.14 **Modificación de proveedores:**

Entrada: PROVEEDOR
Salida: FICHA-PROVEEDOR
Actualiza: PROVEEDORES
Usa:
Efecto: modificación de los datos de un proveedor
Excepciones: El proveedor deberá existir

Función 1.15 **Modificación de servicios:**

Entrada: SERVICIO
Salida: FICHA-SERVICIO
Actualiza: SERVICIO
Usa: PRODUCTO, CLIENTES
Efecto: modificación de los datos de un servicio contratado por un cliente
Excepciones: El servicio deberá existir

Función 1.16 **Listado productos:**

Entrada:
Salida: Listado de productos
Actualiza:
Usa: PRODUCTOS
Efecto: Imprime un listado con los datos de todos los productos que estén dados de alta
Excepciones

Función 1.17 **Listado productos-proveedor:**

Entrada:
Salida: Listado de productos
Actualiza
Usa: PRODUCTOS, PROVEEDORES
Efecto: Imprime un listado con los datos de los productos que suministra cada proveedor

Excepciones

Función 1.18 **Listado clientes:**

Entrada:
Salida: Listado de clientes
Actualiza
Usa: CLIENTES
Efecto: Imprime un listado de los clientes
Excepciones

Función 1. 19 **Listado de servicios:**

Entrada:
Salida: Listado de servicios por clientes
Actualiza:
Usa: SERVICIOS, CLIENTES
Efecto: Imprime un listado de los distintos servicios contratados por cada uno de los clientes
Excepciones

Función 1.20 **Listado de compras**

Entrada:
Salida: Listado de compra
Actualiza:
Usa: PRODUCTOS, PREVISION
Efecto:Realiza el cálculo de la compra de una serie de productos según el cálculo acumulado.
Excepciones

2. Movimientos

Función 2.1 **Ingreso:**

Entrada: MOVIMIENTO-CUENTA
Salida:
Actualiza: CUENTA
Usa: CLIENTE
Efecto: Refleja el ingreso realizado por un cliente en cuenta
Excepciones: El cliente debe existir y tener un servicio contratado

Función 2.2 **Abono:**

Entrada:MOVIMIENTO-CUENTA
Salida
Actualiza: CUENTA
Usa: PROVEEDOR
Efecto: refleja un abono realizado a un proveedor
Excepciones: El proveedor debe existir, y suministrar productos

Función 2.3 **Ultimos movimientos:**

Entrada:
Salida: Listado con los movimientos de cuenta
Actualiza
Usa: CUENTA
Efecto: Imprime los *n* últimos movimientos indicados
Excepciones

Función 2.4 **Ultimos días:**

Entrada:
Salida: Listado con los movimientos de cuenta

Actualiza

Usa: CUENTA

Efecto: Listado con los movimientos de los *n* días indicados

Excepciones

Función 2.5 **Previsión:**

Entrada:

Salida

Actualiza

Usa: PREVISION

Efecto: Listado con la previsión para los próximos 30 días

Excepciones

Función 2.6 **Compras:**

Entrada:

Salida

Actualiza

Usa: PROVEEDOR, PRODUCTO

Efecto: Listado del precio de una compra de una serie de productos a un determinado proveedor

Excepciones: Si el proveedor no existe, o bien si existe y no suministra de producto selecionado se indicará con un mensaje de error, diferenciado para cada uno de los casos.

3. Informes

Función 3.1 **Informe tabular:**

Entrada: PROVEEDOR

Salida: Informe tabular

Actualiza

Usa

Efecto: Genera en pantalla un informe tabular de los proveedores

Excepciones

Función 3.2 **Sectores:**

Entrada: SERVICIO

Salida: Gráfico de Sectores

Actualiza

Usa: CUENTA

Efecto: Gráfico de sectores por pantalla de los ingresos diferenciados por clientes

Excepciones

Función 3.3 **Barras:**

Entrada: PREVISION, CUENTA

Salida: Gráfico de barras

Actualiza

Usa: CUENTA, PREVISION

Efecto: Gráfico de barras por pantalla de los gastos reales con respecto a los presupuestados

Excepciones

Función 3.4 **Impresion:**

Entrada:

Salida: Impresión de gráfico

Actualiza

Usa: Gráfico

Efecto: Impresión del último gráfico generado

Excepciones: Sólo se imprime el último gráfico generado.

Requisitos de capacidad

R.2.1 El software deberá soportar por lo menos 10.000 clientes, 10.000 proveedores, 1.000.000 de productos y 15.000 movimientos en cuenta

R.2.2 La ejecución de las funciones principales a excepción de los listados y la generación de gráficas deberá durar como máximo 2 segundos, en un PC con procesador P-III a 1 Ghz.

R.2.3 El tiempo de los listados dependerá de la impresora que se instale

R.2.4 El tiempo de ejecución de los gráficos será inferior a 30 segundos en una máquina con las caracteristicas indicadas en R.2.2

Requisitos de interfase

No aplicable

Requisitos de operación

R.4.1 La selección de una función se hará mediante un sistema de menús en un entorno de ventanas del tipo Xwindows.

R.4.2 Existirá la posibilidad de realizar y restablecer copias de seguridad.

Requisitos de recursos

R.5.1 El sistema se ejecutará en un PC de gama media, con una configuración mínima de:

- CPU 686 a 1 Ghz.
- Memoria 128 kb
- Pantalla gráfica
- Tarjeta gráfica de 2 Mb
- CDRom x 32
- Disco extraible de 3 ½ " con una capacidad de 1'4 Mb
- Disco fijo: Con capacidad mínima para el S.O., programas y registros permanentes (recomendado un mínimo de 8 Gb).

Requisitos de verificación

R.6.1 (Deseable) Deberá disponer de un sistema de acceso a los registros de clientes, proveedores y productos de forma independiente a la aplicación, y que permita examinar el contenido de los mismos, preferiblemente obteniendo un listado del mismo, para comprobar que la información almacenada es correcta.

R.6.2 Se deberá comprobar el ajuste del sistema de simulación de precios

Requisitos de pruebas de aceptación

R.7.1 Se deberá probar al menos una vez todas y cada una de las funciones tanto con datos correctos como incorrectos.

R.7.2 (Deseable) Función de autoajuste de modificación de precios

Requisitos de documentación

R.8.1 Manual informatizado de operación que describirá el uso del sistema imprimible por temas.

R.8.2 (Deseable) Resumen del manual de operación disponible como ayuda en línea

Requisitos de seguridad

No aplicable

Requisitos de transportabilidad

No aplicable

Requisitos de calidad

No aplicable

Requisitos de fiabilidad

No aplicable

Requisitos de mantenibilidad

No aplicable

Requisitos de salvaguarda

No aplicable

D D S

Proyecto: Sistema de Gestión de Gastos y Control Económico.
Autores: Gruceca
Fecha : Julio 2002
Documento : Gegaco- ADD -2002

_____

Contenido:

1. INTRODUCCION

1.1 Objetivo

El objetivo del sistema es facilitar la gestión de la empresa Gestransa en lo referente a los diversos abonos e ingresos que se le presentan en el siguiente mes vista, de forma que pueda presupuestar ambos conceptos teniendo siempre en cuenta los activos reales de los que dispone.

1.2 Ambito

El sistema a desarrollar se denominará GEGACO-1.0 y consistirá en un conjunto de funciones que actuarán sobre una base de datos. En particular, deberá facilitar las siguientes:

- Gestión de activos
- Previsión de ingresos y abonos
- Previsión coste de compras

El sistema no ha de soportar sin embargo la gestión económica y documental de la propia empresa, aunque si se figurarán dichos conceptos en la gestión y previsión de abonos.

Tampoco supondrá una herramienta que controle de forma alguna la calidad del servicio que reciben los clientes, ya que para ello Gestransa está en periodo de certificación ISO 9004.

El sistema tampoco debe de soportar la emisión de facturas.

1.3 Definiciones, siglas y abreviaturas

Ninguna

1.4 Referencias

Gegaco-SRC-2002: Documento de Requisitos del Software del Sistema de Gestión de Gastros y Control Económico

2. PANORÁ ICA DEL SISTE A

Se recoge aquí un resúmen de la descripción del sistema ya incluida en el documento de especificación de requisitos Gegaco-SRD-2002.

2.1 Objetivo y funciones

La empresa Gestransa recibe sus ingresos a partir de los distintos servicios que contrata con sus clientes, el caracter de dichos servicios es variable y los subcontrata a distintas empresas proveedoras de los productos necesarios. El sistema debe pues tener un control sobre los clientes y productos o servicios contratados, el periodo de pago de los mismos, los proveedores que tiene en un momento dado y los productos que cada uno de ellos suministra, así como el precio que tiene cada uno de ellos.

Un determinado producto puede ser proporcionado por uno o varios proveedores, por lo que el sistema debe de tener capacidad para distinguirlos.

El sistema debe de tener control sobre el estado contable de la empresa, conociendo en todo momento el saldo corriente, así como capacidad de realizar presupuestos de compras y de los gastos que debe de soportar la empresa en un plazo de 30 días.

El sistema permitirá realizar una serie de estudios estadísticos en forma de gráficos.

El objetivo del sistema es pues el control económico sobre la actividad económica de Gestransa y sus clientes, para ello debe de proporcionar las funciones más directamente relacionadas con la gestión de clientes y proveedores. Así mismo proporcionará las funciones necesarias para el desarrollo de los estudios gráficos. Las funciones principales serán:

- Anotar los ingresos y los abonos realizados
- Indicar el estado contable de la empresa
- Gestionar los servicios contratados por los clientes
- Control del precio de los productos
- Gestión de proveedores
- Presupuestos
- Estudios gráficos

Como complemento serán necesarias otras funciones, en concreto:

- Mantener un registro de clientes
- Mantener un registro de proveedores
- Mantener un registro de productos

2.2 Descripción funcional.

El sistema de Gestión Económico está operado por una sóla persona, que dispondrá de un terminal con pantalla y teclado. También existirá una impresora por la que podrán obtenerse listados, fichas y gráficos de los datos especificados en las funciones.

2.2.1 Gestión

Las funciones de gestión son las que proporcionan los datos al sistema para que pueda realizar su función, en ellas se dan de alta clientes, proveedores, etc.

Función 1.1 Alta de productos
Función 1.2 Alta de clientes
Función 1.3 Alta de proveedores
Función 1.4 Alta de previsiones
Función 1.5 Alta de servicios
Función 1.6 Baja de productos
Función 1.7 Baja de clientes
Función 1.8 Baja de proveedores
Función 1.9 Baja de previsiones
Función 1.10 Baja de servicios
Función 1.11 Modificación de productos
Función 1.12 Modificación de clientes
Función 1.13 Modificación de proveedores
Función 1.14 Modificación de previsiones
Función 1.15 Modificación de servicios
Función 1.16 Listado productos
Función 1.17 Listado productos-proveedor
Función 1.18 Listado clientes
Función 1.19 Listado de servicios
Función 1.20 Listado de compras

2.2.2 Movimientos

Las funciones de movimiento son aquellas que actualizan el estado contable de Gestransa y de los datos de previsión.

Función 2.1 Ingresos
Función 2.2 Abonos
Función 2.3 Ultimos movimientos
Función 2.4 Ultimos días
Función 2.5 Previsión
Función 2.6 Compras

2.2.3 Informes

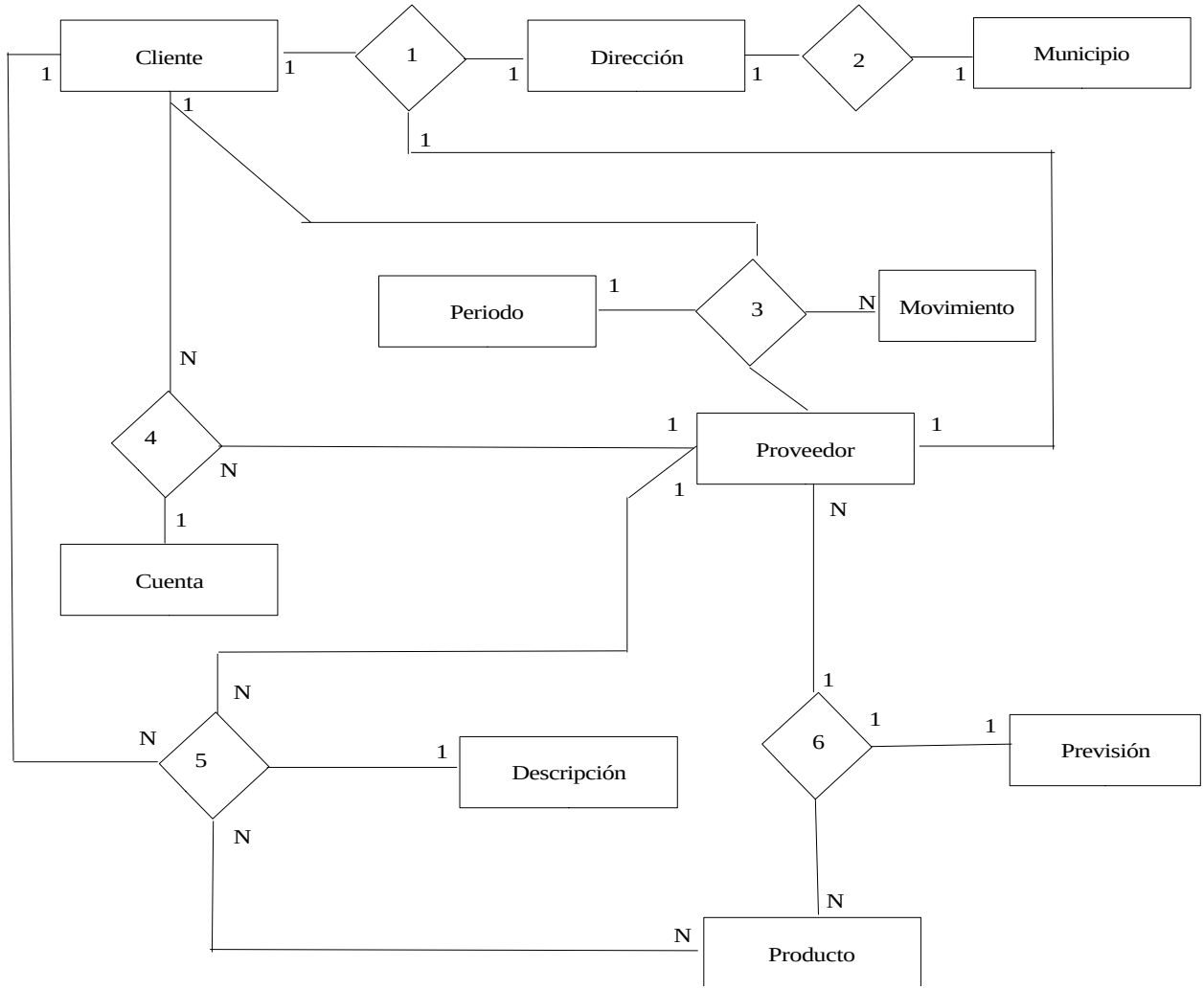
Las funciones de informes son las encargadas de generar los gráficos deseados para estudiar los movimientos.

Función 3.1 Informe tabular
Función 3.2 Sectores
Función 3.3 Barras
Función 3.4 Impresión

2.3Modelo de datos

El modelo conceptual se describe en el diagrama entidad relación, revisado, que aparece en la siguiente figura. Este modelo amplía el descrito en el documento de requisitos GEDACO-SRD-2002.

Las entidades de datos y las relaciones entre ellas son las siguientes:



ENTIDADES:

- Cliente
- Dirección
- Municipio
- Periodo
- Cuenta
- Proveedor
- Previsión
- Descripción
- Producto

RELACIONES

1. Vive_en
2. Pertenece
3. Movimiento-previsto
4. Movimiento
5. Servicio
6. Aproximación

Se debe de tener en cuenta que FICHA-CLIENTE, FICHA-PRODUCTO, FICHA-PROVEEDOR, etc que se hacen referencia en el documento de especificación de requisitos corresponde a una vista externa para ser presentada en pantalla o impresa como ficha.

Los esquemas físicos correspondientes a este modelo conceptual se describen en el apartado 5.1 Módulo: *Base de datos.*

3 CONTE TO DEL SISTE A

No existe conexión con otros sistemas.

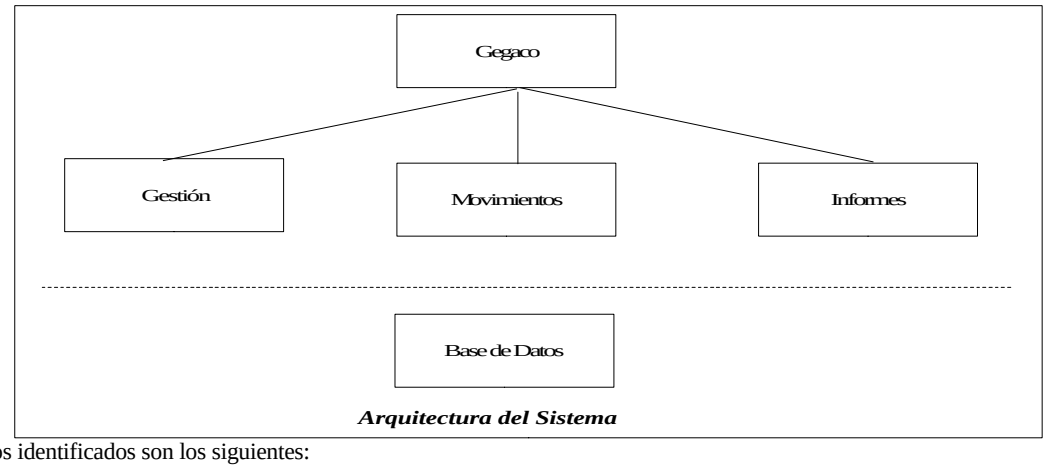
. DISE O DEL SISTE A

4.1 Metodología de diseño de alto nivel

Se utiliza la metodología de Diseño Estructurado, basada en la descomposición funcional del sistema.

4.2 Descomposición del sistema

La estructura modular del sistema aparece representada en la figura siguiente.



Arquitectura del Sistema

Los módulos identificados son los siguientes:

- Gegaco: Es el programa principal, realiza el diálogo con el operador.
- Gestión: Módulo que realiza las funciones de gestión del programa
- Movimientos: Módulo que realiza las funciones de actualización
- Informes: Módulo que se encarga de realizar los informes gráficos
- Base de Datos: Módulo que contiene la base de datos del sistema.

El módulo base de datos está realizado físicamente por el SGBD de código abierto que se utiliza.

. DESCRIPCION DE CO PONENTES

5.1Módulo: Base de Datos.

5.1.1*Tipo*: Base de datos relacional

5.1.2*Objetivo*: Este módulo contiene la base de datos relacional que almacena la información persistente del sistema.

5.1.3*Función*: Almacenar los datos que se indican

5.1.4*Subordinados*: Ninguno

5.1.5*Dependencias*: Gestión, Movimientos, Informes.

5.1.6*Interfaces*: El modelo fisico de datos es el siguiente:

CLIENTES				
Campo	Tipo	Long	Indice	Descripción
Idcliente	Char	9	SI	NIF del cliente
Nombre	Char	15	No	Nombre del cliente
Apellido	Char	15	No	Primer apellido del cliente
Seg_apellido	Char	15	No	Segundo apellido del cliente

PROVEEDOR				
Campo	Tipo	Long	Indice	Descripción
Idproveedor	Char	9	SI	CIF del proveedor
Nombre	Char	50	No	Nombre comercial del proveedor

PRODUCTOS				
Campo	Tipo	Long	Indice	Descripción
Idproducto	Numero	4	SI	Código de referencia del producto
Descripción	Char	50	No	Breve descripción del producto