BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

TESIS

"SISTEMA DE BASE DE DATOS PARA EL NIVEL 0 DE PSP (PERSONAL SOFTWARE PROCESS)"

PARA OBTENER EL GRADO DE:
LICENCIADO EN CIENCIAS DE LA COMPUTACIÓN

PRESENTA
TOMÁS GONZÁLEZ ALVARADO

ASESOR
M.C. PEDRO BELLO LÓPEZ

PUEBLA, PUE. MAYO DE 2009

A mi Madre por su amor, sacrificio y apoyo incondicional
Sin ella no hubiera logrado obtener éste gran triunfo

Con cariño:

el Autor

Índice

Índice de Figuras

Índice de Tablas

INTRODUCCIÓN

La realización de un software, desde la fase de análisis y diseño hasta la implementación, pruebas y entrega del mismo, puede ser un proceso muy largo para aquellos desarrolladores que no tienen la costumbre de planear e incluso pueden tener pérdidas si saben costear el desarrollo de un software. Por lo anterior, es necesario que cada desarrollador tenga varios mecanismos para planear, diseñar, costear y sobre todo cumplir en tiempo y forma con lo planeado y sobre todo con calidad, sin embargo este tipo de disciplinas no se adquieren de la noche a la mañana, es preciso adquirir experiencia de forma inevitable, desarrollando software para diferentes clientes y eso causaría años de tiempo perdido para laborar en una empresa de desarrollo de software.

El proceso de desarrollo de un Software, desde su base, estaría regido por los siguientes principios:

- La calidad de un sistema de software está dada por la calidad del proceso utilizado para desarrollarlo y mantenerlo.
- La calidad de un sistema de software está determinada por la calidad de sus componentes más deficientes.
- La calidad de un componente de software está dada por el individuo que lo desarrolla.
- El desempeño individual está dado por el conocimiento, la disciplina y el compromiso del individuo.
- El desarrollador debe conocer su desempeño personal.
- El desarrollador debe dar seguimiento y analizar su trabajo.
- El desarrollador debe aprender a partir de las variaciones de su desempeño.

En general, el desarrollo de un sistema de software es responsabilidad de cada uno de los desarrolladores, de forma individual, no importa si adquirieron un curso acreditado para trabajo en equipo y comunicación laboral, si el desarrollador no ha adquirido la disciplina necesaria para el desarrollo de su propio módulo, no podrá es parte del sistema de software a desarrollar como grupo.

Por lo anterior, este trabajo de tesis mostrará información de una técnica altamente eficiente para el desarrollo de software denominado PSP (Personal Software Process) “Proceso de Software Personal”, que le ayudará al desarrollador, en especial al Ingeniero en Sistemas a medir costos, estandarizar la codificación, proponer mejoras, estimar el tamaño, planear, generar reportes para documentar el proceso de desarrollo del software, programar revisiones de código y diseño.

OBJETIVO GENERAL

El presente proyecto de tesis pretende documentar, de manera detallada, la metodología de trabajo de PSP (Personal Software Process) únicamente del nivel PSP0, mostrando una introducción de las capacidades que tiene cada etapa. Con esta información, de carácter educativo y no comercial, se busca ayudar a los ingenieros de software e individuos que deseen trabajar con PSP a incrementar su nivel de planeación, diseño, codificación y pruebas para elevar la calidad del producto de software a un menor costo y menor tiempo.

OBJETIVOS PARTICULARES

- Definir la metodología PSP.
- Mostrar la eficacia del uso de la metodología de PSP en el nivel 0.
- Definir las actividades de cada nivel de PSP 0 de manera explícita.

ALCANCES

El alcance de este proyecto pretende, como primera instancia, mostrar a los ingenieros en sistemas, desarrolladores de software, licenciados en sistemas y todo individuo que desee trabajar con la ingeniería de software, la metodología de PSP siendo el instrumento principal a utilizar durante el desarrollo de un sistema de software, proporcionando capacidades inigualables y competitivas en un mundo académico o empresarial.

Implementar la tecnología de PSP en el ámbito académico para los alumnos de carreras afines a Sistemas Computacionales, aumentando su nivel de competitividad en el mundo de las tecnologías del desarrollo de software.

LIMITACIONES

Como toda propuesta para una mejora continua, éste proyecto tiene ciertas limitantes lo cual obliga al individuo a trabajar dentro de un marco de trabajo, las cuales son las siguientes:

- El uso exclusivo de las herramientas de PSP sin oportunidad alguna de integrar actividades o formatos extras.
- La práctica de la metodología de PSP debe ser continua cubriendo todas las actividades en su totalidad
- Se debe respetar el orden de aprendizaje que marca las reglas de PSP.
- Solo existen siete niveles de aprendizaje en la metodología PSP, de las cuales en este documento de tesis, solo se mostrarán los primeros cuatro.

CAPITULO 1. MARCO TEÓRICO

1.1.- INGENIERÍA DE SOFTWARE

La ingeniería de software (SE [Sofware Engineering]) es un enfoque sistemático del desarrollo, operacional, mantenimiento y retiro del software, se considera que la ingeniería de software es la rama de la ingeniería que aplica los principios de la ciencia de la computación y las matemáticas para lograr soluciones costo-efectivas, eficaces en costo, a los problemas de desarrollo de software, es decir, permite elaborar consistentemente productos utilizables y costo-efectivos [10].

El software no sólo implica los programas de computadora y los datos asociados con alguna aplicación o producto, sino que también incluye la documentación necesaria para instalar, utilizar y mantener los programas.

La ingeniería de software surge de la ingeniería de sistemas y de hardware. Abarca un conjunto de 3 elementos clave: métodos, herramientas y procedimientos, estos facilitan al gestor para controlar el proceso de desarrollo de software y suministra a los que practiquen dicha ingeniería las bases para construir software de alta calidad [10].

- ✓ **Métodos de la Ingeniería de Software**. Indican "como" construir técnicamente el software, dentro de las tareas a llevar a cabo se encuentran: la planificación, estimación de proyectos, análisis de requisitos del sistema y de software, diseño de estructuras de datos, arquitectura de programas y procedimientos algorítmicos, codificación, prueba y mantenimiento del software.
- ✓ **Herramientas de la Ingeniería de Software**. Suministran un soporte automático o semiautomático para los métodos. Cuando se integran las herramientas de tal forma que la información creada por una herramienta pueda ser usada por otra, se establece un sistema para el soporte del desarrollo de software llamado Ingeniería de Software asistido por Computadora. Actualmente existen herramientas que asisten cada uno de los métodos mencionados anteriormente, estas son las herramientas CASE.
  CASE (Ingeniería de Software Asistida por Computadora [Compurer Aided Software Engineering]). Combina el software, hardware y base de datos para crear un entorno de la ingeniería de software. Las herramientas son cómo voy a aplicar los métodos para tener un desarrollo.
- ✓ **Procedimientos**. Son el elemento de la ingeniería de software que une a los métodos y a las herramientas y facilita un desarrollo racional, oportuno y completo del software de computadora. Los procedimientos definen la secuencia en la que se aplican los métodos.

Para llevar a cabo el desarrollo de software de calidad es importante contar con los métodos, herramientas y procedimientos útiles que se puedan aplicar en cada uno de los procesos de la ingeniería de software.

1.1.1.- PARADIGMAS

La ingeniería de software está compuesta por una secuencia de pasos que incluyen los métodos, herramientas y procedimientos antes mencionados. Estos pasos se denominan paradigmas de la ingeniería de software. La elección de alguno de estos paradigmas para el desarrollo de software depende de la naturaleza del proyecto y de la aplicación, los métodos, herramientas a usar, los controles y entregas requeridos [16].

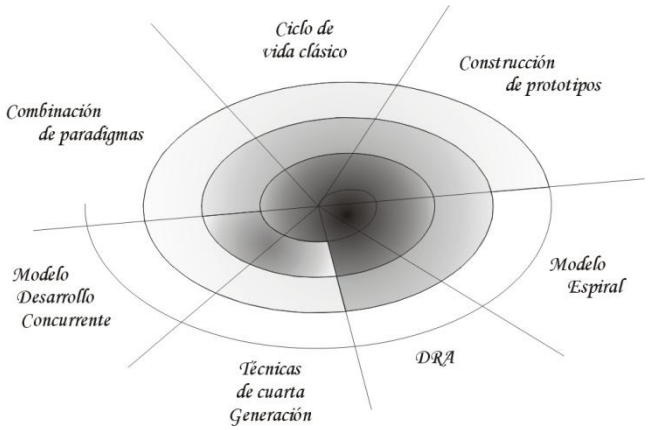


Figura 1.1.- Gama de paradigmas de la Ingeniería de Software

1.1.2.- CICLO DE VIDA CLÁSICO

También llamado modelo en cascada. Este fue el modelo inicial planteado para organizar el proceso de desarrollo, aunque antiguo tiene vigencia en algunos proyectos o como parte de otros modelos, da la medida de los pasos tradicionales de cualquier modelo, diseño, codificación, prueba y mantenimiento [16].

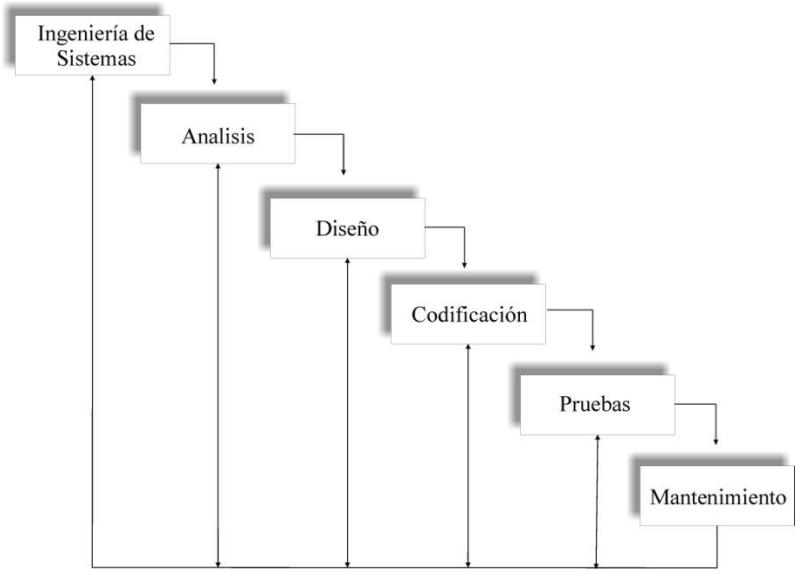


Figura 1.2.- Modelo de Cascada

- ✓ **Ingeniería y Análisis de Sistema**.- Se establecen los requisitos de todos los elementos.
- ✓ **Análisis de requisitos del software**.- El proceso de recopilación de los requisitos se centra y se intensifica para el software. Los requisitos se revisan con el cliente.
- ✓ **Diseño**.- Se enfoca en la estructura de datos, arquitectura, detalles procedimental y características de la interfaz.
- ✓ **Codificación**.- El diseño se traduce para la máquina.
- ✓ **Prueba**.- Una vez teniendo la codificación se realizan pruebas.
- ✓ **Mantenimiento**.- El software puede sufrir cambios.

1.1.3.- EL MODELO DE PROTOTIPOS

La construcción de prototipos es un proceso que facilita al programador la construción de un modelo de software. Se utiliza cuando el programador puede no estar seguro de la eficiencia de un algoritmo, de la adaptabilidad de un sistema operativo o de la forma en que debe realizarse la integración hombre-máquina, esto es debido a que normalmente el cliente define un conjunto de objetivos generales para el software, pero no identifica los requisitos de entrada, proceso o salida [16]. El prototipo puede ser:

- ✓ En papel o modelo que describa la integridad hombre-máquina
- ✓ Prototipo que implemente parte de la función requerida
- ✓ Un programa existente que realice parte de la función deseada.

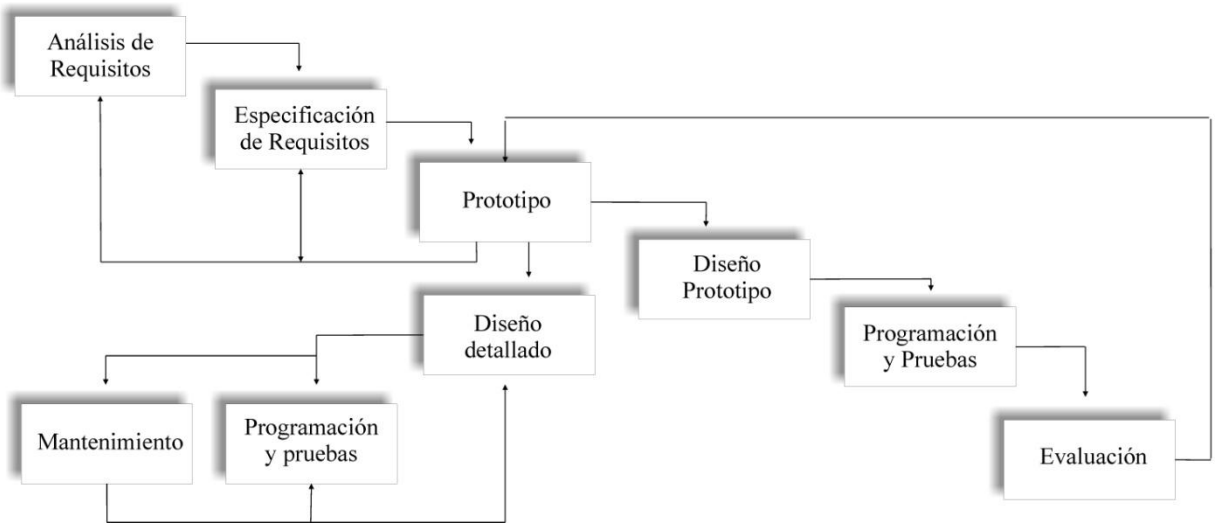


Figura 1.3.- Modelo de Prototipos

1.1.4.- EL MODELO EN ESPIRAL

Este modelo es utilizado para abrir las mejores características de los dos modelos anteriores, añadiendo el análisis de riesgo.

Figura 1.4.- Modelo en Espiral

1. **Planificación**.- Determina objetivos, alternativas y restricciones.
2. **Análisis de riesgos**.- analiza alternativas, identifica y resuelve riesgos.
3. **Ingeniería**.- Desarrollo del producto en el siguiente nivel.
4. **Evaluación del cliente**.- Valoración de los resultados de la ingeniería.

Cuando empieza el proceso evolutivo, el equipo de ingeniería del software gira alrededor de la espiral en la dirección de las agujas del reloj, comenzando por el centro. El primer circuito de la espiral produce el desarrollo de una especificación de productos; los pasos siguientes en la espiral se podrían utilizar para desarrollar un prototipo y progresivamente versiones más sofisticadas del software. Cada paso de la región de planificación produce ajustes en el plan del proyecto. El coste y la planeación se ajustan según la reacción ante la evolución del cliente [16].

1.2.- MODELO ENTIDAD-RELACIÓN

El modelo entidad-relación es el modelo conceptual más utilizado para el diseño conceptual de bases de datos. Fue introducido por Peter Chen en 1976. El modelo entidad-relación está formado por un conjunto de conceptos que permiten describir la realidad mediante un conjunto de representaciones gráficas y lingüísticas [6].

Originalmente, el modelo entidad-relación sólo incluía los conceptos de entidad, relación y atributo. Más tarde, se añadieron otros conceptos, como los atributos compuestos y las jerarquías de generalización, en lo que se ha denominado *modelo entidad-relación extendido*.

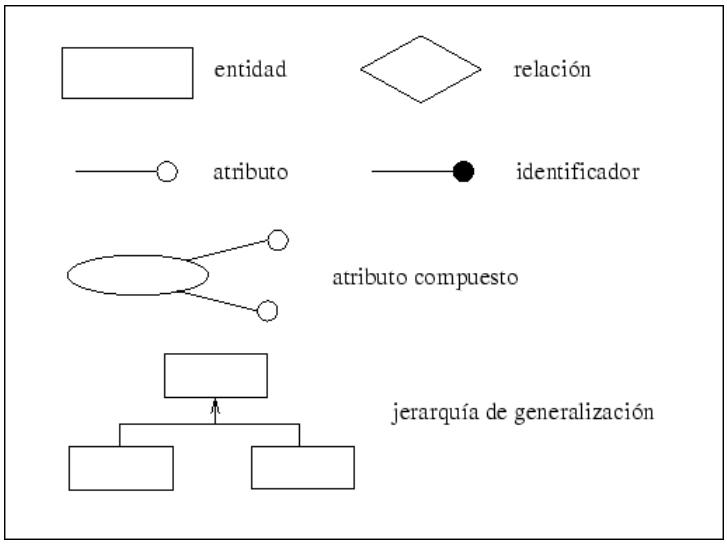


Figura 1.5.- Diagramas del modelo Entidad-Relación

1.2.1.- ENTIDAD

Cualquier tipo de objeto o concepto sobre el que se recoge información: cosa, persona, concepto abstracto o suceso. Por ejemplo: coches, casas, empleados, clientes, empresas, oficios, diseños de productos, conciertos, excursiones, etc. Las entidades se representan gráficamente mediante rectángulos y su nombre aparece en el interior. Un nombre de entidad sólo puede aparecer una vez en el esquema conceptual [6].

Hay dos tipos de entidades: fuertes y débiles. Una *entidad débil* es una entidad cuya existencia depende de la existencia de otra entidad. Una *entidad fuerte* es una entidad que no es débil [10].

1.2.2.- RELACIÓN (INTERRELACIÓN)

Es una correspondencia o asociación entre dos o más entidades. Cada relación tiene un nombre que describe su función. Las relaciones se representan gráficamente mediante rombos y su nombre aparece en el interior [6].

Las entidades que están involucradas en una determinada relación se denominan *entidades participantes*. El número de participantes en una relación es lo que se denomina *grado* de la relación. Por lo tanto, una relación en la que participan dos entidades es una relación *binaria*; si son tres las entidades participantes, la relación es *ternaria*; etc.

Una *relación recursiva* es una relación donde la misma entidad participa más de una vez en la relación con distintos papeles. El nombre de estos papeles es importante para determinar la función de cada participación.

La *cardinalidad* con la que una entidad participa en una relación, especifica el número mínimo y el número máximo de correspondencias en las que puede tomar parte cada

ocurrencia de dicha entidad. La participación de una entidad en una relación es *obligatoria (total),* si la existencia de cada una de sus ocurrencias requiere la existencia de, al menos, una ocurrencia de la otra entidad participante. Si no, la participación es *opcional (parcial)*. Las reglas que definen la cardinalidad de las relaciones son las *reglas de negocio*.

1.2.3.- ATRIBUTO

Es una característica de interés o un hecho sobre una entidad o sobre una relación. Los atributos representan las propiedades básicas de las entidades y de las relaciones. Toda la información extensiva es portada por los atributos. Gráficamente, se representan mediante círculos ligados a las entidades o relaciones a las que pertenecen [6].

Cada atributo tiene un conjunto de valores asociados denominado *dominio*. El dominio define todos los valores posibles que puede tomar un atributo. Puede haber varios atributos definidos sobre un mismo dominio [6].

Los atributos pueden ser simples o compuestos. Un *atributo simple* es un atributo que tiene un solo componente, que no se puede dividir en partes más pequeñas que tengan un significado propio. Un *atributo compuesto* es un atributo con varios componentes, cada uno con un significado por sí mismo. Un grupo de atributos se representa mediante un atributo compuesto cuando tienen afinidad en cuanto a su significado, o en cuanto a su uso. Un atributo compuesto se representa gráficamente mediante un óvalo [6, 13].

1.2.4.- IDENTIFICADOR

Un identificador de una entidad es un atributo o conjunto de atributos que determina de modo único cada ocurrencia de esa entidad [13]. Un identificador de una entidad debe cumplir dos condiciones:

1. No pueden existir dos ocurrencias de la entidad con el mismo valor del identificador.
2. Si se omite cualquier atributo del identificador, la condición anterior deja de cumplirse.

Toda entidad tiene al menos un identificador y puede tener varios identificadores alternativos. Las relaciones no tienen identificadores [13].

1.2.5.- JERARQUÍA DE GENERALIZACIÓN

Una entidad E es una generalización de un grupo de entidades E^1, E^2, E^3, E^n, si cada ocurrencia de cada una de esas entidades es también una ocurrencia de *E*. Todas las propiedades de la entidad genérica *E* son heredadas por las subentidades [6].

Cada jerarquía es total o parcial, y exclusiva o superpuesta. Una jerarquía es total si cada ocurrencia de la entidad genérica corresponde al menos con una ocurrencia de alguna subentidad. Es parcial si existe alguna ocurrencia de la entidad genérica que no corresponde con ninguna ocurrencia de ninguna subentidad. Una jerarquía es exclusiva si cada ocurrencia de la

entidad genérica corresponde, como mucho, con una ocurrencia de una sola de las subentidades. Es superpuesta si existe alguna ocurrencia de la entidad genérica que corresponde a ocurrencias de dos o más subentidades diferentes [6].

1.3.- EL PROCESO DE NORMALIZACIÓN

En la teoría de las bases de datos existe un proceso llamado Normalización. La normalización es el proceso de transformación de representaciones de datos de usuarios en conjuntos estables de estructuras de datos de menor tamaño. Además de ser más sencillas, tales estructuras son más estables [6].

Mediante la normalización se pueden solucionar diversos errores en el diseño de la base de datos así como mejorarlo. También se facilita el trabajo posterior del administrador de la base de datos y de los desarrollos de aplicaciones.

El modelo conceptual de datos obtenido mediante la técnica de entidad-relación será refinado y convertido en un modelo lógico relacional, utilizando la normalización, lo que ofrecerá como resultado el conjunto de tablas a implementar en la base de datos.

Su finalidad es reducir las inconsistencias y redundancias de los datos, facilitar el almacenamiento y evitar las anomalías en las manipulaciones de datos. El objetivo será obtener un modelo lógico normalizado que represente las entidades normalizadas y las interrelacionales existentes entre ellas. Para ello, se toma como punto teórico de partida el concepto de dependencias funcional, que dice: “Un atributo B depende funcionalmente de otro atributo A, de la misma entidad si a cada valor de A le corresponde sólo un valor de B”. Lo anterior se completa mediante la dependencia funcional completa y la dependencia transitiva.

El proceso de normalización consiste en someter a la tablas que representan entidades a un análisis formal para ver si cumplen, o no, las restricciones necesarias que aseguren evitar los problemas citados con anterioridad. A mayor nivel de normalización, mayor calidad en la organización de los datos y menor peligro para la integridad de los mismos [13].

1.3.1.-LOS TRES PASOS PARA LA NORMALIZACIÓN

Al comenzar con la representación de los datos del usuario (por ejemplo lista de campos, posibles entidades básicas y relaciones), el analista normaliza una estructura de datos en tres pasos. Cada paso involucra un procedimiento de simplificación de la estructura de datos [13].

1. *La primera etapa del proceso incluye la eliminación de grupos repetitivos y la identificación de la llave primaria.* Con e fin de hacer esto, la relación necesita desglosarse en dos o más relaciones. En este punto, las relaciones pueden encontrase en la tercera forma normal, pero quizás sean necesarios más pasos para transformar las relaciones a la tercera forma normal.

2. *El segundo paso asegura que todos los atributos no-llave*, *sean completamente dependientes de la llave primaria.* Todas las dependencias normales se eliminan y se colocan en otra relación.
3. *El tercer paso elimina cualquier dependencia transitiva.* Una dependencia transitiva es aquella en la cual sus atributos no-llave son dependientes de otros atributos no-llave.

1.4.- EL LENGUAJE RELACIONAL SQL

SQL se ha convertido en el lenguaje de consulta relacional más popular. El nombre "SQL" es una abreviatura de *Structured Query Language* (Lenguaje de consulta estructurado). En 1974 Donald Chamberlain y otros definieron el lenguaje SEQUEL (*Structured English Query Language*) en IBM Research. Este lenguaje fue implementado inicialmente en un prototipo de IBM llamado SEQUEL-XRM en 1974-75. En 1976-77 se definió una revisión de SEQUEL llamada SEQUEL/2 y el nombre se cambió a SQL. IBM desarrolló un nuevo prototipo llamado System R en 1977. System R implementó un amplio subconjunto de SEQUEL/2 (now SQL) y un número de cambios que se le hicieron a (now SQL) durante el proyecto. System R se instaló en un número de puestos de usuario, tanto internos en IBM como en algunos clientes seleccionados. Gracias al éxito y aceptación de System R en los mismos, IBM inició el desarrollo de productos comerciales que implementaban el lenguaje SQL basado en la tecnología System R.

Durante los años siguientes, IBM y bastantes otros vendedores anunciaron productos SQL tales como SQL/DS (IBM), DB2 (IBM), ORACLE (Oracle Corp.), DG/SQL (Data General Corp.), y SYBASE (Sybase Inc.).

SEQUEL o SQL es una herramienta para organizar, gestionar y recuperar datos almacenados en una base de datos informática. El nombre SQL es una abreviatura de Structured Query Languaje (Lenguaje de Consultas Etructurado). Como su propio nombre lo indica, SQL es un lenguaje informático que se puede utilizar para interaccionar con una base de datos y más concretamente con un tipo especificado llamada base de datos relacional [6].

SQL es también un estándar oficial hoy. En 1982, la American National Standards Institute (ANSI) encargó a su Comité de Bases de Datos X3H2 el desarrollo de una propuesta de lenguaje relacional estándar. Esta propuesta fue ratificada en 1986 y consistía básicamente en el dialecto de IBM de SQL. En 1987, este estándar ANSI fue también aceptado por la Organización Internacional de Estandarización (ISO). Esta versión estándar original de SQL recibió informalmente el nombre de "SQL/86". En 1989, el estándar original fue extendido, y recibió el nuevo nombre, también informal, de "SQL/89". Así mismo, en 1989 se desarrolló un estándar relacionado llamado Database Language Enbeded SQL (ESQL).

1.5.- BASES DE DATOS

La expresión Base de Datos comenzó a popularizarse al principio de 1960. Antes de esa época, en el mundo de la informática se hablaba de archivos y de conjuntos de datos.

Una base de datos es aquella que está constituida por un cierto conjunto de datos persistentes, utilizado por los sistemas de aplicaciones de una empresa determinada. Un Sistema de Base de Datos es un sistema computarizado cuyo propósito general es mantener información y hacer que esté disponible cuando se solicite [6].

Existen cuatro componentes principales que conforman un sistema de Base de Datos: la información, el equipo, los programas y los usuarios.

La información se divide en: Integrada: es la unificación de archivos de datos distintos sin redundancia entre ellos, es decir como un todo, todos los archivos; Compartida: se refiere a la información compartida entre varios usuarios distintos [6].

El equipo lo conforman los dispositivos de almacenamiento como son: dispositivos de entrada y salida, así como los controladores. El procesador o procesadores, memoria principal.

Entre los programas se encuentra el Sistema Administrador de Base de Datos (DBMS), el cual maneja todas las solicitudes de acceso a la Base de Datos formulada por el usuario, las herramientas, utilerías, ayuda y diseño [6].

Entre los usuarios encontramos al Programador de aplicaciones, encargado de escribir los programas de aplicaciones que utilizan las Bases de Datos, el usuario final, el cual interactúa con el sistema desde una terminal en línea, utilizando un procesador del lenguaje de consultas. Y el Administrador de las Bases de Datos (DBA) [6].

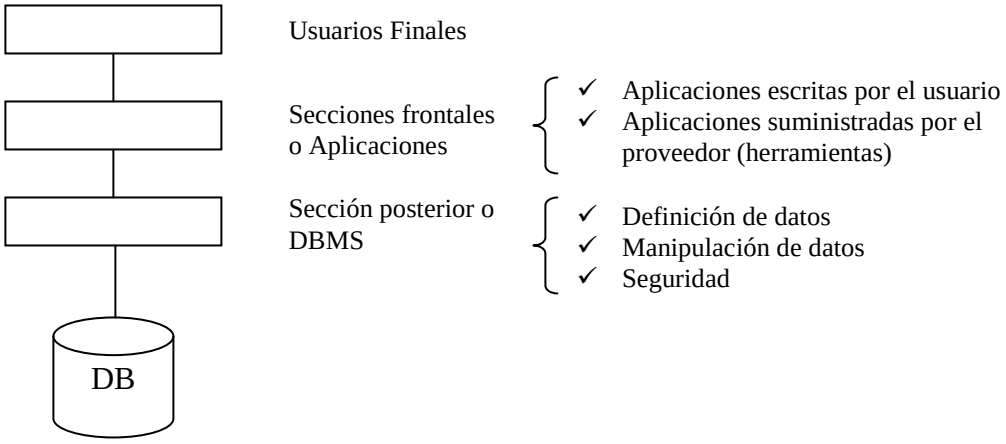


Figura 1.6.- Estructura simple de un Sistema de Bases de Datos

1.5.1.- EL SISTEMA ADMINISTRADOR BASE DE DATOS (DBMS)

El sistema administrador de base de datos (DBMS [Data Base Manager System]) opera de la siguiente manera:

1. Un usuario solicita acceso, empleando algún sub-lenguaje de datos determinado, p/e: SQL.
2. El DBMS interpreta esta solicitud y la analiza.

3. El DBMS inspecciona, en orden, el esquema externo de ese usuario, la correspondencia externa/conceptual, el esquema conceptual, la correspondencia conceptual/interna y la definición de la estructura de almacenamiento.
4. El DBMS ejecuta las operaciones necesarias sobre la Base de Datos almacenada.

1.5.1.1.- FUNCIONES DEL DBMS

- ✓ **Definición de Datos**: El DBMS debe ser capaz de aceptar definiciones de datos en versión fuente y convertirlas en versión objeto apropiado.
- ✓ **Manipulación de Datos**: El DBMS debe ser capaz de atender las solicitudes del usuario para extraer y actualizar datos en la Base de Datos.
- ✓ **Seguridad e Integridad de Datos**: El DBMS debe supervisar las solicitudes de los usuarios y rechazar los intentos de violar las medidas de seguridad e integridad definidas por el Administrador de Bases de Datos (DBA).
- ✓ **Recuperación y Concurrencia de los Datos**: El DBMS debe cuidar del cumplimiento de ciertos controles de recuperación y concurrencia (Administrador de transacciones).
- ✓ **Diccionario de Datos**: Debe contener los diversos esquemas y correspondencias (externas, conceptuales, etc.) en sus versiones fuentes y en sus versiones objetos, incluyendo también referencias cruzadas, es decir, a que dato le corresponde tal dato.
- ✓ **Desempeño**: El DBMS deberá ejecutar todas las funciones de la forma más eficiente posible.

La función general del DBMS constituye la interfaz entre el usuario y el sistema de Bases de Datos.

1.5.2.- ACCESO A BASES DE DATOS

Pasos generales para localizar y presentar información de una Base de Datos [13]:

1. El DBMS decide cual registro almacenado se necesita y pide al manejador de archivos que extraiga ese registro.
2. EL manejador de archivos decide cuál página contiene el registro deseado y pide al manejador de disco que lea esa página (cantidad de datos transferida entre disco y memoria principal, cuyo tamaño puede ser: 1 kb, 2 kb ó 4 kb).
3. El manejador de disco determina la localización física de la página deseada en el disco y realiza la operación de entrada/salida necesaria. Ejemplo:

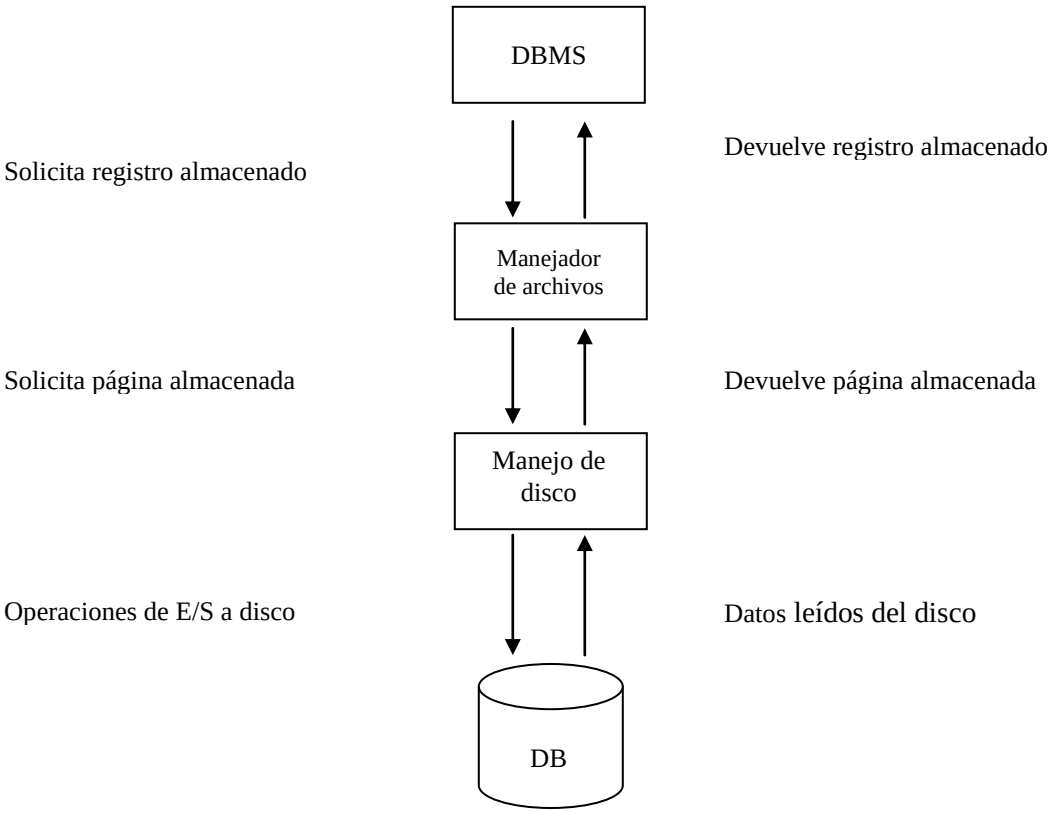


Figura 1.7.- Estructura de un DBMS (Sistema Administrador de Bases de Datos)

1.5.3.- MODELO RELACIONAL

El modelo relacional para la gestión de una base de datos es un modelo de datos basado en la lógica de predicado y en la teoría de conjuntos. Es el modelo más utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente. Tras ser postuladas sus bases en 1970 por Edgar Frank Codd, de los laboratorios IBM en San José (California), no tardó en consolidarse como un nuevo paradigma en los modelos de base de datos [6].

Su idea fundamental es el uso de «relaciones». Estas relaciones podrían considerarse en forma lógica como conjuntos de datos llamados «tuplas». Pese a que ésta es la teoría de las bases de datos relacionales creadas por Edgar Frank Codd, la mayoría de las veces se conceptualiza de una manera más fácil de imaginar, esto es, pensando en cada relación como si fuese una tabla que está compuestas por registros (cada fila de la tabla sería un registro o tupla), y columnas (también llamadas campos) [6].

DESCRIPCIÓN

En este modelo todos los datos son almacenados en relaciones, y como cada relación es un conjunto de datos, el orden en el que estos se almacenen no tiene mayor relevancia (a diferencia de otros modelos como el jerárquico y el de red). Esto tiene la considerable ventaja

de que es más fácil de entender y de utilizar por un usuario no experto. La información puede ser recuperada o almacenada por medio de «consultas» que ofrecen una amplia flexibilidad y poder para administrar la información [6].

Este modelo considera la base de datos como una colección de relaciones. De manera simple, una relación representa una tabla que no es más que un conjunto de filas, cada fila es un conjunto de campos y cada campo representa un valor que interpretado describe el mundo real. Cada fila también se puede denominar tupla o registro y a cada columna también se le puede llamar campo o atributo [6].

Para manipular la información utilizamos un lenguaje relacional, actualmente se cuenta con dos lenguajes formales el Álgebra Relacional y el Cálculo Relacional. El Álgebra relacional permite describir la forma de realizar una consulta, en cambio, el Cálculo relacional sólo indica lo que se desea devolver [6].

El lenguaje más común para construir las consultas a bases de datos relacionales es SQL, Structured Query Language o Lenguaje Estructurado de Consultas, un estándar implementado por los principales motores o sistemas de gestión de bases de datos relacionales [6].

ESQUEMA

Un esquema es la definición de una estructura (generalmente relaciones o tablas de una base de datos), es decir, determina la identidad de la relación y qué tipo de información podrá ser almacenada dentro de ella; en otras palabras, el esquema son los metadatos de la relación [6]. Todo esquema constará de:

- ✓ Nombre de la relación (su identificador).
- ✓ Nombre de los atributos (o campos) de la relación y sus dominios; el dominio de un atributo o campo define los valores permitidos para el mismo, es equivalente al tipo de dato por ejemplo character, integer, date, string, etc.

INSTANCIAS

Una instancia de manera formal es la aplicación de un esquema a un conjunto finito de datos. En palabras no tan técnicas, se puede definir como el contenido de una tabla en un momento dado, pero también es válido referirnos a una instancia cuando trabajamos o mostramos únicamente un subconjunto de la información contenida en una relación o tabla, como por ejemplo:

Ciertos caracteres y números (una sola columna de una sola fila).
Algunas o todas las filas con todas o algunas columnas.
Cada fila es una tupla. El número de filas es llamado cardinalidad.
El número de columnas es llamado paridad o grado.

1.5.4.- BASES DE DATOS RELACIONALES

Una base de datos relacional parte del modelo relacional de base de datos, en el cual los datos son ordenados jerárquicamente, mediante "ordenamiento de burbuja", en el cual se despacha el primero en llegar, si y solo si (sql = 0), por lo que la bandera (Flag) da negativo y se sale del ciclo [13].

1.5.4.1.- CARACTERÍSTICAS

Una base de datos relacional se compone de varias tablas o relaciones.

- ✓ No pueden existir dos tablas con el mismo nombre.
- ✓ Cada tabla es a su vez un conjunto de registros, filas o tuplas.
- ✓ Cada registro representa un objeto del mundo real.
- ✓ Cada una de estos registros consta de varias columnas, campos o atributos.
- ✓ No pueden existir dos columnas con el mismo nombre en una misma tabla.
- ✓ Los valores almacenados en una columna deben ser del mismo tipo de dato.
- ✓ Todas las filas de una misma tabla poseen el mismo número de columnas.
- ✓ No se considera el orden en que se almacenan los registros en las tablas.
- ✓ No se considera el orden en que se almacenan las tablas en la base de datos.
- ✓ La información puede ser recuperada o almacenada por medio de sentencias llamadas «consultas».

1.5.4.2.- ELEMENTOS

- ✓ **Relaciones Base y Derivada:** En una base de datos relacional, todos los datos se almacenan y se acceden a ellos por medio de relaciones. Las relaciones que almacenan datos son llamados "relaciones base" y su implementación es llamada "tabla". Otras relaciones no almacenan datos, pero que son calculadas al aplicar operaciones relacionales. Estas relaciones son llamadas "relaciones derivadas" y su implementación es llamada "vista" o "consulta". Las relaciones derivadas son convenientes ya que expresan información de varias relaciones actuando como si fuera una sola.

- ✓ **Restricciones:** Una restricción es una condición que obliga el cumplimiento de ciertas condiciones en la base de datos. Algunas no son determinadas por los usuarios, sino que son inherentemente definidas por el simple hecho de que la base de datos sea relacional. Algunas otras restricciones las puede definir el usuario, por ejemplo, usar un campo con valores enteros entre 1 y 10.

  Las restricciones proveen un método de implementar reglas en la base de datos. Las restricciones restringen los datos que pueden ser almacenados en las tablas. Usualmente se definen usando expresiones que dan como resultado un valor boleano, indicando si los datos satisfacen la restricción o no.

Las restricciones no son parte formal del modelo relacional, pero son incluidas porque juegan el rol de organizar mejor los datos. Las restricciones son muy discutidas junto con los conceptos relacionales.

- ✓ **Dominios:** Un dominio describe un conjunto de posibles valores para cierto atributo. Como un dominio restringe los valores del atributo, puede ser considerado como una restricción. Matemáticamente, atribuir un dominio a un atributo significa "todos los valores de este atributo deben de ser elementos del conjunto especificado".

  Distintos tipos de dominios son: enteros, cadenas de texto, fecha, etc.

- ✓ **Clave Única:** Cada tabla puede tener uno o más campos cuyos valores identifican de forma única cada registro de dicha tabla, es decir, no pueden existir dos o más registros diferentes cuyos valores en dichos campos sean idénticos. Este conjunto de campos se llama clave única.

  Pueden existir varias claves únicas en una determinada tabla, y a cada una de éstas suele llamársele candidata a clave primaria.

- ✓ **Clave Primaria:** Una clave primaria es una clave única elegida entre todas las candidatas, para especificar los datos que serán relacionados con las demás tablas. La forma de hacer esto es por medio de claves foráneas.

  Sólo puede existir una clave primaria por tabla y ningún campo de dicha clave puede contener valores NULL.

- ✓ **Clave Foránea:** Una clave foránea es una referencia a una clave en otra tabla. Las claves foráneas no necesitan ser claves únicas en la tabla donde están y si a donde están referenciadas.

  Por ejemplo, el código de departamento puede ser una clave foránea en la tabla de empleados, pero obviamente se permite que haya varios empleados en un mismo departamento, pero existirá solo un departamento.

- ✓ **Clave Índice:** Las claves índice surgen con la necesidad de tener un acceso más rápido a los datos. Los índices pueden ser creados con cualquier combinación de campos de una tabla. Las consultas que filtran registros por medio de estos campos, pueden encontrar los registros de forma no secuencial usando la clave índice.

  Las bases de datos relacionales incluyen múltiples técnicas de ordenamiento, cada una de ellas es óptima para cierta distribución de datos y tamaño de la relación.

  Los índices generalmente no se consideran parte de la base de datos, pues son un detalle agregado. Sin embargo, las claves índices son desarrolladas por el mismo grupo de programadores que las otras partes de la base de datos.

1.5.4.3.- PROCEDIMIENTOS ALMACENADOS

Un procedimiento almacenado es código ejecutable que se asocia y se almacena con la base de datos. Los procedimientos almacenados usualmente recogen y personalizan operaciones comunes como insertar un registro dentro de una tabla, recopilar información estadística, o encapsular cálculos complejos. Son frecuentemente usados por un API por seguridad o simplicidad [2].

Los procedimientos almacenados no son parte del modelo relacional, pero todas las implementaciones comerciales los incluyen.

1.5.4.4.- ESTRUCTURA

La base de datos se organiza en dos marcadas secciones; el esquema y los datos (o instancia).
El esquema es la definición de la estructura de la base de datos y principalmente almacena los siguientes datos:

- ✓ El nombre de cada tabla
- ✓ El nombre de cada campo
- ✓ El tipo de dato de cada campo
- ✓ La tabla a la que pertenece cada campo

Las bases de datos relacionales pasan por un proceso al que se le conoce como normalización, el resultado de dicho proceso es un esquema que permite que la base de datos sea usada de manera óptima.

Los datos o instancia es el contenido de la base de datos en un momento dado. Es en sí, el contenido de todos los registros.

1.5.4.5.- MANIPULACIÓN DE LA INFORMACIÓN

Para manipular la información utilizamos un lenguaje relacional, actualmente se cuenta con dos lenguajes formales: el álgebra relacional y el cálculo relacional. El álgebra relacional permite describir la forma de realizar una consulta, en cambio, el cálculo relacional sólo indica lo que se desea devolver.

El lenguaje más común para construir las consultas a bases de datos relacionales es SQL (Structured Query Language), un estándar implementado por los principales motores o sistemas de gestión de bases de datos relacionales.

En el modelo relacional los atributos deben estar explícitamente relacionados a un nombre en todas las operaciones, en cambio, el estándar SQL permite usar columnas sin nombre en conjuntos de resultados, como el asterisco taquigráfico (*) como notación de consultas.

Al contrario del modelo relacional, el estándar SQL requiere que las columnas tengan un orden definido, lo cual es fácil de implementar en una computadora, ya que la memoria es lineal.

Es de notar, sin embargo, que en SQL el orden de las columnas y los registros devueltos en cierto conjunto de resultado nunca está garantizado, a no ser que explícitamente sea especificado por el usuario.

1.5.4.6.- MANEJADORES DE BASES DE DATOS RELACIONALES

Existe software exclusivamente dedicado a tratar con bases de datos relacionales. Este software se conoce como **SGBD** (Sistema de Gestión de Base de Datos Relacional) o **RDBMS** (del inglés *Relational Database Management System*).

Entre los gestores o manejadores más actuales y populares encontramos: MySQL, PostgreSQL, Oracle y Microsoft SQL Server.

1.5.4.7.- VENTAJAS Y DESVENTAJAS

- ✓ **Ventajas**
  - Provee herramientas que garantizan evitar la duplicidad de registros.
  - Garantiza la integridad referencial, así, al eliminar un registro elimina todos los registros relacionados dependientes.
  - Favorece la normalización por ser más comprensible y aplicable.
- ✓ **Desventajas**
  - Presentan deficiencias con datos gráficos, multimedia, CAD y sistemas de información geográfica.
  - No se manipulan de forma manejable los bloques de texto como tipo de dato.
  - Las bases de datos orientadas a objetos (BDOO) se propusieron con el objetivo de satisfacer las necesidades de las aplicaciones anteriores y así, complementar pero no sustituir a las bases de datos relacionales.

1.6.- TECNOLOGÍA .NET

Microsoft .NET es una plataforma de desarrollo y ejecución de aplicaciones. Esto quiere decir que no sólo nos brinda todas las herramientas y servicios que se necesitan para desarrollar modernas aplicaciones empresariales y de misión crítica, sino que también provee de mecanismos robustos, seguros y eficientes para asegurar que la ejecución de las mismas sea óptima [1]. Los componentes principales de la plataforma .NET son:

- ✓ Un entorno de ejecución de aplicaciones, también llamado "Runtime", que es un componente de software cuya función es la de ejecutar las aplicaciones .NET e interactuar con el sistema operativo ofreciendo sus servicios y recursos.
- ✓ Un conjunto de bibliotecas de funcionalidades y controles reutilizables, con una enorme cantidad de componentes ya programados listos para ser consumidos por otras aplicaciones.
- ✓ Un conjunto de lenguajes de programación de alto nivel, junto con sus compiladores y linkers, que permitirán el desarrollo de aplicaciones sobre la plataforma .NET.
- ✓ Un conjunto de utilitarios y herramientas de desarrollo para simplificar las tareas más comunes del proceso de desarrollo de aplicaciones
- ✓ Documentación y guías de arquitectura, que describen las mejores prácticas de diseño, organización, desarrollo, prueba e instalación de aplicaciones .NET

A continuación se presenta una breve descripción de algunas de las características principales de la plataforma Microsoft .NET:

- ✓ Se dice que es una plataforma de ejecución intermedia (Figura 1.8), ya que las aplicaciones .NET no son ejecutadas directamente por el sistema operativo, como ocurre en el modelo tradicional de desarrollo. En su lugar, las aplicaciones .NET están diseñadas para ser ejecutadas contra un componente de software llamado Entorno de Ejecución (muchas veces también conocido como "Runtime" o "Máquina Virtual"). Este componente es el encargado de manejar el ciclo de vida de cualquier aplicación .NET, iniciándola, deteniéndola, interactuando con el Sistema Operativo y proveyéndole servicios y recursos en tiempo de ejecución
- ✓ La plataforma Microsoft .NET está completamente basada en el paradigma de Orientación a Objetos. En mi particular opinión se puede obtener mayor información accesando a la página de Microsoft, específicamente la denominada 5 estrellas.
- ✓ .NET es multi-lenguaje: esto quiere decir que para poder codificar aplicaciones sobre esta plataforma no necesitamos aprender un único lenguaje específico de programación de alto nivel, sino que se puede elegir de una amplia lista de opciones. Los lenguajes más populares son: Visual Basic .Net, C# .Net, J# .Net, C++, entre otros.
- ✓ .NET es una plataforma que permite el desarrollo de aplicaciones empresariales de misión crítica, entendiéndose por esto que permite la creación y ejecución de aplicaciones de porte corporativo que sean críticas para la operación de tipos variados de organizaciones. Si bien también es muy atrayente para desarrolladores no

profesionales, estudiantes y entusiastas, su verdadero poder radica en su capacidad para soportar las aplicaciones más grandes y complejas.

- ✓ .Net fue diseñado de manera tal de poder proveer un único modelo de programación, uniforme y consistente, para todo tipo de aplicaciones (ya sean de formularios Windows, de consola, aplicaciones Web, aplicaciones móviles, etc.) y para cualquier dispositivo de hardware (PC's, Pocket PC's, Teléfonos Celulares Inteligentes, también llamados "SmartPhones", Tablet PC's, etc.). Esto representa un gran cambio con respecto a las plataformas anteriores a .NET, las cuales tenían modelos de programación, bibliotecas, lenguajes y herramientas distintas según el tipo de aplicación y el dispositivo de hardware.
- ✓ Uno de los objetivos de diseño de .NET fue que tenga la posibilidad de interactuar e integrarse fácilmente con aplicaciones desarrolladas en plataformas anteriores, particularmente en COM, ya que aún hoy existen una gran cantidad de aplicaciones desarrolladas sobre esa base.
- ✓ .NET no sólo se integra fácilmente con aplicaciones desarrolladas en otras plataformas Microsoft, sino también con aquellas desarrolladas en otras plataformas de software, sistemas operativos o lenguajes de programación. Para esto hace un uso extensivo de numerosos estándares globales que son de uso extensivo en la industria, y acerca de los cuales iremos aprendiendo a lo largo del curso. Algunos ejemplos de estos estándares son XML, HTTP, SOAP, WSDL y UDDI.

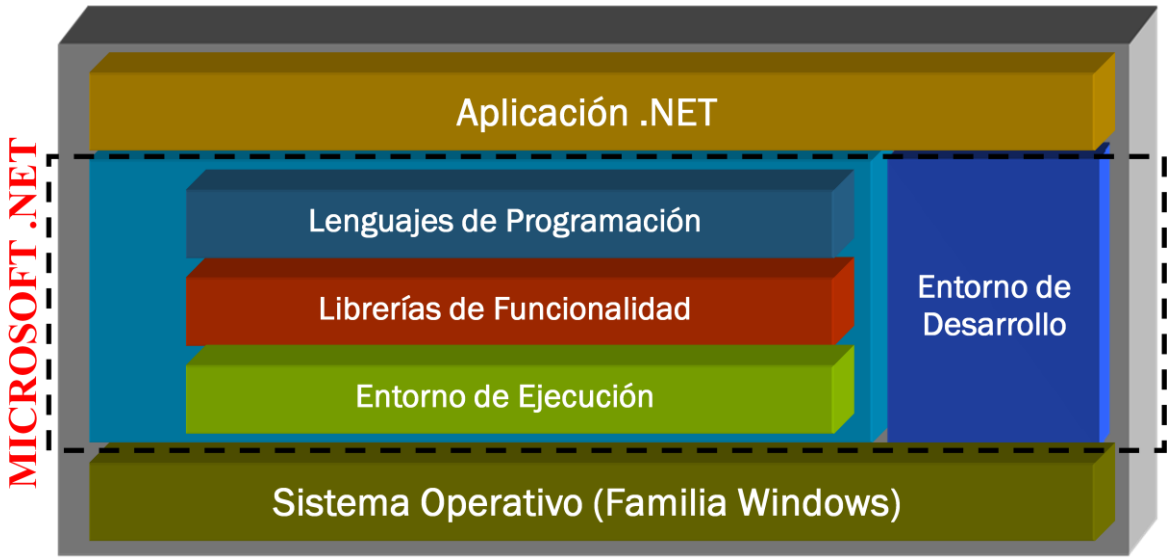


Figura 1.8.- Plataforma de Ejecución Intermedia

1.6.1.- ¿QUÉ ES EL .NET FRAMEWORK?

El .NET Framework (traducido como "Marco de Trabajo") es el componente fundamental de la plataforma Microsoft .NET, necesario tanto para poder desarrollar aplicaciones como para poder ejecutarlas luego en entornos de prueba o producción [1].

El .NET framework tiene tres variantes principales, todas descargables gratuitamente desde Internet

- ✓ .NET Framework Redistributable Package: este es el mínimo componente de la plataforma .NET que se necesita para poder ejecutar aplicaciones. Normalmente ésta es la variante que se instala en los entornos productivos, una vez que el desarrollo y las pruebas de la aplicación han finalizado.
  - Está compuesto por:
  - El entorno de ejecución de la plataforma .NET
  - Las bibliotecas de funcionalidad reutilizable
- ✓ .NET Framework SDK: esta versión contiene herramientas de desarrollo de línea de comandos (compiladores, depuradores, etc.), documentación de referencia, ejemplos y manuales para desarrolladores de aplicaciones. Normalmente ésta variante se instala en los entornos de desarrollo de aplicaciones, y es más útil a los programadores que a los usuarios finales. Para poder instalar la versión SDK (Software Development Kit) es necesario instalar previamente el Redistributable Package.
- ✓ .NET Compact Framework: esta es una versión reducida del .NET Framework Redistributable, especialmente pensada para ser instalada en dispositivos móviles como Pocket PC's y SmartPhones.

El .NET Framework puede ser instalado en cualquier sistema operativo de la familia Windows superior a Windows 98 [8]. Para más información acerca de los prerrequisitos se puede consultar: http://msdn.microsoft.com/netframework/technologyinfo/sysreqs/default.aspx

Actualmente, Windows 2003 Server y Windows XP SP2 traen el .NET Framework preinstalado. Para más información acerca de las descargas gratuitas, por favor consulte http://msdn.microsoft.com/netframework/downloads/updates/default.aspx

1.6.2.- ¿DONDE INSTALAR .NET?

El .NET Framework debe estar instalado en cualquier dispositivo de hardware para que la ejecución de una aplicación .NET sea posible. En el caso de las aplicaciones de escritorio (también llamadas "De Formularios Windows") y las aplicaciones de consola (aplicaciones cuya interfaz de usuario es una consola de comandos) [1], el Framework debe estar presente del lado del cliente (computadora donde se ejecuta la parte de la aplicación que interactúa con el usuario), y en el servidor sólo en caso de que la aplicación sea distribuida y tenga parte de su funcionalidad centralizada en una única computadora tal y como se muestra en la figura 1.9.

En el caso de las aplicaciones Web, el único requisito del lado del cliente es tener un navegador y una conexión de red al servidor, el cual debe tener instalado el .NET Framework [1,5].

Para las aplicaciones móviles, que se ejecutan sobre Windows Mobile en algún dispositivo tipo Pocket PC o SmartPhone, es necesario tener instalado el .NET Compact Framework en el dispositivo [1].

	Cliente	Servidor
Aplicación de Escritorio	✓	✓*
Aplicación Web		✓
Aplicación de Consola	✓	✓*
Aplicación Móvil	.NET Compact Framework	

Figura 1.9.- Modelo de Instalación del .Net Framework

1.6.3.- ARQUITECTURA DEL .NET FRAMEWORK

En la figura 1.10 se pueden apreciar las distintas partes que componen al .NET Framework, incluidas el entorno de ejecución de aplicaciones (CLR, en Verde), el conjunto de bibliotecas de funcionalidad reutilizable (.NET Framework Class Library, en Azul) y los compiladores y herramientas de desarrollo para los lenguajes .NET (en Rojo) tal y como l muestra la figura 1.10. Todos estos componentes se montan por encima de la familia de sistemas operativos Windows [1]. Dentro del conjunto de la .NET Framework Class Library se distinguen 4 sub-componentes principales:

- ✓ La Base Class Library (BCL - Biblioteca de Clases Base), que contiene la funcionalidad más comúnmente utilizada para el desarrollo de todo tipo de aplicaciones. Algunos ejemplos de la funcionalidad provista por la BCL son el manejo de colecciones, cadenas de texto, entrada/salida, threading, operaciones matemáticas y dibujos 2D.
- ✓ ADO.NET, que contiene un conjunto de clases que permiten interactuar con bases de datos relacionales y documentos XML como repositorios de información persistente.
- ✓ ASP.NET, que constituye la tecnología dentro del .NET Framework para construir aplicaciones con interfaz de usuario Web (es decir, aplicaciones cuya lógica se encuentra centralizada en uno o varios servidores y que los clientes pueden acceder usando un browser o navegador mediante una serie de protocolos y estándares como HTTP y HTML).

Windows Forms (o simplemente WinForms), que constituye la tecnología dentro del .NET Framewok que permite crear aplicaciones con interfaz de usuario basada en formularios y ventanas Windows de funcionalidad rica y que se ejecutan directamente en los clientes [8].

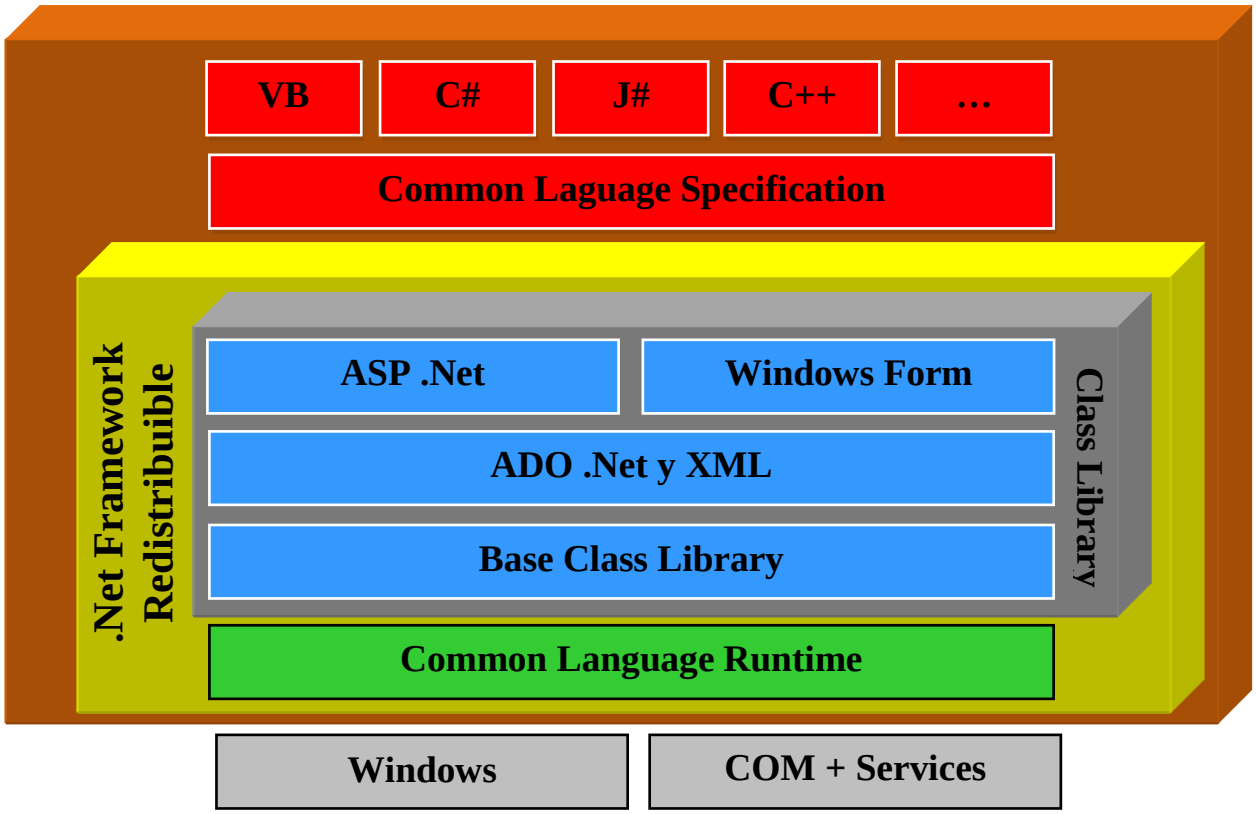


Figura 1.10.- Arquitectura del .Net Framework

1.6.4.- CLR COMMON LANGUAGE RUNTIME

Como ya se mencionó, el CLR actúa como un motor de ejecución de aplicaciones y componentes manejados. Ahora, se explican algunos de los principales servicios que les brinda a las aplicaciones que se ejecutan sobre él [1]:

- ✓ **Compilación Just In Time (o Justo A Tiempo**): el CLR se encarga de compilar las aplicaciones .NET a código de máquina nativo para el sistema operativo y la plataforma de hardware en la que se está ejecutando. Esto lo hace sin intervención alguna del desarrollador o el usuario, y solamente a medida que se necesita.
- ✓ **Gestión Automática de Memoria:** el CLR abstrae a los desarrolladores de tener que pedir y liberar memoria explícitamente. Para esto, uno de sus componentes llamado Garbage Collector (Recolector de Basura) se encarga de liberar periódicamente la memoria que ya no está siendo usada por ninguna aplicación. Por otra parte, el CLR también abstrae a los desarrolladores del uso de punteros y del acceso a memoria de bajo nivel. Si bien estas características pueden ser consideradas poderosas, suelen hacer el desarrollo y mantenimiento de aplicaciones más propenso a errores y menos productivo.
- ✓ **Gestión de Errores Consistente:** como las aplicaciones .NET no se ejecutan directamente contra el Sistema Operativo, cualquier error no manejado que ocurra en tiempo de

ejecución será atrapado por el CLR en última instancia, no afectando a ninguna otra aplicación que se esté ejecutando ni teniendo efecto alguno sobre su estabilidad.

- ✓ **Ejecución Basada en Componentes:** todas las aplicaciones .NET son empaquetadas en componentes reutilizables, denominados genéricamente Assemblies, que el CLR se encarga de cargar en memoria y ejecutar.
- ✓ **Gestión de Seguridad:** el CLR provee una barrera más de contención a la hora de ejecutar aplicaciones manejadas, ya que permite establecer políticas de seguridad muy detalladas que las aplicaciones .NET que se ejecuten en una determinada computadora deberán cumplir.
- ✓ **Multithreading:** el CLR provee un entorno de ejecución multi-hilos por sobre las capacidades del Sistema Operativo, asi como también mecanismos para asegurar su sincronización y acceso concurrente a recursos compartidos.

1.6.5.- .NET FRAMEWORK CLASS LIBRARY

De muy poco serviría a los desarrolladores el contar con una plataforma de ejecución de aplicaciones tan sofisticada y robusta como el CLR sin tener además un conjunto de funcionalidades y componentes empaquetados listos para aprovechar y reutilizar en sus aplicaciones. Justamente ese es el propósito de la .NET Framework Class Library (Biblioteca de Clases del .NET Framework), que provee cientos de tipos básicos (clases e interfaces principalmente) orientados a objetos, extensibles mediante herencia, independientes del lenguaje de programación de alto nivel que se desee utilizar y organizados en Namespaces jerárquicos. Para proporcionar un concepto más claro se muestra en la figura 1.11 las librerías de clase básicas utilizadas en el .net Framework, por lo anterior se muestran las siguientes características [1]:

- ✓ Conjunto de Tipos básicos (clases, interfaces, etc.) que vienen incluidos en el .NET Framework
- ✓ Los tipos están organizados en jerarquías lógicas de nombres, denominados NAMESPACES
- ✓ Los tipos son INDEPENDIENTES del lenguaje de desarrollo
- ✓ Es extensible y totalmente orientada a objetos

Figura 1.11.- Librerías de Clase del .Net Framework

Cabe mencionar que las librerías de clase se encuentran en constante evolución, esto significa que Microsoft publica la existencia de un nuevo Framework, aproximadamente cada año, entonces el primer Framework existe desde el año 2002 denominado .Net Framework 1.0 el cual contiene una colección de clases poderosas y listas para ser utilizadas para el desarrollo de software, sin embargo en el 2003 es publicado el .Net Framework 1.1 que incluye las librerías de clases de la versión anterior y algunas extras, de ésta manera ha ido evolucionando el contenido del .Net Framework desde el nacimiento de la versión 1.0, antes mencionado, el resto de las versiones: 1.1, 2.0, 2.5, 3.0 y el actual y poderoso .Net Framework 3.5.

1.6.6.- VISUAL BASIC .NET 2005

El lenguaje Visual Basic en su versión .Net, contiene en general, la misma forma de programar que la versión 6.0, es decir, existen el uso de las subrutinas utilizando la palabra reservada **sub** y también las funciones con la palabra reservada **function**, para los amantes de Visual Basic, éste es el lenguaje que deberán utilizar para emigrar a la tecnología .Net, sin embargo incluye una serie de componentes que caracterizan a .Net, que es la programación orientada a objetos.

La programación orientada a objetos está incluida en el lenguaje Visual Basic .Net, esto es, cuando se requiera comenzar a desarrollar un programa el desarrollador se topará con el uso de las clases y el uso de características propios de la programación orientada a objetos, como lo son:

- ✓ Objetos
- ✓ Clases
- ✓ Herencia
- ✓ Polimorfismo

Tipos de datos de .NET

Visual Basic .Net 2005 está totalmente integrado con .NET Framework, por lo tanto, los tipos de datos que podremos usar con este lenguaje serán los definidos en este "marco de trabajo", por este motivo vamos a empezar usando algunas de las definiciones que nos encontraremos al recorrer la documentación que acompaña a este lenguaje de programación.

Los tipos de datos que podemos usar en Visual Basic 2005 son los mismos tipos de datos definidos en .NET Framework y por tanto están soportados por todos los lenguajes que usan esta tecnología. Estos tipos comunes se conocen como el *Common Type System* (CTS), que traducido viene a significar el sistema de tipos comunes de .NET. El hecho de que los tipos de datos usados en todos los lenguajes .NET estén definidos por el propio Framework nos asegura que independientemente del lenguaje que estemos usando, siempre utilizaremos el mismo tipo interno de .NET, si bien cada lenguaje puede usar un nombre (o alias) para referirse a ellos, aunque lo importante es que siempre serán los mismos datos, independientemente de cómo se llame en cada lenguaje. Esto es una gran ventaja, ya que nos permite usarlos sin ningún tipo de problemas para acceder a ensamblados creados con otros lenguajes, siempre que esos lenguajes sean compatibles con los tipos de datos de .NET [8].

En la siguiente lista se muestran los tipos de datos soportados en el lenguaje Visual Basic .Net 2005:

- ✓ Tipos primitivos
- ✓ Variables y constantes
- ✓ Enumeraciones: Constantes agrupadas
- ✓ Arrays (matrices)

Clases y Estructuras

En la tecnología de .Net también y específicamente en Visual Basic .Net existen los distintos niveles de accesibilidad que se puede aplicar a los tipos, así como a los distintos miembros de esos tipos de datos [1]. De los distintos miembros que se pueden definir como tipos, nos centraremos en las propiedades para ver en detalle los cambios que han sufrido con respecto a VB6. También veremos temas relacionados con la programación orientada a objetos (POO) en general y de forma particular los que atañen a las interfaces.

- ✓ Clases: Tipos por referencia definidos por el usuario
- ✓ Definir una clase
- ✓ Instanciar: Crear un objeto en la memoria
- ✓ Estructuras: Tipos por valor definidos por el usuario
- ✓ Accesibilidad y ámbito
- ✓ Propiedades
- ✓ Interfaces

Manejo de Excepciones

Es indiscutible que por mucho que nos lo propongamos, nuestras aplicaciones no estarán libres de errores, y no nos referimos a errores sintácticos, ya que, afortunadamente, el IDE (*Integrated Development Envirnment*, entorno de desarrollo integrado) de Visual Basic 2008 nos avisará de cualquier error sintáctico e incluso de cualquier asignación no válida (al menos si tenemos activado *Option Strict On*), pero de lo que no nos avisará, como es lógico, será de los errores que se produzcan en tiempo de ejecución. Para estos casos, Visual Basic pone a nuestra disposición el manejo de excepciones, veamos pues cómo utilizarlo, sobre todo el sistema de excepciones estructuradas que es el recomendable para cualquier desarrollo con .NET Framework [1].

Eventos y Delegados

La forma que tienen nuestras clases y estructuras de comunicar que algo está ocurriendo, es por medio de eventos. Los eventos son mensajes que se lanzan desde una clase para informar al "cliente" que los utiliza de que está pasando algo.

Seguramente estaremos acostumbrados a usarlos, incluso sin tener una noción consciente de que se tratan de eventos, o bien porque es algo tan habitual que no le prestamos mayor atención, es el caso de las aplicaciones de escritorio, cada vez que presionamos un botón, escribimos algo o movemos el mouse se están produciendo eventos.

El compilador de Visual Basic 2005 nos facilita mucho la creación de los eventos y "esconde" todo el proceso que .NET realmente hace "en la sombra". Ese trabajo al que nos referimos está relacionado con los delegados, una palabra que suele aparecer en cualquier documentación que trate sobre los eventos [8].

Y es que, aunque Visual Basic 2005 oculte, o facilite, el trabajo con los eventos, éstos están estrechamente relacionados con los delegados junto con la intercepción de eventos.

En conclusión, .Net tiene utiliza los eventos para poder interactuar de manera efectiva en el desarrollo de software, éstos pueden ser disparados en el momento deseado y logra ser una herramienta útil y eficaz en las aplicaciones .Net.

Atributos

Esta es la definición que nos da la ayuda de Visual Basic sobre lo que es un atributo.

Los atributos son etiquetas descriptivas que proporcionan información adicional sobre elementos de programación como tipos, campos, métodos y propiedades. Otras aplicaciones, como el compilador de Visual Basic, pueden hacer referencia a la información adicional en atributos para determinar cómo pueden utilizarse estos elementos [1].

1.7.- SQL SERVER 2005

Las Bases de Datos han ido emigrando con el paso del tiempo para cubrir las necesidades del ser humano en relación a sistemas informáticos con acceso a bases de datos: Ésta necesidad requiere que los manejadores de bases de datos e incluso las bases de datos tengan características de soporte, consistencia, velocidad, integridad, etc., e incluso innovación, como lo es SQL Server 2005.

SQL Server en su versión 2005, tiene un gran soporte para albergar grandes masas de información en bases de datos, e incluso innovación, con el uso de los Procedimientos Almacenados (Store Procedures), o los disparadores de eventos (Triggers), soporte distribuido y varias herramientas más existentes en ésta versión.

SQL Server 2005, es un manejador de Bases de Datos que se encuentra en competencia con el resto de los manejadores de bases de datos, estando a la vanguardia tecnológica con el simple propósito de cuidar la información de bases de datos de las empresas, instituciones educativas, investigación gobierno, etc.

En lo particular, elegí SQL Server 2005 por su gran facilidad en su instalación y manipulación de datos y por supuesto su gran soporte, he probado éste manejador en varias estaciones de trabajo e incluso en servidores y trabaja sin problema alguno, y todavía mejor, la integración del lenguaje, en éste documento mencionado, Visual Basic 2005, hace mucho más confiable la utilización de éste manejador de base de datos, logrando un complemento eficaz en el mundo del desarrollo de las bases de datos.

Microsoft ofrece una versión express de SQL server (SSE) para poder probar su gran soporte. Se le denomina express porque tiene una limitante en comparación con la versión profesional, la cual consta en que su administración llega únicamente a los 4 Gb., sin embargo, recomiendo ampliamente la utilización de la versión express para probar sus herramientas y características.

SSE es una versión reducida funcionalmente de SQL Server 2005 destinados a varias áreas de mercado. Estos incluyen a estudiantes que no pueden permitirse el lujo de comprar la totalidad de SQL Server 2005 y los usuarios que desean ejecutar SQL Server en la parte inferior de las especificaciones de hardware. Por ejemplo, si se tiene un sitio web basado en un servidor SQL Server anfitrión, puede desarrollar su sistema en la ESS y a continuación, migrar a su host web sin necesidad de comprar la versión completa

SSE puede apoyar una amplia variedad de aplicaciones, tanto para los pequeños usuarios y las empresas más grandes. Sin embargo, SSE entra de lleno en la menor categoría de aplicación, adecuado para un pequeño número de usuarios y, probablemente, uno o dos desarrolladores. Asimismo, las metas aplicaciones que requieren una base de datos construida en GUI, como Microsoft Access y tiene la capacidad para aumentar la escala de tamaño en si

requerido. Si se comienza a trabajar en SQL Server Express, se puede escalar su aplicación a otras de SQL Versiones de servidor con toda la posibilidad utilizando poco trabajo.

Los requerimientos mínimos en hardware recomendados para instalar y utilizar SQL server Express [1] se muestran en la Tabla 1.1.

Requisitos previos de software	Microsoft .NET Framework SP1 de Microsoft Internet Explorer 6.0 o posterior
RAM	Mínimo: 192 MB Recomendado: 512 MB o más
Espacio en el disco duro	600 MB de espacio libre
Procesador	Compatible con Pentium III o superior Mínimo: 500 MHz Recomendado: 1 GHz o más
Sistema operativo	Windows Server 2003 Standard Edition, Enterprise Edition, Datacenter Edition Windows XP Professional, Home Edition (SP2 o posterior) Windows 2000 Professional, Server, Advanced Server, Datacenter Server (SP4 o posterior)

Tabla 1.1.- Requerimientos Mínimos en Hardware para SQL Server Express

Para instalar SQL Server Express solo hay que descargarlo de la página de Microsoft de forma gratuita, en la siguiente dirección URL :
http://www.desarrollaconmsdn.com/msdn/CursosOnline/Curso_SQL_Server/index.html
Se específica con detalle sobre su instalación y administración, mostrando de una forma fácil de efectuar ésta tarea.

1.8.- ADO .NET

Microsoft incluye una serie de herramientas poderosas para acceso a bases de datos, en específico, conectar una aplicación Visual Basic .Net 2005 con SQL server 2005 y para ello es necesario utilizar el concepto ADO .Net.

ADO .Net, no es más que un concepto de accesos a bases de datos, que utiliza una serie de clases, métodos, propiedades, funciones, etc., para conectarse a una base de datos y poderla administrar.

ADO .Net, básicamente, trabaja en dos vertientes denominados: Ambientes Desconectados y Ambientes Conectados, ambas trabajan de forma diferente sin perder de vistas las operaciones básicas sobre una base de datos que son: Inserción, Eliminación, Actualización y Consulta.

Los Ambientes Desconectados trabajan sobre memoria, memoria utilizada de la PC del cliente, y los Ambientes Conectados trabajan directamente sobre la base de datos, cada

ambiente requiere de la utilización de una serie de clases particulares, sin embargo la conexión es única para cualquiera de éstos dos conceptos. Para mostrar ésta explicación se representa en la tabla 1.2.

Nivel	Clases Utilizadas en Ambinetes Conectados	Clases Utilizadas en Ambinetes Desconectados
IDE	ListView, ListBox, Combobox	DataGrid, DataGridView
Memoria	DataReader	DataSet
Trasmisión	Command	DataAdapter
Conexión	Conexión SQL Server: **SQL**Connection Conexión ODBC: **ODBC**Connection Conexión OLEDB (pj. Access): **OLEDB**Connection Conexión Oracle: **ORACLE**Connection	

Tabla 1.2.- Representación Rápida del modelo ADO .Net

1.8.1.- ARQUITECTURA DE ADO .NET

El concepto más importante que hay que tener claro sobre ADO.NET [1] es su modo de funcionar, que se revela claramente al analizar su arquitectura:

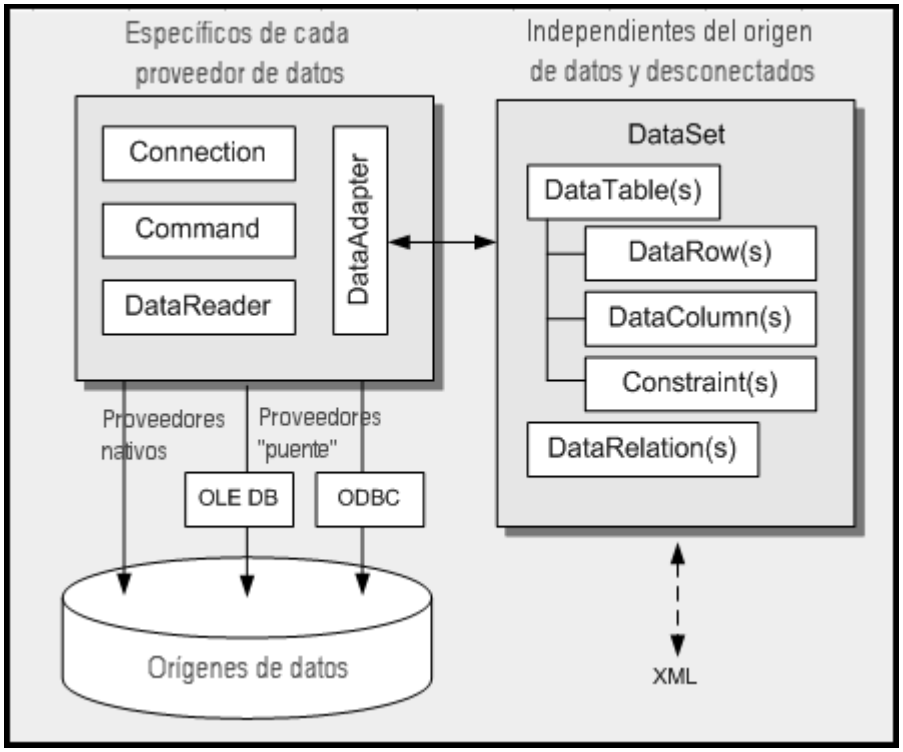


Figura 1.12.- Arquitectura ADO .Net

1.8.2.- AMBIENTES CONECTADOS

La capa conectada de ADO.NET contiene objetos especializados en la conexión con los orígenes de datos. Así, la clase genérica *Connection* se utiliza para establecer conexiones a los orígenes de datos. La clase *Command* se encarga de enviar comandos de toda índole al origen de datos. Por fin la clase *DataReader* está especializada en leer los resultados de los comandos [12].

La clase *DataAdapter* hace uso de las tres anteriores para actuar de puente entre la capa conectada y la desconectada como se verá después.

Estas clases son abstractas, es decir, no tienen una implementación real de la que se pueda hacer uso directamente. Es en este punto en donde entran en juego los proveedores de datos. Cada origen de datos tiene un modo especial de comunicarse con los programas que los utilizan, además de otras particularidades que se deben contemplar. Un proveedor de datos de ADO.NET es una implementación concreta de las clases conectadas abstractas que hemos visto, que hereda de éstas y que tiene en cuenta ya todas las particularidades del origen de datos en cuestión.

Así, por ejemplo, las clases específicas para acceder a SQL Server se llaman **SqlConnection**, **SqlCommand**, **SqlDataReader** y **SqlDataAdapter** y se encuentran bajo el espacio de nombres **System.Data.SqlClient**. Es decir, al contrario que en ADO clásico no hay una única clase **Connection** o **Command** que se use en cada caso, si no que existen clases especializadas para conectarse y recuperar información de cada tipo de origen de datos.

Nota:
El hecho de utilizar clases concretas para acceso a las fuentes de datos no significa que no sea posible escribir código independiente del origen de datos, todo lo contrario. La plataforma .NET ofrece facilidades de escritura de código genérico basadas en el uso de herencia e implementación de interfaces. De hecho la versión 2.0 de .NET ofrece grandes novedades, específicamente en éste ámbito [12].

Existen proveedores nativos, que son los que se comunican directamente con el origen de datos (por ejemplo el de SQL Server o el de Oracle), y proveedores "puente", que se utilizan para acceder a través de *ODBC* u *OLEDB* cuando no existe un proveedor nativo para un determinado origen de datos.

Nota:
Estos proveedores puente, si bien muy útiles en determinadas circunstancias, ofrecen un rendimiento menor debido a la capa intermedia que están utilizando (*ODBC* u *OLEDB*). Un programador novel puede sentir la tentación de utilizar siempre el proveedor puente para *OLEDB* y así escribir código compatible con diversos gestores de datos de forma muy sencilla. Se trata de un error y siempre que sea posible es mejor utilizar un proveedor nativo [12].

La plataforma .NET proporciona "de serie" los siguientes proveedores de acceso a datos.

Proveedor	Espacio de nombres	Descripción
ODBC .NET Data Provider	System.Data.Odbc	Permite conectar nuestras aplicaciones a fuentes de datos a través de ODBC.
OLE DB .NET Data Provider	System.Data.OleDb	Realiza la conexión utilizando un proveedor OLEDB, al igual que el ADO clásico.
Oracle Client .NET Data Provider	System.Data.OracleClient	Proveedor de datos para acceder a Oracle.
SQL Server .NET Data Provider	System.Data.SqlClient	Permite la conexión optimizada a SQL Server 7.0 o posterior, incluyenbdo la última versión SQL Server 2005.

Tabla 1.3.- Plataforma Ambientes Conectados

Los proveedores de acceso a datos que distribuye Microsoft en ADO.NET y algunos desarrollados por otras empresas o terceros, contienen los mismos objetos, aunque los nombres de éstos, sus propiedades y métodos, pueden ser diferentes.

Existen, por supuesto, proveedores para tipos de orígenes de datos de cualquier gestor de datos existente en el mercado. Éstos los desarrolla normalmente la empresa responsable del producto. Si bien éstos optimizan el acceso a estos orígenes de datos siempre existirá la forma de realizarlo mediante *ODBC* u *OLEDB* si existe la necesidad.

En resumen: con la capa conectada de ADO.NET se puede establecer la conexión y comunicación con los orígenes de datos. Cada proveedor de datos implementa su propia versión de las clases *Connection*, *Command*, *DataReader* y *DataAdapter* (entre otras).

- ✓ Las clases derivadas de *Connection* se utilizan para realizar la conexión, enviar y recibir información.
- ✓ Las clases derivadas de *Command* permiten ejecutar sentencias SQL y procedimientos almacenados en el gestor de datos.
- ✓ Las clases derivadas de *DataReader* se emplean para obtener los posibles resultados de un comando utilizando para ello el conducto de comunicación establecido por *Connection*.

1.8.3.- AMBIENTES DESCONECTADOS

Una vez que ya se han recuperado los datos desde un origen de datos la conexión a éste ya no es necesaria. Sin embargo sigue siendo necesario trabajar con los datos obtenidos de una manera flexible. Es aquí cuando la capa de datos desconectada entra en juego. Además, en muchas ocasiones es necesario tratar con datos que no han sido obtenidos desde un origen de datos relacional con el que se requiera una conexión. A veces únicamente es necesario un

almacén de datos temporal pero que ofrezca características avanzadas de gestión y acceso a la información.

Las conexiones con las bases de datos son uno de los recursos más escasos con los que se cuenta al desarrollar. Su mala utilización es la causa más frecuente de cuellos de botella en las aplicaciones y de que éstas no escalen como es debido. Esta afirmación es especialmente importante en las aplicaciones Web en las que se pueden recibir muchas solicitudes simultáneas de cualquier parte del mundo.

Otro motivo por el que es importante el uso de los datos desconectado de su origen es la transferencia de información entre capas de una aplicación. Éstas pueden encontrarse distribuidas por diferentes equipos, e incluso en diferentes lugares del mundo gracias a Internet. Por ello es necesario disponer de algún modo genérico y eficiente de poder transportar los datos entre diferentes lugares, utilizarlos en cualquiera de ellos y posteriormente tener la capacidad de conciliar los cambios realizados sobre ellos con el origen de datos del que proceden. Todo esto y mucho más es lo que otorga el uso de los objetos *DataSet*, núcleo central de la capa desconectada de ADO.NET.

Nota:
Otra interesante característica de los *DataSet* es que permiten gestionar simultáneamente diversas tablas (relaciones) de datos, cada una de un origen diferente si es necesario, teniendo en cuenta las restricciones y las relaciones existentes entre ellas [12].

Los *DataSet*, como cualquier otra clase no sellada de .NET, se pueden extender mediante herencia. Ello facilita una técnica avanzada que consiste en crear tipos nuevos de DataSet especializados en la gestión de una información concreta (por ejemplo un conjunto de tablas relacionadas). Estos nuevos tipos de clases se denominan genéricamente *DataSet* Tipados, y permiten el acceso mucho más cómodo a los datos que representan, verificando reglas de negocio, y validaciones de tipos de datos más estrictas.

Los objetos *DataSet* no dependen de un proveedor de datos y su funcionamiento es independiente de cómo hayan sido obtenidos los datos que contienen. Este es el concepto más importante que se debe recordar.

El proceso general de trabajo de ADO.NET para acceder a un gestor de datos (SQL Server, por ejemplo) es casi siempre el mismo: se abre una conexión (clase *Connection*), se lanza una consulta SQL o procedimiento almacenado mediante un objeto de la clase *Command*, y se almacenan en memoria los resultados dentro de un objeto *DataSet*, cerrando la conexión.

Nota:
Aunque este es el comportamiento habitual de una aplicación de datos existen casos en los que no es recomendable almacenar en memoria (en un *DataSet*) todos los resultados de una consulta, bien por ser muchos registros o por contener datos muy grandes. En este tipo de casos se puede usar un objeto *DataReader* directamente para leer la información, tratarla y no almacenarla en lugar alguno. De todos modos lo más frecuente es realizar el proceso descrito.

1.9.- WEB SERVICE

Introducción

Un servicio Web XML es una entidad programable que proporciona un elemento de funcionalidad determinado, como lógica de aplicación, al que se puede tener acceso desde diversos sistemas potencialmente distintos mediante estándares de Internet muy extendidos, como XML y HTTP [5].

Un servicio Web XML puede ser utilizado internamente por una aplicación o bien ser expuesto de forma externa en Internet por varias aplicaciones. Dado que a través de una interfaz estándar es posible el acceso a un servicio Web XML, éste permite el funcionamiento de una serie de sistemas heterogéneos como un conjunto integrado [5].

Simplificando, y siendo algo más prácticos, podemos definir un servicio Web XML como una clase a la que podemos acceder utilizando estándares de Internet.

Como es de suponer, tener que utilizar esos estándares de comunicación de Internet es porque esa "clase" está alojada en un servidor de Internet, es decir, un servicio Web es una clase que está alojada en la Web y que podemos acceder a ella mediante ciertos estándares como XML, que a su vez utiliza otro estándar: SOAP, (Simple Object Access Protocol), que es el lenguaje que define cómo nos comunicaremos con el servicio Web.

¿Qué son las Web Service?

La expresión "Servicio Web" se oye con fuerza desde hace unos años en el ámbito del desarrollo de aplicaciones e incluso en ambientes poco técnicos y de dirección. Lo cierto es que no se trata de un concepto tan novedoso como cabría esperar y las innovaciones que conlleva no son tanto tecnológicas, como conceptuales.

Historia de los Modelos de Desarrollo

El hecho de poder comunicar componentes de software entre sí tiene una enorme importancia. Hasta hace pocos años era muy típico hacer **aplicaciones de una sola pieza**, "monolíticas" como se muestra en la figura 1.13.

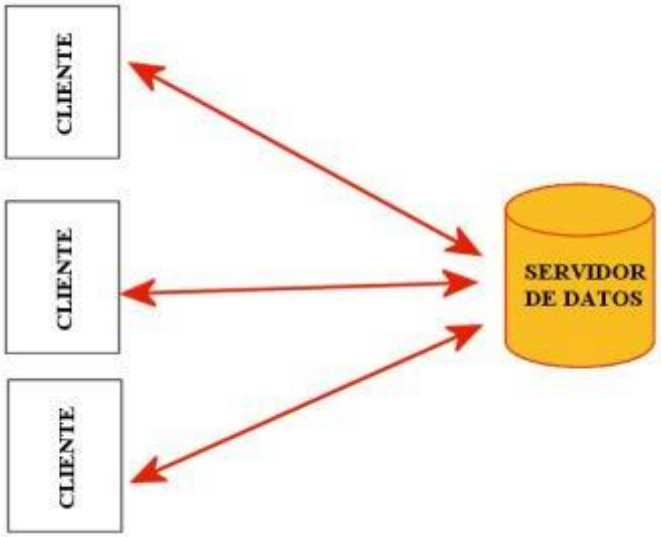


Figura 1.13.- Aplicación "Monolítica" aunque Distribuida

Estos programas podían acceder a un sistema gestor de datos a través de la red, pero toda la lógica del flujo de datos, la seguridad y las interacciones con las personas se encontraban en el ordenador del usuario en forma de un gran ejecutable. Se suelen conocer también como "*fat clients*". La situación descrita no es la ideal ya que implica problemas de toda índole a la hora de instalar las aplicaciones y sobre todo cuando se modifican o actualizan (mantenimiento complejo y engorroso).

Una metodología de desarrollo mucho mejor aunque más laboriosa a la hora de programar es el modelo **Cliente-Servidor en tres capas**:

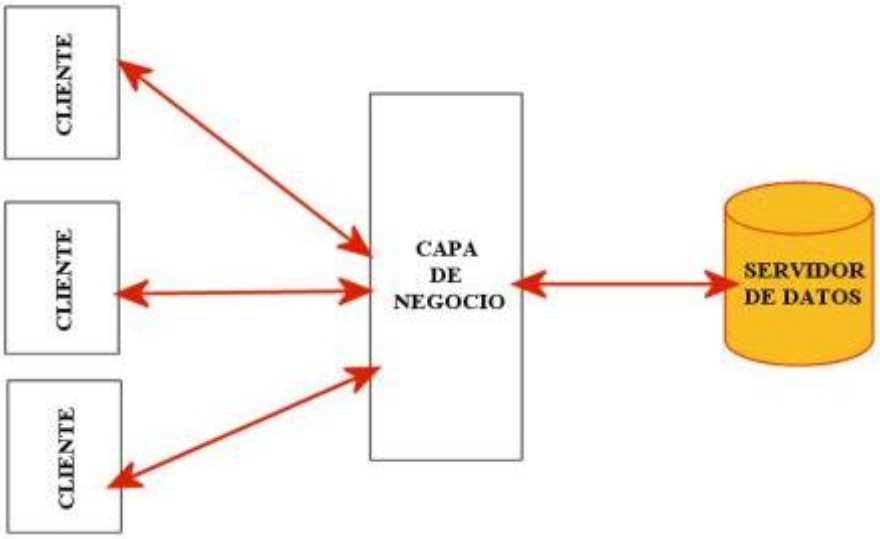


Figura 1.14.- Aplicación Cliente Servidor en Tres Capas

En este modelo toda la lógica de los datos, su validación, los permisos, etc., residen en un servidor intermedio y son utilizados por todos los clientes a través de una red. En este caso en el ordenador del usuario lo único que hay es una capa de presentación que se ocupa básicamente de recoger y recibir datos, es decir, actúa de intermediario entre el usuario y las reglas de negocio residentes en la capa intermedia. Este modelo es más eficiente y está muy evolucionado respecto al anterior pero aún se puede ir más allá.

La arquitectura de **desarrollo en n-capas** (*n-tier* que dicen los anglosajones), figura 1.15 lleva el concepto cliente-servidor un paso hacia adelante, dividiendo la capa intermedia en muchas otras capas especializadas, cada una de las cuales puede residir en un servidor diferente [5]:

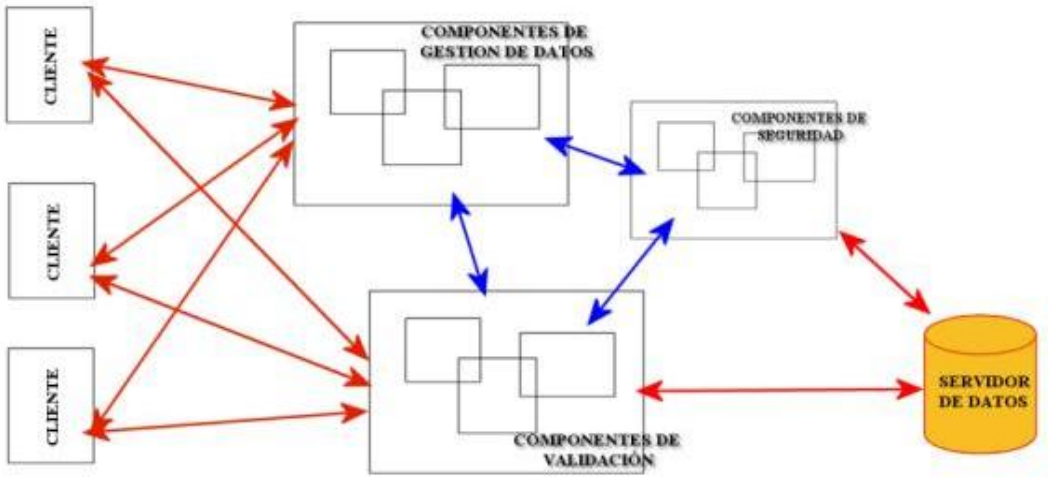


Figura 1.15.- Arquitectura de Desarrollo Basada en Componentes

En este modelo existe una gran variedad de componentes especializados en tareas específicas como la validación de datos, la autenticación y seguridad o el acceso a datos. Dichos componentes deben trabajar unos con otros como piezas de un mecanismo, gestionando la información que circula entre el usuario y el servidor de datos.

La belleza de este modelo radica en que cada uno de ellos (o cada grupo de ellos) puede residir en un servidor diferente, siendo transparente su ubicación para los clientes que los utilizan. Ello aumenta mucho la escalabilidad de las aplicaciones, pues basta con añadir nuevos servidores e instalar los componentes en ellos para poder atender más peticiones.

Por otra parte, y esto es muy interesante también, mientras sus interfaces de programación sean las mismas, es posible sustituir cualquier componente por otro actualizado o que actúe de manera distinta para corregir errores o cambiar el modo de trabajo de la aplicación global, y todo ello sin que los clientes sean conscientes de ello. Obviamente esto ofrece más ventajas aunque, por ejemplo, no es necesario reinstalar la aplicación en cada cliente, sino que basta con sustituir un componente en un único lugar y automáticamente todos

los usuarios tendrán su aplicación actualizada.

El concepto de **Arquitectura Orientada a Servicios** o **SOA** se basa en el uso de este tipo de componentes que suplen las necesidades de una o varias aplicaciones, son independientes entre sí y trabajan independientemente del sistema operativo o la plataforma [5].

Aunque muchos programadores piensan que SOA está relacionado únicamente con los Servicios Web lo cierto es que se pueden conseguir arquitecturas SOA con otras tecnologías.

Comunicación entre Componentes

Existen diversas dificultades técnicas a la hora de llevar a la práctica las arquitecturas orientadas a servicios. Entre éstas están la comunicación entre las distintas capas y componentes que constituyen la aplicación, la gestión de peticiones y el balanceado de carga entre servidores, cuando un mismo componente reside en varios de ellos (para aplicaciones muy grandes con muchos clientes), la gestión de transacciones entre componentes y algunas otras cosas más.

Existe la posibilidad de escribir un protocolo de comunicaciones propio que se ocupe de todas estas cuestiones, pero por supuesto se trata de una opción muy poco realista dada la complejidad que conllevaría. Para responder a estas necesidades de los programadores, diversos fabricantes y asociaciones de la industria crearon servicios y protocolos específicos orientados a la interacción distribuida de componentes. Aunque existe una gran variedad, de todos ellos los más importantes sin duda son:

- ✓ DCOM (Distributed Common Object Model), la propuesta de Microsoft, ligada a sus sistemas Windows. Se trata de algo más que un protocolo de invocación remota de procedimientos (RPC) ya que su última encarnación, COM+, incluye servicios avanzados para balanceado de carga, gestión de transacciones o llamadas asíncronas. Los parámetros son transmitidos a través de la red mediante un formato binario propio llamado NDR (Network Data Representation).
- ✓ RMI (Remote Method Invocation), es la metodología de llamada remota a procedimientos de Java. No se centra en la definición de interfaces para compatibilidad binaria de componentes, ni en otros conceptos avanzados, y se basa en la existencia de un cliente y un servidor que actúan de intermediarios entre los componentes que se quieren comunicar. Es una tecnología bastante simple que es fácil de utilizar para aplicaciones básicas.
- ✓ CORBA (Common Object Request Broker Architecture). Se trata de una serie de convenciones que describen cómo deben comunicarse los distintos componentes, cómo deben transferir los datos de las llamadas y sus resultados o cómo se describen las interfaces de programación de los componentes para que los demás sepan cómo utilizarlos. Fue desarrollado por el OMG (Object Management Group) en la segunda mitad de la década de los '90 y es el modelo que más éxito ha tenido en el mundo UNIX. Su método de empaquetado y transmisión de datos a través de la red se llama CDR

(Common Data Representation). Existen diversas implementaciones de distintos fabricantes.

Estos modelos son buenos y muy eficientes, cumpliendo bien su trabajo pero tienen algunas limitaciones importantes siendo las principales las siguientes:

- ✓ Es difícil la comunicación entre los distintos modelos
- ✓ Están ligados a plataformas de desarrollo específicas, lo que dificulta la comunicación entre ellas
- ✓ Su utilización a través de Internet se complica debido a cuestiones de seguridad.
- ✓ Existen en el mercado puentes CORBA/DCOM que permiten la comunicación entre componentes COM y componentes CORBA, pero su utilización es difícil y añaden una nueva capa de complejidad a las aplicaciones además de disminuir su rendimiento.

SOAP

Las expectativas actuales respecto a los componentes han aumentado. Al igual que podemos usar un navegador web para acceder a cualquier página independientemente del sistema operativo del servidor en que resida, ¿por qué no podríamos invocar métodos de componentes a través de la red independientemente de dónde se encuentren, del lenguaje en el que estén escritos y de la plataforma de computación en la que se ejecuten? [5].

Esto es precisamente lo que ofrecen los Servicios Web. Gracias a ellos se derriban las antiguas divisiones resultantes de los modelos de componentes descritos, y la integración de las aplicaciones, la ubicuidad de sus componentes y su reutilización a través de la red se convierten en una realidad [5].

La tecnología que está detrás de todo ello se llama SOAP (jabón en inglés). Este acrónimo (Simple Object Access Protocol) describe un concepto tecnológico basado en la sencillez y la flexibilidad que hace uso de tecnologías y estándares comunes para conseguir las promesas de la ubicuidad de los servicios, la transparencia de los datos y la independencia de la plataforma que según hemos visto, se hacen necesarios en las aplicaciones actuales [5].

Breve Historia de SOAP

SOAP empezó como un protocolo de invocación remota basado en XML diseñado por Dave Winer de UserLand, llamado XML-RPC. A partir de éste se obtuvo en Septiembre de 1999 la versión 1.0 de SOAP, en la que participo activamente Microsoft y el archiconocido experto en programación Don Box.

Esta primera versión fue más o menos despreciada por los principales fabricantes de software que en esa época tenían en marcha un proyecto más ambicioso llamado ebXML. Esto puso en peligro en su nacimiento la existencia de SOAP ya que si los grandes no lo apoyaban poco se podía hacer. Por fortuna uno de estos grandes fabricantes, IBM, decidió apoyarlo y en la

actualidad no sólo acepta SOAP sino que es uno de lo motores detrás de su desarrollo (dos importantes personas de IBM y su filial Lotus, David Ehnebuske y Noah Mendelsohn). Sun Microsystems también anunció oficialmente en Junio de 2000 que soportaba el estándar. El hecho de que los tres gigantes del software (Microsoft, IBM y Sun) apoyen SOAP ha hecho que muchos fabricantes de Middleware y puentes CORBA-DCOM (como Roguewave o IONA) ofrezcan productos para SOAP, así como otras muchas pequeñas empresas de software.

El paso definitivo para asegurar el éxito de SOAP y los servicios web es su envío al W3C (World Wide Web Consortium) para proponerlo como estándar. La última versión de la especificación se puede encontrar en www.w3.org/TR/SOAP/.

Este soporte mayoritario hace que su éxito y supervivencia estén asegurados y hoy todas las herramientas de desarrollo del mercado ofrecen en menor o mayor medida soporte para SOAP. Por supuesto .NET y Visual Studio son los entornos más avanzados en la adopción de SOAP.

La Base Tecnológica de SOAP

Lo interesante de SOAP es que utiliza para su implementación tecnologías y estándares muy conocidos y accesibles como son XML o el protocolo HTTP.

Dado que los mensajes entre componentes y los datos de los parámetros para llamadas a métodos remotos se envían en formato XML basado en texto plano, SOAP se puede utilizar para comunicarse con cualquier plataforma de computación, consiguiendo la ansiada ubicuidad de los componentes.

El uso de HTTP como protocolo principal de comunicación hace que cualquier servidor web del mercado pueda actuar como servidor SOAP, reduciendo la cantidad de software a desarrollar y haciendo la tecnología disponible inmediatamente. Además en la mayoría de los casos se puede hacer uso de SOAP a través de los cortafuegos que defienden las redes, ya que no suelen tener bloqueadas las peticiones a través del puerto 80, el puerto por defecto de HTTP (de ahí la ubicuidad, aunque se pueden usar otros puertos distintos al 80, por supuesto).

La seguridad se puede conseguir a través de los métodos habituales en los servidores web y por tanto se dispone de autenticación de usuarios y cifrado de información de forma transparente al programador, usando protocolos y técnicas como IPSec o SSL, ampliamente conocidos y usados en el mundo web.

Por otra parte la escalabilidad se obtiene a través del propio servidor web o incluso del sistema operativo, ya que la mayoría de ellos (por ejemplo IIS) poseen capacidades de ampliación mediante clusters de servidores, enrutadores que discriminan las peticiones y otras técnicas para crear Web Farms, o conjuntos de servidores web que trabajan como si fueran uno solo para así poder atender a más clientes simultáneamente.

Nota:
Existen ampliaciones al protocolo SOAP base que definen protocolos y convenciones para tareas específicas como las mocionadas de seguridad, enrutador de mensajes, los eventos y muchas otras cuestiones avanzadas. En .NET se implementan mediante los conocidos Web Services Enhancements (WSE) actualmente por su versión 3.0, y en un futuro inmediato con Windows Communication Foundation, la nueva plataforma de servicios de comunicaciones de Windows[5].

Como vemos, las tecnologías utilizadas son conocidas y la especificación SOAP se refiere más bien a la manera de usarlas. De este modo las áreas cubiertas por la especificación se refieren a cómo se codifican los mensajes XML que contienen las llamadas a procedimientos y sus respuestas, y a la manera en que HTTP debe intercambiar estos mensajes. Si nos referimos a la esencia del estándar, SOAP trata de sustituir a los diferentes formatos propietarios de empaquetamiento de datos que utilizan otras tecnologías (como DCOM o CORBA con NDR y CDR respectivamente), así como los protocolos propietarios empleados para transmitir estos datos empaquetados.

HTTP es el único protocolo definido en el estándar para SOAP pero éste es lo suficientemente abierto como para permitir que se empleen otros protocolos distintos para transmitir mensajes SOAP. Por citar unos pocos, se podría utilizar SMTP (correo electrónico), MSMQ (Microsoft Messaging Queue) para enviar de manera asíncrona las llamadas a procedimientos con SOAP, etc.

Descubrimiento de Servicios: WSDL y UDDI

Otro de los estándares que se definen en SOAP es WSDL (Web Service Definition Language). Se trata de un formato estándar para describir las interfaces de los servicios web. WSDL describe qué métodos están disponibles a través de un servicio Web y cuáles son los parámetros y valores devueltos por éstos. Antes de usar un componente que actúa como servicio web se debe leer su archivo WSDL para averiguar cómo utilizarlo.

Nota:
Para aquellos programadores que conocen otras arquitecturas podemos decir que WSDL es el equivalente en XML a los lenguajes IDL (Interface Description Language) de DCOM y CORBA.

Se ha definido también un formato estándar para publicación de información de servicios web llamado UDDI (Universal Description Discovery and Integration). Esta especificación permite la creación de directorios de servicios web, donde se definen métodos que permiten consultarlos para encontrar fácilmente aquel servicio que se necesite. Windows Server 2003 incluye gratuitamente un servidor para implementar directorios UDDI en organizaciones.

CAPITULO 2. PERSONAL SOFTWARE PROCESS

El trabajo de un ingeniero de software es entregar productos de alta calidad a unos costes establecidos y en un plazo determinado. Por lo anterior, existen tres aspectos que hacen efectivo un trabajo de ingeniero de software: producir productos de alta calidad, hacer el trabajo a los costes esperados y completar el trabajo de acuerdo con la planificación establecida [3].

La disciplina se define como una actividad o ejercicio que desarrolla o mejora habilidades. Contrariamente a la visión popular de la disciplina como una limitación excesiva, es un marco de trabajo para aprender y mejorar personalmente. La disciplina de PSP proporciona un marco de trabajo estructurado para desarrollar las habilidades personales y los métodos que necesitará el ingeniero de software. La cuestión no es si necesita habilidades personales sino cuánto tiempo necesita para desarrollarlas y cómo las utiliza de forma consistente [3].

El Personal Software Process (PSP)[SM] esta basado en un proceso definido y diseñado por Watts Humphrey del SEI (Software Enginnering Institute), PSP está orientado a Ingenieros de Software, Ingenieros de Calidad, Arquitectos, Programadores, Líderes Técnicos, Gerentes o Directores encargados de las áreas de desarrollo de software, Responsables de un programa de mejora de procesos de software basados en CMM, CMMI, PSP o RUP con conocimientos en programación.

Muestra cómo aplicar métodos avanzados de ingeniería a sus tareas diarias. Proporciona métodos detallados de planificación y estimación, muestra cómo controlar su rendimiento frente a estos planes.

PSP es un proceso predefinido con guías y plantillas diseñadas por el SEI que permite a los programadores adoptar un proceso completo de desarrollo de software. Permite a los desarrolladores adoptar las mejores prácticas para que a nivel personal establezcan planes de trabajos confiables y precisos. Permite incorporar las mejores prácticas para mejorar la calidad de los productos de software y minimizar el número de defectos en los programas que desarrolla un programador convirtiéndolos en desarrolladores confiables y de alta calidad [4].

Entonces, PSP es un proceso de aprendizaje intuitivo que se provee con la experiencia y los datos. Podría definirse también como un marco de trabajo diseñado para estimar y planificar un trabajo y de ésta forma mejorar la calidad de los programas.

[SM] Personal Software Process y PSP son marcas registradas de Carnegie Mellon University

2.1.-PRINCIPIOS PSP

- La calidad de un sistema software está condicionada por la calidad del peor de sus componentes.
- La calidad de un componente software está condicionada por el individuo que lo desarrolló.
- Está condicionada por el conocimiento, disciplina y compromiso.
- Como todo profesional de software el individuo conocerá su propio rendimiento.
- Deberá medir, seguir y analizar su trabajo.
- Deberá aprender de las variaciones de su rendimiento.
- Deberá incorporar dichas lecciones a su manera personal de hacer.

PSP incluye siete pasos compatibles y progresivos de análisis los cuales se muestran en figura 2.1, donde cada nivel se define de la siguiente manera:

- PSP0 – Se establece una línea de base de desempeño.
- PSP1 – Se hacen planes sobre tamaño, recursos y calendario.
- PSP2 – Se practica la administración de defectos y del *yield.*
- PSP3 – Se escalan los métodos de PSP a proyectos más grandes.

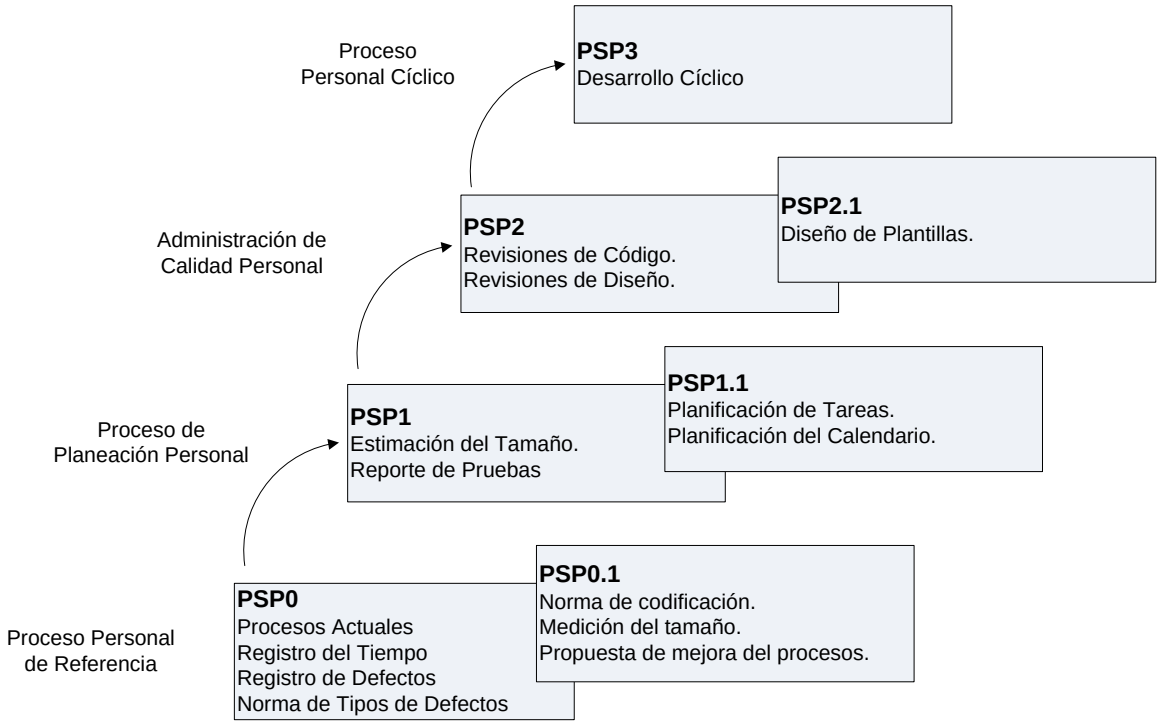


Figura 2.1.- Niveles de PSP

Los ingenieros de software deberán planificar su trabajo, y actuar de acuerdo con dicho plan, el resultado serán productos de alta calidad ajustados a un presupuesto y unos plazos. El Proceso de Software Personal (PSP) es un marco de trabajo diseñado para organizar la construcción de un software [3].

Un ejemplo de las actividades que indica la técnica de PSP es un primer ejercicio para identificar las tareas que realiza un estudiante. El estudiante deberá asistir a clase, escribir programas, leer libros de texto y hacer varios trabajos en casa. En algún momento deberá estudiar para los exámenes. Parte del trabajo en casa consistirá en escribir programas. Una forma de describir estas tareas se demuestra en la siguiente tabla.

Tareas	Frecuencia	Tiempo (Minutos)
Asistir a clase	Lunes ,Miércoles ,Viernes	180/Semana
Leer libros de texto	Semanal	180/semana
Trabajar en casa	Semanal	420/semana
Escribir programas	Semanal	420/semana
Preparar exámenes	Una vez al mes	300/semestre
Revisar apuntes	Durante el trabajo de casa	Incluido en otros tiempos.

Tabla 2.1.- Niveles de PSP

En este ejercicio se definirán las principales actividades del alumno y se escribirán en un formato como en la tabla anterior. Después de hacer esta lista, se estimará las tareas de dichas actividades conforme el tiempo en que le dedica a cada aspecto. Para todas las tareas semanales se debe estimar el tiempo que utilizará en cada semana, y para las tareas mensuales o semestrales estimar los tiempos mensuales y semestrales. Una vez estimado el tiempo se deberá cumplir al pie de la letra dicha planeación, por lo tanto, PSP no está tan lejos de lo que un ingeniero de software hace en su vida diaria.

PSP se basa en una serie de plantillas implementadas en cada etapa, dichas plantillas serán utilizadas por el individuo para registrar información acerca del seguimiento de su trabajo o proyecto, derivado a lo anterior, cada nivel de PSP incluye sus propias plantillas a utilizar, sin embargo, una vez concluido un nivel de PSP no se dejará en el olvido las plantillas del nivel ya terminado, nuevamente se recalca una de las características de PSP que dice “Incluye siete pasos compatibles y progresivos”, esto significa que las plantillas del primer nivel se implementarán de forma ajustada y adecuada al nivel posterior integrando nuevas plantillas a utilizar. Por lo anterior y para una mejor explicación, se muestra el “Mapa de Formas PSP” en la tabla 2.2, utilizadas en todos los niveles de PSP.

Formas, plantillas y estándares *Forma del Resumen del Plan de Proyecto*	**PSP0** *Ver0*	**PSP0.1** *Ver0.1*	**PSP1** *Ver1*	**PSP1.1** *Ver1.1*	**PSP2** *Ver2*	**PSP2.1** *Ver2.1*	**PSP3** *Ver3*
Bitácora de Registro de Tiempo	X	X	X	X	X	X	X
Bitácora de Registro de Defectos	X	X	X	X	X	X	X
Estándar de Tipos de Defectos	X	X	X	X	X	X	X
PIP (Process Improvement Proposal)		X	X	X	X	X	X
Estándar de Codificación		X	X	X	X	X	X
Estándar de Conteo de LOC		X	X	X	X	X	X
Plantilla de Reporte de Pruebas			X	X	X	X	X
Plantilla de Estimación de tamaño			X	X	X	X	X
Plantilla de Planeación de Tareas				X	X	X	X
Plantilla de Planeación de Tiempo				X	x	X	X
Lista de Chequeo de la Rev. de Diseño					X	X	X
Lista de Chequeo de la Rev. de Codificación					X	X	X
Plantilla del Escenario Operacional						X	X
Plantilla de la Especificación Funcional						X	X
Plantilla de la Especificación de Estados						X	X
Plantilla de Especificación de Lógica						X	X
Forma de Resumen del Ciclo							X
Bitácora de Seguimiento de Pendientes							X

Tabla 2.2.- Mapa de Formas PSP

2.2.- PSP0

El objetivo de PSP0, como etapa inicial de esta tecnología, es demostrar el uso de un proceso definido al escribir un programa pequeño, incorporar medidas básicas en el proceso de desarrollo de software, requerir cambios mínimos a las prácticas personales. En conclusión PSP0 provee una estructura bien definida para tareas pequeñas y la medición de éstas [3].

De esta forma los ingenieros registran cada dato que interviene en el desarrollo del programa, es decir, registran su tiempo en cada proceso y los datos de los defectos que se encuentran durante todo el proceso de desarrollo e inclusive desde la planeación. Al final del trabajo, durante la fase de postmortem, miden el tamaño del programa, e incorporan estos datos en el formato del resumen del proyecto. Al final se entrega el producto terminado junto con toda la documentación debidamente redactada, la cual son los formatos que utiliza PSP0.

El desarrollo de un producto de Software utilizando PSP0 está compuesto de una serie de fases o etapas, es simplemente el desarrollo de un plan bien estructurado representado en la fig. 2.2, siendo la definición de cada proceso la siguiente:

- Planeación: Estimar el tiempo desarrollado.
- Desarrollo: Desarrollar el producto utilizando sus métodos actuales.
    - Diseño: Diseñar el programa utilizando sus métodos de diseño actuales.
    - Codificación: construcción del Programa.
    - Compilación: compilar el programa hasta que esté sin errores.
    - Pruebas: Probar todo el programa y eliminar todos los defectos.
- PostMortem: Completar el resumen del Plan del Proyecto, con el tiempo utilizado y los defectos descubiertos e introducidos en cada fase.

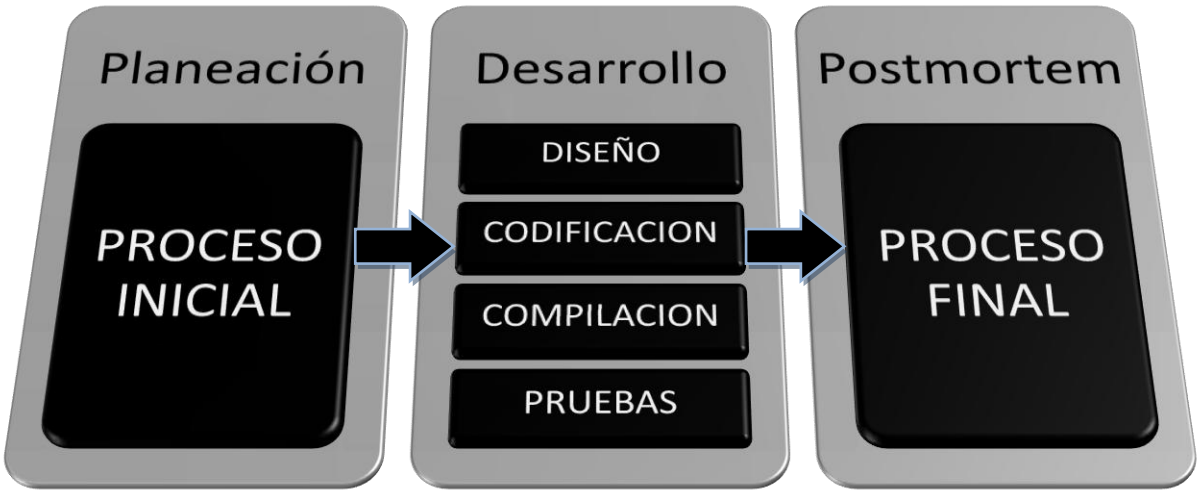


Figura 2.2.- El Proceso de PSP0

PSP0 es un proceso personal simple y definido que utiliza sus métodos actuales de diseño y desarrollo para generar datos acerca de su trabajo: tiempo utilizado por fase y defectos encontrados en la compilación y en las pruebas utilizando formatos de reporteo para la generación de resúmenes. Para el ejercicio del nivel de PSP0 se utilizarán elementos básicos o bien llamadas plantillas mencionadas en la siguiente lista:

- Script de Proceso (Process Script)
- Resumen del Plan del Proyecto (Proyect Plan Summary)
- Bitácora de Registros de Tiempo (Time Recording Log)
- Bitácora de Reporte de Defectos (Defect Recording Log)
    - Estándar de Tipos de Defectos (Defect Type Standard)

Ahora bien, ya mencionada la estructura general de PSP0 se presenta el flujo de trabajo para PSP0 mostrándose en la Fig. 2.3 Es importante resaltar que los formatos proporcionan las plantillas para almacenar los datos y los estándares ayudan a dirigir a los ingenieros durante todo el desarrollo del proyecto, desde el principio hasta el final de éste. Sin embargo, también hay que comentar que si el deseo del usuario es el de adaptar este nivel a los hábitos de programación personales, entonces éste tiene que modificar los scripts, formatos y registros según sus necesidades.

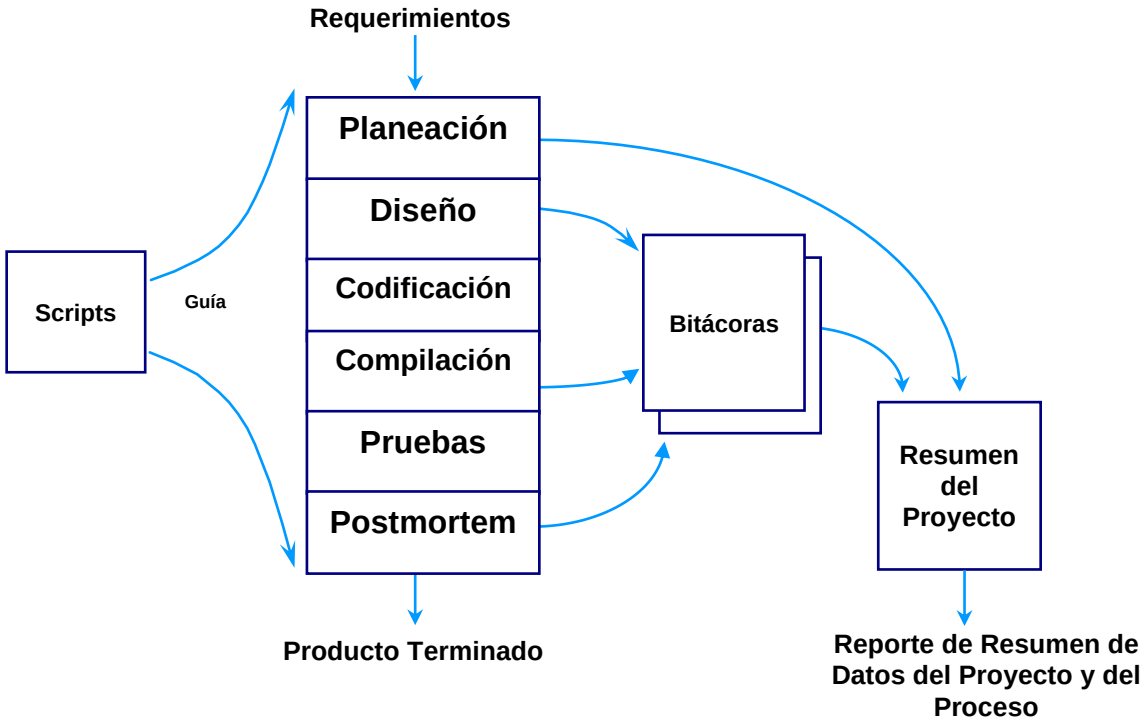


Figura 2.3.- El Proceso de PSP0

Descripción de cada fase del Proceso de PSP0

- ✓ **Planeación:** Primero obtén los requisitos del proyecto y completa las partes no sombreadas del Resumen del Plan del Proyecto. Finalmente escribe el tiempo dedicado a hacer esta planificación en la Bitácora de Registro del tiempo.
- ✓ **Diseño:** Diseña el programa, aunque el diseño necesario no se haga, piensa toda la lógica del programa antes de comenzar a escribir el código. Guarda el diseño en un diagrama de flujo, seudocódigo o en cualquier formato que especifique tu instructor. Al final de la fase de diseño, anota el tiempo de diseño en la Bitácora de Registro del Tiempo.
- ✓ **Codificación:** Implementa el diseño codificándolo en el lenguaje de programación seleccionado. Utiliza un formato de codificación consistente y sigue los estándares de codificación que especifique el instructor. Al final, anota el tiempo de codificación en el Cuaderno de Registro de Tiempos.
- ✓ **Compilación:** Compila el programa y corrige todos los defectos que encuentres. Continúa compilando y corrigiendo los defectos hasta que compiles el programa sin ningún mensaje de error. Todo el tiempo dedicado a ésta fase se contabiliza como tiempo de compilación, aunque corrijas el código o cambies el diseño. Al final de la fase de compilación, anota el tiempo de compilación en la Bitácora de Registro del Tiempo.
- ✓ **Pruebas:** Haz bastantes pruebas para asegurarte que los programas cumplen con todos los requisitos y superan un conjunto adecuado de pruebas sin error. Todo el tiempo que dediques a ésta fase se contabiliza como tiempo de prueba, incluyendo la corrección del código, el cambo de diseño y la re-compilación. Al final de ésta fase, anota el tiempo de pruebas en la Bitácora de Registro del Tiempo.
- ✓ **Postmortem:** Completa los datos reales en el formulario Resumen del Plan del Proyecto, tal y como se describe su instructivo de llenado. Puesto que has registrado el tiempo postmortem antes de que hayas acabado la fase postmortem completa el trabajo como puedas y entonces dedica unos minutos para hacer los cálculos finales y escribirlos en el tiempo postmortem estimado. Utiliza éste tiempo postmortem estimado para calcular el tiempo de desarrollo total y el resto de cálculos.

2.2.1.-EL SCRIPT DE PROCESO

Un Proceso es un conjunto definido de pasos para hacer un trabajo. Cada paso o fase de un trabajo tiene especificado unos criterios de entrada que deben ser satisfechos antes de comenzar la fase. De forma similar cada fase tiene unos criterios de salida que deben satisfacerse antes de terminar la fase. Los pasos del proceso definen las tareas y cómo se hacen. El diseño y gestión de procesos son importantes en ingeniería de software, porque la calidad del proceso del ingeniero, determina en gran parte, la calidad y productividad de su trabajo [4].

Para lograr establecer un buen proceso PSP0 contienen un documento, que tiene como nombre "Tabla C10 PSP0 Script de Proceso" en los anexos A. Es un conjunto de tareas que realizará el ingeniero el cual consta de las siguientes actividades:

- **Planeación:** Utilizado para estimar el tiempo de Desarrollo, representado en el Script de Planeación "Tabla C11 Script de Planeación".
- **Desarrollo:** Utilizado para desarrollar el producto utilizando sus métodos actuales, representado en el Script de Desarrollo "Tabla C12 Script de Desarrollo".
- **PostMortem:** Utilizado para completar el resumen del Plan del Proyecto, con el tiempo utilizado y los defectos descubiertos e introducidos en cada fase, representado en el Script de PostMortem "Tabla C13 Script de Postmortem".

Estas actividades tienen que ser implementadas según el orden mostrado en la figura 2.4.

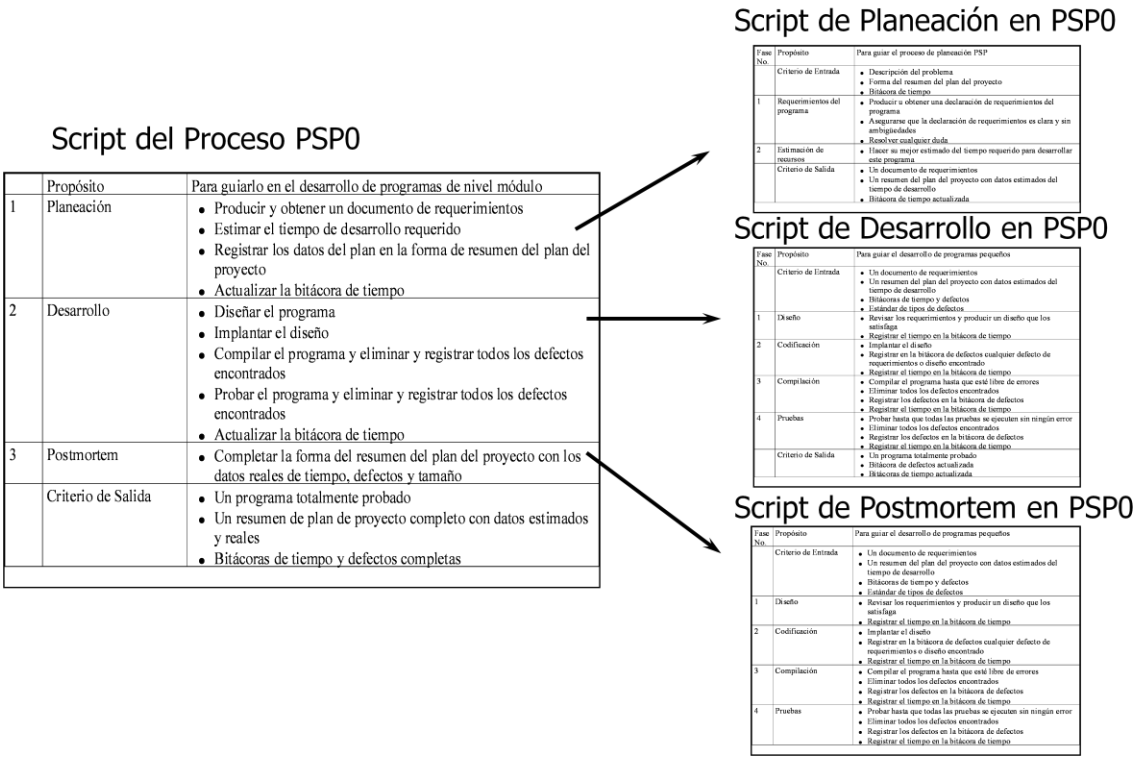


Figura 2.4.- Orden de Actividades PSP0

La exactitud, la estimación y el plan son influenciados en su mayoría por la experiencia que tienen los ingenieros según trabajos ya desarrollados. El diseño conceptual comienza cuando los ingenieros primero definen cómo el producto debe ser diseñado para que después éste sea construido, sin embargo, en esta etapa del proceso es muy temprano realizar un diseño del producto completo desde la fase del planeamiento [4].

Los desarrolladores tienen que estar conscientes de que deben realizar un diseño conceptual que se adapte lo más fácilmente posible al proyecto. Ésta conjetura es esencial porque cada vez que los ingenieros tengan dudas respecto al proyecto, bastaría mirar este diseño para tener bien fundamentadas las bases de lo que se está creando. Más adelante, durante la fase del diseño, los ingenieros examinan las alternativas del diseño y al final todas esas variantes finales producen lo que se conoce como el diseño del producto completo.

La estimación del tamaño y de los recursos del producto debe ser moderada por los equipos o individuos que intervienen en el desarrollo del proyecto. Sin embargo, para los ingenieros que desarrollan software de manera individual, ésta correlación tiene resultados generalmente altos.

En conclusión, PSP0 se encarga de estimar los tamaños del producto donde ésta información junto con sus datos son utilizados para estimar el tiempo requerido para hacer el trabajo, para que posteriormente se desarrolle el proyecto, y al final las pruebas necesarias para la entrega.

2.2.1.1.-El SCRIPT DE PLANEACIÓN

La planificación de una parte crítica del trabajo del ingeniero de software, y por lo tanto para ser un buen ingeniero de software hay que saber hacer una buena planeación. La clave está en la práctica, para adquirirla comienza haciendo planes de inmediato y continuar haciéndolos en todos los proyectos futuros a desarrollar [4].

Cuando un ingeniero trabaja en equipos de desarrollo, necesita planificar su trabajo personal, La planificación proporciona una sólida base para comprometerse a unas fechas de entrega y permite a los ingenieros coordinar sus trabajos con los productos. Los planes individuales del producto, le permite cumplir las fechas para cada una de sus tareas interdependientes y cumplir sus compromisos de forma consistente [4].

El ingeniero también utiliza los planes del producto para entender el estado de su proyecto. Con planes razonables detallados y precisos que pueden juzgar dónde se encuentra el proyecto frente al plan. Puede ser, que esté retrasado y necesite ayuda, o que tenga que retrasar la programación. Pero también puede ir más adelantado con respecto a lo planificado, y ser capaz de ayudar a un compañero o entregar antes. Cuando los ingenieros hacen planes pueden organizar mejor su tiempo y evitar crisis en los últimos minutos. Entonces es menos probable que cometan errores y generalmente producirán mejores resultados [4].

Puesto que la planificación es tan importante, es necesario conocer cómo hacer planes precisos. Y también, saber compararlos con los resultados reales, con el propósito de mejorar una planeación [4]. Para ello PSP0 contiene bien establecida una serie de pasos para efectuar actividades de planeación el cual se reflejan en el primer script de PSP0, el *Script de Planeación*.

El script de planeación tiene como nombre “Tabla C11 Script de Planeación” en los anexos y cuenta como entrada con los requerimientos del programa así como la estimación de los recursos que se emplean en éste, específicamente son los siguientes:

- ✓ Descripción del Problema
- ✓ Formulario Resumen del Plan del Proyecto
- ✓ Bitácora de Registro del Tiempo

Su objetivo es recopilar información de la fase de Requerimientos del Programa y Estimación de Recursos de una forma clara y sin ambigüedades, registrando en todo momento el tiempo utilizado en la Bitácora del Registro del Tiempo.

Tiene como salida información que ayudará a comenzar a desarrollar al trabajo, utilizando como apoyo los siguientes documentos:

- ✓ Un Documento de Requerimientos (no tiene un formato en específico)
- ✓ Un resumen del Plan del Proyecto con datos estimados del tiempo de desarrollo (Anexos A: Tabla C14)
- ✓ Bitácora de Registro de Tiempo (Anexos A: Tabla C16) elaborada.

2.2.1.2.-El SCRIPT DE DESARROLLO

El script de desarrollo tiene como nombre "Tabla C12 Script de Desarrollo" en los anexos y exige al programador los requerimientos de cada fase de desarrollo que toma en cuenta PSP (Diseño, Codificación, Compilación y Pruebas) para que al final se cuente con un programa bien probado y a prueba de errores [4].

El scripts de desarrollo tiene como entrada:

- ✓ Documento de Requerimientos de la fase anterior
- ✓ El Resumen del plan del proyecto (Anexos A: Tabla C14) con datos estimados del tiempo de desarrollo
- ✓ La Bitácora de Registro de Tiempo (Anexos A: Tabla C16)
- ✓ La Bitácora de Defectos (Anexos A: Tabla C18) apoyado del Estándar de Tipos de Defectos (Anexos A: Tabla C20).

Su objetivo es la especificación de las fases de Diseño, Codificación, Compilación y Pruebas, y registrar el tiempo invertido y defectos encontrados durante la Compilación.

Tiene como salida información del producto terminado utilizando lo siguiente:

- ✓ Un Programa totalmente probado
- ✓ La Bitácora de Defectos
- ✓ La Bitácora de Registro del Tiempo elaborado de cada fase.

2.2.1.3.-EL SCRIPT DE POSTMORTEM

El script de Postmortem tiene como nombre "Tabla C13 Script de Postmortem" en los anexos y pide al programador todos los defectos que se encontraron durante la realización del proyecto y también exige el tiempo final que se utilizó durante la realización del programa, esto para que exista un historial final que ayude a tener un margen de comparación para utilizarlo posteriormente y evitar caer en los mismos errores.

El script de postmortem tiene como entrada la siguiente información:

- ✓ El documento con la descripción del problema

- ✓ La especificación de requerimientos
- ✓ El resumen del plan del proyecto elaborado (Tabla C14)
- ✓ Las Bitácoras de Registro del Tiempo (Tabla C16) de cada fase
- ✓ Bitácora de Defectos (Tabla C18)
- ✓ El programa probado y en funcionamiento.

Tiene como objetivo crear un concentrado de todo lo contabilizado para obtener el dato del tiempo total invertido, número de defectos encontrados durante la compilación y escribirlos en la documentación pertinente.

Tiene como salida:

- ✓ El programa plenamente probado
- ✓ El Resumen del Plan de Proyectos elaborado
- ✓ La Bitácora de Defectos
- ✓ La Bitácora de Registro del Tiempo de ésta fase.

2.2.2.- EL RESUMEN DEL PLAN DEL PROYECTO

En el formato de la Tabla C14 del Anexo se muestra el resumen del plan del proyecto, este formato reúne las estimaciones y los datos reales que conforman al proyecto en toda su amplitud para que al final se realicen las comparaciones necesarias y exista un histórico de todos los proyectos realizados.

Como se puede apreciar en este formato, existen tres campos diferentes. Dos de estos campos tienen que ver con los defectos encontrados y removidos en cada fase. En la parte superior del formato se aprecian los campos que el desarrollador empleará para registrar los tiempos que emplea en cada fase del proyecto.

Este formato es esencial ya que es un respaldo para cada proyecto que se desarrolla. En él se pueden encontrar los datos que serán útiles para el siguiente proyecto parecido que se desarrolle. Es importante que los datos se escriban con claridad y con precisión para que cada fase de desarrollo sirva para tener un margen de comparación con proyectos futuros.

Los campos que el programador tiene que llenar en este formato son:

- ✓ Area: Tiempo en Fase.
  - o Plan.- Es el tiempo estimado que se empleará para desarrollar el proyecto completo.
  - o Real.- Es el tiempo real, en minutos, que se emplea en cada una de las fases de desarrollo.
  - o A la fecha.- La suma del tiempo actual con el tiempo a la fecha del último programa desarrollado.
  - o A la fecha %.- Indica el porcentaje del tiempo a la fecha que se emplea en cada fase de desarrollo.
- ✓ Área de Defectos Introducidos.

- Real.- Número de defectos reales encontrados en cada fase de desarrollo.
- A la fecha.- Suma de los valores de los campos Actual (Defectos encontrados) con el campo de A la fecha del último proyecto programado.
- A la fecha %.- Porcentaje de defectos encontrados a la fecha en cada fase de desarrollo.

✓ Área de Defectos Eliminados.

- Real.- Indica el número de defectos removidos en cada etapa.
- A la fecha.- Es la suma del valor que se encuentra en el campo de Actual (Defectos removidos) con el valor A la fecha del último programa desarrollado.
- A la fecha %.- Porcentaje de defectos removidos del campo A la fecha y que se aplica para cada fase de desarrollo.

Cuando el proyecto se termine es importante registrar cualquier defecto que se haya detectado tardíamente. Esto debe ser realizado cuando el programa esté en uso, re-uso o si el programa sufre alguna modificación.

2.2.3.- BITÁCORA DE REGISTRO DEL TIEMPO

A continuación se describe el formato utilizado en PSP 0 el cual se basa por lo general en dos medidas importantes: El tiempo empleado en cada fase y los defectos encontrados en cada fase, estos datos se calculan de manera muy específica.

El formato de la Tabla C16 en el Anexo A, , es el formato del registro de tiempo y que contiene diversos campos, aunque conforme se avance de nivel, se van agregando más campos y demandas.

El contenido de este formato marca el principio del proceso PSP y por lo tanto es parte del nivel inicial de PSP0. El propósito de éste formato es el de registrar el tiempo empleado en cada fase del proyecto. Al mismo tiempo, estos datos son utilizados para complementar el resumen del plan del proyecto.

Como información general lo único que se necesita es registrar el tiempo total que se emplea en el proyecto; este tiempo debe estar registrado en minutos y por último se debe procurar ser lo más preciso posible.

Los campos que forman parte de PSP 0 y de éste formato son:

- ✓ Encabezado.- Los datos que se deben agregar a este campo son el nombre del desarrollador, la fecha actual, el nombre del supervisor o instructor y el número que le corresponde al programa que se está desarrollando.
- ✓ Fecha.- Corresponde a la fecha que tiene que ver con el dato que se introduce.
- ✓ Inicio.- La hora cuando se comienza a trabajar en el proyecto.
- ✓ Término.- La hora cuando se deja de trabajar en el proyecto.

- ✓ Tiempo de interrupción.- Aquí se registran todas las interrupciones que se llevaron a cabo durante el proyecto, es decir, el tiempo que no se emplea en trabajar en el proyecto.
- ✓ Tiempo delta.- Es el tiempo que se obtiene de la resta del tiempo empleado en el proyecto menos el tiempo de interrupción.
- ✓ Fase.- Aquí se introduce el nombre de la fase en la que se trabaja, puede variar, no tiene que ser la misma que en la que se está trabajando desde un principio.
- ✓ Comentarios.- Se tiene que procurar hacer todo tipo de comentarios útiles que puedan recordar ciertas circunstancias.

Todos los tiempos llevan el formato de horas y minutos (HH:MM), en caso de que sólo sean minutos los estudiante entonces se utilizarán el formato (:MM).

Conjuntamente a este formato se debe trabajar con el formato de registro de defectos que se muestra en la Tabla C18 del Anexo y que como se menciona, es una fuente importante de datos que nos ayudará posteriormente a estimar de mejor manera posible el tiempo del proyecto y a evitar errores que se cometen con frecuencia.

2.2.4.- BITÁCORA DE REGISTRO DE DEFECTOS

El propósito general de llevar este registro de defectos reside en promover la mejora contínua cada vez que se haga un proyecto. Cada fase de PSP debe de contar con un registro de defectos, ya sean revisiones, compilaciones y/o pruebas.

Los campos que intervienen en este formato son:

- ✓ Fecha.- Se introduce la fecha cuando se encuentra el defecto.
- ✓ Número.- Se introduce de manera secuencial el número de defecto encontrado en cada programa.
- ✓ Tipo.- Se refiere al tipo de defecto que viene en la tabla adjunta al formato, el tipo de defecto va desde 10 hasta 100 en intervalos de 10 en 10.
- ✓ Encontrado.- Por medio del mejor juicio se introduce el nombre de la fase cuando se encontró el defecto.
- ✓ Removido.- Se refiere al nombre de la fase cuando se removió el defecto encontrado.
- ✓ Tiempo de compostura.- Es el tiempo que tomó reparar el defecto encontrado.
- ✓ Defecto arreglado.- Este campo quiere decir si se encontró algún defecto extra mientras se reparaba el defecto detectado primero, en caso de no haber ninguno, se introduce una X.

Cada vez que se encuentra un defecto, se debe mantener un registro de estos errores ya que para proyectos futuros es importante evitar repetir o caer en los mismos errores. De esta manera se evita que resulte infructuoso utilizar PSP para mejorar constantemente porque el propósito es el de apoyar a los ingenieros a estar innovando sus métodos de desarrollo. Finalmente se presenta el formato más importante de todos y que es el más popular de PSP.

2.2.4.1.- ESTÁNDAR DE TIPOS DE DEFECTOS

El estándar de tipos de defectos proporciona un conjunto general de categorías de defectos.

Aunque usted puede reemplazar este estándar con el suyo propio, es conveniente apegarse a las definiciones simples hasta que se tengan datos para realizar su cambio de tipos de defectos.

CAPITULO 3. ANÁLISIS Y DISEÑO

3.1.- PLANTEAMIENTO DEL PROBLEMA

En clases impartidas, en la Benemérita Universidad Autónoma de Puebla, en las carreras de ingeniería en sistemas computacionales, ciencias de la computación o afines, es necesario utilizar un mecanismo o técnica de enseñanza-aprendizaje más especializada para los alumnos de primer semestre, ya que al principio nos cuesta comprender la lógica de programación estructurada y la orientada a objetos.

Por lo anterior, comprender la sintaxis de los lenguajes de programación es difícil, y no hay nada que la práctica pueda lograr, al invertir el tiempo necesario para dicho aprendizaje no observamos que a veces se invierte mal el tiempo realizando otras actividades.

Para dar solución a ésta necesidad, el presente trabajo tiene como objetivo el desarrollo de un sistema utilizando una técnica denominado PSP y específicamente el nivel 0, que ayudará a los estudiantes de las carreras antes mencionadas a dar seguimiento a su aprendizaje mediante ciertas actividades programadas en el uso de ésta técnica, elaborando registros que le ayudarán al estudiante a observar sus debilidades o fortalezas, logrando mejorar su desempeño en el desarrollo de software.

3.2.- ESPECIFICACIÓN DE REQUERIMIENTOS

Para el desarrollo del sistema para el nivel 0 de PSP es necesario seguir al pie de la letra las actividades que indica todo el proceso de la técnica PSP nivel 0.

En la fase de Planeación, es necesario contar con el nombre del proyecto a desarrollar, un documento con la definición del problema y la especificación de requerimiento, esto le ayudará al estudiante a no salir del objetivo general y de los particulares del sistema a desarrollar.

En la fase de Desarrollo, se debe incluir información acerca del registro de defectos, para ello es necesario contar con una herramienta, incluido en el sistema de PSP0, que almacene todos los defectos encontrados durante la fase de compilación del sistema a desarrollar. Durante la fase de Desarrollo se debe ir llenando un registro denominado Formulario de Registro del Tiempo, éste es necesario para registrar cualquier actividad que ayude o afecte el desarrollo del sistema, por ejemplo: se debe registrar, en minutos, desde 4:12 hasta 19:01 realicé el análisis de la base de datos, pero de 19:01 a las 20:30 estuve viendo una película. Esta actividad puede proporcionar datos más precisos acerca de la inversión de tiempo durante el desarrollo de un sistema de software. Este mecanismo debe ser incluido en el sistema del nivel 0 de PSP, a grandes rasgos, se necesita un botón que sirva como cronómetro pero que tenga las herramientas necesarias para que al estar oscilando del estado Iniciar al estado Detenido, registre la información necesaria de lo acontecido.

En la fase de Postmortem, se conjunta toda la información recabada para crear un resumen de datos estadísticos listos para ser analizados, esto por supuesto cuando el sistema esté terminado (durante la fase de Desarrollo), como por ejemplo registrar el número de defectos encontrados y eliminados, se revisa la bitácora del tiempo para observar el tiempo real invertido durante el desarrollo del sistema, así como el tiempo perdido utilizado para otras actividades ajenas a dicho trabajo de desarrollo.

El sistema debe tener soporte para varios usuarios, esto es, que un grupo de estudiantes participarán en el uso del sistema, cada estudiante deberá tener su propia cuenta en el sistema puesto que le dará el seguimiento a su propio trabajo.

También existe la certeza de que más de un instructor imparta su materia utilizando la técnica PSP por lo que, de forma similar que el estudiante, deberá tener su propia cuenta, pero éste podrá supervisar el desarrollo de los estudiantes.

Como tal, deberá existir un administrador que registre los estudiantes e instructores, así como proporcionar a cada instructor no más grupos de estudiantes participantes. Por lo tanto deberá existir seguridad al momento de que cada rol de usuario quiera acceder al sistema.

Para preservar la información y prevenir cualquier falla, el sistema deberá tener un mecanismo para generar un respaldo de toda la información ya introducida en el sistema.

El sistema requiere la administración de Talleres PSP, los cuales se podrán aperturar en los periodos programados para la inscripción de los estudiantes.

Deberá existir un mecanismo para administrar los estados de los alumnos: Inscripción, Baja Temporal, Baja Definitiva, Egresado, según el acontecer de la situación de cada estudiante.

El sistema deberá mostrar, utilizando la información generada en PSP0, estadísticas en forma gráfica acerca del desempeño de los estudiantes en el desarrollo de software, publicando sus debilidades y fortalezas para el mejoramiento de su nivel de desarrollo.

3.3.- DIAGRAMA DE FLUJO DE DATOS

Una vez terminados los modelos: Funcional y Entidad-Relación, se prepara la fase analítica final. Construir el Diagrama de Flujo de Datos. Este último encuaderna las funciones del negocio y las titula, permite una representación gráfica de sus interrelaciones, muestra el flujo de entradas y salidas de datos a través del negocio total. El modelo de flujo de datos, finalmente sirve como una plataforma para fundamentar la aplicación, como ayuda en el diseño de bases de datos y el diseño de las plantillas de consulta y/o captura para el diseño de los reportes.

Definición. Ilustra las funciones que el sistema debe realizar. Podría describirse como ¿qué transformaciones debe llevar a cabo el sistema? ¿Qué entradas se Transforman en qué salidas? Entre otras.

Los diagramas de flujo de datos contienen los siguientes elementos:

Entidad Externa	✓ Representa personas, organizaciones, o sistemas que no pertenecen al sistema. ✓ En el caso de que las entidades externas se comunicasen entre sí, esto no se contemplaría en el diagrama, por estar fuera del ámbito de nuestro sistema. ✓ Puede aparecer en los distintos niveles de DFD para mejorar su comprensión, aunque normalmente sólo aparecerá en el diagrama de contexto. ✓ Pueden aparecer varias veces en un mismo diagrama, para evitar entrecruzamientos de líneas. ✓ Suministra información acerca de la conexión del sistema con el mundo exterior.
Proceso	✓ Cuando un flujo de datos entra en un proceso sufre una transformación. ✓ Un proceso no es origen ni final de los datos, sólo lugar de transformación de ellos. ✓ Un proceso puede trasformar un dato en varios. ✓ Es necesario un proceso entre una Entidad Externa y un Almacén de datos. ✓ Un proceso puede representarse señalando una localización. La localización expresa la unidad o área dentro de la organización donde se realiza el proceso.
Flujo de Datos	✓ El concepto de flujo de datos es similar al concepto de tubería a través del cual fluye información de estructura conocida. ✓ Los datos no pueden ser creados ni destruidos por un flujo de datos. ✓ Sirve para conectar el resto de los componentes de un DFD. ✓ No es un activador de procesos. ✓ Cuando un proceso almacena datos, la flecha de flujo de datos se indica en la dirección del almacén de datos y a la inversa si es el proceso el que lee datos en el almacén.
Almacén de Datos	✓ Representa la información en reposo. ✓ No puede crear, destruir ni transformar datos. ✓ No puede estar comunicado directamente con otro almacén o Entidad externa. ✓ El flujo de datos (Entrada y Salida) no lleva nombre cuando incide sobre su contenido completo. ✓ No debe estar referido al entorno físico, y por tanto, no se diferencian los ficheros convencionales de las bases de datos. ✓ No se representa la clave de acceso a este almacén sino sólo la operación que se realiza (lectura, escritura, actualización)

DESCOMPOSICIÓN POR NIVELES

- ✓ El sistema deberá contener:
  - Un Diagrama de contexto (primer nivel)
  - Varios DFD en niveles intermedios
  - Varios DFD en el último nivel de detalle
- ✓ En cualquier momento nos puede aparecer un proceso que no necesite descomposición y es lo que denominaremos Proceso Primitivo (PP). En ellos, se detallará la entrada y salida que tenga, además de la descripción asociada que explique lo que realiza.

CONSTRUCCIÓN

- ✓ Representar el diagrama de contexto.
- ✓ Representar el DFD de primer nivel, indicando los distintos subsistemas funcionales en que se descompone nuestro sistema.
- ✓ Descomponer cada uno de los procesos que aparecen en el DFD de primer nivel, hasta llegar a un nivel suficiente de detalle.
- ✓ Se recomienda el utilizar cuatro niveles de descomposición de diagramas.
  - Nivel 0: Diagrama de contexto.
  - Nivel 1: Subsistemas.
  - Nivel 2: Funciones de cada subsistema.
  - Nivel 3: Subfunciones asociadas.
  - Nivel 4: Procesos necesarios para el tratamiento de cada subfunción.

EJEMPLO:

DIAGRAMA DE CONTEXTO PROCESO MATRICULA (NIVEL 0)

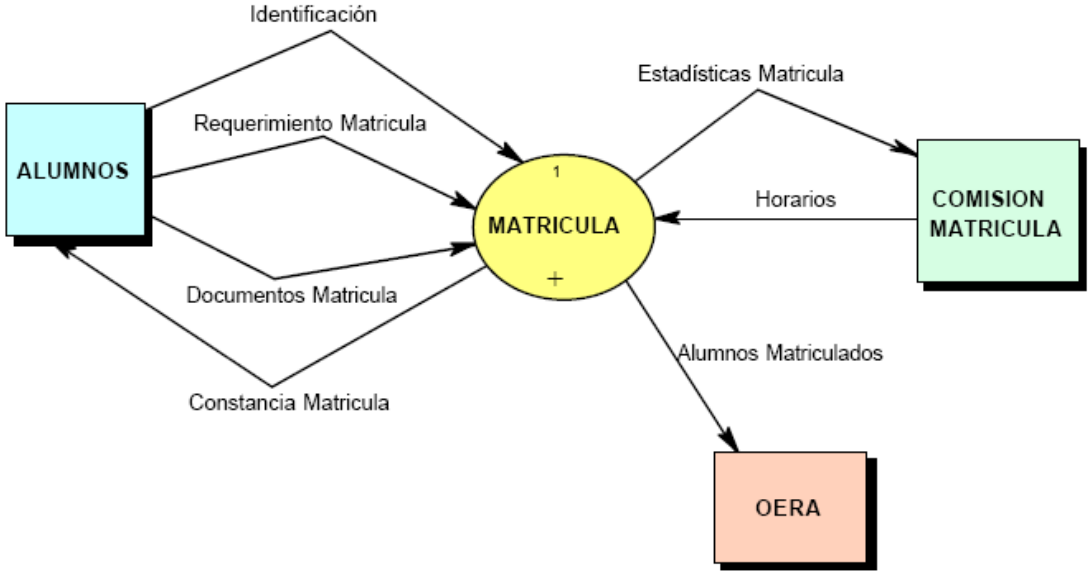
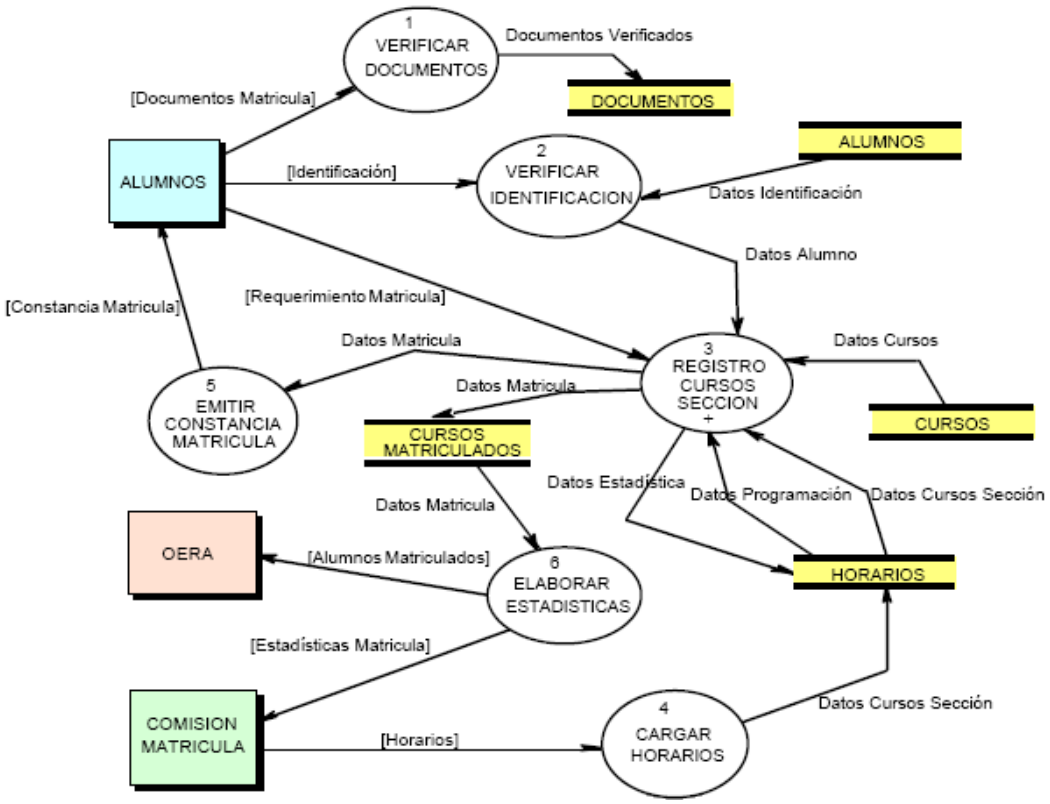
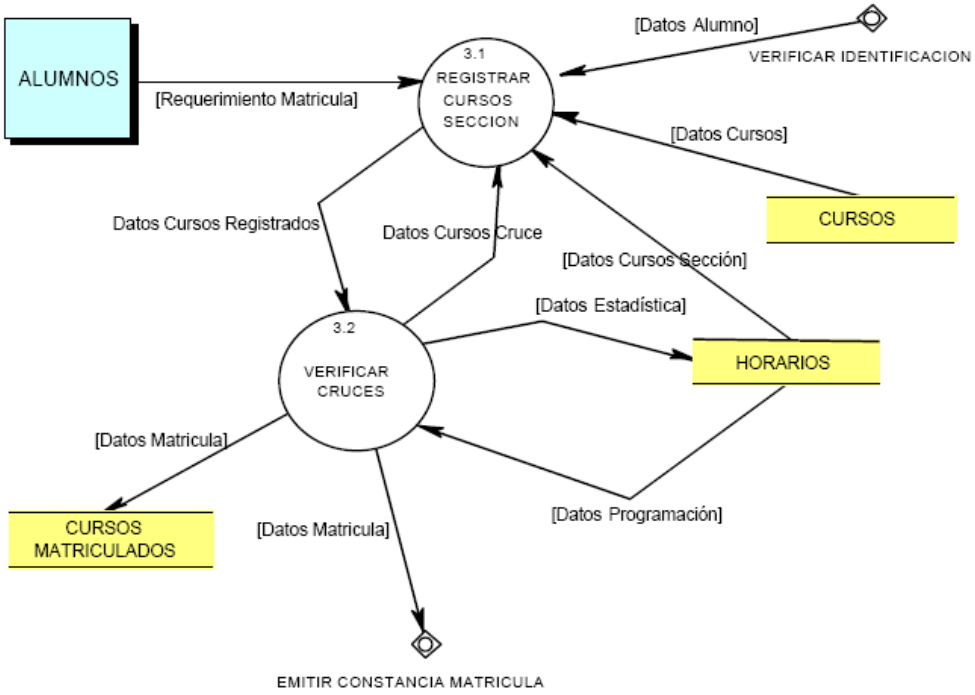


DIAGRAMA DE PRIMER NIVEL



SEGUNDO NIVEL



CAPITULO 4. DISEÑO DE LA BASE DE DATOS

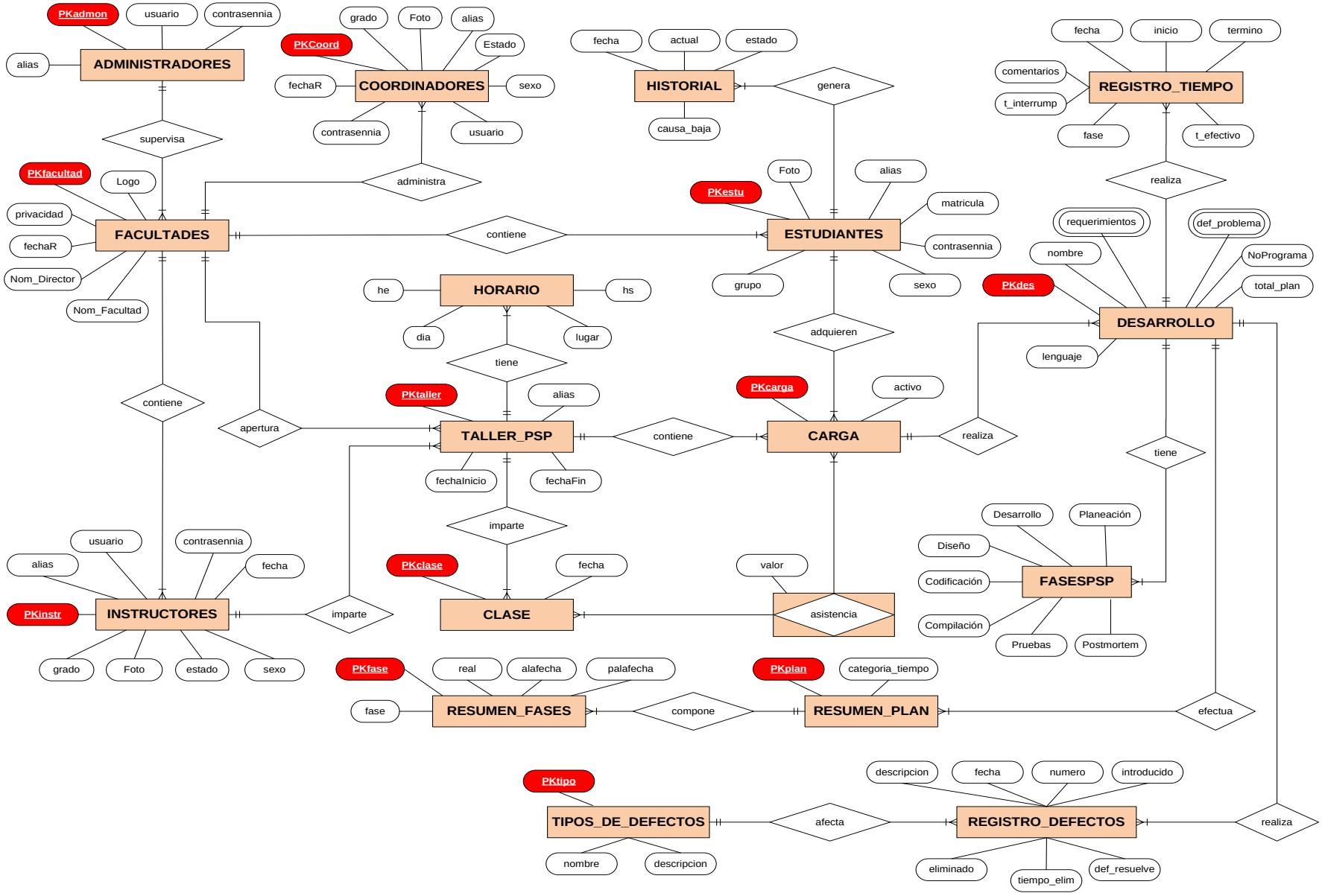
En el presente capítulo se muestra el modelo Entidad-Relación del Sistema de Formación para el Desarrollo de Software con PSP0, indicando con detalle el flujo de la información de datos y de las relaciones que existen entre las entidades.

Inicialmente, como se muestra en el modelo entidad-relación, la base de datos cuenta con una parte para almacenar los usuarios para la administración de roles, los talleres PSP que se aperturen durante los periodos indicados. La supervisión del desarrollo y desempeño de cada estudiante y el cumplimiento de las actividades de PSP nivel 0.

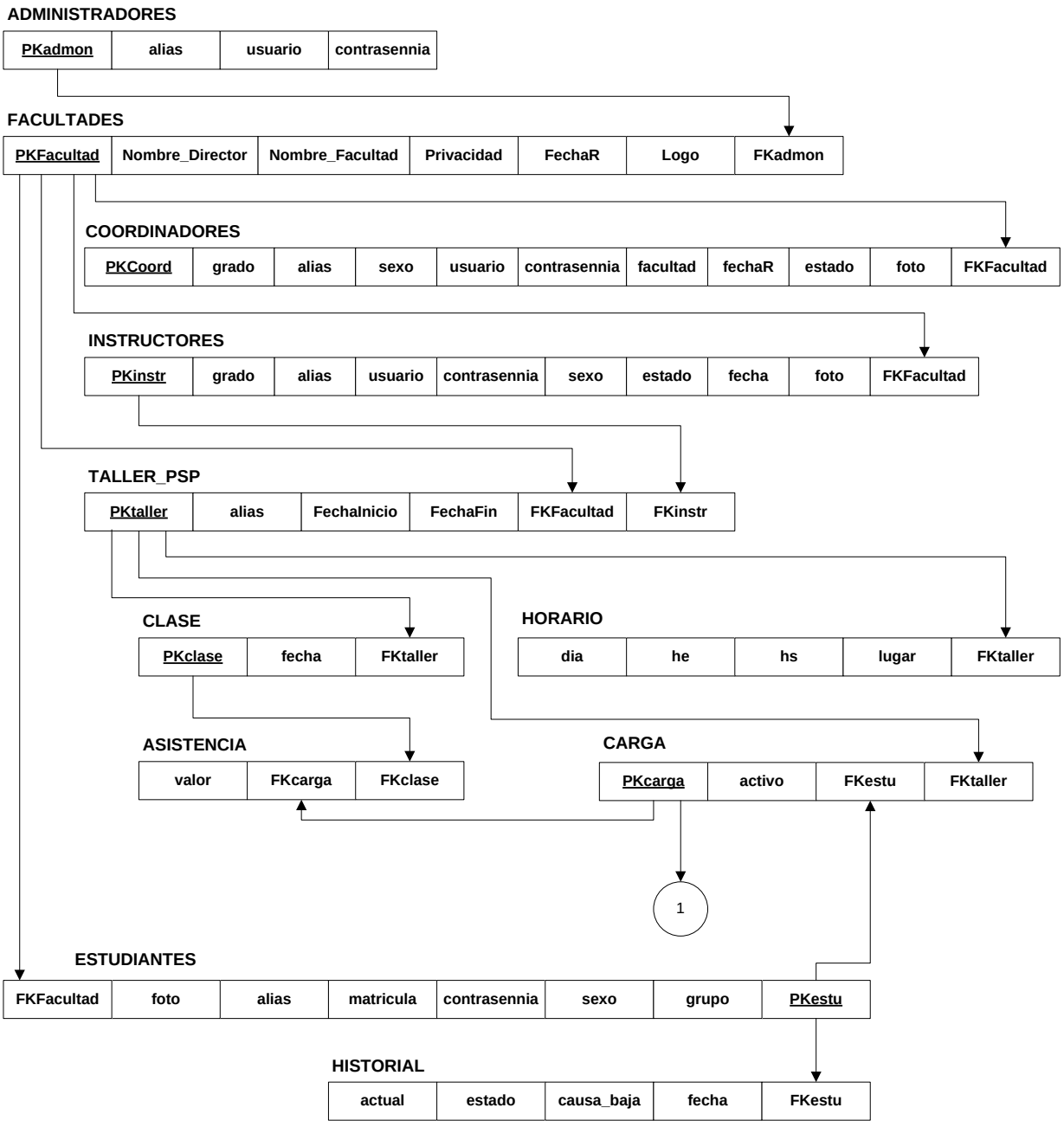
El siguiente diagrama muestra el proceso general del funcionamiento del Sistema de Formación para el Desarrollo de Software PSP0

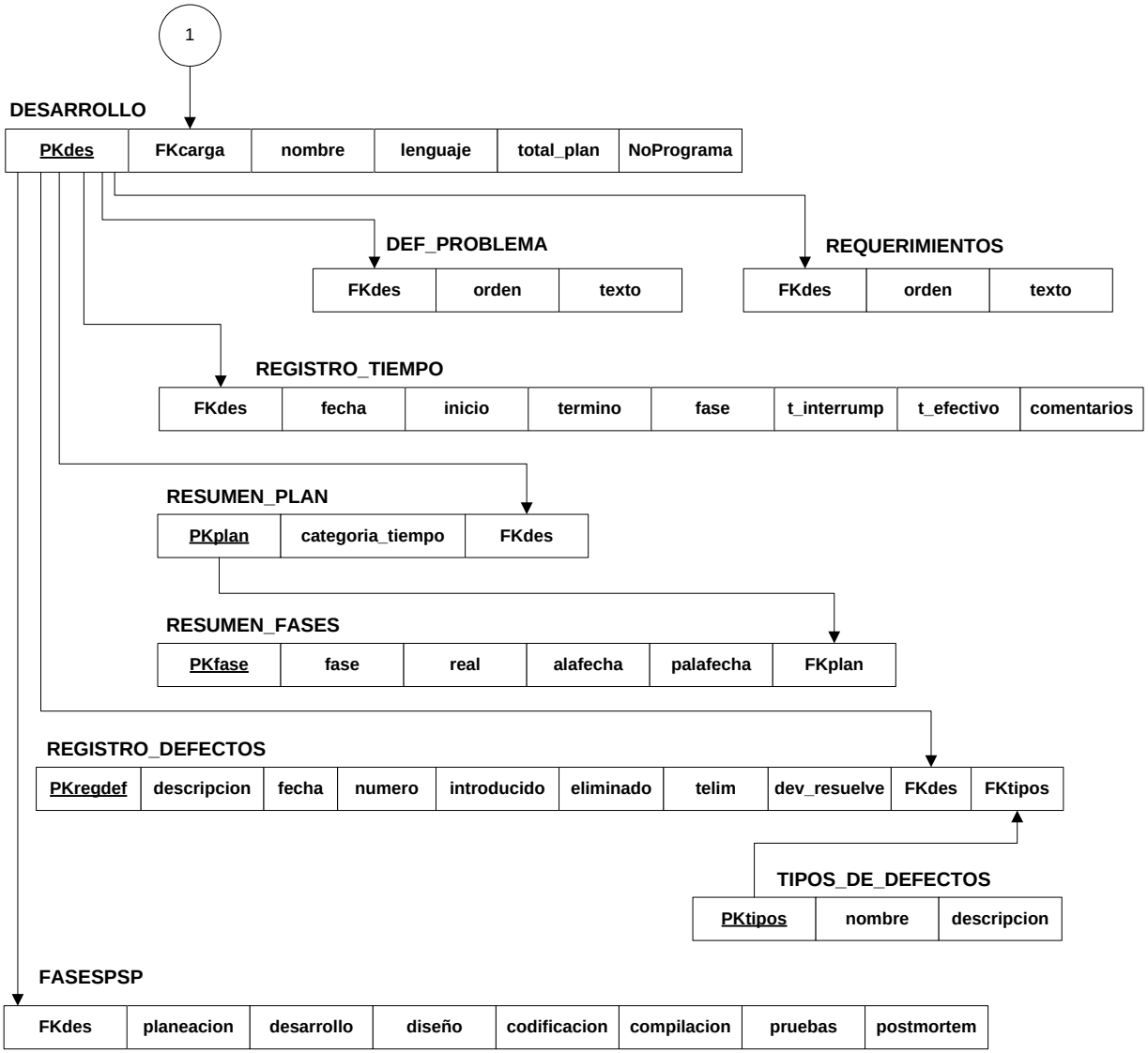
Figura 4.1.- Proceso General del Sistema de Base para el nivel 0 de PSP

4.1.- MODELO ENTIDAD RELACIÓN



4.2.- MODELO RELACIONAL




1
DESARROLLO
PKdes
FKcarga
nombre
lenguaje
total_plan
NoPrograma
DEF_PROBLEMA
FKdes
orden
texto
REQUERIMIENTOS
FKdes
orden
texto
REGISTRO_TIEMPO
FKdes
fecha
inicio
termino
fase
t_interrump
t_efectivo
comentarios
RESUMEN_PLAN
PKplan
categoria_tiempo
FKdes
RESUMEN_FASES
PKfase
fase
real
alafecha
palafecha
FKplan
REGISTRO_DEFECTOS
PKregdef
descripcion
fecha
numero
introducido
eliminado
telim
dev_resuelve
FKdes
FKtipos
TIPOS_DE_DEFECTOS
PKtipos
nombre
descripcion
FASESPSP
FKdes
planeacion
desarrollo
diseño
codificacion
compilacion
pruebas
postmortem

4.3.- NORMALIZACIÓN DE LAS TABLAS

Para recordar, la normalización es el proceso total que puede expresarse de manera informal como un conjunto de reglas a aplicar para reducir de forma sistemática una relación a un conjunto de relaciones más pequeñas, sencillas, estables y que son equivalentes a la relación original. Además las estructuras de datos normalizadas son más fáciles de mantener.

Aplicando el proceso de la normalización a las entidades del modelo Entidad-Relación se muestran los siguientes cambios:

----------TABLA 1----------

FACULTADES

PKFacultad	Nombre_Director	Nombre_Facultad	Privacidad	FechaR	Logo	FKadmon

Aplicando 1FN para la entidad facultad, quedando de la siguiente forma:

FACULTADES

PKFacultad	Nombre_Director	Nombre_Facultad	Privacidad	FechaR	FKadmon

LOGO_FACULTADES

FKFacultad	logo	orden

----------TABLA 2----------

COORDINADORES

PKCoord	grado	alias	sexo	usuario	contrasennia	facultad	fechaR	estado	foto	FKFacultad

Aplicando 1FN para la entidad facultad, quedando de la siguiente forma:

COORDINADORES

PKCoord	grado	alias	sexo	usuario	contrasennia	facultad	fechaR	estado	FKFacultad

FOTOS_COORDINADORES

FKCoord	logo	orden

----------TABLA 3----------

INSTRUCTORES

PKinstr	grado	alias	usuario	contrasennia	sexo	estado	fecha	foto	FKFacultad

Aplicando 1FN para la entidad facultad, quedando de la siguiente forma:

INSTRUCTORES

PKinstr	grado	alias	usuario	contrasennia	sexo	estado	fecha	FKFacultad

FOTOS_INSTRUCTORES

FKInstr	logo	orden

----------TABLA 4----------

ESTUDIANTES

FKFacultad	foto	alias	matricula	contrasennia	sexo	grupo	PKestu

Aplicando 1FN para la entidad fase, quedando de la siguiente forma:

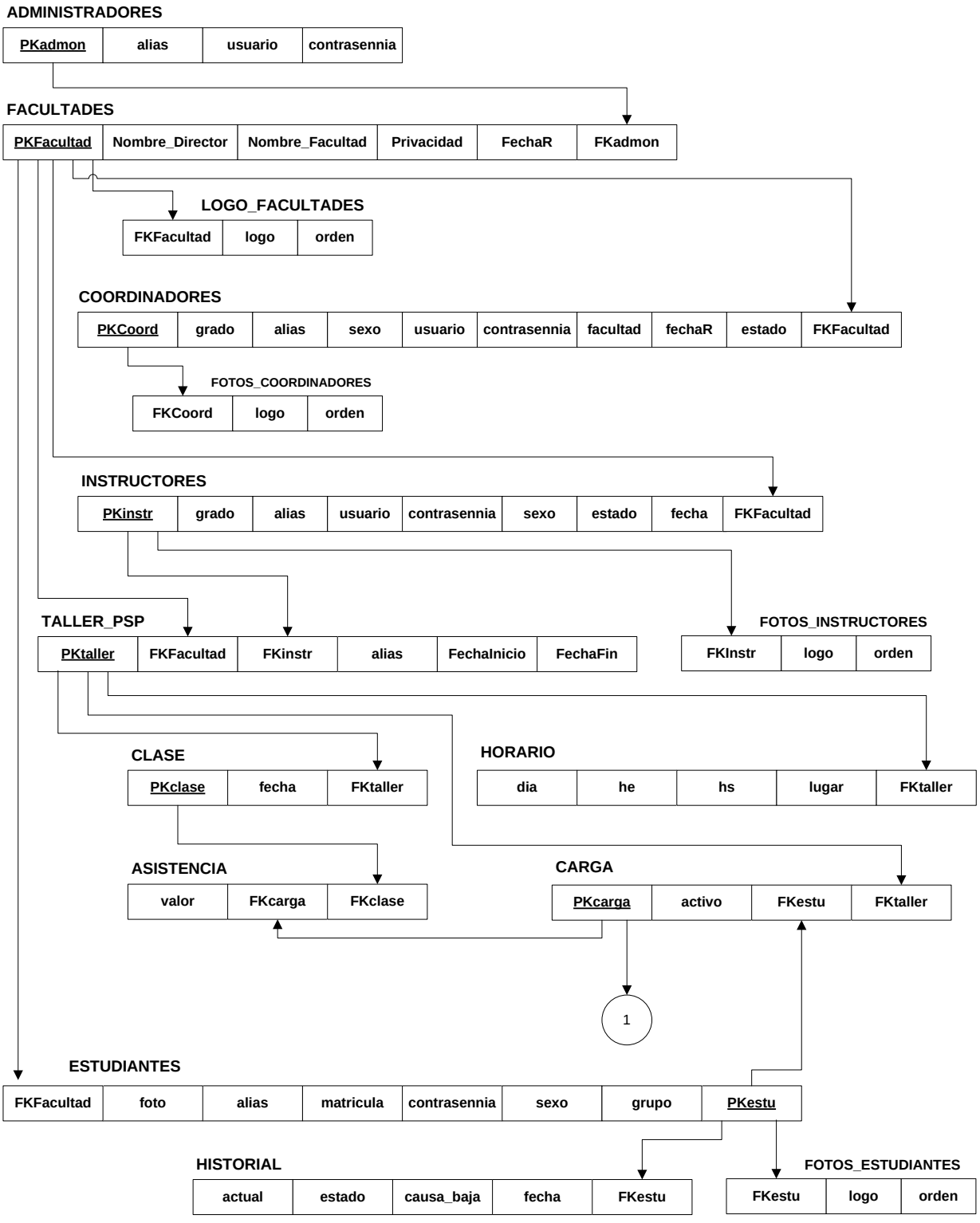
ESTUDIANTES

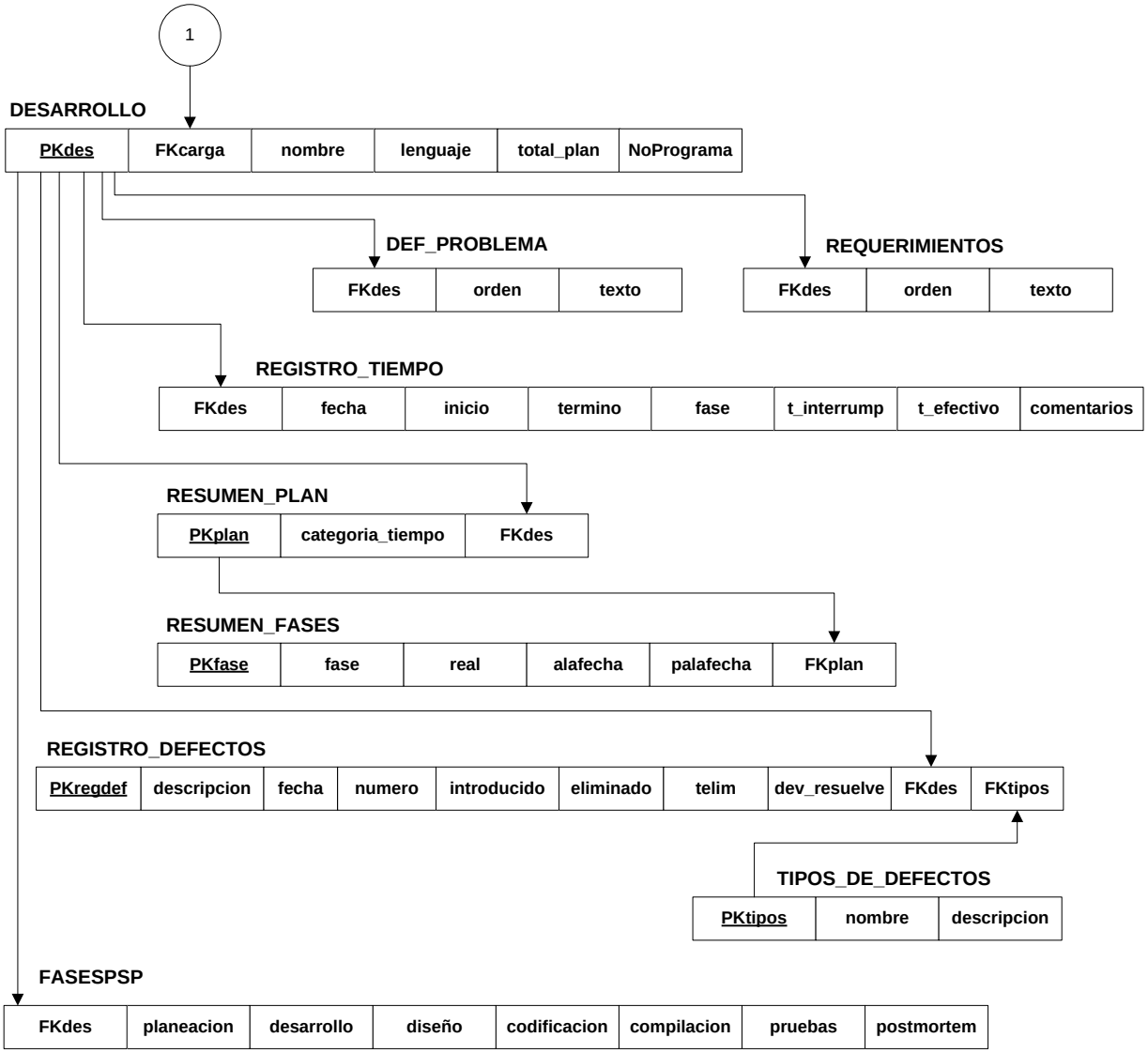
FKFacultad	foto	alias	matricula	contrasennia	sexo	grupo	PKestu

FOTOS_ESTUDIANTES

FKestu	logo	orden

Al terminar de realizar la normalización las tablas relacionadas quedan de la siguiente forma


ADMINISTRADORES
PKadmon
alias
usuario
contrasennia
FACULTADES
PKFacultad
Nombre_Director
Nombre_Facultad
Privacidad
FechaR
FKadmon
LOGO_FACULTADES
FKFacultad
logo
orden
COORDINADORES
PKCoord
grado
alias
sexo
usuario
contrasennia
facultad
fechaR
estado
FKFacultad
FOTOS_COORDINADORES
FKCoord
logo
orden
INSTRUCTORES
PKinstr
grado
alias
usuario
contrasennia
sexo
estado
fecha
FKFacultad
TALLER_PSP
PKtaller
FKFacultad
FKinstr
alias
FechaInicio
FechaFin
FOTOS_INSTRUCTORES
FKInstr
logo
orden
CLASE
PKclase
fecha
FKtaller
HORARIO
dia
he
hs
lugar
FKtaller
ASISTENCIA
valor
FKcarga
FKclase
CARGA
PKcarga
activo
FKestu
FKtaller
1
ESTUDIANTES
FKFacultad
foto
alias
matricula
contrasennia
sexo
grupo
PKestu
HISTORIAL
actual
estado
causa_baja
fecha
FKestu
FOTOS_ESTUDIANTES
FKestu
logo
orden


1
DESARROLLO
PKdes
FKcarga
nombre
lenguaje
total_plan
NoPrograma
DEF_PROBLEMA
FKdes
orden
texto
REQUERIMIENTOS
FKdes
orden
texto
REGISTRO_TIEMPO
FKdes
fecha
inicio
termino
fase
t_interrump
t_efectivo
comentarios
RESUMEN_PLAN
PKplan
categoria_tiempo
FKdes
RESUMEN_FASES
PKfase
fase
real
alafecha
palafecha
FKplan
REGISTRO_DEFECTOS
PKregdef
descripcion
fecha
numero
introducido
eliminado
telim
dev_resuelve
FKdes
FKtipos
TIPOS_DE_DEFECTOS
PKtipos
nombre
descripcion
FASESPSP
FKdes
planeacion
desarrollo
diseño
codificacion
compilacion
pruebas
postmortem

4.4.- DICCIONARIO DE DATOS

TABLA	CAMPO	TIPO	TAMAÑO	DESCRIPCION
ADMINISTRADORES	PKadmon	entero	max	Clave Primaria
	alias	cadena	50	Nombre completo del Administrador
	usuario	cadena	15	Nombre de usuario de la cuenta
	contrasennia	cadena	15	Contraseña del usuario registrado
	sexo	bit	max	Identifica el sexo
	FKfacultad	entero	max	Clave foránea para indicar la facultad de donde proviene
	fecha	fecha		Utilizado para almacenar la fecha en que se dio de alta el administrador
FACULTADES	PKfacultad	entero	max	Clave Primaria
	nombre	cadena	50	Nombre de la facultad
INSTRUCTORES	PKinstr	entero	max	Clave Primaria
	alias	cadena	50	Nombre completo del Instructor
	usuario	cadena	15	Nombre de usuario de la cuenta
	contrasennia	cadena	15	Contraseña del usuario registrado
	sexo	bit	max	Identifica el sexo
	estado	bit	max	Especifica verdadero si está activo o falso si está inactivo
	fecha	fecha	max	Indica la fecha en que fue dado de alta al instructor
	FKfacultad	entero	max	Clave foránea para indicar la faultad de donde proviene
	FKadmon	entero	max	Clave foránea que relaciona ésta tabla con la tabla Administradores
TALLER_PSP	PKtaller	entero	max	Clave Primaria
	alias	cadena	50	Nombre del taller PSP a impartir
	fecha	fecha	max	Fecha en que se dio de alta el taller de PSP
	FKinstr	entero	max	Clave foránea que relaciona ésta tabla con la de Instructores
HORARIO	dia	entero	2	Indica el día en que se impartirá el taller PSP
	he	tiempo	max	Indica la hora en que inicia el Taller PSP
	hs	tiempo	max	Indica la hora en que termina el Taller PSP
	lugar	cadena	30	Se especifica el nombre del laboratorio en que se impartirá el taller PSP
	FKtaller	entero	max	Clave foránea que relaciona ésta tabla con la de TALLER_PSP

TABLA	CAMPO	TIPO	TAMAÑO	DESCRIPCION
CARGA	FKcarga	entero	max	Clave Primaria, ésta clave se construye utilizando un sistema para el control de claves
	activo	bit	max	Especifica el estado de la carga de los alumnos
	fase	entero	max	Indica la fase en la que se encuentra el estudiante
	FKtaller	entero	max	Clave foránea que relaciona ésta tabla con la de Taller_PSP
	FKestu	entero	max	Clave foránea que relaciona ésta tabla con la de Estudiante
CLASE	PKclase	entero	max	Clave Primaria, ésta clave se construye utilizando un sistema para el control de claves
	fecha	fecha	max	Se especificará la fecha en que se impartió el tema
	FKtaller	entero	max	Clave foránea que relaciona ésta tabla con la de TALLER_PSP
ASISTENCIA	valor	entero	3	Valor que recibe el alumno que indica 1.- Si asistió 0.- que no Asistió
	FKcarga	entero	max	Clave foránea que relaciona ésta tabla con la de CARGA
	FKclase	entero	max	Clave foránea que relaciona ésta tabla con la de CLASE
ESTUDIANTES	PKestu	entero	10	Clave Primaria, ésta clave se construye utilizando un sistema para el control de claves
	alias	cadena	30	Nombre completo del alumno a registrar
	matricula	cadena	9	Número de matrícula del alumno
	contrasennia	cadena	40	Se especifica la contraseña del alumno, ésta estará cifrada
	sexo	char	2	Se especifica F para el término femenino y M para el masculino
	grupo	char	2	Se especifica el grupo donde actualmente está inscrito el alumno
	FKfacultad	entero	max	Clave foránea que relaciona ésta tabla con la de FACULTADES
HISTORIAL …	actual	bit	max	Indica el estado en actualmente tenga cada alumno. Verdadero: Estado Actual Falso: Estado Caducado
	estado	entero	2	Contiene mediante un histórico el estado de cada alumno: 1Alta en el Registro 2 Inscrito al Taller PSP 3 Baja Temporal 4 Baja Definitiva 5 Termino su Proceso

TABLA	CAMPO	TIPO	TAMAÑO	DESCRIPCION
… **HISTORIAL**	causa_baja	cadena	255	Se registra la causa de la baja de un alumno cuando el campo "estado" tenga el valor 3 o 4 en otro caso valor nulo
	fecha	fecha	max	Fecha en que ocurre inicia o cambia el evento, pj fecha de alta , fecha de baja temporal, etc.
	FKestu	entero	10	Clave foránea que relaciona ésta tabla con la de estudiante
DESARROLLO	PKdes	entero	10	Clave Primaria, ésta clave se construye utilizando un sistema para el control de claves
	nombre	cadena	255	Nombre del sistema que el desarrollador propone para iniciar las actividades de PSP
	lenguaje	cadena	20	Lenguaje en el desarrolla el sistema
	total_plan	entero	max	Se registra el tiempo total aproximado a utilizar en cada fase según el formulario C14
	FKcarga	entero	10	Clave foránea que relaciona ésta tabla con la de Carga
DEF_PROBLEMA	FKdes	entero	10	Clave foránea que relaciona ésta tabla con la de Desarrollo
	texto	cadena	255	Se inserta todo el texto que requiera el usuario para escribir el documento de definición del problema, si dicho texto sobrepasa los 255 caracteres se ocupará otro registro indicando con el campo orden la secuencia que tiene dicho mensaje al ser recuperado
	orden	entero	3	Se inserta un numero consecutivo para cada bloque de texto que sobrepase los 255 caracteres y que tengan relacion
REQUERIMIENTOS	FKdes	entero	10	Clave foránea que relaciona ésta tabla con la de Desarrollo
	texto	cadena	255	Mismo mecanismo que el campo texto de la tabla DEF_PROBLEMA pero orientado al documento de requerimientos
	orden	entero	3	Se inserta un numero consecutivo para cada bloque de texto que sobrepase los 255 caracteres y que tengan relacion

TABLA	CAMPO	TIPO	TAMAÑO	DESCRIPCION
REGISTRO_TIEMPO	FKdes	entero	10	Clave Primaria, ésta clave se construye utilizando un sistema para el control de claves
	fecha	fecha	max	
	inicio	tiempo	max	Valor con formato de tiempo para definir la hora en que inicia a trabajar con el proyecto de S/W
	termino	tiempo	max	Valor con formato de tiempo para definir la hora en que termina de trabajar con el proyecto de S/W
	t_interrumpido	real	max	Campo utilizado para almacenar el tiempo invertido en interrumpido en minutos
	t_efectivo	real	max	Campo utilizado para almacenar el tiempo bien invertido de trabajo
	fase	cadena	40	Nombre de la fase
	comentarios	cadena	255	Utilizado para escribir algún comentario por el estudiante
RESUMEN_PLAN	PKplan	entero	10	Clave Primaria, ésta clave se construye utilizando un sistema para el control de claves
	categoria_tiempo	cadena	50	Nombre de la categoría de tiempo según el formulario C14: Tiempo en fase Defectos incluidos Defectos Eliminados
	FKdes	entero	10	Clave foránea que relaciona ésta tabla con la de Desarrollo
FASES	PKfase	entero	10	Clave Primaria, ésta clave se construye utilizando un sistema para el control de claves
	fase	cadena	max	Nombre de la fase de cada categoría de tiempo según el formulario C14
RESUMEN_PLAN_FASES …	PKpfase	entero	10	Clave Primaria, ésta clave se construye utilizando un sistema para el control de claves
	FKfase	entero	10	Clave foránea que relaciona ésta tabla con la de Fases
	real	real	max	Utilizado para almacenar el tiempo real en minutos dedicado en cada fase
	alafecha	real	max	El tiempo total invertido
	palafecha	entero	max	Introducir valor del campo alafecha pero en porecntaje

TABLA	CAMPO	TIPO	TAMAÑO	DESCRIPCION
… RESUMEN_PLAN_FASES	FKplan	entero	10	Clave foránea que relaciona ésta tabla con la de Resumen_Plan
REGISTRO_DEFECTOS	PKregdef	entero	10	Clave Primaria, ésta clave se construye utilizando un sistema para el control de claves
	descripcion	cadena	max	Utilizado para que el estudiante pueda escribir alguna descripción
	fecha	fecha	max	Fecha en que efectúa el registro
	numero	entero	max	Numero consecutivo que lleva el numero consecutivo de cada defecto registrado
	introducido	cadena	50	Registre la fase durante la cuál se puede establecer que el defecto fue introducido.
	eliminado	cadena	50	Registre la fase en la cual se encontró y eliminó el defecto
	telim	real	max	Se registra el tiempo que invirtió el estudiante para encontrar y eliminar el error
	dev_resuelve	entero	max	Escribir el numero de defecto corregido y que resolvió otro en caso de que no sepa escribir 0 (X)
	FKdes	entero	max	Clave foránea que relaciona ésta tabla con la de Desarrollo
	FKtipo	entero	max	Clave foránea que relaciona ésta tabla con la de Tipos_de_Defectos
TIPOS_DE_DEFECTOS	PKtipos	entero	max	Clave Primaria, ésta clave se construye utilizando un sistema para el control de claves
	nombre	cadena	50	Nombre del tipo de defecto
	descripcion	cadena	max	Descripción del tipo de defecto
FASESPSP	FKdes	entero	max	Clave foránea que relaciona ésta tabla con la de Desarrollo
	Planeacion	cadena	50	Utilizado para almacenar la fecha para la fase de Planeacion
	Diseño	fecha	max	Utilizado para almacenar la fecha para la fase de Diseño
	Codificación	fecha	max	Utilizado para almacenar la fecha para la fase de Codificación
	Compilación	fecha	max	Utilizado para almacenar la fecha para la fase de Compilación
	Pruebas	fecha	max	Utilizado para almacenar la fecha para la fase de Pruebas
	Postmortem	fecha	max	Utilizado para almacenar la fecha para la fase de Postmortem

CAPITULO 5. IMPLEMENTACIÓN Y PRUEBAS

Este capítulo corresponde a la implementación del sistema, considerando la arquitectura de desarrollo del sistema. Se presenta el ambiente de Desarrollo el Código del Programa de la parte del Ambiente del Servidor, Ambiente del Sistema y Recursos del Hardware.

Es importante reiterar que éste software está desarrollado bajo el lenguaje de Visual Basic .Net 2005, utilizando como manejador de base de datos el SQL Server 2005. Del lado del Servidor se utilizó un sistema operativo Windows Server 2003 y del lado del cliente un Windows XP Pro Service Pack 2.

5.1.- AMBIENTE DE DESARROLLO

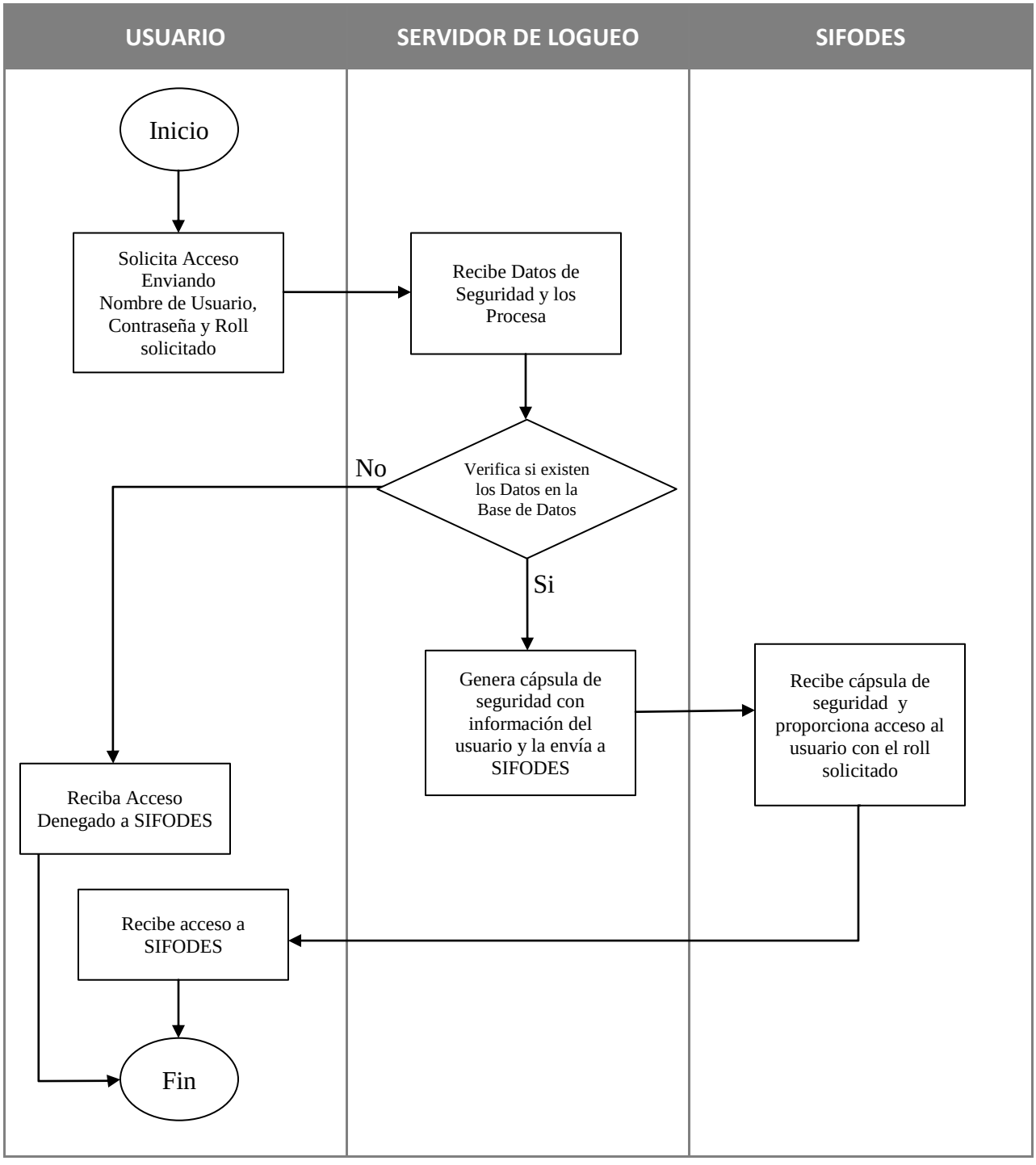
Para comenzar a explicar el funcionamiento del sistema SIFODES (Sistema de Formación para el Desarrollo de Software) es importante mostrar su esquema general del flujo de trabajo la cual consta de de dos componentes básicos: Sistema de Logueo y Sistema de Procesamiento, esto quiere decir que el software soporta 4 tipos de roles: Administrador, Coordinador, Docente y Alumno, cada uno de ellos tiene acceso a las opciones del sistema según su nivel de accesibilidad, para ello se muestran las opciones de accesibilidad para efectuar las actividades correspondientes de cada rol de usuario en la tabla 5.1.

Rol de usuario	Actividades
Administrador	✓ Administración de Facultades ✓ Administración de cuentas para coordinadores de cada Facultad ✓ Capaz de entrar a las opciones de un coordinador mediante el acceso a una facultad, siempre que el coordinador lo permita.
Coordinador	✓ Administración de cuentas para docentes ✓ Administración de cuantas para alumnos ✓ Administración de horarios para talleres PSP ✓ Asignación de cargas PSP para alumnos
Docente	✓ Administración del seguimiento y formación del alumno PSP ✓ Observar los reportes elaborados por los alumnos PSP ✓ Aprobación y acceso a las fases PSP para los alumnos
Alumno	✓ Acceso al seguimiento PSP ✓ Acceso a las fases PSP para su seguimiento ✓ Acceso para el seguimiento del Reporte del tiempo ✓ Acceso para el seguimiento del Registro de Defectos

Tabla 5.1.- Rol de usuario del sistema SIFODES

El uso de un roll para el sistema es utilizado para proporcionarle al usuario los privilegios necesarios para poder acceder a las herramientas del sistema SIFODES, para mostrar esto un poco más fácil se presenta el siguiente Diagrama de Flujo que muestra el flujo de la información únicamente para proporcionar el acceso al usuario.

DIAGRAMA DE FLUJO PARA EL ACCESO A SIFODES



5.2.- CÓDIGO DEL PROGRAMA AMBIENTE SERVIDOR

Dentro de éste código sólo existe la parte de la programación para el acceso al sistema, ésta parte del sistema, la parte del servidor, está elaborada mediante una de las características integrado en Visual Studio .Net 2005, que son las web Service, el cual el código es el siguiente:

```
Imports System.Web
Imports System.Web.Services
Imports System.Web.Services.Protocols
Imports clMisComponentes.Componentes.Net.Conexion.SQL
Imports System.Data

<WebService(Namespace:="http://www.cs.buap.mx/")> _
<WebServiceBinding(ConformsTo:=WsiProfiles.BasicProfile1_1)> _
<Global.Microsoft.VisualBasic.CompilerServices.DesignerGenerated()> _
Public Class MiClaseWeb
    Inherits System.Web.Services.WebService

    Dim cad As String =
ConfigurationManager.ConnectionStrings("CadenaServidorRemoto").ConnectionStrin
g
    'Public cad As String = "Data Source=Atenea; Initial Catalog=Agenda;
Integrated Security=SSPI"
    Public MiConexionSQL As ConexionSQL

    <WebMethod()> _
    Public Function loguear(ByVal user As String, ByVal pass As String, ByRef
isRoot As Boolean) As DataSet
        Dim ds As DataSet = Nothing
        Dim sql As String
        MiConexionSQL = New ConexionSQL()
        If MiConexionSQL.Conectar(cad) = True Then

            sql = "select
administradores.pkadmon,coordinadores.pkcoord,coordinadores.grado,coordinadore
s.alias,coordinadores.usuario,coordinadores.contrasennia,coordinadores.sexo,co
ordinadores.fechar,coordinadores.estado,facultades.pkfacultad,
facultades.nom_facultad from " & _
            "(COORDINADORES inner join FACULTADES on
FACULTADES.PKFacultad=COORDINADORES.FKFacultad) inner join ADMINISTRADORES on
ADMINISTRADORES.PKAdmon=FACULTADES.FKADmon " & _
            "where COORDINADORES.usuario='" & user & "' and
COORDINADORES.contrasennia='" & pass & "'"
            MiConexionSQL.AmbienteDesConectado.EjecutaConsulta(sql, "tabla")
            If
MiConexionSQL.AmbienteDesConectado.getDataSet.Tables("tabla").Rows.Count > 0
Then
                ds = New DataSet()
                ds = MiConexionSQL.AmbienteDesConectado.getDataSet
                isRoot = False
            Else
                sql = "select * from ADMINISTRADORES where usuario='" & user &
"' and contrasennia='" & pass & "'"
                MiConexionSQL.AmbienteDesConectado.EjecutaConsulta(sql,
"tabla")
```

```
                If
MiConexionSQL.AmbienteDesConectado.getDataSet.Tables("tabla").Rows.Count > 0
Then
                    ds = New DataSet()
                    ds = MiConexionSQL.AmbienteDesConectado.getDataSet
                    isRoot = True
                End If
            End If
            MiConexionSQL.Desconectar()
        End If
        Return ds
    End Function

    <WebMethod()> _
    Public Function loguearDocente(ByVal user As String, ByVal pass As String)
As DataSet
        Dim ds As DataSet = Nothing
        Dim sql As String
        MiConexionSQL = New ConexionSQL()
        If MiConexionSQL.Conectar(cad) = True Then
            sql = "select
INSTRUCTORES.PKinstr,INSTRUCTORES.alias,INSTRUCTORES.grado,INSTRUCTORES.usuari
o,INSTRUCTORES.contrasennia,INSTRUCTORES.sexo,INSTRUCTORES.estado,INSTRUCTORES
.fecha,FACULTADES.FKAdmon as PKAdmon " & _
            "from INSTRUCTORES inner join FACULTADES on
FACULTADES.PKFacultad=INSTRUCTORES.FKFacultad " & _
            "where usuario='" & user & "' and contrasennia='" & pass & "'"
            MiConexionSQL.AmbienteDesConectado.EjecutaConsulta(sql, "tabla")
            If
MiConexionSQL.AmbienteDesConectado.getDataSet.Tables("tabla").Rows.Count > 0
Then
                ds = New DataSet()
                ds = MiConexionSQL.AmbienteDesConectado.getDataSet
            End If
            MiConexionSQL.Desconectar()
        End If
        Return ds
    End Function

    <WebMethod()> _
   Public Function loguearEstudiante(ByVal matricula As String, ByVal password
As String) As DataSet
        Dim ds As DataSet = Nothing
        Dim sql As String
        MiConexionSQL = New ConexionSQL()
        If MiConexionSQL.Conectar(cad) = True Then
            sql = "select ADMINISTRADORES.PKAdmon, ESTUDIANTES.PKEstu,
ESTUDIANTES.alias, ESTUDIANTES.matricula, ESTUDIANTES.contrasennia,
ESTUDIANTES.sexo, FACULTADES.PKfacultad " & _
                " from (ESTUDIANTES inner join FACULTADES on
FACULTADES.PKFacultad=ESTUDIANTES.FKFacultad) inner join ADMINISTRADORES on
ADMINISTRADORES.PKAdmon=FACULTADES.FKAdmon" & _
                " where ESTUDIANTES.matricula='" & matricula & "' and
ESTUDIANTES.contrasennia='" & password & "'"
            MiConexionSQL.AmbienteDesConectado.EjecutaConsulta(sql, "tabla")
```

```
            If
MiConexionSQL.AmbienteDesConectado.getDataSet.Tables("tabla").Rows.Count > 0
Then
                ds = New DataSet()
                ds = MiConexionSQL.AmbienteDesConectado.getDataSet
            End If
            MiConexionSQL.Desconectar()
        End If
        Return ds
    End Function

    <WebMethod()> _
    Public Function EstablecerConexion() As Boolean
        Dim ban As Boolean

        MiConexionSQL = New ConexionSQL()
        If MiConexionSQL.Conectar(cad) = True Then
            Dim sql As String = "select * from ADMINISTRADORES"
            MiConexionSQL.AmbienteDesConectado.EjecutaConsulta(sql, "tabla")
            If
MiConexionSQL.AmbienteDesConectado.getDataSet.Tables("tabla").Rows.Count > 0
Then
                ban = True
            Else
                ban = False
            End If
            MiConexionSQL.Desconectar()
        Else
            ban = False
        End If
        Return ban
    End Function

    <WebMethod()> _
    Public Function CargarRegistroTiempo(ByVal registro As DataTable) As
Boolean
        Return True
    End Function

    <WebMethod()> _
    Public Function CargarRegistroDefectos(ByVal registro As DataTable) As
Boolean
        Return False
    End Function
End Class
```

Al invocar la WebService de forma local, mediante una página web (Figura 5.1), muestra los Métodos Web disponibles para consumir, los mismos que se encuentran en el código anterior.

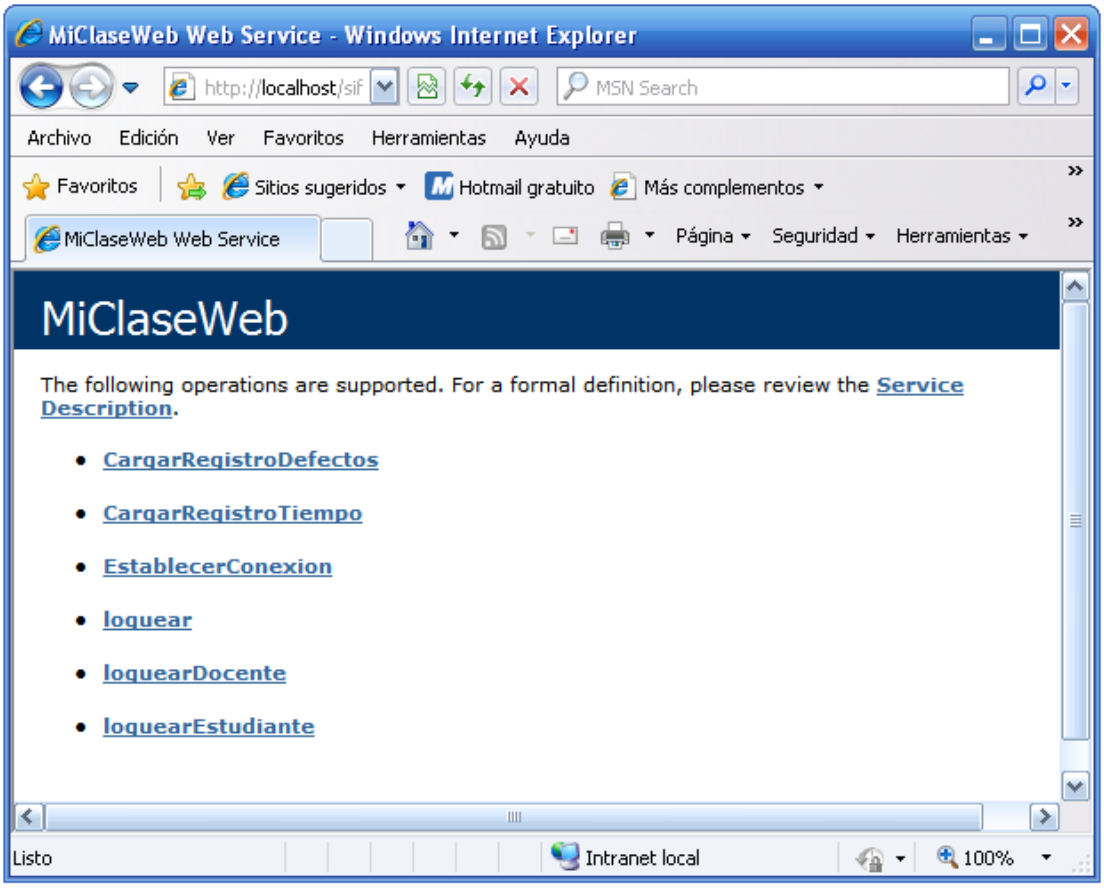


Figura 5.1.- WebService del lado del servidor

5.3.- AMBIENTE DEL SISTEMA

En éste apartado se explica a grandes rasgos el funcionamiento del sistema en tiempo real con datos reales, con el propósito de que el lector se valla familiarizando con la apariencia y las herramientas que ofrecen el software SIFODES.

INICIO Y SISTEMA DE LOGUEO

Existen 4 ejecutables para la parte de logueo en la PC, al instalar se puede optar por instalarlos todos o sólo los que sean necesarios y uno más que es el sistema SIFODES.

Al iniciar el sistema, aparecerá la forma de la metodología PSP (Figura 5.2), esto indica que el sistema está iniciando y contactándose con el servidor para el acceso a SIFODES, esto puede tardar según la velocidad del internet o de la capacidad del servidor.

Figura 5.2.- Forma de espera para el inicio del sistema de logueo

Posteriormente, si se estableció la conexión con el servidor aparecerá la ventana del sistema de logueo, es importante reiterar que existen 4 ejecutables para ésta actividad y es por ello que el color de la ventana y los letreros en el formulario para ello se muestran las cuatro ventanas en las figuras: 5.3, 5.4, 5.5, 5.6.

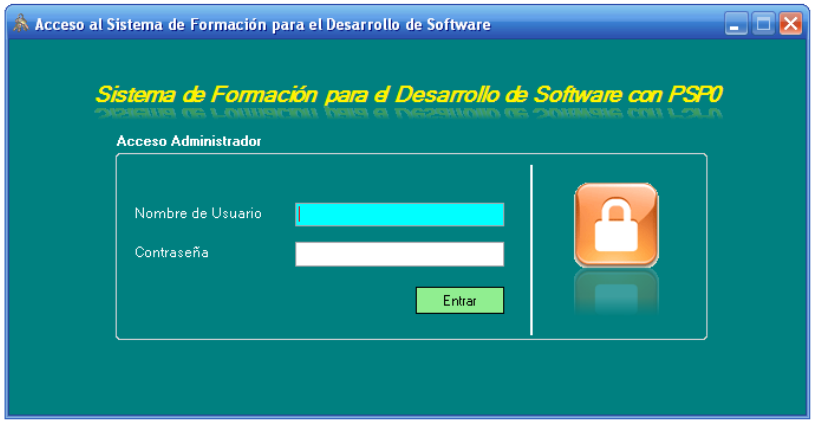


Figura 5.3.- Acceso para el Administrador

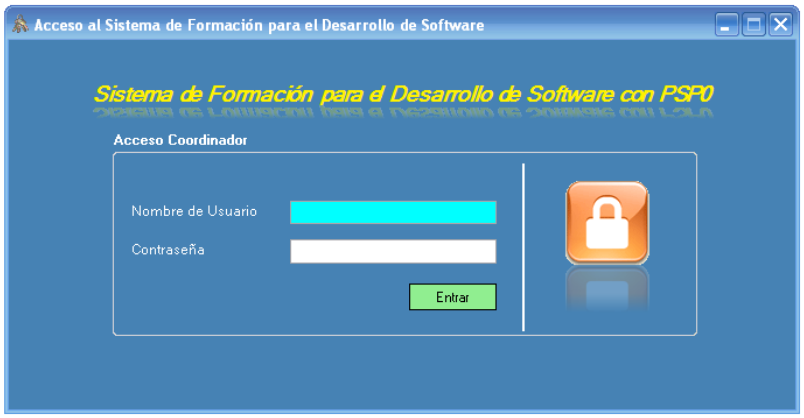


Figura 5.4.- Acceso para el Coordinador

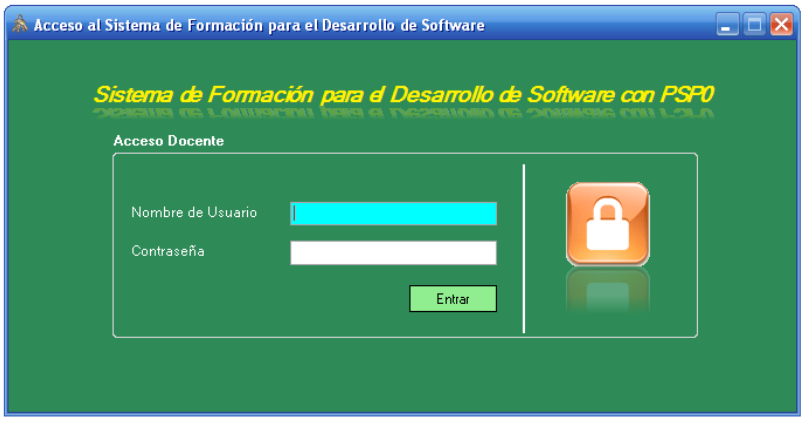


Figura 5.5.- Acceso para el Docente

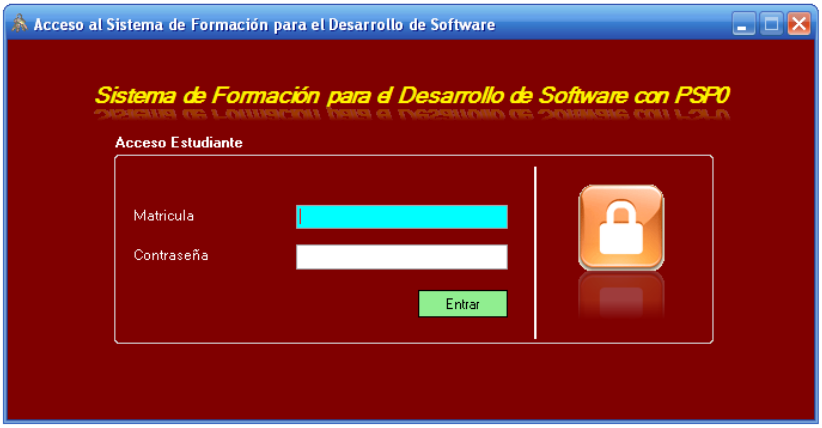


Figura 5.6.- Acceso para el Estudiante

INGRESO AL SISTEMA

Al ingresar al Sistema (Figura 5.7) se muestran las opciones a utilizar, es importante recordar que según la tabla 5.1 los menús y herramientas irán desactivándose dependiendo del roll del usuario.

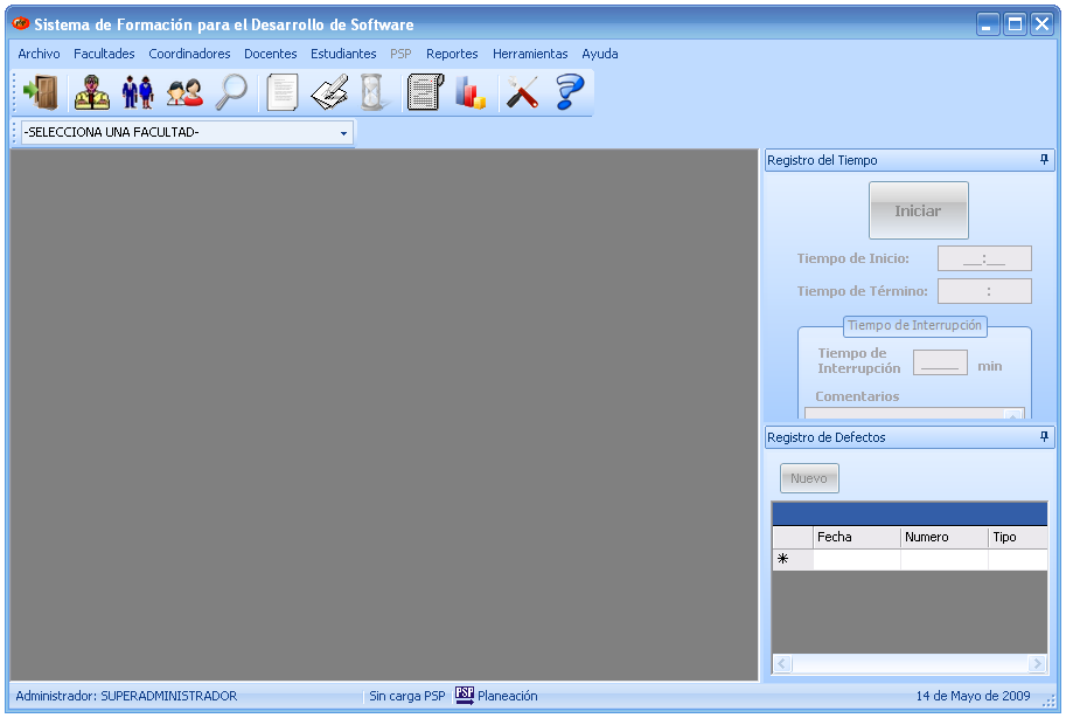


Figura 5.7.- Sistema SIFODES

ADMINISTRACIÓN DE FACULTADES Y COORDINADORES

En el menú facultades existe la opción para invocar el administrador de facultades, utilizado para crear una nueva facultad y comenzar a crear nuevos usuarios para el sistema SIFODES desde las cuentas de los coordinadores. En éste formulario (Figura 5.8) se administrarán las cuentas de usuario para las facultades.

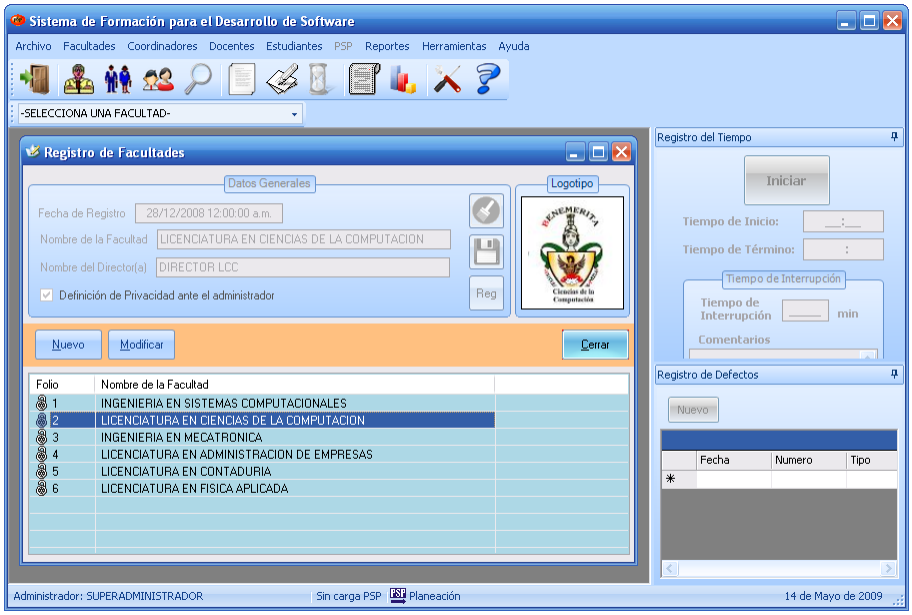


Figura 5.8.- Administrador de Facultades

En la barra de tareas existe un botón (Figura 5.9) para la administración de coordinadores, utilizado para dar de alta o modificar información acerca de los coordinadores que tendrán acceso al sistema.

Figura 5.9.- Icono para el Acceso al Administrador de Coordinadores

El administrador de coordinadores (Figura 5.10) crear nuevos usuarios con el roll de coordinador, y al mismo tiempo es integrado a una facultad para iniciar con usuarios PSP y dar seguimiento al proyecto PSP.

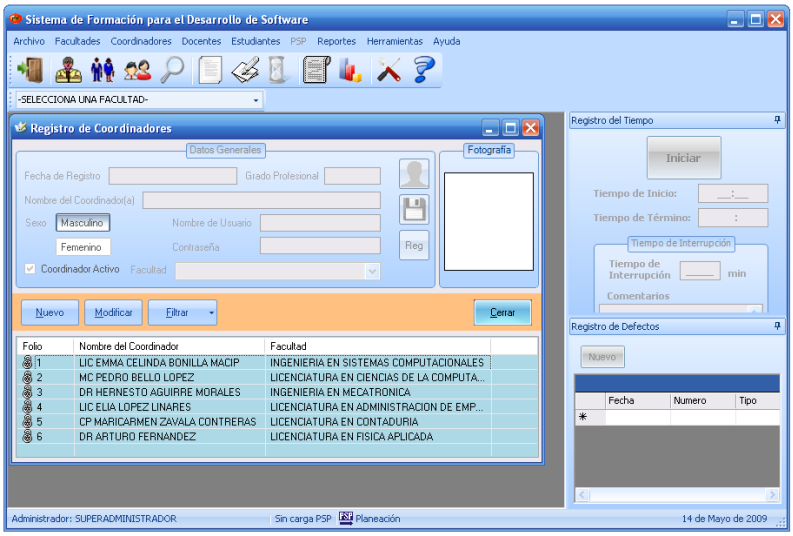


Figura 5.10.- Registro de Coordinadores

En la parte de la Administrador de coordinadores mostrada en la figura 5.11, se observa una lista de coordinadores, previamente registrados, pero esto solo puede observarse cuando el usuario sea de tipo Administrador, en otro caso, si es un usuario de tipo Docente solo se mostrará la información del Docente en curso y podrá activar o desactivar la opción de "Definición de Privacidad Ante el Administrador", utilizado para que el administrador no pueda adoptar el roll de un docente e impida la administración de Docentes, Alumnos, Talleres y Cargas.

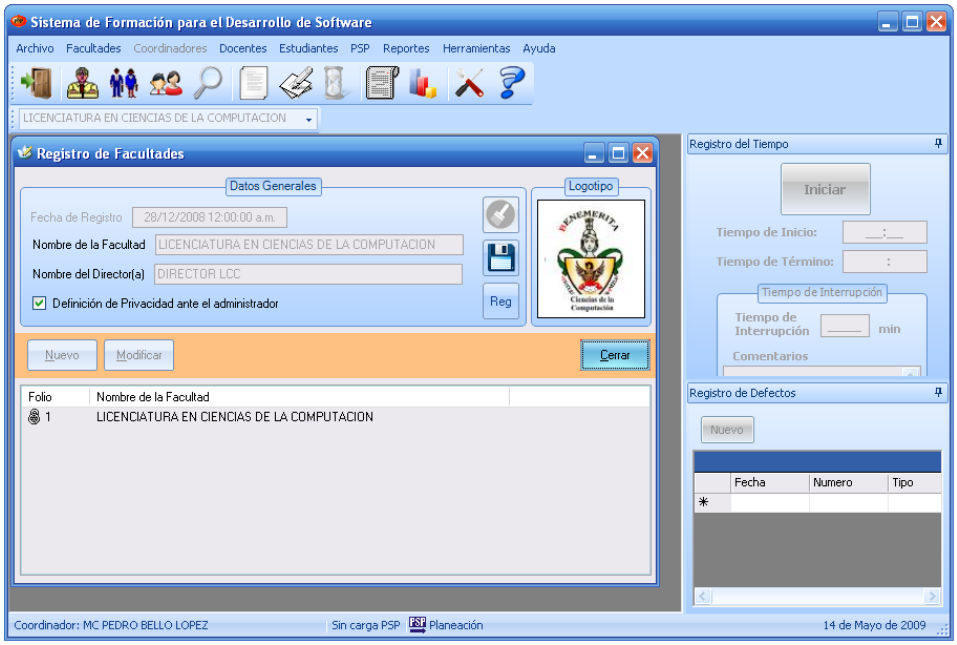


Figura 5.11.- Registro de Facultades

ADMINISTRACIÓN DE DOCENTES

El Administrador de Docentes (Figura 5.12) es utilizado para crear usuario con el roll de docente, éstos serán los encargados de dar seguimiento a los usuarios PSP.

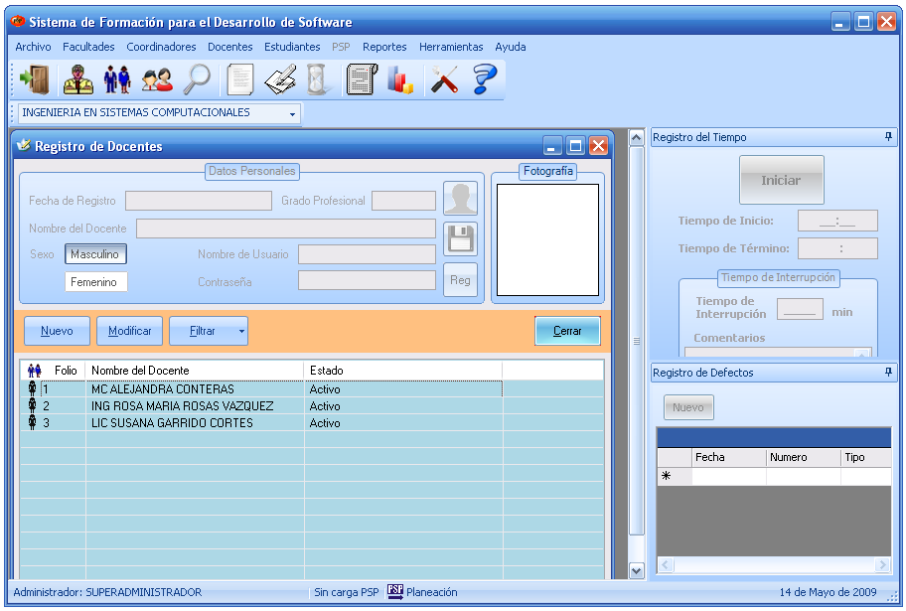


Figura 5.12.- Registro de Docentes

ADMINISTRACIÓN DE ESTUDIANTES

El Administrador de Estudiantes (Figura 5.13) es utilizado para crear usuario con el roll de Estudiantes, al crear un nuevo usuario de éste roll, inicialmente es registrado con al estado denominado "Sin carga PSP", la causa es que aun no ha recibido alguna carga PSP para que éste inicie su seguimiento PSP. Por otro lado, éste administrador es capaz de cambiar de estado a un Estudiante siendo las siguientes: Sin carga PSP, Egresado PSP, Baja Temporal, Baja Definitiva, Reingreso PSP.

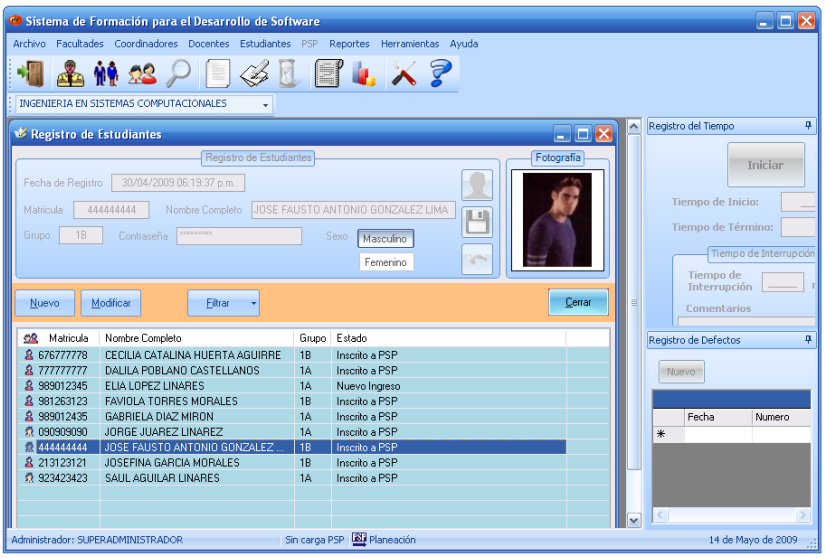


Figura 5.13.- Registro de Estudiantes

ADMINISTRACIÓN DE TALLERES Y CARGAS

El Administrador de Talleres (Figura 5.14) es utilizado para dar de alta los cursos o bien denominados talleres PSP disponibles para los estudiantes, indicando el horario, lugar y Docente que impartirá el taller.

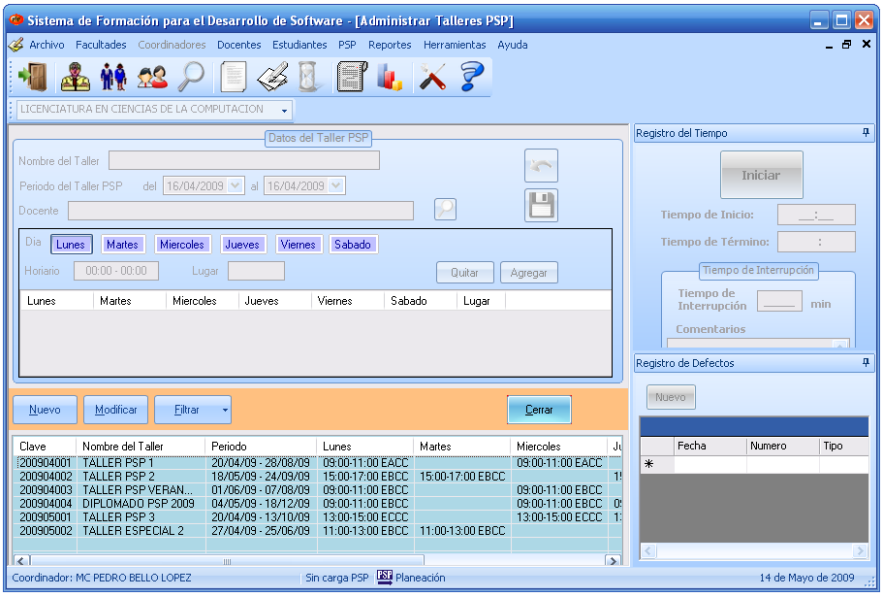


Figura 5.14.- Registro de Talleres PSP

Una vez que el taller PSP esté disponible, mediante una convocatoria por parte de la administración, los alumnos acudirá con el coordinador PSP para su registro utilizando el administrador de cargas PSP (Figura 5.15) y así los alumnos podrán dar seguimiento a su formación PSP.

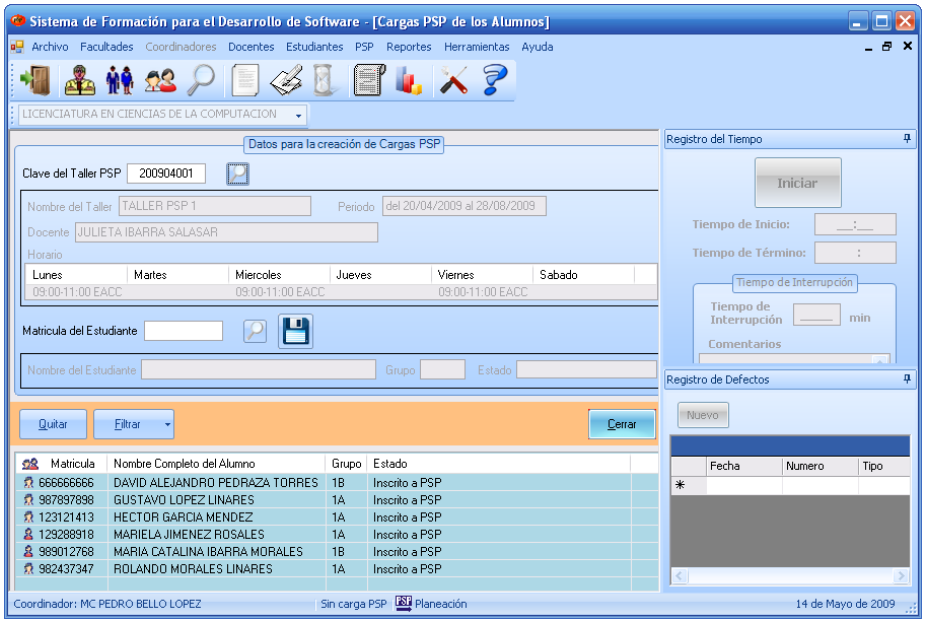


Figura 5.15.- Registro de Cargas PSP para los Alumnos

ADMINISTRACION PARA EL SEGUIMIENTO PSP

Esta herramienta es exclusiva de los Docentes PSP, el cual es utilizada para observar el seguimiento PSP (Figura 5.16) de cada estudiante y conforme el Estudiante avanza con su formación y entregando resultados el Docente irá activando cada fase para que el Estudiante avance al próximo nivel.

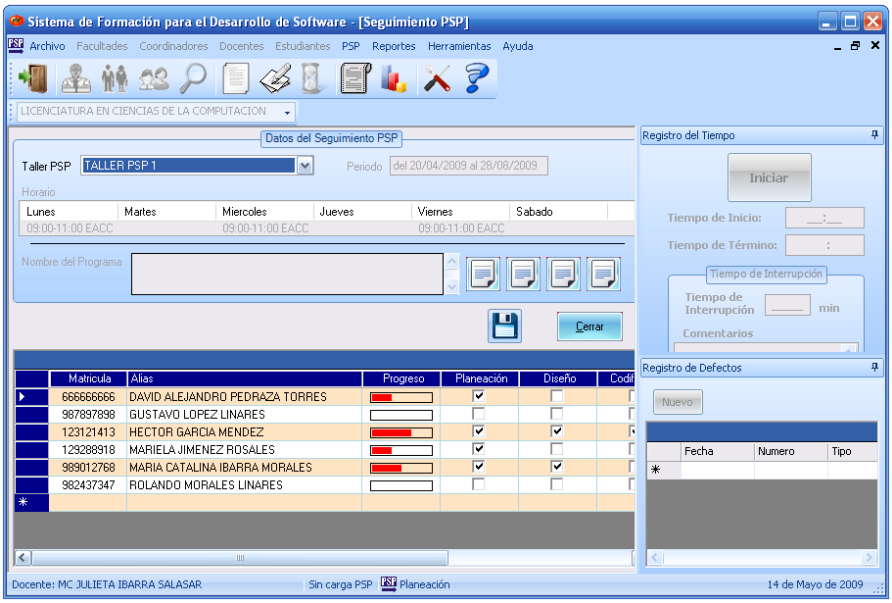


Figura 5.16.- Seguimiento PSP

La herramienta es capaz de mostrar la lista de talleres PSP que tiene asignado el Docente y a su vez mostrar l lista de Estudiantes por cada taller para su seguimiento.

HERRAMIENTAS PARA LA FASE DE PLANEACIÓN PSP

Para la fase de planeación de la metodología PSP se encuentran integrados los formularios denominados: Definición del Problema (Figura 5.17), Plan del registro PSP0 (Figura 5.18) y Documento de Requerimientos (Figura 5.19).

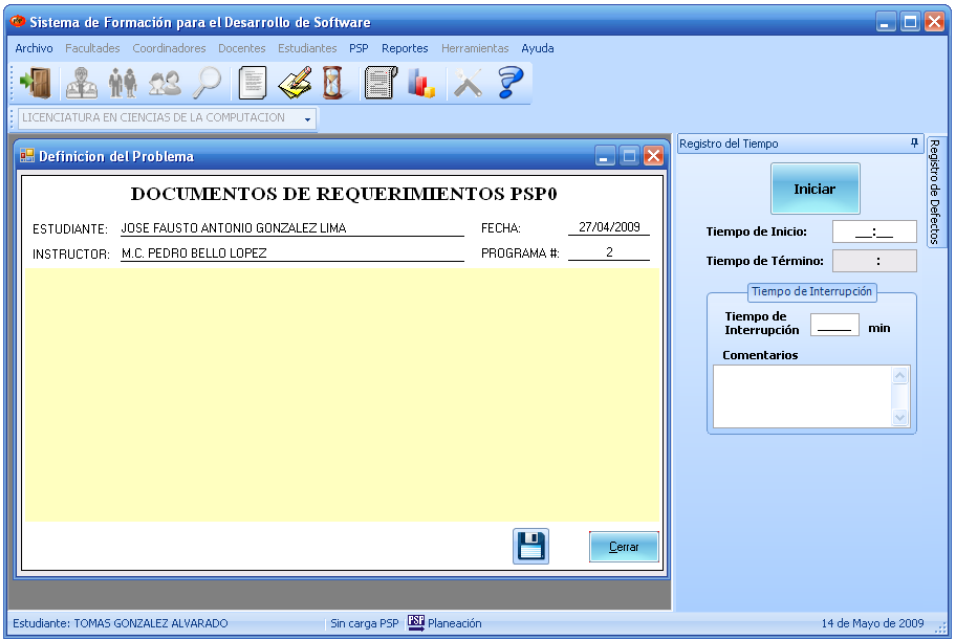


Figura 5.17.- Formulario para el Seguimiento PSP

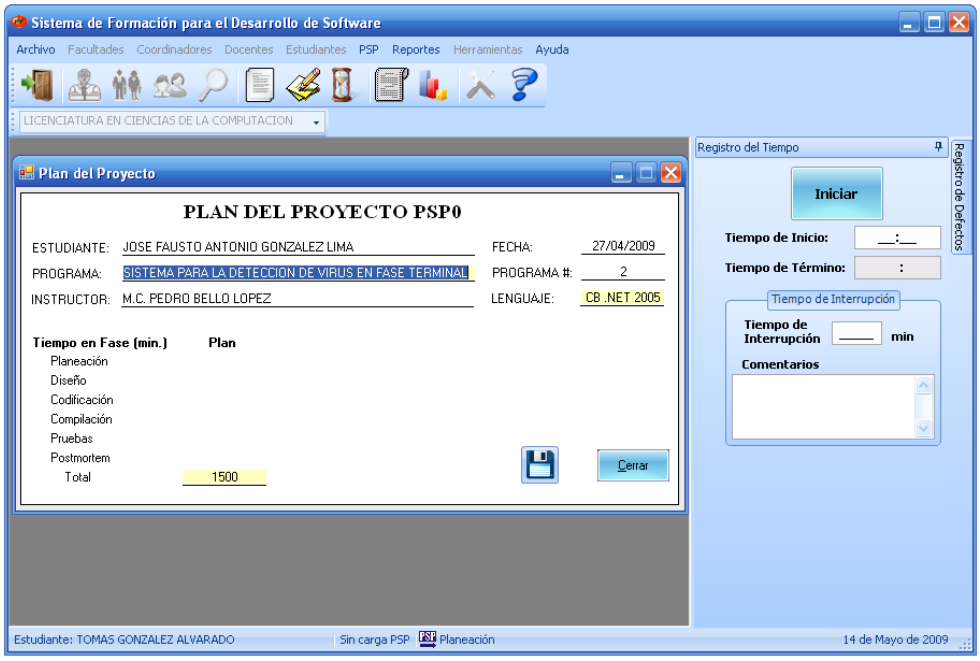


Figura 5.18.- Formulario para el Pan del Proyecto

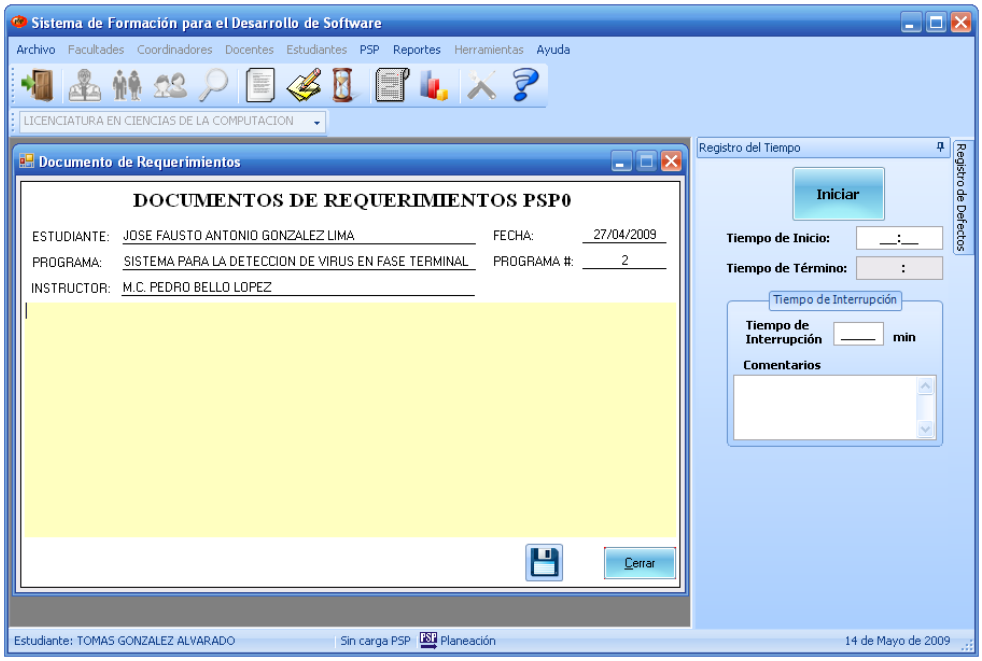


Figura 5.19.- Formulario para el Documento de Requerimientos

Es importante mencionar que para las fases de Diseño, Codificación y Compilación se deberá utilizar un lenguaje de programación como: Java, Visual Studio .Net, PHP, JSP, ASP, etc., sin embargo el registro del tiempo y de defectos para éstas fases deberá utilizar el sistema SIFODES para almacenar información acerca del tiempo utilizado y los defectos encontrados.

REGISTRO DEL TIEMPO Y DE DEFECTOS

Estas herramientas se encuentran a la derecha de la ventana principal del sistema SIFODES, la causa de que se encuentre en ése lugar es para que los Estudiantes puedan tener un acceso rápido a dichas herramientas.

La herramienta para el Registro del Tiempo (Figura 5.20) es utilizada para ir registrando el tiempo invertido durante cada fase de la metodología PSP.

Figura 5.20.- Registro del Tiempo

La herramienta para el Registro de defectos (Figura 5.21) es utilizada para ir registrando los defectos encontrados en el sistema durante cada fase de la metodología PSP.

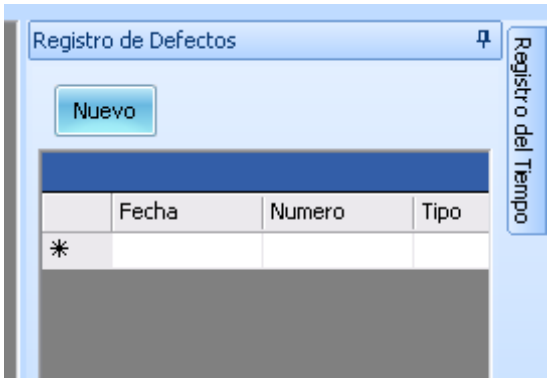


Figura 5.21.- Registro de Defectos

5.4.- RECURSOS DE HARDWARE Y SOFTWARE

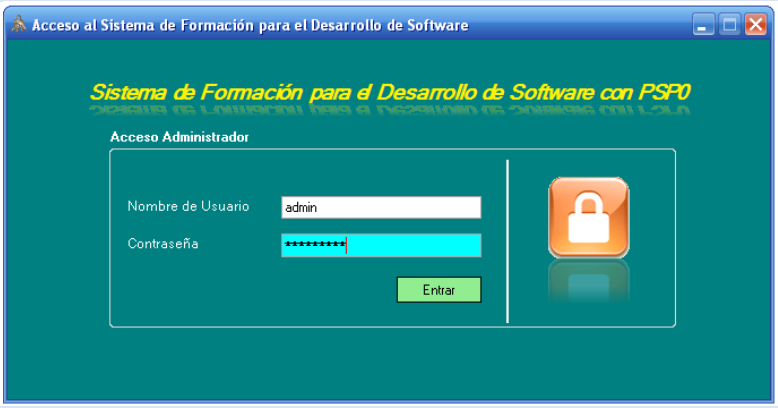
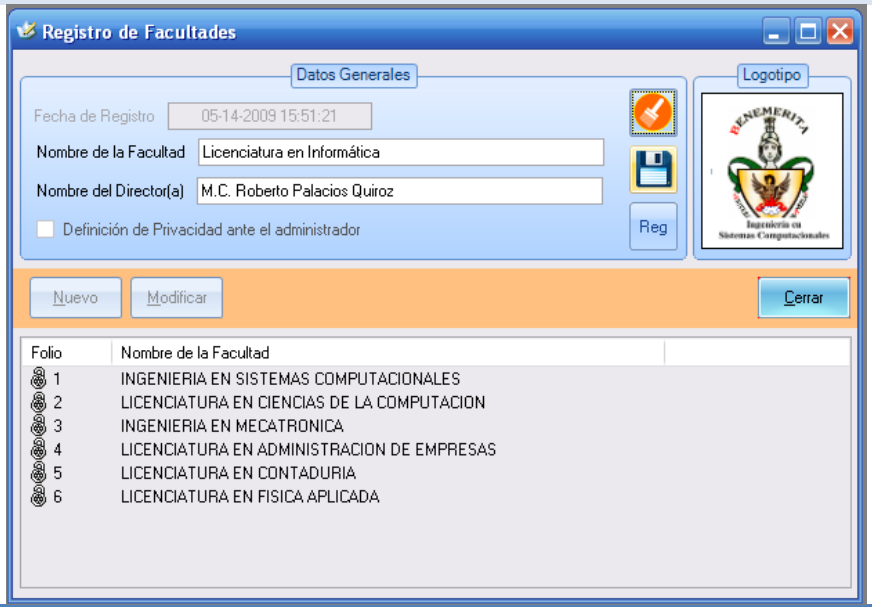
Los siguientes recursos son recomendación del autor de ésta tesis, ya que el sistema se probó utilizando equipos con el hardware presente y con el uso de software presente, cabe mencionar que el servidor y los clientes tienen que estar configurados y tener instalados los programas y parches necesarios para su funcionamiento.

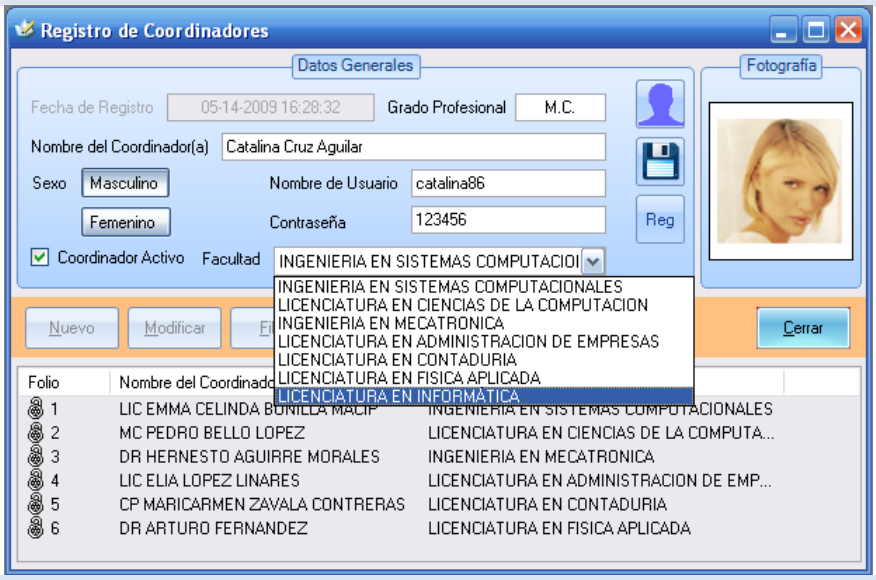
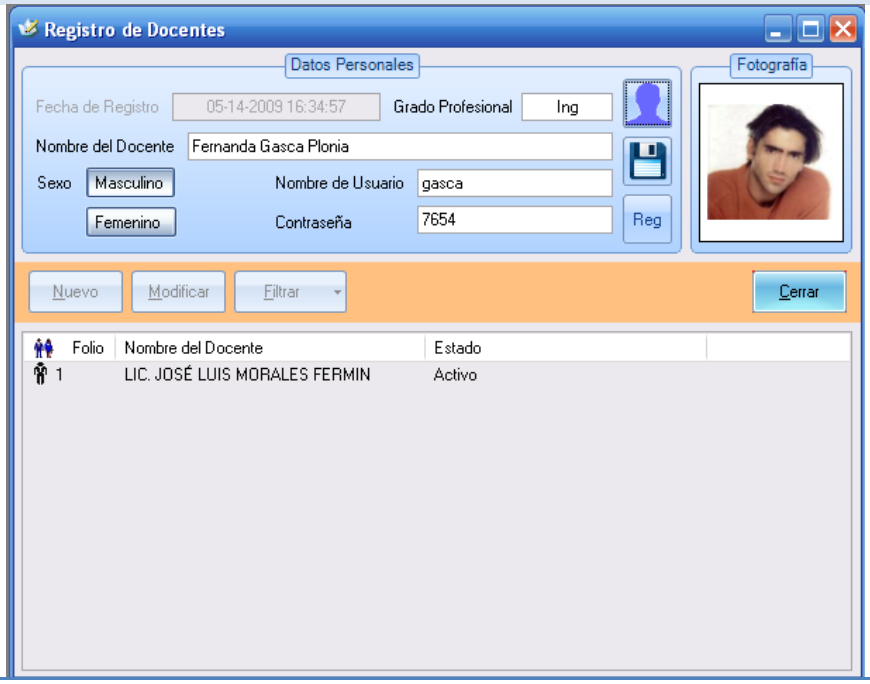
- ✓ **Hardware del lado del Servidor:**
  - Procesador Core 2 Duo de 2.5 Ghz
  - Memoria de 2 Gb
  - Disco de 120 Gb de capacidad
  - Tarjeta de Red 10/100/1000 Gigabit
  - Tarjeta de Video
- ✓ **Hardware del lado del Cliente**
  - Procesador core 2 duo 1 2.3 Ghz.
  - Memoria 1 Gb.
  - Disco duso de 120 Gb
  - Tarjeta de Red 10/100
- ✓ **Software del lado del Servidor**
  - Widows Server 2003 R2 Service pack 1
  - .Net Framework 2.0 o superior
  - SQL Server 2005
  - Windows Installer 3.1
  - IIS 6.0
- ✓ **Software del lado del Cliente**
  - Windows XP Pro Service Pack 2
  - .Net Framework 2.0 o superior
  - Windows Installer 3.1

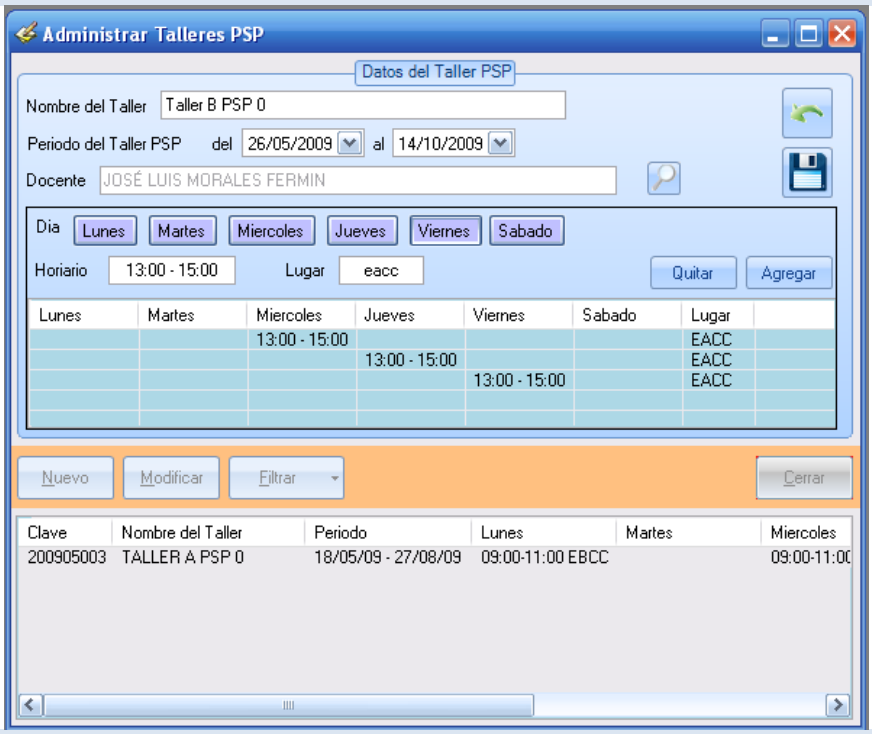
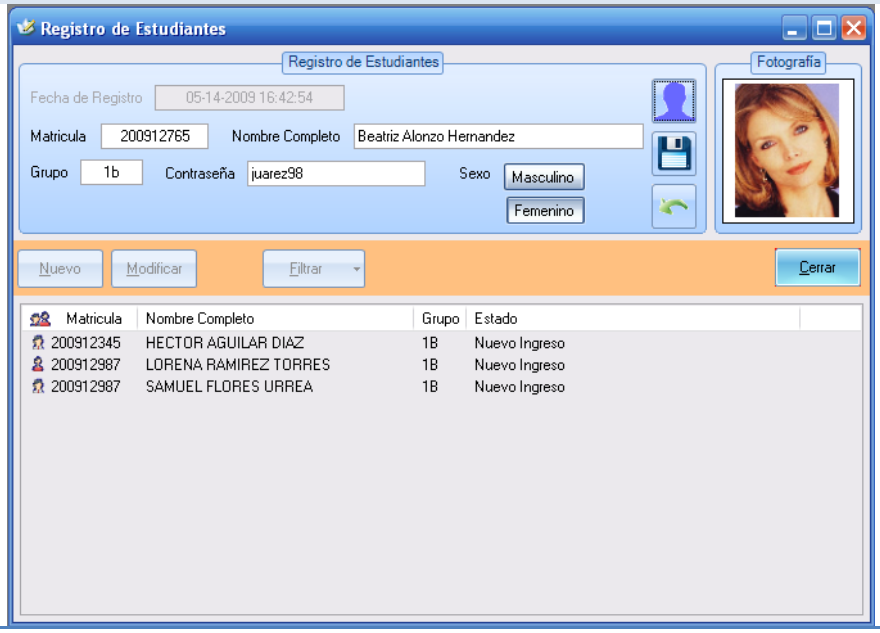
5.5.- ENTRADAS DEL SISTEMA

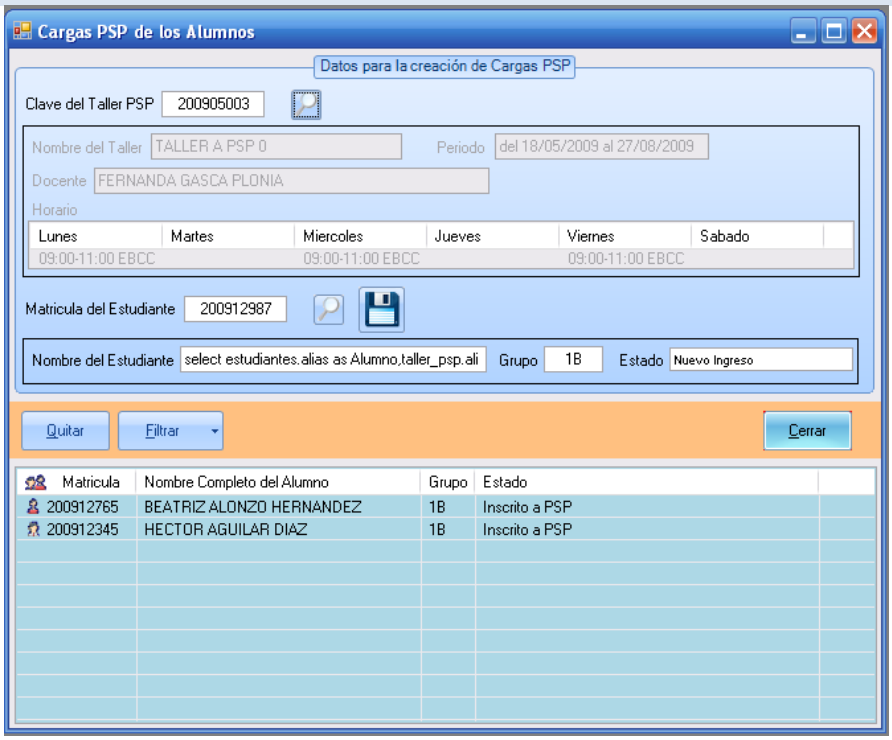
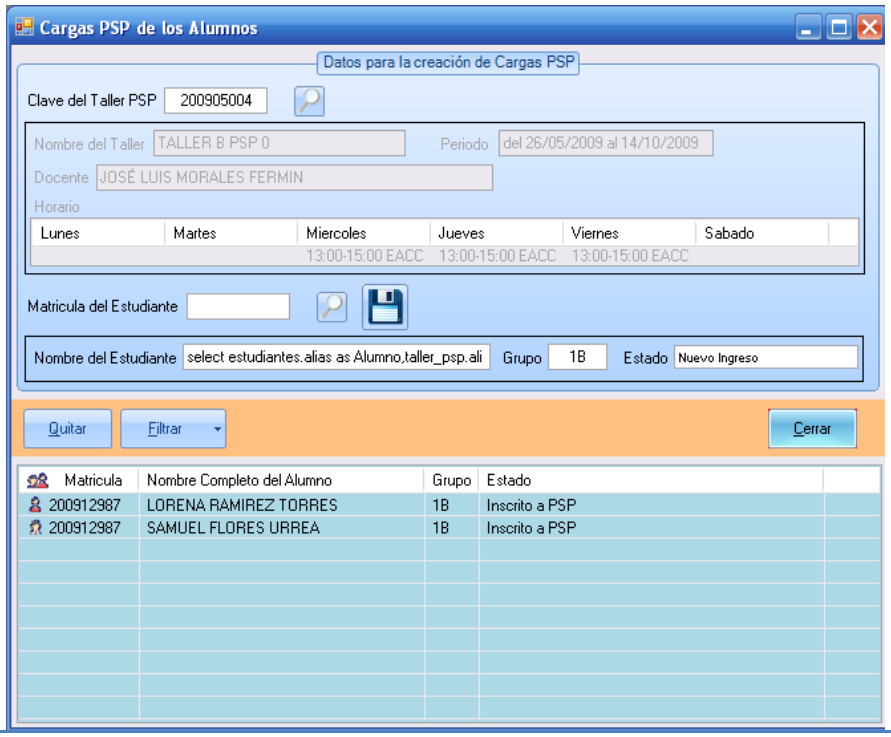
En el presente apartado se mostrará el sistema en acción, con datos reales con el objetivo de mostrar el funcionamiento del sistema SIFODES.

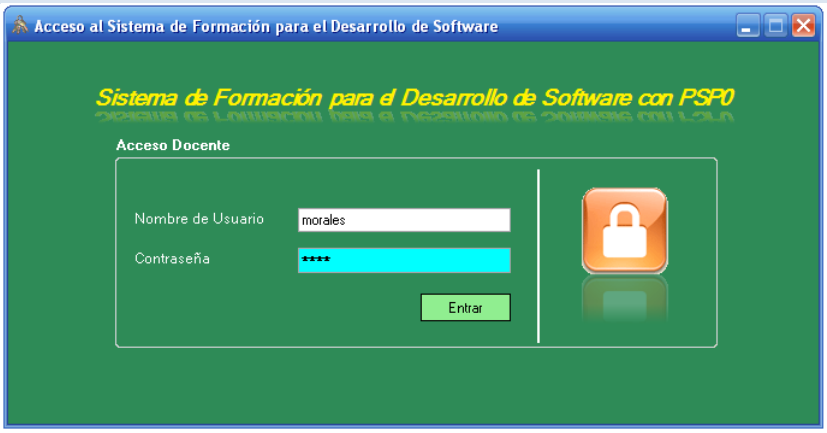
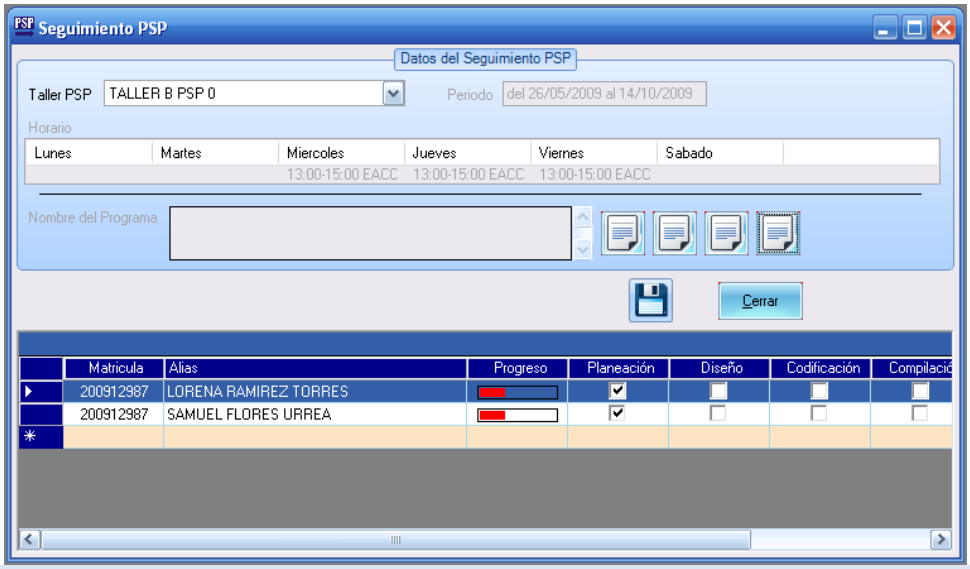
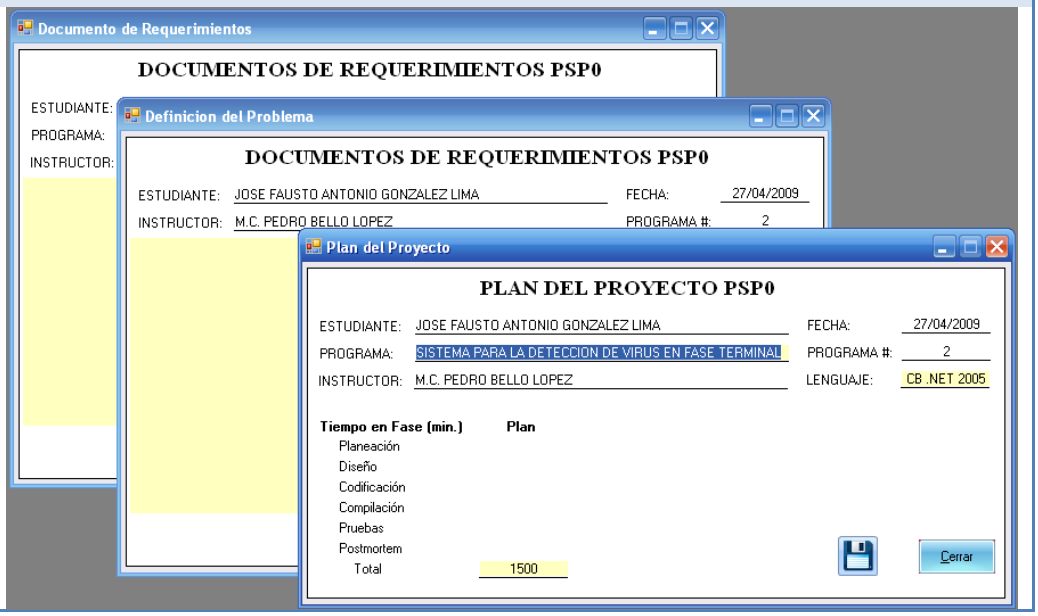
Solo se mostrarán las herramientas más relevantes de cada roll de usuario.

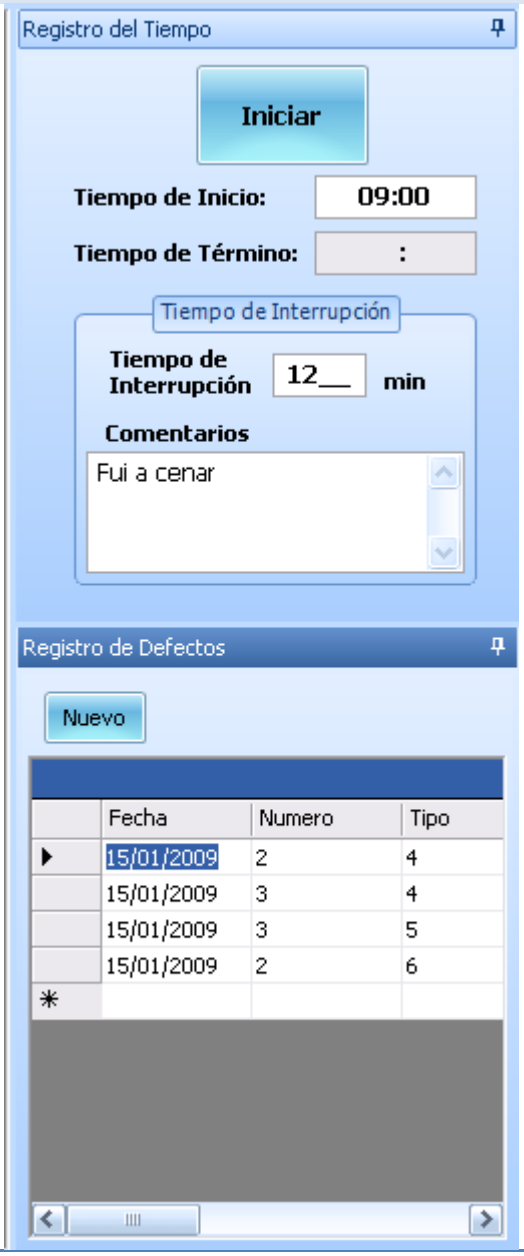
OPERACIÓN	PANTALLA
Acceso a SIFODES con el roll de administrador	
Se crea una nueva carrera denominada *Licenciatura en Informática*	

OPERACIÓN	PANTALLA
Con el registro de coordinadores se le asigna un responsable a la facultad. El responsable será M.C. Catalina Cruz Aguilar Nombre de Usuario: catalina86 Contraseña: 123456	
Una vez creada la facultad y su coordinador, desde ésta cuenta se procederá a crear dos docentes: Lic. José Luis Morales Fermin (U= morales, P=9876), y Ing. Fernanda Gasca Polonia (U=gasca, P=7654)	

OPERACIÓN	PANTALLA
Desde ésta misma cuenta de usuario se crean dos talleres: Taller A PSP 0 y Taller B PSP 0	
Una vez que ya se tienen registrados los docentes y los cursos es hora de registrar los alumnos para los cuales se registrarán 4: Donde uno de ellos será Beatriz Alonso Juárez (M=200912765, P=juarez98	

OPERACIÓN	PANTALLA
Posteriormente registramos dos alumnos a cada taller para que cada docente lleve su seguimiento	 

OPERACIÓN	PANTALLA
En la siguiente pantalla, se accesará a SIFODES pero con el roll de Docente, con el propósito de proporcionar el acceso a la fase inicial de la técnica PSP la cual es Planeación, y entonces el estudiante deberá iniciar su trabajo cumpliendo con ésta fase para que el Docente observe su seguimiento.	 
Una vez que cada docente activó la fase de planeación se muestra una pantalla desde el sistema con el rol de un Estudiante para llenar los formatos de la primera fase.	

OPERACIÓN	PANTALLA
En cada fase el Estudiante deberá utilizar el registro del tiempo para el cálculo de la inversión de tiempo y también la utilización del formulario del Registro de Defectos.	

CONCLUSIONES

El presente proyecto, fue creado en base a la metodología de cascada, utilizando todas sus fases para lograr terminar con todas las herramientas necesarias para su uso y aplicación.

El hábito de planear y mejor aun, seguir al pie de la letra lo planeado logra que el estudiante tenga una formación inigualable y que el sistema desarrollado se termine en tiempo y forma, gracias a la herramienta de Registro del tiempo de SIFODES es posible llevar una buena administración del tiempo invertido. Aunado a lo anterior, el sistema SIFODES es capaz de tener un control de errores de código para observar el avance en el desarrollo de software, y herramientas para el seguimiento de cada etapa PSP0.

El sistema de base de datos para el nivel 0 de PSP o bien denominado SIFODES (Sistema de Formación para el Desarrollo de Software), es una herramienta eficaz utilizado como apoyo en el aprendizaje para el desarrollo de software, gracias a su forma de trabajo extraído de la técnica PSP específicamente nivel 0. Cabe mencionar que es necesaria la presencia de un docente para orientar a los estudiantes y lograr terminar cada fase del nivel PSP0.

Integrado a SIFODES, contiene una serie de herramientas para la parte de la administración de cada roll de usuario, como la administración de: Facultades, coordinadores, docentes, estudiantes, talleres PSP, cargas, etc., esto con el propósito de llevar un buen control de datos para el seguimiento PSP0 de cada estudiante.

Del lado del estudiante, éste tiene herramientas para pueda efectuar sus actividades PSP0 y registrar su seguimiento sin problema alguno, sin embargo, desafortunadamente el sistema no cuenta con un compilador para que el estudiante pueda desarrolla software e ir generando sus registros PSP de forma automática, por el contrario, se deberá utilizar el formulario de SIFODES para llenar los registros de PSP.

Todo el desarrollo del sistema SIFODES, tuvo un gran esfuerzo de análisis para la preparación y creación del mismo: una lista de requerimiento bien establecida, la creación de una base de datos normalizada, el desarrollo del sistema utilizando un lenguaje de programación orientado a

objetos denominado Visual Basic .Net 2005 junto con un manejador de base de datos potente como lo es SQL Server 2005.

El uso de Visual Studio .net 2005 y SQL Server 2005, fue una gran experiencia por su forma de programar utilizando el modelo Orientado a Objetos, la fortaleza que tiene en su ejecución, compatibilidad con la familia Windows, la administración de la memoria y trabajo con un servidor implementando las Web Services. Especialmente recomiendo el uso de tecnología .Net para desarrollar software, puesto que tiene una serie de herramientas muy poderosas para ser implementadas en un desarrollo de software.

Para la parte del servidor, existe una aplicación el cual es el corazón del sistema SIFODES, ésta herramienta es la que logra que los usuario se conecten al sistema y capaz de transferir comandos del lado del cliente para ser procesadas y ejecutadas sobre la base de datos y así devolver algún tipo de resultado, o efectuar cambios sobre los registro de la base como eliminarlos, actualizarlos o insertarlos, por todo esto, SIFODES logra ser una herramienta eficaz, veloz y segura para la implementación de la técnica PSP nivel 0.

Por último quiero recordar de que la técnica PSP contiene una serie de niveles, y que éste sistema solo cubre el nivel 0, quedando libre el desarrollo de un sistema o varios sistemas para lograr terminar el resto de los niveles PSP, proporcionando una oportunidad de mejora para la culminación de éste trabajo para que así la formación de los estudiantes esté completa, ya que quien termine toda la capacitación PSP logrará ser un desarrollador de sistemas totalmente íntegro y autosuficiente.

BIBLIOGRAFÍA

[1] Ceballos, Francisco Javier. ***"Visual Basic .Net"***
Microinformática

[2] Dewson, Robin. ***"Beginning SQL Server 2005 Express for Developers"***
Apress, 2007.

[3] Humprey, Watts S. ***"a Discipline for Software Engineering"***
Addison Wesley, 1996.

[4] Humprey, Watts S. ***"Introducción al Proceso de Software Personal"***
Addison Wesley, 1997.

[5] Johnson, Glen and Tony Northrup. ***"Microsoft .Net Framework 2.0 Web-Based Client Development"***. Microsoft Press, 2007

[6] Korth and Silberschat. ***"Fundamentos de Bases de Datos***"
2ª edición, Mc Graw-Hill.

[7] Luca de Tena , Juan Ignacio. ***"Microsoft Visual Basic 6"***
Anaya Multimedia, S.A., 1999

[8] Northrup, Tony. ***"Microsoft .Net Framework 2.0 Application Development Foundation"***.
Microsoft Press, 2006.

[9] Preoutsos, Evangelos. ***"Microsoft Visual Basic 6 La Biblia"***
Anaya Multimedia.

[10] Pressman, Roger S. "***Ingeniería del Software Un enfoque Práctico"***
5ª Edición, McGraw Hill.

[11] Reid, Fiach. ***"Network Programming in .Net with C# and Visual Basic .Net"***
ELSEVIER Digital Press, 2003

[12] Sceppa, David. ***"Programming Microsoft ADO .net 2.0"***
Microsoft Press, 2005.

[13] Silberchatz, Abraham. ***"Fundamentos de Bases de Datos"***
4ª Edición, Mc Graw Hill, 2002.

[14] Solid Quality Learning. ***"Microsoft SQL Server 2005 Applied Techniques Step by Step"***.
Microsoft Press, 2006.

[15] Solid Quality Learning. ***"Microsoft SQL Server 2005 Implementation and Maintenance"***.
Microsoft Press, 2007.

[16] Sommerville, Ian. ***"Ingeniería de Software"***
6ª Edición, Addison Wesley.

ANEXOS A: PLANTILLAS PSP0

Los siguientes anexos son prácticamente la base para cumplir con la técnica PSP0, éstos proporcionan información general y detallada de las actividades que debe desempeñar el estudiante para cumplir con dicha técnica, los cuales son los siguientes:

- ✓ El **script del proceso** contiene información de la técnica de PSP 0 de una forma general, con una serie de documentos necesarios que deberá tener el estudiante antes de comenzar con la PSP0. También enmarca la serie de fases a seguir las cuales son: *Planeación*, *Desarrollo* y *Postmortem*.
- ✓ Los Scripts de **Planeación**, **Desarrollo** y **Postmortem** muestran de forma detallada las actividades a desempeñar cada fase.

Posteriormente se muestran 3 formatos junto con sus instructivos de llenado que son utilizados para almacenar los registros de PSP0 que enmarca cada fase, los cuales son los siguientes: Resumen del Plan del Proyecto, Bitácora de registro del tiempo, Bitácora de Registro de Defectos; para el llenado del último formato será utilizado el estándar de Tipos de Defectos.

Tabla C10 Script del Proceso PSP0

FASE	PROPÓSITO	PARA GUIARLO EN EL DESARROLLO DE PROGRAMAS DE NIVEL MÓDULO
	Requerimientos de Entrada	• Descripción del Problema • PSP0 Formulario Resumen del Plan del Proyecto • Bitácoras de Registro del tiempo y de defectos • Tipos estándar de defectos • use Cronómetro (opcional)
1	Planeación	• Producir y obtener un documento de Requerimientos. • Estimar el tiempo de desarrollo requerido. • Registrar los datos del plan en la forma de resumen del plan del proyecto. • Actualizar la bitácora del tiempo.
2	Desarrollo	• Diseñar el Programa. • Implementar el Diseño. • Compilar el programa, eliminar y registrar todos los defectos encontrados. • Probar el programa, eliminar y registrar todos los defectos encontrados. • Actualizar la bitácora de tiempo.
3	Postmortem	Contemplar la forma del resumen del plan del proyecto con los datos reales de tiempo, defectos y tamaño.
	Criterio de Salida	• Un programa totalmente Probado • El Resumen del Plan del Proyecto completo con datos estimados y reales. • Terminar los defectos y la Bitácora del Tiempo.

Tabla C11 PSP0 Script de Planeación

FASE	PROPÓSITO	PARA GUIAR EL PROCESO DE PLANEACIÓN PSP.
	Criterios de Entrada	• Descripción del Problema • PSP0 Formulario Resumen del Plan del Proyecto • Bitácora de Registro del Tiempo
1	Requerimientos del Programa	• Producir u obtener una bitácora de requerimientos del programa. • Asegurarse que la declaración de requerimientos es clara y sin ambigüedades. • Resolver cualquier duda.
2	Estimación de Recursos	• Hacer mejor su estimado del tiempo requerido para desarrollar este programa.
	Criterios de Salida	Un documento de requerimientos. Un resumen del Plan del Proyecto con datos estimados del tiempo de desarrollo. Completar la Bitácora de Registro de Tiempo

Tabla C12 PSP0 Script de Desarrollo

FASE	PROPÓSITO	PARA GUIAR EL DESARROLLO DE PROGRAMAS PEQUEÑOS
	Criterios de Entrada	• Documento de Requerimientos. • Resumen del plan del proyecto con datos estimados del tiempo de desarrollo. • Bitácoras de Registro del Tiempo y de Defectos. • Estándar de Tipos de Defectos.
1	Diseño	• Revisar los requerimientos y producir un diseño que los satisfaga. • Registrar el tiempo en la Bitácora de Tiempo.
2	Codificación	• Implementar el Diseño. • Registrar en la bitácora de Defectos cualquier defecto de requerimientos o diseño encontrado. • Registrar el tiempo en la Bitácora de Tiempo.
3	Compilación	• Compilar el programa hasta que esté libre de errores. • Corregir todos lo defectos encontrados. • Registrar los defectos en la bitácora de defectos. • Registrar el tiempo en la Bitácora de Tiempo.
4	Pruebas	• Probar hasta que todas las pruebas se ejecuten sin ningún error. • Corregir todos los registros encontrados. • Registrar los defectos en la bitácora de defectos. • Registrar el tiempo en la Bitácora de Tiempo.
	Criterios de Salida	• Un programa totalmente probado. • Completar la Bitácora de Registro de Defectos. • Completar la Bitácora de Registro de Tiempo

Tabla C13 PSP0 Script de Postmortem

FASE	PROPÓSITO	PARA GUIAR EL PROCESO DE POSTMORTEM DE PSP
	Criterios de Entrada	• Descripción del problema y especificación de requerimientos. • Resumen del plan del proyecto con datos estimados del tiempo de desarrollo. • Bitácora de Registro del Tiempo elaborada • Bitácora de Defectos elaborada • Un programa probado y en funcionamiento.
1	Defectos Introducidos	• Determinar desde la Bitácora de Defectos, los defectos introducidos en cada fase de PSP0. • Introducir este número en virtud de los defectos resultantes actuales en el Resumen del Plan del Proyecto
2	Eliminación de Defectos	• Determinar desde la Bitácora de Defectos los defectos eliminados en cada fase de PSP0. • Introducir este número en virtud de los defectos Eliminados-Actuales en el Resumen del Plan del Proyecto.
3	El Tiempo	• Revisar la Bitácora de Tiempo Terminada. • Introducir el tiempo total gastado en cada fase de PSP0 en virtud del Resumen del Plan del Proyecto Actual.
	Criterios de Salida	• Un programa plenamente probado. • Completar el Resumen del Plan del Proyecto. • Terminar los defectos y la Bitácora de Tiempo.

TABLA C14 Resumen del Plan de Proyecto PSP0

Estudiante		Fecha	
Programa		Programa #	
Instructor		Lenguaje	

Tiempo en Fase (min.)	**Plan**	**Real**	**A la Fecha**	**% a la Fecha**
Planeación				
Diseño				
Codificación				
Compilación				
Pruebas				
Postmortem				
Total				

Defectos Introducidos	**Real**	**A la Fecha**	**% a la Fecha**
Planeación			
Diseño			
Codificación			
Compilación			
Pruebas			
Postmortem			
Total del Desarrollo			

Defectos Eliminados	**Real**	**A la Fecha**	**% a la Fecha**
Planeación			
Diseño			
Codificación			
Compilación			
Pruebas			
Total del Desarrollo			
Después del Desarrollo			

Tabla C15 Instructivo de llenado del Resumen del Plan del Proyecto

Propósito	Este formulario tiene la estimación y proyecto actual de los datos en una forma conveniente y fácilmente recuperable.
Cabecera	Introduzca la Siguiente: • tu nombre y todos los datos • el nombre y número del programa • el nombre del instructor • el lenguaje que tu usas para escribir el lenguaje
Tiempo en Fase	• En el campo Plan, introduzca su estimación original del total del tiempo de desarrollo. • En el campo Real, introduzca el tiempo real en minutos dedicado en cada fase de desarrollo. • En el campo "a la Fecha", introduzca la suma del tiempo real y el tiempo a la fecha de su más reciente programa desarrollado. • En el campo, % a la Fecha, Introduce el porcentaje del tiempo a la fecha en cada fase.
Defectos Introducidos	• En el campo Actual, Introduzca el número de defectos encontrados en cada fase. • En el campo, a la Fecha, introduzca la suma de los números reales de los defectos encontrados en cada fase y los valores, a la fecha, del programa desarrollado recientemente. • En el campo, % a la Fecha, introduzca el porcentaje de defectos encontrados en casa fase hasta a hora.
Defectos Eliminados	• En el campo Actual, introduzca el numero de defectos eliminados en cada fase. • En el campo, a la Fecha, introduzca la suma de la cantidad real de los defectos eliminados en cada fase y el valor hasta la fecha del programa desarrollado recientemente. • En el campo, % ala Fecha, introduzca el porcentaje de los datos eliminados por fase hasta la fecha. • Tras el desarrollo, registrar los defectos encontrados más tarde durante el uso del programa, en la reutilización o modificación.

Tabla C16 Bitácora de Registro del Tiempo

Estudiante	______________________________	Fecha	__________
Instructor	______________________________	Programa #	__________

Fecha	Inicio	Término	Tiempo de Interrupción	Tiempo Efectivo	Fase	Comentarios

Tabla C17 Instructivo de llenado para la Bitácora de Registro del Tiempo

Propósito	Este formulario es para registrar el tiempo empleado en cada fase del proyecto. Estos datos se utilizan para completar el plan de proyecto de Resumen.
General	• Registrar todo el tiempo que invierte en el proyecto. • Registrar el tiempo en minutos. • Ser lo más preciso posible. Si necesitas espacio adicional, usa otra copia de este formulario
Encabezado	Registre el nombre, fecha, instructor y número de programa que se está desarrollando (puede existir más de una hoja que corresponda al programa 1 elaborado en C y mas hojas con el # 2 elaborado en Java)
Fecha	Registre la fecha del día.
Ejemplo	18/10/2008
Inicio	Registre el tiempo en minutos cuando inicie una fase del proyecto.
Ejemplo	8:20
Detener	Registre el tiempo en minutos cuando usted detuvo su trabajo en una fase del proyecto, aún si no ha terminado esa fase.
Ejemplo	10:56
Tiempo de Interrupción	Registre cualquier tiempo que perdió debido a interrupciones dentro del periodo de Inicio y Detención.
Ejemplo	37 – tome un descanso
Tiempo Efectivo	Registre el tiempo transcurrido entre los tiempos de Inicio y Detención menos el tiempo de Interrupción.
Ejemplo	From 8:20 to 10:56, less 37 minutes or 119 minutes.
Fase	▪ Anote la fase en la que usted está trabajando. ▪ Use el nombre de la fase.
Ejemplo	Planeación, código, pruebas, y así sucesivamente.
Comentarios	▪ La interrupción ▪ La tarea que realizó ▪ Cualquier otra cosa que afecte significativamente tu trabajo
Ejemplo	Había un problema de compilación y tuvo que recibir ayuda
Importante	Es importante registrar todo el tiempo laborado. Si se le olvido registrar el inicio, al detenerse o en un tiempo de interrupción para alguna tarea, introduzca su mejor estimación del tiempo.

Tabla C18 Bitácora de Registro de Defectos

Tipos de Defectos	
10 Documentación	60 Chequeo de Tipos
20 Sintaxis	70 Datos
30 Componentes o Configuración	80 Función
40 Asignación	90 Sistema
50 Interfaz	100 Ambiente

Estudiante ______________________ Fecha __________

Instructor ______________________ Programa #. __________

Fecha	Numero	Tipo	Introducido	Eliminado	Tiempo Elim.	Defecto Rel.

Descripción: ______________________

Fecha	Numero	Tipo	Introducido	Eliminado	Tiempo Elim.	Defecto Rel.

Descripción: ______________________

Fecha	Numero	Tipo	Introducido	Eliminado	Tiempo Elim.	Defecto Rel.

Descripción: ______________________

Fecha	Numero	Tipo	Introducido	Eliminado	Tiempo Elim.	Defecto Rel.

Descripción: ______________________

Fecha	Numero	Tipo	Introducido	Eliminado	Tiempo Elim.	Defecto Rel.

Descripción: ______________________

Fecha	Numero	Tipo	Introducido	Eliminado	Tiempo Elim.	Defecto Rel.

Descripción: ______________________

Tabla C19 Instructivo de Llenado de la Bitácora de Registro de Defectos PSP0

Propósito	Registrar los datos en cada defecto encontrado y corregirlo. Usa estos datos para completar el Resumen de Plan del Proyecto
General	• Registrar todos los defectos encontrados en la revisión y pruebas • Registrar por complete cada defecto por separado • Si necesitas espacio adicional, usa otra copia de este formato.
Encabezado	Registre: ▪ Nombre ▪ Fecha ▪ Instructor ▪ Número de programa
Fecha	Registre la fecha cuando encuentre y corrige el defecto.
Numero	Registre un número único para este defecto. Inicie cada proyecto con 1, pro ejemplo 1 ó 001
Tipo	Registre el tipo de defecto a partir del estándar de tipos de defectos.
Introducido	Registre la fase durante la cuál se puede establecer que el defecto fue introducido.
Eliminado	Registre la fase en la cuál se encontró y eliminó el defecto.
Tiempo de Eliminación	Registre el tiempo que le tomó encontrar y eliminar el defecto. Usted puede tomar el tiempo exacto o utilizar su mejor juicio.
Defecto que Resuelve	Si este defecto fue introducido mientras se estaba eliminando otro defecto, registre el número de ese defecto o una X si usted no sabe. Notas – Un defecto es cualquier cosa en el programa que debe ser modificada para que sea apropiadamente desarrollada, mejorada o utilizada.
Descripción	Escriba una descripción detallada de los defectos que son suficientemente claros para recordar más tarde sobre el error y le ayudará a recordar la razón por la que usted hizo

Tabla C20 Estándar de Tipos de Defectos

TIPO	NOMBRE DEL TIPO	DESCRIPCIÓN
10	Documentación	Comentarios, mensajes
20	Sintaxis	ortografía, puntuación, tipos, formatos de instrucción
30	Componentes o configuración	La gestión del cambio, biblioteca, el control de versiones
40	Asignación	declaraciones, nombres duplicados, alcance, limites
50	Interfaz	Llamadas a procedimientos y referencias, I/O, formatos de usuario
60	Chequeo de Tipos	Mensajes de error, controles inadecuados
70	Datos	estructura, contenido
80	Función ó Lógicos	lógicas, apuntadores, ciclos, recursividad, cómputo, función de los defectos
90	Sistema	configuración, calendario, memoria
100	Ambiente	diseño, compilado, pruebas, u otros problemas de soporte