UNIVERSIDAD NACIONAL AUTONOMA DE NICARAGUA
UNAN-LEON

DEPARTAMENTO DE COMPUTACION
FACULTAD DE CIENCIAS

Monografía para optar al título de:

Ingeniero en Sistemas de Información.

Sistema Automatizado de Registro Académico.

Integrantes:

Rebeca Nohemí Ruiz Ortiz

Ana Fabiola Rodríguez Pinell

Tutor: MSc. Danilo Padilla

Julio, 2008

Tema:

Diseño de una aplicación en Visual C # y Mysql Server 5.0, para automatizar los procesos del Área de Registro Académico de la Facultad de Ciencias Económicas y Empresariales UNAN-LEON.

I-Introducción.

Una de las áreas de mayor importancia de las facultades, es Registro Académico, ya que este lleva el control de la información, se debe tener sumo cuidado a la hora de procesarla; es por eso que necesita de un software seguro y completo.

Registro Académico representa todas las gestiones referentes a la información de los estudiantes que incluye desde sus datos personales y toda la información de su vida académica.

Actualmente en Registro Académico esta trabajando bajo el Sistema de bloques el cual es un mecanismo importante para garantizar la formación de profesionales altamente competitivos y trabajo independiente del estudiante en sus procesos de aprendizaje.

Nuestro sistema pretende garantizar, una mayor seguridad, facilitando el trabajo de los usuarios, permitiendo el fácil acceso a la información, realizando esto en menos tiempo, que cuando se realizaba de forma manual. Se podrán obtener Informes y reportes completos del estudiante para una mejor representación de los datos.

El lenguaje que se ha utilizado en esta aplicación, es visual C #, el cual tiene como objetivo permitir a todos los desarrolladores en general y a los de C y C ++ en particular, abordar el desarrollo de aplicaciones complejas con facilidad y rapidez, pero sin sacrificar la potencia y el control que ofrecen C y C++.

II-Antecedentes.

En el departamento de computación de la facultad de Ciencias de nuestra universidad no se ha realizado trabajo monográfico en el lenguaje de visual C # en el que queremos trabajar. Además en la facultad de Ciencias Económicas y Empresariales y específicamente en el Área de Registro Académico a la cual le pretendemos desarrollar esta aplicación no posee un sistema automatizado que cumpla con las necesidades que hoy en día esta demandando, ya que cuenta con un sistema desactualizado, en lenguaje de programación Visual Fox Pro V.6, .

El Registro Académico posee un software basado en Visual Fox Pro V.6, que no posee una interfaz grafica moderna y esto provoca que la información de los diferentes estudiantes sea más difícil de gestionar para los empleados.

De acuerdo a lo anterior, hemos decidido elaborar un sistema que cumpla con todos los requerimientos y que además sea atractivo para el usuario y seguro.

III-Justificación.

Actualmente la forma en que el área de Registro Académico ha venido manipulando el comportamiento de la información no ha sido de forma eficiente, realizando todavía tareas de forma manual. Dicha área cuenta con un sistema que no cumple con todos los requerimientos necesarios para un óptimo manejo de los datos.

Consideramos que la realización de nuestra aplicación facilitará las tareas y funciones principales de Registro Académico, como es la generación de reportes y consultas para los empleados de forma automática en base a la información que se le proporcione al sistema, mejorara la presentación de las constancias de notas.

IV-Objetivos:

General:

- **D**esarrollar un sistema automatizado basado en un entorno visual, que permita manejar la información necesaria para una gestión óptima de los procesos de la oficina de Registro Académico de la Facultad de Ciencias Económicas de la UNAN-LEÓN.

- **C**onocer el potencial de Visual C # frente a otros leguajes de Programación.

Específicos:

- **F**acilitar el manejo para el usuario a través de informes y reportes.

- **M**antener información personalizada y actualizada de los estudiantes.

- **O**ptimizar el sistema minimizando el tiempo de trabajo y de respuesta a una petición, al llevar el proceso de forma manual.

- **E**xplicar las ventajas de Visual C # y su importancia en la actualidad.

V-MARCO TEÓRICO.

Desde hace algún tiempo, el ser humano ha querido ayudarse de máquinas para el cálculo de cualquier tipo. En un principio fue el Ábaco y luego las máquinas de Leibnitz, las fichas perforadas de Jaquard y las máquinas mecánicas de ruedas como calculadora.

Hablar de tecnología es referirnos a la expansión de las computadoras en el uso del correo electrónico y del Internet en las rutinas laborales de una oficina de gobierno, Institución, etc.

La tecnología se aplica a la vida social económica, y cultural no solo nos posibilita producir más en menos tiempo, tener más información o imaginar nuevas soluciones a nuevos problemas, modifica espacios y tiempo, esquemas de socialización nuestra relación con el entorno natural.

Sistemas de Información:

La mayoría de los sistemas de Información ya sean implantado en sistemas de cómputos grandes o pequeños, utilizan una base de datos que pueden abarcar varias aplicaciones, por esta razón estos sistemas utilizan un administrador de base de datos, en este caso el diseñador no construye la base de datos sino que consulta a su administrador para ponerse de acuerdo en el uso de esta en el sistema.

El desarrollo de sistemas tiene dos componentes.

Análisis Es el proceso de clasificación e interpretación de hechos, diagnostico de problemas y empleo de la información para recomendar mejoras al sistemas. Especifica que es lo que el sistema debe hacer.

Diseño: Especifica las características del producto terminado. Establece como alcanzar el objetivo.

El análisis y diseño de sistemas se refiere al proceso de examinar la situación de una empresa con el propósito de mejorar con métodos y procedimientos más adecuados.

Elementos de un Sistema de Información:

Software. Los programas de computadoras, las estructuras de datos y la documentación asociada, que sirve para realizar el método lógico.

Hardware: Los dispositivos electrónicos que proporcionan la capacidad de computación y que proporcionan las funciones del mundo exterior.

Gente: Los individuos que son usuarios y operadores del software y del hardware.

Bases de Datos: Una colección grande y organizada de información a la que se accede mediante el software y que es una parte integral del funcionamiento del sistema.

Documentación: Los manuales, los impresos y otra información descriptiva que explica el uso y / o la operación.

Procesamiento: Los pasos que definen el uso especifico de cada elemento del sistema o el contexto procedimental en que reside el sistema.

Control: Los sistemas trabajan mejor cuando operan dentro de niveles de control tolerables de rendimiento por ejemplo: el sistema de control de un calentador de agua.

Características de un Sistema de Información:

Sus principales características son:

- Suelen lograrse ahorros significativos de mano de obra.
- Son el primer tipo de sistemas de información que se implanta en las organizaciones.
- Son intensivos en entradas y salidas de información.
- Sus cálculos y procesos suelen ser simples y poco sofisticados, requieren mucho manejo de datos para poder realizar sus operaciones y como resultado generan también grandes volúmenes de información.
- Tiene la propiedad de ser recolectores de información.
- Son adaptables de aplicación que se encuentran en el mercado.

Funciones de un Sistema de Información:

- Reservaciones aéreas.
- Departamento de registro hospitalario.
- Control escolar.
- Preparación de nóminas en operaciones bancarias.
- Sistema de intercomunicación electrónica.

Confiabilidad depende de:

- La corrección permanente de su diseño.
- Lo correcto de la correspondencia entre este.
- Lo que quiere el usuario.
- Funcionalidad de los componentes.

Ingeniería del software: Es una disciplina que integra métodos, herramientas y procedimientos para el desarrollo de programas con la intención de brindar el apoyo en la toma de decisiones.

Ciclo de vida de un sistema: También llamado modelo de cascada o SDLC (símbolos en ingles).Es un enfoque sistemático y secuencial por fases del análisis y diseño de un sistema de información.

Características.

Las fases nunca se llevan como un paso a parte.

Varias actividades pueden suceder simultáneamente.

Las actividades pueden repetirse.

CICLO DE VIDA

Ciclo de vida de un sistema:

- Definición del problema.
- Recopilación de información.
- Análisis.
- Programación.
- Diseño.
- Prueba.
- Documentación.
- Implementación.

Base de Datos:

Una base de datos o banco de datos es un conjunto de datos que pertenecen al mismo contexto almacenados sistemáticamente para su posterior uso. En este sentido, una biblioteca puede considerarse una base de datos compuesta en su mayoría por documentos y textos impresos en papel e indexados para su consulta. En la actualidad, y debido al desarrollo tecnológico de campos como la informática y la electrónica, la mayoría de las bases de datos tienen formato electrónico, que ofrece un amplio rango de soluciones al problema de almacenar datos.

En informática existen los sistemas gestores de bases de datos (SGBD), que permiten almacenar y posteriormente acceder a los datos de forma rápida y estructurada. Las propiedades de los sistemas gestores de bases de datos se estudian en informática.

Una base de datos orientada a objetos es una base de datos que incorpora todos los conceptos importantes del paradigma de objetos:

- **Encapsulación:** Propiedad que permite ocultar la información al resto de los objetos, impidiendo así accesos incorrectos o conflictos.
- **Herencia**: Propiedad a través de la cual los objetos heredan comportamiento dentro de una jerarquía de clases.
- **Polimorfismo:** Propiedad de una operación mediante la cual puede ser aplicada a distintos tipos de objetos.

En bases de datos orientadas a objetos, los usuarios pueden definir operaciones sobre los datos como parte de la definición de la base de datos. Una operación (llamada función) se especifica en dos partes. La interfaz (o signatura) de una operación incluye el nombre de la operación y los tipos de datos de sus argumentos (o parámetros). La implementación (o método) de la operación se especifica separadamente y puede modificarse sin afectar la interfaz. Los programas de aplicación de los usuarios pueden operar sobre los datos invocando a dichas operaciones a través de sus nombres y argumentos, sea cual sea la forma en la que se han implementado. Esto podría denominarse independencia entre programas y operaciones.

Modelo de base de datos.

Bases de datos Relacionales:

Las bases de datos Relacionales son una de las formas más efectivas de organizar los datos en una base de datos.

Una base de datos relacional es una base de datos en donde todos los datos visibles al usuario están organizados estrictamente como tablas de valores, y en donde todas las operaciones de la base de datos operan sobre estas tablas.

Estas bases de datos son percibidas por los usuarios como una colección de relaciones normalizadas de diversos grados que varían con el tiempo.

El modelo relacional representa un sistema de bases de datos en un nivel de abstracción un tanto alejado de los detalles de la máquina subyacente, de la misma manera como, por ejemplo, un lenguaje del tipo de PL/1 representa un sistema de programación con un nivel de abstracción un tanto alejado de los detalles de la máquina subyacente. De hecho, el modelo relacional puede considerarse como un lenguaje de programación más bien abstracto, orientado de manera específica hacia las aplicaciones de bases de datos.

En términos tradicionales una relación se asemeja a un archivo, una tupla a un registro, y un atributo a un campo. Pero estas correspondencias son aproximadas, en el mejor de los casos. Una relación no debe considerarse como "solo un archivo", sino más bien como un archivo disciplinado, siendo el resultado de esta disciplina una simplificación considerable de las estructuras de datos con las cuales debe interactuar el usuario, lo cual a su vez simplifica los operadores requeridos para manejar esas estructuras.

Tabla: Es un colección de datos sobre un tema o entidad especifica, como productos o proveedores. La utilización de una tabla diferente para cada tema significa que se almacenan los

datos solo una vez, lo cual hace aumentar la eficacia de las bases de datos y reduce errores de entrada de datos.

Consultas: Es una secuencia de comandos que permite recuperar datos almacenados en tablas de una base de datos, de manera rápida y sencilla.

Informes o Reportes: Un informe es un método eficaz de presentar los datos en formato impreso. Dado que tiene el control sobre el tamaño y el aspecto de todo el informe, puede mostrar la información en la manera que desea verla.

Diseño conceptual:

Una vez recogidos todos los requerimientos, el siguiente paso es crear un esquema conceptual para la base de datos mediante un modelo de datos conceptual de alto nivel.

El esquema conceptual contiene una descripción detallada de los requerimientos de información de los usuarios, y contiene descripciones de los tipos de datos, relaciones entre ellos y restricciones

Nosotros utilizaremos para el diseño de esquemas conceptuales el modelo E-R (Entidad-Relación), que describe los datos cono entidades, vínculos (relaciones) y atributos.

Diseño lógico de la base de datos (transformación de modelo de datos)

El siguiente paso en el proceso de diseño consiste en implementar de hecho la base de datos con un S.G.B.D. comercial, transformando el modelo conceptual al modelo de datos empleados por el S.G.B.D. (jerárquico, red o relacional).

En nuestro módulo haremos la implementación con un S.G.B.D. relacional, por ser el modelo más utilizado por las empresas en la actualidad.

Diseño físico de la base de datos:

En este paso se especifican las estructuras de almacenamiento internas y la organización de los archivos de la base de datos.

¿Qué es una Arquitectura?

Una Arquitectura es un entramado de componentes funcionales que aprovechando diferentes estándares, convenciones, reglas y procesos, permite integrar una amplia gama de productos y servicios informatices de manera que pueden ser utilizados eficazmente dentro de la organización.

Debemos señalar que para seleccionar el modelo de una arquitectura hay que partir del contexto tecnológico y organizativo del momento y que la arquitectura Cliente-Servidor requiere una determinada especialización de cada uno de los diferentes componentes que la integran.

Modelo cliente-servidor:

El procedimiento empleado para intercambiar información en Internet sigue el modelo cliente-servidor.

Los servidores:

Son computadoras donde se almacenan datos y realiza algunas tareas en beneficio de otras aplicaciones llamadas clientes ofreciendo los servicios de archivos.

El cliente:

Es la computadora que realiza la petición al servidor para que este le muestre alguno de los archivos almacenados.

En este modelo, lo único que hace el núcleo es controlar la comunicación entre los clientes y los servidores. Al separar el sistema operativo en partes, cada una de ellas controla una faceta del sistema, como el servicio a ficheros, servicio a procesos, servicio a terminales o servicio a la memoria; cada parte es pequeña y controlable. Además, puesto que todos los servidores se ejecutan como procesos en modo usuario, y no en modo núcleo, no tienen acceso directo al hardware. En consecuencia, si hay un error en el servidor de ficheros éste puede fallar, pero esto no afectará en general a toda la máquina.

Otra de las ventajas del modelo cliente-servidor es su capacidad de adaptación para su uso en sistemas distribuidos. Si un cliente se comunica con un servidor mediante mensajes, el cliente no necesita saber si el mensaje se gestiona de forma local, en su máquina, o si se envía por medio de una red a un servidor en una máquina remota. En lo que respecta al cliente, lo mismo ocurre en ambos casos: se envió una solicitud y se recibió una respuesta. Arquitectura cliente-servidor.

Características del modelo Cliente-Servidor:

- El cliente y el servidor pueden actuar como una sola entidad y tan bien pueden actuar como entidades separadas, realizando actividades o tareas independientes.

- Las funciones de Cliente y Servidor pueden estar en plataformas separadas o en la misma plataforma.
- Un servidor da servicio a múltiples clientes en forma concurrente.
- Cada plataforma puede ser escalable independientemente. Los cambios realizados en las plataformas de los clientes o de los servidores, ya sean por actualización o por reemplazo tecnológico, se realizan de una manera transparente para el usuario final.
- La interrelación entre hardware y software están basados en una infraestructura poderosa, de tal forma que el acceso a los recursos de la red no muestra la complejidad de los diferentes tipos de formatos de datos y de los protocolos.
- Un sistema de servidores realiza múltiples funciones al mismo tiempo representa una imagen de un solo sistema a las estaciones clientes. Esto se logra combinando los recursos de cómputo que se encuentran físicamente separados en un solo sistema lógico, proporcionando de esta manera el servicio más efectivo para el usuario final.

MySQL

MySQL Database Server es la base de datos de código fuente abierto más usada del mundo. Su ingeniosa arquitectura lo hace extremadamente rápido y fácil de personalizar. La extensiva reutilización del código dentro del software y una aproximación mínima para producir características funcionalmente ricas, ha dado lugar a un sistema de administración de la base de datos incomparable en velocidad, compactación, estabilidad y facilidad de despliegue. La exclusiva separación del core server del manejador de tablas, permite funcionar a MYSQL bajo control estricto de transacciones o con acceso a disco no transaccional ultrarrápido.

Qué es MySQL?

MySQL es un sistema de administración de bases de datos. Una base de datos es una colección estructurada de datos. Esta puede ser desde una simple lista de compras a una galería de pinturas o el vasto monto de información en una red corporativa.

Para agregar, accesar y procesar datos guardados en un computador, usted necesita un administrador como MySQL Server.

Dado que los computadores son muy buenos manejando grandes cantidades de información, los administradores de bases de datos juegan un papel central en computación, como aplicaciones independientes o como parte de otras aplicaciones.

MySQL es un sistema de administración relacional de bases de datos. Una base de datos relacional archiva datos en tablas separadas en vez de colocar todos los datos en un gran archivo. Esto

permite velocidad y flexibilidad. Las tablas están conectadas por relaciones definidas que hacen posible combinar datos de diferentes tablas sobre pedido.

MySQL es software de fuente abierta. Fuente abierta significa que es posible para cualquier persona usarlo y modificarlo. Cualquier persona puede bajar el código fuente de MySQL y usarlo sin pagar. Cualquier interesado puede estudiar el código fuente y ajustarlo a sus necesidades.

MySQL usa el GPL (GNU General Public License) para definir que puede hacer y que no puede hacer con el software en diferentes situaciones. Si usted no se ajusta al GLP o requiere introducir código MySQL en aplicaciones comerciales, usted pude comprar una versión comercial licenciada.

Porqué usar MySQL Server?

MySQL Database Server es muy rápido, confiable y fácil de usar. Si eso es lo que usted está buscando, debe tenerlo y usarlo. MySQL Server también tiene un práctico set de características desarrollado en cercana cooperación con nuestros usuarios.

MySQL Server fue desarrollado inicialmente para manejar grandes bases de datos mucho más rápidamente que las soluciones existentes y ha sido usado exitosamente por muchos años en ambientes de producción de alta demanda. A través de constante desarrollo, MySQL Server ofrece hoy una rica variedad de funciones. Su conectividad, velocidad y seguridad hacen a MySQL altamente satisfactorio para accesar bases de datos en Internet.

Las principales características de MySQL

La siguiente lista describe algunas de las características más importantes del software de base de datos MySQL:

- Interioridades y portabilidad
- Escrito en C y en C++
- Probado con un amplio rango de compiladores diferentes
- Funciona en diferentes plataformas.
- Usa GNU Automake, Autoconf, y Libtool para portabilidad.
- APIs disponibles para C, C++, Eiffel, Java, Perl, PHP, Python, Ruby, y Tcl.
- Uso completo de multi-threaded mediante threads del kernel. Pueden usarse fácilmente múltiple CPUs si están disponibles.
- Proporciona sistemas de almacenamientos transaccionales y no transaccionales.
- Usa tablas en disco B-tree (MyISAM) muy rápidas con compresión de índice.

- Relativamente sencillo de añadir otro sistema de almacenamiento. Esto es útil si desea añadir una interfaz SQL para una base de datos propia.
- Un sistema de reserva de memoria muy rápido basado en threads.
- Joins muy rápidos usando un multi-join de un paso optimizado.
- Tablas hash en memoria, que son usadas como tablas temporales.
- Las funciones SQL están implementadas usando una librería altamente optimizada y deben ser tan rápidas como sea posible. Normalmente no hay reserva de memoria tras toda la inicialización para consultas.
- El código MySQL se prueba con Purify (un detector de memoria perdida comercial) así como con Valgrind, una herramienta GPL.
- El servidor está disponible como un programa separado para usar en un entorno de red cliente/servidor. También está disponible como biblioteca y puede ser incrustado (linkado) en aplicaciones autónomas. Dichas aplicaciones pueden usarse por sí mismas o en entornos donde no hay red disponible.

Lenguaje de programación: Es la gramática específica que deberá seguirse para poder dar instrucciones a la computadora; el lenguaje a escoger para desarrollar un programa, depende el objetivo de este. A través de la historia se han desarrollado numerosos lenguajes, por ejemplo el Fortra (traductor de formulas) o el Algol (Lenguaje algorítmico) para el manejo de cálculos matemáticos, el Cobol (lenguaje orientado a negocios comunes) o el Clipper (para manejo de base de datos) así como el Basic (Código internacional estándar para principiantes), el Pascal o el C (para programadores principiantes y creación de programas de aplicación en general). En la época actual, son mas socorridos los lenguajes visuales como el Visual Basic, Visual C++ o el visual Fox Pro, con enormes ventajas y facilidades para la programación de aplicaciones para Windows.

Lenguaje de programación Utilizado:

Microsoft Visual C #.

C #, pronunciado como C Sharp, es actualmente uno de los lenguajes de programación mas populares en Internet. Pero, además, esta disponible para el desarrollo de aplicaciones de propósito general, aplicaciones que muestren una interfaz grafica, aplicaciones para Internet y aplicaciones para móviles.

Se trata de un lenguaje moderno orientado a objetos que permite desarrollar una amplia gama de aplicaciones para la nueva plataforma Microsoft .NET, la cual se caracteriza por proporcionar utilidades y servicios para sacar un provecho total tanto de la informática como de las comunicaciones.

Resumiendo, C #, es un lenguaje seguro y elegante que permite a los desarrolladores construir un amplio rango de aplicaciones seguras y robustas, que se ejecutan sobre .NET Framework. Podemos utilizar C # para crear aplicaciones cliente Windows tradicionales servicios Web XML, componentes distribuidos, aplicaciones cliente-servidor, aplicaciones para base de datos y muchas otras.

Microsoft visual C # 2005 proporciona un editor de código avanzado, diseñadores de interfaces de usuario apropiados, depurador integrado, y muchas otras utilidades para facilitar el desarrollado rápido de aplicaciones basada en la versión 2.0 del lenguaje C # y en .NET FrameWork.

La palabra "Visual hace referencia, desde el lado del diseño, al método que se utiliza para crear la interfaz grafica de usuario si se dispone de la herramienta adecuada(con Microsoft visual estudio se utiliza el ratón para arrastrar y colocar los objetos prefabricados en el lugar deseado dentro de un formulario), y desde el lado de la ejecución al aspecto grafico que toman los objetos cuando se ejecutan el código que los crea, objetos que formaran la interfaz grafica que el usuario de la aplicación utiliza para acceder a los servicios que esta ofrece.

Visual Studio es un conjunto completo de herramientas de desarrollo para construir aplicaciones Web, servicios Web, aplicaciones Windows o de escritorio y aplicaciones para dispositivos móviles. El entorno de desarrollo integrado que ofrece esta plataforma con todas sus herramientas y con la biblioteca de clases .NET framework, es compartido en su totalidad por visual Basic, visual C++, y visual C #, permitiendo así crear con facilidad soluciones en las que intervienen varios lenguajes en las que el diseño se realiza separadamente respecto a la programación.

Visual Studio permite diseñar la interfaz grafica de una aplicaron de manera visual, sin mas que arrastrar con el ratón los controles que necesitemos sobre la ventana destino de los mismos. Una rejilla mostrada sobre la ventana nos ayudara a colocar estos controles y a darles el tamaño adecuado y una página de propiedades nos ayudara a modificar los valores de las propiedades de cada uno de los controles. Todo lo expuesto, lo realizaremos sin tener que escribir ni una sola línea de código. Después, un editor de código inteligente nos ayudara a escribir el código necesario y detectara los errores sintácticos que introduzcamos y un depurador nos ayudara a poner a punto nuestra aplicación cuando lo necesitemos.

Características de C#.

Con la idea de que los programadores más experimentados puedan obtener una visión general del lenguaje, a continuación se recoge de manera resumida las principales características de C# Alguna de las características aquí señaladas no son exactamente propias del lenguaje sino de la

plataforma .NET en general, y si aquí se comentan es porque tienen una repercusión directa en el lenguaje:

- **Sencillez:** C# elimina muchos elementos que otros lenguajes incluyen y que son innecesarios en .NET. Por ejemplo:

    1. El código escrito en C# es autocontenido, lo que significa que no necesita de ficheros adicionales al propio fuente tales como ficheros de cabecera o ficheros IDL.

    2. El tamaño de los tipos de datos básicos es fijo e independiente del compilador, sistema operativo o máquina para quienes se compile (no como en C++), lo que facilita la portabilidad del código.

    3. No se incluyen elementos poco útiles de lenguajes como C++ tales como macros, herencia múltiple o la necesidad de un operador diferente del punto (.) acceder a miembros de espacios de nombres (::).

- **Modernidad: C#** incorpora en el propio lenguaje elementos que a lo largo de los años ha ido demostrándose son muy útiles para el desarrollo de aplicaciones y que en otros lenguajes como Java o C++ hay que simular, como un tipo básico decimal que permita realizar operaciones de alta precisión con reales de 128 bits (muy útil en el mundo financiero), la inclusión de una instrucción foreach que permita recorrer colecciones con facilidad y es ampliable a tipos definidos por el usuario, la inclusión de un tipo básico string para representar cadenas o la distinción de un tipo bool específico para representar valores lógicos.

- **Orientación a objetos:** Como todo lenguaje de programación de propósito general actual, C# es un lenguaje orientado a objetos, aunque eso es más bien una característica del CTS que de C#. Una diferencia de este enfoque orientado a objetos respecto al de otros lenguajes como C++ es que el de C# es más puro en tanto que no admiten ni funciones ni variables globales sino que todo el código y datos han de definirse dentro de definiciones de tipos de datos, lo que reduce problemas por conflictos de nombres y facilita la legibilidad del código.

C# soporta todas las características propias del paradigma de programación orientada a objetos: encapsulación, herencia y polimorfismo.

En lo referente a la encapsulación es importante señalar que aparte de los típicos modificadores public, private y protected, C# añade un cuarto modificador llamado internal, que puede combinarse con protected e indica que al elemento a cuya definición precede sólo puede accederse desde su mismo ensamblado.

Respecto a la herencia -a diferencia de C++ y al igual que Java- C# sólo admite herencia simple de clases ya que la múltiple provoca más quebraderos de cabeza que facilidades y en la mayoría de los casos su utilidad puede ser simulada con facilidad mediante herencia

múltiple de interfaces. De todos modos, esto vuelve a ser más bien una característica propia del CTS que de C#.

Por otro lado y a diferencia de Java, en C# se ha optado por hacer que todos los métodos sean por defecto sellados y que los redefinibles hayan de marcarse con el modificador virtual (como en C++), lo que permite evitar errores derivados de redefiniciones accidentales. Además, un efecto secundario de esto es que las llamadas a los métodos serán más eficientes por defecto al no tenerse que buscar en la tabla de funciones virtuales la implementación de los mismos a la que se ha de llamar. Otro efecto secundario es que permite que las llamadas a los métodos virtuales se puedan hacer más eficientemente al contribuir a que el tamaño de dicha tabla se reduzca.

- **Orientación a componentes:** La propia sintaxis de C# incluye elementos propios del diseño de componentes que otros lenguajes tienen que simular mediante construcciones más o menos complejas. Es decir, la sintaxis de C# permite definir cómodamente propiedades (similares a campos de acceso controlado), eventos (asociación controlada de funciones de respuesta a notificaciones) o atributos (información sobre un tipo o sus miembros).

- **Gestión automática de memoria:** Como ya se comentó, todo lenguaje de .NET tiene a su disposición el recolector de basura del CLR. Esto tiene el efecto en el lenguaje de que no es necesario incluir instrucciones de destrucción de objetos. Sin embargo, dado que la destrucción de los objetos a través del recolector de basura es indeterminista y sólo se realiza cuando éste se active –ya sea por falta de memoria, finalización de la aplicación o solicitud explícita en el fuente-, C# también proporciona un mecanismo de liberación de recursos determinista a través de la instrucción **using.**

- **Seguridad de tipos:** C# incluye mecanismos que permiten asegurar que los accesos a tipos de datos siempre se realicen correctamente, lo que permite evita que se produzcan errores difíciles de detectar por acceso a memoria no perteneciente a ningún objeto y es especialmente necesario en un entorno gestionado por un recolector de basura. Para ello se toman medidas del tipo:

1. Sólo se admiten conversiones entre tipos compatibles. Esto es, entre un tipo y antecesores suyos, entre tipos para los que explícitamente se haya definido un operador de conversión, y entre un tipo y un tipo hijo suyo del que un objeto del primero almacenase una referencia del segundo (downcasting) Obviamente, lo último sólo puede comprobarlo en tiempo de ejecución el CLR y no el compilador, por lo que en realidad el CLR y el compilador colaboran para asegurar la corrección de las conversiones.

2. No se pueden usar variables no inicializadas. El compilador da a los campos un valor por defecto consistente en ponerlos a cero y controla mediante análisis del flujo de control del fuente que no se lea ninguna variable local sin que se le haya asignado previamente algún valor.

3. Se comprueba que todo acceso a los elementos de una tabla se realice con índices que se encuentren dentro del rango de la misma.

4. Se puede controlar la producción de desbordamientos en operaciones aritméticas, informándose de ello con una excepción cuando ocurra. Sin embargo, para conseguirse un mayor rendimiento en la aritmética estas comprobaciones no se hacen por defecto al operar con variables sino sólo con constantes (se pueden detectar en tiempo de compilación).

5. A diferencia de Java, C# incluye delegados, que son similares a los punteros a funciones de C++ pero siguen un enfoque orientado a objetos, pueden almacenar referencias a varios métodos simultáneamente, y se comprueba que los métodos a los que apunten tengan parámetros y valor de retorno del tipo indicado al definirlos.

6. Pueden definirse métodos que admitan un número indefinido de parámetros de un cierto tipo, y a diferencia lenguajes como C/C++, en C# siempre se comprueba que los valores que se les pasen en cada llamada sean de los tipos apropiados.

- **Instrucciones seguras:** Para evitar errores muy comunes, en C# se han impuesto una serie de restricciones en el uso de las instrucciones de control más comunes. Por ejemplo, la guarda de toda condición ha de ser una expresión condicional y no aritmética, con lo que se evitan errores por confusión del operador de igualdad (==) con el de asignación (=); y todo caso de un switch ha de terminar en un break o goto que indique cuál es la siguiente acción a realizar, lo que evita la ejecución accidental de casos y facilita su reordenación.

- **Sistema de tipos unificado:** A diferencia de C++, en C# todos los tipos de datos que se definan siempre derivarán, aunque sea de manera implícita, de una clase base común llamada System.Object, por lo que dispondrán de todos los miembros definidos en ésta clase (es decir, serán "objetos")

  A diferencia de Java, en C# esto también es aplicable a los tipos de datos básicos además, para conseguir que ello no tenga una repercusión negativa en su nivel de rendimiento, se ha incluido un mecanismo transparente de boxing y unboxing con el que se consigue que sólo sean tratados como objetos cuando la situación lo requiera, y mientras tanto puede aplicárseles optimizaciones específicas.

  El hecho de que todos los tipos del lenguaje deriven de una clase común facilita enormemente el diseño de colecciones genéricas que puedan almacenar objetos de cualquier tipo.

- **Extensibilidad de tipos básicos:** C# permite definir, a través de estructuras, tipos de datos para los que se apliquen las mismas optimizaciones que para los tipos de datos básicos. Es decir, que se puedan almacenar directamente en pila (luego su creación, destrucción y acceso serán más rápidos) y se asignen por valor y no por referencia. Para conseguir que lo último no tenga efectos negativos al pasar estructuras como parámetros de métodos, se da la posibilidad de pasar referencias a pila a través del modificador de parámetro ref.

- **Extensibilidad de operadores:** Para facilitar la legibilidad del código y conseguir que los nuevos tipos de datos básicos que se definan a través de las estructuras estén al mismo nivel que los básicos predefinidos en el lenguaje, al igual que C++ y a diferencia de Java, C# permite redefinir el significado de la mayoría de los operadores -incluidos los de conversión, tanto para conversiones implícitas como explícitas- cuando se apliquen a diferentes tipos de objetos.

  Las redefiniciones de operadores se hacen de manera inteligente, de modo que a partir de una única definición de los operadores ++ y -- el compilador puede deducir automáticamente como ejecutarlos de manera prefijas y postfija; y definiendo operadores simples (como +), el compilador deduce cómo aplicar su versión de asignación compuesta

  (+=) Además, para asegurar la consistencia, el compilador vigila que los operadores con opuesto siempre se redefinan por parejas (por ejemplo, si se redefine ==, también hay que redefinir!=).

  También se da la posibilidad, a través del concepto de indizador, de redefinir el significado del operador [] para los tipos de dato definidos por el usuario, con lo que se consigue que se pueda acceder al mismo como si fuese una tabla. Esto es muy útil para trabajar con tipos que actúen como colecciones de objetos.

- **Extensibilidad de modificadores**: C# ofrece, a través del concepto de atributos, la posibilidad de añadir a los metadatos del módulo resultante de la compilación de cualquier fuente información adicional a la generada por el compilador que luego podrá ser consultada en tiempo ejecución a través de la librería de reflexión de .NET . Esto, que más bien es una característica propia de la plataforma .NET y no de C#, puede usarse como un mecanismo para definir nuevos modificadores.

- **Versionable:** C# incluye una política de versionado que permite crear nuevas versiones de tipos sin temor a que la introducción de nuevos miembros provoquen errores difíciles de detectar en tipos hijos previamente desarrollados y ya extendidos con miembros de igual nombre a los recién introducidos.

  Si una clase introduce un nuevo método cuyas redefiniciones deban seguir la regla de llamar a la versión de su padre en algún punto de su código, difícilmente seguirían esta regla miembros de su misma signatura definidos en clases hijas previamente a la definición del mismo en la clase padre; o si introduce un nuevo campo con el mismo nombre que algún método de una clase hija, la clase hija dejará de funcionar. Para evitar que esto ocurra, en C# se toman dos medidas:

1. Se obliga a que toda redefinición deba incluir el modificador override, con lo que la versión de la clase hija nunca sería considerada como una redefinición de la versión de miembro en la clase padre ya que no incluiría override. Para evitar que por accidente un programador incluya este modificador, sólo se permite incluirlo en miembros que tengan la misma signatura que miembros marcados como redefinibles mediante el modificador virtual. Así además se evita el error tan frecuente en Java de creerse haber redefinido un miembro, pues si el miembro con override no existe en la clase padre se producirá un error de compilación.

2. Si no se considera redefinición, entonces se considera que lo que se desea es ocultar el método de la clase padre, de modo que para la clase hija sea como si nunca hubiese existido. El compilador avisará de esta decisión a través de un mensaje de aviso que puede suprimirse incluyendo el modificador new en la definición del miembro en la clase hija para así indicarle explícitamente la intención de ocultación.

- **Eficiente:** En principio, en C# todo el código incluye numerosas restricciones para asegurar su seguridad y no permite el uso de punteros. Sin embargo, y a diferencia de Java, en C# es posible saltarse dichas restricciones manipulando objetos a través de punteros. Para ello basta marcar regiones de código como inseguras (modificador unsafe) y podrán usarse en ellas punteros de forma similar a cómo se hace en C++, lo que puede resultar vital para situaciones donde se necesite una eficiencia y velocidad procesamiento muy grandes.

- **Compatible:** Para facilitar la migración de programadores, C# no sólo mantiene una sintaxis muy similar a C, C++ o Java que permite incluir directamente en código escrito en C# fragmentos de código escrito en estos lenguajes, sino que el CLR también ofrece, a través de los llamados Platform Invocation Services (PInvoke), la posibilidad de acceder a código nativo escrito como funciones sueltas no orientadas a objetos tales como las DLLs de la API Win32. Nótese que la capacidad de usar punteros en código inseguro permite que se pueda acceder con facilidad a este tipo de funciones, ya que éstas muchas veces esperan recibir o devuelven punteros.

  También es posible acceder desde código escrito en C# a objetos COM. Para facilitar esto, el .NET Framework SDK incluye una herramientas llamadas tlbimp y regasm mediante las que es posible generar automáticamente clases proxy que permitan, respectivamente, usar objetos COM desde .NET como si de objetos .NET se tratase y registrar objetos .NET para su uso desde COM.

Finalmente, también se da la posibilidad de usar controles ActiveX desde código .NET y viceversa. Para lo primero se utiliza la utilidad aximp, mientras que para lo segundo se usa la ya mencionada regasm.

VI-Diseño Metodológico.

Para llevar a cabo el desarrollo de nuestro trabajo utilizaremos el Ciclo de Vida Clásico del software o modelo de cascada, ya que este modelo nos permite realizar cada una de las etapas del software de forma independiente.

Las etapas que seguiremos para el desarrollo de nuestro sistema de Registro Académico son las siguientes:

Ingeniería del sistema: El trabajo del desarrollo del software inicia con la ingeniería del mantenimiento del sistema y consiste en hacer un estudio del ámbito del problema, estableciendo los requisitos generales a un nivel superior, a demás se aplican las técnicas de elaboración y evaluación del proyecto.

Análisis del sistema: Los desarrolladores ponen su atención en la información que manejará el software, las funciones que ejecutará el rendimiento y las interfaces hombre maquina requerida. Con resultado del análisis del sistema se obtiene un documento llamado ERS (Especificación de Requisitos Software).

Diseño: Es un proceso de varios pasos que se centra en cuatro aspectos del software:

1. Estructura de Datos.
2. Arquitectura del software.
3. Detalle procedimental.
4. Caracterización de las interfaces.

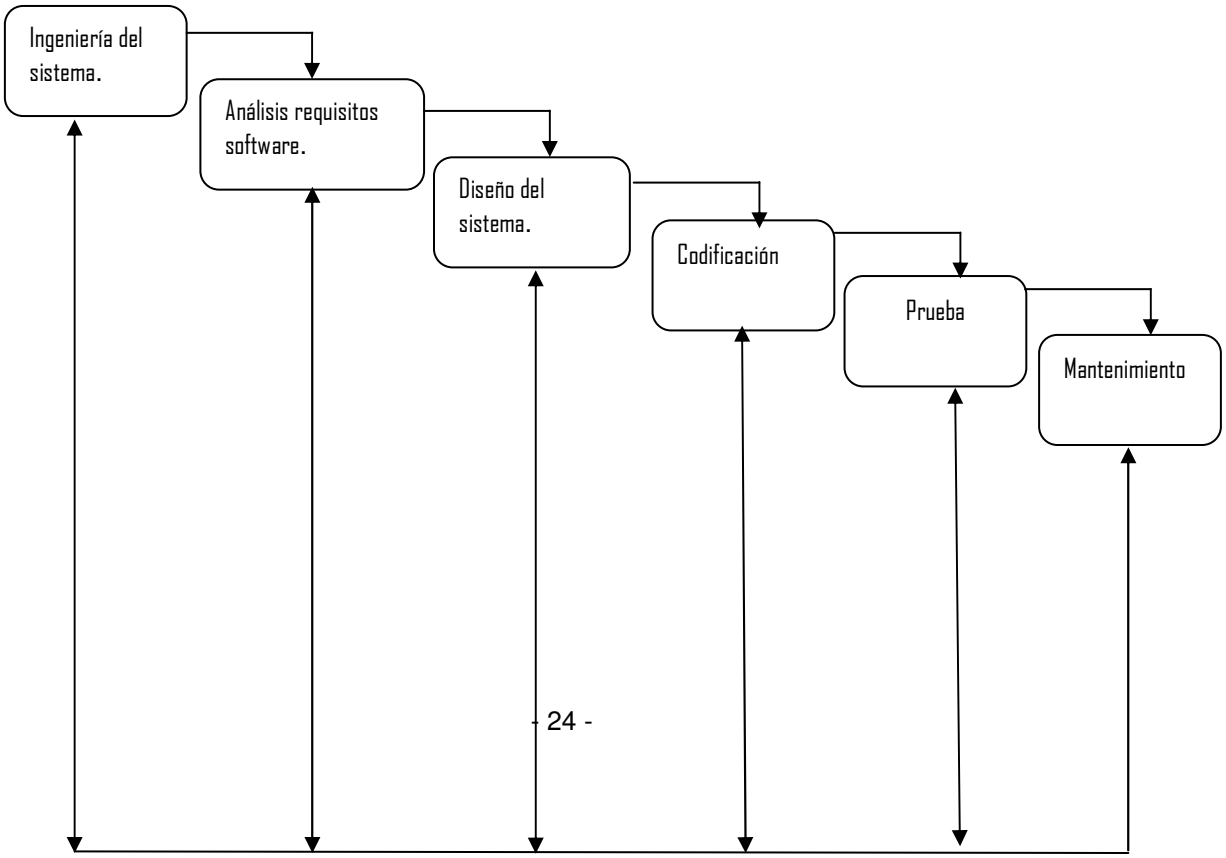
Codificación: consiste en la traducción de un diseño en un formato legible para la maquina.

Pruebas: consiste en asegurar que las entradas del sistema produzcan los resultados deseados.

Mantenimiento: Actividad de mantener el sistema de forma fiable.

Modelo en cascada:

Descompone el proceso de desarrollo en diferentes procesos, este genera una salida que será entrada del siguiente proceso. Este modelo comienza con el nivel de ingeniería del sistema que facilita al gestor controlar el proceso de desarrollo de software. Así va progresando a través del análisis, diseño, codificación, prueba y mantenimiento.



El desarrollo de nuestro sistema esta basado en la utilización de fase o etapas que surgen como la combinación del método del ciclo de vida clásico y el método de canales estructurado, siendo estas las siguientes:

Planificación:

Esta fase permite obtener un plan para la utilización de sistema de información en una empresa que este acorde con las necesidades operativa como de dirección de la misma. Dicha fase permite obtener una visión general de la empresa, sus funciones, objetivos, datos y necesidades de información.

Esta actividad consta de tres etapas:

1. **Aclaración de solicitud:** La solicitud del proyecto debe examinarse para determinar con precisión lo que el solicitante desea, es decir, la solicitud del proyecto debe estar claramente planteada.

2. **Estudio de factibilidad:** Es el análisis de un problema para determinar si puede resuelto efectivamente; a través de una investigación preliminar la cual consta de tres aspectos.

    - Factibilidad técnica.
    - Factibilidad económica.
    - Factibilidad operacional.

3. **Aprobación de solicitud**: No todos los proyectos solicitados son deseables o factibles; sin embargo, aquellos que son deseables deben incorporarse en los planes de los analistas

Análisis de los requerimientos del sistema.

Esta fase consiste en realizar el estudio de un problema antes de realizar una acción.
El aspecto fundamental es comprender todas las facetas importantes de la empresa que se encuentra bajo estudio. Para realizar este estudio se trabaja con los empleados de la empresa para dar respuesta a las siguientes preguntas claves:

1. ¿Qué es lo que se hace?
2. ¿Cómo se hace?
3. ¿Con que frecuencia se presenta?
4. ¿Qué tan grande es el volumen de decisiones?
5. ¿Cuál es el grado de eficiencia con el que se efectuan las tareas?
6. ¿Existe algún problema?
7. si existe algún problema ¿Qué tan serio es?
8. si existe un problema, ¿Cuál es la causa que lo origina?

Conforme se reúnen los detalles, se estudian los datos sobre los requerimientos con la finalidad de identificar las características que debe tener el sistema. El producto que se obtiene es primeramente una especificación de requisitos software (ERS), luego el análisis de flujo de datos, que abarca la construcción de los diagramas de flujos (DFD) y el diccionario de datos; finalizando con la elaboración del modelo conceptual de datos (MCD).

1. **Especificación de requisitos software(ERS):** Es el establecimiento conciso de un conjunto de requisitos que deben ser satisfechos por un producto o proceso, indicando siempre que sea adecuado, el procedimiento mediante el cual se puede determinar si se han logrado satisfacer los requisitos.

Estructura de un ERS.

1. Introducción.

1.1 Propósito.
1.2 Alcance.
1.3 Definiciones, acrónimos y abreviaturas.
1.4 Referencias.
1.5 Visión general.

2. Descripción general.

2.1 Relaciones del producto.
2.2 Funciones del producto.
2.3 Características del usuario.
2.4 Restricciones generales.
2.5 Suposiciones y dependencias.

3. Requisitos específicos.

3.1 requisitos funcionales.
3.1.1 Nombre de la función.
3.1.1.1 Especificación.
3.1.1.1.1 Introducción.
3.1.1.1.2 Entradas.
3.1.1.1.3 Proceso.
3.1.1.1.4 Salidas.

3.2 Interfaces Externas.
3.2.1 Interfaces de usuario.
3.2.2 Interfaces de hardware.

3.2.3 Interfaces de software.
3.2.4 Interfaces de comunicación.

3.3 Requisitos de funcionamiento.
3.3.1 Requisitos estáticos.
3.3.2 Requisitos dinámicos.

3.4 Restricciones de diseño.

3.5 Atributos.
3.5.1 Seguridad.
3.5.2 Mantenimiento.
3.5.3 Ayuda.

3.6 Otros requisitos.
3.6.1 Bases de datos.
3.6.2 Operaciones.

2. **Análisis de flujo de datos**: Examina el empleo de los datos para llevar a cabo procesos específicos de la empresa el cual esta compuesto de:

a) **Diagrama de flujos de datos (DFD)**: Es la descripción grafica del sistema por el método de análisis estructurado; se sigue un proceso descendente (Top-down). Cada proceso puede desglosarse en diagramas de flujo de datos cada vez mas detallados, hasta obtener suficientes detalles que permitan comprender en su totalidad la parte del sistema que se encuentra bajo investigación.

Símbolos utilizados en los DFD:

Flujo de datos:

Procesos:

Almacén de datos:

Entidad Externa:

b) **Diccionario de datos:** En el se encuentran definidos de forma detallada todos los elementos del sistema (flujos de datos, procesos y almacén de datos), además guarda los detalles y descripciones de todos los elementos. El diccionario se desarrolla durante el análisis de flujo de datos.

Importancia:

- ✓ Para manejar los detalles en sistemas grandes.
- ✓ Para comunicar un significado común para todos los elementos del sistema.
- ✓ Para documentar las características del sistema.
- ✓ Para facilitar análisis de los detalles con la finalidad de evaluar las características y determinar donde efectuar cambios en el sistema.
- ✓ Localizar errores y omisiones en el sistema.

Pasos para construir un diccionario de datos:

- ✓ Incorporar el proceso.
- ✓ Catalogar los flujos básicos de datos.
- ✓ Describir la estructura de los datos.
- ✓ Desglosar la estructura de los datos.

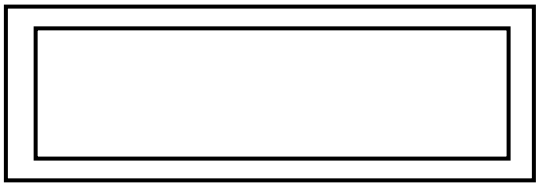
c) **Modelo conceptual de datos (MCD):** este modelo permite representar los datos con independencias de decisiones técnicas, encargándose de las descripciones de los mismos. Este modelo esta formado por un nivel de gran importancia como lo es el nivel conceptual, existiendo diferentes modelos de datos como: Modelo Entidad Relación (E-R), siendo este muy utilizado, el Modelo Orientado a Objeto, el Modelo infologico, el Modelo Binario, etc.

Nivel conceptual: describe que datos son realmente almacenados en la base de datos y las relaciones que existen entre ellos.

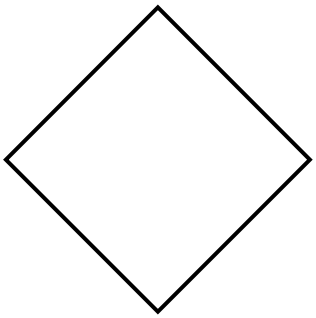
El MCD se hace con la ayuda de cuatro símbolos:

Objeto o individuo:

Objeto o individuo débil:



Relación:



Línea de cardinalidad:

➢ Diseño del sistema.

Esta fase produce los detalles que establecen la forma en que el sistema cumplirá con los requisitos identificados durante la fase de análisis. Es llamado diseño lógico y consiste en:

- El diseño de los reportes y demás salidas que debe producir el sistema.
- La identificación de los datos de entrada, aquellos que serán calculados y los que deben ser almacenados. Así mismo se escriben con detalle los procedimientos de calculo y los individuales; selección de estructuras de los archivos y dispositivos de almacenamiento.

Las especificaciones de diseño se representan a través de diagramas, tablas y símbolos especiales, todo esto para comenzar la fase de desarrollo de software.

La fase de diseño implica trabajar con cuatro tipos de diseños como son:

- Diseño de datos.
- Diseño de los diagramas de dialogo.
- Diseño de interfaces.
- Diseño procedimetal.

➢ Desarrollo del software.

Esta fase consiste en:

- Codificación y prueba de programas en base a las especificaciones del diseño.

- Documentación para probar el programa y llevar a cabo el mantenimiento una vez que la aplicación se encuentre instalada.

Para construir el sistema diseñado se pueden utilizar lenguajes de programación de alto nivel, lenguajes de cuarta generación y gestores de bases de datos, que utilizan generadores de informe, menús, pantallas, etc. Los más utilizados son gestores de bases de datos como: Access, foxpro, oracle, DataBase, Visual FoxPro, SQL Server, etc.

En este documento comentaremos sobre MySQL Server, Visual C Sharp y las demás herramientas utilizadas para el desarrollo de nuestra aplicación.

➢ Prueba del sistema.

Durante esta fase se emplea de manera experimental para asegurarse de que el software no tenga fallas, es decir que funciona de acuerdo con las especificaciones y en la forma que los usuarios esperan de que lo hagan.

➢ Implantación y evaluación.

La implantación es el proceso de verificar e instalar el nuevo equipo, entrenar a los usuarios, instalar la aplicación y construir todos los archivos de datos necesarios para utilizarlos.
La evaluación de un sistema se lleva a cabo para identificar puntos débiles y fuertes; ocurre a lo largo de las siguientes dimensiones:

- Evaluación operacional.
- Impacto organizacional.
- Opinión de los administradores.
- Desempeño de desarrollo.

VII-Especificación de Requisitos del Software (ERS).

Especificación de Requisitos del Software (ERS).

Estructura de un ERS.

1. Introducción:

1.1 Propósito.

Definición del conjunto de especificaciones de requisitos software que debe cumplir la aplicación para automatizar los procesos del área de registro académico de la facultad de ciencias económicas y empresariales.

Este documento se dirige al encargado del área de registro académico y a los usuarios finales que deberán estudiarlo para su aprobación o desacuerdo antes de abordar la fase de análisis.

1.2 Alcance.

El nombre con el que se conocerá a esta aplicación será:

Automatización de procesos del Área de Registro Académico, de la Facultad de Ciencias Económicas y Empresariales UNAN-LEON.

El producto realizará las siguientes funciones:

1) Registrar datos del estudiante.
2) Buscar estudiante por número de carnet.

3) Listados de estudiante.
4) Listados de asignaturas con su código.
5) Registrar las notas del estudiante por asignatura.
6) Consultar nota del estudiante.
7) Modificar las notas del estudiante.
8) Modificar los datos del estudiante.
9) Actualizar e insertar en tabla carrera.
10) Actualizar e insertar en tabla asignaturas.
11) Eliminar estudiante.
12) Registro de profesores.
13) Añadir profesor.
14) Eliminar profesor.
15) Consultar los índices académicos por carrera.
16) Consultar índice académico por año.
17) Imprimir Reporte de notas registradas.
18) Imprimir Reporte semestral de las notas del estudiante.
19) Imprimir Reporte anual de las notas del estudiante.
20) Constancia de alumno activo.
21) Consultar los estudiantes que reprobaron por asignatura.
22) Consultar los estudiantes que reprobaron por carrera.
23) Consultar los estudiantes con becas externas.
24) Consultar los estudiantes con becas internas.
25) Consultar los estudiantes sin beca.
26) Consultar los estudiantes con exoneración de matricula.
27) Respaldo de la base de datos.
28) Crear contraseña de usuario.

1.3 28) Definiciones acrónimos y abreviatura.

1. **Registrar Estudiante:** acción de ingresar los datos del estudiante a la base de datos. De aquí en adelante **R_EST.**

2. **Buscar Estudiante por Numero de Carnet:** Acción de buscar un estudiante por medio de su numero de carnet. De aquí en adelante **B_EST_NUM_CARNET.**

3. **Listados de estudiantes:** acción de visualizar el listado de estudiantes de una carrera. De aquí en adelante **LISTADOS_EST.**

4. **Listados de Asignaturas con su Código:** acción de visualizar el listado de asignaturas. De aquí en adelante **LISTADOS_ASIG_COD.**

5. **Registrar las Notas del Estudiante por Asignatura:** acción de añadir las notas del estudiante por asignatura. De aquí en adelante **R_N_EST_ASIG.**

6. **Consultar Nota del Estudiante:** acción de presentar la nota de un determinado estudiante. De aquí en adelante **CTAR_N_EST.**

7. **Modificar Nota del Estudiante:** acción de corregir la nota del estudiante. De aquí en adelante **MCAR_N_EST.**

8. **Modificar Datos del Estudiante:** acción de corregir los datos del estudiante. Le llamaremos **MCAR_DTOS_EST.**

9. **Actualizar e Insertar en Tabla Carrera:** acción de corregir e insertar una carrera. Le llamaremos **ACTUAL_INSERT_CARRERA.**

10. **Actualizar e Insertar en Tabla Asignaturas:** acción de corregir e insertar asignaturas. Le llamaremos **ACTUAL_INSERT_ASIG.**

11. **Eliminar Estudiante:** acción de eliminar un determinado estudiante. Le llamaremos **ELIMINAR_EST.**

12. **Registro de Profesores:** acción de añadir los datos del profesor. A lo largo del ERS se denominara **R_PROF**.

13. **Añadir Nuevo Profesor:** acción de añadir un nuevo profesor. De aquí en adelante **AÑADIR _ PROF**.

14. **Eliminar Profesor:** acción de borrar un profesor. De aquí en adelante **ELIMINAR _ PROF.**

15. **Consultar los Mejores Índices Académicos por Carrera**: acción de presentar los mejores índices académicos por carrera. A lo largo del ERS se le denominara **CTAR_MEJORES_IA_CARRERA.**

16. **Consultar Índice Académico por Año:** acción de presentar el promedio de notas por semestre. A lo largo del ERS se le denominara **CTAR_IA_AÑO.**

Reporte: acción de mostrar un documento donde se vea reflejado los contenidos de interés de una determinada entidad. En nuestra tesis tendrá la misma característica que del menú consultar con la única diferencia que este se imprimirá de aquí en adelante como **REPORTE_X,** donde X puede ser:

17. **Imprimir Reporte de Notas Registradas:**
    IMPRIMIR_REPORTE_N_REGIST.

18. **Imprimir Reporte Semestral de las Notas del Estudiante:**
    IMPRIMIR _REPORTE_S_N_EST.

19. **Imprimir Reporte Anual de las Notas del Estudiante:**
    IMPRIMIR_REPORTE_A_N_EST.

20. **Constancia de Alumno Activo:** acción de imprimir una constancia sobre la carrera y año que cursa el estudiante. De aquí en adelante**. CTCIA_EST_ACTIVO.**

21. **Consultar los Estudiantes que Reprobaron por Asignatura:** acción de consultar todos los estudiantes que reprobaron una determinada asignatura. De aquí en adelante se le llamara **CTAR_EST_REP_ASIG.**

22. **Consultar los Estudiantes que Reprobaron por Carrera:** acción de consultar todos los estudiantes que reprobaron por carrera. De aquí en adelante se le llamara **CTAR_EST_REP_CARRERA**.

23. **Consultar los Estudiantes con Becas Externas:** acción de presentar los estudiantes que le asignaron becas externas se le llamará **CTAR_EST_BECAS_EX.**

24. **Consultar los Estudiantes con Becas Internas:** acción de presentar los estudiantes que le asignaron becas internas se le llamará **CTAR_EST_BECAS_IN.**

25. **Consultar los Estudiantes sin Becas**: acción de presentar los estudiantes que no tienen beca. Se le llamara **CTAR_EST_SIN_BECAS.**

26. **Consultar los Estudiantes con Exoneración de Matricula:** acción de presentar aquellos estudiantes que obtuvieron exoneración de matricula. Se le llamara **CTAR_EST_EXO_MAT.**

27. **Respaldo de la Base de Datos:** acción de respaldar la base de datos**.** De aquí en adelante **RESPALDO_BD.**

28. **Cambiar Contraseña de Usuario:** acción de modificar la contraseña de usuario. De aquí en adelante **CBIAR_ CONTRASEÑA_USR**.

1.4 Referencias:

Documento donde se tomo información base para el análisis y realización del software:

1-) Direcciones en Internet.

a) www.mysql.com
b) wwww.dev.mysql.com/doc/
c) www.mysql-hispano.org
d) wwww.dev.mysql.com/doc/refmam/5.0/es/manual-info.html
e) wwww.dev.mysql.com/doc/refmam/5.0/es/mysql-databadse-administration.html
f) http://window.to/concepcion.com.do
g) http://es.wikipedia.org/wiki/Base_de_datos

2-) Bibliografía:

a) Monografía sistema automatizado de cuentas por cobrar.
b) Monografía
c) Enciclopedia de Microsoft Visual C Sharp de Francisco Javier Ceballos.
d) Análisis y Diseño de Sistemas, Henry f. korth & Abraham Silberschatz, Segunda Edición, Editorial MC Graw Hill

1.5 Visión general:

Primeramente se realizara una descripción general del producto que se desea desarrollar para pasar posteriormente a estudiar cada uno de los requisitos específicos individualmente.

2. Descripción general:

2.1 Relaciones del producto:

El equipo con el que se desarrollará el producto final es:

- Computadora Compaq Presario V2615LA
- 512 de RAM.
- 80 GB de disco duro.
- La instalación inicial constara de 3 terminales y una impresora. Todas conectadas en red.

Cabe destacar lo siguiente:

- La captura de los datos se realizara de forma interactiva por pantalla.
- El sistema deberá ser implementado en una red de área local por lo tanto se hará uso de terminales que tengan instaladas tarjetas de red.

2.2 Funciones del Producto:

El producto software debe contener todas las tareas que realiza manualmente el personal del área de registro académico, de forma diaria. Estas son:

1. Cuando se necesite R_EST el usuario deberá ingresar en el ordenador los datos que se guardará en la base de datos "REGISTRO_ACADEMICO2", que se utilizará posteriormente.

2. Cuando se necesite B_EST_NUM_CARNET el usuario deberá introducir el numero de carnet del estudiante.

3. Cuando el usuario necesite visualizar LISTADOS_EST deberá introducir los datos en el ordenador.

4. Cuando se necesite LISTADOS_ASIG_COD el usuario podrá visualizarlas en el ordenador.

5. . Una vez R_EST el usuario si así lo desea también podrá R_N_EST_ASIG que se guardará en la base de datos y se utilizara posteriormente.

6. Cuando se reciba la solicitud de un estudiante de conocer su nota se podrá CTAR_N_EST introduciendo por el usuario los datos del mismo.

7. . Al producirse un error en una determinada nota el usuario podrá MCAR_N_EST.

8. . Al producirse un error en R_EST el usuario podrá MCAR_DTOS_EST.

9. Al producirse un error o se quiera insertar en tabla carrera el usuario podrá ACTUAL_INSERT_CARRERA.

10. Al producirse un error o se quiera insertar en tabla asignaturas el usuario podrá ACTUAL_INSERT_ASIG.

11. Cuando el usuario quiera ELIMINAR_EST lo hará pero tiene que estar R_EST.

12. . Cuando se necesite R_PROF el usuario deberá ingresar en el ordenador los datos que se guardaran en la base de datos que se utilizara posteriormente.

13. Cuando se necesite AÑADIR _ PROF el usuario deberá introducir los datos necesarios.

14. Cuando se requiera ELIMINAR_PROF este tiene que estar registrado posteriormente.

15. . Una vez introducido el R_N_EST_ASIG se podrá CTAR_MEJORES_IA_CARRERA ingresando el usuario los datos necesarios del estudiante.

16. . Una vez introducido el R_N_EST_ASIG se podrá CTAR_IA_AÑO ingresando el usuario los datos necesarios.

17. Una vez introducido R_N_EST_ASIG el usuario deberá introducir los datos necesarios para obtener el IMPRIMIR_REPORTE_N_REGIST que se imprimirá posteriormente en papel

18. . Una vez introducido R_N_EST_ASIG el usuario deberá introducir los datos necesarios para obtener el IMPRIMIR_REPORTE_S_N_EST que se imprimirá posteriormente en papel.

19. . Una vez introducido R_N_EST_ASIG el usuario deberá introducir los datos necesarios para obtener el IMPRIMIR_REPORTE_A_N_EST que se imprimirá posteriormente en papel.

20. Cuando se reciba la solicitud de un estudiante de una CTCIA_EST_ACTIVO se debe introducir los datos necesarios en el ordenador y se podrá imprimir.

21. Cuando el usuario necesite CTAR_EST_REP_ASIG este deberá introducir los datos necesarios en el ordenador.

22. . Cuando el usuario necesite CTAR_EST_REP_CARRERA este deberá introducir los datos necesarios en el ordenador.

23. . Cuando el usuario necesite CTAR_EST_BECAS_EX debe introducir los datos en el ordenador.

24. . Cuando el usuario necesite CTAR_EST_BECAS_IN debe introducir los datos en el ordenador.

25. Cuando el usuario necesite CTAR_EST_SIN_BECAS se deberá introducir los datos en el ordenador.

26. . Cuando el usuario necesite CTAR_EST_EXO_MAT debe introducir los datos en el ordenador.

27. . Cuando el usuario necesite realizar un RESPALDO_BD.

28. . Cuando el usuario necesite CBIAR_CONTRASEÑA_USR deberá introducir su contraseña actual, a continuación su nueva contraseña y confirmación de esta.

2.3 Características del usuario:

El usuario de nuestro sistema deberá tener conocimiento acerca del manejo básico de un sistema computarizado.

Restricciones generales:

El lenguaje de programación utilizado será visual C # 2005.

El área de registro académico que utilizará nuestro sistema deberá tener como requisitos mínimos:

- Computadoras de procesador Pentium IV, disco duro de 80 GB y 1 GB de RAM, como mínimo.
- Sistema operativo Microsoft Windows XP.
- MYSQL Server 5.0
- DRIVER MYSQL 3.0.
- MySQL.VisualStudio-1.0.2
- Adobe Reader 8.

3. Requisitos Específicos:

3.1. Requisitos funcionales.

3.1.1. Nombre de la función: R_EST.

3.1.1.1. Especificación.

3.1.1.1.1 Introducción: Esta función permitirá al usuario actualizar en el sistema los datos correspondientes a un estudiante.

3.1.1.1.2. Entradas:

Por pantalla:

- **Numero de carnet:** Es un dato obligatorio debe existir, de lo contrario no se activara el botón de guardar, representa la identificación única del estudiante.
- **Nombre del estudiante:** Es un dato obligatorio debe existir, de lo contrario se presenta un mensaje "Falta el nombre del estudiante".
- **Apellidos**: Es un dato obligatorio debe existir, de lo contrario se presenta el mensaje "Falta los apellidos del estudiante". Este campo nos permitirá saber los apellidos del estudiante.
- **Dirección**: Es un dato obligatorio debe existir de lo contrario se presenta un mensaje "falta la dirección".
- **Cedula:** Es un dato obligatorio debe existir de lo contrario se presenta el mensaje falta el número de cédula.
- **Procedencia:** es un dato obligatorio debe existir de lo contrario se presenta el mensaje falta ciudad o municipio del estudiante.
- **Carrera**: es un dato obligatorio debe existir de lo contrario se presenta el mensaje "Falta la carrera del estudiante".
- **Año de Estudio:** es un dato obligatorio debe existir de lo contrario se presenta el mensaje "Falta el año".
- **Sexo:** es un dato obligatorio debe existir de lo contrario se presenta el mensaje "Falta el sexo del estudiante".
- **Fecha de Nacimiento:** es un dato obligatorio debe existir de lo contrario se presenta el mensaje "Falta la fecha de nacimiento del estudiante".
- **Estado Civil:** es un dato obligatorio debe existir de lo contrario se presenta el mensaje "Falta el estado civil del estudiante".
- **Teléfono**: No es un dato obligatorio, puede o no existir.

- **Monto Exonerado:** es un dato obligatorio debe existir de lo contrario se presenta el mensaje "Falta el monto de exoneración de matricula".

- **Año Lectivo:** No es un dato obligatorio, puede o no existir.
- **Código de Matricula:** Es un dato obligatorio debe existir, de lo contrario no se activara el botón de guardar, representa la identificación única de matricula.
- **Tipo de Beca:** Es un dato obligatorio debe existir de lo contrario se presenta el mensaje "Falta el tipo de beca del estudiante".
- **Situación Académica:** Es un dato obligatorio debe existir de lo contrario se presenta el mensaje "Falta la situación académica del estudiante".

Por el sistema:
El sistema hace uso de la siguiente consulta MYSQL.

```
SELECT    Estudiante.Num_Carnet, Estudiante.Cod_Carrera, Estudiante.Nom_Est,
Estudiante.Apellidos, Estudiante.Direccion, Estudiante.Num_Cedula,
Estudiante.Procedencia, Estudiante.Ano_Estudio, Estudiante.Telefono,

Estudiante.Fecha_Nac, Sexo, Estado_Civil, Estudiante.Tipo_Beca,
Matricula.Cod_Mat, Matricula.Monto_Exo, Matricula.Situacion_Academica,
Matricula.Ano_Lectivo, Carrera.Nom_carrera

FROM      Estudiante, Matricula, Carrera

WHERE  (Estudiante.Num_Carnet = Matricula.Num_Carnet) AND  (Estudiante.Cod_Carrera
= Carrera.Cod_Carrera) AND  (Estudiante.Num_Carnet LIKE ?NCarnet) AND
(Estudiante.Cod_Carrera  LIKE  ?CodCarrera)
```

3.1.1.1.3. Proceso: se muestra la interfaz a través de la cual se podrá ingresar los datos correspondientes a un nuevo estudiante, este se añadirá a la entidad estudiante.

3.1.1.1.4. Salida: Se mostrará el mensaje el estudiante ha sido registrado.

3.1.2. Nombre de la función: B_EST_NUM_CARNET.
3.1.2.1. Especificación.
3.1.2.1.1 Introducción: Esta función permitirá al usuario buscar en el sistema los datos correspondientes a un estudiante por medio de su número de carnet.

3.1.2.1.2. Entradas:
Por pantalla:

- **Numero de carnet.**

Por el sistema:
Esta función no tendrá entradas por el sistema.

3.1.2.1.3. Proceso: se muestra la interfaz a través de la cual se podrá visualizar los datos correspondientes a un estudiante.

3.1.2.1.4. Salida: Se mostrará el mensaje el estudiante ha sido encontrado.

3.1.3. Nombre de la función: LISTADOS_EST.
3.1.3.1. Especificación.
3.1.3.1.1 Introducción: Esta función permitirá al usuario buscar en el sistema los estudiantes correspondientes a una carrera específica.

3.1.3.1.2. Entradas:
Por pantalla:

- **Carrera.**
- **Año**

Por el sistema:
El sistema hace uso de la siguiente consulta MYSQL.

```
SELECT distinct e.Num_Carnet, e.Nom_Est, e.Apellidos, e.Ano_Estudio, c.Nom_carrera
FROM carrera as c inner join estudiante as e on c.Cod_Carrera = e.Cod_Carrera
WHERE   c.Nom_carrera  like ?cadena1 and e.Ano_Estudio like ?cadena2
```

3.1.3.1.3. Proceso: se muestra la interfaz a través de la cual se podrá visualizar los datos correspondientes a un estudiante.

3.1.3.1.4. Salida: Se muestra la lista de los estudiantes según la carrera y año introducido.

3.1.4. Nombre de la función: LISTADOS_ASIG_COD.

3.1.4.1. Especificación.

3.1.4.1.1 Introducción: Esta función permitirá al usuario visualizar en el sistema las asignaturas correspondientes a las diferentes carreras.

3.1.4.1.2. Entradas:

Por pantalla: Esta función no tendrá entradas por pantalla.

Por el sistema:

El sistema hace uso de la siguiente consulta MYSQL.

```
SELECT Asignaturas.Nom_Asig, Asignaturas.Cod_Asig
FROM Asignaturas order by Asignaturas.Cod_Asig
```

3.1.4.1.3. Proceso: se muestra la interfaz a través de la cual se podrá visualizar las asignaturas asignadas a las diferentes carreras.

3.1.4.1.4. Salida: Se mostrará el listado de todas las asignaturas de las diferentes carreras.

3.1.5 Nombre de la función: R_N_EST_ASIG.

3.1.5.1. Especificación.

3.1.5.1.1. Introducción: Esta función permite al usuario registrar en el sistema las notas por asignatura.

3.1.5.1.2. Entradas:

Por pantalla:

- **Carrera:** Es un dato obligatorio debe existir de lo contrario se presenta el mensaje "Falta la carrera del estudiante".

- **Semestre:** Es un dato obligatorio debe existir de lo contrario se presenta el mensaje "Falta el semestre".

- **Asignatura:** Es un dato obligatorio debe existir de lo contrario se presenta el mensaje "Falta la asignatura".

Por el sistema:

El sistema hace uso de la siguiente consulta MYSQL.

SELECT distinct e.Num_Carnet, e.Apellidos, e.Nom_Est, n.IP, n.IIP, n.Prom, n.EF, n.NF, n.EE, n.CV, n.TT, a.Nom_Asig, n.Cod_Asig

FROM asignaturas as a inner join (carrera as c inner join (estudiante as e inner join notas as n on e.Num_Carnet = n.Num_carnet) on c.Cod_Carrera = e.Cod_Carrera) on a.Cod_Asig = n.Cod_Asig

WHERE c.Cod_Carrera like ?CodCarrera and a.NomAsig like ?NomAsig and a.Semestre like ?Semestre

3.1.5.1.3. Proceso: se muestra la interfaz de introducción de datos, y luego se podrá registrar la nota del estudiante.

3.1.5.1.4. Salida: Se muestra la lista de todos los nombres completos y el número de carnet, la asignatura que se ha elegido y las notas a introducir y se guarda el registro de las notas.

3.1.6. Nombre de la función: CTAR_N_EST.

3.1.6.1. Especificación:

3.1.6.1.1. Introducción: Esta función muestra al usuario la nota de un determinado estudiante registrado en el sistema de la Base de datos

3.1.6.1.2. Entradas:

Por pantalla:

- **Carnet.**
- **Carrera.**
- **Semestre**
- **Asignatura.**

Por el sistema:

El sistema hace uso de las siguientes consultas MYSQL.

SELECT distinct e.Num_Carnet, e.Apellidos, e.Nom_Est, e.Ano_Estudio, a.Nom_Asig, a.Semestre, n.IP, n.IIP, n.Prom, n.EF, n.NF, n.EE, n.CV, n.TT

FROM estudiante as e inner join (Asignaturas as a inner join notas as n on a.Cod_Asig = n.Cod_Asig) on e.Num_Carnet = n.Num_Carnet

WHERE e.Num_Carnet like ?Carnet and a.Nom_Asig like ?Asig and a.Semestre like ?Sem

3.1.6.1.3. Proceso: Se muestra la interfaz a través del cual se puede ingresar la información referente a un estudiante.

3.1.6.1.4. Salida: Se muestra del resultado de la consulta efectuada a la base de datos la cual proporciona el nombre del estudiante, la asignatura, y la nota del estudiante.

3.1.7. Nombre de la función: MCAR_N_EST.
3.1.7.1 Especificación:
3.1.7.1.1. Introducción: Esta función permite al usuario modificar la nota del estudiante.

3.1.7.1.2. Entradas:

Por pantalla:

- **Carnet.**

Por el sistema:
Esta función no tendrá entradas por el sistema.

3.1.7.1.3. Proceso: Se presenta al usuario la interfaz de modificación de la nota del estudiante introduciendo los datos necesarios de este.

3.1.7.1.4. Salida: Se muestra el resultado de la consulta la cual proporciona el nombre completo, asignatura, nota y se registra la modificación de la nota del estudiante.

3.1.8. Nombre de la función: MCAR_DTOS_EST.
3.1.8.1 Especificación:
3.1.8.1.1. Introducción: Esta función permite al usuario modificar los datos del estudiante.

3.1.8.1.2. Entradas:

Por pantalla:

- **El Número de carnet a buscar:** este dato no es obligatorio, pero es necesario para encontrar el estudiante al que se modificará algún dato.

Por el sistema:
Esta función no tendrá entradas por el sistema.

3.1.8.1.3. Proceso: Se muestra la interfaz a través del cual se podrá modificar los datos correspondientes al estudiante.

3.1.8.1.4. Salida: Se guarda el registro modificado en la base de datos.

3.1.9. Nombre de la función: ACTUAL_INSERT_CARRERA.
3.1.9.1 Especificación:
3.1.9.1.1. Introducción: Esta función permite al usuario modificar e insertar una carrera.

3.1.9.1.2. Entradas:

Por pantalla:
Esta función no tendrá entradas por pantalla.

Por el sistema:
Esta función no tendrá entradas por el sistema.

3.1.9.1.3. Proceso: Se muestra la interfaz a través del cual se podrá modificar los datos correspondientes a una carrera.

3.1.9.1.4. Salida: Se guarda el registro modificado en la base de datos.

3.2.0. Nombre de la función: ACTUAL_INSERT_ASIG.
3.2.0.1 Especificación:
3.2.0.1.1. Introducción: Esta función permite al usuario modificar e insertar asignaturas.

3.2.0.1.2. Entradas:

Por pantalla:
Esta función no tendrá entradas por pantalla.

Por el sistema:

Esta función no tendrá entradas por el sistema.

3.2.0.1.3. Proceso: Se muestra la interfaz a través del cual se podrá modificar los datos correspondientes a una asignatura.

3.2.0.1.4. Salida: Se guarda el registro modificado en la base de datos.

3.2.1. Nombre de la función: ELIMINAR_EST.

3.2.1.1. Especificación.

3.2.1.1.1 Introducción: Esta función permitirá al usuario eliminar un estudiante registrado en la base de datos.

3.2.1.1.2. Entradas:

Por pantalla:

- **Numero de carnet:** que se presenta actualmente en el formulario.

Por el sistema:

Esta función no tendrá entradas por el sistema.

3.2.1.1.3. Proceso: se muestra la interfaz a través de la cual se podrá visualizar los datos correspondientes a un estudiante.

3.2.1.1.4. Salida: Se mostrará el mensaje el estudiante eliminado correctamente.

3.2.2. Nombre de la función: R_PROF

3.2.2.1 Especificación:

3.2.2.1.1. Introducción: Esta función permitirá al usuario adicionar en el sistema los datos correspondientes a un profesor.

3.2.2.1.2. Entradas:

Por pantalla:

- **Numero de Cedula.**
- **Nombre del profesor.**
- **Apellidos.**

- **Dirección**
- **Teléfono**
- **Procedencia**

Por el sistema:
El sistema hace uso de las siguientes consultas MYSQL.

SELECT Num_Cedula, Nom_Profesor, Apellidos, Direccion, Telefono, Procedencia
FROM Profesor
WHERE (profesor.Num_Cedula LIKE ?cadena)

3.2.2.1.3. Proceso: Se mostrara la interfaz a través del cual se podrá ingresar los datos correspondientes a un profesor

3.2.2.1.4. Salida: Se guarda el registro de los datos del profesor y se muestra el reporte de registro realizado.

3.2.3. Nombre de la función: ACTUALIZAR_PROF
3.2.3.1 Especificación:
3.2.3.1.1. Introducción: Esta función permitirá al usuario actualizar en el sistema los datos correspondientes a un profesor.

3.2.3.1.2. Entradas:

Por pantalla:

- **Número de Cédula:** este dato no es obligatorio, pero es necesario, para encontrar el profesor a actualizar.

Por el sistema:
Esta función no tendrá entradas por el sistema.

3.2.3.1.3. Proceso: Se mostrara la interfaz a través del cual se podrá modificar los datos correspondientes a un profesor.

3.2.3.1.4. Salida: Se guarda el registro con los datos actualizados del profesor.

3.2.4. Nombre de la función: ELIMINAR_PROF
3.2.4.1 Especificación:

3.2.4.1.1. Introducción: Esta función permitirá al usuario eliminar a un profesor de la base de datos.

3.2.4.1.2. Entradas:

Por pantalla:

- **Número de Cédula.** este dato no es obligatorio, pero es necesario, para encontrar el profesor a eliminar.

Por el sistema:
Esta función no tendrá entradas por el sistema.

3.2.4.1.3. Proceso: Se mostrara la interfaz a través del cual se podrá eliminar los datos correspondientes a un profesor.

3.2.4.1.4. Salida: Se guarda el registro de los profesores incluyendo la actualización de la eliminación del profesor.

3.2.5. Nombre de la función: CTAR_IA_CARRERA
3.2.5.1 Especificación:
3.2.5.1.1. Introducción: Esta función muestra al usuario los índices académicos por carrera.

3.2.5.1.2. Entradas:

Por pantalla:

- **Carrera.**

Por el sistema:
El sistema hace uso de las siguientes consultas SQL.

SELECT DISTINCT estudiante.Num_Carnet, estudiante.Nom_Est, estudiante.Apellidos, estudiante.Ano_Estudio, carrera.Nom_Carrera, Max(notas.IA) AS Índice

FROM (carrera INNER JOIN estudiante ON carrera.Cod_Carrera = estudiante.Cod_Carrera) INNER
JOIN notas ON estudiante.Num_Carnet = notas.Num_Carnet

GROUP BY estudiante.Num_Carnet, estudiante.Nom_Est, estudiante.Apellidos, Estudiante.Ano_Estudio, Carrera.Nom_Carrera HAVING carrera.Nom_carrera like ?Carrera

3.2.5.1.3. Proceso: Se muestra la interfaz a través del cual se puede introducir los datos y luego se visualizara los mejores índices académicos.

3.2.5.1.4. Salida: Se muestra el resultado de la consulta efectuada al a base de datos la cual proporciona los nombres de los estudiantes, carrera, año y el índice académico.

3.2.6. Nombre de la función: CTAR_IA_AÑO

3.2.6.1 Especificación:

3.2.6.1.1. Introducción: Esta función muestra al usuario el índice académico por semestre esto se hace a través de una consulta SQL a la base de datos.

3.2.6.1.2. Entradas:

Por pantalla:

- **Carrera.**
- **Año.**

Por el sistema:

El sistema hace uso de las siguientes consultas SQL.

SELECT DISTINCT estudiante.Num_Carnet, estudiante.Nom_Est, estudiante.Apellidos, estudiante.Ano_Estudio, carrera.Nom_Carrera, Max(notas.IA) AS Índice

FROM (carrera INNER JOIN estudiante ON carrera.Cod_Carrera = estudiante.Cod_Carrera) INNER
JOIN notas ON estudiante.Num_Carnet = notas.Num_Carnet

GROUP BY estudiante.Num_Carnet, estudiante.Nom_Est, estudiante.Apellidos, Estudiante.Ano_Estudio, Carrera.Nom_Carrera HAVING carrera.Nom_carrera like ?F11_carrera
AND estudiante.Ano_Estudio like ?F11_año

3.2.6.1.3. Proceso: Se muestra al usuario la interfaz de introducción de datos y luego se visualizara el índice académico por año.

3.2.6.1.4. Salida: Se muestra el resultada de la consulta efectuada al a base de a datos la cual proporciona el nombre de la, carrera, lista de todos los estudiantes con su índice académico según el año especificado.

3.2.7 Nombre de la función: IMPRIMIR_REPORTE_N_REGIST

3.2.7.1. Especificación.

3.2.7.1.1. Introducción: Esta función permite al usuario registrar en el sistema las notas por asignatura y que se permita imprimir.

3.2.7.1.2. Entradas:

Por pantalla:

- **Carrera**
- **Semestre**
- **Asignatura**

Por el sistema:

El sistema hace uso de la siguiente consulta MYSQL.

SELECT distinct e.Num_Carnet, e.Apellidos, e.Nom_Est, n.IP, n.IIP, n.Prom, n.EF, n.NF, n.EE, n.CV, n.TT, a.Nom_Asig, n.Cod_Asig
FROM asignaturas as a inner join (carrera as c inner join (estudiante as e inner join notas as n on e.Num_Carnet = n.Num_carnet) on c.Cod_Carrera = e.Cod_Carrera) on a.Cod_Asig = n.Cod_Asig
WHERE c.Cod_Carrera like ?CodCarrera and a.NomAsig like ?NomAsig and a.Semestre like ?Semestre

3.2.5.1.3. Proceso: se muestra la interfaz de introducción de datos, y luego se presenta la nota del estudiante.

3.2.5.1.4. Salida: Se muestra la lista de todos los nombres completos y el número de carnet, la asignatura que se ha elegido y las notas a introducir y se guarda el registro de las notas.

3.2.6. Nombre de la función: IMPRIMIR_REPORTE_S_N_EST

3.2.6.1 Especificación:

3.2.6.1.1. Introducción: Esta función muestra e imprime al usuario un reporte semestral de las notas del estudiante esto se hace a través de una consulta SQL a la base de datos.

3.2.6.1.2. Entradas:

Por pantalla:

- **Carnet**
- **Carrera**
- **Semestre.**

Por el sistema:
El sistema hace uso de las siguientes consultas SQL.

SELECT e.Num_Carnet, e.Nom_Est, e.Apellidos, c.Nom_Carrera, c.Cod_Carrera, a.Semestre, a.Nom_Asig, n.IP, n.IIP, n.Prom, n.EF, n.NF

FROM carrera as c inner join (estudiante as e inner join (asignaturas as a inner join notas as n on a.Cod_Asig = n.Cod_Asig) on e.Num_Carnet = n.Num_Carnet) on c.Cod_Carrera = e.Cod_Carrera

WHERE e.Num_Carnet like ?cadena4 and c.Nom_Carrera like ?cadena5 and a.Semestre like ?cadena6

3.2.6.1.3. Proceso: Se muestra al usuario la interfaz de introducción de datos y luego se presenta el reporte semestral referente a las notas del estudiante.

3.2.6.1.4. Salida: Se muestra e imprime el resultado de la consulta efectuada en la base de datos la cual proporciona semestralmente el nombre del estudiante, y las asignaturas con sus respectivas notas e índice académico.

3.2.2.7. Nombre de la función: IMPRIMIR_ REPORTE_A_N_EST
3.2.7.1. Especificación:
3.2.7.1.1. Introducción: Esta función muestra e imprime al usuario un reporte anual de las notas del estudiante esto se hace a través de una consulta SQL a la base de datos.

3.2.7.1.2. Entradas:

Por pantalla:

- **Carnet.**
- **Carrera.**

- **Año.**

Por el sistema:
El sistema hace uso de las siguientes consultas SQL.

SELECT e.Nom_Est, e.Apellidos, e.Num_Carnet, e.Ano_Estudio, c.Nom_Carrera, a.Nom_asig, a.Semestre, n.IP, n.IIP, n.Prom, n.EF, n.NF

FROM (carrera as c inner join estudiante as e on c.Cod_carrera = e.Cod_carrera) inner join (asignaturas as a inner join notas as n on a.Cod_Asig = n.Cod_asig) on e.Num_Carnet = n.Num_Carnet

WHERE e.Num_carnet like ?cadena1 and c.Nom_Carrera LIKE ?cadena2 and e.Ano_Estudio like ?cadena3 ORDER BY a.Semestre

3.2.7.1.3. Proceso: Se muestra al usuario la interfaz de introducción de datos y luego se presenta el reporte anual referente a las notas del estudiante.

3.2.7.1.4. Salida: Se muestra e imprime el resultado de la consulta efectuada en la base de datos la cual proporciona anualmente el nombre del estudiante, y las asignaturas con sus respectivas notas e índice académico.

3.2.7. Nombre de la función: CTCIA_EST_ACTIVO.
3.2.7.1 Especificación:
3.2.7.1.1. Introducción: Esta función permite que el usuario visualice o imprima la constancia de alumno activo.

3.2.7.1.2. Entradas:
Por pantalla:

- **Carnet.**
- **Fecha actual.**

Por el sistema:
El sistema hace uso de las siguientes consultas SQL.

SELECT distinct e.Num_Carnet, e.Apellidos, e.Nom_Est, c.Nom_Carrera , e.Ano_Estudio

FROM carrera as c inner join Estudiante as e on c.Cod_Carrera = e.Cod_Carrera

WHERE e.Num_Carnet like ?NCarnet

3.2.7.1.3. Proceso: Se muestra la interfaz a través de la cual el usuario podrá visualizar o imprimir los datos de la constancia pertenecientes a un estudiante.

3.2.7.1.4. Salida: El usuario podrá visualizar en hoja la constancia de alumno activo correspondiente a un determinado estudiante con su nombre, apellidos, número de carnet, carrera, año que cursa.

3.2.8. Nombre de la función: CTAR_EST_REP_ASIG
3.2.8.1 Especificación:
3.2.8.1.1. Introducción: Esta función muestra al usuario los estudiantes que reprobaron por asignatura seleccionando el semestre de la carrera, esto se hace a través de una consulta SQL a la base de datos.

3.2.8.1.2. Entradas:
Por pantalla:

- **Carrera.**
- **Año.**
- **Asignatura.**

Por el sistema: El sistema hace uso de las siguiente consulta SQL.

SELECT e.Num_Carnet, e.Apellidos, e.Nom_Est, e.Ano_Estudio, c.Nom_Carrera, a.Nom_Asig, n.NF

FROM (carrera as c inner join estudiante as e on c.Cod_Carrera = e.Cod_Carrera) inner join (asignaturas as a inner join notas as n on a.Cod_Asig = n.Cod_Asig) on e.Num_Carnet = n.Num_Carnet

WHERE a.Nom_Asig like ?Asig and c.Nom_Carrera like ?Carrera and e.Ano_Estudio like ?Año and n.NF < 60 or n.NF = 0 or n.NF = Null

CV:

SELECT e.Num_Carnet, e.Apellidos, e.Nom_Est, e.Ano_Estudio, c.Nom_Carrera, a.Nom_Asig, n.CV

FROM (carrera as c inner join estudiante as e on c.Cod_Carrera = e.Cod_Carrera) inner join (asignaturas as a inner join notas as n on a.Cod_Asig = n.Cod_Asig) on e.Num_Carnet = n.Num_Carnet

WHERE a.Nom_Asig like ?Asig_CV and c.Nom_Carrera like ?Carrera_CV and e.Ano_Estudio like ?Año_CV and n.CV < 60 or n.CV = 0

EE:

SELECT e.Num_Carnet, e.Apellidos, e.Nom_Est, e.Ano_Estudio, c.Nom_Carrera, a.Nom_Asig, n.EE

FROM (carrera as c inner join estudiante as e on c.Cod_Carrera = e.Cod_Carrera) inner join (asignaturas as a inner join notas as n on a.Cod_Asig = n.Cod_Asig) on e.Num_Carnet = n.Num_Carnet

WHERE a.Nom_Asig like ?Asig_EE and c.Nom_Carrera like ?Carrera_EE and e.Ano_Estudio like ?Año_EE and n.EE < 60 or n.EE = 0 or n.EE = Null

TT:

SELECT e.Num_Carnet, e.Apellidos, e.Nom_Est, e.Ano_Estudio, c.Nom_Carrera, a.Nom_Asig, n.TT

FROM (carrera as c inner join estudiante as e on c.Cod_Carrera = e.Cod_Carrera) inner join (asignaturas as a inner join notas as n on a.Cod_Asig = n.Cod_Asig) on e.Num_Carnet = n.Num_Carnet

WHERE a.Nom_Asig like ?Asig_TT and c.Nom_Carrera like ?Carrera_TT and e.Ano_Estudio like ?Año_TT and n.TT < 60 or n.TT = 0

3.2.8.1.3. Proceso: Se mostrara la pantalla de introducción de datos al usuario y luego se podrá visualizar la información referente a los estudiantes que reprobaron por asignatura.

3.2.8.1.4. Salida: se muestra el resultado de la consulta efectuad a la base de datos la actual proporciona el número de carnet, nombre del estudiante, su carrera, y la nota final.

3.2.9. Nombre de la función: EST_REP_CARRERA.

3.2.9.1 Especificación:

3.2.9.1.1. Introducción: Esta función muestra al usuario los estudiantes que reprobaron por carrera seleccionando el año (semestre) hace a través de una consulta SQL a la base de datos.

3.2.9.1.2. Entradas:

Por pantalla:

- **Carrera**

Por el sistema: El sistema hace uso de las siguiente consulta SQL.

SELECT e.Num_Carnet, e.Apellidos, e.Nom_Est, e.Ano_Estudio, c.Nom_Carrera, a.Nom_Asig, n.NF

FROM (carrera as c inner join estudiante as e on c.Cod_Carrera = e.Cod_Carrera) inner join (asignaturas as a inner join notas as n on a.Cod_Asig = n.Cod_Asig) on e.Num_Carnet = n.Num_Carnet

WHERE c.Nom_Carrera like ?F13_Carrera_NF and n.NF < 60 or n.NF = 0 order by e.Ano_Estudio

CV:

SELECT e.Num_Carnet, e.Apellidos, e.Nom_Est, e.Ano_Estudio, c.Nom_Carrera, a.Nom_Asig, n.CV

FROM (carrera as c inner join estudiante as e on c.Cod_Carrera = e.Cod_Carrera) inner join (asignaturas as a inner join notas as n on a.Cod_Asig = n.Cod_Asig) on e.Num_Carnet = n.Num_Carnet

WHERE c.Nom_Carrera like ?F13_Carrera_CV and n.CV < 60 or n.CV = 0 order by e.Ano_Estudio

EE:

SELECT e.Num_Carnet, e.Apellidos, e.Nom_Est, e.Ano_Estudio, c.Nom_Carrera, a.Nom_Asig, n.EE

FROM (carrera as c inner join estudiante as e on c.Cod_Carrera = e.Cod_Carrera) inner join (asignaturas as a inner join notas as n on a.Cod_Asig = n.Cod_Asig) on e.Num_Carnet = n.Num_Carnet

WHERE c.Nom_Carrera like ?F13_Carrera_EE and n.EE < 60 or n.EE = 0 order by e.Ano_Estudio

TT:

SELECT e.Num_Carnet, e.Apellidos, e.Nom_Est, e.Ano_Estudio, c.Nom_Carrera, a.Nom_Asig, n.TT

FROM (carrera as c inner join estudiante as e on c.Cod_Carrera = e.Cod_Carrera) inner join (asignaturas as a inner join notas as n on a.Cod_Asig = n.Cod_Asig) on e.Num_Carnet = n.Num_Carnet

WHERE c.Nom_Carrera like ?F13_Carrera_TT and n.TT < 60 or n.TT = 0 order by e.Ano_Estudio

3.2.9.1.3. Proceso: Se mostrara la pantalla de introducción de datos al usuario y luego se podrá visualizar la información referente a los estudiantes que reprobaron por carrera.

3.2.9.1.4. Salida: se muestra el resultado de la consulta efectuad a la base de datos la actual proporciona el número de carnet, nombre del estudiante, y las asignaturas reprobadas de la carrera y la nota.

3.3.0. Nombre de la función: CTAR_EST_BECAS_EX.

3.3.0.1 Especificación:

3.3.0.1.1. Introducción: Esta función muestra al usuario los estudiantes con becas externas por carrera y por año, esto se hace a través de una consulta SQL.

3.3.0.1.2. Entradas:

Por pantalla:

- **Tipo beca:**
- **carrera**

Por el sistema: El sistema hace uso de las siguientes consultas SQL.

SELECT e.Num_Carnet, e.Nom_Est, e.Apellidos, e.Ano_Estudio, e.Tipo_Beca, c.Nom_Carrera

FROM carrera as c inner join estudiante as e on c.Cod_Carrera = e.Cod_Carrera

WHERE e.Tipo_Beca like ?cadena1 and c.Nom_Carrera like ?cadena2 order by e.Ano_Estudio

3.3.0.1.3. Proceso: se muestra la interfaz a través de la cual se podrá visualizar la información referente a las becas externas de los estudiantes.

3.3.0.1.4. Salida: Se muestra el resultado de la consulta efectuada a la base de datos la cual proporciona nombres de los estudiantes con becas externas según la carrera y el año introducido.

3.3.1. Nombre de la función: CTAR_EST_BECAS_INT.
3.3.1.1 Especificación:
3.3.1.1.1. Introducción: Esta función muestra al usuario los estudiantes con becas internas por carrera y seleccionando el año, esto se hace a través de una consulta SQL.

3.3.1.1.2. Entradas:

Por pantalla:

- **Tipo beca:**
- **carrera**

Por el sistema: El sistema hace uso de las siguientes consultas SQL.

SELECT e.Num_Carnet, e.Nom_Est, e.Apellidos, e.Ano_Estudio, e.Tipo_Beca, c.Nom_Carrera

FROM carrera as c inner join estudiante as e on c.Cod_Carrera = e.Cod_Carrera

WHERE e.Tipo_Beca like ?cadena1 and c.Nom_Carrera like ?cadena2 order by e.Ano_Estudio

3.3.1.1.3. Proceso: se muestra la interfaz a través de la cual se podrá visualizar la información referente a los estudiantes con becas internas.

3.3.1.1.4. Salida: Se muestra el resultado de la consulta efectuada a la base de datos la cual proporciona nombres de los estudiantes con becas internas según la carrera y el año introducido.

3.3.2. Nombre de la función: CTAR_EST_SIN_BECAS.
3.3.2.1 Especificación:
3.3.2.1.1. Introducción: Esta función muestra al usuario los estudiantes sin becas por carrera y por año, esto se hace a través de una consulta SQL.

3.3.2.1.2. Entradas:

Por pantalla:

- **Tipo de beca**
- **Carrera.**

Por el sistema: El sistema hace uso de las siguientes consultas SQL.

SELECT e.Num_Carnet, e.Nom_Est, e.Apellidos, e.Ano_Estudio, e.Tipo_Beca, c.Nom_Carrera

FROM carrera as c inner join estudiante as e on c.Cod_Carrera = e.Cod_Carrera

WHERE e.Tipo_Beca like ?cadena1 and c.Nom_Carrera like ?cadena2 order by e.Ano_Estudio

3.3.2.1.3. Proceso: se muestra la interfaz a través de la cual se podrá visualizar la información referente a los estudiantes sin becas.

3.3.2.1.4. Salida: Se muestra el resultado de la consulta efectuada a la base de datos la cual proporciona nombres, apellidos, número de carnet, la carrera y el año de los estudiantes sin becas.

3.3.3. Nombre de la función: CTAR_EXO_MAT

3.3.3.1. Especificación:

3.3.3.1.1. Introducción: muestra al usuario los estudiantes con exoneración de matricula por carrera y seleccionando el año. Esto se hace a través de una consulta SQL a la base de datos.

3.3.3.1.2. Entradas:

Por pantalla:

No tiene entradas por pantalla.

Por el sistema:

El sistema hace uso de las siguientes consultas SQL.

SELECT e.Num_Carnet, e.Apellidos, e.Nom_Est, m.Monto_Exo, e.Ano_Estudio, c.Nom_Carrera

FROM (carrera as c inner join estudiante as e on c.Cod_Carrera = e.Cod_Carrera) inner join matricula as m on e.Num_Carnet = m.Num_Carnet

WHERE m.Monto_Exo > 0 order by Nom_carrera

3.3.3.1.3. Proceso: se muestra la interfaz a través de la cual se podrá visualizar la información referente a los estudiantes con exoneración de matricula.

3.3.3.1.4. Salida: Se muestra el resultado de la consulta efectuada a la base de datos la cual proporciona el carnet, nombres, apellidos de los estudiantes con exoneración de matricula, la carrera y el año y el monto exonerado.

3.3.4. Nombre de la función: RESPALDO_BD

3.3.4.1 Especificación:

3.3.4.1.1. Introducción: Esta función permite al usuario respaldar la base de datos con el fin de tener seguridad de la información. Este proceso va a permitir crear un respaldo de la base de datos con el mismo nombre que la del sistema o con otro nombre dependiendo de la necesidad del usuario. Además va a permitir seleccionar la unidad de destino en donde se va a almacenar el respaldo.

3.3.4.1.2. Entradas:

Por pantalla:

- **Nombre de la base de datos y/o respaldo:** Es un dato obligatorio debe existir de lo contrario se mostrara el mensaje falta el nombre de la base de datos.
- **Destino:** Es un dato obligatorio donde se debe de escoger el destino donde se almacenará el respaldo.

Por el sistema:

- Contraseña en MySql: es un dato obligatorio, de lo contrario no se puede realizar el respaldo satisfactoriamente.

3.3.4.1.3. Proceso: Se presenta al usuario la interfaz de respaldo de la base de datos.
3.3.4.1.4. Salida: Se genera una copia de la base de datos del sistema, con una ventana, respaldo satisfactorio.

3.3.5. Nombre de la función: CBIAR_CONTRASEÑA_USR.
3.3.5.1 Especificación:
3.3.5.1.1. Introducción: esta función permite al usuario modificar su contraseña.

3.3.5.1.2. Entradas:

Por pantalla:

- **Usuario:** es un dato obligatorio, representa uno de los dos usuarios del sistema.
- **Contraseña actual:** es un dato obligatorio debe de existir, representa la contraseña actual
- **Nueva contraseña**: es un dato obligatorio debe de existir y representa la nueva contraseña.
- **Confirmar contraseña:** es un dato obligatorio debe de existir y permite confirmar la nueva contraseña

Por el sistema:
No se presenta ninguna entrada por el sistema.

3.3.5.1.3. Proceso: se presenta al usuario la interfaz de modificación de contraseña del usuario que esta actualmente utilizando la contraseña.

3.3.5.1.4. Salida: se registra la modificación de la contraseña.

VIII-Diagramas de Flujo de Datos.

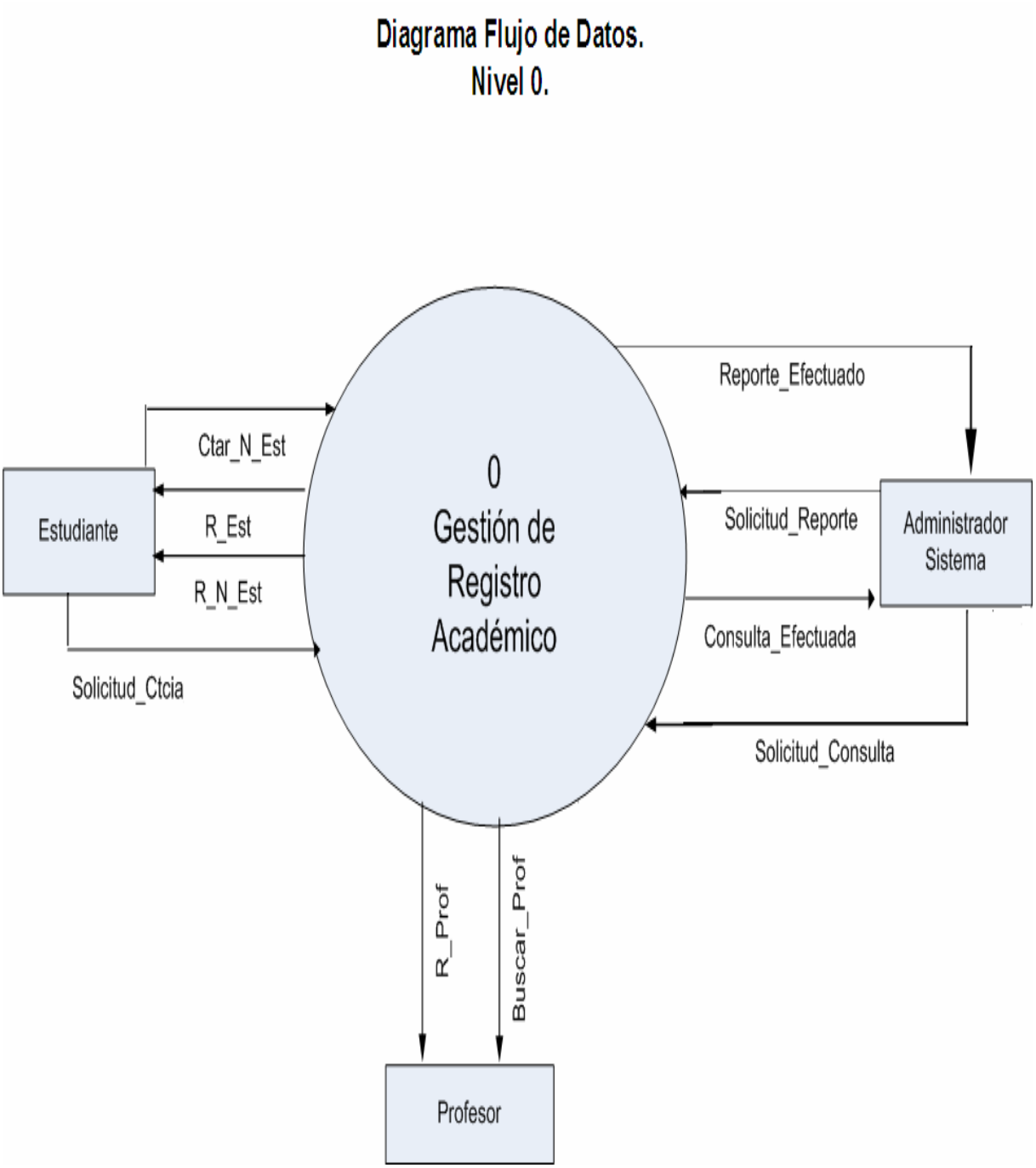


Diagrama Flujo de Datos.
Nivel 1.

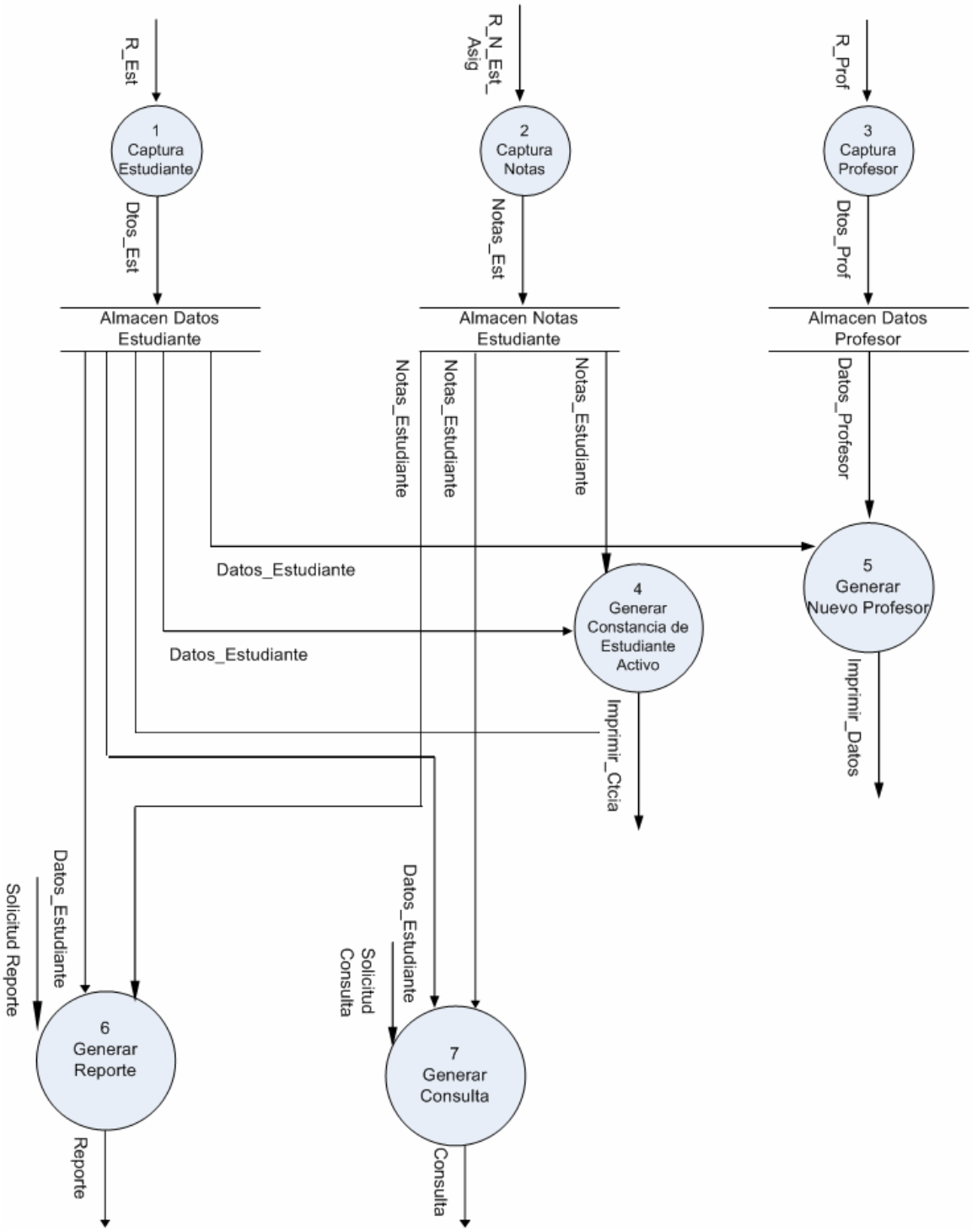
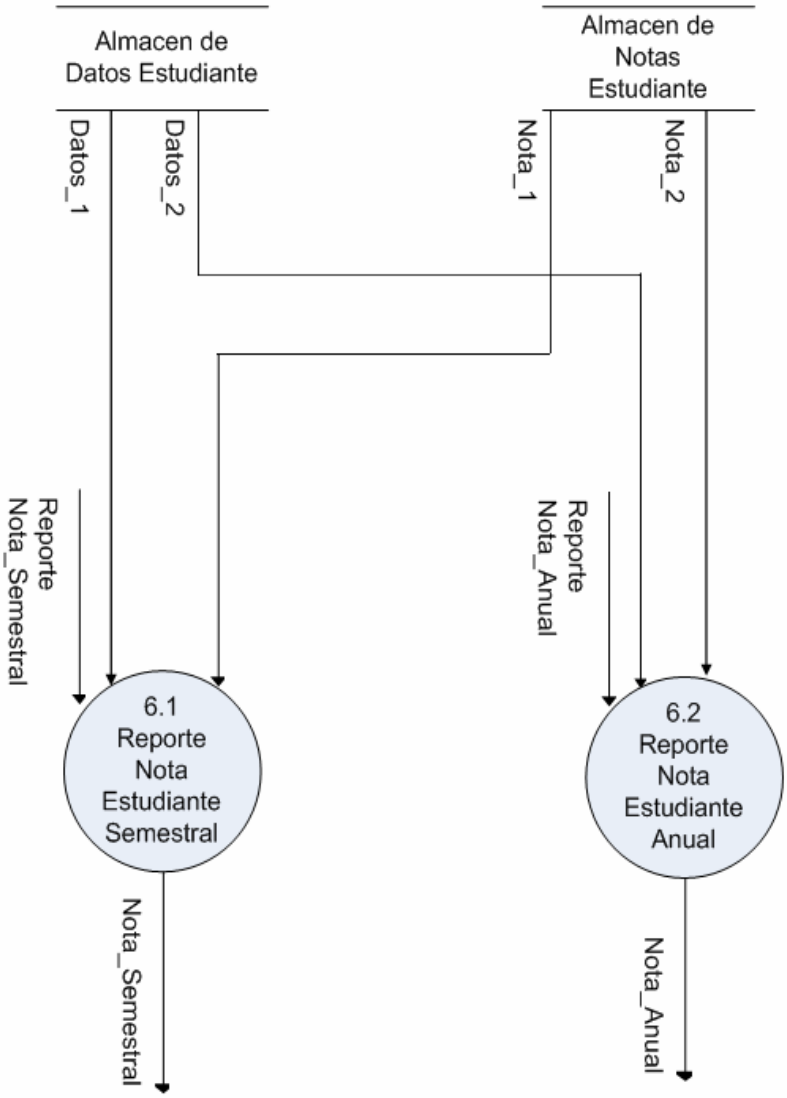


Diagrama 1, Nivel 2



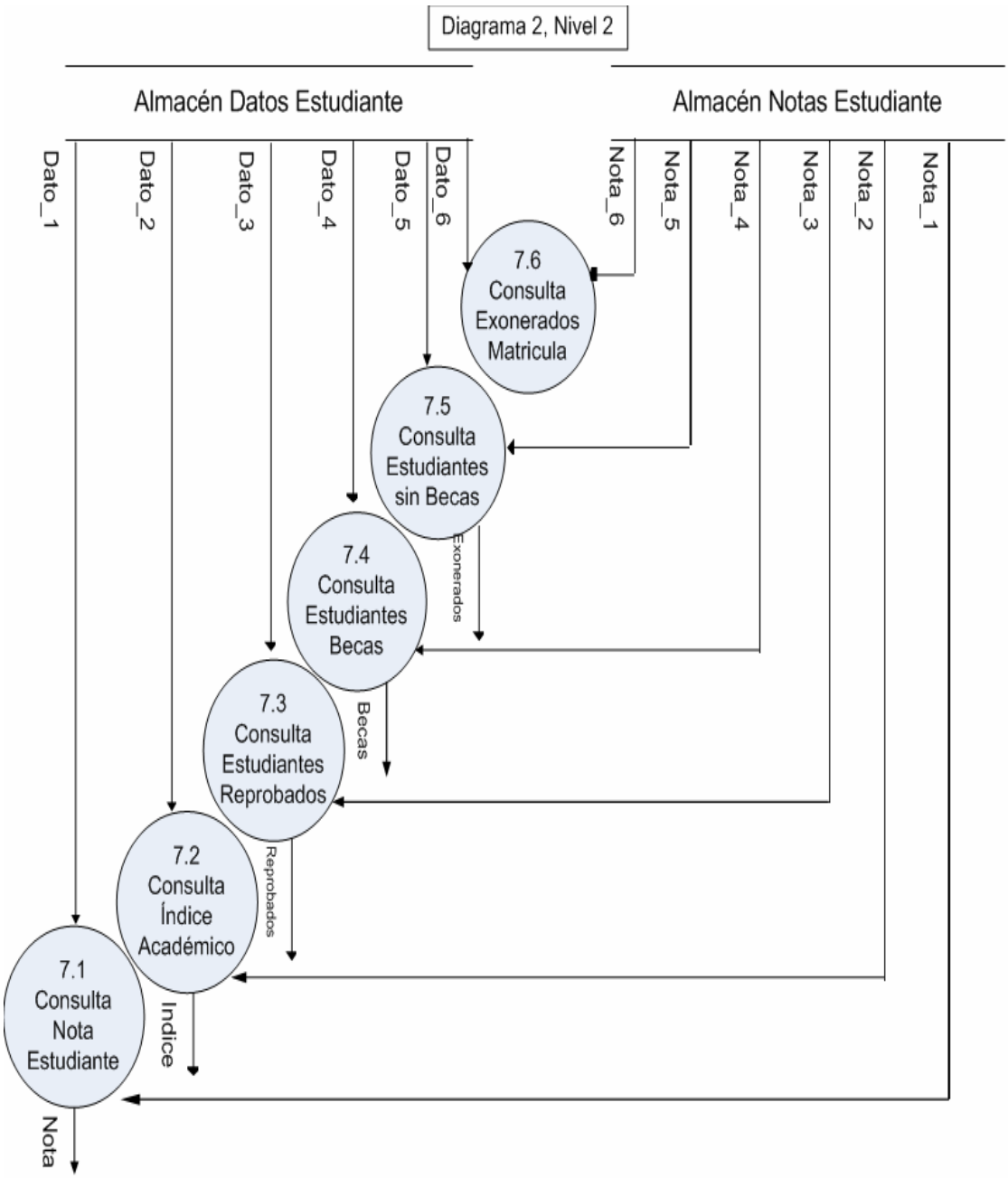


Diagrama 2, Nivel 2
Almacén Datos Estudiante
Almacén Notas Estudiante
Dato_1
Dato_2
Dato_3
Dato_4
Dato_5
Dato_6
Nota_6
Nota_5
Nota_4
Nota_3
Nota_2
Nota_1
7.6 Consulta Exonerados Matricula
7.5 Consulta Estudiantes sin Becas
Exonerados
7.4 Consulta Estudiantes Becas
Becas
7.3 Consulta Estudiantes Reprobados
Reprobados
7.2 Consulta Índice Académico
Indice
7.1 Consulta Nota Estudiante
Nota

Diagrama 1 Nivel 3

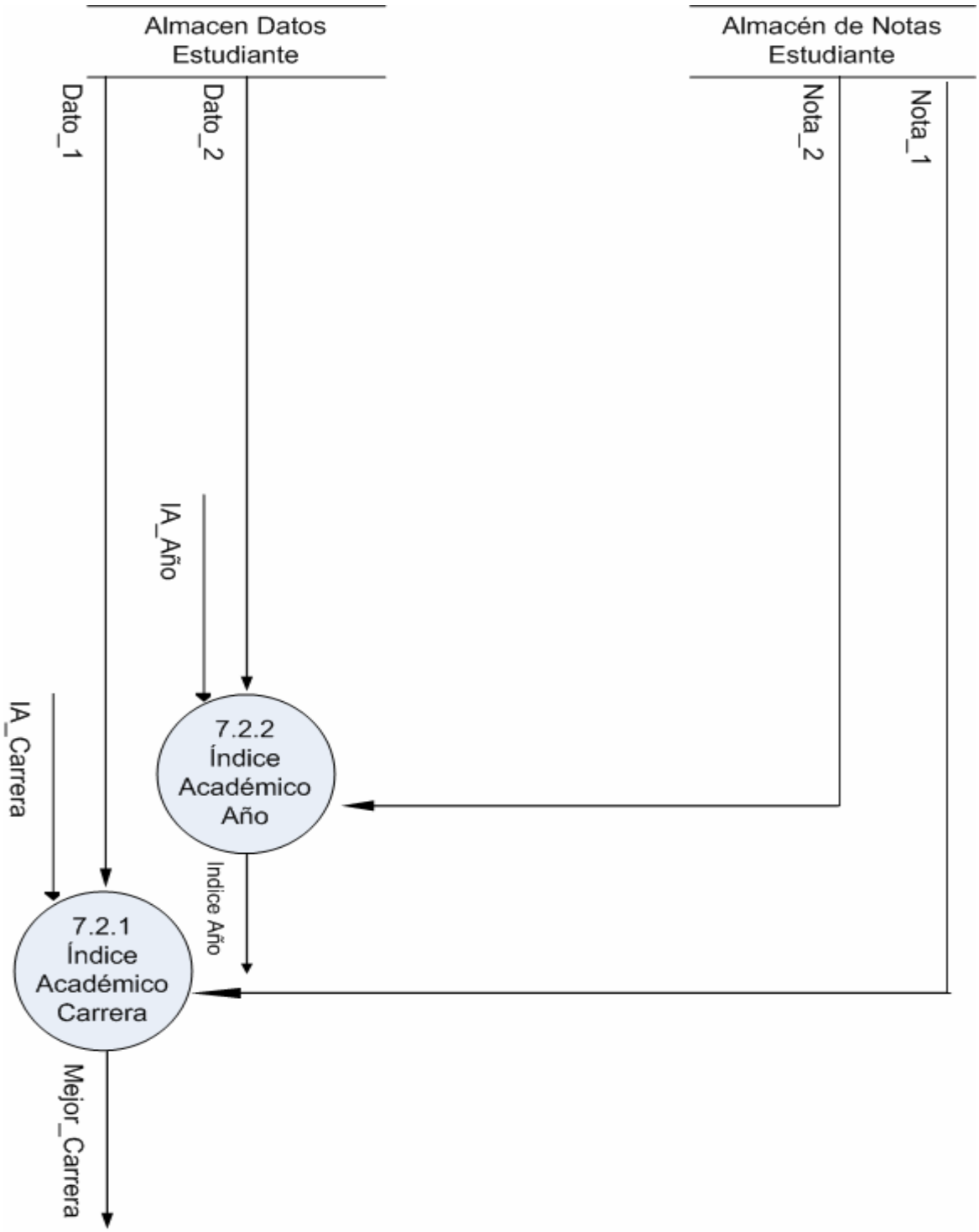
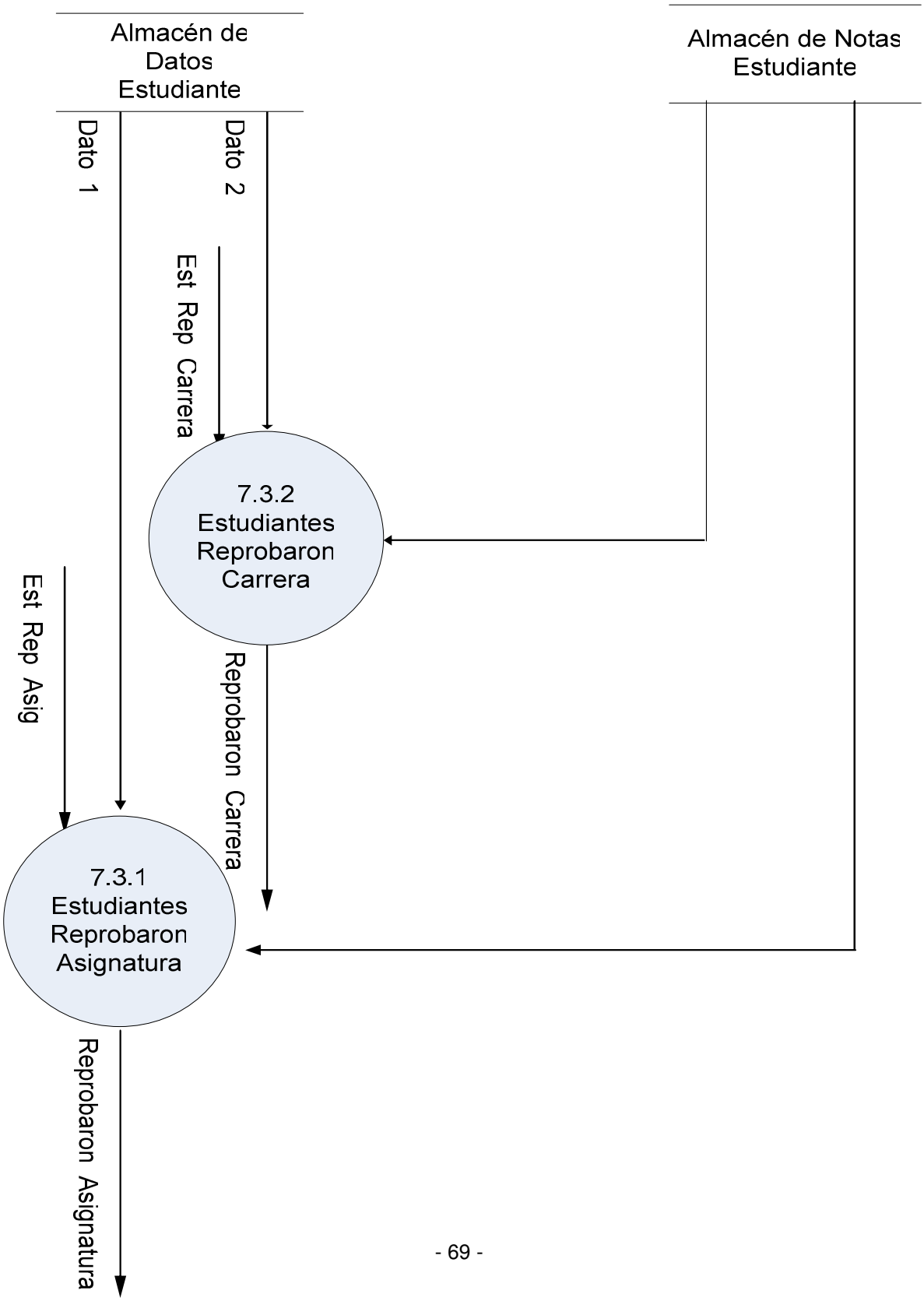


Diagrama 2, Nivel 3



IX-Diccionario de Datos.

Cada uno de los flujos de datos esta desarrollado de la siguiente manera:

Estudiante: Entidad externa.

Administrador del sistema: Entidad externa.

CAPTURA ESTUDIANTE: Este proceso permite adicionar los datos correspondientes a un nuevo estudiante.

R_EST: Numero de carnet + Nombres + Apellidos + Dirección + Cedula + Procedencia + Carrera + Año + Teléfono.
Se usa como entrada en el proceso captura estudiante.

CAPTURA NOTAS: Este proceso permite registrarlas notas del estudiante.

R_N_EST_ASIG: Numero de carnet + Carrera + Semestre + Asignatura + Notas.
Se usa como entrada en el proceso captura notas.

CAPTURA PROFESOR: Este proceso permitirá adicionar los datos correspondientes a un profesor.

R_PROF: Nombres + Apellidos + Dirección + Procedencia + Teléfono + Num_Cedula.
Se usa como entrada en el proceso captura profesor.

IMPRIMIR_CTCIA: Numero de carnet + Nombre completo + Apellidos+ Carrera + Año de Estudio. Se usa como salida en el proceso generar constancia de estudiante activo.

GENERAR CONSTANCIA ESTUDIANTE ACTIVO: Este proceso permite visualizar la constancia de un estudiante estudiante.

GENERAR CONSULTA: Este proceso permite generar una determinada consulta.

SOLICITUD _ CONSULTA: Se usa como entrada en el proceso generar consultas; la cual se descompone en los siguientes: Consulta Nota Estudiante, Consulta Índice Académico, Consulta Estudiantes Reprobados, Consulta Cantidad Estudiantes Becas, Consultar Exoneración Matricula.

CONSULTA: Se usa como salida en el proceso generar consultas.

CONSULTA NOTA ESTUDIANTE: Este proceso permite generar la consulta de la nota del estudiante.

NOTA_EST: Nombre + Asignatura + nota. Se usa como entrada en el proceso Consulta Nota Estudiante.

NOTA: Nombre + Asignatura + nota. Se usa como salida en el proceso Consulta Nota Estudiante.

ÍNDICE _ ACADÉMICO: Se usa como entrada en el proceso Consulta Índice Académico; la cual se descompone en los siguientes: Mejor Índice Académico Carrera, Índice Académico Semestre.

INDICE: Se usa como salida en el proceso Consulta Índice Académico.

INDICE ACADEMICO CARRERA: Este proceso permite consultar los índices académicos por carrera.

MEJOR_IA_CARRERA: Nombres + carrera + año + índice académico. Se usa como entrada en el proceso Mejor Índice Académico por carrera.

MEJOR _ CARRERA: Nombres + carrera + año + índice académico. Se usa como salida en el proceso Mejor Índice Académico por carrera.

INDICE ACADEMICO AÑO: Este proceso permite consultar los mejores índices académicos por año.

IA_AÑO: carrera +nombres + apellidos + IA + Año. Se usa como entrada en el proceso Índice académico año.

INDICE _ AÑO: carrera +nombres + apellidos + IA + Año. Se usa como salida en el proceso Índice Académico año.

EST_REPROBADOS: Se usa como entrada en el proceso Consulta Estudiantes Reprobados, la cual se descompone en: Estudiantes Reprobaron Asignatura, Estudiantes Reprobaron Carrera.

REPROBADOS: Se usa como salida en el proceso Consulta Estudiantes Reprobados.

ESTUDIANTES REPROBARON ASIGNATURA: Este proceso permite consultar los estudiantes que reprobaron por asignatura.

EST_REP_ASIG: Nombre + carrera + Asignatura + nota. Se usa como entrada en el proceso Estudiantes Reprobaron Asignatura.

REPROBARON_ASIGNATURA: Nombre + carrera + Asignatura + nota. Se usa como salida en el proceso Estudiantes Reprobaron Asignatura.

ESTUDIANTES REPROBARON CARRERA: Esta proceso permite consultar los estudiantes que reprobaron por carrera.

EST_REP_CARRERA: nombres + asignaturas de esa carrera reprobada + nota. Se usa como entrada en el proceso Estudiantes Reprobaron Carrera.

REPROBARON_CARRERA: nombres + asignaturas de esa carrera reprobada + nota. Se usa como salida en el Estudiantes Reprobaron Carrera.

GENERAR REPORTE: Este proceso permite generar un determinado reporte.

SOLICITUD REPORTE: Se usa como entrada en el proceso generar reporte, el cual se descompone en los siguientes: Reporte Nota Estudiante Semestral, Reporte Nota Estudiante anual.

REPORTE: Se usa como salida en el proceso generar reportes.

REPORTE NOTA ESTUDIANTE SEMESTRAL: Este proceso permite sacar un reporte de las notas semestrales del estudiante.

REPORTE_NOTA_SEMESTRAL: nombres + asignaturas + notas + IA. Se usa como entrada en el proceso Reporte Nota Estudiante Semestral.

NOTA _ SEMESTRAL: nombres + asignaturas + notas + IA. Se usa como salida en el proceso Reporte Nota Estudiante Semestral.

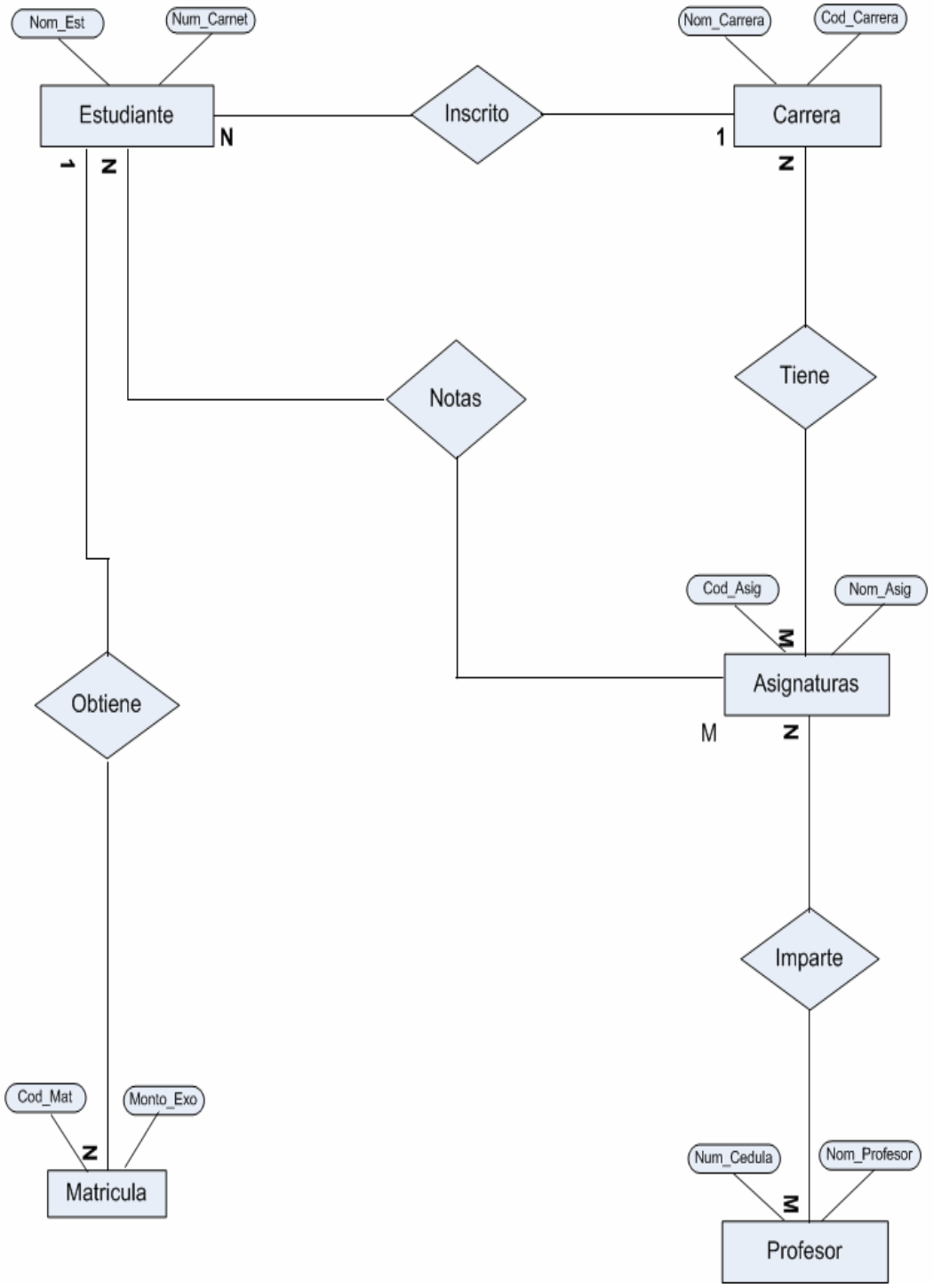
REPORTE NOTA ESTUDIANTE ANUAL: Este proceso permite sacar un reporte de las notas anuales del estudiante.

REPORTE NOTA _ ANUAL: nombres + asignaturas + notas + IA. Se usa como entrada en el proceso Reporte Nota Estudiante Anual.

NOTA _ ANUAL: nombres + asignaturas + notas + IA. Se usa como salida en el proceso Reporte Nota Estudiante Anual.

X-Diseño Entidad-Relación.

Diagrama Entidad Relación



XI-DISEÑO DE DATOS

Tabla Asignaturas:

Nombre _ campo	Tipo _ dato	Longitud	Descripción
Cod_Asig	Char	9	Código de la asignatura.
Nom_Asig	varchar	50	Nombre de la asignatura.
Semestre	integer		Semestre que cursa el estudiante.
Año_Pensum	integer		Año que inicio la carrera y que se imparte la asignatura.

Tabla Carrera:

Nombre _ campo	Tipo_Datos	Longitud	Descripción
Cod_Carrera	Char	9	Código de carrera.
Nom_Carrera	varchar	30	Nombre de la carrera

Tabla Estudiante:

Nombre_Campo	Tipo_Datos	Longitud	Descripcion
Num_Carnet	varchar	30	Numero de carnet del estudiante.
Cod_Carrera	char	9	Código de la carrera del estudiante.
Nom_Est	varchar	50	Nombre del estudiante
Apellidos	varchar	50	Apellidos del estudiante.
Direccion	varchar	80	Dirección del estudiante.
Num_Cedula	varchar	30	Cedula del estudiante.
Procedencia	varchar	20	Ciudad de origen del estudiante.
Año_Estudio	Integer		Año de estudio que cursa el estudiante.
Teléfono	char	9	Teléfono del estudiante.
Fecha_Nac	varchar	20	Fecha que nació el estudiante.
Sexo	char	1	Sexo del estudiante.
Estado_Civil	varchar	9	Estado civil del estudiante.
Tipo_Beca	varchar	10	Tipo de beca, interna, externa.

Tabla Imparte:

Nombre_Campo	Tipo_Datos	Longitud	Descripcion
Num_Cedula	Varchar	30	Numero de cedula del profesor.
Cod_Asig	Varchar	9	Código de la asignatura.

Tabla Matricula:

Nombre_Campo	Tipo_Datos	Longitud	Descripcion
Cod_Mat	Integer	9	Código de matricula.
Num_Carnet	varchar	30	Numero de carnet del estudiante.
Monto_Exo	integer		Monto de exoneración de matricula.
Situacion_Academica	varchar	20	Situación académica, reingreso, Nuevo ingreso, repitente.
Año_Lectivo	char	4	Año lectivo.

Tabla Notas:

Nombre_Campo	Tipo_Datos	Longitud	Descripcion
Num_Carnet	varchar	30	Número de carnet del estudiante
Cod_Asig	char	9	Código de la asignatura.
IP	Integer		Primer parcial nota.
IIP	Integer		Segundo parcial nota.
Prom.	Integer		Promedio Académico.
EF	Integer		Examen final.
NF	Integer		Nota final.
EE	Integer		Examen especial.
CV	Integer		Curso de verano.
TT	Integer		Tutoría.
IA	Integer		Índice Académico del estudiante.

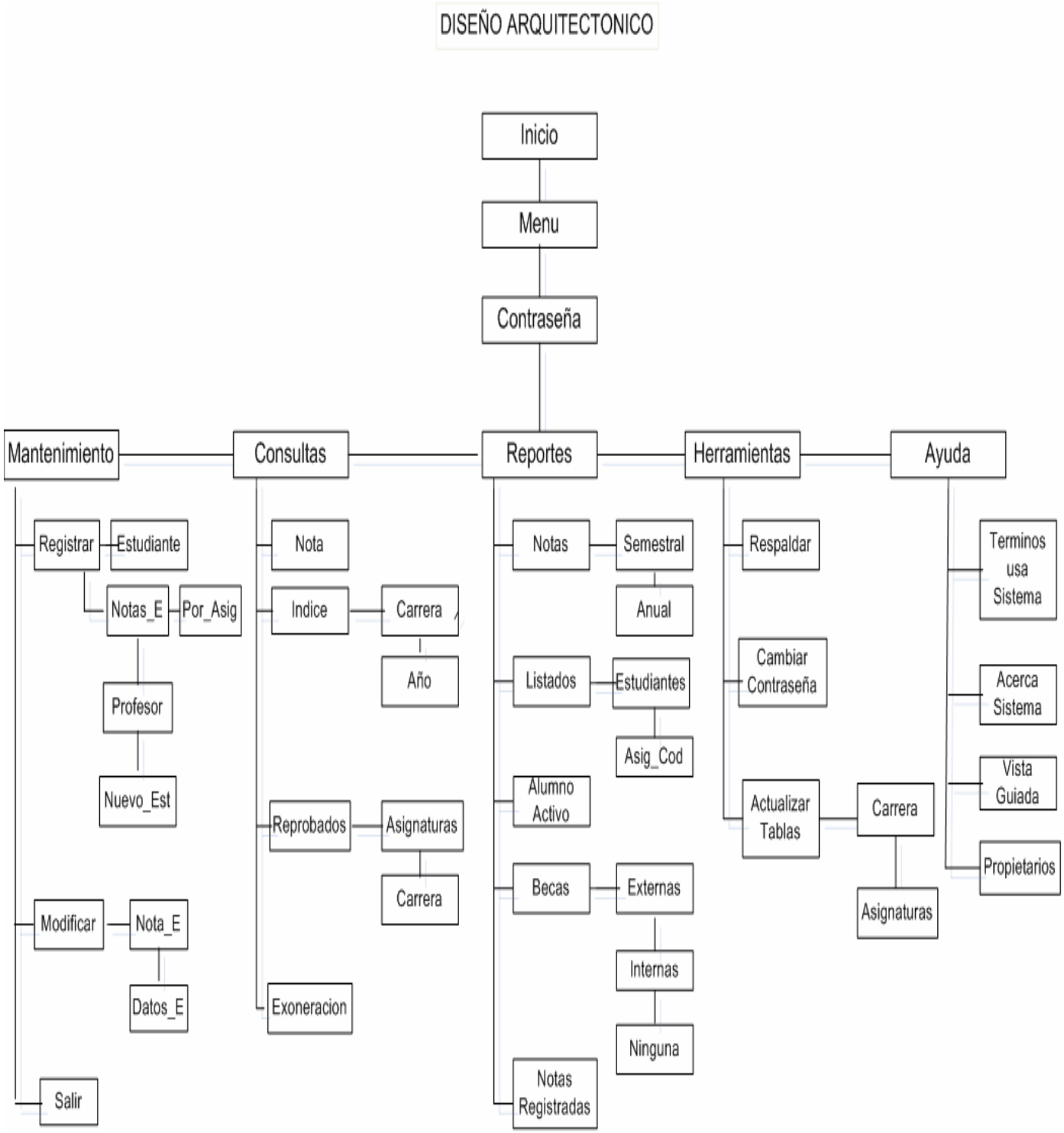
Tabla Profesor:

Nombre_Campo	Tipo_Datos	Longitud	Descripcion
Num_Cedula	char	30	Cedula del profesor
Nom_Profesor	varchar	50	Nombre del profesor
Apellidos	varchar	50	Apellidos del profesor
Direccion	varchar	80	Direccion del profesor
Teléfono	char	9	Numero de teléfono del profesor
Procedencia	varchar	30	Ciudad de origen del profesor

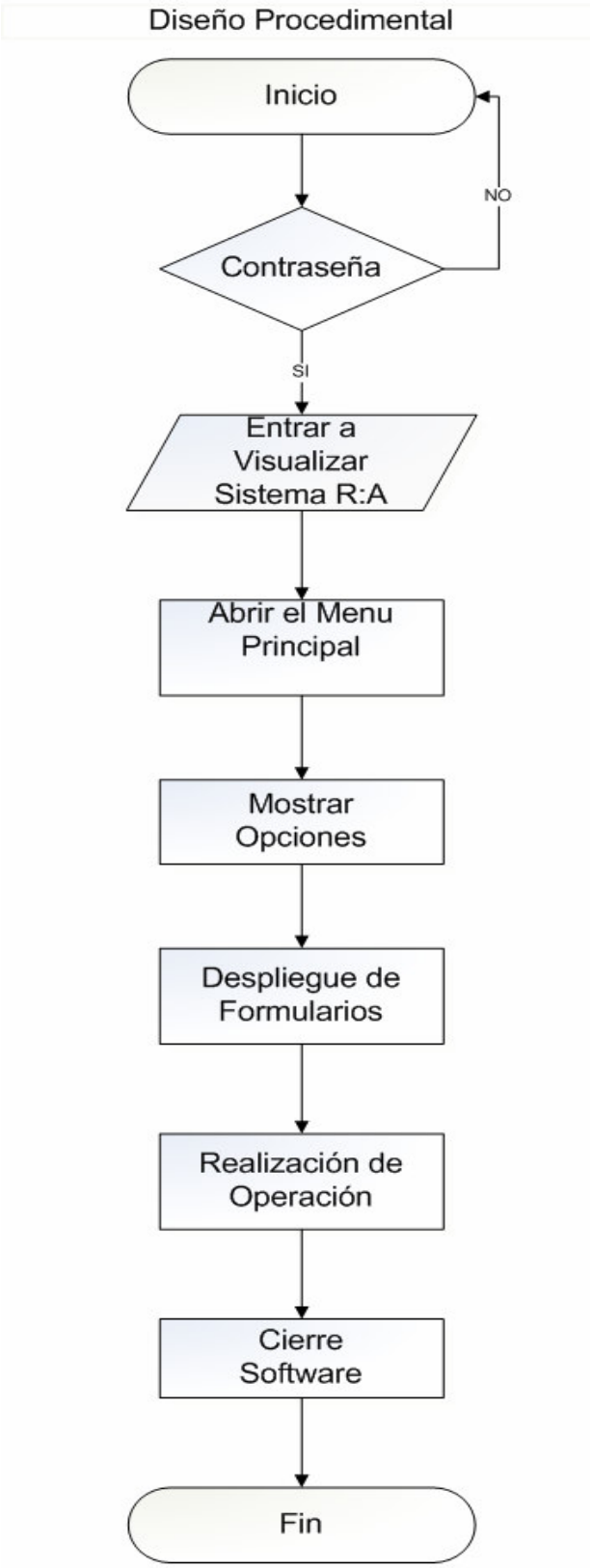
Tabla Tiene:

Nombre_Campo	Tipo_Datos	Longitud	Descripcion
Cod_Carrera	char	9	Código de la carrera.
Cod_Asig	char	9	Código de la asignatura.

XII-DISEÑO ARQUITECTONICO



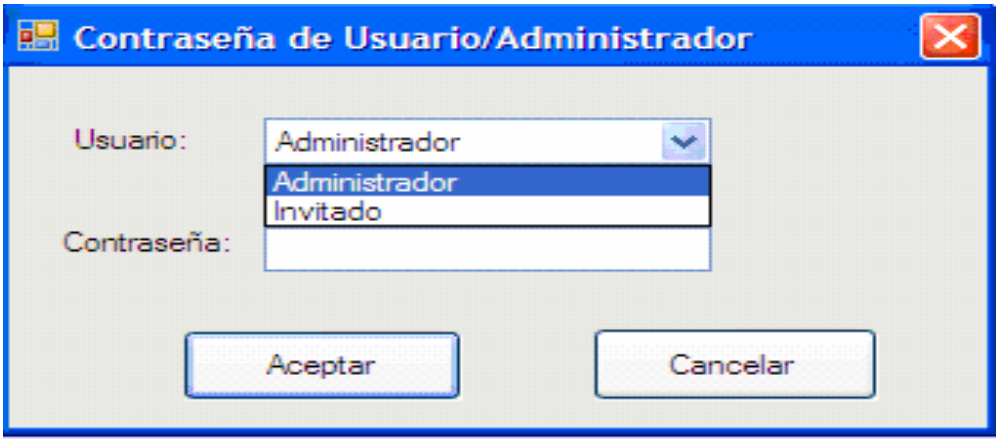
XIII-DISEÑO PROCEDIMENTAL



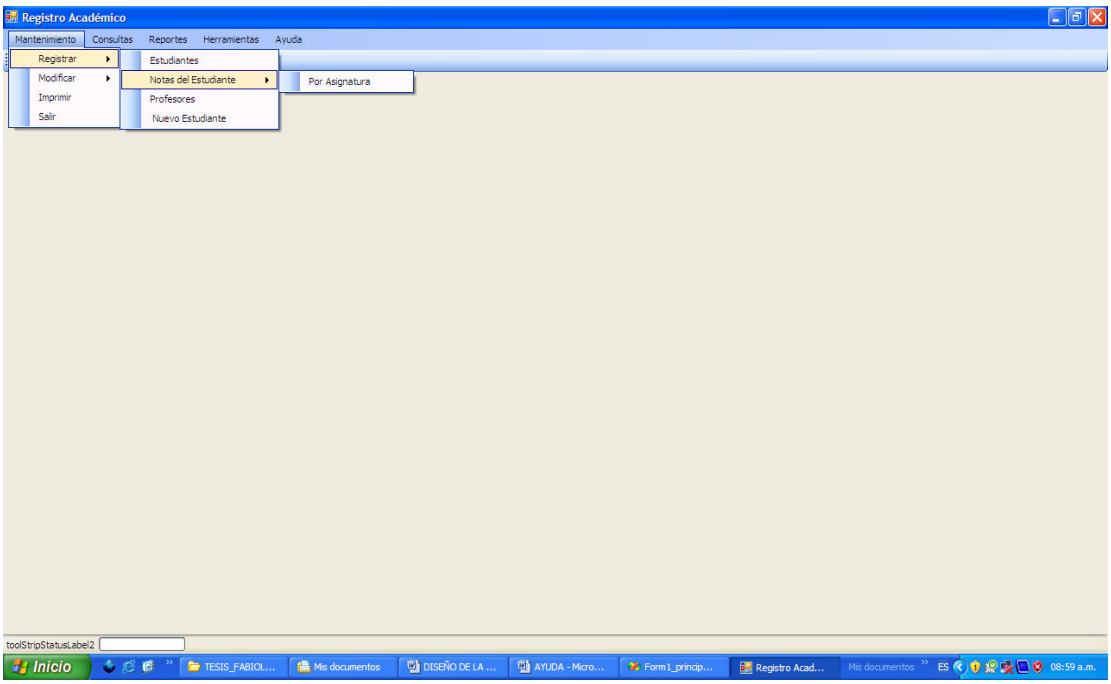
XIV-DISEÑO DE INTERFAZ

DISEÑO DE LA INTERFAZ

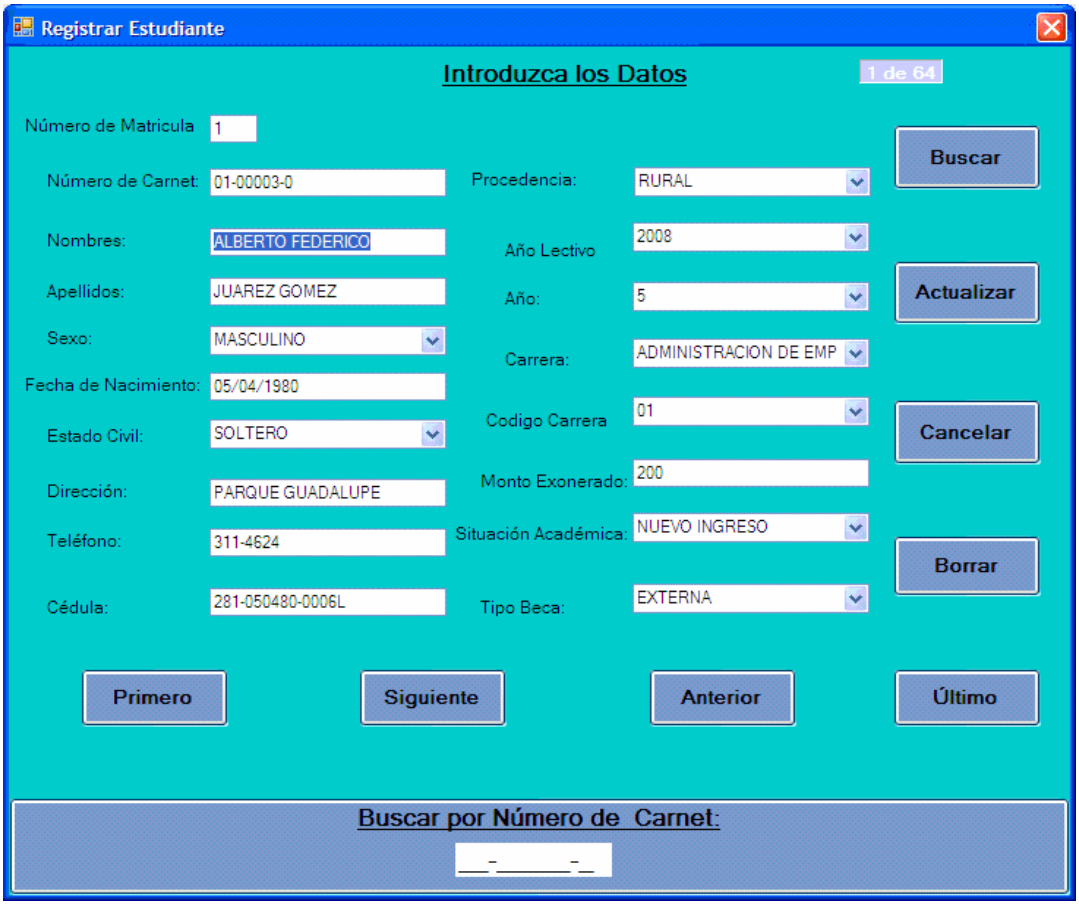
A continuación se muestran algunas de las interfaces (Formularios del software).
Se muestra la ventana de inicio de sesión de nuestro software "Registro Académico de la Facultad Ciencias Económicas y Empresarial".



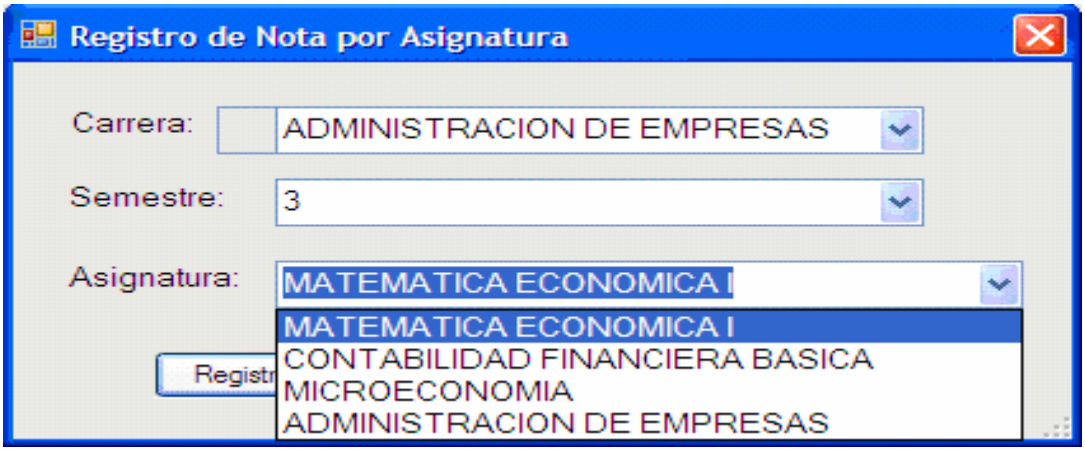
La ventana principal de nuestro software, la cual cuenta con un menú en la parte superior de la misma donde el usuario tendrá la opción de elegir una alternativa a utilizar:



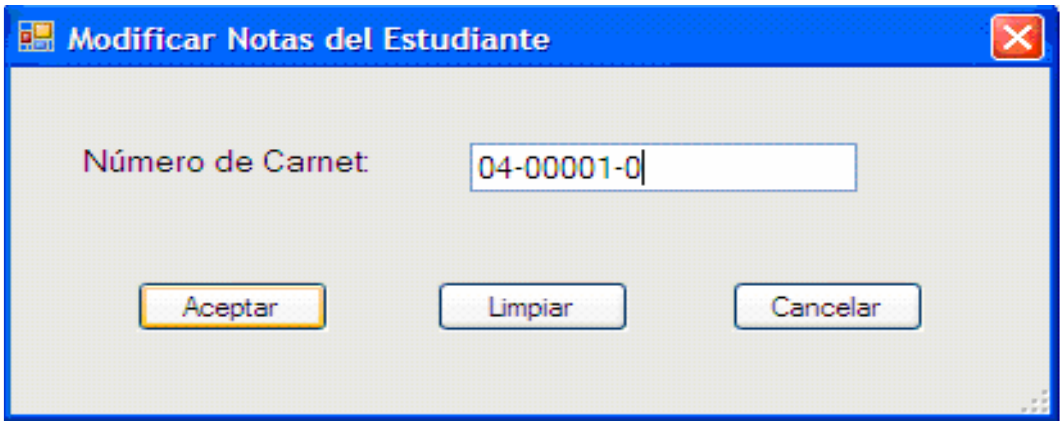
El menú Mantenimiento, el cual cuenta con un submenú Registrar Estudiante. Esta ventana es una de las más importantes ya que permite ver el registro de estudiantes.



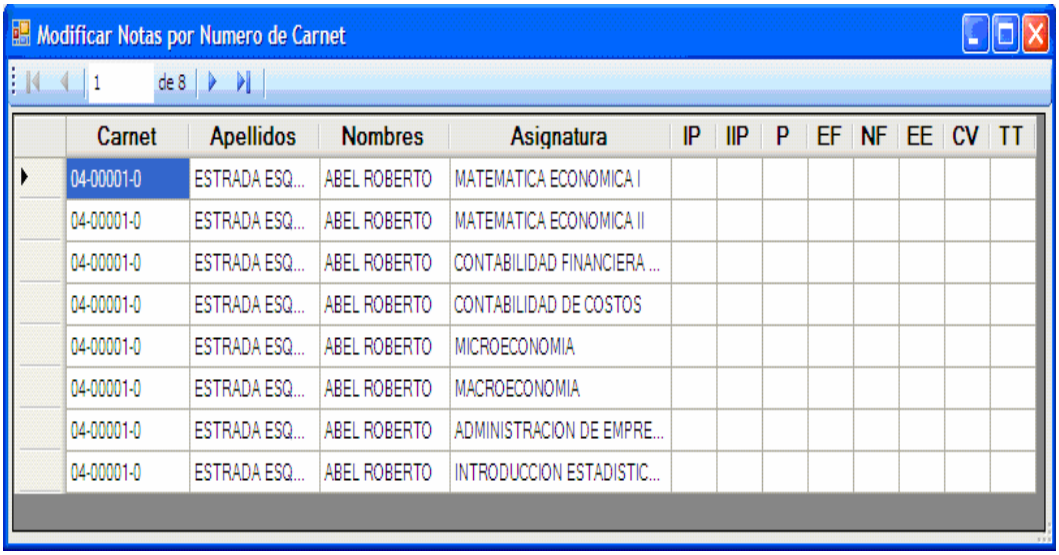
Ventana de captura que permite Registrar notas de una determinada asignatura:



Ventana de captura que permite modificar las notas de un estudiante.

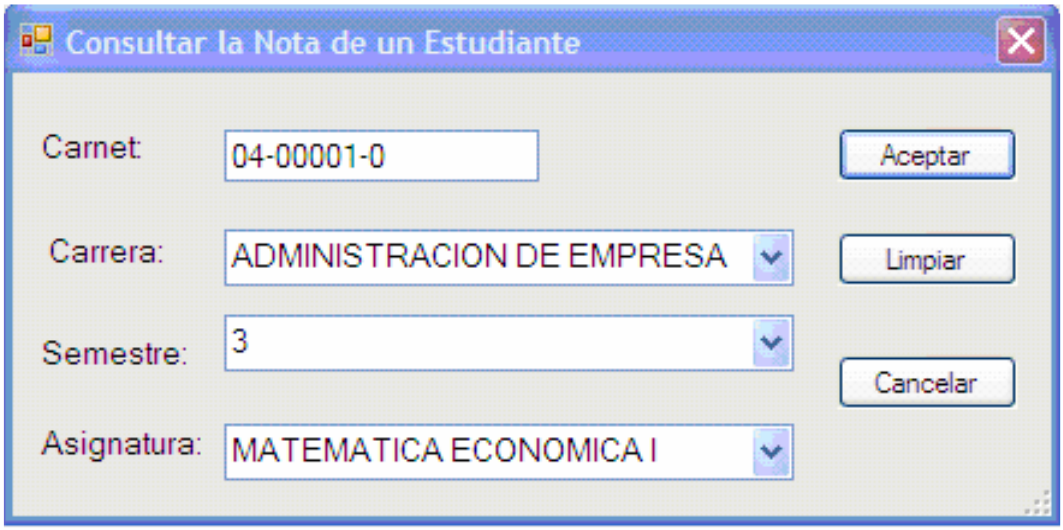


Ventana que se emite cuando el usuario introduce en el formulario Modificar Notas del Estudiante el Número de Carnet del estudiante.

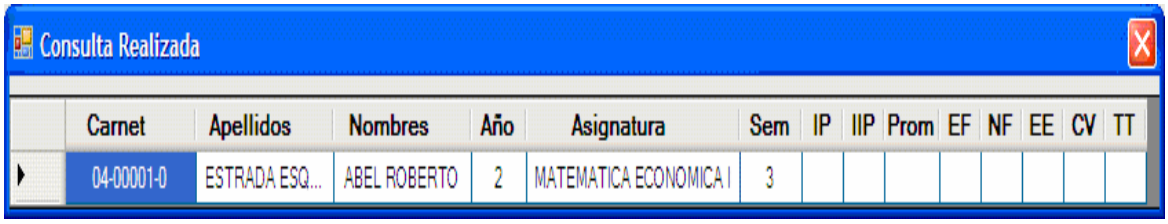


Carnet	Apellidos	Nombres	Asignatura	IP	IIP	P	EF	NF	EE	CV	TT
04-00001-0	ESTRADA ESQ...	ABEL ROBERTO	MATEMATICA ECONOMICA I								
04-00001-0	ESTRADA ESQ...	ABEL ROBERTO	MATEMATICA ECONOMICA II								
04-00001-0	ESTRADA ESQ...	ABEL ROBERTO	CONTABILIDAD FINANCIERA ...								
04-00001-0	ESTRADA ESQ...	ABEL ROBERTO	CONTABILIDAD DE COSTOS								
04-00001-0	ESTRADA ESQ...	ABEL ROBERTO	MICROECONOMIA								
04-00001-0	ESTRADA ESQ...	ABEL ROBERTO	MACROECONOMIA								
04-00001-0	ESTRADA ESQ...	ABEL ROBERTO	ADMINISTRACION DE EMPRE...								
04-00001-0	ESTRADA ESQ...	ABEL ROBERTO	INTRODUCCION ESTADISTIC...								

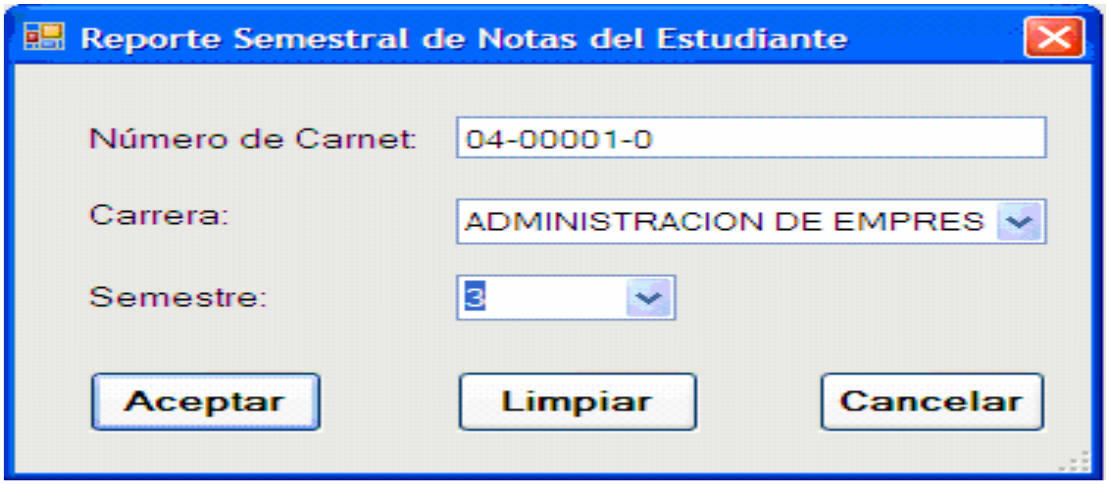
Ventana de captura para consultar la nota del estudiante.



Ventana que se emite cuando el usuario ha llenado el formulario consultar la nota del estudiante al que se le va a modificar la nota.

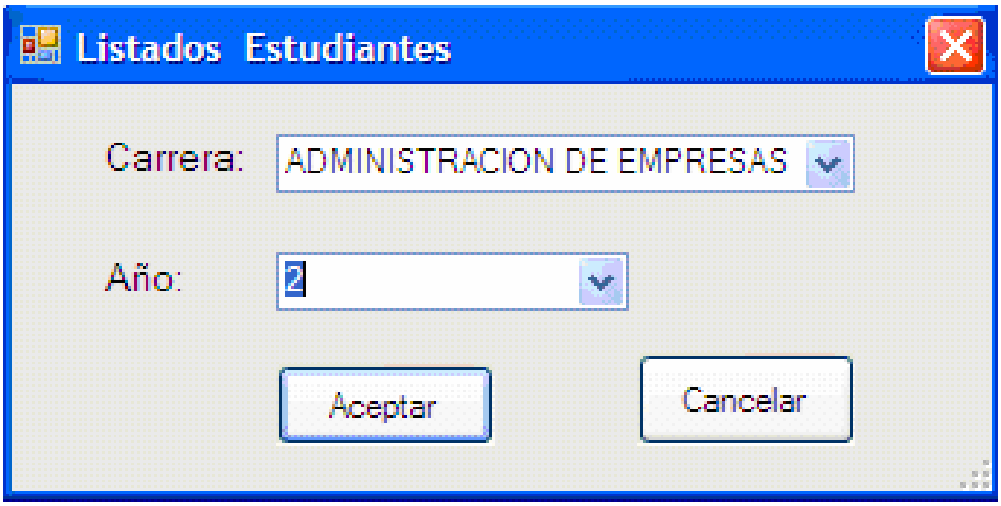


Reporte Semestral de Notas del Estudiante:



Ventana que se emite cuando el usuario ha llenado el formulario reporte semestral de notas del estudiante.

Ventana de captura que muestra el listado de Estudiantes por Carrera y Año:



Reporte de Listados de Estudiantes:

Reporte Listados Estudiantes

1 de 1 100% Buscar

UNIVERSIDAD NACIONAL AUTONOMA DE NICARAGUA(UNAN-LEÓN)

FACULTAD DE CIENCIAS ECONÓMICAS Y EMPRESARIALES.

LISTADO DE ESTUDIANTES.

ADMINISTRACION DE EMPRESAS

Año: 2

Carnet	Apellidos	Nombres
04-00001-0	ESTRADA ESQUIVEL	ABEL ROBERTO
04-00002-0	PEREZ ALTAMIRANO	PEDRO JOAQUIN
04-00003-0	PEVEDA FLORES	ANA MARIA
04-00004-0	GARCIA PEREZ	CARLA PATRICIA
04-00005-0	RODRIGUEZ BARRETO	MARIA MERCEDES

Ventana de captura para Actualizar la tabla Carrera:

Actualizar Carrera

1 de 5

Codigo de Carrera	Nombre carrera
01	ADMINISTRACION DE EMPRES...
02	ECONOMIA
03	CONTADURIA PUBLICA Y FINA...
04	MERCADOTECNIA
05	TURISMO

Ventana de captura que permite visualizar los estudiantes exonerados de matricula.

Consultar Estudiantes Exoneración Matrícula

Exonerados de Matrícula

Carnet	Apellidos	Nombres	Monto - Exo	Año	Carrera
01-00004-0	TORUÑO MONACADA	KARLA VANESA	300	5	ADMINISTRACION D...
01-00005-0	TORREZ GARCIA	JORGE DANILO	400	5	ADMINISTRACION D...
01-00007-0	NEIRA MONCADA	CLAUDIA IVETH	400	5	ADMINISTRACION D...
02-00005-0	JIMENEZ NUÑEZ	BAYARDO ANTONIO	300	4	ADMINISTRACION D...
02-00008-0	RUIZ FLORES	MARBELY ESTELA	300	4	ADMINISTRACION D...
03-00001-0	MAYORGA ESQUIVEL	ERENIA DE LOS ANG...	300	3	ADMINISTRACION D...
03-00002-0	JUAREZ GARCIA	MARCIA URANIA	400	3	ADMINISTRACION D...
03-00003-0	SALINAS MOLINA	AMIN FRANCISCO	200	3	ADMINISTRACION D...
03-00004-0	GONZALES VACA	LARRY ALAN	200	3	ADMINISTRACION D...
03-00005-0	LEON ZAPATA	PEDRO ANTONIO	300	3	ADMINISTRACION D...
04-00001-0	ESTRADA ESQUIVEL	ABEL ROBERTO	200	2	ADMINISTRACION D...
04-00002-0	PEREZ ALTAMIRANO	PEDRO JOAQUIN	200	2	ADMINISTRACION D...
04-00003-0	PEVEDA FLORES	ANA MARIA	300	2	ADMINISTRACION D...
04-00004-0	GARCIA PEREZ	CARLA PATRICIA	400	2	ADMINISTRACION D...
04-00005-0	RODRIGUEZ BARRE...	MARIA MERCEDES	400	2	ADMINISTRACION D...
01-00019-0	RODRIGUEZ OLIVAS	JOSE ANTONIO	300	5	CONTADURIA PUBLI...
02-00022-0	RUIZ ORTIZ	REBECA NOHEMI	300	4	CONTADURIA PUBLI...
03-00017-0	ALVAREZ ALFARO	CAROLINA PAOLA	300	3	CONTADURIA PUBLI...
04-00016-0	RODRIGUEZ FERNA...	LUIS JAIME	400	2	CONTADURIA PUBLI...
04-00020-0	CASTILLO MEDINA	LAURA PATRICIA	200	2	CONTADURIA PUBLI...
01-00011-0	RUIZ VELASQUEZ	NORMAN MARCELO	300	5	ECONOMIA
01-00012-0	CASTELLON HERNA...	AURA IRIS	200	5	ECONOMIA
01-00014-0	ACOSTA GALO	BRENDA ALI	400	5	ECONOMIA
01-00015-0	RUIZ BLANCO	DENIS ENRIQUE	400	5	ECONOMIA
02-00014-0	RAMIREZ TALAVERA	INGRID BENITA	300	4	ECONOMIA

XV-CONCLUSIONES

Con la elaboración de nuestro trabajo monográfico hemos confirmado que el Registro Académico es un área de mucha importancia para cada una de las facultades de una universidad por lo que hay que tener mucha precaución y exigencia para diseñar el software.

Nuestro sistema ha sido realizado de tal manera que se debe tener mucho cuidado a la hora de manipularlo ya que debe ser una persona que tenga conocimientos necesarios para manejar la información tan importante como lo es el registro de notas.

Con el desarrollo de nuestra aplicación consideramos que hemos cumplido satisfactoriamente con los objetivos planteados, ya que se ha facilitado para el usuario manejar el sistema, la información se podrá actualizar de forma permanente, haciendo todo en menos tiempo, y automatizando así las gestiones de registro académico, ahorrándose dinero y disminuyendo el trabajo para la facultad.

C # es un lenguaje seguro y elegante que permite que crear aplicaciones seguras y robustas, ha sido un lenguaje de gran importancia.

Es eficiente y compatible con otros lenguajes de programación.

El diseñar una aplicación en lenguaje visual C # y adquirir todos los conocimientos necesarios nos ha permitido comenzar un nuevo diseño de una aplicación para automatizar cada vez más el registro académico de dicha facultad.

XVI-RECOMENDACIONES

El sistema que hemos realizado debe ser manejado por una persona que tenga conocimientos sobre informática y el Registro de Notas de dicha facultad para que el sistema tenga un correcto funcionamiento y de esta manera obtener los resultados esperados.

Debido que en Registro Académico se lleva un manejo importante de notas de los estudiantes se recomienda hacer un respaldo de la base de datos de forma semestral.

Para el buen funcionamiento del software debe cumplir con los requerimientos aportados en restricciones generales.

Las autoridades correspondientes deben de proveer a los programadores, la información necesaria para el diseño del software, en tiempo y forma, y así evitar atrasos innecesarios, que hacen perder tiempo y dinero tanto a la empresa (universidad) como a los responsables del sistema.

XVII-ANEXOS

XVIII-BIBLIOGRAFIA

Bibliografía:

- **(MySQL 5.0 Reference Manual) Manual de MYSQL.**

  http://www.dev.mysql.com.

  http://www.mysql.com/documentation/index.html

  http://dev.mysql.com/doc/

  http://www.mysql.com/

  (http://downloads.mysql.com/docs/menagerie.tar.gz)

  (http://downloads.mysql.com/docs/menagerie.zip)

- **Bases de Datos.**

  http://es.wikipedia.org/wiki/Base_de_datos

- **Análisis y Diseño de Sistemas**

  http://window.to/concepcion.com.do

  - Análisis y Diseño de Sistemas, Henry f. korth & Abraham Silberschatz, Segunda Edición, Editorial MC Graw Hill

- **Lenguaje de Programación**

  - Enciclopedia de Microsoft Visual C # de Francisco Javier Ceballos.

INDICE

Agradecimiento

A Dios padre y a su hijo Jesucristo por darnos la alegría de poder culminar nuestra carrera, por ser la fuente de nuestra inspiración en cada momento de nuestra vida y regalarnos fortaleza, fé y esperanza en cada situación difícil que se nos presentó.

MSc. Mario Alberto Talavera que a través de su vasto conocimiento nos mostró su gran apoyo incondicional, pero sobre todo su gran dedicación y esmero en la realización de nuestra monografía.

MSc. Álvaro Altamirano, por sus enseñanzas como excelente profesor demostrando siempre su alto nivel de conocimiento, ante todo su amistad y cariño durante todo este tiempo.

A nuestro profesor, tutor, y amigo **MSc. Danilo Panilla**, que con su ayuda, experiencia y aprendizaje hemos llegado a la finalización de este trabajo.

A los **profesores** por tenernos paciencia y compartir sus conocimientos con nosotros.

Rebeca Nohemí Ruiz Ortiz.

Ana Fabiola Rodríguez Pinell

Dedicatoria

A **Dios Padre** por brindarme la oportunidad de culminar mi carrera, dándole gracias por seguir adelante día a día para lograr ser una mejor persona.

A mi Padre **José de Jesús Rodríguez Aráuz**, quien con su amor y esfuerzo he llegado a cumplir mi sueño, agradeciéndole su paciencia y comprensión que ha tenido desde mi niñez hasta hoy en día y por su apoyo en cada momento de mi vida (Gracias Papá).

A la memoria infinita de mi querida y adorada Madre **Sra. Mirtha Nelly Pinell de Rodríguez** (q.e.p.d) por darme la vida, por haber sido una madre incomparable, quién un día soñó en que finalizaría mi carrera y con tanto sacrificio y amor incondicional logró prepararme unos días antes de su partida, viajando al infinito muy satisfecha por lograr sus objetivos hacia mi, lo cuál yo agradezco donde quiera que estés, por que se me darás siempre tu bendición (Te amo Mamá).

A mi hermano **Freddy** por su apoyo incondicional, por su sacrificio y confianza, que a pesar de su gran responsabilidad cree en mí. (Gracias Lindo).

A mis demás hermanos, **Agustina, Dalila, Manuel y Juany** quienes me impulsaron con sus palabras de aliento a seguir adelante.

A todos ellos gracias por su amor y comprensión.

Br. Ana Fabiola Rodríguez Pinell.

A mi madre, **Angela Ortiz**, por todo su esfuerzo y lucha en todos estos años, en donde sus palabras de aliento nunca faltaron , el que me escucharas era para mí alegría y ánimo, pero sobre todo por su gran amor y dedicacion, gracias mamá, tu sabes que te amo mucho.

A mi abuelita, **Angela Ruiz Hernandez,** quien me enseño a tener presente a Dios en todos los momentos de mi vida, fuiste como una madre para mí, con tu amor me enseñastes a amar, hoy ya no estas presente, pero se que un día, te veré, ahora solo me quedan los bellos recuerdos que contigo viví, siempre estarás dentro de mí corazón.

A la memoria de mi padre, **Pablo René Ruiz Hernández**, por enseñarme que lo más importante no es lo material, sino lo que esta dentro del alma de cada ser humano, por tu virtudes de servicio y sencillez aprendi que se puede ser feliz; siempre te recordaré, con mucho amor.

A mi hermana, **Jessy Enoe Ruiz Ortiz**, por animarme, preocuparse por mí en cada etapa de mi vida, tus palabras y tu amor han sido para mi como un tesoro invaluable.

Con amor, su hija, nieta y hermana,

Rebeca Nohemi Ruiz Ortiz.

ANEXOS

CODIGO:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using MySql.Data.MySqlClient;

namespace Form1_principal
{
    public partial class Form2_Reg_Estudiante : Form
    {
        private MySqlConnection conn;
        private MySqlCommand comand_Est, comand_Mat, comand,comand_Notas;

        public static string criterioF2NCarnet, criterioF2CodCarrera,
        Comience2;
        public string Sem1 = "", Sem2 = "";
        string COD_ASIG2 = "";
        int f2_conta = 1;

        MySqlDataAdapter dataadapterEstudiante, dataadapterMatricula,
        dataadapterNotas;

        DataSet datasetEstudiante, datasetMatricula, datasetNotas;
        BindingSource bindingsourceEstudiante, bindingsourceMatricula;

        public Form2_Reg_Estudiante()
        {
            string conexion = ObtenerCadenaDeConexion();
            conn = new MySqlConnection(conexion);
            InitializeComponent();
            CargarRegistros();
        }

        private string ObtenerCadenaDeConexion()
        {
            string conexion = "";
            string server = "localhost";
            string bd = "Registro2";
            string user = "root";
            string pass = "root";

            conexion = "Server=" + server + ";Database=" + bd + ";Uid=" +
            user + ";Pwd=" + pass;
            return conexion;
        }



        public void CargarRegistros()
        {
```

```
// Abro la conexion
conn.Open();

// Establezco la orden a ejecutar
string codCarrera = F2_Cod_carrera_comboBox1.Text;
string comando_Est = "SELECT * FROM  Estudiante";
string comando_Mat = "SELECT * FROM  Matricula ";
string comando_Nota = "SELECT * FROM Notas";

// Creo el conjunto de datos
datasetEstudiante = new DataSet();
datasetMatricula = new DataSet();
datasetNotas = new DataSet();

//Creo el adaptador que me permita conectarme BD en modo
desconectado

dataadapterEstudiante = new MySqlDataAdapter();
dataadapterMatricula = new MySqlDataAdapter();
dataadapterNotas = new MySqlDataAdapter();

//Creo el comando para luego relacionarlo con el adaptador
comand_Est = new MySqlCommand(comando_Est, conn);
comand_Mat = new MySqlCommand(comando_Mat, conn);
comand_Notas = new MySqlCommand(comando_Nota, conn);

//Asocio el comando con el adaptador
dataadapterEstudiante.SelectCommand = comand_Est;
dataadapterMatricula.SelectCommand = comand_Mat;
dataadapterNotas.SelectCommand = comand_Notas;

// Lleno el conjunto de datos con el resultado del comando
dataadapterEstudiante.Fill(datasetEstudiante, "Estudiante");
dataadapterMatricula.Fill(datasetMatricula, "Matricula");
dataadapterNotas.Fill(datasetNotas, "Notas");

//Tabla Estudiante
bindingsourceEstudiante = new BindingSource();
bindingsourceEstudiante.DataSource = datasetEstudiante;
bindingsourceEstudiante.DataMember = "Estudiante";

F2_numero_carnet_maskedTextBox1.DataBindings.Add(new
Binding("Text", bindingsourceEstudiante, "Num_Carnet",
true));

F2_nombres_textBox2.DataBindings.Add(new Binding("Text",
bindingsourceEstudiante, "Nom_Est", true));

F2_apellidos_textBox3.DataBindings.Add(new Binding("Text",
bindingsourceEstudiante, "Apellidos", true));

F2_sexo_comboBox1.DataBindings.Add(new Binding("Text",
bindingsourceEstudiante, "Sexo", true));

F2_fecha_nac_maskedTextBox1.DataBindings.Add(new
Binding("Text",bindingsourceEstudiante,"Fecha_Nac",true));
```

```
            F2_Estado_civil_comboBox1.DataBindings.Add(new
            Binding("Text", bindingsourceEstudiante, "Estado_Civil",
            true));

            F2_direccion_textBox7.DataBindings.Add(new Binding("Text",
            bindingsourceEstudiante, "Direccion", true));

            F2_telefono_maskedTextBox1.DataBindings.Add(new
            Binding("Text", bindingsourceEstudiante, "Telefono", true));

            F2_cedula_maskedTextBox1.DataBindings.Add(new Binding("Text",
            bindingsourceEstudiante, "Num_Cedula", true));

            F2_procedencia_comboBox1.DataBindings.Add(new Binding("Text",
            bindingsourceEstudiante, "Procedencia", true));

            F2_año_comboBox1.DataBindings.Add(new Binding("Text",
            bindingsourceEstudiante, "Ano_Estudio", false));

            // Tabla matricula
            bindingsourceMatricula = new BindingSource();
            bindingsourceMatricula.DataSource = datasetMatricula;
            bindingsourceMatricula.DataMember = "Matricula";

            F2_Monto_Exo_maskedTextBox1.DataBindings.Add(new
            Binding("Text", bindingsourceMatricula, "Monto_Exo", true));

            F2_Cod_Mat_textBox20.DataBindings.Add(new Binding("Text",
            bindingsourceMatricula, "Cod_Mat", true));

            F2_año_lectivo_comboBox1.DataBindings.Add(new Binding("Text",
            bindingsourceMatricula, "Ano_Lectivo", true));

            F2_situacion_academica_comboBox1.DataBindings.Add(new
            Binding("Text", bindingsourceMatricula,
            "Situacion_Academica", true));

            F2_Tipo_Beca_comboBox17.DataBindings.Add(new Binding("Text",
            bindingsourceEstudiante, "Tipo_Beca", true));

            F2_Cod_carrera_comboBox1.DataBindings.Add(new Binding("Text",
            bindingsourceEstudiante, "Cod_Carrera", true));
        }

    private void F2_buscar_button4_Click(object sender, EventArgs e)
        {
            Form2_1Buscar_N_Carnet buscar_NCarnet = new
            Form2_1Buscar_N_Carnet();
            buscar_NCarnet.ShowDialog();
        }
    private void F2_buscar2_button1_Click(object sender, EventArgs e)
        {
            String carnet_est = F2_BuscarCarnet_maskedTextBox1.Text;
            bool res_carnet = false;
```

```
            for (int i=0;i<datasetEstudiante.Tables[0].Rows.Count; i++)
            {
if(datasetEstudiante.Tables[0].Rows[i][0].ToString()==carnet_est)
                    {
                        F2_numero_carnet_maskedTextBox1.Text =
carnet_est;
                        F2_nombres_textBox2.Text =
datasetEstudiante.Tables[0].Rows[i][2].ToString();
                        F2_apellidos_textBox3.Text =
datasetEstudiante.Tables[0].Rows[i][3].ToString();
                        F2_direccion_textBox7.Text =
datasetEstudiante.Tables[0].Rows[i][4].ToString();
                        F2_cedula_maskedTextBox1.Text =
datasetEstudiante.Tables[0].Rows[i][5].ToString();
                        F2_procedencia_comboBox1.Text =
datasetEstudiante.Tables[0].Rows[i][6].ToString();
                        F2_año_comboBox1.Text =
datasetEstudiante.Tables[0].Rows[i][7].ToString();
                        F2_telefono_maskedTextBox1.Text =
datasetEstudiante.Tables[0].Rows[i][8].ToString();
                        F2_fecha_nac_maskedTextBox1.Text =
datasetEstudiante.Tables[0].Rows[i][9].ToString();
                        F2_sexo_comboBox1.Text =
datasetEstudiante.Tables[0].Rows[i][10].ToString();
                        F2_Estado_civil_comboBox1.Text =
datasetEstudiante.Tables[0].Rows[i][11].ToString();
                        F2_Tipo_Beca_comboBox17.Text =
datasetEstudiante.Tables[0].Rows[i][12].ToString();
                        BuscarEnMatricula(carnet_est);

                        res_carnet = true;
                        MessageBox.Show("Estudiante Encontrado");
                        bindingsourceEstudiante.Position = i;
                        MostrarPosicion();
                        return;
                    }
                }

            }
            if (res_carnet == false)
            {
                MessageBox.Show("Número de carnet no existe");
                return;
            }
        }

        private void BuscarEnMatricula(string carnet_est)
        {
         for (int i = 0; i < datasetMatricula.Tables[0].Rows.Count; i++)
            {
      if (datasetMatricula.Tables[0].Rows[i][1].ToString() == carnet_est)
                {
                    F2_Cod_Mat_textBox20.Text =
datasetMatricula.Tables[0].Rows[i][0].ToString();
                    F2_Monto_Exo_maskedTextBox1.Text =
datasetMatricula.Tables[0].Rows[i][2].ToString();
```

```
                    F2_situacion_academica_comboBox1.SelectedItem =
datasetMatricula.Tables[0].Rows[i][3].ToString();
                    F2_año_lectivo_comboBox1.Text =
datasetMatricula.Tables[0].Rows[i][4].ToString();
                    bindingsourceMatricula.Position = i;
                }
            }
        }

    private void F2_Actualizar_button1_Click(object sender,EventArgs e)
        {
            bool res2_carnet = F2_Comparar_N_Carnet();
            if (res2_carnet == true)
            {
                try
                {
string comando1 = "UPDATE Estudiante SET Cod_carrera = ?vCod_carrera,
Nom_Est= ?vNom_Est, Apellidos = ?vApellidos,Direccion= ?vDireccion,
Num_Cedula = ?vNum_Cedula, Procedencia = ?vProcedencia, Ano_Estudio =
?vAno_Estudio, Telefono = ?vTelefono, Fecha_Nac = ?vFecha_Nac, Sexo=
?vSexo, Estado_Civil = ?vEstado_Civil, Tipo_Beca = ?vTipo_Beca WHERE
Num_Carnet = ?vNum_Carnet";

string comando2 = "UPDATE Matricula SET Num_Carnet = ?vNum_Carnet,
Monto_Exo = ?vMonto_Exo, Situacion_Academica = ?vSituacion_Academica,
Ano_Lectivo = ?vAno_Lectivo WHERE Cod_Mat = ?vCod_Mat";

                    comand = new MySqlCommand(comando1, conn);
                    comand.Parameters.Add("?vNum_Carnet",
F2_numero_carnet_maskedTextBox1.Text);
                    comand.Parameters.Add("?vNom_Est",
F2_nombres_textBox2.Text);
                    comand.Parameters.Add("?vApellidos",
F2_apellidos_textBox3.Text);
                    comand.Parameters.Add("?vDireccion",
F2_direccion_textBox7.Text);
                    comand.Parameters.Add("?vNum_Cedula",
F2_cedula_maskedTextBox1.Text);
                    comand.Parameters.Add("?vProcedencia",
F2_procedencia_comboBox1.Text);
                    comand.Parameters.Add("?vAno_Estudio",
F2_año_comboBox1.Text);

                    comand.Parameters.Add("?vCod_Carrera",
F2_Cod_carrera_comboBox1.Text);
                    comand.Parameters.Add("?vTelefono",
F2_telefono_maskedTextBox1.Text);
                    comand.Parameters.Add("?vFecha_Nac",
F2_fecha_nac_maskedTextBox1.Text);
                    comand.Parameters.Add("?vSexo",
F2_sexo_comboBox1.Text);
                    comand.Parameters.Add("?vEstado_Civil",
F2_Estado_civil_comboBox1.Text);
                    comand.Parameters.Add("?vTipo_Beca",
F2_Tipo_Beca_comboBox17.SelectedItem.ToString());
                    comand.Prepare();
```

```
                    comand.ExecuteNonQuery();

                    // Tabla Matricula
    comand = new MySqlCommand(comando2, conn);

comand.Parameters.Add("?vCod_Mat", F2_Cod_Mat_textBox20.Text);
comand.Parameters.Add("?vNum_Carnet",F2_numero_carnet_maskedTextBox1.Text
);
                    comand.Parameters.Add("?vMonto_Exo",
F2_Monto_Exo_maskedTextBox1.Text);
                    comand.Parameters.Add("?vSituacion_Academica",
F2_situacion_academica_comboBox1.SelectedItem.ToString());
                    comand.Parameters.Add("?vAno_Lectivo",
F2_año_lectivo_comboBox1.Text);
                    comand.Prepare();
                    comand.ExecuteNonQuery();

 //METODO PARA INSERTAR EN LA TABLA NOTAS LOS COD/ASIG DE CADA ESTUDIANTE
                    InsetarEnNotas();

MessageBox.Show("Registro   actualizado   correctamente.","Registro   de
Estudiantes", MessageBoxButtons.OK, MessageBoxIcon.Information);

                    criterioF2NCarnet =
this.F2_numero_carnet_maskedTextBox1.Text.ToString();
                    criterioF2CodCarrera =
this.F2_Cod_carrera_comboBox1.Text.ToString();

Form2_Reporte_Reg_Estudiante obj = new Form2_Reporte_Reg_Estudiante();
obj.Show();
 }
                catch (MySqlException exc)
                {
MessageBox.Show("Error  al  actualizar  el  registro  el  registro.  \n"  +
exc.ToString(),   "Registro   de   Estudiantes",   MessageBoxButtons.OK,
MessageBoxIcon.Information);
                }
            }
            else
            {
MessageBox.Show("Ese numero de carnet no existe/ Haga un nuevo Registro
de Estudiantes");
                return;
            }
        }

 private bool F2_Comparar_N_Carnet()
 {
            string carnet_est = F2_numero_carnet_maskedTextBox1.Text;
            bool res2_carnet = false;
            string res2 = "";

    for (int i = 0; i < datasetEstudiante.Tables[0].Rows.Count; i++)
     {
     if (datasetEstudiante.Tables[0].Rows[i][0].ToString() == carnet_est)
         res2 = "si";
```

```
        }


            }
            if (res2 == "si")
            {
                res2_carnet = true;
                return res2_carnet;
            }
            else
                return res2_carnet;
        }


   private void Form2_Reg_Estudiante_Load(object sender, EventArgs e)
        {
// TODO: esta línea de código carga datos en la tabla
'dataSet18_NombreCarrera.DataTable1' Puede moverla o quitarla según sea
necesario.
this.dataTable1TableAdapter1.Fill_NomCarrera(this.dataSet18_NombreCarrera
.DataTable1);

for (int i = 0; i < dataSet18_NombreCarrera.Tables[0].Rows.Count; i++)
F2_carrera_comboBox4.Items.Add(dataSet18_NombreCarrera.Tables[0].Rows[i][
1]);

 // TODO: esta línea de código carga datos en la tabla
'dataSet16_CodAsig.DataTable1' Puede moverla o quitarla según sea
necesario.
this.dataTable1TableAdapter.Fill_Asignaturas(this.dataSet16_CodAsig.DataT
able1);
 MostrarPosicion();
        }

        private void Form2_Reg_Estudiante_FormClosing(object sender,
        FormClosingEventArgs e)
        {
            Form2_Reg_Estudiante f2_salir = new Form2_Reg_Estudiante();
        }
        private void InsetarEnNotas()
        {

string carnet = F2_numero_carnet_maskedTextBox1.Text.ToString();
string comando4 = "DELETE FROM  Notas  WHERE Num_Carnet=?vNum_Carnet";
for (int j = 0; j < datasetNotas.Tables[0].Rows.Count; j++)
        {
  if (datasetNotas.Tables[0].Rows[j][0].ToString() == carnet)
             {
      comand = new MySqlCommand(comando4, conn);
      comand.Parameters.Add("?vNum_Carnet",carnet.ToString());

                    comand.Prepare();
                    comand.ExecuteNonQuery();
                }
            }
```

```
string comando3 = "INSERT INTO Notas
VALUES(?vNum_Carnet,?vCod_Asig,?vIP,?vIIP,?vProm,?vEF,?vNF,?vEE,?vCV,?vTT
,?vIA)";

  dataSet16_CodAsig.EnforceConstraints = false;

this.dataTable1TableAdapter.FillBy_CodAsig(this.dataSet16_CodAsig.DataTab
le1, Sem1, Sem2, Comience2 + "%");

for (int i = 0; i < dataSet16_CodAsig.Tables[0].Rows.Count; i++)
                {
COD_ASIG2 = dataSet16_CodAsig.Tables[0].Rows[i][0].ToString();

                    //TABLA Notas
                    comand = new MySqlCommand(comando3, conn);

comand.Parameters.Add("?vNum_Carnet",
F2_numero_carnet_maskedTextBox1.Text);
comand.Parameters.Add("?vCod_Asig", COD_ASIG2.ToString());
                    comand.Parameters.Add("?vIP", null);
                    comand.Parameters.Add("?vIIP", null);
                    comand.Parameters.Add("?vProm", null);
                    comand.Parameters.Add("?vEF", null);
                    comand.Parameters.Add("?vNF", null);
                    comand.Parameters.Add("?vEE", null);
                    comand.Parameters.Add("?vCV", null);
                    comand.Parameters.Add("?vTT", null);
                    comand.Parameters.Add("?vIA", null);

                    comand.Prepare();
                    comand.ExecuteNonQuery();
                }
         }
```

CÓDIGO:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using MySql.Data.MySqlClient;
using MySql.Data.Types;

namespace Form1_principal
{
    public partial class F5_Reg_NotasEst_Asig : Form
    {
        private MySqlConnection conn = new MySqlConnection("Server=
localhost;Database=Registro2;Uid=root;Pwd=root");
```

```
        private MySqlCommand cmd;
        private static DataGridViewCell celdaSeleccionada;
        bool resp = false;
        string null1 = "";


        public F5_Reg_NotasEst_Asig()
        {
            InitializeComponent();
        }
        private void F5_Reg_NotasEst_Asig_Load(object sender,EventArgs e)
        {
              // TODO: esta línea de código carga datos en la tabla
      'dataSet13_R_N_Est_Asig.DataTable1_Reg_Notas_Asig' Puede moverla o
      quitarla según sea necesario.

this.dataTable1TableAdapter_F5Reg_Notas_ASig.Fill_Reg_N_Est_Asig(this.dat
aSet13_R_N_Est_Asig.DataTable1_Reg_Notas_Asig,
Form5_Reg_N_Est_Asig.F5_criterioCodCarrera.ToString(),
Form5_Reg_N_Est_Asig.F5_criterioAsignatura.ToString(),
Form5_Reg_N_Est_Asig.F5_criterioSemestre.ToString());
        }

        private void
dataTable1_Reg_Notas_AsigDataGridView_CellValueChanged(object sender,
DataGridViewCellEventArgs e)
        {
            if (e.RowIndex >= 0 && e.ColumnIndex >= 0)
            {
                //EXTARIGO EL VALOR DEL CAMPO CARNET
                celdaSeleccionada =
dataTable1_Reg_Notas_AsigDataGridView.Rows[e.RowIndex].Cells[0];
                string carnetNo = celdaSeleccionada.Value.ToString();

                //EXTRAIGO EL VALOR DEL CAMPO NOM_ASIG
                celdaSeleccionada =
dataTable1_Reg_Notas_AsigDataGridView.Rows[e.RowIndex].Cells[12];
                string asignaturaCod =
celdaSeleccionada.Value.ToString();

                // EXTRAIGO EL VALOR DE LA CELDA ACTUAL.
                celdaSeleccionada =
dataTable1_Reg_Notas_AsigDataGridView.Rows[e.RowIndex].Cells[e.ColumnInde
x];
                string valorCelda = celdaSeleccionada.Value.ToString();

                    for(int i=0; i<=100;i++)
                    {
                     while(valorCelda.Equals(i.ToString()) ||
valorCelda.Equals(null1))
                     {
                        goto Continuar;
                         resp=true;
                     }
                    }
                    if (resp == false)
```

```
                    {
                        MessageBox.Show("El valor debe de estar entre o y
100", "Registro de Notas", MessageBoxButtons.OK,
MessageBoxIcon.Information);

celdaSeleccionada.ErrorText ="ESE VALOR NO SE ACTUALIZARA";
                        return;
                    }

            Continuar:
                    celdaSeleccionada.ErrorText = String.Empty;
            Regresar4:
                    if (celdaSeleccionada.ColumnIndex == 4)
                    {

                        cmd = new MySqlCommand(("UPDATE notas SET IP = "
+ valorCelda + " WHERE Num_Carnet LIKE '" + carnetNo + "' AND Cod_Asig
LIKE '" + asignaturaCod + "'"));

                        cmd.CommandType = CommandType.Text;
                        cmd.Connection = conn;
                        conn.Open();
                        try
                        {
                            cmd.ExecuteNonQuery();
                            conn.Close();
                                                    }
                        catch (MySqlException Mysqlex)
                        {
                            valorCelda = "NULL";
                            conn.Close();
                            goto Regresar4;
                        }
                    }

                Regresar5:
                    if (celdaSeleccionada.ColumnIndex == 5)
                    {
                        cmd = new MySqlCommand(("UPDATE notas SET IIP = "
+ valorCelda + " WHERE Num_Carnet LIKE '" + carnetNo + "' AND Cod_Asig
LIKE '" + asignaturaCod + "'"));

                        cmd.CommandType = CommandType.Text;
                        cmd.Connection = conn;
                        conn.Open();
                        try
                        {
                            cmd.ExecuteNonQuery();
                            conn.Close();
                        }
                        catch (MySqlException Mysqlex)
                        {
                            valorCelda = "NULL";
                            conn.Close();
                            goto Regresar5;
                        }
```

```
                    }
                Regresar6:
                    if (celdaSeleccionada.ColumnIndex == 6)
                    {
                        cmd = new MySqlCommand(("UPDATE notas SET Prom =
" + valorCelda + " WHERE Num_Carnet LIKE '" + carnetNo + "' AND Cod_Asig
LIKE '" + asignaturaCod + "'"));

                        cmd.CommandType = CommandType.Text;
                        cmd.Connection = conn;
                        conn.Open();
                        try
                        {
                            cmd.ExecuteNonQuery();
                            conn.Close();
                        }
                        catch (MySqlException Mysqlex)
                        {
                            valorCelda = "NULL";
                            conn.Close();
                            goto Regresar6;
                        }
                    }

                Regresar7:
                    if (celdaSeleccionada.ColumnIndex == 7)
                    {
                        cmd = new MySqlCommand(("UPDATE notas SET EF = "
+ valorCelda + " WHERE Num_Carnet LIKE '" + carnetNo + "' AND Cod_Asig
LIKE '" + asignaturaCod + "'"));

                        cmd.CommandType = CommandType.Text;
                        cmd.Connection = conn;
                        conn.Open();
                        try
                        {
                            cmd.ExecuteNonQuery();
                            conn.Close();
                        }
                        catch (MySqlException Mysqlex)
                        {
                            valorCelda = "NULL";
                            conn.Close();
                            goto Regresar7;
                        }
                    }
                Regresar8:
                    if (celdaSeleccionada.ColumnIndex == 8)
                    {
                        cmd = new MySqlCommand(("UPDATE notas SET NF = "
+ valorCelda + " WHERE Num_Carnet LIKE '" + carnetNo + "' AND Cod_Asig
LIKE '" + asignaturaCod + "'"));

                        cmd.CommandType = CommandType.Text;
                        cmd.Connection = conn;
                        conn.Open();
```

```
                        try
                        {
                            cmd.ExecuteNonQuery();
                            conn.Close();
                             }
                        catch (MySqlException Mysqlex)
                        {
                            valorCelda = "NULL";
                            conn.Close();
                            goto Regresar8;
                        }
                    }
                Regresar9:
                    if (celdaSeleccionada.ColumnIndex == 9)
                    {
                        cmd = new MySqlCommand(("UPDATE notas SET EE = "
+ valorCelda + " WHERE Num_Carnet LIKE '" + carnetNo + "' AND Cod_Asig
LIKE '" + asignaturaCod + "'"));

                        cmd.CommandType = CommandType.Text;
                        cmd.Connection = conn;
                        conn.Open();
                        try
                        {
                            cmd.ExecuteNonQuery();
                            conn.Close();
                         }
                        catch (MySqlException Mysqlex)
                        {
                            valorCelda = "NULL";
                            conn.Close();
                            goto Regresar9;
                        }
                    }

                Regresar10:
                    if (celdaSeleccionada.ColumnIndex == 10)
                    {
                        cmd = new MySqlCommand(("UPDATE notas SET CV = "
+ valorCelda + " WHERE Num_Carnet LIKE '" + carnetNo + "' AND Cod_Asig
LIKE '" + asignaturaCod + "'"));

                        cmd.CommandType = CommandType.Text;
                        cmd.Connection = conn;
                        conn.Open();
                        try
                        {
                            cmd.ExecuteNonQuery();
                            conn.Close();
                        }
                        catch (MySqlException Mysqlex)
                        {
                            valorCelda = "NULL";
                            conn.Close();
                            goto Regresar10;
                        }
```

```
                    }

                Regresar11:
                    if (celdaSeleccionada.ColumnIndex == 11)
                    {
                        cmd = new MySqlCommand(("UPDATE notas SET TT = "
+ valorCelda + " WHERE Num_Carnet LIKE '" + carnetNo + "' AND Cod_Asig
LIKE '" + asignaturaCod + "'"));

                        cmd.CommandType = CommandType.Text;
                        cmd.Connection = conn;
                        conn.Open();
                        try
                        {
                            cmd.ExecuteNonQuery();
                            conn.Close();
                        }
                        catch (MySqlException Mysqlex)
                        {
                            valorCelda = "NULL";
                            conn.Close();
                            goto Regresar11;
                        }
                    }
                }
        }
```

Formulario para Actualizar Estudiante:

Registrar Estudiante

Introduzca los Datos

1 de 64

Número de Matricula 62

Número de Carnet: 02-00040-0

Nombres: BLANCA MARIA

Apellidos: HERNANDEZ PALACIOS

Sexo: FEMENINO

Fecha de Nacimiento: 21/09/1984

Estado Civil: SOLTERO

Dirección: EL COYOLAR

Teléfono: 311-4521

Cédula: 281-210984-0001A

Procedencia: URBANA

Año Lectivo 2008

Año: 4

Carrera: ADMINISTRACION DE EMP

01

00

EINGRESO

Tipo Beca: NINGUNA

Estudiante Encontrado

Aceptar

Buscar

Actualizar

Cancelar

Borrar

Primero

Siguiente

Anterior

Último

Buscar por Número de Carnet:

02-00040-0

Reporte de Matricula de Estudiante:

Reporte de Registro de Estudiante

1 de 1 100% Buscar | Siguiente

FACULTAD DE CIENCIAS ECONOMICAS Y EMPRESARIALES.

REPORTE DE MATRICULA 2008.

Codigo Matricula	62	**Código Carrera**	05
Numero de Carnet:	02-00040-0	**Sexo:**	FEMENINO
Carrera:	TURISMO	**Estado Civil**	SOLTERO
Nombre	BLANCA MARIA	**Tipo de Beca:**	NINGUNA
Apellidos:	HERNANDEZ PALACIOS	**Monto Exonerado:**	300
Direccion:	EL COYOLAR	**Ano Lectivo:**	2008
No Cedula	281-210984-0001A	**Situacion Academica:**	REINGRESO
Procedencia:	URBANA	**Telefono:**	311-4521
Ano Estudio:	4	**Fecha Nacimiento:**	21/09/1984