UNIVERSIDAD NACIONAL AUTÓNOMA DE NICARAGUA

UNAN-LEÓN

DEPARTAMENTO DE COMPUTACIÓN

FACULTAD DE CIENCIAS Y TECNOLOGÍA

MONOGRAFÍA PARA OPTAR AL TITULO DE:

INGENIERO EN SISTEMAS DE INFORMACIÓN

AUTOMATIZACIÓN DE REGISTRO ACADÉMICO DEL COLEGIO TRIDENTINO SAN RAMÓN NONATO DE LA CIUDAD DE LEÓN.

INTEGRANTES:

- Br. MILTON PADILLA SILVA.
- Br. OSWALDO RIVAS ESCOBAR.

TUTOR: M.SC. DENIS ESPINOZA.

León, 07 de noviembre de 2012

AGRADECIMIENTO

Le agradecemos primeramente a ***Dios*** por darnos la oportunidad de haber culminado una fase más de nuestras vidas y darnos el don del conocimiento.

A nuestros ***padres*** por su ayuda incondicional, con su apoyo moral y económico con su esmerada dedicación, confianza que nos ha brindado a lo largo de nuestra vida y la oportunidad que nos dieron para culminar y coronar nuestra carrera.

A nuestros ***familiares*** y ***amigos*** por su apoyo y motivación en los momentos de dificultades.

A nuestro ***tutor Msc. Denis Espinoza***, por brindarnos parte de sus conocimientos, tiempo, paciencia y colaboración para realizar el presente trabajo.

La presente tesis se la dedico con mucho cariño y respeto a:

Dios: por haberme regalado la vida y la fuerza que me dio para seguir adelante a pesar de los problemas que se me presentaron en el camino de mi formación.

Con mucho amor a mi **madre Paula Azucena Escobar** por haberme dado la guía de encontrar la inspiración y apoyo incondicional en la culminación de mi carrera y a mi **Padre Fernando Rivas Gutiérrez** por su confianza.

A mi tutor Msc. Denis Leopoldo Espinoza Hernández por brindarme la ayuda, el tiempo y sobre todo la paciencia necesaria para lograr esta meta.

A mi compañero de tesis Milton Padilla por que sin el este trabajo no hubiese sido posible.

A todos ustedes se la dedico porque de alguna u otra forma llegaron a ser parte de esta carrera que hoy está llegando a su meta.

Y a todas las personas que aportaron un granito de arena a este trabajo.

"A la cima no se llega superando a los demás, sino superándose a sí mismo."

Oswaldo Alberto Rivas Escobar

Doy gracias a **Dios** primeramente por haberme permitido culminar este trabajo monográfico ya que si no fuera por su amor y misericordia yo no estaría hoy aquí escribiendo estas líneas de agradecimientos.

Esta tesis la he dedicado de manera muy especial, con mucho amor y cariño a mi madre ***Martha Zoraida Silva***, por la educación que me ha dado. Sin sus maternales consejos no me habría convertido en la persona que soy hoy y no habría podido completar este ciclo de mi vida. Siempre estaré agradecido con ella y mis hermanos por el amor y apoyo que siempre he recibido.

También la dedico a mi amigo y colega de trabajo ***Oswaldo Alberto Rivas***, por sus buenas recomendaciones y aportaciones para poder hacer realidad este sueño de nuestras vidas, ya que, a como dice la biblia *Más valen dos que uno, porque obtienen más fruto de su esfuerzo. Si caen, el uno levanta al otro... Uno solo puede ser vencido, pero dos pueden resistir. ¡La cuerda de tres hilos no se rompe fácilmente! Eclesiastés 4:9, 10, 12 (NVI), por eso gracias Oswaldo por haber trabajado conmigo.*

Al igual que mi amigo Oswaldo quiero agradecer y dedicar este trabajo a nuestro tutor **Msc*. Denis Leopoldo Espinoza Hernández***, gracias profesor por habernos brindado su ayuda, tiempo y sobre todo su guía para poder llegar a cumplir cada uno de los objetivos que no habíamos planteados, gracias porque siempre estuvo ahí dándonos consejos de cómo se podría llevar a cabo la solución de aquellos módulos que se miraban bastante complicado. Hoy por hoy puedo decir que mi amigo Oswaldo y Yo hemos tenido el mejor tutor.

Por ultimo quiero agradecer y a la vez también dedicar este trabajo a mi amigos, ***Marbelliett Reyes, Jeannevive Padilla, Katheline Reyes, Cristhiam Pacheco, Mauricio Paguaga*** y todos aquellos que siempre estuvieron presente durante mi carrera brindándome su ayuda y consejos en los momentos más difíciles de esta.

"Cuanto mayor es la dificultad, mayor es la gloria por haberla

Superado"

Milton Israel Padilla Silva

Contenido

Índice de Figuras

RESUMEN

El presente trabajo monográfico se trata sobre el desarrollo de un sistema para gestionar la información académica del Colegio Tridentino San Ramón Nonato con la finalidad de lograr un mejor desempeño y agilizar los procesos, garantizando un mejor manejo de la información.

Al finalizar la elaboración de este sistema puede afirmarse que se cumplieron todos los objetivos planteados, adicionalmente para el desarrollo de la aplicación se utilizaron diversas tecnologías como los controles DevExpress para .NET, el gestor de base de datos SQL Express 2005 y Visual Estudio 2008.

Dentro de esta aplicación se crearon varios módulos a los cuales se encargan de agilizar los procesos de gestión académica del Colegio Tridentino San Ramón Nonato entre los cuales tenemos: Profesores, Estudiantes y Notas.

Esperamos que la lectura de este documento le ayude a formar una idea general de las diferentes etapas que se llevaron a cabo para el desarrollo de este sistema.

1. INTRODUCCIÓN

1.1 ANTECEDENTES

Actualmente el Colegio Tridentino San Ramón Nonato realiza todos los procesos de manera manual usando libretas, y hojas de Excel o documentos de Word lo que provoca que el trabajo se realice de una manera muy lenta y que las operaciones tarden demasiado.

El proceso de matrícula se realiza sobre hojas fotocopiadas para la cual existen varios docentes encargados de atender a los alumnos que llegan a matricularse. El proceso consta de varias etapas que incluyen el pago de aranceles, la revisión de las notas del estudiante, el visto bueno del inspector o director para otorgar la matrícula, el llenado de la hoja de matrícula y la firma y sellado.

La creación de los grupos se realiza de forma manual una vez que han finalizado las matrículas. En este punto se llenan unas hojas de Excel que contienen las listas que oficialmente se utilizaran como listas de grupos.

La creación del horario es uno de los procesos más delicados dentro del proceso de gestión académica. Este proceso se realiza con la participación de varios docentes que se encargan de controlar y evitar los choques de profesores, grupos y aulas durante la creación del horario.

El control de nota es de los procesos más complejos de manipular pues para cada asignatura se crea una hoja de Excel dentro de las cuales solamente la secretaría es la encargada de agregar información.

La disciplina y las asistencias son dos elementos muy importantes de controlar. Para tal efecto en el colegio se manejan una lista de asistencia diaria en cada una de las secciones así como un boletín disciplinario por cada uno de los estudiantes en el cual se anotan diversos tipos de incidencias.

En un sector con tanta importancia social como es el sector educativo, en el que no abundan las grandes innovaciones tecnológicas, se hace imprescindible la creación de un

software de gestión académica que permita a los centros educativos que su proceso sea lo más rápido posible.

En la actualidad existen una gran variedad de software de gestión académica que se han elaborado para los colegios del municipio de León y que han venido a beneficiar a los colegios que se le han diseñados este tipo de software.

1.2 PLANTEAMIENTO DEL PROBLEMA

El Colegio Tridentino San Ramón tiene algunos problemas con sus procesos de gestión académica ya que este es muy rudimentario. Algunos de estos problemas se describen a continuación:

Con el proceso de matrícula este es un proceso muy largo ya que los estudiantes tiene que realizar largar filas esperas para ser matriculados y además peligran por no ser admitidos pues debe ser revisado su historial de disciplinas para ser aceptados a ser matrículas y tener el visto bueno por el inspector o el director del colegio. Todos estos pasos atrasan el proceso obligando al estudiante a tener que pasar por varias mesas para ver concluido este proceso.

La creación de grupos y horarios es otro de los problemas ya que este proceso es uno de los más largos y tediosos. Debe asegurarse que no existan choques entre profesores y clases y la asignación de profesores guías que es otro proceso en el que debe tenerse mucho cuidado para que el profesor no esté como guía en otros grupos.

El control de notas también es un problema que tiene el Colegio Tridentino San Ramón este proceso es fundamental siendo la mayor dificultad la obtención rápida de la información para generar certificados de notas y constancias. La disciplina y asistencia son fundamentales ya que son tomadas en cuenta a la hora de matricular y el problema y actualmente no se posee un mecanismo adecuado que permita registrar y procesar todas las incidencias de un estudiante relacionadas con esta área.

La generación de estadísticas es otro proceso muy rudimentario ya que las estadísticas son hechas a mano siendo un proceso muy tedioso pues existe demasiada información lo que no hace sencilla su manipulación.

De todo lo anteriormente descrito, se desglosan una serie de preguntas generales y específicas para nuestro trabajo

¿Podrá crearse un sistema que permita automatizar todos los procesos relacionados con la gestión académica del Colegio Tridentino San Ramón Nonato de la ciudad de León?

¿Se puede agilizar el proceso de matrícula de manera que toda la información académica de los estudiantes pueda ser accedida desde los puestos de los operadores?

¿Cómo optimizar el tiempo de creación del horario manteniendo todas las garantías que se han tenido en la forma de realización que se ha tenido hasta el momento?

¿De qué forma se puede mejorar el procesamiento de las notas de tal forma que se más sencilla la generación de informes y estadísticas?

1.3 JUSTIFICACIÓN

Debido a las dificultades con la que cuenta el Colegio Tridentino San Ramón con respecto al proceso de la gestión académica se decidió desarrollar un software que permita una gestión de todos los procesos académicos de una manera ágil y eficiente.

1.3.1 Originalidad

Es el primer sistema que el colegio empleará. Se han utilizado para su creación herramientas de las más actuales.

1.3.2 Alcance

En las funcionalidades que tiene nuestro sistema de gestión académica tendrá varias en las cuales tenemos profesores, estudiantes, aulas , grupos, horarios de clase, matriculas, pagos de mensualidad, asistencias, información estudiantil, estadísticas y reportes.

1.3.3 Producto

Dentro de las características principales del sistema tenemos que resaltar varias dentro de ellas tenemos autenticación de usuarios y sus roles, la generación de informes estadísticos y gráficos para ver con mayor detalles de la información referente a las matriculas, aprobado y reprobados.

1.3.4 Impacto

Se obtendrán los resultados esperados para una mejor utilización de la información referente al estudiante, profesores, aulas y grupos. Esto ayudara a mejorar que el proceso de la obtención sea más rápida y confiables que evitara algunos errores que son provocados por los usuarios que interactúan con la información detallada referente al usuario final.

1.4 OBJETIVOS

1.4.1 Objetivo General

- ✓ Desarrollar un sistema para la automatización de los procesos relacionados con la gestión académica del Colegio Tridentino San Ramón Nonato de la ciudad de León.

1.4.2 Objetivo Especifico

- ✓ Agilizar el proceso de matrícula del Colegio Tridentino San Ramón de manera que toda la información académica del estudiante pueda ser accedida desde los puestos de los operadores.

- ✓ Optimizar el tiempo de creación del horario manteniendo todas las garantías que se tienen actualmente.

- ✓ Mejorar el almacenamiento y procesamiento de las notas de tal forma que se permita una sencilla generación de informes y estadísticas.

2. Metodología

2.1 Ciclo de vida en espiral

Para el desarrollo del presente proyecto se realizaron una serie de fases, teniendo como modelo a seguir el ciclo de vida espiral del software. Las actividades de este modelo se conforman en una espiral, en la que cada bucle o iteración representa un conjunto de actividades. Las actividades no están fijadas a priori, sino que las siguientes se eligen en función del análisis de riesgo, comenzando por el bucle interior y se sigue con los que están más arriba.

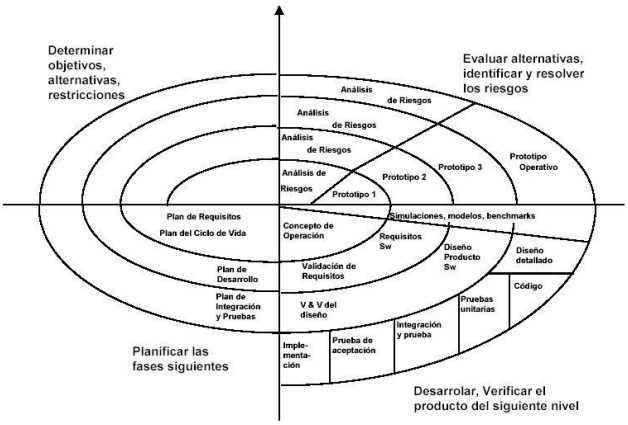


Figura 1 Ciclo de vida en Espiral

Para cada ciclo habrá cuatro actividades:

Determinar o fijar objetivos

Se realizaron visitas a la institución donde será empleado el sistema con el objetivo de conocer su funcionamiento, forma en que tramitan, gestionan, se desempeña el personal, conflictos internos con el fin de conocer a profundidad las necesidades de la institución.

Análisis del riesgo

Uno de los principales riesgos al que nos enfrentamos es la inconsistencia del sistema debido a ese problema presentamos las siguientes alternativas:

1. El uso de contraseña para cada usuario.
2. Uso de diversas herramientas para mayor seguridad en la aplicación.

Desarrollar, verificar y validar (probar)

Basándonos en la Información recolectada se procedió a determinar lo que el sistema requiere para su elaboración, obteniendo de esta etapa:

- Diagrama de casos de uso.
- Diagrama de clases.
- Diagrama de secuencia.
- Diseño de datos.
- Diseño de las interfaces.

Con la información antes recolectada, se procedió al desarrollo de esta parte del software. Luego se procedió a probar el sistema antes de implementarlo con el objetivo de comprobar que el resultado fue el esperado. La fase se culmina solo si las anteriores finalizaron con éxito entonces se procederá a la puesta en ejecución del sistema final.

Planificar

Revisar todo lo que sea hecho, evaluándolo y con ello decidir si se continúa con las fases siguientes y planificar la próxima actividad.

3. MARCO TEÓRICO

3.1 Colegio Tridentino San Ramón Nonato

Casi siglo y medio después de haber sido erigida la Diócesis de León nació el «Colegio Tridentino San Ramón», antes llamado «Seminario-Conciliar San Ramón»; y de éste, la grandiosa Universidad de León, hoy Universidad Nacional Autónoma de Nicaragua. A la par de la Real e Insigne Basílica Catedral de la Asunción de la Bienaventurada Virgen María, se encuentra el complejo educativo más antiguo de nuestra Patria. El origen de ello fue inspirado en el lejano 1621 cuando llegó a la diócesis de Nicaragua, Fray Benito de Baltodano, un español benedictino. Entre las obras destacadas de este obispo de León se cuentan: la fundación de los hospitales de Santa Catalina de Sutiaba y de San Juan de Dios de Granada; la construcción de las iglesias de Guadalupe de León y de Granada; y la fundación de la primera escuela para enseñar castellano, aritmética y doctrina cristiana, algunos ven en esta obra la semilla de nuestra institución educativa.

Desde sus orígenes hasta la actualidad el Colegio Tridentino San Ramón ha garantizado que la educación integral de sus alumnos les de la capacidad de desempeñarse en cualquier ámbito social en pro del desarrollo de nuestra región, grandes hombres ilustres han sido alumnos del tres veces centenario Colegio, y un hecho relevante es que hoy a 331 años de existencia la misión por la cual nace ésta institución sigue en pie: formar al hombre y la mujer de nuestra región, y porque no admitirlo, de nuestro país. La grandeza de ser alumno, docente, trabajador del San Ramón se ve en la labor de asumir el compromiso de transmitir lo que se ha heredado.

Actualmente se garantiza la formación en tres niveles de enseñanza: Pre-escolar; Primaria y Secundaria, la intención es educar integralmente. Hacer que Dios fuente de todo saber, nos ayude a alcanzar la Ciencia necesaria para ser portadores de la Virtud del conocimiento. (Ramón, 1990)

3.2 Tecnologías Empleadas

3.2.1 Visual estudio 2008

3.2.1.1 Qué es Visual Estudio 2008

Visual estudio es un entorno de desarrollo integrado (**IDE**), visual estudio permite crear aplicaciones en varios lenguajes de programación y estos tipos de aplicaciones diseñadas con este entorno se pueden intercomunicar con estaciones de trabajo, además que permite mejorar la interoperabilidad entre código nativo y código manejado por .NET. Esta integración más profunda simplifica el trabajo del diseño y la codificación.

Es un completo conjunto de herramientas para la creación tanto de aplicaciones de escritorio como de aplicaciones web empresariales para trabajo en equipo. Aparte de generar aplicaciones de escritorio de alto rendimiento, se pueden utilizar las eficaces herramientas de desarrollo basado en componentes y otras tecnologías de Visual Studio para simplificar el diseño, desarrollo e implementación en equipo de soluciones empresariales. (Microsoft.com, s.f.)

3.2.1.2 Características de Visual Estudio 2008

- Permite la creación de aplicaciones en una variedad de plataformas
- Provee de un nuevo lenguaje de consultas integrado para el manejo de los datos, es el Microsoft Language Integrated Query (LINQ), este nuevo tipo de lenguaje hace fácil la manipulación de los datos y hace más rápido el proceso de los datos ya que no se va a la base de datos a traer la información sino que desde donde la tenemos ya como es el caso de una lista donde estén los datos.
- Provee a desarrolladores la habilidad de poder escoger entre múltiples versiones del Framework con el mismo entorno de desarrollo, así podemos desarrollar en la versión que queramos ya sea en .NET Framework 2.0, 3.0 o 3.5, y con esto podremos realizar proyectos diferentes en el mismo entorno. (Microsoft T. , 2012)

3.2.2 .NET

3.2.2.1 ¿Qué es .NET?

NET es una parte integral de muchas aplicaciones que se ejecutan en Windows y proporciona funcionalidad común para aquellas aplicaciones que se ejecutan. Esta descarga es para la gente que necesita. NET para ejecutar una aplicación en su ordenador. Para los desarrolladores, el Marco. NET proporciona un modelo de programación completo y coherente para la creación de aplicaciones que tienen experiencias de usuario visualmente impresionantes y una comunicación segura y sin problemas. (http://www.microsoft.com/net, 2012)

Es un framework de Microsoft que hace un énfasis en la transparencia de redes, con independencia de plataforma de hardware y que permita un rápido desarrollo de aplicaciones. Basado en ella, la empresa intenta desarrollar una estrategia horizontal que integre todos sus productos, desde el sistema operativo hasta las herramientas de mercado.

.NET podría considerarse una respuesta de Microsoft al creciente mercado de los negocios en entornos Web, como competencia a la plataforma Java de Oracle Corporación y a los diversos framework de desarrollo web basados en PHP. Su propuesta es ofrecer una manera rápida y económica, a la vez que segura y robusta, de desarrollar aplicaciones –o como la misma plataforma las denomina, soluciones– permitiendo una integración más rápida y ágil entre empresas y un acceso más simple y universal a todo tipo de información desde cualquier tipo de dispositivo.

(http://es.wikipedia.org/wiki/Microsoft_.NET, 2012)

El Framework. NET es un componente integral de Windows que admite la creación y ejecución de la próxima generación de aplicaciones y servicios Web XML. . NET Framework está diseñado para cumplir los siguientes objetivos:

- Para proporcionar un consistente entorno orientado a objetos de programación si el código objeto se almacena y se ejecuta a nivel local, ejecutado a nivel local pero distribuida en Internet o ejecutar de forma remota.
- Proporcionar un entorno de ejecución de código que reduzca al mínimo la implementación de software y los conflictos de versiones.
- Proporcionar un entorno de ejecución de código que promueve la ejecución segura del código, incluyendo el código creado por un tercero desconocido o semi-confianza.
- Proporcionar un entorno de ejecución de código que elimine los problemas de rendimiento de los entornos de secuencias de comandos o interpretado.
- Para hacer que la experiencia del desarrollador consistente a través de muy diversos tipos de aplicaciones, como aplicaciones basadas en Windows y aplicaciones basadas en Web.
- Para desarrollar toda la comunicación en estándares de la industria para asegurar que el código basado en el Marco. NET puede integrarse con cualquier otro código.

3.2.2.2 Las características de Common Language Runtime

El Common Language Runtime administra la ejecución de la memoria, el hilo, la ejecución de código, verificación de la seguridad del código, compilación, y otros servicios del sistema. Estas características son intrínsecas al código administrado que se ejecuta en el tiempo de ejecución de lenguaje común.

Con respecto a la seguridad, los componentes administrados se otorgan diversos grados de confianza, dependiendo de un número de factores que incluyen su origen (tal como la Internet, red de la empresa, o equipo local). Esto significa que un componente administrado podría o no podría ser capaz de realizar las operaciones de acceso a archivos, operaciones de acceso de registro-, u otras funciones sensibles, incluso si se está utilizando en la aplicación activa mismo.

El tiempo de ejecución aplica la seguridad de acceso a código. Por ejemplo, los usuarios pueden confiar en que un archivo ejecutable incrustado en una página Web puede reproducir una animación en la pantalla o cantar una canción, pero no pueden acceder a sus datos personales, sistema de archivos, o la red. Las características de seguridad del tiempo de ejecución por lo tanto permitir que el software legítimo Internet desplegada a ser excepcionalmente rico en características.

3.2.2.3 .NET Framework

La biblioteca de Framework. NET es una colección de tipos reutilizables que se integran estrechamente con Common Language Runtime. La biblioteca de clases está orientada a objetos, proporcionando los tipos de los que su propio código administrado puede derivar funcionalidad. Esto no sólo hace que los tipos de Marco. NET fácil de usar, sino que también reduce el tiempo asociado con el aprendizaje de las nuevas características del Marco. NET. Además, los componentes de terceros se pueden integrar a la perfección con las clases del Framework. NET.

Por ejemplo, las clases de Marco. NET de recogida de poner en práctica un conjunto de interfaces que se pueden utilizar para desarrollar sus propias clases de colección. Sus clases de colección se combinan a la perfección con las clases del Framework. NET.

Como era de esperar de una biblioteca de clases orientada a objetos, los tipos de. NET Framework permiten realizar una serie de tareas de programación comunes, incluyendo tareas como la gestión de cadena, recolección de datos, conectividad de base de datos y acceso a archivos. Además de estas tareas comunes, la biblioteca de clases incluye tipos que admiten una gran variedad de escenarios de desarrollo especializados. Por ejemplo, puede utilizar el NET Framework para desarrollar los siguientes tipos de aplicaciones y servicios.

- Las aplicaciones de consola.
- Applications GUI de Windows (Windows Forms).
- Windows Presentation Foundation (WPF).
- Aplicaciones ASP.NET.
- Servicios Web. Los servicios de Windows.

(Microsoft,msdn, 2012)

3.2.3 Estructura en capas

El tener una arquitectura n-capas bien definida reducirá los hoyos de vulnerabilidad que evidentemente se dan cuando no desarrollamos con esta arquitectura. Dentro de esta arquitectura la capa intermedia o la capa lógica de negocios ocupa un lugar angular en la construcción de una infraestructura de software que nos permitirá el crecimiento y la extensibilidad de servicios para todas las aplicaciones existentes y futuras.

La definición de los linderos de cada capa nos permitirá definir correctamente la colaboración que proveerá cada una de ellas y descubriremos que la capa intermedia es inevitablemente la lógica de negocios, esto hará una infraestructura robusta y lista para la extensibilidad y el crecimiento como proveedora de servicios. Veámoslo más a detalle:

3.2.3.1 Capa de acceso a datos (DAL)

Para el acceso a datos, comúnmente en las aplicaciones se crea una clase que instancia una conexión a la base de datos y otras clases en las cuales se generan manualmente las sentencias SQL. Luego se realizan procedimientos para pasar los parámetros respectivos, abrir la conexión a la base de datos, ejecutar la sentencia, capturar los datos si es que aplica y luego volver a cerrar la conexión.

3.2.3.2 La capa lógica de negocios (BLL)

La capa lógica de negocios (BLL) es donde se comprueban las reglas de negocio y donde se manipulan los datos devueltos por la capa de acceso a datos antes de enviarlos a la capa de objeto de negocio (BO).

Los objetos de negocio (BO) y las clases encargadas de trabajar con dichos objetos (BLL) se encargan de la obtención de datos para después ser manipulados por esta capa. Cada clase BLL contiene los mismos métodos que su clase análoga en la capa inferior (clase DAL). Estos métodos realizarán funciones de validación y de filtrado de datos antes de llamar a sus métodos parejos en la capa de acceso a datos.

3.2.3.3 Capa de objeto de negocio (BO)

En esta capa se maneja los objetos de las clases creadas por cada una de las tablas que se van a ocupar que estas sean ocupas para obtener los datos temporalmente y además de esta se crea otra capa más dentro de esta

3.2.3.4 Capa Lista (List)

En esta se crean las listas dependiendo de cada una de las clase BO que se hayan creado estas lista son encargadas de realizar la obtención de todos los datos solicitadas a través de las capas Lógica de negocio (BLL) y la de acceso a datos (DAL).

Estas dos últimas capas son utilizadas para la obtención y manipulación de los datos ellas se encargar de tener los datos para luego ser utilizados.

3.2.3.5 Diseño de la Arquitectura en capas

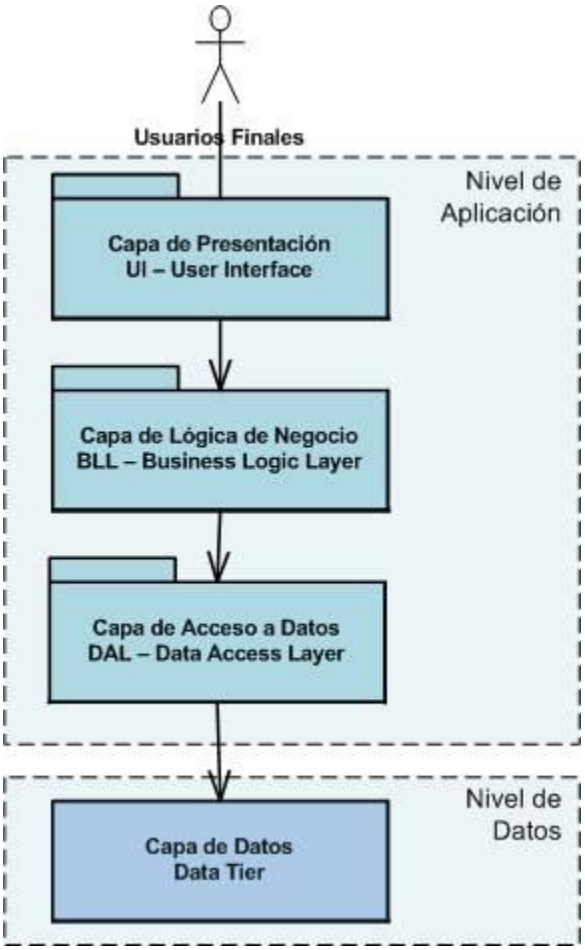


Figura 2 Arquitectura de Capas

Utilizar la arquitectura en capas no necesariamente se basa en utilizar estrictamente 3 capas sino que se utilizan más en el caso de uno de nuestros objetos de negocio utilizamos capas multinivel ya que dentro de la capa acceso a datos(DAL) se hacen llamados a otras capas desde ella para el acceso a los datos como es en el caso la creación de profesores donde se crea dos más una para experiencia laboral y cursos recibidos y estudios realizados de los profesores en este caso se llama dentro de la clase profesor a todas las clases mencionada anteriormente.

3.2.3.6 Justificación del uso de la estructura capas

Existe una gran cantidad de ventajas que se obtienen de usar la arquitectura en capas para los desarrollos, pero las que justifican el uso de esta tecnología para el proyecto actual son:

Compatibilidad con herramientas: este tipo de programación simplifica muchas operaciones por que hace que las herramientas de visual estudio se compatible con este tipo de programación.

Eficacia y flexibilidad: la estructura en capas es muy eficiente ya que reduce líneas de código y maximiza la interacción con los datos además de ser muy flexibles con los datos y su utilización.

Simplicidad: la programación en capas simplifica la utilización de código muy redundante y minimiza el código que se utiliza para el acceso y obtención de datos.

Seguridad: este modelo puedes integran módulos en los que se crea las autenticaciones de usuarios para obtener una mayor seguridad con el acceso a la información y manipulación de dicha información.

Menos líneas de código: se ahorran líneas de código ya que se programa solo las clases que se quieren manipular con el acceso a los datos y se hace una petición de datos uno a laves no varios.

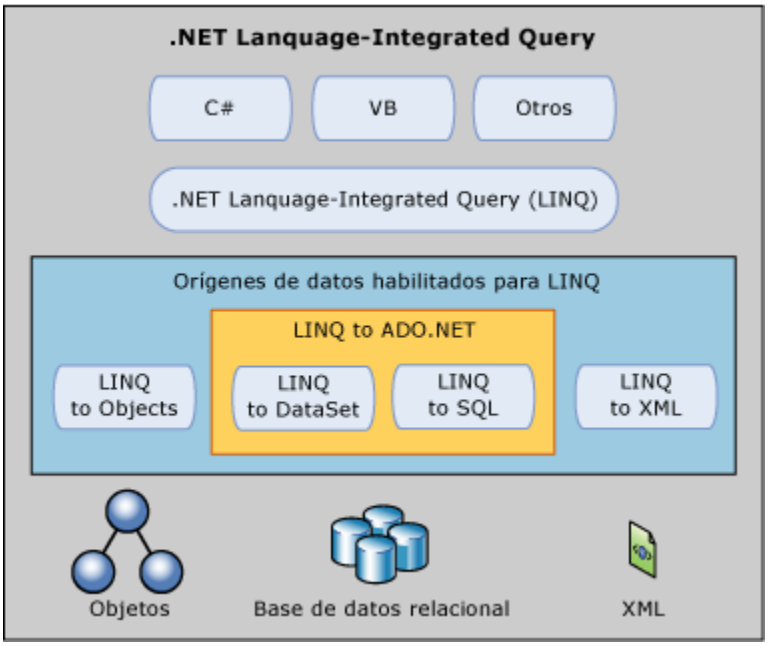
(wikipedia, 2012)

3.2.4 LINQ (Language-Integrated Query)

Language-Integrated Query (LINQ) define un conjunto de operadores de consulta estándar generales que se pueden usar en lenguajes de programación de .NET Framework 3.0. Estos operadores de consulta estándar permiten proyectar, filtrar y recorrer recopilaciones en memoria o tablas de una base de datos. Tenga en cuenta que las consultas de LINQ se expresan en el lenguaje de programación mismo y no como literales de cadena incrustados en el código de la aplicación. Se trata de un cambio significativo respecto a la forma en que la mayoría de aplicaciones se han escrito en versiones anteriores de .NET Framework. Escribir consultas desde el lenguaje de programación ofrece varias ventajas fundamentales. Simplifica las consultas eliminando la necesidad de usar un lenguaje de consulta independiente. Y si utiliza el IDE de Visual Studio 2008, LINQ también permite aprovechar la comprobación en tiempo de compilación, los tipos estáticos e IntelliSense.

LINQ se integra en varios aspectos del acceso a datos de .NET Framework, incluyendo el modelo de programación desconectada del DataSet y los esquemas de base de datos de SQL Server existentes. En esta sección se describe LINQ to ADO.NET, la implementación ADO.NET de LINQ.

El siguiente diagrama proporciona una visión general de cómo se relaciona LINQ to ADO.NET con lenguajes de programación de alto nivel, diferentes de las tecnologías LINQ y orígenes de datos compatibles con LINQ.



3.2.4.1 Ventajas de utilizar LINQ

- Sintaxis familiar para escribir consultas.
- Comprobación en tiempo de compilación de errores de sintaxis y seguridad de tipos.
- Compatibilidad mejorada con el depurador.
- Compatibilidad con IntelliSense.
- Capacidad para trabajar directamente con elementos XML en lugar de crear un documento XML contenedor, que es lo que se requiere en W3C DOM.
- Modificación de documentos XML en memoria de gran eficacia, aún más fácil de usar que XPath o XQuery.
- Funciones de filtrado, ordenación y agrupación eficaces.
- Modelo coherente para trabajar con datos en varios tipos de formatos y orígenes de datos. (Microsoft)

3.2.5 DevExpress v10.2

DevExpress controles de Windows Forms comparten muchas características comunes. Ellos utilizan los mismos métodos para controlar la apariencia configuraciones, datos, enfoques vinculantes mismos valores de formato y localización de recursos, etc.

Figura 3 DevExpress

3.2.5.1 XtraChart

XtraCharts suites un conjunto de controles de gráficos profesionales para. NET (para las formas tanto en plataformas Windows como ASP.NET, así como para XtraReports).

Meticulosamente diseñado con una impresionante variedad de tipos de gráficos diferentes, un conjunto de indicadores específicos listo visuales, monitoreo en tiempo real, análisis comparativo / contrastivo, mientras que también es capaz de gráficos financieros, XtraCharts es la solución a todas sus necesidades de creación de gráficos.

Independientemente de su tipo de origen de datos (ya sea una base de datos, colección o red de pivote) o la cantidad y el tipo de datos de entrada, XtraCharts es la mejor herramienta para la visualización de todos los datos con capacidades de personalización casi infinitas. Con funciones inteligentes de automatización de alto nivel, XtraCharts hace de gráficos 2D y 3D más fácil que nunca, mientras que permanece flexible y eficiente mediante la manifestación de una interfaz amigable e intuitiva.

3.2.5.1.1 Características generales

Soporte para múltiples plataformas de destino : XtraCharts Suite incluye el ChartControl (para Windows Forms) y WebChartControl (para ASP.NET). Ambos ofrecen una funcionalidad similar, mientras que ofrece todas las ventajas de sus plataformas específicas.

Profundo apoyo para los productos DevExpress otros: PivotGridControl (ASPxPivotGrid) y XtraReports . Para obtener más información sobre esto, vea Gráficos Pivot (Integración con un control de cuadrícula Pivot) y XRChart.

Soporte para múltiples tipos de gráficos diferentes: **37** 2D y **22** 3D tipos de gráficos desde básico (por ejemplo, barras y columnas , Punto y línea , Pie y redireccionamiento, o Radar y Polar) a lo específico (por ejemplo, Burbuja , Gantt o Parcela Swift que es especialmente diseñado para su procesamiento inmediato de cantidades muy grandes de puntos de datos).

3.2.5.1.2 Enlace de Datos

- XtraCharts soporta una variedad de tipos de orígenes de datos comunes, que también incluye soporte completo + ADO.
- Modo independiente - Una serie puede ser rellenada con datos manualmente por especifican valores para los (X) e (Y) las coordenadas de los puntos. También es posible combinar unido y sin unir puntos de vista de la serie en un solo diagrama.
- Series Encuadernación Individual - XtraCharts puede mostrar datos de múltiples fuentes de datos, ya que cada serie se puede enlazar a su propia tabla de datos.
- Uso de las plantillas de la Serie - En este modo el ChartControl crea Visto serie automáticamente, y no hay necesidad de rellenar la colección Serie manualmente.
- Serie de datos de filtrado - Esta característica le permite aplicar criterios de filtro para los datos mostrados por series específicas, lo que puede evitar la inundación del control de gráfico con datos irrelevantes.
- Datos de la Serie Clasificación - Puede ordenar los datos dentro de una serie, ya sea por sus argumentos o valores.
- Serie de datos de resumen - Puede aplicar diferentes funciones de resumen para los datos de una serie.
- Impresión y Exportación - Si usted es dueño de la Biblioteca XtraPrinting , usted puede exportar el gráfico a: un archivo de imagen (múltiples formatos de archivo son compatibles), un archivo PDF; una imagen incrustada en una página HTML, una imagen incrustada en una hoja de cálculo de Microsoft Excel, una página impresa.

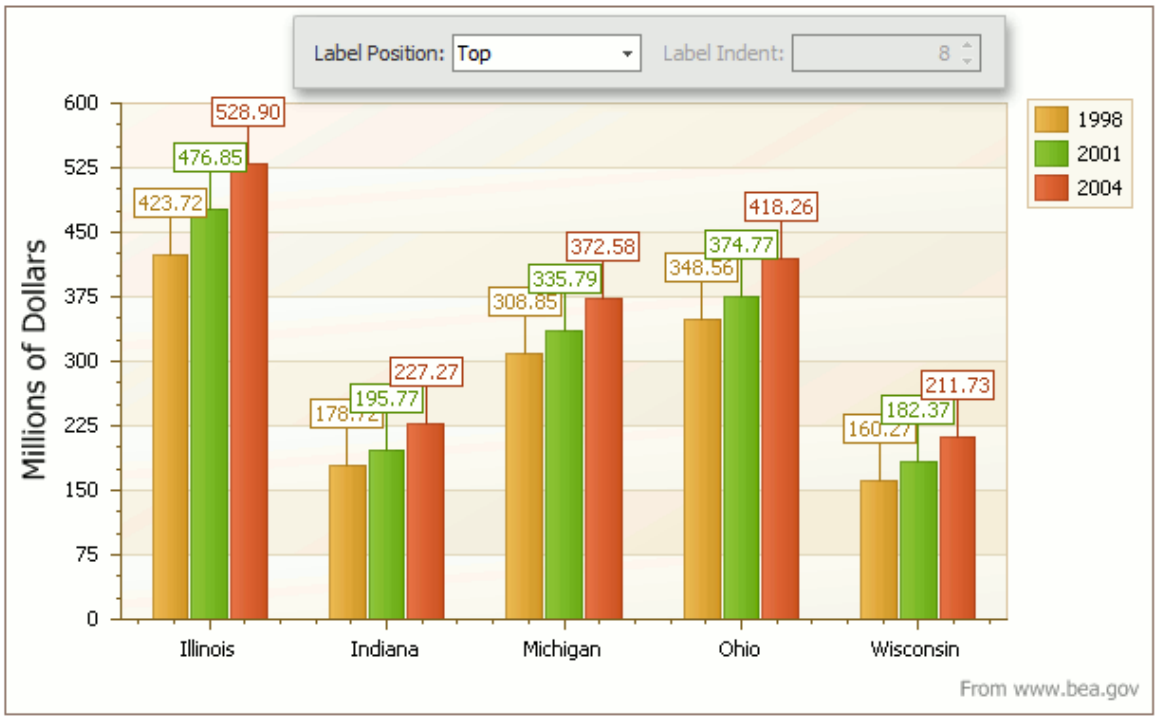


Figura 4 Diseño del XtraChart

3.2.5.2 XtraGrid

Completo ADO + Soporte - XtraGrid aprovecha al máximo ADO, Usando esta arquitectura de acceso a datos y la separación de los módulos de datos internos de presentación de datos, XtraGrid no utiliza ningún tipo de buffer adicionales en cualquiera de sus modos. Incluso cuando se agrupan las columnas, XtraGrid usará memoria mínima y realizar la operación tan rápido como el origen de datos puede proporcionar los datos.

Soporte para listas de datos - XtraGrid funciona con cualquier fuente que admita el IList, ITypedList o interfaz IBindingList, además de todas las interfaces heredadas.

Columnas sin enlazar - Es posible combinar la envolvente y los modos de datos no consolidados. Sólo tiene que añadir columnas sin enlazar y suministra datos a través de un evento. Los usuarios finales pueden editar los valores en las columnas sin enlazar y los cambios se pueden guardar en un origen de datos personalizado.

Verdadero Maestro-Detalle Support - XtraGrid le permite establecer y presentar maestro-detalle la información ya sea en el modo dependiente o independiente. Se puede incluso utilizar el taladro para hacer zoom en el modo de cada nivel de detalle, permitiendo que incluso los conjuntos más complicados de las relaciones para ser manipulado fácilmente.

Del lado del servidor de Gestión de Datos - Controles GridControlGridLookUpEdit y apoyar un nuevo modo de enlace de datos, llamado Modo de servidor, que ha sido específicamente diseñado para soportar grandes volúmenes de datos (que consisten en 50.000 registros y más aún). El modo de servidor de datos en función de las cargas pequeñas porciones y sólo cuando sea necesario. Todas las operaciones de reconocimiento de datos se realizan en el servidor de base de datos que se ha optimizado para estas consultas.

Las características clave del modo de servidor de enlace de datos garantizar el acceso rápido a los datos, independientemente de los datos de operación, tanto si la red utiliza ordenar, agrupar, filtrar o funciones de resumen.

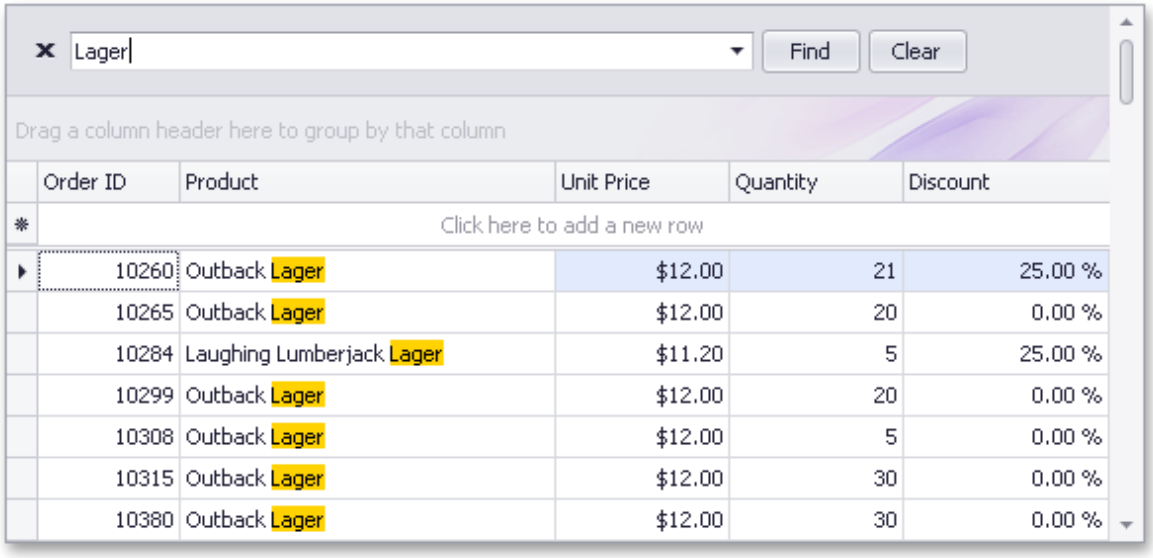


Figura 5 Diseño del XtraGrid

3.2.5.3 XtraReport

Se ha mejorado el Diseñador de informes para apoyar la interfaz de múltiples documentos (MDI). Ahora se puede cargar tantos informes como desee en la misma instancia del diseñador de informes (lo que le permite cambiar entre ellos y copiar / pegar datos y los controles a través de múltiples diseños de informe).

El Diseñador de informes para el usuario final utiliza una interfaz de barra de herramientas del menú, puede elegir entre "clásico" o MDI con pestañas. Cuando se utiliza una cinta, la única opción MDI con pestañas disponibles.

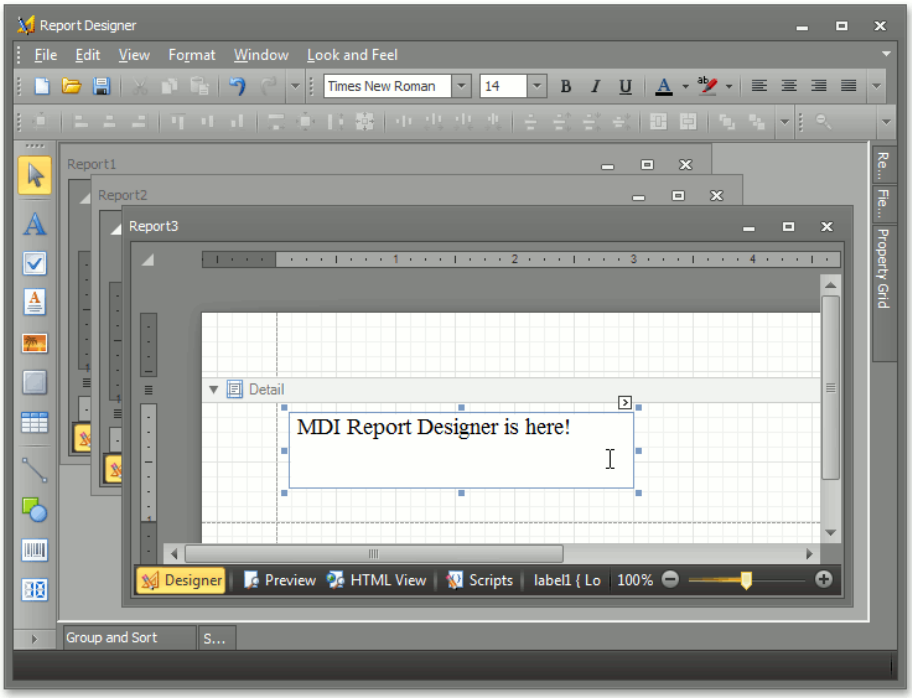


Figura 6 Diseño del XtraReport

En esta versión se ha hecho más fácil de guardar y cargar informes a cualquier fuente de datos en el diseñador de informes del usuario final. Dependiendo de la arquitectura de la aplicación, se puede definir un informe personalizado de almacenamiento para guardar los informes como archivos en un disco, un archivo zip, o directamente a una base de datos.

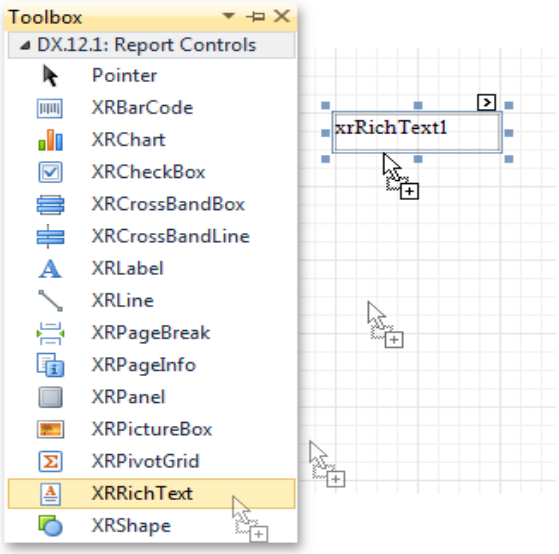


Figura 7 Cuadro de Herramienta del XtraChart

3.2.5.4 LookUp Editor

Este nuevo editor combina la funcionalidad de búsqueda con una nueva función de búsqueda. Al igual que el control de GridLookUpEdit, el Editor de Mapas Buscar incrusta un control de cuadrícula en el menú desplegable, y ofrece numerosas opciones de personalización de la cuadrícula. El built-in cuadro Buscar le permite localizar rápidamente los datos.

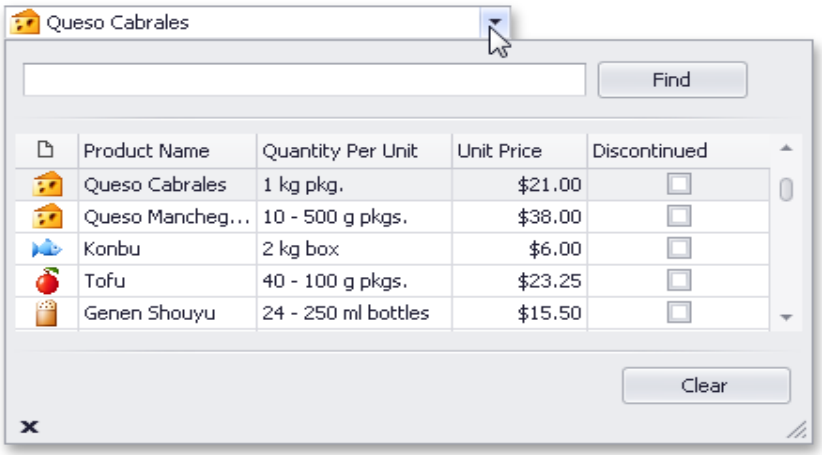


Figura 8 Diseño del LookUp Editor

Para localizar los datos de la consulta, basta con introducir caracteres en el cuadro Buscar y el control filtrar registros en consecuencia (aquellos que cumplan los criterios de búsqueda):

El control proporciona un conjunto de opciones y eventos que le permiten:

- Seleccione el modo de búsqueda automática o manual
- Utilice los Empieza con Contiene u operador de comparación
- Especificar columnas de búsqueda
- Implementar soporte para agregar nuevos registros a través de la ventana desplegable, etc.

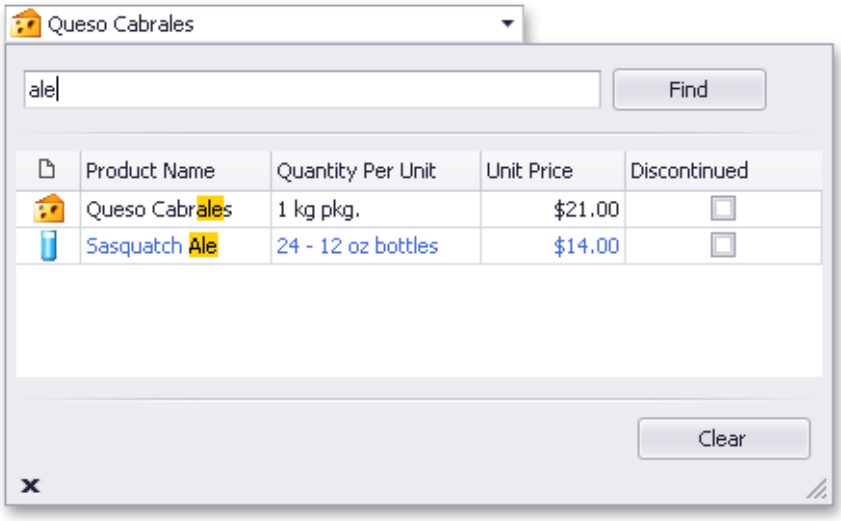


Figura 9 Búsqueda en el LookUpEdit

3.2.5.5 XtraTreeList

3.2.5.5.1 SummaryFooter

El **resumen de pie de página** se ha diseñado para mostrar resúmenes de totales, es decir, resúmenes calculados para el usuario root o todos los nodos en una lista de árbol. El pie de página contiene células pie de página que se muestran los valores del resumen para las columnas particulares. Al hacer clic en una celda de pie invoca el menú contextual de pie de página.

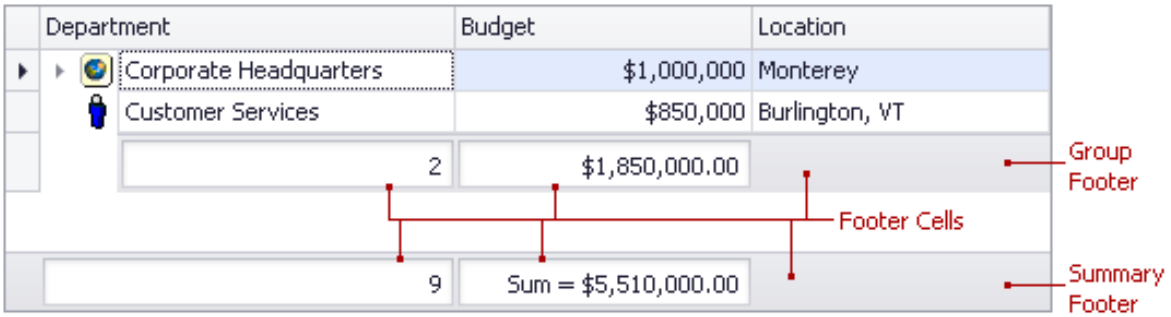


Figura 10 Diseño del SummaryFooter del XtraTreeList

(http://documentation.devexpress.com/, 2012)

3.2.6 SQL Server Express 2005

Es una plataforma global de base de datos que ofrece administración de datos empresariales con herramientas integradas de inteligencia empresarial (BI). El motor de la base de datos SQL Server 2005 ofrece almacenamiento más seguro y confiable tanto para datos relacionales como estructurados, lo que le permite crear y administrar aplicaciones de datos altamente disponibles y con mayor rendimiento.

El motor de datos SQL Server 2005 constituye el núcleo de esta solución de administración de datos empresariales. Asimismo, SQL Server 2005 combina lo mejor en análisis, información, integración y notificación. Esto permite crear soluciones de BI rentables que ayuden a su equipo a incorporar datos.

La integración directa con Microsoft Visual Studio, el Microsoft Office System y un conjunto de nuevas herramientas de desarrollo, incluido el Business Intelligence Development Studio, distingue al SQL Server 2005.

3.2.6.1 Características de SQL Server Express 2005

- ✓ **Mirroring de Bases de Datos:** amplía las posibilidades de duplicación de logs ("log shipping") proporcionando a los administradores de BBDD la opción de mirroring. Los administradores pueden usar esta funcionalidad para garantizar la disponibilidad de sus sistemas SQL mediante la configuración de un servidor en espera para su activación automática en caso de fallo (failover).
- ✓ **Backups duplicados ("mirroredbackup"):** SQL Server 2005 incluye un nuevo soporte para volúmenes de backup espejados, aumentando la disponibilidad de las copias de seguridad de SQL Server. La posibilidad de replicar el backup permite resolver posibles problemas de corrupción del medio físico de copia.

- ✓ **Mejoras en Transact-SQL:** introduce muchas posibilidades nuevas para el desarrollo de aplicaciones de bases de datos escalables. Estas mejoras incluyen el manejo de errores, nuevas posibilidades de consultas recursivas y soporte para nuevas funcionalidades del motor de SQL Server.

- ✓ **Conexión de Administrador dedicada:** introduce la conexión de administración dedicada, que pueden utilizar los administradores de BBDD para acceder a un servidor en explotación aun cuando el servidor está bloqueado o no disponible por cualquier motivo. Así, los administradores podrán ejecutar funciones de diagnóstico, o sentencias Transact SQL, a fin de poder resolver problemas en el servidor. (Server, 2012)

3.2.7 Procedimientos almacenados (storeprocedure)

Un procedimiento almacenado (store procedure) no es más que una secuencia ordenada de instrucciones T-SQL, que pueden recibir y proporcionar parámetros provistos por el usuario y se pueden guardar en el servidor con un nombre, para posteriormente se invocados y ejecutados, por lo regular desde una aplicación (Escritorio o Web). Desde la versión 2005, se incorpora la posibilidad de utilizar procedimientos almacenados usando el CLR de .NET. Es decir tenemos dos tipos de procedimientos almacenados.

Un procedimiento almacenado CLR es una referencia a un método de un ensamble (dll) de .NET Framework que puede aceptar y devolver parámetros suministrados por el usuario. (http://mspnor.wordpress.com/2008/10/31/sqlprocedimientos-almacenados-paso-a-paso/, 2012)

Cuando crea una aplicación con Microsoft SQL Server 2005, el lenguaje de programación Transact-SQL es la principal interfaz de programación entre las aplicaciones y la base de datos de Microsoft SQL Server. Cuando utiliza programas Transact-SQL, dispone de dos métodos para almacenar y ejecutar los programas.

- Puede almacenar los programas localmente y crear aplicaciones que envían los comandos a SQL Server y procesan los resultados.
- Puede almacenar los programas como procedimientos almacenados en SQL Server y crear aplicaciones que ejecutan los procedimientos almacenados y procesan los resultados. (microsoft, 2012)

Los procedimientos almacenados de Microsoft SQL Server son similares a los procedimientos de otros lenguajes de programación en el sentido de que pueden:

- Aceptar parámetros de entrada y devolver varios valores en forma de parámetros de salida al lote o al procedimiento que realiza la llamada.
- Contener instrucciones de programación que realicen operaciones en la base de datos, incluidas las llamadas a otros procedimientos.
- Devolver un valor de estado a un lote o a un procedimiento que realiza una llamada para indicar si la operación se ha realizado correctamente o se han producido errores (y el motivo de éstos).

Puede utilizar la instrucción EXECUTE de Transact-SQL para ejecutar un procedimiento almacenado. Los procedimientos almacenados difieren de las funciones en que no devuelven valores en lugar de sus nombres ni pueden utilizarse directamente en una expresión.

Utilizar procedimientos almacenados en SQL Server en vez de programas Transact-SQL almacenados localmente en equipos cliente presenta las siguientes ventajas:

- Se registran en el servidor.
- Pueden incluir atributos de seguridad (como permisos) y cadenas de propiedad, además se les pueden asociar certificados. Los usuarios pueden disponer de permiso para ejecutar un procedimiento almacenado sin necesidad de contar con permisos directos en los objetos a los que se hace referencia en el procedimiento.
- Mejoran la seguridad de la aplicación.
  Los procedimientos almacenados con parámetros pueden ayudar a proteger la aplicación ante ataques por inyección de código SQL.
- Permiten una programación modular.
  Puede crear el procedimiento una vez y llamarlo desde el programa tantas veces como desee. Así, puede mejorar el mantenimiento de la aplicación y permitir que las aplicaciones tengan acceso a la base de datos de manera uniforme.
- Constituyen código con nombre que permite el enlace diferido. Esto proporciona un nivel de direccionamiento indirecto que facilita la evolución del código.
- Pueden reducir el tráfico de red. Una operación que necesite centenares de líneas de código Transact-SQL puede realizarse mediante una sola instrucción que ejecute el código en un procedimiento, en vez de enviar cientos de líneas de código por la red. (microsoft, 2012)

3.2.8 Procedimientos (Agregar grupo matricula)

```
setANSI_NULLSON
setQUOTED_IDENTIFIERON
GO
ALTERPROCEDURE [dbo].[AgregarGrupoMatricula]
@idmatricula int,
@grupo int,
@nombre varchar(150),
@anyo varchar(4)
as
begin
begintry
begintran
update Tbl_Matricula
set grupo = @grupo
WHERE idmatricula=@idmatricula
update Tbl_Notas
set idmatricula=@idmatricula
where carnet =(select carnet from Tbl_Matricula
where anyolectivo=@anyo and carnet =(select carnet from Tbl_Estudiante
where(nombrecompleto +' '+ nombre2 +' '+ apellidocompleto +' '+
apellido2)= @nombre))
if(@@error<>0)
rollbacktran
endtry
begincatch
rollbacktran
endcatch
if(@@trancount>0)
committran
end
```

Cargar materia grado

```
setANSI_NULLSON
setQUOTED_IDENTIFIERON
GO
ALTERPROCEDURE [dbo].[CargarMateriaGrado1]
@grado varchar(10),
@anyo varchar(10),
@semestre1 varchar(10)=NULL,
@semestre2 varchar(10)=null
as
begin
selectdistinct Codigo,nombre, Simpartido
from PensumClases
INNERJOIN Pensum
ON PensumClases.Pensum = Pensum.pensumid
where Simpartido IN(@semestre1,@semestre2,'AMBOS')
AND @anyo Between Inicio and Fin
AND Grado=@grado
ORDERBY 3,1,2 asc

end
```

4. ANÁLISIS DE REQUISITO

Debido a la necesidad de realizar un mecanismo más eficaz que administre la información que manipula el Colegio Tridentino San Ramón Nonato se pretende realizar un sistema que describa el conjunto de funcionalidades, restricciones, especificaciones de requisitos software que deben cumplirse a la hora de elaborar la aplicación al Colegio San Tridentino Ramón que consistirá en la gestión de datos de estudiantes y profesores del colegio además de la salida de informes de los registros que el colegio necesite.

4.1 Funcionalidades del sistema

EL nombre con el cual se hará referencia a la aplicación será: “SGASR”, el producto realizará las siguientes funciones:

- Crear usuarios
- Modificarlo y borrar un usuario
- Mostrar datos de los usuarios
- Registrar información de los estudiantes
- Modificar información del estudiante
- Mostrar datos de los estudiantes
- Registrar información de los profesores
- Modificar información de los profesores
- Mostrar información de los profesores
- Registrar información de las aulas
- Modificar información de las aulas
- Eliminar información de las aulas
- Mostrar información de las aulas
- Registrar pensum
- Modificar un pensum
- Eliminar un pensum
- Mostrar pensum
- Registrar clases
- Modificar clases
- Eliminar clases
- Mostrar clases
- Crear horario de clase
- Modificar horario de clase
- Eliminar horario de clase

- Mostrar horario de clase por:
  - ✓ Grado
  - ✓ Profesor

- Registrar grupos
- Modificar grupos
- Eliminar grupos
- Mostrar grupos
- Asignar estudiantes a cada grupo
- Quitar estudiantes del grupo
- Registrar notas
- Modificar notas
- Mostrar notas por
  - ✓ Asignaturas
  - ✓ Semestre
  - ✓ Año
  - ✓ Grado
- Información estudiantil
- Información gráfica de estadísticas de matrículas por año
- Registrar matricula
- Eliminar matricula
- Mostrar matriculas
- Generar asistencias
- Registrar asistencias
- Registrar pago de mensualidad
- Mostrar mensualidades

4.2 Definiciones, acrónimos y abreviaturas usadas

- **SGASN:** Sistema de Gestión Académica San Ramón.

4.3 Modelo de caso de uso

El modelo de casos de uso es un mecanismo ampliamente empleado para la captura de requisitos de usuario ya que permite representar gráficamente, lo que el usuario espera de la aplicación.

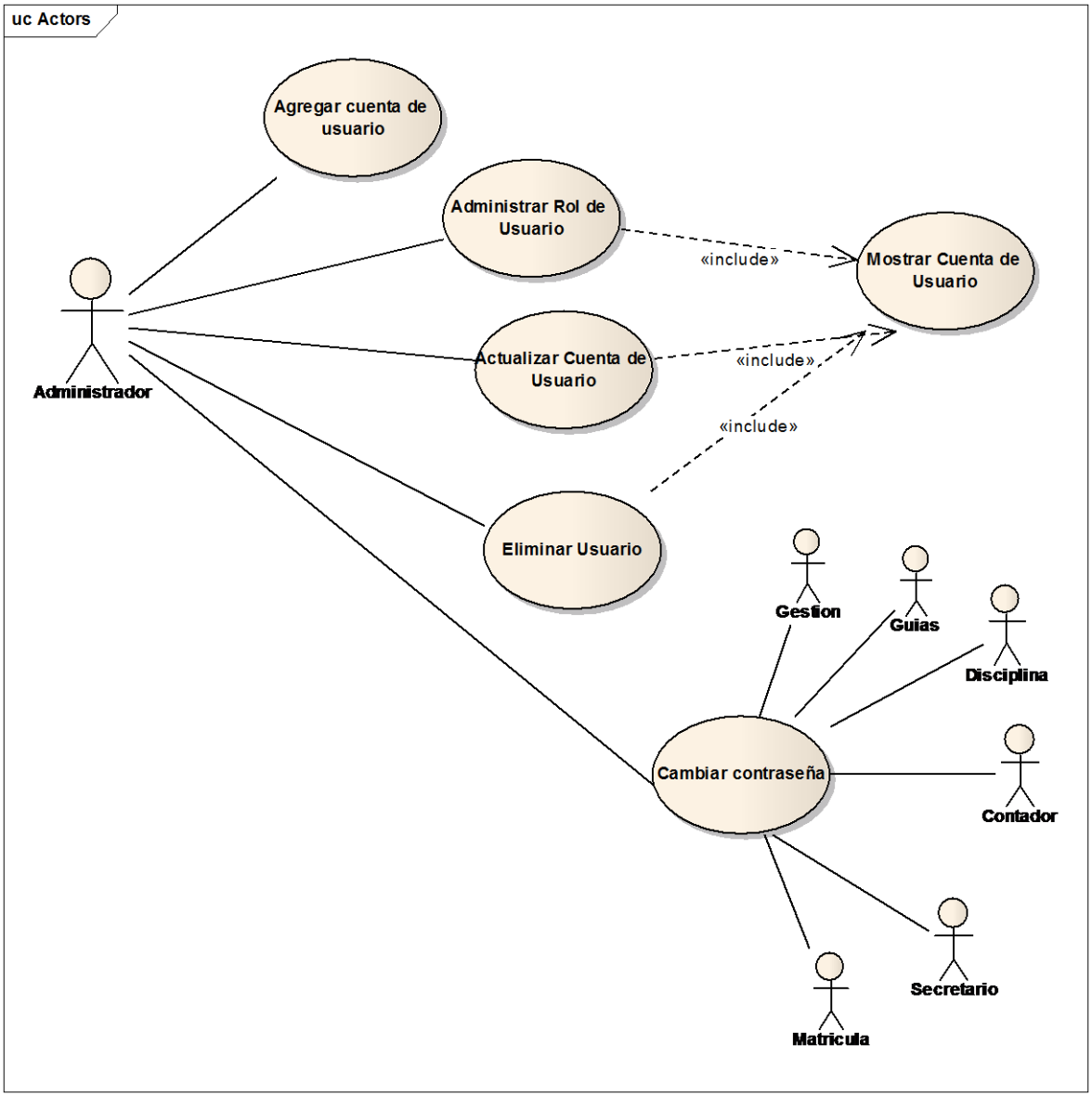


Figura 11 Diagrama de caso de uso: Administración de cuentas de usuario

Descripción de los Actores.

Nombre del actor:	Administrador del sistema.
Definición	Es el encargado de administrar el sistema y cuentas de usuario.
Notas:	Es capaz de acceder a todas las funcionalidades que realiza el sistema.

Nombre del actor:	Gestión.
Definición	Este podrá realizar cambios en su contraseña. Es el encargado de realizar todas las funciones básicas de la Secretaria académica.
Notas:	Es capaz de acceder a casi todos los módulos de la aplicación del sistema.

Nombre del actor:	Disciplina
Definición	Este podrá realizar cambios en su contraseña. El solo puede ver en algunas funciones básicas y encargado de las asistencias e información estudiantil.
Notas:	Es capaz de acceder a dos módulos para realizar cambios los cuales son Información Estudiantil y asistencias y los demás módulos solo podrá ver.

Nombre del actor:	Guías.
Definición	Este podrá realizar cambios en su contraseña. Es el encargado de ver la información estudiantil y ver información del estudiante.
Notas:	Es capaz de acceder a Información Estudiantil y ver en algunos módulos aplicación del sistema.

Nombre del actor:	Contador.
Definición	Este podrá realizar cambios en su contraseña. Es el encargado de realizar el pago de la mensualidad.
Notas:	Es capaz de acceder a un módulo que es Pago de mes de la aplicación del sistema.

Nombre del actor:	Secretario.
Definición	Este podrá realizar cambios en su contraseña. Es el encargado de registrar las notas de los estudiantes.
Notas:	Es capaz de acceder a dos de los módulos de la aplicación del sistema.

Nombre del actor:	Matricula.
Definición	Este podrá realizar cambios en su contraseña. Es el encargado de realizar la matriculas del sistema.
Notas:	Es capaz de acceder a un módulo y ver en dos mas de la aplicación del sistema.

4.4 Descripción de casos de uso

En el diagrama anterior se presentan las diferentes operaciones que se realizan sobre la gestión de la información de los usuarios del sistema las cuales son: agregar usuario, administrar un usuario, mostrar usuario, actualizar usuario, eliminar usuario y cambiar contraseña.

Caso de uso: Agregar Cuenta de Usuario.

Definición: Permite agregar usuarios a la aplicación.

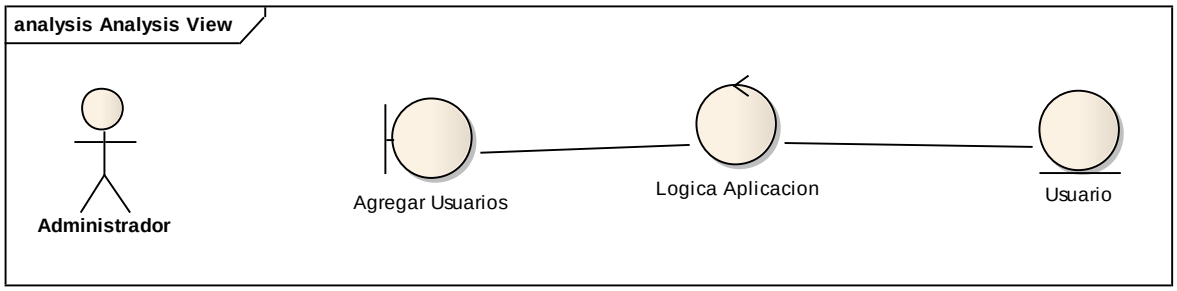


Figura 12 Modelo para el caso de uso agregar Usuario

Descripción del modelo de análisis:

1. Presentación de la interfaz “Agregar Usuario” desde la cual el Administrador puede realizar las opciones antes mencionadas.
2. Lógica de aplicación
   a. Esta se encargar de la verificación de los datos antes de ser almacenado en la base de datos.
3. Actualiza la base de datos después de realizar dicha operación.

Para los casos de usos que mostraremos a continuación solo nos enfocares en hablar del administrador del sistema.

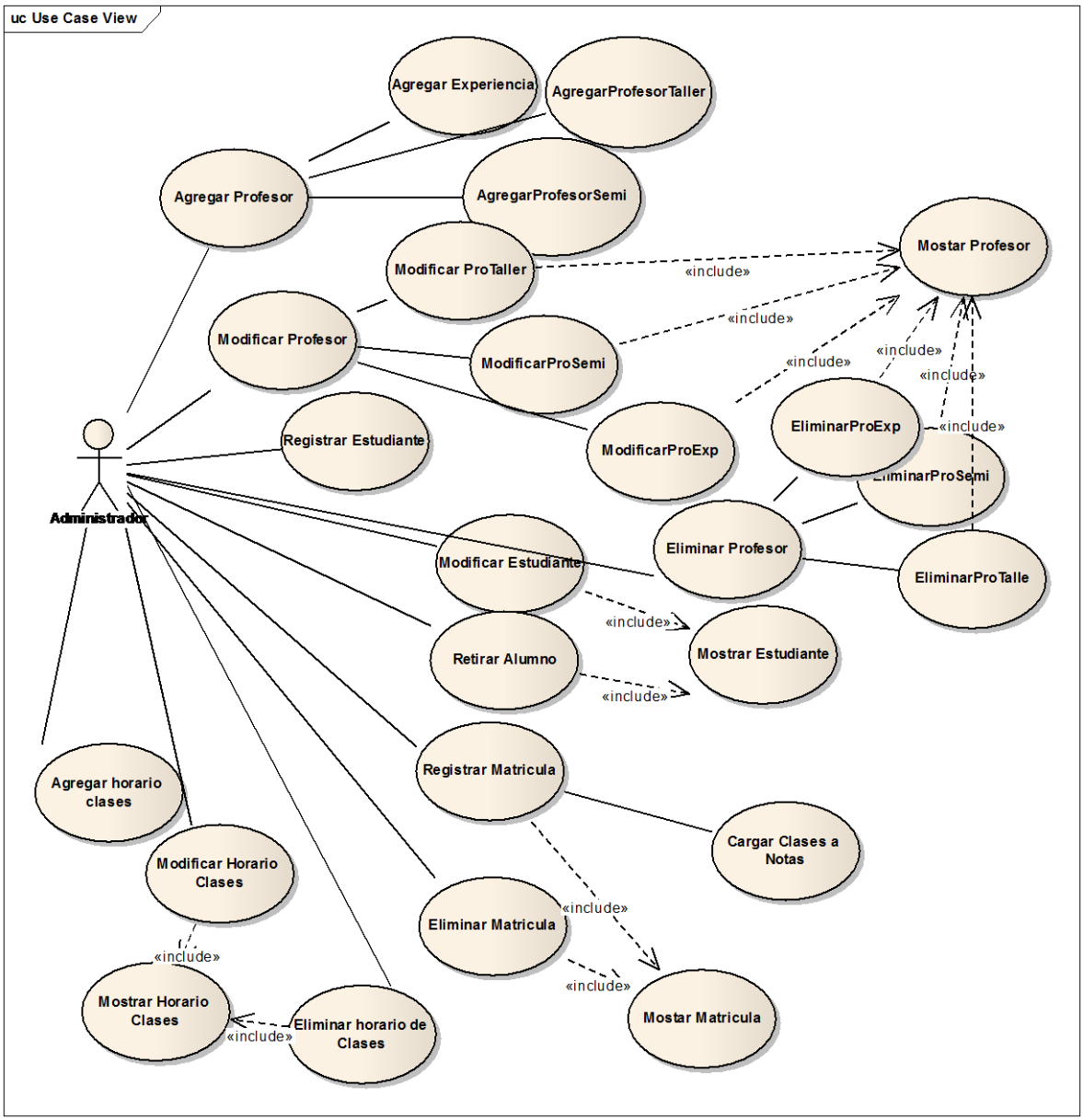


Figura 13 Diagrama de caso de uso Administrador (parte 2)

En la figura 5 se presentan los siguientes casos Registrar, Modificar, Eliminar y Mostar: Profesores, Estudiantes, Horarios de Clase, Matricula.

Caso de uso: Registrar Profesores.

Definición: Permite realizar el registro de los profesores en el sistema, las operaciones realizadas es Editar Profesor.

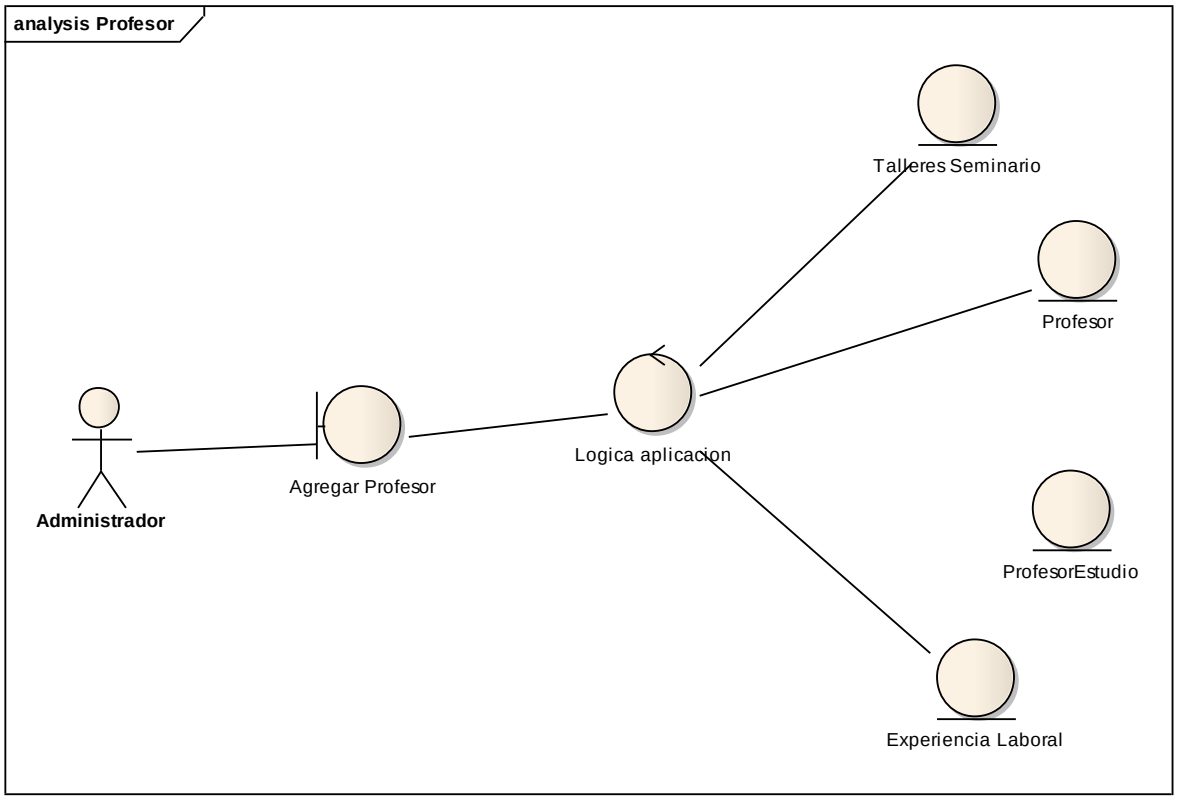


Figura 14 Modelo de Análisis del caso de uso agregar Profesor

Descripción del modelo de análisis:

1. Presentación de la interfaz "Registrar Profesor." Desde la cual la gestión académica puede realizar la función registrar un profesor.
2. Lógica de aplicación.
    a. Este realiza la validación de los datos antes de ser almacenados en la base de datos y la distribución de los datos del profesor en 4clasesenlazadas a 4 tablas en que se divide la información.
3. Dentro de una transacción, se actualiza la información en la base de datos:
    a. Actualiza la tabla de **tbl_Profesor**
    b. Actualiza la tabla de **tbl_ProfesorEstudio**
    c. Actualiza la tabla de **tbl_ProfesorTaller**
    d. Actualiza la tabla de **tbl_ProfesorExperiencia**

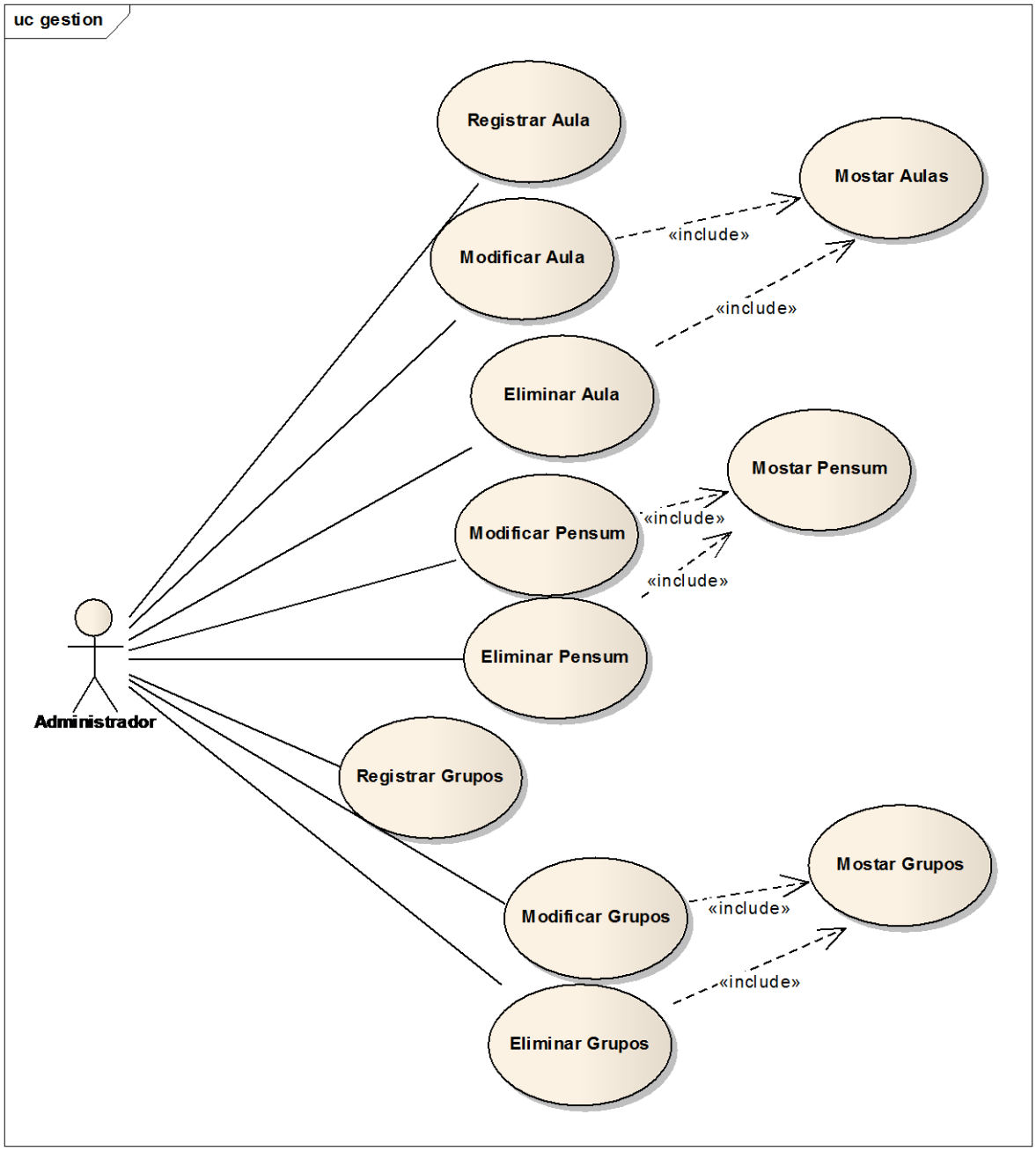


Figura 15 Diagrama de caso de uso administrador (parte 3)

En la figura 7 mostrada, presentan los Casos de Usos los cuales son: Agregar, Modificar, Eliminar, Mostrar: pensum, aulas, grupos de clase

Caso de uso: Registrar Pensum

Definición: Permite realizar el registro de los pensum del Colegio San Ramón Nonato.

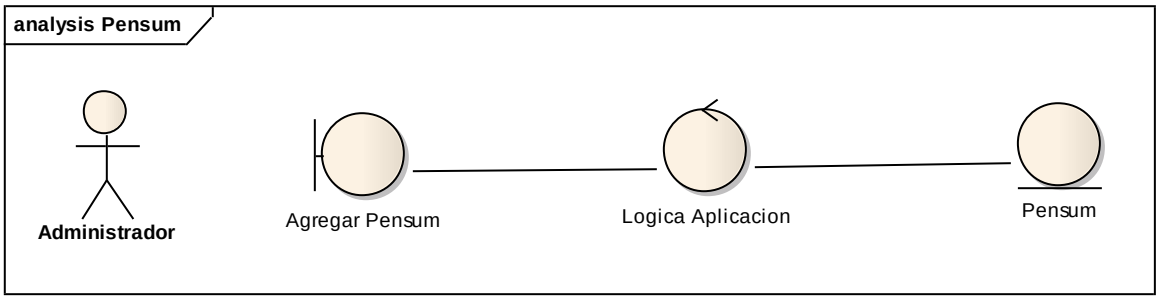


Figura 16 Modelo de Análisis del caso de uso Agregar Pensum

Descripción del modelo de análisis:

1. Presentación de la interfaz “Registrar Pensum” en la cual el gestor académico puede realizar las siguientes operaciones: editar el Pensum.
2. Lógica de Aplicación.
   a. Esta se encarga de la verificación de los datos que se están manipulando para la debida inserción en la base de datos.
3. Actualizar la información en la base de datos.

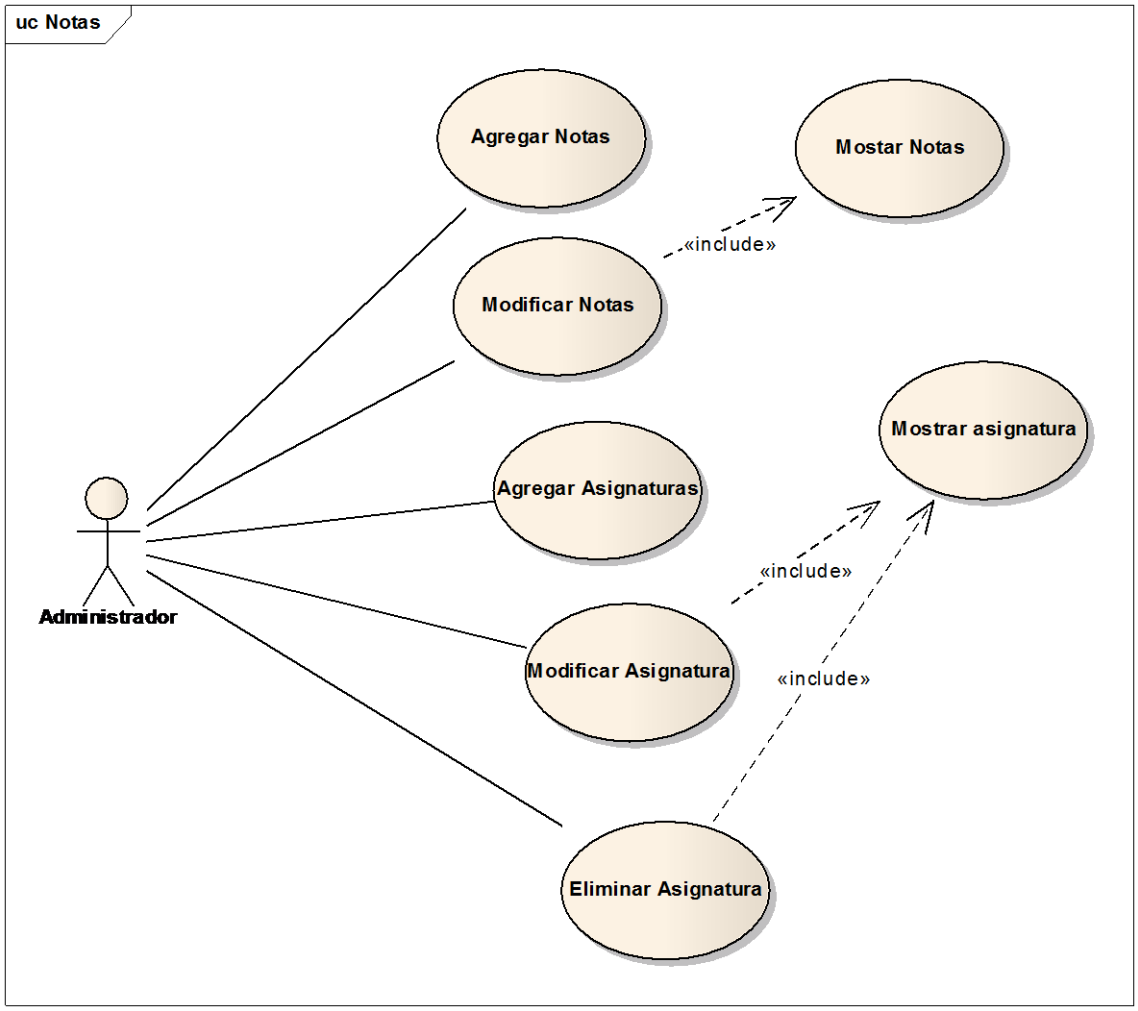


Figura 17 Diagrama de caso de uso Administrador (parte 4)

En la figura 9 mostrada, presentan los casos de usos las cuales son Agregar, Modificar, eliminar, Mostrar: notas y Asignatura.

Caso de uso: Registrar Notas de Estudiantes

Definición: Permite realizar el registro de notas de los estudiante del Colegio San Ramón Nonato según su número de carnet.

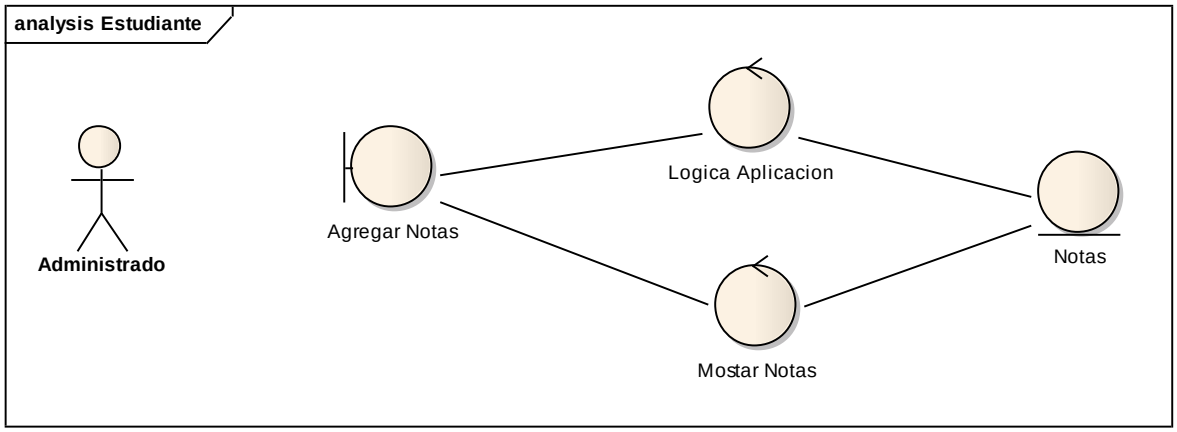


Figura 18 Modelo de Análisis del Caso de Uso Agregar Notas

Descripción del modelo de análisis:

1. Presentación de la interfaz “Registrar Notas” en la cual el gestor académico puede realizar la operación Modificar Notas.
2. Ejecución de la opción seleccionada que puede ser:
   a. Esta se encarga de verificar la nota que se está ingresando al estudiante según el grado y año de estudio.
3. Actualizar la información en la base de datos.
4. Mostar Notas este muestra las notas en la interfaz después de haber ingresado las notas del estudiante.

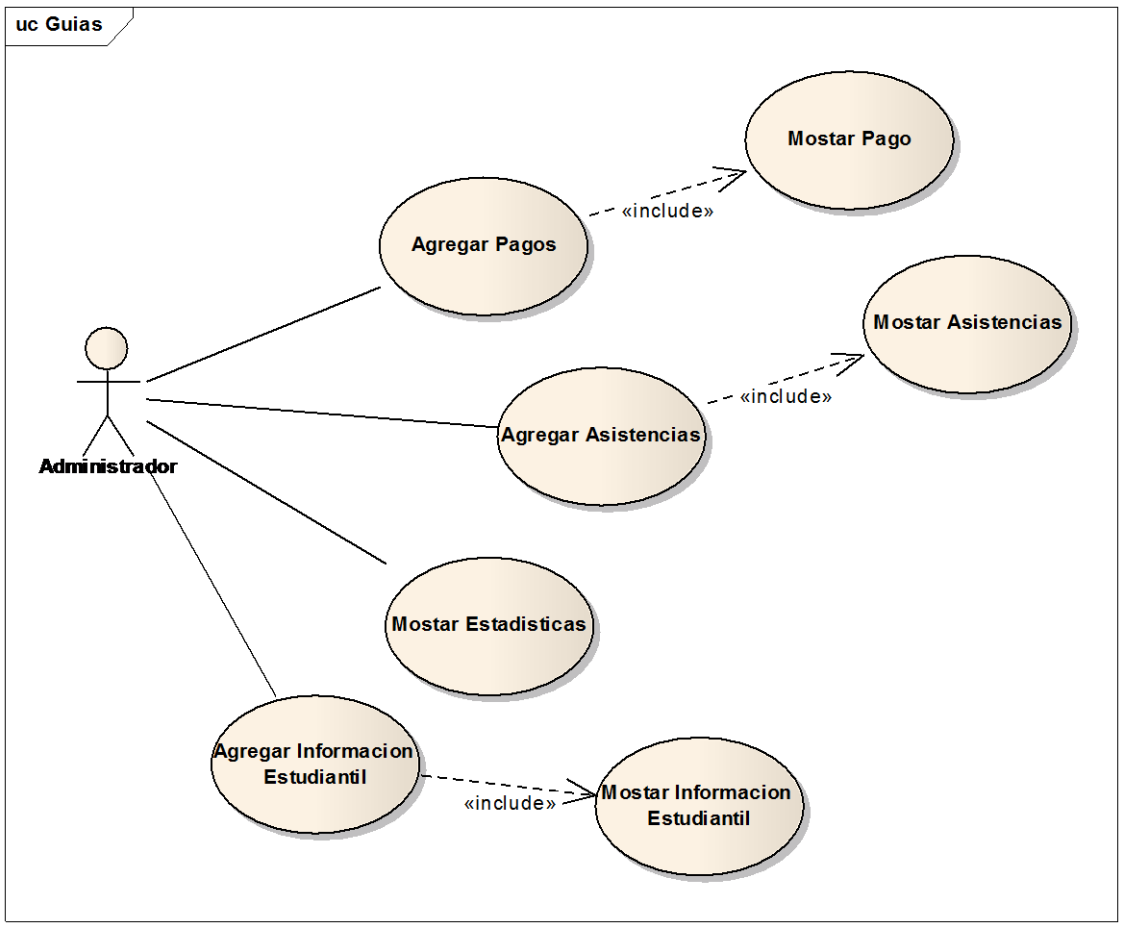


Figura 19 diagrama de caso de uso Administrador (parte 5)

La figura 11 muestra las funciones de los casos de uso Agregar y mostrar: Estadísticas, Información Estudiantil, Pagos, Asistencias.

Caso de uso: Mostrar información Estudiantil.

Definición: Permite mostrar el registro de la información estudiantil y la agregación de opinión referente al estudiante.

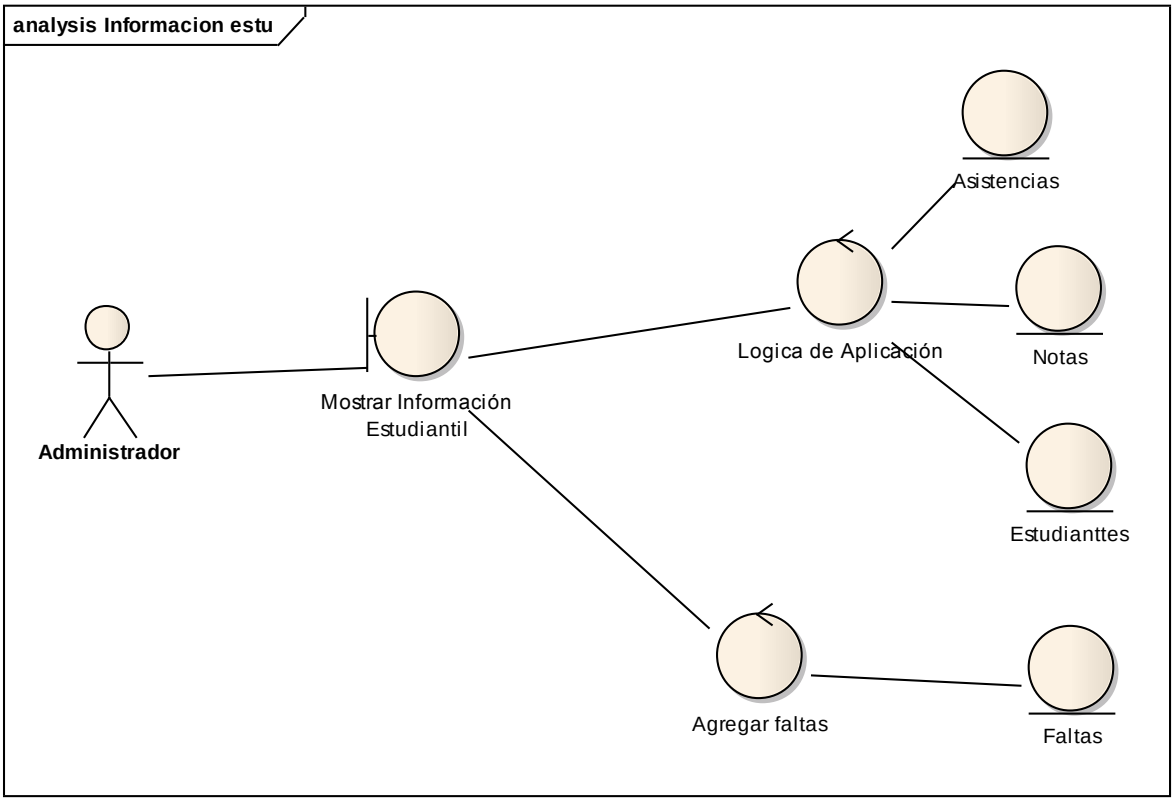


Figura 20 Modelo de análisis de Caso de uso Información Estudiantil.

Descripción del modelo de análisis:

1. Presentación de la interfaz “Mostrar Información Estudiantil” del Estudiante.
2. Lógica de Aplicación esta operación se encarga de mostrar:
   a. Información personal de los estudiantes
   b. Notas de los estudiantes
   c. Información de las asistencias
3. Agregar faltas este proceso realiza la inserción de la información disciplinaria de los estudiantes.
4. Actualizar la información de la base de datos.

4.6 Rol de los Usuarios

Roles	Profesores	Estudiantes	Pensum	Horario
Contador				
Disciplina				
Guías	• Ver Profesores	• Ver Estudiantes		• Ver Horario Clases
Secretario		• Ingresar Estudiantes • Modificar Estudiantes • Ver Estudiantes • Generar información • Matricular • Ver Matricula • Retirar Estudiante • Registrar Nota		
Gestión	• Ingresar Profesores • Modificar Profesores • Ver Profesores • Generar información	• Ingresar Estudiantes • Modificar Estudiantes • Ver Estudiantes • Generar información • Matricular • Ver Matricula • Retirar Estudiante • Registrar Nota	• Ingresar Pensum • Modificar Pensum • Eliminar Pensum • Ver Pensum	• Ingresar Horario • Modificar Horario • Eliminar horario • Ver Horario
Matricula		• Ingresar Estudiantes • Modificar Estudiantes • Ver Estudiantes • Generar información • Matricular • Ver Matricula		
Director	• Ingresar Profesores	• Ingresar Estudiantes	• Ingresar Pensum	• Ingresar Horario

	• Modificar Profesores • Ver Profesores • Generar información	• Modificar Estudiantes • Ver Estudiantes • Generar información • Matricular • Ver Matricula	• Modificar Pensum • Eliminar Pensum • Ver Pensum	• Modificar Horario • Eliminar horario • Ver Horario

Roles	**Grupo**	**Aula**	**Notas**	**Información Estudiantil**
Contador				
Disciplina				• Ingresar Información • Ver Información
Guías	• Ver Aulas	• Ver Aulas		
Secretario			• Ingresar Notas • Modificar Notas • Ver Notas • Generar Reportes • Generar Constancias	
Gestión	• Ingresar Grupos • Modificar Grupos • Eliminar Grupos • Ver Grupos	• Ingresar Aulas • Modificar Aulas • Eliminar Aulas • Ver Aulas		
Matricula			Ver Notas	Ver Información Estudiantil
Director	• Ingresar Grupos • Modificar Grupos • Eliminar Grupos • Ver Grupos	• Ingresar Aulas • Modificar Aulas • Eliminar Aulas • Ver Aulas	• Ingresar Notas • Modificar Notas • Ver Notas • Generar Reportes • Generar Constancias	• Ingresar Información • Ver Información

Roles	Estadística	Asistencia	Pago del Mes
Contador			• Ingresar Pago • Modificar Pago • Ver Pagos
Disciplina		• Generar Asistencia • Agregar Asistencia • Ver Asistencia	
Guías			
Secretario			
Gestión	• Generar Estadísticas • Ver Estadísticas		
Matricula			
Director	• Generar Estadísticas • Ver Estadísticas	• Generar Asistencia • Agregar Asistencia • Ver Asistencia	• Ingresar Pago • Modificar Pago • Ver Pagos

5. Diagrama de clases

Diagrama de Clase

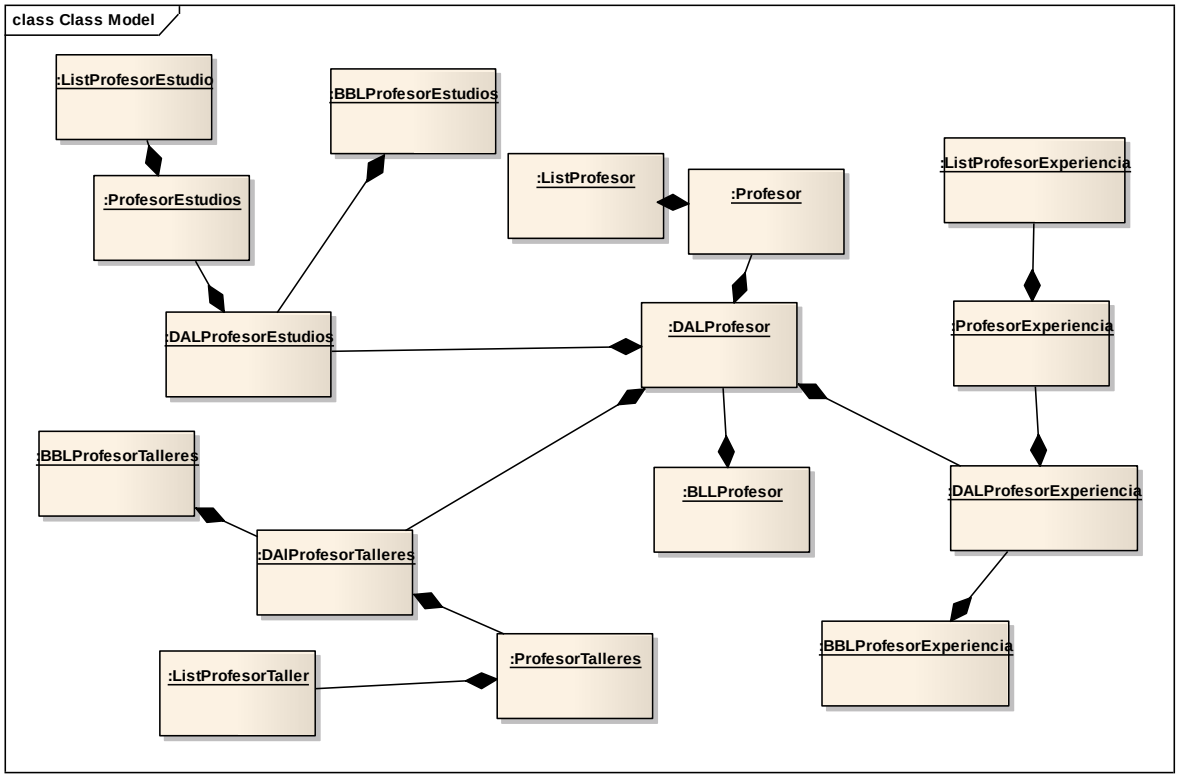


Figura 21 Diagrama de clase profesor

Estas son todas las clases que se encargan de la manipulación de la información de los profesores. A continuación explicamos la utilidad de algunas de ellas:

DALProfesor: Este se encarga del control de la inserción de los datos a cada una de las tablas a través de procedimientos almacenados mediante los cuales accede a cada una de las tablas para la debida inserción de los datos, además esta clase utiliza transacciones para la debida inserción y si ocurre un error no se realiza la operación.

ProfesorEstudio: Esta clase se encarga de la asignación de los datos obtenidos por la clase **DALProfesor** que es la que le envía los datos para su debida manipulación por la clase **ListProfesorEstudio** para la inserción con el procedimiento que le corresponde a dicha clase.

BLLProfesorTaller: Esta clase que es la capa de presentación que es la que se encarga de enlazar las demás clases para la debida utilización y manipulación de los datos.

ListProfesorSeminario: Es una lista que almacena los datos de la información de haberlo obtenidos por la clase DALProfesor este se utiliza para la manipulación de los datos y luego ser insertado con su procedimiento respectivo de la clase.

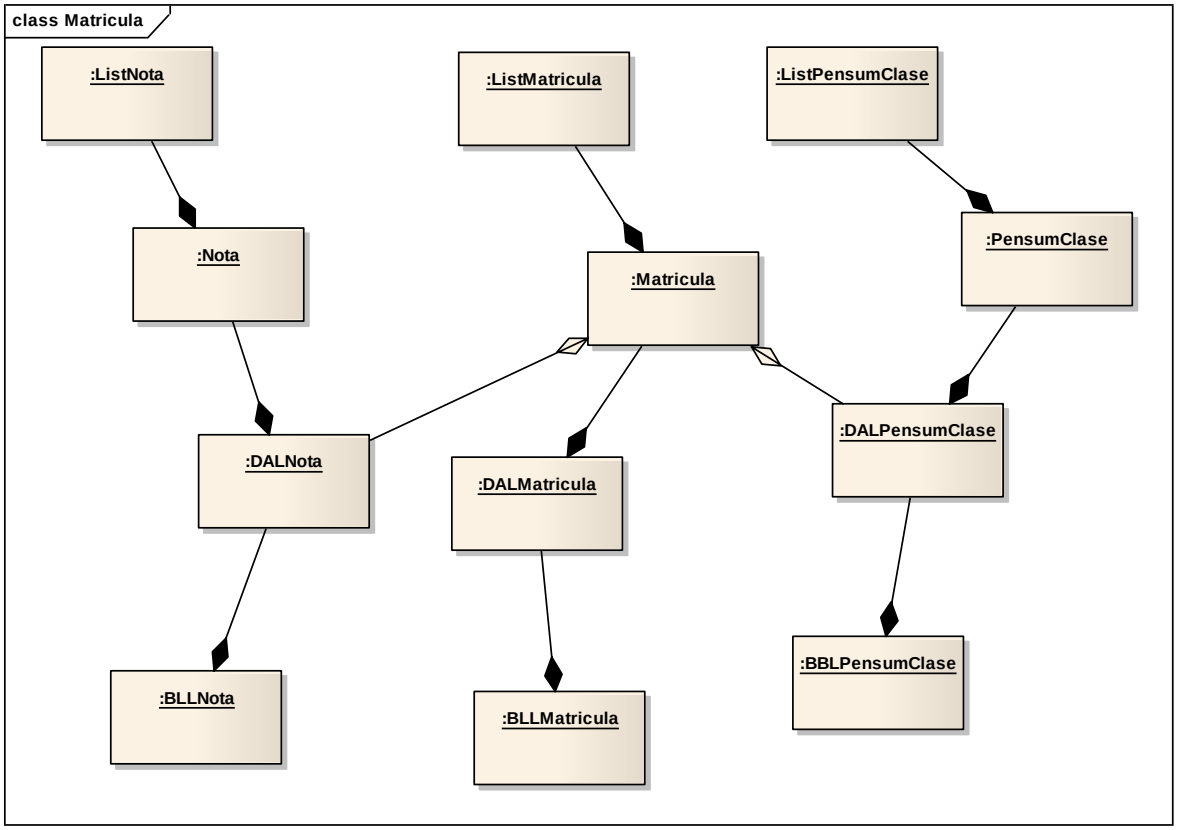


Figura 22 Diagrama de clase matricula

Estas son todas las clases que se encargan de la manipulación de la información de las Matriculas. A continuación explicamos la utilidad de algunas de ellas:

DALMatricula: Esta clase se encarga de realizar las matriculas a los estudiantes y asignarles sus debidas clases según el pensum actual.

Nota: Con esta realizamos la debida inserción de las asignaturas de los profesores para su procedimiento y se le asigna cero por defecto a todas las asignaturas.

BBLPensunclase: En esta clase se realiza la operación de la extracción de las asignaturas debido al pensum actual que se está llevando en el año actual.

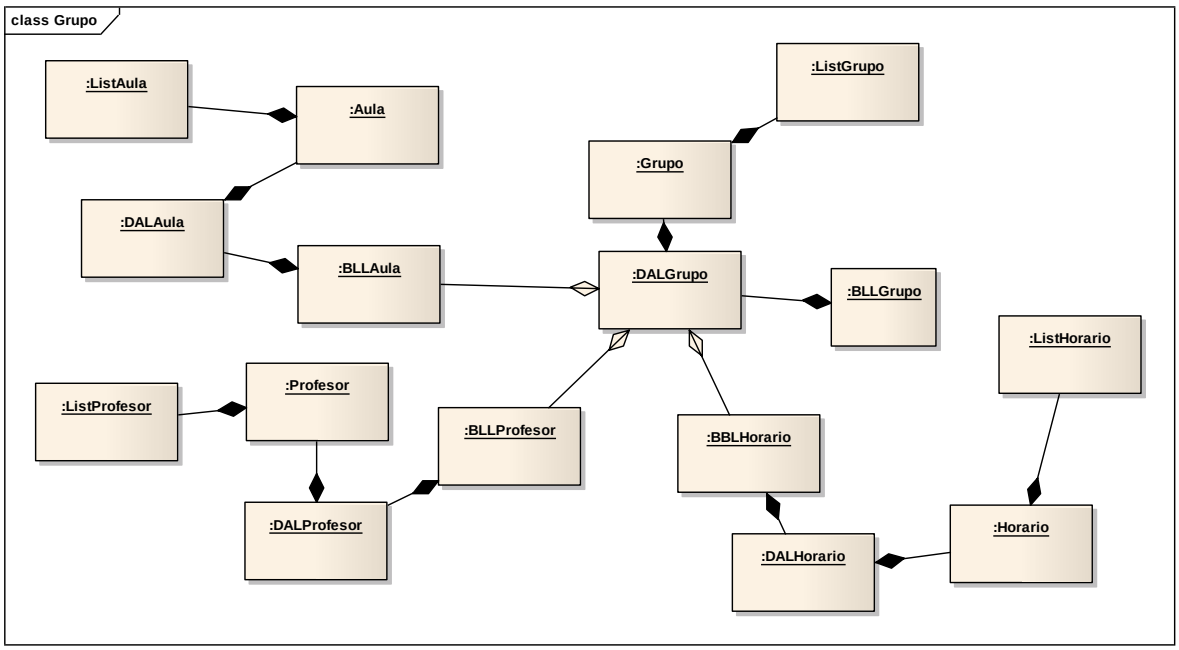


Figura 23 Diagrama de clase grupo

Estas son todas las clases que se encargan de la manipulación de la información de los grupos. A continuación explicamos la utilidad de algunas de ellas:

DALGrupo: Esta clase es encargada de la creación de los grupos con ayuda de las clases para la debida creación de los grupos y sus profesores guía.

Aula: La clase se encarga de la obtención de los datos de las respectivas aulas para su debida utilización de ellas.

BLLProfesor: Esta clase obtiene los datos relacionados de los profesores para su debida asignación a sus respectivos grupos como profesores guía.

ListHorario: Se realiza la manipulación de los datos referente al horario de clase y de cada profesor.

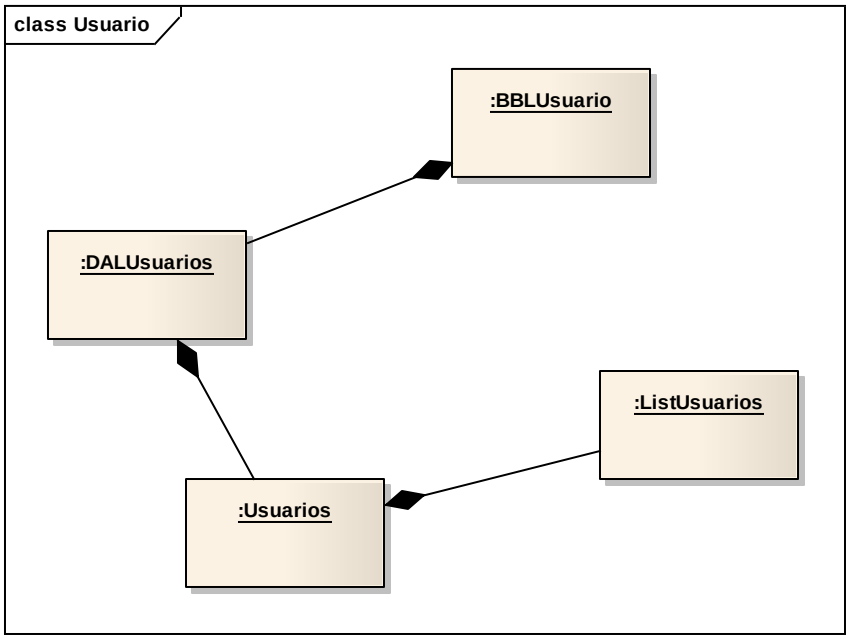


Figura 24 diagrama de clase Usuario

Estas son todas las clases que se encargan de la manipulación de la información de los profesores. A continuación explicamos la utilidad de algunas de ellas:

DALUsuario: En esta clase se realiza la captura de todos los usuario que van a poder acceder a la aplicación en esta se encarga de también de asignarle el rol que tendrá el usuario que va a acceder a la aplicación y a cada módulo que tiene acceso para su manipulación.

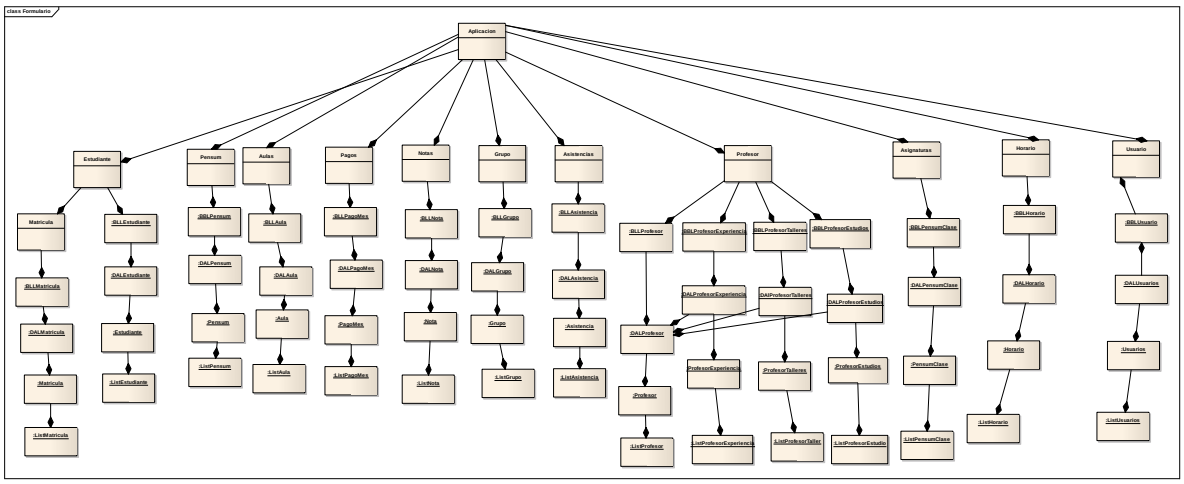


Figura 25 Diagrama de las Clases y Aplicación

Estudiante: Este módulo se encarga de todo lo referente a las clase estudiantes esta llama a las clases para su debido inserción dentro de ella se encuentra al llamado de dos clase una para la debida inserción de los datos de los estudiantes y el otro para ver a los estudiantes y a través de la cual puede acceder a distintos módulos más como insertar notas, matricular ver información en vista de impresión y modificar su información.

Grupo: En este módulo se encarga de la creación de grupos además en este módulo esta ver grupos existentes y además la de asignación de grupos en este módulo se encarga de la asignación de los estudiantes a sus respectivos grupos a los que van a pertenecer cada uno de ellos.

Aula: Esta se encarga de la inserción de las aulas según su código en esta se Procesan todas las aulas que van a ser ocupadas por todos los estudiantes del colegio.

6. Diagrama de secuencia

Los diagramas de secuencia que se muestran a continuación son en referencia a los casos de usos de cuales se realizó el modelo de análisis.

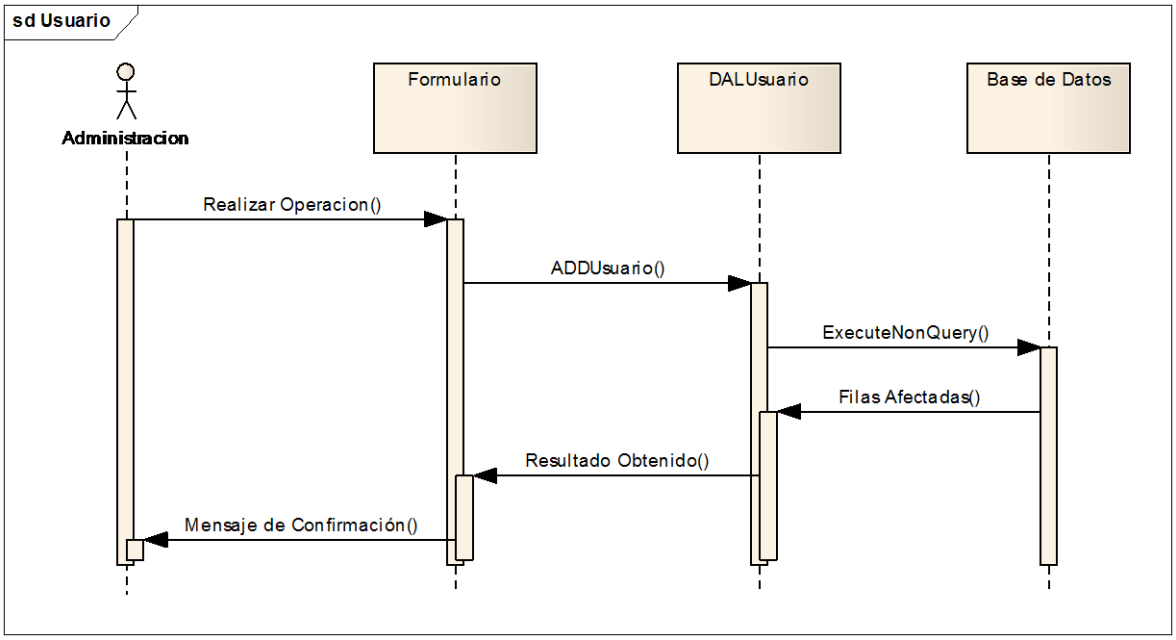


Figura 26 Diagrama de secuencia del caso de uso agregar cuenta de usuario

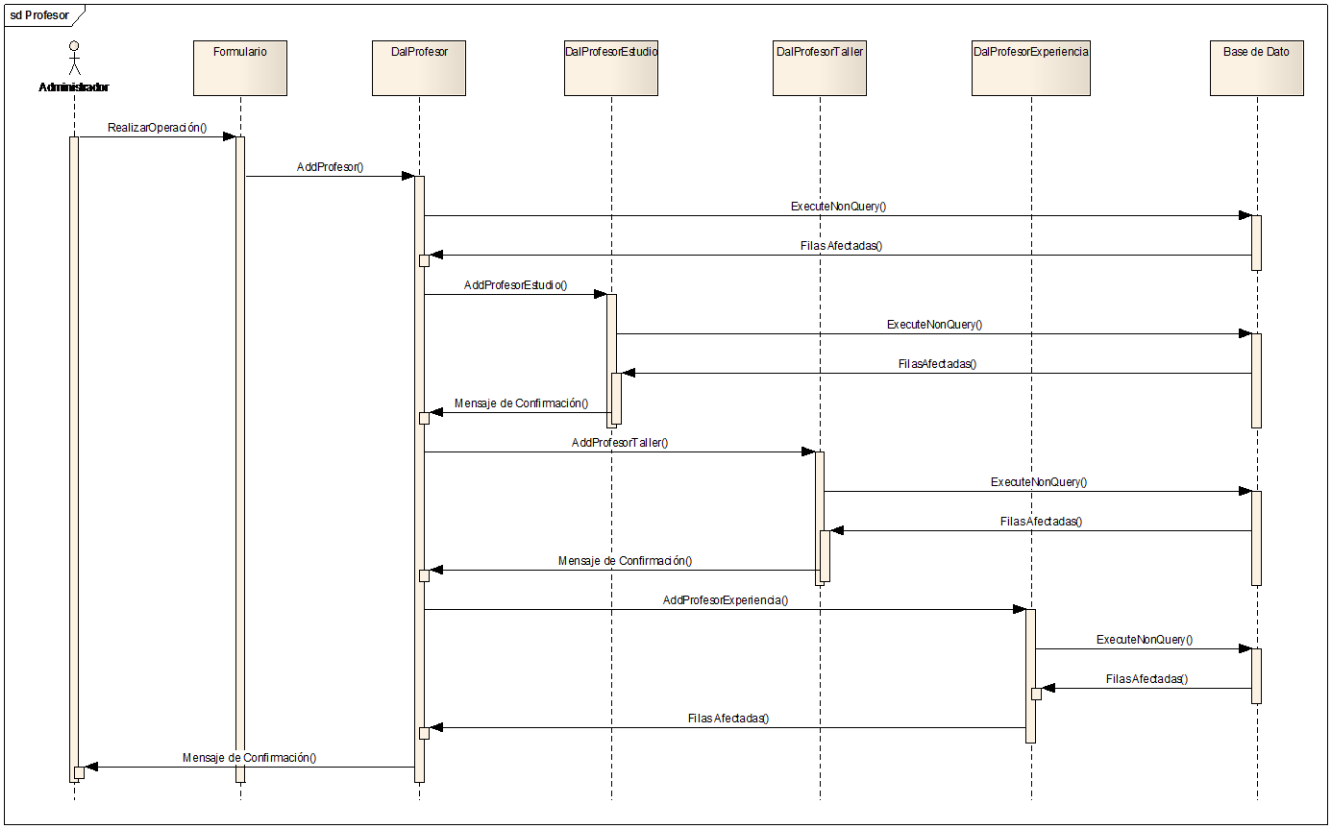


Figura 27 diagrama de secuencia del caso de uso agregar profesor

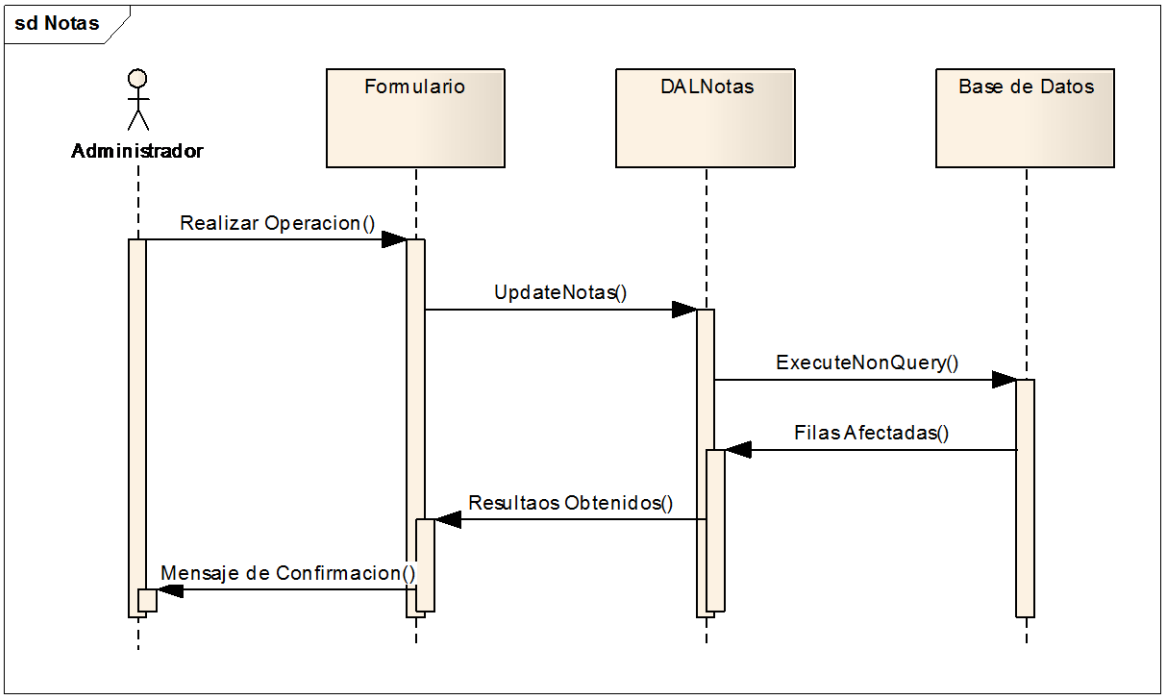


Figura 28 diagrama de secuencia del caso de uso agregar notas

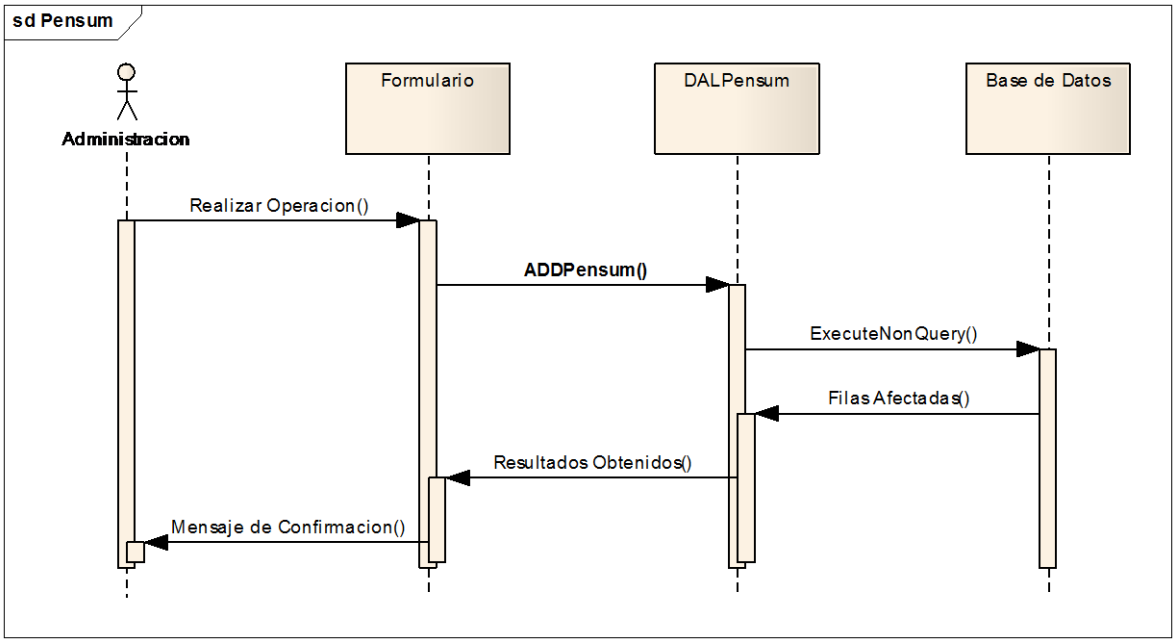


Figura 29 Diagrama de secuencia agregar pensum

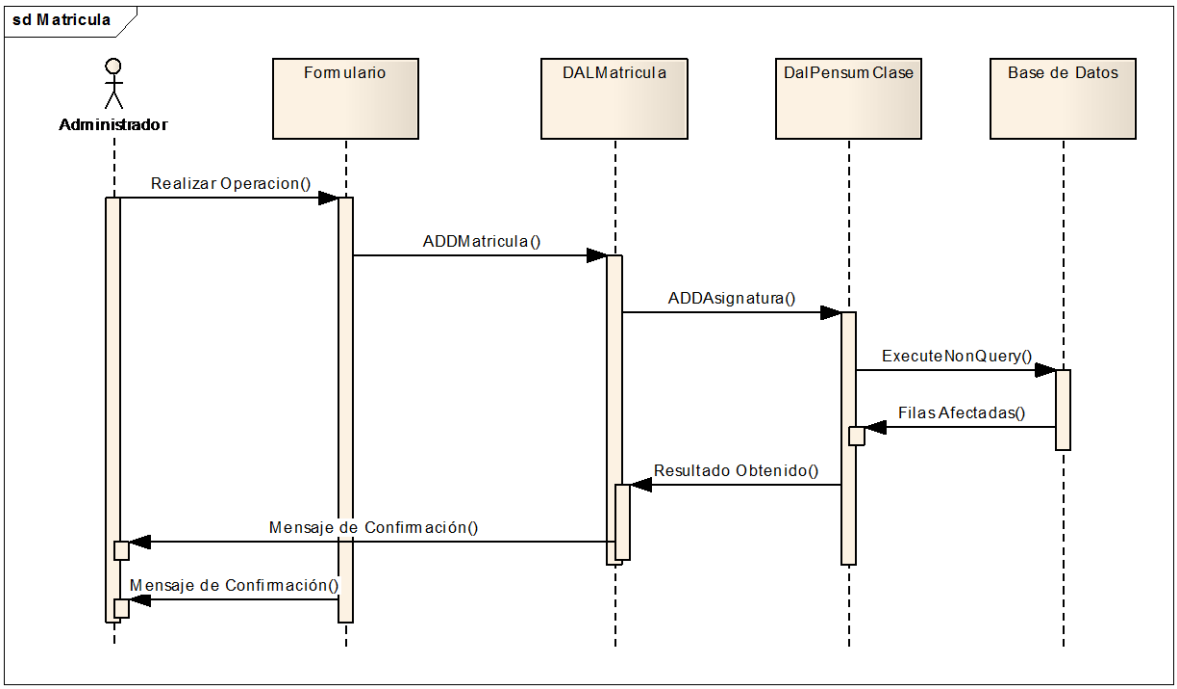


Figura 30 Diagrama de Secuencia Agregar Matricula

7. Diagrama de Despliegue

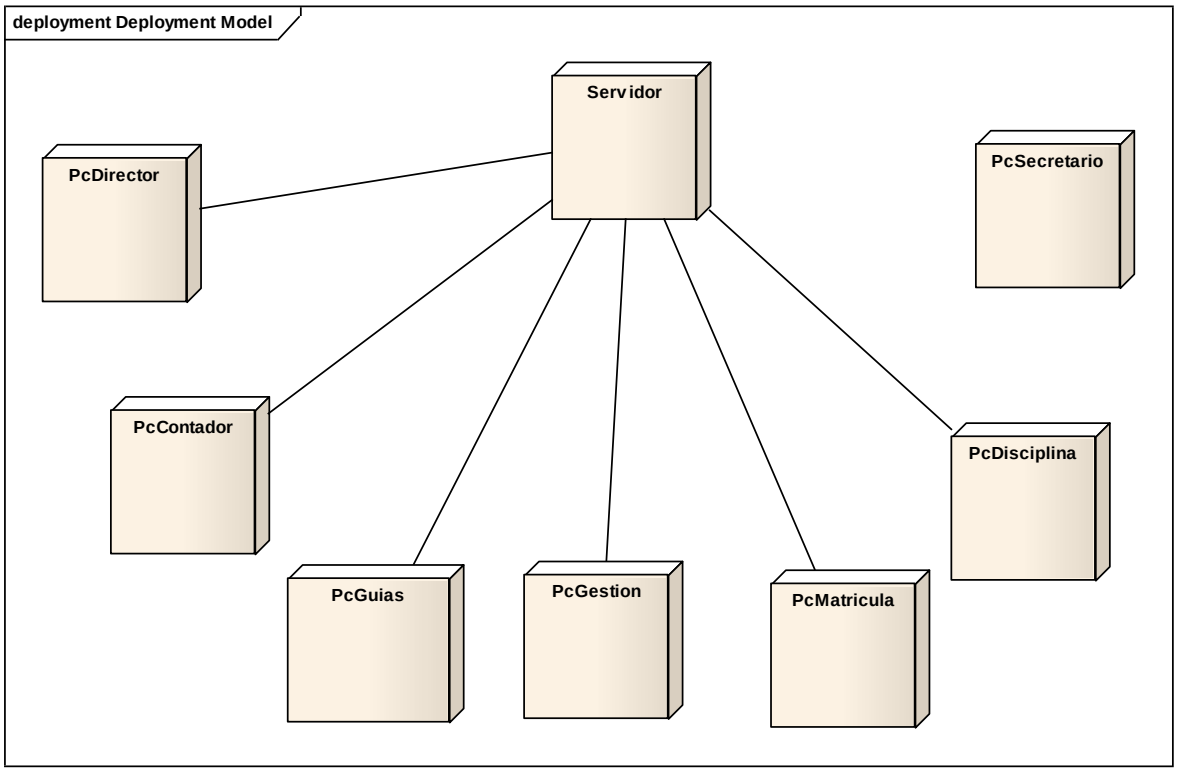


Figura 31 Diagrama de Despliegue

En la figura 31 se muestra la forma de como quedara distribuida la aplicación una vez finalizada, fácilmente en el diagrama se pueden ver los 7 tipos de usuarios que tendrá la aplicación, en el equipo servidor estará instalado Windows server 2003 y SQL Server 2005, a este equipo servidor solamente tendrá acceso el administrador del sistema, los usuarios clientes tendrán acceso únicamente a los módulos que se le sean asignados por el administrador.

8. Diseño de Datos

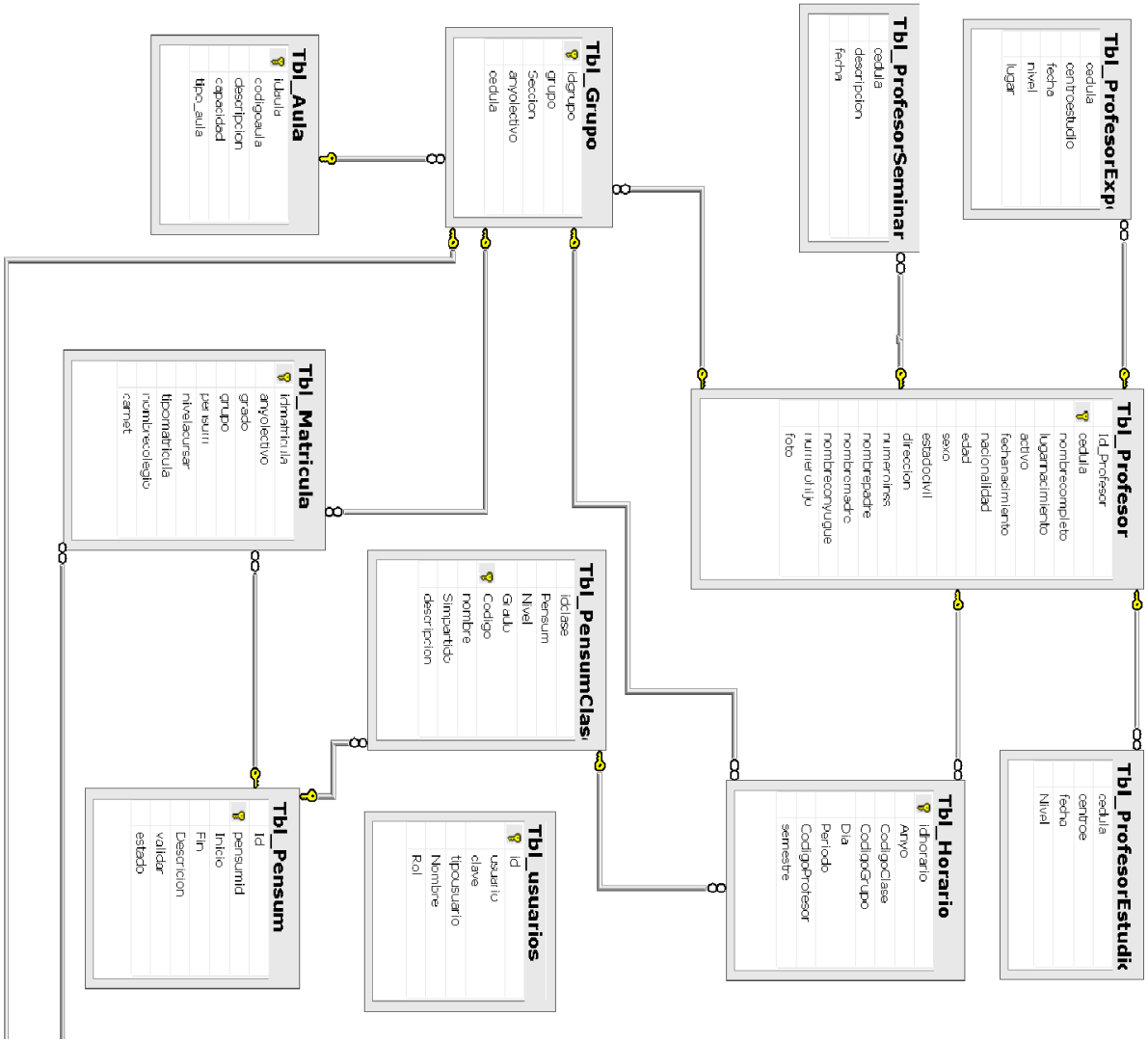


Figura 32 Diagrama de Clase 1 Parte

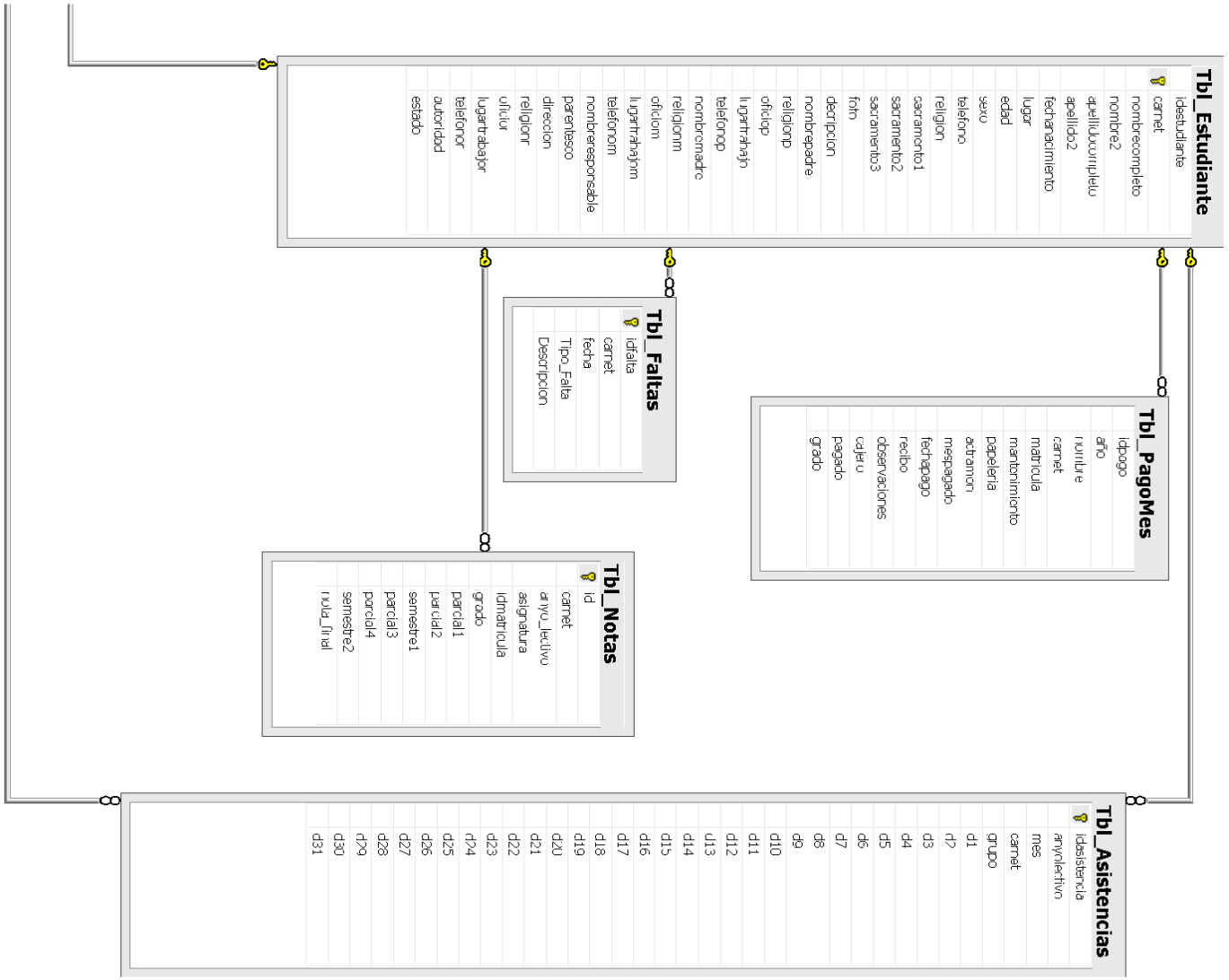


Figura 33 Diagrama de Clase 2 Parte

Tabla Pensum

Almacena todos los datos de los pensum que se encuentran registrados en Colegio Tridentino San Ramón.

Nombre	Descripción	Tipo
Id	Llave primaria.	int
pensumid	Id del Pensum.	Varchar(10)
Inicio	Inicio del pensum.	int
Fin	Fin del pensum.	Int
Descripción	Descripción del Pensum.	Varchar(500)
Validar	Si el Pensum Esta Activo o Inactivo.	Varchar(20)
Estado	Si el Pensum está terminado o en Proceso.	Varchar(10)

Tabla PensumClases

Almacena todos los datos de las asignaturas impartidas en el año lectivo según su pensum para el Colegio San Ramón.

Nombre	Descripción	Tipo
idclase	Identificación del pensum.	int
pensum	Nombre del pensum	Varchar(10)
nivel	Nivel Impartido de las asignaturas	Varchar(11)
grado	Grado en que se recibe la asignatura.	Varchar(4)
Codigo	Llave primaria.	Varchar(20)
nombre	Nombre de La asignatura	Varchar(50)
simpartido	En que semestre es impartido.	Varchar(20)
descripción	Información de la asignatura	Varchar(200)

Tabla Tbl_Asistencia

Almacena todas las operaciones que se realizan en el módulo de asistencias del Colegio Tridentino San Ramón.

Nombre	Descripción	Tipo
idasistencia	Llave primaria.	int
anyolectivo	Año lectivo escolar	Varchar(50)
mes	Mes generado para la asistencia	Varchar(50)
carnet	Número de carnet del estudiante	varchar(50)
grupo	Grado al que pertenece el estudiante	varchar(50)
D1	Día del mes	Varchar(15)
D2	Día del mes	Varchar(15)
D3	Día del mes	Varchar(15)
D4	Día del mes	Varchar(15)
D5	Día del mes	Varchar(15)
D6	Día del mes	Varchar(15)
D7	Día del mes	Varchar(15)
D8	Día del mes	Varchar(15)
D9	Día del mes	Varchar(15)
D10	Día del mes	Varchar(15)
D11	Día del mes	Varchar(15)
D12	Día del mes	Varchar(15)
D13	Día del mes	Varchar(15)
D14	Día del mes	Varchar(15)
D15	Día del mes	Varchar(15)
D16	Día del mes	Varchar(15)
D17	Día del mes	Varchar(15)
D18	Día del mes	Varchar(15)
D19	Día del mes	Varchar(15)
D20	Día del mes	Varchar(15)
D21	Día del mes	Varchar(15)
D22	Día del mes	Varchar(15)
D23	Día del mes	Varchar(15)
D24	Día del mes	Varchar(15)
D25	Día del mes	Varchar(15)
D26	Día del mes	Varchar(15)
D27	Día del mes	Varchar(15)
D28	Día del mes	Varchar(15)
D29	Día del mes	Varchar(15)
D30	Día del mes	Varchar(15)
D31	Día del mes	Varchar(15)

Tabla Tbl_Aula

Almacena todos los datos de las aulas del Colegio Tridentino San Ramón

Nombre	Descripción	Tipo
idaula	Llave primaria.	int
codigoaula	Código del aula	Varchar(50)
descripcion	Descripción del aula	Varchar(50)
capacidad	Cantidad de estudiante por sección	Varchar(50)
Tipo_aula	Tipo de aula compartida o no	varchar(50)

Tabla Tbl_Estudiante

Almacena todos los registros referentes a todos les estudiantes del Colegio Tridentino San Ramón.

Nombre	Descripción	Tipo
Idestudiante	Identificador del estudiante	int
Carnet	Llave primaria	varchar(50)
Nombrecompleto	Primer nombre del estudiante	varchar(50)
Nombre2	Segundo nombre del estudiante	varchar(50)
apellidocompleto	Primer apellido del estudiante	varchar(50)
apellido2	Segundo nombre del estudiante	varchar(50)
Fechanacimiento	Fecha de nacimiento del estudiante	Datetime
lugar	Dirección del estudiante	varchar(50)
edad	Edad del estudiante	varchar(10)
Sexo	Sexualidad del estudiante	varchar(10)
Teléfono	Teléfono del estudiante	varchar(50)
religion	Religión del estudiante	varchar(50)
sacramento1	Sacramento del estudiante	varchar(20)
Sacramento2	Sacramento del estudiante	varchar(20)
Sacramento3	Sacramento del estudiante	varchar(20)
Foto	Foto del estudiante	Image
Decripcion	Documento pendiente del estudiante	varchar(50)
nombrepadre	Nombre del padre	varchar(50)
Religionp	Religión del padre	varchar(50)
Oficiop	Oficio o profesión del padre	varchar(50)
lugartrabajo	Lugar de trabajo del padre	varchar(50)
telefonop	Teléfono del padre	varchar(50)
Nombremadre	Nombre de la madre	varchar(50)

Religionm	Religión de la madre	varchar(50)
Oficiom	Oficio o profesión de la madre	varchar(50)
lugartrabajom	Lugar de trabajo de la madre	varchar(50)
telefonom	Teléfono de la madre	varchar(50)
nombreresponsable	Nombre del responsable	varchar(50)
parentesco	Parentesco con el estudiante	varchar(50)
direccion	Dirección del responsable	varchar(50)
Religionr	Religión del responsable	varchar(50)
oficior	Oficio del responsable	varchar(50)
lugartrabajor	Lugar de trabajo del responsable	varchar(50)
telefonor	Teléfono del responsable	varchar(50)
autoridad	Si tiene autoridad el responsable	varchar(50)
estado	Si o no tiene autoridad	varchar(20)

TablaTbl_Grupo

Almacena todos los registros de los grupos con los profesores guías y sus aulas correspondientes.

Nombre	Descripción	Tipo
Idgrupo	Llave primaria	int
Grado	Grado del estudiante	Varchar(20)
Grupo	Grupo de los estudiante	Varchar(20)
Anyolectivo	Año lectivo al grupo	Varchar(30)
cedula	Numero de cedula del profesor	Varchar(20)

Tabla Tbl_Horario

Almacena todos los registros de todos los horarios de los profesores con sus clases.

Nombre	Descripción	Tipo
Idhorario	Identificación del horario	int
Anyo	Año lectivo escolar	Int
Codigoclase	Código de la clase	Varchar(20)
Codigogrupo	Código del grupo	Int
Dia	Día que se imparte la clase	Varchar(10)
Periodo	Periodo que se imparte la clase	Int
Codigoprofesor	Código del profesor	Varchar(16)
semestre	Semestre impartido	Varchar(50)

Tabla Tbl_Matricula

Almacena todos los registros referentes a las matrículas de los estudiantes del Colegio Tridentino San Ramón.

Nombre	Descripción	Tipo
idmatricula	Llave primaria	Int
Anyolectivo	Año lectivo escolar	Varchar(10)
Grado	Grado que cursa el estudiante	Varchar(10)
Grupo	Id del grupo al que pertenece el estudiante	Int
Pensum	Pensum al que es asociado el estudiante	Varchar(20)
Nivelacursar	Nivel a cursar(primaria, secundaria, preescolar)	Varchar(20)
Tipomatricula	Tipo de matrícula (nuevo, reingreso etc.)	Varchar(30)
Nombrecolegio	Nombre del colegio proveniente si es nuevo ingreso	Varchar(30)
carnet	Número de carnet del estudiante	Varchar(20)

TablaTbl_Notas

Almacena todos los registros de las notas de los estudiantes del Colegio Tridentino San Ramón.

Nombre	Descripción	Tipo
Id	Llave primaria	int
Carnet	Número de carnet del estudiante	Varchar(20)
Anyo_lectivo	Año electivo escolar	Varchar(10)
Asignatura	Asignaturas que lleva el estudiante	Varchar(30)
Idmatricula	Id de la matricula referente al grado que esta	int
Grado	Grado que cursa el estudiante	Varchar(20)
Parcial1	Nota del primer parcial	Int
Parcial2	Nota del segundo parcial	Int
Semestre1	Nota del primer semestre	Int
Parcial3	Nota del tercer parcial	Int
Parcial4	Nota del cuarto parcial	Int
Semestre2	Nota del segundo semestre	Int
Nota_final	Nota final de la asignatura	int

Tabla Tbl_PagoMes

Almacena todos los registros referentes a los pagos de la mensualidad de cada uno de los estudiantes del colegio.

Nombre	Descripción	Tipo
Idpago	Llave primaria	int
Año	Año electivo	Varchar(20)
Nombre	Nombre del estudiante	Varchar(20)
Carnet	Número de carnet	Varchar(20)
Matricula	Pago de matricula	Varchar(20)
Mantenimiento	Pago del mantenimiento	Varchar(20)
Papelería	Pago de papelería	Varchar(20)
Actramon	Pago de actas de san ramón	Varchar(20)
Mespagado	Mes que se pago	Varchar(20)
Fechapagor	Fecha de pago del mes	Varchar(20)
Recibo	Numero de recibo del mes pagado	Varchar(20)
Observaciones	Si el estudiante tiene documentos o pagos pendientes	Varchar(20)
Cajero	Nombre del que realizo la factura de pago	Varchar(20)
Pagado	Si pago o es pendiente	Varchar(20)
grado	Grado del estudiante	Varchar(20)

Tabla Tbl_Profesor

Almacena todos los registros referentes a todos los profesores del Colegio Tridentino San Ramón

Nombre	Descripción	Tipo
Id_Profesor	Identificador del profesor	int
cedula	Llave primaria	varchar(16)
nombrecompleto	Nombre del profesor	varchar(50)
lugarnacimiento	Lugar de nacimiento	varchar(50)
activo	Si el profesor es activo o no	bit
fechanacimiento	Fecha de nacimiento del profesor	Datetime
nacionalidad	Nacionalidad del profesor	varchar(50)
edad	Edad del profesor	int
Sexo	Sexo del profesor	varchar(10)
estadocivil	Estado civil del profesor	varchar(10)
direccion	Dirección de donde vive el profesor	varchar(150)
Numeroinss	Numero de inss del profesor	varchar(9)
nombrepadre	Nombre del padre	varchar(50)
nombremadre	Nombre de la madre	varchar(50)
nombreconyugue	Nombre de la persona con la que vive	varchar(50)
Numerohijo	Número de hijos	int
foto	Foto del profesor	Image

Tabla Tbl_ProfesorEstudios

Almacena todos los registros referentes a los estudios realizados por los profesores del Colegio Tridentino san Ramón.

Nombre	Descripción	Tipo
Cedula	Cedula del profesor	Varchar(16)
Centroe	Centro de estudios del profesor	Varchar(100)
Fecha	Fecha en la que las realizo	Int
nivel	Nivel en los que estuvo y obtenidos	Varchar(50)

Tabla Tbl_ProfesorExperiencia

Almacena todos los registros referentes a las experiencias laborales realizados por los profesores del colegio Tridentino San Ramón.

Nombre	Descripción	Tipo
cedula	Cedula de los profesores	Varchar(16)
Centroestudio	Centro donde trabajo anteriormente	Varchar(50)
Fecha	Fecha que realizo	Int
Nivel	Nivel impartidos	Varchar(50)
lugar	Lugar donde trabajo anteriormente	Varchar(50)

Tabla Tbl_ProfesorSeminario

Almacena todos los registros referentes a los seminarios que a participado los profesores del Colegio Tridentino San Ramón.

Nombre	Descripción	Tipo
cedula	Cedula de los profesores	Varchar(16)
Descripción	Seminarios recibidos	Varchar(500)
Fecha	Fecha que los recibió	int

Tabla Usuarios

Almacena todos los registros de los usuarios que tiene acceso a la aplicación con sus roles

Nombre	Descripción	Tipo
Id	Llave primaria	Int
Usuario	Nombre del usuario	Varchar(15)
Clave	Clave del usuario	Varchar(40)
Tipousuario	Tipo de usuario	Varchar(20)
Nombre	Nombre del usuario registrado	Varchar(60)
rol	Asignación del tipo de rol	int

9. Diseño de interfaz

Figura 34 Inicio de la Aplicación.

Figura 35 Interfaz de Usuario

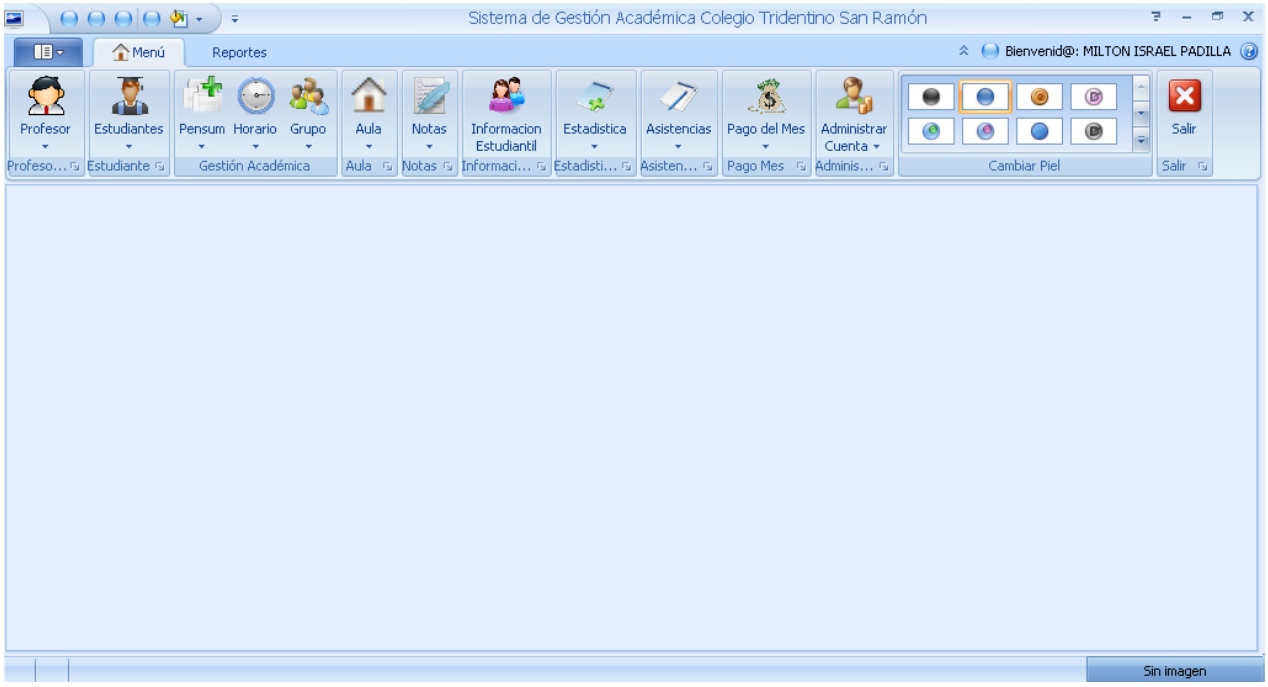


Figura 36 Interfaz del Principal

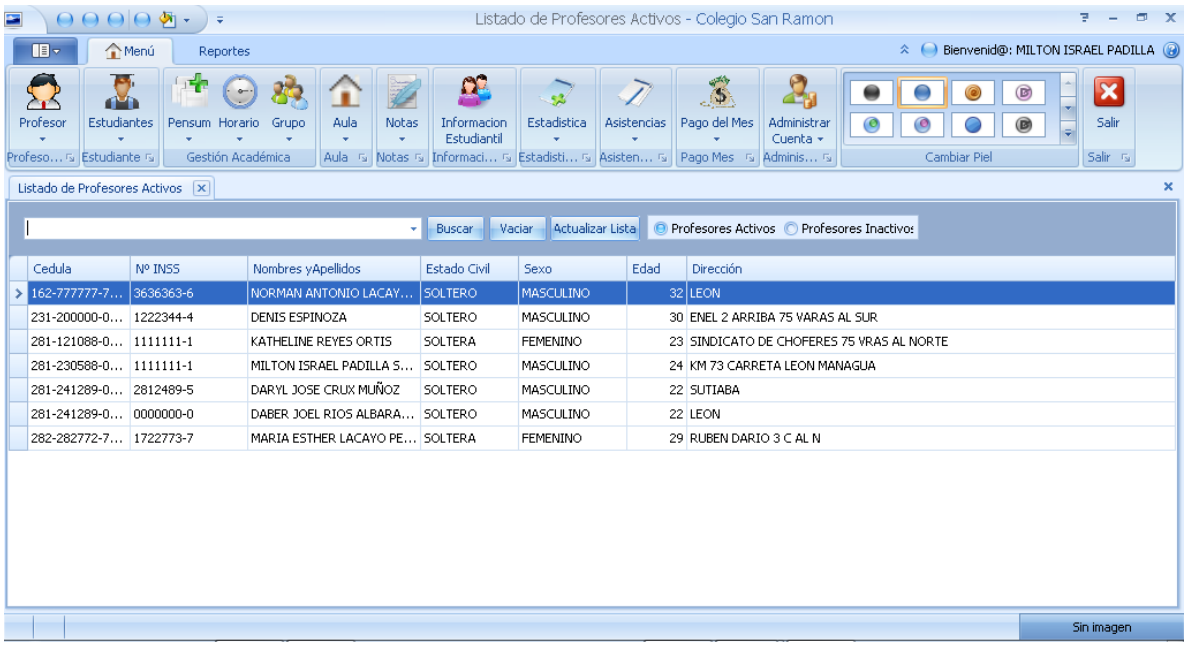


Figura 37 Interfaz Listado de los Profesores

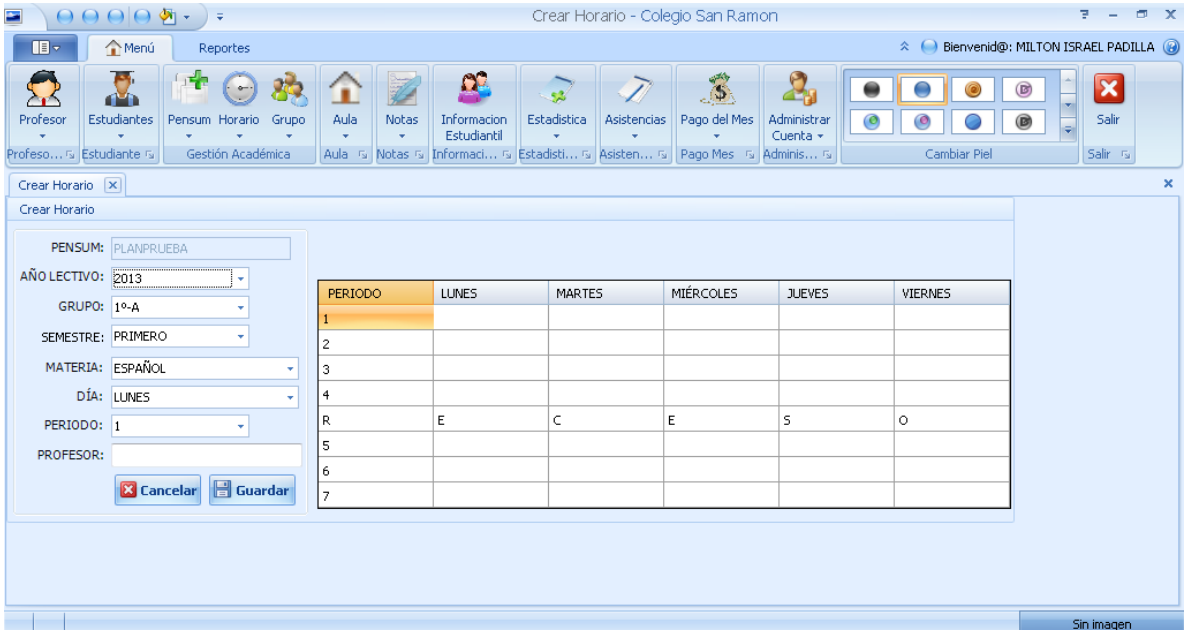


Figura 38 Interfaz de crear horario de clases

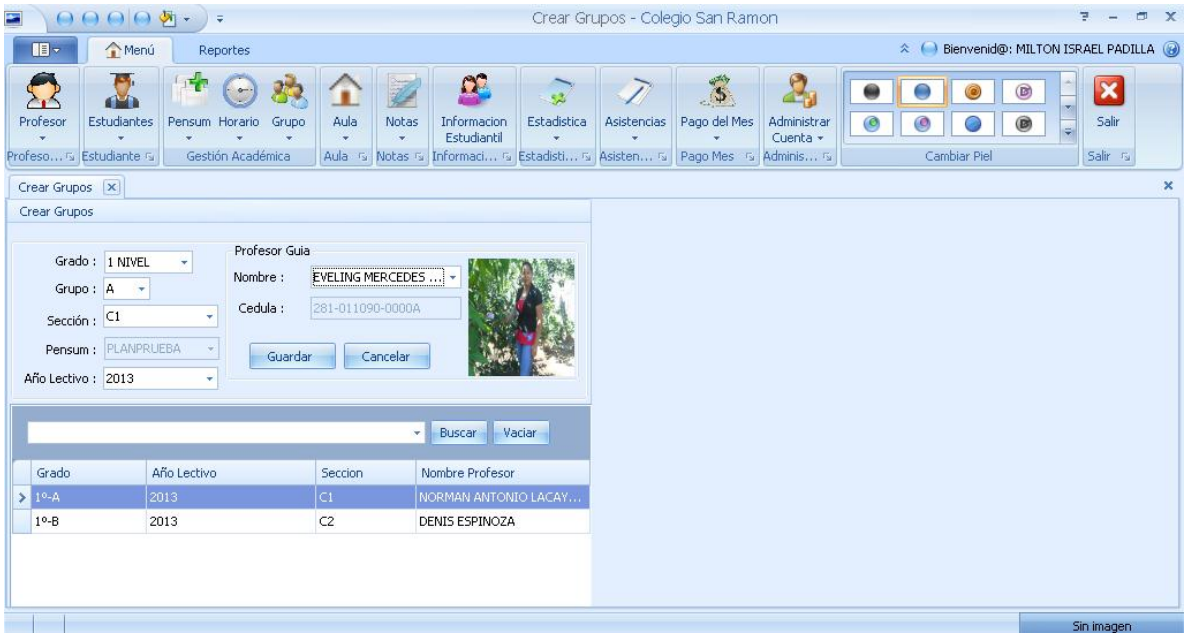


Figura 39 Interfaz de crear grupos de clases

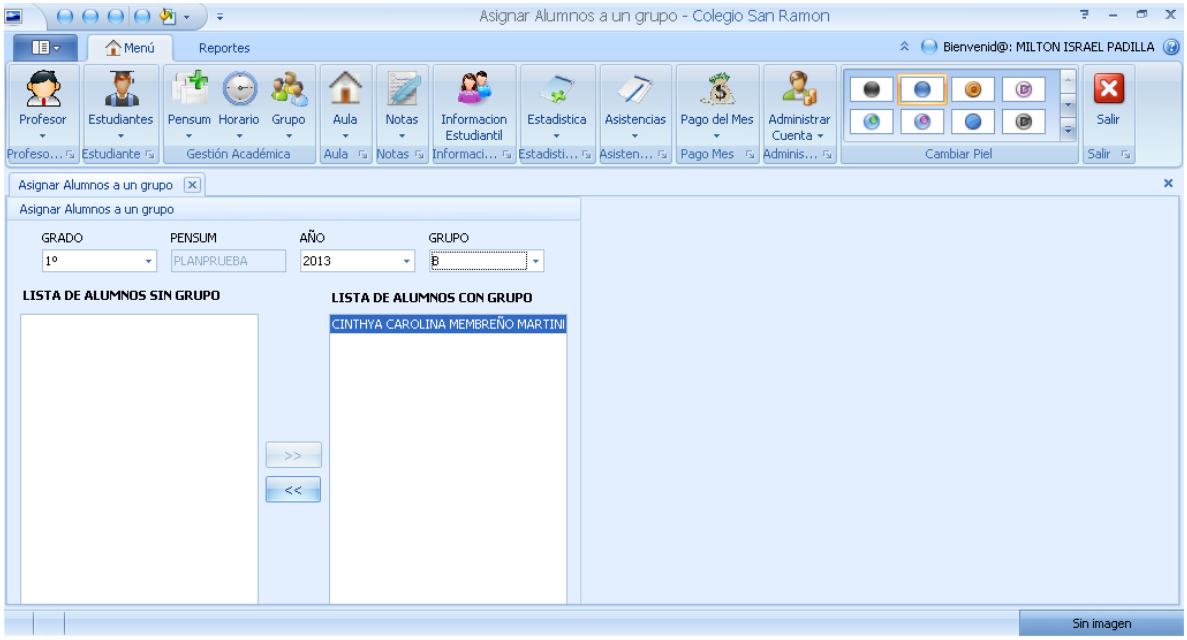


Figura 40 Interfaz de asignar alumnos a grupos de clases

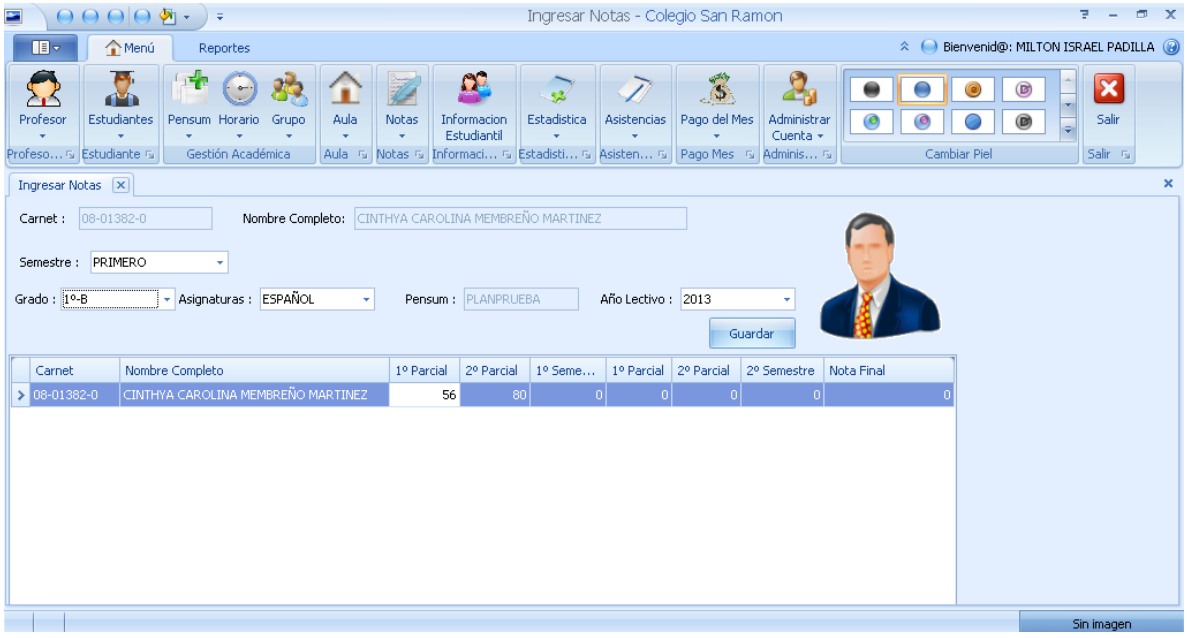


Figura 41 Interfaz de ingresar notas a los estudiantes

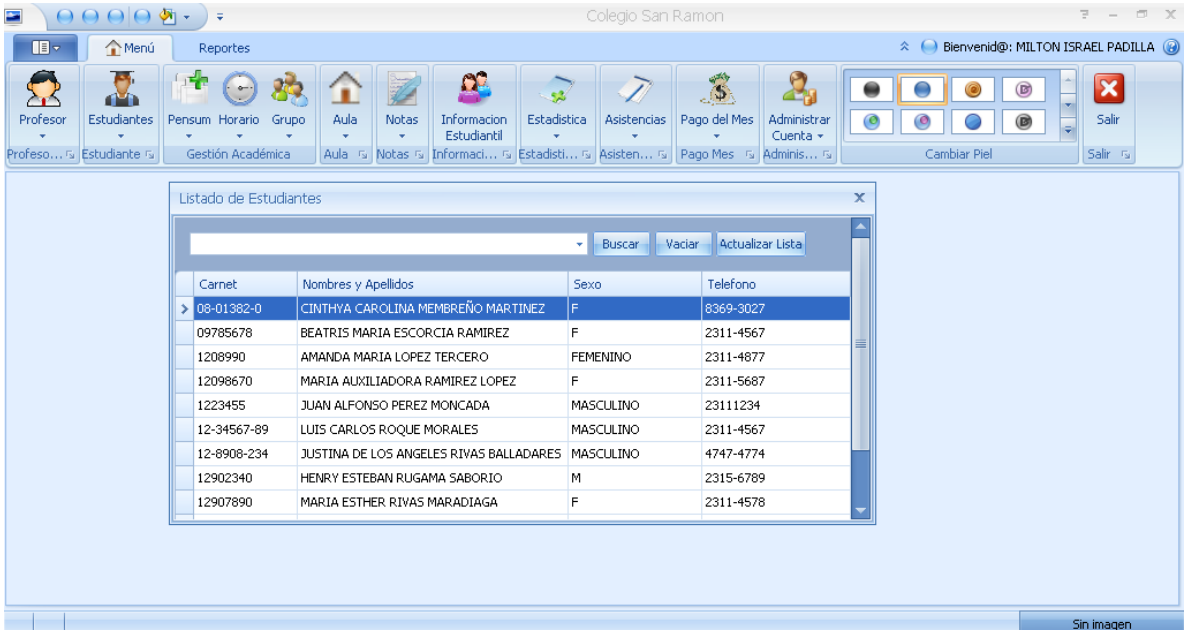


Figura 42 interfaz de información estudiantil (parte 1)

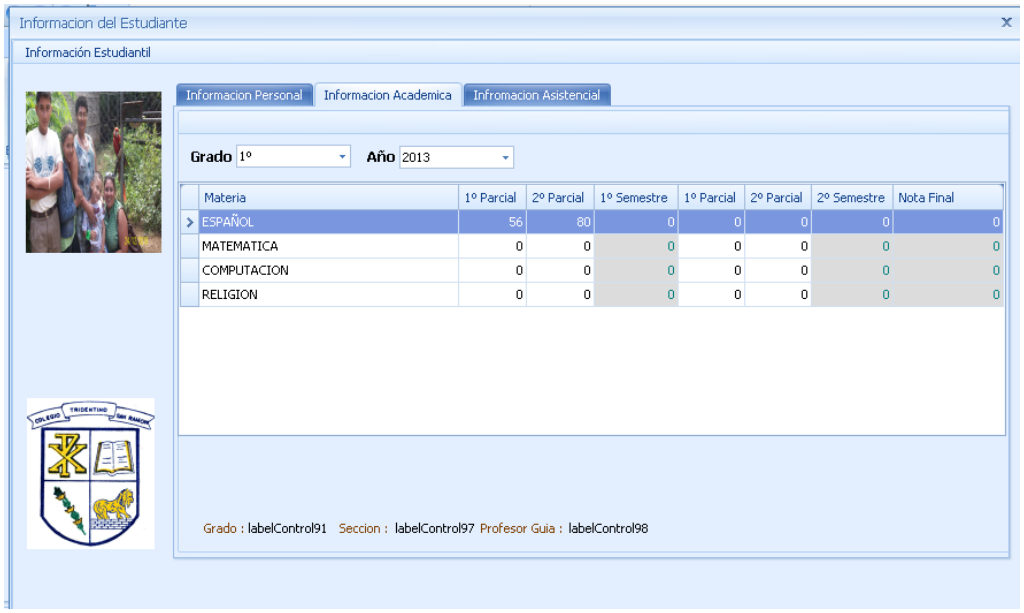


Figura 43 interfaz información estudiantil (parte 2)

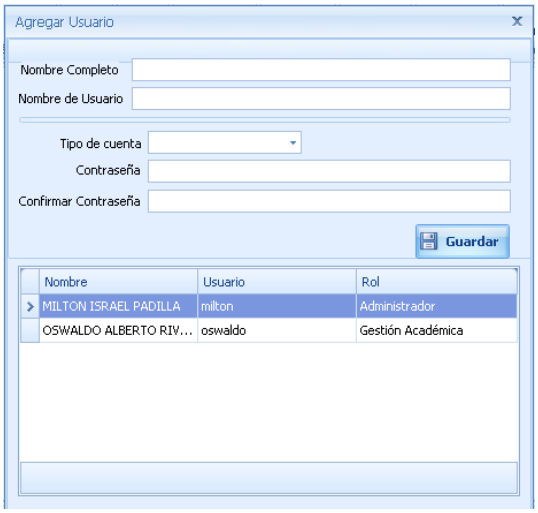


Figura 44 interfaz agregar usuario y asignar rol

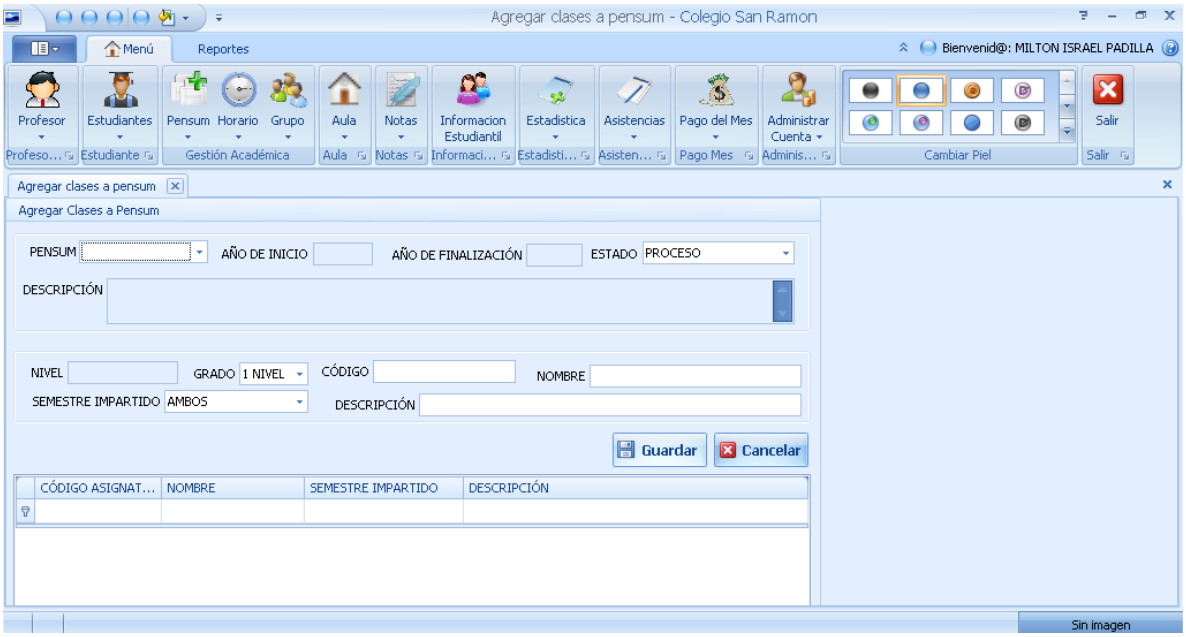


Figura 45 interfaz agregar asignaturas al pensum

Figura 46 Agregar Profesor 1 Parte

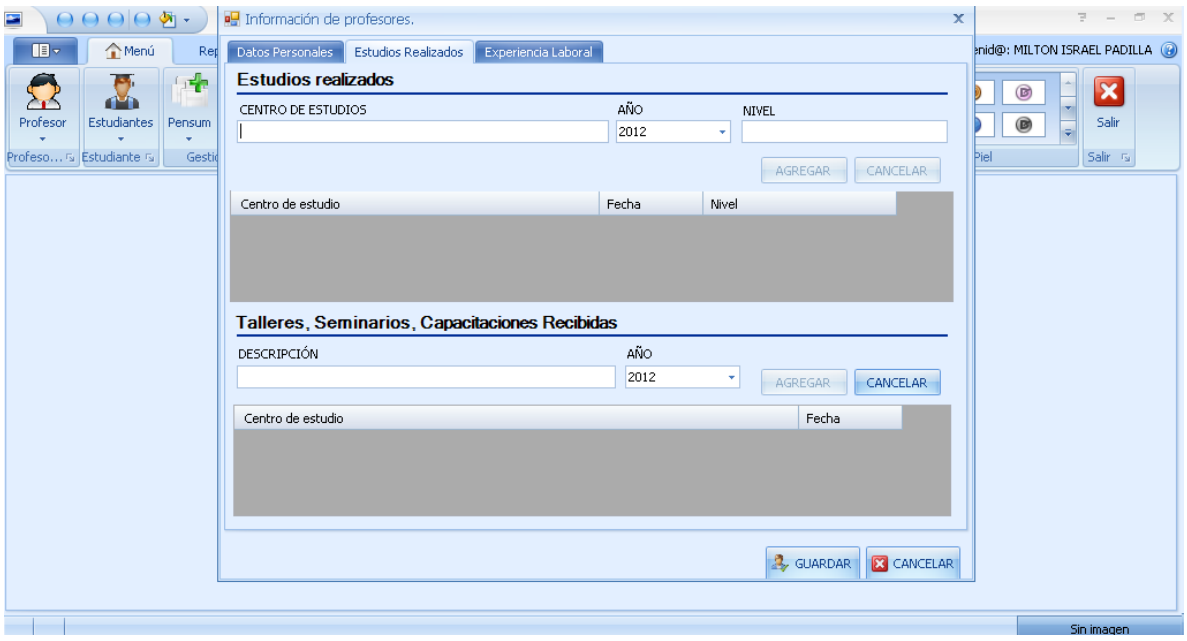


Figura 47 Agregar Profesor 2 Parte

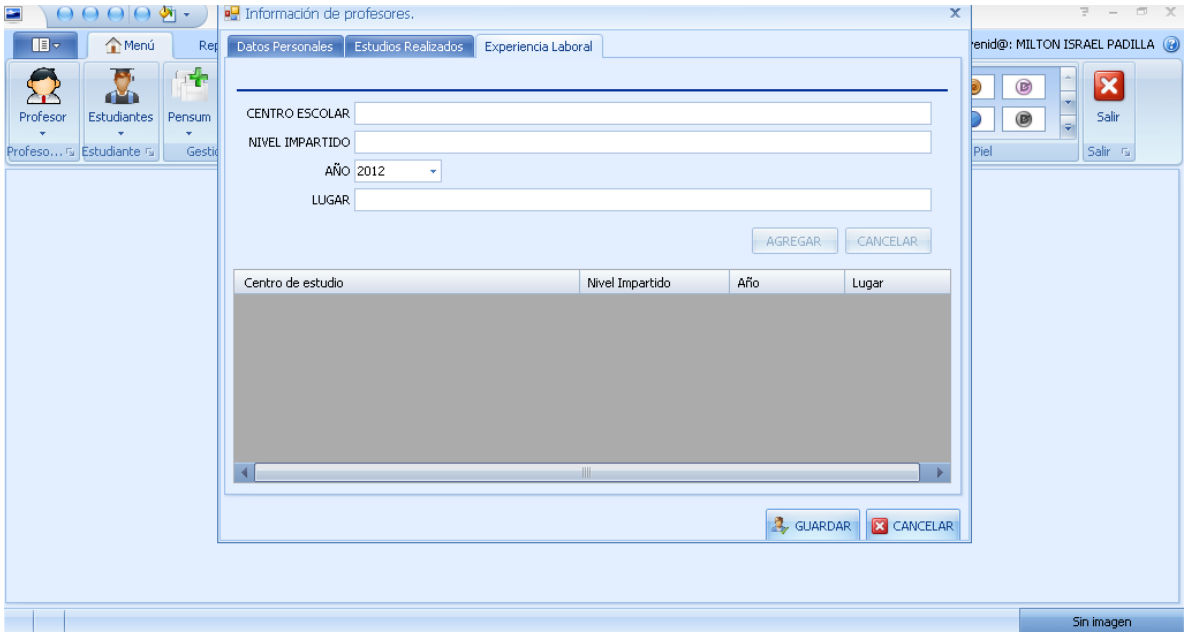


Figura 48 Agregar Profesor 3 Parte

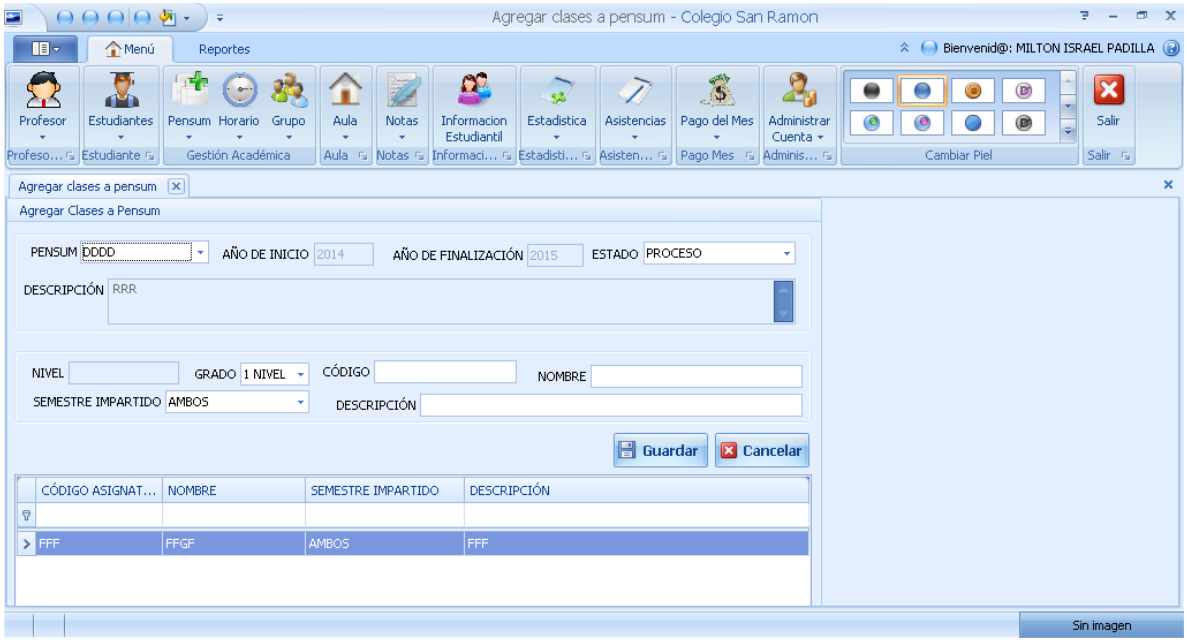


Figura 49 Agregar Pensum

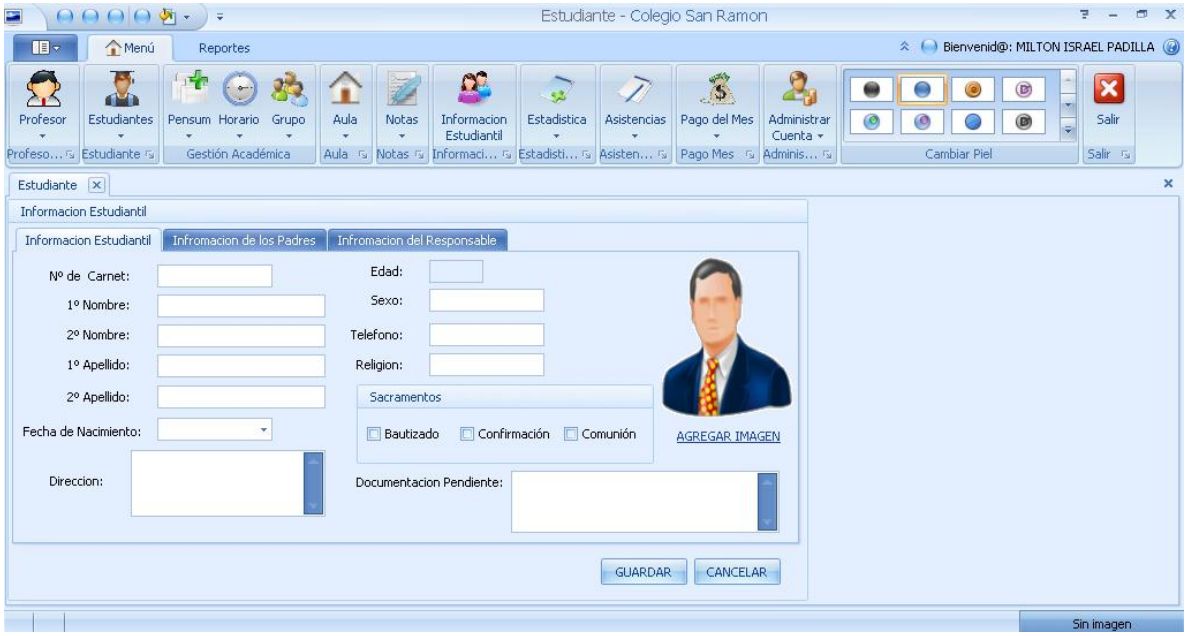


Figura 50 Agregar Estudiante

10. Requerimientos del Sistema

10.1 Requerimiento Hardware

Servidor

- Procesador Dual-Core.
- Disco Duro de 250GB.
- Memoria Ram 2GB.

Clientes

- Procesador Intel Celeron
- Disco Duro de 160GB
- Memoria Ram 2GB.

10.2 Requerimiento Software

Servidor

- Windows Server 2008.
- Microsoft SQL Server 2005.

Clientes

- Windows XP, Windows 7
- .Net Framework 3.5

11.Conclusiones

En conclusión se ha obtenido un sistema que permitirá minimizar el tiempo de respuesta para la generación de reportes y la automatización de todas las operaciones del Colegio Tridentino San Ramón Nonato a través de los módulos creados para agilizar la manipulación de los datos de todos los estudiantes, además de lograr una mayor rapidez en la creación de grupos, horarios de clase y el módulo de las notas logrando que la manipulación de la información sea fácil y rápida de utilizar.

Por ende podemos afirmar que se ha logrado alcanzar el objetivo de nuestro trabajo al proveer al Colegio Tridentino San Ramón Nonato de un software capaz de solventar los problemas que se generaban en él y a nosotros como desarrolladores nos permitió aplicar los conocimientos adquiridos a lo largo de nuestra formación como Ingenieros en Sistemas de Información.

12.Recomendaciones

Terminado nuestro trabajo consideramos los siguientes aspectos interesantes para la continuación y mejora de esta aplicación.

- Desarrollar una aplicación web para el Colegio y la revisión de las notas en línea de los estudiantes del Colegio Tridentino San Ramón Nonato.

- Realizar un blog para mantener informado a los padres de familia y estudiantes sobre las actividades del Colegio y además de sus normativas y disciplinada de cada uno de estudiantes.

- Desarrollar la implementación del moodle para un mayor acompañamiento entre el estudiante y profesor.

13.Bibliografía

Bibliografía

http://documentation.devexpress.com/. (23 de 10 de 2012). *http://documentation.devexpress.com/*.

http://es.wikipedia.org/wiki/Microsoft_.NET. (16 de 10 de 2012). *http://es.wikipedia.org/wiki/Microsoft_.NET*.

http://mspnor.wordpress.com/2008/10/31/sqlprocedimientos-almacenados-paso-a-paso/. (16 de 10 de 2012). *http://mspnor.wordpress.com/2008/10/31/sqlprocedimientos-almacenados-paso-a-paso/*.

http://www.microsoft.com/net. (16 de 10 de 2012). *http://www.microsoft.com/net*.

Microsoft. (5 de 10 de 2012). *http://msdn.microsoft.com/es-es/library/bb385795(v=vs.90).aspx*.

microsoft. (16 de 10 de 2012). *http://msdn.microsoft.com/es-es/library/ms190782(v=sql.90).aspx*.

Microsoft. (s.f.). *LINQ to ADO.NET.* Obtenido de ms-help://MS.VSCC.v90/MS.MSDNQTR.v90.es/wd_linqadonet/html/be3297b9-1b54-4d4c-82a8-add0d79c2006.htm

Microsoft, T. (5 de 10 de 2012). *http://mredison.wordpress.com/2007/12/02/caractersticas-de-visual-studio-2008/*.

Microsoft,msdn. (16 de 10 de 2012). *http://msdn.microsoft.com/en-us/library/zw4w595w(v=vs.90).aspx.* Obtenido de http://msdn.microsoft.com/en-us/library/zw4w595w(v=vs.90).aspx

Microsoft.com. (s.f.). *http://msdn.microsoft.com/es-es/library/52f3sw5c(v=vs.90).aspx*.

Ramón, C. S. (1990). *Historia Del Colegio San Ramón.* León: propio.

Server, M. S. (5 de 10 de 2012). *http://www.microsoft.com/spain/sql/productinfo/features/top30features.mspx*.

wikipedia. (5 de 10 de 2012). *http://es.wikipedia.org/wiki/Desarrollo_en_espiral*.

14.Anexos

14.1 Glosario

- **Framework:** Un framework o infraestructura digital, es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concretos, con base a la cual otro proyecto de software puede ser más fácilmente organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas, y un lenguaje interpretado, entre otras herramientas, para así ayudar a desarrollar y unir los diferentes componentes de un proyecto.
- **CLR (Common Language Runtime):**"Entorno en tiempo de ejecución de lenguaje común" es un entorno de ejecución para los códigos de los programas que corren sobre la plataforma Microsoft .NET. El CLR es el encargado de compilar una forma de código intermedio llamada Common Intermediate Language (CIL, anteriormente conocido como MSIL, por Microsoft Intermediate Language), al código de maquina nativo, mediante un compilador en tiempo de ejecución. No debe confundirse el CLR con una máquina virtual, ya que una vez que el código esta compilado, corre nativamente sin intervención de una capa de abstracción sobre el hardware subyacente.

- **Interfaz:** La **interfaz de usuario** es el medio con que el usuario puede comunicarse con una máquina, un equipo o una computadora, y comprende todos los puntos de contacto entre el usuario y el equipo. Normalmente suelen ser fáciles de entender y fáciles de accionar.

14.2 Código de Profesor

DALProfesor

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Windows.Forms;
using DevExpress.XtraEditors;
namespace sanRamon
{
publicclassDALProfesor
    {
        #region Declaracion de variables

staticSqlConnection Conexion;
staticSqlCommand Comando;
staticbool resultado;
staticSqlTransaction trans;
#endregion

// COnvierte de fila a Objeto de Negocio
publicstaticProfesor FillDataRecord1(IDataRecord myDataRecord)
            {
var profe = newProfesor();
try
                {
profe._nombre =
myDataRecord.GetString(myDataRecord.GetOrdinal("nombrecompleto"));
                    profe._numerocedula =
myDataRecord.GetString(myDataRecord.GetOrdinal("cedula"));
                    profe.foto =
(Image)cgenerica.ByteArrayToObject((byte[])myDataRecord.GetValue(myDataRe
cord.GetOrdinal("foto")));
}
catch (Exception ex)
                {
XtraMessageBox.Show(ex.Message, ConexionSQL.xTitlename,
MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
return profe;
            }

publicstaticProfesor FillDataRecord2(IDataRecord myDataRecord)
            {
var profe = newProfesor();
try
                {
profe._nombre =
myDataRecord.GetString(myDataRecord.GetOrdinal("nombrecompleto"));
profe._numerocedula =
myDataRecord.GetString(myDataRecord.GetOrdinal("cedula"));
                }
catch (Exception ex)
                {
XtraMessageBox.Show(ex.Message, ConexionSQL.xTitlename,
```

```
MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
return profe;
            }

publicstaticProfesor FillDataRecord(IDataRecord myDataRecord)
        {
var profe = newProfesor();
try
            {
                profe.Id =
myDataRecord.GetInt32(myDataRecord.GetOrdinal("Id_Profesor")).ToString();
                profe._nombre =
myDataRecord.GetString(myDataRecord.GetOrdinal("nombrecompleto"));
                profe._lugarnacimiento =
myDataRecord.GetString(myDataRecord.GetOrdinal("lugarnacimiento"));
profe._fechanacimiento =
myDataRecord.GetDateTime(myDataRecord.GetOrdinal("fechanacimiento")).ToSt
ring("dd/MM/yyyy");
                profe._nacionalidad =
myDataRecord.GetString(myDataRecord.GetOrdinal("nacionalidad"));
                profe._edad =
myDataRecord.GetInt32(myDataRecord.GetOrdinal("edad")).ToString();
                profe._sexo =
myDataRecord.GetString(myDataRecord.GetOrdinal("sexo"));
                profe._estadocivil =
myDataRecord.GetString(myDataRecord.GetOrdinal("estadocivil"));
                profe._numerohijo =
myDataRecord.GetInt32(myDataRecord.GetOrdinal("numerohijo")).ToString();
                profe._direccionexacta =
myDataRecord.GetString(myDataRecord.GetOrdinal("direccion"));
                profe._numeroinss =
myDataRecord.GetString(myDataRecord.GetOrdinal("numeroinss"));
                profe._numerocedula =
myDataRecord.GetString(myDataRecord.GetOrdinal("cedula"));
                profe._nombrepapa =
myDataRecord.GetString(myDataRecord.GetOrdinal("nombrepadre"));
                profe._nombremama =
myDataRecord.GetString(myDataRecord.GetOrdinal("nombremadre"));
                profe._nombreconyugue =
myDataRecord.GetString(myDataRecord.GetOrdinal("nombreconyugue"));
               profe.foto =
(Image)cgenerica.ByteArrayToObject((byte[])myDataRecord.GetValue(myDataRe
cord.GetOrdinal("foto")));
}
catch (Exception ex)
            {
XtraMessageBox.Show(ex.Message,ConexionSQL.xTitlename,MessageBoxButtons.O
K,MessageBoxIcon.Error);
            }
return profe;
        }

publicstaticbool Add(Profesor pro)
{

            Conexion=ConexionSQL.GetConexion();
```

```
            Conexion.Open();
trans = Conexion.BeginTransaction();
try
            {

DALProfesor.AddProfesor(pro);
               RealizarOperacion();

for (int i = 0; i < pro.listaEstudios.Count; i++)
{
                  Comando =  DALProfesorEstudios.Add(pro._numerocedula,
pro.listaEstudios,i);
RealizarOperacion();
                }
for (int i = 0; i < pro.listaExperiencia.Count; i++)
{
                  Comando =  DALProfesorExperiencia.Add(pro._numerocedula,
pro.listaExperiencia,i);
RealizarOperacion();
                }
for (int i = 0; i < pro.listaTalleres.Count; i++)
{
                  Comando =  DAlProfesorTalleres.Add(pro,i);
                  RealizarOperacion();
}
                    trans.Commit();

returntrue;
            }
catch (SqlException Ex)
            {
                trans.Rollback();
XtraMessageBox.Show(Ex.Message, ConexionSQL.xTitlename,
MessageBoxButtons.OK, MessageBoxIcon.Information);
returnfalse;
            }
finally
            {
                Conexion.Close();
            }
        }

publicstaticbool Update(Profesor pro)
{
            Conexion = ConexionSQL.GetConexion();
            Conexion.Open();
trans = Conexion.BeginTransaction();
try
            {
DALProfesor.UpdateProfesor(pro);
RealizarOperacion();
if (cgenerica.ModificarTaller)
                {
                    Comando =
DAlProfesorTalleres.Delete(pro._numerocedula);
RealizarOperacion();
```

```
for (int i = 0; i < pro.listaTalleres.Count; i++)
{
                        Comando = DAlProfesorTalleres.Add(pro, i);
                        RealizarOperacion();
                    }
                }
if (cgenerica.ModificarExperiencia)
                {
                    Comando =
DALProfesorExperiencia.Delete(pro._numerocedula);
RealizarOperacion();
for (int i = 0; i < pro.listaExperiencia.Count; i++)
{
                        Comando =
DALProfesorExperiencia.Add(pro._numerocedula, pro.listaExperiencia, i);
                        RealizarOperacion();
                    }
                }
if (cgenerica.ModificarEstudios)
                {
                    Comando =
DALProfesorEstudios.Delete(pro._numerocedula);
RealizarOperacion();
for (int i = 0; i < pro.listaEstudios.Count; i++)
{
                        Comando =
DALProfesorEstudios.Add(pro._numerocedula, pro.listaEstudios, i);
                        RealizarOperacion();
                    }
                }
                trans.Commit();
cgenerica.ModificarEstudios = false;
cgenerica.ModificarExperiencia = false;
cgenerica.ModificarTaller = false;
returntrue;
            }
catch (SqlException Ex)
            {
                trans.Rollback();
XtraMessageBox.Show(Ex.Message, ConexionSQL.xTitlename,
MessageBoxButtons.OK, MessageBoxIcon.Information);
returnfalse;
            }
finally
            {
                Conexion.Close();
            }
        }

publicstaticvoid RealizarOperacion()
        {
            Comando.Connection = Conexion;
            Comando.Transaction = trans;
            Comando.ExecuteNonQuery();
        }

publicstaticvoid AddProfesor(Profesor pro)
```

```
        {
            Comando = newSqlCommand {CommandText = "GuardarProfesor",
CommandType = CommandType.StoredProcedure };

                Comando.Parameters.Add(newSqlParameter("@Nombre",
SqlDbType.VarChar, 100, ParameterDirection.Input, true, 0, 0, "",
DataRowVersion.Current, pro._nombre));
                Comando.Parameters.Add(newSqlParameter("@LugarNac",
SqlDbType.VarChar, 80, ParameterDirection.Input, true, 0, 0, "",
DataRowVersion.Current, pro._lugarnacimiento));
                Comando.Parameters.Add(newSqlParameter("@FechaNa",
SqlDbType.DateTime, 10, ParameterDirection.Input, true, 0, 0, "",
DataRowVersion.Current, pro._fechanacimiento));
                Comando.Parameters.Add(newSqlParameter("@Nacionalidad",
SqlDbType.VarChar, 60, ParameterDirection.Input, true, 0, 0, "",
DataRowVersion.Current, pro._nacionalidad));
                Comando.Parameters.Add(newSqlParameter("@Edad",
SqlDbType.Int, 2, ParameterDirection.Input, true, 0, 0, "",
DataRowVersion.Current, pro._edad));
                Comando.Parameters.Add(newSqlParameter("@Sexo",
SqlDbType.VarChar, 9, ParameterDirection.Input, true, 0, 0, "",
DataRowVersion.Current, pro._sexo));
                Comando.Parameters.Add(newSqlParameter("@EstadoCivil",
SqlDbType.VarChar, 10, ParameterDirection.Input, true, 0, 0, "",
DataRowVersion.Current, pro._estadocivil));
                Comando.Parameters.Add(newSqlParameter("@NHijos",
SqlDbType.Int, 2, ParameterDirection.Input, true, 0, 0, "",
DataRowVersion.Current,Convert.ToUInt32(pro._numerohijo)));
                Comando.Parameters.Add(newSqlParameter("@Direccion",
SqlDbType.VarChar, 150, ParameterDirection.Input, true, 0, 0, "",
DataRowVersion.Current, pro._direccionexacta));
                Comando.Parameters.Add(newSqlParameter("@NInss",
SqlDbType.VarChar, 16, ParameterDirection.Input, true, 0, 0, "",
DataRowVersion.Current, pro._numeroinss));
                Comando.Parameters.Add(newSqlParameter("@NCedula",
SqlDbType.VarChar, 16, ParameterDirection.Input, true, 0, 0, "",
DataRowVersion.Current, pro._numerocedula));
                Comando.Parameters.Add(newSqlParameter("@NombreP",
SqlDbType.VarChar, 50, ParameterDirection.Input, true, 0, 0, "",
DataRowVersion.Current, pro._nombrepapa));
                Comando.Parameters.Add(newSqlParameter("@NombreM",
SqlDbType.VarChar, 50, ParameterDirection.Input, true, 0, 0, "",
DataRowVersion.Current, pro._nombremama));
                Comando.Parameters.Add(newSqlParameter("@NombreC",
SqlDbType.VarChar, 50, ParameterDirection.Input, true, 0, 0, "",
DataRowVersion.Current, pro._nombreconyugue));
                Comando.Parameters.Add(newSqlParameter("@foto",
SqlDbType.Image)).Value = cgenerica.ObjectToByteArray(pro.foto);
                Comando.Parameters.Add(newSqlParameter("@val",
SqlDbType.Bit)).Value = true;
        }

publicstaticvoid UpdateProfesor(Profesor pro)
        {
                Comando = newSqlCommand { Connection = Conexion,
CommandText = "ModificarProfesor", CommandType =
CommandType.StoredProcedure };
```

```
                Comando.Parameters.Add(newSqlParameter("@Id_Profesor",
SqlDbType.Int, 7, ParameterDirection.Input, true, 0, 0, "",
DataRowVersion.Current, pro.Id));
                Comando.Parameters.Add(newSqlParameter("@Nombre",
SqlDbType.VarChar, 100, ParameterDirection.Input, true, 0, 0, "",
DataRowVersion.Current, pro._nombre));
                Comando.Parameters.Add(newSqlParameter("@LugarNac",
SqlDbType.VarChar, 80, ParameterDirection.Input, true, 0, 0, "",
DataRowVersion.Current, pro._lugarnacimiento));
                Comando.Parameters.Add(newSqlParameter("@FechaNa",
SqlDbType.DateTime, 10, ParameterDirection.Input, true, 0, 0, "",
DataRowVersion.Current, pro._fechanacimiento));
                Comando.Parameters.Add(newSqlParameter("@Nacionalidad",
SqlDbType.VarChar, 60, ParameterDirection.Input, true, 0, 0, "",
DataRowVersion.Current, pro._nacionalidad));
                Comando.Parameters.Add(newSqlParameter("@Edad",
SqlDbType.Int, 2, ParameterDirection.Input, true, 0, 0, "",
DataRowVersion.Current, pro._edad));
                Comando.Parameters.Add(newSqlParameter("@Sexo",
SqlDbType.Char, 9, ParameterDirection.Input, true, 0, 0, "",
DataRowVersion.Current, pro._sexo));
                Comando.Parameters.Add(newSqlParameter("@EstadoCivil",
SqlDbType.VarChar, 10, ParameterDirection.Input, true, 0, 0, "",
DataRowVersion.Current, pro._estadocivil));
                Comando.Parameters.Add(newSqlParameter("@NHijos",
SqlDbType.Int, 2, ParameterDirection.Input, true, 0, 0, "",
DataRowVersion.Current, Convert.ToUInt32(pro._numerohijo)));
                Comando.Parameters.Add(newSqlParameter("@Direccion",
SqlDbType.VarChar, 150, ParameterDirection.Input, true, 0, 0, "",
DataRowVersion.Current, pro._direccionexacta));
                Comando.Parameters.Add(newSqlParameter("@NInss",
SqlDbType.VarChar, 16, ParameterDirection.Input, true, 0, 0, "",
DataRowVersion.Current, pro._numeroinss));
                Comando.Parameters.Add(newSqlParameter("@NCedula",
SqlDbType.VarChar, 16, ParameterDirection.Input, true, 0, 0, "",
DataRowVersion.Current, pro._numerocedula));
                Comando.Parameters.Add(newSqlParameter("@NombreP",
SqlDbType.VarChar, 50, ParameterDirection.Input, true, 0, 0, "",
DataRowVersion.Current, pro._nombrepapa));
                Comando.Parameters.Add(newSqlParameter("@NombreM",
SqlDbType.VarChar, 50, ParameterDirection.Input, true, 0, 0, "",
DataRowVersion.Current, pro._nombremama));
                Comando.Parameters.Add(newSqlParameter("@NombreC",
SqlDbType.VarChar, 50, ParameterDirection.Input, true, 0, 0, "",
DataRowVersion.Current, pro._nombreconyugue));
                Comando.Parameters.Add(newSqlParameter("@foto",
SqlDbType.Image)).Value = cgenerica.ObjectToByteArray(pro.foto);
        }

publicstaticbool Delete(string Ncedula)
        {
                Conexion = ConexionSQL.GetConexion();
                Comando = newSqlCommand { Connection = Conexion,
CommandText = "EliminarProfesor", CommandType =
CommandType.StoredProcedure };
                Comando.Parameters.Add(newSqlParameter("@NCedula",
```

```
SqlDbType.VarChar)).Value=Ncedula;
returncgenerica.RealizarOperacion(Comando);
        }

publicstaticbool RetirarProfe(string Ncedula,bool parametro)
        {
            Conexion = ConexionSQL.GetConexion();
            Comando = newSqlCommand { Connection = Conexion, CommandText
= "RetirarProfesor", CommandType = CommandType.StoredProcedure };
            Comando.Parameters.Add(newSqlParameter("@NCedula",
SqlDbType.VarChar)).Value= Ncedula;
            Comando.Parameters.Add(newSqlParameter("@valor",
SqlDbType.VarChar)).Value = parametro;
returncgenerica.RealizarOperacion(Comando);
        }

publicstaticListProfesor obtenerprofe()
        {
Profesor p;
var profe = (ListProfesor)null;
            Conexion = ConexionSQL.GetConexion();
Comando = newSqlCommand { Connection = Conexion, CommandText =
"ConsultarDatos", CommandType = CommandType.StoredProcedure };
            Conexion.Open();
var lector = Comando.ExecuteReader();
if (lector != null)
                profe = newListProfesor();

while (lector.Read())
            {
                p = FillDataRecord1(lector);
                profe.Add(p);
            }
            lector.Close();
            Conexion.Close();
return profe;
        }

publicstaticListProfesor obtenerprofeActivo()
{
Profesor p;
var profe = (ListProfesor)null;
Conexion = ConexionSQL.GetConexion();
            Comando = newSqlCommand { Connection = Conexion, CommandText
= "ConsultarDatosProfesor", CommandType = CommandType.StoredProcedure };
            Conexion.Open();
var lector = Comando.ExecuteReader();
if (lector != null)
                profe = newListProfesor();

while (lector.Read())
            {
                p = FillDataRecord2(lector);
                profe.Add(p);
            }
            lector.Close();
            Conexion.Close();
```

```
return profe;
        }

        #region Reporte
//cONSULTA PARA EL REPORTE
publicstaticListProfesor SelectProfesor(string cedula)
        {
Profesor p;
var profe = (ListProfesor)null;
            Conexion = ConexionSQL.GetConexion();
Comando = newSqlCommand { Connection = Conexion, CommandText =
"ConsultarProfesor", CommandType = CommandType.StoredProcedure };
            Comando.Parameters.Add(newSqlParameter("@cedula",
SqlDbType.VarChar, 16, ParameterDirection.Input, true, 0, 0, "",
DataRowVersion.Current, cedula));
            Conexion.Open();
var lector = Comando.ExecuteReader();
if (lector != null)
                profe = newListProfesor();

while (lector.Read())
            {
                p = FillDataRecord(lector);
p.listaTalleres =
DAlProfesorTalleres.SelectTalleresByCedula(p._numerocedula);
                p.listaEstudios =
DALProfesorEstudios.SelectProfesorByCedula(p._numerocedula);
                p.listaExperiencia =
DALProfesorExperiencia.SelectProfesorByCedula(p._numerocedula);
profe.Add(p);
            }
            lector.Close();
            Conexion.Close();
return profe;
        }
        #endregion

        #region Nueva interfaz 28/08/2012

publicstaticDataTable ObtenerProfesores(bool val)
        {
            Comando = newSqlCommand { CommandType =
CommandType.StoredProcedure, CommandText = "ConsultarProfesorprueba " };
            Comando.Parameters.Add(newSqlParameter("@val",
SqlDbType.Bit)).Value = val;
returncgenerica.ObtenerDatos(Comando);
        }
publicstaticDataTable ObtenerProfesoresActivos()
{
            Comando = newSqlCommand { CommandType =
CommandType.StoredProcedure, CommandText = "ConsultarProfesorActivo " };
returncgenerica.ObtenerDatos(Comando);
        }
publicstaticDataTable ObtenerProfesoreEstudio(string cedula)
        {
            Comando = newSqlCommand { CommandType =
CommandType.StoredProcedure, CommandText = "ConsultarProfesorEstudios" };
```

```
            Comando.Parameters.Add(newSqlParameter("@cedula",
SqlDbType.VarChar, 16, ParameterDirection.Input, true, 0, 0, "",
DataRowVersion.Current, cedula));
returncgenerica.ObtenerDatos(Comando);
        }

publicstaticDataTable ObtenerProfesoreTaller(string cedula)
        {
            Comando = newSqlCommand { CommandType =
CommandType.StoredProcedure, CommandText = "ConsultarProfesorTaller" };
            Comando.Parameters.Add(newSqlParameter("@cedula",
SqlDbType.VarChar, 16, ParameterDirection.Input, true, 0, 0, "",
DataRowVersion.Current, cedula));
returncgenerica.ObtenerDatos(Comando);
        }

publicstaticDataTable ObtenerProfesoreExperiencia(string cedula)
        {
            Comando = newSqlCommand { CommandType =
CommandType.StoredProcedure, CommandText = "ConsultarProfesorExperiencia"
};
            Comando.Parameters.Add(newSqlParameter("@cedula",
SqlDbType.VarChar, 16, ParameterDirection.Input, true, 0, 0, "",
DataRowVersion.Current, cedula));
returncgenerica.ObtenerDatos(Comando);
        }

publicstaticImage ObtenerProfesorfoto(string cedula)
        {
Image imagen = null;
            Conexion = ConexionSQL.GetConexion();
            Comando = newSqlCommand { Connection = Conexion, CommandText
= "Consultarfoto", CommandType = CommandType.StoredProcedure };
            Comando.Parameters.Add(newSqlParameter("@cedula",
SqlDbType.VarChar, 16, ParameterDirection.Input, true, 0, 0, "",
DataRowVersion.Current, cedula));
            Conexion.Open();
var lector = Comando.ExecuteReader();
if (lector != null)
while (lector.Read())
            {
                imagen =
(Image)cgenerica.ByteArrayToObject((byte[])lector.GetValue(lector.GetOrdi
nal("foto")));
            }
            lector.Close();
            Conexion.Close();

return imagen;
}
        #endregion

    }}
```

14.3 Agregar Matricua

```
publicstaticbool Add(Matricula matri,ListPensumClase clase)
        {
            Conexion = ConexionSQL.GetConexion();
            Conexion.Open();
            trans = Conexion.BeginTransaction();
try
            {
                Comando = newSqlCommand { Connection = Conexion,
CommandText = "GuardarMatricula", CommandType =
CommandType.StoredProcedure };
                Comando.Parameters.Add(newSqlParameter("@anyolectivo",
SqlDbType.VarChar, 50, ParameterDirection.Input, true, 0, 0, "",
DataRowVersion.Current, matri.año_lectivo));
                Comando.Parameters.Add(newSqlParameter("@nombrecolegio",
SqlDbType.VarChar, 50, ParameterDirection.Input, true, 0, 0, "",
DataRowVersion.Current, matri.nombre_colegio));
                Comando.Parameters.Add(newSqlParameter("@tipomatricula",
SqlDbType.VarChar, 50, ParameterDirection.Input, true, 0, 0, "",
DataRowVersion.Current, matri.tipo matricula));
                Comando.Parameters.Add(newSqlParameter("@grado",
SqlDbType.VarChar, 50, ParameterDirection.Input, true, 0, 0, "",
DataRowVersion.Current, matri.grado));
                Comando.Parameters.Add(newSqlParameter("@pensum",
SqlDbType.VarChar, 10, ParameterDirection.Input, true, 0, 0, "",
DataRowVersion.Current, matri.grupo));
                Comando.Parameters.Add(newSqlParameter("@nivelacursar",
SqlDbType.VarChar, 50, ParameterDirection.Input, true, 0, 0, "",
DataRowVersion.Current, matri.nivel_cursar));
                Comando.Parameters.Add(newSqlParameter("@carne",
SqlDbType.VarChar, 50, ParameterDirection.Input, true, 0, 0, "",
DataRowVersion.Current, matri.carnet));
RealizarOperacion();


for (int i = 0; i < clase.Count; i++)
{
                    Comando = newSqlCommand { Connection = Conexion,
CommandText = "GuardarNota", CommandType = CommandType.StoredProcedure };
                    Comando.Parameters.Add(newSqlParameter("@carnet",
SqlDbType.VarChar, 50, ParameterDirection.Input, true, 0, 0, "",
DataRowVersion.Current, matri.carnet));

Comando.Parameters.Add(newSqlParameter("@anyo_lectivo",
SqlDbType.VarChar, 50, ParameterDirection.Input, true, 0, 0, "",
DataRowVersion.Current, matri.año_lectivo));
                    Comando.Parameters.Add(newSqlParameter("@grado",
SqlDbType.VarChar, 50, ParameterDirection.Input, true, 0, 0, "",
DataRowVersion.Current, matri.grado));
                    Comando.Parameters.Add(newSqlParameter("@asignatura",
SqlDbType.VarChar, 50, ParameterDirection.Input, true, 0, 0, "",
DataRowVersion.Current,clase[i].Nombre));
                    Comando.Parameters.Add(newSqlParameter("@parcial1",
SqlDbType.VarChar, 50, ParameterDirection.Input, true, 0, 0, "",
DataRowVersion.Current, "0"));
```

```
                    Comando.Parameters.Add(newSqlParameter("@parcial2",
SqlDbType.VarChar, 50, ParameterDirection.Input, true, 0, 0, "",
DataRowVersion.Current, "0"));
                    Comando.Parameters.Add(newSqlParameter("@semestre1",
SqlDbType.VarChar, 50, ParameterDirection.Input, true, 0, 0, "",
DataRowVersion.Current, "0"));
                    Comando.Parameters.Add(newSqlParameter("@parcial3",
SqlDbType.VarChar, 50, ParameterDirection.Input, true, 0, 0, "",
DataRowVersion.Current, "0"));
                    Comando.Parameters.Add(newSqlParameter("@parcial4",
SqlDbType.VarChar, 50, ParameterDirection.Input, true, 0, 0, "",
DataRowVersion.Current, "0"));
                    Comando.Parameters.Add(newSqlParameter("@semestre2",
SqlDbType.VarChar, 50, ParameterDirection.Input, true, 0, 0, "",
DataRowVersion.Current, "0"));
                    Comando.Parameters.Add(newSqlParameter("@nota_final",
SqlDbType.VarChar, 50, ParameterDirection.Input, true, 0, 0, "",
DataRowVersion.Current, "0"));
                    RealizarOperacion();
                }
                trans.Commit();
returntrue;
            }
catch (SqlException Ex)
            {
                trans.Rollback();
XtraMessageBox.Show(Ex.Message, ConexionSQL.xTitlename,
MessageBoxButtons.OK, MessageBoxIcon.Information);
returnfalse;
            }
finally
            {
                Conexion.Close();
}
        }
```