INSTITUTO TECNOLOGICO
"CORDILLERA"

CARRERA ANÁLISIS DE SISTEMAS

"AUTOMATIZACIÓN DE LOS PROCESOS ACADÉMICOS DE INSTITUCIONES EDUCATIVAS DE NIVEL MEDIO-BACHILLERATO MEDIANTE UN SISTEMA INTEGRADO DE GESTIÓN EDUCATIVA: MÓDULO ADMINISTRACIÓN DE ESTUDIANTES"

Proyecto de investigación previo a la obtención del título de Tecnólogo en Análisis de Sistemas.

Autor: Remache Montenegro Victor Gonzalo

Tutor: Ing. Hugo Heredia

Quito, Abril 2015

Declaración de Aprobación tutor y Lector

En mi calidad de tutor del trabajo sobre el tema **"AUTOMATIZACIÓN DE LOS PROCESOS ACADÉMICOS DE INSTITUCIONES EDUCATIVAS DE NIVEL MEDIO-BACHILLERATO MEDIANTE UN SISTEMA INTEGRADO DE GESTIÓN EDUCATIVA MÓDULO ADMINISTRACIÓN DE ESTUDIANTES" ,** presentado por el ciudadano Remache Montenegro Victor Gonzalo estudiante de la escuela de análisis de sistemas, considero que dicho informe reúne los requisitos y méritos suficientes para ser sometido a la evaluación por parte del Tribunal de Grado, que el Honorable Consejo de Escuela designe ,para su correspondiente estudio y calificación.

Quito, Abril 2015

Ing. Hugo Heredia	Ing. Diana Terán
TUTOR	**LECTOR**

Contrato de Cesión sobre Derechos Propiedad Intelectual.

Comparecen a la celebración del presente contrato de cesión y transferencia de derechos de propiedad intelectual, por una parte, el estudiante REMACHE MONTENEGRO VICTOR GONZALO, por sus propios y personales derechos, a quien en lo posterior se le denominará el "CEDENTE"; y, por otra parte, el INSTITUTO SUPERIOR TECNOLÓGICO CORDILLERA, representado por su Rector el Ingeniero Ernesto Flores Córdova, a quien en lo posterior se lo denominará el "CESIONARIO". Los comparecientes son mayores de edad, domiciliados en esta ciudad de Quito Distrito Metropolitano, hábiles y capaces para contraer derechos y obligaciones, quienes acuerdan al tenor de las siguientes cláusulas:

PRIMERA: ANTECEDENTE.- a) El Cedente dentro del pensum de estudio en la carrera de análisis de sistemas que imparte el Instituto Superior Tecnológico Cordillera, y con el objeto de obtener el título de Tecnólogo en Análisis de Sistemas, el estudiante participa en el proyecto de grado denominado "AUTOMATIZACIÓN DE LOS PROCESOS ACADÉMICOS DE INSTITUCIONES EDUCATIVAS DE NIVEL MEDIO – BACHILLERATO MEDIANTE UN SISTEMA INTEGRADO DE GESTIÓN EDUCATIVA MÓDULO ADMINISTRACIÓN DE ESTUDIANTES", el cual incluye la creación y desarrollo del programa de ordenador o software, para lo cual ha implementado los conocimientos adquiridos en su calidad de alumno. b) Por iniciativa y responsabilidad del Instituto Superior Tecnológico Cordillera se desarrolla la creación del programa de ordenador, motivo por el cual se regula de forma clara la

cesión de los derechos de autor que genera la obra literaria y que es producto del proyecto de grado, el mismo que culminado es de plena aplicación técnica, administrativa y de producción.

SEGUNDA: CESIÓN Y TRANSFERENCIA.- Con el antecedente indicado, el Cedente libre y voluntariamente cede y transfiere de manera perpetua y gratuita todos los derechos patrimoniales del programa de ordenador descrito en la cláusula anterior a favor del Cesionario, sin reservarse para sí ningún privilegio especial (código fuente, código objeto, diagramas de flujo, planos, manuales de uso, etc.). El Cesionario podrá explotar el programa de ordenador por cualquier medio o procedimiento tal cual lo establece el Artículo 20 de la Ley de Propiedad Intelectual, esto es, realizar, autorizar o prohibir, entre otros: a) La reproducción del programa de ordenador por cualquier forma o procedimiento; b) La comunicación pública del software; c) La distribución pública de ejemplares o copias, la comercialización, arrendamiento o alquiler del programa de ordenador; d) Cualquier transformación o modificación del programa de ordenador; e) La protección y registro en el IEPI el programa de ordenador a nombre del Cesionario; f) Ejercer la protección jurídica del programa de ordenador; g) Los demás derechos establecidos en la Ley de Propiedad Intelectual y otros cuerpos legales que normen sobre la cesión de derechos de autor y derechos patrimoniales.

TERCERA: OBLIGACIÓN DEL CEDENTE.- El cedente no podrá transferir a ningún tercero los derechos que conforman la estructura, secuencia y organización del programa de ordenador que es objeto del presente contrato, como tampoco emplearlo o utilizarlo a título personal, ya que siempre se deberá guardar la

exclusividad del programa de ordenador a favor del Cesionario.

CUARTA: CUANTIA.- La cesión objeto del presente contrato, se realiza a título gratuito y por ende el Cesionario ni sus administradores deben cancelar valor alguno o regalías por este contrato y por los derechos que se derivan del mismo.

QUINTA: PLAZO.- La vigencia del presente contrato es indefinido.

SEXTA: DOMICILIO, JURISDICCIÓN Y COMPETENCIA.- Las partes fijan como su domicilio la ciudad de Quito. Toda controversia o diferencia derivada de éste, será resuelta directamente entre las partes y, si esto no fuere factible, se solicitará la asistencia de un Mediador del Centro de Arbitraje y Mediación de la Cámara de Comercio de Quito. En el evento que el conflicto no fuere resuelto mediante este procedimiento, en el plazo de diez días calendario desde su inicio, pudiendo prorrogarse por mutuo acuerdo este plazo, las partes someterán sus controversias a la resolución de un árbitro, que se sujetará a lo dispuesto en la Ley de Arbitraje y Mediación, al Reglamento del Centro de Arbitraje y Mediación de la Cámara de comercio de Quito, y a las siguientes normas: a) El árbitro será seleccionado conforme a lo establecido en la Ley de Arbitraje y Mediación; b) Las partes renuncian a la jurisdicción ordinaria, se obligan a acatar el laudo arbitral y se comprometen a no interponer ningún tipo de recurso en contra del laudo arbitral; c) Para la ejecución de medidas cautelares, el árbitro está facultado para solicitar el auxilio de los funcionarios públicos, judiciales, policiales y administrativos, sin que sea necesario recurrir a juez ordinario alguno; d) El procedimiento será confidencial y en derecho; e) El lugar de arbitraje serán las instalaciones del centro de arbitraje y mediación de la Cámara de Comercio de Quito; f) El idioma del arbitraje será el

español; y, g) La reconvención, caso de haberla, seguirá los mismos procedimientos antes indicados para el juicio principal.

SÉPTIMA: ACEPTACIÓN.- Las partes contratantes aceptan el contenido del presente contrato, por ser hecho en seguridad de sus respectivos intereses.

En aceptación firman a los 21 días del mes de Abril del dos mil quince.

f)_____	f)_____
C.C: 171981352-7	Instituto Tecnológico Superior Cordillera
CEDENTE	CESIONARIO

Declaración de autoría del Estudiante

Declaro que la investigación los contenidos y los resultados obtenidos en el presente proyecto, como requerimiento previo para la obtención del Título de **Tecnólogo Analista de Sistemas** es netamente original, personal, y que se han citado las fuentes correspondientes.

_____

Remache Montenegro Victor Gonzalo

C.C: 171981352-7

AGRADECIMIENTO

Gracias al todo el personal docente por todo el aporte profesional, técnico humano que ellos fueron inculcándome semestre tras semestre. A toda mi familia los cuales estuvieron ahí cuando más los necesitaba con una palabra de aliento en especial a mi tía la cual estuvo conmigo en la travesía de mi enfermedad.

Gracias

DEDICATORIA

A mis padres y hermano los cuales siempre estuvieron ahí conmigo dándome apoyo moral siempre, ya que sin ellos no hubiera podido obtener este logro de igual manera a mi tía por todo el apoyo que me brindo durante el trascurso de mi enfermedad.

Remache Montenegro Victor Gonzalo

INDICE GENERAL

ÍNDICE TABLAS

Pág.

ÍNDICE FIGURAS

Pág.

RESUMEN EJECUTIVO

El presente proyecto propone implementar un software para unidades educativas de nivel medio – bachillerato mediante el módulo de Administración de Estudiantes.

Esta propuesta se ha llevado a cabo para mejorar el sistema actual con el que cuentan las instituciones de nivel medio – bachillerato mejorando procesos de notas para los estudiantes que se lleva a cabo dentro de las instituciones.

Con la falta de herramientas tecnológicas provoca la pérdida y duplicidad de la información ya que para el personal administrativo de las instituciones realizan procesos de forma manual provocando algunos inconvenientes a la hora de realizar algunas consultas e incluso malestar de estudiantes y padres de familia.

En el presente proyecto se ha desarrollado el módulo de administración de estudiantes orientado a la web con una interfaz amigable con el personal administrativo de las instituciones, estudiantes y padres de familia, en el cual el personal administrativo podrá realizar el proceso de la matrícula de estudiantes.

El módulo de administración de estudiantes permite al personal administrativo inscribir alumnos con su representante para posteriormente realizar la matrícula permitiendo la asignación de Especialidad, Jornada, Curso, Paralelo.

ABSTRACT

The present project proposes to implement a software for educational units of average level - baccalaureate by means of the module of Students' Administration. This offer has been carried out to improve the current system with which they count the institutions of average level - baccalaureate improving processes of notes for the students who are carried out inside the institutions. With the lack of technological tools it provokes the loss and duplicity of the information since for the clerical staff of the institutions they realize processes of manual form provoking some disadvantages at the moment of realizing some consultations and enclosed discomfort of students and family parents.

In the present project there has developed the module of students' administration orientated to the web with an amicable interface with the clerical staff of the institutions, students and family parents, in which the clerical staff will be able to realize the process of the students' matriculation. The module of students' administration allows to the clerical staff to inscribe pupils with his representative later to realize the matriculation allowing the assignment of Speciality, Day, Course, and Parallel.

INTRODUCCIÓN

Con el paso de los años nuestro mundo ha ido avanzando a paso agigantados en todos los campos, pero el que campo que más ha desarrollado es el campo tecnológico y educativo como observamos diariamente ejemplos claros en cosas que nos rodean como vehículos casi inteligentes, computadoras con funciones infinitas además de rapidez en sus procesos.

Como debemos tener en cuenta la tecnología se encuentra en constante desarrollo provocando que la realización de trabajos que antes lo realizaban varias personas con la ayuda de la tecnología en nuestros tiempos lo puede hacer una sola persona ayudando así a tener una buena calidad de vida de todos.

En el campo de las ciencias informáticas y de mano de la ciencias computacionales ha formado parte del diario vivir en las unidades educativas y como un fututo profesional en este campo propuse desarrollar un sistema escolástico conformado con una estructura sólida que nos permita llevar a cabo algunas funciones que se requieren dentro de los procesos de las mismas.

Capítulo I: Antecedentes

1.01 Antecedentes

El Ministerio de Educación ha dado grandes cambios en lo que respecta al campo educativo la educación de nivel Medio-Bachillerato ha implementado nuevos parámetros para este nuevo año escolar como los detalla el A**rt. 146.** Año lectivo. El año lectivo se debe desarrollar en un régimen escolar de dos (2) quimestres en todas las instituciones educativas públicas, fiscomisionales y particulares, y debe tener una duración mínima de doscientos (200) días de asistencia obligatoria de los estudiantes para el cumplimiento de actividades educativas, constados desde el primer día de clases hasta la finalización de los exámenes del segundo quimestre.

("Documentos Legales y Normativos | Ministerio de Educación," n.d.).

Para el ingreso de los estudiantes a instituciones educativas deben cumplir con ciertos requisitos como los detalla en el **Art. 153.** Requisitos de admisión. La admisión de estudiantes a los diversos niveles educativos para establecimientos públicos, fiscomisionales y particulares se sujeta al cumplimiento de los siguientes requisitos:

Nivel de Bachillerato: presentar el certificado de aprobación de la Educación General Básica.

Las instituciones deben llevar la información de sus estudiantes en expediente de cada uno como lo estipula a continuación en el **Art. 154.** Expediente académico. Les corresponde a las instituciones educativas llevar el archivo de registro

de matrículas y promociones debidamente legalizadas.

Esta documentación constituye el expediente académico del estudiante que, en versión original, debe ser entregado al representante legal del estudiante en caso de cambio de plantel. ("Documentos Legales y Normativos | Ministerio de Educación," n.d.).

El Ministerio de Educación tuvo que dar decisiones a los problemas que se producían al iniciar un nuevo año lectivo una de las soluciones fue sectorizar a las instituciones como lo explica a continuación en el **Art. 155**. Acceso al servicio educativo público. Para el ingreso a las instituciones educativas públicas, los aspirantes deben matricularse en las instituciones educativas del sector de su residencia, en correspondencia con la sectorización implementada por la Autoridad Educativa Zonal y en cumplimiento del principio de acercar el servicio educativo al usuario, se inscribirá a todos los aspirantes y se procederá a determinar el número de aceptados de la siguiente manera:

Para el octavo grado de Educación General Básica y el primer curso de Bachillerato, en las ciudades de mayor población, por sectorización y reconocimiento de méritos académicos; además se reconocerán los méritos deportivos y/o culturales cuando fueren logrados en representación del país, según instructivos que se debe elaborar para el efecto en el Nivel Zonal.

Los alumnos que no fueren favorecidos se inscribirán en el respectivo Distrito educativo, dependencia que se encargará de ubicarlos en los establecimientos oficiales que dispongan de cupos y que, si fuere necesario, le solicitará, al Nivel Zonal, la creación de cupos adicionales.

Todas las instituciones tienen la autorización y obligación de admitir a nuevos

estudiantes como detalla el siguiente **Art. 157.** Admisión de estudiantes de otras Instituciones**.** Para la admisión de estudiantes procedentes de otras instituciones educativas, a un año que no fuere el primero del nivel, se requerirá el expediente académico de conformidad con lo señalado en el presente reglamento.

("Documentos Legales y Normativos | Ministerio de Educación," n.d.).

Después de finalizar un periodo académico y para iniciar otro, todos los estudiantes deben realizar la matrícula, tienen una determinada fecha y necesitan ciertos requisitos como los especifica el **Art. 158.** Matrícula. La matrícula es el registro mediante el cual se legaliza el ingreso y la permanencia del estudiante en un establecimiento educativo durante un año lectivo.

La matrícula del estudiante puede ser de tres (3) tipos: ordinaria, extraordinaria y excepcional.

Art. 159. Matrícula ordinaria. El período de matrícula ordinaria inicia quince (15) días antes del primer día del año lectivo y termina con el inicio del año escolar.

Art. 160. Matrícula extraordinaria. La matrícula extraordinaria será autorizada por el Rector o Director de la institución educativa, en el transcurso de los primeros treinta (30) días del año lectivo, cuando no se hubiere efectuado en el período ordinario por causas de fuerza mayor previamente justificadas.

Art. 161. Matrícula excepcional. La matrícula excepcional debe ser autorizada por el Nivel Distrital, mediante resolución administrativa, hasta noventa (90) días después de iniciado el año escolar, a los estudiantes que desearen continuar sus estudios en instituciones educativas de distinto régimen por razones de movilidad

dentro del país o que provinieren de otros países, previo cumplimiento del procedimiento respectivo en la unidad de Régimen Escolar. ("Documentos Legales y Normativos | Ministerio de Educación," n.d.).

Cuando existen estudiantes que por diversas causas deben cambiarse de ciudad y desean continuar con sus estudios necesitan nuevos requisitos para matricularse como nos detalla a continuación. **Art. 162.** Requisitos**.** Para la concesión de matrícula excepcional, los interesados deben presentar, al Nivel Distrital, la solicitud con los siguientes documentos:

- Certificados de matrícula y promoción de los años de estudio realizados.
- Aceptación de la institución educativa en la que continuará sus estudios
- Informes y convenios, si los hubiere, en el caso de estudiantes que provengan del exterior.

1.02 Justificación.

El precipitado incremento de los alumnos, en las instituciones de Educación Media-Bachillerato, es un fenómeno que exige un trabajo reunido de las distintas áreas del conocimiento, para poder aplicar soluciones reales y prácticas a las distintas debilidades que se pudieran presentar como fenómenos o problemas propios e esenciales al desarrollo de las mismas. Así, el aporte de la informática y la computación, va dirigido a ofrecer distintas alternativas para la simplificación laboral y mejora de la eficiencia institucional, por ello se propone la creación de un sistema escolástico que será diseñado con parámetros de fácil aplicación y manipulación, mismo que para las instituciones de Educación Media- Bachillerato,

será una herramienta que facilitara sus actividades, sin tener que preocuparse por hacer cálculos o tener la información dispuesta en papel, sino más bien de una forma sencilla y eficaz. Además, ofrece también un punto de fácil acceso a datos como las calificaciones, informes de conducta, control de asistencia de estudiantes y docentes, entre otros, con descripciones completas de los mismos, como son sus nombres, edades, dirección, nivel académico, estadísticas de rendimiento tanto individuales como colectivo, entre otros, mejorando el nivel de control sobre las actividades que se realizan, lo que elevara el nivel de competitividad tanto interno como interinstitucional. Considerando también que a través de dicho sistema informático se puedan establecer soluciones rápidas y efectivas a las necesidades de los usuarios tantos padres de familia. Como personal administrativo y docente de estas instituciones, y como un valor agregado

también se logrará mejorar el entorno laboral e imagen pública de las instituciones, tornándose más eficiente, además de convertirse en un excelente punto de enlace con la realidad social que rodea a la institución. Con estos antecedentes, queda plenamente justificada la necesidad que tienen las instituciones de Educación Media-Bachillerato, de poseer un sistema escolástico acorde con los avances tecnológicos, desarrollado con programas y lenguajes computacionales, que permitan un tratamiento seguro y confiable de la información de fácil manejo, y flexible a los requerimientos de sus usuarios.

1.03 Definición del Problema.

Tabla 1

Análisis de matriz de Fuerza T donde se detalla situaciones.

PROBLEMÁTICA AGRAVADA	SITUACIÓN ACTUAL				SITUACIÓN MEJORADA
Cierre de instituciones por deficiencia en el proceso de Admisión y matriculación de sus Estudiantes.	Procesos inadecuados de la gestión del Módulo Administración de Estudiantes.				Automatizar el Moduló Administración Estudiantes de nivel Medio-Bachillerato
Fuerzas Impulsadoras	**I**	**P C**	**I**	**P C**	**Fuerzas Bloqueadoras**
Promocionar la oferta académica de las instituciones particulares.	**4**	**3**	**3**	**2**	Remuneraciones elevadas en pago referente a pensiones en la institución.
Mejorar los procesos de admisión y matriculación en las instituciones particulares.	**5**	**4**	**4**	**3**	Información deficiente acerca del proceso de admisión de las instituciones particulares.
Realizar la matriculación mediante la Web para evitar realizar extensas colas en dicho proceso.	**4**	**5**	**3**	**4**	Las instituciones particulares no se rigen de cierta forma a la implementación de sistemas en el proceso de admisión y matriculación.
Conocer de forma detallada los cambios realizados para instituciones particulares en la admisión y matriculación de estudiantes nuevos y antiguos.	**3**	**4**	**4**	**5**	Los estudiantes nuevos no cumplen con los requisitos impuestos por el Régimen Escolar no podrán ser admitidos en las instituciones.

Fuente: Estudio de campo

Nota: Potencial de Cambio

I: Intensidad

1.01 Análisis de la Matriz de Fuerza T.

Luego de realizar un análisis profundo de Fuerza en esta Matriz, podemos dar a conocer que la intensidad y el potencial de cambio son de vital importancia implementar el sistema Web para llevar un control adecuado en el proceso académico.

Los usuarios tendrán la satisfacción al momento de conocer su información en el momento que dispongan y en lugar que se encuentren, dado a que cada usuario manejará un perfil según su rol.

Adicionalmente ayudara a mantener cualquier tipo de registro actualizado, consistente y sobre todo integro, ya que toda información que se ingrese en el sistema será actualizada diariamente.

Capítulo II: Análisis de Involucrados

2.01 Requerimientos.

2.01.01 Descripción del Sistema Actual.

Administración de Estudiantes: Este proceso se lo realiza en Instituciones por el personal administrativo mediante el cual se realiza la admisión y la matriculación de los estudiantes antiguos y de los nuevos estudiantes que quieran ingresar a dicha institución para empezar un nuevo año lectivo, la información de cada estudiante se la guarda en un expediente manualmente el cual servirá de respaldo de la institución ante cualquier rendición de cuentas ante el Ministerio de Educación.

En las ciertas Instituciones de Nivel Medio-Bachillerato aún no se ha implementado un software que automatice sus procesos, en lo que respecta a la matriculación y asignación de profesores a cada grado lo manejan manualmente y en el mejor de los casos en Excel, esto dificulta al intentar obtener cualquier tipo de información requerida, la creación de reportes se lo hace de manera manual y por ende el docente tarda demasiado.

Otro de los puntos importantes que se ha tomado en cuenta en el levantamiento de requerimientos es que en la mayoría de las matriculas el personal administrativo la falta de documentación y se ha apreciado en el sistema en llevar el control de la documentación de cada alumno, de esta forma el administrador del sistema ingresara al perfil de un alumno determinado y podrá constatar que tipo de documento le falta presentar para legalizar su estadía en la institución, lo que actualmente lo hacen de carpeta en carpeta.

2.01.02 Visión

El propósito del desarrollo de este sistema se basa en automatizar y por ende agilitar los procesos manuales que llevan actualmente en su gran mayoría.

La visión es fortalecer a la institución y personas involucradas en el sistema haciendo que desarrollen de mejor manera sus capacidades a través de las herramientas que nos ofrecen las Nuevas Tecnologías para el mejoramiento del rendimiento institucional y personal.

2.02.03 Alcance.

Satisfacer las necesidades de instituciones de nivel Medio-Bachillerato en los ámbitos de matriculación que describimos a continuación:

2.01.02.01 Registro

Esto implica tener un registrar la documentación que el estudiante presenta, además de ello podemos registrar a estudiantes, representantes, docentes, curso, materias, paralelos, especialidades y jornadas.

2.01.02.02 Matriculación

En este proceso el estudiante previamente ya está inscrito en el sistema con toda su documentación y vamos a proceder a matricularle en donde se le busca al estudiante inscrito por filtro que en este caso será el " Número de Cédula" para posteriormente asignarle la jornada en la que va estudiar, especialidad que seguirá, curso, paralelo, y materias que tomara en ese nivel.

2.01.03 Entrevista.

Tabla 2

Detalle de la Entrevista.

DISEÑO ENTREVISTA		
Identificador: 001		
Preguntas	**Objetivos**	**Análisis posterior**
¿Quiénes tendrán acceso al sistema escolástico?	Obtener el listado de los usuarios que manejarán el sistema escolástico	Se requiere que las personas que tendrán acceso a la información sean:
		Rectorado.
		Secretaria.
		Profesores.
		Padres de familia.
		Alumnos.
¿Qué documentos solicitan al representante para matricular un alumno?	Obtener el listado de documentos/requisitos para matricula	Las documentos que se solicitan como requisitos son:
		Libreta de calificaciones de la Escuela.
		Pase de nivel
		Copia de la cedula del alumno
		Fotos tamaño carnet
		Certificado Médico
¿Con qué sistema o aplicaciones trabajan hoy en día?	Enumerar los sistemas/aplicaciones que utilizan para su trabajo diario	Los sistemas/aplicaciones con las que laboran actualmente son:
		Excel
		Word
		PowerPoint
¿Qué recomienda que se deba mejorar en el proceso?	Determinar los puntos más críticos que actualmente manejan en la institución	Necesitan mejorar en la estandarización de documentos
		Necesitan mejorar en agilidad para presentar informes en general
		Necesitan mejorar el control del proceso de alumnos matriculados
		Necesitan mejorar el control del proceso de estudiantes inscritos.

2.01.04 Matriz de Requerimiento.

Tabla 3

Detalle de requerimientos Funcionales y no Funcionales.

Identificador	Descripción	Fuente	Prioridad	Tipo	Estado	Usuarios Involucrados
REQUERIMIENTOS FUNCIONALES						
RF001	Automatizar los procesos que se realiza manualmente en los registros de estudiantes	Rector	Alta	Software	En ejecución	Administrador Docentes Secretaria
RF002	Registro en la admisión de estudiantes.	Secretaría	Alta	Software	En ejecución	Secretaria
RF003	Entregar listados de estudiantes matriculados correctamente.	Secretaria	Alta	Usuario	En ejecución	Secretaría
REQUERIMIENTOS NO FUNCIONALES						
RNF001	Las personas que tendrán acceso al sistema	Secretaria	Media	Usuario	En ejecución	Administrador Secretaria Docentes Alumnos
RNF002	El sistema puede ejecutarse en cualquier navegador web.	Secretaria	Media	Usuario	En ejecución	Secretaria Docentes Alumnos

2.01.05 Descripción Detallada.

Tabla 4

Detalle de requerimiento Funcional RF001.

Automatizar los procesos que se realiza manualmente en los registros de estudiantes		**Estado**	**En ejecución**
Creado por	Victor Remache	**Actualizado por**	Victor Remache
Fecha Creación	25/12/2014	**Fecha de actualización**	25/12/2014
Identificador	RF 001		
Tipo de requerimiento	Crítico	**Tipo de requerimiento**	Funcional
Datos de entrada	Se registra a todos los registro de actividades de los estudiantes en el sistema el cual provoca duplicidad de la información		
Descripción	Se ingresa la documentación la cual se ira almacenando en una base de datos.		
Datos de salida	Información correctamente validada se emite para registrar a los alumno de las actividades que realizan diariamente.		
Resultados esperados	Procesos automatizados listos para ser ejecutados en el momento que se los requiera.		
Origen	Secretaria		
Dirigido a	Secretaria Docentes Alumnos		
Prioridad	Alta		
Requerimientos asociados	Ninguno		
ESPECIFICACIONES			
Precondiciones	Entrevista realizada para verificar fallos en el proceso.		
Poscondiciones	Ninguno		
Criterios de aceptación	Ninguno		

Tabla 5

Detalle de requerimiento Funcional RF002.

Registro en la admisión de estudiantes.		**Estado**	**En ejecución**
Creado por	Victor Remache	**Actualizado por**	Victor Remache
Fecha Creación	25/12/2014	**Fecha de actualización**	25/12/2014
Identificador	RF 002		
Tipo de requerimiento	Critico	**Tipo de requerimiento**	Funcional
Datos de entrada	Listado de estudiantes registrados.		
Descripción	Después de ser admitidos los estudiantes se procederá a la muestra de los listados		
Datos de salida	Registro correcto de los estudiantes para el nuevo periodo académico.		
Resultados esperados	Cada estudiante será registrado correctamente en el sistema con datos reales.		
Origen	Secretaria		
Dirigido a	Docente Alumno		
Prioridad	Alta		
Requerimientos asociados	Ninguno		
ESPECIFICACIONES			
Precondiciones	Ninguno		
Poscondiciones	Ninguno		
Criterios de aceptación	Los estudiantes estarán respaldados con sus expedientes correctamente registrados.		

Tabla 6

Detalle de requerimiento Funcional RF003.

Entregar listados de estudiantes matriculados correctamente.		**Estado**	**En ejecución**
Creado por	Victor Remache	**Actualizado por**	Victor Remache
Fecha Creación	25/12/2014	**Fecha de actualización**	25/12/2014
Identificador	RF 003		
Tipo de requerimiento	Critico	**Tipo de requerimiento**	Funcional
Datos de entrada	Información de listados de alumnos matriculados.		
Descripción	Con la información se podrá dar inicio de clases como tal del nuevo año escolar.		
Datos de salida	Reporte de listados de todos estudiantes registrados		
Resultados esperados	Se obtiene los resultados, después de haber mostrado el listado de todos los estudiantes		
Origen	Secretaria		
Dirigido a	Alumnos Docentes		
Prioridad	Alta		
Requerimientos asociados	Ninguno		
ESPECIFICACIONES			
Precondiciones	Registro de estudiantes matriculados.		
Poscondiciones	Ninguno		
Criterios de aceptación	Una vez matriculados los estudiantes, se generan reporte con los estudiantes ingresados al sistema.		

Tabla 7

Detalle de requerimiento no Funcional RF001.

Las personas que tendrán acceso al sistema		**Estado**	**En ejecución**
Creado por	Victor Remache	**Actualizado por**	Victor Remache
Fecha Creación	25/12/2014	**Fecha de actualización**	25/12/2014
Identificador	RF 001		
Tipo de requerimiento	Critico	**Tipo de requerimiento**	Funcional
Datos de entrada	Todos los usuario tendrán su Usuario y Contraseña con los cuales podrán ingresar		
Descripción	Los usuarios deberán ingresar sus datos para validar, después de eso deberán buscar lo que necesitan realizar cada usuario.		
Datos de salida	Reporte de Documentos.		
Resultados esperados	Los resultados esperados son que puedan encontrar de manera rápida y eficiente la información.		
Origen	Secretaria		
Dirigido a	Docente Alumno		
Prioridad	Alta		
Requerimientos asociados	Ninguno		
ESPECIFICACIONES			
Precondiciones	Ninguno		
Poscondiciones	Ninguno		
Criterios de aceptación	Los estudiantes tendrán sus credenciales para ingresar al sistema.		

Tabla 8

Detalle de requerimiento no Funcional RF002.

El sistema puede ejecutarse en cualquier navegador web.		**Estado**	**En ejecución**
Creado por	Victor Remache	**Actualizado por**	Victor Remache
Fecha Creación	25/12/2014	**Fecha de actualización**	25/12/2014
Identificador	RF 002		
Tipo de requerimiento	Critico	**Tipo de requerimiento**	Funcional
Datos de entrada	La aplicación deberá tener la facilidad de ejecutarse en cualquier tipo de navegadores.		
Descripción	La aplicación tendrá la facilidad de ejecución en cualquier entorno web.		
Datos de salida	Interfaz del sistema		
Resultados esperados	El uso del sistema será de manera amigable		
Origen	Secretaria		
Dirigido a	Docente Alumno		
Prioridad	Alta		
Requerimientos asociados	Ninguno		
ESPECIFICACIONES			
Precondiciones	Ninguno		
Poscondiciones	Ninguno		
Criterios de aceptación	Los usuarios podrán navegar en cualquier de manera fácil y sencilla		

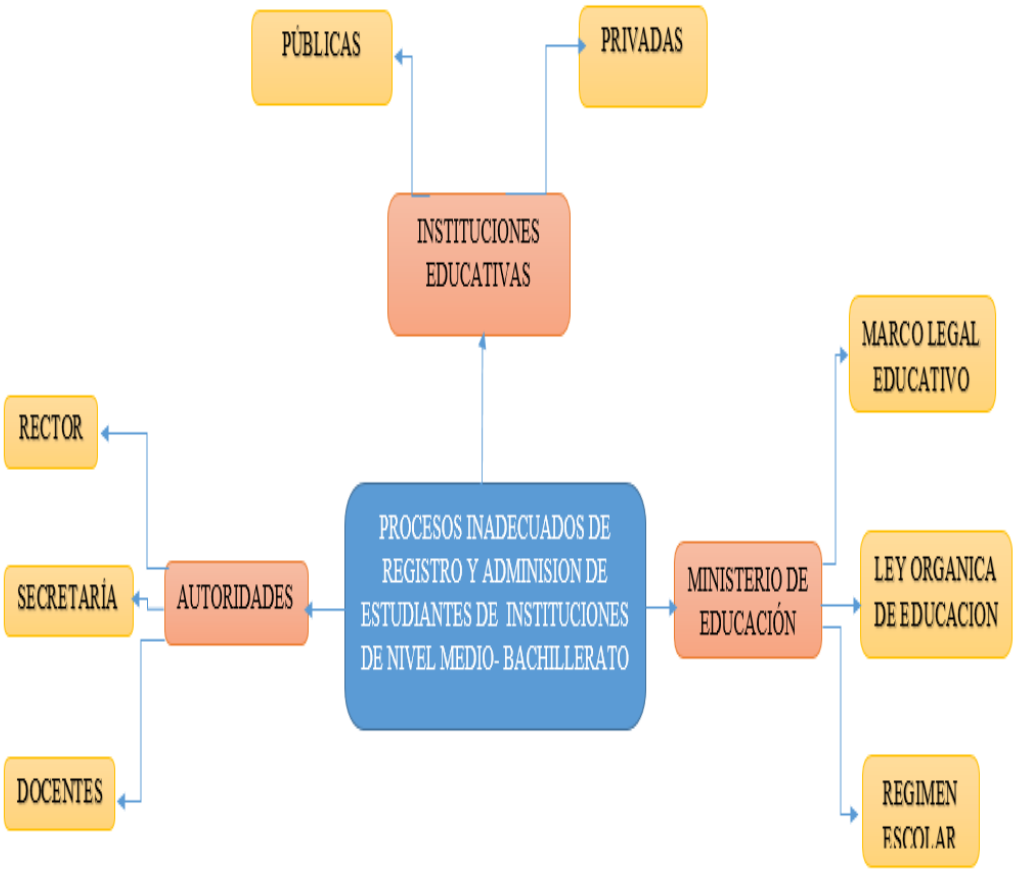


Figura 1 *Detalle al Mapeo de Involucrados con sus relaciones. En esta figura podemos identificar a todas aquellas personas involucrados en cierto proceso, además podemos establecer quienes están afectadas por los objetivos de mismo.*

2.03 Matriz de Análisis de Involucrados.

Tabla 9

Detalle de la Matriz de Análisis de Involucrados

ACTORES INVOLUCRADOS	INTERES SOBRE EL PROBLEMA CENTRAL	PROBLEMAS PERCIBIDOS	RECURSOS, MANDATOS Y CAPACIDADES	INTERESES SOBRE EL PROYECTO	CONFLICTOS POTENCIALES
Autoridades	Promover la calidad total en la ejecución de procedimientos	Autoridades sin conocimiento actualizados	Mejorando el compromiso de las autoridades se obtendrán un incremento favorable dentro de las instituciones.	Liderar el sector de la educación e incrementar la calidad de educación.	Despreocupación por parte de las autoridades para la conducción para mejorar el sistema académico.
Instituciones Educativas	Fortalecimiento de los procesos en los sistemas académicos	Carencia en los procesos de registro de estudiantes	Brindar un excelente servicio a los padres de familia	Satisfacer el servicio en la admisión de estudiantes a las instituciones	Ausencia de involucramiento en el proceso de registro de estudiantes
Ministerio de Educación	Automatizar los procesos del módulo de administración de estudiantes para las instituciones académicas	Sanción a instituciones por fallos en la implementación de sistemas académicos	Reglamentos y normativas regidas por la Ley orgánica de educación	Realizar las admisiones de forma rápido para evitar largas colas en las instituciones	Falta de colaboración con la implementación de las nuevas leyes de la educación

Capitulo III: Problemas y Objetivos

3.01. Árbol de Problemas.

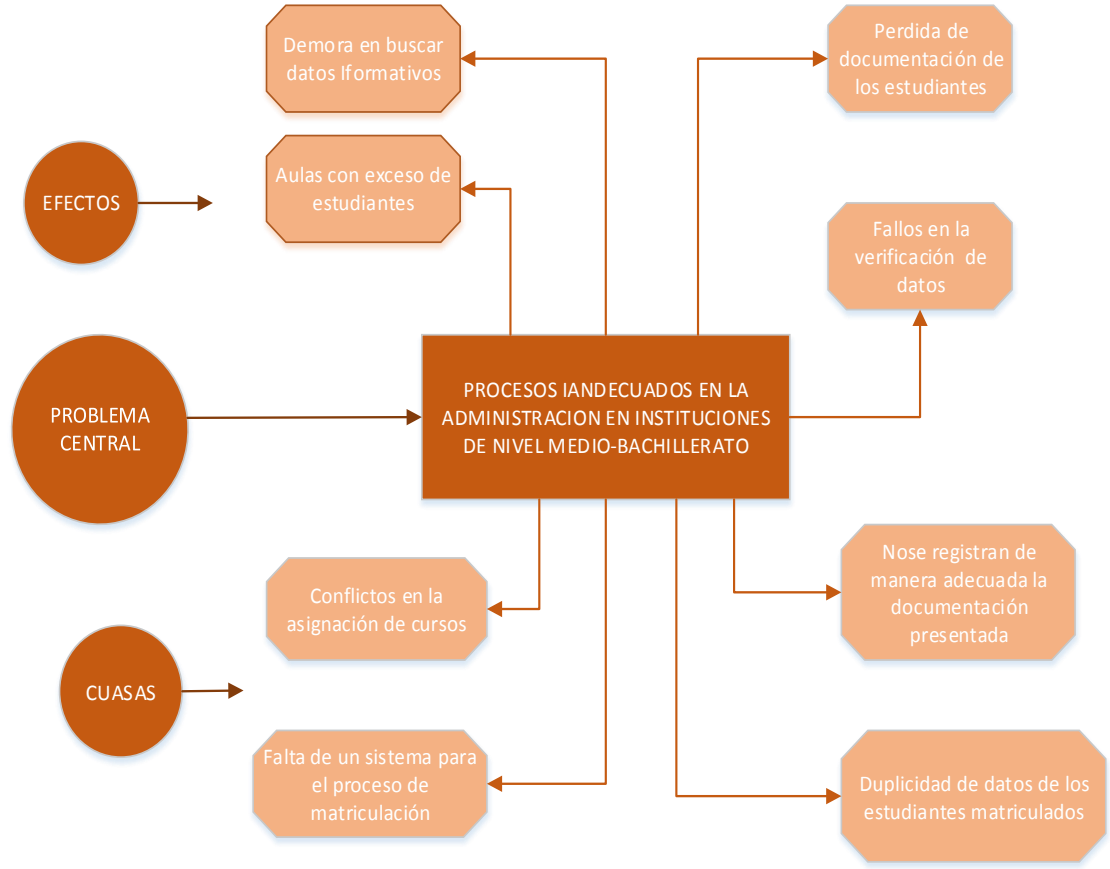


Figura 2 *Detalle del árbol de problema causales.*

Análisis: Para empezar con el análisis de este árbol de problemas empezamos planteando que los procesos es inadecuado en las instituciones provocando que se generen conflictos en la asignación de cursos, no se generen de manera adecuada la documentación además teniendo que estar verificando sino se repite la información provocando esto una demora en el procesos de matriculación, provocando esto que las consultas de la información de los estudiantes sea demorada, siendo esto una molestia tanto para los padres de familia como sus estudiantes.

3.02. Árbol de Objetivos.

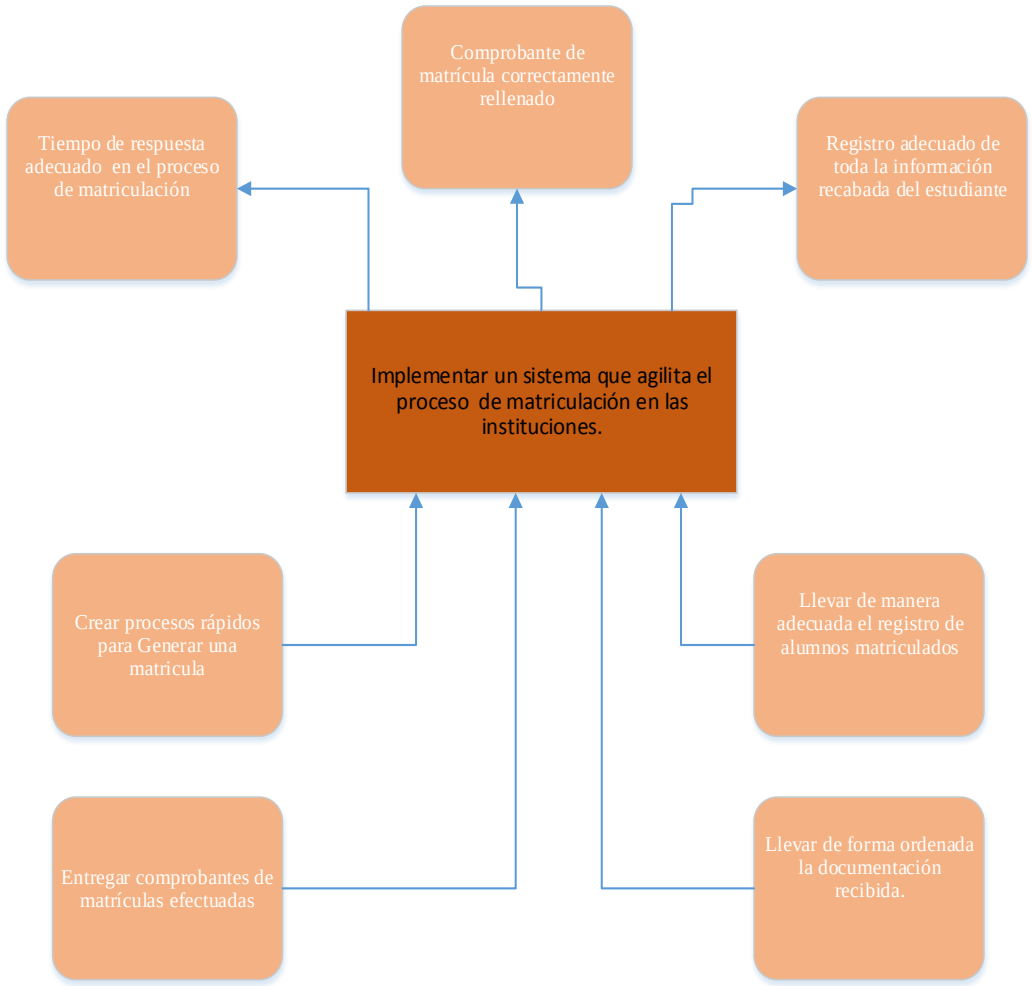


Figura 3 *Detalle del árbol de objetivos.*

Análisis: El árbol de objetivos es la versión positiva del árbol de problemas en cual el objetico central es implementar el sistema para dar soluciones rápidas y eficientes a los problemas que se generaron anteriormente permitiendo así matricular de forma rápida de igual manera puedan entregar los respectivos comprobantes y de igual manera con este implementación el tiempo de respuesta de las consulta de datos de los estudiantes sean rápidas y confiables.

3.03. **Diagrama de Casos de Uso.**

El diagrama de casos de uso representa la forma en como un Secretaría (Actor) opera con el sistema en desarrollo, además de la forma, tipo y orden en como los elementos interactúan (operaciones o casos de uso).

Este diagrama es el principal el cual nos indica el procesos que debe seguir los estudiantes para matricularse sin ningún problema previamente cumpliendo con todo los requisitos de la unidad educativa.

Véase Anexo A.01.

3.03.01 Diagrama Caso de uso de Documentación.

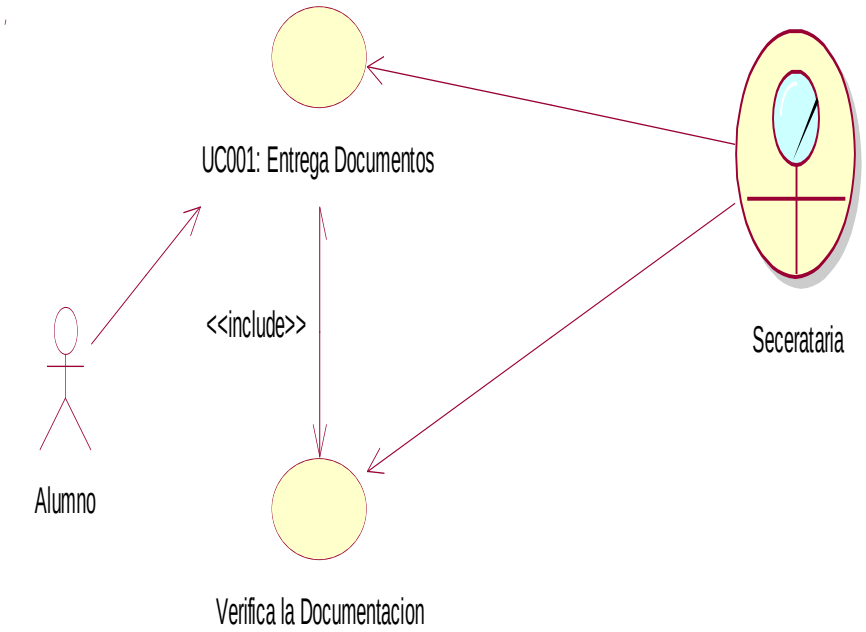


Figura 4. *Caso de uso de Documentación CU001. En este caso de uso el estudiante presenta la documentación completa la secretaria de la institución.*

3.03.02 Diagrama Caso de uso de Admisión.

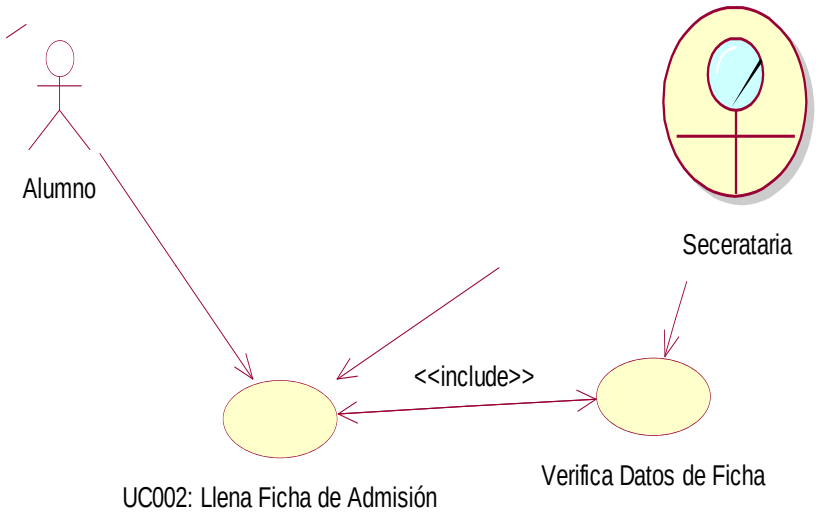


Figura 5. *Caso de uso de Admisión CU002. En este caso de uso el alumno realiza el llenado de un formulario para la inscripción a la institución, después le entrega la secretaria y ella verifica los datos estén correcto.*

3.03.03 Diagrama Caso de uso de Matriculación.

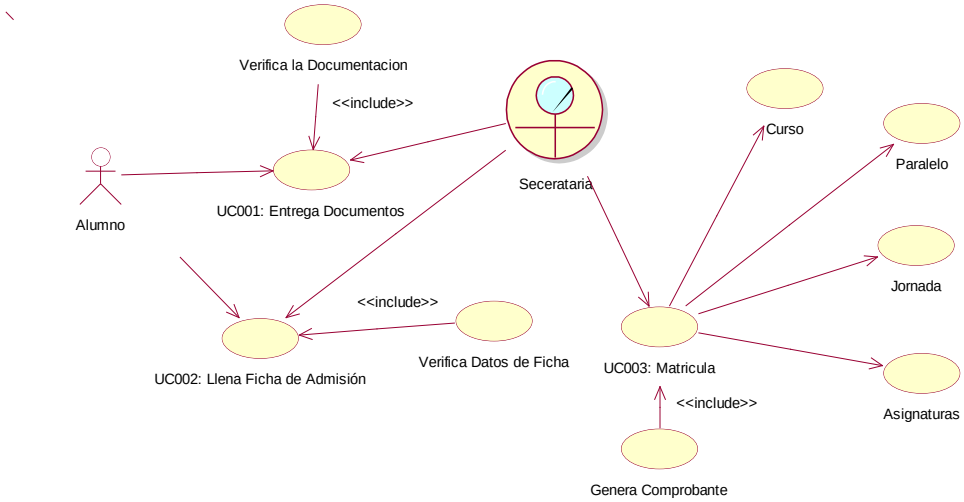


Figura 6 *Caso de uso de Contexto Matriculación CU003. En este caso de uso la secretaria realiza la matrícula de los estudiantes previamente inscritos en la institución.*

3.04 Casos de Uso de Realización.

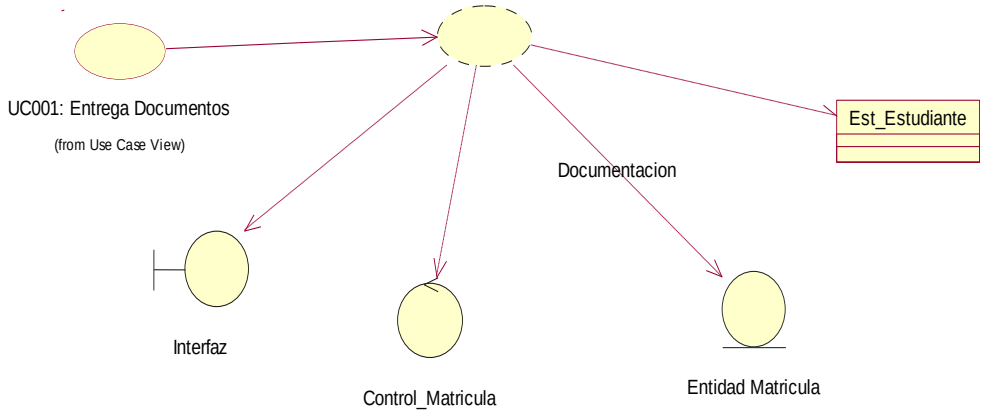


Figura 7. *Caso de uso de Realización Matricula CUR001. En este caso de realización la secretaria ingresa la documentación al sistema y almacena en la tabla de estudiantes.*

Tabla 10

Especificación del Caso Realización Documentación.

Nombre	**Documentación**
Identificador	UCR001
Responsabilidades	Registrar toda la documentación presentada del alumno.
Tipo	Sistema
Referencias de Casos de Uso	UC001
Referencias Requisitos	FR001
PRECONDICIONES	
Instancia Crea registros de la documentación presentada la secretaría: **Relación** - Dirigirse a la interfaz del Estudiante**.**	
POSCONDICIONES	
Instancia Llena en observaciones la documentación presentada. Relación. La tabla estudiantes se utiliza las observaciones.	
SALIDAS PANTALLAS	
Interfaz de usuario registra los Documentos.	

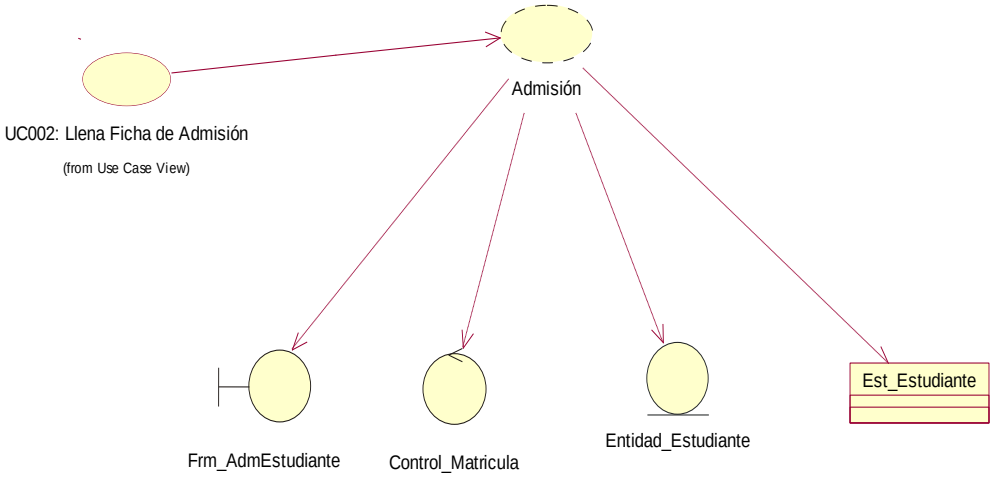


Figura 8. *Caso de uso de Realización Admisión CUR002. En este caso de realización la secretaria le registra en el sistema para posteriormente realizarle la matricula.*

Tabla 11

Especificación del Caso Realización Admisión.

Nombre	**Admisión**
Identificador	UCR002
Responsabilidades	Registrar datos personales del Alumno
Tipo	Sistema
Referencias de Casos de Uso	UC002
Referencias Requisitos	FR02
PRECONDICIONES	
Instancia **Llena todos los datos personales del estudiante.** **Relación** **Se dirige a la interfaz del Estudiante.**	
POSCONDICIONES	
Instancia **Verifica los datos ingresados del Estudiante.** **Relación.** **La tabla estudiantes se utiliza para guardar y recuperar los Datos.**	
SALIDAS PANTALLAS	
Interfaz de usuario guarda los datos del nuevo estudiante	

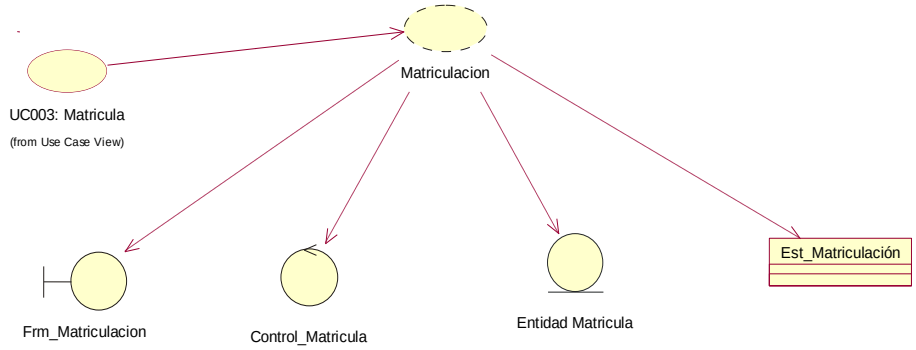


Figura 9 *Caso de uso de Realización Matricula CUR003. En este caso de realización la secretaria procede a matricular al estudiante y guarda los datos en la tabla de matrícula*

.

Tabla 12

Especificación del Caso Realización Matricula.

Nombre	**Matriculación**
Identificador	UCR003
Responsabilidades	Asignación de un curso, paralelo, jornada al alumno
Tipo	Sistema
Referencias de Casos de Uso	UC003
Referencias Requisitos	FR03
PRECONDICIONES	
Instancia Interfaz Matriculación de alumnos. **Relación** Tiene una interfaz para empezar con el proceso de matriculación	
POSCONDICIONES	
Instancia Llena los campos con la verificación de los datos de la matricula **Relación.** Utiliza la tabla Matricula para guardar y recuperar datos	
SALIDAS PANTALLAS	
Curso asignado exitosamente Jornada asignada exitosamente Paralelos Asignados exitosamente	

3.05 Diagrama de Secuencias del Sistema.

Un diagrama de secuencia muestra la interacción de un conjunto de objetos en una aplicación a través del tiempo. Esta descripción es importante porque puede dar detalle a los casos de uso, aclarándolos al nivel de mensajes de los objetos existentes, como también muestra el uso de los mensajes de las clases diseñadas en el contexto de una operación.

Típicamente se examina la descripción de un caso de uso para determinar qué objetos son necesarios para la implementación del escenario. Si se dispone de la descripción de cada caso de uso como una secuencia de varios pasos, entonces se puede "caminar sobre" esos pasos para descubrir qué objetos son necesarios para que se puedan seguir los pasos. Un diagrama de secuencia muestra los objetos que intervienen en el escenario con líneas discontinuas verticales, y los mensajes pasados entre los objetos como flechas horizontales.

3.05.1 Diagrama de Secuencia de Documentación.

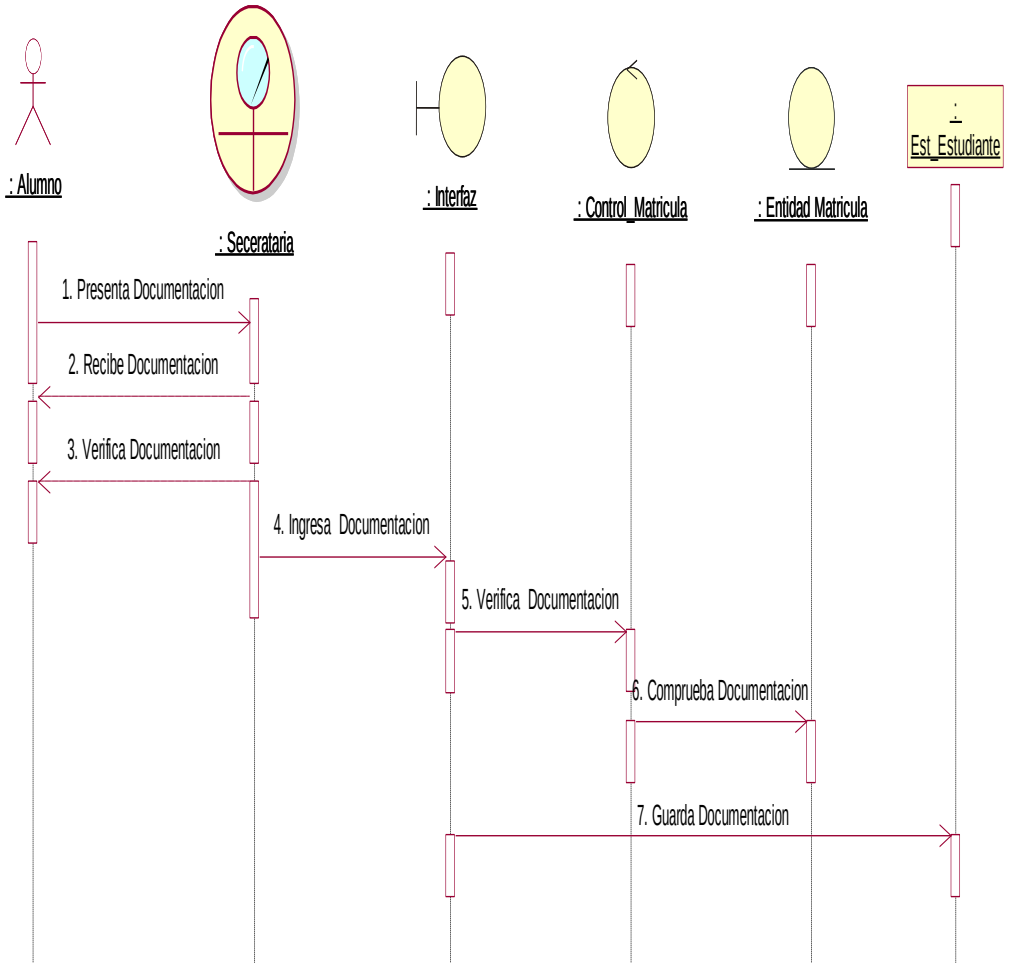


Figura 10. *Diagramas de Secuencia Documentación. En este diagrama de Secuencia los estudiantes presenta la documentación a la secretaria, ella revisa que este completos y registra en el sistema todo lo presentado en la base de datos en la tabla de estudiantes.*

3.05.2 Diagrama de Secuencia de Admisión.

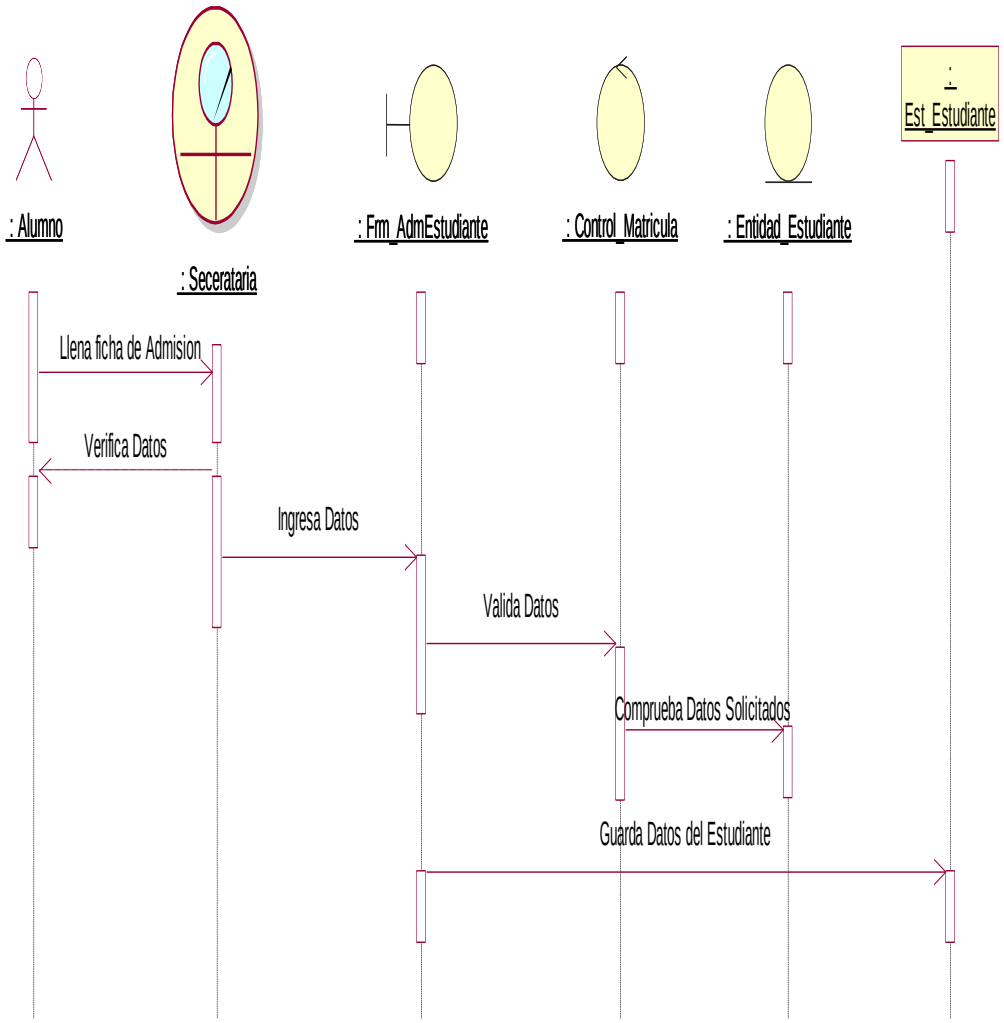


Figura 11. *Diagramas de Secuencia Admisión. En este diagrama de secuencia los estudiantes llenan un formulario de admisión con sus respectivos datos el cual verifica que estén completo la secretaria para proceder a registrar al estudiante en el sistema.*

3.05.3 Diagrama de Secuencia de Matricula.

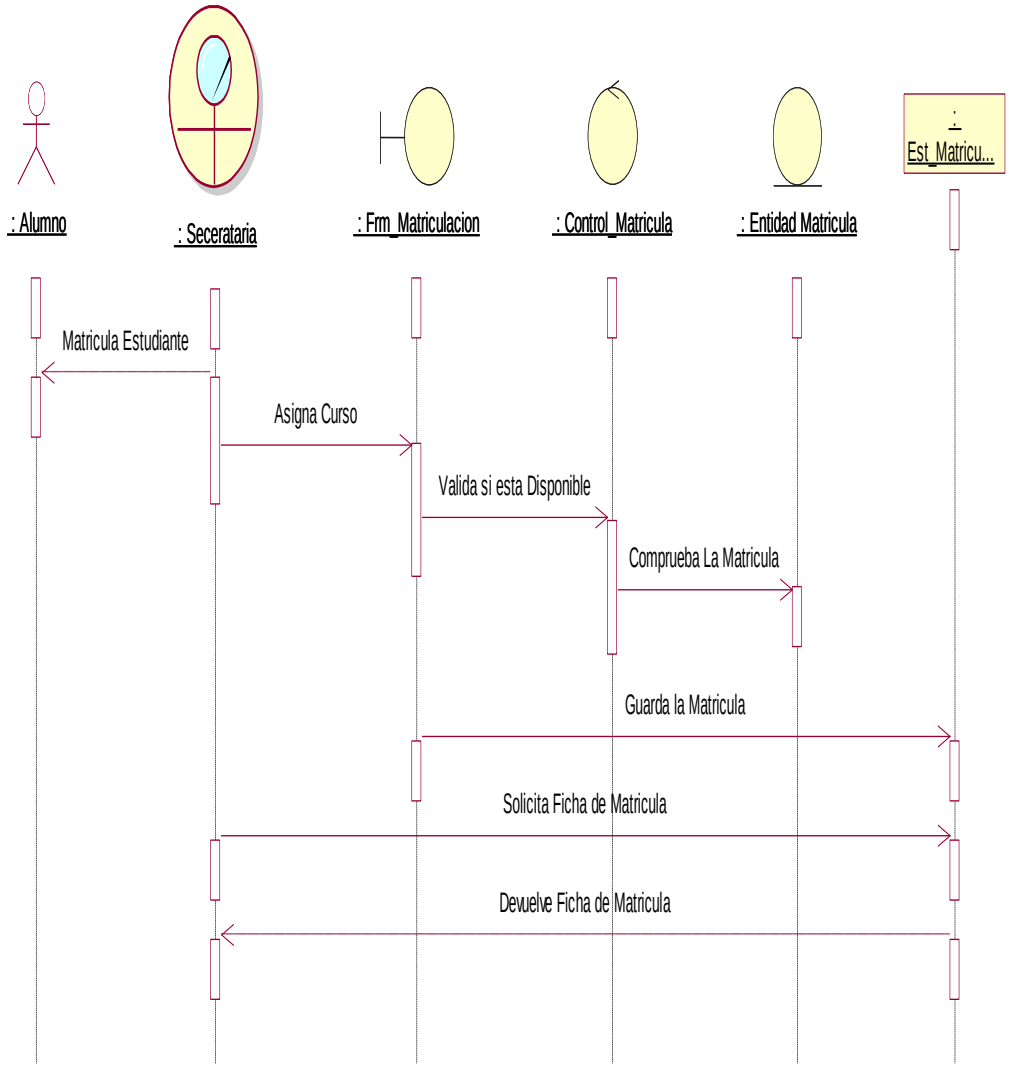


Figura 12 *Diagramas de Secuencia Matricula. En este diagrama de secuencia la secretaria verifica que el estudiante esté inscrito en el sistema para proceder a matricularle y registrar en la base de datos en la tabla de matrícula.*

Capítulo IV: Análisis de Alternativas.

4.01 Matriz de Análisis de Alternativas.

Tabla 13

Detalle donde se realiza un análisis estadístico de alternativas

Objetivos	Impacto sobre el propósito	Factibilidad Técnica	Factibilidad Financiera	Factibilidad Social	Factibilidad Política	Total	Categoría
Obtener Documentos actualizados y ordenados	5	5	3	3	2	18	Alta
Obtener Eficiencia en el manejo de información	5	5	3	3	2	18	Alta
Facilitar la búsqueda de información	5	4	3	3	2	17	Alta
Obtener la información Centralizada	5	5	2	2	2	16	Media Alta
Mejorar presentación y tiempo en la entrega de reportes	4	4	1	1	2	12	Media Baja
Mantener satisfecho al Usuario	4	4	1	5	1	15	Media Alta
Aumentar el avance tecnológico en la institución	5	5	2	5	4	21	Alta
TOTAL:	33	32	16	22	15	135	

4.02. Matriz de Impactos de Objetivos.

Tabla 14

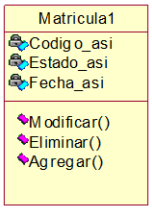
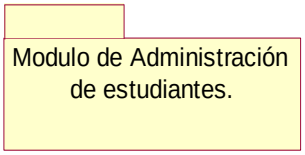
Detalle del análisis de la matriz de Impacto de Objetivos.

Objetivos	Factibilidad de Lograse	Impacto en Género	Impacto Ambiental	Relevancia	Sostenibilidad	Total
	(A-M-B)	**(A-M-B)**	**(A-M-B)**	**(A-M-B)**	**(A-M-B)**	**Puntos**
	(4-2-1)	**(4-2-1)**	**(4-2-1)**	**(4-2-1)**	**(4-2-1)**	
	Los beneficios son mayores que los costos	Incrementa la participación de la mujer	Fomenta el reciclaje	Responde a las expectativas de los beneficiarios	Fortalece a los usuarios involucrados	89
	Es aceptable y conveniente para los beneficiarios	Incrementa el nivel educativo de la mujer	Mejora el entorno social	Es una prioridad sentida por los beneficiarios	Fortalece la Organización local	0 a 10
	Existe tecnología adecuada para su realización	Lo podrán manejar diversidad de género, sin excepción	Mejora el entorno cultural	Beneficia a grupos de mayor carencia y vulnerabilidad	Fortalece la participación de los beneficiarios y población local	BAJA
	Se minimizará tiempo		Protege el uso de los recursos	Los beneficios son deseados por los beneficiarios		7 a 11
	El tiempo para el desarrollo es el adecuado		Favorece la educación ambiental	Los usuarios quedarán satisfechos		MEDIA BAJA
						12 a 16
						MEDIA ALTA
Puntos						17 a 25
						ALTA
	25	13	15	25	11	

4.03 Estándares para el Diseño de Clases.

Tabla

Estándares de Diseño.

Tipo	Descripción	Figura
Clase	El nombre de las clases se inicia con "TBL_NOMBRE" Ej.: "TBL_CURSO"	
Caso de Uso	Este caso de uso nos describe la acción de matricular al estudiante, complementa el comportamiento de un modelo.	 UC003: Matricula
Clase Activa	Esta clase trata en la ejecución del proceso de matriculación se encuentra con hilos de ejecución con el estudiante, curso, jornada.	
Paquete	Empleamos para organizar los elementos de un módulo.	
Asociación	Es una relación estructural que resumen las conexiones entre objetos. En este caso matricula con estudiantes.	
Generalización	Esta relación los elementos generalizados pueden sustituir por cualquiera de los elementos	
Realización	Estudiante cumple con todos los Riquisitos	

4.04 Diagrama de Clases.

Lenguaje unificado de modelado (UML, por sus siglas en inglés, Unified Modelling Language) es un lenguaje estándar para la especificación, visualización, construcción y documentación de artefactos de sistemas de software, muy bueno para la modelación de negocios y otros sistemas que no son software. El UML representa una colección de las mejores prácticas de ingeniería que tienen una probación exitosa en la modelación de sistemas largos y complejos

Véase Anexo A.02.

4.05 Modelo Lógico – Físico.

Estos modelos implementamos para el desarrollo del sistema, mediante el cual se le realizo el diseño de los procesos que interviene en la matricula en las unidades académicas. A continuación se explica cual fueron los pasos para realizar el diseño.

El estudiante reúne todos los requisitos y presenta a la secretaría, la cual le registra en el sistema, después de que se encuentre registrado se procederá a matricularle en una jornada y en donde se le asignara un curso con el respectivo paralelo además de la especialidad de bachillerato unificado.

Véase Anexo A.03.

4.06. Diagrama de Componentes.

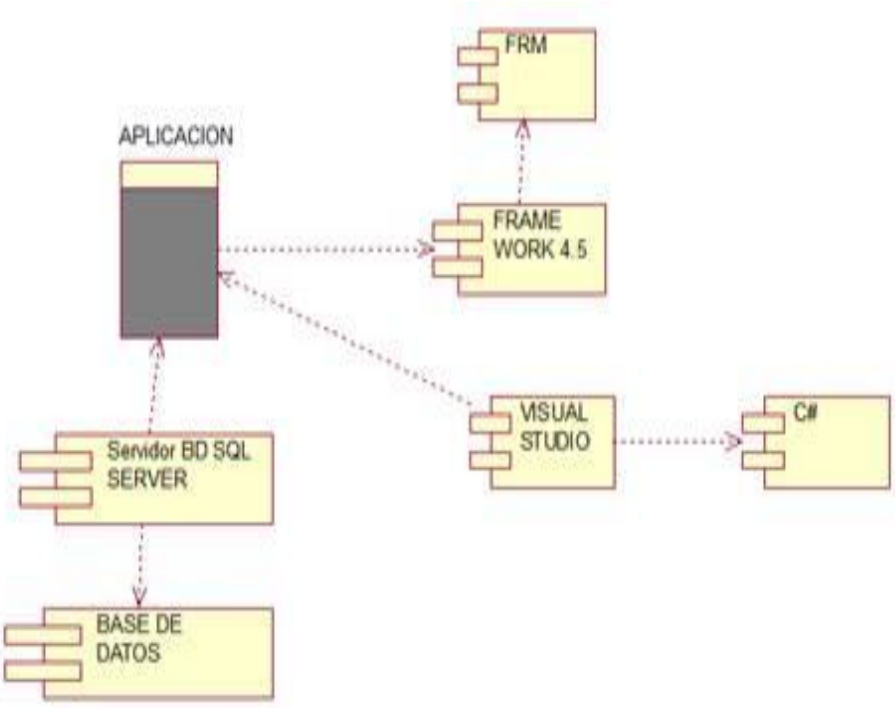


Figura 13 *Detalle de especificación de los módulos y las capas del sistema. Se describe los componentes con los cuales están compuestos nuestro sistema.*

4.07 Diagramas de Estrategias.

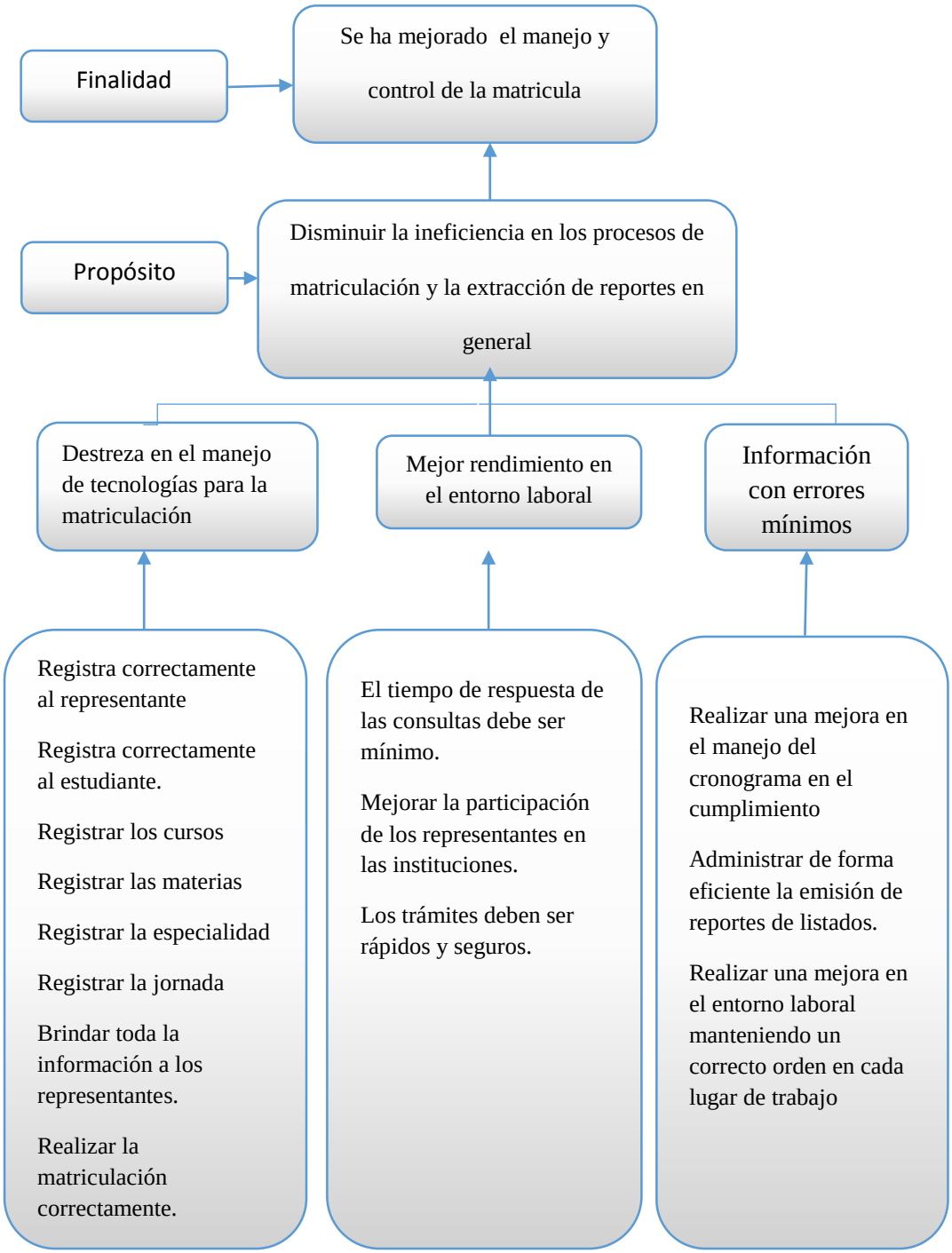


Figura 14. *Donde se especifica las estrategias para llegar a una finalidad. En este diagrama especificamos algunas de las estrategias utilizadas en nuestra aplicación.*

4.08 Matriz de Marco Lógico.

Tabla 15

Resumen del proyecto que destaca lo que se desea Lograr.

Resumen narrativo de objetos	Indicadores	Medios de verificación	Supuestos
Fin: **Mejor el manejo y control de matrículas**	Proceso de obtención de reportes de manera más ágil y eficiente.	Fácil obtención de datos de todos los actores involucrados en el negocio.	Capacitación a empleados que utilicen herramientas tecnológicas.
Propósito: **Disminuir el tiempo en los procesos de matriculación y generar los reportes en general**	Reducir tiempos y aumentar la productividad.	Optimizar recursos y mejorar los procesos.	Mayor satisfacción de los usuarios que utilicen el sistema
Componentes: 1. **Sistema fácil de utilizar y amigable la interfaz con el usuario.**	Realizar pruebas a la aplicación con varios usuarios, comprobando la complejidad de su uso.	Realizar modelos que muestre las mejoras que se está logrando	No contemplar las restricciones de la aplicación.
Actividad:			
1. **Analizar cómo funcionan dichos proceso.**	Tener claras las reglas del negocio para el desarrollo de la aplicación.	Realizar una documentación de todo aquello que se esté realizando según avanza el proyecto.	Tiempo que se dispone es demasiado corto para desarrollar la aplicación en su totalidad.
2. **Levantar requerimientos.**			
3. **Realizar la Base de Datos y posteriormente desarrollar la aplicación.**			

4.09 Vistas Arquitectónicas.

4.09.01. Vista Lógica.

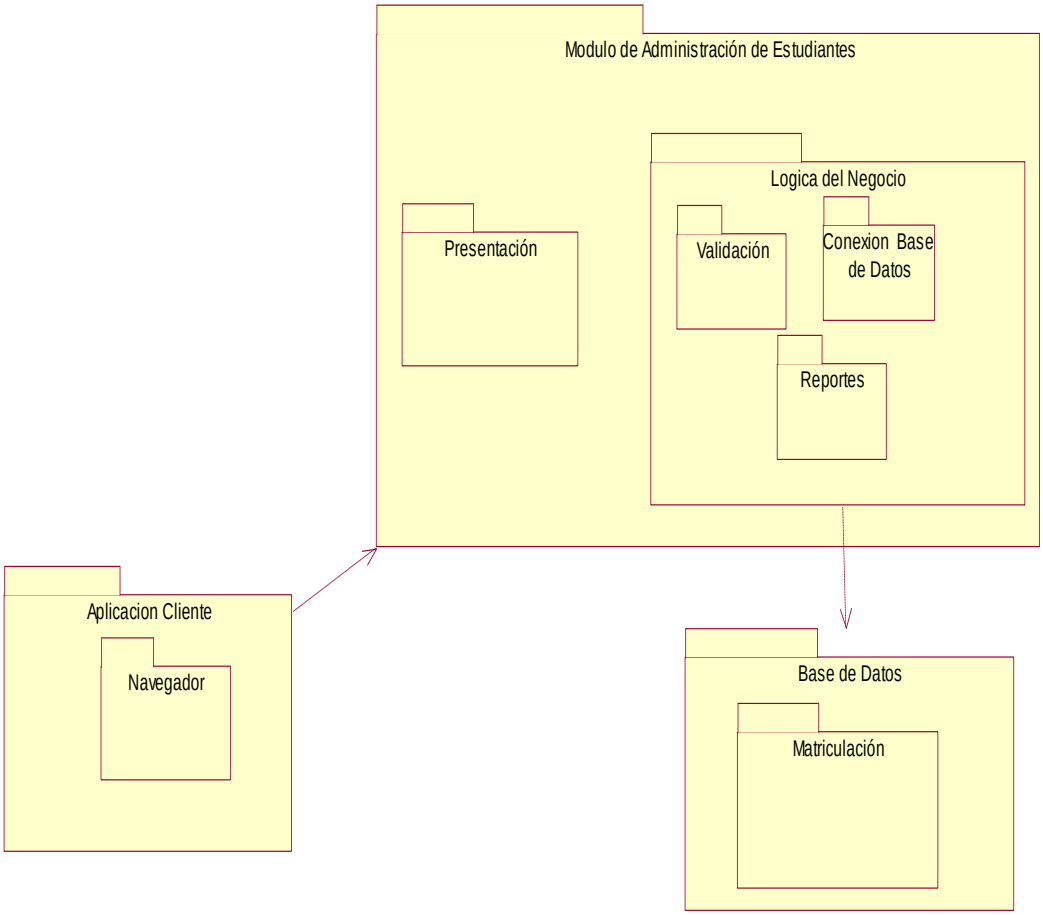


Figura 15. *Descripción de la lógica del sistema.*

4.09.02. Vista física.

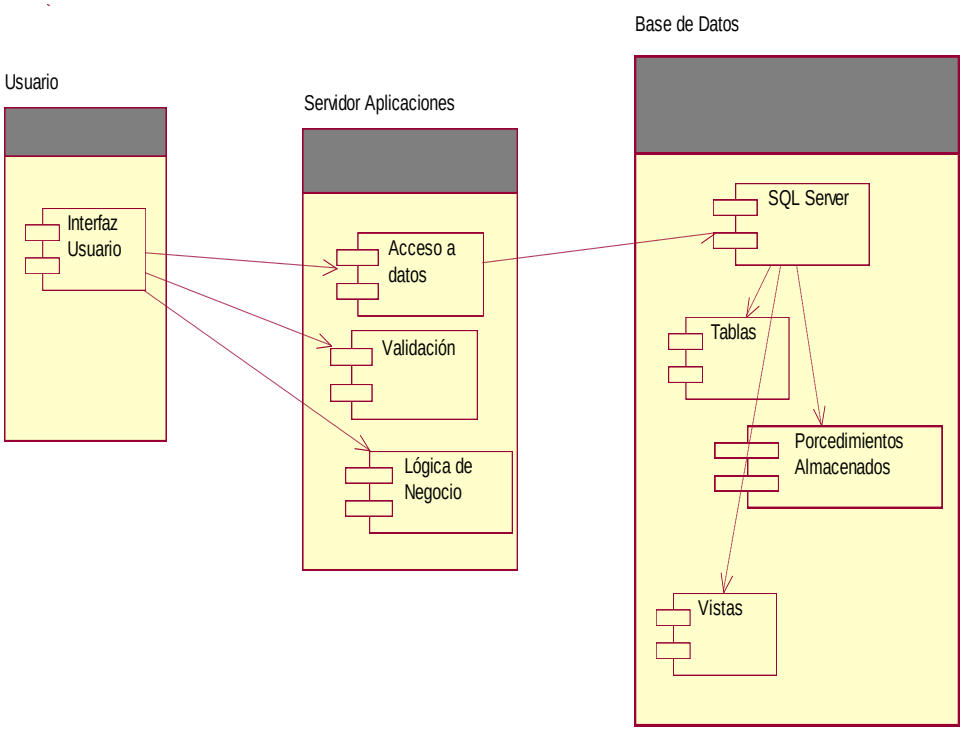


Figura 16. *Descripción física del sistema. Se describe como realizaría los usuarios las consultas de su información.*

4.09.03. Vista de Desarrollo.

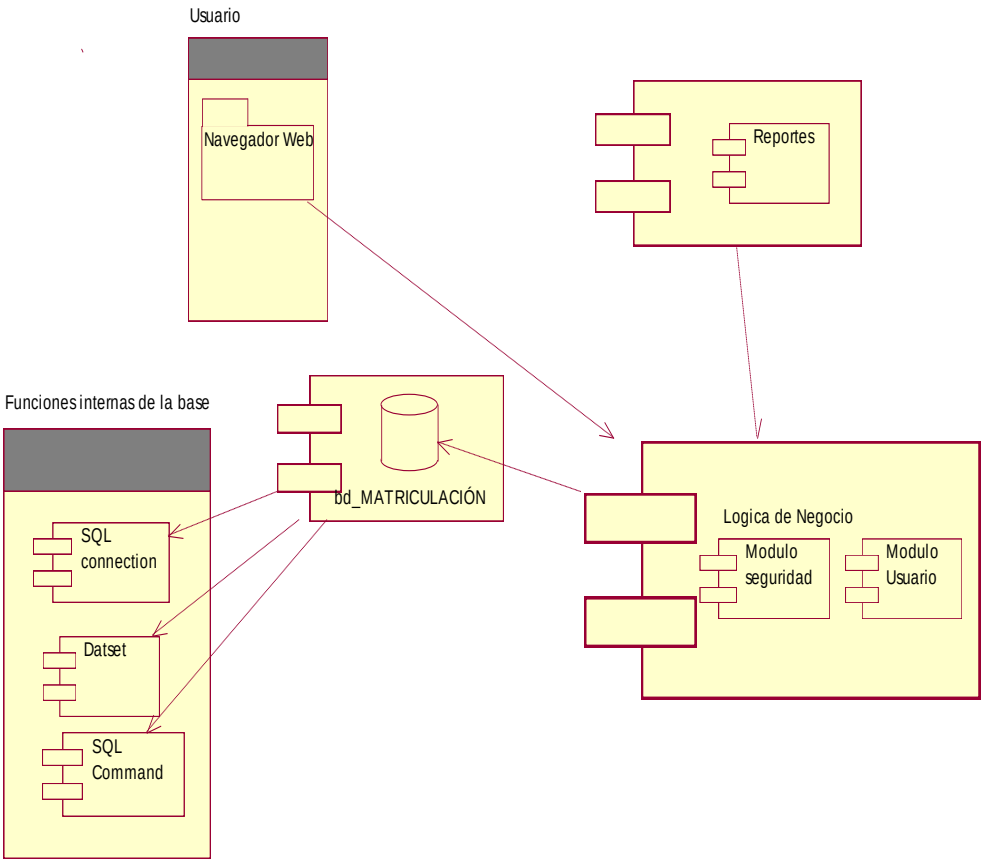


Figura 17. *Descripción detallada del sistema mediante componentes. Se describe como nuestros usuario realizan los procesos de registro, consulta y eliminación de sus datos.*

4.09.04 Vista de Procesos.

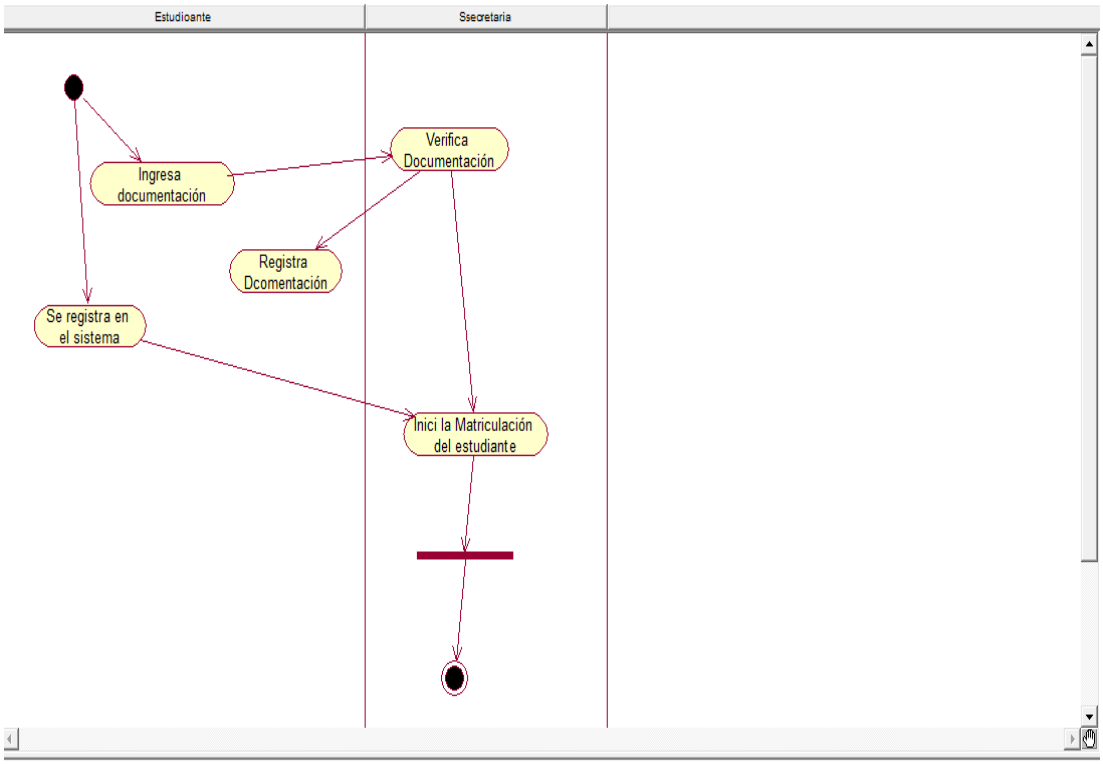


Figura 18 *Vista del Proceso como Matricula la Secretaria.*

Capítulo V: Propuesta.

5.01 Especificación de Estándares de Programación.

Este punto tiene como objetivo reglamentar la forma en que se implementará el código fuente del proyecto, pasando, por las variable, controles, ficheros, archivos y todo aquello que esté implicado en el código.

5.01.1 Declaraciones de Variables.

- La longitud debe ser lo más recomendable posible.
- El tipo de dato al que pertenece la variable.

Tabla 16

Detalle de la descripción de un Variable.

Estructura	Descripción de la Variable
LONGITUD. MAX.	← 1 →← 16 →
FORMATO	Todo con minúsculas
EJEMPLO	$meses, $a = 1; $cadena="Hola amigo";

Tabla 17

Detalle de la descripción de clases.

Título	Descripción
Sintaxis	public partial class
Descripción	El tipo de variables son Private, Public o Protected.
Observaciones	En la declaración de clases no se deberá utilizar caracteres como:
	Letra Ñ o ñ.
	Caracteres especiales ¡^, #, $, %, &, /, (,), ¿‘, +, -, *, {,}, [,].
	Caracteres tildados: á, é, í, ó, ú.
Ejemplo	public partial class
	Indica una clase parcial

- ✓ Calendar (Control de servidor Web)
- ✓ CheckBox y CheckBoxList (Controles de servidor Web)
- ✓ DropDownList (Control de servidor Web)
- ✓ FileUpload (Control de servidor Web)
- ✓ Image (Control de servidor Web)
- ✓ Label (Control de servidor Web)
- ✓ ListBox (Control de servidor Web)
- ✓ RadioButton y RadioButtonList (Controles de servidor Web)
- ✓ Table, TableRow y TableCell (Controles de servidor Web)
- ✓ TextBox (Control de servidor Web)
- ✓ XML (Control de servidor Web)

5.02. Diseño de Interfaces de Usuario.

La interfaz diseñada para el usuario es el proceso de determinar los distintos componentes, tanto de hardware como de software, sus características y su disposición, que se utilizarán para interactuar con una serie de usuarios determinados en un medio ambiente determinado.

Figura 19: *Diseño de interfaz de página general del sistema.*

1. Menú Desplegable

En este botón vamos a encontrar la misión que posee cada institución.

2. Menú Desplegable

En este botón encontraremos la visión que posee cada institución.

3. Menú desplegable

En este botón encontraremos toda la información de la institución.

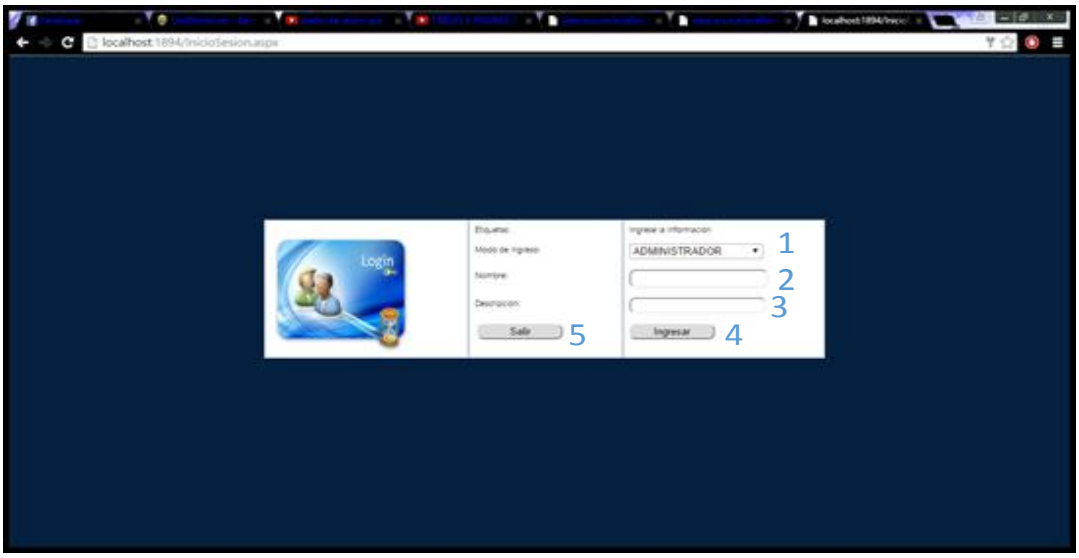


Figura 20: *Diseño de inicio de sesión dependiendo de su perfil.*

1. DropDownList ID.-cmbTipoUsuario.

Debemos seleccionar el tipo de usuario que seamos dependiendo su necesidad.

2. TextBox ID.-txtUsuario.

En este campo se tendrá que ingresar el nombre de usuario, el cual se le proporciona al administrador del sistema en el momento que se realiza el respectivo registro.

3. TextBox ID.-txtContrasena.

En este campo se tendrá que ingresar la contraseña de usuario, la cual proporciona el administrador del sistema en el momento del registro.

4. Button ID.-btnRegistrar.

Una vez llenados los campos anteriores se debe dar clic en este botón para que nos re direccione a la página de administración del sistema.

4. Button ID.-btnSalir.

Este botón hace que el sistema se direccione a la página principal y no está seguro del usuario y contraseña.

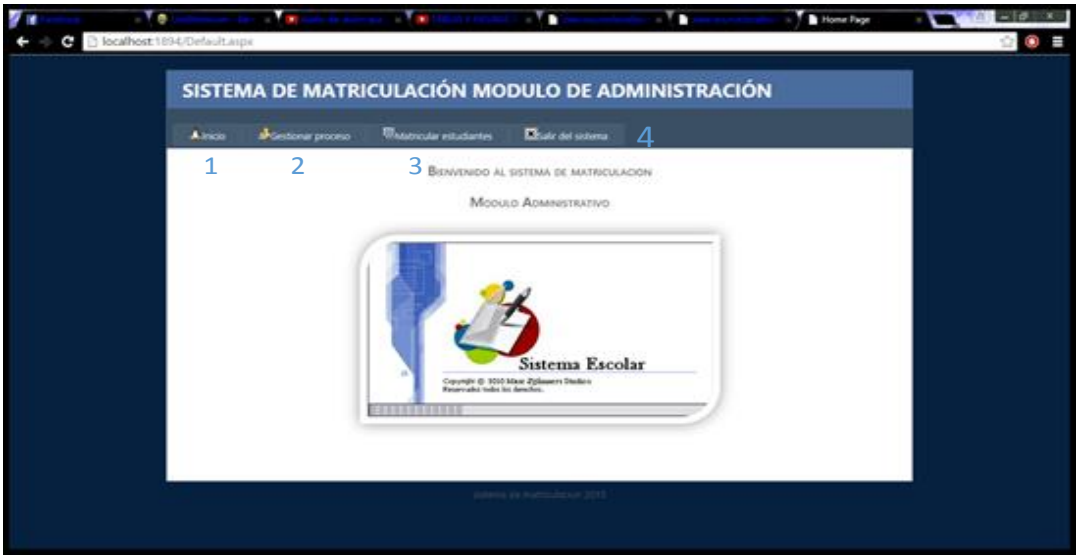


Figura 21: *Diseño de la interfaz general del Administrador.*

1. Menu ID.-NavigationMenu.

Esta opción permite que en cualquier parte que se encuentre el sistema le permite regresar al menú.

2. MenuItem Text.-Gestionar proceso.

Esta opción nos permite acceder al mantenimiento de representantes, estudiantes,

cursos, docentes, materias, especialidades, jornadas donde se podrá almacenar.

3. MenuItem Text.-Matricular estudiantes.

Esta opción nos permite acceder a matricular a los estudiantes previamente inscritos.

4. MenuItem Text.-Salir del sistema. Esta opción nos direcciona a la pantalla, principal del sistema.

Figura 22: *Diseño de la interfaz general del Administrador para registrar representantes*

1. TextBox ID.-txtCedula.

En este campo se deberá ingresar la cedula del representante, cabe destacar que este solo admitirá el ingreso de números.

2. TextBox ID.-txtapellido.

En este campo se deberá ingresar los apellidos del representante, cabe destacar que este solo admitirá el ingreso de letras.

3. TextBox ID.-txtnombre.

En este campo se deberá ingresar los apellidos del representante, cabe destacar que este solo admitirá el ingreso de letras.

4. TextBox ID.-txtdireccion.

En este campo se deberá ingresar el número de teléfono del representante, cabe destacar que este solo admitirá el ingreso de números.

5. TextBox ID.-txtacelular.

En este campo se deberá ingresar el número de celular del representante, cabe destacar que este solo admitirá el ingreso de números.

6. TextBox ID.-txtdireccion.

En este campo se deberá ingresar la dirección completa del representante

7. TextBox ID.-txtcorreo.

En este campo se deberá ingresar el correo del representante, cabe destacar que este solo admitirá el ingreso de letras y números.

8. Button ID.-btnRegistrar.

Este botón guarda los registros una vez que los campos hayan sido llenados.

Figura 23: *Diseño de la interfaz general del Administrador para registrar representantes.*

1. TextBox ID.-txtCedula.

En este campo se deberá ingresar la cedula del representante, que necesitemos verificar si se registró correctamente.

2. Button ID.-btnBuscar.

Este botón buscar permite realizar la búsqueda de los registro recién ingresados

Figura 24*: Diseño de la nos permite buscar al representante con la cedula para modificar sus datos.*

1. TextBox ID="txtCedula"

En este campo se deberá ingresar la cedula del representante, que necesitemos modificar sus datos

2. Button ID.-btnbuscar.

El botón buscar nos permite realizar una búsqueda del representante que necesitemos modificar sus datos.

3. Campos del Representante TextBox

En estos campos se cargan los datos del representante después de haber realizado búsqueda para posteriormente realizar las modificaciones de los campos que se requiera

4. Button ID.-btnModificar.

Este botón una vez que se hayan realizado los cambios guarda las modificaciones.

Figura 25: *Diseño de la interfaz permite la eliminación de registros.*

1. TextBox ID.-txtCedula.

En este campo se deberá ingresar la cedula del representante, al que posteriormente le vayamos a borrar.

2. Button ID.-btnRegistrar.

El botón buscar nos permite realizar una búsqueda del representante al que vayamos a borrar del sistema.

3. GridView ID.-dgvRepresente.

En este griview se muestran todos los representantes que han sido registrados con el sistema a los cuales vamos a seleccionar y borrar todos sus datos.

4. Button ID.-btnEliminar.

Este botón nos permite la eliminación del registro representante.

Figura 26: *Diseño de la interfaz general del Administrador para registrar estudiantes.*

1. TextBox ID.-txtCedula.

En este campo se deberá ingresar la cedula del representante que sea tutoro

padre del estudiante.

2. Button ID.-btnBuscar.

El botón buscar nos permite realizar una búsqueda del representante al que le vamos

asignar su representado.

3. TextBox ID.-txtRepresentante.

En este campo se van a cargar los nombres y apellidos del representante.

4. Campos Estudiante TextBox

En estos campos vamos a ingresar datos de los estudiantes que vayamos a inscribir en el sistema.

5. Button ID.-btnRegistrar.

Este botón guarda los registros una vez que los campos del formulario hayan sido llenados correctamente.

Figura 27: *Diseño de la interfaz nos muestra un listado de los estudiantes registrados.*

1. TextBox ID.-txtCedula.

En este campo se deberá ingresar la cedula del estudiante, que necesitemos verificar si se registró correctamente.

2. Button ID.-btnBuscar.

Este botón buscar permite realizar la búsqueda de los registro recién ingresados.

3. GridView ID.-dgvEstudiante.

En esta tabla se muestran todos los estudiantes que han sido registrados en el sistema.

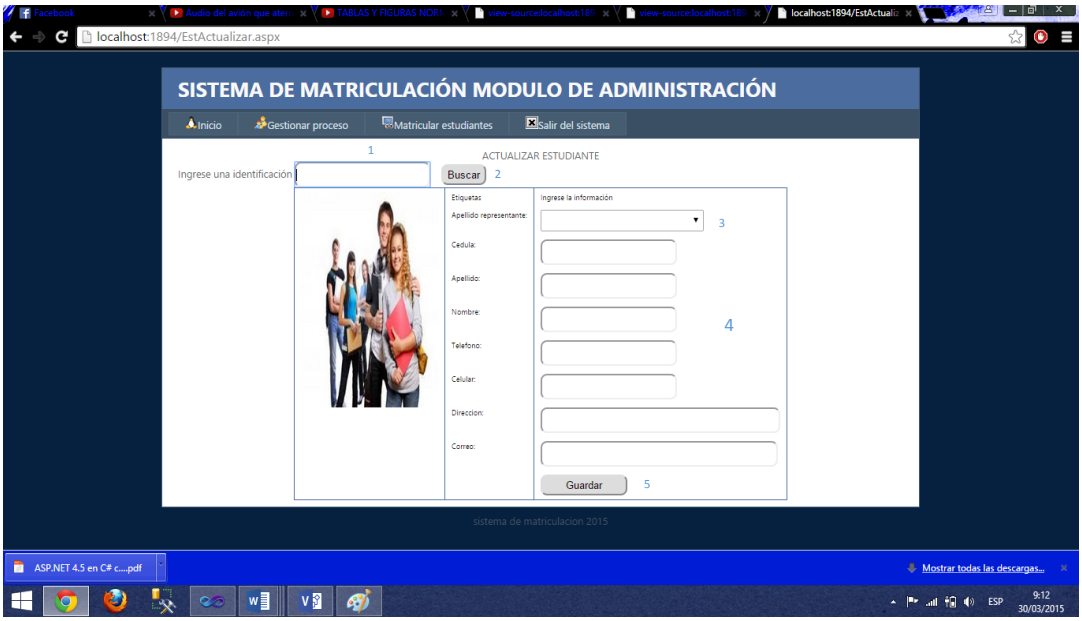


Figura 28: *Diseño de la nos permite buscar al estudiante con la cedula para modificar sus datos.*

1. TextBox ID.-txtCedula.

En este campo se deberá ingresar la cedula del estudiante.

2. Button ID.-btnBuscar.

El botón buscar nos permite realizar una búsqueda del estudiante que necesitemos modificar sus datos.

3. TextBox ID.-txtRepresentante.

En estos campos se cargan los datos del representante después de buscarle.

4. Campo del Estudiante TextBox

En estos campos se cargan los datos del estudiante después de haber realizado búsqueda para posteriormente realizar las modificaciones de los campos.

5. Button ID.-btnModificar.

Este botón una vez que se hayan realizado los cambios guarda las modificaciones.

Figura 29: *Diseño de la interfaz permite la eliminación de registros.*

1. TextBox ID.-txtCedula.

En este campo se deberá ingresar la cedula del estudiante, al que posteriormente le vayan a borrar.

2. Button ID.-btnBuscar.

El botón buscar nos permite realizar una búsqueda del estudiante

3. GridView ID.-dgvRepresente.

En este gridview se muestran todos los estudiantes que han sido registrados con el sistema a los cuales vamos a seleccionar y borrar todos sus datos.

4. Button ID.-btnEiminar.

Este botón nos permite la eliminación del registro estudiante.

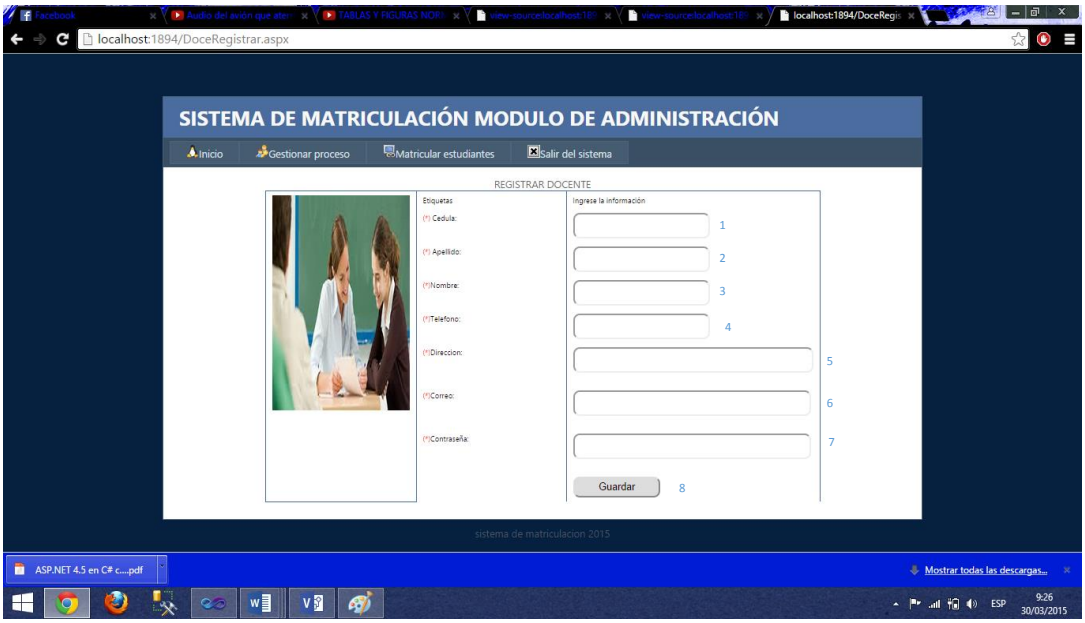


Figura 30: *Diseño de la interfaz general del Administrador para docentes.*

1. TextBox ID.-txtCedula.

En este campo se deberá ingresar la cedula del docente, cabe destacar que este solo admitirá el ingreso de números.

2. TextBox ID.-txtApellido.

En este campo se deberá ingresar los apellidos del docente, cabe destacar que este solo admitirá el ingreso de letras.

3. TextBox ID.-txtNombre.

En este campo se deberá ingresar los nombre del docente, cabe destacar que este solo admitirá el ingreso de letras.

4. TextBox ID.-txtTelefono.

En este campo se deberá ingresar el número de teléfono del docente, cabe destacar que este solo admitirá el ingreso de números.

5. TextBox ID.-txtCeular.

En este campo se deberá ingresar el número de celular del docente, cabe destacar que este solo admitirá el ingreso de números.

6. TextBox ID.-txtDireccion.

En este campo se deberá ingresar la dirección del docente, cabe destacar que este solo admitirá el ingreso de letras.

7. TextBox ID.-txtCorreo.

En este campo se deberá ingresar el correo del docente, cabe destacar que este solo admitirá el ingreso de letras y números.

8. Button ID.-btnRegistrar.

Este botón guarda los registros una vez que los campos hayan sido llenados.

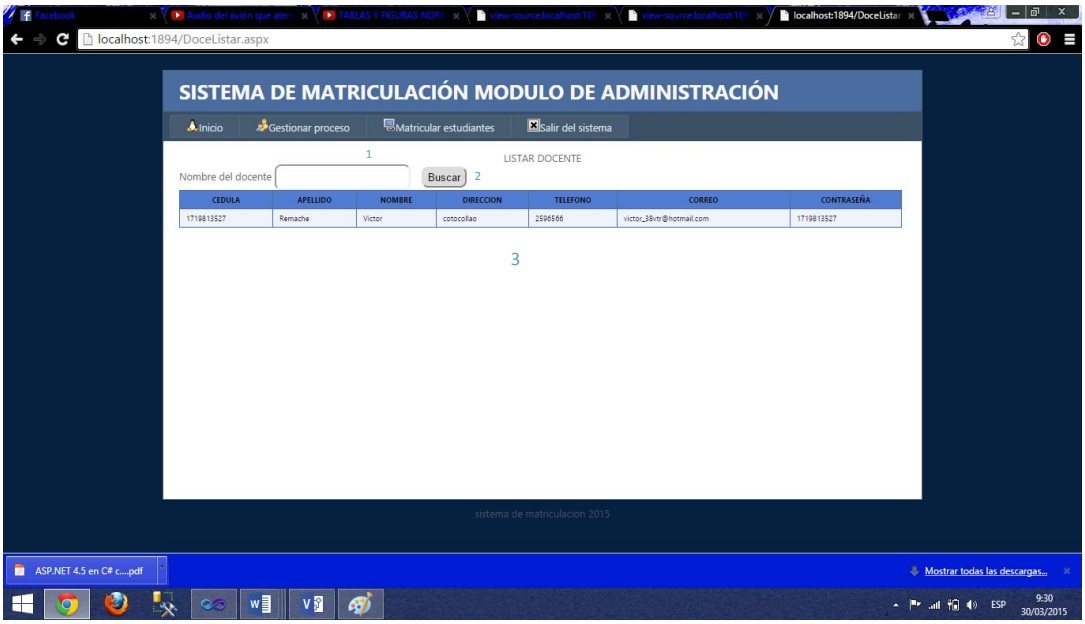


Figura 31: *Diseño de la interfaz nos muestra un listado de los docentes registrados.*

1. TextBox ID.-txtCedula.

En este campo se deberá ingresar la cedula del docente, que necesitemos verificar si se registró correctamente.

2. Button ID.-btnBuscar.

Este botón buscar permite realizar la búsqueda de los registro recién ingresados.

3. GridView ID.-dgvDocentes.

En esta gridview se muestran todos los estudiantes que han sido registrados en el sistema.

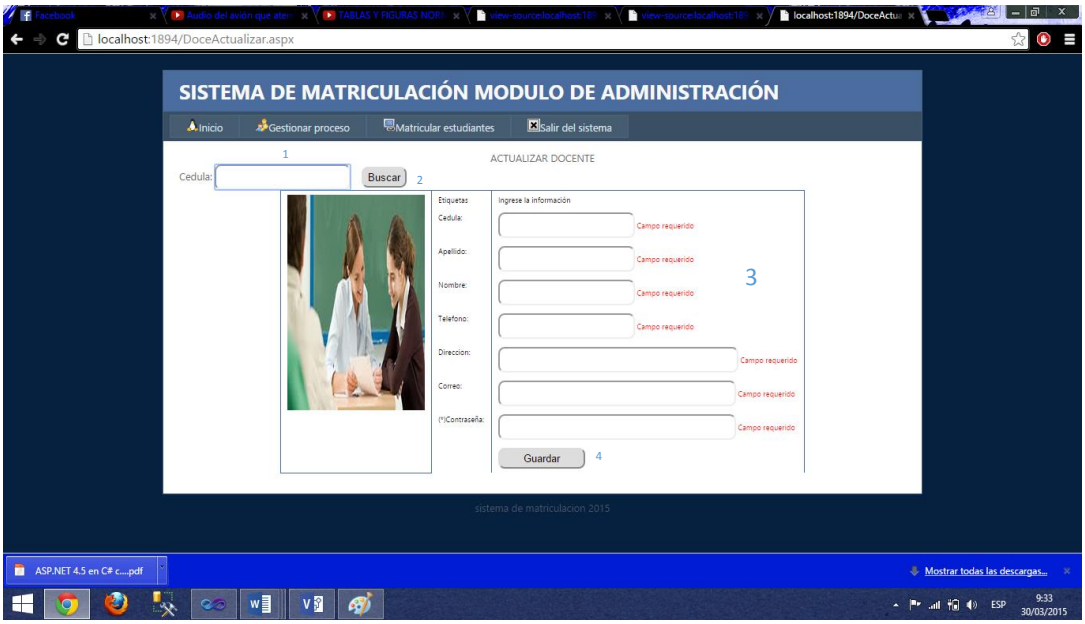


Figura 32: *Diseño de la nos permite buscar al docente con la cedula para modificar sus datos.*

1. TextBox ID.-txtCedula.

En este campo se deberá ingresar la cedula del docente, que necesitemos modificar sus datos

2. Button ID.-btnBuscar.

El botón buscar nos permite realizar una búsqueda del docente.

3. TextBox ID.-txtNombre.

En estos campos se cargan los datos del docente después de haber realizado búsqueda del estudiante.

4. Button ID.-btnModificar.

Este botón una vez que se hayan realizado los cambios guarda las modificaciones.

Figura 33: *Diseño de la interfaz permite la eliminación de registros.*

1. TextBox ID.-txtCedula

En este campo se deberá ingresar la cedula del docente ser eliminado.

2. Button ID.-btnBuscar

El botón buscar nos permite realizar una búsqueda del docente al que vayamos a borrar del sistema.

3. GridView ID.-dgvdocente

En esta gridview se muestran todos los docente que han sido registrados con el sistema a los cuales vamos a seleccionar y borrar todos sus datos.

4. Button ID.-btnEliminar.

Este botón nos permite la eliminación del registro docente.

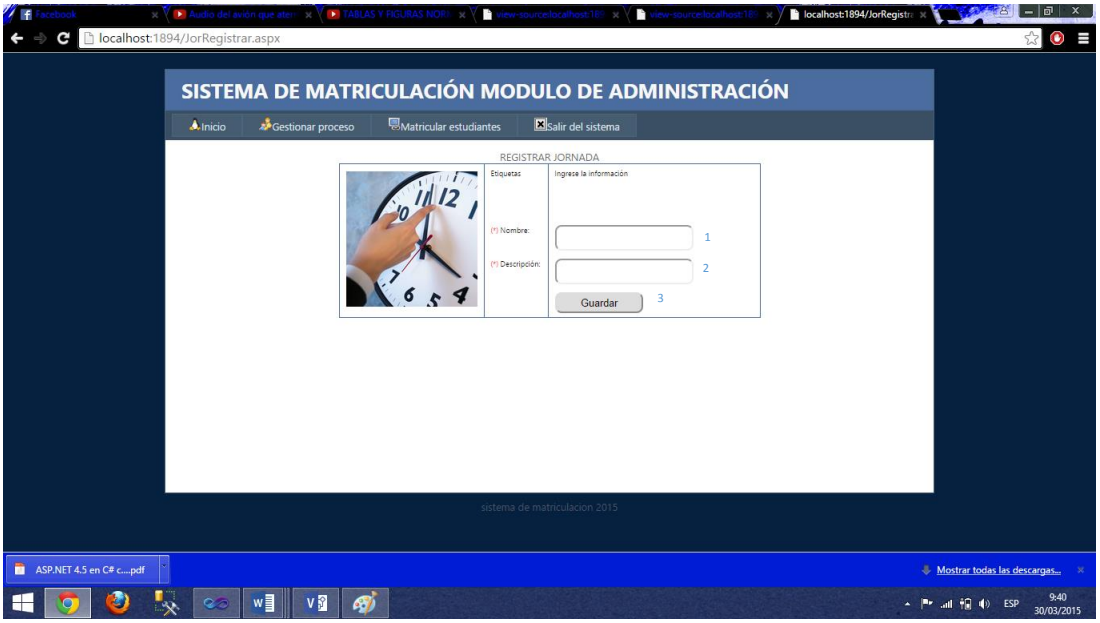


Figura 34: *Diseño de la interfaz general del Administrador para Jornadas.*

1. TextBox ID.-txtCedula.

En este campo se deberá ingresar el nombre de la jornada, cabe destacar que este solo admitirá el ingreso de letras.

2**.** TextBox ID.-Descripción.

En este campo se ingresa una pequeña descripción de la jornada

3. Button ID.-btnRegistrar.

Este botón guarda los registros una vez que los campos hayan sido llenados.

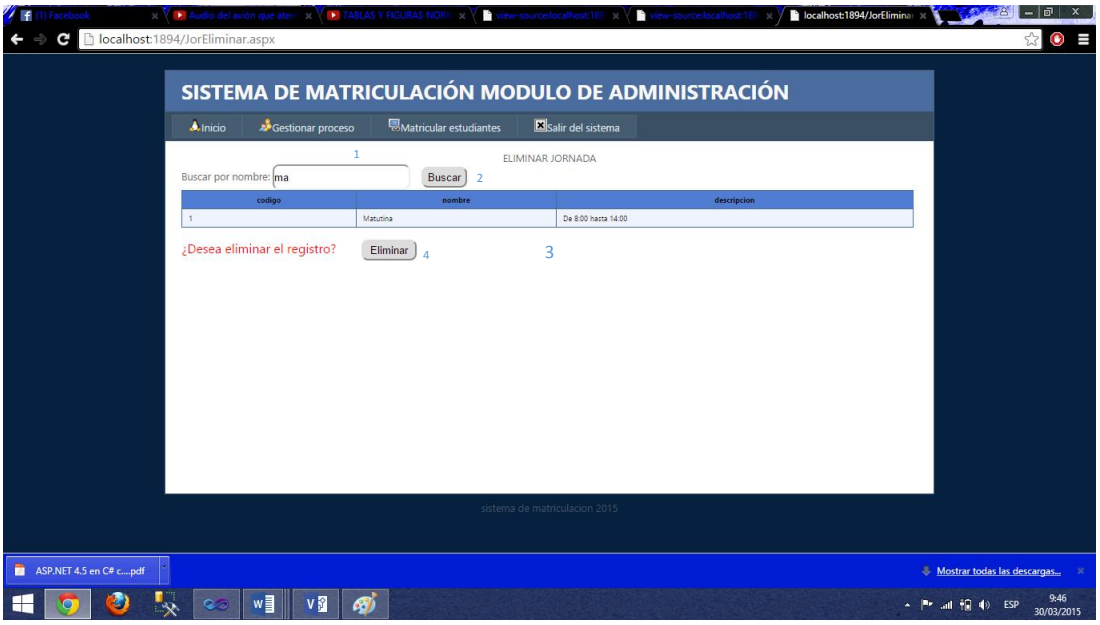


Figura 35: *Diseño de la interfaz permite la eliminación de registros.*

1. TextBox ID.-txtCedula.

En este campo se deberá ingresar el nombre de la jornada al que posteriormente le vayan a borrar.

2. Button ID.-btnBuscar.

El botón buscar nos permite realizar una búsqueda de la jornada .

3. GridView ID="dgvRepresente

En este gridview se muestran todas las jornadas que han sido registrados en el sistema a los cuales vamos a seleccionar y borrar todos sus datos.

4. Button ID.-btnEliminar.

Este botón nos permite la elimina

ción del registro jornada.

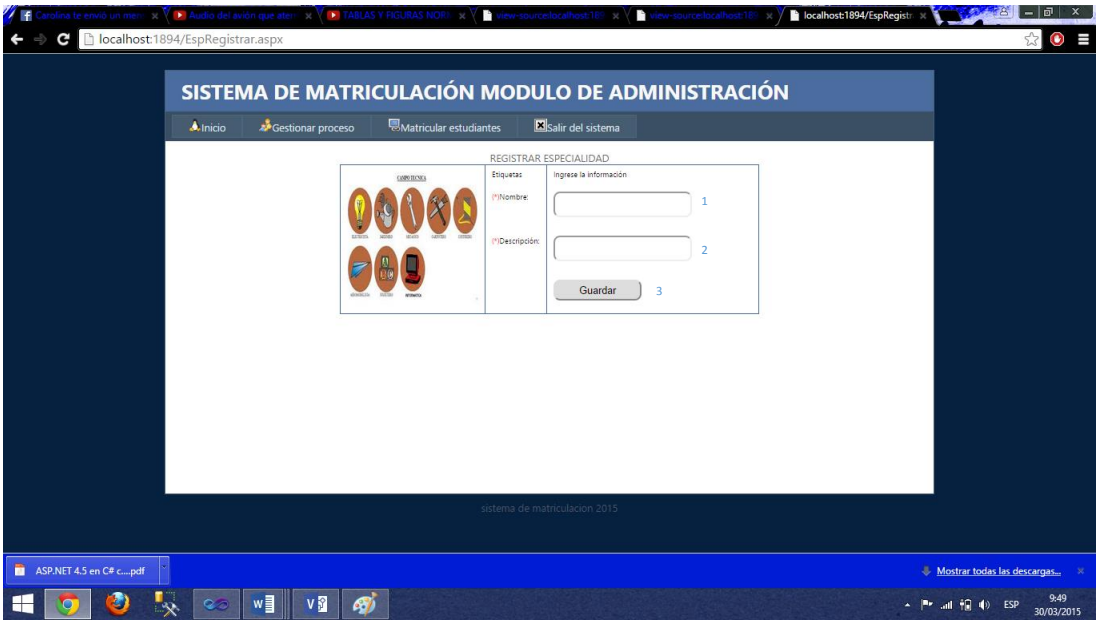


Figura 36: *Diseño de la interfaz general del Administrador para la especialidad.*

1. TextBox ID.-txtEspecialidad.

En este campo se deberá ingresar el nombre de la especialidad, cabe destacar que este solo admitirá el ingreso de letras.

2**.** TextBox ID.-txtDescripción.

En este campo se ingresa una pequeña descripción de la especialidad

3. Button ID.-btnRegistrar.

Este botón guarda los registros una vez que los campos hayan sido llenados.

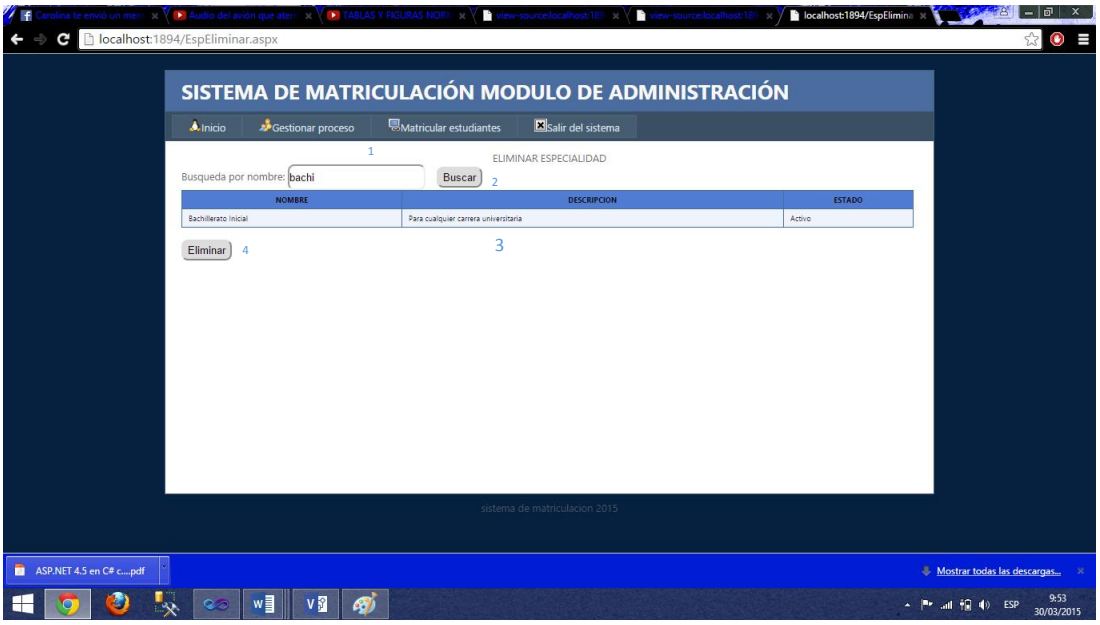


Figura 37: *Diseño de la interfaz permite la eliminación de registros.*

1. TextBox ID.-txtCedula.

En este campo se deberá ingresar el nombre de la especialidad al que posteriormente le vayamos a borrar.

2. Button ID.-btnEliminar.

El botón buscar nos permite realizar la eliminación de la especialidad.

3. GridView ID.-dgvEspecialidad.

En este gridview se muestran todas las especialidad que han sido registrados en el sistema a los cuales vamos a seleccionar y borrar todos sus datos.

4. Button ID.-btnEliminar.

Este botón nos permite la eliminación del registro especialidad.

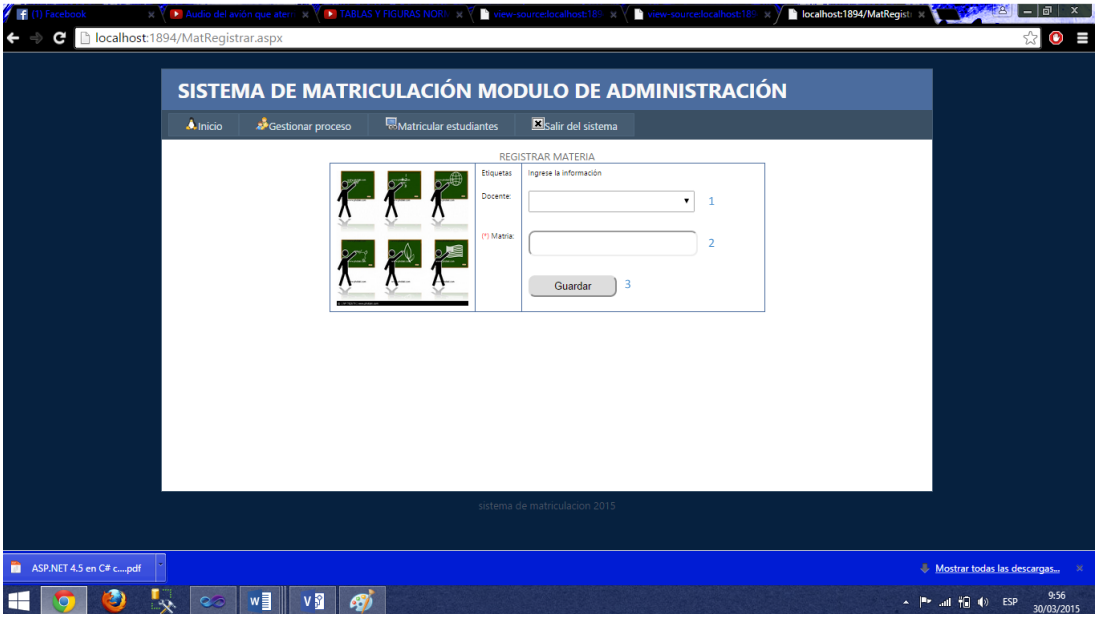


Figura 38: *Diseño de la interfaz general del Administrador para Materia.*

1. DropDownList ID.-cmbDocente.

En este campo se deberá seleccionar al docente que vaya a dictar esa materia.

2**.** TextBox ID.-Materia.

En este campo se ingresa la materia que va a dictar ese docente.

3. Button ID.-btnRegistrar.

Este botón guarda los registros una vez que los campos hayan sido llenados

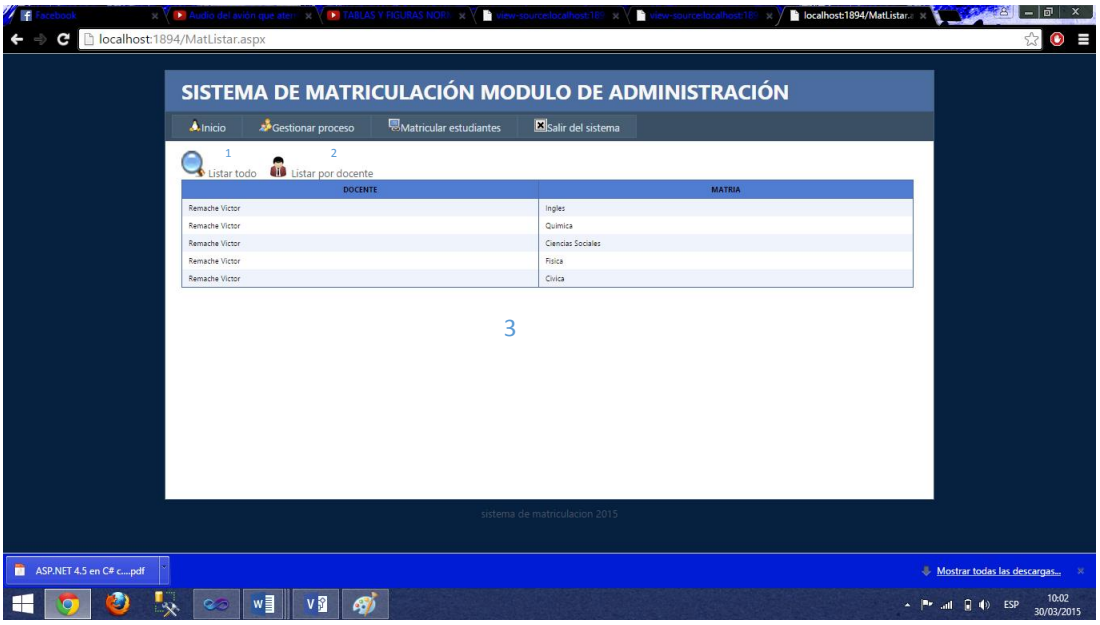


Figura 39: *Diseño de la interfaz permite ver registros de materia con docentes.*

1. GridView ID.-dgvMateria.

En esta opción nos permite observar todos los registros que han sido ingresados en las materias

2. GridView ID.-dgvMateria con Docente.

En esta opción nos permite observar las materias que dictara cada docente.

3. GridView ID.-dgvDocente.

En esta opción podemos observar los registros que hemos ido ingresando en nuestro sistema.

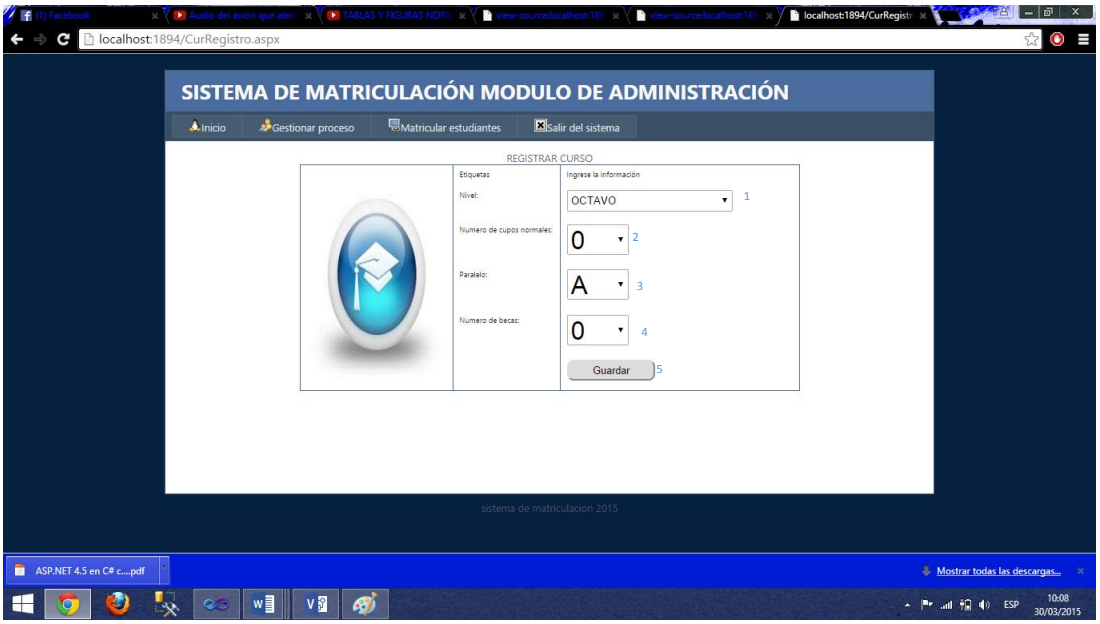


Figura 40: *Diseño de la interfaz general del Administrador para curso.*

1. DropDownList ID.-cmbCurso.

En este campo se deberá seleccionar un curso en el cual se va a inscribir el estudiante.

2. DropDownList ID.-lstNumero.

En esta opción podemos seleccionar la cantidad de estudiantes que se puede inscribir en dicho curso

3. DropDownList ID.-cmbParalelo.

En esta opción podemos seleccionar el paralelo en el cual nos podemos inscribir

4. Button ID.-btnRegistrar.

Este botón guarda los registros una vez que los campos hayan sido llenados

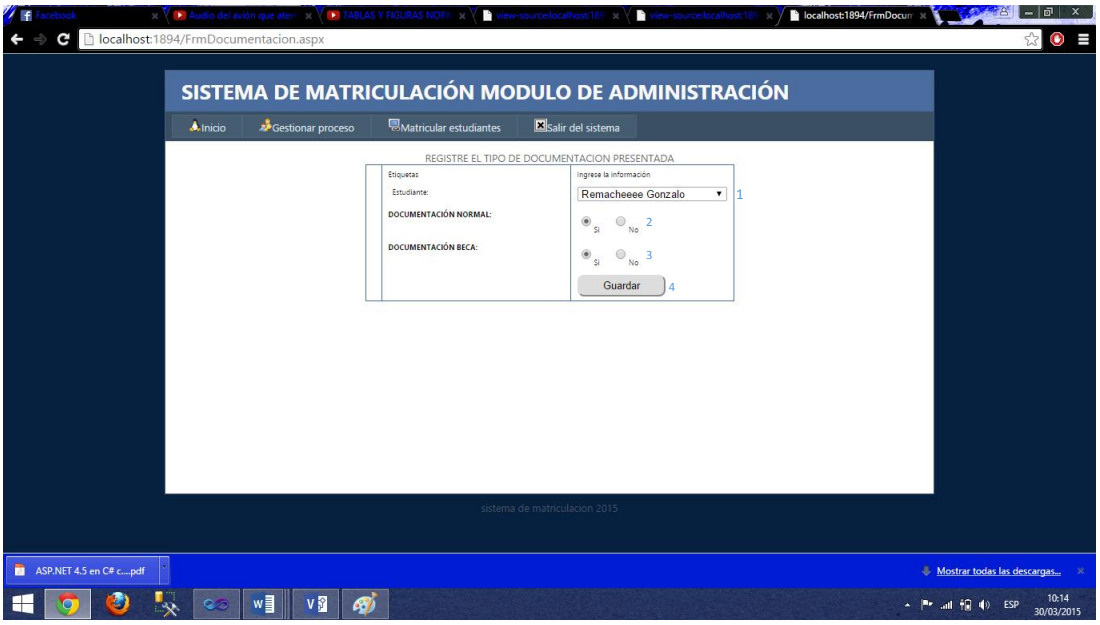


Figura 41: *Diseño de la interfaz general del Administrador para registrar documentos.*

1. DropDownList ID.-cmbEstudiant.

En esta opción debemos seleccionar a un estudiante para registrar la documentación que presenta o la que tenga pendiente por presentar

2. RadioButton ID.-rbBasico

En esta opción debemos seleccionar un check si presenta toda la documentación.

3. RadioButton ID.-rbBeca

En esta opción debemos seleccionar un check si solicita una beca en la cual presenta toda la documentación.

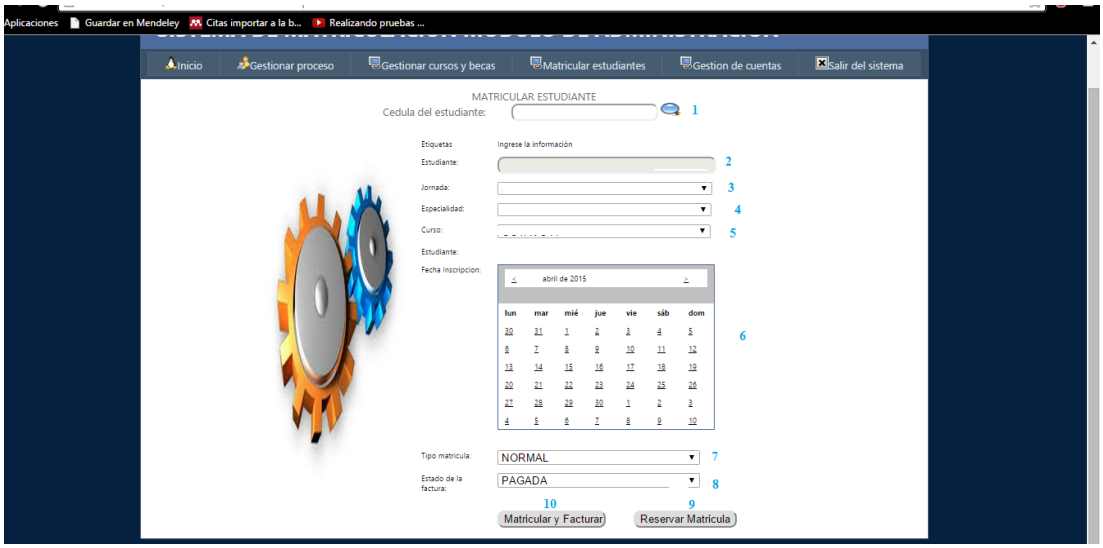


Figura 42: *Diseño de la interfaz general del Administrador para matricular estudiantes.*

1. TextBox ID.-txtCedula.

En esta opción debemos ingresar un numero de cedula

2. Button ID.-btnBuscar.

El botón buscar nos realiza una búsqueda del estudiante que se encuentre registrado en el sistema para posteriormente matricularle.

3. TextBox ID.-txtNombreEstudiante.

En ese campo nos carga los nombres y apellidos de los estudiantes que vamos a proceder a matricular.

4. DropDownList ID="cmbJornada

En esta opción vamos a seleccionar la jornada en la cual nos vamos a matricular.

5. DropDownList ID="cmbEspecialidad

En esta opción vamos a seleccionar la especialidad en la cual nos vamos a matricular.

6. DropDownList ID="cmbEspecialidad

En esta opción vamos a seleccionar le curso en el cual nos vamos a matricular

7. Calendar ID.-dtpFecha.

En este campo vamos a seleccionar la fecha actual en la que vamos a realizar la matricula

8. DropDownList ID.-cmbTipo

En esta opción vamos a seleccionar el tipo de matrícula que vamos a realizar.

9. DropDownList ID.-cmbEstado.

En esta opción vamos a seleccionar el tipo de factura si está pagada o pendiente.

10. Button ID.-btnRegistrar.

En este botón nos va a guardar después de que hayamos seleccionados todos los campos para realizar la matricula.

Figura 43: *Diseño de la interfaz general del Administrador para estudiantes matriculados.*

1. TextBox ID.-txtNombreEstudiante.

En este campo vamos a ingresar el nombre del estudiante que hayamos registrado para verificar si está correctamente ingresado.

2. Button ID.-btnBuscar.

Este botón va a buscar todos los registros almacenados con ese nombre que ingresemos.

3. GridView ID.-dgvEStudianteRepresentado.

En este gridview se podrá visualizar el listado de registros de estudiantes matriculados

5.03. Especificación de pruebas de Unidad.

Una prueba de unidad pretende probar cada función en un archivo de programa simple (una clase en terminología de objetos). Las librerías de pruebas de unidad formalizan este trabajo al proporcionar clases para pruebas. La prueba de unidad ayuda a que el módulo se haga independiente, quiere decir que un módulo que tiene una prueba de unidad se puede probar independientemente del resto del sistema.

Tabla 18

Prueba de interface de Usuario (estándares).

Identificador de la Prueba:	PRU_UNI01
Método a Probar	**Interface**
Objetivo de la Prueba	Examinar las posibles fallas en el manejo de la interface y corregirlos, revisar estándares para facilitar la navegación del usuario.
Datos de Entrada:	
Ingresar un usuario Ingresar una contraseña Valida el usuario Validar la contraseña	
Resultados Esperados	
Validación del usuario correcto Validación de la contraseña	
Comentarios	
Después de realizar las respectivas pruebas del caso se obtuvo como resultado validaciones correctas de usuario y contraseña.	

Tabla 19

Pruebas de Reportes, Resultados eficientes.

Identificador de la Prueba:	PRU_UNI02
Método a Probar	**Reportes**
Objetivo de la Prueba	Realizar una inspección que todos los resultados esperados en el proceso sean los adecuados y correctos.
Datos de Entrada:	
Ingreso de información de estudiantes	
Ingreso de información de representante	
Resultados Esperados	
Generar un reporte de estudiantes correcto	
Generar un reporte de representante correcto	
Comentarios	
Después que se realizó las pruebas se obtuvo un error en la ortografía	

Tabla 20

Pruebas de compilación de Código.

Identificador de la Prueba:	PRU_UNI03
Método a Probar	**Pruebas de Código – Compilación**
Objetivo de la Prueba	Evaluar los resultados obtenidos y analizar los errores del código encontrados
Datos de Entrada:	
Compilar los formularios después de realizar su correcto llenado	
Resultados Esperados	
Mantener el sistema en ejecución, corregir los errores al compilar.	
Comentarios	
Se realizó un analizó del error y se verifico el error dentro del proceso.	

Tabla 21 *Pruebas de Almacenamiento de datos en la Base.*

Identificador de la Prueba:	PRU_UNI04
Método a Probar	**Almacenamiento de datos en la base**
Objetivo de la Prueba	Realizar una evaluación si los datos ingresados en los mantenimientos se han manejado de manera correcta.
Datos de Entrada:	
Realizar un llenado con datos correctos en los formularios para realizarle un mantenimiento.	
Resultados Esperados	
Se verifico el ingreso de los datos se realizó correctamente.	
Comentarios	
Esta prueba indico que los mantenimientos se realizaron correctamente.	

5.04. Especificación de Pruebas de Aceptación.

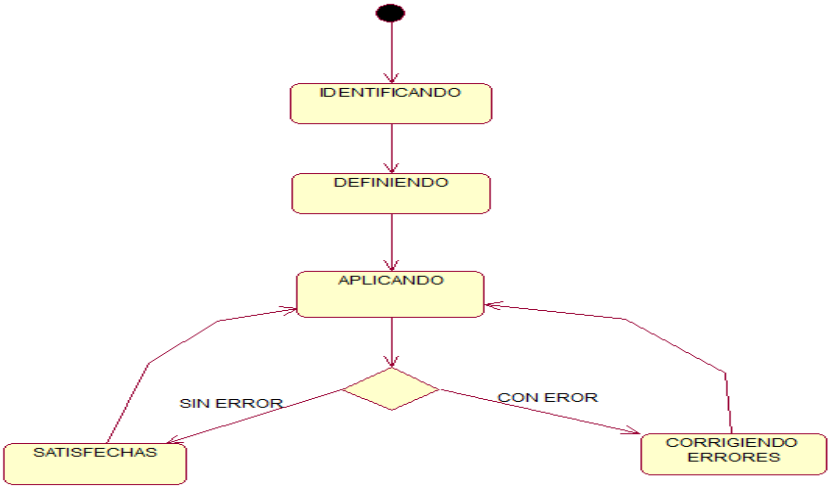


Figura 44: *Diagrama de secuencia del proceso de pruebas de aceptación.*

Tabla 22

Detalle de pruebas de aceptación en la creación de Periodos, Cursos, Paralelos, Especialidades.

Identificador de la Prueba:	PRU_ACE02
Caso de Uso	**Periodos, Cursos, Paralelos, Especialidades CU002**
Tipo de Usuario	**Administrador**
Objetivo de la Prueba	Probar el funcionamiento del proceso general de Periodos, Cursos y Paralelos.
Secuencia de Eventos	
Login de usuario, ingresar Periodos, Cursos, Paralelos, Especialidades, Modificar, Eliminar, Guardar.	
Login de usuario, Cursos, Especialidades , Paralelos, sacar reportes, cerrar sesión	
Resultados Esperados	
Que no tenga inconsistencias con respecto a guardados, validaciones y seguridad en la información.	
Comentarios	
Se realiza las pruebas en el sistema ingresando registros nuevos, consulta y reportes.	
Estado Aceptado/No aceptado	
Aceptado al revisar que los datos se ingresaron correctamente.	

Tabla 23

Detalle de pruebas de aceptación en la creación de áreas de Conocimiento

Identificador de la Prueba:	PRU_ACE03
Caso de Uso	**Áreas de conocimiento CU003**
Tipo de Usuario	**Administrador**
Objetivo de la Prueba	Probar el funcionamiento del proceso general del área de Conocimiento
Secuencia de Eventos	
Login de usuario, ingresar materias, Guardar, Modificar, Eliminar.	
Login de usuario, consultar materias, sacar reportes, cerrar sesión	
Resultados Esperados	
Que no tenga inconsistencias con respecto a guardados, validaciones y seguridad en la información.	
Comentarios	
Se realiza las pruebas en el sistema ingresando registros nuevos, eliminando, modificando, consultando y sacando reportes.	
Estado Aceptado/No aceptado	
Aceptado al observar que los reportes se generaron correctamente.	

Tabla 24

Detalle de pruebas de aceptación en el proceso de Matriculación.

Identificador de la Prueba:	PRU_ACE04
Caso de Uso	**Gestión de Matrícula CU004**
Tipo de Usuario	**Administrador/Usuario**
Objetivo de la Prueba	Probar el funcionamiento del proceso general de matriculación.
Secuencia de Eventos	
Login de usuario, ingresar datos generales del estudiante al Matricularse.	
Resultados Esperados	
Que no tenga inconsistencias con respecto a guardados, validaciones y seguridad en la información de la matrícula.	
Comentarios	
Se realiza las pruebas en el sistema ingresando registros nuevos, eliminando, modificando, consultando y sacando reportes.	
Estado Aceptado/No aceptado	
Aceptado al observar que el estudiante se matriculo correctamente.	

5.05 Especificación de Pruebas de Carga.

El servicio de pruebas de rendimiento de software se centra en determinar la velocidad con la que el sistema bajo pruebas realiza una tarea en las condiciones particulares del escenario de pruebas. Este servicio ayuda a su organización a detectar los cuellos de botella de su aplicación, antes de que, sus usuarios sufran un mal rendimiento, con la consecuente pérdida económica y frustración de sus clientes o empleados.

Estas pruebas no pretenden optimizar todos estos factores sino sólo medir el rendimiento de las aplicaciones entregadas en su ubicación establecida.

Los tipos de pruebas de rendimiento que habitualmente pueden ponerse en marcha son los siguientes:

Prueba normal. Permite establecer los tiempos medios de respuesta cuando sólo un usuario está conectado a la aplicación.

Esta prueba pretende establecer una referencia futura para posteriores comparaciones así como medir unitariamente el software entregado.

Prueba con número mínimo de usuarios. Se realizan las pruebas del sistema con el número de usuarios mínimos concurrentes establecido.

Prueba con número máximo de usuarios. Se realizan las pruebas del sistema con el número de usuarios máximo concurrentes establecido.

Prueba de número máximo soportado de usuarios. Se busca encontrar cuál es el límite del sistema.

Tabla 25

Detalle de un tipo de prueba de carga más baja.

Identificador de la Prueba:	PRCA01
Tipo de Prueba	Prueba normal (Prueba de Carga)
Objetivo de la Prueba	Establecer los tiempos medios de respuesta cuando sólo un usuario está conectado a la aplicación.
Descripción:	
Esta prueba pretende establecer una referencia futura para posteriores comparaciones así como medir unitariamente el software entregado.	
Resultados Esperados	
Hacer que los procesos del sistema sean óptimos y tengan buenos tiempos de respuesta.	
Comentarios	
En esta prueba se realiza pruebas de tiempo para verificar su demora en responder a las consultas con un solo usuario.	

Tabla 26 *Detalle de un tipo de prueba de carga con un número mínimo de usuarios.*

Identificador de la Prueba:	PRCA02
Tipo de Prueba	**Prueba con número mínimo de usuarios**
Objetivo de la Prueba	Conocer si los procesos se están efectuando con normalidad y sin problemas ya con algunos usuarios.
Descripción:	
Se realizan las pruebas del sistema con el número de usuarios mínimos concurrentes establecido.	
Resultados Esperados	
Validar la funcionalidad del sistema con un mínimo de usuarios logueados.	
Comentarios	
En esta prueba se realizó consultas conectados varios usuarios y medimos el tiempo que se demora en responder el sistema.	

Véase Anexo A.04.

Véase Anexo A.05.

Tabla 27

Detalle de un tipo de prueba de carga con un número máximo de usuarios.

Identificador de la Prueba:	PRCA03
Tipo de Prueba	**Prueba con número máximo de usuarios**
Objetivo de la Prueba	Establecer los tiempos de respuesta cuando una gran cantidad de usuarios están conectados a la aplicación.
Descripción:	
Se realizan las pruebas del sistema con el número de usuarios máximo concurrentes establecido	
Resultados Esperados	
Hacer que los procesos del sistema sean óptimos y tengan buenos tiempos de respuesta cuando tengamos una gran cantidad de usuarios conectados en el Sistema.	
Comentarios	
Se realizó prueba conectados varios usuarios para medir el tiempo de respuesta de sus consultas.	

5.06 Configuración del Ambiente Mínima/Ideal.

Para casos de Aplicaciones desarrollados en un ambiente cliente servidor, un cuarto de máquinas envuelve los siguientes factores a tomar en consideración: localización, diseño, hardware, software, fuente de energía, temperatura, humedad, recuperación de desastres, seguridad. El planeamiento adecuado seguro de la localización y el diseño particular son los primeros pasos para crear un ambiente seguro.

Otras consideraciones a tener en cuenta son los insumos básicos con lo que la sala de servidores debe contar (enchufes, espacio, disponibilidad de red).

Un punto importante a considerar en cuanto al ambiente de la sala de servidores, es la humedad. Altos niveles de humedad pueden causar condensación y bajos niveles

pueden causar electrostática. Adicionalmente, la sala de servidores debe contar con detectores de humo y agua, así como también UPS para proteger a los equipos de cortes/picos de electricidad.

Requerimientos Mínimos

Requerimientos de Hardware

Equipo Servidor

- ✓ Procesador: Intel Core 2 Dúo de 2 GHz ó 3Ghz
- ✓ Memoria RAM 4 GB. La cantidad de memoria está establecida para 15 usuarios usando el sistema simultáneamente.
- ✓ Disco Duro con 100 Gb LIBRES para datos

Equipo Cliente

- ✓ Procesador: Intel Core 2 Dúo de 2GHz.
- ✓ Memoria RAM 2 Gb.
- ✓ Disco Duro: 50 Gb de memoria libre son suficientes para alojar archivos temporales y reportes, además para instalación de navegadores Web

Requerimientos de Software

Equipo Servidor

- ✓ Sistemas Operativos
- ✓ Sistema Operativo Windows Server 2008 o superior
- ✓ Manejadores de Base de Datos
- ✓ MySQL 5.1 o superior
- ✓ Office 2007 para visualización de reportes y estadísticas.

Requerimientos recomendados

- ✓ Los requerimientos recomendados listados a continuación determinaran un buen funcionamiento del sistema, en la ejecución del mismo y en los procesos que realiza.

Hardware

- ✓ Procesador: Intel Core i3 de 2GHz.
- ✓ Memoria RAM: 4 Gb.
- ✓ Disco Duro: 50 Gb de memoria libre son suficientes para alojar archivos temporales y reportes, además para instalación de navegadores Web.
- ✓ Pantalla de 17 pulgadas para una buena visualización.

Software

- ✓ Sistema operativo: Windows 7 o superior.
- ✓ Navegador Web: Firefox o Google Chrome.

Capítulo VI: Aspectos Administrativos

6.01 Recursos.

Recursos Humanos

- Tutor del Proyecto: Ing. Hugo Heredia
- Lector: Ing. Diana Terán.
- Promotor: Victor Remache

Recursos Materiales.

- Manuales Físicos
- Computador
- Servicios Basicos
- Alimentación
- Transporte
- Manuales del Ministerio de Educación

Recurso Técnico

- Visual Studio 2010
- Asp.Net
- Rational Rose
- Power Designer
- Adobe Dreamweaver CS6
- Microsoft Office 2013
- Microsoft Project 2013
- Microsoft Excel 2013

6.02 Presupuesto.

Tabla 28 *Detalle de Gastos realizados en el Proyecto.*

PRESUPUESTO DE GASTOS			
RUBROS	**CANTIDAD**	**PRECIO UNITARIO**	**SUB TOTAL**
1. Bienes			
Hojas	**1000**	**$ 0.02**	**$ 20**
Cartuchos	**4**	**$ 8**	**$ 32**
Lápices	**2**	**$ 0.35**	**$ 0.7**
CD´s	**10**	**$ 0.6**	**$ 6**
Carpetas	**6**	**$ 0.5**	**$ 3**
		TOTAL	**$ 61.7**
2. Servicios			
Movilidad			**$ 80**
Impresiones			**$ 60**
Teléfono			**$ 15**
Fotocopias			**$ 40**
Luz			**$ 30**
Agua			**$ 25**
		TOTAL	**$ 250**
		RUBRO TOTAL	**$ 311.7**

6.03 Cronograma.

Para poder comprender a la perfección el significado del término que nos ocupa, cronograma, es importante que, en primer lugar, procedamos a establecer su origen etimológico. Al hacerlo descubrimos que emana del griego, ya que se encuentra conformado por dos vocablos que lo son: el sustantivo "chronos", que puede traducirse como "tiempo", y la palabra "grama", que es equivalente a "mensaje escrito".

Cronograma es un concepto que se utiliza en varios países latinoamericanos para mencionar a un calendario de trabajo o de actividades

Véase Anexo A.06.

Capítulo VII: Conclusiones y Recomendaciones

7.01 Conclusiones.

El Proyecto de Tesis planteado en este documento nos ha brindado la oportunidad de desarrollar una Aplicación Web de Administración de estudiantes, así como implementar los conocimientos en el análisis y desarrollo de Software adquiridos a lo largo de toda nuestra carrera, profundizando en la investigación.

Se ha esta cumplido satisfactoriamente el desarrollo de este proyecto, permitiendo así a las instituciones de particulares de Nivel Medio Bachillerato, automatizar su funcionalidad y dando así un mejor servicio y atención a todos sus usuarios.

Una de sus principales ventajas el desarrollo de esa aplicación está orientada a la web para la Administración de Estudiantes, en la instalación y la accesibilidad, bastara con solo tener instalado en un computador con acceso a Internet cualquier versión actualizada de Google Chrome, o cualquier otro navegador actual, la accesibilidad al Sistema Escolástico será inmediata desde el Internet mediante la validación de los respectivos datos de cada usuario.

El diseñó cuenta con una interface amigable e interactiva de acuerdo a los requerimientos indicados por los beneficiarios. Si bien es cierto muchas de las configuraciones han requerido de mucho tiempo, pero nos ha dado buen resultado ya que por medio de este proyecto hemos adquirido nuevas técnicas de desarrollo.

7.02 Recomendaciones.

- Para un correcto funcionamiento del proceso del sistema, todos y cada uno de los estudiantes inmersos en la solución, deben tener registrado su usuario y Contraseña, ya que mediante estos registros pueden validar sus datos para poder ingresar al sistema a verificar su información.
- La persona encargada de la administración del sistema debe tener conocimientos en informática, ya que para realizar algún cambio y luego ponerlo a producción se debe tener los conocimientos necesarios y así permitir que el sistema continúe funcionando correctamente.
- Dar una charla a los alumnos y padres de familia acerca de sistema.
- Capacitar a los Alumnos en el manejo de la aplicación explicándoles las bondades y los beneficios que ofrece.
- Realizar actualizaciones de datos periódicamente

ANEXOS

Véase Anexo A.01. ***Diagrama de Caso de Uso.***

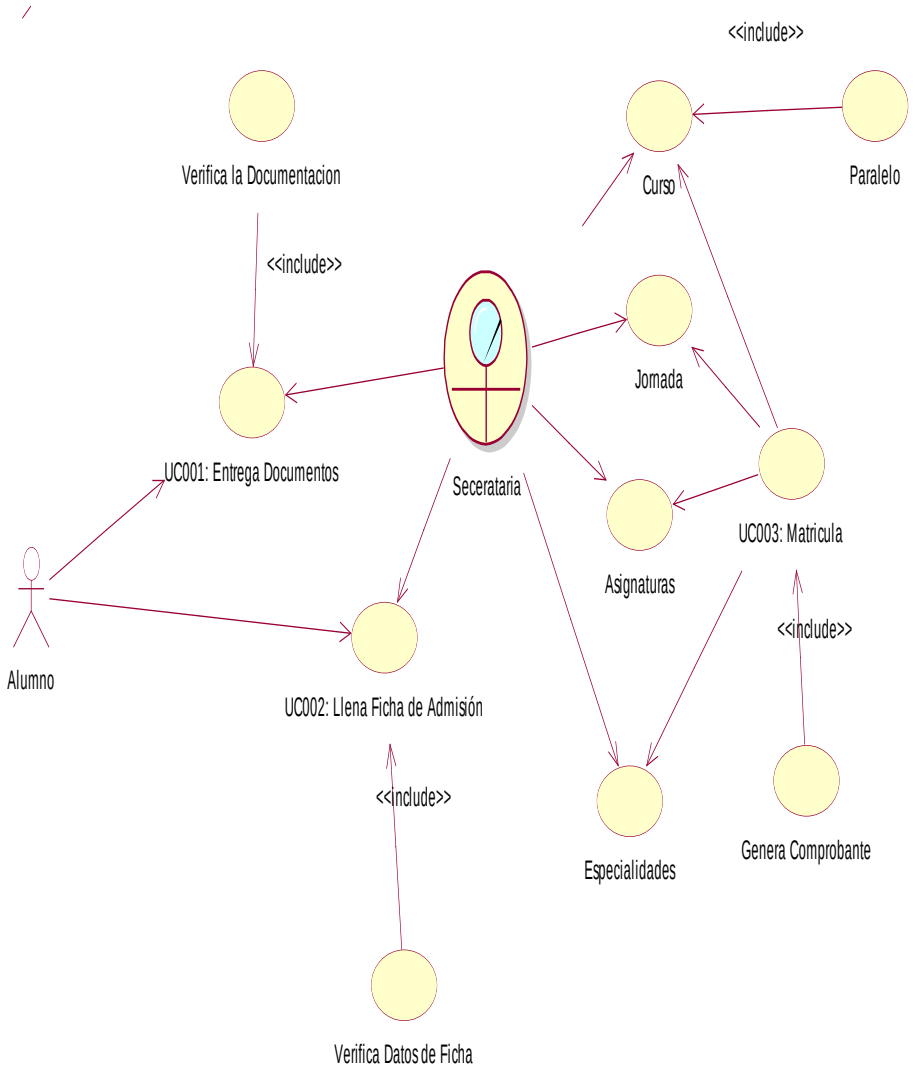


Figura 45: *Diagrama de caso de uso en este diagrama se realizó el desarrollo del proceso que tiene que realizar un estudiante que quera ingresar a una unidad educativa.*

Véase Anexo A.02. **Diagrama de Clases**

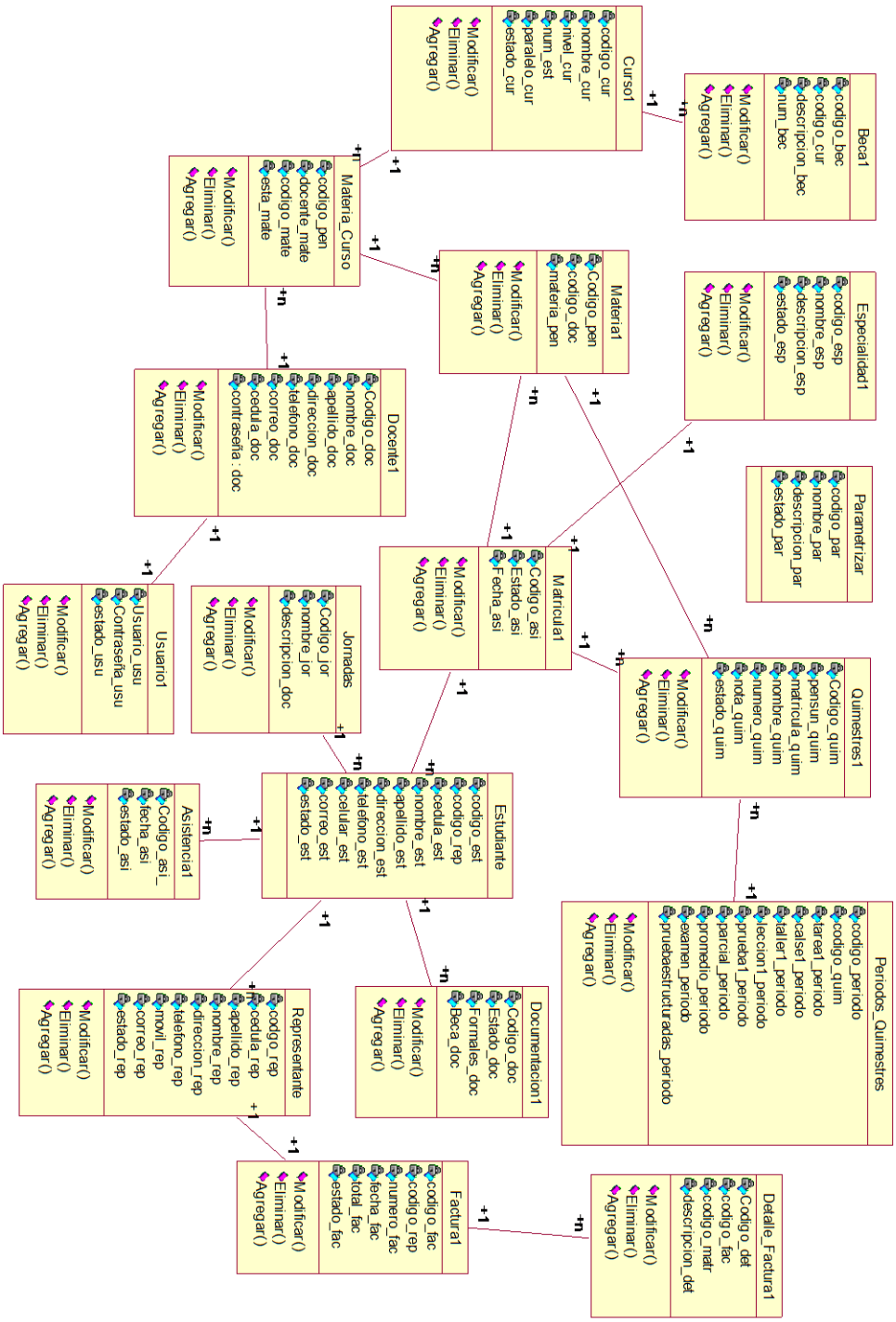


Figura 46: *Donde se especifica los atributos y acciones del sistema.*

Véase Anexo A.03. **Modelo Físico.**

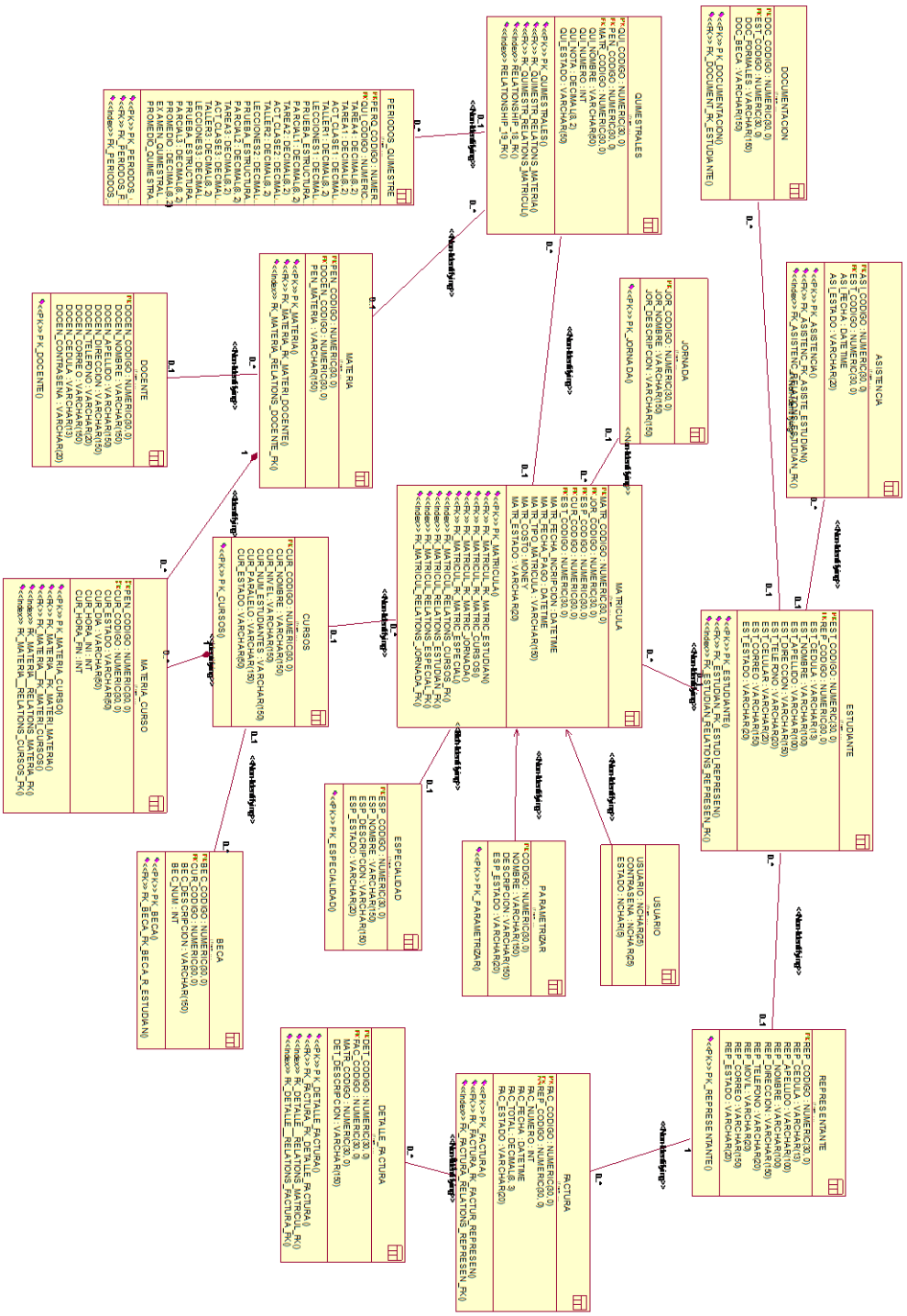


Figura 457: *Donde se genera las tablas en base a los diagramas anteriores.*

Véase Anexo A.04. **Prueba de Carga.**

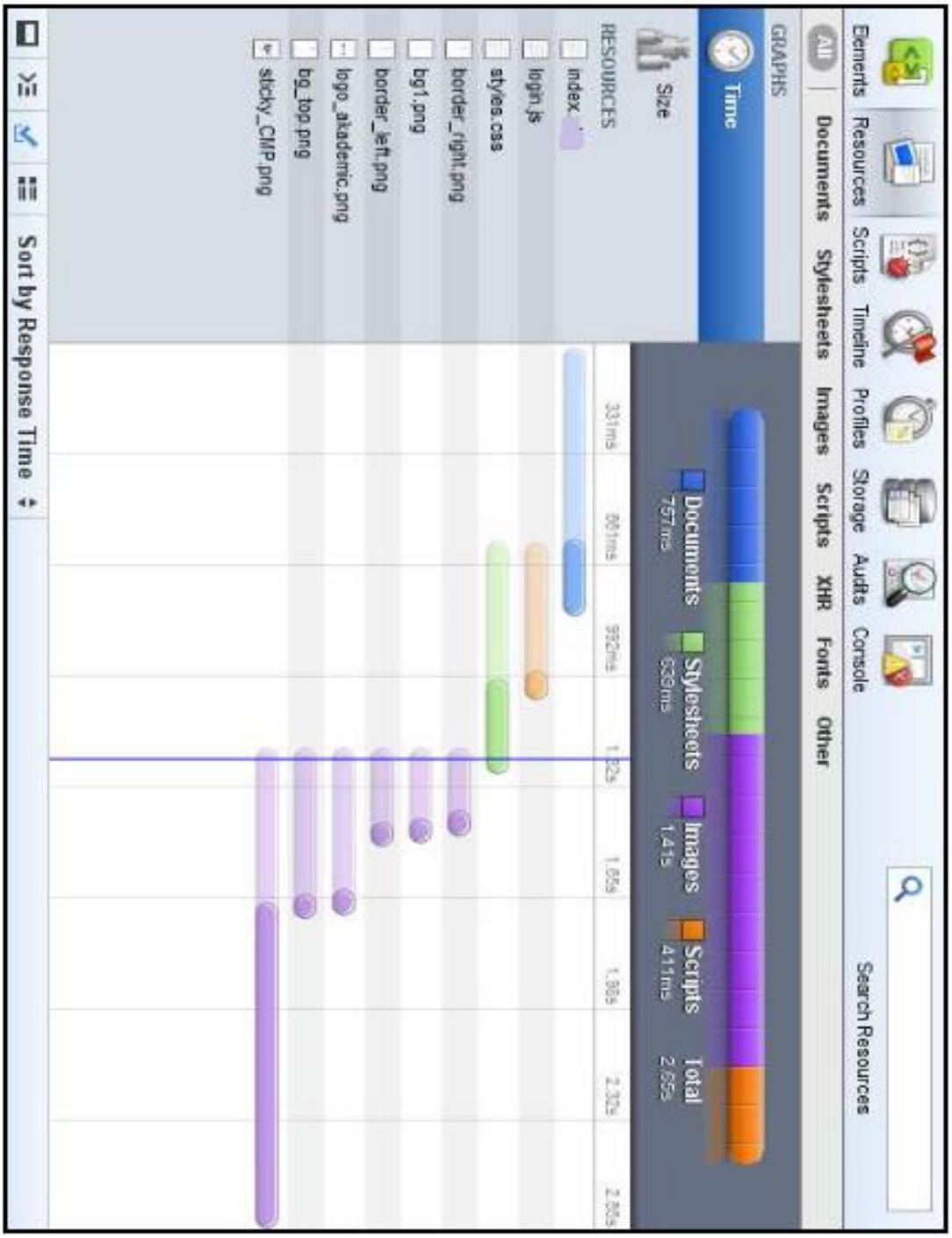


Figura 47: *Detalle de las pruebas de carga. Esta prueba nos permitió verificar el tiempo respuesta al cargar el sistema al ingresar un usuario.*

Véase Anexo A.05. **Prueba de Carga de Aceptación.**

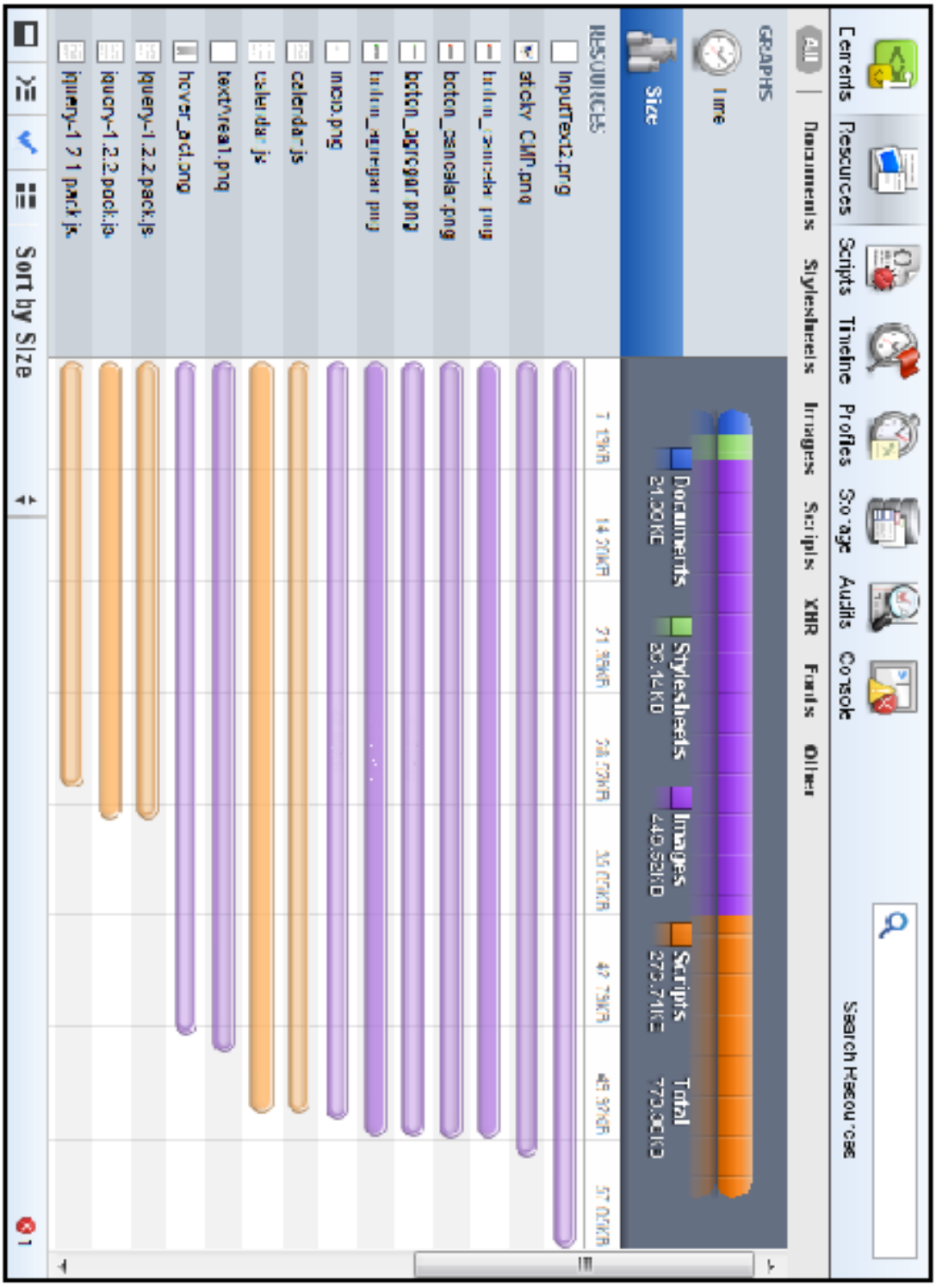


Figura 48: *Essta prueba nos permite tomar el tiempo que se tarda en aceptar el login ingresado.*

Véase Anexo A.0.6. **Cronograma.**

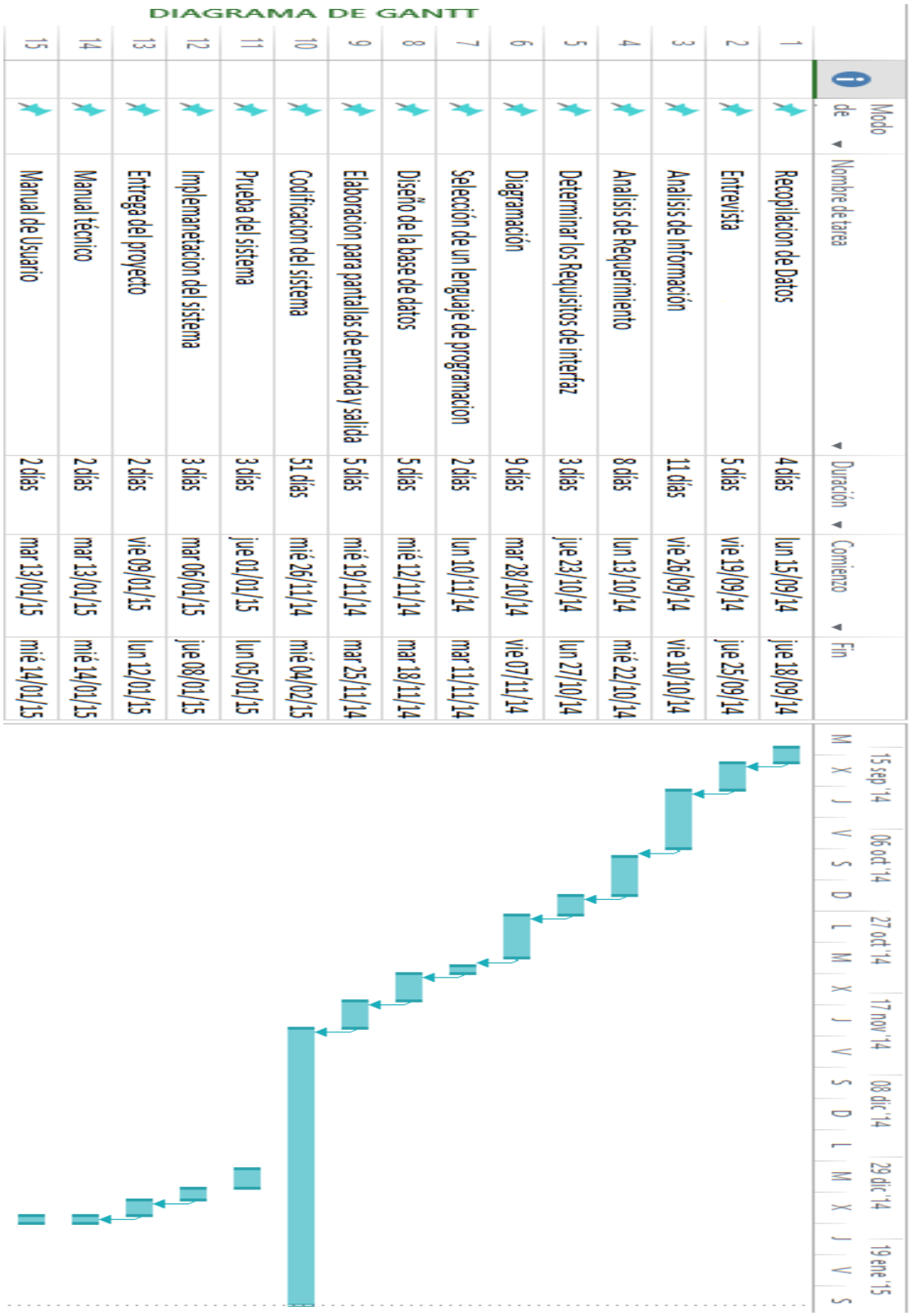


Figura 46: *Detalle del Cronograma de actividades.*

SCRIPT BASE DE DATOS

```
CREATE TABLE [dbo].[REPRESENTANTE](
    [REP_CODIGO] [numeric](30, 0) IDENTITY(1,1) NOT NULL,
    [REP_CEDULA] [varchar](13) NULL,
    [REP_APELLIDO] [varchar](100) NULL,
    [REP_NOMBRE] [varchar](100) NULL,
    [REP_DIRECCION] [varchar](150) NULL,
    [REP_TELEFONO] [varchar](20) NULL,
    [REP_MOVIL] [varchar](20) NULL,
    [REP_CORREO] [varchar](150) NULL,
    [REP_ESTADO] [varchar](20) NULL,
 CONSTRAINT [PK_REPRESENTANTE] PRIMARY KEY NONCLUSTERED

CREATE TABLE [dbo].[CURSOS](
    [CUR_CODIGO] [numeric](30, 0) IDENTITY(1,1) NOT NULL,
    [CUR_NOMBRE] [varchar](150) NULL,
    [CUR_NIVEL] [varchar](150) NULL,
    [CUR_NUM_ESTUDIANTES] [varchar](150) NULL,
    [CUR_PARALELO] [varchar](150) NULL,
    [CUR_ESTADO] [varchar](50) NULL,
 CONSTRAINT [PK_CURSOS] PRIMARY KEY NONCLUSTERED

CREATE TABLE [dbo].[DOCENTE](
    [DOCEN_CODIGO] [numeric](30, 0) IDENTITY(1,1) NOT NULL,
    [DOCEN_NOMBRE] [varchar](150) NULL,
    [DOCEN_APELLIDO] [varchar](150) NULL,
    [DOCEN_DIRECCION] [varchar](150) NULL,
    [DOCEN_TELEFONO] [varchar](20) NULL,
    [DOCEN_CORREO] [varchar](150) NULL,
    [DOCEN_CEDULA] [varchar](13) NULL,
    [DOCEN_CONTRASENA] [varchar](20) NULL,
 CONSTRAINT [PK_DOCENTE] PRIMARY KEY NONCLUSTERED

CREATE TABLE [dbo].[ESPECIALIDAD](
```

```
    [ESP_CODIGO] [numeric](30, 0) IDENTITY(1,1) NOT NULL,
    [ESP_NOMBRE] [varchar](150) NULL,
    [ESP_DESCRIPCION] [varchar](150) NULL,
    [ESP_ESTADO] [varchar](20) NULL,
CONSTRAINT [PK_ESPECIALIDAD] PRIMARY KEY NONCLUSTERED

CREATE TABLE [dbo].[PARAMETRIZAR](
    [CODIGO] [numeric](30, 0) IDENTITY(1,1) NOT NULL,
    [NOMBRE] [varchar](150) NULL,
    [DESCRIPCION] [varchar](150) NULL,
    [ESP_ESTADO] [varchar](20) NULL,
CONSTRAINT [PK_PARAMETRIZAR] PRIMARY KEY NONCLUSTERED

CREATE TABLE [dbo].[JORNADA](
    [JOR_CODIGO] [numeric](30, 0) IDENTITY(1,1) NOT NULL,
    [JOR_NOMBRE] [varchar](150) NULL,
    [JOR_DESCRIPCION] [varchar](150) NULL,
CONSTRAINT [PK_JORNADA] PRIMARY KEY NONCLUSTERED
CREATE TABLE [dbo].[USUARIO](
    [USUARIO] [nchar](50) NULL,
    [CONTRASENA] [nchar](50) NULL,
    [ESTADO] [nchar](10) NULL)

CREATE TABLE [dbo].[MATERIA](
    [PEN_CODIGO] [numeric](30, 0) IDENTITY(1,1) NOT NULL,
    [DOCEN_CODIGO] [numeric](30, 0) NULL,
    [PEN_MATERIA] [varchar](150) NULL,
CONSTRAINT [PK_MATERIA] PRIMARY KEY NONCLUSTERED

CREATE TABLE [dbo].[FACTURA](
    [FAC_CODIGO] [numeric](30, 0) IDENTITY(1,1) NOT NULL,
    [REP_CODIGO] [numeric](30, 0) NOT NULL,
    [FAC_NUMERO] [int] NULL,
    [FAC_FECHA] [datetime] NULL,
```

```
    [FAC_TOTAL] [decimal](8, 3) NULL,
    [FAC_ESTADO] [varchar](20) NULL,
CONSTRAINT [PK_FACTURA] PRIMARY KEY NONCLUSTERED(

CREATE TABLE [dbo].[ESTUDIANTE](
    [EST_CODIGO] [numeric](30, 0) IDENTITY(1,1) NOT NULL,
    [REP_CODIGO] [numeric](30, 0) NULL,
    [EST_CEDULA] [varchar](13) NULL,
    [EST_NOMBRE] [varchar](100) NULL,
    [EST_APELLIDO] [varchar](100) NULL,
    [EST_DIRECCION] [varchar](150) NULL,
    [EST_TELEFONO] [varchar](20) NULL,
    [EST_CELULAR] [varchar](20) NULL,
    [EST_CORREO] [varchar](150) NULL,
    [EST_ESTADO] [varchar](20) NULL,
CONSTRAINT [PK_ESTUDIANTE] PRIMARY KEY NONCLUSTERED

CREATE TABLE [dbo].[BECA](
    [BEC_CODIGO] [numeric](30, 0) IDENTITY(1,1) NOT NULL,
    [CUR_CODIGO] [numeric](30, 0) NULL,
    [BEC_DESCRIPCION] [varchar](150) NULL,
    [BEC_NUM] [int] NULL,
CONSTRAINT [PK_BECA] PRIMARY KEY NONCLUSTERED

CREATE TABLE [dbo].[DOCUMENTACION](
    [DOC_CODIGO] [numeric](30, 0) IDENTITY(1,1) NOT NULL,
    [EST_CODIGO] [numeric](30, 0) NULL,
    [DOC_FORMALES] [varchar](150) NULL,
    [DOC_BECA] [varchar](150) NULL,
CONSTRAINT [PK_DOCUMENTACION] PRIMARY KEY NONCLUSTERED

CREATE TABLE [dbo].[ASISTENCIA](
    [ASI_CODIGO] [numeric](30, 0) IDENTITY(1,1) NOT NULL,
    [EST_CODIGO] [numeric](30, 0) NULL,
```

```
    [ASI_FECHA] [datetime] NULL,
    [ASI_ESTADO] [varchar](20) NULL,
 CONSTRAINT [PK_ASISTENCIA] PRIMARY KEY NONCLUSTERED
CREATE TABLE [dbo].[DETALLE_FACTURA](
    [DET_CODIGO] [numeric](30, 0) IDENTITY(1,1) NOT NULL,
    [FAC_CODIGO] [numeric](30, 0) NULL,
    [MATR_CODIGO] [numeric](30, 0) NULL,
    [DET_DESCRIPCION] [varchar](150) NULL,
 CONSTRAINT [PK_DETALLE_FACTURA] PRIMARY KEY CONCLUSTERED

CREATE TABLE [dbo].[MATERIA_CURSO](
    [PEN_CODIGO] [numeric](30, 0) NOT NULL,
    [CUR_CODIGO] [numeric](30, 0) NOT NULL,
    [CUR_ESTADO] [varchar](50) NULL,
    [CUR_DIA] [varchar](50) NULL,
    [CUR_HORA_INI] [int] NULL,
    [CUR_HORA_FIN] [int] NULL,
 CONSTRAINT [PK_MATERIA_CURSO] PRIMARY KEY NONCLUSTERED

CREATE TABLE [dbo].[MATRICULA](
    [MATR_CODIGO] [numeric](30, 0) IDENTITY(1,1) NOT NULL,
    [JOR_CODIGO] [numeric](30, 0) NULL,
    [ESP_CODIGO] [numeric](30, 0) NULL,
    [CUR_CODIGO] [numeric](30, 0) NULL,
    [EST_CODIGO] [numeric](30, 0) NULL,
    [MATR_FECHA_INCRIPCION] [datetime] NULL,
    [MATR_FECHA_PAGO] [datetime] NULL,
    [MATR_TIPO_MATRICULA] [varchar](150) NULL,
    [MATR_COSTO] [money] NULL,
    [MATR_ESTADO] [varchar](20) NULL,
 CONSTRAINT [PK_MATRICULA] PRIMARY KEY

CREATE TABLE [dbo].[QUIMESTRALES](
    [QUI_CODIGO] [numeric](30, 0) IDENTITY(1,1) NOT NULL,
```

```
    [PEN_CODIGO] [numeric](30, 0) NULL,
    [MATR_CODIGO] [numeric](30, 0) NULL,
    [QUI_NOMBRE] [varchar](50) NULL,
    [QUI_NUMERO] [int] NULL,
    [QUI_NOTA] [decimal](8, 2) NULL,
    [QUI_ESTADO] [varchar](50) NULL,
CONSTRAINT [PK_QUIMESTRALES] PRIMARY KEY NONCLUSTERED

CREATE TABLE [dbo].[PERIODOS_QUIMESTRE](
    [PERQ_CODIGO] [numeric](30, 0) IDENTITY(1,1) NOT NULL,
    [QUI_CODIGO] [numeric](30, 0) NULL,
    [TAREA4] [decimal](8, 2) NULL,
    [TAREA1] [decimal](8, 2) NULL,
    [ACT_CLASE1] [decimal](8, 2) NULL,
    [TALLER1] [decimal](8, 2) NULL,
    [LECCIONES1] [decimal](8, 2) NULL,
    [PRUEBA_ESTRUCTURADA1] [decimal](8, 2) NULL,
    [PARCIAL1] [decimal](8, 2) NULL,
    [TAREA2] [decimal](8, 2) NULL,
    [ACT_CLASE2] [decimal](8, 2) NULL,
    [TALLER2] [decimal](8, 2) NULL,
    [LECCIONES2] [decimal](8, 2) NULL,
    [PRUEBA_ESTRUCTURADA2] [decimal](8, 2) NULL,
    [PARCIAL2] [decimal](8, 2) NULL,
    [TAREA3] [decimal](8, 2) NULL,
    [ACT_CLASE3] [decimal](8, 2) NULL,
    [TALLER3] [decimal](8, 2) NULL,
    [LECCIONES3] [decimal](8, 2) NULL,
    [PRUEBA_ESTRUCTURADA3] [decimal](8, 2) NULL,
    [PARCIAL3] [decimal](8, 2) NULL,
    [PROMEDIO] [decimal](8, 2) NULL,
    [EXAMEN_QUIMESTRAL] [decimal](8, 2) NULL,
    [PROMEDIO_QUIMESTRAL] [decimal](8, 2) NULL,
CONSTRAINT [PK_PERIODOS_QUIMESTRE] PRIMARY KEY NONCLUSTERED
```

```
****************************** VISTAS ********************************
create view [dbo].[representanteView]
as
Select
r.REP_CEDULA,r.REP_APELLIDO,r.REP_NOMBRE,r.REP_TELEFONO,r.REP_MOVIL
,r.REP_DIRECCION,r.REP_CORREO  from Representante r

create view [dbo].[facturaDescendente]
as
select top(1)F.* from FACTURA F order by FAC_CODIGO desc

create view [dbo].[listarEstudiantesAllMatricula]
as
select * from ESTUDIANTE E WHERE E.EST_ESTADO='Activo'
GO

CREATE VIEW [dbo].[MATERIA_DOCENTE]
AS
select D.DOCEN_APELLIDO,D.DOCEN_NOMBRE,M.PEN_MATERIA from MATERIA
M, DOCENTE D WHERE M.DOCEN_CODIGO=D.DOCEN_CODIGO
GO


CREATE VIEW [dbo].[maximaAsistencia]
as
SELECT TOP(1) A.EST_CODIGO,count(A.ASI_FECHA)AS 'ASISTEN' FROM
ASISTENCIA A  GROUP BY A.EST_CODIGO ORDER BY ASISTEN DESC
GO

CREATE VIEW [dbo].[ESTUDIANTESMETRICULADOS]
AS
select
ES.EST_CEDULA,ES.EST_APELLIDO,ES.EST_NOMBRE,J.JOR_NOMBRE,E.ESP_NO
```

```
MBRE,C.CUR_NIVEL+' '+C.CUR_PARALELO AS
'CURSO',M.MATR_FECHA_INCRIPCION,M.MATR_COSTO
 from MATRICULA M,JORNADA J,ESPECIALIDAD E,CURSOS C,ESTUDIANTE ES
WHERE
 M.CUR_CODIGO=C.CUR_CODIGO AND M.ESP_CODIGO=E.ESP_CODIGO AND
M.EST_CODIGO=ES.EST_CODIGOAND M.JOR_CODIGO= J.JOR_CODIGO
GO
create view [dbo].[ultimaMatriculaReporte]
as
select TOP(1)M.*,R.REP_CODIGO,DT.DET_CODIGO FROM MATRICULA
M,ESTUDIANTE E, REPRESENTANTE R,DETALLE_FACTURA DT WHERE
E.REP_CODIGO=R.REP_CODIGO AND M.EST_CODIGO=E.EST_CODIGO AND
DT.MATR_CODIGO=M.MATR_CODIGO order by DT.DET_CODIGO desc
GO

CREATE VIEW [dbo].[LISTANOTASFINALES]
AS
SELECT DISTINCT D.DOCEN_CEDULA,D.DOCEN_APELLIDO+'
'+D.DOCEN_NOMBRE AS 'DOCENTE',E.EST_CEDULA, E.EST_APELLIDO+' '+
E.EST_NOMBRE AS 'ESTUDIANTE', (SELECT
SUM(A.QUI_NOTA)/COUNT(A.PEN_CODIGO) FROM QUIMESTRALES A WHERE
A.MATR_CODIGO=Q.MATR_CODIGO )AS 'PROMEDIO_GENERAL'
,CASE WHEN (SELECT COUNT (A.PEN_CODIGO) FROM QUIMESTRALES A
WHERE   A.QUI_ESTADO='REPROBADO' )>0
THEN 'REPROBADO' ELSE 'APROBADO' END AS 'ESTADO',C.CUR_NIVEL
FROM ESTUDIANTE E, MATRICULA M,QUIMESTRALES Q,MATERIA
MAT,DOCENTE D,CURSOS C
WHERE M.EST_CODIGO=E.EST_CODIGO AND
M.MATR_CODIGO=Q.MATR_CODIGO AND MAT.PEN_CODIGO=Q.PEN_CODIGO
AND MAT.DOCEN_CODIGO=D.DOCEN_CODIGO AND
C.CUR_CODIGO=M.CUR_CODIGO
GO

create view [dbo].[estudiantesPorDocente]
```

```
as
select DISTINCT E.*,D.DOCEN_CEDULA from ESTUDIANTE E,MATRICULA
M,QUIMESTRALES Q, MATERIA MAT,DOCENTE D
WHERE E.EST_CODIGO=M.EST_CODIGO AND
M.MATR_CODIGO=Q.MATR_CODIGO AND Q.PEN_CODIGO=MAT.PEN_CODIGO
AND MAT.DOCEN_CODIGO=D.DOCEN_CODIGO
GO

CREATE VIEW [dbo].[CALIFICACIONESPORESTUDIANTE]
 AS
SELECT     MAT.PEN_MATERIA, Q.QUI_NOMBRE, P.PERQ_CODIGO,
P.QUI_CODIGO, P.TAREA4, P.TAREA1, P.ACT_CLASE1, P.TALLER1,
P.LECCIONES1,
                      P.PRUEBA_ESTRUCTURADA1, P.PARCIAL1, P.TAREA2,
P.ACT_CLASE2, P.TALLER2, P.LECCIONES2, P.PRUEBA_ESTRUCTURADA2,
P.PARCIAL2, P.TAREA3,
                      P.ACT_CLASE3, P.TALLER3, P.LECCIONES3,
P.PRUEBA_ESTRUCTURADA3, P.PARCIAL3, P.PROMEDIO,
P.EXAMEN_QUIMESTRAL, P.PROMEDIO_QUIMESTRAL,
                      M.EST_CODIGO,
J.JOR_NOMBRE,CUR.CUR_NIVEL,CUR.CUR_PARALELO
FROM         dbo.ESTUDIANTE AS E INNER JOIN
                      dbo.MATRICULA AS M ON E.EST_CODIGO = M.EST_CODIGO INNER
JOIN
                      dbo.QUIMESTRALES AS Q ON M.MATR_CODIGO = Q.MATR_CODIGO
INNER JOIN
                      dbo.PERIODOS_QUIMESTRE AS P ON Q.QUI_CODIGO =
P.QUI_CODIGO INNER JOIN
                      dbo.MATERIA AS MAT ON Q.PEN_CODIGO = MAT.PEN_CODIGO
INNER JOIN
                      dbo.JORNADA AS J ON M.JOR_CODIGO = J.JOR_CODIGO INNER JOIN
                      dbo.CURSOS AS CUR ON M.CUR_CODIGO = CUR.CUR_CODIGO

********** PORCEDIMIENTOS ALMACENADOS **********
```

```
CREATE PROCEDURE [dbo].[spVerificarCupos]
@codCurso int
as
DECLARE @numNormal int, @numBeca int, @normalCur int,@becaCur int
 SELECT @numNormal=COUNT(M.CUR_CODIGO) FROM MATRICULA M WHERE
M.MATR_TIPO_MATRICULA='NORMAL' AND M.CUR_CODIGO=@codCurso
 SELECT @numBeca=COUNT(M.CUR_CODIGO) FROM MATRICULA M WHERE
M.MATR_TIPO_MATRICULA='BECA' AND M.CUR_CODIGO=@codCurso
 SELECT @normalCur=C.CUR_NUM_ESTUDIANTES, @becaCur=B.BEC_NUM FROM
CURSOS C,BECA B WHERE C.CUR_CODIGO=B.CUR_CODIGO AND
C.CUR_CODIGO=@codCurso

select (@normalCur-@numNormal) as 'NUM_NORMAL',(@becaCur-@numBeca)AS
'NUM_BECA'
PRINT @numNormal
PRINT @numBeca
PRINT @normalCur
PRINT @becaCur
GO

CREATE proc [dbo].[sp_insertar_document_estudiante]
AS BEGIN declare @valorId int
select top(1)@valorId=E.EST_CODIGO from ESTUDIANTE E order by E.EST_CODIGO
desc
CREATE VIEW [dbo].[NOTASESTUDIANTELOGEADO] AS
SELECT DISTINCT D.DOCEN_APELLIDO+' '+D.DOCEN_NOMBRE AS 'DOCENTE',
E.EST_CEDULA,E.EST_APELLIDO+' '+E.EST_NOMBRE AS
'ESTUDIANTE',MAT.PEN_MATERIA ,(SELECT SUM(QUI_NOTA) FROM
QUIMESTRALES  WHERE PEN_CODIGO=Q.PEN_CODIGO)AS 'NOTA
FINAL',Q.QUI_ESTADO FROM ESTUDIANTE E, MATRICULA M,QUIMESTRALES
Q,MATERIA MAT,DOCENTE D WHERE M.EST_CODIGO=E.EST_CODIGO AND
M.MATR_CODIGO=Q.MATR_CODIGO AND MAT.PEN_CODIGO=Q.PEN_CODIGO
AND MAT.DOCEN_CODIGO=D.DOCEN_CODIGO
GO
```

```
CREATE proc [dbo].[sp_insertar_notas]
@codCurso int ,
@codMetricula int
As begin declare
@codMateria int
declare Cursor1 Cursor
for select PEN_CODIGO from dbo.MATERIA_CURSO C WHERE
C.CUR_CODIGO=@codCurso
open Cursor1 fetch Cursor1 into @codMateria
while(@@FETCH_STATUS=0)
BEGIN
if(@codMateria is not null)
        BEGIN
        INSERT INTO [MATRICULACION].[dbo].[QUIMESTRALES]
       ([PEN_CODIGO]
       ,[MATR_CODIGO]
        ,[QUI_NOMBRE]
       ,[QUI_NUMERO]
       ,[QUI_NOTA]
       ,[QUI_ESTADO])
   VALUES(@codMateria --codpen
       ,@codMetricula --codMatr
       ,'PRIMERO'
       ,1
       ,0
       ,'REPROBADO')

       INSERT INTO [MATRICULACION].[dbo].[QUIMESTRALES]
       ([PEN_CODIGO]
       ,[MATR_CODIGO]
        ,[QUI_NOMBRE]
       ,[QUI_NUMERO]
       ,[QUI_NOTA]
```

```
        ,[QUI_ESTADO])
    VALUES(@codMateria --codpen
        ,@codMetricula --codMatr
        ,'SEGUNDO'
        ,2
        ,0
        ,'REPROBADO')
    END
    fetch Cursor1 into @codMateria
END
```

MANUAL INSTALACIÓN

Instalación de Sql Server2008

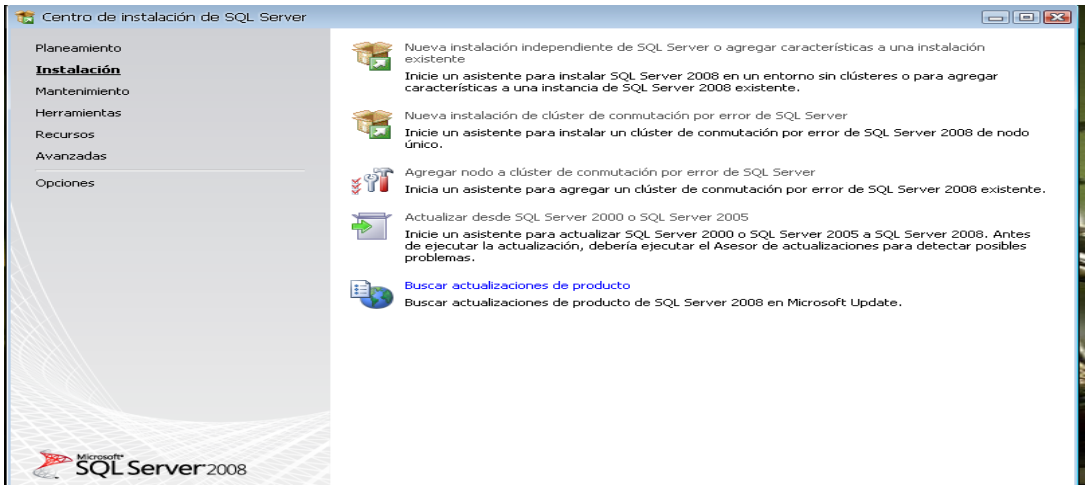


Figura 1*: Ventana de instalación.*

Primero se ejecuta la instalación de SQL SERVER 2008 desde el CD, luego nos sale un ventana que es el asistente de la Instalación, hacemos clic en la Pestaña "Instalación" y luego en la opción "Nueva Instalación independiente de SQL Server o agregar características a una instalación existente".

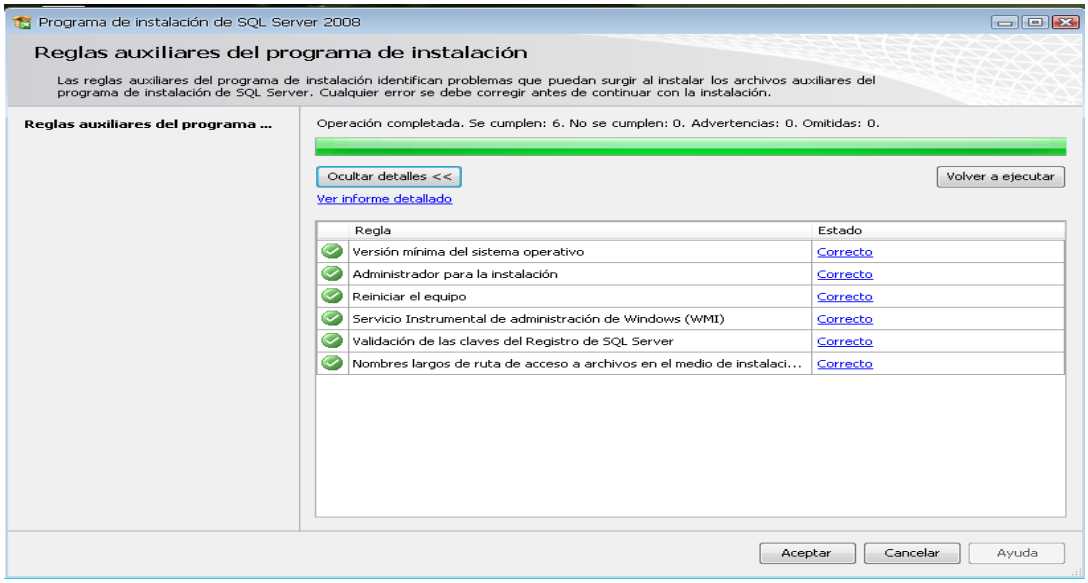


Figura 2: *Ventana de reglas auxiliares.*

En la siguiente ventana comienza a cargar algunas Reglas auxiliares que son necesarias para la instalación de SQL Server 2008, si todos están en Estado **Correcto** hacer clic en el botón siguiente de lo contrario si no están algunos en Estado **Correcto** no se puede seguir o continuar con la instalación (para ver las reglas clic en el botón mostrar detalles).

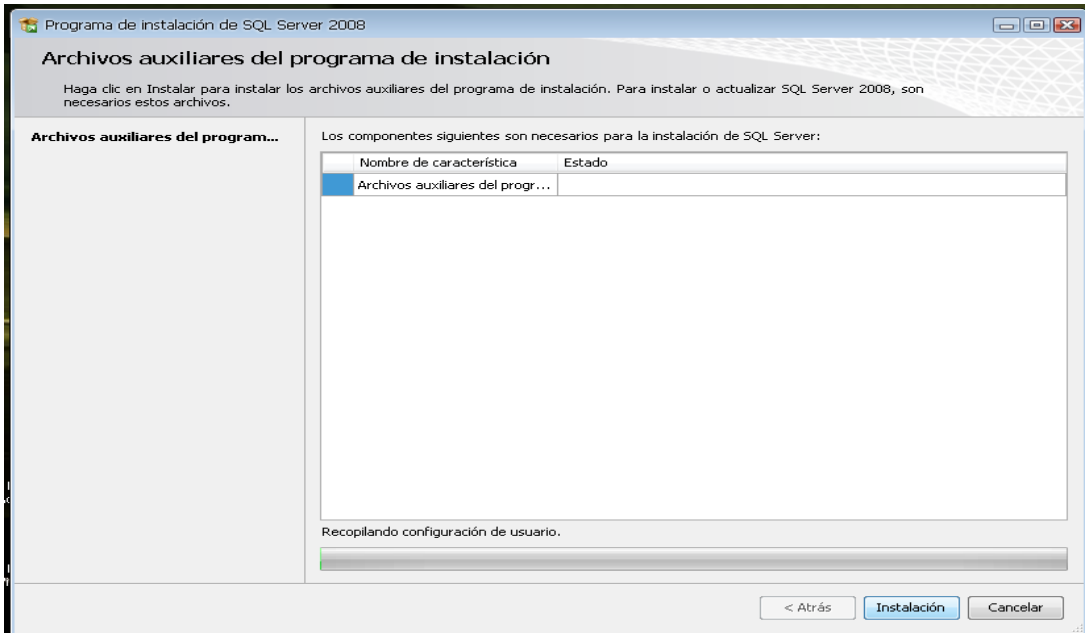


Figura 3: *Ventana de archivos auxiliares.*

En la siguiente ventana solo hacemos clic en el botón instalación para instalar algunos archivos auxiliares del programa de Instalación.

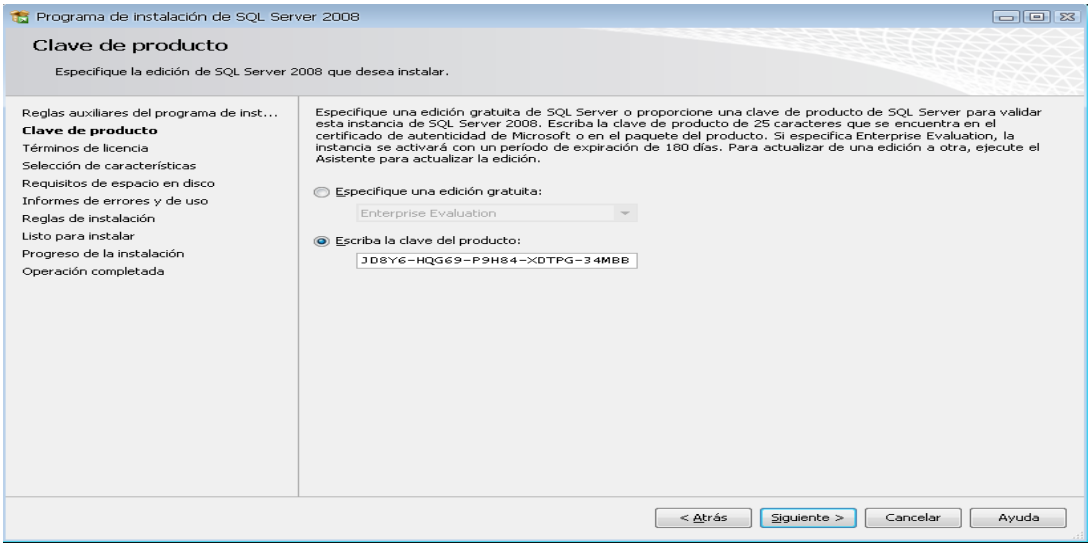


Figura 4*: Ventana Clave del producto.*

Luego de haber instalado los archivos auxiliares que son necesarios para la Instalación de SQL Server 2008, la siguiente ventana que nos sale es la clave del producto y nos dan dos opciones: "Especifique una edición gratuita" y "Escriba la clave del Producto", por defecto sale seleccionado la segunda opción y la clave, así que solo damos clic en el botón siguiente.

Figura 5: *Ventana términos de licencia.*

En esta ventana es si sobre los términos de la licencia del producto, hacemos clic en la opción "Acepto los términos de licencia" y luego clic en Siguiente

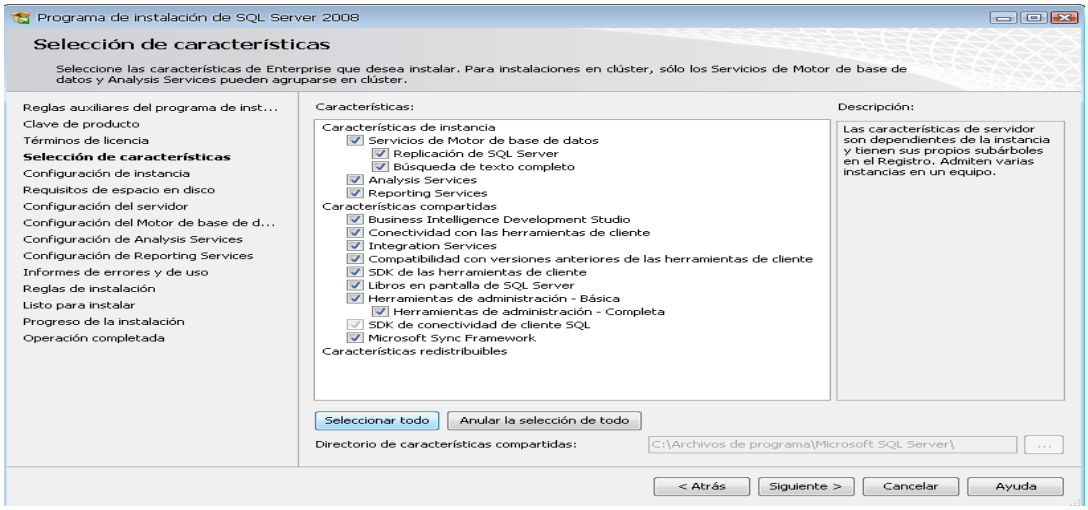


Figura 6: *Ventana selección de características.*

La siguiente ventana que nos muestra son las características que queremos agregar a la instalación, lo recomendable es seleccionar todas las características y luego clic en Siguiente.

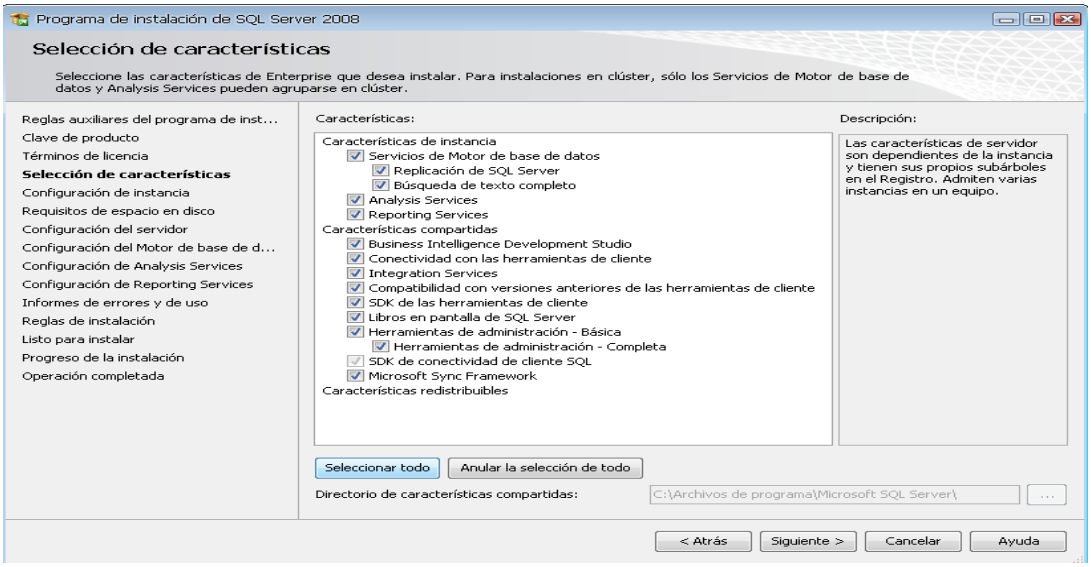


Figura 7: *Ventana selección de características.*

La siguiente ventana que nos muestra son las características que queremos agregar a

la instalación, lo recomendable es seleccionar todas las características y luego clic en Siguiente.

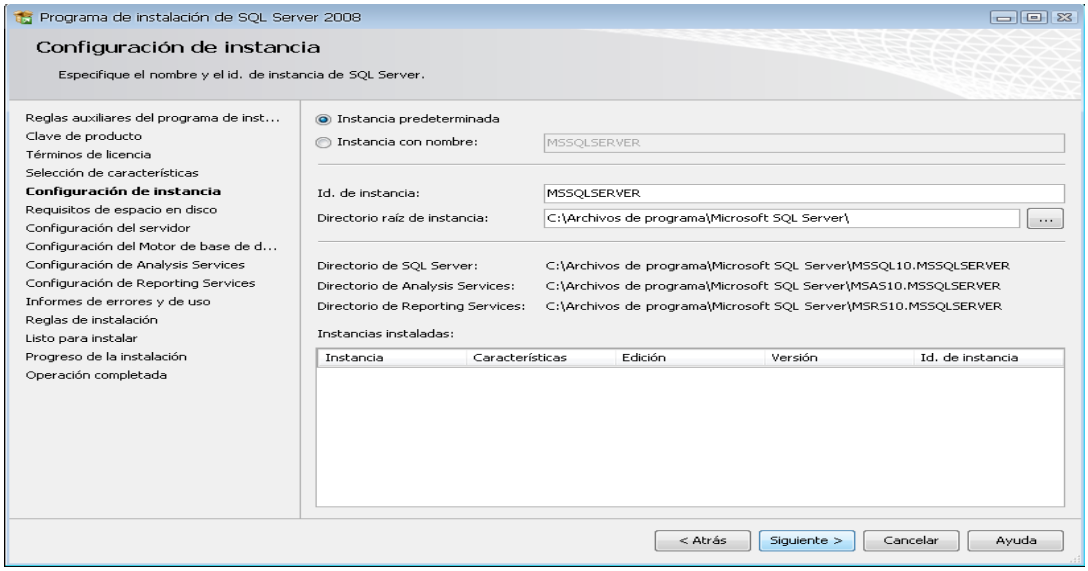


Figura 8: *Ventana configuración de instancia.*

Luego de haber agregado las características, la siguiente ventana a mostrar es sobre la configuración de la instancia de SQL Server 2008; es decir el nombre de la instalación, su ID y su ubicación en donde se instalará el producto. Nos dan 2 opciones: "Instancia predeterminada" y "Instancia con nombre", seleccionar la primera opción y luego clic en siguiente. En la siguiente ventana también damos click en siguiente.

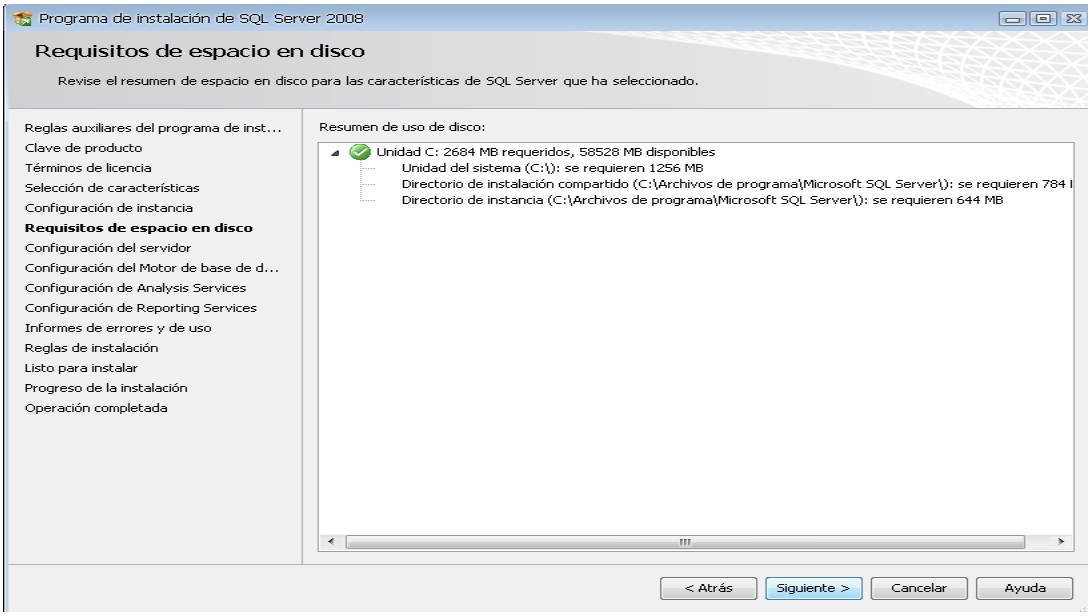


Figura 9: *Ventana requisitos de espacio.*

En esta ventana le damos siguiente ya que solo nos indica el espacio requerido en el disco y la ruta en el cual se va a guardar los archivos que genera la instalación del SQL.

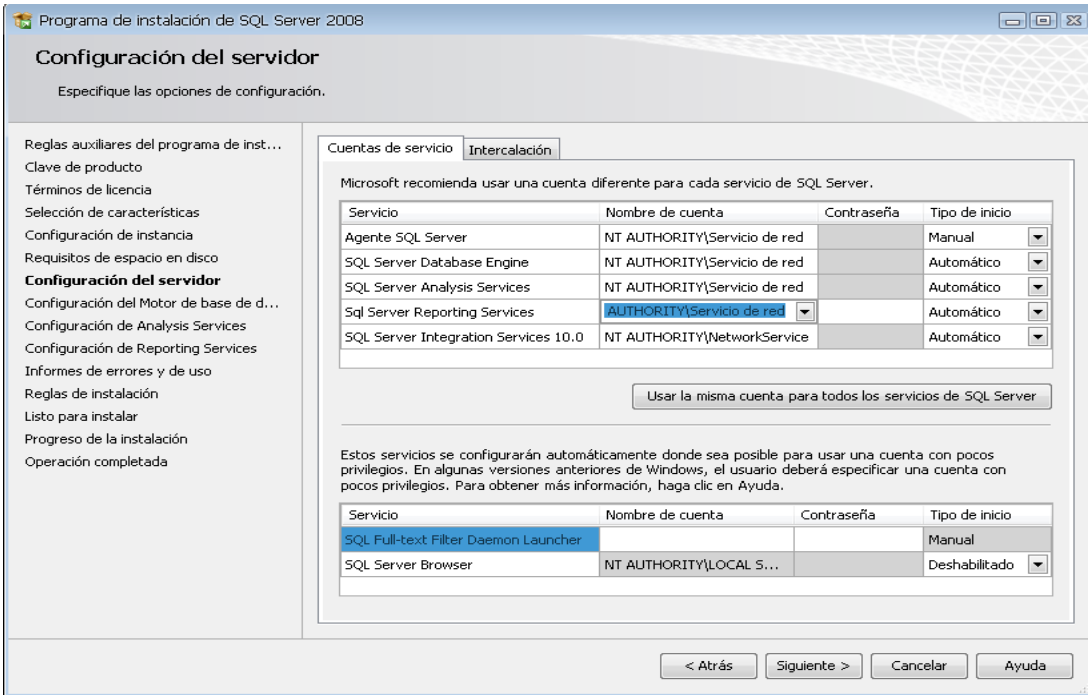


Figura 10: *Ventana configuración del servidor.*

En la siguiente ventana configuramos el servidor y para eso nos pide una cuenta de usuario, lo cual podemos crear un usuario específico para la instalación SQL, o de lo contrario podemos usar el usuario actual. En este caso usaremos "NT AUTHORITY\Servicio de red", luego damos clic en la pestaña "INTERCALACION".

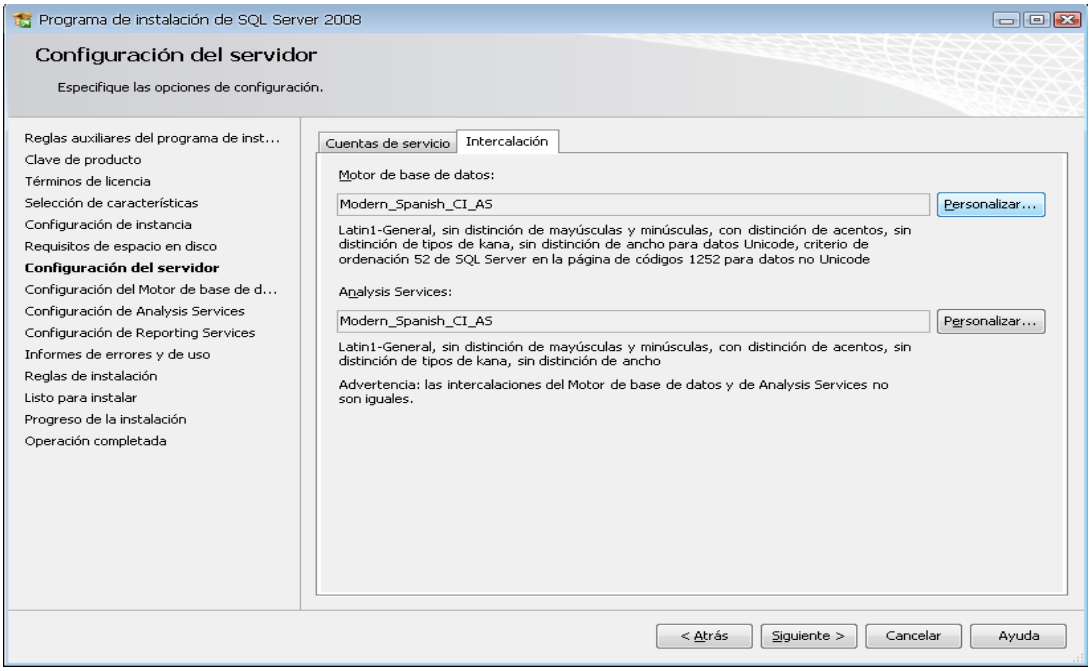


Figura 11: *Ventana configuración del servidor 1.*

En esta pestaña de Intercalación configuramos el "motor de Base de Datos" y el "Servicio de Análisis", para lo cual hacemos clic en el botón Personalizar en cada uno. Luego lo personalizamos como nos muestra las figuras "Motor de Base de Datos" y "AnalysisServices" y luego hacemos clic en siguiente.

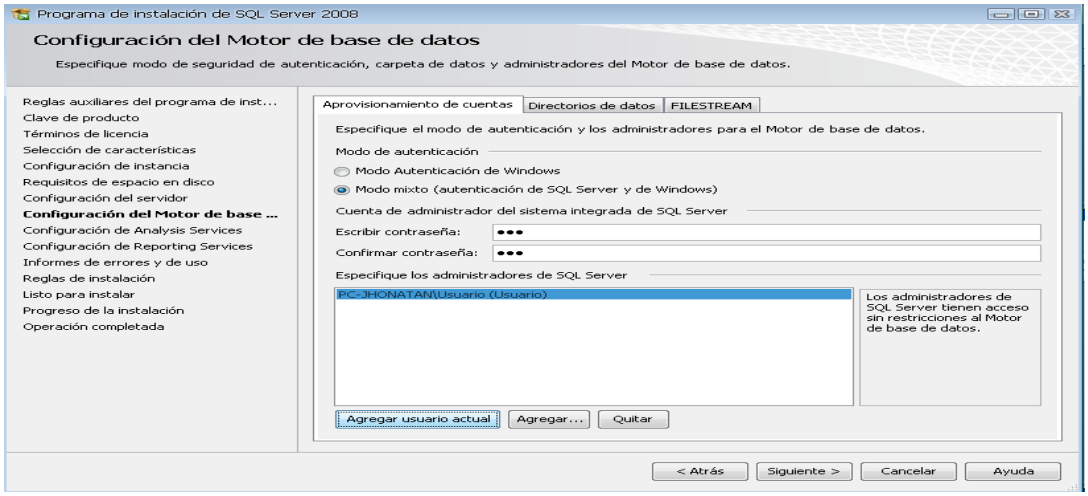


Figura 12: *Ventana configuración del motor de base de datos.*

Luego la siguiente ventana que nos muestra es la configuración del MOTOR DE BASE DE DATOS, nos dan 2 opcines MODO AUTENTICACION DE WINDOWS Y MODO MIXTO (autenticación de SQL Server y de Windows). Lo recomendable es seleccionar el MODO MIXTO, al seleccionar esta opción se activan las cajas de contraseña para el inicio de sesión de autenticación de SQL Server (inicio de sesión de sa). **NO OLVIDAR LA CONTRASEÑA PORQUE VA A SER NECESARIO MÁS ADELANTE.**

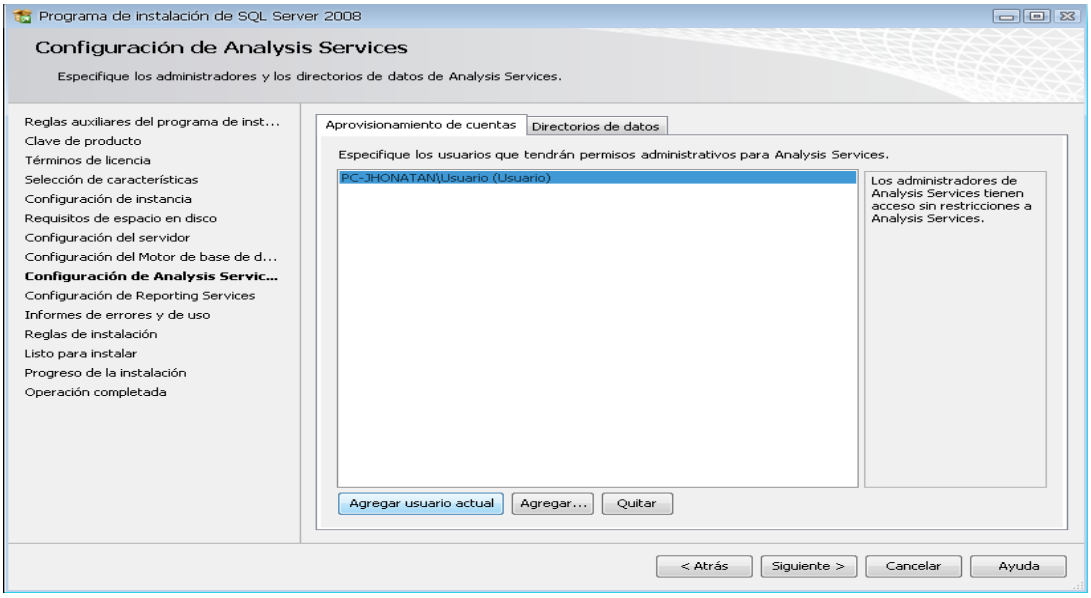


Figura 13: *Ventana configuración de analyisis services*.

Luego nos pide que agreguemos los administradores de SQL Server para lo cual agregamos el usuario actual; luego cli en siguiente.

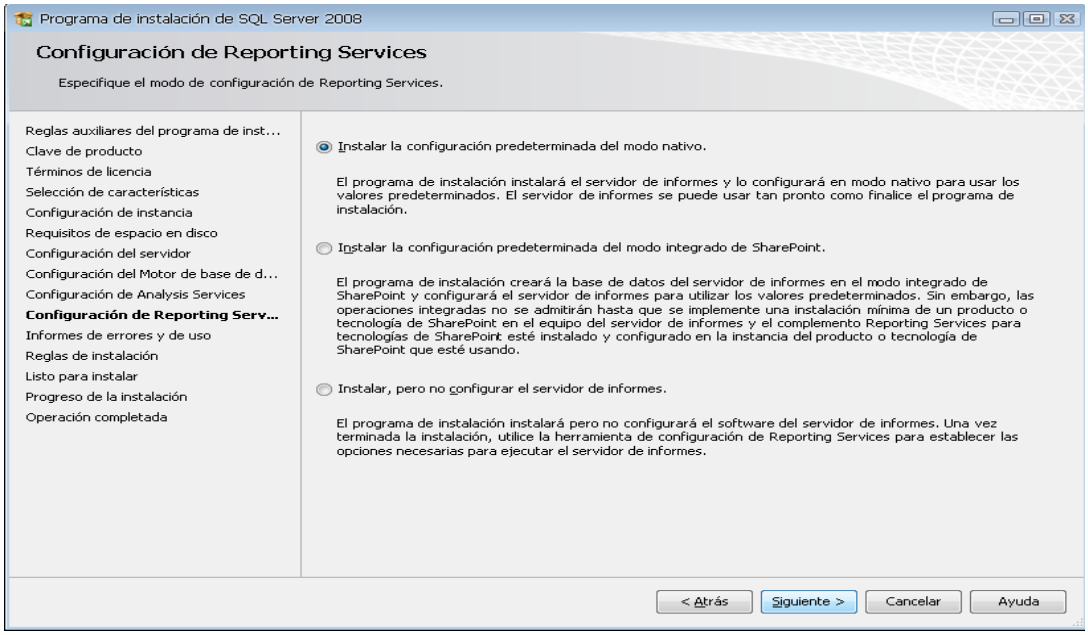


Figura 14: *Ventana configuración de reporting services.*

Aquí seleccionamos una opción en la cual nos permite instalar el SQL en forma nativa dependiendo de su necesidad.

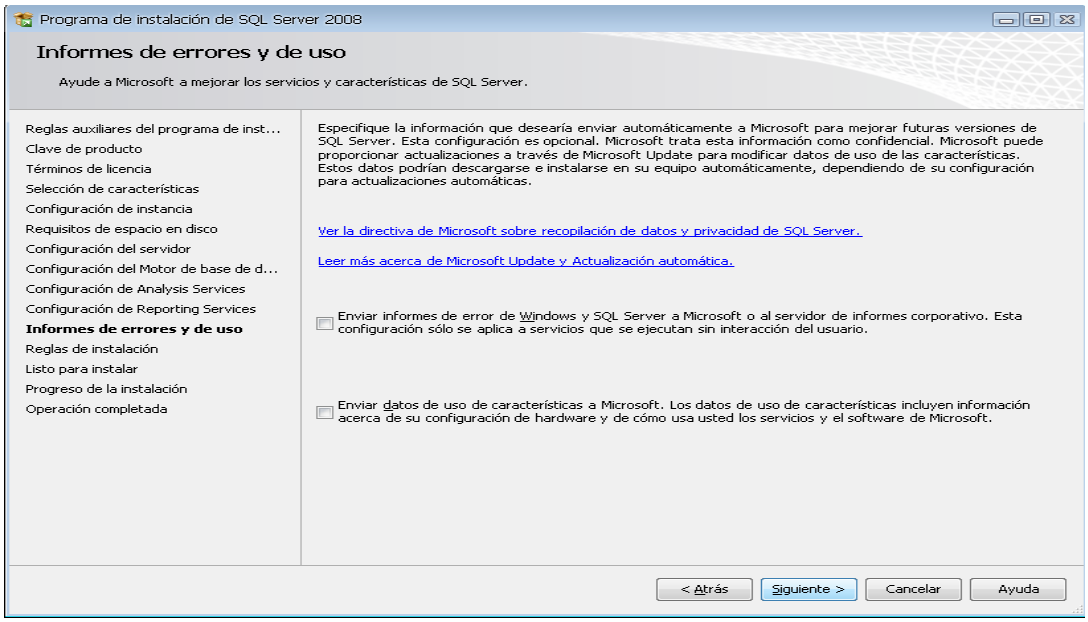


Figura 15: *Ventana informes de errores.*

En esta ventana de Configuración de Reporting Services también nos dan 3 opciones, por defecto sale seleccionado la primera opción, así que solo damos clic n siguiente (2 veces).

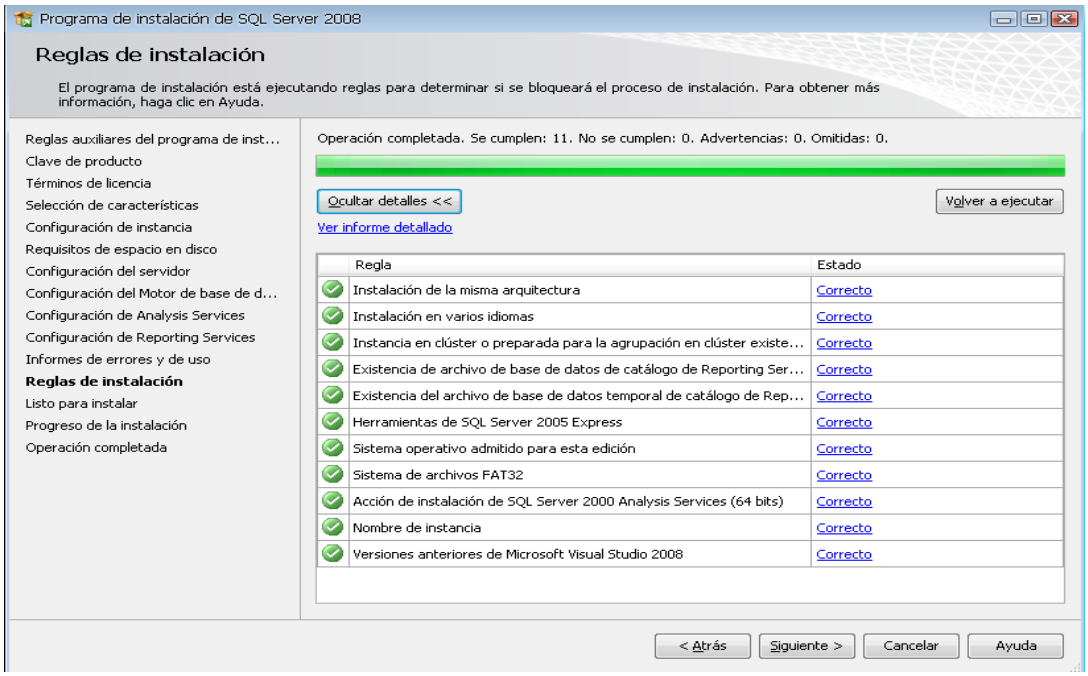


Figura 16: *Ventana reglas de instalación.*

Luego la siguiente ventana a mostrar son reglas que nos piden para poder INSTALAR SQL SERVER. Si todas las reglas están en estado **Correcto** hacer clic en siguiente y luego hacemos clic en instalación y

NOTA: Si todas las reglas no están en estado **Correcto** no se podrá seguir con la instalación (para ver las reglas clic en el botón mostrar detalles).

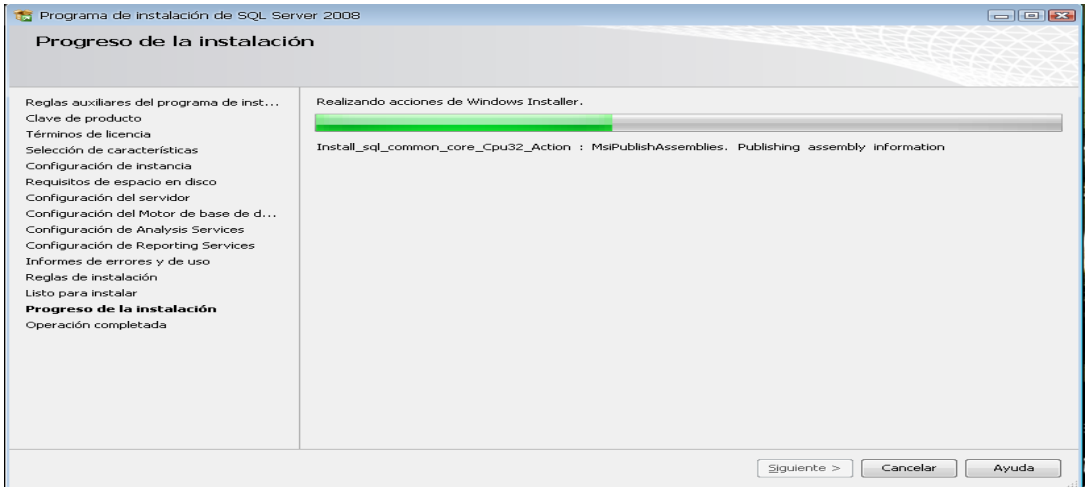


Figura 17: *Ventana progreso de instalación.*

Esperamos que termine la instalación, luego clic en siguiente y finalmente en cerrar.

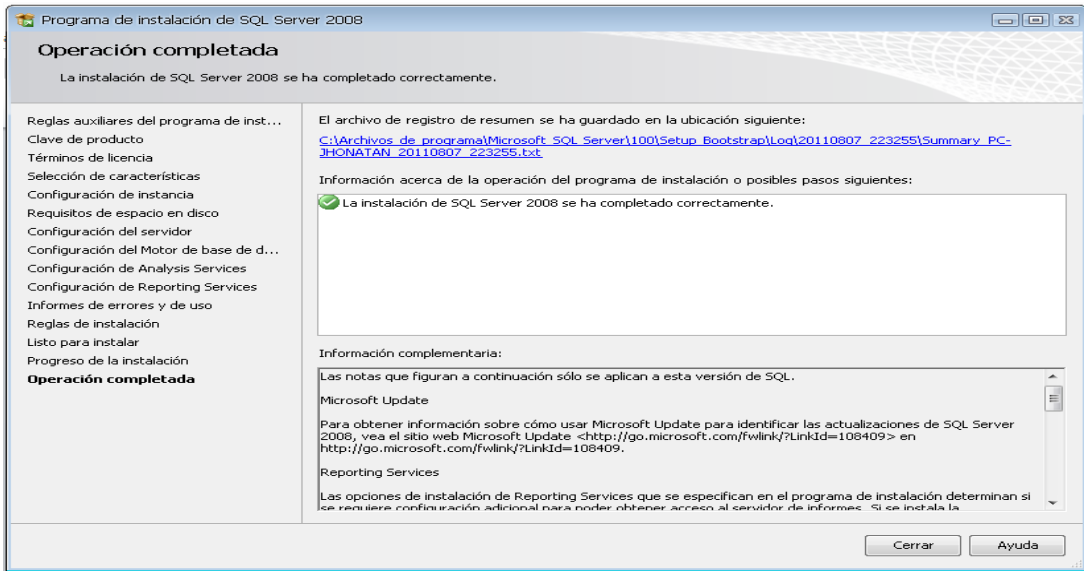


Figura 18: *Ventana operación completada*

Instalación de Visual Estudio 2010

Figura 1: *Ventana instalación.*

Primero se ejecuta Instalar Microsoft Visual Studio 2010 desde el CD, luego nos aparecerá una ventana que es el asistente de instalación, aceptamos los términos y condiciones del visual y "Siguiente" y empezara la instalación.

Figura 2: *Ventana Asistente de instalación.*

En el primer paso de la instalación de Microsoft Visual Studio 2010 Ultimante, desmarcaremos "Sí, enviar a Microsoft Corporation información sobre la instalación" (si no queremos enviar esta información) y pulsaremos "Siguiente"

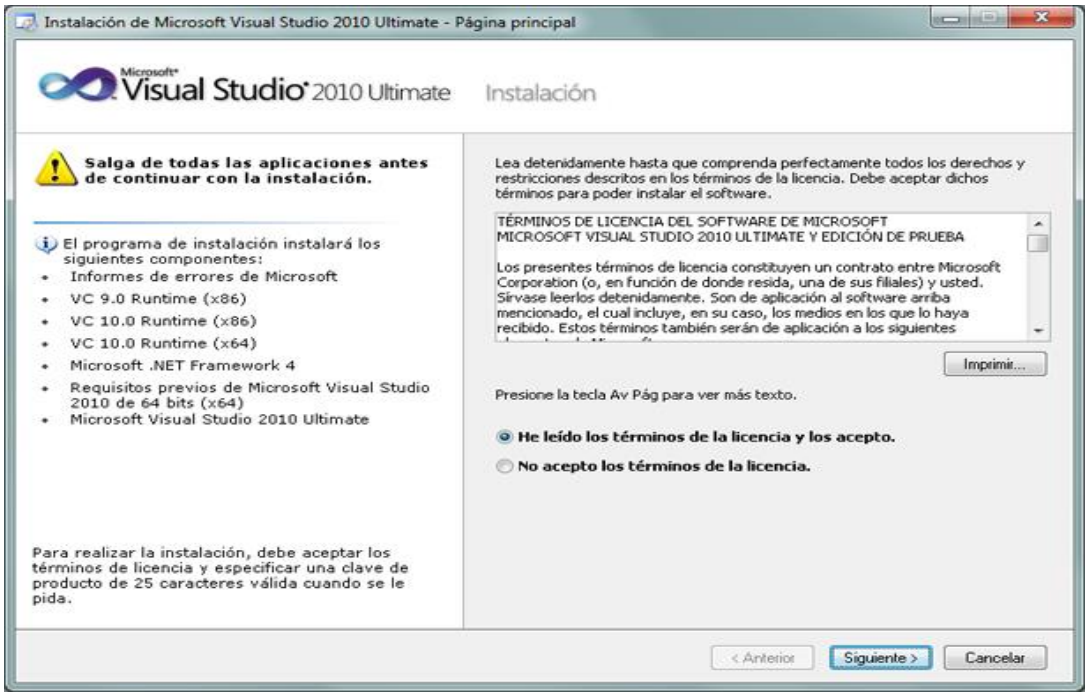


Figura 3: *Ventana de términos.*

Leeremos los términos de licencia del software de Microsoft Visual Studio 2010 Ultimate (edición de prueba). Si estamos de acuerdo marcaremos "He leído los términos de la licencia y los acepto". Pulsaremos "Siguiente" para continuar:

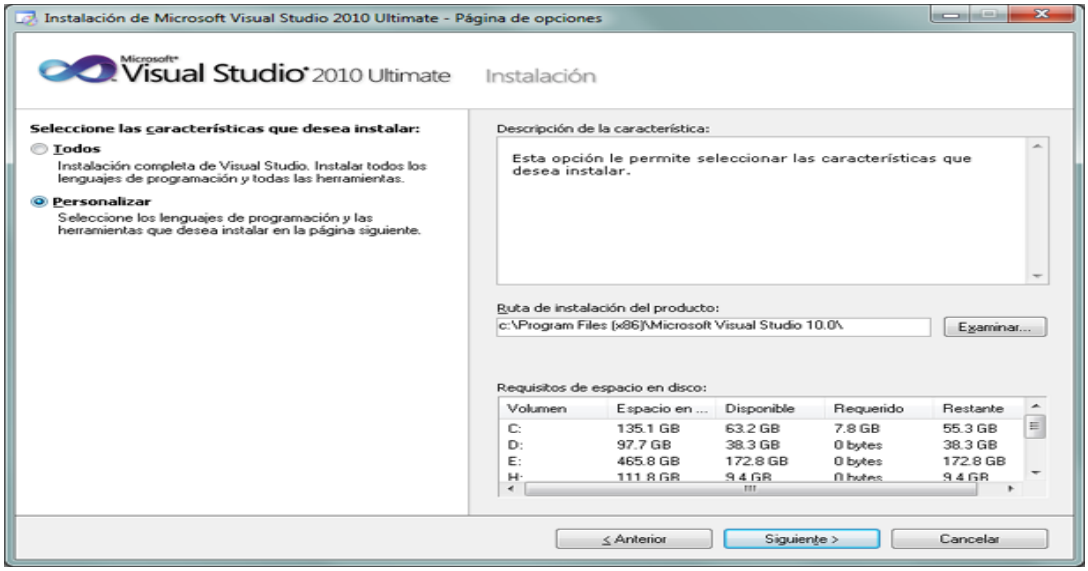


Figura 4:.*Ventana de personalización.*

A continuación marcaremos "Personalizar" para seleccionar los lenguajes de programación y las herramientas que se quieran instalar. En "Ruta de instalación del producto" indicaremos la unidad y carpeta de destino de la instalación:

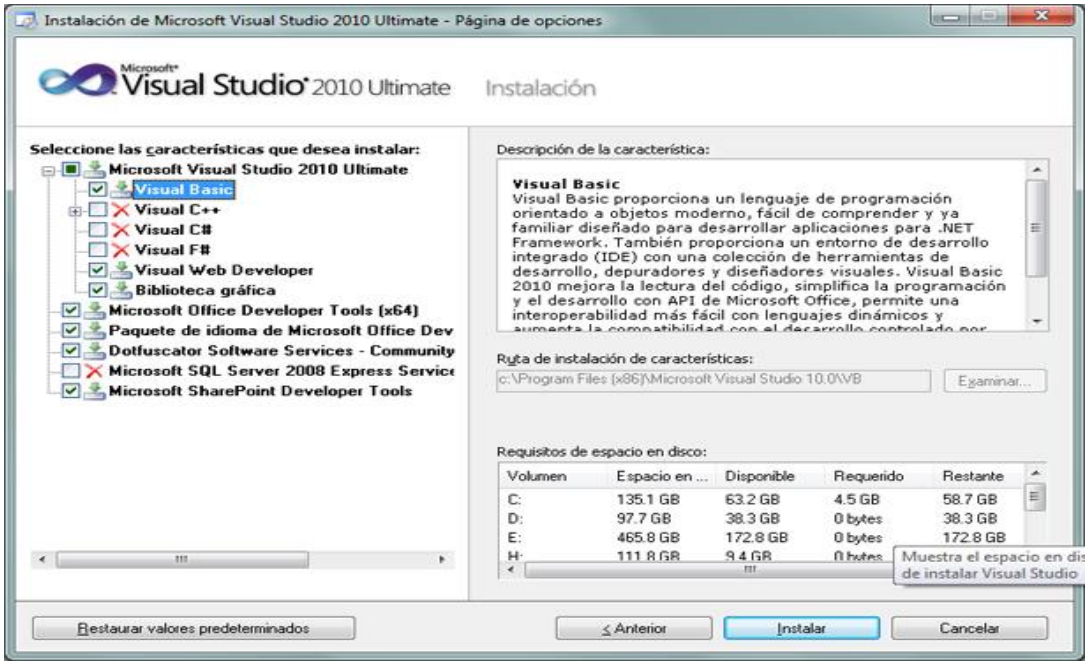


Figura 5: *Ventana selección de características.*

Seleccionaremos los lenguajes a instalar:

- Visual C++.
- Visual C#.
- Visual F#.

Seleccionaremos también las características a instalar:

- Microsoft Office Developer Tools.
- Dotfuscator Software Services.
- Microsoft SQL Server 2008 Express.

Una vez seleccionadas las características a instalar pulsaremos en el botón "Instalar":

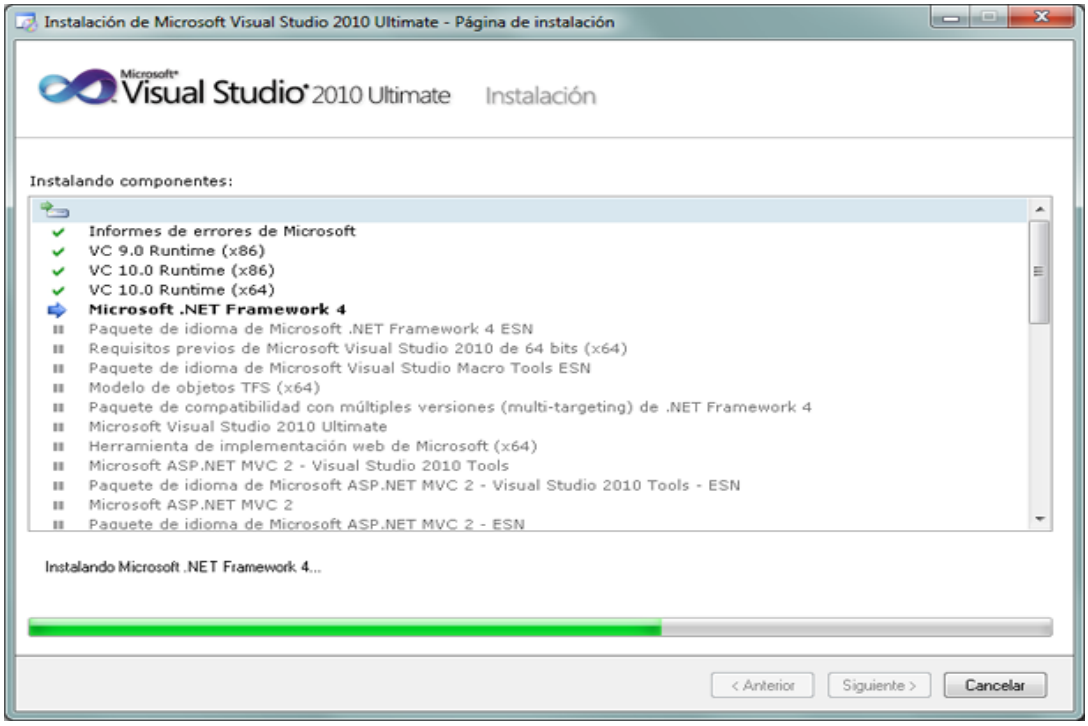


Figura 6: *Ventana de instalación de componentes.*

Se iniciará la instalación de Microsoft Visual Studio 2010 Ultimate: informe de errores de Microsoft, VC 9.0 Runtime, VC 10.0 Runtime, Microsoft .Net Framework 4, Microsoft Visual Studio 2010 Ultimate, Microsoft ASP .Net, etc.:

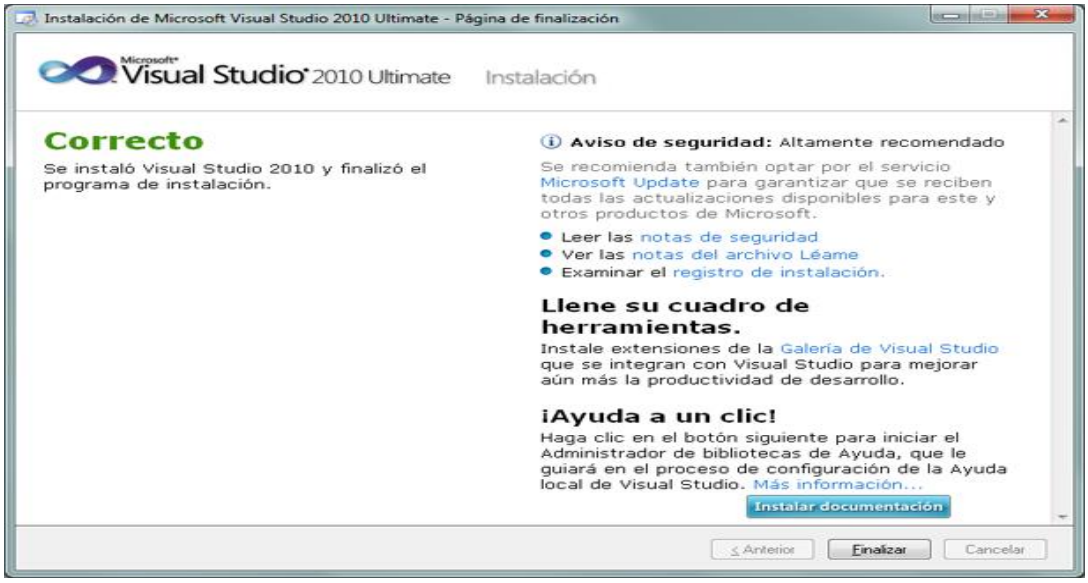


Figura 7: *Ventana de instalación Correcta*

Después que haya terminado la instalación seleccionar "Finalizar".

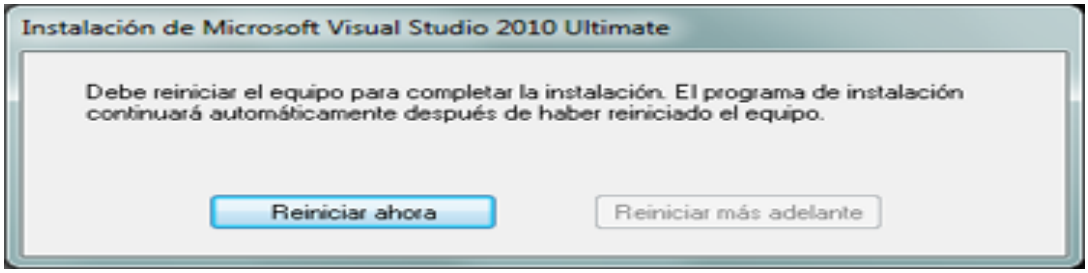


Figura 8: *Ventana de reinicio de equipo*

Debe reiniciar el equipo para completar la instalación. El programa de instalación continuará automáticamente después de haber reiniciado el equipo.

MANUAL USUARIO

Figura 1: *Esta pantalla es la principal del sistema donde se encuentra la Misión Visión y Contactos de la institución.*

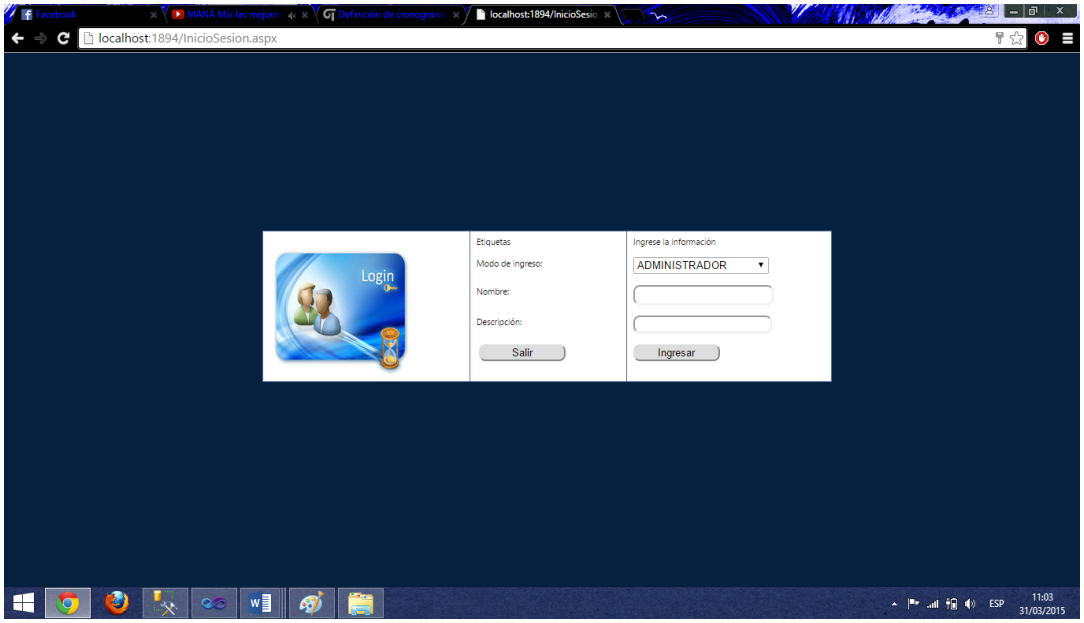


Figura 2: *Esta pantalla es la secundaria en la cual nos logeamos para ingresar al sistema utilizando las credenciales facilitadas por el administrador.*

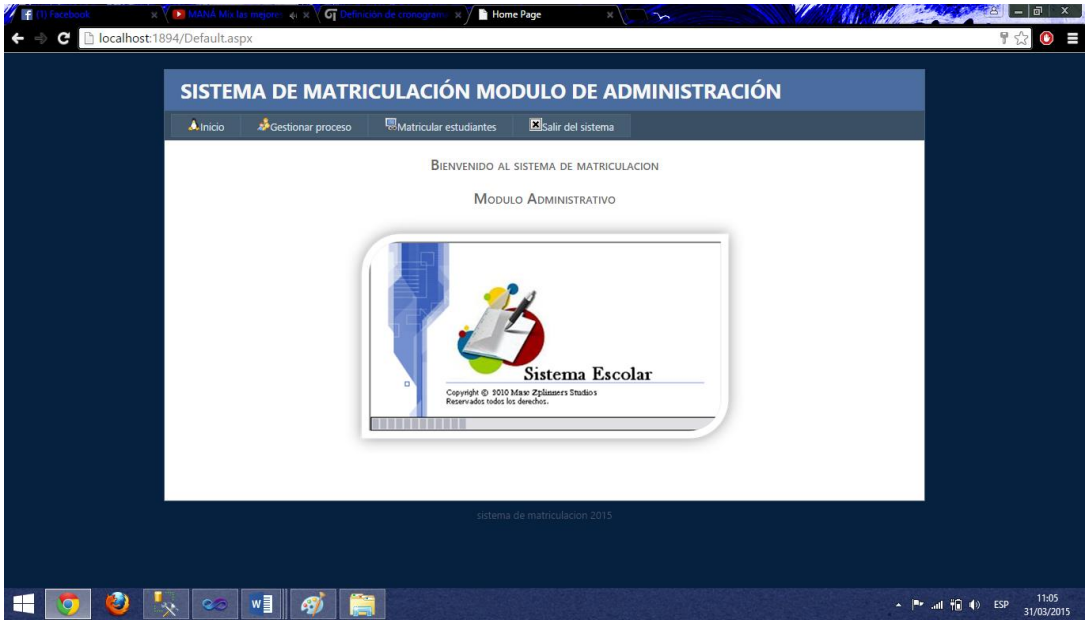


Figura 3: *Esta pantalla nos muestra los campos del administrador los cuales vamos a utilizar para matricular a los estudiantes.*

Figura 4: *Esta pantalla muestra el formulario del representante el cual se debe llenar con sus respectivos datos.*

Figura 5: *Esta pantalla muestra el formulario del estudiante el cual se debe realizar el llenado con sus respectivos datos.*

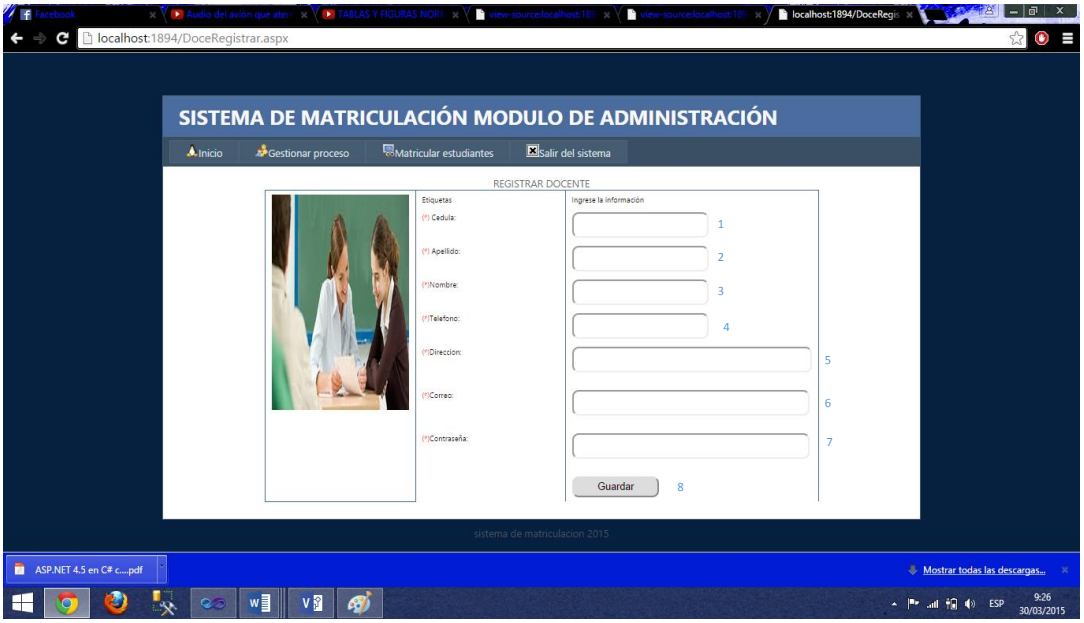


Figura 6: *Pantalla del formulario del docente se debe realizar el llenado con sus respectivos datos.*

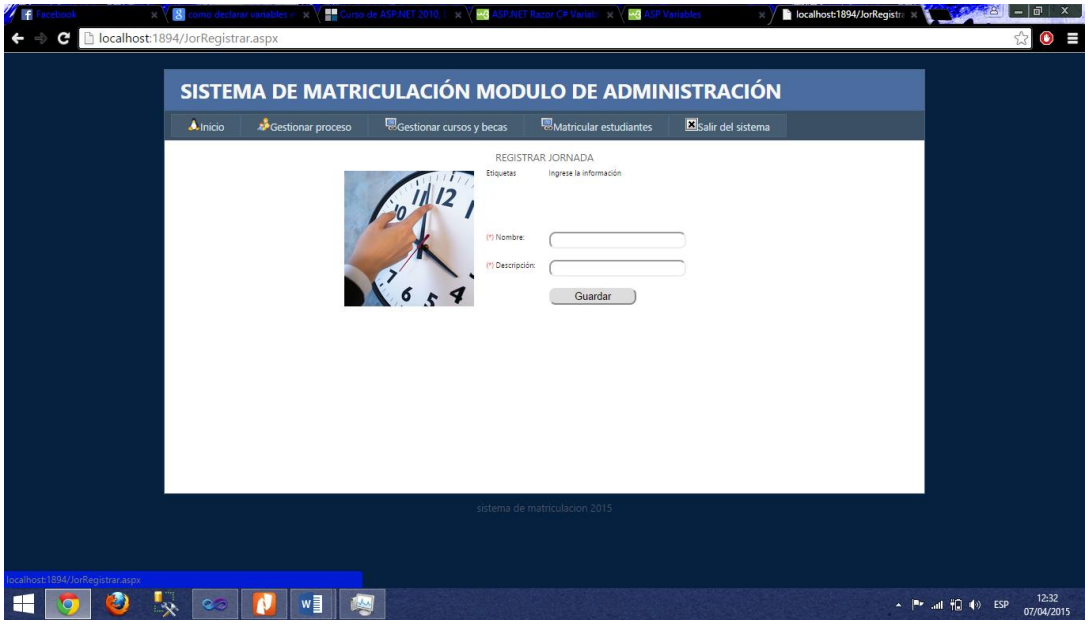


Figura 7: *Pantalla del formulario de las jornadas se debe realizar el llenado con las jornadas que dispongan las instituciones.*

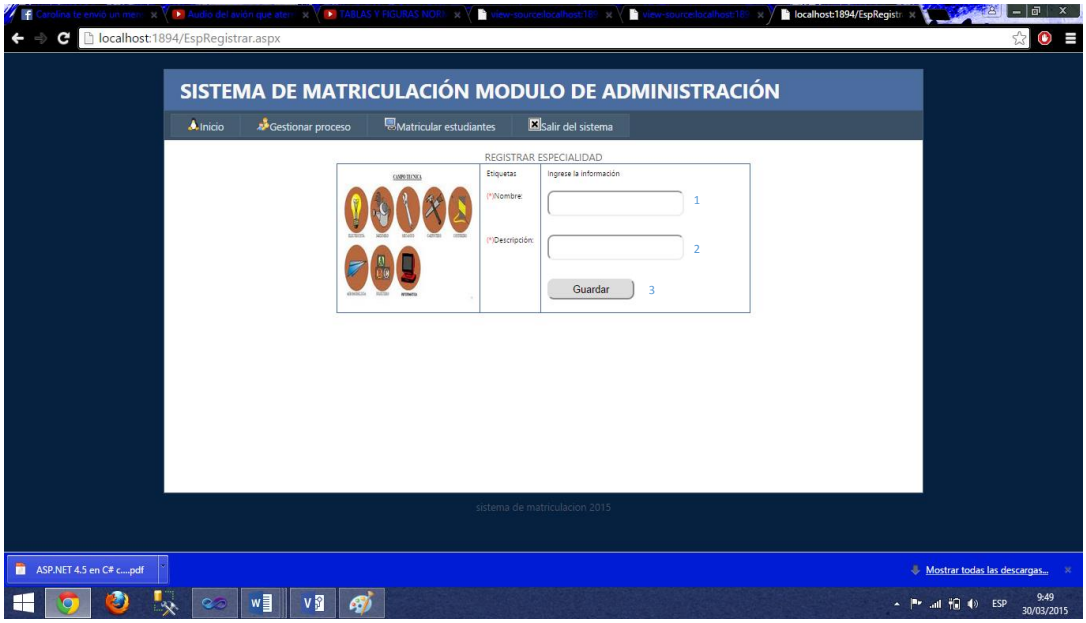


Figura 8: *Esta pantalla muestra el formulario de especialidad en la cual se debe realizar el llenado con las especialidades que disponga la institución.*

Figura 9: *Esta pantalla muestra el formulario de materias en el cual se va a registrar las materias que se dictan.*

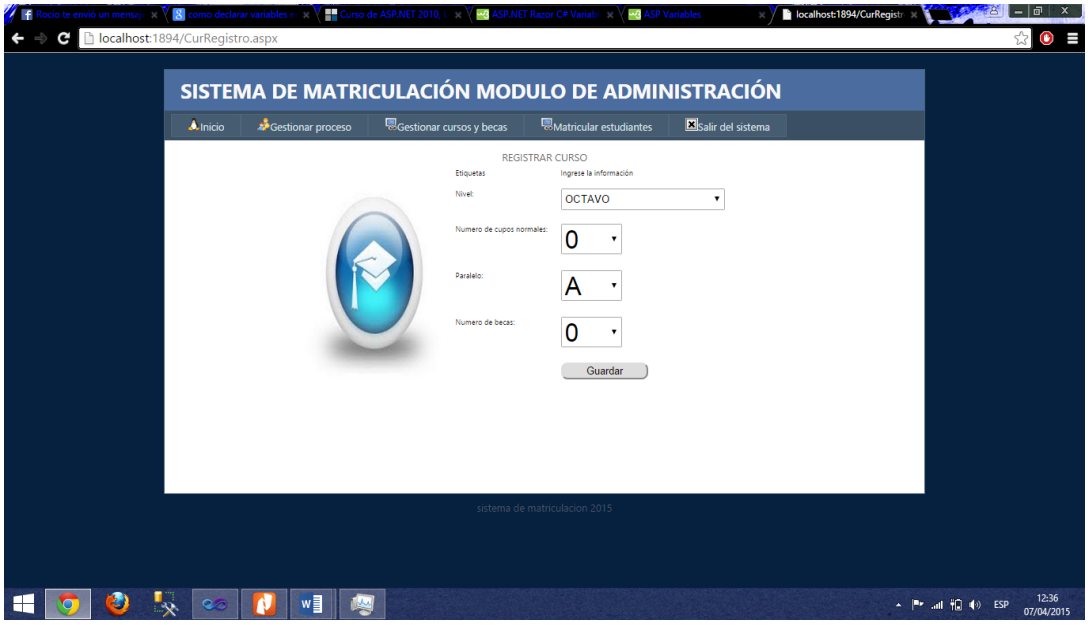


Figura 10: *Esta pantalla muestra el formulario de cursos en el cual se debe registrar los cursos en los cuales vamos a matricular al estudiante.*

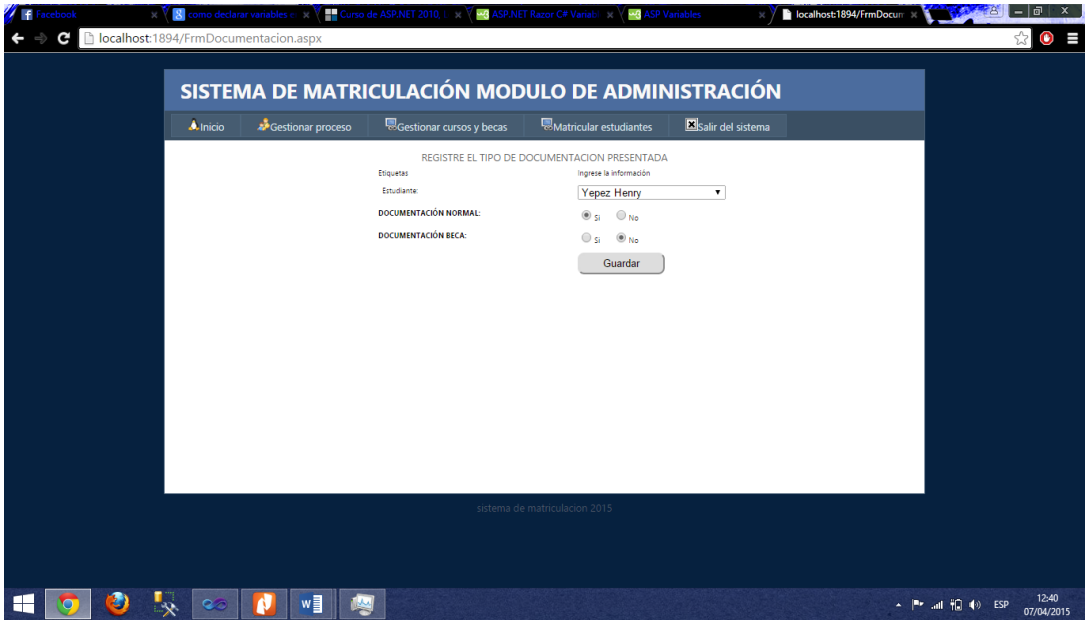


Figura 11: *Esta pantalla muestra el formulario de registro de documentación la cual se debe seleccionar si el estudiante entrega toda la documentación.*

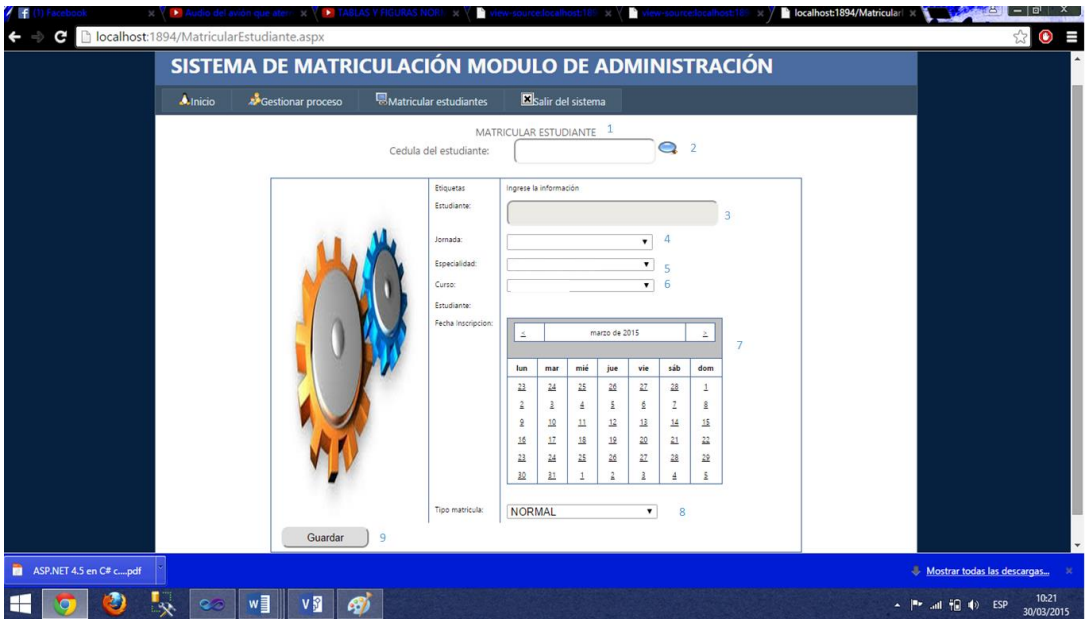


Figura 12: *Esta pantalla muestra el formulario Matrícula en el cual se le asigna el curso, especialidad y jornada en la que va estudiar.*

MANUAL TÉCNICO

Diccionario de Datos

Diccionario de se desarrollara durante el análisis de flujo de datos brinda apoyo a los analistas que participan en la determinación de los requerimientos del sistema

VICTOR_RM.MATR...N - dbo.USUARIO

Nombre de columna	Tipo de datos	Permitir v...
USUARIO	nchar(50)	☑
CONTRASENA	nchar(50)	☑
ESTADO	nchar(10)	☑
		☐

Figura 1: *Tabla de datos del usuario.*

VICTOR_RM.MAT....REPRESENTANTE

Nombre de columna	Tipo de datos	Permitir v...
REP_CODIGO	numeric(30, 0)	☐
REP_CEDULA	varchar(13)	☑
REP_APELLIDO	varchar(100)	☑
REP_NOMBRE	varchar(100)	☑
REP_DIRECCION	varchar(150)	☑
REP_TELEFONO	varchar(20)	☑
REP_MOVIL	varchar(20)	☑
REP_CORREO	varchar(150)	☑
REP_ESTADO	varchar(20)	☑

Figura 1: *Tabla de datos representante.*

VICTOR_RM.MATR...dbo.ESTUDIANTE

Nombre de columna	Tipo de datos	Permitir v...
EST_CODIGO	numeric(30, 0)	☐
REP_CODIGO	numeric(30, 0)	☑
EST_CEDULA	varchar(13)	☑
EST_NOMBRE	varchar(100)	☑
EST_APELLIDO	varchar(100)	☑
EST_DIRECCION	varchar(150)	☑
EST_TELEFONO	varchar(20)	☑
EST_CELULAR	varchar(20)	☑
EST_CORREO	varchar(150)	☑
EST_ESTADO	varchar(20)	☑
EST_CONTRASENA	varchar(20)	☑

Figura 3: *Tabla de datos del Estudiante.*

VICTOR_RM.MATR...N - dbo.JORNADA

Nombre de columna	Tipo de datos	Permitir v...
JOR_CODIGO	numeric(30, 0)	☐
JOR_NOMBRE	varchar(150)	☑
JOR_DESCRIPCION	varchar(150)	☑

Figura 4: *Tabla de datos de la Jornada.*

VICTOR_RM.MATR...N - dbo.MATERIA

Nombre de columna	Tipo de datos	Permitir v...
PEN_CODIGO	numeric(30, 0)	☐
DOCEN_CODIGO	numeric(30, 0)	☑
PEN_MATERIA	varchar(150)	☑

Figura5*: Tabla de datos Materia*

VICTOR_RM.MATRI...bo.ESPECIALIDAD

Nombre de columna	Tipo de datos	Permitir v...
ESP_CODIGO	numeric(30, 0)	☐
ESP_NOMBRE	varchar(150)	☑
ESP_DESCRIPCION	varchar(150)	☑
ESP_ESTADO	varchar(20)	☑

Figura 6: *Tabla de datos Especialidad.*

VICTOR_RM.MATR...N - dbo.CURSOS

Nombre de columna	Tipo de datos	Permitir v...
CUR_CODIGO	numeric(30, 0)	☐
CUR_NOMBRE	varchar(150)	☑
CUR_NIVEL	varchar(150)	☑
CUR_NUM_ESTUDIAN...	varchar(150)	☑
CUR_PARALELO	varchar(150)	☑
CUR_ESTADO	varchar(50)	☑

Figura 7: *Tabla de datos Curso.*

El desarrollo del sistema, las interfaces se realizaron en Microsoft Visual Studio 2010 con formularios Web, c#.

Login Usuario.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.ec.ClsLogicaNegocio.Entidades;
using com.ec.ClsLogicaNegocio.Vistas;
using System.Data;
using com.ec.ClsComunBase;

namespace com.ec.ClsLogicaNegocio
{

   public class ClsLogicaNegocio
   {
#region Usuarios

       //administrador
       public Usuario consultarUsuario(string buscar)
       {
          Usuario aux = new Usuario();
          List<Usuario> estu = new List<Usuario>();
          DataSet ds = new ClsComunBase.ClsConexion().ExecuteDataSet("select *
from USUARIO U WHERE U.USUARIO='" + buscar + "'");
          for (int i = 0; i < ds.Tables[0].Rows.Count; i++)
```

```
        {

            aux.usuario = ds.Tables[0].Rows[i][0].ToString();
            aux.contrasena = ds.Tables[0].Rows[i][1].ToString();
            aux.estado = ds.Tables[0].Rows[i][2].ToString();

        }
        return aux;
    }
    #endregion
```

Representante.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.ec.ClsLogicaNegocio.Entidades;
using com.ec.ClsLogicaNegocio.Vistas;
using System.Data;
using com.ec.ClsComunBase;

namespace com.ec.ClsLogicaNegocio
{

  public class ClsLogicaNegocio
  {

    #region Representante

    public Representante listarRepresentanteporCedula(string buscar)
    {
      Representante aux = new Representante();

      DataSet ds = new ClsComunBase.ClsConexion().ExecuteDataSet("Select*
from Representante R where R.REP_CEDULA = '" + buscar + "'");

      if (ds.Tables[0].Rows.Count > 0)
      {

        for (int i = 0; i < ds.Tables[0].Rows.Count; i++)
```

```
            {

                aux.repCodigo = ds.Tables[0].Rows[i][0].ToString();
                aux.cedula = ds.Tables[0].Rows[i][1].ToString();
                aux.apellido = ds.Tables[0].Rows[i][2].ToString();
                aux.nombre = ds.Tables[0].Rows[i][3].ToString();
                aux.direccion = ds.Tables[0].Rows[i][4].ToString();
                aux.telefono = ds.Tables[0].Rows[i][5].ToString();
                aux.celular = ds.Tables[0].Rows[i][6].ToString();
                aux.coreo = ds.Tables[0].Rows[i][7].ToString();
                aux.estado = ds.Tables[0].Rows[i][8].ToString();

            }


        }
        else
        {
            aux = null;
        }

        return aux;
    }
    public List<Representante> listarRepresentante(string buscar)
    {
        List<Representante> repre = new List<Representante>();

        DataSet ds = new ClsComunBase.ClsConexion().ExecuteDataSet("Select*
from Representante R where R.REP_CEDULA LIKE '%" + buscar + "%' ORDER
BY R.REP_APELLIDO ");
        for (int i = 0; i < ds.Tables[0].Rows.Count; i++)
        {
            Representante aux = new Representante();
            aux.repCodigo = ds.Tables[0].Rows[i][0].ToString();
            aux.cedula = ds.Tables[0].Rows[i][1].ToString();
            aux.apellido = ds.Tables[0].Rows[i][2].ToString();
            aux.nombre = ds.Tables[0].Rows[i][3].ToString();
            aux.direccion = ds.Tables[0].Rows[i][4].ToString();
            aux.telefono = ds.Tables[0].Rows[i][5].ToString();
            aux.celular = ds.Tables[0].Rows[i][6].ToString();
            aux.coreo = ds.Tables[0].Rows[i][7].ToString();
```

```
            aux.estado = ds.Tables[0].Rows[i][8].ToString();
            repre.Add(aux);

        }
        return repre;
    }
    public void insertarRep(Representante rep)
    {

        string SQL = "insert into REPRESENTANTE values('" + rep.cedula + "','" +
rep.apellido + "','" + rep.nombre + "','" + rep.direccion + "','" + rep.telefono + "','" +
rep.celular + "','" + rep.coreo + "','Activo')";
        new ClsConexion().ExecuteNonQuery(SQL);
    }

    public void actualizarRep(Representante rep)
    {
        try
        {

            string SQL = "UPDATE REPRESENTANTE SET REP_CEDULA='" +
rep.cedula + "', REP_NOMBRE='" + rep.nombre + "',REP_APELLIDO='" +
rep.apellido + "',REP_DIRECCION='" + rep.direccion + "',REP_TELEFONO='" +
rep.telefono + "',REP_MOVIL='" + rep.celular + "',REP_CORREO='" + rep.coreo +
"' WHERE REP_CODIGO='" + rep.repCodigo + "'";
            new ClsConexion().ExecuteNonQuery(SQL);
        }
        catch (Exception)
        {

            throw;
        }
    }
    public void eliminarRep(string codBorrar)
    {
        try
        {

            string SQL = "delete from REPRESENTANTE  where REP_CODIGO='"
+ codBorrar + "'";
            new ClsConexion().ExecuteNonQuery(SQL);
```

```
        }
        catch (Exception)
        {

            throw;
        }
    }

    public List<RepresentantesView> listarRepresentanteView(string buscar)
    {
        List<RepresentantesView> repre = new List<RepresentantesView>();

        DataSet ds = new ClsComunBase.ClsConexion().ExecuteDataSet("select *
from representanteView where REP_CEDULA like '%" + buscar + "%' order by
REP_APELLIDO ");
        for (int i = 0; i < ds.Tables[0].Rows.Count; i++)
        {
            RepresentantesView aux = new RepresentantesView();
            aux.CEDULA = ds.Tables[0].Rows[i][0].ToString();
            aux.APELLIDO = ds.Tables[0].Rows[i][1].ToString();
            aux.NOMBRE = ds.Tables[0].Rows[i][2].ToString();
            aux.TELEFONO = ds.Tables[0].Rows[i][3].ToString();
            aux.CELULAR = ds.Tables[0].Rows[i][4].ToString();
            aux.DIRECCION = ds.Tables[0].Rows[i][5].ToString();
            aux.CORREO = ds.Tables[0].Rows[i][6].ToString();
            repre.Add(aux);

        }
        return repre;
    }


    public Representante listarRepresentantePorCodigo(string codRep)
    {
        Representante aux = new Representante();

        DataSet ds = new ClsComunBase.ClsConexion().ExecuteDataSet("select *
from REPRESENTANTE R WHERE R.REP_CODIGO='" + codRep + "'");

        if (ds.Tables[0].Rows.Count > 0)
        {
```

```
            for (int i = 0; i < ds.Tables[0].Rows.Count; i++)
            {

                aux.repCodigo = ds.Tables[0].Rows[i][0].ToString();
                aux.cedula = ds.Tables[0].Rows[i][1].ToString();
                aux.apellido = ds.Tables[0].Rows[i][2].ToString();
                aux.nombre = ds.Tables[0].Rows[i][3].ToString();
                aux.direccion = ds.Tables[0].Rows[i][4].ToString();
                aux.telefono = ds.Tables[0].Rows[i][5].ToString();
                aux.celular = ds.Tables[0].Rows[i][6].ToString();
                aux.coreo = ds.Tables[0].Rows[i][7].ToString();
                aux.estado = ds.Tables[0].Rows[i][8].ToString();

            }


        }
        else
        {
            aux = null;
        }
        return aux;
    }
    #endregion
```

Mantenimiento de Estudiantes.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.ec.ClsLogicaNegocio.Entidades;
using com.ec.ClsLogicaNegocio.Vistas;
using System.Data;
using com.ec.ClsComunBase;

namespace com.ec.ClsLogicaNegocio
{

    public class ClsLogicaNegocio
```

```
{

 #region Estudiante
    public List<Estudiante> listarEstudiantePorRepresentante(string buscar)
    {
        List<Estudiante> repre = new List<Estudiante>();

        DataSet ds = new ClsComunBase.ClsConexion().ExecuteDataSet("select *
from ESTUDIANTE where REP_CODIGO='" + buscar + "'");
        for (int i = 0; i < ds.Tables[0].Rows.Count; i++)
        {
            Estudiante aux = new Estudiante();
            aux.codEstudiante = ds.Tables[0].Rows[i][0].ToString();
            aux.codRepresentante = ds.Tables[0].Rows[i][1].ToString();
            aux.cedula = ds.Tables[0].Rows[i][2].ToString();
            aux.nombre = ds.Tables[0].Rows[i][3].ToString();
            aux.apellido = ds.Tables[0].Rows[i][4].ToString();
            aux.direccion = ds.Tables[0].Rows[i][5].ToString();
            aux.telefono = ds.Tables[0].Rows[i][6].ToString();
            aux.celular = ds.Tables[0].Rows[i][7].ToString();
            aux.correo = ds.Tables[0].Rows[i][8].ToString();
            aux.estado = ds.Tables[0].Rows[i][9].ToString();
            repre.Add(aux);
        }
        return repre;
    }

    //select * from ESTUDIANTE
    public List<Estudiante> listarEstudiantesAll()
    {
        List<Estudiante> estu = new List<Estudiante>();
        DataSet ds = new ClsComunBase.ClsConexion().ExecuteDataSet("select *
from ESTUDIANTE");
        for (int i = 0; i < ds.Tables[0].Rows.Count; i++)
        {
            Estudiante aux = new Estudiante();
            aux.codEstudiante = ds.Tables[0].Rows[i][0].ToString();
            aux.codRepresentante = ds.Tables[0].Rows[i][1].ToString();
            aux.cedula = ds.Tables[0].Rows[i][2].ToString();
            aux.nombre = ds.Tables[0].Rows[i][3].ToString();
            aux.apellido = ds.Tables[0].Rows[i][4].ToString();
```

```
            aux.direccion = ds.Tables[0].Rows[i][5].ToString();
            aux.telefono = ds.Tables[0].Rows[i][6].ToString();
            aux.celular = ds.Tables[0].Rows[i][7].ToString();
            aux.correo = ds.Tables[0].Rows[i][8].ToString();
            aux.estado = ds.Tables[0].Rows[i][9].ToString();
            estu.Add(aux);
        }
        return estu;
    }

    public List<Estudiante> listarEstudiantesAllPorDocente(string cedulaDocen)
    {
        List<Estudiante> estu = new List<Estudiante>();
        DataSet ds = new ClsComunBase.ClsConexion().ExecuteDataSet("select *
from estudiantesPorDocente where DOCEN_CEDULA='" + cedulaDocen + "'");
        for (int i = 0; i < ds.Tables[0].Rows.Count; i++)
        {
            Estudiante aux = new Estudiante();
            aux.codEstudiante = ds.Tables[0].Rows[i][0].ToString();
            aux.codRepresentante = ds.Tables[0].Rows[i][1].ToString();
            aux.cedula = ds.Tables[0].Rows[i][2].ToString();
            aux.nombre = ds.Tables[0].Rows[i][3].ToString();
            aux.apellido = ds.Tables[0].Rows[i][4].ToString();
            aux.direccion = ds.Tables[0].Rows[i][5].ToString();
            aux.telefono = ds.Tables[0].Rows[i][6].ToString();
            aux.celular = ds.Tables[0].Rows[i][7].ToString();
            aux.correo = ds.Tables[0].Rows[i][8].ToString();
            aux.estado = ds.Tables[0].Rows[i][9].ToString();
            estu.Add(aux);
        }
        return estu;
    }
    public List<Estudiante> listarEstudiantes(string buscar)
    {
        List<Estudiante> estu = new List<Estudiante>();
        DataSet ds = new ClsComunBase.ClsConexion().ExecuteDataSet("select
E.*,R.REP_APELLIDO + R.REP_NOMBRE from ESTUDIANTE E,
REPRESENTANTE R where E.REP_CODIGO=R.REP_CODIGO AND
E.EST_CEDULA like'%" + buscar + "%' order by EST_APELLIDO ");
        for (int i = 0; i < ds.Tables[0].Rows.Count; i++)
        {
```

```
            Estudiante aux = new Estudiante();
            aux.codEstudiante = ds.Tables[0].Rows[i][0].ToString();
            aux.codRepresentante = ds.Tables[0].Rows[i][1].ToString();
            aux.cedula = ds.Tables[0].Rows[i][2].ToString();
            aux.nombre = ds.Tables[0].Rows[i][3].ToString();
            aux.apellido = ds.Tables[0].Rows[i][4].ToString();
            aux.direccion = ds.Tables[0].Rows[i][5].ToString();
            aux.telefono = ds.Tables[0].Rows[i][6].ToString();
            aux.celular = ds.Tables[0].Rows[i][7].ToString();
            aux.correo = ds.Tables[0].Rows[i][8].ToString();
            aux.estado = ds.Tables[0].Rows[i][9].ToString();
            aux.representante = ds.Tables[0].Rows[i][10].ToString();
            estu.Add(aux);
        }
        return estu;
    }

    public List<EstudianteView> listarEstudiantesView(string buscar)
    {
        List<EstudianteView> estu = new List<EstudianteView>();
        DataSet ds = new ClsComunBase.ClsConexion().ExecuteDataSet("select
E.EST_CEDULA,E.EST_APELLIDO,E.EST_NOMBRE,E.EST_TELEFONO,E.ES
T_CELULAR,E.EST_DIRECCION,E.EST_CORREO,R.REP_APELLIDO+'
'+R.REP_NOMBRE, E.EST_ESTADO from ESTUDIANTE E, REPRESENTANTE
R where E.REP_CODIGO=R.REP_CODIGO AND E.EST_CEDULA like'%" +
buscar + "%' order by EST_APELLIDO ");
        for (int i = 0; i < ds.Tables[0].Rows.Count; i++)
        {
            EstudianteView aux = new EstudianteView();
            aux.CEDULA = ds.Tables[0].Rows[i][0].ToString();
            aux.APELLIDO = ds.Tables[0].Rows[i][1].ToString();
            aux.NOMBRE = ds.Tables[0].Rows[i][2].ToString();
            aux.TELEFONO = ds.Tables[0].Rows[i][3].ToString();
            aux.CELULAR = ds.Tables[0].Rows[i][4].ToString();
            aux.DIRECCION = ds.Tables[0].Rows[i][5].ToString();
            aux.CORREO = ds.Tables[0].Rows[i][6].ToString();
            aux.REPRESENTANTE = ds.Tables[0].Rows[i][7].ToString();
            aux.ESTADO = ds.Tables[0].Rows[i][8].ToString();
            estu.Add(aux);
        }
        return estu;
```

```
        }

        public Estudiante listarEstudiantesPorCedula(string buscar)
        {
            Estudiante aux = new Estudiante();
            List<Estudiante> estu = new List<Estudiante>();
            DataSet ds = new ClsComunBase.ClsConexion().ExecuteDataSet("select
E.*,R.REP_APELLIDO + R.REP_NOMBRE from ESTUDIANTE E,
REPRESENTANTE R where E.REP_CODIGO=R.REP_CODIGO AND
E.EST_CEDULA like'%" + buscar + "%' order by EST_APELLIDO ");
            for (int i = 0; i < ds.Tables[0].Rows.Count; i++)
            {
                aux.codEstudiante = ds.Tables[0].Rows[i][0].ToString();
                aux.codRepresentante = ds.Tables[0].Rows[i][1].ToString();
                aux.cedula = ds.Tables[0].Rows[i][2].ToString();
                aux.nombre = ds.Tables[0].Rows[i][3].ToString();
                aux.apellido = ds.Tables[0].Rows[i][4].ToString();
                aux.direccion = ds.Tables[0].Rows[i][5].ToString();
                aux.telefono = ds.Tables[0].Rows[i][6].ToString();
                aux.celular = ds.Tables[0].Rows[i][7].ToString();
                aux.correo = ds.Tables[0].Rows[i][8].ToString();
                aux.estado = ds.Tables[0].Rows[i][9].ToString();
                aux.representante = ds.Tables[0].Rows[i][10].ToString();
                estu.Add(aux);
            }
            return aux;
        }
        public void insertarEstudiante(Estudiante est)
        {

            string SQL = "INSERT INTO ESTUDIANTE VALUES ('" +
est.codRepresentante + "','" + est.cedula + "','" + est.nombre + "','" + est.apellido +
"','" + est.direccion + "','" + est.telefono + "','" + est.celular + "','" + est.correo + "','"
+ est.estado + "','" + est.contrasena + "')";
            new ClsConexion().ExecuteNonQuery(SQL);

            SQL = "EXEC dbo.sp_insertar_document_estudiante";
            new ClsConexion().ExecuteNonQuery(SQL);
        }
        public void actualizarEstudiante(Estudiante est)
        {
```

```
        string SQL = "UPDATE [MATRICULACION].[dbo].[ESTUDIANTE]" +
                     "SET [REP_CODIGO] ='" + est.codRepresentante + "'" +
                       ",[EST_CEDULA] = '" + est.cedula + "'" +
                       ",[EST_NOMBRE] = '" + est.nombre + "'" +
                       ",[EST_APELLIDO] = '" + est.apellido + "'" +
                       ",[EST_DIRECCION] = '" + est.direccion + "'" +
                      " ,[EST_TELEFONO] ='" + est.telefono + "'" +
                     "  ,[EST_CELULAR] = '" + est.celular + "'" +
                     "   ,[EST_CORREO] = '" + est.correo + "'" +
                    "    ,[EST_ESTADO] = '" + est.estado + "'" +
                   " ,[EST_CONTRASENA] = '"+est.contrasena+"'"+
                   "WHERE EST_CODIGO='" + est.codEstudiante + "'";
        new ClsConexion().ExecuteNonQuery(SQL);
    }
    public void eliminarEstudiante(string codEst)
    {

        string SQL = "delete from ESTUDIANTE  where EST_CODIGO='" +
codEst + "'";
        new ClsConexion().ExecuteNonQuery(SQL);
    }


    //para la matricula
    public List<Estudiante> listarEstudiantesAllMatricula()
    {
        List<Estudiante> estu = new List<Estudiante>();
        DataSet ds = new ClsComunBase.ClsConexion().ExecuteDataSet("select *
from listarEstudiantesAllMatricula");
        for (int i = 0; i < ds.Tables[0].Rows.Count; i++)
        {
            Estudiante aux = new Estudiante();
            aux.codEstudiante = ds.Tables[0].Rows[i][0].ToString();
            aux.codRepresentante = ds.Tables[0].Rows[i][1].ToString();
            aux.cedula = ds.Tables[0].Rows[i][2].ToString();
            aux.nombre = ds.Tables[0].Rows[i][3].ToString();
            aux.apellido = ds.Tables[0].Rows[i][4].ToString();
            aux.direccion = ds.Tables[0].Rows[i][5].ToString();
            aux.telefono = ds.Tables[0].Rows[i][6].ToString();
            aux.celular = ds.Tables[0].Rows[i][7].ToString();
```

```
            aux.correo = ds.Tables[0].Rows[i][8].ToString();
            aux.estado = ds.Tables[0].Rows[i][9].ToString();
            estu.Add(aux);
        }
        return estu;
    }
    public Estudiante listarEstudiantesAllMatriculaPorCedula(string cedula)
    {
        Estudiante aux = new Estudiante();
        List<Estudiante> estu = new List<Estudiante>();
        DataSet ds = new ClsComunBase.ClsConexion().ExecuteDataSet("select *
from listarEstudiantesAllMatricula where EST_CEDULA='" + cedula + "'");
        for (int i = 0; i < ds.Tables[0].Rows.Count; i++)
        {
            aux.codEstudiante = ds.Tables[0].Rows[i][0].ToString();
            aux.codRepresentante = ds.Tables[0].Rows[i][1].ToString();
            aux.cedula = ds.Tables[0].Rows[i][2].ToString();
            aux.nombre = ds.Tables[0].Rows[i][3].ToString();
            aux.apellido = ds.Tables[0].Rows[i][4].ToString();
            aux.direccion = ds.Tables[0].Rows[i][5].ToString();
            aux.telefono = ds.Tables[0].Rows[i][6].ToString();
            aux.celular = ds.Tables[0].Rows[i][7].ToString();
            aux.correo = ds.Tables[0].Rows[i][8].ToString();
            aux.estado = ds.Tables[0].Rows[i][9].ToString();
        }
        return aux;
    }
    #endregion
```

Mantenimiento Jornada.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.ec.ClsLogicaNegocio.Entidades;
using com.ec.ClsLogicaNegocio.Vistas;
using System.Data;
using com.ec.ClsComunBase;
```

```
namespace com.ec.ClsLogicaNegocio
{

   public class ClsLogicaNegocio
   {

 #region Jornada

       public void insertarJornada(Jornada est)
       {

          string SQL = "INSERT INTO JORNADA VALUES ('" + est.nombre + "','" +
est.descripcion + "')";
          new ClsConexion().ExecuteNonQuery(SQL);
       }
       public List<Jornada> listarJornada(string buscar)
       {
          List<Jornada> estu = new List<Jornada>();
          DataSet ds = new ClsComunBase.ClsConexion().ExecuteDataSet("select *
from JORNADA J WHERE J.JOR_NOMBRE LIKE '%" + buscar + "%'");
          for (int i = 0; i < ds.Tables[0].Rows.Count; i++)
          {
             Jornada aux = new Jornada();
             aux.codigo = ds.Tables[0].Rows[i][0].ToString();
             aux.nombre = ds.Tables[0].Rows[i][1].ToString();
             aux.descripcion = ds.Tables[0].Rows[i][2].ToString();
             estu.Add(aux);
          }
          return estu;
       }

       //select J.JOR_NOMBRE,J.JOR_DESCRIPCION from JORNADA J
       public List<JornadaView> listarJornadaView(string buscar)
       {
          List<JornadaView> estu = new List<JornadaView>();
          DataSet ds = new ClsComunBase.ClsConexion().ExecuteDataSet("select
J.JOR_NOMBRE,J.JOR_DESCRIPCION from JORNADA J  WHERE
J.JOR_NOMBRE LIKE '%" + buscar + "%'");
          for (int i = 0; i < ds.Tables[0].Rows.Count; i++)
          {
             JornadaView aux = new JornadaView();
```

```
            aux.NOMBRE = ds.Tables[0].Rows[i][0].ToString();
            aux.DESCRIPCION = ds.Tables[0].Rows[i][1].ToString();

            estu.Add(aux);
        }
        return estu;
    }
    public void eliminarJornada(Jornada est)
    {

        string SQL = "delete from JORNADA where JOR_CODIGO='" + est.codigo
+ "'";
        new ClsConexion().ExecuteNonQuery(SQL);
    }
    public void actualizarJornada(Jornada est)
    {

        string SQL = "update JORNADA set JOR_DESCRIPCION='" +
est.descripcion + "',JOR_NOMBRE='" + est.nombre + "' where JOR_CODIGO='" +
est.codigo + "'";
        new ClsConexion().ExecuteNonQuery(SQL);
    }
    #endregion
```

Mantenimiento Especialidad.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.ec.ClsLogicaNegocio.Entidades;
using com.ec.ClsLogicaNegocio.Vistas;
using System.Data;
using com.ec.ClsComunBase;

namespace com.ec.ClsLogicaNegocio
{

  public class ClsLogicaNegocio
  {
```

```
#region Especialidad

        public void insertarEspecialidad(Especialidad est)
        {
            string SQL = "insert into ESPECIALIDAD values ('" + est.NOMBRE + "','"
+ est.DESCRIPCION + "','" + est.ESTADO + "')";
            new ClsConexion().ExecuteNonQuery(SQL);
        }
        public List<Especialidad> listarEspecialidad(string buscar)
        {
            List<Especialidad> estu = new List<Especialidad>();
            DataSet ds = new ClsComunBase.ClsConexion().ExecuteDataSet("SELECT
* FROM ESPECIALIDAD ES WHERE ES.ESP_NOMBRE LIKE '%" + buscar +
"%'");
            for (int i = 0; i < ds.Tables[0].Rows.Count; i++)
            {
                Especialidad aux = new Especialidad();
                aux.CODIGO = ds.Tables[0].Rows[i][0].ToString();
                aux.NOMBRE = ds.Tables[0].Rows[i][1].ToString();
                aux.DESCRIPCION = ds.Tables[0].Rows[i][2].ToString();
                aux.ESTADO = ds.Tables[0].Rows[i][3].ToString();
                estu.Add(aux);
            }
            return estu;
        }

        public List<EspecialidadView> listarEspecialidadView(string buscar)
        {
            List<EspecialidadView> estu = new List<EspecialidadView>();
            DataSet ds = new ClsComunBase.ClsConexion().ExecuteDataSet("SELECT
ES.ESP_NOMBRE,ES.ESP_DESCRIPCION,ES.ESP_ESTADO FROM
ESPECIALIDAD ES WHERE ES.ESP_NOMBRE LIKE '%" + buscar + "%'");
            for (int i = 0; i < ds.Tables[0].Rows.Count; i++)
            {
                EspecialidadView aux = new EspecialidadView();
                aux.NOMBRE = ds.Tables[0].Rows[i][0].ToString();
                aux.DESCRIPCION = ds.Tables[0].Rows[i][1].ToString();
                aux.ESTADO = ds.Tables[0].Rows[i][2].ToString();

                estu.Add(aux);
            }
```

```
        return estu;
    }

    public void eliminarEspecialidad(string codigo)
    {

        string SQL = "delete from ESPECIALIDAD where ESP_CODIGO='" +
codigo + "'";
        new ClsConexion().ExecuteNonQuery(SQL);
    }

    public void actualizarEspecialidad(Especialidad ESP)
    {

        string SQL = "update ESPECIALIDAD set ESP_NOMBRE='" +
ESP.NOMBRE + "', ESP_DESCRIPCION='" + ESP.DESCRIPCION + "' where
ESP_CODIGO='" + ESP.CODIGO + "'";
        new ClsConexion().ExecuteNonQuery(SQL);
    } #endregion
```

Mantenimiento Matería.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.ec.ClsLogicaNegocio.Entidades;
using com.ec.ClsLogicaNegocio.Vistas;
using System.Data;
using com.ec.ClsComunBase;

namespace com.ec.ClsLogicaNegocio
{

  public class ClsLogicaNegocio
  {
#region MATERIA

    public void insertarMateria(Materia mat)
```

```
        {

            string SQL = "INSERT INTO MATERIA VALUES ('" + mat.codDocente +
"','" + mat.nombre + "')";
            new ClsConexion().ExecuteNonQuery(SQL);
        }
        public List<MateriaView> listarMateriaView()
        {
            List<MateriaView> estu = new List<MateriaView>();
            DataSet ds = new ClsComunBase.ClsConexion().ExecuteDataSet("select *
from MATERIA_DOCENTE");
            for (int i = 0; i < ds.Tables[0].Rows.Count; i++)
            {
                MateriaView aux = new MateriaView();
                aux.DOCENTE = ds.Tables[0].Rows[i][0].ToString() + " " +
ds.Tables[0].Rows[i][1].ToString();
                aux.MATRIA = ds.Tables[0].Rows[i][2].ToString();

                estu.Add(aux);
            }
            return estu;
        }
        public List<MateriaDocente> listarMateriaDocente(string buscar)
        {
            List<MateriaDocente> estu = new List<MateriaDocente>();
            DataSet ds = new ClsComunBase.ClsConexion().ExecuteDataSet("SELECT
M.PEN_MATERIA FROM MATERIA M WHERE DOCEN_CODIGO='" + buscar
+ "'");
            for (int i = 0; i < ds.Tables[0].Rows.Count; i++)
            {
                MateriaDocente aux = new MateriaDocente();
                aux.MATERIA = ds.Tables[0].Rows[i][0].ToString();
                estu.Add(aux);
            }
            return estu;
        }
        public List<Materia> listarMateriaAll()
        {
            List<Materia> estu = new List<Materia>();
            DataSet ds = new ClsComunBase.ClsConexion().ExecuteDataSet("SELECT
* FROM MATERIA M ORDER BY M.PEN_MATERIA ASC");
```

```
        for (int i = 0; i < ds.Tables[0].Rows.Count; i++)
        {
            Materia aux = new Materia();
            aux.codMateria = ds.Tables[0].Rows[i][0].ToString();
            aux.codDocente = ds.Tables[0].Rows[i][1].ToString();
            aux.nombre = ds.Tables[0].Rows[i][2].ToString();
            estu.Add(aux);
        }
        return estu;
    }


    #endregion
```

Mantenimiento Curso.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.ec.ClsLogicaNegocio.Entidades;
using com.ec.ClsLogicaNegocio.Vistas;
using System.Data;
using com.ec.ClsComunBase;

namespace com.ec.ClsLogicaNegocio
{

    public class ClsLogicaNegocio
    {


#region CURSOS

        public void insertarCurso(Curso curso)
```

```
        {

            string SQL = "INSERT INTO [MATRICULACION].[dbo].[CURSOS] " +
           " ([CUR_NOMBRE]" +
           " ,[CUR_NIVEL]" +
          " ,[CUR_NUM_ESTUDIANTES]" +
           " ,[CUR_PARALELO]" +
           " ,[CUR_ESTADO])" +
            " VALUES ('" + curso.nombre + "','" + curso.nivel + "','" +
curso.numeroEstudiantes + "','" + curso.paralelo + "','" + curso.estado + "')";
            new ClsConexion().ExecuteNonQuery(SQL);
        }
        public List<Curso> listarCursoRegistrado(string curso, string paralelo)
        {
            List<Curso> estu = new List<Curso>();
            DataSet ds = new ClsComunBase.ClsConexion().ExecuteDataSet("select *
from CURSOS C WHERE C.CUR_NIVEL='" + curso + "' AND
C.CUR_PARALELO='" + paralelo + "'");
            for (int i = 0; i < ds.Tables[0].Rows.Count; i++)
            {
                Curso aux = new Curso();
                aux.codCurso = ds.Tables[0].Rows[i][0].ToString();
                aux.nivel = ds.Tables[0].Rows[i][2].ToString();
                aux.paralelo = ds.Tables[0].Rows[i][4].ToString();
                estu.Add(aux);
            }
            return estu;
        }

        public List<Curso> listarCursoRegistrado()
        {
            List<Curso> estu = new List<Curso>();
            DataSet ds = new ClsComunBase.ClsConexion().ExecuteDataSet("select *
from CURSOS");
            for (int i = 0; i < ds.Tables[0].Rows.Count; i++)
            {
                Curso aux = new Curso();
                aux.codCurso = ds.Tables[0].Rows[i][0].ToString();
                aux.nivel = ds.Tables[0].Rows[i][2].ToString();
                aux.paralelo = ds.Tables[0].Rows[i][4].ToString();
                estu.Add(aux);
```

```
            }
            return estu;
        }
        public List<CursoView> listarCursoPorNivel(string buscar)
        {
            List<CursoView> estu = new List<CursoView>();
            DataSet ds = new ClsComunBase.ClsConexion().ExecuteDataSet("select
C.CUR_NIVEL,C.CUR_PARALELO,C.CUR_NUM_ESTUDIANTES,B.BEC_NU
M,C.CUR_ESTADO from CURSOS C, BECA B WHERE
B.CUR_CODIGO=C.CUR_CODIGO  AND C.CUR_NIVEL LIKE '%" + buscar +
"%'");
            for (int i = 0; i < ds.Tables[0].Rows.Count; i++)
            {
                CursoView aux = new CursoView();
                aux.NOMBRE = ds.Tables[0].Rows[i][0].ToString();
                aux.PARALELO = ds.Tables[0].Rows[i][1].ToString();
                aux.NUMERO_MAXIMO_DE_ESTUDIANTES =
ds.Tables[0].Rows[i][2].ToString() + " Cupos";
                aux.NUMERO_MAXIMO_DE_BECAS =
ds.Tables[0].Rows[i][3].ToString() + " Cupos";
                aux.ESTADO = ds.Tables[0].Rows[i][4].ToString();
                estu.Add(aux);
            }
            return estu;
        }

        public Curso ultimoCursoReg()
        {
            Curso aux = new Curso();
            List<Curso> estu = new List<Curso>();
            DataSet ds = new ClsComunBase.ClsConexion().ExecuteDataSet("select
top(1)* from CURSOS C ORDER BY C.CUR_CODIGO DESC");
            for (int i = 0; i < ds.Tables[0].Rows.Count; i++)
            {

                aux.codCurso = ds.Tables[0].Rows[i][0].ToString();

            }
            return aux;
        }
        #endregion
```

Enlaces Bibliográficos

- ("Documentos Legales y Normativos | Ministerio de Educación," n.d.)
- ("René Ramírez destacó el proceso de transformación de la educación superior en su visita a Francia | Secretaría de Educación Superior, Ciencia, Tecnología e Innovación," n.d.)
- ("El BID se interesa por la reforma educativa ecuatoriana | Ministerio Coordinador de Conocimiento y Talento Humano," n.d.)
- http://www.educarecuador.gob.ec/index.php/alumnos.
- http://educacion.gob.ec/inscripciones-y-matriculas/
- http://www.google.com.ec/url?sa=t&rct=j&q=&esrc=s&frm=1&source=web&cd=5&cad=rja&uact=8&sqi=2&ved=0CDsQFjAE&url=http%3A%2F%2Fwww.oocities.org%2Ffarp81%2FDocumento_de_Estandares_de_Programacion_Final.doc&ei=Guv7U7LSHq_NsQTyxYLQBQ&usg=AFQjCNEawYRENc8hDfPTIUv4eJj6qnS6FQ&sig2=dg_g1e5-w6m_UBBH2skkQg&bvm=bv.73612305,d.cWc
- http://www.codecompiling.net/files/slides/IS_clase_04_diseno_UI.pdf