I. EL PROYECTO

I.1 PRESENTACION DEL PROYECTO

I.1.1 Titulo

Mejoramiento de la administración y gestión académica de las unidades Educativas del nivel secundario, para la dirección distrital Cercado Red Bel grano.

I.1.2 Área del proyecto

Mejoramiento en la administración y gestión académica Educativa, comunicación y acceso a nuevas tecnologías.

I.1.3 Responsable del Proyecto

Carrera de Ingeniería Informática - Taller III – grupo 2

I.1.4 Entidades asociadas

Establecimientos Educativos de la Red Belgrano distrito Cercado.

I.1.5 Compromiso del director de Proyecto

Yo, Laura Elisa Quispe Ojeda directora del proyecto "MAGABEL", acepto la responsabilidad de cumplir los compromisos de ejecución del proyecto "Mejoramiento de la administración y gestión académica de las unidades Educativas del nivel secundario, para la dirección distrital Cercado Red Bel grano", en caso de aprobarse.	
Laura Elisa Quispe Ojeda **Nombre de Director**	**Firma de director**

Tabla 1. Compromiso de la Directora de Proyecto

I.1.6 Duración del Proyecto

El proyecto tiene una duración de 6 meses.

I.1.7 Director responsable del Proyecto

NOMBRE	TALLER/ GRUPO
Laura Elisa Quispe Ojeda	Taller III grupo 2
Email	**Telefono**
laurita_tja@hotmail.com	6661668 - 76194094

Tabla 2. Directora responsable del proyecto

I.2 DESCRIPCION DEL PROYECTO

I.2.1 Resumen Ejecutivo del Proyecto

En la actualidad la planificación administrativa, académica e infraestructura escolar de cada distrito, se hace en base a información estadística específica primeramente a nivel departamental y luego a nivel distrital, es decir que cada distrito debe encargarse de conseguir la información acerca de la situación actual en cada una de sus unidades educativas, información que se presenta en el cuadro resumen ejecutivo.

Dado que el departamento de Tarija cuenta con once distritos, y cada distrito cuenta con varios núcleos educativos la obtención de dicha información tiende a ser demasiado morosa.

El presente trabajo "MEJORAMIENTO DE LA ADMINISTRACION Y GESTION ACADEMICA DE LAS UNIDADES EDUCATIVAS DEL NIVEL SECUNDARIO PARA LA DIRECCION DISTRITAL CERCADO. RED BEGRANO", comprende un sistema dirigido A directores de unidades educativas el cual permitirá facilitar en gran medida la información acerca de, la administración del personal docente, administración y registro de estudiantes, organización educativa e infraestructura por lo tanto una efectiva planificación administrativa y académica.

Para lograr este propósito se pretende:

- Desarrollar un sistema informático para la generación de reportes, estadísticas, mejor administración y otros requerimientos.
- Realizar capacitaciones del personal en el uso del software, para el uso eficiente.

La información automatizada proporcionara estadísticas que apoyara en una efectiva planificación, permitiendo a sus autoridades mejorar en la toma de decisiones.

I.2.2 Descripción, Fundamentación y Justificación del Proyecto

La Tecnología de Información (TI) nació como soporte a las necesidades dentro de las organizaciones, entre sus aplicaciones están los sistemas de información automatizados que presentan información con características de importancia, relevancia, claridad, sencillez y oportunidad de tal forma que sea útil para las personas a quienes se les entrega. Esto se refiere a que cuando los usuarios reciben los datos, éstos son preparados e introducidos a la Base de Datos de la empresa mediante una aplicación de computadora. El procesamiento de estos datos depende de los propósitos de cada aplicación de tal forma que se genere la información adecuada y se presente los respectivos reportes.

El presente proyecto tiene por finalidad optimizar los procesos de la administración y gestión académica de las unidades Educativas del nivel secundario, para la dirección distrital Cercado Red Bel grano. Para lograr este fin se realizara los siguientes componentes: un Sistema Informático y la capacitación al personal de la Unidad Educativa para el uso del mismo. El software que se desarrolla, basado en la Tecnología de Información y Comunicación, es un sistema de información automatizado, utilizado para la administración y la generación de reportes de la gestión, con el propósito de registrar y manejar la información necesaria para la Unidad Educativa sobre todo para el director. Esta información será manejada por cada usuario en las máquinas que dicha Unidad Educativa destine para su uso, pero la misma será almacenada en un sólo equipo (arquitectura Cliente/Servidor).

En la actualidad los directores distritales, al comienzo de cada gestión escolar se ven en la gran tarea de realizar la planificación administrativa y académica de su distrito, la asignación de personal docente a cada una de las unidades educativas (U.E.) o contratación de personal, es una de las tareas más morosas, por falta de información acerca de la situación actual de las U.E., información acerca del estudiantado e

infraestructura, le imposibilita una organización oportuna y dar comienzo normal de la gestión escolar.

Necesariamente cada director distrital debe supervisar y hacer un seguimiento en cada una de las U.E. para conocer a cabalidad las necesidades de cada U.E. y conseguir la información oportuna.

Los directores de cada U.E. pasan por similar situación, ya que son estos los encargados de realizar los reportes de sus establecimientos acerca de la cantidad de estudiantado, pero la supervisión y la realización de cada uno de estos reportes tiende a ser muy tediosa y morosa ya que llegan a su culminación cuando la gestión escolar esta por culminar también, otro problema detectado por parte de los directores de la U.E. es la planificación de los horarios y asignación de los docentes a cada curso.

Dicha falta de información da como resultado mucha desorganización, por ejemplo; docentes con un nivel y especialización dando clases en otro nivel, estudiantes pasando clases en ambientes incómodos(infraestructura inadecuada).

Por esta razón es una necesidad para la U.E. ejecutar el presente proyecto que pretende optimizar estos procesos mediante:

- El desarrollo de un sistema informático para la administración, generación de reportes, estadísticas y otros según requerimiento del director de la Unidad Educativa.

- Capacitar al personal de la Unidad Educativa en el uso del software.

I.3 Objetivos

I.3 .1 Objetivo General

Optimizar la administración y gestión académica de las Unidades Educativas del nivel secundario para la dirección distrital Cercado Red Belgrano.

I.3.2 Objetivos Específicos

- Desarrollar el Sistema Informático para la administración, generación de reportes, estadísticas y otros según requerimientos.
- Capacitar al personal de la Unidad Educativa en el uso del software.

I.3.3 Metodología

En el presente proyecto se contempla la realización de dos componentes: un sistema informático para la generación de reportes, estadísticas, mejor administración y otros según los requerimientos, y la capacitación del personal en el uso del software.

Las metodologías a utilizar se describen a continuación:

I.3.3.1 Desarrollar el Sistema Informático para la administración, generación de reportes, estadísticas y otros según requerimientos

❖ Se utilizará la metodología RUP (Racional Unified Process), que mejora considerablemente la calidad de desarrollo del sistema, ya que la misma utiliza el Lenguaje Unificado de Modelado (UML) para preparar todos los esquemas de un sistema software.

RUP es un proceso ágil de desarrollo que se repite a lo largo de una serie de ciclos que constituyen la vida de un sistema. Cada ciclo concluye con una versión del producto para los clientes

El flujo de trabajo fundamental tiene los contemplar los siguientes pasos:

- Requerimientos: Trasladando las necesidades del negocio a un sistema automatizado.
- Análisis y Diseño: Trasladando los requerimientos dentro de la arquitectura de software.
- Programación e Implementación: Creando software que se ajuste a la arquitectura y que tenga el comportamiento deseado.
- Pruebas: Asegurándose que el comportamiento requerido es el correcto y que todo los solicitado está presente.

❖ Se realizara un modelo de datos basado en estándares que nos permitan mantener la integridad de la información.

❖ La programación del Sistema Informático se realizó en base a patrones de diseño que nos permiten la reutilización de componentes y así obtener la estabilidad, escalabilidad y seguridad del sistema

I.3.3.2 Capacitación del personal en el uso del software.

❖ Se realizara ejemplos y demostraciones utilizando tecnología moderna (computadora, data show, pizarras acrílicas, manuales de usuario y sistema.).pretendiendo así llegar a una mayor comprensión de los usuarios

I.3.4 Bibliografía Consultada para la realización del Perfil de Proyecto

KHAWAR ZAMAN AHMED CARY E. UMRYSH	Developing Eterprise Aplications with Java and UML Editorial Addison Wesley Longman Diciembre 2001.
KIM HAMILTON, RUSSELL MILES, O'REILLY	Learning UML 2.0 April 2006
ÖVERGAARD KARIN PALMKVIST GUNNAR	Use Case Patterns and Blueprints Editorial Addison Wesley Professional Edition November 12, 2004
SHMULLER JOSEP	Aprendiendo UML en 24 horas Editorial Prentice Hall
VILALTA JOSEPH	UML Guía Visual Editorial Vilalta Consultores 2001
UNIVERSIDAD AUTÓNOMA "JUAN	Plan de Desarrollo Estratégico

I.3.5 Resultados esperados

- Sistema informático para la mejor administración, generación de reportes, estadísticas y otros según requerimientos, desarrollado con el cual se optimizaran todos los procesos de obtención de información, documentación y manuales.
- Personal administrativo de las Unidades Educativas de El Puente capacitado en el uso del software implementado, con lo que se pretende formar al personal en el manejo de las utilidades del sistema, con el objetivo de que la formación recibida les permita manejar el sistema sin dificultad.

I.3.6 Transferencia de resultados

I.3.6.1 Medios y estrategias para la transferencia de resultados.

- ❖ Capacitar a los diferentes directores de las Unidades Educativas pertenecientes a la Red Belgrano de la Dirección distrital de Cercado en la utilización de esta aplicación informática, resaltando sus ventajas administrativas
- ❖ Capacitaciones a las diferentes Autoridades Educativas para difusión del sistema.
- ❖ Presentación final del sistema informático a las autoridades universitarias.

I.3.6.2 Grupo de beneficiarios de los resultados

- ❖ Director de la Unidad Educativa
- ❖ Personal docente y administrativo
- ❖ Estudiantes

I.3.8 Marco Lógico del Proyecto

Resumen Narrativo del Proyecto	Indicadores	Medios de Verificación	Supuestos
Fin Contribuir al proceso de formación de conocimientos, destrezas, conductas y valores para los estudiantes del Distrito Cercado Red Belgrano.	Al segundo año de la implementación del proyecto, se cuenta con la información vital en la toma de decisiones en la organización curricular en un 70%.	Informes de administración de los directores de las Unidades Educativas del distrito Cercado, organizados y acabados hasta 30 días después de iniciada la gestión escolar.	El proyecto es implementado en la totalidad de las Unidades Educativas del distrito Cercado
Objetivo General (Propósito) Administración y gestión académica de las Unidades Educativas del nivel secundario para la dirección distrital Cercado Red Belgrano optimizada.	• Al finalizar la ejecución del proyecto en noviembre de 2010, el personal administrativo de las Unidades Educativas de Cercado cuentan con información oportuna de estudiantes, memorándums asignados, personal docente, asignación de horarios, inscripciones y otros en un 70%, en relación al año base 2008	• Informe de conformidad sobre la funcionalidad del Sistema de administración sobre la gestión académica proporcionado y firmado por las Autoridades de la Unidad Educativa.	• Las autoridades de la Unidad Educativa manifiesta apoyo y disposición para la ejecución del proyecto. • Los equipos de computación de cada Unidad Educativa cuentan con el hardware necesario para el software presentado

Objetivos Específicos (Componentes)			
1. Sistema Informático para la administración de la gestión académica, generación de reportes y otros según requerimientos, desarrollado.	**1.1.** En fecha 30 de julio de 2010 se concluye en un 90% la etapa de análisis y diseño del Sistema. **1.2** En fecha 30 de Septiembre de 2010 se concluye en un 80% la etapa de construcción del Sistema Informático. **1.3** En junio de 2011 se pone en funcionamiento el sistema de gestión académica, optimizando un 60% de los procesos realizados en la administración.	**1.1.1** Documento entregado y corregido por los docentes de la materia Taller III **1.2.1** Sistema presentado al asesor y a los Docentes de Taller III **1.3.1** Carta de aceptación y conformidad firmada, por parte de las autoridades educativas de la Unidad Educativa, posterior a la implementación	• Predisposición del personal administrativo y autoridades educativas para brindar información requerida para el desarrollo del sistema. • Reportes generados, son de satisfacción de las autoridades educativas

2. Personal capacitado en el uso del software, permite un registro eficiente.	**2.1** En julio del 2011 se capacita a un 50% del personal administrativo y autoridades educativas, en el manejo del sistema. **2.2** Al final de la capacitación, 60% de los usuarios consiguen manejar sin ninguna dificultad el sistema.	**2.1.1** Lista de asistencia de los participantes de la capacitación. **2.1.2** Certificados otorgados a los participantes de la capacitación.	• Asistencia masiva al curso de capacitación por parte de las autoridades educativas (director de U.E.) y personal administrativo. • La capacitación se lleva a cabo en la fecha establecida. • El personal administrativo y autoridades educativas hacen uso del sistema sin ninguna dificultad.
Actividades **Componente 1** **1. Determinación de requisitos** 1.1 Elaboración de documento de requerimiento de sistema 1.2 Entrevista con autoridades involucradas 1.3 Elaboración de cuestionarios	**Componente 1** • **20000** 100 bs • **22000** 1140 bs • **25000** 8380 bs • **30000** 2400 bs • **32000** 1194 bs • **39000** 303 bs • **40000** 1000 bs Total **componente 1** 14517 bs.	• Informe del presupuesto de gastos.	• Existe disponibilidad de las Autoridades educativas para brindar información requerida • Cumplimiento del

2. Análisis y diseño del sistema 2.1 Análisis y diseño del sistema mediante los diagramas UML con la metodología RUP 2.2 Análisis y diseño de la base de datos del Sistema. 2.3 Elaboración y corrección de la documentación del análisis y diseño **3. desarrollo del sistema** 3.1 Diseño del Prototipo del Sistema. 3.2 Programación del Sistema. 3.3 Diseño grafico **4. validación del software** 4.1 Pruebas al sistema´ **5. elaboración del informe final del sistema** 5.1 Instaladores del sistema 5.2 Elaboración de manuales de			calendario de actividades

usuario e instalación 5.3 Presentación final Entrega del sistema, documentación y manuales **Componente 2** **6. Capacitación al personal** 6.1 Elaboración del material de capacitación. 6.2 Preparación del ambiente para la capacitación. 6.3 Realizar la capacitación.	**Componente 2** • **23000** 540 bs **Total componente 2** 540 bs. **Total proyecto** 15057 bs.		

Tabla 4. Matriz de Marco Logico

I.4 Presupuesto / Justificación

ITEM	RUBROS	Aporte Universidad	Otro Aporte	TOTAL (Bs)
20000	**SERVICIOS NO PERSONALES**			100
	21000. Servicios Básicos			
	22000. Servicios de transporte			1140
	23000. Alquileres			540
	24000. Mantenimiento y reparación			
	25000. Servicios Profesionales y Comerciales			8380
	1.1.1 Sub total rubro			**10160**
30000	**MATERIALES Y SUMINISTROS**			
	31000. Alimentos y Productos Forestales			2400
	32000. Productos de Papel, Cartón e Impresos			1194
	33000. Textiles y Vestuario.			
	34000. Productos Químicos, Combustibles y Lubricantes			
	39000. Productos Varios.			303
	1.1.2 Sub total rubro			**3897**
40000	**ACTIVOS REALES**			

	43000. Maquinaria y Equipo.			1000
	46000. Descripción de estudios y proyectos para inversión			
	49000. Otros Activos			
	1.1.3 Sub total rubro			**1000**
	1.1.4 TOTAL			15057

1) GRUPO 20000. SERVICIOS NO PERSONALES

b) SUB GRUPO 21000. Descripción de los gastos de servicios básicos

Partida	**Tipo de servicio básico ***	**Costo**	**Tiempo mes**	**Costo Total**
21100	Comunicación	100		100
21200	Energía Eléctrica			
21300	Agua			
21400	Servicios Telefónicos			
Total				100

* Se refiere principalmente a los gastos por servicios; como: servicio de correo, radiogramas, servicio telefónico, fax, Internet.

c) SUB GRUPO 22000. Descripción de los gastos de viajes y transporte de personal

Partida	**Personal**	**Lugar**	**Nº de viajes**	**Costo unitario***	**Costo total**

22100	Pasajes		6	30	180
Total					180

* En el caso de pasajes debe indicarse el costo de ida y vuelta (costo unitario), indicando el número de viajes.

Partida	**Personal**	**Lugar**	**Duración (días)**	**Costo unitario***	**Costo total**
22200	Viáticos				
22300	Fletes y Almacenamientos				
22600	Transporte de Personal	Dpto. de Investigación Ciencias y Tecnología	480	2	960
Total					960
Total sub grupo 22000					1140

* En el caso de los viáticos, debe considerarse la escala establecida por la UAJMS.

d) SUB GRUPO 23000. Descripción de los gastos por concepto de alquileres de equipos y maquinarias

Partida	**Alquiler de equipo y maquinaria**	**Costo unitario**	**Tiempo mes**	**Costo total**
23100	Alquiler de Edificios			
23200	Alquiler de Equipos y Maquinaria ➢ 30 equipos de Computación	6	3 días	540
23300	Alquiler de Tierras y Terrenos			

Total			540

* Se refiere principalmente a los gastos por el uso de edificios y equipos y maquinaria en general

e) SUB GRUPO 24000. Descripción mantenimiento y reparación

Partida	**Mantenimiento y reparación de equipo y maquinaria**	**Costo unitario**	**Tiempo mes**	**Costo total**
24100	Mantenimiento y Reparación de Edificios y Equipos			
24300	Otros Gastos por Mantenimiento y Reparación			
Total				

* Se refiere principalmente a los gastos por el mantenimiento y reparación de edificios y equipos y maquinaria en general

f) SUB GRUPO 25000. Descripción de los gastos en servicios profesionales y comerciales

Partida	**Tipo de servicio profesional y comercial** *	**Cantidad**	**Costo unitario**	**Tiempo mes**	**Costo total**
25500	Publicidad				
25600	Imprenta ➢ Certificado de Participación ➢ Empastados	60 9	8 100		480 900
25700	Capacitación de Personal				
25800	Estudios e Investigaciones Para				

	Proyectos de Inversión				
25810	Consultores por Producto ➢ Consultor Informático en Análisis de Sistema ➢ Consultor de Diseño Grafico	1 1	500 3000	8 1	4000 3000
25820	Consultores en Línea ➢ Director	1		0	0
Total					8380

* Se refiere a gastos por servicios profesionales de asesoramiento especializado, se incluyen, estudios, investigaciones, publicidad, imprenta, fotocopias, capacitación de personal y otros ejecutados por terceros.

2) GRUPO 30000. MATERIALES Y SUMINISTROS

g) SUB GRUPO 31000. Descripción de los gastos Alimentos y Productos Agroforestales

Partida	**Tipo de material** *	**Cantidad**	**Costo/Unitario**	**Total**
31110	Refrigerios y Gastos Administrativos ➢ Varios	960	2.5	2400
31200	Alimento para Animales			
31300	Productos Agroforestales y Pecuarios			

Total			2400

* Se refiere a la adquisición de materiales y bienes como: alimentos y productos agroforestales, alimentos y bebidas para personas (indicar el total de refrigerios), alimentos para animales, productos pecuarios.

h) SUB GRUPO 32000. Descripción del gasto de Productos de Papel, Cartón e Impresos

Partida	Tipo de material *	Cantidad	Costo/Unitario	Total
32100	Papel de Escritorio ➢ Resma de Hojas Bond	15	40	600
32200	Productos de Artes Graficas, Papel y Cartón ➢ Tinta Negra ➢ Tinta de Color	6 9	45 36	270 324
32300	Libros y Revistas			
32400	Textos de Enseñanza			
Total				1194

* Se refiere a la adquisición de; papel y cartón en sus diversas formas y clases, impresos y publicaciones, periódicos, revistas, libros, fotocopias, etc.

i) SUB GRUPO 33000. Descripción del gasto en textiles y vestuario

Partida	Productos textiles y vestuarios	Cantidad	Costo/Unitario	Total
33100	Hilados y Telas			
33200	Confecciones Textiles			

33300	Prendas de vestir			
33400	Calzados			
Total				

* Se refiere principalmente a los gastos por vestuario uniformes, ropa de trabajo

j) SUB GRUPO 34000. Combustibles, Productos Químicos, Farmacéuticos y Otros

Partida	Combustibles, Productos Químicos, Farmacéuticos y Otros	**Cantidad**	**Costo/Unitario**	**Total**
34110	Combustibles y Lubricantes para Consumo			
34200	Productos químicos y Farmacéuticos			
34400	Productos de Cuero y Caucho			
34500	Productos de Minerales no Metálicos y Plásticos			
34600	Productos Metálicos			
34700	Minerales			
34800	Herramientas Menores			
Total				

* Se refiere a gastos de combustibles, químicos, productos farmacéuticos, llantas etc.

k) SUB GRUPO 39000. Descripción del gasto en productos varios

Partida	Productos de cuero y caucho	Cantidad	Costo/Unitario	Total
39100	Material de Limpieza			
39400	Instrumental Menor Médico - Quirúrgico			
39500	Útiles de Escritorio y de Oficina ➢ Agenda Personal ➢ Memoria Flash ➢ Bolígrafos, Lapiceras ➢ Cd's, DVD's	 3 2 6 10	 25 90 3 3	 75 180 18 30
39700	Útiles y Materiales Eléctricos			
39800	Otros Repuestos y Accesorios			
Total				303

*Se refiere principalmente a los gastos por productos de limpieza, todo le referente a la funcionamiento de la oficina en material de escritorio.

3) GRUPO 40000. ACTIVOS REALES

l) SUB GRUPO 43000. Descripción del gasto de Maquinaria y Equipo

Partida	Tipos de productos	Cantidad	Costo/Unitario	Total
43100	Equipo de Oficina y Muebles ➢ Cámara digital	 1	 1000	 1000
43200	Maquinaria y Equipo de Producción			
43300	Equipos de Transporte, Tracción y			

	Elevación			
43400	Equipo Medico y de Laboratorio			
43700	Otra Maquinaria y Equipo			
Total				1000

* Se refiere principalmente a los gastos por muebles y enseres, equipo de oficina, comunicación, equipamiento.

m) SUB GRUPO 46000. Descripción de estudios y proyectos para inversión

Partida	**Productos textiles y vestuarios**	**Cantidad**	**Costo/Unitario**	**Total**
46100	Para Construcción de Bienes de Dominio Privado			
Total				

* Se refiere principalmente a los gastos por servicios de terceros para la realización de investigaciones y otras actividades técnico – Profesionales necesarias para la construcción y mejoramiento de bienes.

n) SUB GRUPO 49000. Descripción del gasto de Otros Activos

Partida	**Tipos de productos ***	**Cantidad**	**Costo/Unitario**	**Total**
49100	Activos Intangibles			
Total				

* Se refiere a los gastos en la compra de sotfwear, licencias.

Anexo 1 Cuadro de Involucrados

ACTORES	INTERES	PROBLEMAS	RECURSOS Y MANDATOS
Director de la unidad educativa	Contar con la información necesaria, para una organización académica a inicio de gestión en las Unidades Educativas de educación Formal	La falta de información oportuna ocasiona retrasos en la organización académica debido a: • Organización académica -Horarios -Cantidad estudiantil -Creación de nuevos paralelos -Solicitud temprana de nuevos ítems a la dirección distrital según requerimiento. • Infraestructura -Planificación de distribución de aulas -Aulas para paralelos según cantidad de estudiantes El modo actual de obtención de información tiende a ser muy tedioso e insuficiente. • Existe un retardo en la generación de reportes acerca de los docentes y su asignación de horarios	**R.** Información precisa del personal docente **R.** Registro de docentes actualizada **R.** Registro de estudiantes mejorada **M.** Planificación y organización académica de la Unidad Educativa, permite tomar decisiones más acertadas y eficaces **M.** Identificación, análisis, y organización temprana y efectiva de la Unidad Educativa, personal docente y estudiantes

Tabla nº cuadro Involucrados

		• La manera actual como se realiza el registro estudiantil (matricula), no permite tener estadísticas requeridas necesarias a la hora de asignación docente.	
Personal docente y administrativo	Contar con Horarios y aulas asignadas, según el nivel y especialización al que pertenece, a inicio de gestión, en una sola Unidad Educativa de ser posible.	• No cuentan con un horario y aula fijo asignado a principio de gestión. • Docentes asignados a más de un curso en un mismo horario	**R.** Mayor énfasis en el registro estudiantil **M.** Asignación docente organizada de acuerdo a especialización y nivel evita choque de horarios
Estudiantes	Contar con un horario, aula, docente asignados fijos para asegurar su aprendizaje y mayor aprovechamiento.	• Cambio de docente después de iniciado el avance • Bajo nivel de aprendizaje de materia gracias a la asignación inadecuada de docentes • Aulas insuficientes para la cantidad de estudiantes	R. Docentes asignados en las materias que pertenecen a su nivel y especialidad, desde un comienzo de gestión. M. Rendimiento escolar realzado notablemente.

Anexo 2 Árbol de Problemas

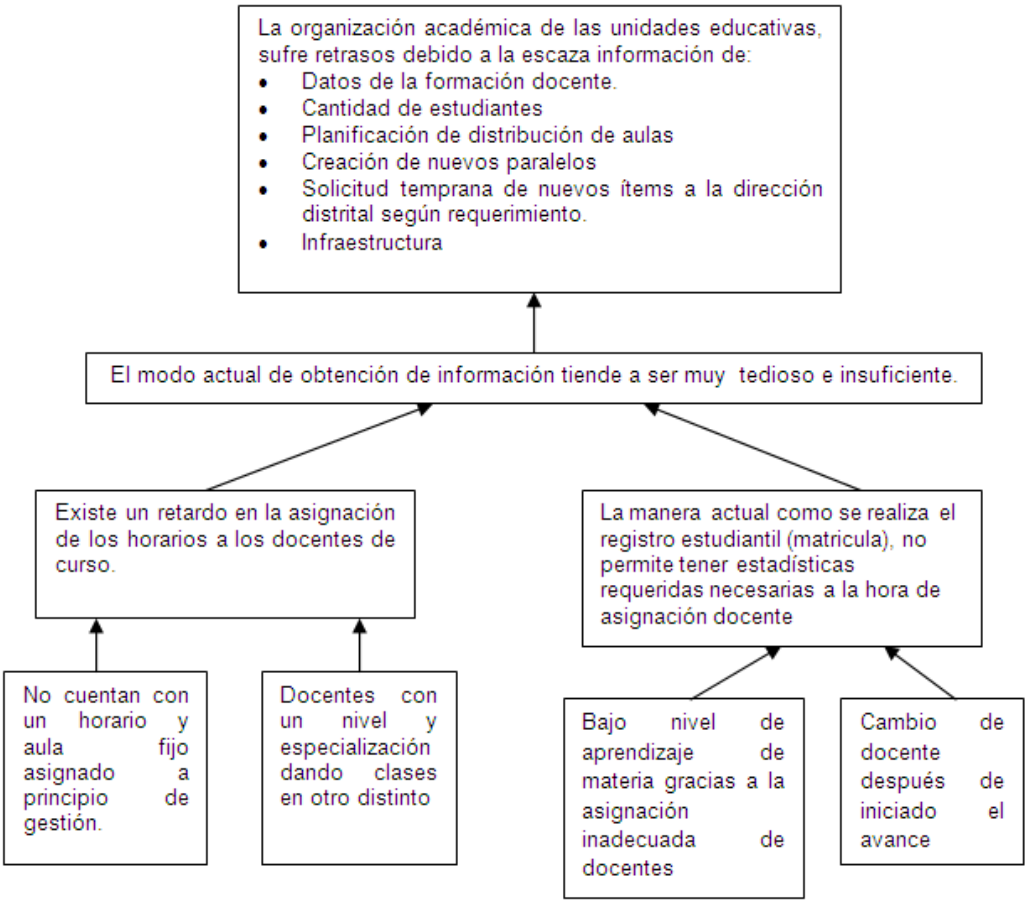


Figura 1. Árbol de Problemas

Anexo 3 Árbol de Objetivos

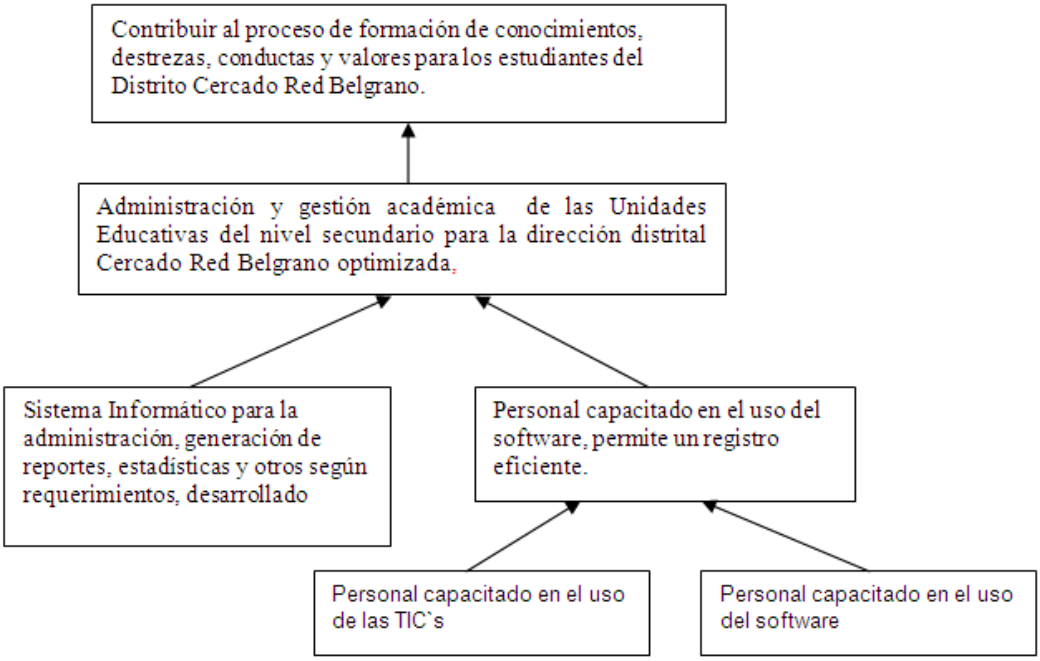


Figura 2. Árbol de Objetivos

II. COMPONENTES

II.1 ADMINISTRACION Y GESTION ACADEMICA DE LAS UNIDADES EDUCATIVAS DEL NIVEL SECUNDARIO, PARA LA DIRECCION DISTRITAL CERCADO.

II.1.1 Plan de Desarrollo del Software

II.1.1.1 Introducción

Este Plan de Desarrollo de Software es una versión preparada para ser incluida en la propuesta elaborada como respuesta al proyecto de la asignatura de Taller III de la Carrera de Ingeniería Informática de la Facultad de Ciencias y Tecnología de la Universidad Autónoma Juan Misael Saracho. Este documento provee la visión global del plan de desarrollo del software propuesto para la Red Belgrano de la dirección Distrital de Cercado Tarija.

El proyecto ha sido ofertado por Quispe Ojeda Laura Elisa basado en una metodología de Rational Unified Process en la que únicamente se procederá a cumplir con las tres primeras fases que marca la metodología, constando únicamente en la tercera fase de dos iteraciones. Es importante destacar esto puesto que utilizaremos la terminología RUP en este documento. Se incluirá el detalle para las fases de Inicio y Elaboración y adicionalmente se esbozarán las fases posteriores de Construcción y Transición para dar una visión global de todo proceso. -

El enfoque desarrollo propuesto constituye una configuración del proceso RUP de acuerdo a las características del proyecto, seleccionando los roles de los participantes, las actividades a realizar y los artefactos (entregables) que serán generados. Este documento es a su vez uno de los artefactos de RUP.

II.1.1.1.1 Propósito

El propósito del Plan de Desarrollo de Software es proporcionar la información necesaria para controlar el proyecto. En él se describe el enfoque de desarrollo del software.

Los usuarios del Plan de Desarrollo del Software son:

- El jefe del proyecto lo utiliza para organizar la agenda y necesidades de recursos, y para realizar su seguimiento.
- Los miembros del equipo de desarrollo lo usan para entender lo qué deben hacer, cuándo deben hacerlo y qué otras actividades dependen de ello.

II.1.1.1.2 Alcance

El Plan de Desarrollo de Software analiza y describe la Optimización de la organización académica y administrativa de las Unidades Educativas del nivel secundario del Distrito Cercado, utilizando las tres primeras fases de la metodología RUP. Las fases se beneficiaran con un cronograma de cada una de las actividades a realizar. Especificando los detalles de construcción del proyecto. Este Plan contiene los objetivos, alcances y propósito del proyecto, además de las responsabilidades, suposiciones y restricciones, etc.

II.1.1.1.3 Resumen

Después de esta introducción, el resto del documento está organizado en las siguientes secciones:

Vista General del Proyecto — proporciona una descripción del propósito, alcance y objetivos del proyecto, estableciendo los artefactos que serán producidos y utilizados durante el proyecto.

Organización del Proyecto — describe la estructura organizacional del equipo de desarrollo.

Gestión del Proceso — explica los costos y planificación estimada, define las fases e hitos del proyecto y describe cómo se realizará su seguimiento.

Planes y Guías de aplicación — proporciona una vista global del proceso de desarrollo de software, incluyendo métodos, herramientas y técnicas que serán utilizadas.

II.1.1.2 Vista General del Proyecto

II.1.1.2.1 Propósito, Alcance y Objetivos

La información que a continuación se incluye ha sido extraída de las diferentes reuniones que se han realizado con los usuarios finales del producto desde el inicio del proyecto.

II.1.1.2.1.1 Propósito

La Dirección Distrital de la Provincia Cercado es la encargada de velar por el bienestar de las Unidades Educativas (U.E.) de la Provincia Cercado. La Red Belgrano es una de las muchas redes Educativas en las que está dividida nuestra ciudad. La Red Belgrano consta de tres Unidades Educativas (U.E.) Col. General José Manuel Belgrano Turno mañana, Col. General José Manuel Belgrano Turno Tarde y Col. General Manuel Belgrano Sema Nocturno.

El presente proyecto tiene como propósito el desarrollo de un Software a medida que ayude en ***la optimización de la Organización académica y administrativa de las Unidades Educativas del nivel secundario del distrito Cercad, Red Belgrano.***

II.1.1.2.1.2 Alcance

Entre los alcances del mismo tenemos los siguientes:

- ❖ Mantener una información actualizada de los estudiantes de la Unidad Educativa
- ❖ Desarrollar un entorno de fácil uso para todos los usuarios del sistema, los que podrán tener acceso a información de manera apropiada.
- ❖ Tener un software eficiente que se encargue de la administración y registro de los estudiantes desde el momento de la inscripción.
- ❖ Tener un registro de los docentes mucho más eficiente al comienzo de cada gestión, que almacenara los datos personales de los docentes, datos profesionales, y datos en general, requeridos por la Unidad Educativa. permitiendo una información general del docente-administrativo en el momento que sea requerido.

- ❖ Automatización de la generación de Kardex de estudiantes, y que además permitirá adicionar observaciones acerca de la conducta, meritos o faltas (memorándums) para historial académico de cada gestión
- ❖ Automatización de la generación de Kardex de docente-administrativo, y que además permitirá adicionar observaciones acerca la conducta, meritos o faltas (memorándums), para historial académico de cada gestión.
- ❖ Asignación de aulas y horas académicas para cada docente, por parte del director de la U.E., evitando choque de horarios mediante alertas.
- ❖ Mantener la información actualizada acerca del nivel y especialidad de los docentes, a la hora de la asignación de los horarios.
- ❖ Permitir a los docentes tutores de curso tener la información acerca de los estudiantes del curso del que están a cargo, en el momento que se requiera, e ingresar observaciones de ser necesario.
- ❖ Los directores podrán tener información precisa de la cantidad de estudiantes, permitiendo una toma de decisiones oportuna, en caso de creación de nuevos paralelos, ampliación de ambientes, solicitud de nuevos docentes, y otros.
- ❖ Automatización de la generación de reportes por sexo
- ❖ Automatización de la generación de reportes por edad
- ❖ Automatización de la generación de reportes por curso
- ❖ Automatización de la generación de reportes por paralelo
- ❖ El sistema será de libre acceso pero el ingreso a los módulos administrativos solo será accedido bajo contraseñas asignadas a la secretaria, tutores y director de la Unidad Educativa.

II.1.1.2.1.3 Objetivos

II.1.1.2.1.3.1 Objetivos Generales

El objetivo Principal del presente producto es la Optimización de la Organización académica y administrativa de las Unidades Educativas del nivel secundario del distrito Cercado Red Belgrano.

II.1.1.2.1.3.2 Objetivos Específicos

- ❖ Sistema informático para la generación de reportes, estadísticas, mejor administración y otros según requerimientos, desarrollado.
- ❖ **Personal capacitado en el uso del software, permite un registro eficiente.**

II.1.1.2.2 Suposiciones y Restricciones

II.1.1.2.2.1 Suposiciones

- Las autoridades de la Unidad Educativa manifiesta apoyo y disposición para la ejecución del proyecto.
- Los equipos de computación de cada Unidad Educativa de la Red Belgrano, cuentan con el hardware necesario para el software presentado
- Predisposición del personal administrativo y autoridades educativas de la Red Belgrano para brindar información requerida para el desarrollo del sistema.
- Reportes generados, son de satisfacción de las autoridades educativas
- Asistencia masiva al curso de capacitación por parte de las autoridades educativas (director de U.E.) y personal administrativo.
- La capacitación se lleva a cabo en la fecha establecida.
- El personal administrativo y autoridades educativas hacen uso del sistema sin ninguna dificultad.

II.1.1.2.2.2 Restricciones

- ✓ No se controlara la asistencia del personal docente-administrativo.
- ✓ No se controlara la asistencia de los estudiantes.
- ✓ No se generara reporte alguno después de los acordados con el director de la Unidad Educativa en las reuniones previas a la presentación de este documento.
- ✓ El sistema no controlara notas de los estudiantes, está dirigido únicamente a la organización académica de la Unidad Educativa

II.1.1.2.3 Entregables del proyecto

A continuación se indican y describen cada uno de los artefactos que serán generados y utilizados por el proyecto y que constituyen los entregables. Esta lista constituye la configuración de RUP desde la perspectiva de artefactos, y que proponemos para este proyecto.

Es preciso destacar que de acuerdo a la filosofía de RUP (y de todo proceso iterativo e incremental), todos los artefactos son objeto de modificaciones a lo largo del proceso de desarrollo, con lo cual, sólo al término del proceso podríamos tener una versión definitiva y completa de cada uno de ellos. Sin embargo, el resultado de cada iteración y los hitos del proyecto están enfocados a conseguir un cierto grado de completitud y estabilidad de los artefactos. Esto será indicado más adelante cuando se presenten los objetivos de cada iteración.

1) Plan de Desarrollo del Software

Es el presente documento.

2) Modelo de Casos de Uso del Negocio

Es un modelo de las funciones de negocio vistas desde la perspectiva de los actores externos (Agentes de registro, solicitantes finales, otros sistemas etc.). permite situar al sistema en el contexto organizacional haciendo énfasis en los objetivos en este ámbito. Este modelo se representa con un Diagrama de Casos de Uso usando estereotipos específicos para este modelo.

3) Modelo de Objetos del Negocio

Es un modelo que describe la realización de cada caso de uso del negocio, estableciendo los actores internos, la información que en términos generales manipulan y los flujos de trabajo (workflows) asociados al caso de uso del negocio. Para la representación de este modelo se utilizan Diagramas de Colaboración (para mostrar actores externos, internos y las entidades (información) que manipulan, un Diagrama de Clases para mostrar gráficamente las entidades del sistema y sus relaciones, y Diagramas de Actividad para mostrar los flujos de trabajo.

4) Glosario

Es un documento que define los principales términos usados en el proyecto. Permite establecer una terminología consensuada.

5) Modelo de Casos de Uso

El modelo de Casos de Uso presenta las funciones del sistema y los actores que hacen uso de ellas. Se representa mediante Diagramas de Casos de Uso.

6) Visión

Este documento define la visión del producto desde la perspectiva del cliente, especificando las necesidades y características del producto. Constituye una base de acuerdo en cuanto a los requisitos del sistema.

7) Especificaciones de Casos de Uso

Para los casos de uso que lo requieran (cuya funcionalidad no sea evidente o que no baste con una simple descripción narrativa) se realiza una descripción detallada utilizando una plantilla de documento, donde se incluyen: precondiciones, post-condiciones, flujo de eventos, requisitos no-funcionales asociados. También, para casos de uso cuyo flujo de eventos sea complejo podrá adjuntarse una representación gráfica mediante un Diagrama de Actividad.

8) Especificaciones Adicionales

Este documento capturará todos los requisitos que no han sido incluidos como parte de los casos de uso y se refieren requisitos no-funcionales globales. Dichos requisitos incluyen: requisitos legales o normas, aplicación de estándares, requisitos de calidad del producto, tales como: confiabilidad, desempeño, etc., u otros requisitos de ambiente, tales como: sistema operativo, requisitos de compatibilidad, etc.

9) Prototipos de Interfaces de Usuario

Se trata de prototipos que permiten al usuario hacerse una idea más o menos precisa de las interfaces que proveerá el sistema y así, conseguir retroalimentación de su parte respecto a los requisitos del sistema. Estos prototipos se realizarán como:

dibujos a mano en papel, dibujos con alguna herramienta gráfica o prototipos ejecutables interactivos, siguiendo ese orden de acuerdo al avance del proyecto. Sólo los de este último tipo serán entregados al final de la fase de Elaboración, los otros serán desechados. Asimismo, este artefacto, será desechado en la fase de Construcción en la medida que el resultado de las iteraciones vayan desarrollando el producto final.

10) Modelo de Análisis y Diseño

Este modelo establece la realización de los casos de uso en clases y pasando desde una representación en términos de análisis (sin incluir aspectos de implementación) hacia una de diseño (incluyendo una orientación hacia el entorno de implementación), de acuerdo al avance del proyecto.

11) Modelo de Datos

Previendo que la persistencia de la información del sistema será soportada por una base de datos relacional, este modelo describe la representación lógica de los datos persistentes, de acuerdo con el enfoque para modelado relacional de datos. Para expresar este modelo se utiliza un Diagrama de Clases (donde se utiliza un profile UML para Modelado de Datos, para conseguir la representación de tablas, claves, etc.).

12) Modelo de Implementación

Este modelo es una colección de componentes y los subsistemas que los contienen. Estos componentes incluyen: ficheros ejecutables, ficheros de código fuente, y todo otro tipo de ficheros necesarios para la implantación y despliegue del sistema. (Este modelo es sólo una versión preliminar al final de la fase de Elaboración, posteriormente tiene bastante refinamiento).

13) Modelo de Despliegue

Este modelo muestra el despliegue la configuración de tipos de nodos del sistema, en los cuales se hará el despliegue de los componentes.

14) Casos de Prueba

Cada prueba es especificada mediante un documento que establece las condiciones de ejecución, las entradas de la prueba, y los resultados esperados. Estos casos de prueba son aplicados como pruebas de regresión en cada iteración. Cada caso de prueba llevará asociado un procedimiento de prueba con las instrucciones para realizar la prueba, y dependiendo del tipo de prueba dicho procedimiento podrá ser automatizable mediante un script de prueba.

15) Solicitud de Cambio

Los cambios propuestos para los artefactos se formalizan mediante este documento. Mediante este documento se hace un seguimiento de los defectos detectados, solicitud de mejoras o cambios en los requisitos del producto. Así se provee un registro de decisiones de cambios, de su evaluación e impacto, y se asegura que éstos sean conocidos por el equipo de desarrollo. Los cambios se establecen respecto de la última baseline (el estado del conjunto de los artefactos en un momento determinado del proyecto) establecida. En nuestro caso al final de cada iteración se establecerá una baseline.

16) Plan de Iteracion

Es un conjunto de actividades y tareas ordenadas temporalmente, con recursos asignados. Se realiza para cada iteración, y para todas las fases.

17) Evaluación de Iteración

Este documento incluye le evaluación de los resultados de cada iteración, el grado en el cual se han conseguido los objetivos de la iteración, las lecciones aprendidas y los cambios a ser realizados.

18) Lista de Riesgos

Este documento incluye una lista de los riesgos conocidos y vigentes en el proyecto, ordenados en orden decreciente de importancia y con acciones específicas de contingencia o para su mitigación.

19) Manual de Instalación

Este documento incluye las instrucciones para realizar la instalación del producto.

20) Material de Apoyo al Usuario Final

Corresponde a un conjunto de documentos y facilidades de uso del sistema, incluyendo: Guías del Usuario, Guías de Operación, Guías de Mantenimiento y Sistema de Ayuda en Línea

21) Producto

Los ficheros del producto empaquetados y almacenadas en un CD con los mecanismos apropiados para facilitar su instalación. El producto, a partir de la primera iteración de la fase de Construcción es desarrollado incremental e iterativamente, obteniéndose una nueva release al final de cada iteración.

Los artefactos 19, 20 y 21 se generarán a partir de la fase de Construcción, con lo cual se han incluido aquí sólo para dar una visión global de todos los artefactos que se generarán en el proceso de desarrollo.

II.1.1.2.4 Evolución del Plan de Desarrollo del Software

El Plan de Desarrollo del Software se revisará semanalmente y se refinará antes del comienzo de cada iteración.

II.1.1.3 Organización del Proyecto

II.1.1.3.1 Participantes en el Proyecto

Director de Proyecto.

Quispe Ojeda Laura Elisa

Por ser la única integrante del presente proyecto debo de estar al pendiente de todas las tareas, mantener responsabilidad para mantener el orden y coordinación durante el desarrollo del proyecto.

Recabar la información necesaria, analizarla, identificar los problemas y proponer las soluciones.

Facilidad en el manejo de programas y dominio en el área de la programación.

Tener conocimiento metodológico en el desarrollo del software, herramientas de programación.

II.1.1.3.2 Interfaces Externas

Contara con una interfaz principal que describirá la visión y misión de la Unidad Educativa y distrito Cercado, información de interés público.

Esta interfaz contendrá enlaces tanto para propietarios como para personal administrativo.

En la interfaz el director podrá manejar todo el sistema visualizar reportes, es decir una administración general del sistema.

En la interfaz del personal administrativo se podrá adicionar, modificar, dar de baja, dar de alta y otros.

En la interfaz se podrá obtener información es decir la generación de reportes.

II.1.1.3.3 Roles y Responsabilidades

A continuación se describen las principales responsabilidades de cada uno de los puestos en el equipo de desarrollo durante las fases de Inicio y Elaboración, de acuerdo con los roles que desempeñan en RUP.

Puesto	**Responsabilidad**
Director de Proyecto	El jefe de proyecto asigna los recursos, gestiona las prioridades, coordina las interacciones con los clientes y usuarios, y mantiene al equipo del proyecto enfocado en los objetivos. El jefe de proyecto también establece un conjunto de prácticas que aseguran la integridad y calidad de los artefactos del proyecto. Además, el jefe de proyecto se encargará

	de supervisar el establecimiento de la arquitectura del sistema. Gestión de riesgos. Planificación y control del proyecto. Captura, especificación y validación de requisitos, interactuando con el cliente y los usuarios mediante entrevistas. Elaboración del Modelo de Análisis y Diseño. Colaboración en la elaboración de las pruebas funcionales y el modelo de datos. Construcción de prototipos. Colaboración en la elaboración de las pruebas funcionales, modelo de datos y en las validaciones con el usuario Gestión de requisitos, gestión de configuración y cambios, elaboración del modelo de datos, preparación de las pruebas funcionales, elaboración de la documentación. Elaborar modelos de implementación y despliegue.

Tabla 5. Tabla de roles y responsabilidades

II.1.1.4 Gestión del Proceso

II.1.1.4.1 Estimaciones del Proyecto

El presupuesto del proyecto y los recursos involucrados se encuentran detallados en el perfil del proyecto.

II.1.1.4.2 Plan del Proyecto

En esta sección se presenta la organización en fases e iteraciones y el calendario del proyecto.

II.1.1.4.2.1 Plan de las Fases

El desarrollo se llevará a cabo en base a fases con una o más iteraciones en cada una de ellas. La siguiente tabla muestra una la distribución de tiempos y el número de iteraciones de cada fase (para las fases de Construcción y Transición es sólo una aproximación muy preliminar)

Fase	**Nro. Iteraciones**	**Duración**
Fase de Inicio	3	30
Fase de Elaboración	2	55
Fase de Construcción	1	80
Fase de Transición	1	20

Tabla 6. Tabla de fases

Los hitos que marcan el final de cada fase se describen en la siguiente tabla.

Descripción	**Hito**
Fase de Inicio	En esta fase desarrollarán los requisitos del producto desde la perspectiva del usuario, los cuales serán establecidos en el artefacto Visión. Los principales casos de uso serán identificados y se hará un refinamiento del Plan de Desarrollo del Proyecto. La aceptación del cliente /usuario del artefacto Visión y el Plan de Desarrollo marcan el final

	de esta fase.
Fase de Elaboración	En esta fase se analizan los requisitos y se desarrolla un prototipo de arquitectura (incluyendo las partes más relevantes y / o críticas del sistema). Al final de esta fase, todos los casos de uso correspondientes a requisitos que serán implementados en la primera release de la fase de Construcción deben estar analizados y diseñados (en el Modelo de Análisis / Diseño). La revisión y aceptación del prototipo de la arquitectura del sistema marca el final de esta fase. En nuestro caso particular, por no incluirse las fases siguientes, la revisión y entrega de todos los artefactos hasta este punto de desarrollo también se incluye como hito. La primera iteración tendrá como objetivo la identificación y especificación de los principales casos de uso, así como su realización preliminar en el Modelo de Análisis / Diseño, también permitirá hacer una revisión general del estado de los artefactos hasta este punto y ajustar si es necesario la planificación para asegurar el cumplimiento de los objetivos. Ambas iteraciones tendrán una duración de una semana.

Tabla 7. Tabla de fin de fase

Fase de Construcción	Durante la fase de construcción se terminan de analizar y diseñar todos los casos de uso, refinando el Modelo de Análisis / Diseño. El producto se construye en base a 1 iteraciones, cada una produciendo una release a la cual

	se le aplican las pruebas y se valida con el cliente / usuario. Se comienza la elaboración de material de apoyo al usuario. Con la capacidad operacional parcial del producto que se haya considerado como crítica, lista para ser entregada a los usuarios para pruebas beta.
Fase de Transición	En esta fase se prepararán una releases para distribución, asegurando una implantación y cambio del sistema previo de manera adecuada, incluyendo el entrenamiento de los usuarios. El hito que marca el fin de esta fase incluye, la entrega de toda la documentación del proyecto con los manuales de instalación y todo el material de apoyo al usuario, la finalización del entrenamiento de los usuarios y el empaquetamiento del producto.

Tabla 8. Tabla de la construcción de fase

II.1.1.4.2.2 Calendario del Proyecto

A continuación se presenta un calendario de las principales tareas del proyecto incluyendo sólo las fases de Inicio y Elaboración. Como se ha comentado, el proceso iterativo e incremental de RUP está caracterizado por la realización en paralelo de todas las disciplinas de desarrollo a lo largo del proyecto, con lo cual la mayoría de los artefactos son generados muy tempranamente en el proyecto pero van desarrollándose en mayor o menor grado de acuerdo a la fase e iteración del proyecto. La siguiente figura ilustra este enfoque, en ella lo ensombrecido marca el énfasis de cada disciplina (workflow) en un momento determinado del desarrollo.

Para este proyecto se ha establecido el siguiente calendario. La fecha de aprobación indica cuando el artefacto en cuestión tiene un estado de completitud suficiente par

someterse a revisión y aprobación, con una probabilidad de modificación para posterior refinamiento y cambios.

II.1.1.4.3 Seguimiento y Control del Proyecto

II.1.1.4.3.1 Gestión de Requisitos

Este proceso se encarga de la identificación, asignación y seguimiento de los requisitos, conociendo las necesidades que tienen los interesados.

Los requisitos que se han empezado con el proyecto se encuentran en la etapa de análisis, posteriormente, dichos requisitos en el ciclo de vida del proyecto se irán modificando por lo que se establece el tratamiento y control de las actualizaciones y cambios a los mismos.

Debido a que todo proyecto está sujeto a de cambios, se irá actualizando incorporando funcionalidades y/o eliminar otras. Para logra esto se mantendrá un control y documentación del producto.

Reuniendo las necesidades del proyecto, e implementación, los requisitos son:

- Tamaño y complejidad del proyecto,
- Experiencia del personal del proyecto,
- Experiencia de los clientes del proyecto,
- Dominio de la aplicación,
- El propósito y uso de esta aplicación.

Tareas principales de la Gestión de Requisitos

ACTIVIDADES	**DESCRIPCIÓN**
Recolección	Se realizo la recolección y documentación de requisitos, para descubrir, y registrar una representación precisa de los requisitos del producto, identificando las aéreas de mayor importancia. Se procedió a realizar Entrevistas

	donde pretendemos obtener una información general de la Institución y así conocer las fronteras del sistema, conocer sus reglas, aumentar el entendimiento y el compromiso de los participantes.
Documentación	La información recopilada se analizó a detalle y se documentó.
Verificación	Una vez que la descripción de los requisitos ha sido desarrollada, se ha verificado.
Gestión de Cambios	Como el proyecto va evolucionando, Los requerimientos pueden cambiar o expandirse y deberán ajustarse al Proyecto. Para así lograr identificar, evaluar, trazar y reportar cambios propuestos y aprobados a la especificación del producto.

Tabla 9. Tabla de gestión de requisitos

Requisitos para el desarrollo del Sistema

1. Requisitos Organizacionales

El personal debe conocer la metodología RUP bajo un modelado con el lenguaje UML.

Se debe contar con un personal que conozca sobre la plataforma java. El modo de aplicación y trabajo. Contar con el software necesario para el desarrollo del sistema.

2. Requisitos de Personal y Usuarios

Se debe contar por lo menos con una persona informada y capacitada en el manejo del sistema.

Todo usuario debe contar con un usuario y clave.

3. Requisitos de Hardware y funcionamiento

Para el funcionamiento del sistema se debe contar por lo menos con diez ordenadores.

Una LAN (Red de Área Local) con un protocolo TCP/IP.

Los equipos deben contar con un procesador Dual Core y memoria RAM de 1Gb o superior.

Los monitores deben soportar la resolución de 1024 X 768 px.

4. Requisitos de Software y Construcción

El sistema debe tener la capacidad de administrar entre 2 a 3 equipos de computación con una concurrencia de 10 usuarios en línea en un intervalo x de tiempo.

El sistema debe utilizar tecnologías actuales.

Interactuar en una plataforma Windows.

El sistema se lo debe construir bajo una Java Virtual Machine v 6.

El sistema debe tener la capacidad de interactuar con algún motor de base de datos.

5. Requisitos de Software

Java Runtime versión 6 update 2.

Base de datos PostgreSQL 8.4 adelante.

II.1.1.4.3.1.2 Control de Plazos

ID	**FASES**	**Duración Días**	**FECHA DE ENTREGA**	**OBSERVACIONES**
ID	**Fase de Inicio**			
0	**Plan de Desarrollo**	**7**	**29/11/2010**	Se cumplió con la fecha establecida.

	de Software (borrador)			Durante el desarrollo de esta fase se ha trabajado en los artefactos: Visión, Glosario, Caso de Uso del Negocio, Modelos de Objetos del Negocio, Casos del uso del sistema, Diagramas de Actividades, Diagramas de Secuencia, Diagramas de Colaboración y el diseño de Pantallas. Durante el desarrollo de esta fase se ha trabajado en los artefactos: Visión, Glosario, Caso de Uso del Negocio, Modelos de Objetos del Negocio, Casos del uso del sistema, Diagramas de Actividades, Diagramas de Secuencia, Diagramas **de** Colaboración, Diagramas de Clases, Diseño de la Base de Datos, Pantallas UML, Mapas Navegacionales y el prototipo en un 40% . Durante el desarrollo de esta fase se ha trabajado en los artefactos: Visión, Glosario, Caso de Uso del Negocio,

				Modelos de Objetos del Negocio, Casos del uso del sistema, Diagramas de Actividades, Diagramas de Secuencia, Diagramas de Colaboración, Diagramas de Clases, Diseño de la Base de Datos, Pantallas UML, Mapas Navegacionales, el prototipo en un 80%, Casos de Prueba (Prueba de Caja Blanca y Prueba de Caja Negra) y Especificación de Métodos. Se cumplió con la fecha establecida.

Tabla 10. Tabla Control de Plazos

II.1.1.4.3.3 Control de Calidad

Para garantizar la calidad en el análisis, diseño y desarrollo del sistema el grupo de desarrollo se baso en tres principios que nos parece importante:

- Control de Variación.
- Calidad en el Diseño.
- Calidad de Concordancia.
- Uso de Herramientas CASE para detección de Errores de Código

Control de Variación: Se fueron realizando controles a las variaciones que se hacía en cada versión del sistema.

Calidad en el Diseño: Se enfatizó el desarrollo de cada uno de los elementos del sistema y se tomó en cuenta los requerimientos planteados por el cliente para evitar problemas con el mismo al momento de su entrega.

Calidad de Concordancia: A medida que se desarrollaba el software se fue concordando el sistema con los requerimientos del cliente.

Se tomo en cuenta la apariencia del sistema verificando los siguientes aspectos.

- Autonomía.-Las interfaces de usuario presentan un ambiente flexible, y de fácil utilización.
- Percepción del color, se eligio el color e imágenes de acuerdo al trabajo del cliente, para dar una apariencia acorde al desarrollo y uso en la ejecución del sistema.
- Eficiencia del usuario.- se dispuso de iconos que vayan de acuerdo a su funcion, para evitar confusiones.
- Interfaces amigables.- Las interfaces son muy amigables, presentando una navegación sencilla, donde cada pantalla presenta enlaces faciles de usar.
- Consistencia, manteniendo una concordancia con el ambiente de trabajo.
- Legibilidad.- La Información presentada es legible por el usuario.

II.1.1.4.3.4 Control de Cambios

Es de vital importancia para el grupo el control de cambios ya que una diminuta perturbación en el código puede crear un gran fallo en el desarrollo del producto, para ello se debe llevar un control de cambios seguro, en el cual el grupo o cliente debe seguir el Siguiente Proceso:

Flujo Principal	**Flujo Alterno**

Se reconoce la necesidad del cambio	
El usuario suscribe la petición de cambio	
El desarrollador la evalúa	
Se genera un informe de cambio	
La autoridad de control de cambios decide	
La petición queda pendiente de actuación se genera la OCI	**Petición de cambios denegada**
Asignación personalizada a los objetos de configuración	**Información del usuario**
"Dar de Baja" objetos de configuración	
Realización del cambio	
Revisión de cambio (Auditoria)	
Los elementos de configuración de han cambiado son "Dados de Alta"	
Establecimiento de una línea base para la prueba	
Realización de actividades de garantía de calidad y de prueba	
"Promoción " de los cambios para ser incluidos en la siguiente versión	
Reconstrucción de la versión adecuada del	

software	
Revisión (auditoria) de los cambios en todos los elementos de configuración	
Cambios incluidos en la nueva versión	
Distribución de la nueva versión	

Tabla 11. Control de Cambios

II.1.1.4.3.5 Gestión de Configuración

Se realizará una gestión de configuración para llevar un registro de las versiones y de las modificaciones que éstas produzcan, informando y publicando dichos cambios para que sean accesibles a todo los participantes en el proyecto.

Para realizar la gestión de configuración realizaremos las siguientes actividades:

1. Identificación de Elementos de Gestión de Configuración.
2. Control de Versiones.
3. Control de Cambios.
4. Auditoria de Configuración.
5. Generación de Informes.

Se identifico cada uno de los elementos para la realización de la gestión de configuración para un mayor control de los mismos

ID	PDS
Nombre	Plan de Desarrollo de Software
Versión	0.0
Tipo	Documento

Proyecto	SISED
Fecha	29/11/2010
Origen del Elemento	Primera fase del proyecto
Responsable	Jefe de Proyecto Laura Quispe
Información del Cambio	Se dio una vista preliminar del proyecto, objetivos y alcances.

Tabla 12. Gestión de configuración 1

ID	VS
Nombre	Visión
Versión	0.2
Tipo	Documento
Proyecto	SISED
Fecha	29/11/2010
Origen del Elemento	Visión y requisitos del Sistema
Responsable	Laura Quispe
Información del Cambio	Documento de visión del Proyecto

Tabla 13. Gestión de configuración 2

II.1.1.4.3.6 Referencias

Para realizar el Plan de Desarrollo de Software se utilizaron las siguientes referencias:

Referencia
Análisis y diseño de Kendall & Kendall
Ingeniería de Software un enfoque Practico
Análisis y Diseño con Rational
Lenguaje de Modelado Unificado
http://www.dsic.upv.es/asignaturas/facultad/lsi/ejemplorup/Implementacion2.html
http://www.monografias.com/trabajos14/aplicacion-distrib/aplicacion-distrib.shtml
http://www.dsic.upv.es/asignaturas/facultad/lsi/ejemplorup/
http://www.inf.udec.cl/~aimcon/grupouml/apuntecolombia/implementacion01.html

Tabla 14. Referencias

II.1.2 Visión

Este documento define la visión del producto desde la perspectiva del cliente, especificando las necesidades y características del producto. Constituye una base de acuerdo en cuanto a los requisitos del sistema.

II.1.2.1 Introducción

II.1.2.1.1 Propósito

El propósito de éste documento es recoger, analizar y definir las necesidades de alto nivel y las características para el mejoramiento de la administración y gestión académica de las unidades educativas del nivel secundario, para la dirección distrital Cercado Red Belgrano. El documento se centra en la funcionalidad requerida por los participantes en el proyecto y los usuarios finales, de modo que dicho departamento sea capaz de atender los distintos trámites, de una forma automatizada.

Los detalles de cómo el sistema cubre los requerimientos se pueden observar en la especificación de los casos de uso y otros documentos adicionales.

II.1.2.1.2 Alcance

Este documento Visión se ocupa, del sistema de administración y gestión académica de las unidades educativas de Cercado Red Belgrano, tomando como prueba piloto a la Unidad Educativa Gral. Manuel Belgrano Turno Mañana Nivel Secundario. Dedicada a la enseñanza y educación de los estudiantes del nivel medio.

El sistema permitirá a los encargados de la Institución Publica controlar todo lo relativo del seguimiento de documentación de trámites, proporcionando facilidad de información sobre el estado de los mismos.

II.1.2.1.3 Definiciones, Acrónimos, y Abreviaciones

U.E. Son las siglas de Unidad Educativa

R.U.P. Son las siglas de Rational Unified Process. Se trata de una metodología para escribir el proceso de desarrollo de software.

II.1.2.1.4 Referencias

- Glosario.
- Plan de desarrollo de software.
- RUP (Rational Unified Process).
- Diagrama de casos de uso.

I.1.2.2 Posicionamiento

I.1.2.2.1 Oportunidad de Negocio

Este sistema permitirá a la Institución Publica automatizar la administración general, el cual permitirá a nuestro usuario acceder de manera factible y simple. Mediante interfaces gráficas manejables, los datos se almacenaran en una base de datos que se mantendrá siempre actualizada, permitiendo una administración eficiente y mejor. Coayudando a la toma de decisiones a el director de la Unidad Educativa.

II.1.2.2.2. Sentencia que define el problema

El problema de	Controlar el registro e inscripción de los estudiantes de la U.E. Gestionar el estado actual del estudiante. Controlar el registro de los docentes. Gestionar la administración de los memorandus de la U.E. Gestionar la administración de materias de la U.E. Gestionar la administración de la asignación horario docente curso.

afecta a	Unidades Educativas - Estudiantes - Parte docente administrativo -Parte administrativa
El impacto asociado es	Gestionar la administración y gestión académica de las U.E. en la parte de registro y control de inscripción estudiante, registro y control administrativo y asignación de horarios para los docentes. Mantener actualizado el estado de los estudiantes registrados en la U.E. por gestión. Mantener actualizado el estado de los docentes administrativos registrados en la U.E. por gestión
una adecuada solución sería	Automatizar el proceso, usando un sistema que contenga información al momento que se la requiera. Desarrollando una base de datos accesible desde los distintos puntos de la red local (si contase con ella) Generar una interfaz amigable y sencillas con las que las que se accede a la base de datos.

Tabla 15. Sentencia de Problema

II.1.2.2.3. Sentencia que define la posición del Producto

Para	**Unidades Educativas en general** **-Director del establecimiento** **-Secretaria de la Unidad Educativa** **- Docentes administrativos tutores de algún curso** **Área de administración** **-Memorandus** **-Materias** **-Asignación horario** **-Reportes**
Quienes	Controlan la administración general de la Unidad Educativa, los estudiantes y administrativos, sobre todo la asignación docente que llega a ser la situación más conflictiva a inicios de gestión.
El nombre del producto	Es una herramienta software. "MAGABEL" Mejoramiento de la Administración y Gestión de las Unidades Educativas del nivel secundario distrito Cercado Red Belgrano.

Que	Permite administración y gestión de la Unidad Educativa, administrar todo tramite que se realiza en cuanto a la administración, actualización de datos de los estudiantes y docentes administrativos que se encuentren en estado activo durante la gestión. Además de almacenar información necesaria actualizada en todo momento.
No como	Se maneja actualmente, de manera manual.
Nuestro producto	Permite administrar y planificar de manera cómoda y sencilla las distintas actividades de la institución, planificación temprana de horarios y distribución de ambientes, toma de decisiones confiable, información actualizada en todo momento, permite tener datos exactos de la cantidad estudiantil y solitud ítems en caso de requerirlos. El sistema en realidad viene a alivianar la carga de todo director y personal administrativo, (funciones y servicios definidos a comienzo del proyecto.)

Tabla 16. Definición de posición del Proyecto

II.1.2.3 Descripción de Stakeholders (Participantes en el Proyecto) y Usuarios

Para proveer de una forma efectiva productos y servicios que se ajusten a las necesidades de los usuarios, es necesario identificar e involucrar a todos los participantes en el proyecto como parte del proceso de modelado de requerimientos. También es necesario identificar a los usuarios del sistema y asegurarse de que el conjunto de participantes en el proyecto los representa adecuadamente. Esta sección muestra un perfil de los participantes y de los usuarios involucrados en el proyecto, así como los problemas más importantes que éstos perciben para enfocar la solución propuesta hacia ellos. No describe sus requisitos específicos ya que éstos se capturan mediante otro artefacto. En lugar de esto proporciona la justificación de por qué estos requisitos son necesarios.

II.1.2.3.1 Resumen de Stakeholders

Nombre	descripción	responsabilidades
Director de la U.E.	Representante de la unidad Educativa frente a reuniones realizadas por el distrito educativo de Cercado . además de ser la autoridad maxima dentro de dicha Unidad Educativa	El stakeholder realiza: Representa a todos los usuarios posibles del sistema. Seguimiento del desarrollo del proyecto. Aprueba requisitos y funcionalidades
Coordinador D.D.U.	Encargado de coordinar todas las actividades de la U.E. de manera interna realizar control docente y	El stakeholder realiza: Coordinar con los docentes de curso sobre

	otros.	algunas reuniones internas, informar al director cualquier anomalía en la U.E., representante directa del director en causa de ausencia.
		Coordinar reuniones con el director para presentación de sistema, cursos de capacitación y facilidad de contacto con los docentes.

Tabla 17. Stakeholder

II.1.2.3.2 Resumen de Usuarios (Tomando en cuenta la Unidad Educativa Gral. José Manuel Belgrano Turno Manaña. Como prueba piloto)

Nombre	Descripcion	Stakeholder
Director de U.E.	Responsable de controlar la administración y coordinación de la U.E. aprobar o rechazar nuevos ingresos solicitar nuevos ítems, solicitud para nueva ampliación de ambientes, asignación de	Prof. Antonio Paco

	horarios docente curso. Y otros de administración en la gestión académica y toma de decisiones.	
Secretaria del establecimiento.	Responsable de controlar al plantel docente, docentes y docentes tutores de cada curso, también encargada de realizar toda la correspondencia de la U.E.	
Secretaria del establecimiento.	Responsable de controlar al plantel docente, docentes y docentes tutores de cada curso, también encargada de realizar toda la correspondencia de la U.E.	
Docente Tutor	Profesor encargado de un curso en diferentes actividades, representante y organizador del mismo	Prof. Martha Sanchez

	ante situaciones mayores	

Tabla 18. Descripción de Usuarios

II.1.2.3.3 Entorno de usuario

Los usuarios entrarán al sistema identificándose sobre un ordenador con un sistema operativo Windows Vista y tras este paso entrarán introduciendo login y password a la parte de aplicación diseñada para cada uno según su papel en la Intitucion (U.E.) Este sistema es similar a cualquier aplicación Windows y por tanto los usuarios estarán familiarizados con su entorno.

Los informes serán generados mediante herramientas especializadas.

II.1.2.3.4 Perfil de los Stakeholders

1 Director De Unidad Educativa.

Representante	Prof. Antonio Paco
Descripción	Director de la U.E.
Tipo	Profesor
Responsabilidades	Encargado de la administración y coordinación de la Unidad Educativa, realizar tareas de coordinación general al comienzo de cada gestión y durante el año según sea requerido.
Criterio de Éxito	A definir por el cliente
Grado de participación	Tienes privilegios para todos los módulos del sistema
Comentarios	Ninguno

Tabla 19. Stakeholder de Director

2 Secretaria de la Unidad Educativa

Representante	Falta nombre
Descripción	Secretaria encargada de la coordinación en la Unidad Educativa
Tipo	Secretaria
Responsabilidades	Encargado de coordinar las actividades internas de la Unidad Educativa, coordinar con docentes y cursos.
Criterio de Éxito	A definir por el cliente
Grado de participación	A definir por el cliente
Comentarios	Ninguno

Tabla 20. Stakeholder de secretaria

3 Docente tutor

Representante	Prof. Martha Sánchez
Descripción	Docente tutor encargado de un curso en la U. E.
Tipo	Profesor
Responsabilidades	Encargado de coordinar las actividades de los cursos con los estudiantes.
Criterio de Éxito	A definir por el cliente
Grado de participación	A definir por el cliente

Comentarios	Ninguno

Tabla 21. Stakeholder de Docente

II.1.2.3.5 Perfiles de usuario

1 Director de la Unidad Educativa

Representante	Prof. Antonio Paco
Descripción	Director de la U.E.
Tipo	Profesor
Responsabilidades	Encargado de la administración de la Unidad Educativa, registro de docentes, memorandus a docentes, y administrar las materias.
Criterio de Éxito	A definir por el cliente
Grado de participación	Tienes privilegios para todos los módulos del sistema
Comentarios	Ninguno

Tabla 22. Usuario Director

2 Secretaria de la Unidad Educativa

Representante	Falta nombre
Descripción	
Tipo	Secretaria
Responsabilida	Encargado de realizar inscripción, registrar estudiantes, dar

des	memorandus, generar reportes, registro de materias.
Grado de participación	A definir por el cliente

Tabla 23. Usuario Secretaria

3 Docente tutor de curso

Representante	Prof. Martha Sánchez
Descripción	Encargado de ver y dar memorandus a los estudiantes.
Tipo	Profesor
Responsabilidades	Encargado de coordinar los departamentos de la D.D.U.
Criterio de Éxito	A definir por el cliente
Grado de participación	A definir por el cliente
Comentarios	Ninguno

Tabla 24. Usuario Docente

II.1.2.4 Descripción Global del Producto

II.1.2.4.1 Perspectiva del producto

El producto a desarrollar es un sistema dirigido a las distintas Unidades Educativas del distrito Cercado. Este sistema está siendo realizado tomando en cuenta la Reb Belgrano, de la cual se tomo en cuenta para la prueba piloto la Unidad Educativa Gral. Manuel Belgrano Turno Mañana, con la intención de agilizar su funcionamiento

y tener un mejor control de las actividades que se realizan dentro. Las áreas a tratar por el sistema son: administración, control, asignación y gestión académica.

II.1.2.4.2 Resumen de características

A continuación se mostrará un listado con los beneficios que obtendrá el cliente a partir del producto:

Beneficio del cliente	**Características que lo apoyan**
Mayor agilidad al obtener las observaciones	Fácil acceso a la base de datos.
Mayor control en el estado de los tramites	Restringir los procesos no permitidos a algunos usuarios.
Mayor control en la administración de memorandus existentes dentro de la norma establecida en el distrito Cercado.	El sistema no permitirá irregularidades en la administración de los memorandus.
Generar advertencias en el momento de la asignación de horarios, docente curso	Generar advertencias.
Mayor facilidad en la generación y recepción de reportes.	Obteniéndolas mediante herramientas especializadas.

Tabla 25. Resumen de caracteristicas

Modulo de Reg_estudiante

Con interfaces sencillas y amigables de registro de estudiantes donde se realiza la preinscripción de los estudiantes, registrando sus datos e información necesaria para la Unidad Educativa, permitiendo

-Adicionar estudiante

-Modificar datos de estudiante.

-Dar de baja.

-Dar de alta

-Kardex estudiante.

Modulo Inscripción

Modulo en el que se realiza la inscripción del estudiante previamente registrado, permitiendo

- Inscribir al estudiante
- Eliminar la inscripción

Modulo reg_administrativo

En este modulo el Usuario se encarga de registrar los datos personales del docente antes de ser registrados en el registro de la Unidad Educativa.

Modulo registro

Modulo en el que registra los datos profesionales del docente, previamente registrados antes de ser registrados en el registro de la Unidad Educativa, su profesión y rol a desempeñar en dicha Unidad.

Modulo de reg_materias

Modulo en que administra las materias que se van a dictar en todos los cursos del nivel secundario de la Unidad Educativa

Modulo administrar_horario

Modulo en el que se administra la asignación de los horarios a los docentes en los distintos cursos permite realizar modificaciones en caso de error o modificaciones de horario

Modulo de Reportes

En este modulo el usuario genera reportes requeridos previa verificación de su clave

Restricciones

- *Poco uso del Teclado.*
- Poco uso de memoria.

II.1.3 GLOSARIO

II.1.3.1. Introducción

Este documento recoge todos y cada uno de los términos manejados a lo largo de todo el Proyecto MAGABEL Mejoramiento de la Administración y Gestión Académica de las unidades educativas del nivel secundario, para la dirección distrital Cercado Red Belgrano. Se trata de un diccionario informal de datos y definiciones de la nomenclatura que se maneja, de tal modo que se crea un estándar para todo el proyecto.

II.1.3.1.1 Propósito

El propósito de este glosario es definir con exactitud y sin ambigüedad la terminología manejada en el proyecto de desarrollo, también sirve como guía de consulta para la clarificación de las palabras confusas.

II.1.3.1.2 Alcance

El alcance del presente documento se extiende a todos los componentes del proyecto. De tal manera que la terminología empleada en MAGABEL, se refleje con claridad en este documento.

II.1.3.1.3 Referencias

El presente Glosario hace referencia a los siguientes documentos:

- Perfil del proyecto

- Documento Plan de Desarrollo de Software

- Documento de Caso de Uso de Negocio

- Documento Modelos de Objetos del Negocio

- Documento Visión

- Documento Modelo de Casos de Uso

- Documento Especificación de los Casos de Uso

- Documento Modelo de Análisis y Diseño

- Documento Modelo de Datos

- Documento Prototipos de Interfaces de Usuario

II.1.3.1.4 Organización del Glosario

El presente documento está organizado por definiciones de términos ordenados de forma ascendente según la ordenación alfabética tradicional de español.

II.1.3.2 Definiciones

A continuación se presentan todos los términos manejados a lo largo del desarrollo del proyecto MAGABEL.

- **Base de datos** Se encarga de almacenar toda la información requerida de la unidad Educativa.
- **Java (Java)** Lenguaje de programación desarrollado por la empresa Sun para la elaboración de pequeñas aplicaciones exportables a la red (*applets*) y capaces de operar sobre cualquier plataforma a través, normalmente, de navegadores WWW. Permite dar dinamismo a las páginas web y aplicaciones de Escritorio.
- **Núcleo** Conjunto de Redes en que están organizadas las unidades educativas**,** conformada por director de núcleo, quien tiene jurisdicción y competencia sobre el conjunto de unidades educativas que conforman un núcleo educativo**Red** Conformadas por unidades educativas, cada red está compuesta por el director, docentes, padres de familia, juntas escolares y alumnos
- **Prueba Piloto** Prueba del sistema por un periodo de tiempo, para monitorear el desenvolvimiento de los distintos actores, para detectar y corregir errores antes de implementarlo.

- **RUP** Son las siglas de Rational Unified Process. Se trata de una metodología para describir el proceso de desarrollo de software, basada en cuatro fases y cada una de ellas en iteraciones.
- **Sistema Informático** Conjunto de partes (hardware y software) que funcionan relacionándose entre sí con un objetivo preciso. Los usuarios son parte del sistema informático.
- **Sistema Operativo** Un sistema operativo (SO) es un conjunto de programas o software destinado a permitir la comunicación del usuario con un ordenador y gestionar sus recursos de manera cómoda y eficiente. Comienza a trabajar cuando se enciende el ordenador, y gestiona el hardware de la máquina desde los niveles más básicos .Ejemplos Windows, Linux, MacOS, Solaris.
- **TIC** Siglas que significan Tecnologías de información y Comunicación. Las TIC son medios que aportan un flujo ininterrumpido de información.
- **UML** Son las siglas de Lenguaje Unificado de Modelación. Es una lenguaje grafico para visualizar, especificar construir y documentar un sistema de software

II.1.4 MODELO DE CASO DE USO DEL NEGOCIO

II.1.4.1 Introducción

El modelo de Casos de Uso del Negocio. Es un artefacto de la disciplina Requisitos en la metodología RUP la cual se está implementando.

II.1.4.1.1 Propósito

- Comprender la estructura y la dinámica del establecimiento educativo
- Comprender los problemas actuales e identificar posibles mejoras.
- Comprender los procesos del negocio del establecimiento educativo

II.1.4.1.2 Alcance

- Describir los procesos de negocio y los usuarios
- Identificar y definir los procesos del negocio según los objetivos del establecimiento.
- Definir un caso de uso del negocio para cada proceso del negocio (diagrama de casos de uso del negocio puede mostrar el contexto y los límites del establecimiento.)

II.1.4.2 Diagrama de Casos de Uso del Negocio

II.1.4.2 .1 DIRECTOR

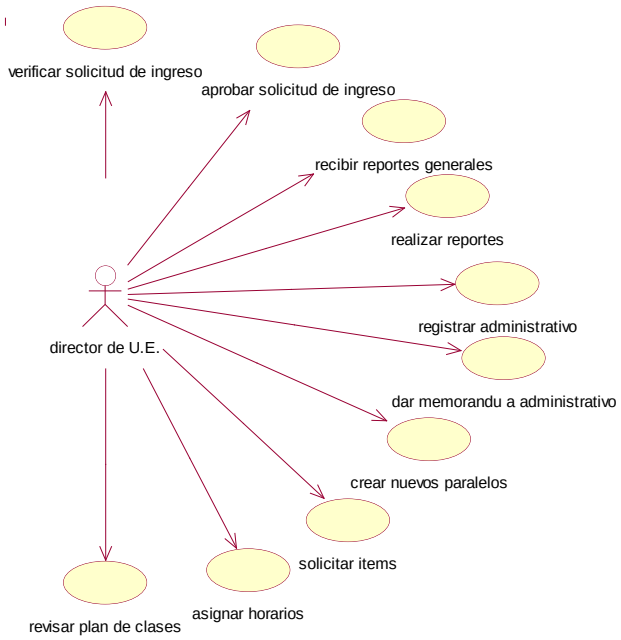


Figura 3 D.C.U. director

II.1.4.2 .2 SECRETARIA

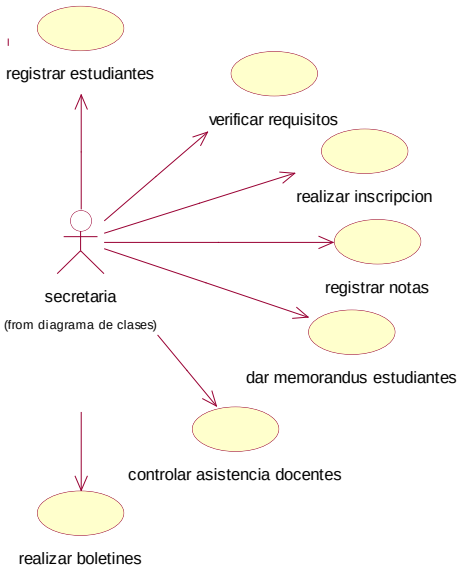


Figura 4. D.C.U. secretaria

II.1.4.2 .3 DOCENTE

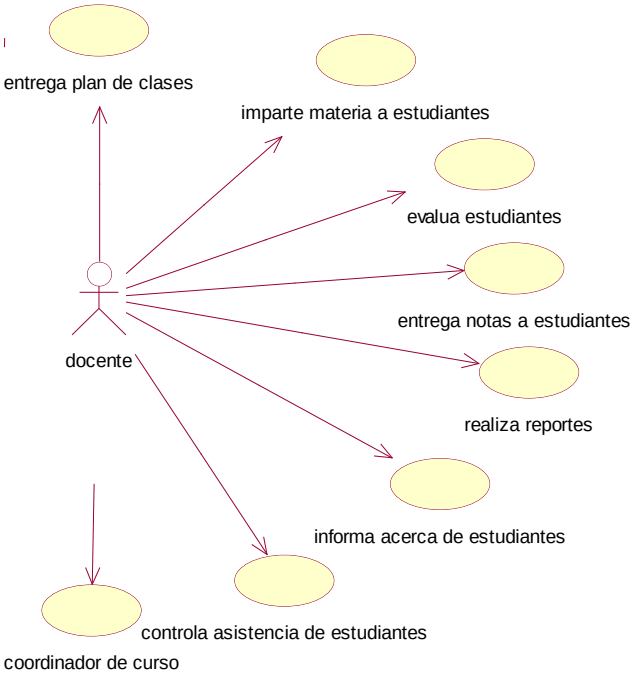


Figura 5. D.C.U. docente

II.1.4.2 .4 PADRE DE FAMILIA

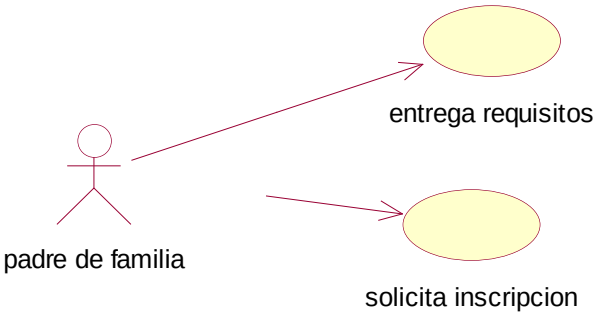


Figura 6. D.C.U. Padre de Familia

II.1.5 MODELO DE OBJETO DEL NEGOCIO

II.1.5.1 Introducción

El Modelo de Objeto del Negocio es un artefacto de la disciplina Requisitos en la metodología RUP que se está implementando.

II.1.5.1.1 Propósito

- Comprender la estructura dinámica de los casos de uso del negocio de MAGABEL
- Comprender los procesos de negocio de la organización

II.1.5.1.2 Alcance

- Describe el comportamiento de los procesos de negocio
- Identificar y definir los objetos del negocio

II.1.5.2 Diagrama de Objetos del Negocio

II.1.5.2.1 DIRECTOR

II.1.5.2.1.1 MODELO DE OBJETO VERIFICAR SOLICITUD INGRESO

Figura 7. M.O.N. VERIFICAR SOLICITUD INGRESO

II.1.5.2.1.2 MODELO DE OBJETO APROBAR SOLICITUD DE INGRESO

Figura 8. M.O.N. APROBAR SOLICITUD DE INGRESO

II.1.5.2.1.3 MODELO DE OBJETO RECIBIR REPORTES GENERALES

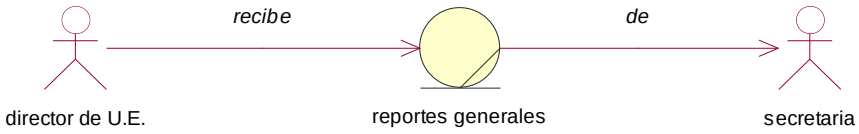


Figura 9. M.O.N. RECIBIR REPORTES GENERALES

II.1.5.2.1.4 MODELO DE OBJETO REALIZAR REPORTES

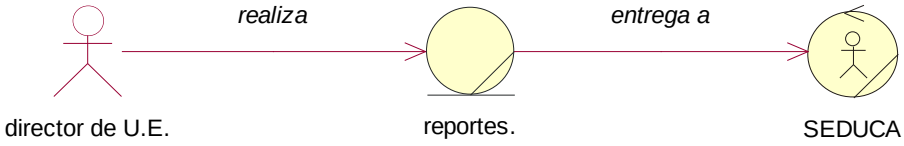


Figura 10. M.O.N. REALIZAR REPORTES

II.1.5.2.1.5 MODELO DE OBJETO REGISTRAR ADMINISTRATIVO

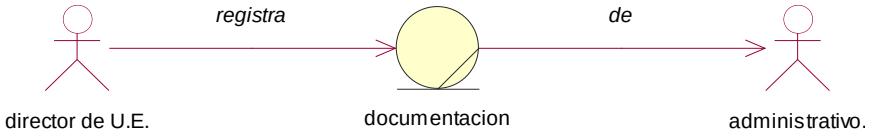


Figura 11. M.O.N. REGISTRAR ADMINISTRATIVO

II.1.5.2.1.6 MODELO DE OBJETO DAR MEMORANDU A ADMINISTRATIVO

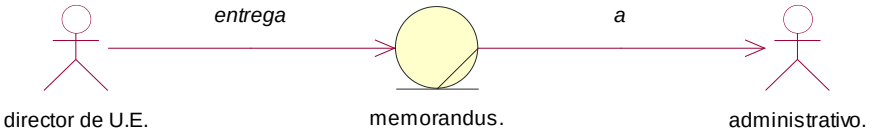


Figura 12. M.O.N. DAR MEMORANDU A ADMINISTRATIVO

II.1.5.2.1.7 MODELO DE OBJETO CREAR PARALELOS

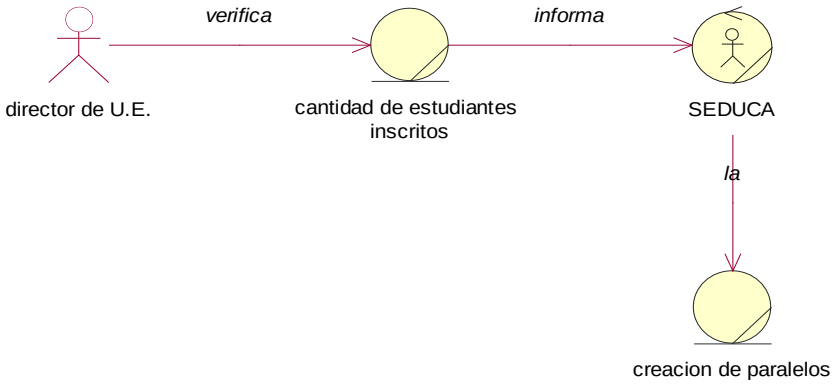


Figura 13. M.O.N. CREAR PARALELOS

II.1.5.2.1.8 MODELO DE OBJETO SOLICITAR ITEMS

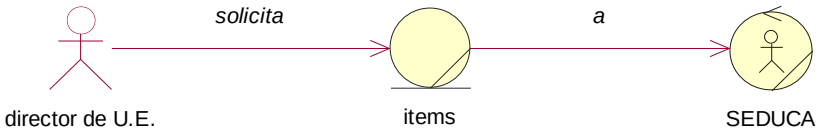


Figura 14. M.O.N. SOLICITAR ITEMS

II.1.5.2.1.9 MODELO DE OBJETO ASIGNAR HORARIOS

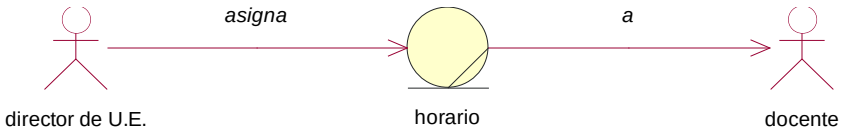


Figura 15. M.O.N. ASIGNAR HORARIOS

II.1.5.2.1.10 MODELO DE OBJETO REVISAR PLAN DE CLASES

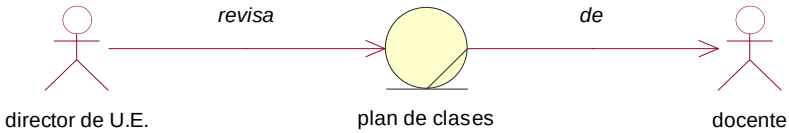


Figura 16. M.O.N. REVISAR PLAN DE CLASES

II.1.5.2.2 SECRETARIA

II.1.5.2.2.1 MODELO DE OBJETO REGISTRAR ESTUDIANTES

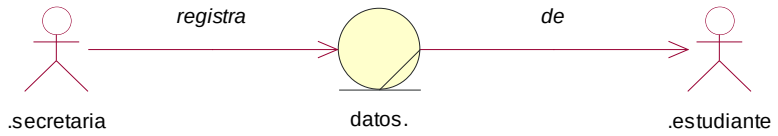


Figura 17. M.O.N. REGISTRAR ESTUDIANTES

II.1.5.2.2.2 MODELO DE OBJETO VERIFICAR REQUISITOS

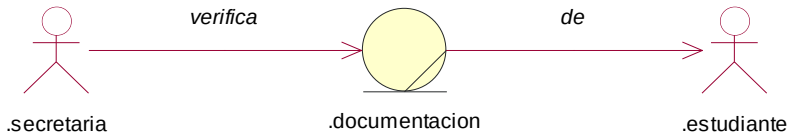


Figura 18. M.O.N. VERIFICAR REQUISITOS

II.1.5.2.2.3 MODELO DE OBJETO REGISTRAR INSCRIPCION

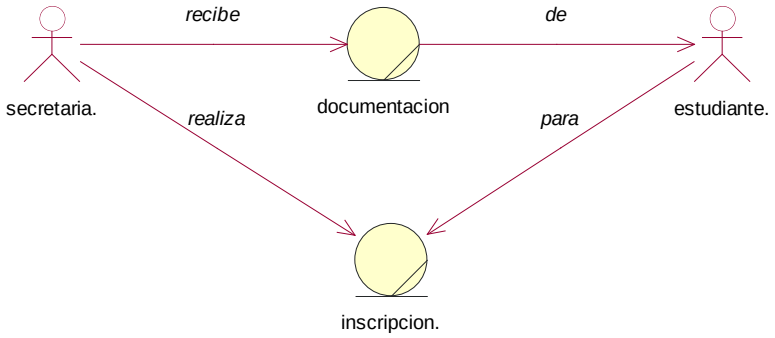


Figura 19. M.O.N. REGISTRAR INSCRIPCION

II.1.5.2.2.4 MODELO DE OBJETO REGISTRAR NOTAS

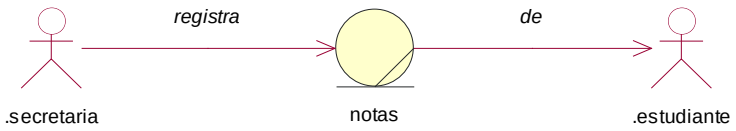


Figura 20. M.O.N. REGISTRAR NOTAS

II.1.5.2.2.5 MODELO DE OBJETO DAR MEMORANDUS ESTUDIANTES

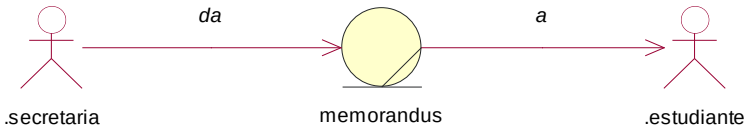


Figura 21. M.O.N. DAR MEMORANDUS ESTUDIANTES

II.1.5.2.2.6 MODELO DE OBJETO CONTROLAR ASISTENCIA DOCENTES

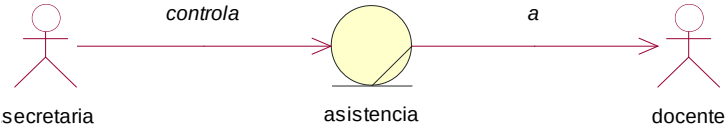


Figura 22. M.O.N. CONTROLAR ASISTENCIA DOCENTES

II.1.5.2.2.7 MODELO DE OBJETO REALIZAR BOLETINES

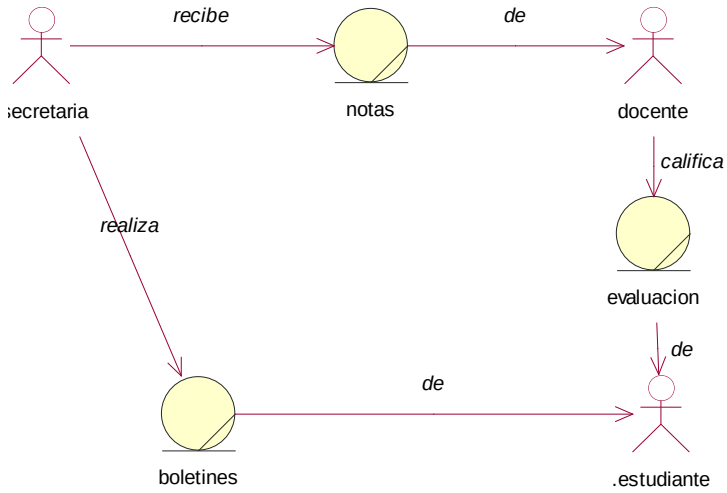


Figura 23. M.O.N. REALIZAR BOLETINES

II.1.5.2.3 DOCENTE

II.1.5.2.3 .1 MODELO DE OBJETO ENTREGA PLAN DE CLASES

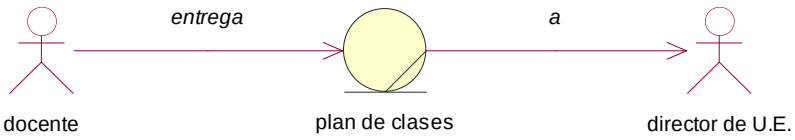


Figura 24. M.O.N. ENTREGA PLAN DE CLASES

II.1.5.2.3.2 MODELO DE OBJETO IMPARTE MATERIA A ESTUDIANTES

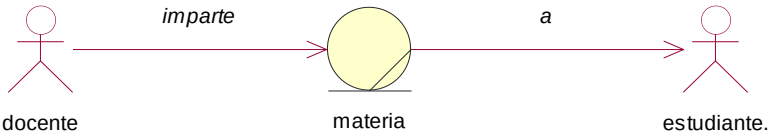


Figura 25. M.O.N. IMPARTE MATERIA A ESTUDIANTES

II.1.5.2.3 .3 MODELO DE OBJETO EVALUA A ESTUDIANTES

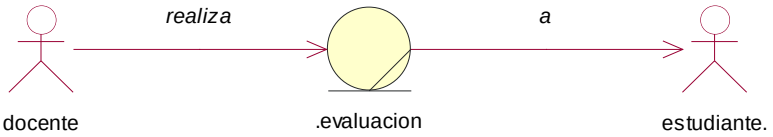


Figura 26. M.O.N. EVALUA A ESTUDIANTES

II.1.5.2.3 .4 MODELO DE OBJETO ENTREGA NOTAS A ESTUDIANTES

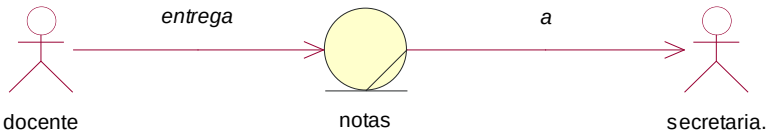


Figura 27. M.O.N. ENTREGA NOTAS A ESTUDIANTES

II.1.5.2.3.5 MODELO DE OBJETO REALIZA REPORTES

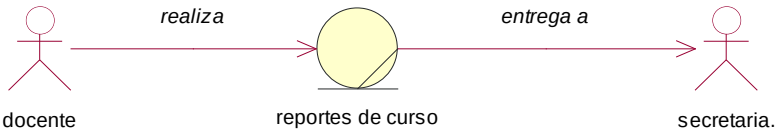


Figura 28. M.O.N. REALIZA REPORTES

II.1.5.2.3.6 MODELO DE OBJETO INFORMA RENDIMIENTO DE ESTUDIANTES

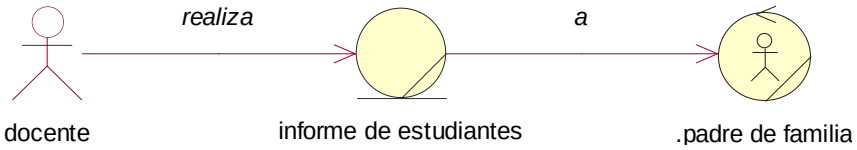


Figura 29. M.O.N. INFORMA RENDIMIENTO DE ESTUDIANTES

II.1.5.2.3.7 MODELO DE OBJETO CONTROLA ASISTENCIA DE ESTUDIANTES

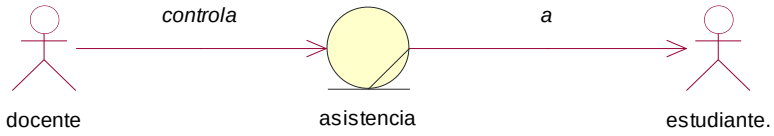


Figura 30. M.O.N. CONTROLA ASISTENCIA DE ESTUDIANTES

II.1.5.2.3.8 MODELO DE OBJETO SE ENCARGA DE UN CURSO

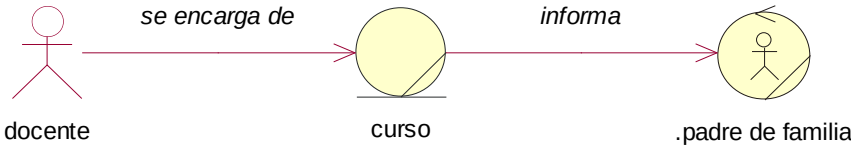


Figura 31. M.O.N. ENCARGA DE UN CURSO

II.1.5.2.4 PADRES DE FAMILIA

II.1.5.2.4.1 MODELO DE OBJETO ENTREGA REQUISITOS

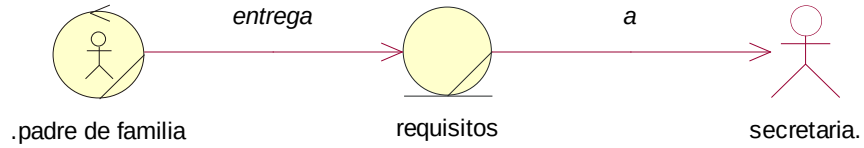


Figura 32. M.O.N. ENTREGA REQUISITOS

II.1.5.2.4.2 MODELO DE OBJETO SOLICITA INSCRIPCION

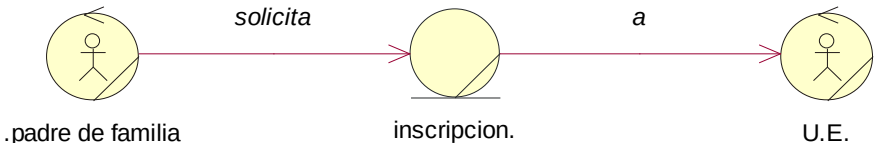


Figura 33. M.O.N. SOLICITA INSCRIPCION

II.1.6 MODELOS DE CASO DE USO

II.1.6.1 Introducción

Los casos de uso nos sirven para presentar la manera de cómo el cliente interactúa con el sistema que se desarrolla

II.1.6.1.1 Propósito

- Modelar el contexto del sistema
- Modelar los requerimientos del sistema
- Identificar los procesos del sistema
- Estimular a que los usuarios potenciales hablen del sistema desde su propio punto de vista
- Involucrar a los usuarios en las etapas iniciales del análisis y diseño del sistema
- Ayudar en la obtención de los requerimientos desde el punto de vista del usuario

II.1.6.1.2 Alcance

- Describir lo que el sistema informático realizara dentro del negocio
- Describir los alcances del sistema
- Describir los procesos del sistema y del cliente

II.1.6.2 Diagrama de Casos De Uso

II.1.6.2.1 Caso de Uso General

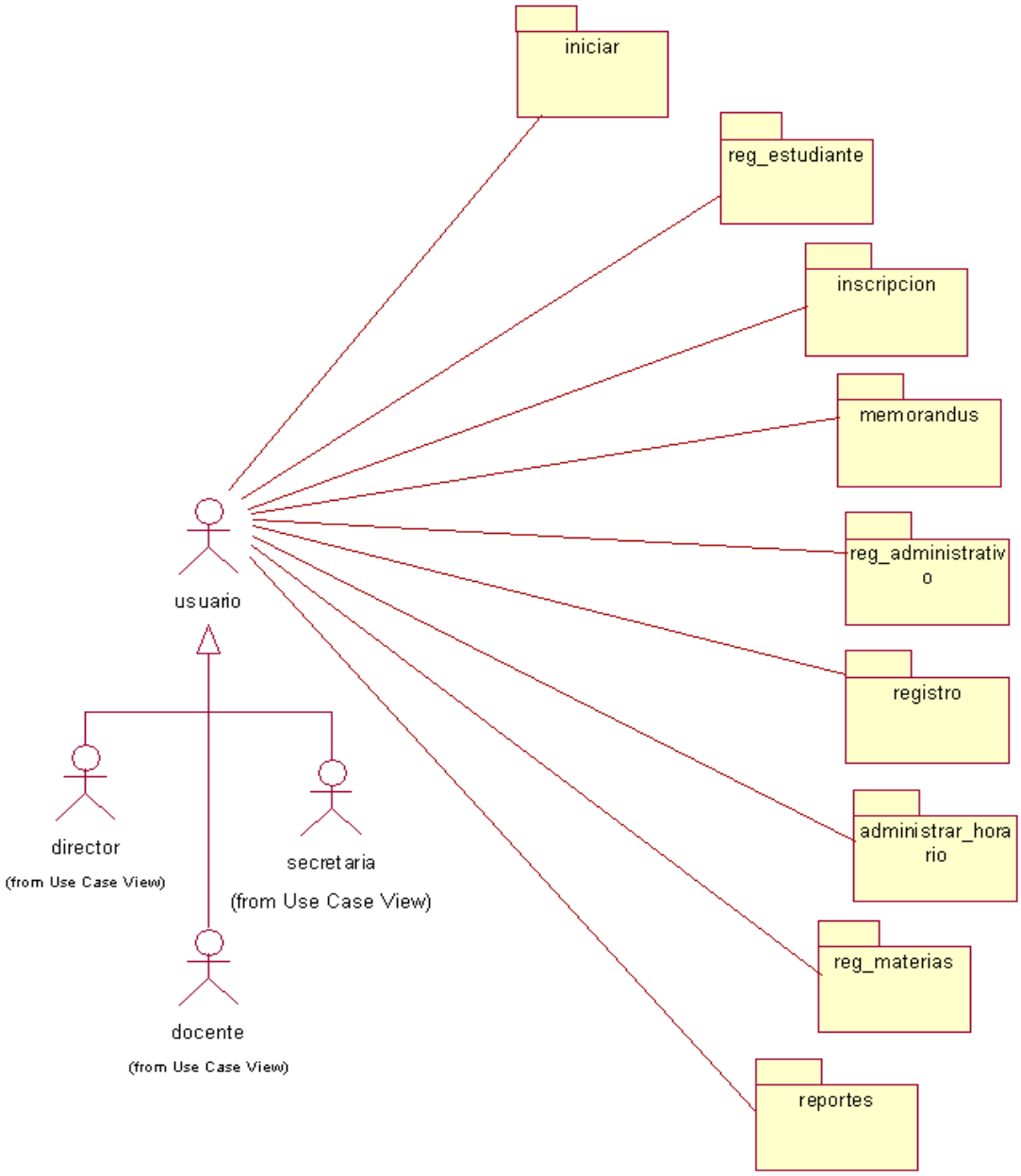


Figura 34. D.C.U. Caso de Uso General

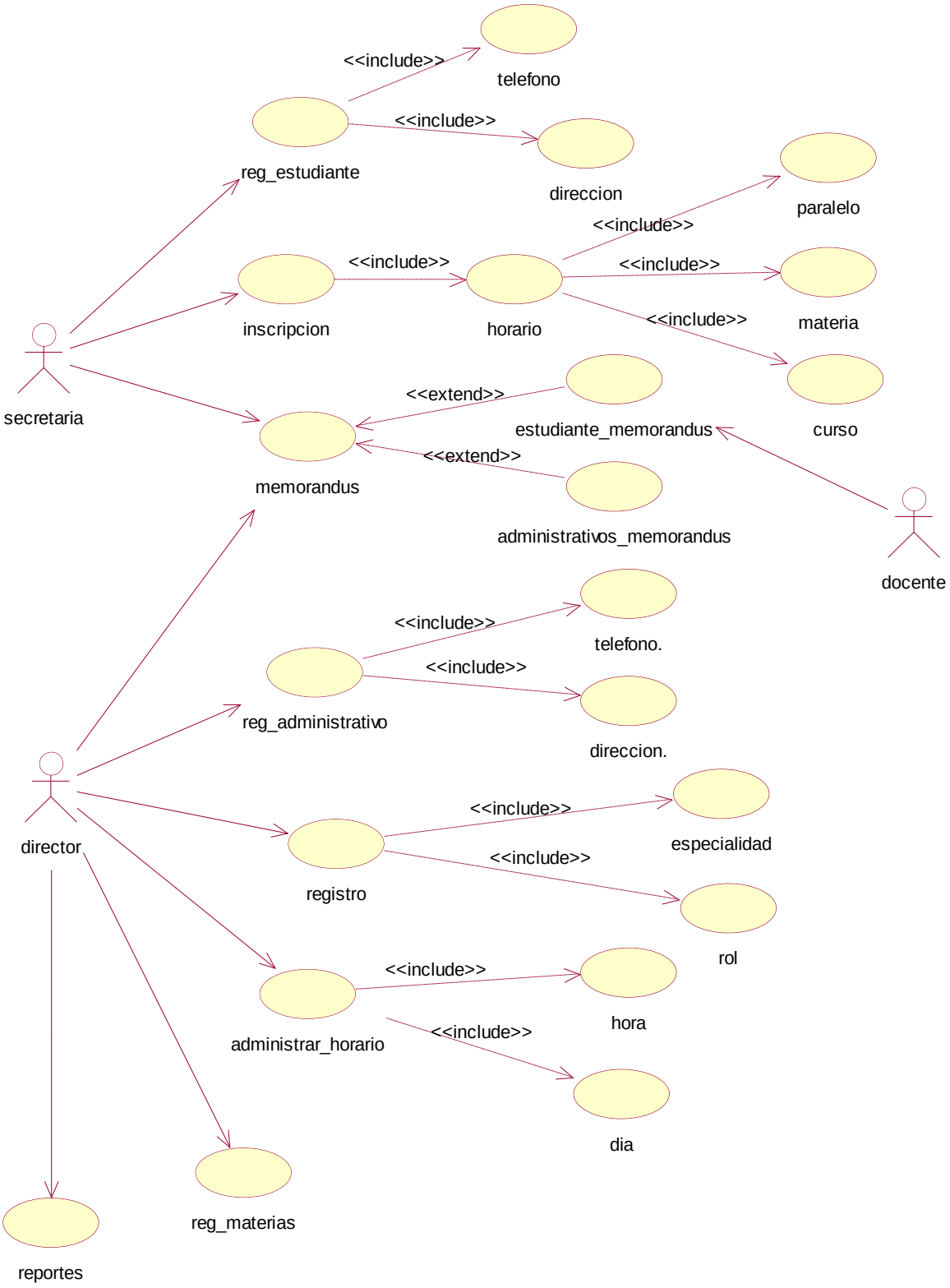


Figura 35. D.C.U. Caso de Uso General

II.1.6.2.2 Caso de Uso Específico

II.1.6.2.2.1. INICIAR

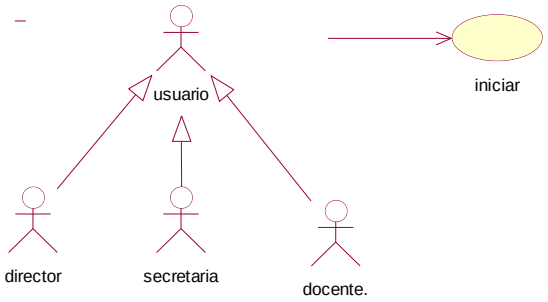


Figura 36. D.C.U. INICIAR

II.1.6.2.2.2. REG ESTUDIANTES

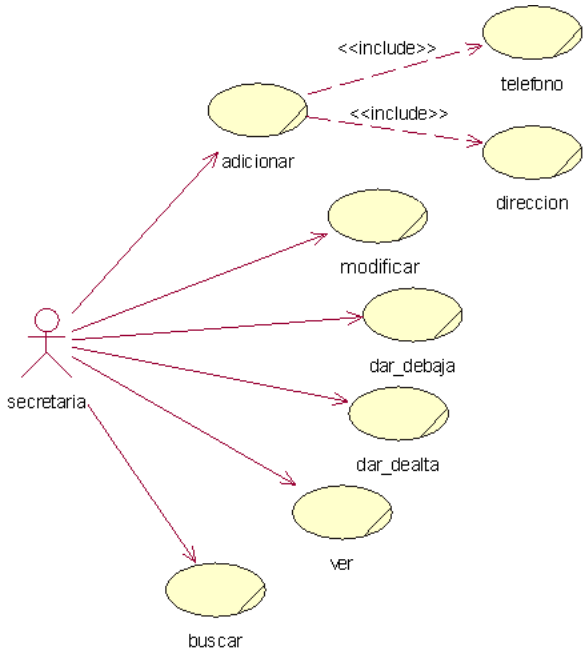


Figura 37. D.C.U. REG ESTUDIANTES

II.1.6.2.2.3. INSCRIPCION

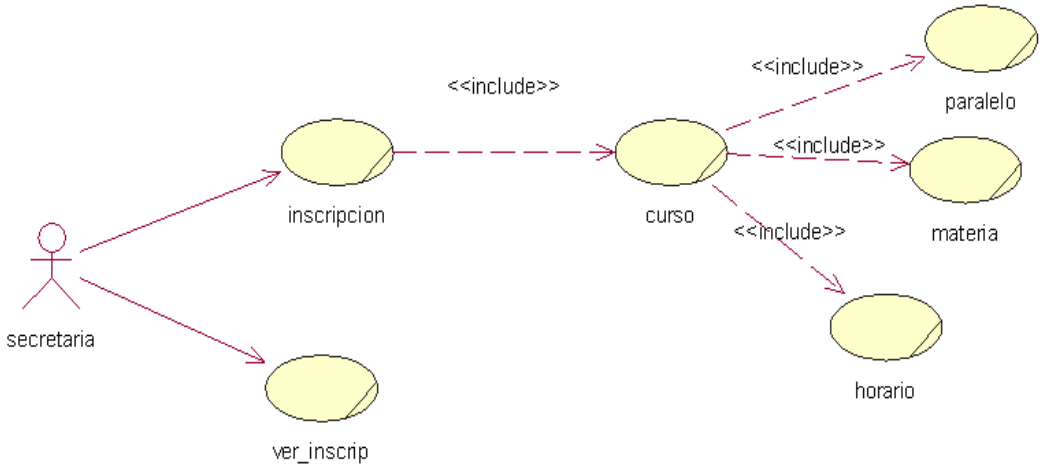


Figura 38. D.C.U. INSCRIPCION

II.1.6.2.2.4. MEMORANDUS

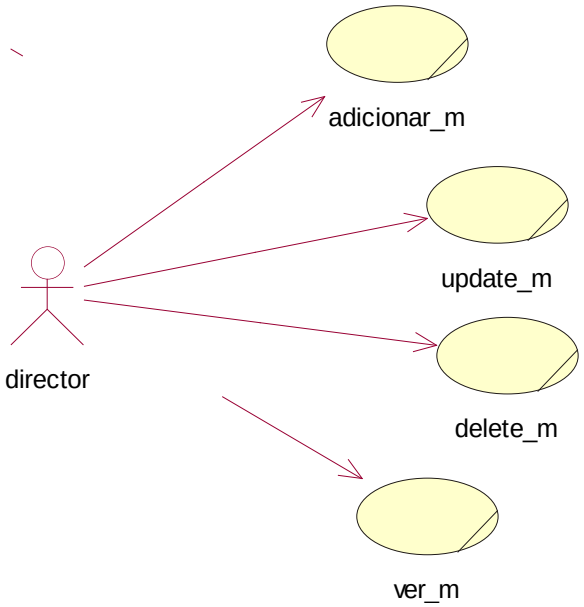


Figura 39. D.C.U. MEMORANDUS

II.1.6.2.2.5. REG_ADMINISTRATIVO

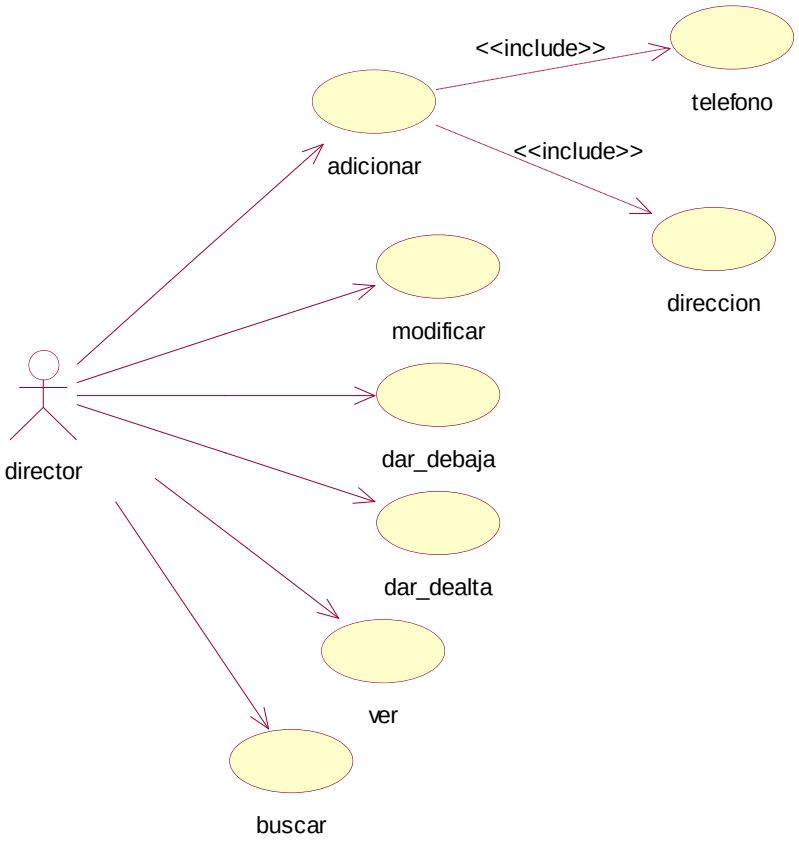


Figura 40. D.C.U. REG_ADMINISTRATIVO

II.1.6.2.2.6. REGISTRO

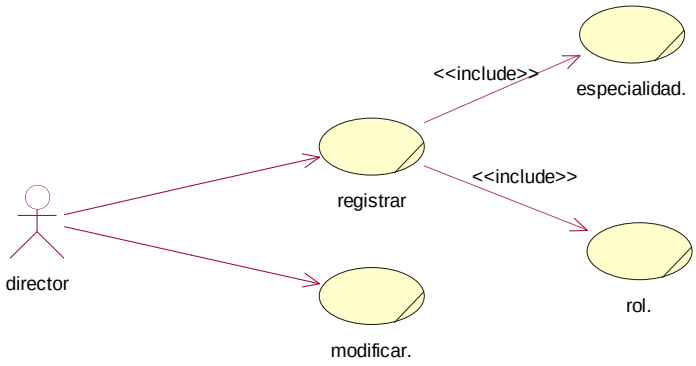


Figura 41. D.C.U. REGISTRO

II.1.6.2.2.7. REG MATERIAS

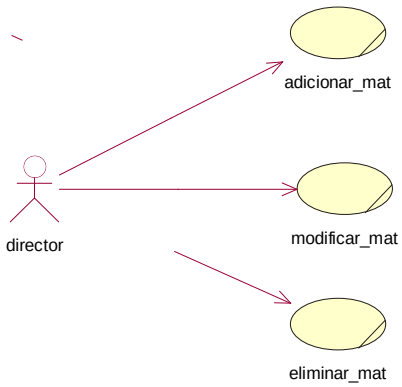


Figura 42. D.C.U. REG MATERIAS

II.1.6.2.2.8. ADMINISTRAR HORARIO

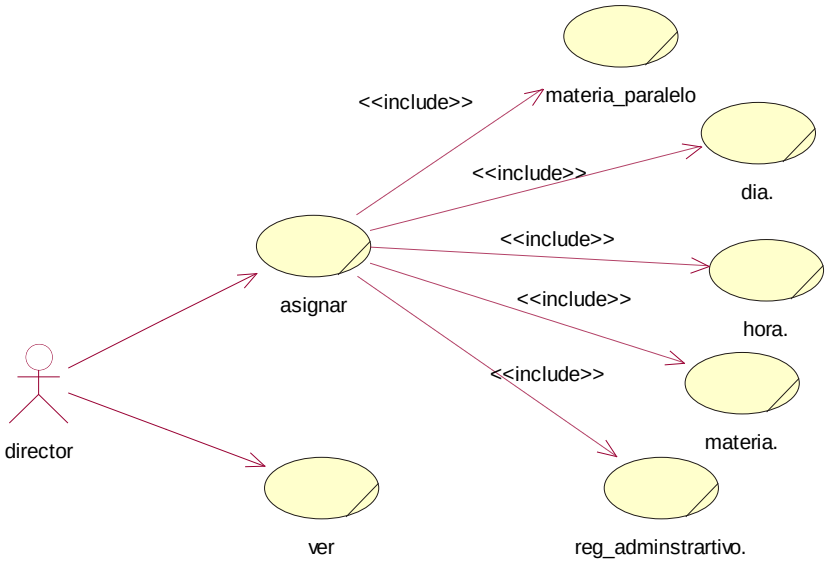


Figura 43. D.C.U. ADMINISTRAR HORARIO

II.1.6.2.2.9. REPORTES

Figura 44. D.C.U. REPORTES

II.1.7 Especificaciones de Casos de Uso

Para los casos de uso que lo requieran (cuya funcionalidad no sea evidente o que no baste con una simple descripción narrativa) se realiza una descripción detallada utilizando una plantilla de documento, donde se incluyen: precondiciones, post-condiciones, flujo de eventos, requisitos no-funcionales asociados. También, para casos de uso cuyo flujo de eventos sea complejo podrá adjuntarse una representación gráfica mediante un Diagrama de Actividad.

II.1.7.1. Introducción

Las especificaciones de los Casos de Uso es una descripción detallada de los casos de uso identificados en el análisis.

II.1.7.1.1. Propósito

Describir específicamente cada caso de uso. Comprender el funcionamiento de los casos de Uso del Sistema.

II.1.7.1.2. Alcance

- Describir los procesos internos del sistema en cada Caso de Uso
- Detallar los flujos de cada caso según lo establecido.

II.1.7.2. Especificación de los actores

II.1.7.2.1 Director

El director es el actor que se encarga de administrar el sistema, posee el rol con mas privilegios y puede acceder a todas las funcionalidades del sistema.

II.1.7.2.2. Secretaria

La secretaria es el actor que se encarga de administrar el sistema, posee un rol con privilegios limitados.

II.1.7.2.3. Administrativo

Este es el actor que representa a los administrativos que trabajan en el colegio. Solo el actor administrativo que tenga el rol de docente, tiene acceso a ver el kardex del estudiante y comentar sobre ellos. Los administrativos que tengan un rol distinto solo pueden acceder a ver el sistema, es decir que no tienen ningún privilegio ni clave de acceso.

II.1.7.2.4. Padre de Familia

Es el actor que representa a los padres de los estudiantes de la Unida Educativa, no tiene acceso directo al sistema, no goza de ningún privilegio, usuario o contraseña para ingresar al sistema, este es visualizado por un docente y enseñado a los padres de familia para informar acerca del desempeño educativo de sus hijos.

II.1.7.2.5. Estudiante

Es el actor que representa a los estudiantes de la Unida Educativa, no tienen acceso directo al sistema, no goza de ningún privilegio, usuario o contraseña para ingresar al sistema.

II.1.7.3. Especificación de casos de Uso

II.1.7.3.1 Iniciar

Caso de Uso	**Iniciar**
Usuario:	**Usuario**
Descripción: En este caso de uso el usuario debe introducir su usuario y clave	
Propósito: Tener acceso al sistema.	

Flujo Principal: 1.- El usuario introduce su usuario. 2.- El usuario introduce su clave.
Subflujo: Acceso al sistema Mensaje: Clave i/o Contraseña incorrectas.
Flujo de Excepción: Mensaje error: Introducir Clave i/o Contraseña

Tabla 26. E.C.U. Iniciar

II.1.7.3.2 reg estudiante

Caso de Uso	**reg estudiante**
Usuario:	**secretaria**
Descripción: En este caso de uso la secretaria registra al estudiante antes de su inscripción, permitiendo hacer modificaciones, dar de baja o dar de alta a los estudiantes, para ello se lista a todos los estudiantes que se hayan registrado en la base de datos de estudiante. También cuenta con un filtro que permite buscar a dicho estudiante introduciendo las iniciales de su apellido, visualizar los estudiantes activos y los estudiantes pasivos.	
Propósito: Realizar un registro previo a la inscripción del estudiante, mediante la cual también podemos:	

1.- Agregar un nuevo estudiante

2.- Modificar un estudiante.

3.- Dar de baja un estudiante

4.- Dar de Alta un estudiante

5.- Ver estudiante

Flujo Principal:

1.- El usuario Selecciona reg estudiante.

2.- Se listan los datos de los estudiantes ya registrados.

3.- El usuario selecciona alguna opción

3.1.-El usuario puede seleccionar adicionar estudiante.

3.2.-El usuario puede Seleccionar modificar estudiante, en donde modificará algunos datos.

3.3.-El usuario puede seleccionar dar de baja a un estudiante.

3.4.- El usuario puede seleccionar dar de alta a un estudiante, en caso de que el estudiante este dado de baja.

3.5.- El usuario puede seleccionar ver estudiante.

Subflujo:

Mensaje: Se adiciono satisfactoriamente.

Mensaje: Se modifico satisfactoriamente.

Mensaje: (dar de baja) ¿seguro de cambiar de estado? Si (se cambio de estado satisfactoriamente)

Mensaje: (dar de alta) ¿seguro de cambiar de estado? Si (se cambio de estado satisfactoriamente)

Flujo de Excepción: Mensaje error: el ci introducido ya se encuentra registrado. (Ocurre al momento de agregar un nuevo estudiante, si el ci ya estaba registrado en la base de datos). Mensaje error: los campos marcados están vacíos o son erróneos. (Este mensaje se da cuando al realizar el registro existen algunos campos que están vacíos o están con datos que no corresponden a ese campo para su registro).

Tabla 27. E.C.U. reg_estudiante

II.1.7.3.3 adicionar (estudiante)

Caso de Uso	**adicionar (estudiante)**
Usuario:	**Secretaria**
Descripción: Es el caso de uso donde la secretaria puede adicionar un nuevo estudiante.	
Propósito: Adicionar un nuevo estudiante.	
Precondición : Estar dentro del Sistema Estar logeada como secretaria o director, para gozar de estos privilegios.	
Flujo Principal: Selecciona Opción Adicionar (estudiante) Llena los datos del nuevo estudiante Selecciona Adicionar	

Flujo de Excepción: **Mensaje:** Los campos no son validos **Mensaje:** campo CI solo aceptan números **Mensaje:** El estudiante ya existe

Tabla 28. E.C.U. adicionar

II.1.7.3.4 modificar (estudiante)

Caso de Uso	**modificar (estudiante)**
Usuario:	**secretaria**
Descripción: Es el caso de uso donde la secretaria puede modificar algunos datos del estudiante dentro de la base de datos.	
Propósito: Modificar los datos del estudiante	
Precondición : Estar dentro del Sistema. Estar Logueada como secretaria o director.	
Flujo Principal: Selecciona estudiante de la Lista Selecciona opción modificar Llena los nuevos datos Selecciona modificar	
Flujo de Excepción:	

Mensaje: Los campos no son validos. (Si alguno de los datos no es válido)

Tabla 29. E.C.U. modificar

II.1.7.3.5 dar de baja (estudiante)

Caso de Uso	**dar de baja (estudiante)**
Usuario:	**Secretaria**
Descripción: dar de baja a algún estudiante, por motivo de abandono o retiro de la Unidad Educativa u otros.	
Propósito: Dar de baja al estudiante seleccionado.	
Precondición : Estar dentro del Sistema Estar Logueado como Secretaria o director.	
Flujo Principal: Selecciona un estudiante de la Lista Selecciona opción estado	
Subflujo: **Mensaje:** ¿Seguro de cambiar de estado? **Mensaje:** se cambio de estado satisfactoriamente	
Flujo de Excepción: **Mensaje:** ..	

Tabla 30. E.C.U. dar de baja

II.1.7.3.6 dar de alta (estudiante)

Caso de Uso	**dar de alta (estudiante)**
Usuario:	**Secretaria**
Descripción: dar de alta a algún estudiante, por motivo de reincorporación a la Unidad Educativa u otros motivos.	
Propósito: Dar de alta al estudiante seleccionado.	
Precondición : Estar dentro del Sistema Estar Logueado como Secretaria o director.	
Flujo Principal: Selecciona un estudiante de la Lista Selecciona opción estado	
Subflujo: **Mensaje:** ¿Seguro de cambiar de estado? **Mensaje:** se cambio de estado satisfactoriamente	
Flujo de Excepción: **Mensaje:** ..	

Tabla 31. E.C.U. dar de alta

II.1.7.3.7 ver (estudiante)

Caso de Uso	**ver (estudiante)**

Usuario:	**Secretaria**
Descripción: Ver los datos del estudiante.	
Propósito: Llevar un control de los estudiantes, y ver los datos actualizados del mismo en el momento que sea requerido.	
Precondición : Estar dentro del Sistema Estar Logueado como secretaria o director	
Flujo Principal: Selecciona estudiante de la Lista Selecciona opción ver	

Tabla 32. E.C.U. ver

II.1.7.3.8 Inscripción

Caso de Uso	**Inscripción**
Usuario:	**Secretaria o Director**
Descripción: En este caso de uso el director o la secretaria realiza la inscripción del estudiante después de registrar al estudiante. En esta opción se hace click en el curso y paralelo, para luego visualizar la lista de todos los estudiantes registrados (aun no inscritos), lista en la cual se selecciona al estudiante que se desea inscribir en el curso seleccionado, también existe un buscador que	

buscara al estudiante introduciendo las iniciales del apellido.
Propósito: Realizar la inscripción correcta del estudiante en la Unidad Educativas, en esta opción podemos realizar las siguientes: 1.- Realizar inscripción. 2.- Ver las inscripción es realizadas.
Flujo Principal: 1.- El usuario Selecciona opción inscripción 2.- Se listan los cursos con sus respectivos paralelos. 3.- El usuario selecciona alguna opción 3.1.-El usuario selecciona el curos y el paralelo donde desea inscribir al estudiante 3.2.-El usuario selecciona al estudiante que va a inscribir.
Subflujo: Mensaje: Se adiciono satisfactoriamente. Mensaje: No se modifico satisfactoriamente.

Tabla 33. E.C.U. inscripcion

II.1.7.3.9 Ver_inscrip

Caso de Uso	**Ver_inscrip**
Usuario:	**Director o secretaria**
Descripción: Es el caso de uso donde el director o la secretaria pueden ver las	

inscripciones que realizaron en cada curso.
Propósito: Ver las inscripciones de los cursos
Precondición : Estar dentro del Sistema Estar logeada como director o secretaria, para gozar de estos privilegios.
Flujo Principal: Selecciona Opción ver inscripcion Selecciona el curso que desea visualizar

Tabla 34. E.C.U. ver_inscrip

II.1.7.3.10 memorandus

Caso de Uso	**memorandus**
Usuario:	**Secretaria o Director**
Descripción: En este caso de uso el director o la secretaria administra los memorandus, permitiendo adicionar, modificar, eliminar y ver los memorandus, para ello se lista todos los memorandus que se hayan registrado en la base de datos memorandus.	
Propósito: Realizar una administración de memorandus existentes en la Unidad Educativas y Distrito Escolar, mediante el cual podemos: 1.- Adicionar un nuevo memorandus. 2.- Modificar un memorandus.	

3.-Elimianr un memorandus 4.- Ver meorandus
Flujo Principal: 1.- El usuario Selecciona opcion memorandus 2.- Se listan los memorandus ya registrados. 3.- El usuario selecciona alguna opción 3.1.-El usuario puede seleccionar adicionar memorandus. 3.2.-El usuario puede Seleccionar modificar memorandus, en donde modificará algunos datos. 3.3.-El usuario puede seleccionar eliminar un memorandus. 3.4.-El usuario puede seleccionar ver memorandus
Subflujo: Mensaje: Se adiciono satisfactoriamente. Mensaje: Se modifico satisfactoriamente. Mensaje: ¿seguro de eliminar? Si (se elimino satisfactoriamente)
Flujo de Excepción: Mensaje error: los campos están vacíos o son erróneos. (Este mensaje se da cuando existe algún error al introducir los datos).

Tabla 35. E.C.U. memorandus

II.1.7.3.11 adicionar_m (memorandus**)**

Caso de Uso	**adicionar_m (**memorandus**)**
Usuario:	**Director o secretaria**
Descripción: Es el caso de uso donde el director o la secretaria puede adicionar un nuevo memorandus.	
Propósito: Adicionar un nuevo memorandus	
Precondición : Estar dentro del Sistema Estar logeada como director o secretaria, para gozar de estos privilegios.	
Flujo Principal: Selecciona Opción Adicionar (memoradus) Llena los datos del nuevo memorandus Selecciona Adicionar	
Flujo de Excepción: **Mensaje:** Los campos no son validos	

Tabla 36. E.C.U. adicionar_m

II.1.7.3.12 modificar_m (memorandus)

Caso de Uso	**modificar_m (memorandus)**
Usuario:	**Director o secretaria**
Descripción: Es el caso de uso donde el director o la secretaria puede modificar algunos datos del memorandus dentro de la base de datos.	

Propósito: Modificar los datos del memorandus
Precondición : Estar dentro del Sistema. Estar Logueada como director o secretaria.
Flujo Principal: Selecciona memorandus de la Lista Selecciona opción modificar Llena los nuevos datos Selecciona modificar
Flujo de Excepción: **Mensaje:** Los campos no son validos. (Si alguno de los datos no es válido)

Tabla 37. E.C.U. modificar_m

II.1.7.3.13 eliminar_m

Caso de Uso	**eliminar_m**
Usuario:	**Director o secretaria**
Descripción: eliminar memorandus, por motivo de eliminación de algún memorandus del sistema distrital, Unidad Educativa u otros.	
Propósito: Eliminar el memorandus seleccionado.	
Precondición :	

Estar dentro del Sistema Estar Logueado como director o secretaria.
Flujo Principal: Selecciona un memorandus de la Lista
Subflujo: **Mensaje:** ¿Seguro de eliminar memorandus? **Mensaje:** se elimino satisfactoriamente
Flujo de Excepción: **Mensaje:** ..

Tabla 38. E.C.U. eliminar_m

II.1.7.3.14 Ver_m (memorandus)

Caso de Uso	**Ver (memerandus)**
Usuario:	**Director o secretaria**
Descripción: Ver memorandus	
Propósito: Llevar un control de los memorandus para ver los datos actualizados del mismo en el momento que sea requerido.	
Precondición : Estar dentro del Sistema Estar Logueado como director o secretaria	
Flujo Principal:	

Selecciona memorandus de la Lista Selecciona opción ver

Tabla 39. E.C.U. Ver (memerandus)

II.1.7.3.15 Reg administrativo

Caso de Uso	**Reg administrativo**
Usuario:	**Director**
Descripción: En este caso de uso el director registra al administrativo antes de su registro, permitiendo hacer modificaciones, dar de baja o dar de alta a los administrativos, para ello se lista a todos los administrativos que se hayan registrado en la base de datos de administrativo. También cuenta con un filtro que permite buscar a dicho administrativo introduciendo las iniciales de su apellido, visualizar los administrativos activos y los administrativos pasivos.	
Propósito: Realizar un registro del administrativo previo al registro profesional del administrativo, mediante la cual también podemos: 1.- Agregar un nuevo administrativo 2.- Modificar un administrativo. 3.- Dar de baja un administrativo 4.- Dar de Alta un administrativo 5.- Ver administrativo	

Flujo Principal:

1.- El usuario Selecciona reg administrativo.

2.- Se listan los datos de los administrativo ya registrados.

3.- El usuario selecciona alguna opción

3.1.-El usuario puede seleccionar adicionar administrativo.

3.2.-El usuario puede Seleccionar modificar administrativo, en donde modificará algunos datos.

3.3.-El usuario puede seleccionar dar de baja a un administrativo.

3.4.- El usuario puede seleccionar dar de alta a un administrativo en caso de que el administrativo este dado de baja.

3.5.- El usuario puede seleccionar ver administrativo.

Subflujo:

Mensaje: Se adiciono satisfactoriamente.

Mensaje: Se modifico satisfactoriamente.

Mensaje: (dar de baja) ¿seguro de cambiar de estado? Si (se cambio de estado satisfactoriamente)

Mensaje: (dar de alta) ¿seguro de cambiar de estado? Si (se cambio de estado satisfactoriamente)

Flujo de Excepción:

Mensaje error: el ci introducido ya se encuentra registrado. (Ocurre al momento de agregar un nuevo estudiante, si el ci ya estaba registrado en la base de datos).

Mensaje error: los campos marcados están vacíos o son erróneos. (Este mensaje se da cuando al realizar el registro existen algunos campos que están vacíos o

están con datos que no corresponden a ese campo para su registro).

Tabla 40. E.C.U. Reg administrativo

II.1.7.3.16 Adicionar (administrativo**)**

Caso de Uso	**Adicionar (**administrativo**)**
Usuario:	**Director**
Descripción: Es el caso de uso donde el director puede adicionar un nuevo administrativo.	
Propósito: Adicionar un nuevo administrativo.	
Precondición : Estar dentro del Sistema Estar logeada como director, para gozar de estos privilegios.	
Flujo Principal: Selecciona Opción Adicionar (administrativo) Llena los datos del nuevo administrativo Selecciona Adicionar	
Flujo de Excepción: **Mensaje:** Los campos no son validos **Mensaje:** campo CI solo aceptan números **Mensaje:** El estudiante ya existe	

Tabla 41. E.C.U. Adicionar

II.1.7.3.17 Modificar (administrativo)

Caso de Uso	**Modificar (administrativo)**
Usuario:	**Director**
Descripción: Es el caso de uso donde el director puede modificar algunos datos del administrativo dentro de la base de datos.	
Propósito: Modificar los datos del administrativo	
Precondición : Estar dentro del Sistema. Estar Logueada como director.	
Flujo Principal: Selecciona administrativo de la Lista Selecciona opción modificar Llena los nuevos datos Selecciona modificar	
Flujo de Excepción: **Mensaje:** Los campos no son validos. (Si alguno de los datos no es válido)	

Tabla 42. E.C.U. Modificar

II.1.7.3.18 Dar de baja (administrativo)

Caso de Uso	**Dar de baja (administrativo)**
Usuario:	**Director**
Descripción: dar de baja a algún administrativo, por motivo de abandono o retiro	

de la Unidad Educativa u otros.
Propósito: Dar de baja al administrativo seleccionado.
Precondición : Estar dentro del Sistema Estar Logueado como director.
Flujo Principal: Selecciona un administrativo de la Lista Selecciona opción estado
Subflujo: **Mensaje:** ¿Seguro de cambiar de estado? **Mensaje:** se cambio de estado satisfactoriamente
Flujo de Excepción: **Mensaje:** ..

Tabla 43. E.C.U. Dar de baja

II.1.7.3.19 Dar de alta (administrativo)

Caso de Uso	**Dar de alta (administrativo)**
Usuario:	**Director**
Descripción: dar de alta a algún administrativo, por motivo de reincorporación a la Unidad Educativa u otros motivos.	
Propósito:	

Dar de alta al administrativo seleccionado.
Precondición : Estar dentro del Sistema Estar Logueado como director.
Flujo Principal: Selecciona un administrativo de la Lista Selecciona opción estado
Subflujo: **Mensaje:** ¿Seguro de cambiar de estado? **Mensaje:** se cambio de estado satisfactoriamente
Flujo de Excepción: **Mensaje:** ..

Tabla 44. E.C.U. Dar de alta

II.1.7.3.20 Ver (administrativo)

Caso de Uso	**Ver (administrativo)**
Usuario:	**Director**
Descripción: Ver los datos del administrativo.	
Propósito: Llevar un control de los adminstrativos, y ver los datos actualizados del mismo en el momento que sea requerido.	
Precondición :	

Estar dentro del Sistema Estar Logueado como director
Flujo Principal: Selecciona administrativo de la Lista Selecciona opción ver

Tabla 45. E.C.U. Ver (administrativo)

II.1.7.3.21 reg materias

Caso de Uso	**reg materias**
Usuario:	**Secretaria o Director**
Descripción: En este caso de uso el director o la secretaria administra las materias, permitiendo adicionar, modificar y eliminar las materias, para ello se lista todas las materias que se hayan registrado en la base de datos materia.	
Propósito: Realizar una administración de las materias existentes en la Unidad Educativas, mediante el cual podemos: 1.- Adicionar una nueva materia. 2.- Modificar materias. 3.-Eliminar materias	
Flujo Principal: 1.- El usuario Selecciona opcion reg_materia 2.- Se listan las materias ya registrados.	

3.- El usuario selecciona alguna opción 3.1.-El usuario puede seleccionar adicionar materia. 3.2.-El usuario puede Seleccionar modificar materia en donde modificará algunos datos. 3.3.-El usuario puede seleccionar eliminar un materia.
Subflujo: Mensaje: Se adiciono satisfactoriamente. Mensaje: Se modifico satisfactoriamente. Mensaje: ¿seguro de eliminar? Si (se elimino satisfactoriamente)
Flujo de Excepción: Mensaje error: los campos están vacíos o son erróneos. (Este mensaje se da cuando existe algún error al introducir los datos).

Tabla 46. E.C.U. reg materias

II.1.7.3.22 adicionar_mat (materias)

Caso de Uso	**adicionar_mat (materias)**
Usuario:	**Director o secretaria**
Descripción: Es el caso de uso donde el director o la secretaria puede adicionar una nueva materia	
Propósito: Adicionar una nueva materia	
Precondición : Estar dentro del Sistema	

Estar logeada como director o secretaria, para gozar de estos privilegios.
Flujo Principal: Selecciona Opción adicionar (materias) Llena los datos de la nueva materia Selecciona Adicionar
Flujo de Excepción: **Mensaje:** Los campos no son validos

Tabla 47. E.C.U. adicionar_mat (materias)

II.1.7.3.23 modificar_mat (materias)

Caso de Uso	**modificar_m (materias)**
Usuario:	**Director o secretaria**
Descripción: Es el caso de uso donde el director o la secretaria puede modificar algunos datos de la materia dentro de la base de datos.	
Propósito: Modificar los datos de la metria seleccionada	
Precondición : Estar dentro del Sistema. Estar Logueada como director o secretaria.	
Flujo Principal: Selecciona materia de la Lista	

Selecciona opción modificar Llena los nuevos datos Selecciona modificar
Flujo de Excepción: **Mensaje:** Los campos no son validos. (Si alguno de los datos no es válido)

Tabla 48. E.C.U. modificar_m (materias)

II.1.7.3.23 eliminar_mat (materias)

Caso de Uso	**eliminar_mat (materias)**
Usuario:	**Director o secretaria**
Descripción: eliminar materias seleccionada, por motivo de eliminación de alguna materia del sistema distrital, Unidad Educativa u otros.	
Propósito: Eliminar la materia seleccionado.	
Precondición : Estar dentro del Sistema Estar Logueado como director o secretaria.	
Flujo Principal: Selecciona una materia de la Lista	
Subflujo: **Mensaje:** ¿Seguro de eliminar materia?	

Mensaje: se elimino satisfactoriamente
Flujo de Excepción: **Mensaje:** ..

Tabla 49. E.C.U. eliminar_mat (materias)

II.1.7.3.24 administrar horario

Caso de Uso	**administrar horario**
Usuario:	**Director**
Descripción: En este caso de uso el director se encarga de administrar los horarios, es decir la asignación de horarios a los docentes en los cursos que corresponde, ver la asignación realizada y eliminar la asignación realizada.	
Propósito: Realizar una administración de horarios a los cursos existentes, asignados a los docentes registrados: 1.- Asignar horario. 2.-Eliminar asignación 3.- ver asignación	
Flujo Principal: 1.- El usuario Selecciona opción administrar horario 2.- Se listan las materias ya asignadas. 3.- El usuario selecciona el curso y paralelo en el que desea administrar materias 4.- El usuario selecciona alguna opción	

4.1.-El usuario puede seleccionar asignar. 4.2.-El usuario puede Seleccionar eliminar asignación. 4.3.-El usuario puede seleccionar ver asignación.
Subflujo: Mensaje: Se asigno satisfactoriamente
Flujo de Excepción

Tabla 50. E.C.U. administrar horario

II.1.7.3.25 Asignar

Caso de Uso	**Asignar**
Usuario:	**Director**
Descripción: Es el caso de uso donde el director o la secretaria puede ver la asignación realizada.	
Propósito: ver la asignación de materia en el curso y paralelo seleccionado, día y hora y docente	
Precondición : Estar dentro del Sistema Estar logeada como director para gozar de estos privilegios.	
Flujo Principal: 1.- El usuario Selecciona opción administrar horario 2.- El usuario selecciona un curso y paralelo 3.- El usuario selecciona un día y hora	

4.- El usuario seleccionar una materia de la lista de materias disponibles 5.- El usuario selecciona un docente de la lista de docentes disponible

Tabla 51. E.C.U. Asignar

II.1.7.3.26 ver

Caso de Uso	**ver**
Usuario:	**Director**
Descripción: Es el caso de uso donde el director o la secretaria puede ver la asignación realizada.	
Propósito: ver la asignación de materia en el curso y paralelo seleccionado, día y hora y docente	
Precondición : Estar dentro del Sistema Estar logeado como director para gozar de estos privilegios.	
Flujo Principal: Selecciona Opción ver la asignación dando doble clik sobre la materia que se desea ver. Selecciona aceptar para volver	

Tabla 52. E.C.U. ver

II.1.7.3.27 reportes

Caso de Uso	**reportes**
Usuario:	**Director**
Descripción: En este caso de uso el director se encarga de ver los reportes requeridos y definidos en las entrevistas realizadas en el inicio del desarrollo del proyecto. En este caso el usuario selecciona la opción ver el reporte de la lista ya definida.	
Propósito: Ver lo reportes requeridos en el momento que sea necesario para una buena toma de decisiones, tomando en cuenta los datos actualizados de la Unidad Educativa. 1.- ver reportes.	
Flujo Principal: 1.- El usuario Selecciona opción reportes 2.- Se listan la lista de reportes definidos. 3.- El usuario selecciona el reporte que desea visualizar.	

Tabla 53. E.C.U. reportes

II.1.8 MODELO DE ANALISIS Y DISEÑO

II.1.8.1 Modelado de Diagramas de actividades

II.1.8.1.1 Introducción

El diagrama de actividad es un artefacto de la disciplina Requisitos en la metodología RUP la cual estamos implementando.

Los diagramas de actividad se utilizan para modelar los aspectos dinámicos de un sistema, esto implica modelar los pasos secuenciales de un proceso.

II.1.8.1.1.1 Propósito

- Comprender la estructura y la dinámica del sistema deseado para la organización
- Identificar posibles mejoras

II.1.8.1.1.2 Alcance

- Describir los procesos de sistema y los usuarios
- Identificar y definir los procesos de los casos de uso según los objetivos de la organización
- Definir un diagrama de actividad para cada caso de uso.

II.1.8.1.2 Diagramas de Actividad

II.1.8.1.2.1 INICIAR

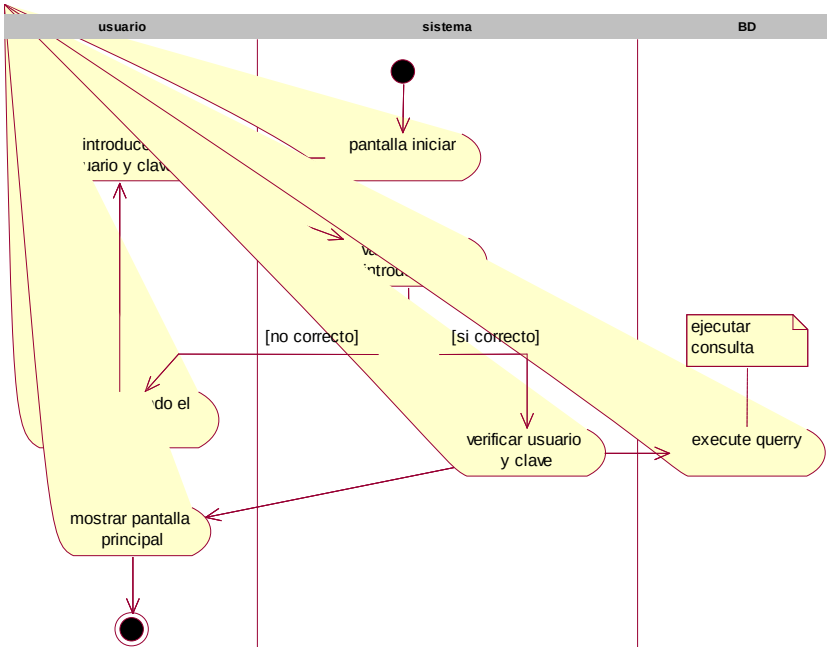


Figura 45. D.A. INICIAR

II.1.8.1.2.2 REG ESTUDIANTES

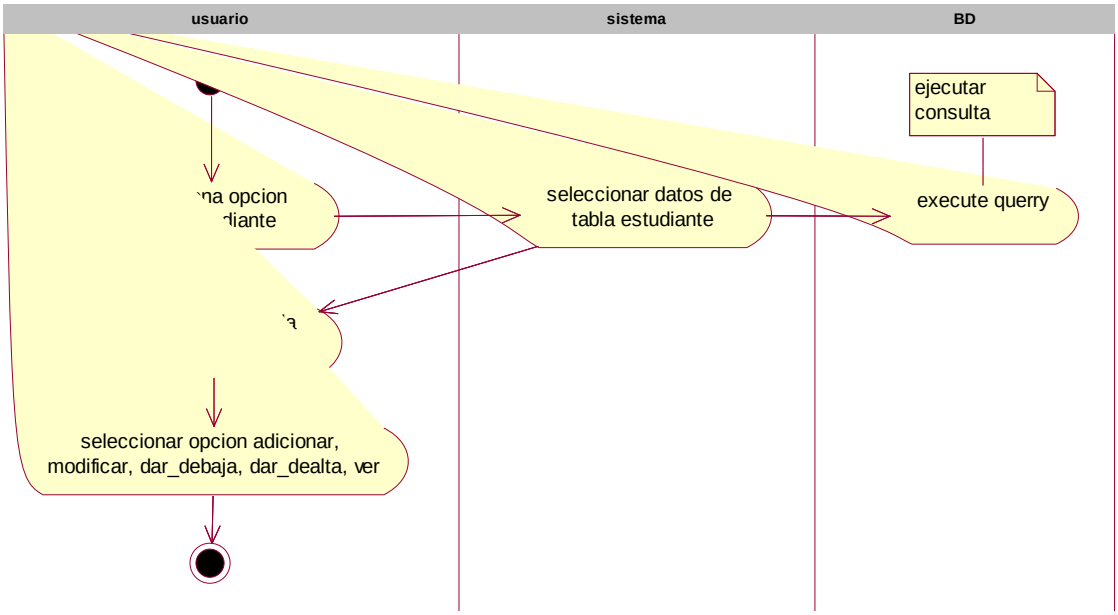


Figura 46. D.A. REG ESTUDIANTES

II.1.8.1.2.3 ADICIONAR

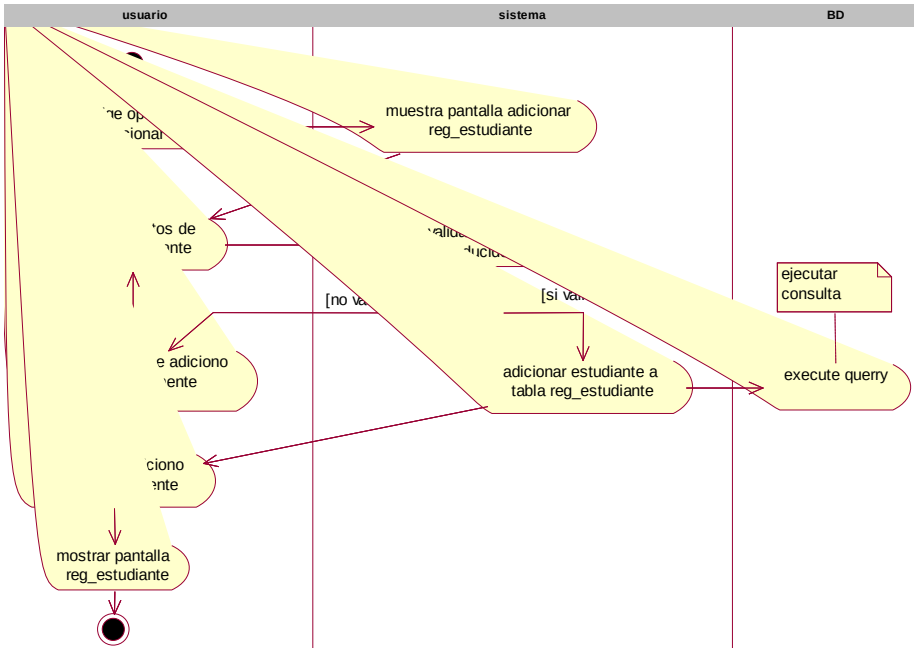


Figura 47. Add_datos

II.1.8.1.2.4 MODIFICAR

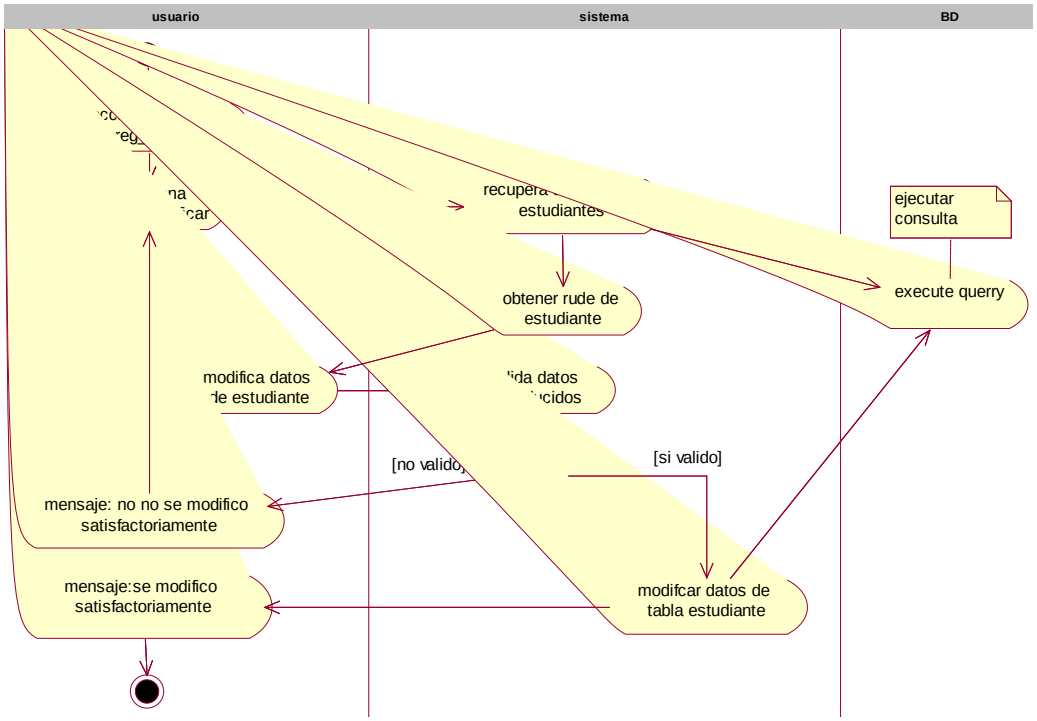


Figura 48. D.A. modificar_datos

II.1.8.1.2.5 DAR DE BAJA

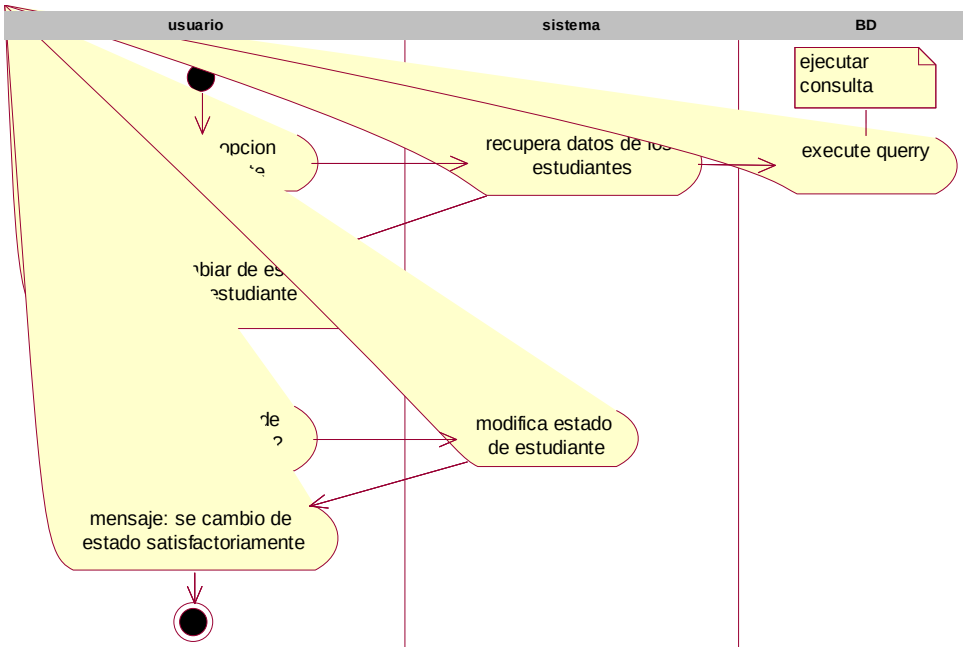


Figura 49. D.A. eliminar_datos

II.1.8.1.2.6 reg_datos

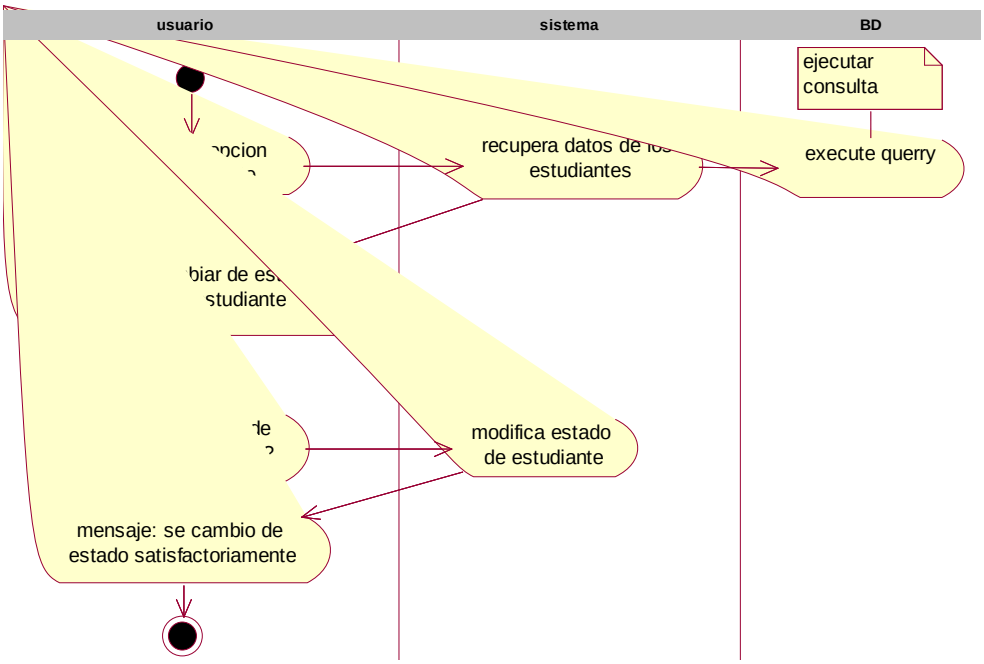


Figura 50. D.A. reg_datos

II.1.8.1.2.7 VER

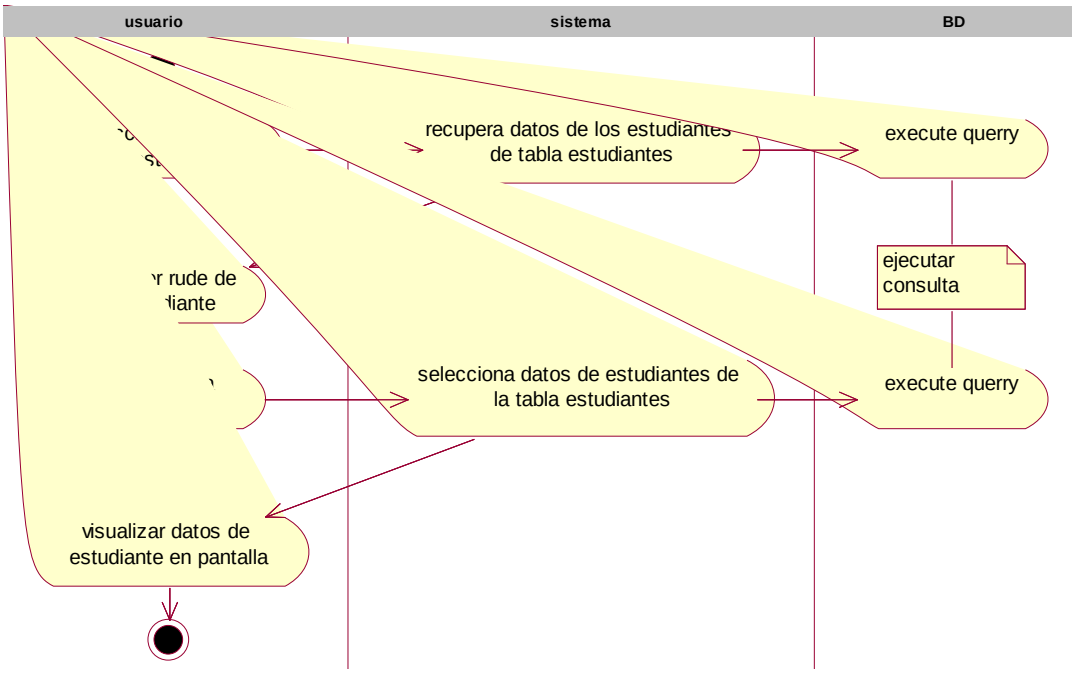


Figura 51. D.A. VER

II.1.8.1.2.8 INSCRIPCION

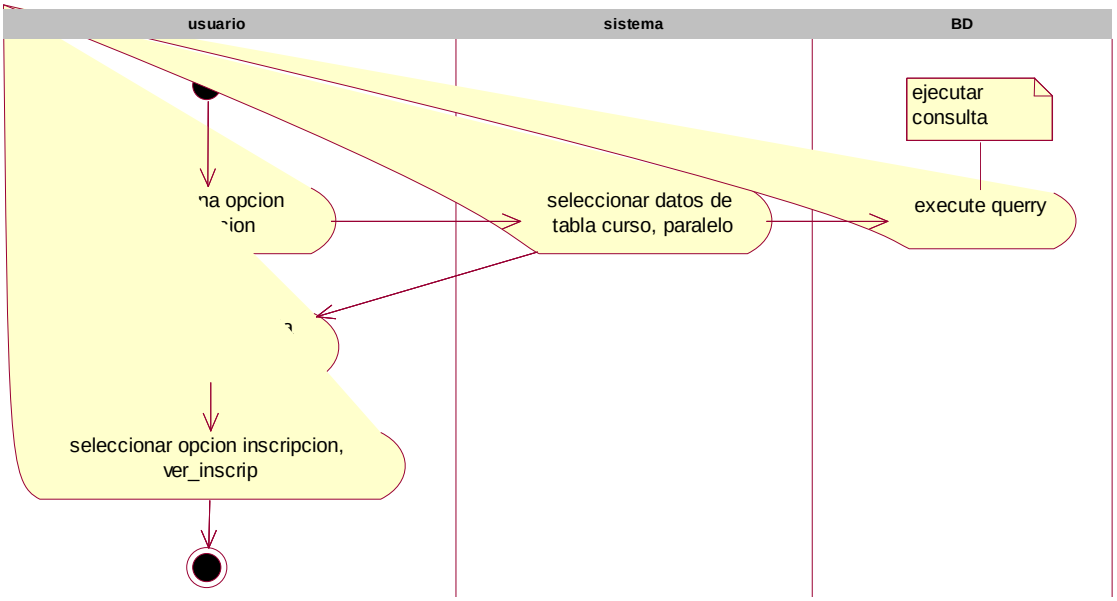


Figura 52. D.A. INSCRIPCION

II.1.8.1.2.9 INSCRIPCION

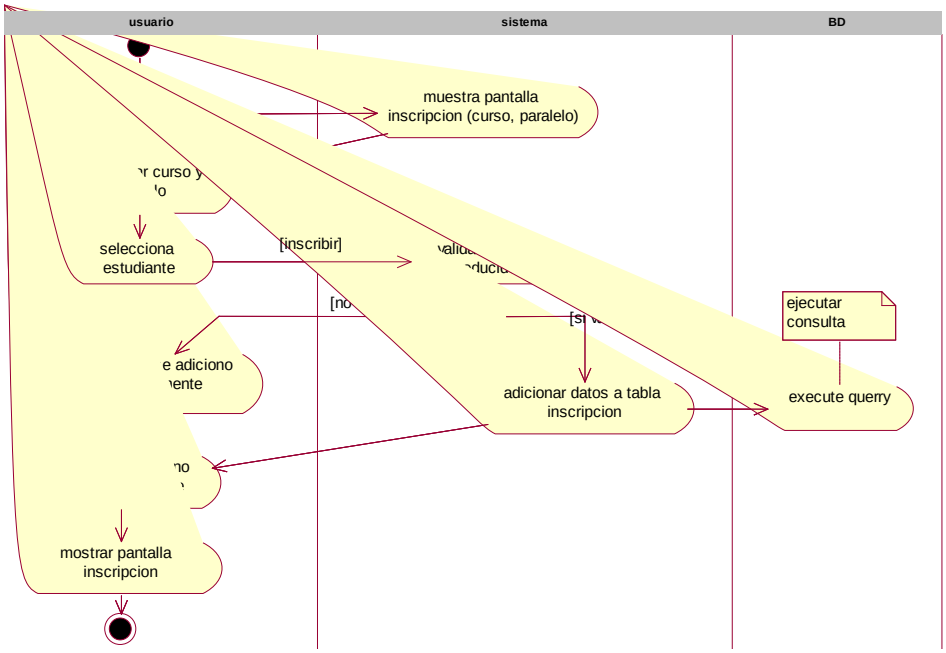


Figura 53. D.A. INSCRIPCION

II.1.8.1.2.10 VER INSCRIP

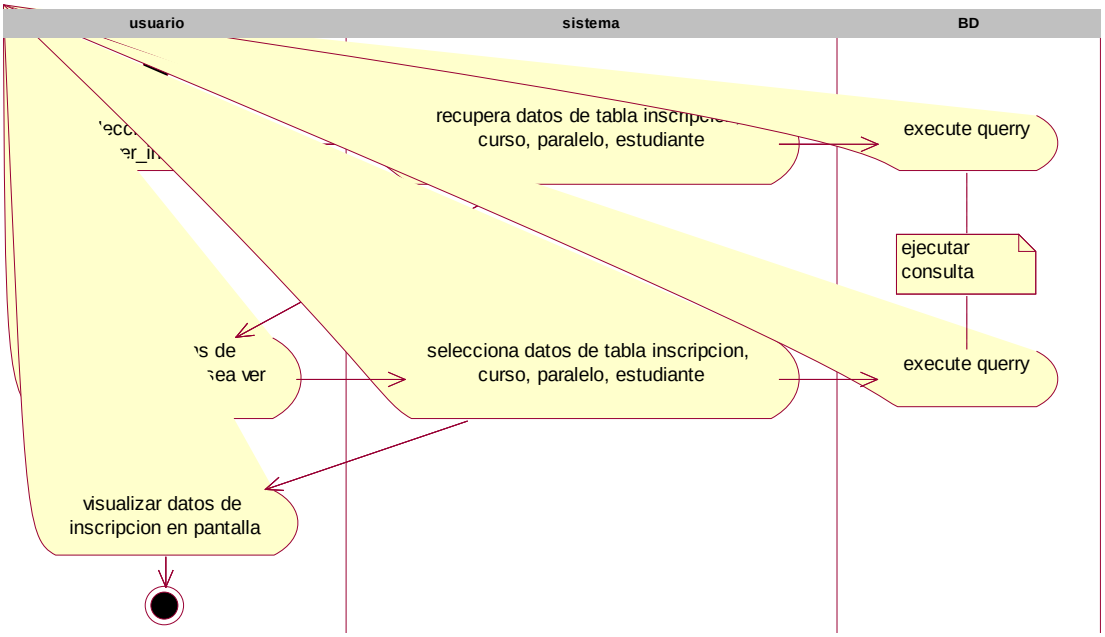


Figura 54. D.A. VER INSCRIP

II.1.8.1.2.11 MEMORANDUS

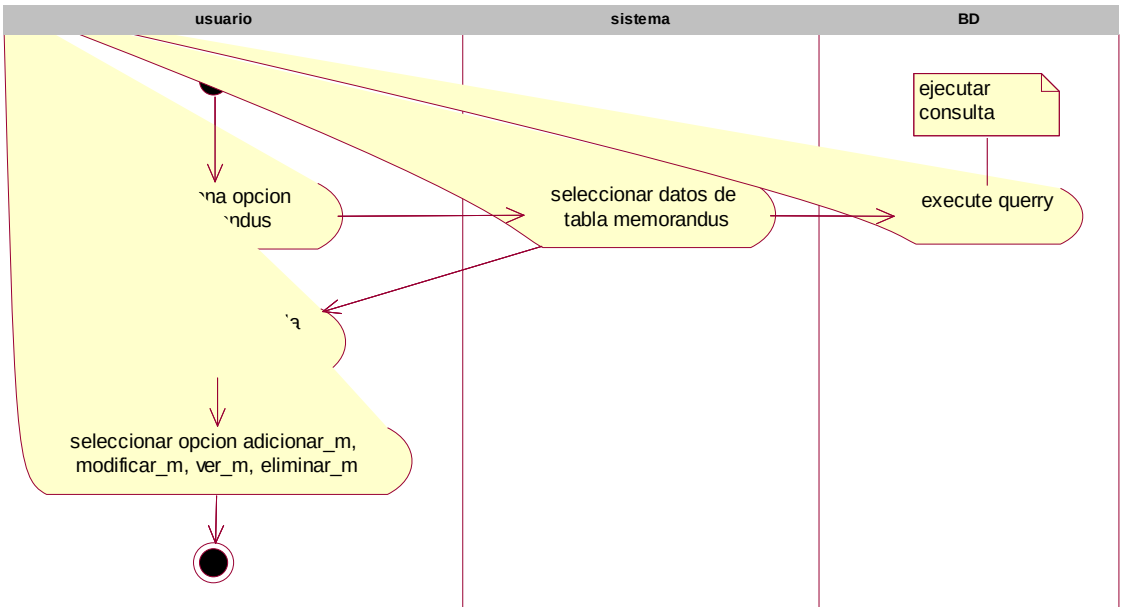


Figura 55. D.A. MEMORANDUS

II.1.8.1.2.12 ADICIONAR_M

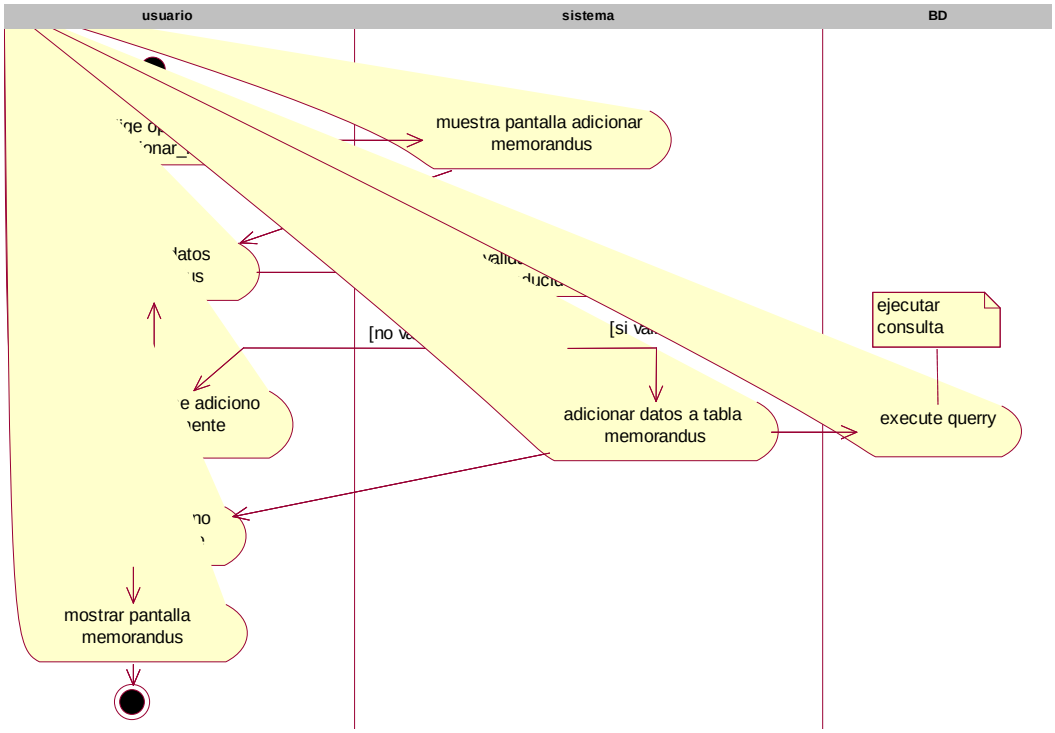


Figura 56. D.A. ADICIONAR_M

II.1.8.1.2.13 MODIFICAR_M

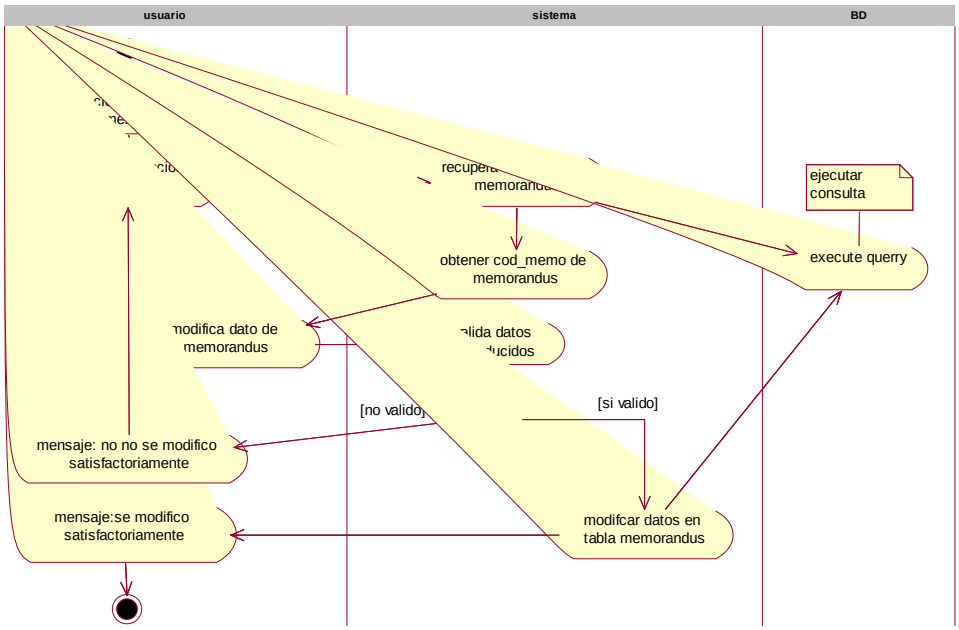


Figura 57. D.A. MODIFICAR_M

II.1.8.1.2.14 ELIMINAR_M

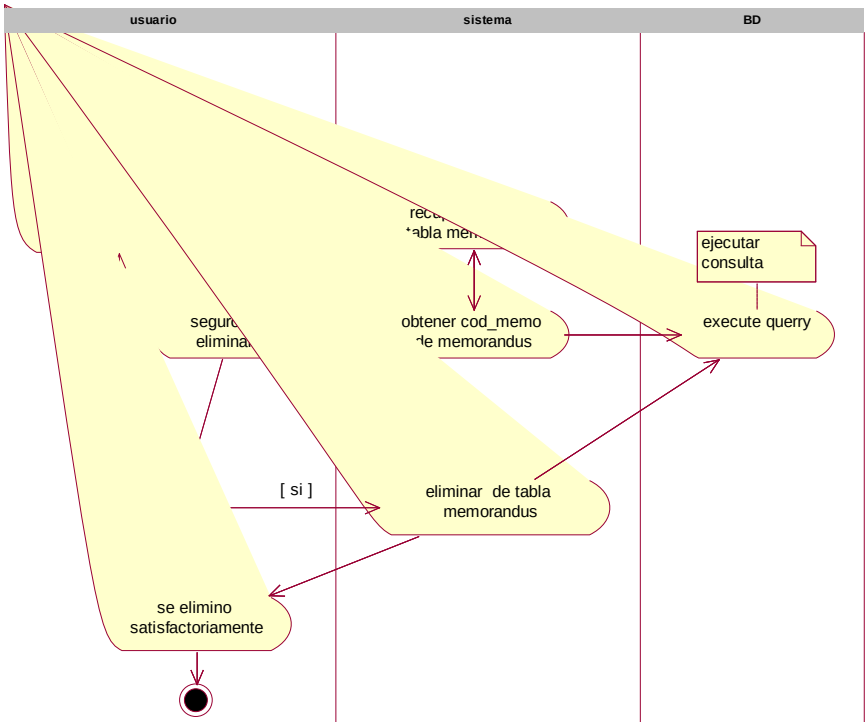


Figura 58. D.A ELIMINAR _M

II.1.8.1.2.15 VER_M

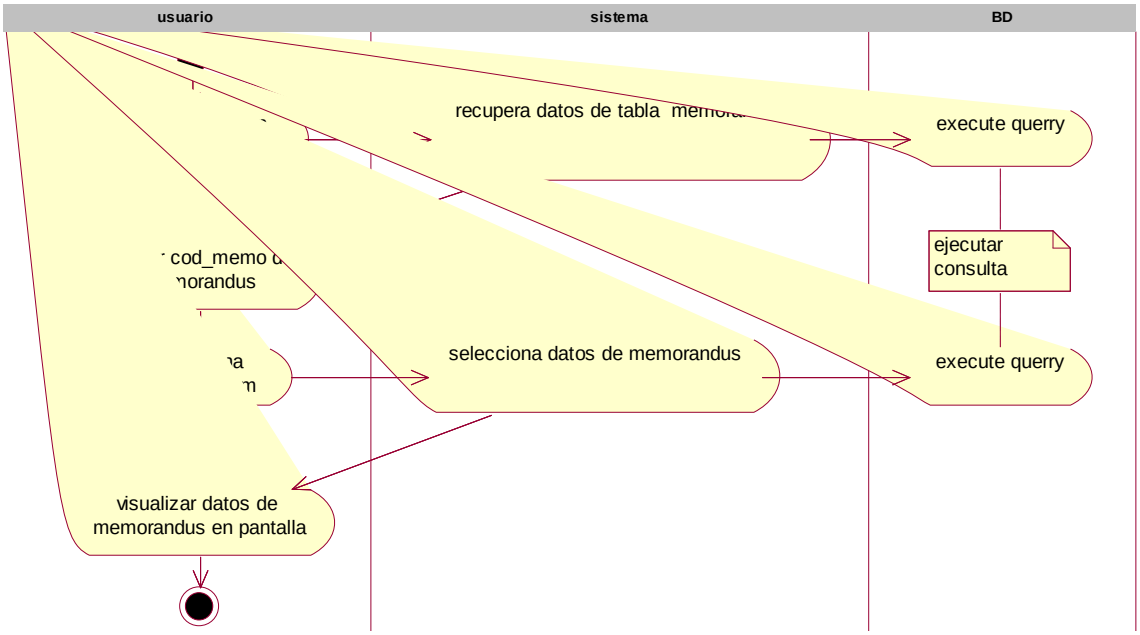


Figura 59. D.A. VER_M

II.1.8.1.2.16 REG_ADMINISTRATIVO

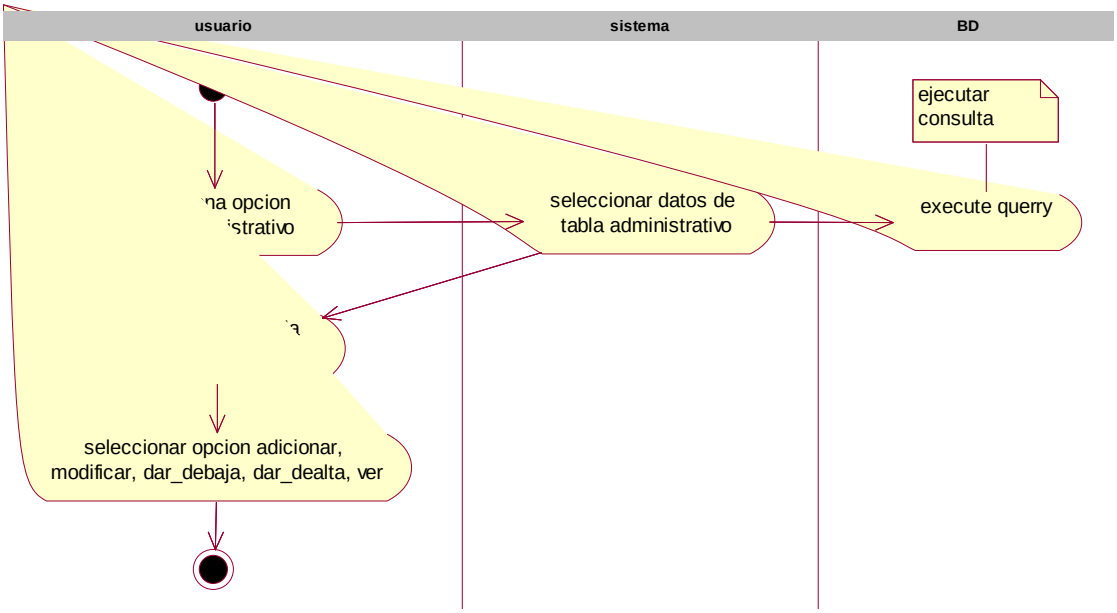


Figura 60. D.A. REG_ADMINISTRATIVO

II.1.8.1.2.17 ADICIONAR

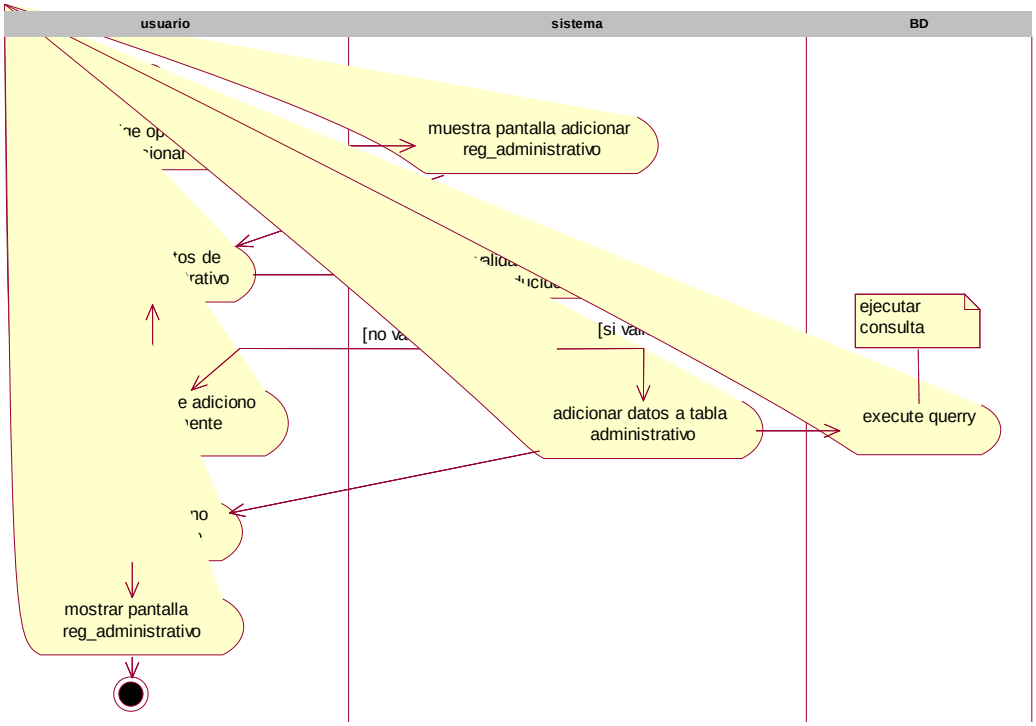


Figura 61. D.A. ADICIONAR

II.1.8.1.2.18 MODIFICAR

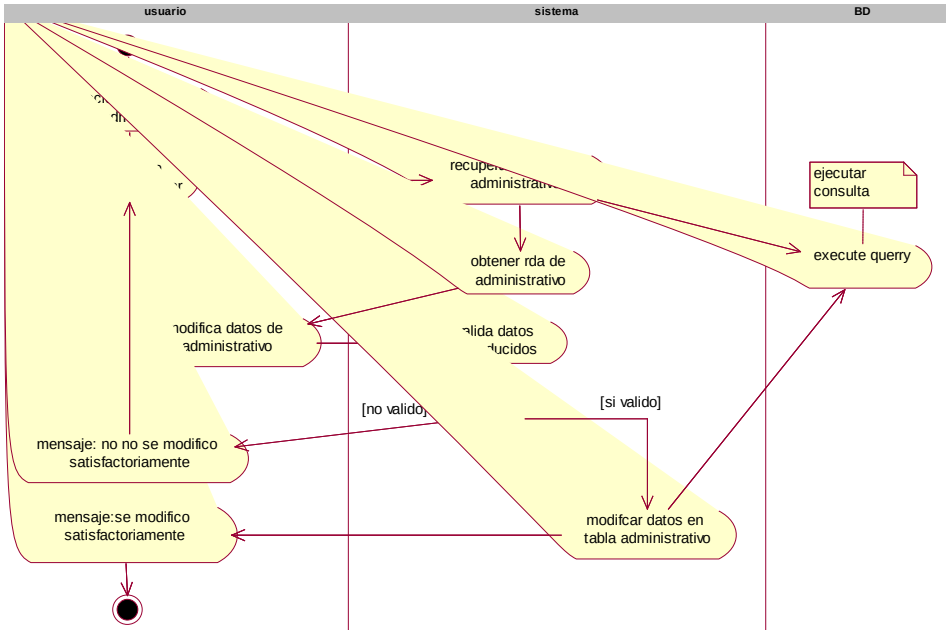


Figura 62. D.A. MODIFICAR

II.1.8.1.2.19 DAR DE BAJA

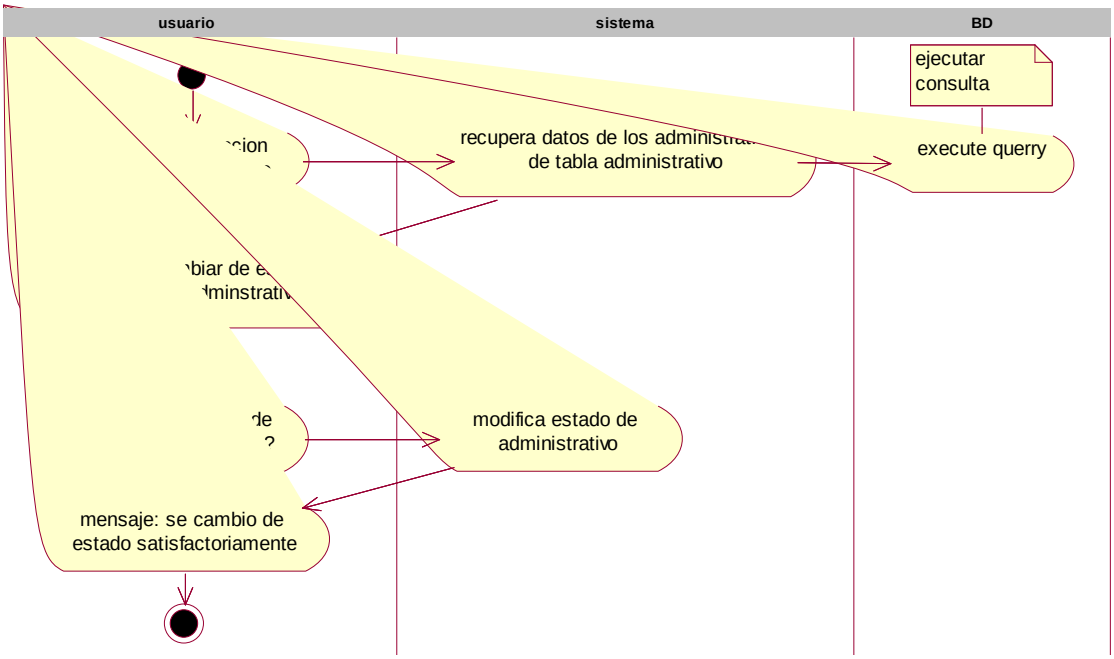


Figura 63. D.A. DAR DE BAJA

II.1.8.1.2.20 DAR DE ALTA

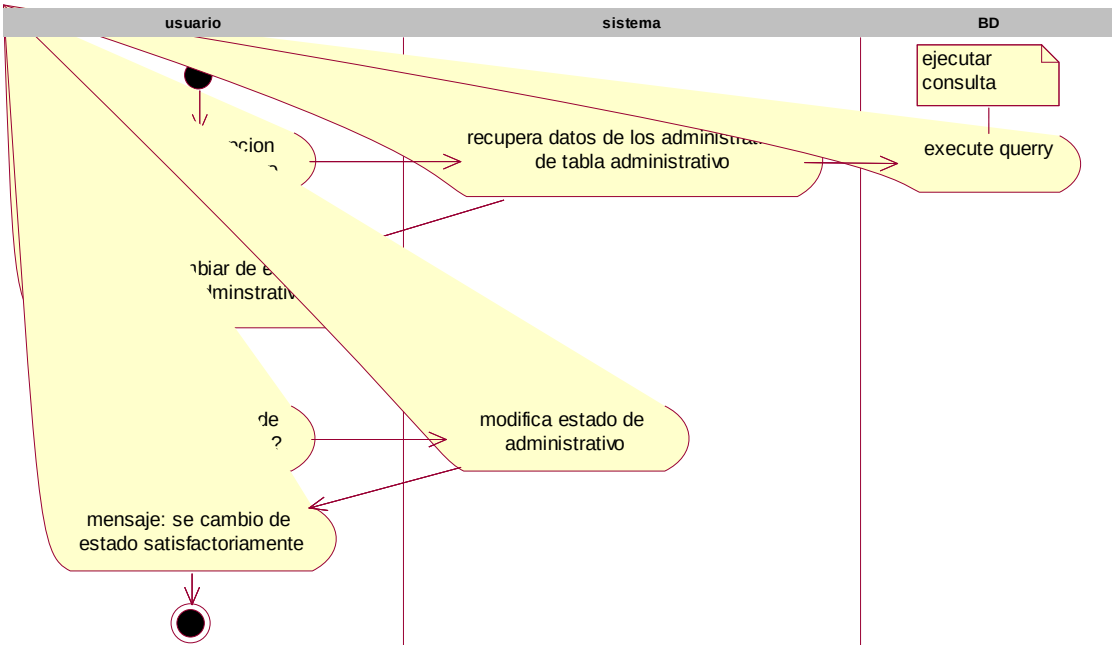


Figura 64. D.A. DAR DE ALTA

II.1.8.1.2.21 VER

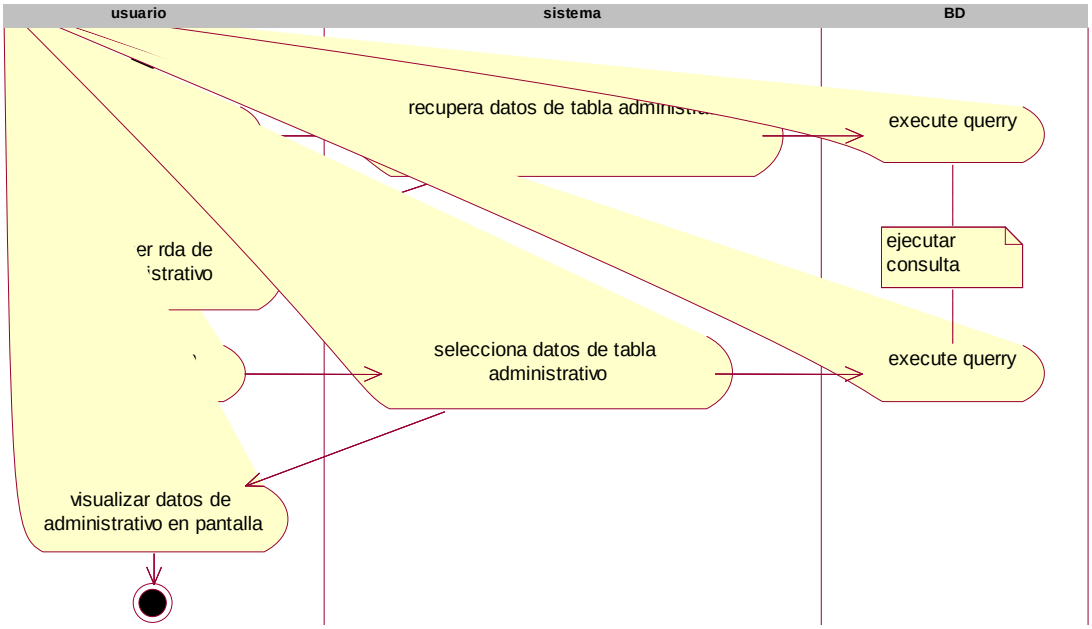


Figura 65. D.A. VER

II.1.8.1.2.22 REGISTRO

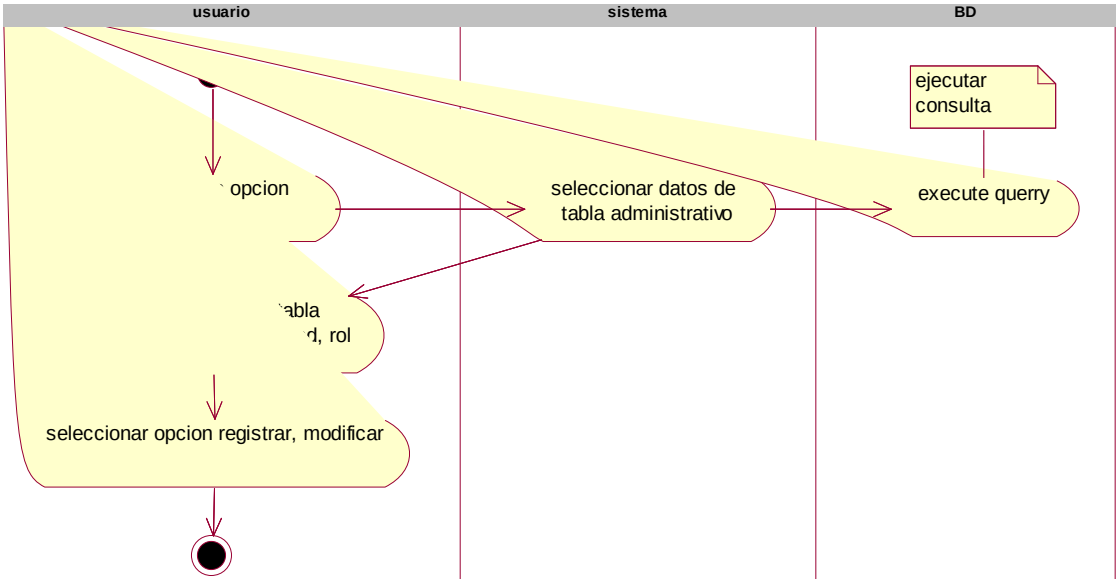


Figura 66. D.A. REGISTRO

II.1.8.1.2.23 REGISTRAR

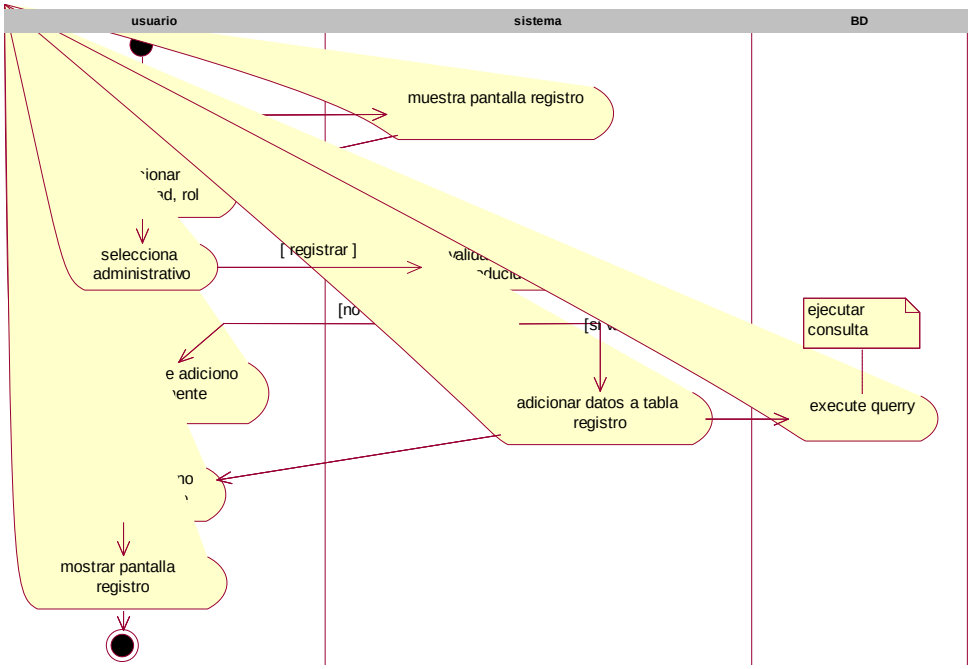


Figura 67. D.A. REGISTRAR

II.1.8.1.2.24 MODIFICAR

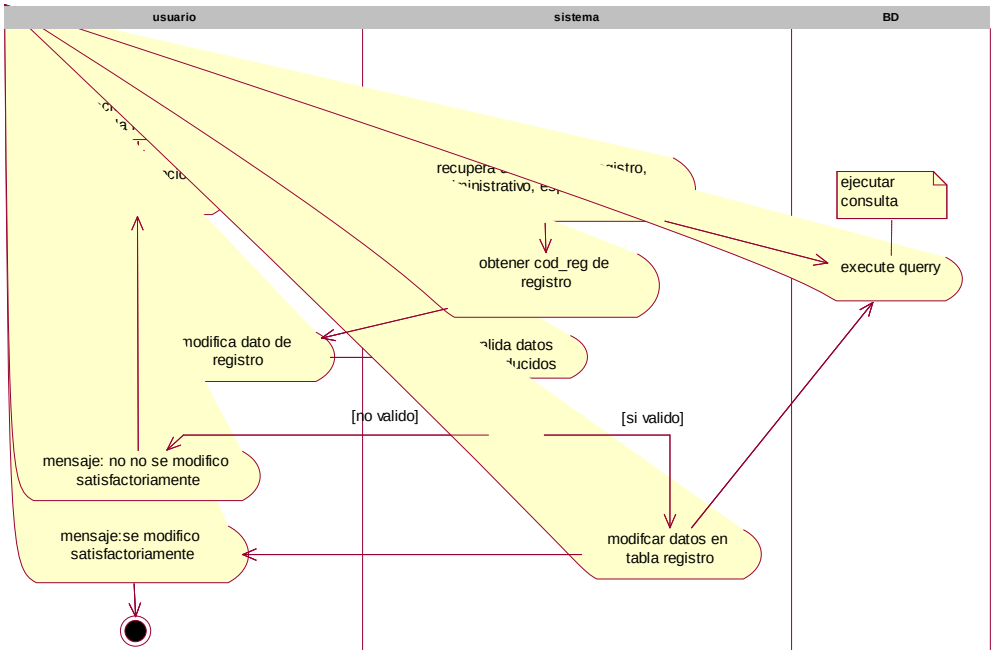


Figura 68. D.A. MODIFICAR

II.1.8.1.2.25 REG_MATERIAS

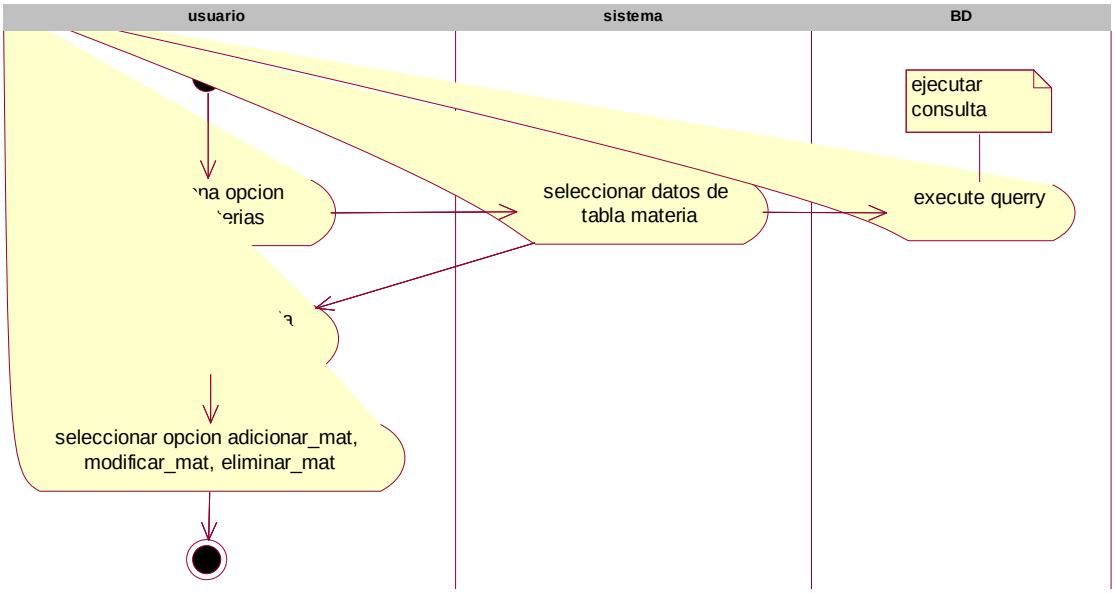


Figura 69. D.A. REG_MATERIAS

II.1.8.1.2.26 ADICIONAR_MAT

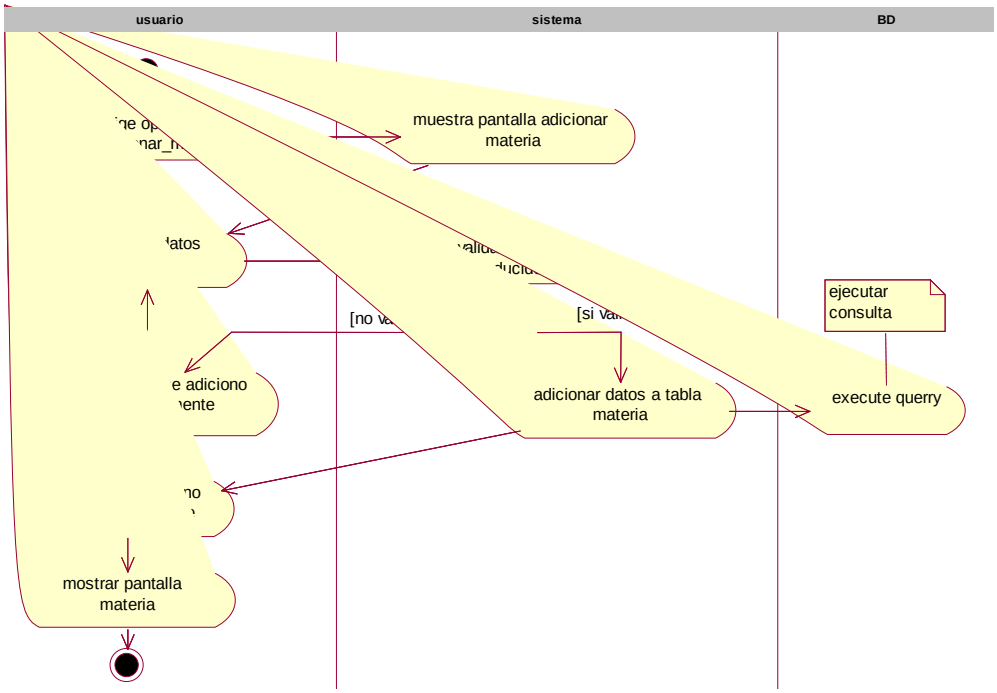


Figura 70. D.A. ADICIONAR_MAT

II.1.8.1.2.27 MODIFICAR_MAT

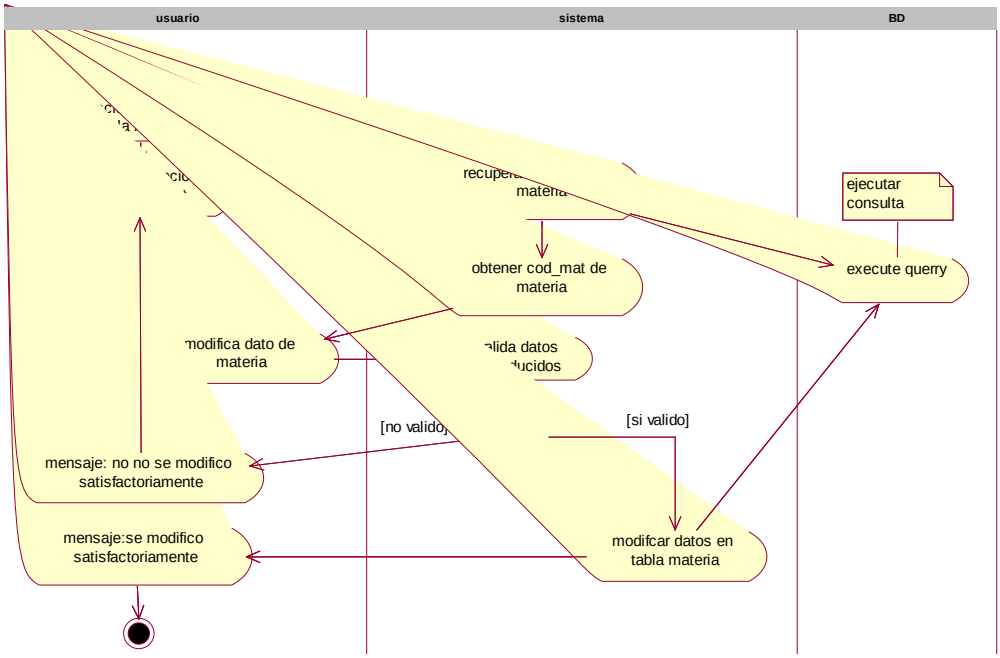


Figura 71. D.A. MODIFICAR_MAT

II.1.8.1.2.28 ELIMINAR_ MAT

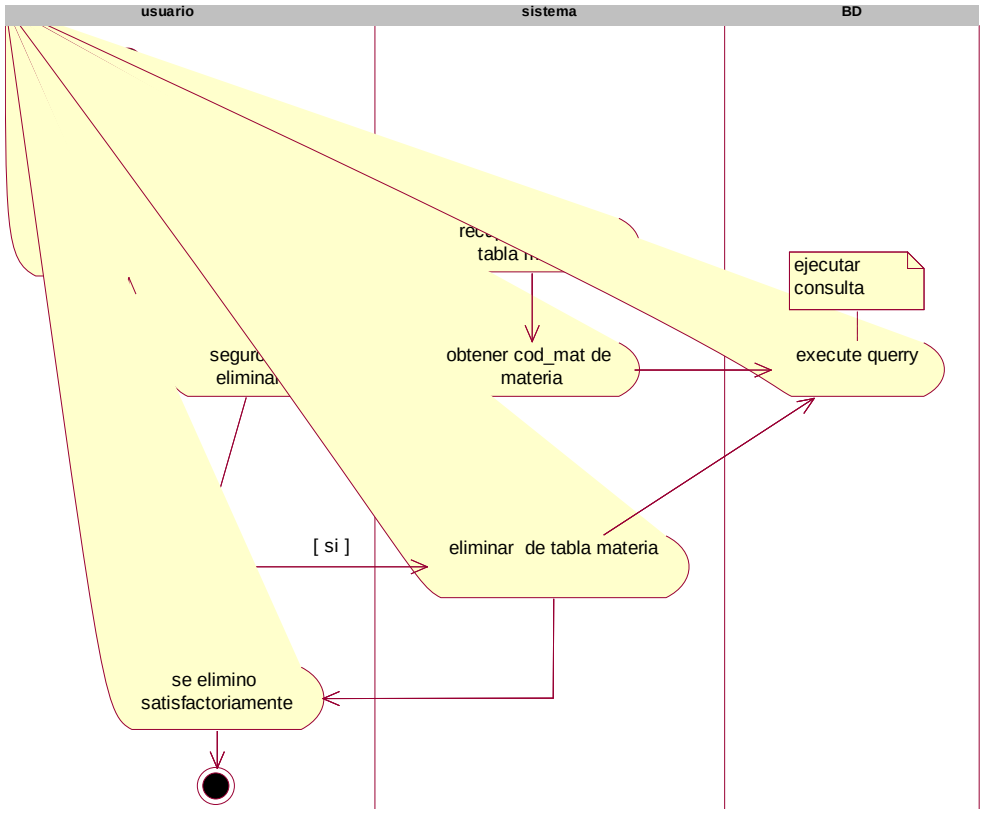


Figura 72. D.A. ELIMINAR_ MAT

II.1.8.1.2.29 ADMINISTRAR HORARIO

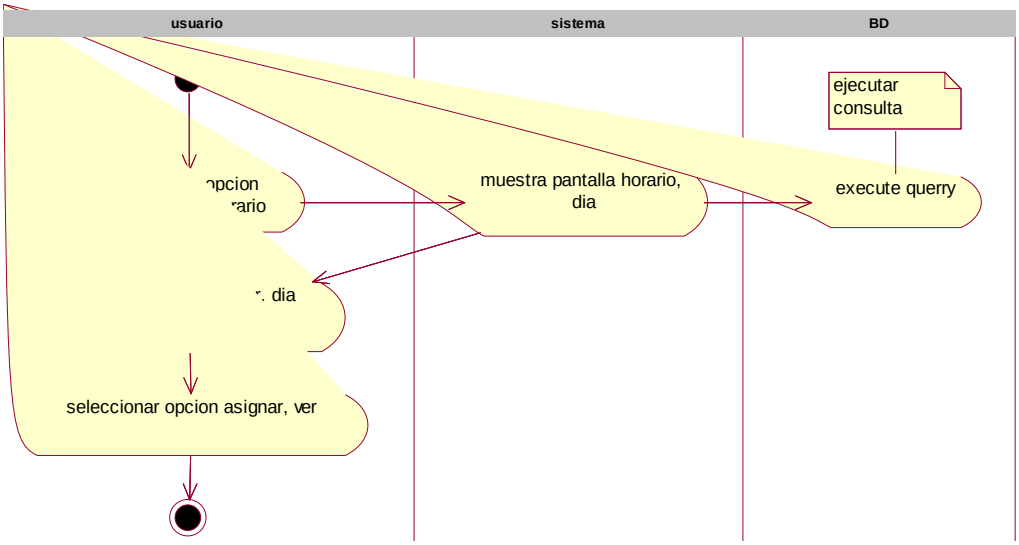


Figura 73. D.A. ADMINISTRAR HORARIO

II.1.8.1.2.30 ASIGNAR

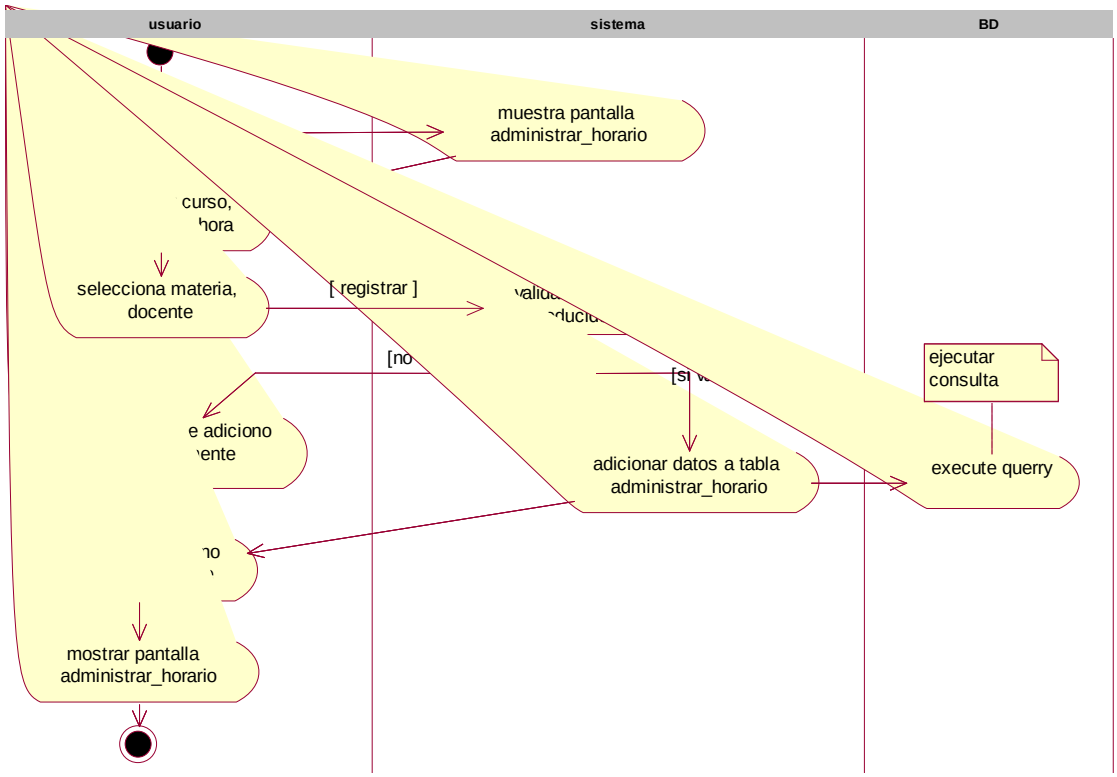


Figura 74. D.A. ASIGNAR

II.1.8.1.2.31 MODIFICAR ASIGNACION

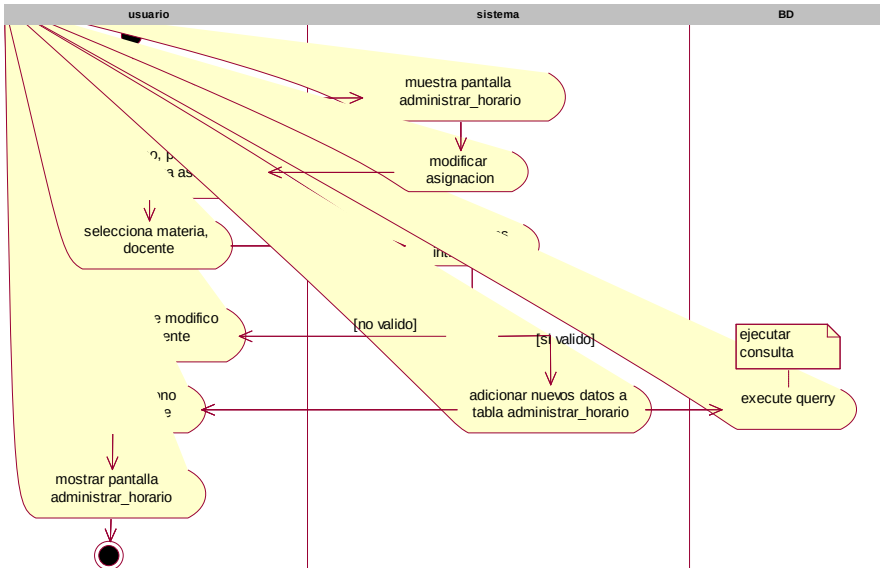


Figura 75. D.A. MODIFICAR ASIGNACION

II.1.8.1.2.32 VER

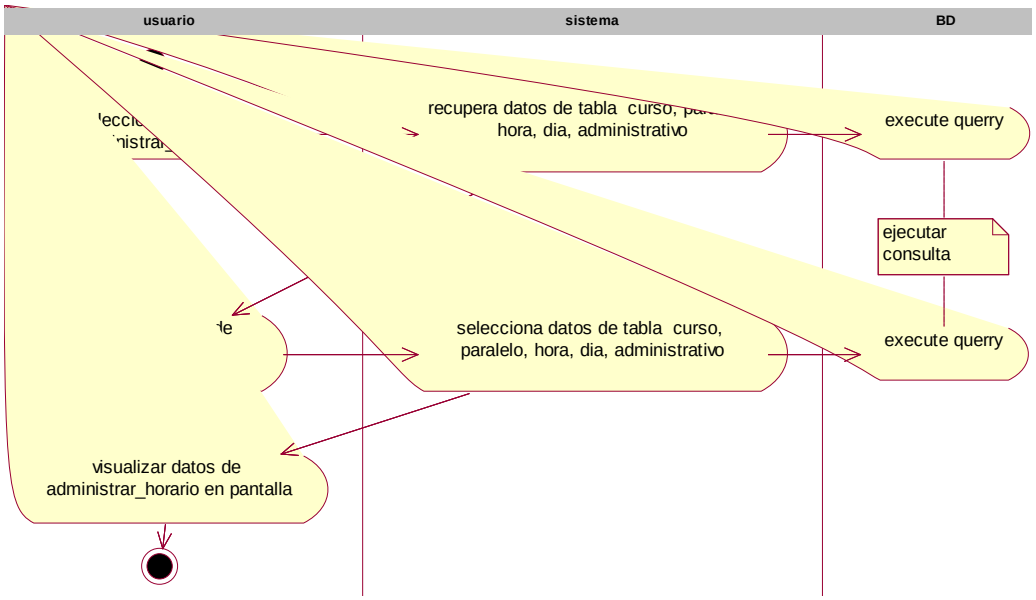


Figura 76. D.A. VER

II.1.8.1.2.33 REPORTES CREAR_REP

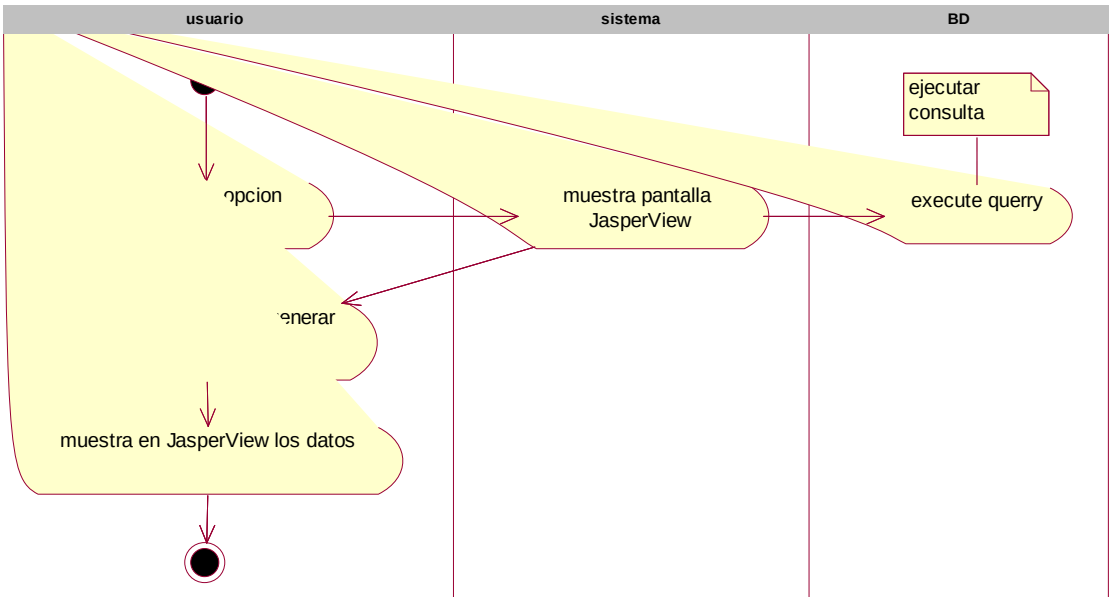


Figura 77. D.A. REPORTES CREAR_REP

II.1.8.2 Modelado de Diagramas de Secuencia

II.1.8.2.1 Introducción

El diagrama de secuencia consiste en un conjunto de objetos y sus relaciones.

II.1.8.2.1.1 Propósito

- Comprender la dinámica, sistema deseado para la organización
- Identificar las clases de análisis y diseño

II.1.8.2.1.2 Alcance

- Describir la dinámica del sistema en el tiempo de vida de las clases u objetos
- Definir un diagrama de secuencia para cada caso de uso.

II.1.8.2.2 Diagramas de Secuencia

II.1.8.2.2.1 INICIAR

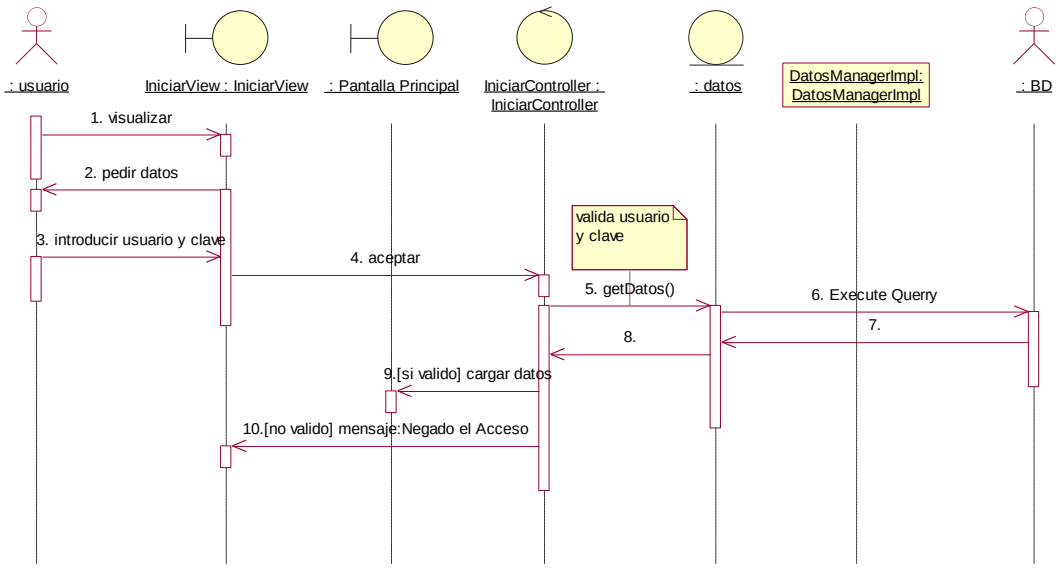


Figura 78. D.S. INICIAR

II.1.8.2.2.2 Reg_estudiante

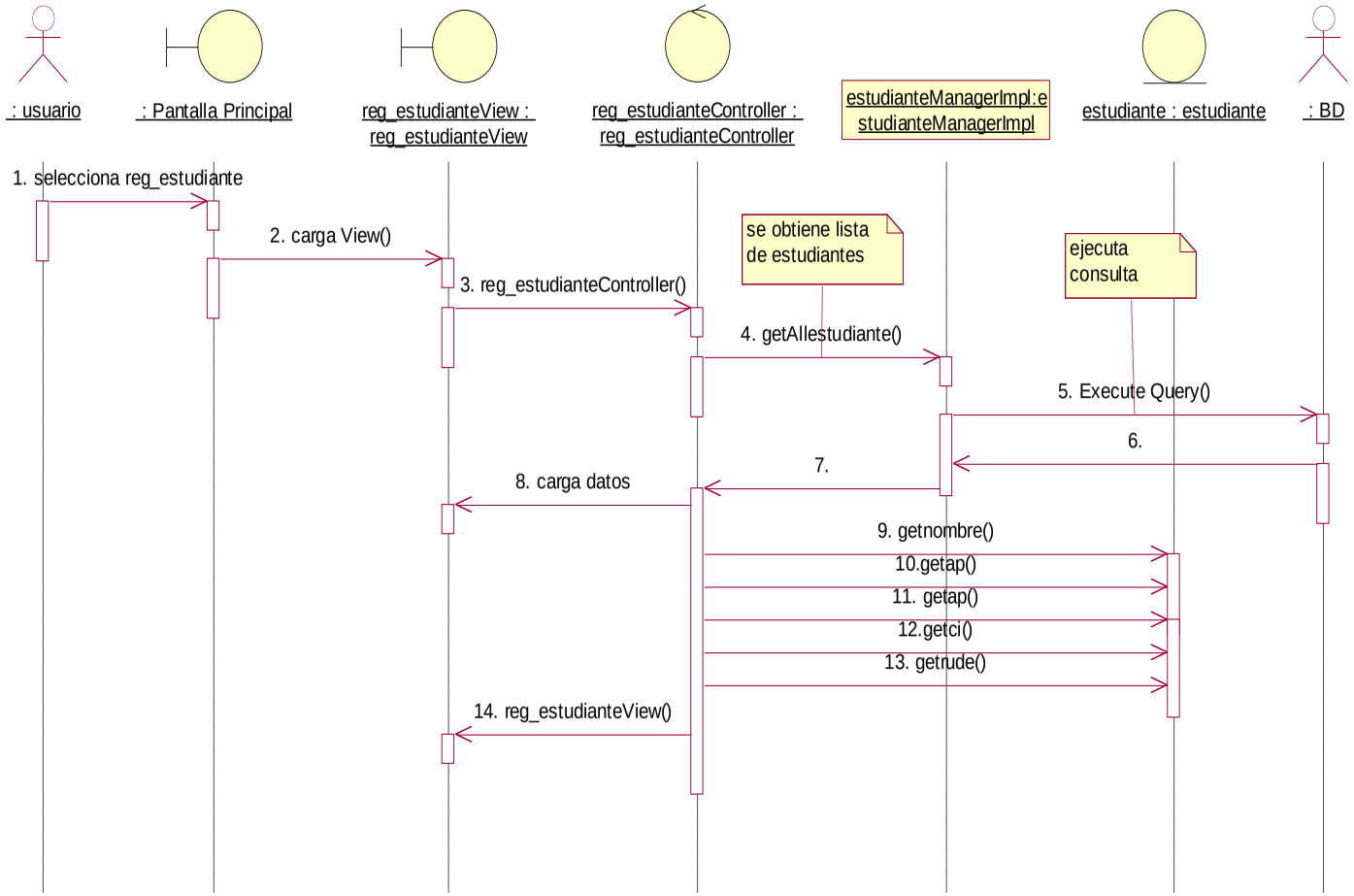

: usuario
: Pantalla Principal
reg_estudianteView : reg_estudianteView
reg_estudianteController : reg_estudianteController
estudianteManagerImpl:estudianteManagerImpl
estudiante : estudiante
: BD
1. selecciona reg_estudiante
2. carga View()
3. reg_estudianteController()
se obtiene lista de estudiantes
4. getAllestudiante()
ejecuta consulta
5. Execute Query()
6.
7.
8. carga datos
9. getnombre()
10.getap()
11. getap()
12.getci()
13. getrude()
14. reg_estudianteView()

Figura 79. D.S. Reg_estudiante

II.1.8.2.2.3 adicionar

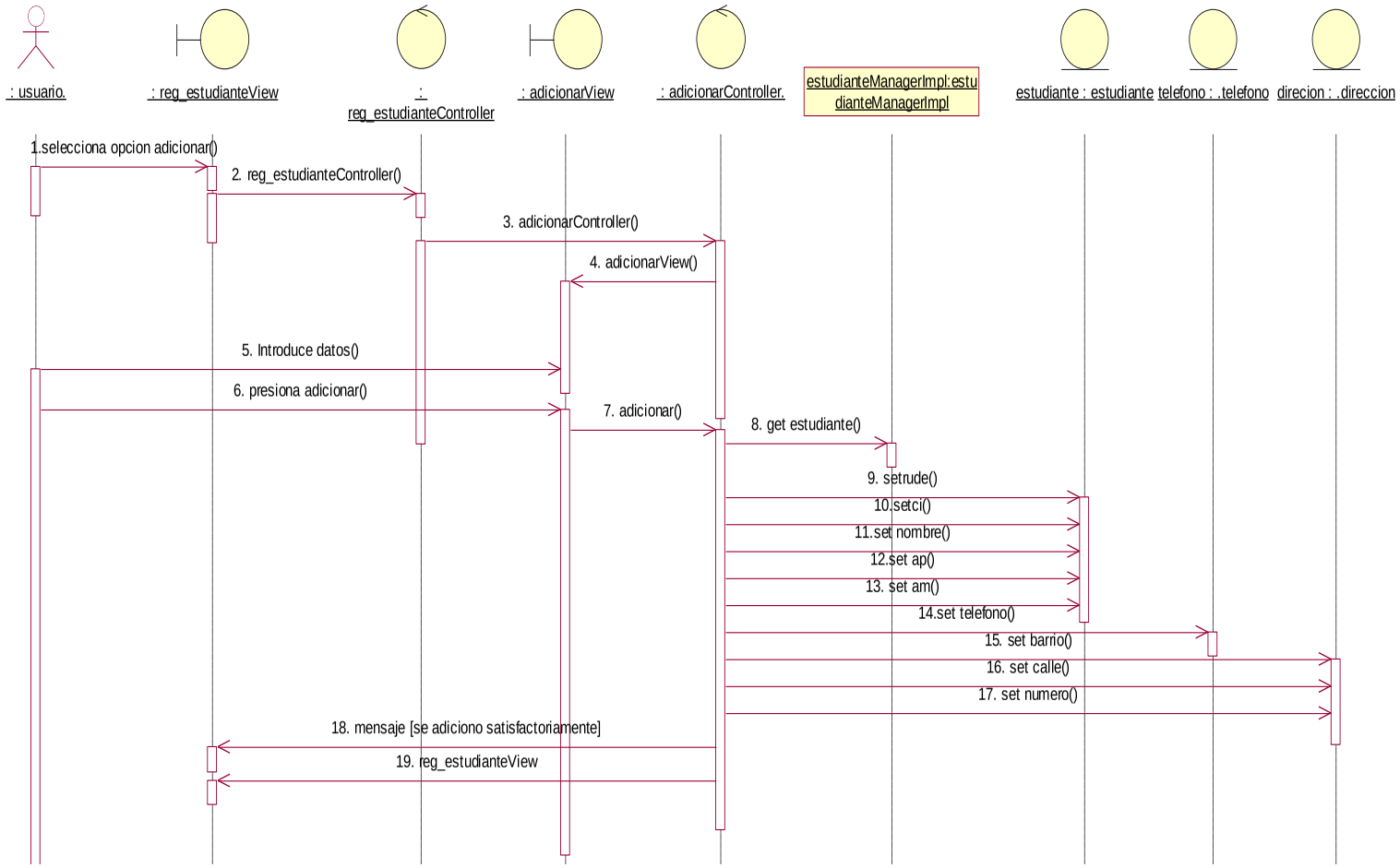

: usuario.
: reg_estudianteView
: reg_estudianteController
: adicionarView
: adicionarController.
estudianteManagerImpl:estudianteManagerImpl
estudiante : estudiante
telefono : .telefono
direcion : .direccion
1.selecciona opcion adicionar()
2. reg_estudianteController()
3. adicionarController()
4. adicionarView()
5. Introduce datos()
6. presiona adicionar()
7. adicionar()
8. get estudiante()
9. setrude()
10.setci()
11.set nombre()
12.set ap()
13. set am()
14.set telefono()
15. set barrio()
16. set calle()
17. set numero()
18. mensaje [se adiciono satisfactoriamente]
19. reg_estudianteView

Figura 80. D.S. Adicionar

II.1.8.2.2.4 Modificar

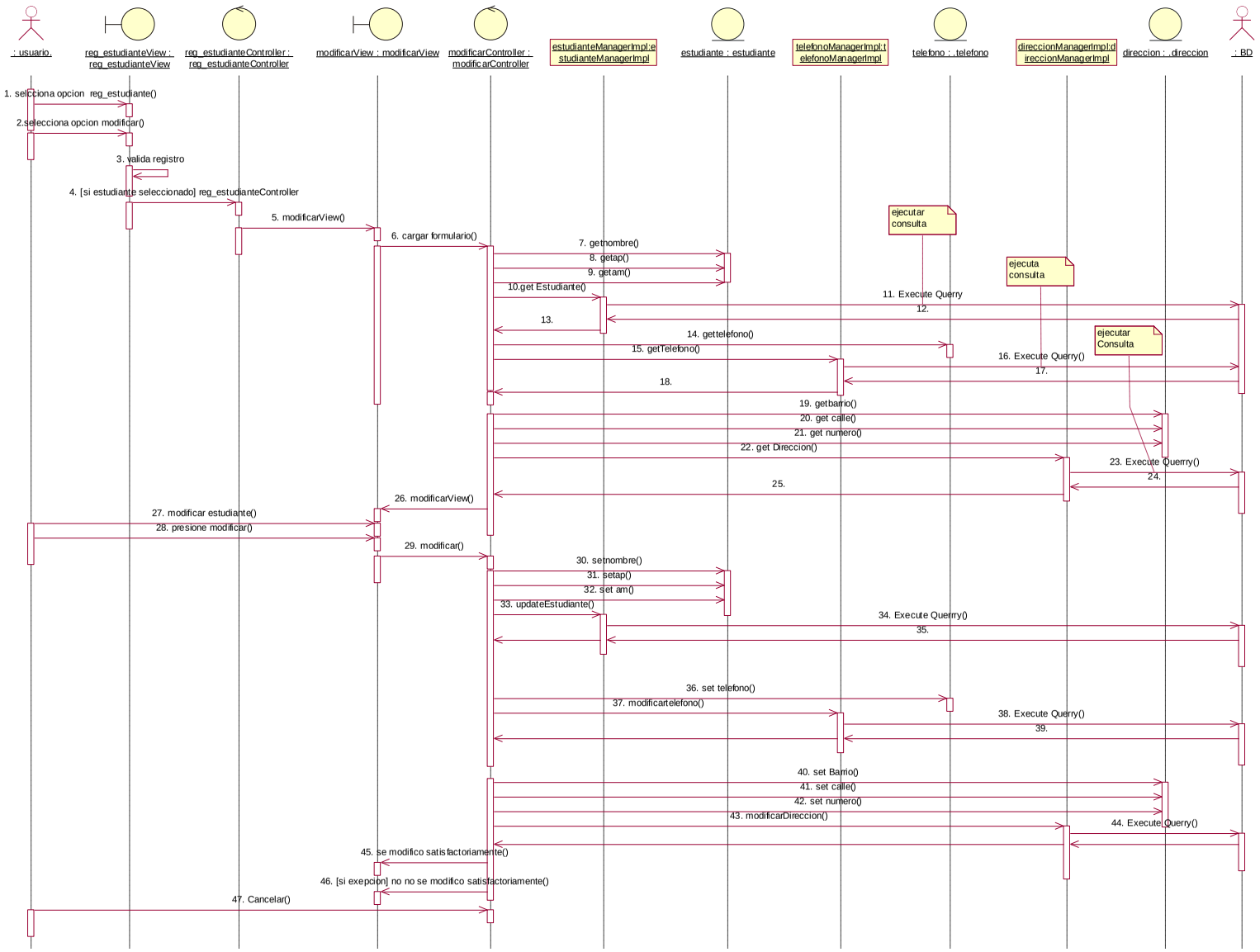

: usuario.
reg_estudianteView : reg_estudianteView
reg_estudianteController : reg_estudianteController
modificarView : modificarView
modificarController : modificarController
estudianteManagerImpl:estudianteManagerImpl
estudiante : estudiante
telefonoManagerImpl:telefonoManagerImpl
telefono : .telefono
direccionManagerImpl:direccionManagerImpl
direccion : .direccion
: BD
1. selecciona opcion reg_estudiante()
2.selecciona opcion modificar()
3. valida registro
4. [si estudiante seleccionado] reg_estudianteController
5. modificarView()
6. cargar formulario()
7. getnombre()
8. getap()
9. getam()
10.get Estudiante()
11. Execute Querry
12.
13.
14. gettelefono()
15. getTelefono()
16. Execute Querry()
17.
18.
19. getbarrio()
20. get calle()
21. get numero()
22. get Direccion()
23. Execute Querrry()
24.
25.
26. modificarView()
27. modificar estudiante()
28. presione modificar()
29. modificar()
30. setnombre()
31. setap()
32. set am()
33. updateEstudiante()
34. Execute Querrry()
35.
36. set telefono()
37. modificartelefono()
38. Execute Querry()
39.
40. set Barrio()
41. set calle()
42. set numero()
43. modificarDireccion()
44. Execute Querry()
45. se modifico satisfactoriamente()
46. [si exepcion] no no se modifico satisfactoriamente()
47. Cancelar()
ejecutar consulta
ejecuta consulta
ejecutar Consulta

Figura 81. D.S. Modificar

II.1.8.2.2.5 dar de baja

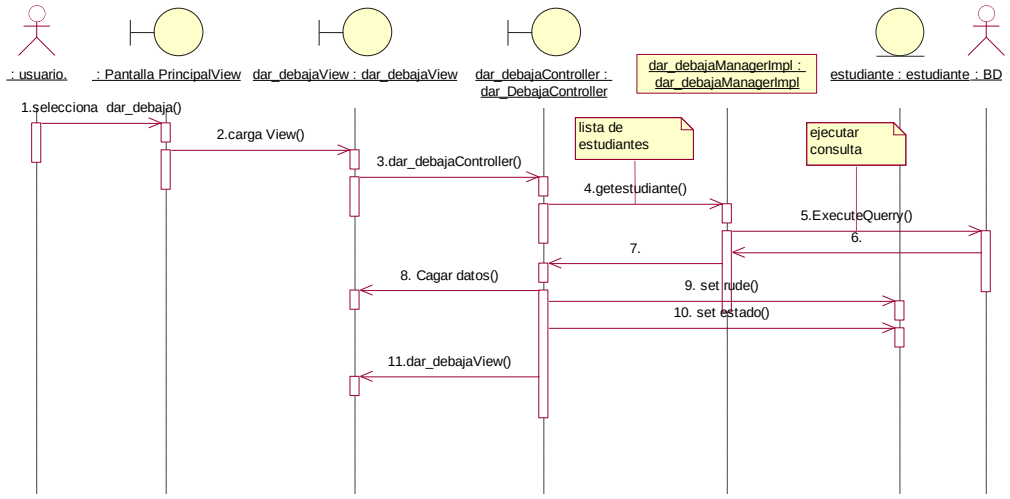


Figura 82. D.S. Dar de baja

II.1.8.2.2.6 dar de alta

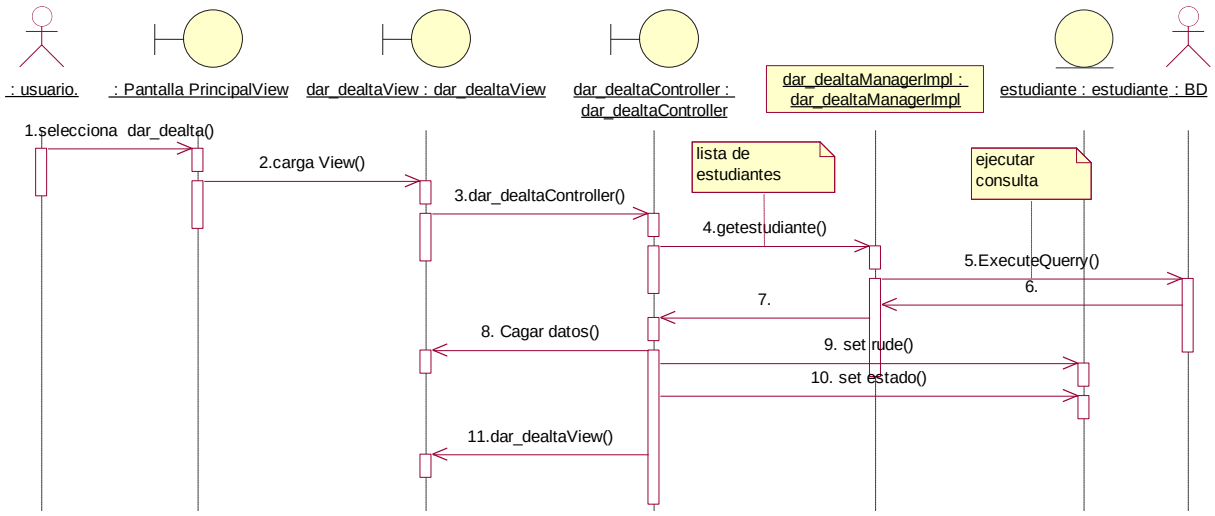


Figura 83. D.S. dar de alta

II.1.8.2.2.7 Ver

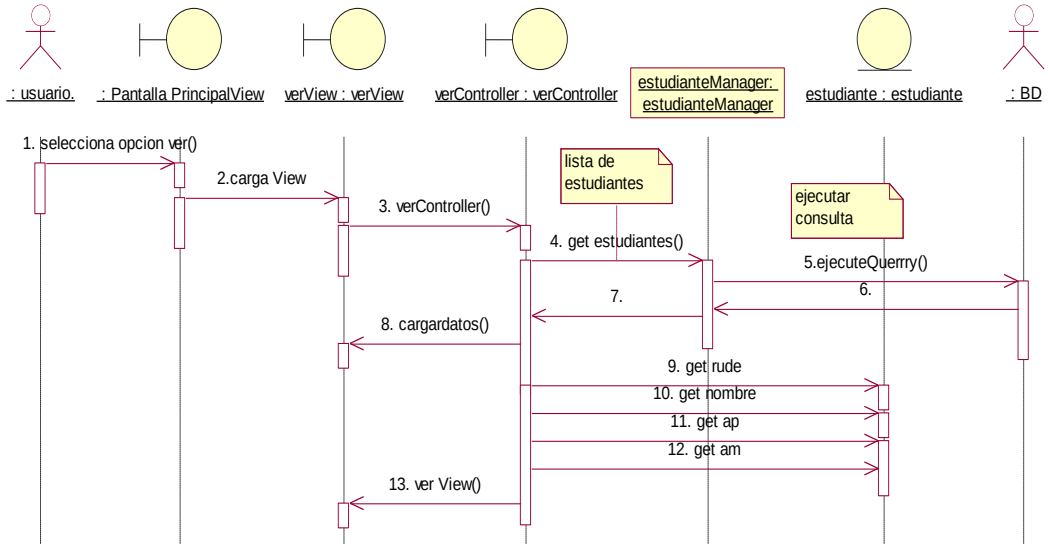


Figura 84. D.S. Ver

II.1.8.2.2.8 MEMORANDUS

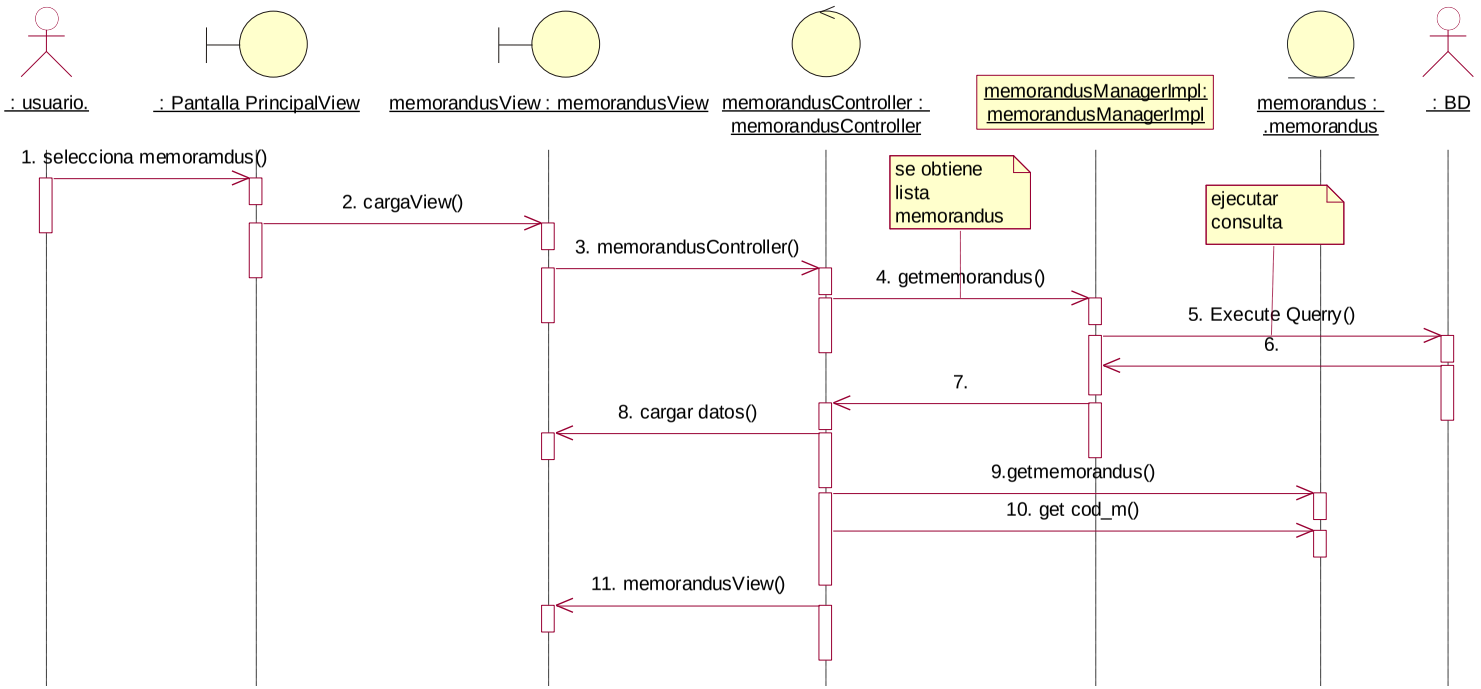

: usuario.
: Pantalla PrincipalView
memorandusView : memorandusView
memorandusController : memorandusController
memorandusManagerImpl: memorandusManagerImpl
memorandus : .memorandus
: BD
1. selecciona memoramdus()
2. cargaView()
3. memorandusController()
se obtiene lista memorandus
4. getmemorandus()
ejecutar consulta
5. Execute Querry()
6.
7.
8. cargar datos()
9.getmemorandus()
10. get cod_m()
11. memorandusView()

Figura 85. D.S. MEMORANDUS

II.1.8.2.2.9 adicionar_m

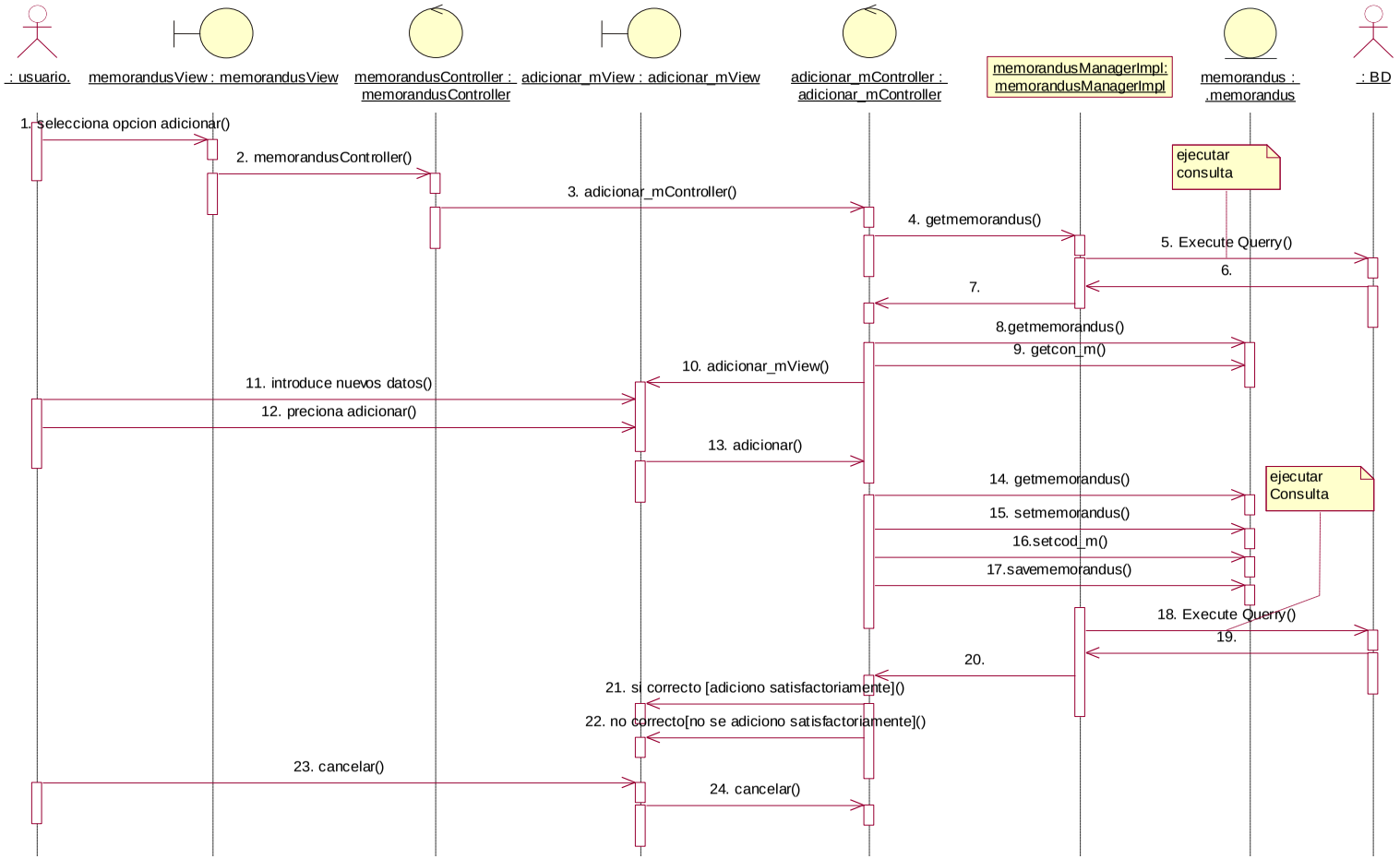

: usuario.
memorandusView : memorandusView
memorandusController : memorandusController
adicionar_mView : adicionar_mView
adicionar_mController : adicionar_mController
memorandusManagerImpl: memorandusManagerImpl
memorandus : .memorandus
: BD
1. selecciona opcion adicionar()
2. memorandusController()
3. adicionar_mController()
4. getmemorandus()
ejecutar consulta
5. Execute Querry()
6.
7.
8.getmemorandus()
9. getcon_m()
10. adicionar_mView()
11. introduce nuevos datos()
12. preciona adicionar()
13. adicionar()
14. getmemorandus()
ejecutar Consulta
15. setmemorandus()
16.setcod_m()
17.savememorandus()
18. Execute Querry()
19.
20.
21. si correcto [adiciono satisfactoriamente]()
22. no correcto[no se adiciono satisfactoriamente]()
23. cancelar()
24. cancelar()

Figura 86. D.S. adicionar_m

II.1.8.2.2.10 modificar_m

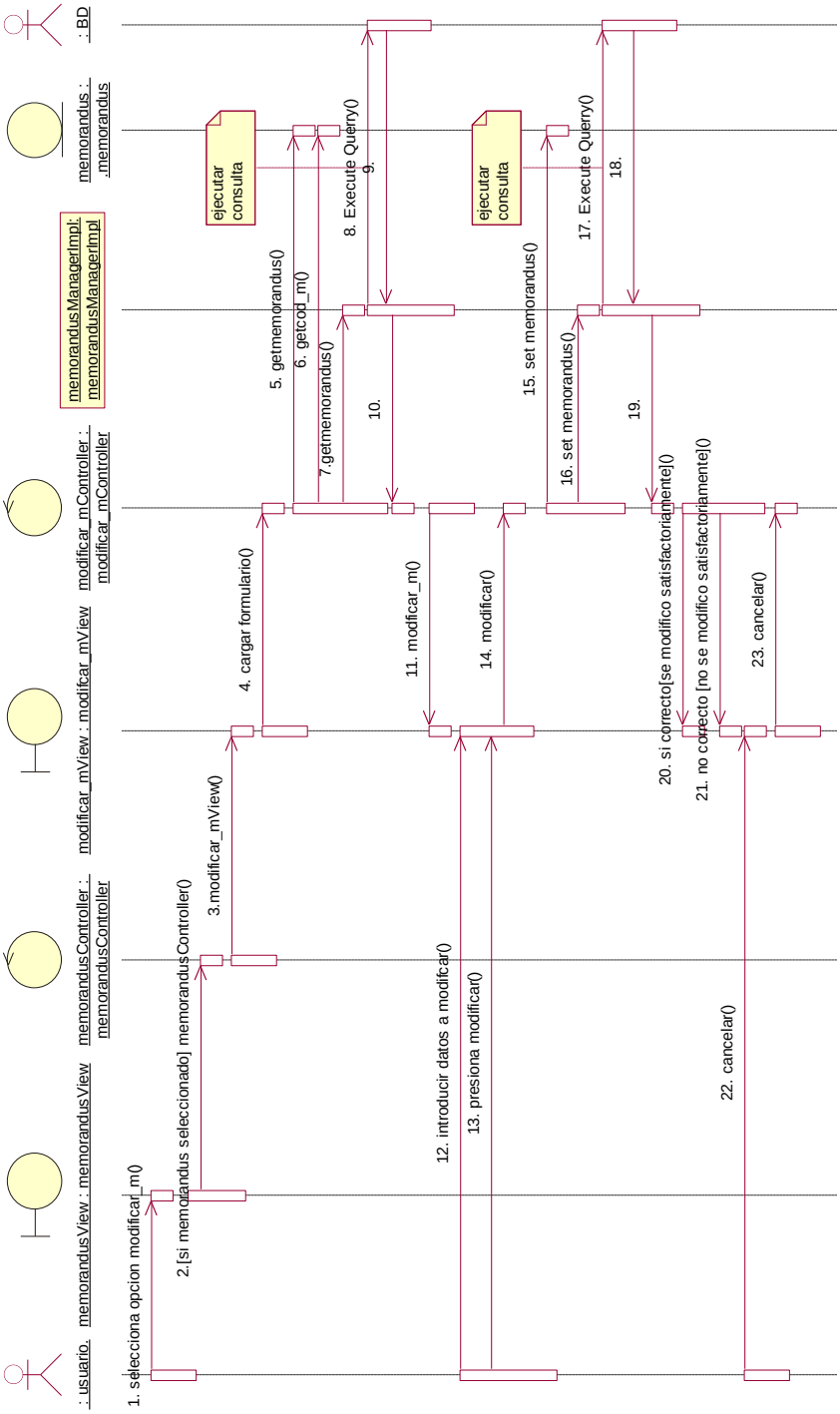


Figura 87. D.S. modificar_m

II.1.8.2.2.11 eliminar_m

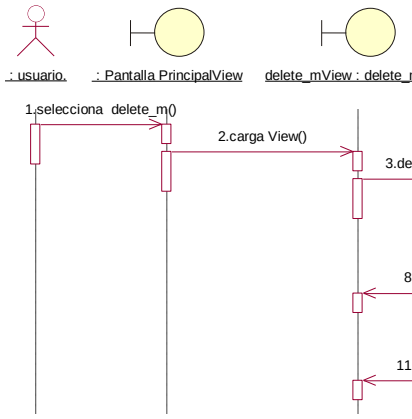


Figura 88. D.S. eliminar_m

II.1.8.2.2.12 ver_m

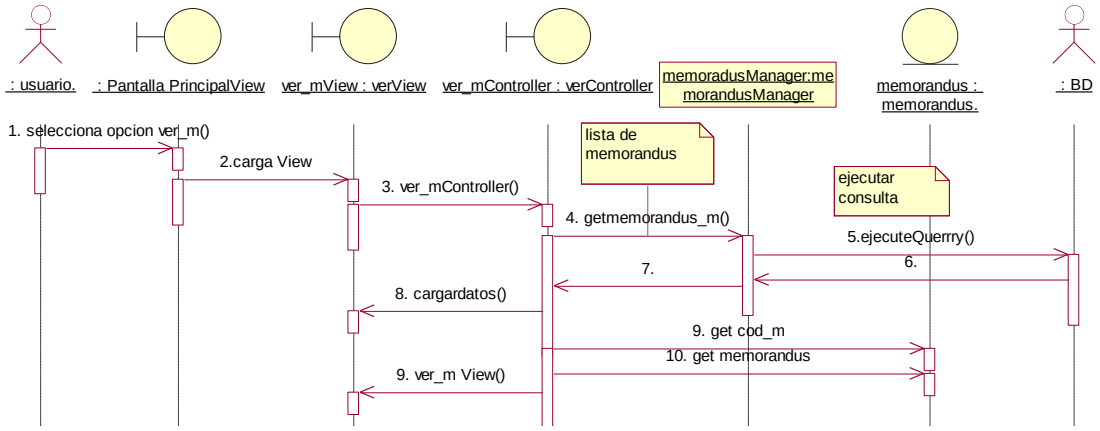


Figura 89. D.S. ver_m

II.1.8.2.2.13 REG_ADMINISTRATIVO

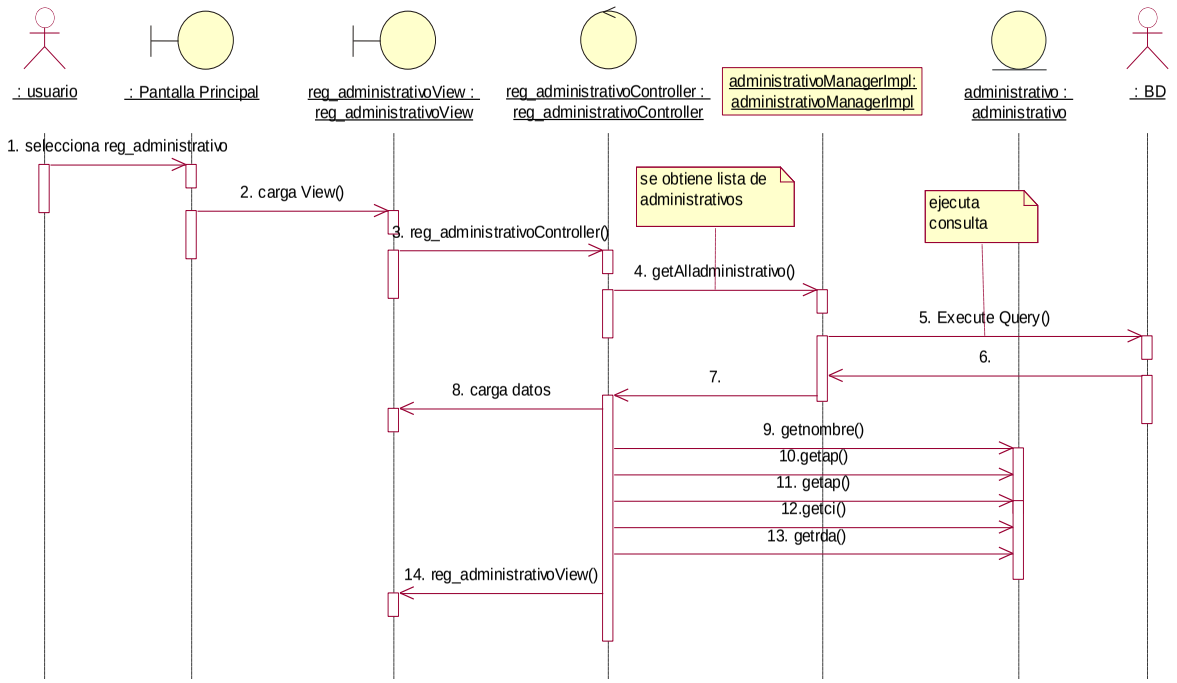

: usuario
: Pantalla Principal
reg_administrativoView : reg_administrativoView
reg_administrativoController : reg_administrativoController
administrativoManagerImpl: administrativoManagerImpl
administrativo : administrativo
: BD
1. selecciona reg_administrativo
2. carga View()
3. reg_administrativoController()
se obtiene lista de administrativos
4. getAlladministrativo()
ejecuta consulta
5. Execute Query()
6.
7.
8. carga datos
9. getnombre()
10.getap()
11. getap()
12.getci()
13. getrda()
14. reg_administrativoView()

Figura 90. D.S. REG_ADMINISTRATIVO

II.1.8.2.2.14 Adicionar

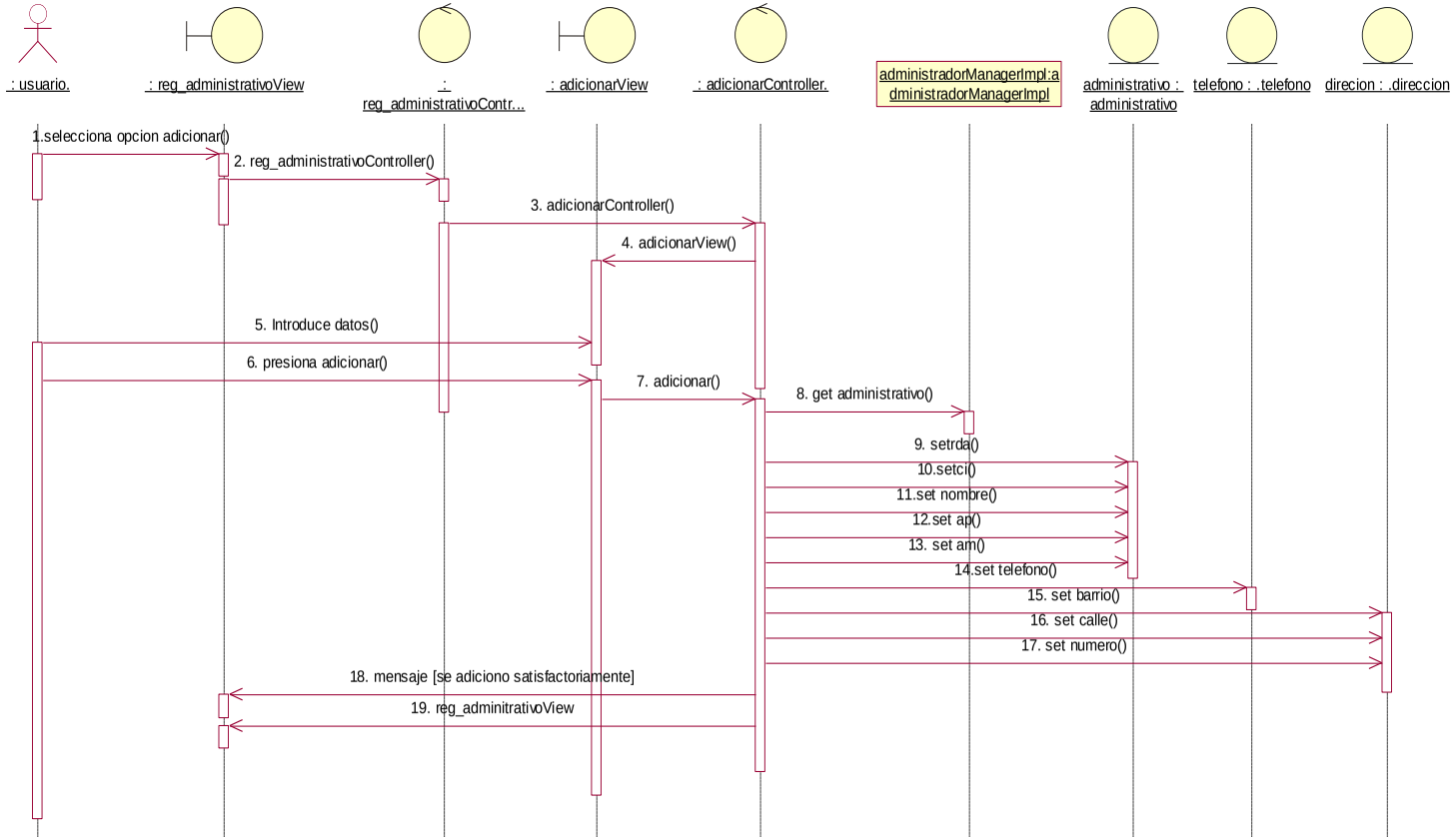

: usuario.
: reg_administrativoView
: reg_administrativoContr...
: adicionarView
: adicionarController.
administradorManagerImpl:administradorManagerImpl
administrativo : administrativo
telefono : .telefono
direcion : .direccion
1.selecciona opcion adicionar()
2. reg_administrativoController()
3. adicionarController()
4. adicionarView()
5. Introduce datos()
6. presiona adicionar()
7. adicionar()
8. get administrativo()
9. setrda()
10.setci()
11.set nombre()
12.set ap()
13. set am()
14.set telefono()
15. set barrio()
16. set calle()
17. set numero()
18. mensaje [se adiciono satisfactoriamente]
19. reg_adminitrativoView

Figura 91. D.S. Adicionar

II.1.8.2.2.15 Modificar

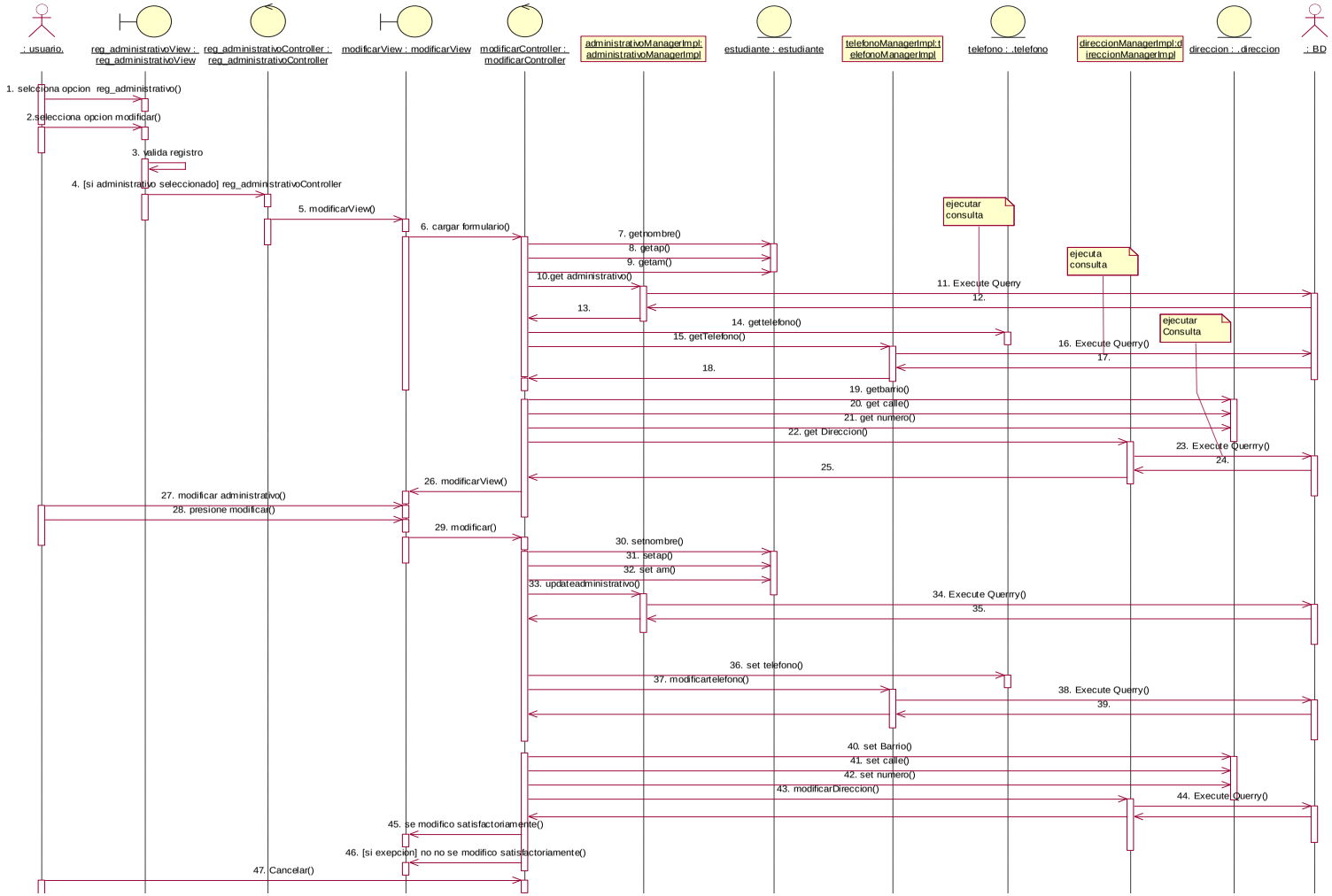

: usuario.
reg_administrativoView : reg_administrativoView
reg_administrativoController : reg_administrativoController
modificarView : modificarView
modificarController : modificarController
administrativoManagerImpl: administrativoManagerImpl
estudiante : estudiante
telefonoManagerImpl:telefonoManagerImpl
telefono : .telefono
direccionManagerImpl:direccionManagerImpl
direccion : .direccion
: BD
1. selecciona opcion reg_administrativo()
2.selecciona opcion modificar()
3. valida registro
4. [si administrativo seleccionado] reg_administrativoController
5. modificarView()
6. cargar formulario()
7. getnombre()
8. getap()
9. getam()
10.get administrativo()
11. Execute Querry
12.
13.
14. gettelefono()
15. getTelefono()
16. Execute Querry()
17.
18.
19. getbarrio()
20. get calle()
21. get numero()
22. get Direccion()
23. Execute Querrry()
24.
25.
26. modificarView()
27. modificar administrativo()
28. presione modificar()
29. modificar()
30. setnombre()
31. setap()
32. set am()
33. updateadministrativo()
34. Execute Querrry()
35.
36. set telefono()
37. modificartelefono()
38. Execute Querry()
39.
40. set Barrio()
41. set calle()
42. set numero()
43. modificarDireccion()
44. Execute Querry()
45. se modifico satisfactoriamente()
46. [si exepcion] no no se modifico satisfactoriamente()
47. Cancelar()
ejecutar consulta
ejecuta consulta
ejecutar Consulta

Figura 92. D.S. Modificar

II.1.8.2.2.16 dar_debaja

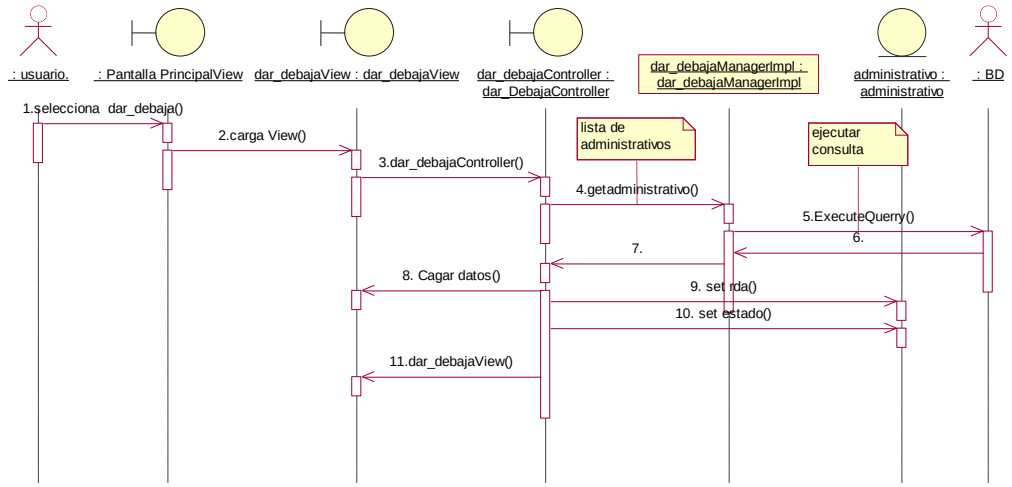


Figura 93. D.S. dar_debaja

II.1.8.2.2.17 dar_dealta

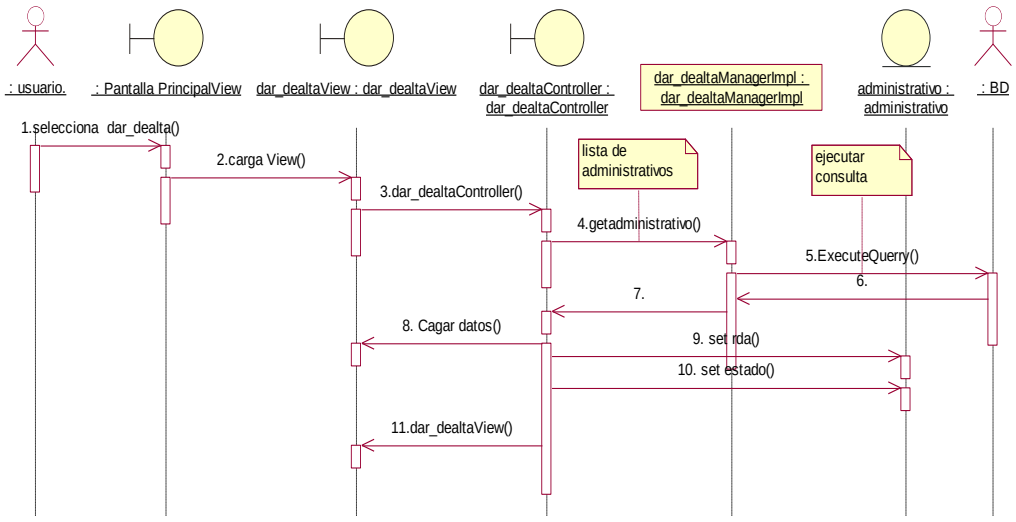


Figura 94. D.S. dar_dealta

II.1.8.2.2.18 Ver

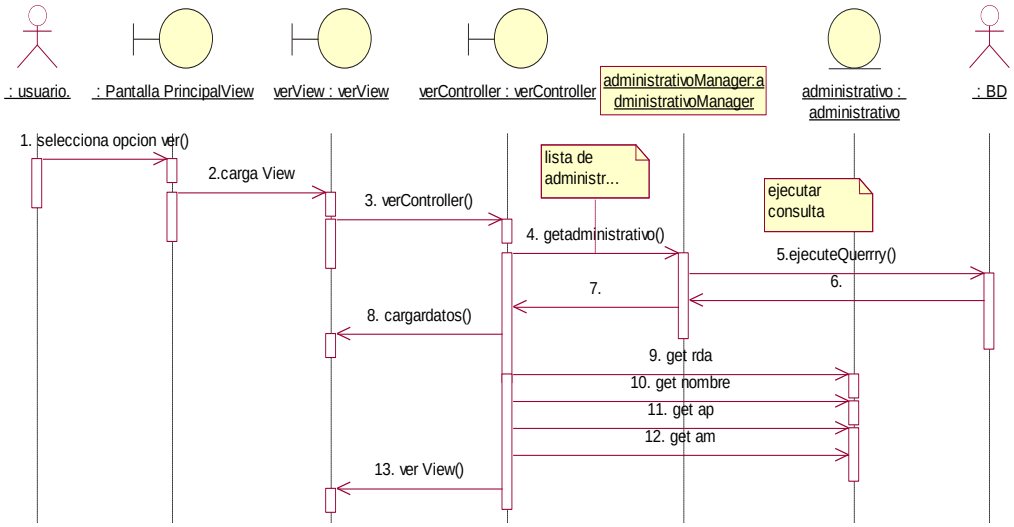


Figura 95. D.S. Ver

II.1.8.2.2.19 REG_ MATERIAS

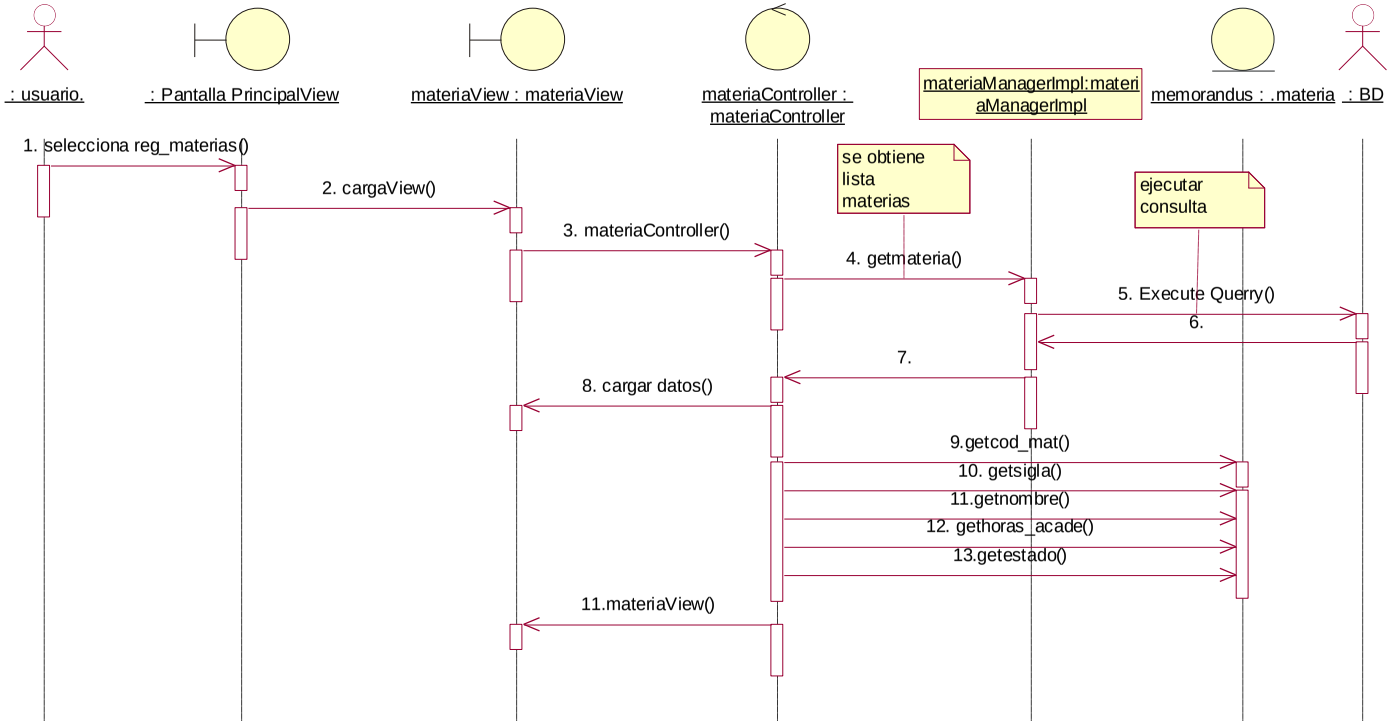

: usuario.
: Pantalla PrincipalView
materiaView : materiaView
materiaController : materiaController
materiaManagerImpl:materiaManagerImpl
memorandus : .materia
: BD
1. selecciona reg_materias()
2. cargaView()
3. materiaController()
se obtiene lista materias
4. getmateria()
ejecutar consulta
5. Execute Querry()
6.
7.
8. cargar datos()
9.getcod_mat()
10. getsigla()
11.getnombre()
12. gethoras_acade()
13.getestado()
11.materiaView()

Figura 96. D.S. ADMINISTRAR MATERIAS

II.1.8.2.2.20 adicionar_mat

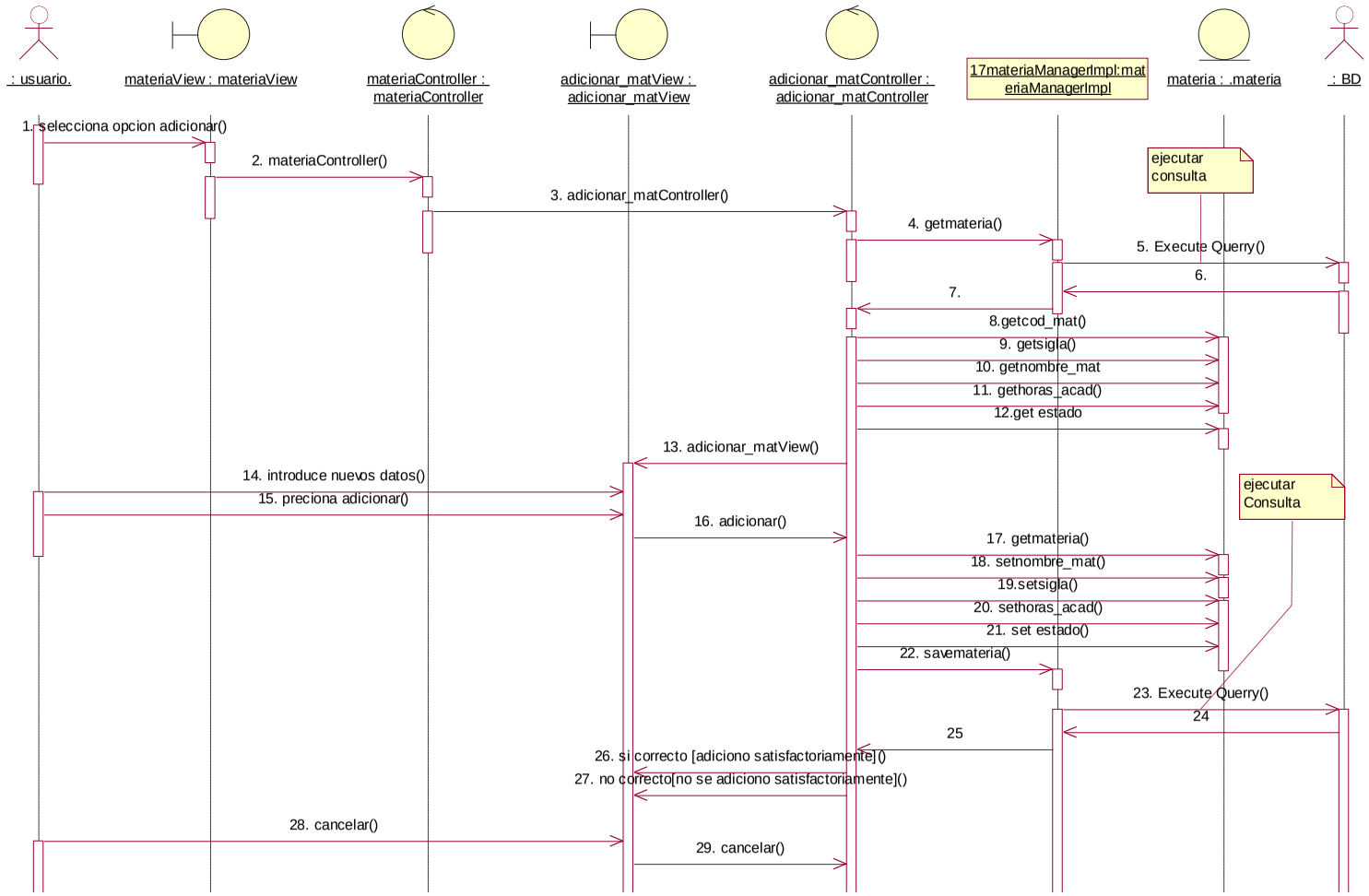

: usuario.
materiaView : materiaView
materiaController : materiaController
adicionar_matView : adicionar_matView
adicionar_matController : adicionar_matController
17materiaManagerImpl:materiaManagerImpl
materia : .materia
: BD
1. selecciona opcion adicionar()
2. materiaController()
3. adicionar_matController()
4. getmateria()
ejecutar consulta
5. Execute Querry()
6.
7.
8.getcod_mat()
9. getsigla()
10. getnombre_mat
11. gethoras_acad()
12.get estado
13. adicionar_matView()
14. introduce nuevos datos()
15. preciona adicionar()
ejecutar Consulta
16. adicionar()
17. getmateria()
18. setnombre_mat()
19.setsigla()
20. sethoras_acad()
21. set estado()
22. savemateria()
23. Execute Querry()
24
25
26. si correcto [adiciono satisfactoriamente]()
27. no correcto[no se adiciono satisfactoriamente]()
28. cancelar()
29. cancelar()

Figura 97. D.S. adicionar_mat

II.1.8.2.2.21 modificar_mat

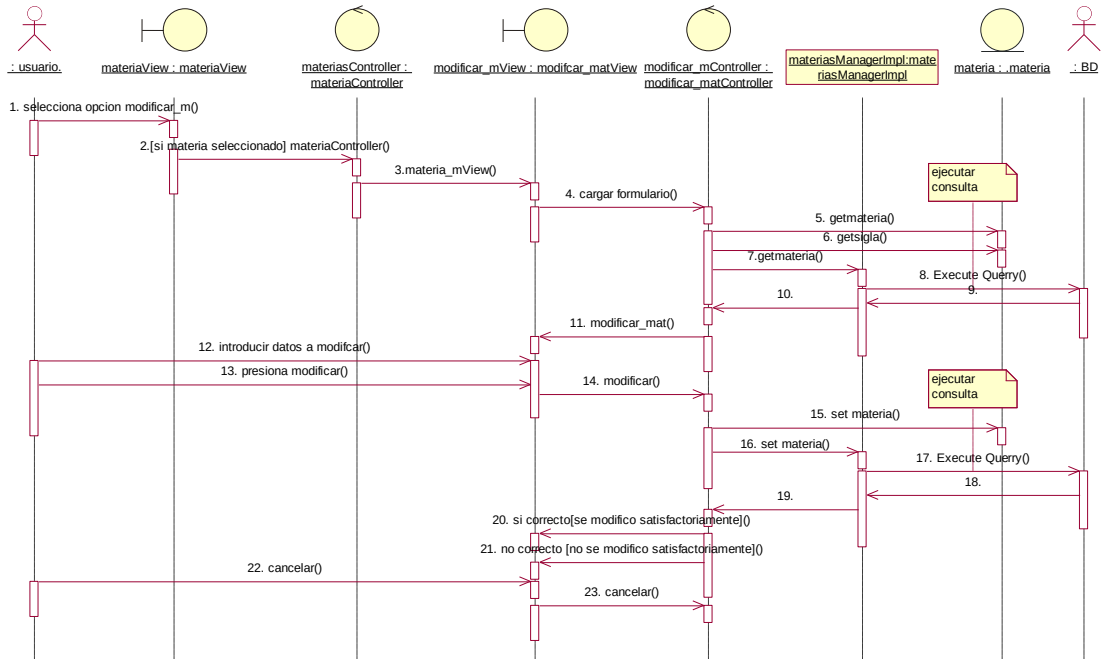


Figura 98. D.S. modificar_mat

II.1.8.2.2.22 eliminar_mat

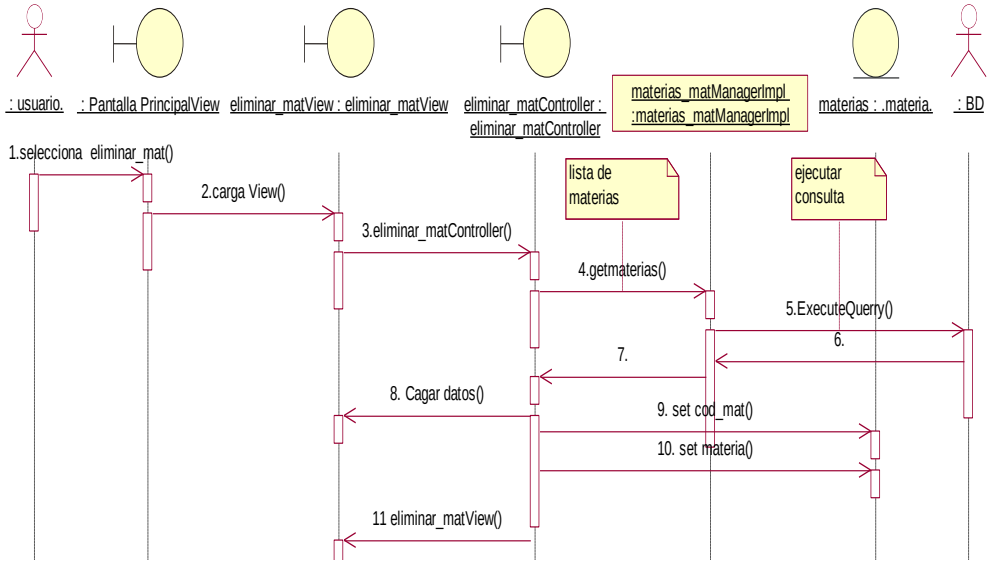


Figura 99. D.S. eliminar_mat

II.1.8.2.2.23 ADMINISTRAR HORARIO Asignar

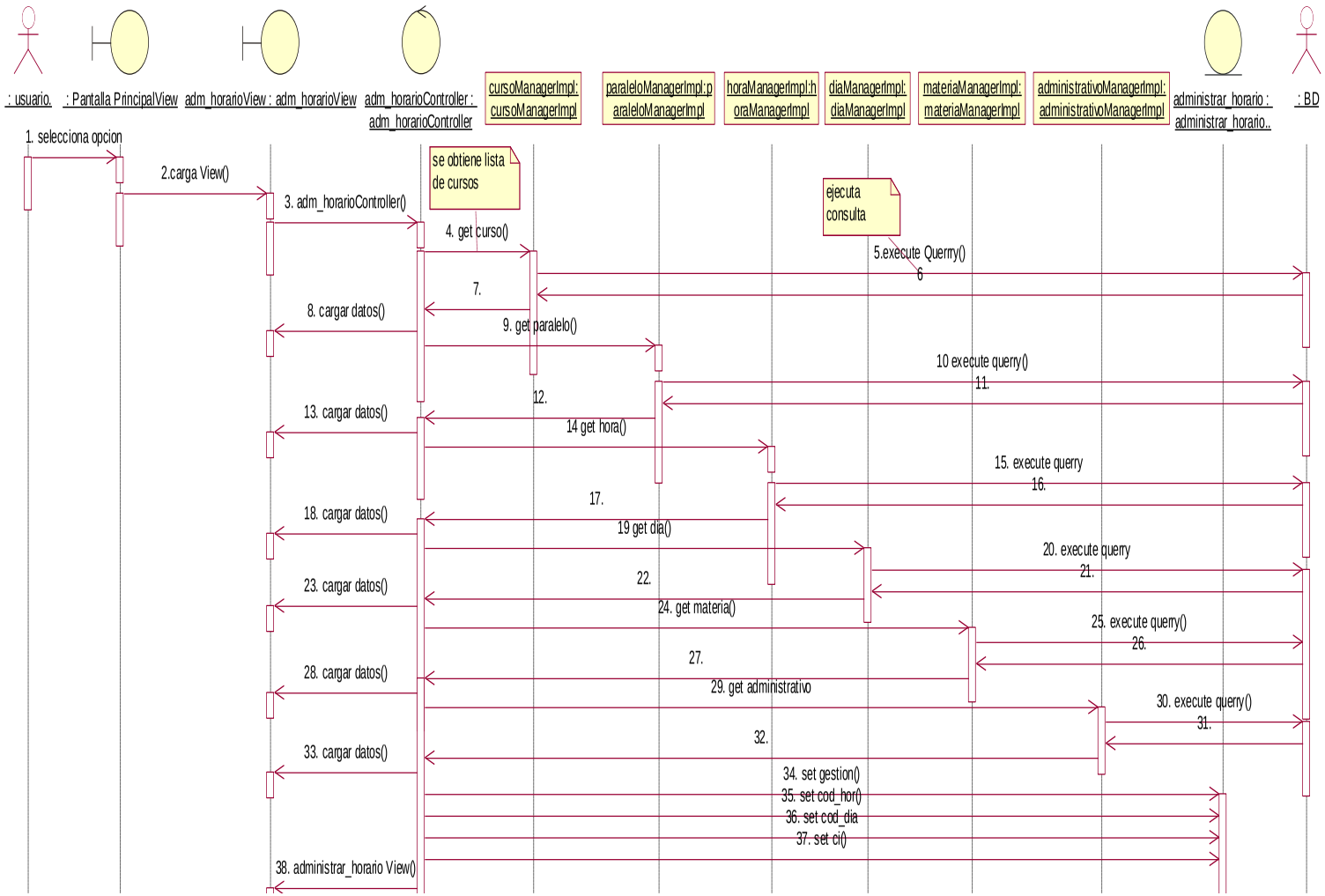

: usuario.
: Pantalla PrincipalView
adm_horarioView : adm_horarioView
adm_horarioController : adm_horarioController
cursoManagerImpl: cursoManagerImpl
paraleloManagerImpl:paraleloManagerImpl
horaManagerImpl:horaManagerImpl
diaManagerImpl: diaManagerImpl
materiaManagerImpl: materiaManagerImpl
administrativoManagerImpl: administrativoManagerImpl
administrar_horario : administrar_horario..
: BD
1. selecciona opcion
2.carga View()
3. adm_horarioController()
se obtiene lista de cursos
4. get curso()
ejecuta consulta
5.execute Querrry()
6
7.
8. cargar datos()
9. get paralelo()
10 execute querry()
11.
12.
13. cargar datos()
14 get hora()
15. execute querry
16.
17.
18. cargar datos()
19 get dia()
20. execute querry
21.
22.
23. cargar datos()
24. get materia()
25. execute querry()
26.
27.
28. cargar datos()
29. get administrativo
30. execute querry()
31.
32.
33. cargar datos()
34. set gestion()
35. set cod_hor()
36. set cod_dia
37. set ci()
38. administrar_horario View()

Figura 100. D.S. ADMINISTRAR HORARIO Asignar

II.1.8.2.2.24 MODIFICAR ASIGNACION

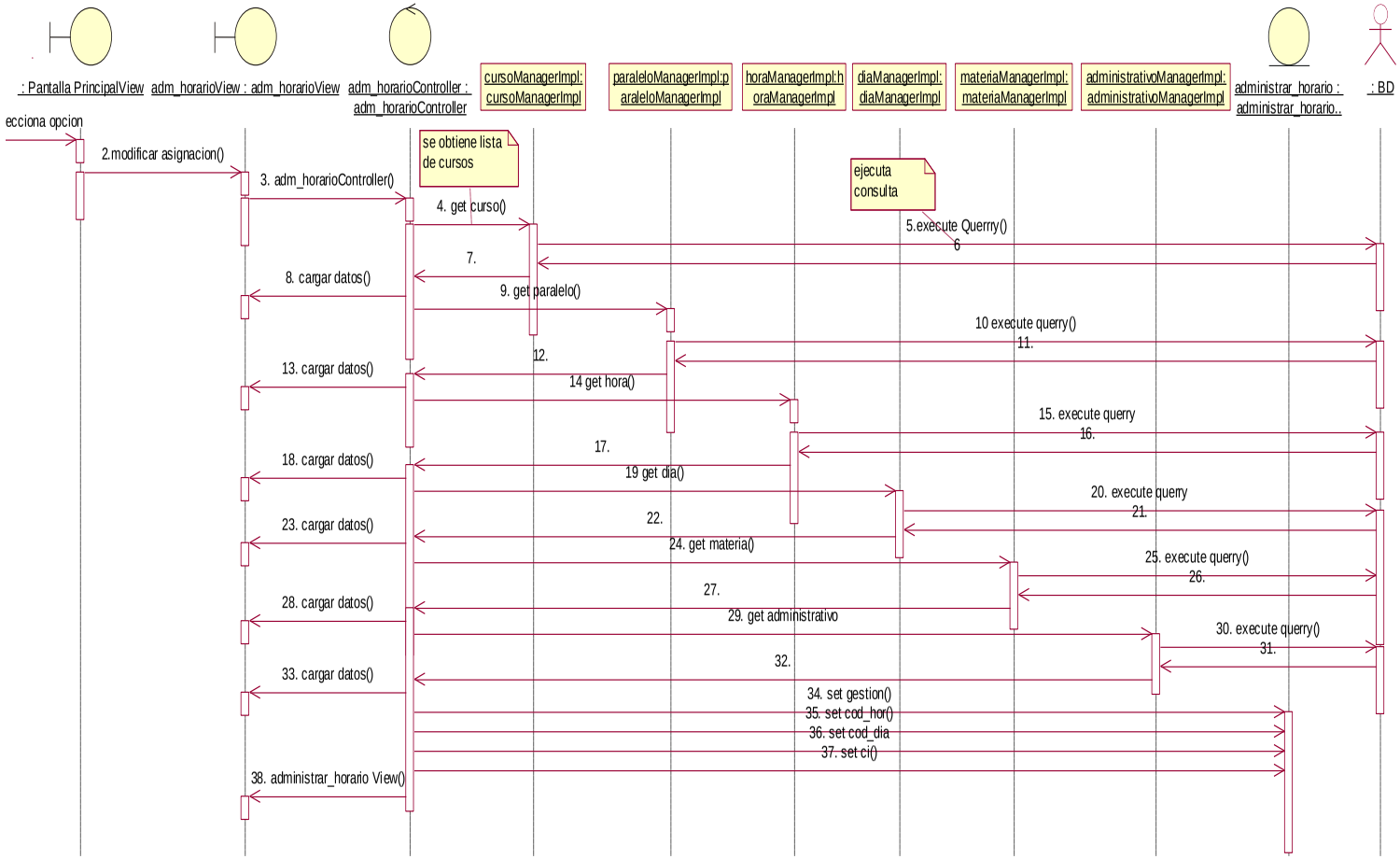

: Pantalla PrincipalView
adm_horarioView : adm_horarioView
adm_horarioController : adm_horarioController
cursoManagerImpl: cursoManagerImpl
paraleloManagerImpl:paraleloManagerImpl
horaManagerImpl:horaManagerImpl
diaManagerImpl: diaManagerImpl
materiaManagerImpl: materiaManagerImpl
administrativoManagerImpl: administrativoManagerImpl
administrar_horario : administrar_horario..
: BD
ecciona opcion
2.modificar asignacion()
3. adm_horarioController()
se obtiene lista de cursos
4. get curso()
ejecuta consulta
5.execute Querrry()
6
7.
8. cargar datos()
9. get paralelo()
10 execute querry()
11.
12.
13. cargar datos()
14 get hora()
15. execute querry
16.
17.
18. cargar datos()
19 get dia()
20. execute querry
21.
22.
23. cargar datos()
24. get materia()
25. execute querry()
26.
27.
28. cargar datos()
29. get administrativo
30. execute querry()
31.
32.
33. cargar datos()
34. set gestion()
35. set cod_hor()
36. set cod_dia
37. set ci()
38. administrar_horario View()

Figura 101. D.S. MODIFICAR ASIGNACION

II.1.8.2.2.25 REPORTES CREAR_REP

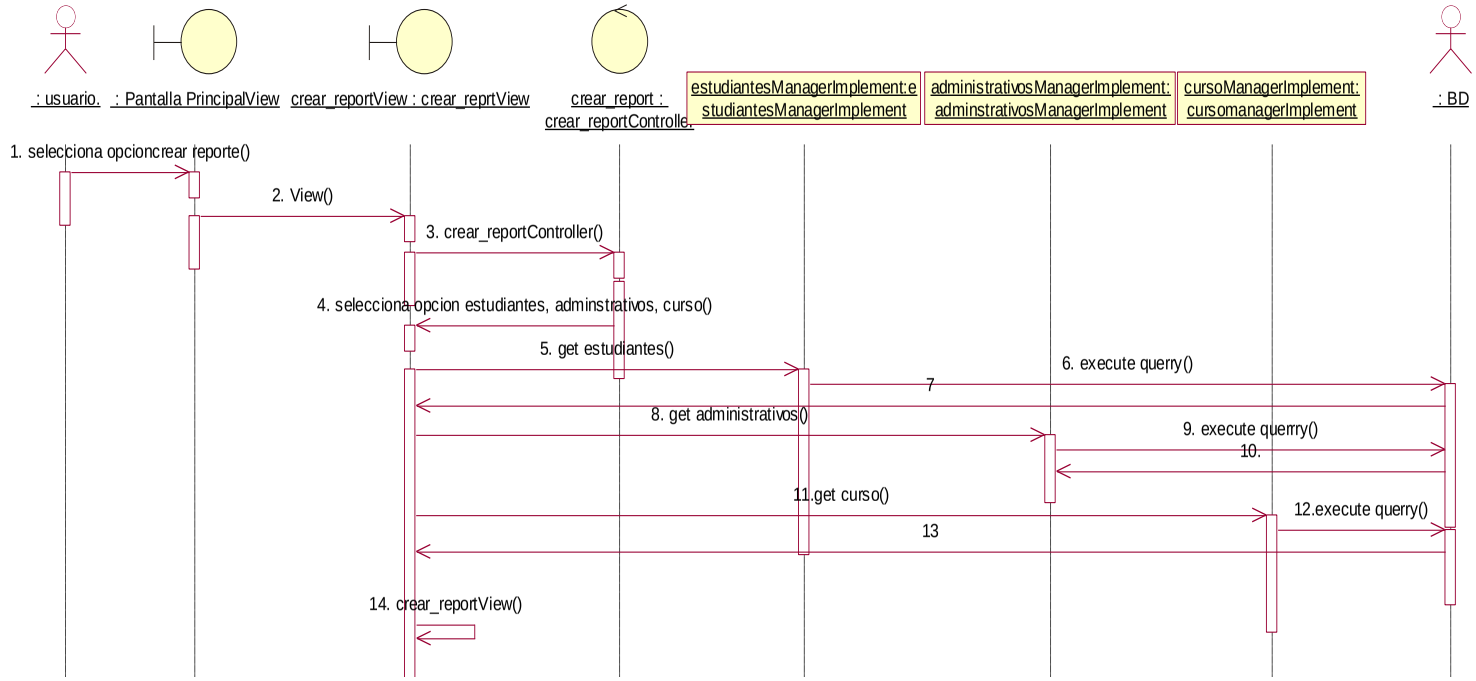

: usuario.
: Pantalla PrincipalView
crear_reportView : crear_reprtView
crear_report : crear_reportControlle
estudiantesManagerImplement:estudiantesManagerImplement
administrativosManagerImplement:adminstrativosManagerImplement
cursoManagerImplement:cursomanagerImplement
: BD
1. selecciona opcioncrear reporte()
2. View()
3. crear_reportController()
4. selecciona opcion estudiantes, adminstrativos, curso()
5. get estudiantes()
6. execute querry()
7
8. get administrativos()
9. execute querry()
10.
11.get curso()
12.execute querry()
13
14. crear_reportView()

Figura 102. D.S. crear_reporte

II.1.8.3 Modelado de Diagramas de Colaboracion

II.1.8.3.1 INICIAR

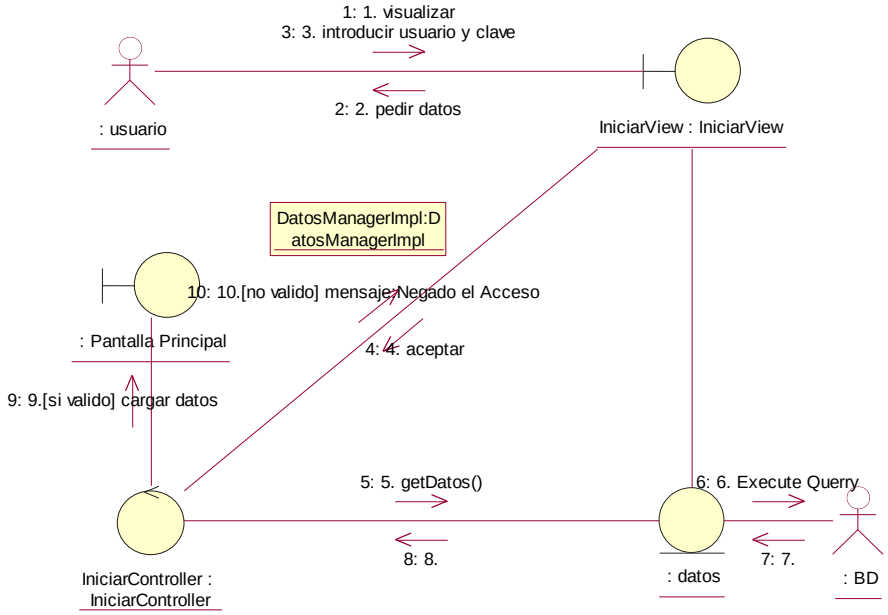


Figura 103. D.C. INICIAR

II.1.8.3.2 Reg_estudiante

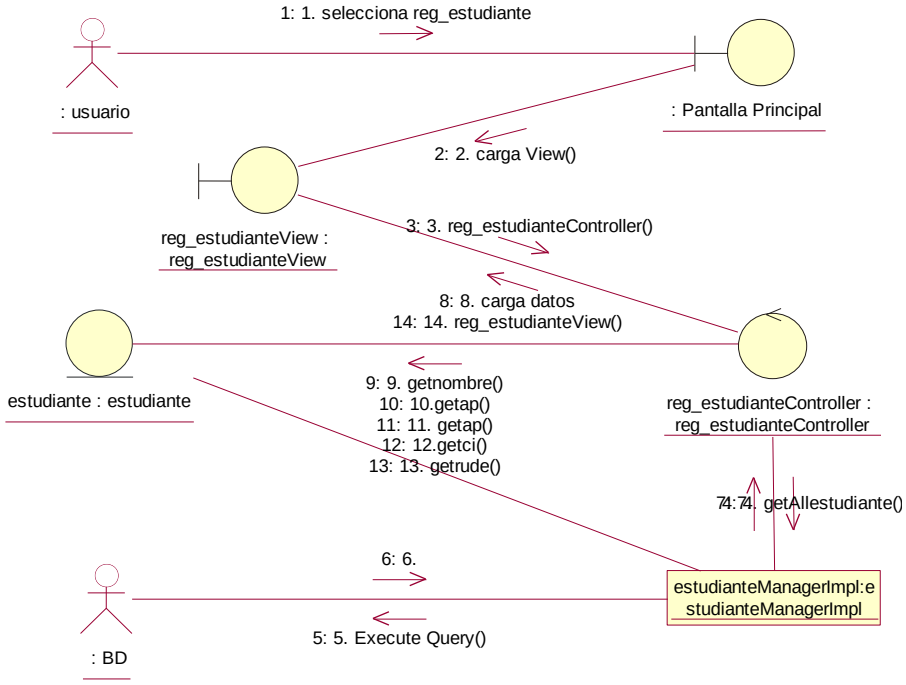


Figura 104. D.C. Reg_estudiante

II.1.8.3.3 adicionar

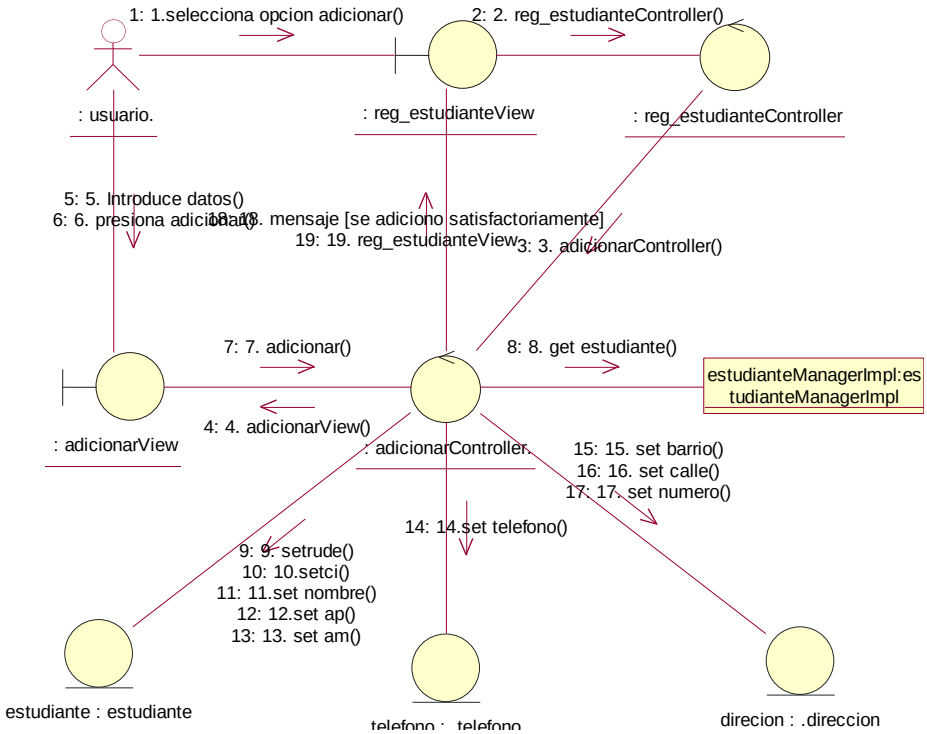


Figura 105. D.C. adicionar

II.1.8.3.4 Modificar

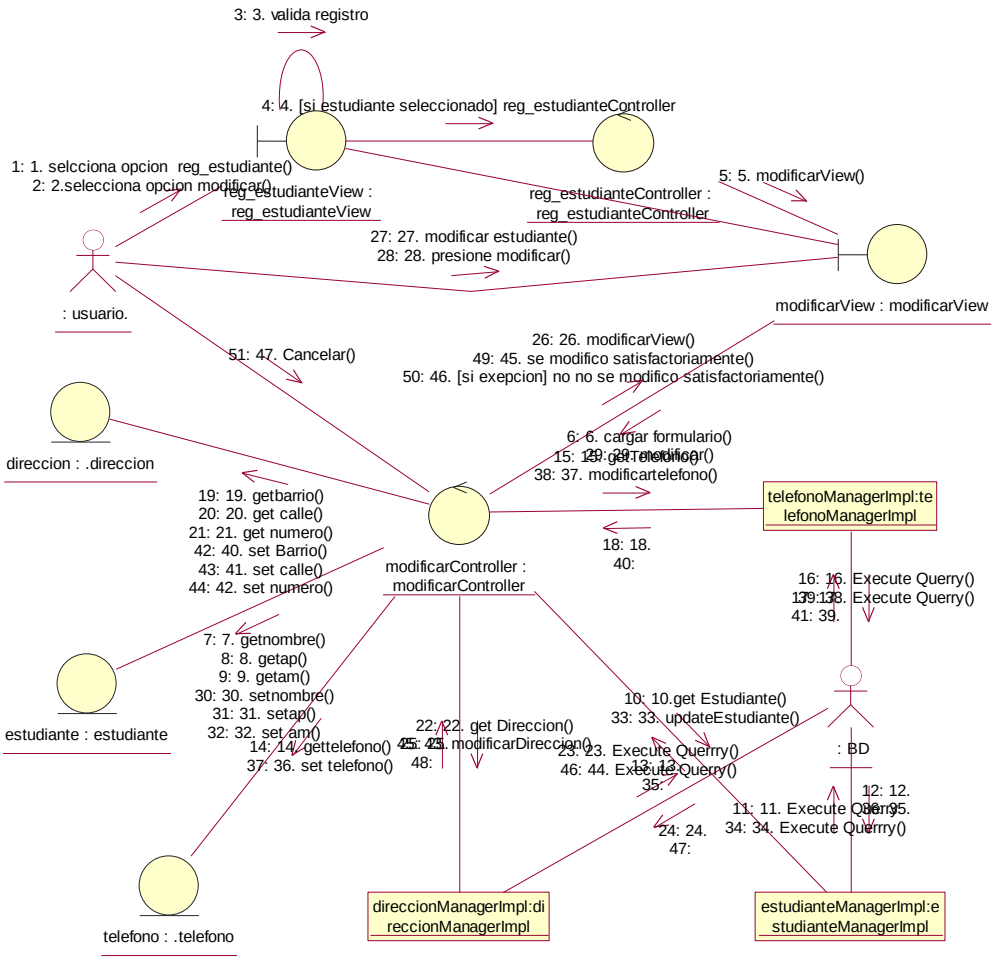


Figura 106. D.C. Modificar

II.1.8.3.5 dar de baja

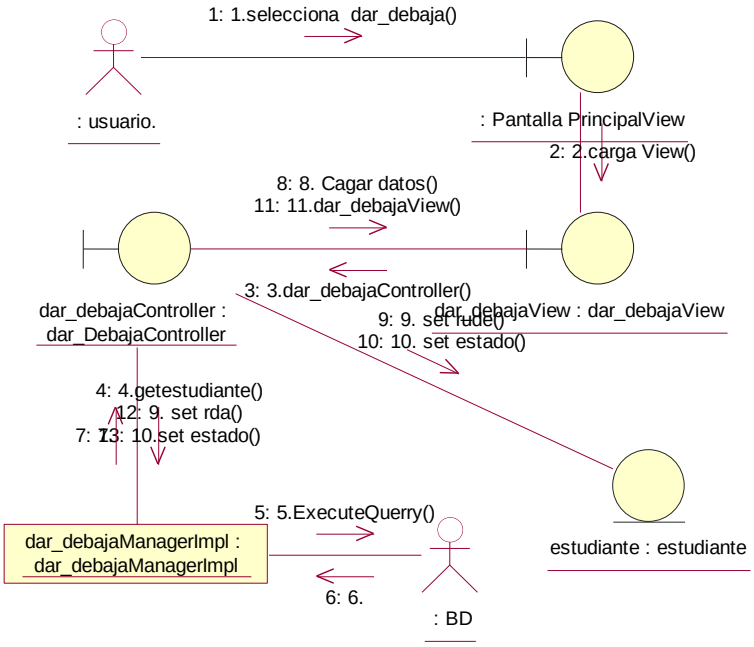


Figura 107. D.C. dar de baja

II.1.8.3.6 dar de alta

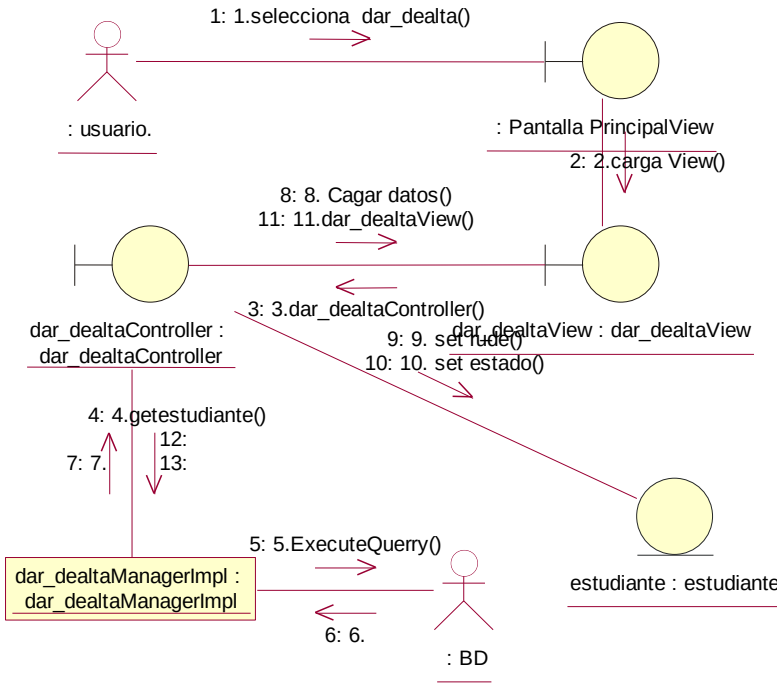


Figura 108. D.C. dar de alta

II.1.8.3.7 Ver

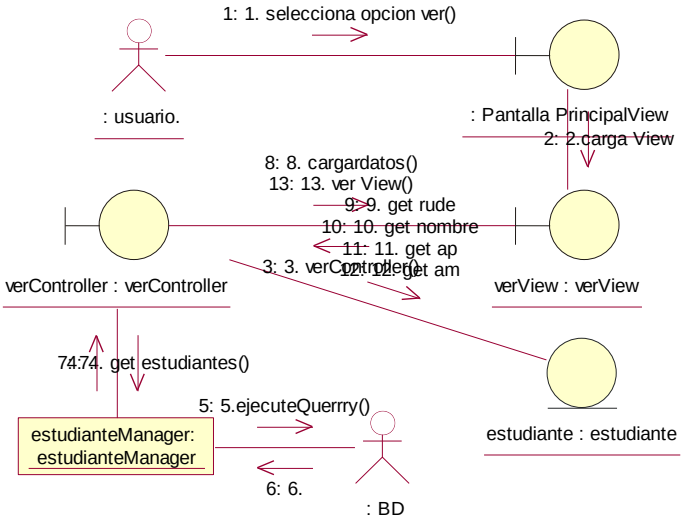


Figura 108. D.C. Ver

II.1.8.3.8 MEMORANDUS

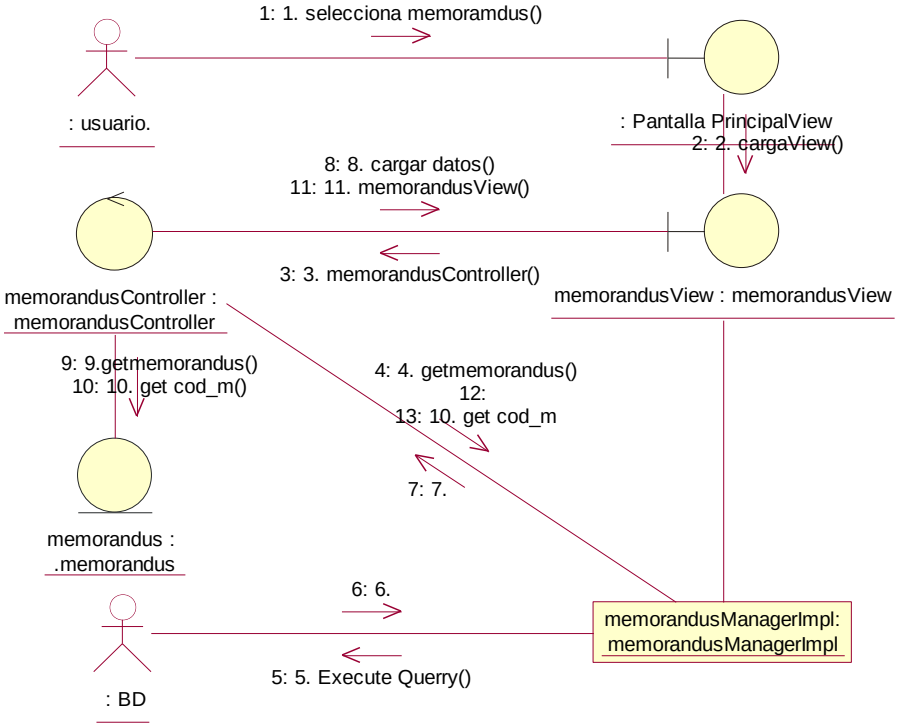


Figura 109. D.C. MEMORANDUS

II.1.8.3.9 adicionar_m

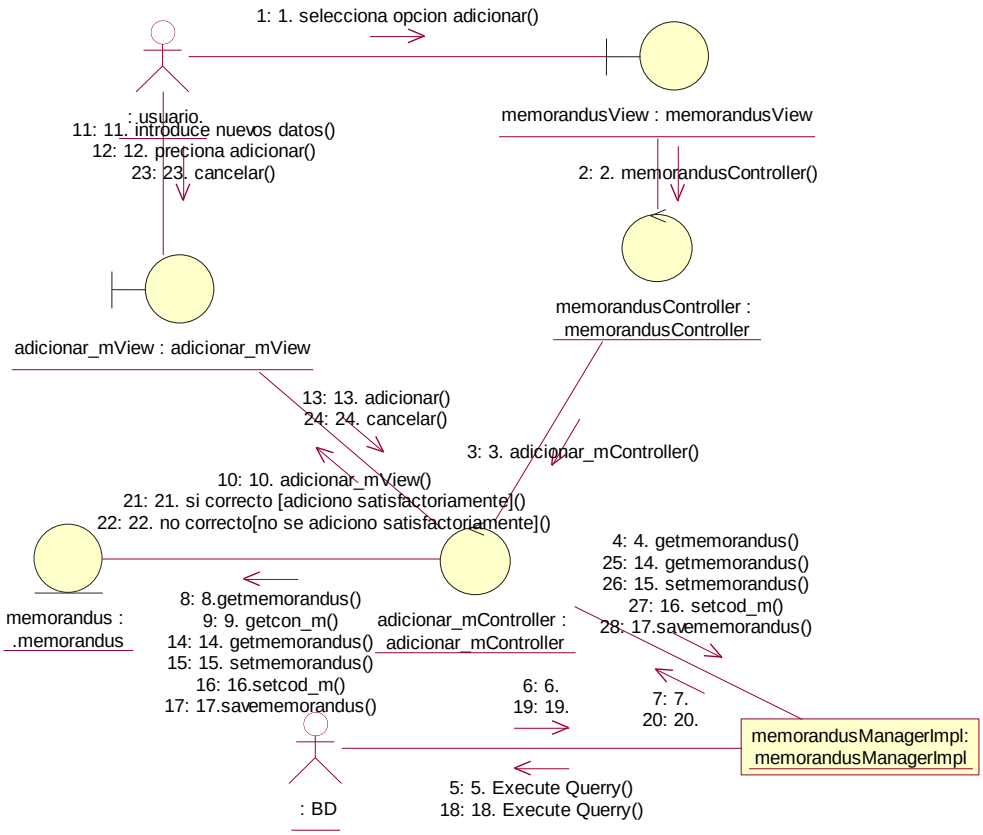


Figura 110. D.C. adicionar_m

II.1.8.3.10 modificar_m

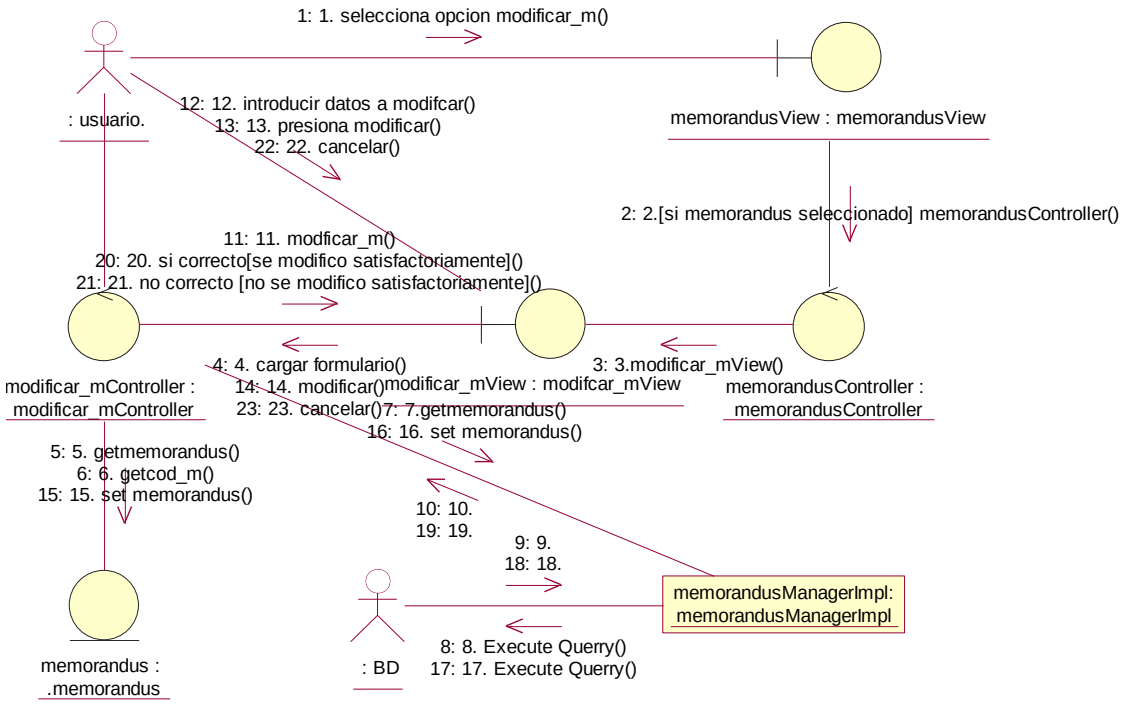


Figura 111. D.C. modificar_m

II.1.8.3.11 eliminar_m

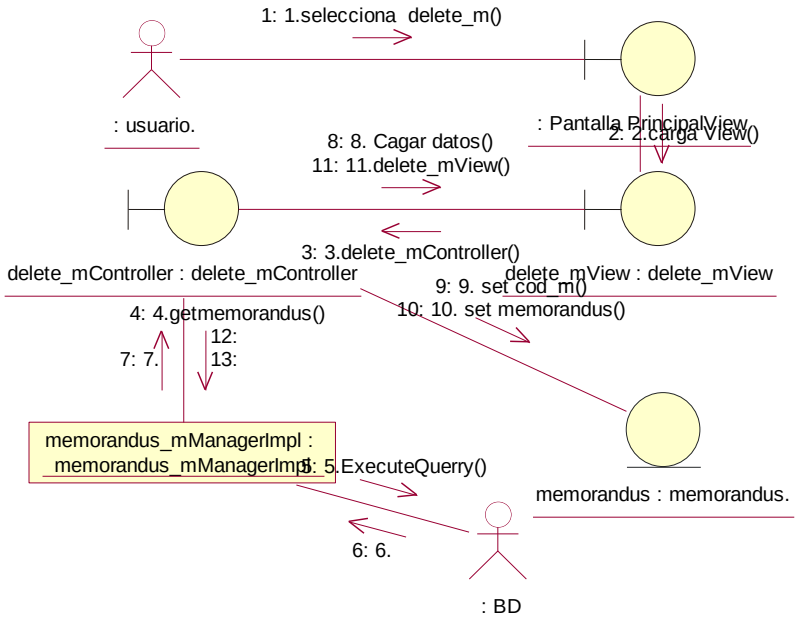


Figura 112. D.C. eliminar_m

II.1.8.3.12 ver_m

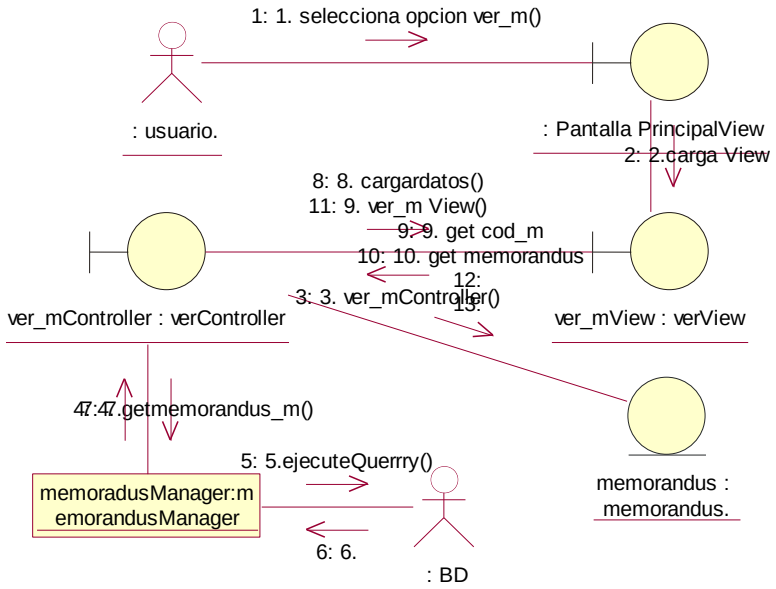


Figura 113. D.C. ver_m

II.1.8.3.13 REG_ADMINISTRATIVO

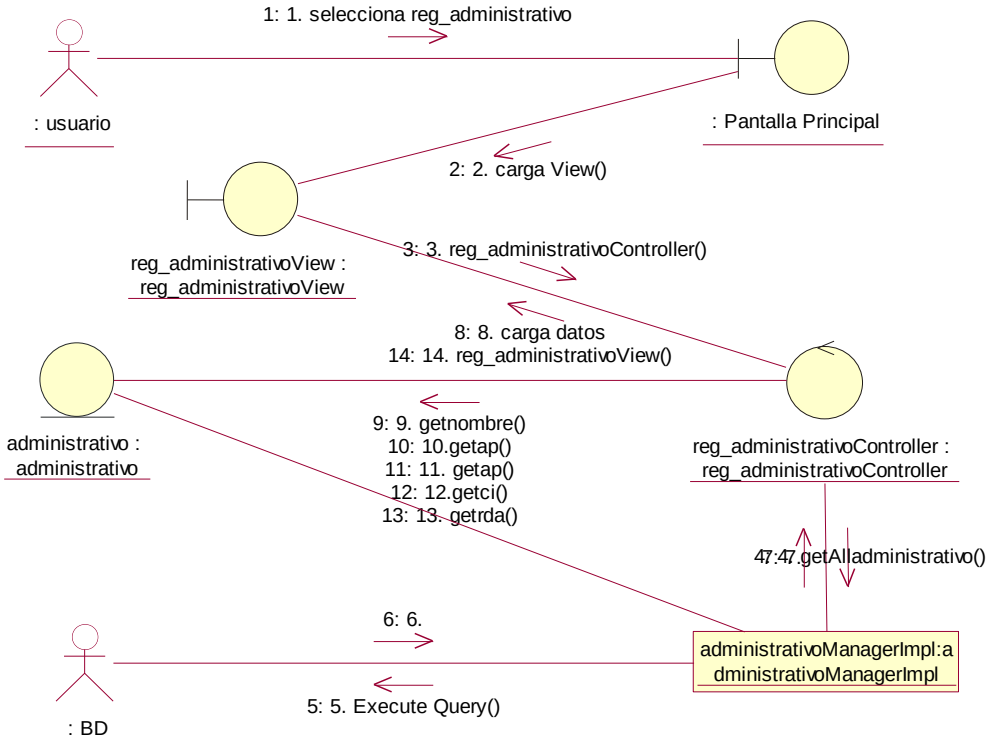


Figura 114. D.C. REG_ADMINISTRATIVO

II.1.8.3.14 Adicionar

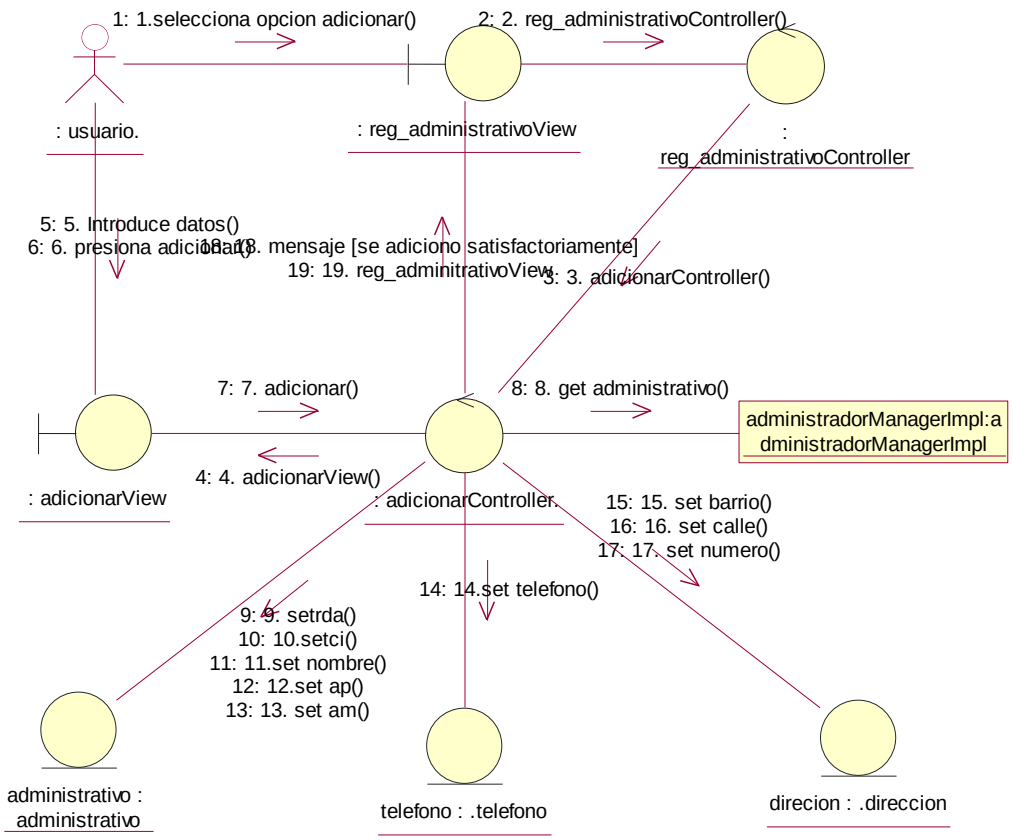


Figura 115. D.C. Adicionar

II.1.8.3.15 Modificar

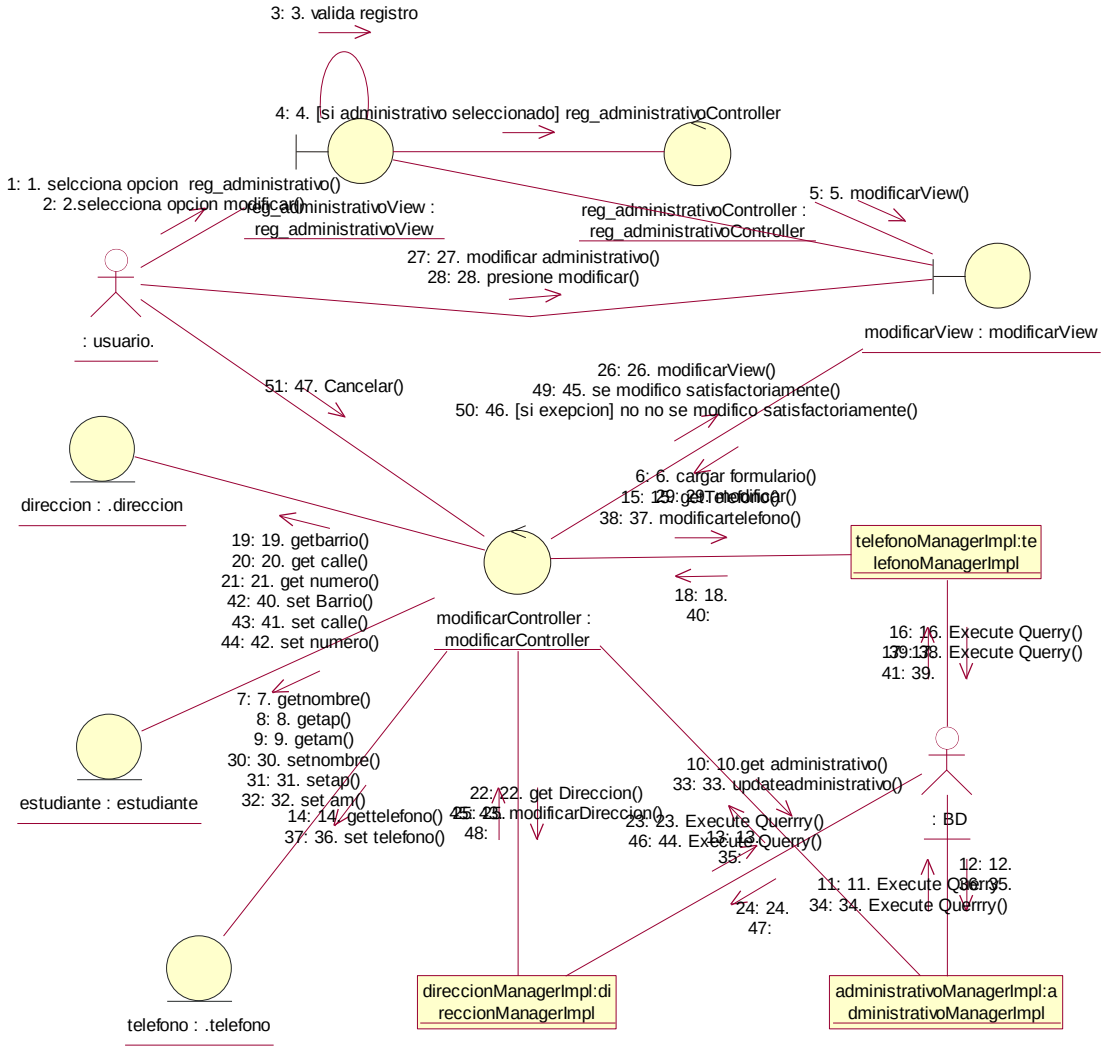


Figura 116. D.C. Modificar

II.1.8.3.16 dar_debaja

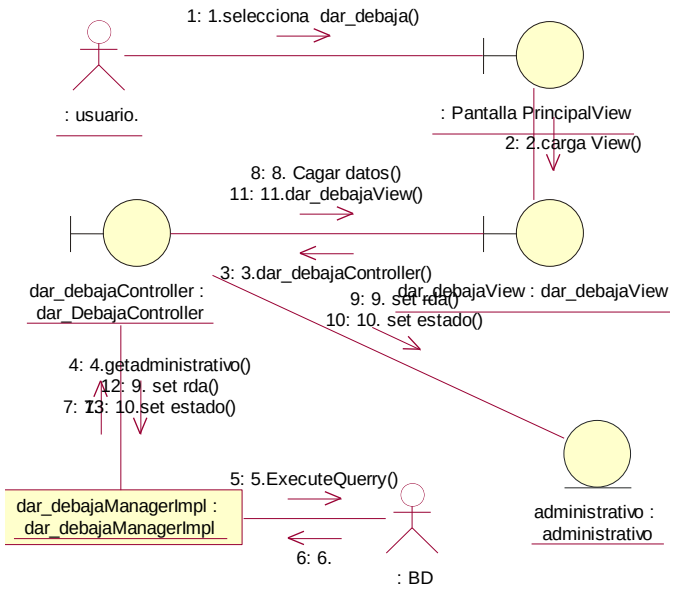


Figura 117. D.C. dar_debaja

II.1.8.3.17 dar_dealta

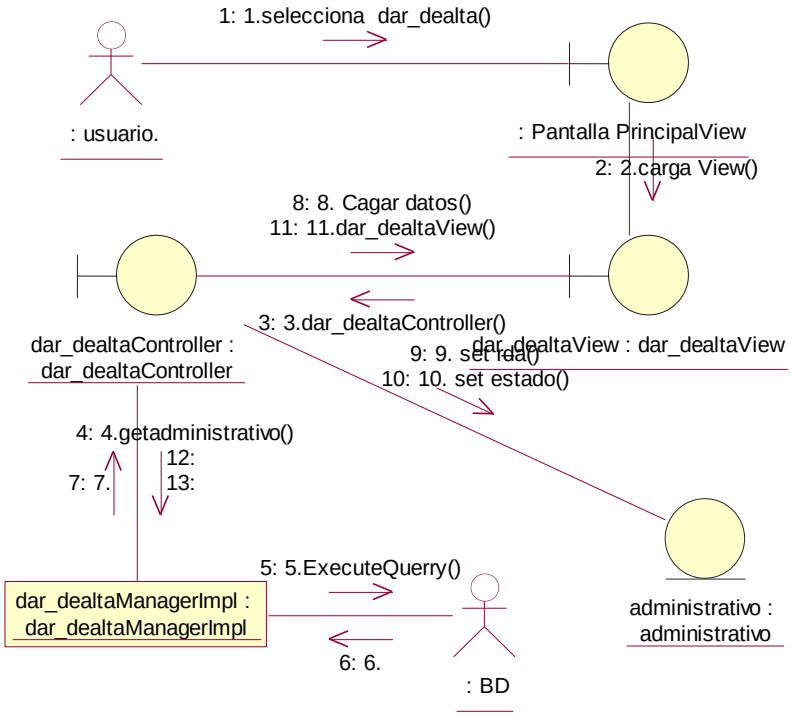


Figura 118. D.C. dar_dealta

II.1.8.3.18 Ver

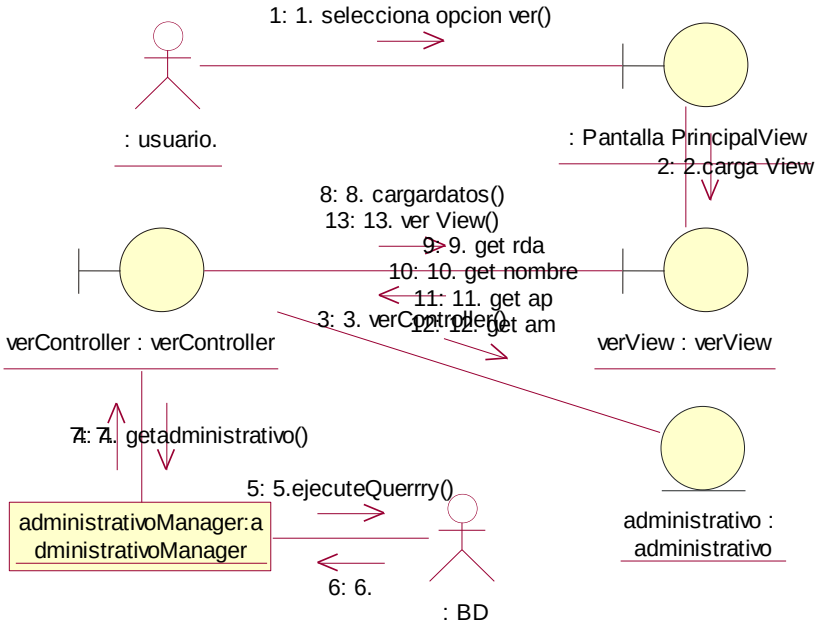


Figura 119. D.C. Ver

II.1.8.3.19 REG_MATERIAS

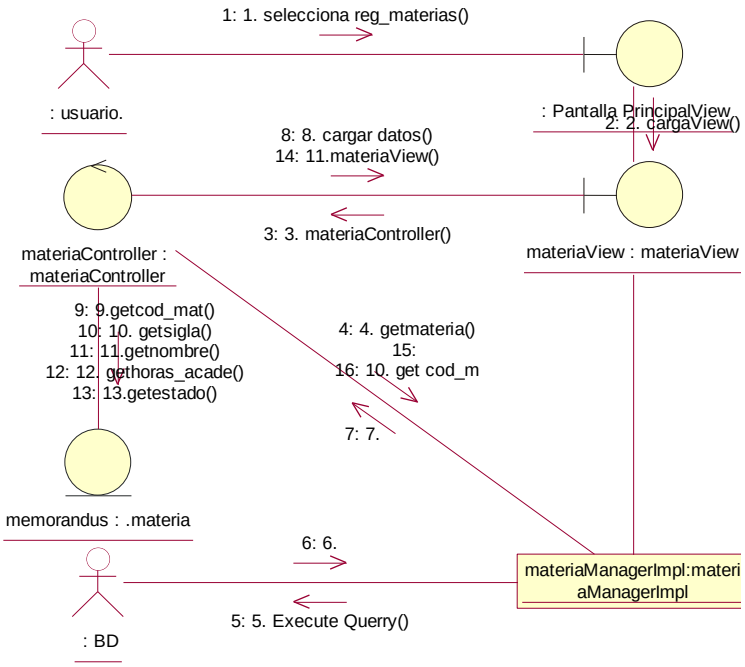


Figura 120. D.C. REG_MATERIAS

II.1.8.3.20 adicionar_mat

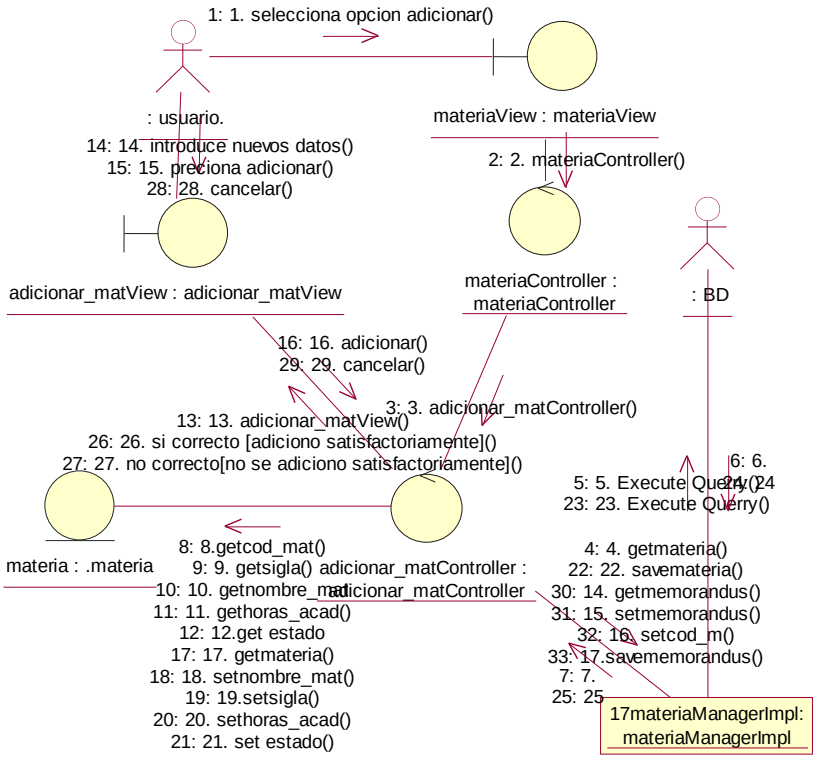


Figura 121. D.C. adicionar_mat

II.1.8.3.21 modificar_mat

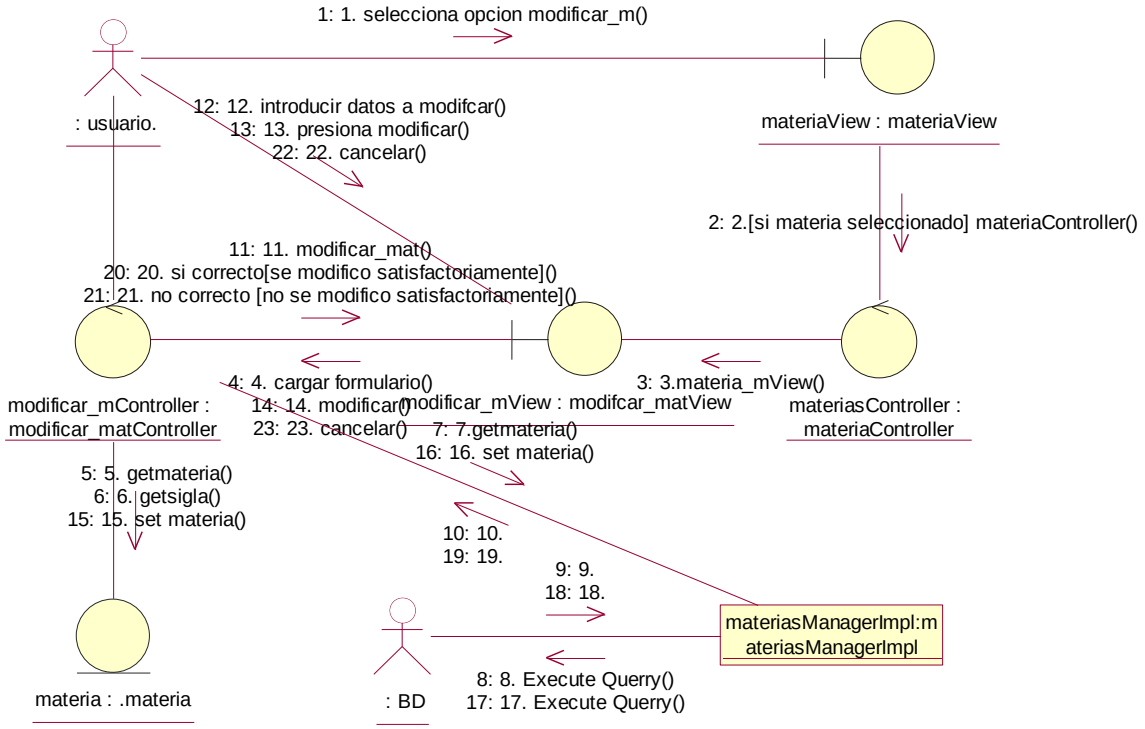


Figura 122. D.C. modificar_mat

II.1.8.3.22 eliminar_mat

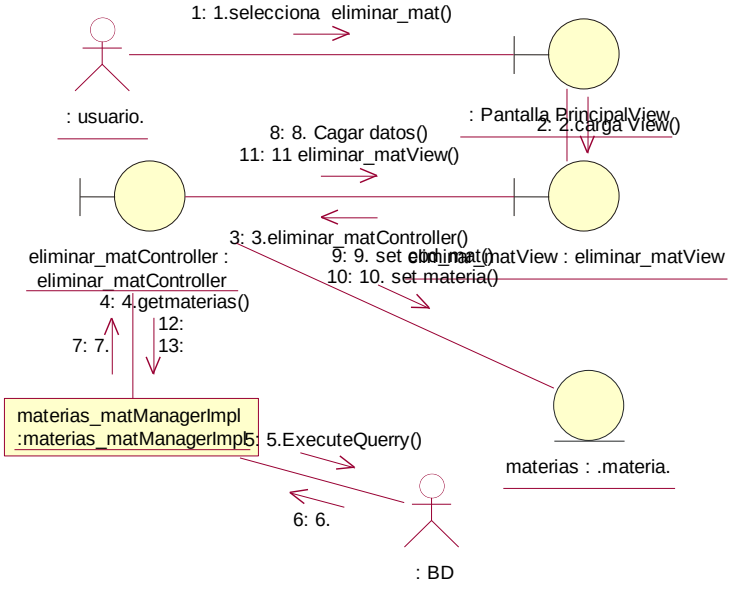


Figura 123. D.C. eliminar_mat

II.1.8.3.23 ADMINISTRAR HORARIO Asignar

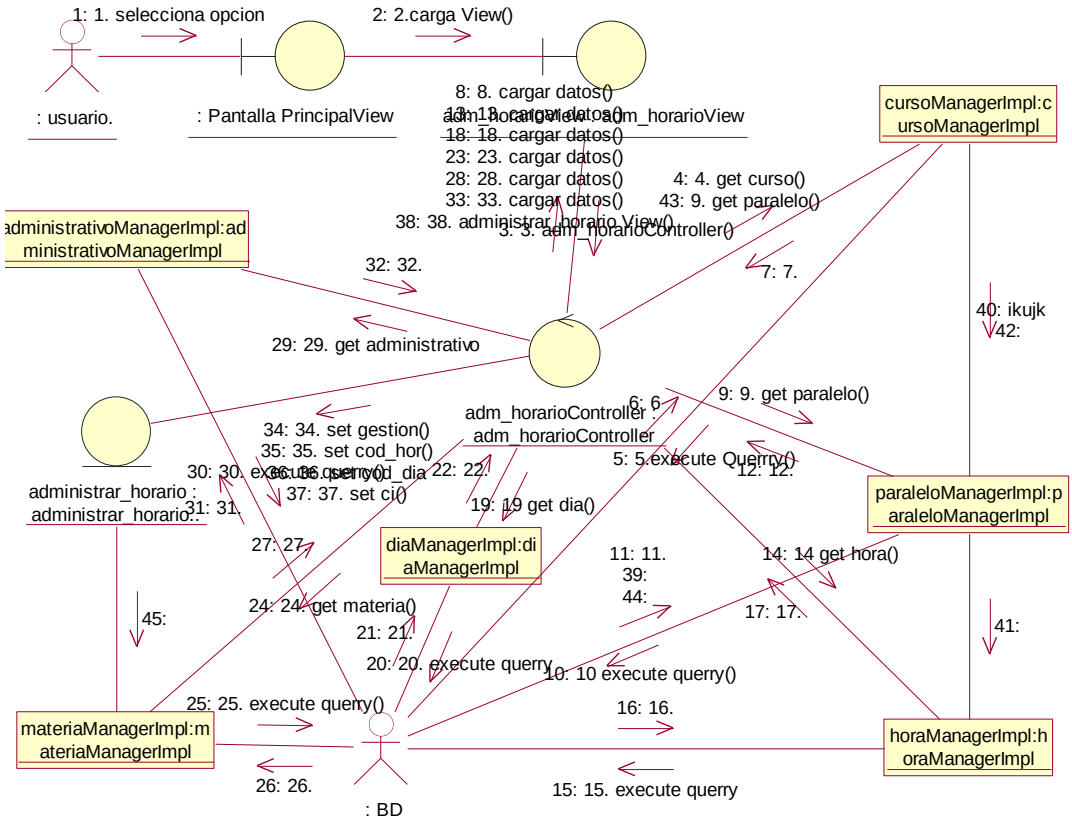


Figura 124. D.C. ADMINISTRAR HORARIO Asignar

II.1.8.3.24 MODIFICAR

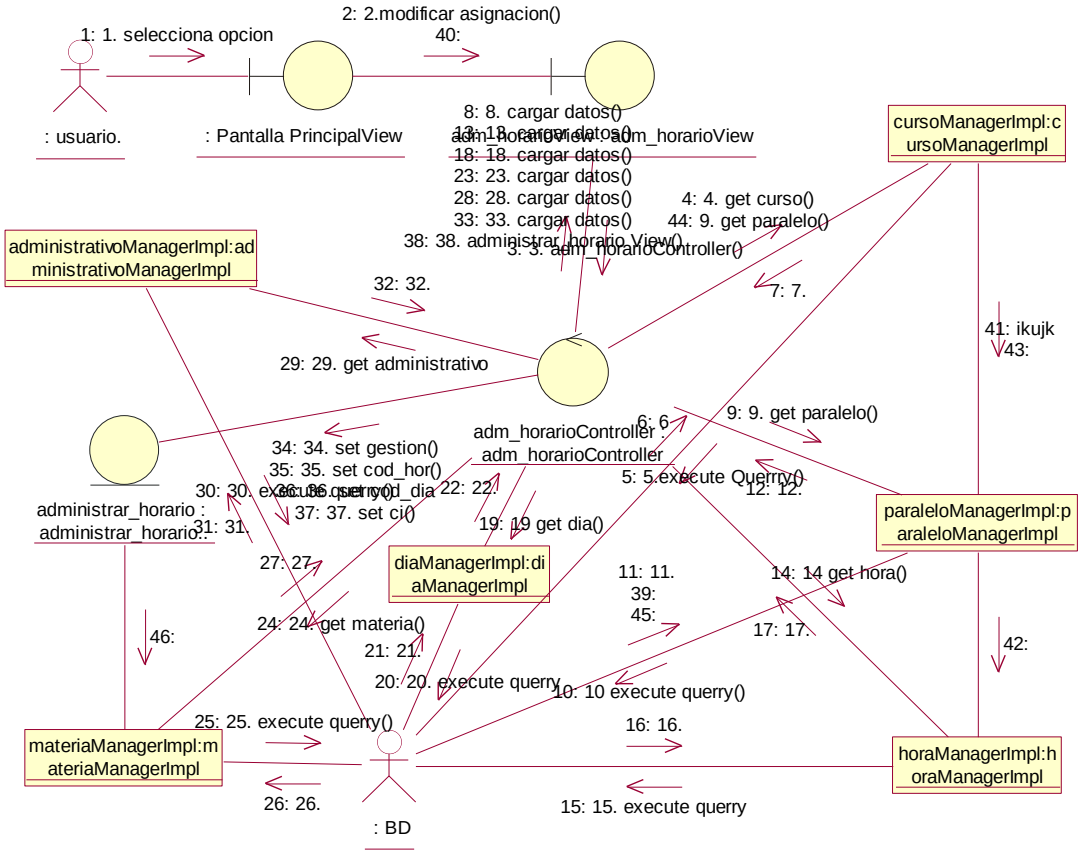


Figura 125. D.C. MODIFICAR

II.1.8.3.25 REPORTES

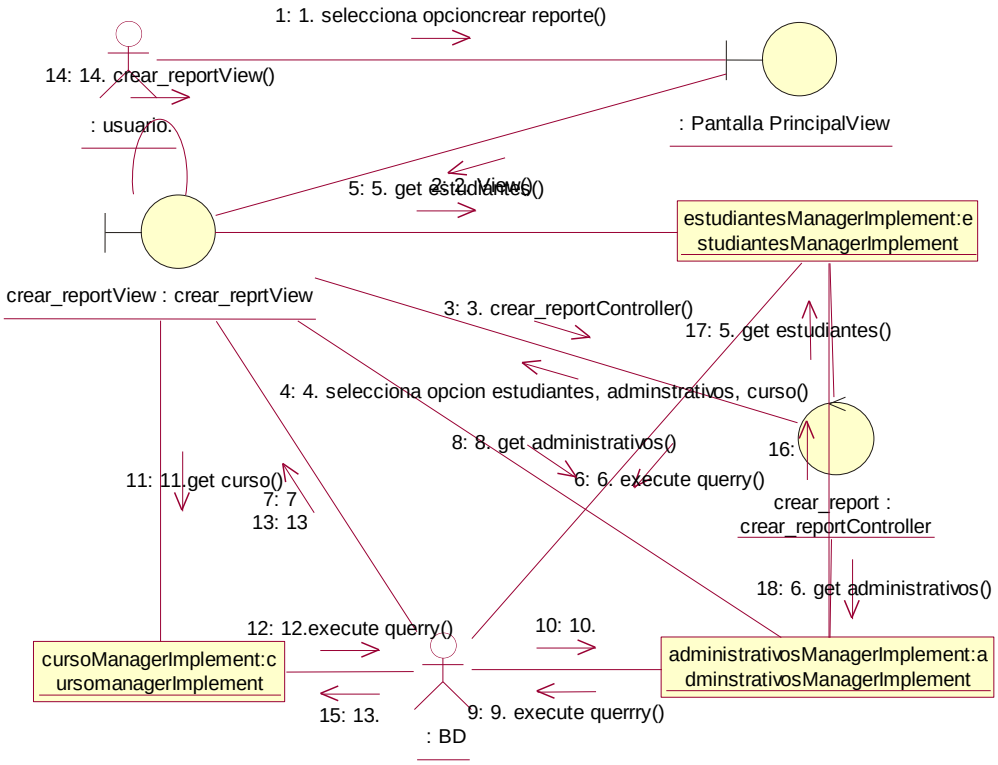


Figura 126. D.C. REPORTES

II.1.9 MODELO DE DATOS

II.1.9.1 Modelado de Diagramas de clases

II.1.9.1.1 Introducción

El modelado de Diagrama de Clases en un artefacto de la disciplina de Análisis y Diseño en la Metodología RUP que se está implementando

II.1.9.1.1.1 Propósito

- Comprender la estructura del sistema deseado para los establecimientos educativos
- Identificar clases de análisis y diseño.

II.1.9.1.1.2 Alcance

- Describir las clases y objetos de diseño del sistema en su segunda iteración
- Identificar y definir los objetos del sistema según los objetos del sistema deseado, aprobado para la Unidad Educativa

II.1.9.1.2 DIAGRAMAS DE CLASES

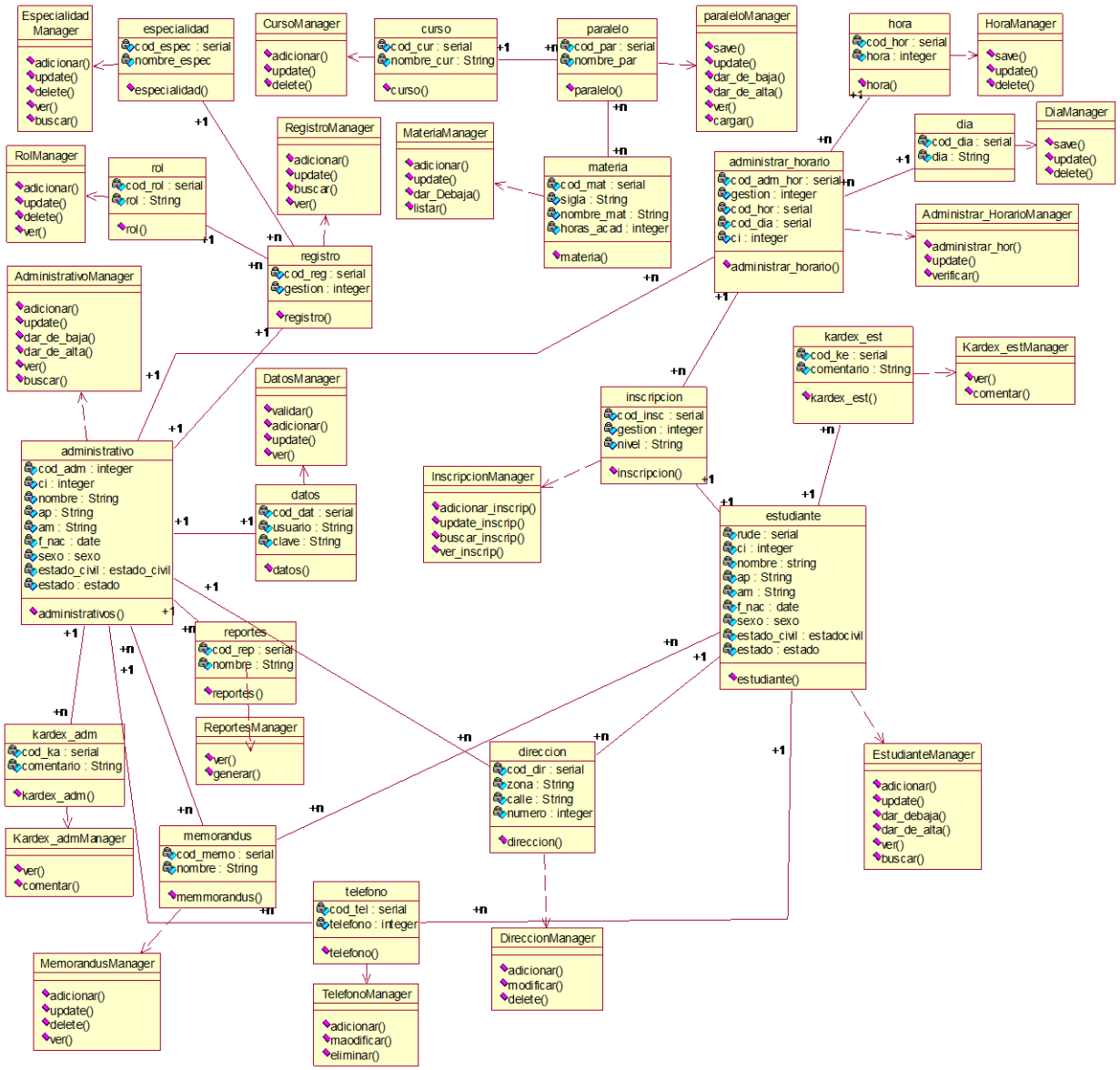


Figura 127. DIAGRAMAS DE CLASES

II.1.9.2 MODELO DE DATOS

II.1.9.2.1 Introducción

El modelado de Datos nos sirve para tener un detalle de las tablas con sus respectivos campos de la base de datos

II.1.9.2.1.1 Propósito

- Comprender las estructuras de las tablas y sus campos, en la base de datos de nuestro sistema deseado para los establecimientos educativos
- Identificar los tipos de campos de cada tabla de la base de datos

II.1.9.2.1.2 Alcance

- Describir los campos de cada tabla de la base de datos especificando el tipo, longitud y descripción de cada campo.
- Identificar y definir relaciones entre las diferentes tablas de la base de datos de nuestro sistema deseado para los establecimientos educativos

II.1.9.2.2 Scrip de la Base de Datos

II.1.9.2.2.1 Tabla Rol

```
CREATE TABLE rol (
   cod_rol SERIAL NOT NULL,
   rol VARCHAR(20),
   PRIMARY KEY (cod_rol)
);
```

II.1.9.2.2.2 Tabla especialidad

```
CREATE TABLE especialidad (
cod_espec SERIAL NOT NULL,
 nombre_esp VARCHAR(40) NOT NULL,
 PRIMARY KEY (cod_espec)
```

);

II.1.9.2.2.3 Tabla registro

```
CREATE TABLE registro (
    cod_reg SERIAL NOT NULL,
    gestion INT4 NOT NULL,
    cod_rol INT4 NOT NULL,
    cod_espec INT4 NOT NULL,
    ci VARCHAR(40) NOT NULL,
    horas FLOAT8,
    PRIMARY KEY (cod_reg)
);
```

II.1.9.2.2.4 Tabla paralelo

```
CREATE TABLE paralelo (
    cod_par SERIAL NOT NULL,
    nom_par VARCHAR(40) NOT NULL,
    cod_cur INT4 NOT NULL,
    PRIMARY KEY (cod_par)
);
```

II.1.9.2.2.5 Tabla curso

```
CREATE TABLE curso (
    cod_cur SERIAL NOT NULL,
    nombre_cur VARCHAR(40) NOT NULL,
    PRIMARY KEY (cod_cur)
```

);

II.1.9.2.2.6 Tabla materia

```
CREATE TABLE materia (
   cod_mat SERIAL NOT NULL,
   sigla VARCHAR(20) NOT NULL,
   nombre_mat VARCHAR(20) NOT NULL,
   horas_acad FLOAT8 NOT NULL,
   PRIMARY KEY (cod_mat)
);
```

II.1.9.2.2.7 Tabla hora

```
CREATE TABLE hora (
   cod_hor SERIAL NOT NULL,
   hora VARCHAR(20) NOT NULL,
   PRIMARY KEY (cod_hor)
);
```

II.1.9.2.2.8 Tabla dia

```
CREATE TABLE dia (
   cod_dia SERIAL NOT NULL,
   dia VARCHAR(10),
   PRIMARY KEY (cod_dia)
);
```

II.1.9.2.2.9 Tabla administrar_horario

```
CREATE TABLE administrar_horario (
   cod_adm_hor SERIAL NOT NULL,
   cod_hor INT4 NOT NULL,
   cod_dia INT4 NOT NULL,
   ci VARCHAR(40) NOT NULL,
   cod_m_p INT4 NOT NULL,
   PRIMARY KEY (cod_adm_hor)
);
```

II.1.9.2.2.10 Tabla inscripcion

```
CREATE TABLE inscripcion (
   cod_ins SERIAL NOT NULL,
   nivel VARCHAR(10),
   rude INT8 NOT NULL,
   cod_adm_hor INT4 NOT NULL,
   PRIMARY KEY (cod_ins, cod_adm_hor)
);
```

II.1.9.2.2.11 Tabla datos

```
CREATE TABLE datos (
   cod_dat SERIAL NOT NULL,
   usuario VARCHAR(10) NOT NULL,
   clave VARCHAR(10) NOT NULL,
   ci VARCHAR(40) NOT NULL,
   PRIMARY KEY (cod_dat)
```

);

II.1.9.2.2.12 Tabla administrativo

```
CREATE TABLE administrativo (
    ci VARCHAR(8) NOT NULL,
    rda INT8 NOT NULL,
    nombre VARCHAR(20) NOT NULL,
    ap VARCHAR(20),
    am VARCHAR(20) NOT NULL,
    f_nac DATE,
    sexo VARCHAR(1),
    estado_civil VARCHAR(1),
    estado INT2 DEFAULT 1 NOT NULL,
    tipo INT2,
    PRIMARY KEY (ci)
);
```

II.1.9.2.2.13 Tabla estudiante

```
CREATE TABLE estudiante (
    rude INT8 NOT NULL,
    ci VARCHAR(8) UNIQUE,
    nombre VARCHAR(20) NOT NULL,
    ap VARCHAR(20),
    am VARCHAR(20) NOT NULL,
    f_nac DATE,
```

```
    sexo VARCHAR(1),
    estado INT2 DEFAULT 1 NOT NULL,
    PRIMARY KEY (rude)
);
```

II.1.9.2.2.14 Tabla kardex_est

```
CREATE TABLE kardex_est (
    cod_ke SERIAL NOT NULL,
    comentario VARCHAR(500),
    rude INT8 NOT NULL,
    PRIMARY KEY (cod_ke)
);
```

II.1.9.2.2.15 Tabla reportes

```
CREATE TABLE reportes (
    cod_rep SERIAL NOT NULL,
    nombre VARCHAR(20) NOT NULL,
    ci VARCHAR(40) NOT NULL,
    PRIMARY KEY (cod_rep)
);
```

II.1.9.2.2.16 Tabla teléfono_adm

```
CREATE TABLE telefono_adm (
    cod_tel SERIAL NOT NULL,
    telefono INT8,
    ci VARCHAR(40) NOT NULL,
```

```
    PRIMARY KEY (cod_tel)
);
```

II.1.9.2.2.17 Tabla dirección_adm

```
CREATE TABLE direccion_adm (
    cod_dir SERIAL NOT NULL,
    zona VARCHAR(30) NOT NULL,
    calle VARCHAR(20) NOT NULL,
    numero VARCHAR(10),
    ci VARCHAR(40) NOT NULL,
    PRIMARY KEY (cod_dir) );
```

II.1.9.2.2.18 Tabla memorandus

```
CREATE TABLE memorandus (
    cod_memo SERIAL NOT NULL,
    nombre VARCHAR(40) NOT NULL,
    PRIMARY KEY (cod_memo)
);
```

II.1.9.2.2.19 Tabla kardex_adm

```
CREATE TABLE kardex_adm (
    cod_ka SERIAL NOT NULL,
    comentario VARCHAR(500) NOT NULL,
    ci VARCHAR(40) NOT NULL,
    PRIMARY KEY (cod_ka)
);
```

II.1.9.2.2.20 Tabla administrativos_memorandus

```
CREATE TABLE administrativos_memorandus (
    cod_memo INT4 NOT NULL,
    ci VARCHAR(40) NOT NULL,
    detalle VARCHAR(5000) NOT NULL,
    fecha DATE NOT NULL,
    PRIMARY KEY (cod_memo)
);
```

II.1.9.2.2.21 Tabla estudiante_memorandus

```
CREATE TABLE estudiante_memorandus (
    cod_memo INT4 NOT NULL,
    rude INT8 NOT NULL,
    detalle VARCHAR(500) NOT NULL,
    fecha DATE,
    PRIMARY KEY (cod_memo)
);
```

II.1.9.2.2.22 Tabla paralelo_materia

```
CREATE TABLE paralelo_materia (
    cod_m_p SERIAL NOT NULL,
    cod_mat INT4,
    gestion INT8 NOT NULL,
    cod_par INT4 NOT NULL,
    PRIMARY KEY (cod_m_p)
```

);

II.1.9.2.2.23 Tabla teléfono_est

```
CREATE TABLE telefono_est (
   cod_tel SERIAL NOT NULL,
   telefono VARCHAR(20) NOT NULL,
   rude INT8 NOT NULL,
   PRIMARY KEY (cod_tel)
);
```

II.1.9.2.2.24 Tabla dirección_est

```
CREATE TABLE direccion_est (
   cod_dir SERIAL NOT NULL,
   zona VARCHAR(30) NOT NULL,
   calle VARCHAR(30) NOT NULL,
   numero VARCHAR(10),
   rude INT8 NOT NULL,
   PRIMARY KEY (cod_dir)
);
```

II.1.9.2.3 Foreign Keys

```
ALTER TABLE registro
   ADD FOREIGN KEY (cod_rol) REFERENCES rol (cod_rol);
ALTER TABLE registro
   ADD FOREIGN KEY (cod_espec) REFERENCES especialidad (cod_espec);
ALTER TABLE registro
```

```
    ADD FOREIGN KEY (ci) REFERENCES administrativo (ci);
ALTER TABLE paralelo
    ADD FOREIGN KEY (cod_cur) REFERENCES curso (cod_cur);
ALTER TABLE administrar_horario
    ADD FOREIGN KEY (cod_hor) REFERENCES hora (cod_hor);
ALTER TABLE administrar_horario
    ADD FOREIGN KEY (cod_dia) REFERENCES dia (cod_dia);
ALTER TABLE administrar_horario
    ADD FOREIGN KEY (ci) REFERENCES administrativo (ci);
ALTER TABLE administrar_horario
    ADD FOREIGN KEY (cod_m_p) REFERENCES paralelo_materia (cod_m_p);
ALTER TABLE inscripcion
    ADD FOREIGN KEY (rude) REFERENCES estudiante (rude);
ALTER TABLE inscripcion
    ADD FOREIGN KEY (cod_adm_hor) REFERENCES administrar_horario
(cod_adm_hor);
ALTER TABLE datos
    ADD FOREIGN KEY (ci) REFERENCES administrativo (ci);
ALTER TABLE kardex_est
    ADD FOREIGN KEY (rude) REFERENCES estudiante (rude);
ALTER TABLE reportes
    ADD FOREIGN KEY (ci) REFERENCES administrativo (ci);
ALTER TABLE telefono_adm
    ADD FOREIGN KEY (ci) REFERENCES administrativo (ci);
```

```
ALTER TABLE direccion_adm
   ADD FOREIGN KEY (ci) REFERENCES administrativo (ci);
ALTER TABLE kardex_adm
   ADD FOREIGN KEY (ci) REFERENCES administrativo (ci);
ALTER TABLE administrativos_memorandus
   ADD FOREIGN KEY (cod_memo) REFERENCES memorandus (cod_memo);
ALTER TABLE administrativos_memorandus
   ADD FOREIGN KEY (ci) REFERENCES administrativo (ci);
ALTER TABLE estudiante_memorandus
   ADD FOREIGN KEY (rude) REFERENCES estudiante (rude);
ALTER TABLE estudiante_memorandus
   ADD FOREIGN KEY (cod_memo) REFERENCES memorandus (cod_memo);
ALTER TABLE paralelo_materia
   ADD FOREIGN KEY (cod_mat) REFERENCES materia (cod_mat);
ALTER TABLE paralelo_materia
   ADD FOREIGN KEY (cod_par) REFERENCES paralelo (cod_par);
ALTER TABLE telefono_est
   ADD FOREIGN KEY (rude) REFERENCES estudiante (rude);
ALTER TABLE direccion_est
   ADD FOREIGN KEY (rude) REFERENCES estudiante (rude);
```

II.1.9.2.3 Indexes

```
CREATE UNIQUE INDEX IDX_rol1 ON rol (cod_rol);

CREATE UNIQUE INDEX IDX_especialidad1 ON especialidad (cod_espec);

CREATE INDEX IDX_registro1 ON registro (cod_rol);

CREATE INDEX IDX_registro2 ON registro (cod_espec);

CREATE INDEX IDX_registro3 ON registro (ci);

CREATE INDEX IDX_paralelo1 ON paralelo (cod_cur);

CREATE UNIQUE INDEX IDX_paralelo2 ON paralelo (cod_par);

CREATE UNIQUE INDEX IDX_curso1 ON curso (cod_cur);

CREATE UNIQUE INDEX IDX_materia1 ON materia (cod_mat);

CREATE UNIQUE INDEX IDX_hora1 ON hora (cod_hor);

CREATE UNIQUE INDEX IDX_dia1 ON dia (cod_dia);

CREATE INDEX IDX_administrar_horario1 ON administrar_horario (cod_hor);

CREATE INDEX IDX_administrar_horario2 ON administrar_horario (cod_dia);

CREATE INDEX IDX_administrar_horario3 ON administrar_horario (ci);

CREATE INDEX IDX_administrar_horario4 ON administrar_horario (cod_m_p);

CREATE UNIQUE INDEX IDX_administrar_horario5 ON administrar_horario
(cod_adm_hor);

CREATE INDEX IDX_inscripcion1 ON inscripcion (rude);

CREATE INDEX IDX_inscripcion2 ON inscripcion (cod_adm_hor);

CREATE INDEX IDX_datos1 ON datos (ci);

CREATE UNIQUE INDEX IDX_administrativo1 ON administrativo (ci);

CREATE UNIQUE INDEX IDX_estudiante1 ON estudiante (rude);

CREATE INDEX IDX_kardex_est1 ON kardex_est (rude);

CREATE INDEX IDX_reportes1 ON reportes (ci);
```

```
CREATE INDEX IDX_telefono_adm1 ON telefono_adm (ci);

CREATE INDEX IDX_direccion_adm1 ON direccion_adm (ci);

CREATE UNIQUE INDEX IDX_memorandus1 ON memorandus (cod_memo);

CREATE INDEX IDX_kardex_adm1 ON kardex_adm (ci);

CREATE INDEX IDX_administrativos_memorandus1 ON administrativos_memorandus (cod_memo);

CREATE INDEX IDX_administrativos_memorandus2 ON administrativos_memorandus (ci);

CREATE INDEX IDX_estudiante_memorandus1 ON estudiante_memorandus (rude);

CREATE INDEX IDX_estudiante_memorandus2 ON estudiante_memorandus (cod_memo);

CREATE INDEX IDX_paralelo_materia1 ON paralelo_materia (cod_mat);

CREATE INDEX IDX_paralelo_materia2 ON paralelo_materia (cod_par);

CREATE UNIQUE INDEX IDX_paralelo_materia3 ON paralelo_materia (cod_m_p);

CREATE INDEX IDX_telefono_est1 ON telefono_est (rude);

CREATE INDEX IDX_direccion_est1 ON direccion_est (rude);
```

II.1.10 DISEÑO DE PANTALLAS DE USUARIO

II.1.10.1 Introducción

Trata de diseños que permiten al usuario tener una idea sobre las interfaces que proveerá el sistema.

II.1.10.1.1 Propósito

Presentar los diseños de pantallas para que el usuario tenga idea de la interfaz que le presenta el sistema.

II.1.10.1.2 Alcance

Mostrar los diseños de pantallas, sujeto a modificaciones a lo largo del desarrollo del sistema

II.1.10.2 Diagrama Navegacional

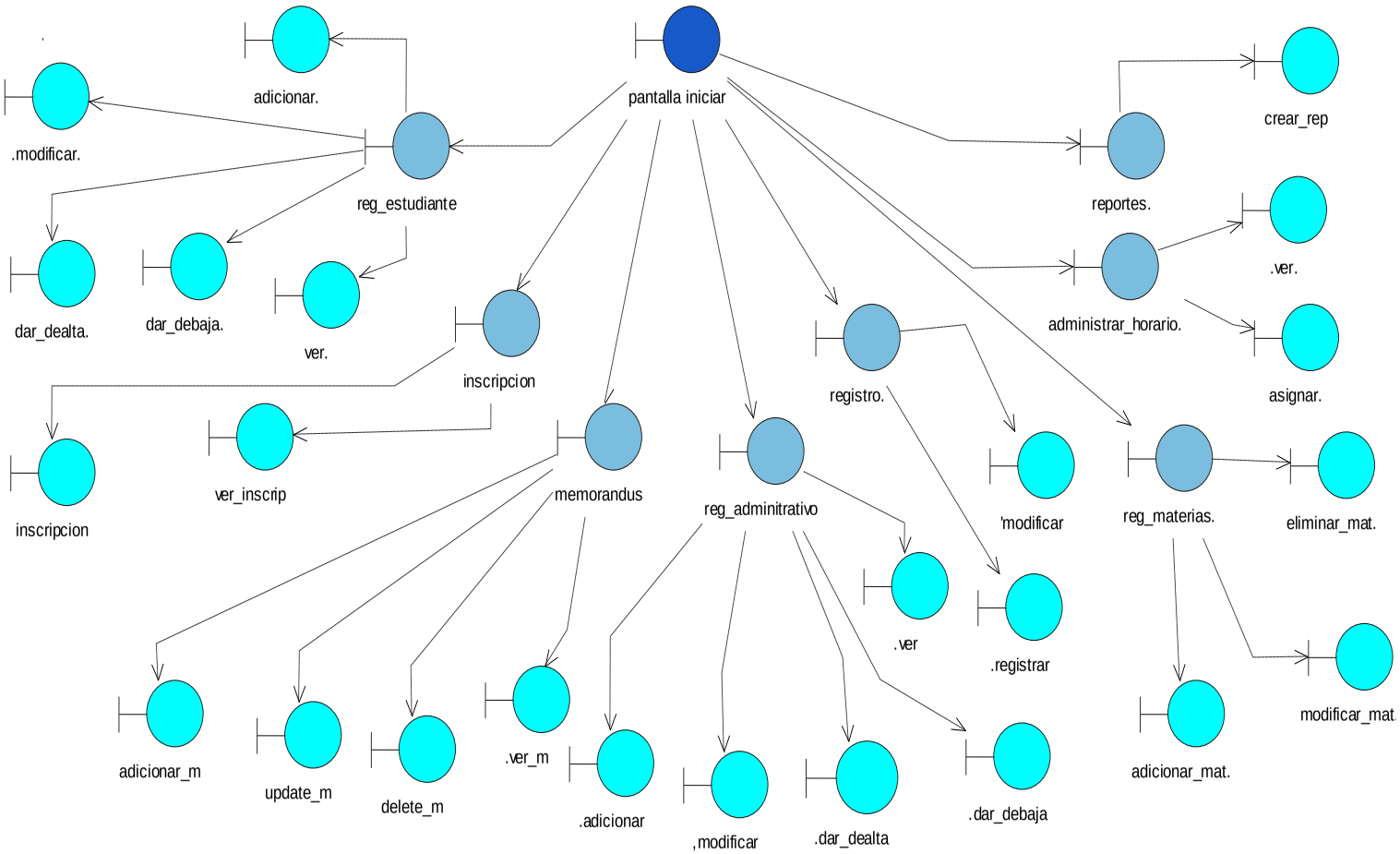

adicionar.
.modificar.
pantalla iniciar
crear_rep
reg_estudiante
reportes.
dar_dealta.
dar_debaja.
ver.
.ver.
administrar_horario.
inscripcion
registro.
asignar.
inscripcion
ver_inscrip
memorandus
reg_adminitrativo
'modificar
reg_materias.
eliminar_mat.
.ver
.registrar
adicionar_m
update_m
delete_m
.ver_m
.adicionar
,modificar
.dar_dealta
.dar_debaja
adicionar_mat.
modificar_mat.

Figura 128. Diagrama Navegacional

II.1.10.3 Pantallas

II.1.10.3.1 PANTALLA INICIAR

Figura 129.Pantalla iniciar

Figura 130.Pantalla Principal

II.1.10.3.2 PANTALLA REG_ESTUDIANTE

Figura 131. Pantalla Reg_estudiantes

II.1.10.3.3 ADICIONAR

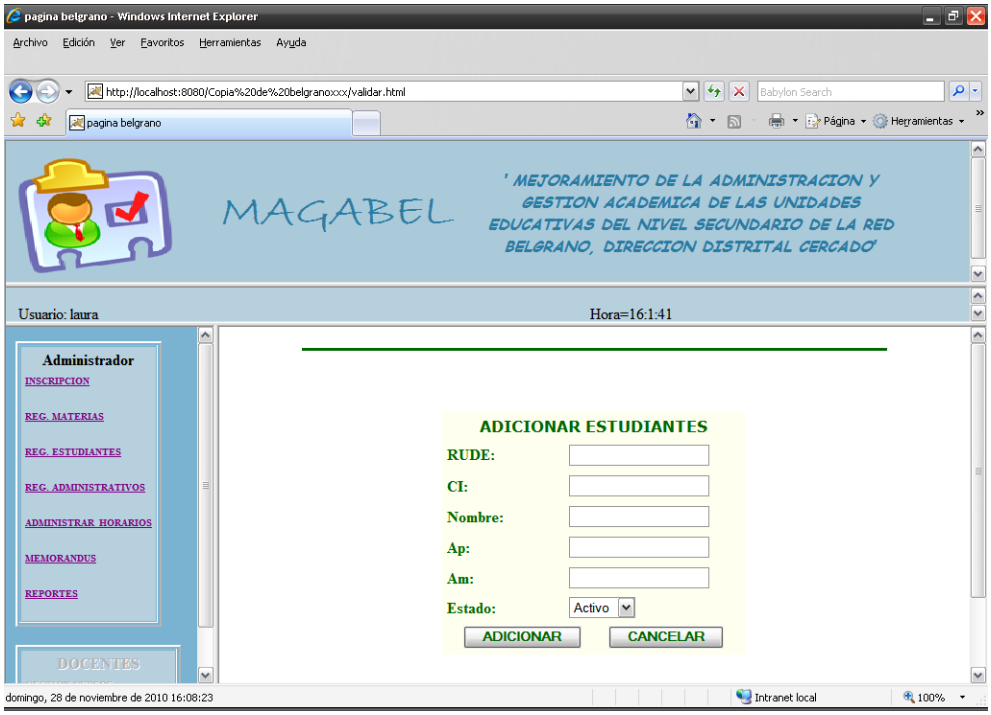


Figura 132. Pantalla adicionar

II.1.10.3.4 MODIFICAR

Figura 133. Pantalla Modificar

II.1.10.3.5 DAR DE BAJA

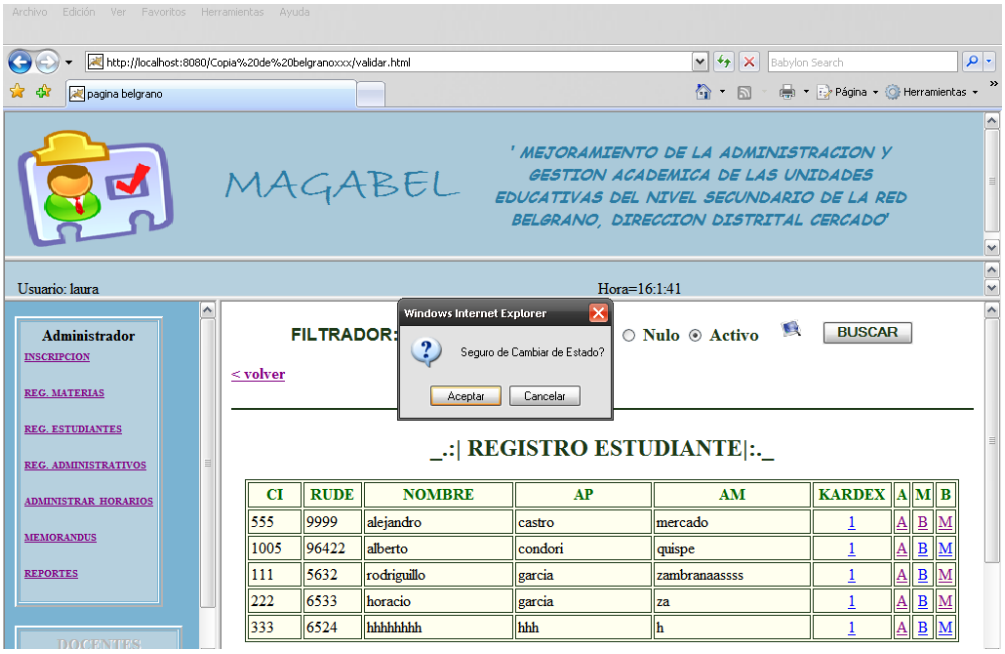


Figura 134. Pantalla dar_debaja

II.1.10.3.6 DAR DE ALTA

Figura 135. Pantalla dar_dealta

II.1.10.3.7 VER

Figura 136. Pantalla ver

II.1.10.3.8 INSCRIPCION

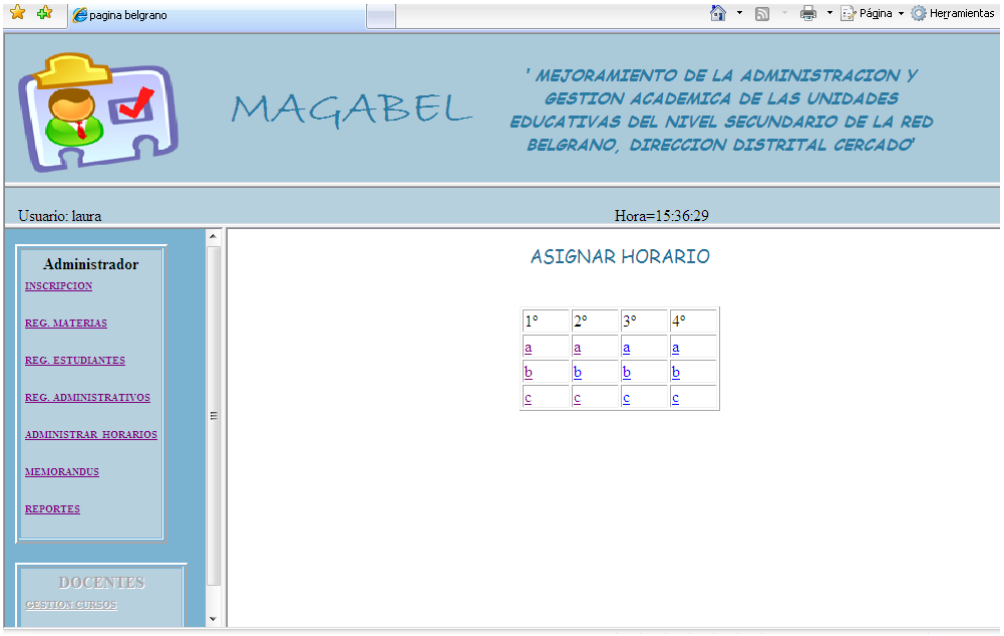


Figura 137. Pantalla inscripcion

II.1.10.3.9 INSCRIPCION

Figura 138. Pantalla inscripcion

II.1.10.3.10 VER INSCRIP

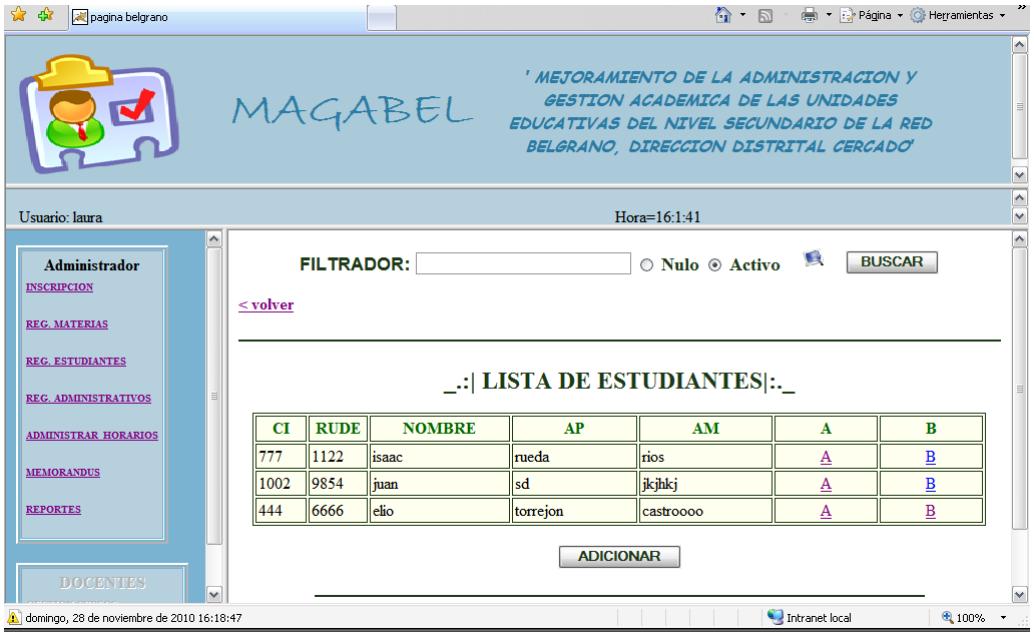


Figura 139. Pantalla ver_inscripcion

II.1.10.3.11 MEMORANDUS

Figura 140. Pantalla memorandus

II.1.10.3.12 ADICIONAR_M

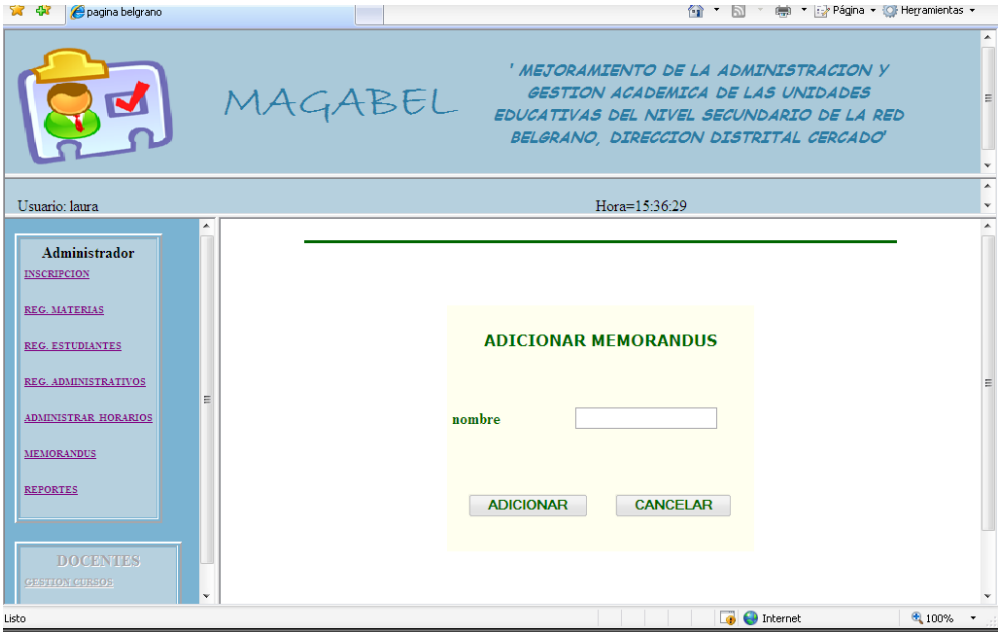


Figura 141. Pantalla adicionar_m

II.1.10.3.13 MODIFICAR_M

Figura 142. Pantalla modificar_m

II.1.10.3.14 ELIMINAR_M

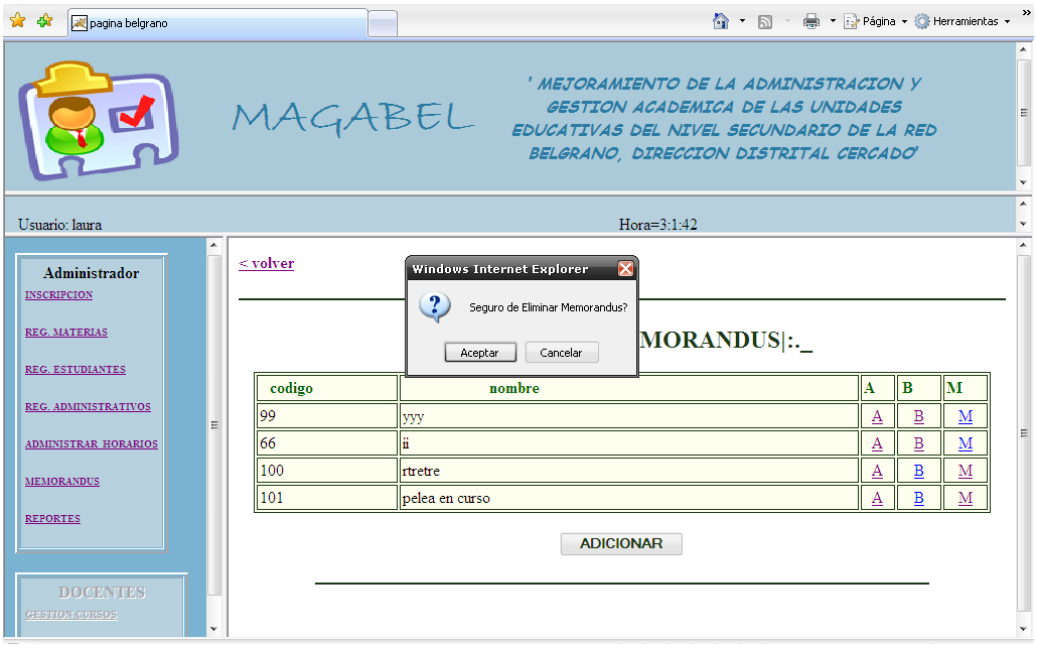


Figura 143. Pantalla eliminar_m

II.1.10.3.15 VER_M

Figura 144. Pantalla ver_m

II.1.10.3.16 REG_ADMINISTRATIVO

Figura 145. Pantalla Reg_administrativo

II.1.10.3.17 ADICIONAR

Figura 146. Pantalla Adicionar

II.1.10.3.18 MODIFICAR

Figura 147. Pantalla Modificar

II.1.10.3.19 DAR_DEBAJA

Figura 148. Pantalla Dar_debaja

II.1.10.3.20 DAR_DEALTA

Figura 148. Pantalla Dar_dealta

II.1.10.3.21 VER

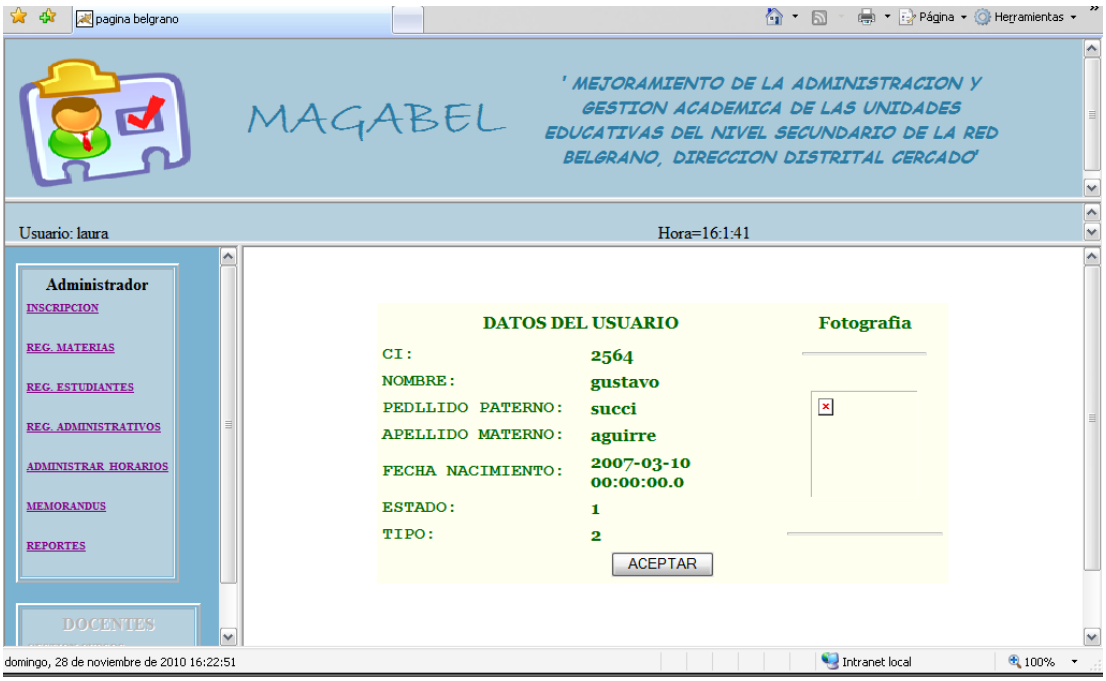


Figura 149. Pantalla Ver

II.1.10.3.22 REG_MATERIAS

Figura 150. Pantalla Reg_materias

II.1.10.3.23 ADICIONAR_MAT

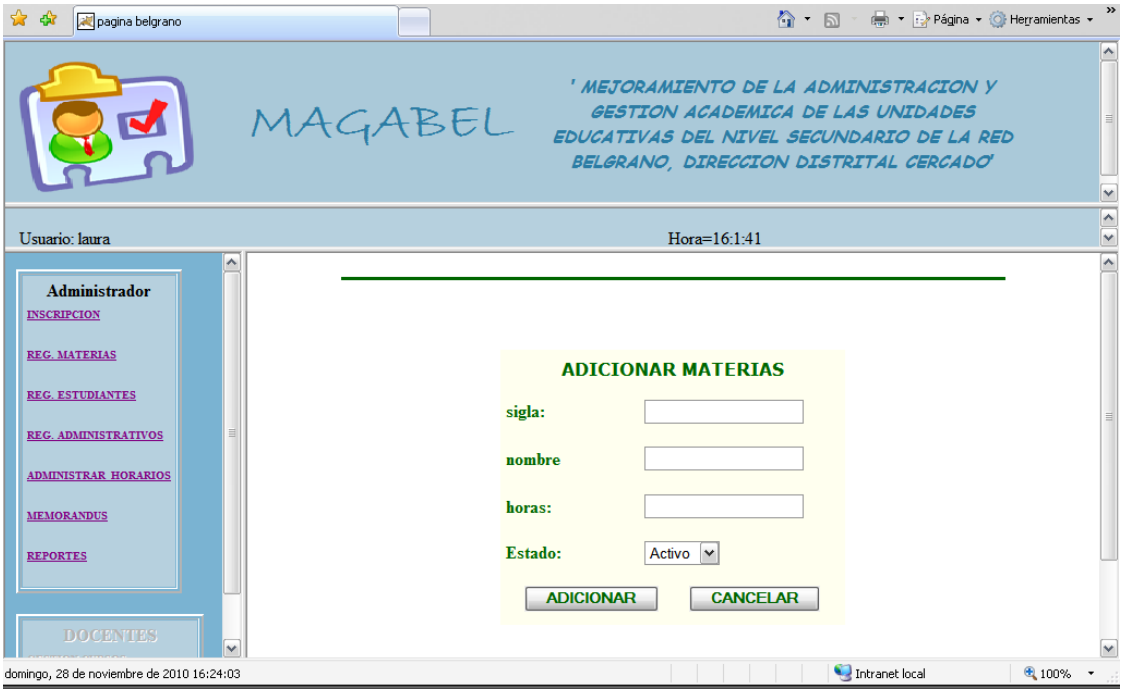


Figura 151. Pantalla Adicionar_mat

II.1.10.3.24 MODIFICAR_MAT

Figura 152. Pantalla Modificar_mat

II.1.10.3.25 ELIMINAR_ MAT

Figura 153. Pantalla Eliminar_mat

II.1.10.3.26 ADMINISTRAR HORARIO

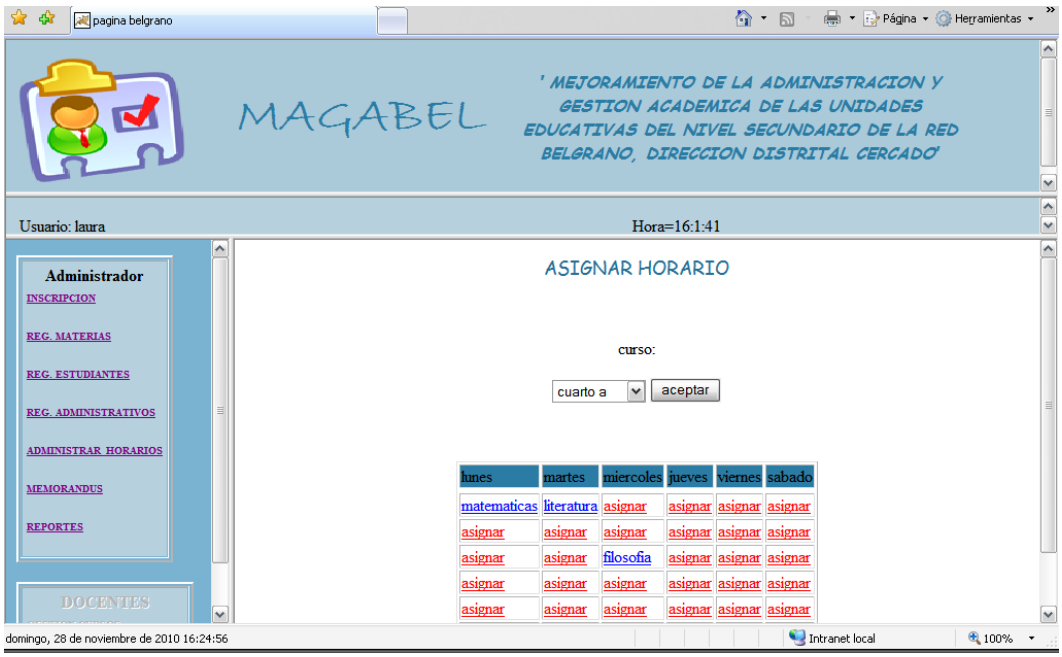


Figura 154. Pantalla administrar_horario

II.1.10.3.27 ASIGNAR

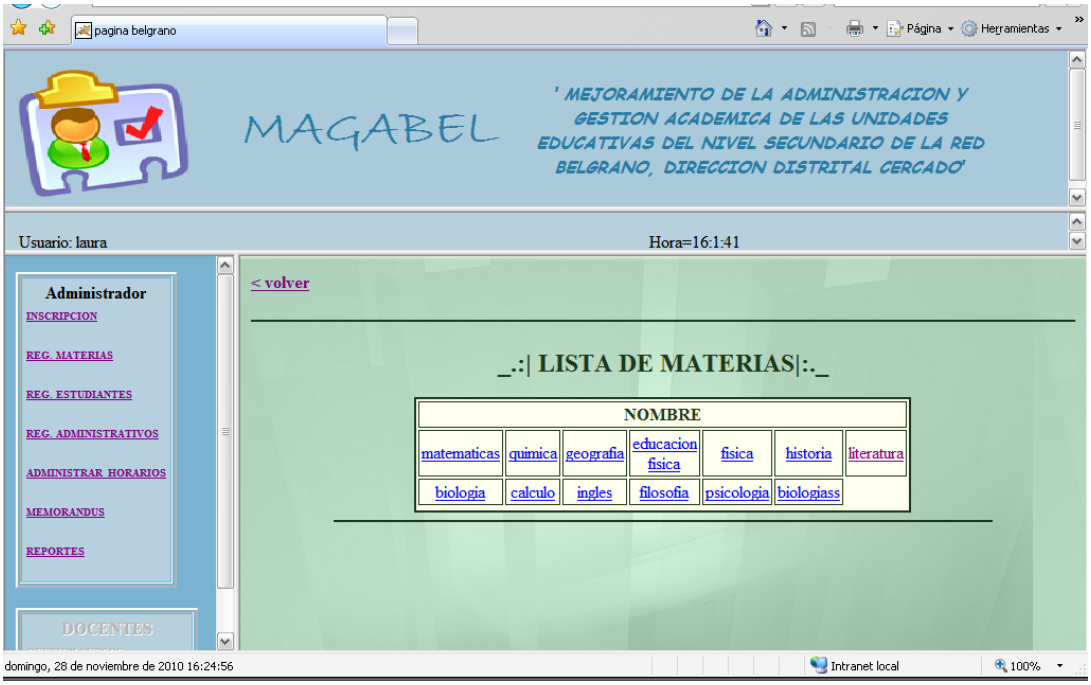


Figura 155. Pantalla asignar

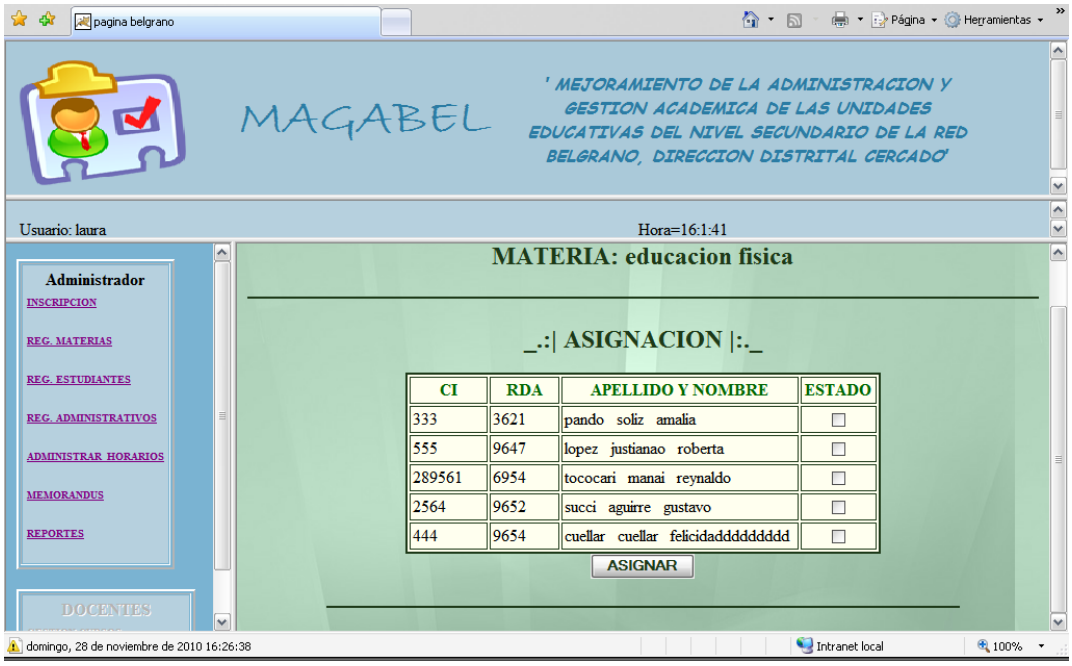


Figura 156. Pantalla asignar 2

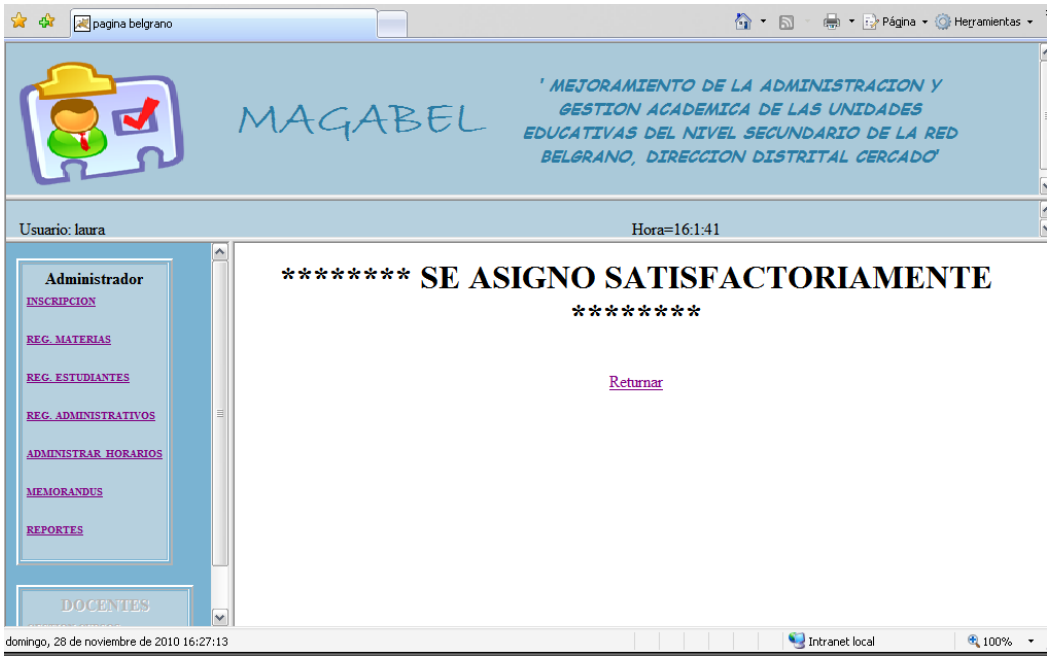


Figura 157. Pantalla asignar

II.1.10.3.28 VER

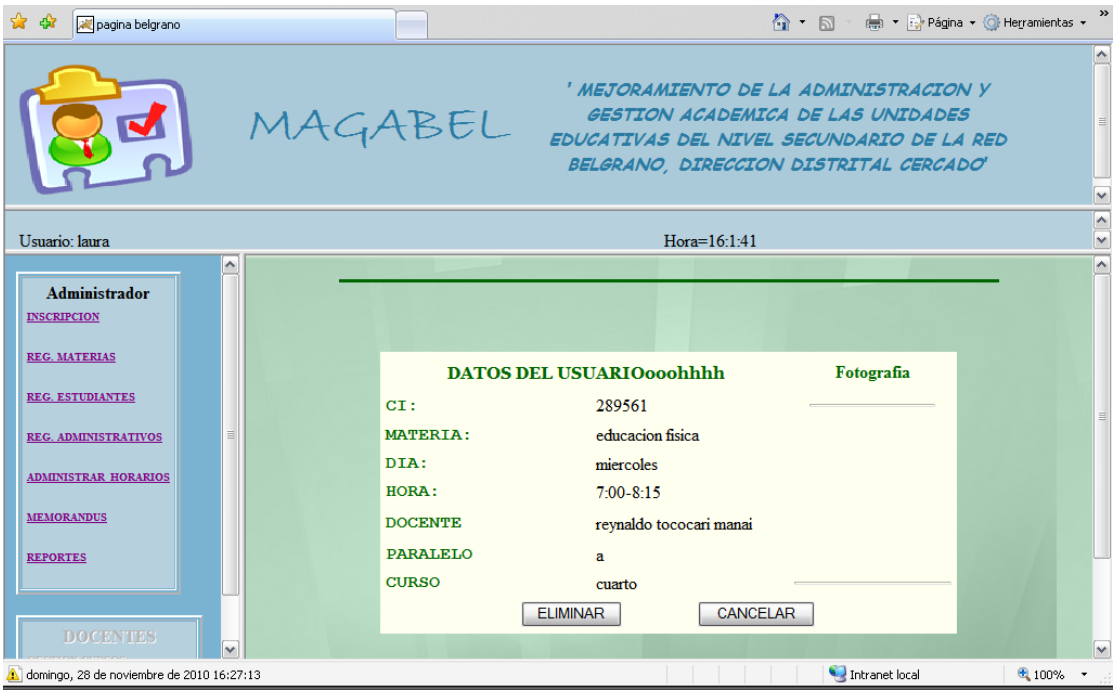


Figura 158. Pantalla Ver

II.1.11 MODELO DE IMPLEMENTACION

II.1.11.1 Introducción

Este modelo es una colección de componentes y subsistemas que los contienen. Estos componentes incluyen ficheros ejecutables, ficheros de código fuente y de tipo accesorios para implantación y despliegue del sistema

II.1.11.2 Diagrama de Componentes

Sistema de Seguimiento de Documentación, lenguaje de desarrollo: java, los accesos a la información son a través de la base de datos.

II.1.11.2.1 Diagrama de Componentes Iniciar

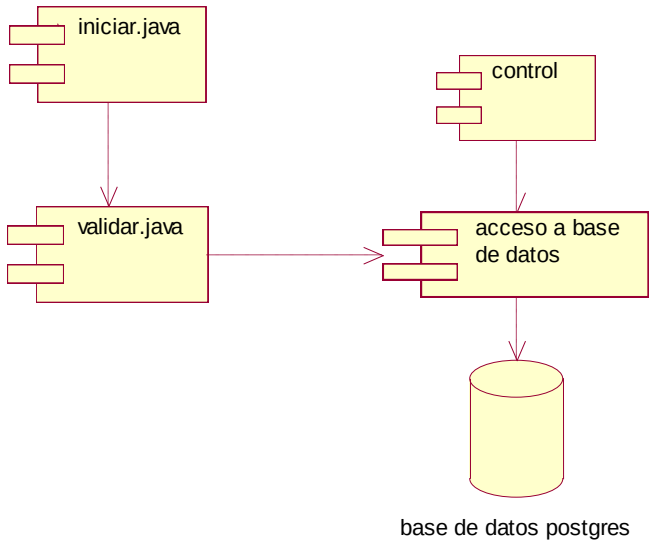


Figura 159. D. Componentes Iniciar

II.1.11.2.2 Diagrama de Componentes Pantalla Principal

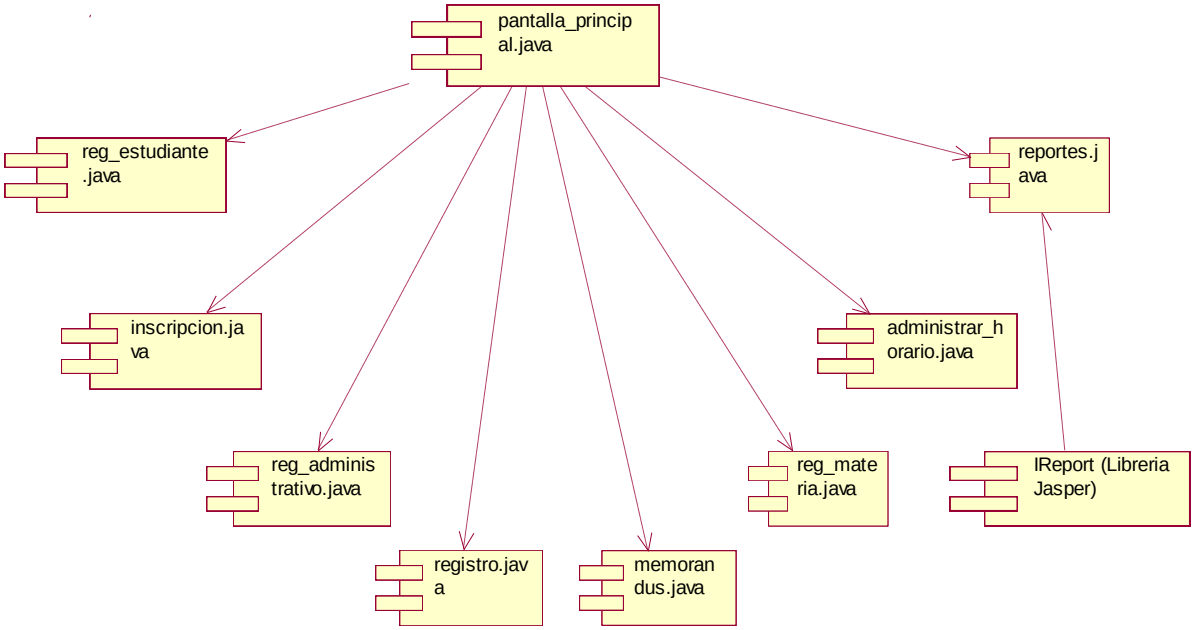


Figura 160. D. Componentes Pantalla Principal

II.1.11.2.3 Diagrama de Componentes reg_estudiante

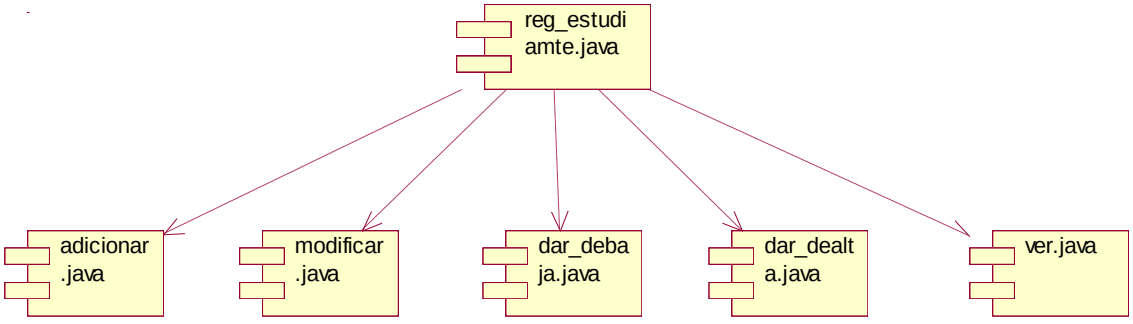


Figura 161. D. Componentes reg_estudiante

II.1.11.2.4 Diagrama de Componentes inscripción

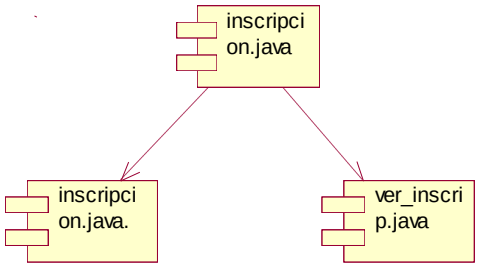


Figura 162. D. Componentes inscripción

II.1.11.2.5 Diagrama de Componentes Memorandus

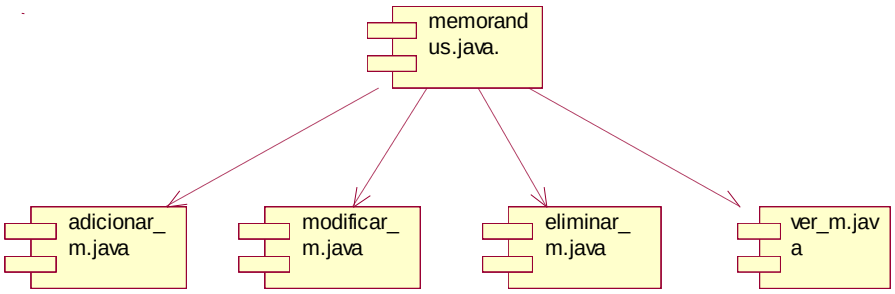


Figura 163. D. Componentes memorandus

II.1.11.2.6 Diagrama de Componentes reg_administrativo

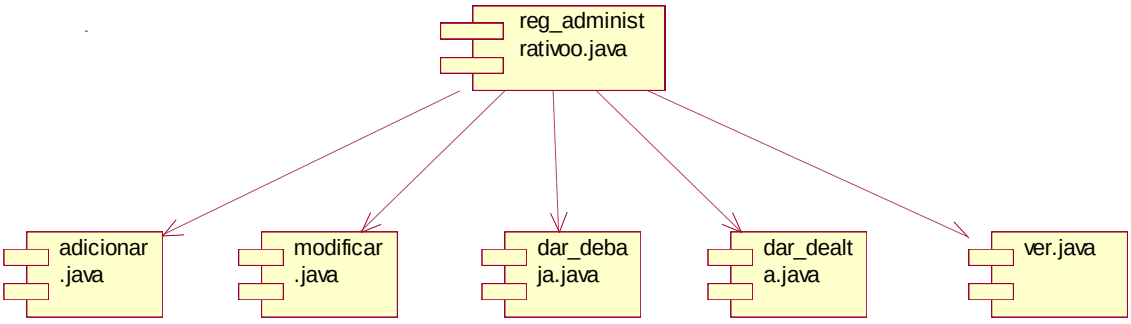


Figura 164. D. Componentes reg_administrativos

II.1.11.2.7 Diagrama de Componentes Registro

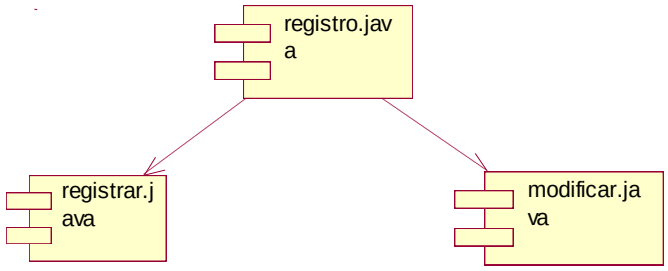


Figura 165. D. Componentes registro

II.1.11.2.8 Diagrama de Componentes reg_materias

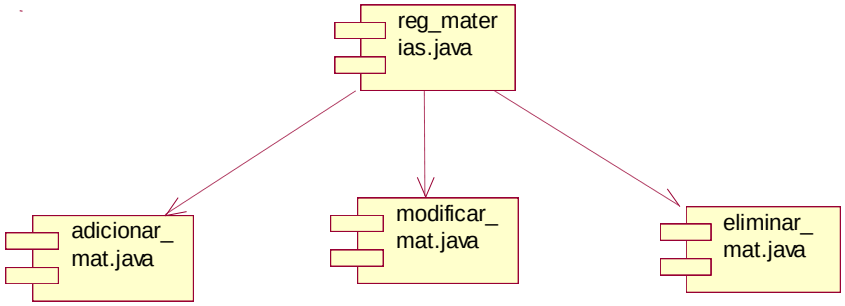


Figura 166. D. Componentes reg_materias

II.1.11.2.9 Diagrama de Componentes administrar_horario

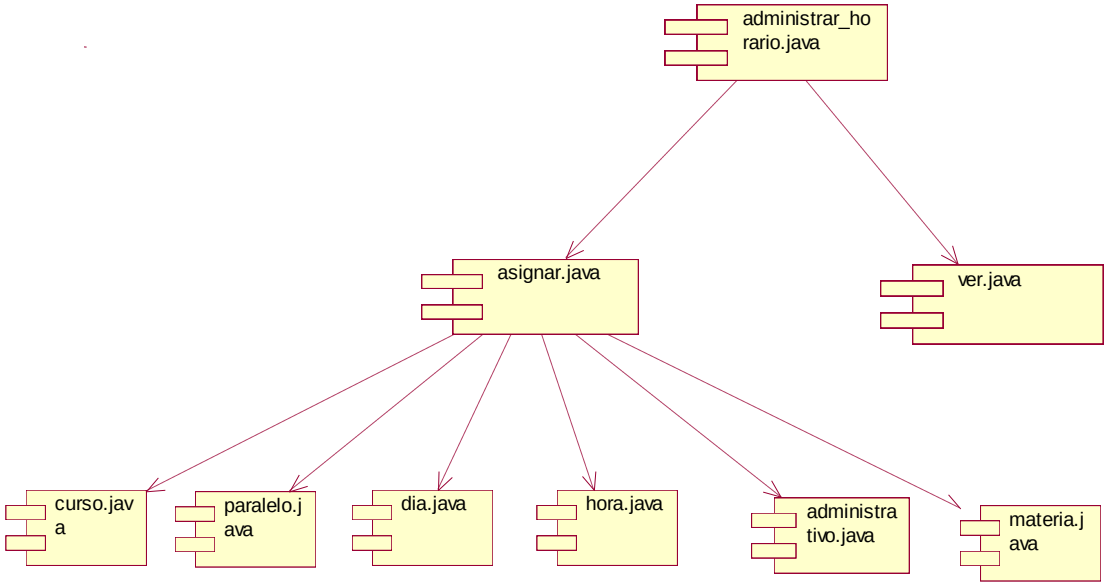


Figura 167. D. Componentes administrar_horario

II.1.12 MODELO DE DESPLIEGUE

II.1.12.1 Introducción

El modelo de Despliegue provee un modelo detallado de la forma en la que los componentes se desplegaran a lo largo de la infraestructura del sistema.

II.1.12.2 Propósito

Mostrar la disposición física de los elementos que intervienen en el Administrador

II.1.12.2 Diagrama de Despliegue

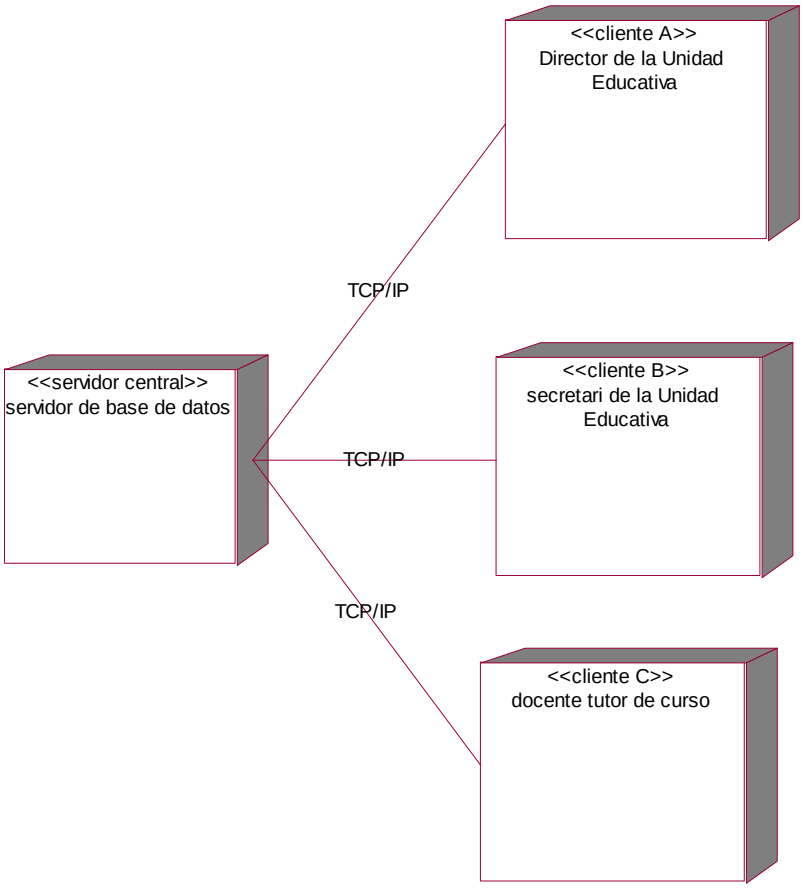


Figura 168. D. Despliegue

II.1.13 ESPECIFICACION DE METODOS

II.1.13.1 Introducción

Las especificaciones de métodos describen el comportamiento de un procedimiento en términos de sus precondiciones y pos condiciones, también referirse solamente a los campos de especificación, a los argumentos del método y a las variables globales, pero nunca a procedimientos concretos.

II.1.13.1.1 Propósito

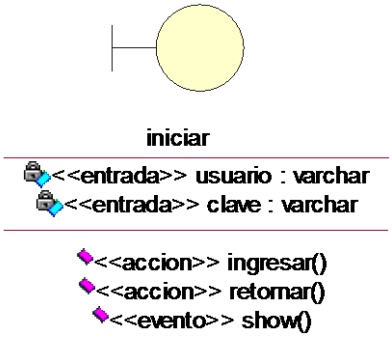
- Mostrar el código fuente concluido
- Comprender la estructura de la programacion

II.1.13.1.2 Alcance

- Se describen solo los métodos que implican la navegación en una pantalla específica.
- Identificar los algoritmos de los procedimientos.

II.1.13.2 Especificación de Métodos

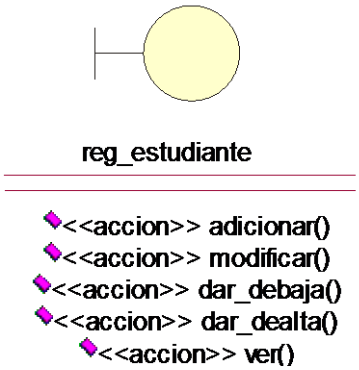
II.1.13.2.1 INICIAR




```
import javax.servlet.ServletException;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;
```

```
import org.springframework.beans.factory.InitializingBean;
import org.springframework.web.servlet.ModelAndView;
import org.springframework.web.servlet.mvc.multiaction.MultiActionController;
public class iniciar extends MultiActionController implements InitializingBean{
//	Metodo que siempre debe estar
	public void afterPropertiesSet() throws Exception {
		// TODO Auto-generated method stub
	}
//Constructor clase uno
	public iniciar() {
		// TODO Auto-generated constructor stub
	}
//Lista de materias
	public ModelAndView index(HttpServletRequest
request,HttpServletResponse response) throws Exception,ServletException {
			return new ModelAndView("usuarios/inicio");
   }
	}
```

II.1.13.2.2 REG ESTUDIANTES



Controlador

```
public ModelAndView reg_estudiantes(HttpServletRequest request,HttpServletResponse response) throws Exception,ServletException {
        Map p = new HashMap();
        String xy=new String();
        ArrayList milistax = new ArrayList();
        Map mapx = new HashMap();
        String datofiltro="";
        String estado="1";
        int rango=1;
        String dato=request.getParameter("xdatofiltro");
        //System.out.println("este es el dato"+ request.getParameter("xdatofiltro"));
            if(request.getParameter("xdatofiltro")==null){
                datofiltro="";
            }else{
                datofiltro=request.getParameter("xdatofiltro");
```

```
                }
                if(request.getParameter("xinter")==null){
                        rango=1;
                }else{
                        rango=Integer.parseInt(request.getParameter("xinter"));
                }
                if(request.getParameter("xopcion")==null){
                        estado="1";
                }else{
                        estado=request.getParameter("xopcion");
                }
        try {
        String cod_parx;
                List Total=null;
                if(request.getParameter("cod_par")==null &&
request.getParameter("cod_par2")==null ){
                        Total =
this.estudianteManager.getListaCursosFiltroTotal(datofiltro,Integer.parseInt(estado))
;
                }
                if(request.getParameter("cod_par")!=null){
System.out.println(" veamosss che "+request.getParameter("cod_par"));
```

```
                    Total =
this.estudianteManager.getListaCursosFiltroTotalxy(datofiltro,Integer.parseInt(estado),Integer.parseInt(request.getParameter("cod_par")));
                }
                    if(request.getParameter("cod_par2")!=null){
                    System.out.println(" veamosss che cheee "+request.getParameter("cod_par2"));
                    Total =
this.estudianteManager.getListaCursosFiltroTotalxyxy(datofiltro,Integer.parseInt(estado));
                    }
                    double num=Total.size();
                    double xx=Math.ceil(num / 5);
                    if(rango>1){
                    mapx = new HashMap();
                    if(request.getParameter("cod_par")==null && request.getParameter("cod_par2")==null ){
                        mapx.put("rango","<a id='x' onMouseOver='Over(this.id)' onMouseOut='Out(this.id)' href='reg_estudiantes.html?xdatofiltro="+datofiltro+"&xinter="+(rango-1)+"&xopcion="+estado+"'>"+"<<<"+"</a>");
                    }
if(request.getParameter("cod_par")!=null){
mapx.put("rango","<a id='x' onMouseOver='Over(this.id)' onMouseOut='Out(this.id)' href='reg_estudiantes.html?cod_par="+request.getParameter("cod_par")+"&xdatofiltro="+datofiltro+"&xinter="+(rango-1)+"&xopcion="+estado+"'>"+"<<<"+"</a>");
```

```
                }
                if(request.getParameter("cod_par2")!=null){
                        mapx.put("rango","<a id='x'
onMouseOver='Over(this.id)' onMouseOut='Out(this.id)'
href='reg_estudiantes.html?cod_par2="+request.getParameter("cod_par2")+"&xdatof
iltro="+datofiltro+"&xinter="+(rango-1)+"&xopcion="+estado+"'>"+"<<<"+"</a>");
                }
            milistax.add(mapx);
            }
            for(int i=1;i<=xx;i++){
                mapx = new HashMap();
                if(i==rango){
                        mapx.put("rango",""+i+"");
                }else{
                        if(request.getParameter("cod_par")==null &&
request.getParameter("cod_par2")==null ){
                                mapx.put("rango","<a id='"+i+"'
onMouseOver='Over(this.id)' onMouseOut='Out(this.id)'
href='reg_estudiantes.html?xdatofiltro="+datofiltro+"&xinter="+i+"&xopcion="+est
ado+"'>"+i+"</a>");                             }
if(request.getParameter("cod_par")!=null){
                                mapx.put("rango","<a id='"+i+"'
onMouseOver='Over(this.id)' onMouseOut='Out(this.id)'
href='reg_estudiantes.html?cod_par="+request.getParameter("cod_par")+"&xdatofiltr
o="+datofiltro+"&xinter="+i+"&xopcion="+estado+"'>"+i+"</a>");
                        }
```

```
                    if(request.getParameter("cod_par2")!=null){
                        mapx.put("rango","<a id='"+i+"'
onMouseOver='Over(this.id)' onMouseOut='Out(this.id)'
href='reg_estudiantes.html?cod_par2="+request.getParameter("cod_par2")+"&xdatof
iltro="+datofiltro+"&xinter="+i+"&xopcion="+estado+"'>"+i+"</a>");
                    }
                                        }
                milistax.add(mapx);
            }
            if(rango<xx){
                mapx = new HashMap();
                if(request.getParameter("cod_par")==null &&
request.getParameter("cod_par2")==null ){
                    mapx.put("rango","<a id='y'
onMouseOver='Over(this.id)' onMouseOut='Out(this.id)'
href='reg_estudiantes.html?xdatofiltro="+datofiltro+"&xinter="+(rango+1)+"&xopci
on="+estado+"'>"+">>>"+"</a>");
                }
if(request.getParameter("cod_par")!=null){
                    mapx.put("rango","<a id='y'
onMouseOver='Over(this.id)' onMouseOut='Out(this.id)'
href='reg_estudiantes.html?cod_par="+request.getParameter("cod_par")+"&xdatofiltr
o="+datofiltro+"&xinter="+(rango+1)+"&xopcion="+estado+"'>"+">>>"+"</a>");
                }
                if(request.getParameter("cod_par2")!=null){
```

```
                                    mapx.put("rango","<a id='y'
onMouseOver='Over(this.id)' onMouseOut='Out(this.id)'
href='reg_estudiantes.html?cod_par2="+request.getParameter("cod_par2")+"&xdatof
iltro="+datofiltro+"&xinter="+(rango+1)+"&xopcion="+estado+"'>"+">>>"+"</a>")
;
                        }
                        milistax.add(mapx);
                }
                List mat = null;
                if(request.getParameter("cod_par")==null &&
request.getParameter("cod_par2")==null){
                        mat =
this.estudianteManager.getListaCursosFiltro(datofiltro,rango,Integer.parseInt(estado)
);
                }
                if(request.getParameter("cod_par")!=null){
                        mat =
this.estudianteManager.getListaCursosFiltroxy(datofiltro,rango,Integer.parseInt(esta
do),Integer.parseInt(request.getParameter("cod_par")));                 }
if(request.getParameter("cod_par2")!=null){
                                    mat =
this.estudianteManager.getListaCursosFiltroxyxy(datofiltro,rango,Integer.parseInt(es
tado));
                }
                p.put("datos", mat);
                p.put("intervalo",milistax);
```

```
                } catch(Exception s){System.out.println("MENSAJE error aqui
"+s.getLocalizedMessage());}
                if(estado.equals("1")){
                        p.put("xestadox"," ");
                        p.put("yestadoy","checked");
                }else{
                        p.put("xestadox","checked");
                        p.put("yestadoy"," ");
                }
                p.put("datofiltradox",datofiltro);
                if(request.getParameter("cod_par")==null &&
request.getParameter("cod_par2")==null){
                        xy="estudiantes/reg_estudiantes";
                }
                if(request.getParameter("cod_par")!=null){
                xy="administrador/reg_estudiantes";
p.put("cod_par",request.getParameter("cod_par"));
                }
                if(request.getParameter("cod_par2")!=null){
                        xy="administrador/reg_estudiantes2";
                        //p.put("volver",);
                        p.put("cod_par2",request.getParameter("cod_par2"));
                }
            return new ModelAndView(xy,p);
```

```
    }
```

EstudianteManagerImpl

```
import java.util.List;
import model.domain.Administrativo;
import model.domain.Estudiante;
import org.hibernate.Session;
import org.hibernate.SessionFactory;


public class EstudianteManagerImpl implements EstudianteManager{
SessionFactory sessionFactory;
        public SessionFactory getSessionFactory()    {
        return sessionFactory;    }
    public void setSessionFactory(SessionFactory sessionFactory)
    {        this.sessionFactory = sessionFactory;
    }
        public List getListaCursosFiltroTotal(String datofiltro,int estado) {
                        Session session = this.sessionFactory.getCurrentSession();
                        return session.createQuery("from Estudiante where
(estado="+estado+") and upper(ap) like upper('"+datofiltro+"%') order by ap
asc").list();
            }
    public List getListaCursosFiltroTotalxy(String datofiltro,int estado,int cod_par) {
                        Session session = this.sessionFactory.getCurrentSession();
```

```
        return session.createQuery("from Estudiante where rude in(select rude
from Inscripcion where cod_par='"+cod_par+"') and (estado="+estado+") and
upper(ap) like upper('"+datofiltro+"%') order by ap asc").list();
    }
  public List getListaCursosFiltroTotalxyxy(String datofiltro,int estado) {

        Session session = this.sessionFactory.getCurrentSession();
        return session.createQuery("from Estudiante where rude not in(select
rude from Inscripcion ) and (estado="+estado+") and upper(ap) like
upper('"+datofiltro+"%') order by ap asc").list();
    }
  public List getListaCursosFiltro(String datofiltro,int i,int estado) {
        Session session = this.sessionFactory.getCurrentSession();
      return session.createQuery("from Estudiante where (estado="+estado+") and
upper(ap) like upper('"+datofiltro+"%') order by ap asc").setFirstResult((i-
1)*5).setMaxResults(5).list();
    }
  public List getListaCursosFiltroxy(String datofiltro,int i,int estado,int cod_par) {
        Session session = this.sessionFactory.getCurrentSession();
        return session.createQuery("from Estudiante where rude in(select rude
from Inscripcion where cod_par='"+cod_par+"') and (estado="+estado+") and
upper(ap) like upper('"+datofiltro+"%') order by ap asc").setFirstResult((i-
1)*5).setMaxResults(5).list();
    }
  public List getListaCursosFiltroxyxy(String datofiltro,int i,int estado) {
```

```
                Session session = this.sessionFactory.getCurrentSession();
                return session.createQuery("from Estudiante where rude not in(select
rude from Inscripcion ) and (estado="+estado+") and upper(ap) like
upper('"+datofiltro+"%') order by ap asc").setFirstResult((i-
1)*5).setMaxResults(5).list();
        }
    public Estudiante getEstudiante (int xrude){
        Session session = this.sessionFactory.getCurrentSession();
        return (Estudiante) session.get(Estudiante.class, new Integer(xrude));
    }
    public List getEstudiante2(){
        Session session=this.sessionFactory.getCurrentSession();
        return session.createQuery("from Estudiante").list();    }
    public void saveEstudiante(Estudiante m){
        Session session = this.sessionFactory.getCurrentSession();
       session.merge(m);    }
    public void updateEstudiante(Estudiante m){
        Session session = this.sesionFactory.getCurrentSession();
        session.update(m);    }
 /*   public void deleteEstudiante(int rude){
        Session session = this.sessionFactory.getCurrentSession();
        session.getTransaction().begin();
        session.delete(session.load(Estudiante.class, new Integer(rude)));
        session.getTransaction().commit();
```

}*/ }

II.1.13.2.3 ADICIONAR

controlador

```
public ModelAndView addestudiante(HttpServletRequest
request,HttpServletResponse response) throws Exception,ServletException {
        Map p = new HashMap();
        return new ModelAndView("estudiantes/addestudiante",p);
    }
        public ModelAndView addestudiante2(HttpServletRequest
request,HttpServletResponse response) throws Exception,ServletException {
                Map p = new HashMap();
                try {
                        Estudiante m = new Estudiante();
                        m.setRude(Integer.parseInt(request.getParameter("xrude")));
```

m.setCi(request.getParameter("xci"));

m.setNombre(request.getParameter("xnombre"));

m.setAp(request.getParameter("xap"));

m.setAm(request.getParameter("xam"));

m.setEstado(Integer.*parseInt*(request.getParameter("xestado")));

this.estudianteManager.saveEstudiante(m);

} **catch**(Exception s){System.*out*.println("MENSAJE ERROR AL ADICIONAR addmat2 "+s.getLocalizedMessage());

p.put("men","NO NO SE ADICIONO SATISFACTORIAMENTE");

p.put("url", "reg_estudiantes.html");

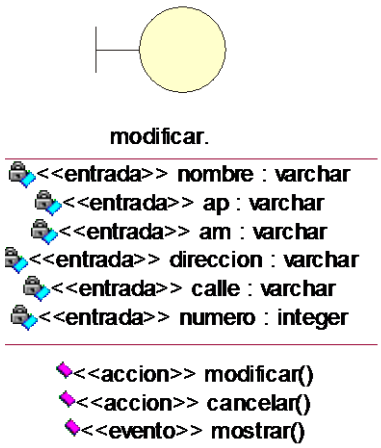
return new ModelAndView("mensaje",p); }

p.put("men","SE ADICIONO SATISFACTORIAMENTE");

p.put("url", "reg_estudiantes.html");

return new ModelAndView("mensaje",p); }

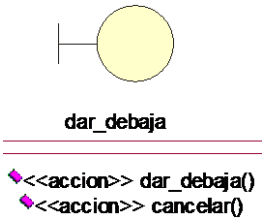
II.1.13.2.4 MODIFICAR



```
public ModelAndView modestudiante(HttpServletRequest
request,HttpServletResponse response) throws Exception,ServletException {
Map p = new HashMap();
                Estudiante m =
this.estudianteManager.getEstudiante(Integer.parseInt(request.getParameter("xrude")
));
                p.put("mat",m);
                 return new ModelAndView("estudiantes/modestudiante",p);
   }
//Ahora si Modifica
        public ModelAndView modestudiante2(HttpServletRequest
request,HttpServletResponse response) throws Exception,ServletException {
Map p = new HashMap();
                Estudiante m =
this.estudianteManager.getEstudiante(Integer.parseInt(request.getParameter("xrude")
));
                try {
                m.setCi(request.getParameter("xci"));
                m.setNombre(request.getParameter("xnombre"));
                m.setAm(request.getParameter("xam"));
                m.setAp(request.getParameter("xap"));
        //      m.setEstado(Integer.parseInt(request.getParameter("xestado")));
                this.estudianteManager.updateEstudiante(m);
                }   catch(Exception s){
System.out.println("MENSAJE MODMATERIAS  "+s.getLocalizedMessage());
```

```
                    p.put("men","NO NO SE MODIFICO
SATISFACTORIAMENTE");
                    p.put("url","reg_estudiantes.html");
                    return new ModelAndView("mensaje",p);
                }
                p.put("men","SE MODIFICO SATISFACTORIAMENTE");
                p.put("url","reg_estudiantes.html");
                return new ModelAndView("mensaje",p);
    }
```

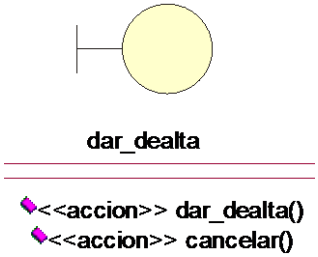
II.1.13.2.5 DAR DE BAJA




```
public ModelAndView estestudiante(HttpServletRequest
request,HttpServletResponse response) throws Exception,ServletException {
                Map p = new HashMap();
                //System.out.println("nooooooo
"+Integer.parseInt(request.getParameter("xrude")));          Estudiante m =
this.estudianteManager.getEstudiante(Integer.parseInt(request.getParameter("xrude")
));
                int xestadox=Integer.parseInt(request.getParameter("xestado"));
                try {
                    if(String.valueOf(xestadox).equals("1")){
xestadox=0;
```

```
                }else{
                        xestadox=1;
                }
            m.setEstado((xestadox));
            this.estudianteManager.updateEstudiante(m);
            }   catch(Exception s)
            {System.out.println("MENSAJE MODMATERIAS
"+s.getLocalizedMessage());
p.put("men","NO NO SE CAMBIO DE ESTADO SATISFACTORIAMENTE");
            p.put("url","reg_estudiantes.html");
            return new ModelAndView("mensaje",p);
            }
            p.put("men","SE CAMBIO DE ESTADO
SATISFACTORIAMENTE");
            p.put("url","reg_estudiantes.html");
            return new ModelAndView("mensaje",p);
   }
```

II.1.13.2.6 DAR DE ALTA




```
public ModelAndView dar_dealta(HttpServletRequest
```

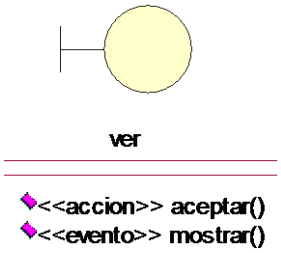
```
request,HttpServletResponse response) throws Exception,ServletException {
                Map p = new HashMap();
                //System.out.println("nooooooo
"+Integer.parseInt(request.getParameter("xrude")));
                Estudiante m =
this.estudianteManager.getEstudiante(Integer.parseInt(request.getParameter("xrude")
));

int xestadox=Integer.parseInt(request.getParameter("xestado"));
try {
                        if(String.valueOf(xestadox).equals("1")){
                                xestadox=0;
                        }else{
                                xestadox=1;
                        }
                m.setEstado((xestadox));
                this.estudianteManager.updateEstudiante(m);
}   catch(Exception s)
                {System.out.println("MENSAJE MODMATERIAS
"+s.getLocalizedMessage());
                p.put("men","NO NO SE CAMBIO DE ESTADO
SATISFACTORIAMENTE");
                p.put("url","reg_estudiantes.html");
                return new ModelAndView("mensaje",p);           }
```

p.put("men","SE CAMBIO DE ESTADO SATISFACTORIAMENTE");

p.put("url","reg_estudiantes.html");

return new ModelAndView("mensaje",p); }

II.1.13.2.7 VER



public ModelAndView kardex(HttpServletRequest request,HttpServletResponse response) **throws** Exception,ServletException {

HttpSession session=request.getSession(**true**);

//String usuario=(String) session.getAttribute("usuario");

Map p = **new** HashMap();

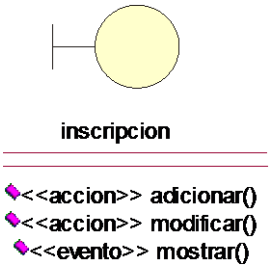
Administrativo m = **this**.administrativoManager.getAdministrativo(request.getParameter("xci"));

p.put("mat",m);

return new ModelAndView("administrador/kardex",p);

}

II.1.13.2.8 INSCRIPCION



Controlador

```
public class inscripcion extends MultiActionController implements
InitializingBean{

        private InscripcionManager inscripcionManager;

        private ParaleloManager paraleloManager;

        private CursoManager cursoManager;

        private EstudianteManager estudianteManager;

        public InscripcionManager getInscripcionManager() {

                return inscripcionManager;

        }

        public void setInscripcionManager(InscripcionManager inscripcionManager)
{
this.inscripcionManager = inscripcionManager;      }

        public CursoManager getCursoManager() {

                return cursoManager;          }

        public void setCursoManager(CursoManager cursoManager) {

                this.cursoManager = cursoManager; }

        public ParaleloManager getParaleloManager() {

                return paraleloManager;       }
```

```
    public void setParaleloManager(ParaleloManager paraleloManager) {
        this.paraleloManager = paraleloManager;    }
    public void afterPropertiesSet() throws Exception {
        }
    public EstudianteManager getEstudianteManager() {
        return estudianteManager;    }
    public void setEstudianteManager(EstudianteManager estudianteManager) {
        this.estudianteManager = estudianteManager;
    }
    /**
     * @return the inscripcionManager
     */
    public inscripcion() { }
```

```
public ModelAndView inscripcion(HttpServletRequest request,HttpServletResponse response) throws Exception,ServletException {
        return new ModelAndView("administrador/inscripcion");    }
```

InscripcionManager

```
import model.domain.Inscripcion;
import org.hibernate.SessionFactory;
import org.hibernate.event.SaveOrUpdateEvent;
public interface InscripcionManager {
    public SessionFactory getSessionFactory();
```

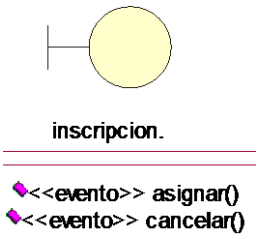
```
        public void setSessionFactory(SessionFactory sessionFactory);
        public void saveInscripcion(Inscripcion ins);
                /*
        * public Usuarios  getUsuarios (String xci);
 public void updateUsuarios (Usuarios  m);*/
        public void deleteInscripcion (int rude);
}
```

****InscripcionManagerImpl**

```
import model.domain.Inscripcion;
import org.hibernate.Session;
import org.hibernate.SessionFactory;

public class InscripcionManagerImpl implements InscripcionManager {
SessionFactory sessionFactory;
      public SessionFactory getSessionFactory()    {
      return sessionFactory;
   }    public void setSessionFactory(SessionFactory sessionFactory)
   {
      this.sessionFactory = sessionFactory;
   }
     public void saveInscripcion(Inscripcion ins){
         Session session = this.sessionFactory.getCurrentSession();
                 session.save(ins);
```

```
        }
    public void deleteInscripcion(int rude){
         int vamos=rude+0;
          System.out.println("habwer carfajop "+rude +"   "+vamos);
        Session session = this.sessionFactory.getCurrentSession();
        session.getTransaction().begin();
        session.delete(session.load(Inscripcion.class, new Integer(vamos)));
        session.getTransaction().commit();
          }
  }
```

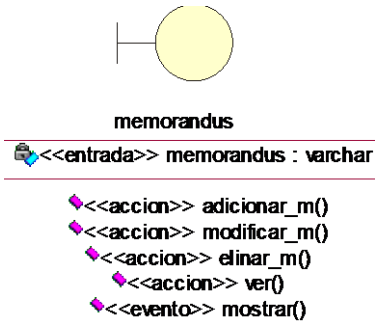
II.1.13.2.9 VER_INSCRIPCION




```
public ModelAndView addinscripcion(HttpServletRequest
request,HttpServletResponse response) throws Exception,ServletException {
                Map p = new HashMap();
                ArrayList milista=new ArrayList();
                try {
                        long aleatorio=(long)(Math.random()*10000);
                        String s=request.getParameter("rude");
                   Inscripcion ins=new Inscripcion();
                   ins.setCod_ins(aleatorio);
```

```
s.setCod_par(Integer.parseInt(request.getParameter("cod_par2")));
                    ins.setRude(Integer.parseInt(request.getParameter("rude")));
                    this.inscripcionManager.saveInscripcion(ins);
                } catch ( Exception e) {
                        System.out.println("che "+e.getLocalizedMessage());
                        p.put("men","estudiante no inscrito");
                        p.put("url",
"reg_estudiantes.html?cod_par='"+(request.getParameter("cod_par2")+"'"));
                    return new ModelAndView("mensaje",p);                    }
String cod_par2=request.getParameter("cod_par2");
                p.put("men","estudiante inscrito exitosamente");
                p.put("url", "reg_estudiantes.html?cod_par="+cod_par2);
            return new ModelAndView("mensaje",p);    }
```

II.1.13.2.10 MEMORANDUS



Controlador

```
package controlador1;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
```

```
import java.util.Map;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import model.domain.Administrativo;
import model.domain.Estudiante;
import model.domain.Materia;
import model.domain.Memorandus;
import model.manager.EstudianteManager;
import model.manager.MateriaManager;
import model.manager.MemorandusManager;
import org.springframework.beans.factory.InitializingBean;
import org.springframework.web.servlet.ModelAndView;
import org.springframework.web.servlet.mvc.multiaction.MultiActionController;
public class memorandus extends MultiActionController implements
InitializingBean{
	private MemorandusManager memorandusManager;
	public MemorandusManager getMemorandusManager() {
		return memorandusManager;
	}
	public void setMemorandusManager(MemorandusManager
memorandusManager) {
		this.memorandusManager = memorandusManager;
	}
```

```
	public void afterPropertiesSet() throws Exception {
		// TODO Auto-generated method stub
	}
	public memorandus() {
		// TODO Auto-generated constructor stub
	}
	public ModelAndView memorandus(HttpServletRequest
request,HttpServletResponse response) throws Exception,ServletException {
		Map p = new HashMap();
		ArrayList milistax = new ArrayList();
		Map mapx = new HashMap();
		//String estado="1";
		int rango=1;
		System.out.println("este es el dato"+
request.getParameter("xdatofiltro"));
			if(request.getParameter("xinter")==null){
				rango=1;
			}else{
				rango=Integer.parseInt(request.getParameter("xinter"));
			}

		//try {
			List m = this.memorandusManager.getMemorandus();
		//}  catch(Exception s){System.out.println("MENSAJE error aqui ");}
```

```
                p.put("datos",m);
            return new ModelAndView("memorandus/memorandus",p);
    }
```

****MemorandusManager**

```
package model.manager;
import model.domain.Administrativo;
import model.domain.Estudiante;
import model.domain.Materia;
import model.domain.Memorandus;
import model.domain.Usuarios;
import org.hibernate.SessionFactory;
import java.util.*;
public interface MemorandusManager {
        public abstract SessionFactory getSessionFactory();
        public abstract void setSessionFactory(SessionFactory sessionFactory);
        public abstract List getMemorandus();
        public void saveMemorandus (Memorandus  m);
        public void updateMemorandus(Memorandus m);
        public abstract Memorandus getMemorandus(long xcod_memo);
       }
```

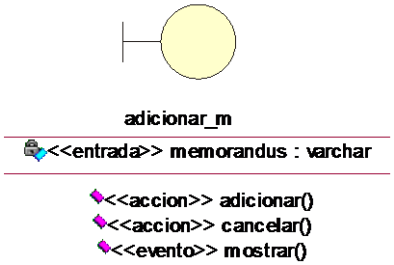
****MemorandusManagerImpl implements**

```
package model.manager;
import java.util.List;
```

```
import model.domain.Estudiante;
import model.domain.Materia;
import model.domain.Memorandus;
import model.manager.MemorandusManager;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
public class MemorandusManagerImpl implements MemorandusManager {
SessionFactory sessionFactory;
        public SessionFactory getSessionFactory()    {
        return sessionFactory;
    }
    public void setSessionFactory(SessionFactory sessionFactory)
    {
        this.sessionFactory = sessionFactory;
    }
    public List getMemorandus() {
                    Session session = this.sessionFactory.getCurrentSession();
                    return session.createQuery("from Memorandus").list();
            }
    public void saveMemorandus(Memorandus m){
            Session session = this.sessionFactory.getCurrentSession();
        session.merge(m);
    }
```

```
    public void updateMemorandus(Memorandus m){
        Session session = this.sessionFactory.getCurrentSession();
        session.update(m);
    }
    public Memorandus getMemorandus (long xcod_memo){
        Session session = this.sessionFactory.getCurrentSession();
        return (Memorandus) session.get(Memorandus.class, new
Long(xcod_memo));
    }
  }
```

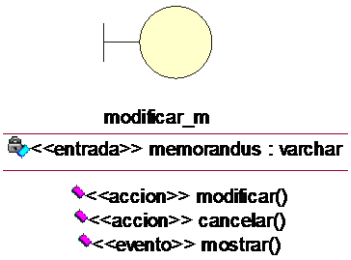
II.1.13.2.11 ADICIONAR_M




```
        public ModelAndView add_memorandus(HttpServletRequest
request,HttpServletResponse response) throws Exception,ServletException {

            Map p = new HashMap();
        return new ModelAndView("memorandus/add_memorandus",p);
    }
Public ModelAndView add_memorandus2(HttpServletRequest
request,HttpServletResponse response) throws Exception,ServletException {
```

```
Map p = new HashMap();
try {
    Memorandus m = new Memorandus();
    m.setNombre(request.getParameter("xnombre"));
    this.memorandusManager.saveMemorandus(m);
    } catch(Exception s){System.out.println("MENSAJE ERROR
AL ADICIONAR add "+s.getLocalizedMessage());
        p.put("men","NO NO SE ADICIONO
SATISFACTORIAMENTE");
        p.put("url","memorandus.html");
        return new ModelAndView("mensaje",p);
    }
    p.put("men","SE ADICIONO SATISFACTORIAMENTE");
    p.put("url","memorandus.html");
    return new ModelAndView("mensaje",p);
  }
```

II.1.13.2.12 MODIFICAR_M

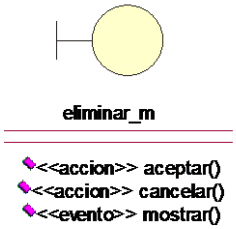



```
public ModelAndView modificar_m(HttpServletRequest
request,HttpServletResponse response) throws Exception,ServletException {
```

```
        Map p = new HashMap();


        Memorandus m =
this.memorandusManager.getMemorandus(Long.parseLong(request.getParameter("x
cod_memo")));
        p.put("mat",m);
    return new ModelAndView("memorandus/modificar_m",p);
}

//Ahora si Modifica
public ModelAndView modificar2_m(HttpServletRequest
request,HttpServletResponse response) throws Exception,ServletException {


        Map p = new HashMap();
        Memorandus m =
this.memorandusManager.getMemorandus(Long.parseLong(request.getParameter("x
cod_memo")));
        try {


        m.setNombre(request.getParameter("xnombre"));
        this.memorandusManager.updateMemorandus(m);
        }   catch(Exception s){
                System.out.println("MENSAJE MODMATERIAS
"+s.getLocalizedMessage());
                p.put("men","NO NO SE MODIFICO SATISFACTORIAMENTE");
```

```
            p.put("url","memorandus.html");
            return new ModelAndView("mensaje",p);
        }
        p.put("men","SE MODIFICO SATISFACTORIAMENTE");
        p.put("url","memorandus.html");
        return new ModelAndView("mensaje",p);
}
```

II.1.13.2.13 ELIMINAR_M




```
public ModelAndView del_materias(HttpServletRequest
request,HttpServletResponse response) throws Exception,ServletException {
            Map p = new HashMap();
            try{
                List datosx =
this.materiaManager.getValidaUsuario2(Long.parseLong(request.getParameter("xco
d_mat")));
                int tam=datosx.size();
                    try{
System.out.println("borro  "+datosx.iterator());
        this.materiaManager.deleteMaterias(Long.parseLong(request.getParameter("x
co_mat")));
```

}**catch**(Exception s){

System.*out*.println("eliminadooo "+s.getLocalizedMessage());

}

}**catch**(Exception e){

System.*out*.println("eeeeeeeerrrrrrrrrrooooooooooorrrrrrr "+e.getLocalizedMessage());

p.put("men","NO NO NO SE ELIMINO SATISFACTORIAMENTE");

p.put("url","reg_materias.html");

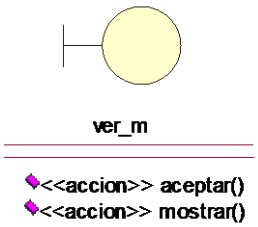
return new ModelAndView("mensaje",p); }

p.put("men","SE ELIMINO SATISFACTORIAMENTE");

p.put("url","reg_materias.html");

return new ModelAndView("mensaje",p); }

II.1.13.2.14 VER_M



public ModelAndView ver_m(HttpServletRequest request,HttpServletResponse response) **throws** Exception,ServletException {

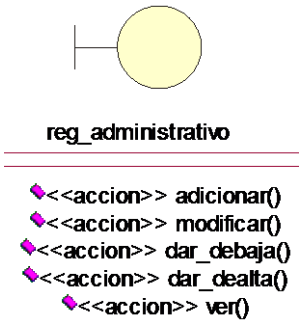
HttpSession session=request.getSession(**true**);

//String usuario=(String) session.getAttribute("usuario");

Map p = **new** HashMap();

```
                Administrativo m =
this.administrativoManager.getAdministrativo(request.getParameter("xci"));
                p.put("mat",m);
                 return new ModelAndView("administrador/kardex",p);
    }
```

II.1.13.2.15 REG_ADMINISTRATIVO



****Controlador**

```
package controlador1;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.HashMap;
import java.util.Iterator;
import java.util.List;
import java.util.Map;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
```

```
import javax.servlet.http.HttpSession;
import model.domain.Administrativo;
import model.domain.Estudiante;
import model.domain.Usuarios;
import model.manager.AdministrativoManager;
import org.apache.velocity.Template;
import org.hibernate.type.TextType;
import org.springframework.beans.factory.InitializingBean;
import org.springframework.web.servlet.ModelAndView;
import org.springframework.web.servlet.mvc.multiaction.MultiActionController;
public class reg_administrativo extends MultiActionController implements
InitializingBean{
	private AdministrativoManager administrativoManager;//
	public AdministrativoManager getAdministrativoManager() {
		return administrativoManager;
	}
	public void setAdministrativoManager(AdministrativoManager
administrativoManager) {
		this.administrativoManager = administrativoManager;
	}
	public void afterPropertiesSet() throws Exception {
		// TODO Auto-generated method stub
	}
```

```
	public reg_administrativo() {
		// TODO Auto-generated constructor stub
	}
	public ModelAndView registro(HttpServletRequest
request,HttpServletResponse response) throws Exception,ServletException {
		Map p = new HashMap();
		ArrayList milista=new ArrayList();
		ArrayList milistax = new ArrayList();
		Map mapx = new HashMap();
		String datofiltro="";
		int rango=1;
		System.out.println("este el de dato de
administrador"+request.getParameter("xdatofiltro"));
			if(request.getParameter("xdatofiltro")==null){
				datofiltro="";
			}else{
				datofiltro=request.getParameter("xdatofiltro");
			}
			if(request.getParameter("xinter")==null){
				rango=1;
			}else{
				rango=Integer.parseInt(request.getParameter("xinter"));
			}
```

```
try {
    //generador de el paginador
    List Total =
this.administrativoManager.getListaUsuariosFiltroTotal(datofiltro);
    double num=Total.size();
    double xx=Math.ceil(num / 5);
    if(rango>1){
        mapx = new HashMap();
        mapx.put("rango","<a id='x'
onMouseOver='Over(this.id)' onMouseOut='Out(this.id)'
href='registro.html?xdatofiltro="+datofiltro+"&xinter="+(rango-
1)+"'>"+"<<<"+"</a>");
        milistax.add(mapx);
    }
    for(int i=1;i<=xx;i++){
        mapx = new HashMap();
        if(i==rango){
            mapx.put("rango",""+i+"");
        }else{
            mapx.put("rango","<a id="'+i+'"
onMouseOver='Over(this.id)' onMouseOut='Out(this.id)'
href='registro.html?xdatofiltro="+datofiltro+"&xinter="+i+"'>"+i+"</a>");
        }
        milistax.add(mapx);
    }
```

```
                    if(rango<xx){
                        mapx = new HashMap();
                        mapx.put("rango","<a id='y'
onMouseOver='Over(this.id)' onMouseOut='Out(this.id)'
href='registro.html?xdatofiltro="+datofiltro+"&xinter="+(rango+1)+"">"+">>>"+"</
a>");
                        milistax.add(mapx);
                    }
                    //lista de los datos en 5 filas
                    List mat =
this.administrativoManager.getListaUsuariosFiltro(datofiltro,rango);
                    p.put("datos", mat);
                    //envio los intervalos
                    p.put("intervalo",milistax);
                } catch(Exception s){System.out.println("MENSAJE error  ");}

                p.put("datofiltradox",datofiltro);

                return new ModelAndView("administrador/registro",p);
    }
```

****AdministrativoManager**

```
package model.manager;
import java.util.List;
import model.domain.Administrativo;
```

```
import model.domain.Usuarios;
import org.hibernate.SessionFactory;
public interface AdministrativoManager {
        public abstract SessionFactory getSessionFactory();
        public abstract void setSessionFactory(SessionFactory sessionFactory);
        public abstract List getTipoadm(String elci);
        public void saveAdministrativo (Administrativo  m);
        public void updateAdministrativo(Administrativo m);
        public void deleteAdministrativo(String xci);
        public abstract List getListaUsuariosFiltro(String datofiltro,int i);
        public abstract List getListaUsuariosFiltroTotal(String datofiltro);
        public abstract Administrativo getAdministrativo(String xci);
        public abstract List getListaAdministrativo ();
        public abstract List getListaAdministrativo2 (String letra1,String letra2);
        public abstract List getListaAdministrativo3 (Integer xy);
        public abstract List getListaAdministrativo4 (String letra1,String
letra2,Integer xy);
        }
```

AdministrativoManagerImpl

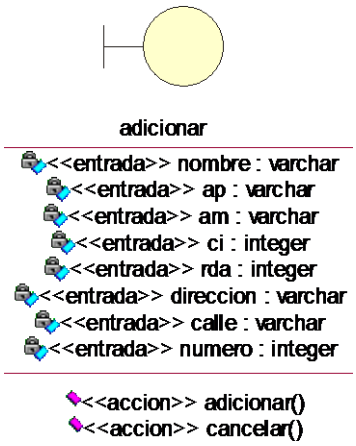
```
package model.manager;
import java.util.List;
import model.domain.Administrativo;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
```

```
public class AdministrativoManagerImpl implements AdministrativoManager{
SessionFactory sessionFactory;
     public SessionFactory getSessionFactory()    {
        return sessionFactory;
    }
    public void setSessionFactory(SessionFactory sessionFactory)
    {
        this.sessionFactory = sessionFactory;
    }
    public Administrativo getAdministrativo (String ci){
            session session = this.sessionFactory.getCurrentSession();
            return (Administrativo) session.get(Administrativo.class, new String(ci));
    }
    public void saveAdministrativo(Administrativo m){
            Session session = this.sessionFactory.getCurrentSession();
             session.merge(m);
    }
    public void updateAdministrativo(Administrativo m){
            Session session = this.sessionFactory.getCurrentSession();
            session.update(m);
    }
    public void deleteAdministrativo(String xci){
            Session session = this.sessionFactory.getCurrentSession();
```

```
        session.getTransaction().begin();
        session.delete(session.load(Administrativo.class, new String(xci)));
        session.getTransaction().commit();
    }
    public List getListaUsuariosFiltroTotal(String datofiltro) {
                Session session = this.sessionFactory.getCurrentSession();
                return session.createQuery("from Administrativo where upper(ap||'
'||am||' '||nombre) like upper('"+datofiltro+"%') order by ap asc").list();
        }
    public List getListaUsuariosFiltro(String datofiltro,int i) {
                Session session = this.sessionFactory.getCurrentSession();
                return session.createQuery("from Administrativo where upper(ap||'
'||am||' '||nombre) like upper('"+datofiltro+"%') order by ap asc").setFirstResult((i-
1)*5).setMaxResults(5).list();
        }
    public List getTipoadm(String elci) {
                Session session = this.sessionFactory.getCurrentSession();
                return session.createSQLQuery("Select tipo from administrativo where
(ci='"+elci+"')").list();
        }
 public List getListaAdministrativo() {
        String sql="from Administrativo where tipo=2 and estado=1";
                Session session = this.sessionFactory.getCurrentSession();
                return session.createQuery(sql).list();
```

```
        }
    public List getListaAdministrativo2 (String letra1,String letra2) {
        //especificamos la consulta
        String xcon="FROM Administrativo  where (ap like '"+letra1+"%' or ap like
'"+letra2+"%')";
                Session session = this.sessionFactory.getCurrentSession();
                return session.createQuery(xcon).list();
        }
    public List getListaAdministrativo3(Integer xy) {
                Session session = this.sessionFactory.getCurrentSession();
                //return session.createQuery(" FROM Usuarios limit 5 offset '"+xy+"'
").list();
                return session.createQuery(" FROM
Administrativo").setFirstResult(xy).setMaxResults(5).list();
        }
      public List getListaAdministrativo4(String letra1,String letra2,Integer xy) {
        Session session = this.sessionFactory.getCurrentSession();
                //return session.createQuery(" FROM Usuarios  where (ap like
'"+xx+"%') limit 5 offset '"+xy+"' ").list();
                return session.createQuery("FROM Administrativo  where (ap like
'"+letra1+"%' or ap like '"+letra2+"%')").setFirstResult(xy).setMaxResults(5).list();
    }
}
```

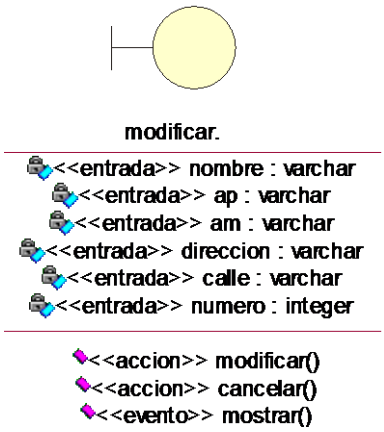
II.1.13.2.16 ADICIONAR




```
public ModelAndView adicionar_administrativo(HttpServletRequest request,HttpServletResponse response) throws Exception,ServletException {
        Map p = new HashMap();
    return new ModelAndView("administrador/adicionar_administrativo",p);
  }

    public ModelAndView adicionar_administrativo2(HttpServletRequest request,HttpServletResponse response) throws Exception,ServletException {
        Map p = new HashMap();
        try {
            Date d;
            SimpleDateFormat fechay = new SimpleDateFormat("dd-MM-yyyy");
            d = fechay.parse(request.getParameter("xf_nac"));
            Administrativo m=new Administrativo();
        m.setRda(Integer.parseInt(request.getParameter("xrda")));
```

```
                m.setCi(request.getParameter("xci"));
                m.setNombre(request.getParameter("xnombre"));
                m.setAp(request.getParameter("xap"));
                m.setAm(request.getParameter("xam"));
                m.setF_nac(d);
                m.setSexo(request.getParameter("xsexo"));
                m.setEstado_civil(request.getParameter("xestado_civil"));
            m.setEstado(Integer.parseInt(request.getParameter("xestado")));
                m.setTipo(Integer.parseInt(request.getParameter("xtipo")));
                this.administrativoManager.saveAdministrativo(m);
                } catch (Exception e) {
                p.put("men","No se Adicionado exitosamente");
            System.out.println("segundooooooooo"+e.getLocalizedMessage());
                p.put("url", "registro.html");
                return new ModelAndView("mensaje",p);
            }
            p.put("men","Adicionado exitosamente ");
            p.put("url", "registro.html");
            return new ModelAndView("mensaje",p);
    }
```

II.1.13.2.17 MODIFICAR

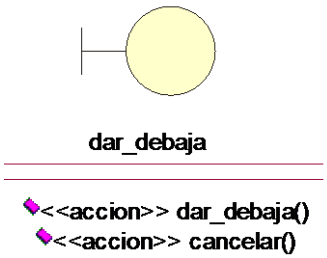



```
public ModelAndView modificar_administrativo(HttpServletRequest request,HttpServletResponse response) throws Exception,ServletException {
        HttpSession session=request.getSession(true);
        String usuario=(String) session.getAttribute("usuario");
        Map p = new HashMap();
        /*if(usuario==null){
            p.put("men","ACCESO DENEGADO");
          return new ModelAndView("mensaje",p);
        }*/
        try {
Administrativo m = this.administrativoManager.getAdministrativo(request.getParameter("xci"));
            System.out.println("tamanño adm "+m);
            p.put("mat",m);
        } catch (Exception e) {
            System.out.println(" [[[[[[ "+e.getLocalizedMessage());
```

```
                }


        return new ModelAndView("administrador/mod_administrativo",p);
    }
        //Ahora si Modifica
public ModelAndView modificar_administrador2(HttpServletRequest request,HttpServletResponse response) throws Exception,ServletException {
                Map p = new HashMap();
                Administrativo m = this.administrativoManager.getAdministrativo(request.getParameter("xci"));
                try {
                        m.setNombre(request.getParameter("xnombre"));
                        m.setRda(Integer.parseInt(request.getParameter("xrda")));
                        m.setAp(request.getParameter("xap"));
                        m.setAm(request.getParameter("xam"));
                        //Se modifica en la Base de datos
                        this.administrativoManager.updateAdministrativo(m);
                }   catch(Exception s){
                        System.out.println("MENSAJE MODMATERIAS "+s.getLocalizedMessage());
                        p.put("men"," NO NO SE MODIFICO SATISFACTORIAMENTE");
                        p.put("url","registro.html");
                        return new ModelAndView("mensaje",p);
```

```
            }
            p.put("men","SE MODIFICO SATISFACTORIAMENTE");
            p.put("url","registro.html");
            return new ModelAndView("mensaje",p);
    }
```

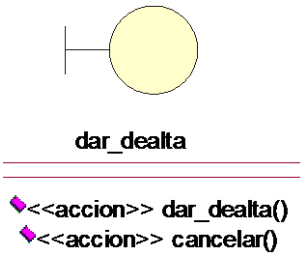
II.1.13.2.18 DAR DE BAJA




```
public ModelAndView dar_debaja(HttpServletRequest request,HttpServletResponse response) throws Exception,ServletException {
            Map p = new HashMap();
            //System.out.println("nooooooo "+Integer.parseInt(request.getParameter("xrude")));
            Administrativo m = this.administrativoManager.getAdministrativo(request.getParameter("xci"));
            int xestadox=Integer.parseInt(request.getParameter("xestado"));
            try {
                if(String.valueOf(xestadox).equals("1")){
                    xestadox=0;
                }else{
                    xestadox=1;
                }
```

```
                m.setEstado((xestadox));
                this.administrativoManager.updateAdministrativo(m);
                }   catch(Exception s)
                {System.out.println("MENSAJE MODMATERIAS
"+s.getLocalizedMessage());
                p.put("men","NO NO SE CAMBIO DE ESTADO
SATISFACTORIAMENTE");
                p.put("url","reg_administrativo.html");
                return new ModelAndView("mensaje",p);
                }
                p.put("men","SE CAMBIO DE ESTADO
SATISFACTORIAMENTE");
                p.put("url","reg_administrativo.html");
                return new ModelAndView("mensaje",p);
    }
```

II.1.13.2.19 DAR DE ALTA

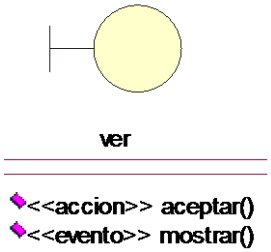



```
public ModelAndView dar_debaja(HttpServletRequest request,HttpServletResponse
response) throws Exception,ServletException {
                Map p = new HashMap();
```

```
		//System.out.println("nooooooo "+Integer.parseInt(request.getParameter("xrude")));
		Administrativo m = this.administrativoManager.getAdministrativo(request.getParameter("xci"));
		int xestadox=Integer.parseInt(request.getParameter("xestado"));
		try {
			if(String.valueOf(xestadox).equals("1")){
				xestadox=0;
			}else{
				xestadox=1;
			}
		m.setEstado((xestadox));
		this.administrativoManager.updateAdministrativo(m);
		}  catch(Exception s)
		{System.out.println("MENSAJE MODMATERIAS "+s.getLocalizedMessage());
		p.put("men","NO NO SE CAMBIO DE ESTADO SATISFACTORIAMENTE");
		p.put("url","reg_administrativo.html");
		return new ModelAndView("mensaje",p);
		}
		p.put("men","SE CAMBIO DE ESTADO SATISFACTORIAMENTE");
		p.put("url","reg_administrativo.html");
```

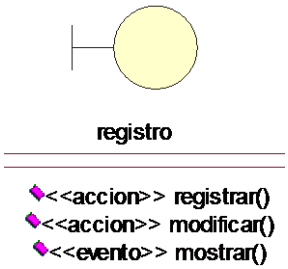
```
            return new ModelAndView("mensaje",p);
    }
```

II.1.13.2.20 VER




```
public ModelAndView kardex(HttpServletRequest request,HttpServletResponse
response) throws Exception,ServletException {
            HttpSession session=request.getSession(true);
            //String usuario=(String) session.getAttribute("usuario");
            Map p = new HashMap();
            Administrativo m =
this.administrativoManager.getAdministrativo(request.getParameter("xci"));
            p.put("mat",m);
 return new ModelAndView("administrador/kardex",p);
    }
```

II.1.13.2.21 REGISTRO



****Controlador**

```
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
import java.util.List;
import java.util.Map;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import model.manager.*;
import model.domain.*;
import org.springframework.beans.factory.InitializingBean;
import org.springframework.web.servlet.ModelAndView;
import org.springframework.web.servlet.mvc.multiaction.MultiActionController;
public class inscripcion extends MultiActionController implements InitializingBean{
private InscripcionManager inscripcionManager;
private ParaleloManager paraleloManager;
private CursoManager cursoManager;
private EstudianteManager estudianteManager;
public InscripcionManager getInscripcionManager() {
return inscripcionManager;}
public void setInscripcionManager(InscripcionManager inscripcionManager) {
this.inscripcionManager = inscripcionManager;
}
```

```
public CursoManager getCursoManager() {
return cursoManager;
}
public void setCursoManager(CursoManager cursoManager) {
this.cursoManager = cursoManager;
}
public ParaleloManager getParaleloManager() {
return paraleloManager;
}
public void setParaleloManager(ParaleloManager paraleloManager) {
this.paraleloManager = paraleloManager;
}
public void afterPropertiesSet() throws Exception {}
public EstudianteManager getEstudianteManager() {
return estudianteManager;
}
public void setEstudianteManager(EstudianteManager estudianteManager) {
this.estudianteManager = estudianteManager;}
/**
* @return the inscripcionManager
*/
public inscripcion() {}
public ModelAndView inscripcion(HttpServletRequest request,HttpServletResponse
response) throws Exception,ServletException {
```

```
return new ModelAndView("administrador/inscripcion");
}
```

****RegistroManager**

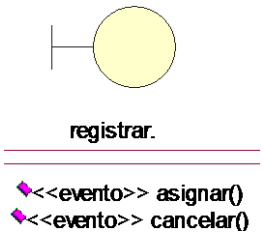
```
import model.domain.Inscripcion;
import org.hibernate.SessionFactory;
import org.hibernate.event.SaveOrUpdateEvent;
public interface RegistroManager {
        public SessionFactory getSessionFactory();
        public void setSessionFactory(SessionFactory sessionFactory);
        public void saveInscripcion(Inscripcion ins);          /*
public Usuarios  getUsuarios (String xci);
        public void updateUsuarios (Usuarios  m);*/
        public void deleteInscripcion (int rude);}
```

****RegistroManagerImpl**

```
import model.domain.Inscripcion;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
public class RegistroManagerImpl implements RegistroManager {
SessionFactory sessionFactory;
 public SessionFactory getSessionFactory()    {
       return sessionFactory;
   }
```

```
  public void setSessionFactory(SessionFactory sessionFactory)
   {
      this.sessionFactory = sessionFactory;    }
  public void saveInscripcion(Inscripcion ins){
        Session session = this.sessionFactory.getCurrentSession();
              session.save(ins);                    }
 public void deleteInscripcion(int rude){
          int vamos=rude+0;
System.out.println("habwer carfajop "+rude +"   "+vamos);
        Session session = this.sessionFactory.getCurrentSession();
        session.getTransaction().begin();
        session.delete(session.load(Inscripcion.class, new Integer(vamos)));
        session.getTransaction().commit();   }
```

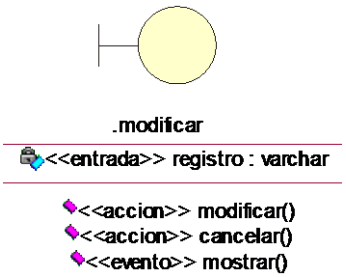
II.1.13.2.22 REGISTRAR




```
public ModelAndView addinscripcion(HttpServletRequest
request,HttpServletResponse response) throws Exception,ServletException {
              Map p = new HashMap();
              ArrayList milista=new ArrayList();
              try {
```

```
                    long aleatorio=(long)(Math.random()*10000);
                    String s=request.getParameter("rude");
                Inscripcion ins=new Inscripcion();
                ins.setCod_ins(aleatorio);
ins.setCod_par(Integer.parseInt(request.getParameter("cod_par2")));
                ins.setRude(Integer.parseInt(request.getParameter("rude")));
            this.inscripcionManager.saveInscripcion(ins);
} catch ( Exception e) {
                    System.out.println("che "+e.getLocalizedMessage());
                    p.put("men","estudiante no inscrito");
                    p.put("url",
"reg_estudiantes.html?cod_par="'+(request.getParameter("cod_par2")+""'));
                    return new ModelAndView("mensaje",p);                    }
                String cod_par2=request.getParameter("cod_par2");
                p.put("men","estudiante inscrito exitosamente");
                p.put("url", "reg_estudiantes.html?cod_par="+cod_par2);
            return new ModelAndView("mensaje",p);    }
```

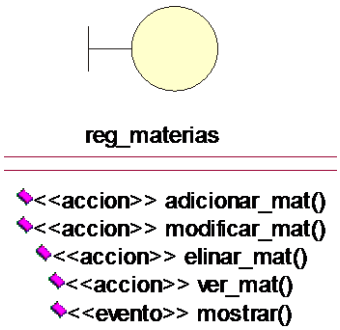
II.1.13.2.23 MODIFICAR




```
public ModelAndView modif_registro(HttpServletRequest request,
HttpServletResponse response)throws ServletException{
```

```
HashMap p=new HashMap();
try {
    int haber= Integer.parseInt(request.getParameter("rude"));
this.inscripcionManager.deleteInscripcion(haber);
}catch(Exception e){
System.out.println("exception del inscripcion
"+e.getLocalizedMessage());
p.put("men", "se elimino correctamente la inscripcion del estudiante");
p.put("url","reg_estudiantes.html?cod_par="+request.getParameter("cod_par"
));
return new ModelAndView("mensaje",p);
}
p.put("men", "se elimino correctamente la inscripcion");
p.put("url","reg_estudiantes.html?cod_par="+request.getParameter("cod_par"
));
return new ModelAndView("mensaje",p);  }
```

II.1.13.2.24 REG_ MATERIAS



****Controlador**

```
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import model.domain.Administrativo;
import model.domain.Estudiante;
import model.domain.Materia;
import model.manager.EstudianteManager;
import model.manager.MateriaManager;
import org.springframework.beans.factory.InitializingBean;
import org.springframework.web.servlet.ModelAndView;
import org.springframework.web.servlet.mvc.multiaction.MultiActionController;
public class reg_materias extends MultiActionController implements
InitializingBean{
	private MateriaManager materiaManager;
	public MateriaManager getMateriaManager() {
		return materiaManager;	}
	public void setMateriaManager(MateriaManager materiaManager) {
		this.materiaManager = materiaManager;
	}
	public void afterPropertiesSet() throws Exception {
```

```
		// TODO Auto-generated method stub 			}
public reg_materias() {
		// TODO Auto-generated constructor stub	}
		public ModelAndView reg_materias(HttpServletRequest
request,HttpServletResponse response) throws Exception,ServletException {
		Map p = new HashMap();
		ArrayList milistax = new ArrayList();
		Map mapx = new HashMap();
		String datofiltro="";
		String estado="1";
		int rango=1;
		String dato=request.getParameter("xdatofiltro");
		System.out.println("este es el dato"+
request.getParameter("xdatofiltro"));
			if(request.getParameter("xdatofiltro")==null){
				datofiltro="";
			}else{
				datofiltro=request.getParameter("xdatofiltro");
	}
			if(request.getParameter("xinter")==null){
				rango=1;
			}else{
				rango=Integer.parseInt(request.getParameter("xinter"));
			}
```

```
if(request.getParameter("xopcion")==null){
                    estado="1";
                }else{
                    estado=request.getParameter("xopcion");          }
        try {
                    List Total =
this.materiaManager.getListaCursosFiltroTotal(datofiltro,Integer.parseInt(estado));
                    double num=Total.size();
                    double xx=Math.ceil(num / 5);
                    if(rango>1){
                        mapx = new HashMap();
                        mapx.put("rango","<a id='x'
onMouseOver='Over(this.id)' onMouseOut='Out(this.id)'
href='reg_materias.html?xdatofiltro="+datofiltro+"&xinter="+(rango-
1)+"&xopcion="+estado+"'>"+"<<<"+"</a>");
                        milistax.add(mapx);
                    }
                    for(int i=1;i<=xx;i++){
                        mapx = new HashMap();
                        if(i==rango){
                            mapx.put("rango",""+i+"");
                        }else{
mapx.put("rango","<a id="'+i+'" onMouseOver='Over(this.id)'
onMouseOut='Out(this.id)'
```

```
href='reg_materias.html?xdatofiltro="+datofiltro+"&xinter="+i+"&xopcion="+estado+"'>"+i+"</a>");
                                        }
                                        milistax.add(mapx);
                                }
                                if(rango<xx){
                                        mapx = new HashMap();
                                        mapx.put("rango","<a id='y' onMouseOver='Over(this.id)' onMouseOut='Out(this.id)' href='reg_materias.html?xdatofiltro="+datofiltro+"&xinter="+(rango+1)+"&xopcion="+estado+"'>"+">>>"+"</a>");
                                        milistax.add(mapx);
                                }
                                List mat = this.materiaManager.getListaCursosFiltro(datofiltro,rango,Integer.parseInt(estado));
                                System.out.println("mierda   "+mat.size());
                                p.put("datos", mat);
                                p.put("intervalo",milistax);
                        }  catch(Exception s){System.out.println("MENSAJE error aqui ");}
                        if(estado.equals("1")){
                                p.put("xestadox"," ");
p.put("yestadoy","checked");
                        }else{
                                p.put("xestadox","checked");
```

```
                    p.put("yestadoy"," ");          }
        p.put("datofiltradox",datofiltro);
          return new ModelAndView("materias/reg_materias",p);    }
```

****MateriaManager**

```
import model.domain.Administrativo;
import model.domain.Estudiante;
import model.domain.Materia;
iport model.domain.Usuarios;
import org.hibernate.SessionFactory;
import java.util.*;
public interface MateriaManager {
        public abstract SessionFactory getSessionFactory();
        public abstract void setSessionFactory(SessionFactory sessionFactory);
        public abstract List getListaMateria(String nom_par,String nombre_cur);
        public abstract List getListaMateria2();
        public abstract List getListaCursosFiltroTotal(String datofiltro,int estado);
        public abstract List getListaCursosFiltro(String datofiltro,int i,int estado);
        public abstract List getValidaUsuario2(long xcod_mat);
        public void saveMaterias (Materia  m);
        public void updateMaterias(Materia m);
        public void deleteMaterias(long scod_mat);
        public abstract Materia getMateria(long xcod_mat);}
```

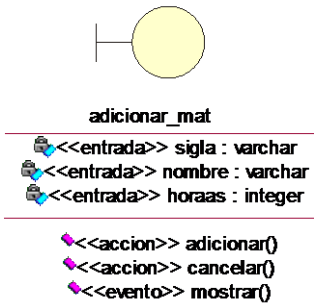
****MateriaManagerImpl**

```
import java.util.List;
import model.domain.Estudiante;
import model.domain.Materia;
import model.manager.MateriaManager;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
public class MateriaManagerImpl implements MateriaManager {
SessionFactory sessionFactory;
        public SessionFactory getSessionFactory()    {
        return sessionFactory;    }
    public void setSessionFactory(SessionFactory sessionFactory)
    {
        this.sessionFactory = sessionFactory;    }
        public List getListaMateria2() {
                    Session session = this.sessionFactory.getCurrentSession();
return session.createQuery("from Materia").list();
            }
    public List getListaMateria(String nom_par,String nombre_cur) {
                    Session session=null;
                    session= this.sessionFactory.getCurrentSession();
                    return session.createQuery("select
m.cod_mat,m.nombre_mat,h.cod_hor,h.cod_dia,h.ci,h.cod_m_p from Materia m,
```

```
Administrar_Horario h where (m.cod_mat IN((select cod_mat from Paralelo_Materia "+
                " where cod_m_p=h.cod_m_p and cod_par IN(select cod_par from Paralelo "+
                " where nom_par='"+nom_par+"' and cod_cur IN(select cod_cur from Curso where nombre_cur='"+nombre_cur+"')))) and "+
                " (h.cod_m_p IN(select cod_m_p from "+
                " Administrar_Horario where (cod_dia IN(select cod_dia from Dia ) and cod_hor IN(select cod_hor from Hora ))))) order by (h.cod_hor,h.cod_dia)").list();
    }
    public List getValidaUsuario2(long xcod_mat) {
        //especificamos la consulta
        String xsql="from Materia where cod_mat='"+xcod_mat+"'";
                Session session = this.sessionFactory.getCurrentSession();
                return session.createQuery(xsql).list();
        }
    public List getListaCursosFiltroTotal(String datofiltro,int estado) {
                Session session = this.sessionFactory.getCurrentSession();
                return session.createQuery("from Materia where (estado="+estado+") and upper(nombre_mat) like upper('"+datofiltro+"%') order by nombre_mat asc").list();
        }
    public List getListaCursosFiltro(String datofiltro,int i,int estado) {
                Session session = this.sessionFactory.getCurrentSession();
```

```
                return session.createQuery("from Materia where (estado="+estado+")
and upper(nombre_mat) like upper('"+datofiltro+"%') order by nombre_mat
asc").setFirstResult((i-1)*5).setMaxResults(5).list();
        }
   public Materia getMateria (long xcod_mat){
        Session session = this.sessionFactory.getCurrentSession();
        return (Materia) session.get(Materia.class, new Long(xcod_mat));
   }
      public void saveMaterias(Materia m){
        Session session = this.sessionFactory.getCurrentSession();
      session.merge(m);
   }
   public void updateMaterias(Materia m){
        Session session = this.sessionFactory.getCurrentSession();
        session.update(m);    }
   public void deleteMaterias(long xcod_mat){
        Session session = this.sessionFactory.getCurrentSession();
        session.getTransaction().begin();
        session.delete(session.load(Materia.class, new Long(xcod_mat)));
        session.getTransaction().commit();
   }}
```

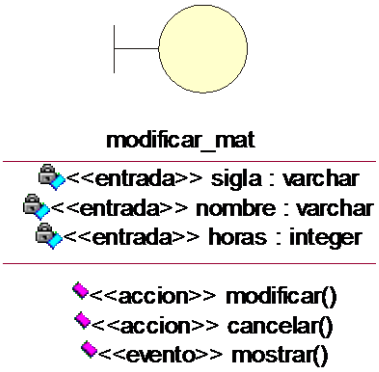
II.1.13.2.25 ADICIONAR_MAT




```
public ModelAndView add_materias(HttpServletRequest
request,HttpServletResponse response) throws Exception,ServletException {
                Map p = new HashMap();
        return new ModelAndView("materias/add_materias",p);    }
        public ModelAndView add_materias2(HttpServletRequest
request,HttpServletResponse response) throws Exception,ServletException {
                                Map p = new HashMap();
                try {                   Materia m = new Materia();
//m.setCod_mat(Long.parseLong(request.getParameter("xcod_mat")));
                        m.setSigla(request.getParameter("xsigla"));
m.setNombre_mat(request.getParameter("xnombre_mat"));
        m.setHoras_acad(Double.parseDouble(request.getParameter("xhoras_acad")))
;
        m.setEstado(Integer.parseInt(request.getParameter("xestado")));
                        this.materiaManager.saveMaterias(m);
                }   catch(Exception s){System.out.println("MENSAJE ERROR AL
ADICIONAR addmat2 "+s.getLocalizedMessage());
                                p.put("men","NO NO SE ADICIONO
SATISFACTORIAMENTE");
```

```
                    p.put("url","reg_materias.html");

                    return new ModelAndView("mensaje",p);          }
            p.put("men","SE ADICIONO SATISFACTORIAMENTE");

            p.put("url","reg_materias.html");

            return new ModelAndView("mensaje",p);    }
```

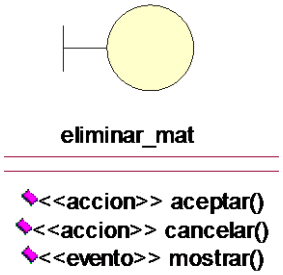
II.1.13.2.26 MODIFICAR_MAT




```
public ModelAndView mod_materias(HttpServletRequest
request,HttpServletResponse response) throws Exception,ServletException {

            Map p = new HashMap();

Materia m =
this.materiaManager.getMateria(Long.parseLong(request.getParameter("xcod_mat"))
);

            p.put("mat",m);

      return new ModelAndView("materias/mod_materias",p);    }

        //Ahora si Modifica

        public ModelAndView mod_materias2(HttpServletRequest
request,HttpServletResponse response) throws Exception,ServletException {

            Map p = new HashMap();
```

```
                Materia m =
this.materiaManager.getMateria(Long.parseLong(request.getParameter("xcod_mat"))
);
                try {
                m.setSigla(request.getParameter("xsigla"));
                m.setNombre_mat(request.getParameter("xnombre_mat"));
        m.setHoras_acad(Double.parseDouble(request.getParameter("xhoras_acad")))
;
                m.setEstado(Integer.parseInt(request.getParameter("xestado")));
                this.materiaManager.updateMaterias(m);
                }   catch(Exception s){
                        System.out.println("MENSAJE MODMATERIAS
"+s.getLocalizedMessage());
                        p.put("men","NO NO SE MODIFICO
SATISFACTORIAMENTE");
p.put("url","reg_materias.html");
                        return new ModelAndView("mensaje",p);            }
                p.put("men","SE MODIFICO SATISFACTORIAMENTE");
                p.put("url","reg_materias.html");
                return new ModelAndView("mensaje",p);    }
```

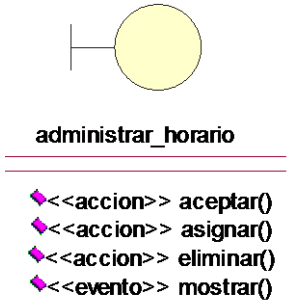
II.1.13.2.27 ELIMINAR_ MAT




```
public ModelAndView del_materias(HttpServletRequest
request,HttpServletResponse response) throws Exception,ServletException {
        Map p = new HashMap();
        try{
            List datosx =
this.materiaManager.getValidaUsuario2(Long.parseLong(request.getParameter("xco
d_mat")));
            int tam=datosx.size();
              try{
                  System.out.println("borro  "+datosx.iterator());
      this.materiaManager.deleteMaterias(Long.parseLong(request.getParameter("x
cod_mat"))); }catch(Exception s){
System.out.println("eliminadooo "+s.getLocalizedMessage()); }
        }catch(Exception e){
            System.out.println("eeeeeeeerrrrrrrrrrrooooooooooorrrrrrr
"+e.getLocalizedMessage());
            p.put("men","NO NO NO SE ELIMINO
SATISFACTORIAMENTE");
            p.put("url","reg_materias.html");
```

```
                    return new ModelAndView("mensaje",p);
                    }
                p.put("men","SE ELIMINO SATISFACTORIAMENTE");
                p.put("url","reg_materias.html");
                return new ModelAndView("mensaje",p);
    }
```

II.1.13.2.28 ADMINISTRAR HORARIO



****Controlador**

```
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.HashMap;
import java.util.Iterator;
import java.util.List;
import java.util.Map;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
```

```
import javax.servlet.http.HttpSession;
import model.domain.*;
import model.manager.*;
import org.springframework.beans.factory.InitializingBean;
import org.springframework.web.servlet.ModelAndView;
import org.springframework.web.servlet.mvc.multiaction.MultiActionController;
public class administrar_horario extends MultiActionController implements
InitializingBean{
	private HoraManager horaManager;//para llamar al manager
	private CursoManager cursoManager;
	private ParaleloManager paraleloManager;
	private DiaManager diaManager;
	private MateriaManager materiaManager;
	private Paralelo_MateriaManager paralelo_materiaManager;
	private Administrar_HorarioManager administrar_horarioManager;
	private AdministrativoManager administrativoManager;
	public HoraManager getHoraManager() {
		return horaManager;
	}
	public void setHoraManager(HoraManager horaManager) {
		this.horaManager = horaManager;
	}
	public CursoManager getCursoManager() {
		return cursoManager;
```

```
	}
	public void setCursoManager(CursoManager cursoManager) {
		this.cursoManager = cursoManager;
	}
	public ParaleloManager getParaleloManager() {
		return paraleloManager;
	}
	public void setParaleloManager(ParaleloManager paraleloManager) {
		this.paraleloManager = paraleloManager;
	}
	public DiaManager getDiaManager() {
return diaManager;	}
	public void setDiaManager(DiaManager diaManager) {
		this.diaManager = diaManager;
	}
	public MateriaManager getMateriaManager() {
		return materiaManager;
	}
	public void setMateriaManager(MateriaManager materiaManager) {
		this.materiaManager = materiaManager;
	}
	public Paralelo_MateriaManager getParalelo_materiaManager() {
		return paralelo_materiaManager;
```

```
	}
	public void setParalelo_materiaManager(
				Paralelo_MateriaManager paralelo_materiaManager) {
		this.paralelo_materiaManager = paralelo_materiaManager;
	}
	public Administrar_HorarioManager getAdministrar_horarioManager() {
		return administrar_horarioManager;
	}
	public void setAdministrar_horarioManager(
Administrar_HorarioManager administrar_horarioManager) {
		this.administrar_horarioManager = administrar_horarioManager;
	}
	public AdministrativoManager getAdministrativoManager() {
		return administrativoManager;	}
	public void setAdministrativoManager(AdministrativoManager
administrativoManager) {
		this.administrativoManager = administrativoManager;
	}
	public void afterPropertiesSet() throws Exception {
		// TODO Auto-generated method stub				}
	public administrar_horario() {
		// TODO Auto-generated constructor stub	}
public ModelAndView administrar_horario(HttpServletRequest
request,HttpServletResponse response) throws Exception,ServletException {
```

```
Map p = new HashMap();
ArrayList milista = new ArrayList();
ArrayList milista2 = new ArrayList();
ArrayList milista3 = new ArrayList();
ArrayList milista4 = new ArrayList();
Long cod_hor=null;
Long cod_dia=null;
String  nom_cur=null;
String nom_par=null;
try {
        List mat = this.horaManager.getListaHorario();
        for (Iterator i = mat.iterator(); i.hasNext();) {
                Map map = new HashMap();
                Object[] row1 = (Object[]) i.next();
                cod_hor= (Long) row1[0];
                String hora= (String) row1[1];
           map.put("cod_hor",cod_hor.toString());
           map.put("hora",hora);
           milista.add(map);
        }
        p.put("datos",milista);
        List mat2 = this.cursoManager.getListaCurso();
for (Iterator i = mat2.iterator(); i.hasNext();) {
```

```
                    Map map2 = new HashMap();
                    Object[] row1 = (Object[]) i.next();
                    Long cod_cur= (Long) row1[0];
                    nom_cur= (String) row1[1];
                    Long cod_par= (Long) row1[2];
nom_par= (String) row1[3];
                    map2.put("cod_cur",cod_cur);
                    map2.put("nombre_cur",nom_cur);
                    map2.put("nom_par",nom_par);
                    milista2.add(map2);
                }
                p.put("datos2",milista2);
                List mat3 = this.diaManager.getListaDia();
                for (Iterator i = mat3.iterator(); i.hasNext();) {
                    Map map3 = new HashMap();
                    Object[] row1 = (Object[]) i.next();
                    cod_dia= (Long) row1[0];
                    String dia= (String) row1[1];
                    map3.put("cod_dia",cod_dia);
                    map3.put("dia",dia);
                    milista3.add(map3);
                }
                p.put("datos3", milista3);
```

```
                                String
select=(String)request.getParameter("TipoCurso");
                                if(request.getParameter("TipoCurso")==null) {
                                }else{
String CursoParalelo[]=select.split(",");
                nom_cur=CursoParalelo[0];
                nom_par=CursoParalelo[1];
                                        }
                    p.put("opcion_paralelo",nom_par);
                    p.put("opcion_curso", nom_cur);
                List mat4 = this.materiaManager.getListaMateria(nom_par, nom_cur);
                    for (Iterator x = mat4.iterator(); x.hasNext();) {
                                Object []row1 = (Object[])x.next();
                                Map map4 = new HashMap();
                                Long cod_mat=(Long)row1[0];
                                String nombre_mat= (String)row1[1];
                                Integer cod_horx= (Integer)row1[2];
                                Integer cod_diax= (Integer)row1[3];
                                String ci=(String)row1[4];
                                Integer cod_m_p= (Integer)row1[5];
                                map4.put("cod_mat", cod_mat);
                                map4.put("nombre_mat", nombre_mat);
                                map4.put("cod_hor",cod_horx);
                            map4.put("cod_dia",cod_diax);
```

```
                                        map4.put("ci",ci);
map4.put("cod_m_p", cod_m_p);
            milista4.add(map4);                         }
                    p.put("datos4",milista4);
            } catch (Exception e) {
                            System.out.println("MENSAJE3333333
"+e.getLocalizedMessage());
                    }
                    return new ModelAndView("administrador/administrar_horario",p);
    }
```

****Administrar_HorarioManager**

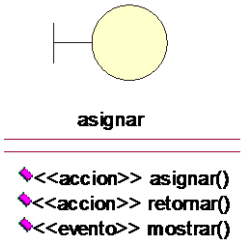
```
import model.domain.Administrar_Horario;
import org.hibernate.SessionFactory;
import java.util.*;
public interface Administrar_HorarioManager {
            public abstract SessionFactory getSessionFactory();
            public abstract void setSessionFactory(SessionFactory sessionFactory);
            public abstract List getAsignarHorario(Integer cod_m_p);
            public void saveAdministrar_Horario(Administrar_Horario m);
            public void deleteEstudiante(long cod_m_p);
             /* * public Usuarios  getUsuarios (String xci);
            public void updateUsuarios (Usuarios  m);
            public void deleteUsuarios (String xci);*/   }
```

****Administrar_HorarioManagerImpl**

```
import java.util.List;
import model.domain.Administrar_Horario;
import model.domain.Estudiante;
import model.domain.Hora;
import model.domain.Paralelo_Materia;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
public class Administrar_HorarioManagerImpl implements
Administrar_HorarioManager  {
SessionFactory sessionFactory;
        public SessionFactory getSessionFactory()    {
           return sessionFactory;
        }
        public void setSessionFactory(SessionFactory sessionFactory)
        {
           this.sessionFactory = sessionFactory;       }
   public List getAsignarHorario(Integer cod_m_p){
                        Session session = this.sessionFactory.getCurrentSession();
                        return session.createQuery("select  distinct
p.cod_par,p.nom_par,c.cod_cur,c.nombre_cur,ah.cod_m_p from
Administrar_Horario ah,Paralelo p, Paralelo_Materia pm,Curso c "+
" where  ah.cod_m_p=pm.cod_m_p and p.cod_par=pm.cod_par and
ah.cod_m_p='"+cod_m_p+"' and p.cod_cur=c.cod_cur").list();              }
        public void saveAdministrar_Horario(Administrar_Horario m){
```

```
        Session session = this.sessionFactory.getCurrentSession();
      session.merge(m);
    }
    public void deleteEstudiante(long cod_m_p){
        Session session = this.sessionFactory.getCurrentSession();
        session.getTransaction().begin();
        session.delete(session.load(Paralelo_Materia.class, new
Long(cod_m_p)));
        session.getTransaction().commit();
    }  }
```

II.1.13.2.29 ASIGNAR




```
public ModelAndView asignar_horario(HttpServletRequest
request,HttpServletResponse response) throws Exception,ServletException {
        HashMap p=new HashMap();
        try {
            p.put("nombre_cur",request.getParameter("nombre_cur"));
p.put("nom_par",request.getParameter("nom_par"));
            p.put("cod_hor",request.getParameter("cod_hor"));
            p.put("cod_dia",request.getParameter("cod_dia"));
            List mat = this.materiaManager.getListaMateria2();
```

```
                    p.put("datos",mat);
} catch (Exception e) {
                    System.out.println("ecception asignar "+e.getMessage());
                }
                return new ModelAndView("administrador/asignar_horario",p);  }
        public ModelAndView asignar_horario2(HttpServletRequest
request,HttpServletResponse response) throws Exception,ServletException {
                HashMap p=new HashMap();
                try {
                    p.put("nombre_cur",request.getParameter("xnombre_cur"));
                    p.put("nom_par",request.getParameter("xnom_par"));
                    p.put("cod_hor",request.getParameter("xcod_hor"));
                    p.put("cod_dia",request.getParameter("xcod_dia"));
                    p.put("cod_mat",request.getParameter("xcod_mat"));
                    p.put("nombre_mat",request.getParameter("xnombre_mat"));
                    List mat=this.administrativoManager.getListaAdministrativo();
                    p.put("datos", mat);
                } catch (Exception e) {
System.out.println("ecception asignar 2 "+e.getMessage());
                }
                return new ModelAndView("administrador/asignar_horario2",p);
        }
        public ModelAndView asignar_horario3(HttpServletRequest
request,HttpServletResponse response) throws Exception,ServletException {
```

```
HashMap p=new HashMap();
        try {           List
m1=this.paraleloManager.getListaParalelo(request.getParameter("xnom_par"),reques
t.getParameter("xnombre_cur"));
                        Long m2=(Long)m1.iterator().next();
                        int m3=m2.intValue();
                        Paralelo_Materia mm=new Paralelo_Materia();
                        long aleatorio=(long)(Math.random()*1000);
                        mm.setCod_m_p(aleatorio);
        m.setCod_mat(Integer.parseInt(request.getParameter("xcod_mat")));
                        mm.setGestion(2010);
                        mm.setCod_par(m3);
                        this.paralelo_materiaManager.saveParalelo_Materia(mm);
                        if(request.getParameter("xseleccion")!=null){
                                String
datoss[]=request.getParameterValues("xseleccion");
                                Administrar_Horario m=new Administrar_Horario();
m.setCod_adm_hor(aleatorio);
        m.setCod_hor(Integer.parseInt(request.getParameter("xcod_hor")));
        m.setCod_dia(Integer.parseInt(request.getParameter("xcod_dia")));
                                m.setCi(datoss[0]);
                                m.setCod_m_p((int)aleatorio);
        this.administrar_horarioManager.saveAdministrar_Horario(m);             }
                } catch(Exception s)
```

{System.*out*.println("MENSAJE asginar horario 3 "+s.getLocalizedMessage());

p.put("men","NO NO SE ASIGNO SATISFACTORIAMENTE");

p.put("url","administrar_horario.html");

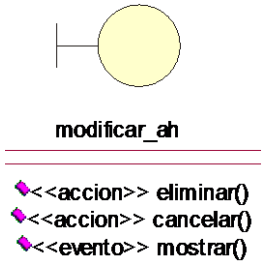
return new ModelAndView("mensaje",p); }

p.put("men","SE ASIGNO SATISFACTORIAMENTE");

p.put("url","administrar_horario.html");

return new ModelAndView("mensaje",p); }

II.1.13.2.30 MODIFICAR



public ModelAndView modificar_horario(HttpServletRequest request,HttpServletResponse response) **throws** Exception,ServletException {

HashMap p=**new** HashMap();

ArrayList milista=**new** ArrayList();

try {

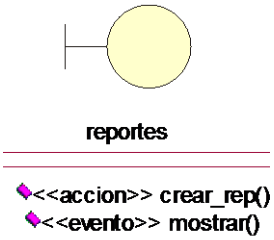
Hora mmm=**this**.horaManager.getHora(Long.*parseLong*(request.getParameter("cod_hor")));

Dia mm=**this**.diaManager.getDia(Long.*parseLong*(request.getParameter("cod_dia")));

```
                    Administrativo m =
this.administrativoManager.getAdministrativo(request.getParameter("ci"));
                    List
mat=this.administrar_horarioManager.getAsignarHorario(Integer.parseInt(request.ge
tParameter("cod_m_p")));
                    p.put("cod_mat",
Long.parseLong(request.getParameter("cod_mat")));
                    p.put("nombre_mat", request.getParameter("nombre_mat"));
                    p.put("mmm",mmm);
                    p.put("mm",mm);
                    p.put("m",m);
                .put("cod_m_p",Integer.parseInt(request.getParameter("cod_m_p")));
                    for (Iterator i=mat.iterator();i.hasNext();) {
                        Object filas[]=(Object[])i.next();
                        HashMap map=new HashMap();
                        map.put("cod_par",(Long)(filas[0]));
map.put("nom_par",filas[1]);
map.put("cod_cur", (Long)filas[2]);
                        map.put("nombre_cur",filas[3]);
                        map.put("cod_m_p",(Integer)filas[4]);
                        milista.add(map);                    }
                    p.put("datos", milista);
                    } catch (Exception e) {
                    System.out.println("ecception "+e.getMessage());
```

```
        }
        return new ModelAndView("administrador/modificar_horario",p);
    }
```

II.1.13.2.31 REPORTES

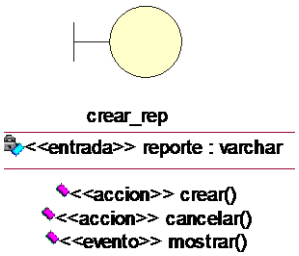



```
    public ModelAndView Reportes(HttpServletRequest
request,HttpServletResponse response) throws Exception,ServletException {
        Map p = new HashMap();
        ArrayList milistax = new ArrayList();
        Map mapx = new HashMap();
        String datofiltro="";
        String estado="1";
        int rango=1;
        String dato=request.getParameter("xdatofiltro");
        System.out.println("este es el dato"+
request.getParameter("xdatofiltro"));
            if(request.getParameter("xdatofiltro")==null){
                datofiltro="";
            }else{
                datofiltro=request.getParameter("xdatofiltro");
    }
```

```
                    if(request.getParameter("xinter")==null){
                        rango=1;
                    }else{
        rango=Integer.parseInt(request.getParameter("xinter"));}
                    if(request.getParameter("xopcion")==null){
                        estado="1";
                    }else{
                        estado=request.getParameter("xopcion");              }
                try {                           List Total =
this.materiaManager.getListaCursosFiltroTotal(datofiltro,Integer.parseInt(estado));
                    double num=Total.size();
                    double xx=Math.ceil(num / 5);
if(rango>1){
mapx = new HashMap();
mapx.put("rango","<a id='x' onMouseOver='Over(this.id)' onMouseOut='Out(this.id)'
href='crear report.html?xdatofiltro="+datofiltro+"&xinter="+(rango-
1)+"&xopcion="+estado+"'>"+"<<<"+"</a>");
                        milistax.add(mapx);                    }
                    for(int i=1;i<=xx;i++){
                        mapx = new HashMap();
                        if(i==rango){
                            mapx.put("rango",""+i+"");
                        }else{                          mapx.put("rango","<a
id="'+i+"' onMouseOver='Over(this.id)' onMouseOut='Out(this.id)' href= crear
```

```
report..html?xdatofiltro="+datofiltro+"&xinter="+i+"&xopcion="+estado+"'>"+i+"</
a>");    }
                                    milistax.add(mapx);                    }
                            if(rango<xx){
                                    mapx = new HashMap();
                                    mapx.put("rango","<a id='y'
onMouseOver='Over(this.id)' onMouseOut='Out(this.id)' href= crear
report..html?xdatofiltro="+datofiltro+"&xinter="+(rango+1)+"&xopcion="+estado+"
'>"+">>>"+"</a>");
                                    milistax.add(mapx);            }
List mat = is. crear report.Manager.getLista crear
report.Filtro(datofiltro,rango,Integer.parseInt
(estado));
System.out.println("mlk   "+mat.size());
p.put("datos", mat);
                            p.put("intervalo",milistax);
                    }  catch(Exception s){System.out.println("MENSAJE error aqui ");}
                    if(estado.equals("1")){
                            p.put("xestadox"," ");
                            p.put("yestadoy","checked");
                    }else{                    p.put("xestadox","checked");
                            p.put("yestadoy"," ");            }
                    p.put("datofiltradox",datofiltro);
            return new ModelAndView("materias/reg_materias",p);    }
```

II.1.13.2.32 CREAR_REP




```
public ModelAndView Reportes(HttpServletRequest request,HttpServletResponse
response) throws Exception,ServletException {
                Map p = new HashMap();
                ArrayList milistax = new ArrayList();
                Map mapx = new HashMap();
                String datofiltro="";
                String estado="1";
                int rango=1;
                String dato=request.getParameter("xdatofiltro");
                System.out.println("este es el dato"+
request.getParameter("xdatofiltro"));
                        if(request.getParameter("xdatofiltro")==null){
                                datofiltro="";
                        }else{
                                datofiltro=request.getParameter("xdatofiltro");
        }
                        if(request.getParameter("xinter")==null){
                                rango=1;
```

```
                    }else{
        rango=Integer.parseInt(request.getParameter("xinter"));}
                    if(request.getParameter("xopcion")==null){
                        estado="1";
                    }else{
                        estado=request.getParameter("xopcion");              }
                try {                           List Total =
this.materiaManager.getListaCursosFiltroTotal(datofiltro,Integer.parseInt(estado));
                    double num=Total.size();
                    double xx=Math.ceil(num / 5);
if(rango>1){
mapx = new HashMap();
mapx.put("rango","<a id='x' onMouseOver='Over(this.id)' onMouseOut='Out(this.id)'
href='crear report.html?xdatofiltro="+datofiltro+"&xinter="+(rango-
1)+"&xopcion="+estado+"'>"+"<<<"+"</a>");
                        milistax.add(mapx);                    }
                    for(int i=1;i<=xx;i++){
                        mapx = new HashMap();
                        if(i==rango){
                            mapx.put("rango",""+i+"");
                        }else{                          mapx.put("rango","<a
id="'+i+"' onMouseOver='Over(this.id)' onMouseOut='Out(this.id)' href= crear
report..html?xdatofiltro="+datofiltro+"&xinter="+i+"&xopcion="+estado+"'>"+i+"</
a>");    }
                        milistax.add(mapx);                    }
```

```
                    if(rango<xx){
                            mapx = new HashMap();
                            mapx.put("rango","<a id='y'
onMouseOver='Over(this.id)' onMouseOut='Out(this.id)' href= crear
report..html?xdatofiltro="+datofiltro+"&xinter="+(rango+1)+"&xopcion="+estado+"
'>"+">>>"+"</a>");
                            milistax.add(mapx);          }
List mat = is. crear report.Manager.getLista crear
report.Filtro(datofiltro,rango,Integer.parseInt
(estado));
System.out.println("mlk   "+mat.size());
p.put("datos", mat);
                    p.put("intervalo",milistax);
                }  catch(Exception s){System.out.println("MENSAJE error aqui ");}
                if(estado.equals("1")){
                    p.put("xestadox"," ");
                    p.put("yestadoy","checked");
                }else{                    p.put("xestadox","checked");
                    p.put("yestadoy"," ");          }
                p.put("datofiltradox",datofiltro);
            return new ModelAndView("crear report ",p);    }
```

II.1.14 CASOS DE PRUEBA

II.1.14.1 INTRODUCCION

Cada prueba es especificada mediante un documento que establece las condiciones de ejecución como las entradas y los resultados esperados. Estos casos son aplicados como pruebas de regresión en cada iteración. Cada caso llevara asociado un procedimiento de prueba con las instrucciones para realizar las mismas y dependiendo del tipo que se utilice, este procedimiento podrá ser automatizable mediante un script.

II.1.14.2 DEFINICION

La prueba es un proceso de ejecución de un programa con la intención de descubrir errores.

Un buen caso de prueba es aquel que tiene una alta probabilidad de mostrar un error desconocido hasta entonces.

Todos los productos de Software son probados de dos formas:

- Conociendo la función específica para la que fue diseñado el producto, se puede llevar a cabo pruebas que demuestren que cada función es completamente operativa denominada prueba de caja negra.
- Conociendo el funcionamiento del producto, se pueden realizar pruebas que aseguren que la especificación interna se ajuste a las especificaciones y que todos los componentes internos se hayan comprobado de forma adecuada, esta forma se denomina prueba de caja blanca.

Para la prueba al sistema se utilizara prueba de caja negra.

II.1.14.3 PRUEBA DE CAJA NEGRA.

La prueba funcional o de caja negra se centra en el estudio de la especificación del software, del análisis de las funciones que debe realizar de las entradas y de las salidas.

Los errores que pretenden detectar mediante las pruebas de caja negra son:

- Funcionalidad incorrecta o ausente.
- Errores de rendimiento.
- Errores de interfaces.
- Errores en la estructura de datos.
- Errores de inicialización y terminación.

Para realizar las pruebas al sistema, se utilizara el método de particiones o clases de equivalencia a todos aquellos formularios que sean necesarios y tengan entrada de datos. El diseño de casos de este método consiste en:

- Identificación de clases de equivalencia.
- Creación de los casos de prueba correspondientes.

Para identificar las posibles clases de equivalencia de un programa a partir de su especificación se deben seguir los siguientes pasos:

- Identificación de las condiciones de las entradas del programa, es decir, restricciones de formato o contenido de los datos de entrada.
- A partir de ellas se identifican las clases de equivalencia que pueden ser:
    - De datos validos
    - De datos no validos o erróneos.

Técnica: Partición Equivalente

II.1.14.4 PRUEBA DE CAJA NEGRA AL SISTEMA

II.1.14.4.1 INICIAR

Datos contenidos en una aplicación de automatización, el software captura los datos de la siguiente forma:

Datos:

Usuario: letra, tamaño 10

Clave: letra, tamaño 10

Condiciones de Entrada	Clases Equivalencia Validas	Clases Equivalencia Invalidas
Usuario especificado	**1**. Si	**2**. No
Usuario	**3**. Letra	**4**. Cualquier otro carácter

Tamaño de Usuario	**5**. Valor <=10	**6**. <=0
Clave especificada	**7**. Si	**8**. No
Contraseña	**9**. Letra	**1**0. Cualquier otro carácter
Tamaño de Contraseña	**1** Valor <=10	**1**. <=0

Identificación de los Casos de Prueba que sean uno o más clases de equivalencia Clases Válidas

Cp.	**Usuario**	**Clave**
	juan	jurad

Cubre las clases de equivalencia válidas: 1-3-5-7-9-11

Clases No Válidas

Cp.	**Usuario**	**Clave**
		jurad

Cubre las clases de equivalencia no válida:2-4-6

Cp.	**Usuario**	**Clave**
	--..:) ☺	++ -☺☺

Clases: 4-6-10-12

II.1.14.4.2 REG ESTUDIANTES

II.1.14.4.2.1 Adicionar

Datos contenidos en una aplicación de automatización de Administración de estudiantes, el software captura los datos de la siguiente forma:

Datos:

rude: número.

CI: letra, de 8 dígitos

Nombre: letra, tamaño 20

Ap.: letra, tamaño 20

Ap.: letra, tamaño 20

Direccion:

Nº número, tamaño 10

Calle: letra, tamaño 30

Zona: letra, tamaño 30

Teléfono: letra, tamaño 20

Condiciones de Entrada	**Clases Equivalencia Validas**	**Clases Equivalencia Invalidas**
rude especificado	**1.** Si	**2.** No
CI	**3.** Número	**4.** Cualquier otro carácter
Tamaño de CI	**5.** 8	**6.** ¡!=7
Nombre	**7.** Letra	**8.** Cualquier otro carácter
Tamaño de Nombre	**9.** Valor <=20	**10.** >20
Ap.	**11.** Letra	**12.** Cualquier otro carácter
Tamaño de Ap.	**13.** Valor <=20	**14.** >20
Am.	**15.** Letra	**16.** Cualquier otro carácter
Tamaño de Am.	**17.** Valor <=20	**18.** >20
Domicilio	**19.** Numero	**20.** Cualquier otro carácter
Tamaño de Nº	**21.** Valor <=10	**22.** >10
Calle	**23.** Letra	**24.** Cualquier otro carácter
Tamaño de Calle	**25.** Valor <=30	**26.** >30
Zona	**27.** Letra	**28.** Cualquier otro carácter

Tamaño de Zona	**29.** Valor <=30	**30.** >30
Teléfono	**31.** Letra y numero	**32.** Cualquier otro carácter
Tamaño de teléfono	**33.** Valor <=20	**34.** >20

Identificación de los Casos de Prueba que sean uno o más clases de equivalencia Clases Válidas

rude	**CI**	**nombre**	**Ap**	**Am.**	**Domicilio**	**Calle**	**Zona**	**Teléfono**
11111	7654321	Laura	Ojeda	Quispe	07654	España	El Tejar	66-61668

Cubre las clases de equivalencia válidas: 1-3-5-7-9-11-13-15-17-19-21-23-25-27-29-31-33

Clases No Válidas

rude	**CI**	**nombre**	**Ap**	**Am**	**Nº Domicilio**	**Calle**	**Zona**	**Teléfono**
				Quispe		España		

Cubre las clases de equivalencia no válida:2-4-6-8-10-12-14-20-22-28-30-32-34

rude	**CI**	**nombre**	**Ap**	**Am**	**Nº**	**Calle**	**Zona**	**Teléfono**
	hola	…☺,..	*****be	☺☺☺☺****	No se	(>☺<)	;) ;;;;	***{{,,,

Clases: 4-6-8-10-12-14-16-18-20-22-24-26-28-30-32-34

II.1.14.4.2.2 Modificar

Datos contenidos en una aplicación de automatización de modificar estudiantes, el software captura los datos de la siguiente forma:

Datos:

Nombre: letra, tamaño 20

Ap.: letra, tamaño 20

Ap.: letra, tamaño 20

Direccion:

Nº número, tamaño 10

Calle: letra, tamaño 30

Zona: letra, tamaño 30

Teléfono: letra, tamaño 20

Condiciones de Entrada	Clases Equivalencia Validas	Clases Equivalencia Invalidas
Tamaño de Ap. Paterno	**1.** Valor <=20	**2.** >20
Ap. Materno	**3.** Letra	**4.** Cualquier otro carácter
Tamaño de Ap. Materno	**5.** Valor <=20	**6.** >20
Nº Domicilio	**7.** Numero	**8.** Cualquier otro carácter
Tamaño de Nº Domicilio	**9.** Valor <=5	**10** >5
Calle	**11** Letra	**12** Cualquier otro carácter
Tamaño de Calle	**13** Valor <=30	**14** >30
Zona	**15** Letra	**16** Cualquier otro carácter
Tamaño de Zona	**17** Valor <=50	**18** >50
Teléfono	**19** Letra y numero	**20** Cualquier otro carácter
Tamaño de teléfono	**21** Valor <=30	**22** >30

Identificación de los Casos de Prueba que sean uno o más clases de equivalencia Clases Válidas

Cp.	nombre	Ap	Am.	Domicilio	Calle	Zona	Teléfono
	Laura	Ojeda	Quispe	07654	España	El Tejar	66-61668

Cubre las clases de equivalencia válidas: 1-3-5-7-9-11-13-15-17-19-21

Clases No Válidas

Cp.	nombre	Ap	Am	Nº Domicilio	Calle	Zona	Teléfono
			Quispe		España		

Cubre las clases de equivalencia no válida:2-4-6-8-10-12-14-16-18-20-22

Cp.	rude	CI	nombre	Ap	Am	Nº Domicilio	Calle	Zona	Teléfono
		hola	…☺,..	;) *****be	☺☺☺☺****	No se	(>☺<)	;) ;;;;	***{{,,,

Clases: 2-4-6-8-10-12-14-16-18-20-22

II.1.14.4.3 MEMORANDUS

II.1.14.4.3.1 adicionar_m

Datos contenidos en una aplicación de automatización de creación de memorandus, el software captura los datos de la siguiente forma:

Datos:

nombre: letra, tamaño 40

Condiciones de Entrada	Clases Equivalencia Validas	Clases Equivalencia Invalidas
nombre	**1.** si	**2.** no
nombre	**3.** Letra	**4.** Cualquier otro carácter
Tamaño de nombre	**5.** Valor <=40	**6.** >40

Identificación de los Casos de Prueba que sean uno o más clases de equivalencia Clases Válidas

Cp.	nombre
	Pelea con sus compañeros

Cubre las clases de equivalencia válidas: 1-3-5

Clases No Válidas

Cp.	nombre

Cubre las clases de equivalencia no válida: 2-6

Cp.	nombre
	;) ;) ☺☺☺***☺☺☺ ;) ;)

Clases: 2-4

II.1.14.4.3.2 Modificar_m

Datos contenidos en una aplicación de automatización de modificacion de memoramdus, el software captura los datos de la siguiente forma:

Datos:

nombre: letra, tamaño 40

Condiciones de Entrada	Clases Equivalencia Validas	Clases Equivalencia Invalidas

nombre	**7.** si	**8.** no
nombre	**9.** Letra	**10** Cualquier otro carácter
Tamaño de nombre	**11** Valor <=40	**12** >40

Identificación de los Casos de Prueba que sean uno o más clases de equivalencia Clases Válidas

Cp.	**nombre**
	Pelea con sus compañeros

Cubre las clases de equivalencia válidas: 1-3-5

Clases No Válidas

Cp.	**nombre**

Cubre las clases de equivalencia no válida: 2-6

Cp.	**nombre**
	;) ;) ☺☺☺***☺☺☺ ;) ;)

Clases: 2-4

II.1.14.4.4.4 REG_ADMINISTRATIVO

II.1.14.4.4.4.1 Adicionar

Datos contenidos en una aplicación de automatización de Administración de administrativos, el software captura los datos de la siguiente forma:

Datos:

rude: número.

CI: letra, de 8 dígitos

Nombre: letra, tamaño 20

Ap.: letra, tamaño 20

Ap.: letra, tamaño 20

Direccion:

Nº número, tamaño 10

Calle: letra, tamaño 30

Zona: letra, tamaño 30

Teléfono: letra, tamaño 20

Condiciones de Entrada	**Clases Equivalencia Validas**	**Clases Equivalencia Invalidas**
rda especificado	**1.** Si	**2.** No
CI	**3.** Número	**4.** Cualquier otro carácter
Tamaño de CI	**5.** Valor <=8	**6.** ¡= 8
Nombre	**7.** Letra	**8.** Cualquier otro carácter
Tamaño de Nombre	**9.** Valor <=20	**10.** >20
Ap.	**11.** Letra	**12.** Cualquier otro carácter
Tamaño de Ap.	**13.** Valor <=20	**14.** >20
Am	**15.** Letra	**16.** Cualquier otro carácter
Tamaño de Am	**17.** Valor <=20	**18.** >20
F. Nacimiento	**19.** Fecha	**20.** Cualquier otro carácter
Nº	**21.** Numero	**22.** Cualquier otro carácter
Tamaño de Nº	**23.** Valor <=10	**24.** >10
Calle	**25.** Letra	**26.** Cualquier otro carácter
Tamaño de Calle	**27.** Valor <=20	**28.** >20

Zona	**29.** Letra	**30.** Cualquier otro carácter
Tamaño de Zona	**31.** Valor <=30	**32.** >30
Teléfono	**33.** Letra y numero	**34.** Cualquier otro carácter
Tamaño de teléfono	**35.** Valor <=40	**36.** >40

Identificación de los Casos de Prueba que sean uno o más clases de equivalencia Clases Válidas

rda	**CI**	**nombre**	**Ap**	**Am.**	**Domicilio**	**Calle**	**Zona**	**Teléfono**
11111	7654321	Laura	Ojeda	Quispe	07654	España	El Tejar	66-61668

Cubre las clases de equivalencia válidas: 1-3-5-7-9-11-13-15-17-19-21-23-25-27-29-31-33-35

Clases No Válidas

Cp.	**rda**	**CI**	**nombre**	**Ap**	**Am**	**Nº Domicilio**	**Calle**	**Zona**	**Teléfono**
					Quispe		España		

Cubre las clases de equivalencia no válida:2-4-6-8-10-12-14-20-22-28-30-32-34-36

rude	**CI**	**nombre**	**Ap**	**Am**	**Nº Domicilio**	**Calle**	**Zona**	**Teléfono**
	hola	…☺,..	;) *****	☺☺☺☺****	No se	(>☺<)	;) ;;;;	***{{,,,

Clases: 4-6-8-10-12-14-16-18-20-22-24-26-28-30-32-34

II.1.14.4.4.4.2 Modificar

Datos contenidos en una aplicación de automatización de modificación de administrativos, el software captura los datos de la siguiente forma:

Datos:

rude: número.

CI: letra, de 8 dígitos

Nombre: letra, tamaño 20

Ap.: letra, tamaño 20

Ap.: letra, tamaño 20

Direccion:

Nº número, tamaño 10

Calle: letra, tamaño 30

Zona: letra, tamaño 30

Teléfono: letra, tamaño 20

Condiciones de Entrada	Clases Equivalencia Validas	Clases Equivalencia Invalidas
Nombre	**1.** Letra	**2.** Cualquier otro carácter
Tamaño de Nombre	**3.** Valor <=30	**4.** >30
Ap. Paterno	**5.** Letra	**6.** Cualquier otro carácter
Tamaño de Ap. Paterno	**7.** Valor <=20	**8.** >20
Ap. Materno	**9.** Letra	**10** Cualquier otro carácter
Tamaño de Ap. Materno	**11** Valor <=20	**12** >20
Nº Domicilio	**13** Numero	**14** Cualquier otro carácter
Tamaño de Nº Domicilio	**15** Valor <=5	**16** >5
Calle	**17** Letra	**18** Cualquier otro carácter
Tamaño de Calle	**19** Valor <=30	**20** >30
Zona	**21** Letra	**22** Cualquier otro carácter
Tamaño de Zona	**23** Valor <=50	**24** >50
Teléfono	**25** Letra y numero	**26** Cualquier otro carácter
Tamaño de teléfono	**27** Valor <=30	**28** >30

Identificación de los Casos de Prueba que sean uno o más clases de equivalencia Clases Válidas

Cp.	nombre	Ap	Am.	Domicilio	Calle	Zona	Teléfono
	Laura	Ojeda	Quispe	07654	España	El Tejar	66-61668

Cubre las clases de equivalencia válidas: 1-3-5-7-9-11-13-15-17-19-21-23-25-27

Clases No Válidas

Cp.	nombre	Ap	Am	Nº Domicilio	Calle	Zona	Teléfono
			Quispe		España		

Cubre las clases de equivalencia no válida:2-4-6-8-10-12-14-20-22-28

Cp.	nombre	Ap	Am	Nº Domicilio	Calle	Zona	Teléfono
	…☺,..	;) *****be	☺☺☺☺****	No se	(>☺<)	;) ;;;;	***{{,,,

Clases: 4-6-8-10-12-14-16-18-20-22-24-26-28

II.1.14.4.4.5 REG MATERIAS

II.1.14.4.4.5.1 Adicionar_mat

Datos contenidos en una aplicación de automatización de Administración de registro de materias, el software captura los datos de la siguiente forma:

Datos:

sigla: letra, tamaño 20

nombre: letra, tamaño 20

horas_acad: integer

Condiciones de Entrada	Clases Equivalencia Validas	Clases Equivalencia Invalidas
sigla	**1.** Si	**2.** No
sigla	**3.** Letra	**4.** Cualquier otro carácter
Tamaño de sigla	**5.** Valor <=20	**6.** >20

II.1.15 LISTA DE RIESGOS

II.1.15.1 Introducción

Son los riesgos identificados a lo largo del desarrollo del sistema que se podrá minimizar si se sigue con las estrategias de atenuación.

II.1.15.1.1 Propósito

Identificar los posibles riesgos que puede haber en el desarrollo e implementación del sistema.

II.1.15.1.2 Alcance

Garantizar que los riesgos se minimicen durante el periodo de desarrollo del Sistema.

II.1.15.2 Lista de Riesgos

Riesgo	**Acción**
Vago conocimiento de uno de las herramientas utilizadas a lo largo del proyecto.	Por ejemplo: No manejar la notación U.M.L. resultado: Retraso en la presentación.
Falta de coordinación en los nombres de los métodos y los casos de uso	Métodos no coordinan a la hora de ejecutar el sistema
No terminar todos los componentes, no entregar el proyecto en tiempo estimado	No es posible presentar en el tiempo previsto el proyecto. Proyecto incompleto

Errores en el producto	Errores encontrados en el código del software.

Tabla 54. Lista de Riesgos

II.1.15.2.1 GESTION DE RIESGOS

Riesgo	Descripción	Probabi-lidad %	Impacto	Plan frente al riesgo	Supervisión
Vago conocimiento de uno de las herramientas utilizadas a lo largo del proyecto.	**Por ejemplo: el no manejar correctamente la notación U.M.L.**	**20%**	**Retraso en la presentación.**	**Buscar capacitación acerca del manejo de la herramienta.**	**Director de proyecto.**
Falta de coordinación en los nombres de los métodos y los casos de uso	**Métodos no coordinan a la hora de ejecutar el sistema**	**50%**	**Proyecto incompleto**	**Mejorar coordinación de los nombres de los métodos.**	**Director de proyecto.**
No terminar todos los componentes,	**No es posible presentar en el tiempo**	**20%**	**Retraso en la presentación.**	**Coordinar mejor el tiempo de**	**Director de proyecto.**

no entregar el proyecto en tiempo estimado	**previsto el proyecto. Proyecto incompleto**			**desarrollo del proyecto**	
Errores en el producto	**Se presenta errores en el código del sistema**	**30%**	**Bajo nivel en el rendimiento del producto.**	**Realizar pruebas constantes sistema**	**Director de proyecto.**

Tabla 55. Lista de Riesgos 2

II.1.15.2.2 Análisis de Riesgos

Riesgo y Descripción	**Ponderación y Efecto**	**Estrategias de Atenuación**	**Estrategias de Anulación**	**Plan de Contingencia**	**Políticas de Supervisión y Seguimiento**
Vago conocimiento de uno de las herramientas utilizadas a lo largo del proyecto. **Tipo** : Negocio	50% Moderado	Actualización de los conocimientos		mejorar los conocimientos respeto al rol a desarrollar	Capacitar constantemente
Falta de coordinación	65%	Reducción al mínimo	Mejorar la	Capacitación en el	Mejorar la

en los nombres de los métodos y los casos de uso **Tipo**: Negocio	Alto	los problemas de comunicación	coordinación de los diagramas	rol que se maneja	organización
No terminar todos los componentes, no entregar el proyecto en tiempo estimado **Tipo**: Proyecto	90% Catastrófico	Minimizar complejidad de Componentes a desarrollar	Establecer prioridad de componentes	Reutilizar componentes ya creados	Controlar constantemente la calendarización para el desarrollo de componentes.
Errores en el producto **Tipo**: Producto	20% Bajo	Utilizar software que realiza corrección de ciertos errores	Realizando iteraciones en el proyecto	Realizar pruebas.	Revisiones periódicas en el producto, pruebas

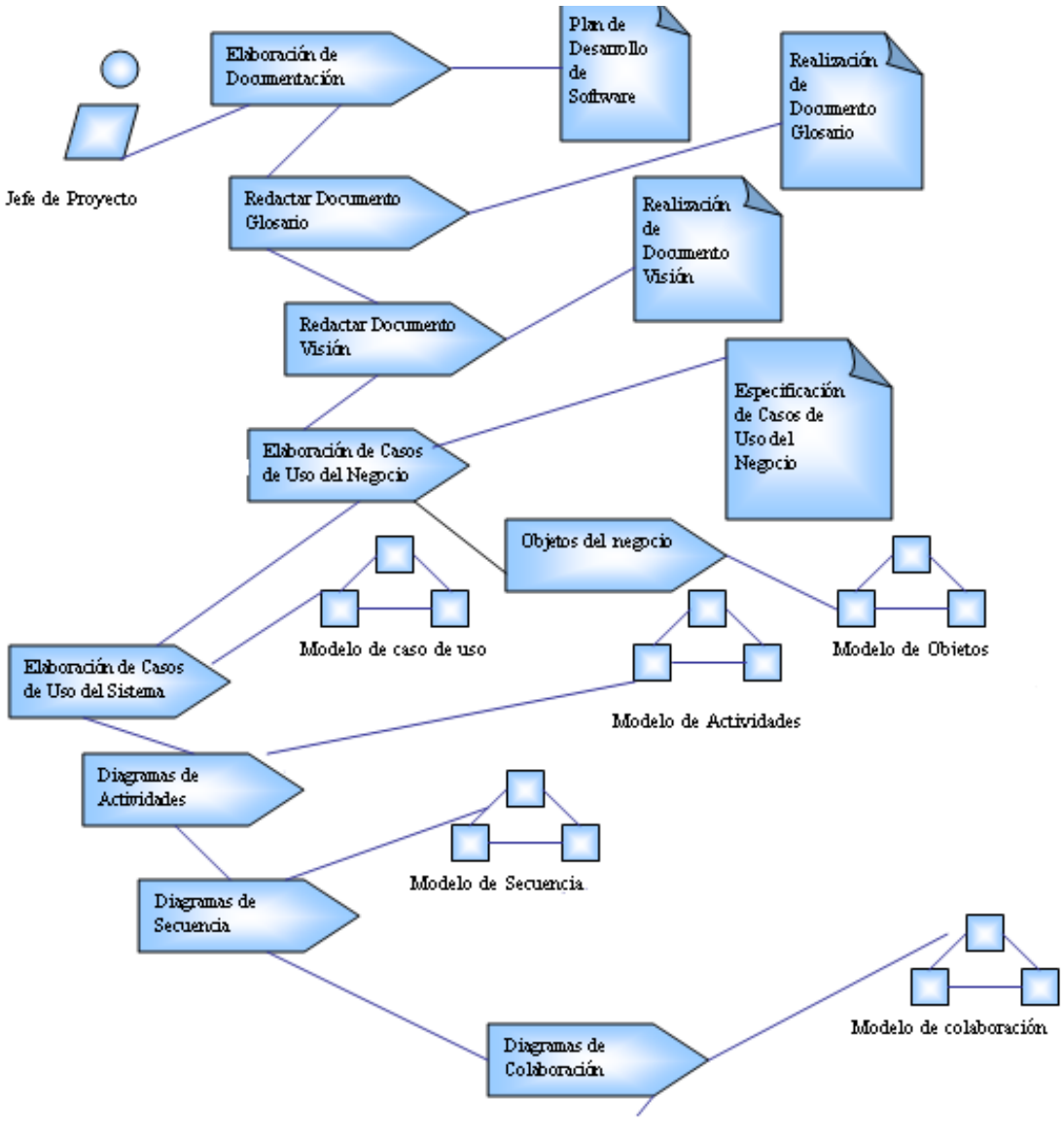
Tabla 56. Analisis de Riesgos

II.1.16 PLAN DE ITERACION

II.1.16.1 Introducción

Es un conjunto de actividades y tareas ordenadas temporalmente, con recursos asignados, dependencias entre ellas. Se realiza para cada iteración, y para todas las fases.

II.1.16.2 Iteración



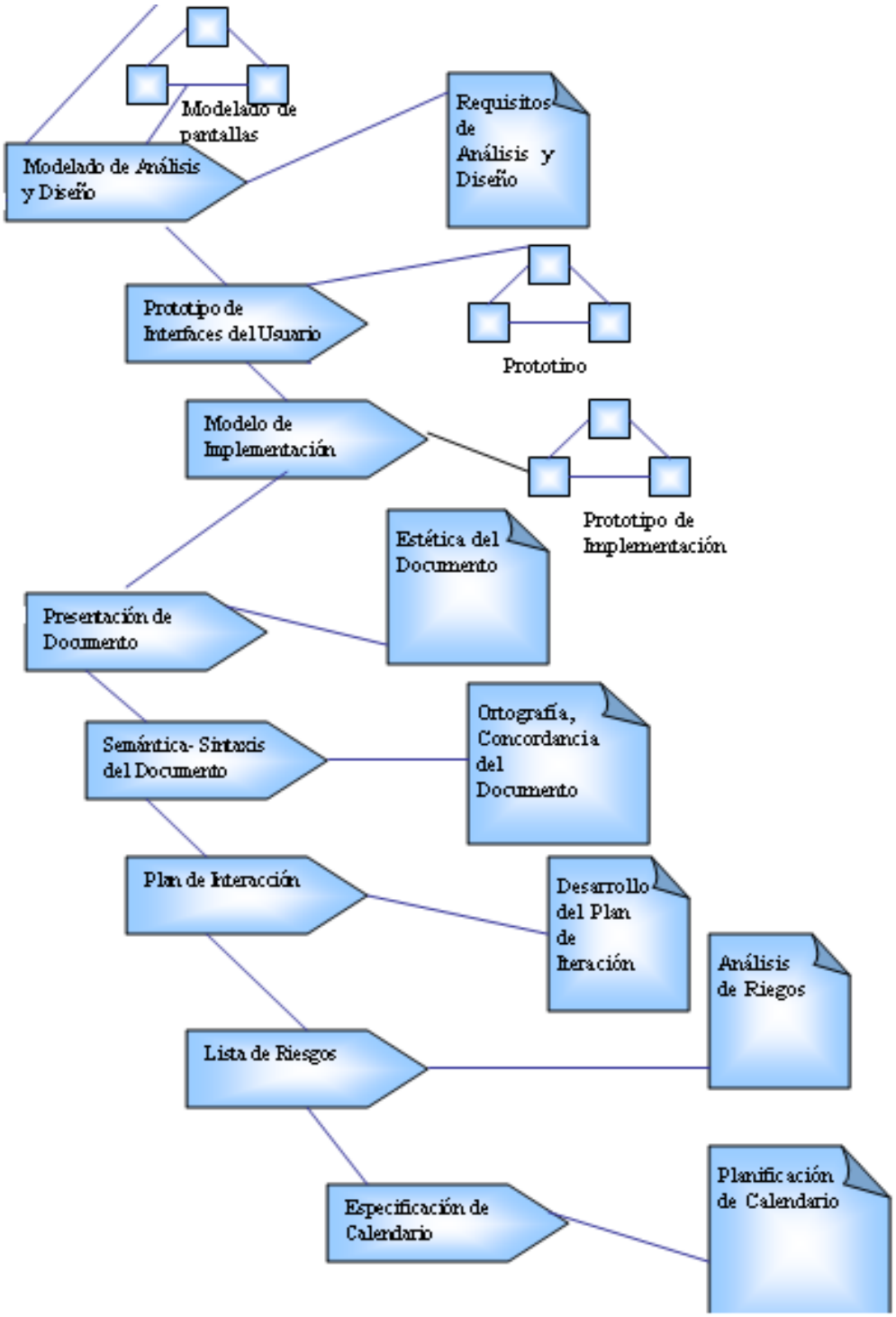


Figura 169. Plan de Iteracion

II.1.16.3 Evaluación de Iteración

Este documento incluye le evaluación de los resultados de cada iteración, el grado en el cual se han conseguido los objetivos de la iteración, las lecciones aprendidas y los cambios a ser realizados.

Rol	**Nombre de Actividad**	**Artefacto**	**Estado**	**Detalle de Cumplimiento**
Director de Proyecto	Elaboración de Documentación	Plan de desarrollo de software	En desarrollo	Se está cumpliendo de acuerdo a lo establecido
Director de Proyecto	Redacción de Documento Glosario	Realización de Documento Glosario	En desarrollo	Se está cumpliendo de acuerdo a lo establecido
Director de Proyecto	Redacción de Documento Visión	Realización de Documento Visión	En desarrollo	Se está cumpliendo de acuerdo a lo establecido
Director de Proyecto	Casos de Uso de Negocio	Especificación de Casos de Uso del Negocio	Terminado sujeto a modificaciones	Se está cumpliendo de acuerdo a lo establecido
Director de Proyecto	Objetos del Negocio	Modelo de Objetos	Terminado sujeto a modificaciones	Se está cumpliendo de acuerdo a lo

				establecido
Director de Proyecto	Diagramas de Secuencia	Modelado de secuencia	En desarrollo	Se está realizado modificaciones a lo planeado debido a cambios en los nombres de los métodos porque no concordaban
Director de Proyecto	Diagramas de colaboración	Modelado de Colaboración	En desarrollo	Se está realizado modificaciones a lo planeado debido a cambios en los nombres de los métodos porque no concordaban
Director de Proyecto	Modelado de Análisis y Diseño	Modelado de Pantallas	En Desarrollo	Se está realizado modificaciones a lo planeado debido a cambios en los nombres de los métodos porque no concordaban
Director de	Modelado de Análisis y	Requisitos de Análisis y	En desarrollo	Se está cumpliendo de

Proyecto	Diseño	Diseño		acuerdo a lo establecido
Director de Proyecto	Prototipo de Interfaz del Usuario	Prototipo	En Desarrollo	Se está realizado modificaciones a lo planeado debido a cambios en los nombres de los métodos porque no concordaban
Director de Proyecto	Modelado de Implementación	Prototipo de Implementación	En Desarrollo	Se esta cumpliendo de acuerdo a lo establecido
Director de Proyecto	Presentación del Documento	Estética del Documento	En Desarrollo	Se esta cumpliendo de acuerdo a lo establecido
Director de Proyecto	Semántica-Sintaxis del Documento	Ortografía, Concordancia del Documento	En Desarrollo	Se esta cumpliendo de acuerdo a lo establecido
Director de Proyecto	Plan de Interacción	Desarrollo del Plan de Iteración	En Desarrollo	Se a realizado modificaciones a lo planeado debido a cambios

				realizados en los elementos de control.
Director de Proyecto	Lista de Riesgos	Análisis de Riegos	En Desarrollo	Se esta cumpliendo de acuerdo a lo establecido
Director de Proyecto	Especificación de Calendario	Planificación de Calendario	En Desarrollo	Se esta cumpliendo de acuerdo a lo establecido

Figura 170. Evaluacion de iteracion

II.2 Personal capacitado en el uso del software

II.2.1 Capacitación

En la actualidad la capacitación de los recursos humanos en la respuesta al necesidad que tienen las empresas o instituciones de contar con un personal calificado y productivo.

La obsolescencia también es una de las razones por la cual, las instrucciones se preocupan por capacitar a sus recursos humanos, pues esta procura actualizar sus conocimientos con las nuevas técnicas y métodos de trabajo que garantizan eficiencia.

Los procesos de capacitación son recibidos en muchas ocasiones como una oportunidad de crecimiento y de aprendizaje con el fin de, no sólo de mejorar la tarea y el desempeño para el cual hemos sido contratados, sino también para crecer como personas, para interiorizar contenidos que quizá no tenga aplicación inmediata pero que dan temple y seguridad para las oportunidades futuras.

II.2.2 Importancia de la Capacitación

A mayor desarrollo tecnológico en la sociedad, mayor necesidad de talento, de personas competentes técnica y emocionalmente capaces de crear, innovar, crear valor, afrontar retos en los negocios, elaborar bienes y servicios de calidad y contribuyan a que la organización aprenda a mantenerse en un mercado globalizado, la tendencia es que las organizaciones se conviertan en comunidades de aprendizaje que lo generen, lo conserven y lo traduzcan en acciones de valor agregado, la sobrevivencia en el mundo global y competitivo depende, en estos momentos, de la inversión que hagan las empresas en intangibles, como innovación tecnológica, organización flexible y desarrollo de capital humano. Así, la utilización del conocimiento apropiado se convierte en la principal fuente de ventaja competitiva para una organización en la época actual.

II.2.3 Metodología DICE

DICE es una metodología dirigida especialmente para los responsables de capacitación en las organizaciones, que permite fortalecer las competencias de quienes tienen esa responsabilidad. Esta metodología integra cuatro elementos importantes que son Diagnostico, Intervención, Comprobación, Evaluación.

Como bien se menciono anteriormente esta metodología es para la capacitación del personal de organizaciones o empresas, a si que se trato de adecuar esta metodología para efectuar la capacitación de las autoridades de la Unidad Educativa (director, secretarias).

II.2.3.1 Diagnostico

Es el análisis de necesidades de capacitación, permite al responsable de capacitación encontrar no solamente los temas sobre los cuales se debe trabajar, sino también permite definir la profundidad requerida para cada uno de los temas, establecer la intensidad horaria de cada intervención, definir los grupos de participantes, es decir nos permite estructurar la capacitación.

En base a los requerimientos expresados por el director y las secretarias se realizo el presente proyecto, además se pudo constatar que no existe un buen conocimiento sobre el uso de los equipos de computación, que en realidad se tienen pero no se utilizan por lo mismo.

II.2.3.1.1 Estructura de la capacitación

II.2.3.1.1.1 Objetivos

El objetivo central de la capacitación es capacitar a las autoridades educativas (Director, secretaria) de las Unidades Educativas de la Red Belgrano del distrito Cercado en el Uso del sistema y las aplicaciones que se requieren para el mismo.

Permitiendo a los participantes de la capacitación conocer nuevas tecnologías que faciliten su trabajo en lo que se refiere a la administración.

II.2.3.1.2 Cronograma

Presentación del proyecto a las Unidades Educativas pertenecientes a la Red Belgrano

Viernes 29 de julio de 2011 10:00 a.m.

Viernes 29 de julio de 2011 16:00 p.m.

Laboratorio de Informática de la Unidad Educativa Gral. José Manuel Belgrano

Tiempos de Ejecución: 2 horas por turno

II.2.3.1.3 Dirigido a:

La capacitación estuvo dirigida a las Autoridades Educativas de la red Belgrano, a los directores y secretarias como también a los profesores que quisieran asistir sin limitación de la participación de otras personas. Como bien se dijo la capacitación estuvo dirigida a profesionales pero no se limito la participación de otras personas.

II.2.3.2 Intervención

La intervención entonces es la ejecución de los programas de capacitación que abarca diferentes escenarios de acción, En el tiempo de intervención es preciso tener en cuenta también la disponibilidad de los participantes.

En el proceso de intervención o desarrollo de la capacitación se tomo en cuenta la disponibilidad de tiempo de los asistentes, por lo que se solicito permiso para la asistencia del personal administrativo.

Para la capacitación se hizo uso del laboratorio de informática del colegio Gral. Manuel Belgrano, un datadiplay, y diapositivas.

II.2.3.3 Comprobación

Al final del taller de capacitación se comprobó que hubo una buena captación de la capacitación, porque todos los participantes podían manejar el sistema y tenían mucho interés en aplicarlo en la administración de la Unidad Educativa.

II.2.3.4 Evaluación

Con el ánimo de conocer cuánto comprendieron de la capacitación, se solicito que cada uno de los participantes haga uso del sistema, realice inscripciones, asignación de horarios y cualquier otra herramienta explicada en la capacitación.

II.2.4 Medios de Verificación

Los medios de verificación de este componente son las cartas de aceptación del proyecto por parte del director de la Unidad Educativa, trípticos y diplomas entregados a los participantes de esta capacitación.

II.2.5 Socialización

La socialización organizacional, en palabras de Schein, es la forma de "ponerse al tanto", el proceso de adoctrinamiento y adiestramiento, en el cual se señala lo que es importante en una organización o en alguna parte de la misma.

La socialización es un proceso, que a pesar de su continua presencia, resulta fácil pasarlo por alto. Sin embargo, puede hacer o deshacer una carrera y los planes del personal en una organización. La rapidez y eficacia de la socialización determinan la lealtad, el compromiso, la productividad de los empleados, así como su permanencia o salida. La estabilidad y eficacia de las organizaciones dependerán de la habilidad que tengan éstas para socializar a sus componentes.

La sociabilización del proyecto MAGABEL se realizo con la capacitación donde se dio a conocer el sistema y sus aplicaciones.

III. Conclusiones y Recomendaciones

III.1 Conclusiones

- Durante el desarrollo del sistema se utilizo la metodología de Proceso Unificado, RUP ya que más que un simple proceso, está enfocado a una variedad de sistemas de cualquier tamaño y además permite retroalimentarlo.
- Se diseño una interfaz amigable y sencilla para los usuarios, para asegurar la fácil operación del mismo.
- Se pudo concluir que el desarrollo del proyecto para institución fue satisfactoria por la automatización de la información.
- A pesar de la dificultad de conseguir la información requerida para el desarrollo del proyecto se pudo llegar a satisfacer las expectativas del usuario final.
- Los profesores mostraron bastante interés en la implementación del sistema a pesar de las limitaciones de hardware que existe en su Unidad Educativa, por lo que se comprometieron a solicitar más equipos en la próxima gestión.

III.2 Recomendaciones

- Para optimizar el rendimiento del sistema se recomienda usarlo en un equipo que cumpla con los requisitos del software
- Se recomienda usar las herramientas enfocadas a las TIC, debido a que esto le proporcionara una constante actualización en la forma de llevar a cabo su trabajo de forma eficiente y eficaz.
- Se recomienda que los profesores se actualicen en el manejo de los equipos de computación para un uso del sistema mucho más rápido y eficiente.