CAPITULO I

EL PROYECTO

Proyecto

I.1 Presentación del Proyecto

I.1.1 Título

El Título del presente proyecto es:

"Mejoramiento de la Difusión de la Información, del área de Deportes para el Municipio de Cercado del Departamento de Tarija" (Tarija Deportiva).

I.1.2 Área del Proyecto

El Proyecto abarcará el área Social y Administrativa

I.1.3 Carrera

Ing. Informática

I.1.4 Facultad

Facultad de Ciencias y Tecnología

I.1.5 Institución/Centro Cooperante

Departamento de Deportes.

I.1.6 Provincia/Municipio

Honorable Alcaldía Municipal de la Provincia Cercado.

I.1.7 Duración del proyecto

La duración de la realización del proyecto será de 12 meses de acuerdo a lo establecido.

I.1.8 Responsable del Proyecto

Carrera de Ingeniería Información – Taller III – Grupo 3

I.1.9 Área/línea de investigación priorizada

Tecnología Web e Informática.

I.2 Personal Vinculado al Proyecto

I.2.1 Director del Proyecto

Univ. Carla Vaneza Michel Romero

I.2.2 Equipo de trabajo de: Empresas/Instituciones/Organizaciones participantes/cooperantes

Institución “Departamento de Deportes del Municipio de Cercado de la Ciudad de Tarija”.

Dirección: Av. Víctor Paz, final García Agreda (Palacio de Deportes).

Director del Departamento de Deportes: Dr. Roberto Pablo Aban Lenz.

I.2.3 Actividades previstas para los integrantes del equipo de investigación

Director:

- Planificar y controlar del cronograma del proyecto.
- Asignar y gestionar recursos y prioridades a los distintos componentes y actividades del proyecto.
- Establecer un conjunto de prácticas que aseguren la calidad e integridad del proyecto.
- Supervisar el desarrollo del proyecto.

Para los componentes:

- Elaborar el análisis y diseño del sistema.
- Elaborar la base de datos del sistema.
- Realizar la programación del sistema informático.
- Elaborar las pruebas funcionales del sistema informático.
- Implementar el sistema en la empresa.

I.3 Descripción del Proyecto

I.3.1 Resumen Ejecutivo del Proyecto

El presente proyecto tiene como propósito el Mejoramiento de la Difusión de la Información del área de Deportes para el Municipio de Cercado del Departamento de Tarija, que busca contribuir el mejoramiento donde podrán tener acceso el jefe del Departamento de Deportes, teniendo un informe detallado de cada entrenador, sus horarios de trabajo y lugar donde se realiza el mismo, el sistema web contará con un informe detallado de cada alumno y a que disciplina pertenece, también dará a conocer las diferentes actividades programadas por el Gobierno Municipal a través del Departamento de Deportes como ser: la organización de campeonatos, la generación de roles para partidos y sus tablas de posiciones de determinadas disciplinas, las cuales se difundirán en los diferentes medios de comunicación además de cualquier otra información referente a la actividad física y deportiva.

Para este propósito se quiere proporcionar a la comunidad deportiva y población en general un sistema computarizado orientado a la web, el cual tiene como finalidad Mejorar la difusión de la información, fortalecimiento e integración en la comunicación para el área de Deportes del Municipio de Cercado del Departamento de Tarija.

Así mismo se procederá a la formación de los Directivos del Departamento de Deportes y otros funcionarios dependientes de esta unidad de trabajo, en la utilización de recursos TIC’s (Tecnologías de Información y Comunicación).

Las TIC’s pueden tener diversos papeles en el seno de un organización. Más aún, desempeñan diversas funciones al mismo tiempo. Algunas de ellas son necesarias e imprescindibles, pero no necesariamente estratégicas; otras son clave y fundamento del funcionamiento mismo de la organización moderna.

Infraestructura necesaria para el control de gestión. Ésta es una función fundamental en una organización. La definición de un sistema de información de estas características es una responsabilidad clave de la dirección de la organización.

La ventaja que proporciona es la capacidad de dirección táctica y estratégica de los altos responsables de una empresa. Las TIC’s desempeñan un papel clave para su fortalecimiento, sin un eficaz sistema de información de gestión, es imposible objetivar y cuantificar los problemas o alternativas a tiempo.

Para poder lograr el propósito del proyecto es necesario cumplir los siguientes objetivos:

Sistema de gestión orientado a la Web

El sistema de gestión orientado a la web contribuirá en el mejoramiento de la información y control del Departamento de Deportes de la ciudad de Tarija, beneficiará de forma directa y permitirá coadyuvar a mejorar la administración de la información deportiva y que los usuarios puedan informarse vía web dando más confiabilidad, lo cual significara el éxito del proyecto basado en nuestro propósito. Así también, la comunidad deportiva tendrá una mejor atención y servicio ya que se les brindará reportes de manera eficaz e información oportuna.

Programa de Formación en el uso de las TIC’s.

Se realizará la formación del personal involucrado en el manejo adecuado de las TIC’s, para que de esta manera se pueda hacer un uso correcto y eficaz de los recursos TIC’s.

I.3.2 Descripción y Fundamentación del Proyecto (qué y por qué)

La tecnología de Información 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 datos éstos son preparados e introducidos a la base de datos por funcionarios del Departamento de Deportes mediante una aplicación de computadora.

Se va automatizar toda la información por lo que hasta ahora se viene haciendo manualmente con el miedo a que se pueda perder información o también pueda ocurrir mala manipulación de la información por el cual se pueda manejar toda la información de forma correcta, que será asimilada por los usuarios donde se pueda verificar todos los eventos realizados y también por los ciudadanos que podrán acceder a alguna información pública dentro del Sistema Web.

El Departamento de Deportes del Municipio de Cercado cuenta con una organización que carece de un Sistema Web, motivo por el cual le dificulta tener una información precisa para el desarrollo y el funcionamiento de las actividades programadas ya sea en el funcionamiento de la escuela de deportes, la organización de campeonatos de determinadas disciplinas, convocatorias, noticias y tablas de posiciones.

Son esos los motivos para desarrollar un Sistema Web que permitirá mejorar la difusión de la información, fortalecimiento e integración en la comunicación para el Área de Deportes del Municipio de Cercado de la Ciudad de Tarija, automatizar la información del Departamento de Deportes que ayudará a mejorar las organizaciones en los diferentes campeonatos, horarios, roles de partidos y tabla de posiciones, conocer mayor la información del personal administrativo.

Para poder lograr el propósito del proyecto se plantean las siguientes estrategias:

- Desarrollo de un Sistema Web

  Con este sistema se realizará la gestión de información sobre : los registros de cada uno de los entrenadores de la escuela de deportes, teniendo un informe detallado de cada entrenador, sus horarios de trabajo y lugar donde se realiza el mismo, el sistema web contará con un registro y un informe detallado de cada alumno y a que disciplina pertenece, también dará a conocer las diferentes actividades programadas por el Gobierno Municipal a través del Departamento de Deportes como ser: la organización de campeonatos, la generación de roles para partidos, sus tablas de posiciones de los mismos que se difundirán en los diferentes medios de comunicación y otra información referente a la actividad física y deportiva.

- Capacitación para todos los usuarios del Sistema

  Se aplicará este componente porque es necesario tener recursos humanos formados para el manejo apropiado de las TIC's, una vez concluido el entrenamiento de los recursos humanos se podrá dar el uso adecuado al sistema para el beneficio de la organización.

  Se enfoca en el objetivo, lograr que el personal maneje el sistema implementado sin dificultades, la disposición del personal y los principios pedagógicos de aprendizaje, los que se toman en cuenta para esta capacitación son los de participación, repetición y retroalimentación.

  El Departamento de Deportes cuenta con aproximadamente 100 funcionarios entre encargados, auxiliares, administrador de campos deportivos, coordinadores, profesores barriales, secretarias, etc. Los cuales serán capacitados en el manejo de Internet y sobre las Tecnologías de Información y Comunicación lo cual se llevará a cabo en las instalaciones acordadas con el director del Departamento de Deportes.

I.3.3 Alcances

El sistema web contribuirá al mejoramiento de la organización en el área de Deportes, donde podrán tener acceso el jefe del Departamento de Deportes y otros funcionarios dependientes de esta unidad de trabajo del cual habrá una seguridad en el sistema donde podrán ver qué actividades realizan los entrenadores de la escuela de deportes, teniendo un informe detallado de cada entrenador, sus horarios de trabajo y lugar donde se realiza el mismo.

El sistema contará con un informe detallado de cada jugador y a que disciplina pertenece, también dará a conocer las diferentes actividades programadas por el Gobierno Municipal a través del Departamento de Deportes como ser: la organización de campeonatos, la generación de roles para partidos, tabla de posiciones de los mismos que se difundirá en los diferentes medios de comunicación y otra información referente a la actividad física y deportiva.

I.3.4 Limitaciones

El sistema computarizado orientado a la web sólo contempla la gestión de información el cual no está orientado a la parte contable, como también a inventarios.

El sistema no contempla la elaboración del planillaje de los partidos.

El sistema sólo hace la generación de roles de partidos como también su tabla de posiciones en determinadas disciplinas.

El sistema sólo podrá ser manipulado por los funcionarios dependientes del Departamento de Deportes como también de la alcaldía de la ciudad de Tarija y otros, los cuales solo tienen acceso a la página principal.

El sistema no cubrirá la parte de almacenes de entradas y salidas de material indumentaria de la institución.

I.3.5 Objetivos

Objetivo General

Mejorar la difusión de la información, del área de Deportes para el Municipio de Cercado del Departamento de Tarija.

Objetivo Específicos

Los objetivos específicos del proyecto son:

- Desarrollar un Sistema Web para el apoyo a la gestión de la información para el área de Deportes del departamento de Tarija.
- Capacitación al personal administrativo del Departamento de Deportes de Tarija en el uso del Sistema Web.

I.3.6 Estrategias y planes de acción

1. **Sistema de gestión orientado a la Web "Tarija Deportiva"**

1.1 Recopilación de la información para el desarrollo del proyecto propuesto.

1.2 Análisis y diseño de la página web.

1.3 Diseño de la Base de Datos.

1.4 Programación de la página web y el software del sistema.

1.5 Prueba y modificaciones del sistema y la página web.

1.6 Diseño y documentación del manual de usuario.

2. **Programa de Formación en el uso de las TIC´s y el sistema "Tarija Deportiva".**

2.1 Definición de las estrategias para el curso de formación en recursos TIC's.

2.2 Desarrollar el contenido del curso de Formación.

2.3 Coordinación del director y el personal involucrado del Departamento de Deportes.

2.4 Preparar el material del curso de formación.

2.5 Formación del personal en el uso del sistema y la página web "Tarija Deportiva".

I.3.7 Plan del Desarrollo del Personal

Uno de los componentes del Proyecto a realizarse es el Programa de Formación en el uso de las TIC's y del sistema "Tarija Deportiva".

Para este componente de mucha importancia para el proyecto y para el personal, tratamos de ver la manera más adecuada para que el personal pueda aprender de manera más simple y didáctica el manejo de las TIC's y del sistema "Tarija Deportiva". Los materiales a utilizar para realizar dicha formación son:

- Data Display
- Diapositivas
- Guías de enseñanza

Al final de la capacitación se entregará Certificados de Asistencia firmados por el Departamento de Informática el cual certifica con su conformidad que se desarrolló la formación.

Plan de Asistencia

Para poder desarrollar el proyecto de una manera satisfactoria vemos que sería muy conveniente de poder contar con la ayuda de un analista y diseñador de sistemas el que se encargue del análisis de requerimientos que deberá cumplir el software a desarrollar, así mismo con un equipo de programadores que en base a la carpeta de análisis y diseño que se les vaya a entregar y así empezar a construir a desarrollar el software que una vez terminado deberá someterse a las diferentes pruebas de calidad del software.

Vinculación del Plan Estratégico y el Proyecto

Este proyecto está siendo elaborado con el propósito de mejorar la difusión de la información del área de deportes para Municipio de Cercado del Departamento de Tarija.

Las estrategias que se originan con el proyecto ayudará a cumplir el propósito principal de proyecto, de este modo coadyuvar y alcanzar el fin principal del proyecto.

Todos los pasos definidos en el transcurso de la preparación del proyecto alcanzarán un grado de aceptación óptimo para la ejecución del mismo.

Todos los componentes generados durante la ejecución del proyecto ayudarán a lograr el propósito fundamental del proyecto.

Sistema de Gestión Orientado a la Web

Se utilizará este componente porque teniendo un sistema automatizado de información, beneficiará en el trabajo de las asociaciones involucradas así como también se beneficiará la comunidad deportiva porque tendrá una mejor atención y un mejor servicio contando con información oportuna.

Programa de Formación en el uso de las TIC's y del sistema "Tarija Deportiva".

Se utilizará este componente porque se necesita el personal formado para el manejo adecuado de las TIC's y del sistema, así una vez formado el personal se podrá dar el adecuado uso al sistema para su beneficio.

I.3.8 Metodología

Proceso Unificado de Rational RUP

Para el desarrollo del presente proyecto se utilizará la metodología RUP (Proceso Unificado de Rational) y UML(Lenguaje Unificado de Modelado), que es un lenguaje gráfico que utiliza diagramas ya definidos para especificar o describir métodos o procesos y definir un sistema.

En RUP se siguen cuatro fases para el desarrollo del software, al final de las cuales, y tras una serie de iteraciones se establece objetivos precisos a conseguir.

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 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 lo solicitado está presente.

(Lenguaje Unificado de Modelado) UML.

El Lenguaje Unificado de Modelado prescribe un conjunto de notaciones y diagramas estándar, para modelar sistemas orientados a objetos, describe la semántica esencial de lo que estos diagramas y símbolos significan. Mientras que ha habido muchas notaciones y métodos usados para el diseño orientado a objetos, ahora los modeladores sólo tienen que aprender una notación.

UML se puede usar para modelar distintos tipos de sistemas: Sistemas de Software, sistemas de Hardware, y organizaciones del mundo real. UML ofrece diagramas de los cuales del sistema modelaremos:

- Diagrama de Casos de Uso, para modelar los procesos ‘Business’.
- Diagrama de Secuencia, para modelar el paso de mensajes entre objetos.
- Diagrama de Actividad para modelar el comportamiento de los Casos de Uso, objetos u operaciones.
- Diagrama de Clases, para modelar la estructura estática de las clases en el sistema.

Rational Rose

Rational Rose es una herramienta para "**modelado visual**", que forma parte de un conjunto más amplio de herramientas que juntas cubren todo el ciclo de vida del desarrollo de software.

Rational Rose permite completar una gran parte de las **disciplinas** (flujos fundamentales) del proceso unificado de Rational (RUP), en concreto:

- Modelado del negocio
- Captura de requisitos (parcial)
- Análisis y diseño (completo)
- Implementación (como ayuda)
- Control de cambios y gestión de configuración (parte)

Las vistas de Rational Rose son las siguientes:

a) **La Vista de Casos de Uso, Use Case View,** que es la vista en la que se presenta el comportamiento deseado del sistema: en ella se encontrarían los modelos relacionados con la captura de requisitos. Según el proceso que hemos visto en clase, en esta vista se ubicarían el modelo del negocio, el modelo conceptual y el modelo de casos de uso del sistema.
b) **La Vista Lógica, Logical View,** en la que encontraríamos los modelos que muestran el vocabulario y la funcionalidad (estructura y comportamiento) del sistema, a través de un conjunto de colaboraciones que realizan los casos de uso de la vista de casos de uso (colaboraciones que se modelan mediante diagramas de clases y diagramas de interacción: secuencia y colaboración).
c) **La Vista de Componentes, Component View,** en la que se representa la implementación del sistema mediante componentes, la organización modular del software. Esta vista está relacionada con la gestión de la

configuración del software. Los paquetes en esta vista se organizan en niveles.

Un componente está relacionado con un archivo de software y un lenguaje de programación. Las clases de la vista lógica se asignarían a los componentes de la vista de componentes.

d) **La Vista de Despliegue, Deployment View,** en la que se modela la distribución o despliegue de los componentes a los nodos de procesamiento del sistema. Muestra la topología, distribución e instalación del sistema. Visto las herramientas de que utilizaremos para el análisis y diseño para el desarrollo del proyecto continuaremos con el lenguaje y herramientas de programación que utilizaremos:

El lenguaje de programación a utilizarse para el desarrollo del componente será Java se dará una breve descripción del mismo.

Java

Java es un lenguaje originalmente desarrollado por un grupo de ingenieros de Sun, utilizado por Netscape posteriormente como base para Javascript. Si bien su uso se destaca en el Web, sirve para crear todo tipo de aplicaciones (locales, intranet o internet).

Java es un lenguaje de objetos, independiente de la plataforma.

Algunas características notables:

- Robusto.
- Gestiona la memoria automáticamente.
- No permite el uso de técnicas de programación inadecuadas.
- Multithreading.
- Cliente-Servidor.
- Mecanismos de seguridad incorporados.
- Herramientas de documentación incorporadas.

Java posee ciertas características que hoy día se consideran estándares en los lenguajes OO:

- Objetos.
- Clases.
- Métodos.
- Subclases.
- Herencia simple.
- Enlace dinámico.
- Encapsulamiento.

Java es un lenguaje que ha sido diseñado para producir software:

- Confiable: Minimiza los errores que se escapan a la fase de prueba.
- Multiplataforma: Los mismos binarios funcionan correctamente en Windows/NT/XP/VISTA/SEVEN, Linux, Unix/Motif y Power/Mac.
- Seguro: Applets recuperados por medio de la red no pueden causar daño a los usuarios.
- Orientado a objetos: Beneficioso tanto para el proveedor de bibliotecas de clases como para el programador de aplicaciones.
- Robusto: Los errores se detectan en el momento de producirse, lo que facilita la depuración.

I.3.9 Descripción, Fundamentación y Justificación del Proyecto

I.3.10 Resultados Esperados

Desarrollo de un Sistema Web

Con el Sistema Informático, el Departamento de Deportes mejorará en la calidad de organización de eventos, además de brindar información sobre datos personales de los alumnos como también de los funcionarios dependientes de esta unidad, de manera rápida y confiable agilizando todos los procesos manuales. Con este sistema se realizará la gestión de información sobre: los

entrenadores de la escuela de deportes, teniendo un informe detallado de cada entrenador, sus horarios de trabajo y lugar donde se realiza el mismo.

El sistema web contará con un informe detallado de cada jugador y a qué disciplina pertenece, también dará a conocer las diferentes actividades programadas por el Gobierno Municipal a través del Departamento de Deportes como ser: la organización de campeonatos, la generación de roles para partidos, resultados de los mismos que se difundirá en los diferentes medios de comunicación y otra información referente a la actividad física y deportiva.

I.3.11 Transferencia de Resultados

I.3.11.1 Medios y estrategias para la transferencia de resultados.

En el presente proyecto se establecen 2 etapas que se realizarán de manera secuencial y mediante un convenio con la Institución del Departamento de Deportes, las mismas que se detallan a continuación:

- Desarrollar un Sistema Informático para el control en la administración interna de las diferentes disciplinas involucradas y contar con una fuente de información para la comunidad del Departamento de Deportes y público en general de una manera más eficiente.
- Capacitar al personal de la Institución para el manejo del Sistema Informático: La capacitación al plantel administrativo, mediante ejemplos y demostración en instalación.

I.3.11.2 Grupo de beneficiarios de los resultados

Los beneficiarios directos están sujetos al convenio que se realice:

El sistema beneficiará al Departamento de Deportes para su mejor organización de la administración personal y diferentes eventos deportivos que existen en dicha institución.

I.3.12 Cronograma de Actividades

Nº	Actividad	Nº días	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12
ÁREA FORMATIVA														
1	Sistema del marco lógico	12		X										
2	Modelamiento del Negocio con RUP	5			X									
3	Tendencias Pedagógicas	11				X								
4	Oratoria	18					X							
5	Metodologías de la investigación	12						X						
6	Relación entre la Universidad y la Industria	12							X					
ÁREA DE DESARROLLO DE TRABAJO														
1	Presentación del Perfil de Proyecto				X									
2	Entrega de observaciones					X								

3	**Entrega de Perfiles modificados**					**X**								
4	**Primera revisión**						**X**							
5	**Segunda revisión**							**X**						
6	**Tercera revisión**								**X**					
7	**Cuarta revisión**											**X**		
8	**Entrega de Borradores**											**X**		
9	**Observaciones por parte del tribunal**												**X**	
10	**Entrega de borradores finales**													**X**
11	**Elaboración de rol de defensas**													**X**
12	**Defensas finales**	**14**												**X**
AREA DE INVESTIGACIÓN														
1	**Resumen ejecutivo del proyecto de**					**X**								

	acuerdo a Formato establecido													
2	**Artículo sobre Informática Jurídica**								**X**					
3	**Artículo sobre Estándares de e-Learning**											**X**		
4	**Artículo sobre su proyecto desarrollado**													**X**

Tabla 1 **Cronograma de actividades**

I.3.13 Marco Lógico del Proyecto

I.3.13.1 Cuadro de Involucrados del Proyecto

GRUPO	INTERESES	PROBLEMAS	RECURSOS Y MANDATOS
DEPARTAMENTO DE DEPORTES	Tener un sistema para el mejoramiento en el control de la administración en las diferentes disciplinas del Departamento de Deportes del Municipio de Tarija.	Ineficiencia en la manipulación de información de la administración dentro de las disciplinas del Municipio de Tarija.	La disponibilidad de colaboración para la ejecución del proyecto.
PERSONAL	Evitar la morosidad en la búsqueda de toda la parte administrativa como ser: Entrenadores, Alumnos y Campeonatos y así tener una mejor manipulación de la información. .	Demora en la búsqueda del registro del alumno por motivo que existe numerosos alumnos en todas las disciplinas que existen dentro del Departamento de Deportes.	La disponibilidad de información para apoyar el desarrollo del sistema. Realizar seguimiento del proyecto con las autoridades del Departamento de Deportes.

ALUMNOS	Tener un sistema confiable para resguardo de sus datos personales.	Mala utilización de los documentos. Demora en el momento de realizar una búsqueda para mayor información de alumnos.	

Tabla 2 Análisis de involucrado

I.3.13.2 Árbol de Problemas

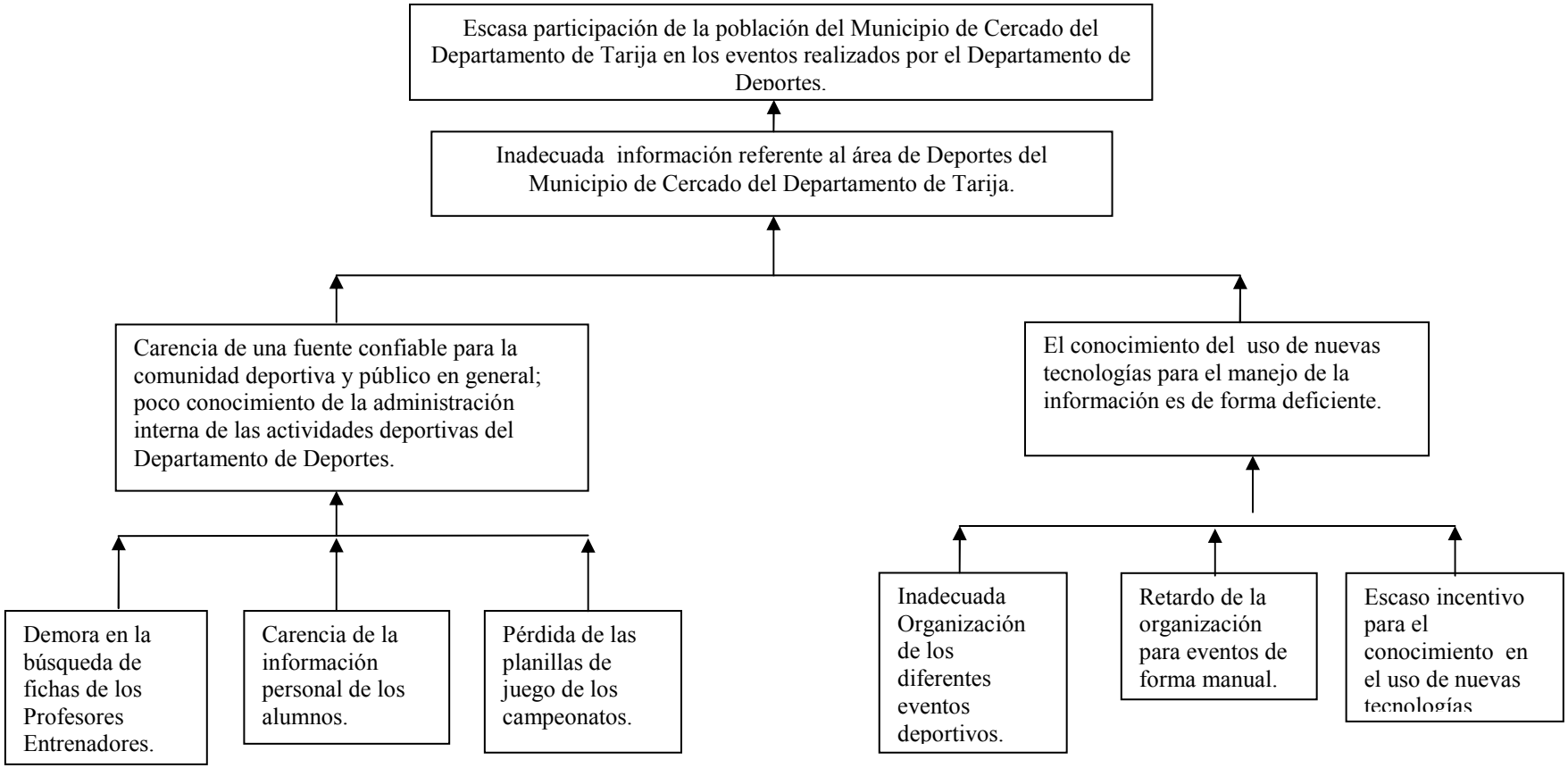


Figura 1 **Árbol de problemas**

*** Ver Resultados de la Encuesta del Anexo del Componente I**

I.3.13.3 Árbol de Objetivo

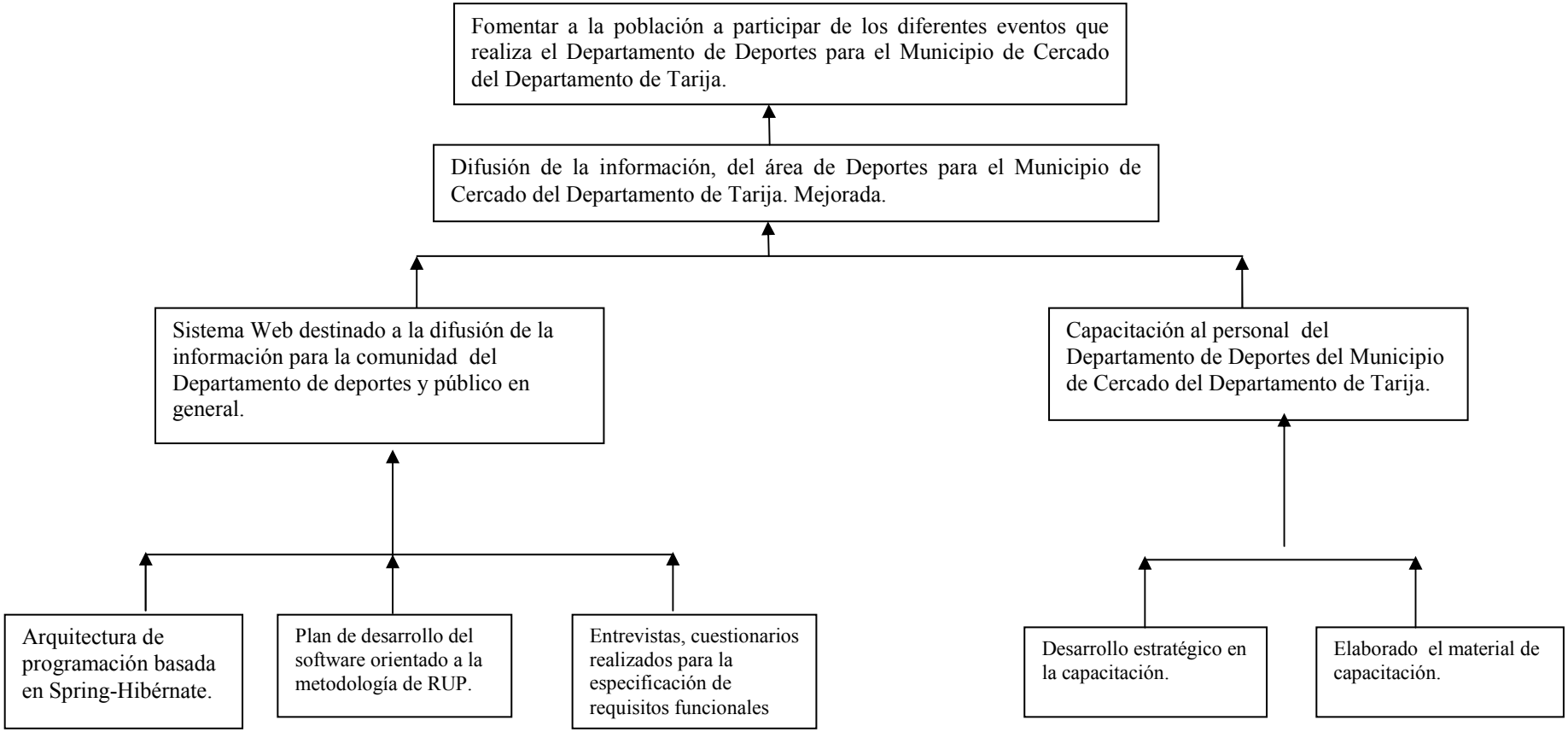


Figura 2 **Árbol de objetivos**

I.3.13.4 Marco lógico del Proyecto

Resumen Narrativo del Proyecto	**Indicadores**	**Medios de Verificación**	**Supuestos**
Fin Fomentar a la población a participar de los diferentes eventos que realiza el Departamento de Deportes para el Municipio de Cercado del Departamento de Tarija.	A partir del año 2013 la información se procesa con rapidez y seguridad en un 80 % en el Departamento de Deportes.(EFICACIA) Nº registro de disciplinas – Nº de gestión de disciplinas *100 Nº registro de disciplinas	Informe de evaluación proporcionado y avalado por el Departamento de Deportes expresando conformidad con los resultados obtenidos.	Buena predisposición del personal del Departamento de Deportes para coadyuvar con el desarrollo del proyecto.

Objetivo General (Propósito) La difusión de la información, del área de Deportes para el Municipio de Cercado del Departamento de Tarija. Mejorado.	Se mejora el desempeño de las actividades a través del control y seguimiento del 50% en las actividades relativas a la gestión anterior de la institución, en base al año 2011. (EFICACIA). Nº de actividades desarrolladas - Nº de actividades que se realizaba / Nº de actividades que se realizaba *100	Informe de conformidad de funcionamiento emitido por el Director del Departamento de Deportes.	Los miembros del Departamento de Deportes apoyan al proyecto que va beneficio de los mismos como también al público en general.

Objetivos Específicos (Componentes) 1.- Sistema Web destinado a la difusión de la información para la comunidad del Departamento de deportes y público en general. 2.- Capacitación al personal del Departamento de Deportes del Municipio de Cercado del	Sistema computarizado orientado a la web desarrollado y disponible para su explotación, mediante la supervisión del Departamento de Deporte al finalizar el proyecto 50% de acuerdo a la norma IEEE830. Al finalizar el proyecto se ha capacitado al personal en el uso de las TIC's y del sistema orientado a la web "Tarija Deportiva" en un 60%.	Carta avalada por el Director del Departamento de Deportes certificando que la página web responde a los requerimientos acordados. Manual de usuario de la página web. Entrega de certificados de la capacitación al finalizar el curso al Departamento de Deportes. Lista de Personal Asistido formados en recursos TIC's. Material didáctico para los	Interés del Director hacia la página web. Disponibilidad de herramienta de software, recursos y hardware para su desarrollo. Buena predisposición del personal que conforma

Departamento de Tarija.		talleres. Fotos.	la institución participa activamente en las capacitaciones. Disponibilidad de ambientes para el curso de formación.
Actividades: **Componente 1:** 1. Recopilación de la información para el desarrollo del proyecto propuesto. 2. Análisis y diseño	**Presupuesto** Recursos Humanos 1000 Equipos Computarizados 1680 Materiales	Informe de ejecución humanística del proyecto avalado por el director del Departamento de Deportes del Municipio de Cercado del Departamento de Tarija.	Disponibilidad de equipos de computación. Existencia y disponibilidad de recursos económicos.

de la página web. 3. Diseño de la Base de Datos. 4. Prueba y modificaciones del sistema y la página web. 5. Diseño y documentación del manual de usuario. **Componente 2:** 1. Coordinación con el Jefe del Departamento de Deportes para los cursos de	300 Total 2980 Bs		

formación. 2. Desarrollar el contenido del curso de formación. 3. Formación del personal en el uso del sistema y la página web. 4. Entrega de certificado de participación.			

Tabla 3 Matriz de Marco Lógico

I.4 Presupuesto / Justificación

ITEM	RUBROS	Aporte Universidad	Otro Aporte	TOTAL (Bs.)
10000	**SERVICIOS PERSONALES**			
	12000 Empleados no Permanentes		1500	1500
	Sub total rubro		1500	1500
20000	**SERVICIOS NO PERSONALES**			
	21000. Servicios Básicos			
	22000. Servicios de transporte		80	80
	23000. Alquileres			
	24000. Mantenimiento y reparación			
	25000. Servicios Profesionales y Comerciales		700	700

	Sub total rubro		780	780
30000	**MATERIALES Y SUMINISTROS**			
	31000. Alimentos y Productos Forestales		150	150
	32000. Productos de Papel, Cartón e Impresos		500	500
	33000. Textiles y Vestuario.			
	34000. Productos Químicos, Combustibles y Lubricantes			
	39000. Productos Varios.		50	50
	Sub total rubro		700	700
40000	**ACTIVOS REALES**			
	43000. Maquinaria y Equipo.			

	46000. Descripción de estudios y proyectos para inversión			
	49000. Otros Activos			
	Sub total rubro			
	TOTAL		2980	2980
	TOTAL + 40% Incentivo			

Tabla 4 Presupuesto general

1) **GRUPO 10000. SERVICIOS PERSONALES**

a) SUB GRUPO 12000. Empleados no Permanentes

Partida	**Personal**	**Remuneración**	**Tiempo/meses**	**Total**
12100	Personal Eventual	250	6	1500
Total				

Tabla 5 Empleados no permanentes

2) 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			
21200	Energía Eléctrica			
21300	Agua			
21400	Servicios Telefónicos			
Total				

Tabla 6 Descripción de los gastos de servicios básicos

* 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	Ubicación de la empresa	80	1	80
Total					80

Tabla 7 Descripción de los gastos de viajes y transporte de personal

* 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	Tarija			
Total					
Total sub grupo 22000					

Tabla 8 Descripción de los gastos de viajes y transporte de personal

* 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			
23300	Alquiler de Tierras y Terrenos			
Total				

Tabla 9 Descripción de los gastos por concepto de alquileres de equipos y maquinarias

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				

Tabla 10 Descripción mantenimiento y reparación

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**
25200	Estudios e Investigaciones				
25500	Publicidad				
25600	Imprenta				
25700	Capacitación de Personal	2	350	1	700
25800	Estudios e Investigaciones Para Proyectos de Inversión				
Total					700

Tabla 11 Descripción de los gastos en servicios profesionales y comerciales

* 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.

3) 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 • Refrigerio(empanadas • Refrigerio(refresco))	• 50 • 5	• 2 • 10	• 100 • 50
31200	Alimento para Animales			
31300	Productos Agroforestales y Pecuarios			
Total				150

Tabla 12 Descripción de los gastos Alimentos y Productos Agroforestales

* 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 Resmas de papel tamaño carta	5	70	350
32200	Productos de Artes Gráficas, Papel y Cartón			
32300	Libros y Revistas			
32400	Textos de Enseñanza	10	15	150
32500	Periódicos			
Total				500

Tabla 13 Descripción del gasto de Productos de Papel, Cartón e Impresos

* 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				

Tabla 14 Descripción del gasto en textiles y vestuario

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				

Tabla 15 Combustibles, Productos Químicos, Farmacéuticos y Otros

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	25	2	50
39700	Útiles y Materiales Eléctricos			
39800	Otros Repuestos y Accesorios			
Total				50

Tabla 16 Descripción del gasto en productos varios

4) 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			
43200	Maquinaria y Equipo de Producción			
43300	Equipos de Transporte, Tracción y Elevación			
43400	Equipo Médico y de Laboratorio			
43600	Equipo Educacional y Recreativo			
43700	Otra Maquinaria y Equipo			
Total				

Tabla 17 Descripción del gasto de Maquinaria y Equipo

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				

Tabla 18 Descripción de estudios y proyectos para inversión

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

Partida	Tipos de productos *	Cantidad	Costo/Unitario	Total
49100	Activos Intangibles			
49200	Compra de Bienes Muebles Existentes (Usados)			
49300	Semovientes y otros Animales			
49900	Otros Activos			
Total				

Tabla 19 Descripción del gasto de Otros Activos

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

CAPITULO II

COMPONENTES

COMPONENTE I

SISTEMA COMPUTARIZADO ORIENTADO A LA WEB PARA EL MEJORAMIENTO DE LA ADMINISTRACION DEL DEPARTAMENTO DE DEPORTES, DESARROLLADO.

II Componentes

- **Componente 1: Desarrollar un Sistema Web para el apoyo a la gestión de la información para el área de Deportes del departamento de Tarija.**

II.1.1 Plan de Desarrollo del Software

II.1.1.1 Introducción

Este Plan de Desarrollo del Software es una versión corregida 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 una visión global del enfoque de desarrollo propuesto.

El proyecto será desarrollado por la universitaria Carla Vaneza Michel Romero basado en una metodología de Rational Unified Process (RUP) en la que se procederá a cumplir con las cuatro fases que marca la metodología. 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.1.1.2 Propósito

El propósito del Plan de Desarrollo de Software es dar a conocer 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 director 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 que deben hacer, cuándo deben hacerlo y qué otras actividades dependen de ello.

II.1.1.3 Alcance

Con el Plan de Desarrollo del software se pretende analizar y elaborar un proyecto de gran magnitud abarcando todas las fases requeridas en la terminología RUP.

Para la primera versión la base está en la captura de requisitos realizada con la información proporcionada. Posteriormente, el avance del proyecto y la realización de cada una de las iteraciones ocasionarán el ajuste de este documento produciendo nuevas actualizadas.

II.1.1.3.1 Suposiciones y Restricciones

Acontecimientos que deben ocurrir para que el proyecto sea ejecutado con el éxito pero que están totalmente fuera del ámbito del control del equipo de proyecto.

II.1.1.3.1.1 Suposiciones

- La sociedad requiera conocer las últimas noticias sobre el deporte.
- Presupuesto suficiente para la implementación del Sistema de gestión orientado a las web.
- Los deportistas tienen interés por poder tener acceso a la información actualizada que brinda el Sistema de gestión orientado a la Web.
- Disponibilidad de una organización encargada de la administración del Sistema de gestión orientado a la Web.
- Interés del Departamento de Deportes para utilizar el sistema de gestión a la web “Tarija Deportiva”.
- Predisposición de los miembros de la asociación y público en general.

- La formación del personal encargado del manejo del sistema se lleve en la fecha determinada.
- Disponibilidad de equipos de computación.
- Se asume que los requisitos descritos en este documento son estables una vez que sea aprobado por las diferentes autoridades del departamento de Deportes de la Alcaldía Municipal de la Ciudad de Tarija.

II.1.1.3.1.2 Restricciones

Limitaciones generalmente fuera del ámbito de control del equipo de proyecto que pueden afectar negativamente a su alcance.

- No contar con un servidor Web que presente las características necesarias para la ejecución del Sistema.
- No contar con los fondos suficientes para llevar a cabo la Capacitación a los administradores del sistema.
- Tiempo limitado para la ejecución del proyecto.
- El sistema no contará con herramientas que brindan reportes de transacciones económicas o financieras dentro del Departamento de Deportes para el Municipio de Cercado.
- El sistema no funciona correctamente en el navegador Internet Explorer porque no tiene buen soporte para el nuevo estándar html5 css3.

II.1.1.4 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 RUP desde la perspectiva de artefactos, y que proponemos para este proyecto.

Es previsto destacar que de acuerdo a la metodología de Rational Unified Process 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.

- **Plan de Desarrollo del Software**

Es el presente documento.

- **Modelo de Casos de Uso del Negocio**

Es un modelo de las funciones vistas desde las perspectivas de los actores externos (agentes de proyecto, 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.

- **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 de ellos. Para la representación de este modelo se utiliza Diagrama 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 los Diagramas de Actividad para mostrar los flujos de trabajo.

- **Glosario**

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

- **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 Diagrama de Casos de Uso.

- **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.

- **Especificaciones de Casos de Uso**

Para los casos de uso que lo requieran (cuya funcionalidad no sea evidente o que no baste con la simple especificación 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.

- **Especificaciones Adicionales**

Este documento captura todos los requisitos que no han sido incluidos como parte de los casos de uso y se refieren a 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 requerimientos de ambiente, tales como: sistema operativo, requisitos de compatibilidad, etc.

- **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 grafica 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.

- **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.

- **Modelo de Implementación**

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

- **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 perfil UML para Modelado de Datos, para conseguir la representación de tablas, claves, etc.).

- **Casos de Prueba**

Cada caso de 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 prueba con las instrucciones para realizar la prueba y dependiendo de tipo de prueba dicho procedimiento podrá ser automatizado mediante un script de prueba.

- **Manual de Instalación**

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

- **Material de Apoyo al Usuario Final**

Corresponde a un conjunto de documentos y facilidades de uso del sistema, como guías del usuario.

- **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 en desarrollo incremental e iterativamente, obteniéndose una nueva entrega al final de cada iteración.

II.1.2 Marco Metodológico

Con el presente proyecto se pretende aplicar la tecnología de Información para optimizar todos los procesos de la institución, es decir, el registro, almacenamiento de información, modificación y recuperación de todo tipo de información generada las actividades que se realiza, mediante la implementación de un sistema informático.

Este Sistema Informático aplicado a los procesos de gestión de mejoramiento de la administración del Departamento de Deportes que ayudará a aumentar la eficacia y eficiencia, mejorando las capacidades de automatización y seguimiento de las actividades para optimizar las actividades del Departamento de Deportes.

II.1.2.1 Rational Unifiel Process (RUP) *

La metodología RUP, denominada así por sus siglas en ingles Rational Unified Process, divide en 4 fases el desarrollo del software:

- Inicio, el objetivo en esta fase es determinar la visión del proyecto. Se desarrolla una descripción del producto final a partir de una idea.
- Elaboración, el objetivo es determinar la arquitectura óptima. Se especifica en detalle la mayoría de los casos de uso del producto.
- Construcción, en esta fase la finalidad es obtener la capacidad operacional inicial, la línea base de la arquitectura crece hasta convertirse en el sistema completo.
- Transición, el objetivo es obtener el realce del proyecto. Un número reducido de usuarios con experiencia prueba el producto en forma de defectos y deficiencias.

  Rational Unified Process es un proceso de negocios genéricos para la ingeniería de software orientada a objetos.

  Proporciona un acercamiento disciplinado a la asignación de tareas y responsabilidades.

  El ciclo de vida que se desarrolla por cada iteración, es llevada bajo dos: disciplinas:

Dentro de la disciplina de desarrollo se tiene:

- Ingeniería de Negocios: Entendiendo las necesidades del negocio.
- Requerimientos: Transferencia a las necesidades del negocio a un sistema automatizado.
- Análisis y Diseño: Requerimientos dentro de la arquitectura de software.
- Implementación: Creando software que se ajuste a la arquitectura y tenga el comportamiento deseado.
- Pruebas: Asegurándose que el comportamiento requerido es el correcto y que todo lo solicitado está presente.

En la Disciplina de Soporte las características son:

- Configuración y administración del cambio: Guardando todas las versiones del proyecto.
- Administrando el proyecto: Horarios y recursos.
- Distribución: Hacer todo lo necesario para la salida del proyecto.

  Es recomendable que a cada de estas iteraciones se clasifique y ordene según su prioridad, para posteriormente convertirse en un entregable al cliente. Esto trae como beneficio la retroalimentación que se tendría en cada entregable o en cada iteración.
- Actividades, son los procesos que se determinan en cada iteración.
- Trabajadores, se constituyen en las personas involucradas en cada proceso.
- Artefactos, pueden ser un documento, un modelo o un elemento de modelo.

Una particularidad de esta metodología es que, en cada ciclo de iteración, se hace exigente el uso de artefactos, siendo por este motivo, una de las metodologías más importantes para alcanzar un grado de certificación en el desarrollo del software.

---

* http://www.programacionesxtrema.org/articulos/newMetodology.es.html

II.1.2.2 Ingeniería de requerimientos

Concepto

El proceso de recopilar, analizar y verificar las necesidades del cliente o usuario para un sistema es llamado ingeniería de requerimientos. La meta de la ingeniería de requerimientos (IR) es entregar una especificación de requisitos de software correcta y completa.

Algunos otros conceptos de ingeniería de requerimientos son:

"Ingeniería de Requerimientos ayuda a los ingenieros de software a entender mejor el problema en cuya solución trabajarán. Incluye el conjunto de tareas que conducen a comprender cuál será el impacto del software sobre el negocio, qué es lo que el cliente quiere y cómo interactuarán los usuarios finales con el software". (Pressman, 2006: 155)

"La ingeniería de requerimientos es el proceso de desarrollar una especificación de software. Las especificaciones pretender comunicar las necesidades del sistema del cliente a los desarrolladores del sistema". (Sommerville, 2005: 82)

En síntesis, el proceso de ingeniería de requerimientos se utiliza para definir todas las actividades involucradas en el descubrimiento, documentación y mantenimiento de los requerimientos para un producto de software determinado, donde es muy importante tomar en cuenta que el aporte de la IR vendrá a ayudar a determinar la viabilidad de llevar a cabo el software (si es factible llevarlo a cabo o no), pasando posteriormente por un subproceso de obtención y análisis de requerimientos, su especificación formal, para finalizar con el subproceso de validación donde se verifica que los requerimientos realmente definen el sistema que quiere el cliente.

Importancia de la ingeniería de requerimientos

Según la autora Lizka Johany Herrera en su documento de la ingeniería de requerimientos, los principales beneficios que se obtienen de la Ingeniería de Requerimientos son (2003: 3):

Permite gestionar las necesidades del proyecto en forma estructurada: Cada actividad de la IR consiste de una serie de pasos organizados y bien definidos.

Mejora la capacidad de predecir cronogramas de proyectos, así como sus resultados: La IR proporciona un punto de partida para controles subsecuentes y actividades de mantenimiento, tales como estimación de costos, tiempo y recursos necesarios.

Disminuye los costos y retrasos del proyecto: es sabido que reparar errores por un mal desarrollo no descubierto a tiempo, es sumamente caro; especialmente aquellas decisiones tomadas durante la IR, ya que es una de las etapas de mayor importancia en el ciclo de desarrollo de software y de las primeras en llevarse a cabo.

Mejora la calidad del software: La calidad en el software tiene que ver con cumplir un conjunto de requerimientos (funcionalidad, facilidad de uso, confiabilidad, desempeño, etc.).

II.1.2.2.1 Requerimientos funcionales

Expresan la naturaleza del funcionamiento del sistema (cómo interacciona el sistema con su entorno y cuáles van a ser su estado y funcionamiento).

II.1.2.2.2 Requerimientos no funcionales

Son restricciones en el espacio de posibles soluciones, como ser:

- Rendimiento del sistema: fiabilidad, tiempo de respuesta, disponibilidad
- Interfaces: dispositivos de E/S, usabilidad, interoperabilidad
- Proceso de desarrollo: estándares, herramientas, plazo de entrega.

II.1.2.3 (Lenguaje Unificado de Modelado) UML*

El Lenguaje Unificado de Modelado prescribe un conjunto de notaciones y diagramas estándar para modelar sistemas orientados a objetos, describe la semántica esencial de lo que estos diagramas y símbolos significan. Mientras que ha habido muchas notaciones y métodos usados, para el diseño orientado a objetos, ahora los modeladores sólo tienen que aprender una notación.

UML se puede usar para modelar distintos tipos de sistemas: Sistemas de Software, sistemas de Hardware, y organizaciones del mundo real. UML ofrece diagramas de los cuales del sistema modelaremos:

- Diagrama de Casos de Uso, para modelar los procesos 'Business'.
- Diagrama de Secuencia, para modelar el paso de mensajes entre objetos.
- Diagrama de Actividad para modelar el comportamiento de los Casos de Uso, objetos u operaciones.
- Diagrama de Clases, para modelar la estructura estática de las clases en el sistema.

UML es una consolidación de muchas de las notaciones y conceptos más usados orientados a objetos. Empezó como una consolidación el trabajo de Grade Boch, James Rumbaugh, e Ivar Jacobson, creadores de tres de las Metodologías orientadas objetos más populares.

UML es ante todo un lenguaje. Un lenguaje proporciona un vocabulario y unas reglas para permitir una comunicación. En este caso, este lenguaje se centra en la representación gráfica de un sistema.

Se ha convertido en el estándar de facto de la industria, debido a que ha sido impulsado por los autores de los tres métodos más usados de orientación a objetos: Grady Booch, Ivar Jacobson y JimRumbaugh.

En 1996, el Object Management Group (OMG), un pilar estándar para la comunidad del diseño orientado a objetos, publicó una petición con propósito de una meta

modelo orientado a objetos de semántica y notación estándares. UML, en su versión 1.0, fue propuesto como una respuesta a esta petición en enero de 1997.

Hubo otras cinco propuestas rivales. Durante el transcurso de 1997, los seis promotores de las propuestas, unieron su trabajo y presentaron al OMG un documento revisado de UML, llamado UML versión 1.1. Este documento fue aprobado por el OMG en Noviembre de 1997. El OMG llama a este documento OMG UML versión 1.1.

El OMG está actualmente en proceso de mejorar una edición técnica de esta especificación, prevista su finalización para el 1de abril de 1999.

---

* UML:y Patrones – Graing Larman

II.1.3 Programación Orientada a Objetos

Es un paradigma de programación que define los programas en términos de "clases de objetos", los objetos que son entidades que combinan estado (es decir, datos), comportamiento (esto es, procedimientos o métodos) e identidad (propiedad del objeto que lo diferencia del resto). La programación orientada a objeto expresa un programa como un conjunto de estos objetos, que colaboran entre ellos para realizar tareas. Esto permite hacer los programas y módulos más fáciles de escribir, mantener y reutilizar.

De esta forma, un objeto contiene toda la información, (los denominados atributos) que permite definirlo e identificarlo frente a otros objetos pertenecientes a otras clases (e inclusive entre objetos de una misma clase, al poder tener valores bien diferenciados en sus atributos). A su vez, dispone de mecanismos de interacción (los llamados métodos) que favorecen a la comunicación entre objetos (de una misma clase o de distintas), y en consecuencia, el cambio de estado en los propios objetos. Esta característica lleva a tratarlos como unidades indivisibles, en las que no se separan información (datos) y procesamiento (métodos).

De esta propiedad de conjunto de una clase de objetos, que al contar con una serie de atributos definitorios, requiere de unos métodos para poder tratarlos (lo que hace que estos conceptos estén íntimamente entrelazados), el programador debe pensar indistintamente en ambos términos, ya que nunca debe separar o dar mayor importancia a los atributos a favor de métodos, ni viceversa. Hacerlo puede llevar al programador a seguir el hábito erróneo de crear clases contenedoras de información por un lado y clases con métodos que manejen esa información por otro.

Esto defiere de la programación estructurada tradicional, en la que los datos y los procedimientos están separados y sin relación ya que lo que se busca es el procesamiento de unos datos de entrada para obtener otros de salida. La programación estructurada anima al programador a pensar sobre todo en

términos de procedimientos o funciones, y en segundo lugar en las estructuras de datos que esos procedimientos manejan. En la programación estructurada se escriben funciones y después le pasan datos. Los programadores que emplean lenguajes orientados a objetos definen con datos y métodos y después envían mensajes a los objetos diciendo que realicen esos métodos en sí mismos.

II.1.4 PostgreSQL*

Es un sistema de gestión de base de datos relacional orientada a objetos de software libre, publicado bajo la licencia BSD. Como muchos otros proyectos open source, el desarrollo de **PostgreSQL** no es manejado por una sola compañía sino que es dirigido por una comunidad de desarrolladores y organizadores comerciales las cuales trabajan en su desarrollo. Dicha comunidad es denominada el PGDG **(PostgreSQL Global Development Group)**

II.1.4.1 Razones para usar postgreSQL:

Disponibilidad en las plataformas múltiples PostgresSQL, está disponible en cada sistema operativo Unix- compatible moderno, ventanas y los puertos están también disponibles para Novell NetWare y OS/2.

Herramientas profesionales del desarrollo y de la administración.

La lista siguiente resume apenas algunas de las herramientas disponibles para los reveladores de PostgresSQL:

1. **Modelado de la Base de Datos:** Varios productos de la fuente comercial están abiertas en su disposición para el modelado de datos, para lo cual incluya al arquitecto visual del caso y de los datos.
2. **La administración y desarrollo:** Hay numerosos esfuerzos impresionantes que entran de manera importante en esta área, de los cuales tres productos son particularmente prometedores. El pgAdmin III tiene una historia particularmente larga del desarrollo y es capaz de manejar prácticamente cualquier tarea que se extiende de la creación de la

tabla simple a la réplica de manejo a través de los servidores múltiples. Navicat PostgreSQL ofrece las características similares al pgAdmin III y se empaqueta en un interfaz muy bien diseñado. Una herramienta buena, en Internet es phpPgAdmin.

3. **Información** PostgreSQL interconecta con todas las herramientas de corriente de información, incluyendo los informes cristalinos, Cognos ReportNet, y el paquete cada vez más popular JasperReports de la información de la fuente abierta.

4. **Ayuda:** Para los lenguajes de programación múltiples PostgreSQL apoya C++, C#, JDBC, Perl, PHP, phayton, Tcl, Ada, Pascal.

5. **Apoya requisitos de la empresa:** La empresa puede elegir EnterpriseDB, una versión de PostgreSQL, que re implementa características tales como tipos de datos, disparadores, opiniones y cursores que copian el comportamiento de Oracle, en vez de Oracle perceptiblemente más caro.

---

* http://www.postgresql-cl/

II.1.5 Análisis y diseño

II.1.5.1 Requerimientos funcionales

- Se deben registrar los datos de todos los usuarios que utilizan el sistema.
- El sistema asignará un código numérico que identificara únicamente al usuario en el sistema.
- Se deberá especificar el nombre del usuario
- El nombre del usuario debe ser único
- Se debe guardar todos los procesos que realiza el usuario desde el momento que entra al sistema y sale.

II.1.5.2 Requerimientos no funcionales

- Los usuarios tienen restringido sus roles en el sistema de acuerdo a su desempeño laboral.
- Las convocatorias deber ser cargadas con anticipación.
- Los campeonatos realizados deben ser respetados de acuerdo a fecha y horario establecido.
- La carga de resultados del campeonato realizado debe ser introducida con cuidado para que se guarde correctamente en la tabla de posiciones.

Referencias

- Visual Modeling winh Rational Rose and UML, Terry Quatrani. – Addison – Wesley.
- Documentación de Rational Unified Process, manual de ayuda, tutoriales, etc.
- UML y patrones. Craig Larman
- Diseño de aplicaciones Web usando UML, por Miguel Reynolds.
- Especificación de requisitos de Software Proyecto:

  HipoSoft Revisión 1.4

II.1.6 Modelo de Casos de Uso del Negocio

II.1.6.1 Introducción

El Modelo de Casos de Uso del Negocio es un artefacto de la disciplina requisitos en la metodología RUP la cual estamos implementando.

II.1.6.2 Propósito

- Comprender la estructura y la dinámica de la organización.
- Comprender problemas actuales e identificar posibles mejoras.
- Comprender los procesos de negocio de la organización.

II.1.6.3 Alcance

- Describe los procesos de negocio y los clientes.
- Identifica y Definir los procesos de negocio según los objetivos de la organización.
- Define 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 de la organización).

II.1.6.4 Diagrama de Casos de Uso del Negocio

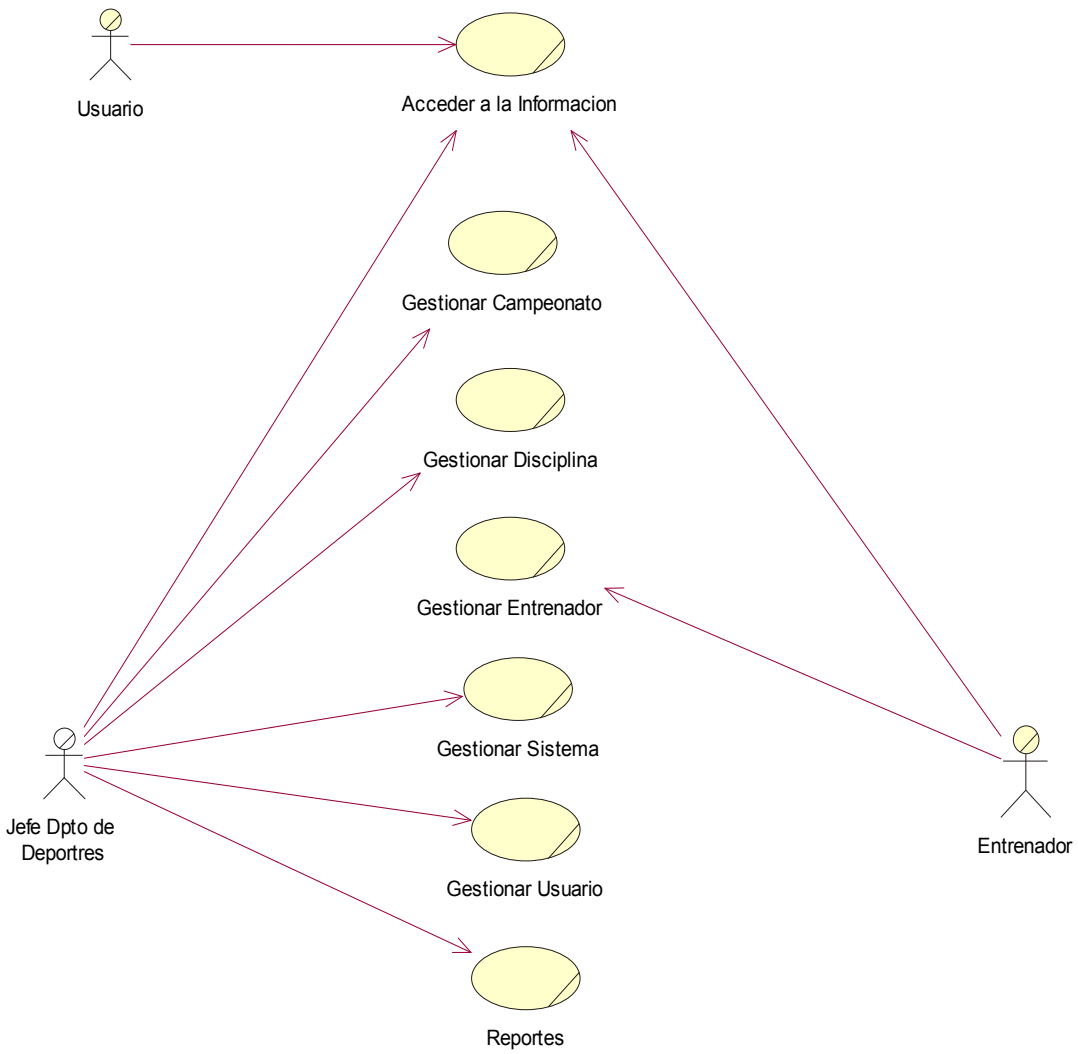


Figura 3 **Diagrama de Casos de Uso del Negocio**

II.1.7 Modelo de Objetos del Negocio

II.1.7.1 Introducción

El modelo de objetos del negocio es un artefacto de la disciplina requisitos en la metodología RUP la cual estamos implementando.

II.1.7.2 Propósito

- Comprender la estructura dinámica de la organización.
- Comprender los procesos de negocio de la organización.

II.1.7.3 Alcance

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

II.1.7.4 Modelo de Objetos del Negocio

Jefe del Departamento de Deportes Encargado

Figura 4 **Modelo de objetos Jefe del Dpto. de Deportes**

El Coordinador encargado

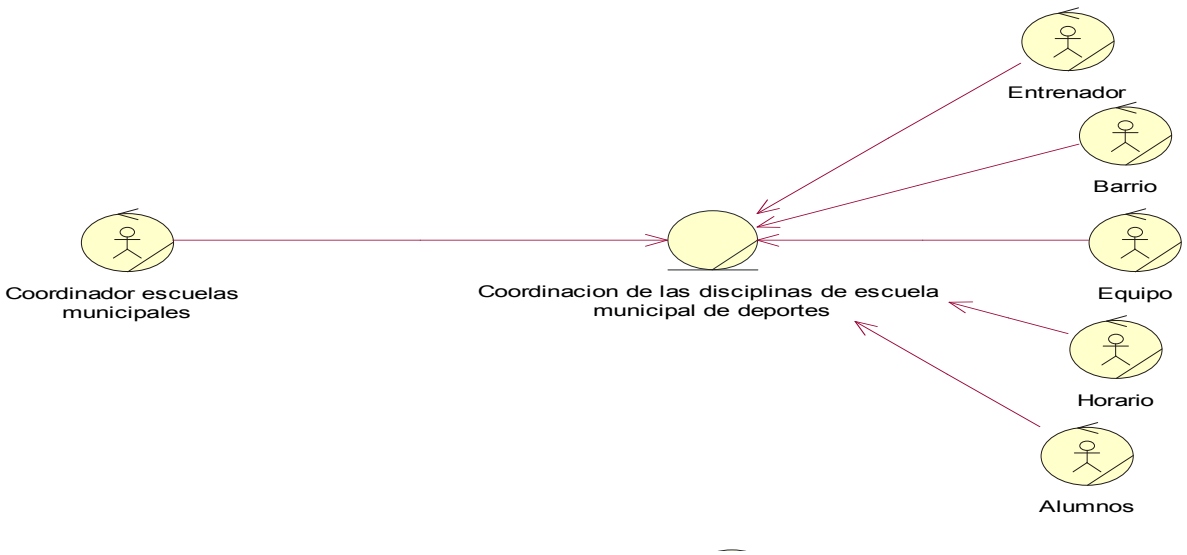


Figura 5 **Modelo de objetos del Coordinador del Dpto. de Deportes**

Coordinador de la Adm. del Dpto. de Deportes

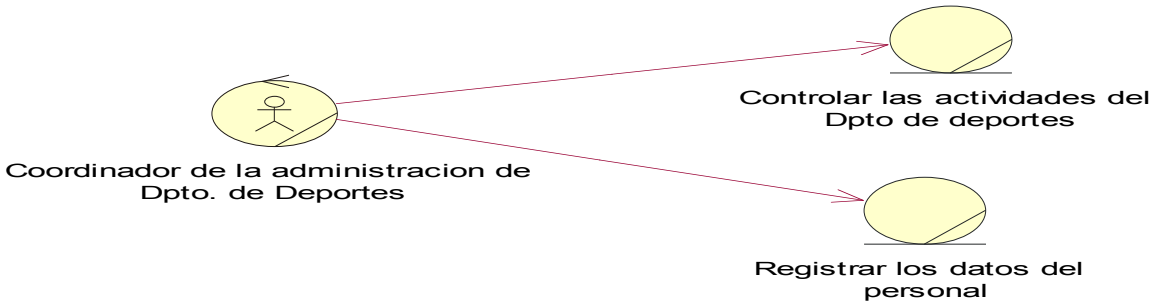


Figura 6 **Modelo del Coordinador Jefe del Dpto. de Deportes**

Supervisor

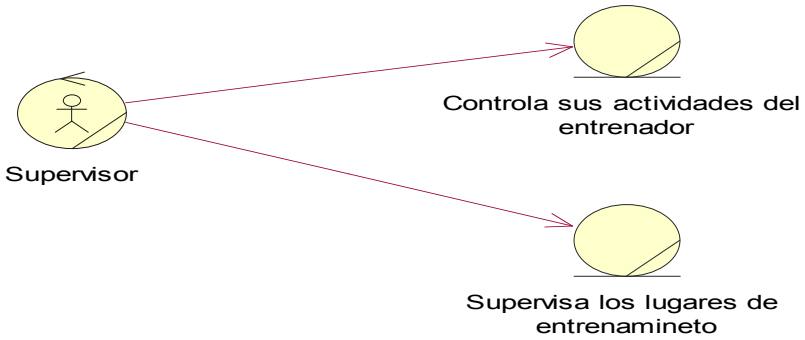


Figura 7 **Modelo del Supervisor del Dpto. de Deportes**

Entrenadores

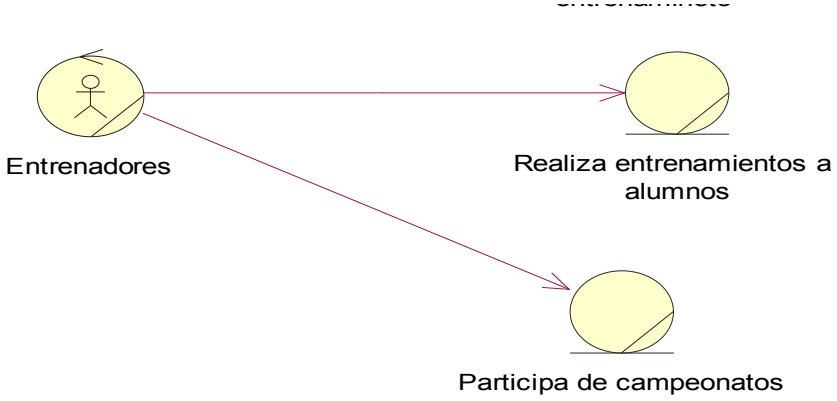


Figura 8 Modelo del Entrenador del Dpto. de Deportes

II.1.8 Glosario

II.1.8.1 Introducción

El presente documento recoge todos y cada uno de los términos manejados a lo largo de todo el proyecto de desarrollo del Sistema de información comunal. Se trata de un diccionario informal de datos y de definiciones de la nomenclatura que se maneja de tal modo que se crea estándar para todo el proyecto.

II.1.8.2 Propósito

El propósito de este glosario es definir con exactitud y sin ambigüedad la terminología manejada en el proyecto, también sirve como guía de consulta para la clarificación de los puntos conflictivos o poco esclarecidos del proyecto.

II.1.8.3 Alcance

El alcance del presente documento se extiende a todos los subsistemas definidos para el Departamento de Deportes, de tal modo que la terminología empleada en la unidad se refleja con claridad en este documento.

II.1.8.4 Referencias

El presente glosario hace referencia a los siguientes documentos:

- Plan de Desarrollo de Software
- Modelo de Casos de uso del Negocio

II.1.8.5 Definiciones

A continuación se presentan los términos manejados en el desarrollo del sistema.

- **Jefe del Departamento de Deportes**

Es la persona que se encarga de controlar que todas las funciones del sistema que lleven a cabo.

- **Coordinador**

Es la persona que se encarga de realizar los campeonatos.

- **Entrenadores**

El entrenador es una persona externa al Dpto. de deportes el cual puede ver sus horarios de entrenamiento, alumnos.

- **Alumnos**

Los alumnos pueden hacer que sea registrado y pueda ser controlado, con el fin de obtener una referencia de los datos.

- **Lugar de entrenamiento**

Es saber que barrios están dispuestos para los entrenamientos o campeonatos y con cuantas disciplinas se trabaja ese barrio.

- **Campeonato**

Es el conjunto de equipos que se programa para la realización de diferentes campeonatos.

- **Control**

El control es una etapa primordial en la administración, pues, aunque una empresa cuente con magníficos planes, una estructura organizacional adecuada y una dirección eficiente, el ejecutivo no podrá verificar cuál es la situación real de la organización no existe un mecanismo que se cerciore e informe si los hechos van de acuerdo con los objetivos.

- **Sistema**

Programa o conjunto de programas que efectúan la gestión de los procesos básicos de un sistema informático, y permite la normal ejecución del resto de las operaciones.

- **Tecnología**

Conjunto de teorías y de técnicas que permiten el aprovechamiento práctico del conocimiento científico.

- **Dato**

Información dispuesta de manera adecuada para su tratamiento por un ordenador.

II.1.9 Modelo de Casos de Uso

II.1.9.1 Introducción

El modelo de casos de uso es un artefacto de la disciplina de requisitos en la metodología RUP, la cual se está implementando.

Es un modelo del sistema que contiene actores, casos de uso y sus relaciones. En cada manera en que los actores usan el sistema se representa con un caso de uso, los mismos son fragmentados de funcionalidad, especifican una secuencia de acciones que el sistema puede llevar a cabo interactuando con sus actores.

II.1.9.2 Propósito

- Comprender la estructura y la dinámica del sistema deseado para la organización, para describir las acciones de los usuarios que interactúan con el sistema.

II.1.9.3 Alcance

- Definir los límites del sistema y las relaciones entre el sistema y el entorno.
- Identificar y definir los procesos del sistema según los objetivos de la organización.
- Modelar un caso de uso para cada proceso del sistema.

II.1.9.4 Diagrama de Casos de Uso

Diagrama de Casos de Usos: General

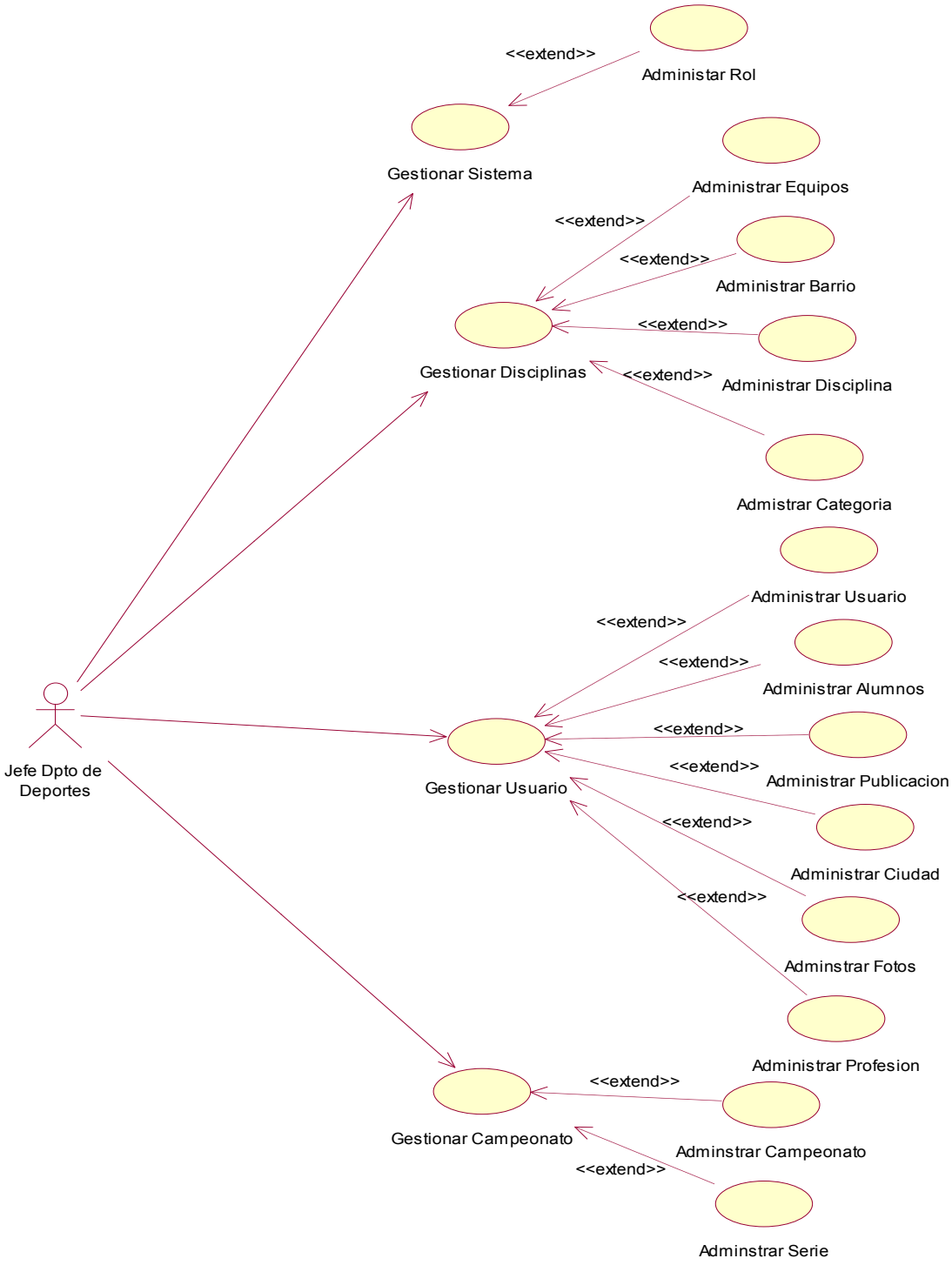


Figura 9 **Diagrama de Casos de Usos: General**

Iniciar Sesión

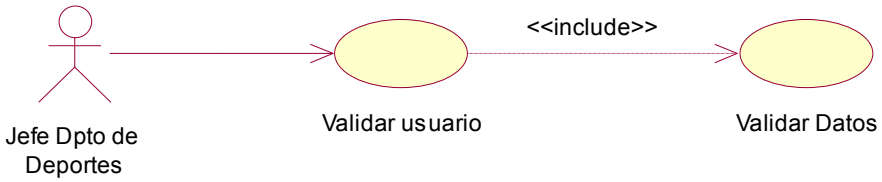


Figura 10 **Diagrama de Casos de Uso: Iniciar Sesión**

Gestionar Menú

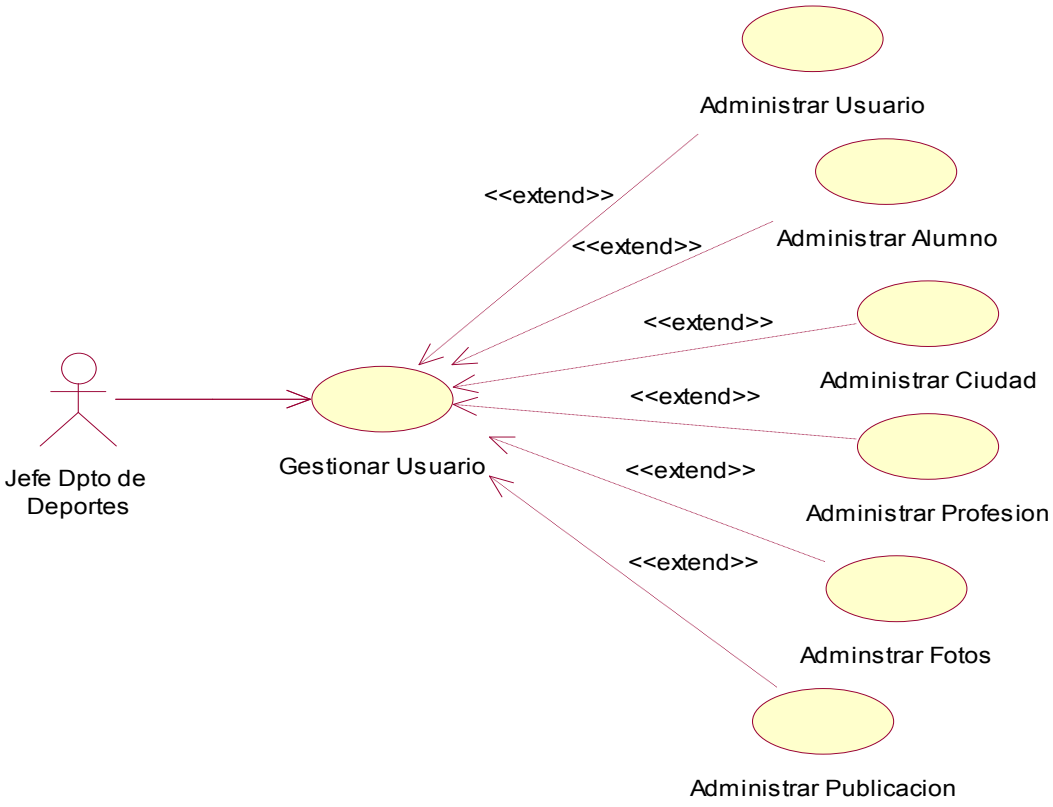


Figura 11 **Diagramas de Casos de Uso: Administrar Menú**

Administrar Usuarios

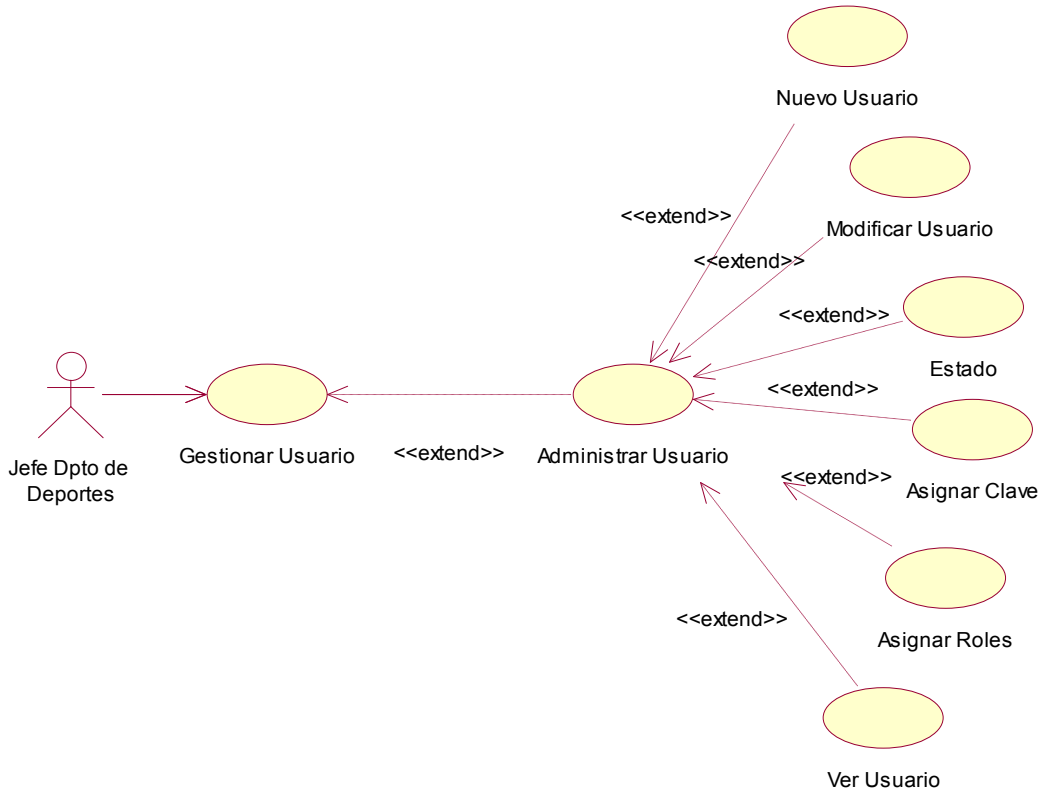


Figura 12 **Diagramas de Casos de Uso: Administrar Usuarios**

Administrar Profesión

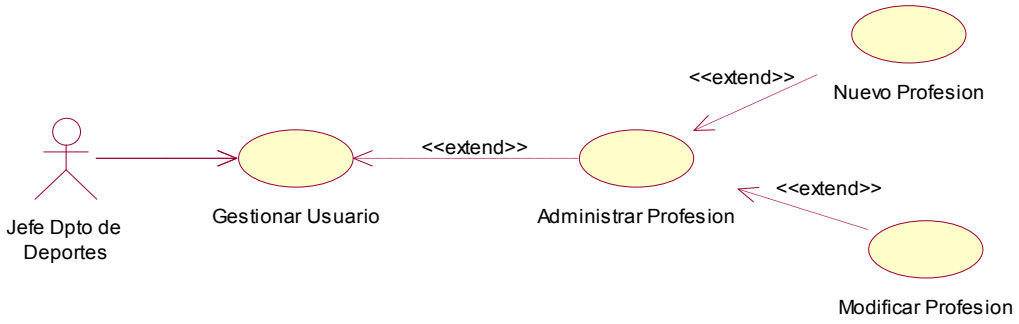


Figura 13 Diagramas de Casos de Uso: Administrar Profesión

Administrar Alumno

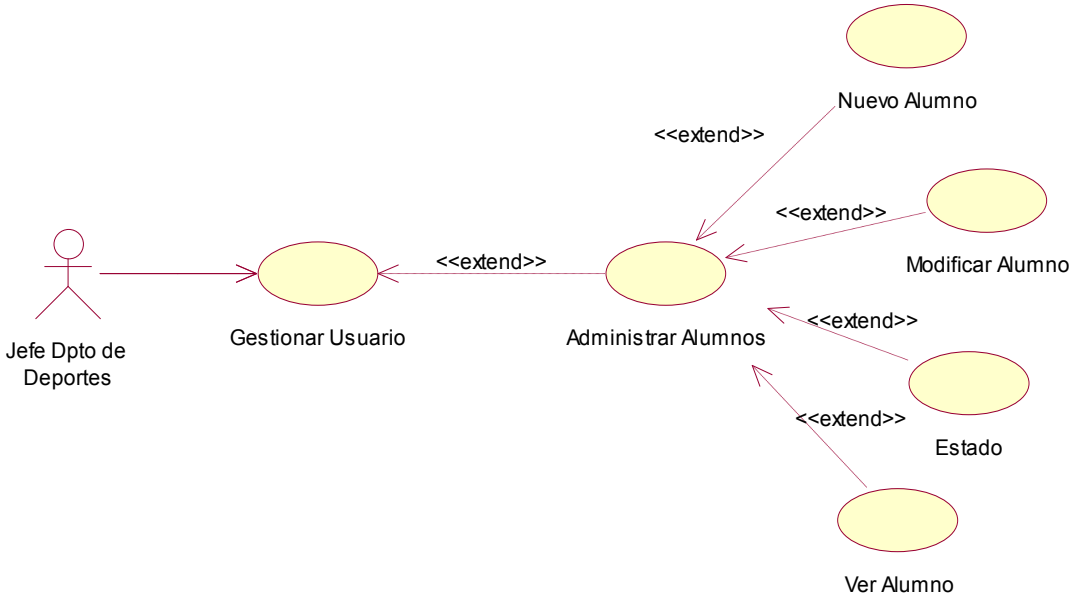


Figura 14 Diagramas de Casos de Uso: Administrar Alumno

Administrar Ciudad

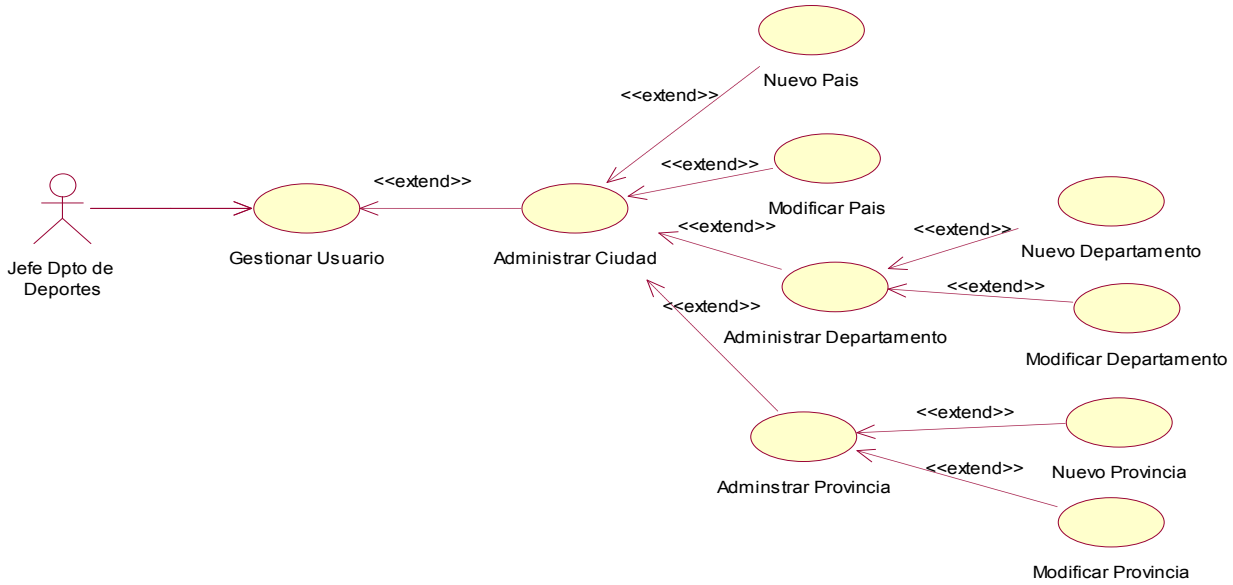


Figura 15 Diagramas de Casos de Uso: Administrar Ciudad

Gestionar Disciplina

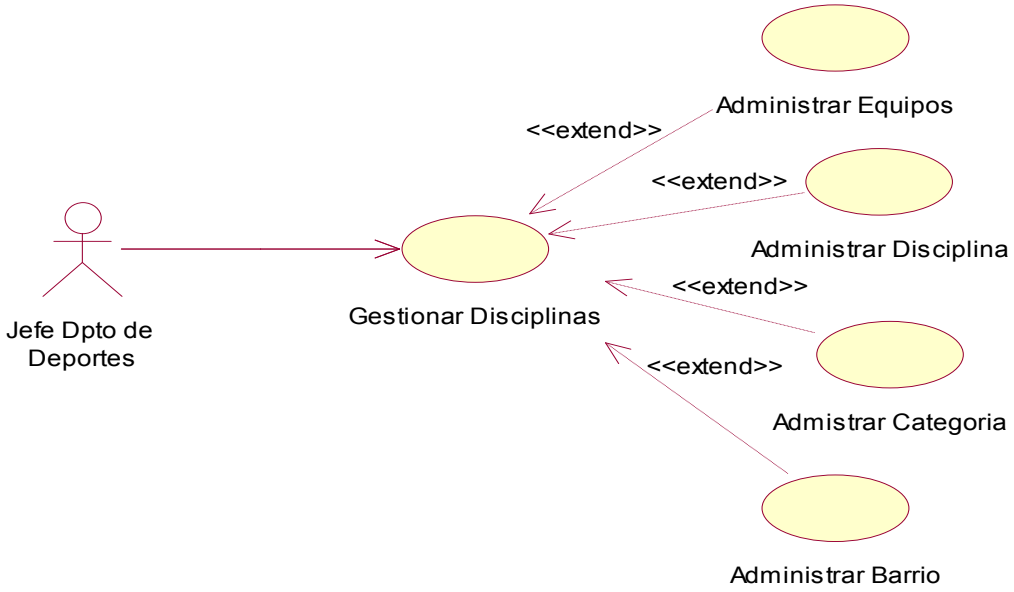


Figura 16 Diagramas de Casos de Uso: Gestionar Disciplina

Administrar Disciplina

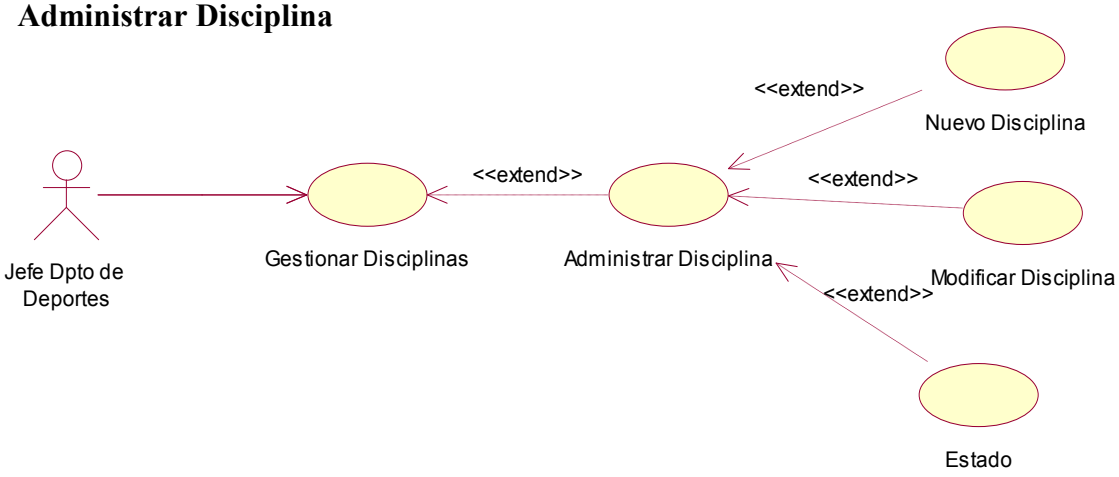


Figura 17 Diagramas de Casos de Uso: Administrar Disciplina

Administrar Barrio

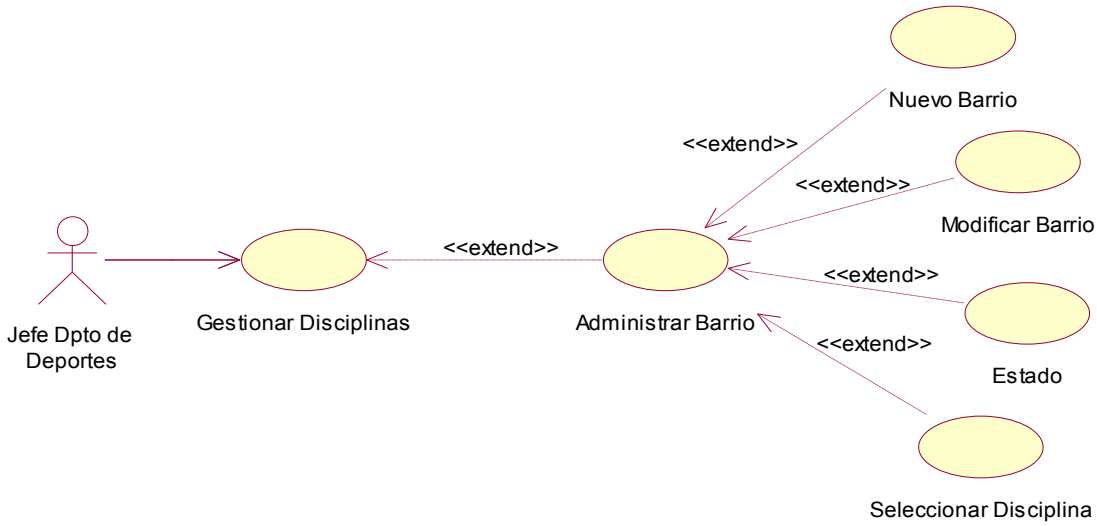


Figura 18 Diagramas de Casos de Uso: Administrar Barrio

Administrar Categoría

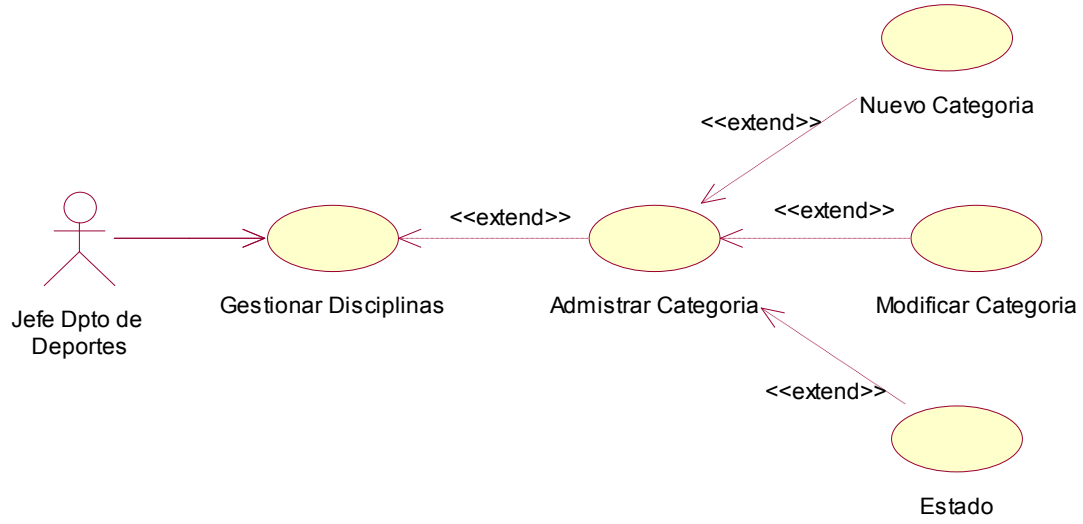


Figura 19 Diagramas de Casos de Uso: Administrar Categoría

Administrar Equipo

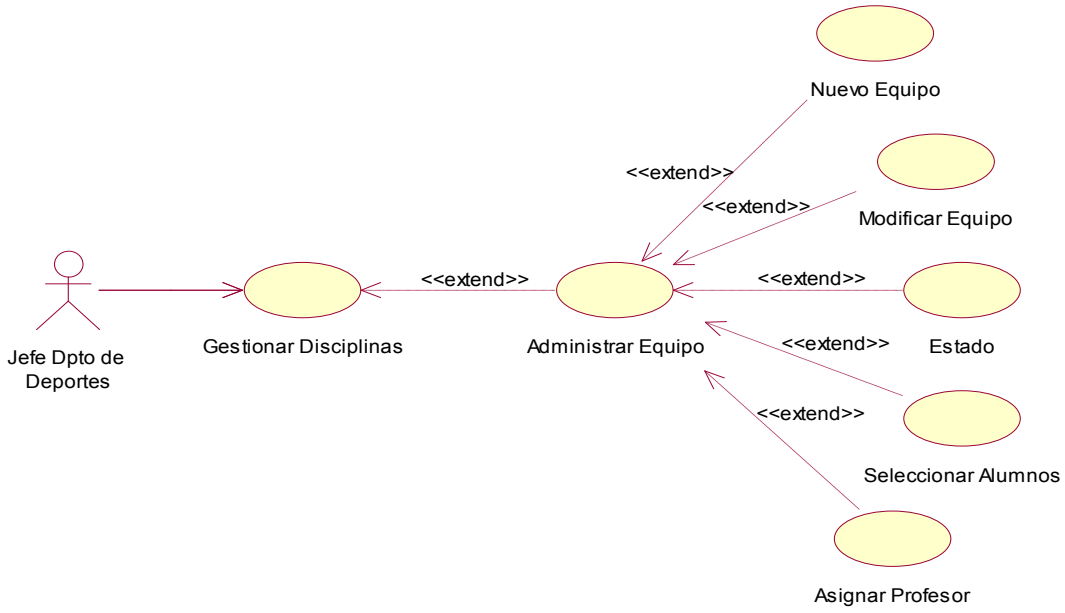


Figura 20 Diagramas de Casos de Uso: Administrar Equipo

Gestionar Campeonato

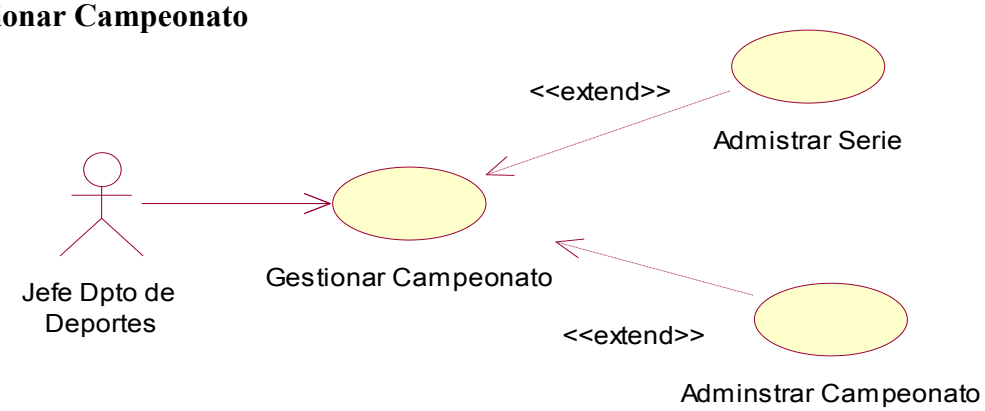


Figura 21 Diagramas de Casos de Uso: Gestionar Campeonato

Administrar Serie

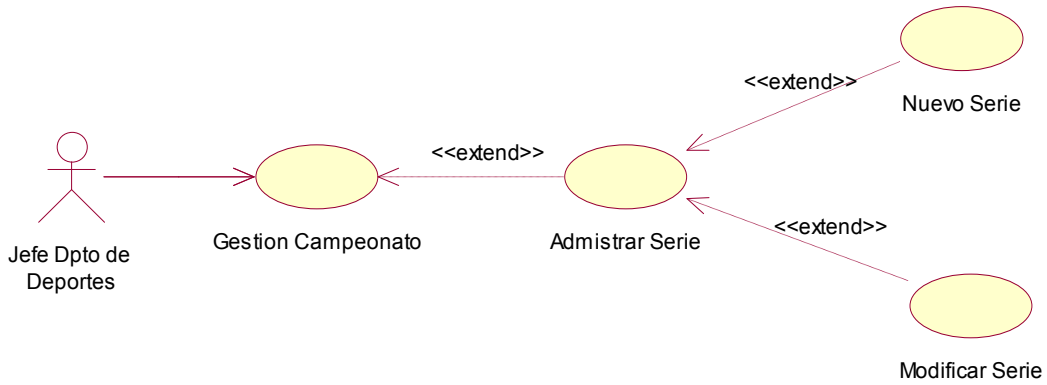


Figura 22 Diagramas de Casos de Uso: Adminstrar Serie

Administración Campeonato

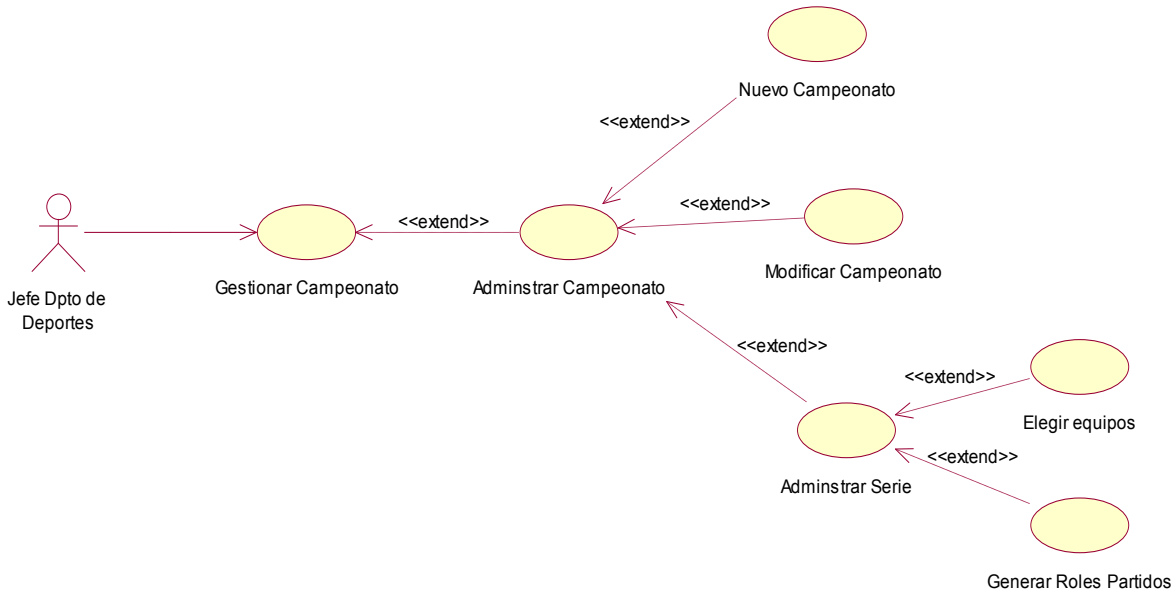


Figura 23 Diagramas de Casos de Uso: Administración Campeonato

Gestionar Sistema

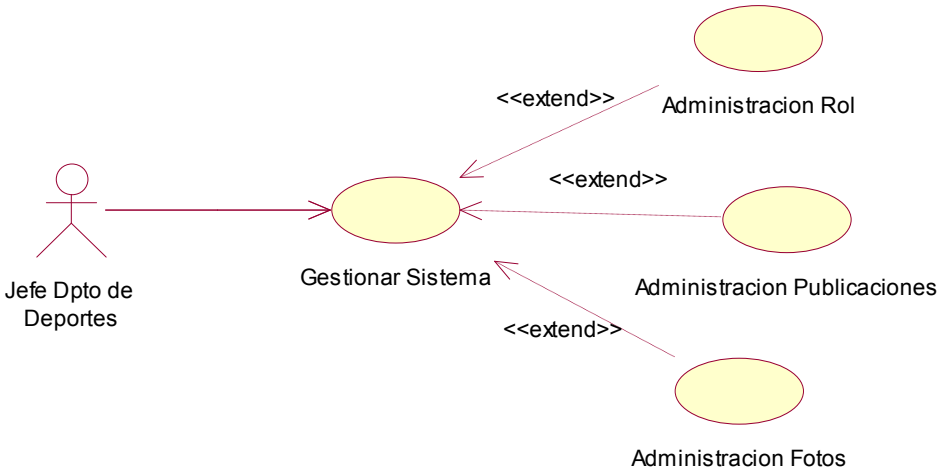


Figura 24 **Diagramas de Casos de Uso: Gestionar Sistema**

Administrar Rol

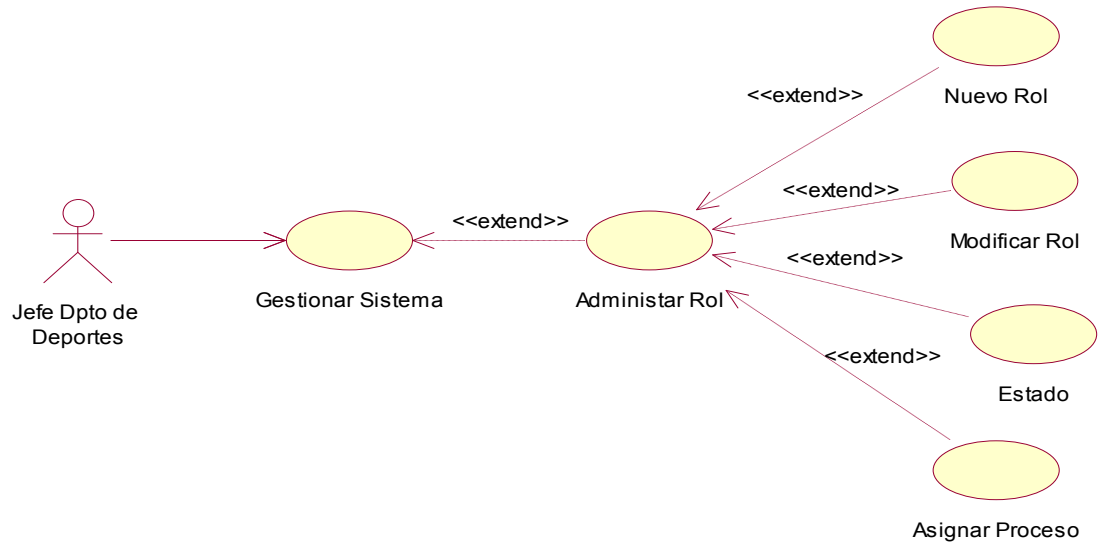


Figura 25 Diagramas de Casos de Uso: Administrar Rol

Administrar Publicaciones

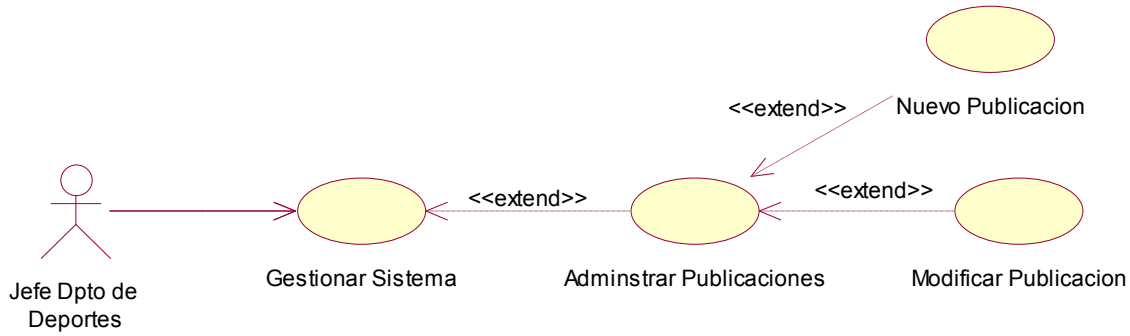


Figura 26 **Diagramas de Casos de Uso: Administrar Publicaciones**

Administrar Fotos

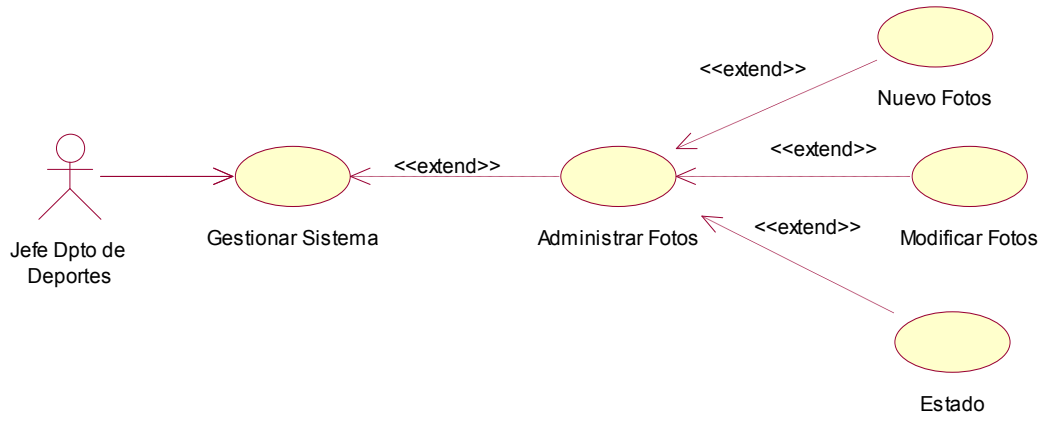


Figura 27 **Diagramas de Casos de Uso: Administrar Fotos**

Reporte

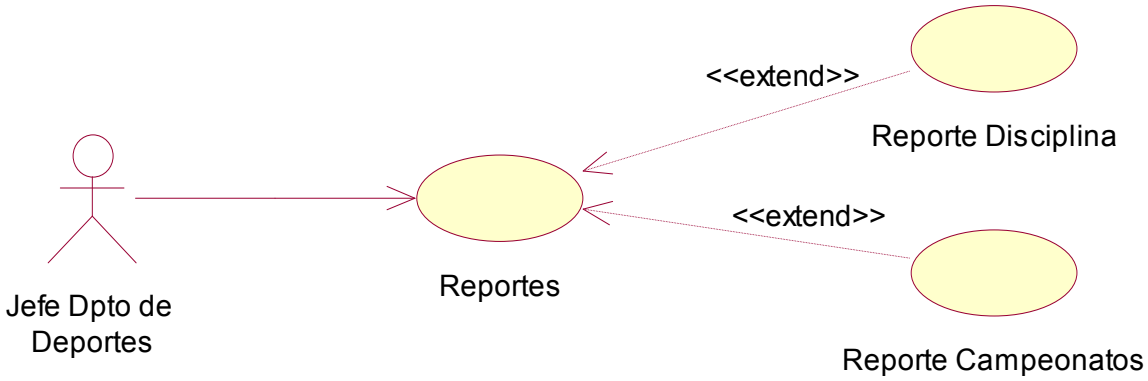


Figura 28 **Diagramas de Casos de Uso: Reporte**

II.1.10 Visión

II.1.10.1 Introducción

II.1.10.2 Propósito

El propósito de este documento es recoger, analizar y definir las necesidades de alto nivel, además de las características del "Mejoramiento de la Difusión de la Información, del área de Deportes para el Municipio de Cercado del Departamento de Tarija". El documento se centra en la funcionalidad requerida por la participante en el proyecto y los usuarios finales.

II.1.10.3 Alcance

El documento Visión se ocupa, como ya se había dicho anteriormente del **"**Mejoramiento de la Difusión de la Información, del área de Deportes para el Municipio de Cercado del Departamento de Tarija**".** Dicho sistema será desarrollado por la estudiante Univ. Carla Michel Romero alumna de quinto año de la materia Taller III de la Carrera de Ingeniería Informática, de la Facultad de Ciencias y Tecnología de la Universidad Autónoma Juan Misael Saracho.

El sistema permitirá administrar y publicar las actividades que realiza el Departamento de Deportes.

II.1.10.4 Definiciones, Acrónimos y Abreviaciones

RUP: Son las siglas de Rational Unified Process. Se trata de una metodología para describir el proceso de desarrollo de software.

II.1.10.5 Referencias

- Glosario
- Plan de Desarrollo de Software
- RUP (Rational Unified Process)
- Diagrama de Casos de Uso.

II.1.10.6 Posicionamiento

II.1.10.6.1 Oportunidad de Negocio

Este Sistema permitirá al Departamento de Deportes informatizar todas sus actividades y servicios de tal manera todos los datos accedidos por personas externas al Departamento estarán siempre actualizados, lo cual es un factor muy importante para poder proveer de una información confiable del Departamento de Deportes.

II.1.10.6.2 Sentencia que define el problema

El problema de	El no contar con una técnica de difusión de la información para poder llegar a las personas, con una información adecuada.
Afectan a	• Director del Departamento de Deportes. • Municipio del Departamento de Tarija. • Al personal del Departamento de Deportes.
El Impacto asociado es	Almacenar toda la información referente a las actividades realizadas y administración del personal de la unidad y que esta información este al instante y actualizada lo cual sería imposible sin un sistema web.
Una solución adecuada	Informar los respectivos procesos, usando el sistema web generando interfaces amigables y sencillas para los usuarios para que puedan navegar con una menor dificultad, permitiendo a los mismos informarse de toda la actividad

	del Departamento de Deportes.

Tabla 20: Sentencia que define el problema

II.1.10.6.3 Sentencia que define la posición del Producto

Para	• Personal involucrado del Departamento de Deportes. • Director del Departamento de Deportes.
Quienes	Controlan y llevan a cabo los diferentes procesos del Departamento de Deportes.
El nombre del producto	Es una herramienta de software.
Que	Almacenan la información necesaria para difundirla.
No como	El sistema actual.
El producto	Permite administrar las distintas actividades del Departamento de Deportes y propagarla mediante una interfaz gráfica sencilla y amigable.

Tabla 21: Sentencia que define la posición del producto

II.1.10.6.4 Descripción de Stakeholders (Participantes en el proyecto) y Usuarios

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

II.1.10.6.5 Resumen de Stakeholders

Nombre	Descripción	Responsabilidades
Dr. Roberto Pablo Aban Lenz	Director del Departamento de Deportes	El Stakeholders realiza: Representa a todos los posibles del sistema. Seguimiento del desarrollo del proyecto. Aprueba requisitos y funcionalidades
Lic. Deysi Arancibia	Docentes de Taller III	Seguimiento y control del Desarrollo del Proyecto.

Tabla 22: Resumen de Stakeholders

II.1.10.6.6 Perfil de los Stakeholders (participantes en el proyecto)

Director

Representante	Dr. Roberto Pablo Aban Lenz
Descripción	Director del Departamento de Deportes
Responsabilidades	Encargado de mostrar todas las necesidades de cada usuario del sistema. Además lleva a cabo un seguimiento del desarrollo del proyecto y aprobación de los requisitos.
Comentarios	Ninguno

Tabla 23: Perfil de usuario – director

Docente

Representante	Lic. Deysi Arancibia
Descripción	Docente de Taller III
Responsabilidades	Seguimiento y control de desarrollo
Comentario	Ninguno

Tabla 24: Perfil de usuario – Docente

II.1.10.6.7 Descripción Global del Producto

II.1.10.6.8 Perspectiva del Producto

El producto a desarrollar es un sistema para el Departamento de Deportes de la Ciudad de Tarija, con la intención de Difundir las actividades realizadas.

II.1.10.6.9 Resumen de Características

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

Beneficio del Usuario	**Características que lo apoyan**
Con el sistema web se pretende difundir las actividades que realiza el Departamento de Deportes, que sea una fuente de información a la población en general y un control mejor de los funcionarios como también de los alumnos inscritos en las diferentes disciplinas.	Sistema de difusión de la información de las actividades realizadas como también administrar a los funcionarios dependientes del Departamento de Deportes.
Mayor facilidad para la actualización de los datos a difundir.	Base de Datos con acceso remoto desde la que se puede gestionar toda la información.

Tabla 25: Resumen de características

I.1.1 Especificaciones de Caso de Uso

I.1.1.1 Introducción

Las especificaciones de los Casos de Usos es una descripción detallada de los Casos de Uso del sistema.

I.1.1.2 Propósito

- Comprender los casos de uso del sistema.
- Describir específicamente cada caso de uso.

I.1.1.3 Alcance

- Describe los procesos internos de los casos de uso.
- Detallar los procesos del sistema y los clientes.

I.1.1.4 Especificaciones de Caso de Uso

Caso de Uso: Administrar usuarios

Actor: Administrador

Explicación: En el caso Administrar usuarios, el administrador del sistema podrá tener un control de los datos de los usuarios del comercial, con la finalidad de que se pueda adicionar y modificar a dichos usuarios.

Descripción:

Administrar Usuarios

Descripción: Nuevo Usuario

Caso de Uso: Adicionar Usuario
Actor: Administrador
1. El administrador ingresa los datos del Nuevo Usuario.
2. El sistema guarda los datos del Nuevo Usuario y devuelve una lista actualizada.
3. El administrador observa la lista actualizada donde se visualiza el Nuevo Usuario.
4. Fin del caso.

Tabla 26: Especificación caso de uso Nuevo Usuario

Descripción: Modificar Usuario

Caso de Uso: Modificar Usuario
Actor: Administrador
1. El administrador selecciona el Usuario a Modificar.
2. El sistema devuelve una pantalla para Modificar datos del Usuario seleccionado.
3. El administrador Modifica datos del Usuario seleccionado.
4. El sistema guarda datos del Usuario Modificado y devuelve una lista actualizada de Usuarios.
5. El administrador observa la lista actualizada donde se visualiza los datos Modificados del Usuario.
6. Fin del Caso.

Tabla 27: Especificación caso de uso Modificar Usuario

Descripción: Estado al Usuario

Caso de Uso: Estado al Usuario
Actor: Administrador
1. El administrador selecciona el Usuario para poner en un Estado de activo o inactivo.
2. El administrador si selecciona inactivo al Usuario entonces está bloqueado para sus funciones dentro del sistema.
3. El administrador si selecciona activo al Usuario entonces está habilitado para sus funciones dentro del sistema.
4. El administrador observa la lista actualizada donde se visualiza los datos de Estado del Usuario.
5. Fin del Caso.

Tabla 28: Especificación caso de uso Estado al Usuario

Descripción: Asignar Login / Clave Usuario

Caso de Uso: Asignar Clave Usuario
Actor: Administrador
1. El administrador selecciona el Usuario para Asignar su Login y Clave.
2. El sistema devuelve una pantalla para Asignar Clave al Usuario seleccionado.
3. El administrador escribe su login y clave para el Usuario seleccionado.
4. El sistema guarda su login y clave del Usuario y devuelve una lista actualizada de Usuarios.
5. Fin del Caso.

Tabla 29: Especificación caso de uso Asignar Login / Clave Usuario

Descripción: Asignar Roles al Usuario

Caso de Uso: Asignar Roles al Usuario
Actor: Administrador
1. El administrador selecciona el Usuario para Asignar Roles.
2. El sistema devuelve una pantalla para Asignar Roles al Usuario seleccionado.
3. El administrador selecciona roles para Asignar al Usuario seleccionado.
4. El sistema guarda sus Roles Asignados al Usuario y devuelve una lista actualizada de Usuarios.
5. Fin del Caso.

Tabla 30: Especificación caso de uso Asignar Roles al Usuario

Descripción: Buscar Usuario

Caso de Uso: Buscar Usuario
Actor: Administrador
1. El administrador escribe el nombre o apellido del Usuario en el Buscador.
2. El sistema Busca y le lista al Usuario buscado.
3. Fin del Caso.

Tabla 31: Especificación caso de uso Buscar Usuario

Descripción: Ver Usuario

Caso de Uso: Ver Usuario
Actor: Administrador
1. El administrador.
2. El sistema Busca y le lista al Usuario buscado.
3. Fin del Caso.

Tabla 32: Especificación caso de uso Ver Usuario

Descripción: Nuevo Profesión

Caso de Uso: Nuevo Profesión
Actor: Administrador
1. El administrador ingresa los datos de la nueva profesión
2. El sistema guarda los datos de la nueva profesión y devuelve una lista actualizada.
3. El administrador observa la lista actualizada donde se visualiza la nueva profesión.
4. Fin del caso.

Tabla 33: Especificación caso de uso Nueva Profesión

Descripción: Modificar Profesión

Caso de Uso: Modificar Profesión
Actor: Administrador
1. El administrador selecciona la Profesión a Modificar.
2. El sistema devuelve una pantalla para Modificar datos de la Profesión seleccionada.
3. El administrador Modifica datos de la Profesión seleccionada.
4. El sistema guarda datos de la Profesión Modificados y devuelve una lista actualizada de Profesiones.
5. El administrador observa la lista actualizada donde se visualiza los datos Modificados de la Profesión.
6. Fin del Caso.

Tabla 34: Especificación caso de uso Modificar Profesión

Descripción: Nuevo Alumno

Caso de Uso: ***Nuevo Alumno***
Actor: Administrador
5. El administrador ingresa los datos del ***Nuevo Alumno***.
6. El sistema guarda los datos del ***Nuevo Alumno*** y devuelve una lista actualizada.
7. El administrador observa la lista actualizada donde se visualiza el ***Nuevo Alumno.***
8. Fin del caso.

Tabla 355: Especificación caso de uso Nuevo Alumno

Descripción: Modificar Alumno

Caso de Uso: ***Modificar Alumno***
Actor: Administrador
7. El administrador selecciona el Alumno a Modificar.
8. El sistema devuelve una pantalla para Modificar datos del Alumno seleccionado.
9. El administrador Modifica datos del Alumno seleccionado.
10. El sistema guarda datos del Alumno Modificado y devuelve una lista actualizada de Usuarios.
11. El administrador observa la lista actualizada donde se visualiza los datos Modificados del Alumno.
12. Fin del Caso.

Tabla 36: Especificación caso de uso Modificar Alumno

Descripción: Estado al Alumno

Caso de Uso: Estado al **Alumno**
Actor: Administrador
6. El administrador selecciona el **Alumno** para poner en un Estado de activo o inactivo.
7. El administrador si selecciona inactivo al **Alumno** entonces está bloqueado para sus funciones dentro del sistema.
8. El administrador si selecciona activo al **Alumno** entonces está habilitado para sus funciones dentro del sistema.
9. El administrador observa la lista actualizada donde se visualiza los datos de Estado del **Alumno**.
10. Fin del Caso.

Tabla 367: Especificación caso de uso Estado Alumno

Descripción: Nuevo País

Caso de Uso: Nuevo País
Actor: Administrador
1. El administrador ingresa los datos del nuevo país.
2. El sistema guarda los datos del nuevo país y devuelve una lista actualizada.
3. El administrador observa la lista actualizada donde se visualiza el nuevo país.
4. Fin del caso.

Tabla 38: Especificación caso de uso Nueva País

Descripción: Modificar País

Caso de Uso: Modificar país
Actor: Administrador
1. El administrador selecciona el país a Modificar.
2. El sistema devuelve una pantalla para Modificar datos del país seleccionada.
3. El administrador Modifica datos del país seleccionado.
4. El sistema guarda datos del país modificado y devuelve una lista actualizada de países.
5. El administrador observa la lista actualizada donde se visualiza los datos Modificados del país.
6. Fin del Caso.

Tabla 39: Especificación caso de uso Modificar País

Descripción: Nuevo Departamento

Caso de Uso: Nuevo departamento
Actor: Administrador
1. El administrador ingresa los datos del nuevo departamento.
2. El sistema guarda los datos del nuevo departamento y devuelve una lista actualizada.
3. El administrador observa la lista actualizada donde se visualiza el nuevo departamento.
4. Fin del caso.

Tabla 40: Especificación caso de uso Nuevo Departamento

Descripción: Modificar Departamento

Caso de Uso: Modificar Departamento
Actor: Administrador
1. El administrador selecciona el departamento a Modificar.
2. El sistema devuelve una pantalla para Modificar datos del departamento seleccionado.
3. El administrador Modifica datos del departamento seleccionado.
4. El sistema guarda datos del país modificado y devuelve una lista actualizada de departamentos.
5. El administrador observa la lista actualizada donde se visualiza los datos Modificados del departamento.
6. Fin del Caso.

Tabla 41: Especificación caso de uso Modificar Departamento

Descripción: Nueva Provincia

Caso de Uso: Nueva Provincia
Actor: Administrador
1. El administrador ingresa los datos de la nueva provincia
2. El sistema guarda los datos de la nueva provincia y devuelve una lista actualizada.
3. El administrador observa la lista actualizada donde se visualiza la nueva provincia.
4. Fin del caso.

Tabla 42: Especificación caso de uso Nueva Provincia

Descripción: Modificar Provincia

Caso de Uso: Modificar Provincia
Actor: Administrador
1. El administrador selecciona la provincia a Modificar.
2. El sistema devuelve una pantalla para Modificar datos de la provincia seleccionada.
3. El administrador Modifica datos de la provincia seleccionada.
4. El sistema guarda datos de la provincia Modificados y devuelve una lista actualizada de provincias.
5. El administrador observa la lista actualizada donde se visualiza los datos Modificados de la provincia.
6. Fin del Caso.

Tabla 43: Especificación caso de uso Modificar Provincia

Administrar Disciplina

Descripción: Nueva disciplina

Caso de Uso: Nueva Disciplina
Actor: Administrador
1. El administrador ingresa los datos la nueva disciplina
2. El sistema guarda los datos de la nueva disciplina y devuelve una lista actualizada.
3. El administrador observa la lista actualizada donde se visualiza la nueva disciplina.
4. Fin del caso.

Tabla 44: Especificación caso de uso Nueva Disciplina

Descripción: Modificar Disciplina

Caso de Uso: Modificar Rol
Actor: Administrador
1. El administrador selecciona la disciplina a Modificar.
2. El sistema devuelve una pantalla para Modificar datos de la disciplina seleccionado.
3. El administrador Modifica datos de la disciplina seleccionada.
4. El sistema guarda datos del Rol Modificado y devuelve una lista actualizada de Roles.
5. El administrador observa la lista actualizada donde se visualiza los datos Modificados de la disciplina.
6. Fin del Caso.

Tabla 45: Especificación caso de uso Modificar Disciplina

Descripción: Estado Disciplina

Caso de Uso: Estado Disciplina
Actor: Administrador
1. El administrador selecciona la disciplina
2. El administrador selecciona activo o inactivo.
3. Fin del caso.

Tabla 46: Especificación caso de uso Estado

Descripción: Nuevo Equipo

Caso de Uso: Nuevo equipo
Actor: Administrador
1. El administrador ingresa los datos del nuevo equipo
2. El sistema guarda los datos del nuevo equipo y devuelve una lista actualizada.
3. El administrador observa la lista actualizada donde se visualiza el nuevo equipo.
4. Fin del caso.

Tabla 47: Especificación caso de uso Nuevo Equipo

Descripción: Modificar Equipo

Caso de Uso: Modificar Equipo
Actor: Administrador
1. El administrador selecciona la disciplina a Modificar.
2. El sistema devuelve una pantalla para Modificar datos del equipo seleccionado.
3. El administrador Modifica datos del equipo seleccionado.
4. El sistema guarda datos del equipo Modificado y devuelve una lista actualizada de equipos.
5. El administrador observa la lista actualizada donde se visualiza los datos Modificados del equipo.
6. Fin del Caso.

Tabla 48: Especificación caso de uso Modificar Equipo

Descripción: Estado Equipo

Caso de Uso: Estado Equipo
Actor: Administrador
1. El administrador selecciona el equipo
2. El administrador selecciona activo o inactivo.
3. Fin del caso.

Tabla 49: Especificación caso de uso Estado Equipo

Descripción: Nueva Categoría

Caso de Uso: Nueva Categoría
Actor: Administrador
1. El administrador ingresa los datos de la nuevo categoría
2. El sistema guarda los datos de la nueva categoría y devuelve una lista actualizada.
3. El administrador observa la lista actualizada donde se visualiza la nueva categoría.
4. Fin del caso.

Tabla 50: Especificación caso de uso Nueva Categoría

Descripción: Modificar Categoría

Caso de Uso: Modificar Categoría
Actor: Administrador
1. El administrador selecciona la categoría a Modificar.
2. El sistema devuelve una pantalla para Modificar datos de la categoría seleccionada.
3. El administrador Modifica datos de la categoría seleccionada.
4. El sistema guarda datos de la categoría Modificado y devuelve una lista actualizada de equipos.
5. El administrador observa la lista actualizada donde se visualiza los datos Modificados de la categoría.
6. Fin del Caso.

Tabla 51: Especificación caso de uso Modificar Categoría

Descripción: Estado Categoría

Caso de Uso: Estado Categoría
Actor: Administrador
1. El administrador selecciona la categoría.
2. El administrador selecciona activo o inactivo.
3. Fin del caso.

Tabla 52: Especificación caso de uso Estado Categoría

Descripción: Nuevo Barrio

Caso de Uso: Nuevo Barrio
Actor: Administrador
1. El administrador ingresa los datos del nuevo barrio
2. El sistema guarda los datos del nuevo barrio y devuelve una lista actualizada.
3. El administrador observa la lista actualizada donde se visualiza el nuevo barrio.
4. Fin del caso.

Tabla 53: Especificación caso de uso Nuevo Barrio

Descripción: Modificar Barrio

Caso de Uso: Modificar Barrio
Actor: Administrador
7. El administrador selecciona el barrio a Modificar.
1. El sistema devuelve una pantalla para Modificar datos del barrio seleccionado.
2. El administrador Modifica datos del barrio seleccionado.
3. El sistema guarda datos del barrio Modificado y devuelve una lista actualizada de barrios.
4. El administrador observa la lista actualizada donde se visualiza los datos Modificados del barrio.
5. Fin del Caso.

Tabla 54: Especificación caso de uso Modificar Barrio

Descripción: Estado Barrio

Caso de Uso: Estado Barrio
Actor: Administrador
1. El administrador selecciona el barrio
2. El administrador selecciona activo o inactivo.
3. Fin del caso.

Tabla 55: Especificación caso de uso Estado Barrio

Administración Campeonato

Descripción: Nuevo Campeonato

Caso de Uso: Adicionar campeonato
Actor: Administrador
1. El administrador ingresa los datos del nuevo campeonato
2. El sistema guarda los datos del nuevo campeonato y devuelve una lista actualizada.
3. El administrador observa la lista actualizada donde se visualiza el nuevo campeonato.
4. Fin del caso.

Tabla 56: Especificación caso de uso Nuevo Campeonato

Descripción: Modificar Campeonato

Caso de Uso: Modificar Campeonato
Actor: Administrador
1. El administrador selecciona el campeonato a Modificar.
2. El sistema devuelve una pantalla para Modificar datos del campeonato seleccionado.
3. El administrador Modifica datos del campeonato seleccionado.
4. El sistema guarda datos del campeonato Modificado y devuelve una lista actualizada de campeonatos.
5. El administrador observa la lista actualizada donde se visualiza los datos Modificados del campeonato.
6. Fin del Caso.

Tabla 57: Especificación caso de uso Modificar Campeonato

Descripción: Nueva Serie

Caso de Uso: Nueva Serie
Actor: Administrador
1. El administrador ingresa los datos de la nueva serie
2. El sistema guarda los datos de la nueva serie y devuelve una lista actualizada.
3. El administrador observa la lista actualizada donde se visualiza la nueva serie.
4. Fin del caso.

Tabla 58: Especificación caso de uso Nueva Serie

Descripción: Modificar Serie

Caso de Uso: Modificar Serie
Actor: Administrador
1. El administrador selecciona la serie a Modificar.
2. El sistema devuelve una pantalla para Modificar datos de la serie seleccionada.
3. El administrador Modifica datos de la serie seleccionada.
4. El sistema guarda datos dela serie modificada y devuelve una lista actualizada de series.
5. El administrador observa la lista actualizada donde se visualiza los datos Modificados de la serie.
6. Fin del Caso.

Tabla 59: Especificación caso de uso Modificar Serie

Administrar Sistema

Descripción: Nuevo Rol

Caso de Uso: Nuevo Rol
Actor: Administrador
1. El administrador ingresa los datos del Nuevo Rol.
2. El sistema guarda los datos del Nuevo Rol y devuelve una lista actualizada.
3. El administrador observa la lista actualizada donde se visualiza el Nuevo Rol.
4. Fin del caso.

Tabla 60: Especificación caso de uso Nuevo Rol

Descripción: Modificar Rol

Caso de Uso: Modificar Rol
Actor: Administrador
1. El administrador selecciona el Rol a Modificar.
2. El sistema devuelve una pantalla para Modificar datos del Rol seleccionado.
3. El administrador Modifica datos del Rol seleccionado.
4. El sistema guarda datos del Rol Modificado y devuelve una lista actualizada de Roles.
5. El administrador observa la lista actualizada donde se visualiza los datos Modificados del Rol.
6. Fin del Caso.

Tabla 61: Especificación caso de uso Modificar Rol

Descripción: Nueva Publicación

Caso de Uso: ***Nueva*** publicación
Actor: Administrador
1. El administrador ingresa los datos de la nueva publicación.
2. El sistema guarda los datos de la nueva publicación y devuelve una lista actualizada.
3. El administrador observa la lista actualizada donde se visualiza la nueva publicación.
4. Fin del caso.

Tabla 62: Especificación caso de uso Nueva Publicación

Descripción: Modificar Publicación

Caso de Uso: Modificar publicación
Actor: Administrador
1. El administrador selecciona la publicación a Modificar.
2. El sistema devuelve una pantalla para Modificar datos de la publicación seleccionada.
3. El administrador Modifica datos de la publicación seleccionada.
4. El sistema guarda datos de la publicación modificada y devuelve una lista actualizada de publicaciones.
5. El administrador observa la lista actualizada donde se visualiza los datos Modificados de la publicación.
6. Fin del Caso.

Tabla 63: Especificación caso de uso Modificar Publicación

***Descripción* Nueva Foto**

Caso de Uso: Nueva Foto
Actor: Administrador
1. El administrador ingresa la nueva foto.
2. El sistema guarda la nueva fotografía y devuelve una lista actualizada.
3. El administrador observa la lista actualizada donde se visualiza la nueva fotografía.
4. Fin del caso.

Tabla 64: Especificación caso de uso Nueva Foto

Descripción: Modificar Foto

Caso de Uso: Modificar foto
Actor: Administrador
1. El administrador selecciona la fotografía a Modificar.
2. El sistema devuelve una pantalla para Modificar la fotografía.
3. El administrador observa la lista actualizada donde se visualiza las fotografías del departamento.
4. Fin del Caso.

Tabla 65: Especificación caso de uso Modificar Foto

I.1.2 Modelado Diagrama de Actividades

I.1.2.1 Introducción

Mediante el uso de los diagramas de actividades podemos modelar el flujo de control entre actividades del sistema. La idea es generar una especie de diagrama Pert, en el que se puede ver el flujo actividades que tienen lugar a lo largo del tiempo, así como las tareas concurrentes que pueden realizarse a la vez. Gráficamente es un conjunto de arcos y nodos desde un punto de vista conceptual, el diagrama de actividades muestra como fluye el control de unas clases a otras con la finalidad de culminar con un flujo de control total que se corresponde con la consecución de un proceso más complejo. Por este motivo, aparecerán acciones y actividades correspondientes a distintas clases, colaborando todas ellas para conseguir un mismo fin.

I.1.2.2 Propósito

- Comprender los casos de uso del sistema
- Comprender la estructura y la dinámica del sistema.

I.1.2.3 Alcance

- Describir los procesos del sistema y los clientes
- 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 del sistema.

I.1.1.1 Diagrama de Actividades

Iniciar Sesión

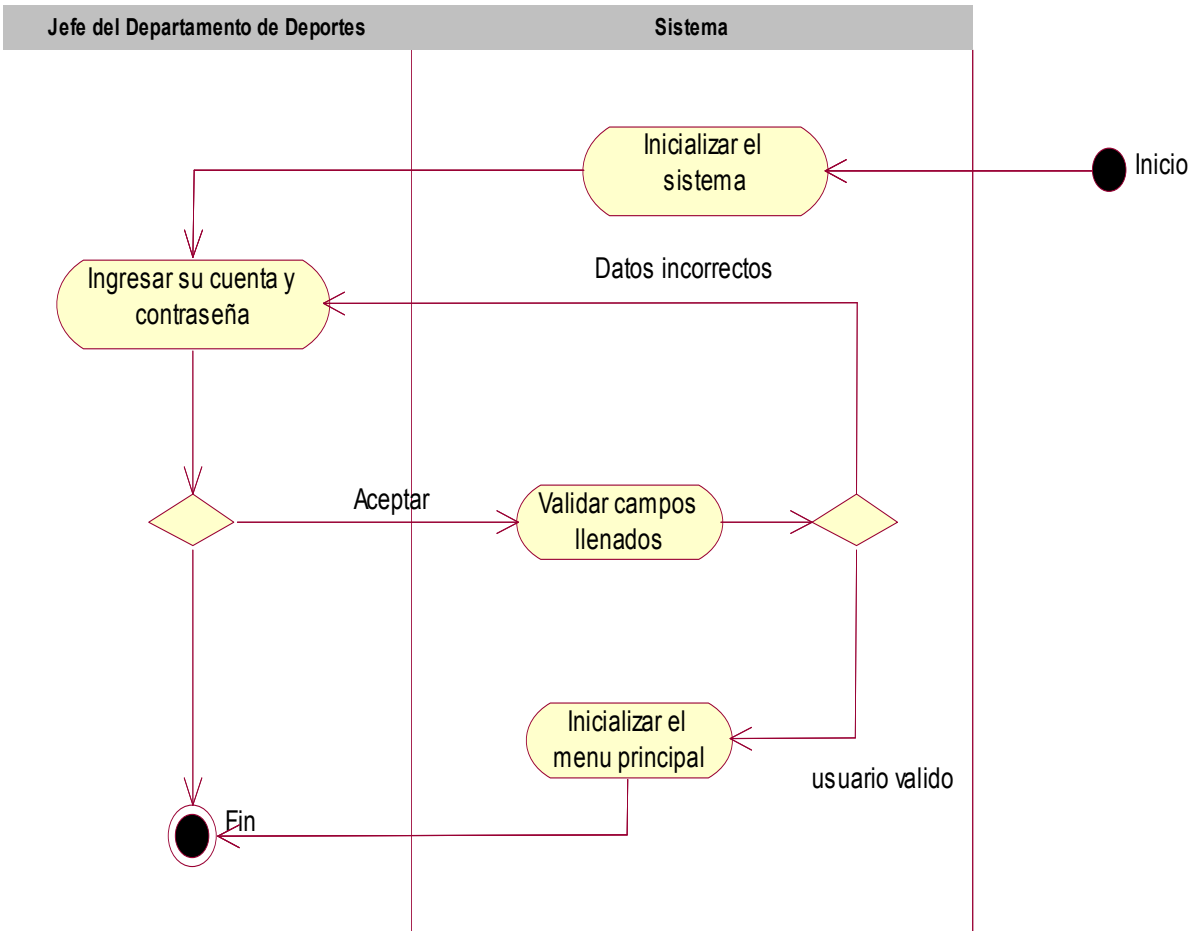


Figura 66: Diagrama de actividades: Iniciar Sesión

Administrar Usuario

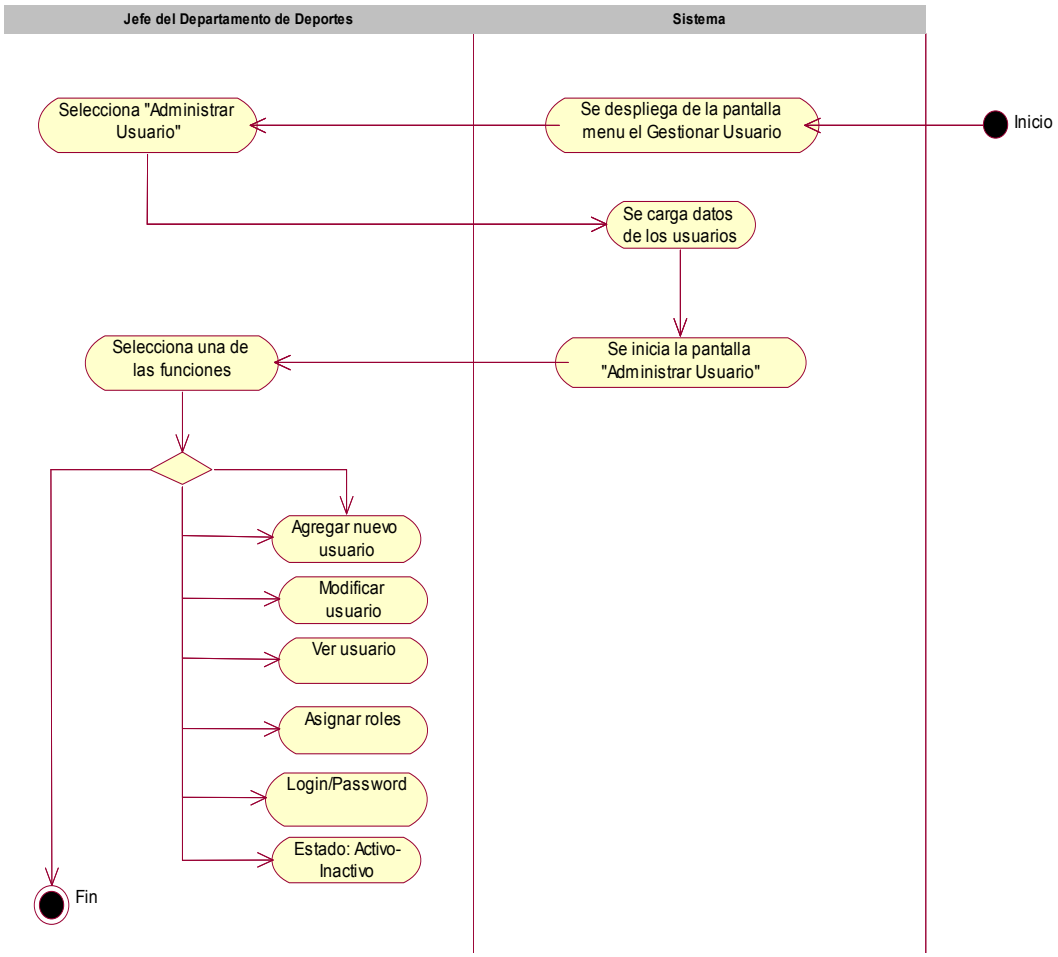


Figura 67: Diagrama de actividades: Administrar Usuario

Nuevo Usuario

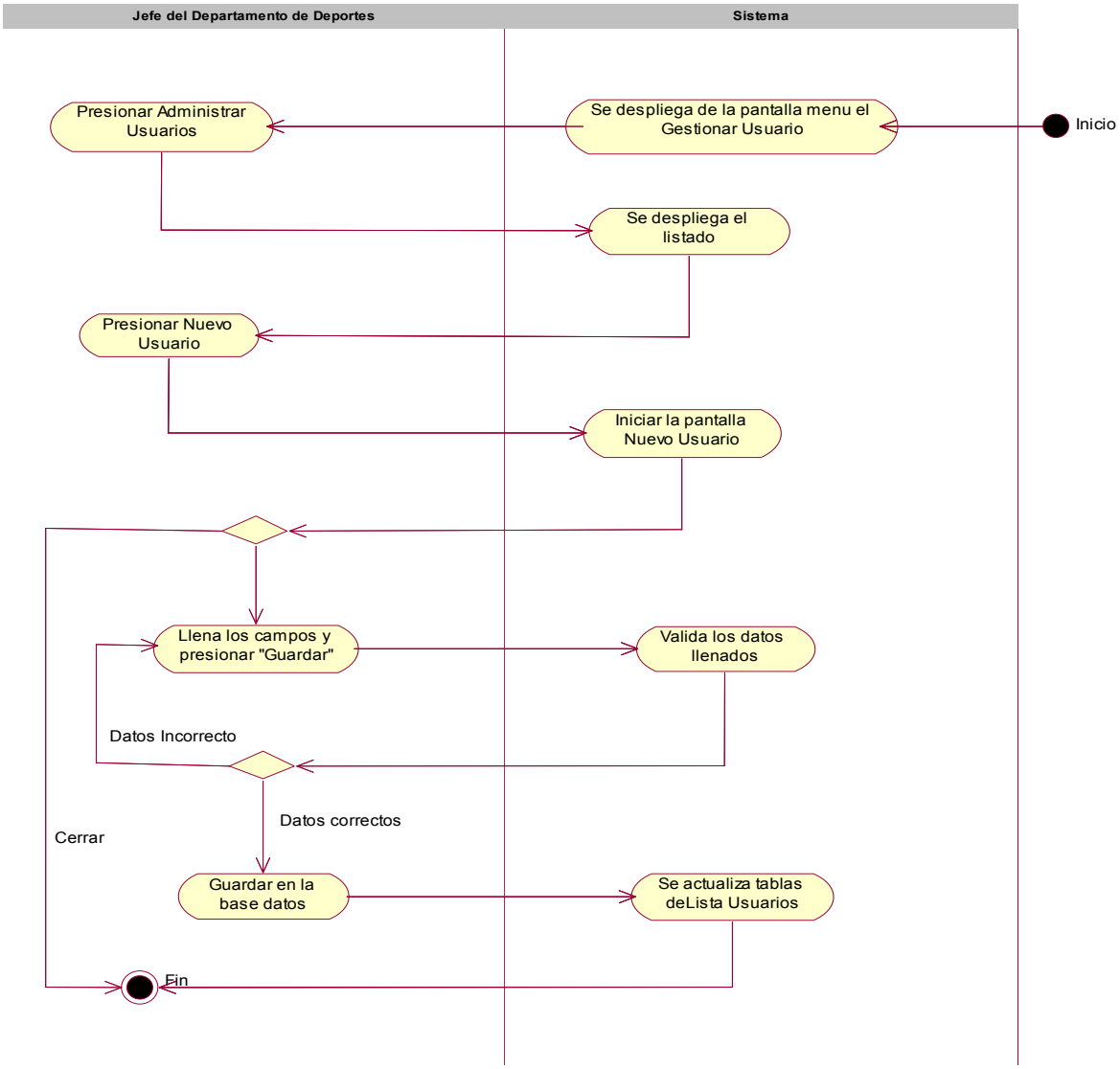


Figura 68: Diagrama de actividades: Nuevo Usuario

Modificar Usuario

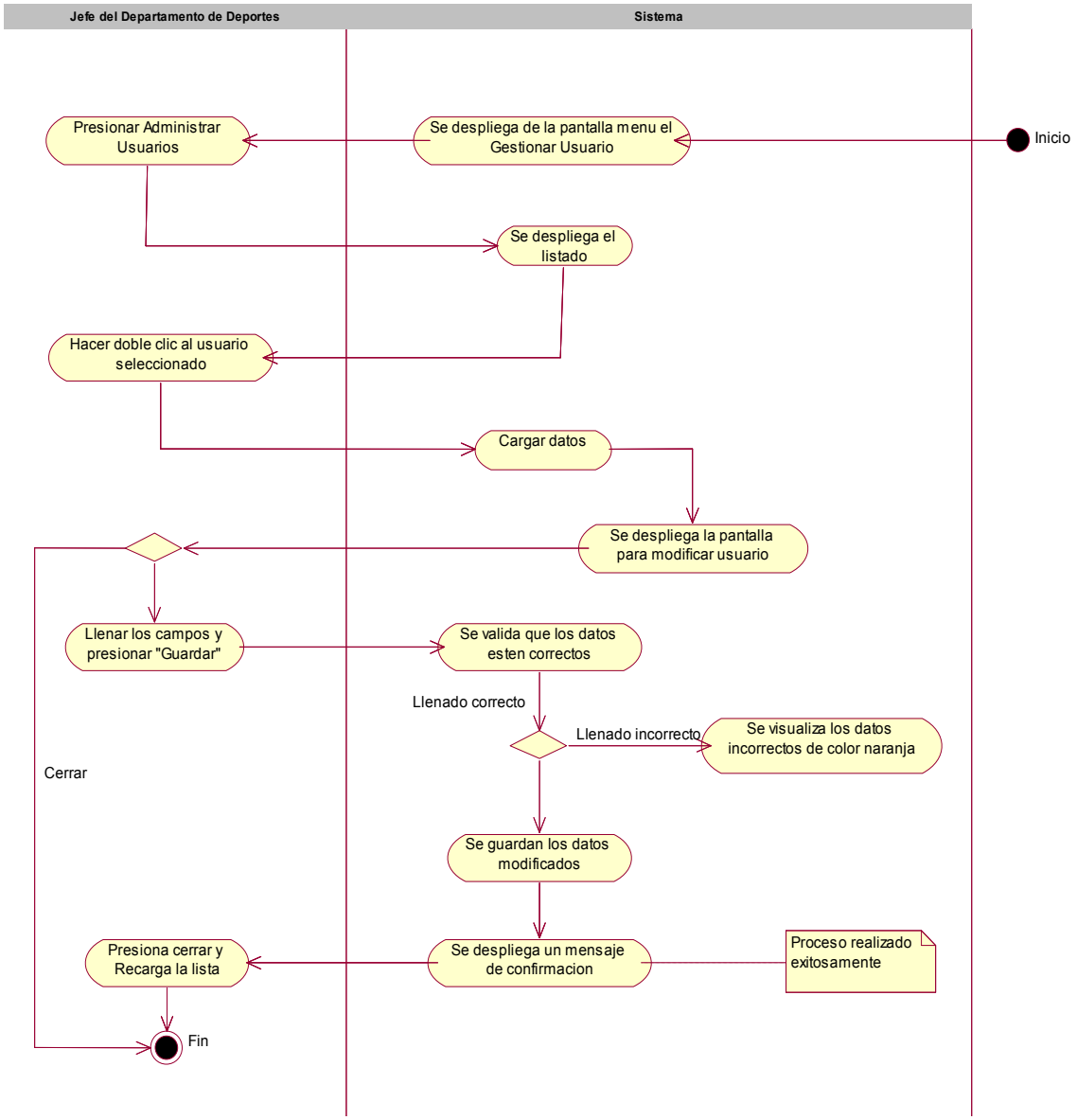


Figura 69: Diagrama de actividades: Modificar Usuario

Estado Usuario

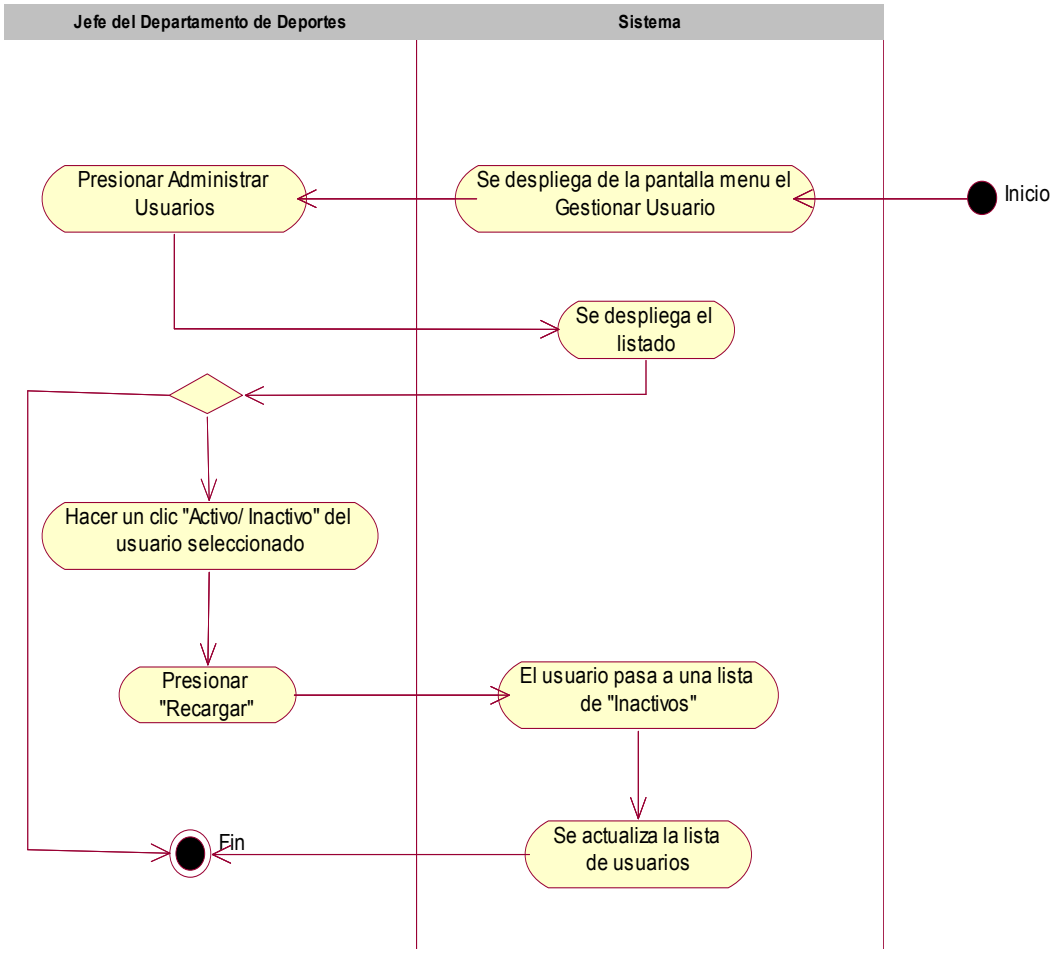


Figura 70: Diagrama de actividades: Estado Usuario

Asignar Login/Clave Al Usuario

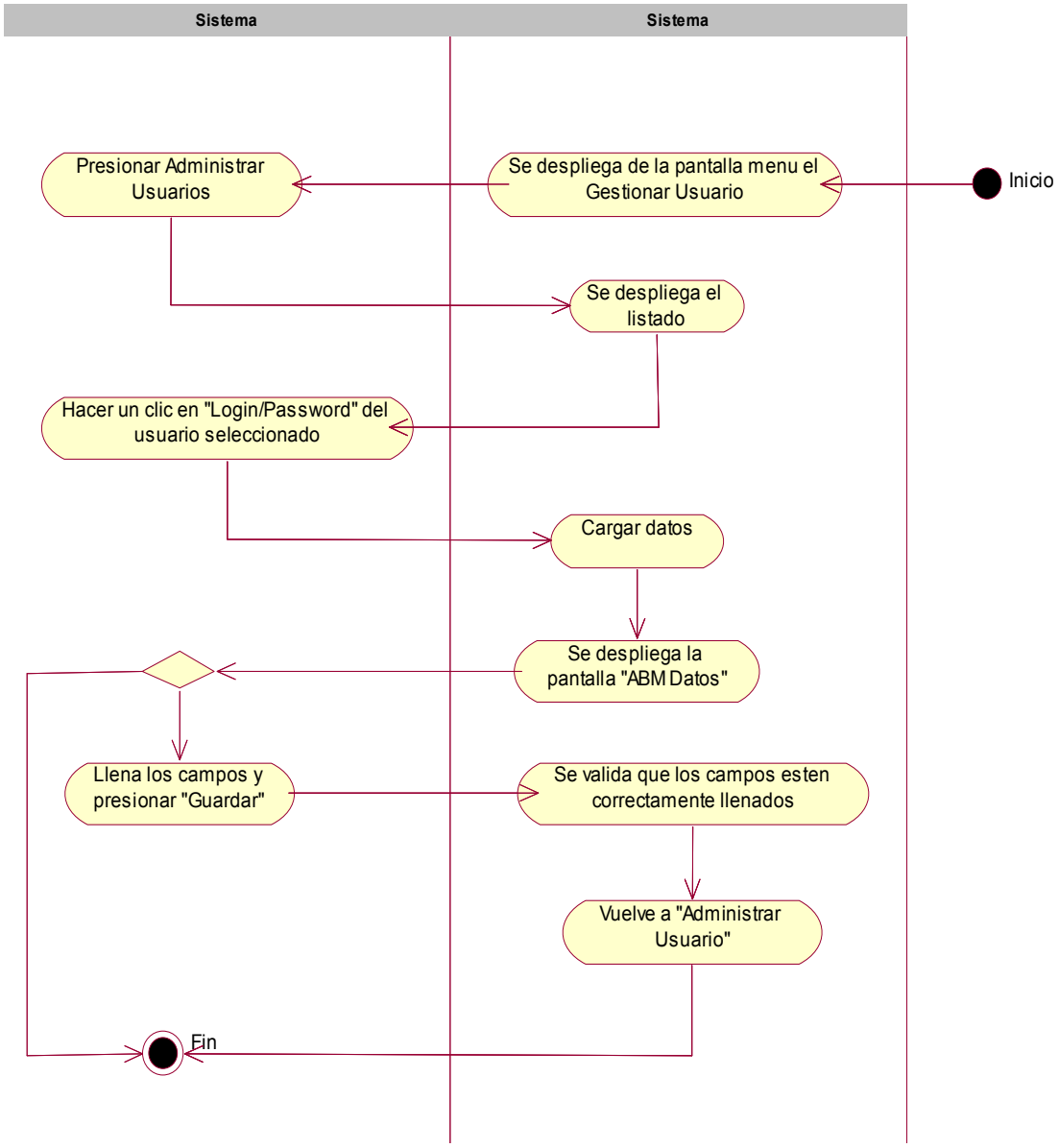


Figura 71: Diagrama de actividades Asignar Login/Clave Al Usuario

Ver Usuario

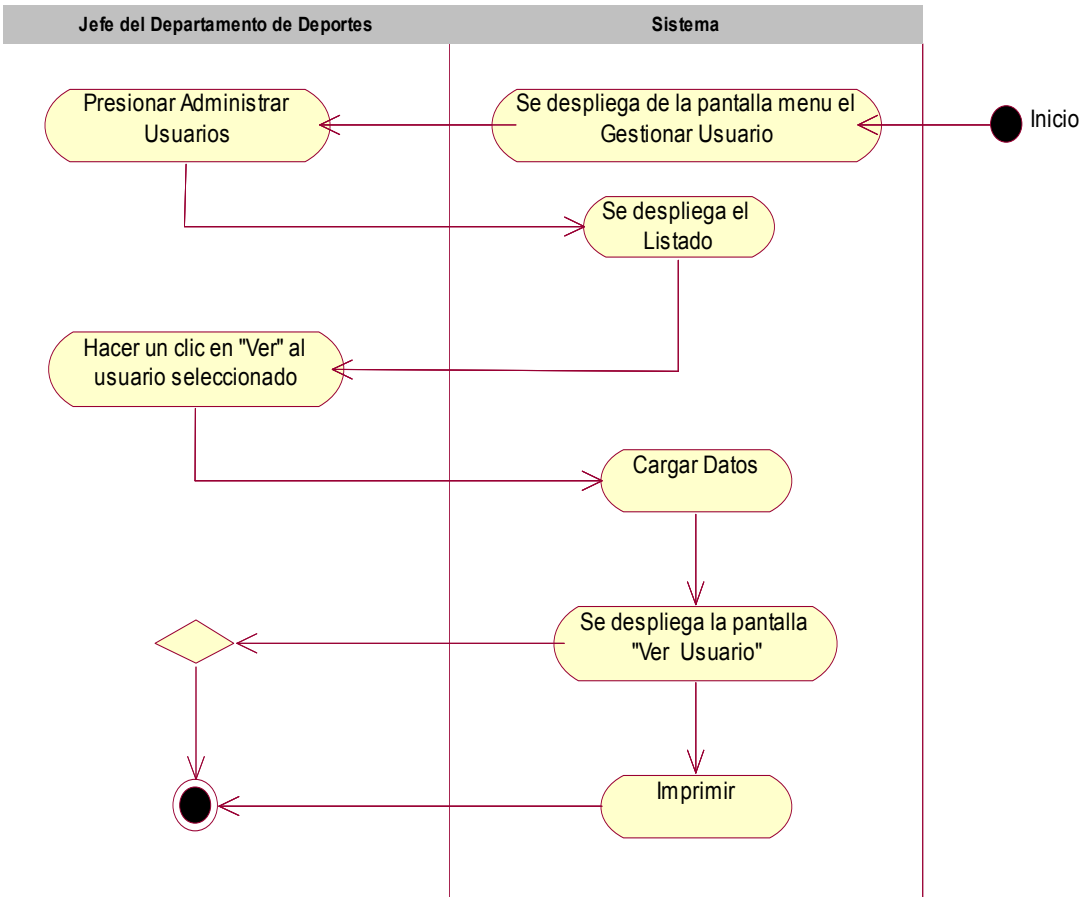


Figura 72: Diagrama de actividades Ver Usuario

Asignar Rol/ Usuario

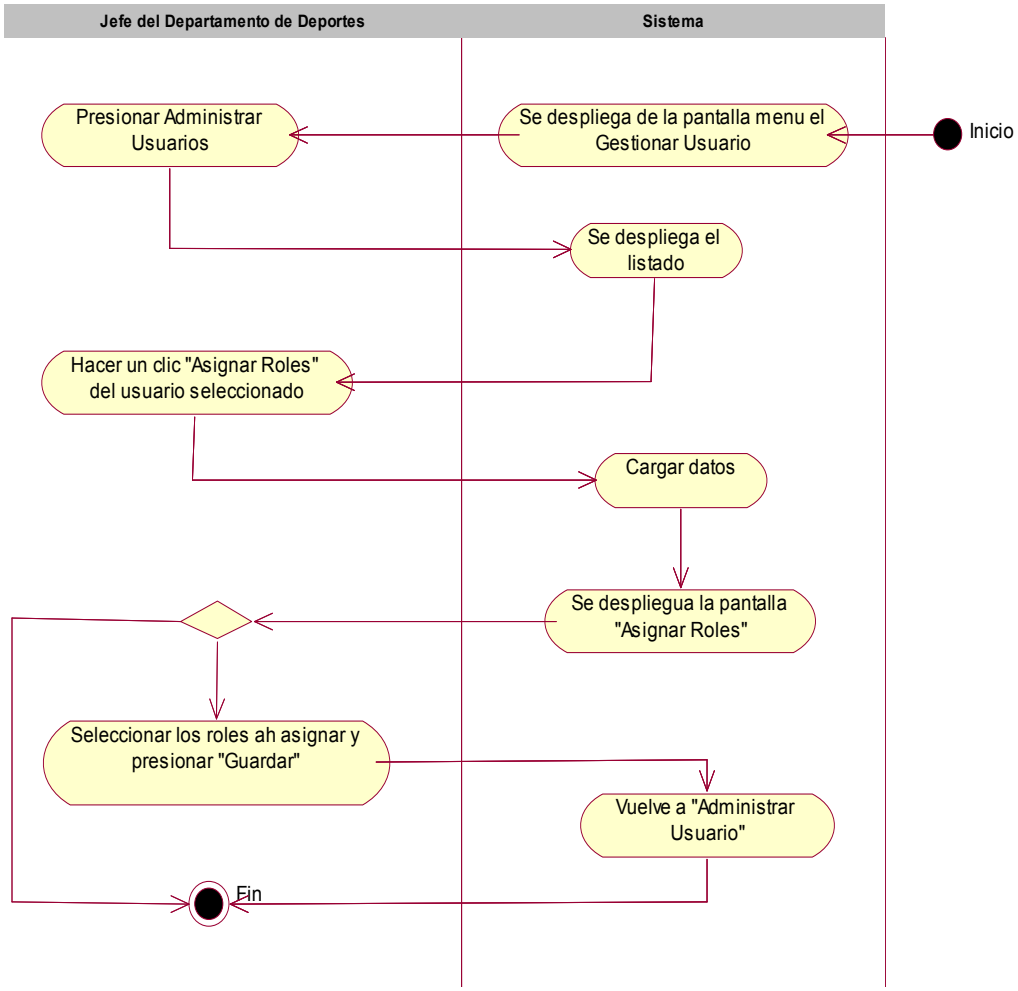


Figura 73: Diagrama de actividades Asignar Rol/ Usuario

Nuevo Profesión

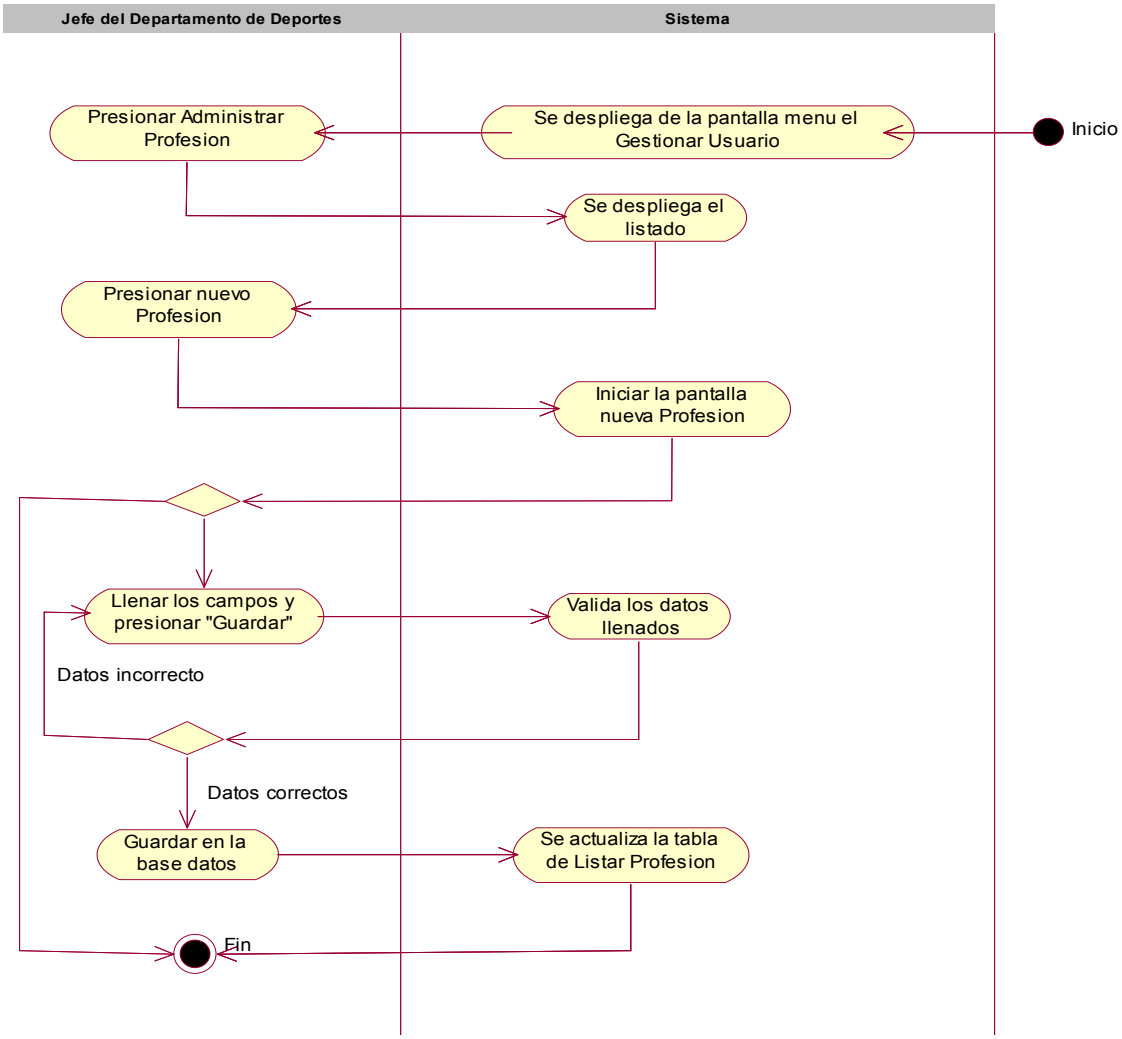


Figura 74: Diagrama de actividades Nuevo Profesión

Modificar Profesión

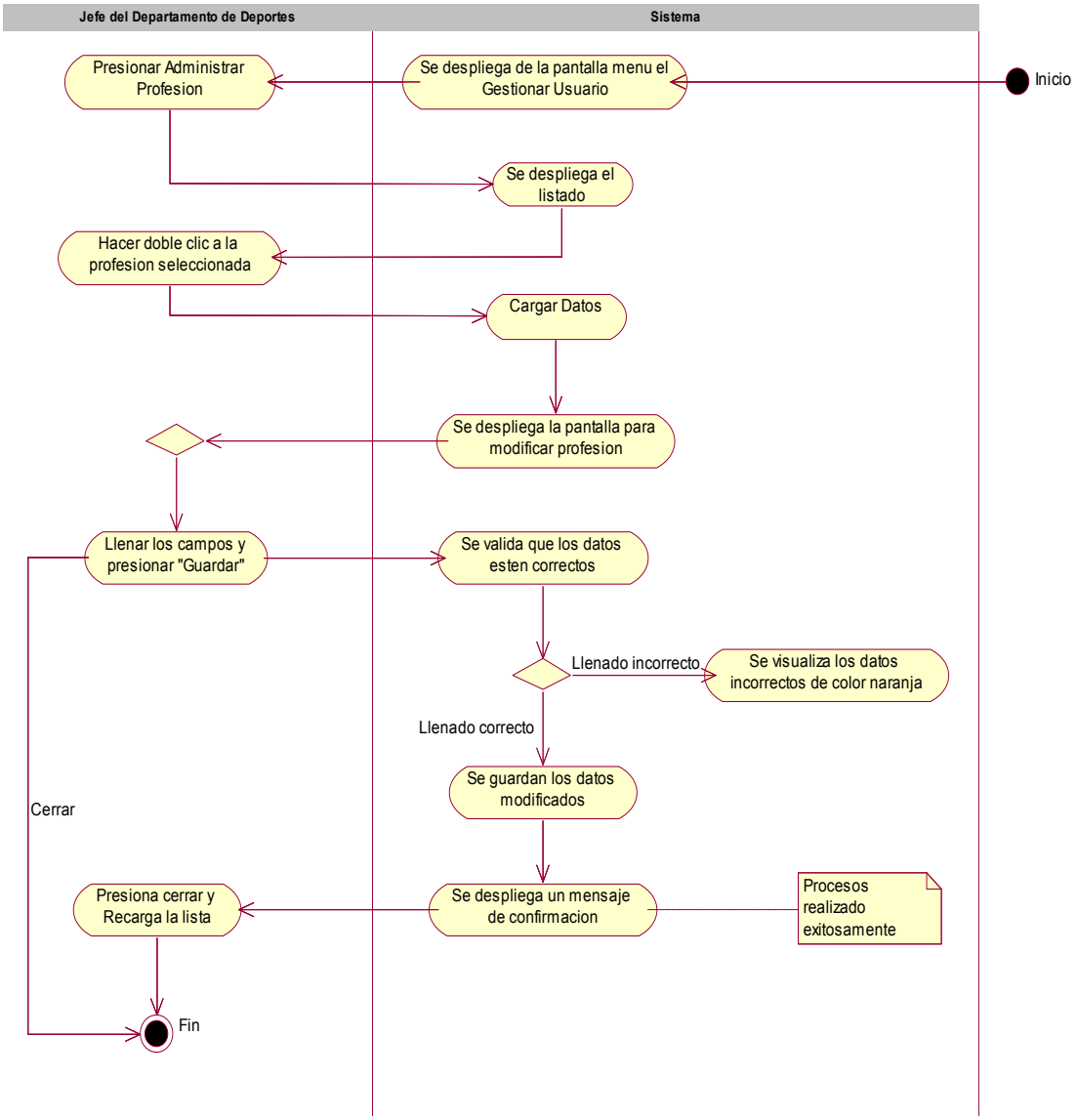


Figura 75: Diagrama de actividades Modificar Profesión

Nuevo Alumnos

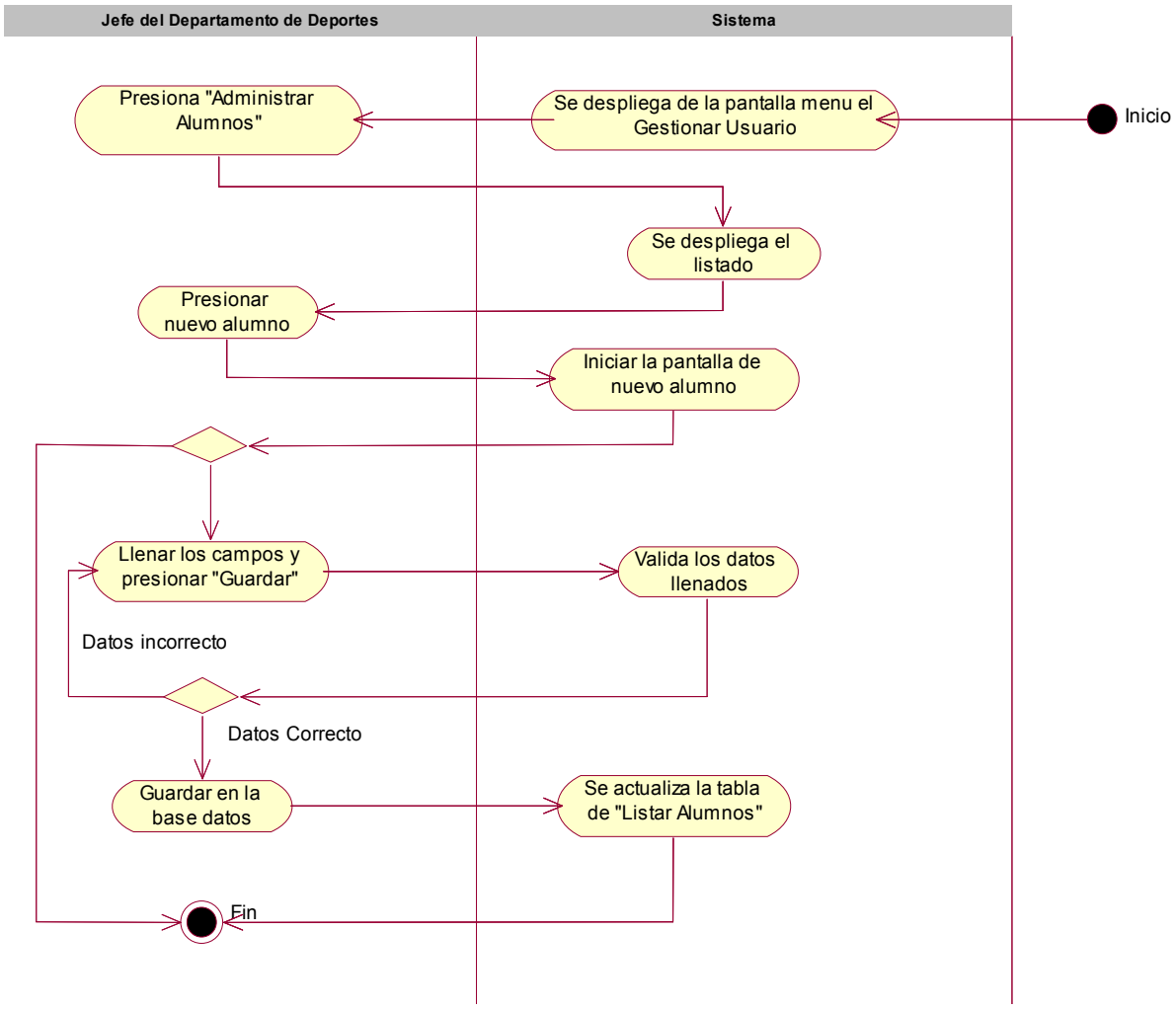


Figura 78: Diagrama de actividades Nuevo Alumnos

Modificar Alumnos

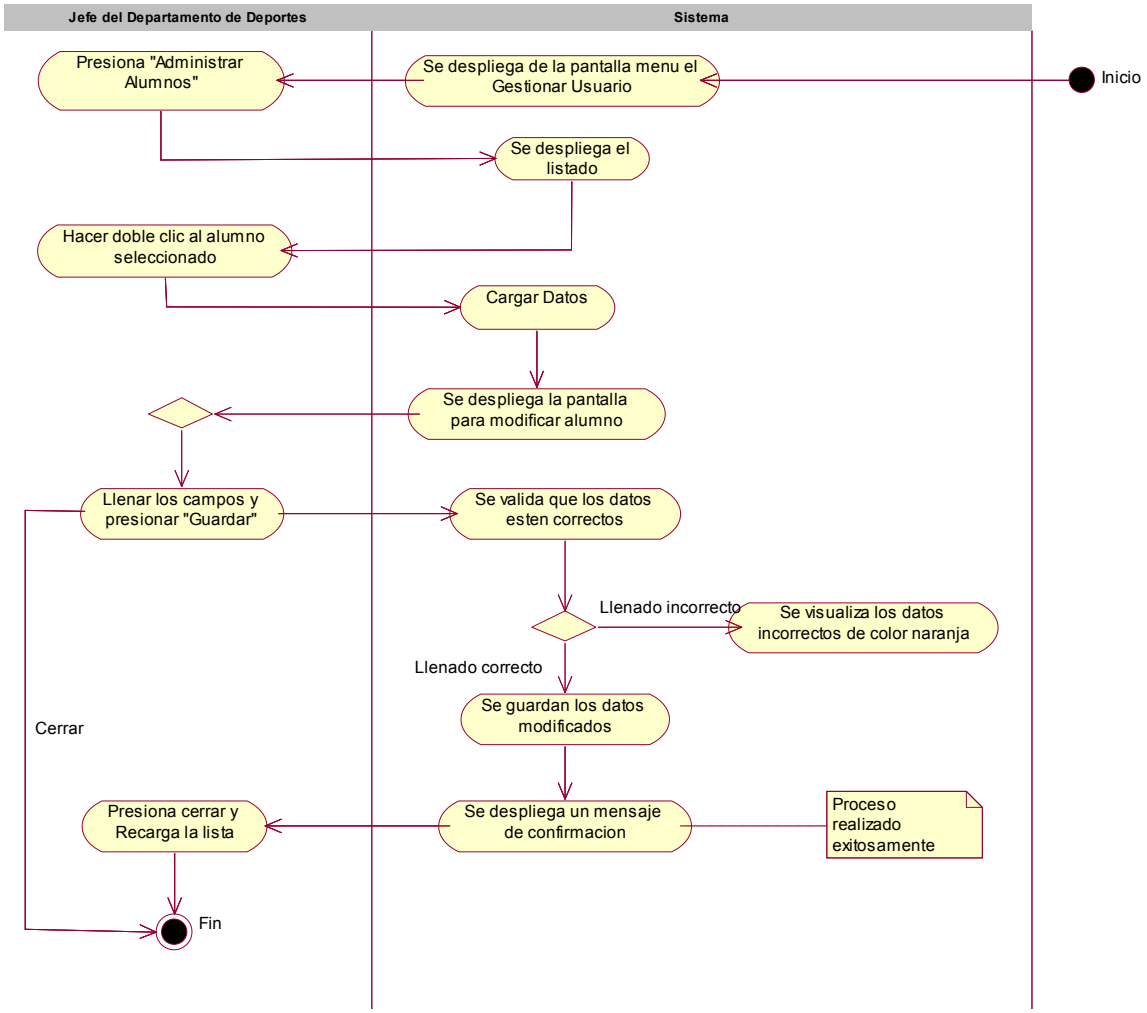


Figura 79: Diagrama de actividades Modificar Alumnos

Estado Alumno

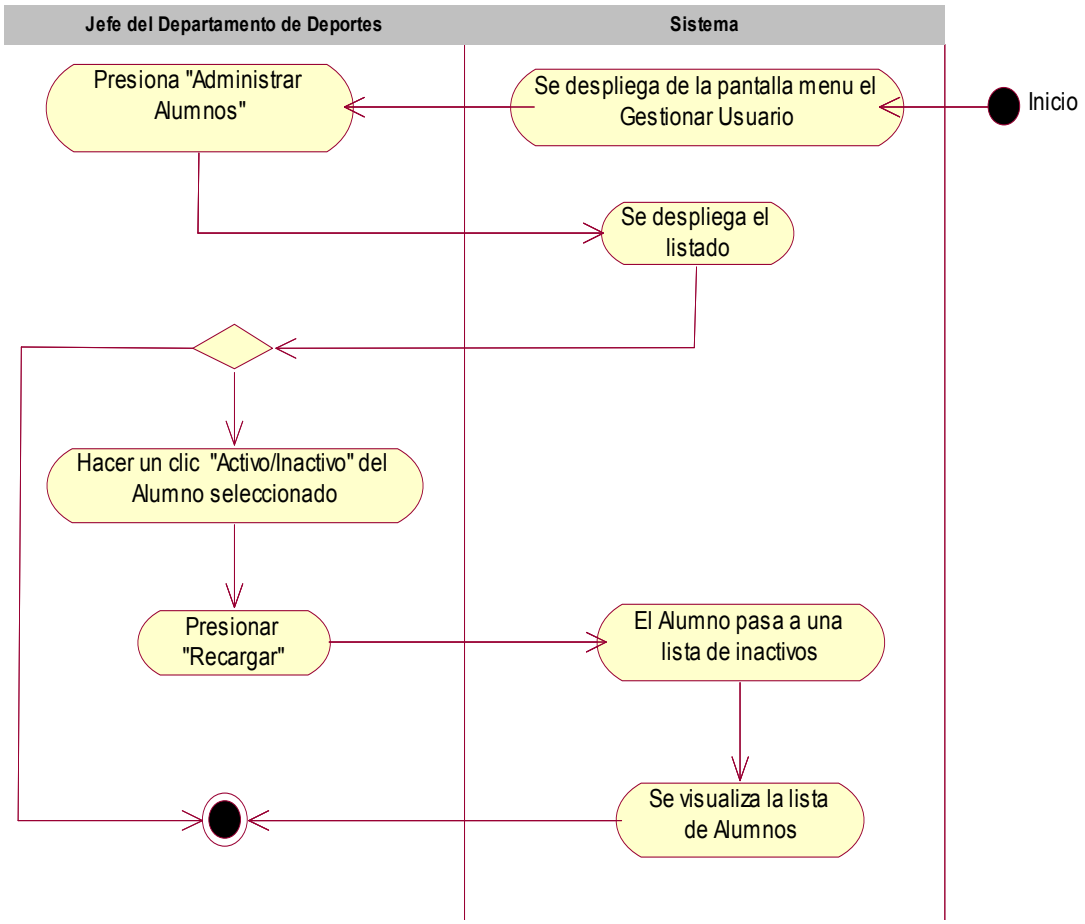


Figura 80: Diagrama de actividades Estado Alumno

Nuevo Ciudad

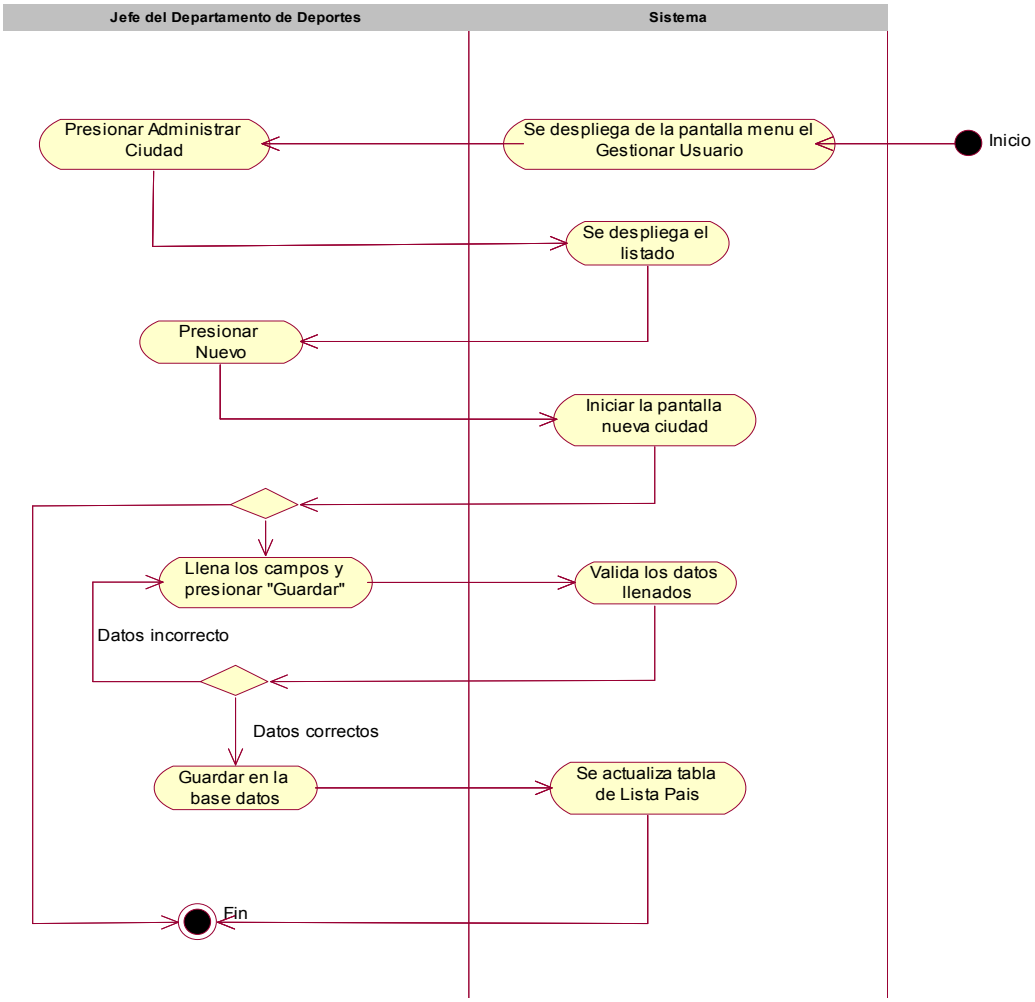


Figura 83: Diagrama de actividades Nuevo Ciudad

Modificar Ciudad

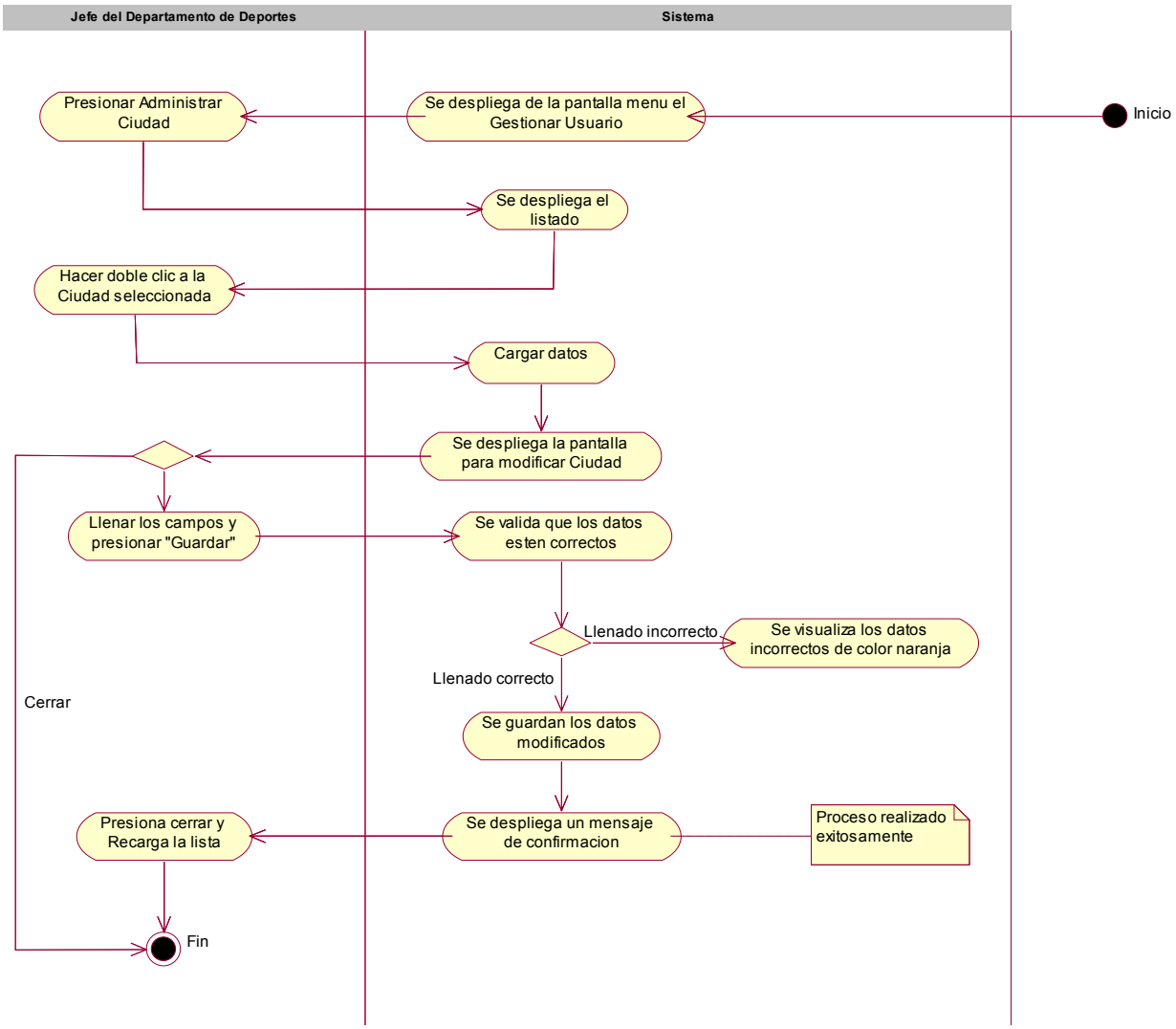


Figura 84: Diagrama de actividades Modificar Ciudad

Nuevo Departamento

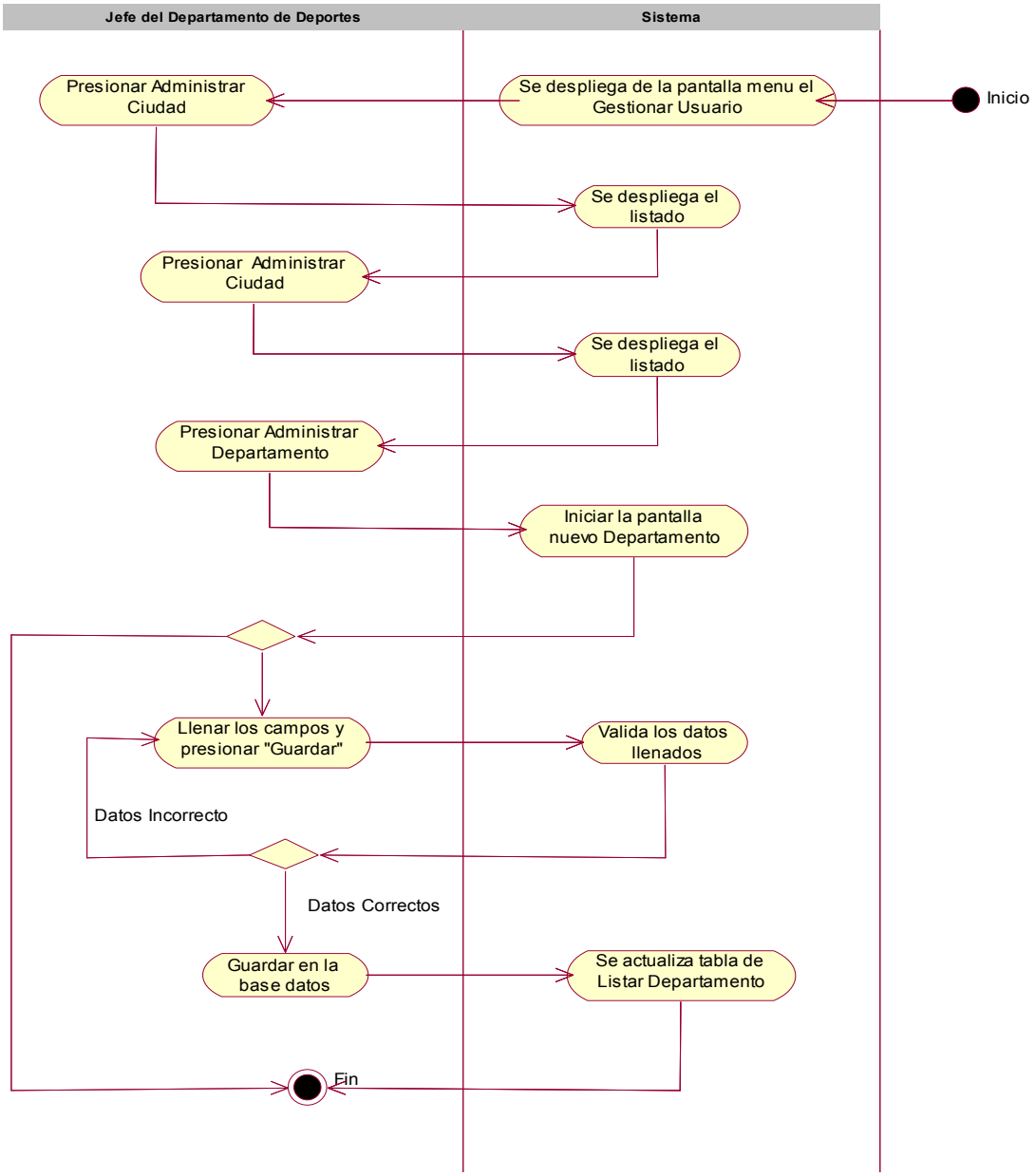


Figura 85: Diagrama de actividades Nuevo Departamento

Modificar Departamento

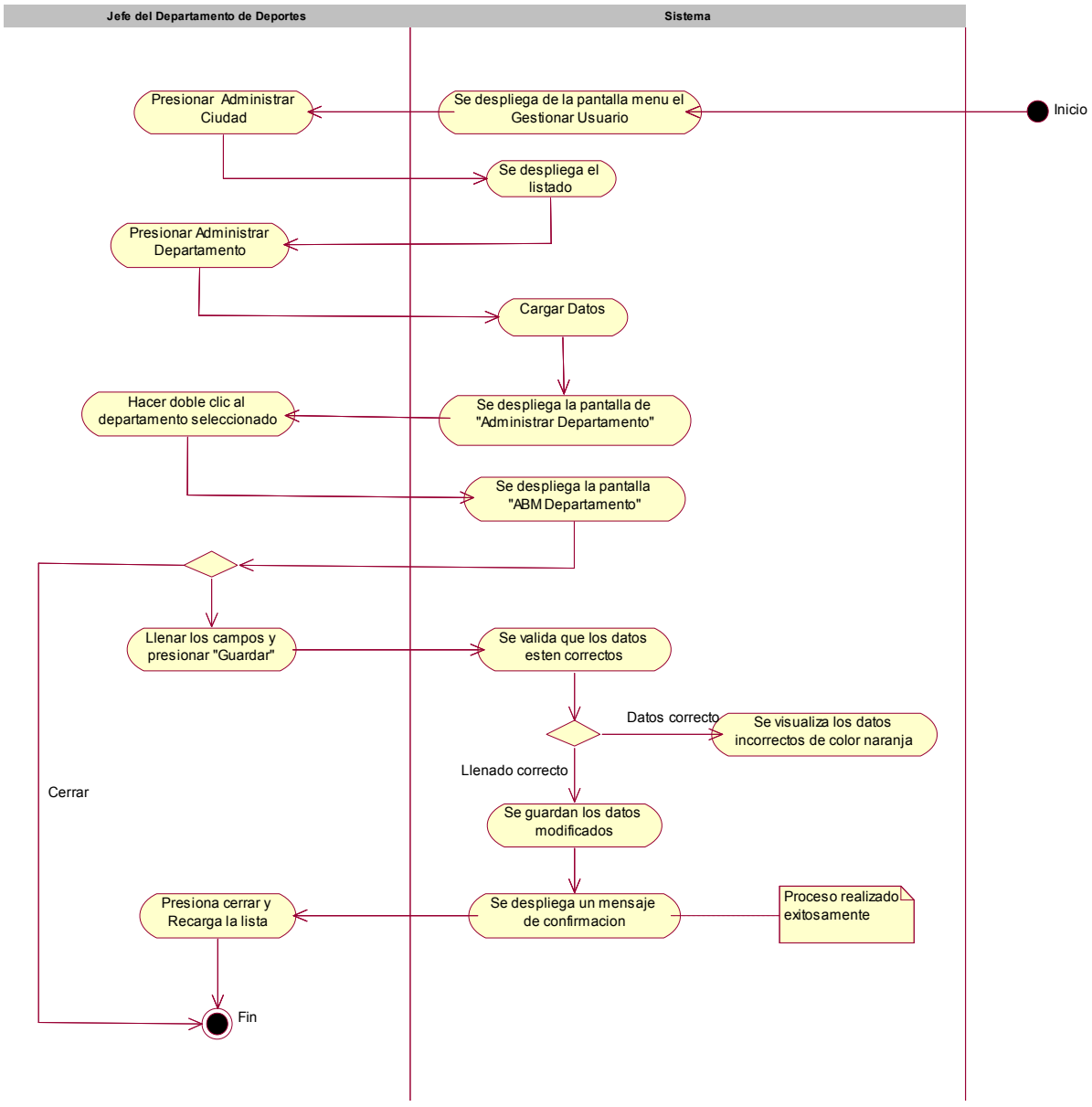


Figura 86: Diagrama de actividades Modificar Departamento

Nuevo Provincia

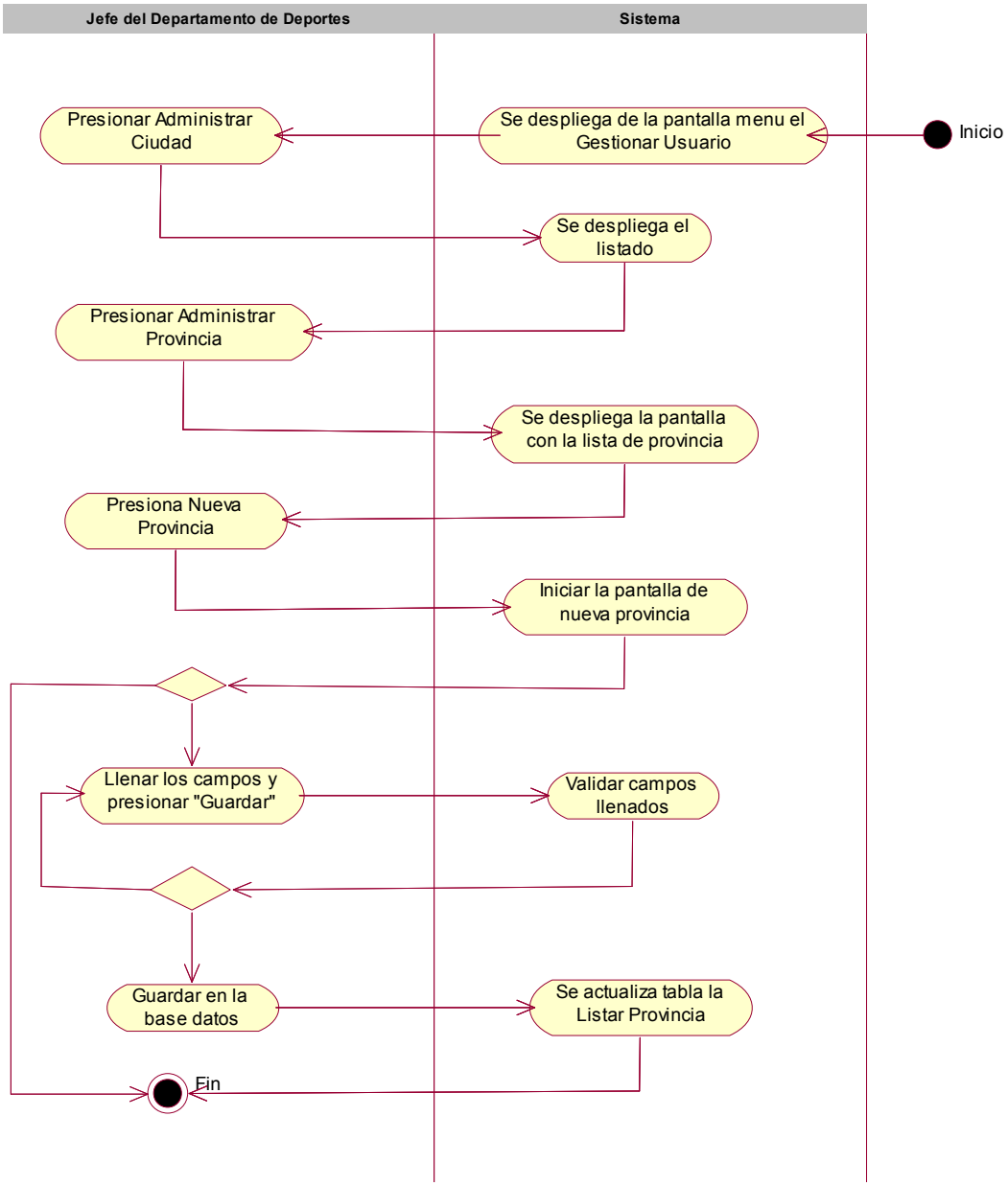


Figura 87: Diagrama de actividades Nuevo Provincia

Modificar Provincia

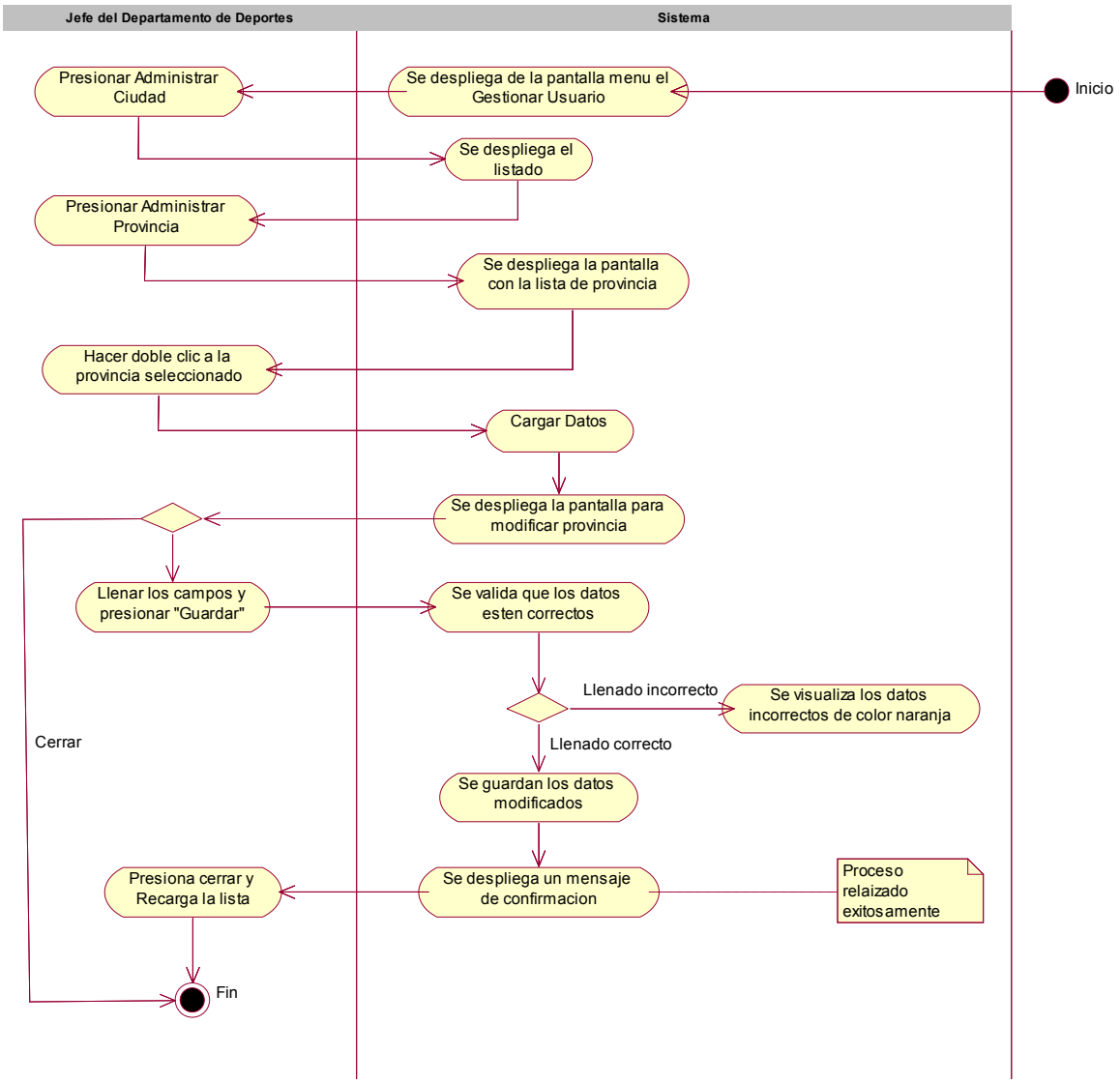


Figura 88: Diagrama de actividades Modificar Provincia

Nuevo Disciplina

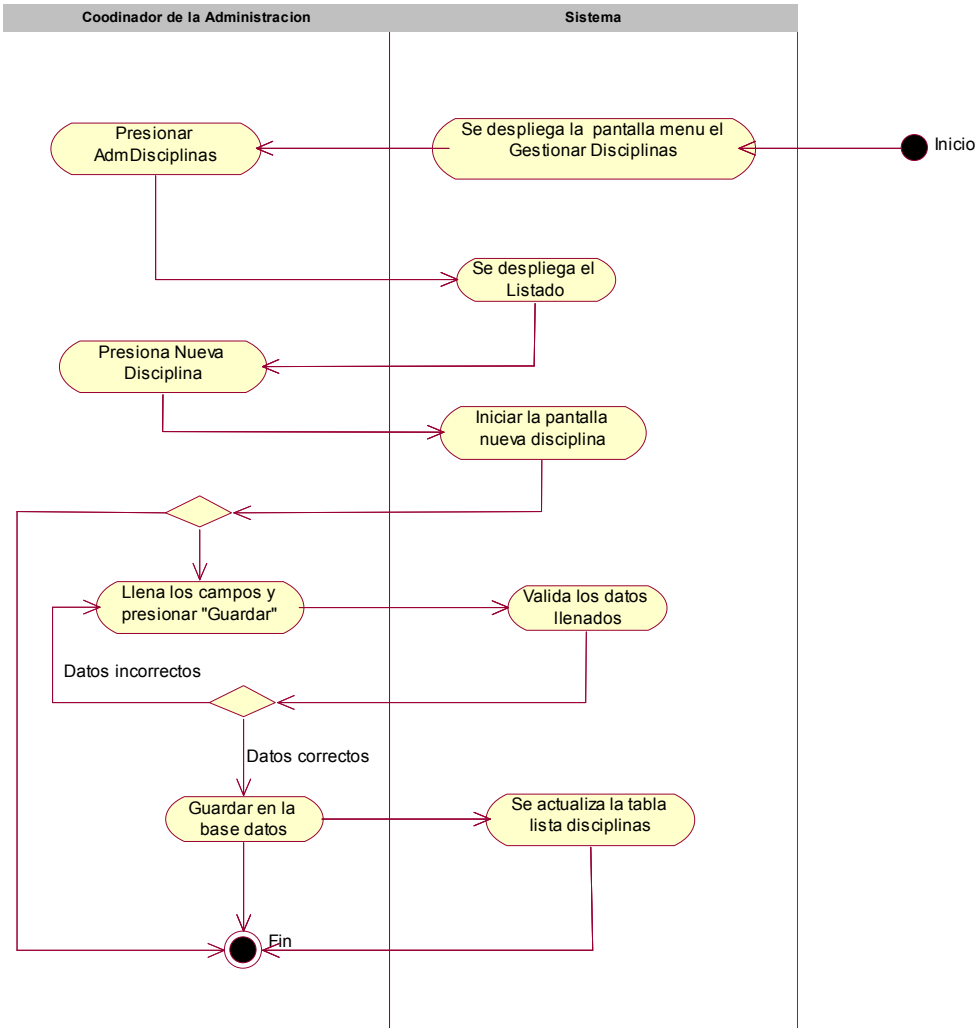


Figura 89: Diagrama de actividades Nuevo Disciplina

Modificar Disciplina

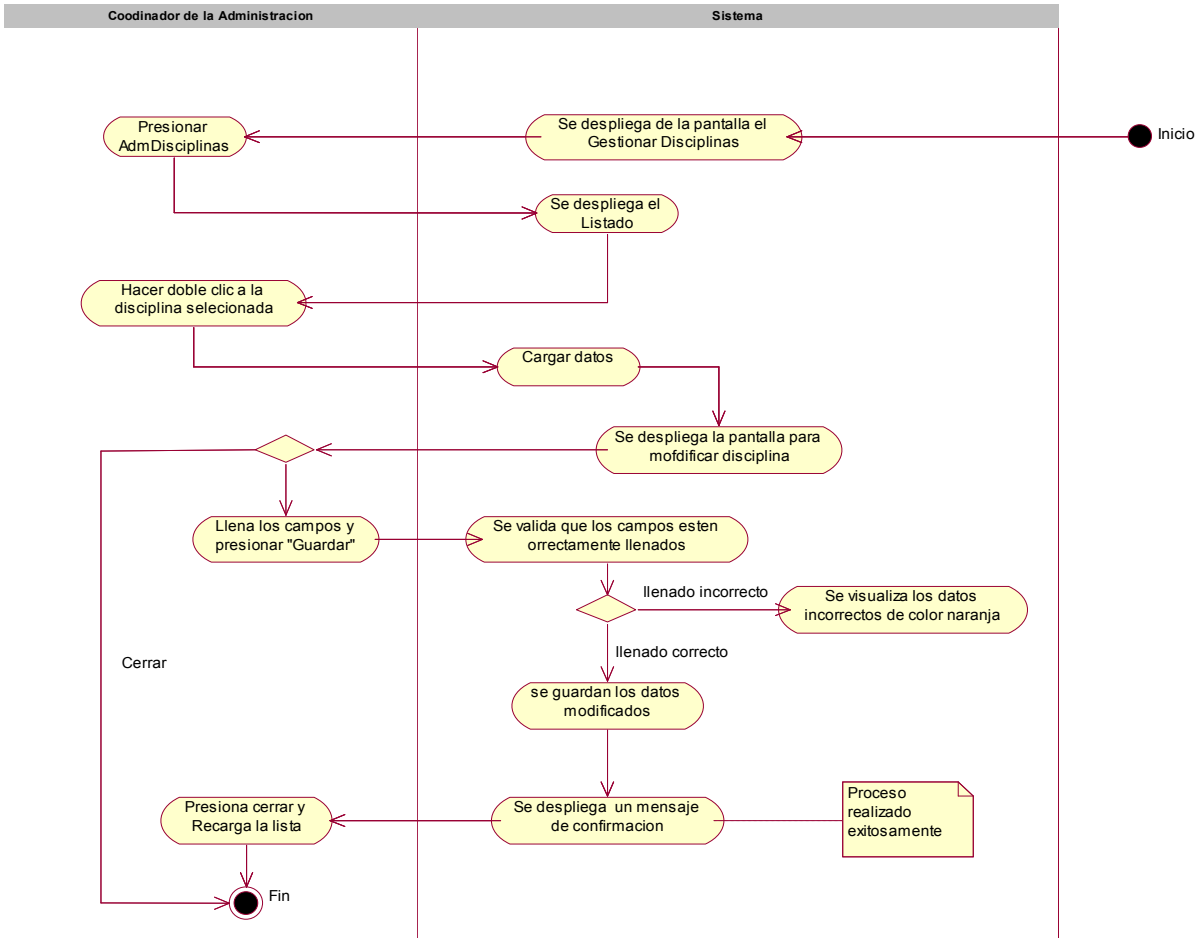


Figura 90: Diagrama de actividades Modificar Disciplina

Estado Disciplina

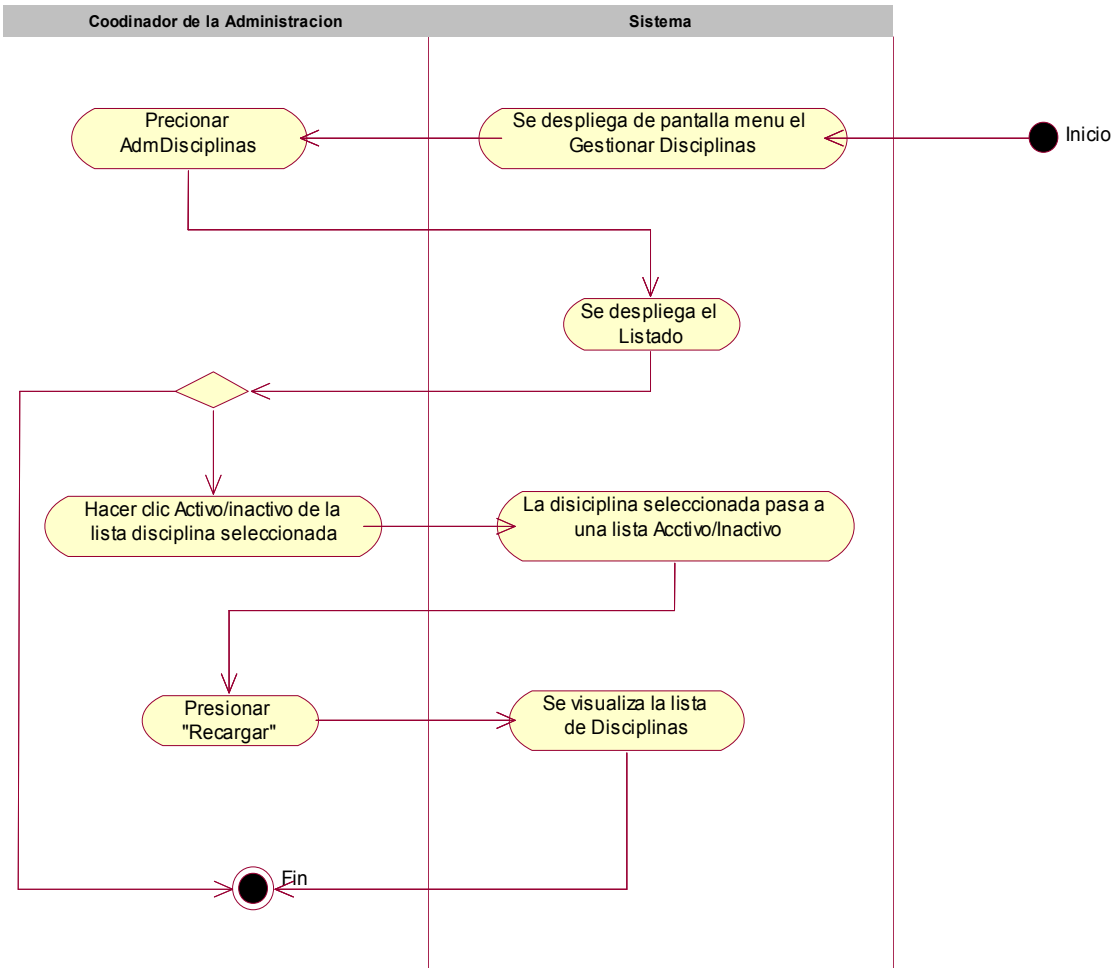


Figura 91: Diagrama de actividades Estado Disciplina

Nuevo Barrio

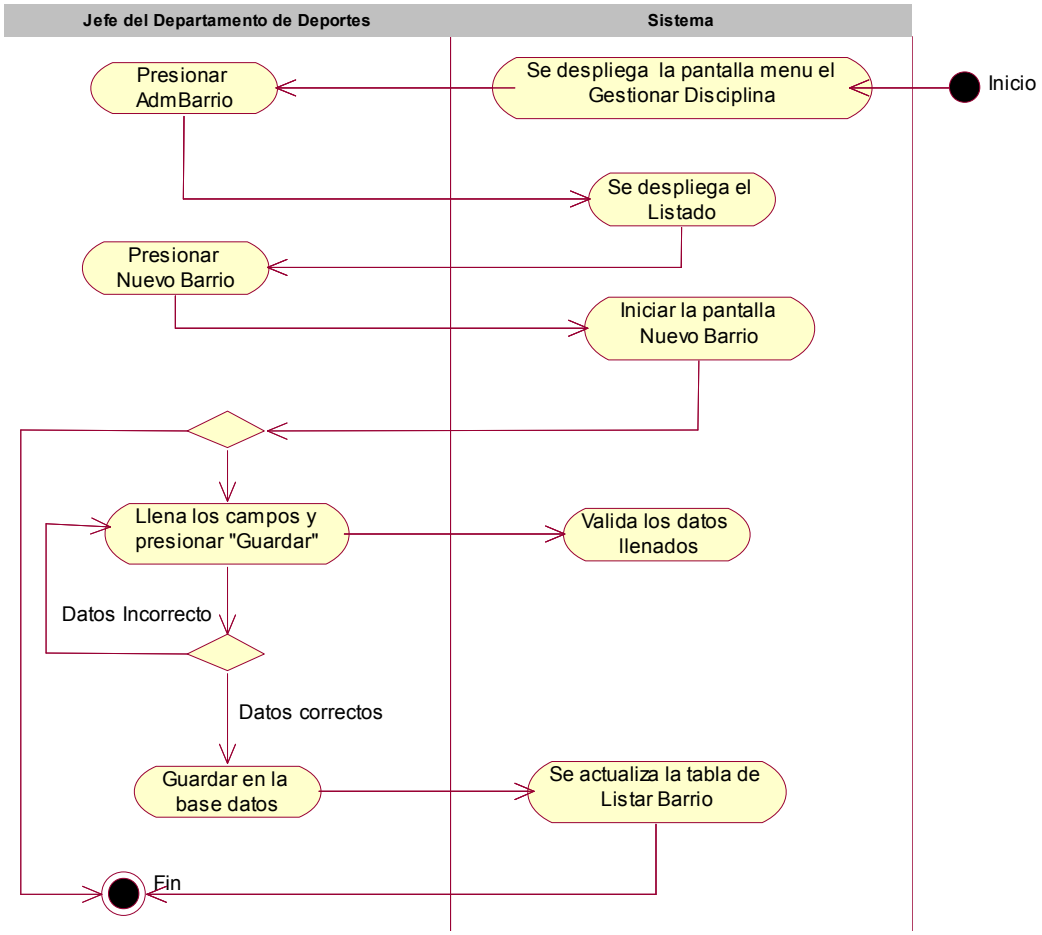


Figura 92: Diagrama de actividades Nuevo Barrio

Modificar Barrio

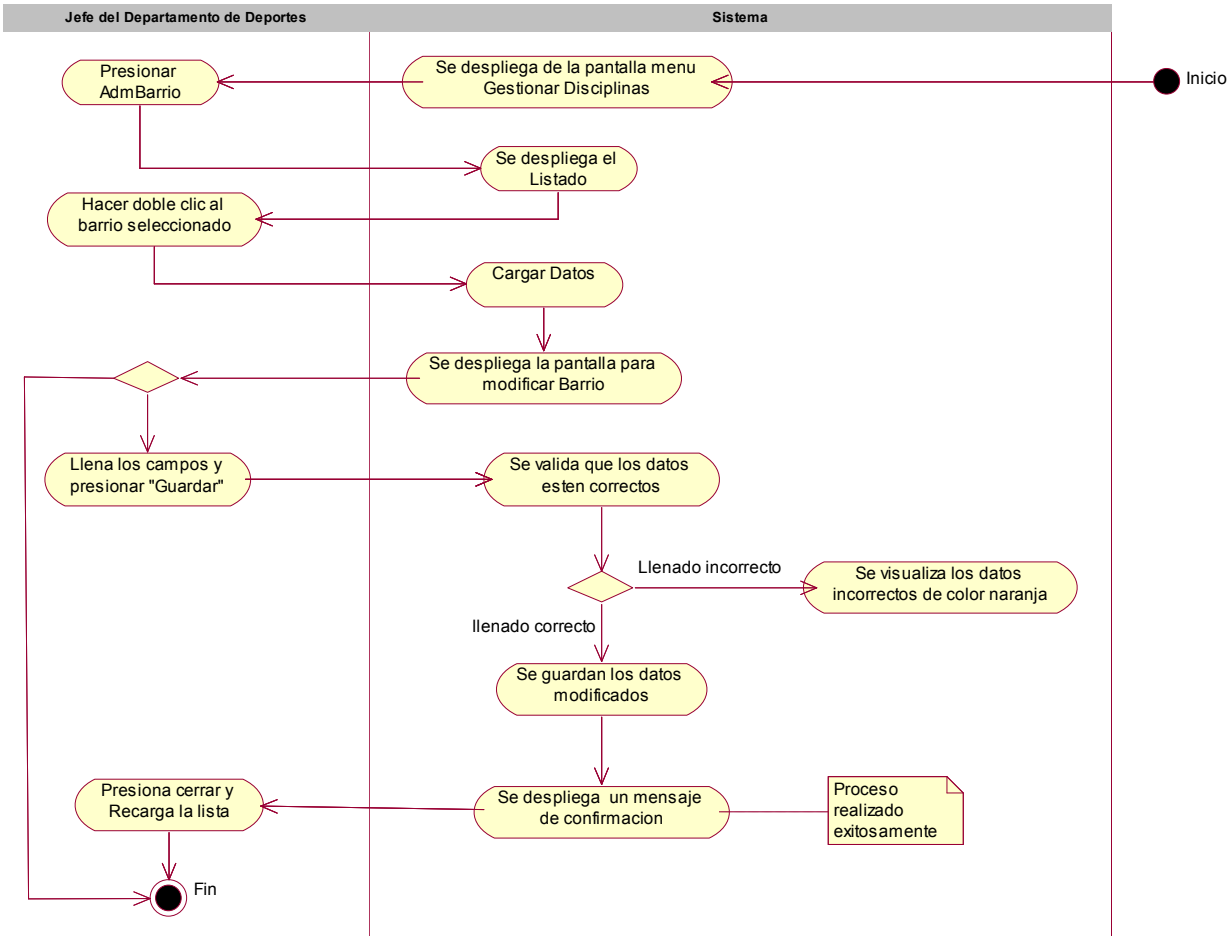


Figura 93: Diagrama de actividades Modificar Barrio

Estado Barrio

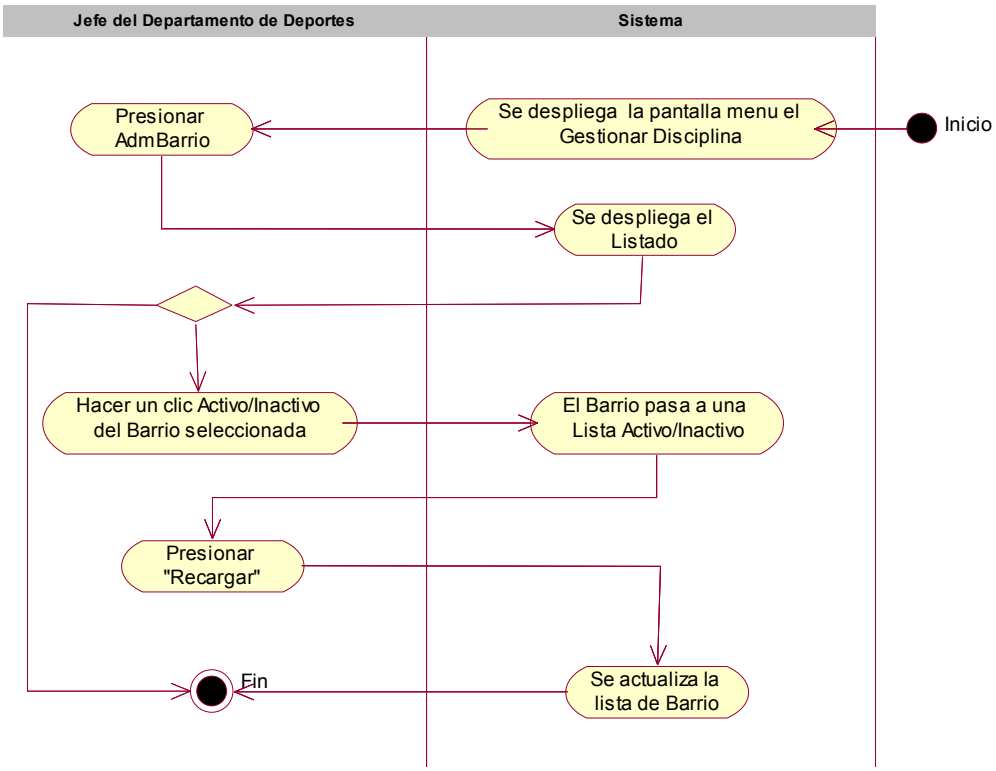


Figura 94: Diagrama de actividades Estado Barrio

Nuevo Categoría

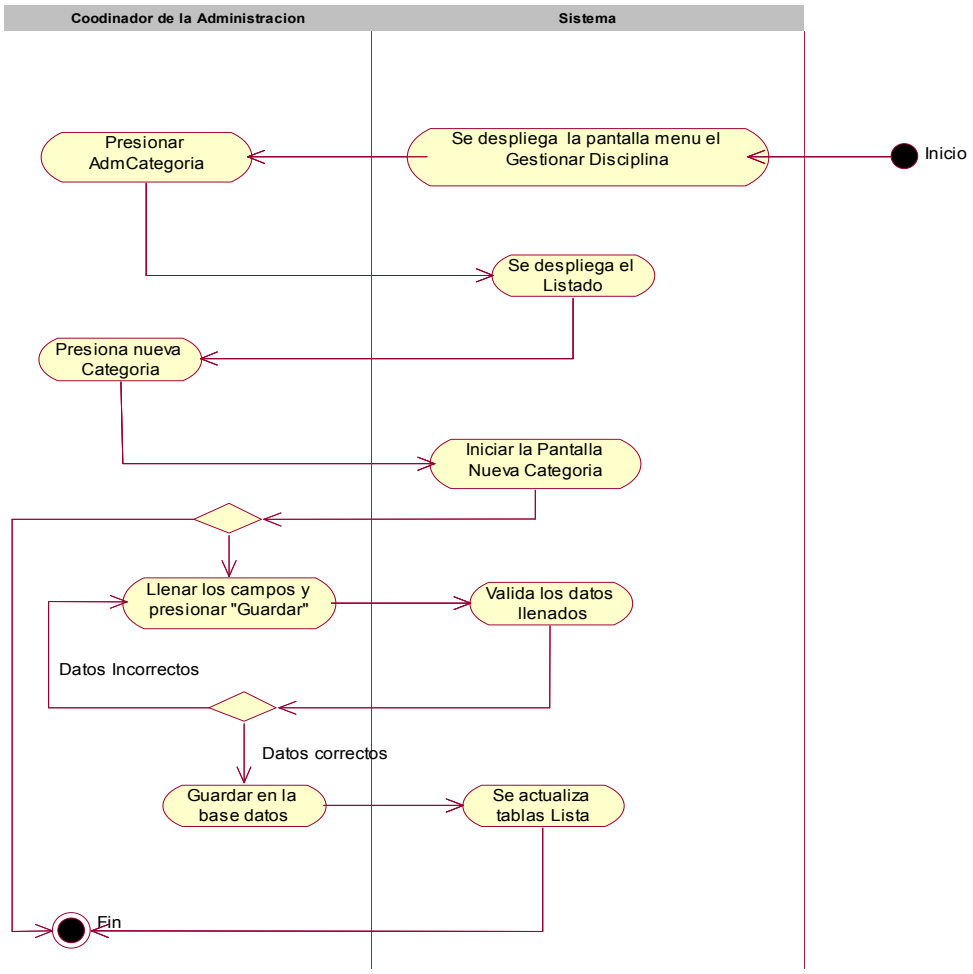


Figura 95: Diagrama de actividades Nuevo Categoría

Modificar Categoría

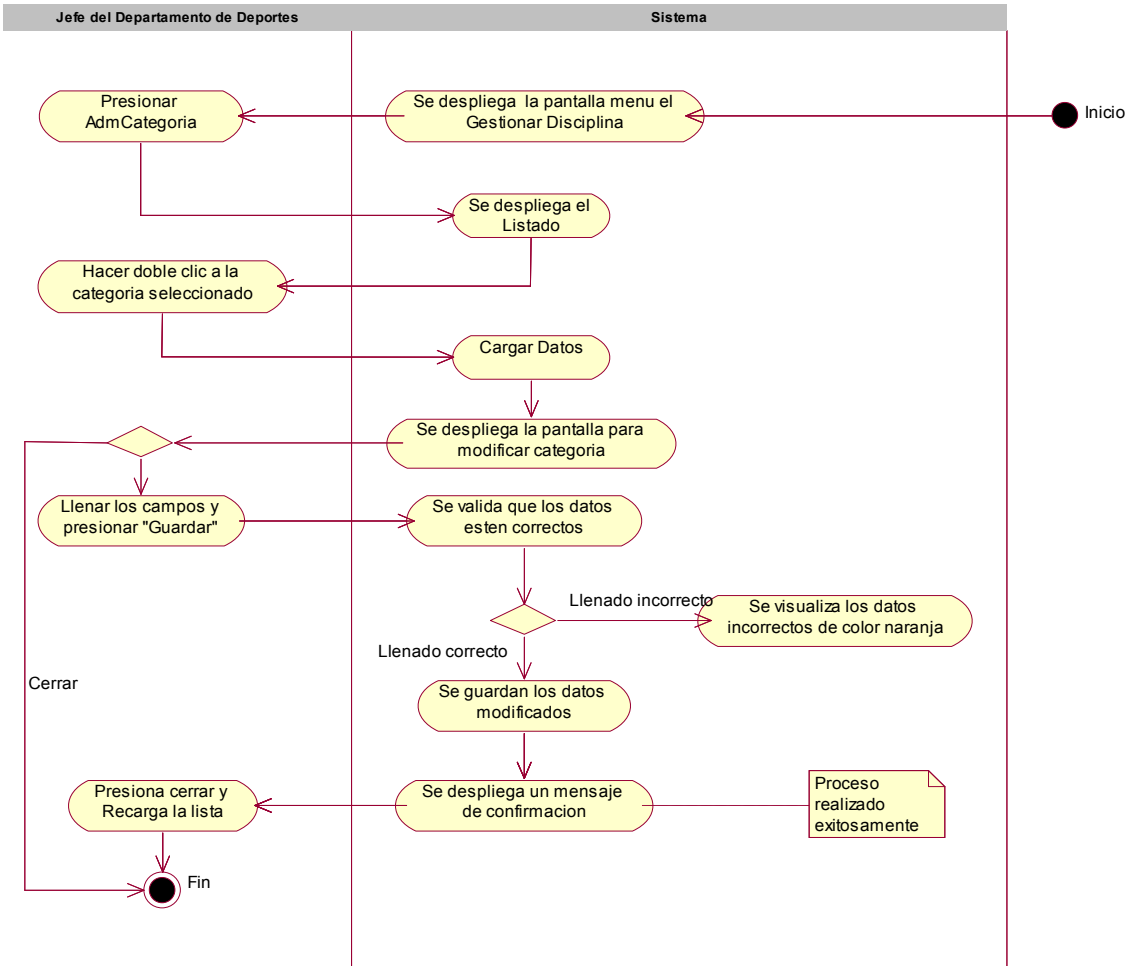


Figura 96: Diagrama de actividades Modificar Categoría

Estado Categoría

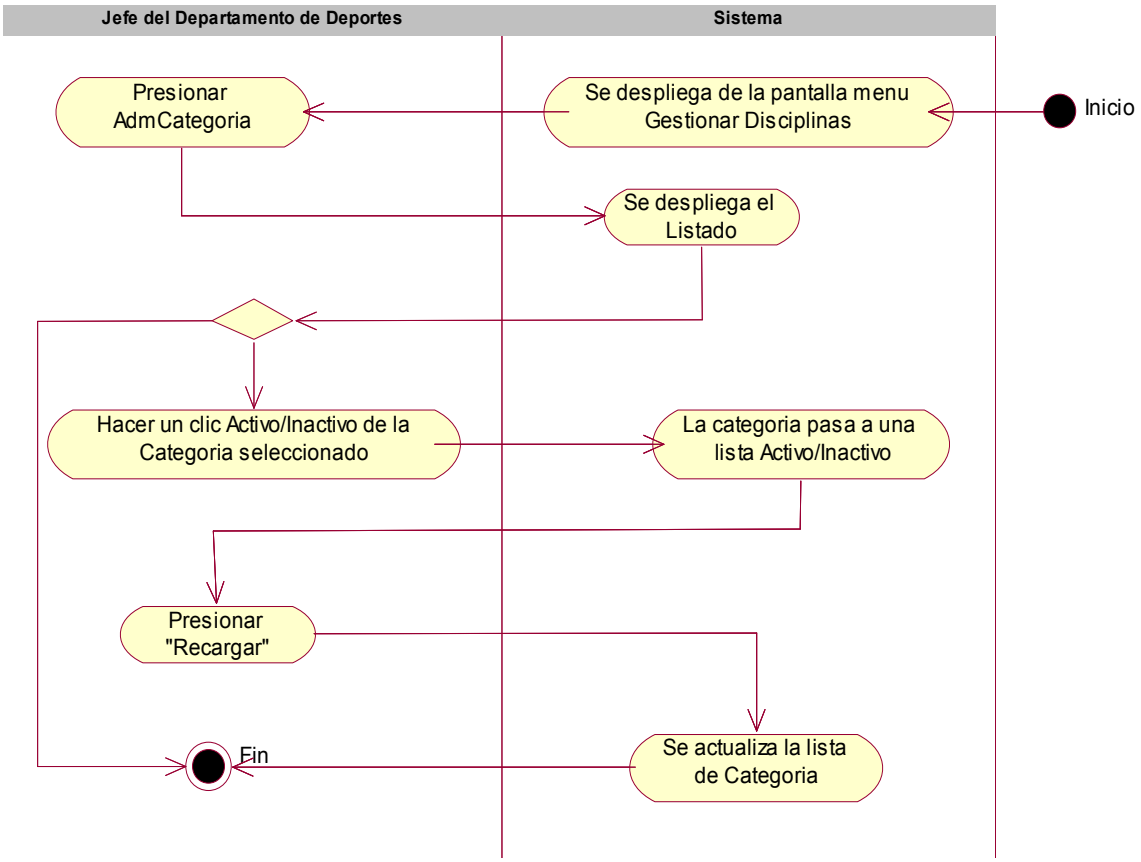


Figura 97: Diagrama de actividades Estado Categoría

Nuevo Equipo

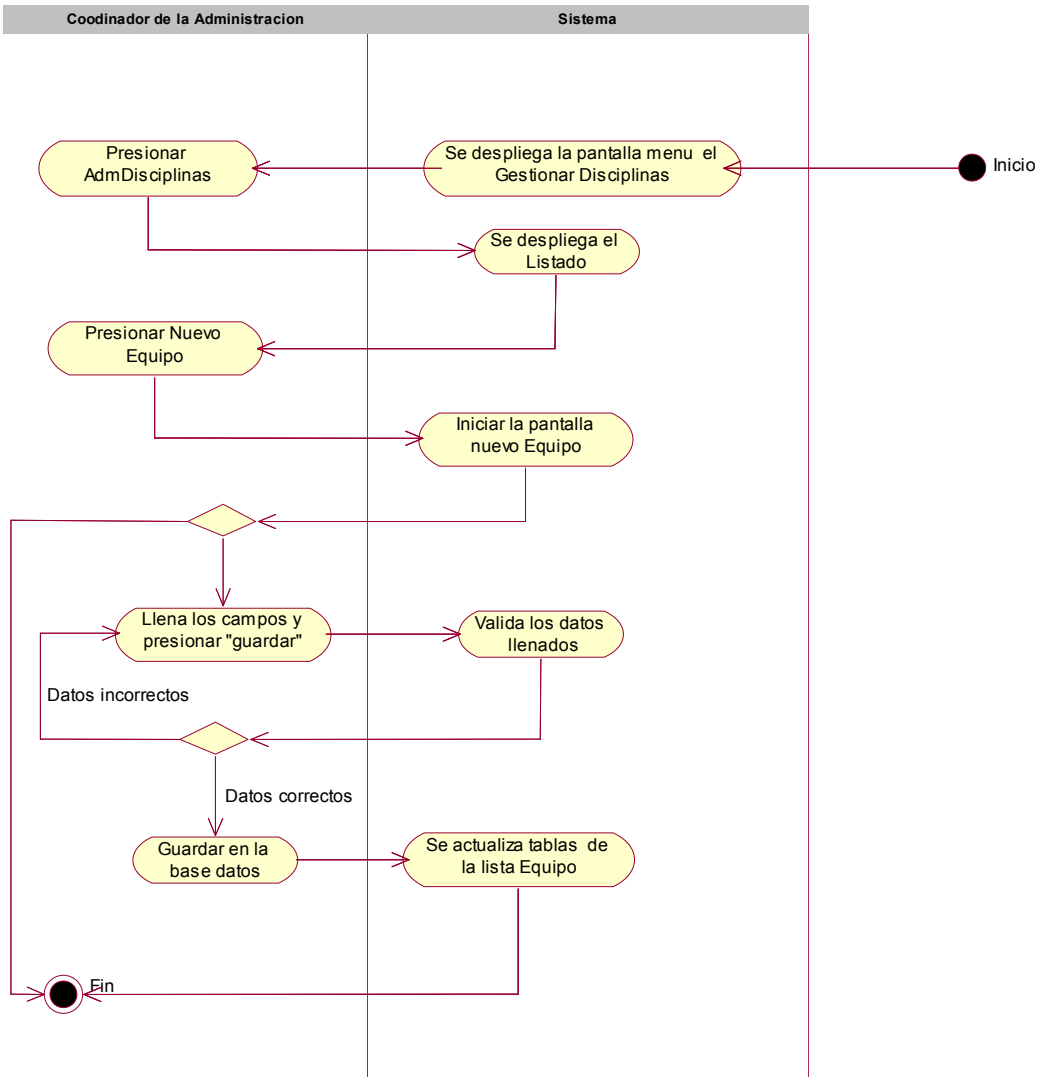


Figura 98: Diagrama de actividades Nuevo Equipo

Modificar Equipo

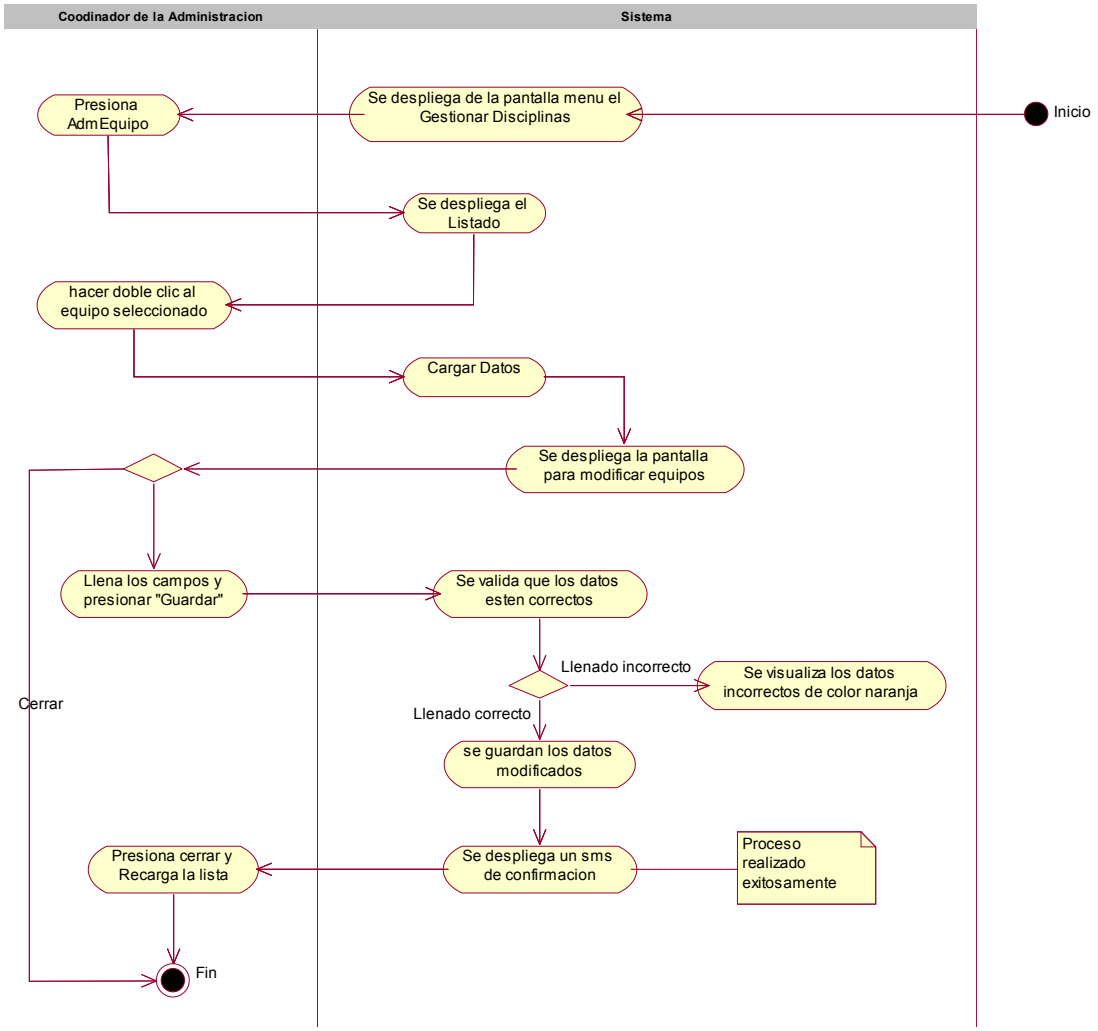


Figura 99: Diagrama de actividades Modificar Equipo

Estado Equipo

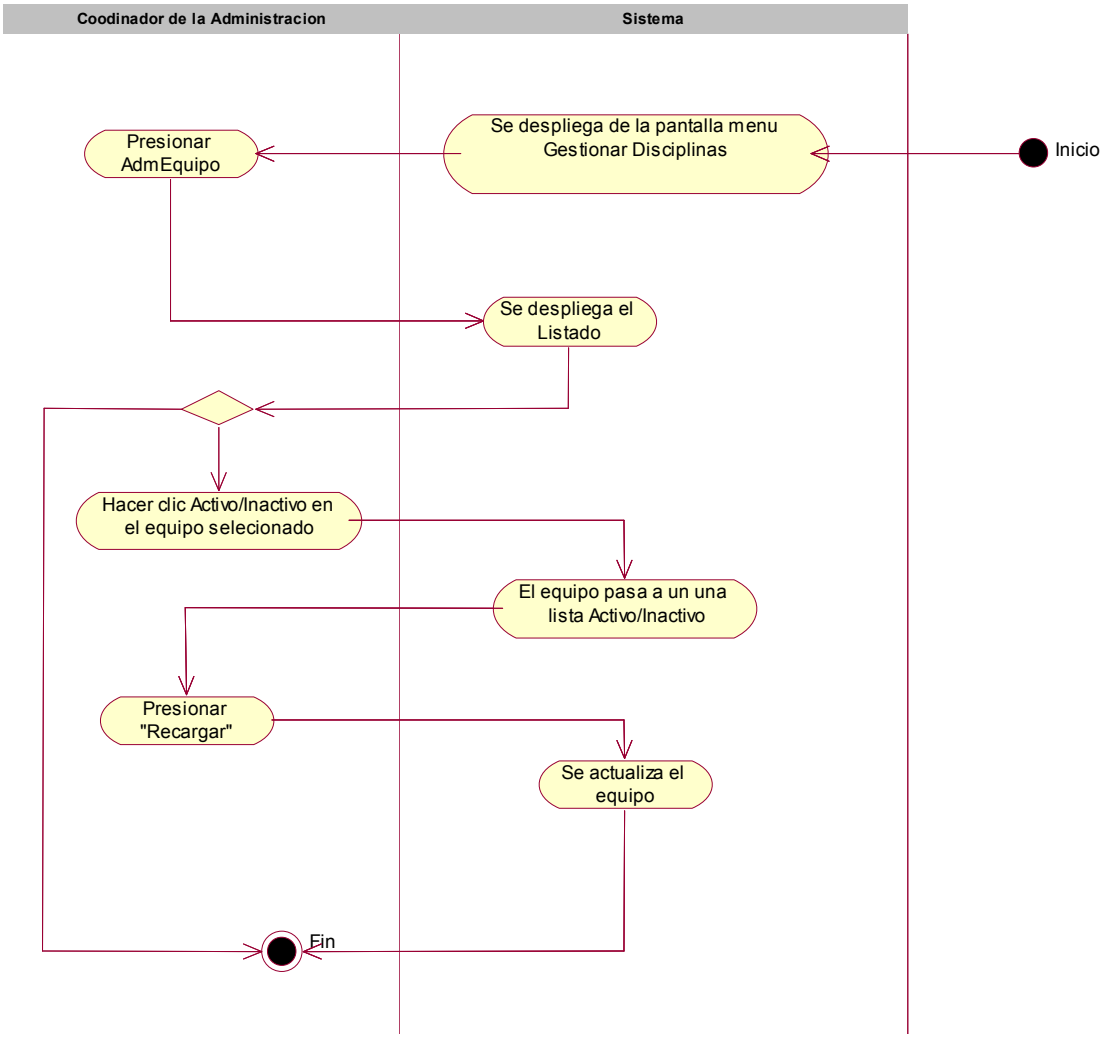


Figura 100: Diagrama de actividades Estado Equipo

Nuevo Campeonato

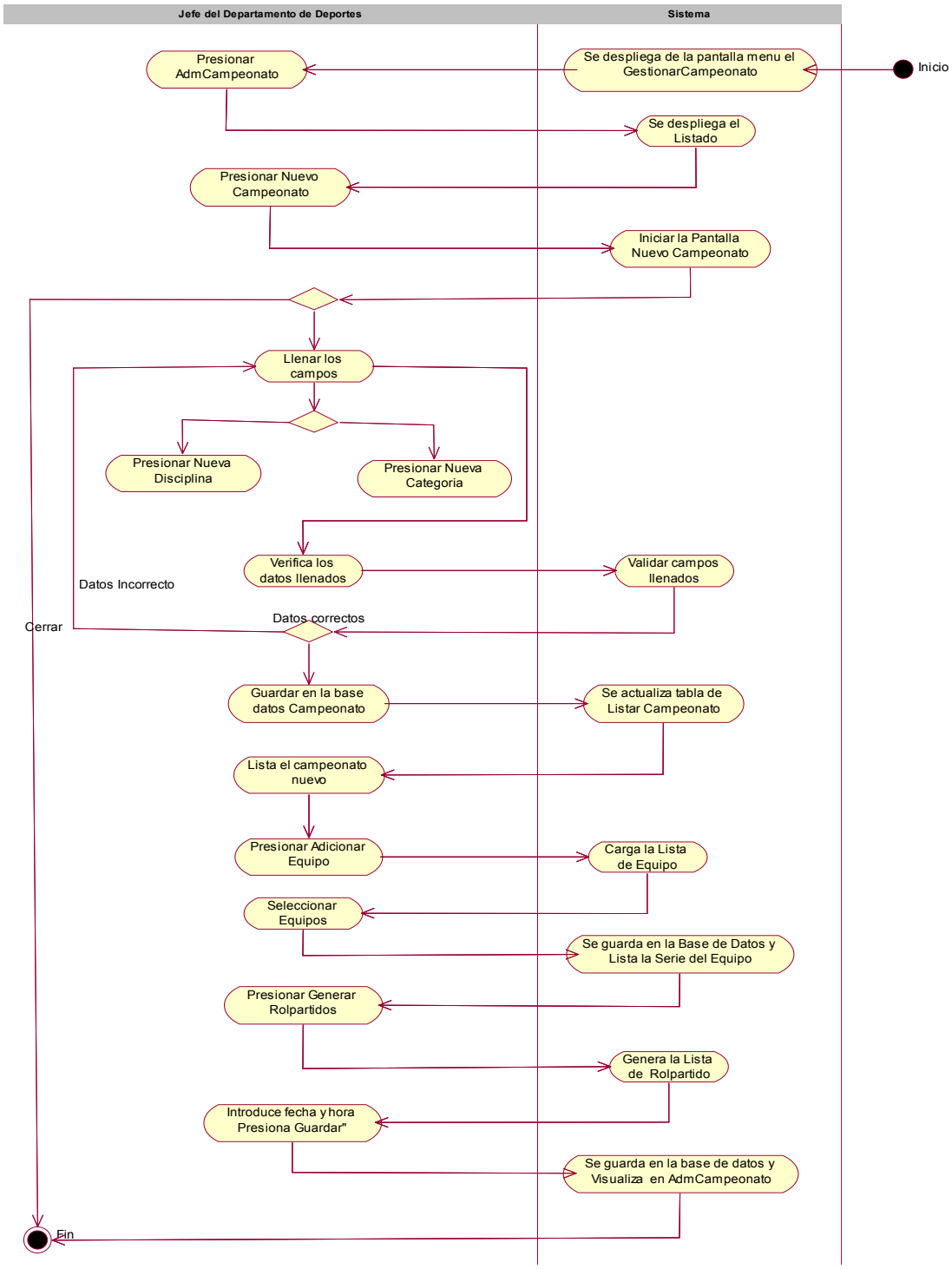


Figura 103: Diagrama de actividades Nuevo Campeonato

Campeonato Modificar

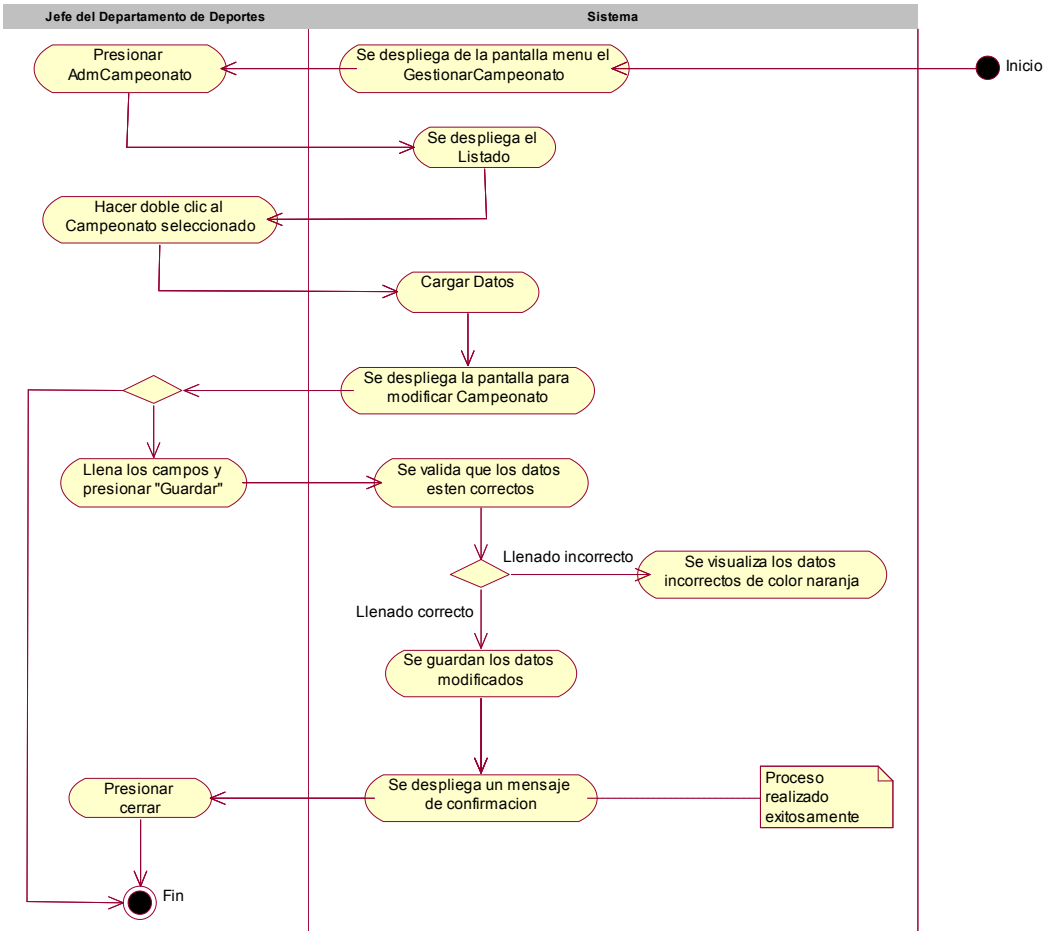


Figura 104: Diagrama de actividades Campeonato Modificar

Nuevo Rol

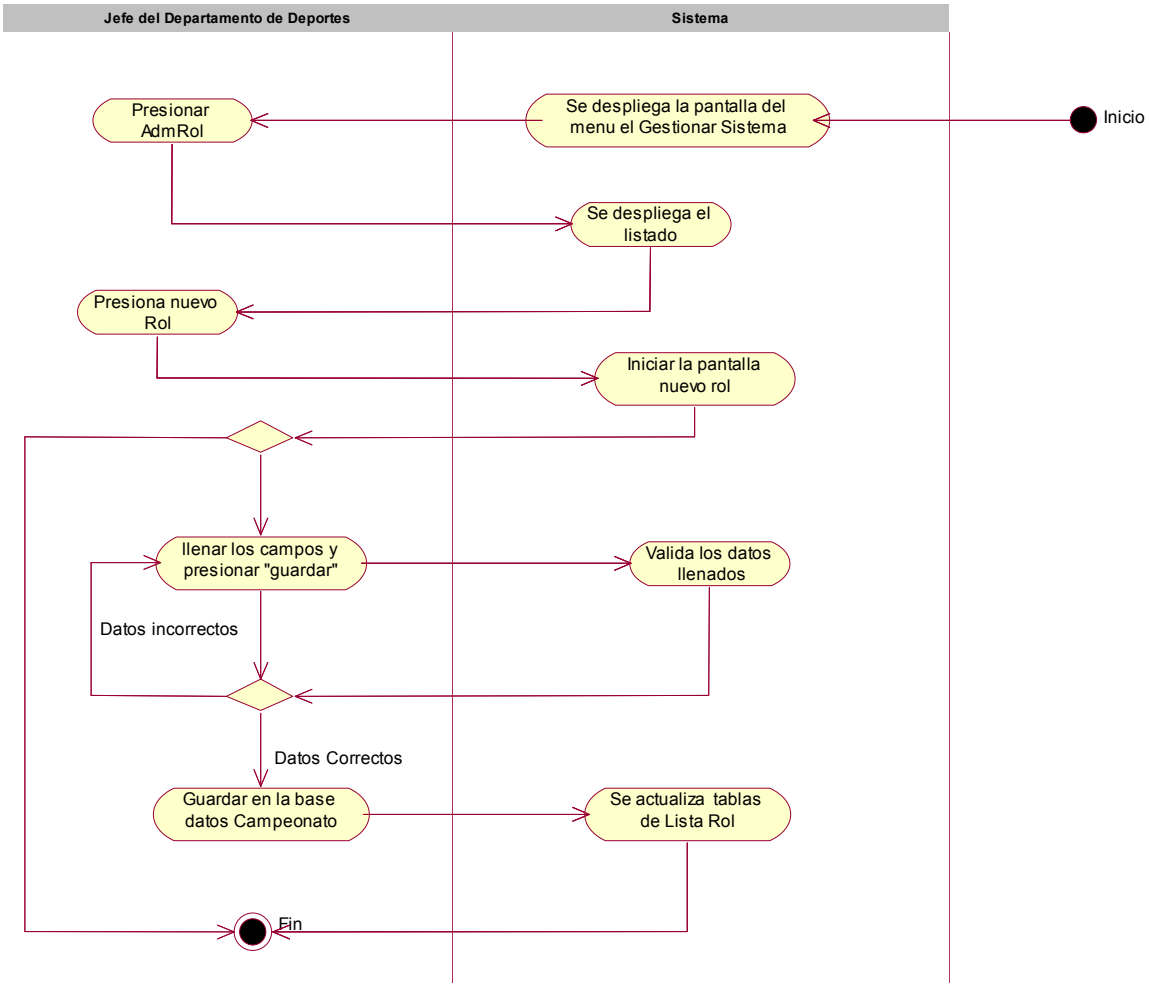


Figura 105: Diagrama de actividades Nuevo Rol

Modificar Rol

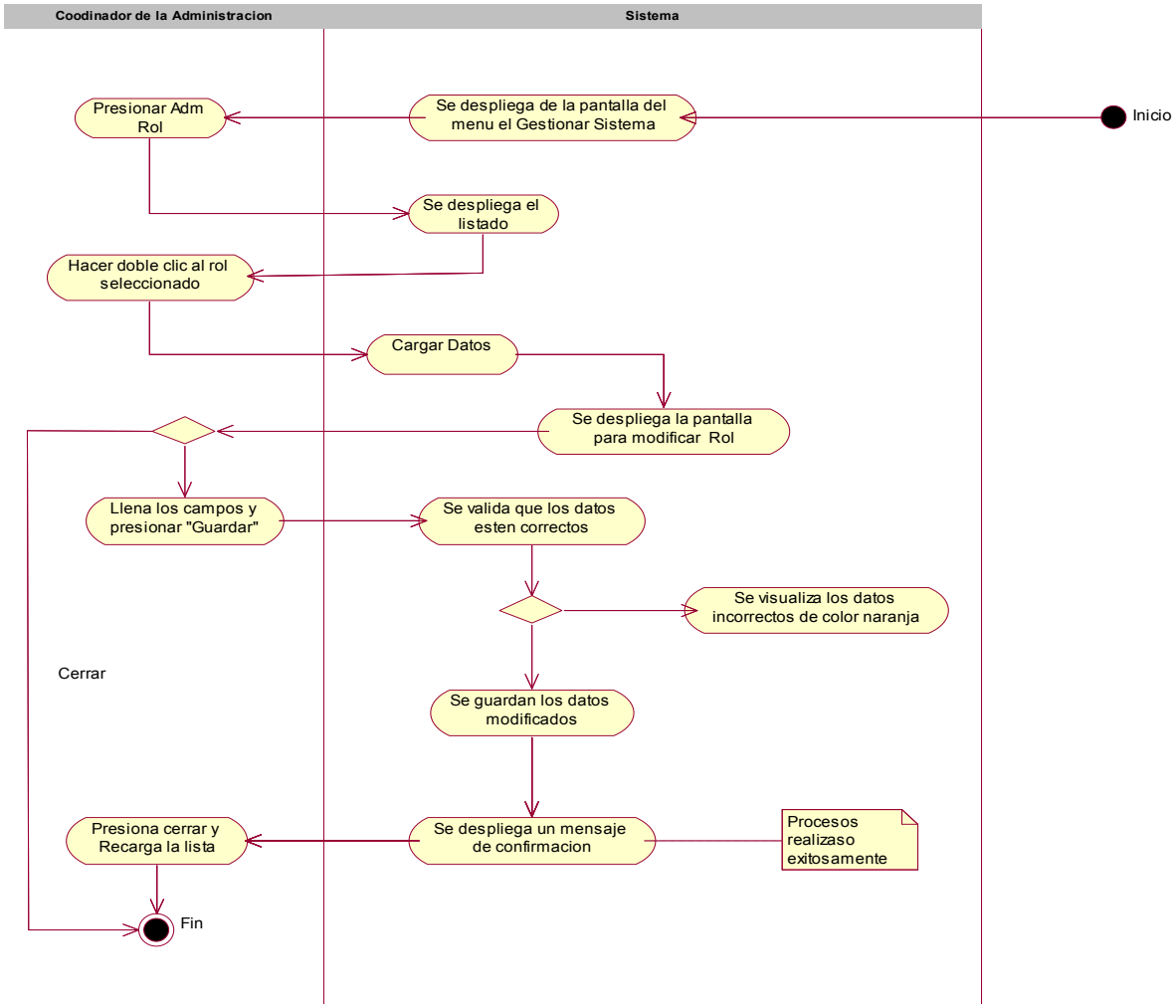


Figura 106: Diagrama de actividades Modificar Rol

Estado Rol

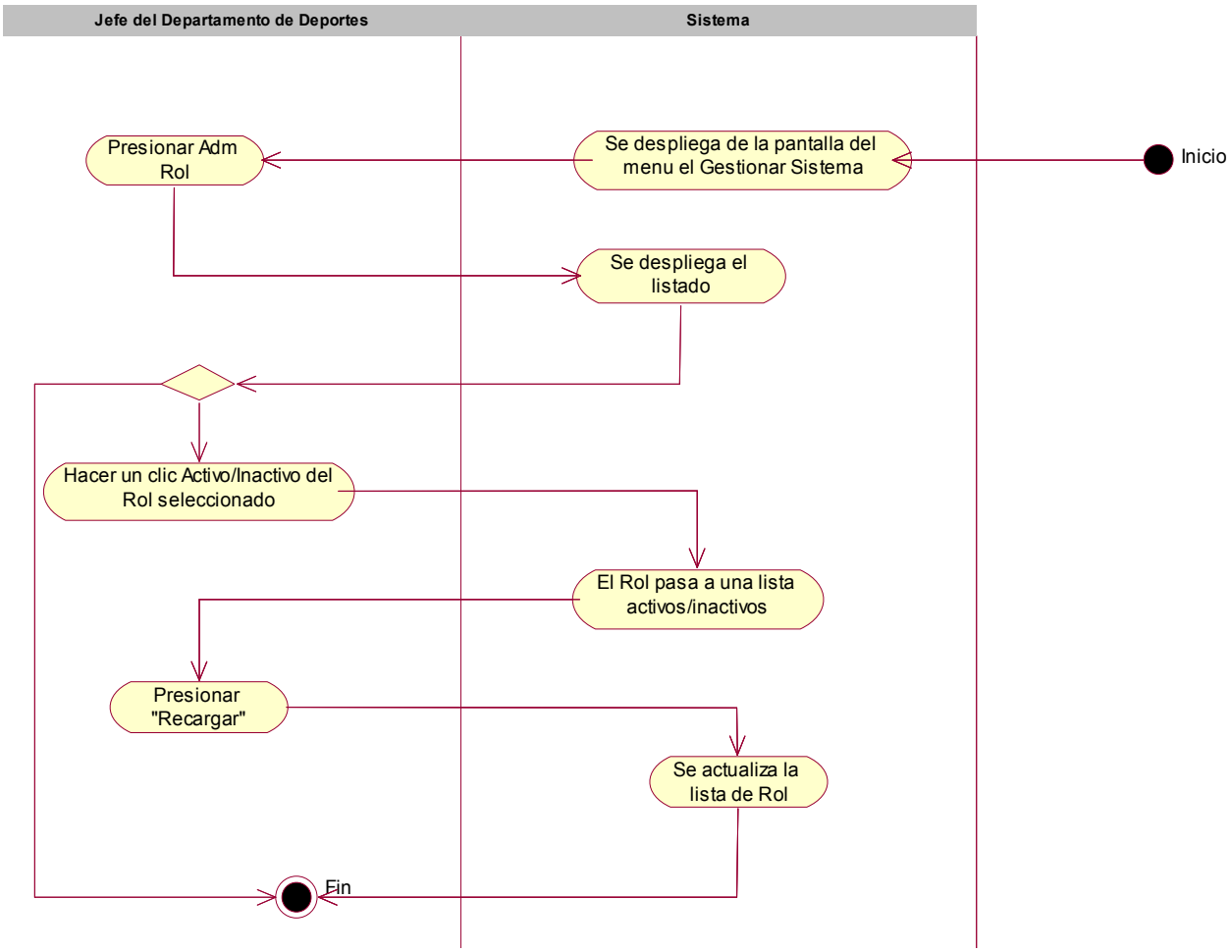


Figura 107: Diagrama de actividades Estado Rol

Nueva Foto

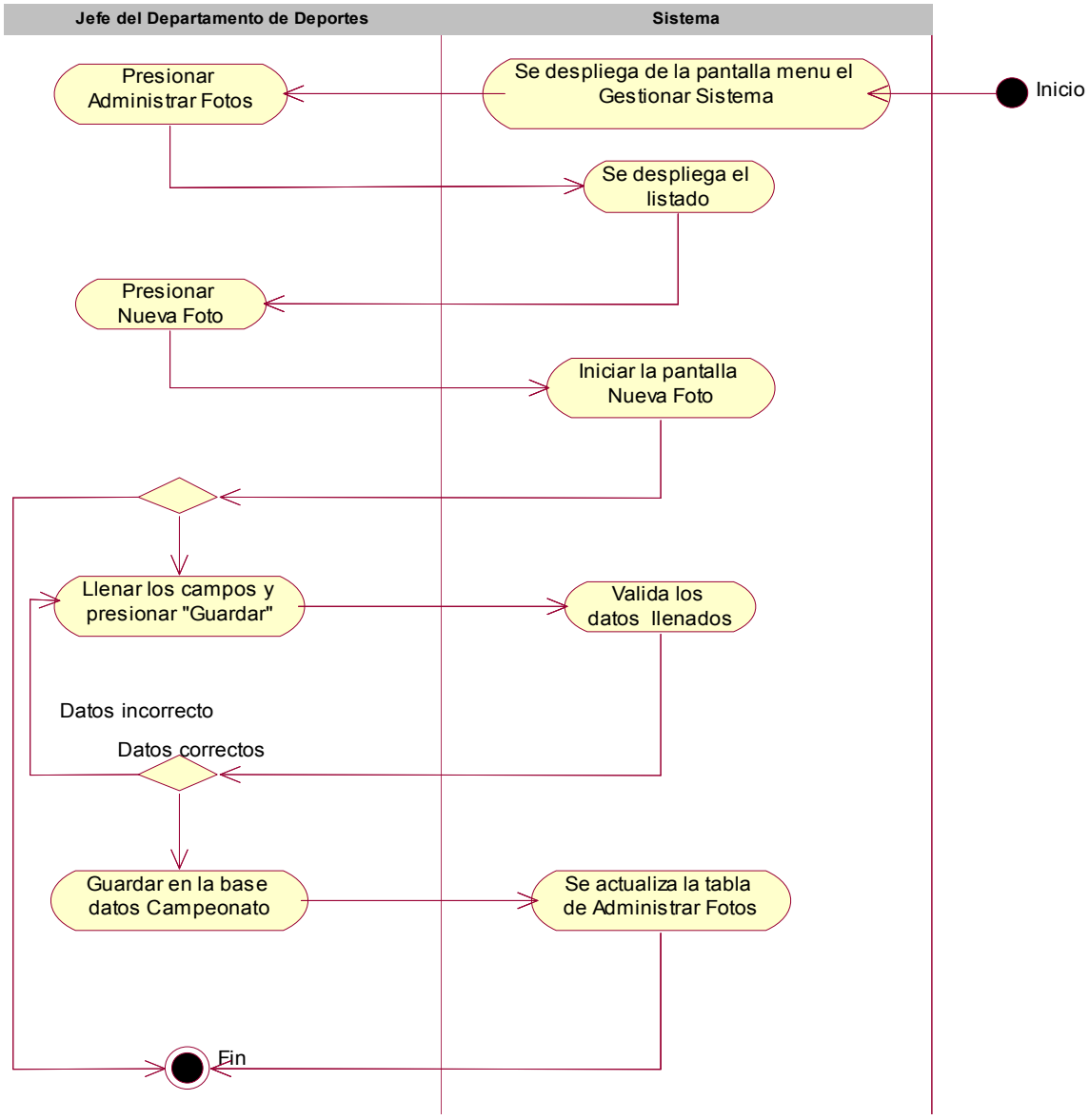


Figura 76: Diagrama de actividades Nuevo Foto

Modificar Foto

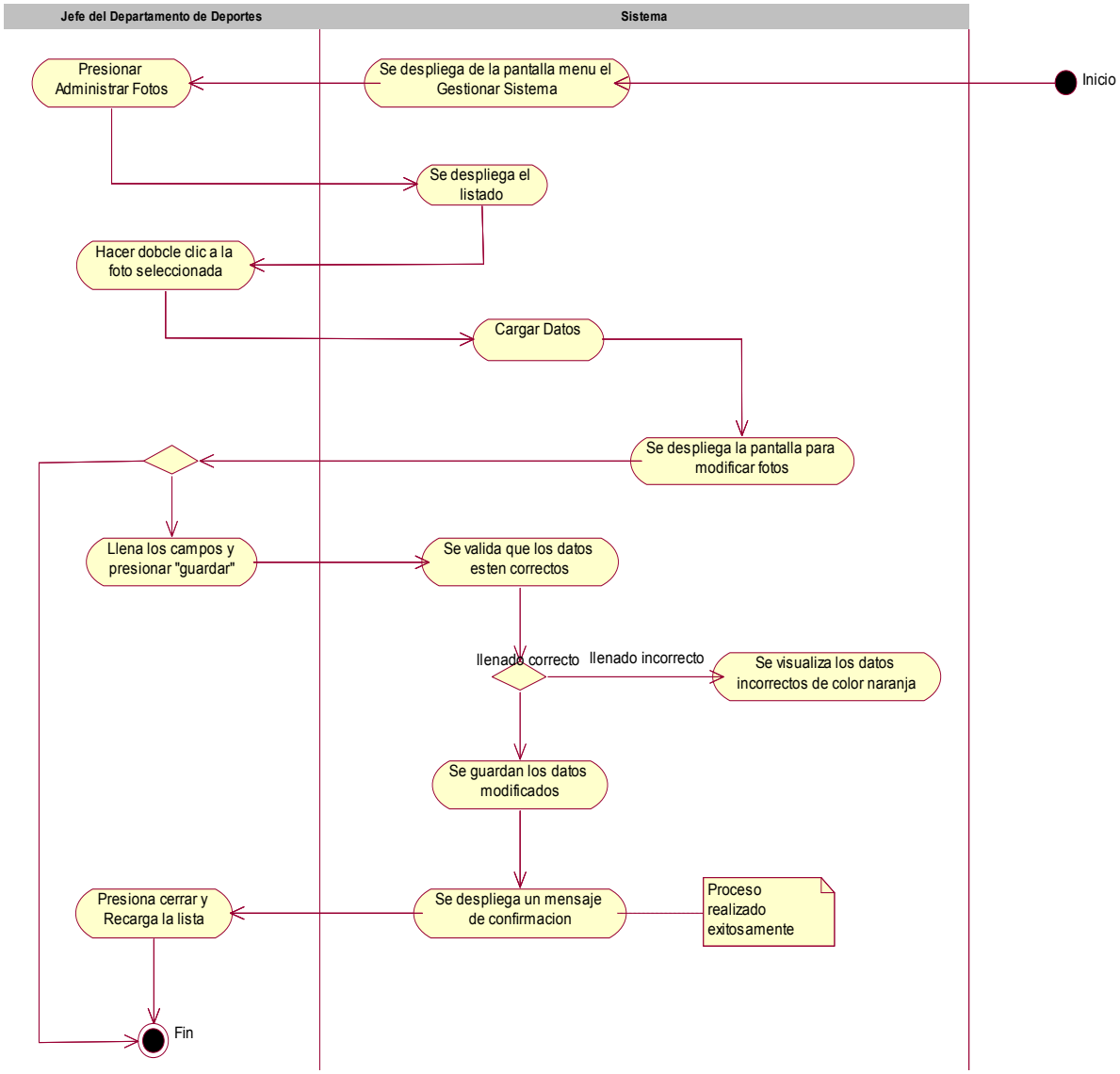


Figura 77: Diagrama de actividades Modificar Foto

Nuevo Publicaciones

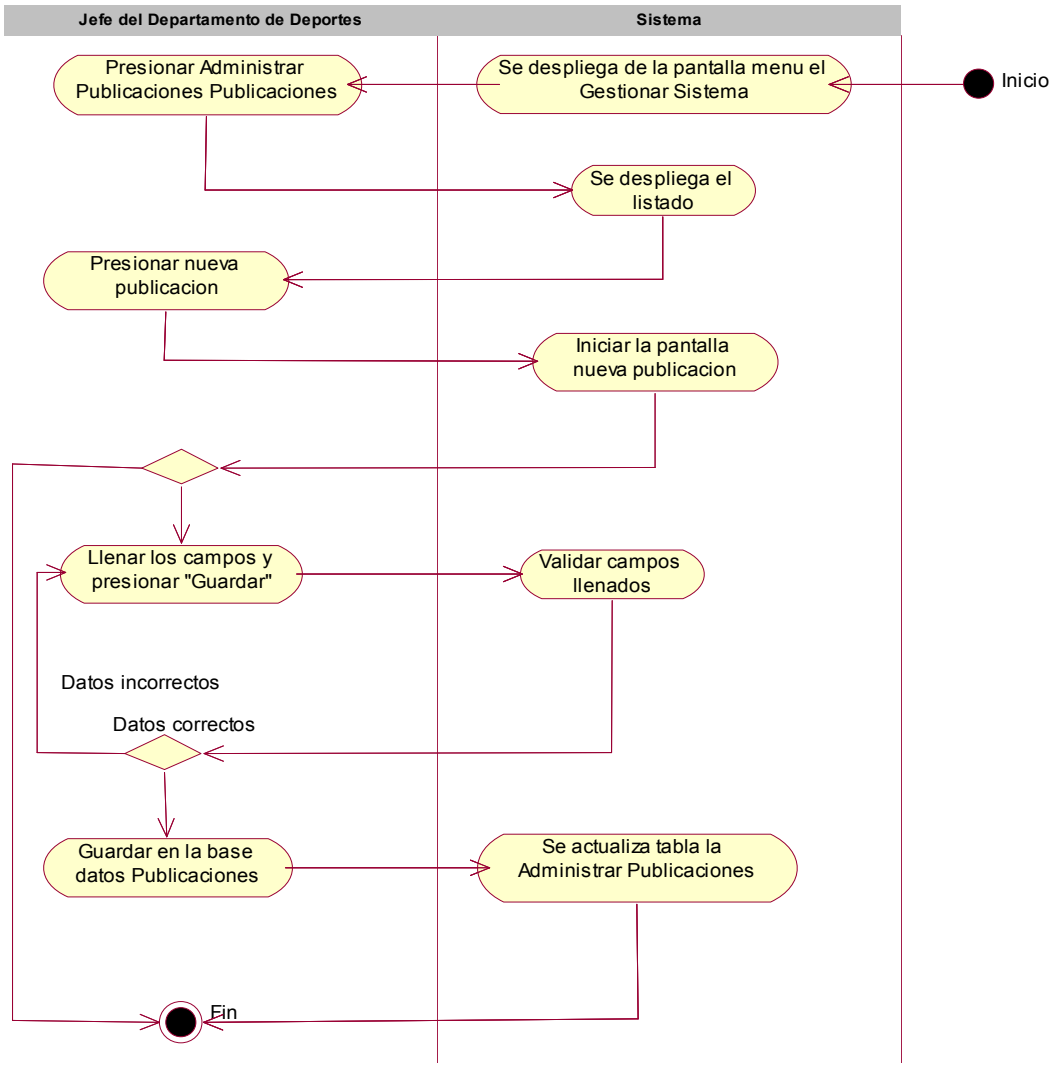


Figura 81: Diagrama de actividades Nuevo Publicaciones

Modificaciones Publicaciones

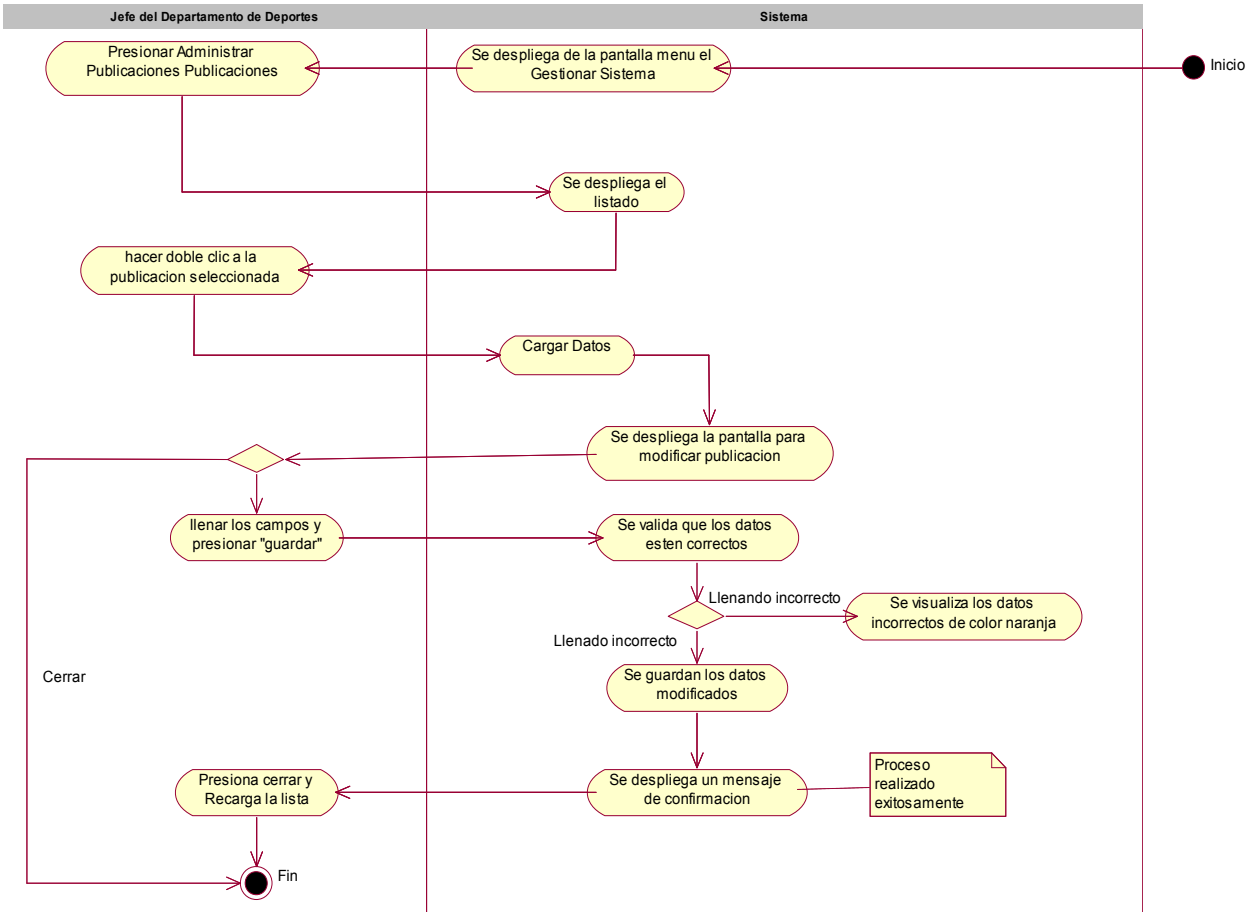


Figura 82: Diagrama de actividades Modificaciones Publicaciones

I.1.2 Modelado de Diagrama de Secuencia

I.1.2.1 Introducción

El diagrama de Secuencia es uno de los diagramas más efectivos para modelar interacción entre objetos en un sistema. Un diagrama de secuencia se modela para cada caso de uso.

Mientras que el diagrama de caso de uso permite el modelado de una vista de negocio del escenario, el diagrama de secuencia contiene detalles de implementación del escenario, incluyendo los objetos y clases que se usan para implementar el escenario, mensajes pasados entre los objetos.

Un diagrama de secuencia muestra los objetos que intervienen en el escenario con líneas verticales y los mensajes pasados entre los objetos vectores horizontales. Los mensajes se dibujan cronológicamente desde la parte superior del diagrama a la parte inferior, la distribución horizontal de los objetos es arbitraria.

I.1.2.2 Propósito

- Comprender la estructura del sistema deseado para la organización.
- Identificar posibles mejoras.
- Identificar clases de análisis y diseño.

I.1.2.3 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 por la organización.
- Definir un diagrama de secuencia para cada caso de uso del sistema.

Diagramas De Secuencia:

Iniciar Sesión

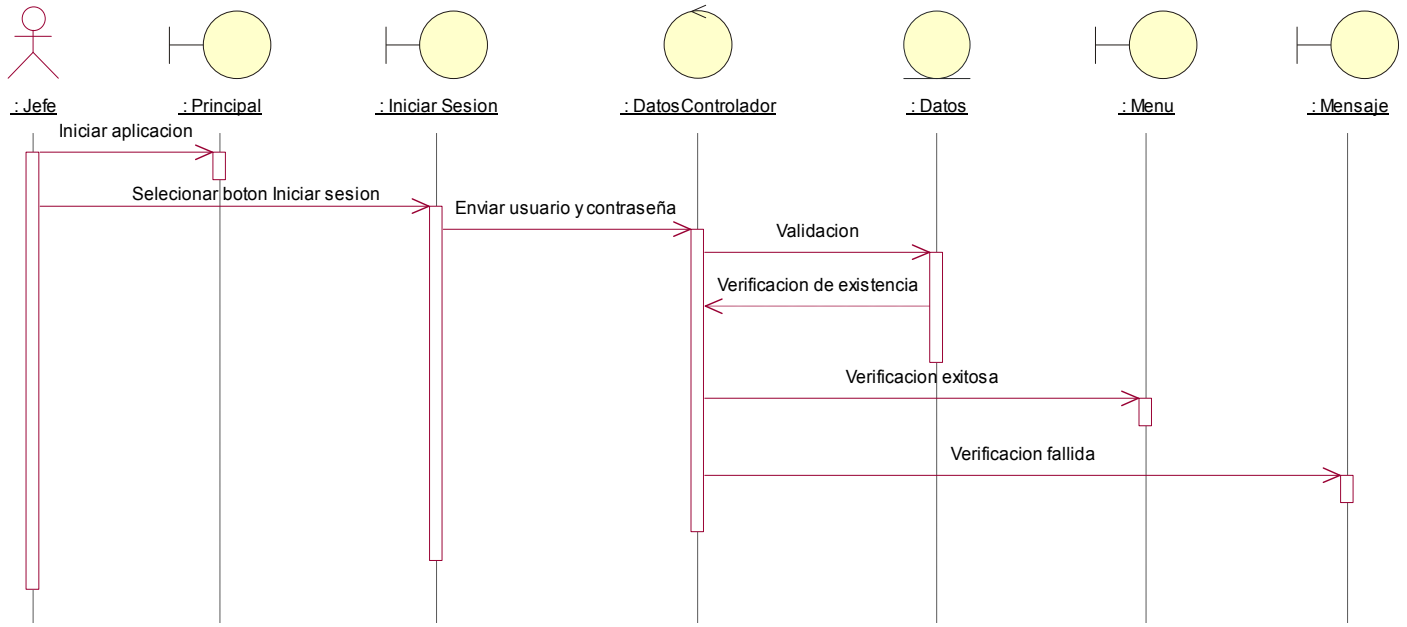


Figura 83 **Diagrama de Secuencia Iniciar Sesión**

Lista Menú: Administrar Usuarios

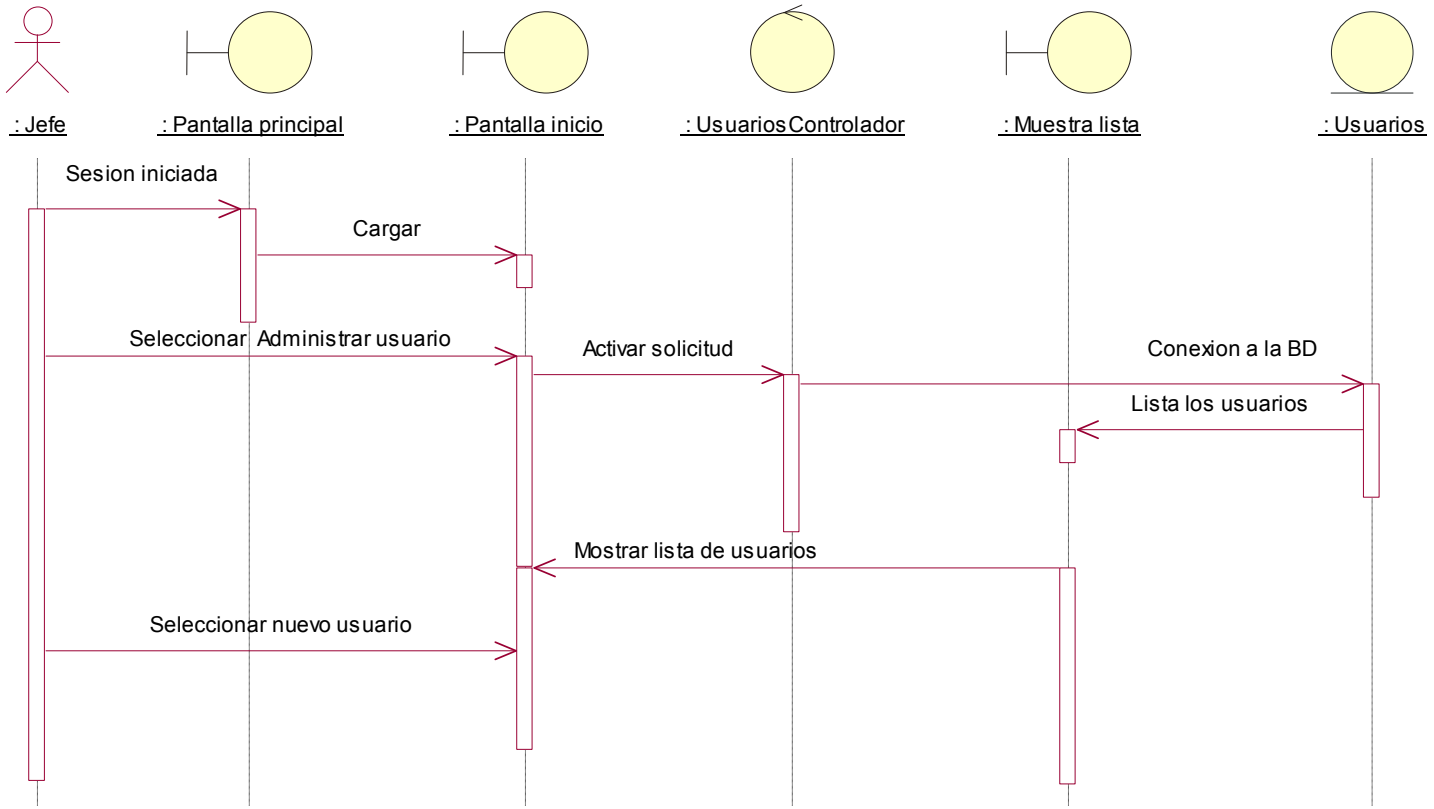


Figura 84 **Diagrama de Secuencia Administrar Usuarios**

Administrar Usuarios

Nuevo Usuario:

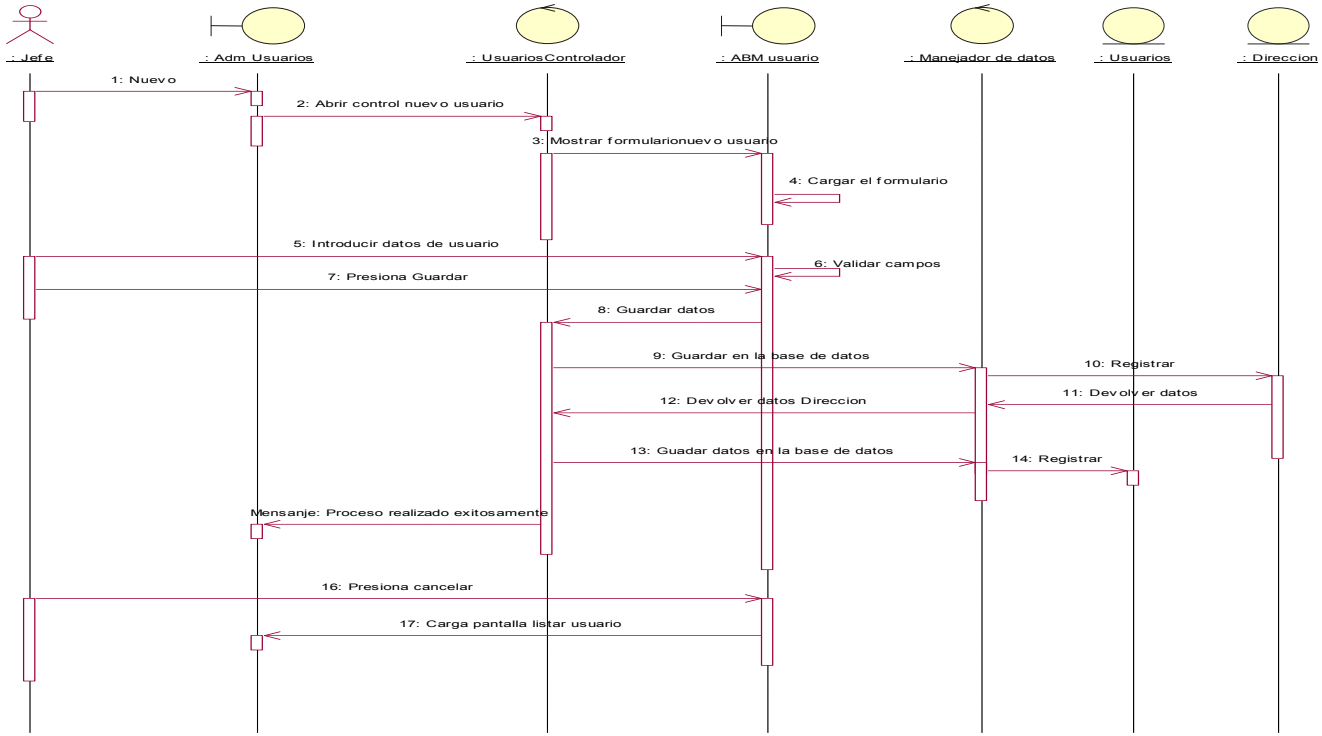


Figura 85 Diagrama de Secuencia Nuevo Usuario

Modificar Usuario

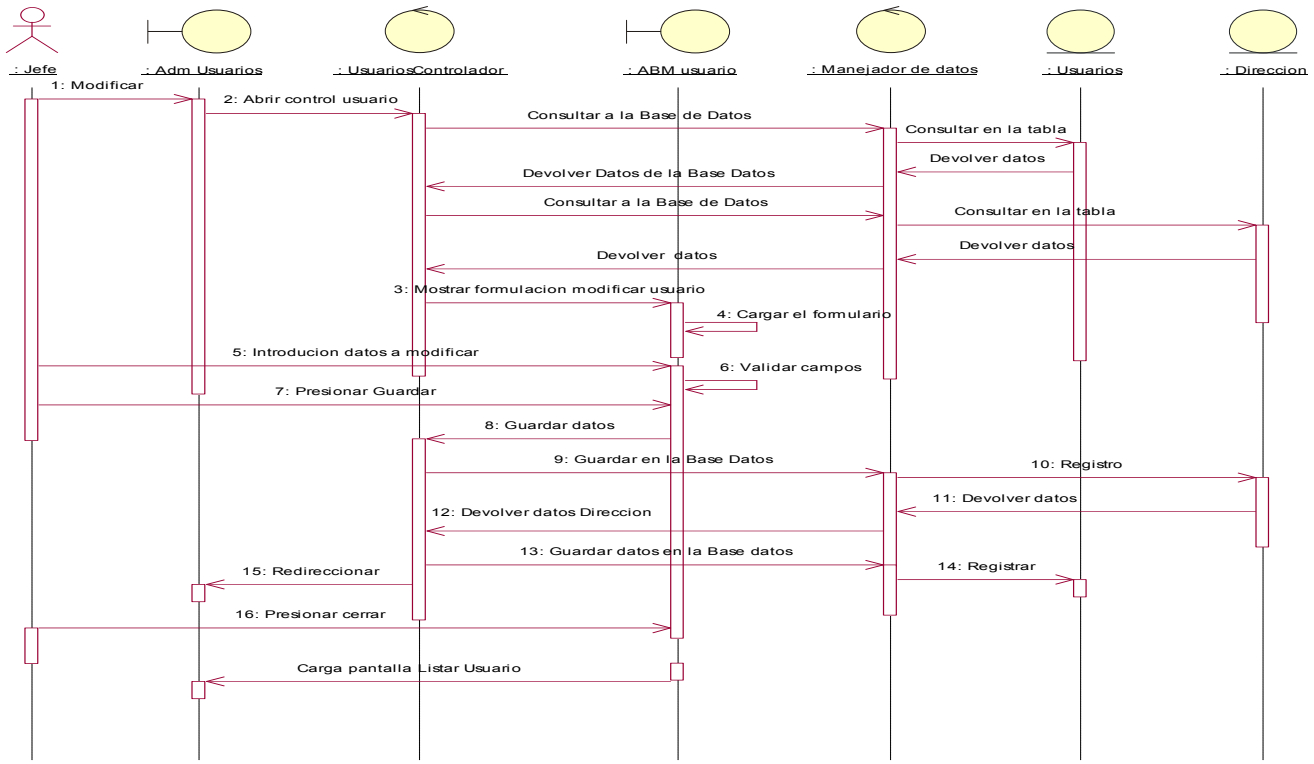


Figura 86 Diagrama de Secuencia Modificar Usuario

Estado Usuario

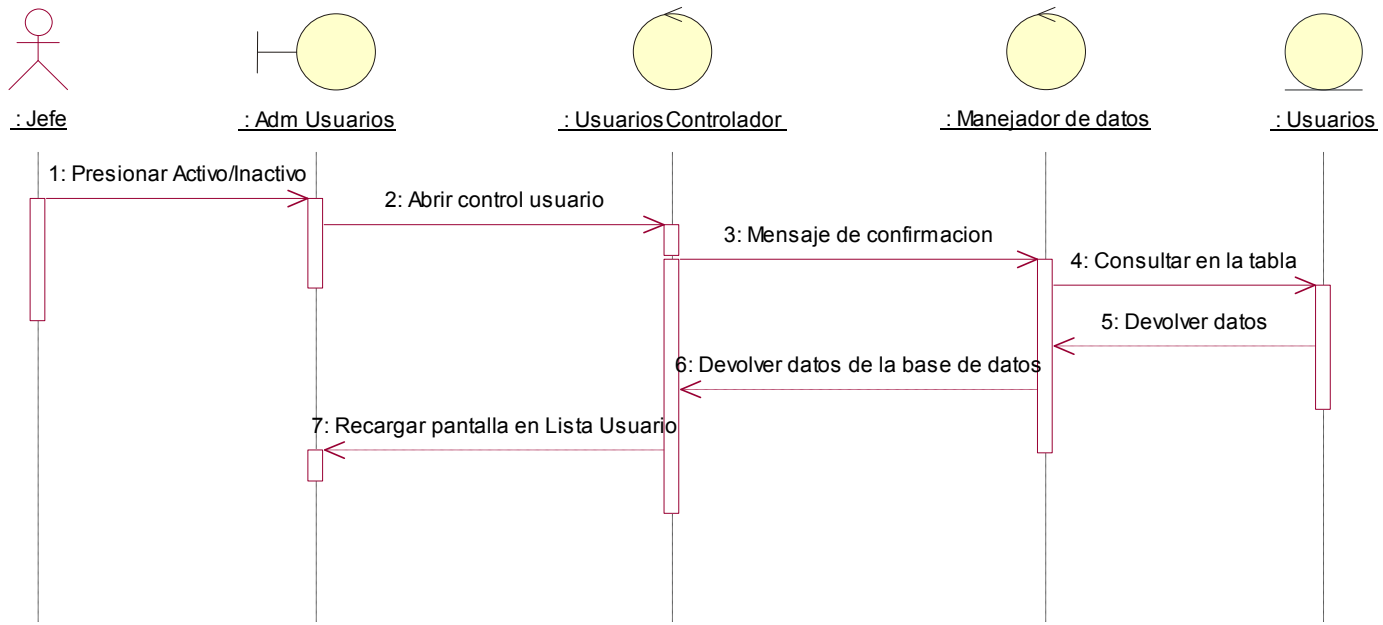


Figura 87 Diagrama de Secuencia Estado Usuario

Nuevo Profesión

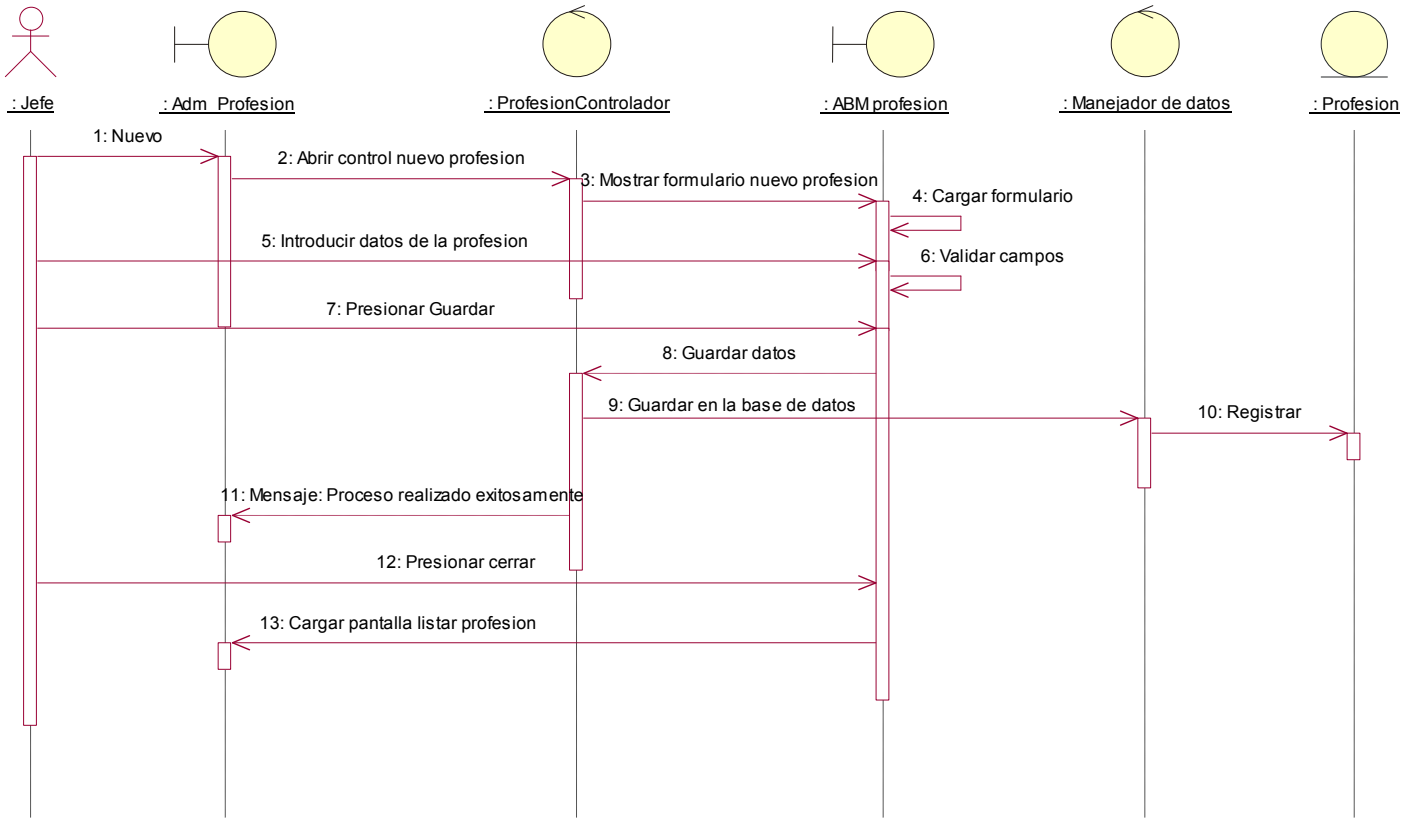


Figura 88 Diagrama de Secuencia Nuevo Profesión

Modificar Profesión

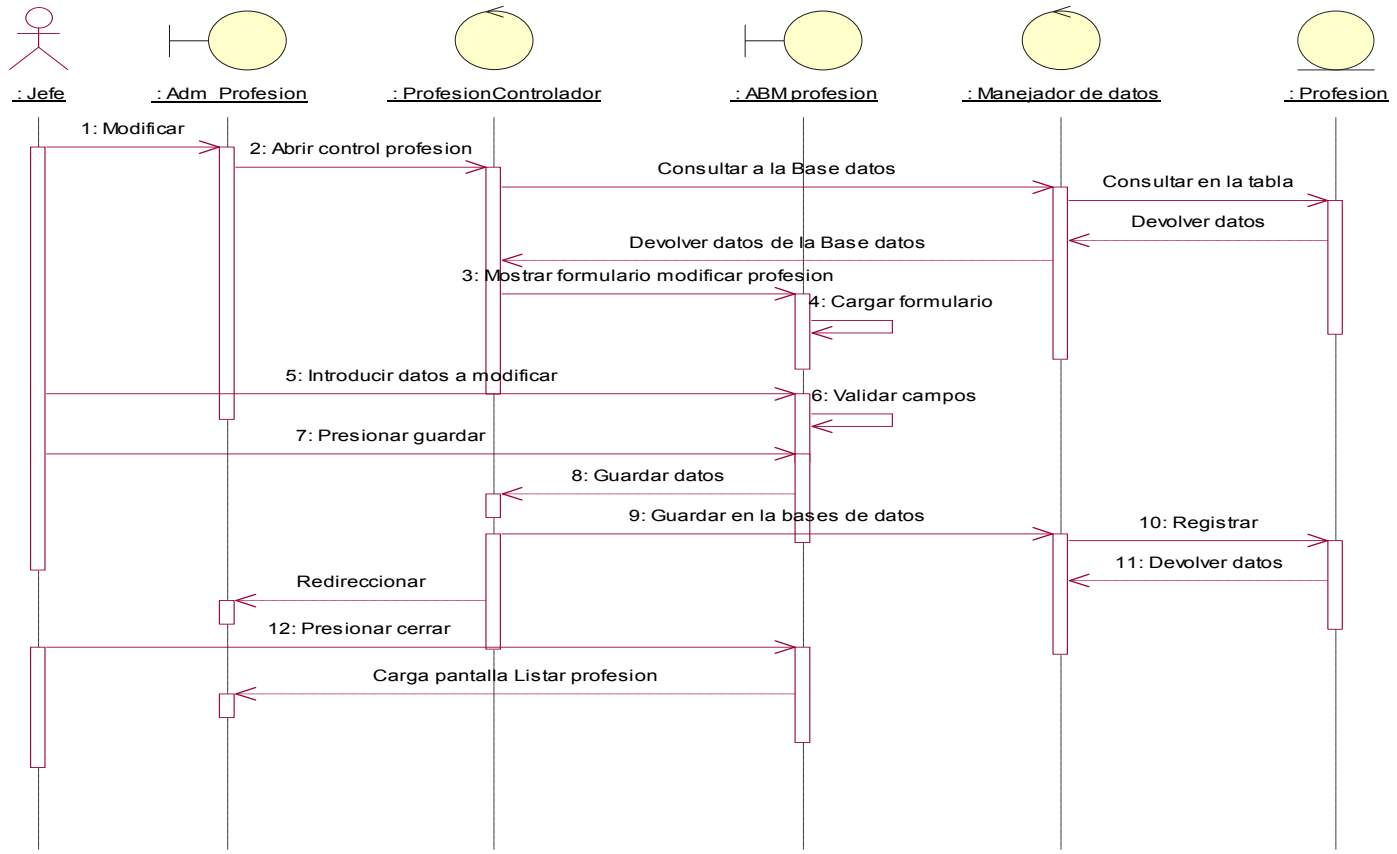


Figura 89 **Diagrama de Secuencia Modificar Profesión**

Nuevo Alumno

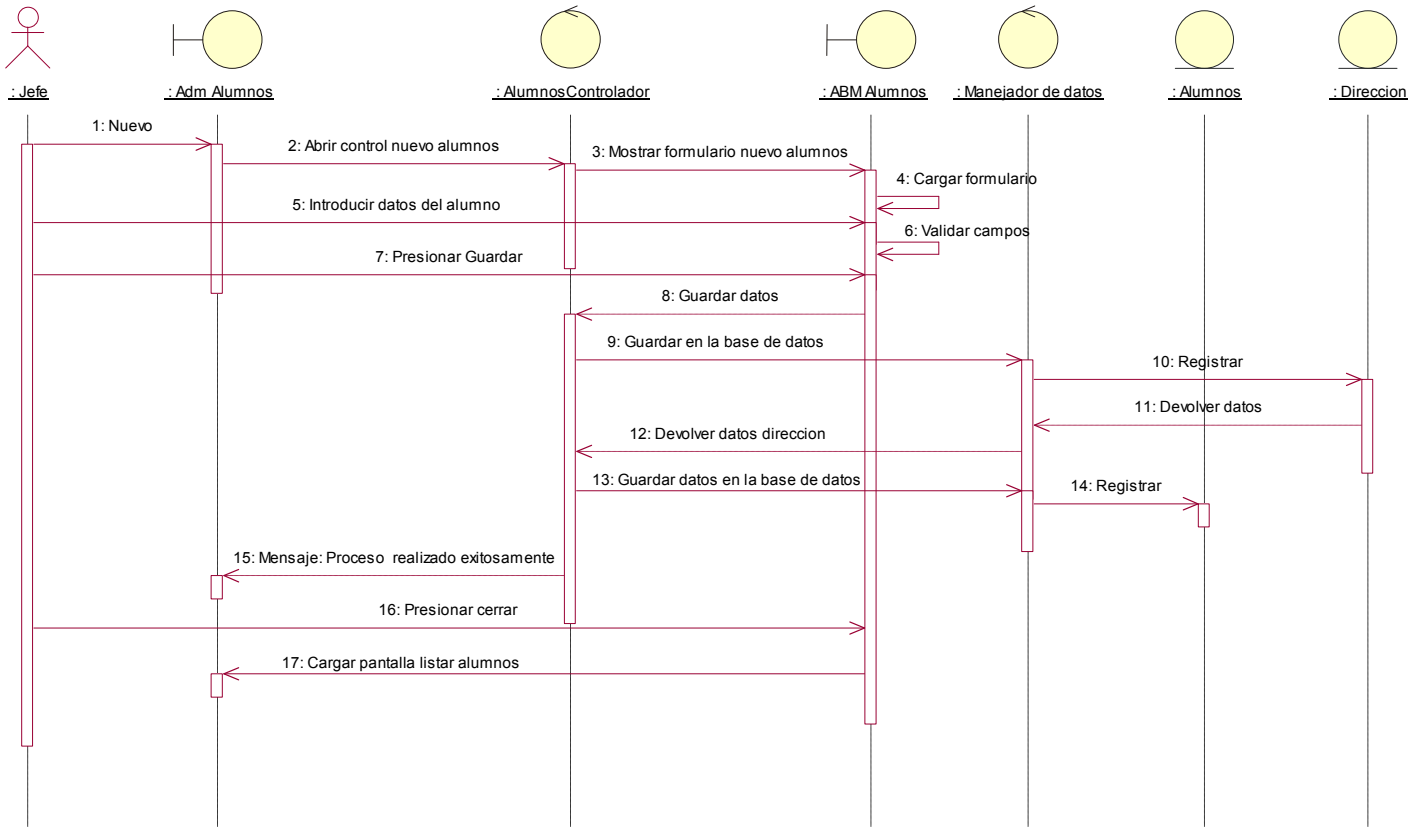


Figura 90 **Diagrama de Secuencia Nuevo Alumno**

Modificar Alumnos

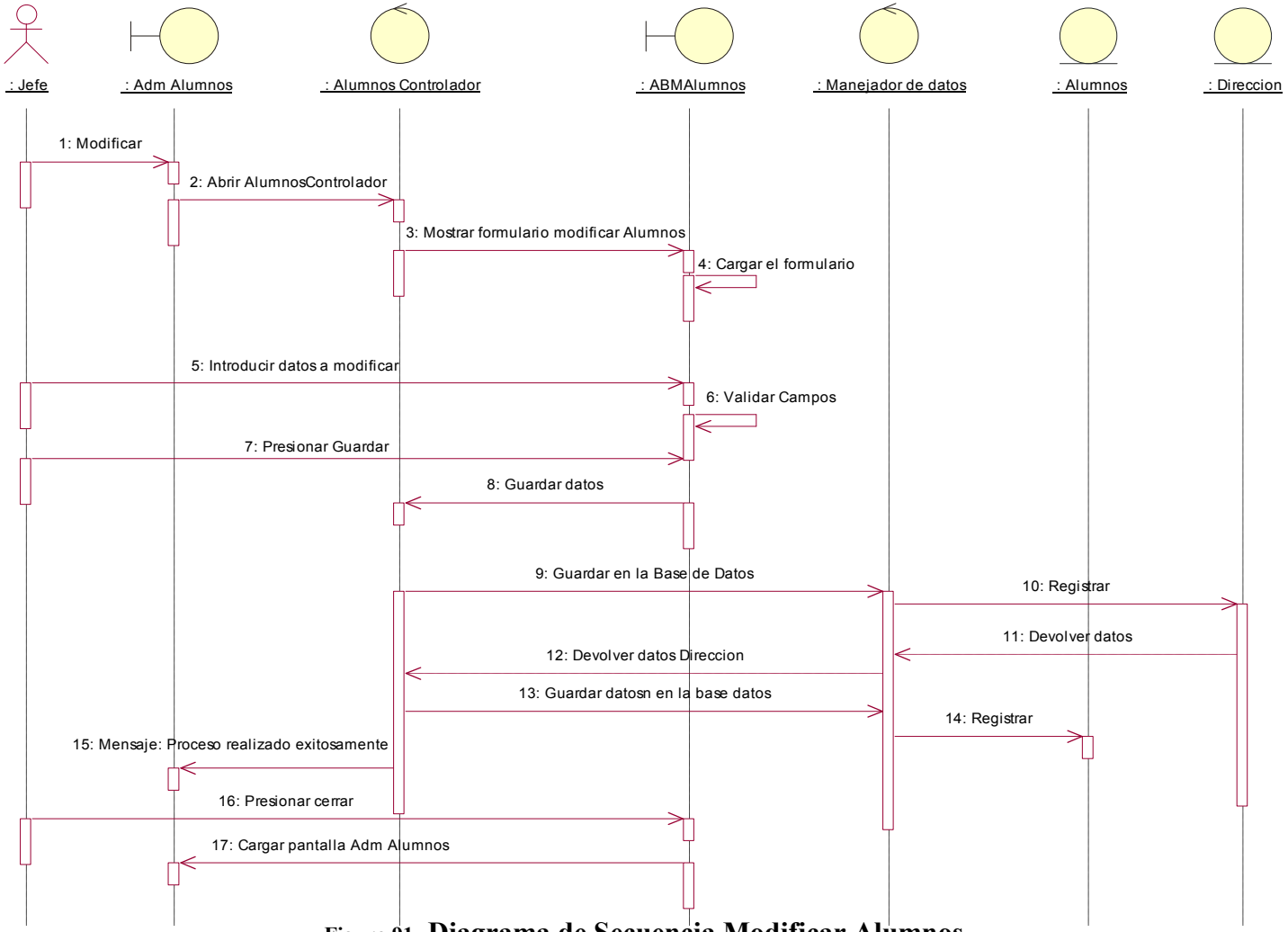


Figura 91 Diagrama de Secuencia Modificar Alumnos

Estado Alumno

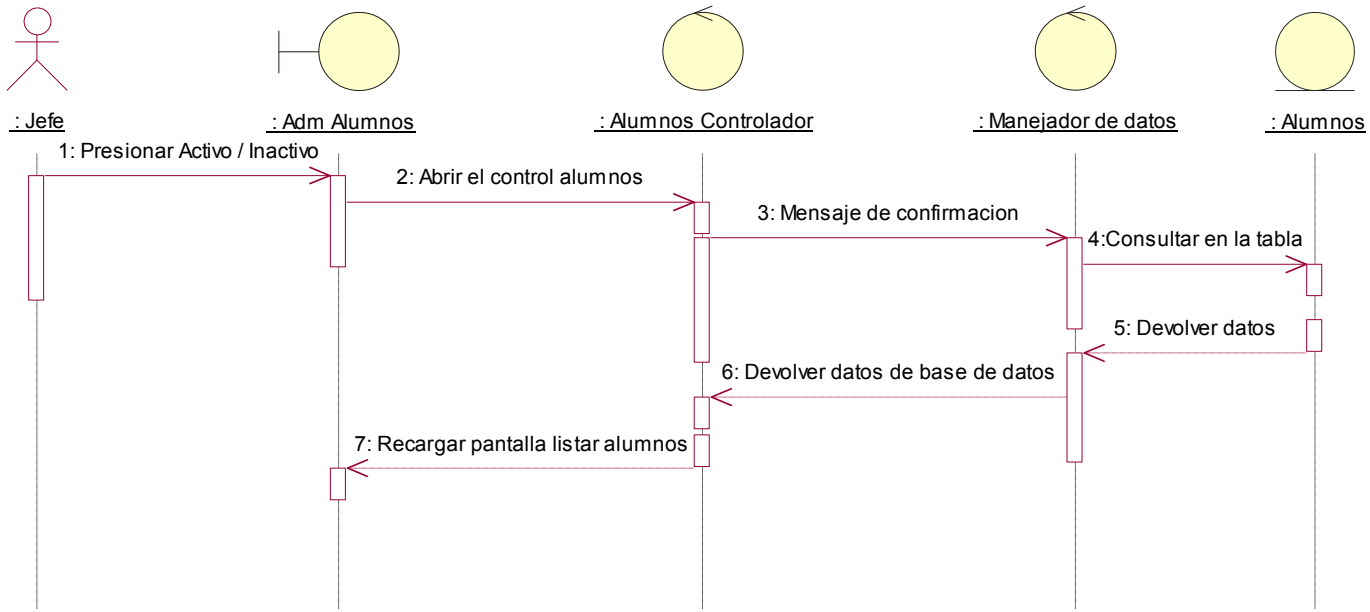


Figura 92 **Diagrama de Secuencia Estado Alumno**

Nueva Ciudad

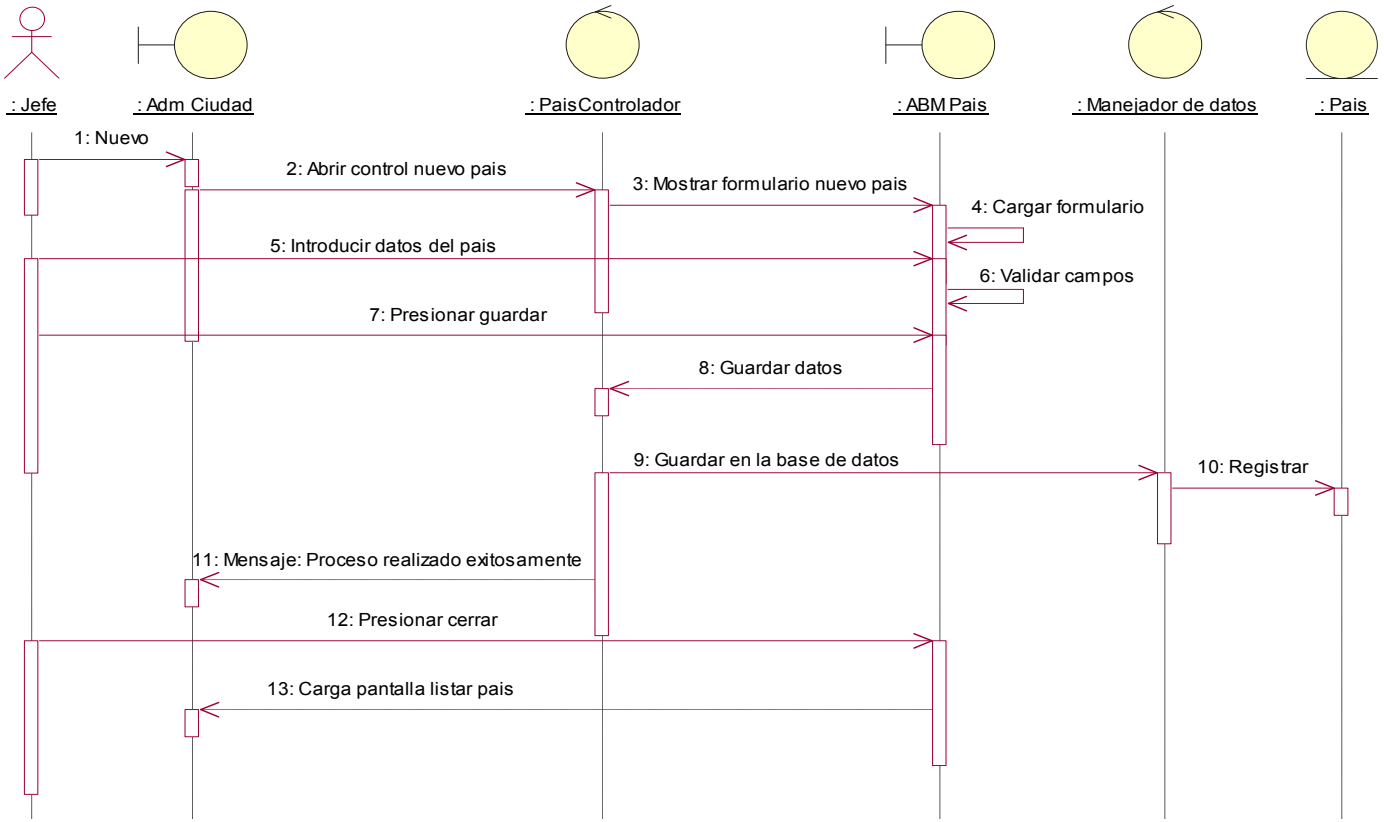


Figura 93 **Diagrama de Secuencia Nueva Ciudad**

Modificar Ciudad

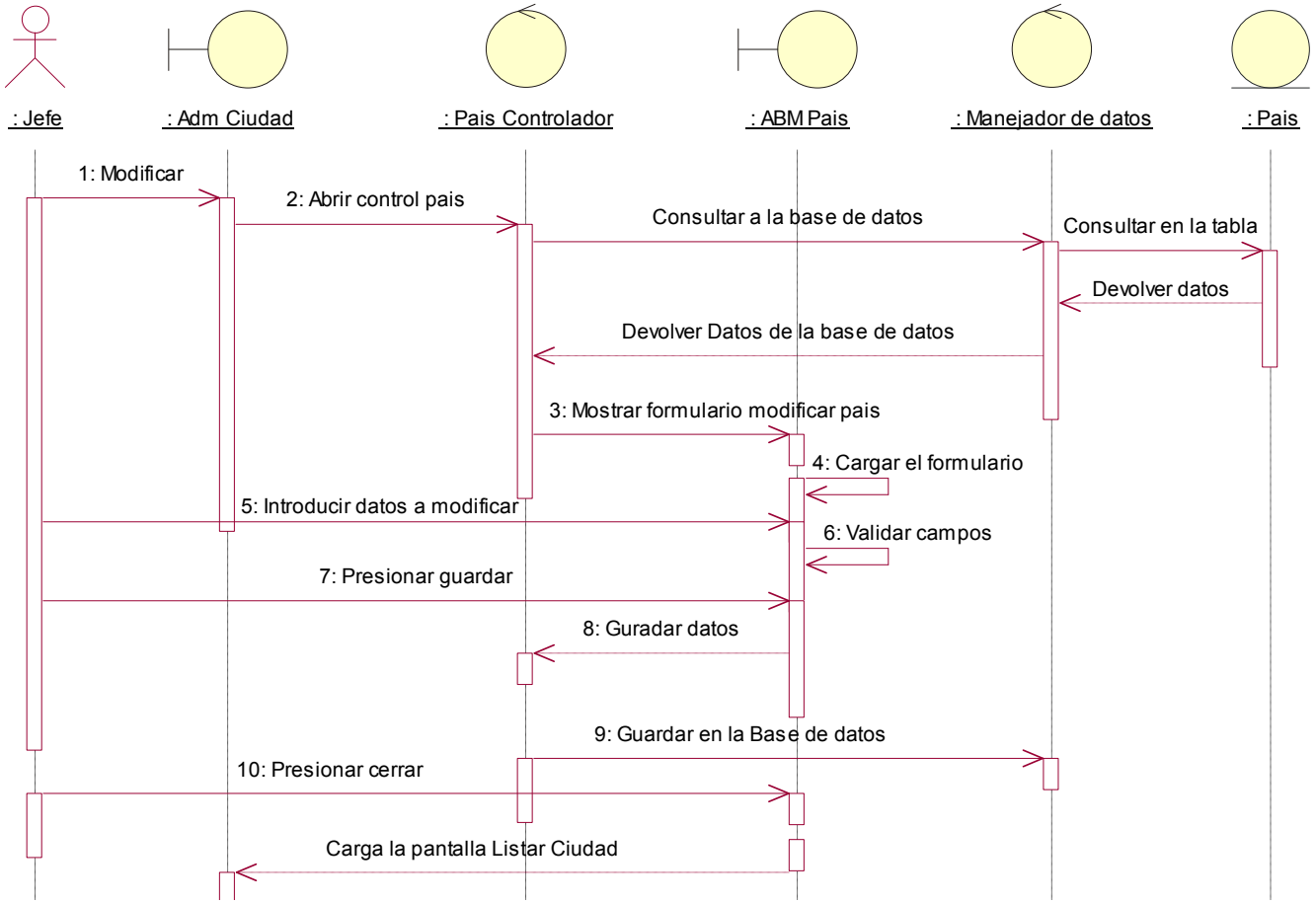


Figura 94 **Diagrama de Secuencia Modificar Ciudad**

Nuevo Departamento

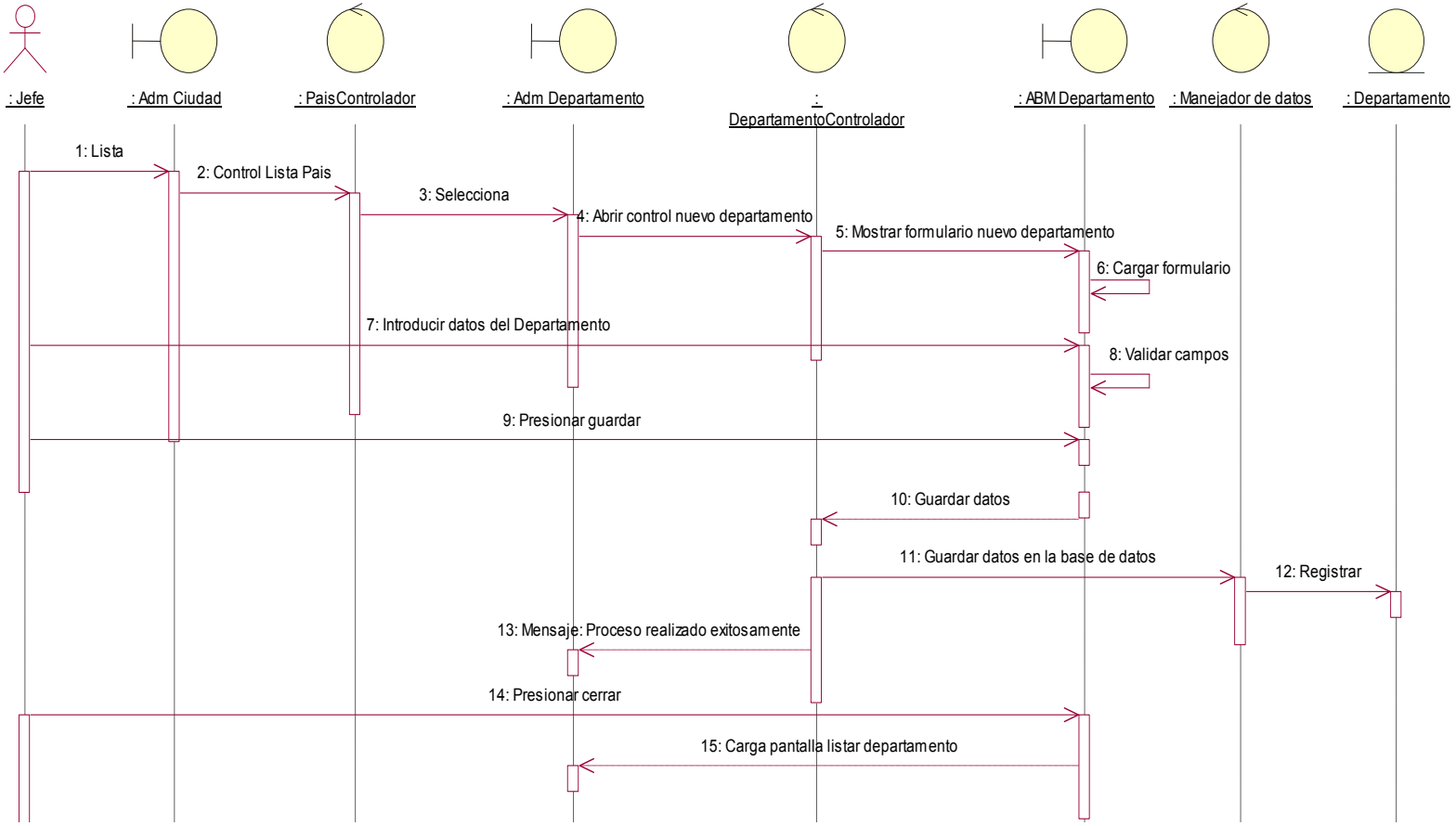


Figura 95 **Diagrama de Secuencia Nuevo Departamento**

Modificar Departamento

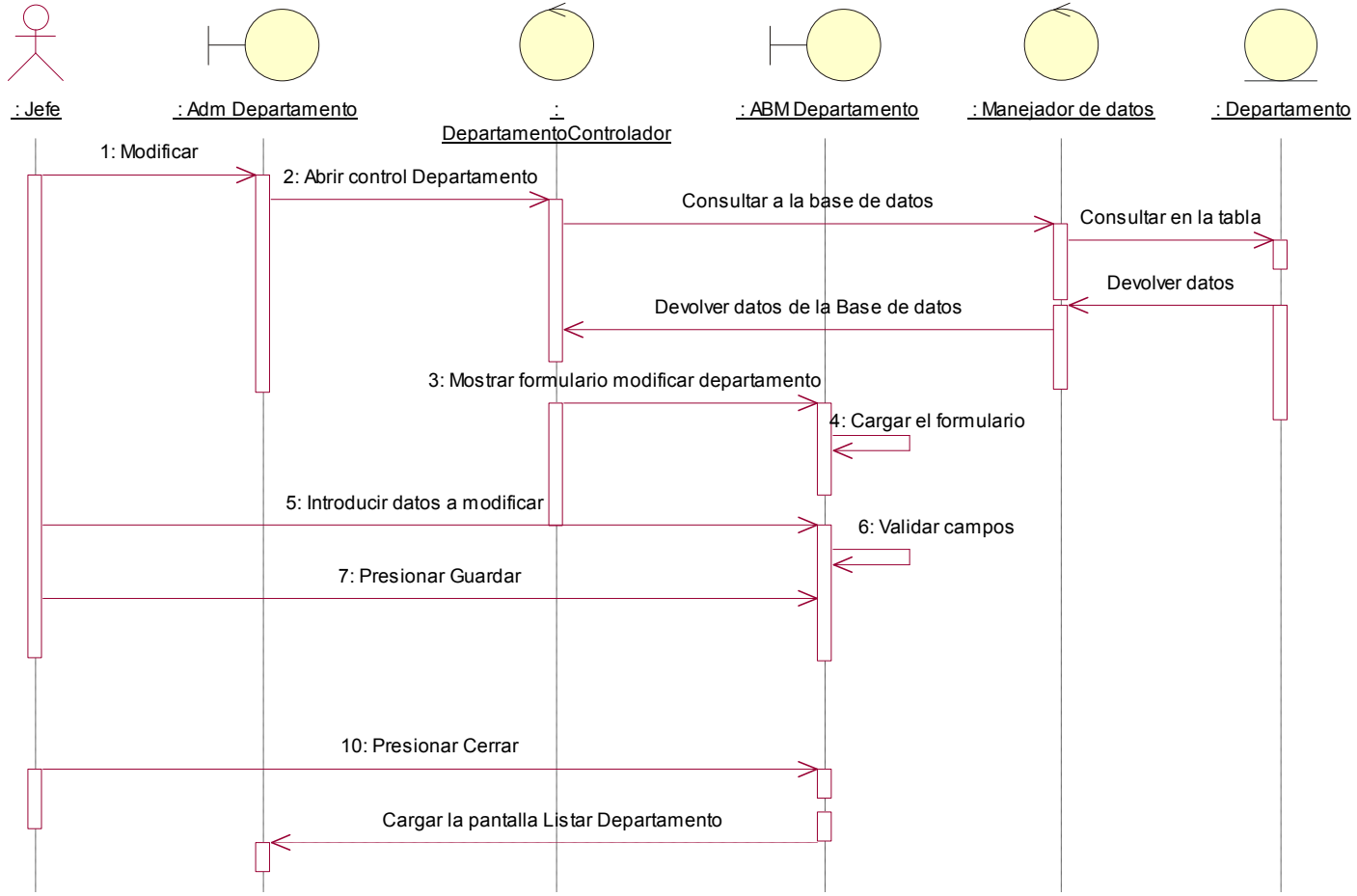


Figura 96 Diagrama de Secuencia Modificar Departamento

Nuevo Provincia

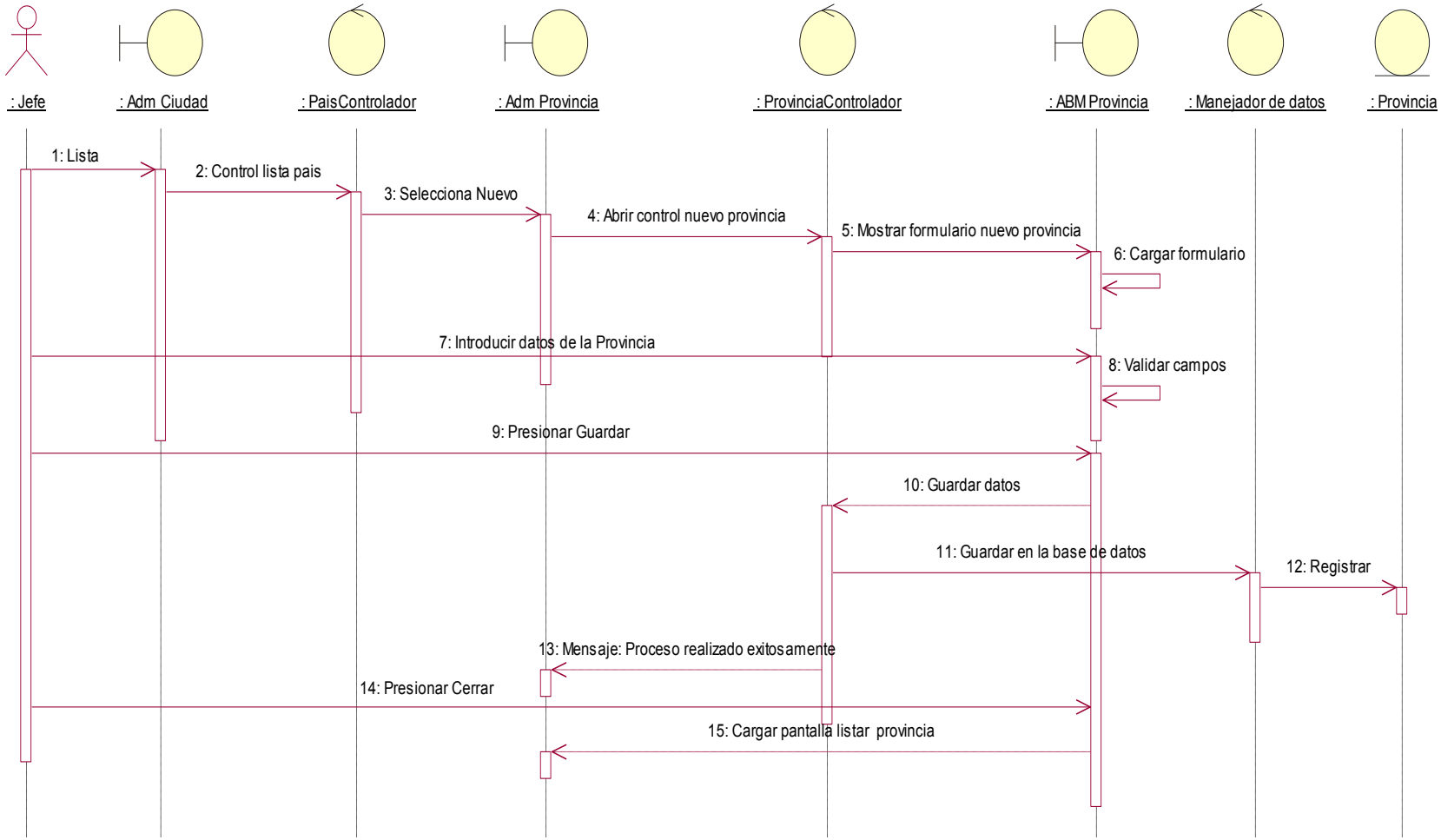


Figura 97 **Diagrama de Secuencia Nuevo Provincia**

Modificar Provincia

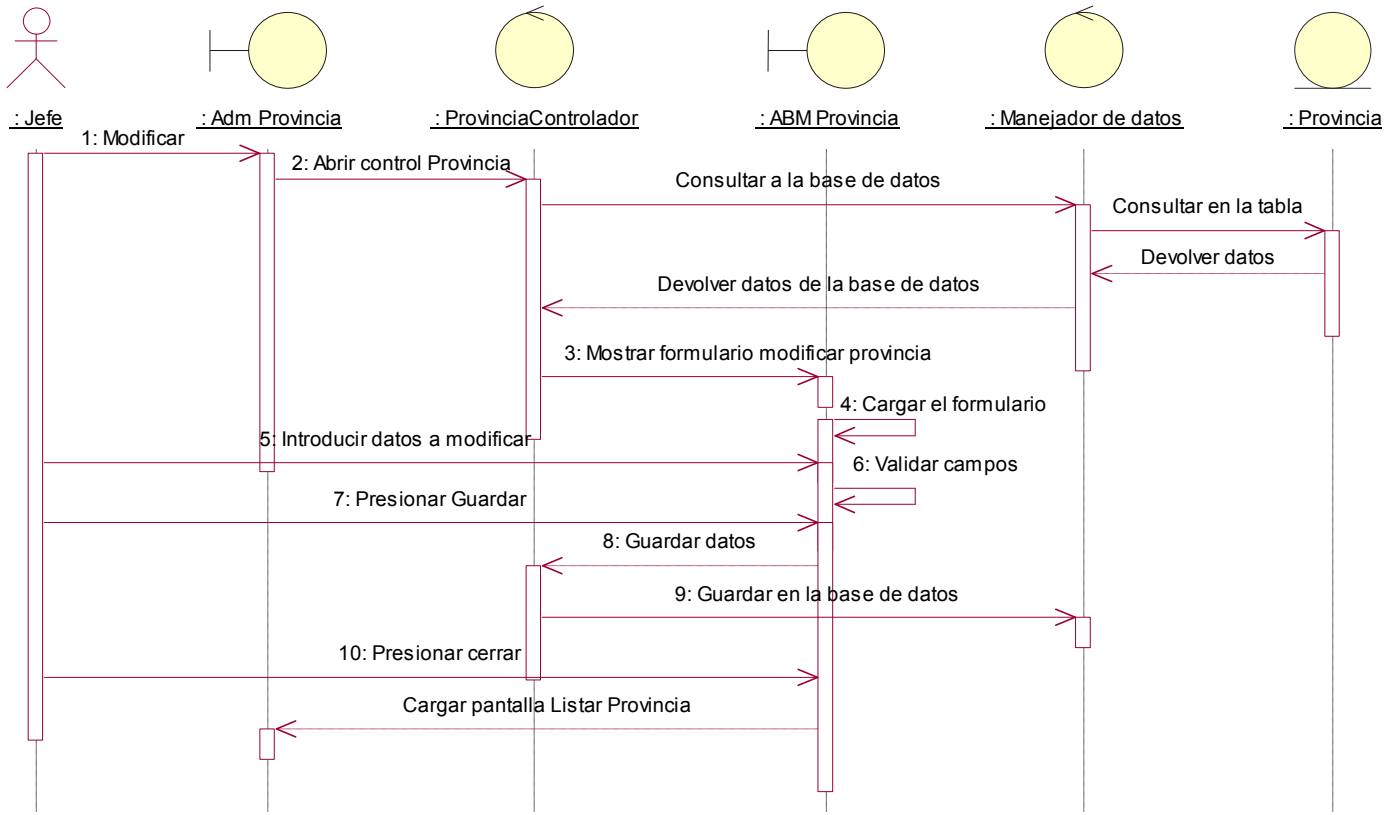


Figura 98 **Diagrama de Secuencia Modificar Provincia**

Nuevo Disciplina

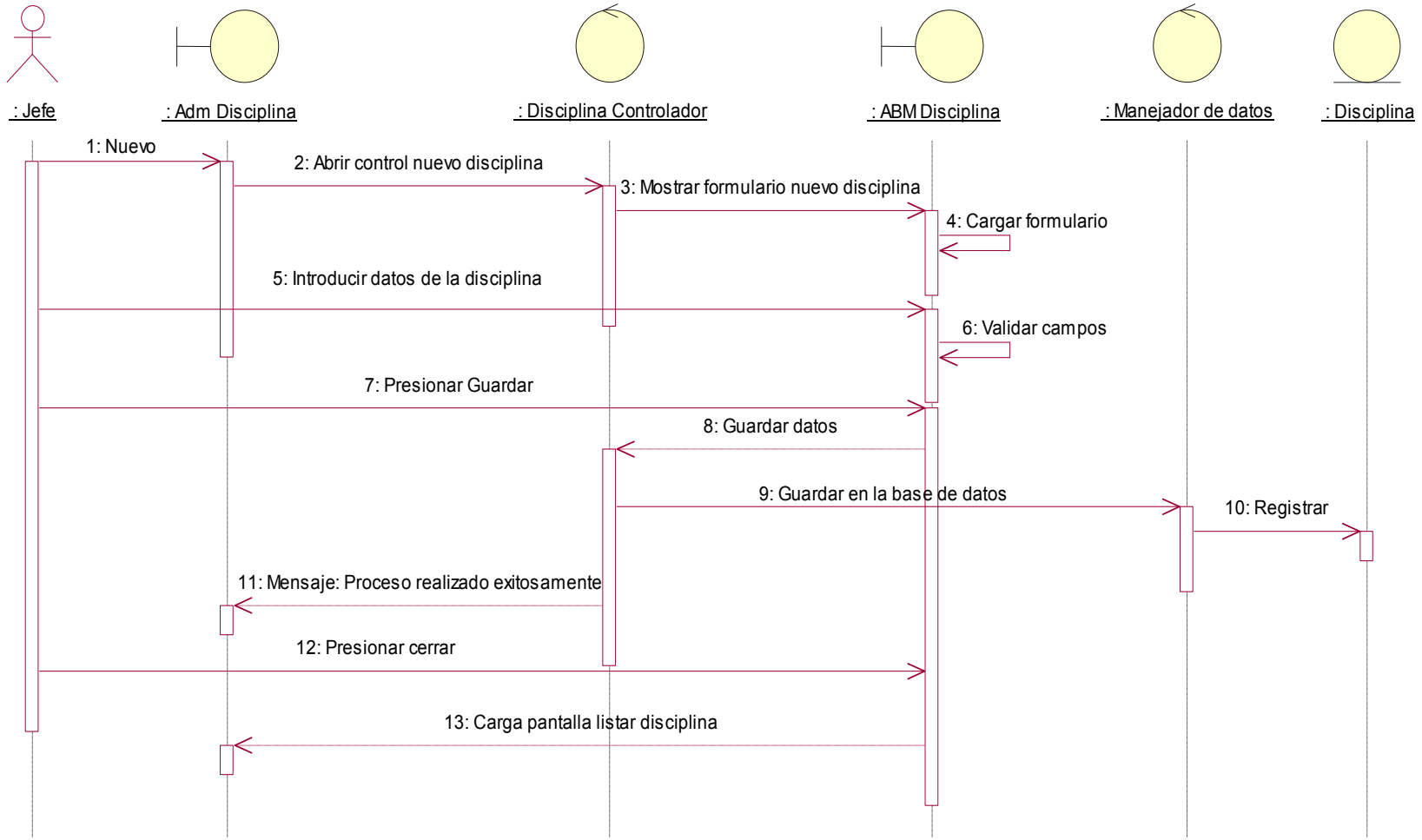


Figura 99 Diagrama de Secuencia Nuevo Disciplina

Modificar Disciplina

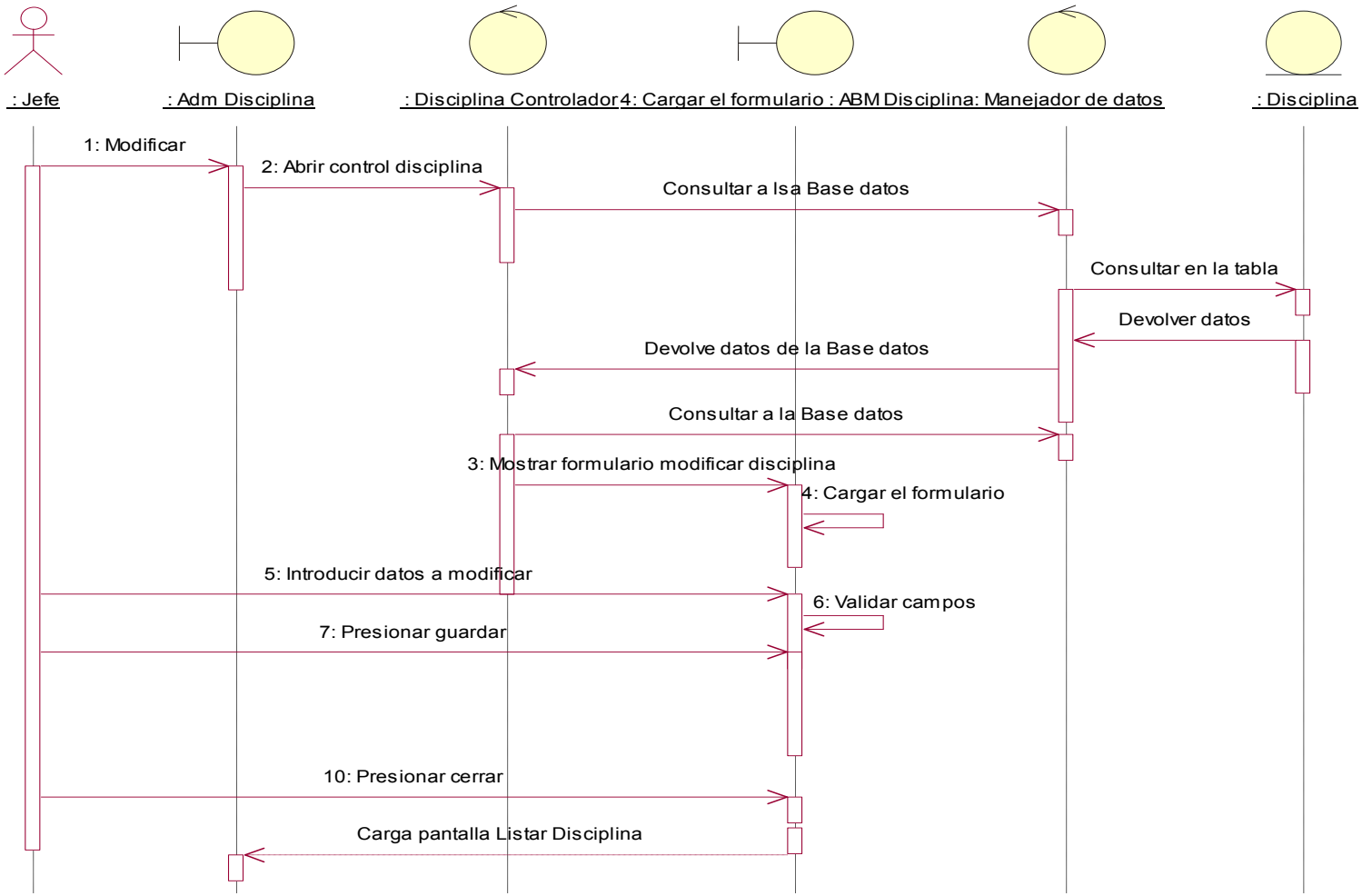


Figura 100 **Diagrama de Secuencia Modificar Disciplina**

Estado Disciplina

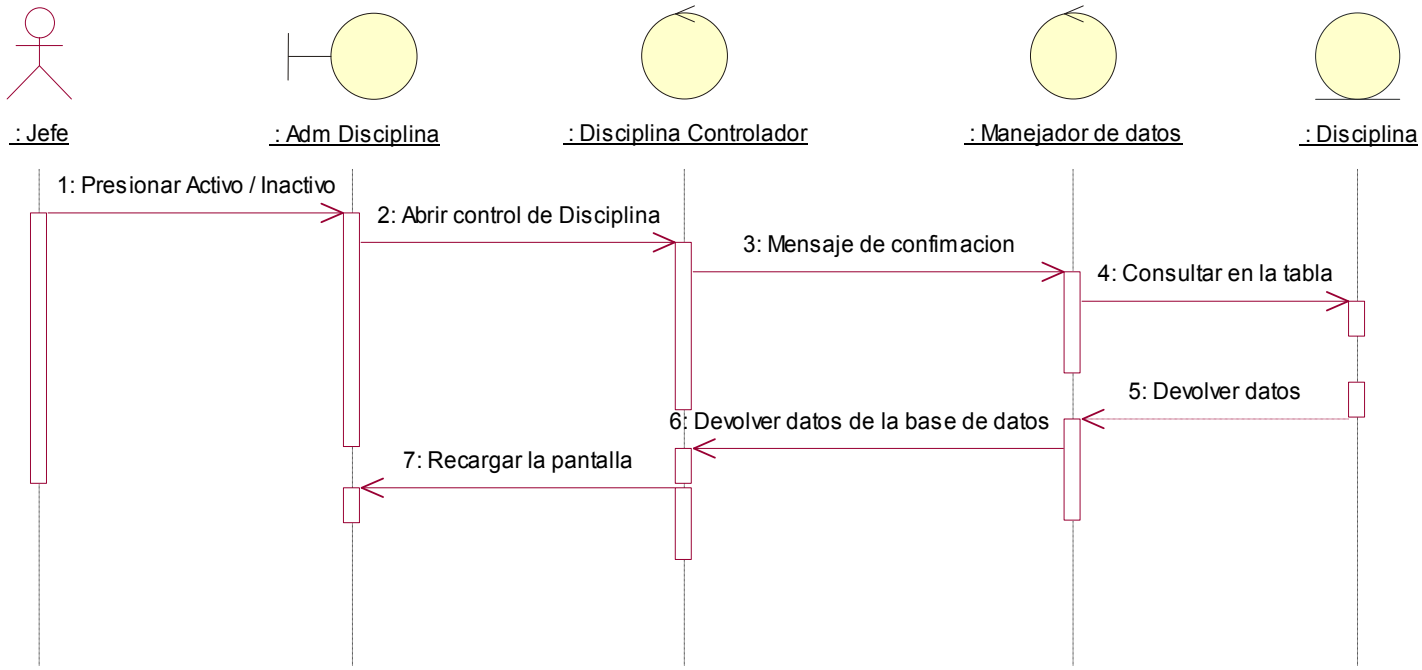


Figura 101 **Diagrama de Secuencia Estado Disciplina**

Nuevo Barrio

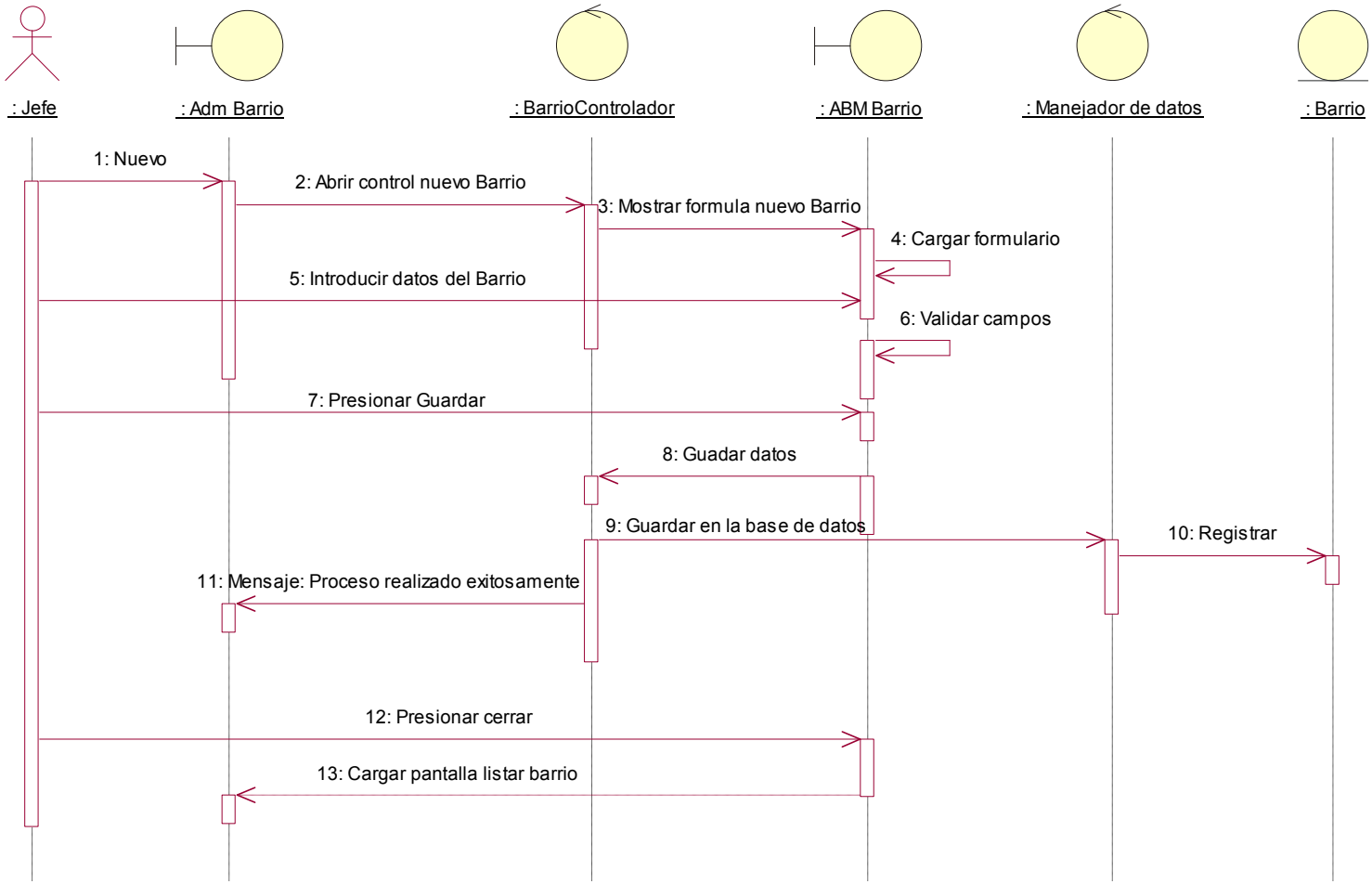


Figura 102 Diagrama de Secuencia Nuevo Barrio

Modificar Barrio

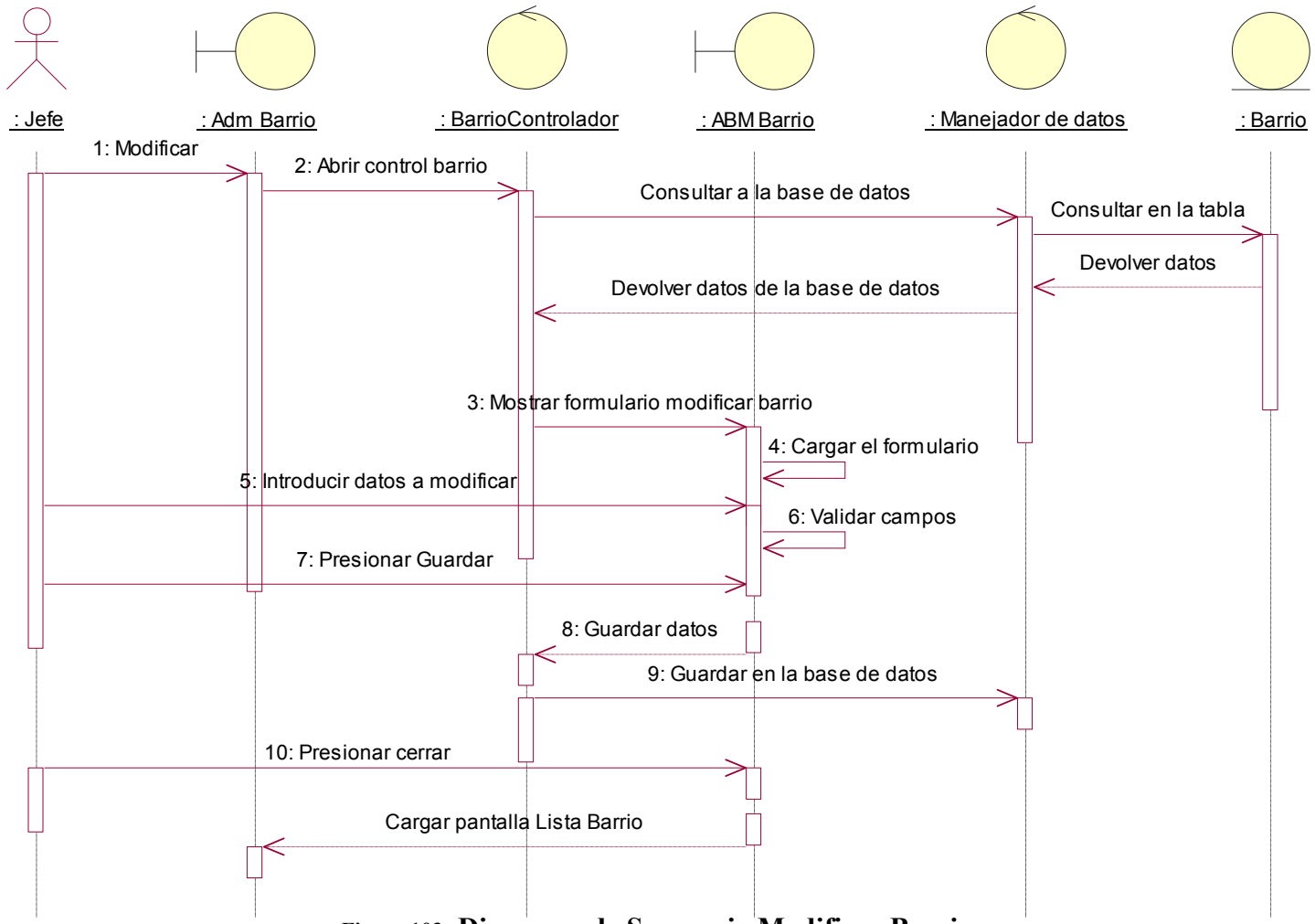


Figura 103 Diagrama de Secuencia Modificar Barrio

Estado Barrio

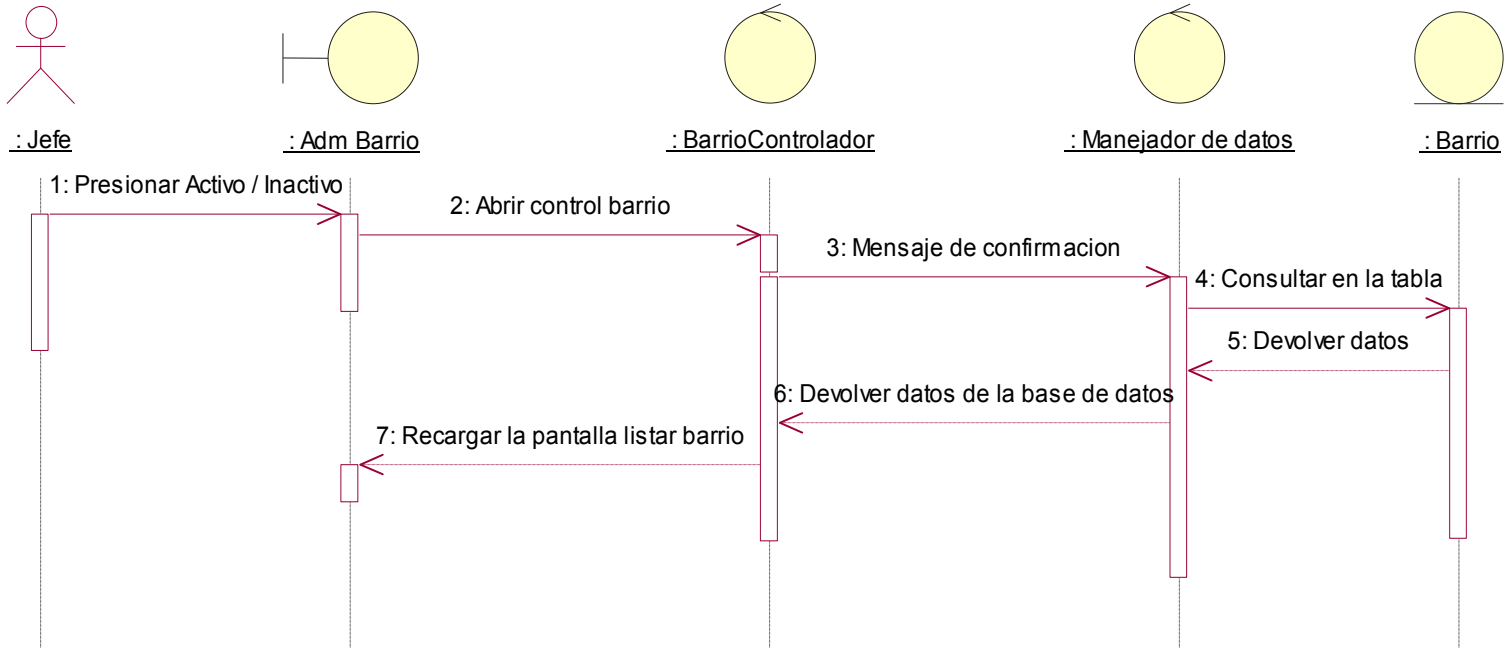


Figura 104 **Diagrama de Secuencia Estado Barrio**

Nuevo Categoría

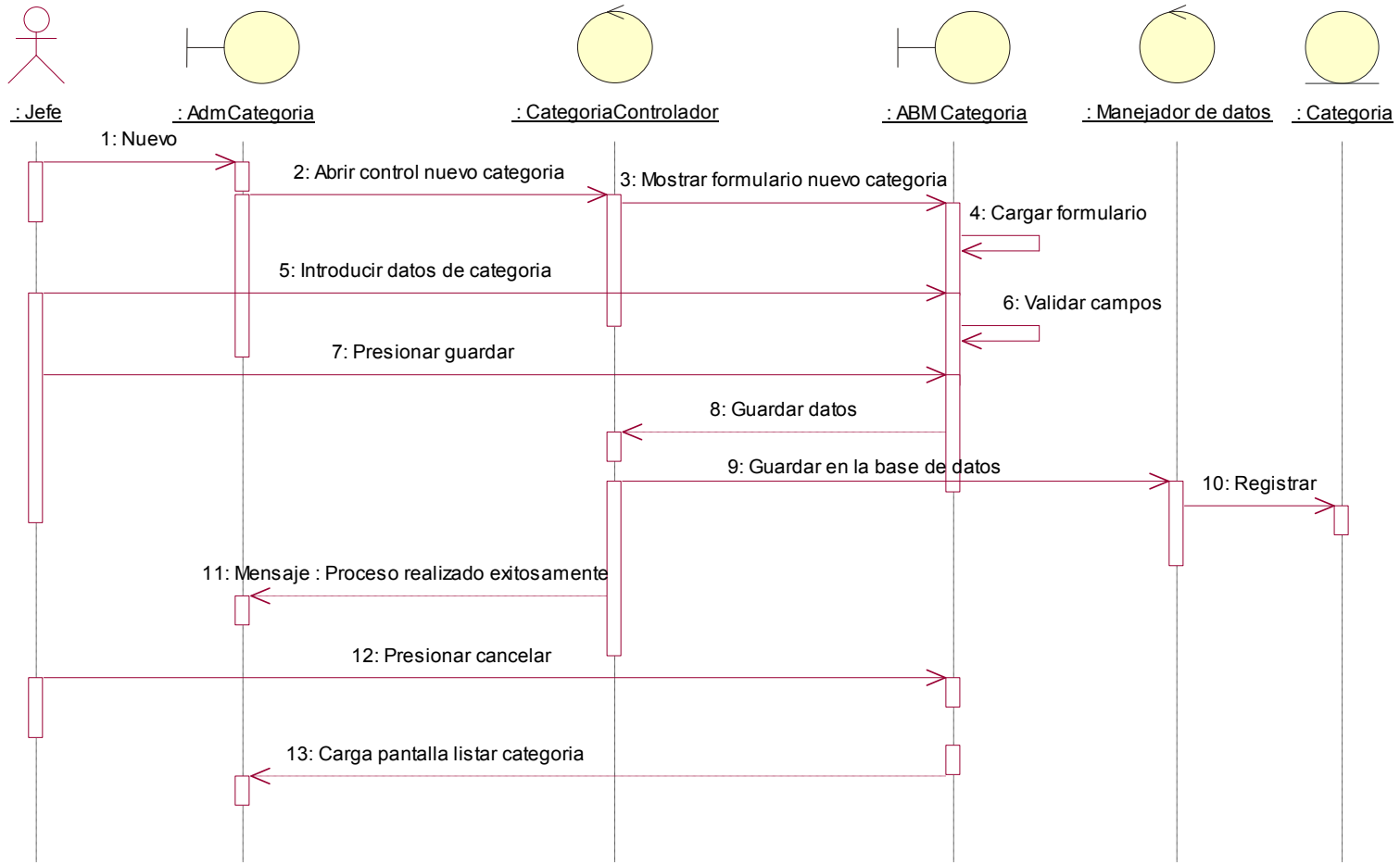


Figura 105 Diagrama de Secuencia Nuevo Categoría

Modificar Categoría

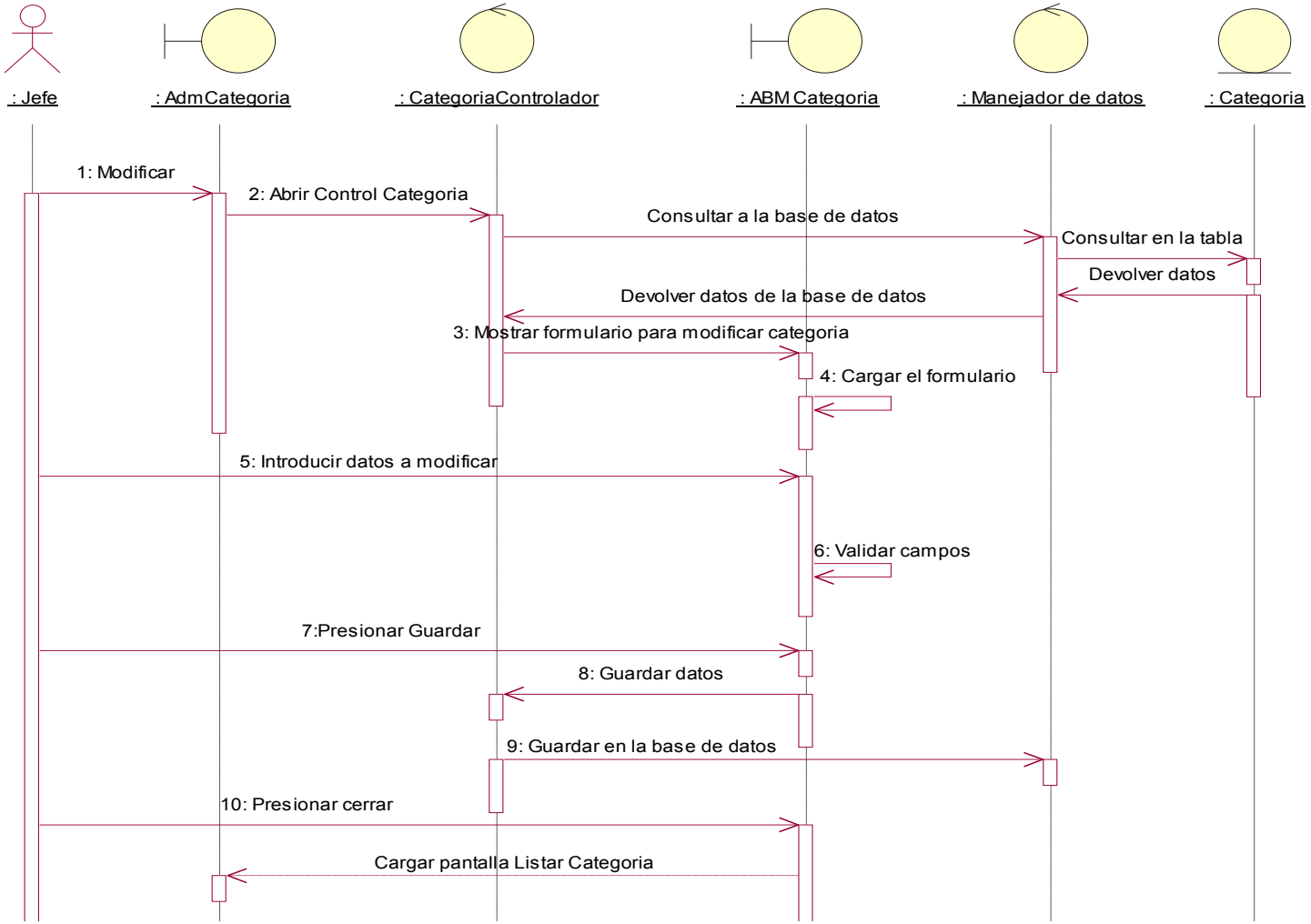


Figura 106 Diagrama de Secuencia Modificar Categoría

Estado Categoría

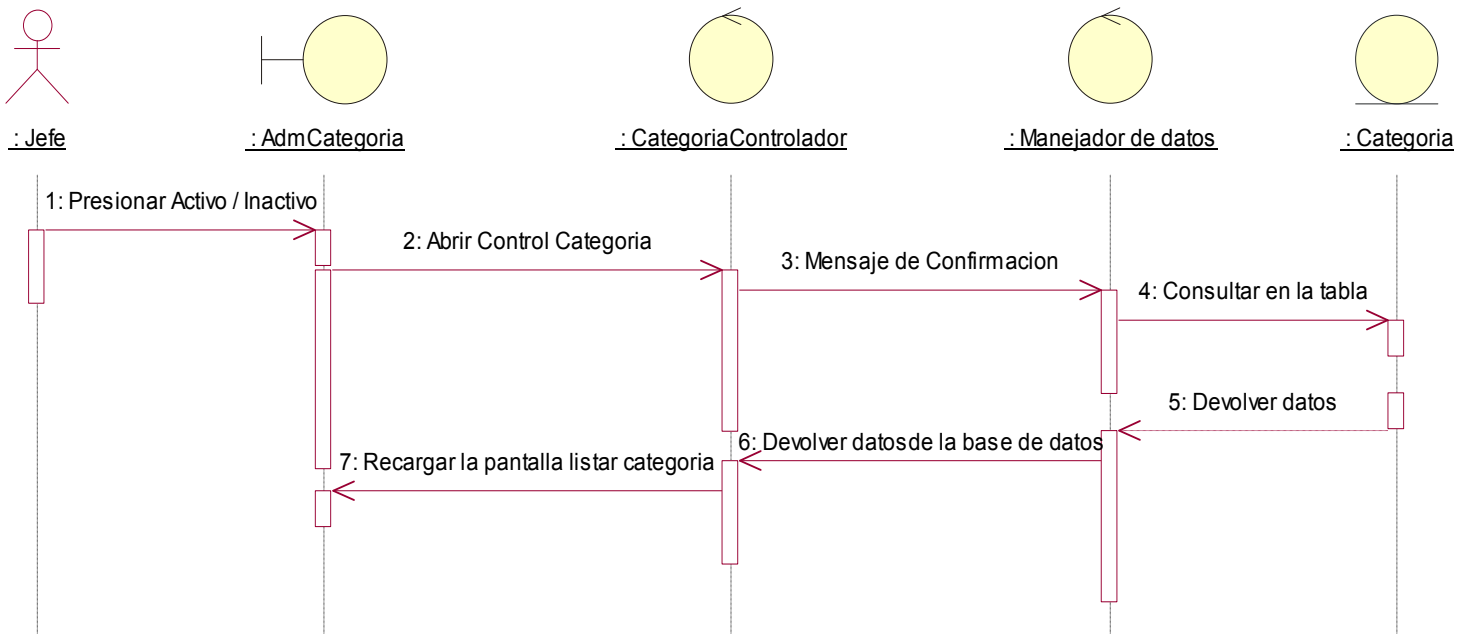


Figura 107 **Diagrama de Secuencia Estado Categoría**

Nuevo Equipo

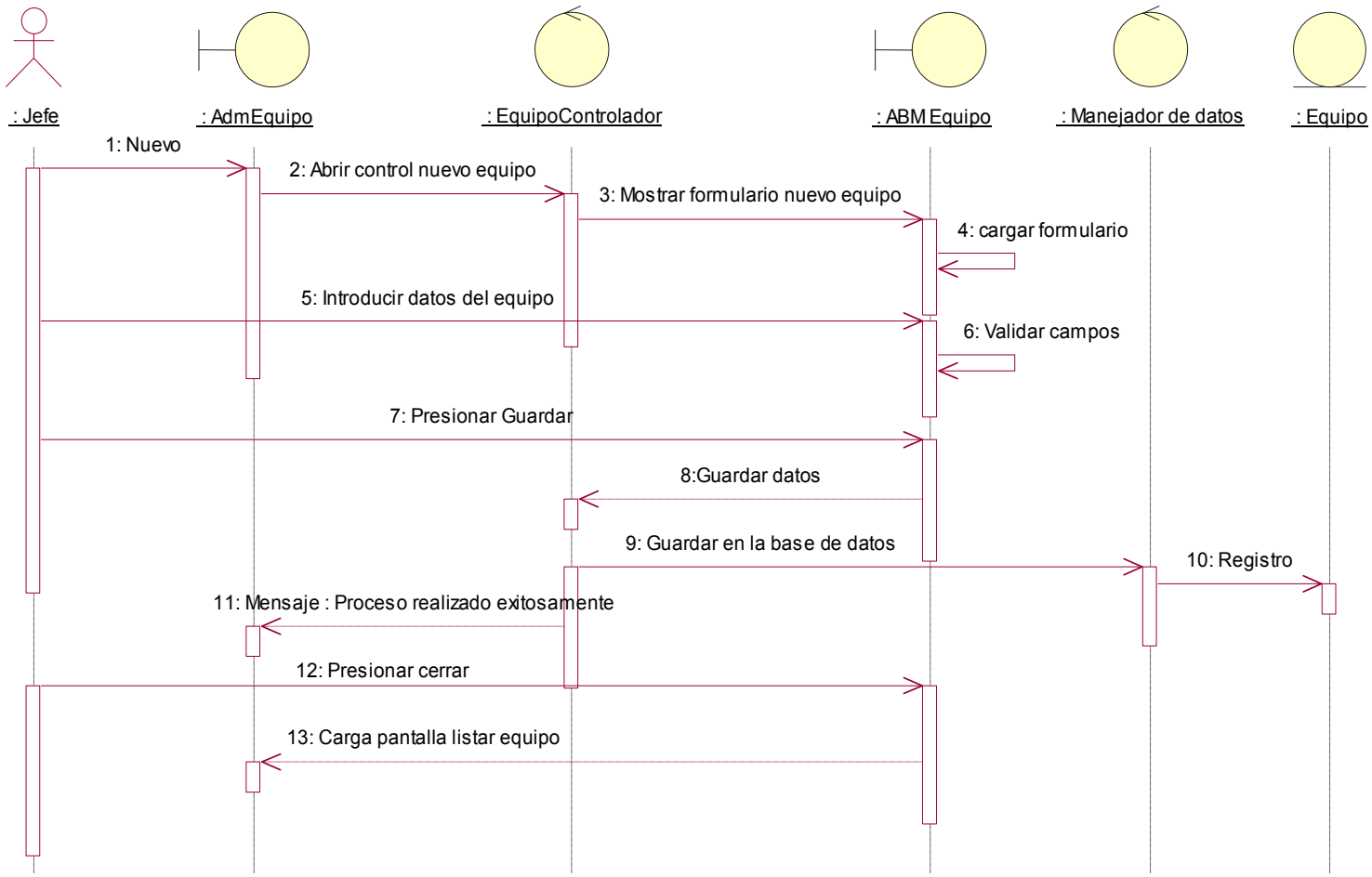


Figura 108 Diagrama de Secuencia Nuevo Equipo

Modificar Equipo

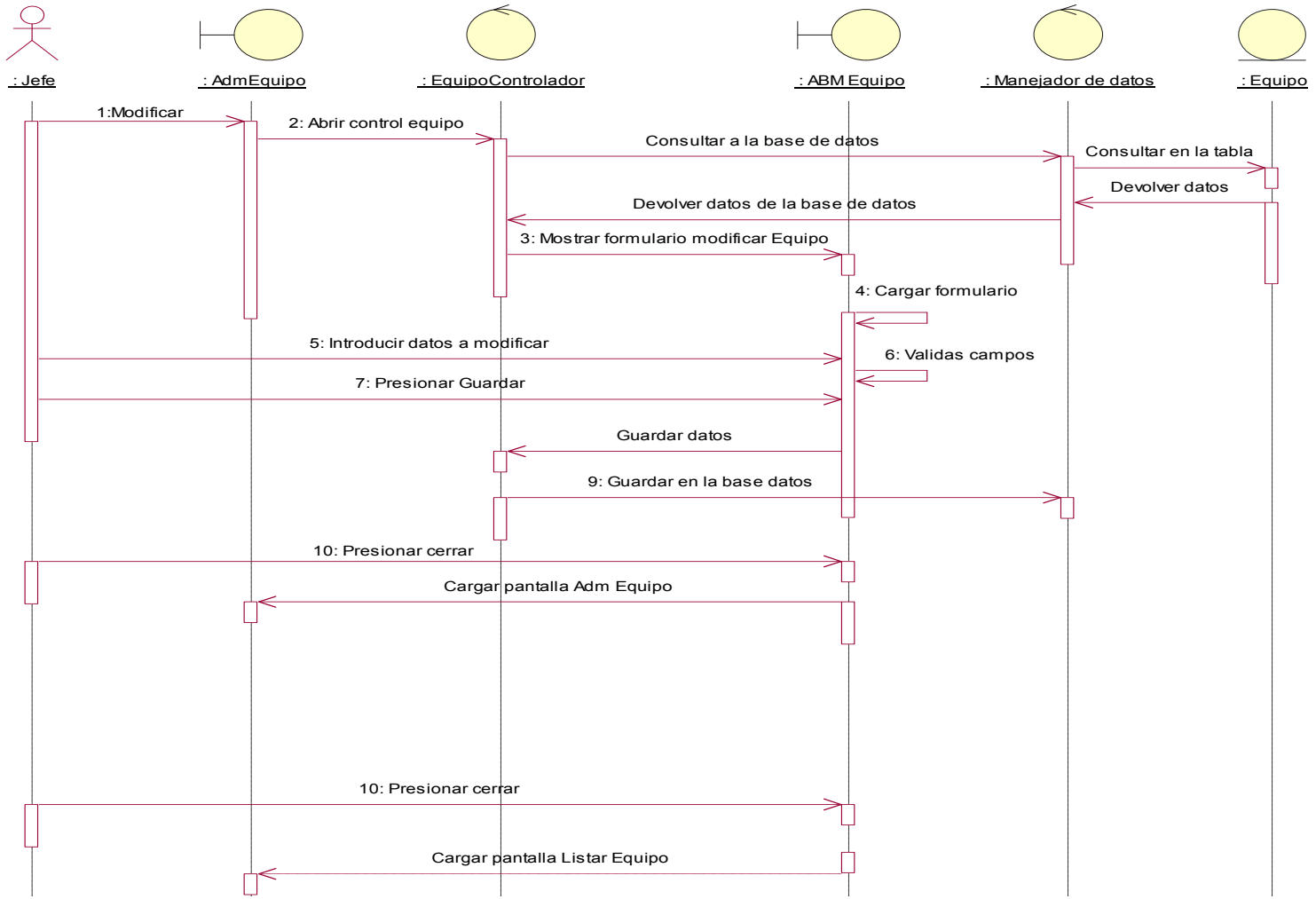


Figura 109 Diagrama de Secuencia Modificar Equipo

Estado Equipo

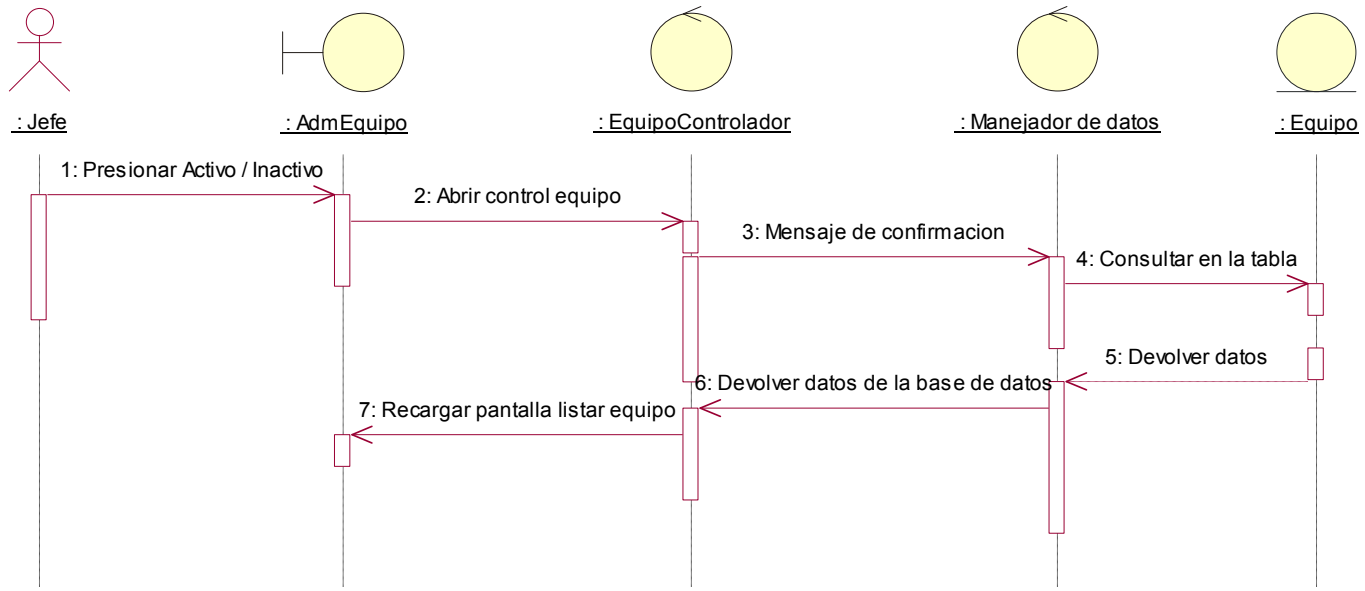


Figura 110 **Diagrama de Secuencia Estado Equipo**

Nuevo Serie

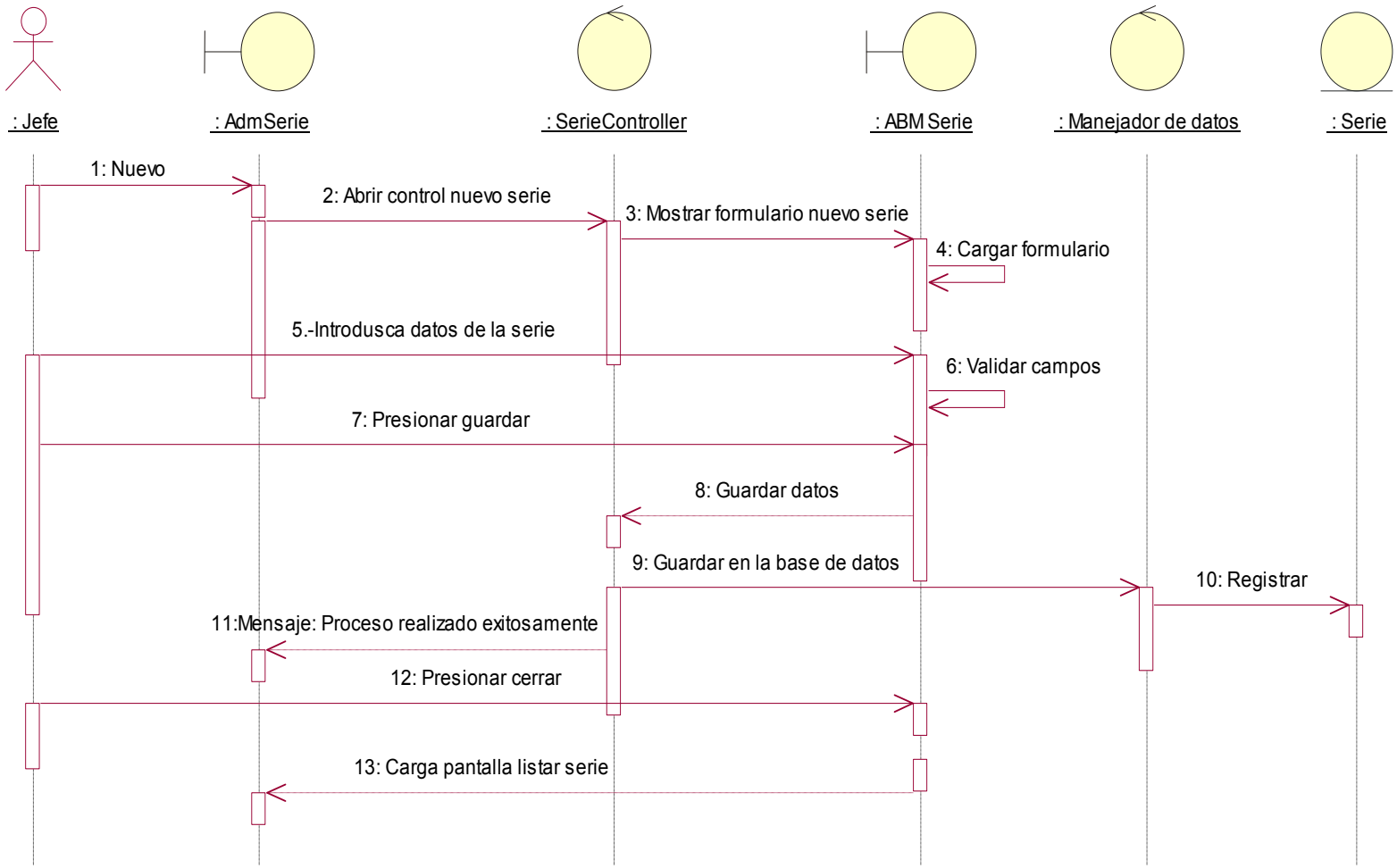


Figura 111 **Diagrama de Secuencia Nuevo Serie**

Modificar Serie

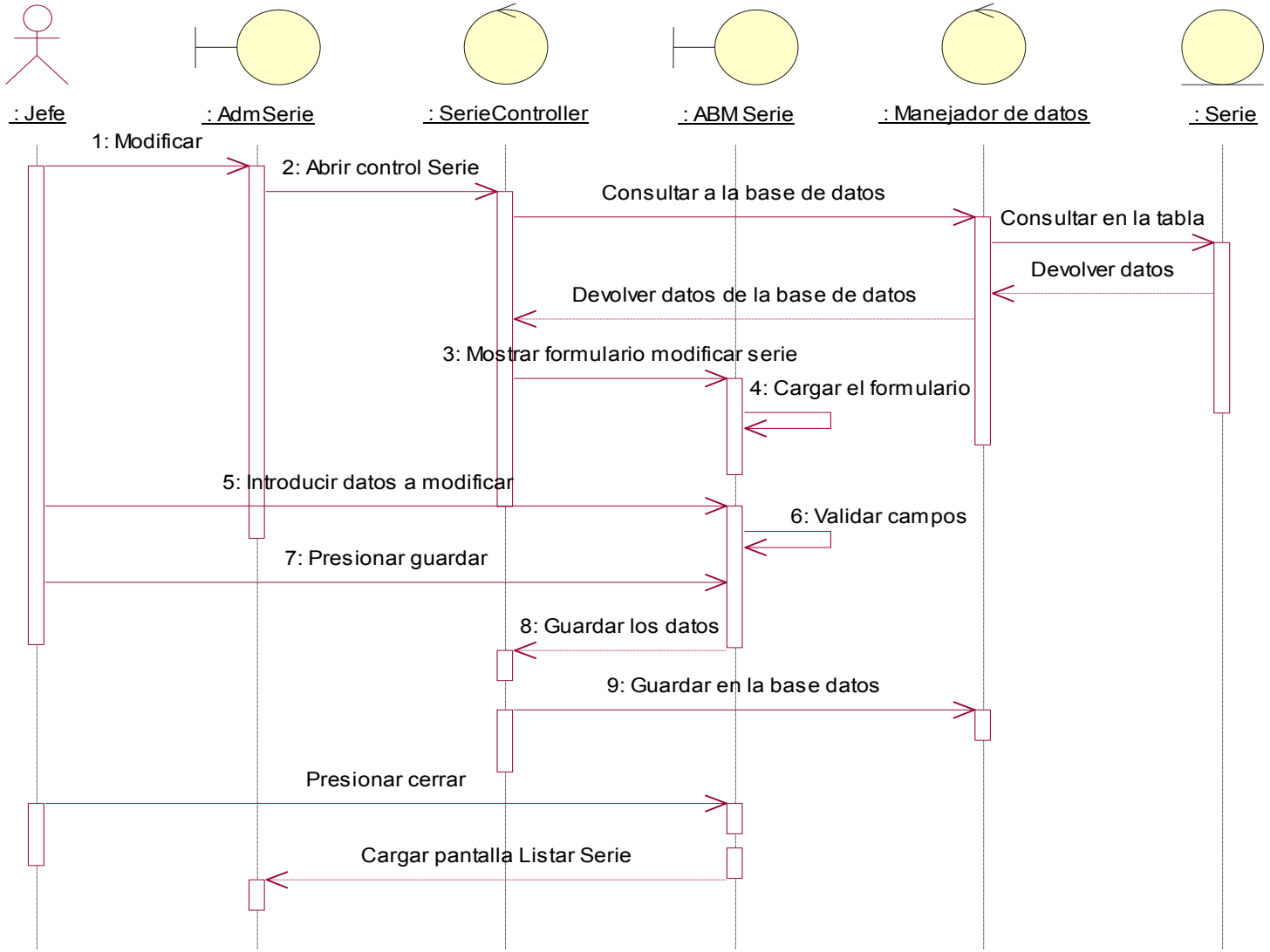


Figura 112 Diagrama de Secuencia Modificar Serie

Nuevo Campeonato

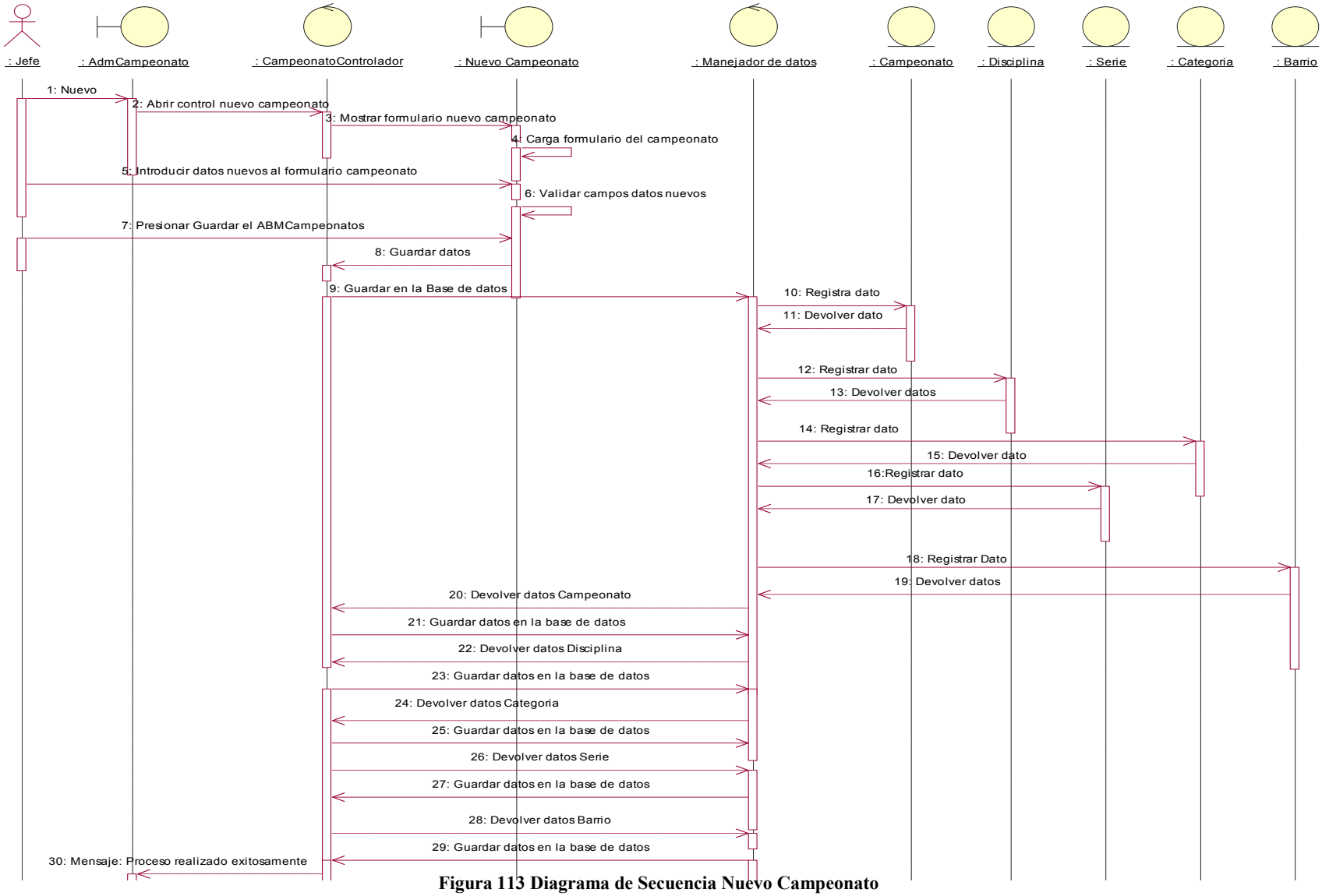


Figura 113 Diagrama de Secuencia Nuevo Campeonato

Modificar Campeonato

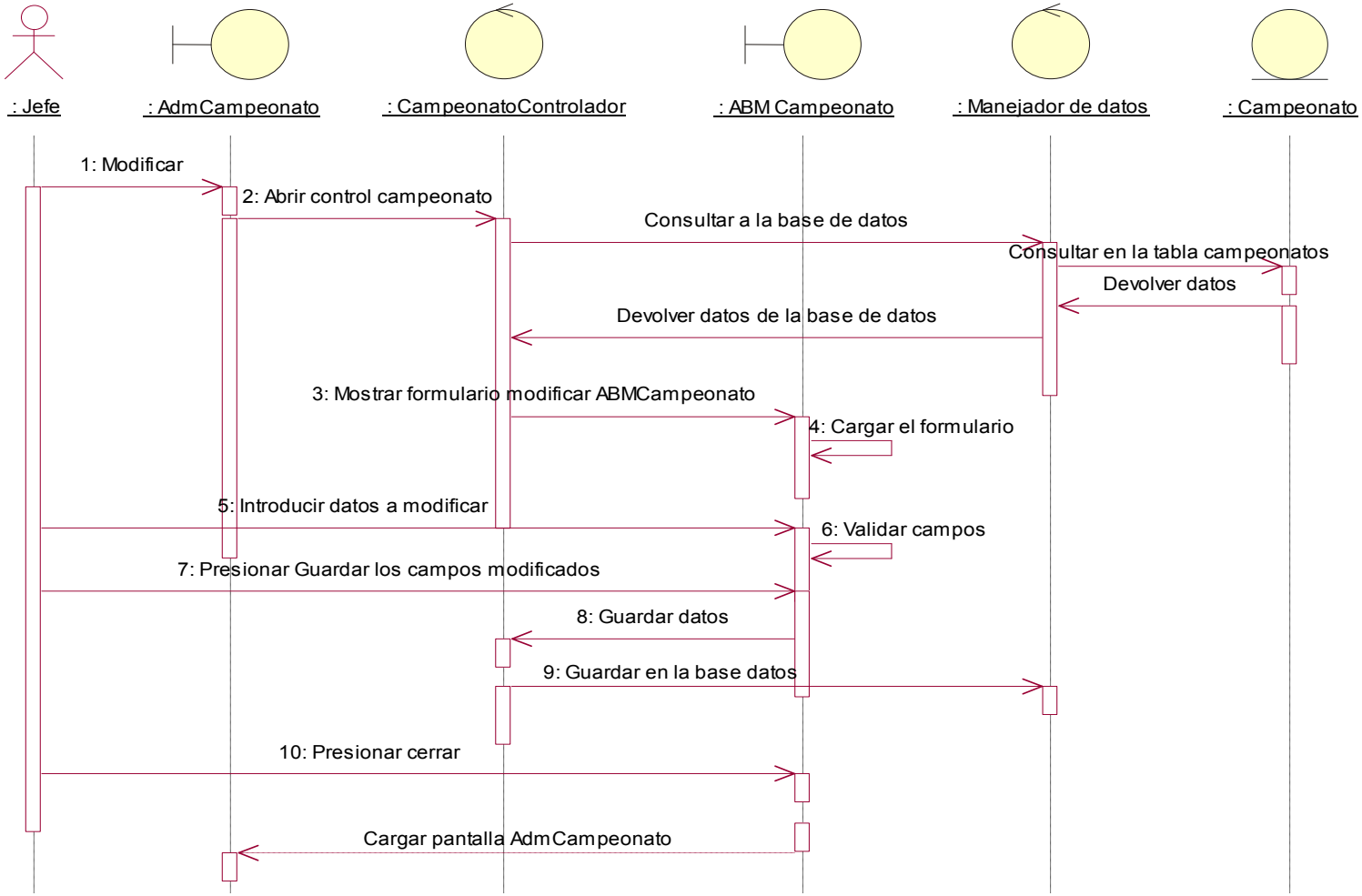


Figura 114 Diagrama de Secuencia Modificar Campeonato

Nuevo Rol

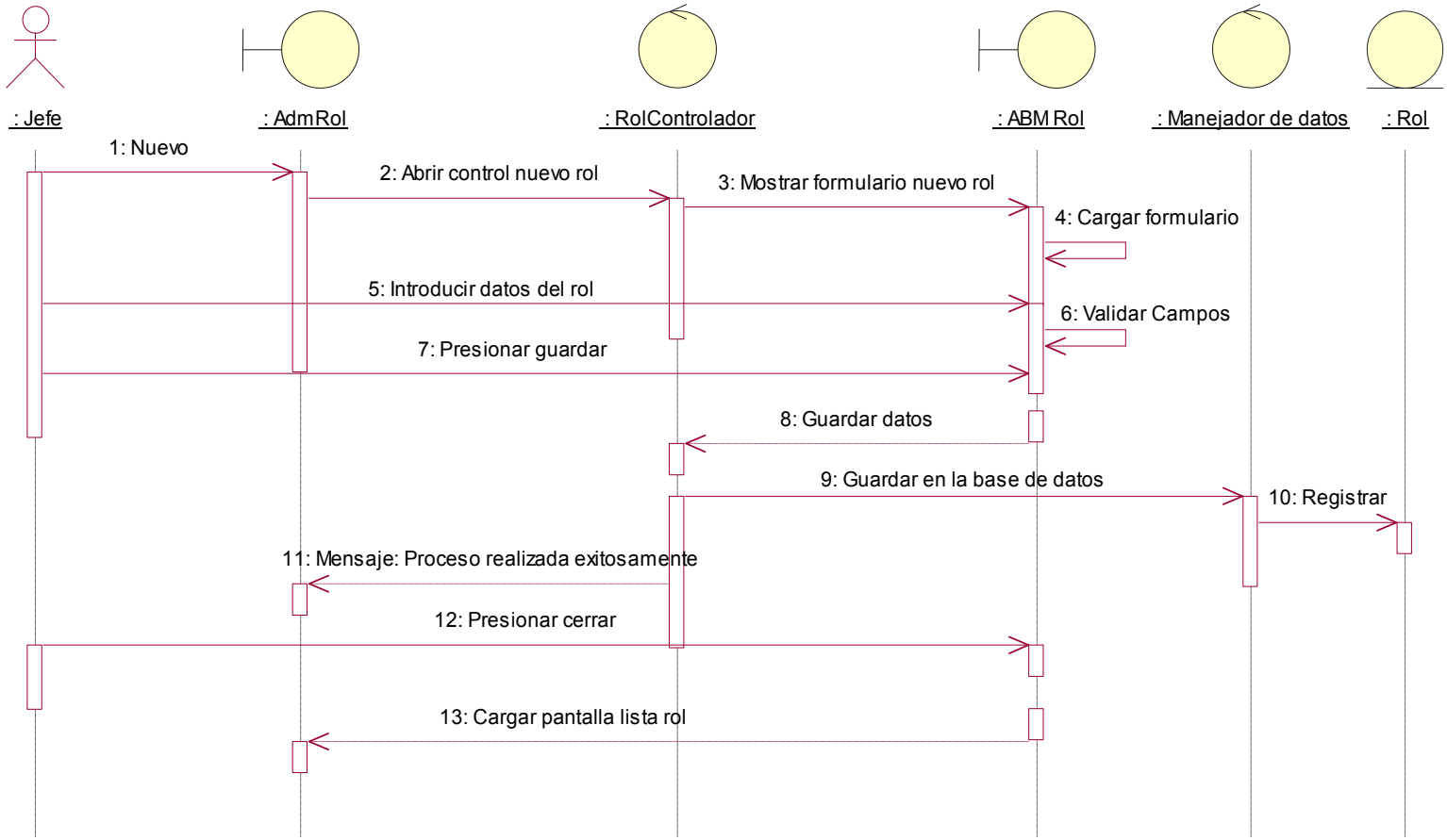


Figura 115 Diagrama de Secuencia Nuevo Rol

Modificar Rol

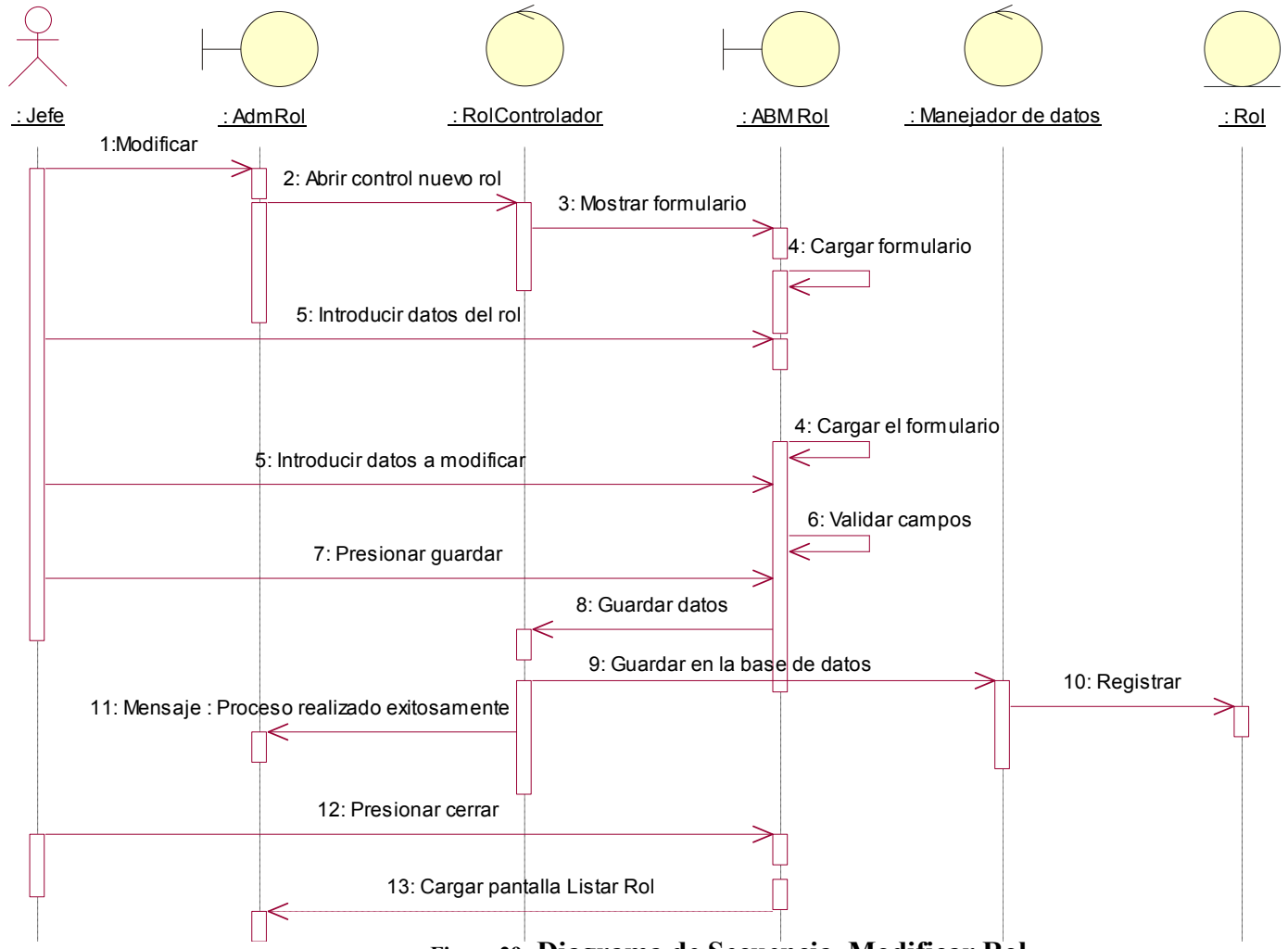


Figura 29 Diagrama de Secuencia Modificar Rol

Estado Rol

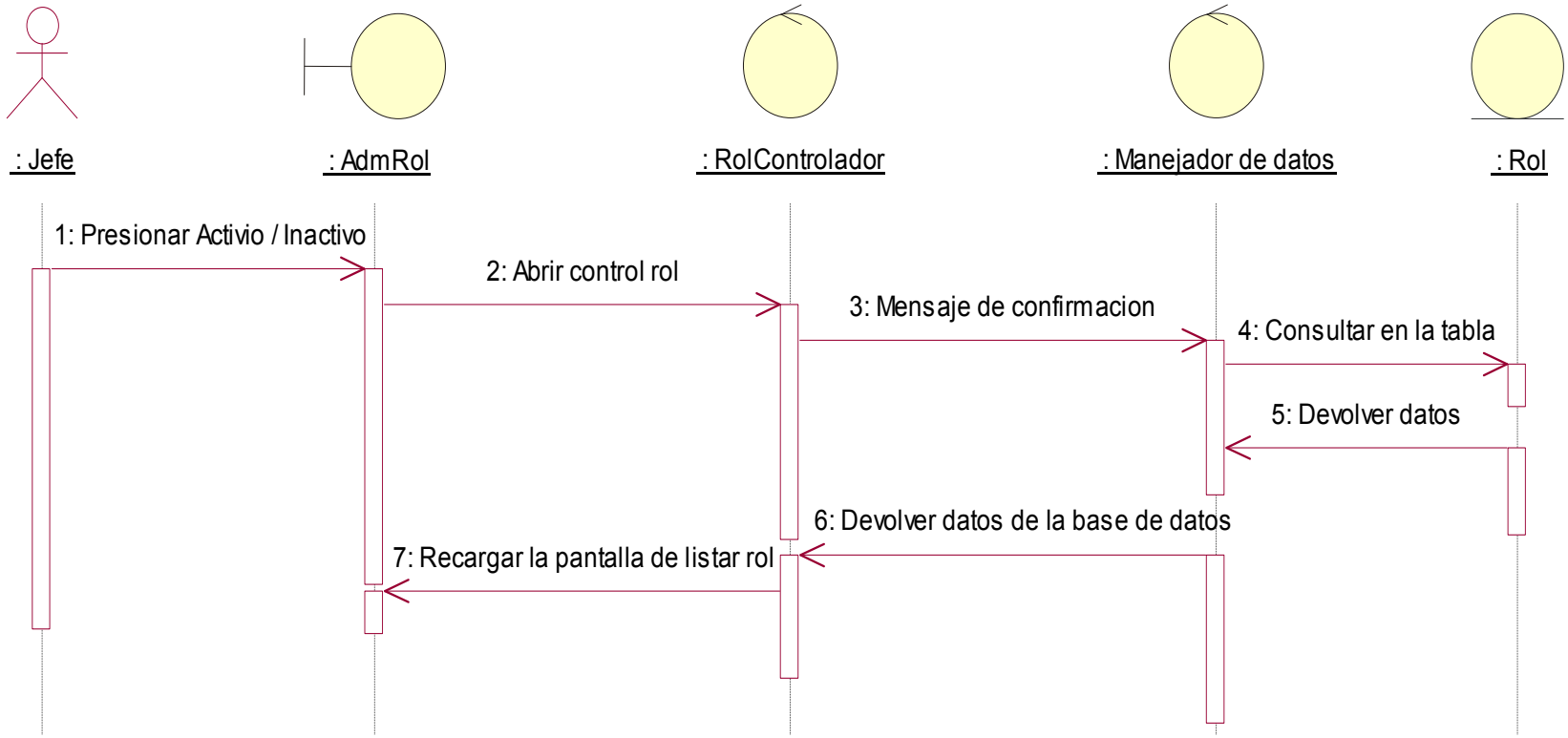


Figura 117 **Diagrama de Secuencia Estado Rol**

Nuevo Publicaciones

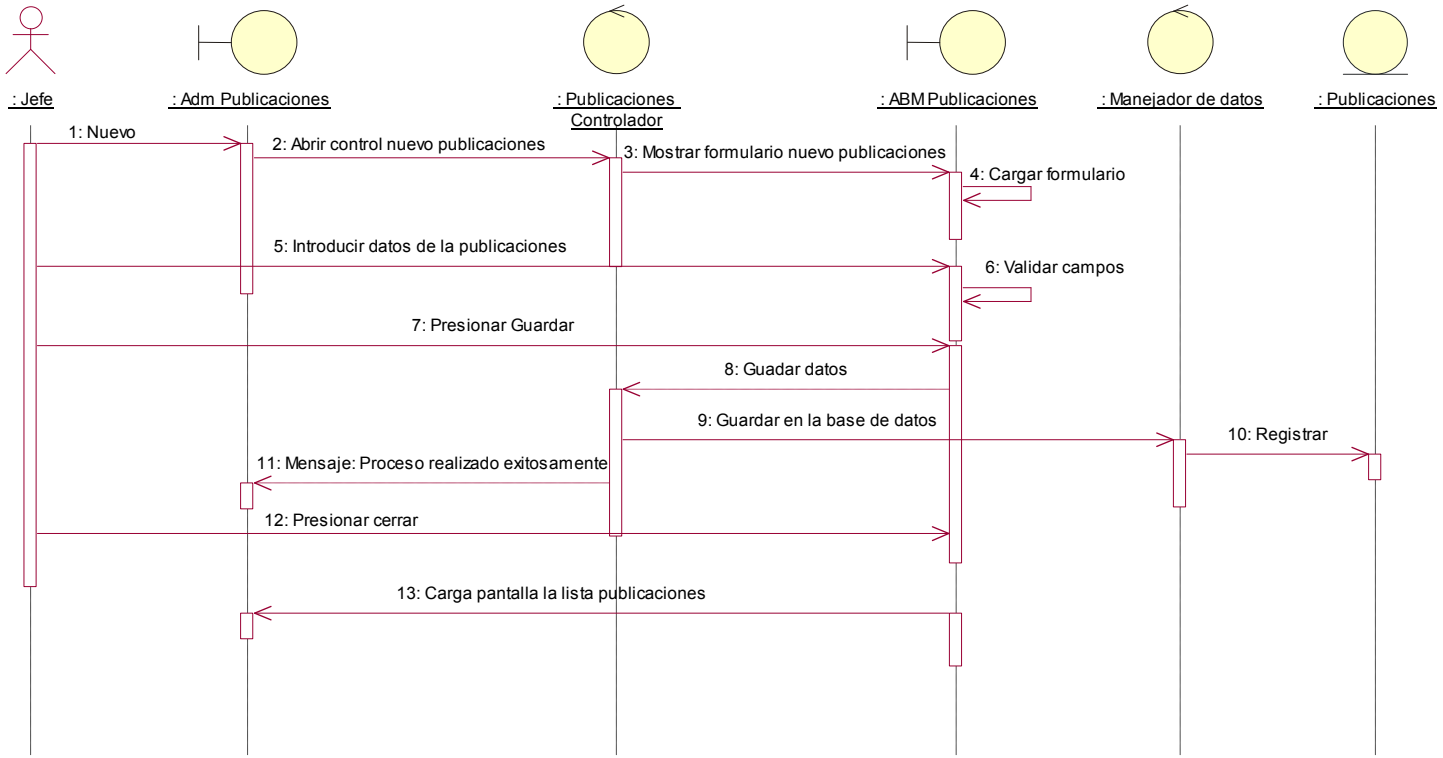


Figura 118 **Diagrama de Secuencia Nuevo Publicaciones**

Modificar Publicaciones

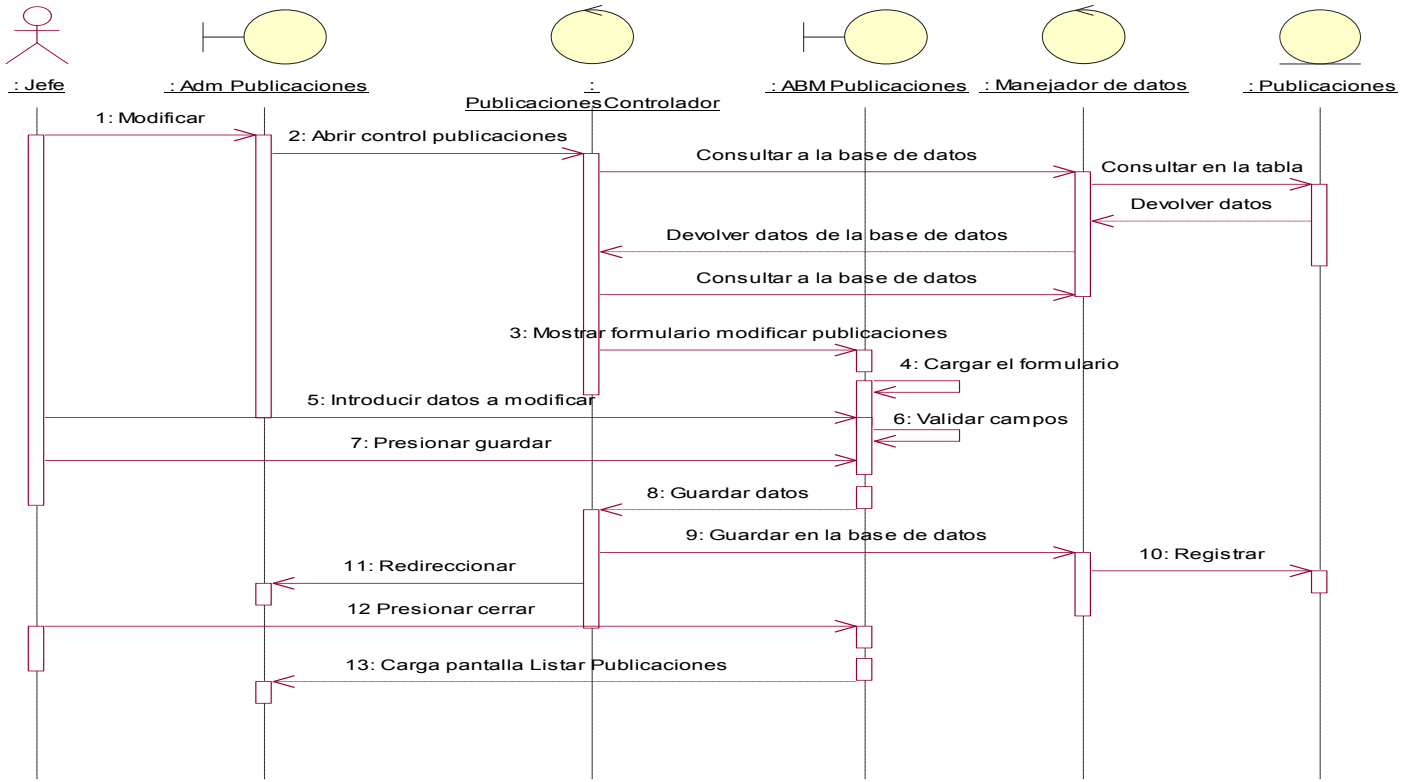


Figura 119 **Diagrama de Secuencia Modificar Publicaciones**

Nuevo Foto

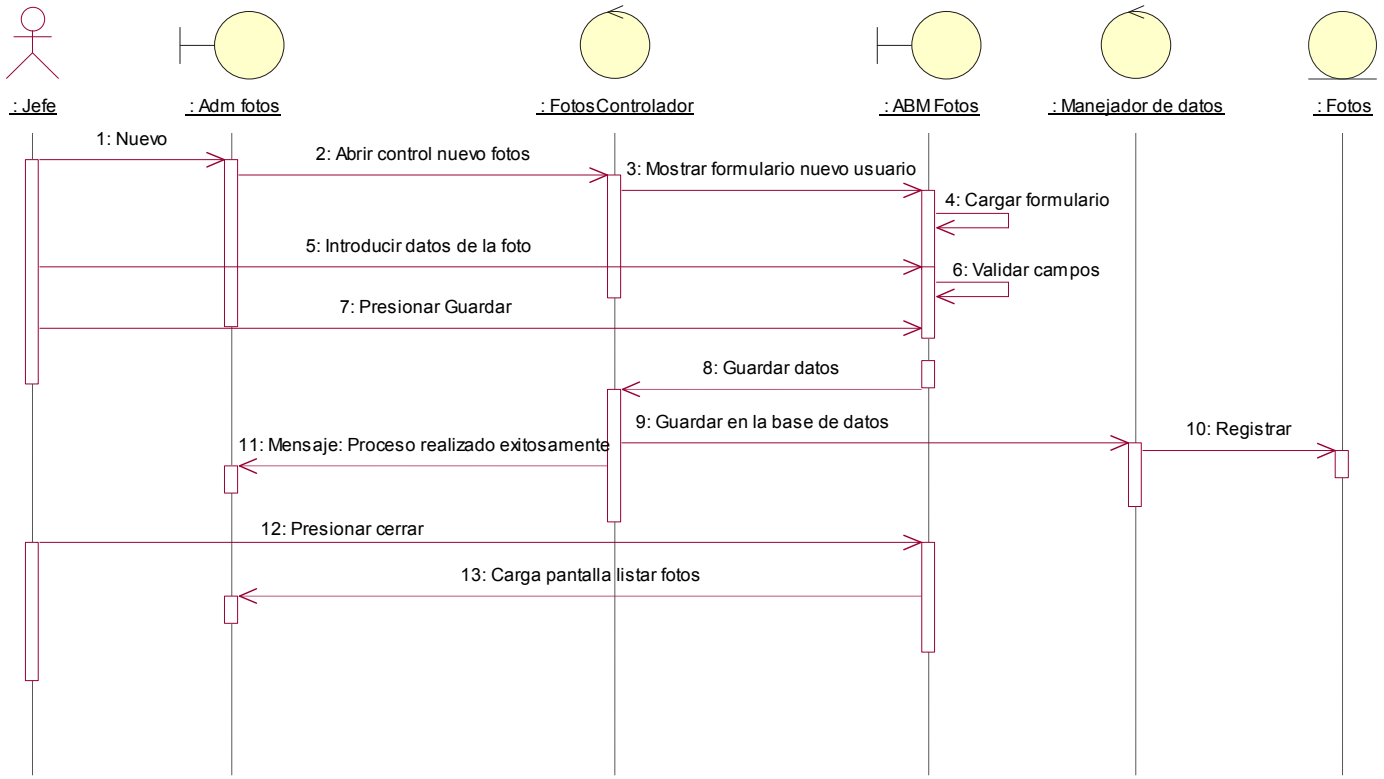


Figura 120 **Diagrama de Secuencia Nuevo Foto**

Modificar Foto

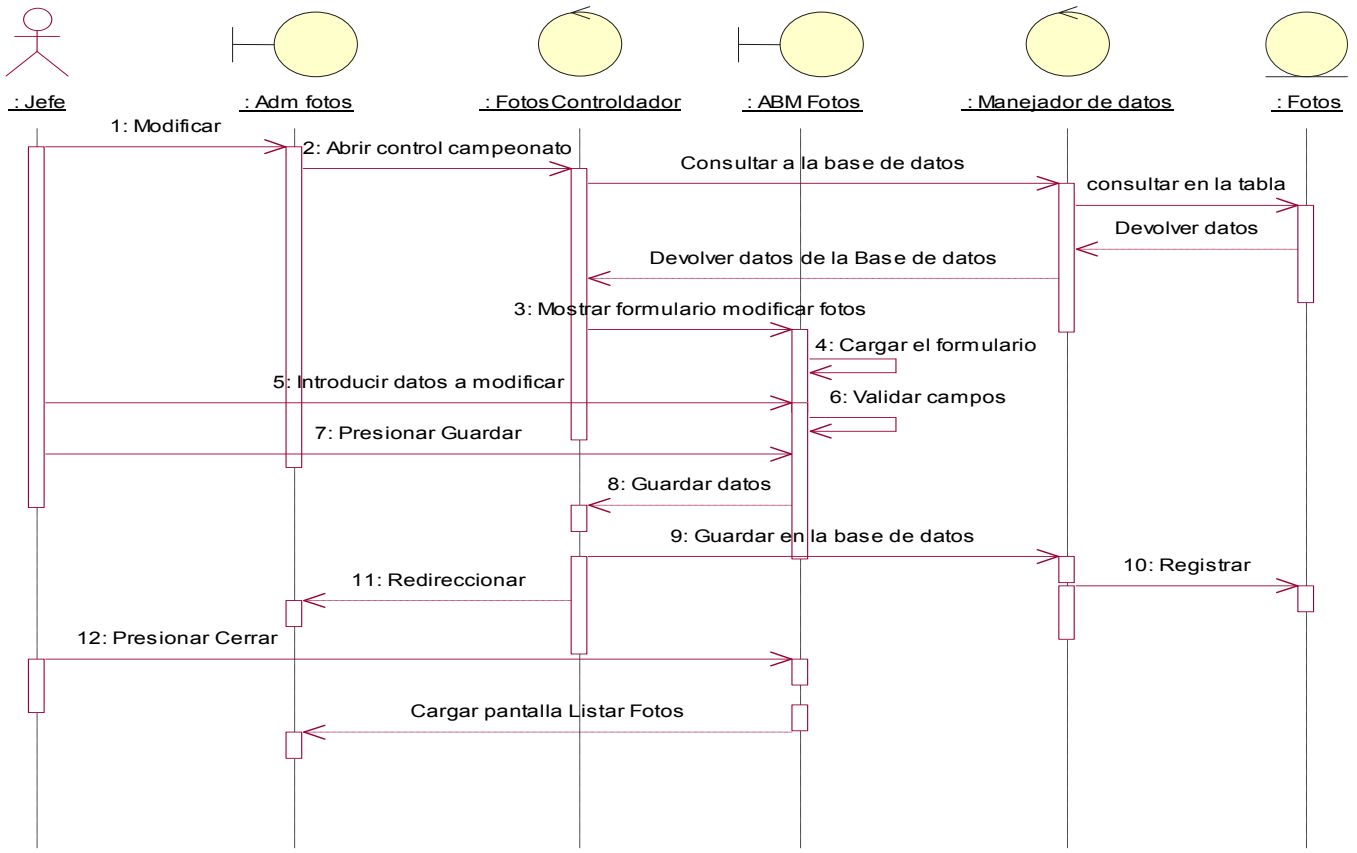


Figura 121 Diagrama de Secuencia Modificar Foto

Estado Fotos

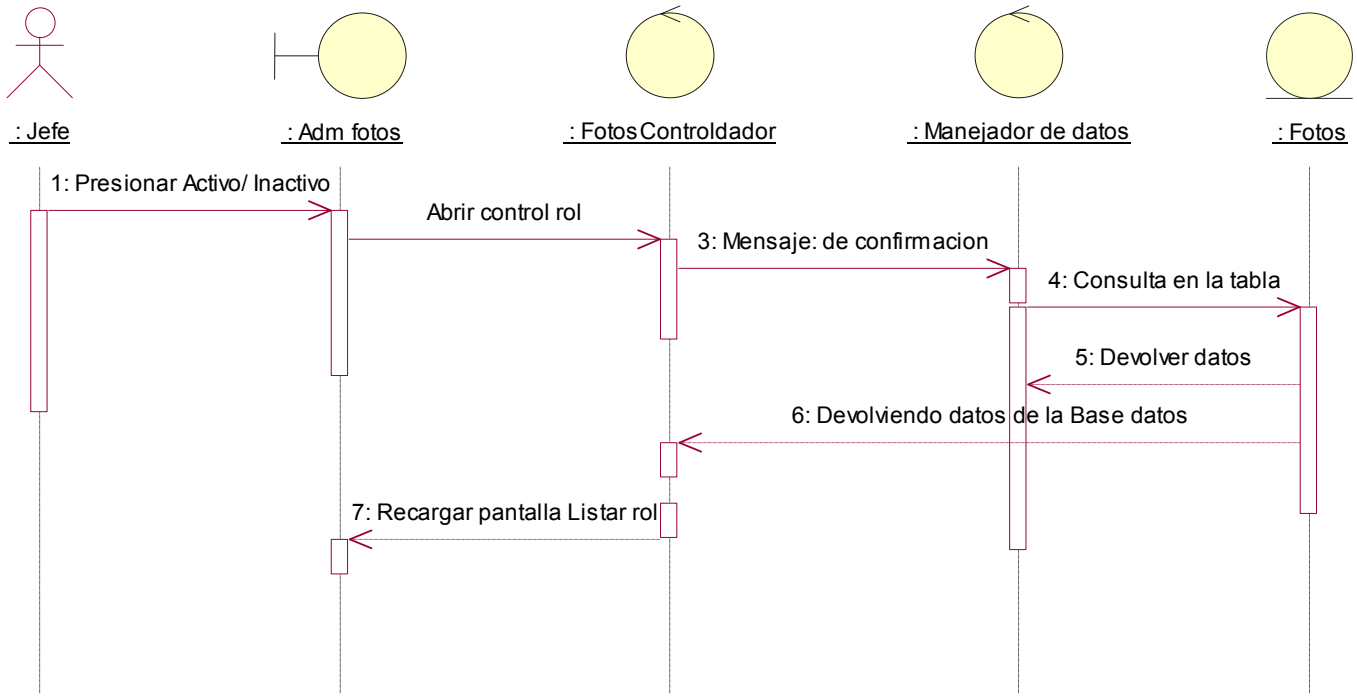


Figura 122 **Diagrama de Secuencia Estado Fotos**

I.1.3 Modelado de Diagrama de Clases

I.1.3.1 Introducción

El modelado de diagrama de clases es un artefacto de la disciplina análisis diseño en la metodología RUP la cual estamos implementando.

I.1.3.2 Propósito

- Comprender la estructura del sistema deseado para organización.
- Identificar clases de análisis y diseño.

I.1.3.3 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 por la organización.

I.1.4 Modelado de Datos

I.1.4.1 Introducción

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

I.1.4.2 Propósito

- Comprender la estructura de las tablas y sus campos, en la base de datos de nuestro sistema deseado para la organización.
- Identificar los tipos de campos de cada tabla de la base de datos

I.1.4.3 Alcance

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

I.1.1.1 Descripción de los campos para la base de Datos

Tabla: Alumnos

Column name	Primary key	Data type	Not NULL	Comment
codalum	Yes	INTEGER	Yes	Código de alumno
nombres	No	CHARACTER VARYING(40)	Yes	Nombre de alumno
ap	No	CHARACTER VARYING(30)	Yes	Apellido materno de alumno
am	No	CHARACTER VARYING(40)	No	Apellido paterno de alumno
fechanac	No	DATE	No	Fecha de nacimiento
celular	No	INTEGER	No	Celular del alumno
estado	No	INTEGER	No	Estado
fechaini	No	DATE	No	Fecha de registro del alumno
edad	No	INTEGER	No	Edad del alumno
peso	No	INTEGER	No	Peso del estudiante
estatura	No	DOUBLE PRECISION	No	Estatura del estudiante

telefono	No	INTEGER	Yes	Teléfono
colegio	No	CHARACTER VARYING(40)	No	Colegio
curso	No	CHARACTER VARYING(40)	No	Curso en el colegio
nombrepa	No	CHARACTER VARYING(40)	No	Nombre del padre
nombrema	No	CHARACTER VARYING(40)	No	Nombre de la madre
nombreapo	No	CHARACTER VARYING(40)	No	Nombre del apoderado
profesionpa	No	CHARACTER VARYING(40)	No	Profesión del padre
profesionma	No	CHARACTER VARYING(40)	No	Profesión de la madre
profesionapo	No	CHARACTER VARYING(40)	No	Profesión del apoderado
teltrabajoma	No	INTEGER	No	Teléfono de la madre
teltrabajopa	No	INTEGER	No	Teléfono del padre
teltrabajoap	No	INTEGER	No	Teléfono del apoderado
coddir	No	INTEGER	Yes	Código de dirección del alumno

codprov	No	INTEGER	Yes	Código de provincial de la provincial del alumno
sexo	No	CHARACTER VARYING(40)	Yes	Sexo del alumno
code	No	INTEGER	No	Código del equipo al que pertenece
coddis	No	INTEGER	No	Código de la disciplina
codcat	No	INTEGER	No	Código de la categoría

Tabla 66 **Descripción de los campos de alumnos**

Tabla: barrio

Column name	Primary key	Data type	Not NULL	Comment
codbarrio	Yes	INTEGER	Yes	Código del barrio
nombre	No	CHARACTER VARYING(40)	Yes	Nombre del barrio
estado	No	INTEGER	Yes	Estado del barrio

Tabla 37 **Descripción de los campos de barrio**

Tabla: barrio_campeonato

Column name	Primary key	Data type	Not NULL	Comment
Codbarrio	Yes	INTEGER	Yes	Código del barrio
Codcam	Yes	INTEGER	Yes	Código de campeonato

Tabla 6738 Descripción de los campos de barrio_campeonato

Tabla: campeonato

Column name	Primary key	Data type	Not NULL	Comment
Codcam	Yes	INTEGER	Yes	Código de campeonato
Nombre	No	CHARACTER VARYING(40)	Yes	Nombre del campeonato
Fechaini	No	DATE	Yes	Fecha de inicio campeonato
Fechafin	No	DATE	Yes	Fecha de fin del campeonato
estado	No	INTEGER	Yes	Estado del campeonato
coddis	No	INTEGER	Yes	Código de disciplina
codcat	No	INTEGER	No	Código de categoría

Tabla 69 Descripción de los campos de campeonato

Tabla: campeonato_serie

Column name	Primary key	Data type	Not NULL	Comment
idcamserie	Yes	INTEGER	Yes	Id del campeonato y serie
codcam	No	INTEGER	Yes	Código de campeonato
codserie	No	INTEGER	Yes	Código de la serie
estado	No	INTEGER	No	Estado

Tabla 7039 Descripción de los campos de campeonato_serie

Tabla: campeonato_serie_equipo

Column name	Primary key	Data type	Not NULL	Comment
idcamserie	Yes	INTEGER	Yes	Id de campeonato y serie
code	Yes	INTEGER	Yes	Código del equipo
Puntos	No	INTEGER	Yes	Puntos del equipo en el campeonato
Goles	No	INTEGER	Yes	Goles del equipo en el campeonato
Partidos	No	INTEGER	Yes	Partidos jugados

Partidos	No	INTEGER	Yes	Partidos empatados
Partidos	No	INTEGER	Yes	Partidos perdidos
Partidos	No	INTEGER	Yes	Partidos ganador
goles contra	No	INTEGER	Yes	Goles en contra

Tabla 71 **Descripción de los campos de campeonato_serie_equipo**

Tabla: categoría

Column name	Primary key	Data type	Not NULL	Comment
codcat	Yes	INTEGER	Yes	Código de la categoría
nombre	No	CHARACTER VARYING(40)	Yes	Nombre de la categoría
estado	No	INTEGER	Yes	Estado de la categoría

Tabla 72 **Descripción de los campos de categoría**

Tabla: Curriculum

Column name	Primary key	Data type	Not NULL	Comment
cod_curr	Yes	INTEGER	Yes	Es el código del Curriculum.
estudios	No	CHARACTER VARYING(500)	Yes	Es el estudio que tiene el usuario.
deportista	No	CHARACTER VARYING(500)	Yes	Es el deporte que el usuario realiza.
cursos	No	CHARACTER VARYING(500)	Yes	Es los cursos que realizó el usuario.
entrenador	No	CHARACTER VARYING(500)	Yes	Como entrenador que disciplina entrena.

Tabla 73 Descripción de los campos de Curriculum

Tabla: datos

Column name	Primary key	Data type	Not NULL	Comment
iddatos	Yes	INTEGER	Yes	Es el código de datos.
login	No	CHARACTER VARYING(40)	Yes	Es el login del usuario con el cual podrá tener acceso al sistema.

clave	No	CHARACTER VARYING(40)	Yes	Es la clave con la cual el usuario para poder acceder al sistema.
idusuario	No	INTEGER	Yes	Es el código de los usuarios del sistema.

Tabla 74 Descripción de los campos de datos

Tabla: departamento

Column name	Primary key	Data type	Not NULL	Comment
coddep	Yes	INTEGER	Yes	Es el código de departamento
nombred	No	CHARACTER VARYING(40)	Yes	Es el nombre de Departamento
codpais	No	INTEGER	Yes	Es el código de País

Tabla 75 Descripción de los campos de departamento

Tabla: dirección

Column name	Primary key	Data type	Not NULL	Comment
coddir	Yes	INTEGER	Yes	Es el código de la dirección
barrio	No	CHARACTER VARYING(40)	Yes	Es el nombre del barrio
calle	No	CHARACTER VARYING(40)	Yes	Nombre de la calle
numerocasa	No	INTEGER	No	Es el número de la casa

Tabla 76 Descripción de los campos de dirección

Tabla: disciplina

Column name	Primary key	Data type	Not NULL	Comment
coddis	Yes	INTEGER	Yes	Es el código de la disciplina
nombredis	No	CHARACTER VARYING(40)	Yes	Nombre de la disciplina
estado	No	INTEGER	Yes	Estado activo o inactivo

Tabla 77 Descripción de los campos de disciplina

Tabla: disciplina_barrio

Column name	Primary key	Data type	Not NULL	Comment
coddis	Yes	INTEGER	Yes	Es el código de la disciplina
codbarrio	Yes	INTEGER	Yes	Código de barrio

Tabla 40 **Descripción de los campos de disciplina_barrio**

Tabla: disciplina_categoria

Column name	Primary key	Data type	Not NULL	Comment
coddis	Yes	INTEGER	Yes	Código de disciplina
codcat	Yes	INTEGER	Yes	Código de categoría

Tabla 79 **Descripción de los campos de disciplina_categoria**

Tabla: equipo

Column name	Primary key	Data type	Not NULL	Comment
code	Yes	INTEGER	Yes	Código del equipo
nombre	No	CHARACTER VARYING(40)	Yes	Nombre del equipo
estado	No	INTEGER	Yes	Estado del equipo
codcat	No	INTEGER	Yes	Código de categoría
codhor	No	INTEGER	No	Código de horario
coddis	No	INTEGER	Yes	Código de disciplina
idusuario	No	INTEGER	No	Id del entrenador

Tabla 80 Descripción de los campos de equipo

Tabla: equipo_rolpartido

Column name	Primary key	Data type	Not NULL	Comment
code	Yes	INTEGER	Yes	Código de equipo
codrolp	Yes	INTEGER	Yes	Código de Rol de partido
resultado	No	INTEGER	Yes	Resultado del partido
ganador	No	INTEGER	No	Equipo ganador del partido

Tabla 81 Descripción de los campos de equipo_rolpartido

Tabla: fotos

Column name	Primary key	Data type	Not NULL	Comment
idfoto	Yes	INTEGER	Yes	Es el código de la foto.
url	No	CHARACTER VARYING(40)	Yes	Es la dirección de la foto.
titulo	No	CHARACTER VARYING(40)	Yes	Es el título de la foto.
contenido	No	CHARACTER VARYING(40)	No	Es el contenido de la foto.
estado	No	INTEGER	Yes	Es el estado de la foto,

				puede ser 1 activo y 0 inactivo.
idusuario	No	INTEGER	No	El es idusuario que hereda de la tabla Usuario.

Tabla 82 Descripción de los campos de fotos

Tabla: horario

Column name	Primary key	Data type	Not NULL	Comment
codhor	Yes	INTEGER	Yes	Código de horario
horaini	No	TIME	Yes	Hora de inicio
horafin	No	TIME	Yes	Hora de fin

Tabla 83 Descripción de los campos de horario

Tabla: pais

Column name	Primary key	Data type	Not NULL	Comment
codpais	Yes	INTEGER	Yes	Código de País
nombre	No	CHARACTER VARYING(40)	Yes	Nombre de país
Sigla	No	CHARACTER VARYING(40)	Yes	Sigla

Tabla 84 Descripción de los campos de pais

Tabla: procesos

Column name	Primary key	Data type	Not NULL	Comment
codpro	Yes	INTEGER	Yes	Es el código de procesos.
nombre	No	CHARACTER VARYING(40)	Yes	Este campo representa el nombre del proceso.
estado	No	INTEGER	Yes	Es el estado del proceso, puede ser 1 activo y 0 inactivo.
link	No	CHARACTER VARYING(60)	Yes	El link es una dirección que lleva a una lista de procesos.

Tabla 85 Descripción de los campos de procesos

Tabla de profesión

Column name	Primary key	Data type	Not NULL	Comment
Codprof	Yes	INTEGER	Yes	Es el código de profesión.
Nombreprof	No	CHARACTER VARYING(40)	Yes	Es el nombre de la profesión.

Tabla 86 Descripción de los campos de profesión

Tabla: provincia

Column name	Primary key	Data type	Not NULL	Comment
codprov	Yes	INTEGER	Yes	Código de provincial
nombre	No	CHARACTER VARYING(40)	Yes	Nombre de la provincial
coddep	No	INTEGER	Yes	Código de departamento

Tabla 87 Descripción de los campos de provincia

Tabla: publicaciones

Column name	Primary key	Data type	Not NULL	Comment
codpubli	Yes	INTEGER	Yes	Es el código de publicación.
nombre	No	CHARACTER VARYING(40)	Yes	Es el nombre de publicación.
contenido	No	CHARACTER VARYING(999)	Yes	Es el contenido de las publicaciones.
fecha	No	DATE	Yes	Representa la fecha en la que se realizó la publicación.
fecha_fin	No	DATE	Yes	La fecha fin representa el fin de la publicación.
idusuario	No	INTEGER	Yes	El idusuario hereda de la tabla usuario.
img	No	CHARACTER VARYING(40)	No	Es la foto de la noticia en las publicaciones.

Tabla 88 Descripción de los campos de publicaciones

Tabla: rol

Column name	Primary key	Data type	Not NULL	Comment
codrol	Yes	INTEGER	Yes	Código de rol
nombre	No	CHARACTER VARYING(40)	Yes	Nombre de rol
descripcion	No	CHARACTER VARYING(40)	Yes	Descripción del rol
estado	No	INTEGER	Yes	Estado del rol

Tabla 89 Descripción de los campos de rol

Tabla: rolpartido

Column name	Primary key	Data type	Not NULL	Comment
codrolp	Yes	INTEGER	Yes	Código
ronda	No	INTEGER	Yes	Numero de ronda del partido
fecha	No	DATE	Yes	Fecha del partido
hora	No	CHARACTER VARYING(40)	Yes	Hora del partido

observacion	No	CHARACTER VARYING(40)	No	Observaciones
idcamserie	No	INTEGER	No	Id de Campeonato y serie
estado	No	INTEGER	No	Estado del partido
orden	No	INTEGER	No	Numero de partido en la ronda

Tabla 90 Descripción de los campos de rolpartido

Tabla: rol_procesos:

Column name	Primary key	Data type	Not NULL	Comment
codrol	Yes	INTEGER	Yes	Código de rol
codpro	Yes	INTEGER	Yes	Código de proceso

Tabla 91 Descripción de los campos de rol_procesos

Tabla: serie

Column name	Primary key	Data type	Not NULL	Comment
codserie	Yes	INTEGER	Yes	Código de la serie
nombre	No	CHARACTER VARYING(40)	Yes	Nombre de la serie

Tabla 92 Descripción de los campos de serie

Tabla: registroacciones

Column name	Primary key	Data type	Not NULL	Comment
id	Yes	INTEGER	Yes	Código de registro de la acción
idusuario	No	INTEGER	Yes	Código de usuario que realice la acción
fecha	No	DATE	Yes	Fecha en la que se realice la acción
hora	No	TIME	Yes	Hora en la que se realiza la acción
accion	No	CHARACTER VARYING(40)	No	Acción que el usuario realiza en el sistema

Tabla 93 Descripción de los campos de registroacciones

Tabla: usuarios

Column name	Primary key	Data type	Not NULL	Comment
idusuario	Yes	INTEGER	Yes	Es el identificador o código de los usuarios del sistema.
ci	No	CHARACTER VARYING(40)	Yes	Se refiere al carnet de identidad de los usuarios.
nombre	No	CHARACTER VARYING(40)	Yes	Es el nombre del usuario.
ap	No	CHARACTER VARYING(40)	Yes	Es el apellido paterno del usuario.
am	No	CHARACTER VARYING(40)	No	Es el apellido materno del usuario.
fechadnac	No	DATE	Yes	Es la fecha de nacimiento del usuario.
telefono	No	INTEGER	Yes	Teléfono del usuario
email	No	CHARACTER VARYING(40)	No	Es el correo electrónico usuario.
estado	No	INTEGER	Yes	Es el estado del usuario, puede ser 1 activo y 0 inactivo.

foto	No	CHARACTER VARYING(40)	No	Es la foto o imagen del producto que se publica.
coddir	No	INTEGER	Yes	Es el código de la dirección a la que pertenece el usuario
codprov	No	INTEGER	Yes	Es el código de la provincia a la que pertenece el usuario.
cod_curr	No	INTEGER	Yes	Es el código del curricular a la que pertenece el usuario
codprof	No	INTEGER	No	Es el código de profesión a la que pertenece el usuario

Tabla 94 Descripción de los campos usuarios

Tabla: usuarios_rol:

Column name	Primary key	Data type	Not NULL	Comment
idusuario	Yes	INTEGER	Yes	Id de usuario
codrol	Yes	INTEGER	Yes	Código de rol
coddis	Yes	INTEGER	Yes	Código de disciplina
fecha	No	DATE	No	Fecha en que se asigna el rol

Tabla 95 Descripción de los campos de usuarios_rol

I.1.1.1 Script para la Base de Datos

Tables

Add table "procesos"

```
CREATE TABLE procesos (
codpro INTEGER  NOT NULL,
nombre CHARACTER VARYING(40)  NOT NULL,
estado INTEGER  NOT NULL,
link CHARACTER VARYING(60)  NOT NULL,
CONSTRAINT PK_procesos PRIMARY KEY (codpro)
);
```

Add table "rol"

```
CREATE TABLE rol (
codrol INTEGER  NOT NULL,
nombre CHARACTER VARYING(40)  NOT NULL,
descripcion CHARACTER VARYING(40)  NOT NULL,
estado INTEGER  NOT NULL,
CONSTRAINT PK_rol PRIMARY KEY (codrol)
);
```

Add table "rol_procesos"

```
CREATE TABLE rol_procesos (
codrol INTEGER  NOT NULL,
codpro INTEGER  NOT NULL,
CONSTRAINT PK_rol_procesos PRIMARY KEY (codrol, codpro)
);
```

Add table "usuarios"

```
CREATE TABLE usuarios (
idusuario INTEGER  NOT NULL,
ci CHARACTER VARYING(40)  NOT NULL,
nombre CHARACTER VARYING(40)  NOT NULL,
ap CHARACTER VARYING(40)  NOT NULL,
am CHARACTER VARYING(40),
fechadnac DATE  NOT NULL,
telefono INTEGER  NOT NULL,
email CHARACTER VARYING(40),
estado INTEGER  NOT NULL,
foto CHARACTER VARYING(40),
coddir INTEGER  NOT NULL,
codprov INTEGER  NOT NULL,
```

```
cod_curr INTEGER  NOT NULL,
codprof INTEGER,
CONSTRAINT PK_usuarios PRIMARY KEY (idusuario)
);
Add table "usuarios_rol"
CREATE TABLE usuarios_rol (
idusuario INTEGER  NOT NULL,
codrol INTEGER  NOT NULL,
coddis INTEGER  NOT NULL,
fecha DATE,
CONSTRAINT PK_usuarios_rol PRIMARY KEY (idusuario, codrol, coddis)
);

Add table "datos"

CREATE TABLE datos (
iddatos INTEGER  NOT NULL,
login CHARACTER VARYING(40)  NOT NULL,
clave CHARACTER VARYING(40)  NOT NULL,
idusuario INTEGER  NOT NULL,
CONSTRAINT PK_datos PRIMARY KEY (iddatos)
);
```

Add table "direccion"

```
CREATE TABLE direccion (
coddir INTEGER  NOT NULL,
barrio CHARACTER VARYING(40)  NOT NULL,
calle CHARACTER VARYING(40)  NOT NULL,
numerocasa INTEGER,
CONSTRAINT PK_direccion PRIMARY KEY (coddir)
);
```

Add table "profesion"

```
CREATE TABLE profesion (
codprof INTEGER  NOT NULL,
nombreprof CHARACTER VARYING(40)  NOT NULL,
CONSTRAINT PK_profesion PRIMARY KEY (codprof)
);

COMMENT ON COLUMN profesion.codprof IS 'Es el codigo de profesion.';

COMMENT ON COLUMN profesion.nombreprof IS 'Es el nombre de la profesion.';
```

Add table "disciplina"

```
CREATE TABLE disciplina (
coddis INTEGER  NOT NULL,
nombredis CHARACTER VARYING(40)  NOT NULL,
estado INTEGER  NOT NULL,
CONSTRAINT PK_disciplina PRIMARY KEY (coddis)
);
```

Add table "barrio"

```
CREATE TABLE barrio (
codbarrio INTEGER  NOT NULL,
nombre CHARACTER VARYING(40)  NOT NULL,
estado INTEGER  NOT NULL,
CONSTRAINT PK_barrio PRIMARY KEY (codbarrio)
);
```

Add table "publicaciones"

```
CREATE TABLE publicaciones (
codpubli INTEGER  NOT NULL,
nombre CHARACTER VARYING(40)  NOT NULL,
contenido CHARACTER VARYING(999)  NOT NULL,
fecha DATE  NOT NULL,
```

```
fecha_fin DATE  NOT NULL,
idusuario INTEGER  NOT NULL,
img CHARACTER VARYING(40),
CONSTRAINT PK_publicaciones PRIMARY KEY (codpubli)
);

Add table "disciplina_barrio"

CREATE TABLE disciplina_barrio (
coddis INTEGER  NOT NULL,
codbarrio INTEGER  NOT NULL,
CONSTRAINT PK_disciplina_barrio PRIMARY KEY (coddis, codbarrio)
);

Add table "alumnos"
CREATE TABLE alumnos (
codalum INTEGER  NOT NULL,
nombres CHARACTER VARYING(40)  NOT NULL,
ap CHARACTER VARYING(30)  NOT NULL,
am CHARACTER VARYING(40),
fechanac DATE,
celular INTEGER,
estado INTEGER,
```

```
fechaini DATE,
edad INTEGER,
peso INTEGER,
estatura DOUBLE PRECISION,
telefono INTEGER  NOT NULL,
colegio CHARACTER VARYING(40),
curso CHARACTER VARYING(40),
nombrepa CHARACTER VARYING(40),
nombrema CHARACTER VARYING(40),
nombreapo CHARACTER VARYING(40),
profesionpa CHARACTER VARYING(40),
profesionma CHARACTER VARYING(40),
profesionapo CHARACTER VARYING(40),
teltrabajoma INTEGER,
teltrabajopa INTEGER,
teltrabajoap INTEGER,
coddir INTEGER  NOT NULL,
codprov INTEGER  NOT NULL,
sexo CHARACTER VARYING(40)  NOT NULL,
code INTEGER,
coddis INTEGER,
codcat INTEGER,
CONSTRAINT PK_alumnos PRIMARY KEY (codalum) );
```

Add table "categoria"

```
CREATE TABLE categoria (
codcat INTEGER  NOT NULL,
nombre CHARACTER VARYING(40)  NOT NULL,
estado INTEGER  NOT NULL,
CONSTRAINT PK_categoria PRIMARY KEY (codcat)
);
```

Add table "equipo"

```
CREATE TABLE equipo (
code INTEGER  NOT NULL,
nombre CHARACTER VARYING(40)  NOT NULL,
estado INTEGER  NOT NULL,
codcat INTEGER  NOT NULL,
codhor INTEGER,
coddis INTEGER  NOT NULL,
idusuario INTEGER,
CONSTRAINT PK_equipo PRIMARY KEY (code)
);
```

Add table "horario"

```
CREATE TABLE horario (
codhor INTEGER  NOT NULL,
horaini TIME  NOT NULL,
horafin TIME  NOT NULL,
CONSTRAINT PK_horario PRIMARY KEY (codhor)
);
```

Add table "campeonato"

```
CREATE TABLE campeonato (
codcam INTEGER  NOT NULL,
nombre CHARACTER VARYING(40)  NOT NULL,
fechaini DATE  NOT NULL,
fechafin DATE  NOT NULL,
estado INTEGER  NOT NULL,
coddis INTEGER  NOT NULL,
codcat INTEGER,
CONSTRAINT PK_campeonato PRIMARY KEY (codcam)
);
```

Add table "barrio_campeonato"

```
CREATE TABLE barrio_campeonato (
codbarrio INTEGER  NOT NULL,
codcam INTEGER  NOT NULL,
CONSTRAINT PK_barrio_campeonato PRIMARY KEY (codbarrio, codcam)
);
```

Add table "serie"

```
CREATE TABLE serie (
codserie INTEGER  NOT NULL,
nombre CHARACTER VARYING(40)  NOT NULL,
CONSTRAINT PK_serie PRIMARY KEY (codserie)
);
```

Add table "rolpartido"

```
CREATE TABLE rolpartido (
codrolp INTEGER  NOT NULL,
ronda INTEGER  NOT NULL,
fecha DATE  NOT NULL,
hora CHARACTER VARYING(40)  NOT NULL,
observacion CHARACTER VARYING(40),
```

```
cod_arb INTEGER  NOT NULL,
idcamserie INTEGER,
estado INTEGER,
orden INTEGER,
CONSTRAINT PK_rolpartido PRIMARY KEY (codrolp)
);
Add table "equipo_rolpartido"
CREATE TABLE equipo_rolpartido (
code INTEGER  NOT NULL,
codrolp INTEGER  NOT NULL,
resultado INTEGER  NOT NULL,
ganador INTEGER,
CONSTRAINT PK_equipo_rolpartido PRIMARY KEY (code, codrolp)
);

Add table "departamento"
CREATE TABLE departamento (
coddep INTEGER  NOT NULL,
nombred CHARACTER VARYING(40)  NOT NULL,
codpais INTEGER  NOT NULL,
CONSTRAINT PK_departamento PRIMARY KEY (coddep)
);
```

Add table "provincia"

```
CREATE TABLE provincia (
codprov INTEGER  NOT NULL,
nombre CHARACTER VARYING(40)  NOT NULL,
coddep INTEGER  NOT NULL,
CONSTRAINT PK_provincia PRIMARY KEY (codprov)
);
```

Add table "pais"

```
CREATE TABLE pais (
codpais INTEGER  NOT NULL,
nombre CHARACTER VARYING(40)  NOT NULL,
sigla CHARACTER VARYING(40)  NOT NULL,
CONSTRAINT PK_pais PRIMARY KEY (codpais)
);
```

Add table "Curriculum"

```
CREATE TABLE Curriculum (
cod_curr INTEGER  NOT NULL,
estudios CHARACTER VARYING(500)  NOT NULL,
deportista CHARACTER VARYING(500)  NOT NULL,
cursos CHARACTER VARYING(500)  NOT NULL,
entrenador CHARACTER VARYING(500)  NOT NULL,
CONSTRAINT PK_Curriculum PRIMARY KEY (cod_curr)
);
```

Add table "arbitro"

```
CREATE TABLE arbitro (
cod_arb INTEGER  NOT NULL,
nombre CHARACTER VARYING(40)  NOT NULL,
apellidos CHARACTER VARYING(40)  NOT NULL,
telefono INTEGER,
correo CHARACTER VARYING(40),
coddis INTEGER  NOT NULL,
estado INTEGER,
CONSTRAINT PK_arbitro PRIMARY KEY (cod_arb)
);
```

Add table "campeonato_serie"

```
CREATE TABLE campeonato_serie (
idcamserie INTEGER  NOT NULL,
codcam INTEGER  NOT NULL,
codserie INTEGER  NOT NULL,
estado INTEGER,
CONSTRAINT PK_campeonato_serie PRIMARY KEY (idcamserie)
);
```

Add table "campeonato_serie_equipo"

```
CREATE TABLE campeonato_serie_equipo (
idcamserie INTEGER  NOT NULL,
code INTEGER  NOT NULL,
puntos INTEGER DEFAULT 0  NOT NULL,
goles INTEGER DEFAULT 0  NOT NULL,
partidos INTEGER  NOT NULL,
partidose INTEGER  NOT NULL,
partidosp INTEGER  NOT NULL,
partidosg INTEGER  NOT NULL,
golescontra INTEGER  NOT NULL,
CONSTRAINT PK_campeonato_serie_equipo PRIMARY KEY (idcamserie, code)
);
```

Add table "disciplina_categoria"

```
CREATE TABLE disciplina_categoria (
coddis INTEGER  NOT NULL,
codcat INTEGER  NOT NULL,
CONSTRAINT PK_disciplina_categoria PRIMARY KEY (coddis, codcat)
);
```

Add table "fotos"

```
CREATE TABLE fotos (
idfoto INTEGER  NOT NULL,
url CHARACTER VARYING(40)  NOT NULL,
titulo CHARACTER VARYING(40)  NOT NULL,
contenido CHARACTER VARYING(40),
estado INTEGER  NOT NULL,
idusuario INTEGER,
CONSTRAINT PK_fotos PRIMARY KEY (idfoto)
);
```

Add table "registroacciones"

```
CREATE TABLE registroacciones (
id INTEGER  NOT NULL,
idusuario INTEGER  NOT NULL,
fecha DATE  NOT NULL,
hora TIME  NOT NULL,
accion CHARACTER VARYING(40),
CONSTRAINT PK_ registroacciones PRIMARY KEY (id)
);
```

Foreign key constraints

```
ALTER TABLE rol_procesos ADD CONSTRAINT rol_rol_procesos
FOREIGN KEY (codrol) REFERENCES rol (codrol);

ALTER TABLE rol_procesos ADD CONSTRAINT procesos_rol_procesos
FOREIGN KEY (codpro) REFERENCES procesos (codpro);

ALTER TABLE usuarios ADD CONSTRAINT direccion_usuarios
FOREIGN KEY (coddir) REFERENCES direccion (coddir);

ALTER TABLE usuarios ADD CONSTRAINT provincia_usuarios
```

```
FOREIGN KEY (codprov) REFERENCES provincia (codprov);

ALTER TABLE usuarios ADD CONSTRAINT Curriculum_usuarios
FOREIGN KEY (cod_curr) REFERENCES Curriculum (cod_curr);

ALTER TABLE usuarios ADD CONSTRAINT profesion_usuarios
FOREIGN KEY (codprof) REFERENCES profesion (codprof);

ALTER TABLE usuarios_rol ADD CONSTRAINT usuarios_usuarios_rol
FOREIGN KEY (idusuario) REFERENCES usuarios (idusuario);

ALTER TABLE usuarios_rol ADD CONSTRAINT rol_usuarios_rol
FOREIGN KEY (codrol) REFERENCES rol (codrol);

ALTER TABLE usuarios_rol ADD CONSTRAINT disciplina_usuarios_rol
FOREIGN KEY (coddis) REFERENCES disciplina (coddis);

ALTER TABLE datos ADD CONSTRAINT usuarios_datos
FOREIGN KEY (idusuario) REFERENCES usuarios (idusuario);

ALTER TABLE publicaciones ADD CONSTRAINT usuarios_publicaciones
FOREIGN KEY (idusuario) REFERENCES usuarios (idusuario);
```

```
ALTER TABLE disciplina_barrio ADD CONSTRAINT disciplina_disciplina_barrio
FOREIGN KEY (coddis) REFERENCES disciplina (coddis);

ALTER TABLE disciplina_barrio ADD CONSTRAINT barrio_disciplina_barrio
FOREIGN KEY (codbarrio) REFERENCES barrio (codbarrio);

ALTER TABLE alumnos ADD CONSTRAINT direccion_alumnos
FOREIGN KEY (coddir) REFERENCES direccion (coddir);

ALTER TABLE alumnos ADD CONSTRAINT provincia_alumnos
FOREIGN KEY (codprov) REFERENCES provincia (codprov);

ALTER TABLE alumnos ADD CONSTRAINT equipo_alumnos
FOREIGN KEY (code) REFERENCES equipo (code);

ALTER TABLE alumnos ADD CONSTRAINT disciplina_alumnos
FOREIGN KEY (coddis) REFERENCES disciplina (coddis);

ALTER TABLE alumnos ADD CONSTRAINT categoria_alumnos
FOREIGN KEY (codcat) REFERENCES categoria (codcat);

ALTER TABLE equipo ADD CONSTRAINT categoria_equipo
FOREIGN KEY (codcat) REFERENCES categoria (codcat);
```

```
ALTER TABLE equipo ADD CONSTRAINT horario_equipo
FOREIGN KEY (codhor) REFERENCES horario (codhor);

ALTER TABLE equipo ADD CONSTRAINT disciplina_equipo
FOREIGN KEY (coddis) REFERENCES disciplina (coddis);

ALTER TABLE equipo ADD CONSTRAINT usuarios_equipo
FOREIGN KEY (idusuario) REFERENCES usuarios (idusuario);

ALTER TABLE campeonato ADD CONSTRAINT disciplina_campeonato
FOREIGN KEY (coddis) REFERENCES disciplina (coddis);

ALTER TABLE campeonato ADD CONSTRAINT categoria_campeonato
FOREIGN KEY (codcat) REFERENCES categoria (codcat);

ALTER TABLE barrio_campeonato ADD CONSTRAINT
barrio_barrio_campeonato
FOREIGN KEY (codbarrio) REFERENCES barrio (codbarrio);

ALTER TABLE barrio_campeonato ADD CONSTRAINT
campeonato_barrio_campeonato
FOREIGN KEY (codcam) REFERENCES campeonato (codcam);
```

```
ALTER TABLE rolpartido ADD CONSTRAINT arbitro_rolpartido
FOREIGN KEY (cod_arb) REFERENCES arbitro (cod_arb);

ALTER TABLE rolpartido ADD CONSTRAINT campeonato_serie_rolpartido
FOREIGN KEY (idcamserie) REFERENCES campeonato_serie (idcamserie);

ALTER TABLE equipo_rolpartido ADD CONSTRAINT equipo_equipo_rolpartido
FOREIGN KEY (code) REFERENCES equipo (code);

ALTER TABLE equipo_rolpartido ADD CONSTRAINT
rolpartido_equipo_rolpartido
FOREIGN KEY (codrolp) REFERENCES rolpartido (codrolp);

ALTER TABLE departamento ADD CONSTRAINT pais_departamento
FOREIGN KEY (codpais) REFERENCES pais (codpais);

ALTER TABLE provincia ADD CONSTRAINT departamento_provincia
FOREIGN KEY (coddep) REFERENCES departamento (coddep);

ALTER TABLE arbitro ADD CONSTRAINT disciplina_arbitro
FOREIGN KEY (coddis) REFERENCES disciplina (coddis);

ALTER TABLE campeonato_serie ADD CONSTRAINT
campeonato_campeonato_serie
```

```
FOREIGN KEY (codcam) REFERENCES campeonato (codcam);

ALTER TABLE campeonato_serie ADD CONSTRAINT serie_campeonato_serie
FOREIGN KEY (codserie) REFERENCES serie (codserie);

ALTER TABLE campeonato_serie_equipo ADD CONSTRAINT
campeonato_serie_campeonato_serie_equipo
FOREIGN KEY (idcamserie) REFERENCES campeonato_serie (idcamserie);

ALTER TABLE campeonato_serie_equipo ADD CONSTRAINT
equipo_campeonato_serie_equipo
FOREIGN KEY (code) REFERENCES equipo (code);

ALTER TABLE disciplina_categoria ADD CONSTRAINT
disciplina_disciplina_categoria
FOREIGN KEY (coddis) REFERENCES disciplina (coddis);

ALTER TABLE disciplina_categoria ADD CONSTRAINT
categoria_disciplina_categoria
FOREIGN KEY (codcat) REFERENCES categoria (codcat);

ALTER TABLE fotos ADD CONSTRAINT usuarios_fotos
FOREIGN KEY (idusuario) REFERENCES usuarios (idusuario);
ALTER TABLE registroacciones ADD CONSTRAINT usuarios_ registroacciones
```

FOREIGN KEY (idusuario) REFERENCES usuarios (idusuario);

Addtable "disciplina"

CREATE TABLE disciplina (

coddis INTEGER NOT NULL,

nombredis CHARACTER VARYING(40) NOT NULL,

estado INTEGER NOT NULL,

CONSTRAINT PK_disciplina PRIMARY KEY (coddis)

);

Add table "barrio"

CREATE TABLE barrio (

codbarrio INTEGER NOT NULL,

nombre CHARACTER VARYING(40) NOT NULL,

estado INTEGER NOT NULL,

CONSTRAINT PK_barrio PRIMARY KEY (codbarrio)

);

Add table "publicaciones"

```
CREATE TABLE publicaciones (
codpubli INTEGER  NOT NULL,
nombre CHARACTER VARYING(40)  NOT NULL,
contenido CHARACTER VARYING(999)  NOT NULL,
fecha DATE  NOT NULL,
fecha_fin DATE  NOT NULL,
idusuario INTEGER  NOT NULL,
img CHARACTER VARYING(40),
CONSTRAINT PK_publicaciones PRIMARY KEY (codpubli)
);
```

Addtable "disciplina_barrio"

```
CREATE TABLE disciplina_barrio (
coddis INTEGER  NOT NULL,
codbarrio INTEGER  NOT NULL,
CONSTRAINT PK_disciplina_barrio PRIMARY KEY (coddis, codbarrio)
);
```

Addtable "alumnos"

```
CREATE TABLE alumnos (
codalum INTEGER  NOT NULL,
nombres CHARACTER VARYING(40)  NOT NULL,
ap CHARACTER VARYING(30)  NOT NULL,
am CHARACTER VARYING(40),
fechanac DATE,
celular INTEGER,
estado INTEGER,
fechaini DATE,
edad INTEGER,
peso INTEGER,
estatura DOUBLE PRECISION,
telefono INTEGER  NOT NULL,
colegio CHARACTER VARYING(40),
curso CHARACTER VARYING(40),
nombrepa CHARACTER VARYING(40),
nombrema CHARACTER VARYING(40),
nombreapo CHARACTER VARYING(40),
profesionpa CHARACTER VARYING(40),
profesionma CHARACTER VARYING(40),
```

```
profesionapo CHARACTER VARYING(40),
teltrabajoma INTEGER,
teltrabajopa INTEGER,
teltrabajoap INTEGER,
coddir INTEGER  NOT NULL,
codprov INTEGER  NOT NULL,
sexo CHARACTER VARYING(40),
CONSTRAINT PK_alumnos PRIMARY KEY (codalum)
);
```

Add table "categoria"

```
CREATE TABLE categoria (
codcat INTEGER  NOT NULL,
nombre CHARACTER VARYING(40)  NOT NULL,
estado INTEGER  NOT NULL,
CONSTRAINT PK_categoria PRIMARY KEY (codcat)
);
```

Add table "equipo"

```
CREATE TABLE equipo (
code INTEGER  NOT NULL,
nombre CHARACTER VARYING(40)  NOT NULL,
estado INTEGER  NOT NULL,
codcat INTEGER  NOT NULL,
codhor INTEGER  NOT NULL,
coddis INTEGER,
CONSTRAINT PK_equipo PRIMARY KEY (code)
);
```

Add table "horario"

```
CREATE TABLE horario (
codhor INTEGER  NOT NULL,
horaini TIME  NOT NULL,
horafin TIME  NOT NULL,
CONSTRAINT PK_horario PRIMARY KEY (codhor)
);
```

Add table "campeonato"

```
CREATE TABLE campeonato (
codcam INTEGER  NOT NULL,
nombre CHARACTER VARYING(40)  NOT NULL,
fechaini DATE  NOT NULL,
fechafin DATE  NOT NULL,
estado INTEGER  NOT NULL,
coddis INTEGER  NOT NULL,
CONSTRAINT PK_campeonato PRIMARY KEY (codcam)
);
```

Add table "barrio_campeonato"

```
CREATE TABLE barrio_campeonato (
codbarrio INTEGER  NOT NULL,
codcam INTEGER  NOT NULL,
CONSTRAINT PK_barrio_campeonato PRIMARY KEY (codbarrio, codcam)
);
```

Add table "serie"

```
CREATE TABLE serie (
codserie INTEGER  NOT NULL,
nombre CHARACTER VARYING(40)  NOT NULL,
CONSTRAINT PK_serie PRIMARY KEY (codserie)
);
```

Add table "rolpartido"

```
CREATE TABLE rolpartido (
codrolp INTEGER  NOT NULL,
ronda INTEGER  NOT NULL,
fecha DATE  NOT NULL,
hora CHARACTER(40)  NOT NULL,
observacion CHARACTER(40),
cod_arb INTEGER  NOT NULL,
idcamserie INTEGER,
estado INTEGER,
CONSTRAINT PK_rolpartido PRIMARY KEY (codrolp)
);
```

Add table "equipo_rolpartido"

```
CREATE TABLE equipo_rolpartido (
code INTEGER  NOT NULL,
codrolp INTEGER  NOT NULL,
resultado CHARACTER(40)  NOT NULL,
CONSTRAINT PK_equipo_rolpartido PRIMARY KEY (code, codrolp)
);
```

Add table "departamento"

```
CREATE TABLE departamento (
coddep INTEGER  NOT NULL,
nombred CHARACTER VARYING(40)  NOT NULL,
codpais INTEGER  NOT NULL,
CONSTRAINT PK_departamento PRIMARY KEY (coddep)
);
```

Add table "provincia"

```
CREATE TABLE provincia (
codprov INTEGER  NOT NULL,
nombre CHARACTER VARYING(40)  NOT NULL,
coddep INTEGER  NOT NULL,
CONSTRAINT PK_provincia PRIMARY KEY (codprov)
);
```

Add table "pais"

```
CREATE TABLE pais (
codpais INTEGER  NOT NULL,
nombre CHARACTER VARYING(40)  NOT NULL,
sigla CHARACTER VARYING(40)  NOT NULL,
CONSTRAINT PK_pais PRIMARY KEY (codpais)
);
```

Add table "Curriculum"

```
CREATE TABLE Curriculum (
cod_curr INTEGER  NOT NULL,
estudios CHARACTER VARYING(500)  NOT NULL,
deportista CHARACTER VARYING(500)  NOT NULL,
cursos CHARACTER VARYING(500)  NOT NULL,
entrenador CHARACTER VARYING(500)  NOT NULL,
CONSTRAINT PK_Curriculum PRIMARY KEY (cod_curr)
);
```

Add table "arbitro"

```
CREATE TABLE arbitro (
cod_arb INTEGER  NOT NULL,
nombre CHARACTER VARYING(40)  NOT NULL,
apellidos CHARACTER VARYING(40)  NOT NULL,
telefono INTEGER,
correo CHARACTER VARYING(40),
coddis INTEGER,
estado INTEGER,
CONSTRAINT PK_arbitro PRIMARY KEY (cod_arb)
);
```

Add table "campeonato_serie"

```
CREATE TABLE campeonato_serie (
idcamserie INTEGER  NOT NULL,
codcam INTEGER  NOT NULL,
codserie INTEGER  NOT NULL,
estado INTEGER,
CONSTRAINT PK_campeonato_serie PRIMARY KEY (idcamserie)
);
```

Add table "campeonato_serie_equipo"

```
CREATE TABLE campeonato_serie_equipo (
idcamserie INTEGER  NOT NULL,
code INTEGER  NOT NULL,
CONSTRAINT PK_campeonato_serie_equipo PRIMARY KEY (idcamserie, code)
);
```

Addtable "disciplina_categoria"

```
CREATE TABLE disciplina_categoria (
coddis INTEGER  NOT NULL,
codcat INTEGER  NOT NULL,
CONSTRAINT PK_disciplina_categoria PRIMARY KEY (coddis, codcat)
);
```

Add table "fotos"

```
CREATE TABLE fotos (
idfoto INTEGER  NOT NULL,
url CHARACTER VARYING(40)  NOT NULL,
titulo CHARACTER VARYING(40)  NOT NULL,
contenido CHARACTER VARYING(40),
estado INTEGER  NOT NULL,
idusuario INTEGER,
CONSTRAINT PK_fotos PRIMARY KEY (idfoto)
);
```

Foreign key constraints

```
ALTER TABLE rol_procesos ADD CONSTRAINT rol_rol_procesos
FOREIGN KEY (codrol) REFERENCES rol (codrol);

ALTER TABLE rol_procesos ADD CONSTRAINT procesos_rol_procesos
FOREIGN KEY (codpro) REFERENCES procesos (codpro);

ALTER TABLE usuarios ADD CONSTRAINT direccion_usuarios
FOREIGN KEY (coddir) REFERENCES direccion (coddir);

ALTER TABLE usuarios ADD CONSTRAINT provincia_usuarios
FOREIGN KEY (codprov) REFERENCES provincia (codprov);

ALTER TABLE usuarios ADD CONSTRAINT Curriculum_usuarios
FOREIGN KEY (cod_curr) REFERENCES Curriculum (cod_curr);

ALTER TABLE usuarios ADD CONSTRAINT profesion_usuarios
FOREIGN KEY (codprof) REFERENCES profesion (codprof);

ALTER TABLE usuarios_rol ADD CONSTRAINT usuarios_usuarios_rol
FOREIGN KEY (idusuario) REFERENCES usuarios (idusuario);
```

```
ALTER TABLE usuarios_rol ADD CONSTRAINT rol_usuarios_rol
FOREIGN KEY (codrol) REFERENCES rol (codrol);

ALTER TABLE usuarios_rol ADD CONSTRAINT disciplina_usuarios_rol
FOREIGN KEY (coddis) REFERENCES disciplina (coddis);

ALTER TABLE datos ADD CONSTRAINT usuarios_datos
FOREIGN KEY (idusuario) REFERENCES usuarios (idusuario);

ALTER TABLE publicaciones ADD CONSTRAINT usuarios_publicaciones
FOREIGN KEY (idusuario) REFERENCES usuarios (idusuario);

ALTER TABLE disciplina_barrio ADD CONSTRAINT disciplina_disciplina_barrio
FOREIGN KEY (coddis) REFERENCES disciplina (coddis);

ALTER TABLE disciplina_barrio ADD CONSTRAINT barrio_disciplina_barrio
FOREIGN KEY (codbarrio) REFERENCES barrio (codbarrio);

ALTER TABLE alumnos ADD CONSTRAINT direccion_alumnos
FOREIGN KEY (coddir) REFERENCES direccion (coddir);

ALTER TABLE alumnos ADD CONSTRAINT provincia_alumnos
FOREIGN KEY (codprov) REFERENCES provincia (codprov);
```

```
ALTER TABLE equipo ADD CONSTRAINT categoria_equipo
FOREIGN KEY (codcat) REFERENCES categoria (codcat);

ALTER TABLE equipo ADD CONSTRAINT horario_equipo
FOREIGN KEY (codhor) REFERENCES horario (codhor);

ALTER TABLE equipo ADD CONSTRAINT disciplina_equipo
FOREIGN KEY (coddis) REFERENCES disciplina (coddis);

ALTER TABLE campeonato ADD CONSTRAINT disciplina_campeonato
FOREIGN KEY (coddis) REFERENCES disciplina (coddis);

ALTER TABLE barrio_campeonato ADD CONSTRAINT
barrio_barrio_campeonato
FOREIGN KEY (codbarrio) REFERENCES barrio (codbarrio);

ALTER TABLE barrio_campeonato ADD CONSTRAINT
campeonato_barrio_campeonato
FOREIGN KEY (codcam) REFERENCES campeonato (codcam);

ALTER TABLE rolpartido ADD CONSTRAINT arbitro_rolpartido
FOREIGN KEY (cod_arb) REFERENCES arbitro (cod_arb);
ALTER TABLE rolpartido ADD CONSTRAINT campeonato_serie_rolpartido
```

FOREIGN KEY (idcamserie) REFERENCES campeonato_serie (idcamserie);

ALTER TABLE equipo_rolpartido ADD CONSTRAINT equipo_equipo_rolpartido
FOREIGN KEY (code) REFERENCES equipo (code);

ALTER TABLE equipo_rolpartido ADD CONSTRAINT rolpartido_equipo_rolpartido
FOREIGN KEY (codrolp) REFERENCES rolpartido (codrolp);

ALTER TABLE departamento ADD CONSTRAINT pais_departamento
FOREIGN KEY (codpais) REFERENCES pais (codpais);

ALTER TABLE provincia ADD CONSTRAINT departamento_provincia
FOREIGN KEY (coddep) REFERENCES departamento (coddep);

ALTER TABLE arbitro ADD CONSTRAINT disciplina_arbitro
FOREIGN KEY (coddis) REFERENCES disciplina (coddis);

ALTER TABLE campeonato_serie ADD CONSTRAINT campeonato_campeonato_serie
FOREIGN KEY (codcam) REFERENCES campeonato (codcam);

ALTER TABLE campeonato_serie ADD CONSTRAINT serie_campeonato_serie
FOREIGN KEY (codserie) REFERENCES serie (codserie);

```
ALTER TABLE campeonato_serie_equipo ADD CONSTRAINT
campeonato_serie_campeonato_serie_equipo
FOREIGN KEY (idcamserie) REFERENCES campeonato_serie (idcamserie);

ALTER TABLE campeonato_serie_equipo ADD CONSTRAINT
equipo_campeonato_serie_equipo
FOREIGN KEY (code) REFERENCES equipo (code);

ALTER TABLE disciplina_categoria ADD CONSTRAINT
disciplina_disciplina_categoria
FOREIGN KEY (coddis) REFERENCES disciplina (coddis);

ALTER TABLE disciplina_categoria ADD CONSTRAINT
categoria_disciplina_categoria
FOREIGN KEY (codcat) REFERENCES categoria (codcat);

ALTER TABLE fotos ADD CONSTRAINT usuarios_fotos
FOREIGN KEY (idusuario) REFERENCES usuarios (idusuario);
```

I.1.5 Modelo Físico de la Base de Datos del Sistema

El Modelado físico de la base de Datos consiste en definir las estructuras de almacenamiento y de acceso para que de esta manera se alcance un rendimiento óptimo de la base de datos. Al modelado se aplicó la normalización y Reglas de Integridad cuya finalidad es reducir las inconsistencias y redundancias de los datos, evitar las anomalías en las manipulaciones de los mismos y facilitar el mantenimiento.

I.1.6 Prototipos de Interfaces de Usuario

I.1.6.1 Introducción

Se trata de la representación de prototipos (modelos), que permiten al usuario hacerse una idea más o menos precisa que proveerá el sistema.

I.1.6.2 Propósito

Presentar los prototipos de pantallas para que el usuario tenga una idea de la interfaz que se presentarán en el Sistema.

I.1.6.3 Alcance

Mostrar los Prototipos de pantallas para que el usuario tenga una idea de la interfaz que presentará el sistema.

I.1.6.4 Prototipo de Pantallas

I.1.6.4.1 Pantalla 1: Portada

Pantalla principal se puede visualizar noticias, convocatorias, eventos realizados por el Departamento de Deportes

Figura 96 Pantalla Principal

II.1.7.4.2 Pantalla 2: Login/ Clave

Entrar al Sistema para Administrar sus diferentes funciones.

Figura 97 Pantalla Login/Clave

II.1.7.4.3 Pantalla 3: Dentro del Sistema

Pantalla principal dentro del sistema.

Figura 98 **Pantalla Dentro del Sistema**

II.1.7.4.4 Pantalla 4: Gestionar Usuario

Administra todos los usuarios registrados.

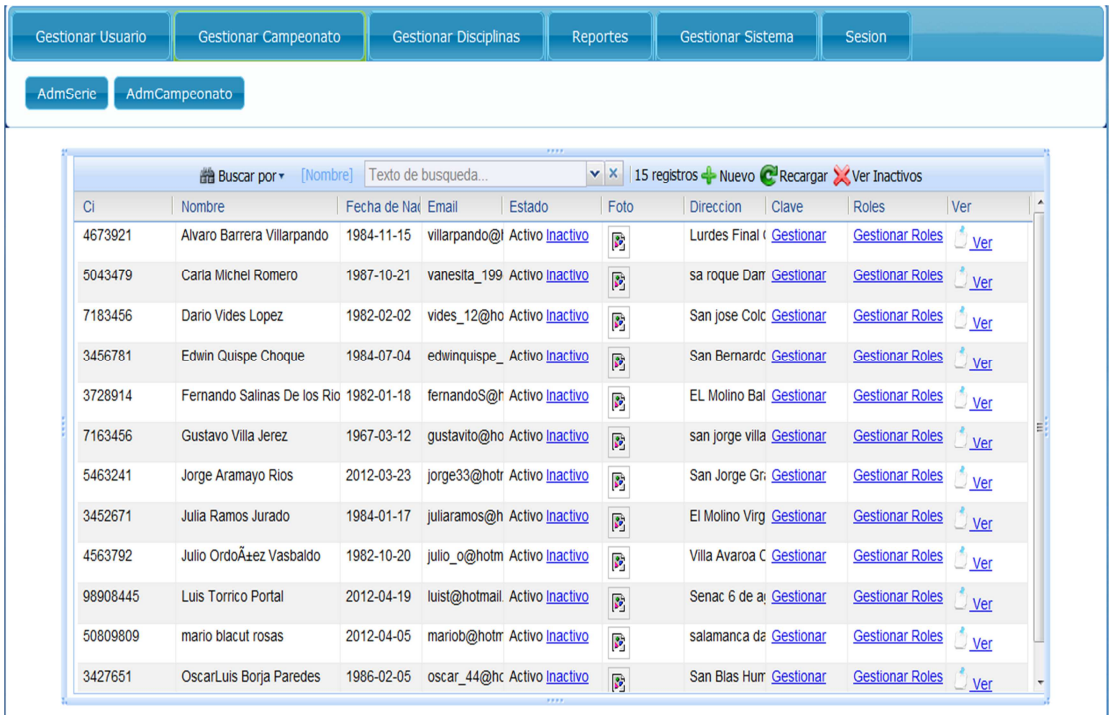


Figura 99 Pantalla Administrar Usuario

II.1.7.4.5 Pantalla 5: Administrar Usuario

La pantalla es de adicionar un nuevo usuario

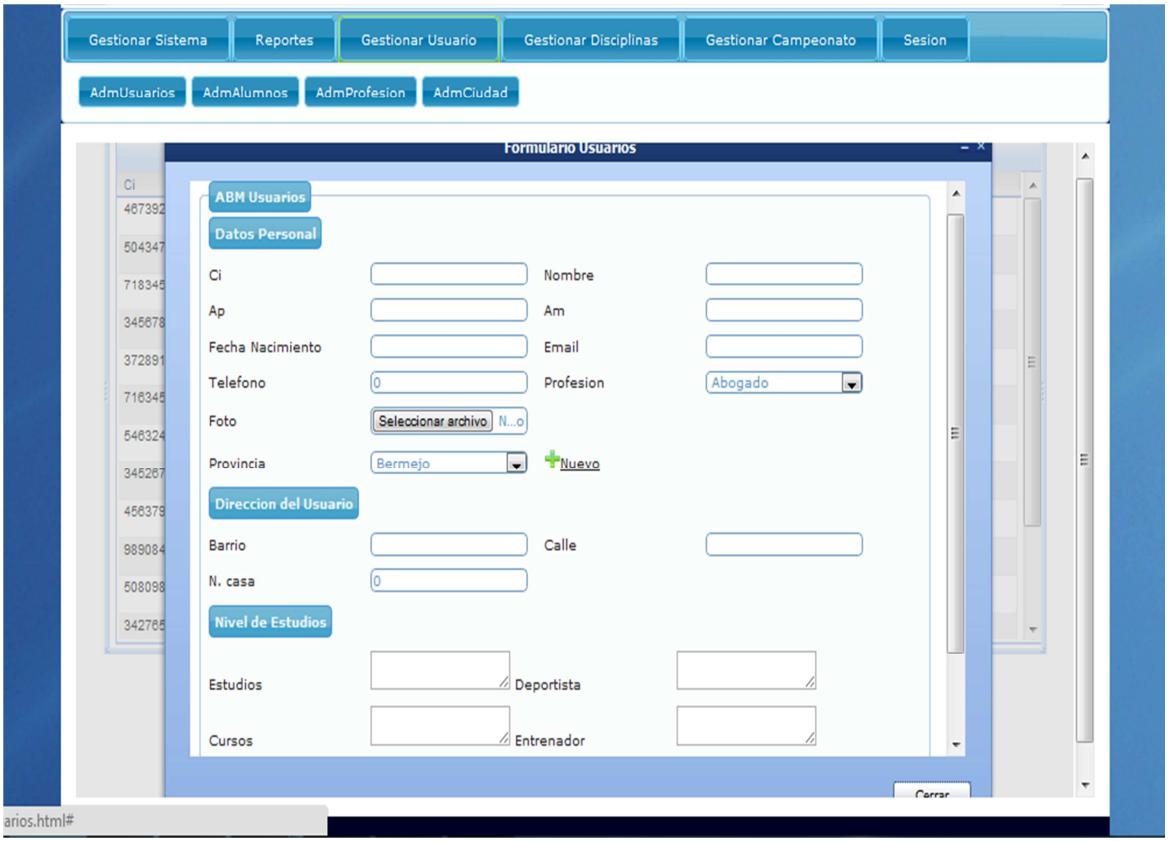


Figura 100 Pantalla de adicionar un nuevo Usuario

II.1.7.4.6 Pantalla 6: Administrar usuario

Haciendo doble clic en el usuario seleccionado y se visualiza la pantalla modificar.

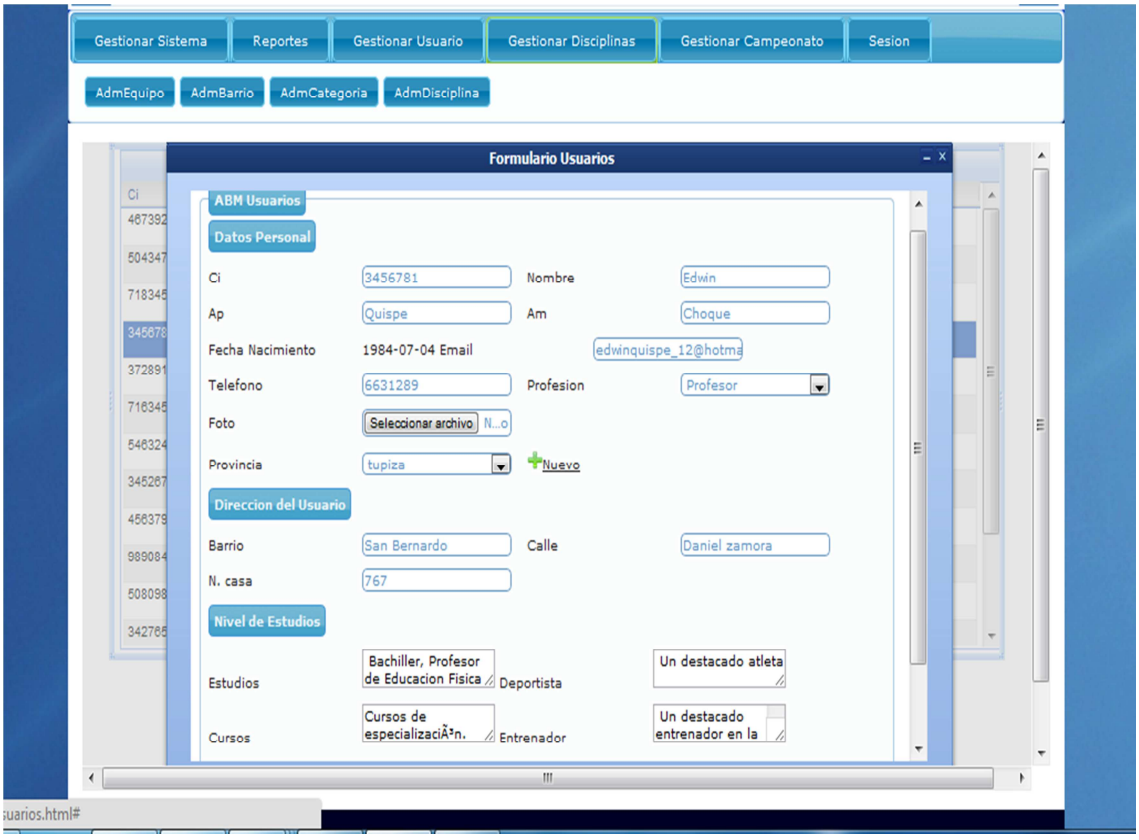


Figura 101 Pantalla Modificar Usuario

I.1.7.4.7Pantalla 7: Administrar usuario

Mensaje de confirmación de los datos llenados

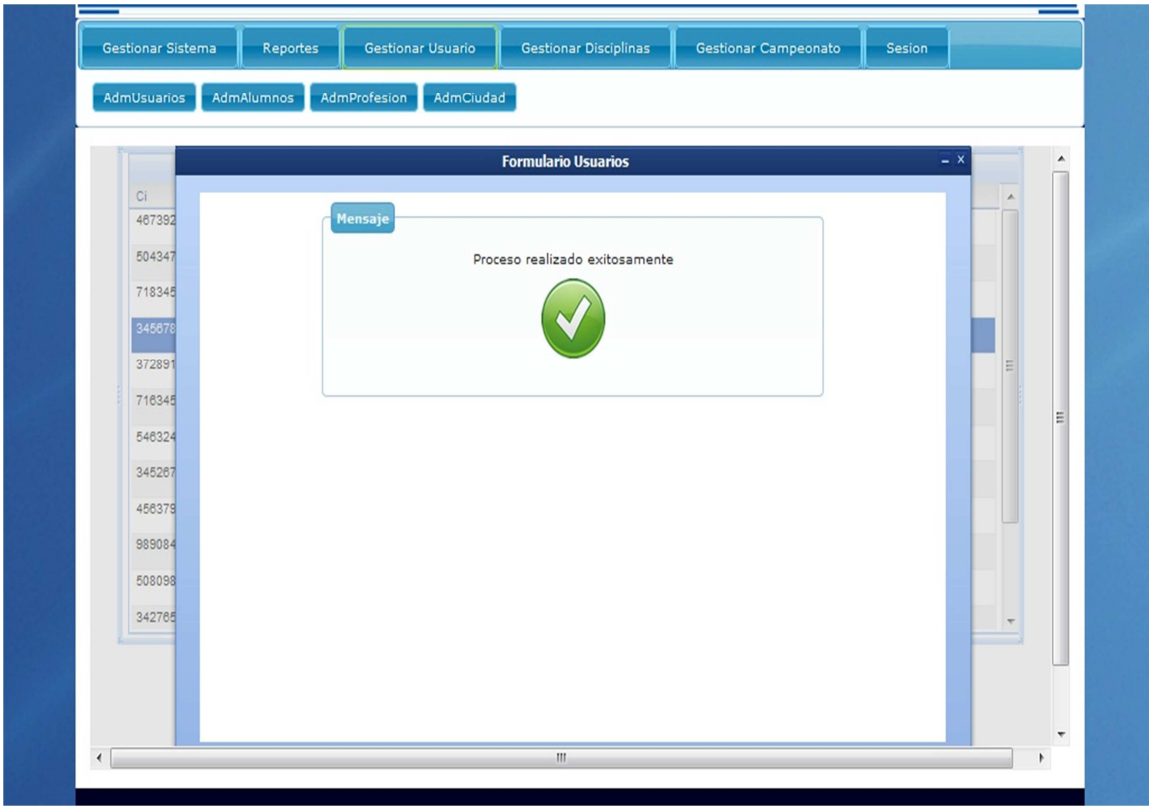


Figura 3002 **Pantalla de Mensaje de Confirmación**

I.1.7.4.8 Pantalla 8: Administrar usuarios

Ver toda su información personalizada de un usuario registrado

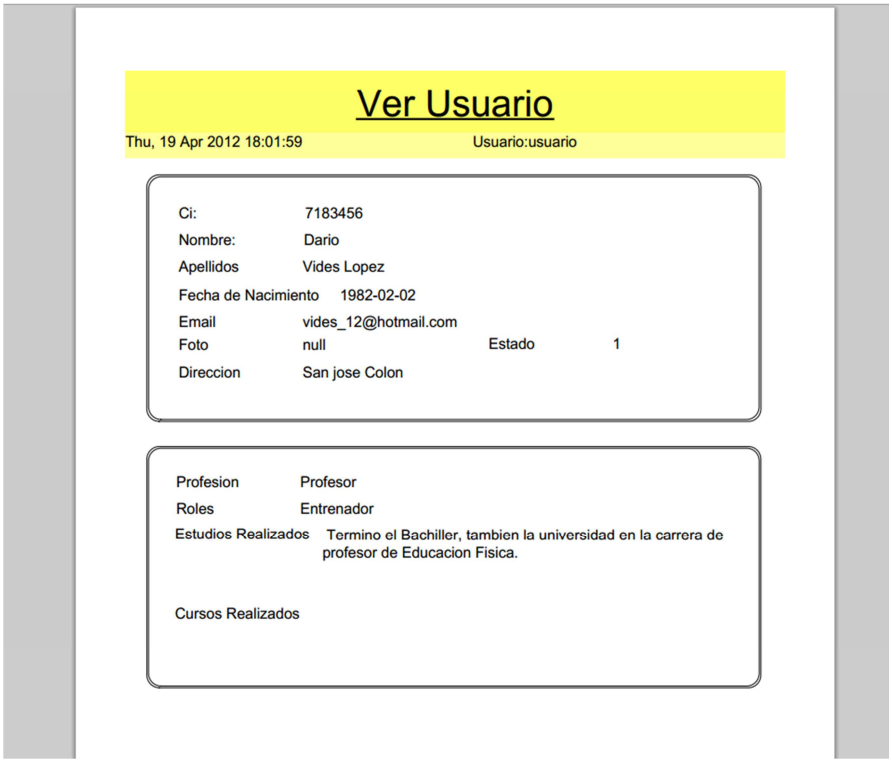


Figura 103 Pantalla Ver Usuario

I.1.7.4.9 Pantalla 9: Gestionar Usuarios

Administra Alumno:

En esta pantalla se visualiza la lista a todos los alumnos registrados.

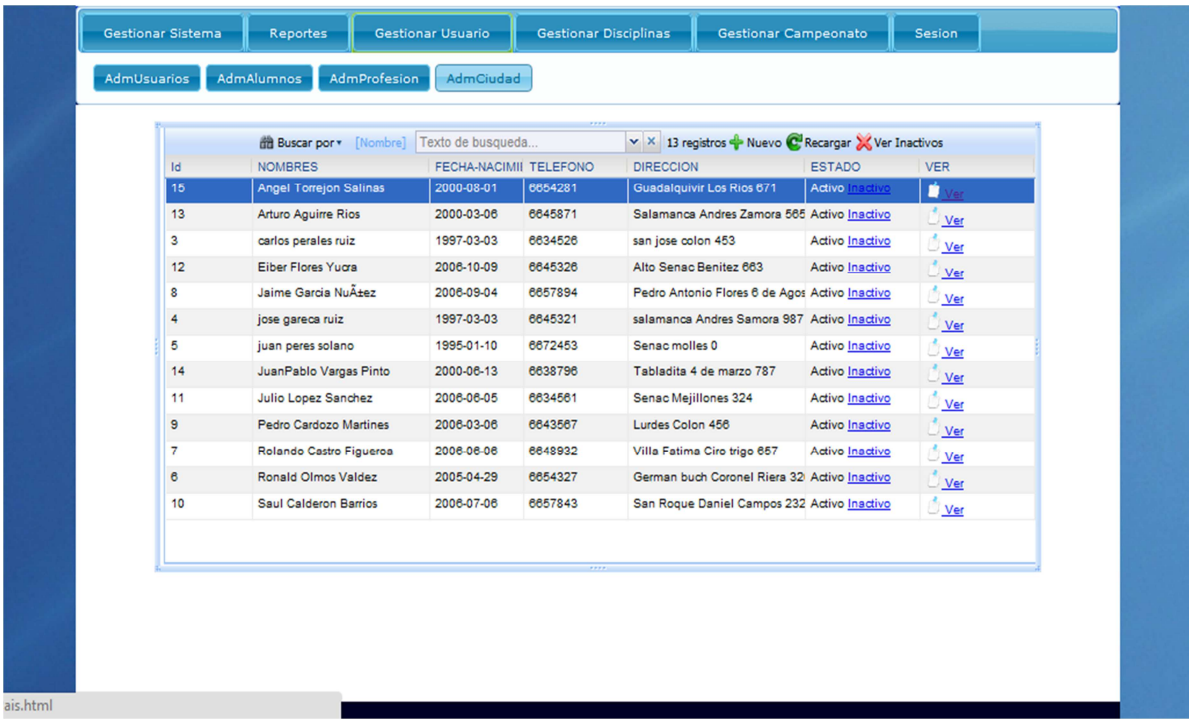


Figura 104 Pantalla Listar Alumno

I.1.7.4.10 Pantalla 10: Gestionar Usuarios

Administrar Alumno:

Adicionar a un nuevo alumno

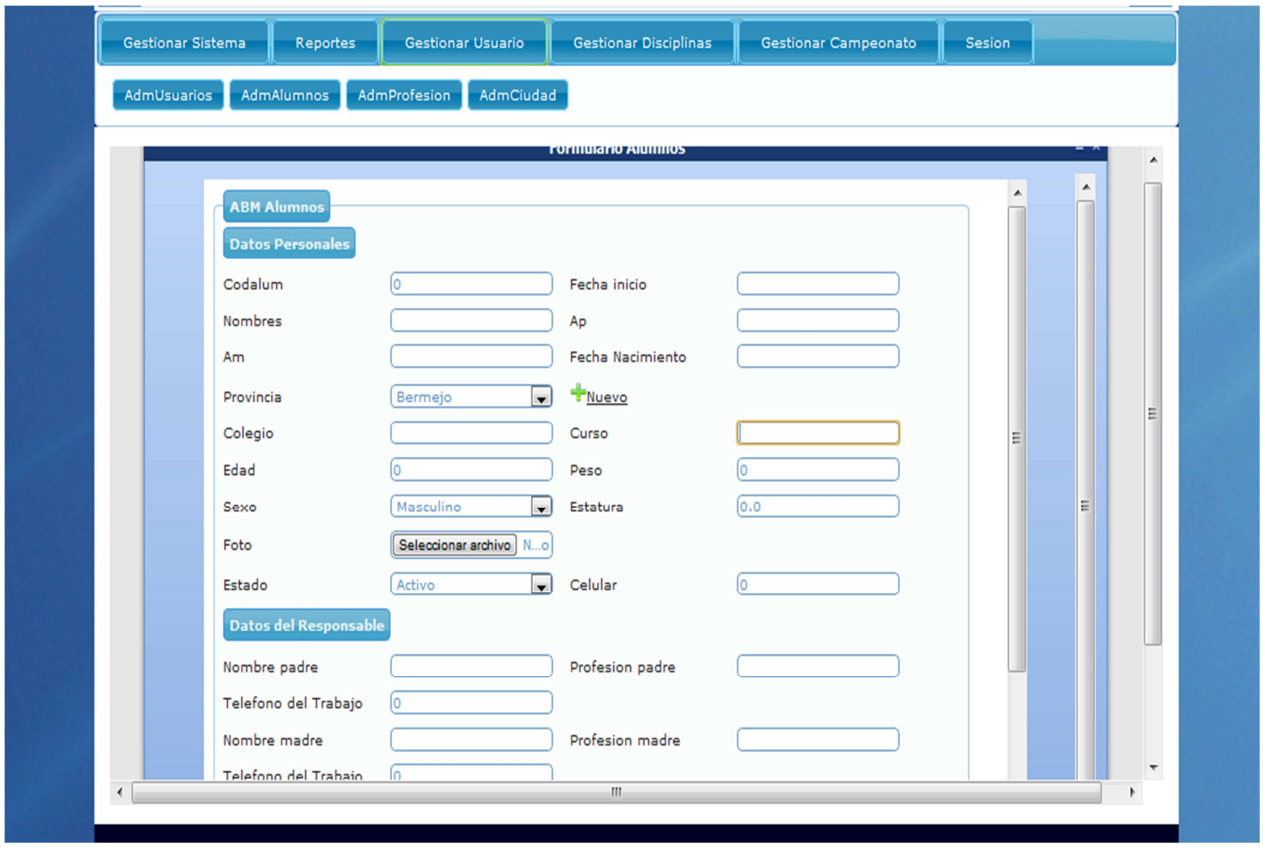


Figura 105 Pantalla Adicionar un nuevo Alumno

I.1.7.4.11 Pantalla 11: Administrar usuario

Listar Alumno:

Validando los datos

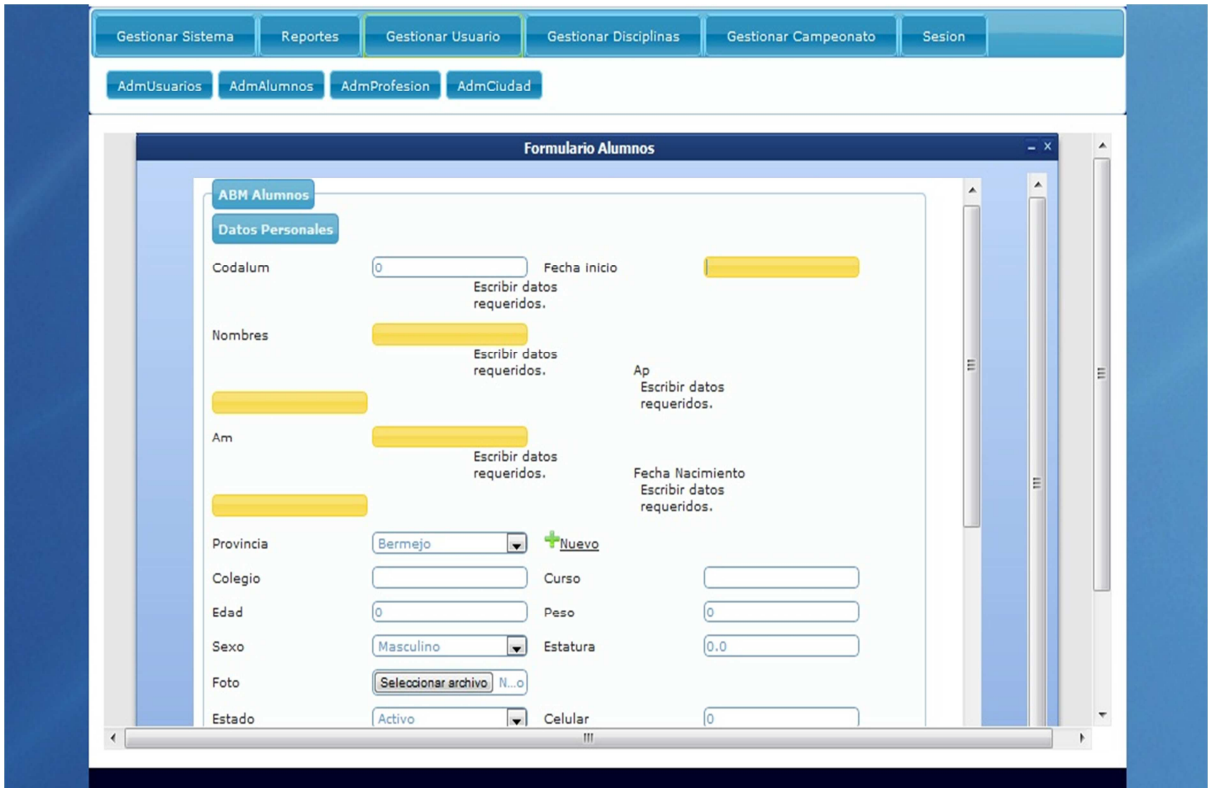


Figura 106 **Pantalla de validación de datos del Alumno**

I.1.7.4.12 Pantalla 12: Gestionar Usuario

Administrar Alumno:

Ver su información del alumno registrado

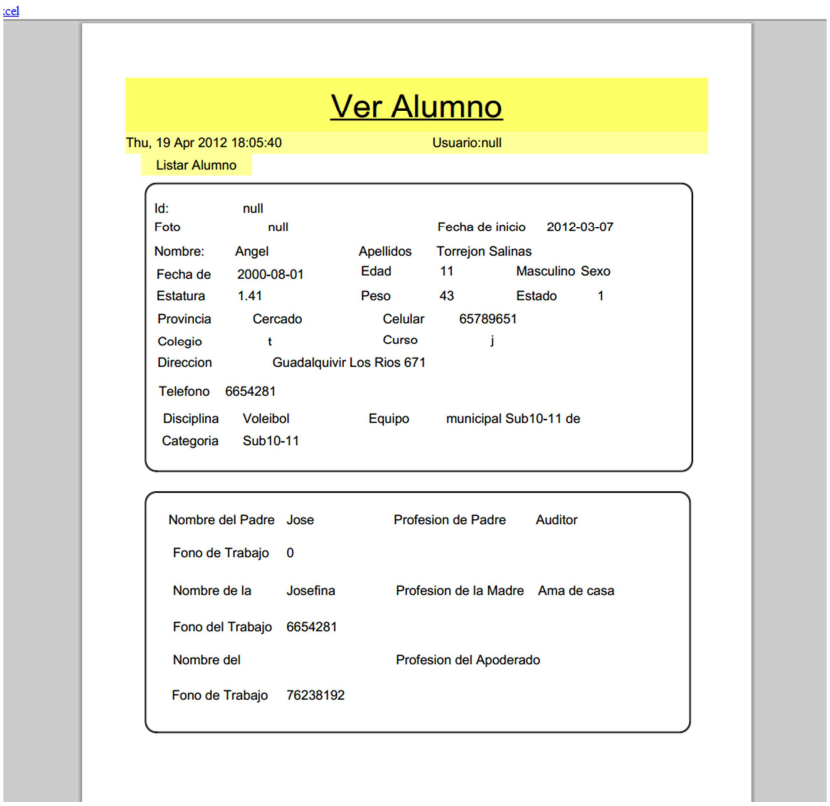


Figura 107 Pantalla Ver Alumno

I.1.7.4.13 Pantalla 13: Gestionar Usuario

Administrar Alumno:

Lista de alumnos inactivos

Figura 108 Pantalla Listar Alumno Inactivos

I.1.7.4.14 Pantalla 14: Gestionar Sistema

Administrar Publicaciones:

Lista de todas las publicaciones cargadas.

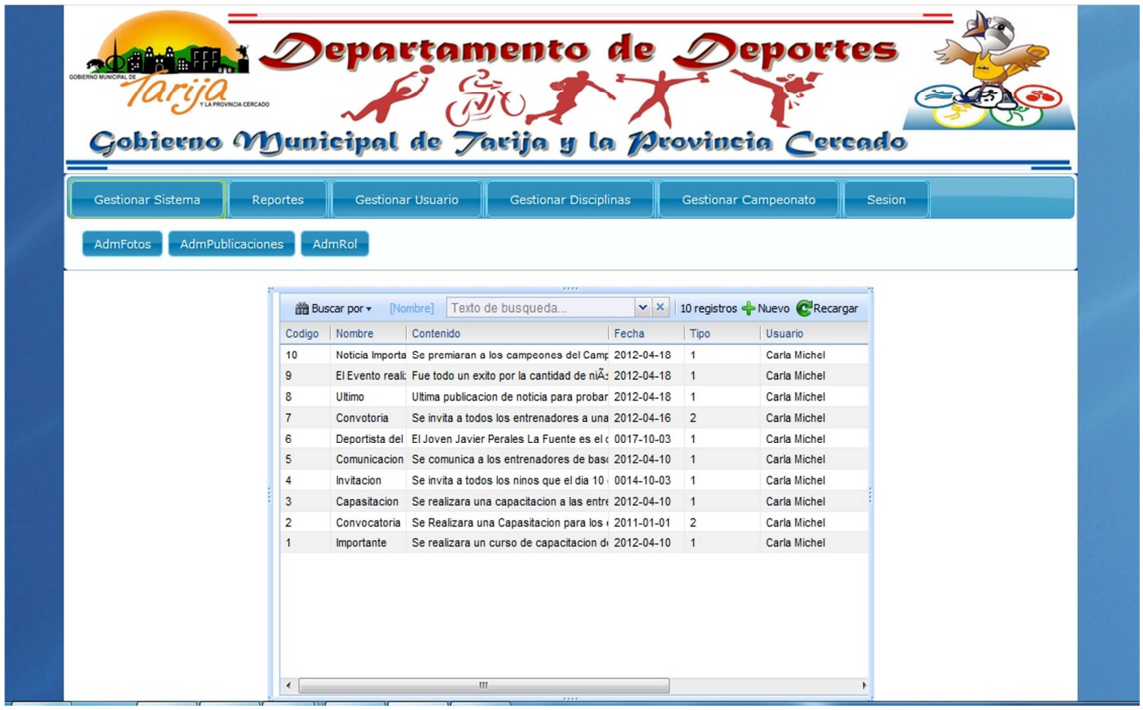


Figura 109 Pantalla Administrar Publicaciones

I.1.7.4.15 Pantalla 15: Gestionar Sistema

Administrar Publicaciones:

Visualiza la pantalla de nueva Publicación

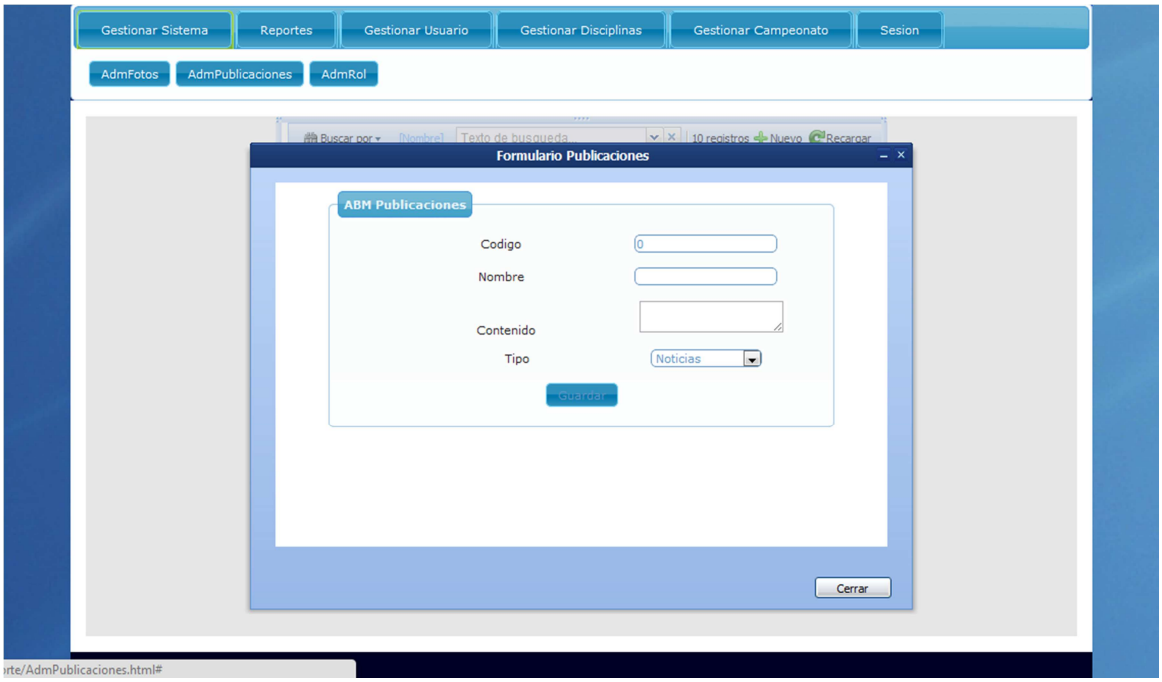


Figura 110 Pantalla Adicionar nueva Publicaciones

I.1.7.4.16 Pantalla 16: Administrar Sistema

Administrar Rol:

Visualiza la lista de Roles del sistema.

Figura 111 Pantalla Administrar Rol

I.1.7.4.17 Pantalla 17: Gestionar Sistema

Administrar Rol:

Visualiza la pantalla de asignar procesos

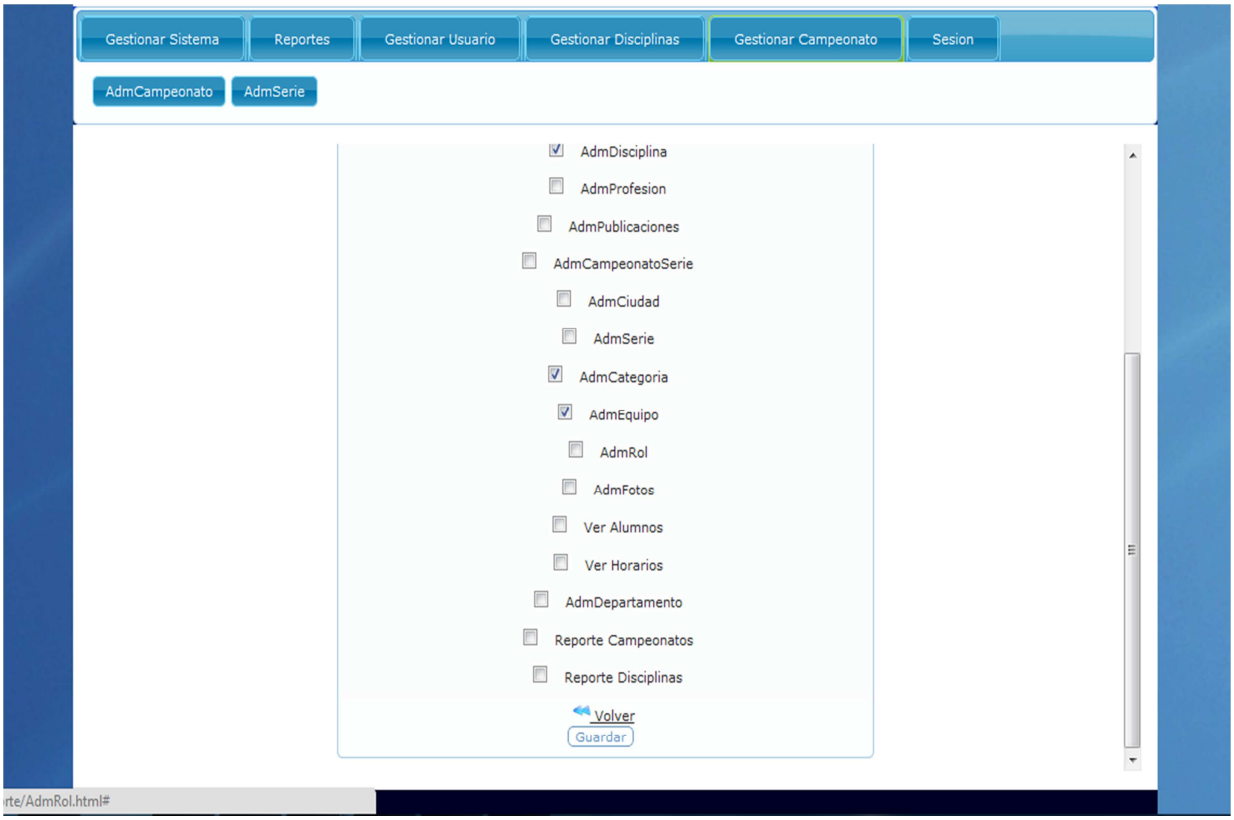


Figura 112 Pantalla de Asignar procesos

I.1.7.4.18 Pantalla 18: Gestionar Disciplina

Administrar Disciplina

Visualiza la lista de disciplinas que tiene el Departamento de Deportes.

Figura 113 Pantalla Listar Disciplina

I.1.7.4.19 Pantalla 19: Gestionar Disciplina

Administrar Disciplina

Pantalla de Nuevo Disciplina

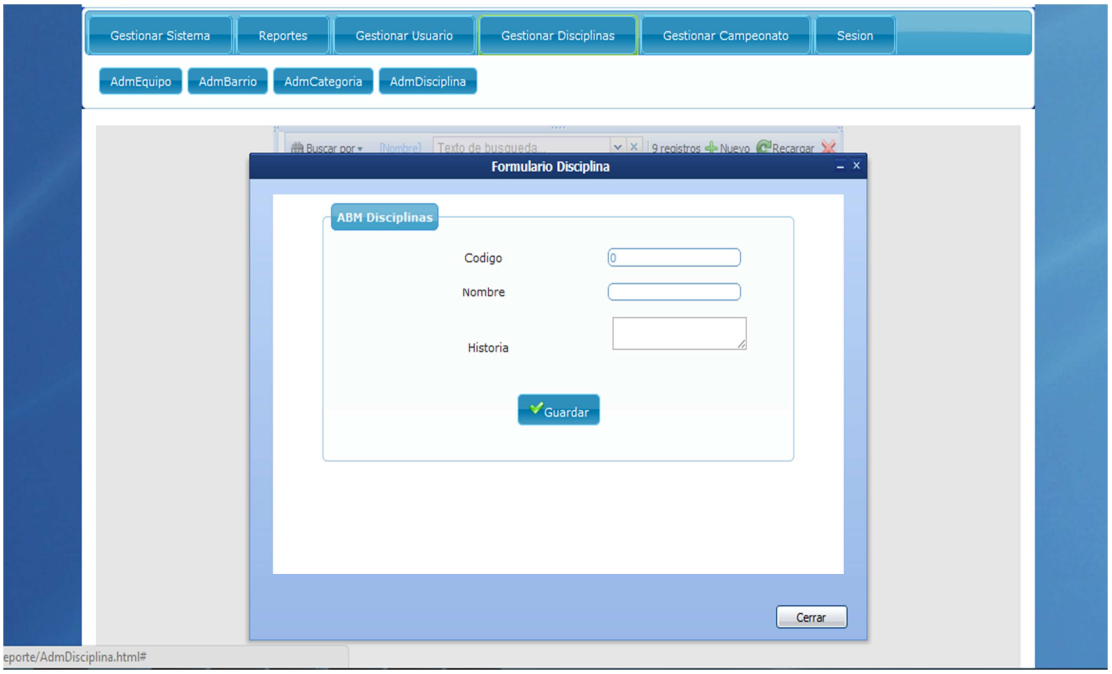


Figura 114 Pantalla Adicionar una nueva Disciplina

I.1.7.4.20 Pantalla 20: Gestionar Disciplina

Administrar Equipo

Visualiza la pantalla de listar todos los Equipos

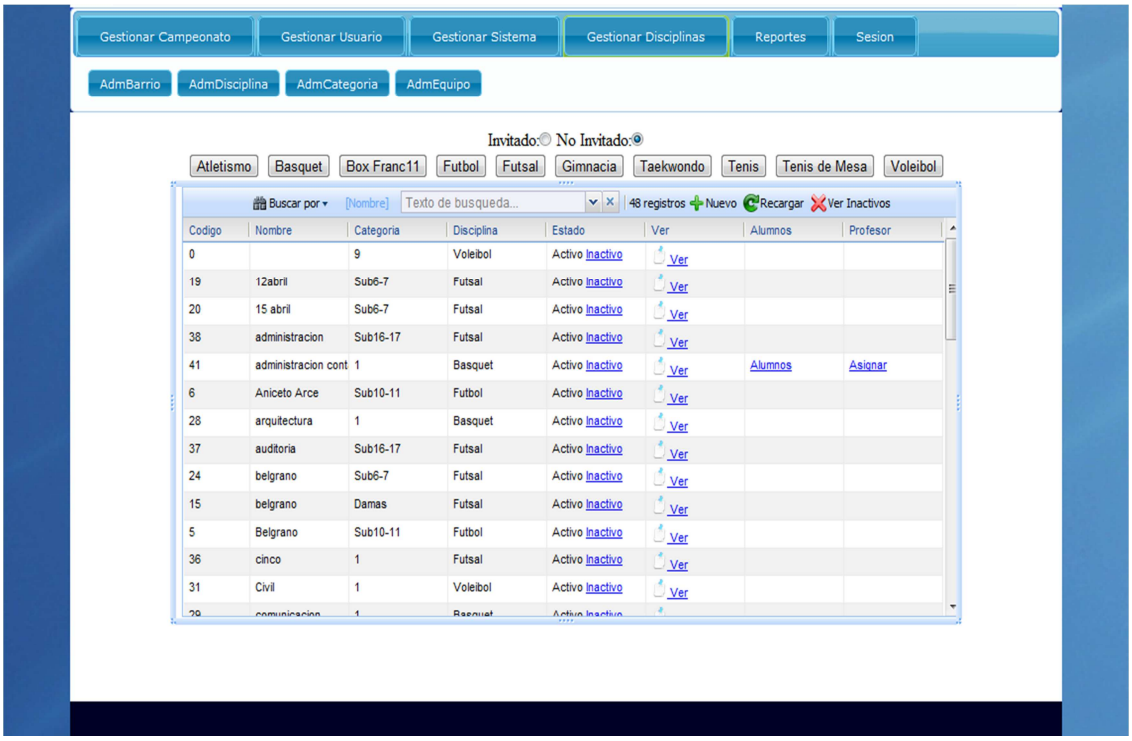


Figura 115 Pantalla Listar Equipos

.1.7.4.21 Pantalla 21: Gestionar Disciplina

Administrar Equipo:

Visualiza la pantalla para nuevo equipo

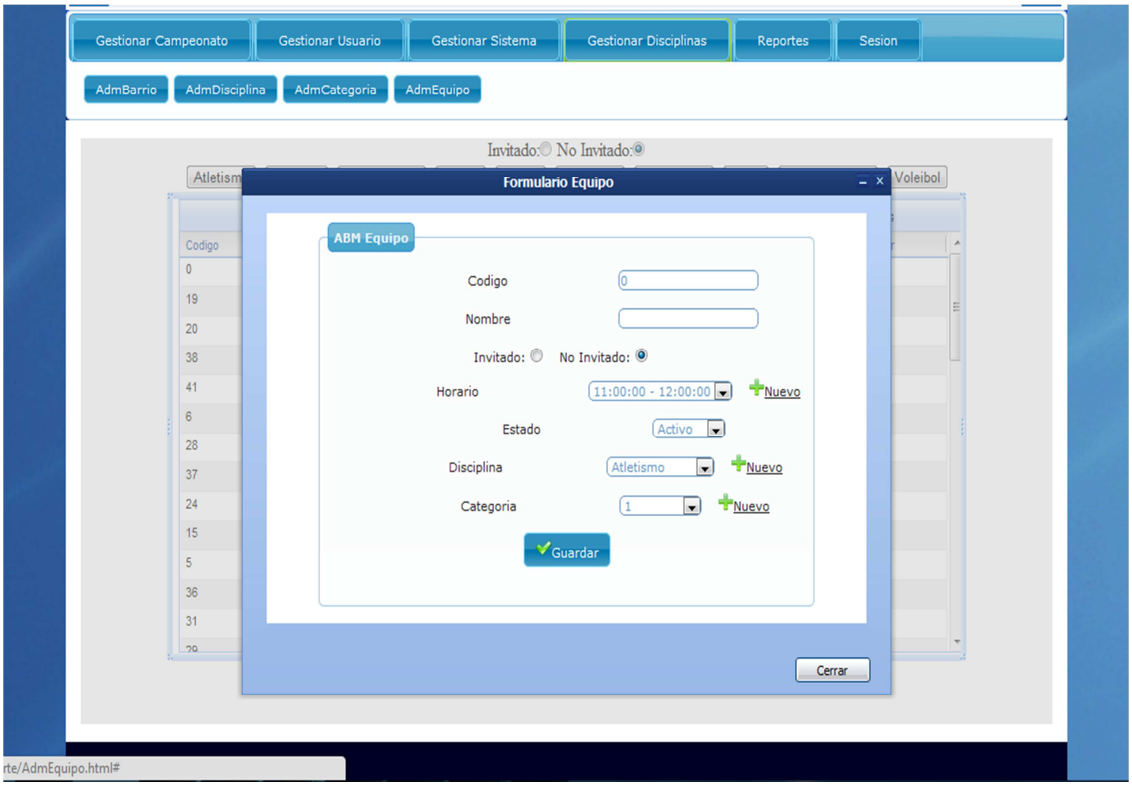


Figura 116 Pantalla Adicionar un nuevo Equipo

I.1.7.4.22 Pantalla 22: Gestionar Disciplina

Administrar Equipo:

Valida los campos

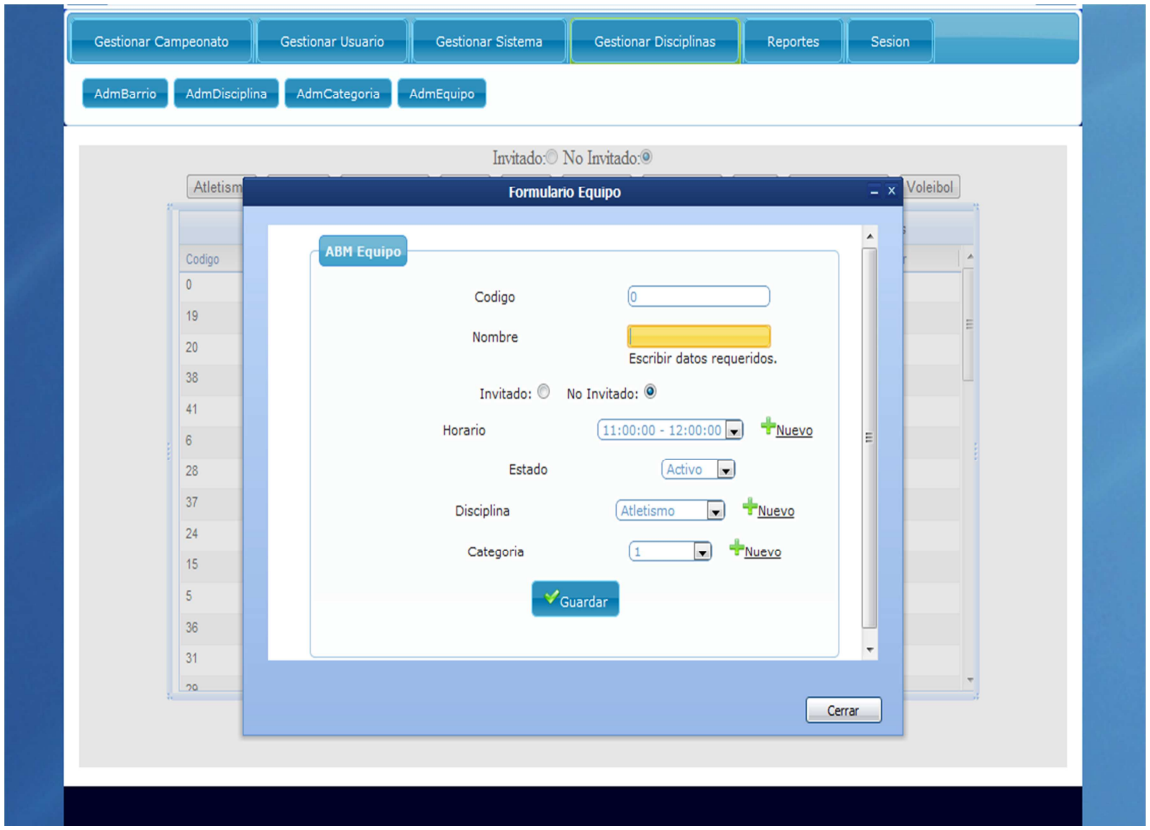


Figura 117 Pantalla Validación de datos

I.1.7.4.23 Pantalla 23: Gestionar Campeonato

Administrar Campeonato:

Visualiza el listado de los campeonatos realizados

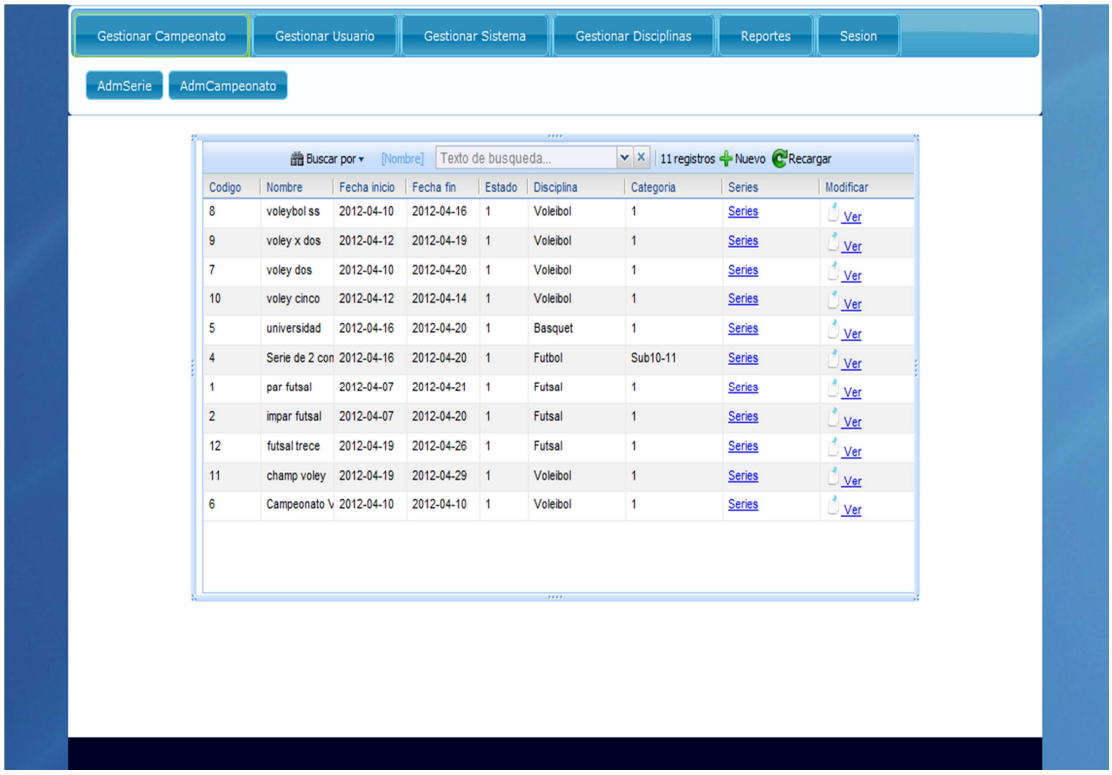


Figura 118 Pantalla de Listar Campeonato

I.1.7.4.24 Pantalla 24: Gestionar Campeonato

Administrar Campeonato:

Se visualiza la pantalla para nuevo campeonato

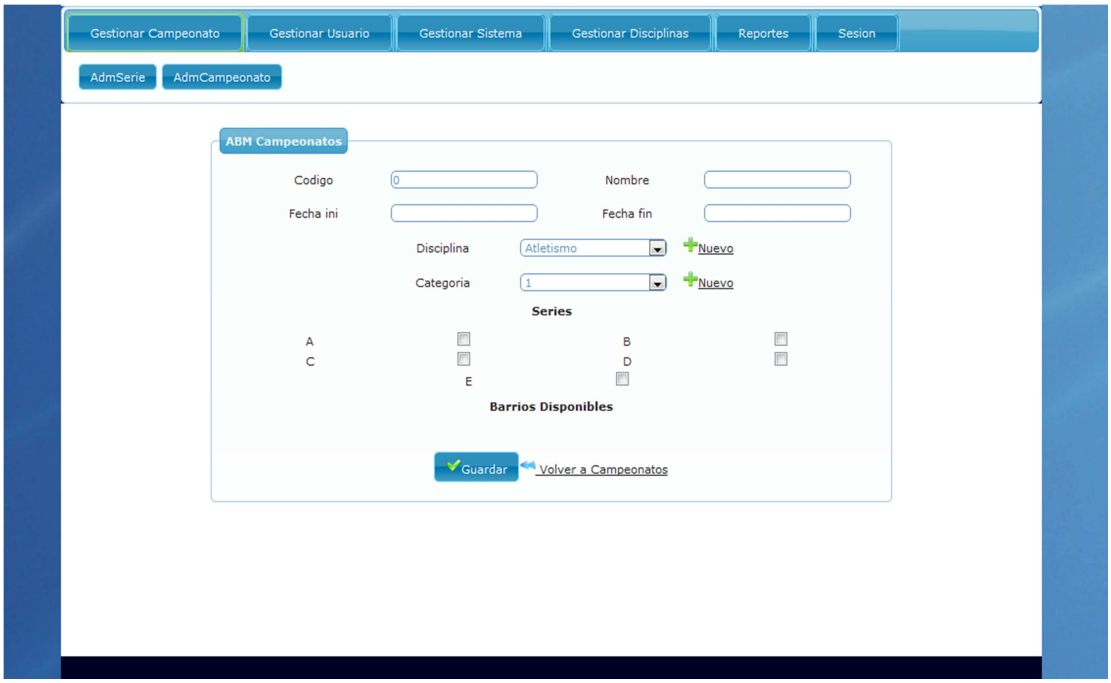


Figura 119 Pantalla Adicionar un nuevo Campeonato

I.1.7.4.25 Pantalla 25: Gestionar Campeonato

Administrar Campeonato:

Es la lista de Series para un nuevo campeonato

Figura 120 Pantalla Listar las Series

I.1.7.4. 26 Pantalla 26: Gestionar Campeonato

Administrar Campeonato:

Se genera automáticamente los roles de partidos en diferentes fechas para un campeonato asignado.

Figura 121 Pantalla de Insertar Resultados

I.1.7.4.27 Pantalla 27: Gestionar Campeonato

Administrar Campeonato:

Se visualiza la tabla de posiciones de acuerdo vayan jugando los equipos, se actualiza automáticamente dando a conocer qué equipo va ganando

Figura 3122 Pantalla Tabla de Posiciones

I.1.7.4.28 Pantalla 28: Administrar Campeonato

Reporte

Todos los campeonatos realizados.

Figura 3223 Pantalla Reporte de Campeonatos

I.1.7 Modelo de Implementación

I.1.7.1 Introducción

La implementación se empieza con el resultado del diseño y se implementa el sistema en términos de componentes, es decir ficheros de códigos fuente, scripts, tablas de la base de datos y otros similares.

I.1.7.2 Propósito

El propósito de la implementación de la arquitectura es imaginar el modelo de implementación y su arquitectura mediante identificación de componentes significativos arquitectónicamente.

I.1.7.3 Alcance

Mostar como distintos subsistemas de software conforman la estructura general del sistema, que se crea en una base de datos centralizada.

I.1.8 Modelo de Despliegue

I.1.8.1 Diagrama de Desplazamiento

I.1.8.2 Introducción

El modelo de despliegue es el que representa o muestra la parte física de la arquitectura del sistema que se está modelando.

I.1.8.3 Propósito

Modelar la arquitectura del sistema.

I.1.2.4 Diagrama de Despliegue

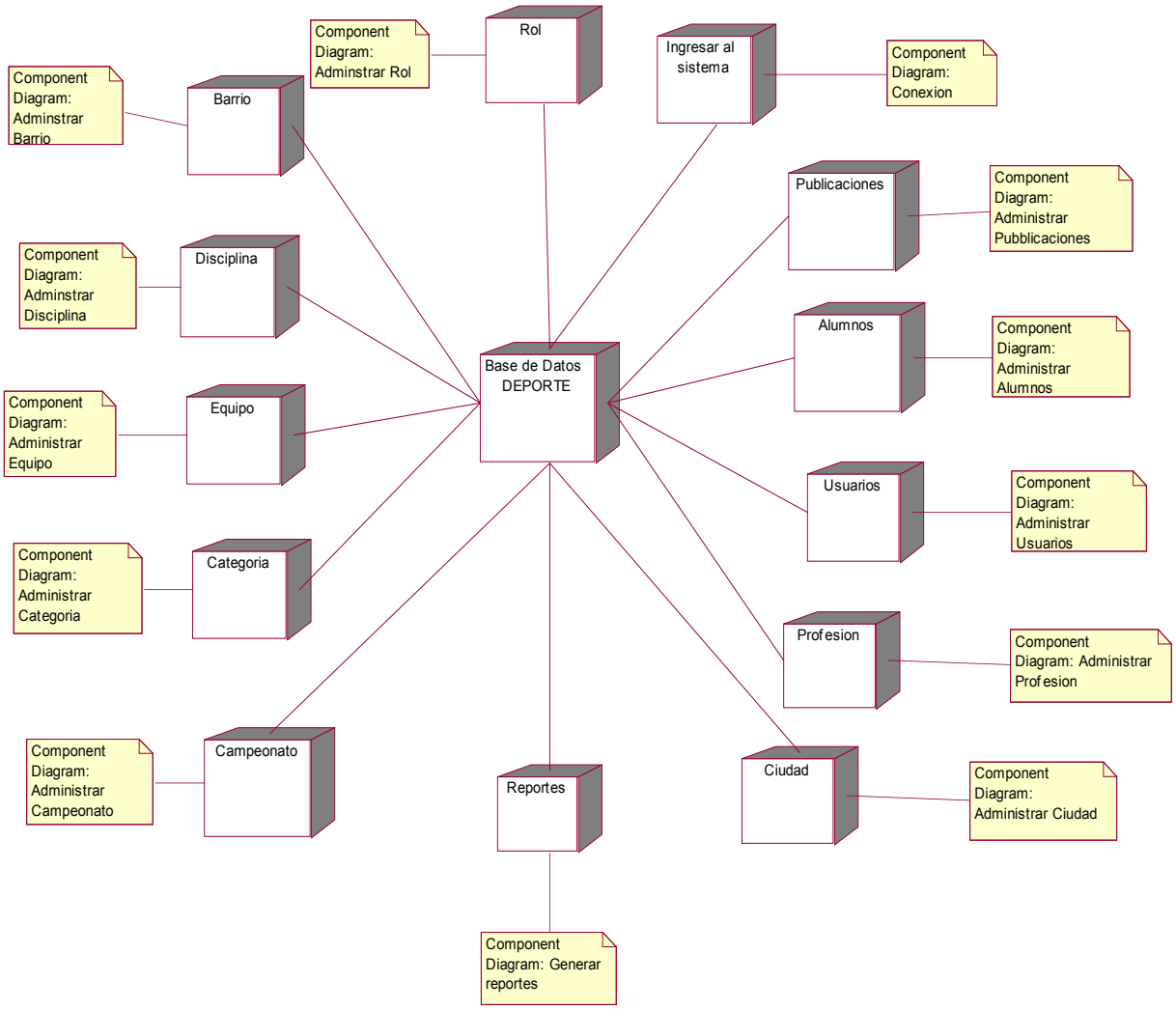


Figura 124 Diagrama de Despliegue

I.1.9 Casos de Prueba

I.1.9.1 Introducción

Las pruebas de software son un artefacto de la disciplina en la metodología RUP la cual se está implementando.

Cada prueba es especificada mediante un documento que establece las condiciones de ejecución, las entradas de las pruebas y los resultados esperados estos casos de prueba llevara asociado un procedimiento prueba con las instrucciones para realizar la misma y dependiendo del tipo que se utilice dicho procedimiento podrá ser automatizado mediante un script de prueba.

I.1.9.2 Propósito

- Verificar el correcto funcionamiento del sistema.
- Identificar posibles mejoras.

I.1.9.3 Alcance

Describe la funcionalidad de cada componente individualmente una vez codificado.

I.1.9.4 Pruebas de Caja Blanca

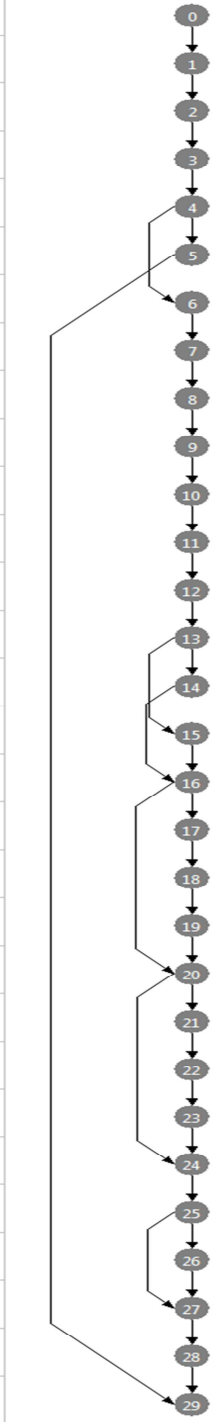
En programación, se denomina cajas blancas a un tipo de pruebas de software que se realiza sobre las funciones internas de un módulo. Así como las pruebas de caja negra ejercitan los requisitos funcionales desde el exterior del módulo, las de caja blanca están dirigidas a las funciones internas. Entre las técnicas usadas se encuentran; la cobertura de caminos pruebas sobre las expresiones lógico-aritméticas, pruebas de camino de datos.

En los sistemas orientados a objetos, las pruebas de caja blanca pueden aplicarse a los métodos de la clase, pero según varias opiniones, ese esfuerzo debería dedicarse a otro tipo de pruebas más especializadas (un argumento podría ser que los métodos de una clase suelen ser menos complejos que los de una función de programación estructurada). Dentro de las Pruebas de Caja Blanca encontramos las llamadas coberturas (sentencia, decisión, condición y múltiple además de los mencionados caminos ciclomáticos). QUE ES LO Q SE PRUEBA

I.1.9.5 Pruebas de Caja Blanca

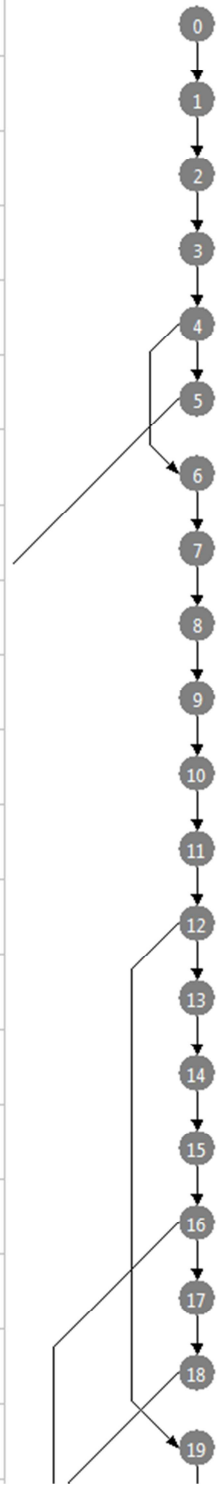
CLASE: UsuariosControlador.java

METODO: admusuarios

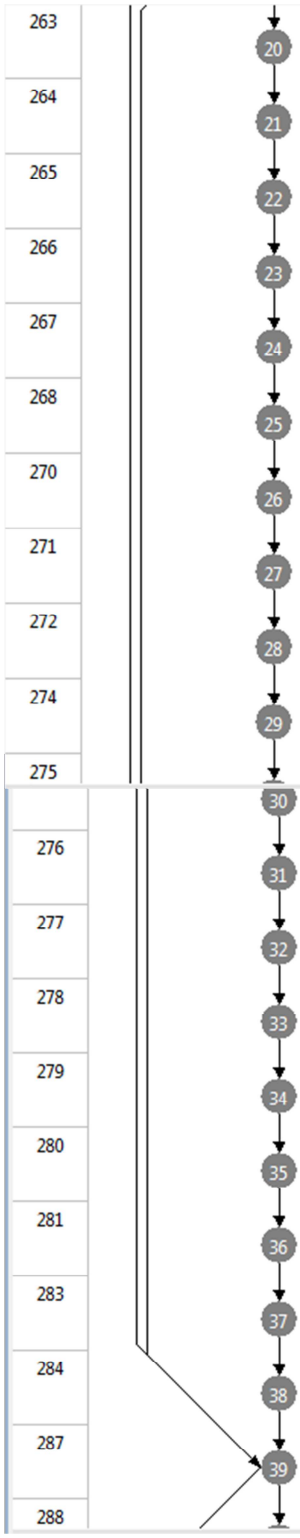
187		1	u(xci), u(estado), u(ssesion), u(session), u(map), u(usuario)	public ModelAndView admusuarios(HttpServletRequest
187		2		public ModelAndView admusuarios(HttpServletRequest
188		3	d(session)	HttpSession session=request.getSession(true);
189		4	d(xci), r(session)	String xci= (String) session.getAttribute("ci");
190		5, 6	r(xci)	if(xci==null)
193		29		return new ModelAndView("Salir");
197		7	d(ssesion)	Registroacciones ssesion=new Registroacciones();
198		8	r(ssesion)	ssesion.setFecha(new Date());
199		9	r(ssesion)	ssesion.setHora(new Date());
200		10	r(xci), r(ssesion)	ssesion.setUsuarios(this.usuariosDao.getUsuarios(Integer.parseInt(xci)));
201		11	r(ssesion)	ssesion.setAccion("lista de usuarios ");
202		12	r(ssesion)	this.registroaccionesDao.guardarRegistroacciones(ssesion);//
204		13	d(map)	Map map = new HashMap();
207		14, 15		if(request.getParameter("filtro")==null){
208		16	r(map)	map.put("datos",this.usuariosDao.getAllUsuarios());
210		16	r(map)	map.put("datos",this.usuariosDao.getListaUsuariosFiltro(request.getPar
213		17, 20		if(request.getParameter("baja")!=null)
215		18	d(usuario)	Usuarios usuario=this.usuariosDao.getUsuari
216		19	r(usuario)	usuario.setEstado(0);
217		20	r(usuario)	this.usuariosDao.modificarUsuarios(usuario);
219		21, 24		if(request.getParameter("alta")!=null)
221		22		Usuarios usuario=this.usuariosDao.getUsuari
222		23		usuario.setEstado(1);
223		24		this.usuariosDao.modificarUsuarios(usuario);
225		25	d(estado)	int estado=1;
226		26, 27		if(request.getParameter("estado")!=null){
227		27	d(estado)	estado=Integer.parseInt(request.getParameter("estado"));
229		28	r(estado), r(map)	map.put("estado", estado);
230		29	r(map)	return new ModelAndView("AdmUsuarios",ma
232			u(xci), u(estado), u(ssesion), u(session), u(map), u(usuario)	}

CLASE: UsuariosControlador.java

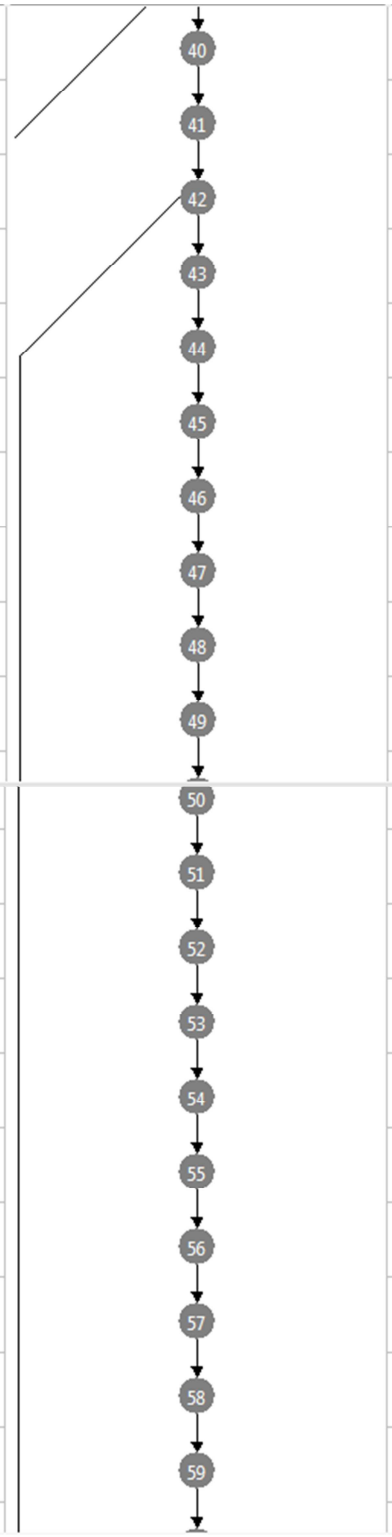
METODO: abmusuarios



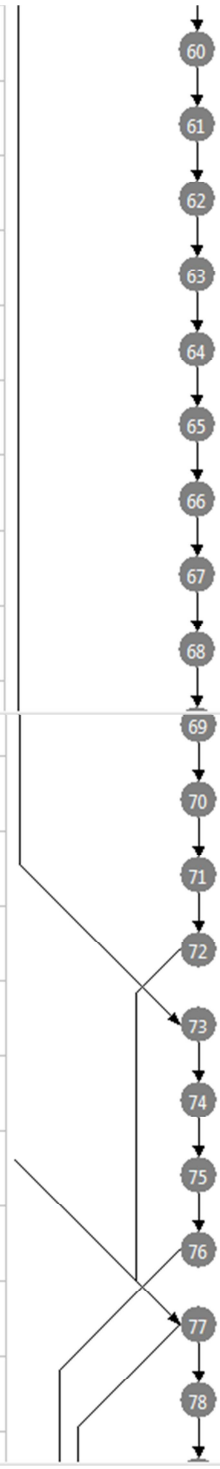
236	1	u(dir), u(ver), u(direccion), u(simpleDateFormat), u(date),	public ModelAndView abmusuarios(HttpServletRequest
236	2		public ModelAndView abmusuarios(HttpServletRequest
237	3	d(session)	HttpSession session=request.getSession(true);
238	4	d(xci), r(session)	String xci= (String) session.getAttribute("ci");
239	5, 6	r(xci)	if(xci==null)
242	99		return new ModelAndView("Salir");
244	7	d(map)	Map map = new HashMap();
245	8		Usuarios usuarios;
248	9	d(ssesion)	Registroacciones ssesion=new Registroacciones(); //
249	10	r(ssesion)	ssesion.setFecha(new Date());
250	11	r(ssesion)	ssesion.setHora(new Date());
251	12	r(xci), r(ssesion)	ssesion.setUsuarios(this.usuariosDao.getUsuarios(Integer.parseInt(xci)))
253	13, 19		if(null != request.getParameter("ci") & null ==
254	14	d(usuarios)	usuarios=this.usuariosDao.getUsuarios(Integer.parseInt(request.getPara
255	15	r(map), r(usuarios)	map.put("Usuarios", usuarios);
256	16	r(map), r(usuarios)	map.put("Direccion",usuarios.getDireccion());
257	17, 39		if(null != request.getParameter("ver")){
258	18	d(ver)	int ver=1;
259	39	r(ver), r(map)	map.put("ver", ver);
262	20	d(usuarios)	usuarios=new Usuarios();



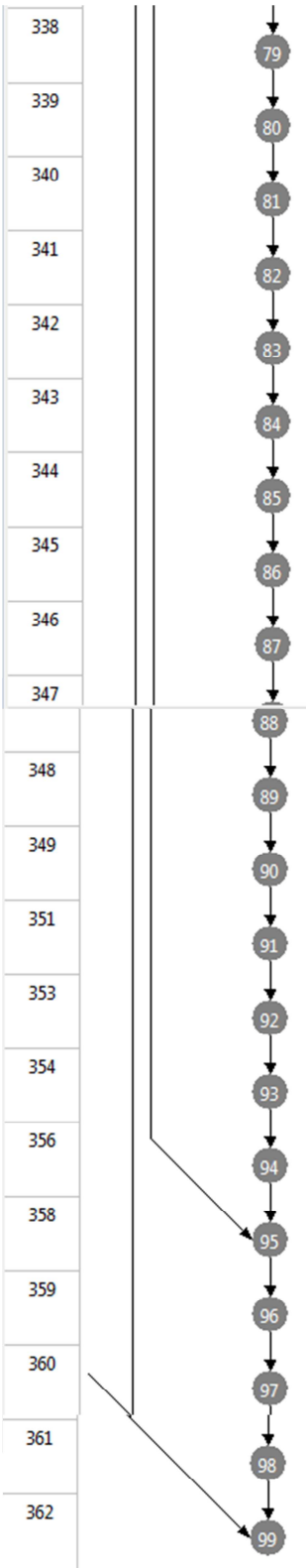
263	21	r(usuarios)	usuarios.setCi("");
264	22	r(usuarios)	usuarios.setNombre("");
265	23	r(usuarios)	usuarios.setAp("");
266	24	r(usuarios)	usuarios.setAm("");
267	25	r(usuarios)	usuarios.setEmail("");
268	26	d(direccion)	Direccion direccion=new Direccion();
270	27	r(direccion)	direccion.setBarrio("");
271	28	r(direccion)	direccion.setCalle("");
272	29	r(direccion)	direccion.setNumerocasa(0);
274	30	r(direccion), r(usuarios)	usuarios.setDireccion(direccion);
275	31	r(usuarios)	usuarios.setEstado((short)1);
276	32	d(date)	Date date = new Date();
277	33	d(simpleDateFormat)	SimpleDateFormat simpleDateFormat = new SimpleDateFormat("dd-MM-yyyy");
278	34	r(simpleDateFormat)	simpleDateFormat.setLenient(false);
279	35	r(simpleDateFormat), r(date), d(fnacdate)	String fnacdate = simpleDateFormat.format(date);
280	36	r(simpleDateFormat), r(fnacdate), r(usuarios)	usuarios.setFechadnac(simpleDateFormat.parse(fnacdate));
281	37	r(usuarios)	usuarios.setFoto("");
283	38	r(direccion), r(map)	map.put("Direccion",direccion);
284	39	r(map), r(usuarios)	map.put("Usuarios",usuarios);
287	40, 77		if(null != request.getParameter("save")){
288	41		System.out.println("[Guardando



288	41		System.out.println("[Guardando]");
289	42	d(usuarios)	usuarios=new Usuarios();
290	43, 73		if(this.usuariosDao.getUsuarios(Integer.parseInt(request.getParameter("ci"))).getCi().equals("00000"))
292	44	r(usuarios)	usuarios.setCi(request.getParameter("ci"));
293	45	r(usuarios)	usuarios.setNombre(request.getParameter("nombre"));
294	46	r(usuarios)	usuarios.setAp(request.getParameter("ap"));
295	47	r(usuarios)	usuarios.setAm(request.getParameter("am"));
296	48	r(usuarios)	usuarios.setEstado((short)1);
297	49	r(usuarios)	usuarios.setEmail(request.getParameter("email"));
298	50	r(usuarios)	usuarios.setTelefono(Integer.parseInt(request.getParameter("telefono")));
299	51	d(fecha)	SimpleDateFormat fecha = new SimpleDateFormat("dd-MM-yyyy");
300	52	r(fecha)	fecha.setLenient(true);
301	53	r(fecha), d(fnac)	Date fnac = fecha.parse(request.getParameter("fecha"));
302	54	r(usuarios), r(fnac)	usuarios.setFechadnac(fnac);
303	55	r(usuarios)	usuarios.setFoto(request.getParameter("foto"));
304	56	d(dir)	Direccion dir=new Direccion();
305	57	r(dir)	dir.setBarrio(request.getParameter("barrio"));
306	58	r(dir)	dir.setCalle(request.getParameter("calle"));
307	59	r(dir)	dir.setNumerocasa(Integer.parseInt(request.getParameter("casa")));
308	60	d(dir2), r(dir)	Direccion dir2=this.direccionDao.guardarDireccion(dir);
309	61	r(usuarios), r(dir2)	usuarios.setDireccion(dir2);



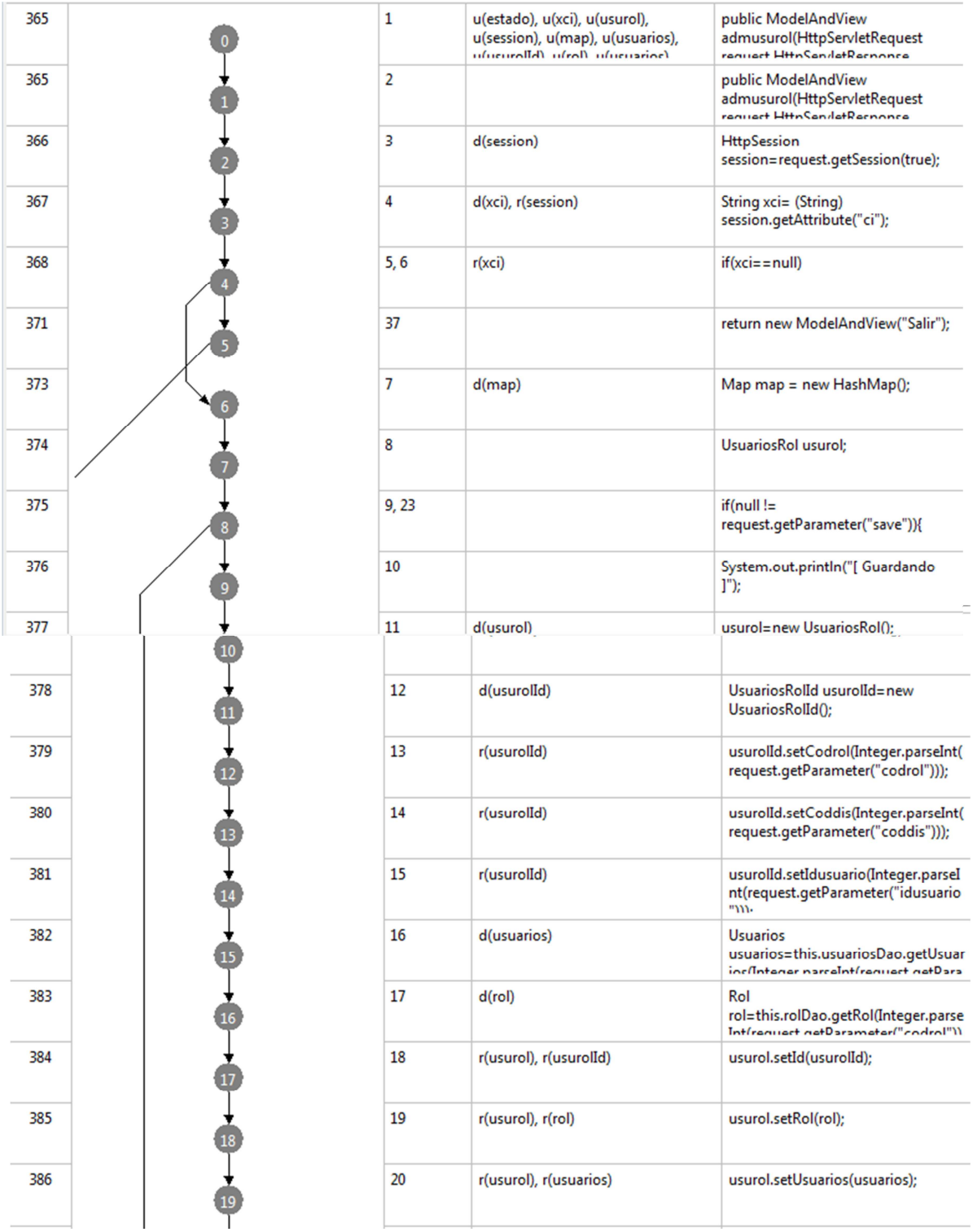
309	61	r(usuarios), r(dir2)	usuarios.setDireccion(dir2);
311	62	r(usuarios)	usuarios.setProvincia(this.provinciaDao.getProvincia(Integer.parseInt(re
313	63	d(curriculum)	Curriculum curriculum=new Curriculum();
314	64	r(curriculum)	curriculum.setCursos(request.getParameter("cursos"));
315	65	r(curriculum)	curriculum.setDeportista(request.getParameter("deportista"));
316	66	r(curriculum)	curriculum.setEntrenador(request.getParameter("entrenador"));
317	67	r(curriculum)	curriculum.setEstudios(request.getParameter("estudios"));
318	68	r(curriculum)	this.curriculumDao.guardarCurriculum(curriculum);
319	69	r(curriculum)	System.out.println("id de nuevo curriculum
320	70	r(usuarios), r(curriculum)	usuarios.setCurriculum(curriculum);
321	71	r(usuarios)	usuarios.setProfesion(this.profesionDao.getProfesion(Integer.parseInt(r
323	72	r(usuarios)	this.usuariosDao.guardarUsuarios(usuarios);
328	77		response.sendRedirect("mensaje.html");
330	74	r(map)	map.put("mensaje", "Error: Usuario Ya existe");
331	75	r(map)	map.put("listaprovincia", this.provinciaDao.getAllProvincia());
332	76	r(map)	map.put("listaprofesion", this.profesionDao.getAllProfesion());
333	99	r(map)	return new ModelAndView("abmUsuarios",map);
336	78, 95		if(null != request.getParameter("update")){
337	79		System.out.println("[Actualizando datos]");
338	80	d(usuarios)	usuarios=this.usuariosDao.getUsuar



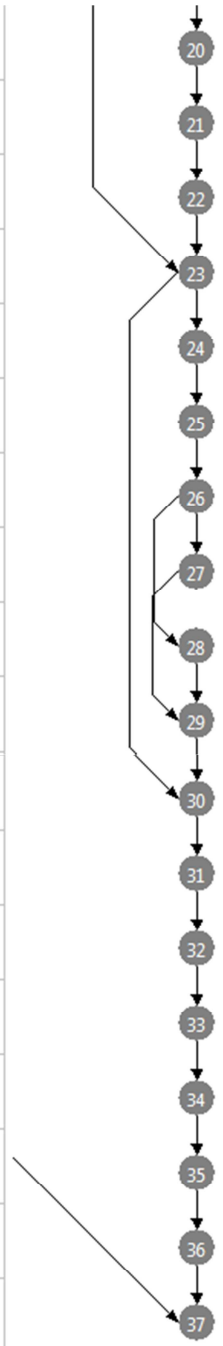
338	79	80	d(usuarios)	usuarios=this.usuariosDao.getUsuarios(Integer.parseInt(request.getPara
339	80	81	r(usuarios)	usuarios.setNombre(request.getParameter("nombre"));
340	81	82	r(usuarios)	usuarios.setAp(request.getParameter("ap"));
341	82	83	r(usuarios)	usuarios.setAm(request.getParameter("am"));
342	83	84	r(usuarios)	usuarios.setEmail(request.getParameter("email"));
343	84	85	r(usuarios)	usuarios.setTelefono(Integer.parseInt(request.getParameter("telefono"))
344	85	86	d(dir), r(usuarios)	Direccion dir=usuarios.getDireccion();
345	86	87	r(dir)	dir.setBarrio(request.getParameter("barrio"));
346	87	88	r(dir)	dir.setCalle(request.getParameter("calle"));
347	88	89	r(dir)	dir.setNumerocasa(Integer.parseInt(request.getParameter("casa")));
348	89	90	r(dir)	this.direccionDao.modificarDireccion(dir);
349	90	91	r(usuarios)	usuarios.setProvincia(this.provinciaDao.getProvincia(Integer.parseInt(re
351	91	92	r(usuarios)	this.usuariosDao.modificarUsuarios(usuarios);
353	92	93	r(ssesion), r(usuarios)	ssesion.setAccion("actualizado datos de usuario con id
354	93	94	r(ssesion)	this.registroaccionesDao.guardarRegistroacciones(ssesion);
356	94	95		response.sendRedirect("mensaje.html");
358	95	96	r(map)	map.put("mensaje", "");
359	96	97	r(map)	map.put("listaprovincia", this.provinciaDao.getAllProvincia());
360	97	98	r(map)	map.put("listaprofesion", this.profesionDao.getAllProfesion());
361	98	99	r(map)	return new ModelAndView("abmUsuarios",ma
362	99		u(dir), u(ver), u(direccion), u(simpleDateFormat), u(date),	}

CLASE: UsuariosControlador.java

METODO: admusurol



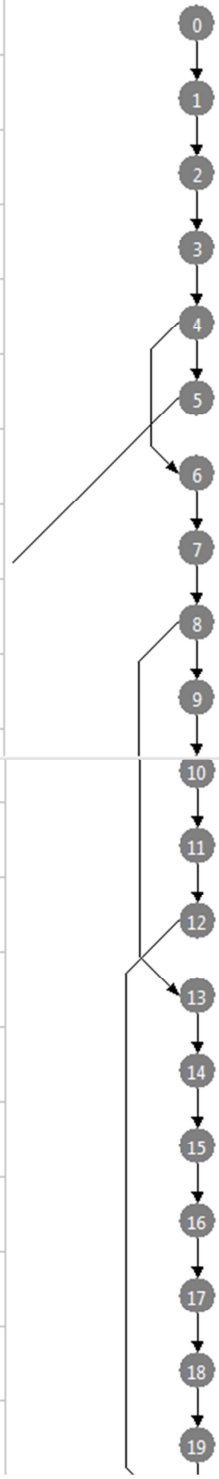
365	0	1	u(estado), u(xci), u(usurol), u(session), u(map), u(usuarios),	public ModelAndView admusurol(HttpServletRequest
365	1	2		public ModelAndView admusurol(HttpServletRequest
366	2	3	d(session)	HttpSession session=request.getSession(true);
367	3	4	d(xci), r(session)	String xci= (String) session.getAttribute("ci");
368	4	5, 6	r(xci)	if(xci==null)
371	5	37		return new ModelAndView("Salir");
373	6	7	d(map)	Map map = new HashMap();
374	7	8		UsuariosRol usurol;
375	8	9, 23		if(null != request.getParameter("save")){
376	9	10		System.out.println("[Guardando]");
377	10	11	d(usurol)	usurol=new UsuariosRol();
378	11	12	d(usurolId)	UsuariosRolId usurolId=new UsuariosRolId();
379	12	13	r(usurolId)	usurolId.setCodrol(Integer.parseInt(request.getParameter("codrol")));
380	13	14	r(usurolId)	usurolId.setCoddis(Integer.parseInt(request.getParameter("coddis")));
381	14	15	r(usurolId)	usurolId.setIdusuario(Integer.parseI nt(request.getParameter("idusuario
382	15	16	d(usuarios)	Usuarios usuarios=this.usuariosDao.getUsuar
383	16	17	d(rol)	Rol rol=this.rolDao.getRol(Integer.parse
384	17	18	r(usurol), r(usurolId)	usurol.setId(usurolId);
385	18	19	r(usurol), r(rol)	usurol.setRol(rol);
386	19	20	r(usurol), r(usuarios)	usurol.setUsuarios(usuarios);



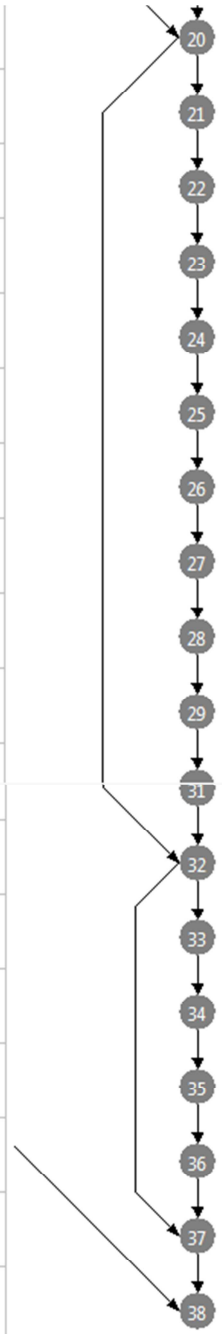
387		21	r(usurol)	usurol.setDisciplina(this.disciplinaDao.getDisciplina(Integer.parseInt(re
388		22	r(usurol)	this.usuariosrolDao.guardarUsuariosRol(usurol);
390		23		response.sendRedirect("AdmUsuarios.html");
392		24, 30		if((null != request.getParameter("estado"))&(
393		25	d(usurol)	usurol=this.usuariosrolDao.getUsuariosRol(request.getParameter("ci"),
394		26	d(estado)	int estado =Integer.parseInt(request.getParameter("estado"));
395		27, 28	r(estado)	if(estado==1){
396		29	d(estado)	estado=0;
398		29	d(estado)	estado=1;
401		30	r(usurol)	this.usuariosrolDao.modificarUsuariosRol(usurol);
403		31	d(usuarios)	Usuarios usuarios = this.usuariosDao.getUsuarios(Integer.parseInt(request.getParameter("ci"
404		32	r(map), r(usuarios)	map.put("listausurol",usuarios.getUsuariosRols());
405		33	r(map), r(usuarios)	map.put("idusuario", usuarios.getIdusuario());
406		34	r(map)	map.put("ci", request.getParameter("ci"));
407		35	r(map)	map.put("listaroles", this.rolDao.getAllRol());
408		36	r(map)	map.put("disciplinas", this.disciplinaDao.getAllDisciplina());
410		37	r(map)	return new ModelAndView("AdmUsuRol",map);
411			u(estado), u(xci), u(usurol), u(session), u(map), u(usuarios), u(usurolId), u(rol), u(usuarios)	}

CLASE: UsuariosControlador.java

METODO: abmusurol



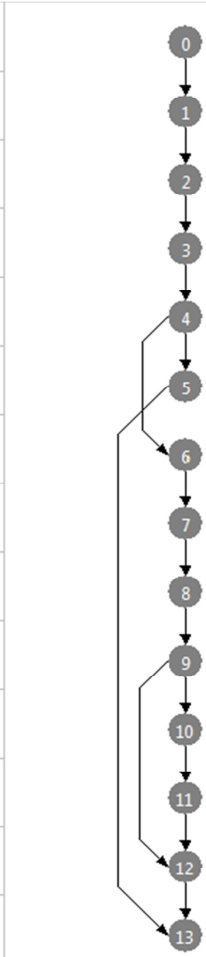
413	1	u(usuarios), u(xci), u(usurol), u(session), u(map), u(usurolId), u(rol), u(usuarios)	public ModelAndView abmusurol(HttpServletRequest request,HttpServletResponse
413	2		public ModelAndView abmusurol(HttpServletRequest request,HttpServletResponse
414	3	d(session)	HttpSession session=request.getSession(true);
415	4	d(xci), r(session)	String xci= (String) session.getAttribute("ci");
416	5, 6	r(xci)	if(xci==null)
419	38		return new ModelAndView("Salir");
421	7	d(map)	Map map = new HashMap();
422	8		UsuariosRol usurol;
423	9, 13		if(null != request.getParameter("ci")& null!=request.getParameter("codrol
424	10	d(usurol)	usurol=this.usuariosrolDao.getUsuariosRol(request.getParameter("ci"), Integer.parseInt(request.getParamet
425	11	r(usurol), r(map)	map.put("UsuariosRol", usurol);
426	12	r(map)	map.put("ci", request.getParameter("ci"));
427	20	r(map)	map.put("codrol", request.getParameter("codrol"));
429	14	d(usurol)	usurol=new UsuariosRol();
430	15	r(usurol), r(map)	map.put("UsuariosRol", usurol);
431	16	r(map)	map.put("ci", request.getParameter("ci"));
432	17	d(usuarios)	Usuarios usuarios=this.usuariosDao.getUsuarios(Integer.parseInt(request.getPara
433	18	r(usuarios), r(map)	map.put("ListaUsuRol", usuarios.getUsuariosRols());
434	19	r(map)	map.put("ListaRol",this.rolDao.getAllRol());
435	20	r(map)	map.put("codrol",0);



437	21, 32		if(null != request.getParameter("save")){
438	22		System.out.println("[Guardando]");
439	23	d(usurol)	usurol=new UsuariosRol();
440	24	d(usurolId)	UsuariosRolId usurolId=new UsuariosRolId();
441	25	r(usurolId)	usurolId.setCodrol(Integer.parseInt(request.getParameter("codrol")));
442	26	d(usuarios)	Usuarios usuarios=this.usuariosDao.getUsuar
443	27	d(rol)	Rol rol=this.rolDao.getRol(Integer.parse
444	28	r(usurol), r(usurolId)	usurol.setId(usurolId);
445	29	r(usurol), r(rol)	usurol.setRol(rol);
446	30	r(usurol), r(usuarios)	usurol.setUsuarios(usuarios);
447	31	r(usurol)	this.usuariosrolDao.guardarUsuarios
			.html?ci= "+request.getParameter(
450	33, 37		if(null != request.getParameter("update")){
451	34		System.out.println("[Actualizando datos]");
452	35	d(usurol)	usurol=this.usuariosrolDao.getUsua riosRol(request.getParameter("ci"),
454	36	r(usurol)	this.usuariosrolDao.modificarUsuari osRol(usurol);
455	37		response.sendRedirect("ListaUsuRol .html?ci="+request.getParameter("
458	38	r(map)	return new ModelAndView("abmUsuRol",map);
459		u(usuarios), u(xci), u(usurol), u(session), u(map), u(usurolId),	}

CLASE: UsuariosControlador.java

METODO: veralumnos

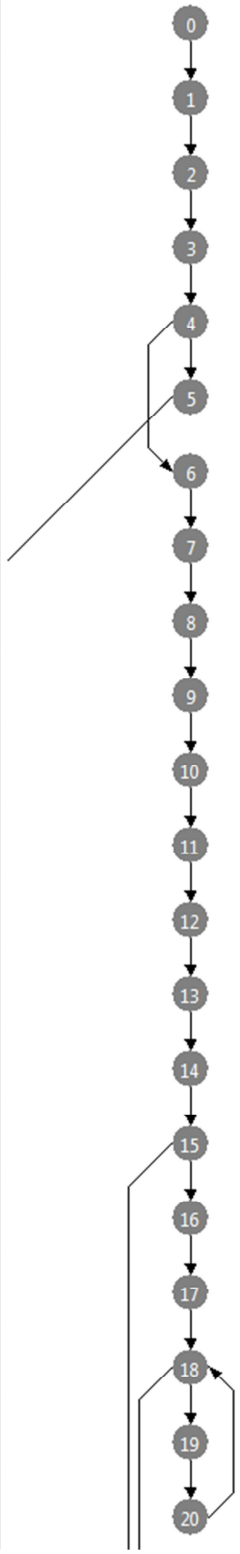
460		1	u(xci), u(session), u(map), u(usuarios)	public ModelAndView veralumnos(HttpServletRequest
460		2		public ModelAndView veralumnos(HttpServletRequest
461		3	d(session)	HttpSession session=request.getSession(true);
462		4	d(xci), r(session)	String xci= (String) session.getAttribute("ci");
463		5, 6	r(xci)	if(xci==null)
465		13		return new ModelAndView("Salir");
467		7	d(map)	Map map = new HashMap();
469		8	r(xci), d(usuarios)	Usuarios usuarios=this.usuariosDao.getUsuar
470		9	r(map), r(usuarios)	map.put("Equipos", usuarios.getEquipos());
472		10, 12		if(null != request.getParameter("ver")){
473		11		System.out.println("[ver alumnos de]");
474		12	r(map)	map.put("Alumnos", this.equipoDao.getEquipo(Integer.p
477		13	r(map)	return new ModelAndView("veralumnos",map) ;
478			u(xci), u(session), u(map), u(usuarios)	}

CLASE: UsuariosControlador.java

METODO: verhorarios

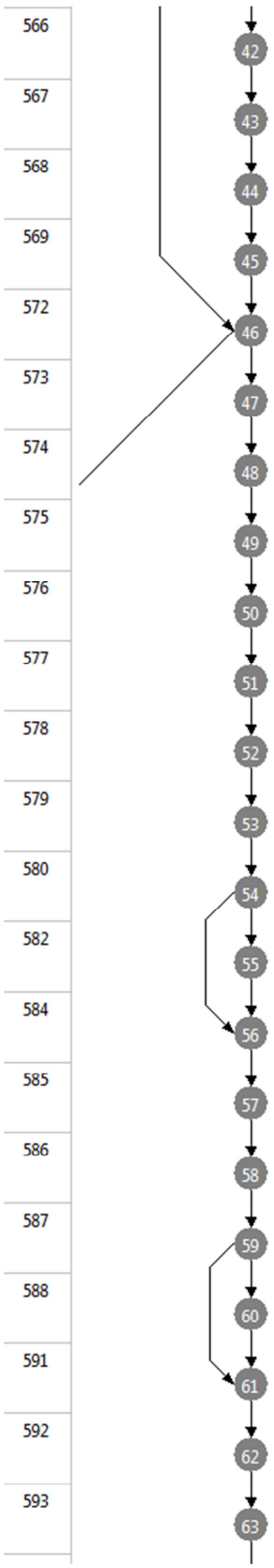
479		1	u(xci), u(session), u(map), u(usuarios)	public ModelAndView verhorarios(HttpServletRequest
479		2		public ModelAndView verhorarios(HttpServletRequest
480		3	d(session)	HttpSession session=request.getSession(true);
481		4	d(xci), r(session)	String xci= (String) session.getAttribute("ci");
482		5, 6	r(xci)	if(xci==null)
484		10		return new ModelAndView("Salir");
486		7	d(map)	Map map = new HashMap();
488		8	r(xci), d(usuarios)	Usuarios usuarios=this.usuariosDao.getUsuar
489		9	r(map), r(usuarios)	map.put("Equipos", usuarios.getEquipos());
491		10	r(map)	return new ModelAndView("verhorarios",map);
492			u(xci), u(session), u(map), u(usuarios)	}

METODO: Reporte

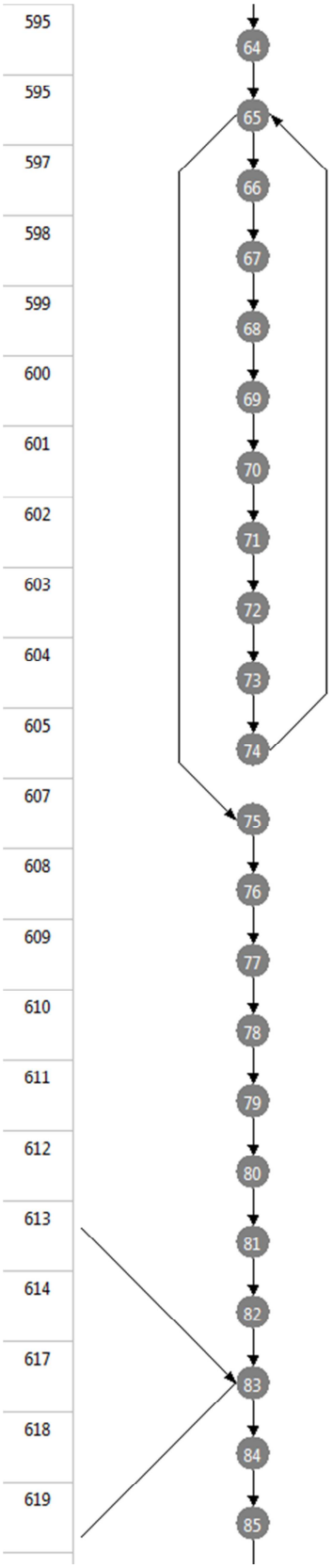


502		1	u(obj), u(alumno), u(listacampeonatos), u(equipo),	public ModelAndView reportes(HttpServletRequest
502		2		public ModelAndView reportes(HttpServletRequest
504		3	d(session)	HttpSession session=request.getSession(true);
505		4	d(xci), r(session)	String xci=(String) session.getAttribute("ci");
506		5, 6	r(xci)	if(xci==null)
509		122		return new ModelAndView("Salir");
511		7	d(map)	Map map = new HashMap();
512		8	r(xci), d(usuarios)	Usuarios usuarios=this.usuariosDao.getUsuar
514		9	d(ssesion)	Registroacciones ssesion=new Registroacciones();
515		10	r(ssesion)	ssesion.setFecha(new Date());
516		11	r(ssesion)	ssesion.setHora(new Date());
517		12	r(ssesion), r(usuarios)	ssesion.setUsuarios(usuarios);
520		13	d(outFile)	File outFile = new File(request.getParameter("reporte"
522		14	d(writer), r(outFile)	BufferedWriter writer = new BufferedWriter(new
524		15	r(writer)	writer.write("<?xml version=\"1.0\" encoding=\"UTF-8\"?><Deporte>")
525		16, 21		if(request.getParameter("reporte").toString().equals("Campeonatos")){
526		17	d(listacampeonatos)	List listacampeonatos = this.campeonatoDao.getAllCampeo
527		18	r(listacampeonatos), d(li)	for (ListIterator li = listacampeonatos.listIterator();
527		19, 21	r(li)	for (ListIterator li = listacampeonatos.listIterator();
528		20	d(obj), r(li)	Campeonato obj = (Campeonato) li.next();
529		18	r(obj), r(obj), r(obj), r(obj), r(obj), r(obj), r(writer)	writer.write("<Campeonato>" +

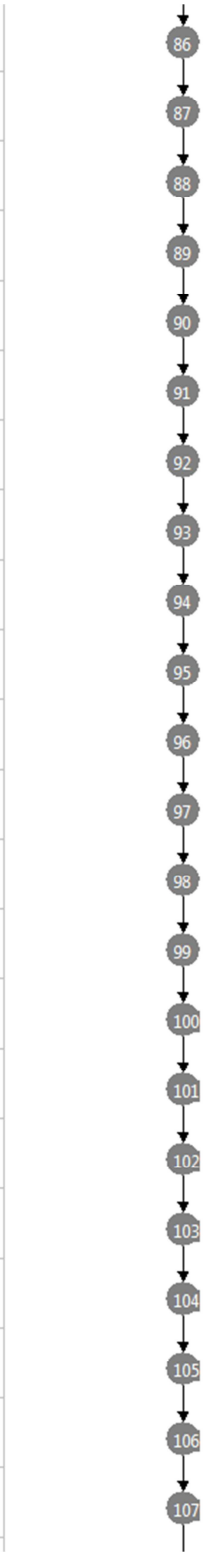
540	22, 32		if(request.getParameter("reporte").toString().equals("ReporteDisciplina"
541	23	r(session)	session.setAttribute("coddis", request.getParameter("coddis"));
542	24	r(session)	session.setAttribute("disciplina", this.disciplinaDao.getDisciplina(Inte
543	25	d(listacategorias)	List listacategorias = this.categoriaDao.getAllCategoria();
544	26	r(listacategorias)	System.out.println("Tamaño "+listacategorias.size());
545	27	r(listacategorias)	for (ListIterator li = listacategorias.listIterator();
545	28, 30		for (ListIterator li = listacategorias.listIterator();
546	29	d(obj)	Categoria obj = (Categoria) li.next();
547	27	r(writer), r(obj), r(obj)	writer.write("<Categoria>" +
552	31	r(writer)	writer.write("<Equipo></Equipo>");
553	32	r(writer)	writer.write("<Alumno></Alumno>");
556	33, 46		if(request.getParameter("reporte").toString().equals("Usuario")){
557	34	d(usuario)	Usuarios usuario=this.usuariosDao.getUsuari
558	35		request.getSession().setAttribute("Usuario", "usuario");
559	36	r(usuario)	request.getSession().setAttribute("ci", usuario.getCi().toString());
560	37	r(usuario)	request.getSession().setAttribute("nombre",
561	38	r(usuario), r(usuario)	request.getSession().setAttribute("apellidos", usuario.getAp()+"
562	39	r(usuario)	request.getSession().setAttribute("fechadenacimiento",
563	40	r(usuario)	request.getSession().setAttribute("email", usuario.getEmail().toString());
564	41		request.getSession().setAttribute("roles", "");
565	42	r(usuario)	request.getSession().setAttribute("estado",



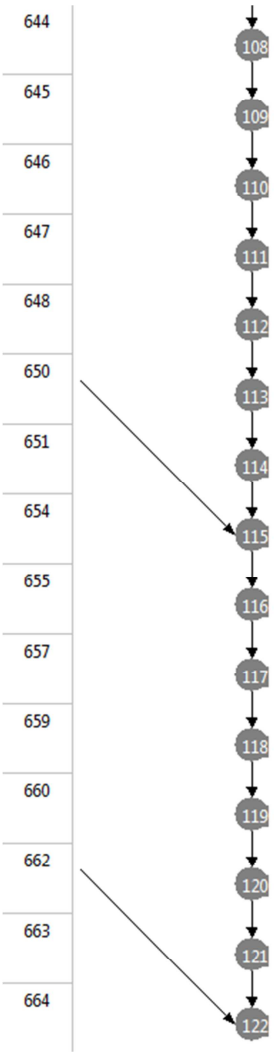
566	43	r(usuario)	request.getSession().setAttribute("foto", usuario.getFoto());
567	44	r(usuario), r(usuario), r(usuario)	request.getSession().setAttribute("direccion",
568	45	r(writer)	writer.write("<Equipo></Equipo>");
569	46	r(writer)	writer.write("<Alumno></Alumno>");
572	47, 83		if(request.getParameter("reporte").toString().equals("Equipo")){
573	48	d(equipo)	Equipo equipo=this.equipoDao.getEquipo(I
574	49	d(alumnos)	List alumnos=this.alumnosDao.getAlu
575	50	r(alumnos)	System.out.println("tamaño de alumnos "+alumnos.size());
576	51		request.getSession().setAttribute("Usuario", "usuario");
577	52	r(equipo)	request.getSession().setAttribute("code", equipo.getCode());
578	53	r(equipo)	request.getSession().setAttribute("nombre", equipo.getNombre());
579	54	d(estado)	int estado=0;
580	55, 56	r(equipo)	if(equipo.getEstado()==1)
582	56	d(estado)	estado=1;
584	57	r(estado)	request.getSession().setAttribute("estado", Integer.toString(estado));
585	58	r(equipo)	request.getSession().setAttribute("codcat",
586	59	d(horario)	String horario="Invitado";
587	60, 61	r(equipo)	if(equipo.getInvitado()==0){
588	61	r(equipo), r(equipo), d(horario)	horario=equipo.getHorario().getHoraini()+" -
591	62	r(horario)	request.getSession().setAttribute("codhor", horario);
592	63	r(equipo)	request.getSession().setAttribute("coddis",
593	64	r(equipo), r(equipo)	request.getSession().setAttribute("idusuario",



65	r(alumnos)	for (ListIterator li = alumnos.listIterator(); li.hasNext();)
66, 75		for (ListIterator li = alumnos.listIterator(); li.hasNext();)
67	r(writer)	writer.write("<Alumno>");
68	d(obj)	Alumnos obj=(Alumnos) li.next();
69	r(obj), r(writer)	writer.write("<id>"+obj.getCodalum()+"</id>");
70	r(obj), r(writer)	writer.write("<nombre>"+obj.getNombres()+"</nombre>");
71	r(obj), r(obj), r(writer)	writer.write("<apellidos>"+obj.getAp()+"
72	r(obj), r(writer)	writer.write("<edad>"+obj.getEdad()+"</edad>");
73	r(obj), r(writer)	writer.write("<grado>"+obj.getCurso()+"</grado>");
74	r(obj), r(obj), r(obj), r(writer)	writer.write("<direccion>"+obj.getDireccion().getBarrio()+"
65	r(writer)	writer.write("</Alumno>");
76	r(writer)	writer.write("<Alumno>");
77	r(writer)	writer.write("<id> </id>");
78	r(writer)	writer.write("<nombre> </nombre>");
79	r(writer)	writer.write("<apellidos> </apellidos>");
80	r(writer)	writer.write("<edad> </edad>");
81	r(writer)	writer.write("<grado> </grado>");
82	r(writer)	writer.write("<direccion> </direccion>");
83	r(writer)	writer.write("</Alumno>");
84, 115		if(request.getParameter("reporte").toString().equals("Alumno")){
85	d(alumno)	Alumnos alumno=this.alumnosDao.getAlumnos(Integer.parseInt(request.getPar
86		request.getSession().setAttribute("Usuario", "Usuario");



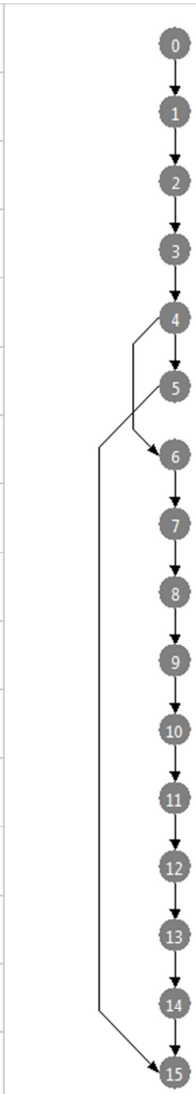
621	87	r(alumno)	request.getSession().setAttribute("nombre", alumno.getNombres().toString());
622	88	r(alumno), r(alumno)	request.getSession().setAttribute("apellidos", alumno.getAp().toString()+"
623	89	r(alumno)	request.getSession().setAttribute("fechadenacimiento", alumno.getFechanac().toString());
625	90	r(alumno)	request.getSession().setAttribute("celular", alumno.getCelular().toString());
626	91	r(alumno)	request.getSession().setAttribute("estado", alumno.getEstado().toString());
627	92	r(alumno)	request.getSession().setAttribute("fechainicio", alumno.getFechaini().toString());
628	93	r(alumno)	request.getSession().setAttribute("edad", alumno.getEdad().toString());
629	94	r(alumno)	request.getSession().setAttribute("peso", alumno.getPeso().toString());
630	95	r(alumno)	request.getSession().setAttribute("estatura", alumno.getEstatura().toString());
631	96	r(alumno)	request.getSession().setAttribute("telefono", Integer.toString(alumno.getTelefon
632	97	r(alumno)	request.getSession().setAttribute("colegio", alumno.getColegio().toString());
633	98	r(alumno)	request.getSession().setAttribute("curso", alumno.getCurso().toString());
634	99	r(alumno), r(alumno), r(alumno)	request.getSession().setAttribute("direccion", alumno.getDireccion().getBarrio().t
635	100	r(alumno)	request.getSession().setAttribute("provincia", alumno.getProvincia().getNombre(
636	101	r(alumno)	request.getSession().setAttribute("sexo", alumno.getSexo().toString());
637	102	r(alumno)	request.getSession().setAttribute("code", alumno.getEquipo().getNombre().t
638	103	r(alumno)	request.getSession().setAttribute("coddis", alumno.getDisciplina().getNombre
639	104	r(alumno)	request.getSession().setAttribute("codcat", alumno.getCategoria().getNombre(
640	105	r(alumno)	request.getSession().setAttribute("nombrepa", alumno.getNombrepa().toString());
641	106	r(alumno)	request.getSession().setAttribute("nombrema", alumno.getNombrema().toString());
642	107	r(alumno)	request.getSession().setAttribute("nombreapo", alumno.getNombreapo().toString()
643	108	r(alumno)	request.getSession().setAttribute("profesionpa", alumno.getProfesionpa().toString()



109	r(alumno)	request.getSession().setAttribute("profesionma",
110	r(alumno)	request.getSession().setAttribute("profesionapo",
111	r(alumno)	request.getSession().setAttribute("teltrabajoma",
112	r(alumno)	request.getSession().setAttribute("teltrabajopa",
113	r(alumno)	request.getSession().setAttribute("teltrabajoapo",
114	r(writer)	writer.write("<Equipo></Equipo>");
115	r(writer)	writer.write("<Alumno></Alumno>");
116	r(writer)	writer.write("</Deporte>");
117	r(writer)	writer.newLine();
118	r(writer)	writer.close();
119		System.err.println(e);
120		System.exit(1);
121	r(map)	map.put("reporte",request.getParameter("reporte").toString());
122	r(map)	return new ModelAndView("Reportes",map);
	u(obj), u(alumno), u(listacampeonatos), u(equipo),	}

CLASE: CampeonatoControlador.java

METODO: admcampeonato

132		1	u(xci), u(ssesion), u(session), u(map)	public ModelAndView admcampeonato(HttpServletReque
132		2		public ModelAndView admcampeonato(HttpServletReque
133		3	d(session)	HttpSession session=request.getSession(true);
134		4	d(xci), r(session)	String xci= (String) session.getAttribute("ci");
135		5, 6	r(xci)	if(xci==null)
138		15		return new ModelAndView("Salir");
142		7	d(ssesion)	Registroacciones ssesion=new Registroacciones();
143		8	r(ssesion)	ssesion.setFecha(new Date());
144		9	r(ssesion)	ssesion.setHora(new Date());
145		10	r(xci), r(ssesion)	ssesion.setUsuarios(this.usuariosDao.getUsuarios(Integer.parseInt(xci))) ;
146		11	r(ssesion)	ssesion.setAccion("lista de series ");
147		12	r(ssesion)	this.registroaccionesDao.guardarRegistroacciones(ssesion);//
149		13	d(map)	Map map = new HashMap();
151		14	r(map)	map.put("datos",this.campeonatoDao.getAllCampeonato());
153		15	r(map)	return new ModelAndView("AdmCampeonato"
154			u(xci), u(ssesion), u(session), u(map)	}

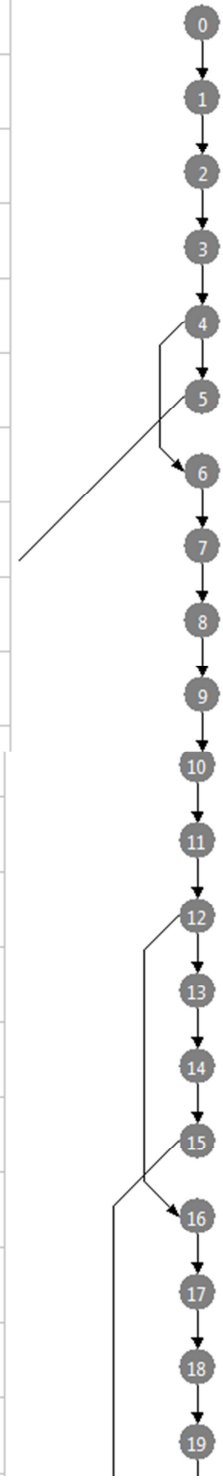
CLASE: CampeonatoControlador.java

METODO: reporteCampeonato

156		1	u(xci), u(ssesion), u(session), u(map)	public ModelAndView reporteCampeonato(HttpServletRequest request,HttpServletResponse
156		2		public ModelAndView reporteCampeonato(HttpServletRequest request,HttpServletResponse
157		3	d(session)	HttpSession session=request.getSession(true);
158		4	d(xci), r(session)	String xci= (String) session.getAttribute("ci");
159		5, 6	r(xci)	if(xci==null)
162		13		return new ModelAndView("Salir");
165		7	d(ssesion)	Registroacciones ssesion=new Registroacciones(); //
166		8	r(ssesion)	ssesion.setFecha(new Date());
167		9	r(ssesion)	ssesion.setHora(new Date());
168		10	r(xci), r(ssesion)	ssesion.setUsuarios(this.usuariosDao.getUsuarios(Integer.parseInt(xci))) . //
171		11	d(map)	Map map = new HashMap();
172		12	r(map)	map.put("datos",this.campeonatoDao.getAllCampeonato());
173		13	r(map)	return new ModelAndView("reporteCampeonato",map);
174			u(xci), u(ssesion), u(session), u(map)	}

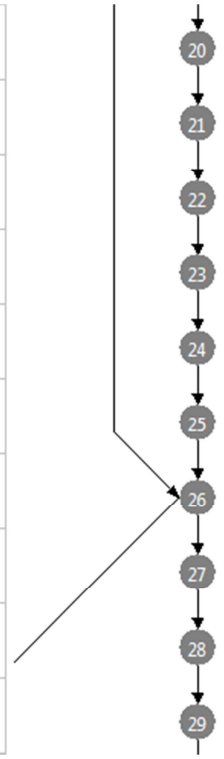
CLASE: CampeonatoControlador.java

METODO: abmcampeonato

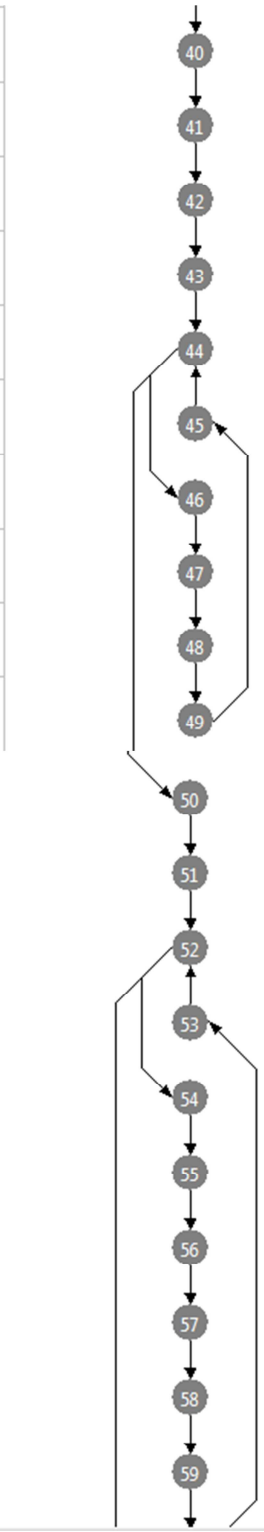


176	1	u(fechaini), u(fecha), u(fechafin), u(dis), u(codserie), u(campeonatos), u(codbarrio)	public ModelAndView abmcampeonato(HttpServletRequest request,HttpServletResponse
176	2		public ModelAndView abmcampeonato(HttpServletRequest request,HttpServletResponse
177	3	d(session)	HttpSession session=request.getSession(true);
178	4	d(xci), r(session)	String xci=(String) session.getAttribute("ci");
179	5, 6	r(xci)	if(xci==null)
182	74		return new ModelAndView("Salir");
184	7	d(ssesion)	Registroacciones ssesion=new Registroacciones(); //
185	8	r(ssesion)	ssesion.setFecha(new Date());
186	9	r(ssesion)	ssesion.setHora(new Date());
187	10	r(xci), r(ssesion)	ssesion.setUsuarios(this.usuariosDao.getUsuarios(Integer.parseInt(xci))) . //
189	11	d(map)	Map map = new HashMap();
190	12		Campeonato campeonato;
192	13, 16		if(null != request.getParameter("codcam")){
193	14	d(campeonato)	campeonato=this.campeonatoDao.getCampeonato(Integer.parseInt(request.getParameter("codcam")));
194	15	r(map), r(campeonato)	map.put("Campeonato", campeonato);
195	26	r(map)	map.put("listadisc", this.disciplinaDao.getAllDisciplina()) .
197	17	d(campeonato)	campeonato=new Campeonato();
198	18	r(campeonato)	campeonato.setNombre("");
199	19	d(fecha)	Date fecha=new Date();
201	20	r(fecha), r(campeonato)	campeonato.setFechafin(fecha);

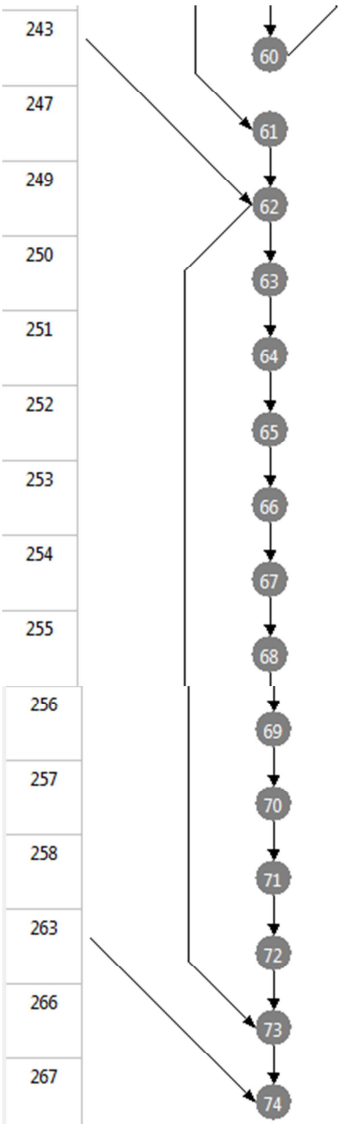
202	
203	
204	
205	
206	
207	
210	
211	
212	
213	



21	r(fecha), r(campeonato)	campeonato.setFechaini(fecha);
22	r(map)	map.put("barrios", this.barrioDao.getAllBarrio());
23	r(map)	map.put("listadisc", this.disciplinaDao.getAllDisciplina()) ;
24	r(map)	map.put("listacategoria", this.categoriaDao.getAllCategoria()) ;
25	r(map)	map.put("series", this.serieDao.getAllSerie());
26	r(map), r(campeonato)	map.put("Campeonato", campeonato);
27, 62		if(null != request.getParameter("save")){
28		System.out.println("[Guardando]");
29	d(campeonato)	campeonato=new Campeonato();
30	r(campeonato)	campeonato.setNombre(request.getParameter("nombre"));



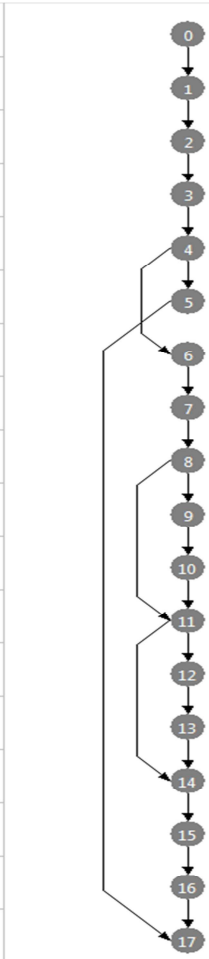
224	40	41	r(campeonato)	this.campeonatoDao.guardarCampeonato(campeonato);
226	41	42	d(codbarrio)	String [] codbarrio=request.getParameterValuer("codbarrio");
227	42	43	d(campeonatos)	Set campeonatos=new HashSet();
228	43	44	d(i)	for (int i = 0; i < codbarrio.length; i++) {
228	44	46, 50	r(codbarrio), r(i)	for (int i = 0; i < codbarrio.length; i++) {
228	45	44	d(i)	for (int i = 0; i < codbarrio.length; i++) {
229	46	47	r(codbarrio), d(barrio), r(i)	Barrio barrio=this.barrioDao.getBarrio(Integer.parseInt(codbarrio[i]));
230	47	48	r(campeonatos), r(campeonato)	campeonatos.add(campeonato);
231	48	49	r(campeonatos), r(barrio)	barrio.setCampeonatos(campeonatos);
232	49	45	r(barrio)	this.barrioDao.modificarBarrio(barrio);
234	50	51	d(codserie)	String [] codserie=request.getParameterValuer("codserie");
235	51	52		for (int i = 0; i < codserie.length; i++) {
235	52	53	r(codserie)	for (int i = 0; i < codserie.length; i++) {
235	53	52		for (int i = 0; i < codserie.length; i++) {
236	54	55	d(campeonatoSerie)	CampeonatoSerie campeonatoSerie=new CampeonatoSerie();
237	55	56	r(campeonatoSerie), r(campeonato)	campeonatoSerie.setCampeonato(campeonato);
238	56	57	r(codserie), r(campeonatoSerie)	campeonatoSerie.setSerie(this.serieDao.getSerie(Integer.parseInt(codserie[i])));
239	57	58	r(campeonatoSerie)	campeonatoSerie.setEstado(0);
240	58	59	r(campeonatoSerie)	this.campeonatoSerieDao.guardarCampeonatoSerie(campeonatoSerie);
242	59	60	r(ssesion), r(campeonato)	ssesion.setAccion("creado campeonato con id= "+campeonato.getCodcam()); //
243		53	r(ssesion)	this.registroaccionesDao.guardarRe



243	53	r(ssesion)	this.registroaccionesDao.guardarRegistroacciones(ssesion);
247	62	r(campeonato)	response.sendRedirect("AdmCampeonatoSerie.html?codcam="+campeonato.getCodcam());
249	63, 73		if(null != request.getParameter("update")){
250	64		System.out.println("[Actualizando datos]");
251	65	r(campeonato)	campeonato.setNombre(request.getParameter("nombre"));
252	66	d(fecha)	SimpleDateFormat fecha = new SimpleDateFormat("dd-MM-yyyy");
253	67	r(fecha)	fecha.setLenient(true);
254	68	d(fechaini), r(fecha)	Date fechaini = fecha.parse(request.getParameter("fechaini"));
255	69	r(fecha), d(fechafin)	Date fechafin = fecha.parse(request.getParameter("fechafin"));
256	70	r(fechaini), r(campeonato)	campeonato.setFechaini(fechaini);
257	71	r(fechafin), r(campeonato)	campeonato.setFechafin(fechafin);
258	72	r(campeonato)	this.campeonatoDao.modificarCampeonato(campeonato);
263	73		response.sendRedirect("mensaje.html");
266	74	r(map)	return new ModelAndView("abmCampeonato",map);
267		u(fechaini), u(fecha), u(fechafin), u(dis), u(codserie), u(campeonatos), u(codbarrio)	}

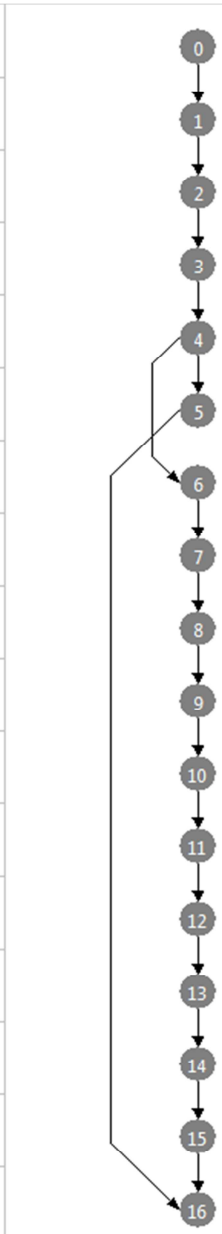
CLASE: CampeonatoControlador.java

METODO: campeonatobarrio

269		1	u(xci), u(session), u(map), u(campeonato)	public ModelAndView campeonatobarrio(HttpServletRequ
269		2		public ModelAndView campeonatobarrio(HttpServletRequ
270		3	d(session)	HttpSession session=request.getSession(true);
271		4	d(xci), r(session)	String xci=(String) session.getAttribute("ci");
272		5, 6	r(xci)	if(xci==null)
275		17		return new ModelAndView("Salir");
278		7	d(map)	Map map = new HashMap();
279		8		Campeonato campeonato;
282		9, 11		if(null != request.getParameter("codcam")){
283		10	d(campeonato)	campeonato=this.campeonatoDao.getCampeonato(Integer.parseInt(re
285		11	r(map), r(campeonato)	map.put("campeonato",campeonato);
287		12, 14		if(null != request.getParameter("guardar")){
288		13		System.out.println("[Guardando]");
290		14		response.sendRedirect("mensaje.html");
294		15		System.out.println("MENSAJE ERRORS rolpro"+e);
296		16	r(map)	map.put("barrios", this.barrioDao.getAllBarrio());
297		17	r(map)	return new ModelAndView("RolPro",map);
298			u(xci), u(session), u(map), u(campeonato)	}

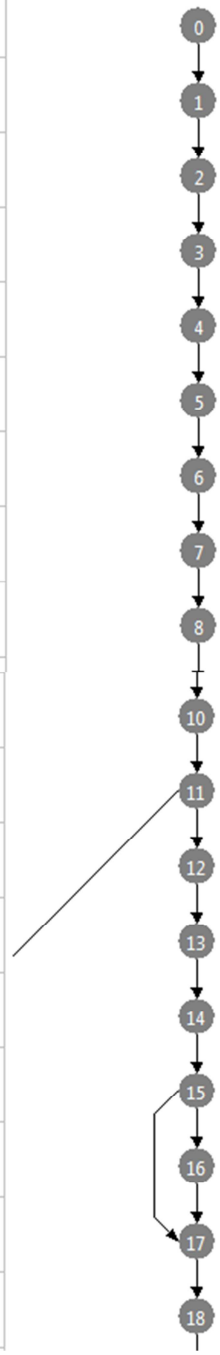
CLASE: RolpartidoControlador.java

METODO: admrolpartido

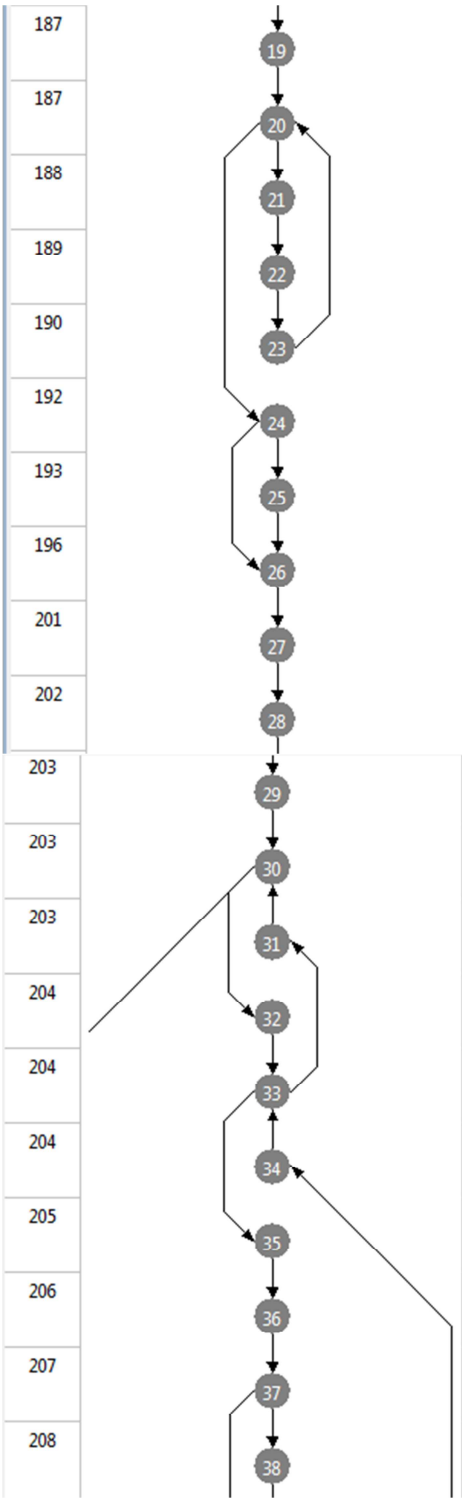
134		1	u(xci), u(ssesion), u(session), u(map), u(listarolpartido)	public ModelAndView admrolpartido(HttpServletRequest request,HttpServletResponse
134		2		public ModelAndView admrolpartido(HttpServletRequest request,HttpServletResponse
135		3	d(session)	HttpSession session=request.getSession(true);
137		4	d(xci), r(session)	String xci=(String) session.getAttribute("ci");
138		5, 6	r(xci)	if(xci==null)
141		16		return new ModelAndView("Salir");
145		7	d(ssesion)	Registroacciones ssesion=new Registroacciones();
146		8	r(ssesion)	ssesion.setFecha(new Date());
147		9	r(ssesion)	ssesion.setHora(new Date());
148		10	r(xci), r(ssesion)	ssesion.setUsuarios(this.usuariosDao.getUsuarios(Integer.parseInt(xci)));
149		11	r(ssesion)	ssesion.setAccion("lista de rolde partidos ");
150		12	r(ssesion)	this.registroaccionesDao.guardarRegistroacciones(ssesion);//
152		13	d(map)	Map map = new HashMap();
154		14	d(listarolpartido)	List listarolpartido = this.rolpartidoDao.getAllRolpartido();
155		15	r(map), r(listarolpartido)	map.put("listarolpartido",listarolpartido);
158		16	r(map)	return new ModelAndView("AdmRolpartido",map);
159			u(xci), u(ssesion), u(session), u(map), u(listarolpartido)	}

CLASE: RolpartidoControlador.java

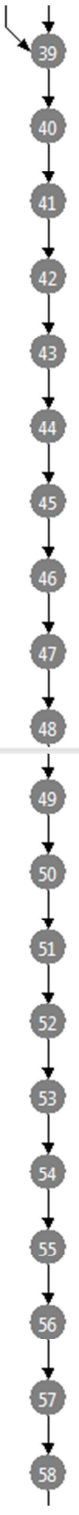
METODO: generarrolpartido



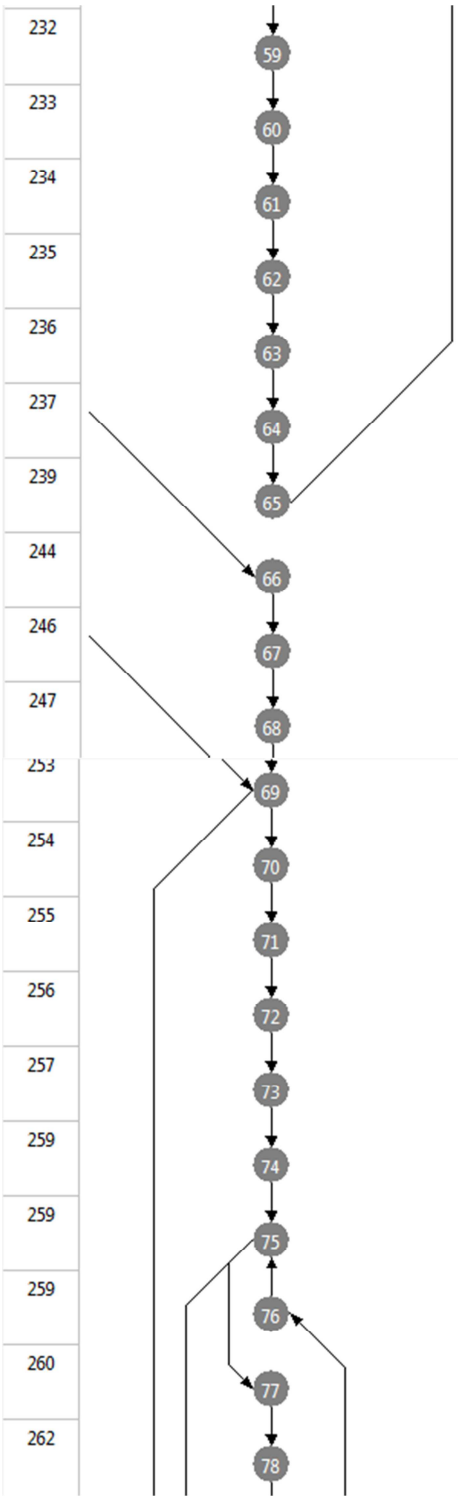
160	1	u(fechas), u(fnac), u(rolpartido), u(pr), u(teams),	public ModelAndView generarrolpartido(HttpServletReque
160	2		public ModelAndView generarrolpartido(HttpServletReque
161	3	d(session)	HttpSession session=request.getSession(true);
162	4	d(xci), r(session)	String xci=(String) session.getAttribute("ci");
163	5	r(xci)	if(xci==null)
169	6	d(ssesion)	Registroacciones ssesion=new Registroacciones(); //
170	7	r(ssesion)	ssesion.setFecha(new Date());
171	8	r(ssesion)	ssesion.setHora(new Date());
172	9	r(xci), r(ssesion)	ssesion.setUsuarios(this.usuariosDao.getUsuarios(Integer.parseInt(xci))) . //
176	11	d(idcamserie)	int idcamserie=Integer.parseInt(request
178	12, 69		if(request.getParameter("generar")!=null &
179	13	d(cequipos), r(idcamserie)	Set cequipos=this.campeonatoSerieDa
180	14	r(cequipos)	System.out.println("generar rol de partidos tamano "+cequipos.size());
181	15	r(cequipos), d(tamano)	int tamano=cequipos.size();
182	16, 17	r(cequipos)	if(cequipos.size() % 2 == 1){
183	17	r(cequipos), d(tamano)	tamano=cequipos.size()+1;
185	18	d(equipos), r(tamano)	int[] equipos=new int[tamano];
186	19	d(io)	int io=0;



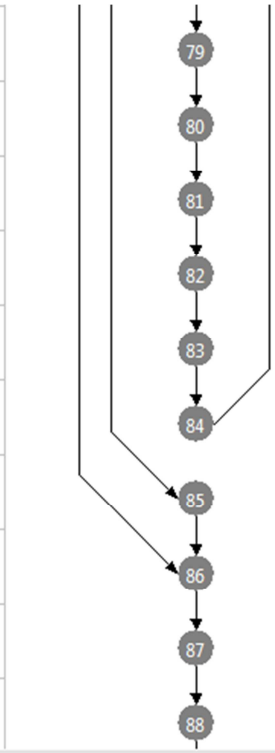
187	20	r(cequipos), d(iterator)	for (Iterator iterator = cequipos.iterator(); iterator.hasNext()) {
187	21, 24	r(iterator)	for (Iterator iterator = cequipos.iterator(); iterator.hasNext()) {
188	22	d(pr), r(iterator)	CampeonatoSerieEquipo pr = (CampeonatoSerieEquipo)iterator.next();
189	23	r(pr), d(equipos), r(io)	equipos[io]=pr.getEquipo().getCode();
190	20	d(io)	io++;
192	25, 26	r(cequipos)	if(cequipos.size() % 2 == 1){
193	26	d(equipos), r(io)	equipos[io]=0;
196	27	d(teams), r(equipos)	int teams = equipos.length;
201	28	r(teams), d(totalRounds)	int totalRounds = teams - 1;
202	29	r(teams), d(matchesPerRound)	int matchesPerRound = teams / 2;
203	30	d(round)	for (int round = 0; round < totalRounds; round++) {
203	32, 66	r(totalRounds), r(round)	for (int round = 0; round < totalRounds; round++) {
203	30	d(round)	for (int round = 0; round < totalRounds; round++) {
204	33	d(match)	for (int match = 0; match < matchesPerRound; match++) {
204	35, 31	r(matchesPerRound), r(match)	for (int match = 0; match < matchesPerRound; match++) {
204	33	d(match)	for (int match = 0; match < matchesPerRound; match++) {
205	36	r(teams), r(match), d(home), r(round)	int home = (round + match) % (teams - 1);
206	37	r(teams), r(teams), r(match), d(away), r(round)	int away = (teams - 1 - match + round) % (teams - 1);
207	38, 39	r(match)	if (match == 0) {
208	39	r(teams), d(away)	away = teams - 1;



210	39	40	d(rolpartido)	Rolpartido rolpartido=new Rolpartido();
211	40	41	r(rolpartido), r(round)	rolpartido.setRonda((round+1));
212	41	42	r(match), r(rolpartido)	rolpartido.setOrden(match+1);
213	42	43	r(rolpartido)	rolpartido.setFecha(new Date());
214	43	44	r(rolpartido)	rolpartido.setHora("12:00am");
215	44	45	r(rolpartido)	rolpartido.setObservacion("");
216	45	46	r(rolpartido)	rolpartido.setCampeonatoSerie(this.campeonatoSerieDao.getCampeonatoSerie(Integer.parseInt(request.ge
217	46	47	r(rolpartido)	this.rolpartidoDao.guardarRolpartido(rolpartido);
219	47	48	d(equipoRolpartido)	EquipoRolpartido equipoRolpartido=new EquipoRolpartido();
220	48	49	r(equipoRolpartido), r(rolpartido)	equipoRolpartido.setRolpartido(rolpartido);
221	49	50	r(equipoRolpartido)	equipoRolpartido.setResultado(0);
222	50	51	r(equipos), r(home), r(equipoRolpartido)	equipoRolpartido.setEquipo(this.equipoDao.getEquipo(equipos[(home
223	51	52	d(id)	EquipoRolpartidoId id=new EquipoRolpartidoId();
224	52	53	r(equipos), r(id), r(home)	id.setCode(equipos[(home)]);
225	53	54	r(id), r(rolpartido)	id.setCodrolp(rolpartido.getCodrolp());
226	54	55	r(id), r(equipoRolpartido)	equipoRolpartido.setId(id);
227	55	56	r(equipoRolpartido)	this.equiporolpartidoDao.guardarEquipoRolpartido(equipoRolpartido);
229	56	57	d(equipoRolpartido)	equipoRolpartido=new EquipoRolpartido();
230	57	58	r(equipoRolpartido), r(rolpartido)	equipoRolpartido.setRolpartido(rolpartido);
231	58		r(equipoRolpartido)	equipoRolpartido.setResultado(0);

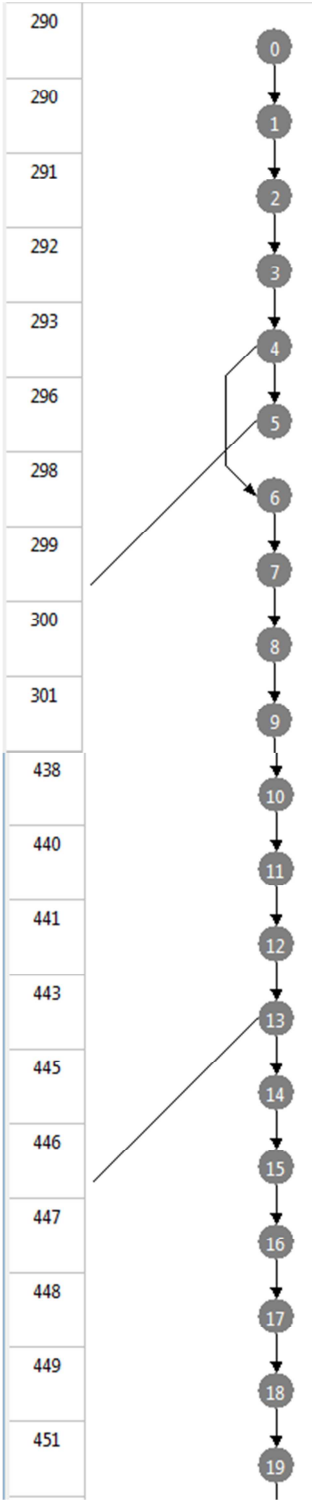


232	60	r(equipos), r(away), r(equipoRolpartido)	equipoRolpartido.setEquipo(this.equipoDao.getEquipo(equipos[(away)]));
233	61	d(id)	id=new EquipoRolpartidoId();
234	62	r(equipos), r(id), r(away)	id.setCode(equipos[(away)]);
235	63	r(id), r(rolpartido)	id.setCodrolp(rolpartido.getCodrolp());
236	64	r(id), r(equipoRolpartido)	equipoRolpartido.setId(id);
237	65	r(equipoRolpartido)	this.equiporolpartidoDao.guardarEquipoRolpartido(equipoRolpartido);
239	34	r(equipos), r(equipos), r(match), r(home), r(away), r(round)	System.out.println(" Ronda : "+(round+1)+" Partido: "+(match+1)+" "+equipos[(home)]
244	67	d(campeonatoSerie), r(idcamserie)	CampeonatoSerie campeonatoSerie=this.campeonatoSerieDao.getCampeonatoSerie(idca
246	68	r(campeonatoSerie)	campeonatoSerie.setEstado(2);
247	69	r(campeonatoSerie)	this.campeonatoSerieDao.modificarCampeonatoSerie(campeonatoSerie);
253	70, 86		if(request.getParameter("update")!=null){
254	71	d(codrolp)	String [] codrolp=request.getParameterValues("codrolp");
255	72	d(codArb)	String [] codArb=request.getParameterValues("codArb");
256	73	d(fecha)	String [] fecha=request.getParameterValues("fecha");
257	74	d(hora)	String [] hora=request.getParameterValues("hora");
259	75	d(i)	for (int i = 0; i < codrolp.length; i++) {
259	77, 85	r(codrolp), r(i)	for (int i = 0; i < codrolp.length; i++) {
259	75	d(i)	for (int i = 0; i < codrolp.length; i++) {
260	78	d(rolpartido), r(codrolp), r(i)	Rolpartido rolpartido=this.rolpartidoDao.getRolpartido(Integer.parseInt(codrolp[i]));
262	79	d(fechas)	SimpleDateFormat fechas = new SimpleDateFormat("dd-MM-yyyy");

263		80	r(fechas)	fechas.setLenient(true);
264		81	r(fechas), d(fnac), r(fecha), r(i)	Date fnac = fechas.parse(fecha[i]);
265		82	r(fnac), r(rolpartido)	rolpartido.setFecha(fnac);
267		83	r(rolpartido), r(hora), r(i)	rolpartido.setHora(hora[i]);
268		84	r(rolpartido)	rolpartido.setEstado(1);
269		76	r(rolpartido)	this.rolpartidoDao.modificarRolpartido(rolpartido);
274		86	r(idcamserie)	response.sendRedirect("generarrolpartido.html?idcamserie="+idcamserie);
279		87	r(map), r(idcamserie)	map.put("partidos", this.rolpartidoDao.getRolpartidoBycamserie(idcamserie));
281		88	r(map), r(idcamserie)	map.put("sets", this.setsDao.getSetsBycamserie(idcamserie));
282		89	r(map)	map.put("equiporolpartido", this.equiporolpartidoDao.getAllEquipoRolpartido());

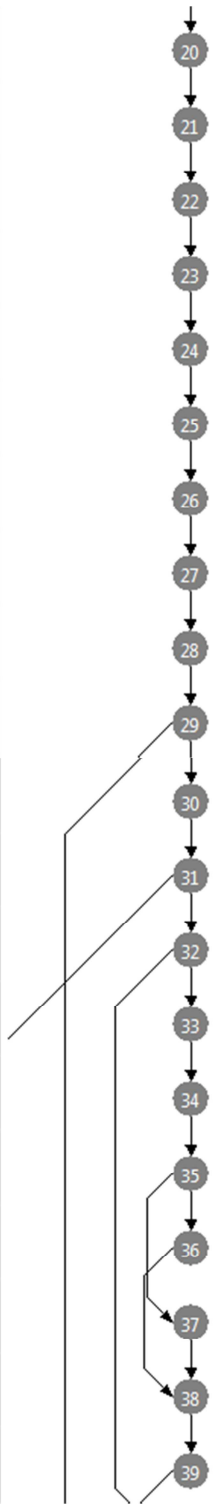
CLASE: RolpartidoControlador.java

METODO Resultados

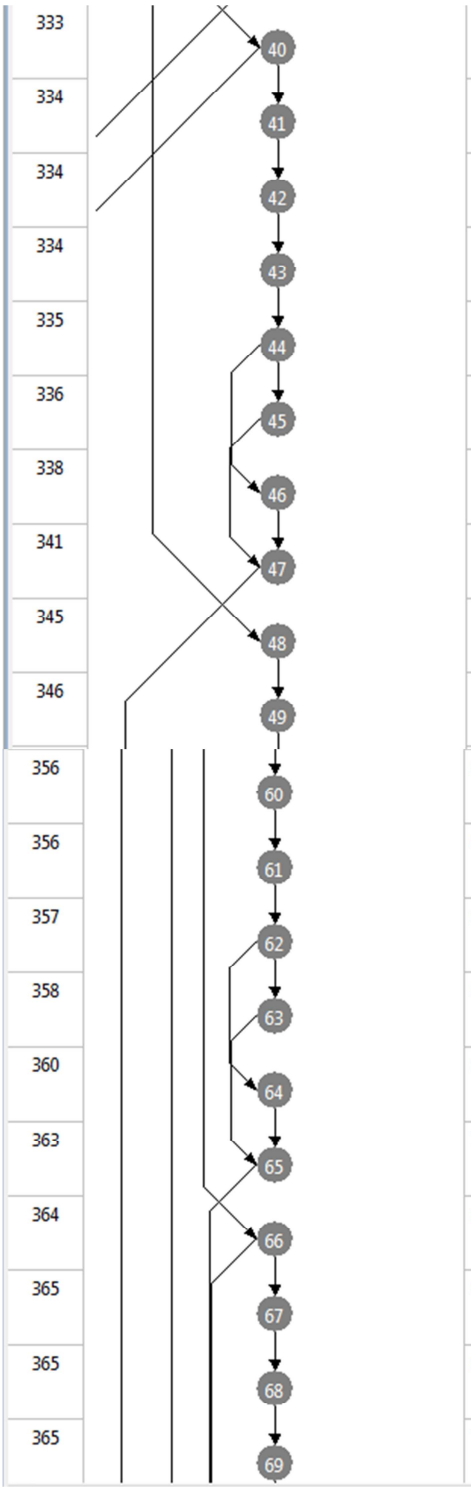


290	1	u(xci), u(session), u(map), u(idcamserie), u(disc), u(walko), u(campeonatoSerieEquipoId1)	public ModelAndView resultados(HttpServletRequest request,HttpServletResponse
290	2		public ModelAndView resultados(HttpServletRequest request,HttpServletResponse
291	3	d(session)	HttpSession session=request.getSession(true);
292	4	d(xci), r(session)	String xci= (String) session.getAttribute("ci");
293	5, 6	r(xci)	if(xci==null)
296	116		return new ModelAndView("Salir");
298	7	d(map)	Map map = new HashMap();
299	8	r(map)	map.put("equipos", this.rolpartidoDao.getEquipoRolpartidobyCodPartido(Integer.parseInt(re
300	9	d(idcamserie)	String idcamserie= Integer.toString(this.rolpartidoDao.getRolpartido(Integer.parseInt(requ
301	10	r(map), r(idcamserie)	map.put("idcamserie",idcamserie);
438	11	r(map)	map.put("codrolp", request.getParameter("codrolp"));
440	12	d(disc)	int disc=this.rolpartidoDao.getRolpartido(Integer.parseInt(request.getPara
441	13	r(map)	map.put("campeonato", this.rolpartidoDao.getRolpartido(Integer.parseInt(request.getParameter(
443	14, 147		if(request.getParameter("save")!=null){
445	15	d(codrolp)	int codrolp=Integer.parseInt(request.getParameter("codrolp"));
446	16	d(rolpartido)	Rolpartido rolpartido=this.rolpartidoDao.getRolpartido(Integer.parseInt(request.get
447	17	r(rolpartido)	rolpartido.setObservacion(request.getParameter("observacion"));
448	18	r(rolpartido)	rolpartido.setEstado(2);
449	19	d(coddis), r(rolpartido)	int coddis=rolpartido.getCampeonatoSerie().getCampeonato().getDisciplin
451	20	d(teams)	String [] teams=request.getParameterValues("team");

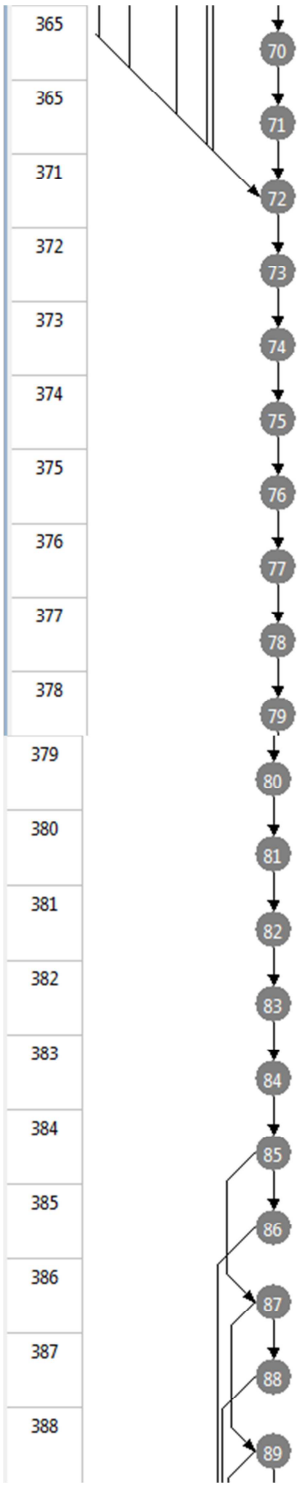
315	
316	
317	
317	
318	
318	
318	
318	
319	
321	
322	
323	
325	
326	
326	
327	
328	
330	
332	
332	



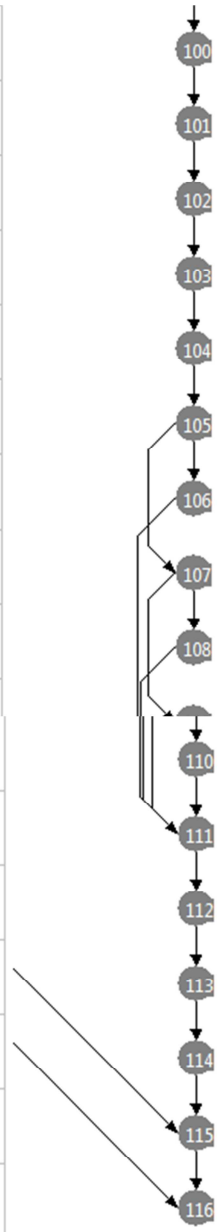
21	r(codrolp), r(teams), d(equipo1id)	EquipoRolpartidoId equipo1id=new EquipoRolpartidoId(Integer.parseInt (teams[0]), codrolp);
22	r(codrolp), r(teams), d(equipo2id)	EquipoRolpartidoId equipo2id=new EquipoRolpartidoId(Integer.parseInt (teams[1]), codrolp);
23	d(resultado1)	int resultado1=0, resultado2=0;
24	d(resultado2)	int resultado1=0, resultado2=0;
25	d(ganador1)	int ganador1=0, ganador2=0, puntos1=0, puntos2=0;
26	d(ganador2)	int ganador1=0, ganador2=0, puntos1=0, puntos2=0;
27	d(puntos1)	int ganador1=0, ganador2=0, puntos1=0, puntos2=0;
28	d(puntos2)	int ganador1=0, ganador2=0, puntos1=0, puntos2=0;
29	d(resul)	int resul=0;
30, 48		if(request.getParameter("partido").equals("wo")){
31	d(walko)	String walko=request.getParameter("walko");
32, 72	r(walko)	if(walko.equals("0")){
33, 40	r(walko), r(teams)	}else if(walko.equals(teams[0])){
34	d(ganador1)	ganador1=1; ganador2=0;
35	d(ganador2)	ganador1=1; ganador2=0;
36, 37	r(coddis)	if(coddis==8){
38	d(puntos1)	puntos1=2;
38	d(puntos1)	puntos1=3;
39	d(puntos2)	puntos2=0; resul=1;
72	d(resul)	puntos2=0; resul=1;



333	41, 72	r(walko), r(teams)	}else if(walko.equals(teams[1])){
334	42	d(ganador1)	ganador1=0; ganador2=1; puntos1=0;
334	43	d(ganador2)	ganador1=0; ganador2=1; puntos1=0;
334	44	d(puntos1)	ganador1=0; ganador2=1; puntos1=0;
335	45, 46	r(coddis)	if(coddis==8){
336	47	d(puntos2)	puntos2=2;
338	47	d(puntos2)	puntos2=3;
341	72	d(resul)	resul=2;
345	49	d(resultado1), r(teams)	resultado1=Integer.parseInt(request .getParameter(teams[0]));
346	50	d(resultado2), r(teams)	resultado2=Integer.parseInt(request .getParameter(teams[1]));
356	61	d(ganador2)	ganador1=0; ganador2=1; puntos1=0;
356	62	d(puntos1)	ganador1=0; ganador2=1; puntos1=0;
357	63, 64	r(coddis)	if(coddis==8){
358	65	d(puntos2)	puntos2=2;
360	65	d(puntos2)	puntos2=3;
363	72	d(resul)	resul=2;
364	67, 72	r(resultado1), r(resultado2)	}else if(resultado1==resultado2){
365	68	d(ganador1)	ganador1=0; ganador2=0; puntos1=1; puntos2=1; resul=3;
365	69	d(ganador2)	ganador1=0; ganador2=0; puntos1=1; puntos2=1; resul=3;
365	70	d(puntos1)	ganador1=0; ganador2=0; puntos1=1; puntos2=1; resul=3;

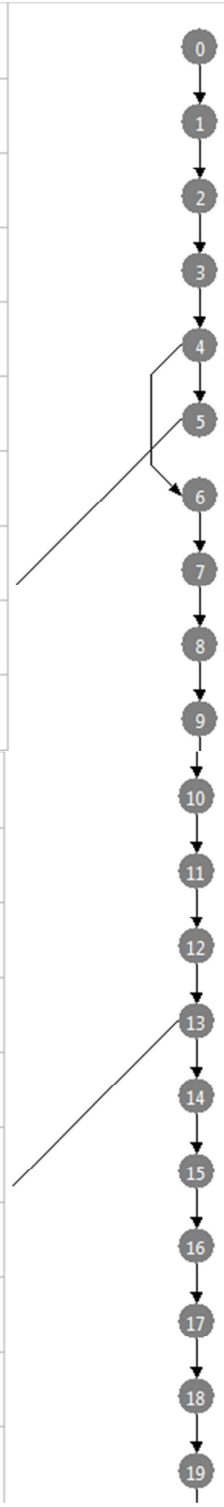


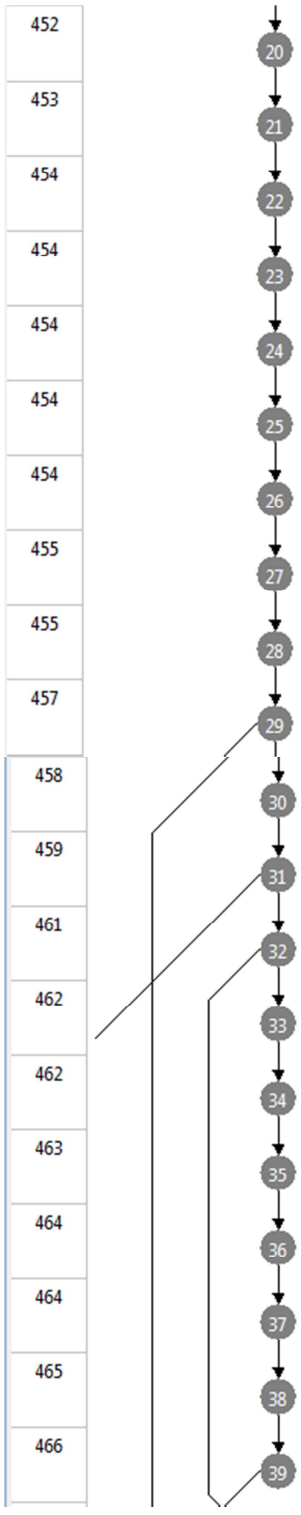
71	d(puntos2)	ganador1=0; ganador2=0; puntos1=1; puntos2=1; resul=3;
72	d(resul)	ganador1=0; ganador2=0; puntos1=1; puntos2=1; resul=3;
73	d(equipo1)	EquipoRolpartido equipo1=new EquipoRolpartido();
74	r(equipo1id), r(equipo1)	equipo1.setId(equipo1id);
75	r(ganador1), r(equipo1)	equipo1.setGanador(ganador1);
76	r(resultado1), r(equipo1)	equipo1.setResultado(resultado1);
77	r(puntos1), r(equipo1)	equipo1.setPuntos(puntos1);
78	r(equipo1)	this.equiporolpartidoDao.modificar EquipoRolpartido(equipo1);
79	d(campeonatoSerieEquipoId1), r(teams)	CampeonatoSerieEquipoId campeonatoSerieEquipoId1=new
80	r(teams), d(campeonatoSerieEquipo1)	CampeonatoSerieEquipo campeonatoSerieEquipo1=this.cam
81	r(campeonatoSerieEquipoId1), r(campeonatoSerieEquipo1)	campeonatoSerieEquipo1.setId(cam peonatoSerieEquipoId1);
82	r(campeonatoSerieEquipo1)	System.out.println(campeonatoSeri eEquipo1.getGoles());
83	r(resultado1), r(campeonatoSerieEquipo1),	campeonatoSerieEquipo1.setGoles(campeonatoSerieEquipo1.getGoles(
84	r(puntos1), r(campeonatoSerieEquipo1),	campeonatoSerieEquipo1.setPuntos (campeonatoSerieEquipo1.getPunt
85	r(campeonatoSerieEquipo1), r(campeonatoSerieEquipo1)	campeonatoSerieEquipo1.setPartid os(campeonatoSerieEquipo1.getPar
86, 87	r(resul)	if(resul==3){
91	r(campeonatoSerieEquipo1), r(campeonatoSerieEquipo1)	campeonatoSerieEquipo1.setPartid ose(campeonatoSerieEquipo1.getPa
88, 89	r(resul)	}else if(resul==1){
91	r(campeonatoSerieEquipo1), r(campeonatoSerieEquipo1)	campeonatoSerieEquipo1.setPartid osg(campeonatoSerieEquipo1.getP
90, 91	r(resul)	}else if(resul==2){

401		101	d(campeonatoSerieEquipo2), r(teams)	CampeonatoSerieEquipo campeonatoSerieEquipo2=this.campeonatoSerieEquipoDao.getCampe
402		102	r(campeonatoSerieEquipoId2), r(campeonatoSerieEquipo2)	campeonatoSerieEquipo2.setId(campeonatoSerieEquipoId2);
403		103	r(resultado2), r(campeonatoSerieEquipo2), r(campeonatoSerieEquipo2)	campeonatoSerieEquipo2.setGoles(campeonatoSerieEquipo2.getGoles()+resultado2);
404		104	r(puntos2), r(campeonatoSerieEquipo2), r(campeonatoSerieEquipo2)	campeonatoSerieEquipo2.setPuntos(campeonatoSerieEquipo2.getPuntos()+puntos2);
405		105	r(campeonatoSerieEquipo2), r(campeonatoSerieEquipo2)	campeonatoSerieEquipo2.setPartidos(campeonatoSerieEquipo2.getPartidos()+1);
406		106, 107	r(resul)	if(resul==3){
407		111	r(campeonatoSerieEquipo2), r(campeonatoSerieEquipo2)	campeonatoSerieEquipo2.setPartidose(campeonatoSerieEquipo2.getPartidose()+1);
408		108, 109	r(resul)	}else if(resul==2){
409		111	r(campeonatoSerieEquipo2), r(campeonatoSerieEquipo2)	campeonatoSerieEquipo2.setPartidosg(campeonatoSerieEquipo2.getPartidosg()+1);
410		110, 111	r(resul)	}else if(resul==1){
411		111	r(campeonatoSerieEquipo2), r(campeonatoSerieEquipo2)	campeonatoSerieEquipo2.setPartidosp(campeonatoSerieEquipo2.getPartidosp()+1);
413		112	r(resultado1), r(campeonatoSerieEquipo2), r(campeonatoSerieEquipo2)	campeonatoSerieEquipo2.setGolescontra(campeonatoSerieEquipo2.getGolescontra()+resultado1);
414		113	r(campeonatoSerieEquipo2)	this.campeonatoSerieEquipoDao.modificarCampeonatoSerieEquipo(campeonatoSerieEquipo2);
416		114	r(rolpartido)	this.rolpartidoDao.modificarRolpartido(rolpartido);
418		115	r(idcamserie)	response.sendRedirect("generarrolpartido.html?idcamserie="+idcamserie);
421		116	r(map)	return new ModelAndView("resultados",map);
422			u(xci), u(session), u(map), u(idcamserie), u(disc), u(walko), u(campeonatoSerieEquipoId1)	}

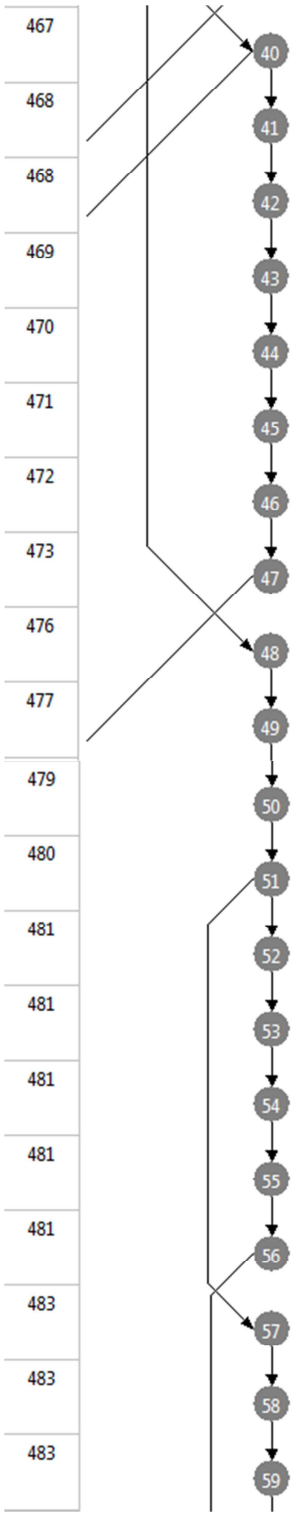
CLASE: RolpartidoControlador.java

METODO: resultadosvoley

426		1	u(resultado1), u(campeonatoSerieEquipoId1), u(resultado2)	public ModelAndView resultadosvoley(HttpServletRequest request,HttpServletResponse
426		2		public ModelAndView resultadosvoley(HttpServletRequest request,HttpServletResponse
427		3	d(session)	HttpSession session=request.getSession(true);
428		4	d(xci), r(session)	String xci= (String) session.getAttribute("ci");
429		5, 6	r(xci)	if(xci==null)
432		148		return new ModelAndView("Salir");
434		7	d(map)	Map map = new HashMap();
435		8	r(map)	map.put("equipos", this.rolpartidoDao.getEquipoRolpartidoby(CodPartido(Integer.parseInt(re
436		9	d(idcamserie)	String idcamserie= Integer.toString(this.rolpartidoDao.getRolpartido(Integer.parseInt(requ
437		10	r(map), r(idcamserie)	map.put("idcamserie",idcamserie);
438		11	r(map)	map.put("codrolp", request.getParameter("codrolp"));
440		12	d(disc)	int disc=this.rolpartidoDao.getRolpartido(Integer.parseInt(request.getPara
441		13	r(map)	map.put("campeonato", this.rolpartidoDao.getRolpartido(Integer.parseInt(request.getParameter(
443		14, 147		if(request.getParameter("save")!=null){
445		15	d(codrolp)	int codrolp=Integer.parseInt(request.getParameter("codrolp"));
446		16	d(rolpartido)	Rolpartido rolpartido=this.rolpartidoDao.getRolpartido(Integer.parseInt(request.get
447		17	r(rolpartido)	rolpartido.setObservacion(request.getParameter("observacion"));
448		18	r(rolpartido)	rolpartido.setEstado(2);
449		19	d(coddis), r(rolpartido)	int coddis=rolpartido.getCampeonatoSerie().getCampeonato().getDisciplin
451		20	d(teams)	String [] teams=request.getParameterValues("team");

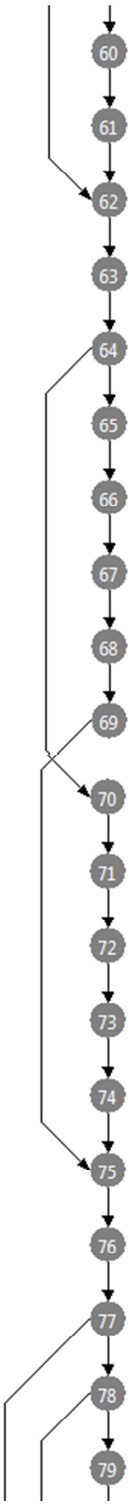


21	r(codrolp), r(teams), d(equipo1id)	EquipoRolpartidoId equipo1id=new EquipoRolpartidoId(Integer.parseInt (teams[0]), codrolp);
22	r(codrolp), r(teams), d(equipo2id)	EquipoRolpartidoId equipo2id=new EquipoRolpartidoId(Integer.parseInt (teams[1]), codrolp);
23	d(ganador1set)	int ganador1set=0, ganador2set=0, puntos1=0, puntos2=0; int resul=0;
24	d(ganador2set)	int ganador1set=0, ganador2set=0, puntos1=0, puntos2=0; int resul=0;
25	d(puntos1)	int ganador1set=0, ganador2set=0, puntos1=0, puntos2=0; int resul=0;
26	d(puntos2)	int ganador1set=0, ganador2set=0, puntos1=0, puntos2=0; int resul=0;
27	d(resul)	int ganador1set=0, ganador2set=0, puntos1=0, puntos2=0; int resul=0;
28	d(puntosteamuno)	int puntosteamuno=0, puntosteamdos=0;
29	d(puntosteamdos)	int puntosteamuno=0, puntosteamdos=0;
30, 48		if(request.getParameter("partido").equals("wo")){
31	d(walko)	String walko=request.getParameter("walko");
32, 119	r(walko)	if(walko.equals("0")){
33, 40	r(walko), r(teams)	}else if(walko.equals(teams[0])){
34	d(ganador1set)	ganador1set=1; ganador2set=0;
35	d(ganador2set)	ganador1set=1; ganador2set=0;
36	d(puntos1)	puntos1=2;
37	d(puntos2)	puntos2=0; resul=1;
38	d(resul)	puntos2=0; resul=1;
39	d(puntosteamuno)	puntosteamuno=2;
119	d(puntosteamdos)	puntosteamdos=0;

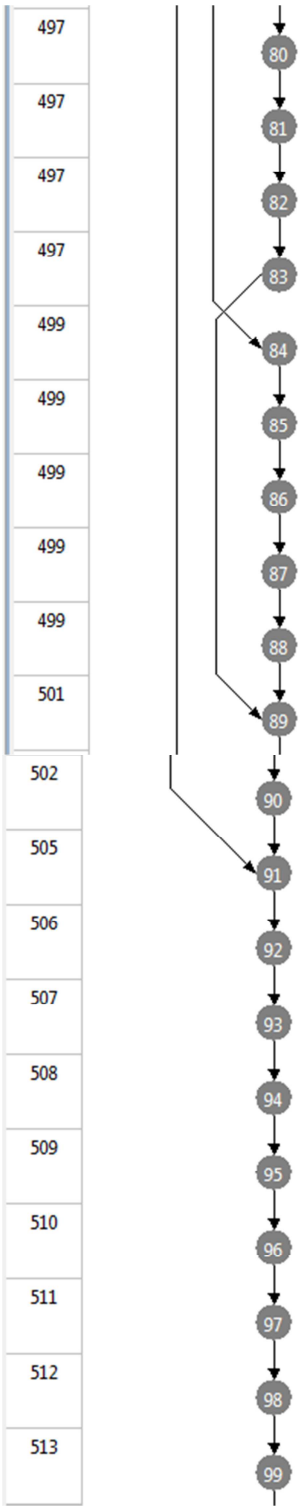


41, 119	r(walko), r(teams)	}else if(walko.equals(teams[1])){
42	d(ganador1set)	ganador1set=0; ganador2set=1;
43	d(ganador2set)	ganador1set=0; ganador2set=1;
44	d(puntos1)	puntos1=0;
45	d(puntos2)	puntos2=2;
46	d(resul)	resul=2;
47	d(puntosteamuno)	puntosteamuno=0;
119	d(puntosteamdos)	puntosteamdos=2;
49	d(resultado1), r(teams)	String resultado1[]=request.getParameter
50	d(resultado2), r(teams)	String resultado2[]=request.getParameter
51	r(resultado1)	System.out.println(resultado1.length);
52, 57	r(resultado1), r(resultado2)	if(Integer.parseInt(resultado1[0])>Integer.parseInt(resultado2[0])){
53	d(ganador1set), r(ganador1set)	ganador1set=ganador1set+1; ganador2set=ganador2set+0;
54	d(ganador2set), r(ganador2set)	ganador1set=ganador1set+1; ganador2set=ganador2set+0;
55	d(puntos1)	ganador1set=ganador1set+1; ganador2set=ganador2set+0;
56	d(puntos2)	ganador1set=ganador1set+1; ganador2set=ganador2set+0;
62	d(resul)	ganador1set=ganador1set+1; ganador2set=ganador2set+0;
58	d(ganador1set), r(ganador1set)	ganador1set=ganador1set+0; ganador2set=ganador2set+1;
59	d(ganador2set), r(ganador2set)	ganador1set=ganador1set+0; ganador2set=ganador2set+1;
60	d(puntos1)	ganador1set=ganador1set+0; ganador2set=ganador2set+1;

483	
483	
485	
486	
487	
488	
488	
488	
488	
488	
490	
490	
490	
490	
490	
492	
493	
495	
496	
497	



61	d(puntos2)	ganador1set=ganador1set+0; ganador2set=ganador2set+1;
62	d(resul)	ganador1set=ganador1set+0; ganador2set=ganador2set+1;
63	r(resultado1), d(puntosteamuno), r(puntosteamuno)	puntosteamuno=puntosteamuno+Integer.parseInt(resultado1[0]);
64	r(resultado2), d(puntosteamdos), r(puntosteamdos)	puntosteamdos=puntosteamdos+Integer.parseInt(resultado2[0]);
65, 70	r(resultado1), r(resultado2)	if(Integer.parseInt(resultado1[1])>Integer.parseInt(resultado2[1])){
66	d(ganador1set), r(ganador1set)	ganador1set=ganador1set+1; ganador2set=ganador2set+0;
67	d(ganador2set), r(ganador2set)	ganador1set=ganador1set+1; ganador2set=ganador2set+0;
68	d(puntos1)	ganador1set=ganador1set+1; ganador2set=ganador2set+0;
69	d(puntos2)	ganador1set=ganador1set+1; ganador2set=ganador2set+0;
75	d(resul)	ganador1set=ganador1set+1; ganador2set=ganador2set+0;
71	d(ganador1set), r(ganador1set)	ganador1set=ganador1set+0; ganador2set=ganador2set+1;
72	d(ganador2set), r(ganador2set)	ganador1set=ganador1set+0; ganador2set=ganador2set+1;
73	d(puntos1)	ganador1set=ganador1set+0; ganador2set=ganador2set+1;
74	d(puntos2)	ganador1set=ganador1set+0; ganador2set=ganador2set+1;
75	d(resul)	ganador1set=ganador1set+0; ganador2set=ganador2set+1;
76	r(resultado1), d(puntosteamuno), r(puntosteamuno)	puntosteamuno=puntosteamuno+Integer.parseInt(resultado1[1]);
77	r(resultado2), d(puntosteamdos), r(puntosteamdos)	puntosteamdos=puntosteamdos+Integer.parseInt(resultado2[1]);
78, 91		if(request.getParameter("sets").equals("3")){
79, 84	r(resultado1), r(resultado2)	if(Integer.parseInt(resultado1[2])>Integer.parseInt(resultado2[2])){
80	d(ganador1set), r(ganador1set)	ganador1set=ganador1set+1; ganador2set=ganador2set+0;

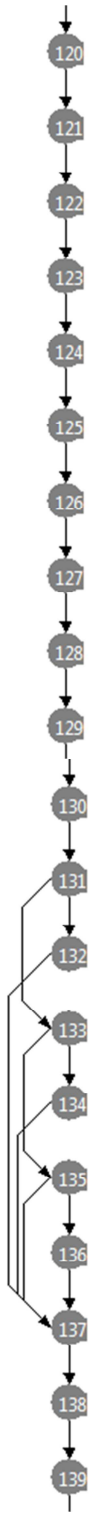


497		81	d(ganador2set), r(ganador2set)	ganador1set=ganador1set+1; ganador2set=ganador2set+0;
497		82	d(puntos1)	ganador1set=ganador1set+1; ganador2set=ganador2set+0;
497		83	d(puntos2)	ganador1set=ganador1set+1; ganador2set=ganador2set+0;
497		89	d(resul)	ganador1set=ganador1set+1; ganador2set=ganador2set+0;
499		85	d(ganador1set), r(ganador1set)	ganador1set=ganador1set+0; ganador2set=ganador2set+1;
499		86	d(ganador2set), r(ganador2set)	ganador1set=ganador1set+0; ganador2set=ganador2set+1;
499		87	d(puntos1)	ganador1set=ganador1set+0; ganador2set=ganador2set+1;
499		88	d(puntos2)	ganador1set=ganador1set+0; ganador2set=ganador2set+1;
499		89	d(resul)	ganador1set=ganador1set+0; ganador2set=ganador2set+1;
501		90	r(resultado1), d(puntosteamuno), r(puntosteamuno)	puntosteamuno=puntosteamuno+Integer.parseInt(resultado1[2]);
502		91	r(resultado2), d(puntosteamdos), r(puntosteamdos)	puntosteamdos=puntosteamdos+Integer.parseInt(resultado2[2]);
505		92	r(puntosteamdos), r(puntosteamuno)	System.out.println("puntos del team uno "+puntosteamuno+"
506		93	r(ganador1set), r(ganador2set)	System.out.println("sets ganados team uno "+ganador1set+" sets
507		94	d(equipo1)	EquipoRolpartido equipo1=new EquipoRolpartido();
508		95	r(equipo1), r(equipo1id)	equipo1.setId(equipo1id);
509		96	r(equipo1), r(ganador1set)	equipo1.setGanador(ganador1set);
510		97	r(equipo1), r(puntosteamuno)	equipo1.setResultado(puntosteamuno);
511		98	r(equipo1), r(puntos1)	equipo1.setPuntos(puntos1);
512		99	r(equipo1)	this.equiporolpartidoDao.modificarEquipoRolpartido(equipo1);
513		100	d(campeonatoSerieEquipoId1), r(teams)	CampeonatoSerieEquipoId campeonatoSerieEquipoId1=new

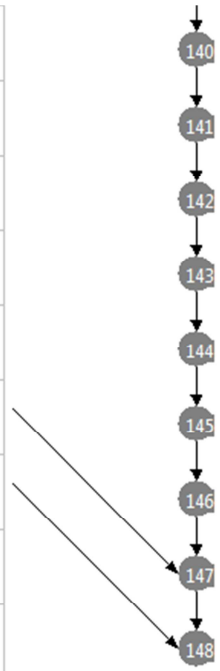
514	100
515	101
516	102
517	103
518	104
519	105
520	106
521	107
522	108
523	109
524	110
525	111
527	112
528	113
530	114
531	115
532	116
533	117
534	118
536	119

101	d(campeonatoSerieEquipo1), r(teams)	CampeonatoSerieEquipo campeonatoSerieEquipo1=this.campeonatoSerieEquipoDao.getCampe
102	r(campeonatoSerieEquipoId1), r(campeonatoSerieEquipo1)	campeonatoSerieEquipo1.setId(campeonatoSerieEquipoId1);
103	r(campeonatoSerieEquipo1)	System.out.println(campeonatoSerieEquipo1.getGoles());
104	r(campeonatoSerieEquipo1), r(campeonatoSerieEquipo1), r(puntosteamuno)	campeonatoSerieEquipo1.setGoles(campeonatoSerieEquipo1.getGoles()+puntosteamuno);
105	r(campeonatoSerieEquipo1), r(campeonatoSerieEquipo1), r(puntos1)	campeonatoSerieEquipo1.setPuntos(campeonatoSerieEquipo1.getPuntos()+puntos1);
106	r(campeonatoSerieEquipo1), r(campeonatoSerieEquipo1)	campeonatoSerieEquipo1.setPartidos(campeonatoSerieEquipo1.getPartidos()+1);
107, 108	r(resul)	if(resul==3){
112	r(campeonatoSerieEquipo1), r(campeonatoSerieEquipo1)	campeonatoSerieEquipo1.setPartidose(campeonatoSerieEquipo1.getPartidose()+1);
109, 110	r(resul)	}else if(resul==1){
112	r(campeonatoSerieEquipo1), r(campeonatoSerieEquipo1)	campeonatoSerieEquipo1.setPartidosg(campeonatoSerieEquipo1.getPartidosg()+1);
111, 112	r(resul)	}else if(resul==2){
112	r(campeonatoSerieEquipo1), r(campeonatoSerieEquipo1)	campeonatoSerieEquipo1.setPartidosp(campeonatoSerieEquipo1.getPartidosp()+1);
113	r(campeonatoSerieEquipo1), r(campeonatoSerieEquipo1), r(puntosteamdos)	campeonatoSerieEquipo1.setGolescontra(campeonatoSerieEquipo1.getGolescontra()+puntosteamdos);
114	r(campeonatoSerieEquipo1)	this.campeonatoSerieEquipoDao.modificarCampeonatoSerieEquipo(campeonatoSerieEquipo1);
115	d(setsId1), r(teams)	SetsId setsId1=new SetsId(Integer.parseInt(request.getParameter("idcamserie"))
116	d(setsteam1), r(idcamserie), r(teams)	Sets setsteam1=this.setsDao.getSetsBycamserieandEquipo(idcamserie
117	r(setsteam1), r(setsteam1), r(ganador1set)	setsteam1.setSetg(setsteam1.getSetg()+ganador1set);
118	r(setsteam1), r(setsteam1), r(ganador2set)	setsteam1.setSetp(setsteam1.getSetp()+ganador2set);
119	r(setsteam1)	this.setsDao.modificarSets(setsteam1);
120	d(equipo2)	EquipoRolpartido equipo2=new EquipoRolpartido();

537	
538	
539	
540	
541	
542	
543	
544	
545	
546	
547	
548	
549	
550	
551	
552	
553	
555	
556	
558	

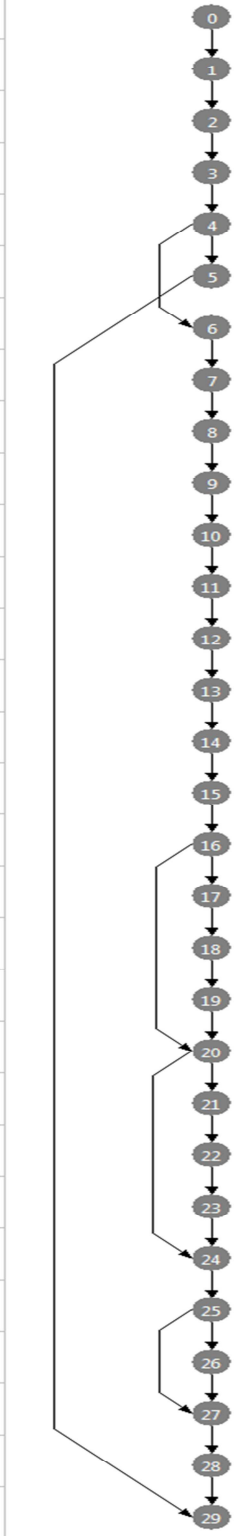


121	r(equipo2id), r(equipo2)	equipo2.setId(equipo2id);
122	r(ganador2set), r(equipo2)	equipo2.setGanador(ganador2set);
123	r(puntosteamdos), r(equipo2)	equipo2.setResultado(puntosteamdos);
124	r(puntos2), r(equipo2)	equipo2.setPuntos(puntos2);
125	r(equipo2)	this.equiporolpartidoDao.modificarEquipoRolpartido(equipo2);
126	d(campeonatoSerieEquipoId2), r(teams)	CampeonatoSerieEquipoId campeonatoSerieEquipoId2=new CampeonatoSerieEquipoId(Integer
127	d(campeonatoSerieEquipo2), r(teams)	CampeonatoSerieEquipo campeonatoSerieEquipo2=this.campeonatoSerieEquipoDao.getCampe
128	r(campeonatoSerieEquipoId2), r(campeonatoSerieEquipo2)	campeonatoSerieEquipo2.setId(campeonatoSerieEquipoId2);
129	r(campeonatoSerieEquipo2), r(campeonatoSerieEquipo2), r(puntosteamdos)	campeonatoSerieEquipo2.setGoles(campeonatoSerieEquipo2.getGoles()+puntosteamdos);
130	r(puntos2), r(campeonatoSerieEquipo2), r(campeonatoSerieEquipo2)	campeonatoSerieEquipo2.setPuntos(campeonatoSerieEquipo2.getPuntos()+puntos2);
131	r(campeonatoSerieEquipo2), r(campeonatoSerieEquipo2)	campeonatoSerieEquipo2.setPartidos(campeonatoSerieEquipo2.getPartidos()+1);
132, 133	r(resul)	if(resul==3){
137	r(campeonatoSerieEquipo2), r(campeonatoSerieEquipo2)	campeonatoSerieEquipo2.setPartidose(campeonatoSerieEquipo2.getPartidose()+1);
134, 135	r(resul)	}else if(resul==2){
137	r(campeonatoSerieEquipo2), r(campeonatoSerieEquipo2)	campeonatoSerieEquipo2.setPartidosg(campeonatoSerieEquipo2.getPartidosg()+1);
136, 137	r(resul)	}else if(resul==1){
137	r(campeonatoSerieEquipo2), r(campeonatoSerieEquipo2)	campeonatoSerieEquipo2.setPartidosp(campeonatoSerieEquipo2.getPartidosp()+1);
138	r(campeonatoSerieEquipo2), r(campeonatoSerieEquipo2), r(puntosteamuno)	campeonatoSerieEquipo2.setGolescontra(campeonatoSerieEquipo2.getGolescontra()+puntosteamuno);
139	r(campeonatoSerieEquipo2)	this.campeonatoSerieEquipoDao.modificarCampeonatoSerieEquipo(campeonatoSerieEquipo2);
140	d(setsId2), r(teams)	SetsId setsId2=new SetsId(Integer.parseInt(request.getParameter("idcamserie"))

559		141	r(idcamserie), r(teams), d(setsteam2)	Sets setsteam2=this.setsDao.getSetsBycamserieandEquipo(idcamserie
560		142	r(campeonatoSerieEquipo2), r(setsteam2)	setsteam2.setCampeonatoSerieEquipo(campeonatoSerieEquipo2);
561		143	r(ganador2set), r(setsteam2), r(setsteam2)	setsteam2.setSetg(setsteam2.getSetg()+ganador2set);
562		144	r(ganador1set), r(setsteam2), r(setsteam2)	setsteam2.setSetp(setsteam2.getSetp()+ganador1set);
563		145	r(setsteam2)	this.setsDao.modificarSets(setsteam2);
565		146	r(rolpartido)	this.rolpartidoDao.modificarRolpartido(rolpartido);
567		147	r(idcamserie)	response.sendRedirect("generarrolpartido.html?idcamserie="+idcamserie);
570		148	r(map)	return new ModelAndView("resultadosvoley", map);
571			u(resultado1), u(campeonatoSerieEquipoId1), u(resultado2)	}

CLASE: DisciplinaControlador.java

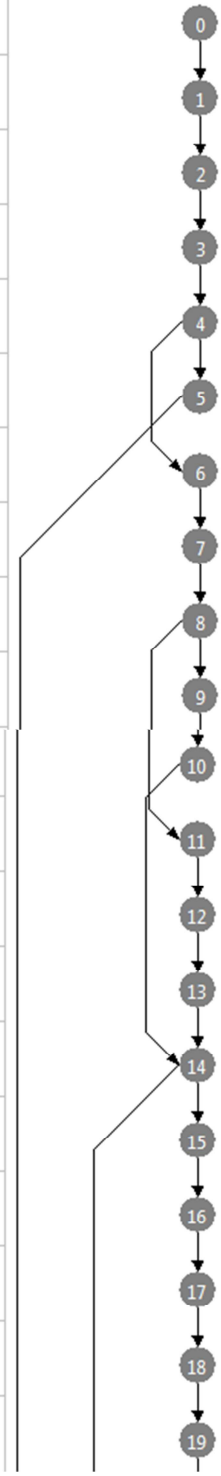
METODO: admdisciplina



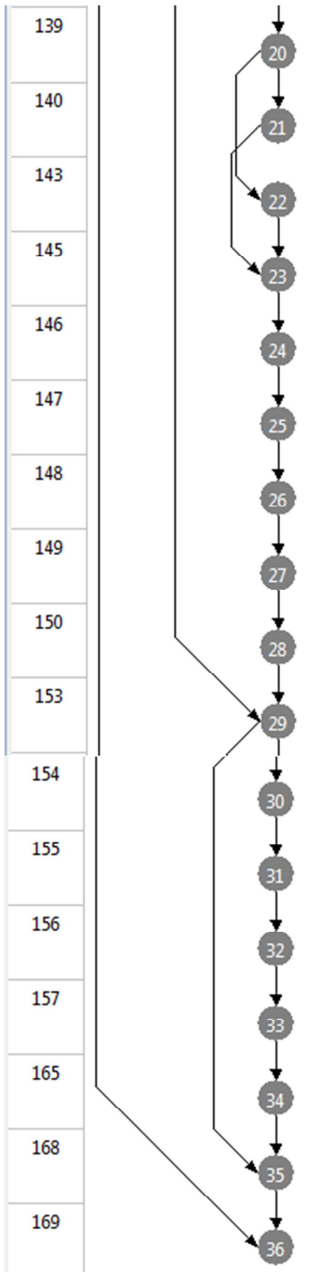
60	1	u(disciplina), u(listadisciplina), u(xci), u(estado), u(ssesion),	public ModelAndView admdisciplina(HttpServletRequest
60	2		public ModelAndView admdisciplina(HttpServletRequest
61	3	d(session)	HttpSession session=request.getSession(true);
62	4	d(xci), r(session)	String xci=(String) (String) session.getAttribute("ci");//PREGUN
63	5, 6	r(xci)	if(xci==null)
66	29		return new ModelAndView("Salir");
70	7	d(ssesion)	Registroacciones ssesion=new Registroacciones();
71	8	r(ssesion)	ssesion.setFecha(new Date());
72	9	r(ssesion)	ssesion.setHora(new Date());
73	10	r(xci), r(ssesion)	ssesion.setUsuarios(this.usuariosDao.getUsuarios(Integer.parseInt(xci)));
74	11	r(ssesion)	ssesion.setAccion("lista de disciplina ");
75	12	r(ssesion)	this.registroaccionesDao.guardarRegistroacciones(ssesion);//
78	13	d(map)	Map map = new HashMap();
81	14	d(listadisciplina)	List listadisciplina = this.disciplinaDao.getAllDisciplina();
82	15	r(listadisciplina), r(map)	map.put("listadisciplina",listadisciplina);
84	16		System.out.println("MENSAJE ERRORS listadisciplina");
87	17, 20		if(request.getParameter("baja")!=null)
89	18	d(disciplina)	Disciplina disciplina=this.disciplinaDao.getDis
90	19	r(disciplina)	disciplina.setEstado(0);
91	20	r(disciplina)	this.disciplinaDao.modificarDisciplina(disciplina);
93	21, 24		if(request.getParameter("alta")!=null)
95	22		Disciplina disciplina=this.disciplinaDao.getDis
96	23		disciplina.setEstado(1);
97	24		this.disciplinaDao.modificarDisciplina(disciplina);
100	25	d(estado)	int estado=1;
101	26, 27		if(request.getParameter("estado")!=null){
102	27	d(estado)	estado=Integer.parseInt(request.getParameter("estado"));
104	28	r(estado), r(map)	map.put("estado", estado);
106	29	r(map)	return new ModelAndView("AdmDisciplina",m
107		u(disciplina), u(listadisciplina), u(xci), u(estado), u(ssesion),	}

CLASE: DisciplinaControlador.java

METODO: abmdisciplina



110	1	u(disciplina), u(xci), u(session), u(map), u(ssesion)	public ModelAndView abmdisciplina(HttpServletRequest
110	2		public ModelAndView abmdisciplina(HttpServletRequest
111	3	d(session)	HttpSession session=request.getSession(true);
112	4	d(xci), r(session)	String xci=(String) session.getAttribute("ci");
113	5, 6	r(xci)	if(xci==null)
116	36		return new ModelAndView("Salir");
121	7	d(map)	Map map = new HashMap();
122	8		Disciplina disciplina;
124	9, 11		if(null != request.getParameter("coddis")){
125	10	d(disciplina)	disciplina=this.disciplinaDao.getDisciplina(Integer.parseInt(request.get
126	14	r(disciplina), r(map)	map.put("Disciplina", disciplina);
128	12	d(disciplina)	disciplina=new Disciplina();
129	13	r(disciplina)	disciplina.setNombredis("");
131	14	r(disciplina), r(map)	map.put("Disciplina", disciplina);
133	15, 29		if(null != request.getParameter("save")){
134	16		System.out.println("[Guardando]");
135	17	d(disciplina)	disciplina=new Disciplina();
136	18	r(disciplina)	disciplina.setNombredis(request.getParameter("nombre"));
137	19	r(disciplina)	disciplina.setEstado(1);
138	20	r(disciplina)	this.disciplinaDao.guardarDisciplina(disciplina);



139	21, 22		if(request.getParameter("volver")!= null){
140	23		response.sendRedirect(request.getParameter("volver"));
143	23		response.sendRedirect("mensaje.html");
145	24	d(ssesion)	Registroacciones ssesion=new Registroacciones(); //
146	25	r(ssesion)	ssesion.setFecha(new Date());
147	26	r(ssesion)	ssesion.setHora(new Date());
148	27	r(xci), r(ssesion)	ssesion.setUsuarios(this.usuariosDao.getUsuarios(Integer.parseInt(xci))) . //
149	28	r(disciplina), r(ssesion)	ssesion.setAccion("creado disciplina con id= "+disciplina.getCoddis()); //
150	29	r(ssesion)	this.registroaccionesDao.guardarRegistroacciones(ssesion);
153	30, 35		if(null != request.getParameter("update")){
154	31		System.out.println("[Actualizando datos]");
155	32	d(disciplina)	disciplina=this.disciplinaDao.getDisciplina(Integer.parseInt(request.getParameter("coddis")));
156	33	r(disciplina)	disciplina.setNombredis(request.getParameter("nombre"));
157	34	r(disciplina)	this.disciplinaDao.modificarDisciplina(disciplina);
165	35		response.sendRedirect("mensaje.html");
168	36	r(map)	return new ModelAndView("abmDisciplina",map);
169		u(disciplina), u(xci), u(session), u(map), u(ssesion)	}