Universidad de las Ciencias Informáticas

Facultad 6

Sistema de gestión de calificaciones de los test de ingreso a la UCI.

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autores:

Marisley Guevara Fernández.

Yudith Orama Rojas.

Tutores:

Msc. Antonio Rey Roque.

Ing. Maypher Román Durán.

Asesor:

Ing. Yannia Moreira Gamboa.

Ciudad de La Habana, julio 2007.

"Año 49 de la Revolución"

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Marisley Guevara Fernández

_____
Firma del Autor

Antonio Rey Roque

_____
Firma del Tutor

Yudith Orama Rojas

_____
Firma del Autor

Maypher Román Durán

_____
Firma del Tutor

DATOS DE CONTACTO

Tutores: Msc. Antonio Rey Roque.
antrey@uci.cu
Universidad de las Ciencias Informáticas.

Ing. Maypher Román Durán.
maypher@uci.cu
Universidad de las Ciencias Informáticas.

Asesor: Ing. Yannia Moreira Gamboa.
ymoreira@uci.cu
Universidad de las Ciencias Informáticas.

OPINIÓN DEL TUTOR DEL TRABAJO DE DIPLOMA

Título: Sistema de gestión de calificaciones de los test de ingreso a la UCI.

Autores: Marisley Guevara Fernández

Yudith Orama Rojas.

Fecha: julio 2007.

El tutor del presente Trabajo de Diploma considera que durante su ejecución los estudiantes mostraron las cualidades que a continuación se detallan:

Los estudiantes comenzaron a trabajar en el proyecto de certificaciones desde etapas tempranas de concepción del sistema, mostrando gran interés y preocupación por el desarrollo del proyecto. Es de destacar la disposición en la preparación autodidacta de herramientas de desarrollo de software que fue necesario estudiar. La comunicación con el tutor fue muy positiva y constante, además los estudiantes manifestaron un interés continuo en aspectos relacionados con el desarrollo de su proyecto, acudiendo a su tutor en todo momento. El intercambio de conocimientos fue recíproco. Ambas compañeras se caracterizan por un sentido de responsabilidad muy elevado. A pesar de los obstáculos que enfrentaron en los inicios del proyecto, eso no las desanimó a y continuaron interesándose por el desarrollo de su trabajo de tesis. La solución informática propuesta no es más que un sistema de gestión de los Test de Ingreso a la Universidad de las Ciencias Informáticas que pretende dar solución a un problema actual en dicha organización, y para lograr eso fue necesario un estudio profundo de algunas tecnologías.

Por todo lo anteriormente expresado considero que las estudiantes están aptas para ejercer como Ingenieros Informáticos; y propongo que se le otorgue al Trabajo de Diploma la calificación de Excelente. Además, considero que los resultados poseen valor para ser publicados.

“Nuestra recompensa se encuentra en el esfuerzo y no en el resultado. Un esfuerzo total es una victoria completa.”

Monadas Karamchan Gandhi.

AGRADECIMIENTOS

…Es justamente la posibilidad de realizar un sueño lo que torna la vida interesante…

Paolo Coelho

Muchas son las personas que nos han ayudado a lo largo de este camino y en las diferentes etapas de nuestras vidas, gracias a ellos hoy alcanzamos una de nuestras metas. Queremos que llegue a todos nuestro más profundo agradecimiento.

Agradecemos especialmente a nuestro Comandante en Jefe Fidel Castro Ruz por darnos la posibilidad de ser partes de este gran proyecto.

A nuestros tutores: Maypher Román Durán por brindarnos sus conocimientos y guiarnos para que este trabajo tuviera la calidad esperada. A Antonio Rey Roque por habernos transmitido apoyo y esperanzas durante estos cinco años, y conducirnos por el mejor camino, en esta última etapa de nuestros estudios. A Mary por la corrección y estilo científicos.

A Yannia Moreira Gamboa por siempre sacar un espacio de su tiempo para atendernos y aclarar nuestras dudas.

A Arieskien Mendoza Guerra por llegar en este momento importante y siempre estar dispuesto a ayudarnos con sus conocimientos de forma incondicional, transmitiéndonos mucho apoyo y fe.

A nuestros padres por ser nuestra inspiración, porque sin ellos este sueño nunca hubiera sido posible.

A todos muchas gracias.

DEDICATORIA

De Marisley

Dedico este trabajo:

A quien ha sido mi ejemplo en todos estos años y me ha enseñado a luchar por lo que quiero mi papá Delso.

A quien a sido mi inspiración, mi vida, mi mamita Maribel.

A mi hermana Dariadne por darme su confianza y amor.

A mis abuelos Rafael, Nora y Margarita por educarme y enseñarme que cuando se lucha con tenacidad se pueden alcanzar los Sueños.

A mi abuelo Armando, que la vida no le dio la posibilidad de verme en este momento, por estar presente en cada uno de los días de mi vida.

A mi familia por creer en mí y apoyarme en todo momento.

A mis amigos por todas las penas y alegrías vividas juntas: Yamilka, Yanedi, Ramón, Yili, Yudelkis, Yudith, Keila, Sianny, Heydi, Karel, Yordan; gracias por su amistad siempre estarán en mi corazón.

A Daichel y a Yayi por haber llegado en un momento importante y brindarme su apoyo y confianza.

Por cuestiones de espacio se quedaron nombres por mencionar, a estos solo les pido que estén convencidos de que los quiero y este trabajo es también para ustedes.

De Yudith

Dedico este trabajo:

A mi nani Arquela con todo el amor del mundo, por ser la luz que ilumina mi vida.

A mi papá Raúl con inmenso amor, por enseñarme que la vida es una y hay que vivirla sin miedo.

A mi hermano Rauli porque lo quiero mucho y adoro el tiempo que pasamos juntos.

A Lila por ser mi segunda madre, abrirme las puertas de su hogar y dejarme formar parte de el.

A Yayi, por ser mi hermana mayor incondicionalmente y apoyarme siempre, además a su esposo Gert porque a pesar de no conocerme hace mucho tiempo me ha dado ánimos en momentos difíciles.

A mis familiares y vecinos que se han preocupado por mí a lo largo de toda la carrera.

A mis amigos por ser parte importante de mi vida, y compartir conmigo los buenos y malos momentos:

- *Los viejos amichis que aun perduran: Ale (donde quiera que estes), Lizbet(mi hermana que nació de dia), Yaritza, Rosa, Yisel, Gricel, Monica, Yailin, Yander, Yoanis, Wendi, Mildre, Yordan (mi primo), Yairi, Zaili, Addiel, Beti, Lores…*
- *Mis amichis de Universidad: Yasser, Keila, Marita, Yudelkis (Y), Heydi, Sianny, Yordan, Miltico…*
- *Carlos (señor mayor) mi simpático amigo por correpondencia.*

Por cuestiones de espacio se quedaron nombres por mencionar, a estos solo les pido que estén convencidos de que los quiero y este trabajo es también para ustedes.

RESUMEN

Las fuentes de ingreso a la Universidad de las Ciencias Informáticas se han diversificado, y la cantidad de aspirantes crece exponencialmente. Cada curso ingresan alrededor de 2000 estudiantes, provenientes de todos los municipios del país. Este año abrieron sus puertas las Facultades Regionales en tres provincias del país, incrementando el número de aspirantes y de posibilidades de estudiar Ciencias Informáticas.

Los optantes realizan un examen psicométrico, este resultado y el aval político constituyen los requisitos fundamentales para el ingreso. Ante la apertura de matrícula, dada fundamentalmente por la creación de las tres Facultades Regionales y su futura creación en otras provincias del país, se hace necesario mejorar la gestión de la información asociada al ingreso a la UCI. Con el objetivo de solucionar esta problemática se propone analizar, diseñar e implementar una aplicación Web que complemente el Sistema de pre-ingreso a la UCI. La aplicación se desarrolló bajo ambiente multiplataforma, utilizando herramientas de software libre y cumpliendo con las políticas de informatización del centro.

ÍNDICE

AGRADECIMIENTOS ..... **II**

DEDICATORIA ..... **III**

RESUMEN ..... **V**

INTRODUCCIÓN **1**

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA. **4**

1.1 TENDENCIAS ACTUALES. 4
1.1.1 Sistemas de ingreso a universidades en América Latina. *5*
1.1.2 Sistemas de ingreso a las universidades en Cuba *6*
1.1.3 Sistemas de ingreso a la Universidad de las Ciencias Informáticas (UCI) *6*
1.2 ANTECEDENTES Y SISTEMAS EXISTENTES. 6
1.2.1 Sistema De ingreso Nacional. *6*
1.2.2 Sistema de Pre-ingreso existente. *6*
1.3 SITUACIÓN PROBLÉMICA. 7
1.4 NECESIDADES ACTUALES DE LOS TRABAJADORES. 7
1.5 TENDENCIAS DE LAS TECNOLOGÍAS EN LAS POLÍTICAS DE INFORMATIZACIÓN DE LA UCI. 7
1.5.1 Tecnologías del lado del cliente. *8*
1.5.2 Tecnologías del lado del servidor. *9*
1.5.3 Herramientas de desarrollo. *11*
CodeIgniter *11*
1.5.4 Servidor de Base de Datos. *13*
1.6 SERVIDOR APACHE. 14
1.7 HERRAMIENTA DE DESARROLLO. 15
1.8 METODOLOGÍA DE DESARROLLO DE SOFTWARE. 16
1.8.1 Rational Unified Process (RUP). *16*
1.8.2 Extreme Programing (XP). *18*
1.8.3 Microsoft Solution Framework (MSF). *19*
1.9 HERRAMIENTA CASE. 19
1.10 LENGUAJE DE MODELACIÓN. 20
UNIFIED MODELING LANGUAJE (UML). 20
1.11 CONCLUSIONES 21

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA. **22**

2.1 Problema y Situación problémica. *22*
2.2 OBJETO DE AUTOMATIZACIÓN. 22
2.2.1 Gestión de los test. *22*
2.2.2 Gestión de códigos de anonimato. *22*
2.2.3 La realización de los reportes. *23*
2.2.4 Implementación del algoritmo. *23*
2.3 INFORMACIÓN QUE SE MANEJA. 24
2.4 PROPUESTA DE SISTEMA. 24
2.4.1 Reglas del negocio. *25*
2.5 DESCRIPCIÓN DE LOS PROCESOS DEL NEGOCIO. 25

2.6 Modelo del Dominio. 26
2.6.1 Conceptos que se utilizan en el modelo de dominio. *26*
2.7.1 Requisitos funcionales. 27
2.7.2 Requisitos no funcionales. 30
2.8 Descripción del sistema propuesto. 32
2.9 Definición de los actores del sistema. 33
2.10 Diagrama de casos de uso del sistema. 40
2.11 Casos de uso por ciclo. 42
2.12 Conclusiones. 42

CAPÍTULO 3: ANÁLISIS Y DISEÑO DEL SISTEMA. **44**

3.1 Análisis. 44
3.2 Modelo conceptual de clases de análisis. 44
3.2.1 Clases de análisis. *44*
3.2.2 Clases de Interfaz *45*
3.2.3 Clases de Entidad *45*
3.2.4 Clases controladoras. *45*
3.3 Diseño. 50
3.3.1 Diagramas de interacción. *51*
3.3.2 Diagramas de clases del diseño *51*
3.4 Diseño de la BD. 59
3.4.1 Diagrama de clases persistentes. *59*
3.4.2 Modelo de datos *60*
3.4.3 Descripción de las tablas. *62*
3.5 Definiciones de diseño que se apliquen. 62
3.6 Conclusiones. 63

CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBA **64**

4.1 Implementación. 64
4.1.1 Diagramas de Despliegue *64*
4.1.2 Diagrama de componentes. *65*
4.2 Conclusiones 71

CONCLUSIONES **72**

RECOMENDACIONES **73**

REFERENCIAS BIBLIOGRÁFICAS. **74**

BIBLIOGRAFÍA. **76**

ANEXO I: CASOS DE USO POR CICLO. **78**

ANEXOS II: DESCRIPCIÓN DE LOS CASOS DE USOS CRÍTICOS. **79**

ANEXOS III: DIAGRAMAS DE SECUENCIA DE LOS CASOS DE USO CRÍTICOS. **87**

ANEXOS IV: DESCRIPCIÓN DE LAS CLASES DEL DISEÑO **100**

ANEXOS V: DESCRIPCIÓN DE LAS ENTIDADES DE LA BASE DE DATOS. **113**

GLOSARIO DE TÉRMINOS. **116**

INTRODUCCIÓN

El desarrollo de aplicaciones de software juega un papel importante en el proceso de informatización y automatización que se lleva a cabo actualmente en todas las esferas de nuestro país, con vistas a mejorar los servicios brindados a la sociedad. La Universidad de las Ciencias Informáticas (UCI), surgida en el año 2002 producto de una idea del Comandante en Jefe Fidel Castro Ruz, tiene como objetivo principal contribuir a la formación de nuevos ingenieros, con la tarea de introducir a nuestro país en el mundo de la Informática. Para esto, cada año ingresan a la UCI alrededor de 2000 estudiantes, provenientes de los 169 municipios del país. Las fuentes de ingreso son: Instituto Pre-Universitario Vocacional de Ciencias Exactas (IPVCE), Instituto Pre-Universitario Urbano (IPU), Instituto Pre-Universitario de Escuelas al Campo (IPUEC), Instituto Politécnico de Informática (IPI) y Orden 18.

La UCI lleva a cabo todo un proceso de informatización en todos sus servicios, la investigación está respondiendo a una necesidad de este proyecto. La comunidad informática crece y se expande cada día más, la UCI como toda universidad, surgida al calor de la Batalla de Ideas, va abriendo sus puertas a los cambios que se originan en las diferentes esferas de la población, tal es el caso del proceso de universalización de la sociedad. Este centro de altos estudios cuenta ya con sus tres primeras Facultades Regionales en las provincias La Habana (Artemisa), Ciego de Ávila (Ceballo) y Granma (Manzanillo), las cuales tienen la matrícula conformada por estudiantes provenientes de los Institutos Politécnicos de Informática (IPI). Todos estos cambios han incidido en el sistema de ingreso tradicional, pues al aumentar las posibilidades de ingresos el número de aspirantes es proporcional y se requiere de una mayor disponibilidad de recursos humanos en el cumplimiento de esta tarea.

EL sistema de ingreso a la UCI es un proceso complejo, que consta de varias etapas. La elección de los estudiantes comienza a nivel de centro, en todos los IPVCE, IPU, IPUEC, e IPI. Para esto se crean las Comisiones provinciales de ingreso, que dirigen el proceso de captación en cada provincia. Seguido de esto se publica en cada uno de los centros un escalafón con los estudiantes que cumplen con las condiciones que se exige; para optar por la UCI se debe tener más de 90 puntos en Matemática, en el caso de los estudiantes de IPI se les miden además las asignaturas de lenguaje y técnicas de la Programación. Todo estudiante que cumpla con estas características puede asistir a una reunión con directivos de la UCI, en la cual se les explica con claridad cómo es el proceso. Luego, a los aspirantes se les dan dos o tres días para que decidan. A los que finalmente optan por la carrera se les da un aval político, el cual tiene que ser otorgado por la Federación de Estudiantes de la Enseñanza Media y la Unión

de Jóvenes Comunistas de cada centro, estos estudiantes deben cumplir con la condición de ser ciudadano cubano. De ser aprobado este aval y cumplir con las condiciones requeridas, tienen la opción de presentarse a resolver el test de ingreso. Los Test son trasladados a la UCI donde son calificados, estos resultados son procesados para obtener la nota real que va a tener el estudiante. Lo anterior conforma el escalafón en cada lugar por el cual se otorgan las plazas, en función de las cantidades asignadas a cada provincia y valorando el equilibrio de sexo.

Actualmente existe un sistema que realiza la gestión de las calificaciones para un Test de ingreso determinado, dificultando la posibilidad de trabajar con diferentes Test. No se cuenta con ningún tipo de documentación, lo que dificultó agregar nuevas funcionalidades que se requieren en la actualidad. En resumen, el actual sistema de pre-ingreso no es flexible ante los nuevos requerimientos que se necesitan.

Luego de un análisis de la situación del actual sistema, la presente investigación se plantea como problema científico:

¿Cómo mejorar la gestión de la información del sistema de pre-ingreso a la UCI?

El Objeto de estudio es: El proceso de gestión de la información de los sistemas de ingreso a la Educación Superior.

El campo de acción: El proceso de gestión de la información del pre-ingreso a la UCI.

El objetivo general que se persigue es: Desarrollar una aplicación informática que mejore la gestión de los test de ingreso a la UCI.

Los objetivos específicos que se persiguen son:

1. Analizar, diseñar e implementar los cinco módulos que conforman la aplicación.
2. Crear los manuales de usuario necesarios para el uso del sistema.

Se efectuaron una serie de tareas con vistas a la realización del trabajo:

- Diseño de modelos que sirven de base para el desarrollo del sistema.
- Estudio del funcionamiento del sistema de ingreso a la UCI, específicamente la calificación de los test.
- Realización de entrevistas a los clientes.

- Estudio del lenguaje y la metodología utilizada.
- Estudio de las tecnologías de software libre que proponen los requisitos de informatización de la UCI.
- Estudio de las características que requiere la confección de un manual de usuario o documento de ayuda.

El contenido del presente Trabajo de Diploma está estructurado de la siguiente manera:

Capítulo 1. Fundamentación Teórica: En este capítulo se exponen los fundamentos generales que sirven de base teórica a la solución y concepción del problema. Se tratan las tendencias actuales y los antecedentes y sistemas existentes. Se explica el porqué se hace necesario un nuevo sistema, detallando las tecnologías y lenguajes de programación empleados en su creación. Se exponen tres de las metodologías de desarrollo de software más utilizadas: Rational Unified Process (RUP), Extreme Programing (XP) y Microsoft Solution Framework (MSF), basadas en el lenguaje de modelado: Unified Modeling Languaje (UML).

Capítulo 2. Características del sistema: El capítulo aborda el objeto de estudio, problema y situación problémica, se detalla el objeto de automatización, la información que se maneja, se realiza una descripción general de la propuesta de sistema. Además se enumeran los requisitos funcionales y no funcionales con los que debe contar el software. Se definen las reglas del negocio e ilustran los diferentes modelos: Modelo del Dominio y Modelo de Casos de Uso del Sistema.

Capítulo 3. Análisis y diseño del sistema: Este capítulo aborda el flujo de trabajo Análisis y Diseño. Para ello se definen las clases del análisis y el Modelo de clases de análisis. En el diseño se realizan los modelos de clases del diseño para cada caso de uso, se definen los diagramas de interacción por cada realización de casos de uso. Se describen las clases de diseño y las tablas de la base de datos; de esta última se define su diagrama de clases persistentes y su modelo de datos.

Capítulo 4. Implementación: En el presente capítulo se aborda lo referente al flujo de trabajo Implementación, para ello se expone el diagrama de despliegue que corresponde a la aplicación que se ha presentado y descrito en los capítulos anteriores. En esta sección quedan reflejados los diagramas de componentes que forman la base de la programación de este proyecto.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA.

En este capítulo se exponen los fundamentos generales que sirven de base teórica a la solución y concepción del problema. Se tratan las tendencias actuales y los antecedentes y sistemas existentes. Se explica el porqué se hace necesario un nuevo sistema, detallando las tecnologías que serán usadas para la realización del mismo. Se mencionan las tecnologías y lenguajes de programación empleados en su creación. Se exponen tres de las metodologías de desarrollo de software más utilizadas: Rational Unified Process (RUP), Extreme Programing (XP) y Microsoft Solution Framework (MSF), basadas en el lenguaje de modelado: Unified Modeling Languaje (UML).

1.1 Tendencias actuales.

Al estudiar la historia de las universidades se puede demostrar que desde sus inicios hasta la actualidad han ido evolucionando y cambiando en muchos aspectos. Uno de estos es el proceso de captación de sus estudiantes. En la antigüedad estos centros de altos estudios eran solo para los hombres nobles que tenían dinero. Sin embargo, en la actualidad, las mujeres tienen el mismo derecho que los hombres a cultivarse, y a ingresar a la Universidad. En muchos países el tema del dinero afecta a personas con talento, pero sin el capital necesario para entrar. Según las características del país se les dificulta en más o menos medida el acceso a las Universidades.

En todas las regiones del planeta existen universidades, y cada una de ellas tiene sus características, las cuales varían según las especialidades estudiadas en ellas. De esa misma manera si lo que se va estudiar es una carrera de ciencia cada universidad de este tipo decide como evaluar a los aspirantes.

Elegidas una carrera y una universidad donde cursarla, inscribirse parece de lo más sencillo. Sin embargo, es un paso que demanda trámites sucesivos, con calendarios estrictos y documentación precisa que presentar, y que sigue con el ingreso, que puede incluir cursos, exámenes y entrevistas. [1]

Quienes están a cargo de las inscripciones en universidades públicas y privadas repiten consejos: leer en detalle la información sobre la institución y la carrera, tener presentes las fechas para cada trámite, tener claro el sistema de ingreso y sus exigencias, y aprovechar el momento para eliminar todas las dudas. [1]

Los consejos antes mencionados tienen un gran valor para todo el interesado en ingresar a una universidad, estar informado es la garantía de no cometer errores en las diferentes decisiones que se deben tomar a lo largo del transcurso del ingreso. Es decir, que este proceso varía según el tipo de

universidad, ejemplo si es publica o privada, si está en un país capitalista o socialista, e incluso según la materia que se estudia (ciencias, humanidades, etc.)

1.1.1 Sistemas de ingreso a universidades en América Latina.

Dos son las modalidades que existen a lo largo de nuestro continente, para ingresar a la educación superior. En algunos casos como Argentina y Uruguay se puede entrar de manera directa, sin prueba y en otros como *Brasil* y *Colombia* hay que aprobar un examen de carácter nacional. [2]

Para hacer una comparación en mayor profundidad, se presentan algunos de los distintos sistemas de ingreso en algunos países de América Latina. [2]

Argentina: Tanto para alumnos argentinos como para extranjeros residentes en la nación, la manera de ingreso es la misma. Cada universidad nacional (pública y gratuita) determina el cupo de cada carrera y la forma de selección de los estudiantes. [2]

De preferencia, se toma un examen de ingreso con materias afines a lo que se quiere estudiar. Por ejemplo: Matemática, Física, Química y Biología para el caso de Medicina; Física, Química, Matemática, Análisis matemático o Algebra para Ingeniería; Biología, Química, Matemática, Semiología para Psicología, etc. entran quienes obtienen las mejores notas hasta llenar el número de vacantes. [2]

Las instituciones de carácter privado, por lo general, no toman un test de admisión, basta con matricularse. [2]

Ecuador: No existe prueba de selección, sin embargo, los interesados deben aprobar algunos certámenes que cada universidad tiene preparado para los alumnos que quieran acceder a la educación superior. [2]

En el caso que el estudiante salga mal evaluado existen los preuniversitarios. Estos preparan al alumno para su buen desempeño en el examen de ingreso. [2]

Colombia: Semestralmente se realizan las pruebas ICFES (Instituto Colombiano para el Fomento de la Educación Superior). Se aplican a jóvenes estudiantes en su onceavo año de educación y son realizados para permitir el acceso a la educación superior. [2]

Esta mide los conocimientos en materias como Matemáticas, Lenguaje, Historia, Filosofía, Biología, Química, Física e Idiomas, este último es escogido por el alumno, de acuerdo con el idioma que se ve en su plantel escolar (inglés, francés o alemán). [2]

1.1.2 Sistemas de ingreso a las universidades en Cuba

En Cuba el curso académico se organiza por semestres; se inicia el primer lunes de septiembre y culmina aproximadamente en las diferentes enseñanzas alrededor de la 2da semana de julio de cada año. Como requisito de ingreso se establecen el título de Bachiller no universitario equivalente a 12 grados de enseñanza preuniversitaria debidamente legalizado, certificación de nacimiento, certificado de salud y 6 fotos tipo pasaporte. Los estudiantes se rigen por los reglamentos vigentes en la Educación Superior Cubana.

1.1.3 Sistemas de ingreso a la Universidad de las Ciencias Informáticas (UCI)

El 2002 fue el año que vio nacer a la UCI bajo el calor de la batalla de ideas, producto de una idea del Comandante en Jefe Fidel Castro. Por la rapidez con que se comienza, los primeros 2000 estudiantes se seleccionan por su disposición a formar parte de la construcción de un sueño, por contar con un aval político aprobado por la UJC y por tener un índice académico por encima de los 90 puntos en las asignaturas de Matemática, Física y Computación. Ya para el curso 2003-2004 los directivos de la Universidad se habían encargado de organizar el ingreso de otra manera. Se decidió aplicar un test de ingreso a los aspirantes, el cual en conjunto con su índice académico de los tres años de bachiller, determinan la asignación de la carrera

1.2 Antecedentes y sistemas existentes.

1.2.1 Sistema De ingreso Nacional.

A nivel nacional existe un sistema que gestiona las calificaciones de las pruebas de ingreso a la Universidad. Los estudiantes de bachillerato u otra fuente de ingreso se presentan a los exámenes de ingreso, los mismos son calificados en sus provincias y los resultaos se procesan en el sistema nacional, el cual se encarga de asignar las carreras según las solicitudes de cada estudiante y el escalafón que alcanzó a nivel provincial.

1.2.2 Sistema de Pre-ingreso existente.

En la Universidad de las Ciencias Informáticas (UCI) existe un sistema para la gestión de los test de ingresos a la Universidad, es una aplicación Web que cuenta con un menú donde se le da la opción a los usuarios de: modificar planilla, modificar administradores, modificar usuarios, calificar y generar Reporte. Está diseñado para un Test determinado, sus interfaces no cuentan con diseño gráfico terminado. Se divide en tres módulos principales, administración, calificación y gestión de reportes.

1.3 Situación problémica.

Los directivos de la UCI necesitan conocer los resultados reales que obtienen los estudiantes en los Test de ingreso, de esto depende la decisión de si el aspirante reúne las condiciones necesarias para ingresar al centro. Con vistas a alcanzar estos objetivos la Universidad cuenta con un sistema de pre-ingreso que gestiona las calificaciones obtenidas en los Test, dando como resultado la nota real que tiene cada estudiante. Esta aplicación no está cumpliendo con todos los requisitos exigidos. No cuenta con los documentos necesarios para su funcionamiento: manual de usuario, documentación técnica, manual de instalación y configuración. No permite realizar la mejor gestión de los test debido a su rigidez de desarrollo. Está diseñado para un examen determinado, esto trae consigo que en el momento de realizar un cambio en los test (preguntas), en cuanto al valor máximo y la pregunta de aciertos, es preciso hacer varios cambios en el código fuente hasta el punto de hacerlo casi todo de nuevo. Este sistema carece de diseño grafico (todo está en blanco). Está implementado en lenguaje (Perl), el cual no cumple con las políticas de informatización del centro.

El software existente está diseñado para dar cumplimiento a los requerimientos que el cliente solicitó al inicio de la Universidad, de tal manera que ahora, después de cinco años de creada, el cliente solicita cambios, que son difíciles de realizar, se hace necesario la creación de un nuevo sistema que siga una metodología de desarrollo y que cumpla con los requisitos que se piden en la actualidad.

1.4 Necesidades actuales de los trabajadores.

El personal de Secretaría General necesita tener disponible un software que preste el servicio de gestionar la nota final de los test de ingreso y los distintos reportes que de estas se puedan derivar. Requiere que la aplicación sea flexible a futuros cambios y constituya una herramienta rápida de búsqueda donde se notifiquen las acciones realizadas por los usuarios en el sistema. Necesitan además tener total acceso para cambiar toda la información que se almacena en el sistema.

1.5 Tendencias de las tecnologías en las políticas de informatización de la UCI.

Organizar el trabajo, garantiza siempre en gran medida la eficiencia de la actividad que se realiza. La UCI está trabajando arduamente para informatizar los procesos internos de todas las áreas que manejan la vida de la Universidad. Los directivos de informatización están definiendo las políticas de desarrollo que se

deben seguir para realizar todas las aplicaciones. Este trabajo forma parte de este gran proyecto, a continuación se mencionan los puntos de esta política por los que se rige.

1.5.1 Tecnologías del lado del cliente.

Las tecnologías del lado del cliente están orientadas preferentemente, como su nombre indica, para ejecutarse en los puestos cliente: HTML, CSS, JavaScript, etc. Esto proporciona las capacidades del cliente que hacen posible la creación de aplicaciones dinámicas de Internet al aprovechar el poder de procesamiento local de las PC y los dispositivos.

JavaScript.

JavaScript es el lenguaje que nos permite interactuar con el navegador de manera dinámica y eficaz, proporcionando a las páginas web dinamismo y vida. [4]

JavaScript no es un lenguaje de programación propiamente dicho. Es un lenguaje script u orientado a documento, como pueden ser los lenguajes de macros que tienen muchos procesadores de texto. [5]

Se trata de un lenguaje de programación del lado del cliente, porque es el navegador el que soporta la carga de procesamiento. Gracias a su compatibilidad con la mayoría de los navegadores modernos, es el lenguaje de programación del lado del cliente más utilizado. [6]

El navegador del cliente es el encargado de interpretar las instrucciones JavaScript y ejecutarlas para realizar estos efectos e interactividades, de modo que el mayor recurso, y tal vez el único, con que cuenta este lenguaje es el propio navegador. [6]

Entre las acciones típicas que se pueden realizar en JavaScript tenemos dos vertientes. Por un lado los efectos especiales sobre páginas web, para crear contenidos dinámicos y elementos de la página que tengan movimiento, cambien de color o cualquier otro dinamismo. Por el otro, JavaScript nos permite ejecutar instrucciones como respuesta a las acciones del usuario, con lo que podemos crear páginas interactivas con programas como calculadoras, agendas, o tablas de cálculo. [6]

JavaScript es un lenguaje con muchas posibilidades, permite la programación de pequeños scripts, pero también de programas más grandes, orientados a objetos, con funciones, estructuras de datos complejas, etc. Además, JavaScript pone a disposición del programador todos los elementos que forman la página web, para que éste pueda acceder a ellos y modificarlos dinámicamente. [6]

HTML.

HTML (HyperText Markup Language). Lenguaje de marcado de Hipertexto. Es el lenguaje estándar para describir el contenido y la apariencia de las páginas en el WWW. [7]

Es el lenguaje más utilizado para la presentación de textos estructurados en formato hipertexto, estándar de las páginas Web. HTML es utilizado por la práctica totalidad de navegadores Web del mercado con el fin de presentar al visitante de una página Web el contenido de la misma tal como el diseñador quiere que se muestre a su público. [8]

CSS.

CSS (Cascading Style Sheets), en español Hojas de estilo en Cascada, fue introducido en 1996, es una tecnología utilizada para definir la presentación de un documento HTML.

Dentro del diseño de páginas de Internet se presenta esta como la vanguardia en cuanto a definición de estilos dentro de las plantillas de diseño. A través de instrucciones en código HTML se definen los estándares del conjunto de páginas que conforman el proyecto. La meta es uniformizar nuestro diseño. [9]

1.5.2 Tecnologías del lado del servidor.

Esta tecnología proporciona un entorno rápido de creación de scripts, integración empresarial, conectividad con clientes y soporte para los estándares más importantes. Además de las aplicaciones tradicionales de bases de datos, las aplicaciones dinámicas de Internet prometen la integración de las comunicaciones bidireccionales y los datos en tiempo real en las aplicaciones, de manera que también necesitan una nueva generación de capacidades de servidores de comunicación.

PHP.

PHP (Preprocessed Hypertext Pages), es un lenguaje de scripting embebido en HTML. Mucha de su sintaxis es tomada de C, Java y Perl con un par de características adicionales únicas y específicas de PHP. [10]

PHP es un lenguaje de programación (originario del nombre PHP Tools, o Personal Home Page Tools) que sirve principalmente para proporcionar características dinámicas a una página Web. Puede combinarse con bases de datos MySQL, ofreciendo resultados muy interesantes para todas aquellas páginas Web que pretendan figurar como activas y dinámicas. [11]

El lenguaje PHP tiene la característica de poder mezclarse con el lenguaje HTML. PHP, al contrario que este último, se interpreta y ejecuta directamente en el servidor en el que está albergada la página Web, con lo que el cliente únicamente recibe el resultado buscado por el código en el que está escrito. [11]

PHP es un lenguaje de propósito general. Debido a su naturaleza open-source, si hay algo que actualmente no se pueda hacer en PHP no hay ningún impedimento en escribir un módulo o una extensión en código C para extender la funcionalidad y así utilizar PHP con el fin deseado.

Ventajas: Muy sencillo de aprender. Similar en sintaxis a C y a PERL. Soporta en cierta medida la orientación a objeto. Clases y herencia. El análisis léxico para recoger las variables que se pasan en la dirección lo hace PHP de forma automática. Librándose el usuario de tener que separar las variables y sus valores. Se puede incrustar código PHP con etiquetas HTML. Excelente soporte de acceso a base de datos. La comprobación de que los parámetros son válidos se hace en el servidor y no en el cliente (como se hace con JavaScript) (…). Además PHP viene equipado con un conjunto de funciones de seguridad que previenen la inserción de órdenes dentro de una solicitud de datos. Se puede hacer de todo lo que se pueda transmitir por vía HTTP. [12]

Desventajas: Todo el trabajo lo realiza el servidor y no delega al cliente. Por tanto puede ser más ineficiente a medida que las solicitudes aumenten de número. La legibilidad del código puede verse afectada al mezclar sentencias HTML y PHP. [12]

Por qué utilizar PHP y no otras opciones: [12]

PHP no soporta directamente punteros, como el C, de forma que no existen los problemas de depuración provocados por estos. Se pueden hacer grandes cosas con pocas líneas de código. Lo que hace que merezca la pena aprenderlo. El código PHP es mucho más legible que el de PERL, todo el que haya programado PERL podrá corroborar esta afirmación. Viene acompañado por una excelente biblioteca de funciones que permite realizar cualquier labor (acceso a base de datos, encriptación, envió de correo, XML, creación de PDF). Al poderse encapsular dentro de código HTML se puede recoger el trabajo del diseñador gráfico e incrustar el código PHP posteriormente. Esta siendo utilizado con éxito en varios millones de sitios Web. Hay multitud de aplicaciones PHP para resolver problemas concretos (weblogs, tiendas virtuales, periódicos) listas para usar. Es multiplataforma, funciona en todas las plataformas que soporten apache. Es software libre. Se puede obtener en la web y su código esta disponible bajo la licencia GPL. [12]

1.5.3 Herramientas de desarrollo.

Para desarrollar este trabajo se utilizan una serie de herramientas entre las que se encuentran:

CodeIgniter

Es un excelente framework para los desarrolladores que usan PHP, construido para codificadores de PHP que necesitan una herramienta simple y elegante para crear sitios web. Pensado para aquellas aplicaciones que se ejecutan en hosting compartido que ejecutan muchas versiones de PHP con diferentes configuraciones. Es muy potente y fácil de usar. A diferencia de otros frameworks cuenta con una buena guía de usuario, instalación sencilla y buenos videos para los principiantes que deseen comenzar a utilizarlo.

CodeIgniter no te obliga a aprender ningún tipo de sintaxis para su configuración, solo se necesita un fichero PHP. Para utilizar CodeIgniter se requiere tener una versión PHP 4.3.2 o alguna más actualizada. Además este framework soporta bases de datos MySQL, Postgre, ODBC, etc. Está diseñado bajo el patrón de diseño de software: Modelo Vista Controlador (MVC), patrón que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. El Modelo: representa la estructura de datos. Típicamente la clase modelo contiene funciones que ayudan a devolver, insertar y actualizar información en su base de datos. La Vista: es la información que se presenta al usuario. Una vista puede ser normalmente una página web, pero en CodeIgniter, una vista puede además ser un fragmento como por ejemplo la cabecera y el pie de página. La Controladora: sirve como un intermediario entre el Modelo, la Vista, y cualquier otro recurso necesitado para procesar la petición HTTP y generar una página web.

Para un mayor entendimiento del patrón Modelo-Vista-Controlador, a continuación se hace una breve descripción del mismo:

El patrón Modelo-Vista-Controlador (MVC)

El patrón de diseño Modelo-Vista-Controlador se utiliza para el diseño de aplicaciones con sofisticadas interfaces.

El patrón Modelo-Vista-Controlador (MVC) descompone una aplicación interactiva en tres bloques:

- Modelo: datos y reglas de negocio.
- Vista: muestra la información del modelo al usuario.

- Controlador: gestiona las entradas del usuario.

El modelo es el responsable de: [13]

- Acceder a la capa de almacenamiento de datos. Lo ideal es que el modelo sea independiente del sistema de almacenamiento.
- Define las reglas de negocio (la funcionalidad del sistema). Un ejemplo de regla puede ser: "Si la mercancía pedida no está en el almacén, consultar el tiempo de entrega estándar del proveedor".
- Lleva un registro de las vistas y controladores del sistema.
- Si estamos ante un modelo activo, notificará a las vistas los cambios que en los datos pueda producir un agente externo (por ejemplo, un fichero que actualiza los datos, un temporizador que desencadena una inserción, etc.).

El controlador es responsable de: [13]

- Recibe los eventos de entrada (un clic, un cambio en un campo de texto, etc.).
- Contiene reglas de gestión de eventos, del tipo "SI Evento Z, entonces Acción W". Estas acciones pueden suponer peticiones al modelo o a las vistas. Una petición al modelo puede ser "Obtener_tiempo_de_entrega (nueva_orden_de_venta)".

Las vistas son responsables de: [13]

- Recibir datos del modelo y lo muestra al usuario.
- Tienen un registro de su controlador asociado (normalmente porque además lo instancia).

Este patrón de diseño implementa la **arquitectura de tres capas**. La ventaja principal es que el desarrollo se puede llevar a cabo en varios niveles y en caso de algún cambio sólo se centra la atención en el nivel requerido, sin tener que revisar entre código mezclado. Además permite distribuir el trabajo de creación de una aplicación por niveles, de este modo, cada grupo de trabajo está totalmente abstraído del resto de los niveles. Añadido a que para una aplicación Web, la arquitectura en tres capas es más eficiente, porque la comunicación entre la interfaz gráfica y el modelo es local, mientras que en la de cuatro capas, la comunicación entre la interfaz gráfica y el modelo es remota aspecto que no es factible para la aplicación

que aquí se propone. Para un mayor entendimiento de esta arquitectura a continuación se explica brevemente las responsabilidades de cada capa:

- Capa de presentación: es la que ve el usuario, presenta el sistema al usuario, le comunica la información y captura la información del usuario dando un mínimo de proceso (realiza un filtrado previo para comprobar que no hay errores de formato). Esta capa se comunica únicamente con la capa de negocio.
- Capa de negocio: es donde residen los programas que se ejecutan, recibiendo las peticiones del usuario y enviando las respuestas tras el proceso. Se denomina capa de negocio (e incluso de lógica del negocio) pues es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de base de datos para almacenar o recuperar datos de él.
- Capa de datos: es donde residen los datos. Está formada por uno o más gestor de bases de datos que realiza todo el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio. Todas estas capas pueden residir en un único ordenador, si bien lo más usual es que haya una multitud de ordenadores donde reside la capa de presentación (los clientes de la arquitectura cliente/servidor). Las capas de negocio y de datos pueden residir en el mismo ordenador, y si el crecimiento de las necesidades lo aconseja se pueden separar en dos o mas ordenadores. Así, si el tamaño o complejidad de la base de datos aumenta, se puede separar en varios ordenadores los cuales recibirán las peticiones del ordenador en que resida la capa de negocio.

1.5.4 Servidor de Base de Datos.

En el desarrollo de Aplicaciones Web se utilizan varios gestores de base de datos, entre los que se destacan PostgreSQL.

PostgreSQL es un sistema de administración de base de datos objeto-relacional (ORDBMS, por sus siglas en inglés) basado en Postgre v4.2 desarrollado en la Universidad de California en el Departamento de Ciencias de la Computación de Berkeley. [15]

PostgreSQL es un descendiente de código abierto de este código original de Berkeley. Soporta SQL92 y SQL99 y ofrece muchas características modernas: Consultas complejas. Foreign keys. Triggers. Vistas. Integridad transaccional. Control de concurrencia multiversión. [15]

A su vez, PostgreSQL puede ser extendido por el usuario en múltiples formas; por ejemplo, agregando nuevos tipos de datos, funciones, operadores, métodos de indexación, funciones de agregación y lenguajes procedurales. [15]

Además, debido a la licencia libre, PostgreSQL puede ser usado, modificado y distribuido libre de cargos para cualquier propósito, sea privado, comercial o académico. [15]

Las políticas de informatización de la Universidad contemplan el uso de PostgreSQL y no otro gestor de bases de datos. La aplicación que existe está desarrollada en SQL Server 2000, servidor que no encuentra dentro de las políticas.

1.6 Servidor Apache.

Hoy día es el servidor Web más utilizado del mundo, encontrándose muy por encima de sus competidores, tanto gratuitos como comerciales. Es un software de código abierto que funciona sobre cualquier plataforma. Por supuesto, se distribuye prácticamente con todas las implementaciones de Linux. [16]

Tiene capacidad para servir páginas tanto de contenido estático, para lo que nos serviría sencillamente un viejo ordenador 486, como de contenido dinámico a través de otras herramientas soportadas que facilitan la actualización de los contenidos mediante bases de datos, ficheros u otras fuentes de información. [16]

Apache está diseñado para ser un servidor Web potente y flexible que pueda funcionar en la más amplia variedad de plataformas y entornos. Las diferentes plataformas y los diferentes entornos, hacen que a menudo sean necesarias diferentes características o funcionalidades, o que una misma característica o funcionalidad sea implementada de diferente manera para obtener una mayor eficiencia. Apache se ha adaptado siempre a una gran variedad de entornos a través de su diseño modular. Este diseño permite a los administradores de sitios Web elegir que características van a ser incluidas en el servidor seleccionando que módulos se van a cargar, ya sea al compilar o al ejecutar el servidor. [17]

Las principales características de Apache son: Funcionalidad en múltiples plataformas. Soporte del último protocolo http 1.1. Sencilla administración basada en la configuración de un único archivo. Soporte para CGI (Common Gatewav Interface) y FastCGI. [18]

Funcionamiento:

Apache extiende este diseño modular hasta las funciones más básicas de un servidor Web. El servidor viene con una serie de Módulos de Multiprocesamiento que son responsables de conectar con los puertos de red de la máquina, aceptar las peticiones, y generar los procesos hijo que se encargan de servirlas. [17]

1.7 Herramienta de desarrollo.

Son muchas las herramientas utilizadas actualmente para desarrollar aplicaciones utilizando el lenguaje PHP. Por ejemplo el DreamWeaver es un editor de páginas Web, para diseñar y codificar sitios, páginas y aplicaciones Web, con este el código HTML es fácil de manejar. También existe el NuSphere, editor de PHP reconocido por su buen completamiento de código. Otro ejemplo es el ZendStudio, actualmente el compilador de PHP más completo, razón que fundamenta su uso en el desarrollo de esta aplicación.

ZendStudio

ZendStudio es una herramienta de desarrollo para diseñadores que incluyen todos los componentes necesarios para el ciclo de vida de una aplicación PHP. Brinda una serie de ayudas: creación y gestión de proyectos, depuración de código, etc. Está implementado en Java, es multiplataforma. Cuenta con varias ventanas, por ejemplo una para visualizar el contenido de las variables, dispone de otras donde muestra la salida del script según se va generando, y otra donde se pueden ver las alertas y errores.

ZendStudio consta de dos partes en las que se dividen las funcionalidades de parte del cliente y las del servidor. Las dos partes se instalan por separado, la del cliente contiene el interfaz de edición y la ayuda. Permite además hacer depuraciones simples de scripts, aunque para disfrutar de toda la potencia de la herramienta de depuración habrá que disponer de la parte del servidor, que instala Apache y el módulo PHP o, en caso de que estén instalados, los configura para trabajar juntos en depuración. [14]

Lo más destacable es que contiene una ayuda contextual con todas las librerías de funciones del lenguaje que asiste en todo momento ofreciendo nombres de las funciones y parámetros que deben recibir. Aunque esta ayuda contextual no solo se queda en las funciones definidas en el lenguaje, sino que también

reporta ayudas con las funciones que vayamos creando nosotros, incluso en páginas que tengamos incluidas con la función include(). [14]

ZendStudio dispone de una herramienta muy interesante de debug o depuración. Gracias a ella se puede ejecutar páginas y conocer en todo momento el contenido de las variables de la aplicación y las variables del entorno como las cookies, las recibidas por formulario o en la sesión. Se pueden colocar puntos de parada de los scripts y realizar las acciones típicas de depuración. [14]

1.8 Metodología de desarrollo de software.

Las metodologías de desarrollo de software son un conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a realizar un nuevo software. Indican paso a paso todas las actividades a realizar para lograr el producto informático deseado, señalan además que personas deben participar en el desarrollo de las actividades y qué papel deben tener. Detallan la información que se debe producir como resultado de una actividad y la información necesaria para comenzarla. No existe una metodología de software universal, las características de cada proyecto (equipo de desarrollo, recursos, etc.) exigen que el proceso sea configurable. El desarrollo de software es riesgoso y difícil de controlar, pero si no se utiliza una metodología, lo que se obtiene es clientes insatisfechos con el resultado y desarrolladores aún más insatisfechos.

Es necesario tomar en cuenta una metodología adecuada para el diseño de una aplicación. Para la selección de la metodología que se adapta más al medio en el que se sitúa este trabajo, se mencionan tres de las más importantes: Rational Unified Process (RUP), Extreme Programing (XP) y Microsoft Solution Framework (MSF).

1.8.1 Rational Unified Process (RUP).

RUP es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado (UML), constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. RUP no es un sistema con pasos firmemente establecidos, sino un conjunto de metodologías adaptables al contexto y necesidades de cada organización. Es un producto de Rational (IBM); creado por Jacobson, Rumbaugh y Booch. Unifica los mejores elementos de metodologías anteriores, está preparado para desarrollar grandes y complejos proyectos. Se caracteriza por ser guiado por los casos de uso, centrado en la arquitectura, iterativo e incremental. Es una forma disciplinada de asignar tareas y responsabilidades en una empresa de desarrollo.

Tiene como objetivo principal asegurar la producción de un software con calidad, dentro de plazos y presupuestos predecibles. Aumenta la productividad de los desarrolladores mediante acceso a: base de conocimiento, plantillas y herramientas. Se centra en la producción y mantenimiento de modelos del sistema más que en producir documentos. RUP es una guía de cómo usar UML de la forma más efectiva. Existen herramientas de apoyo a todo el proceso: modelación visual, programación, pruebas, etc. Esta metodología ayuda a planificar, diseñar, implementar, ejecutar y evaluar pruebas que verifiquen estas cualidades.

RUP pretende implementar las mejores prácticas actuales en ingeniería de software: Desarrollo iterativo del software. Administración de requerimientos. Uso de arquitecturas basadas en componentes. Modelación visual del software. Verificación de la calidad del software. Control de cambios

RUP divide en 4 fases el desarrollo del software: [19]

- Inicio: El Objetivo en esta etapa es determinar la visión del proyecto.
- Elaboración: En esta etapa el objetivo es determinar la arquitectura óptima.
- Construcción: En esta etapa el objetivo es llevar a obtener la capacidad operacional inicial.
- Transmisión: El objetivo es llegar a obtener el realce del proyecto.

Cada una de estas etapas es desarrollada mediante el ciclo de iteraciones, la cual consiste en reproducir el ciclo de vida en cascada a menor escala. Los Objetivos de una iteración se establecen en función de la evaluación de las iteraciones precedentes. [19]

Divide el flujo de trabajo en: Modelación del Negocio, Análisis y Diseño. Implementación, Prueba (Testeo), Instalación, Administración del proyecto, Administración de configuración y cambios, Ambiente.

Es recomendable que a cada una de estas iteraciones se les clasifique y ordene según su prioridad, y que cada una se convierte luego en un entregable al cliente. Esto trae como beneficio la retroalimentación que se tendría en cada entregable o en cada iteración. [19]

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. [19]

1.8.2 Extreme Programing (XP).

Es una de las metodologías de desarrollo de software más exitosas en la actualidad utilizada para proyectos de corto plazo, pequeño equipo de trabajo y cuyo plazo de entrega era ayer. La metodología consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar al éxito del proyecto. [19]

Características de XP, la metodología se basa en: [19]

- **Pruebas Unitarias:** se basa en las pruebas realizadas a los principales procesos, de tal manera que adelantándonos en algo hacia el futuro, podamos hacer pruebas de las fallas que pudieran ocurrir. Es como si nos adelantáramos a obtener los posibles errores.
- **Refabricación:** se basa en la reutilización de código, para lo cual se crean patrones o modelos estándares, siendo más flexible al cambio.
- **Programación en pares:** una particularidad de esta metodología es que propone la programación en pares, la cual consiste en que dos desarrolladores participen en un proyecto en una misma estación de trabajo. Cada miembro lleva a cabo la acción que el otro no está haciendo en ese momento. Es como el chofer y el copiloto: mientras uno conduce, el otro consulta el mapa.

¿Qué es lo que propone XP? [19]

- Empieza en pequeño y añade funcionalidad con retroalimentación continua.
- El manejo del cambio se convierte en parte sustantiva del proceso.
- El costo del cambio no depende de la fase o etapa.
- No introduce funcionalidades antes que sean necesarias.
- El cliente o el usuario se convierte en miembro del equipo.

Lo fundamental en este tipo de metodología es: [19]

- La comunicación, entre los usuarios y los desarrolladores.
- La simplicidad, al desarrollar y codificar los módulos del sistema.
- La retroalimentación, concreta y frecuente del equipo de desarrollo, el cliente y los usuarios finales.

1.8.3 Microsoft Solution Framework (MSF).

Esta es una metodología flexible e interrelacionada con una serie de conceptos, modelos y prácticas de uso, que controlan la planificación, el desarrollo y la gestión de proyectos tecnológicos. MSF se centra en los modelos de proceso y de equipo dejando en un segundo plano las elecciones tecnológicas. [19]

MSF tiene las siguientes características: [19]

- Adaptable: es parecido a un compás, usado en cualquier parte como un mapa, del cual su uso es limitado a un específico lugar.
- Escalable: puede organizar equipos tan pequeños entre 3 ó 4 personas, así como también, proyectos que requieren 50 personas a más.
- Flexible: es utilizada en el ambiente de desarrollo de cualquier cliente.
- Tecnología Agnóstica: porque puede ser usada para desarrollar soluciones basadas sobre cualquier tecnología.

MSF se compone de varios modelos encargados de planificar las diferentes partes implicadas en el desarrollo de un proyecto: Modelo de Arquitectura del Proyecto, Modelo de Equipo, Modelo de Proceso, Modelo de Gestión del Riesgo, Modelo de Diseño de Proceso y finalmente el modelo de Aplicación. [19]

Tras un estudio de estas metodologías de desarrollo se decide utilizar Rational Unified Process (RUP), su elección se puede fundamentar debido a que RUP es más adaptable para proyectos de largo plazo, definiendo las tareas y los roles desempeñado por cada uno de los integrantes del equipo de desarrollo. Extreme Programing (XP) la cual se recomienda para proyectos de corto plazo, presenta falta de documentación. La Metodología MSF por su parte se adapta a proyectos de cualquier dimensión y de cualquier tecnología. En este trabajo se necesita contar con una documentación explícita del proceso de trabajo, que ayude a formar el documento final.

1.9 Herramienta Case.

Las Herramientas CASE (Computer Aided Software Engineering, Ingeniería de Software Asistida por Ordenador), son aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero. Estas herramientas ayudan al desarrollo del software, en tareas como realizar un diseño del proyecto, implementación de parte del código automáticamente con el diseño dado, documentación o detección de errores, entre otras.

Tienen como objetivos: mejorar la productividad en el desarrollo y mantenimiento del software, aumentando con esto la calidad del mismo. Mejorar la planificación de un proyecto, el tiempo, coste de desarrollo y mantenimiento de los sistemas informáticos. Además garantizar la gestión global en todas las fases de desarrollo de software con una misma herramienta. Facilitar el uso de las distintas metodologías propias de la ingeniería del software.

El Rational Rose y el Visual Paradigm son herramientas CASE, que se utilizan en el desarrollo de un software. Ambas usan RUP como metodología de desarrollo y UML como lenguaje para la creación de diagramas necesarios y generar parte del código del producto según las clases definidas en el Diseño. El Visual Paradigm tiene implementado UML 2 propiedad que lo pone en ventaja frete al Rational. Este último tiene la desventaja de que hay lenguajes de programación para los que no genera código o lo hace a medias; no sucediendo así con el Visual Paradigm, el cual realiza este trabajo con más leguajes de programación, y de forma más completa. Además este último permite importar diagramas hechos en el Rational Rose y trabajar con ellos, característica que lo favorece y lo hace el preferido de miles de usuarios en el mundo. Otra de las desventajas del Rational es que una vez que se tiene el diagrama de clases persistentes a partir del cual se genera la base de datos del sistema, no existe la posibilidad de exportar ese modelo hacia algún sistema gestor de bases de datos, cosa que si se puede hacer con el Visual Paradigm y que lo define como el ideal para realizar este trabajo.

1.10 Lenguaje de modelación.

El lenguaje de modelado de objetos es un conjunto estandarizado de símbolos y del modo de disponerlos para modelar parte de un diseño de software orientado a objetos, cuyo mejor ejemplo es Unified Modeling Languaje (UML).

Unified Modeling Languaje (UML).

UML es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de software. [20]

Está compuesto por diversos elementos gráficos que se combinan para conformar diagramas. UML cuenta con reglas para combinar tales elementos. Es un lenguaje que permite la modelación de sistemas con tecnología orientada a objetos.

Ofrece un estándar para describir el sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables.

Entre sus características más generales se encuentran:

- Corrección de errores
- Tecnología orientada a objetos.
- Desarrollo iterativo e incremental.
- Participación del cliente en todas las etapas del proyecto.

1.11 Conclusiones

Partiendo de la necesidad de creación de un nuevo sistema de Pre-Ingreso a la UCI que cumpla con los cambios que se han originado en el proceso de Ingreso en general, y previendo futuras transformaciones, en este capítulo se fundamenta teóricamente el porqué de crear un nuevo sistema que cumpla con los nuevos requisitos y de cabida a futuras solicitudes por parte de los clientes. Para ello se define:

Utilizar como metodología de desarrollo de software Rational Unified Process (RUP) y como lenguaje de modelado (Unified Modeling Languaje (UML)). Realizar la aplicación en: PHP, PostgreSQL, se utiliza un servidor web apache, CodeIgniter como framework. Usar Visual Paradigm versión 2.3 como herramienta para el desarrollo de los distintos artefactos ingenieriles.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA.

El capítulo aborda el objeto de estudio, problema y situación problémica, se detalla el objeto de automatización, la información que se maneja, se realiza una descripción general de la propuesta de sistema. Además se enumeran los requisitos funcionales y no funcionales con los que debe contar el software. Se definen las reglas del negocio e ilustran los diferentes modelos: Modelo del Dominio y Modelo de Casos de Uso del Sistema, el cual permite tener una noción general de los procesos que responden a las funcionalidades definidas en los requerimientos funcionales e identificar las relaciones entre los actores que interactúen con el sistema y las diferentes actividades con las que se relacionan.

2.1 Problema y Situación problémica.

El sistema existente no cuenta con una documentación completa, por ejemplo la ayuda, que familiariza al usuario con el sistema. Está diseñado para un examen determinado, esto trae consigo que en el momento que se hace necesario realizar un cambio en los test (preguntas), en cuanto al valor máximo y la pregunta de aciertos, se necesite hacer varios cambios en el código fuente hasta el punto de hacerlo casi todo de nuevo. Este sistema carece de diseño grafico (todo está en blanco). Además esta desarrollado en Perl, lenguaje que no está en correspondencia con las políticas de informatización del centro. Estas problemáticas llevan a buscar como solución la realización de un nuevo sistema capaz de aceptar los cambios que vayan surgiendo en el sistema de Ingreso a la Universidad.

2.2 Objeto de automatización.

Con la realización del sistema se pretende automatizar lo siguientes procesos.

2.2.1 Gestión de los test.

Los directivos de la Universidad necesitan contar con la posibilidad de crear los test según las habilidades que se deseen evaluar en el test de ingreso. Por ejemplo cantidad de preguntas, y las características de cada una (valor, peso, tipo de pregunta).

2.2.2 Gestión de códigos de anonimato.

Se necesita asignarle a cada estudiante que se presenta al test un código, para ello se requiere que los mismos sean generados, de manera tal que dos estudiantes de un proceso (IPI, CPT, PRE) en un curso y un test dado no tengan el mismo código. Esto ayuda a diferenciar a dichos estudiantes y a no cometer errores a la hora de calificar los test.

2.2.3 La realización de los reportes.

Los administradores necesitan conocer los datos que se obtienen en la gestión de calificaciones de los test: reporte de notas y de calificaciones por procesos (IPI, CPT, PRE) y por curso test, los códigos de los estudiantes que se presentaron al test y que no tienen nota. Además de un control de todas las acciones que realizan los usuarios en el sistema. Por lo que se necesita para la realización de los reportes un buscador que actué con dinamismo en dependencia de las solicitudes que le realicen.

Con el reporte de calificaciones los directivos de la Universidad pueden ver los resultados que obtuvieron los estudiantes cuando fueron calificados sus test por los profesores. Para el reporte de notas se utiliza un algoritmo de ponderación el cual calcula la nota real que va a tener el estudiante en cada test y su promedio total. La implementación de este algoritmo se define como un proceso a automatizar.

2.2.4 Implementación del algoritmo.

Para obtener la nota real que un estudiante obtuvo en el test se utiliza un algoritmo de promedio ponderado.

Se llevan las notas sobre la base de 100 puntos y se halla el promedio, se busca el máximo y el mínimo de cada test, se analizan todos los resultados obtenidos en el test, donde el mínimo sería 0 y el máximo 100, se redondea la nota de cada test a dos cifras decimales, a continuación se representa la fórmula matemática en la que se basa este algoritmo:

$$PP = \frac{\sum_{i=0}^{i=n} P_i * V_i}{\sum_{i=0}^{i=n} P_i}$$

Se calcula el promedio ponderado (PP), donde cada test tiene su peso (P), para esto se realiza una sumatoria de la multiplicación de los pesos (P) de cada pregunta por el valor (V) de la nota obtenida en la misma, este resultado se divide entre la sumatoria de los pesos (P) que se les fueron asignados a las preguntas del test, el resultado de esta operación se redondea a dos cifras decimales.

El test puede estar compuesto por aciertos, errores, omisiones y vistas, en este caso el algoritmo se comporta de la siguiente forma: Se le da un valor muy pequeño al test, por ejemplo (-1000).

El valor del test va a ser igual al valor de las vistas por (el valor de los aciertos menos el valor de los errores entre el valor de los aciertos más el valor de las omisiones), en caso de que el valor de los aciertos más el de las omisiones sea mayor que cero se realiza la operación, en caso contrario se deja con el valor

muy pequeño, ya que la operación de división no podría realizarse (división por cero). El valor del test se redondea a dos cifras después de la coma.

2.3 Información que se maneja.

En el sistema que se propone se maneja toda la información referente a la calificación de los test de ingreso a la UCI. Los estudiantes que cuentan con un aval político aprobado, se les da la posibilidad de presentarse a realizar el test de ingreso a la Universidad. El término test, en este caso, se considera como un examen, el cual cuenta con varios test en su interior, donde cada uno tiene definido un peso de acuerdo a su valor de ponderación, el cual fue aprobado previamente por el personal capacitado para ello.

La información procesada referente a los test de ingreso a la UCI constituye el principal argumento en la decisión de si el estudiante puede o no ingresar a la Universidad. Las calificaciones que obtuve el aspirante en cada uno de los test es utilizada para calcular la nota real que obtuvo en el examen en general.

En la aplicación se maneja toda la información referente a los test: creación de los mismos con su número y el curso en el que se va a aplicar. Además la cantidad de preguntas que va a tener el test, tipo de pregunta, valor y peso que se le asigna a cada una. También se manipulan los códigos de anonimato los cuales son generados a partir de las solicitudes que realice el cliente.

2.4 Propuesta de sistema.

Se desarrollará una aplicación Web utilizando PHP y PostgreSQL, como servidor Web Apache y CodeIgniter como framework. Estas tecnologías ayudan al sistema a actuar de forma dinámica, el correcto funcionamiento de la aplicación requiere funcionalidades que las estáticas no permiten, el dinamismo en los sitios Web da la posibilidad de extraer la información a mostrar desde una base de datos como Postgre.

Las aplicaciones Web son utilizadas con el objetivo de acceder a un servidor Web a través de Internet o de una intranet. Desarrollar un sistema Web aporta grandes ventajas. Por su facilidad de administración centralizada, son ideales para su despliegue en Internet como en intranets corporativas. Reduce el tiempo de aprendizaje gracias a su facilidad de uso y el hecho de que cada día aumenta más el número de usuarios navegando por Internet y utilizando este medio para ampliar sus conocimientos. Las aplicaciones Web que soporten las características de los browsers estándar funcionan igual independientemente de la versión del sistema operativo instalado en el cliente. En vez de crear clientes para Windows, Mac OS X, GNU/Linux, y otros sistemas operativos, la aplicación es escrita una vez y es mostrada casi en todos lados.

Por la importancia de la información que se maneja en el sistema se requiere una gran seguridad, que se garantizan con el uso del CodeIgniter como framework, este implementa el patrón de diseño modelo vista controlador, el cual separa la apariencia, los datos y la lógica de la programación en capas diferentes.

En el desarrollo del nuevo sistema de ingreso a la UCI, se emplea una arquitectura de tres capas la cual fue explicada en el capítulo anterior. El uso de tecnologías de software libre le da al sistema la característica de mostrarse y funcionar de la misma forma en el cliente, independientemente del sistema operativo que este instalado en el mismo, además cumplirá con las políticas de informatización de la UCI.

2.4.1 Reglas del negocio.

- Para registrar las calificaciones los códigos deben existir.
- Determinar correctamente los códigos y las calificaciones.
- Ninguna persona esta autorizada a ver los nombres de los estudiantes.
- Debe existir un responsable al frente de la actividad calificar.

2.5 Descripción de los Procesos del Negocio.

Los procesos de negocio son funciones que se desarrollan en el ambiente o entorno que se define como negocio. En el presente trabajo se definen tres procesos del negocio:

Gestión de test: incluye la creación de los test definiendo: número y curso en el que se va a aplicar, cantidad de preguntas, cada una con su tipo, valor y peso. Este proceso lo realiza el personal de secretaría general, que a la vez son los que se benefician con la realización del mismo.

Gestión de códigos de anonimato: este proceso da la posibilidad de tratar al estudiante a lo largo del negocio como un código de anonimato, de manera tal que el nombre del estudiante no sea de conocimiento público. El personal de secretaría es quien se beneficia de este proceso y a su vez es quien lo realiza.

Gestión de reportes: Se necesita tener un control de las acciones realizadas por los usuarios calificadores, y el personal de secretaría, de las calificaciones y de las notas que obtuvieron los estudiantes en el examen. El personal de secretaría es el que se encarga de realizar los reportes y a la vez es quien se beneficia de los mismos.

Calificación de test: Para la calificación de los test los profesores deben calificar cada una de las preguntas que conforman el test, con estos resultados se calcula posteriormente la nota real que va a

tener el estudiante en el test. El personal de secretaría es el que se encarga de realizar los reportes y a la vez es quien se beneficia de los mismos.

2.6 Modelo del Dominio.

En la sección anterior se definieron los procesos del negocio, en la misma se evidencia un solapamiento de responsabilidades, ya que el personal de secretaría puede actuar como trabajador a la vez de ser quien se beneficie con la realización del proceso de negocio. Además existen múltiples responsabilidades ejemplo de esto lo podemos ver nuevamente en el personal de secretaría general el cual tiene varias responsabilidades en el negocio. Difícil establecimiento de reglas de funcionamiento.

Basándose en los criterios anteriores, se decide desarrollar un modelo de dominio. El mismo captura los tipos más importantes de objetos que existen o los eventos que suceden en el entorno donde estará el sistema, es decir permite de manera visual mostrar al usuario los principales conceptos que se manejan en el dominio del sistema en desarrollo.

Para ellos se hace necesario definir los conceptos más importantes que se manejan con el objetivo de ayudar a los usuarios, clientes, desarrolladores e interesados, a utilizar un vocabulario común para poder entender el contexto en que se coloca el sistema. Estos conceptos facilitan la capturar de los requisitos y dan la posibilidad de tener un buen conocimiento del funcionamiento del objeto de estudio y con esto construir el sistema de acuerdo a las necesidades de los clientes.

2.6.1 Conceptos que se utilizan en el modelo de dominio.

Para la realización del Modelo del Dominio se necesita identificar primeramente los conceptos que se utilizarán en el diagrama, para ello se utiliza un glosario de términos con los nombres de los conceptos o clases más importantes:

- Test: Examen de ingreso a la UCI.
- Pregunta: Preguntas que componen el test de ingreso.
- Estudiante: Persona que responde el test de ingreso a la Universidad.
- Usuario_Calificador: Persona con permiso de calificar los test de ingreso.
- Personal_Secretaría: Persona con permisos de calificar, crear y editar el examen de ingreso y las preguntas, además de definir el usuario calificador y otorgar el código de anonimato a los estudiantes.

- Calificación: Se considera calificación, a los resultados que obtiene el estudiante en el test de ingreso.
- Código: Secuencia de caracteres que se le asignan al test y conforman el código de anonimato del estudiante.
- Valor: Se considera valor a la calificación máxima que se le asigna a la pregunta.
- Peso: Es el valor de ponderación que se le da a la pregunta.

A continuación se representa el diagrama de clases del modelo de dominio.

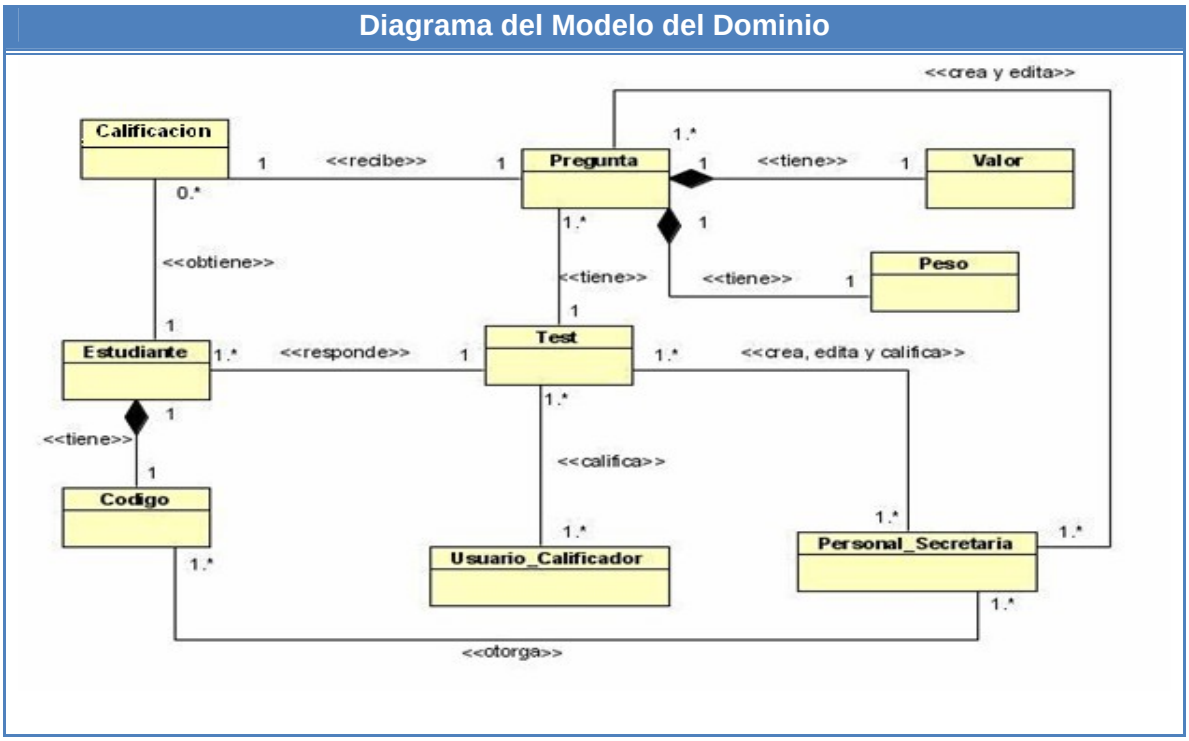

Figura 1 Diagrama de clases del Modelo del Dominio.

2.7 Especificación de los requisitos de software.

Los requerimientos de un software son las condiciones o capacidades que tiene que tener un sistema para satisfacer un contrato o documento formal.

2.7.1 Requisitos funcionales.

Los Requerimientos funcionales definen acciones que el sistema debe ser capaz de realizar, sin tomar en consideración ningún tipo de restricción física. Teniendo en cuenta los objetivos de los futuros usuarios del sistema y la descripción de cómo debe funcionar el mismo, se pueden deducir los requisitos funcionales siguientes:

El sistema debe ser capaz de:

R1 Autenticarse en el sistema.

- Se introduce el usuario y la contraseña.
- Verificar si el usuario pertenece al ldap.
- Verificar si existe en el sistema.

R2 Asignar rol.

- Se especifica login, rol y curso-test.

R3 Agregar rol.

- Se especifica el nuevo rol.

R4 Buscar rol de un usuario.

- Se especifica usuario y se muestra el rol.

R5 Modificar usuario.

- Se especifica el usuario y se modifica el rol.

R6 Eliminar usuario.

- Se especifica el usuario y se elimina.

R7 Mostrar todos los usuario.

R8 Buscar todos los usuarios de un rol.

- Se especifica el rol y se muestran todos los usuarios que tienen ese tipo de rol.

R9 Generar código.

- Se especifica el tipo de estudiante, el curso-test, y la cantidad de Códigos que desea generar.

R10 Insertar código.

- Se especifica el código, el tipo de estudiante (IPI, CPT, PRE) y el curso-test.

R11 Listar códigos por proceso.

- Se especifica el tipo de estudiante y el curso-test, se muestran los códigos.

R12 Controlar Log de actividades.

- Visualizar Log por usuario, fecha, url, acción, dirección ip y hora.
- Se especifica el usuario y se muestran los logs realizados.
- Se especifica la fecha y se muestran los usuarios con (fecha, url, acción, dirección ip, hora) que han realizado acción en el sistema en ese día.

R13 Insertar calificación.

- Se especifica el curso-test, proceso (IPI, CPT y PRE) y código al que pertenece el estudiante. Se insertan las calificaciones en cada uno de los test.

R14 Modificar una calificación de un estudiante dado.

- Se indica el curso-test, proceso (IPI, CPT y PRE), código del estudiante al que pertenece, y se actualiza las calificaciones.

R15 Mostrar Código sin calificaciones.

- Se selecciona el tipo de estudiante (IPI, CPT, PRE) y el curso-test, y se muestran los códigos sin calificaciones.

R16 Generar reporte por tipo de estudiante con notas y promedio.

- Se especifica el tipo de estudiante, el tipo de reporte notas y el curso-test, y se muestran las notas de cada código con sus promedios.

R17 Generar reporte de calificaciones

- Se especifica el tipo de estudiante, el tipo de reporte calificaciones y el curso-test, y se muestran las calificaciones de cada código.

R18 Crear test.

- Se Inserta el número, curso, cantidad de preguntas normales y cantidad de preguntas especificaciones.
- Se especifica para cada pregunta valor y peso.

R19 Crear curso.

- Se especifica el nuevo curso.

R20 Listar todos los test con sus atributos.

- Se muestra el número del test, curso, cantidad de preguntas y los atributos de cada una: valor, peso, tipo (si es normal o de especificaciones).

2.7.2 Requisitos no funcionales.

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener. Pueden ser determinados como los primeros requisitos no funcionales del sistema los siguientes:

Apariencia e interfaz externa.

- La interfaz del producto será sencilla y fácil de entender, para evitar que se desvié el objetivo central del sistema.
- La interfaz debe ser adaptable a los patrones de diseño de la UCI.
- El sistema guiará las operaciones a seguir en cada momento, además debe permitir que se realicen las operaciones que sean válidas, cuando la información que se halla entrado sea la correcta.
- Permitirá a los usuarios realizar las operaciones disponibles de forma amena y sin complicaciones.

Usabilidad.

- El sistema debe contar con una ayuda para que cada usuario realice su trabajo sin mayores complicaciones.

Rendimiento.

- El sistema funcionará en tiempo real, sus respuestas serán inmediatas.
- La eficiencia del sistema va a estar dada por la velocidad de procesamiento, disponibilidad y tiempo de respuesta.

Soporte.

- El sistema funcionará sobre una aplicación desarrollada utilizando software libre, y tecnología cliente servidor.
- El sistema permitirá modificar en función de mejorar los módulos ya existentes, por el personal autorizado.

Portabilidad.

- En las terminales clientes se garantizará sistema operativo Linux desde KDE 3.1 o Windows desde versión 98.
- El sistema será implementado en PHP, como gestor de base de dato PostgreSQL, para garantizar el cumplimiento de las políticas de informatización del centro.

Seguridad.

- La información manejada por el sistema esta protegida de acceso no autorizado y divulgación, la misma es almacenada para no perderla de un curso para otro
- El acceso al sistema será mediante la autenticación de usuarios, permitiendo acceder solamente al personal autorizado a utilizar el mismo.
- El sistema debe contar con un módulo de administración para garantizar mayor seguridad, y tener definido los diferentes niveles de acceso de los usuarios a la información.
- El sistema debe mostrar sus diferentes utilidades en dependencia del tipo de usuario que este activo.
- La información manejada por el sistema será protegida contra la corrupción y estados inconsistentes.
- A los usuarios autorizados se les garantiza el acceso a la información, los dispositivos o mecanismos utilizados para lograr la seguridad no retrasaran los datos deseados a los usuarios.
- El administrador de red debe realizar una salva de la base de datos al concluir el curso académico.

Confiabilidad.

- El sistema estará disponible en todo momento permitiendo el trabajo de los usuarios y las acciones de mantenimiento.
- El sistema debe ser capaz de recuperarse rápidamente de los fallos ocurridos.

Funcionalidad:

- Reducir el tiempo en que carga cada una de las páginas del sistema.

Software.

- En la PC cliente se requiere navegador compatible o superior con Internet Explorer 4, o compatible.

- Para el servidor PostgreSQL8, Apache 2.5.7 y PHP 4

Hardware.

- Se requiere tanto para la PC cliente como para el servidor mínimo una PC con procesador Pentium IV o compatible. Se necesita además en el cliente una tarjeta de red de 10/100 MB/s para conexión con el servidor.
- Para el servidor como mínimo se necesita 256 MB de memoria RAM, 60 GB de espacio libre en disco duro para almacenamiento, tarjeta de red de 10/100 MB/s para la conexión.
- Se requiere de una impresora en las terminales clientes.

Ayuda y documentación en línea.

- El sistema debe contar con una ayuda o manual de usuario que le facilite al mismo la interacción con el sistema.

2.8 Descripción del sistema propuesto.

El sistema que se propone para dar cumplimiento a los objetivos planteados al inicio del trabajo, cuenta con cinco módulos principales, autenticación de usuario, introducción de la información, administración del sistema, reportes de salida de la información procesada y gestión de test de ingreso, para su realización se tendrá en cuanta los requerimientos que fueron definidos a partir de las necesidades del cliente. Existen dos roles: el usuario que califica los test y el administrador del sistema.

El usuario que califica los test, sólo tiene permiso de autenticarse y de introducir las calificaciones en el sistema. El administrador del sistema se encarga de la gestión de los usuarios, gestión de los códigos, gestión de los test; puede además insertar y modificar calificaciones en el sistema, es quien solicita los reportes que se necesiten, y controla los Logs de actividades.

El módulo de autenticación: es usado por los usuarios que desean acceder al sistema, tiene el objetivo de comprobar sus permisos y lograr con esto mayor seguridad.

El módulo de introducción de la información: podrá ser usado por el usuario que califica los test, siempre y cuando esté autorizado por un administrador para entrar al sistema; el administrador también puede acceder a este módulo.

El módulo de administración del sistema: será usado por los administradores, para garantizar la seguridad requerida. Este módulo tiene acceso restringido, es donde se les asignan los permisos a los usuarios, se

crean nuevos roles y se modifica el rol de un usuario ya existente en el sistema. Además se administra todo lo referente a los códigos de anonimato: listar, insertar y generar códigos.

Al módulo de reportes de salida de la información procesada: sólo van a tener acceso los administradores del sistema, este módulo da la posibilidad al usuario de generar los reportes de acuerdo a sus necesidades.

Al módulo de gestión de test de ingreso: los únicos que tienen permiso son los administradores, los cuales pueden crear, modificar y eliminar un test dado.

El sistema que se propone, pretende crear un paquete de software Pre-Ingreso UCI, con el desarrollo de sus cinco módulos y la documentación requerida: manual de usuario y documentación técnica. Manual de instalación y configuración

2.9 Definición de los actores del sistema.

Los actores representan usuarios y otros sistemas que interactúan con el sistema, en esta sección se definen los actores del sistema propuesto.

Actores del Sistema	Justificación
Calificador CPT	Tiene permiso de autenticarse e introducir las calificaciones de los estudiantes de CPT en el sistema.
Calificador IPI	Tiene permiso de autenticarse e introducir las calificaciones de los estudiantes de IPI en el sistema.
Calificador pre	Tiene permiso de autenticarse e introducir las calificaciones de los estudiantes de Pre en el sistema.
Administrador	Se encarga de la gestión de: usuarios, códigos, test, nota real de los estudiantes en el examen; puede además insertar calificaciones en el sistema, es quien obtiene la información de los reportes, y además de los de los Log de actividades.
Usuario_Calificador	Este usuario es una generalización de los actores: Calificador cpt, Calificador IPI, Calificador pre, Administrador. Tiene como tarea introducir las calificaciones en el sistema.
Usuario_Sistema	Este usuario es una generalización de los actores que intervienen en el sistema, es quien se autentica en el sistema.

Tabla 1 Actores del Sistema.

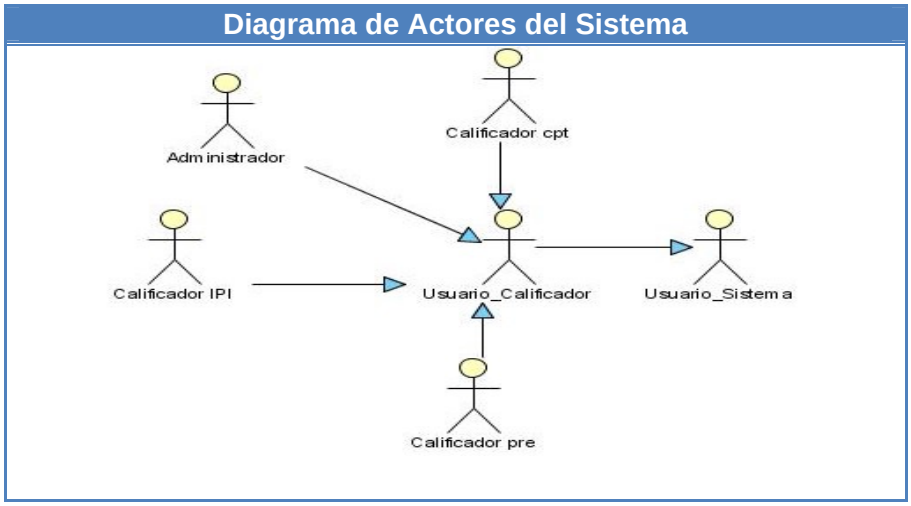

Figura 2 Diagrama de la relación de actores del sistema.

Listado de casos de uso del sistema

CU-1	Autenticarse
Actor	Usuario_Sistema
Descripción	Este caso de uso se inicia cuando un usuario decide entrar al sistema, para esto introduce su usuario y contraseña, el sistema verifica que el usuario pertenezca a Ldap, si pertenece, verifica con la información que hay en la base de datos del sistema. Luego le permite el acceso al sistema al usuario autorizado.
Referencia	R1

CU-2	Insertar_Calificacion
Actor	Usuario _ Calificador
Descripción	Este caso de uso se inicia cuando el Usuario_Calificador (generalización de actores: administrador, calificador IPI, calificador CPT, calificador PRE) entra al sistema. Si es un calificador de (IPI, CPT, PRE), la aplicación le muestra la interfaz que corresponde al test específico que él tiene asignado para calificar. Este especifica el código e inserta las calificaciones que obtuvo el estudiante en los test. Si el rol es de administrador el sistema muestra una interfaz para que este seleccione el curso-test. Una ves realizada esta acción se muestra habilitada una nueva interfaz con el test que el seleccionó y un menú para que seleccione el proceso, e introduzca el código y las calificaciones del

	estudiante.
Referencia	R13

CU-3	Modificar_Calificacion
Actor	Administrador
Descripción	Este caso de uso inicia cuando el administrador entra a la sección "Modificar Calificación", aquí selecciona el curso y el número del test al que pertenece el estudiante que se le desea modificar las calificaciones, luego introduce el proceso al que pertenece este estudiante y el código. El sistema muestra las calificaciones de este código. El administrador hace los cambios que desea; al aceptar, estos se guardan automáticamente en la base de datos. Puede realizar la misma acción si lo desea es ver las calificaciones de un código determinado, y no realizar cambios. Además si una vez mostrada las calificaciones cambia el código y da actualizar se hace una reasignación de calificaciones de un código a otro.
Referencia	R14

CU-4	Agregar_Rol
Actor	Administrador
Descripción	Este caso de uso inicia cuando el administrador entra a la sección "Agregar Rol", el sistema le permite entrar el rol, y cuando acepta se guarda en la base de datos el nuevo rol.
Referencia	R3

CU-5	Asignar_Rol
Actor	Administrador
Descripción	Este caso de uso inicia cuando el administrador entra a la sección "Asignar Rol", donde introduce el login, el rol y el curso-test del nuevo usuario, el sistema verifica que pertenezca a Ldap y que no esté en la base de datos del sistema, de ser así

	almacena la información en la base de datos, sino muestra mensajes.
Referencia	R2,

CU-6	Gestionar _ Usuario
Actor	Administrador
Descripción	Este caso de uso inicia cuando el administrador entra a la sección “Listar Usuarios”, el sistema muestra la lista de los usuarios y sus roles con las opciones: “Eliminar Usuario” y “Editar Usuario”. Si el administrador selecciona la opción editar el sistema muestra la interfaz para esta acción, con el usuario que el seleccionó y un menú con los roles que este puede tener, él define cual le reasignará y acepta, el sistema guarda automáticamente este cambio en la base de datos. Si lo que desea es eliminar un usuario, el sistema le verifica si está seguro de realizar la acción y automáticamente se guarda este cambio.
Referencia	R4, R5, R6, R7, R8

CU-7	Buscar_Usuario
Actor	Administrador
Descripción	Este caso de uso inicia cuando el administrador entra la sección “Buscar Usuarios” y el sistema le muestra una interfaz donde selecciona lo que desea buscar: Usuario o Rol; si selecciona la primera, introduce el login del usuario o parte de este, el sistema le muestra el usuario y su rol, si introdujo parte del login el sistema muestra todos los login con sus respectivos roles, que contienen la parte especificada. Si selecciona la segunda, especifica el rol, y el sistema muestra los usuarios que tienen ese rol.
Referencia	R4, R8

CU-8	Generar_Codigo
Actor	Administrador
Descripción	Este caso de uso inicia cuando el administrador entra al sistema y elige la opción “Generar Código”. El sistema muestra una interfaz con dos menú donde aparecen los

	tipos de estudiantes y curso-test que existen, y da la opción de especificar la cantidad de códigos que desea generar. El administrador especifica todos los datos y el sistema ejecuta un algoritmo para generar los códigos de anonimato, automáticamente se cargan en la base de datos la cantidad de códigos que él definió para el tipo de estudiante y el curso-test que seleccionó.
Referencia	R9

CU-9	Insertar_Codigo
Actor	Administrador
Descripción	Este caso de uso inicia cuando el administrador entra al sistema y elige la opción "Insertar Nuevo Código" el sistema le muestra una interfaz lista para realizar esta acción. El administrador elige el curso, el proceso en el que desea insertar el código y luego introduce este. Cuando marca la opción guardar, el sistema verifica que el código con ese curso-test y ese proceso no esté en la base de datos, sino está inserta el nuevo código y se muestra un mensaje "Se ha insertado el código del estudiante", si está muestra mensaje "El estudiante no pudo ser insertado, el código ya existe".
Referencia	R10

CU-10	Listar_Codigo
Actor	Administrador
Descripción	Este caso de uso inicia cuando el administrador entra al sistema y selecciona la opción "Listar Códigos", Para ello el sistema le muestra una interfaz que le permite seleccionar el Tipo de estudiante (proceso al que pertenece) y el curso-test. Cuando pulsa el botón "Buscar", el sistema le muestra una lista con todos los códigos que pertenecen a dicho proceso y curso-test, en caso de no existir códigos muestra un mensaje "No existen códigos"
Referencia	R11

CU-11	Confeccionar_Reporte
Actor	Administrador
Descripción	Este caso de uso inicia cuando el administrador selecciona la opción "Confeccionar

	reporte", el sistema le muestra una interfaz en la que puede seleccionar el tipo de reporte deseado, de notas o calificaciones. Luego define el tipo de estudiante y el curso test para saber el reporte. Al aceptar el sistema busca la información en la base de datos y muestra el reporte, en caso de no existir la información, muestra un mensaje: "No existen datos para este reporte".
Referencia	R16, R17

CU-12	Buscar_Codigo_Sin_Nota
Actor	Administrador
Descripción	Este caso de uso inicia cuando el administrador selecciona la opción "Código sin nota", El sistema le muestra una interfaz que le permite seleccionar el Tipo de estudiante (Proceso), y el Curso-Test. Al dar en el botón "Buscar" el sistema le muestra una lista con todos los códigos de los estudiantes que no tienen nota, que corresponden a los datos seleccionados.
Referencia	R15

CU-13	Controlar_Log
Actor	Administrador
Descripción	Este caso de uso inicia cuando el administrador entra a la parte del sistema que le permite controlar los "Log de actividades". El sistema le muestra una interfaz habilitada para confeccionar el log de actividades. En la misma aparece un menú con todos los usuarios que han realizado acciones en el sistema y uno con todas las fechas en las que se han realizado acciones. El administrador define un usuario y una fecha, el sistema le muestra la información (usuario, fecha, url, acción, dirección ip, hora) que existe sobre las acciones realizadas por dicho usuario en dicha fecha.
Referencia	R12

CU-14	Crear_Nuevo_Curso
Actor	Administrador

Descripción	Este caso de uso inicia cuando el administrador entra el la sección “Crear nuevo Curso” y el sistema le muestra una interfaz en la que puede realizar esta acción. Introduce el nuevo curso y acepta, el sistema verifica que no existe y guarda automáticamente la información en la base de datos del sistema, si ya existe muestra un mensaje” Ya existe el curso”
Referencia	R19

CU-15	Crear_Nuevo_Test
Actor	Administrador
Descripción	Este caso de uso inicia cuando el administrador entra en el sistema y selecciona la opción “Crear nuevo Test”, El sistema le muestra una interfaz en la que define el número del test, selecciona el curso, define la cantidad de preguntas normales que tendrá su test, y la cantidad de preguntas especificaciones. Luego marca la opción “Agregar” y el sistema le muestra una nueva interfaz con el número de la pregunta y le permite definir el valor (cantidad máxima que puede obtener un estudiante en la pregunta) y el peso (valor de ponderación que le dan a la pregunta los psicólogos); el sistema automáticamente guarda estos cambios, en la base de datos.
Referencia	R18

CU-16	Listar_ Test
Actor	Administrador
Descripción	Este caso de uso inicia cuando el administrador entra al sistema y selecciona la opción “Listar Test”. El sistema le muestra habilitada una interfaz con la lista de todos los test que están en la base de datos. Su número, el curso al que pertenece, la cantidad de preguntas y la opción “Ver datos del test”, luego selecciona esta última opción para el test del que desea saber la información; el sistema le muestra otra interfaz con el número, y el curso de dicho test además las preguntas con su valor, peso y tipo. Esto le permite al administrador ver las características de los test ya creados y tomar decisiones en su trabajo.
Referencia	R20

Para ver las descripciones de los casos de usos expandidos Consulte **Anexo II.**

2.10 Diagrama de casos de uso del sistema.

UML introduce la noción de un *paquete* como el ítem universal para agrupar elementos, permitiendo a los modeladores subdividir y categorizar sistemas. Los paquetes pueden ser usados en cualquier nivel, desde el nivel más alto, donde son usados para subdividir el sistema en dominios, hasta el nivel más bajo, donde son usados para agrupar casos de uso individuales, clases, o componentes. [21]

El diagrama de casos de uso muestra la interacción entre actores y casos de uso. Los casos de uso pueden agruparse por diferentes criterios para su mejor entendimiento conformando paquetes, a continuación se representa mediante un diagrama, los paquetes que componen el sistema.

Diagrama de paquetes que componen el sistema.

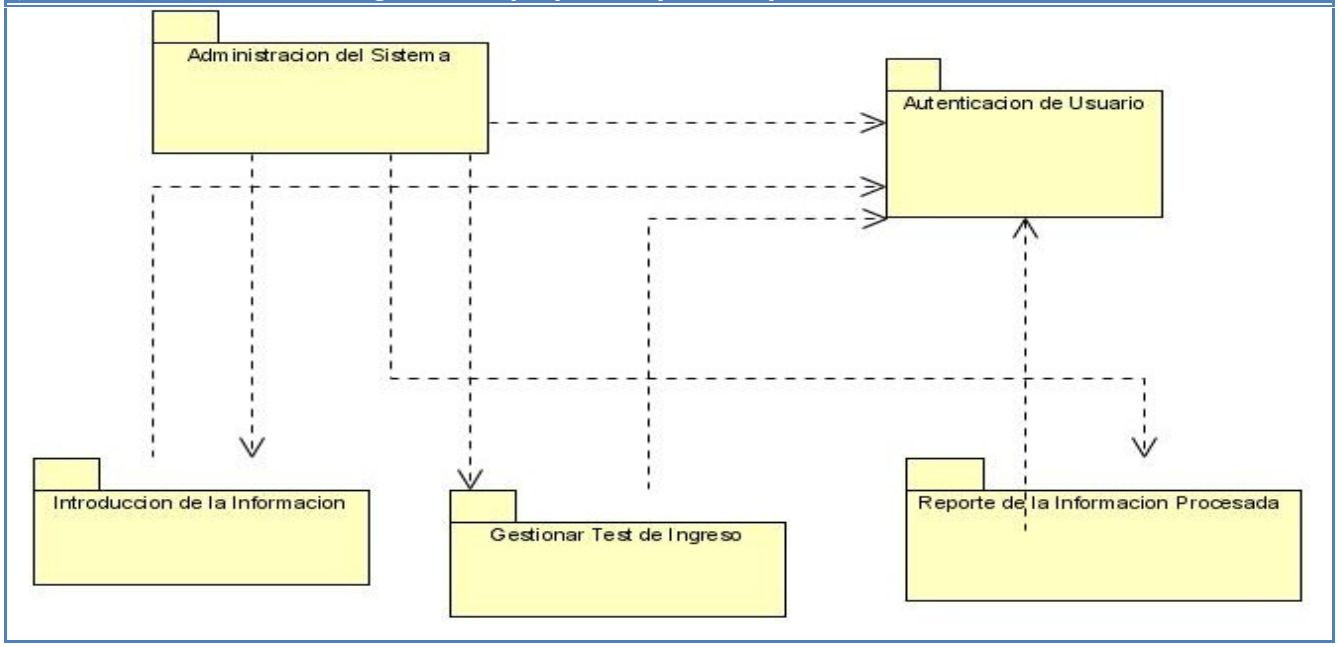

Figura 3 Diagrama de paquetes que componen el sistema.

Estructuración de los Casos de Uso del Sistema en paquetes.

Los Casos de Uso del sistema son procesos que responden a las funcionalidades definidas en los requerimientos, es decir, describen cómo el sistema debe funcionar e interactuar con el usuario.

Casos de Uso del sistema agrupados en el paquete Autenticación de Usuario.

Diagrama de CUS del paquete Autenticación de Usuario.

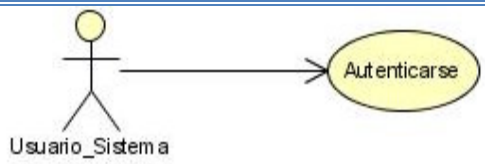

Figura 4 Diagrama de CUS del paquete Autenticación de Usuario.

Casos de Uso del sistema agrupados en el paquete Administración del Sistema.

Diagrama de CUS del paquete Administración del Sistema.

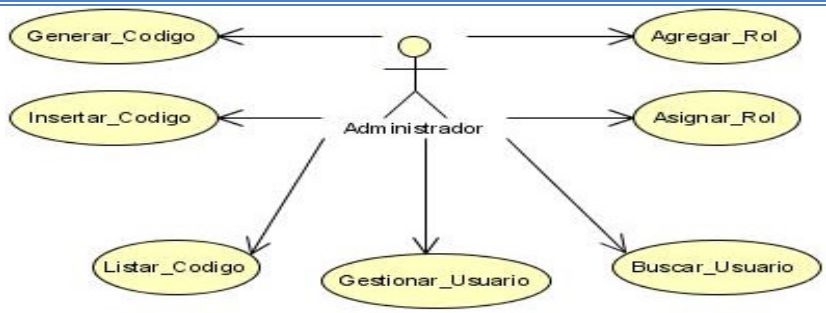

Figura 5 Diagrama de CUS del paquete Administración del Sistema.

Casos de Uso del sistema agrupados en el paquete Gestionar Test de Ingreso.

Diagrama de CUS del paquete Gestionar Test de Ingreso

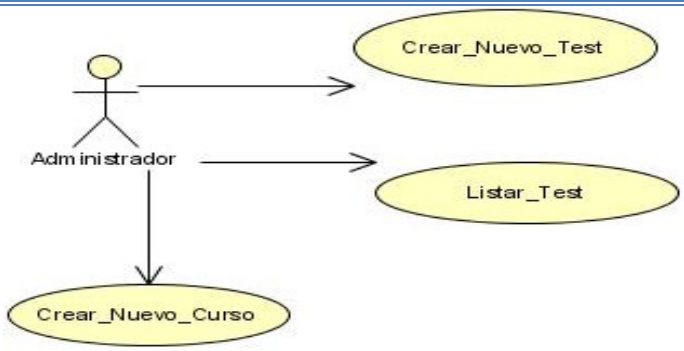

Figura 6 Diagrama de CUS del paquete Gestionar Test de ingreso.

Casos de Uso del sistema agrupados en el paquete Introducción de la Información.

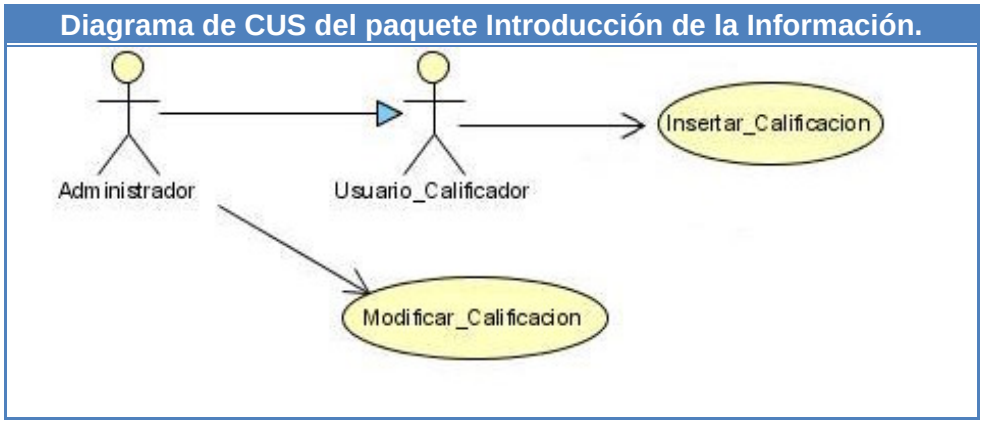

Figura 7 Diagrama de CUS del paquete Introducción de la Información.

Casos de Uso del sistema agrupados en el paquete Reporte de la Información Procesada.

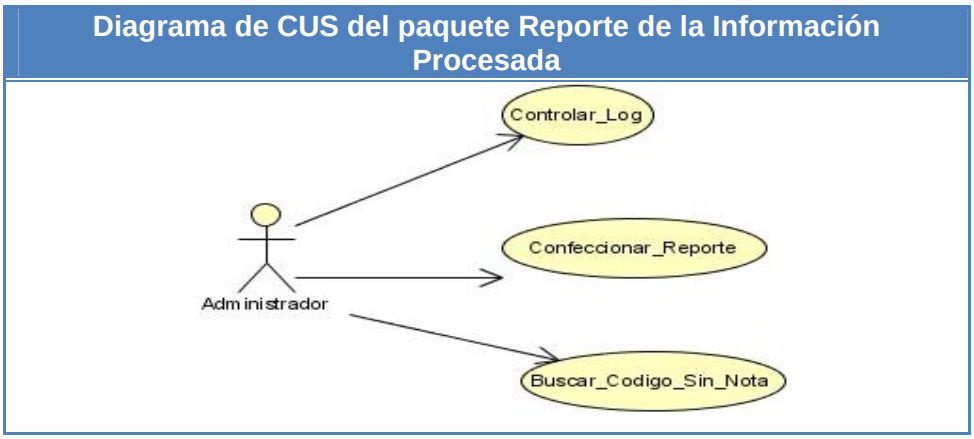

Figura 8 Diagrama de CUS del paquete Reporte de la Información Procesada.

2.11 Casos de uso por ciclo.

En esta sección se realiza una breve justificación de la agrupación de los casos de uso por paquetes. Ver **Anexo I.**

2.12 Conclusiones.

En este capítulo quedó reflejado: el Modelo de Dominio que representa los conceptos fundamentales: Calificación, Pregunta, Valor, Peso, Estudiante, Test, Código, Usuario_Calificador, Personal_Secretaría. El diagrama de paquetes que componen el sistema. Se definieron seis actores, dieciséis casos de uso del sistema, agrupados en cinco paquetes. Se definieron cuatro reglas del negocio. Veinte requerimientos funcionales. Los requerimientos no funcionales que el sistema debe cumplir. Queda bien detallada la

descripción del sistema propuesto y de sus casos de uso. Al concluir se tiene la base para comenzar el flujo de trabajo de Análisis y Diseño.

CAPÍTULO 3: ANÁLISIS Y DISEÑO DEL SISTEMA.

Este capítulo aborda el flujo de trabajo de Análisis y Diseño del software que se propone. Para ello se definen las clases del análisis y el Modelo de clases de análisis. En el diseño se realizan los modelos de clases del diseño para cada caso de uso, se definen los diagramas de interacción por cada realización de casos de uso. Se describen las clases de diseño y las tablas de la base de datos; de esta última se define el diagrama se clases persistentes y el modelo de datos.

3.1 Análisis.

En el flujo de trabajo del análisis se ofrece una especificación más precisa del modelo de casos de uso y los requisitos que se definieron en el flujo de trabajo Levantamiento de requisitos. El análisis se describe utilizando el lenguaje de los desarrolladores para felicitar la comprensión, preparación, modificación, y en general el mantenimiento de los requisitos, además de los conocimientos internos del sistema.

Un modelo de análisis puede considerarse como una primera aproximación al modelo de diseño (aunque es un modelo por si mismo), y es por tanto una entrada fundamental cuando se da forma al sistema en el diseño y en la implementación. [23]

3.2 Modelo conceptual de clases de análisis.

El modelo conceptual de clases del análisis se describe teniendo en cuanta el lenguaje del desarrollador. Representa la vista interna del sistema. Se utiliza por los desarrolladores para comprender como debería ser diseñado e implementado. Sirve como una primera aproximación al diseño. Define realizaciones de casos de uso, y cada una de ellas representa el análisis de un caso de uso del modelo de casos de uso. [23]

3.2.1 Clases de análisis.

Las clases del análisis son una representación más conceptual del diseño del sistema. Estas clases se centran en el tratamiento de los requisitos funcionales y pospone los no funcionales, hasta llegar a las actividades de diseño e implementación. En estas clases se definen atributos conceptuales reconocibles en el dominio del problema, a diferencias de los atributos de las clases del diseño y la implementación, que corresponden a un tipo de un leguaje de programación. Las relaciones entre este tipo de clases es también a nivel de concepto, en ellas la navegabilidad de las asociaciones no es tan importante como en

las clases de diseño e implementación. Interfaz, control y entidad son los tres estereotipos básicos en los que siempre van ajustar las clases del análisis.

3.2.2 Clases de Interfaz

Las clases de interfaz se utilizan para modelar la interacción entre el sistema y sus actores (es decir usuarios y sistemas externos). Esta interacción a menudo implica recibir (y presentar) información y peticiones de (y hacia) los usuarios y los sistemas externos. [23]

Las clases de interfaz representan a menudo abstracciones de ventanas, formularios, paneles, interfaces de comunicaciones, interfaces de impresoras, sensores, terminales. [23]

3.2.3 Clases de Entidad

Las clases de entidad se utilizan para modelar información que posee una vida larga y que es a menudo persistente. Las clases de entidad modelan la información y el comportamiento asociado de algún fenómeno o concepto, como una persona, un objeto del mundo real, o un suceso del mundo real. [23]

En la mayoría de los casos, las clases de entidad se derivan directamente de una clase de entidad del negocio (o de una clase del dominio) correspondiente, tomada del modelo de objetos del negocio (o del modelo del dominio). [23]

3.2.4 Clases controladoras

Las Clases de control representan coordinación, secuencia, transacción y control de otros objetos y se usan con frecuencia para encapsular el control de un caso de uso concreto. Las clases de control también se utilizan para representar derivaciones y cálculos complejos, como la lógica del negocio, que no pueden asociarse con ninguna información concreta, de larga duración, almacenada por el sistema (es decir una clase de entidad concreta). [23]

Los aspectos dinámicos del sistema se modelan con clases de control, debido a que ellas manejan y coordinan las acciones y los flujos de control principales, y delegan trabajo a otros objetos (es decir objetos de interfaz y de entidad). [23]

A continuación los diagramas de clase del análisis de los casos de uso críticos.

Diagrama Clases del Análisis para el Caso de Uso Autenticarse

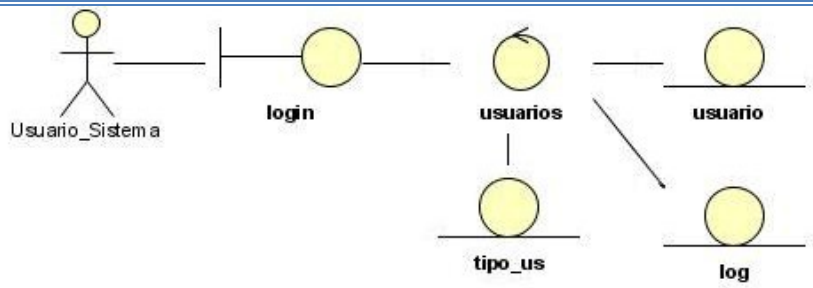

Figura 9 Diagrama de Clases del Análisis para el Caso de Uso Autenticarse.

Diagrama Clases del Análisis para el Caso de Uso Insertar_Calificacion

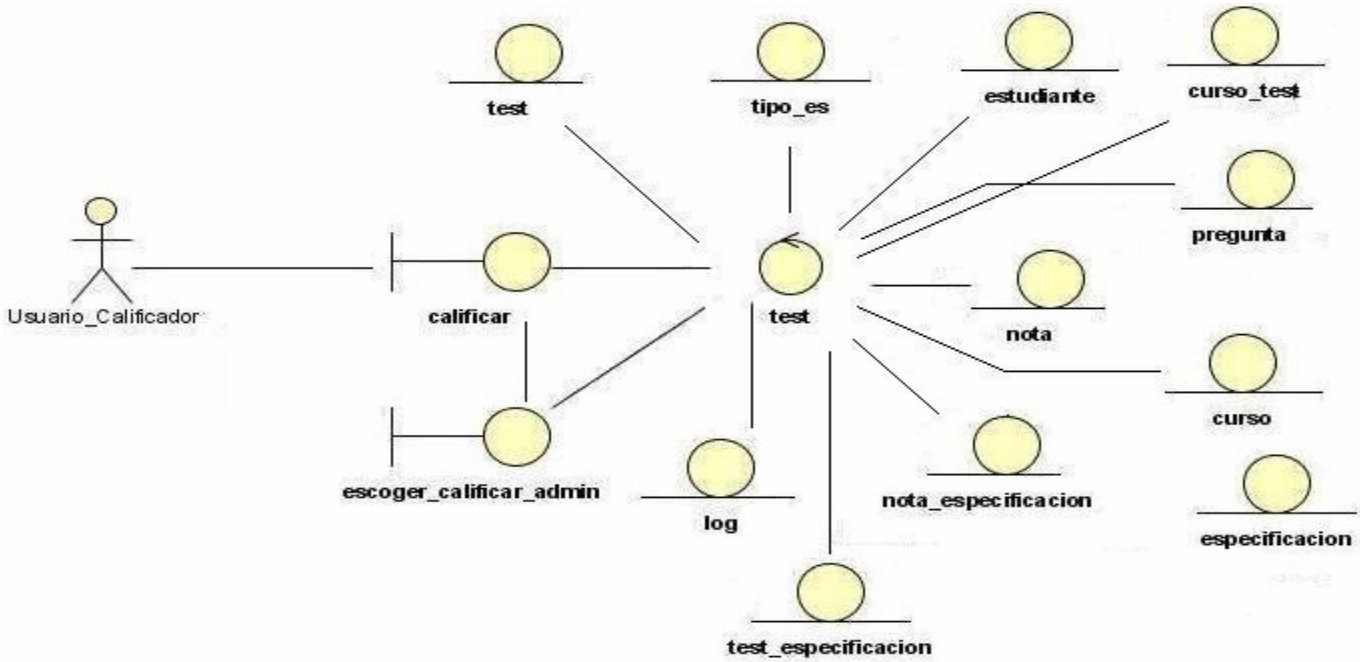

Figura 10 Diagrama de Clases del Análisis para el Caso de Uso Insertar_Calificacion.

Diagrama Clases del Análisis para el Caso de Uso Modificar_Calificacion

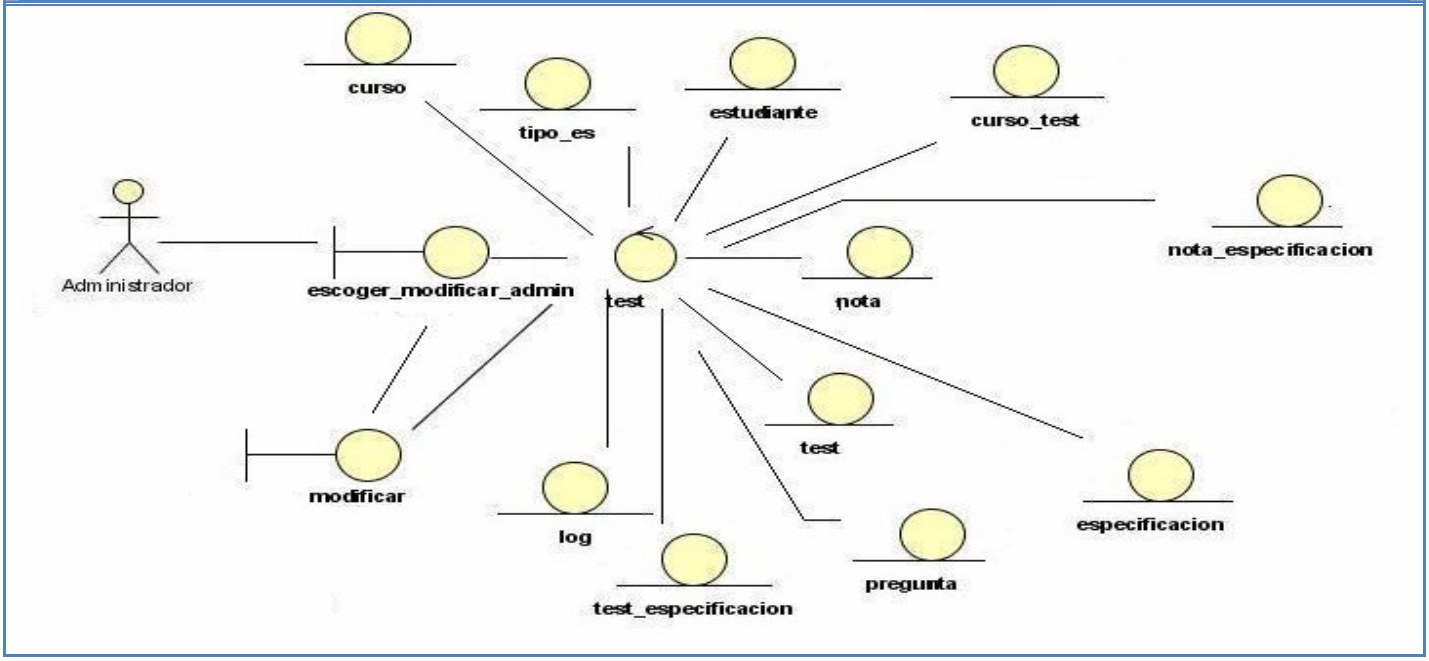

Figura 11 Diagrama de Clases del Análisis para el Caso de Uso Modificar_Calificacion

Diagrama Clases del Análisis para el Caso de Uso Asignar_Rol

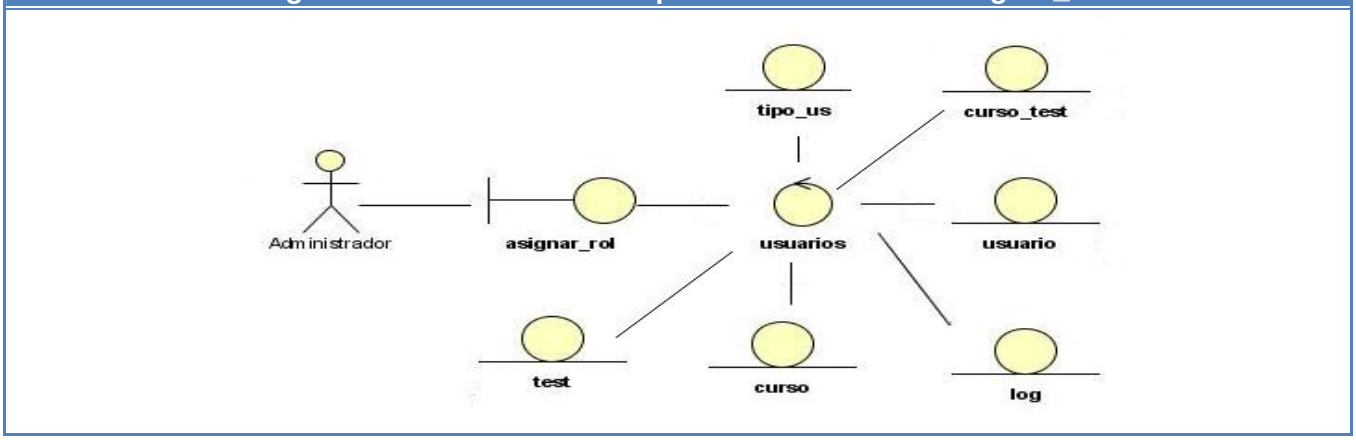

Figura 12 Diagrama de Clases del Análisis para el Caso de Uso Asignar_Rol.

Diagrama Clases del Análisis para el Caso de Uso Generar_Codigo

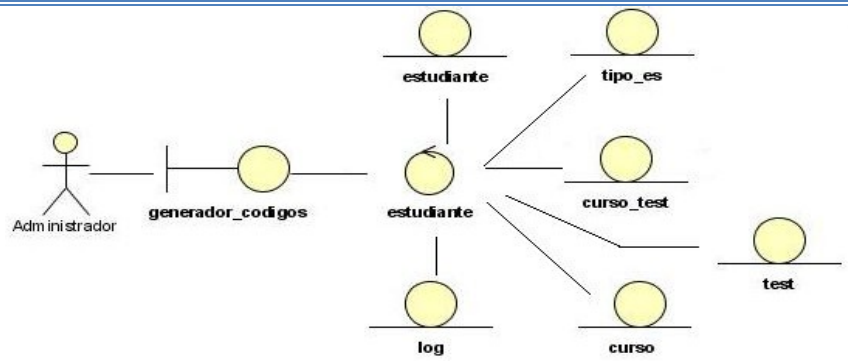

Figura 13 Diagrama de Clases del Análisis para el Caso de Uso Generar_Codigo.

Diagrama Clases del Análisis para el Caso de Uso Insertar_Codigo

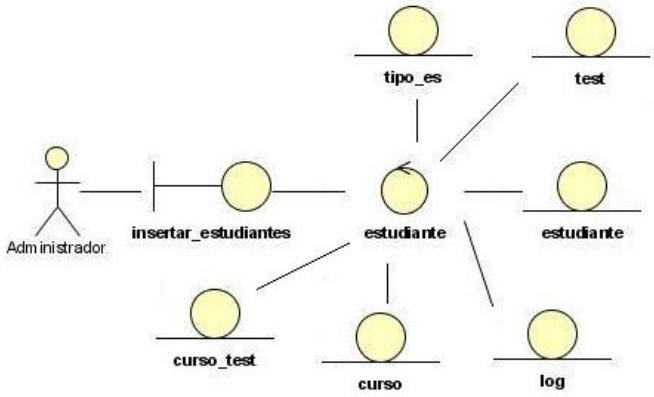

Figura 14 Diagrama de Clases del Análisis para el Caso de Uso Insertar_Codigo.

Diagrama Clases del Análisis para el Caso de Uso Confeccionar_Reporte

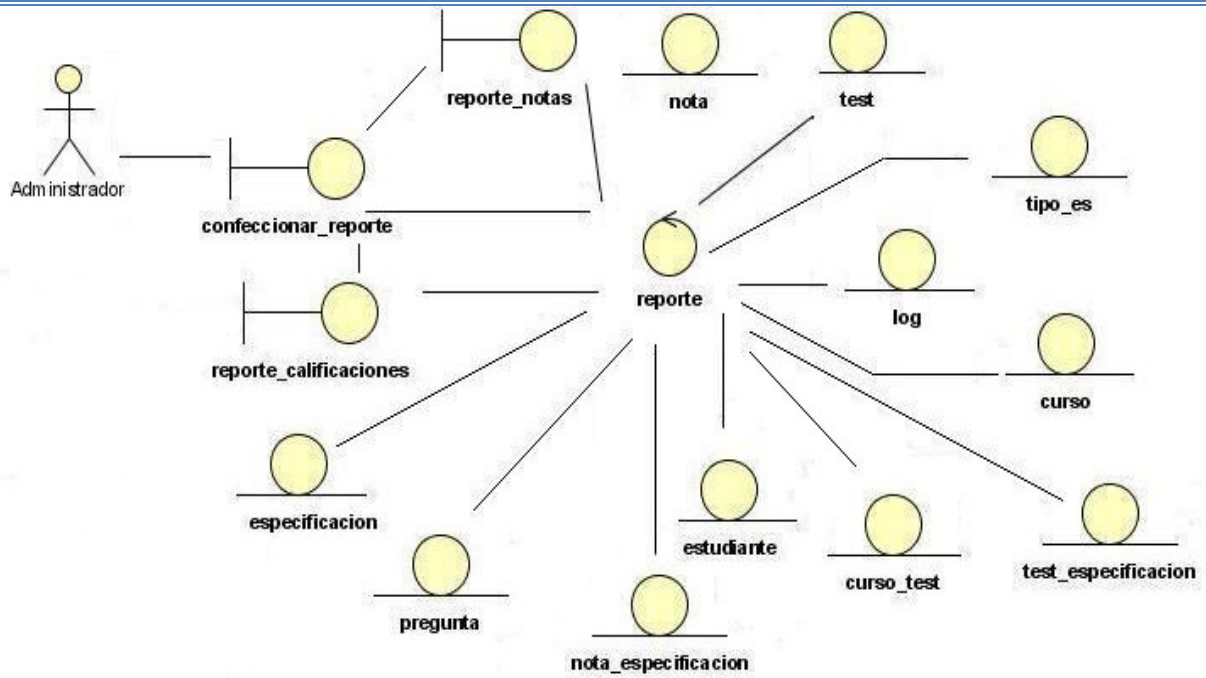

Figura 15 Diagrama de Clases del Análisis para el Caso de Uso Confeccionar_Reporte.

Diagrama Clases del Análisis para el Caso de Uso Crear_Nuevo_Curso

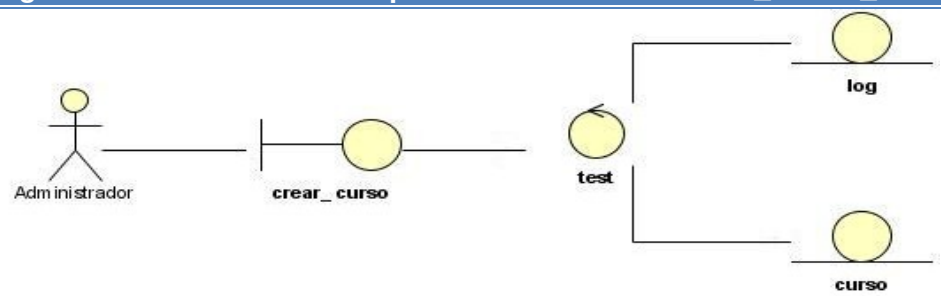

Figura 16 Diagrama de Clases del Análisis para el Caso de Uso Crear_Nuevo_Curso.

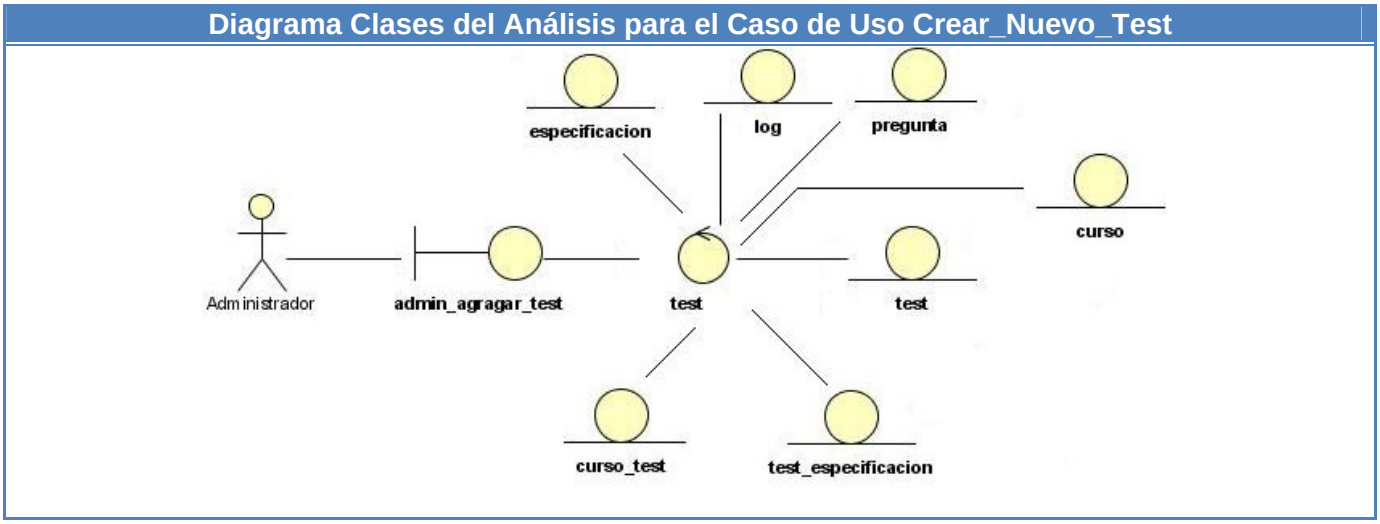

Figura 17 Diagrama de Clases del Análisis para el Caso de Uso Crear_Nuevo_Test.

3.3 Diseño.

El diseño es el centro de atención al final de la fase de elaboración y el comienzo de las iteraciones de construcción. Este contribuye a una arquitectura estable y sólida y a crear un plano del modelo de implementación.

En el diseño modelamos el sistema y encontramos su forma (incluida la arquitectura) para que soporte todos los requisitos **–** incluyendo los requisitos no funcionales y otras restricciones- que se le supone. En concreto, los propósitos del diseño son: [23]

- Adquirir una comprensión en profundidad de los aspectos relacionados con los requisitos no funcionales y restricciones relacionadas con los leguajes de programación, componentes reutilizables, sistemas operativos, tecnologías de distribución y concurrencia, tecnologías de interfaz de usuario, tecnologías de interfaz de transacciones, etc. [23]
- Crear una entrada apropiada y un punto de partida para actividades de implementación subsiguientes capturando los requisitos o subsistemas individuales, interfaces y clases. [23]
- Ser capaces de descomponer los trabajos de implementación en partes mas manejables que puedan ser llevadas a cabo por diferentes equipos de desarrollo, teniendo en cuenta la posible concurrencia. Esto resulta útil en los casos en los que la descomposición no puede ser hecha basándose en los resultados de la captura de requisitos. [23]

3.3.1 Diagramas de interacción.

Los diagramas de secuencia muestran la secuencia de mensajes entre objetos durante un escenario concreto. Cada objeto viene dado por una barra vertical y el tiempo transcurre de arriba abajo. Estos diagramas constituyen una guía importante para el programador; a continuación los diagramas de secuencia de la aplicación:

Ver diagramas de interacción **Anexo III.**

3.3.2 Diagramas de clases del diseño

El Diagrama de Clases del diseño es el diagrama principal en este flujo de trabajo. Un diagrama de clases presenta las clases del sistema con sus relaciones estructurales y de herencia. La definición de clase incluye definiciones para atributos y operaciones. El modelo de casos de uso aporta información para establecer las clases, objetos, atributos y operaciones. Cada clase se representa en un rectángulo con tres compartimientos: nombre de la clase, atributos de la clase, operaciones de la clase.

A continuación los diagramas de clases del diseño de los casos de uso críticos.

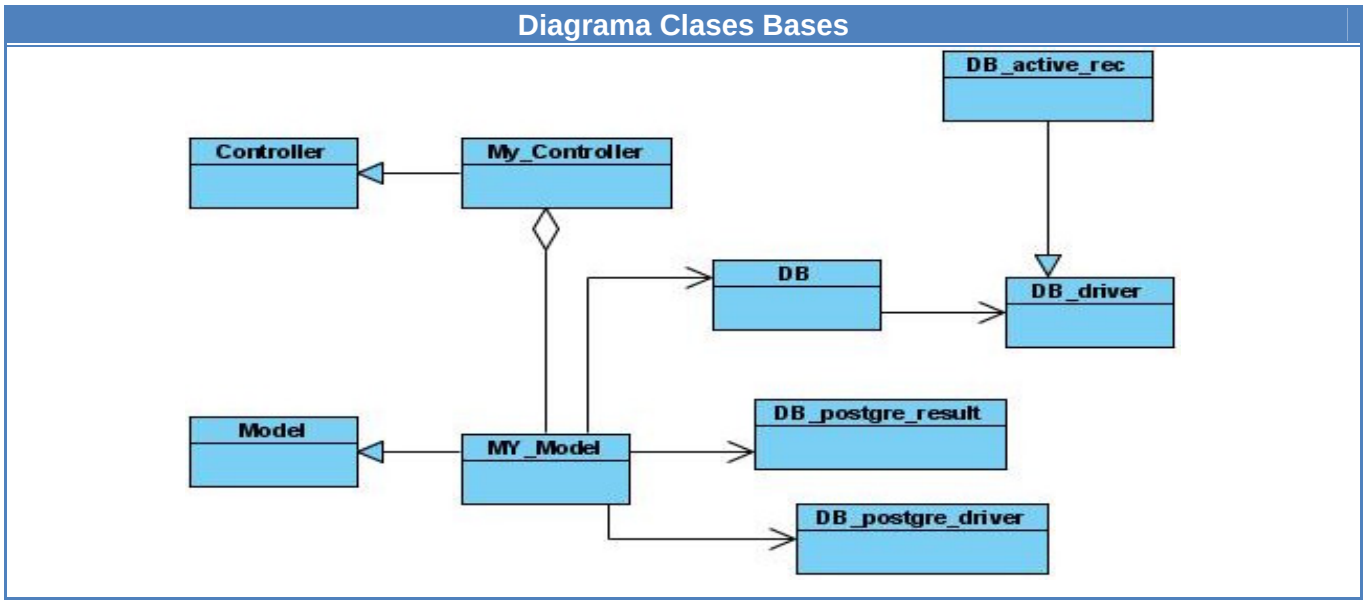

Figura 32 Diagrama de Clases Bases.

Diagrama Clases del Diseño para el Caso de Uso Autenticarse

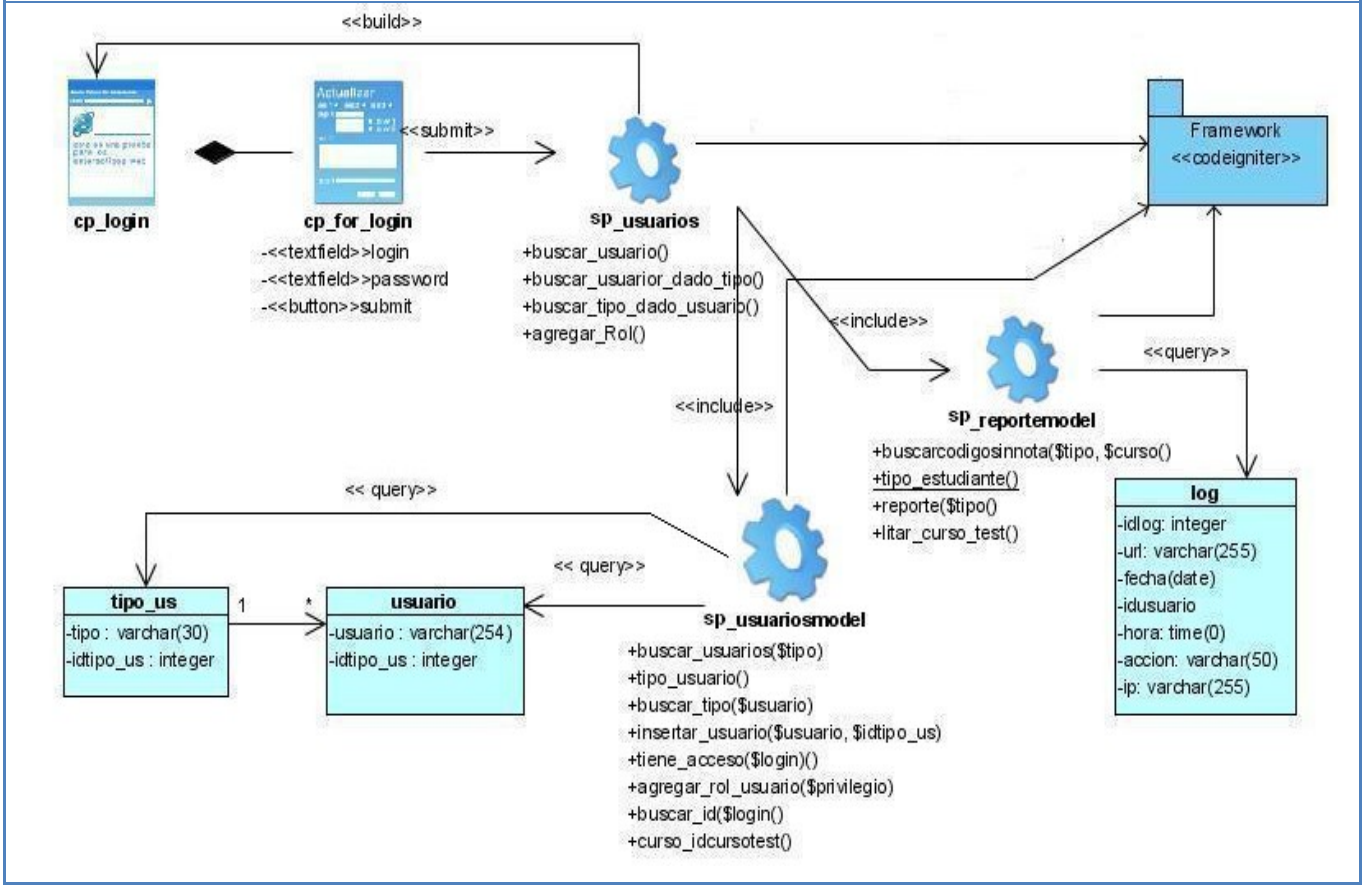

Figura 33 Diagrama de Clases del Diseño del Caso de Uso Autenticarse.

Diagrama Clases del Diseño para el Caso de Uso Insertar_Calificacion

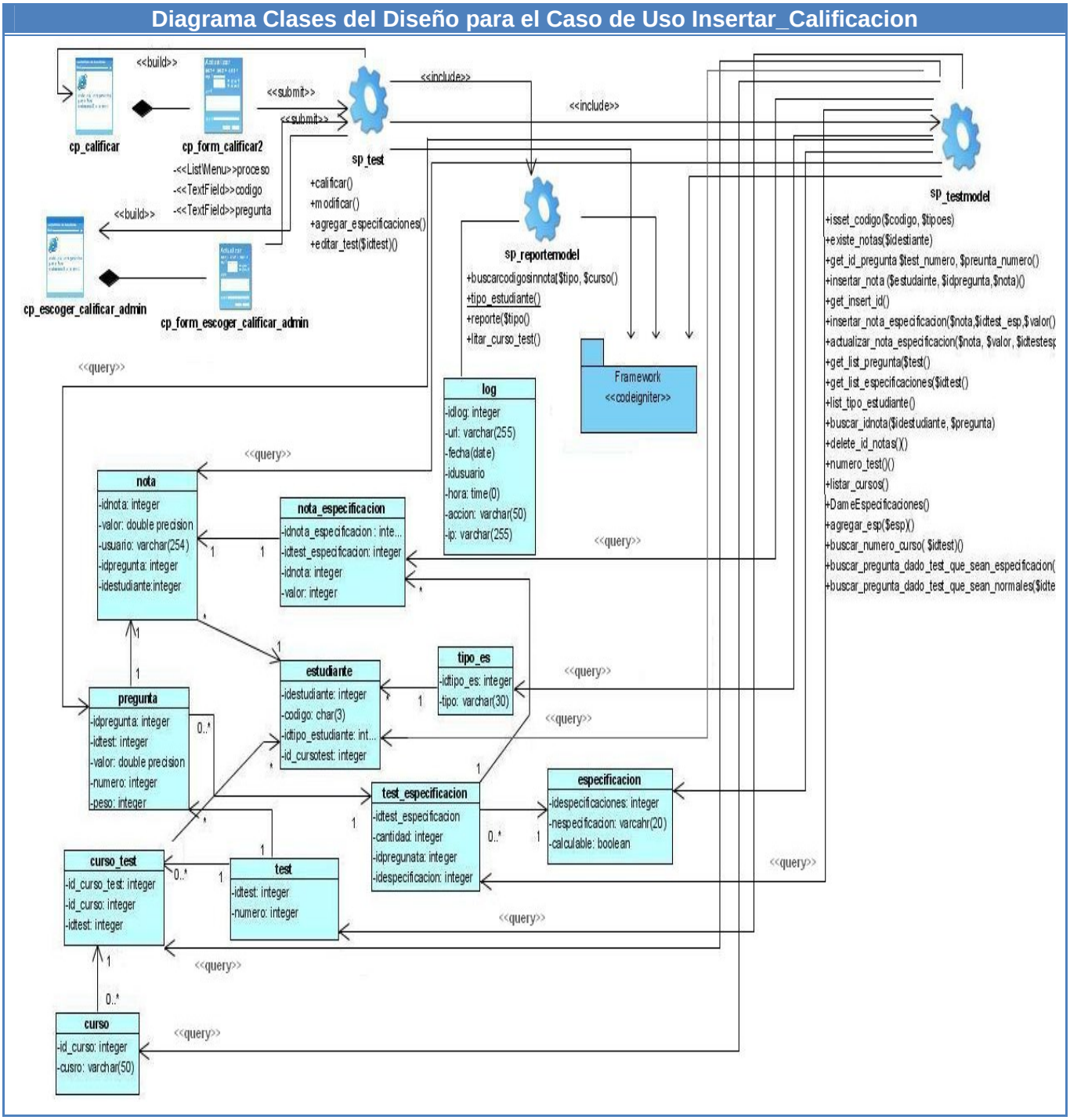

Figura 34 Diagrama de Clases del Diseño del Caso de Uso Insertar_Calificacion.

Diagrama Clases del Diseño para el Caso de Uso Modificar_Calificacion

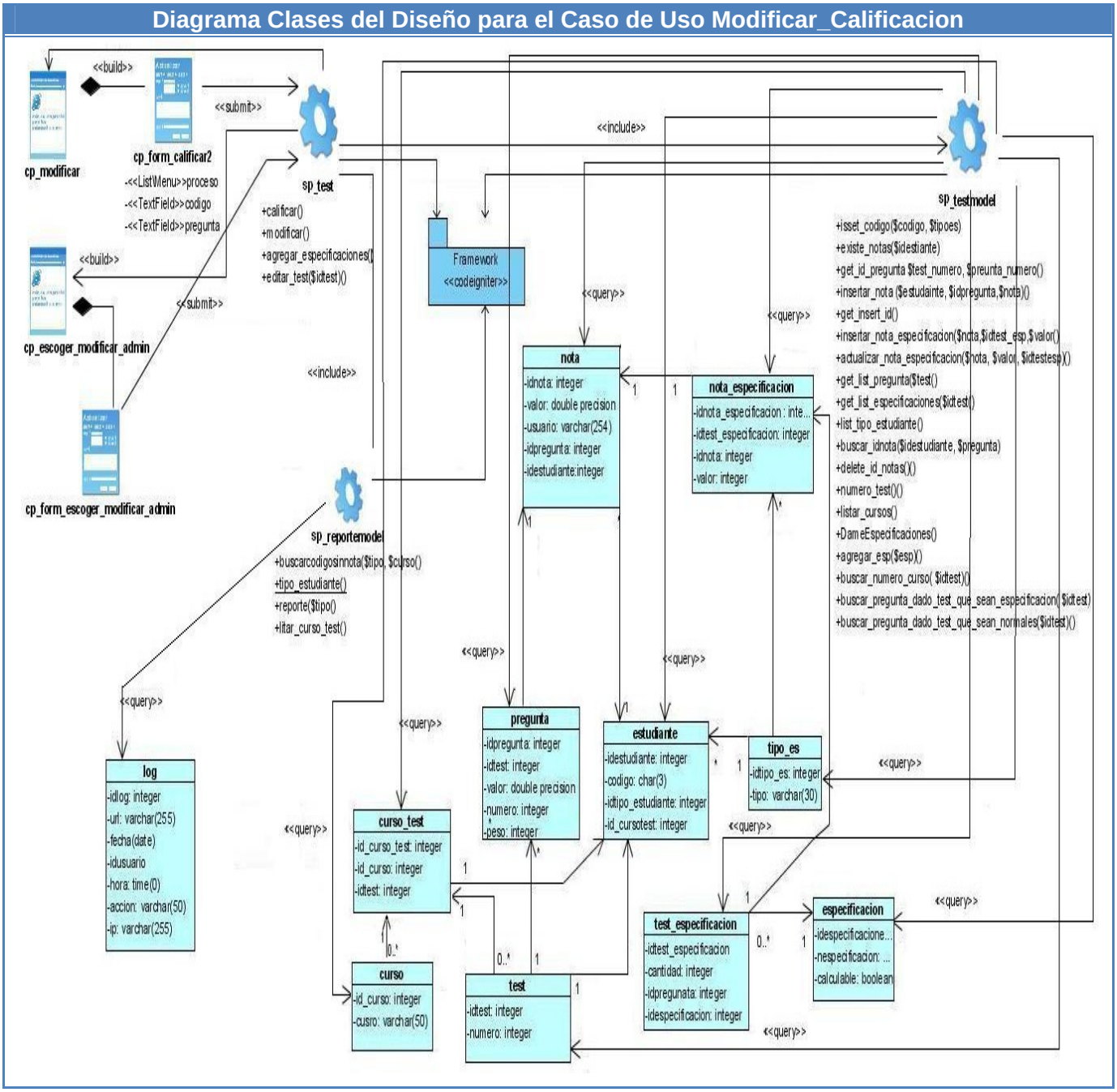

Figura 35 Diagrama de Clases del Diseño del Caso de Uso Modificar_Calificacion.

Diagrama Clases del Diseño para el Caso de Uso Asignar_Rol

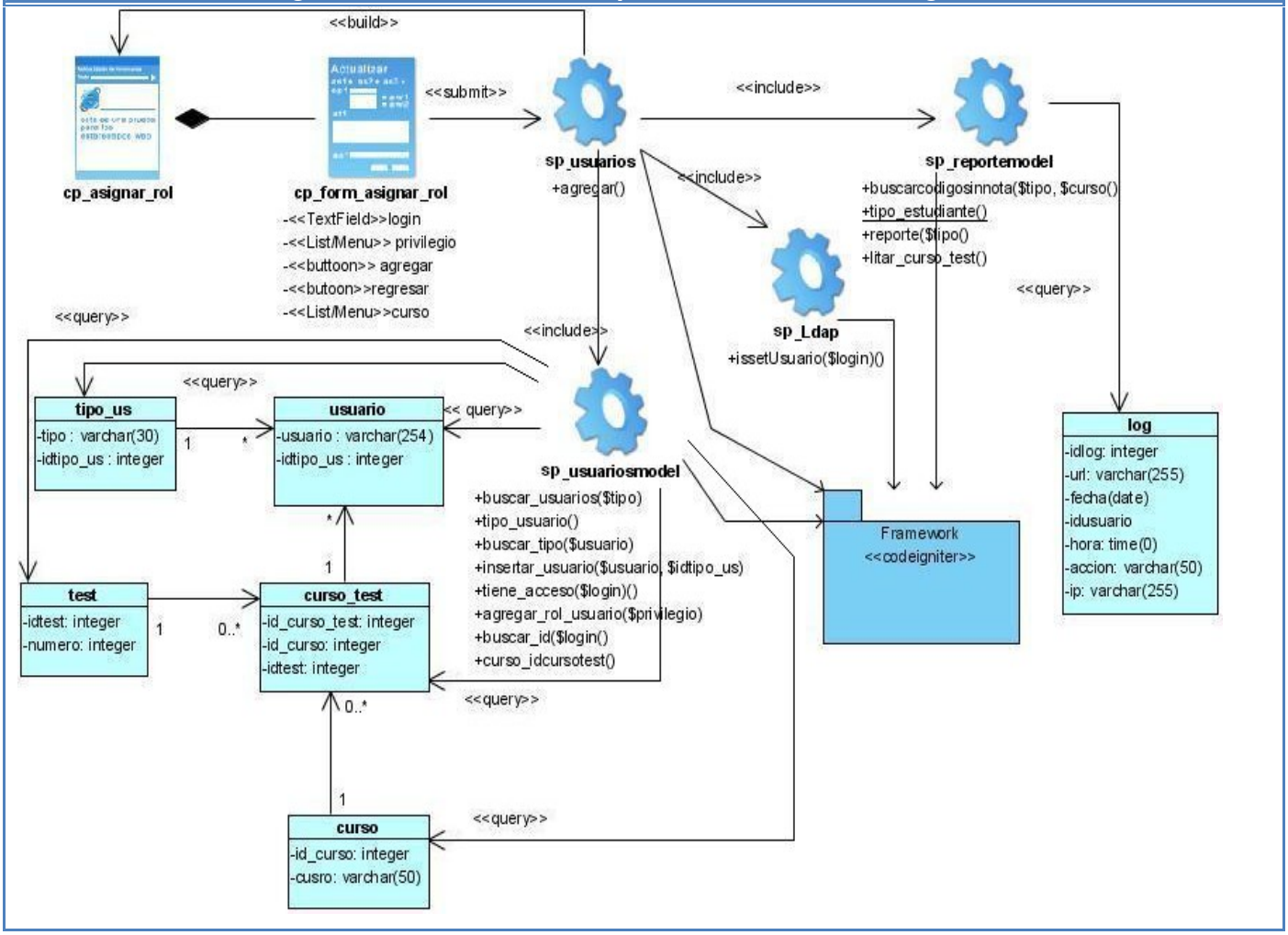

Figura 36 Diagrama de Clases del Diseño del Caso de Uso Asignar_Rol.

Diagrama Clases del Diseño para el Caso de Uso Generar_Codigo

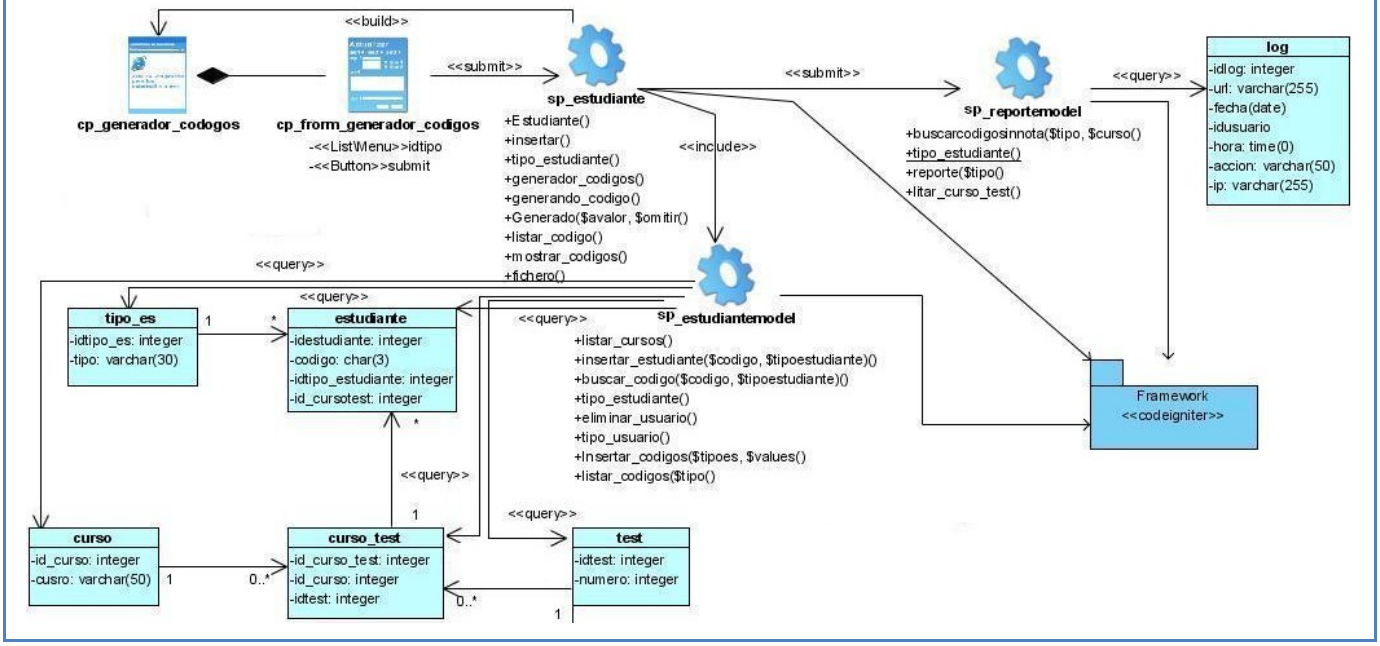

Figura 37 Diagrama de Clases del Diseño del Caso de Uso Generar_Codigo.

Diagrama Clases del Diseño para el Caso de Uso Insertar_Codigo

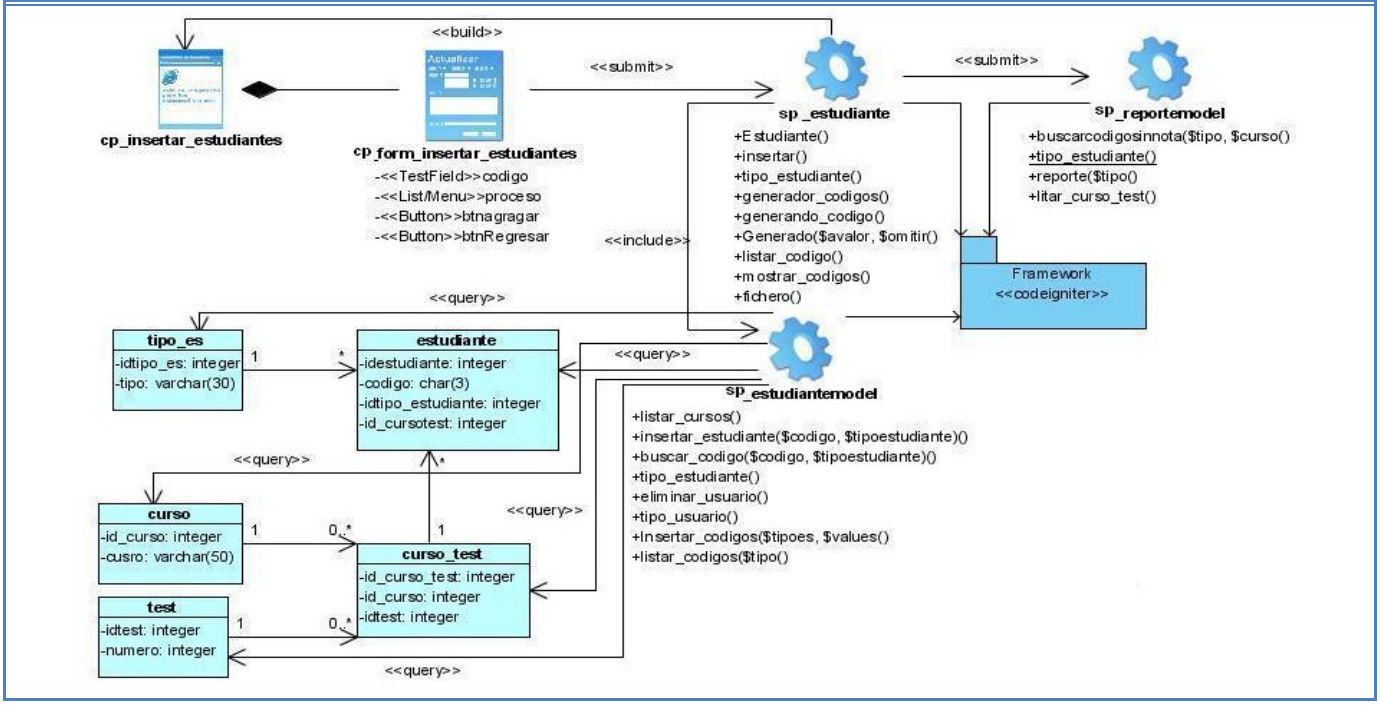

Figura 38 Diagrama de Clases del Diseño del Caso de Uso Insertar_Codigo.

Diagrama Clases del Diseño para el Caso de Uso Confeccionar_Reporte

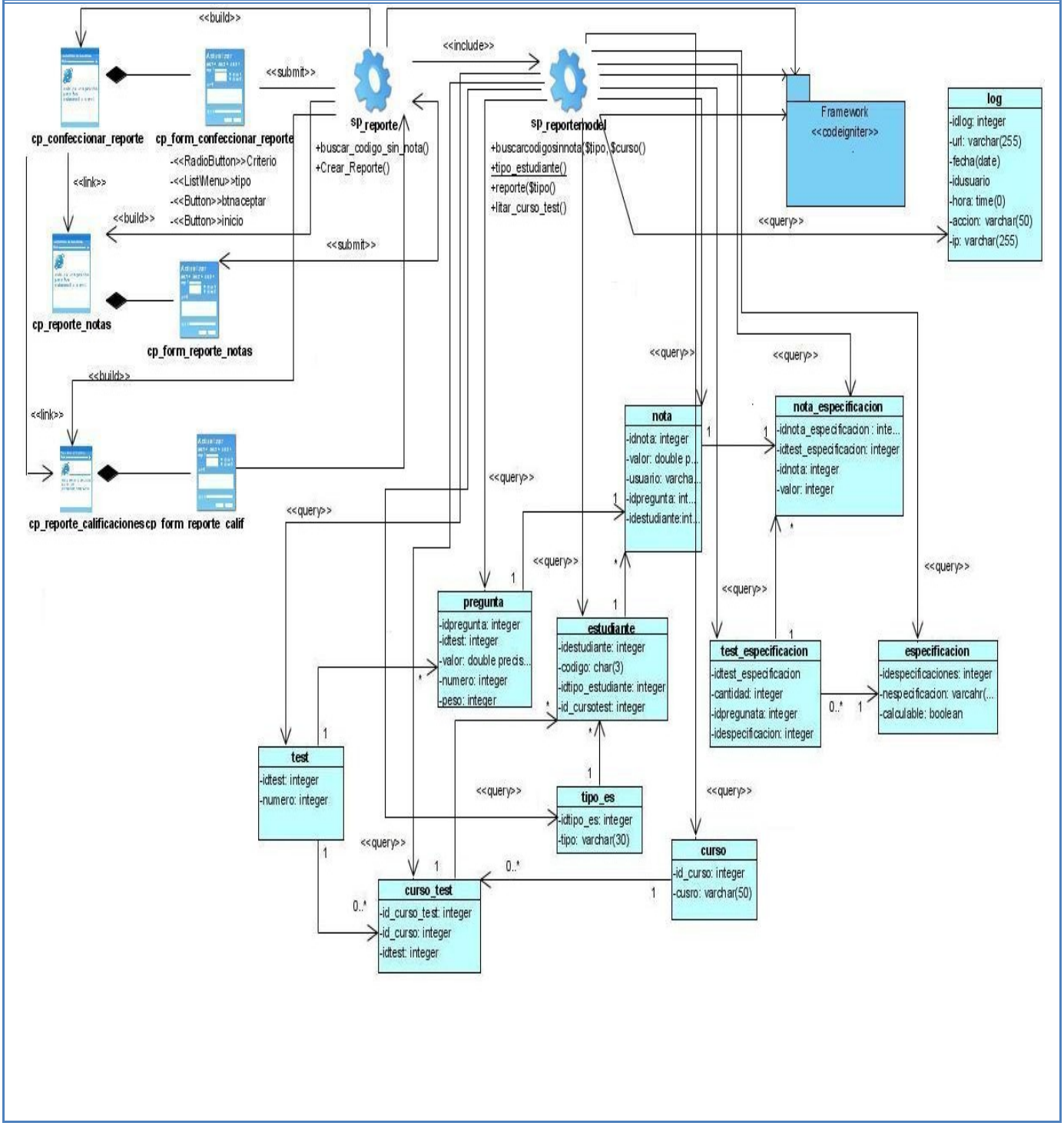

Figura 39 Diagrama de Clases del Diseño del Caso de Confeccionar_Reporte

Diagrama Clases del Diseño para el Caso de Uso Crear_Nuevo_Curso

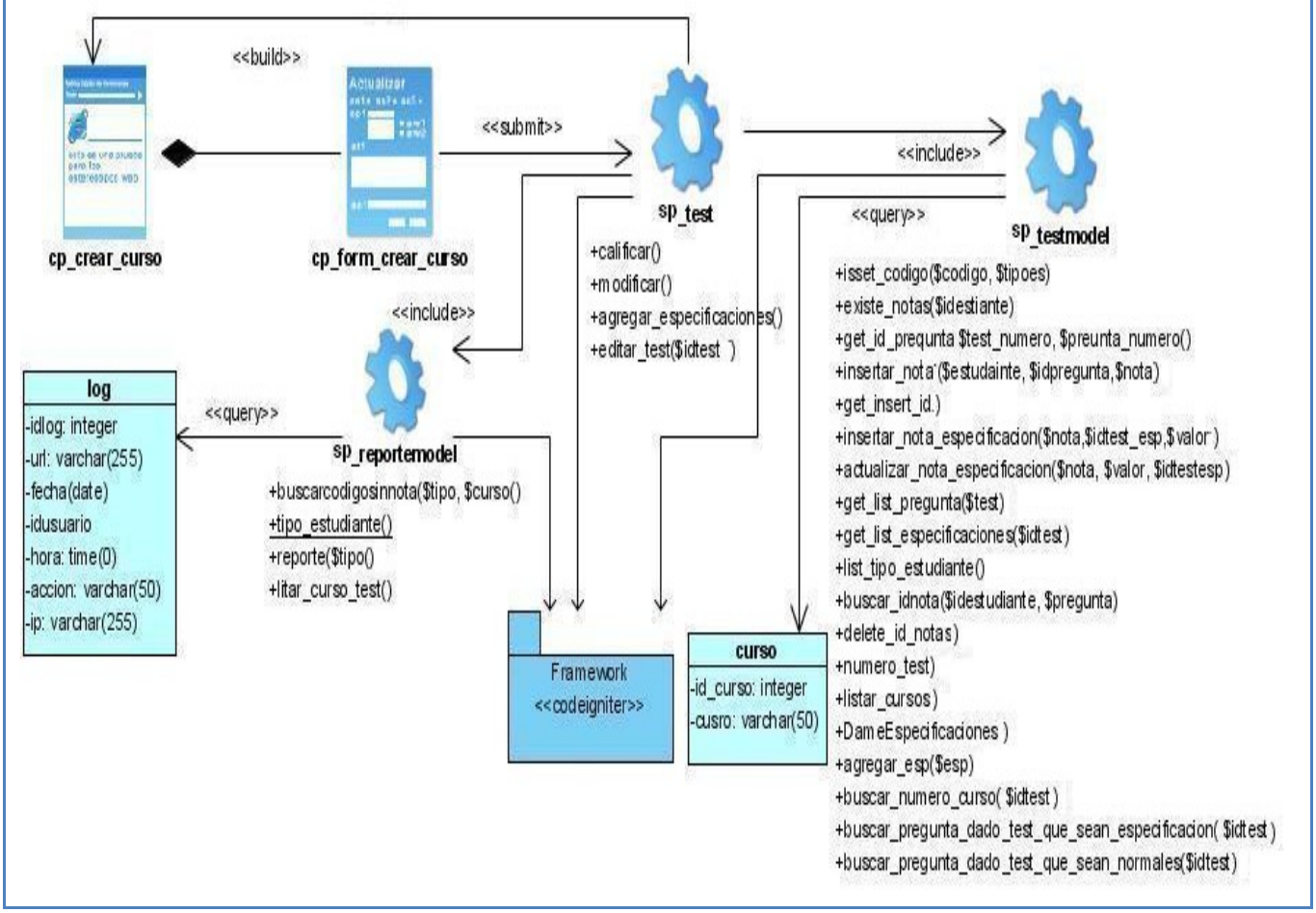

Figura 40 Diagrama de Clases del Diseño del Caso de Uso Crear_Nuevo_Curso.

Diagrama Clases del Diseño para el Caso de Uso Crear_Nuevo_Test

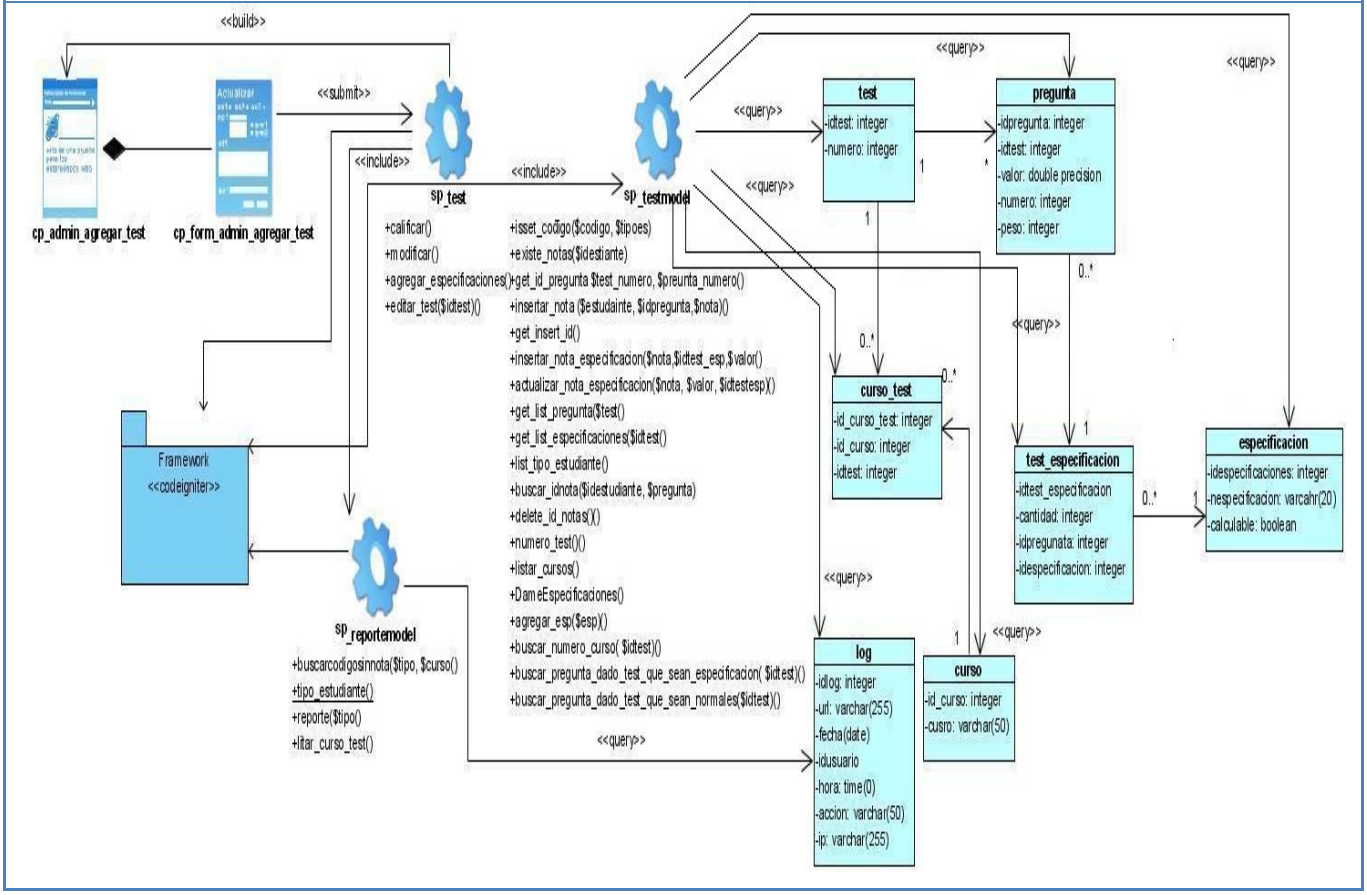

Figura 41 Diagrama de Clases del Diseño del Caso de Uso Crear_Nuevo_Test.

A continuación se describen las clases que conforman el diseño: **Ver Anexo IV.**

3.4 Diseño de la BD.

El diseño de la base de datos tiene gran importancia, está dirigido a facilitar el entendimiento de cómo es almacenada la información en el sistema.

3.4.1 Diagrama de clases persistentes.

Las clases persistentes son las clases que tienen la capacidad de mantener su valor en el espacio y en el tiempo, la necesidad de guardar su estado esta dado por al almacenamiento físico permanente de la información de la clase, para la copia de seguridad en caso del fracaso del sistema, o para el intercambio de información. A continuación se muestra el diagrama de clases persistentes.

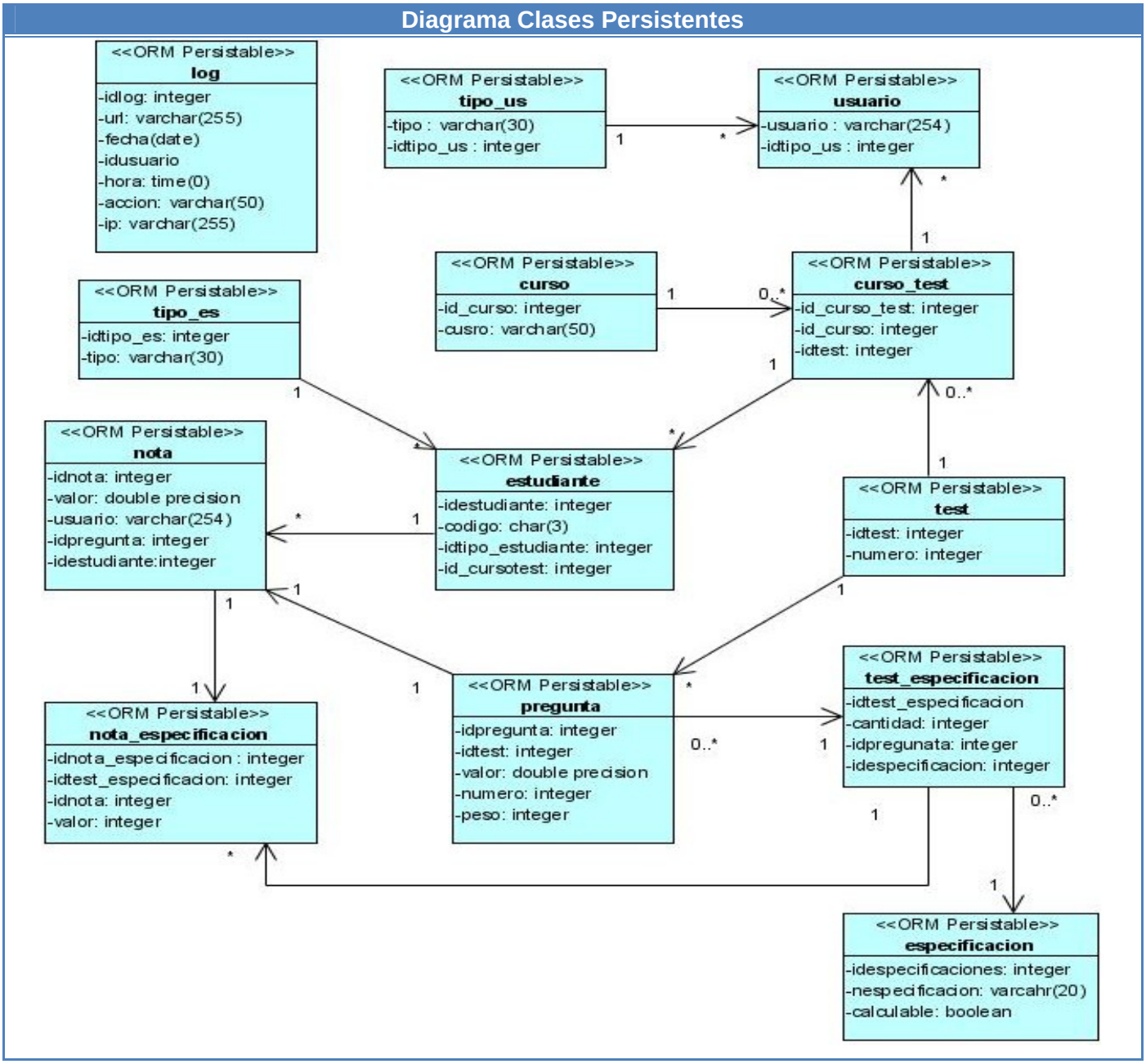

Figura 42 Diagrama de Clases persistentes.

3.4.2 Modelo de datos

El modelo de datos muestra la representación lógica y física de datos persistentes en el sistema. Seguidamente se representa el modelo de datos del sistema.

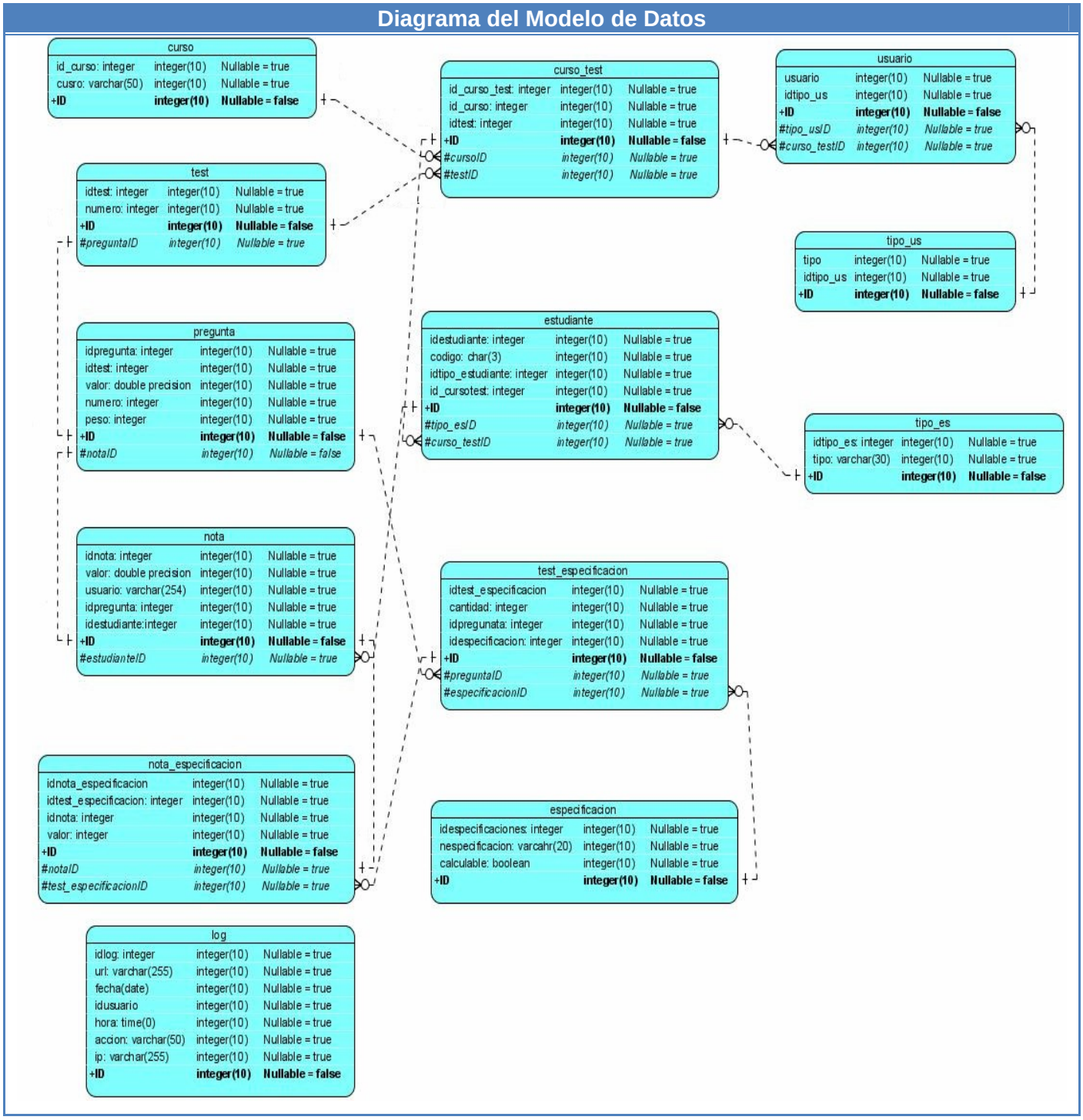

Figura 43 Diagrama del Modelo de datos.

3.4.3 Descripción de las tablas.

A continuación se describen las tablas que conforman el modelo de datos del sistema. **Ver Anexo V.**

3.5 Definiciones de diseño que se apliquen.

Para toda aplicación informática deben tenerse en cuenta los principios de diseño, porque hacer buen uso de estos, ayuda en gran medida a lograr aplicaciones sencillas, y amigables para el uso de los clientes: en el presente trabajo se tuvieron en cuenta:

Tratamiento de errores.

El tratamiento de errores es un tema de gran importancia para las aplicaciones informáticas. Para evitar posibles errores en la aplicación que se propone en este trabajo; se utilizaron las validaciones en JavaScript para evitar con estas que los usuarios y clientes introduzcan información errónea en el sistema y entorpecer así el buen funcionamiento del mismo. Además se implementó en estas una técnica de envío de mensajes de error advirtiendo o señalando los momentos en que estos errores son cometidos. El framework utilizado para la realización de la aplicación cuenta con una clase de tratamiento de errores, la cual trata todos los errores que se pueden devolver al ejecutar una acción incorrecta.

Seguridad.

Para garantizar la seguridad de la aplicación se siguió el patrón de diseño Modelo Vista Controlador (MVC) como lo indica la política de informatización del centro.

Quiere esto decir que se mantendrá una clara separación de la lógica de negocios, presentación y acceso a datos. Permitiendo flexibilidad y facilidad a la hora de hacer futuros cambios. Se hará una separación en módulos y ficheros de configuración, de forma que hacer cambios resulte rápido e intuitivo. [3]

Además se tuvo en cuenta las validaciones del lado del cliente, para ello se utilizó JavaScript, esto permite que no se le introduzcan datos incorrectos a la aplicación. En conjunto con las validaciones del lado del usuario se validaron las funciones en el servidor utilizando PHP. Estas evitan las inyecciones SQL a la base de datos, es decir impiden que se le introduzcan caracteres extraños, los mismos pueden causar pérdida de la información almacenada.

Se tuvo en cuenta el tratamiento de sesiones con variables de sesión que se hacen globales para mantener la integridad del sitio, con las misma se implementó una seguridad en niveles. Los niveles restringen el acceso de personal no autorizado a la información. Se cuanta con un módulo de

autenticación de usuarios, donde se comprueba que el usuario que está intentando acceder al sistema, primeramente pertenezca al Ldap, de pertenecer, se verifica que exista en la base de datos del sistema; si el usuario no se encuentra en esta no puede acceder a la aplicación.

Interfaz.

La interfaz es uno de los elementos más importantes en una aplicación ya terminada, pues debe ser sencilla, agradar al cliente y cumplir con las necesidades, sin ser muy sobrecargadas y así no distraer al usuario en otra actividad que no sea la que fue prevista por el propio cliente. En el sistema se utiliza el mismo color para todas las interfaces, y lograr con esto una homogeneidad en el diseño, se empleó un banner sencillo con el logo que representa el objetivo fundamental para el que fue creada la aplicación. El menú se ubica en un solo lado (izquierdo) para centrar toda la atención del cliente en ese punto.

Concepción de la ayuda.

Para garantizar un buen uso y aprovechamiento de todas las actividades que realiza una aplicación informática es aconsejable realizar una ayuda o manual de usuario. En este trabajo se elabora la misma de manera sencilla y ventajosa, pues mediante imágenes de las interfaces, se le explica al cliente todas las acciones que puede realizar en el sistema. Además el sistema cuenta con validaciones que envían mensajes de error, y ayudan al usuario a usar correctamente la aplicación.

3.6 Conclusiones.

El presente capítulo abordó el flujo de trabajo “Análisis” y “Diseño”, se definieron: las clases del análisis y los dieciséis diagramas de clases de análisis. Las clases del diseño, y los diecisiete diagramas de clases del diseño. Veintisiete diagramas de secuencia. Se representó el diagrama de clases persistentes y las descripciones de las tablas que conforman el modelo de datos. Quedan descritas además todas las clases del diseño. Al concluir quedan listos los artefactos para dar paso al flujo de trabajo Implementación.

CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBA

En el presente capítulo se aborda lo referente al flujo de trabajo “Implementación”, para ello se expone el diagrama de despliegue que corresponde a la aplicación que se ha presentado y descrito en los capítulos anteriores. También quedan reflejados en esta sección los diagramas de componentes que forman la base de la programación de este proyecto.

4.1 Implementación.

El propósito fundamental de la implementación es desarrollar la arquitectura y el sistema como un todo. Los artefactos que se desarrollan, son: el modelo de implementación, los diagramas de componentes, los subsistemas de implementación, las Interfaces, la descripción de la arquitectura (vista del modelo de implementación), el diagrama de despliegue y el plan de integración de construcciones. Las principales actividades de este flujo de trabajo son: implementar la arquitectura, implementar los subsistemas, implementar clases, implementar el sistema, y realizar la prueba unidad. Es en esta parte del flujo de trabajo donde se concreta la mayor cantidad de tareas para realizar un software.

4.1.1 Diagramas de Despliegue

Los diagramas de despliegue son necesarios si se desarrolla un software que interactúa con dispositivos que normalmente no gestiona el Sistema Operativo. Y el sistema está distribuido físicamente sobre varios procesadores.

La mayoría de las veces cuando se trata de la relación entre hardware y software se utilizan los diagramas de despliegue para razonar sobre la topología de procesadores y dispositivos sobre los que se ejecuta el software; es decir modelar la topología del hardware sobre el que se ejecuta el sistema.

Un diagrama de despliegue es un grafo de nodos unidos por conexiones de comunicación. Muestra las relaciones físicas entre los componentes hardware y software en el sistema final, es decir, la configuración de los elementos de procesamiento en tiempo de ejecución y los componentes software. [24]

A continuación se expone el diagrama de despliegue que corresponde al presente proyecto.

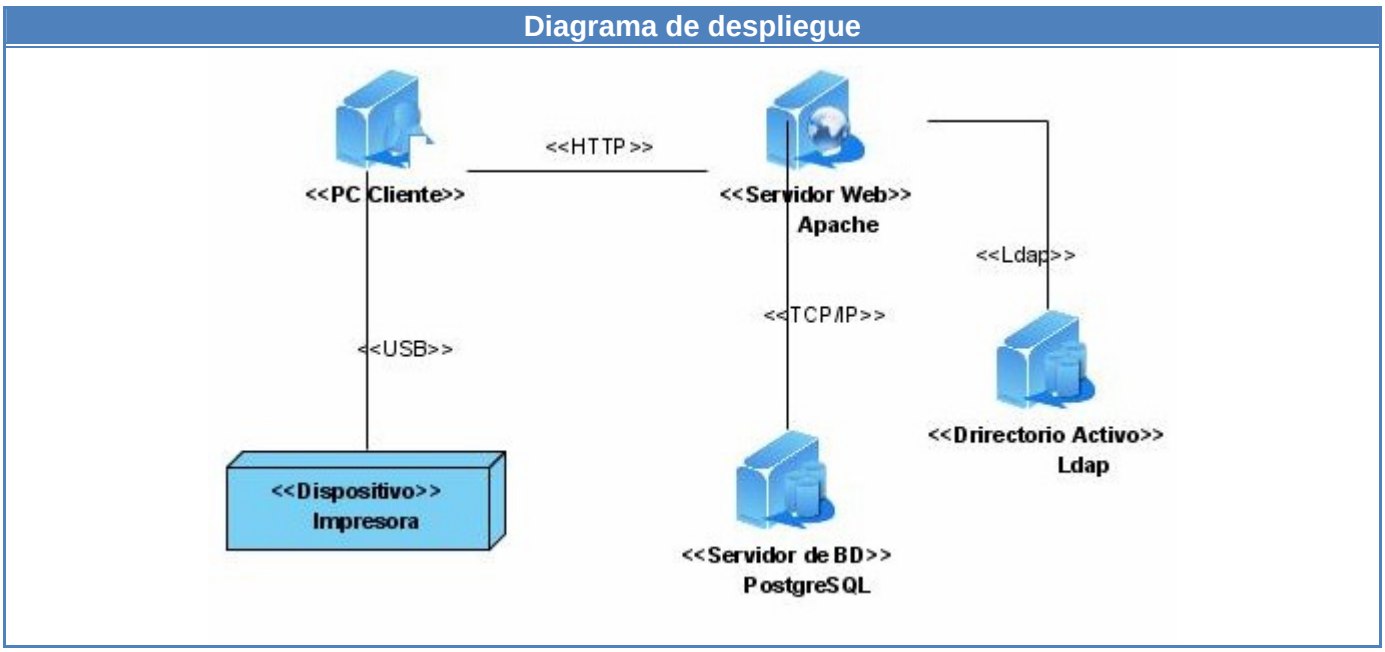

Figura 44 Diagrama de despliegue.

4.1.2 Diagrama de componentes.

Un diagrama de componentes muestra las organizaciones y dependencias lógicas entre componentes software, sean éstos componentes de código fuente, binarios o ejecutables. Los elementos de modelado dentro de este tipo de diagramas serán componentes y paquetes. Estos diagramas muestran las interfaces, controladoras, acceso a datos, y su interrelación, en muchos aspectos se puede considerar que un diagrama de componentes es un diagrama de clases a gran escala. Además este diagrama se representa como un grafo de componentes software unidos por medio de relaciones de dependencia (generalmente de compilación).

A continuación se representan los diagramas de componentes.

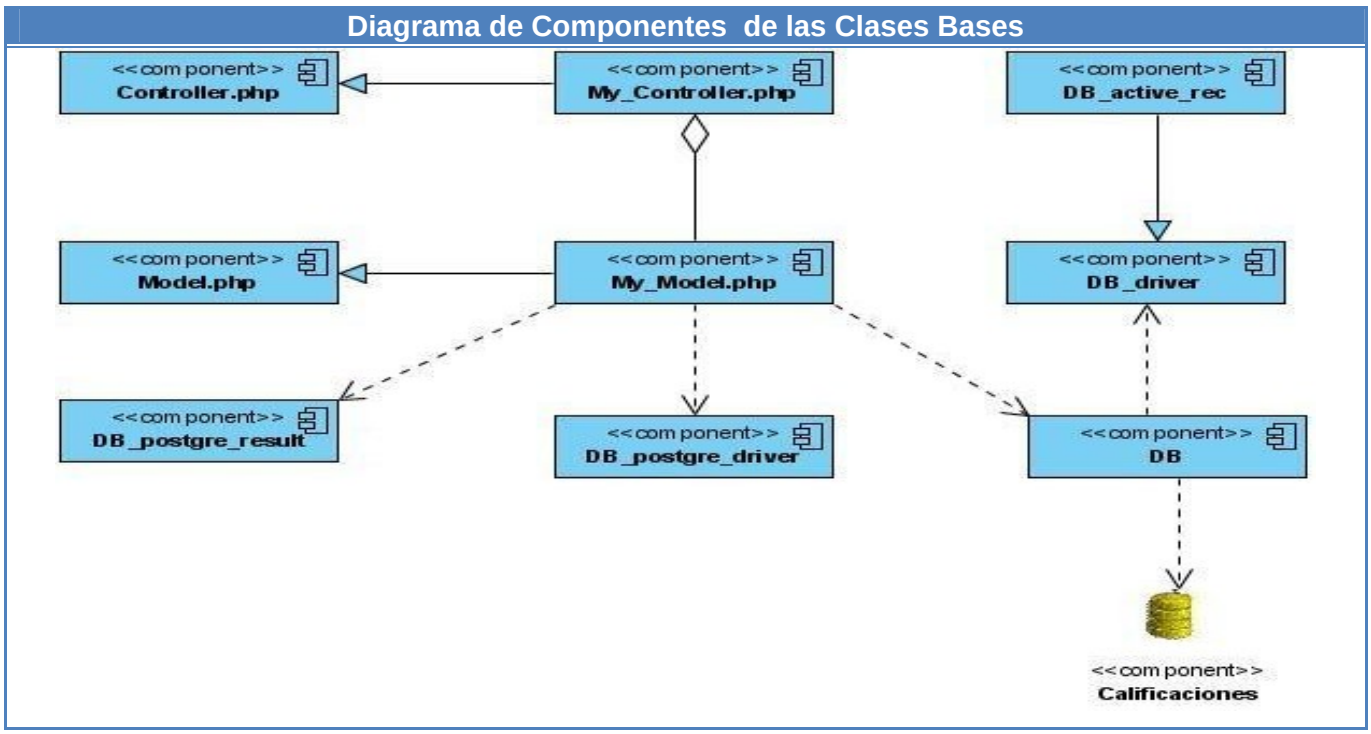

Figura 45 Diagrama de Componentes de las Clases Bases.

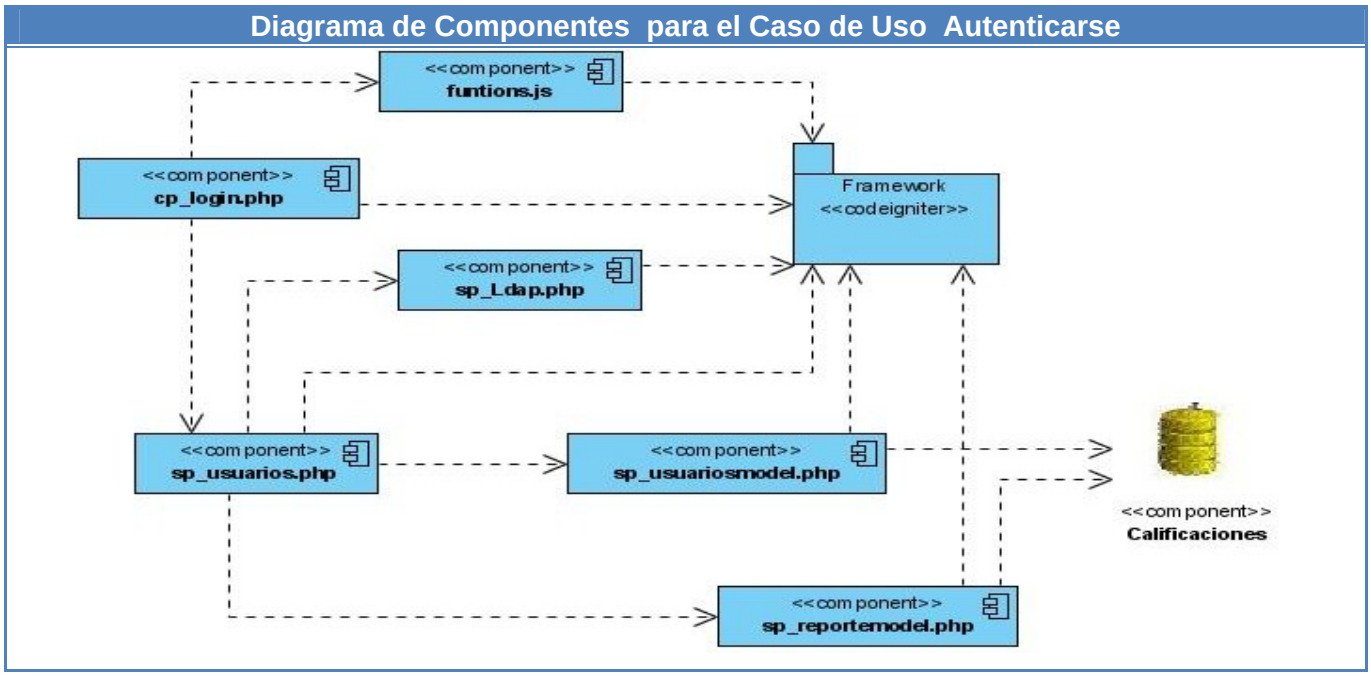

Figura 46 Diagrama de Componentes del Case de Uso Autenticarse.

Diagrama de Componentes para el Caso de Uso Insertar_Calificacion

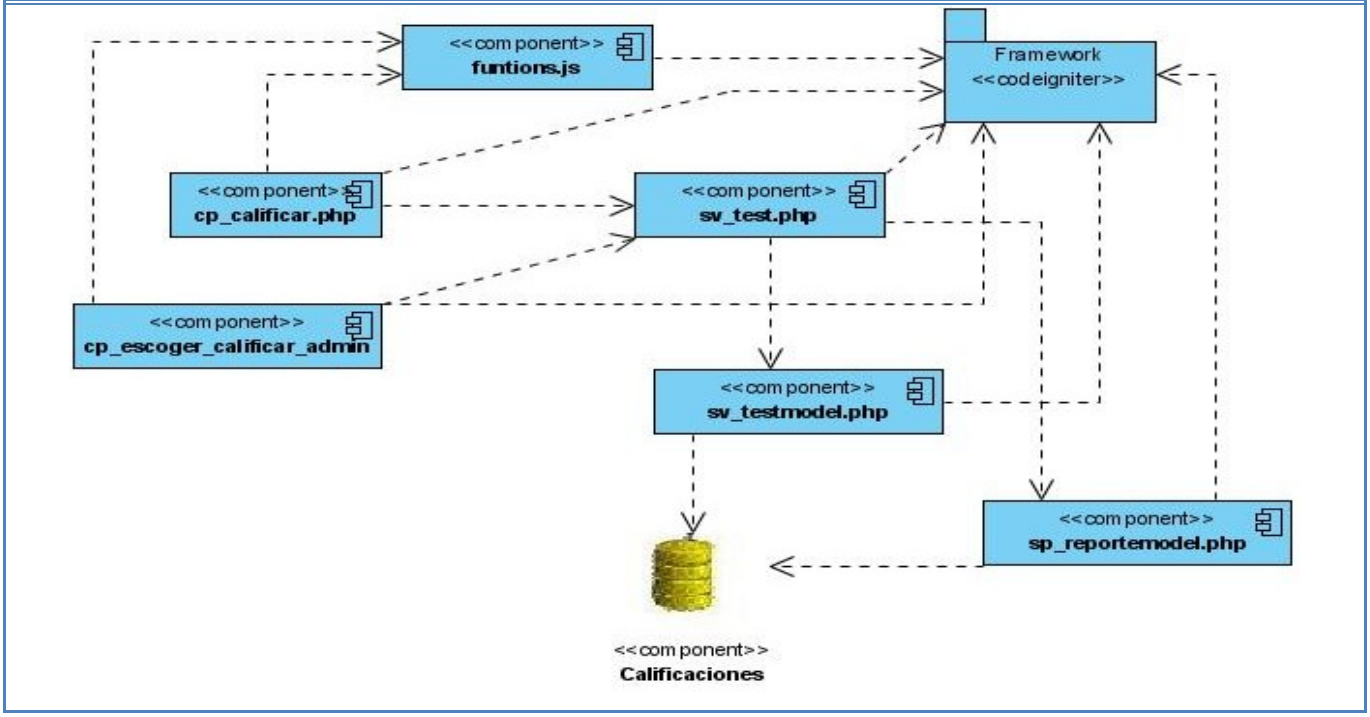

Figura 47 Diagrama de Componentes del Case de Uso Insertar_Calificacion.

Diagrama de Componentes para el Caso de Uso Modificar_Calificacion

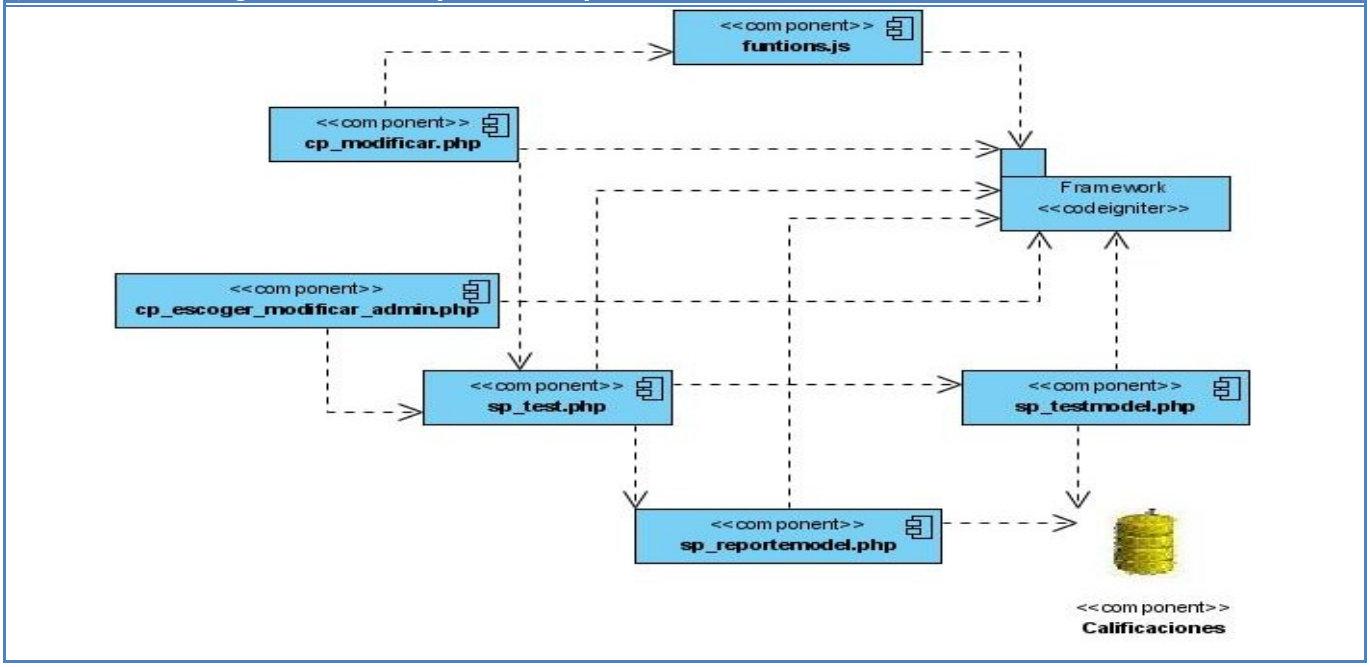

Figura 48 Diagrama de Componentes del Case de Uso Modificar_Calificacion.

Diagrama de Componentes para el Caso de Uso Asignar_Rol

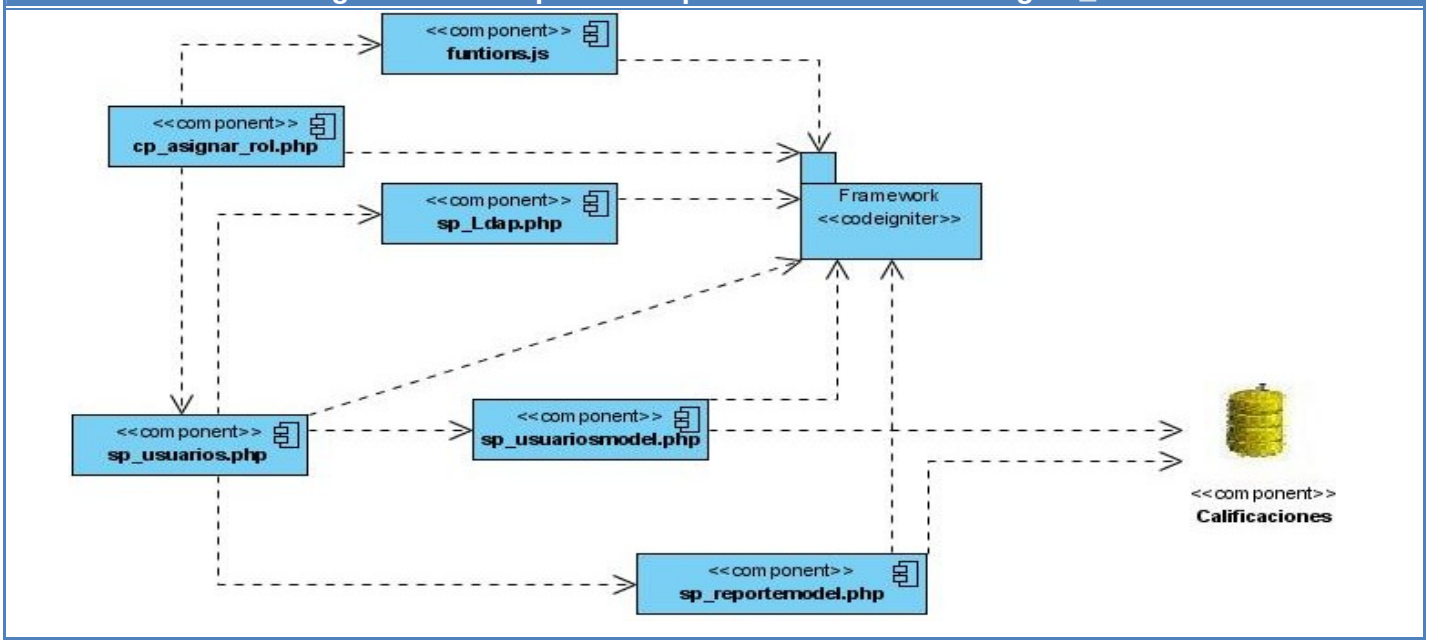

Figura 49 Diagrama de Componentes del Case de Uso Asignar_Rol.

Diagrama de Componentes para el Caso de Uso Generar_Codigo

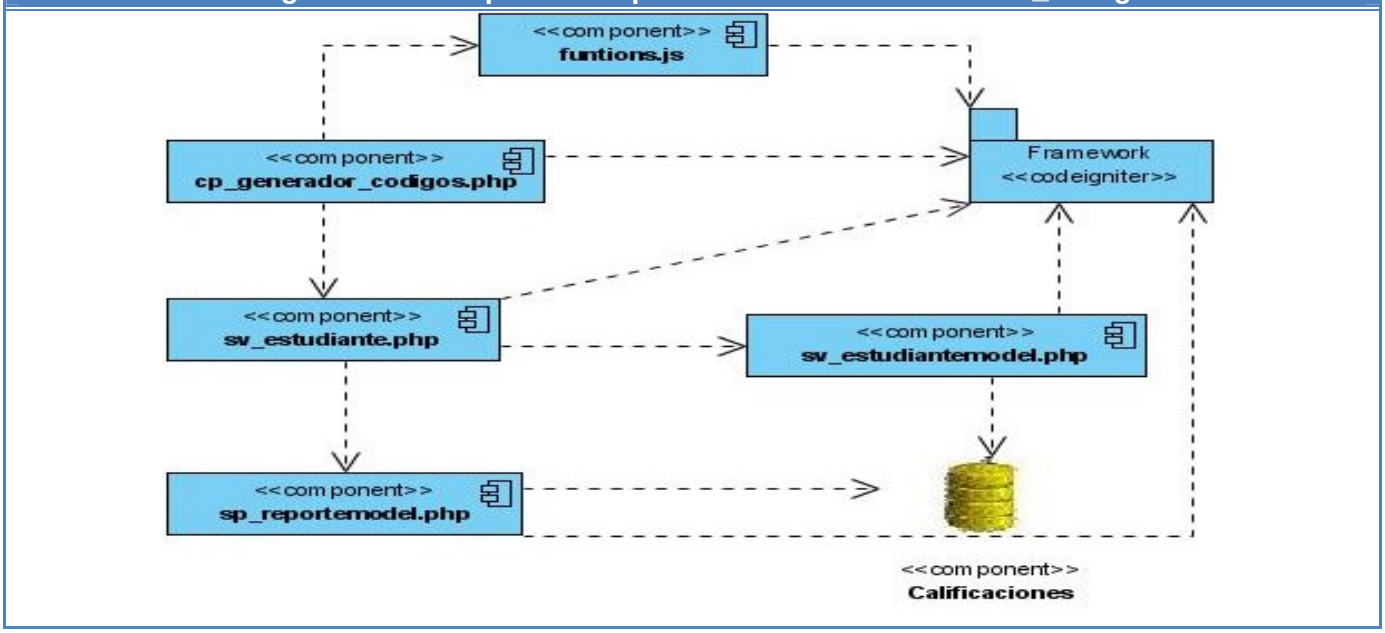

Figura 50 Diagrama de Componentes del Case de Uso Generar_Codigo.

Diagrama de Componentes para el Caso de Uso Insertar_Codigo

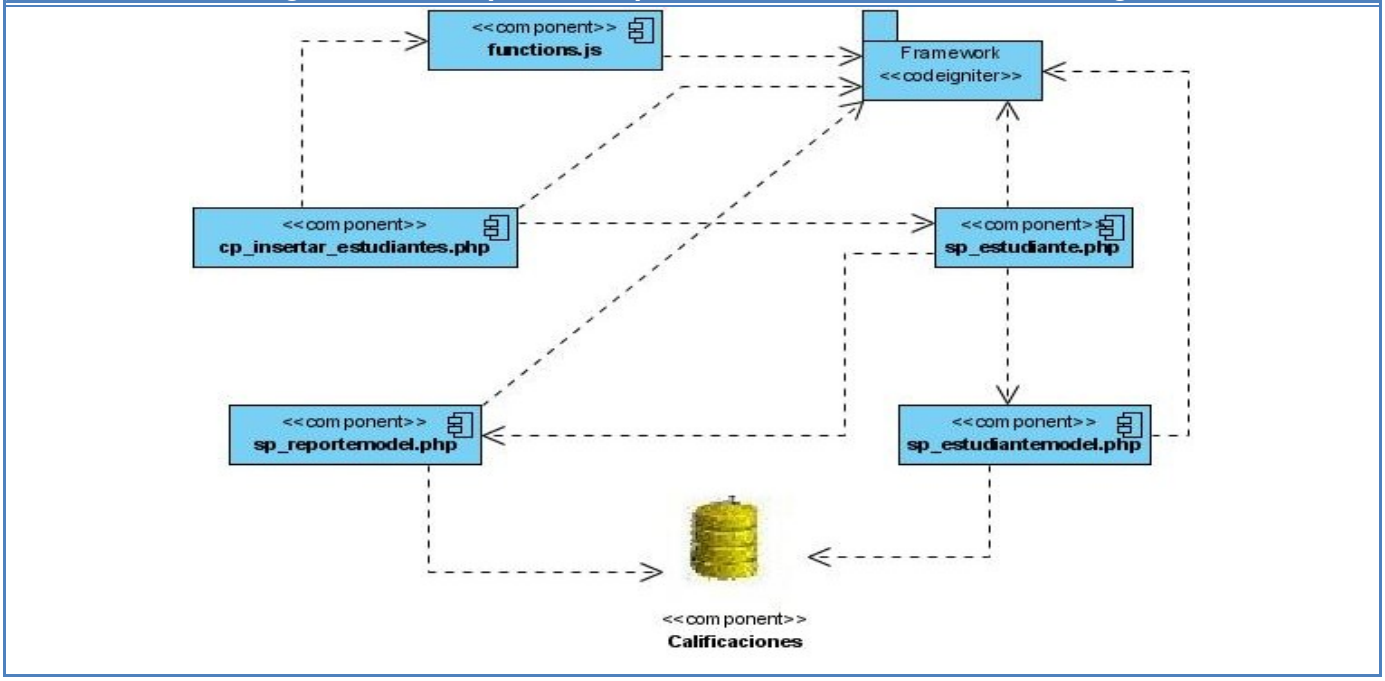

Figura 51 Diagrama de Componentes del Case de Uso Insertar_Codigo.

Diagrama de Componentes para el Caso de Uso Confeccionar_Reporte

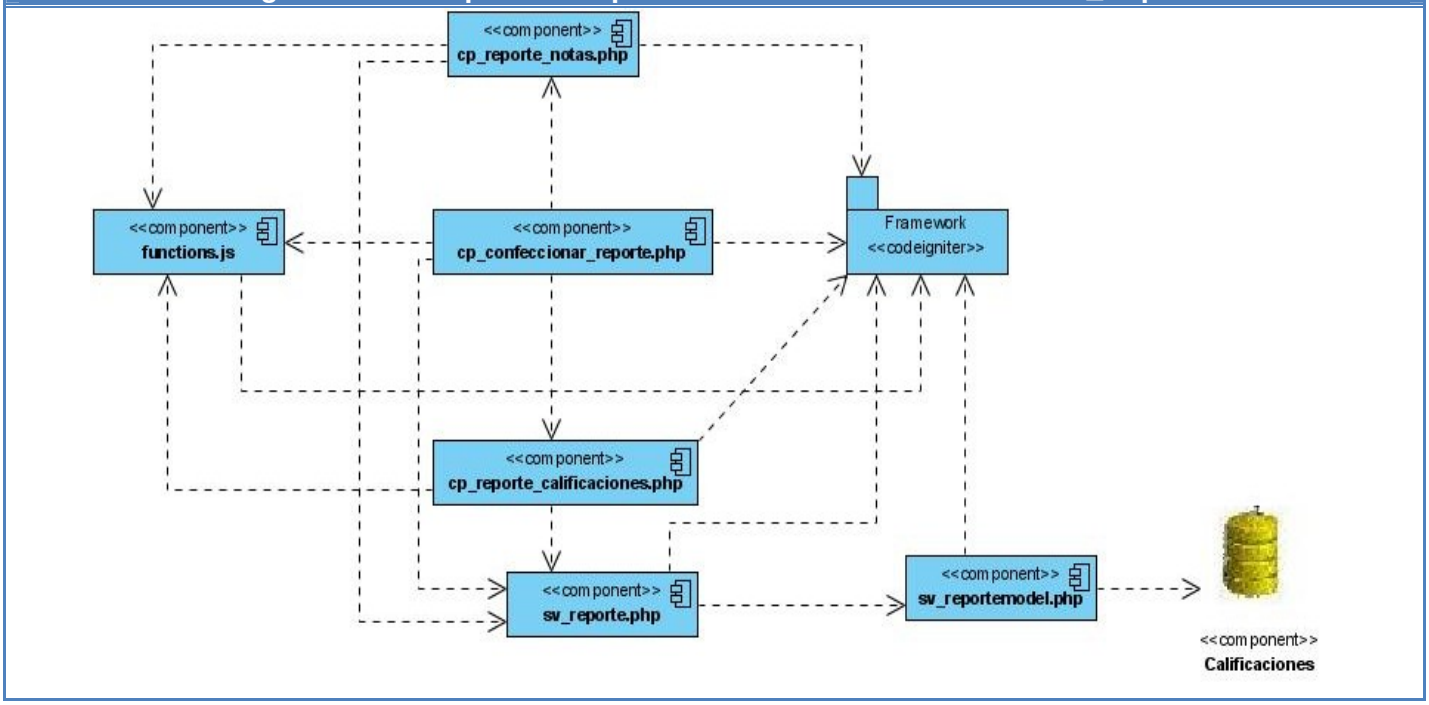

Figura 52 Diagrama de Componentes del Case de Uso Confeccionar_Reporte.

Diagrama de Componentes para el Caso de Uso Crear_Nuevo_Curso

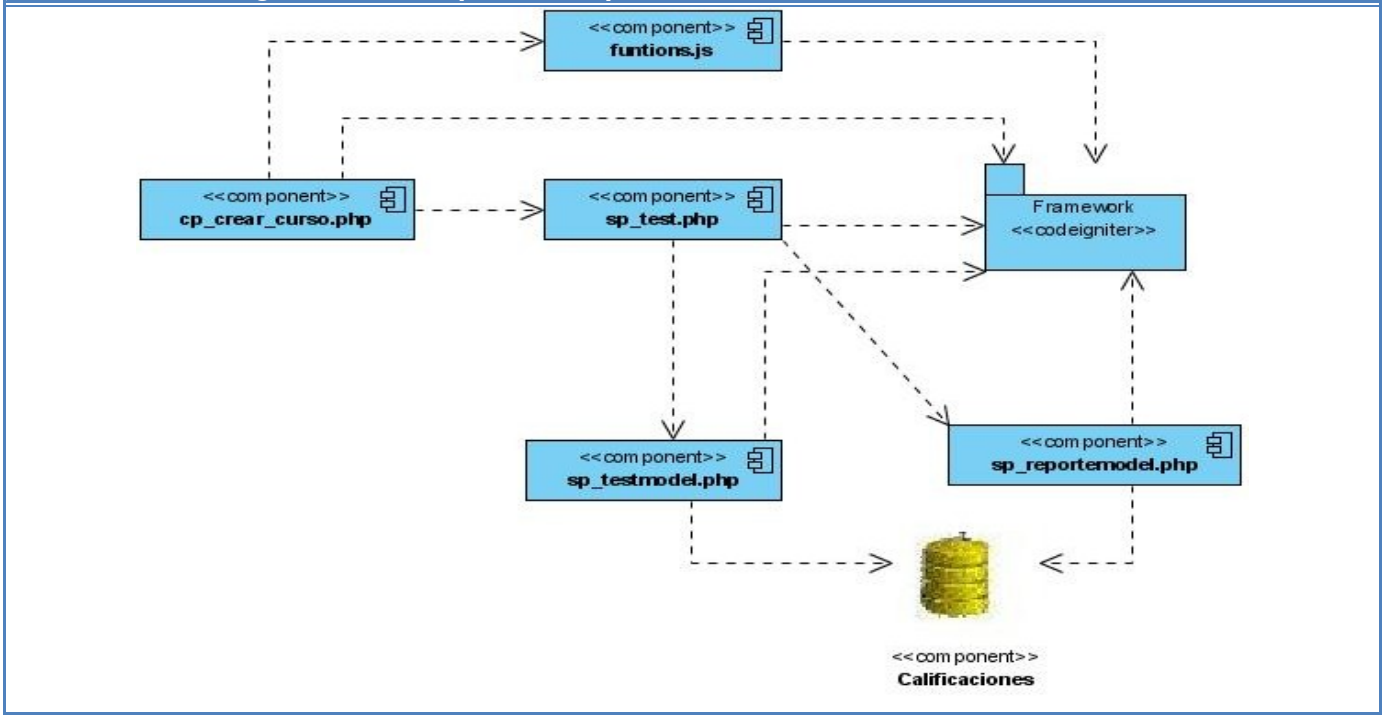

Figura 53 Diagrama de Componentes del Case de Uso Crear_Nuevo_Curso.

Diagrama de Componentes para el Caso de Uso Crear_Nuevo_Test

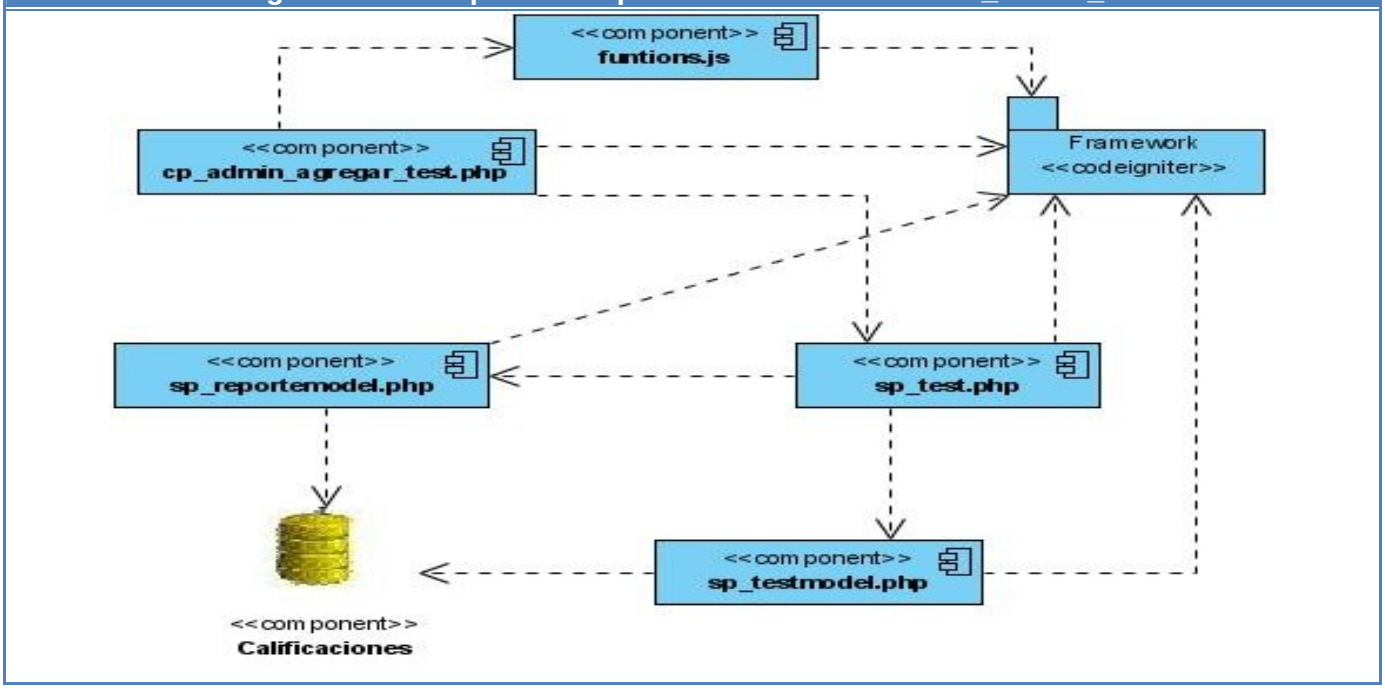

Figura 54 Diagrama de Componentes del Case de Uso Crear_Nuevo_Test.

4.2 Conclusiones

En este capítulo se trató el flujo de trabajo Implementación en el mismo de definió: un diagrama de despliegue y diecisiete diagramas de componentes. Al concluir se completan los artefactos que son la base de la aplicación del Sistema de Calificación de Pruebas de Ingreso a la UCI, se han cumplido los requerimientos planteados en el flujo de trabajo Levantamiento de requisitos.

CONCLUSIONES

Al inicio del presente trabajo de diploma de definieron como objetivos: desarrollar el análisis, diseño e implementación de los cinco módulos que conforman la aplicación. Además de crear los manuales de usuario necesarios para el uso del sistema. Al finalizar, se puede concluir que estos objetivos se cumplieron satisfactoriamente.

1. Para lograr un buen entendimiento entre usuarios y desarrolladores de identificó un modelo del dominio el cual recoge los conceptos fundamentales del proyecto.
2. A partir de un riguroso levantamiento de requisitos se obtuvieron dieciséis casos de uso del sistema, agrupados en cinco paquetes. Se definieron además seis actores del sistema con sus respectivas funcionalidades.
3. Se obtuvo el análisis de la aplicación, descrito en los dieciséis diagramas de clases de análisis que agrupan todas las funcionalidades que tiene el sistema. Basándose en esto se realizó un diseño especificando diecisiete diagramas de clases del diseño y veintisiete diagramas de secuencia.
4. Queda diseñada la base de datos para el almacenamiento de la información, la misma cuenta con trece tablas.
5. Utilizando el diseño como base se definieron diecisiete diagramas de componentes y un diagrama de despliegue conformando estos el flujo de trabajo de implementación.
6. Se creó una documentación de ayuda que le facilita al usuario una mejor interactividad con la aplicación.

A lo largo de toda la investigación quedaron descritos todos los artefactos necesarios para que finalmente se obtuviera una aplicación que garantiza la gestión de las calificaciones de los test ingreso a la UCI.

RECOMENDACIONES

Se recomienda:

- Mejorar el diseño de la aplicación.
- Realizar la generación de códigos de anonimato de manera tal que se le asigne cada uno, al carnet de identidad de cada aspirante.
- Poner a prueba el sistema durante un período, para comprobar que cumpla con las funcionalidades esperadas.

REFERENCIAS BIBLIOGRÁFICAS.

1. **Martín, Raquel San.** Universidad Nacional de San Luis. *Universidad Nacional de San Luis.* [En línea] Universidad Nacional de San Luis, 2006. [Citado el: 13 de Mayo de 2007.] http://www.ingreso.unsl.edu.ar/news/07_guia%20ingreso.html.

2. **Universidad de Chile.** Sistemas-Ingreso-Universidades-America-Latina. *Sistemas-Ingreso-Universidades-America-Latina.* [En línea] Universidad de Chile, 22 de Abril de 2007. [Citado el: 13 de Mayo de 2007.] http://www.universia.cl/portada/actualidad/noticia_119996.html.

3. **Dirección de Informatización de la Universidad de las Ciencias Informáticas.** UDDI UCI. *UDDI UCI.* [En línea] 9 de Mayo de 2007. [Citado el: 17 de Mayo de 2007.] http://uddi.uci.cu/arquitectura.2007.5.9.pdf.

4. **Gracia, Joaquin.** WebEstilo. *WebEstilo.* [En línea] 1998-2004. [Citado el: 16 de Febrero de 2007.] http://www.webestilo.com/javascript/.

5. **Interactive Programmers Community.** La Web del Programador. *La Web del Programador.* [En línea] 2000. [Citado el: 16 de Febrero de 2007.] http://www.lawebdelprogramador.com/diccionario/mostrar.php?letra=J .

6. **Guiarte Multimedia S.L.** desarrolloweb.com. *desarrolloweb.com.* [En línea] Guiarte Multimedia S.L. [Citado el: 17 de Febrero de 2007.] http://www.desarrolloweb.com/articulos/25.php.

7. **Interactive Programmers Community .** La Web del Programador. *La Web del Programador.* [En línea] 2000. [Citado el: 17 de Febrero de 2007.] http://lawebdelprogramador.com/diccionario/mostrar.php?letra=H&pagina=3.

8. **Espaweb Internet S.L.** Espaweb Internet . *Espaweb Internet S.* [En línea] Espaweb Internet S.L, 1997. [Citado el: 18 de Febrero de 2007.] http://www.espaweb.com/respuestas_online/HTML.html.

9. **Interactive Programmers Community .** La WEb del Programador. *La WEb del Programador.* [En línea] Interactive Programmers Community , 2000. http://www.lawebdelprogramador.com/diccionario/buscar.php?cadena=css.

10. **—** La Web del Programador. *La Web del Programador.* [En línea] 2000. [Citado el: 9 de Febrero de 2007.] http://lawebdelprogramador.com/diccionario/buscar.php?cadena=php .

11. **Espaweb Internet S.L.** Espaweb Internet . *Espaweb Internet .* [En línea] Espaweb Internet, 1997. [Citado el: 15 de Febrero de 2007.] http://www.espaweb.com/respuestas_online/PHP.html.

12. **ascii.** Asociación para el Conocimiento y la innovación de la informática. *Asociación para el Conocimiento y la innovación de la informática.* [En línea] Asociación para el Conocimiento y la Innovación

de la Informática (AsCII). [Citado el: 16 de Febrero de 2007.] http://ascii.eii.us.es/docs/2002-03/php/php4.html.

13. **Lago, Ramiro.** Proactiva Calidad. *Proactiva Calidad.* [En línea] Abril de 2007. [Citado el: 19 de Febrero de 2007.] http://www.proactiva-calidad.com/java/patrones/mvc.html.

14. **Maestros del Web.** Maestros del Web. *Maestros del Web.* [En línea] [Citado el: 19 de Febrero de 20007.] http://www.maestrosdelweb.com/editorial/zendstudio/ .

15. **Cecconi, Pablo.** Aplicaciones Web Espaciales con Software Libre. *Aplicaciones Web Espaciales con Software Libre.* [En línea] [Citado el: 16 de Febrero de 2007.] http://mapa.buenosaires.gov.ar/sig/AplicacionesWebEspacialesConSoftLibre.html.

16. **Saorín, Antonio, y otros.** [En línea] 12 de Marzo de 2003. [Citado el: 20 de Febrero de 2007.] http://es.tldp.org/Tutoriales/doc-servir-web-escuela/sirviendo-web-escuela.pdf.

17. **Linux Para Todos.** Linux Para Todos. *Linux Para Todos.* [En línea] 1 de Marzo de 2005. [Citado el: 17 de Febrero de 2007.] http://www.linuxparatodos.net/geeklog/staticpages/index.php?page=servidor-web.

18. **Linalco.** Linalco Especialistas en Linux y Software Libres. *Linalco Especialistas en Linux y Software Libres.* [En línea] Linalco. [Citado el: 20 de Febrero de 2007.] http://www.linalco.com/apache.html.

19. **Mendoza Sanchez, María A.** Informatizate. *www.informatizate.net.* [En línea] 7 de Junio de 2004. [Citado el: 17 de febrero de 2007.] http://www.informatizate.net/articulos/metodologias_de_desarrollo_de_software_07062004.html.

20. **Jacobson, I, Booch, G y Rumbaugh, J.** *El Proceso Unificado de Desarrollo de Software.* 2004. pág. 13. Ciudad de la Habana : Felix Varela, 2004. I-S-B-N.

21. **Popkin Software and Systems.** TLDP-ES/LuCAS. *TLDP-ES/LuCAS.* [En línea] 2007. [Citado el: 4 de mayo de 2007.] http://es.tldp.org/Tutoriales/doc-modelado-sistemas-UML/multiple-html/index.html.

22. **Alex Solano.** Netveloper. *Netveloper.* [En línea] 26 de Septiembre de 2002. [Citado el: 20 de Febrero de 2007.] http://www.netveloper.com/contenido2.aspx?IDC=64_0.

23. **Jacobson, Ivar, Booch, Grady y Rumbauch, James.** *El Proceso Unificado de Desarrollo de. Pág. 167.* Ciudad de La Habana : Felix Varela, 2004. I-S-B-N.

24. **Fernández Vilas, Ana.** [En línea] 20 de Marzo de 2001. [Citado el: 28 de Mayo de 2007.] http://www-gris.det.uvigo.es/~avilas/UML/node50.html.

BIBLIOGRAFÍA.

1. **Jacobson, Ivar, Booch, Grady y Rumbauch, James.** *El Proceso Unificado de Desarrollo de.* Ciudad de La Habana : Felix Varela, 2004. I-S-B-N.
2. Framework CodeIgniter. **Díaz Terrero, Eliurkis**. La Habana : s.n., 2007.
3. *El modelo de dominio como solución a la representación de entornos organizacionales difusos y complejos.* **Ciudad Ricardo, Ing. Febe Ángel.** La Habana : s.n., 2007.
4. **Code Igniter.** Code Igniter User Guide Version 1.5.0.
5. **CodeIgniter.** CodeIgniter. [En línea] http://www.codeigniter.com/
6. **Larman, Craig.** *UML y patrones.* Prentice Hall Iberoamericana : Féliz Varela, 1999. Capítulos 13, 16, 17, 18, 19, 21, 34 y 35. Vol. 1.
7. **Dirección de Informatización de la Universidad de las Ciencias Informáticas.** UDDI UCI. [En línea]. http://uddi.uci.cu/arquitectura.2007.5.9.pdf.
8. **La Web del Programador.** [En línea]. http://www.lawebdelprogramador.com/diccionario.
9. **Asociación para el Conocimiento y la innovación de la informática.** [En línea] http://ascii.eii.us.es/docs/2002-03/php/php4.html
10. **Lago, Ramiro.** Proactiva Calidad [En línea]. http://www.proactiva-calidad.com/java/patrones/mvc.html
11. **Mendoza Sanchez, María A.** Informatizate. [En línea]. http://www.informatizate.net/articulos/metodologias_de_desarrollo_de_software_07062004.html
12. **Generator FD**. [En línea] 2005. http://www.generatorfd.com/Arquitectura.aspx.
13. **msdn.** [En línea] 2007. http://www.microsoft.com/spanish/msdn/comunidad/mtj.net/voices/art140.asp
14. **Sitio php en catellano**. [Online] 2006. http://www.programacion.net/php/
15. **Proactiva calidad**. [Online] Abril 2007. http://www.proactiva-calidad.com/java/patrones/mvc.htmlinformacion .
16. **AKELARREWEB**. [Online] 2007. http://www.akelarreweb.com/modelo-vista-controlador
17. Monografias.com. [Online] http://www.monografias.com/trabajos14/educ-cuba/educ-cuba.shtml#ALGUNAS.
18. **Revista la Universidad**. [Online] Abril 2007. http://www.unsj.edu.ar/revista/Revista%20U/revista27/entrevistas.htm

19. **Sentido web**. [Online] Abril 2007. http://sentidoweb.com/2007/04/24/codeigniter-framework-para-php.php.
20. **Visual-Paradigm**. [Online] 2005. http://www.visual-paradigm.com/product/vpuml/
21. **version cero**. [Online] http://www.versioncero.com/noticia/210/visual-paradigm-for-uml.

ANEXO I: CASOS DE USO POR CICLO.

Cód.	Nombre de caso de uso	Paquete	Justificación de la selección.
CU-1	Autenticarse	Autenticación de Usuario.	Asociado con el control de la seguridad del sistema.

Tabla 2 Ciclo de desarrollo del paquete Autenticación de Usuario.

Cód.	Nombre de caso de uso	Paquete	Justificación de la selección.
CU-2	Insertar_Calificacion	Introducción de la Información.	Asociado con la necesidad de introducir toda la información de las calificaciones.
CU-3	Modificar_Calificacion		

Tabla 3 Ciclo de desarrollo del paquete Introducción de la Información.

Cód.	Nombre de caso de uso	Paquete	Justificación de la selección.
CU-4	Agregar_Rol	Administración del Sistema.	Asociado con toda la administración del sistema, incluye administración de usuarios y administración de toda la información referente a los códigos de anonimato.
CU-5	Asignar_Rol		
CU-6	Gestionar_Usuario		
CU-7	Buscar_Usuario		
CU-8	Generar_Codigo		
CU-9	Insertar_Codigo		
CU-10	Listar_Codigo		

Tabla 4 Ciclo de desarrollo del paquete Administración del Sistema.

Cód.	Nombre de caso de uso	Paquete	Justificación de la selección.
CU-11	Confeccionar_Reporte	Reporte de la Información Procesada	Se asocian por tratar la necesidad de mostrar reportes a los clientes.
CU-12	Buscar_Codigo_Sin_Nota		
CU-13	Controlar_Log		

Tabla 5 Ciclo de desarrollo del paquete Reporte de la Información Procesada.

Cód.	Nombre de caso de uso	Paquete	Justificación de la selección.
CU-14	Crear_Nuevo_Curso	Gestionar Test de Ingreso	Asociado con la necesidad de crear nuevos test.
CU-15	Crear_Nuevo_Test		
CU-16	Listar_Curso		

Tabla 6 Ciclo de desarrollo del paquete Gestionar Test de Ingreso.

ANEXOS II: DESCRIPCIÓN DE LOS CASOS DE USOS CRÍTICOS.

.

Caso de uso		
CU-1	Autenticarse	
Propósito	Este caso de uso se realiza con el objetivo de garantizar que sólo entre al sistema el personal autorizado, y preservar así la seguridad de la información que se maneja.	
Actores Usuario_Sistema (autenticarse)		
Resumen: Este caso de uso se inicia cuando un usuario decide entrar al sistema, para esto introduce su usuario y contraseña, el sistema verifica que el usuario pertenezca a Ldap, si pertenece, verifica con la información que hay en la base de datos del sistema. Luego le permite el acceso al sistema al usuario autorizado.		
Referencias	R1	
Precondiciones:		
Poscondiciones:		
Curso Normal de los Eventos		
Acción del actor	**Respuesta del sistema**	
1. El usuario introduce su usuario y contraseña.	2. Verifica que el usuario y la contraseña pertenezcan a Ldap. Si pertenece verifica que el usuario esté en la base de datos del sistema. Sino realiza curso alterno de los eventos.	
	3. Comprueba si es correctos el usuario en la base de datos del sistema. Si lo es permite entrar al sistema, sino ejecuta el flujo alterno	
	4. El sistema muestra la interfaz según el rol que el usuario tiene definido en el sistema.	
Flujo alterno		
Acción del actor	**Respuesta del sistema**	
	3.1. Sistema muestra cartel " acceso denegado"	
Prioridad: Crítico		

Tabla 7 Descripción del Caso de Uso del Sistema Autenticarse.

Caso de uso	
CU-2	Insertar_Calificacion
Propósito	Este caso de uso se realiza con el objetivo de permitirle a un usuario calificador, entrar las calificaciones que obtuvo el estudiante. Esta información se almacena en la base de datos para usos posteriores. Además le permite al administrador realizar esta acción.
Actores: Usuario_Calificador. (inicia)	

Resumen: Este caso de uso se inicia cuando el Usuario_Calificador (generalización de actores: administrador, calificador IPI, calificador CPT, calificador PRE) entra al sistema. Si es un calificador de (IPI, CPT, PRE), la aplicación le muestra la interfaz que corresponde al test específico que él tiene asignado para calificar. Este especifica el código e inserta las calificaciones que obtuvo el estudiante en los test. Si el rol es de administrador el sistema muestra una interfaz para que este seleccione el curso-test. Una ves realizada esta acción se muestra habilitada una nueva interfaz con el test que el seleccionó y un menú para que seleccione el proceso, e introduzca el código y las calificaciones del estudiante.	
Referencias	R13
Precondiciones: Debe existir el código del estudiante en el sistema.	
Poscondiciones: Se afecta la información del sistema.	
Acción del actor	**Respuesta del sistema**
1. Selecciona insertar calificación.	2. Verifica el rol y si es calificador de (IPI, CPT, PRE) muestra habilitada la interfaz que corresponde al test específico que él tiene asignado para calificar. De ser administrador ejecuta sección para el administrador.
3. Especifica el código del estudiante que tiene las calificaciones que va a entrar en el sistema.	
4. Entra las calificaciones que obtuvo el estudiante en el test de ingreso.	5. Verifica si el código existe en la base de datos del sistema y sino esta siendo usado en ese proceso. De ser así permite entrar las calificaciones, y guarda toda la información en la base de datos. Sino existe, ejecuta el flujo alternativo (5.1).
	6. Si existe el código y esta siendo usado para ese proceso ejecuta el flujo alternativo (6.1).
Flujo alternativo	
Acción del actor	**Respuesta del sistema**
	5.1 Muestra mensaje “No se han insertado las calificaciones. No existe el Código”.
	6.1 Muestra un mensaje “No se han insertado las notas (Código:’’).Ya existe.”
Sección para el administrador	
Acción del actor	**Respuesta del sistema**
	3. Muestra una interfaz con un menú para seleccionar el curso-test.
4. Selecciona el curso-test y acepta.	5. Muestra una nueva interfaz con un menú para seleccionar el proceso, y el test seleccionado listo para entrar las calificaciones.
6. Selecciona el proceso.	
7. Especifica el código del estudiante.	
8. Introduce las calificaciones que obtuvo el estudiante en el examen.	9. Verifica si el código existe en la base de datos del sistema y sino esta siendo usado en ese proceso. De ser así permite entrar las calificaciones, y guarda toda la información en la base de datos. Sino existe, ejecuta el flujo alternativo(9.1)

	10.Si existe el código y esta siendo usado para ese proceso ejecuta el flujo alternativo(10.1)
Flujo alternativo	
Acción del actor	**Respuesta del sistema**
	9.1 Muestra mensaje “No se han insertado las calificaciones. No existe el Código”.
	10.1 Muestra un mensaje “No se han insertado las notas (Código:’’).Ya existe.”
Prioridad: Crítico	

Tabla 8 Descripción del Caso de Uso del Sistema Insertar_Calificacion.

Caso de uso	
CU-3	Modificar_Informacion
Propósito	Este caso de uso se crea con el propósito de permitirle al administrador hacer cambios en la información, es en este aspecto donde podrá cambiar calificaciones, reasignar calificaciones de un código a otro, es decir manejar la información ya almacenada en el sistema.
Actores: Administrador (Inicia)	
Resumen: Este caso de uso inicia cuando el administrador entra a la sección “Modificar Calificación”, aquí selecciona el curso y el número del test al que pertenece el estudiante que se le desea modificar las calificaciones, luego introduce el proceso al que pertenece este estudiante y el código. El sistema muestra las calificaciones de este código. El administrador hace los cambios que desea; al aceptar, estos se guardan automáticamente en la base de datos. Puede realizar la misma acción si lo desea es ver las calificaciones de un código determinado, y no realizar cambios. Además si una vez mostrada las calificaciones cambia el código y da actualizar se hace una reasignación de calificaciones de un código a otro.	
Referencias	R14
Precondiciones: Deben existir las calificaciones del código especificado en el sistema.	
Poscondiciones: Se afecta la información del sistema.	
Acción del actor	**Respuesta del sistema**
1. Entre a la sección Modificar Calificación.	2. El sistema le muestra la interfaz para escoger el curso y el número del test.
3. Selecciona el curso-test en el que desea modificar las calificaciones.	4. Muestra la interfaz de modificar habilitada..
5. Selecciona el proceso en el que desea trabajar.	
6. Introduce el código en el que desea realizar las modificaciones. Pulsa la opción “Buscar”	
	7. Verifica si para ese proceso, ese código existe en la base de datos. Si existe muestra la información que hay sobre él en la base de datos, todo listo para hacer el cambio. Sino ejecuta el curso alternativo de los eventos.

8. Realiza los cambios que desea en la información	
	9. Guarda automáticamente en la Base de Datos los cambios efectuados
Flujo alternativo	
Acción del actor	**Respuesta del sistema**
	7.1. Muestra mensaje “El código no existe”
Prioridad: Crítico	

Tabla 9 Descripción del Caso de Uso del Sistema Modificar_Calificacion.

Caso de uso	
CU-5	Asignar_Rol
Propósito	Este caso de uso se crea con el propósito de permitirle al administrador, asignarle a los usuarios del sistema, el rol que estos tendrán para acceder al mismo.
Actores: Administrador (inicia)	
Resumen: Este caso de uso inicia cuando el administrador entra a la sección “Asignar Rol”, donde introduce el login, el rol y el curso-test del nuevo usuario, el sistema verifica que pertenezca a Ldap y que no esté en la base de datos del sistema, de ser así almacena la información en la base de datos, sino muestra mensajes.	
Referencias	R2
Precondiciones: El usuarios debe existir en el dominio y el sistema debe contar con roles definidos.	
Poscondiciones: Se afecta la información en el sistema.	
Acción del actor	**Respuesta del sistema**
1. Elige la opción “Asignar Rol”	2. Muestra interfaz para asignar rol habilitada.
3. Introduce el login del nuevo usuario.	
4. Selecciona el rol que debe tener el nuevo usuario.	
5. Selecciona el curso-test Acepta esta acción.	7. Verifica que el usuario pertenezca a Ldap, si pertenece, comprueba que no exista en la base de datos del sistema, de ser así inserta el nuevo usuario, sino, ejecuta flujo alternativo (7.1).
	8. Si existe en el Ldap y en la base de datos del sistema, muestra mensaje “El usuario ya existe”.
Flujo alternativo	
	7.1 Muestra mensaje “Error: El usuario debe existir en el dominio”.
Prioridad: Crítico	

Tabla 10 Descripción del Caso de Uso del Sistema Asignar_Rol.

Caso de uso	
CU-8	Generar_Codigo
Actores: Administrador (inicia)	
Propósito	Este caso de uso se realiza con el propósito de que el administrador pueda cargar en la base de datos la lista de códigos que se utilizarán en el proceso de calificación sin necesidad de pasarlos de forma manual.

Resumen: Este caso de uso inicia cuando el administrador entra al sistema y elige la opción "Generar Código". El sistema muestra una interfaz con dos menú donde aparecen los tipos de estudiantes y curso-test que existen, y da la opción de especificar la cantidad de códigos que desea generar. El administrador especifica todos los datos y el sistema ejecuta un algoritmo para generar los códigos de anonimato, automáticamente se cargan en la base de datos la cantidad de códigos que él definió para el tipo de estudiante y el curso-test que seleccionó.	
Referencias	R9
Precondiciones: El sistema debe tener definido tipos de estudiantes, curso y test.	
Poscondiciones: Se afecta la información en el sistema, se han insertado nuevos códigos.	
Acción del actor	**Respuesta del sistema**
1. Accede a la opción "Generar Códigos."	2. Muestra interfaz de la opción "Generar Códigos." habilitada
3. Selecciona el proceso en el que quiere insertar los códigos.	
4. Selecciona el curso y el test en el que quiere insertar los códigos.	
5. Especifica la cantidad de estos que desea.	6. Comprueba que el proceso seleccionado en ese curso y ese test, aún tiene capacidad para generar códigos, si hay capacidad ejecuta un algoritmo que genera la cantidad de códigos deseados por el usuario y los inserta en la base de datos.
	7. Sino existe la capacidad muestra mensaje informando la situación.
Flujo alternativo	
Acción del actor	**Respuesta del sistema**
Prioridad: Crítico	

Tabla 11 Descripción del Caso de Uso del Sistema Generar_Codigo.

Caso de uso	
CU-9	Insertar_Codigo
Actores: Administrador (Inicia)	
Propósito	Este Caso de uso se crea con el propósito de permitirle al administrador agregar un Código, e inmediatamente insertarle calificaciones si fuera necesario, esto se utiliza generalmente cuando la primera parte de introducción de información se hizo, y surgen nuevos estudiantes que entrar al sistema.
Resumen: Este caso de uso inicia cuando el administrador entra al sistema y elige la opción "Insertar Nuevo Código" el sistema le muestra una interfaz lista para realizar esta acción. El administrador elige el curso, el proceso en el que desea insertar el código y luego introduce este. Cuando marca la opción guardar, el sistema verifica que el código con ese curso-test y ese proceso no esté en la base de datos, sino está inserta el nuevo código y se muestra un mensaje "Se ha insertado el código del estudiante", si está muestra mensaje "El estudiante no pudo ser insertado, el código ya existe".	
Referencias	R10
Precondiciones: El sistema debe tener definido tipos de estudiantes, curso y test.	
Poscondiciones: Se afecta la información en el sistema.	
Acción del actor	**Respuesta del sistema**

1. Elije la opción Insertar Nuevo Código.	2. Muestra la interfaz Insertar Nuevo Código habilitada.
3. Elige el proceso en el que desea hacer la inserción.	
4. Elige el curso-test en el que desea hacer la inserción.	
5. Especifica el nuevo código y acepta.	6. Verifica si el código insertado no está en la base de datos siendo usado en ese proceso y en ese curso-test, de ser así inserta el nuevo código y muestra mensaje "Se ha insertado el código del estudiante".
	7. Sino muestra mensaje "El estudiante no pudo ser insertado, el código ya existe".
Flujo alternativo	
Acción del actor	**Respuesta del sistema**
Prioridad: Crítico.	

Tabla 12 Descripción del Caso de Uso del Sistema Insertar_Codigo.

Caso de uso	
CU-11	Confeccionar_Reporte
Propósito	Este caso de uso se crea con el propósito de permitirle al administrador ver los reportes de notas y calificaciones que el sistema tiene almacenado en la base de datos. De esta manera podrá analizar la información para tomar decisiones posteriores si fuera necesario.
Actores: Administrador (Inicia)	
Resumen: Este caso de uso inicia cuando el administrador selecciona la opción "Confeccionar reporte", el sistema le muestra una interfaz en la que puede seleccionar el tipo de reporte deseado, de notas o calificaciones. Luego define el tipo de estudiante y el curso test para saber el reporte. Al aceptar el sistema busca la información en la base de datos y muestra el reporte, en caso de no existir la información, muestra un mensaje: "No existen datos para este reporte".	
Referencias	R16, R17
Precondiciones: Debe existir la información en el sistema.	
Poscondiciones: No se afecta la información en el sistema.	
Acción del actor	**Respuesta del sistema**
1. Selecciona la opción "Confeccionar reporte".	2. Muestra interfaz para confeccionar reporte habilitada.
3. Selecciona de que desea el reporte, si de notas o de calificaciones.	
4. Selecciona el curso-test, y el proceso del que desea el reporte	5. Verifica si existe en la base de datos la información solicitada. Si existe la información muestra el reporte solicitado.
	6. Sino muestra mensaje "No existen datos para este reporte"
Flujo alternativo	
Acción del actor	**Respuesta del sistema**

Prioridad: Crítico.

Tabla 13 Descripción del Caso de Uso del Sistema Confeccionar_Reporte.

Caso de uso	
CU-14	Crear_Nuevo_Curso
Propósito	Este caso de uso se crea con el propósito de permitirle al administrador crear un nuevo curso, y utilizar este en otras partes de la aplicación ejemplo para asignar rol es necesario definir el curso.
Actores: Administrador (Inicia)	
Resumen: Este caso de uso inicia cuando el administrador entra el la sección “Crear nuevo Curso” y el sistema le muestra una interfaz en la que puede realizar esta acción. Introduce el nuevo curso y acepta, el sistema verifica que no existe y guarda automáticamente la información en la base de datos del sistema, si ya existe muestra un mensaje” Ya existe el curso”	
Referencias	R19
Precondiciones:	
Poscondiciones: Se afectó la información del sistema.	
Acción del actor	**Respuesta del sistema**
1. Selecciona la opción “Crear nuevo Curso”	2. Muestra habilitada la interfaz que permite crear un nuevo curso.
3. Introduce el nuevo curso, y pulsa agregar.	4. Verifica que el curso no esté en la base de datos de ser así se inserta de forma correcta el nuevo curso. Muestra mensaje “Insertado correctamente”
	5. Si existe en la base de datos muestra mensaje “Ya existe el curso”
Flujo alternativo	
Acción del actor	**Respuesta del sistema**
Prioridad: Crítico	

Tabla 14 Descripción del Caso de Uso del Sistema Crear_Nuevo_Curso.

Caso de uso	
CU-15	Crear_Nuevo_Test
Propósito	Este caso de uso se crea con el propósito de permitirle al administrador crear un nuevo test con las características que desee, así siempre que sea necesario aplicar un nuevo test, el sistema será flexible, y permitirá la calificación del mismo
Actores: Administrador (inicio)	
Resumen: Este caso de uso inicia cuando el administrador entra en el sistema y selecciona la opción “Crear nuevo Test”, El sistema le muestra una interfaz en la que define el número del test, selecciona el curso, define la cantidad de preguntas normales que tendrá su test, y la cantidad de preguntas especificaciones. Luego marca la opción “Agregar” y el sistema le muestra una nueva interfaz con el número de la pregunta y le permite definir el valor (cantidad máxima que puede obtener un estudiante en la pregunta) y el peso (valor de ponderación que le dan a la pregunta los psicólogos); el sistema automáticamente guarda estos cambios, en la base de datos.	
Referencias	R18
Precondiciones: Deben existir cursos definidos en el sistema.	

Poscondiciones: Se afectó la información del sistema.	
Acción del actor	**Respuesta del sistema**
1. Entra a la sección de gestionar test. “Crear nuevo Test”,	2. Muestra la interfaz para crear nuevo test habilitada.
3. Define el numero del test el curso y la cantidad de preguntar normales, y especificaciones que tendrá el nuevo test. Selecciona Agregar.	4. Guarda automáticamente el cambio en la Base de Datos, y muestra habilitada una interfaz para editar las preguntas.
5. Define el valor, y el peso de cada pregunta y pulsa agregar.	6. Sistema guarda automáticamente esta información en la base de datos y muestra mensaje “Se editaron las preguntas satisfactoriamente”.
Flujo alternativo	
Acción del actor	**Respuesta del sistema**
Prioridad: Crítico	

Tabla 15 Descripción del Caso de Uso del Sistema Crear_Nuevo_Test.

ANEXOS III: DIAGRAMAS DE SECUENCIA DE LOS CASOS DE USO CRÍTICOS.

Diagrama de Secuencia para el Caso de Uso Autenticarse

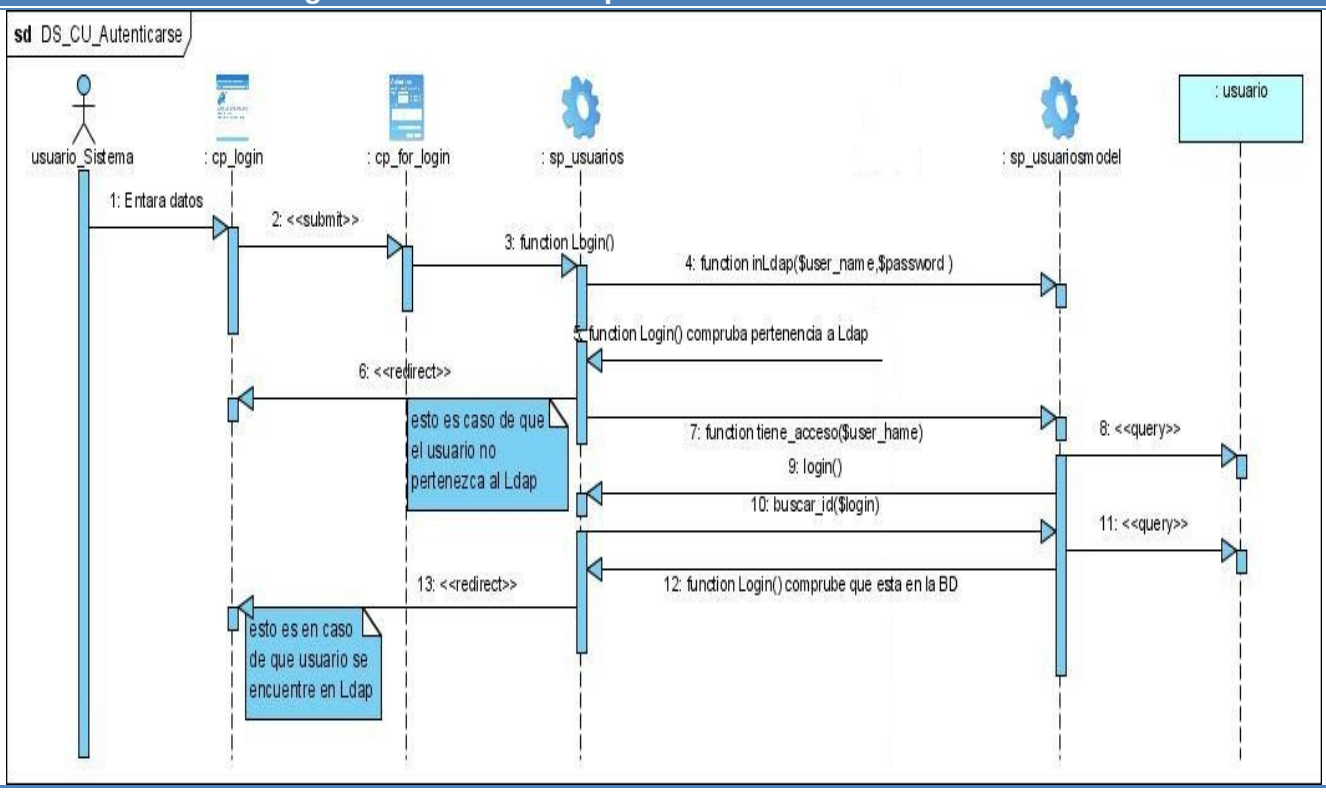

Figura 18 Diagrama de Secuencia del Caso de uso Autenticarse.

Diagrama de Secuencia para el Caso de Uso Insertar_Calificacion

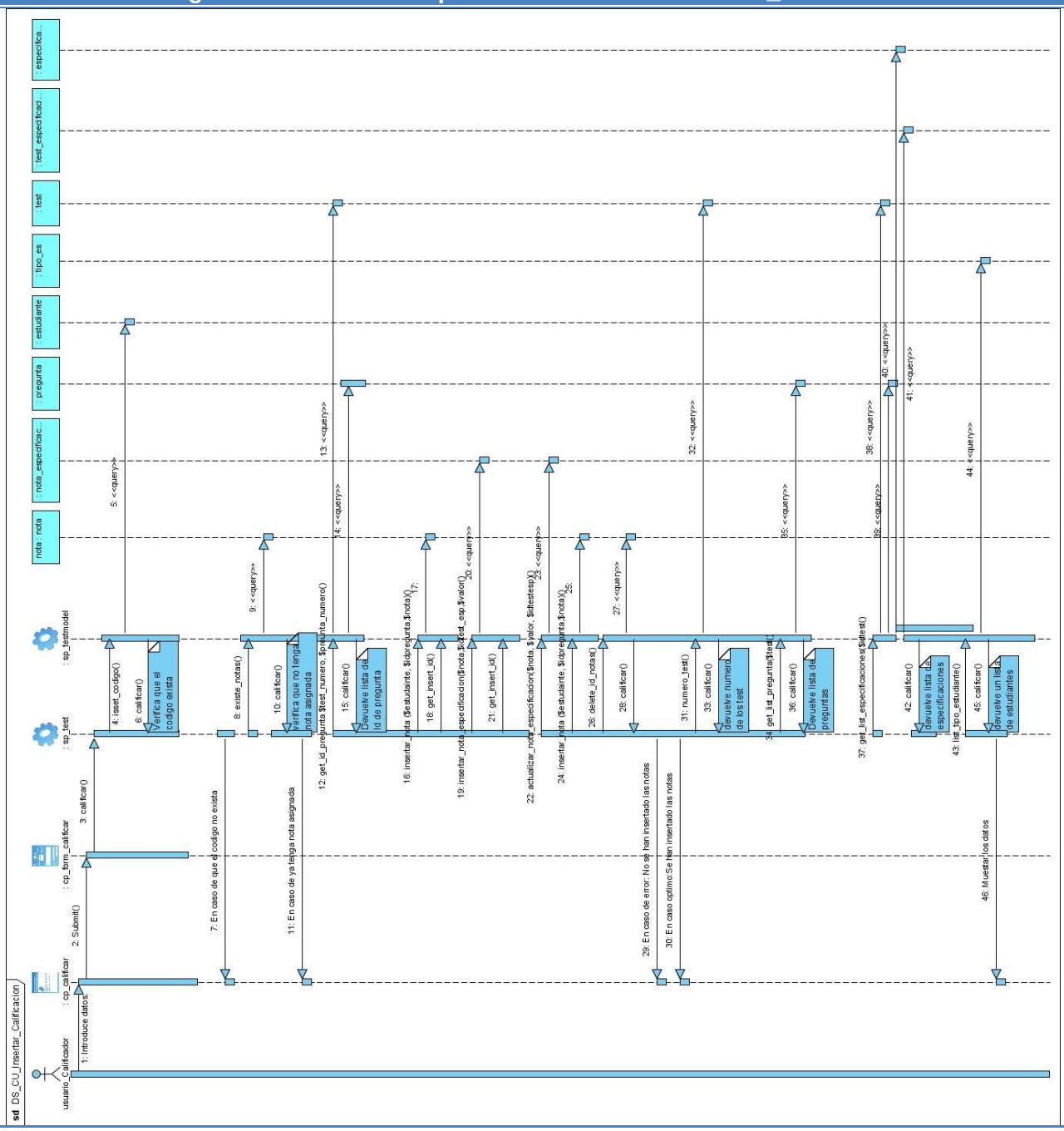

Figura 19 Diagrama de Secuencia del Caso de uso Insertar_Calificacion.

Diagrama de Secuencia para el Caso de Uso Insertar_Calificacion(Alternativa 1)

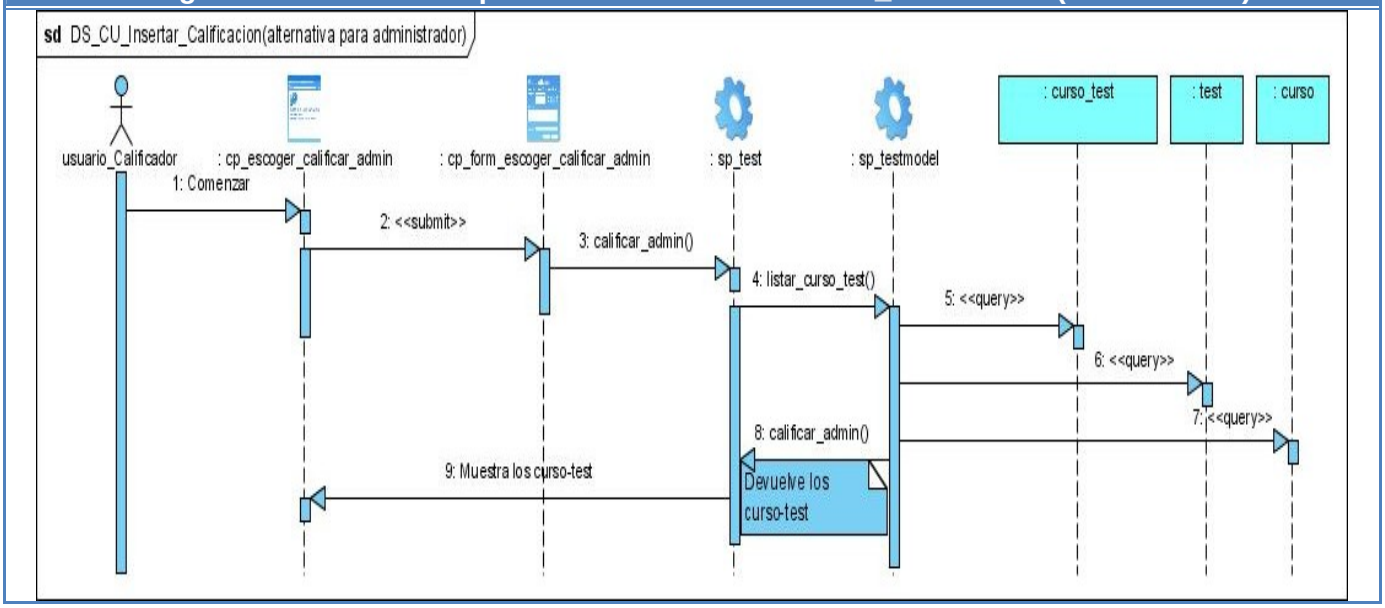

Figura 20 Diagrama de Secuencia del Caso de uso Insertar_Calificacion (Alternativa 1).

Diagrama de Secuencia para el Caso de Uso Modificar_Calificacion(Alternativa 1)

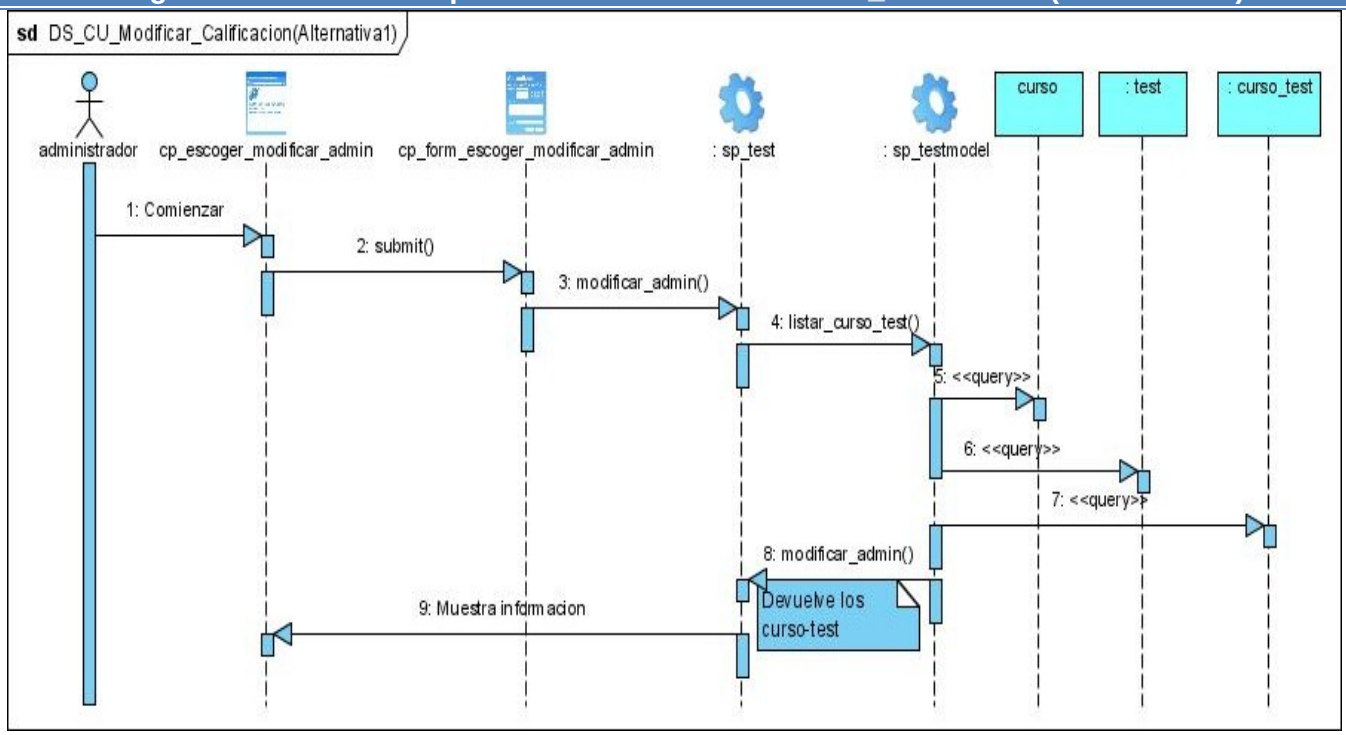

Figura 21 Diagrama de Secuencia del Caso de uso Modificar_Calificacion (Alternativa 1).

Diagrama de Secuencia para el Caso de Uso Modificar_Calificacion

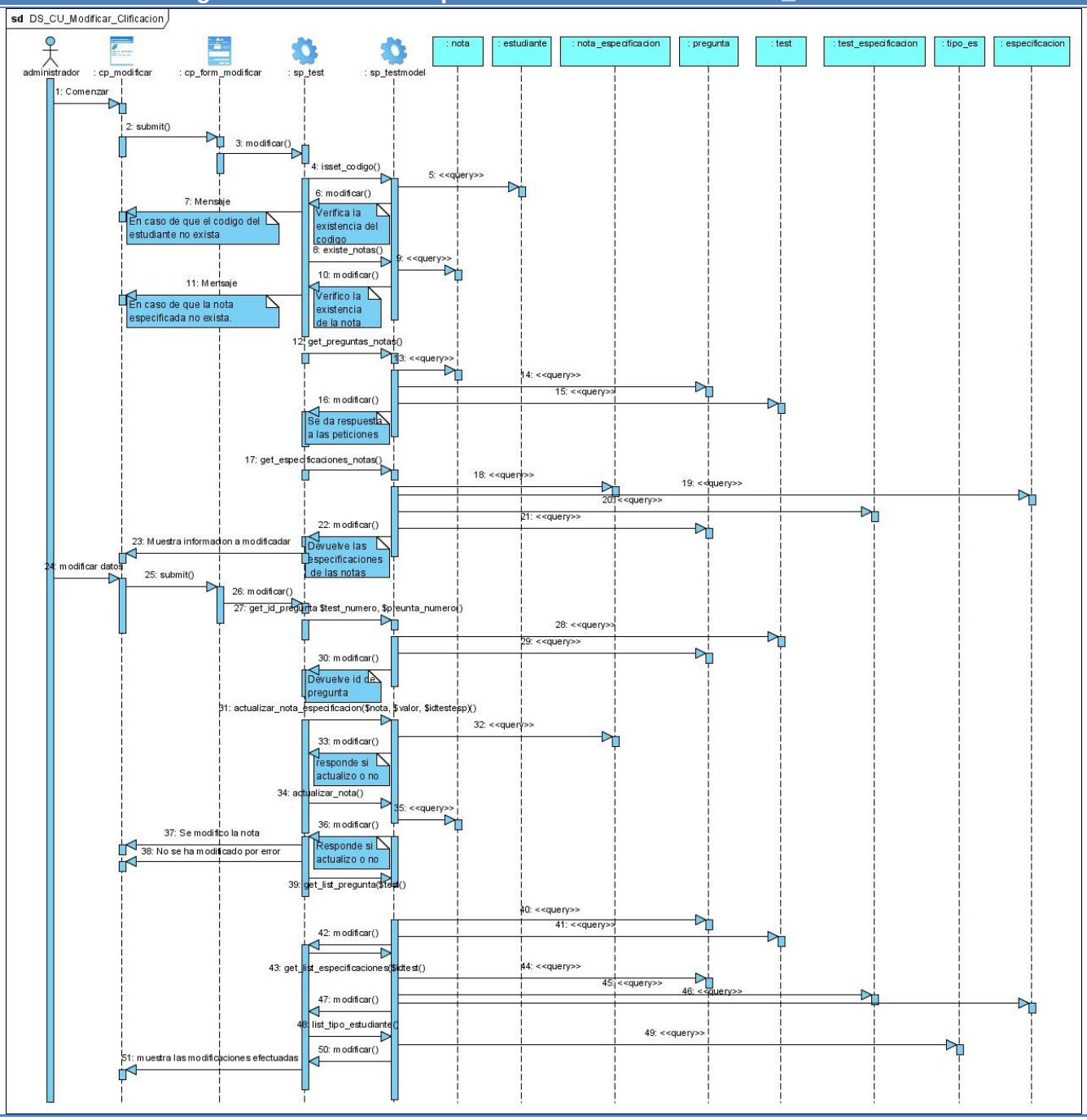

Figura 22 Diagrama de Secuencia del Caso de uso Modificar_Calificacion.

Diagrama de Secuencia para el Caso de Uso Asignar_Rol

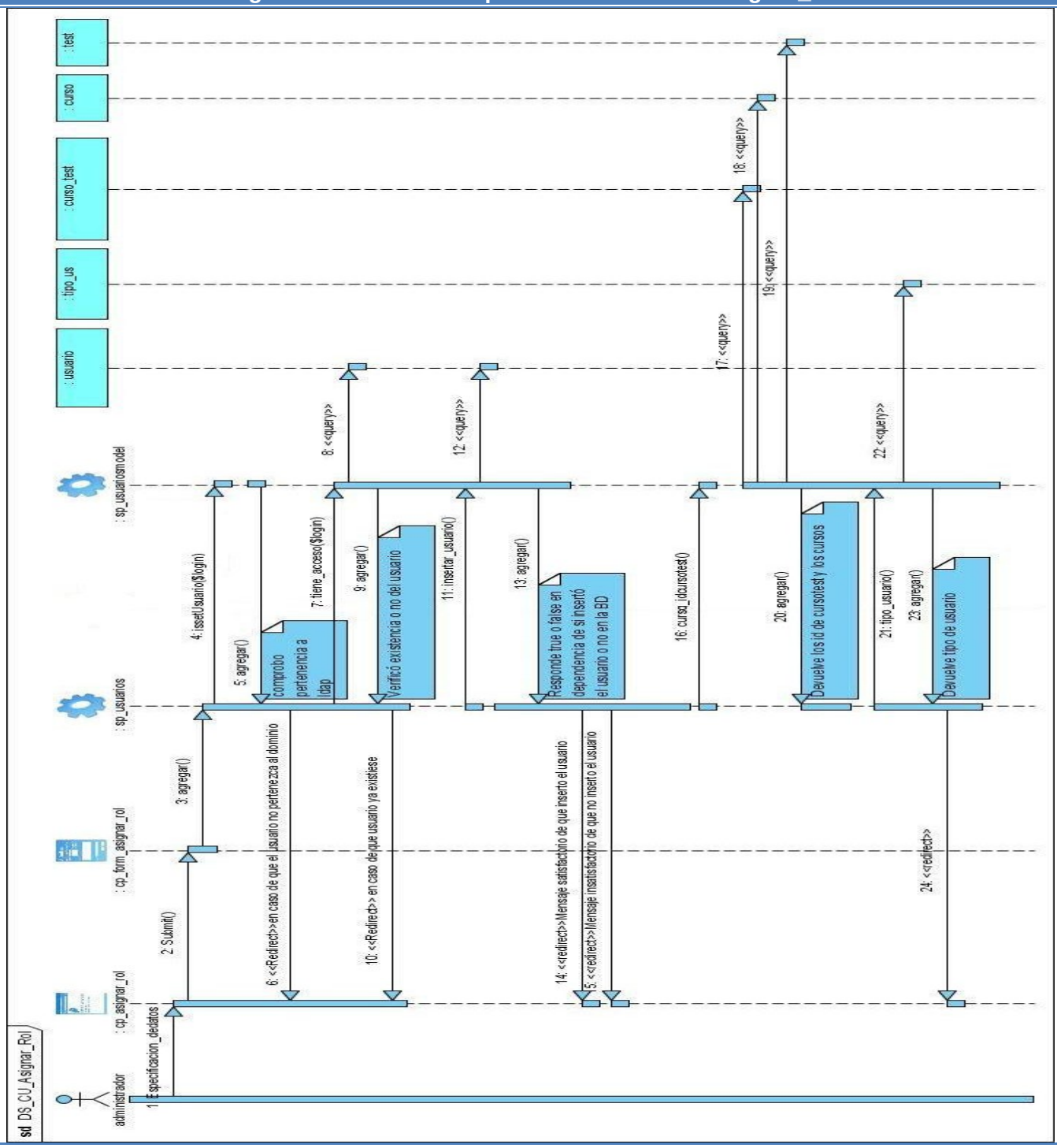

Figura 23 Diagrama de Secuencia del Caso de uso Asignar_Rol.

Diagrama de Secuencia para el Caso de Uso Generar_Codigo

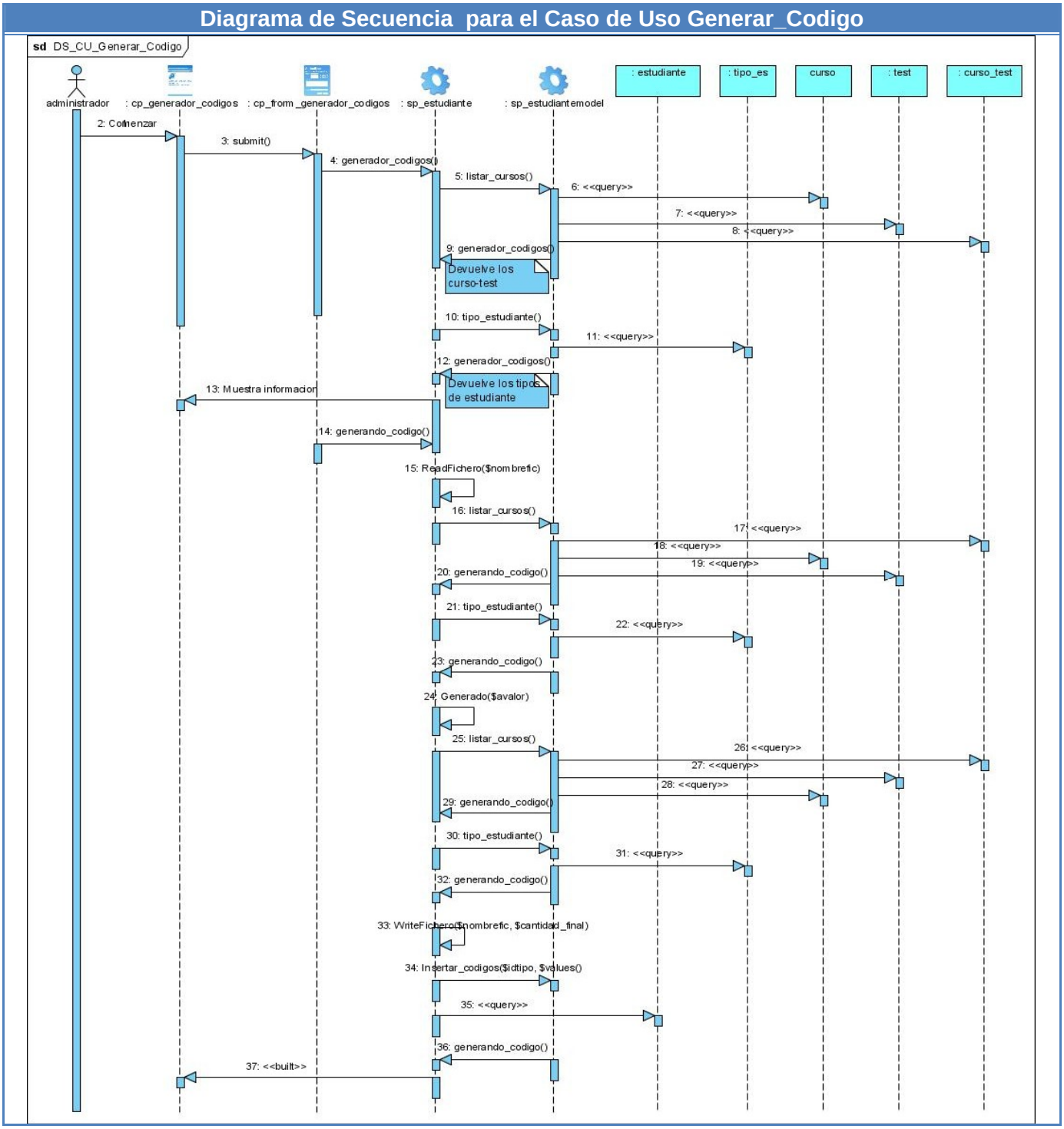

Figura 24 Diagrama de Secuencia del Caso de uso Generar_Codigo.

Diagrama de Secuencia para el Caso de Uso Insertar_Codigo

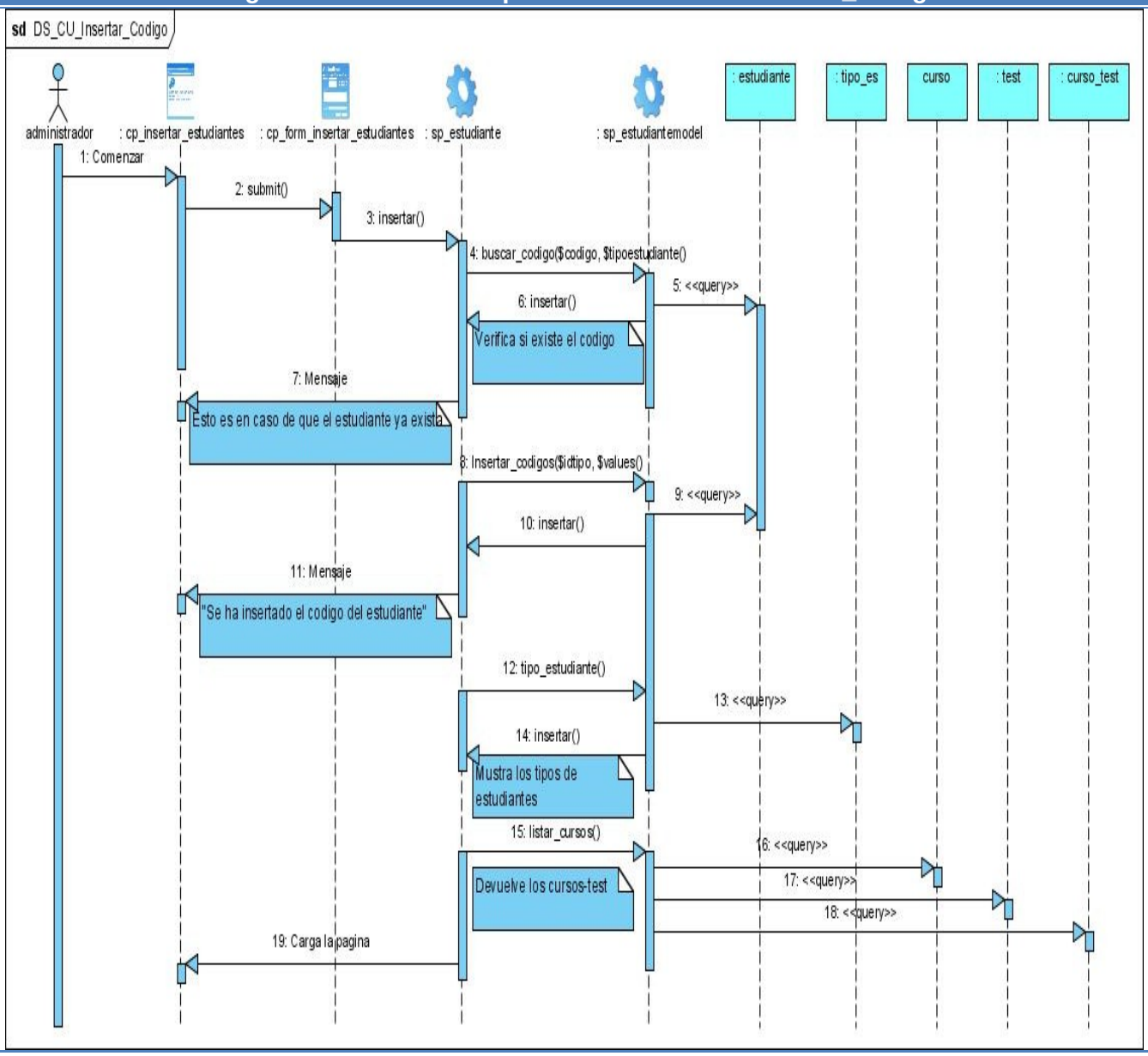

Figura 25 Diagrama de Secuencia del Caso de uso Insertar_Codigo.

Diagrama de Secuencia para el Caso de Uso Confeccionar_Reporte(Alternativa1)

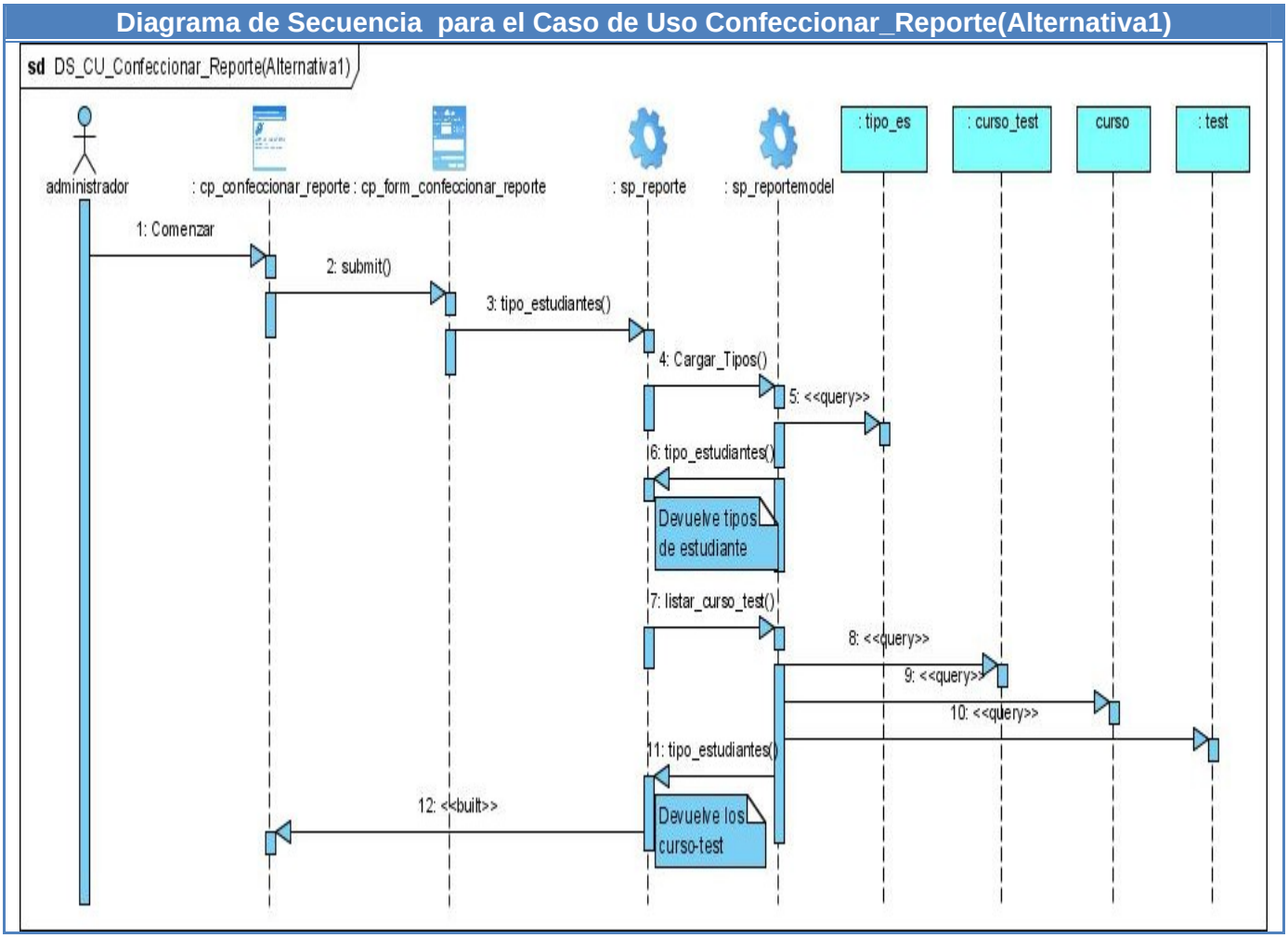

Figura 26 Diagrama de Secuencia del Caso de uso Confeccionar_Reporte (Alternativa 1).

Diagrama de Secuencia para el Caso de Uso Confeccionar_Reporte(para calificaciones)

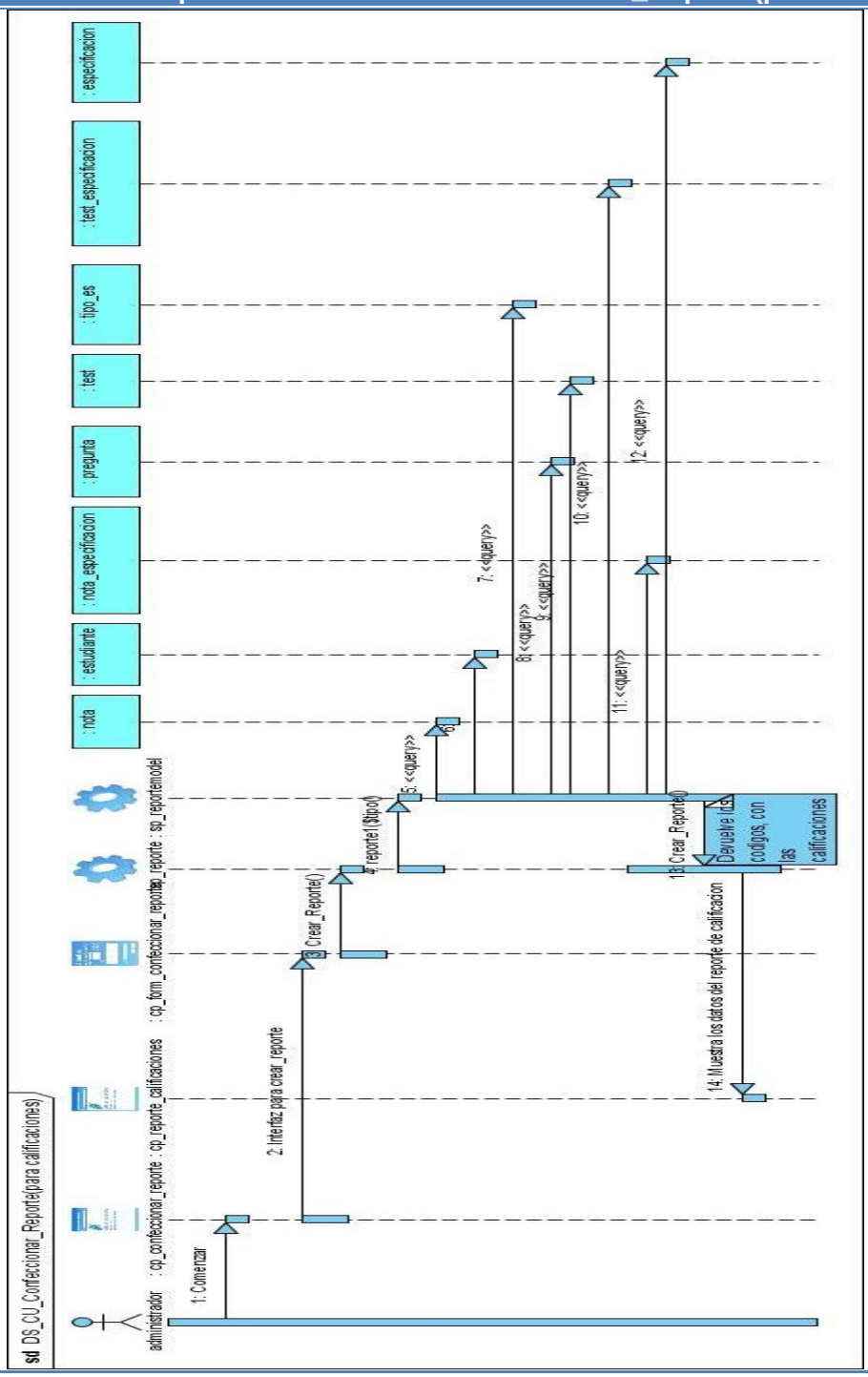

Figura 27 Diagrama de Secuencia del Caso de uso Confeccionar_Reporte (para calificaciones).

Diagrama de Secuencia para el Caso de Uso Confeccionar_Reporte(para notas)

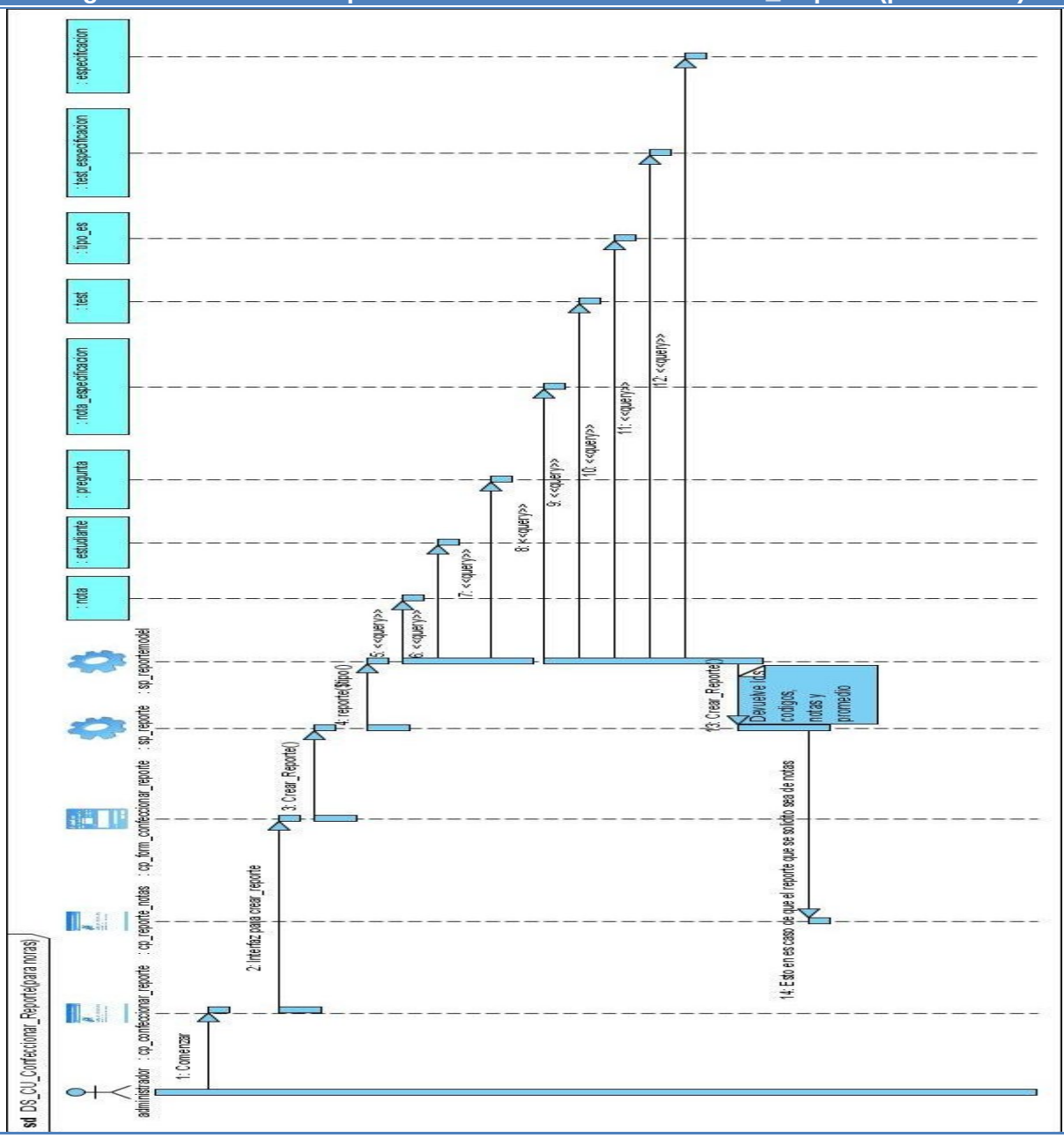

Figura 28 Diagrama de Secuencia del Caso de uso Confeccionar_Reporte (para notas).

Diagrama de Secuencia para el Caso de Uso Crear_Nuevo_Curso

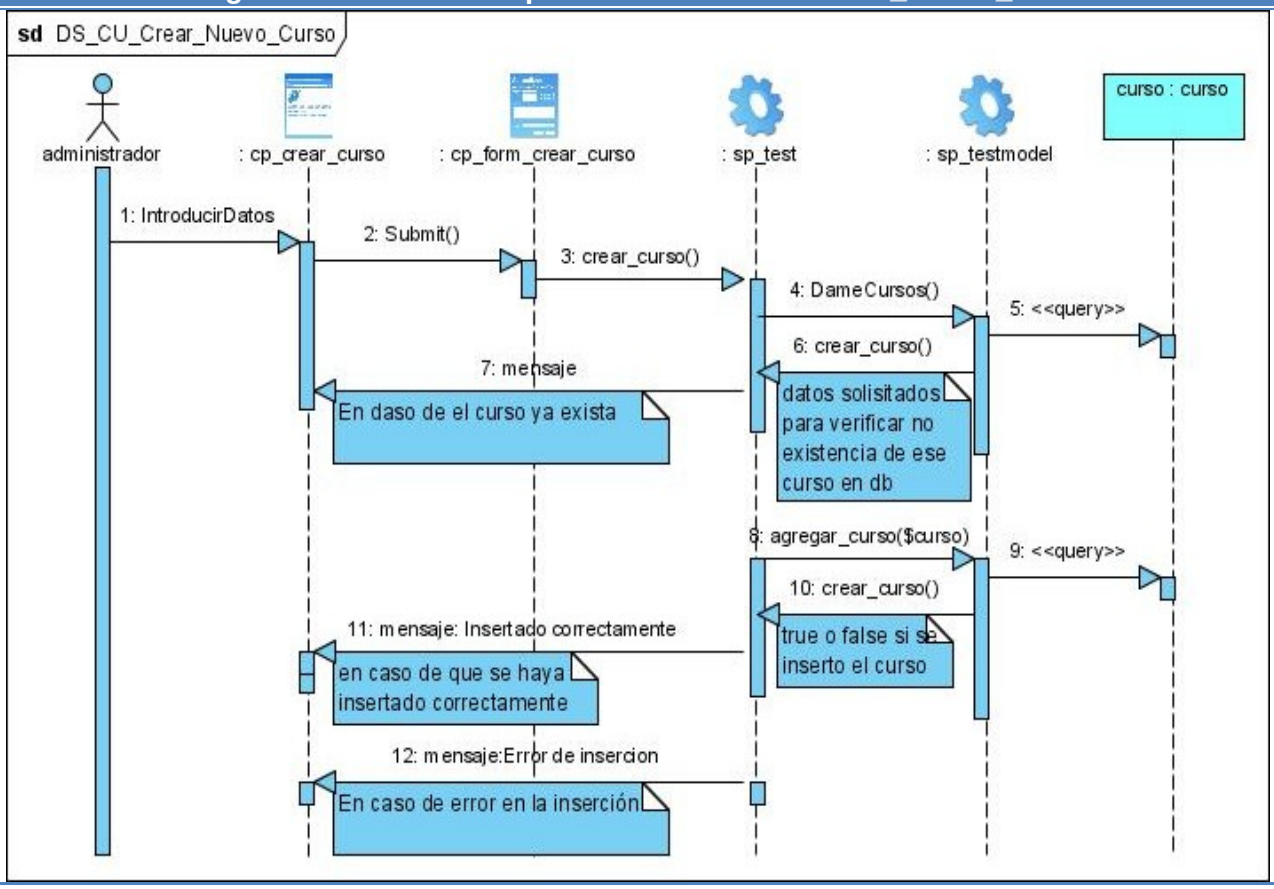

Figura 29 Diagrama de Secuencia del Caso de uso Crear_Nuevo_Curso.

Diagrama de Secuencia para el Caso de Uso Crear_Nuevo_Test(Alternativa 1)

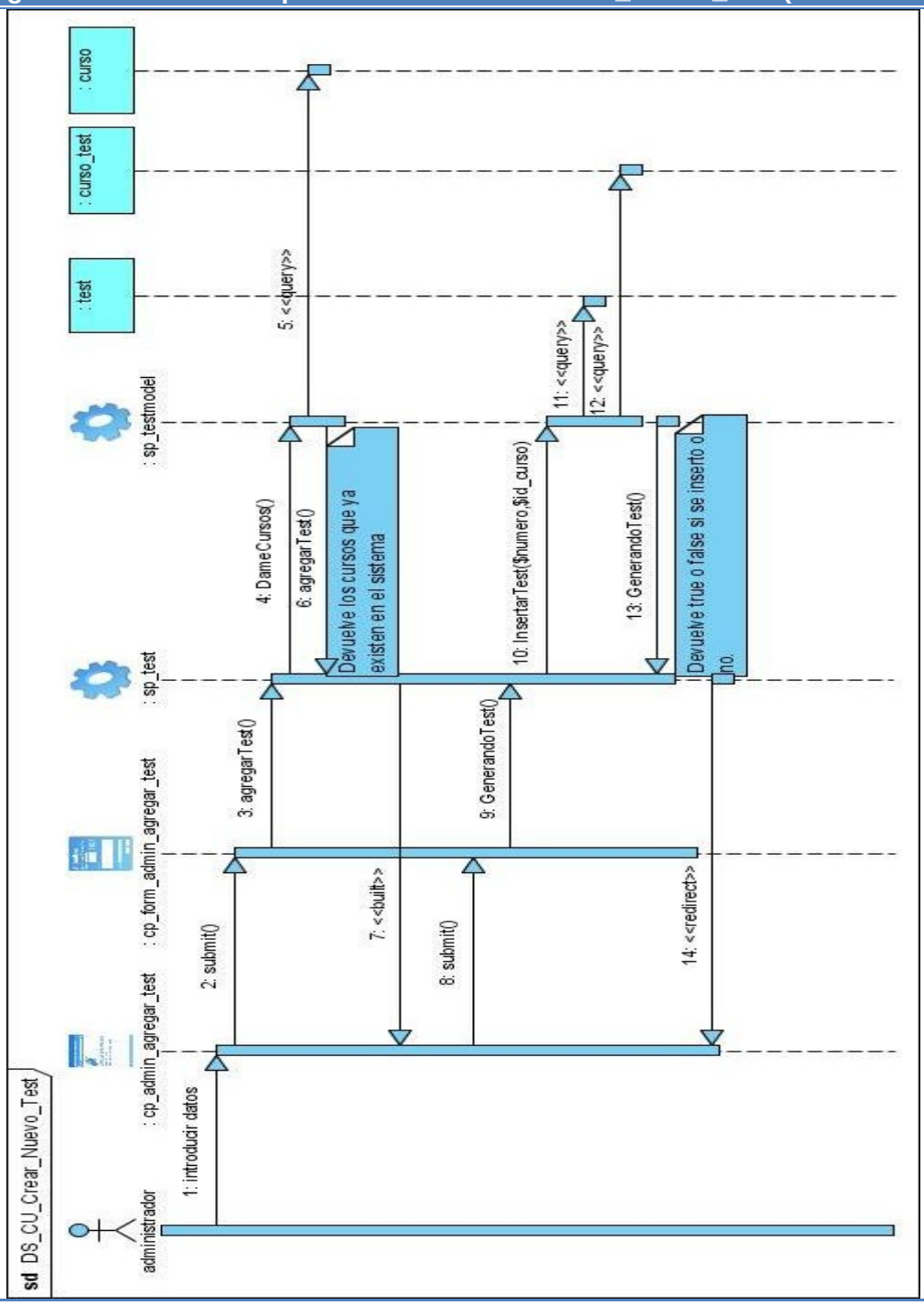

Figura 30 Diagrama de Secuencia del Caso de uso Crear_Nuevo_Test (Alternativa 1).

Diagrama de Secuencia para el Caso de Uso Crear_Nuevo_Test(Alternativa 2)

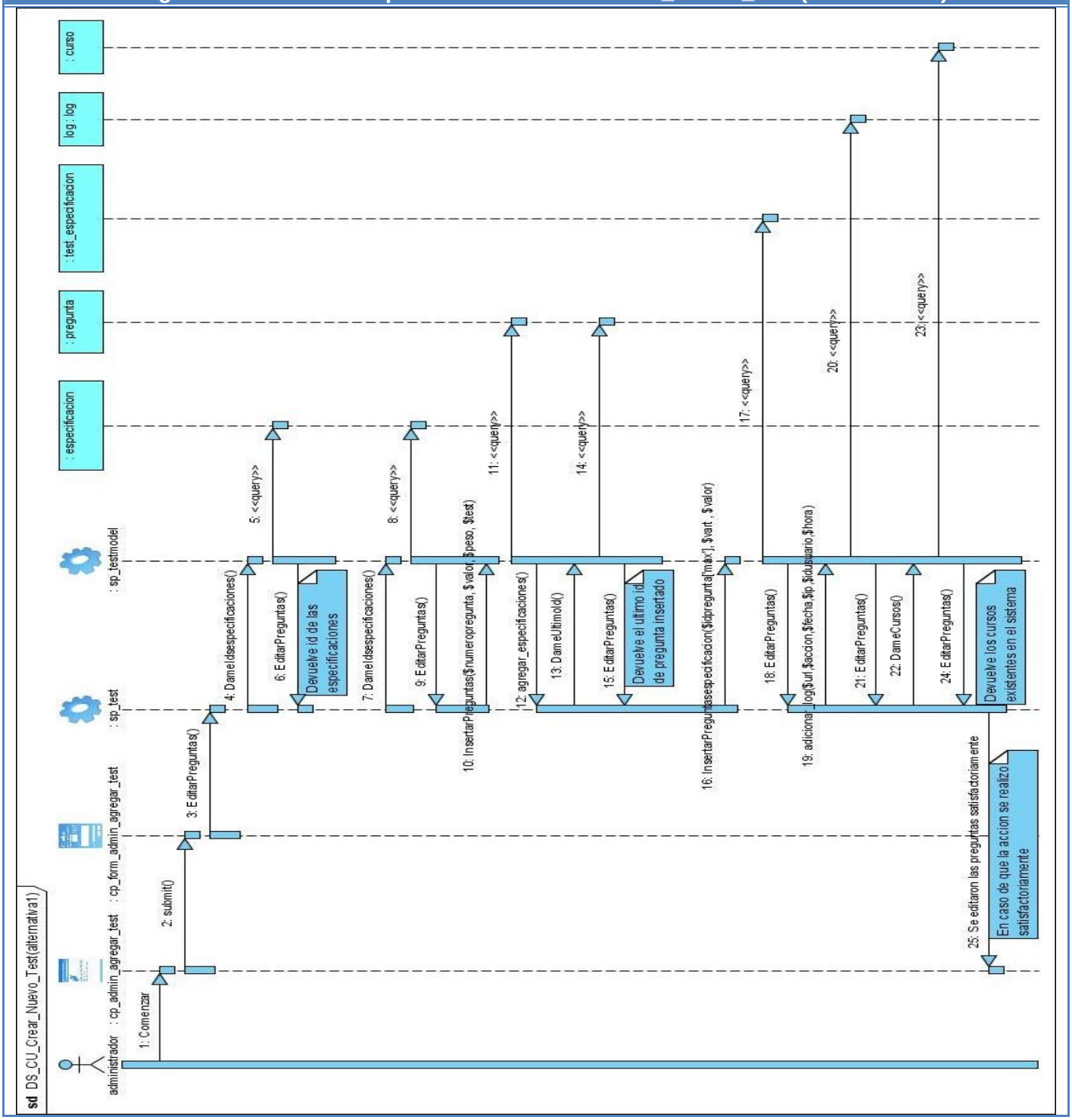

Figura 31 Diagrama de Secuencia del Caso de uso Crear_Nuevo_Test (Alternativa 2).

ANEXOS IV: DESCRIPCIÓN DE LAS CLASES DEL DISEÑO.

Controladoras

Nombre: estudiante	
Tipo de clase: controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	Estudiante()
Descripción:	Es función es el constructor de la clase.
Nombre:	index()
Descripción:	Esta función llama al menú principal de todas la interfaces.
Nombre:	generador_codigos ()
Descripción:	Esta función habilita en la vista las condicione para generar los códigos.
Nombre:	generando_codigo()
Descripción:	Esta función permite generar los códigos de anonimato.
Nombre:	Generado($avalor, $omitir)
Descripción:	Esta función permite generar los códigos de anonimato teniendo en cuenta las permutaciones de letras.
Nombre:	ReadFichero($nombrefic)
Descripción:	Esta función lee de un fichero para saber a partir de qué número debe empezar a generar los códigos.
Nombre:	WriteFichero($nombrefic, $nuevacantidad)
Descripción:	Escribe en el fichero el número en el que terminó la generación de códigos.
Nombre:	insertar()
Descripción:	Esta función insertar estudiante(nuevo código)
Nombre:	Generado()
Descripción:	Esta función en conjunto con la función generador códigos () genera los códigos de anonimato.
Nombre:	listar_codigo()
Descripción:	Esta función permite listar códigos por proceso
Nombre:	mostrar_codigos()
Descripción:	Esta función muestra los tipos de procesos para de ahí según el seleccionado se muestren los códigos

Nombre:	tipo_estudiante()
Descripción:	Esta función permite mostrar en el combo_box los tipos de estudiantes.

Tabla 16 Descripción de la Clase (controladora) de Diseño estudiante.

Nombre: reporte	
Tipo de clase: controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	Reporte()
Descripción:	Esta función es el constructor de la clase.
Nombre:	Crear_Reporte()
Descripción:	Esta función es para crear los reportes de notas y calificaciones
Nombre:	tipo_estudiantes()
Descripción:	Esta función es la que llama la vista de confeccionar reporte
Nombre:	buscar_codigo_sin_nota()
Descripción:	Esta función muestra un reporte de código sin nota.
Nombre:	log_actividades()
Descripción:	Esta función permite confeccionar los log.
Nombre:	log()
Descripción:	Esta función permite mostrar los usuarios y las fechas para de ahí saber los log de actividades.

Tabla 17 Descripción de la Clase (controladora) de Diseño reporte.

Nombre: test	
Tipo de clase: controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	Test()
Descripción:	Esta función es el constructor de la clase.
Nombre:	index()
Descripción:	Esta función llama al menú principal de todas la interfaces.
Nombre:	calificar()
Descripción:	Esta función califica la inserta las calificaciones de los estudiantes.

Nombre:	calificar_admin()
Descripción:	Esta función es para mostrarle los cursos y los números de los test al administrador y de ahí el seleccione curso-test que desee calificar.
Nombre:	modificar()
Descripción:	Esta función permite modificar la información que hay guardada sobre los test.
Nombre:	modificar_admin()
Descripción:	Esta función es para mostrarle los cursos y los números de los test al administrador y de ahí el seleccione curso-test que desee modificar.
Nombre:	agregarTest()
Descripción:	Esta función es para crear un nuevo test, con todas sus características bien definidas.
Nombre:	menu()
Descripción:	Esta función permite ver el menú con las distintas actividades q se pueden realizar.
Nombre:	Crear_Test()
Descripción:	Esta función completa la creación de un nuevo test, con todas sus características bien definidas.
Nombre:	GenerandoTest()
Descripción:	Esta función completa la creación de un nuevo test, con todas sus características bien definidas.
Nombre:	EditarPreguntas()
Descripción:	Esta función es para crear las preguntas de los test, con sus características.
Nombre:	listar_test()
Descripción:	Esta función es para mostrar los test que existen la base de datos del sistema.
Nombre:	crear_curso()
Descripción:	Esta función es para crear un nuevo curso.

Tabla 18 Descripción de la Clase (controladora) de Diseño test.

Nombre: usuarios	
Tipo de clase: controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	Usuarios()
Descripción:	Esta función es el constructor de la clase.
Nombre:	Login()
Descripción:	Esta función es la que permite la autenticación en el sistema.

Nombre:	index()
Descripción:	Esta función llama al menú principal de todas la interfaces.
Nombre:	agregar()
Descripción:	Esta función es para agregar un usuario
Nombre:	agregar_rol()
Descripción:	Esta función es para agregar un nuevo rol si fuera necesario.
Nombre:	InLdap()
Descripción:	Esta función es para comprobar los datos entrados con la información de Ldap
Nombre:	buscar_usuario()
Descripción:	Esta función es para organizar la búsqueda de los usuarios que están en al base de datos del sistema
Nombre:	buscar_usuarior_dado_tipo()
Descripción:	Esta función es para buscar los usuarios que pertenecen a un tipo.
Nombre:	buscar_tipo_dado_usuario()
Descripción:	Esta función es para buscar el tipo al que pertenece un usuario.
Nombre:	eliminar()
Descripción:	Esta función es para eliminar un usuario
Nombre:	actualizar
Descripción:	Esta función es para actualizar los cambios que se realicen referentes a los usuarios.
Nombre:	Salir()
Descripción:	Esta función es para salir de esta parte del sistema

Tabla 19 Descripción de la Clase (controladora) de Diseño usuarios.

Clases de acceso a datos

Nombre: estudiantemodel	
Tipo de clase: controladora (de acceso a datos)	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	EstudianteModel()
Descripción:	Esta función es el constructor de esta clase.
Nombre:	insertar_estudiante()
Descripción:	Esta función inserta un código en la Base de datos

Nombre:	buscar_codigo()
Descripción:	Esta función hace una búsqueda en la Base de datos
Nombre:	tipo_estudiante()
Descripción:	Esta función busca los tipos de estudiantes en la Base de datos.
Nombre:	eliminar_usuario()
Descripción:	Esta función elimina un usuario de la Base de datos.
Nombre:	tipo_usuario()
Descripción:	Esta función devuelve los tipos de usuarios existentes.
Nombre:	Insertar_codigos()
Descripción:	Esta función inserta un código en la Base de datos según el tipo de estudiante.
Nombre:	listar_codigos()
Descripción:	Esta función lista los códigos que exciten en la base de datos según el proceso al que pertenecen.
Nombre:	listar_cursos()
Descripción:	Esta función busaca el a base de datos la lista de los cursos que existen en el sistema

Tabla 20 Descripción de la Clase (acceso a datos) de Diseño estudiantemodel.

Nombre: reportemodel	
Tipo de clase: controladora (de acceso a datos)	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	ReporteModel()
Descripción:	Esta función es el constructor de esta clase.
Nombre:	Reporte($tipo, $id_curso_test)
Descripción:	Esta función hace una consulta ala Base de datos para buscar la información necesaria para realizar el reporte de notas.
Nombre:	Reporte1($tipo, $id_curso_test)
Descripción:	Esta función hace una consulta ala Base de datos para buscar la información necesaria para realizar el reporte de calificaciones.
Nombre:	adicionar_log($url, $accion, $fecha, $ip, $idusuario,$hora)
Descripción:	Se encarga de adicionar los log de actividades.
Nombre:	log_actividades($idusuario)

Descripción:	Se encarga de seleccionar los log de actividades dado un usuario determinado.
Nombre:	log_actividades_usuario_fecha($idusuario,$fecha)
Descripción:	Se encarga de seleccionar los log de actividades dado un usuario determinado y una fecha.
Nombre:	log_actividades_fecha($fecha)
Descripción:	Se encarga de seleccionar los log de actividades dado una fecha determinada.
Nombre:	usuario()
Descripción:	Esta función se encarga se seleccionar los usuarios existentes.
Nombre:	fecha()
Descripción:	Esta función se encarga se seleccionar las fechas existentes.
Nombre:	buscarcodigosinnota($tipo,$curso)
Descripción:	Se encarga de seleccionar los códigos sin notas.
Nombre:	tipo_estudiante()
Descripción:	Se encarga de seleccionar los tipos de estudiantes existentes.
Nombre:	listar_curso_test()
Descripción:	Se encarga de seleccionar los cursos y los números de los test existentes

Tabla 21 Descripción de la Clase (acceso a datos) de Diseño reportemodel.

Nombre: testmodel	
Tipo de clase: controladora (de acceso a datos)	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	TestModel()
Descripción:	Esta función es el constructor de esta clase.
Nombre:	get_list_preguntas()
Descripción:	Esta función hace una consulta a la Base de datos para saber una lista de preguntas pertenecientes de un test.
Nombre:	get_list_especificaciones()
Descripción:	Esta función hace una consulta a la Base de datos para saber una lista de preguntas especificaciones pertenecientes a un test.
Nombre:	get_id_pregunta()
Descripción:	Esta función hace una consulta a la Base de datos para obtener información de la pregunta.

Nombre:	numero_test()
Descripción:	Esta función inserta el número de test en la Base de datos.
Nombre:	buscar_idnota()
Descripción:	Esta función hace una consulta a la base de datos para saber el id de una nota
Nombre:	existe_notas()
Descripción:	Esta función verifica si para un estudiante determinado existen notas.
Nombre:	insertar_nota()
Descripción:	Esta función permite insertar notas en la base de datos.
Nombre:	actualizar_nota()
Descripción:	Esta función permite guardar en la base de datos los cambios que se realicen en las notas.
Nombre:	actualizar_notaesp()
Descripción:	Esta función permite guardar en la base de datos los cambios que se realicen en las notas de las especificaciones.
Nombre:	insertar_nota_especificacion()
Descripción:	Esta función permite insertar en la base de las notas de las especificaciones.
Nombre:	actualizar_nota_especificacion()
Descripción:	Esta función permite guardar en la base de datos los cambios que se realicen en las notas de las especificaciones.
Nombre:	delete_id_notas($arr_notas)
Descripción:	Esta función permite borrar una nota de la base de datos.
Nombre:	isset_codigo($codigo,$tipoes)
Descripción:	Esta función devuelve true si el código existe.
Nombre:	get_preguntas_notas($idtest, $idestudiante)
Descripción:	Esta función realiza una consulta a la base de datos para obtener las notas que pertenecen a un estudiante y un test especifico.
Nombre:	get_especificaciones_notas($idtest,$idestudiante)
Descripción:	Esta función realiza una consulta a la base de datos para obtener las especificaciones que pertenecen a un estudiante y un test específico.
Nombre:	get_insert_id()
Descripción:	Devuelve el último id insertado.
Nombre:	list_tipo_estudiante()

Descripción:	Selecciona los tipos de estudiantes existentes.
Nombre:	InsertarTest($numero,$id_curso)
Descripción:	Inserta el test.
Nombre:	InsertarPreguntas($numero, $valor, $peso, $idtest)
Descripción:	Inserta las preguntas.
Nombre:	DameUltimoId()
Descripción:	Esta función realiza una consulta a la base de datos para obtener el id de la última pregunta.
Nombre:	DameIdsespecificaciones()
Descripción:	Selecciona los id de las especificaciones.
Nombre:	InsertarPreguntasespecificacion($idpregunta, $idespecificacion, $cantidad)
Descripción:	Inserta las preguntas que tienen especificaciones.
Nombre:	Listar_Test()
Descripción:	Esta función hace una consulta a la base de datos para obtener los test.
Nombre:	buscar_numero_curso($idtest)
Descripción:	Esta función realiza una consulta a la base de datos para obtener el id de la última pregunta.
Nombre:	buscar_preguntas_dado_test_que_sean_normales($idtest)
Descripción:	Esta función realiza una consulta a la base de datos para obtener las preguntas normales de un test determinado.
Nombre:	buscar_preguntas_dado_test_que_sean_especificacion()
Descripción:	Esta función realiza una consulta a la base de datos para obtener las preguntas que tienen especificación de un test determinado.
Nombre:	listartiposestudiantes()
Descripción:	Esta función hace una consulta a la base de datos para obtener los tipos de estudiantes.
Nombre:	listartest()
Descripción:	Esta función selecciona los test.
Nombre:	DameCursos()
Descripción:	Esta función hace una consulta en la base de datos para obtener los cursos.
Nombre:	DameEspecificaciones()
Descripción:	Esta función hace una consulta en la base de datos para obtener las especificaciones.

Nombre:	agregar_esp()
Descripción:	Esta función permite insertar un nuevo rol en la base de datos.
Nombre:	agregar_curso()
Descripción:	Esta función permite insertar un nuevo curso en la base de datos.
Nombre:	listar_curso_test()
Descripción:	Esta función selecciona los cursos y los números de los test.

Tabla 22 Descripción de la Clase (acceso a datos) de Diseño testmodel.

Nombre: usuariosmodel	
Tipo de clase: controladora (de acceso a datos)	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	UsuariosModel()
Descripción:	Esta función es el constructor de esta clase.
Nombre:	Login()
Descripción:	Esta función permite verificar en la librería Ldap si el usuario y la contraseña que se está entrando al sistema existen y son correctas. Además comprueba que el usuario exista en la base de datos del sistema.
Nombre:	insertar_usuario($usuario, $idtipo_us,$id_cursotest)
Descripción:	Esta función inserta un usuario, y le asigna el tipo de usuario y el curso test que elige el administrador que esta haciendo la inserción.
Nombre:	curso_idcuresotest()
Descripción:	Esta función permite insertar un nuevo curso en la base de datos.
Nombre:	actualizar_usuario($idusuario, $idtipo_us)
Descripción:	Esta función actualiza en la base de datos los cambios que se realizan referentes a los usuarios.
Nombre:	tiene_acceso($login)
Descripción:	Esta función dado un usuario busca en la base de datos del sistema si este tiene acceso o no tiene.
Nombre:	listarU()
Descripción:	Esta función busca en la base de datos los usuarios que están en el sistema para mostrarlos.
Nombre:	eliminar_usuario()
Descripción:	Esta función es para eliminar usuarios de la base de datos del sistema.

Nombre:	tipo_usuario()
Descripción:	Esta función hace una consulta a la base de datos para obtener los tipos de usuarios.
Nombre:	agregar_rol_usuario($privilegio)
Descripción:	Esta función permite insertar un nuevo usuario, y definirle el rol que tendrá en el sistema.
Nombre:	tipo_de_usuario()
Descripción:	Esta función inserta en la base de datos el tipo de usuario.
Nombre:	buscar_usuarios()
Descripción:	Esta función permite que dado un tipo de usuario se busque en la base de datos todos los usuarios que pertenecen a este tipo.
Nombre:	buscar_tipo()
Descripción:	Esta función permite que dado un usuario busca su tipo en la base de datos.
Nombre:	buscar()
Descripción:	Esta función busca información en la base de datos. Según el usuario que se entra al sistemas
Nombre:	buscar_id()
Descripción:	Esta función busca en la base de datos el id del usuario que entra en el sistemas.

Tabla 23 Descripción de la Clase (acceso a datos) de Diseño usuariosmodel.

Nombre: asignar_rol	
Tipo de clase: interfaz	
Atributo	**Tipo**
privilegio	TextField
agregar	Button
Para cada responsabilidad:	
Nombre:	Funciones
Descripción:	Descripción de la función.

Tabla 24 Descripción de la Clase (interfaz) de Diseño asignar_rol.

Nombre: calificar	
Tipo de clase: interfaz	
Atributo	**Tipo**
proceso	List/Menu
codigo	TextField
pregunta	TextField
Para cada responsabilidad:	
Nombre:	Funciones
Descripción:	Descripción de la función.

Tabla 25 Descripción de la Clase (interfaz) de Diseño calificar.

Nombre: confeccionar_reporte	
Tipo de clase: interfaz	
Atributo	**Tipo**
criterio	Radio Button
tipo	List/Menu
curso	List/Menu
btnaceptar	Button
inicio	Button
Para cada responsabilidad:	
Nombre:	Funciones
Descripción:	Descripción de la función.

Tabla 26 Descripción de la Clase (interfaz) de Diseño confeccionar_reporte.

Nombre: generador_codigos	
Tipo de clase: interfaz	
Atributo	**Tipo**
idtipo	List/Menu
curso	List/Menu
cantidad	TextField
submit	Button
Para cada responsabilidad:	
Nombre:	Funciones
Descripción:	Descripción de la función.

Tabla 27 Descripción de la Clase (interfaz) de Diseño generador_codigos.

Nombre: insertar_estudiantes	
Tipo de clase: interfaz	
Atributo	**Tipo**
codigo	TextField
proceso	List/Menu
curso	List/Menu
btnagregar	Button
Para cada responsabilidad:	
Nombre:	Funciones
Descripción:	Descripción de la función.

Tabla 28 Descripción de la Clase (interfaz) de Diseño insertar_estudiantes.

Nombre: login	
Tipo de clase: interfaz	
Atributo	**Tipo**
login	TextField
password	TextField

submit	Button
Para cada responsabilidad:	
Nombre:	Funciones
Descripción:	Descripción de la función.

Tabla 29 Descripción de la Clase (interfaz) de Diseño login.

Nombre: modificar	
Tipo de clase: interfaz	
Atributo	**Tipo**
proceso	List/Menu
codigo	TextField
buscar	Button
pregunta	TextField
actualizar	Button
regresar	Button
Para cada responsabilidad:	
Nombre:	Funciones
Descripción:	Descripción de la función.

Tabla 30 Descripción de la Clase (interfaz) de Diseño modificar.

Nombre: reporte_calificaciones	
Tipo de clase: interfaz	
Atributo	**Tipo**
regresar	Button
Para cada responsabilidad:	
Nombre:	Funciones
Descripción:	Descripción de la función.

Tabla 31 Descripción de la Clase (interfaz) de Diseño reporte_calificaciones.

Nombre: reporte_notas	
Tipo de clase: interfaz	
Atributo	**Tipo**
regresar	Button
Para cada responsabilidad:	
Nombre:	Funciones
Descripción:	Descripción de la función.

Tabla 32 Descripción de la Clase (interfaz) de Diseño reporte_notas.

Nombre: admin_agregar_test	
Tipo de clase: interfaz	
Atributo	**Tipo**
numero	TextField
curso	List/Menu
preguntas_nor	TextField

preguntas_esp	TextFiel
agregar	Button
valor	TextField
peso	TextField
registrar	Button
Para cada responsabilidad:	
Nombre:	Funciones
Descripción:	Descripción de la función.

Tabla 33 Descripción de la Clase (interfaz) de Diseño admin_agregar_test.

Nombre: escoger_modificar_admin	
Tipo de clase: interfaz	
Atributo	**Tipo**
testingcurso	List/Menu
Submit	Button
Para cada responsabilidad:	
Nombre:	Funciones
Descripción:	Descripción de la función.

Tabla 34 Descripción de la Clase (interfaz) de Diseño escoger_modificar_admin.

Nombre: escoger_calificar_admin	
Tipo de clase: interfaz	
Atributo	**Tipo**
testingcurso	List/Menu
Submit	Button
Para cada responsabilidad:	
Nombre:	Funciones
Descripción:	Descripción de la función.

Tabla 35 Descripción de la Clase (interfaz) de Diseño escoger_calificar_admin.

Nombre: crear_curso	
Tipo de clase: interfaz	
Atributo	**Tipo**
curso	TextField
agregar	Button
Para cada responsabilidad:	
Nombre:	Funciones
Descripción:	Descripción de la función.

Tabla 36 Descripción de la Clase (interfaz) de Diseño crear_curso.

ANEXOS V: DESCRIPCIÓN DE LAS ENTIDADES DE LA BASE DE DATOS.

A continuación se realiza una.

Nombre: curso		
Descripción: Registro del nombre de los cursos.		
Atributo	**Tipo**	**Descripción**
id_curso	integer	
curso	varchar(50)	

Tabla 37 Descripción de la tabla curso.

Nombre: curso_test		
Descripción: Esta tabla surge producto de la relación de muchos a muchos entre la tabla curso y la tabla test, y registra la fecha en que se aplica un test.		
Atributo	**Tipo**	**Descripción**
id_cursotest	integer	
idtest	integer	
Id_curso	integer	

Tabla 38 Descripción de la tabla curso_test.

Nombre: test		
Descripción: Registro de los test de ingreso que se le aplican a los aspirantes a entrar a la UCI		
Atributo	**Tipo**	**Descripción**
idtest	integer	
numero	integer	
curso	varchar(15)	

Tabla 39 Descripción de la tabla test.

Nombre: estudiante		
Descripción: Registra los códigos que pertenecen a los estudiantes y el identificador del curso-test que este realizó.		
Atributo	**Tipo**	**Descripción**
idestudiante	integer	
codigo	char(3)	
idtipo_estudiante	integer	
idtest	integer	

Tabla 40 Descripción de la tabla estudiante.

Nombre: tipo_es		
Descripción: Guarda los tipos de estudiantes que existen.		
Atributo	**Tipo**	**Descripción**
idtipo_es	integer	
tipo	varchar(30)	

Tabla 41 Descripción de la tabla tipo_es.

Nombre: especificacion		
Descripción: Guarda el nombre de las especificaciones que forman la pregunta de especificaciones.		
Atributo	**Tipo**	**Descripción**
idespecificaciones	integer	
nespecificacion	varchar(20)	
calculable	boolean	

Tabla 42 Descripción de la tabla especificacion.

Nombre: test_especificacion		
Descripción: Surge de la relación de muchos a muchos entre pregunta y especificación y tiene un atributo cantidad.		
Atributo	**Tipo**	**Descripción**
Idtest_especificacion	integer	
idpregunta	integer	
idespecificacion	integer	
cantidad	integer	

Tabla 43 Descripción de la tabla test_especificacion.

Nombre: pregunta		
Descripción: Registra las preguntas que tienen los teste de ingreso.		
Atributo	**Tipo**	**Descripción**
idpregunta	integer	
idtest	integer	
valor	Double precision	
numero	integer	
peso	integer	

Tabla 44 Descripción de la tabla pregunta.

Nombre: nota		
Descripción: Registra las notas que obtienen los estudiantes.		
Atributo	**Tipo**	**Descripción**
idnota	integer	
valor	double precision	
idpregunta	integer	
usuario	varchar(254)	
Ip	varchar(254)	
idestudiante	integer	
fecha	date	
pc	varchar(254)	
hora	time(0)	

Tabla 45 Descripción de la tabla nota.

Nombre: nota_especificacion		
Descripción: Registra las notas de las preguntas de especificaciones de los estudiantes.		
Atributo	**Tipo**	**Descripción**

Idnota_especificaciones	integer	
idnota	integer	
idtest_especificacion	integer	
valor	integer	

Tabla 46 Descripción de la tabla nota_especificacion.

Nombre: usuario		
Descripción: Registra todos los usuarios que tienen acceso al sistema.		
Atributo	**Tipo**	**Descripción**
idusuario	integer	
usuario	varchar(254)	
Idtipo_us	integer	

Tabla 47 Descripción de la tabla usuario.

Nombre: tipo_us		
Descripción: Registra los distintos tipos de roles que puede ser un usuario en el sistema.		
Atributo	**Tipo**	**Descripción**
Idtipo_us	integer	
tipo	Varchar(30)	

Tabla 48 Descripción de la tabla tipo_us.

Nombre: log		
Descripción: Registra todas las acciones de los usuarios en el sistema.		
Atributo	**Tipo**	**Descripción**
idlog	integer	
url	varchar(254)	
accion	varchar(50)	
fecha	date	
ip	varchar(255)	
idusuario	integer	

Tabla 49 Descripción de la tabla log.

GLOSARIO DE TÉRMINOS.

1. APACHE: servidor HTTP de código abierto para plataformas Unix (BSD, GNU/Linux, etcétera), Windows y otras, que implementa el protocolo HTTP/1.1.
2. Arquitectura Cliente/Servidor: es un modelo para el desarrollo de sistemas de información, en el que las transacciones se dividen en elementos independientes que cooperan entre sí para intercambiar información, servicios o recursos.
3. CASE: (Computer Aided Software Engineering). Serie de herramientas, lenguajes y técnicas de programación que permiten la generación de aplicaciones de manera semiautomática.
4. Código abierto: Es un software de libre distribución que publica su código fuente, lo que permite que cualquiera pueda modificarlo y colaborar así a su desarrollo.
5. CPT: Curso para Trabajadores.
6. Framework: plataforma para el fácil desarrollo de proyectos, que cuenta con funcionalidades definidas con de forma genérica de manera tal que los proyectos se implementen sobre el mismo de una manera rápida fácil y segura.
7. Herramientas CASE: Liberan al programador de parte de su trabajo y aumentan la calidad del programa a la vez que disminuyen sus posibles errores.
8. HTML: (HyperText Markup Language). Lenguaje de marcado de Hipertexto. Es el lenguaje estándar para describir el contenido y la apariencia de las páginas en el WWW.
9. IPI: Instituto Politécnico de Informática.
10. http: (Hiper Text Transfer Protocol). Protocolo de transferencia de HiperTexto. Es el protocolo de Internet que permite que los exploradores del WWW recuperen información de los servidores.
11. Licencia GPL: Permite utilizar un programa GPL en un ordenador sin limitación. CGI: (Common Gatewav Interface). Interface de intercambio de datos estándar en WWW a través del cual se organiza el envío de recepción de datos entre visualizadores y programas residentes en servidores WWW.
12. Open source: Es el software que, una vez obtenido, puede ser usado, copiado, estudiado, modificado y redistribuido libremente.
13. Perl: Lenguaje de programación muy utilizado para la elaboración de aplicaciones CGI. Es multiplataforma y funciona bajo UNIX
14. PostgreSQL: Sistema de Gestión de Bases de Datos Objeto-Relacionales.

15. SOA: Arquitectura Orientada a Servicios (en inglés Service-Oriented Architecture).
16. SQL: (Structured Query Language). Es un estándar en el lenguaje de acceso a bases de datos. En la actualidad está adoptado por ISO.