Universidad de las Ciencias Informáticas

SISTEMA DE CONTROL DE INVERSIONES

“ConInver”.

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

AUTORES

Alejandro Manuel Carreras Leyva

Yaniris Flores Rodríguez

TUTORES

Ing. Irene Jiménez Alcázar

Msc. Rafael Luis Torralbas Ezpeleta

Ciudad de La Habana

Mayo 2009

"Un inversor necesita hacer muy pocas cosas bien si evita grandes errores. No es necesario hacer cosas extraordinarias para conseguir resultados extraordinarios"

Warren Buffett.

DECLARACIÓN DE AUTORÍA

Declaramos ser los únicos autores de este trabajo y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales del mismo, con carácter exclusivo.

Para que así conste firmo la presente a los____días del mes de_____del año_____.

_____

Firma del Autor

Alejandro Manuel Carreras Leyva

_____

Firma del Autor

Yaniris Flores Rodríguez

_____

Firma del Tutor

Ing. Irene Jiménez Alcázar

_____

Firma del Tutor

Msc. Rafael Luis Torralbas Ezpeleta

OPINIÓN DEL USUARIO

El Trabajo de Diploma, titulado **Sistema de Control de Inversiones "ConInver"**, fue realizado en la **Unidad Presupuestada Inversionista (UPI) "Proyecto Futuro"**. Esta entidad considera que, en correspondencia con los objetivos trazados, el trabajo realizado le satisface

□ Totalmente

□ Parcialmente en un ____ %

Los resultados de este Trabajo de Diploma le reportan a esta entidad los beneficios siguientes (cuantificar):

Como resultado de la implantación de este trabajo se reportará un efecto económico que asciende a <**valor en MN o USD del efecto económico**>

Y para que así conste, se firma la presente a los ____ días del mes de _____ del año _____.

_____	_____
Representante de la entidad	Cargo
_____	_____
Firma	Cuño

OPINIÓN DEL TUTOR

Título: **Sistema de Control de Inversiones.**

Autores: Alejandro Manuel Carreras Leyva.

Yaniris Flores Rodríguez

Ing. Irene Jiménez Alcázar

_____ _____

Firma del Tutor Fecha

Msc. Rafael Luis Torralbas Ezpeleta

_____ _____

Firma del Tutor Fecha

DATOS DE CONTACTO

Ing. Irene Jiménez Alcázar

Msc. Rafael Luis Torralbas Ezpeleta

Graduado de Ingeniería en Automática en la Universidad de Oriente, Santiago de Cuba, en Julio de 1996. Entre 1996 y el 2001 ha trabajado en Copextel, S.A. División Holguín como: Especialista de Servicios Técnicos Informáticos, Especialista Comercial de Equipamiento Informático, Gerente Grupo Informática y Redes de Datos de la División. Gerente Comercial de la División. Luego en el Grupo Inversiones de la Batalla de Ideas como: Inversionista Principal Escuela Formadora de Trabajadores Sociales de Holguín. (2001), Inversionista Principal de la Universidad de las Ciencias Informáticas ejecutando 4 Etapas de esta inversión (2002-2006). En la Universidad de las Ciencias Informáticas se desempeñó como Vicerrector de Inversiones Mantenimiento y Servicios Generales. (2006-2007). Continúo atendiendo la inversión de la UCI y además las Direcciones de Mantenimiento y Servicios Generales. Actualmente se desempeña en la Unidad Presupuestada Inversionista “Proyecto Futuro” como Director (2007-2009). Entidad surgida para la gestión del proceso inversionista de la UCI y sus Facultades Regionales. Impartió las Conferencias sobre las Regulaciones del Proceso Inversionista en Cuba según la Resolución 91 del Ministerio de Economía y Planificación. Graduado de Máster en Dirección, en el ISPJAE, Ciudad de La Habana, en julio de 2007. Pertenece a la Unión Nacional de Arquitectos e Ingenieros de Cuba.

AGRADECIMIENTOS

A mi mamá y mi abuela, por dar todo de sí y sacrificarse incondicionalmente en hacer de mí el hombre que hoy soy.

A mi novia Elizabeth por apoyarme, comprenderme y darme fuerzas para llegar.

A mis tutores, Rafael Luis Torralbas e Irene Jiménez por su valiosa experiencia y desempeño profesional, por esforzarse cada día en convertirme en un mejor profesional.

A mi primos Yisell y Otto, por guiarme y asesorarme en los momentos más difíciles de mi carrera.

A todos mis profesores, en especial a: Héctor Rodríguez, María Cristina, Hilda Esther, Yenisleydi Cariaga, (Pernia) x2, Annia Surós, Kelvys Gálvez, Andro Gilberto, Susel Vázquez, Maidely Calderón y Cesar.

A mis amigos que sin ellos no hubiese llegado hasta aquí: Sucel, Walfrido y Yudmila, (Carlitos) x2, Emilio Lago, Aluisco, Reinier, Rafa, Luis y Yanedy.

A mi grupo por soportarme estos 5 años y compartir conmigo experiencias inolvidables.

Especialmente a todas estas personas que nos brindaron su apoyo incondicional: Iosev Perez, Michel David, Reinier Elejalde, Carlos M. Hernandez, Dayana Daniel, Rolando Bonal.

A la revolución cubana y su máximo líder nuestro Comandante Invicto Fidel Castro Ruz.

Alejandro Manuel.

A mi mami, por siempre tener un consejo y apoyarme en los momentos más difíciles…

A mi papi por educarme, enseñarme todo lo que se y regañarme cuando era necesario…

A mima, por darme ese papi tan lindo y quererme tanto…

A una personita especial en vida, Betty, por ayudarme, aconsejarme y cuidar tanto esa otra parte de mi, Lore…

A mis tres hermanitos, José Carlos, Yani y Lorencito por ser tan bellos y por brindarme toda su alegría…

A mi novio Alci, por tener tanto amor para mí, por siempre estar para mí cuando lo necesitaba, por escucharme, aconsejarme y soportarme durante todo este tiempo…

A mis tutores Irene, por estar siempre a nuestro lado, en los días buenos y en los malos, por tantos consejos y Torralbas, por su experiencia…

A mi compañero de tesis, Ale, por compartir este tiempo de trabajo intenso, por siempre llegar a un acuerdo, aunque en ocasiones fuera un poco difícil…

A mis profes por todo lo que me enseñaron a lo largo de mi carrera, en especial a Héctor por siempre estar cuando lo necesitaba…

A mis compañeros de grupo, los que no están también, por compartir un poquito de su tiempo. A todos mis amigos, los que están aquí hoy y a los que se alegran en la distancia que este día tan esperado llegue…

En especial gracias a la Revolución, a Fidel y a Raúl por haberme dado la oportunidad de estudiar en este centro universitario...

A todos, mi eterno agradecimiento con la seguridad de que siempre estarán en mi corazón…

Yaniris.

DEDICATORIA

A las personas más importantes en mi vida, mi abuela del alma María Esther y mi mamita Marisol.

A mis tías por todo su amor y apoyo incondicional, Miriam, Marlene y Margarita.

A mi tío Roberto Cordovés, por estar siempre y demostrarme que siempre puedo dar más.

Alejandro Manuel.

A mami y mima por confiar en mí y brindarme todo su amor.

A mis tres hombres, al más pequeño, Lore, por ser para él el ejemplo a seguir, al del medio, Alci, por estar todos los días con ese amor para mí y al más grande, mi papi, por educarme y quererme tanto.

Yaniris.

RESUMEN

El Departamento (Dpto.) de Control de la Unidad Presupuestada Inversionista (UPI)"Proyecto Futuro" y sus Direcciones Integradas de Proyecto (DIP) no cuentan con un sistema automatizado que les permita llevar el control del proceso inversionista, por lo que, requieren contar con un software, que le brinde las prestaciones necesarias, para obtener los datos más importantes, resumirlos y presentarlos de la forma más comprensible posible.

Para el desarrollo de la aplicación se utilizó como lenguaje de programación Java, con el entorno de desarrollo integrado (IDE) NetBeans en su versión 6.5, como Framework Hibernate, como sistema gestor de bases de datos (SGBD) PostgreSQL versión 8.3, como metodología de desarrollo el Proceso Unificado de Desarrollo (RUP), para especificar, construir y documentar el sistema se hace uso del Lenguaje Unificado de Modelado (UML) y Visual Paradigm para UML 2.0 para la creación de los artefactos que se generan durante el ciclo de vida del software.

La creación del sistema de control de inversiones "ConInver" le permitirá al Dpto. de Control de la UPI conocer en todo momento el estado de las inversiones en cuanto a producción, contratación, estado de la facturación, avance de las obras y estado económico, con vista a establecer una comparación entre el estado planificado y el real alcanzado.

Palabras Claves: ConInver, Control, Inversiones, UPI.

ABSTRACT

The Control Department of the Investment Budgeted Unit Proyecto Futuro (Project Future) and its Project Integrated Departments do not have an automated system which allows them to control the investment process; that is why they need software that offers the required features, in order to obtain the most relevant data, summarize them and present them in a user-friendly manner.

Java was the programming language used for the application, using NetBeans as IDE, in its version 6.5. The framework Hibernate was used, as well as PostgreSQL, version 8.3, as the database management system. The Rational Unified Process (RUP) was the development methodology used and the Unified Modeling Language (UML) was used to document, specify and model the whole system. Visual Paradigm for UML 2.0 was used for the creation of the artifacts generated during the project lifecycle.

The creation of *ConInver* Investment Control System will allow the Control Department of UPI, to know the state of investments at every moment, in regards to production, contracting, invoice status, work progress and economic data, in order to establish a comparison between the planned state and the actual achieved state.

Keywords: ConInver, Control, Investments, UPI

ÍNDICE DE CONTENIDOS

INTRODUCCIÓN

El alto grado de desarrollo alcanzado en la rama de la computación, ha motivado que la gran mayoría de las empresas se encuentren en algún grado informatizadas, convirtiéndose los sistemas computacionales en un recurso imprescindible para la gestión de la sociedad moderna. Estos sistemas proporcionan la infraestructura suficiente y necesaria para un eficiente manejo de la información, elimina el trabajo engorroso de realizarlo manualmente, evita que se cometan errores por el agotamiento del cerebro humano y permite obtener informes actualizados de todos los procesos que se encuentren digitalizados, facilitando el control y seguimiento de los mismos.

El control de las inversiones es una tarea vital para toda entidad inversionista. Conocer el estado de sus obras, de las contrataciones, del presupuesto y de los planes anuales, garantiza el buen funcionamiento de la misma. En la actualidad existen varios tipos de sistemas de gestión económica y financiera[1] que le permiten a las empresas llevar controles de los procesos inversionistas vistos desde una óptica contable, destacando que la mayoría de estos pertenecen a economías capitalistas, elemento que entra en contradicción con el modelo económico de Cuba. En nuestro caso específico la entidad beneficiada es una Unidad Presupuestada, cuya gestión económica gira en torno al presupuesto y a los planes anuales aprobados, los cuales no pueden ser sobre ejecutados.

Los sistemas existentes, están diseñados, como se mencionaba anteriormente, para modelos distintos al cubano y para el control económico de las empresas, por lo que sus funcionalidades no responden a todas las necesidades de las entidades inversionistas del país. En casi todos los casos, incluyen módulos que no son empleados, por tanto, se está pagando la licencia de estos productos sin obtener las funcionalidades requeridas, siendo los mismos subexplotados. La Unidad Presupuestada Inversionista (UPI) “Proyecto Futuro”, no está ajena a esto, ya que usa el Assets[2] como sistema contable. La información que se obtiene a partir de este sistema está basada en los procesos contables y el control de

[1] Ejemplo de sistemas contables: Assets, Envestnet Asset Management, Fairbairn, Man Investments, Hencorp, GimParagon, SgestMan.

[2] Assets: Sistema de Gestión Integral estándar y parametrizado que permite el control de los procesos de Compras, Ventas, Producción, Taller, Inventario, Finanzass, Contabilidad, Presupuesto, Activos Fijos, Útiles y Herramientas y Recursos Humanos desde una perspectiva económica.

una inversión no es solamente la contabilidad de la misma. Para su utilización por parte de los inversionistas, el Departamento (Dpto) de Control de la UPI, requiere contar con un software que le brinde las prestaciones necesarias para obtener los datos más importantes para controlar el proceso inversionista, resumirlos y presentarlos de la forma más comprensible posible, que sea modificable y que además posibilite futuras actualizaciones con nuevas prestaciones, que se le pueda dar soporte y que se relacione con las informaciones que se registran en el sistema contable de la Universidad de las Ciencias Informáticas (UCI).

En la entidad solo se tienen documentos Excel[3] (tablas dinámicas) donde se recoge toda la información generada durante el proceso inversionista, siendo esta insuficiente para todos los controles que se requieren, además, existe demora al dar respuesta a las solicitudes de información debido al tiempo que toma llegar a la misma producto del gran cúmulo de datos a procesar. Estos datos son propensos a errores por ser procesados a mano, además de tener que consultar un gran número de estos documentos para llegar a una respuesta final.

Por estas razones el **problema científico** del presente trabajo se plantea de la siguiente manera ¿Cómo resolver de forma automatizada el control de los procesos inversionistas en la Unidad Presupuestada Inversionista “Proyecto Futuro”?

De lo anteriormente expresado se deriva como **objeto de estudio** de la presente investigación los procesos de control de inversiones en entidades presupuestadas. De ello se deriva que el **campo de acción** que comprende este trabajo, son los procesos de control de inversiones en la UPI “Proyecto Futuro”.

El objetivo **general** trazado, es desarrollar una aplicación de escritorio que le permita al Dpto. de Control de la UPI “Proyecto Futuro”, así como a todas las Direcciones Integradas de Proyectos (DIP), la obtención de información actualizada e inmediata a través de reportes que le brinden, en tiempo real, información de las ejecuciones reales de las constructoras y de sus obras.

[3] Excel: Programa de Microsoft, el cual consiste en una hoja de cálculo, utilizada para realizar fórmulas matemáticas y cálculos aritméticos exhaustivos.

Derivados del mismo, se presentan además, una serie de **objetivos específicos** para el correcto desarrollo de la aplicación:

- Realizar un estudio sobre formas y mecanismos de los sistemas contables existentes en el ámbito nacional e internacional.
- Efectuar un levantamiento de información acerca de los procesos contables en la UPI para informatizarlos y lograr su optimización.
- Conocer los tipos de controles que requiere la entidad.
- Automatizar la dinámica del proceso inversionista.
- Certificar la integridad de la información contenida en el sistema a través de los principios básicos de seguridad informática.

Para darle cumplimiento a los objetivos específicos antes expuestos, se plantean las siguientes **tareas de investigación:**

- Estudio de los procesos contables y las resoluciones ministeriales vigentes.
- Entrevistas a los especialistas para conocer la documentación generada durante el proceso.
- Análisis de la situación tecnológica en el área internacional.
- Selección de la metodología de Análisis y Diseño de sistemas informáticos, que facilite la creación y garantice la calidad del sistema.
- Elección de las herramientas para llevar a cabo el desarrollo de la aplicación, fundamentando las elecciones.
- Elaboración del modelo de negocios y el levantamiento de requisitos.
- Diseño de una base de datos que soporte las funcionalidades del sistema.
- Formulación de un proceso que garantice los principios básicos de la seguridad de la información.
- Realización del análisis, diseño e implementación de las funcionalidades.
- Realización de las pruebas al sistema.

Se pretende que el sistema y todos los componentes que garantizarán su funcionamiento sean extensibles o exportables a otras instituciones, que sea multiplataforma, con un alto rendimiento y un elevado nivel de seguridad. Se requiere que las herramientas a utilizar sean todas libres, punto este que permitirá su redistribución, modificación y la propia utilización sin conflicto legal.

Para realizar la investigación se usaron métodos **teóricos** como el **analítico-sintético** que sirven de marco para un profundo análisis de los temas que motivaron el trabajo. Para ello se realizaron búsquedas minuciosas sobre los sistemas de control, haciendo énfasis en los procesos inversionistas, extrayendo los aspectos más importantes que contribuyeron al desarrollo de nuestra investigación. Para la implementación del sistema se realizó un estudio previo del proceso de control de inversiones, y los principales elementos que integran los módulos de la entidad. Los resultados alcanzados, junto con el conocimiento empírico, permitieron conseguir un dominio abarcador del tema al que proponemos solución.

Fue necesario el uso del método **inductivo-deductivo** porque son analizados los temas de investigación de forma general para arribar a puntos específicos dentro de la investigación.

En cuanto a los métodos **empíricos**, se aplicaron las **entrevistas**, que fueron útiles por el enriquecimiento del conocimiento adquirido con personas experimentadas en el ámbito de las inversiones y los sistemas contables. Se utilizó como método auxiliar la **observación** de los procesos que se desarrollan en las DIP.

El presente documento se estructura en cuatro capítulos, además de las secciones de Conclusiones, Recomendaciones, Bibliografía, Referencias Bibliográficas, Glosario de Términos y Anexos.

Capítulo 1 - Fundamentación teórica: Se analizan las tendencias actuales, se describen algunos conceptos y características de los sistemas de control que se deben tener en cuenta para comprender los objetivos del trabajo. Además se realiza un análisis de las tecnologías de software empleadas para desarrollar el sistema.

Capítulo 2 - Características del sistema: Se abordan las descripciones iniciales del sistema, se realiza un estudio de la UPI, así como un análisis crítico de los procesos involucrados, se modela el negocio, se presenta la propuesta del sistema, se definen los requerimientos funcionales y no funcionales y los casos de uso del sistema.

Capítulo 3 - Análisis y diseño del sistema: Se determinan las clases que se utilizarán en la implementación del sistema y las relaciones entre ellas. Se describe el diseño del sistema que se propone. Se representan los diagramas de interacción, los diagramas de clases y el diseño de la base de datos.

Capítulo 4 - Implementación y prueba del sistema: Se describe el sistema que se propone a través de una óptica de programación, basada en diagramas de componentes, diagrama de despliegue y una descripción de los casos de prueba.

CAPÍTULO 1
FUNDAMENTACIÓN TEÓRICA

1.1 Introducción.

Con el presente capítulo se pretende lograr un basamento teórico sólido, acerca de la situación actual que genera la necesidad de realizar este trabajo, del marco en que se desenvuelve, las tendencias actuales en el mundo para resolver problemas como éste y la definición de cómo llegar a su resolución. Se hace un estudio minucioso de las tecnologías, métodos y herramientas que serán utilizadas en el desarrollo de este proyecto. Para desarrollar la parte práctica con la calidad adecuada y tener una gama de conocimientos que permita tomar las mejores decisiones en el transcurso de la producción, es preciso dedicar un espacio a un estudio, como el que a continuación se refleja.

1.2 ¿Qué es un sistema de control?

Un sistema de control se puede definir como la integración de un conjunto de componentes que pueden regular su propia conducta o la de otro sistema, con el fin de lograr un funcionamiento predeterminado, de modo que se reduzcan las probabilidades de fallos y se obtengan los resultados buscados. Es un proceso mediante el cual se asegura que las actividades reales se ajusten a las actividades planificadas. Permite mantener una organización o sistema en buen camino, estableciendo medidas para corregir las actividades, de tal forma que se alcancen los planes exitosamente. El control actúa en todas las áreas y en todos los niveles de una empresa, prácticamente todas las actividades están bajo alguna forma de control o monitoreo. *(Iglesias Morell, 2008)*

1.2.1 ¿Qué es un sistema de control de inversiones?

Entiéndase como un sistema de control de inversiones todo aquel que busque como objetivo *(Iglesias Morell, 2008)*:

- Conocer el estado de las inversiones en cuanto a producción, contratación, estado de la facturación, plan anual, avance de las obras, estado económico con vistas a establecer una comparación entre lo planificado y lo real alcanzado, analizar las causas de la desviación, trazar las estrategias y tomar las medidas necesarias para corregir dicha desviación.

1.3 Sistemas automatizados existentes vinculados al campo de acción.

Existen varios sistemas contables que brindan datos que pueden ser usados para el control de las inversiones pero en principio no son diseñados para gerenciar esta actividad, por lo que no le brindan a un especialista en inversiones todas las funcionalidades requeridas por este.

1.3.1 SGestMan.

Tecnología integral de gestión de mantenimiento, acompañada de un sistema informático para la organización y control de la actividad de mantenimiento. Sus principios generales son *(Inversiones GAMMA S.A.)*:

- Organización de la información del patrimonio y los recursos humanos (RRHH), proyección, programación y planificación de las actividades preventivas de mantenimiento.
- Ofrecer al operario, técnico y directivo de mantenimiento toda la información para que pueda actuar con eficacia en el aumento de la disponibilidad de los equipos y disminución de los costos.
- Integración con los sistemas informáticos en explotación.

El sistema cuenta con los siguientes módulos de trabajo, que recogen y procesan la información necesaria para el mantenimiento en la empresa: Patrimonio, RRHH, Mantenimiento Preventivo, Orden de Servicio, Economía, Logística, Producción, Informes y Administración y Seguridad. *(Inversiones GAMMA S.A.)*

1.3.2 Assets NS.

Este es un sistema comercializado por la firma panameña D´MARCO S.A. y distribuido en Cuba por INFOMASTER, entidad informática perteneciente a la Empresa Nacional de Producción y Servicios a la Educación Superior del MES. *(Assets)*

ASSETS NS es un Sistema de Gestión Integral estándar y parametrizado que permite el control de los procesos de Compras, Ventas, Producción, Taller, Inventario, Finanzas, Contabilidad, Presupuesto, Activos Fijos, Útiles y Herramientas y RRHH. Como Sistema Integral todos sus módulos trabajan en estrecha relación, generando, automáticamente, al Módulo de Contabilidad los Comprobantes de Operaciones por cada una de las transacciones efectuadas, esto permite que se pueda trabajar bajo el principio de Contabilidad al Día. Dispone, además, de métodos novedosos para administración y planificación de inventarios, así como una amplia gama de Análisis y Consultas que le permitirán no sólo conocer exactamente la situación actual, sino proyectar decisiones futuras. *(Assets)*

Es un sistema flexible, amigable, con ayuda en línea, que puede ser instalado en una microcomputadora o sobre varias, funcionando en ambiente multiusuario incluidas estaciones remotas. Asimismo, proporciona opciones de seguridad que le permiten limitar el acceso a los diferentes procesos del sistema de acuerdo con el perfil de cada usuario. Se facilita el uso de la parametrización para adaptarse a las exigencias de cada entidad en particular, garantizando que sus reportes tengan la forma y el contenido que el usuario les defina. *(Assets)*

Está diseñado para Multi Compañía, con una estructura organizativa a varios niveles, en la que podrán existir: Grupo Corporativo, Corporativo, Grupo de Agrupaciones, Agrupación, Almacenes y Centros de Costos. Para entidades con esta estructura se brinda un Módulo de Comunicaciones que facilita poder intercambiar información entre ellas, con el fin de consolidar información sobre la Gestión Comercial y Contable, pudiéndose obtener los Estados Financieros, Resúmenes de Compras, Ventas, etc. a distintos niveles. *(Assets)*

1.3.3 Man Investments.

Es la División de Gestión de activos del Man Group plc, fundado en 1783, es reconocido mundialmente como proveedor de productos y soluciones de inversión alternativa[4] (por ejemplo, fondos de cobertura o “Hedge Funds”), además de ser uno de los corredores de futuros más importantes en el mundo. Man

[4] Inversión alternativa: Tiene como objetivo la inversión en largo plazo.

Group plc está cotizado en la bolsa de Londres y su empresa está incluida en el Índice FTSE 100[5]. *(JMS Investments)*

Man Investments Limited, miembro del grupo Man (Man Group plc), es un líder mundial en inversiones alternativas. Fue fundada en 1983, y en fecha de 31 marzo de 2006, tenía aproximadamente US[6] $ 49.9 billones bajo manejo. A través de los gerentes de sus fondos principales Man Investments ha logrado desarrollar fortalezas especializadas en el campo de inversiones alternativas. *(JMS Investments)*

1.4 Tendencias y tecnologías actuales.

Para el desarrollo del sistema que se desea implementar con el fin de gestionar la información referente al control de inversiones en la UPI, se tuvo en cuenta el análisis de las tecnologías y la propuesta planteada. A continuación se fundamentan las tecnologías de software empleadas y algunas definiciones que es importante conocer.

1.4.1 Metodologías de desarrollo de Software.

El objetivo de un proceso de desarrollo de software es elevar la calidad de este, a través de la transparencia y control sobre su desarrollo, y lograr que dichas pautas sean reproducibles en cada proyecto, independientemente de su magnitud. Actualmente, existe una gran cantidad de procesos de desarrollo, agrupados en dos tendencias: los métodos ágiles y los métodos pesados.

1.4.1.1 Rational Unified Process (RUP).

RUP (Proceso Unificado de Desarrollo) es uno de los procesos más generales que existen actualmente, su finalidad no está restringida a guiar desarrollo de software[7], sino cualquier tipo de proyecto. La

[5] Índice FTSE 100: Lo componen los 100 principales valores ("blue chips") de la Bolsa de Londres (London Stock Exchange). Ftse es un acrónimo de Financial Times Stock Exchange. El principal indicador del FTSE 100 es el FTSE 100 Index. El índice fue desarrollado con un nivel base de 1000.

[6] US: Dólar.

[7] Software: Es el conjunto de programas e instrucciones asociados a una computadora. La parte intangible que hace funcionar un sistema informático y que puede ser modificada con facilidad.

estrategia de este proceso es conseguir su objetivo por medio de orden y documentación, lo que lo convierte en el más fiel exponente de los métodos pesados. RUP define cuatro fases (inicio, elaboración, construcción y transición) y dentro de cada una de ellas el equipo de trabajo pasa por todos los flujos que son transversales a las fases, inclusive en varias iteraciones.

Flujos de trabajo *(Kruchten, 2003)***:**

Modelamiento del negocio: Describe los procesos de negocio, identificando quiénes participan y las actividades que requieren automatización.

Requerimientos: Define qué es lo que el sistema debe hacer, para lo cual se identifican las funcionalidades requeridas y las restricciones que se imponen.

Análisis y diseño: Describe cómo el sistema será realizado a partir de la funcionalidad prevista y las restricciones impuestas (requerimientos), por lo que indica con precisión lo que se debe programar.

Implementación: Define cómo se organizan las clases y objetos en componentes, cuáles nodos se utilizarán y la ubicación en ellos de los componentes y la estructura de capas de la aplicación.

Prueba (Testeo): Busca los defectos a los largo del ciclo de vida.

Instalación: Produce un entregable del producto y realiza actividades como empaque, instalación, asistencia a usuarios, etc. para entregar el software a los usuarios finales.

Administración del proyecto: Involucra actividades con las que se busca producir un producto que satisfaga las necesidades de los clientes.

Administración de configuración y cambios: Describe cómo controlar los elementos producidos por todos los integrantes del equipo de proyecto en cuanto a: utilización/actualización concurrente de elementos, control de versiones, etc.

Ambiente: Contiene actividades que describen los procesos y herramientas que soportarán el equipo de trabajo del proyecto; así como el procedimiento para implementar el proceso en una organización.

Fases del proceso (Kruchten, 2003)**:**

Inicio: Se describe el negocio y se delimita el proyecto, describiendo sus alcances con la identificación de los casos de uso del sistema.

Elaboración: Se define la arquitectura del sistema y se obtiene una aplicación ejecutable que responde a los casos de uso que la comprometen. A pesar de que se desarrolla a profundidad una parte del sistema, las decisiones sobre la arquitectura se hacen sobre la base de la comprensión del sistema completo y los requerimientos (funcionales y no funcionales) identificados de acuerdo con el alcance definido.

Construcción: Se logra un producto listo para su utilización que está documentado y tiene un manual de usuario. Se obtiene una o varias versiones del producto que han pasado las pruebas. Se ponen estos entregables a consideración de un subconjunto de usuarios.

Transición: La versión ya está lista para su instalación en las condiciones reales. Puede implicar reparación de errores.

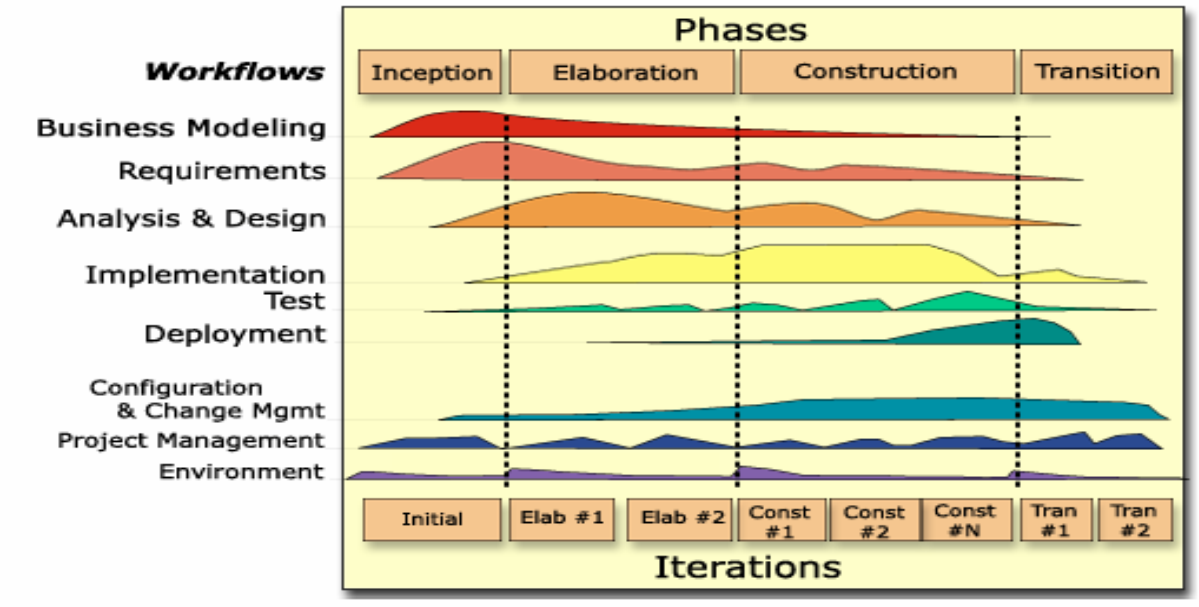


Figura 1.1: RUP en dos dimensiones.

El ciclo de vida de RUP se caracteriza por estar:

a. Dirigido por casos de uso: Los casos de uso reflejan lo que los usuarios futuros necesitan y desean, lo cual se capta cuando se modela el negocio y se representa a través de los requerimientos. A partir de aquí los casos de uso guían el proceso de desarrollo, ya que los modelos que se obtienen, como resultado de los diferentes flujos de trabajo, representan la realización de los casos de uso.

b. Centrado en la arquitectura: La arquitectura muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo, describiendo los elementos del modelo que son más importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente. RUP se desarrolla mediante iteraciones, comenzando por los casos de uso relevantes desde el punto de vista de la arquitectura. El modelo de arquitectura se representa a través de vistas(4+1)[8], o sea, perspectivas del sistema, que logran una abstracción particular en cada uno de los casos, en otras palabras, se trata de que en cada una de las llamadas vistas se represente el sistema en su totalidad, teniendo en cuenta solo determinados aspectos.

c. Iterativo e incremental: RUP propone que cada fase se desarrolle en iteraciones. Una iteración involucra actividades de todos los flujos de trabajo, aunque desarrolla fundamentalmente algunos más que otros. Las iteraciones hacen referencia a pasos en los flujos de trabajo, y los incrementos, al crecimiento del producto.

1.4.1.2 Extreme Programming (XP).

En el mundo está teniendo un gran impacto el uso de otra metodología de desarrollo de software, XP (Programación Extrema), que es clasificada como ágil y como tal trata de reducir la complejidad del producto orientando el trabajo directamente al objetivo, basado en las relaciones interpersonales y la velocidad de reacción, teniendo como principal peculiaridad la presencia, a tiempo completo en el desarrollo, de un representante del cliente.

[8] 4+1 vistas de la arquitectura: Vista de casos de uso, vista lógica, vista de procesos, vista de componentes y vista de despliegue.

XP define UserStories[9] como la base del software a desarrollar; estas historias son escritas por el cliente y describen las interacciones entre los clientes y el sistema, y generalmente son complementadas con otro tipo de descripción. A partir de ellas y de la arquitectura que se utilizará se planifican las entregas, así como los objetivos de cada una y las iteraciones con que contará. Esta planificación será valorada por el cliente y discutida hasta lograr el entendimiento en caso de que no esté de acuerdo desde un principio. Junto a los UserStories están los escenarios de prueba que comprobarán el correcto cumplimiento de las necesidades del cliente. El primer paso en una iteración es escribir las pruebas que se harán al código y que serán lo más automatizadas posible, para comprobar dinámicamente el software antes de cada entrega. Se usan mucho las pruebas de unidad para garantizar el funcionamiento de cada parte por sí sola.

Figura 1.2: Proyecto con XP.

[9] UserStories: Historias de usuario.

Una característica distintiva de XP es la programación en parejas, con el objetivo de que el código sea revisado y validado antes de ser escrito; la refactorización de código está presente durante todo el desarrollo, y no es más que escribir el mismo código fuente nuevamente buscando claridad, pero sin cambiar la funcionalidad resultante. Las parejas no serán siempre las mismas, sino que se pretende que cada desarrollador haya formado dupla al menos una vez con todos los demás, de donde se desprende que el código es de propiedad colectiva y cada uno es responsable por todo el proyecto. *(Letelier & Penades, 2006)*

En XP se programa solo la funcionalidad requerida para la entrega actual, excepto que se presente la necesidad de programar algo más flexible en un momento dado, se codifica buscando la simplicidad, y se hacen entregas frecuentemente. Es iterativo e incremental. De manera general se puede expresar que es recomendable usar XP en proyectos en los que los requisitos tienen altas probabilidades de cambiar con el tiempo (por ejemplo, porque el cliente no tiene claro lo que quiere, o porque el cambio de requisitos está ligado al dominio del problema a resolver), en proyectos con alto riesgo (por ejemplo, proyectos con una fecha de entrega que es indispensable cumplir, o proyectos totalmente novedosos para la industria) o en proyectos con un grupo pequeño de programadores (entre 2 y 12), aunque el equipo completo sea bastante más extenso (incluye a jefes de equipo y representantes de clientes). *(Letelier & Penades, 2006)*

1.4.1.3 Análisis de la selección de la Metodología de Desarrollo.

La metodología propuesta para continuar con el desarrollo del sistema es RUP ya que es un proceso que junto con el Lenguaje Unificado de Modelado (UML, por sus siglas en inglés, Unified Modeling Language), constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. Este es un proceso que garantiza la elaboración de todas las fases de un producto de software de este tipo.

RUP proporciona una guía para las actividades de un equipo de desarrollo, dirige las tareas de cada desarrollador por separado y ofrece criterios para el control, medición de los productos y actividades. Esta metodología de desarrollo al estar basada en una fuerte interacción con el cliente y usuarios, permite obtener productos adecuados a las necesidades reales, ahorrando esfuerzos y aumentando la satisfacción del usuario final.

1.4.2 Lenguaje de modelado.

Para el desarrollo de la aplicación se utilizará UML, como el lenguaje con que se modelarán los artefactos que se creen en el proceso de desarrollo del software.

UML es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software. UML ofrece un estándar para describir un plano del sistema, 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. En fin, UML es el enlace entre quien tiene la idea y el desarrollador, la comunicación es su principal objetivo. *(Popkin Software and Systems)*

Es importante resaltar que UML es un lenguaje para especificar y no para describir métodos o procesos. Se utiliza para definir un sistema de software, para detallar los artefactos en el sistema y para documentar y construir. En otras palabras, es el lenguaje en el que está descrito el modelo. Se puede aplicar en una gran variedad de formas para dar soporte a una metodología de desarrollo de software (tal como RUP), pero no especifica en sí mismo qué metodología o proceso usar. El UML está compuesto por diversos elementos gráficos que se combinan para conformar diagramas. Debido a que el UML es un lenguaje, cuenta con reglas para combinar tales elementos.

UML visualiza un sistema desde diferentes perspectivas *(Rumbaugh, et al., 2007)*:

Los **Diagramas de Estructura** enfatizan en los elementos que deben existir en el sistema modelado; incluye, entre otros, Diagrama de clases, Diagrama de componentes, Diagrama de objetos y Diagrama de paquetes.

Los **Diagramas de Comportamiento o Dinámicos** enfatizan en lo que debe suceder en el sistema modelado; ejemplo son el Diagrama de actividades y Diagrama de casos de uso.

Los **Diagramas de Interacción** son un subtipo de diagramas de comportamiento, que enfatiza sobre el flujo de control y de datos entre los elementos del sistema modelado; aquí se puede ver el Diagrama de

secuencia y de Diagrama de colaboración. El uso de UML y el trabajo continuo que se ha venido realizando sobre él lo convierten en una práctica que asegura la especificación de los procesos en el desarrollo del software, y es muy favorable su uso como vía de comunicación y documentación.

1.4.3 Herramientas CASE.

Las Herramientas CASE (Computer Aided Software Engineering, Ingeniería de Software Asistida por Computadora) son aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software, reduciendo su coste en términos de tiempo y dinero. Estas herramientas pueden ayudar en muchos de los aspectos del ciclo de vida de desarrollo del software, en tareas como la realización del diseño del proyecto, cálculo de costes, implementación de una parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores, entre otras. *(Visual Paradigm, 2008)*

Las herramientas CASE de modelado con UML permiten analizar y diseñar orientado a objetos, abstraer el código fuente, a un nivel donde la arquitectura y el diseño se tornan más obvios y más fáciles de entender y modificar. Cuanto más grande es un proyecto, más importante es utilizar una herramienta CASE.

Entre los principales beneficios que ofrece la utilización de una Herramienta Case se encuentran *(Visual Paradigm, 2008)*:

- Mejorar la comunicación entre usuario y especialista, ya que al tener incorporada la visualización de diagramas y de otras herramientas del análisis estructurado, actúa como un elemento que acelera la relación usuario/especialista.
- Permitir la facilidad de revisión de aplicaciones instaladas.
- Ser capaces de generar automáticamente las instrucciones del programa fuente. Esto permite acelerar el tiempo dedicado a la elaboración de programas y asegura además una estructura estándar y consistente para el programa.
- Dar la posibilidad de generar documentación técnica.

Dentro de la gama de herramientas de modelado se encuentran Rational Rose y Visual Paradigm, que son las más usadas en la actualidad.

1.4.3.1 Rational Rose.

Es la herramienta CASE desarrollada por los creadores de UML (Booch, Rumbaugh y Jacobson), que cubre todo el ciclo de vida de un proyecto: concepción y formalización del modelo, construcción de los componentes, transición a los usuarios y certificación de las distintas fases y entregables. El navegador UML de Rational Rose permite establecer una trazabilidad real entre el modelo (análisis y diseño) y el código ejecutable. Facilita el desarrollo de un proceso cooperativo en el que todos los agentes tienen sus propias vistas de información (4+1), pero utilizan un lenguaje común para comprender y comunicar la estructura y la funcionalidad del sistema en construcción. Rational Rose presenta un entorno de diseño muy atractivo y los cambios rápidos en el código a medida que se va desarrollando el modelado son muy buenos. Permite la generación de documentos en formato HTML[10], XML[11] y PDF[12] de los reportes. *(Pérez López & Iglesias Fernández, 2003)*

A pesar de que este producto está pensado para un ciclo evolutivo en espiral, se adapta muy bien al ciclo en cascada al que el proyecto se enfrenta. Independientemente del tipo de plataforma o aplicación (Eclipse[13], Java[14], .NET[15], aplicaciones integradas o de informática móvil), Rational complementa todo el proceso de principio a fin.

[10] HTML: Sigla en inglés de HyperText Markup Language (Lenguaje de Marcas de Hipertexto), es el lenguaje de marcado predominante para la construcción de páginas web.

[11] XML: Sigla en inglés de Extensible Markup Language (Lenguaje de Marcas Ampliable), es un metalenguaje extensible de etiquetas.

[12] PDF: Sigla en inglés de Portable Document Format (Formato de Documento Portátil), es un formato de almacenamiento de documentos.

[13] Eclipce: Entorno de desarrollo integrado de código abierto multiplataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido".

[14] Java: Lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los años 90.

[15] .NET: Plataforma de desarrollo de software con énfasis en transparencia de redes, con independencia de plataforma de hardware y que permita un rápido desarrollo de aplicaciones.

Tras configurar e instanciar la generación de código sobre una estructura de datos, se comprueba que esta generación se automatiza sobremanera; el programa intenta realizar acciones por su cuenta que no representan la voluntad del programador, por lo que es una tarea de especial cuidado.

1.4.3.2 Visual Paradigm (VP).

Visual Paradigm para UML es un galardonado producto que facilita a las organizaciones el diseño visual de los distintos diagramas a integrar y desplegar sus aplicaciones. Esta herramienta de desarrollo de software ayuda a los equipos de desarrollo en la confección de los distintos modelos que van desde la construcción hasta el despliegue, aumentando al máximo la productividad.

Es una herramienta diseñada para desarrollar software con programación orientada a objetos. Busca reducir la duración del ciclo de desarrollo, brindando ayuda a arquitectos, analistas, diseñadores y desarrolladores; permite el uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación y tiene capacidades de ingeniería directa e inversa.

Presenta una interfaz de uso intuitiva y con muchas facilidades a la hora de modelar los diagramas que soportan la Ingeniería de Requerimientos. Una de las características más importantes de su uso es que brinda la posibilidad de sincronización del modelo de diseño y el código en todo el ciclo de desarrollo una vez que se integra con el Eclipse, permitiendo la facilidad de programar directamente sobre el código fuente generado y a su vez actualizar el diseño con cambios que se realicen en la programación. Tiene una alta disponibilidad de múltiples versiones para cada necesidad, al igual que en las plataformas Linux[16], MacOS[17] y Windows[18]. *(Visual Paradigm, 2008)*

Presenta un innovador analizador textual donde se introduce texto extraído de conversaciones con el cliente, se definen actores, entidades, casos de uso disponibles para la generación de artefactos posteriores. Asimismo, posee una herramienta de generación de reportes en formato PDF o HTML

[16] Linux: Es un término genérico para referirse a sistemas operativos similares a Unix basados en GNU con el núcleo Linux.

[17] MacOS: Sistema operativo multitarea de Apple, basado en sistemas UNIX.

[18] Windows: Sistema operativo desarrollado por Microsoft.

configurable y selectiva, se integra con entornos como Eclipse, Hibernate[19] y Subversion[20], e importa o exporta formatos estándares de otras herramientas CASE como el Rational Rose. *(Visual Paradigm, 2008)*

1.4.3.3 Análisis de la selección de la herramienta CASE

El VP constituye un candidato más fuerte en la elección de una herramienta CASE para el sistema a desarrollar, debido principalmente a que iguala a Rational Rose en cuanto a capacidades y es mucho más integrable con el resto de las herramientas y estándares escogidos.

1.4.4 Sistemas Gestores de Bases de Datos (SGBD).

Un SGBD (en inglés: DataBase Management System) puede definirse como un paquete generalizado de software, que se ejecuta en un sistema computacional anfitrión, centralizando los accesos a los datos y actuando de interfaz entre los datos físicos y el usuario. Las principales funciones que debe cumplir un SGBD se relacionan con la creación y mantenimiento de la base de datos, el control de accesos, la manipulación de datos de acuerdo con las necesidades del usuario, el cumplimiento de las normas de tratamiento de datos, evitando redundancias e inconsistencias y manteniendo la integridad. Los SGBD permiten al programador convencional ahorrarse horas de trabajo dedicadas a la seguridad, gestión de los datos, chequeo de errores, etc.

Entre los SGBD comúnmente utilizados en el mundo tenemos MySQL[21], Oracle[22], PostgreSQL[23], Microsoft SQL Server[24], Interbase[25], entre otros. Todos estos presentan un enfoque relacional con un buen basamento matemático centrado en el Álgebra Relacional. *(Soto López & Saborit Ramírez, 2004)*

[19] Hibernate: Herramienta de Mapeo objeto-relacional para la plataforma Java.

[20] Subversion: Software de sistema de control de versiones.

[21] MySQL: SGBD relacional, multihilo y multiusuario con más de seis millones de instalaciones.

[22] Oracle: Es una herramienta cliente/servidor que se desarrolla para la gestión de Bases de Datos.

[23] PostgreSQL: SGBD relacional orientada a objetos de software libre, publicado bajo la licencia BSD.

[24] Microsoft SQL Server: SGBD basado en el lenguaje Transact-SQL.

[25] Interbase: Sistema de Administración de Base de Datos Relacionales (RDBMS) desarrollada y comercializada por la compañía Borland Software Corporation.

1.4.4.1 MySQL.

MySQL un sistema de gestión de base de datos relacional, multihilo y multiusuario. Es una idea original de las empresas open source[26] MySGL AB[27]. Desde enero de 2008 MySQL AB pertenece a Sun Microsystems[28]. MySGL AB desarrolla MySQL como software libre en un esquema de licenciamiento dual.

Por un lado se ofrece bajo la GNU/GPL[29]para cualquier uso compatible con esta licencia, pero las empresas que quieran incorporarlo en productos privativos pueden comprar a la empresa una licencia específica que les permita este uso.

MySQL es muy rápido, fiable y fácil de usar, surge para manipular bases de datos muy grandes. Es un sistema multiplataforma de base de datos relacionales, lo que da velocidad y flexibilidad, cuenta con un sistema muy seguro de privilegios que permite la autenticación básica para el acceso al servidor. Es muy compatible con lenguajes de programación Web[30] como el PHP.

1.4.4.2 Oracle.

Oracle es un Sistema de Gestión de Bases de Datos Relacional (SGBDR), fabricado por Oracle Corporation[31]. Es considerado uno de los sistemas de bases de datos más completos, por su estabilidad, soporte de transiciones, escalabilidad y ser multiplataforma, sin embargo el precio de Oracle es sumamente elevado.

Es la base de datos más utilizada actualmente. El acceso a los datos se otorga según privilegios concedidos por el administrador y es posible acceder a datos de Oracle usando software de otros

[26] Open source: Código abierto, es el término con el que se conoce al software distribuido y desarrollado libretmente.

[27] MySGL AB: Compañía de software fundada en 1995, creadora del sistema administrador de bases de datos relacionales MySQL, y una de las más grandes empresas de software libre del mundo.

[28] Microsystems: Empresa informática de Silicon Valley, fabricante de semiconductores y software.

[29] GNU/GPL: Acrónimo de Licencia Pública General de GNU o más conocida por su nombre en inglés GNU General Public License.

[30] Web: Sistema distribuido con mecanismos de hipertexto. Que permite mezclar texto, gráficos y archivos de sonido juntos.

[31] Oracle Corporation: Es una de las mayores compañías de software del mundo. Sus productos van desde bases de datos (Oracle) hasta sistemas de gestión.

fabricantes. Además cuenta con procedimientos muy buenos para hacer copias de seguridad y recuperar datos.

Para desarrollar en Oracle se utiliza PL/SQL[32], un lenguaje de quinta generación, bastante potente para tratar y gestionar la base de datos, y muy versátil.

1.4.4.3 PostgreSQL.

Es un magnífico gestor de bases de datos relacional orientado a objetos de software libre, se encuentra publicado bajo la licencia BSD[33] y es multiplataforma. Permite una fácil gestión de los usuarios y de las bases de datos que contenga el sistema. Sirve de soporte a los lenguajes más populares como PHP, C, C++, Java, Python, Ruby[34], entre otros, y al protocolo de comunicación encriptado por SSL[35]. El número de base de datos que puede contener es ilimitado. Se rige por el estándar ANSI-SQL 92/99. *(PostgreSQL)*

Posee una gran escalabilidad. Es la primera base de datos de código abierto en implementar recorrido sincronizado. Es capaz de ajustarse al número de Unidades Centrales de Procesamiento (CPUs) y a la cantidad de memoria que posee el sistema de forma óptima, haciéndolo capaz de soportar una mayor cantidad de peticiones simultáneas de manera correcta.

Implementa el uso de rollback[36], subconsultas y transacciones, haciendo su funcionamiento mucho más eficaz. Tiene la capacidad de comprobar la integridad referencial, así como también la de almacenar procedimientos en la propia base de datos. Además, PostgreSQL, permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos. *(Ventajas de PostgreSQL, 2005)*

---

[32] PL/SQL: Lenguaje de programación embebido en Oracle y PostgreSQL.

[33] BSD: Licencia de software otorgada principalmente para los sistemas BSD (Berkeley Software Distribution, en español, Distribución de Software Berkeley). Pertenece al grupo de licencias de software Libre.

[34] C, C++, Java, Python, Ruby: Lenguajes de programación.

[35] SSL: Sigla en inglés de Secure Sockets Layer, Protocolo de Capa de Conexión Segura. Proporciona cifrado de datos, autenticación de servidores e integridad de mensajes.

[36] Rollback: Es una operación que devuelve a la base de datos a algún estado previo.

1.4.4.4 Análisis de la selección del SGBD.

Luego de haber realizado un estudio detallado de los principales SGBD y valorando el entorno en que se va a desempeñar, así como la complejidad que presenta el modelo relacional propuesto, se optó para la implementación del sistema utilizar PostgreSQL por las siguientes razones:

- Mantiene Integridad referencial.
- Es extensible a las necesidades de los usuarios.
- Es multiplataforma.
- Se encuentra bajo licencia BSD.

1.4.5 Fundamentación del lenguaje de programación a utilizar.

La plataforma Java pertenecía a la empresa Sun Microsystem Inc. que en fecha 20 de abril de 2009 fue adquirida por Oracle Corporation, ambas empresas tienen nacionalidad norteamericana por lo que están sometidas a las normas de exportación y reexportación de los Estados Unidos de América. Estas normas impiden que algunos países, entre ellos Cuba, puedan adquirir productos informáticos. Las normas jurídicas tienen en todo momento un carácter nacional, por lo que las normas mencionadas sólo tienen aplicación en territorio norteamericano y son las empresas que ofertan estas tecnologías las que tienen que adoptar medidas para que estos productos no sean accesibles desde los países embargados. Por tal razón una vez que Cuba ha obtenido el producto informático, en este caso un producto gratuito, no tiene que cumplir con normas que no son cubanas ni que forman parte de ningún tratado bilateral con ese país.

Java es un lenguaje simple, orientado a objetos, distribuido, intérprete, robusto, seguro, de arquitectura neutra, portátil, de alto desempeño, de hilos múltiples y dinámico. *(Flanagan, 1999)*

Análisis de sus características *(Flanagan, 1999)***:**

Orientado a objetos: Para el programador, esto significa que debe poner atención especial a los datos de la aplicación y a los métodos que manipulan los datos, y que no debe pensar estrictamente en términos de procedimientos.

Intérprete: el compilador de Java genera bytecode[37] para la máquina virtual de Java (JVM-Java Virtual Machine), en vez de código nativo de máquina. Para que se ejecute realmente un programa de Java, se usa el intérprete de Java para ejecutar los bytecode compilados. Como los bytecode de Java no dependen de la plataforma, los programas de Java se pueden ejecutar en cualquier plataforma en la que la JVM se haya instalado (el intérprete y el sistema de tiempo de ejecución).

Arquitectura neutral y portátil: Como los programas de Java se compilan en un formato de bytecode de arquitectura neutral, una aplicación de Java se puede ejecutar en cualquier sistema, siempre y cuando dicho sistema instrumente la máquina virtual de Java. Esto resulta muy importante para las aplicaciones. Toda aplicación de Java se podrá ejecutar sobre cualquier plataforma. El hecho de que Java sea intérprete y defina un formato de bytecode estándar, de arquitectura neutral, es lo que le da la característica de ser portátil. Pero Java va más allá, pues asegura que no haya aspectos "dependientes de instrumentación" en la especificación del lenguaje.

Dinámico y distribuido: En todo momento, una clase de Java se puede cargar en un intérprete de Java en ejecución. Dinámicamente se pueden crear casos de estas clases cargadas. Las bibliotecas de código nativo también se pueden cargar de manera dinámica. Las clases en Java se representan mediante la clase Class; usted puede obtener información dinámicamente acerca de una clase al momento de la ejecución. A Java también se le denomina lenguaje distribuido. Esto significa, simplemente, que proporciona un soporte de alto nivel para redes.

Simple: Los diseñadores de Java intentaron crear un lenguaje que un programador pudiera aprender con rapidez, por lo que el número de constructores del lenguaje se ha mantenido relativamente bajo. Otra meta del diseño fue lograr que el lenguaje fuese familiar a la mayoría de los programadores, para facilitar la migración.

Robusto: Java se ha diseñado para escribir software robusto y muy confiable. Java ciertamente no elimina la necesidad de asegurar la calidad del software, ya que todavía es posible escribir software defectuoso en Java. Sin embargo, Java sí elimina ciertos tipos de errores de programación, cosa que hace

[37] Bytecode: Código intermedio más abstracto que el código máquina. Habitualmente es tratado como un fichero binario que contiene un programa ejecutable similar a un módulo objeto, que es un fichero binario producido por el compilador cuyo contenido es el código objeto o código máquina.

considerablemente más sencilla la escritura de software confiable. Java es un lenguaje de tipografía rígida, por lo que permite una verificación exhaustiva al tiempo de la compilación, en busca de posibles problemas de no concordancia de letras.

Seguro: Uno de los aspectos más resonados de Java es que se trata de un lenguaje seguro. Esto es especialmente importante, dada la naturaleza distribuida de Java. Sin una garantía real de la seguridad, usted no querría descargar código de un sitio al azar en la Internet, ni lo dejaría ejecutar en su computadora. Java se diseñó con la seguridad en mente, y proporciona varias capas de controles de seguridad que protegen contra código malicioso; estas capas permiten a los usuarios ejecutar con comodidad programas desconocidos.

Para llevar a cabo el desarrollo de la aplicación, que es objeto de esta investigación, se decidió el uso del lenguaje de programación Java por las ventajas y potencialidades que tiene frente a otros lenguajes, al poder implementarse con independencia de la plataforma y como software libre, permitiendo obtener productos de excelente calidad, en menor tiempo y, por consiguiente, con menores costos.

1.4.6 Entorno de desarrollo integrado.

En inglés Integrated Development Environment (IDE), es un programa compuesto por un conjunto de herramientas para un programador. Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz de usuario gráfica (GUI). Los IDEs pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes.

Los IDE proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación tales como C++, Python, Java, C#, Delphi, Visual Basic, etc. En algunos lenguajes, un IDE puede funcionar como un sistema en tiempo de ejecución, en donde se permite utilizar el lenguaje de programación en

forma interactiva, sin necesidad de trabajo orientado a archivos de texto, como es el caso de Smalltalk[38] u Objective-C[39].

Puede dedicarse exclusivamente para un lenguaje de programación o bien para varios. Las opciones más atractivas para la elección de un IDE son Eclipse y NetBeans[40], que gozan de reconocimiento a nivel mundial.

1.4.6.1 Eclipse.

Eclipse es un IDE de código abierto independiente de una plataforma, es una aplicación de cliente enriquecido, emplea plug-ins[41] para proporcionar toda su funcionalidad, a diferencia de otros entornos monolíticos donde las funcionalidades están todas incluidas, las necesite el usuario o no. Eclipse es, en el fondo, únicamente una armazón sobre la que se pueden montar herramientas de desarrollo. La arquitectura de plug-ins permite, además de integrar diversos lenguajes, introducir otras aplicaciones accesorias que pueden resultar útiles durante el proceso de desarrollo, tales como: herramientas UML, editores visuales de interfaces, ayuda en línea para librerías, entre otros. *(Burnette, 2005)*

1.4.6.2 NetBeans.

NetBeans es una herramienta de desarrollo para aplicaciones, escrita puramente sobre la base de la tecnología Java, de modo que puede ejecutarse en cualquier ambiente que ejecute Java. Es un producto libre y gratuito, sin restricciones de uso y de código abierto.

[38] Smalltalk: es un sistema informático que permite realizar tareas de computación mediante la interacción con un entorno de objetos virtuales. Metafóricamente, se puede considerar que un Smalltalk es un mundo virtual donde viven objetos que se comunican mediante el envío de mensajes.

[39] Objective-C: es un lenguaje de programación orientado a objetos creado como un superconjunto de C pero que implementase un modelo de objetos parecido al de Smalltalk.

[40] NetBeans: Plataforma para el desarrollo de aplicaciones de escritorio usando Java y a un entorno de desarrollo integrado.

[41] Plug-ins: Complemento de una aplicación que se relaciona con otra para aportarle una función nueva y generalmente muy especifica. Esta aplicación adicional es ejecutada por la aplicación principal e interactúan por medio de la API.

Soporta el desarrollo de todos los tipos de aplicaciones Java (J2SE[42], Web, EJB[43] y aplicaciones móviles). Existe un número importante de módulos para extenderlo. Entre sus características se encuentra un sistema de proyectos basado en control de versiones y refactorización. *(NetBeans)*

Todas las funciones del IDE son provistas por plug-ins, al igual que Eclipse. La versión 6.5 incluye numerosas mejoras y soportes a un nuevo lenguaje. Algunas de las novedades a señalar son *(Marioko, 2008):*

- Soporte para Groovy[44].
- Soporte para Grails[45].
- Mejoras en la gestión de bases de datos.
- Actualización a la versión 1.7 de Ant[46], gracias a esta se compilan los proyectos mucho más rápido.
- Quick Find[47], que permite buscar funcionalidades del IDE utilizando un buscador.
- Integración a Glassfish 3 Prelude[48].
- Permite desarrollar fácilmente todo tipo de aplicaciones de escritorio, empresariales, móviles y WEB, con los lenguajes C/C++, PHP, JavaScript[49], Java, Groovy y Ruby, en las plataformas Windows, Linux, Mac OS X y Solaris[50].

Entre sus nuevas características se encuentran: Mejoras al soporte de PHP, JavaScript, Ajax[51], soporte mejorado para Hibernate[52], JPA[53], e integración con Glassfish v3 y MySQL. *(NetBeans.org , 2008)*

---

[42] J2SE: Siglas en inglés de Java 2 Standard Edition. Edición Estándar. La versión estándar es la más común y cuenta con todo lo necesario para desarrollos de software y acceso a aplicaciones Java.

[43] EJB: Los Enterprise JavaBeans son una de las API que forman parte del estándar de construcción de aplicaciones empresariales J2EE de Sun Microsystems.

[44] Groovy: Lenguaje de programación orientado a objetos implementado sobre la plataforma Java.

[45] Grails: Framework (estructura de soporte) para aplicaciones web de código abierto desarrollado sobre el lenguaje de programación Groovy.

[46] Ant: Herramienta usada en programación para la realización de tareas mecánicas y repetitivas, normalmente durante la fase de compilación y construcción.

[47] Quick Find: Es un sencillo menú utilitario de la barra del sistema.

[48] Glassfish 3 Prelude: Servidor modular basado en Java.

[49] JavaScript: Lenguaje de programación interpretado, es decir, que no requiere compilación, utilizado principalmente en páginas web.

[50] Solaris: Sistema operativo de tipo Unix desarrollado por Sun Microsystems desde 1992 como sucesor de SunOS.

[51] Ajax: Acrónimo de Asynchronous JavaScript And XML (JavaScript asíncrono y XML),

1.4.6.3 Análisis de la selección del entorno de desarrollo.

Ambos IDE son similares en cuanto a las funcionalidades y ventajas que ofrecen para el programador como delante está claro que la nueva versión liberada del NetBeans cuenta con mejores prestaciones, nuevas funcionalidades incorporadas y brinda mayores facilidades para el diseño de las aplicaciones de escritorio, por lo tanto, como IDE seleccionado para desarrollar la aplicación se escogió el NetBeans.

1.4.7 Framework utilizado en la solución.

Antes de llegar a la definición de qué es un framework, es imprescindible mencionar los conceptos de componente y componente de software.

Un componente es un bloque de construcción reusable de software: una pieza de código encapsulada de una aplicación que puede ser combinada con otros componentes y con códigos adicionales para producir una aplicación específica. Los componentes pueden ser simples o complejos. No existe un acuerdo general sobre qué es o qué no es un componente. Ellos vienen en una variedad de formas y tamaños. Un componente puede ser muy pequeño, como lo es una GUI widget[54] (por ejemplo, un botón), o este puede implementar un complejo servicio de una aplicación, tal como una función para administrar la red. *(Kaiser, 2005)*

El concepto comúnmente aceptado de un componente es llamado “componente binario”, el cual es un artefacto de software compilado que es integrado dentro de la aplicación en tiempo de ejecución (runtime). El código fuente no es usualmente disponible, por tanto los componentes no pueden ser modificados. Si los componentes están implementados en un lenguaje de programación orientado a objeto, ellos pueden ser extendidos a través de la herencia o delegación. *(Kaiser, 2005)*

[52] Hibernate: Herramienta de Mapeo objeto-relacional para la plataforma Java.

[53] JPA: Es la API de persistencia desarrollada para la plataforma Java EE.

[54] Widget: Objeto cuyo nombre no se sabe o se olvidó; símbolo gráfico de interface que permite interacción entre el usuario y el ordenador (símbolo, Icono, ventana y etcétera.).

Una aplicación está compuesta por una colección de componentes. Las aplicaciones proveen un ambiente, glue code[55] o un esqueleto de código, dentro del cual los componentes son insertados. Los componentes interactúan con su ambiente y pudieran interactuar con el sistema operativo de la computadora sobre la cual ellos son ejecutados. Además, ellos interactúan unos con otros a través de interfaces. Cambiar una interfaz de un componente pudiera requerir cambiar el componente que usa esta interface. Sin embargo, cambiar la implementación del componente no debería requerir cambiar a los clientes de ese componente. *(Kaiser, 2005)*

Un componente de software es un paquete entregable independiente de servicios de software, en sentido general. Philippe Kruchten of Rational Software cree que un componente no es trivial, que es casi independiente y una parte reemplazable de un sistema que satisface una función clara en el contexto de una arquitectura bien definida. Un componente es para conformar y proveer la realización física de un conjunto de interfaces. Por otro lado Szyperski en su libro "Component-Based Software: Beyond Object-Oriented Programming", ve a un componente de software como una unidad de composición con interfaces especificadas contractualmente y un explícito contexto de dependencias. Un componente de software puede ser desarrollado independientemente y estar sujeto a composición de terceras partes. De esta manera, un componente de negocio representa la implementación de software de una operación autónoma de negocio o proceso. *(Kaiser, 2005)*

Recientemente, el interés en reutilizar software ha sido cambiado de la reutilización de componentes simples a diseño de sistemas enteros o estructuras de aplicaciones. Un sistema software que pudiera ser reutilizado en este nivel para la creación de aplicaciones completas es llamado framework. Los frameworks son basados en la idea de que deberían permitir la producción fácil de un conjunto de sistemas específicos pero similares, dentro de un cierto dominio comenzando desde una estructura genérica. Brevemente, los frameworks son arquitecturas genéricas integradas por un extensible conjunto de componentes. Además, los frameworks pueden contener subframeworks que representen subconjuntos de componentes de un sistema más grande. *(Kaiser, 2005)*

[55] Glue code: en términos de programación, es el código que no realiza ninguna funcionalidad para satisfacer algún requerimiento del programa en cuestión.

Estos enfatizan en la reutilización del diseño sobre el código, aunque el código resultante puede ser reutilizado. Usualmente definen la estructura total de todas las aplicaciones derivadas de él, una organización entre clases y objetos, principales responsabilidades de clases individuales, cómo ellos colaboran y el hilo de control de estas. El desarrollador es el responsable para personalizar el framework para una aplicación en particular. *(Kaiser, 2005)*

Los frameworks invierten el rol de control entre la aplicación y la infraestructura, es decir la infraestructura llama o invoca rutinas de la aplicación. En sistemas convencionales, los propios desarrolladores de programas proveen toda la estructura y control del flujo de ejecución y realizan llamadas a librerías de funciones según sea necesario; pero usando un framework, los desarrolladores tienen solamente que preparar estos componentes que no están en el framework acorde al comportamiento deseado de la aplicación. *(Kaiser, 2005)*

Un framework contiene clases o estructuras que implementan los componentes de una aplicación genérica también como componentes concretos que satisfacen tareas especializadas. Para desarrollar programas completos, los desarrolladores buscan e instancian los componentes apropiados.

No hay una definición común para los frameworks, pero la mayoría tiene un tema común: la reutilización. Una definición ampliamente aceptada es dada por R. E. Johnson, and B. Foote en 1988 en su publicación "Designing Reusable Classes *(Johnson & Foote, Designing Reusable Classes, 1988)*:

> *"Un framework es un conjunto de clases que personifican un diseño abstracto para soluciones de una familia de problemas relacionados..."*

Otro punto de vista, según R. E. Johnson y V. F. Russo en "Reusing Object-Oriented Designs" incluye *(Johnson & Russo, Reusing Object-Oriented Designs, 1991)*:

> *"Una clase abstracta es un diseño para un objeto simple. Un framework es el diseño de un conjunto de objetos que colaboran para llevar a cabo un conjunto de responsabilidades. De esta manera los frameworks son diseños a escalas más grandes que las clases abstractas. Los frameworks son una forma de reutilizar el diseño a un nivel superior."*

Hibernate

Hibernate es un potente framework de mapeo objeto/relacional y servicio de consultas para Java. Es la solución ORM (Object-Relational Mapping) más popular en el mundo Java. Permite diseñar objetos persistentes que podrán incluir polimorfismo, relaciones, colecciones, y un gran número de tipos de datos. De una manera muy rápida y optimizada podremos generar bases de datos en cualquiera de los entornos soportados: PostgreSQL, Oracle, MySQL, etcétera.

Sus principales características son:

- Permite expresar consultas utilizando SQL nativo o consultas basadas en criterios.
- Soporta todos los sistemas gestores de bases de datos SQL y se integra de manera elegante y sin restricciones con los más populares servidores de aplicaciones J2EE[56] y contenedores web, y por supuesto también puede utilizarse en aplicaciones de escritorio.
- Puede operar proporcionando persistencia de una manera transparente para el desarrollador.
- Soporta el paradigma de orientación a objetos de una manera natural: herencia, polimorfismo, composición y el framework de colecciones de Java.
- Soporte para modelos de objetos con una granularidad muy fina Permite una gran variedad de mapeos para colecciones y objetos dependientes.
- Provee un sistema de caché de dos niveles y puede ser usado en un clúster. Permite inicialización perezosa (lazy) de objetos y colecciones.
- Proporciona el lenguaje HQL[57] en cual provee una independencia del SQL de cada base de datos, tanto para el almacenamiento de objetos como para su recuperación.

[56] J2EE: (Java 2 Enterprise Edition). Define un estándar para el desarrollo de aplicaciones empresariales multicapa diseñado por Sun Microsystems. J2EE simplifica las aplicaciones empresariales basándolas en componentes modulares y estandarizados, proveyendo un completo conjunto de servicios a estos componentes, y manejando muchos de las funciones de la aplicación de forma automática, sin necesidad de una programación compleja.

[57] HQL: Hibernate Query Language, está diseñado como un "mínimo" de extensión orientada a objetos a SQL, proporciona un puente entre el objeto y las bases de datos relacionales.

- Presenta un potente mecanismo de transacciones de aplicación llegando incluso a permitir la interacciones largas (aquellas que requieren la interacción con el usuario durante su ejecución).
- Soporta los diversos tipos de generación de identificadores que proporcionan los sistemas gestores de bases de datos así como generación independiente de la base de datos, incluyendo identificadores asignados por la aplicación o claves compuestas.

Se decidió el uso de Hibernate para una solución como la nuestra argumentando que es un poderoso medio para mapear clases de Java a tablas de la base de datos, desacopla la base de datos de las clases de la aplicación. Hibernate también provee con HQL una poderosa vía de comunicación entre el programador y la base de datos, al dotarlo de un lenguaje invariante con respecto a esta y además sintácticamente muy parecido al SQL. Hibernate además es de código abierto y la licencia del producto está eximida de costo.

1.5 Conclusiones.

A lo largo de este capítulo se ha realizado un estudio de los sistemas contables, dejando claro que estos sistemas son para controles contables empresariales y no para controlar un proceso inversionista en una unidad presupuestada en toda su magnitud.

Para efectuar el diseño del sistema propuesto, se hace uso de RUP modelando con UML 2.0 y utilizando como herramienta CASE a Visual Paradigm 6.0, que en su conjunto conforman en la actualidad una de las metodologías más utilizadas en el desarrollo de grandes proyectos.

De las tecnologías que han sido objeto de estudio en este capítulo se seleccionó un grupo de ellas para conformar la propuesta tecnológica y así desarrollar un sistema robusto, multiplataforma y extensible, que permita a los clientes finales un entorno de trabajo amigable y flexible.

- Sistema gestor de base de datos: PostgreSQL (Versión 8.3, 4 de febrero de 2008).
- Lenguaje de programación: Java.
- Framework: Hibernate.
- Entorno de desarrollo integrado: NetBeans (Versión 6.5).

CAPÍTULO 2 CARACTERÍSTICAS DEL SISTEMA

2.1 Introducción.

En el presente capítulo se hace un estudio de la UPI “Proyecto Futuro”, se enuncian y describen los procesos de negocio, se detallan los que serán objetos de automatización, así como la información que se maneja, además de describirse los actores y trabajadores que intervienen en el mismo, así como la realización de los diagramas de casos de uso del negocio, diagramas de actividades y el modelo de objeto. También se identifican cuáles son los requisitos funcionales y no funcionales que debe cumplir la aplicación. Se muestra además una descripción detallada de los casos de uso del sistema. De forma general se abordan las características del sistema dejando muy clara la propuesta final que se pretende.

2.2 ¿Qué es la UPI “Proyecto Futuro”?

La UPI “Proyecto Futuro” es una entidad jurídicamente independiente creada el 16 de marzo de 2007 con efectos legales a partir del 9 de marzo de 2007, con posibilidades de manejar cuentas en Moneda Nacional (MN) y Divisa (CUC), capaz de dirigir el proceso inversionista de la UCI y sus Facultades Regionales, con todas las obras que se indiquen desde el Organismo Rector, según el financiamiento recibido, y con facultades para desempeñar correctamente las funciones que se le asignen.

Conceptualmente, la demanda financiera de la UPI “Proyecto Futuro”, es sufragada con fondos centralizados del Estado Cubano, para el Programa de la Batalla de Ideas, siendo aplicable el mecanismo de solicitud, aprobación y autorización para su otorgamiento, definido por el Ministerio de Economía y Planificación.

La UPI “Proyecto Futuro” es la responsable de que se cumpla la Resolución No 91, del 16/03/06 del MEP para las inversiones que atiende y es la entidad responsable de la inversión desde su concertación inicial

hasta su puesta en explotación y terminación total, supervisando y controlando, la marcha adecuada de todo el proceso en sus diferentes etapas.

Tiene como **objeto social**:

- Normar, planificar, preparar, y controlar el proceso inversionista del Proyecto Futuro, que incluye la Universidad de las Ciencias Informáticas, sus Facultades Regionales y cualquier otro caso que así lo requiera y autorice el Organismo Rector.
- Brindar servicios de dirección integrada y/o administración de proyectos de inversiones en pesos cubanos.
- Prestar servicios de supervisión técnica-ingenieril, dirección facultativa de obras y procuración, evaluación, gestión y provisión de ofertas de suministros en pesos cubanos.
- Prestar servicios de consultoría y asistencia técnica en estimaciones y presupuestos económicos, estudios técnicos económicos de preinversión y postinversión, localización y mercado, en pesos cubanos.

2.2.1 Objetivos de la UPI “Proyecto Futuro”.

Tiene como **objetivos principales**:

- Dirigir y organizar el proceso inversionista de la UCI, desde la perspectiva de áreas funcionales capaces de gestionar y controlar todos los procesos de la inversión desde las ideas conceptuales, proyectos, contrataciones, planeación y ejecución, hasta la creación de un nuevo activo tangible funcional y estético, con la completa observancia de la legalidad.
- Preservar, ahorrar, y utilizar con la mayor eficiencia, calidad y rentabilidad los recursos materiales y financieros puestos a disposición de la actividad.
- Preparar, elaborar y controlar la ejecución del Plan de Inversiones de la UCI, tanto en sus aspectos financieros como del avance físico de su materialización.
- Tener bajo su jurisdicción y mando directo a las Direcciones Integradas de Proyecto que atienden las Inversiones del Plan Director de la Universidad.
- Crear o modificar las DIP adscriptas a su estructura funcional.

- Velar por el cumplimiento de la legislación vigente, normas, instrucciones, procedimientos y su actualización, emitidos por los Organismos de la Administración Central del Estado en lo referente al Proceso Inversionista y su ejecución.
- Asegurar la compatibilización de la marcha del proceso Inversionista con las diferentes entidades que integran la UCI y las instituciones del estado.
- Participar y coordinar con el resto de las Unidades Organizativas de la UCI en especial con las vicerrectorías de Atención a la Ciudad y Económica, las funciones principales que se interrelacionen con las Inversiones que atiende.
- Establecer, en los diferentes niveles que se le subordinan, los requisitos de personalidad jurídica a otorgar para celebrar los contratos, definidos por el Reglamento del Proceso Inversionista.
- Garantizar la emisión de informaciones periódicas a la dirección de la UCI, al Organismo Rector y a los Órganos correspondientes de la Administración Central del Estado.

2.2.2 Procesos claves en la actividad inversionista.

1. **Programación:** elaboración y aprobación de ideas conceptuales.

2. **Diseño:** elaboración de los proyectos técnico ejecutivos.

3. **Preparación técnica:** elaboración de presupuestos, cronogramas de ejecución y balances de equipos y fuerza de trabajo. Garantiza el trabajo de solicitud de financiamiento que es responsabilidad del Inversionista Principal.

4. **Control de la ejecución:** incluye el seguimiento detallado de los cronogramas de ejecución, el control de los presupuestos, la certificación para proceder a los pagos y el control de la calidad.

5. **Entrega:** garantiza la entrega adecuada del objeto de obra terminado a la propiedad y al sistema de mantenimiento de la institución.

El inversionista de una Entidad Presupuestada es el responsable de custodiar los recursos financieros de los que el país dispone para una determinada inversión los cuales deben ser administrados de forma racional y consciente para optimizarlos, entre estos recursos también se encuentra el tiempo. Para poder

cumplir con tal responsabilidad se deben llevar controles que vayan desde los alcances pactados para el año, cuanto se va ejecutando en el mes, cuanto ya ha sido aprobado y pagado y cuanto dinero se ha dado por anticipado al constructor.

Actualmente es muy difícil llevar de manera efectiva y con la rapidez de respuesta que se requiere todo el control de la información que de forma eficiente requiere el proceso inversionista de una entidad presupuestada. La misma trabaja con el dinero del Estado y no puede, bajo ningún concepto, desviarse o excederse de los montos y alcances, que le fueron aprobados para el período. Por tanto, el control del plan es la tarea fundamental que lleva cada inversionista, alrededor del mismo gira todo el proceso de control.

El flujo de información y su control se realiza por tablas en Excel y documentos de recepción de forma manual, el único sistema que se emplea es el sistema contable ASSET que por ser su fin la contabilidad de la empresa, no tiene en cuenta muchos datos por no representar interés a la parte económica, pero que sí revisten especial importancia para el control de los inversionistas a sus obras.

2.3 Objeto de automatización.

Este sistema tiene como objetivo automatizar todo el seguimiento de la documentación generada en el proceso productivo del proceso inversionista con vista a establecer un control más detallado y riguroso, que permita ampliar la cantidad de datos e información a entregar.

El procedimiento que da inicio a todos los otros y del cual depende el sistema para su correcta ejecución es el de la solicitud y aprobación de los códigos de obra. Esto es lo primero que se realiza una vez aprobada la inclusión de una determinada obra en el plan de inversiones del año. La solicitud es llenada por la DIP y entregada en el Dpto. de Control, quien una vez aprobado lo entrega a la Subdirección Económica (SDE), que son los encargados de otorgarle un número de acuerdo a la DIP y al consecutivo que le corresponda. Sin este código, ningún proceso técnico económico podría llevarse a cabo, pues toda la documentación del proceso inversionista debe ser registrada y controlada por obra.

El proceso más importante es el de certificación y facturación que se inicia en el momento en que dicha documentación es entregada a la Inversión por parte de la Contratista General para su revisión, aprobación y posterior pago como parte de la aceptación del trabajo de construcción realizado en el mes por las diferentes brigadas constructoras. De esta documentación se controla la fecha en que es recibida por la Inversión, quien solamente cuenta con 5 días hábiles para su revisión; el importe del valor certificado que corresponde con el valor de producción del mes.

La factura tiene un número único e irrepetible que sirve como ID de la documentación para su registro y seguimiento. Una vez aprobada por la Inversión la certificación y la factura se procede a solicitar su pago al Departamento de Finanzas, el cual emite un documento de Solicitud de Pago que contiene un número de folio y al documento a pagar se le da un informe de recepción (IR) en el momento en que es introducido en el sistema contable ASSET. El proceso concluye con la emisión del cheque como pago efectivo de la ejecución respaldada con el documento primario.

Otro proceso es el de los registros de contratos[58], el cual es llevado a cabo por el Área Jurídica y permite conocer el estado de la contratación por obra, cuáles y cuántos han sido presentados por parte del constructor o contratista y de estos cuántos y cuáles ya han sido aprobados y firmados por la Inversión estableciéndose un compromiso formal de acuerdo entre las partes, tal y como establece la legislación vigente en el país para el proceso inversionista. A través de las facturas se controla la ejecución de lo pactado en el contrato.

En estos momentos los procesos económicos son llevados a cabo por la SDE quien lo registra en su sistema contable, el cual como ya ha sido expuesto en el Capítulo 1, no ofrece todos los detalles de la documentación que requiere el inversionista para su control, por lo que este debe llevar un control paralelo en tablas dinámicas en Excel, el cual es comparado con el registro económico de manera mensual y es la llamada conciliación.

[58] Contrato: Documento en el que se establece de los trabajos a realizar y el valor de los mismos. Así como el tiempo en que serán ejecutados (cronograma de ejecuciones).

2.4 Información que se maneja.

Los documentos que son procesados son las solicitudes de código de obra, el plan aprobado de la inversión para el año por obra y objetos de obras, contra el cual se compararán las ejecuciones en el año y el cual tiene carácter de límite máximo para cada obra para ejecutar en el año; las certificaciones, ofertas y facturas que son las que dan fe de las ejecuciones realizadas y constituyen un compromiso de pago por parte de la inversión una vez que las aprueba; los contratos que son quienes establecen las relaciones entre las entidades y que son de carácter indispensable como establece la legislación del país para el proceso inversionista.

2.5 Propuesta de sistema.

Teniendo en cuenta que la información que se genera es manipulada de forma manual y no dispone de una herramienta que le permita llevar estos controles de una forma más sencilla, se propone la elaboración de un sistema informático que brinde soporte a los procesos de control de inversiones de la UPI descritos anteriormente, siendo estos procesos sus principales funcionalidades.

Esta aplicación expondrá funciones que le permitirán al inversionista gestionar el objeto de obra, controlar las facturas generadas durante el desarrollo de todo el proceso inversionista, permitiéndole agregar, modificar las facturas y obtener las que faltan por definir contrato, listar la facturación por obras y definir el estado de la factura, permitirá tener un control detallado de cómo está la inversión en el momento deseado.

La mayor ventaja que tendrá el nuevo sistema informático que se propone es que será una aplicación de escritorio que podrá ser utilizado por los Directivos de la entidad, los Especialistas, el Asesor Jurídico y el Dpto. de Control y que tendrá las siguientes ventajas respecto a cualquier aplicación Web:

- Navegación e interfaz de usuario más rápida.
- Los usuarios no necesitan acceder a la aplicación desde un lugar diferente a su puesto de trabajo ya que estas herramientas se instalan directamente en la PC del usuario.
- No se necesita disponer de Internet o Intranet para acceder a la aplicación.

- Fácil acceso a recursos locales: disco duro, memoria de vídeo, sonido, ratón, teclado, etc.
- Son aplicaciones multiplataforma, es decir, pueden funcionar en diferentes sistemas operativos y/o ordenadores, pero el código fuente es el mismo. Da la libertad al usuario de poder utilizar la máquina que más le guste o la que necesite usar.

Esta aplicación tendrá por nombre “ConInver” asociado a la unión de las palabras Control e Inversiones.

2.6 Modelo de Negocio.

El modelo del negocio es una técnica para comprender los procesos del negocio de la organización. El objetivo fundamental del mismo es identificar los casos de uso del software y las entidades relevantes que este debe soportar de forma que se podría modelar solo lo necesario para comprender el contexto.

2.6.1 Reglas del negocio.

Las reglas de negocio describen políticas que deben cumplirse o condiciones que deben satisfacerse, por lo que regulan algún aspecto relacionado con el mismo. El proceso de especificación implicó identificarlas dentro del negocio, evaluar si son relevantes dentro del campo de acción que se está modelando e implementarlas en la propuesta de solución. En este caso se identificaron las siguientes reglas que debe seguir la aplicación que se desarrolle, a fin de respetar y garantizar las restricciones que existen en el negocio:

- La aprobación del código de objeto de obra es lo primero que se realiza una vez aprobada la inclusión de una determinada inversión en el plan del año en curso.
- La solicitud de código de objeto de obra es llenada por la DIP.
- A la solicitud de código de objeto de obra se le otorga un número de acuerdo a la DIP y al consecutivo que le corresponda.
- Sin el código de objeto de obra ningún proceso técnico económico podría llevarse acabo.
- Si la obra se aprueba se debe contratar quien ejecute estos trabajos.
- Una vez aprobada por la Inversión la certificación y la factura se procede a solicitar su pago al departamento contable.
- Para revisar una factura, el Especialista cuenta solamente con 5 días hábiles.

2.6.2 Procesos del negocio.

Aprobación del código de obra. Este proceso comienza cuando la DIP le envía la solicitud de código de objeto de obra impresa y acuñada al Dpto. de Control. Este revisa la solicitud que de ser aprobada se remite a la Subdirección Técnica (SDT) para que la valide y después de ser firmada es devuelta al Dpto. de Control. El Dpto. de Control la entrega a la SDE quien le asigna el código y la devuelve. El Dpto. de Control saca dos copias del documento, el original va al Dpto. de Contabilidad, una de las copias es entregada a la DIP para la confección del expediente de inversiones, y la segunda copia con el recibido de la DIP, que no es más que el cuño y firma de la persona que recibe la copia en la DIP, es archivada en el Dpto. de Control, para posteriores análisis. Uno de los problemas principales que existen en este proceso del negocio es que no se lleva el control de códigos de obra abiertos, cerrados o en garantía, en qué fecha se abrió o cerró el mismo, además de las observaciones, que pueden ser las causas por las que se cierra el código de obra.

Aprobación del contrato. Es otro de los procesos que se llevan a cabo en la entidad. Este comienza cuando el Contratista le entrega al Asesor Jurídico el contrato. Este contrato cuenta con 3 copias, una de estas es entregada a la DIP, otra al Dpto. de Control y la otra a la SDE para que sea revisada por estos. En un comité de contratación compuesto por todas estas personas se aprueba o no el contrato. De ser aprobado es firmado por la DIP. En caso contrario, se devuelve al Contratista con un resumen de los señalamientos. El Asesor Jurídico archiva el original de este contrato de forma general para en caso de algún problema consultarlo. No se lleva un control de los contratos lo que imposibilita saber en cada momento el estado de una obra y que es lo que va quedando por hacer en esta.

Aprobación de las facturas. Comienza cuando la Unidad Básica Contratista (UBC) le entrega a la DIP la factura, esta es revisada por el Especialista que corresponde. Si la factura es aprobada pasa a la Económica de la DIP para que elabore la Solicitud de Pago. El especialista archiva una copia de la factura para posterior consulta, además de archivarla inserta algunos datos de interés como entidad, descripción de la misma, número, fecha, importe y otros elementos en un Excel de Control. Si la factura es rechazada es devuelta por el Especialista a la UBC. Aquí encontramos un problema del negocio y es que no se lleva un seguimiento y control del estado de las facturas y el Excel de Control no brinda todos los elementos necesarios para poder llevar el control de estas.

Realizar Solicitud de Pago. Se ejecuta posterior a la aprobación de las facturas, una vez aprobada esta es enviada por el Especialista de la DIP a la Económica de la DIP. La Económica elabora la Solicitud de pago y la entrega al Dpto. de Control de la UPI dejando la copia rosada en la DIP, esta es archivada, además de llenar en el Excel de Control algunos datos para poder llevar el control mínimo de estos datos, los elementos que son almacenados son número de la Solicitud de Pago, fecha, ejecución financiera y física y contrato al que está relacionada. Es entregada al Dpto. de Control donde es aprobada, después es entregada a la SDT donde es firmada. Luego es entregada a la Subdirección General (SDG), donde también es firmada. Es devuelta a la Económica de la DIP y luego es entregada al Dpto. de Finanzas para la elaboración del cheque el original, una copia (azul) y una factura. Después de confeccionado el cheque el Dpto. de Finanzas devuelve a la Económica de la DIP la copia azul con los datos del cheque.

2.6.3 Actores del Negocio.

Un actor del negocio no es más que cualquier individuo, grupo, entidad, organización, máquina o sistema de información externo que interactúa con el negocio y desempeña un rol determinado, no representa un usuario físico, ya que varios usuarios físicos pueden realizar el mismo papel en el negocio, por otro lado, un mismo usuario puede actuar como diferentes actores. Los involucrados con el descrito son:

Tabla 2.1: Actores del Negocio.

Actores del Negocio	Justificación
UBC	Entidad que entrega la factura a la DIP y entrega contrato para ser aprobado.
DIP	Dirección que hace la solicitud de código de objeto de obra al Dpto. de Control.

2.6.4 Trabajadores del Negocio.

Un trabajador del negocio representa a personas o sistema dentro del negocio que son los que realizan las actividades que están comprendidas dentro de un caso de uso. Estos trabajadores están dentro de la

frontera del negocio, son los que en un futuro se convertirán en usuarios del sistema que se quiere construir. Los involucrados con el descrito son:

Tabla 2.2: Trabajadores del Negocio.

Trabajadores del Negocio	Justificación
Especialista	Es la persona que revisa, aprueba y archivas la factura, además confecciona con datos de las facturas un documento Excel llamado Relación de Facturas.
SDE	Responsable de asignar un código de objeto de obra y es miembro de la comisión de contratación.
Económica DIP	Persona que recibe la factura ya aprobada para que elabore la Solicitud de pago.
Dpto. de Control	Responsable de revisar y aprobar la Solicitud de código de obra, revisa los contratos para ser aprobados y aprueba la Solicitud de pago.
SDT	Responsable de firmar la solicitud de código de objeto de obra y la Solicitud de pago.
Asesor Jurídico	Recibe el contrato y junto al Comité de Contratación compuesto por la DIP, el Dpto. de Control y la Económica de la DIP lo aprueban y de ser así archiva los contratos.
DIP	Es miembro del Comité de Contratación y de ser aprobados los contratos los firma.
SDG	Responsable de firmar la Solicitud de pago después de ser aprobada.
Dpto. de Contabilidad	Recibe la Solicitud de código de objeto de obra con el código asignado y firmada.
Dpto. de Finanzas	Responsable de elaborar el cheque luego de ser aprobada la Solicitud de pago.

2.6.5 Diagrama de Casos de Uso del Negocio.

Se representan gráficamente los casos de uso del negocio identificados y su interacción con los actores a través del presente diagrama.

Diagrama de Casos de Uso del Negocio.

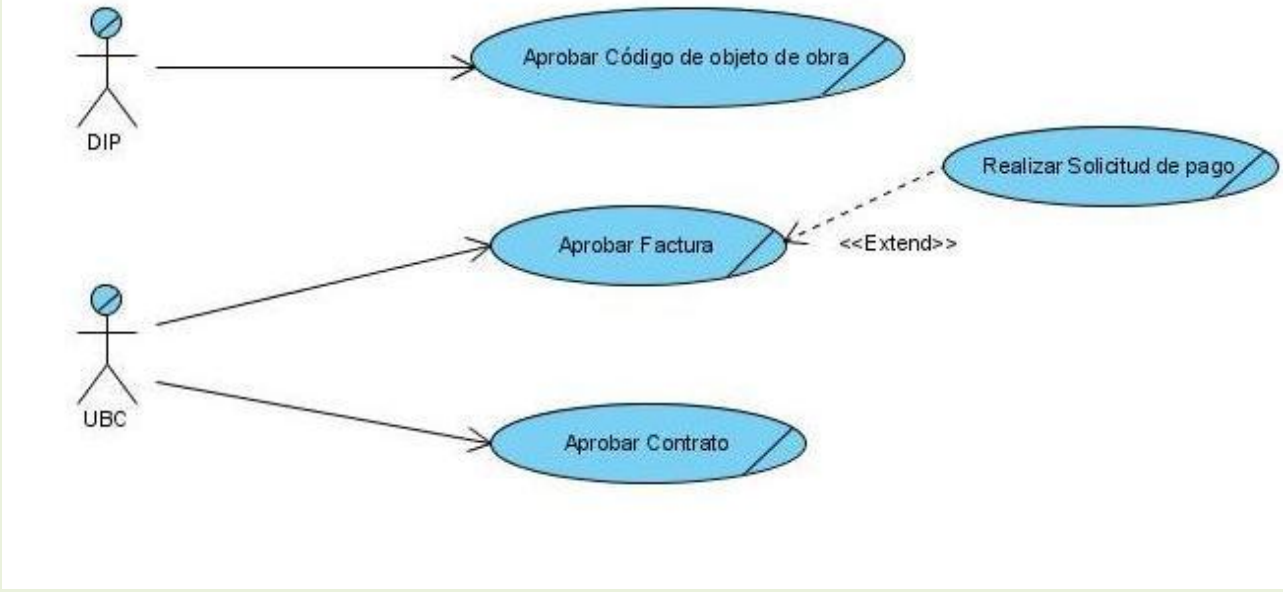


Figura 2.1: Diagrama de Casos de Uso del Negocio.

2.6.6 Realización de los casos de usos del negocio.

La realización de un caso de uso de negocio muestra cómo colaboran los trabajadores y entidades de negocio para ejecutar el proceso.

2.6.6.1 Caso de Uso “Aprobar Código de objeto de obra”.

Tabla 2.3: Descripción del Caso de Uso del Negocio “Aprobar código de objeto de obra”.

Caso de Uso:	“Aprobar Código de objeto de obra”
Actores:	DIP.
Trabajadores:	Dpto. de Control, SDT, SDE, Dpto. de Contabilidad.
Resumen:	El caso de uso inicia cuando la DIP le envía al Dpto. de Control la solicitud de código de objeto de obra firmada y acuñada, para que esta sea aprobada. La SDE le asigna el código y termina cuando la DIP le entrega la solicitud firmada para la confección del expediente de inversiones.

Diagrama de Actividades

Los casos de uso del negocio consisten en la descripción de la de secuencias de actividades que, en conjunto, producen algo para el actor del negocio. El proceso consiste en un flujo básico de una o más alternativas. La estructura del mismo se describe gráficamente con la ayuda de un diagrama de actividad.

Diagrama de Actividades – Caso de Uso “Aprobar Solicitud de Código de Objeto de Obra”. Ver **Anexos**

Anexo 1..

Diagrama de clases del modelo de objetos.

El modelo de objetos del negocio es un modelo interno de un negocio, consiste en describir cómo cada caso de uso del negocio es llevado a cabo por parte de un conjunto de trabajadores que utilizan un conjunto de entidades del negocio y unidades de trabajo.

Diagrama de Clases del Modelo de Objetos – Caso de Uso “Aprobar Solicitud de Código de Objeto de Obra”. Ver **Anexo 2.**.

2.6.6.2 Caso de Uso “Aprobar Factura”.

Tabla 2.4: Descripción del Caso de Uso del Negocio “Aprobar Factura”.

Caso de Uso:	“Aprobar Factura”
Actores:	UBC.
Trabajadores:	Especialista, Económica DIP.
Resumen:	El caso de uso inicia cuando la UBC le entrega a la DIP la factura. El especialista correspondiente la revisa y la aprueba y termina cuando es enviada a la Económica para que elabore la Solicitud de pago

Diagrama de Actividades – Caso de Uso “Aprobar Factura”. Ver **Anexo 3..**

Diagrama de Clases del Modelo de Objetos – Caso de Uso “Aprobar Factura”. Ver **Anexo 4..**

2.6.6.3 Caso de Uso “Aprobar Contrato”.

Tabla 2.5: Descripción del Caso de Uso del Negocio “Aprobar Contrato”.

Caso de Uso:	“Aprobar Contrato”
Actores:	UBC.
Trabajadores:	Asesor Jurídico, DIP, Dpto. de Control, SDE.
Resumen:	El caso de uso inicia cuando la UBC le entrega al Asesor Jurídico el contrato, este es revisado por un comité de contratación y de ser aprobado es firmado por la DIP.
Casos de uso asociados:	“Realizar Solicitud de pago” <<extend>>

Diagrama de Actividades – Caso de Uso “Aprobar Contrato”. Ver **Anexo 5..**

Diagrama de Clases del Modelo de Objetos – Caso de Uso “Aprobar Contrato”. Ver **Anexo 6..**

2.6.6.4 Caso de Uso “Realizar Solicitud de pago”.

Tabla 2.6: Descripción del Caso de Uso del Negocio “Realizar Solicitud de pago”.

Caso de Uso:	“Realizar Solicitud de pago”
Actores:	Especialista.
Trabajadores:	Económica DIP, Dpto. de Control, SDT, SDG, Dpto. de Finanzas.
Resumen:	El caso de uso inicia cuando el especialista le entrega la factura ya revisada a la Económica DIP para que realice la Solicitud de pago y sea confeccionado el/los cheque/s correspondiente/s.

Diagrama de Actividades – Caso de Uso “Aprobar Contrato”. Ver **Anexo .**

Diagrama de Clases del Modelo de Objetos – Caso de Uso “Aprobar Contrato”. Ver **Anexo 8.**

2.7 Especificación de los requisitos de software.

Fase de especificación de requisitos es en la cual debe conseguirse el catálogo de requisitos del sistema que englobe: la definición de los objetivos del sistema, los requisitos de almacenamiento de información, la descripción de los actores del sistema, los requisitos funcionales, descritos a través de los casos de uso, los requisitos de interacción, en lo que se recogerá el sistema de navegación de la aplicación, la interacción con el usuario y los requisitos no funcionales.

El proceso de definir y especificar los requerimientos del software consiste en establecer los servicios que un producto de software debe cumplir; proceso que lleva a la construcción de un documento conocido como Especificación de Requerimientos del Software, esencial para lograr un producto de alta calidad.

Obtener los Requerimientos del Software es paso esencial para la construcción de un producto de software para el posterior desarrollo de las etapas. Un error en este proceso, traería graves consecuencias en la obtención de un producto, pues podría no cumplir las expectativas del cliente.

2.7.1 Requisitos Funcionales.

Los Requerimientos Funcionales son capacidades o condiciones con las que debe cumplir el producto a elaborar y están fuertemente ligados a las opciones del programa, no alteran la funcionalidad del software, se mantienen invariables sin importar con que cualidades o propiedades se relacionen.

A partir del estudio y las investigaciones que se realizaron de los procesos del negocio, se obtuvieron una serie de requerimientos funcionales que ha cumplir el software:

R1. Gestionar Usuario.

El sistema debe permitirle al Dpto. de Control gestionar sus usuarios.

R1.1 Registrar Usuario.

El sistema debe permitirle al Dpto. de Control introducir la información del usuario nuevo:

1. Nombre.
2. Apellidos.
3. Usuario.
4. Contraseña.
5. Cargo.
6. DIP.

Al registrar un usuario su estado será activo, pero esto no debe ser especificado por el Dpto. de Control.

R1.2 Modificar Usuario.

El sistema debe permitirle al Dpto. de Control modificar la información de sus usuarios:

1. Contraseña.
2. Nombre.
3. Apellidos.

4. Cargo.
5. DIP.
6. Estado.

R2. Autenticar Usuario:

El sistema debe permitir autenticar a los usuarios, para que estos tengan acceso a las funcionalidades asignadas a este usuario.

R3. Activar/Desactivar Usuario:

El sistema debe permitirle al Dpto. de Control activar o desactivar los usuarios que están registrados en el sistema, los usuarios desactivados no podrán acceder al sistema, solo los que estén activados lo podrán hacer, por lo que, el Dpto. de Control es el único que permite el acceso al sistema:

R4. Listar Usuarios:

El sistema debe permitirle al Dpto. de Control verificar la información de los usuarios y listarlos:

1. Nombre.
2. Apellidos.
3. Usuario.
4. Cargo.
5. DIP.
6. Estado.

R5. Gestionar Objeto de Obra.

El sistema debe permitirle al Dpto. de Control gestionar la información requerida de los objetos de obra siendo lo mismo que los centros de costo.

R5.1 Registrar Objeto de Obra:

El sistema debe permitirle al Dpto. de Control registrar los datos de los objetos de obras, asignarlo a la DIP que le corresponda, así como inicializar dicho objeto con los presupuestos correspondientes aprobados en el Plan.

1. Código de objeto de obra.
2. Descripción.

3. Plan por componentes.
4. Nombre.
5. Plan por monedas.
6. Estado del código de objeto de obra.
7. Observaciones.

R5.2 Modificar Objeto de Obra:

El sistema debe permitirle al Dpto. de Control actualizar los datos de los objetos de obras.

1. Descripción.
2. Plan por componentes.
3. Nombre.
4. Plan por monedas.
5. Estado del código de objeto de obra.
6. Observaciones.

R6. Listar Objetos de Obra.

El sistema permite listar todos los objetos de obra que están registrados en el mismo.

R7. Buscar Objetos de Obra.

El sistema debe permitirle al Dpto. de Control realizar una búsqueda antes de actualizar un objeto de obra o centro de costo, el criterio que tendrá esta búsqueda es por el código del objeto de obra o centro de costo.

R8. Generar Reporte de Conciliación.

El sistema debe permitirle al Especialista generar un reporte que será usado para la conciliación con los datos que brinda el sistema contable ASSET.

R9. Generar Reporte Ejecuciones.

El sistema debe brindarle al Director un resumen de las ejecuciones por obras e importes, este puede ver el resumen de todas las obras registradas en el sistema.

R9.1 Por fecha.

El sistema debe brindarle al Director un resumen de las ejecuciones por obras e importes, en el intervalo de fechas que este desee.

R9.2 Acumulado.

El sistema debe brindarle al Director un resumen de las ejecuciones por obras e importes en acumulado.

R10. Generar Reporte Catálogo de Centros de Costos y Áreas de Responsabilidad.

El sistema debe brindarle al Director un reporte donde se muestre la descripción del centro de costo así como la descripción del grupo que desarrolla el mismo.

R11. Gestionar Factura.

El sistema debe brindar la posibilidad a los Especialistas de gestionar las facturas generadas durante el proceso inversionista.

R11.1 Registrar Factura.

El sistema debe brindar la posibilidad a los Especialistas de ingresar una nueva factura en dependencia del código de objeto de obra que pertenezca, debe ingresar los siguientes datos:

1. Código de objeto de obra a la que corresponde.
2. Código de la Factura.
3. Número del contrato. (Puede ser que cuando se registre una factura no tenga definido el contrato asociado)
4. Proveedor.
5. Importe en monedas (MN, CUC).
6. Identificar los componentes que afecta (Construcción y Montaje, Equipos, Otros).
7. Descripción.
8. Descuento (MN, CUC).
9. Tipo de descuento.
10. Código de Entidad.
11. Tipo de factura.
12. Estado.
13. Fecha de recepción.

14. Observaciones.
15. Fecha de devolución. (En caso que no sea aprobada, se le pone la fecha de devolución)
16. Fecha de aprobación. (De ser aprobada es que se le pone la fecha de aprobación)

R11.2 Actualizar factura.

El sistema debe brindar la posibilidad a los Especialistas de modificar los datos de las facturas.

1. Número del contrato. (Se muestra solo si está por definir aun el contrato que la respalda)
2. Importe en monedas (MN, CUC).
3. Identificar los componentes afecta (Construcción y Montaje, Equipos, Otros).
4. Descripción.
5. Descuento (MN, CUC).
6. Tipo de descuento.
7. Estado.

R12. Listar Facturas por definir contrato.

El sistema debe brindar la posibilidad al Asesor Jurídico de archivar las facturas que no presenten contrato, es decir, que esté por definir, debe saberse a que obra pertenece la misma y debe dar la opción de definir el contrato.

R13. Buscar Facturas.

El sistema le permite al Especialista buscar una factura a la cual se le modificará algún dato.

R14. Listar Facturas.

Permite mostrarle al Especialista todas las facturas que hay en el sistema.

R14.1 Listar Facturas por objeto de obra.

El sistema debe brindar la posibilidad al Especialista de listar todas las facturas referentes a una obra específica seleccionada por este.

R15. Definir estado de factura.

El sistema debe brindar la posibilidad a los Especialistas de especificar el estado de la factura, puede ser: aprobada, devuelta, rechazada o en revisión. En dependencia de su estado es la manera que ingresa al

sistema. Si la factura ha sido aprobada pasa a la facturación real de la obra, en otro caso debe archivarse y no figura dentro del sistema contable, pero sí, dentro de las producciones reales del mes.

R16. Gestionar Solicitud de pago.

El sistema debe brindarle la posibilidad al Especialista de realizar una Solicitud de pago y actualizar la información de estas.

R16.1 Realizar Solicitud de pago.

Una vez ingresada la factura en el sistema este debe brindar a los Especialistas la posibilidad de confeccionar la Solicitud de pago, la misma puede ser para varias facturas. Como precondición las facturas deben estar registradas en el sistema y su estado tiene que ser aprobado. La solicitud debe de incluir los siguientes aspectos:

1. Código de la/s factura/s correspondiente/s.
2. Número de la Solicitud de pago.
3. IR (número de informe de recepción).
4. Fecha de solicitud.
5. Importe (MN-CUC).
6. Descripción.
7. Especificar beneficiario para ambas monedas.
8. Proveedores.
9. Observaciones.
10. Estado.

R16.2 Actualizar Solicitud de pago.

El sistema debe brindar la posibilidad de que si el estado de la factura es en proceso esta pueda ser actualizada, ya cuando su estado sea aprobada, esta solicitud no puede ser modificada.

1. Fecha de solicitud.
2. Importe (MN-CUC).
3. Descripción.
4. Especificar beneficiario para ambas monedas.

5. Proveedores.
6. Observaciones.
7. Estado.

R17. Alertar Solicitud de pago sin contrato.

El sistema debe alertarle al Especialista cuando se esté realizando una Solicitud de pago para una factura que no presente contrato que la respalde.

R18. Buscar Solicitud de pago.

El sistema debe permitirle al Especialista buscar una solicitud de pago que ya esté registrada en el sistema.

R19. Asignar Cheques.

Una vez aprobada la Solicitud de pago el sistema debe brindarle la posibilidad al Especialista de asignar un cheque a una solicitud de pago y serían los siguientes datos:

1. Número de cheque MN.
2. Número de cheque CUC.
3. Importe MN-CUC.
4. Fecha aprobada la solicitud.
5. Observaciones.

R20. Generar Reporte de producción.

Cuando la factura llegue a la DIP correspondiente, aunque no haya sido aprobada, significa que representa como tal una producción; por lo tanto debe figurar, para los efectos, como una producción real dentro de esa obra (**R14.1**) tanto como las facturas aprobadas. El sistema debe generar un reporte donde muestre la producción en un período de tiempo definido por el Director.

R20.1 Por objeto de obra.

Podrá seleccionar la obra de la cual desea obtener el reporte.

R20.2 Por DIP.

Si el Director desea obtener el reporte de toda una DIP podrá seleccionar esa opción y brindarle al sistema la DIP de la que desea obtener el reporte.

R20.3 Todas las obras.

Obtendrá un reporte de producción de todos los objetos de obras registrados en el sistema.

R21. Gestionar Contrato.

El sistema debe permitirle al Asesor Jurídico registrar los contratos realizados y actualizar los mismos.

R21.1 Registrar Contrato.

El sistema debe permitirle al Asesor Jurídico registrar los contratos realizados de los cuales es necesario almacenar los siguientes datos.

1. ID del contrato.
2. Código de objeto de obra asociado.
3. Código de Entidad.
4. Fecha de entrada.
5. Importe MN-CUC.
6. Estado.

Si este contrato es devuelto por tener errores, se necesita también.

7. Fecha devuelto.

Cuando este sea aprobado recoger.

8. Fecha aprobado.

R21.2 Actualizar Contrato.

El sistema debe permitirle al Asesor Jurídico actualizar los contratos realizados mientras estos no hayan sido aprobados, cuando estos sean aprobados ya no podrán ser cambiados ninguno de sus datos.

1. Fecha de entrada.
2. Importe MN-CUC.
3. Estado.

4. Fecha devuelto.
5. Fecha aprobado.

R22. Buscar por Organismo.

El sistema le permite al Asesor Jurídico seleccionando un organismo de los que está registrado y mediante este se listan todas las entidades con que cuenta este organismo, permitiéndole seleccionar la entidad a la que se le contraerá contrato.

R23. Buscar por DIP.

El sistema le permite al Asesor Jurídico seleccionando una DIP le listará todos los objetos de obra que esta tiene y escogerá de esta lista a que obra le asociará el contrato.

R24. Listar Contratos.

El sistema debe permitirle al Asesor Jurídico listar todos los contratos que tiene registrados en el sistema.

R25. Gestionar Suplemento.

Permite al Asesor Jurídico tanto registrar un nuevo suplemento a un contrato como modificar uno ya existente, esto solo lo puede hacer mientras este suplemento no haya sido aprobado.

R25.1 Registrar Suplemento.

El sistema debe permitirle al Asesor Jurídico registrar los suplementos de los contratos que estos tendrán los siguientes datos.

1. ID suplemento.
2. ID contrato.
3. Importe MN-CUC.
4. Fecha del suplemento.
5. Fecha de aprobación.
6. Fecha de devolución.
7. Estado.

8. Observaciones.

R25.2 Actualizar Suplemento.

El sistema debe permitirle al Asesor Jurídico actualizar los datos de los suplementos de los contratos mientras estos no hayan sido aprobados.

1. ID suplemento.
2. ID contrato.
3. Importe MN-CUC.
4. Fecha del suplemento.
5. Fecha de aprobación.
6. Fecha de devolución.
7. Estado.
8. Observaciones.

R26. Buscar Contrato.

El sistema debe permitirle al Asesor Jurídico buscar un contrato al cual asignarle un suplemento, así también cuando desea modificar este contrato.

R26.1 Buscar Contrato por código de contrato.

El sistema debe permitirle al Asesor Jurídico buscar un contrato mediante el código de este.

R26.2 Buscar Contrato por Entidad.

El sistema debe permitirle al Asesor Jurídico buscar un contrato mediante la entidad con la cual se realizó el contrato.

R26.3 Buscar Contrato por código de objeto de obra.

El sistema debe permitirle al Asesor Jurídico buscar un contrato mediante el código de objeto de obra a la cual se le asignó el contrato.

R26.4 Buscar Contrato por código de objeto de obra y Entidad.

El sistema debe permitirle al Asesor Jurídico buscar un contrato mediante dos criterios de búsqueda, código de objeto de obra y Entidad.

R27. Gestionar Entidad de Contratación.

El sistema debe permitirle al Asesor Jurídico registrar las entidades de contratación así como actualizarlas.

R27.1 Registrar Entidad de Contratación.

El sistema debe permitirle al Asesor Jurídico registrar las entidades de contratación que contraen acuerdos con la Inversión.

1. Código de la Entidad.
2. Código del Organismo.
3. Nombre de la Entidad.

R27.2 Actualizar Entidad de Contratación.

El sistema debe permitirle al Asesor Jurídico actualizar los datos de las entidades de contratación que contraen acuerdos con la Inversión.

1. Código de la Entidad.
2. Código del Organismo.
3. Nombre de la Entidad.

R28. Buscar Organismo.

Para registrar una nueva Entidad de Contratación es necesario saber a qué Organismo esta será adicionada, por lo que se realiza una búsqueda por el código del Organismo, por el nombre o por ambos, en dependencia de la elección del Asesor Jurídico y posteriormente ya podrá registrar la nueva Entidad.

R29. Gestionar Organismo.

El sistema debe permitirle al Asesor Jurídico registrar los organismos así como actualizarlas.

R29.1 Registrar Organismo.

El sistema debe permitirle al Asesor Jurídico registrar organismos.

1. Código del Organismo.
2. Nombre del Organismo.

R29.2 Actualizar Organismo.

El sistema debe permitirle al Asesor Jurídico actualizar los datos de los organismos.

1. Código del Organismo.
2. Nombre del Organismo.

R30. Listar Organismo.

Permite al Asesor Jurídico listar todos los organismos de contratación que están registrados en el sistema.

2.7.2 Requisitos no Funcionales.

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido y confiable. A continuación se muestran los requisitos no funcionales.

Apariencia o interfaz externa:

- Diseño sencillo, con pocas entradas, permitiendo que no sea necesario mucho entrenamiento para utilizar el sistema.
- El producto debe ser legible y con colores adecuados, agradables y poco llamativos.
- Diseño perfectamente encuadrado para resoluciones de 1024x768, pero preparado para verse en otras resoluciones.

Usabilidad:

- El sistema podrá ser usado por cualquier persona que posea conocimientos básicos en el manejo de la computadora y de aplicaciones de escritorio en sentido general.

Rendimiento:

- Tiempos de respuestas rápidos, al igual que la velocidad de procesamiento de la información.

Soporte:

- Se requiere un servidor de bases de datos con las siguientes características:
  - ✓ Soporte para grandes volúmenes de datos y velocidad de procesamiento.
  - ✓ Tiempo de respuesta rápido en accesos concurrentes.
  - ✓ Tecnología libre.
- Por parte del cliente se requiere soporte Java instalado, Versión 6.0 o superior.

Portabilidad:

- Necesidad de que el sistema sea multiplataforma.

Seguridad:

- Identificar al usuario antes de que pueda realizar cualquier acción sobre el sistema.
- Garantizar que la información sea vista únicamente por quien tiene derecho a verla.
- Garantizar que las funcionalidades del sistema se muestren de acuerdo al nivel de usuario que esté activo.
- Protección contra acciones no autorizadas o que puedan afectar la integridad de los datos.
- Verificación sobre acciones irreversibles (eliminación de datos).

Legales:

- El empleo de este producto no debe violar ninguna ley o licencia por lo que la plataforma escogida para el desarrollo de la aplicación, está basada en la licencia GNU/GPL.
- Este documento y la aplicación pertenecen a la UPI “Proyecto Futuro”.

Confiabilidad:

- La herramienta de implementación a utilizar tiene soporte para recuperación ante fallos y errores.

Funcionalidad:

- Mínima cantidad de vistas para ejecutar todas las funciones posibles (preferentemente que estén relacionadas).
- Operaciones realizadas mediante el teclado.

Software:

- Java.
- PostgreSQL.

2.8 Definición de los casos de uso.

2.8.1 Actores del Sistema.

Los actores de sistema pueden representar el rol que juega una o varias personas, un equipo o un sistema automatizado y pueden intercambiar información con él o ser recipientes pasivos de información. Los actores suelen corresponderse con los trabajadores o actores del negocio. Teniendo en cuenta todos los requerimientos planteados, se definen varios roles para diferenciar el nivel de acceso al sistema: Dpto. de Control, Director, Especialista y Asesor Jurídico.

Tabla 2.7: Actores del Sistema.

Actores del Sistema	Justificación
Dpto. de Control	Es el administrador del sistema y accede a todas las funcionalidades del mismo. Gestiona los usuarios y los objetos de obra, además puede obtener en cualquier momento una lista de todos los usuarios que hay registrados en el sistema.
Especialista	Es el encargado de gestionar las facturas y las solicitudes pago, asigna los cheques luego de aprobar las solicitudes de pago, tiene acceso al reporte de conciliación, además de saber todas las facturas que hay en un objeto de obra.
Director	Persona que accede al sistema para obtener de él reportes como el de ejecuciones y el de producción.
Asesor Jurídico	Persona encargada de gestionar los contratos, registra las entidades de contratación y los organismos nuevos.
Usuario	Rol general que representa a todos los anteriores y que se

	autentica.

2.8.3 Listado de casos de uso.

Tabla 2.8: Breve descripción del caso de uso del sistema "Gestionar Usuario".

CU_1	"Gestionar Usuario"
Actor	Dpto. de Control.
Descripción	Permite al Dpto. de Control registrar un nuevo usuario y actualizar alguno ya registrado.
Referencia	R1, R1.1, R1.2, R3.

Tabla 2.9: Breve descripción del caso de uso del sistema "Autenticar Usuario".

CU_2	"Autenticar Usuario"
Actor	Usuario (Especialista, Director, Asesor Jurídico, Dpto. de Control).
Descripción	Permite a los usuarios ya registrados en el sistema por el Dpto. de Control autenticarse en la aplicación con el rol definido, permitiendo que solo tengan acceso a los recursos que le son permitidos según su rol.
Referencia	R2.

Tabla 2.10: Breve descripción del caso de uso del sistema "Listar Usuarios".

CU_3	"Listar Usuarios"
Actor	Dpto. de Control.
Descripción	Permitirá visualizar todos los usuarios registrados en el sistema, mostrando los datos de estos. Permite que este listado sea guardado en formato Excel o PDF para una dirección especificada, además de poder imprimirlo en uno de estos dos formatos.
Referencia	R4.

Tabla 2.11: Breve descripción del caso de uso del sistema "Gestionar Objeto de Obra".

CU_4	"Gestionar Objeto de Obra"
Actor	Dpto. de Control.
Descripción	Permite gestionar la información de los objetos de obras, tales como, registrar un objeto de obra y actualizar los datos introducidos por en Dpto. de Control.
Referencia	R5, R5.1, R5.2, R7, "Buscar Objeto de Obra" <<extend>>.

Tabla 2.12: Breve descripción del caso de uso del sistema "Listar Objetos de Obra".

CU_5	"Listar Objetos de Obra"
Actor	Dpto. de Control.
Descripción	Permite al Dpto. de Control acceder al listado de todas las obras que tiene.
Referencia	R6.

Tabla 2.13: Breve descripción del caso de uso del sistema "Generar Reporte de Conciliación".

CU_6	"Generar Reporte de Conciliación"
Actor	Especialista, Dpto. de Control.
Descripción	Permite generar un reporte que será usado para la conciliación con los datos que brinda el sistema contable ASSET. Este reporte contiene datos como: la fecha en que se realiza el cierre contable, las facturas, con la entidad que la hace y la fecha en que fue aprobada la misma, el total a pagar en ambas monedas (MN, CUC), además de un desglose de esto por componentes.
Referencia	R8.

Tabla 2.14: Breve descripción del caso de uso del sistema "Generar Reporte de Ejecuciones".

CU_7	"Generar Reporte de Ejecuciones"
Actor	Director, Dpto. de Control.
Descripción	Permite generar un reporte con un resumen de las ejecuciones de las obras, brindando el importe de estas, se le brindará al Director la posibilidad de seleccionar el intervalo de fechas que desea conocer o podrá seleccionar la opción de ver el acumulado que mostraría todas las obras que están el sistema. Este reporte muestra las obras más el plan de esta y lo que esta facturado aprobado.
Referencia	R6, R6.1, R6.2.

Tabla 2.15: Breve descripción del caso de uso del sistema "Generar Reporte Catálogo de Centros de Costos y Áreas de Responsabilidad".

CU_8	"Generar Reporte Catálogo de Centros de Costos y Áreas de Responsabilidad"
Actor	Director, Dpto. de Control.
Descripción	El sistema le brinda al Director la posibilidad de generar un reporte donde le muestre la descripción del centro de costo que puede definirse como objeto de obra, así como la descripción del grupo que desarrolla el mismo.
Referencia	R10.

Tabla 2.16: Breve descripción del caso de uso del sistema "Gestionar Factura".

CU_9	"Gestionar Factura"
Actor	Especialista, Dpto. de Control.
Descripción	Permite al Especialista gestionar las facturas generadas durante el proceso inversionista; podrá ingresar al sistema una nueva factura en dependencia al código de objeto de obra que pertenezca, así mismo, podrá actualizar los datos ya insertados.
Referencia	R11, R11.1, R11.2, R13, "Buscar Factura" <<extend>>.

Tabla 2.17: Breve descripción del caso de uso del sistema "Listar Facturas por definir contrato".

CU_10	"Listar Facturas por definir contrato"
Actor	Asesor Jurídico, Dpto. de Control.
Descripción	Permite al Asesor Jurídico conocer las facturas que están en el sistema que aún no tienen un contrato asociado o sea que la respalde, le muestra a que obra pertenece esta factura y le de

	la posibilidad de definir el contrato al que estará asociada.
Referencia	R12.

Tabla 2.18: Breve descripción del caso de uso del sistema "Listar Facturas".

CU_11	**"Listar Facturas"**
Actor	Especialista, Dpto. de Control.
Descripción	Permite al Especialista obtener un listado de todas las facturas registradas en el sistema.
Referencia	R14, R14.1, "Listar Facturas por objeto de obra" <<>extend>.

Tabla 2.19: Breve descripción del caso de uso del sistema "Gestionar Solicitud de pago".

CU_12	**"Gestionar Solicitud de pago"**
Actor	Especialista, Dpto. de Control.
Descripción	Permite gestionar la información de las solicitudes de pago, brinda la posibilidad de que luego que sea realizada una solicitud de pago se pueda actualizar la información referente a esta.
Referencia	R16, R16.1, R16.2, R18, "Buscar Solicitud de Pago" <<extend>>.

Tabla 2.20: Breve descripción del caso de uso del sistema "Alertar Solicitud de pago sin contrato".

CU_13	"Alertar Solicitud de pago sin contrato"
Actor	Especialista, Dpto. de Control.
Descripción	El sistema muestra una alerta al Especialista y al Dpto. de Control cuando se esté realizando una solicitud de pago para una factura que no presente contrato que la respalde.
Referencia	R16, R16.1, R16.2, R17.

Tabla 2.21: Breve descripción del caso de uso del sistema "Asignar Cheque".

CU_14	"Asignar Cheque"
Actor	Especialista, Dpto. de Control.
Descripción	Permite asignarle un cheque a una solicitud de pago ya aprobada.
Referencia	R18, R19, "Buscar Solicitud de pago" <<include>>.

Tabla 2.22: Breve descripción del caso de uso del sistema "Generar Reporte de Producción".

CU_15	"Generar Reporte de Producción"
Actor	Director, Dpto. de Control.
Descripción	Permite obtener un reporte de la producción real dentro de una obra, esta producción real está dada tanto el importe de una factura aprobada como la que no haya sido aprobada, ya que ambas representan una producción, este reporte se muestra en dependencia del intervalo de fechas seleccionadas por el Director o el Dpto. de Control.
Referencia	R14, R20, R20.1, R20.2, R20.3.

Tabla 2.23: Breve descripción del caso de uso del sistema "Gestionar Contrato".

CU_16	"Gestionar Contrato"
Actor	Asesor Jurídico, Dpto. de Control.
Descripción	Permite gestionar la información de los contratos, el Asesor Jurídico tanto como el Dpto. de Control pueden registrar un nuevo contrato, así como, actualizar los datos de uno ya registrado.
Referencia	R16, R16.1, R16.2, "Buscar por Organismo" <<extend>>, "Buscar por DIP" <<extend>>, "Buscar Contrato" <<extend>>, "Listar Contratos" <<extend>>.

Tabla 2.24: Breve descripción del caso de uso del sistema "Gestionar Suplemento".

CU_17	"Gestionar Suplemento"
Actor	Asesor Jurídico, Dpto. de Control.
Descripción	Permite registrar un suplemento a un contrato, y modificarlos mientras este no esté aprobado.
Referencia	R25, R25.1, R25.2 R26, R26.1, R26.2, R26.3, R26.4.

Tabla 2.25: Breve descripción del caso de uso del sistema "Gestionar Entidad de Contratación".

CU_18	"Gestionar Entidad de Contratación"
Actor	Asesor Jurídico, Dpto. de Control.
Descripción	Permite registrar una nueva Entidad de Contratación y modificar los datos de una ya existente.
Referencia	R27, R27.1, R27.2, R28, "Buscar Organismo"<<extend>>, "Listar Organismos"<<extend>>.

Tabla 2.26: Breve descripción del caso de uso del sistema "Gestionar Organismo".

CU_19	"Gestionar Organismo"
Actor	Asesor Jurídico, Dpto. de Control.
Descripción	Permite registrar un nuevo organismo y actualizar uno ya registrado.
Referencia	R29, R29.1, R29.2.

Tabla 2.27: Breve descripción del caso de uso del sistema "Listar Contratos".

CU_20	"Listar Contratos"
Actor	Asesor Jurídico, Dpto. de Control.
Descripción	Permite al Asesor Jurídico listar todos los contratos que tiene registrados.
Referencia	R24, Extendido de "Gestionar Contratos", extendido de "Gestionar Suplemento".

Tabla 2.28: Breve descripción del caso de uso del sistema "Listar Facturas por objeto de obra".

CU_21	"Listar Facturas por objeto de obra"
Actor	Especialista, Dpto. de Control.
Descripción	Permite listar todas las facturas dada una obra.
Referencia	R14, R14.1, Extendido de "Listar Facturas".

Tabla 2.29: Breve descripción del caso de uso del sistema "Buscar Factura".

CU_22	"Buscar Factura"

Actor	Especialista, Dpto. de Control.
Descripción	Permite al Especialista buscar una factura a la cual le modificara alguno de sus datos.
Referencia	R13, Extendido de “Gestionar Facturas”.

Tabla 2.30: Breve descripción del caso de uso del sistema “Buscar Solicitud de pago”.

CU_23	**“Buscar Solicitud de pago”**
Actor	Especialista, Dpto. de Control.
Descripción	Permite al Especialista buscar una solicitud de pago a la cual se le asignará un cheque, así como le dará la posibilidad de buscarla para actualizar sus datos.
Referencia	R18, Extendido de “Gestionar Solicitud de pago”, incluido de “Asignar Cheque”.

Tabla 2.31: Breve descripción del caso de uso del sistema “Buscar Objeto de Obra”.

CU_24	**“Buscar Objeto de Obra”**
Actor	Dpto. de Control.
Descripción	Permite al Dpto. de Control buscar objeto de obra al cual se le hará alguna modificación.
Referencia	R7, Extendido de “Gestionar Objeto de Obra”.

Tabla 2.32: Breve descripción del caso de uso del sistema “Buscar por Organismo”.

CU_25	**“Buscar por Organismo”**
Actor	Asesor Jurídico, Dpto. de Control.

Descripción	Permite al Asesor Jurídico realizar una búsqueda mediante el organismo con el que se contrajo contrato para buscar el mismo y realizarle alguna actualización.
Referencia	R21, R21.1, R21.2, R22, Extendido de “Gestionar Contrato”.

Tabla 2.33: Breve descripción del caso de uso del sistema “Buscar por DIP”.

CU_26	**“Buscar por DIP”**
Actor	Asesor Jurídico, Dpto. de Control.
Descripción	Permite al Asesor Jurídico realizar una búsqueda mediante el la DIP a la que pertenece el contrato para buscar el mismo y realizarle alguna actualización.
Referencia	R21, R21.1, R21.2, R23, Extendido de “Gestionar Contrato”.

Tabla 2.34: Breve descripción del caso de uso del sistema “Buscar Contrato”.

CU_27	**“Buscar Contrato”**
Actor	Asesor Jurídico, Dpto. de Control.
Descripción	Permite al Asesor Jurídico buscar un contrato al cual asignarle un suplemento.
Referencia	R21, R21.1, R21.2, R25, R26, Extendido de “Gestionar Suplemento”, extendido de “Gestionar Contratos”.

Tabla 2.35: Breve descripción del caso de uso del sistema “Listar Organismos”.

CU_28	**“Listar Organismos”**
Actor	Asesor Jurídico, Dpto. de Control.

Descripción	Permite al Asesor Jurídico listar todos los organismos que están registrados en el sistema.
Referencia	R27, R27.1, R27.2, R30, Extendido de “Gestionar Entidad de Contratación”.

Tabla 2.36: Breve descripción del caso de uso del sistema “Buscar Organismo Organismos”.

CU_29	**“Buscar Organismo”**
Actor	Asesor Jurídico, Dpto. de Control.
Descripción	Permite al Asesor Jurídico buscar un organismo de los que está registrado en el sistema para insertarle una entidad de contratación.
Referencia	R27, R27.1, R27.2, R28, Extendido de “Gestionar Entidad de Contratación”.

2.8.4 Diagrama de Casos de Uso del Sistema.

El Diagrama de Casos de Uso del Sistema es un artefacto de Ingeniería de Software que describe, bajo la forma de acciones y reacciones, el comportamiento del sistema desde el punto de vista del usuario, permitiendo de esta forma el establecimiento de un acuerdo entre clientes y desarrolladores sobre las condiciones y requerimientos que debe cumplir el sistema. Este modelo está formado por actores, casos de uso y las relaciones que se establecen entre estos, es decir, representa gráficamente a los procesos y su interacción con los actores y constituye una entrada de gran valor para las siguientes fases de construcción de un software.

Diagrama de Casos de Uso del Sistema.

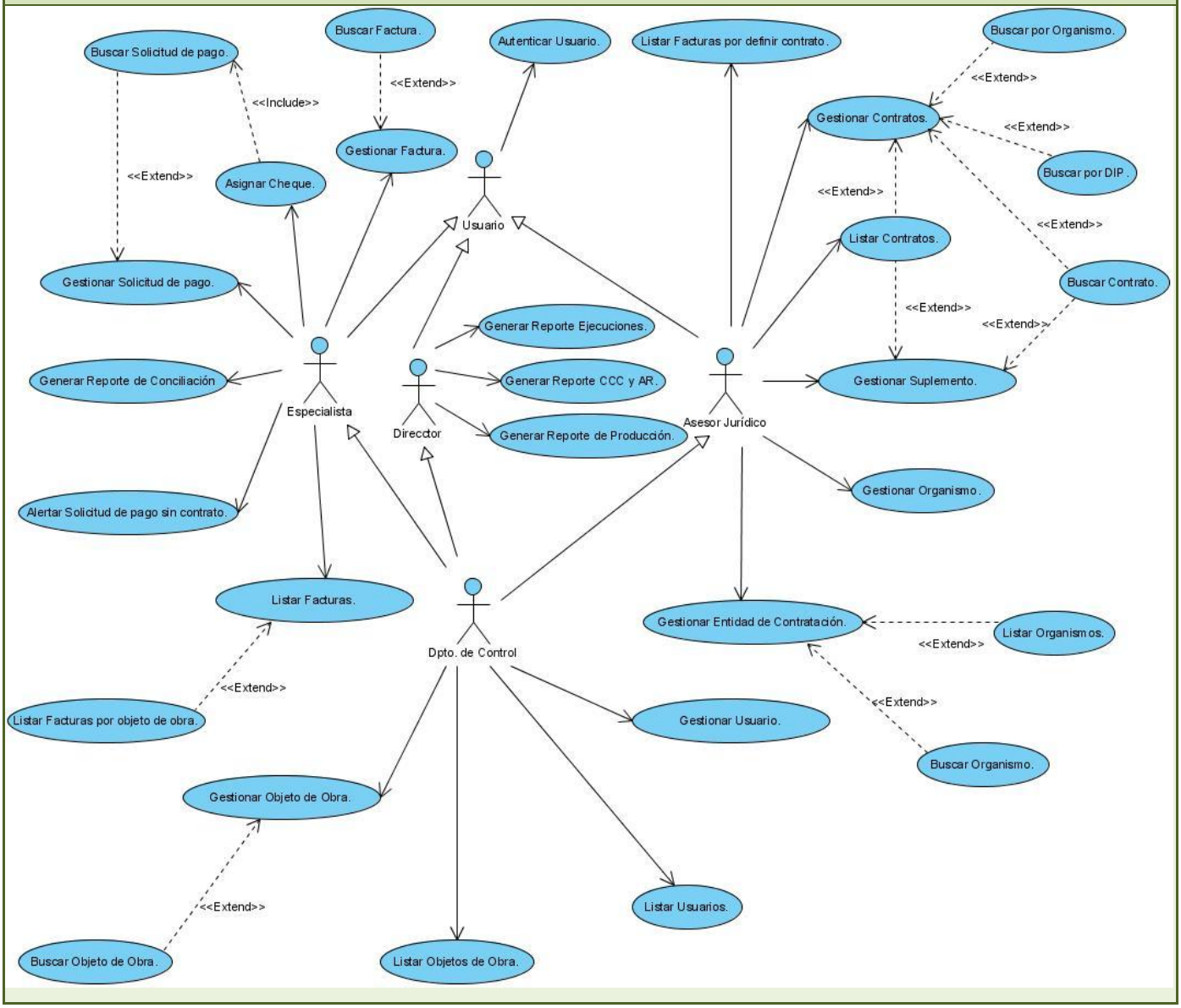


Figura 2.2: Diagrama de Casos de Uso del Sistema.

2.8.5 Patrones de casos de uso.

Tras el proceso de análisis de los requisitos y la definición de los casos de uso que se tendrán en el sistema, que responden y abarcan esos requisitos, se representan los actores y casos de uso mediante el Diagrama de Casos de Uso del Sistema (**Figura 2.2)**. En este diagrama se utilizaron varios patrones de casos de uso, estos son:

CRUD parcial: Ya que se representa en los casos de uso de Gestionar Factura, Gestionar Usuarios, Gestionar Solicitud de pago y Gestionar Contratos, procesos como crear y modificar.

Múltiples actores, específicamente el patrón **Rol común**: Pues se destacan actividades comunes para varios actores, por lo que se define un actor del cual heredan los actores con estas funcionalidades comunes.

2.8.6 Casos de Usos a desarrollar por ciclo.

Al culminar un ciclo de un proyecto de software, se obtiene una versión del producto. Por eso se debe determinar con anterioridad los casos de usos que se van a desarrollar en cada ciclo de vida del proyecto para poder tener una guía por la cual trabajar, así como trazar las metas.

Tabla 2.40: Distribución de Casos de Uso por ciclos de desarrollo.

Ciclo	Código	Nombre del caso de uso	Justificación de la selección.
1	CU_1	Gestionar Usuario.	Es importante que el sistema lleve un estricto control de los usuarios, ya que de estos dependerán los restantes datos. Además de los
	CU_2	Autenticar Usuario.	
	CU_3	Listar Usuarios.	

	CU_4	Gestionar Objeto de Obra.	objetos de obras que a partir de aquí es donde se comienza a insertar todos los datos, de no haber obras no se puede registrar ningún otro dato.
	CU_5	Listar Objetos de Obra.	
	CU_24	Buscar Objeto de Obra.	
2	CU_16	Gestionar Contratos.	Estos casos de usos se implementarían en un segundo ciclo dado que son necesarios para poder registrar las facturas y hacer las solicitudes de pago. Luego de tener todos estos casos de uso implementados se podrá pasar a un próximo ciclo donde se implementen los restantes casos de uso.
	CU_20	Listar Contratos.	
	CU_25	Buscar por Organismo.	
	CU_26	Buscar por DIP.	
	CU_27	Buscar Contrato.	
	CU_17	Gestionar Suplementos.	
	CU_19	Gestionar Organismo.	
	CU_18	Gestionar Entidad de Contratación.	
	CU_28	Listar por Organismos.	
	CU_29	Buscar Organismo.	
3	CU_7	Generar Reporte Ejecuciones.	Este sería el tercer y último ciclo de desarrollo de esta aplicación, luego de tener los usuarios registrados en el sistema, las obras y los contratos, ya podremos comenzar a manejar las facturas y las solicitudes de pago. Además, al tener ya todos los datos necesarios dentro del sistema podemos obtener los reportes necesarios para la UPI, permitiendo con esto que se tenga la aplicación completa y que cubra todas las necesidades existentes.
	CU_8	Generar Reporte Catálogo de Centros de Costo y Áreas de Responsabilidad.	
	CU_15	Generar Reporte de producción.	
	CU_10	Listar Facturas por definir contrato.	
	CU_11	Listar Facturas.	
	CU_21	Listar Facturas por Objeto de Obra.	
	CU_13	Alertar Solicitud de Pago sin contrato.	
	CU_6	Generar Reporte de Conciliación.	
	CU_12	Gestionar Solicitud de Pago.	
	CU_23	Buscar Solicitud de Pago.	

	CU_14	Asignar Cheque.	
	CU_9	Gestionar Facturas.	
	CU_22	Buscar Facturas.	

2.9 Casos de Uso expandidos.

La descripción textual de los casos de uso del sistema se podrán encontrar en los Anexos y a continuación se brinda un vínculo para cada uno de ellos.

Descripción Textual del Caso de Uso del Sistema “Gestionar Usuario”. Ver **Anexo 9.**

Descripción Textual del Caso de Uso del Sistema “Autenticar Usuario”. Ver **Anexo 10.**

Descripción Textual del Caso de Uso del Sistema “Listar Usuario”. Ver **Anexo 11.**

Descripción Textual del Caso de Uso del Sistema “Gestionar Objeto de Obra”. Ver **Anexo 12.**

Descripción Textual del Caso de Uso del Sistema “Listar Objeto de Obra”. Ver **Anexo 13.**

2.10 Conclusiones

Con la culminación de este capítulo han quedado descritas todas las consideraciones necesarias para los primeros pasos en el cumplimiento de los objetivos propuestos. Se han desarrollado y obtenido hasta el momento los artefactos y documentación resultante de los flujos de trabajo modelado del negocio y requerimientos. Se describen cada uno de los casos de uso y se le da prioridad a aquellos que serán implementados en el primer ciclo de desarrollo, se especificaron los requisitos funcionales y no funcionales obteniendo una total descripción del sistema.

CAPÍTULO 3
ANÁLISIS Y DISEÑO DEL SISTEMA

3.1 Introducción.

En este capítulo se expone el análisis y diseño del sistema, que aportará una nueva visión del sistema propuesto sobre los requisitos funcionales identificados. Primeramente se realiza el análisis, que representa una aproximación al diseño, con los diagramas de clases correspondientes. A continuación se muestra el diseño del sistema, para ello se realizan los diagramas de clases, así como los diagramas de secuencias. Además en este capítulo se presenta el diagrama de clases persistentes y el diagrama entidad relación, que constituyen la base para la construcción de la base de datos.

3.2 Modelo de Análisis.

El Modelo de análisis se realiza para obtener una visión del sistema sobre los requisitos funcionales expresados ya en un lenguaje técnico, constituye una primera aproximación al modelo de diseño. El mismo ofrece ventajas tales como: suavizar la transición al diseño, apoyar el cambio a otra plataforma de programación, servir para obtener una visión general de la propuesta del sistema así como para planificar y dividir el diseño e implementación en pequeños módulos, apoya la aplicación de reingeniería a aplicaciones existentes (al ser en un lenguaje menos técnico ayuda a entender mejor la propuesta de solución). Está orientado a analizar cómo el sistema va a cumplir sus funcionalidades. El análisis debe capturar los requisitos de usuario sin adoptar prematuramente decisiones de implementación, es decir, omitiendo detalles dependientes de la tecnología, y utilizando conceptos extraídos únicamente del dominio del problema.

3.2.1 Modelo de clases de análisis.

Los diagramas de clases del análisis muestran las clases participantes en el caso de uso y las relaciones que existen entre éstas.

Tabla 3.1: Clases del Análisis.

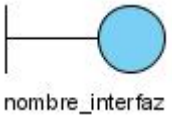
Nombre.	Características.	Representación.
Interfaz	Modelan la interacción entre el sistema y sus actores.	
Entidad	Modelan información que posee larga vida y que es a menudo persistente.	
Control	Coordinan la realización de uno o unos pocos casos de uso coordinando las actividades de los objetos que implementan la funcionalidad del caso de uso.	

Diagrama de Clases del Análisis – Caso de Uso "Gestionar Usuario".

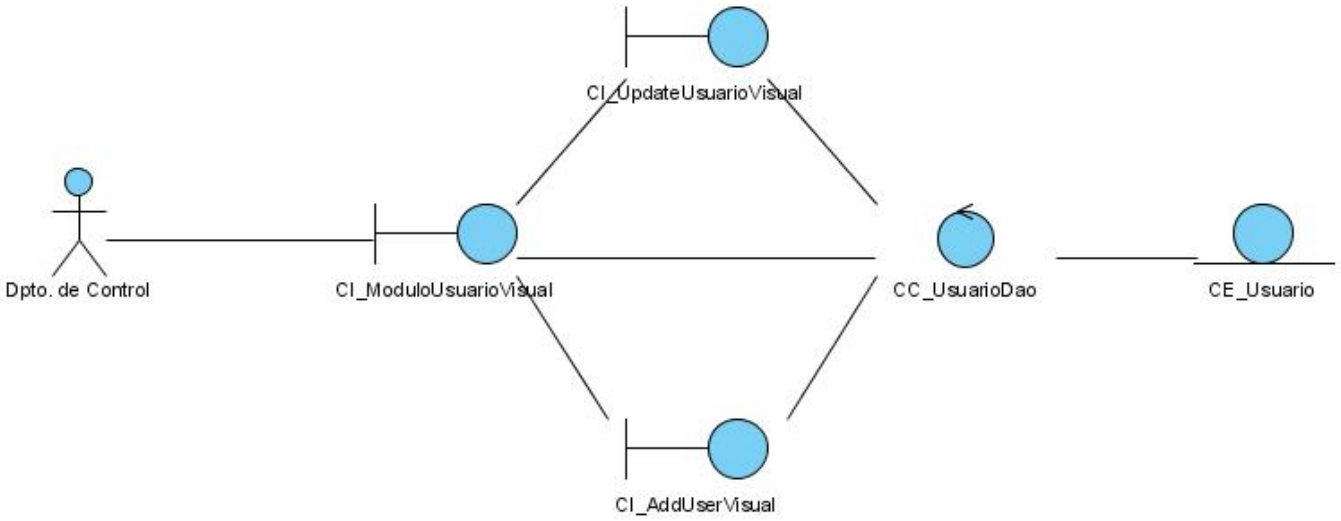


Figura 3.1: Diagrama de Clases del Análisis –Caso de Uso "Gestionar Usuario".

Diagrama de Clases del Análisis – Caso de Uso "Listar Usuarios".

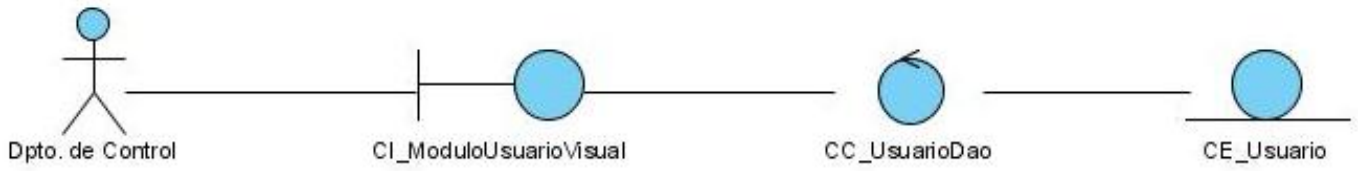


Figura 3.2: Diagrama de Clases del Análisis –Caso de Uso "Listar Usuarios".

Diagrama de Clases del Análisis – Caso de Uso "Autenticar Usuario".

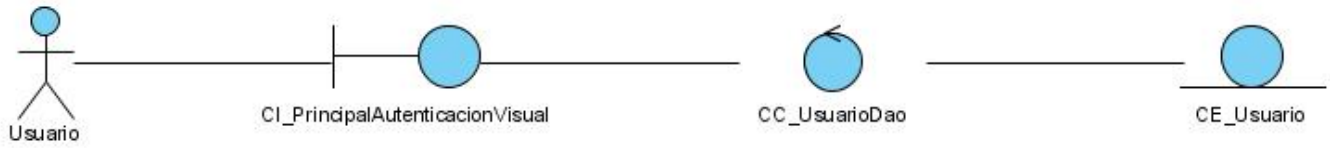


Figura 3.3: Diagrama de Clases del Análisis –Caso de Uso "Autenticar Usuario".

Diagrama de Clases del Análisis – Caso de Uso "Gestionar Objeto de Obra".

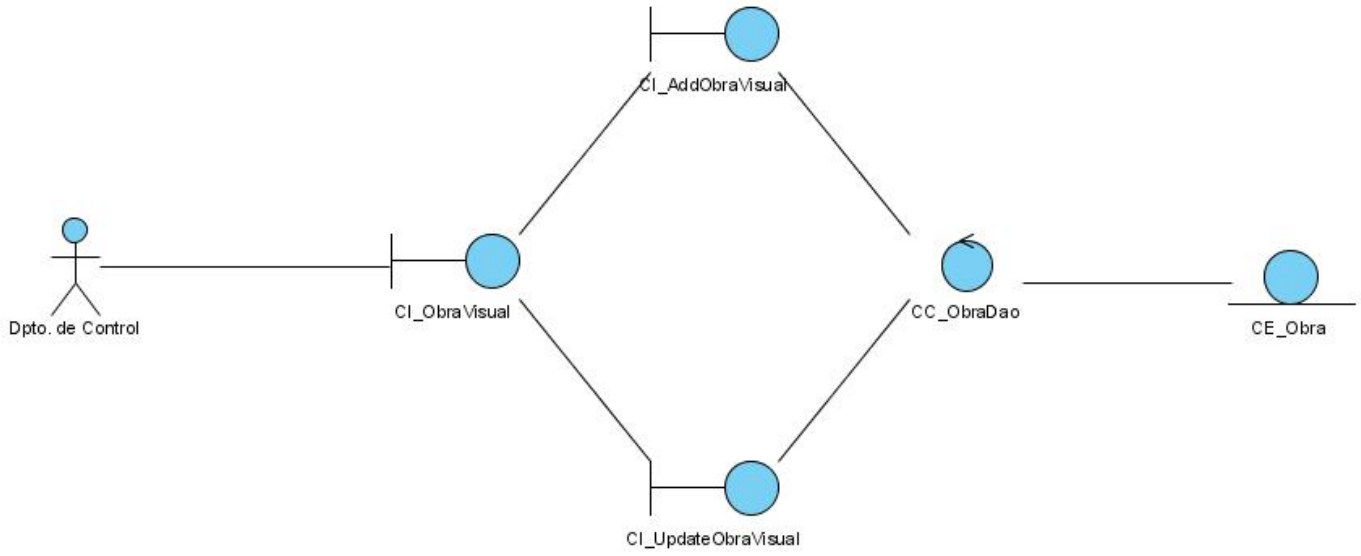


Figura 3.4: Diagrama de Clases del Análisis –Caso de Uso "Gestionar Objeto de Obra".

Diagrama de Clases del Análisis – Caso de Uso "Listar Objeto de Obra".

Figura 3.5: Diagrama de Clases del Análisis –Caso de Uso "Listar Objetos de Obra".

3.2.2 Arquitectura del sistema.

Con el objetivo de comprender mejor el sistema y organizar el desarrollo se hace necesario describir la arquitectura definida por para el sistema, esta se mantendrá durante el diseño y la implementación. La arquitectura que se empleará para el desarrollo de esta aplicación responde a la Arquitectura en 3 capas, la cual proporciona una buena organización y estructuración entre los distintos niveles de abstracción, donde, un cambio en uno de estos niveles no debe proporcionar cambios en los restantes. El diseño más empleado en la actualidad es el de tres capas, siendo estas:

Capa de presentación: es la capa mediante la cual el sistema le brinda la posibilidad al usuario de interactuar con él, se encarga de mostrar la información y captura la misma con el objetivo de procesarla, además en ella se realiza una primera validación de la información con el objetivo de evitar posibles errores. Esta capa solo se comunica con la capa lógica del negocio.

Capa lógica de negocio: es una de las capas más importante, se encarga de recibir las peticiones de los usuarios y enviar las respuestas, aquí es donde se establecen todas las reglas que deben cumplirse y se manipula toda la información. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y mostrar los resultados, y con la capa de acceso a datos, para solicitar al gestor de base de datos almacenar o recuperar datos de él, para el acceso a datos esta capa se relaciona con una clase que implementa la interfaz del modelo de persistencia que responde a un patrón llamado Factory la cual es una puerta de enlace entre la capa de Acceso a Datos y la capa de Lógica de Negocio.

Capa de acceso a datos: Es la capa encargada de manejar todo el flujo de información que entra y sale de la fuente de datos, así como la conexión a la misma. Como se explicaba en la capa anterior aquí se implementa una clase interfaz que se encarga de establecer un enlace con la capa lógica del negocio. Ya que es la capa encargada de establecer la conexión con la base de datos ella implementa el patrón de diseño Singlenton cuyo principal objetivo, y para el cual está diseñado, es restringir la creación de objetos pertenecientes a una clase, logrando que una clase sólo tenga una instancia.

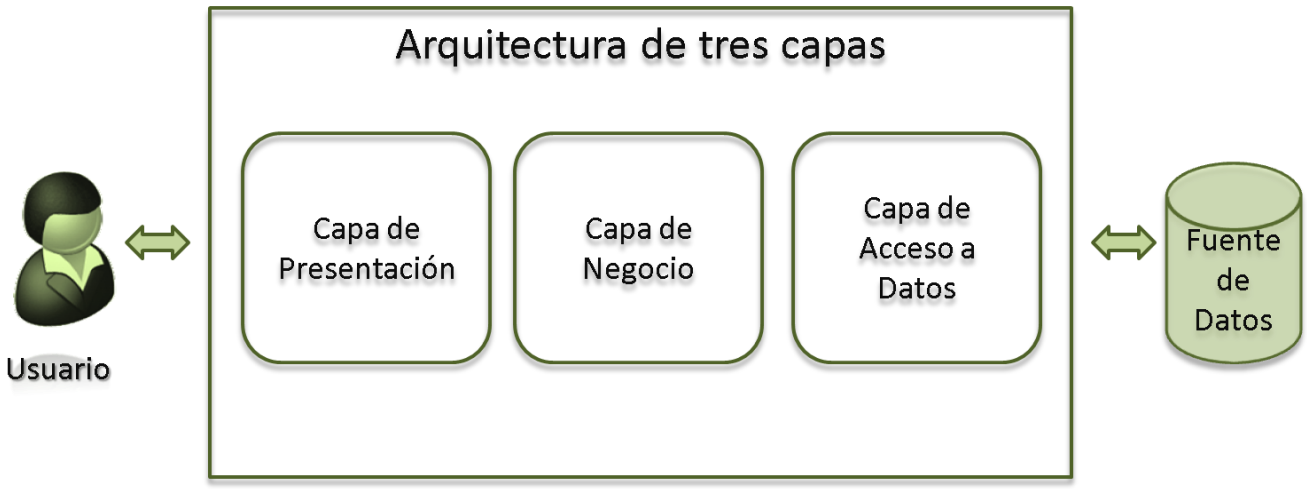


Figura 3.6: Arquitectura de tres capas.

3.2.3 Patrones de Diseño.

"Una arquitectura orientada a objetos bien estructurada está llena de patrones. La calidad de un sistema orientado a objetos se mide por la atención que los diseñadores han prestado a las colaboraciones entre sus objetos. Los patrones conducen a arquitecturas más pequeñas, más simples y más comprensibles". *(Booch, 1996)*

La capa de acceso a datos de ConInver utiliza el patrón DAO (Data Access Object); el problema que viene a resolver este patrón es el de contar con diversas fuentes de datos De tal forma que se encapsula la forma de acceder a la fuente de datos. Este patrón surge históricamente de la necesidad de gestionar una diversidad de fuentes de datos, aunque su uso se extiende al problema de encapsular no sólo la fuente de datos, sino además ocultar la forma de acceder a los datos. Se trata de que el software cliente se centre en los datos que necesita y se olvide de cómo se realiza el acceso a los datos o de cuál es la fuente de almacenamiento; así un cambio en el origen de los datos o en la manera de recuperarlos no afecta la capa superior siempre que se implemente la interfaz correspondiente.

Algunas características:

- No es imprescindible, pero en proyectos de cierta complejidad resulta útil que el DAO implemente una interfaz. De esta forma los objetos cliente tienen una forma unificada de acceder a los DAO.
- El DAO accede a la fuente de datos y la encapsula para los objetos clientes, que oculta tanto la fuente como el modo (JDBC[59]) de acceder a ella.

3.3 Modelo de Diseño.

El modelo de diseño es un refinamiento del análisis. Un modelo de objetos que describe la realización física de los casos de uso. Se centra en los impactos que producen en el sistema a desarrollar los requerimientos funcionales y no funcionales. Se determina la arquitectura general del sistema y su comportamiento dinámico, adaptando la especificación realizada en la etapa anterior. En esta fase se establece el comportamiento dinámico del sistema, es decir, como debe reaccionar ante los acontecimientos.

El resultado obtenido de la etapa de Diseño facilita enormemente la implementación posterior del sistema, pues proporciona la estructura básica del sistema y como los diferentes componentes actúan y se relacionan entre ellos.

3.3.1 Diagramas de Interacción.

Los diagramas de secuencia y los diagramas de colaboración (ambos llamados diagramas de interacción) son dos de los cinco tipos de diagramas de UML que se utilizan para modelar los aspectos dinámicos de los sistemas. Un diagrama de interacción muestra una interacción, que consiste en un conjunto de objetos y sus relaciones, incluyendo los mensajes que se pueden enviar entre ellos.

[59] Es el acrónimo de Java Database Connectivity, un API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java.

Un diagrama de secuencia es un diagrama de interacción que destaca la ordenación temporal de los mensajes; un diagrama de colaboración es un diagrama de interacción que destaca la organización estructural de los objetos que envían y reciben mensajes.

Los diagramas de interacción se utilizan para modelar los aspectos dinámicos de un sistema. La mayoría de las veces, esto implica modelar instancias concretas o prototípicas de clases, interfaces, componentes y nodos, junto con los mensajes enviados entre ellos, todo en el contexto de un escenario que ilustra un comportamiento. Los diagramas de interacción pueden utilizarse para visualizar, especificar, construir y documentar la dinámica de una sociedad particular de objetos, o se pueden utilizar para modelar un flujo de control particular de un caso de uso.

3.3.1.1 Diagrama de Secuencia.

El diagrama de secuencia contiene detalles de implementación del escenario, incluyendo los objetos y clases de secuencia que se usan para implementar este, y mensajes pasados entre los objetos. Para realizar los diagramas que a continuación se muestran se examinó la descripción de los casos de uso para determinar qué objetos eran necesarios para la implementación, convirtiendo los pasos de este en un flujo de acciones sobre el cual caminar.

Los diagramas de secuencia son de gran importancia en el diseño de un sistema debido a que permiten observar las interacciones que ocurren entre los distintos objetos que participan en un escenario determinado.

Un escenario de un caso de uso es un "camino" completo a través del caso de uso. Los usuarios finales del sistema pueden seguir muchos caminos cuando ejecutan la funcionalidad especificada en el caso de uso. Siguiendo el flujo básico serían un escenario.

Diagrama de secuencia – Caso de Uso “Gestionar Usuario” – Escenario “Registrar Usuario”. Ver **Anexo 14.**

Diagrama de secuencia – Caso de Uso “Gestionar Usuario” – Escenario “Actualizar Usuario”. Ver **Anexo 15.**

Diagrama de secuencia – Caso de Uso “Gestionar Usuario” – Escenario “Activar/Desactivar Usuario”. Ver **Anexo 16.**

3.3.2 Diagrama de clases del diseño.

A continuación se muestra el diagrama del diseño donde se describe la estructura del sistema mostrando sus clases, atributos y las relaciones entre ellos. Este diagrama es muy utilizado durante el diseño de los sistemas, y a partir de él se crea el diseño conceptual de la información que se manejará en el sistema, y los componentes que se encargarán del funcionamiento y la relación entre uno y otro.

Diagrama de clases del diseño.

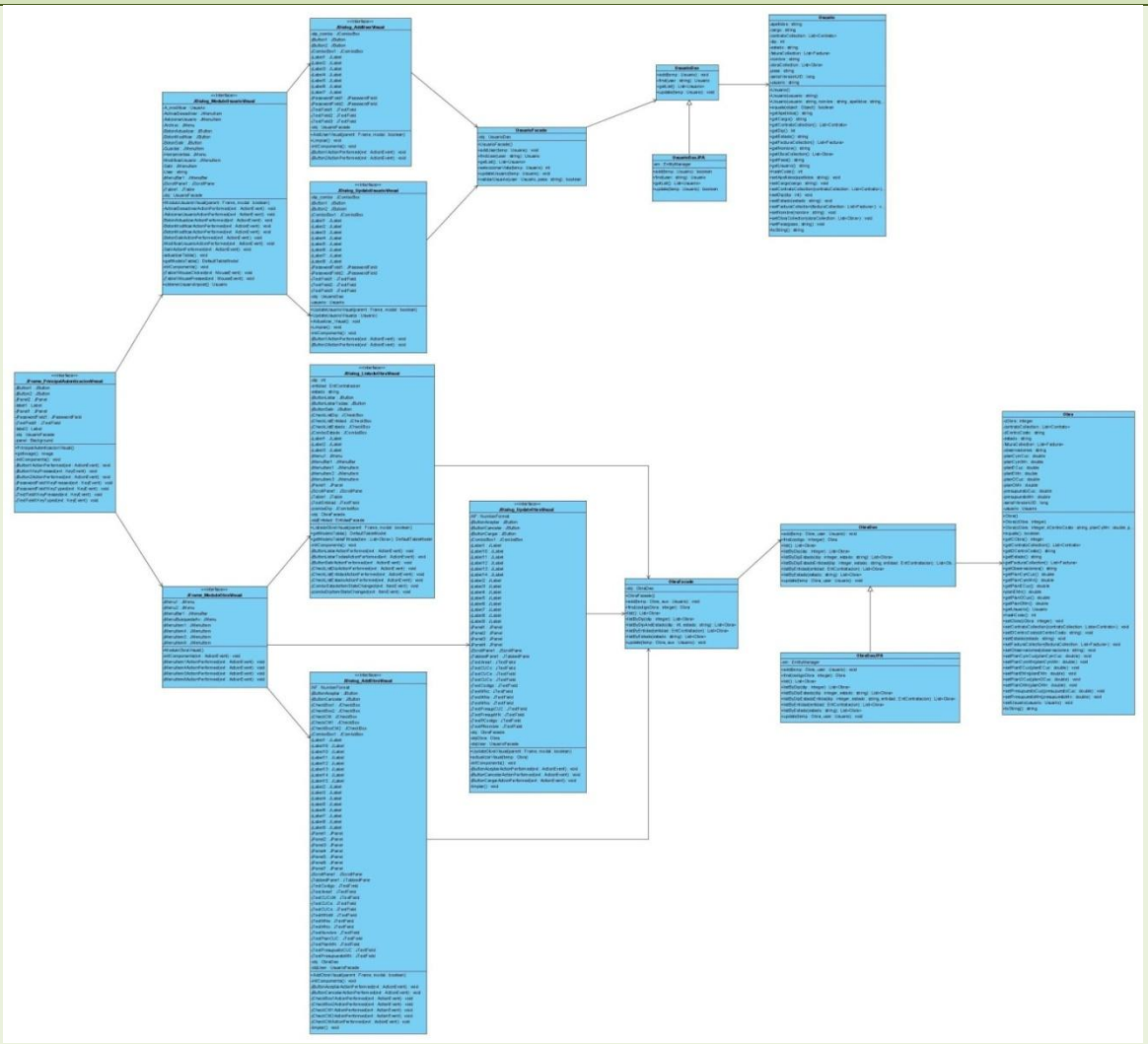


Figura 3.7: Diagrama de clases del diseño.

3.3.2.1 Breve descripción de las clases contenidas en el diagrama de clases del diseño.

Capa de presentación.

JFrame_PrincipalAutenticacionVisual: Esta clase es el respaldo de la vista encargada de presentar una interfaz al usuario para acceder al sistema en dependencia del rol que desempeñe en el mismo.

JDialog_ModuloUsuarioVisual: Esta clase es el respaldo de la vista encargada de presentar una interfaz al usuario con todas las funcionalidades posibles a realizar en el módulo de usuarios, como registrar un nuevo usuario, modificar uno ya existente y activar o desactivar un usuario.

JDialog_AddUserVisual: Esta clase es el respaldo de la vista encargada de presentar una interfaz al usuario con todos los campos necesarios para crear y adicionar un usuario nuevo, mostrando los campos necesarios para ello, como lo son: usuario, nombre, apellidos, cargo, DIP, contraseña y repetir contraseña.

JDialog_UpdateUsuarioVisual: Esta clase es el respaldo de la vista encargada de presentar una interfaz al usuario con todos los datos que tiene un usuario, posibles a modificar, dando la posibilidad de modificarlos todos menos el usuario, dato que no podrá ser modificado nunca.

JFrame_ModuloObraVisual: Esta clase es el respaldo de la vista encargada de presentar una interfaz al usuario con todas las funcionalidades posibles a realizar en el módulo de obras, donde el Dpto. de Control podrá registrar una nueva obra, modificar una ya existente y tener un listado actualizado de todas las obras registradas en el sistema.

JDialog_ListadoObraVisual: Esta clase es el respaldo de la vista encargada de presentar una interfaz al usuario con las posibles formas de listar los objetos de obras. Estas obras pueden ser listadas por varios criterios de búsqueda, se pueden filtrar por la DIP a la que pertenecen y por el estado de esta obra, si es abierta, cerrada o en garantía, además de mostrar el listado general de todas las obras.

JDialog_AddObraVisual: Esta clase es el respaldo de la vista encargada de presentar una interfaz al usuario con todos los campos necesarios para crear y adicionar una obra nueva. Para hacer el registro

tendrá que insertar datos como: el código del centro de costo, el nombre de este, el plan, que puede ser desglosado por componentes, siendo estos Construcción y Montaje, Equipos y Otros, incorporando de cada uno el presupuesto en MN y en CUC, además de su total; esta interfaz cuenta con un campo de Observaciones en el cual se pueden hacer anotaciones que sean de interés para posteriores consultas.

JDialog_UpdateObraVisual: Esta clase es el respaldo de la vista encargada de presentar una interfaz al usuario con las posibilidades de buscar y mostrar todos los datos que tiene una obra, posibles a modificar. Para realizar la actualización es necesario cargar esta obra, se le debe introducir al sistema el código de obra a cargar; luego de ser especificado se mostraran todos los datos referentes a la obra, dando la posibilidad de modificarlos, exceptuando el código de centro de costo.

Capa de acceso a datos.

UsuarioFacade: Esta clase representa una interfaz unificada y sencilla que funge como intermediaria entre las capas de presentación del módulo de usuarios y la capa de acceso a datos correspondiente al mismo.

ObraFacade: Esta clase representa una interfaz unificada y sencilla que funge como intermediaria entre las capas de presentación del módulo de usuarios y la capa de acceso a datos correspondiente al mismo.

Capa lógica de negocio.

UsuarioDao: Es la interfaz que expone a la capa de lógica de negocio los métodos del DAO para persistir la entidad de dominio Usuario.

UsuarioDaoJPA: Esta clase es la implementación de la interfaz que realmente se conecta a la base de datos usando Hibernate. Dando de esta forma la funcionalidad real a los métodos de UsuarioDao.

ObraDao: Es la interfaz que expone a la capa de lógica de negocio los métodos del DAO para persistir la entidad de dominio Obra.

ObraDaoJPA: Esta clase es la implementación de la interfaz que realmente se conecta a la base de datos usando Hibernate. Dando de esta forma la funcionalidad real a los métodos de ObraDao.

Entidades de dominio.

Obra: Es la entidad de dominio a ser persistida por los métodos expuesto en la interfaz ObraDAO.

Usuario: Es la entidad de dominio a ser persistida por los métodos expuesto en la interfaz UsuarioDAO.

3.3.3 Diseño de la Base de Datos.

Uno de los pasos cruciales en la construcción de una aplicación que maneje una base de datos, es sin duda el diseño de la base de datos. En la base de datos que se utiliza en la aplicación que se está desarrollando como propuesta de solución, para lograr el acceso eficiente a la información con redundancia mínima, se toman varias consideraciones, entre las que se encuentran: la velocidad y facilidad de acceso a la información para extraerla.

En el diseño de la base de datos que soporte los datos para el sistema a desarrollar es necesario aclarar que éste se hará completo, o sea, en el capítulo 1 describimos solos los casos de usos que serán implementados en el primer ciclo de desarrollo de la aplicación, pero teniendo en cuanta la importancia de la base de datos y para evitar una modificación posterior se decide realizar el modelado de la base de datos en su totalidad, incluyendo todas las entidades persistentes que están relacionadas.

3.3.3.1 Clases persistente.

Todas las clases identificadas en el dominio del análisis no son persistentes. La persistencia es la capacidad de un objeto de mantener su valor en el espacio y en el tiempo. Lo contrario son las clases temporales que son manejadas y almacenadas por el sistema en tiempo de ejecución por lo que dejan de existir cuando termina el programa.

Una clase del diseño es una construcción similar a la implementación del sistema:

- Las relaciones entre clases de diseño se traducen de manera directa al lenguaje:
  - ✓ Generalización: herencia.
  - ✓ Asociaciones, agregaciones: atributos.

- Los métodos de una clase del diseño tienen correspondencia directa con el correspondiente método en la implementación de las clases.
- Una clase de diseño puede proporcionar interfaces si tiene sentido hacerlo en el lenguaje de programación.

Diagrama de clases persistentes. Ver **Anexo 17.**

3.3.3.2 Modelo de datos.

Una base de datos es una serie de datos organizados y relacionados entre sí, los cuales pueden ser recolectados y explotados por sistemas de gestión de información. Con el objetivo de lograr la persistencia de los datos y que estos puedan ser utilizados en cualquier momento por el usuario del sistema encargado de gestionar la información se utiliza una base de datos relacional. El uso de esta técnica de almacenamiento está muy difundido actualmente debido a las garantías que ofrece en cuanto a la durabilidad de los datos se refiere.

El modelo de los datos es el encargado de describir de una forma abstracta la representación lógica y física de los datos persistentes en el sistema. Básicamente consiste en una colección de conceptos que se emplean para describir la estructura de la base de datos y que está constituida por entidades, atributos y relaciones.

Modelo de datos.

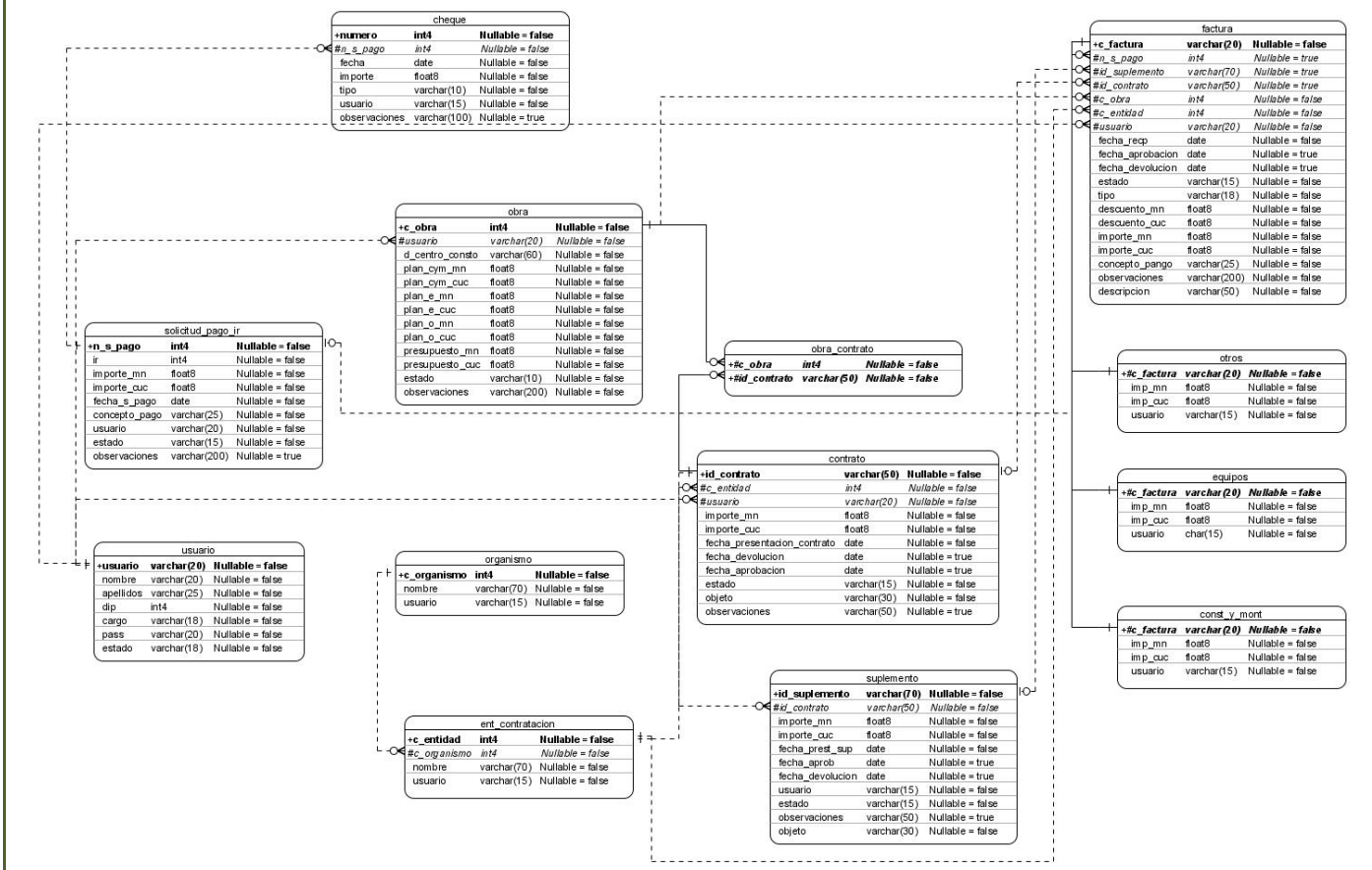


Figura 3.8: Modelo de datos.

3.3.3.2.1 Descripción de las tablas de la Base de Datos.

Se describen las tablas en las que se almacena toda la información necesaria para desarrollar el sistema, permitiendo conocer el atributo, su tipo y descripción.

Tabla 3.2: Descripción de la tabla de la base de datos “solicitud_pago_ir”.

Nombre: solicitud_pago_ir

Descripción: Esta tabla contiene los datos de las solicitudes de pago.		
Atributo	**Tipo**	**Descripción**
n_s_pago	int	Representa el identificador de la solicitud de pago.
ir	int	Representa el número del informe de recepción.
importe_mn	float	Representa el valor que será pagado en MN.
importe_cuc	float	Representa el valor que será pagado en CUC.
fecha_s_pago	date	Representa la fecha en que se realiza la solicitud de pago.
concepto_pago	varchar (25)	Representa el concepto de pago de la solicitud de pago.
usuario	varchar (20)	Representa el usuario que incidió sobre la solicitud de pago.
estado	varchar (15)	Representa el estado en el que se encuentra la solicitud de pago
observaciones	varchar (200)	Representa las observaciones que se realizan de la solicitud de pago.

Tabla 3.3: Descripción de la tabla de la base de datos “cheque”.

Nombre: cheque		
Descripción: Esta tabla contiene los datos de los cheques.		
Atributo	**Tipo**	**Descripción**
numero	int	Representa el identificador del cheque.
n_s_pago	int	Representa el número de la solicitud de pago que respalda el cheque.
fecha	date	Representa la fecha en que es emitido el cheque.
Importe	float	Representa el valor por el que es emitido el cheque.
tipo	varchar (10)	Representa el tipo de cheque, si es de MN o CUC.
usuario	varchar (15)	Representa el usuario que incide sobre el cheque.
observaciones	varchar (100)	Representa las observaciones que se realizan del cheque.

Tabla 3.4: Descripción de la tabla de la base de datos "usuario".

Nombre: usuario		
Descripción: Esta tabla contiene todos los datos de los usuarios.		
Atributo	**Tipo**	**Descripción**
usuario	varchar (20)	Representa el identificador del usuario.
nombre	varchar (20)	Representa el nombre del usuario.
apellidos	varchar (25)	Representa los apellidos del usuario.
dip	int	Representa la DIP a la que pertenece el usuario.
cargo	varchar ()18	Representa el cargo que tiene el usuario.
pass	varchar (20)	Representa la contraseña del usuario para acceder a la aplicación.
estado	varchar (18)	Representa el estado en el que se encuentra el usuario.

Tabla 3.5: Descripción de la tabla de la base de datos "organismo".

Nombre: organismo		
Descripción: Esta tabla contiene los datos de los organismos.		
Atributo	**Tipo**	**Descripción**
c_organismo	int	Representa el identificador del organismo.
usuario	varchar (15)	Representa el usuario que accedió a este recurso.
nombre	varchar (70)	Representa el nombre del organismo.

Tabla 3.6: Descripción de la tabla de la base de datos "factura".

Nombre: factura		
Descripción: Esta tabla contiene los datos de las facturas.		
Atributo	**Tipo**	**Descripción**

c_factura	varchar (20)	Representa el identificador de la factura.
n_s_pago	int	Representa el identificador de la solicitud de pago asociada.
id_suplemento	varchar (70)	Representa el identificador del suplemento asociado a la factura.
id_contrato	varchar (50)	Representa el identificador del contrato asociado a la factura.
c_obra	int	Representa el identificador de la obra que respalda esta factura.
c_entidad	int	Representa el identificador de la entidad a la que pertenece la factura.
usuario	varchar (20)	Representa el usuario que accedió a este recurso.
fecha_recp	date	Representa la fecha en la que es recibida la factura.
fecha_aprobacion	date	Representa la fecha en la que es aprobada la factura.
fecha_devolucion	date	Representa la fecha en que fue devuelta la factura en caso de que esta no haya sido aprobada.
estado	varchar (15)	Representa el estado en el que se encuentra la factura.
tipo	varchar (18)	Representa el tipo de la factura.
descuento_mn	float	Representa el valor en MN del descuento.
descuento_cuc	float	Representa el valor en CUC del descuento.
importe_mn	float	Representa el importe en MN de la factura.
importe_cuc	float	Representa el importe en CUC de la factura.
concepto_pago	varchar (25)	Representa el concepto por el que se paga la factura.
observaciones	varchar (200)	Representa las observaciones que se le hacen a la factura.
descripción	varchar (50)	Representa la descripción de la factura.

Tabla 3.7: Descripción de la tabla de la base de datos “ent_contratacion”.

Nombre: ent_contratacion

Descripción: Esta tabla contiene los datos de las entidades de contratación.		
Atributo	**Tipo**	**Descripción**
c_entidad	int	Representa el identificador de la entidad de contratación.
c_organismo	int	Representa el identificador del organismo al que pertenece la entidad de contratación.
nombre	varchar (70)	Representa el nombre de la entidad de contratación.
usuario	varchar (15)	Representa el usuario que accedió a este recurso.

Tabla 3.8: Descripción de la tabla de la base de datos "obra".

Nombre: obra		
Descripción: Esta tabla contiene los datos de los objetos de obras o centros de costo.		
Atributo	**Tipo**	**Descripción**
c_obra	int	Representa el identificador de la obra.
usuario	varchar (15)	Representa el usuario que accedió a este recurso.
d_centro_costo	varchar (60)	Representa la descripción de la obra.
plan_cym_mn	float	Representa el plan del componente construcción y montaje en MN.
plan_cym_cuc	float	Representa el plan del componente construcción y montaje en CUC.
plan_e_mn	float	Representa el plan del componente equipos en MN.
plan_e_cuc	float	Representa el plan del componente equipos en CUC.
plan_o_mn	float	Representa el plan del componente otros en MN.
plan_o_cuc	float	Representa el plan del componente otros en CUC.
presupuesto_mn	float	Representa el valor del presupuesto de la obra en MN.
presupuesto_cuc	float	Representa el valor del presupuesto de la obra en CUC.
estado	varchar (10)	Representa el estado de la obra.
observaciones	varchar (200)	Representa las observaciones que se le hacen a la obra.

Tabla 3.9: Descripción de la tabla de la base de datos "obra_contrato".

Nombre: obra_contrato		
Descripción: Esta tabla contiene los datos de los objetos de obras y contratos.		
Atributo	**Tipo**	**Descripción**
c_obra	int	Representa el identificador de la obra.
id_contrato	varchar (50)	Representa el identificador del contrato.

Tabla 3.10: Descripción de la tabla de la base de datos "contrato".

Nombre: contrato		
Descripción: Esta tabla contiene los datos de los contratos.		
Atributo	**Tipo**	**Descripción**
id_contrato	varchar (50)	Representa el identificador del contrato.
c_entidad	int	Representa el identificador de la entidad de contratación.
usuario	varchar (20)	Representa el usuario que accedió a este recurso.
importe_mn	float	Representa el valor en MN del importe del contrato.
importe_cuc	float	Representa el valor en CUC del importe del contrato.
fecha_presentacion_ contrato	date	Representa la fecha en la que se es presentado el contrato.
fecha_devolucion	date	Representa la fecha en la que es devuelto el contrato.
fecha_aprobacion	date	Representa la fecha en la que se aprobó el contrato.
estado	varchar (15)	Representa el estado del contrato.
objeto	varchar (30)	Representa el nombre de la obra asociada.
observaciones	varchar (50)	Representa las observaciones hechas al contrato.

Tabla 3.11: Descripción de la tabla de la base de datos "suplemento".

Nombre: suplemento		
Descripción: Esta tabla contiene los datos de los suplementos.		
Atributo	**Tipo**	**Descripción**
id_suplemento	varchar (70)	Representa el identificador del suplemento.
id_contrato	varchar (50)	Representa el identificador del contrato.
usuario	varchar (20)	Representa el usuario que accedió a este recurso.
importe_mn	float	Representa el valor en MN del importe del suplemento.
importe_cuc	float	Representa el valor en CUC del importe del suplemento.
fecha_prest_sup	date	Representa la fecha en la que se es presentado el suplemento.
fecha_devolucion	date	Representa la fecha en la que es devuelto el suplemento.
fecha_aprob	date	Representa la fecha en la que se aprobó el suplemento.
estado	varchar (15)	Representa el estado del suplemento.
objeto	varchar (30)	Representa el nombre de la obra asociada.
observaciones	varchar (50)	Representa las observaciones hechas al suplemento.

Tabla 3.12: Descripción de la tabla de la base de datos "otros".

Nombre: otros		
Descripción: Esta tabla contiene los datos del componente otros.		
Atributo	**Tipo**	**Descripción**
id_factura	varchar (20)	Representa el identificador de la factura.
usuario	varchar (15)	Representa el usuario que accedió a este recurso.
importe_mn	float	Representa el valor en MN del importe del componente otros.
importe_cuc	float	Representa el valor en CUC del importe de componente otros.

Tabla 3.13: Descripción de la tabla de la base de datos "equipos".

Nombre: equipos		
Descripción: Esta tabla contiene los datos del componente equipos.		
Atributo	**Tipo**	**Descripción**
id_factura	varchar (20)	Representa el identificador de la factura.
usuario	varchar (15)	Representa el usuario que accedió a este recurso.
importe_mn	float	Representa el valor en MN del importe del componente equipos.
importe_cuc	float	Representa el valor en CUC del importe del componente equipos.

Tabla 3.14: Descripción de la tabla de la base de datos "const_y_mont".

Nombre: const_y_mont		
Descripción: Esta tabla contiene los datos de los suplementos.		
Atributo	**Tipo**	**Descripción**
id_factura	varchar (20)	Representa el identificador de la factura.
usuario	varchar (15)	Representa el usuario que accedió a este recurso.
importe_mn	float	Representa el valor en MN del importe del componente construcción y montaje.
importe_cuc	float	Representa el valor en CUC del importe del componente construcción y montaje.

3.3.4 Interfaz.

La interfaz es la ventana del software, por lo que debe ser amigable, sencilla, de fácil entendimiento y manejo para el usuario. Este es el elemento más importante para el usuario, ya que es la parte del sistema con la que interactúa y que le facilita además el acceso a los recursos. El diseño de esta interfaz comenzó con la identificación de los requisitos del usuario, de las tareas y del entorno. Debe reflejar

equilibrio en la organización de la información, mostrando en todas las interfaces la información en el mismo orden, optimizar la cantidad de elementos en la pantalla, ayudando a una mejor comprensión de la información mostrada, además, cada elemento de la pantalla seguirá el mismo patrón de tamaño, color y forma.

3.3.5 Tratamiento de errores.

Para una mayor seguridad y confiabilidad de los usuarios con el sistema se lleva a cabo el tratamiento de los errores que se puedan generar durante el trabajo con la aplicación. Se ve de manifiesto cuando utilizamos expresiones regulares brindadas por la tecnología de programación Java, que proporciona una fortaleza, garantizando rapidez de ejecución del sistema, recibiendo el servidor solo la información correcta, de forma tal que las operaciones de inserción o modificación de registros en la base de datos se realicen correctamente. Esto también se garantiza en las clases interfaz de usuario donde se implementan funciones que validan toda la entrada de datos permitiendo esto que no se introduzcan datos erróneos, en caso que el usuario desee introducir algún dato en un formato diferente al requerido se le notificará del error, también lo hará cuando deje algún campo obligatorio en blanco. Cada vez que es lanza una excepción u ocurre un error en el sistema el usuario inmediatamente es notificado de dicho error y de las posibles causas que generaron el mismo. Estos son descritos en un lenguaje que el usuario entiende y los mensajes proporcionan consejos constructivos para recuperarse del error.

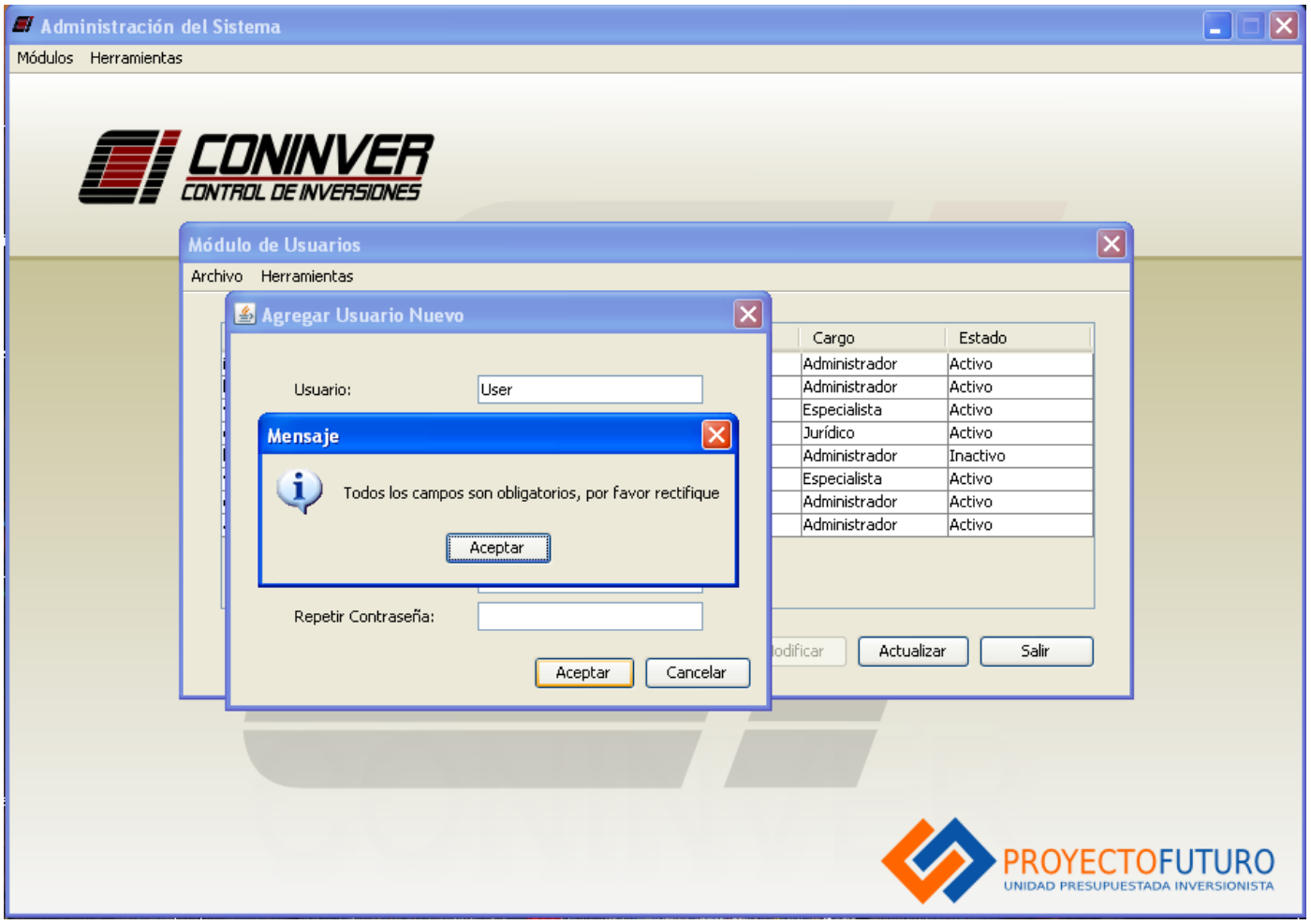


Figura 3.9: Mensaje de notificación.

Las entradas de datos a las descripciones de cada documento son validadas para evitar que se produzcan errores en el futuro. Utilizando mensajes de confirmación para acciones tales como la creación y modificación de la información, estos se manejan de forma que el usuario sepa en cada momento como se realizan las acciones que él lleva a cabo.

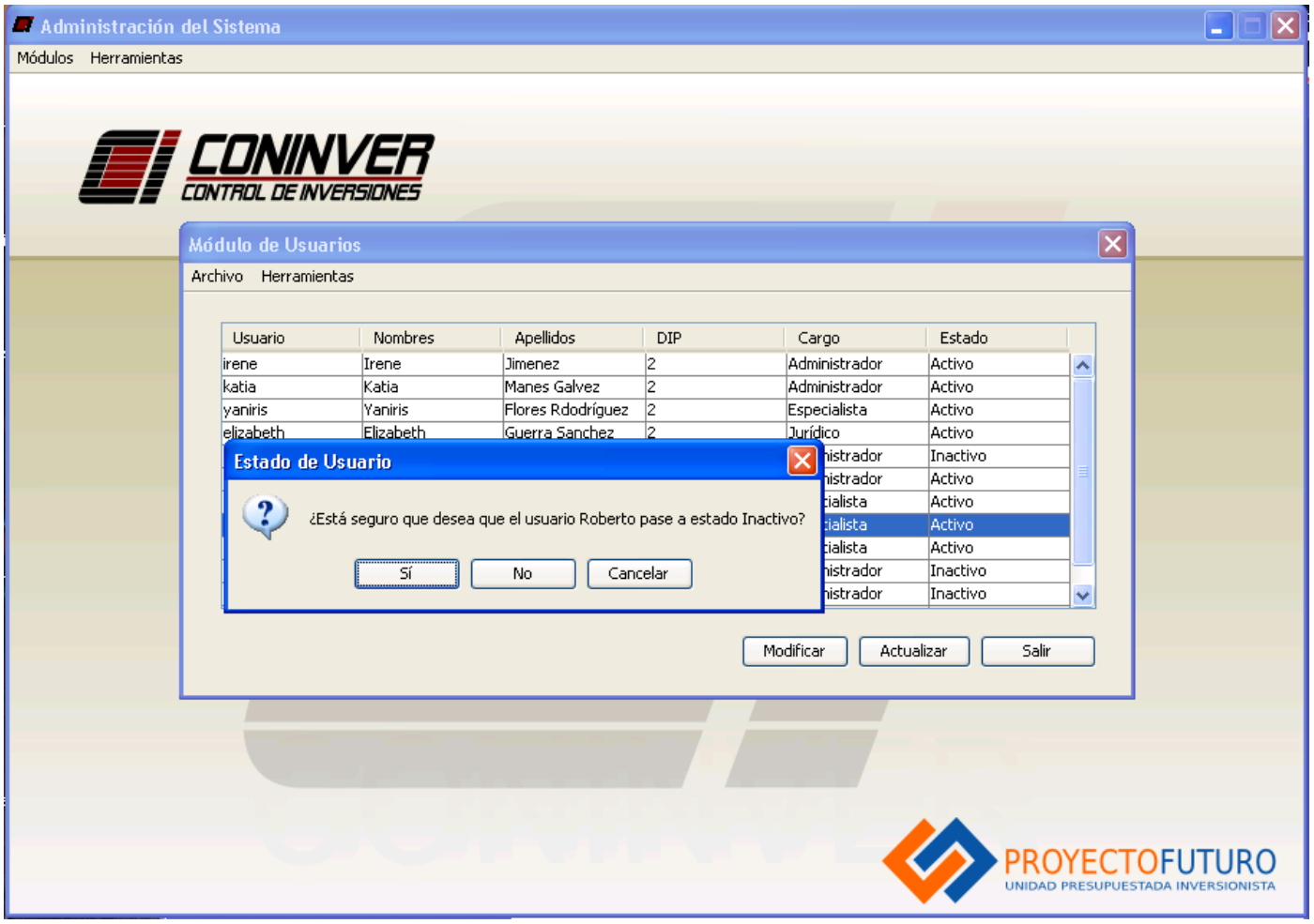


Figura 3.10: Manejo de confirmación.

La notificación se realiza con el objetivo de que el usuario sepa en todo momento la acción que va a realizar, ya que la misma puede traer problemas posteriores.

Figura 3.11: Mensaje de error.

La notificación de los errores se realiza mediante mensajes de error que genera el sistema, de tal modo que el usuario entienda la naturaleza del error y pueda corregirlo.

3.3.6 Concepción de la Ayuda.

Se realizará un Manual de Usuario o Ayuda en formato PDF adjunta al sistema, donde se incluirá una explicación detallada de cómo acceder y trabajar con el mismo, las restricciones que tienen como usuario y cada una de las funcionalidades a las que tienen acceso. Esto le facilitará el trabajo con la aplicación, permitiéndole a cada momento la posibilidad de saber cómo hacer algo o cómo resolver cierto problema que se le presentase, asegurando con esto que el uso del producto sea lo más eficiente posible.

3.3.7 Seguridad.

Debido a la importancia y sensibilidad de los datos que se manejan en la aplicación se hace necesario garantizar la protección y seguridad de la información para lograr el óptimo funcionamiento del sistema y preservar la integridad, autenticidad y confiabilidad de la misma. La seguridad es un elemento esencial cuando se va a desarrollar un sistema informático por tanto, se hace necesario implementar algunas tareas de estricto cumplimiento que permitan asegurar la integridad de los datos. La base de datos permanecerá en lugar restringido y asegurado, además de realizarse salvas de la información acumulada.

Hay que tener en cuenta que el acceso a la aplicación es de forma controlada, cada usuario tiene acceso a la información que le concierne. De forma general, en el sistema ninguna información registrada podrá ser eliminada, esto garantiza que no sea eliminado algún elemento por error, uno de los aspectos más importantes es que tampoco los usuarios registrados podrán ser eliminados, ya que, en cada acción realizada en el sistema se queda guardado el nombre de usuario que realizó la misma, permitiendo saber en todo momento quien realizó algún cambio en el sistema, lo que garantiza la seguridad de la información almacenada. Están validados todos los campos de entrada de datos, garantizando que no se introduzca información que posteriormente pueda afectar el buen funcionamiento de la aplicación.

Para lograr una mayor seguridad en los datos se usa la librería Jasypt 1.5. Jasypt (Java Simplified Encryption) es una librería orientada a permitir a los desarrolladores añadir capacidades de cifrado a sus proyectos de manera sencilla, incluyendo: password digesting, cifrado de textos o binarios, integración con Hibernate para cifrado transparente de los datos persistidos e integración con Spring Security[60] para mejorar la seguridad de los passwords manejados por este framework. También permite el cifrado de valores en archivos .properties y su lectura de manera transparente. Además de usar como se explicó anteriormente el protoclo de encriptación SSL para la conexión segura con el servidor de base de datos.

[60] Spring Security: Antes conocido como Acegi es una API código abierto que provee de servicios de autenticación y autorización a las aplicaciones basadas en Spring.

3.4 Conclusiones.

La fase de Análisis y Diseño es una de las más importantes propuestas por la metodología RUP, ya que brinda una idea completa de lo que es realmente el software y se materializan con bastante precisión los requerimientos del cliente.

En este capítulo se obtuvo el modelo de análisis y el modelo del diseño para el sistema propuesto. Sé obtuvo diferentes artefactos de este flujo de trabajo como son los diagramas de clases del análisis y del diseño, mediante la cual se explicó de forma más detallada las responsabilidades de cada clase, lográndose de esta forma una mayor visión y mejor comprensión de lo que se quiere obtener en el sistema. Se plantearon los principios del diseño incluyendo en él aquellos patrones utilizados para modelar en diseño teniendo en cuenta la arquitectura propuesta. Además, se describieron la interfaz, el tratamiento a los errores, la concepción de ayuda y la seguridad del sistema.

CAPÍTULO 4

IMPLEMENTACIÓN Y PRUEBA DEL SISTEMA

4.1 Introducción.

En el presente capítulo se llevará a cabo el desarrollo de los flujos de trabajo de implementación y prueba. Se muestran además la situación física de los distintos componentes lógicos desarrollados, los que son mostrados a través del modelo de despliegue y la organización del sistema mediante el modelo de componentes el cual representa la vista estática del sistema. Se definirá el modelo de prueba, describiendo cómo hacer las pruebas al sistema verificando que los requerimientos hayan sido implementados.

4.2 Implementación.

En la implementación empezamos con el resultado del diseño e implementamos el sistema en términos de componentes, es decir, ficheros de código fuente, scripts, ficheros de código binario, ejecutables y similares. El flujo de trabajo de implementación describe cómo los elementos del modelo del diseño se implementan en términos de componentes y cómo estos se organizan de acuerdo a los nodos específicos en el modelo de despliegue.

4.2.1 Modelo de Implementación.

Los diagramas de despliegue y componentes conforman lo que se conoce como un modelo de implementación al describir los componentes a construir, su organización y dependencia entre nodos físicos en los que funcionará a aplicación. Entre los componentes podemos encontrar datos, archivos, ejecutables, código fuente y los directorios. Fundamentalmente se describe la relación que existe desde

los paquetes y clases del modelo de diseño a subsistemas y componentes físicos, es decir, toma el resultado del modelo del diseño para generar el código final.

Este artefacto describe cómo se implementan los componentes, congregándolos en subsistemas organizados en capas y jerarquías, además señala las dependencias entre estos.

4.2.1.1 Diagrama de despliegue.

El modelo de despliegue describe como una aplicación se despliega a través de una infraestructura. La intención del modelo de despliegue no es para describir la infraestructura, es el camino en el cual los componentes específicos deben corresponder a una aplicación que despliega a través de él.

El modelo de despliegue muestra la configuración (relaciones físicas) de los nodos que participan en la ejecución y de los componentes hardware y software que residen en ellos. Un nodo es un elemento físico que existe en tiempo de ejecución y representa un recurso computacional, que generalmente tiene algo de memoria y, a menudo, capacidad de almacenamiento. Son los elementos donde se ejecutan los componentes.

El diagrama de distribución, como también se conoce contiene: nodos (servidores o procesadores y dispositivos) y relaciones de dependencia y asociación.

Diagrama de despliegue.

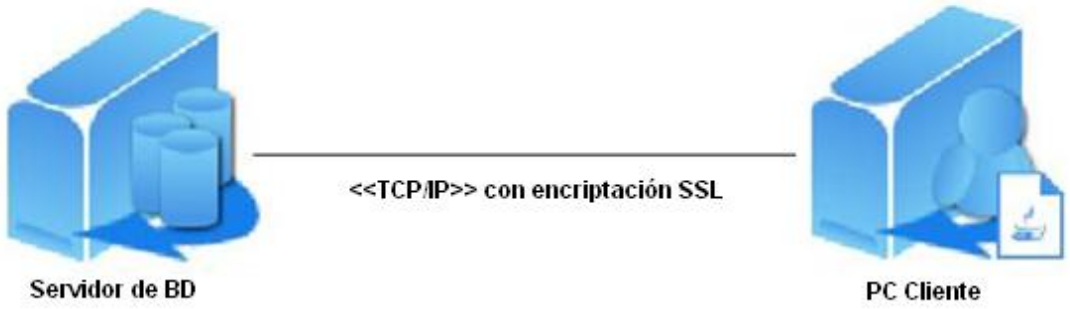


Figura 4.1: Diagrama de despliegue.

4.2.1.1.1 TCP/IP.

Por sus siglas en inglés Transmission Control Protocol/Internet Protocol, Protocolo de Control de Transmisión/Protocolo Internet es un conjunto de casi 100 programas de comunicación de datos usados para organizar computadoras en redes. Norma de comunicación en Internet, compuesta por dos partes: el TCP/IP. El IP desarma los envíos en paquetes y los rutea, mientras que el TCP se encarga de la seguridad de la conexión, comprueba que los datos lleguen todos, completos, y que compongan finalmente el envío original.

El TCP / IP es la base del Internet que sirve para enlazar computadoras que utilizan diferentes sistemas operativos, incluyendo PC, minicomputadoras y computadoras centrales sobre redes de área local y área extensa.

Como funciona TCP/IP.

Una red TCP/IP transfiere datos mediante el ensamblaje de bloques de datos en paquetes, cada paquete comienza con una cabecera que contiene información de control; tal como la dirección del destino, seguida de los datos. Cuando se envía un archivo por la red TCP/IP, su contenido se envía utilizando una serie de paquetes diferentes. El IP, un protocolo de la capa de red, permite a las aplicaciones ejecutarse transparentemente sobre redes interconectadas. Cuando se utiliza IP, no es necesario conocer que hardware se utiliza, por tanto ésta corre en una red de área local.

TCP, un protocolo de la capa de transporte, asegura que los datos sean entregados, que lo que se recibe, sea lo que se pretendía enviar y que los paquetes que sean recibidos en el orden en que fueron enviados. TCP terminará una conexión si ocurre un error que haga la transmisión fiable imposible.

4.2.1.1.2 SSL.

Para desplegar el sistema utilizamos en la conexión del servidor de base de datos y la PC cliente el protocolo SSL (Secure Sockets Layer, en español, Protocolo de Capa de Conexión Segura), para garantizar la seguridad de los datos, ya que es un protocolo criptográfico que proporciona comunicación segura por la red.

SSL proporciona autenticación y privacidad de la información entre extremos mediante el uso de la criptografía. Habitualmente, solo el servidor es autenticado (es decir, se garantiza su identidad) mientras que el cliente se mantiene sin autenticar; la autenticación mutua requiere un despliegue de infraestructura de claves públicas para los clientes. Los protocolos permiten a las aplicaciones comunicarse de una forma diseñada para prevenir escuchas (eavesdropping), la falsificación de la identidad del remitente (phishing), mantener la integridad del mensaje. Proporciona cifrado de datos, autenticación de servidores, integridad de mensajes y autenticación de cliente para conexiones TCP/IP.

SSL se ejecuta en una capa entre los protocolos de aplicación como HTTP[61], SMTP[62], NNTP[63] y sobre el protocolo de transporte TCP[64], que forma parte de la familia de protocolos TCP/IP. Aunque pueda proporcionar seguridad a cualquier protocolo que use conexiones de confianza (tal como TCP).

Por la seguridad del sistema a desplegar y integridad de los datos es que se decide usar el protocolo SSL para estableces conexiones seguras.

4.2.1.1.3 Requerimientos.

Se requiere que el servidor de base de datos tenga las características siguientes: soporte grande volúmenes de datos y velocidad de procesamiento, tiempo de respuesta rápido en accesos concurrentes, tecnología libre y tenga instalado PostgreSQL versión 8.3 y la/s PC clientes soporte Java (máquina virtual de Java) instalado, Versión 6.0 o superior. Ambos pueden estar sobre cualquier plataforma (Linux o Windows).

4.2.1.2 Diagrama de componentes.

El diagrama de componente ilustra los componentes del software que serán usados para contribuir el sistema .Estos pueden ser construidos para el modelo de clase y escritos para satisfacer los requisitos del nuevo sistema. Un componente puede ser siempre considerado como una unidad autónoma dentro de un sistema o subsistema. UML define cinco estereotipos estándar que se aplican a los componentes: ejecutable, biblioteca, tabla, archivo y documento. Los distintos componentes pueden agruparse en

[61] HTTP: Por sus siglas en inglés HyperText Transfer Protocol, Protocolo de transferencia de hipertexto, es el protocolo usado en cada transacción de la Web

[62] SMTP: Por sus siglas en inglés Simple Mail Transfer Protocol, Protocolo Simple de Transferencia de Correo, es el protocolo estándar de Internet para intercambiar mensajes de e-mail (correo).

[63] NNTP: Por sus siglas en inglés Network News Transport Protocol, Protocolo para la transferencia de noticias en red, es un protocolo cread para la lectura y publicación de artículos de noticias.

[64] TCP: Por sus siglas en inglés Transmission Control Protocol, Protocolo de Control de Transmisión, es un conjunto de protocolos de comunicación que se encargan de la seguridad y la integridad de los paquetes de datos que viajan por Internet. Complemento del IP en el TCP/IP.

paquetes según un criterio lógico y con vistas a simplificar la implementación. Son paquetes estereotipados en subsistemas. Estos organizan la vista de realización de un sistema y pueden contener componentes y otros subsistemas.

La aplicación contará con cuatro paquetes de implementación, un subsistema visual compuesto por tres paquetes y además dos componentes; todo esto encapsulado dentro del subsistema Código, que compila el componente Tesis.jar y que este a su vez usa el componente db.properties y el paquete Lib, que contiene las librerías usadas para realizar la aplicación.

Diagrama de componentes en subsistemas.

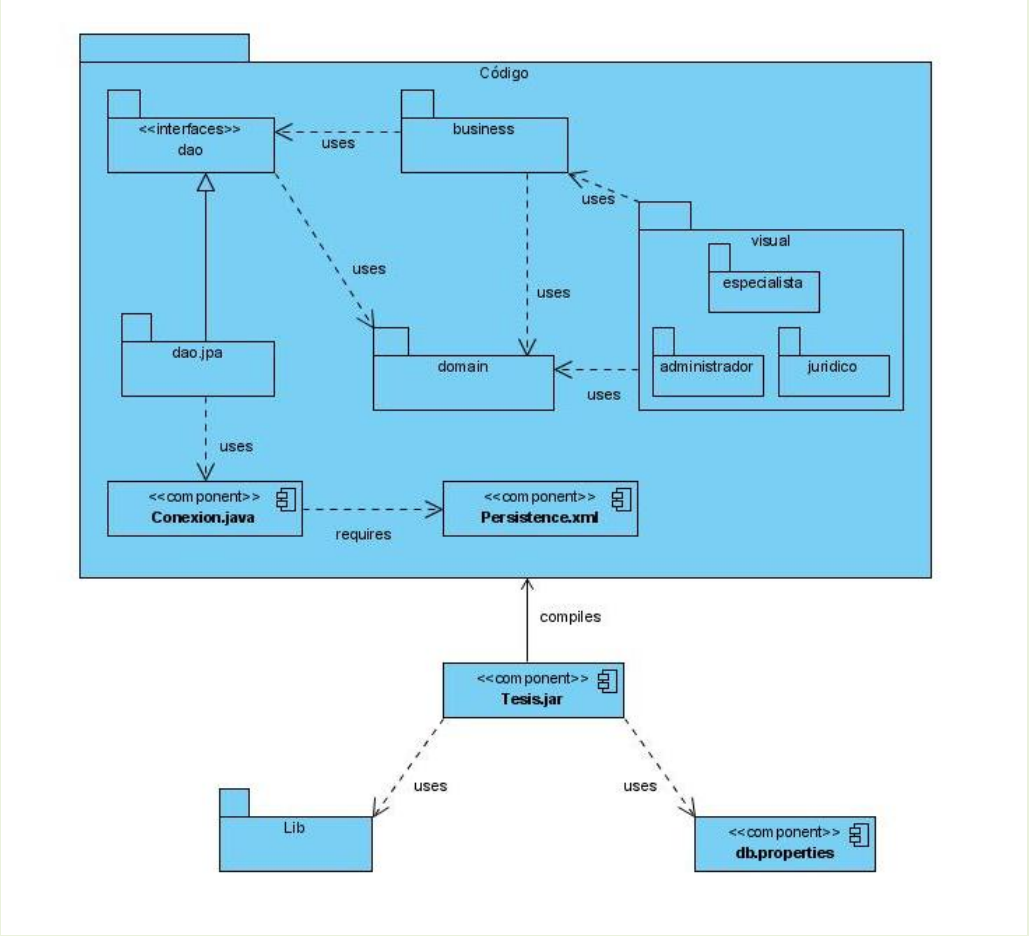


Figura 4.2: Diagrama de componentes en subsistemas.

4.3 Prueba.

Una de las últimas fases del ciclo de vida del desarrollo de un software es el flujo de trabajo de pruebas, dicha actividad se lleva a cabo cuando se ha obtenido un resultado después de la implementación y está encaminada a encontrar errores en el software.

Se le realizan pruebas a la documentación, a un módulo de la aplicación, a toda la aplicación etc, en dependencia de lo que se quiera probar se traza la estrategia, se escoge el nivel, el tipo y el método de prueba a utilizar. Para lograr que estas tengan éxito n es necesario planificarlas y tener claro el objetivo de las mismas.

Las pruebas se pueden hacer a distintos niveles, tales como unidad, integración, sistema y aceptación. La prueba de caja negra es una prueba de sistema y se centran en los requisitos funcionales del software, permitiendo obtener un conjunto de condiciones de entrada que incluyan completamente los requisitos funcionales del programa.

La etapa de prueba es tan o más importante que todas las realizadas hasta el momento puesto que en ella se refleja la calidad con que ha sido llevada a cabo la proyección del sistema.

4.3.1 Pruebas de caja negra.

Se refiere a las pruebas que se llevan a cabo sobre la interfaz del software, por lo que los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta, así como que la integridad de la información externa se mantiene. Esta prueba examina algunos aspectos del modelo fundamentalmente del sistema sin tener mucho en cuenta la estructura interna del software. Estas pruebas son conocidas también como pruebas de comportamiento.

Permite encontrar:

- Funciones incorrectas o ausentes.
- Errores de interfaz.
- Errores de estructuras de datos o en accesos a las Bases de Datos externas.
- Errores de rendimiento.
- Errores de inicialización y terminación.

4.3.1.1 Casos de prueba.

Los casos de pruebas no son más que un conjunto de entradas con datos de prueba, unas condiciones de ejecución, y unos resultados esperados con el propósito de identificar y comunicar las condiciones que se llevarán a cabo en la prueba. Los casos de pruebas son necesarios para verificar la aplicación exitosa y aceptable de los requisitos del producto (que se especifican en los casos de uso).

Los casos de pruebas serían entonces un conjunto de entradas con datos de prueba, unas condiciones de ejecución, y unos resultados esperados cuyo propósito es identificar y comunicar las condiciones que se llevarán a cabo en la prueba. Los casos de la prueba son necesarios para verificar la aplicación exitosa y aceptable de los requisitos del producto (casos de uso).

Deben verificar:

- Si el producto satisface los requerimientos del usuario, tal y como se describe en las especificaciones de los requerimientos.
- Si el producto se comporta como se desea, tal como se describe en las especificaciones funcionales del diseño.

Para la realización de los casos de prueba que a continuación se describen exceptuando el Caso de Uso “Autenticar Usuario” la condición necesaria para que todos se cumplan es que el usuario registrado que realizará las acciones tiene que estar autenticado con el rol de Dpto. de Control, de lo contrario no podrá acceder a las opciones que son probadas.

Tabla 4.1: Caso de Prueba –Caso de Uso "Gestionar Usuario".

Entrada	Resultados	Condiciones
El Dpto. de Control selecciona la opción "Registrar Usuario".	El sistema muestra una interfaz con los datos requeridos para registrar un nuevo usuario.	
El Dpto. de Control introduce los datos del usuario dejando alguno de los campos vacios, o simplemente los deja todos vacios.	El sistema muestra un mensaje de alerta informando que debe introducir los datos faltantes. Ejemplo: "Todos los campos son obligatorios, por favor rectifique"	No se registren valores vacios en los campos de entrada de datos para registrar un nuevo usuario.
El Dpto. de Control introduce los siguientes datos: Usuario: "alberto". Nombres: "Alberto". Apellidos: "Luis Camargo". Cargo: "Especialista". DIP: "Docencia". Contraseña: "••••••". (123456) Repetir Contraseña: "••••••".(123456)	El sistema registra el nuevo usuario, confirmando la operación. Ejemplo: "El Usuario: Alberto se ha insertado correctamente".	No exista el usuario en la aplicación.
El Dpto. de Control introduce los siguientes datos: Usuario: "alberto". Nombres: "Juan". Apellidos: "Hernández Díaz". Cargo: "Directivo". DIP: "Residencia". Contraseña: "••••••". (123456) Repetir Contraseña: "••••••".(123456)	El sistema le informa que el usuario que pretende registrar ya está registrado, mostrando un mensaje. Ejemplo: "Ya se encuentra registrado otra persona con ese identificador, seleccione otro".	Existe el usuario en la aplicación.
El Dpto. de Control introduce los	El sistema le notifica al usuario	Las contraseñas son

datos de un usuario a registrar, introduciendo las contraseñas diferentes. Ejemplo: Usuario: “juan”. Nombres: “Juan”. Apellidos: “Hernández Díaz”. Cargo: “Directivo”. DIP: “Residencia”. Contraseña: “••••••”. (123456) Repetir Contraseña: “••••••”.(123457)	que las contraseñas introducidas en el campo Contraseña y Repetir Contraseña. Ejemplo: “Las contraseñas deben coincidir”.	diferentes.
El Dpto. de Control selecciona la opción “Modificar Usuario” pero no ha marcado el usuario que desea actualizar.	El sistema muestra un mensaje para informarle que debe tener seleccionado un usuario. Ejemplo: “Debe seleccionar un usuario del listado antes de realizar esta operación”.	El usuario a modificar debe estar seleccionado.
El Dpto. de Control marca el usuario a modificar y selecciona la opción “Modificar Usuario”.	El sistema muestra una interfaz con los datos del usuario, reflejando que el identificador del usuario (Usuario) no puede ser cambiado.	
El Dpto. de Control selecciona la opción “Activar/Desactivar Usuario” pero no ha marcado el usuario que desea actualizar.	El sistema muestra un mensaje para informarle que debe tener seleccionado un usuario. Ejemplo: “Debe seleccionar un usuario del listado antes de realizar esta operación”.	El usuario a activar/desactivar debe estar seleccionado.
El Dpto. de Control marca el usuario a desactivar (alberto) y selecciona la opción “Activar/Desactivar Usuario”.	El sistema lanza una pregunta para garantizar que no se realice una operación errada. Ejemplo:	El usuario tenga estado activo.

	"¿Está seguro que desea que el usuario alberto pase a estado Inactivo?" (Si, No, Cancelar).	
El Dpto. de Control presiona en el botón Si de la pregunta lanzada por el sistema anteriormente.	El sistema le informa al Dpto. de Control que el usuario pasó a estado inactivo. Ejemplo: "El usuario alberto se ha desactivado".	Sea presionado el botón Si.
El Dpto. de Control marca el usuario a activar (alberto) y selecciona la opción "Activar/Desactivar Usuario".	El sistema lanza una pregunta para garantizar que no se realice una operación errada. Ejemplo: "¿Está seguro que desea que el usuario alberto pase a estado Activo?" (Sí, No, Cancelar).	El usuario tenga estado inactivo.
El Dpto. de Control presiona en el botón Si de la pregunta lanzada por el sistema anteriormente.	El sistema le informa al Dpto. de Control que el usuario pasó a estado activo. Ejemplo: "El usuario alberto se ha activado".	Sea presionado el botón Si.

Tabla 4.2: Modelo de Prueba –Caso de Uso "Autenticar Usuario".

Entrada	Resultados	Condiciones
El usuario desea acceder al sistema, por lo que introduce los datos: Usuario: "alberto". Contraseña: "••••••" (123456).	El sistema le muestra la interfaz principal en dependencia del rol que tenga este usuario.	Exista conexión entre la PC cliente y el Servidor de BD.
El usuario desea acceder al sistema, por lo que introduce los datos: Usuario: "alberto".	El sistema le notifica a usuario que hay algún problema con la conexión a la BD. Ejemplo: "Error	No hay conexión con la BD.

Contraseña: "••••••" (123456).	de conexión".	
El usuario desea acceder al sistema, por lo que introduce los datos: Usuario: "alberto". Contraseña: "•••••" (12345).	El sistema le notifica al usuario que la contraseña es incorrecta. Ejemplo: "Contraseña no válida para el usuario".	La contraseña para el acceso al sistema está mal.
El usuario desea acceder al sistema, por lo que introduce los datos: Usuario: "luis". Contraseña: "••••••••" (12345678).	El sistema le notifica al usuario que el usuario con el que pretende acceder no existe en el sistema. Ejemplo: "Este usuario no se encuentra registrado en el sistema".	El usuario no esté registrado en la aplicación.
El usuario desea acceder al sistema, pero no sabe que su usuario está en estado desactivado, introduce los datos: Usuario: "alberto". Contraseña: "••••••" (123456).	El sistema le informa al usuario que está en estado desactivado y que cualquier duda vea al Dpto. de Control. Ejemplo: "Cuenta deshabilitada, consulte al Departamento de Control para más detalles".	El usuario esté en estado inactivo.

Tabla 4.3: Modelo de Prueba –Caso de Uso "Gestionar Objeto de Obra".

Entrada	Resultados	Condiciones
El Dpto. de Control selecciona la opción "Registrar Objeto de Obra".	El sistema muestra una interfaz con los datos requeridos para registrar un nuevo objeto de obra o centro de costo.	
El Dpto. de Control introduce los siguientes datos: Código Centro Costo: "12333".	El sistema registra el centro de costo y le muestra un mensaje para preguntarle al Dpto. de	No exista un objeto de obra registrado con ese mismo código.

Nombre: “Manzana 500”. Presupuesto MN: “15000”. Presupuesto CUC: “1500”.	Control si desea introducir otra obra nueva. Ejemplo: “¿Desea insertar otra obra?” (Si, No, Cancelar).	
El Dpto. de Control introduce los siguientes datos: Código Centro Costo: “12333”. Nombre: “Manzana 501”. Presupuesto MN: “33000”. Presupuesto CUC: “4500”.	El sistema muestra un mensaje de error notificándole al Dpto. de Control que ya existe una obra registrada en el sistema con ese código. Ejemplo: “Actualmente existe otro Centro de Costo con ese código, por favor rectifique”.	Ya este registrado un centro de costo con el mismo código.
El Dpto. de Control decide actualizar una obra y presiona el botón Cargar.	El sistema le muestra un mensaje para sea introducido el código de la obra que se desea cargar. Ejemplo: “Debe introducir el Código”.	No se haya introducido el código de la obra a cargar.
El Dpto. de Control desea actualizar una obra e introduce los siguientes datos: Código Centro Costo: “2002”.	El sistema carga los datos de esta obra permitiendo que sean modificados los mismos.	La obra con ese código se encuentra registrada en el sistema.
El Dpto. de Control desea actualizar una obra e introduce los siguientes datos: Código Centro Costo: “1001”.	El sistema le notifica al Dpto. de Control que no existe ninguna obra con ese código. Ejemplo: “El Código no se corresponde con ninguna Obra registrada en la Base de Datos…”.	La obra con ese código no está registrada en el sistema.
El Dpto. de Control luego de haber realizado las modificaciones requeridas presiona el botón Aceptar.	El sistema le pregunta si desea hacer algún otro cambio en la obra. Ejemplo: “¿Desea realizar otra modificación a la Obra?” (Si,	Se haya cargado una obra.

	No, Cancelar).	
El Dpto. de Control al actualizar los datos de la obra deja un campo vacio.	El sistema le notifica que todos los campos son obligatorios, por lo que no puede dejar ninguno de estos en blanco. Ejemplo: “Todos los campos son Obligatorios”.	Se haya cargado una obra.

Tabla 4.4: Modelo de Prueba –Caso de Uso “Listar Objetos de Obra”.

Entrada	Resultados	Condiciones
El Dpto. de Control selecciona la opción “Listar Objetos de Obra”.	El sistema muestra una interfaz que le permite seleccionar las obras a listar por varios criterios.	
El Dpto. de Control desea obtener el listado de las obras por la DIP a la que pertenece y marca la opción filtrar por DIP, sin especificar la DIP y presiona el botón Listar.	El sistema le informa al usuario que debe seleccionar una DIP antes de marcar la opción. Ejemplo: “Debe seleccionar una DIP primero”.	No se encuentre especificada la DIP por la que desea filtrar.
El Dpto. de Control desea obtener el listado de las obras por el estado en que están y marca la opción filtrar por Estado, sin especificar el estado y presiona el botón Listar.	El sistema le informa al usuario que debe seleccionar un Estado antes de marcar la opción. Ejemplo: “Debe seleccionar un estado primero”.	No se encuentre especificado el Estado por la que desea filtrar.
El Dpto. de Control desea obtener el listado de las obras por la Entidad de contratación y marca la opción filtrar Entidad, sin especificar el código de la Entidad y presiona el botón Listar.	El sistema le informa al usuario que debe introducir el código de la Entidad una antes de marcar la opción. Ejemplo: “Debe introducir el Código de la Entidad de primero”.	No se encuentre especificado el código de la Entidad por la que desea filtrar.

El Dpto. de Control desea obtener el listado general de todas las obras que hay registradas en el sistema, presionando el botón Listado General.	El sistema lista todas las obras que hay registradas, mostrando el nombre, código y estado en el que se encuentra.	
El Dpto. de Control desea obtener el listado de las obras por la DIP a la que pertenece, selecciona la DIP y luego marca la opción Filtrar por DIP. Ejemplo: Selecciona la DIP: Preparación de Obras y presiona el botón Listar.	El sistema lista todas las obras que pertenecen a la DIP especificada, mostrando el nombre, código y estado en el que se encuentra.	
El Dpto. de Control desea obtener el listado de las obras por el Estado en que se encuentra, selecciona el Estado y luego marca la opción Filtrar por Estado. Ejemplo: Selecciona el Estado: Abierto y presiona el botón Listar.	El sistema lista todas las obras cuyo Estado es Abierto, mostrando el nombre, código y estado en el que se encuentra.	
El Dpto. de Control desea obtener el listado de las obras por el Código de la Entidad, introduce el código y luego marca la opción Filtrar por Entidad. Ejemplo: Introduce el código de Entidad: 201 y presiona el botón Listar.	El sistema lista todas las obras que su Entidad de Contratación corresponde con el código especificado, mostrando el nombre, código y estado en el que se encuentra.	
El Dpto. de Control presiona el botón Listar sin antes seleccionar una opción por la que se desee filtrar.	El sistema le notifica que debe seleccionar al menos una opción por la que se desee filtrar. Ejemplo: “Debe especificar al	No se ha selecciona ninguna opción por la que se desea filtrar.

	menos un criterio de búsqueda.”.	

4.4 Conclusiones.

En este capítulo se realizó la modelación del despliegue que tendrán los nodos donde será distribuida la aplicación, especificando los protocolos de comunicación que existirán entre los mismo, mediante el diagrama de despliegue. Se presentaron las dependencias entre los componentes de software a través del diagrama de componentes. Además se realizó la descripción de los casos de prueba de caja negra para los casos de uso del sistema.

CONCLUSIONES

Con la realización de este trabajo se logró dar cumplimiento al objetivo trazado en el proceso de investigación y desarrollo. La aplicación se desarrolló siguiendo la metodología de desarrollo RUP, utilizándose para su representación UML en la modelación de todas las fases del proyecto y como herramienta CASE Visual Paradigm. Se modelaron los flujos de trabajo propuestos por RUP: Modelado del Negocio, Gestión de Requisitos, Análisis y Diseño, Implementación, Prueba y Despliegue.

Luego de un estudio de los sistemas contables existentes se demostró que estos no cumplen con los requerimientos exigidos por la UPI “Proyecto Futuro” para llevar el control del proceso inversionista.

Al realizar un análisis detallado del Modelo de Negocio y el Sistema se obtuvieron 30 requisitos funcionales como resultado del levantamiento de requisitos de software, que establecen los servicios que el producto debe cumplir.

Se implementaron dos ciclos de desarrollo de la aplicación, obteniendo una herramienta libre y multiplataforma que permitirá a la UPI “Proyecto Futuro” y a todas las DIP controlar el proceso inversionista.

RECOMENDACIONES

Teniendo como base los resultados de esta investigación y la experiencia adquirida durante el desarrollo de la misma, se proponen las siguientes recomendaciones:

- Una vez terminada la implementación, completar el despliegue de la primera versión del sistema en todas las unidades funcionales de la UPI “Proyecto Futuro”, con el objetivo de poner a prueba el sistema durante un período de tiempo a fin de comprobar su correcto funcionamiento y agregar, modificar o corregir cualquier requisito o necesidad del cliente.
- Continuar la investigación en la UPI “Proyecto Futuro”, para el futuro desarrollo de la aplicación con el objetivo de incorporarle nuevas funcionalidades.
- Capacitar nuevos programadores para las próximas etapas de desarrollo del sistema, apoyándose en la experiencia de los actuales.
- Realizar un estudio del negocio y levantamiento de requisitos en otras entidades cubanas destinadas a la gestión de procesos inversionistas, presupuestadas o no, empresas de servicios ingenieros y otras similares, con el objetivo de que utilicen este sistema para el control de las obras.
- Proponer la inclusión del sistema en el ERP cubano, como un módulo para la gestión y control de las inversiones.

BIBLIOGRAFÍA

1. C, Larman. UML y Patrones. Año 2000.
2. Desarrollo de aplicaciones multiplataforma con NetBeans IDE. Disponible en: http://www.sun.com/emrkt/innercircle/newsletter/latam/0207latam_feature.html
3. El Lenguaje Unificado de Modelado (UML). Disponible en: http://www.disca.upv.es/enheror/pdf/ActaUML.PDF
4. Fundamentos de Ingeniería de Software. Disponible en: http://www.inf.utfsm.cl/~visconti/ili236/Documentos/08-Patrones.pdf
5. Jacobson, I. Booch, G. Rumbaugh, J. El Proceso Unificado de Desarrollo de Software. Pearson Addisson-Wesley. Año 2000.
6. LaWebDelProgramador. [En línea] http://www.lawebdelprogramador.com
7. Lenguajes de Programación mas populares, Abril 18th, 2007. [en línea]. Disponible en Web: http://www.kabytes.com/actualidad/lenguajes-de-programacion- mas-populares/
8. Lenguajes de Programación. Programación Java. Disponible en http://www.lenguajes-de-programacion.com/programacion-java.shtml
9. Letelier, P; Penadés, C. Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP). 2006. [en línea]. Disponible en Web: http://www.willydev.net/descargas/masyxp.pdf
10. Mendoza, M. Metodologías De Desarrollo De Software. 2004. [en línea]. Disponible en Web: http://www.informatizate.net/articulos/pdfs/metodologias_de_desarrollo_de_software_07062004.pdf
11. Peláes, J. A. Metodología para el Desarrollo de Software. 2004. [en línea] Disponible en Web: http://www.lcc.uma.es/~jignacio/index_archivos/TEMA4.pdf
12. Pimentel, L. Pérez, I., ArBaWeb: Arquitectura Base sobre la Web. Trabajo para optar por el título de ingeniero informático, Universidad de las Ciencias Informáticas, junio 2007.

13. PostgreSQL Práctico. Disponible en: http://www.sobl.org/traducciones/practical-postgres/node12.html

14. ¿Qué significa y qué ventajas aporta la arquitectura en tres capas? http://www.solmicro.es/inicio.php?ID_CATEGORIA=25

15. Rumbaugh, y otros. 2000. *El Lenguaje Unificado de Modelado*. Año 2000.

16. Teoría Organizativa y Teoría Administrativa aplicadas a la educación superior. Material documental de apoyo. Universidad de Carabobo, área de estudios de Postgrado PEDES. Prof. Cecilia Guerra. Valencia, Año 2000.

17. Terry, George R., Principios Administrativos. Compañía Editorial Continental, S.A. México, Año 1977.

18. UCI. Teleformación. [En línea] http://Teleformacion.uci.cu

19. Visual Paradigm for UML. Disponible en: http://www.versioncero.com/?pg=34

20. Visual Paradigm. 2007. [en línea]. Disponible en Web: http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_(Iglesia_Anglicana)_%5BMac_OS_X_cuenta_14717_p/

REFERENCIA BIBLIOGRÁFICA

Assets. (s.f.). *ASSETS Sistema de Gestión Integral*. Recuperado el 11 de Diciembre de 2008, de ASSETS Sistema de Gestión Integral: http://assets.co.cu/assets.asp

Booch, G. (1996). *Análisis y diseño orientado a objetos con aplicaciones*. (2 ed.). (J. M. Cerva Lovelle, Trad.) Addison-Wesley.

Burnette, E. (2005). *Eclipse IDE Pocket Guide* (ilustrado ed.). (O'Reilly, Ed.)

Flanagan, D. (1999). *Java en pocas palabras* (2da ed., Vol. 1). (McGRAW-HILL, Ed.) México.

Iglesias Morell, A. (2008). *Bibliografía básica, 1ra Edición de la Maestría de Administración Pública del Consejo de Estado*. Ciudad de la Habana.

Inversiones GAMMA S.A. (s.f.). *GAMMA Tecnología y Medio Ambiente*. Recuperado el 11 de Diciembre de 2008, de GAMMA Tecnología y Medio Ambiente: http://www.gamma.com.cu/servicio.php?serv=24

JMS Investments. (s.f.). *JMS Investments*. Recuperado el 11 de Diciembre de 2008, de JMS Investments: http://www.jms-investments.com/services_es.html

Johnson, R., & Foote, B. (1988). *Designing Reusable Classes*.

Johnson, R., & Russo, V. (1991). *Reusing Object-Oriented Designs*.

Kaiser, S. H. (2005). *Software Paradigms*.

Kruchten, P. (2003). *The Rational Unified Process An Introduction*. Third Edition.

Letelier, P., & Penades, M. d. (2006). Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP). *Técnica Administrativa, Buenos Aires , 05* (26).

M Autor Deitel, H., & J Autor Deitel, P. (1995). *Como programar en C/C++*. México: Pearson Educación.

Marioko. (19 de Noviembre de 2008). *javaHispano*. Recuperado el 15 de Diciembre de 2008, de javaHispano: http://www.javahispano.org/contenidos/es/netbeans_6_5_final/?menuId=NEWS

Montejava. (2006). *Montejava, Solucioenes empresariales basadas en web*. Recuperado el 8 de Enero de 2008, de Montejava, Solucioenes empresariales basadas en web: http://www.montejava.es/articulo15.asp

NetBeans. (s.f.). *NetBeans*. Recuperado el 6 de Diciembre de 2008, de NetBeans: http://www.netbeans.org/index_es.html

NetBeans.org . (20 de Noviembre de 2008). *Soluciones en Ciencias de la Computación que Perfeccionan Tu Empresa* . Recuperado el 15 de Diciembre de 2008, de Soluciones en Ciencias de la Computación que Perfeccionan Tu Empresa : http://www.linperial.com/news/details/?newsid=128

Pérez López, J., & Iglesias Fernández, C. A. (2003). *Extensión de la herramienta Rational Rose para el desarrollo del modelo de la experiencia según la metodología orientada a agentes mas-commonkads*. (J. Pérez López, Ed.)

Popkin Software and Systems. (s.f.). *La Web del Programador*. Recuperado el 6 de Enero de 2009, de La Web del Programador: http://es.tldp.org/Tutoriales/doc-modelado-sistemas-UML/doc-modelado-sistemas-uml.pdf

PostgreSQL. (s.f.). *PostgreSQL*. Recuperado el 10 de diciembre de 2008, de PostgreSQL: http://www.postgresql.org/about/

Rumbaugh, J., Booch, G., Jacobson, I., Sanjuán Martínez, Ó., Castán Rodríguez, H., Fuente Alarcón, M., y otros. (2007). *El lenguaje unificado de modelado: Manual de referencia* (2 ed.). Addison-Wesley.

Soto López, N. M., & Saborit Ramírez, Y. (2004). *Sistema de catalogación y recuperación de recursos de información Hubble*. Ciudad de la Habana, Cuba.

TiendaLinux.com. (26 de Diciembre de 2005). Obtenido de TiendaLinux.com: http://soporte.tiendalinux.com/portal/Portfolio/postgresql_ventajas_html

Visual Paradigm. (2008). *Visual Paradigm*. Recuperado el 12 de Diciembre de 2008, de Visual Paradigm.

GLOSARIO DE TÉRMINOS

.NET: Plataforma de desarrollo de software con énfasis en transparencia de redes, con independencia de plataforma de hardware y que permita un rápido desarrollo de aplicaciones.

4+1 vistas de la arquitectura: Vista de casos de uso, vista lógica, vista de procesos, vista de componentes y vista de despliegue.

Ajax: Acrónimo de Asynchronous JavaScript And XML (JavaScript asíncrono y XML),

Ant: Herramienta usada en programación para la realización de tareas mecánicas y repetitivas, normalmente durante la fase de compilación y construcción.

Assets: Sistema de Gestión Integral estándar y parametrizado que permite el control de los procesos de Compras, Ventas, Producción, Taller, Inventario, Finanzas, Contabilidad, Presupuesto, Activos Fijos, Útiles y Herramientas y Recursos Humanos desde una perspectiva económica.

BSD: Licencia de software otorgada principalmente para los sistemas BSD (Berkeley Software Distribution, en español, Distribución de Software Berkeley). Pertenece al grupo de licencias de software Libre.

Bytecode: Código intermedio más abstracto que el código máquina. Habitualmente es tratado como un fichero binario que contiene un programa ejecutable similar a un módulo objeto, que es un fichero binario producido por el compilador cuyo contenido es el código objeto o código máquina.

C, C++, Java, Python, Ruby: Lenguajes de programación.

Contrato: Documento en el que se establece de los trabajos a realizar y el valor de los mismos. Así como el tiempo en que serán ejecutados (cronograma de ejecuciones).

Eclipce: Entorno de desarrollo integrado de código abierto multiplataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido".

EJB: Los Enterprise JavaBeans son una de las API que forman parte del estándar de construcción de aplicaciones empresariales J2EE de Sun Microsystems.

Excel: Programa de Microsoft, el cual consiste en una hoja de cálculo, utilizada para realizar fórmulas matemáticas y cálculos aritméticos exhaustivos.

Glassfish 3 Prelude: Servidor modular basado en Java.

Glue code: en términos de programación, es el código que no realiza ninguna funcionalidad para satisfacer algún requerimiento del programa en cuestión.

GNU/GPL: Acrónimo de Licencia Pública General de GNU o más conocida por su nombre en inglés GNU General Public License.

Grails: Framework (estructura de soporte) para aplicaciones web de código abierto desarrollado sobre el lenguaje de programación Groovy.

Groovy: Lenguaje de programación orientado a objetos implementado sobre la plataforma Java.

Herramienta CASE: Aplicación informática destinada a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero, se utiliza para la modelación del sistema.

Hibernate: Potente framework de mapeo objeto/relacional y servicio de consultas para Java. Es la solución ORM (Object-Relational Mapping) más popular en el mundo Java.

HQL: Hibernate Query Language, está diseñado como un "mínimo" de extensión orientada a objetos a SQL, proporciona un puente entre el objeto y las bases de datos relacionales.

HTML: Sigla en inglés de HyperText Markup Language (Lenguaje de Marcas de Hipertexto), es el lenguaje de marcado predominante para la construcción de páginas web.

HTTP: Por sus siglas en inglés HyperText Transfer Protocol, Protocolo de transferencia de hipertexto, es el protocolo usado en cada transacción de la Web

Índice FTSE 100: Lo componen los 100 principales valores ("blue chips") de la Bolsa de Londres (London Stock Exchange). Ftse es un acrónimo de Financial Times Stock Exchange. El principal indicador del FTSE 100 es el FTSE 100 Index. El índice fue desarrollado con un nivel base de 1000.

Interbase: Sistema de Administración de Base de Datos Relacionales (RDBMS) desarrollada y comercializada por la compañía Borland Software Corporation.

Inversión alternativa: Tiene como objetivo la inversión en largo plazo.

J2EE: (Java 2 Enterprise Edition). Define un estándar para el desarrollo de aplicaciones empresariales multicapa diseñado por Sun Microsystems. J2EE simplifica las aplicaciones empresariales basándolas en componentes modulares y estandarizados, proveyendo un completo conjunto de servicios a estos componentes, y manejando muchos de las funciones de la aplicación de forma automática, sin necesidad de una programación compleja.

J2SE: Siglas en inglés de Java 2 Standard Edition. Edición Estándar. La versión estándar es la más común y cuenta con todo lo necesario para desarrollos de software y acceso a aplicaciones Java.

Java: Lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los años 90.

JavaScript: Lenguaje de programación interpretado, es decir, que no requiere compilación, utilizado principalmente en páginas web.

JDBC: Es el acrónimo de Java Database Connectivity, un API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java, independientemente del sistema operativo

donde se ejecute o de la base de datos a la cual se accede utilizando el dialecto SQL del modelo de base de datos que se utilice.

JPA: Es la API de persistencia desarrollada para la plataforma Java EE.

Linux: Es un término genérico para referirse a sistemas operativos similares a Unix basados en GNU con el núcleo Linux.

MacOS: Sistema operativo multitarea de Apple, basado en sistemas UNIX.

Microsoft SQL Server: SGBD basado en el lenguaje Transact-SQL.

Microsystems: Empresa informática de Silicon Valley, fabricante de semiconductores y software.

MySGL AB: Compañía de software fundada en 1995, creadora del sistema administrador de bases de datos relacionales MySQL, y una de las más grandes empresas de software libre del mundo.

MySQL: SGBD relacional, multihilo y multiusuario con más de seis millones de instalaciones.

NNTP: Por sus siglas en inglés Network News Transport Protocol, Protocolo para la transferencia de noticias en red, es un protocolo cread para la lectura y publicación de artículos de noticias.

Objeto de obra: Edificio u otra construcción que compone una obra, al que se le reconoce una función diferenciada y límites fí-sicos precisos y que, en función de estos, posee presu-puesto y documentación de proyectos propios, también es llamada centro de costo.

Open source: Código abierto, es el término con el que se conoce al software distribuido y desarrollado libretmente.

Oracle Corporation: Es una de las mayores compañías de software del mundo. Sus productos van desde bases de datos (Oracle) hasta sistemas de gestión.

Oracle: Es una herramienta cliente/servidor que se desarrolla para la gestión de Bases de Datos.

PDF: Sigla en inglés de Portable Document Format (Formato de Documento Portátil), es un formato de almacenamiento de documentos.

PL/SQL: Lenguaje de programación embebido en Oracle y PostgreSQL.

PostgreSQL: SGBD relacional orientada a objetos de software libre, publicado bajo la licencia BSD.

Quick Find: Es un sencillo menú utilitario de la barra del sistema.

Rollback: Es una operación que devuelve a la base de datos a algún estado previo.

SMTP: Por sus siglas en inglés Simple Mail Transfer Protocol, Protocolo Simple de Transferencia de Correo, es el protocolo estándar de Internet para intercambiar mensajes de e-mail (correo).

Software: Es el conjunto de programas e instrucciones asociados a una computadora. La parte intangible que hace funcionar un sistema informático y que puede ser modificada con facilidad.

Solaris: Sistema operativo de tipo Unix desarrollado por Sun Microsystems desde 1992 como sucesor de SunOS.

SSL: Sigla en inglés de Secure Sockets Layer, Protocolo de Capa de Conexión Segura. Proporciona cifrado de datos, autenticación de servidores e integridad de mensajes.

Subversion: Software de sistema de control de versiones.

TCP: Por sus siglas en inglés Transmission Control Protocol, Protocolo de Control de Transmisión, es un conjunto de protocolos de comunicación que se encargan de la seguridad y la integridad de los paquetes de datos que viajan por Internet. Complemento del IP en el TCP/IP.

US: Dólar.

UserStories: Historias de usuario.

Web: Sistema distribuido con mecanismos de hipertexto. Que permite mezclar texto, gráficos y archivos de sonido juntos.

Widget: Objeto cuyo nombre no se sabe o se olvidó; símbolo gráfico de interface que permite interacción entre el usuario y el ordenador (símbolo, Icono, ventana y etcétera.).

Windows: Sistema operativo desarrollado por Microsoft.

XML: Sigla en inglés de Extensible Markup Language (Lenguaje de Marcas Ampliable), es un metalenguaje extensible de etiquetas.

ANEXOS

Anexo 1.

Diagrama de Actividades – Caso de Uso "Aprobar Solicitud de Código de Objeto de Obra".

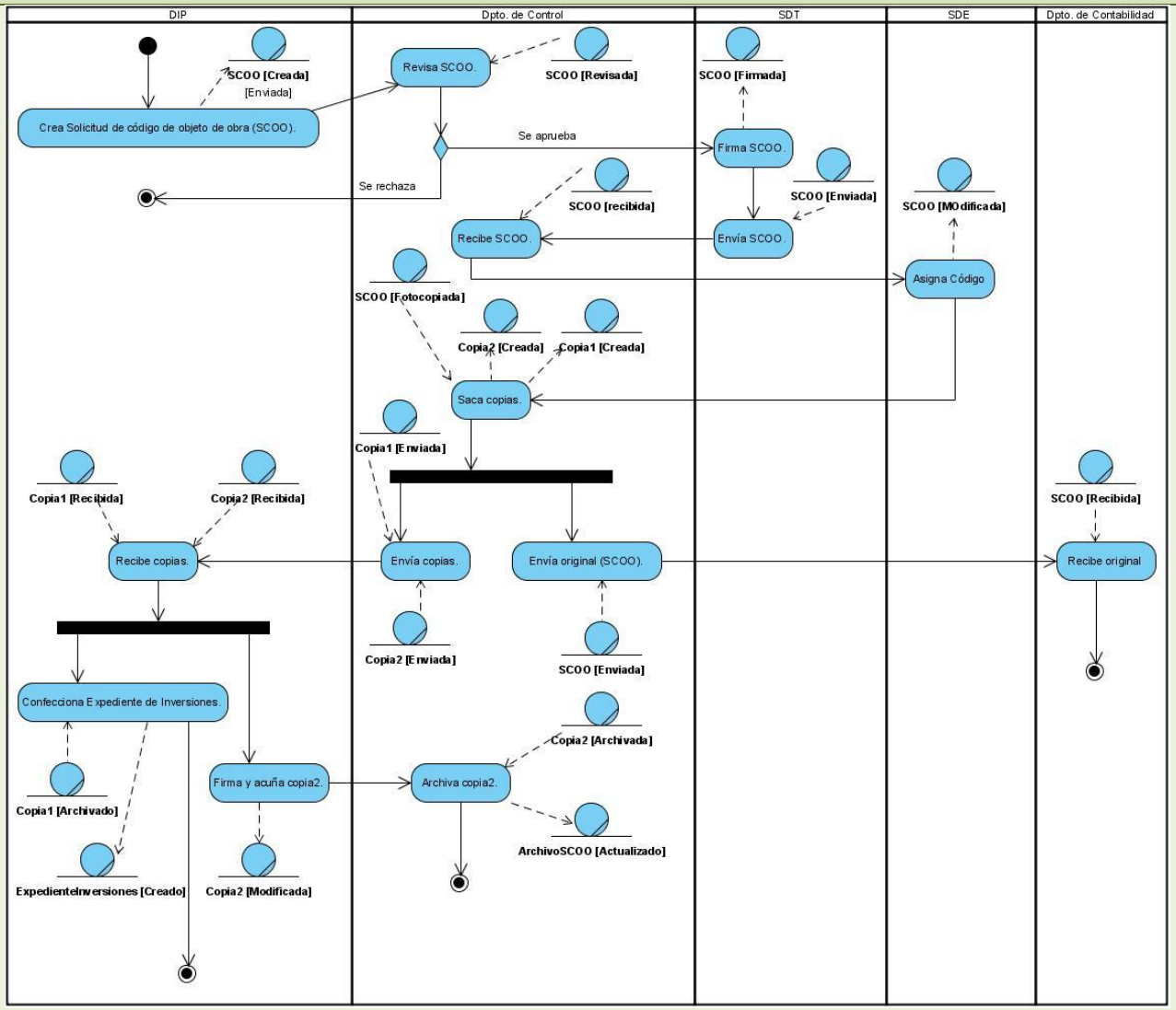


Figura A.1: Diagrama de Actividades – Caso de Uso "Aprobar Solicitud de Código de Objeto de Obra".

Anexo 2.

Diagrama de Clases del Modelo de Objetos – Caso de Uso "Aprobar Solicitud de Código de Objeto de Obra".

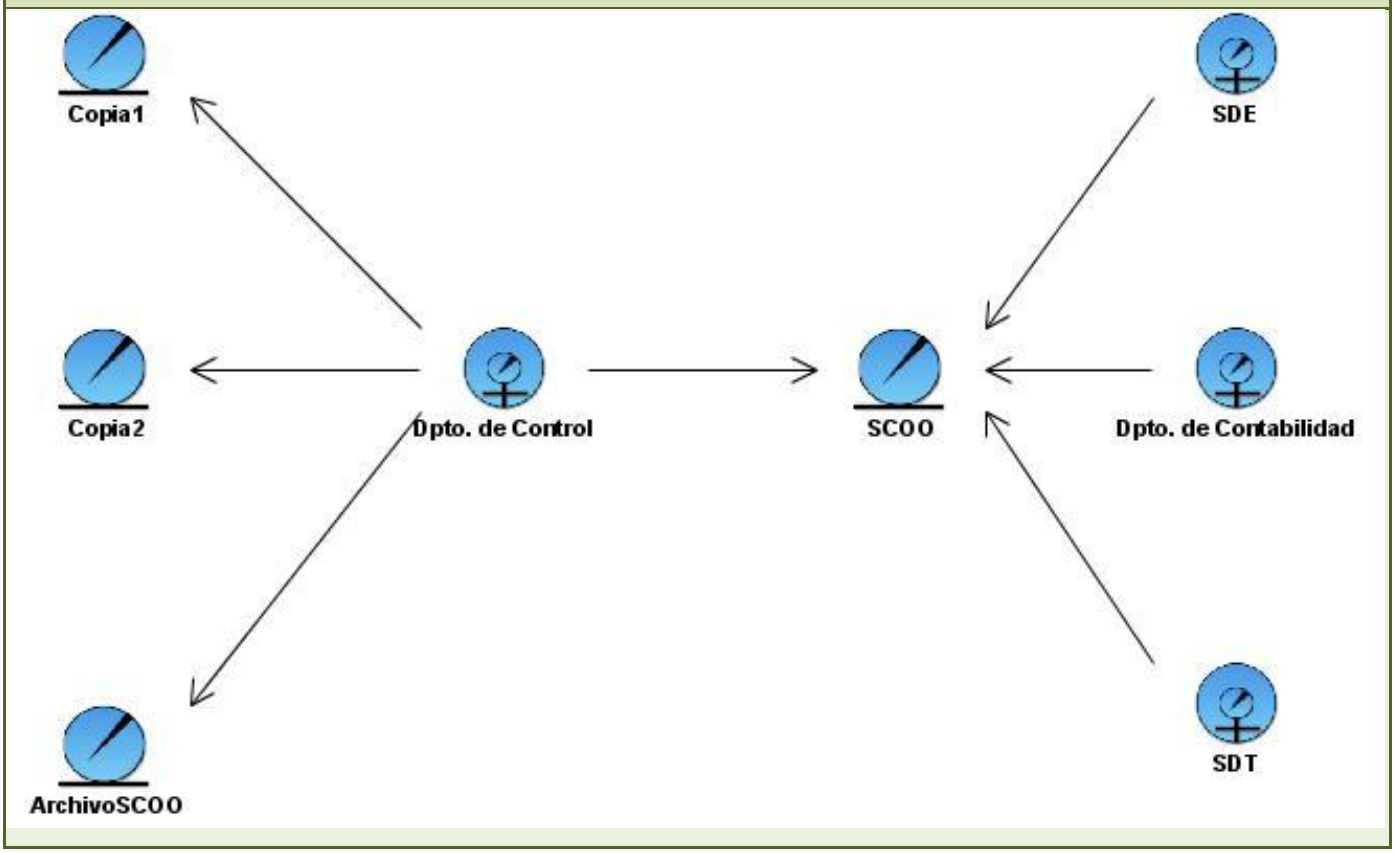


Figura A.2: Diagrama de Clases del Modelo de Objetos – Caso de Uso "Aprobar Solicitud de Código de Objeto de Obra".

Anexo 3.

Diagrama de Actividades – Caso de Uso "Aprobar Factura".

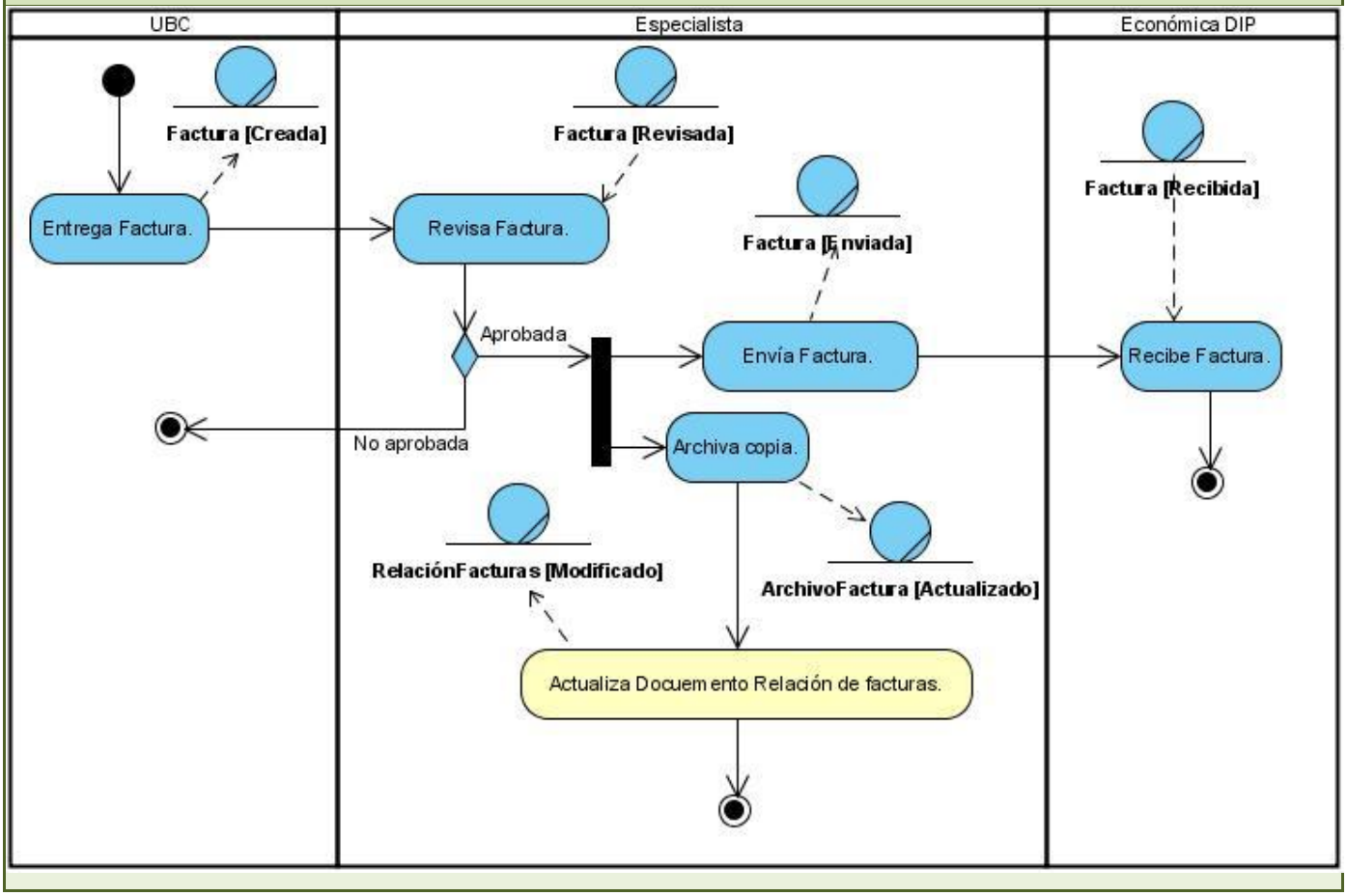


Figura A.3: Diagrama de Actividades – Caso de Uso "Aprobar Factura".

Anexo 4.

Diagrama de Clases del Modelo de Objetos – Caso de Uso "Aprobar Factura".

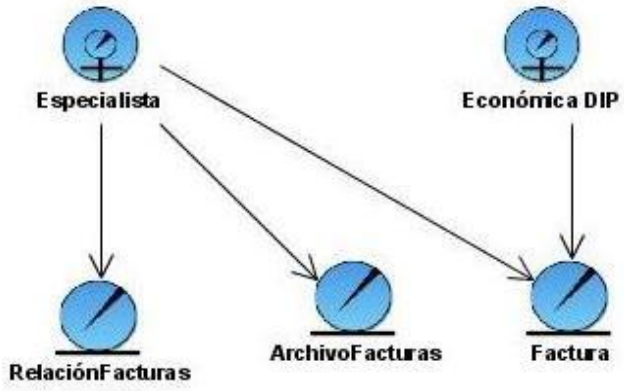


Figura A.4: Diagrama de Clases del Modelo de Objetos – Caso de Uso "Aprobar Factura".

Anexo 5.

Diagrama de Actividades – Caso de Uso "Aprobar Contrato".

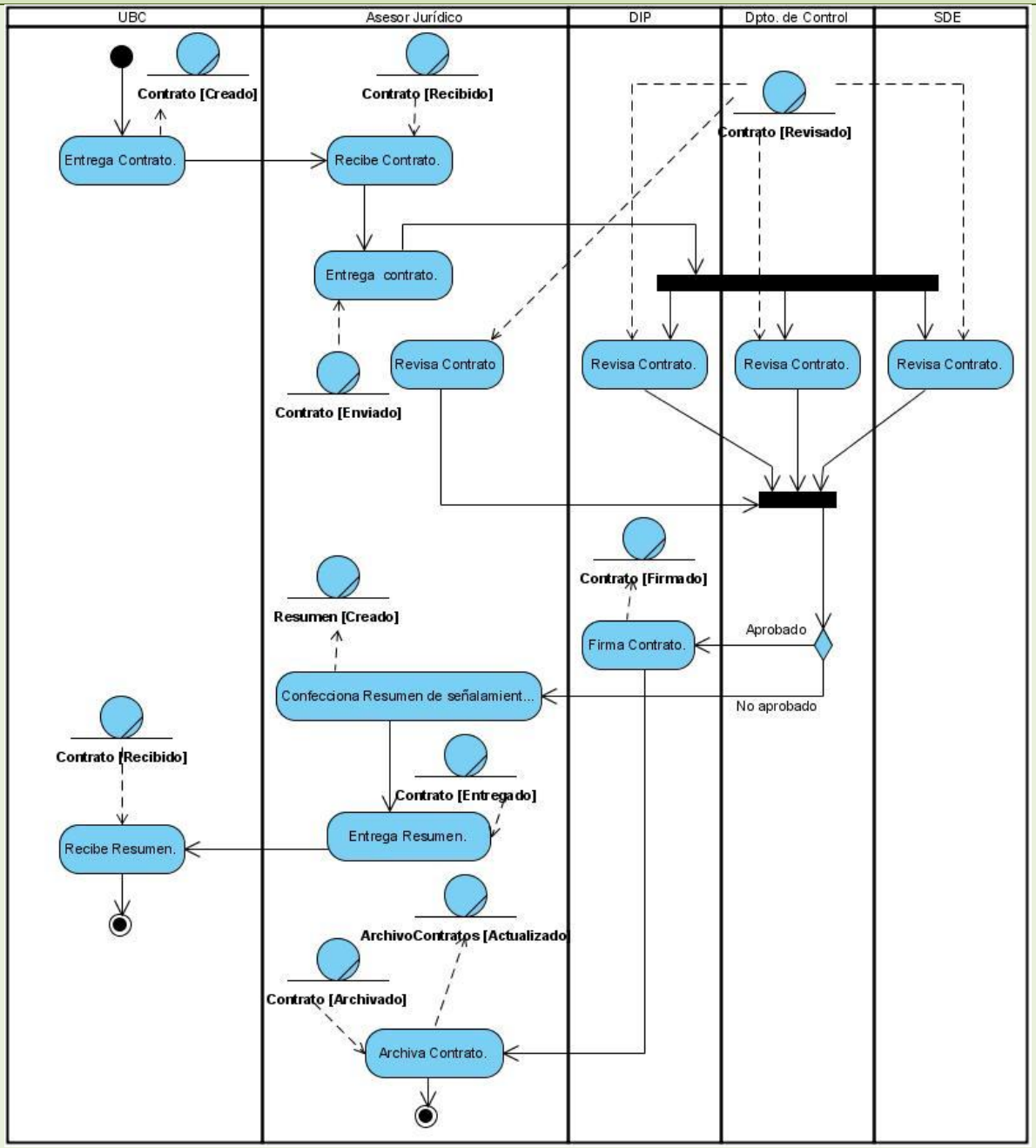


Figura A.5: Diagrama de Actividades – Caso de Uso "Aprobar Contrato".

Anexo 6.

Diagrama de Clases del Modelo de Objetos – Caso de Uso “Aprobar Factura”.

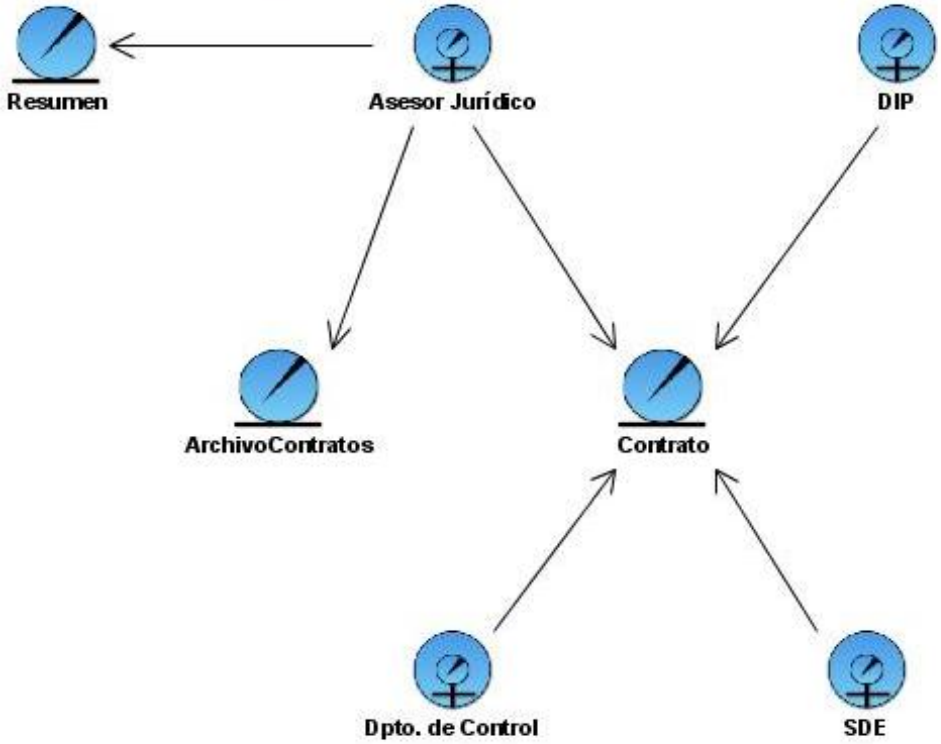


Figura A.6: Diagrama de Clases del Modelo de Objetos – Caso de Uso “Aprobar Contrato”.

Anexo 7.

Diagrama de Actividades – Caso de Uso “Realizar Solicitud de pago”.

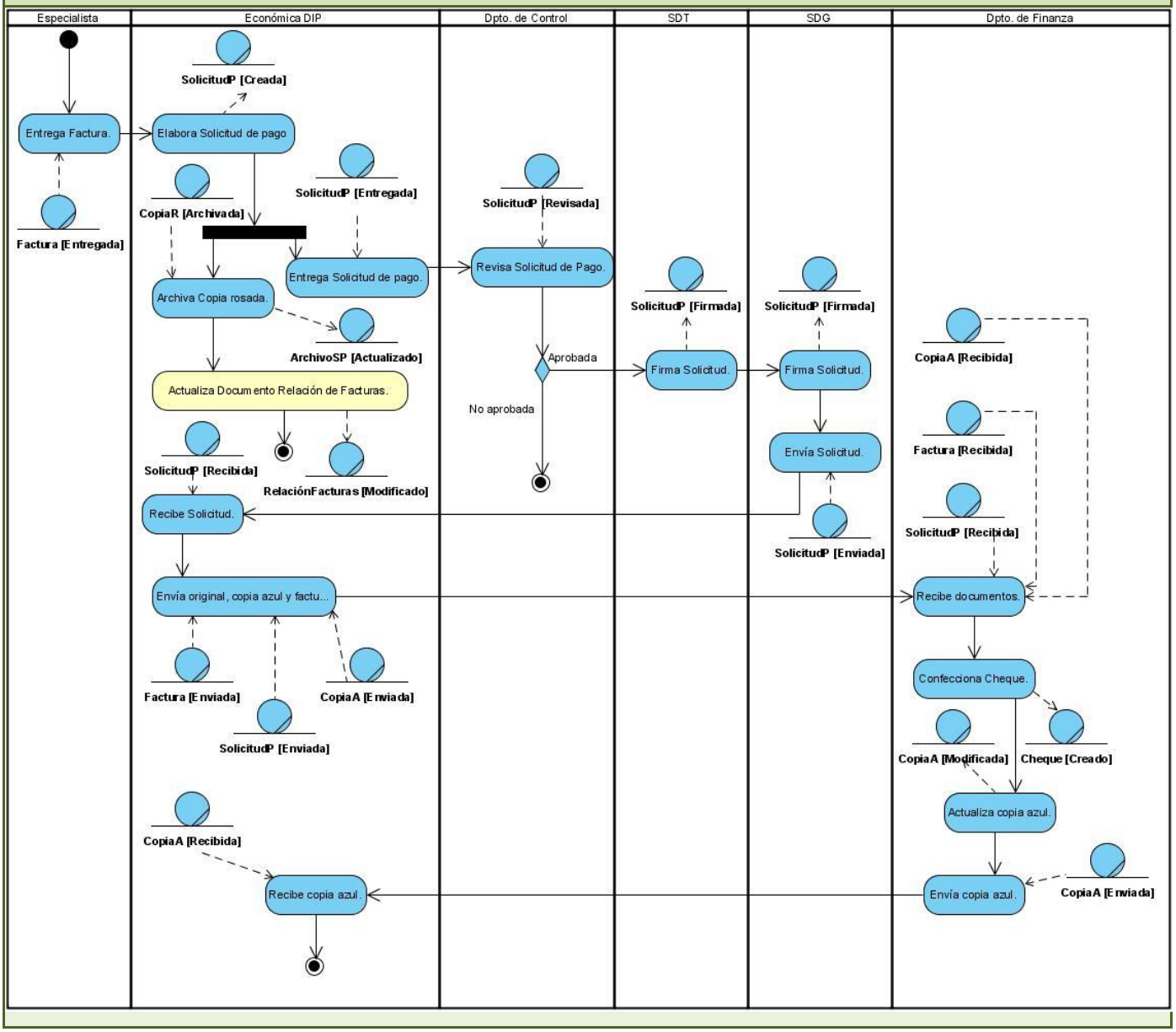


Figura A.7: Diagrama de Actividades – Caso de Uso “Realizar Solicitud de pago”.

Anexo 8.

Diagrama de Clases del Modelo de Objetos – Caso de Uso "Realizar Solicitud de pago".

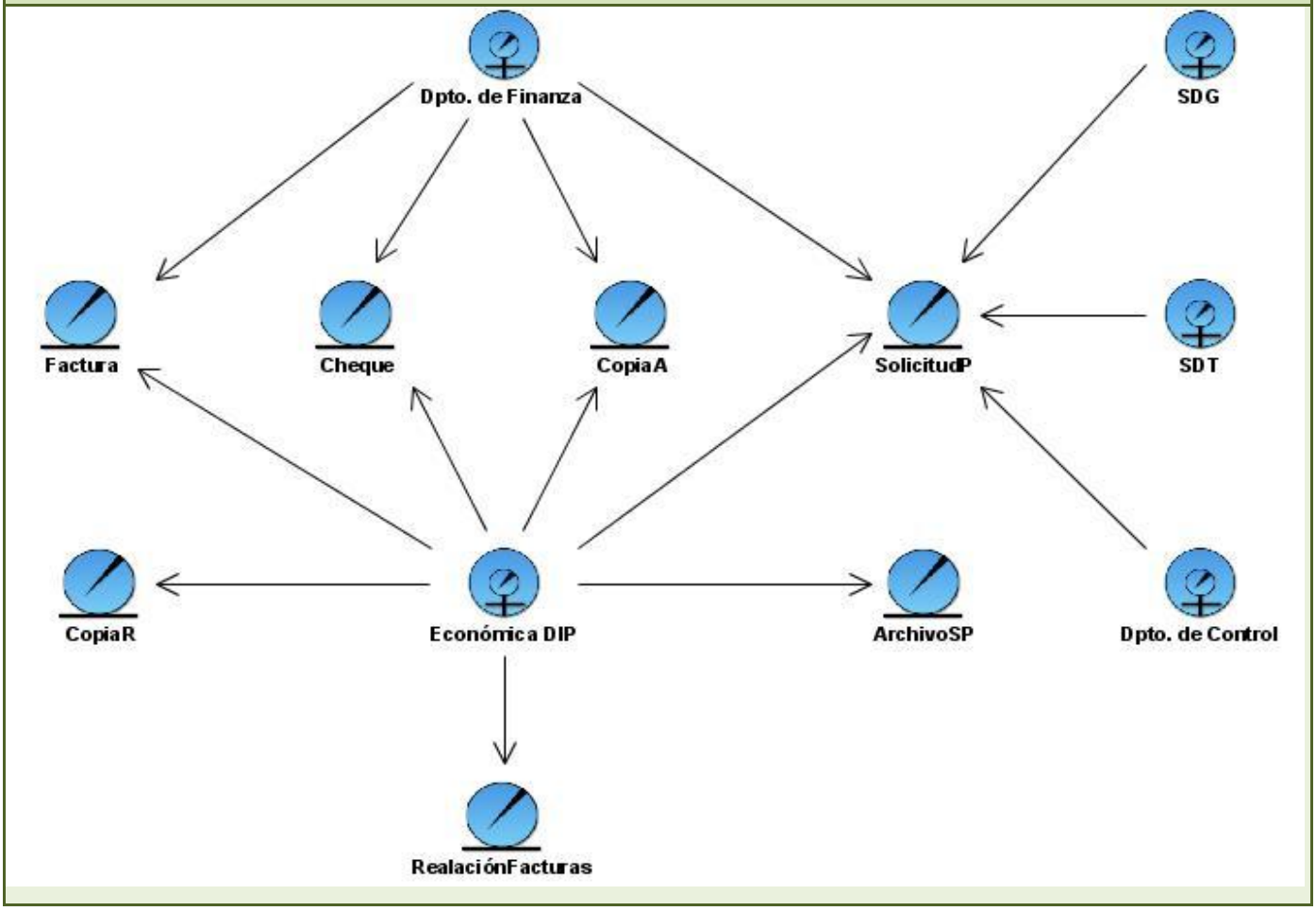


Figura A.8: Diagrama de Clases del Modelo de Objetos – Caso de Uso "Realizar Solicitud de pago".

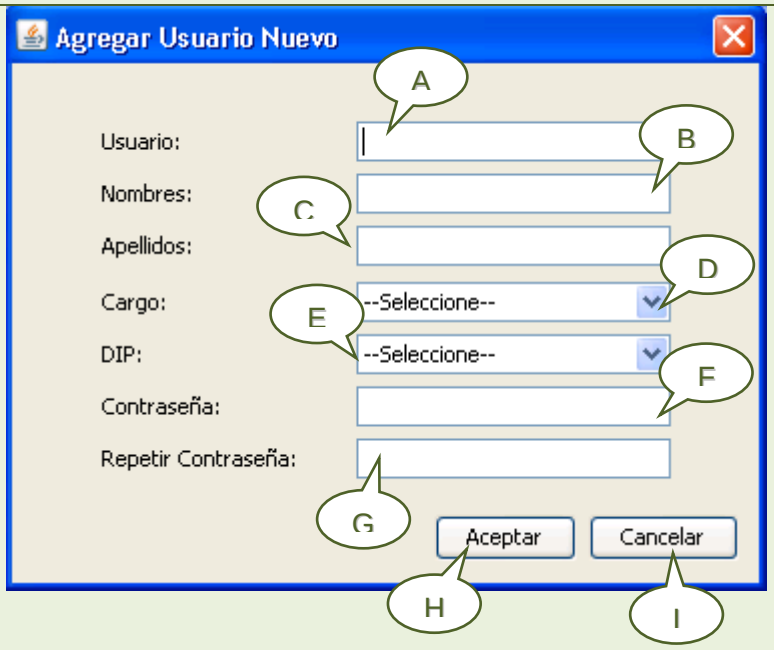
Anexo 9.

Tabla A.1: Descripción Textual del Caso de Uso del Sistema "Gestionar Usuarios".

Caso de Uso "Gestionar Usuarios"	
CU_1	**"Gestionar Usuarios"**
Propósito	Permitir al Dpto. de Control gestionar los usuarios en el sistema.
Actores	Dpto. de Control (inicia).
Resumen	El caso de uso inicia cuando el Dpto. de Control decide registrar un nuevo usuario o modificar algún dato de uno ya existente. Realiza la operación deseada y finaliza así el caso de uso.
Referencias	R1, R1.1, R1.2, R3.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El Dpto. de Control necesita registrar o modificar los datos de un usuario.	1.1 El sistema ejecuta alguna de las siguientes acciones: a) Si decide registrar un nuevo usuario, ir a la sección "Registrar Usuario". b) Si decide modificar los datos de un usuario, ir a la sección "Modificar Usuario". c) Si decide activar o desactivar a un usuario, ir a la sección "Activar/Desactivar Usuario".
Sección "Registrar Usuario"	
Acción del Actor	**Respuesta del Sistema**
2. El Dpto. de Control entra los datos necesarios para registrar un usuario al sistema (Usuario (A), Nombres (B), Apellidos (C), Cargo (D), DIP a la que pertenece (E), Contraseña (F), Repetir contraseña (G)) y presiona el botón Aceptar (H).	2.1 El sistema verifica que todos los campos estén llenos. 2.2 El sistema verifica que este nombre de usuario no exista. 2.3 El sistema verifica que el campo Contraseña (F) y Repetir contraseña (G) sean iguales.

	2.4 El usuario se almacena en el sistema, finalizando así el caso de uso.

Prototipo de Interfaz (Interfaz I)



A. Control para introducir el nombre de usuario que va a tener el nuevo usuario del sistema.

Nombre: Usuario Tipo: JTextField

B. Control para introducir el/los nombre/s del usuario nuevo.

Nombre: Nombres Tipo: JTextField

C. Control para introducir los apellidos del nuevo usuario.

Nombre: Apellidos Tipo: JTextField

D. Control para seleccionar el cargo que tiene este nuevo usuario. Los cargos pueden ser: Administrador, Directivo, Especialista y Jurídico. Este cargo es quien les da a los usuarios el rol que tendrán en el sistema y el nivel de acceso que tendrá cada uno de ellos.

Nombre: Cargo Tipo: JComboBox

E. Control para seleccionar la DIP a la que pertenece el usuario. Estas pueden ser: Preparación de Obras, Docencia, Infra-Estructura Productiva, Logística, Residencia y Facultades Regionales.

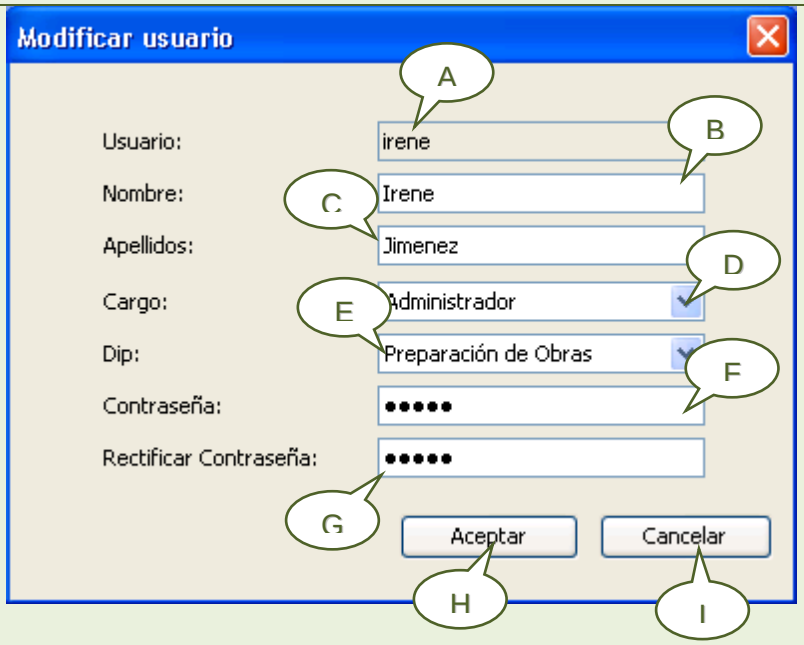
Nombre: DIP Tipo: JComboBox

F. Control para registrar la contraseña que tendrá este usuario.

<table>
<tr><td colspan="2">Nombre: Contraseña Tipo: JPasswordField
G. Control repetir la contraseña, lo que hace es que verifica si concuerda con la puesta en el campo anterior.
Nombre: RepetirContraseña Tipo: JPasswordField
H. Control de aceptar una acción (Registrar el nuevo usuario).
Nombre: Aceptar Tipo: JButton
I. Control de cancelar una acción.
Nombre: Cancelar Tipo: JButton</td></tr>
<tr><th colspan="2">Flujo Alterno de Eventos</th></tr>
<tr><th>Acción del Actor</th><th>Respuesta del Sistema</th></tr>
<tr><td></td><td>2.1 Se emite un mensaje para que llene todos los campos “Todos los campos son obligatorios, por favor rectifique”.
2.2 Si las contraseñas no coinciden se emite un mensaje “Las contraseñas deben coincidir.”
2.3 Si el usuario existe se emite un mensaje informando la existencia del mismo y se finaliza así el caso de uso “Ya se encuentra registrado otra persona con ese identificador, seleccione otro”.</td></tr>
<tr><th colspan="2">Sección “Modificar Usuario”</th></tr>
<tr><th>Acción del Actor</th><th>Respuesta del Sistema</th></tr>
<tr><td>2. El Dpto. de Control selecciona el usuario a modificar.</td><td>2.1 El sistema brinda la posibilidad de modificar los datos, menos el nombre de usuario.</td></tr>
<tr><td>3. El Dpto. de Control realiza las actualizaciones deseadas en los campos (Nombres (B), Apellidos (C), Cargo (D), DIP a la que pertenece (E), Contraseña (F) y Repetir contraseña (G)) y presiona el botón Aceptar (H) si desea guardar los cambios realizados</td><td>3.1 El sistema verifica que todos los campos estén llenos.
3.2 El sistema verifica que las contraseñas de los campos Contraseña (F) y Repetir contraseña (G) coinciden.
3.3 Se actualiza la información incorporada al</td></tr>
</table>

	usuario, finalizando así el caso de uso.

Prototipo de Interfaz (Interfaz II)



A. Control donde aparece el nombre de usuario de usuario a modificar, este control aparece desactivado, o sea, es lo único que no podrá ser cambiado.

Nombre: Usuario Tipo: JTextField

B. Control donde sale el/los nombre/s actuales del usuario que puede ser modificado.

Nombre: Nombres Tipo: JTextField

C. Control aparecen los apellidos del usuario que puede ser modificado.

Nombre: Apellidos Tipo: JTextField

D. Control que define el cargo que tiene este usuario. Los cargos pueden ser: Administrador, Directivo, Especialista y Jurídico. Este cargo es quien les da a los usuarios el rol que tendrán en el sistema y el nivel de acceso que tendrá cada uno de ellos.

Nombre: Cargo Tipo: JComboBox

E. Control donde aparece la DIP a la que pertenece el usuario. Estas pueden ser: Preparación de Obras, Docencia, Infra-Estructura Productiva, Logística, Residencia y Facultades Regionales.

Nombre: DIP Tipo: JComboBox

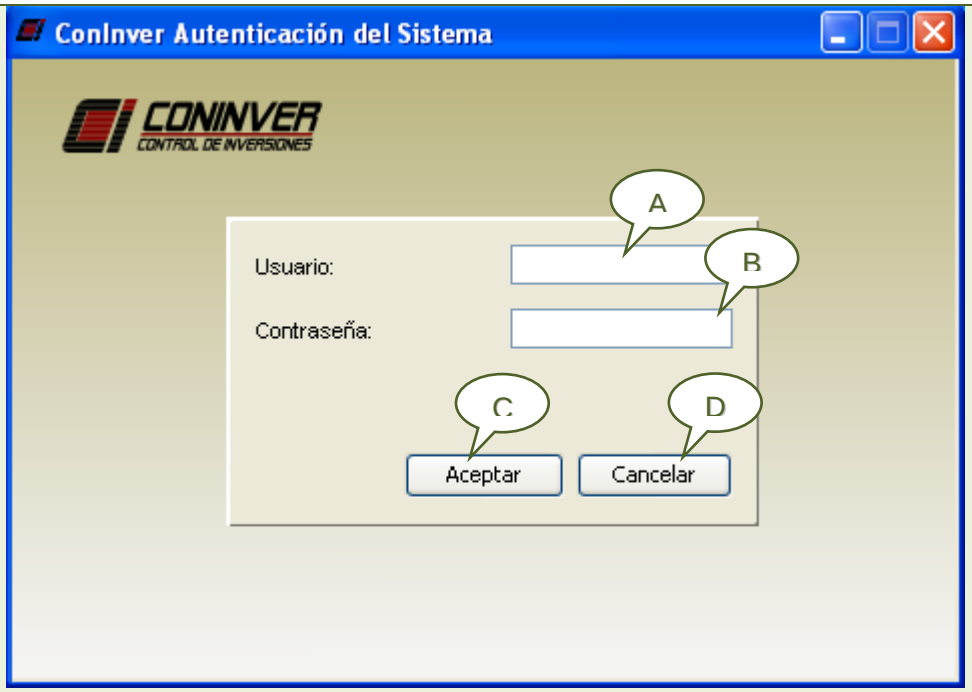
F. Control para mostrar la contraseña del usuario.

<table>
<tr><td colspan="3">Nombre: Contraseña Tipo: JPasswordField
G. Control para mostrar la contraseña, lo que hace es que verifica si concuerda con la puesta en el campo anterior.
Nombre: RepetirContraseña Tipo: JPasswordField
H. Control de aceptar una acción (Modificar usuario).
Nombre: Aceptar Tipo: JButton
I. Control de cancelar una acción.
Nombre: Cancelar Tipo: JButton</td></tr>
<tr><td colspan="3">Flujo Alterno de Eventos</td></tr>
<tr><td>Acción del Actor</td><td colspan="2">Respuesta del Sistema</td></tr>
<tr><td></td><td colspan="2">3.1 Se emite un mensaje informando que todos los campos deben ser llenados “Todos los campos son obligatorios, por favor rectifique”.
3.2 Se emite un mensaje de que los campos Contraseña (F) y Repetir contraseña (G) deben ser iguales “Las contraseñas deben coincidir”.</td></tr>
<tr><td colspan="3">Sección “Activar/Desactivar Usuario”</td></tr>
<tr><td>Acción del Actor</td><td colspan="2">Respuesta del Sistema</td></tr>
<tr><td>4. El Dpto. de Control decide activar/desactivar a un usuario ya registrado en el sistema. Por lo que marca el usuario deseado y presiona la opción Activar/Desactivar Usuario.</td><td colspan="2">4.1 El usuario seleccionado cambia a su estado contrario, si estaba activado a desactivado y si estaba desactivado a activado. Una vez el usuario este desactivado, no podrá acceder al sistema hasta que su estado pase a ser nuevamente activado, terminando así el caso de uso.</td></tr>
<tr><td>Prioridad</td><td colspan="2">Crítico.</td></tr>
</table>

Anexo 10.

Tabla A.2: Descripción Textual del Caso de Uso del Sistema "Autenticar Usuarios".

Caso de Uso "Autenticar Usuario"	
CU_2	**"Autenticar Usuario"**
Propósito	Permitir al usuario del sistema poder acceder al sistema.
Actores	Usuario (inicia).
Resumen	El caso de uso inicia cuando el usuario decide acceder a la aplicación, por lo que debe autenticarse para garantizar la seguridad del sistema, entra los datos que se especifican para acceder al sistema, estos se, verifican y finaliza cuando el sistema le otorga los permisos según el rol que desempeña.
Referencias	R2.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El usuario abre la aplicación.	1.1 El sistema le muestra la Interfaz III con los campos para introducir el Usuario (A) y Contraseña (B) con las opciones Aceptar (C) desactivada que no lo hará hasta que introduzca los datos en los campos usuario y contraseña y Cancelar (D) activada.
2. El usuario completa los campos: Usuario (A), Contraseña (B) y pulsa el botón Aceptar (C).	2.1 El sistema encripta la contraseña. 2.2 Verifica que el nombre de usuario y la contraseña sean correctos. 2.3 En el caso de ser correctos se le asignan los permisos.
Prototipo de Interfaz (Interfaz III)	



A. Control para introducir el nombre de usuario por el que se va a entrar al sistema.

Nombre: Usuario Tipo: JTextField

B. Control para introducir la contraseña por la que el usuario se autenticará en el sistema.

Nombre: Contraseña Tipo: JPasswordField

C. Control de aceptar una acción.

Nombre: Aceptar Tipo: JButton

D. Control de cancelar una acción.

Nombre: Cancelar Tipo: JButton

Flujo Alterno de Eventos

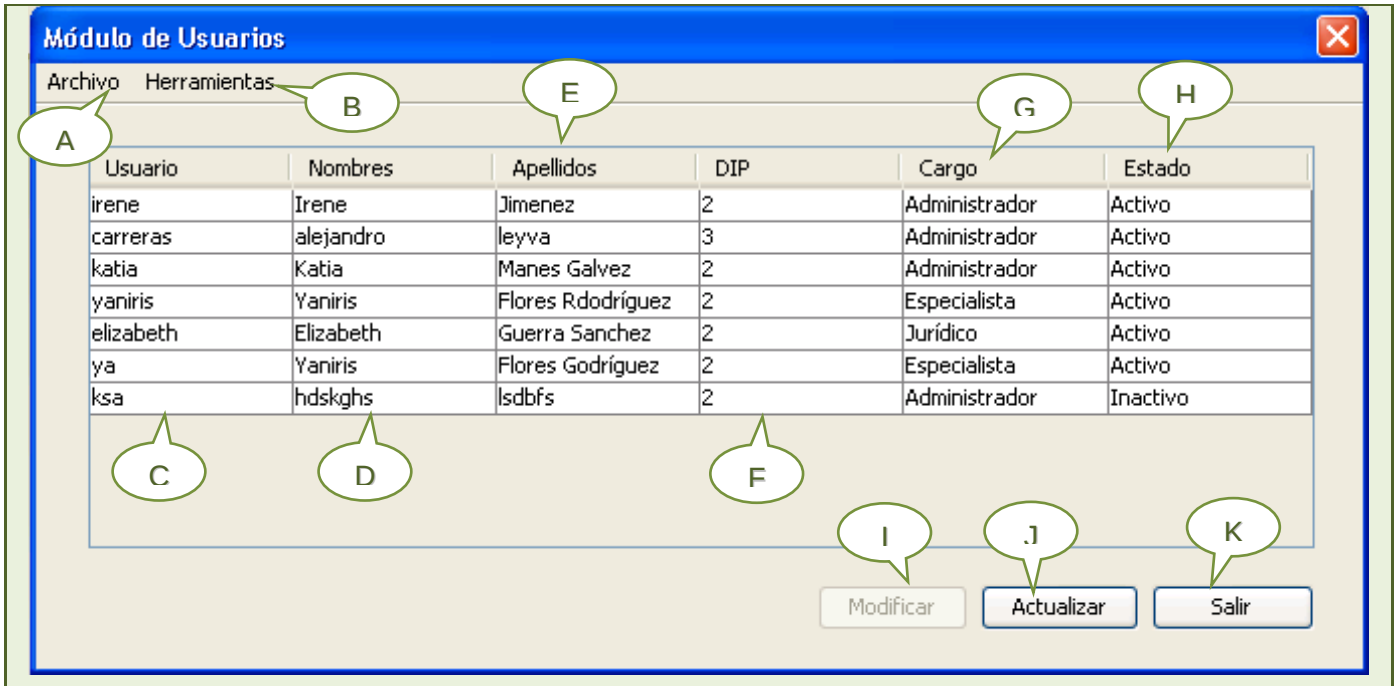
Acción del Actor	Respuesta del Sistema
	2.2 En caso de tener existir el usuario y no coincidir la contraseña, el sistema le muestra un mensaje de error “Contraseña no válida para el usuario.”
	2.3 En caso que el usuario no exista, le muestra un mensaje de error “Este usuario no se encuentra

	registrado en el sistema.”
	2.4 En caso que el usuario exista, pero el estado de este es desactivado, le muestra un mensaje de error “Cuenta deshabilitada, consulte al Departamento de Control para más detalles.”.
Prioridad	Crítico.

Anexo 11.

Tabla A.3: Descripción Textual del Caso de Uso del Sistema “Listar Usuarios”.

Caso de Uso “Listar Usuarios”	
CU_3	**“Listar Usuarios”**
Propósito	Permitir al Dpto. de Control listar todos los usuarios registrados en el sistema.
Actores	Dpto. de Control (inicia).
Resumen	El caso de uso inicia cuando el Dpto. de Control entra en la aplicación al módulo de usuarios y se muestra la lista de todos los usuarios registrados en el sistema, finalizando así el caso de uso.
Referencias	R4.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El Dpto. de Control selecciona en el sistema la opción Usuarios.	1.1 El sistema muestra una lista de todos los usuarios que hay registrados en el sistema, finalizando así el caso de uso.
Prototipo de Interfaz (Interfaz IV)	



A. Archivo y B. Herramientas, son parte de un menú donde

A. Archivo, da la opción de salir del sistema, al igual que I. Salir que es un botón para salir del modulo de usuarios. En las Herramientas aparecen las opciones: Adicionar Usuario, Modificar Usuario (no se activa hasta que tengas marcado a un usuario), lo mismo pasa con el J. Modificar que es un botón que permite modificar también un usuario seleccionado.

Nombre: Menú Tipo: JMenu

C. Columna que muestra los nombres de usuarios de todos los usuarios registrados en el sistema.

D. Columna que muestra todos los nombres de los usuarios registrados en el sistema.

E. Columna que muestra todos los apellidos de todos los usuarios del sistema.

F. Columna que muestra el número de la DIP a la que pertenece el usuario, cada nombre de DIP tiene asignado un número y ese es el que se muestra aquí. Preparación de Obras es el 2, Docencia es el 3, Infra-Estructura Productiva el 4, Logística el 5, Residencia el 6 y Facultades Regionales el 7

G. Columna que muestra el cargo de cada usuario registrado en el sistema. Pueden ser Administrador que es el Dpto. de Control, Directivo que es el Director, Especialista y Jurídico que es el Asesor Jurídico.

H. Columna que muestra el estado de cada usuario, el cual puede ser: activo, si puede acceder al

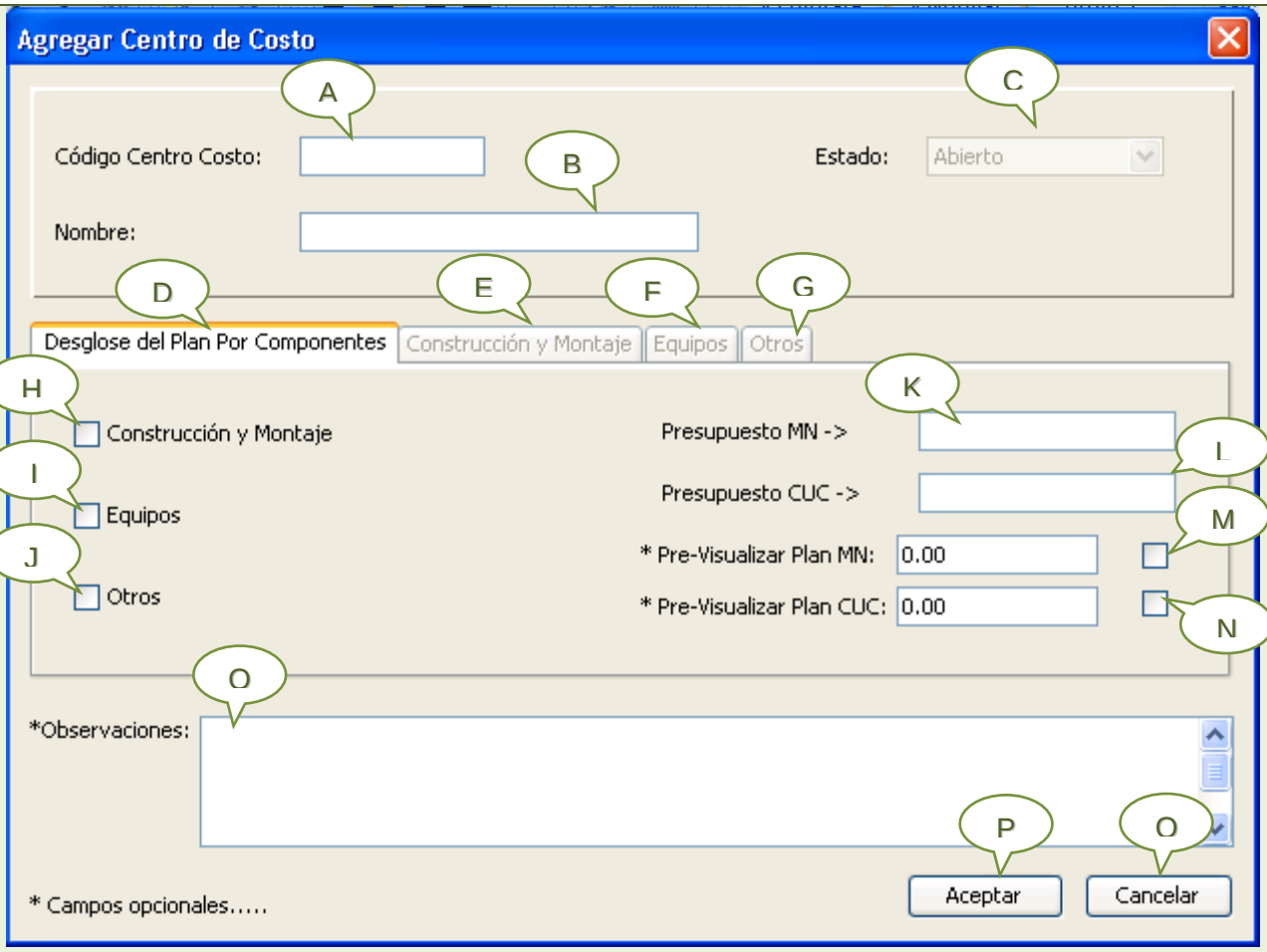
sistema, e inactivo si el usuario no puede acceder al sistema. I. Control que permite modificar un usuario que este seleccionado, este control aparece desactivado hasta tanto se seleccione el usuario al que se desea modificar sus datos. Nombre: Modificar Tipo: JButton J. Control que permite actualizar los cambios realizados en la gestión de los usuarios. Nombre: Actualizar Tipo: JButton K. Control que permite salir del módulo de usuarios de la misma forma que la opción Salir que hay en Archivo, en el menú principal del módulo. Nombre: Salir Tipo: JButton	
Prioridad	Crítico.

Anexo 12.

Tabla A.4: Descripción Textual del Caso de Uso del Sistema "Gestionar Objeto de Obra".

Caso de Uso "Gestionar Objeto de Obra"	
CU_4	**"Gestionar Objeto de Obra"**
Propósito	Permitir al Dpto. de Control gestionar los objetos de obras.
Actores	Dpto. de Control (inicia).
Resumen	El caso de uso inicia cuando el Dpto. de Control decide gestionar los objetos de obra. El sistema le permite registrar un nuevo objeto de obra y modificar uno ya existente. Se realiza la operación deseada, finalizando así el caso de uso.
Referencias	R5, R5.1, R5.2.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El Dpto. de Control necesita registrar o modificar los datos de un objeto de obra.	1.1 El sistema ejecuta alguna de las siguientes acciones: a) Si decide registrar un objeto de obra, ir a la sección "Registrar Objeto de Obra".

	b) Si decide modificar los datos de un objeto de obra ir a la sección “Modificar Objeto de Obra”.
Sección “Registrar Objeto de Obra”	
Acción del Actor	**Respuesta del Sistema**
2. El Dpto. de Control entra los datos necesarios para registrar un objeto de obra (que es lo mismo que centro de costo) al sistema (código de objeto de obra, descripción, plan por componentes, plan por monedas, estado del código de objeto de obra, observaciones).	2.1 El sistema verifica que todos los campos estén llenos. 2.2 El sistema verifica que este código de objeto de obra no exista. 2.3 El objeto de obra se almacena en el sistema, finalizando así el caso de uso.
Prototipo de Interfaz (Interfaz V)	



A. Control para insertar el código del centro de costo o el objeto de obra, que se desea registrar en el sistema

Nombre: Código Tipo: JTextField

B. Control para introducir el nombre que tendrá esta obra o centro de costo.

Nombre: Nombre Tipo: JTextField

C. Control que muestra el estado de la obra, cuando se registra una nueva esta opción sale por defecto como Abierta.

Nombre: Estado Tipo: JComboBox

D. Control para agrupar las opciones de la H a la N, donde se seleccionarán los componentes (Construcción y Montaje, Equipos y Otros) a los que se le pondrán los planes en base a las dos monedas MN y CUC, además aparecen dos campos para introducir el presupuesto completo de la obra

en ambas monedas.

Nombre: Desglose del plan por componentes Tipo: JPanel

E. Control para insertar el plan del componente Construcción y Montaje con su importe en MN y su importe en CUC.

Nombre: Construcción y Montaje Tipo: JPanel

F. Control para insertar el plan del componente Equipos con su importe en MN y su importe en CUC.

Nombre: Equipos Tipo: JPanel

G. Control para insertar el plan del componente Otros con su importe en MN y su importe en CUC.

Nombre: Otros Tipo: JPanel

H. Control para activar el panel Construcción y Montaje.

Nombre: Activar Construcción y Montaje Tipo: JCheckBox

I. Control para activar el panel Equipos.

Nombre: Activar Equipos Tipo: JCheckBox

J. Control para activar el panel Otros.

Nombre: Activar Otros Tipo: JCheckBox

K. Control para introducir el presupuesto de MN para la obra.

Nombre: Presupuesto MN Tipo: JTextField

L. Control para introducir el presupuesto de CUC para la obra.

Nombre: Presupuesto CUC Tipo: JTextField

M. Control que al ser marcado te pre visualiza el plan en MN de todos los componente, o sea la suma de todos los planes de MN de los tres componentes ya mencionados anteriormente.

Nombre: Plan MN Tipo: JCheckBOx

N. Control que al ser marcado te pre visualiza el plan en CUC de todos los componentes, o sea la suma de todos los planes de CUC de los tres componentes mencionados anteriormente.

Nombre: Plan CUC Tipo: JCheckBOx

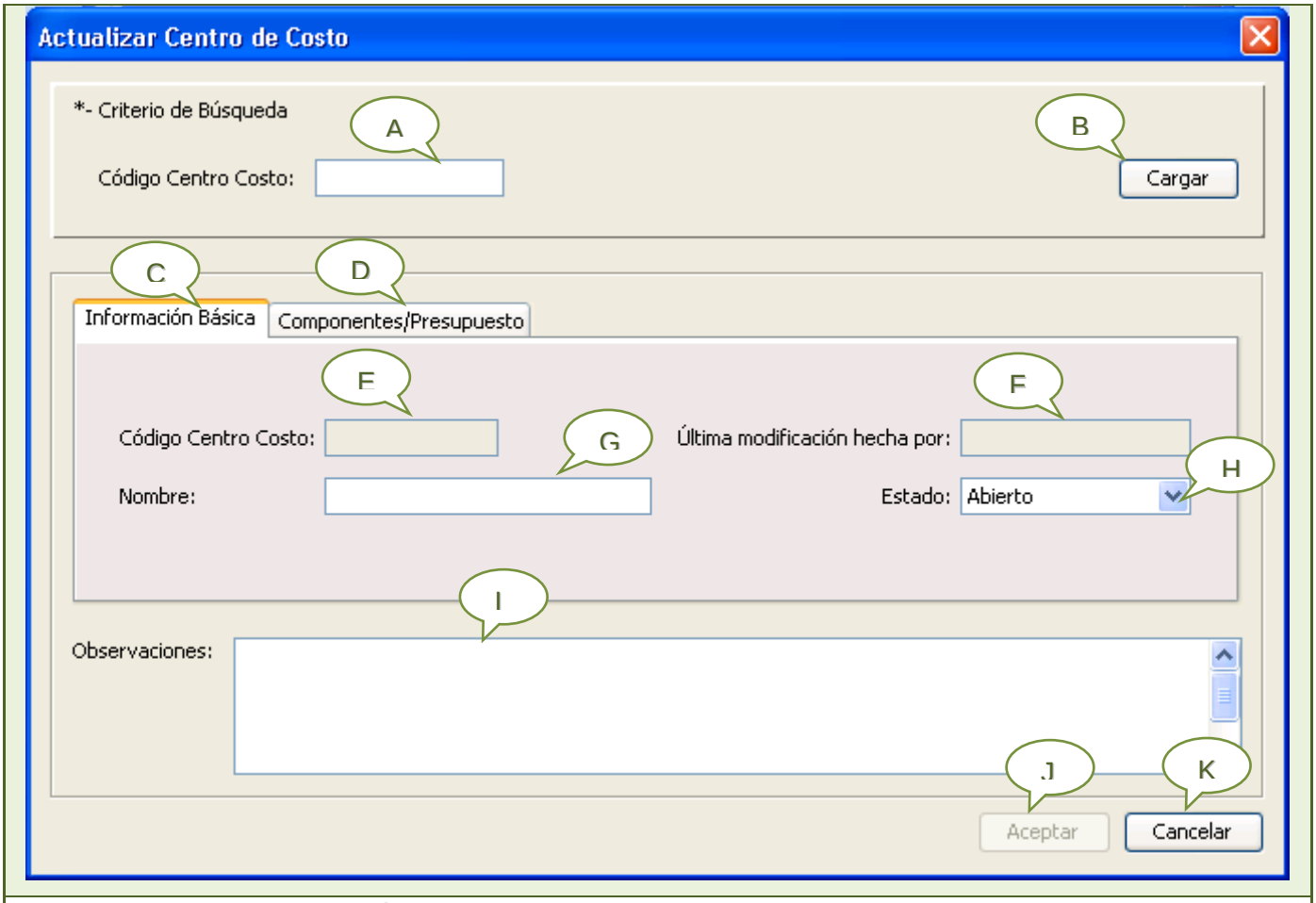
O. Control donde se escriben las observaciones, aquí se pueden insertar datos o cualquier información que se desee conocer con posterioridad.

Nombre: Observaciones Tipo: JTextArea

P. Control para aceptar la acción de registrar un nuevo centro de costo.

Nombre: Aceptar Tipo: JButton

<table>
<tr><td colspan="2">Q. Control para cancelar la acción de registrar un centro de costo y salir de esta sección.
Nombre: Cancelar Tipo: JButton</td></tr>
<tr><th colspan="2">Flujo Alterno de Eventos</th></tr>
<tr><th>Acción del Actor</th><th>Respuesta del Sistema</th></tr>
<tr><td></td><td>2.1 Se emite un mensaje para que llene todos los campos “Todos los campos son obligatorios”.
2.2 Si el código de objeto de obra existe se emite un mensaje informando la existencia del mismo y se finaliza así el caso de uso “Actualmente existe otro Centro de Costo con ese código, por favor rectifique”.</td></tr>
<tr><th colspan="2">Sección “Modificar Objeto de Obra”</th></tr>
<tr><th>Acción del Actor</th><th>Respuesta del Sistema</th></tr>
<tr><td>2. El Dpto. de Control introduce en un campo el código de objeto de obra (A) que desea modificar y pulsa el botón Cargar (B).</td><td>2.1 El sistema carga todos los datos de este centro de costo y le da la posibilidad de modificar todos los datos, menos el código de objeto de obra.</td></tr>
<tr><td>3. El Dpto. de Control realiza las actualizaciones deseadas y pulsa el botón Aceptar (J).</td><td>3.4 El sistema verifica que todos los campos estén llenos.
3.5 Se actualiza la información incorporada al usuario, finalizando así el caso de uso.</td></tr>
<tr><th colspan="2">Prototipo de Interfaz (Interfaz VI)</th></tr>
</table>



A. Control para introducir el código de la obra que se desea modificar.

Nombre: Código Tipo: JTextField

B. Control para cargar los datos del centro de costo que tiene ese código.

Nombre: Cargar Tipo: JButton

C. Control que agrupa algunos de los datos que pueden ser modificados de los centros de costo y otros que no pueden ser modificados

Nombre: Información Básica Tipo: JPanel

D. Control que agrupa las opciones que pueden ser modificadas de las obras, tales datos son Planes en ambas monedas (NM y CUC) para los tres tipos de componentes (Construcción y Montaje, Equipos y Otros), además del Presupuesto en ambas monedas.

Nombre: Componentes/Presupuesto Tipo: JPanel

E. Control que muestra el código de centro de costo, este código no puede ser modificado, por lo que se encuentra desactivado.

Nombre: Código Centro de Costo Tipo: JTextField

F. Control que muestra el nombre de usuario que hizo la última actualización al centro de costo.

Nombre: Última modificación Tipo: JTextField

G. Control que muestra el nombre del centro de costo, el cual puede ser modificado.

Nombre: Nombre Tipo: JTextField

H. Control que muestra el estado de la obra, el cual puede ser abierto, cerrado y garantía, estos estados pueden ser modificados.

Nombre: Estado Tipo: JComboBox

I. Control donde aparecen las observaciones hechas, estas pueden ser modificadas.

Nombre: Observaciones Tipo: JTextArea

J. Control para aceptar la acción de actualización del centro de costo.

Nombre: Aceptar Tipo: JButton

K. Control para cancelar la acción de actualizar el centro de costo y salir de esta sección.

Nombre: Cancelar Tipo: JButton

Flujo Alterno de Eventos	
Acción del Actor	**Respuesta del Sistema**
	2.1 Si el código de centro de costo no está registrado en el sistema se muestra un mensaje informando al usuario “El código no se corresponde con ninguna obra registrada en la base de datos”.
	3.1 Se emite un mensaje informando que todos los campos deben ser llenados, “Todos los campos son obligatorios”.
Prioridad	Crítico.

Anexo 13.

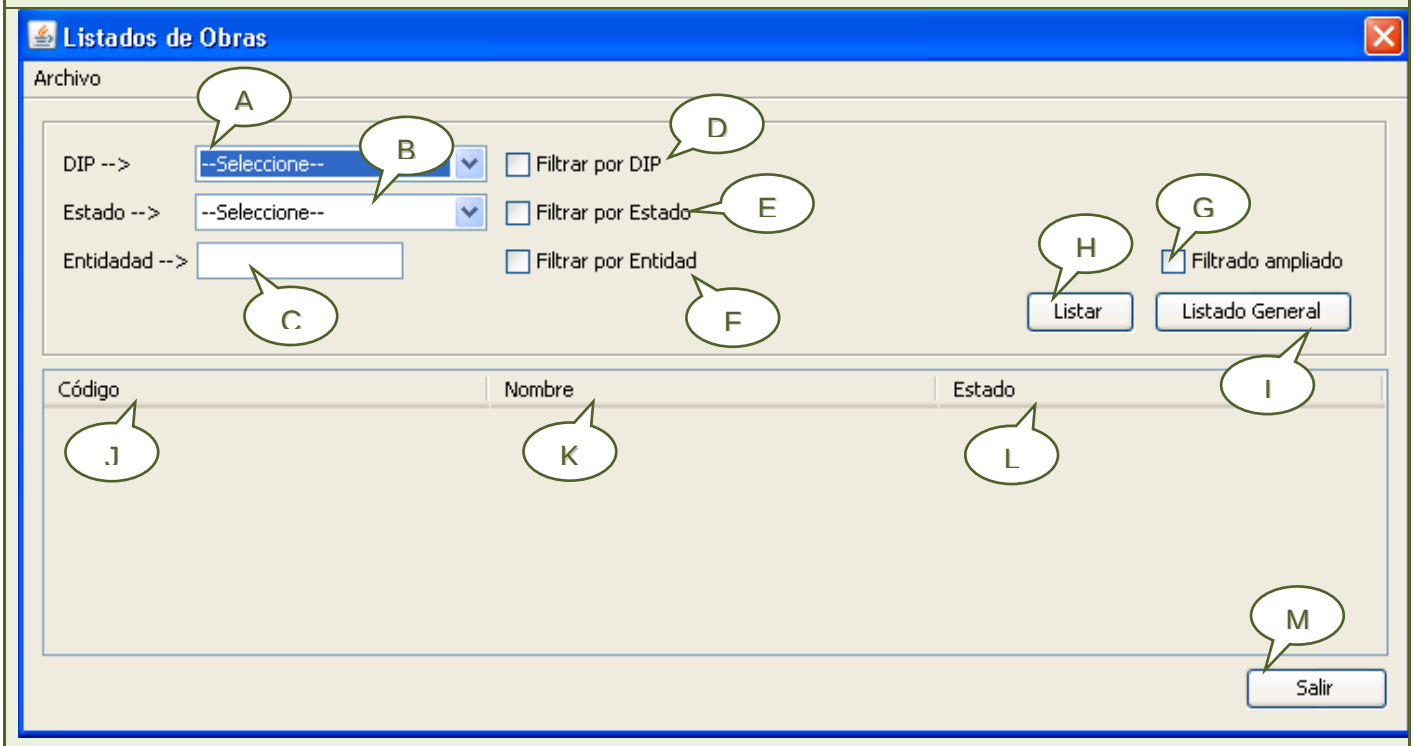
Tabla A.5: Descripción Textual del Caso de Uso del Sistema "Listar Objetos de Obra".

Caso de Uso "Listar Objetos de Obra"	
CU_5	**"Listar Objetos de Obra"**
Propósito	Permitir al tener un listado de los centros de costo cuando este lo desee.
Actores	Dpto. de Control (inicia).
Resumen	El caso de uso inicia cuando el Dpto. de Control decide acceder a la aplicación por lo que debe autenticarse para garantizar la seguridad del sistema, entra los datos que se especifican para acceder al sistema, estos se verifican y finaliza cuando el sistema le otorga los permisos según el rol que desempeña.
Referencias	R6.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El Dpto. de Control accede a listar centros de costo.	1.1 El sistema le muestra la Interfaz VII, dándole la posibilidad de hacer una selección de los aspectos por los cuales desea listar los centros de costo.
2. El Dpto. de Control selecciona la opción por la que desea filtrar los centros de costo.	2.1 El sistema ejecuta alguna de las siguientes acciones: a) Si selecciona la opción Filtrar por DIP (D), ir a la sección "Filtrado por DIP". b) Si selecciona la opción Filtrar por Estado (E), ir a la sección "Filtrado por Estado". c) Si selecciona la opción Filtrar por Entidad (f), ir a la sección "Filtrado por Entidad". d) Si selecciona más de un criterio de los antes puestos (D, E y F), ir a la sección "Filtrado ampliado". e) Si selecciona la opción Listado General (I),

	ir a la sección “Listado General”.
Sección “Filtrado por DIP”	
Acción del Actor	**Respuesta del Sistema**
3. El Dpto. de Control selecciona la DIP (A) y pulsa en el botón Listar (H).	3.1 El sistema lista todos los centros de costo que coinciden con la DIP seleccionada por el usuario, mostrando los datos Código (J), Nombre (K) y Estado (L).
Sección “Filtrado por Entidad”	
Acción del Actor	**Respuesta del Sistema**
4. El Dpto. de Control selecciona la Entidad (B) y pulsa en el botón Listar (H).	4.1 El sistema lista todos los centros de costo que coinciden con el Estado seleccionada por el usuario, mostrando los datos Código (J), Nombre (K) y Estado (L).
Sección “Filtrado por Entidad”	
Acción del Actor	**Respuesta del Sistema**
5. El Dpto. de Control selecciona la Entidad (C) y pulsa en el botón Listar (H).	5.1 El sistema lista todos los centros de costo que coinciden con la Entidad seleccionada por el usuario, mostrando los datos Código (J), Nombre (K) y Estado (L).
Sección “Filtrado por Ampliado”	
Acción del Actor	**Respuesta del Sistema**
6. El Dpto. de Control selecciona más de una opción (A, B, C) y pulsa en el botón Listar (H).	6.1 El sistema lista todos los centros de costo que coinciden con los campos seleccionados por el usuario, mostrando los datos Código (J), Nombre (K) y Estado (L).
Sección “Filtrado General”	
Acción del Actor	**Respuesta del Sistema**
7. El Dpto. de Control pulsa en el botón Listado General (I).	7.1 El sistema lista todos los centros de costo que estén registrados en el sistema, mostrando los datos

	Código (J), Nombre (K) y Estado (L).

Prototipo de Interfaz (Interfaz VII)



A. Control que permite seleccionar la DIP a la que pertenecen las obras que se quieren listar.

Nombre: DIP Tipo: JComboBox

B. Control que permite seleccionar el estado de los centros de consto que se quieres listar, pueden ser abierto, cerrado y en garantía.

Nombre: Estado Tipo: JComboBox

C. Control que permite mediante una Entidad listar los centros de costo que tiene la misma.

Nombre: Entidad Tipo: JComboBox

D. Control que activa la búsqueda de centros de costo a listar por la DIP, de no estar marcado no se hace el filtrado por este parámetro.

Nombre: Filtrar por DIP Tipo: JCcheckBox

E. Control que activa la búsqueda de centros de conto a listar por el estado de los mismos, de no estar activada esta opción no se hace el filtrado por el estado.

Nombre: Filtrar por Estado Tipo: JCcheckBox F. Control que activa la búsqueda de centros de conto a listar por la Entidad, de no estar activada esta opción no se hace el filtrado por la entidad. Nombre: Filtrar por Entidad Tipo: JCcheckBox G. Control que permite al ser marcado que el filtrado se realice por más de un criterio, listando los centros de costo que coincidan en las opciones marcadas. Nombre: Filtrado Amplio Tipo: JCcheckBox H. Control que permite al pulsarlo que se listen los centros de costo que coincidan con los criterios definidos anteriormente. Nombre: Listar Tipo: JButton I. Control que permite listar todos los centros de costo registrados en el sistema, no es necesario definir ningún criterio para que sean listados los centros de costo. Nombre: Listado General Tipo: JButton J. Columna donde aparecen listados todos los códigos de los centros de costo que serán mostrados. K. Columna donde aparecen listados todos los nombres de los centros de costo que serán mostrados. L. Columna que muestra el estado de los centros de costo a mostrar. M. Control que permite salir de la interfaz listar centros de costo.

Flujo Alterno de Eventos	
Acción del Actor	**Respuesta del Sistema**
	2.1 En caso que el usuario no haya seleccionado ninguno de los campos, le muestra el mensaje de error “Seleccione al menos un opción para listar”.
	3.1 En caso que no existan centros de costo que respondan a las opciones seleccionada por el usuario, le muestra un mensaje de error “No existe/n registrado/s en el sistema ningún centro de costo que responda a la/s opción/es marcada/s”.
Prioridad	Crítico.

Anexo 14.

Diagrama de Secuencia – Caso de Uso "Gestionar Usuario" – Escenario "Registrar Usuario".

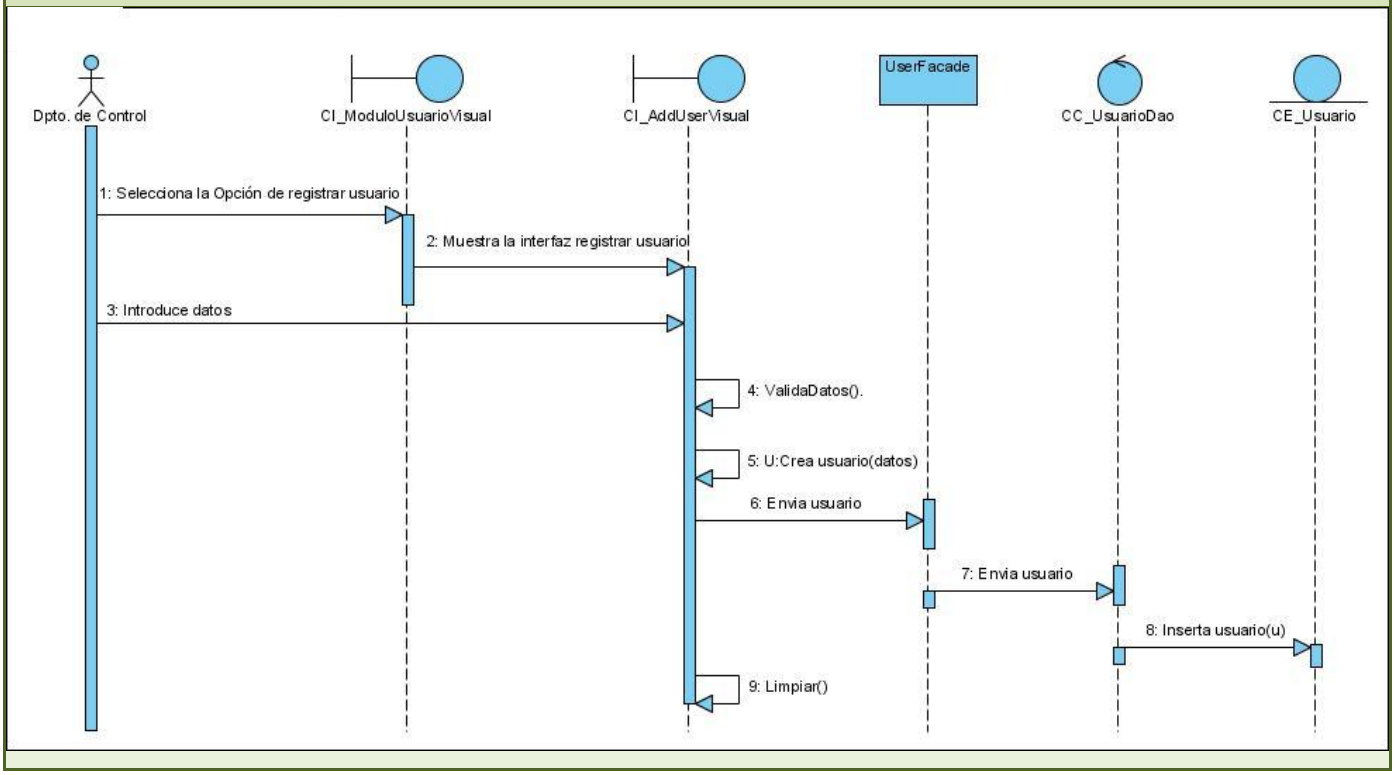


Figura A.9: Diagrama de Secuencia – Caso de Uso "Gestionar Usuario" – Escenario "Registrar Usuario".

Anexo 15.

Diagrama de Secuencia – Caso de Uso "Gestionar Usuario" – Escenario "Actualizar Usuario".

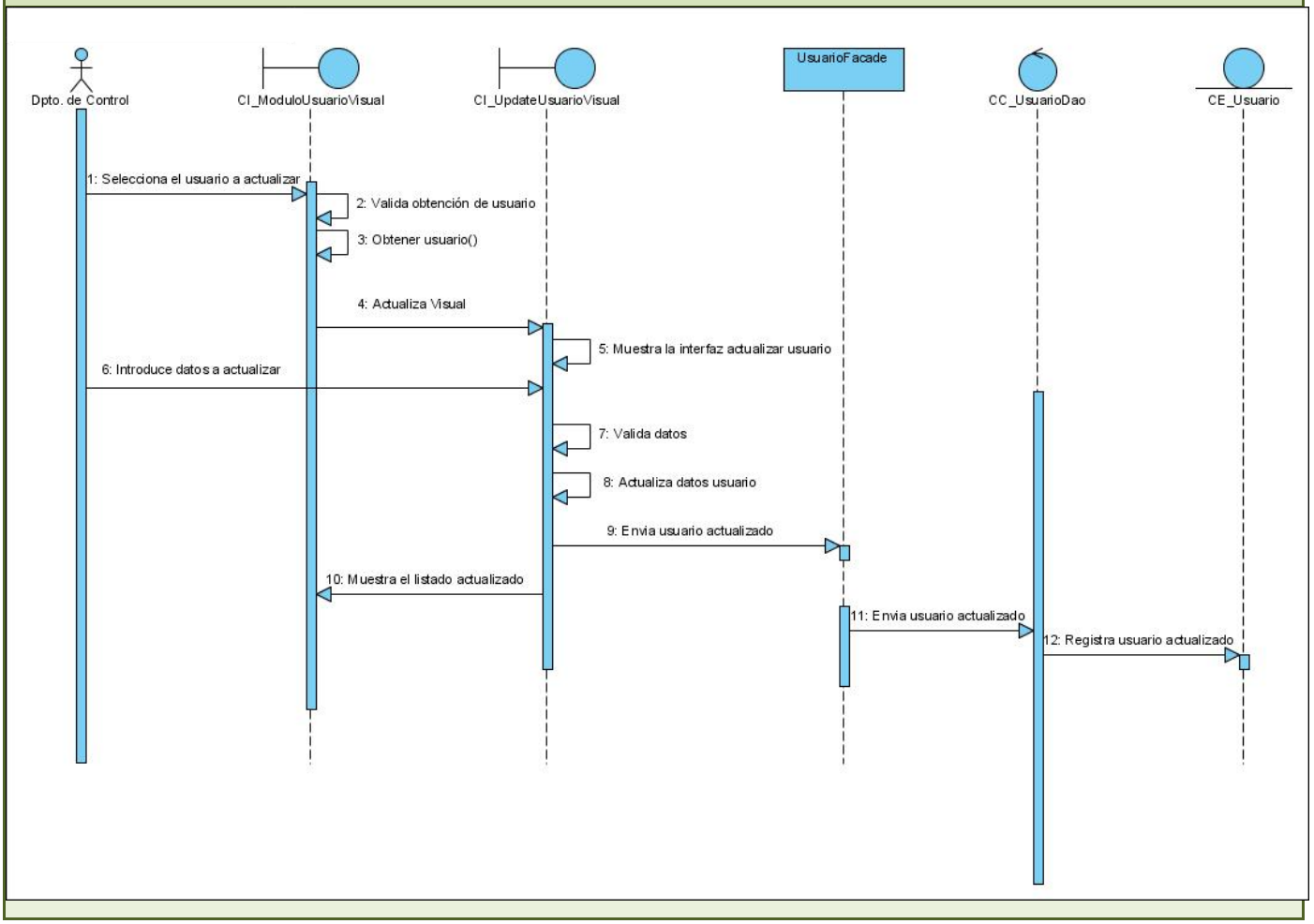


Figura A.10: Diagrama de Secuencia – Caso de Uso "Gestionar Usuario" – Escenario "Actualizar Usuario".

Anexo 16.

Diagrama de Secuencia – Caso de Uso "Gestionar Usuario" – Escenario "Activar/Desactivar Usuario".

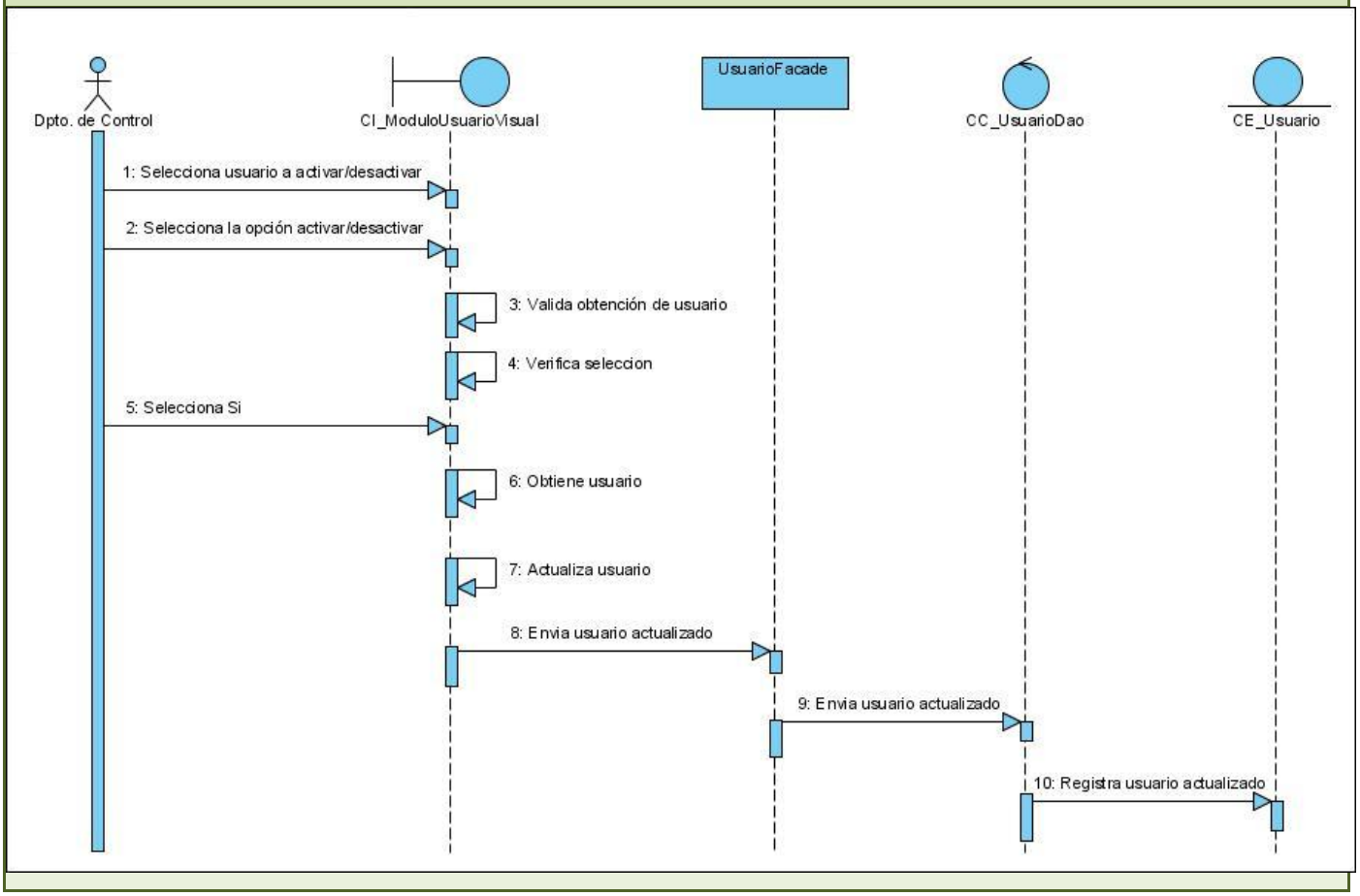


Figura A.11: Diagrama de Secuencia – Caso de Uso "Gestionar Usuario" – Escenario "Activar/Desactivar Usuario".

Anexo 17.

Diagrama de clases persistentes.

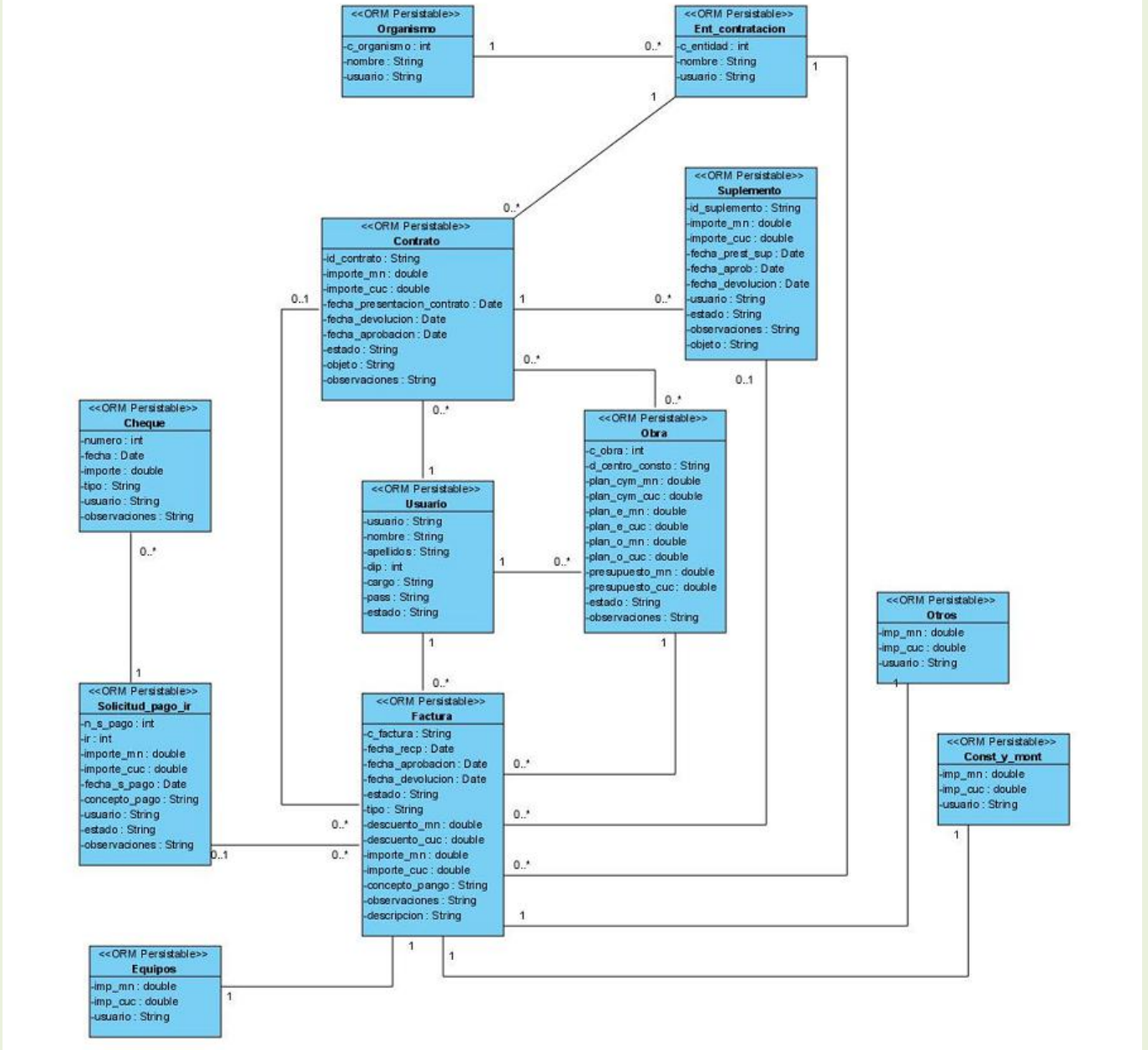


Figura A.12: Diagrama de clases persistentes.