UNIVERSIDAD DE ORIENTE
NÚCLEO DE ANZOATEGUI
ESCUELA DE INGENIERIA Y CIENCIAS APLICADAS
DEPARTAMENTO DE INGENIERIA DE COMPUTACIÓN Y SISTEMAS

DESARROLLO DE UN SOFTWARE PARA EL CONTROL DE LOS PROCESOS ADMINISTRATIVOS DE LA EMPRESA COOPERATIVA SAIT, R.L.

Presentado por:
Ronal Ricardo Rodríguez Velásquez
C.I.: 14.477.562

Trabajo de Grado presentado ante la Universidad de Oriente como Requisito Parcial para optar al Título de Ingeniero en Computación

Barcelona, Octubre de 2009

UNIVERSIDAD DE ORIENTE
NÚCLEO DE ANZOATEGUI
ESCUELA DE INGENIERIA Y CIENCIAS APLICADAS
DEPARTAMENTO DE INGENIERIA DE COMPUTACIÓN Y SISTEMAS

DESARROLLO DE UN SOFTWARE PARA EL CONTROL DE LOS PROCESOS ADMINISTRATIVOS DE LA EMPRESA COOPERATIVA SAIT, R.L.

ASESOR:

Ing. Claudio Cortinez
Asesor Académico

Barcelona, Octubre de 2009

UNIVERSIDAD DE ORIENTE

NÚCLEO DE ANZOATEGUI

ESCUELA DE INGENIERIA Y CIENCIAS APLICADAS

DEPARTAMENTO DE INGENIERIA DE COMPUTACIÓN Y SISTEMAS

DESARROLLO DE UN SOFTWARE PARA EL CONTROL DE LOS PROCESOS ADMINISTRATIVOS DE LA EMPRESA COOPERATIVA SAIT, R.L.

JURADO CALIFICADOR:

Ing. Rhonald Rodríguez
Jurado Principal

Ing. Gabriela Veracierta
Jurado Principal

Ing. Claudio Cortinez
Asesor Académico

Barcelona, Octubre de 2009

ARTÍCULO 44

De acuerdo con el artículo 44 del reglamento de Trabajo de Grado:

"Los trabajos de grado son de exclusiva propiedad de la Universidad y sólo podrán ser utilizados para otros fines con el conocimiento del Consejo de Núcleo respectivo, quién lo participará al Consejo Universitario".

DEDICATORIA

Este proyecto es la culminación de mi carrera de pregrado, una etapa de la que estoy muy orgulloso y feliz por todos los buenos momentos que pasé mientras recorría este camino.

Le dedico este proyecto primeramente y de corazón a Armando que todos los días desde hace tres años anhelaba que cumpliera con el objetivo, al fin terminé papá.

A mis dos mamás, Norma y Canchunchú por ayudarme a ser un hombre de bien y a mi amada Carolina, mi esposa linda, que siempre ha estado a mi lado apoyándome con todo su amor.

A mi tía Nere, la Sra. Marbellys y a las personas que siempre que podían me recordaban que debía seguir adelante y con sus palabras me animaban.

Y por último una dedicatoria especial para mi bebecita linda Isabel, que la quiero tanto, a mi abuelita Felicita, mi hermana Analyz y a las amistades con las que tuve el gusto de compartir durante mi carrera y que complementaron mi aprendizaje de formación personal y de vida.

Ronal Rodríguez

AGRADECIMIENTO

Dios mío te agradezco todos los días por que me has bendecido con tantas cosas buenas y me distes unos padres y una familia maravillosos, muchas gracias por ser tan bueno conmigo tu que haces que las cosas lleguen a ser, gracias Jehová.

Les agradezco a mis padres porque desde siempre me apoyaron con mucho amor en mis estudios y en todo lo que he hecho a lo largo de mi vida, haciendo de mí un hombre de bien.

A mi Carolina mi amiga, mi novia y esposa, con la que he compartido tanto y me ayudó a lo largo de toda la carrera y mucho más en la realización de este proyecto.

Gracias a José Fernando, Daniel, Dulmy, Amable, y todos aquellos amigos que me prestaron su apoyo de forma incondicional para poder alcanzar el objetivo.

A mi asesor Prof. Claudio Cortínez por su apoyo, ayuda e interés en este proyecto.

A la Universidad de Oriente por haberme dado la oportunidad de ser un profesional y por todos los bellos momentos que me brindó durante mi carrera de pregrado.

A la Cooperativa SAIT, RL por toda la colaboración y ayuda prestada para la realización de este proyecto.

Para finalizar les agradezco a todos los amigos con los que tuve el gusto de pasarla bien, a mis amigos de Ysaroa que nunca olvidaré, a Leo, Elic, Adri, Juan,

AGRADECIMIENTO

Marcos, Hely, Francys, Wladi, y una lista larguísima que en este pequeño espacio no cabe pero que está en mi memoria y mi corazón.

A todos muchas gracias.

RESUMEN

La Cooperativa SAIT, RL es una empresa orientada a proveer bienes y servicios en el área de la computación. La empresa presenta problemas en el manejo de sus procesos administrativos y de control interno y no cuenta con las herramientas ni el tiempo necesario para ello, trayendo como consecuencia acumulación de trabajo, malas decisiones, uso ineficiente de recursos financieros e incumplimiento con la Superintendencia Nacional de Cooperativas (SUNACOOP). Por las razones expuestas se propuso el presente trabajo de grado, el cual consiste en desarrollar un software para el control de los procesos administrativos de SAIT, RL entre los que destacan: compras, ventas, control de inventario, gestión de clientes y proveedores, historial de servicio. Para el desarrollo se utilizó la metodología del Proceso Unificado de Desarrollo de Software y el Lenguaje de Modelado UML, aplicando las fases de Inicio, Elaboración y Construcción a través de una o más iteraciones de los flujos de trabajo: captura de requisitos, análisis, diseño, implementación y prueba. Se usó el IDE C++Builder11, basado en el lenguaje C++, como herramienta de programación. La base de datos relacional del sistema fue implementada en MySQL Server 5.0, creando así una arquitectura tipo cliente/servidor para el manejo centralizado de la información.

CONTENIDO

CONTENIDO

CONTENIDO

CONTENIDO

CONTENIDO

LISTA DE FIGURAS

LISTA DE FIGURAS

LISTA DE TABLAS

LISTA DE TABLAS

CAPÍTULO I

EL PROBLEMA

1.1. Planteamiento Del Problema

La Cooperativa SAIT, RL es una empresa dedicada a proveer bienes y servicios en diversas áreas de la computación. Las siglas significan Soluciones Avanzadas en Informática y Tecnología.

SAIT, R.L. se conformó en octubre del año 2005 a través de la iniciativa de un grupo de estudiantes de ingeniería en computación de la Universidad de Oriente que buscaban solucionar problemas en distintas áreas de la computación, utilizando para ello los conocimientos adquiridos en la academia.

SAIT, R.L., está ubicada en el centro comercial Novocentro II, local 2-7 en la ciudad de Puerto La Cruz. Inicia formalmente sus actividades en marzo del año 2007 con un pequeño grupo de trabajadores y un presupuesto reducido pero con la visión de convertirse en una empresa líder en el mercado de servicios en computación.

La empresa ofrece soluciones de: Venta de equipos y accesorios de computación, soporte técnico a empresas, desarrollo de software, capacitación y adiestramiento en el área informática, mantenimiento de redes y equipos de computación, instalación de sistema operativos y soluciones de software, migración de bases de datos y aplicaciones, entre otros, pero además de los servicios que presta, SAIT, RL debe cumplir con los procesos administrativos de toda empresa como tener al día la información sobre compra y ventas, mantener accesible y actualizada la información de los clientes y proveedores; también, debe presentar informes y otros adicionales aplicados a las cooperativas como por ejemplo, los informes trimestrales de su estado económico a la Superintendencia Nacional de Cooperativas

(SUNACOOP), para elaborar dichos informes se debe sintetizar y tener al día la información de los servicios facturados.

Estos procesos administrativos y de control son el punto de flaqueza de la empresa, ya que el personal ha enfocado la mayor parte de su esfuerzo en cumplir con los compromisos adquiridos con los clientes, dejando poco espacio para las labores administrativas, las cuales se van “solventando” sobre la marcha y con poca experiencia.

Tampoco se ha invertido tiempo en desarrollar un sistema de control administrativo o mejorar los procesos, trayendo como consecuencia que diariamente se acumule y retrase el trabajo y se haga más difícil el manejo de la empresa, lo que a su vez genera otras consecuencias más graves como mala toma de decisiones, manejo poco eficiente de los recursos financieros y de la información administrativa, y haciendo difícil estar al día con los informes que pide la SUNACOOP.

Entre los problemas y deficiencias encontrados se pueden mencionar:

- La facturación de los servicios se hace a través de hojas de cálculo en las cuales no se guardan los datos del cliente ni de lo que se está facturando, y cada vez que se realiza este proceso se deben cargar nuevamente los datos del cliente y también trae retrasos en la búsqueda de información sobre determinada factura a la hora de cobrar un servicio.
- El inventario y los activos de la empresa es otro proceso que se lleva de forma manual, se deben cargar los distintos productos según su código y los seriales de fabricación, este proceso suele generar errores de trascripción lo que trae como consecuencia inconsistencia con la realidad.

- Se manejan presupuestos de venta de equipos por cada cliente, pero esta información también se lleva a través de hojas de cálculo y siempre debe crearse una nueva hoja y rellenar los datos del cliente, generando trabajo repetitivo. Además a la hora de facturar un equipo presupuestado deben cargarse nuevamente todos los datos del presupuesto en la factura ya que no hay bases de datos ni relación de información de ningún tipo.
- No hay manera de obtener rápidamente reportes significativos como históricos de compras o de ventas, o resúmenes por cliente o proveedor, entre otros; información que solo puede ser obtenida a medias e invirtiendo mucho tiempo.
- Puede perderse información importante o haber inconsistencias en la misma al no contarse con una base de datos que centralice e interrelacione la información.

Lo expuesto anteriormente causa gran pérdida de tiempo y descontrol en la gestión de la empresa, por lo que se plantea desarrollar un software adaptado a las necesidades de SAIT, R.L. que facilite el control de sus principales labores administrativas como: facturación, presupuestos, órdenes de servicio e inventario. También deberá contar con una base de datos para almacenar la información y permitir el uso eficiente de la misma. Deberá facilitar información de los clientes, proveedores, trabajadores, control de acceso de los usuarios y generar reportes significativos que ayuden en la gestión de la empresa y en la toma de decisiones.

El software se desarrollará utilizando la metodología del Proceso Unificado de Desarrollo de Software y el Lenguaje Unificado de Modelado (UML) que son de gran aceptación y utilización a nivel mundial. Se utilizará el paradigma de Programación

Orientada a Objetos para asegurar la escalabilidad, portabilidad y actualización del sistema y una base de datos basada en el Modelo Relacional.

Respecto al alcance que tendrá el proyecto, estará orientado a los procesos administrativos más importantes de SAIT, R.L. e incluirá el levantamiento de información y diseño del sistema y de las bases de datos necesarias, la codificación del software y las pruebas de funcionamiento.

Para SAIT, R.L. es muy importante automatizar sus procesos internos y contar con una herramienta nueva adaptada a sus necesidades, que haga más eficaz y eficiente el trabajo administrativo, e impulse su productividad sobre todo ahora que se está consolidando en el mercado de servicios de computación, crece rápidamente y para lograr sus metas futuras necesitará un mejor control y gestión de la empresa y acceso rápido a la información.

Este proyecto basa su originalidad en que por primera vez se desarrollará un software para la empresa SAIT, R.L. y dado que es una cooperativa tendrán que estudiarse los procesos y terminologías asociados a este tipo de empresa.

1.2. Objetivos

1.2.1. Objetivo General

- Desarrollar un software para el control de los procesos administrativos de la empresa Cooperativa SAIT, R.L.

1.2.2. Objetivos Específicos

- Recopilar información sobre los procesos administrativos que se realizan en la empresa.
- Realizar los modelos de análisis.
- Definir la arquitectura del sistema, adaptada a las necesidades de la empresa.

- Diseñar la base de dato relacional adecuada a la arquitectura del sistema.
- Codificar los módulos del sistema utilizando el paradigma de programación orientada a objetos.
- Realizar pruebas de validación, funcionalidad e integración de los módulos del sistema y con la base de datos.

CAPÍTULO II

MARCO TEÓRICO

2.1. Antecedentes

En la empresa Cooperativa SAIT, R.L. los procesos administrativos se realizan manualmente. Hasta la fecha no se ha implementado un sistema de información ni base de datos de algún tipo, que permita gestionar de manera eficiente la información y que ofrezca un soporte físico de almacenamiento de los datos.

No obstante en la Universidad de Oriente se han realizado numerosos trabajos de investigación orientados a la automatización de los procesos de una empresa a través del desarrollo de sistemas de información en los cuales se ha empleado la metodología y las técnicas necesarias para su realización.

Para el desarrollo del presente proyecto han sido necesarias la recopilación de conocimientos teóricos previos y la consulta de algunas de las mencionadas investigaciones pertenecientes, específicamente, al Departamento de Computación y Sistemas de la Universidad de Oriente-Núcleo de Anzoátegui, estos son:

- Fuad, N. (2008) desarrolló un trabajo de grado titulado: **"Desarrollo de un software para el control de inventario del material importado de una ensambladora de vehículos".** Este trabajo de grado tuvo como finalidad el desarrollo de un software que permitió una mayor eficiencia y rapidez de las actividades referentes al control de inventario del material importado de la empresa Toyota de Venezuela C.A., específicamente del departamento de Control de Producción. Se utilizó Proceso Unificado de Desarrollo de Software, y UML para la representación de la arquitectura del software, el

Modelo Relacional para la base de datos, y el paradigma de la Programación Orientada a Objetos para programar en lenguaje Visual Basic 6. [6]

- Zacarías, E. y Arcila, A. (2007) desarrollaron un trabajo de grado titulado: "Desarrollo un software que sirva de soporte para la automatización de las labores administrativas de la Unidad Educativa "Elisa Elvira Zuloaga" y Guardería-Preescolar "Trencito Mi Rosita Mística". Estos estudiantes elaboraron como trabajo de grado un sistema que tiene como propósito la automatización de los procedimientos administrativos de una Unidad Educativa utilizando para ello el Proceso Unificado de Desarrollo de Software, el Modelo Relacional de Datos, el Modelo Cliente – Servidor y el paradigma de la Programación Orientada a Objetos y como programa de codificación el Borland C++Builder 6 y manejador de datos MySQL Server.[8]

- Marmoud B. y Caraballo, A. (2007) desarrollaron un trabajo de grado titulado: "Desarrollo un software para la automatización del proceso de compras del 'Departamento de Compras No Productivas' de una empresa automotriz". Este trabajo de grado se asentó en el desarrollo de un software que permite la automatización de los procesos de control administrativos de la empresa automotriz, en lo relacionado con la gestión de requisiciones de compras. Se usó Proceso Unificado de Desarrollo de Software, y UML para la representación de la arquitectura del software, el Modelo Relacional para la base de datos, y el paradigma de la Programación Orientada a Objetos para programar en lenguaje Visual Basic 6.[7]

- Salazar, L. (2003) desarrolló un trabajo de grado titulado: **"Desarrollo de un software para automatizar las actividades de control interno de una empresa de transporte de alimentos"**. En este trabajo de grado se desarrolló un sistema de consultas y reportes asociados al control de los procesos asociados a las unidades de una empresa de transporte como: viajes realizados

por unidad, generación de pagos, etc. Y también de los datos referentes a los conductores, mercancías, rutas, fletes, entre otros. El desarrollo del proyecto se llevó a cabo utilizando UML, se codificó con Borland Delphi 7 y se utilizó una base de datos relacional en Microsoft Access. [15]

- Pugliese, I. (2001) desarrolló un trabajo de grado titulado: **"Desarrollo del sistema de compras en ambiente Cliente/Servidor para la Universidad de Oriente núcleo de Anzoátegui"**. Este proyecto, elaborado como trabajo de grado, consistió en el desarrollo de un software con el objetivo de gestionar el procesamiento de solicitudes de compras, ordenes de compra e informes, aplicando el modelo cliente servidor. El análisis y diseño fueron realizados bajo el proceso unificado de desarrollo de software utilizando el lenguaje Gráfico UML, y para su codificación se usó la herramienta Power Builder 6.5. El modelo de datos fue implementado en una Base de datos Oracle 7. [16]

2.2. Reseña De La Empresa [1]

SAIT, R.L. es una Cooperativa de Servicios Informáticos, conformada por un equipo de ingenieros en computación, técnicos en informática y licenciados en administración, con experiencia en ofrecer soluciones tecnológicas adaptadas a las necesidades o requerimientos de nuestros clientes y ajustándose a los cambios tecnológicos que vive el país.

2.2.1. Misión de SAIT, R.L.

Ofrecer servicios de alta calidad, gran valor, eficientes y confiables; para brindarle a nuestros clientes satisfacción total y garantizada de sus necesidades en el área informática y de tecnología.

2.2.2. Visión de SAIT, R.L

Ser una cooperativa líder en su campo, reconocida como una organización sólida y por su alta calidad de servicio, excelencia, productividad y confiabilidad; con el mejor recurso humano altamente capacitado y profesional, siempre innovando y en expansión, ofreciendo los mejores beneficios a sus trabajadores y en un ambiente de equidad, responsabilidad y seguridad social.

2.2.3. Servicios Ofrecidos en SAIT, R.L.

- **Capacitación y Soporte:** cursos de programación de aplicaciones web con PHP, Python, C++, JavaScript, ASP, Ajax; manejo de aplicaciones ofimáticas opensource (OpenOffice) y propietarias (Word, Excel, PowerPoint) en distintos niveles de instrucción. Soporte y capacitación a domicilio para empresas. Asesoramiento tecnológico bajo la filosofía del software libre. También ofrecen contratos anuales de mantenimiento.
- **Administración de Redes y Mantenimiento de Equipos de Computación**: Administración de servidores de red. Instalación de servicios en GNU/Linux, firewall, proxy, correo, LDAP, respaldo, CUPS, NFS. Diagnostico de red y equipos de computación, mantenimiento y asesoría de equipos de computación, instalación de sistemas operativos libre o propietario. Recuperación de información pérdida, eliminación de virus y spyware.
- **Diseño de Bases de Datos y Desarrollo de Software**: Diseño de bases de datos relacionales. conversión y migración de bases de datos Oracle a PostgreSQL. Asesoría en manejadores de base de datos PostgreSQL y MySQL. Desarrollo de aplicaciones web o de escritorio, utilizando la metodología del Proceso Unificado y el lenguaje UML. Software hecho a la medida del cliente. Programación en Python, PHP, C++ y JavaScript.

- **Venta de Equipos Personal y Corporativo**: Ofrecen una variedad de equipos de computación de escritorio, portátiles, servidores; equipos de red, impresoras, equipos para puntos de venta, consumibles y accesorios para computadoras.

2.3. El Cooperativismo [2]

El Cooperativismo es una doctrina económico social basada en la conformación de asociaciones económicas cooperativistas en las que todos los miembros son beneficiarios de su actividad según el trabajo que aportan a la actividad de la cooperativa. El trabajo que aporta cada socio de una cooperativa se convierte en beneficio para él mismo y, para todo el grupo de trabajo conformado por todos los miembros de la cooperativa.

El Cooperativismo promueve la libre asociación de individuos y familias con intereses comunes. Su intención, es poder construir una empresa en la que todos tienen igualdad de derechos y en las que el beneficio obtenido se reparte entre sus asociados según el trabajo que aporta cada uno de los miembros.

A nivel Económico su objetivo es la reducción del precio de venta, de compra, mejorar la calidad de vida de los participantes, etc.

Como organización social, el cooperativismo promueve la gestión democrática y la eliminación del beneficio capitalista. Esto, además de defender el trabajo como factor generador de la riqueza.

2.3.1. Origen del Cooperativismo

Los primeros registros del cooperativismo datan de mitad de siglo XIX cuando en Inglaterra nacen las primeras cooperativas de consumo. Posteriormente, en Francia se originan las de producción. Las de créditos nacen en Alemania. Luego, a finales de

ese mismo siglo, el cooperativismo llegó a Latinoamérica con los torrentes de inmigrantes europeos a Argentina, Uruguay y Sur de Brasil. En ese momento el movimiento se bifurcaba en dos grandes tendencias ideológicas. Una, la utópica representada por Charles Gide y su "República Cooperativa". Otra, la pragmática, que se fundamentaba en las cajas Raiffeisen de Alemania y en la herencia de la cooperativa de Rochadle, modelos estos que buscaban mejorar las condiciones de vida de los trabajadores a través de cooperativas de consumo, ahorro y crédito.

En Venezuela el cooperativismo llegó a partir de 1930 por lo que se hace un sistema aún nuevo, además de que, básicamente se ha limitado al ahorro y crédito y al consumo como áreas de producción.

2.3.2. La Cooperativa

La Cooperativa es una empresa de producción, obtención, consumo o crédito de participación libre y democrática, conformada por personas que persiguen un objetivo en común económico y social en donde la participación de cada socio, en el beneficio, es determinado por el trabajo incorporado al objetivo común y no por la cantidad de dinero que haya aportado.

La Cooperativa, a diferencia de las compañías anónimas, es una sociedad de personas, no de capitales. Se fundamenta en la igualdad de derechos de sus integrantes en cuanto a la gestión social. Además, las cooperativas reparten sus excedentes o ganancias en función de la actividad realizada por sus asociados en el logro del propósito común. En cambio, en una empresa mercantil, la ganancia se distribuye entre los socios de manera proporcional al capital económico que cada uno aportó.

2.3.3. Tipos de Cooperativa

Las empresas cooperativas se clasifican según la actividad para la que fueron creadas. Así tenemos que existen:

- Cooperativas de Producción de Bienes y Servicios
- Cooperativas de Consumo de Bienes y Servicios
- Cooperativas de Ahorro y Crédito
- Cooperativas Mixtas

2.3.3.1. Cooperativas de Producción de Bienes y Servicios

Consisten en agrupaciones de personas de un mismo oficio o con un fin común, que por medios propios producen ciertos artículos vendiéndolos directamente y distribuyéndose entre ellos las ganancias. Este tipo de Cooperativas tienen como meta principal la producción de bienes o prestación de servicios, tales como: la producción industrial o artesanal, la producción agropecuaria o pesquera, la producción minera, transporte colectivo o de carga, producción de diversos servicios del hogar, mantenimiento, reparaciones menores y mayores, salud, hogares de cuidados de infantes y, en general, de todas aquellas actividades que son demandadas por otras personas o instituciones. SAIT, RL entra dentro de esta categoría.

2.3.3.2. Cooperativas de Consumo de Bienes y Servicios

Existen las Cooperativas de obtención de bienes y servicios. Aquí entran las Cooperativas de Consumo y las Cooperativas de Ahorro y Préstamo. La primera tiene como objetivo satisfacer mejor y más económicamente las necesidades de sus miembros. Esto, se da por medio de la mejora de los servicios de compra y venta de artículos de primera necesidad.

2.3.3.3. Cooperativas de Ahorro y Crédito

Son aquellas que tienen por objeto fundamental fomentar el ahorro y otorgar préstamos a sus asociados con los recursos aportados por los mismos, a un interés muy bajo, con el fin de eliminar los altos costos que representan los créditos otorgados por los bancos comerciales.

2.3.3.4. Cooperativas Mixtas

Persiguen dos objetivos a saber: la producción de bienes y servicios para terceros y la obtención de bienes y servicios para sus asociados. Las Cooperativas Mixtas deben su nombre a la posibilidad de tener, al mismo tiempo, dos o más de los perfiles que se han descrito anteriormente.

2.4. Proceso Unificado De Desarrollo De Software [17]

El Proceso Unificado es un proceso de desarrollo de software, es decir un conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema software. El Proceso Unificado es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas software, para diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes tipos de aptitud y diferentes tamaños de proyectos; utiliza el Lenguaje Unificado de Modelado (UML) para preparar todos los esquemas de un sistema software.

El Proceso Unificado se caracteriza por ser:

- **Dirigido por casos de usos:** Un caso de uso es un fragmento de funcionalidad del sistema, es decir representa los requisitos funcionales del sistema. Basados en el modelo de casos de uso, los desarrolladores de software crean una serie de modelos de diseño e implementación que llevan a cabo los casos de uso.

- **Centrado en la arquitectura:** La arquitectura del software incluye los aspectos estáticos y dinámicos más significativos, surge de las necesidades de la empresa, como la perciben los usuarios y los inversores y se refleja en los casos de uso. También se ve influida por factores como la plataforma en la que tiene que funcionar el software (Arquitectura hardware, sistema operativo, sistema gestor de bases de datos, etc.). La arquitectura es una vista general de las características más resaltantes del sistema. Es importante destacar que la arquitectura debe diseñarse para permitir que el sistema evolucione.
- **Iterativo e Incremental:** Se plantea la división del trabajo en mini proyectos, los cuales se irán desarrollando de manera progresiva (incremental). Cada mini proyecto es una iteración que resulta en un incremento. Las iteraciones hacen referencia a pasos del flujo del trabajo y los incrementos al crecimiento del producto

2.4.1. Ciclo de Vida del Proceso Unificado

El proceso unificado se repite a lo largo de una serie de ciclos que constituyen la vida de un sistema, como se muestra la Figura 2.1 Cada ciclo consta de cuatro fases: inicio, elaboración, construcción y transición.

2.4.1.1. Fase de Inicio

El objetivo de la fase de inicio es desarrollar el análisis de negocio hasta el punto necesario para justificar la puesta en marcha del proyecto. La fase de inicio no es un estudio completo del sistema propuesto, sino que en ella se busca el porcentaje de casos de uso necesarios para fundamentar el análisis de negocio inicial. En esta fase se identifican los requisitos funcionales del sistema y priorizan los riesgos más importantes, se planifica en detalle la fase de elaboración. En pocas palabras sus funciones son:

- Describir o esbozar propuestas de la arquitectura del sistema, y en especial de aquellas partes del sistema que son nuevas, arriesgadas o difíciles.
- Delimitar el ámbito del sistema propuesto, es decir, definir los límites del sistema y empezar a identificar las interfaces con sistemas relacionados que están fuera de los límites.
- Identificar los riesgos críticos, es decir, los que afectan a la capacidad de construir el sistema, y determinar si podemos encontrar una forma de mitigarlos.

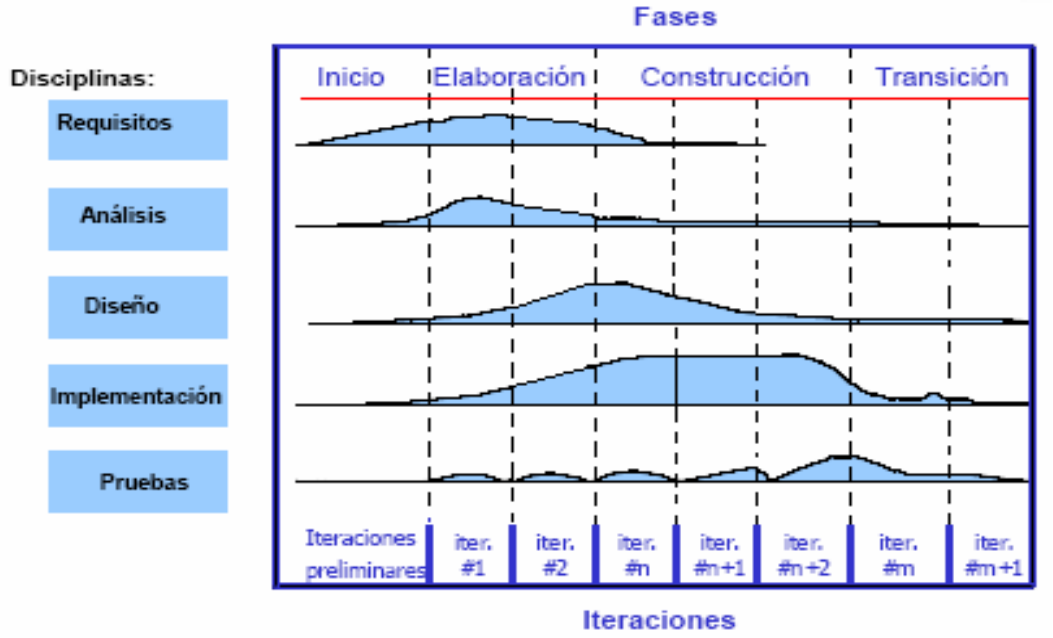


Figura 2.1. Flujos de trabajo y modelos asociados del Proceso Unificado – Fuente: [17]

2.4.1.2. Fase de Elaboración

Se especifican en detalle la mayoría de los casos de uso del producto y se diseña la arquitectura del sistema. La relación entre la arquitectura del sistema y el propio sistema. Los requisitos y la arquitectura (en el análisis, el diseño y la

implementación) representan el grueso del esfuerzo en las fases de inicio y la elaboración., Los principales objetivos son:

- Crear la línea base de la arquitectura que cubre la funcionalidad del sistema significativa arquitectónicamente y las características importantes para los actores involucrados. Esta línea base consiste en los artefactos de los modelos, la descripción de la arquitectura y en una implementación ejecutable de esta.
- Continuar con la observación de los posibles riesgos. Se identifican los riesgos que podrían perturbar el desarrollo del proyecto.
- Especificar los niveles por atributos de calidad.
- Recopilar la mayor parte de los requisitos que quedan pendientes para casos de uso, para completar aproximadamente el 80% de los requisitos funcionales.
- Completar los detalles del plan de proyecto.

2.4.1.3. Fase de Construcción

El objetivo general de la fase viene indicado por su tarea fundamental: la capacidad de operación inicial. Es decir, el producto listo para pruebas. En esta fase, la línea base de la arquitectura crece hasta convertirse en el sistema completo. La descripción evoluciona hasta convertirse en un producto preparado para ser entregado a la comunidad de usuarios. Al final de esta fase, el producto contiene todos los casos de uso que la dirección y el cliente han acordado.

Generalmente esta fase requiere de un mayor número de iteraciones que las fases anteriores. En muchos casos el tener un trabajo de requisitos, análisis y diseño pobre hace que la construcción genere varias iteraciones, debido a esto la construcción puede consumir una cantidad de tiempo relativamente grande. Una de las grandes ventajas de construir un software utilizando un enfoque que use múltiples fases y un desarrollo iterativo e incremental es permitir la asignación de recursos y de

tiempo a lo largo de ciclo de vida. Ente las actividades de la fase de construcción, se tienen:

- La extensión de la identificación, descripción y realización de casos de uso a todos los casos de uso.
- La finalización del análisis.
- El mantenimiento de la integridad de la arquitectura.
- La monitorización de los riesgos críticos y significativos arrastrados desde las dos primeras fases, y su mitigación si se materializan

2.4.1.4. Fase de Transición

Cubre el periodo durante el cual el producto se convierte en versión beta. En la versión beta un número de usuarios con experiencia prueban el producto e informan de defectos y deficiencias. Los desarrolladores corrigen el problema e incorporan algunas de las mejores sugerencias en una versión general dirigida a la totalidad de la comunidad de usuarios.

- Preparar las actividades así como la preparación del lugar de pruebas.
- Preparar los manuales y otros documentos para la entrega del producto.
- Ajustar el software, al entorno del usuario.
- Corregir defectos encontrados a lo largo de las pruebas realizadas.

2.4.2. Flujos de Trabajo y Modelos Asociados

- **Requi**sitos: Modelo de casos de uso, con todos los casos de uso y su relación con los usuarios.
- Análisis: Modelo de análisis, con dos propósitos: refinar los casos de uso con más detalle y establecer la asignación inicial de funcionalidad del sistema a un conjunto de objetos que proporcionan el comportamiento.

- Diseño: Modelo de análisis, con dos propósitos: refinar los casos de uso con más detalle y establecer la asignación inicial de funcionalidad del sistema a un conjunto de objetos que proporcionan el comportamiento. Modelo de despliegue, que define los nodos físicos (ordenadores) y la correspondencia de los componentes con esos nodos.
- Implementación: Modelo de implementación, que incluye componentes (que representan al código fuente) y la correspondencia de las clases con los componentes.
- Prueba: Modelo de prueba, que describe los casos de prueba que verifican los casos de uso.

2.5. Lenguaje Unificado De Modelado (UML)

El Lenguaje de Modelamiento Unificado (UML - Unified Modeling Language) es un lenguaje gráfico para visualizar, especificar y documentar cada una de las partes que comprende el desarrollo de software. UML entrega una forma de modelar cosas conceptuales como lo son procesos de negocio y funciones de sistema, además de cosas concretas como lo son escribir clases en un lenguaje determinado, esquemas de base de datos y componentes de software reusables. [13]

2.5.1. Características de UML [13]

- Es una especificación de notación orientada a objetos. Se basa en las anteriores especificaciones BOOCH, RUMBAUGH y COAD-YOURDON. Divide cada proyecto en un número de diagramas que representan las diferentes vistas del proyecto. Estos diagramas juntos son los que representa la arquitectura del proyecto.
- Intenta solucionar el problema de propiedad de código que se da con los desarrolladores, al implementar un lenguaje de modelado común para todos los desarrollos se crea una documentación también común, que cualquier

desarrollador con conocimientos de UML será capaz de entender, independientemente del lenguaje utilizado para el desarrollo.

- Es ahora un estándar, no existe otra especificación de diseño orientado a objetos, ya que es el resultado de las tres opciones existentes en el mercado. Su utilización es independiente del lenguaje de programación y de las características de los proyectos, ya que UML ha sido diseñado para modelar cualquier tipo de proyectos, tanto informáticos como de arquitectura, o de cualquier otro ramo.
- Permite la modificación de todos sus miembros mediante estereotipos y restricciones. Un estereotipo nos permite indicar especificaciones del lenguaje al que se refiere el diagrama de UML. Una restricción identifica un comportamiento forzado de una clase o relación, es decir mediante la restricción estamos forzando el comportamiento que debe tener el objeto al que se le aplica

2.5.2. Vistas y Diagramas UML [9]

La explicación se basará en los diagramas, en lugar de en vistas o anotación, ya que son estos la esencia de UML. Cada diagrama usa la anotación pertinente y la suma de estos diagramas crean las diferentes vistas. Las vistas existentes en UML son:

- Vista casos de uso: Se forma con los diagramas de casos de uso, colaboración, estados y actividades.
- Vista de diseño: Se forma con los diagramas de clases, objetos, colaboración, estados y actividades.
- Vista de procesos: Se forma con los diagramas de la vista de diseño. Recalcando las clases y objetos referentes a procesos.

- Vista de implementación: Se forma con los diagramas de componentes, colaboración, estados y actividades.
- Vista de despliegue: Se forma con los diagramas de despliegue, interacción, estados y actividades.
- Se dispone de dos tipos diferentes de diagramas los que dan una vista estática del sistema y los que dan una visión dinámica.

Los diagramas estáticos son:

- **Diagrama de Clases**: muestra las clases, interfaces, colaboraciones y sus relaciones. Son los más comunes y dan una vista estática del proyecto.
- **Diagrama de Objetos**: Es un diagrama de instancias de las clases mostradas en el diagrama de clases. Muestra las instancias y como se relacionan entre ellas. Se da una visión de casos reales.
- **Diagrama de Componentes**: Muestran la organización de los componentes del sistema. Un componente se corresponde con una o varias clases, interfaces o colaboraciones.
- **Diagrama de Despliegue**.: Muestra los nodos y sus relaciones. Un nodo es un conjunto de componentes. Se utiliza para reducir la complejidad de los diagramas de clases y componentes de un gran sistema. Sirve como resumen e índice.
- **Diagrama de Casos de Uso**: Muestran los casos de uso, actores y sus relaciones. Muestra quien puede hacer que y relaciones existen entre acciones (casos de uso). Son muy importantes para modelar y organizar el comportamiento del sistema.

Lo diagramas dinámicos son:

- **Diagrama de Secuencia**, **Diagrama de Colaboración**: Muestran a los diferentes objetos y las relaciones que pueden tener entre ellos, los mensajes que se envían entre ellos. Son dos diagramas diferentes, que se puede pasar de uno a otro sin perdida de información, pero que nos dan puntos de vista diferentes del sistema. En resumen, cualquiera de los dos es un Diagrama de Interacción.
- **Diagrama de Estados**: muestra los estados, eventos, transiciones y actividades de los diferentes objetos. Son útiles en sistemas que reaccionen a eventos.
- **Diagrama de Actividades**: Es un caso especial del diagrama de estados. Muestra el flujo entre los objetos. Se utilizan para modelar el funcionamiento del sistema y el flujo de control entre objetos.

2.6. Programación Orientada A Objetos [5]

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

La programación orientada a objetos introduce nuevos conceptos, que superan y amplían conceptos antiguos ya conocidos. Entre ellos destacan los siguientes:

- **Objeto**: entidad provista de un conjunto de propiedades o atributos (datos) y de comportamiento o funcionalidad ("métodos"). Corresponden a los objetos reales del mundo que nos rodea, o a objetos internos del sistema (del programa).

- **Clase**: definiciones de las propiedades y comportamiento de un tipo de objeto concreto. La instanciación es la lectura de estas definiciones y la creación de un objeto a partir de ellas.
- **Método**: algoritmo asociado a un objeto (o a una clase de objetos), cuya ejecución se desencadena tras la recepción de un mensaje. Desde el punto de vista del comportamiento, es lo que el objeto puede hacer.
- **Evento**: un suceso en el sistema (tal como una interacción del usuario con la máquina, o un mensaje enviado por un objeto). El sistema maneja el evento enviando el mensaje adecuado al objeto pertinente.
- **Mensaje**: una comunicación dirigida a un objeto, que le ordena que ejecute uno de sus métodos con ciertos parámetros asociados al evento que lo generó.
- **Propiedad o atributo**: contenedor de un tipo de datos asociados a un objeto (o a una clase de objetos), que hace los datos visibles desde fuera del objeto, y cuyo valor puede ser alterado por la ejecución de algún método.
- **Estado interno**: es una propiedad invisible de los objetos, que puede ser únicamente accedida y alterada por un método del objeto, y que se utiliza para indicar distintas situaciones posibles para el objeto (o clase de objetos).

2.6.1. Características de la Programación Orientada a Objetos

- **Abstracción**: cada objeto en el sistema sirve como modelo de un agente abstracto que puede realizar trabajo, informar y cambiar su estado, y comunicarse con otros objetos en el sistema sin revelar cómo se implementan estas características. Los procesos, las funciones o los métodos pueden también ser abstraídos y cuando lo están, una variedad de técnicas son requeridas para ampliar una abstracción.
- **Encapsulamiento**: también llamado "ocultación de la información". Cada objeto está aislado del exterior, es un módulo natural, y cada tipo de objeto

expone una interfaz a otros objetos que específica cómo pueden interactuar con los objetos de la clase. El aislamiento protege a las propiedades de un objeto contra su modificación por quien no tenga derecho a acceder a ellas, solamente los propios métodos internos del objeto pueden acceder a su estado.

- **Polimorfismo**: comportamientos diferentes, asociados a objetos distintos, pueden compartir el mismo nombre, al llamarlos por ese nombre se utilizará el comportamiento correspondiente al objeto que se esté usando. O dicho de otro modo, las referencias y las colecciones de objetos pueden contener objetos de diferentes tipos, y la invocación de un comportamiento en una referencia producirá el comportamiento correcto para el tipo real del objeto referenciado.
- **Herencia**: las clases no están aisladas, sino que se relacionan entre sí, formando una jerarquía de clasificación. Los objetos heredan las propiedades y el comportamiento de todas las clases a las que pertenecen. La herencia organiza y facilita el polimorfismo y el encapsulamiento permitiendo a los objetos ser definidos y creados como tipos especializados de objetos preexistentes. Estos pueden compartir (y extender) su comportamiento sin tener que reimplementar su comportamiento.

2.7. Base De Datos [14]

Una base de datos es una **c**olección de datos organizados bajo normas establecidas en un modelo seleccionado. La base de datos es un conjunto de información almacenada bajo la estructura diseñada e implementada en un sistema manejador de base de datos y relacionada con un asunto o con una finalidad.

La gestión de los datos implica la definición de estructuras para almacenar información y los mecanismos para manipulación de la información.

2.7.1. Lenguaje Definición de Datos

Es un conjunto de declaraciones o definiciones que permiten expresar las especificaciones del esquema de la base de datos. A partir de estas se genera el Diccionario de Datos. Igualmente permite:

- Determinar la estructura de almacenamiento y los métodos de acceso.
- Crear, modificar y eliminar las bases de datos y las tablas que las conforman.
- Definir, modificar y eliminar índices y reglas de integridad.

2.7.2. Lenguaje de Manipulación de Datos

A diferencia del anterior este tiene estrecha relación con las operaciones que los usuarios realizan sobre los datos almacenados. Entre las operaciones que se pueden realizar tenemos: Recuperación o consulta de datos, inserción, modificación y eliminación de los datos.

2.7.3. Modelo Relacional

El modelo relacional fue expuesto por el Dr. Codd en la primera mitad de los años 70, es un modelo de datos basado en la lógica de predicado y en la teoría de conjuntos.

Este modelo considera la base de datos como una colección de relaciones. De manera simple, una relación representa una tabla, en que cada fila representa una colección de valores que describen una entidad del mundo real. Cada fila se denomina tupla o registro y cada columna campo.

Entre las ventajas de este modelo están:

- Garantiza herramientas para evitar la duplicidad de registros, a través de campos claves o llaves.
- Garantiza la integridad referencial: Así al eliminar un registro elimina todos los registros relacionados dependientes.
- Favorece la normalización por ser más comprensible y aplicable.
- Los sistemas manejadores de bases de datos relacionales cuentan con el mayor número de instalaciones comerciales en el mundo de hoy.

2.7.3.1. Características del Modelo Relacional

Son estructuras de datos simples. Consiste en tablas de dos dimensiones donde los elementos son ítem de datos. Esto permite un alto grado de independencia de la representación física de los datos.

- El modelo relacional provee una sólida fundamentación para la consistencia de los datos. El diseño de la bases de datos es asistido por los procesos de normalización que elimina las anomalías en los datos.
- El modelo relacional permite la manipulación de las relaciones. Esta característica puede ser encargada a potentes lenguajes no procedimentales basados en la teoría (álgebra relacional) o en la lógica (cálculo relacional)
- Cada tabla es única dentro de la base de datos.
- Las columnas corresponden a los atributos de la tabla, los cuales tienen un nombre único, estas columnas se encuentran ordenadas. Las filas corresponden a las tuplas o registros de la tabla. Cada tabla puede tener mínimo 1 columna y 0 filas. (Figura 2.2)

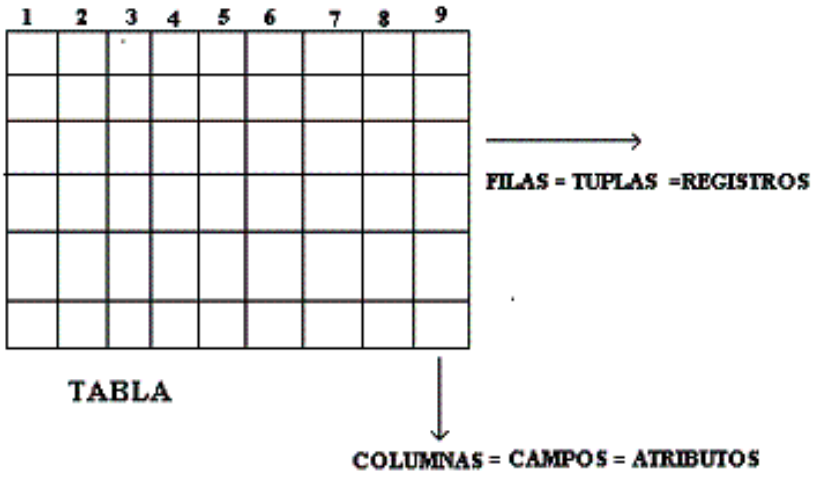


Figura 2.2. Tabla de un modelo relacional – Fuente: [14]

2.8. Conceptos de arquitectura Cliente/Servidor [3]

La arquitectura cliente/servidor es la tecnología que proporciona al usuario final el acceso transparente a las aplicaciones, datos, servicios de cómputo o cualquier otro recurso del grupo de trabajo y/o, a través de la organización, en múltiples plataformas. El modelo soporta un medio ambiente distribuido en el cual los requerimientos de servicio hechos por estaciones de trabajo inteligentes o "clientes", resultan en un trabajo realizado por otros computadores llamados servidores. Es una arquitectura de procesamientos cooperativos donde uno de los componentes pide servicios a otro.

Las características más importantes que se distinguen son:

- **Orientado a servicios**. El servidor los ofrece y el cliente los consume.
- **Compartición de recursos**. Servicios ofrecidos a muchos clientes. Un servidor puede atender muchos clientes que solicitan esos servicios.

- **Transparencia de ubicación**. El servidor es un proceso que puede residir en el mismo aparato que el cliente o en un aparato distinto a lo largo de una red. Un programa puede ser un servidor en un momento y convertirse en un cliente posteriormente.
- **Mezcla e igualdad**. Tal vez de las más importantes ventajas de este paradigma. Una aplicación cliente/servidor, idealmente es independiente del hardware y de sistemas operativos; mezclando e igualando estas plataformas.
- **Interacción a través de mensajes**, para envío y respuesta de servicios.
- **Servicios encapsulados**, exponiendo los servicios a través de interfaces, lo que facilita la sustitución de servidores son afectar los clientes; permitiendo a la vez una fácil escalabilidad.

2.9. C++ Builder [10]

C++Builder es un IDE (Entorno de Desarrollo Integrado) en lenguaje C++ para Windows inicialmente de la empresa Borland, actualmente de su ex-filial CodeGear.

C++Builder permite obtener a los desarrolladores la potencia de los lenguajes y librerías C y C++, a la vez que también les ofrece la productividad del desarrollo rápido y visual de aplicaciones. El entorno de desarrollo de C++Builder incluye un avanzado editor y depurador de código, también cuenta con la VCL (Visual Component Library) que es un sistema de manejo de librerías que cuenta con un sinfín de componentes visuales; y por último posee un potente compilador que soporta hasta los nuevos estándares C++0x. Una gran ventaja de la VCL es que permite usar tanto las potentes librerías que incluye como también las librerías en C o C++ de terceros para crear aplicaciones nativas Windows para todo tipo de sectores. Todas las tecnologías mencionadas permiten a los programadores desarrollar rápidamente interfaces de usuario y aplicaciones de gestión de base de datos con conectividad a base de datos heterogéneas.

2.9.1. Características de C++ Builder Versión 11

- Desarrollo con VCL para Windows Vista incluyendo el famoso Escritorio Aero y servicio de apoyo a las API de Windows Vista.
- Desarrollo con VCL para la Web incluyendo soporte para AJAX y PHP, que gracias a los componentes visuales permite crear aplicaciones web sin la necesidad de tener un amplio conocimiento de HTML, JavaScript o CSS por lo que podrá centrarse directamente sobre el código y la interfaz de usuario.
- Arquitectura DBX 4 que racionaliza la conectividad con las bases de datos y soporta las últimas versiones de SQL.
- Actualización de código abierto para la librería de conectores Indy la cual está incluida dentro de la suite de protocolos de Internet.
- Ampliación del ANSI C a través de las librerías de apoyo Dinkumware.
- Función de apoyo a DUnit integrado dentro del código para mejorar las pruebas de unidad y los ensayos normalizados, proveyendo así de mayor estabilidad a la aplicación.

2.10. Lenguaje De Consultas Estructurado (Sql) [11]

El Lenguaje de Consulta Estructurado SQL (Structured Query Language) es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones sobre las mismas. Aúna características del álgebra y el cálculo relacional permitiendo lanzar consultas con el fin de recuperar información de interés de una base de datos, de una forma sencilla.

El SQL es un lenguaje de acceso a bases de datos que explota la flexibilidad y potencia de los sistemas relacionales permitiendo gran variedad de operaciones sobre los mismos. Es un lenguaje declarativo de alto nivel o de no procedimiento, que gracias a su fuerte base teórica y su orientación al manejo de conjuntos de registros, y

no a registros individuales, permite una alta productividad en codificación. De esta forma una sola sentencia puede equivaler a uno o más programas que utilizasen un lenguaje de bajo nivel orientado a registro.

SQL proporciona una rica funcionalidad más allá de la simple consulta (o recuperación) de datos. Asume el papel de lenguaje de definición de datos (LDD), lenguaje de definición de vistas (LDV) y lenguaje de manipulación de datos (LMD). Además permite la concesión y denegación de permisos, la implementación de restricciones de integridad y controles de transacción, y la alteración de esquemas. Las primeras versiones del SQL incluían funciones propias de lenguaje de definición de almacenamiento (LDA) pero fueron suprimidas en los estándares más recientes con el fin de mantener el lenguaje sólo a nivel conceptual y externo.

El SQL permite fundamentalmente dos modos de uso:

- Un uso interactivo, destinado principalmente a los usuarios finales avanzados u ocasionales, en el que las diversas sentencias SQL se escriben y ejecutan en línea de comandos, o un entorno semejante.
- Un uso integrado, destinado al uso por parte de los programadores dentro de programas escritos en cualquier lenguaje de programación anfitrión. En este caso el SQL asume el papel de sublenguaje de datos.

En el caso de hacer un uso embebido del lenguaje podemos utilizar dos técnicas alternativas de programación. En una de ellas, en la que el lenguaje se denomina SQL estático, las sentencias utilizadas no cambian durante la ejecución del programa. En la otra, donde el lenguaje recibe el nombre de SQL dinámico, se produce una modificación total o parcial de las sentencias en el transcurso de la ejecución del programa. La utilización de SQL dinámico permite mayor flexibilidad y mayor complejidad en las sentencias, pero como contra punto obtenemos una eficiencia

menor y el uso de técnicas de programación más complejas en el manejo de memoria y variables.

2.11. Manejador De Base De Datos Mysql

MySQL es un sistema de gestión de base de datos relacional, multihilo y multiusuario con más de seis millones de instalaciones. MySQL AB - desde enero de 2008 una subsidiaria de Sun Microsystems y ésta a su vez de Oracle Corporation desde abril de 2009 - desarrolla MySQL como software libre en un esquema de licenciamiento dual.

2.11.1. Breve Historia de MySQL [4]

SQL (Lenguaje de Consulta Estructurado) fue comercializado por primera vez en 1981 por IBM, el cual fue presentado a ANSI y desde entonces ha sido considerado como un estándar para las bases de datos relacionales. Desde 1986, el estándar SQL ha aparecido en diferentes versiones como por ejemplo: SQL: 92, SQL: 99, SQL: 2003. MySQL es una idea originaria de la empresa opensource MySQL AB establecida inicialmente en Suecia en 1995 y cuyos fundadores son David Axmark, Allan Larsson, y Michael "Monty" Widenius. El objetivo que persigue esta empresa consiste en que MySQL cumpla el estándar SQL, pero sin sacrificar velocidad, fiabilidad o usabilidad.

Michael Widenius en la década de los 90 trató de usar mSQL para conectar las tablas usando rutinas de bajo nivel ISAM, sin embargo, mSQL no era rápido y flexible para sus necesidades. Esto lo llevó a crear una API SQL denominada MySQL para bases de datos muy similar a la de mSQL pero más portable.

La procedencia del nombre de MySQL no es clara. Desde hace más de 10 años, las herramientas han mantenido el prefijo My. También, se cree que tiene relación con el nombre de la hija del cofundador Monty Widenius quien se llama My.

Por otro lado, el nombre del delfín de MySQL es Sakila y fue seleccionado por los fundadores de MySQL AB en el concurso “Name the Dolphin”. Este nombre fue enviado por Ambrose Twebaze, un desarrollador de Open source Africano, derivado del idioma SiSwate, el idioma local de Swazilandia y corresponde al nombre de una ciudad en Arusha, Tanzania, cerca de Uganda la ciudad origen de Ambrose.

2.11.2. Características de MySQL [4]

Algunas de sus principales características son:

- Interioridades y portabilidad
  - Escrito en C y en C++
  - Probado con un amplio rango de compiladores diferentes
  - Funciona en diferentes plataformas.
  - Usa GNU Automake, Autoconf, y Libtool para portabilidad.
  - APIs disponibles para C, C++, Eiffel, Java, Perl, PHP, Python, Ruby, y Tcl.
  - Uso completo de multi-threaded mediante threads del kernel. Pueden usarse fácilmente multiple CPUs si están disponibles.
  - Proporciona sistemas de almacenamiento transaccional y no transaccional.
  - Usa tablas en disco B-tree (MyISAM) muy rápidas con compresión de índice.
  - Relativamente sencillo de añadir otro sistema de almacenamiento. Esto es útil si desea añadir una interfaz SQL para una base de datos propia.

- Un sistema de reserva de memoria muy rápido basado en threads.
- Joins muy rápidos usando un multi-join de un paso optimizado.
- Tablas hash en memoria, que son usadas como tablas temporales.
- Las funciones SQL están implementadas usando una librería altamente optimizada y deben ser tan rápidas como sea posible. Normalmente no hay reserva de memoria tras toda la inicialización para consultas.
- El servidor está disponible como un programa separado para usar en un entorno de red cliente/servidor. También está disponible como biblioteca y puede ser incrustado (linkado) en aplicaciones autónomas. Dichas aplicaciones pueden usarse por sí mismas o en entornos donde no hay red disponible.

- Tipos de columnas
  - Diversos tipos de columnas: enteros con/sin signo de 1, 2, 3, 4, y 8 bytes de longitud, FLOAT, DOUBLE, CHAR, VARCHAR, TEXT, BLOB, DATE, TIME, DATETIME, TIMESTAMP, YEAR, SET, ENUM, y tipos espaciales OpenGIS.
  - Registros de longitud fija y longitud variable.
- Sentencias y funciones
  - Soporte completo para operadores y funciones en las cláusulas de consultas SELECT y WHERE.
  - Soporte completo para las cláusulas SQL GROUP BY y ORDER BY. Soporte de funciones de agrupación (COUNT(), COUNT(DISTINCT...), AVG(), STD(), SUM(), MAX(), MIN(), y GROUP_CONCAT()).
  - Soporte para LEFT OUTER JOIN y RIGHT OUTER JOIN cumpliendo estándares de sintaxis SQL y ODBC.
  - Soporte para alias en tablas y columnas como lo requiere el estándar SQL.

- DELETE, INSERT, REPLACE, y UPDATE devuelven el número de filas que han cambiado (han sido afectadas). Es posible devolver el número de filas que serían afectadas usando un flag al conectar con el servidor.
- El comando específico de MySQL SHOW puede usarse para obtener información acerca de la base de datos, el motor de base de datos, tablas e índices. El comando EXPLAIN puede usarse para determinar cómo el optimizador resuelve una consulta.
- Los nombres de funciones no colisionan con los nombres de tabla o columna. Por ejemplo, ABS es un nombre válido de columna. La única restricción es que para una llamada a una función, no se permiten espacios entre el nombre de función y el '(' a continuación.
- Puede mezclar tablas de distintas bases de datos en la misma consulta (como en MySQL 3.22).

Seguridad

- Un sistema de privilegios y contraseñas que es muy flexible y seguro, y que permite verificación basada en el host. Las contraseñas son seguras porque todo el tráfico de contraseñas está encriptado cuando se conecta con un servidor.

Escalabilidad y límites

- Soporte a grandes bases de datos. Usamos MySQL Server con bases de datos que contienen 50 millones de registros. También conocemos a usuarios que usan MySQL Server con 60.000 tablas y cerca de 5.000.000.000.000 de registros.
- Se permiten hasta 64 índices por tabla (32 antes de MySQL 4.1.2). Cada índice puede consistir desde 1 hasta 16 columnas o partes de columnas. El máximo ancho de límite son 1000 bytes (500 antes de MySQL 4.1.2).Un

índice puede usar prefijos de una columna para los tipos de columna CHAR, VARCHAR, BLOB, o TEXT.

- Conectividad
  - Los clientes pueden conectar con el servidor MySQL usando sockets TCP/IP en cualquier plataforma. En sistemas Windows de la familia NT (NT, 2000, XP o 2003), los clientes pueden usar named pipes para la conexión. En sistemas Unix, los clientes pueden conectar usando ficheros socket Unix.
  - En MySQL 5.0, los servidores Windows soportan conexiones con memoria compartida si se inicializan con la opción --shared-memory. Los clientes pueden conectar a través de memoria compartida usando la opción --protocol=memory.
  - La interfaz para el conector ODBC (MyODBC) proporciona a MySQL soporte para programas clientes que usen conexiones ODBC (Open Database Connectivity). Por ejemplo, puede usar MS Access para conectar al servidor MySQL. Los clientes pueden ejecutarse en Windows o Unix. El código fuente de MyODBC está disponible. Todas las funciones para ODBC 2.5 están soportadas, así como muchas otras.
  - La interfaz para el conector J MySQL proporciona soporte para clientes Java que usen conexiones JDBC. Estos clientes pueden ejecutarse en Windows o Unix. El código fuente para el conector J está disponible.
- Localización
  - El servidor puede proporcionar mensajes de error a los clientes en muchos idiomas.
  - Soporte completo para distintos conjuntos de caracteres, incluyendo latin1 (ISO-8859-1), german, big5, ujis, y más. Por ejemplo, los

caracteres escandinavos 'â', 'ä' y 'ö' están permitidos en nombres de tablas y columnas. El soporte para Unicode está disponible

- Todos los datos se guardan en el conjunto de caracteres elegido. Todas las comparaciones para columnas normales de cadenas de caracteres son case-insensitive.
- La ordenación se realiza acorde al conjunto de caracteres elegido (usando colación Sueca por defecto). Es posible cambiarla cuando arranca el servidor MySQL. Para ver un ejemplo de ordenación muy avanzada, consulte el código Checo de ordenación. MySQL Server soporta diferentes conjuntos de caracteres que deben ser especificados en tiempo de compilación y de ejecución.

- Clientes y herramientas
  - MySQL server tiene soporte para comandos SQL para chequear, optimizar, y reparar tablas. Estos comandos están disponibles a través de la línea de comandos y el cliente mysqlcheck. MySQL también incluye myisamchk, una utilidad de línea de comandos muy rápida para efectuar estas operaciones en tablas MyISAM.

CAPÍTULO III

FASE DE INICIO

3.1. Introducción

Para realizar este proyecto se escogió la metodología del Proceso Unificado, que junto con el lenguaje UML, nos brinda la posibilidad de modelar, documentar y obtener un software de calidad. Esta metodología consta de cuatro fases: Inicio, elaboración, construcción y transición.

Durante este capitulo se aborda la primera fase del proceso unificado: La Fase de Inicio. Esta fase constituye el inicio del desarrollo del sistema de información APLICAD (Aplicación de Control Administrativo), nombre con el cual será referenciado a lo largo de la realización de este trabajo.

En la fase de inicio se da comienzo al Proceso Unificado de Desarrollo de Software; es donde se pone en marcha el proyecto a partir de una idea inicial, definiendo el dominio del sistema y esbozando las arquitecturas candidatas; así como también la captura de requisitos, para el análisis y diseño, indicando así que en la fase de inicio se identifican y prevalecen los riesgos más importantes presentes en el proyecto.

3.2. Requisitos

Un requisito es una característica de diseño, una propiedad o un comportamiento de un sistema. Normalmente la mayoría de los sistemas interactúa con muchos usuarios, los cuales solo conocen la superficie del mismo, por lo tanto no saben como podría ser más eficiente la operación que manejan ni cuales son los requisitos ni como especificarlos de forma precisa.

Por tal motivo la captura de los requisitos para el desarrollador de software se hace complicada a la hora de abordar los fundamentos del flujo de trabajo de requisitos. Este flujo de trabajo incluye los siguientes pasos:

3.2.1. Requisitos Candidatos

Los requisitos candidatos de un sistema son un conjunto de ideas que aparecen durante la vida de un sistema, por parte de los clientes, usuarios, analistas y desarrolladores. Para identificar los mismos fue necesario realizar una serie de entrevistas dentro del área de trabajo, para conocer cuales eran las necesidades del cliente. De esta forma se entrevistó tanto a supervisores y técnicos (futuros usuarios del sistema de información). Recopilada toda esta información, se identificaron los siguientes requerimientos:

- Controlar el acceso al sistema.
- Manejar presupuestos de productos y servicios.
- Manejar la compra y venta de productos y servicios.
- Gestionar clientes, proveedores, productos y servicios.
- Consultar información de los servicios realizados.
- Consultar el historial de mantenimiento de un equipo.
- Manejar las cuentas por cobrar a los clientes.
- Generar resúmenes de ventas de productos y servicios

3.2.2. Requisitos Funcionales

Los requisitos funcionales definen las funciones que el sistema será capaz de realizar o los aspectos que cada usuario quiere que el sistema haga, los cuales serán llevados por el analista a casos de uso que representarán los diferentes modos en que

el usuario puede utilizar el sistema. Dentro de los requerimientos funcionales exigidos para APLICAD tenemos los siguientes:

- Gestionar y consultar información de los clientes, proveedores, productos y tarifas de servicio.
- Ajustar precio de los productos y de las tarifas de servicio.
- Cargar, consultar e imprimir las ordenes de servicio de mantenimiento de equipos de computación.
- Cargar, consultar e imprimir presupuestos de venta de equipos y/o servicios.
- Consultar el historial de servicio de cada equipo reparado.
- Cargar, imprimir y consultar la facturación de productos y/o servicios vendidos y gestionar los pagos asociados.
- Consultar las facturas por cobrar.
- Cargar, procesar y consultar la compra de productos a los proveedores y manejar la información referente al modo de pago (cheques utilizados o transferencias realizadas desde las cuentas de la empresa).
- Consultar y procesar el inventario de productos.
- Imprimir consultas de inventario, proveedores, clientes, entre otros.
- Generar reportes significativos como: Resumen de facturación, relación de servicios o ventas en un tiempo determinado, facturas por cobrar, resumen de compras.

3.2.3. Requisitos no Funcionales

Define las características que de una u otra forma puedan limitar el sistema, encontrándose entre ellos:

- El sistema no puede ser manipulado por personal no autorizado por la empresa.
- Los reportes generados deben llevar el logo e identificación de la empresa.
- El sistema podrá ser accedido desde cualquier computador de la red interna de empresa.
- El servidor donde se encontrará el sistema y la base de datos debe cumplir con el requerimiento mínimo para que el sistema funcione.

3.2.4. Contexto del Sistema

El contexto del sistema, se define a través del conocimiento que debe tener el desarrollador del software acerca del entorno donde se desarrollará el sistema.

Para comprender el contexto actual se observó como el personal de SAIT, RL realizaba sus trabajos administrativos dentro de la empresa para comprender el manejo de la información en la misma; así mismo se realizó una serie de entrevistas a los trabajadores de cada departamento para obtener información más detallada de los procesos administrativos.

Entre las actividades más importantes que se realizan está la facturación de los productos y servicios que venden a sus clientes, la cual es llevada a través de hojas de cálculo en Microsoft Excel, creándose un libro nuevo para cada factura que se entrega y teniendo que llevar una cuenta manual del próximo número de factura. Este procedimiento se repite a la hora de hacer los presupuestos. Al momento de la facturar un servicio se recibe el pago si es persona natural o se le da un plazo de espera si es persona jurídica y para este último caso se lleva un control de las cuentas por cobrar con las fechas de vencimiento de las mismas. Todos los pagos son

registrados en una hoja de cálculo con los detalles como forma de pago, fecha y monto.

Otra actividad relevante es el control de los equipos que se reparan en el departamento técnico. Cuando se recibe un equipo del cliente se elabora una orden de servicio a través del programa Microsoft Word, esto se hace copiando manualmente una y otra vez los datos del cliente en una nueva hoja y asignando un número de control manual el cual a veces se pierde o se sobrescribe por error. En esta misma orden de servicio se anotan los detalles del servicio realizado y de los repuestos utilizados (de ser el caso) para su posterior facturación.

Por último está lo relacionado al inventario de productos que se venden a los clientes. El mismo se actualiza a la hora de comprar y vender los productos pero no maneja información detallada de los productos y los seriales de los mismos solo se registran la cantidad y la fecha en que se compran o se venden los productos. El inventario al igual que los otros procesos mencionados se maneja a través de hojas de cálculo y en ocasiones se retarda la entra de alguna compra o venta y no se maneja el stock actualizado.

Al finalizar las actividades anteriormente mencionadas, se estableció el contexto en el cual se creará el software APLICAD (Aplicación de Control Administrativo), logrando determinar la lista de requerimientos de información con los cuales debería cumplir.

3.2.5. Modelo de Dominio

El modelo de dominio, describe los conceptos importantes del contexto como objeto del dominio, el cual representa cosas que existen o eventos que suceden en el entorno del trabajo del sistema.

Muchos de los objetos del dominio o clases pueden obtenerse de una especificación de requisitos o mediante entrevistas con los expertos del dominio.

En el Modelo de Dominio se debe obviar cualquier información acerca del comportamiento interno del sistema y de cómo soportará las funcionalidades esperadas, ya que tiene un fin de tratamiento y comprensión del sistema y no de la solución.

Este modelo se describe mediante diagramas de UML, especialmente diseñados para tal fin, como es el diagrama de clases de análisis.

3.2.6. Listas de Clases (Diccionario de Dominio del Proyecto APLICAD)

- **Recepción**: Personal de SAIT que atiende a los clientes, se encarga de recibir y entregar el equipo al cliente, entregar la factura o presupuesto al cliente y recibir el pago. También anotar los datos del cliente y su requerimiento en una orden de servicio.
- **Técnico**: Personal de SAIT que se encarga de hacer el mantenimiento a los equipos de computación y anotar en la orden de servicio los detalles del trabajo realizado y los repuestos utilizados.
- **Vendedor**: Personal de SAIT que realiza la facturación de productos y servicios, también elabora presupuestos y llevar el control de los pagos del cliente.
- **Comprador**: Personal de SAIT que maneja los referente a la compra de productos y equipos, elaboración de órdenes de compra, y actualizar el inventario de productos.

- **Cliente**: Es cualquier ente que solicita productos y/o servicios a SAIT como: mantenimiento, compra de equipos, instalación de redes, desarrollo de software, capacitación, etc.
- **Proveedor**: Es cualquier ente que provee servicios, suministros, productos y/o equipos a SAIT.
- **Orden de Servicio**: Hoja utilizada en primera instancia para anotar los datos del cliente, equipos entregados por el cliente, requerimiento o problema que presenta el equipo y en segunda instancia para detallar el servicio realizado al equipo y los repuestos utilizados. La facturación de servicios se realizará en base a la información contenida en la orden de servicio.
- **Cotización**: Hoja utilizada para consultar a los proveedores sobre la disponibilidad y precio de un pedido de productos determinado.
- **Orden de Compra**: Es una hoja utilizada para concretar la compra de equipos a un proveedor.
- **Presupuesto**: Es utilizada para informar al cliente de cuanto será el precio de los equipos en el caso de una venta o el costo de un proyecto en caso de un desarrollo de software, instalación de redes o cualquier otro servicio solicitado; antes de la ejecución de los mismos.
- **Factura**: Es el medio por donde se le informa al cliente el precio por los servicios prestados y/o equipos vendidos.
- **Equipo**: El computador que trae el cliente para que se le haga mantenimiento o un equipo nuevo que se le vende.
- **Repuestos**: Partes o suministros necesarios para la reparación o expansión de capacidades de un equipo.

- **Pago**: El dinero que entrega el cliente para recibir su equipo o para la puesta en marcha de un proyecto.
- **Inventario**: Es una cuenta de la cantidad de elementos disponibles sobre algún producto específico, siendo actualizado cuando se compran o se venden productos.

3.2.7. Diagrama de Clases de Dominio

La interacción entre las clases de dominio se ha estructurado en un diagrama que permite visualizar las relaciones entre éstas.

La Figura 3.1, describe el diagrama de modelo de clase de dominio del proyecto en desarrollo, donde se permite establecer las relaciones, asociaciones, herencia y composiciones que existen entre cada una de las clases anteriormente descritas.

3.2.8. Identificación y Descripción de Riesgos

Un riesgo es una probabilidad de que el proyecto se vea afectado en su desarrollo o posteriormente en su comportamiento y deben ser tratados de acuerdo a su importancia o prioridad.

3.2.8.1. Riesgos del Sistema

Son los eventos o condiciones que pueden provocar que la entrega de un sistema se postergue, sobrepase el presupuesto, no cumpla con los requisitos o hasta llegue a cancelarse.

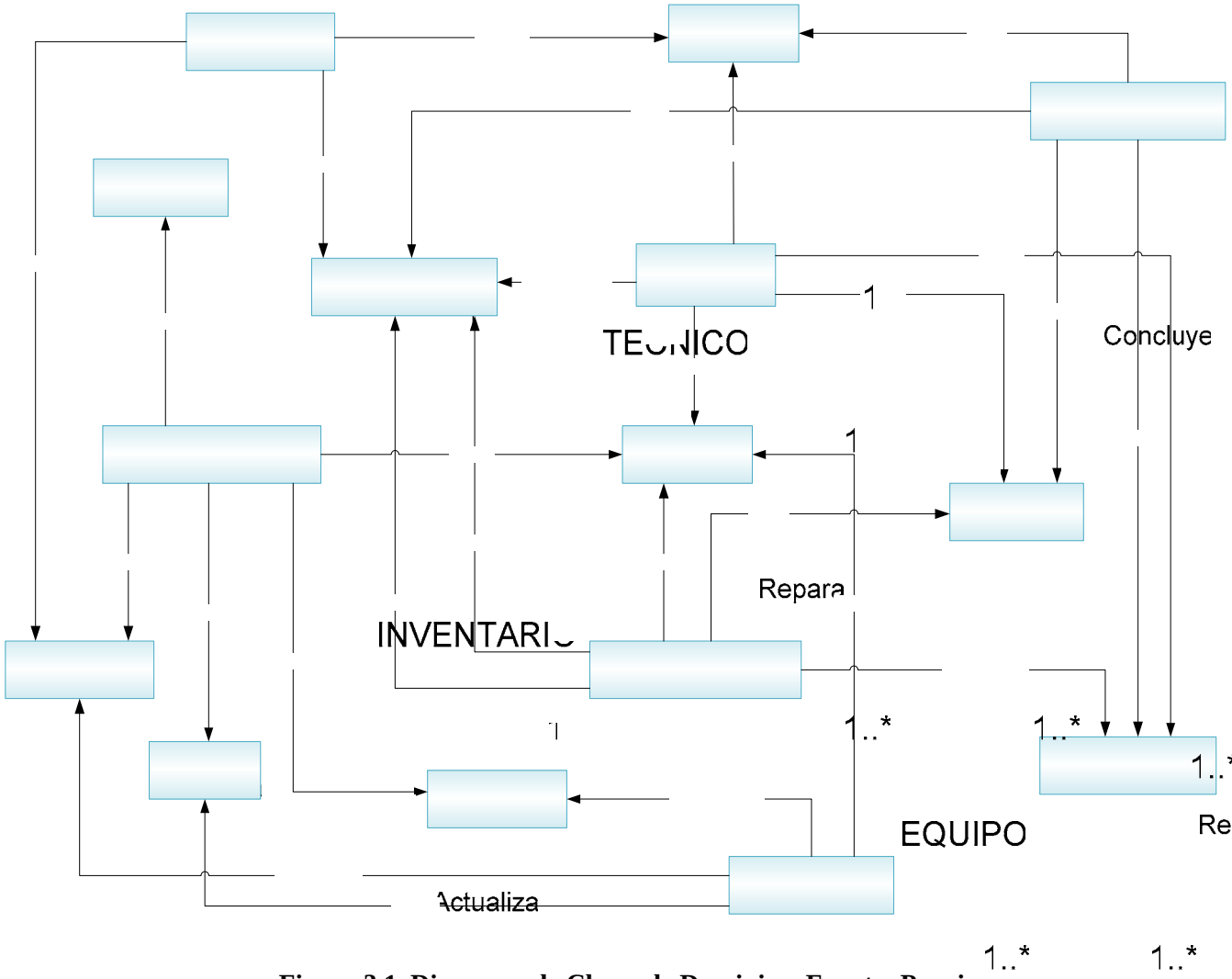


Figura 3.1. Diagrama de Clases de Dominio – Fuente: Propia

Estos riesgos pueden ser de tres tipos: técnico, de requisito o de arquitectura. A continuación mencionaremos los riesgos a tomar en cuenta para el desarrollo del sistema.

3.2.8.2. Riesgos Técnicos:

- El hardware actual del cliente no soporte el software a desarrollar.
- El programador no esté familiarizado con el entorno y/o lenguaje de programación en el cual se va a construir el sistema.

- El sistema de bases de datos no sea el más adecuado para los requerimientos del sistema.
- El paradigma de programación seleccionado no sea compatible con la arquitectura del sistema.
- El software no sea soportado por el Sistema Operativo del cliente.

3.2.8.2.1. Riesgos de Requerimientos:

- Falta de comprensión y/o interpretación del dominio.
- El sistema no contemple todas las funciones que requiera el cliente.
- Requisitos repetidos, mal formulados o no refinados.

3.2.8.2.2. Riesgos de Arquitectura:

- El sistema no sea robusto, falta de acoplamiento entre los módulos.
- Sea difícil de actualizar, expandir y/o corregir errores.

3.2.8.2.3. Como Minimizar los Riesgos

Se deberán estudiar los requisitos mínimos de hardware indicados por los fabricantes del entorno de desarrollo, sistema operativo, sistema de base de datos y cualquier otra herramienta que sea necesaria para el funcionamiento de la aplicación. Para garantizar el funcionamiento del software los requisitos mínimos de hardware deberán ser sobrepasados.

Para construir el sistema se deberá elegir un lenguaje y entorno de programación que sea fácil de utilizar por el programador o por lo menos en donde éste tenga experiencia, para facilitar la construcción y poder cumplir con el cronograma establecido.

El paradigma de programación debe contar con características que permitan la escalabilidad y/o modificación de los módulos del sistema sin afectarlo en su

totalidad con la finalidad de obtener un sistema robusto y acorde a la arquitectura deseada.

Se debe elegir un sistema de base de datos que ofrezca opciones de seguridad y respaldo de los datos, además, debe proveer interfaces de conexión compatibles con el entorno de programación, soportar gran cantidad de registros y búsquedas rápidas de información así como la concurrencia y lenguaje de consulta SQL.

Se deberán estudiar las características del Sistema Operativo del cliente para determinar si podrá soportar sin inconvenientes a la aplicación.

Para minimizar los riesgos de requerimiento es necesario entender lo más posible los procesos que se manejan en la empresa. A través de entrevistas, glosarios de términos y encuestas a los trabajadores se podrá obtener una clara comprensión del dominio.

Para obtener una buena arquitectura del sistema será necesario determinar que casos de uso son críticos y darles prioridad. También se debe tomar en cuenta que el sistema se compone de distintos módulos que a su vez están acoplados pero que no debe llegarse al punto de que sean dependientes entre si porque esto irá contra la robustez y solidez del sistema. Es recomendable documentar de la mejor manera posible.

3.2.9. Modelo de Casos de Uso

El modelo de caso de uso permite que los desarrolladores de software y los clientes lleguen a un acuerdo sobre las necesidades y condiciones que debe cumplir el sistema.

Un caso de uso es un fragmento de funcionalidad del sistema. También se puede afirmar que es una forma en que los usuarios se relacionan con una parte del sistema. El modelo de caso de uso describe lo que hace el sistema para cada tipo de usuario o actor.

En el diagrama de casos de uso se representa también el sistema como una caja rectangular con el nombre en su interior. Los casos de uso están en el interior de la caja del sistema, y los actores fuera, y cada actor está unido a los casos de uso en los que participa mediante una línea.

3.2.10. Actores

Un actor es una entidad humana o máquina que interactúa con el sistema para desarrollar un trabajo significativo. Un actor posee las siguientes características:

- Cualquier entidad externa al sistema.
- Cada usuario del sistema se representa con un actor.
- Suelen corresponderse con trabajadores (actores del negocio).
- Los actores pueden obtener o ingresar información al sistema.
- Interactúan con el sistema. Los actores pueden utilizar funcionalidades provistas por el sistema, pero también pueden proveer funcionalidades al sistema.

3.2.10.1. Identificación de los actores del Sistema de Información APLICAD

- **Recepcionista:** Elabora las órdenes de servicio, llenando los datos de la misma. También puede incorporar nuevos clientes y sus equipos que no estén registrados en la base de datos.
- **Comprador:** Se encarga de elaborar y procesar las órdenes de compra, registrando el pago y cargando los seriales de los productos. También se

encarga de incorporar nuevos proveedores y productos en la base de datos y puede requerir reportes significativos asociados a las compras.

- **Vendedor**: Se encarga de la facturación de los productos y servicios detallados en una orden de servicio o de cualquier manera en que sean vendidos. También actualiza los precios de venta de productos y servicios y registra los pagos de los clientes y puede requerir reportes significativos asociados a las ventas.
- **Técnico:** Es quien agrega los detalles de servicios y/o repuestos utilizados a una orden de servicio. También puede consultar el historial de servicio del equipo de un cliente determinado.
- **Administrador del Sistema:** Es el encargado de manejar las cuentas de usuario y sus privilegios y por lo tanto es el responsable de controlar el acceso al sistema.
- **S.M.B.D:** Sistema Manejador de la Base de Datos, el cual se encarga de la gestión de las consultas, agregación y actualización de la base de datos.

3.2.11. Casos de Uso

Un Caso de Uso es la representación de una parte del comportamiento del sistema desde el punto de vista del usuario, como se mencionó anteriormente el caso de uso puede verse como un fragmento de la funcionalidad del sistema.

3.2.11.1. Identificación de los Casos de Usos del Sistema APLICAD

- Gestionar Archivos de Datos: Este caso de uso contempla todo lo referente a la incorporación, actualización y eliminación de registros de los clientes, proveedores, productos, servicios o de cualquier entidad con información necesaria para los procesos administrativos de la empresa. Realizado por los actores: Comprador, Vendedor y Recepcionista.

- Vender Productos y Servicios: El proceso de venta de productos y servicios consiste de varios subprocesos como: facturación, presupuesto, orden de servicio, pagos, entre otros, los cuales están contemplados dentro de este caso de uso. Realizado por los actores: Vendedor, Recepcionista y Técnico.
- Comprar Productos: Este caso de uso contempla lo referente al proceso de compra de equipos, como son: elaborar órdenes de compra, actualización de inventario y procesamiento de la compra. Realizado por el Comprador.
- Producir Reportes: Todo los resúmenes de información o impresos que genera el sistema han sido ubicados dentro del caso de uso producir reportes. Realizados por los actores: Comprador y Vendedor.
- Controlar Acceso al Sistema: Consiste en la gestión de los usuarios del sistema y los privilegios de uso que tienen los mismos. Realizado por el Administrador del Sistema.

3.2.11.2. Diagrama de Caso de Uso

El diagrama de caso de uso representa un conjunto de escenarios unidos por un objeto común. Cada escenario describe una secuencia de eventos, cada secuencia es iniciada por un actor u otro sistema.

La Figura 3.2, muestra el diagrama de caso de uso general de la Aplicación de Control Administrativo (APLICAD), la cual esta formada por un conjunto de casos y sus respectivos actores, según lo que se ha definido en párrafos anteriores.

Como se visualiza en el referido diagrama el actor S.M.B.D (Sistema Manejador de Base de Datos) se vincula con todos los escenarios debido a que cada una de las operaciones que este caso de uso ejecuta necesitan el acceso a la base de datos.

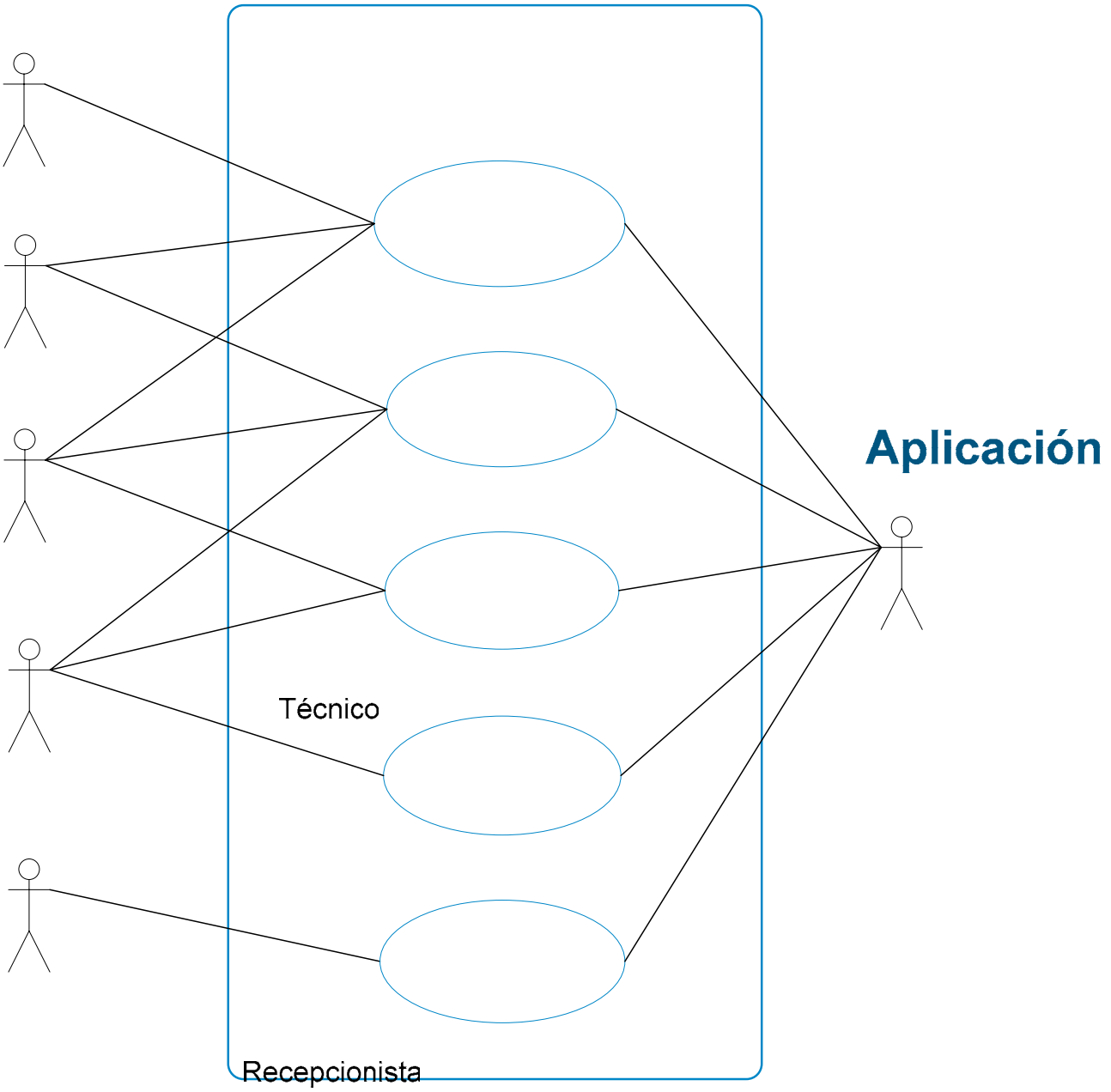


Figura 3.2. Diagrama de Caso de Uso General de APLICAD – Fuente: Propia

3.2.11.3. Descripción de los Casos de Uso del Modelo Contextual APLICAD

- Caso de Uso: Gestionar Archivos de Datos

Actores: Comprador, Vendedor, Recepcionista y S.M.B.D

Descripción:

Inicio del caso de uso: El Comprador, Vendedor o Recepcionista necesita incorporar, actualizar o eliminar información de la base de datos.

Fin del caso de uso: Los datos son registrados, actualizados o eliminados satisfactoriamente.

Funcionalidad del sistema: En este caso de uso se realiza la gestión de los archivos de clientes, proveedores, productos y servicios, que son indispensables para otros procesos.

Interacción con los actores: Para incorporar un registro el Comprador, Vendedor o Recepcionista deberá ingresar la información básica necesaria. Si quiere actualizar o desincorporar un registro primero deberá seleccionar uno antes de realizar los cambios. En todos los casos el sistema mostrará un mensaje si fue posible o no la aplicación de los cambios en la base de datos.

Repetición del Comportamiento:

Mientras (existan requerimiento de ingreso, actualización o eliminación)

Examinar e identificar los datos disponibles

Seleccionar los campos correspondientes para cada uno de los datos

Verificar que no existan errores en los datos

Guardar en registros adecuados

Fin Mientras

- Caso de Uso: Vender Productos y Servicios

Actores: Vendedor, Recepcionista, Técnico y S.M.B.D

Descripción:

Inicio del caso de uso: Es activado cuando alguno de los actores involucrados desea realizar una venta de productos o servicios.

Fin del caso de uso: Este finaliza cuando el sistema procesa de manera satisfactoria la petición.

Funcionalidad del sistema: Este caso de uso realizada la gestión de cualquier tipo de proceso de venta de equipos y servicios deseados.

Interacción con los actores: El Vendedor, Recepcionista o Técnico activa el proceso de venta que desea como: facturación, presupuesto, orden de servicio y los cambios son guardados mediante el S.M.B.D.

Repetición del Comportamiento:

Mientras (exista un requerimiento de venta de equipos o servicios)

Seleccionar que tipo de requerimiento

Procesar el requerimiento de venta de equipos o servicios

Actualizar la base de datos

Fin Mientras

- Caso de Uso: Comprar Productos

Actores: Comprador y S.M.B.D

Descripción:

Inicio del caso de uso: Es activado cuando alguno de los actores involucrados desea realizar una compra de productos.

Fin del caso de uso: Este finaliza cuando el sistema procesa de manera satisfactoria la petición.

Funcionalidad del sistema: Este caso de uso realizada la gestión de cualquier compra de productos.

Interacción con los actores: El Comprador activa el proceso de compra que desea como: ordenes de compra, actualización de inventario y

procesamiento de la compra, y los cambios son guardados mediante el S.M.B.D.

Repetición del Comportamiento:

Mientras (exista un requerimiento de compra de equipos)

Seleccionar que tipo de requerimiento

Procesar el requerimiento de compra

Actualizar la base de datos

Fin Mientras

- Caso de Uso: Producir Reportes

Actores: Comprador, Vendedor y S.M.B.D

Descripción:

Inicio del caso de uso: Es activado cuando alguno de los actores involucrados necesite realizar una consulta a la base de datos.

Fin del caso de uso: Este finaliza cuando los datos de la consulta son mostrador por pantalla y/o impresos de manera satisfactoria.

Funcionalidad del sistema: Este caso de uso realizada la gestión de cualquier tipo de consulta deseada a la base de datos, las cuales están divididas por las siguientes secciones: compras, ventas, servicios, archivos.

Interacción con los actores: El Comprador o Vendedor activa una o más secciones de consulta y configuran el tipo de consulta que quieren realizar. Luego el S.M.B.D, responde mostrando los resultados de la consulta por pantalla y si lo desea el actor puede imprimir la información.

Repetición del Comportamiento:

Mientras (los datos requeridos no sean obtenidos)

Seleccionar que tipo de consulta

Seleccionar los campos por los que se desea consultar

Seleccionar los campos que se desean mostrar

Fin Mientras

- Caso de Uso: Controlar Acceso al Sistema

Actores: Administrador del Sistema y S.M.B.D

Descripción:

Inicio del caso de uso: Es activado cuando el Administrador del Sistema alguno de los actores involucrados desea configurar las cuentas de usuarios.

Fin del caso de uso: Este finaliza cuando se configura exitosamente las cuentas de usuarios.

Funcionalidad del sistema: Este caso de uso permite manejar las cuentas y contraseñas de acceso del sistema de todos los usuarios y por lo tanto controlar el acceso al sistema.

Interacción con los actores: el Administrador del Sistema configura los privilegios de los otros usuarios del sistema estos cambios se guardan en S.M.B.D.

Repetición del Comportamiento:

Mientras (exista un requerimiento de controlar el acceso al sistema)

Configurar Privilegios.

Guardar cambios en la base de datos

Fin Mientras

3.2.12. Captura de Requisitos Adicionales

Son requisitos no funcionales, que no pueden asociarse a ningún caso de uso en concreto, en cambio cada uno de estos requisitos tienen impacto en varios casos de uso o en ninguno. Especifican propiedades como restricciones del entorno o de la implementación, rendimiento, dependencias a la plataforma, facilidad de mantenimiento, extensibilidad y fiabilidad. Estos requisitos se capturan de forma muy parecida a como se hacia en la especificación de requisitos funcionales, es decir, como una lista de requisitos.

3.2.12.1. Lista de Requisitos Adicionales del Proyecto APLICAD

- Diseñar interfaces que le permitan al usuario llevar a cabo los casos de uso de manera eficiente.
- Es necesario para el funcionamiento del sistema la creación del un servidor de base de datos.
- Los usuarios deben encontrase dentro de una red local, dependiendo de la configuración del servidor, para tener acceso al sistema.

3.3. Análisis

El propósito del análisis es examinar y refinar los requisitos del sistema, utilizando un lenguaje más formal, a diferencia del empleado en el caso de uso. Este lenguaje se basa en un modelo de objeto conceptual, llamado modelo de análisis, donde se refinan los requisitos, se deducen los aspectos internos del sistema y se realiza una estructura de los requisitos manipulados. Además el modelo de análisis se considera como una primera aproximación del modelo de diseño, y es por lo tanto un aspecto fundamental cuando se ingresa al diseño e implementación de cualquier proyecto.

Para crear este modelo, se comienza con el análisis de la arquitectura del sistema; seguidamente se realiza cada caso de uso en términos de clase de análisis, exponiendo los requisitos de comportamiento de cada clase, los cuales serán especificados para integrarlos dentro de las clases.

En la fase de inicio, se realiza un ligero análisis de los requisitos, el cual se utiliza para obtener una versión inicial del modelo de análisis, que servirá para tener noción de lo que se va a realizar en la etapa de diseño.

3.3.1. Análisis de los Casos de Uso

El análisis de los casos de uso se realiza para identificar las clases de análisis cuyos objetos son necesarios para llevar a cabo el flujo de sucesos del caso de uso, distribuir el comportamiento de los casos de uso entre los objetos del análisis y capturar requisitos especiales sobre la realización del caso de uso.

Una clase de análisis representa abstracciones de clases y posiblemente de subsistemas del diseño del proyecto. Dentro del modelo de análisis, los casos de uso se describen mediante clases de análisis y sus objetos.

3.3.2. Identificación de las Clases de Análisis

En la identificación de las clases de análisis, se especifican las clases de control, entidad e interfaz necesarias para realizar los casos de uso, para luego esbozar sus nombres, responsabilidades, atributos y relaciones.

3.3.2.1. Clases de Control

Las clases de análisis de control, representan coordinación, secuencia, transacciones y control de otros objetos y se usan con frecuencia para encapsular el control de un caso de uso en concreto.

Las clases de control para el proyecto APLICAD se muestran a continuación:

- **:Gestor Archivo:** Controla todo lo referente a los datos de clientes, proveedores, productos y servicios en el sistema.
- **:Gestor Ventas:** Es el encargado de realizar los procesos de venta de productos y servicios.
- **:Gestor Compras:** Se encarga de gestionar los procesos concernientes a la compra de productos y actualizar el inventario.

- **:Gestor Reportes:** Se encarga de controlar la generación de reportes del sistema.
- **:Gestor Control Acceso:** Se encarga de gestionar las cuentas de usuarios y sus privilegios.

3.3.2.2. Clases de Entidad

Las clases entidad modelan la información y el comportamiento asociado a algún fenómeno o concepto, como una persona, un objeto o un suceso del mundo real.

Las clases de control para el proyecto APLICAD se muestran a continuación:

- **:Cliente:** Almacena un registro con los datos principales de los clientes.
- **:Proveedor:** Almacena un registro con los datos principales de los proveedores.
- **:Producto:** Almacena un registro con los datos de los productos.
- **:Servicio:** Almacena un registro con los datos de los servicios ofrecidos en la empresa.
- **:Factura:** Representa la información concerniente a una factura.
- **:Orden Servicio:** Representa la información concerniente a una orden de servicio.
- **:Presupuesto:** Información relacionada a un presupuesto de venta.
- **:Compra:** Información relacionada con una compra de productos.
- **:Orden de Compra:** Información relacionada con una orden de compra de productos.

- **:Inventario:** Guarda información referente a las cantidades disponibles de los productos que se venden.
- **:Pago:** Guarda información referente a los pagos de las facturas y/o de las compras de productos.
- **:Usuario:** Almacena un registro con los datos de conexión y privilegios de un usuario del sistema.

3.3.2.3. Clases de Interfaz

Las clases de interfaz se utilizan para modelar la interacción entre el sistema y sus actores.

Las clases de control para el proyecto APLICAD se muestran a continuación:

- **:IU Principal:** Permite el acceso por parte del personal de SAIT, RL a los distintos módulos y funciones del sistema.

3.3.3. Diagrama de Clases de Análisis

Luego de analizar los casos de uso e identificar las clases de análisis, las mismas pueden observarse en modo gráfico a través del siguiente diagrama general de clases de análisis (Figura 3.3), el cual es una vista general que será ampliada en el próximo capítulo para los principales casos de uso del proyecto APLICAD.

3.3.4. Análisis de la Arquitectura

El objetivo del análisis de la arquitectura es esbozar el modelo de análisis, mediante la identificación de paquetes de análisis.

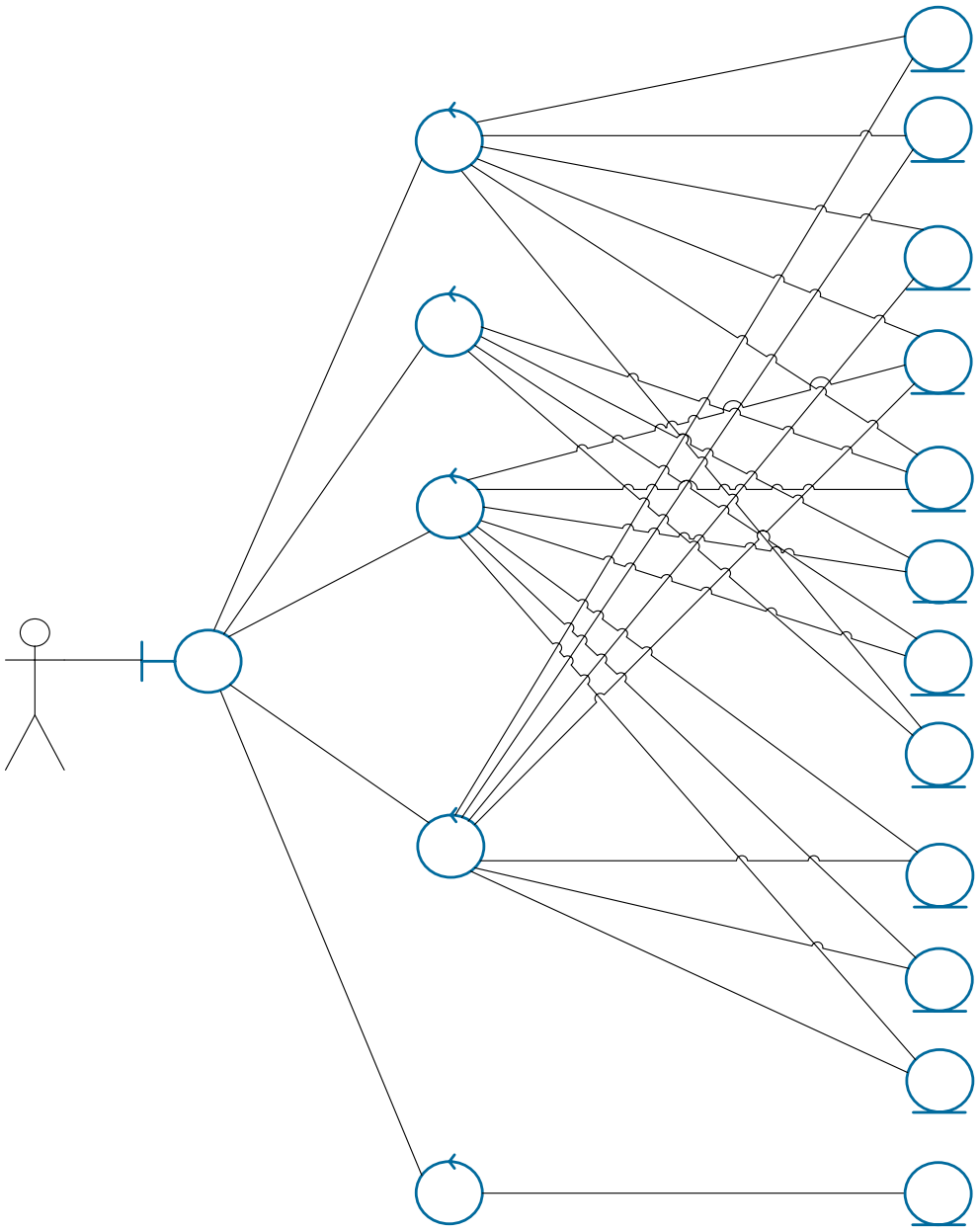


Figura 3.3. Diagrama General de Clases de Análisis APLICAD – Fuente: Propia

3.3.4.1. Identificación de Paquetes de Análisis

El paquete de análisis representa un medio de organizar el modelo de análisis en piezas más pequeñas, basándose en los requisitos funcionales y en el dominio del problema.

Un paquete define un espacio de nombre (Figura 3.4), de modo que dos elementos diferentes, contenidos en dos paquetes diferentes, pueden tener el mismo nombre.

Un paquete puede contener otros paquetes, sin límite de nivel de anidamiento. Un nivel dado puede contener una mezcla de paquetes y de otros elementos de modelado, de la misma manera que un directorio puede contener directorios y archivos.

Figura 3.4. Representación de un paquete – Fuente: [13]

3.3.4.2. Identificación de los paquetes de análisis para el proyecto APLICAD

Paquete de Gestionar Archivos de Datos: este paquete incluye todas las actividades relacionadas con la inserción, actualización y eliminación de datos, el cual tiene una relación directa con el caso de uso del mismo nombre (Figura 3.5)

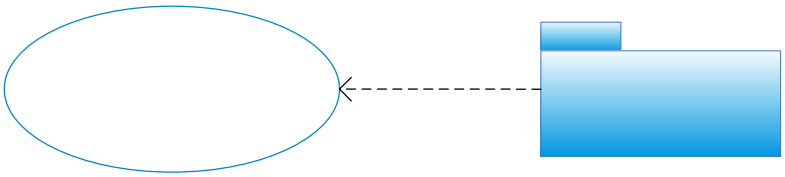


Figura 3.5. Paquete de Análisis – Gestionar Archivos de Datos – Fuente: Propia

Paquete de Análisis Vender Productos y Servicios: este paquete incluye todas las acciones necesarias para procesar una venta sea de equipos o servicios, en el sistema, este paquete tiene una relación directa con el caso de uso del mismo nombre (Figura 3.6)

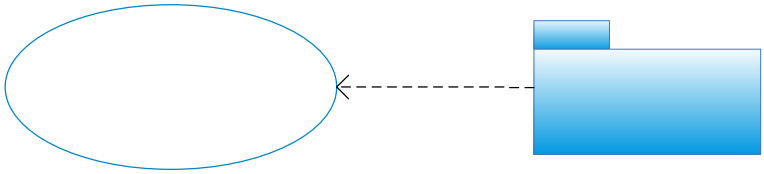


Figura 3.6. Paquete de Análisis – Vender Productos y Servicios – Fuente: Propia

Paquete de Comprar Productos: este paquete incluye todas las actividades relacionadas con las funciones con el proceso de compra de equipos, el cual tiene una relación directa con el caso de uso del mismo nombre (Figura 3.7)

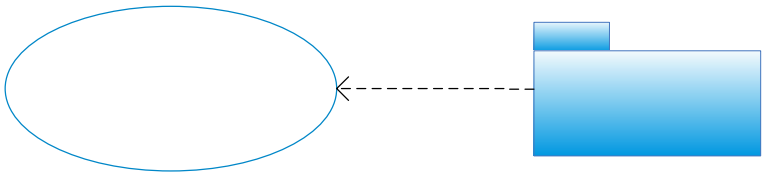


Figura 3.7. Paquete de Análisis Comprar Productos – Fuente: Propia

Vender Productos y Servicios

Paquete de Producir Reportes: este paquete incluye todas las actividades relacionadas con la generación de reportes ya sean por pantalla o impresos, el cual tiene una relación directa con el caso de uso del mismo nombre (Figura 3.8)

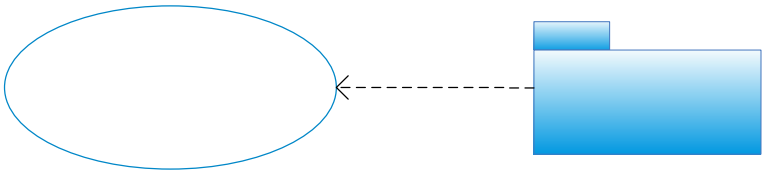


Figura 3.8. Paquete de Análisis Producir Reportes – Fuente: Propia

Paquete de Análisis Procesos Administrativos: este paquete incluye todas las actividades relacionadas con las funciones administrativas de la empresa Cooperativa SAIT, R.L. (Figura 3.9).

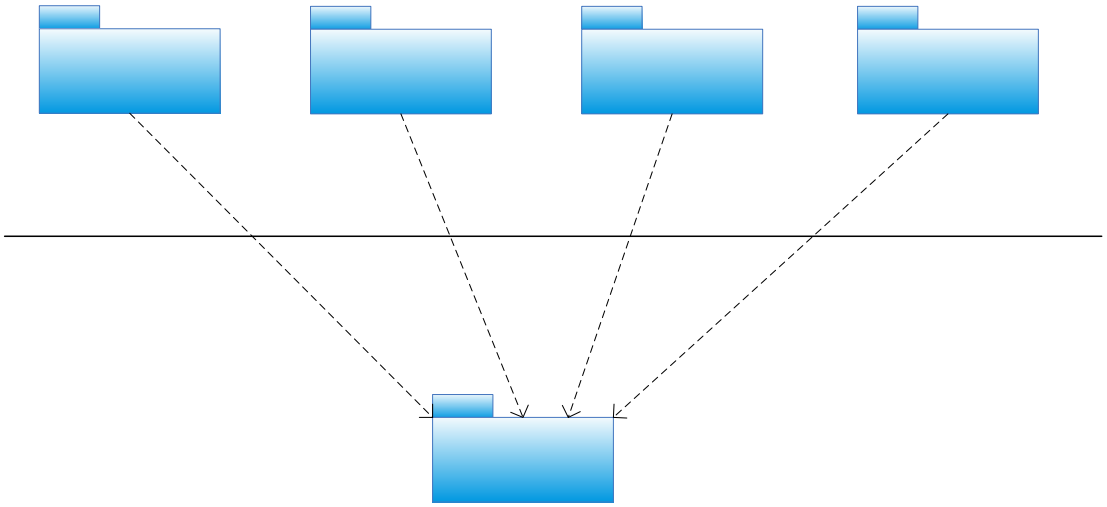


Figura 3.9. Paquete de Análisis Procesos Administrativos – Fuente: Propia

Gestionar Archivos de Datos

Vender Productos y Servicios

3.4. Logros De La Fase De Inicio

Después de haber completado los flujos de trabajo necesarios para la culminación de la fase de Inicio de APLICAD, se obtuvieron los siguientes logros:

En el flujo de los Requisitos, se consiguió establecer el modelo de casos de uso inicial, en donde se muestra la funcionalidad del sistema representada mediante la interacción de los actores y los casos de uso.

En el flujo de Análisis, se esbozó la arquitectura candidata de APLICAD a partir de la elaboración de los modelos de clases de análisis para la realización de los casos de uso identificados en el flujo requisitos. De igual manera, se determinaron los

Proc
Admini

paquetes de análisis, cada uno de los cuales permitió organizar mediante trazas los casos de uso identificados en el flujo de requisitos.

Por último, se establecieron los riesgos que pudiesen afectar el desarrollo de APLICAD, esto con la finalidad de mitigarlos en la medida de lo posible en las futuras fases.

CAPÍTULO IV

FASE DE ELABORACIÓN

4.1. Introducción

La fase de elaboración construye la línea base de la arquitectura del sistema. Los principales objetivos para esta fase son:

- Recopilar la mayor parte de los requisitos que aún queden pendientes, formulando los requisitos funcionales como casos de uso.
- Establecer una base de la arquitectura sólida -la línea base de la arquitectura- para guiar el trabajo durante las fases de construcción y transición, así como en las posteriores generaciones del sistema.
- Continuar la observación y control de los riesgos críticos que aún queden, e identificar riesgos significativos hasta el punto de que se pueda estimar su impacto en el análisis negocio, y en particular en la apuesta económica.
- Completar los detalles del plan del proyecto.

4.2. Requisitos

En este flujo de trabajo se refinan los requisitos encontrados en la fase de inicio, hasta especificar la mayoría de casos de uso y actores indispensables para el desarrollo del sistema, también se incluyen los nuevos requerimientos encontrados como casos de uso.

4.2.1. Actores

En esta iteración no se encontraron nuevos actores ni se realizaron actualizaciones a los obtenidos en la fase de inicio.

4.2.2. Casos de Uso Detallados

A continuación se describen y detallan los casos de uso principales del sistema, dando una especificación de los nuevos actores vinculados, las condiciones de inicio y fin de los mismos así como su comportamiento.

4.2.2.1. Caso de Uso Vender Productos y/o Servicios

Este caso de uso trata del proceso de venta de productos y/o servicios a los clientes de SAIT, RL. Para realizar una venta de productos y/o servicios, algunas veces el cliente primero solicita la elaboración de un presupuesto de venta que le informe al detalle el precio de los productos y/o servicios que va a adquirir. El Vendedor elabora el presupuesto cargando los datos del cliente y las cantidades y precios de los productos y/o servicios requeridos. Finalmente el presupuesto es entregado al cliente quien lo aprobará o rechazará.

El otro tipo de venta es la facturación de productos y/o servicios realizados por el departamento técnico de SAIT, RL. Primeramente la recepcionista elabora una orden de servicio cargando los datos del cliente y del equipo entregado por el cliente, y la descripción del problema o requerimiento. Posteriormente el técnico deberá completar la orden de servicio agregando los detalles del servicio que realizado y los repuestos utilizados con sus seriales si viene al caso.

Todas las ventas son realizadas a través del proceso de facturación el cual crea un registro de la venta con los datos del cliente, fecha de venta, monto total y un desglose detallado de los productos y servicios adquiridos por el cliente, teniendo que actualizar el inventario (descarga) al momento de vender un producto. Por último se procede al registro del pago de la factura, cargando la forma de pago y demás datos asociados al pago.

El caso de uso Vender Productos y Servicios se muestra en la Figura 4.2.

- **Caso de Uso:** Vender Productos y Servicios.

  Actores: Técnico, Recepcionista, Vendedor y S.M.B.D.

  Descripción:

  Inicio del caso de uso: Es activado cuando alguno de los actores involucrados desea realizar una venta de productos o servicios.

  Fin del caso de uso: Este finaliza cuando el sistema procesa de manera satisfactoria la petición.

  Funcionalidad del sistema: Este caso de uso realizada la gestión de cualquier tipo de proceso de venta de equipos y servicios deseados.

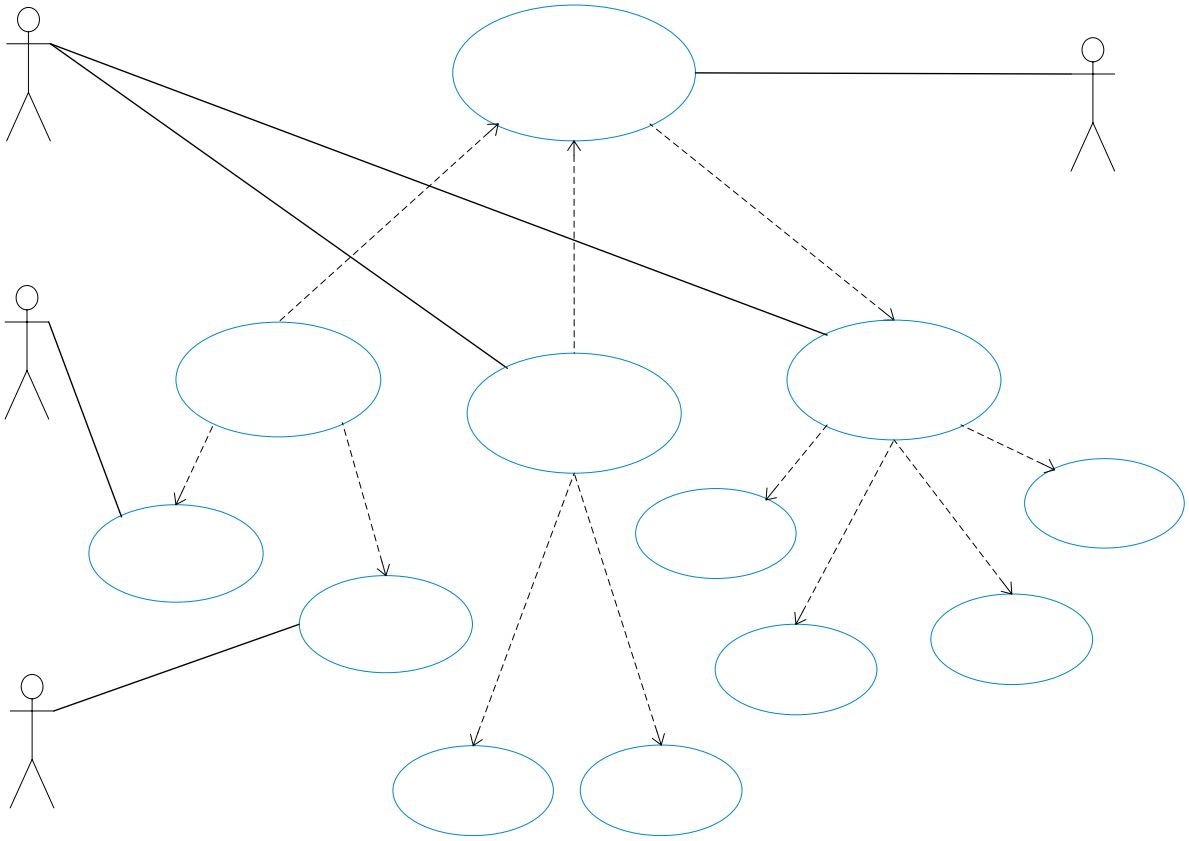


Figura 4.2. Diagrama del Caso de Uso Vender Productos y Servicios – Fuente: Propia

Ven

Interacción con los actores: El Personal de SAIT activa el proceso de venta que desea como: facturación, presupuesto, orden de servicio, pagos, entre otros, y los cambios son guardados mediante el S.M.B.D.

Repetición del Comportamiento:

Mientras (exista un requerimiento de venta de equipos o servicios)

Seleccionar que tipo de requerimiento

Procesar el requerimiento de venta de equipos o servicios

Actualizar la base de datos

Fin Mientras

4.2.2.2. Caso de Uso Comprar Productos

Este caso de uso describe los procesos que se realizan para la adquisición de equipos y piezas de computación para la venta.

Lo primero que hace el comprador es consultar el inventario para ver que piezas tienen baja disponibilidad o necesitan reponerse en el inventario. Luego elabora una orden de compra, para solicitar mercancía al proveedor, que contiene todos los datos del pedido y los datos del proveedor.

El último paso es procesar la compra, primero se cargan los datos desde una orden de compra y si hubo alguna variación en el pedido final se realiza la actualización correspondiente, cabe mencionar que en este paso también se realiza la reposición o carga del inventario de productos. Luego se procede a registrar el pago de la compra y por último, si es necesario, se cargan los seriales de los productos comprados con los que posteriormente podrán ser vendidos o se validen garantías. Todos los cambios son almacenados en la base de datos a través del S.M.B.D. Podemos observar este caso de uso en la Figura 4.3.

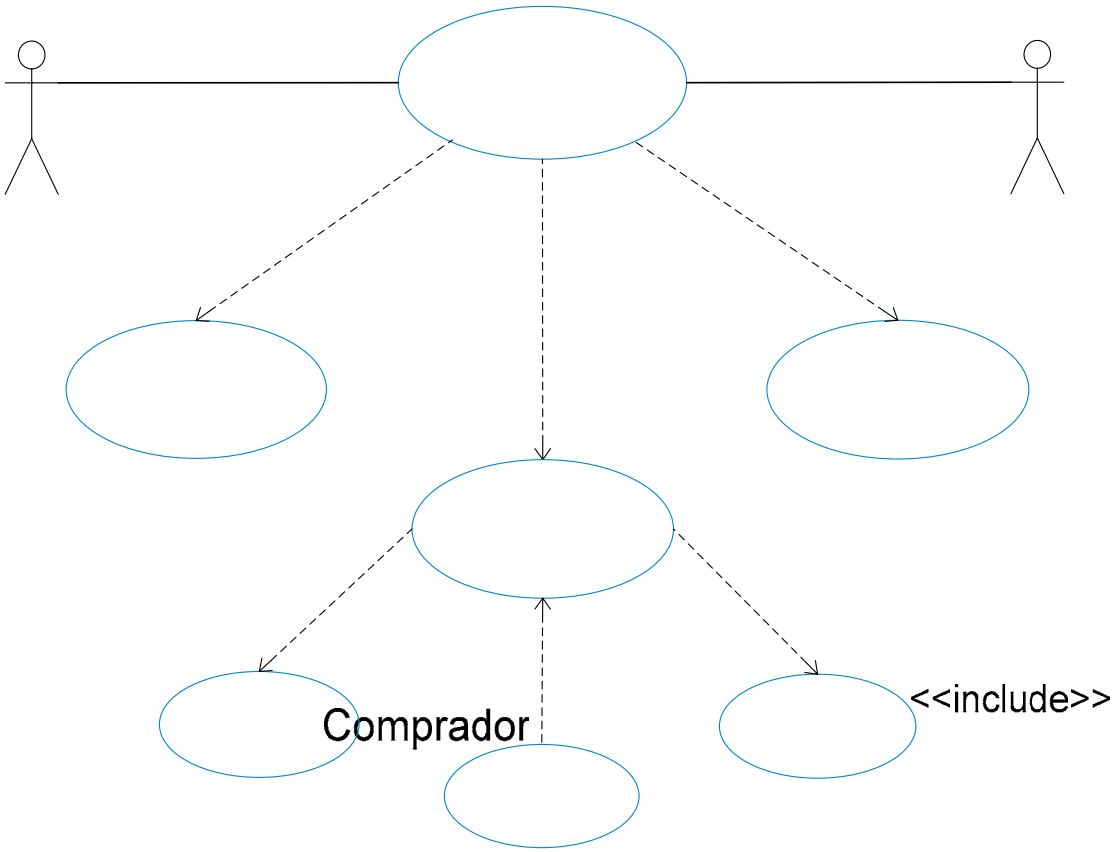


Figura 4.3. Diagrama del Caso de Uso Comprar Productos – Fuente: Propia

- **Caso de Uso:** Comprar Productos.

  Actores: Comprador y S.M.B.D.

  Descripción:

  Inicio del caso de uso: Es activado cuando alguno de los actores involucrados desea realizar una compra de productos.

  Fin del caso de uso: Este finaliza cuando el sistema procesa de manera satisfactoria la petición.

  Funcionalidad del sistema: Este caso de uso realizada la gestión de cualquier compra de productos.

Consultar Inventario

<<include>>

Pagar Compra

Interacción con los actores: El Comprador activa el proceso de compra que desea como: órdenes de compra, consulta o actualización de inventario y procesamiento de la compra, y los cambios son guardados mediante el S.M.B.D.

Repetición del Comportamiento:

Mientras (exista un requerimiento de compra de equipos)

Seleccionar que tipo de requerimiento

Procesar el requerimiento de compra

Actualizar la base de datos

Fin Mientras

4.3. Análisis

En la fase de inicio se desarrolló el análisis de la arquitectura sólo hasta el punto de determinar que había una arquitectura del sistema factible. Ahora en esta fase se extiende el análisis hasta el punto de que pueda servir como base a la línea principal de la arquitectura ejecutable.

4.3.1. Identificación de las Clases de Análisis

Se identificaron nuevas clases de análisis y se detallaron otras necesarias para los procesos de vender productos y servicios y comprar productos.

4.3.1.1. Clases de Control

- **:Gestor Datos Factura:** Se encarga de la carga y validación de los datos de la factura.
- **:Gestor Detalles Factura:** Se encarga de gestionar los detalles de la factura validando y asociando los mismos a la factura.

- **:Gestor Datos Presupuesto:** Se encarga de la carga y validación de los datos del presupuesto.
- **:Gestor Detalles Presupuesto:** Se encarga de gestionar los detalles del presupuesto validando y asociando los mismos al presupuesto.
- **:Gestor Datos Orden de Servicio:** Se encarga de la carga y validación de los datos de la orden de servicio.
- **:Gestor Detalles Orden de Servicio:** Se encarga de gestionar los detalles del presupuesto validando y asociando los mismos a la orden de servicio.
- **:Gestor Pago:** Es el encargado de gestionar los datos concernientes al pago de una compra o factura de productos y/o servicios.
- **:Gestor Inventario:** Se encarga de la carga y descarga de unidades de productos cuando se hace una compra o venta.
- **:Gestor Datos Orden de Compra:** Se encarga de la carga y validación de los datos de la orden de compra.
- **:Gestor Detalles Orden de Compra:** Se encarga de gestionar los detalles de la orden de compra validando y asociando los mismos a la orden de compra.
- :**Gestor Compra:** Se encarga de gestionar los datos de la compra procedentes de una orden de compra ya elaborada.
- :**Gestor Detalles Compra:** Se encarga de gestionar los detalles de la compra validando y asociando los mismos a la compra.
- **:Gestor Buscar Datos**. Se encargar de responder todas las solicitudes de búsqueda de datos de clientes, proveedores, órdenes de servicio, facturas, productos y servicios.
- **:Gestor Verificar Usuario:** Se encarga verificar si el usuario tiene privilegios de acceso a la interfaz.

4.3.1.2. Clases de Entidad

Se identificaron nuevas clases de entidad que no aparecieron en el diagrama general.

- **:Detalles Presupuesto:** Información relacionada con un presupuesto de venta.
- **:Detalles Orden de Servicio:** Información relacionada con una orden de servicio.
- **:Detalles Factura:** Información relacionada con una factura de venta de productos y servicios.
- **:Detalles Orden de Compra:** Información relacionada con una orden de compra de productos.
- **:Detalles Compra:** Información relacionada con una orden de compra de productos.
- **:Serial:** Información relacionada con los seriales asociados a cada producto que se compra y vende.

4.3.1.3. Clases de Interfaz

Las clases de interfaz para comprar productos y vender productos y servicios son las siguientes:

- **IU Factura**: Permite gestionar las facturas de venta de productos y servicios.
- **IU Presupuesto**: Permite gestionar los presupuestos de venta.
- **IU Orden de Servicio**: Permite gestionar las ordenes de servicio.
- **IU Pago**: Permite gestionar los pagos de las facturas y las compras.
- **IU Compra**: Permite gestionar las compras de productos.
- **IU Orden de Compra**: Permite gestionar las ordenes de compra.

4.3.2. Diagrama de Clase de Análisis

Para esta fase se presentan los diagramas de clases de análisis para los casos de uso más importantes del sistema, extendiendo el análisis hasta el punto de que pueda servir como base a la línea principal de la arquitectura ejecutable.

4.3.2.1. Diagrama de Clase de Análisis Para el Caso de Uso Vender Productos y Servicios

En la Figura 4.4, se muestra el diagrama de clase de análisis para el caso de uso Vender Productos y Servicios.

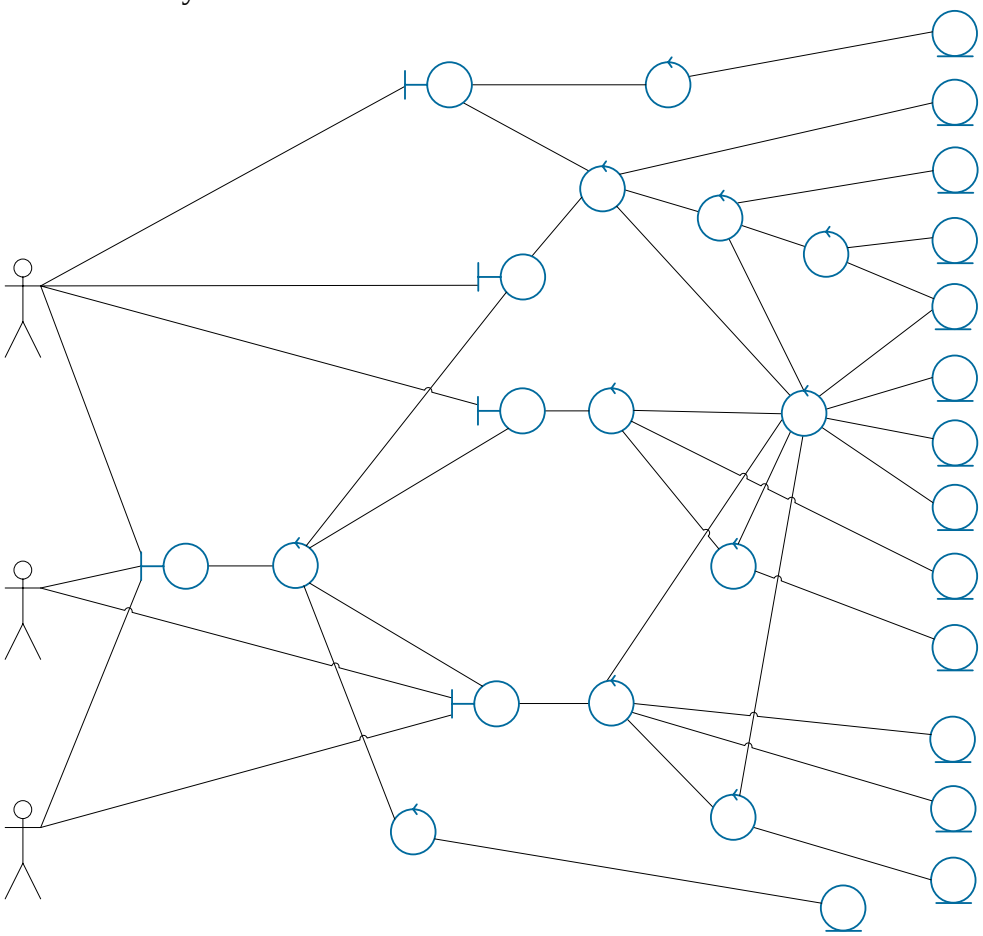


Figura 4.4. Diagrama de Clase de Análisis para el caso de uso Vender Productos y Servicios – Fuente: Propia

4.3.2.2. Diagrama de Clase de Análisis Para el Caso de Uso Comprar Productos

En la Figura 4.5, se muestra el diagrama de clase de análisis para el caso de uso Comprar Servicios.

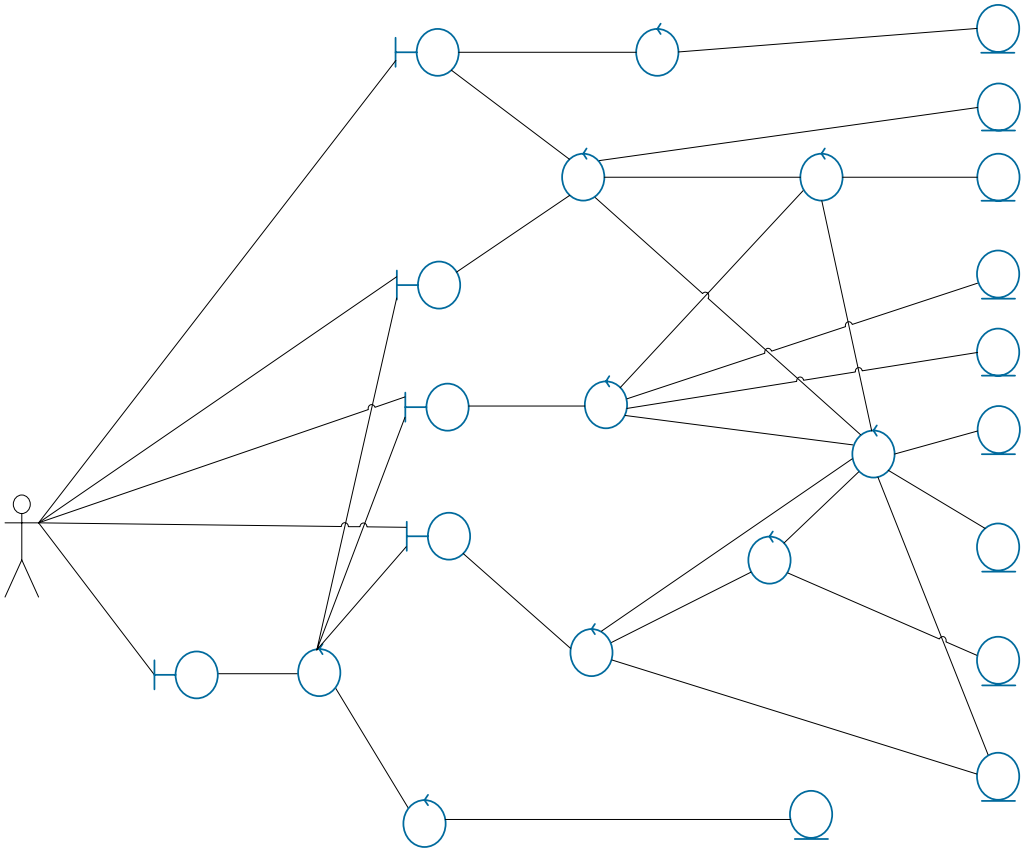
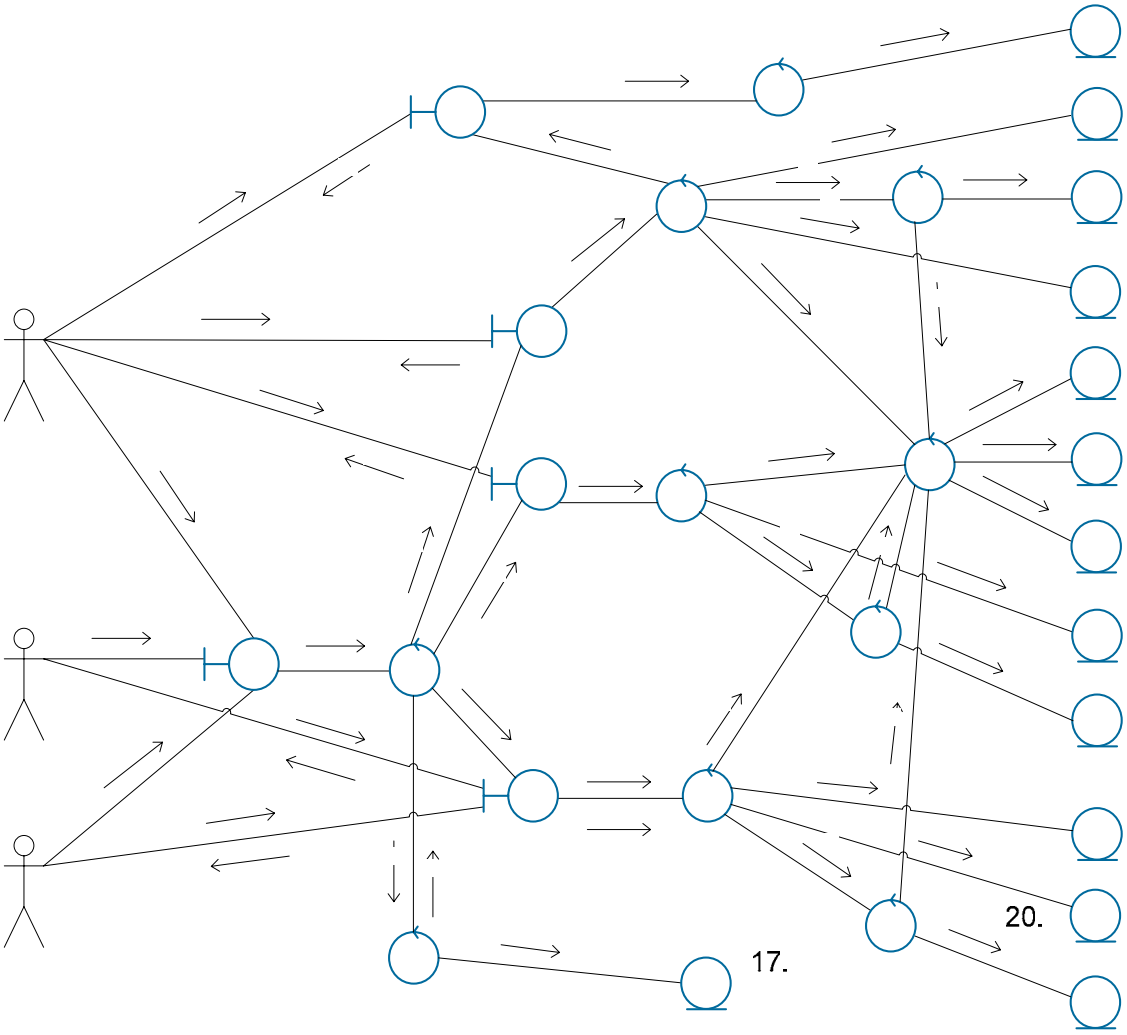


Figura 4.5. Diagrama de Clase de Análisis para el caso de uso Comprar Productos – *Fuente: Propia*

En el siguiente apartado se detalla el flujo de mensajes entre las clases de análisis para dos casos de uso importantes para el sistema APLICAD.

4.3.3. Diagramas de Colaboración para el Caso de Uso Vender Productos y Servicios

En la Figura 4.6, se muestra el diagrama de Colaboración para el caso de uso Vender Productos y Servicios.



Leyenda

1. Acceder a interfaz	2. Solicitar interfaz	3. Solicitar acceso	4.Consultar privilegios 7.	5. Conceder acceso

6. Activar interfaz	7. Acceder a interfaz	8. Elaborar factura	9. Cargar datos cliente	10. Validar y guardar datos
11. Agregar detalles a factura	12. Cargar datos productos, servicios	13. Validar y guardar datos	14. Actualizar inventario	15. Activar interfaz
16. Datos guardados correctamente	17. Acceder a interfaz	18. Registrar pago	19. Validar y guardar datos	20. Pago registrado correctamente
21. Activar interfaz	22. Acceder a interfaz	23. Elaborar presupuesto	24. Cargar datos cliente	25. Validar y guardar datos
26. Agregar detalles a presupuesto	27. Cargar datos productos, servicios	28. Validar y guardar datos	29. Datos guardados correctamente	30. Acceder a interfaz
31. Activar interfaz	32. Acceder a interfaz	33. Elaborar orden de servicio	34. Cargar datos cliente	35. Validar y guardar datos
36. Validar y guardar datos	37. Datos guardados correctamente	38. Acceder a interfaz	39. Acceder a interfaz	40. Modificar orden de servicio
41. Agregar detalles a orden de servicio	42. Validar y guardar datos	43. Datos guardados correctamente	44. Buscar datos cliente	45. Buscar datos servicio
46. Buscar datos producto				

Figura 4.6. Diagrama de Colaboración para el caso de uso Vender Productos y Servicios – Fuente: Propia

El actor **Vendedor** accede a la *IU Principal* (1) la cual solicita acceso a ventas (2) y (3). El *Gestor Verificar Usuario* concede el acceso a la interfaz (3), (4) y (5) y el *Gestor Ventas* activa la interfaz (6). El **Vendedor** accede a la *IU Factura* (7), y selecciona la opción de elaborar nueva factura (8). El *Gestor Datos Factura* activa la carga de datos por el usuario y controla la validación y almacenamiento de los mismos (9) y (10). Se activa el *Gestor Detalles Factura* para agregar (11) los ítems asociados a la factura para ello se seleccionan los productos y servicios (12) se

ingresan las cantidades y se guardan (13) los detalles de la factura asociándolos con la factura correspondiente en la clase de entidad *Detalles Factura*. Se actualizan (14) las cantidades de productos disponibles en la clase de entidad *Inventario.* Luego se activa la *IU Pago* (15) y se le informa al **Vendedor** que el proceso ha sido realizado (16). El **Vendedor** accede (17) a la *IU Pago* para elaborar un nuevo pago (18), así se activa el *Gestor Datos Pago* el cual se encarga de validar y registrar los datos del pago en la clase de entidad *Pago* (19) y se le informa al **Vendedor** que el pago ha sido registrado (20).

Una vez que se la concedido el acceso a la interfaz se activa la *IU* Presupuesto (21) y el actor **Vendedor** accede a la misma (22) y selecciona la opción de elaborar presupuesto (23), luego el *Gestor de Presupuestos* activa la carga de los datos por parte del usuario y controla la validación y almacenamiento de los mismos (24) y (25) en la clase de entidad *Presupuesto*. Se activa el *Gestor Detalles Presupuesto* para agregar (26) los ítems asociados al presupuesto y para ello se seleccionan los productos o servicios (27) se ingresan las cantidades y se guardan (28) por último se le informa el resultado del proceso al Vendedor (29).

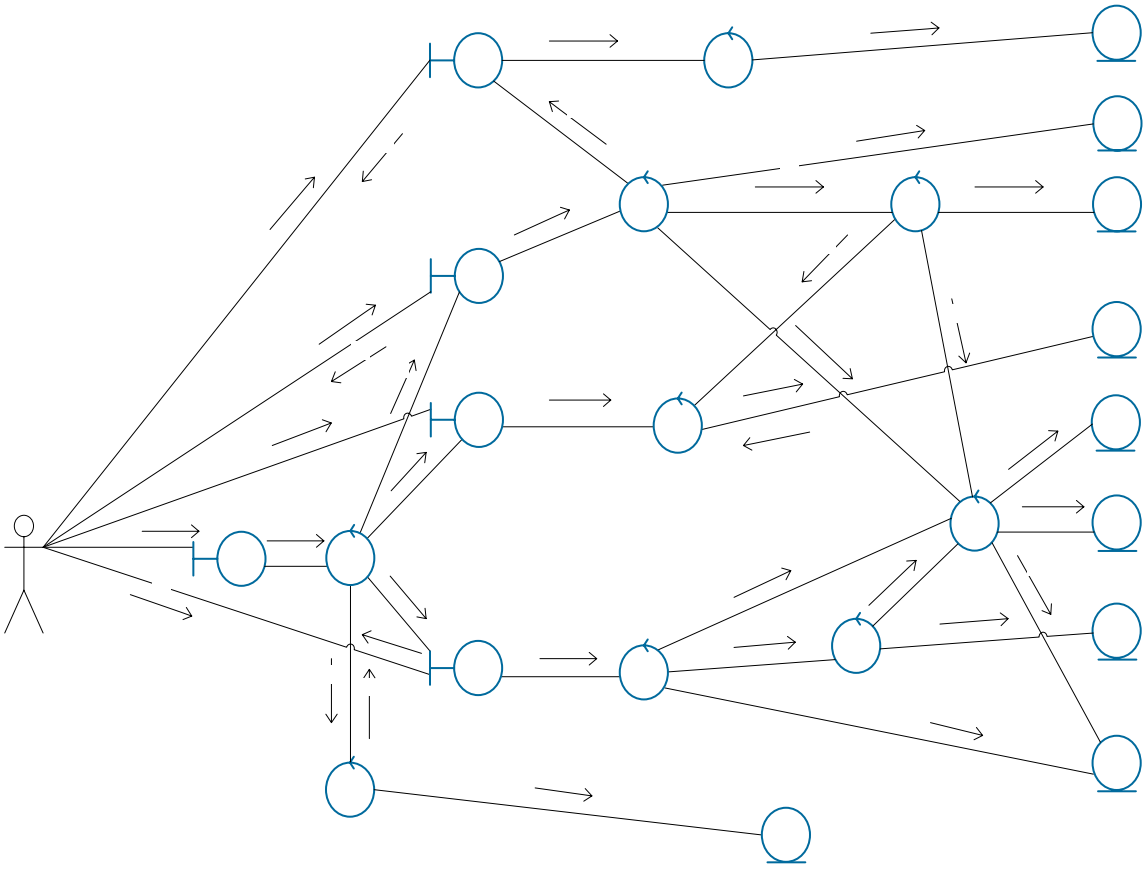
El actor **Recepcionista** accede a la *IU Principal* (30) la cual solicita acceso a ventas (2) y (3). El *Gestor Verificar Usuario* concede el acceso a la interfaz (3), (4) y (5) y el *Gestor Ventas* activa la interfaz (31). La **Recepcionista** accede a la *IU Orden de Servicio* (32) y selecciona la opción elaborar orden de servicio (33), y el *Gestor Orden de Servicio* activa la carga de los datos de la orden y su almacenamiento (34), (35) en la clase de entidad *Orden Servicio*, así como de los equipos del cliente (36) y al finalizar se le informa el resultado de la operación al actor (37).

El actor **Técnico** accede a la *IU Principal* (38) la cual solicita acceso a ventas (2) y (3). El *Gestor Verificar Usuario* concede el acceso a la interfaz (3), (4) y (5) y el *Gestor Ventas* activa la interfaz (31). El **Técnico** accede a la *IU Orden Servicio* (49) y

busca una orden de servicio para editarla (40) y agregar información adicional a la orden, luego el *Gestor Detalles Orden Servicio* activa la carga de los ítems de servicios que se realizaron (41), se buscan los servicios y/o productos (34), se guardan los detalles en la clase de entidad *Detalles Orden Servicio* (42) y se le informa el fin del proceso al actor (43). El *Gestor Buscar Datos* se encarga de responder a las solicitudes de datos de cliente, productos y servicios (44), (45) y (46).

4.3.4. Diagramas de Colaboración para el Caso de Uso Comprar Productos

En la Figura 4.7, se muestra el diagrama de Colaboración para el caso de uso Comprar Productos.



Leyenda

1. Acceder a interfaz	2. Solicitar interfaz	3. Solicitar acceso	4.Consultar privilegios	5. Conceder acceso
6. Activar interfaz	7. Acceder a interfaz	8. Consultar inventario	9. Buscar disponibilidad	10. Número disponibles
11. Activar interfaz	12. Acceder a interfaz	13. Elaborar orden de compra	14. Cargar datos proveedor	15. Validar y guardar datos
16. Agregar detalles a orden de compra	17. Cargar datos producto	18. Validar y guardar datos	19. Datos guardados correctamente	20. Activar interfaz
21. Acceder a interfaz	22. Procesar compra	23. Cargar datos orden de compra	24. Validar y guardar datos	25. Agregar Detalles Compra
26. Cargar Datos Producto	27. Validar y Guardar datos	28. Actualizar Inventario	29. Activar interfaz	30. Datos guardados correctamente
31. Acceder a Interfaz	32. Registrar pago	33. Validar y guardar	34. Pago registrado correctamente	35. Buscar datos proveedor
36. Buscar datos producto	37. Buscar datos orden de compra			

Figura 4.7. Diagrama de Colaboración para el caso de uso Comprar Productos – Fuente: Propia

El actor **Comprador** accede a la *IU Principal* (1) la cual solicita acceso a compras (2) y (3). El *Gestor Verificar Usuario* concede el acceso a la interfaz (3), (4) y (5) y el *Gestor Compras* activa la interfaz (6). El **Comprador** accede a la *IU Consulta Inventario* (7) y configura los parámetros de la consulta (8), entonces el *Gestor Consultar Inventario* procede a buscar los datos y la disponibilidad de los productos requeridos en la clase de entidad *Inventario* (9) y (10).

El actor **Comprador** accede a la *IU Orden de Compra* (12) y selecciona la opción de elaborar una nueva orden de compra (13), luego el *Gestor de Datos Orden de Compra* activa la carga de los datos de la orden y controla la validación y almacenamiento de los mismos (14) y (15). Se activa el *Gestor Detalles Orden de Compra* para agregar, validar y guardar (16), (17) y (18) los ítems asociados a la orden de compra. Se le informa al **Comprador** los resultados del proceso (19).

El actor **Comprador** accede a la *IU Procesar Compra* (21) y selecciona la opción de procesar compra (22). El *Gestor de Datos Compra* activa la cargar los datos desde una orden de compra o por entrada directa del usuario (23) y controla la validación y almacenamiento de los mismos (24) y de los detalles de la compra (25), (26) y (27). Se actualizan (28) las cantidades de productos disponibles en la clase de entidad *Inventario*. Luego se activa la *IU Pago* (29) y se le informa al actor del fin del proceso (30). El **Comprador** accede a la *IU Pago* (31) para registrar un nuevo pago (32), así se activa el *Gestor Pago* el cual se encarga de la carga de los datos del pago, la validación de los mismos y son almacenados en la clase de entidad *Pago* (33). Se le informa al **Comprador** del resultado de la operación de registro de pago (34). El *Gestor Buscar Datos* se encarga de responder a las solicitudes de datos de proveedor, productos y órdenes de compra (35), (36) y (37).

4.4. Diseño

El objetivo del diseño de la arquitectura es esbozar los modelos de diseño y despliegue y su arquitectura.

4.4.1. Identificación de Subsistemas de Diseño a Partir de los Paquetes de Análisis

Los subsistemas constituyen un medio para organizar el modelo de diseño en piezas manejables. Pueden bien identificarse inicialmente como forma de dividir el

trabajo de diseño o irse encontrando a medida que el modelo del diseño evoluciona y va “creciendo” hasta llegar a convertirse en una gran estructura que debe ser descompuesta.

4.4.1.1. Diagrama de Capas

Este diagrama define cómo organizar el modelo de diseño en capas, lo cual quiere decir que los componentes de una capa sólo pueden hacer referencia a componentes en capas inmediatamente inferiores.

En el diseño de la arquitectura además de las capas general y especificas de la aplicación, se mostraran dos capas más que incluirán los sistemas relacionados con software prefabricado que ejecute y dé soporte a las funcionalidades requeridas por el sistema, tales como sistemas operativos, sistemas de gestión de base de datos, componentes reutilizables, etc. Estas capas son: capa intermedia y capa de software del sistema, y constituyen las bases de la aplicación.

La capa intermedia de la aplicación es aquella que ofrece bloques de construcción reutilizables (paquetes o subsistemas) a marcos de trabajo y servicios independientes de la plataforma.

La capa de software del sistema contiene toda la infraestructura de computación y comunicación, como sistemas operativos y sistemas de gestión de bases de datos. Todas estas capas para el sistema del proyecto en curso se muestran en la en la Figura 4.8.

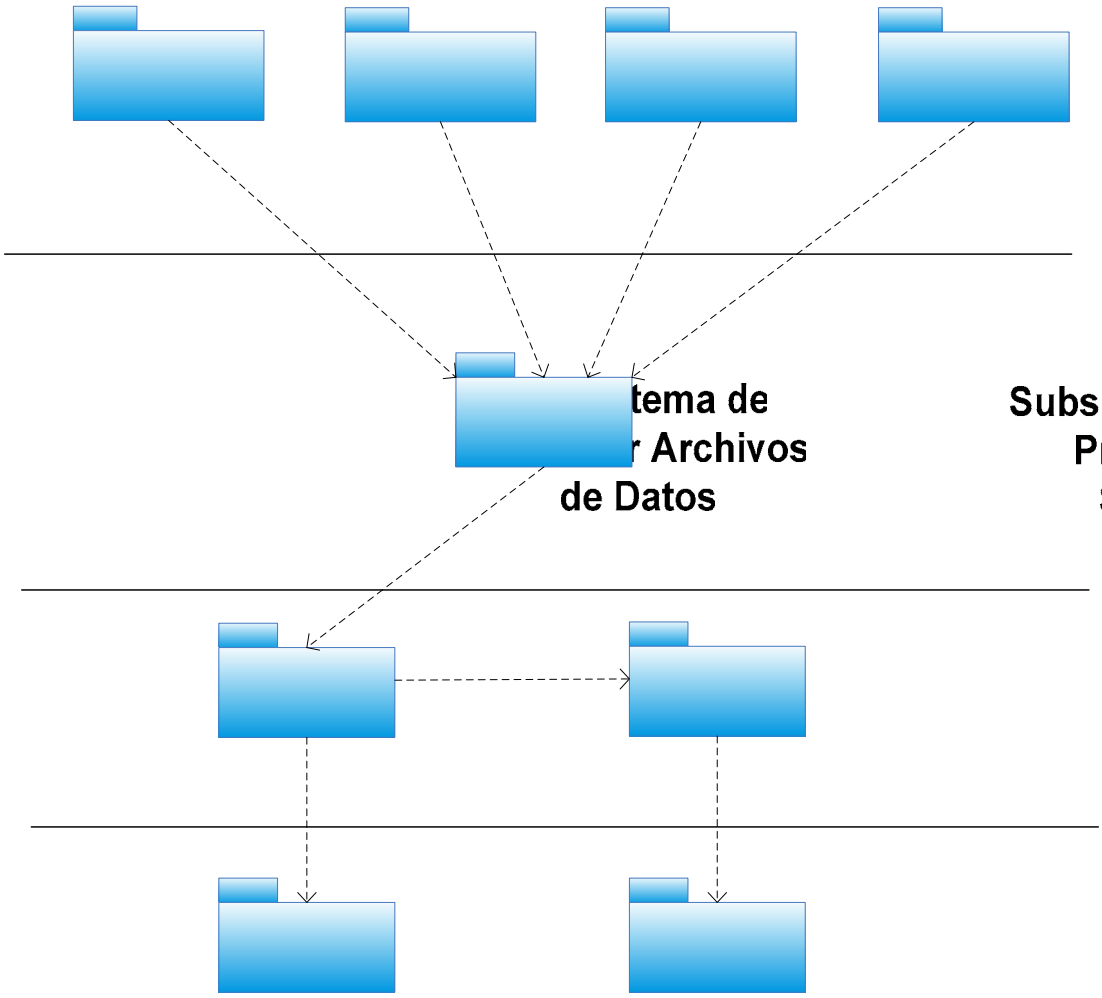


Figura 4.8. Diagrama de Capas de APLICAD – Fuente: Propia

4.4.2. Diagrama de Secuencia

La secuencia de acciones en un caso de uso comienza cuando un actor invoca el caso de uso mediante el envío de algún tipo de mensaje al sistema. Si se considera el interior del sistema, se tendrá algún objeto de diseño que reciba el mensaje del actor.

Después el objeto de diseño puede llamar algún otro objeto y así los objetos implicados interactúan para llevar a cabo el caso de uso. En el diseño es preferible

C++

representar esto con diagramas de secuencia, ya que el centro de atención principal es el encontrar secuencias de interacciones detalladas y ordenadas en el tiempo.

En los diagramas de secuencia se muestran las iteraciones entre los objetos, mediante la transferencia de mensajes entre éstos y los subsistemas. El nombre de cada mensaje dentro del diagrama de secuencia debe indicar una operación del objeto que recibe la invocación o de una interfaz que el objeto proporciona.

A continuación se muestra el diagrama de secuencia para dos casos de uso muy importantes en el sistema APLICAD, con una explicación de los flujos que se llevan a cabo. Dichos diagramas se realizaron a partir de los diagramas de colaboración mostrados en secciones anteriores.

4.4.2.1. Diagrama de Secuencia Para la Realización del Caso de Uso Comprar Productos

En este diagrama (Figura 4.9) se muestra las secuencias de acciones del usuario al interactuar con el sistema, cuando desea comprar un equipo, ingresando una serie de datos correspondientes a los mismos.

4.4.2.2. Diagrama de Secuencia Para la Realización del Caso de Uso Vender Productos y Servicios

En este diagrama (Figura 4.10) se muestra las secuencias de acciones del usuario al interactuar con el sistema, cuando desea vender un equipo y/o servicio, ingresando una serie de datos correspondientes a los mismos.

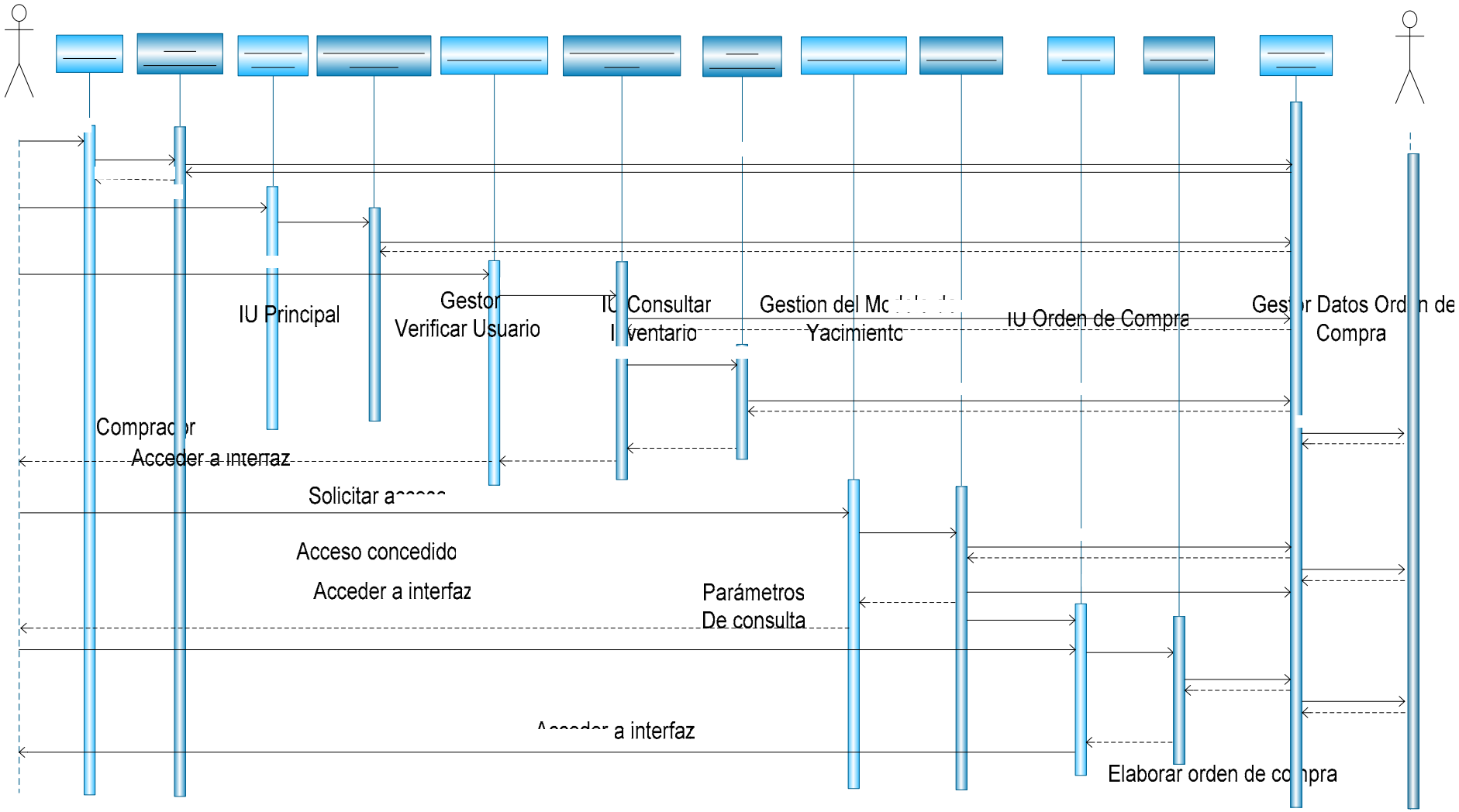


Figura 4.9. Diagrama de Secuencia el caso de Uso Comprar Productos – Fuente: Propia

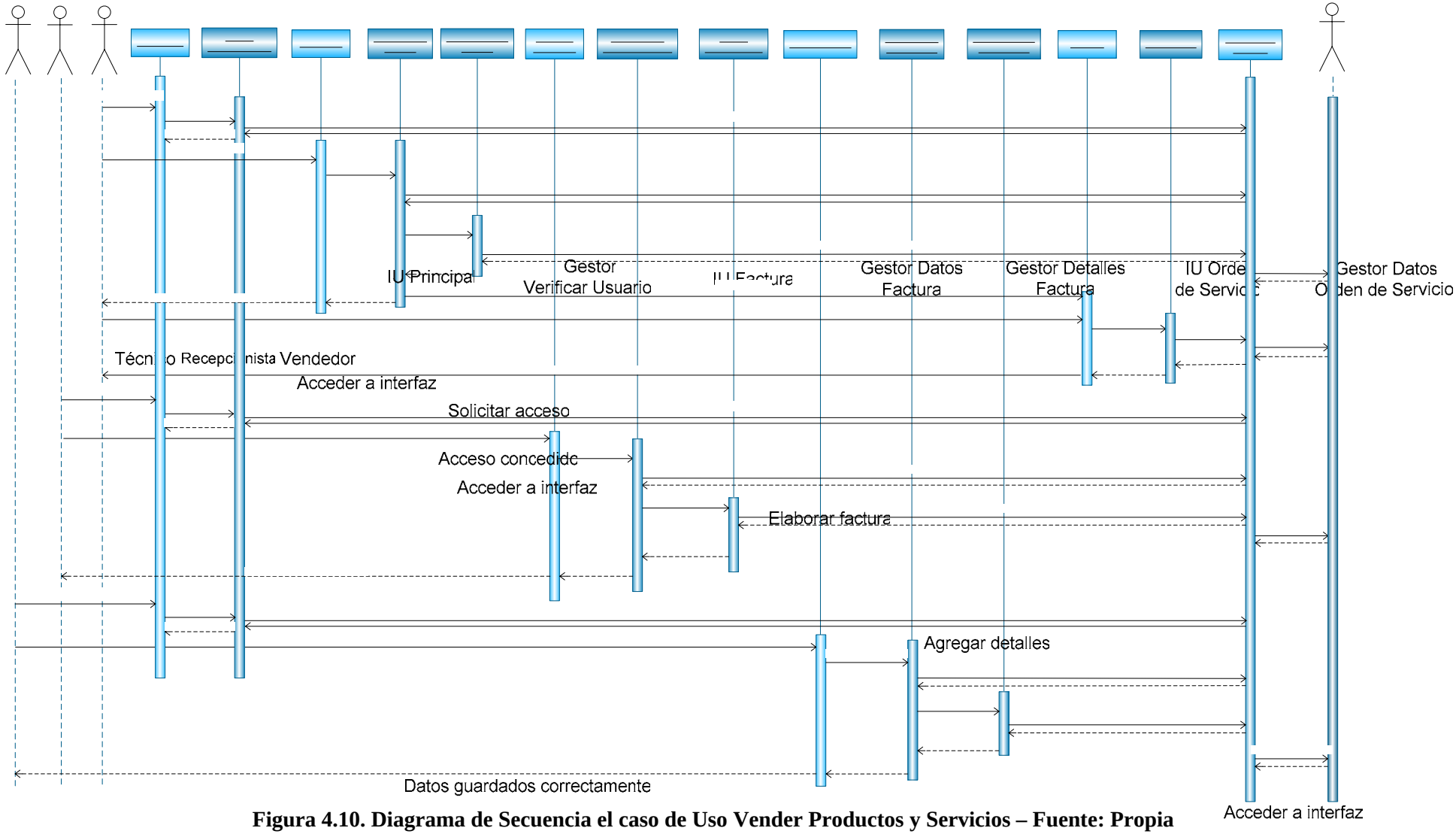


Figura 4.10. Diagrama de Secuencia el caso de Uso Vender Productos y Servicios – Fuente: Propia

4.4.3. Diseño de la Base De Datos Relacional

Conociendo el volumen de datos que podrá manejar el sistema APLICAD, se determinó el diseño de una base de datos relacional BD-APLICAD para el control y gestión de los datos.

Una base de datos relacional es un conjunto de dos o más tablas estructuradas en registros (filas) y campos (columnas), que se vinculan entre sí por un campo en común, en ambos casos poseen las mismas características como por ejemplo el nombre de campo, tipo y longitud; a este campo generalmente se le denomina ID, identificador o clave. A esta manera de construir bases de datos se le denomina modelo relacional, es un modelo de datos basado en la lógica de predicado y en la teoría de conjuntos.

Éste es el modelo más utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente.

Para el diseño de una Base de Datos relacional tienen que tomarse en cuenta las siguientes leyes básicas:

- Una tabla sólo contiene un número fijo de campos.
- El nombre de los campos de una tabla es distinto.
- Cada registro de la tabla es único.
- El orden de los registros y de los campos no están determinados.
- Para cada campo existe un conjunto de valores posible.

4.4.3.1. Modelo Entidad Relación

El modelo entidad relación de la base de datos que maneja el sistema APLICAD se muestra en la Figura 4.11, destacando en adelante los atributos que componen a cada una de las entidades de la base de datos.

- Tabla PROVEEDOR

Contiene toda la información referente a los proveedores de materiales, equipos o servicios que necesite la cooperativa.

Tabla 4.1. Cuadro Descriptivo de los Campos de la Tabla PROVEEDOR.

Nombre en Tabla	Descripción	Tipo de Dato	Tamaño	Propiedades	Comentarios
cod_proveedor	Código de proveedor	int		Autonumérico, Obligatorio	Clave principal
Nombre	Nombre del proveedor	varchar	45	Obligatorio	
Dirección	Dirección del proveedor	varchar	75	Obligatorio	
Identidad	Número de identidad del proveedor	varchar	12	Obligatorio, Único	Cédula para natural y Rif para jurídico
Teléfono	Número de teléfono del proveedor	varchar	30	Obligatorio	Puede contener varios números de teléfono
Fax	Número de fax del proveedor	varchar	30		Puede contener varios números de fax
Objeto	Objeto del Proveedor	varchar	45		A que se dedica el proveedor
Web	Sitio web	varchar	20		
Email	Correo electrónico	varchar	20		
Contacto	Persona de contacto	varchar	25		Este campo es utilizado con los clientes jurídicos
Extensión	Extensión telefónica del contacto	varchar	6		
Celular	Número de teléfono celular del contacto	varchar	12		

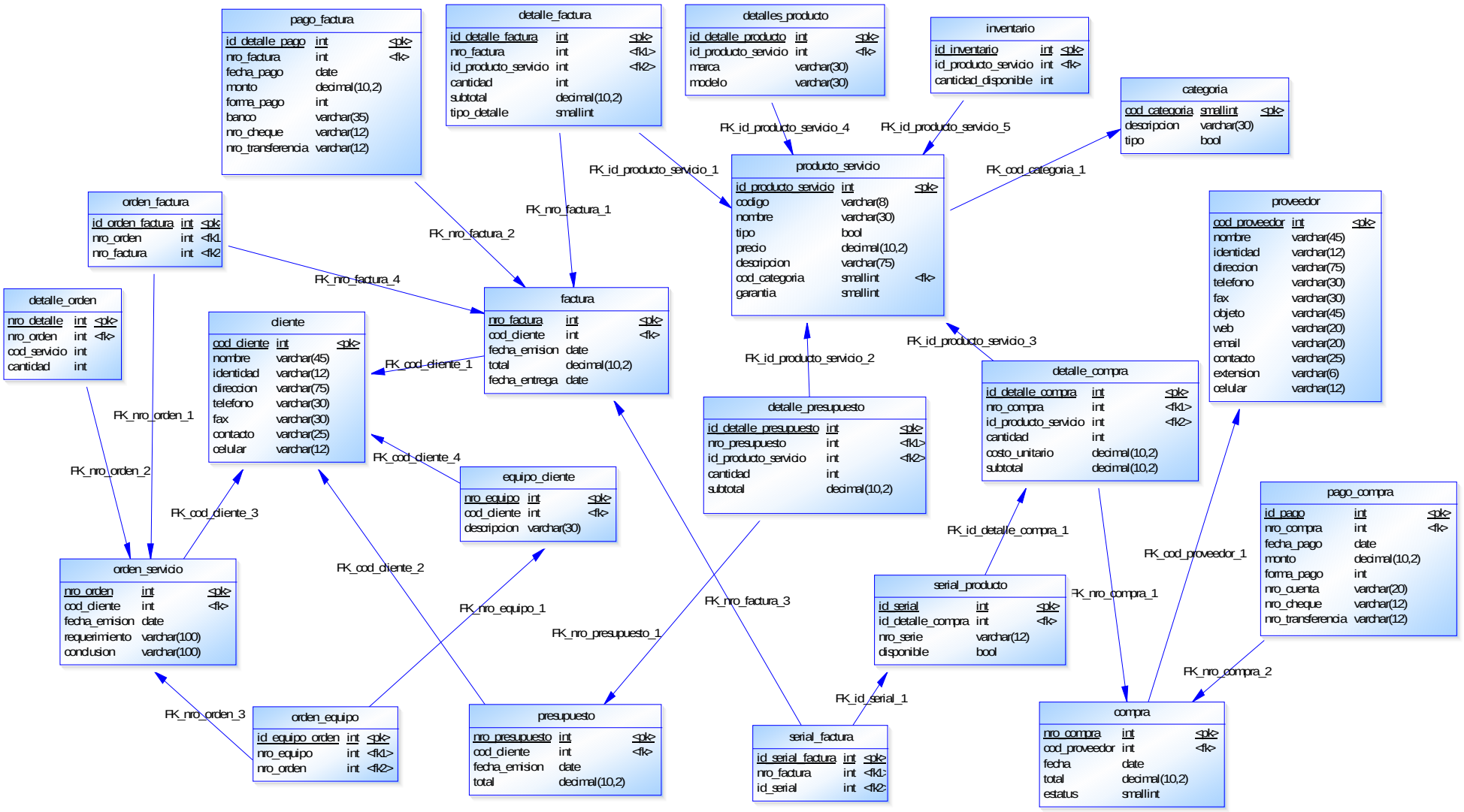


Figura 4.11. Modelo de Entidad –Relación de APLICAD – Fuente: Propia

- Tabla PRODUCTO_SERVICIO

Contiene información referente a los equipos y servicios que vende la cooperativa.

Tabla 4.2. Cuadro Descriptivo de los Campos de la Tabla PRODUCTO_SERVICIO.

Nombre en Tabla	Descripción	Tipo de Dato	Tamaño	Propiedades	Comentarios
id_producto_servicio	Código de producto o servicio	Int		Autonumérico, Obligatorio	Clave principal
Código	Nombre código del producto o servicio	varchar	8	Obligatorio	Sirve como identificador rápido
Nombre	Nombre largo del producto	varchar	30	Obligatorio	Nombre corto del producto
Precio	Precio de venta del producto	decimal	10,2	Obligatorio	Precio referencial para la venta
Tipo	Tipo del producto o servicio	bool		Obligatorio	**0:** Producto **1:** Servicio
Descripción	Descripción detallada del producto	varchar	75		
cod_categoria	Código de categoría del producto	smallint		Obligatorio Relación Foránea	Sirve para saber en que categoría está el producto o servicio. Relación con la tabla CATEGORIA
Garantía	Garantía del producto o servicio	smallint			Meses de garantía

- Tabla INVENTARIO

Esta tabla almacena las cantidades disponibles de los productos que vende SAIT.

Tabla 4.3. Cuadro Descriptivo de los Campos de la Tabla INVENTARIO.

Nombre en Tabla	Descripción	Tipo de Dato	Tamaño	Propiedades	Comentarios
id_inventario	Id del inventario	int		Autonumérico, Obligatorio	Clave principal
id_producto_servio	Id producto asociado	int		Obligatorio, Relación Foránea	Código del producto al que se asocia el registro de inventario. Relación con la tabla PRODUCTO_SERVICIO
cantidad_disponible	Número disponible en inventario de un producto	int			Actualizado por el sistema cuando se compra o vende ítems de un producto

- Tabla SERIAL_PRODUCTO

Esta tabla almacena los números de serie de cada producto que se encuentra en el inventario de la cooperativa. Se relaciona con la tabla DETALLE_COMPRA.

Tabla 4.4. Cuadro Descriptivo de los Campos de la Tabla SERIAL_PRODUCTO.

Nombre en Tabla	Descripción	Tipo de Dato	Tamaño	Propiedades	Comentarios
id_serial	Id del serial del producto	Int		Autonumérico, Obligatorio	Clave principal
id_detalle_compra	Id del detalle de compra asociado	Int		Obligatorio, Relación Foránea	Código del producto al que se asocia el serial. Relación con la tabla DETALLE_COMPRA
nro_serie	Número de serie	varchar	12	Obligatorio	Número de serie marcado en el producto
Disponible	Serial disponible	Bool		Obligatorio	**0:** no disponible, **1:** disponible

- Tabla CATEGORIA

Contiene toda la información referente a la categoría de un producto o servicio.

Tabla 4.5. Cuadro Descriptivo de los Campos de la Tabla CATEGORIA.

Nombre en Tabla	Descripción	Tipo de Dato	Tamaño	Propiedades	Comentarios
cod_categoria	Código de categoría	Int		Autonumérico, Obligatorio	Clave principal
Descripción	Descripción de la categoría	varchar	30	Obligatorio	Descripción de la categoría
Tipo	Tipo categoría	Bool		Obligatorio	Categoría asociada: **0:** Producto, **1:** Servicio

- Tabla ORDEN_SERVICIO

Contiene información referente a las solicitudes de servicio de reparación o mantenimiento de equipos de computación hechas por los clientes. Se relaciona con la tabla CLIENTE.

Tabla 4.6. Cuadro Descriptivo de los Campos de la Tabla ORDEN_SERVICIO.

Nombre en Tabla	Descripción	Tipo de Dato	Tamaño	Propiedades	Comentarios
nro_orden	Número de orden de servicio	int		Autonumérico, Obligatorio	Clave principal
cod_cliente	Código de cliente asociado	int		Obligatorio, Relación Foránea	Código de cliente de la orden de servicio. Relación con la tabla CLIENTE
fecha_emision	Fecha de elaboración de la orden	date		Obligatorio	
Requerimiento	Descripción del requerimiento	varchar	100		Explicación detallada del requerimiento o problema del cliente asociado a un equipo
Conclusión	Conclusiones del técnico sobre el servicio realizado	varchar	100		Conclusiones del técnico sobre el servicio realizado

- Tabla EQUIPO_CLIENTE

En esta tabla se guarda información de los equipos entregados por el cliente. Se relaciona con la tabla CLIENTE.

Tabla 4.7. Cuadro Descriptivo de los Campos de la Tabla EQUIPO_CLIENTE.

Nombre en Tabla	Descripción	Tipo de Dato	Tamaño	Propiedades	Comentarios
nro_equipo	Número de equipo	int		Autonumérico, Obligatorio	Clave principal
cod_cliente	Código del cliente asociado	int		Obligatorio, Relación Foránea	Código del cliente. Relación con la tabla CLIENTE
Descripción	Descripción del equipo	varchar	30	Obligatorio	Se describe el equipo y los accesorios con que se entregó
Serial	Número serial del equipo	varchar	20		Se debe llenar si el equipo tiene un número serial

- Tabla PRESUPUESTO

Contiene información referente a las solicitudes de presupuesto hechas por los clientes sean de venta de equipos, de servicio o mixtos. Se relaciona con la tabla CLIENTE.

Tabla 4.8. Cuadro Descriptivo de los Campos de la Tabla PRESUPUESTO.

Nombre en Tabla	Descripción	Tipo de Dato	Tamaño	Propiedades	Comentarios
nro_presupuesto	Número de orden de servicio	int		Autonumérico, Obligatorio	Clave principal
cod_cliente	Código de cliente asociado	int		Obligatorio, Relación Foránea	Código de cliente al que se le está haciendo la orden de servicio. Relación con la tabla CLIENTE
fecha_emision	Fecha de elaboración del presupuesto	date		Obligatorio	
Total	Total del presupuesto	decimal	10,2		Resultado de la suma de los subtotales de todos los detalles del PRESUPUESTO

- Tabla DETALLE_PRESUPUESTO

Esta tabla contiene los detalles asociados a un presupuesto. Se relaciona con la tabla PRESUPUESTO. Para agregar productos al presupuesto deben existir unidades disponibles en inventario o en transición.

Tabla 4.9. Cuadro Descriptivo de los Campos de la Tabla DETALLE_PRESUPUESTO.

Nombre en Tabla	Descripción	Tipo de Dato	Tamaño	Propiedades	Comentarios
id_detalle_presupuesto	Id del detalle del presupuesto	int		Autonumérico, Obligatorio	Clave principal
Cantidad	Cantidad del producto o servicio	int		Obligatorio	
nro_presupuesto	Número del presupuesto asociado	int		Obligatorio, Relación Foránea	Número del presupuesto al que pertenece el detalle. Relación con tabla PRESUPUESTO
id_producto_servicio	Id del producto o servicio asociado	int		Obligatorio, Relación Foránea	Número del producto_servicio relacionado con el detalle. Relación con la tabla PRODUCTO_SERVICIO
Subtotal	Subtotal	decimal	10,2	Obligatorio	Resultado de la multiplicación de la cantidad por el precio del producto o servicio

- Tabla COMPRA

En esta tabla se guarda información de las compras hechas por la cooperativa a sus proveedores. Se relaciona con la tabla PROVEEDOR. El estatus determina la fase del proceso de compra, solicitud de cotización, ordenación de compra, transición (mientras no llegue la mercancía), finalizada.

Tabla 4.10. Cuadro Descriptivo de los Campos de la Tabla COMPRA.

Nombre en Tabla	Descripción	Tipo de Dato	Tamaño	Propiedades	Comentarios
nro_compra	Número de compra	int		Autonumérico, Obligatorio	Clave principal
cod_proveedor	Código del proveedor asociado	int		Obligatorio, Relación Foránea	Código del proveedor del producto. Relación con la tabla PROVEEDOR
Fecha	Fecha de la compra	date		Obligatorio	
Total	Monto total de la compra	decimal	10,2	Obligatorio	Resultado de la suma de los subtotales de detalles de compra
Estatus	Estatus de la compra	smallint		Obligatorio	**0:** Solicitud, **1:** Ordenación, **2:** Transición, **3:** Finalizada

- Tabla DETALLE_COMPRA

En esta almacena información sobre los detalles asociados a una compra. Se relaciona con las tablas COMPRA y PRODUCTO.

Tabla 4.11. Cuadro Descriptivo de los Campos de la Tabla DETALLE_COMPRA.

Nombre en Tabla	Descripción	Tipo de Dato	Tamaño	Propiedades	Comentarios
id_detalle_compra	Id del detalle de compra	int		Autonumérico, Obligatorio	Clave principal
nro_compra	Número de compra asociado	int		Obligatorio, Relación Foránea	Número de compra al que pertenece el detalle. Relación con la tabla COMPRA
id_producto_servicio	Código del producto asociado	int		Obligatorio, Relación Foránea	Código del producto del detalle. Relación con la tabla PRODUCTO
Cantidad	Cantidad del producto	int		Obligatorio	
costo_unitario	Costo unitario	decimal	10,2	Obligatorio	El precio por unidad
Subtotal	Subtotal	decimal	10,2	Obligatorio	Resultado de la multiplicación de la cantidad por el precio de compra del producto

- Tabla FACTURA

Esta tabla contendrá información de la facturación de los productos y servicios vendidos por la cooperativa. Se relaciona con la tabla CLIENTE.

Tabla 4.12. Cuadro Descriptivo de los Campos de la Tabla FACTURA.

Nombre en Tabla	Descripción	Tipo de Dato	Tamaño	Propiedades	Comentarios
fecha_emision	Fecha de emisión de la factura	date		Obligatorio	Día en que se elabora la factura
nro_factura	Número de factura	int		Autonumérico, Obligatorio	Clave principal,
cod_cliente	Código del cliente asociado	int		Obligatorio, Relación Foránea	Código del cliente. Relación con la tabla CLIENTE
fecha_entrega	Fecha de entrega de la factura	date			Día en que el cliente recibe su factura
Total	Total	decimal	10,2	Obligatorio	Resultado de la suma de los subtotales

- Tabla DETALLE_FACTURA

Contiene información detallada de los productos y/o servicios que se cobrarán a través de la factura. Se relaciona con la tabla FACTURA.

Tabla 4.13. Cuadro Descriptivo de los Campos de la Tabla DETALLE_FACTURA.

Nombre en Tabla	Descripción	Tipo de Dato	Tamaño	Propiedades	Comentarios
id_detalle_factura	Id del detalle de la factura	int		Autonumérico, Obligatorio	Clave principal
id_producto_servicio	Id del producto o servicio asociado	int		Obligatorio, Relación Foránea	Número del producto_servicio relacionado con el detalle. Relación con la tabla PRODUCTO_SERVICIO
nro_factura	Número de factura asociada	int		Obligatorio, Relación Foránea	Factura a la que está asociado el detalle. Relación con la tabla FACTURA
Cantidad	Cantidad del producto o servicio	int		Obligatorio	
Subtotal	Fecha de entrega de la factura	decimal	10,2	Obligatorio	Resultado de la multiplicación de la cantidad por el precio del producto o servicio
tipo_detalle	Tipo de detalle	smallint		Obligatorio	0 servicio, 1 producto

- Tabla PAGO_FACTURA

Contiene información detallada sobre la forma de pago de la factura. Dependiendo como se realice el pago puede haber uno o más registros asociados a la misma factura. Se relaciona con la tabla FACTURA.

Tabla 4.14. Cuadro Descriptivo de los Campos de la Tabla PAGO_FACTURA.

Nombre en Tabla	Descripción	Tipo de Dato	Tamaño	Propiedades	Comentarios
id_pago_factura	Id del detalle del pago de factura	int		Autonumérico, Obligatorio	Clave principal
nro_factura	Número de factura asociada	int		Obligatorio, Relación Foránea	Factura a la que está asociado el detalle. Relación con la tabla FACTURA
fecha_pago	Fecha de pago	date		Obligatorio	Día en que se realizó el pago
Monto	Monto	int		Obligatorio	Monto total o parcial abonado
forma_pago	forma de pago	int		Obligatorio	Forma de pago: 0 Efectivo, 1 Cheque, 2 Transferencia
Banco	Entidad Bancaria	varchar	35		Banco de emisión del cheque.
nro_cheque	Número de Cheque	varchar	12		Número de cheque
nro_transferencia	Número de transferencia o depósito	varchar	12		Número del comprobante de depósito o de transferencia electrónica

- Tabla PAGO_COMPRA

Contiene información detallada sobre la forma de pago de una compra. Dependiendo como se realice el pago puede haber uno o más registros asociados a una misma compra. Se relaciona con la tabla COMPRA.

Tabla 4.15. Cuadro Descriptivo de los Campos de la Tabla PAGO_COMPRA.

Nombre en Tabla	Descripción	Tipo de Dato	Tamaño	Propiedades	Comentarios
id_pago_compra	Id del detalle del pago de compra	int		Autonumérico, Obligatorio	Clave principal
nro_compra	Número de factura asociada	int		Obligatorio, Relación Foránea	Compra a la que está asociado el detalle. Relación con la tabla COMPRA
fecha_pago	Fecha de pago	date		Obligatorio	Día en que se realizó el pago
Monto	Monto	int		Obligatorio	Monto total o parcial abonado
forma_pago	forma de pago	int		Obligatorio	Forma de pago: 0 Efectivo, 1 Cheque, 2 Transferencia
nro_cuenta	Número de Cuenta	varchar	10		Número de cuenta de la cooperativa emisora del cheque o transferencia
nro_cheque	Número de Cheque	varchar	12		Número de cheque de la cooperativa de la compra
nro_trasnferencia	Número de transferencia	varchar	12		Número de transferencia electrónica de la compra

- Tablas de Relación

Para eliminar la redundancia y mejorar la integridad de los datos tenemos una serie de tablas que solo trabajan relacionando las claves principales de otras tablas no manejan información descriptiva sino que relacionan otras tablas.

- Tabla SERIAL_FACTURA

Relaciona serial con factura al momento de vender un producto, para determinar que seriales quedan disponibles y cuales se entregaron con una factura determinada.

Tabla 4.16. Cuadro Descriptivo de los Campos de la Tabla SERIAL_FACTURA.

Nombre en Tabla	Descripción	Tipo de Dato	Tamaño	Propiedades	Comentarios
id_serial_factura	Id de la relación serial factura	int		Autonumérico, Obligatorio	Clave principal
nro_factura	Número de factura asociada	int		Obligatorio, Relación Foránea	Relación con la tabla FACTURA
id_serial	Id del serial del producto	int		Obligatorio, Relación Foránea	Relación con la tabla SERIAL

- Tabla ORDEN_FACTURA

Relaciona una orden de servicio con la factura al momento de facturar un servicio. Se utiliza también para determinar cuando se le hizo servicio a un equipo.

Tabla 4.17. Cuadro Descriptivo de los Campos de la Tabla ORDEN_FACTURA

Nombre en Tabla	Descripción	Tipo de Dato	Tamaño	Propiedades	Comentarios
id_orden_factura	Id de la relación orden factura	int		Autonumérico, Obligatorio	Clave principal
nro_factura	Número de factura asociada	int		Obligatorio, Relación Foránea	Relación con la tabla FACTURA
nro_orden	Número de orden asociada	int		Obligatorio, Relación Foránea	Relación con la tabla ORDEN_SERVICIO

- Tabla ORDEN_EQUIPO

Relaciona una orden de servicio con el equipo del cliente, esto servirá para generar el histórico de servicio de un equipo del cliente.

Tabla 4.18. Cuadro Descriptivo de los Campos de la Tabla ORDEN_EQUIPO

Nombre en Tabla	Descripción	Tipo de Dato	Tamaño	Propiedades	Comentarios
id_orden_equipo	Id de la relación orden equipo	int		Autonumérico, Obligatorio	Clave principal
nro_equipo	Número de equipo asociado	int		Obligatorio, Relación Foránea	Relación con la tabla EQUIPO_CLIENTE
nro_orden	Número de orden asociada	int		Obligatorio, Relación Foránea	Relación con la tabla ORDEN_SERVICIO

4.4.4. Clases de Diseño

Las clases de diseño definen los atributos y operaciones con que debe contar cada clase de análisis para llevar a cabo sus responsabilidades, así como las relaciones existentes entre ellas. Los atributos y operaciones se encuentran en su mayoría inmersos dentro de los diagramas y artefactos obtenidos en los diferentes flujos de trabajo, por lo que se necesita realizar una revisión de estos y agrupar las distintas responsabilidades de las clases para ofrecer soporte a todos sus roles.

El diagrama de diseño permite visualizar las clases que componen el sistema, y la estructura estática de los casos de uso, reflejando las relaciones de generalización, agregación y composición entre clases; lo que permitirá visualizar lo que el sistema puede hacer, además de cómo puede ser construido.

Las clases de diseño son el resultado de la trazabilidad de las clases del análisis, que han sido creadas para lograr una abstracción de los objetos de iguales características y comportamiento. Para la realización del diagrama de clases de diseño de APLICAD, se tomó como base las clases de análisis del modelo de análisis, diseñando nuevas clases que no habían sido consideradas; para así representar completamente la estructura del software.

A continuación, manteniendo el esquema de los capítulos anteriores, se muestran los diagramas de clases de diseño correspondientes a los procesos más importantes de sistema APLICAD como son Compra y Venta de productos y servicios.

4.4.4.1. Diagrama de Clases de Diseño para la Realización del Caso de Uso Vender Productos y Servicios

En el diagrama mostrado en la Figura 4.12 se expone la interacción de las clases de diseño que realizan el caso de uso Vender Productos y Servicios, esta realización está conformada conceptualmente por tres clases activas encargadas del proceso de facturación y de la creación de presupuestos y órdenes de servicio. Se cuenta con 3 clases de interfaz, *IU Factura*, *IU Orden de Servicio* y *IU Presupuesto* las cuales están vinculadas a la interfaz base *IU Principal* y adicionalmente está la *IU Pagos* la cual es activada por el *Gestor Factura* cada vez que se procesa. Los gestores de detalles de factura, orden de servicio y presupuesto son activados toda vez que han sido guardados los datos de la factura, orden de servicio o presupuesto correspondiente.

4.4.4.2. Diseño de Clases para la Realización del Caso de Uso Vender Productos y Servicios

En la Figura 4.13 se detallan las clases de diseño incorporando sus atributos.

Las clases se detallan a continuación:

- **Clase IU Principal**: Representa la interfaz de usuario que permite al usuario acceder a las demás clases de interfaz.
- **Clase IU Factura**: Representa la interfaz de usuario que permite gestionar las facturas de venta de productos y servicios.
- **Clase IU Presupuesto**: Representa la interfaz de usuario que permite gestionar los presupuestos de venta.
- **Clase IU Orden de Servicio**: Representa la interfaz de usuario que permite gestionar las ordenes de servicio.

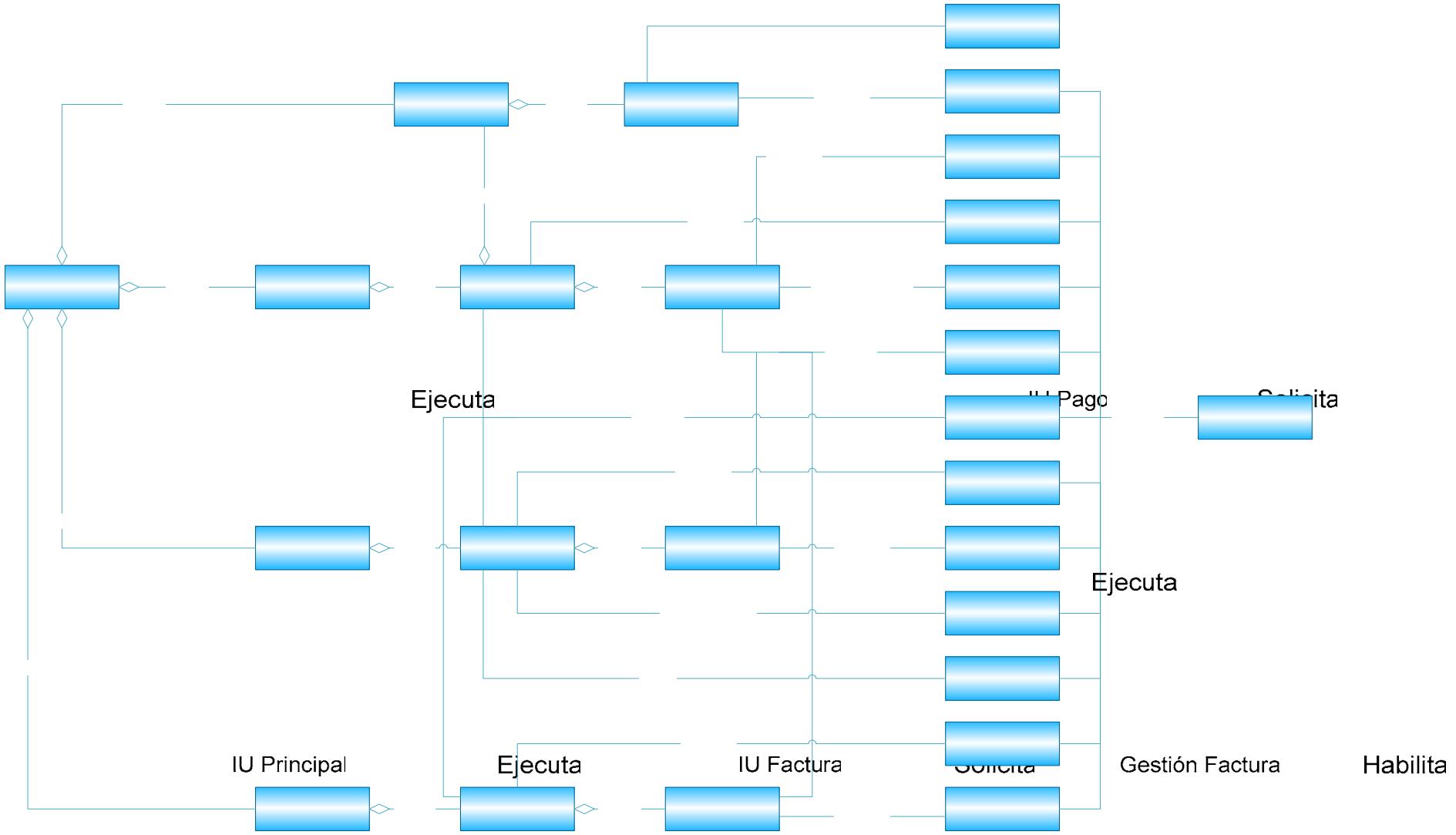


Figura 4.12. Diagrama de Clase de Diseño para la Realización del Caso de Uso Vender Productos y Servicios – Fuente: Propia

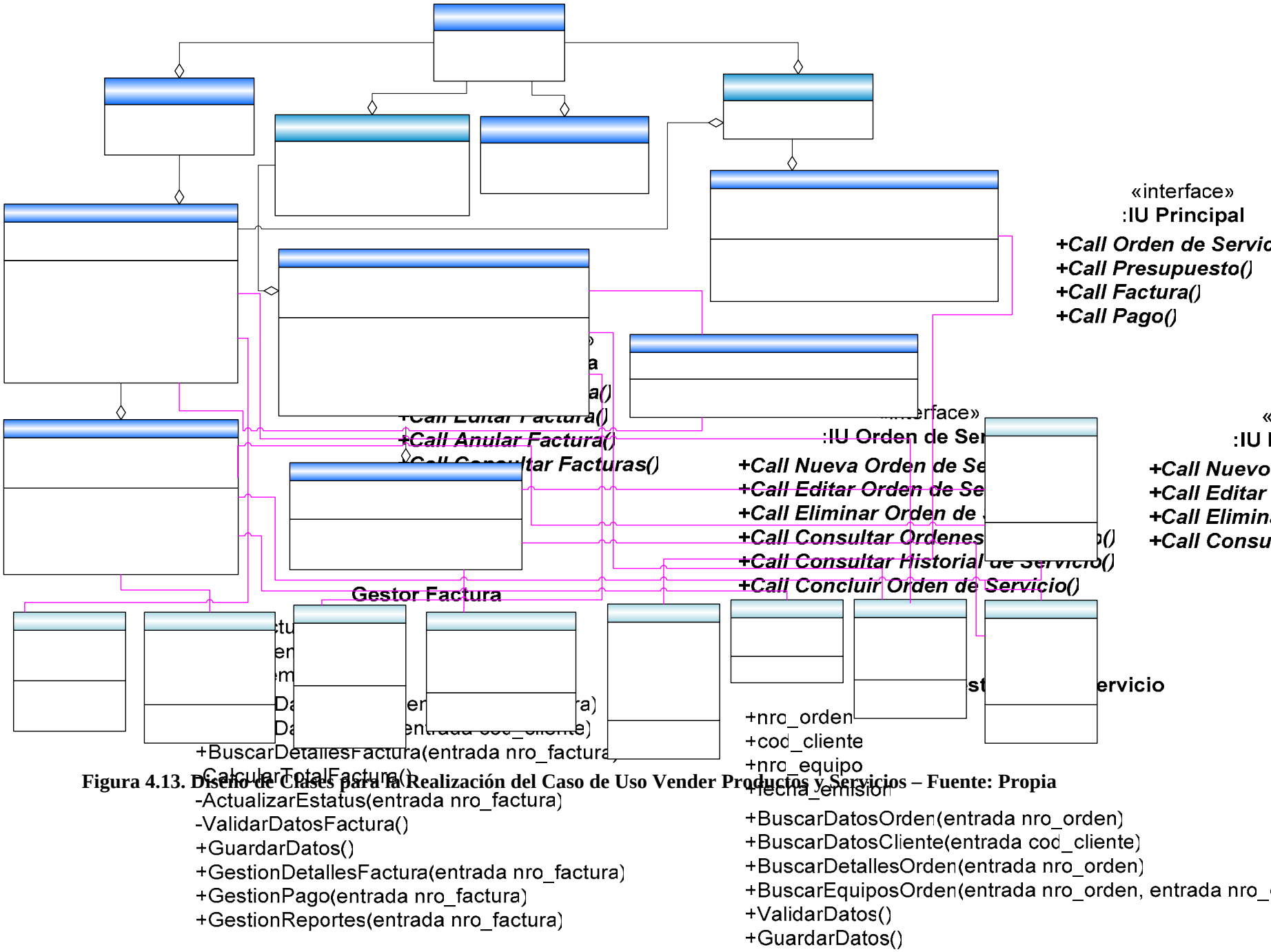


Figura 4.13. Diseño de Clases para la Realización del Caso de Uso Vender Productos y Servicios – Fuente: Propia

- **Clase IU Pago**: Representa la interfaz de usuario que permite gestionar los pagos de las facturas.
- **Clase IU Pago**: Representa la interfaz de usuario que permite gestionar los pagos de las facturas.
- **Clase Gestor Factura**: Esta clase se encarga de realizar las operaciones referentes al proceso de facturación de productos y servicios.
- **Clase Gestor Detalles Factura**: Se encarga de las operaciones asociadas a los detalles de productos y servicios que se incluyen en una factura.
- **Clase Gestor Orden de Servicio**: Esta clase se encarga de realizar y controlar las operaciones referentes al proceso de elaboración de órdenes de servicio.
- **Clase Gestor Pago**: Se encarga de realizar y controlar las operaciones referentes al registro del pago asociado a una factura.
- **Clase Gestor Reportes**: Se encarga de las operaciones necesarias para la generación de reportes y documentos.
- **Clase Gestor Buscar Datos**: Se encarga de resolver las búsquedas de información del sistema necesarias para la realización de otros procesos.
- **Clase Gestor Factura**: Esta clase se encarga de realizar las operaciones referentes al proceso de facturación de productos y servicios.
- **Clase Gestor Detalles Factura**: Se encarga de las operaciones asociadas a los detalles de productos y servicios que se incluyen en una factura.
- **Clase Gestor Orden de Servicio**: Esta clase se encarga de realizar y controlar las operaciones referentes al proceso de elaboración de órdenes de servicio.
- **Clase Gestor Pago**: Se encarga de realizar y controlar las operaciones referentes al registro del pago asociado a una factura.

- **Clase Gestor Reportes**: Se encarga de las operaciones necesarias para la generación de reportes y documentos.
- **Clase Gestor Buscar Datos**: Se encarga de resolver las búsquedas de información del sistema necesarias para la realización de otros procesos.
- **Clase Factura**: Se encarga de llevar a cabo el proceso de administración y control de las facturas de venta.
- **Clase Orden de Servicio**: Se encarga de llevar a cabo el proceso de administración y control de las órdenes de servicio.
- **Clase Pago:** Se encarga de llevar a cabo el proceso de administración y control de los pagos de las facturas de venta.
- **Clase Cliente:** La administración y control de los clientes de la empresa son realizados por esta clase.
- **Clase Producto:** Administra los productos que vende la empresa.
- **Clase Factura**: Se encarga de llevar a cabo el proceso de administración y control de las facturas de venta.
- **Clase Servicio:** Administra los servicios que vende la empresa.
- **Clase Orden de Servicio**: Se encarga de llevar a cabo el proceso de administración y control de las órdenes de servicio.
- **Clase Pago:** Se encarga de llevar a cabo el proceso de administración y control de los pagos de las facturas de venta.
- **Clase Cliente:** La administración y control de los clientes de la empresa son realizados por esta clase.
- **Clase Producto:** Administra los productos que vende la empresa.

- **Clase Inventario:** Lleva la cuenta del número disponibles de productos y se encarga de reponer o descargar dichas cantidades a través de la compra o venta de productos.

4.4.4.3. Diagrama de Clases de Diseño para la Realización del Caso de Uso Comprar Productos

En el diagrama mostrado en la Figura 4.14 se expone la interacción de las clases de diseño que realizan el caso de uso Comprar Productos, esta realización está conformada conceptualmente por tres clases activas encargadas del proceso de facturación y de la creación de presupuestos y órdenes de servicio. Se cuenta con 3 clases de interfaz, *IU Compra*, *IU Orden de Compra* y *IU Inventario* las cuales están vinculadas a la interfaz base *IU Principal* y adicionalmente está la *IU Pagos* la cual es activada por el *Gestor Compra* cada vez que se procesa. Los gestores de detalles de compra y orden de compra son activados toda vez que han sido guardados los datos de la compra, orden de compra correspondiente.

4.4.4.4. Diseño de Clases para la Realización del Caso de Uso Comprar Productos

En la Figura 4.15 se detallan las clases de diseño incorporando sus atributos.

Las clases se detallan a continuación:

- **Clase IU Principal**: Representa la interfaz de usuario que permite al usuario acceder a las demás clases de interfaz.
- **Clase IU Compra**: Representa la interfaz de usuario que permite gestionar las compras de productos.
- **Clase IU Orden de Compra**: Representa la interfaz de usuario que permite gestionar las ordenes de compra.

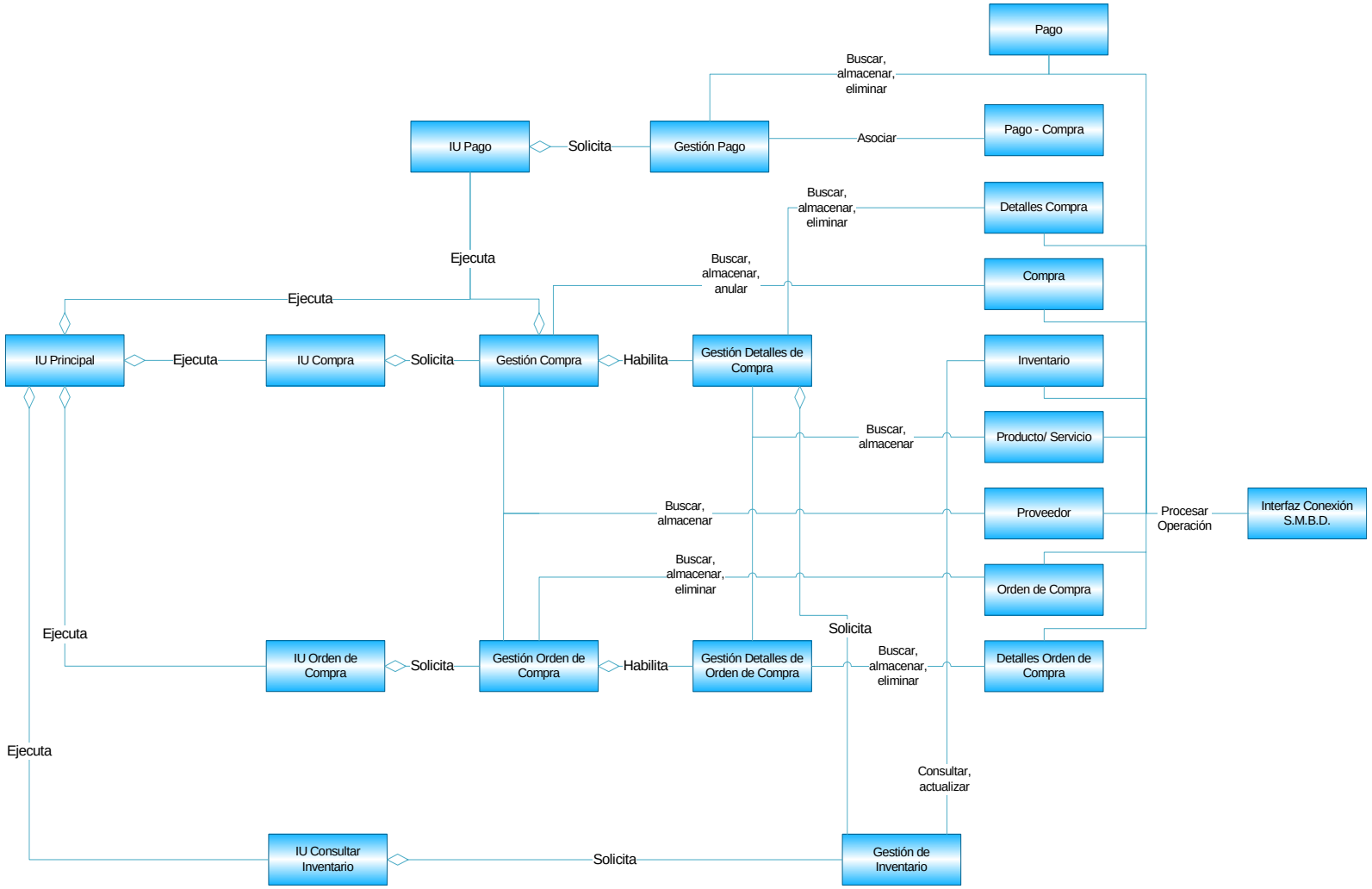


Figura 4.14. Diagrama de Clase de Diseño para la Realización del Caso de Uso Comprar Productos – Fuente: Propia

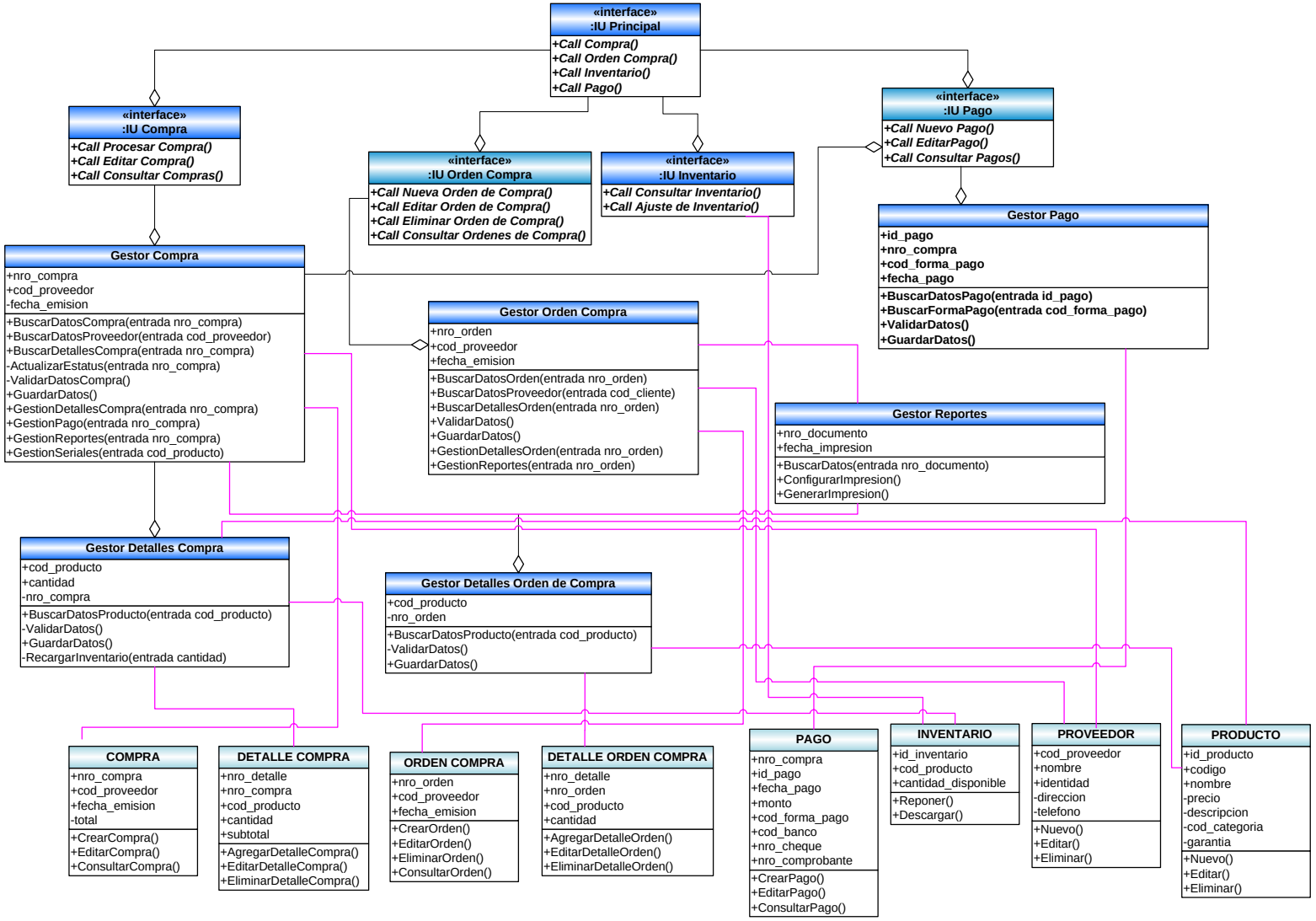


Figura 4.15. Diseño de Clases para la Realización del Caso de Uso Comprar Productos – Fuente: Propia

- **Clase IU Pago**: Representa la interfaz de usuario que permite gestionar los pagos de las compras.
- **Clase Gestor Compra**: Esta clase se encarga de realizar las operaciones referentes al proceso de compra de productos.
- **Clase Gestor Detalles Compra**: Se encarga de las operaciones asociadas a los detalles de productos que se incluyen en una compra.
- **Clase Gestor Orden de Compra**: Esta clase se encarga de realizar y controlar las operaciones referentes al proceso de elaboración de órdenes de compra.
- **Clase Gestor Pago**: Se encarga de realizar y controlar las operaciones referentes al registro del pago asociado a una compra.
- **Clase Gestor Reportes**: Se encarga de las operaciones necesarias para la generación de reportes y documentos.
- **Clase Gestor Buscar Datos**: Se encarga de resolver las búsquedas de información del sistema necesarias para la realización de otros procesos.
- **Clase Compra**: Se encarga de llevar a cabo el proceso de administración y control de la compra de productos.
- **Clase Orden de Compra**: Se encarga de llevar a cabo el proceso de administración y control de las órdenes de compra.
- **Clase Pago:** Se encarga de llevar a cabo el proceso de administración y control de los pagos de las compras.
- **Clase Proveedor:** La administración y control de los proveedores de la empresa son realizados por esta clase.
- **Clase Producto:** Administra los productos que vende la empresa.

- **Clase Inventario:** Lleva la cuenta del número disponibles de productos y se encarga de reponer o descargar dichas cantidades a través de la compra o venta de productos.

4.5. Logros De La Fase De Elaboración

Después de haber completado los flujos de trabajo necesarios para la culminación de la fase de Elaboración se considera que el diseño de las clases y subsistemas, así como el de las base de datos, son suficientemente completos como para ser utilizados en la fase de construcción durante la codificación del software.

CAPÍTULO V

CONSTRUCCIÓN

5.1. Introducción

La fase de construcción tiene como resultado la primera versión del software con capacidad operativa inicial, es durante esta fase cuando el diseño se lleva a la implementación, se codifican las clases diseñadas y, de ser necesario, se complementa dicho diseño modificando la estructura base del software para incorporar los cambios que surgen durante el proceso de construcción.

En esta fase, aunque aún se trabaja en los flujos de trabajo Análisis y Diseño, el mayor esfuerzo se concentra en el flujo Implementación. El objetivo principal es construir la versión beta de APLICAD realizando las primeras pruebas.

La planificación de la fase de construcción se basa en el siguiente esquema:

- Finalizar el análisis y diseño del sistema APLICAD.
- Escoger el lenguaje de programación.
- Codificar la base de datos.
- Codificar el sistema agrupando por subsistemas de implementación de casos de uso.
- Realizar pruebas de cada subsistema de implementación por separado.

5.2. Requisitos

A lo largo de las fases de Inicio y Elaboración se identificaron numerosos requisitos funcionales del sistema, estos fueron añadidos y reflejados en el Modelo de

Caso de Uso a medida que se avanzaba en el diseño y se tenía una mejor comprensión del mismo.

Para la fase de Construcción no fueron identificados nuevos requisitos y por esta razón tampoco se identificaron nuevos actores o casos de uso.

5.3. Análisis

El flujo de trabajo análisis en las fases anteriores se realizó sobre los casos de uso que fueron identificados, sin embargo en la fase de Construcción no se identificaron nuevos casos de uso y por lo tanto no se realizaron nuevos diagramas para el Modelo de Análisis.

5.4. Diseño

El modelo de diseño es un conjunto de objetos que describe la realización física de los casos de uso centrándose en como los requisitos funcionales y no funcionales tienen impacto en el sistema. Además el modelo de diseño sirve de entrada fundamental de las actividades de implementación.

Por lo general en la fase de construcción no se añaden subsistemas de diseño, ni subsistemas de servicio. En general ya se cuenta con la arquitectura base resultado de la fase de Elaboración.

5.5. Implementación

El flujo de trabajo Implementación es el de mayor relevancia en la fase de Construcción, durante éste se construirá la base de datos necesaria para implantar APLICAD y se codificarán todas las clases del sistema, agrupando el desarrollo por subsistemas de implementación en tres construcciones o iteraciones, la primera para construir toda la realización del caso de uso Modelar Yacimiento, la segunda

permitirá codificar todas las clases necesarias en la implementación del caso de uso principal del sistema, es decir, Determinar Comportamiento del Yacimiento y finalmente la tercera iteración dará pie a la implementación de los diversos tipos de ayuda y la integración de todo el sistema.

5.5.1. Escogencia del Lenguaje de programación

Para el desarrollo del sistema APLICAD se debe contar con un lenguaje que soporte la programación orientada a objetos ya que esta promueve una mejor comprensión de los requisitos, diseños más limpios y sistemas mas fáciles de mantener.

Para la construcción de APLICAD se escogió el lenguaje de programación C++ con el entorno integrado de desarrollo (IDE) C++Builder 2007, el cual es una potente herramienta para los programadores ya que permite crear una aplicación beta funcional de forma rápida y sencilla y a su vez basada en un lenguaje sólido como C++. Cuenta con una serie de componentes y librerías las cuales reducen el número de líneas de código, entre estos componentes se pueden mencionar los de conexión con distintos manejadores de bases de datos, controles visuales de edición de datos, manejadores de eventos de aplicación y hasta un generador de reportes que se integra con el IDE y todo hecho para un uso intuitivo y sencillo.

5.5.2. Construcción de la Base de Datos

Para la construcción de la base de datos de APLICAD se escogió MySQL 5.1 que es un servidor de base de datos relacional libre que cumple con todos los estándares del ANSI SQL, su código fuente está disponible para todos sin costo y posee una estabilidad y confiabilidad legendarias.

A continuación se muestra el código SQL correspondiente a la creación de cada la base de datos APLICAD y de cada tabla en su orden de creación.

- **Base de Datos APLICAD**

```
CREATE DATABASE IF NOT EXISTS sait;
ENGINE=InnoDB
DEFAULT CHARSET=latin1;
```

- **Tabla CATEGORIA**

```
CREATE TABLE `categoria` (
  `id_categoria` int(10) unsigned NOT NULL auto_increment,
  `tipo` tinyint(1) default NULL,
  `categoria` varchar(20) NOT NULL,
  PRIMARY KEY  USING BTREE (`id_categoria`)
);
```

- **Tabla CLIENTE**

```
CREATE TABLE `cliente` (
  `cod_cliente` int(11) NOT NULL auto_increment,
  `nombre` varchar(45) NOT NULL,
  `identidad` varchar(12) NOT NULL,
  `direccion` varchar(75) NOT NULL,
  `telefono` varchar(30) NOT NULL,
  `fax` varchar(30) default NULL,
  `contacto` varchar(25) default NULL,
  `celular` varchar(12) default NULL,
  PRIMARY KEY  (`cod_cliente`),
  UNIQUE KEY `IDENTIDAD_2` (`identidad`),
  KEY `NOMBRE_CLIENTE` (`nombre`) );
```

- **Tabla COMPRA**

```
CREATE TABLE `compra` (
 `nro_compra` int(11) NOT NULL auto_increment,
 `cod_proveedor` int(11) NOT NULL,
 `fecha` date NOT NULL,
 `total` decimal(10,2) NOT NULL,
 `estatus` smallint(6) default NULL,
 PRIMARY KEY  (`nro_compra`),
 KEY `FK_cod_proveedor_1` (`cod_proveedor`),
 CONSTRAINT `FK_cod_proveedor_1` FOREIGN KEY (`cod_proveedor`)
REFERENCES `proveedor` (`cod_proveedor`)
);
```

- **Tabla DETALLE_COMPRA**

```
CREATE TABLE `detalle_compra` (
 `id_detalle_compra` int(11) NOT NULL auto_increment,
 `nro_compra` int(11) NOT NULL,
 `cod_ps` varchar(8) NOT NULL,
 `cantidad` int(11) NOT NULL,
 `costo_unitario` decimal(10,2) default NULL,
 `subtotal` decimal(10,2) default NULL,
 PRIMARY KEY  (`id_detalle_compra`),
 KEY `FK_id_producto_servicio_3` (`cod_ps`),
 KEY `FK_nro_compra_1` (`nro_compra`),
 CONSTRAINT `FK_id_producto_servicio_3` FOREIGN KEY (`cod_ps`)
REFERENCES `producto_servicio` (`cod_ps`),
 CONSTRAINT `FK_nro_compra_1` FOREIGN KEY (`nro_compra`)
REFERENCES `compra` (`nro_compra`)
);
```

- **Tabla DETALLE_FACTURA**

```
CREATE TABLE `detalle_factura` (
 `id_detalle_factura` int(11) NOT NULL auto_increment,
 `nro_factura` int(11) NOT NULL,
 `cod_ps` varchar(8) NOT NULL,
 `cantidad` int(11) NOT NULL,
 `subtotal` decimal(10,2) NOT,
 `tipo_detalle` smallint(6) NOT,
 PRIMARY KEY  (`id_detalle_factura`),
 KEY `FK_id_producto_servicio_1` (`cod_ps`),
 KEY `FK_nro_factura_1` (`nro_factura`),
 CONSTRAINT `FK_id_producto_servicio_1` FOREIGN KEY (`cod_ps`)
REFERENCES `producto_servicio` (`cod_ps`),
 CONSTRAINT `FK_nro_factura_1` FOREIGN KEY (`nro_factura`)
REFERENCES `factura` (`nro_factura`)
);
```

- **Tabla DETALLE_ORDEN**

```
CREATE TABLE `detalle_orden` (
 `nro_detalle` int(11) NOT NULL,
 `nro_orden` int(11) NOT NULL,
 `cod_servicio` int(11) NOT NULL,
 `cantidad` int(11) default NULL,
 PRIMARY KEY  (`nro_detalle`),
 KEY `FK_nro_orden_2` (`nro_orden`),
 CONSTRAINT `FK_nro_orden_2` FOREIGN KEY (`nro_orden`) REFERENCES
`orden_servicio` (`nro_orden`)
);
```

- **Tabla DETALLE_PRESUPUESTO**

```
CREATE TABLE `detalle_presupuesto` (
 `id_detalle_presupuesto` int(11) NOT NULL auto_increment,
 `nro_presupuesto` int(11) NOT NULL,
 `cod_ps` varchar(8) NOT NULL,
 `cantidad` int(11) NOT NULL,
 `subtotal` decimal(10,2) NOT NULL,
 PRIMARY KEY  (`id_detalle_presupuesto`),
 KEY `FK_id_producto_servicio_2` (`cod_ps`),
 KEY `FK_nro_presupuesto_1` (`nro_presupuesto`),
 CONSTRAINT `FK_id_producto_servicio_2` FOREIGN KEY (`cod_ps`)
REFERENCES `producto_servicio` (`cod_ps`),
 CONSTRAINT `FK_nro_presupuesto_1` FOREIGN KEY (`nro_presupuesto`)
REFERENCES `presupuesto` (`nro_presupuesto`)
);
```

- **Tabla DETALLE_PRODUCTO**

```
CREATE TABLE `detalle_producto` (
 `id_detalle_producto` int(11) NOT NULL auto_increment,
 `cod_ps` varchar(8) NOT NULL,
 `id_marca` int(10) unsigned default NULL,
 `id_modelo` int(10) unsigned default NULL,
 PRIMARY KEY  (`id_detalle_producto`)
);
```

- **Tabla EQUIPO_CLIENTE**

```
CREATE TABLE `equipo_cliente` (
 `nro_equipo` int(11) NOT NULL auto_increment,
 `cod_cliente` int(11) default NULL,
```

```
  `descripcion` varchar(30) NOT NULL,
  PRIMARY KEY  (`nro_equipo`),
  KEY `FK_cod_cliente_4` (`cod_cliente`),
  CONSTRAINT `FK_cod_cliente_4` FOREIGN KEY (`cod_cliente`)
REFERENCES `cliente` (`cod_cliente`)
);
```

- **Tabla FACTURA**

```
CREATE TABLE `factura` (
  `nro_factura` int(11) NOT NULL auto_increment,
  `cod_cliente` int(11) NOT NULL,
  `fecha_emision` date NOT NULL,
  `total` decimal(10,2) NOT NULL,
  `fecha_entrega` date default NULL,
  PRIMARY KEY  (`nro_factura`),
  KEY `FK_cod_cliente_1` (`cod_cliente`),
  CONSTRAINT `FK_cod_cliente_1` FOREIGN KEY (`cod_cliente`)
REFERENCES `cliente` (`cod_cliente`)
);
```

- **Tabla INVENTARIO**

```
CREATE TABLE `inventario` (
  `id_inventario` int(11) NOT NULL,
  `cod_ps` varchar(8) default NULL,
  `disponibles` int(11) default NULL,
  `en_transicion` int(11) default NULL,
  PRIMARY KEY  (`id_inventario`),
  KEY `FK_id_producto_servicio_5` (`cod_ps`),
```

```
 CONSTRAINT `FK_id_producto_servicio_5` FOREIGN KEY (`cod_ps`)
REFERENCES `producto_servicio` (`cod_ps`)
);
```

- **Tabla ORDEN_EQUIPO**

```
CREATE TABLE `orden_equipo` (
 `id_equipo_orden` int(11) NOT NULL auto_increment,
 `nro_equipo` int(11) default NULL,
 `nro_orden` int(11) default NULL,
 PRIMARY KEY  (`id_equipo_orden`),
 KEY `FK_equipo_cliente_1` (`nro_equipo`),
 KEY `FK_nro_orden_3` (`nro_orden`),
 CONSTRAINT `FK_nro_orden_3` FOREIGN KEY (`nro_orden`) REFERENCES
`orden_servicio` (`nro_orden`),
 CONSTRAINT `FK_equipo_cliente_1` FOREIGN KEY (`nro_equipo`)
REFERENCES `equipo_cliente` (`nro_equipo`)
);
```

- **Tabla ORDEN_FACTURA**

```
CREATE TABLE `orden_factura` (
 `id_orden_factura` int(11) NOT NULL,
 `nro_orden` int(11) default NULL,
 `nro_factura` int(11) default NULL,
 PRIMARY KEY  (`id_orden_factura`),
 KEY `FK_nro_factura_4` (`nro_factura`),
 KEY `FK_nro_orden_1` (`nro_orden`),
 CONSTRAINT `FK_nro_factura_4` FOREIGN KEY (`nro_factura`)
REFERENCES `factura` (`nro_factura`),
```

```
 CONSTRAINT `FK_nro_orden_1` FOREIGN KEY (`nro_orden`) REFERENCES
`orden_servicio` (`nro_orden`)
);
```

- **Tabla ORDEN_SERVICIO**

```
CREATE TABLE `orden_servicio` (
 `nro_orden` int(11) NOT NULL auto_increment,
 `cod_cliente` int(11) NOT NULL,
 `fecha` date NOT NULL,
 `requerimiento` varchar(100) default NULL,
 `conclusion` varchar(100) default NULL,
 PRIMARY KEY  (`nro_orden`),
 KEY `FK_cod_cliente_3` (`cod_cliente`),
 CONSTRAINT `FK_cod_cliente_3` FOREIGN KEY (`cod_cliente`)
REFERENCES `cliente` (`cod_cliente`)
);
```

- **Tabla PAGO_COMPRA**

```
CREATE TABLE `pago_compra` (
 `id_pago` int(11) NOT NULL auto_increment,
 `nro_compra` int(11) NOT NULL,
 `fecha_pago` date NOT NULL,
 `monto` decimal(10,2) NOT NULL,
 `forma_pago` smallint(6) NOT NULL,
 `nro_cuenta` varchar(20) default NULL,
 `nro_cheque` varchar(12) default NULL,
 `nro_transferencia` varchar(12) default NULL,
 PRIMARY KEY  (`id_pago`),
 KEY `FK_nro_compra_2` (`nro_compra`),
```

```
 CONSTRAINT `FK_nro_compra_2` FOREIGN KEY (`nro_compra`)
REFERENCES `compra` (`nro_compra`)
);
```

- **Tabla PAGO_FACTURA**

```
CREATE TABLE `pago_factura` (
 `id_detalle_pago` int(11) NOT NULL auto_increment,
 `nro_factura` int(11) NOT NULL,
 `fecha` date NOT NULL,
 `monto` decimal(10,2) NOT NULL,
 `forma_pago` smallint(6) NOT NULL,
 `banco` varchar(30) default NULL,
 `nro_cheque` varchar(12) default NULL,
 `nro_cuenta` varchar(20) default NULL,
 `nro_transferencia` varchar(12) default,
 PRIMARY KEY  (`id_detalle_pago`),
 KEY `FK_nro_factura_2` (`nro_factura`),
 CONSTRAINT `FK_nro_factura_2` FOREIGN KEY (`nro_factura`)
REFERENCES `factura` (`nro_factura`)
);
```

- **Tabla PRESUPUESTO**

```
CREATE TABLE `presupuesto` (
 `nro_presupuesto` int(11) NOT NULL auto_increment,
 `cod_cliente` int(11) NOT NULL,
 `fecha` date NOT NULL,
 `total` decimal(10,2) default NULL,
 PRIMARY KEY  (`nro_presupuesto`),
 KEY `FK_cod_cliente_2` (`cod_cliente`),
```

```
 CONSTRAINT `FK_cod_cliente_2` FOREIGN KEY (`cod_cliente`)
REFERENCES `cliente` (`cod_cliente`)
);
```

- **Tabla PRODUCTO_SERVICIO**

```
CREATE TABLE `producto_servicio` (
 `cod_ps` varchar(8) NOT NULL,
 `nombre` varchar(30) NOT NULL,
 `tipo` tinyint(1) default '0',
 `precio` decimal(10,2) NOT NULL default '0.00',
 `descripcion` varchar(75) default NULL,
 `garantia` smallint(5) unsigned default NULL,
 `id_categoria` int(10) unsigned default NULL,
 PRIMARY KEY  (`cod_ps`),
 KEY `NOMBRE_PS` (`nombre`),
 KEY `CATEGORIA` USING BTREE (`id_categoria`),
 CONSTRAINT `FK_categoria` FOREIGN KEY (`id_categoria`) REFERENCES
`categoria` (`id_categoria`)
);
```

- **Tabla PROVEEDOR**

```
CREATE TABLE `proveedor` (
 `cod_proveedor` int(11) NOT NULL auto_increment,
 `nombre` varchar(45) NOT NULL,
 `identidad` varchar(12) NOT NULL,
 `direccion` varchar(75) NOT NULL,
 `telefono` varchar(30) NOT NULL,
 `fax` varchar(30) default NULL,
 `objeto` varchar(45) default NULL,
```

```
  `web` varchar(20) default NULL,
  `email` varchar(20) default NULL,
  `contacto` varchar(25) default NULL,
  `extension` varchar(6) default NULL,
  `celular` varchar(12) default NULL,
  PRIMARY KEY  (`cod_proveedor`),
  UNIQUE KEY `IDENTIDAD_1` (`identidad`),
  KEY `NOMBRE_PROVEEDOR` (`nombre`)
);
```

- **Tabla SERIAL_FACTURA**

```
CREATE TABLE `serial_factura` (
  `id_serial_factura` int(11) NOT NULL,
  `nro_factura` int(11) default NULL,
  `id_serial` int(11) default NULL,
  PRIMARY KEY  (`id_serial_factura`),
  KEY `FK_id_serial_1` (`id_serial`),
  KEY `FK_nro_factura_3` (`nro_factura`),
  CONSTRAINT `FK_id_serial_1` FOREIGN KEY (`id_serial`) REFERENCES
`serial_producto` (`id_serial`),
  CONSTRAINT `FK_nro_factura_3` FOREIGN KEY (`nro_factura`)
REFERENCES `factura` (`nro_factura`)
);
```

- **Tabla SERIAL_PRODUCTO**

```
CREATE TABLE `serial_producto` (
  `id_serial` int(11) NOT NULL auto_increment,
  `id_detalle_compra` int(11) default NULL,
  `nro_serie` varchar(12) NOT NULL,
```

```
 `disponible` tinyint(1) NOT NULL default '1',
 PRIMARY KEY  (`id_serial`),
 UNIQUE KEY `NRO_SERIE` (`nro_serie`),
 KEY `FK_id_detalle_compra_1` (`id_detalle_compra`),
 CONSTRAINT `FK_id_detalle_compra_1` FOREIGN KEY (`id_detalle_compra`)
REFERENCES `detalle_compra` (`id_detalle_compra`)
);
```

5.5.3. Planificación de la Fase de Construcción

Para la fase de construcción se consideró realizar dos iteraciones las cuales se desarrollan siguiendo el siguiente esquema:

- Definición del diagrama de componentes.
- Implementación de las clases.

5.5.3.1. Primera Iteración

En esta primera iteración se implementarán los casos de uso correspondientes a Vender Productos y Servicios.

5.5.3.1.1. Diagrama de Componentes

A continuación en la Figura 5.1 se muestra el diagrama de componentes para la implementación del caso de uso Vender Productos y Servicios.

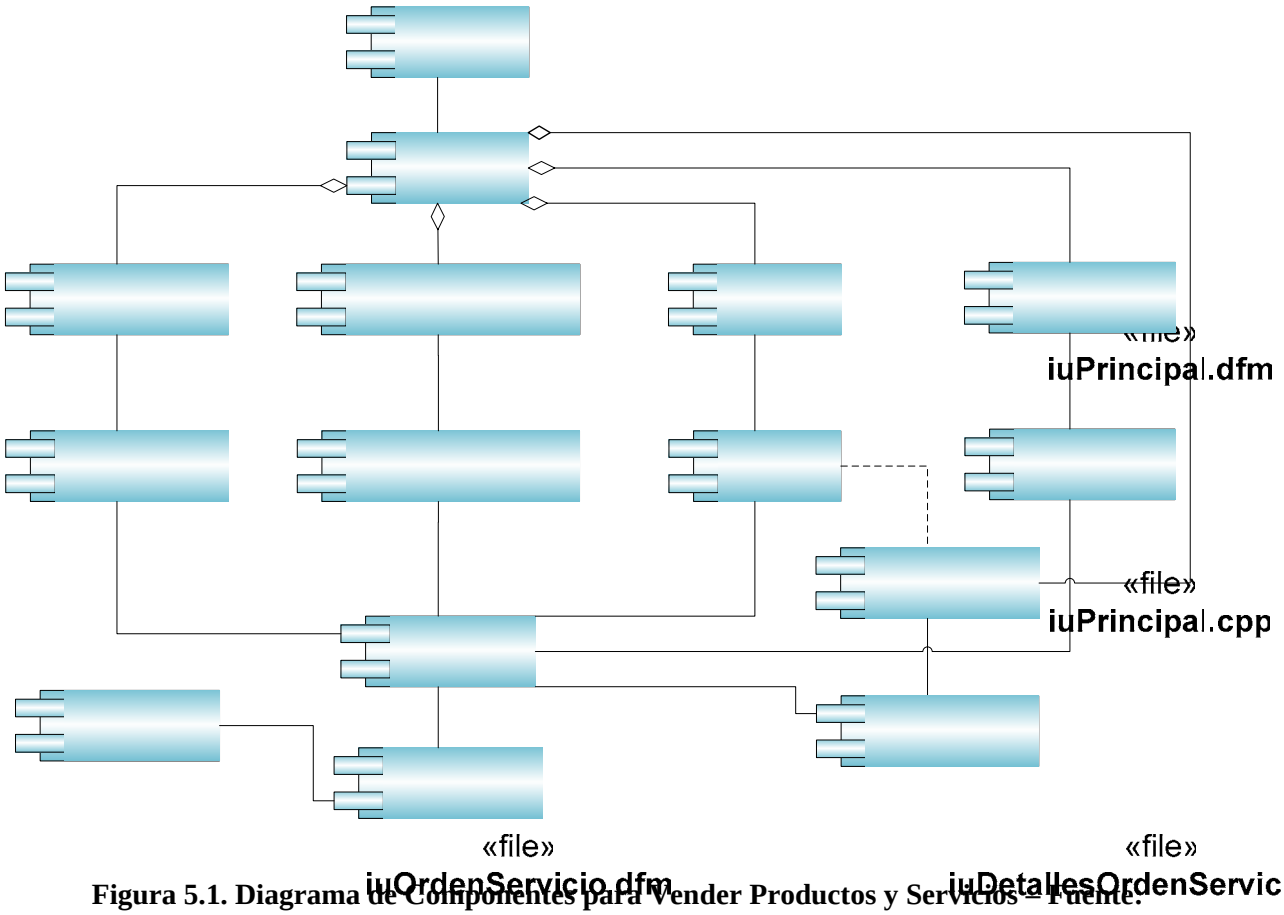


Figura 5.1. Diagrama de Componentes para Vender Productos y Servicios – Fuente: Propia

5.5.3.1.2. Implementación de las Clases

Esta actividad consiste en implementar en código las clases de diseño que fueron definidas al comienzo de este capítulo y que están relacionadas específicamente al caso de uso Vender Productos y Servicios. El código fuente en lenguaje C++ está contenido dentro de ficheros de texto generado a partir de las clases de diseño.

- **Componente: iuOrdenServicio.dfm**

La clase *IU Orden de Servicio* es una clase derivada de la clase TForm de C++Builder y es implementada en el componente **iuOdenServicio.dfm**. La orden de servicio es elaborada por la recepción y en un proceso posterior el técnico se encargará de llenar los detalles de dicha orden de servicio.

Vista de la ventana:

Figura 5.2. Vista de la ventana de iuOrdenServicio – Fuente: Propia

La clase de diseño *Gestor Orden de Servicio* está implementada en el código del componente **iuOrdenServicio.cpp**; a continuación de muestra parte del código fuente de este gestor.

```
void __fastcall TfOrdenServicio::FormCreate(TObject *Sender)
{
	lbOrden->Caption=fDatos->tbOrdenServicio->RecordCount+1;
	if(!fDatos->tbOrdenServicio->Active)
		fDatos->tbOrdenServicio->Open();
	if(!fDatos->tbEquipoOrden->Active)
		fDatos->tbEquipoOrden->Open();
	if(!fDatos->tbEquipoCliente->Active)
		fDatos->tbEquipoCliente->Open();
	if(!fDatos->tbCliente->Active)
		fDatos->tbCliente->Open();
```

```
}
void __fastcall TfOrdenServicio::DBNavigator1Click(TObject *Sender,
        TNavigateBtn Button)
      if(Button==nbInsert)
      {
            if(fDatos->tbOrdenServicio->State==dsInsert)
            {
                  fDatos->tbOrdenServicio->FieldByName("fecha")-
>Value=decodificarFecha(dtFecha);
                  fDatos->tbOrdenServicio->Post();
            }
            fDatos->tbEquipoOrden->FieldByName("nro_equipo")-
>Value=fDatos->tbEquipoCliente->FieldByName("nro_equipo")->Value;
            fDatos->tbEquipoOrden->FieldByName("nro_orden")->Value=fDatos-
>tbOrdenServicio->FieldByName("nro_orden")->Value;
            cbEquipo->Enabled=true;
            btEquipo->Enabled=true;
      }
      else if(Button==nbPost)
      {
            equipos->Active=false;
            equipos->ParamByName("orden")->Value=lbOrden->Caption;
            equipos->ExecSQL();
            equipos->Active=true;
            cbEquipo->Enabled=false;
            btEquipo->Enabled=false;
      }
void __fastcall TfOrdenServicio::btGuardarClick(TObject *Sender)
{
      try

      {
            if(fDatos->tbOrdenServicio->State!=dsEdit||fDatos-
>tbOrdenServicio->State!=dsInsert)
                  fDatos->tbOrdenServicio->Edit();
            fDatos->tbOrdenServicio->FieldByName("fecha")-
>Value=decodificarFecha(dtFecha);
            fDatos->tbOrdenServicio->Post();
            Close();
```

```
        }
        catch(Exception *E)
        {
                ShowMessage("Ha ocurrido un error de bd: "+E->Message);
        }
}
void __fastcall TfPrincipal::ApplicationEvents1Exception(TObject *Sender,
      Exception *E)
{
    if(AnsiContainsStr(E->Message,"is not a valid"))
    {
        TEdit *Texto = (TEdit*) Sender;
             Texto->SetFocus();
             Texto->Clear();
        }

        else if(AnsiContainsStr(E->Message,"must have a value"))
        {
                ShowMessage("Debe completar los datos esenciales");
        }
        else if(AnsiContainsStr(E->Message,"Out of range"))
        {
                ShowMessage("Valor fuera del rango establecido");
        }
        else if(AnsiContainsStr(E->Message,"Key violation"))
        {
                ShowMessage("El registro ya existe, ingrese otro código");
        }
}
```

- **Componente: iuFactura.dfm**

La clase *IU Factura* es una clase derivada de la clase TForm de C++Builder y es implementada en el componente **iuFactura.dfm**. Este componente está conformado por la grilla de los detalles de factura y controles para el ingreso de los datos, también tiene acceso directo a la creación de un cliente si éste no se encuentra registrado y la opción de liquidar la factura que llama directamente a la *IU Pago*.

Vista de la ventana:

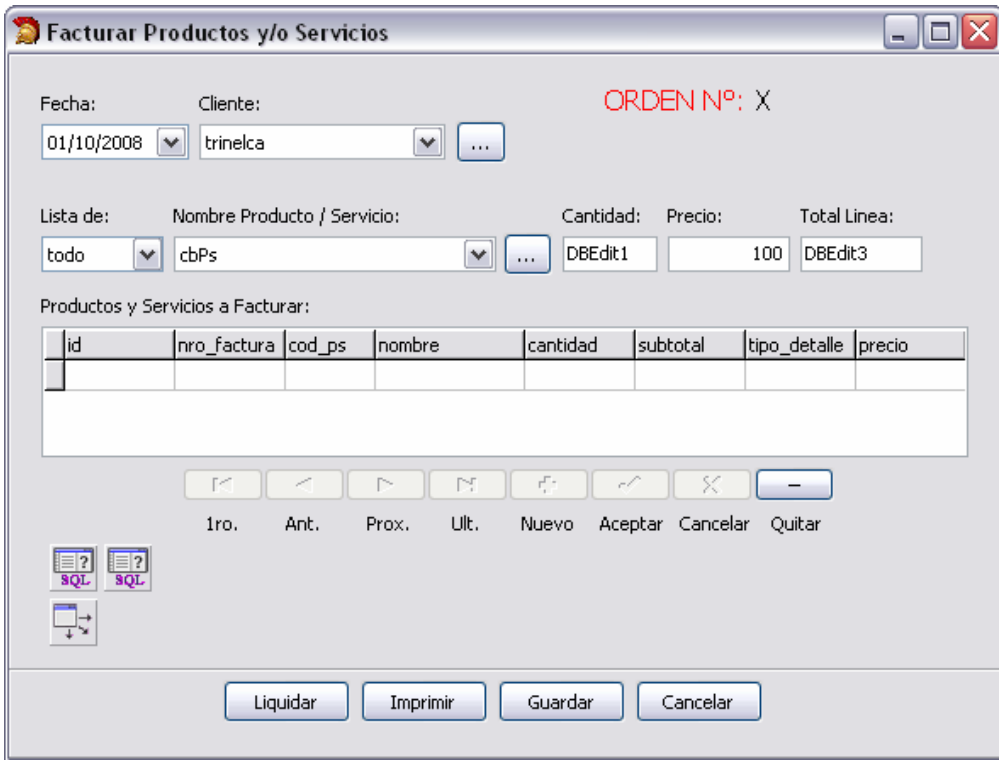


Figura 5.3. Vista de la ventana de iuFactura – Fuente: Propia

La clase de diseño Gestor Factura está implementada en el código del componente **iuFactura.cpp** cuenta con diversos cálculos de totales, impuestos y descuentos, y rutinas de búsqueda y validación de los datos, a continuación se puede apreciar una de las funciones mas importantes de este gestor.

```
void __fastcall TfDatos::tbDetallesFacturaAfterPost(TDataSet *DataSet)
{
      tbFactura->Edit();
      tbFacturatotal->Value=tbFacturatotal-
>Value+tbDetallesFacturasubtotal->Value;
}
void __fastcall TfDatos::tbDetallesFacturaBeforePost(TDataSet *DataSet)
{
```

```
	tbDetallesFacturanro_factura->Value=tbFacturanro_factura->Value;
	tbDetallesFacturatipo_detalle->Value=tbPstipo->Value;
	tbDetallesFacturacod_ps->Value=fFactura->lbFactura->Caption;
}
void __fastcall TfDatos::tbDetallesFacturaBeforeInsert(TDataSet *DataSet)
{
	tbFactura->Post();
}
void __fastcall TfDatos::tbFacturaBeforePost(TDataSet *DataSet)
{
	tbFacturafecha_emision->Value=decodificarFecha(fFactura->dtFecha);
}
void __fastcall TfDatos::tbFacturaAfterInsert(TDataSet *DataSet)
{
	tbFacturatotal->Value=0;
}
void __fastcall TfFactura::FormCreate(TObject *Sender)
{
	lbFactura->Caption=fDatos->tbFactura->RecordCount+1;
	if(!fDatos->tbFactura->Active)
		fDatos->tbFactura->Open();
	if(!fDatos->tbDetallesFactura->Active)
		fDatos->tbDetallesFactura->Open();
	if(!fDatos->tbPs->Active)
		fDatos->tbPs->Open();
	if(!fDatos->tbCliente->Active)
		fDatos->tbCliente->Open();
}
void __fastcall TfFactura::DBEdit1Change(TObject *Sender)
{
	if(fDatos->tbDetallesFactura->FieldByName("cantidad")>0)
		fDatos->tbDetallesFactura->FieldByName("subtotal")-
>Value=fDatos->tbDetallesFactura->FieldByName("cantidad")->Value*fDatos-
>tbPs->FieldByName("precio")->Value;
}
void __fastcall TfFactura::btGuardarClick(TObject *Sender)
{
	try
	{
```

```
            fDatos->tbFactura->Post();
            Close();
        }
        catch(Exception *E)
        {
            ShowMessage("Ha ocurrido un error de bd: "+E->Message);
        }
}
```

- **Componente: iuPresupuesto.dfm**

La clase IU Presupuesto es derivada de la clase TFactura por la similitud de ambas y hereda parte de las funciones y características de la clase Factura; es implementada en el componente **iuPresupuesto.dfm**.

Vista de la ventana:

Figura 5.4. Vista de la ventana de iuPresupuesto – Fuente: Propia

La clase de diseño Gestor Presupuesto también está implementada en el código del componente **iuPresupuesto.cpp**. El código tiene gran similitud al mostrado en el componente anterior así que será omitido en este caso.

- **Componente: iuRegistroPago.dfm**

La clase *IU Pago* es una clase derivada de la clase TForm de C++Builder y es implementada en el componente **iuRegistroPago.dfm**. Para llegar aquí hay dos maneras una de ellas es al momento de la elaboración de la factura en la opción de liquidar; la otra es usando el menú principal y acceder directamente a la interfaz indicando antes cual factura se va a pagar.

Vista de la ventana:

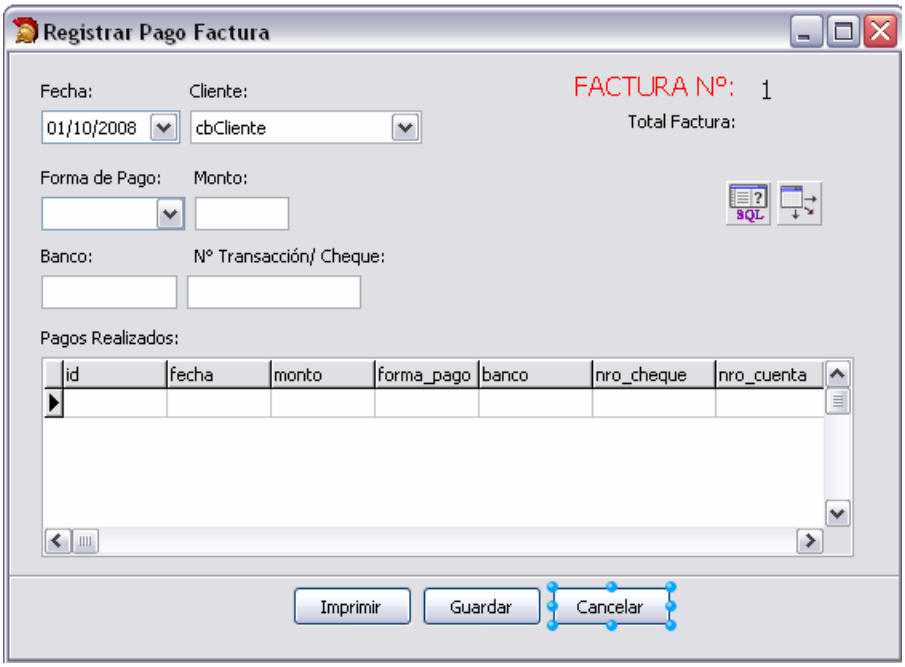


Figura 5.5. Vista de la ventana de iuRegistrarPago – Fuente: Propia

Como en los casos anteriores, la clase de diseño Gestor Pago está implementada en el código del componente **iuRegistroPago.cpp**, y a continuación se presenta parte del código del mismo.

```
void __fastcall TfPagoFactura::btGuardarClick(TObject *Sender)
{
      try
      {
            fDatos->tbPagoFactura->FieldByName("fecha")->Value=fDatos-
>decodificarFecha(dtFecha);
            fDatos->tbPagoFactura->Post();
            if(Application->MessageBox("¿Desea  Imprimir  pago?","Se  ha
registrado el pago correctamente", MB_OKCANCEL)==1)
            {
                  imprimirPago(fDatos->tbPagoFactura);
            }
      }
      catch(Exception *E)
      {
            ShowMessage("Hubo el siguiente error de bd: "+E->Message);
      }
}

void __fastcall TfPagoFactura::cbFormaPagoChange(TObject *Sender)
{
      if(cbFormaPago->ItemIndex<4)
      {
            DBEdit3->Enabled=true;
            DBEdit4->Enabled=true;
      }
      else //efectivo
      {
            DBEdit3->Enabled=false;
            DBEdit4->Enabled=false;
      }
}

void __fastcall TfPagoFactura::FormCreate(TObject *Sender)
```

```
{
	if(!fDatos->tbPagoFactura->Active)
		fDatos->tbPagoFactura->Open();
	if(!fDatos->tbFactura->Active)
		fDatos->tbFactura->Open();

	pagos->Active=false;
	pagos->SQL->Clear();
	pagos->SQL->Add("select * from pago_factura"
				"where nro_factura="
				+fDatos->tbFactura-
>FieldByName("nro_factura")->Value);
}

void __fastcall TfPagoFactura::btCancelarClick(TObject *Sender)
{
	fDatos->tbPagoFactura->Cancel();
	Close();
}
void __fastcall TfPagoFactura::btImprimirClick(TObject *Sender)
{
	imprimirPago(fDatos->tbPagoFactura);
}
```

5.5.3.2. Segunda Iteración

En la segunda iteración se implementarán los casos de uso correspondientes a Comprar Productos.

5.5.3.2.1. Diagrama de Componentes

A continuación en la Figura 5.6 se muestra el diagrama de componentes para la implementación del caso de uso Comprar Productos.

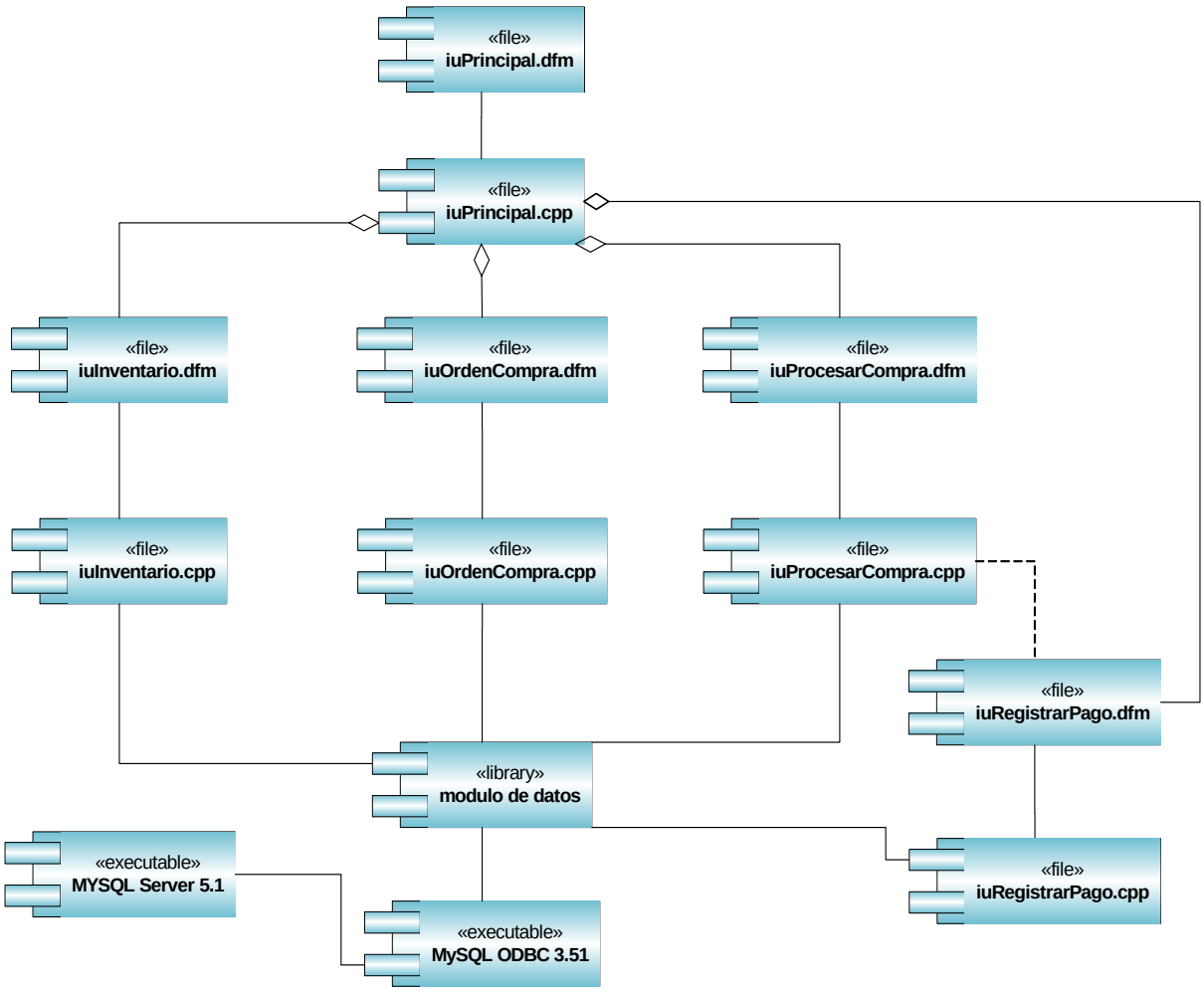


Figura 5.6. Diagrama de Componentes para Comprar Productos – Fuente: Propia

5.5.3.2.2. Implementación de las Clases

- **Componente: iuInventario.dfm**

La clase *IU Inventario* es una clase derivada de la clase TForm de C++Builder y es implementada en el componente **iuInventario.dfm**. Este componente incluye una grilla de visualización de los productos que vende SAIT, RL y las cantidades disponibles, también se pueden hacer búsquedas y filtrar información.

Vista de la ventana:

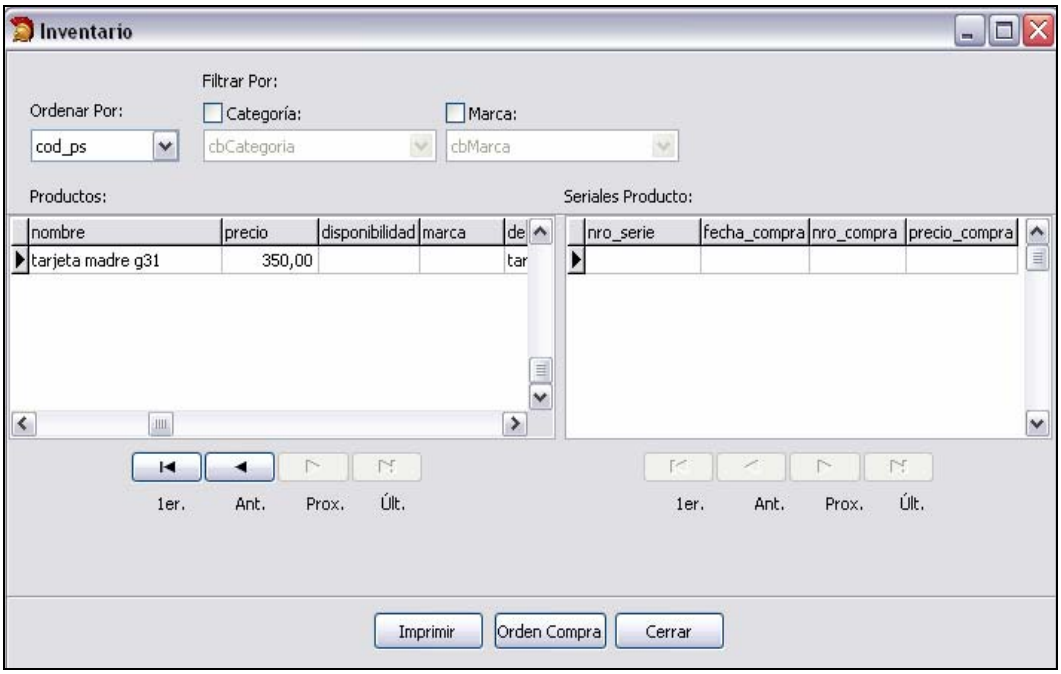


Figura 5.7. Vista de la ventana de iuInventario – Fuente: Propia

La clase de diseño *Gestor Inventario* está implementada en el código del componente **iuInventario.cpp** y se encarga de las descargas y reposiciones de inventario; a continuación de muestra parte del código fuente de este gestor.

```
void __fastcall TfInventario::btOrdenCompraClick(TObject *Sender)
{
            if(!Application->FindComponent("fOrdenCompra"))
            Application->CreateForm(__classid(TfOrdenCompra),
&fOrdenCompra);
            fOrdenCompra->ShowModal();
}

void __fastcall TfInventario::cbOrdenarChange(TObject *Sender)
{
  fDatos->tbOrdenServicio->IndexName=fDatos->tbOrdenServicio->IndexDefs-
>Items[cbOrdenar->ItemIndex]->Name;
}
```

```
void __fastcall TfInventario::ckCategoriaClick(TObject *Sender)
{
      if(ckCategoria->Checked)
            cbCategoria->Enabled=true;
      else if(!ckCategoria->Checked)
            cbCategoria->Enabled=false;
}


void __fastcall TfInventario::ImprimirClick(TObject *Sender)
{
      imprimirListado(fDatos->tbInventario);
}


void __fastcall TfInventario::FormCreate(TObject *Sender)
{
      if(fDatos->tbCategoria->Active)
            fDatos->tbCategoria->Open();
      if(fDatos->tbInventario->Active)
            fDatos->tbInventario->Open();
      if(fDatos->tbSerialProducto->Active)
            fDatos->tbSerialProducto->Open();
}


void __fastcall TfInventario::FormClose(TObject *Sender, TCloseAction
&Action)
{
      fInventario->Release();
}
```

- **Componente: iuOrdenCompra.dfm**

La clase *IU Orden de Compra* es una clase derivada de la clase TForm de C++Builder y es implementada en el componente **iuOrdenCompra.dfm**, permite cargar el pedido de productos a un proveedor.

Vista de la ventana:

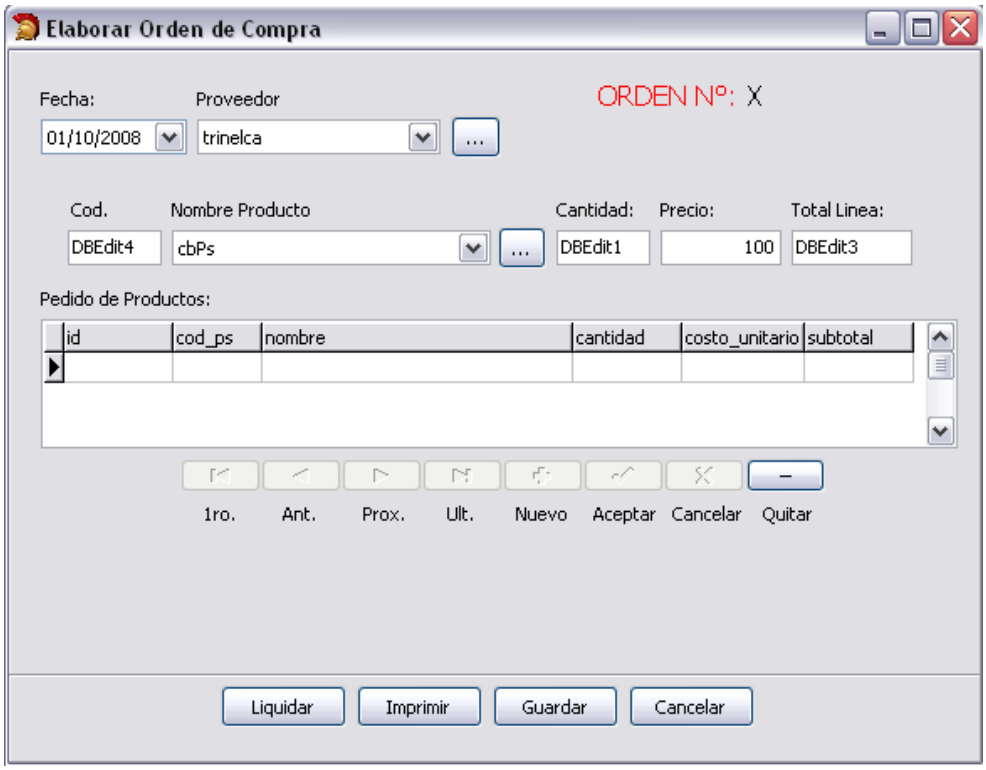


Figura 5.8. Vista de la ventana de iuOrdenCompra – Fuente: Propia

La clase de diseño Gestor Orden de Compra está implementada en el código del componente **iuOrdenCompra.cpp**, a continuación se puede apreciar una de las funciones mas importantes de este gestor.

```
void __fastcall TfOrdenCompra::Button1Click(TObject *Sender)
{
      if(!Application->FindComponent("fProvedores"))
            Application->CreateForm(__classid(TfProveedores),
&fProveedores);
      fProveedores->ShowModal();
}


void __fastcall TfOrdenCompra::Button5Click(TObject *Sender)
```

```
{
	imprimirDocumentoo(fDatos->tbOrdenCompra);
}


void __fastcall TfOrdenCompra::Button6Click(TObject *Sender)
{
	fPagarCompra->ShowModal();
}

void __fastcall TfOrdenCompra::Button3Click(TObject *Sender)
{
	Close();
}

void __fastcall TfOrdenCompra::dtFechaChange(TObject *Sender)
{
	if(fDatos->tbOrdenCompra->State!=dsEdit||fDatos->tbOrdenCompra-
>State!=dsInsert)
		fDatos->tbOrdenCompra->Edit();
	fDatos->tbOrdenCompra->FieldByName("fecha_emision")->Value=fDatos-
>decodificarFecha(dtFecha);
}


void __fastcall TfOrdenCompra::FormCreate(TObject *Sender)
{
	lbOrden->Caption=fDatos->tbOrdenCompra->RecordCount+1;
	if(!fDatos->tbOrdenCompra->Active)
		fDatos->tbOrdenCompra->Open();
	if(!fDatos->tbDetallesOrdenCompra->Active)
		fDatos->tbDetallesOrdenCompra->Open();
	if(!fDatos->tbPs->Active)
		fDatos->tbPs->Open();
	if(!fDatos->tbCliente->Active)
		fDatos->tbCliente->Open();
}
```

```
void __fastcall TfOrdenCompra::DBEdit1Change(TObject *Sender)
{

      if(fDatos->tbDetallesOrdenCompra->FieldByName("cantidad")>0)
            fDatos->tbDetallesOrdenCompra->FieldByName("subtotal")-
>Value=fDatos->tbDetallesOrdenCompra->FieldByName("cantidad")-
>Value*fDatos->tbPs->FieldByName("precio")->Value;

}


void __fastcall TfOrdenCompra::FormClose(TObject *Sender, TCloseAction
&Action)
{
      fOrdenServicio->Release();
}

void __fastcall TfOrdenCompra::btGuardarClick(TObject *Sender)
{
      try
      {
            fDatos->tbOrdenServicio->Post();
            ShowMessage("Registrado Correctamente");
            Close();
      }

      catch(Exception *E)
      {
            ShowMessage("Ha ocurrido un error de bd: "+E->Message);
      }
}
```

- **Componente: iuProcesarCompra.dfm**

La clase IU Procesar Compra es derivada de la clase TForm; es implementada en el componente **iuProcesarCompra.dfm**. Para procesar la compra se debe seleccionar una orden de compra y procesar el pago de la misma.

Vista de la ventana:

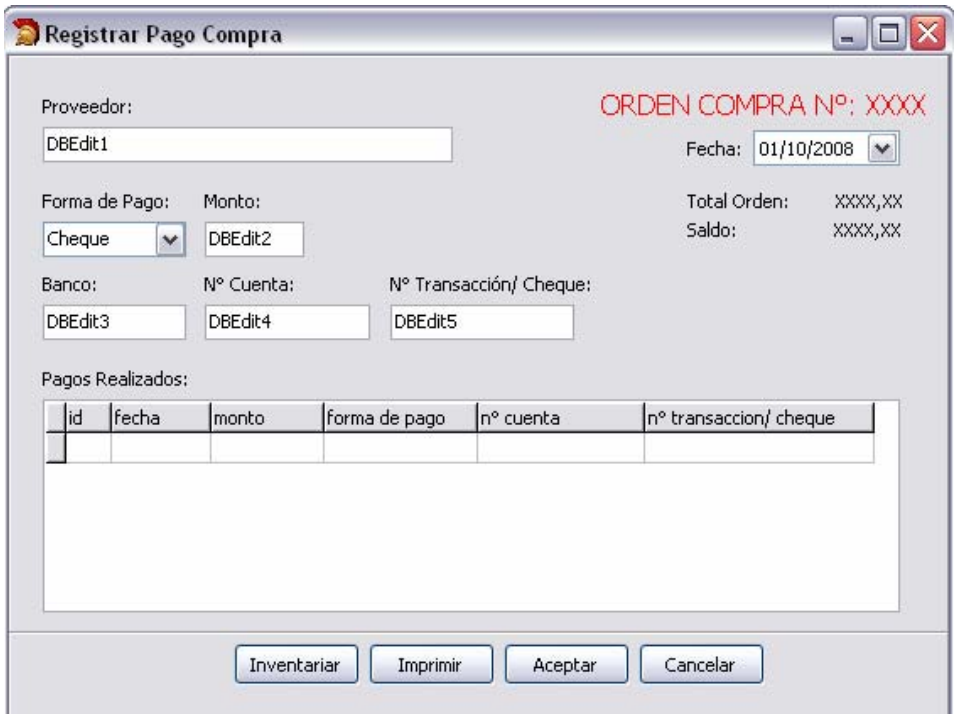


Figura 5.9. Vista de la ventana de iuProcesarCompra – Fuente: Propia

La clase de diseño Gestor Procesar Compra o también está implementada en el código del componente **iuPresupuesto.cpp** y se encarga entre otras cosas de cargar los datos de la orden y realizar la carga del inventario respectiva.

5.6. Pruebas

El modelo de pruebas fue elaborado para tener un patrón a seguir en la prueba del sistema, se identificaron los casos de pruebas necesarios para lograr ese objetivo, preparando un procedimiento de prueba en cada caso.

Para la prueba del sistema se estableció la siguiente estrategia: las pruebas se realizaron en su totalidad en forma manual, se probó el funcionamiento completo del

sistema dividiendo los casos de prueba por funcionalidades en grupos de acuerdo a los casos de uso implementados.

5.6.1. Primer Conjunto de Pruebas

Estas pruebas se realizaron para verificar la funcionalidad del caso de uso Vender Productos y Servicios.

- **Elaboración de la Orden de Servicio**

Este caso de prueba verifica la funcionalidad de la implementación del caso de uso Elaborar Orden de Servicio.

Entrada:

Tabla 5.1. Datos de entrada para el caso de prueba Elaborar Orden de Servicio

Fecha:	19/10/2009
Cliente:	Trinelca
Descripción Problema:	El equipo se apaga
Nro. EquipoCliente:	1

Resultado:

Orden de servicio creada correctamente con los datos suministrados.

Condiciones:

Ninguno.

Procedimiento de Prueba:

1. El usuario activa el caso de uso Elaborar Orden de Servicio y accede a la interfaz de usuario.
2. Selecciona la fecha.
3. Selecciona el cliente. También está la opción de agregar un nuevo cliente.
4. Introduce la descripción del problema.

5. Se pulsa el botón de agregar los equipos recibidos a la orden. También puede crear nuevos equipos para el cliente.
6. El sistema guarda de forma temporal los datos de la orden para mantener la integridad de los datos y luego añade los equipos a la orden.
7. Se pulsa el botón guardar y el sistema almacena los datos.
8. El sistema genera un mensaje de que la operación se ha hecho con éxito.

En la Figura 5.10 se pueden apreciar las opciones escogidas, en este caso de prueba.

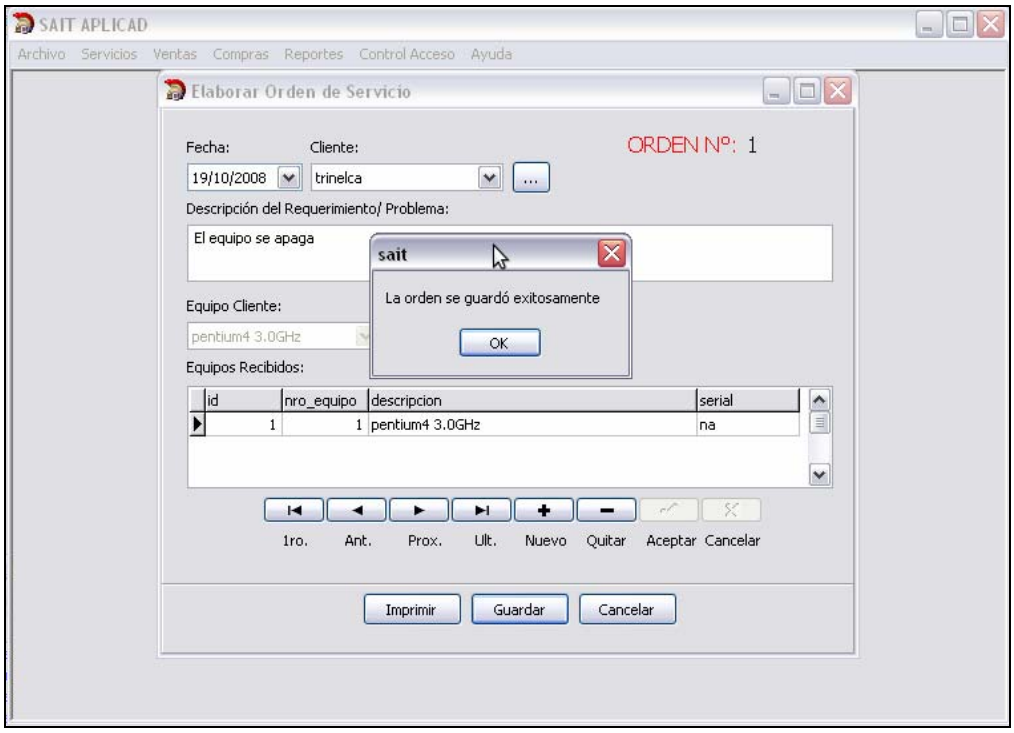


Figura 5.10. Elaboración de Orden de Servicio – Fuente: Propia

Facturación de Productos y Servicios

Siguiendo el ejemplo anterior en este caso de prueba se verifica la funcionalidad de la implementación del caso de uso Facturar Productos y Servicios.

Entrada:

Tabla 5.2. Datos de entrada para el caso de prueba Facturar Productos y Servicios

Fecha:	22/10/2009
Cliente:	Trinelca
Cod. Producto/Servicio:	3
Cantidad	1
Cod. Producto/Servicio:	pw-f500
Cantidad	1

Resultado:

Factura creada correctamente con los datos suministrados.

Condiciones:

Ninguna.

Procedimiento de Prueba:

1. El usuario activa el caso de uso Facturar Productos y Servicios y accede a la interfaz de usuario.
2. Selecciona la fecha de emisión de la factura.
3. Selecciona el cliente. También está la opción de agregar un nuevo cliente.
4. Se pulsa el botón Activar para guardar de forma temporal los datos de la factura y cargar los detalles de la factura a través del Gestor Detalles Factura.
5. Se pulsa el botón de agregar, para crear un nuevo detalle.
6. Se selecciona el producto o servicio respectivo y se ingresa la cantidad.
7. Se pulsa el botón de guardar el sistema almacena los datos.
8. El sistema genera un mensaje de que la operación se ha hecho con éxito.

En las Figuras 5.11 y 5.12 se pueden apreciar las opciones escogidas, en este caso de prueba.

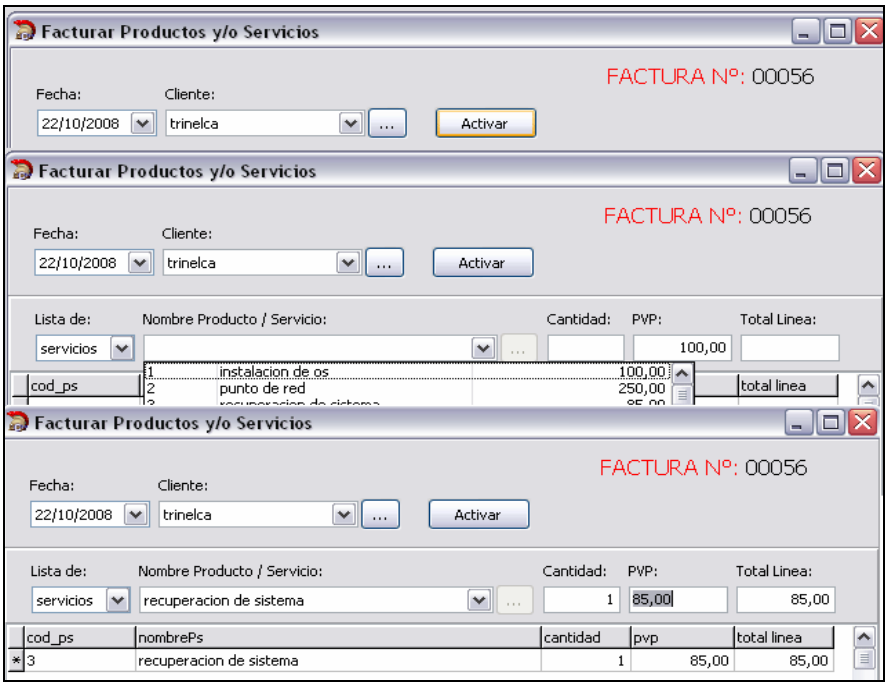


Figura 5.11. Carga de datos de la factura. – Fuente: Propia

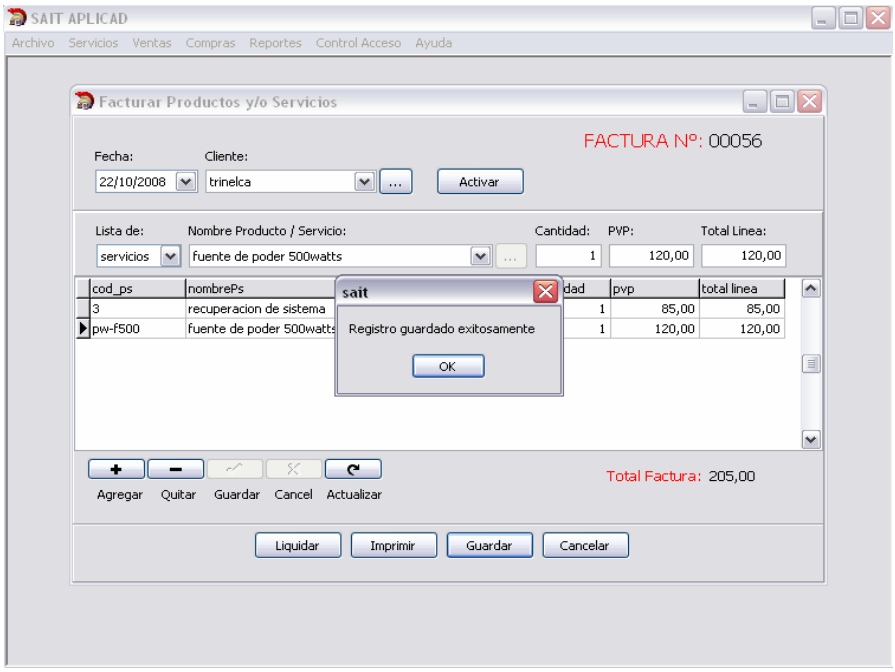


Figura 5.12. Facturación de Productos y Servicios. – Fuente: Propia

- **Introducir Datos del Pago de la Factura**

Este caso de prueba verifica la funcionalidad de la implementación del caso de uso Registrar Pago.

Entrada:

Tabla 5.3. Datos de entrada para el caso de prueba Registrar Pago

Fecha:	22/10/2009
Forma de Pago:	Transferencia
Monto:	205,00
Banco:	Mercantil
Nro. Comprobante:	003455550321

Resultado:

Se registra el pago exitosamente.

Condiciones:

Debe haberse cargado previamente la factura a la que se va a asociar el pago.

Procedimiento de Prueba:

1. El usuario registra una factura y activa el caso de uso Registrar Pago a través del botón liquidar y accede a la interfaz de usuario.
2. Selecciona la fecha.
3. Selecciona la forma de pago
4. Introduce el monto a pagar y según la forma de pago introduce los datos de banco y número de comprobante.
5. Se pulsa el botón guardar y el sistema almacena los datos.
6. El sistema genera un mensaje de que la operación se ha hecho con éxito.

En la Figura 5.13 se pueden apreciar las opciones escogidas, en este caso de prueba.

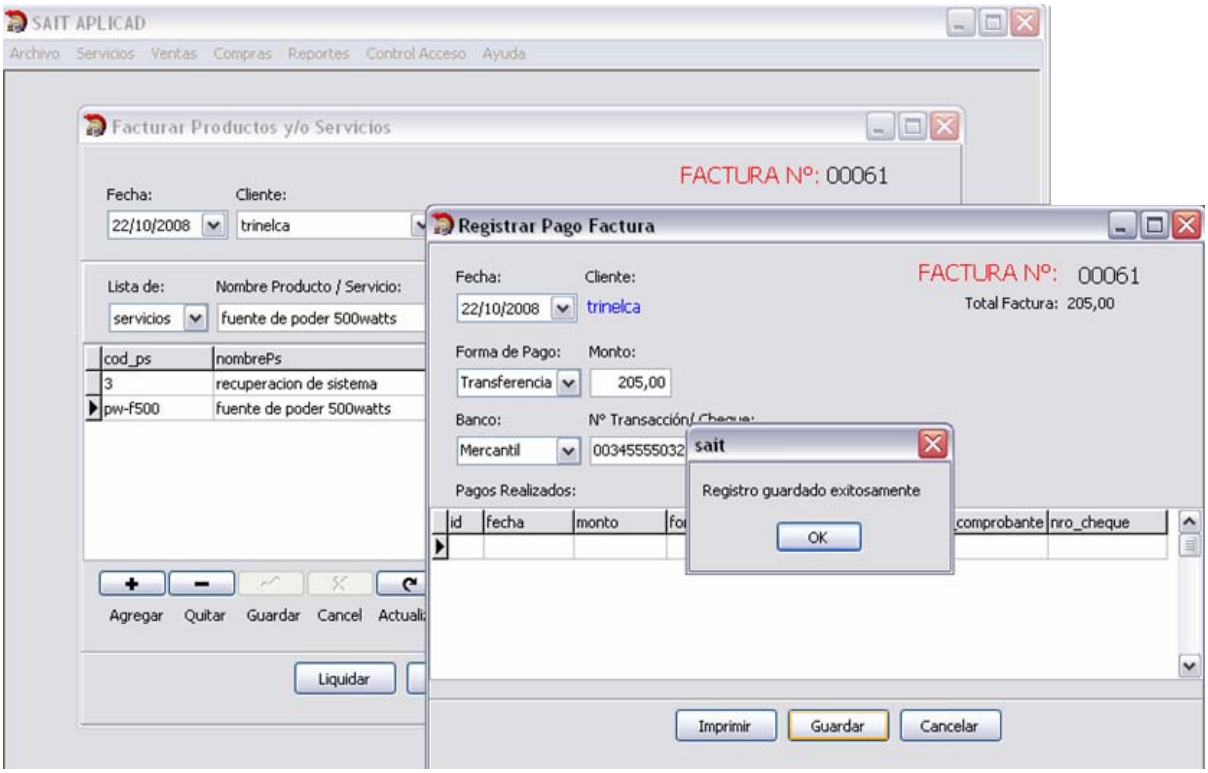


Figura 5.13. Registrar Pago – Fuente: Propia

5.6.2. Segundo Conjunto de Pruebas

Estas pruebas se realizaron para verificar la funcionalidad del caso de uso Comprar Productos.

- **Consultar Inventario**

Este caso de prueba verifica la funcionalidad de la implementación del caso de uso Consultar Disponibilidad Inventario.

Entrada:

Ninguna.

Resultado:

Se muestra el resultado de la consulta realizada.

Condiciones:

Deben existir registros de compras, productos y seriales de productos para que se puedan mostrar los resultados de la consulta.

Procedimiento de Prueba:

1. El usuario activa el caso de uso Consultar Disponibilidad Inventario y accede a la interfaz de usuario.
2. Selecciona el orden de los datos.
3. Selecciona una categoría específica
4. Selecciona una marca de producto específica.
5. El sistema muestra los resultados instantáneamente, según los parámetros escogidos.
6. Si un producto tiene baja disponibilidad puede el usuario puede ordenar un pedido.

En las Figura 5.14 y 5.15 se pueden apreciar las opciones escogidas, en este caso de prueba.

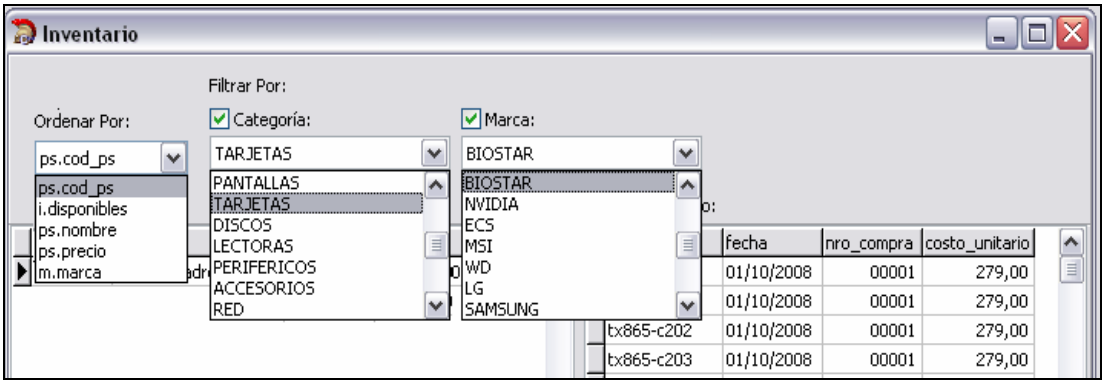


Figura 5.14. Opciones de Consultar Inventario – Fuente: Propia

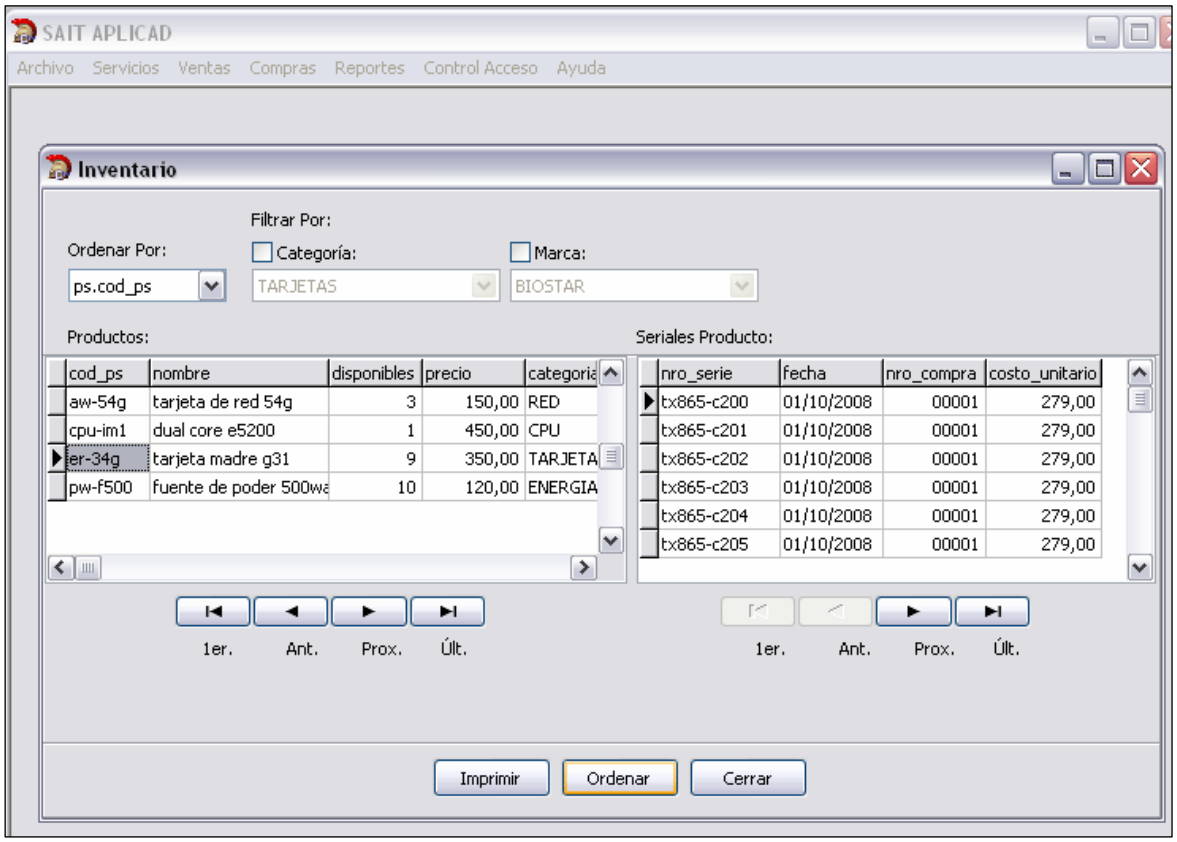


Figura 5.15. Consulta de Inventario – Fuente: Propia

Elaborar Orden de Compra

Este caso de prueba verifica la funcionalidad de la implementación del caso de uso Elaborar Orden de Compra.

Entrada:

Tabla 5.4. Datos de entrada para el caso de prueba Facturar Productos y Servicios

Fecha:	22/10/2009
Proveedor:	Greentech c.a.
Cod. Producto/Servicio:	cpu-im1
Cantidad	1
Precio:	385,00

Resultado:

Orden de Compra creada correctamente con los datos suministrados.

Condiciones:

Ninguna.

Procedimiento de Prueba:

1. El usuario activa el caso de uso Elaborar Orden de Compra y accede a la interfaz de usuario.
2. Selecciona la fecha de la orden.
3. Selecciona el proveedor. También está la opción de agregar un nuevo proveedor.
4. Se pulsa el botón Activar para guardar temporalmente los datos de la orden y cargar los detalles de la misma a través del Gestor Detalles Orden de Compra.
5. Se pulsa el botón de agregar, para crear un nuevo detalle.
6. Se selecciona el producto y se ingresa la cantidad y el precio de compra.
7. Se pulsa el botón de guardar el sistema almacena los datos.
8. El sistema genera un mensaje de que la operación se ha hecho con éxito.

En la Figura 5.16 se pueden apreciar las opciones escogidas, en este caso de prueba.

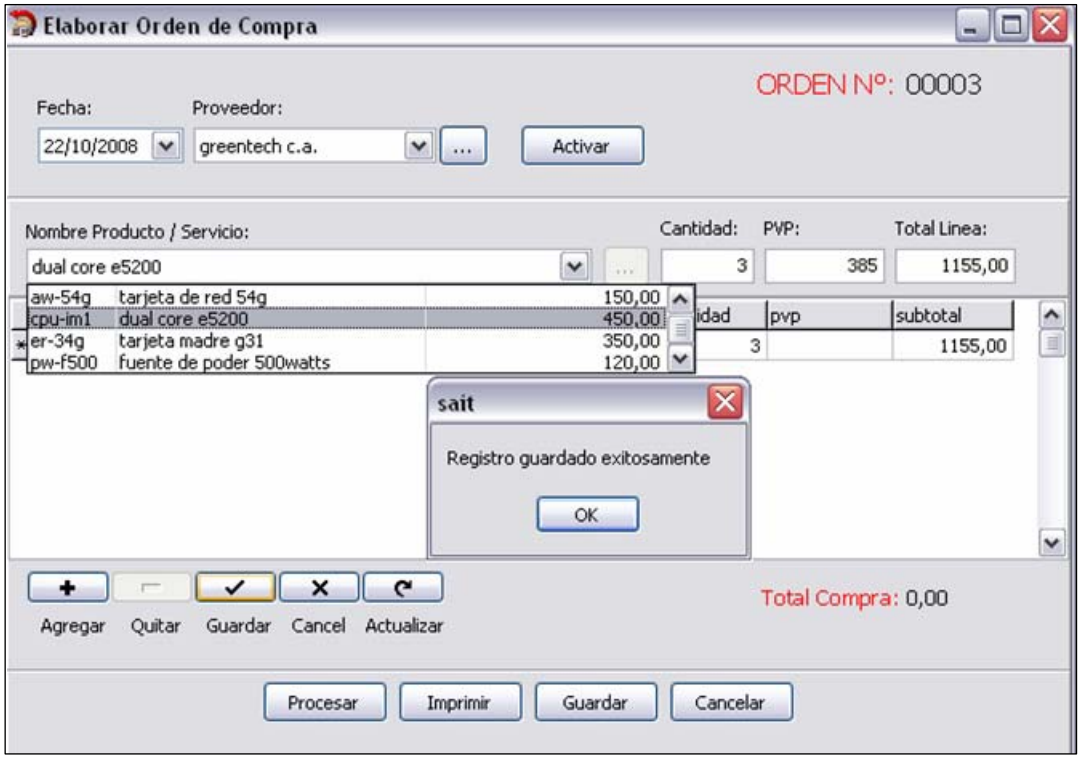


Figura 5.16. Elaboración de Orden de Compra – Fuente: Propia

5.7. Resumen de la fase de Construcción

Durante la fase de construcción se complementó el diseño de la arquitectura base de APLICAD y se codificaron e integraron todos los componentes del sistema.

Los casos de prueba fueron de utilidad para comprobar el funcionamiento de cada parte del sistema y se agruparon por casos de uso, además permitieron identificar y corregir errores asociados a los datos de entrada.

Durante la construcción de APLICAD se amortiguaron los riesgos correspondientes a esta fase, lográndose el cometido de tener lista la primera versión operativa del sistema.

CONCLUSIONES

1. Dentro de una cooperativa se llevan a cabo muchas labores administrativas al igual que en cualquier otro tipo de empresa y el poder contar con una herramienta que controle la información manejada por cada proceso administrativo permite mayor organización y una mejor toma de decisiones, y analizar el comportamiento de la empresa a través del tiempo.
2. A partir de las entrevistas al personal de SAIT, RL para la recolección de requerimientos hasta la implementación y prueba del sistema, el uso de la metodología del Proceso Unificado de Desarrollo de Software permitió una mejor comprensión de los procesos administrativos de SAIT, RL y poderlos plasmar en la primera versión del programa APLICAD.
3. La Evaluación de las fases de Inicio y Elaboración fue necesaria para tomar los modelos descriptivos de la arquitectura de APLICAD como punto de partida para la continuación del desarrollo.
4. Los diagramas de clases, de secuencias, de componentes y de capas, realizados durante las fases de Elaboración y Construcción, complementaron el diseño de la arquitectura base de APLICAD antes de pasar al flujo de trabajo Implementación.
5. Para la construcción de APLICAD se escogió el lenguaje de programación C++, el cual soporta programación orientada a objeto, y se utilizó el entorno de desarrollo integrado (IDE) C++Builder 2007 que permitió, a través de sus herramientas, la rápida construcción del software.
6. La base de datos de APLICAD fue construida exitosamente en MySQL que es un manejador de base de datos relacional, software libre y cumple con todos los estándares del ANSI SQL.

7. El flujo de trabajo Implementación fue el más importante en la fase de Construcción debido a que en éste se codificaron e integraron todos los componentes del sistema.
8. Los casos de prueba fueron de utilidad para comprobar el funcionamiento de cada parte del sistema y se agruparon por casos de uso, además permitieron identificar y corregir errores asociados a los datos de entrada.
9. La implementación de APLICAD en los procesos administrativos de SAIT, RL permitirá agilizar los mismos y obtener información y reportes significativos para una mejor toma de decisiones.

RECOMENDACIONES

1. La utilización del sistema APLICAD para realizar los procesos administrativos en la Cooperativa SAIT, R.L.
2. Realizar planes de entrenamiento a los usuarios que faciliten la manipulación de la aplicación.
3. La ampliación de funcionalidades del sistema para que incluya otras funciones de interés para la empresa, de tal manera que sea un sistema integral.
4. La elaboración de otros proyectos en la Universidad de Oriente, especialmente dedicados al desarrollo de software para el manejo de otros aspectos de las empresas cooperativas.
5. La utilización de bases de datos de software libre para el desarrollo de todo tipo de aplicaciones ya que esto disminuye los costos de mantenimiento del sistema sin bajar la calidad del producto ya que esto disminuye los costos sin bajar la calidad del producto.

BIBLIOGRAFÍA

[1] Cooperativa SAIT, RL, "**¿Qué es SAIT, RL?**", (2009), http://sait.com.ve.

[2] Superintendencia Nacional de Cooperativas, "**Cooperativas**", (2009), http://www.sunacoop.gob.ve/contenido.php?id=205.

[3] TANENBAUM, A., (2003), "**Redes de Computadoras**". Editorial Prentice Hall, 4ta. Edición, México.

[4] MYSQL AB, (2007), "**MySQL 5.0 Reference Manual**". Versión en Español.

[5] MITCHELL, J., (2003), "**Conceptos en Lenguajes de Programación**". Prensa Universidad de Cambridge, USA.

[6] FUAD, N., (2008), "**Desarrollo de un software para el control de inventario del material importado de una ensambladora de vehículos**". Universidad de Oriente, Puerto la Cruz, Venezuela.

[7] MARMOUD, B. y CARABALLO, A., (2007), "**Desarrollo un software para la automatización del proceso de compras del 'Departamento de Compras No Productivas' de una empresa automotriz".** Universidad de Oriente, Puerto la Cruz, Venezuela.

[8] ZACARÍAS, E. y ARCILA, A., (2007), **"Desarrollo un software que sirva de soporte para la automatización de las labores administrativas de la Unidad Educativa "Elisa Elvira Zuloaga" y Guardería-Preescolar "Trencito Mi Rosita Mística"".** Universidad de Oriente, Puerto la Cruz, Venezuela.

[9] ARLOW, J. y NEUSTADT, I., (2006), "**Programación UML 2**". Editorial Anaya Multimedia, 1era. Edición. España.

[10] CHARTE, F., (2006), "**C++ BUILDER 2006**". Editorial Anaya Multimedia, 1era. Edición. España.

[11] CHARTE, F., (2009), "**SQL**", Editorial Anaya Multimedia, 1era. Edición, España.

[12] AVILEZ, J., (2005), **"Recolección de datos"** http://www.monografias.com/trabajos/recoldat/recoldat.shtml.

[13] MARTRA, P., (2005), "**UML"**. www.programacion.com/docs/uml/1.html, (2005).

[14] Universidad Nacional de Colombia, (2004), "**Bases de Datos**". http://www.virtual.unal.edu.co/cursos/sedes/manizales/4060029/index.html.

[15] SALAZAR, L., (2003), "**Desarrollo de un software para automatizar las actividades de control interno de una empresa de transporte de alimentos"**. Universidad de Oriente, Puerto la Cruz, Venezuela.

[16] PUGLIESE, I., (2001), **"Desarrollo del sistema de compras en ambiente Cliente/Servidor para la Universidad de Oriente núcleo de Anzoátegui"**. Universidad de Oriente, Puerto la Cruz, Venezuela.

[17] RUMBAUGH J., JACOBSON I. y BOOCH G., (2000), **"El Proceso Unificado de Desarrollo de Software".** Primera Edición. Editorial Addison-Wesley. Madrid, España.

METADATOS PARA TRABAJOS DE GRADO, TESIS Y ASCENSO:

TÍTULO	**"DESARROLLO DE UN SOFTWARE PARA EL CONTROL DE LOS PROCESOS ADMINISTRATIVOS DE LA EMPRESA COOPERATIVA SAIT, RL"**
SUBTÍTULO	

AUTOR (ES):

APELLIDOS Y NOMBRES	CÓDIGO CVLAC / E_MAIL
Rodríguez V., Ronal R.	**CVLAC: 14.477.562** **E_MAIL: sagiro60@gmail.com**
	CVLAC: **E_MAIL:**
	CVLAC: **E_MAIL:**
	CVLAC: **E_MAIL:**

PALÁBRAS O FRASES CLAVES:

Desarrollo de Software

Cooperativa

Procesos administrativos

UML

C++Builder

MySQL

METADATOS PARA TRABAJOS DE GRADO, TESIS Y ASCENSO:

ÁREA	SUBÁREA
Ingeniería y Ciencias Aplicadas	Ingeniería en Computación

RESUMEN (ABSTRACT):

La Cooperativa SAIT, RL es una empresa orientada a proveer bienes y servicios en el área de la computación. La empresa presenta problemas en el manejo de sus procesos administrativos y de control interno y no cuenta con las herramientas ni el tiempo necesario para ello, trayendo como consecuencia acumulación de trabajo, malas decisiones, uso ineficiente de recursos financieros e incumplimiento con la Superintendencia Nacional de Cooperativas (SUNACOOP). Por las razones expuestas se propuso el presente trabajo de grado, el cual consiste en desarrollar un software para el control de los procesos administrativos de SAIT, RL entre los que destacan: compras, ventas, control de inventario, gestión de clientes y proveedores, historial de servicio. Para el desarrollo se utilizó la metodología del Proceso Unificado de Desarrollo de Software y el Lenguaje de Modelado UML, aplicando las fases de Inicio, Elaboración y Construcción a través de una o más iteraciones de los flujos de trabajo: captura de requisitos, análisis, diseño, implementación y prueba. Se usó el IDE C++Builder11, basado en el lenguaje C++, como herramienta de programación. La base de datos relacional del sistema fue implementada en MySQL Server 5.0, creando así una arquitectura tipo cliente/servidor para el manejo centralizado de la información.

METADATOS PARA TRABAJOS DE GRADO, TESIS Y ASCENSO:

CONTRIBUIDORES:

APELLIDOS Y NOMBRES	ROL / CÓDIGO CVLAC / E_MAIL				
Cortinez, Claudio	ROL	CA	AS(X)	TU	JU
	CVLAC:	12.155.334			
	E_MAIL	cl_cortinez@cantv.net			
	E_MAIL				
Rodríguez, Rhonald	ROL	CA	AS	TU	JU(X)
	CVLAC:	14.077.186			
	E_MAIL	rhoen2003@hotmail.com			
	E_MAIL				
Veracierta, Gabriela	ROL	CA	AS	TU	JU(X)
	CVLAC:	14.616.683			
	E_MAIL	gveracierta@hotmail.com			
	E_MAIL				
	ROL	CA	AS	TU	JU
	CVLAC:				
	E_MAIL				
	E_MAIL				

FECHA DE DISCUSIÓN Y APROBACIÓN:

2009	10	22
AÑO	MES	DÍA

LENGUAJE. SPA

METADATOS PARA TRABAJOS DE GRADO, TESIS Y ASCENSO:

ARCHIVO (S):

NOMBRE DE ARCHIVO	TIPO MIME
TESIS DESARROLLO SOFTWARE CONTROL ADMINISTRATIVO COOPERATIVA SAIT.DOC	application/msword

CARACTERES EN LOS NOMBRES DE LOS ARCHIVOS: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z. a b c d e f g h i j k l m n o p q r s t u v w x y z. 0 1 2 3 4 5 6 7 8 9.

ALCANCE

ESPACIAL: _____ **(OPCIONAL)**

TEMPORAL: _____ **(OPCIONAL)**

TÍTULO O GRADO ASOCIADO CON EL TRABAJO:
Ingeniero en Computación
_____

NIVEL ASOCIADO CON EL TRABAJO:
Pre-Grado
_____

ÁREA DE ESTUDIO:
Departamento de Computación y Sistemas
_____

INSTITUCIÓN:
Universidad de Oriente
_____

METADATOS PARA TRABAJOS DE GRADO, TESIS Y ASCENSO:

DERECHOS

"Los trabajos de Grado son de exclusiva propiedad de la Universidad y sólo podrán ser utilizados a otros fines con el consentimiento del Consejo de Núcleo respectivo, quién lo participará al Consejo Universitario."

Rodríguez V., Ronal R.
AUTOR

Rodríguez, Rhonald	**Cortinez, Claudio**	**Veracierta, Gabriela**
JURADO	**TUTOR ACADÉMICO**	**JURADO**

POR LA SUBCOMISION
DE TESIS