PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR

FACULTAD DE INGENIERIA

ESCUELA DE SISTEMAS

DISEÑO, DESARROLLO E IMPLANTACIÓN DEL SISTEMA ADMINISTRATIVO Y FINANCIERO (SAF) PARA MUEBLES PRIETO

AUTORES:

PRIETO REYES LUIS MIGUEL Y PRIETO REYES PATRICIO FERNANDO

QUITO - 2010

1 Marco Teórico

1.1 Principios y conceptos de la ingeniería del software

1.1.1 Definición de ingeniería del software

La Ingeniería del software es una disciplina que se encarga de todos los aspectos necesarios para la elaboración del producto software, utilizando métodos, teorías y herramientas basadas en un proceso. [1]

1.1.2 Modelos de Proceso del software

Los modelos de proceso o paradigmas de la Ingeniería del Software son una representación abstracta de los procesos de software desde cierta perspectiva y que se usan como marco de trabajo.

Modelo en Cascada

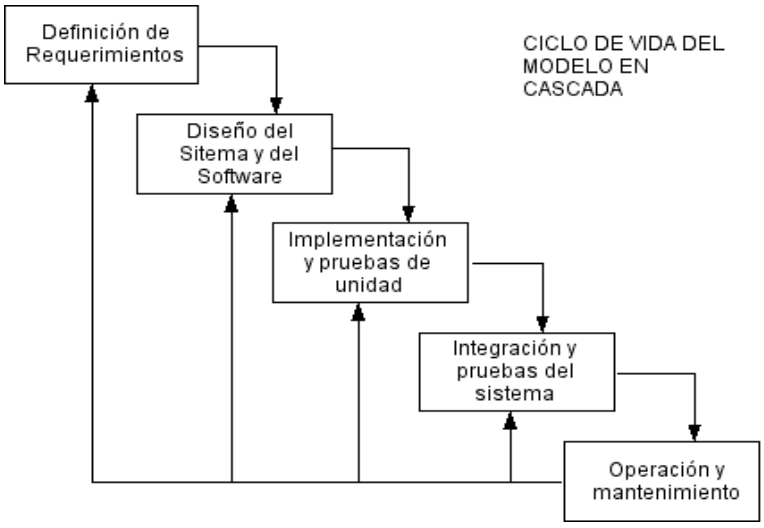


Figura 1.1 Modelo cascada, Fuente (Sommerville, 2004).

1.1.2.1 Desarrollo Evolutivo

Su idea fundamental es construir una primera implementación y luego continuarla refinando hasta cumplir los requerimientos del cliente. Las actividades de especificación, desarrollo y validación están entrelazadas con retroalimentación inmediata entre ellas.

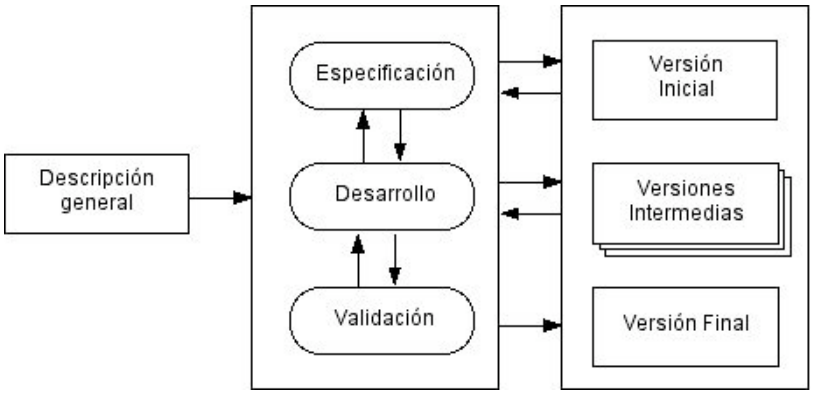


Figura 1.2 Desarrollo evolutivo, Fuente (Sommerville, 2004).

1.1.2.2 Ingeniería del Software Basada en Componentes

Ingeniería del Software Basada en Componentes o Component-Based Software Engineering (CBSE). Este paradigma se fundamenta en la base re-usable de componentes de software y un marco de trabajo integrador. A veces estos componentes son sistemas propiamente dichos o Commercial Off-The-Shelf (COTS)

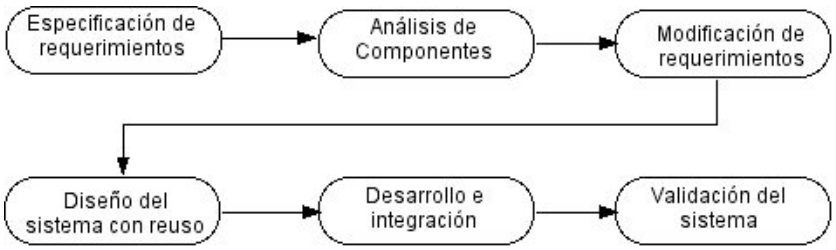


Figura 1.3 Ingeniería del software basada en componentes, Fuente (Sommerville, 2004).

1.1.3 Gestión de proyectos

Es una parte esencial de la Ingeniería del software en donde el Gestor de Proyectos tiene el trabajo de planear y programar el desarrollo del proyecto. Las principales actividades de gestión son: Planeación del proyecto, programación del proyecto y gestión del riesgo.

1.1.3.1 Planeación del Proyecto

La planeación del progreso del proyecto puede tener la siguiente estructura:

```
Establecer los límites del proyecto.
Hacer una valoración inicial de los parámetros del proyecto.
Definir hitos y entregables del proyecto.
mientras el proyecto no ha sido completado o cancelado haga
        Preparar calendarización del proyecto
        Iniciar actividades de acuerdo al calendario
        Esperar ( por un momento 2 o 3 semanas)
        Revisar el progreso del proyecto
        Revisar los estimados de los parámetros del proyecto
        Actualizar la calendarización del proyecto
        Renegociar los límites del proyecto y entregables
        si ( aparecen problemas ) entonces
                iniciar revisión técnica
        fin del si
fin del mientras
```

Figura 1.4 Algoritmo de planeación de proyectos, Fuente (Sommerville, 2004).

Identificamos los límites o restricciones como: La fecha límite de entrega, el personal disponible y el costo aproximado. Después los parámetros como la estructura, el tamaño y distribución de funciones. Luego definimos los hitos y entregables. El proyecto entra en un “mientras” y no se detiene hasta que se complete o cancele lo que sugiere una revisión regular.[2]

1.1.3.2 Gestión del Riesgo

El objetivo principal de gestionar el riesgo es anticipar los problemas que puedan afectar el programa del proyecto o la calidad del software a desarrollarse. [3]

1.1.5 Verificación y Validación

La verificación y validación o (V&V) son procesos de chequeo y análisis. La verificación prueba si el sistema cumple con la especificación de requerimientos mientras que la validación demuestra si el sistema cumple con las expectativas del cliente. V&V pueden ser muy costosas, cuando el software a construir es de tiempo real y crítico, fácilmente se puede gastar la mitad del presupuesto solo en ellas. La planeación debe equilibrarlas para obtener lo mejor de las inspecciones y pruebas y controlar la inversión en la validación y verificación. Las técnicas principales de la V&V son las inspecciones del software y las pruebas del software. [4]

1.1.4 Gestión de Calidad del Software

Es medir de la calidad del software a base de estándares y procedimientos. Sin olvidar que existen aspectos intangibles como la elegancia, legibilidad, rendimiento, capacidad de mantenimiento, seguridad, eficiencia y otros factores más que claramente afectan a la calidad.

1.2 El Proceso Unificado de Desarrollo de Software

El Proceso Unificado de Desarrollo de Software o Unified Software Development Process (USDP) es una propuesta para enfrentar las nuevas demandas del mercado. La necesidad de sistemas que integren varias soluciones al negocio y que al mismo tiempo cumplan las demandas mas exigentes del cliente están presentes, por lo tanto el desarrollo de software se ha vuelto mas complejo y si a esto le sumamos la necesidad de construirlo en corto tiempo, estamos obligados a mejorar nuestro métodos con el fin de atender estas exigencias. El USDP usa tres pilares que atacan a los problemas de desarrollo actuales y que describimos a continuación.

1.2.1 Características principales del USDP

- El USDP esta dirigido por casos de uso: Los casos de uso detallan un conjunto de acciones que el sistema realiza para entregarnos un resultado, ellos se encuentran enfocados en responder preguntas como ¿A quién ayudan? ¿Qué necesidades del negocio satisfacen? Y ¿Cuánto valor añaden al negocio? Lo que nos permite encontrar los requisitos verdaderos y representarlos para que los usuarios, clientes y desarrolladores los comprendan. En el USDP Los casos de uso inician el proceso con el modelo de casos de uso, luego son conductores en las demás fases completando los diferentes modelos y por último enlazan los flujos fundamentales de trabajo lo que establece la dependencia de traza entre modelos.

- El USDP es centrado en la arquitectura: La arquitectura es importante para comprender el sistema puesto que divide su complejidad, permite organizar el desarrollo del sistema

fortaleciendo la comunicación entre los grupos de trabajo, fomenta la reutilización mediante el uso de estándares y evoluciona el sistema porque es tolerante a los cambios. Una vez que hemos establecido los casos de uso claves, la arquitectura toma forma y permite el desarrollo inicial y avance del proceso, la arquitectura se completa a través de las diferentes fases siguientes. La arquitectura y los casos de uso evolucionan en forma paralela.

- El USDP es iterativo e incremental: porque se puede dividir las fases en pequeñas partes o miniproyectos, los cuales resultan en un incremento al sistema cada vez. Al ser un miniproyecto indica que esta controlado y planificado, es decir se sabe que hacer (objetivos a cumplir) y cuando hacerlo (hay límite de tiempo). Los miniproyectos o iteraciones existen en cada una de las fases del proceso y cada iteración ejecuta los flujos fundamentales completándolos.

1.2.2 Los flujos de trabajo fundamentales

Existen cinco flujos fundamentales con los que trabajo el USDP y son: Requisitos, análisis, diseño, implementación y prueba. [5]

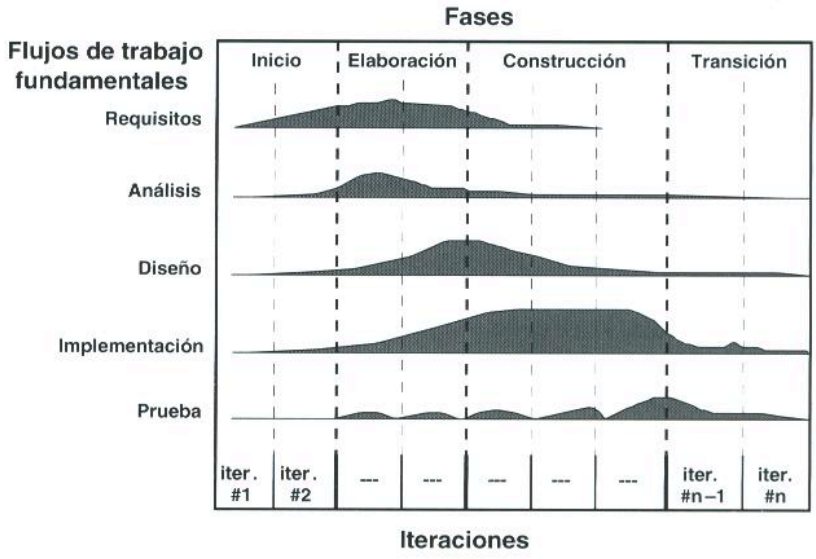


Figura 1.5 Flujos de trabajo, Fuente (Jacobson, 2000).

1.2.2.1 Requisitos

La recopilación de requisitos es el proceso de averiguar normalmente lo que se debe construir en situaciones difíciles para el analista, debido a que el usuario generalmente no sabe cuales son los requisitos para el sistema, ni como especificarlos.

El analista es el medio para que los usuarios nos permitan obtener una la lista de requisitos correcta, completa y consistente.[6]

1.2.2.2 Análisis

El análisis se enfoca en refinar y estructurar los requisitos del flujo de trabajo anterior, puesto que tiene como clave la formalización. Entre sus características principales están:

- En el análisis temas pendientes de la captura de requisitos como concurrencia, interferencia y conflictos entre casos de uso son resueltos.
- En el análisis utilizamos el lenguaje de los desarrolladores y se puede expresar el funcionamiento interno detalladamente.[7]

1.2.2.3 Diseño

El diseño tiene las siguientes características:

- Es un modelo físico que se crea del modelo conceptual del análisis, así deja de ser genérico y se hace específico.[8]
- Detalla todos los requisitos funcionales y no funcionales.
- Esta enfocado al hardware, sistemas operativos y lenguaje de programación.
- Sus estereotipos dependen del lenguaje de programación.
- En la figura 1.6 se observa el ciclo del diseño.

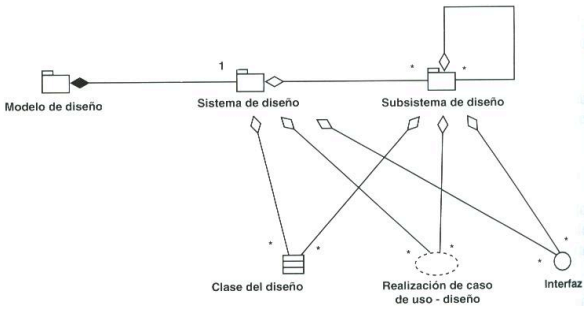


Figura 1.6 Modelo de Diseño, Fuente (Jacobson, 2000).

1.2.2.4 Implementación

En el ciclo de vida del sistema la implementación tiene su foco en la fase de construcción pero se presenta en la elaboración para crear la línea base de la arquitectura y en la transición para soportar defectos tardíos.[9] Tiene el objetivo general de implementar los componentes y específicamente los siguientes objetivos:

- Implementar el sistema incrementalmente.
- Distribuir el sistema a los nodos con clases activas.
- Implementar clases y subsistemas.
- Realizar pruebas individuales e integrales de los componentes.

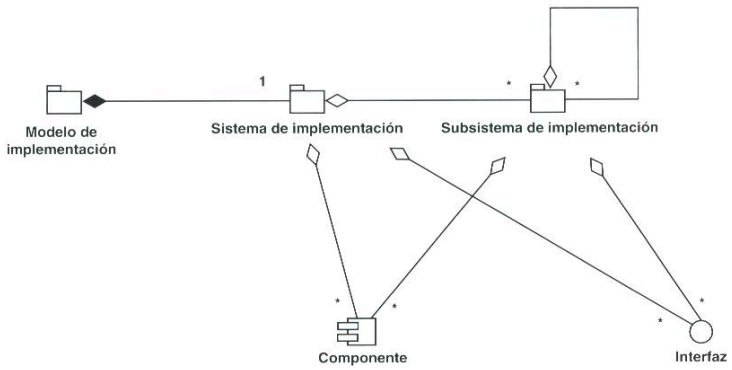


Figura 1.7 Modelo de Implementación, Fuente (Jacobson, 2000).

1.2.2.5 Pruebas

Las pruebas se realizan primordialmente en la elaboración para probar la línea base de la arquitectura y en la construcción cuando el sistema esta implementado.[10] El objetivo general es probar cada construcción y específicamente realizamos lo siguiente:

- Planificar las pruebas de integración y sistema.
- Diseñar e implementar casos de uso de prueba.
- Realizar las pruebas, guardar y analizar los resultados de las pruebas.

1.2.3 Fases de Desarrollo

Después de conocer sobre los cinco flujos de trabajo fundamentales el siguiente paso es la planificación que incluye la administración de riesgos, estimación de recursos y como llevar

a cabo y ejecutar cada fase. Se busca un equilibrio entre las actividades de cada iteración y su ejecución eficiente.

En las primeras iteraciones nos concentramos en actividades como identificar y mitigar riesgos críticos, casos de uso fundamentales, cuestiones arquitectónicas, elección del entorno de desarrollo, cuestiones de investigación. Mientras que en las últimas iteraciones nos concentramos en el desarrollo, problemas de evaluación, rendimiento, y despliegue del sistema. Las fases son la primera división del trabajo pero pueden seccionarse en más de una iteración. [11]

Fases	Modelo del negocio completado	Casos de uso identificados	Casos de uso detallados	Casos de uso analizados	Casos de uso diseñados, implementados y probados
Inicio	50% - 70%	50%	10%	5%	Un pequeño porcentaje
Elaboración	Casi el 100%	80% o más	40% - 80%	20% - 40%	Menos del 10%
Construcción	100%	100%	100%	100% si se mantienen	100%
Transición	-	-	-	-	-

Tabla 1.1 Trabajo de los casos de uso en las fases de desarrollo, Fuente (Jacobson, 2000).

1.3 Lenguaje Unificado de Modelado

1.3.1 Modelado e Importancia

Para construir software de grandes magnitudes o requerimientos se debe minimizar el software, identificando los requerimientos útiles y desechando los que no son válidos para el sistema, con esto determinamos los elementos más importantes lo cuales nos ayudarán a construir nuestro software, así, podremos partir desde un modelo simple pero óptimo a hasta llegar a construir un modelo grande y manejable.

Un modelo provee los planos generales o detallados de un sistema. Los modelos bien hechos incluyen la información que se necesita y desecha la que no es necesaria. Los modelos son representaciones estándar de la realidad y no pueden ser leídos en forma diferente, es decir, que los modelos no son interpretativos.

Los objetivos fundamentales que se consiguen al modelar nos ayudan a manejar y entender sistemas muy grandes, disminuyendo su dificultad y complejidad, estos son:

1. Visualizar el sistema
2. Especificar la estructura y comportamiento del sistema
3. Obtener las plantillas para la construcción del sistema
4. Documentarlas ideas que se han planteado

Una de las claves del éxito del desarrollo de software de calidad es el modelado, el cual es una técnica de ingeniería que nos permite visualizar, comprender la arquitectura y partes de lo que queremos construir.

1.3.2 Lenguaje Unificado de Modelado

UML es un lenguaje estándar que proporciona un vocabulario muy amplio y reglas para la representación física y conceptual de un sistema, al mismo tiempo este vocabulario y reglas permiten crear y leer modelos bien formados.

El Lenguaje de Modelado Unificado o Unified Modeling Language (UML), se define como un lenguaje estándar para crear planos de software, los cuales pueden usarse para visualizar, especificar, construir y documentar los artefactos de un sistema con gran cantidad de software.[12]

2 Viabilidad

2.1 Situación Actual

Muebles Prieto es un negocio familiar que esta enfocado a la comercialización de muebles para el hogar y oficina, satisfaciendo necesidades del cliente, tanto en cuanto a servicios y a calidad de producto. El negocio necesita optimizar sus ganancias organizando la parte administrativa financiera. Por muchos años lo ha venido haciendo satisfactoriamente, pero actualmente el control de inventarios, reportes y contabilidad se ha tornado muy difícil de manejar ya que el negocio está creciendo y se va extendiendo cada vez más, una de la más grande desventaja que tiene el negocio es el ineficaz manejo de inventarios, ya que se lo hace manualmente y esto causa demoras en la revisión del mismo, no se obtienen reportes de muebles que no se venden y errores por parte de los vendedores que son quienes revisan conjuntamente con el administrador el inventario. La contabilidad del negocio no se la lleva por lo que los propietarios no tienen cifras exactas de sus ganancias y gastos.

Actualmente se requiere un sistema informático que pueda manejar estos datos, facilitando así el manejo de inventarios, clientes, proveedores y la parte contables, es decir, que se reducirá las perdidas en el inventario ya que será exacto, se puede llevar un historial de sus clientes y proveedores y otro punto fundamental será la parte contable, que al final de cada periodo podremos revisar y obtener reportes de todos los movimientos y asientos que se han hecho, verificando nuestras ganancias netas al final de cada periodo.

El negocio carece de los elementos necesarios para la implantación del sistema, este es un punto importante porque esto se debe considerar para la inversión inicial que se deberá hacer, debemos considerar también que la infraestructura de los locales no esta diseñada para el uso de redes ni equipos informáticos, por lo que se debe adecuar de tal manera que los mismos puedan funcionar adecuadamente.

2.2 Propuesta

La propuesta hecha a los propietarios de Muebles Prieto es la construcción de un software administrativo financiero que permita el control en su totalidad de cada módulo que el negocio tiene. El sistema permitirá el manejo de clientes, productos, proveedores y

pedidos, de esta manera se podrá llevar un registro y facilitar el manejo de los mismos, además se podrá registrar cada venta por lo tanto al final de cada periodo informar las ventas que se han tenido. El sistema podrá emitir reportes de las existencias de los productos, de los más vendidos y los que no se venden, permitiendo al administrador del negocio clasificar mejor sus pedidos a proveedores, de esta manera no se pedirá mercadería que no se vende. El manejo de los inventarios será más sencillo ya que será exacto, por lo que se podrán evitar pérdidas de productos y robos por parte de los vendedores, esta es una de las grandes ventajas del sistema porque al tener un inventario exacto, las pérdidas serán minimizadas. La parte financiera permitirá registrar las entradas y salidas de dinero es decir el flujo que se va a tener y si en verdad se tendrá ganancias al final de cada período y se podrán hacer asientos contables que registrarán los movimientos en cada local.

El sistema en general podrá optimizar las ganancias del negocio, la organización financiera, y permitirá administrar mejor su mercadería, por lo cual se justifica la construcción del mismo.

2.3 Análisis del Costo Beneficio

2.3.1 Costos del proyecto

- Desarrollo del sistema

  El sistema tendrá un costo por su construcción

- Compra de nuevos equipos

  Lo propietarios tiene que adquirir equipos como computadoras, servidores, switches y cables para la instalación del software.

- Instalación de equipos

  El costo de estos equipos tendrá un costo ya que la infraestructura necesaria no se encuentra lista.

- Instalación del sistema operativo

El sistema operativo base no requiere de licencias, pero la adquisición del mismo e instalación tiene un costo extra.

- Implantación del sistema

Esto implica poner en marcha el sistema, realizando las pruebas necesarias para el correcto funcionamiento.

- Instalación de redes

Se creará una red LAN para cada local y se extenderá la misma si es posible o se creará una red WAN para que los locales puedan estar comunicados entre si.

- Capacitación del personal

Para que cada uno de los involucrados en el negocio tengan conocimiento de la herramienta se dictará cursos de capacitación, de esta manera ningún empleado va a depender de otro para el manejo la herramienta.

- Mantenimiento

Cada cierto período se dará mantenimiento tanto a los equipos como al sistema.

- Suministros de electricidad

Toda tecnología funciona a base de electricidad, por lo cual que cada almacén habrá un incremento de consumo.

2.3.2 Beneficios del proyecto

- Incremento de la productividad.
- Reducción o eliminación de errores.
- Mejorar las ventas competitivas.
- Precisión y rapidez en los cálculos
- Consultas de precios y de stock más eficientemente.
- Seguridad en el almacenamiento de datos.
- Optimización de búsqueda de productos.
- Mayor Organización en los pedidos a proveedores.
- Inventario más organizado.
- Registro de ingresos y egresos exactos.

- Agilidad en el manejo de datos.
- Registro de contabilidad básica automática.
- Mejor atención al cliente.

2.4 Alternativas

- Sistema Operativo

Parámetros	***Windows***	***Linux***	***Mac os X***
Amigabilidad	7/10	6/10	8/10
Costos de licencia	2/10	8/10	1/10
Vulnerabilidad a virus	4/10	8/10	8/10
Popularidad del S.O.	9/10	6/10	7/10
Totales /40	22	**28**	24

Tabla 2.1 Comparación de sistemas operativos, Elaborado por: Luis Prieto y Patricio Prieto.

Por puntaje el sistema operativo que mejor califica para que la herramienta funciones es Linux.

- Hardware

Parámetros	***Clon***	***Marca***
Reducción de costos	Si	No
Garantía	Si	Si

Tabla 2.2 Comparación de equipos, Elaborado por: Luis Prieto y Patricio Prieto.

La reducción de costos es la característica esencial por la que se van a obtener clones y no equipos de marca.

- Herramientas

Las alternativas que se presentan son herramientas que compiten y que se ajustan con las características que los propietarios del negocio han requerido para el negocio.

Parámetros	***SAF MUEBLES PRIETO***	***TMAX***	***MONICA***	***SAFI***
Multiplataforma	Si	No	No	No
Capacitación	Si	Si	Si	Si
Conexión a Bases de Datos	Si	No	No	Si
Trabajo en Red	Si	No	No	Si
Seguridad	Si	Si	Si	SI

Tabla 2.3 Comparación de software, Elaborado por: Luis Prieto y Patricio Prieto.

Los puntos positivos son los que han permitido elegir a la opción ganadora y claramente vemos que es mejor construir una herramienta que utilizar una que ya esta en el mercado que no se ajustará a la infraestructura de hardware ni a los requerimientos de software que el negocio necesita.

- Tecnologías requeridas para el funcionamiento del sistema

Herramientas de servicio	***Descripción***
Equipo Servidor	Servidor HP Proliant ML150
PCs	PC Genérica
S.O. Servidor	Linux Debian
S.O. PCs	Linux Debian
Herramientas de desarrollo	Java Studio Creator 2.1, NetBeans 5.1

Tabla 2.4 Tecnologías requeridas, Elaborado por: Luis Prieto y Patricio Prieto.

El criterio de elección de las herramientas de servicio esta basado en la calificación de las alternativas, sin quitar el valor que cada una de ellas tiene, las herramientas de servicio que hemos elegido podrán satisfacer de la mejor manera las necesidades del sistema, usuarios y de los propietarios, ya que hemos tratado de reducir la inversión inicial en equipos y licencias.

2.5 Riesgos Críticos

Riegos	***Prevención***
El tiempo de desarrollo del sistema puede sobrepasar el tiempo requerido para crearlo.	Realizar un cuadro de planificación de cada una de las etapas de desarrollo.
Los usuarios no se familiaricen con el sistema	El sistema debe ser intuitivo para que se puedan familiarizar con el mismo y se debe dar la respectiva capacitación y soporte técnico.
El sistema tenga errores de diseño	Se debe crear estándares de diseño
EL sistema no se ajuste a los requerimientos del negocio	La toma de requisitos solucionará este problema, pero también se debe familiariza con el negocio y con los involucrados, quienes nos ayudaran a evitar este problema.
El sistema de errores cuando ya este en funcionamiento	Realizar pruebas en cada etapa de desarrollo emulando casos reales que se darían en el negocio.
Los equipos nuevos sufran daños por mal uso	Capacitar a los usuarios y dar un servicio técnico frecuentemente
Perdida de información por daños de hardware	Realizar un back up semanalmente
Inseguridad de datos	Crear usuarios, los cuales tendrán cierta prioridad para el manejo de datos

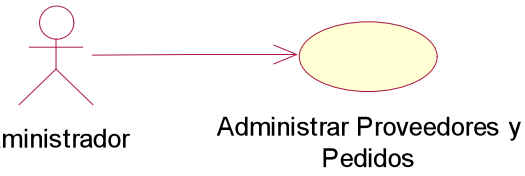
Tabla 2.5 Riesgos, Elaborado por: Luis Prieto y Patricio Prieto.

3 Desarrollo

3.1 Requisitos

3.1.1 Caso de uso administrar proveedores y pedidos CU#01

Administrar Proveedores y Pedidos CU#01



Descripción breve

Este caso de uso permite al administrar pedidos al proveedor y registrar las compras a los mismos. El administrador puede registrar, modificar la información de los productos, pedidos y proveedores El administrador puede registrar una compra a un proveedor.

Actores

1. Actor Primario: Administrador
2. Actor Secundario: Proveedor

Flujo de eventos

Flujo básico

- El administrador necesita satisfacer el stock del almacén.
- El administrador hace una lista de faltantes en el almacén.
- El administrador llama al proveedor y le indica el pedido y la fecha limite de entrega.
- El administrador anota todos los datos en su registro de pedidos.
- El proveedor entrega los productos del pedido.
- El administrador registra la compra de los productos del proveedor.

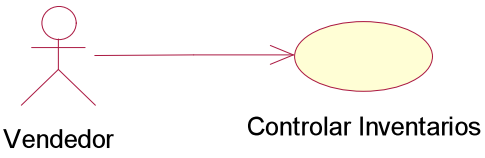
Flujos Alternativos

- **REGISTRAR PRODUCTO**
    - El administrador registra un producto en su lista de productos, si lo desea puede modificarlo.

- **REGISTRAR PROVEEDOR**
  - Surge la necesidad de tener un nuevo proveedor.
  - El administrador llama a un nuevo proveedor.
  - El proveedor proporciona sus datos personales al administrador.
  - El administrador registra los datos.
- **MONITOREAR PEDIDO**
  - El administrador revisa el registro de pedidos, en el cual puede ver pedidos por recibir y recibidos.
  - Cuando la fecha límite ha expirado el administrador llama al proveedor y le indica que su fecha de entrega ha expirado.
  - El administrador puede registrar los pedidos incompletos, cuando el proveedor entrega una parte del pedido, el administrador registra los que se ha entregado y lo demás lo registra por entregar a una fecha límite.
- **REGISTRAR COMPRA**
  - El administrador recibe los productos que el proveedor ha traído
  - El administrador registra los nuevos productos entrantes al almacén.
  - El administrador cancela los productos nuevos a el proveedor.

3.1.2 Caso de uso controlar inventarios CU#02

Controlar Inventarios CU#02



Descripción breve

Este caso de uso permite al administrar los ingresos, egresos y transferencias de los productos. El vendedor puede Ingresar productos de los proveedores y administrar productos de otras sucursales.

Actores

1. Actor Primario: Vendedor
2. Actor Secundario: Proveedor, Cliente, Administrador

Flujo de eventos

Flujo Básico

- **INGRESOS**
  - El proveedor llega con la mercadería que el administrador pidió.
  - El vendedor recibe la mercadería y controla el estado de la misma.
  - La mercadería en mal estado no se recibe y se devuelve al proveedor.
  - El vendedor almacena en el almacén la nueva mercadería
  - El vendedor registra el ingreso de mercadería en el inventario del almacén.
- **EGRESOS**
  - El vendedor vende la mercadería que se encuentra en el almacén
  - El vendedor entrega la mercadería al cliente y registra el egreso por venta en el inventario.

Flujos Alternativos

- **TRANSFERENCIAS**

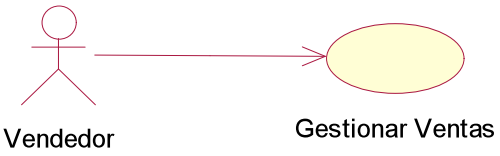
- Se requiere el traspaso de mercadería a otro almacén por venta, daño.
- Cuando el daño se produce se envía la mercadería al taller
- El vendedor registra el egreso del almacén por daño.
- Se puede transferir mercadería porque en uno de los almacenes hay un sobrante de mercadería y se necesita distribuir los productos en los distintos almacenes para tener equilibrado el stock.
- Por venta o distribución de productos el vendedor revisa la mercadería que falta en su almacén y crea un listado y a su vez informa al vendedor de otro almacén.
- El vendedor de otro almacén recibe la información de forma escrita o telefónica, revisa su mercadería, embarca la mercadería que posee y envía al almacén que lo requiere.
- EL vendedor que recibe registra en el inventario como ingreso de otro almacén y el vendedor que envía registra como egreso a otro almacén.

• **PERDIDA**

- El vendedor revisa su inventario y nota que hay faltantes de mercadería.
- El vendedor registra el o los productos que faltan e informa al administrador.
- El administrador registra el egreso como pérdida de mercadería.

3.1.3 Caso de uso gestionar ventas CU#03

Gestionar Ventas CU#03



Descripción breve

Este caso de uso permite al vendedor emitir proformas a los clientes y crear facturas cuando la venta ya se ha realizado.

Actores

1. Actor Primario: Vendedor
2. Actor Secundario: Cliente

Flujo de eventos

Flujo Básico

- El cliente pide información sobre la mercadería que necesita
- El vendedor muestra la mercadería disponible en el local y si no la tiene busca en otros locales.
- El cliente elige la mercadería que va adquirir y elige una forma de pago.
- El vendedor llena la factura con los datos del cliente.
- El cliente cancela su factura.
- EL vendedor despacha la mercadería.

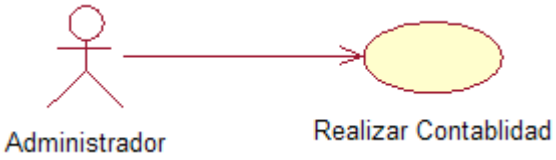
Flujos Alternativos

- **REGISTRAR CLIENTES**
  - El vendedor registra los datos de los nuevos clientes para tener un respaldo informativo, además, puede modificar la información de los mismos.
- **EMITIR PRO FORMA**
  - El cliente únicamente pide una cotización de la mercadería que necesita.
  - El vendedor registra los datos del cliente y la mercadería en una proformas, las

pro formas van numeradas y tienen validez de 15 días, pasado ese período ya no tiene validez.

- **ANULAR FACTURAS**
  - El Vendedor anula las facturas cuando el cliente cancela la compra, o cuando se ha cometido un error de escritura.

3.1.4 Caso de uso realizar contabilidad CU#04

Realizar Contabilidad CU#04

Descripción breve Este caso de uso describe como se piensa llevar a cabo la contabilidad en Muebles Prieto.
Actores 1. Actores primarios: Administrador
Flujos de eventos ***Flujo Básico*** • **CREAR PLAN DE CUENTAS** o Crear plan de cuentas clasificándolas en cinco grupos: Activo, pasivo, patrimonio, ventas y gastos. • **REGISTRAR ASIENTOS** o El administrador registra los asientos de transacciones en libro diario con fecha, detalle, valores del debe y haber. • **REALIZAR LA MAYORIZACIÓN** o El administrador registra todos los asientos del libro diario en libro mayor clasificados por cuenta. • **CERRAR PERIODO** o Cuando se termine el periodo, el administrador realiza los asientos de ajuste pendientes y asientos de cierre en el libro diario y mayor. • **CREAR ESTADOS FINANCIEROS DE CIERRE** o El administrador crea los tres estados financieros: Balance general, estado de rentas y gastos, estado de superávit-ganancias retenidas.

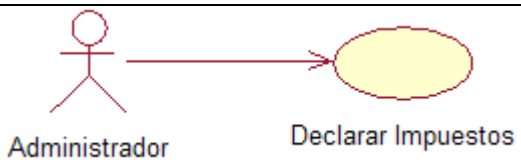
Flujos alternativos

- **ALMACENAR COMPROBANTES**
    - El administrador archiva las facturas de compras y ventas, comprobantes de retención, cortes de movimientos de tarjetas de crédito, estados de cuenta de bancos, papeletas de depósitos, planillas de aportes, recibos, roles de pago, vales entre otros.
- **CONTROLAR KÁRDEX**
    - El administrador detalla el nombre del artículo, la existencia máxima, la existencia mínima, el método de valoración de mercancías que convenga: FIFO, LIFO o promedio ponderado, la unidad de medida, los ingresos, egresos y saldos.
- **HACER BALANCE DE COMPROBACIÓN**
    - En cualquier momento el administrador puedo revisar los valores de debe, haber y saldos del plan de cuentas y comprobar que todo este cuadrado mediante el balance de comprobación.
- **CREAR AJUSTES**
    - Durante un periodo o ciclo contable pueden darse ajustes acumulados, diferidos, depreciaciones, amortizaciones, consumos, provisiones, regulaciones entre otros.
- **REALIZAR CONCILIACIÓN BANCARIA**
    - El administrador comprueba los saldos de las cuentas bancarias cuando llegue el estado de cuenta de los bancos.
- **REALIZAR ANÁLISIS FINANCIEROS**
    - El administrador crea análisis financieros verticales y horizontales.
- **CONTROLAR CUENTAS POR COBRAR**
    - El vendedor registran nuevas cuentas por cobrar, cancelación de cuentas por cobrar y devoluciones. El administrador determinará si hay cuentas que se convierten en incobrables.

- **CONTROLAR CUENTAS POR PAGAR**
    - El vendedor registra cuentas por pagar. El administrador registra el pago de cuentas por pagar y descuentos.

3.1.5 Caso de uso declarar impuestos CU#05

Declarar Impuestos CU#05



Descripción breve

Este caso de uso describe el procedimiento para declarar el Impuesto al Valor Agregado (IVA) y el Impuesto a la Renta (IR).

Actores

2. Actores primarios: Administrador

Flujos de eventos

Flujo Básico

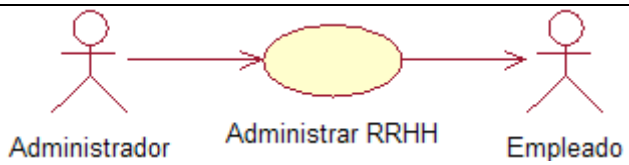
- **CALCULAR IVA POR DECLARAR**
  - El administrador cada mes totaliza los valores de: IVA causado, retenciones del uno por ciento a la renta, treinta por ciento al IVA, comisiones de tarjetas de crédito y extras por emisión de cheques de tarjetas de crédito, para las facturas de compras se suman todos los totales, el IVA y el IVA cero.
  - El administrador llena el formulario del SRI con los valores correspondientes y cancela en la entidad autorizada para recibir la declaración.
  - El administrador registra los valores de los impuestos y archiva los comprobantes.

Flujos alternativos

- **CALCULAR IR POR DECLARAR**
  - Para el periodo correspondiente el administrador calcula la base imponible para el IR y según la tabla de ingresos anuales el valor del impuesto.
  - El administrador llena el formulario de declaración del IR y cancela en la entidad autorizada para recibir la declaración.
  - El administrador registra los valores del impuesto y archiva los comprobantes.

3.1.6 Caso de uso administrar RRHH CU#06

Administrar RRHH CU#06



Descripción breve

Este caso de uso especifica las tareas del administrador en recursos humanos. El administrador establece políticas salariales, contrata empleados, registra anticipos, multas y elabora liquidaciones todo de acuerdo con el código laboral.

Actores

3. Actores primarios: Administrador
4. Actores secundarios: Empleado

Flujos de eventos

Flujo Básico

- **ESTABLECER POLÍTICAS SALARIALES**
    - Muebles Prieto según el código laboral ha establecido que el empleado del almacén tiene derecho a:
        - Un sueldo básico.
        - Remuneración por horas extraordinarias y suplementarias del cincuenta y cien por ciento más de la hora normal respectivamente.
        - No serle retenido más del diez por ciento del sueldo por multas del reglamento interno.
        - Pagar por seis días si falta injustificadamente más de media jornada y si es jornada completa solo por cinco días.
        - Vacaciones de quince días de descanso obligatorio e ininterrumpido, incluidos días no laborables. Tendrá un día extra por cada año después de los cinco años pero no más de treinta días.

- Liquidaciones por vacaciones, décimo tercera remuneración y décimo cuarta remuneración.
- Seguro en el IESS.
- Fondo de reserva equivalente a un mes de sueldo por cada año después del primer año. Es depositado anualmente.
- Bonificaciones por desahucio equivalente al veinte y cinco por ciento de la última remuneración multiplicada por el número de años trabajados.

- **CREAR CONTRATO**
  - El administrador elabora un contrato que tome en cuenta todas las regulaciones del código laboral y reglamento interno y obtiene el visto bueno del Ministerio de Trabajo.
- **CONTRATAR EMPLEADO**
  - El administrador expone un aviso de necesidad de personal.
  - El aspirante presenta su hoja de vida más certificados.
  - El administrador selecciona los candidatos más aptos.
  - El administrador contrata al aspirante y verifica sus datos.
  - El aspirante firma un contrato de trabajo.
  - El administrador notifica al IESS.
- **HACER APROBAR CONTRATO**
  - El administrador le hace firmar al empleado nuevo un contrato de trabajo aprobado y lo hace reconocer como válido en el ministerio.

Flujos alternativos

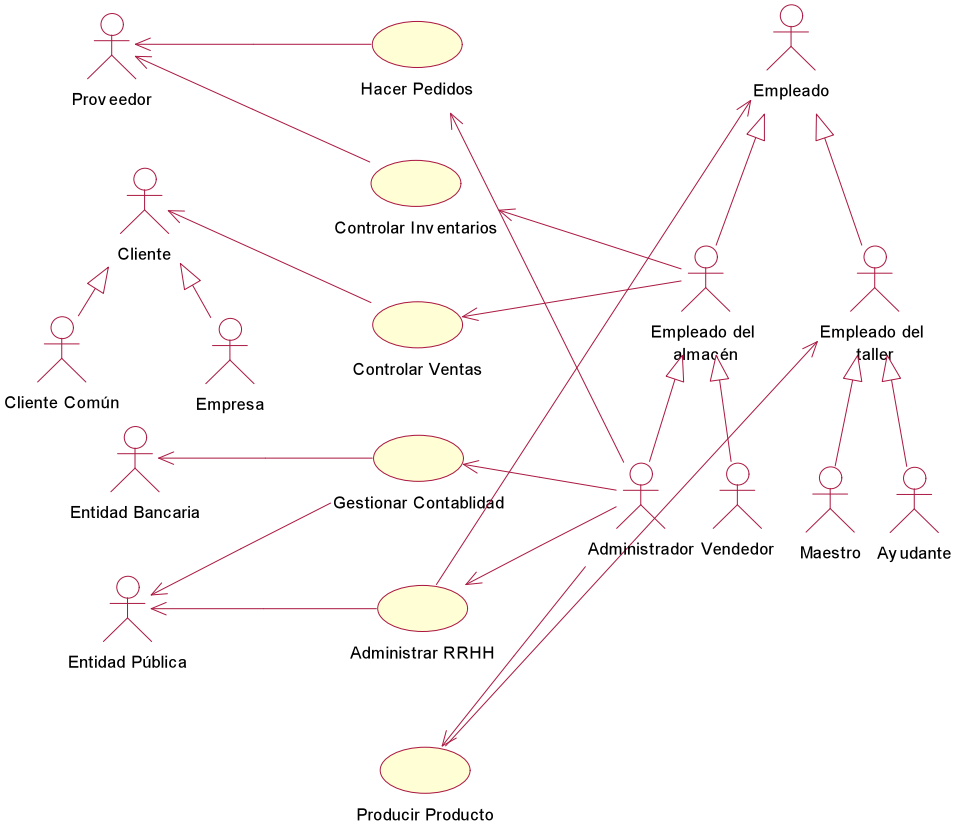
- **REGISTRAR ANTICIPO**
  - El administrador registra la fecha y valor de un adelanto al empleado.
  - El empleado recibe el adelanto.
- **REGISTRAR MULTA**

- El administrador registra la multa que afecta a la remuneración.
- El empleado reconoce la multa.

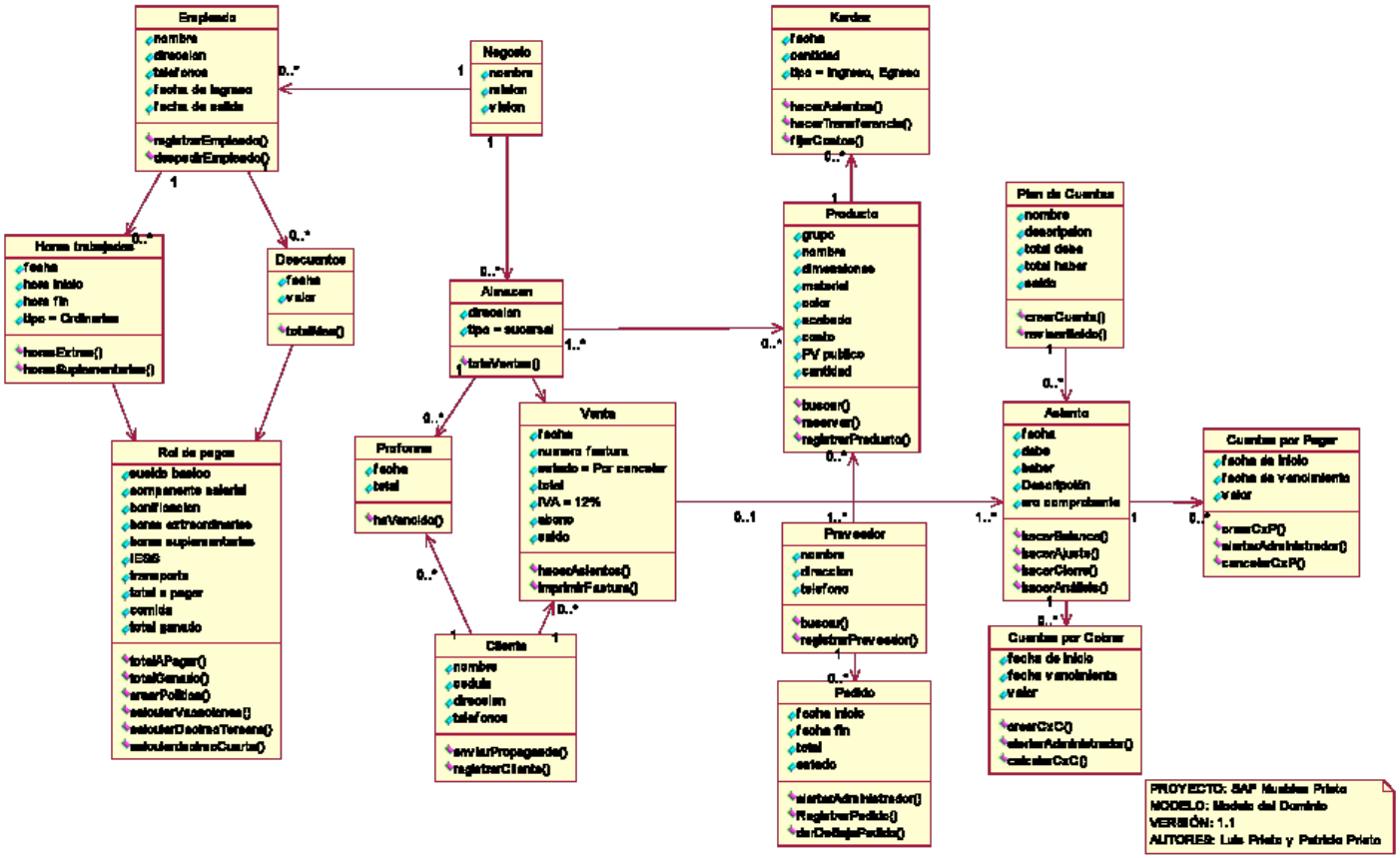
- **GENERAR ROL DE PAGOS**
  - El administrador generalmente pero no siempre cada fin de mes, verifica las horas suplementarias, horas extraordinarias, adelantos, descuentos, multas de cada empleado, más los valores de aporte al IESS obligatorios, préstamos al IESS e impuesto a la renta si se causa.
  - El administrador calcula el total e imprime el rol.
  - El empleado acepta y firma el rol de pago.
- **ELABORAR LIQUIDACIÓN**
  - El administrador puede liquidar empleado por la décimo tercera remuneración, décimo cuarta remuneración, vacaciones y desahucio del empleado.
  - El empleado firma y acepta la liquidación
  - Si es una liquidación por desahucio del empleado el administrador notifica al IESS.
- **CONTROL DE VACACIONES**
  - El administrador controla los días que el empleado toma de los días de vacaciones que le corresponden anualmente.
- **REGISTRAR FONDO DE RESERVA**
  - Cumplido los dos años de trabajo del empleado, el administrador deposita el valor del fondo de reserva en el IESS.
  - El administrador registra el valor del fondo depositado y archiva el comprobante.
- **REGISTRAR APORTES AL IESS**
  - El administrador cada mes ingresa al sistema del IESS vía Web, genera los comprobantes de pago de aporte al IESS y realiza el pago.
  - El administrador registra el valor del aporte y archiva los comprobantes.

3.2 Análisis

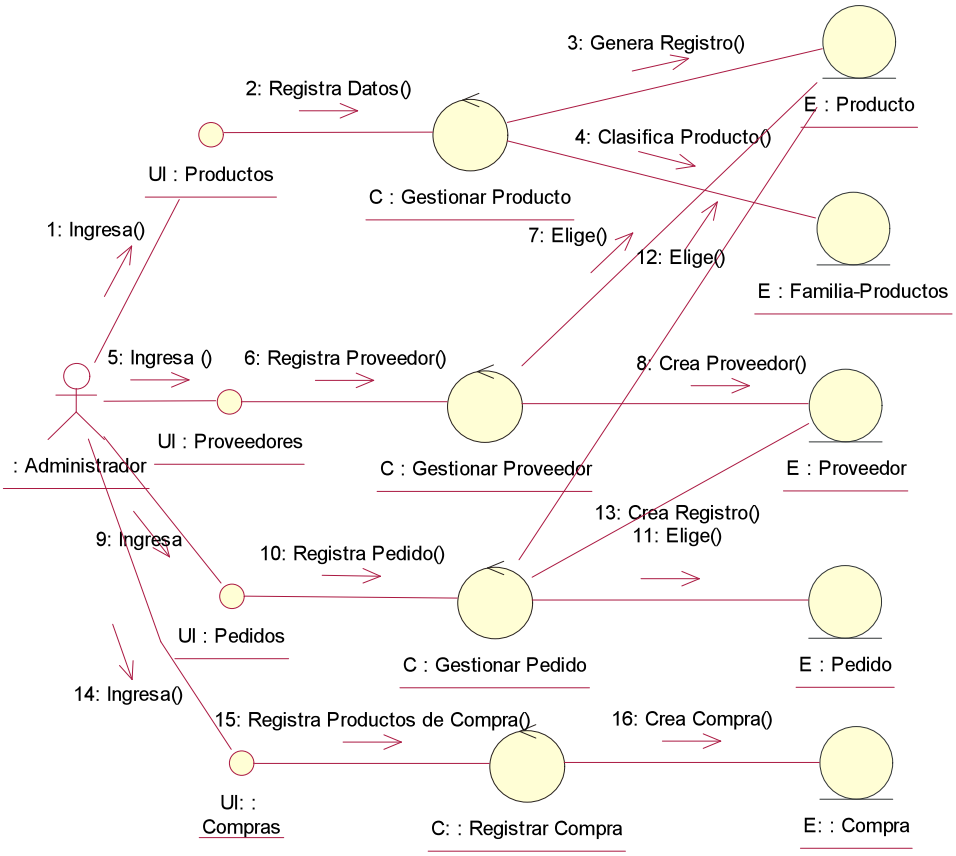
3.2.1 Modelo del Negocio Muebles Prieto



3.2.2 Modelo del Dominio



3.2.3 Realización de caso de uso administrar proveedores y pedidos CU#01

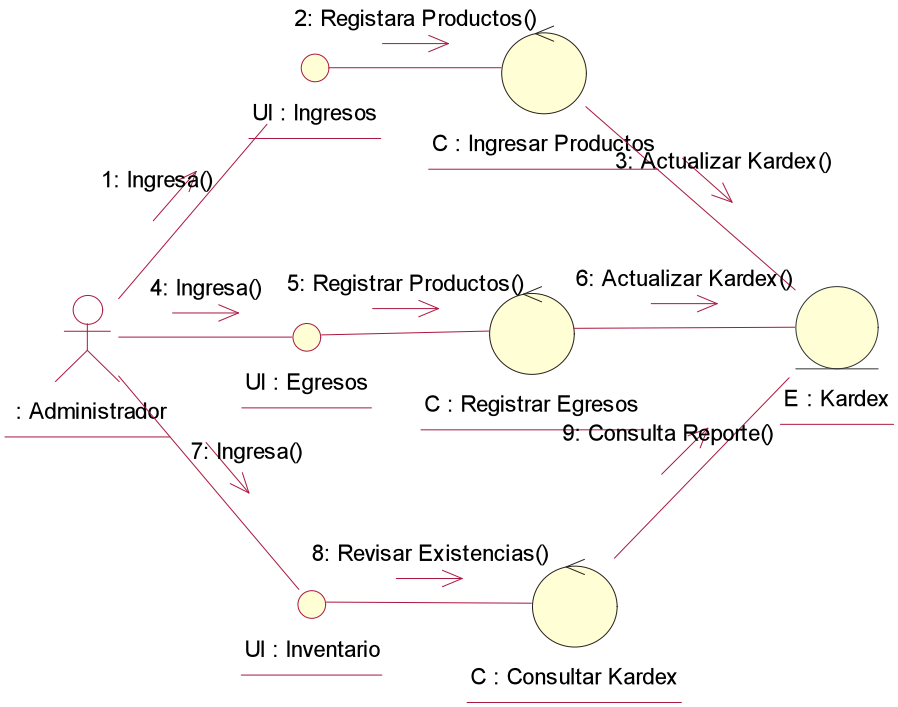


PROYECTO: SAF Muebles Prieto
MODELO: Diagrama de Colaboración - Análisis
CASO DE USO: Administrar Proveedores y Pedidos CU#1
VERSIÓN: 2.0
AUTORES: Luis Prieto y Patricio Prieto

3.2.3.1 Flujo de eventos CU#01: Administrar proveedores y pedidos

1. El administrador ingresa a la interfaz de productos
2. El administrador registra el nuevo producto
3. El control genera el registro de los datos
4. El control clasifica el producto según la familia
5. El administrador ingresa a la interfaz de proveedor
6. El administrador registra al nuevo proveedor
7. El control elige el producto seleccionado
8. El control crea el proveedor
9. El administrador ingresa a la interfaz de pedidos
10. El administrador registra el nuevo pedido
11. El control elige el producto de la base
12. El control elige el proveedor de la base
13. El control crea el registro del nuevo pedido
14. El administrador Ingresa a la Interfaz compras
15. El administrador registra los productos de entrada
16. El control crea la compra en compra

3.2.4 Realización de caso de uso controlar inventarios CU#02

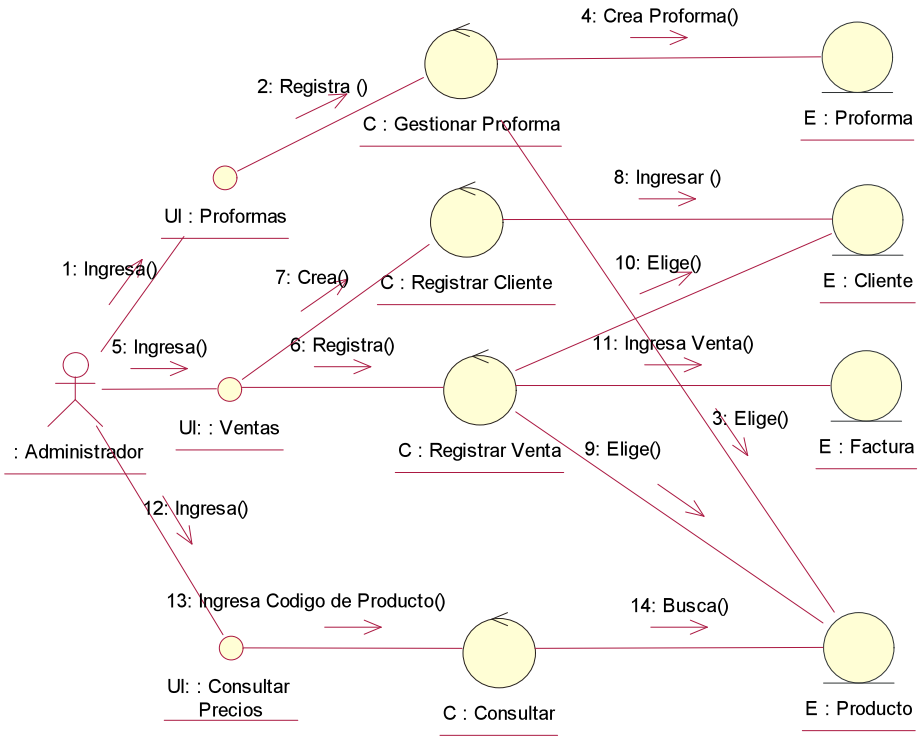


PROYECTO: SAF Muebles Prieto
MODELO: Diagrama de Colaboración - Analisis
CASO DE USO: Controlar Inventarios CU#2
VERSIÓN: 2.0
AUTORES: Luis Prieto y Patricio Prieto

3.2.4.1 Flujo de eventos CU#02: Controlar inventarios

1. El vendedor ingresa a la interfaz ingresos
2. El vendedor registra los productos de entrada
3. El control actualiza los nuevos productos en el kárdex
4. El vendedor Ingresa a la interfaz de Egresos
5. El administrador registra los productos de salida
6. El control actualiza el kárdex
7. El vendedor ingresa a la interfaz inventario
8. El vendedor consulta las existencias de un producto
9. el control emite el reporte del producto del kárdex

3.2.5 Realización de caso de uso del análisis gestionar ventas CU#03

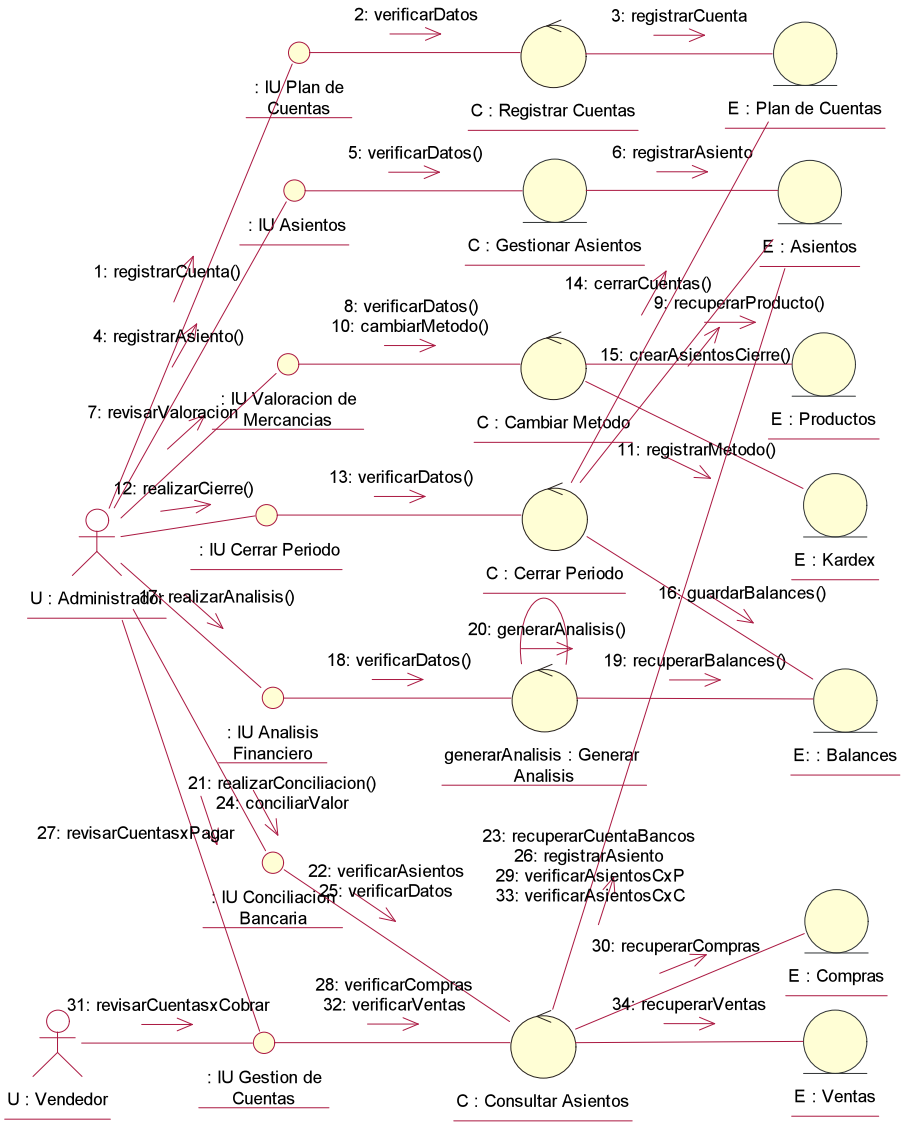


PROYECTO: SAF Muebles Prieto
MODELO: Diagrama de Colaboración - Análisis
CASO DE USO: Gestionar Ventas CU#3
VERSIÓN: 2.0
AUTORES: Luis Prieto y Patricio Prieto

3.2.5.1 Flujo de eventos CU#03: Gestionar ventas

1. El vendedor ingresa a la interfaz pro forma
2. El vendedor registra el detalle de la pro forma
3. El control ingresa el producto
4. EL control crea la pro forma
5. El administrador ingresa a la interfaz ventas
6. El administrador registra datos de la factura
7. El administrador ingresa un nuevo cliente si no existe
8. El control crea el cliente en cliente
9. EL control elige los productos
10. EL control elige el cliente
11. El control crea la venta
12. El administrador ingresa a la interfaz consultar precios
13. El administrador ingresa el código del producto
14. El control busca el producto en producto

3.2.6 Realización de caso de uso realizar contabilidad CU#04



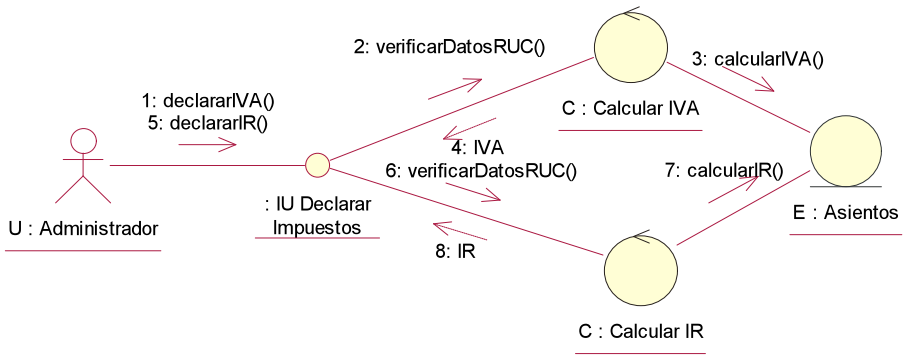
PROYECTO: SAF Muebles Prieto
MODELO: Diagrama de Colaboración - Análsis
CASO DE USO: Realizar Contabilidad CU#04
VERSIÓN: 2.0
AUTORES: Luis Prieto y Patricio Prieto

3.2.6.1 Flujo de eventos CU#04: Realizar contabilidad

1. El administrador crear una cuenta.
2. La interfaz manda a verificar los datos.
3. El control verifica los datos y registra la cuenta.
4. El administrador crear un asiento.
5. La interfaz manda a verificar los datos.
6. El control verifica los datos y registra el asiento
7. El administrador revisa la valoración de mercancías de un producto.
8. La interfaz manda a obtener información de ese producto.
9. El control recupera la información del producto
10. El administrador decide cambiar el método de valoración del producto.
11. El control registra el cambio en el kárdex del producto.
12. El administrador realiza un cierre.
13. La interfaz verifica los datos del cierre.
14. El control cierra las cuentas.
15. El control realiza los asientos de cierre.
16. El control registra los balances correspondientes al cierre.
17. El administrador compara mediante análisis verticales u horizontales los balances.
18. La interfaz manda a verificar los datos de los balances.
19. El control recupera los balances seleccionados.
20. El control compra los balances y entrega el informe.
21. El administrador realiza la conciliación.
22. La interfaz manda a verificar los datos para la conciliación.
23. El control recupera los datos de la cuenta bancos.
24. El administrador concilia un valor.
25. La interfaz manda a verificar los datos del valor conciliado.
26. El control registra el valor conciliado en el asiento.
27. El administrador revisa las cuentas por pagar.
28. La interfaz manda a verificar las compras pendientes.

29. El control trae los asientos de las compras pendientes.
30. El control trae las compras pendientes que pertenecen a los asientos.
31. El vendedor revisa las cuentas por cobrar.
32. La interfaz manda a revisar las ventas con saldos.
33. El control trae los asientos de las ventas pendientes.
34. El control recupera las ventas pendientes que pertenecen a los asientos.

3.2.7 Realización de caso de uso declarar impuestos CU#05

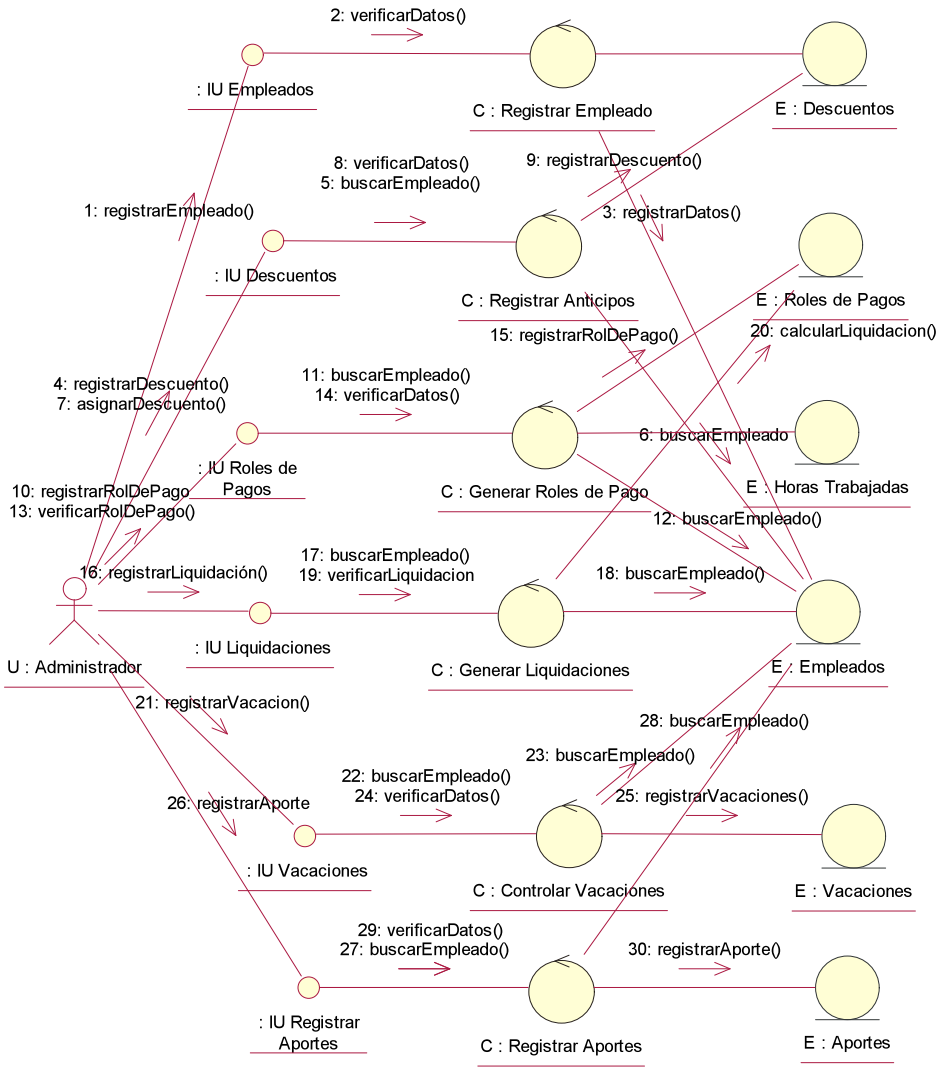


PROYECTO: SAF Muebles Prieto
MODELO: Diagrama de Colaboración - Análsis
CASO DE USO: Declarar Impuestos CU#05
VERSIÓN: 2.0
AUTORES: Luis Prieto y Patricio Prieto

3.2.7.1 Flujo de eventos CU#05: Declarar impuestos

1. El administrador declara el IVA por RUC.
2. La interfaz manda a verificar los datos del IVA que pertenecen al RUC.
3. El control recupera la información del IVA de ese RUC.
4. El control presenta el resumen del cálculo del IVA.
5. El administrador declara el IR por RUC.
6. La interfaz manda a verificar los datos del IR del RUC.
7. El control recupera la información del IR de ese RUC.
8. El control presenta el resumen del cálculo del IR.

3.2.8 Realización de caso de uso administrar RRHH CU#06



PROYECTO: SAF Muebles Prieto
MODELO: Diagrama de Colaboración - Análisis
CASO DE USO: Administrar RRHH CU#6
VERSIÓN: 2.0
AUTORES: Luis Prieto y Patricio Prieto

3.2.8.1 Flujo de eventos CU#06: Administrar recursos humanos

1. El administrador registra el empleado.
2. La interfaz manda a verificar los datos del empleado.
3. El control registra los datos del empleado.
4. El administrador registra un descuento a un empleado
5. La interfaz manda a buscar al empleado.
6. El control recupera el empleado.
7. El administrado le asigna el descuento.
8. La interfaz manda a verificar los datos del descuento.
9. El control registra el descuento.
10. El administrador realiza un rol de pago.
11. La interfaz manda a buscar al empleado.
12. El control recupera el empleado.
13. El administrador verifica que estén correctos los valores del rol de pago.
14. La interfaz manda a verificar los datos modificados por el administrador.
15. El control registra el rol de pago una vez verificado su validez.
16. El administrador registra una liquidación.
17. La interfaz manda a buscar al empleado.
18. El control recupera al empleado y presenta el informe.
19. La interfaz manda a verificar los valores de la liquidación.
20. El control verifica los datos de la liquidación, calcula los valores y presenta el informe.
21. El administrador registra la vacación.
22. La interfaz manda a buscar al empleado.
23. El control recupera al empleado.
24. La interfaz manda a verificar los datos de la vacación.
25. El control después de verificar, almacena los datos de la vacación.
26. El administrador registra un aporte para un empleado.
27. La interfaz manda a buscar a un empleado.
28. El control recupera al empleado.

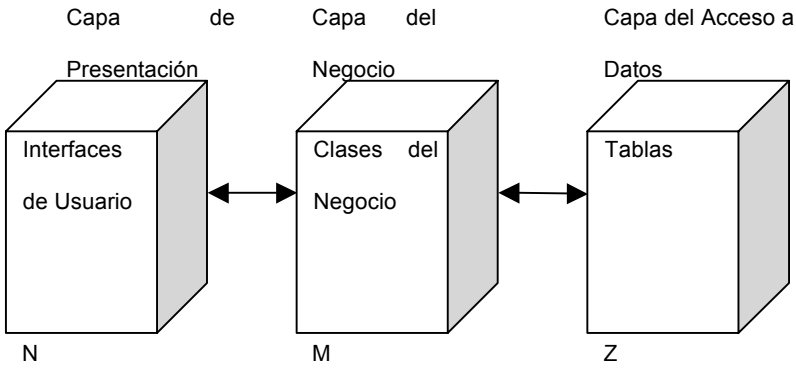
29. La interfaz manda a verificar los datos del aporte.
30. El control verifica y guarda los datos del aporte.

3.3 Diseño

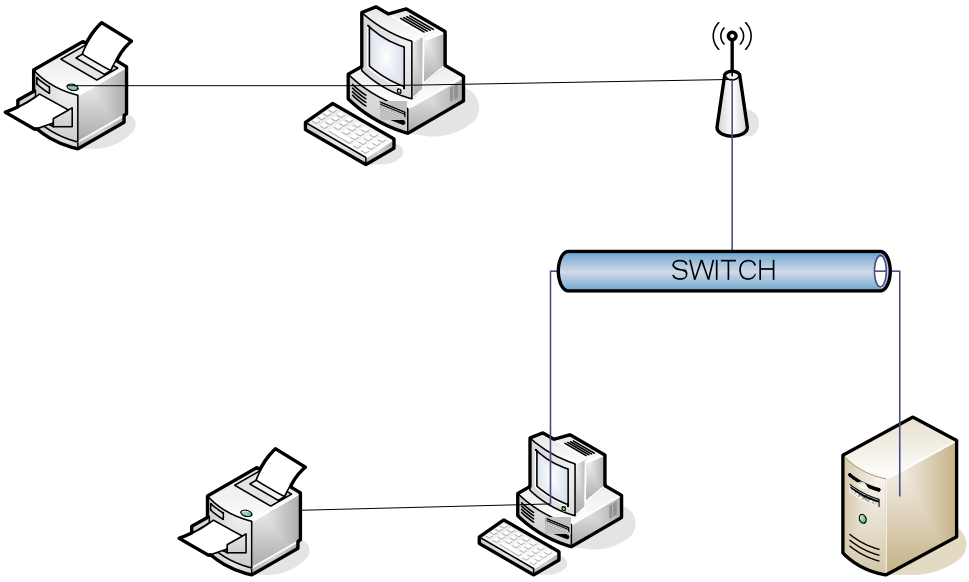
3.3.1 Diseño Arquitectónico

Descripción de la arquitectura:

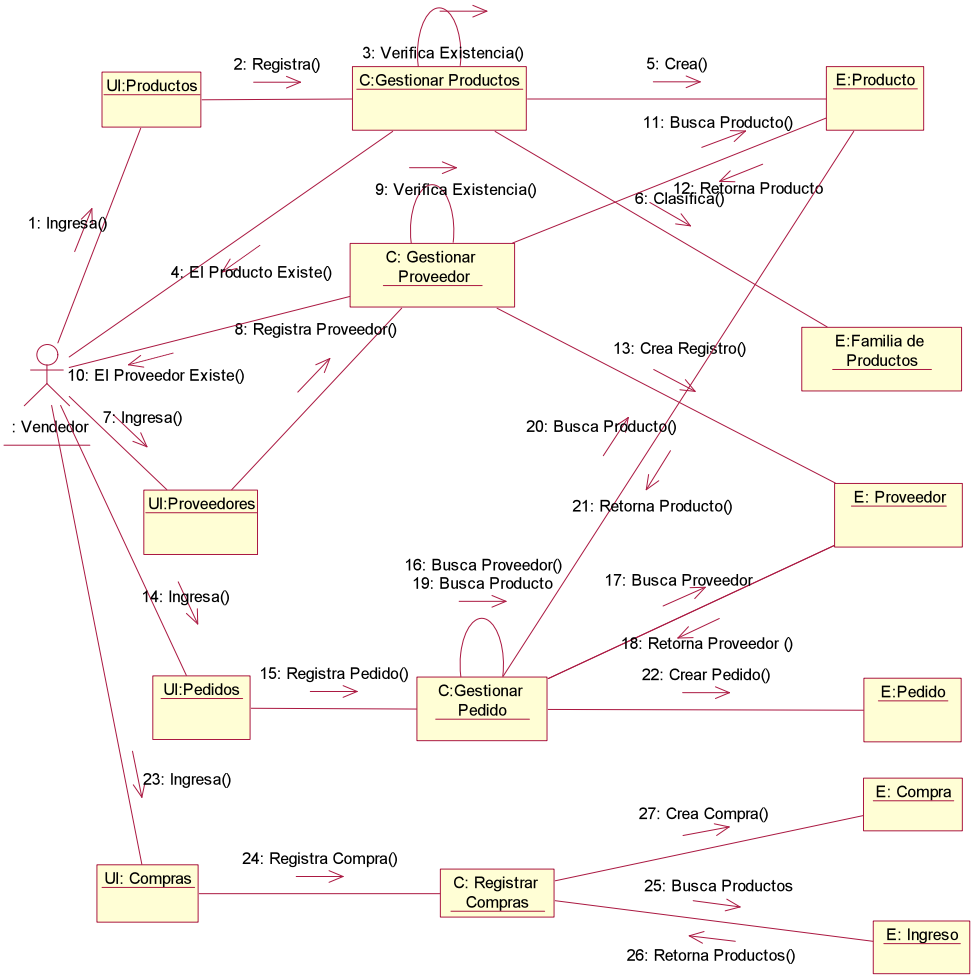
La arquitectura escogida es la de tres capas que separa la aplicación en capa de presentación, capa de las reglas del negocio y la capa de acceso de datos. La capa de presentación es en donde se validan los datos y peticiones que hace y recibe el usuario mediante la interfaz, además esta capa es responsable de la comunicación con la capa intermedia o capa de reglas del negocio. La capa de las reglas del negocio como su nombre lo indica se encarga de las reglas que permiten que el negocio funcione, esta puede dividirse en más capas pero condiciona el rendimiento. La capa de acceso de datos permite que la aplicación lea y almacene datos persistentes. Esta arquitectura proporciona beneficios como: escalabilidad, flexibilidad y capacidad para aumentar el rendimiento. Estas características son importantes para la reducción de costos en mantenimiento y cambios que se puedan introducir a una aplicación. Lógicamente se representa de la siguiente manera:



3.3.2 Diagrama de Despliegue del Diseño

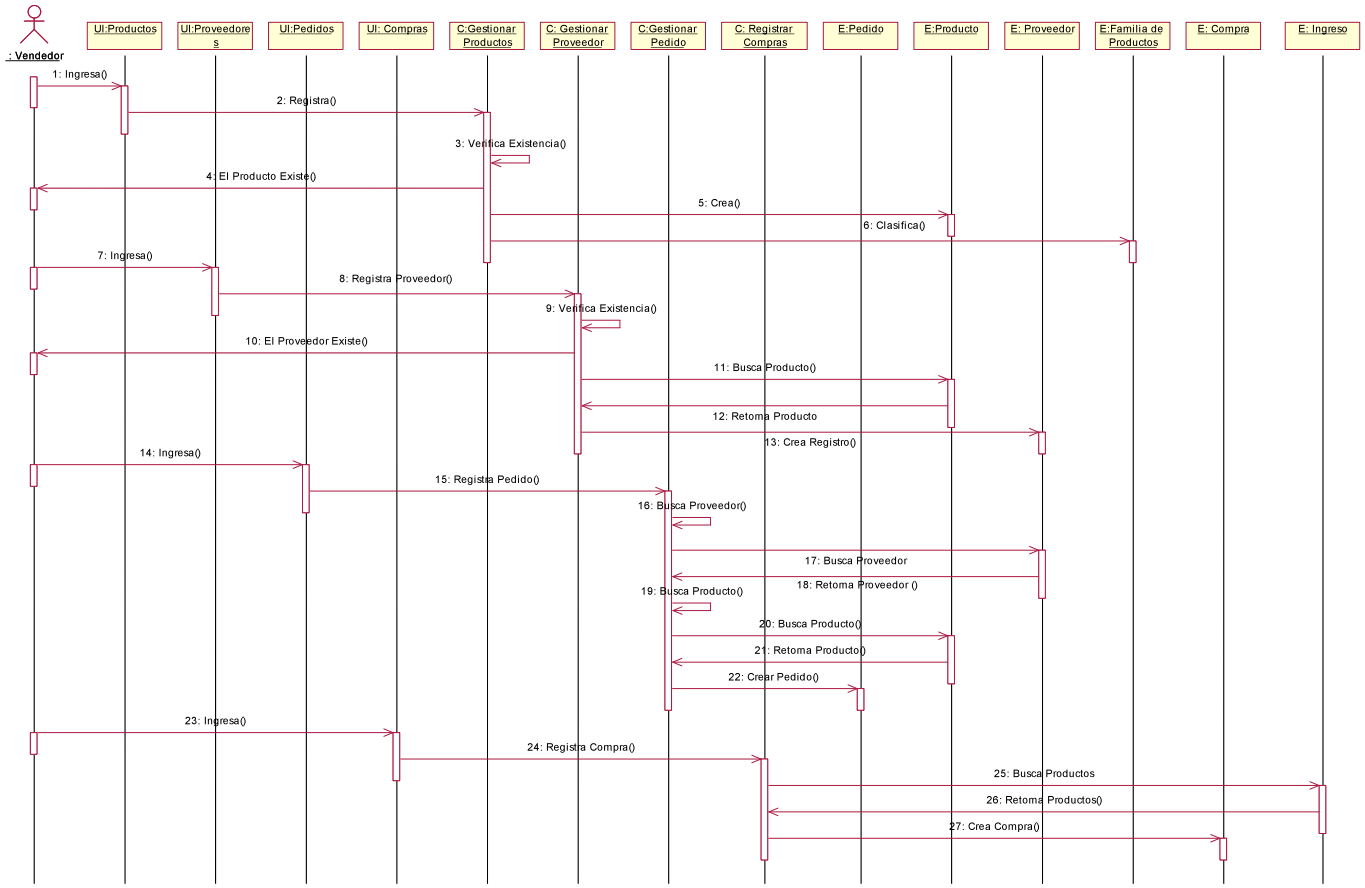


3.3.3 Realización de caso de uso de administrar proveedores y pedidos CU#01

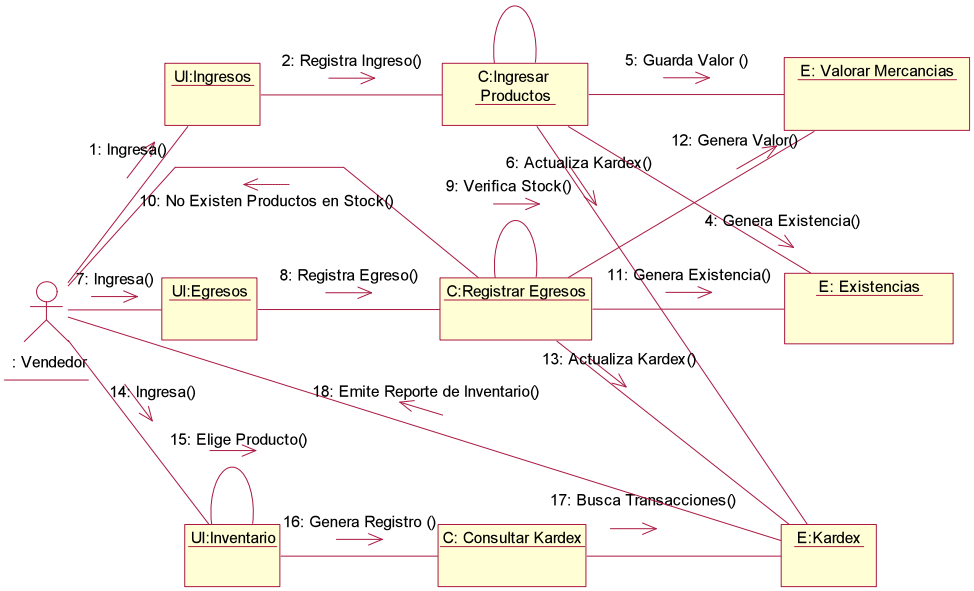


PROYECTO: SAF Muebles Prieto
MODELO: Diagrama de Colaboración - Diseño
CASO DE USO: Administrar Proveedores y Pedidos CU#1
VERSIÓN: 2.0
AUTORES: Luis Prieto y Patricio Prieto

3.3.3.1 Diagrama de secuencia administrar proveedores y pedidos

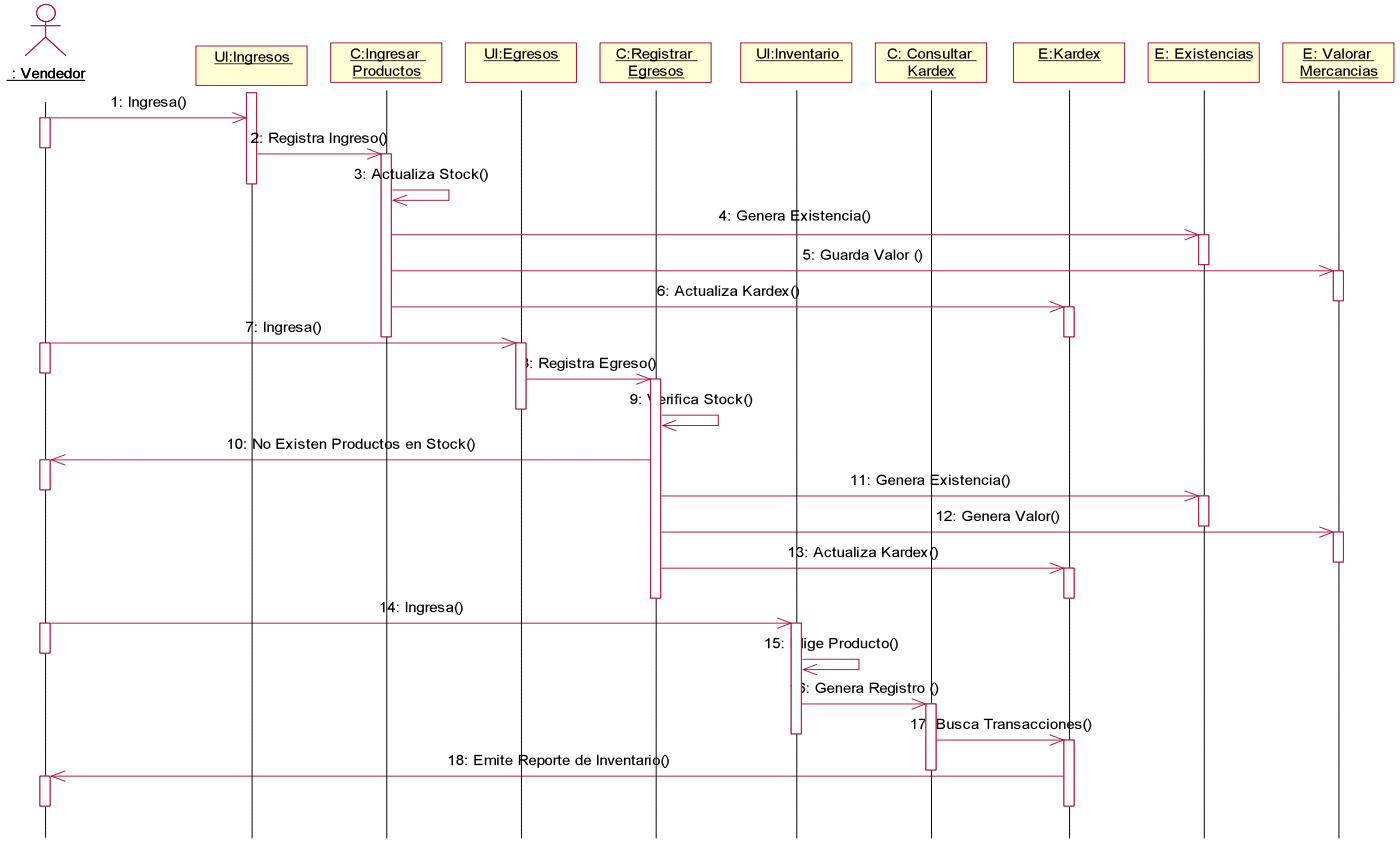


3.3.4 Realización de caso de uso controlar inventarios CU#02

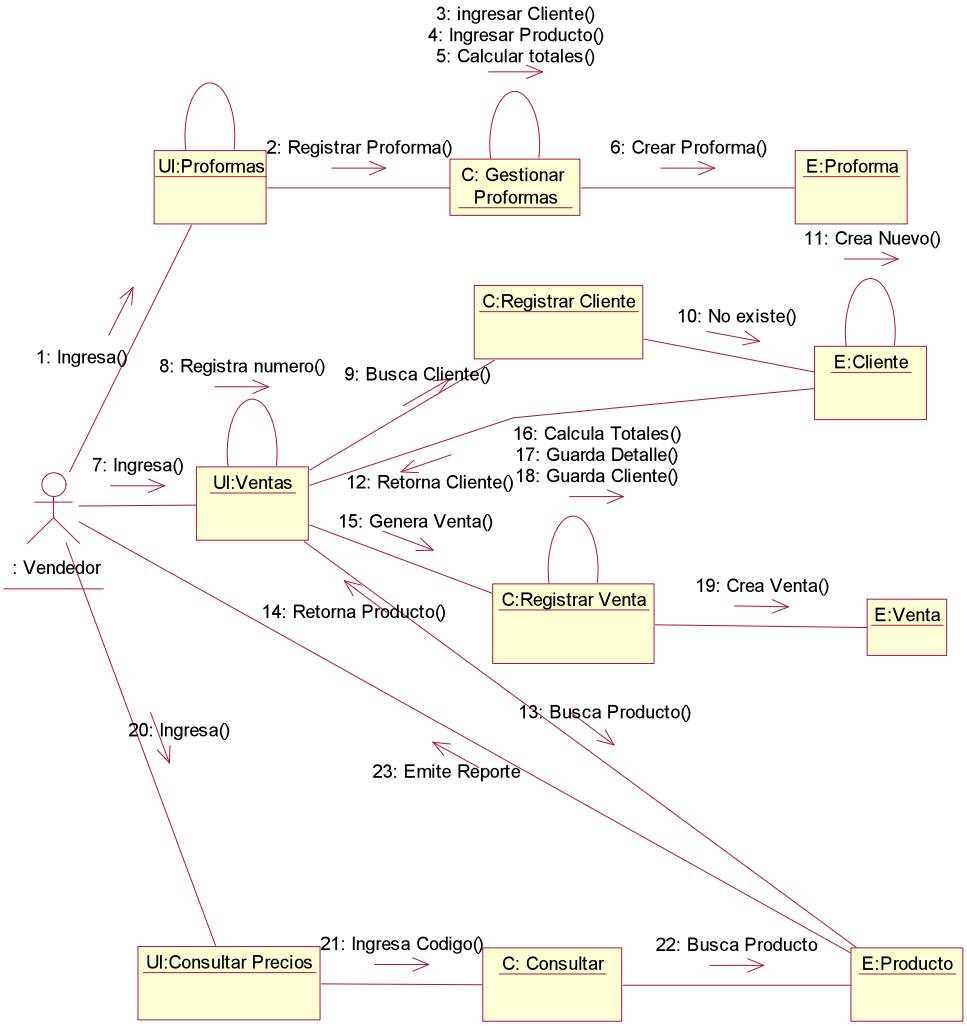


PROYECTO: SAF Muebles Prieto
MODELO: Diagrama de Colaboración - Analisis
CASO DE USO: Controlar Inventarios CU#2
VERSIÓN: 2.0
AUTORES: Luis Prieto y Patricio Prieto

3.3.4.1 Diagrama de secuencia controlar inventarios

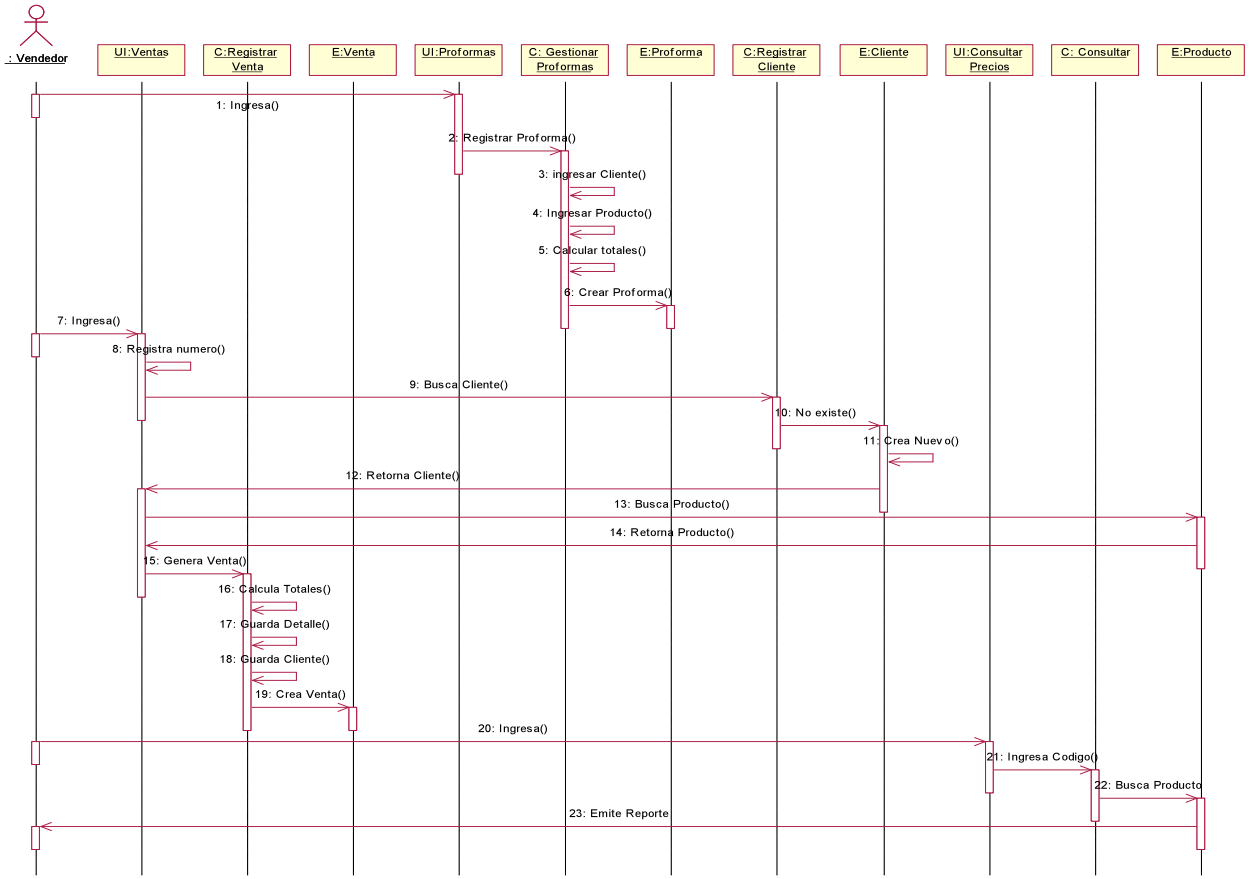


3.3.5 Realización de caso de uso gestionar ventas CU#03

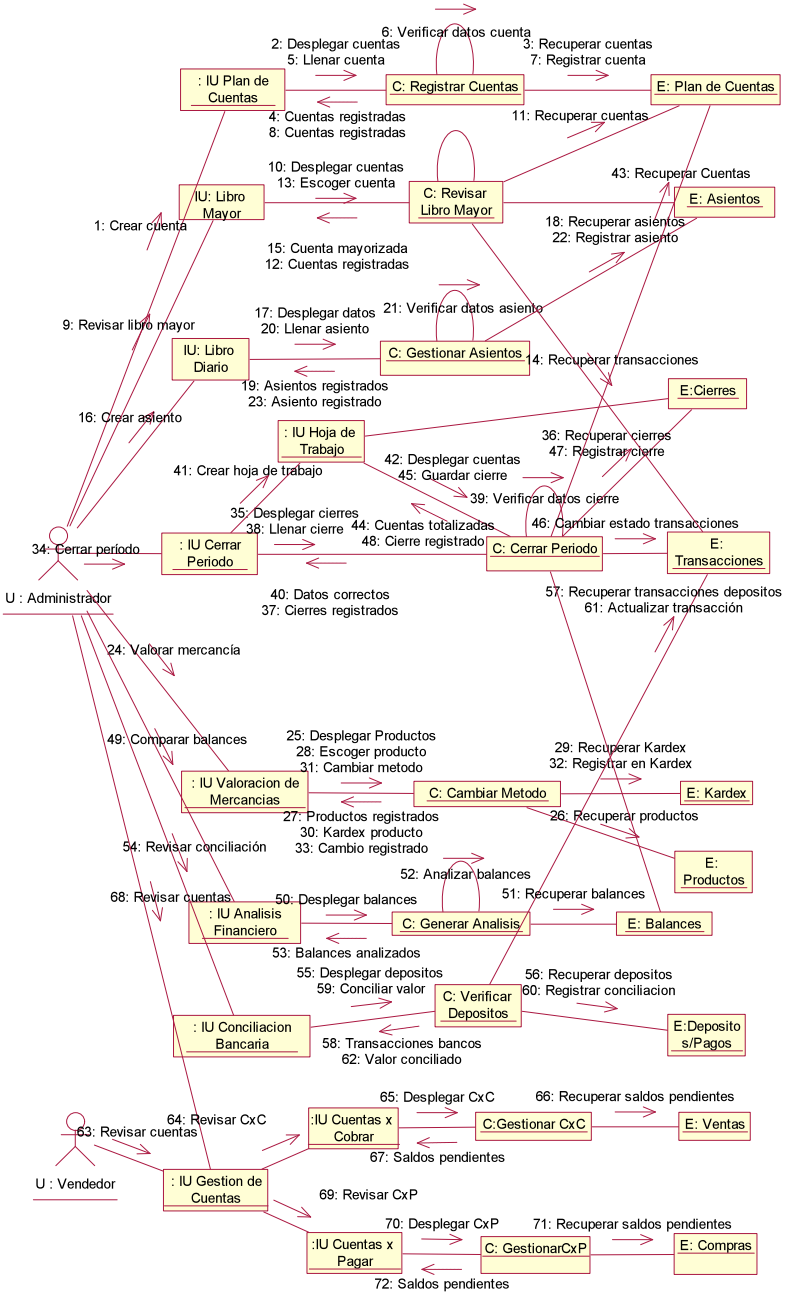


PROYECTO: SAF Muebles Prieto
MODELO: Diagrama de Colaboración - Diseño CU#3
CASO DE USO: Gestionar Ventas
VERSIÓN: 2.0
AUTORES: Luis Prieto y Patricio Prieto

3.3.5.1 Diagrama de secuencia gestionar ventas

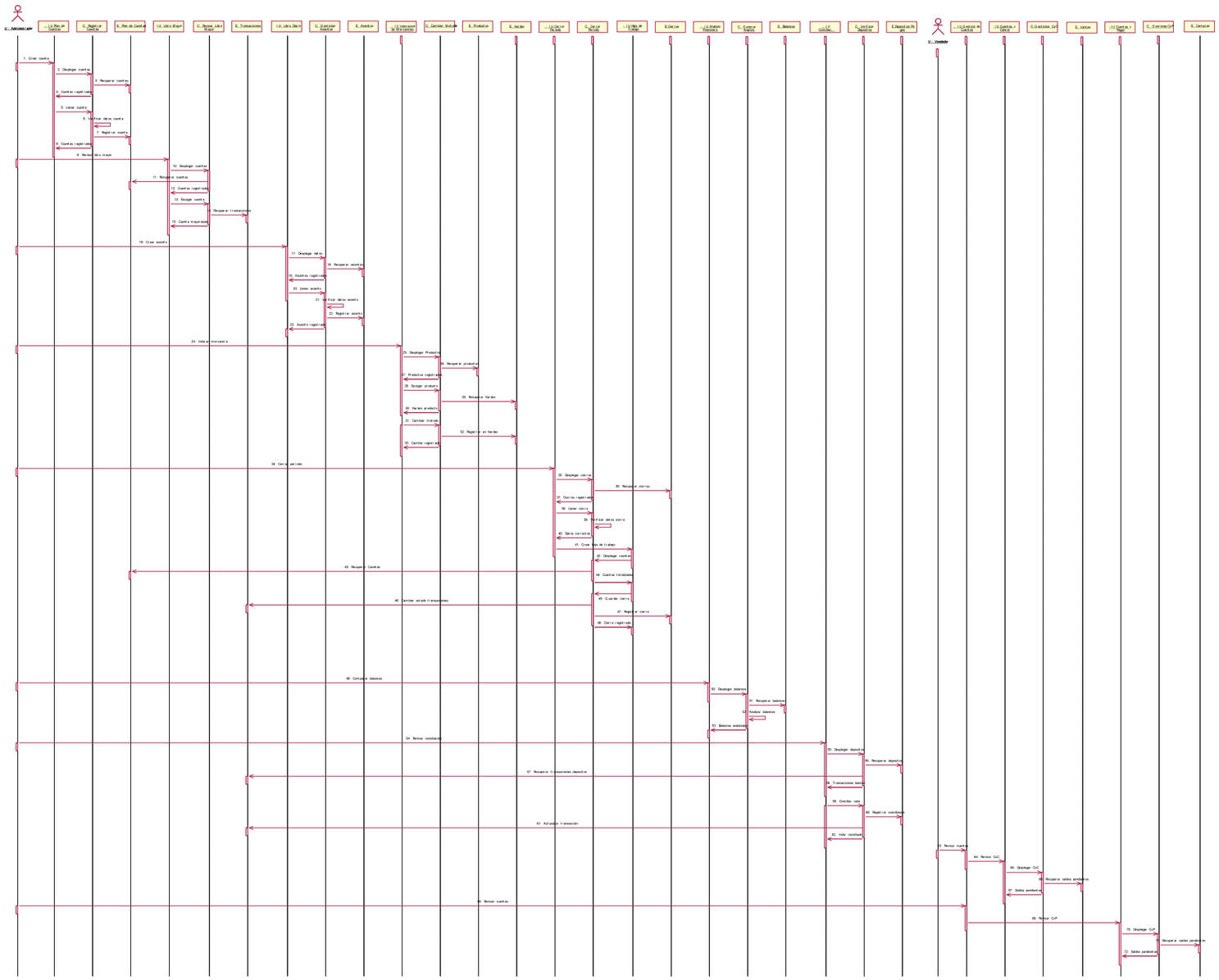


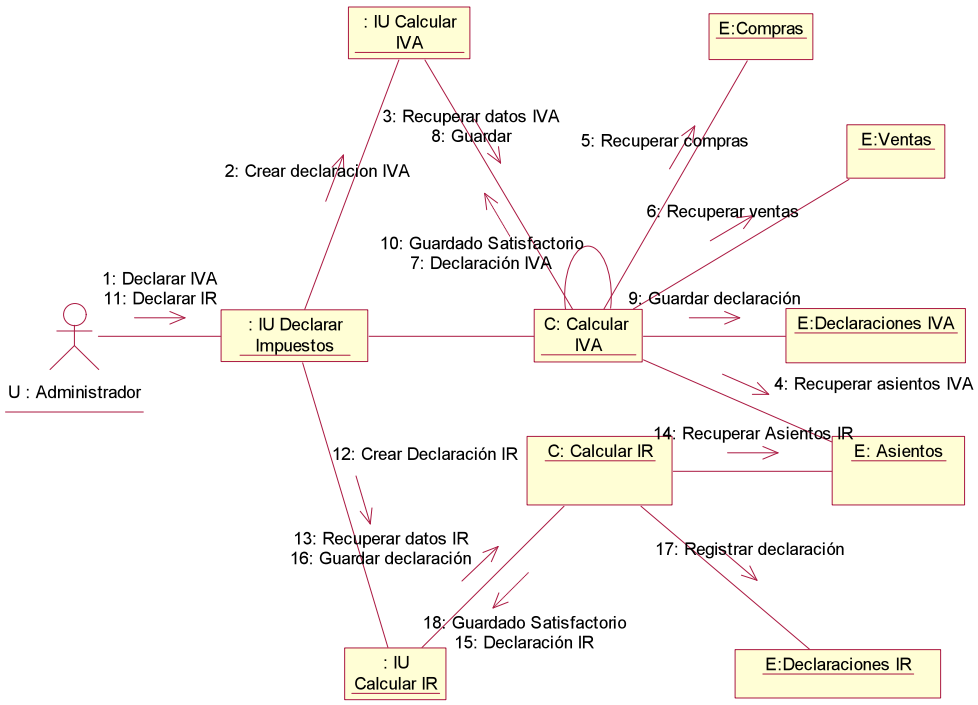
3.3.6 Realización de caso de uso realizar contabilidad CU#04



PROYECTO: SAF Muebles Prieto
MODELO: Diagrama de Colaboración - Diseño
CASO DE USO: Realizar Contabilidad CU#04
VERSIÓN: 0.1
AUTORES: Luis Prieto y Patricio Prieto

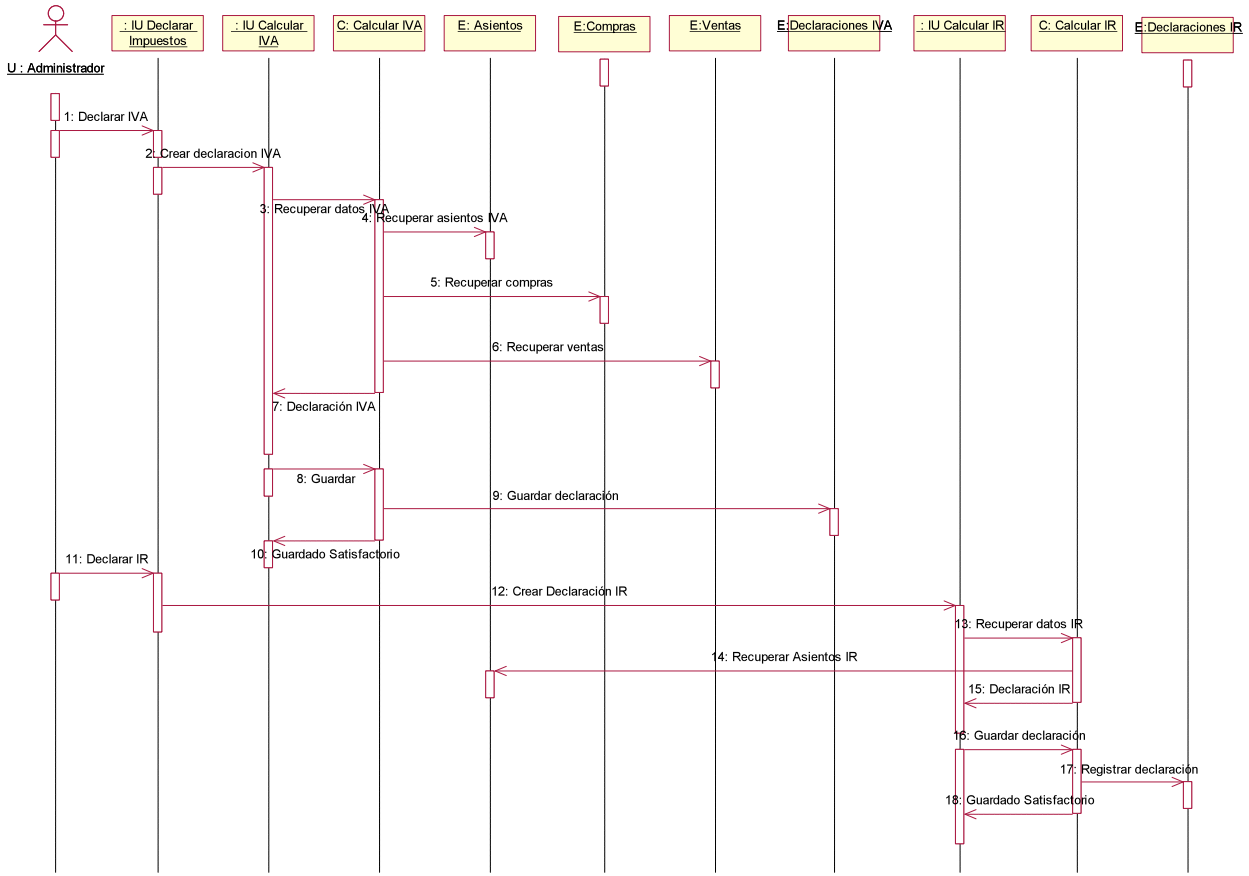
3.3.6.1 Diagrama de secuencia realizar contabilidad



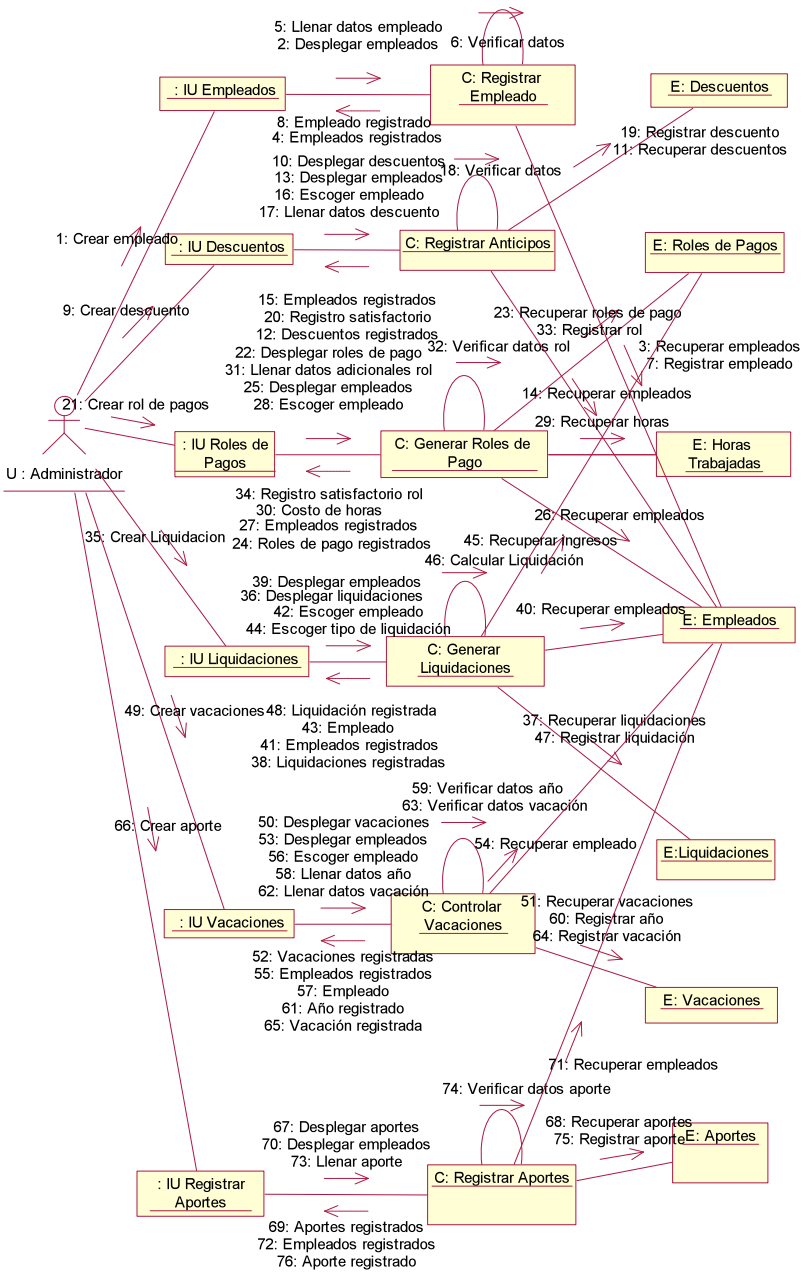


PROYECTO: SAF Muebles Prieto
MODELO: Diagrama de Colaboración - Diseño
CASO DE USO: Declarar Impuestos CU#05
VERSIÓN: 2.0
AUTORES: Luis Prieto y Patricio Prieto

3.3.7.1 Diagrama de secuencia declarar impuestos

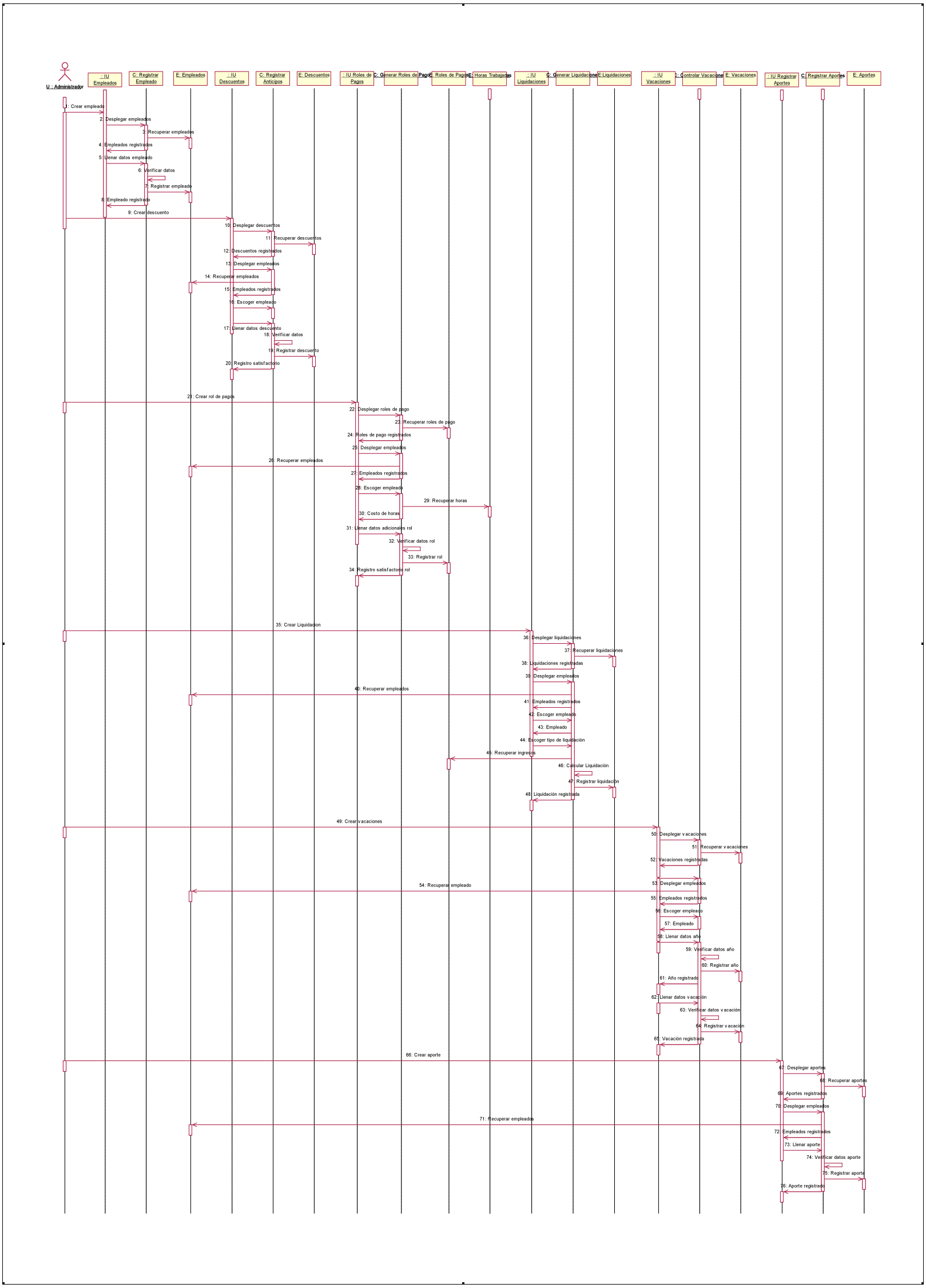


3.3.8 Realización de caso de uso administrar RRHH CU#06



PROYECTO: SAF Muebles Prieto
MODELO: Diagrama de Colaboración - Diseño
CASO DE USO: Administrar RRHH CU#06
VERSIÓN: 0.1
AUTORES: Luis Prieto y Patricio Prieto

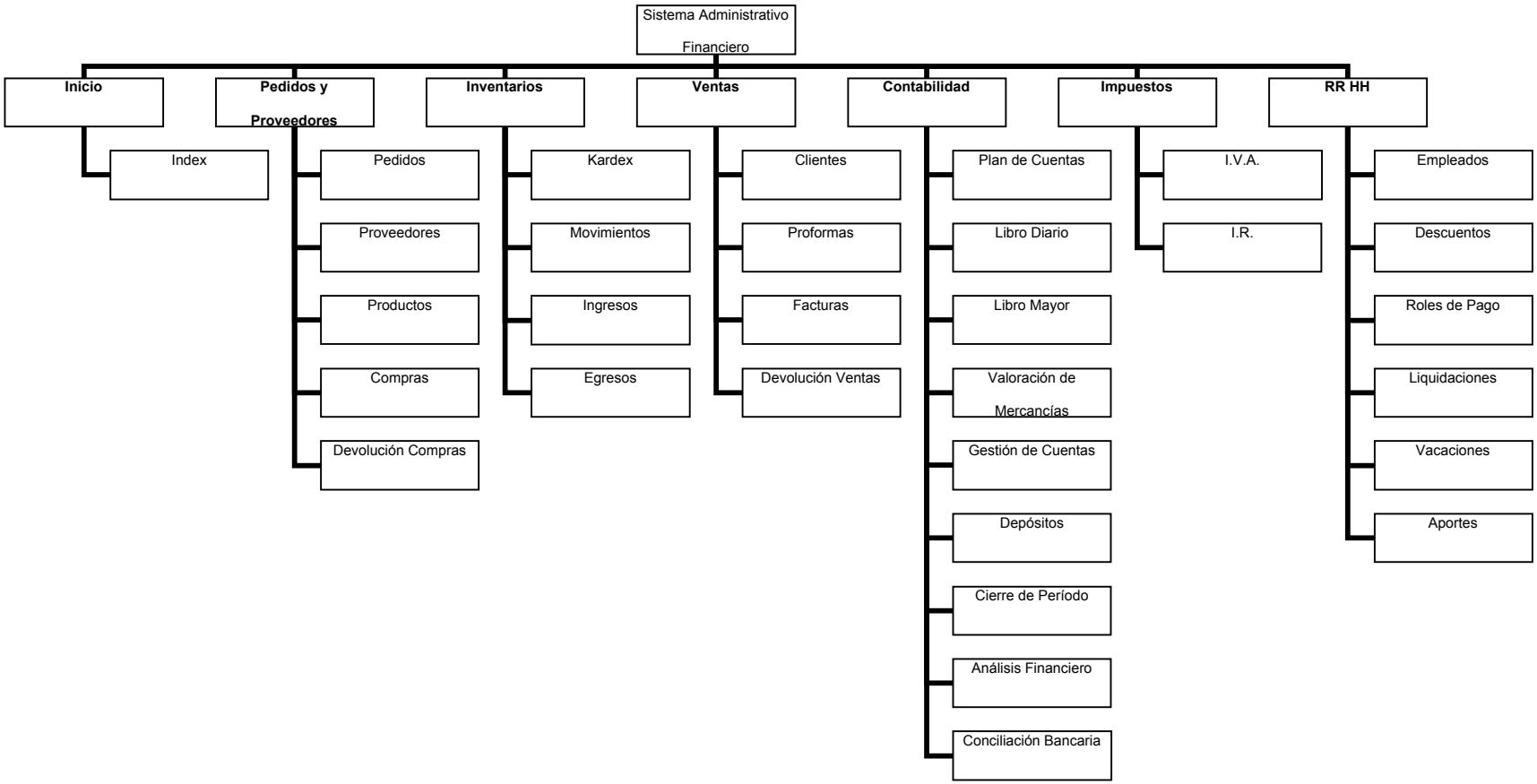
3.3.8.1 Diagrama de secuencia Administrar RRHH



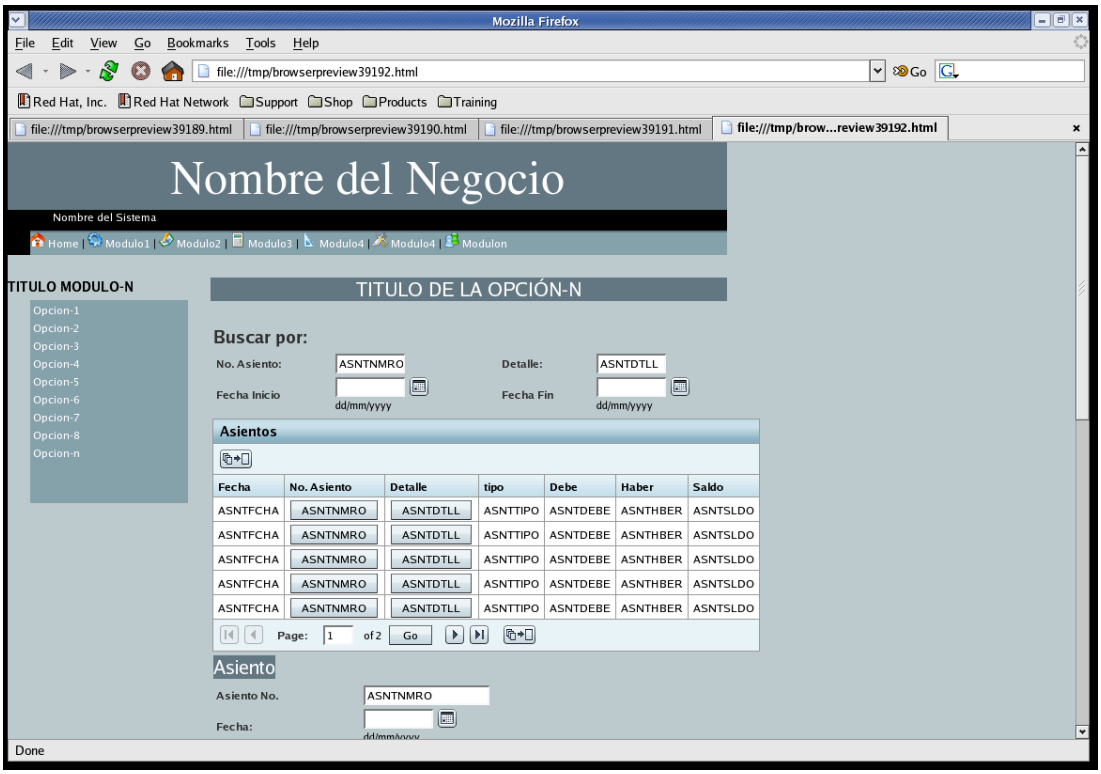
3.3.9 Diseño de Clases

3.3.9.1 Clases UI

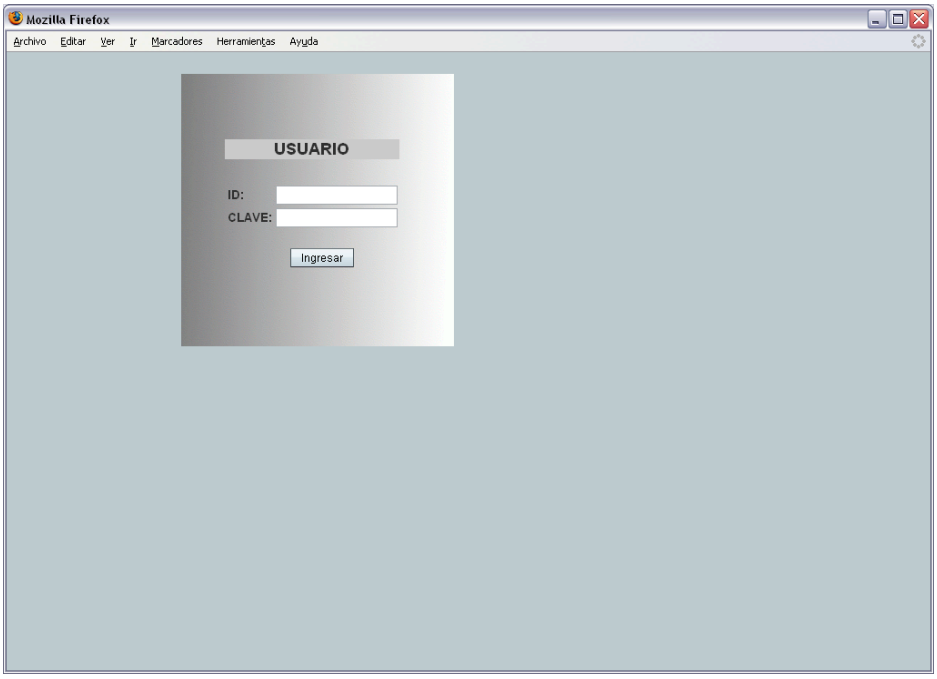
3.3.9.1.1 Modelo de Navegación



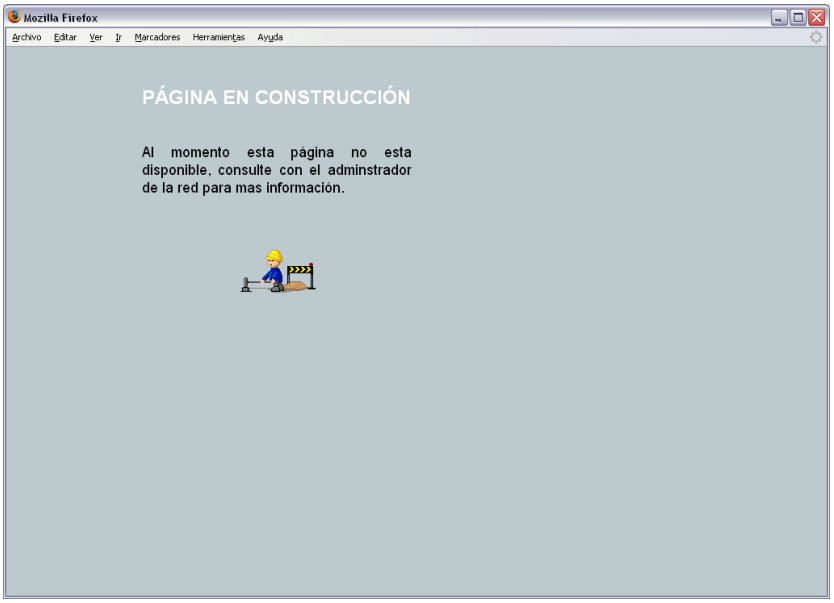
3.3.9.1.2 Interfaz de Menú



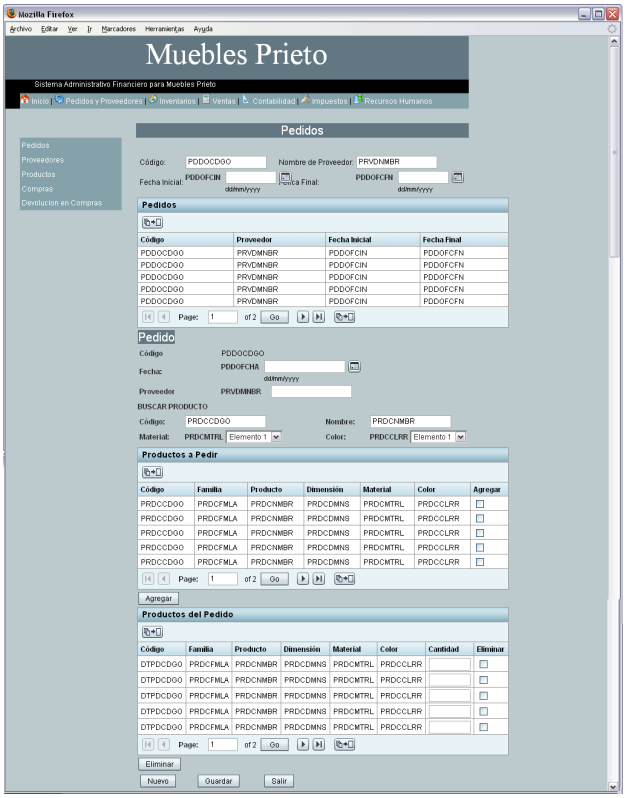
3.3.9.1.3 Interfaz de Seguridad



3.3.9.1.4 Interfaz en Construcción



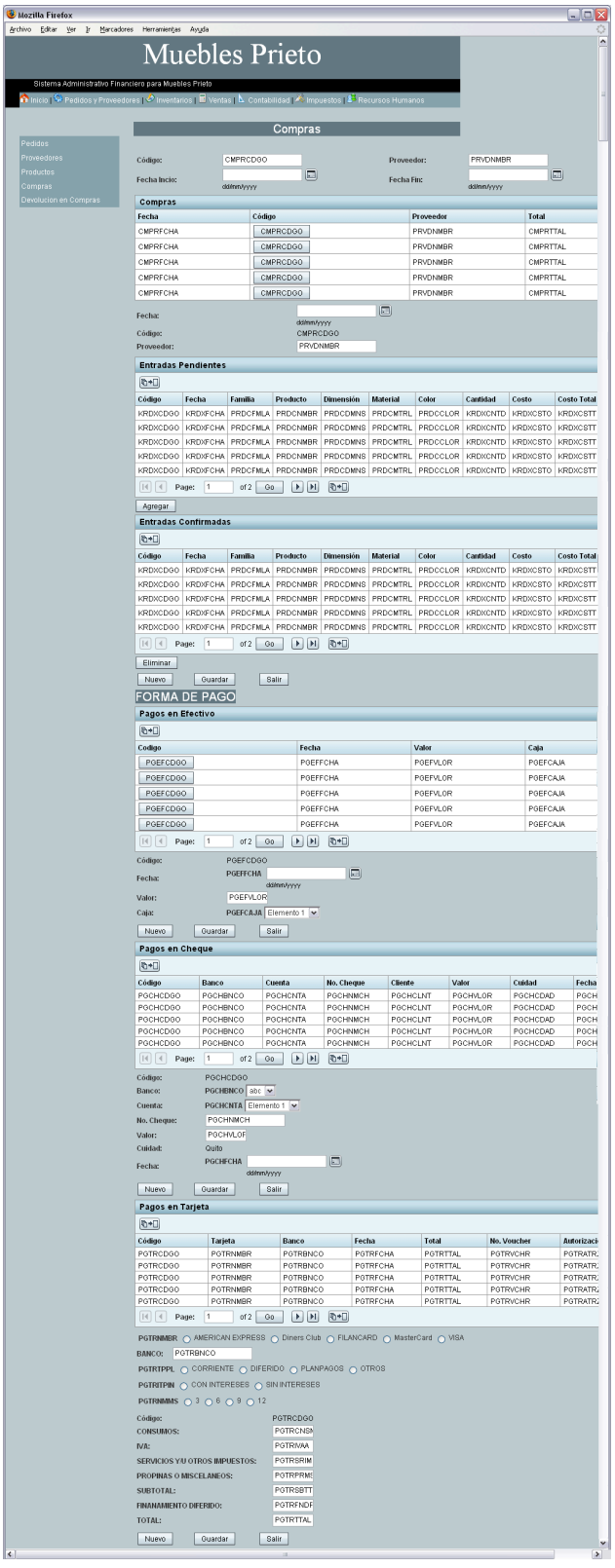
3.3.9.1.5 UI # 01 Caso de Uso #01: Pedidos



3.3.9.1.6 UI # 02 Caso de Uso #01: Proveedores

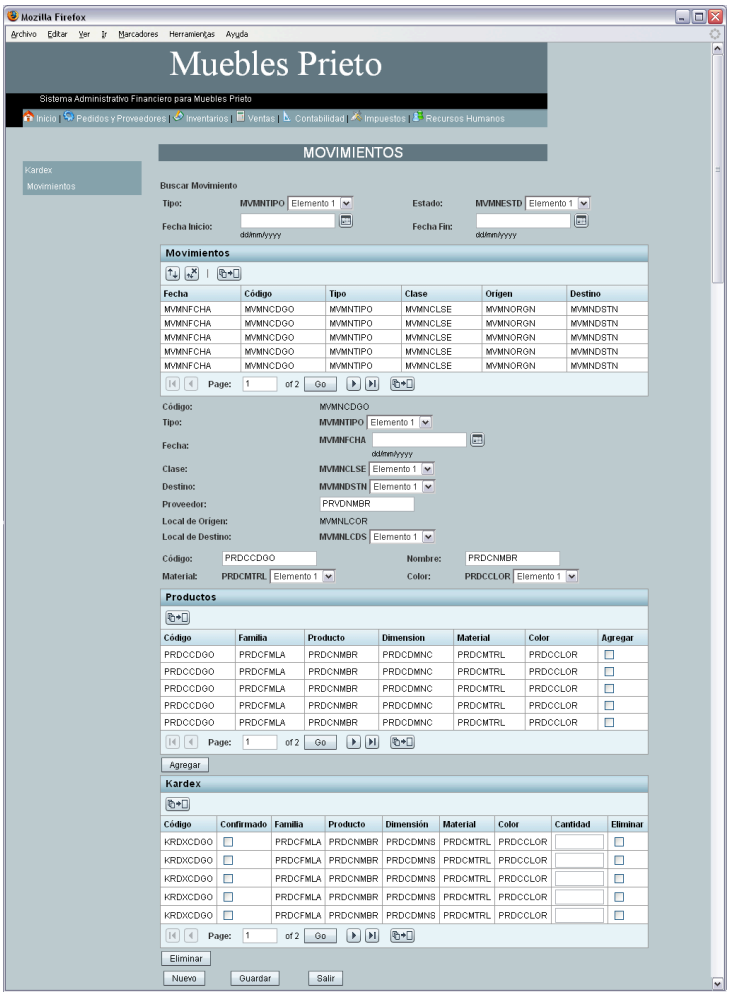
3.3.9.1.7 UI # 03 Caso de Uso #01: Productos

3.3.9.1.8 UI # 04 Caso de Uso #01: Compras

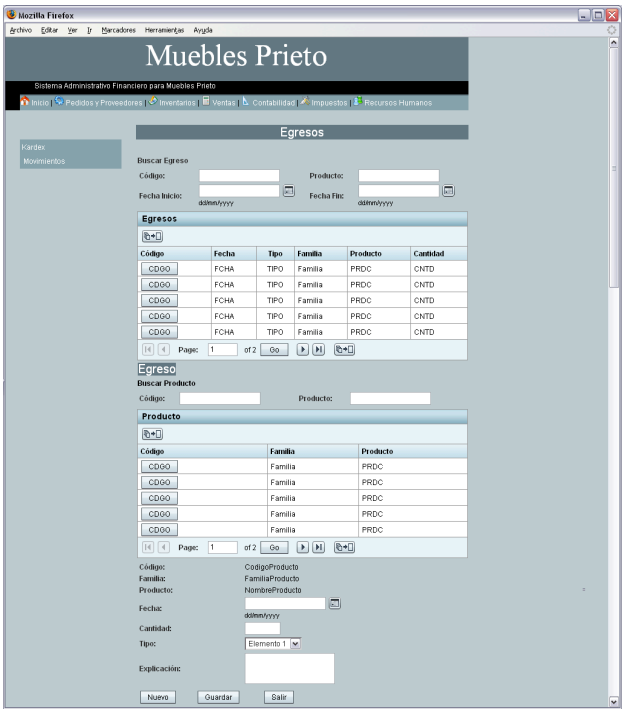


3.3.9.1.9 UI # 05 Caso de Uso #02: Kardex

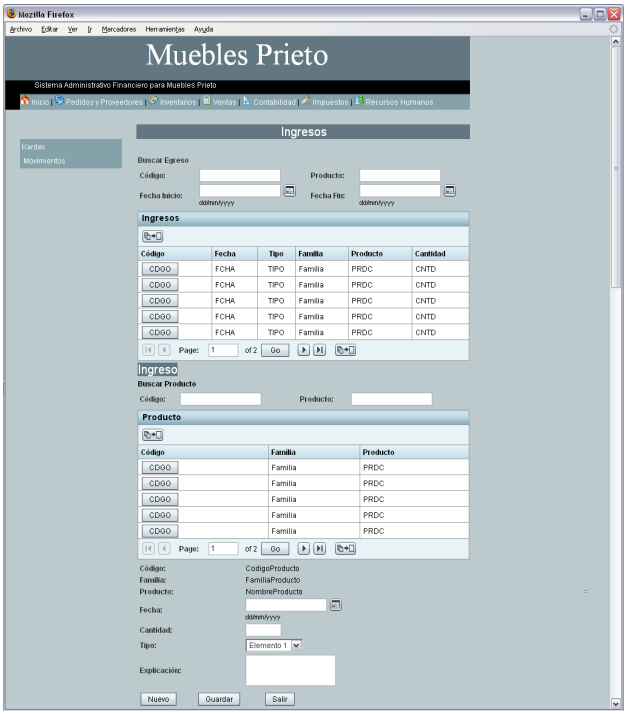
3.3.9.1.10 UI # 06 Caso de Uso #02: Movimientos



3.3.9.1.11 UI # 07 Caso de Uso #02: Egresos

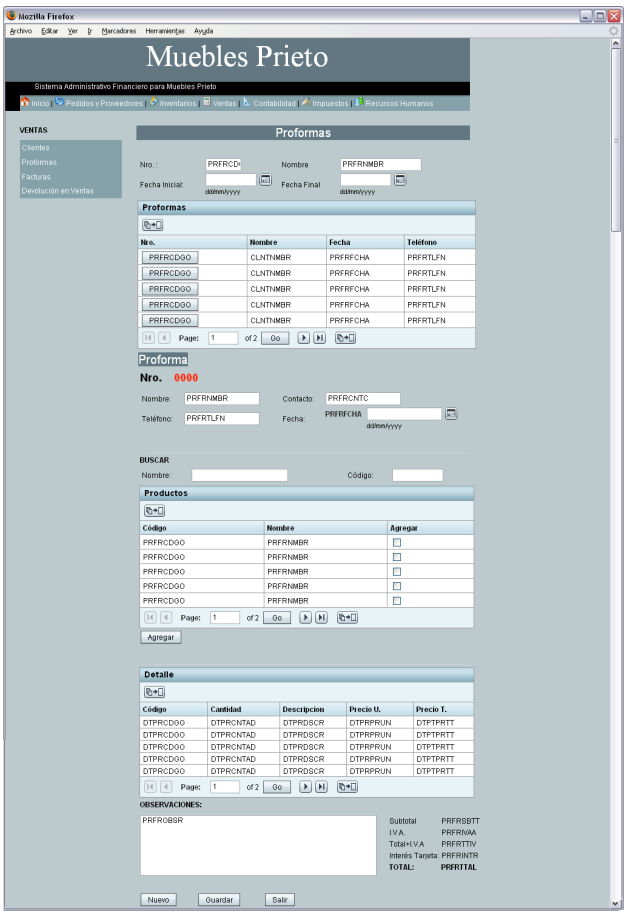


3.3.9.1.12 UI # 08 Caso de Uso #02: Ingresos

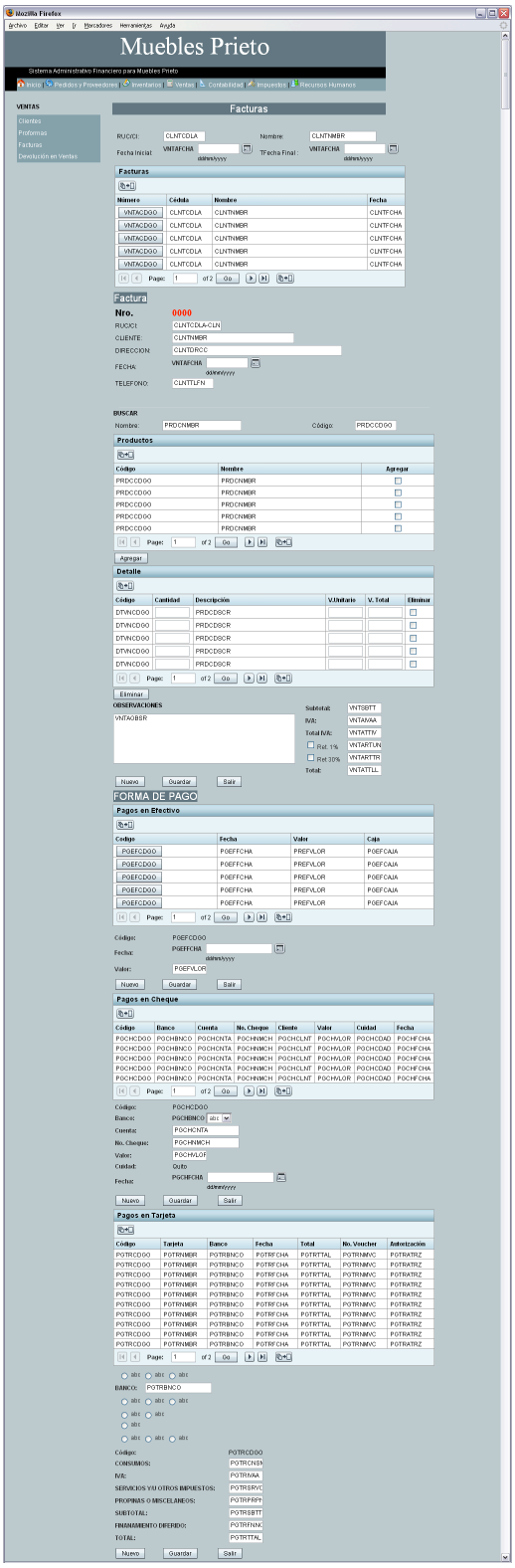


3.3.9.1.13 UI # 09 Caso de Uso #03: Clientes

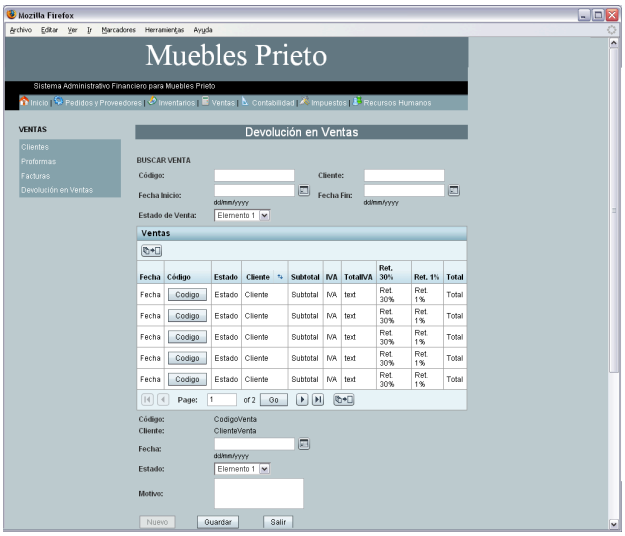
3.3.9.1.14 UI # 10 Caso de Uso #03: Proformas



3.3.9.1.15 UI # 11 Caso de Uso #03: Facturas

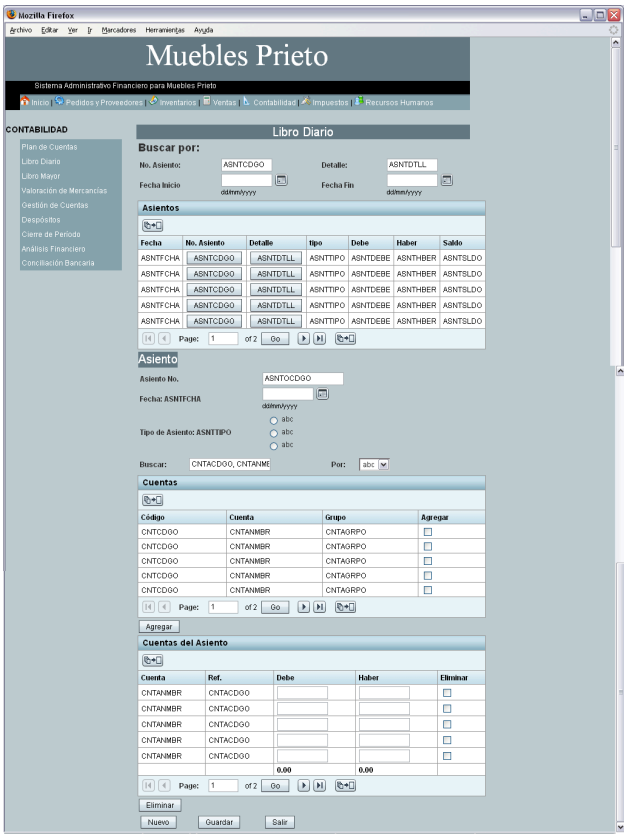


3.3.9.1.16 UI # 12 Caso de Uso #03: Devolución en Ventas

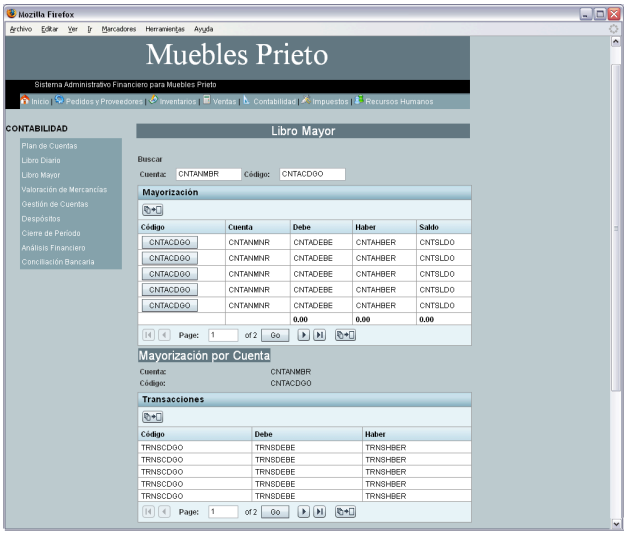


3.3.9.1.17 UI # 12 Caso de Uso #04: Plan de Cuentas

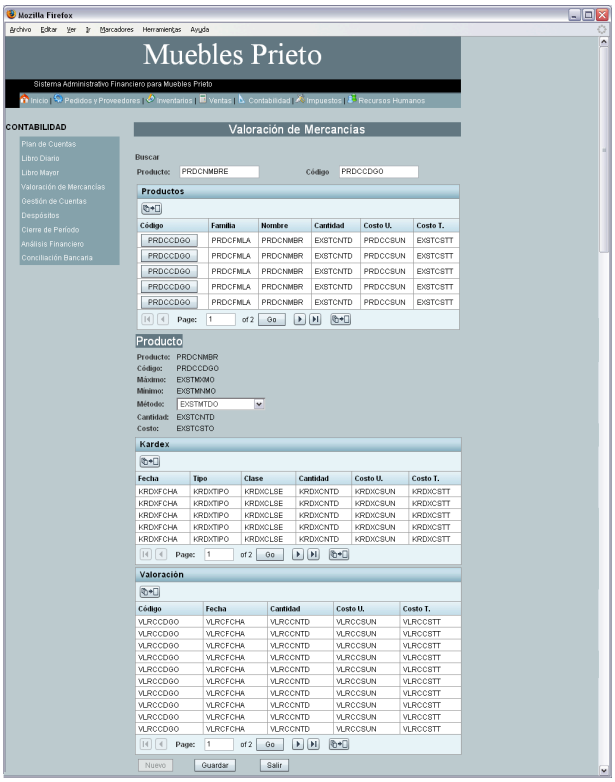
3.3.9.1.18 UI # 13 Caso de Uso #04: Libro Diario



3.3.9.1.19 UI # 15 Caso de Uso #04: Libro Mayor

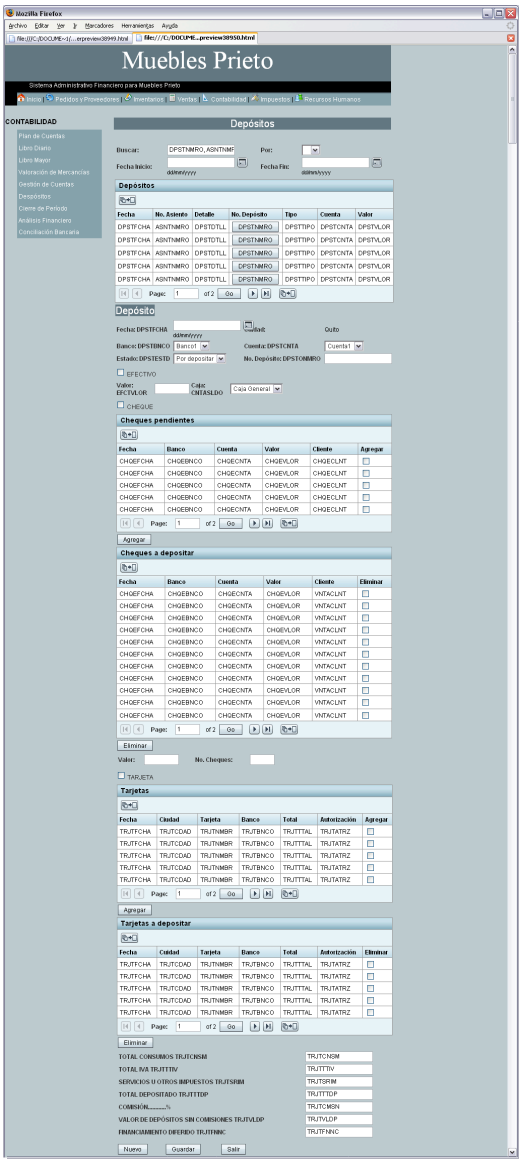


3.3.9.1.20 UI # 16 Caso de Uso #04: Valoración de Mercancías

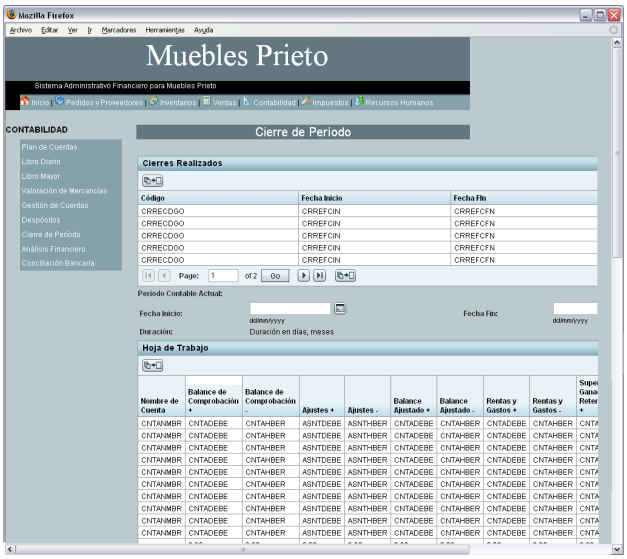


3.3.9.1.21 UI # 17 Caso de Uso #04: Gestión de Cuentas

3.3.9.1.22 UI # 18 Caso de Uso #04: Depósitos



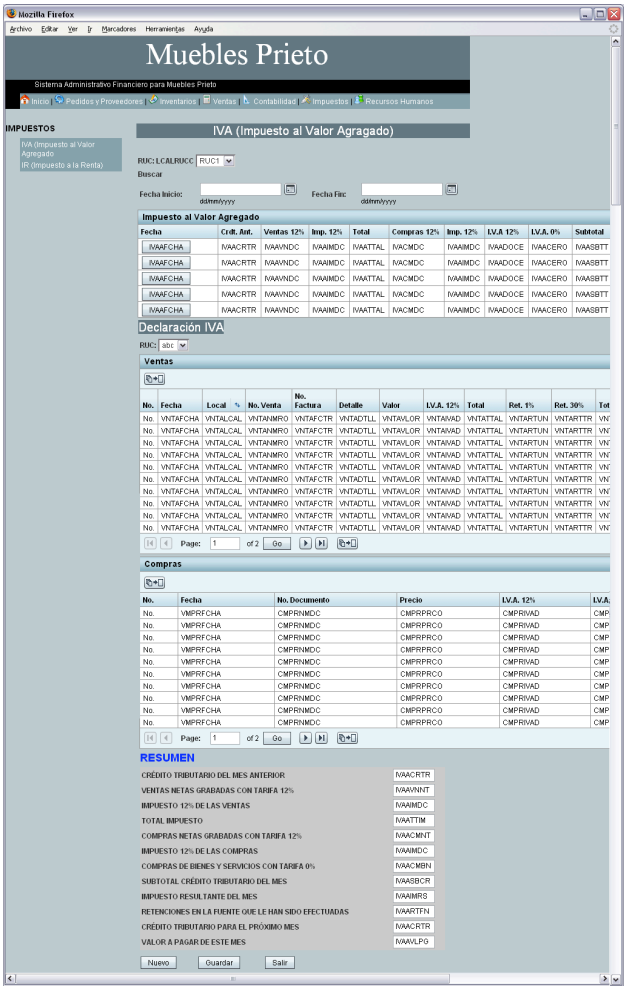
3.3.9.1.23 UI # 19 Caso de Uso #04: Cierre del Periodo



3.3.9.1.24 UI # 20 Caso de Uso #04: Análisis Financiero

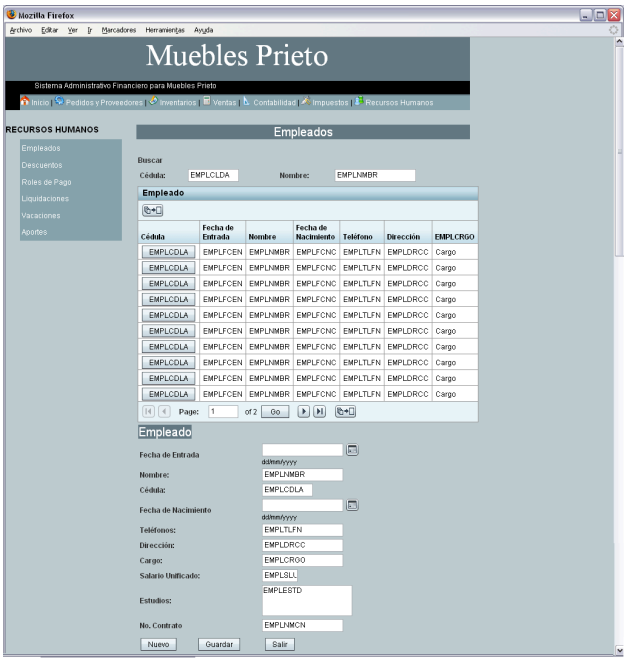
3.3.9.1.25 UI # 21 Caso de Uso #04: Conciliación Bancaria

3.3.9.1.26 UI # 22 Caso de Uso #05: I.V.A.



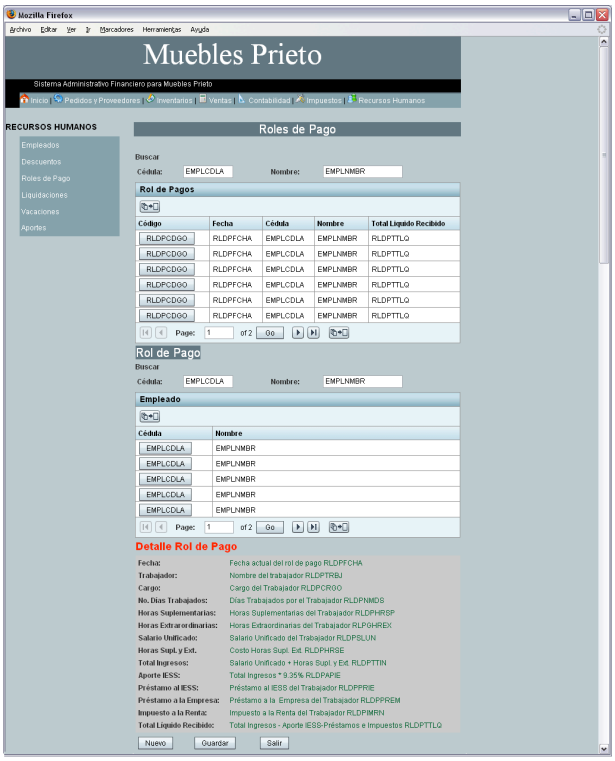
3.3.9.1.27 UI # 23 Caso de Uso #05: I.R.

3.3.9.1.28 UI # 24Caso de Uso #06: Empleados

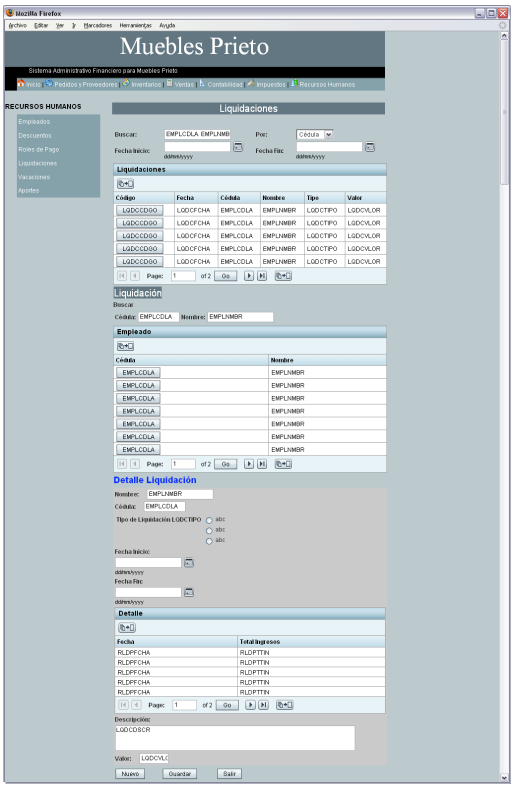


3.3.9.1.29 UI # 25Caso de Uso #06: Descuentos

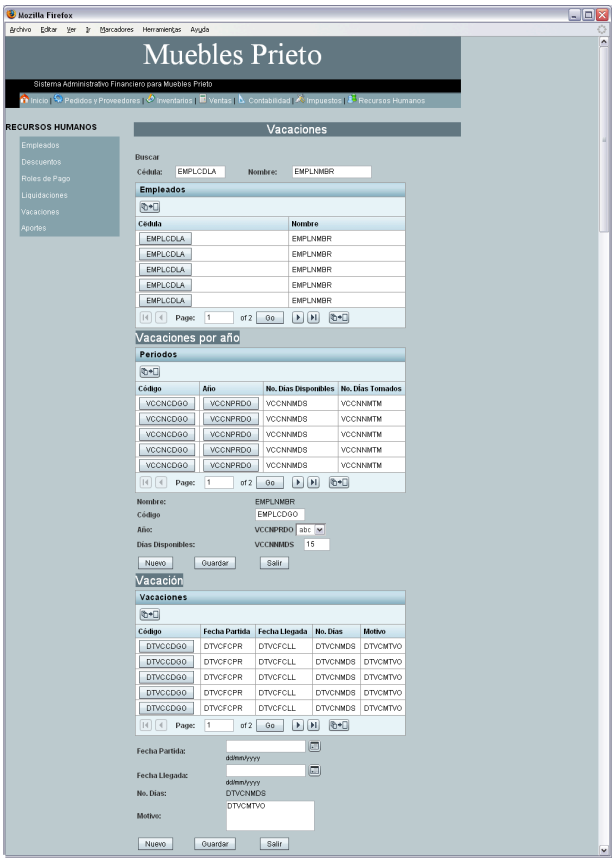
3.3.9.1.30 UI # 26Caso de Uso #06: Roles de Pago



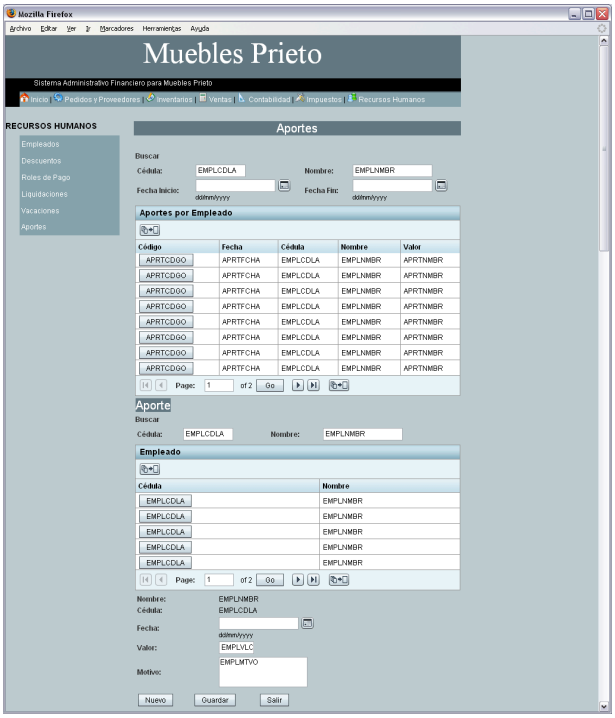
3.3.9.1.31 UI # 27Caso de Uso #06: Liquidaciones



3.3.9.1.32 UI # 28Caso de Uso #06: Vacaciones



3.3.9.1.33 UI # 29Caso de Uso #06: Aportes



3.3.10 Clases de Entidad

Transformación del Modelo de Diseño de Clases de Entidad en Entidad relación.

Reglas:

- Cada clase se convierte en entidad.
- Cada atributo se convierte en atributo de una entidad
- Cada método se convierte en stored procedure o una clase de control.

CLASE: Roles de pago	**TABLA: RLDP**
codigo	RLDPCDGO
fecha	RLDPFCHA
diasTrabajados	RLDPDSTR
horasSuplementarias	RLDPHRSP
horasExtraordinarias	RLDPEXTR
cargo	RLDPCRGO
salarioUnificado	RLDPSLUN
HorasSupYExt	RLDPHRSE
TotalIngresos	RLDPTTIN
AporteIESS	RLDPAPIS
PrestamoIESS	RLDPPRIS
prestamoEmpresa	RLDPPREM
totalLiquidoRecibido	RLDPTTLQ
transporte	RLDPTRNS
comida	RLDPCMDA
extras	RLDPEXTR

CLASE: Aportes	**TABLA: APRT**
codigo	APRTCDGO
fecha	APRTFCHA
valor	APRTVLOR

descripcion	APRTDSCR

CLASE: Empleados	**TABLA: EMPL**
cedula	EMPLCDLA
nombre	EMPLNMBR
direccion	EMPLDRCC
telefono	EMPLTLFN
fechaDeIngreso	EMPLFCIN
fechaDeSalida	EMPLFCSL
fechaDeNacimiento	EMPLFCNC
cargo	EMPLCRGO
salario	EMPLSLRR

CLASE: Vacaciones	**TABLA: VCCN**
codigo	VCCNCDGO
periodo	VCCNPRDO
diasTomados	VCCNDSTM
diasDisponibles	VCCNDSDS

CLASE:DetalleVacaciones	**TABLA: DTVC**
codigo	DTVCCDGO
fechaInicio	DTVCFCIN
fechaFin	DTVCFCFN
noDias	DTVCNODS
motivo	DTVCMTVO

CLASE: Descuentos	**TABLA: DSCN**

codigo	DSCNCDGO
fecha	DSCNFCHA
valor	DSCNVLOR
descripción	DSCNDSCR

CLASE: HorasTrabajadas	**TABLA: HRTB**
codigo	HRTBCDGO
fecha	HRTBFCHA
horaInicio	HRTBHRIN
horaFin	HRTBHRFN
tipo	HRTBTIPO

CLASE: Negocio	**TABLA: NGCO**
codigo	NGCOCDGO
nombre	NGCONMBR
mision	NGCOMSNN
vision	NGCOVSNN
fechaInicioCont	NGCOFCIN

CLASE: Local	**TABLA: LCAL**
nombre	LCALCDGO
direccion	LCALDRCC
telefono	LCALTLFN
ruc	LCALRUCC
tipo	LCALTIPO

CLASE: Cliente	**TABLA: CLNT**

RUC	CLNTRUCC
nombre	CLNTNMBR
contacto	CLNTCNTC
direccion	CLNTDRCC
telefono	CLNTTLFN
email	CLNTEMLL

CLASE: Venta	**TABLA: VNTA**
codigo	VNTACDGO
noComprobante	VNTANOCM
fecha	VNTAFCHA
estado	VNTAESTD
totalArticulos	VNTATTAR
subtotalArticulos	VNTASBAR
iva12	VNTAIVAD
iva30	VNTAIVAT
retencion1	VNTARTNC
total	VNTATTAL
abonoTotal	VNTAABTT
saldoTotal	VNTASLTT

CLASE: DetalleVenta	**TABLA: DTVN**
codigo	DTVNCDGO
cantidad	DTVNCNTD
precioUnitario	DTVNPRUN
precioTotal	DTVNPRTT

CLASE: Abono	**TABLA: ABNO**
codigo	ABNOCDGO
valor	ABNOVLOR
formaDePago	ABNOFRPG
fecha	ABNOFCHA
observaciones	ABNOOBSR

CLASE: PagoTarjeta	**TABLA: PGTR**
codigo	PGTRCDGO
numeroTarjeta	PGTRNMTR
nombreTarjeta	PGTRNMTR
banco	PGTRBNCO
fecha	PGTRFCHA
cuidad	PGTRCDAD
tipo	PGTRTIPO
totalConsumos	PGTRTTCN
totalIVA	PGTRTTIV
servicios	PGTRSRVC
propinas	PGTRPRPN
subtotal	PGTRSBTT
financiamiento	PGTRFNNC
total	PGTRTTAL
noMeses	PGTRNOMS
interes	PGTRINTR
corriente	PGTRCRRN
autorización	PGTRATRZ
estado	PGTRESTD

CLASE: PagoCheque	**TABLA: PGCH**
codigo	PGCHCDGO
cheque	PGCHCHQE
cuenta	PGCHCNTA
banco	PGCHBNCO
fecha	PGCHFCHA
valor	PGCHVLOR
estado	PGCHESTD

CLASE: ProForma	**TABLA: PRFR**
codigo	PRFRCDGO
nombre	PRFRNMBR
telefono	PRFRTLFN
fecha	PRFRFCHA
observaciones	PRFROBSR
subtotal	PRFRSBTT
totalIva	PRFRTTIV
total	PRFRTTAL
interes	PRFRINTR

CLASE: DetalleProForma	**TABLA: DTPR**
codigo	DTPRCDGO
cantidad	DTPRCNTD
valorUnitario	DTPRVLUN
valorTotal	DTPRVLTT

CLASE: Compra	**TABLA: CMPR**
codigo	CMPRCDGO
noDocumento	CMPRNODC
detalle	CMPRDTLL
subtotal	CMPRSBTT
iva	CMPRIVAA
total	CMPRTTAL
iva0	CMPRIVAC
totalIva0	CMPRTTIV

CLASE: Kardex	**TABLA: KRDX**
codigo	KRDXCDGO
fecha	KRDXFCHA
cantidad	KRDXCNTD
valorUnitario	KRDXVLUN
valorTotal	KRDXVLTT
explicación	KRDXEXPL
tipoMovimiento	KRDXTIMV
claseMovimiento	KRDXCLMV

CLASE: Existencias	**TABLA: EXST**
codigo	EXST CDGO
maximo	EXSTMXMO
minimo	EXSTMNMO
cantidad	EXSTCNTD
costo	EXSTCSTO
metodo	EXSTMTDO

CLASE: Valoración	**TABLA: VLRC**
codigo	VLRCCDGO
fecha	VLRCFCHA
cantidad	VLRCCNTD
costoUnitario	VLRCCSUN
costoTotal	VLRCCSTT

CLASE: Producto	**TABLA: PRDC**
codigo	PRDCCDGO
nombre	PRDCNMBR
descripción	PRDCDSCR
costo	PRDCCSTO
precio	PRDCPRCO

CLASE: FamiliaDeProductos	**TABLA: FMPR**
codigo	FMPRCDGO
familia	FMPRFMLA

CLASE: Proveedor	**TABLA: PRVD**
ruc	PRVDRUCC
Nombre	PRVDNMBR
Direccion	PRVDDRCC
telefono	PRVDTLFN
celular	PRVDCLLR
Email	PRVDEMLL

CLASE: Pedido	**TABLA: PDDO**

codigo	PDDOCDGO
numero	PDDONMRO
fechaInicial	PDDOFCIN
fechaFinal	PDDOFCFN
estado	PDDOESTD

CLASE: DetallePedido	**TABLA: DTPD**
codigo	DTPDCDGO
cantidad	DTPDCNTD
valorUnitario	DTPDVLUN
valorTotal	DTPDVLTT

CLASE: Cuenta	**TABLA: CNTA**
codigo	CNTACDGO
nombre	CNTANMBR
grupo	CNTAGRPO
tipo	CNTATIPO
debe	CNTADEBE
haber	CNTAHBER
saldo	CNTASLDO

CLASE: Transaccion	**TABLA: TRNS**
codigo	TRNSCDGO
debe	TRNSDEBE
haber	TRNSHBER

CLASE: Asiento	**TABLA: ASNT**

noAsiento	ASNTCDGO
fecha	ASNTFCHA
totalDebe	ASNTTTDB
totalHaber	ASNTTTHB
explicación	ASNTEXPL
tipo	ASNTTIPO

CLASE: CuentaCierre	**TABLA: CNCR**
codigo	CNCRCDGO
nombre	CNCRNMBR
grupo	CNCRGRPO
tipo	CNCRTIPO
debe	CNCRDEBE
haber	CNCRHBER

CLASE: Cierre	**TABLA: CRRE**
codigo	CRRECDGO
fechaInicio	CRREFCIN
fechaFin	CRREFCFN
duracion	CRREDRCN

CLASE: Balance	**TABLA: BLNC**
codigo	BLNCCDGO
nombre	BLNCNMBR
descripción	BLNCDSCR
fecha	BLNCFCHA

3.3.11 Construcción y Pruebas

3.3.11.1 Selección de las Herramientas

- Back End

Parámetros	***Prioridad%***	***PostGreSQL 8***	***Oracle XE 10g***	***MySQL 5.0***
Escalabilidad y Portabilidad	10	7	8	8
Documentación y Soporte Técnico	15	6	7	6
Seguridad de accesos	25	7	8	7
Replicación	15	8	8	7
Migración de Datos	10	8	7	7
Confiabilidad	10	8	8	8
Estabilidad en Fallas	15	7	9	8
	100%	51/70	**55/70**	51/70

- Front End

Parámetros	***Prioridad%***	***ASP.Net Web Matrix***	***Java Creator 2***	***Aptana***
Facilidad de Uso	25	8	9	5
Extensibilidad	15	7	7	6
Conectividad con base de Datos	25	7	9	6
Multiplataforma	20	6	8	7

Requerimientos del sistema	15	8	6	8
	100%	36/50	**39/50**	32/50

Las herramientas con mayor puntaje son la que se ajustan a las necesidades del desarrollo, como se muestra en la tabla de back end y front end.

3.3.11.2 Manual de Programación

Manual de programación:

Para el software de java usar los siguientes sufijos:

Tipo de archivo	Sufijo
Fuente java	.java
Código de bits java	.class

Todos los archivos fuente deben comenzar con la siguiente cabecera:

```
/*
 * @archivo:    Ejemplo.java
 *
 * @versión:    0.83 27 Abril 2006
 *
 * @autor:      Nombre Apellido
 *
 * @fecha:      4 Agosto 2006
 *
 * @copyright: Muebles Prieto
 *
 *
 * Descripción de la clase.
 *
 */
```

Declaraciones de clases e interfaces:

REGLA	EJEMPLO
Usar /**…*/ como comentario de documentación de interfaz/clase, describir su	/** *Codigo explicativo */

naturaleza, propósito, pre-condiciones, efectos, notas de algoritmos, instrucciones de uso, recordatorios entre otros.	

Usar // para explicar código no obvio, cada línea debe contener una sola sentencia	`int index = -1; // -1 sirve como bandera que significa que el index no es válido.`
	
	`argv++;  // Correcto`
	
	`argc--;  // Correcto`
	
	`argv++; argc--;  // EVITAR!`

Comentario de implementación `/*…*/` Se utiliza para describir detalles algorítmicos, notas y documentación relacionada que mide más de una pocas líneas de código.	/* * Estrategia: * 1. Encontrar el nodo * 2. Clonarlo * 3. pedir al insertador agregar el clon * 4. Si se logra, borrar el nodo */

Tabulaciones:

CONDICIÓN	REGLA
Tabulación	Unidad definida por 8 espacios como un TAB
Largo de líneas	Cada línea no debe pasar los 80 caracteres
Rompimiento de líneas	Después de una coma, antes de un

	operador, fuera de un paréntesis.

Declaraciones:

REGLA	EJEMPLO

Hacer una declaración por línea	`int level; // nivel de identación`
	`int size;  // tamaño de la tabla`

No poner diferentes tipos en la misma línea	`int foo,  fooarray[]; //MAL!`

Inicializar las variables al declarar	int level = 0; // nivel de identación
	int size = 0; // tamaño de la tabla

Declarar las variables al inicio de los bloques	`void myMethod() {`
	`    int int1 = 0;         //`
	`beginning of method block`
	
	`    if (condition) {`
	`        int int2 = 0;     //`
	`beginning of "if" block`
	`        ...`
	`    }`
	`}`

Para el for si esta permitido declarar dentro del bloque	`for (int i = 0; i < maxLoops;` `i++) { ... }`

Declaraciones de clases e interfaces:

REGLA	EJEMPLO
No hay espacio entre el nombre del método y el paréntesis	`int emptyMethod()`

La llave { aparece en la misma línea que la declaración. La llave } tiene una línea por si sola y concuerda en tabulación con el principio de la sentencia.	`class Sample extends Object {`
	`  int ivar1;`
	`  int ivar2;`
	`}`

Cuando es una sentencia vacía la llave } va inmediatamente después de {	`int emptyMethod() {}`

Formas de sentencias:

REGLA FORMA

Las sentencias IF, IF-ELSE, IF ELSE-IF ELSE, deben tener la siguiente forma:

```
if (condition) {
    statements;
}

if (condition) {
    statements;
} else {
    statements;
}

if (condition) {
    statements;
} else if (condition) {
    statements;
} else {
    statements;
}
```

Las sentencias de FOR tienen que tener la siguiente forma:

```
for (initialization; condition; update) {
    statements;
}
```

Una sentencia de FOR vacía debe tener la siguiente forma:

```
for (initialization; condition; update);
```

Las sentencias de WHILE deben tener la siguiente forma:

```
while (condition) {
    statements;
}
```

Una sentencia de WHILE vacía debe tener la siguiente forma:

```
while (condition);
```

Una sentencia de DO-WHILE tiene la siguiente forma:

```
do {
    statements;
} while (condition);
```

Una sentencia de SWITCH tiene la siguiente forma:

```
switch (condition) {
case ABC:
    statements;
    /* falls through */

case DEF:
    statements;
    break;

case XYZ:
    statements;
    break;

default:
    statements;
    break;
}
```

Una sentencia TRY-CATCH debe tener la siguiente forma:

```
try {
    statements;
} catch (ExceptionClass e) {
    statements;
}
```

Una sentencia TRY-CATCH con FINALLY:

```
try {
    statements;
} catch (ExceptionClass e) {
    statements;
} finally {
    statements;
}
```

Uso de espacios y líneas en blanco:

REGLA	EJEMPLO

Dos líneas en blanco de separación entre secciones de código fuente y entre	`public class Ejemplo {`
	
	`metodoEjemplo1() {}`

definiciones de clases e interfaces	`metodoEjemplo2() {}` `}`

Una sola línea en blanco entre métodos.
Una sola línea entre variables y el primer método,
	`public class Ejemplo {`
	
	`int variable1 = 0;`
	`int variable2 = 0;`
	
	`metodoEjemplo1() {}`
	
	`metodoEjemplo2() {}`
	`}`

Una sola línea en blanco antes de comentarios en bloque o comentarios únicos	`// Cometario único`
	
	`/*…*/ Comentario en bloque`

Un espacio en blanco debe seguir a cualquier palabra reservada	`while (true) {`
	`...`
	`}`

Un espacio en blanco debe aparecer después de comas en una lista de argumentos o en un FOR de la siguiente manera:	`metodoEjemplo1(arg1, arg2, arg3)` `for (expr1; expr2; expr3)`

Todos los operadores binarios están separados de sus operadores por un espacio, excepto los unarios que no deben separarse	`a += c + d;`
	`a = (a + b) / (c * d);`
	
	`while (d++ = s++) {`
	`n++;`
	`}`
	`printSize("size is " + foo);`

La conversión CAST debe estar seguida por un espacio en blanco	`myMethod((byte) aNum, (Object) x);` `myMethod((int) (cp + 5), ((int) (i + 3)) + 1);`

Convenciones para nombres:

IDENTIFICADOR	EJEMPLOS
Paquetes minúsculas	`com.sun.eng com.apple.quicktime.v2` `edu.cmu.cs.bovik.cheese`
Clases LetraCapitalConPalabrasConLetraCapital	`class Raster;` `class ImageSprite;`

Interfaces LetraCapitalConPalabrasConLetraCapital	`interface RasterDelegate;` `interface Storing;`

Métodos
 	
primeraPMinusculasPInternasConLetraCapital
	`run();`
	`runFast();`
	`getBackground();`

Variables
 	
primeraPMinusculasPInternasConLetraCapital
	`int i;`
	`char c;`
	`float myWidth;`

Constantes
 	
MAYUSCULAS_CON_SUBGUIONES
	`static final int MIN_WIDTH = 4;`
	
	`static final int MAX_WIDTH = 999;`
	
	`static final int GET_THE_CPU = 1;`

Ejemplo:

```
/*
 * Archivo:     Blah.java
 *
 * Versión:     1.82 18 Marzo 1999
 *
 * Autor:       Nombre Apellido
 *
 * Fecha:       4 Agosto 2006
 *
 * Copyright:  1994-1999 Sun Microsystems, Inc
 * 901 San Antonio Road, Palo Alto, California, 94303, U.S.A.
 * All rights reserved.
 *
 * This software is the confidential and proprietary information of
 * Sun Microsystems, Inc. ("Confidential Information").  You shall
not
 * disclose such Confidential Information and shall use it only in
 * accordance with the terms of the license agreement you entered
into
 * with Sun.
 */




package java.blah;

import java.blah.blahdy.BlahBlah;

/**
 * La descripción de la clase va aquí.
```

```
 *
 * @version    1.82 18 Marzo 1999
 * @author     Patricio Prieto
 */
public class Blah extends SomeClass {
    /* Una implementación de clase va aquí. */

    /** classVar1 comentario de documetación */
    public static int classVar1;

    /**
     * classVar2 comentario de documentación que ocurre
     * en más de una línea de logitud
     */
    private static Object classVar2;

    /** instanceVar1 comentario de documentación */
    public Object instanceVar1;

    /** instanceVar2 comentario de documentación */
    protected int instanceVar2;

    /** instanceVar3 comentario de documentación */
    private Object[] instanceVar3;

    /**
     * ...comentario de documentación del constructor Blah...
     */
    public Blah() {
        // ...La implementación va aquí...
    }

    /**
     * ...comentario de documentación del método doSomething...
     */
    public void doSomething() {
        // ...La implementación va aquí...
    }

    /**
     * ...comentario de documetación del método doSomethingElse...
     * @param someParam descripción del parámetro someParam
     */
    public void doSomethingElse(Object someParam) {
        // ...La implementación va aquí...
    }
}
```

A. Apéndice

A.1. Glosario

COTS Commercial off-the-shel. Es un producto desarrollado bajo estándares y especificaciones comerciales y esta listo para usarse.

US DoD United States Department of Defence. Departamento de defensa de los estados unidos.

ANSI American National Standards Institute. Instituto Americano Nacional de Estándares.

BSI British Standard Institution. Institución Británica de Estándares.

NATO The North Atlantic Treaty Organisation. La Organización del Tratado del Norte

IEEE Instutute of Electrical and Electronics Engineers. Instituto de Ingenieros Eléctricos y Electrónicos.

LAN (Local Area Network) Red de Área Local.

WAN (Wide Area Network) Red de Área Extendida.

CU Caso de uso.

A.2. Referencias

[1] Ian Sommerville, Software Engineering 7, Seventh Edition. United States: Addison Wesley, 2004, pp. 25-30.

[2] Ian Sommerville, Software Engineering 7, , Seventh Edition. United States: Addison Wesley, 2004, pp. 99-105.

[3] Ian Sommerville, Software Engineering 7, , Seventh Edition. United States: Addison Wesley, 2004, pp. 106-112.

[4] Ian Sommerville, Software Engineering 7, , Seventh Edition. United States: Addison Wesley, 2004, pp. 515-536.

[5] Ivar Jacobson, Grady Booch, James Rumbaught, El Proceso Unificado de Desarrollo de Software, Madrid: Addison Wesley, 2000, pp. 83-102.

[6] Ivar Jacobson, Grady Booch, James Rumbaught, El Proceso Unificado de Desarrollo de Software, Madrid: Addison Wesley, 2000, cap 6.

[7] Ivar Jacobson, Grady Booch, James Rumbaught, El Proceso Unificado de Desarrollo de Software, Madrid: Addison Wesley, 2000, cap 8.

[8] Ivar Jacobson, Grady Booch, James Rumbaught, El Proceso Unificado de Desarrollo de Software, Madrid: Addison Wesley, 2000, cap 9.

[9] Ivar Jacobson, Grady Booch, James Rumbaught, El Proceso Unificado de Desarrollo de Software, Madrid: Addison Wesley, 2000, cap 10, pp. 256-259.

[10] Ivar Jacobson, Grady Booch, James Rumbaught, El Proceso Unificado de Desarrollo de Software, Madrid: Addison Wesley, 2000, cap 11.

[11] Ivar Jacobson, Grady Booch, James Rumbaught, El Proceso Unificado de Desarrollo de Software, Madrid: Addison Wesley, 2000, cap 12.

[12] Ivar Jacobson, Grady Booch, James Rumbaught, El Lenguaje Unificado de Modelado, Madrid: Addison Wesley, 2000, pp. 13-301.

A.3. Bibliografía

JACOBSON, I. *El Proceso e Unificado de Desarrollo de Software.* Madrid: Addison Wesley, 2000. 464p.

RUMBAUGH, James. *El lenguaje Unificado de Modelado: manual de referencia*. Madrid: Addison Wesley, 1999.

BOOCH, G.. *El Lenguaje Unificado de Modelado*. Madrid: Addison Wesley, 1999.

SOMMERVILLE, I. *Software Engineering*. Seventh Edition. United States: Addison Wesley,2004.

FRAUDE, E. *Ingeniería del Software: una perspectiva orientada a objetos*. México: Alfaomega, 2003.

HUGES, B.; COTTEREL, M. *Software Project Management*. Second Edition. England,1999.

PRESSMAN, R. *Ingeniería del Software un Enfoque Práctico*. Quinta Edición. España: McGraw-Hill, 2002.

BRAVO, M. *Contabilidad General*. Quinta Edición. Ecuador: Nuevodia, 2004.

KATACORA, F. *Sistemas y Procedimientos Contables*. Colombia: McGraw-Hill, 2000.

Referencias Web:

http://www.tfd.com/. Diccionario de Inglés.

http://www.fao.org/docrep/007/y4851s/y4851s0b.htm. Viabilidad de Proyectos.

A.4. Índice de Figuras y Tablas