Benemérita
Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

"Diseño de Aplicaciones Utilizando Patrones de Ingeniería"

Tesis de Licenciatura

Que para obtener el titulo de:
Ingeniero en Ciencias de la Computación

Presenta
Edgar Carmelo García Olivares

Asesor
Dr. Abraham Sánchez López

Puebla, Pue. Otoño 2008

Resumen

En el presente trabajo de tesis se hace una descripción detallada de los patrones de diseño mas usados para la asignación de responsabilidades. Los patrones en el campo del software se pueden observar de una manera análoga a los patrones en otros campos de desarrollo como los patrones de arquitectura y de construcción o en campos de estudio como en la psicología los patrones del comportamiento, así en el campo de software los patrones de diseño son usados para retomar una forma de solucionar un problema utilizado anteriormente y aplicarlo a un nuevo problema que este en desarrollo.

Los patrones GRASP nos ayudan en el descubrimiento y la asignación de las responsabilidades que deben ser cumplidas en cada caso de uso. Estos patrones hacen referencia a cuestiones básicas, comunes y fundamentales en el diseño.

En este trabajo se realiza el diseño de una aplicación de ventas de gas primero llevando a cabo un diagrama de secuencia sin usar los patrones y en un capitulo posterior utilizando estos patrones para a partir de aquí establecer una diferencia entre ambos diagramas y continuar con el diseño de la solución.

Se termina el diseño con los Diagramas de Clases de Diseño para cada caso de uso analizado y estudiando como se establece una correspondencia directa de una tabla en una base de datos con una clase de objetos persistente.

Agradecimientos

Agradezco a todas aquellas personas que creyeron en mí, que me apoyaron brindándome un consejo o una palabra de aliento y tuvieron la paciencia suficiente, haciendo posible este trabajo de titulación. Gracias.

Índice General

1. **Introducción: El Reuso en el Diseño de Aplicaciones**.....3

1.1. Patrones en Distintos Campos de Aplicación.....3
1.2. Niveles de Reuso en Diseño de Sistemas 4
1.3. Los Patrones de Diseño.....5
1.4. Identificación de un Patrón de Diseño.....5

2. **Introducción a los patrones de diseño**.....7

2.1. Descripción de los Patrones.....7
2.2. Los Patrones GRASP.....8
2.2.1. Patrón Experto.....9
2.2.2. Patrón Creador.....10
2.2.3. Patrón Bajo Acoplamiento.....11
2.2.4. Patrón Alta Cohesión.....12
2.2.5. Patrón Controlador.....13
2.2.5.1. Sistemas del Manejo de Mensajes y el Patrón Comando.....16
2.3. Patrones Factoría y Fachada.....17
2.3.1. Patrón Factoría.....17
2.3.2. Patrón Fachada.....18

3. Análisis de Requerimientos del Sistema.....21

3.1. Análisis Preliminar de Casos de Uso (Casos de Uso clásicos).....23
3.2. Análisis del Modelo Conceptual.....28
3.3. Diseño del Modelo Conceptual.....28
3.4. Modelo Conceptual.....29

4. Modelado de los Casos de Uso y Análisis de Una Posible Solución de Manera Intuitiva
.....31

4.1. Introducción.....31
4.2. Casos de Uso en Formato Completo con un Acercamiento clásico al Diseño de Diagramas de Secuencia y Colaboraciones.....31
4.2.1. Caso de uso UC1: Registrar Ventas de Pipas.....32
4.2.2. Caso de uso UC2: Registrar Ventas de Cilindros.....35
4.2.3. Caso de uso UC3: Registrar Corte de Caja.....37
4.2.4. Caso de uso UC4: Alta de Trabajadores en el Sistema.....39
4.2.5. Caso de uso UC5: Baja de Trabajadores en el Sistema.....41
4.2.6. Caso de uso UC6: Modificación de Trabajadores en el Sistema.....44
4.2.7. Caso de uso UC7: Alta de Unidades en el Sistema.....46
4.2.8. Caso de uso UC8: Baja de Unidades en el Sistema.....48
4.2.9. Caso de uso UC9: Modificación de Unidades en el Sistema.....50
4.2.10. Caso de uso UC10: Alta de Clientes en el Sistema.....52
4.2.11. Caso de uso UC11: Baja de Clientes en el Sistema.....54
4.2.12. Caso de uso UC12: Modificación de Clientes en el Sistema.....56
4.2.13. Caso de uso UC13: Obtención de Reporte para Nómina.....58
4.2.14. Caso de uso UC14: Obtención de Reporte de Cuentas x Cobrar.....60
4.2.15. Caso de uso UC15: Obtención de Reporte de Ventas.....62
4.2.16. Caso de uso UC16: Obtención de Reporte de Cortes de Caja.....64
4.2.17. Caso de uso UC17: Modificación de Precios de Kg. / Lt. de Gas.....66
4.2.18. Caso de uso UC18: Modificación de Ventas Registradas.....68

5. Análisis y diseño utilizando patrones.....71

5.1. Caso de uso UC1: Registrar ventas de pipas.....71
5.2. Caso de uso UC2: Registrar ventas de cilindros.....78
5.3. Caso de uso UC3: Registrar corte de caja.....83
5.4. Caso de uso UC4: Alta de trabajadores en el sistema.....87
5.5. Caso de uso UC5: Baja de trabajadores en el sistema.....90
5.6. Caso de uso UC6: Modificación de trabajadores en el sistema.....92
5.7. Caso de uso UC7: Alta de unidades en el sistema.....94
5.8. Caso de uso UC8: Baja de unidades en el sistema.....97
5.9. Caso de uso UC9: Modificación de unidades en el sistema.....99
5.10.Caso de uso UC10: Alta de clientes en el sistema.....101
5.11.Caso de uso UC11: Baja de clientes en el sistema.....104
5.12.Caso de uso UC12: Modificación de clientes en el sistema.....106
5.13.Caso de uso UC13: Obtención de reporte para nómina. 108
5.14.Caso de uso UC14: Obtención de reporte de cuentas x cobrar.....111
5.15.Caso de uso UC15: Obtención de reporte de ventas.....114
5.16.Caso de uso UC16: Obtención de reporte de cortes de caja.....117
5.17.Caso de uso UC17: Modificación de precios de kilo / litro de gas.....120
5.18.Caso de uso UC18: Modificación de ventas registradas.....123
5.19.Otros patrones de diseño.....127
5.19.1. Patrón: Representación de Objetos como Tabla.....127
5.19.2. Patrón: Identificador de Objeto.....127

6. Conclusiones.....131

7. Bibliografía.....133

CAPITULO I

El reuso en el diseño de aplicaciones

En la actualidad el desarrollo de proyectos de software requiere un mayor esfuerzo en las fases previas al desarrollo que el que se le dedicaba hace unos años, pues anteriormente, cuando el desarrollo de software se hacia de manera procedural, se podía realizar un proyecto siguiendo diagramas de flujo, esto en la actualidad no es suficiente.

Debido a las nuevas tendencias en desarrollo de software, hay que tratar de apoyarse de la mayor cantidad de herramientas que sea posible, así pues con el desarrollo de un lenguaje estándar propio para el diseño, como el UML, se tiene una forma ordenada de diseñar un sistema por un método orientado a objetos. Hoy en día estos métodos introducen esa noción de objetos reagrupando estructuras de datos, controles y tratamientos, los cuales proponen especificaciones globales de un sistema de información por medio de especificaciones complementarias estáticas y dinámicas.

UML son las siglas de Unified Modeling Language (Lenguaje Unificado de Construcción de Modelos), notación esquemática con que se construyen sistemas por medio de conceptos orientados a objetos.

UML es un lenguaje que permite modelar, construir y documentar los elementos que forman un sistema software orientado a objetos. La notación UML ha sido ampliamente aceptada debido a que incorpora las principales ventajas de cada uno de los métodos de particulares en los que se basa: BOCH, OMT, y OOSE. UML ha puesto fin a las llamadas guerras de métodos que se mantuvieron a lo largo de los 90´s en las que los principales métodos sacaban nuevas versiones que incorporaban las técnicas de los demás. Con UML se fusiona la notación de estas técnicas para formar una herramienta compartida entre todos los ingenieros software que trabajan en el desarrollo orientado a objetos. Hay que tener en cuenta que el UML no define un proceso de desarrollo específico, tan solo se trata de una notación. Es por eso que el proyecto que aquí se desarrolla basa su diseño y la aplicación de patrones de diseño en esta notación.

Un nuevo enfoque basado sobre la reutilización de componentes se empieza a imponer. Inicialmente confinado en las etapas de implementación de los sistemas, la reutilización de componentes se afirma cada vez más en las etapas tempranas (definición de requerimientos, análisis, concepción, etc.). Una de las voces más pertinentes es hoy en día los patrones de ingeniería: patrones de concepción (patrones de diseño), patrones de oficio (u objeto de oficio), etc.

El desarrollo por reutilización introduce este nuevo enfoque de desarrollo de sistemas de información según el cual es posible construir un nuevo sistema a partir de componentes existentes. La aplicación de este tipo de enfoque necesita de los métodos y de las técnicas para la identificación de componentes reutilizables, para la concepción y la realización de sistemas de reutilización y para la explotación de componentes en el contexto del desarrollo de un nuevo sistema.

1.1 Patrones en distintos campos de aplicación

El estudio de patrones está bien establecido en muchos otros campos incluyendo arquitectura, antropología, música, y sociología. Los recientes enfoques en ingeniería de software adoptando los patrones de software fueron altamente influenciados por Christopher Alexander, un investigador de la Universidad de California, Berkeley, quien ha escrito muchos trabajos sobre patrones encontrados en la arquitectura para casas, edificios, y comunidades.

Muchos campos usan patrones de varias maneras: en música y literatura, un patrón es la estructura o el diseño coherente de una canción o de un libro. En arte, un patrón es la composición o plan de un trabajo de arte gráfico o plástico. En arquitectura, un patrón es un diseño o un estilo arquitectónico, o los patrones de comportamiento de las personas, estudio del que se encarga la psicología.

Los patrones se convierten en bloques de construcción para el diseño y desarrollo. Encontrar y aplicar patrones indica progreso en un campo del esfuerzo humano.

Los patrones ayudan a aliviar la complejidad del software en varias fases del ciclo de vida. Aunque los patrones no son un método o un proceso del desarrollo de software, complementan métodos y procesos existentes. Por ejemplo, los patrones ayudan a relacionar las abstracciones en el análisis y las fases arquitectónicas del diseño con las realizaciones concretas de estas abstracciones en las fases de implementación y mantenimiento.

La idea del reuso es simple, pero su implementación no lo es y debe planearse cuidadosamente, ya que requiere un cambio en la cultura corporativa, en el desarrollo de software y de un conjunto de herramientas y habilidades.

El reuso se formaliza cuando el proceso de desarrollo de software, las herramientas, los programas de incentivos, los programas de entrenamiento y las métricas incluyen explícitamente el reuso. El reuso de componentes en ingeniería de software no es nuevo; sin embargo, sólo se ha utilizado informalmente entre las personas que trabajan en proyectos similares. Esto se debe principalmente a que lo primordial en el desarrollo de un sistema es su liberación y es muy difícil reconocer piezas de experiencia reusables [12].

1.2 Niveles de reuso en diseño de sistemas

El reuso dentro de lo que se considera Ingeniería de Software se puede aplicar a diferentes niveles [3].

1. Reuso de experiencia: una vez que se ha logrado formar un grupo de personas en un área determinada, estas personas pasan a formar parte de nuevos desarrollos aportando su conocimiento anterior.

2. Reuso de procesos: cuando un proceso de desarrollo muestra su utilidad, éste se replica a nuevos desarrollos.

3. Reuso de especificaciones: es la especificación de una función y desempeño que se puede utilizar en más de un sistema.

4. Reuso de diseños: es la especificación de la forma de implementar un requerimiento que se puede utilizar en más de un sistema. El diseño de un sistema establece una arquitectura de software que constituye un marco para el reuso posterior, pues determina funciones e interfaces de conexión.

5. Reuso de código: es la implementación de un diseño en un lenguaje específico y muchas veces en una plataforma determinada, que se puede utilizar en más de un sistema.

6. Reuso de pruebas: es la especificación de un conjunto de pruebas que debe satisfacer un tipo de componente. Se podría aplicar un componente de pruebas a las diferentes implementaciones de una especificación de requerimientos o de un diseño.

7. Reuso de documentación: es la síntesis explicativa de un componente y/o proceso del sistema (documentación de cada componente reusable, manuales del usuario, manuales de procedimientos, etc.) que se reusa.

Es aquí donde los patrones entran en acción, pues de todos estos niveles de reuso se puede obtener un catalogo de patrones, los cuales se irán acuñando de acuerdo a la experiencia que cada persona va acumulando. Los diferentes pasos que se van dando en cada etapa de los proyectos, la experiencia que se va acumulando y la identificación de los módulos funcionales en cada uno de estos proyectos así como la identificación y delimitación clara de los módulos en cada uno de los proyectos, irán dando lugar a cada uno de los patrones que los desarrolladores irán generando a lo largo de su carrera.

Los patrones ayudan a aliviar la complejidad del software en varias fases en el ciclo de vida del software. Aunque los patrones no son un método o un proceso de desarrollo de software, estos complementan métodos y procesos existentes. Por ejemplo, los patrones ayudan a ligar las abstracciones en el análisis de dominios y las fases de diseño arquitectónico con las realizaciones concretas de estas abstracciones en las fases de implementación y mantenimiento. En las fases de análisis y diseño, los patrones ayudan a guiar a los desarrolladores en la selección de arquitecturas de software que han demostrado ser exitosas. En las fases de implementación y mantenimiento, ayudan a documentar las propiedades estratégicas de los sistemas de software en un nivel más alto que el código fuente y modelos de módulos de software individual [25].

Así pues, en cada uno de estos niveles se pueden ir generando patrones de acuerdo a la experiencia, los cuales se pueden ir documentando de una manera mas formal, para que puedan ser compartidos con los demás miembros de su grupo de trabajo, en aplicaciones posteriores que se encuentren en un entorno similar a el que se tenia cuando fue generado el patrón.

1.3 Los patrones de diseño

Un patrón se podría definir como una solución recurrente para un problema en un contexto establecido. Podríamos entender un contexto como un entorno, situación o condiciones interrelacionadas dentro de las cuales existe algo. Un problema se podría ver como una cuestión insatisfecha, algo que se necesita investigar y resolver. Un problema se puede especificar mediante un conjunto de causas y efectos. Normalmente un problema está restringido al contexto en el que ocurre. La solución puede hacer alusión a la respuesta al problema dentro del contexto, que nos permite resolver las dificultades.

En general los patrones podrían quedar definidos como soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos. Son soluciones basadas en la experiencia y que se ha demostrado que funcionan.

Un patrón se puede agrupar en un catálogo de acuerdo a sus principales características:
Contexto, problema y solución, junto con otros aspectos importantes, como causas y consecuencias.

Un patrón describe, con algún nivel de abstracción, una solución experta a un problema. Normalmente, un patrón está documentado en forma de una plantilla. Aunque es una práctica estándar documentar los patrones en un formato de plantilla especializado, esto no significa que sea la única forma de hacerlo. Además, hay tantos formatos de plantillas como autores de patrones, esto permite la creatividad en la documentación de patrones. Los patrones solucionan problemas que existen en muchos niveles de abstracción. Hay patrones que describen soluciones para todo, desde el análisis hasta el diseño y desde la arquitectura hasta la implementación. Además, los patrones existen en diversas áreas de interés y tecnologías.

1.4 Identificación de un patrón de diseño

A lo largo del manejo de proyectos se puede notar que ocurren problemas similares en todos los proyectos, también se puede ver que emergen soluciones similares para la mayoría de los proyectos, aunque varíen las estrategias de implementación, las soluciones generales por lo regular han de ser bastante parecidas, es decir, que responden a un cierto patrón.

Luego, hay una "Regla de Tres", como se la conoce en la comunidad de los patrones. Esta regla es una guía para ascender un patrón candidato al catálogo de patrones. De acuerdo a esta regla, una solución permanece como un patrón candidato hasta que se ha verificado en al menos tres sistemas diferentes. Ciertamente que hay mucho espacio para la interpretación en reglas como ésta, pero ayudan a proporcionar un contexto para la identificación de patrones.
Muchas veces, soluciones similares podrían representar un sólo patrón. Cuando se decide cómo formar un patrón, es importante considerar cual es la mejor forma de comunicar la solución. Algunas veces, un nombre independiente mejora la comunicación entre los desarrolladores. Si es así, es mejor considerar la documentación de dos soluciones similares como dos patrones diferentes.

CAPITULO II

Introducción a los patrones de diseño

Una habilidad clave y fundamental en el análisis y diseño orientado a objetos es la asignación cuidadosa de responsabilidades a los componentes software.

¿Cómo deberíamos asignar las responsabilidades a las clases de objetos? ¿Qué clases deberían hacer que? Estas son preguntas claves en el diseño de un sistema.

Ciertas soluciones contrastadas a problemas de diseño se pueden expresar como principios, heurísticas, o patrones de buenas practicas, llamadas formulas de solución de problemas que codifican principios de diseño ejemplares. En este capitulo se presentan los principales patrones de diseño GRASP para la asignación responsabilidades.

Desde luego que existen otras habilidades necesarias para el Análisis y Diseño Orientado a Objetos, pero la asignación de responsabilidades suele ser una habilidad que requiere esfuerzo para llegar a dominar y al mismo tiempo tiene una importancia vital. En un proyecto real, un desarrollador podría no tener la oportunidad de abordar ninguna otra actividad de análisis o diseño, el principio de diseño con prisas de codificar, pero incluso en estos casos, la asignación de responsabilidades en inevitable.

También hay que prestar atención en la definición de objetos software y en como colaboran para satisfacer los requisitos.

2.1 Descripción de los patrones

Un sistema orientado a objetos se compone de objetos que envían mensajes a otros objetos para que lleven a cabo las operaciones.
Booch y Rumbaugh definen la responsabilidad como un contrato u obligación de un tipo o clase [6].

Los diagramas de interacción constituyen una descripción grafica de la solución muy clara que estas responsabilidades y postcondiciones satisfacen.

La calidad del diseño de la interacción de los objetos y la asignación de responsabilidades presentan gran variación. Las decisiones poco acertadas dan origen a sistemas y componentes frágiles y difíciles de mantener, entender, reutilizar o extender.

Una implementación hábil se funda en los principios cardinales que rigen un buen diseño orientado a objetos. En los patrones GRASP (General Responsability Asignment Software Patterns) se codifican algunos de ellos, que se aplican al preparar los diagramas de interacción, cuando se asignan las responsabilidades o durante ambas actividades.

Los diagramas de interacción son unos de los artefactos mas importantes que se preparan en el análisis y diseño orientado a objetos.

Es muy importante asignar acertadamente las responsabilidades al momento de elaborar los diagramas de interacción.

El tiempo y el esfuerzo que se dedican a su elaboración, así como un examen riguroso de la asignación de responsabilidades, deberían absorber parte considerable de la fase de diseño de un proyecto.

Los patrones nos pueden servir de mucho para mejorar la calidad del diseño.

Las responsabilidades pueden pertenecer principalmente a dos categorías:

1. Conocer.
2. Hacer.

Entre las responsabilidades de un objeto relacionadas con hacer podemos encontrar:

- Hacer algo en uno mismo.
- Iniciar una acción en otros objetos.
- Controlar y coordinar actividades en otros objetos.

Entre las responsabilidades relacionadas con conocer se encuentran:

- Estar enterado de los datos privados encapsulados.
- Estar enterado de la existencia de objetos conexos.
- Estar enterado de cosas que se puede derivar o calcular.

Las granularidad de la responsabilidad influye en su traducción a clases y métodos.

Responsabilidad no es lo mismo que método: los métodos se ponen en práctica para cumplir con las responsabilidades. Estas se implementan usando métodos que operen solos o en colaboración con otros métodos y objetos.

En los artefactos UML, las responsabilidades (implantadas como métodos) suelen tenerse en cuenta al momento de preparar los diagramas de interacción, pues estos muestran las decisiones referentes a la asignación de responsabilidades entre objetos, cuando se preparan, se toman decisiones sobre la asignación que se reflejan en los mensajes que son enviados a varias clases de objetos.

A medida que se va ganando experiencia se va formando un repertorio de principios generales y de expresiones que nos guían a crear Software. A algunos de estos podremos nombrarles patrones, si se registran en un formato estructurado que describa un problema y su solución, y si se le asigna un nombre. Así pues tenemos que en la terminología de objetos, el patrón es una descripción de un problema y su solución que recibe un nombre y que puede emplearse en otros contextos y que indica la manera de utilizarlo en circunstancias diversas.

De esta manera tenemos que; Asignar correctamente las responsabilidades es muy importante en el diseño orientado a objetos, La asignación de responsabilidades a menudo se asignan en el momento de preparar los diagramas de interacción.

Los patrones son parejas de problema/solución con un nombre, que estructuran buenos principios y sugerencias relacionados frecuentemente con la asignación de responsabilidades.

Es importante entender y aplicar estos principios durante la preparación de un diagrama de interacción, pues constituyen un fundamento de cómo se diseñara el sistema.

2.2 Los patrones GRASP

GRASP es un acrónimo que significa General Responsability Asignment Software Patterns (Patrones Generales de Software Para Asignar Responsabilidades).

A continuación se explicaran los primeros cinco patrones de GRASP:

- **Experto**
- **Creador**
- **Alta Cohesión**
- **Bajo Acoplamiento**
- **Controlador**

2.2.1 Patrón Experto

Solución:

Asignar una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad.

Problema:

¿Cuál es el principio fundamental en virtud del cual se asignan las responsabilidades en diseño orientado a objetos?

Un modelo de clase puede definir docenas y hasta cientos de clases de software, y una aplicación tal vez requiera el cumplimiento de cientos o miles de responsabilidades.

Durante el diseño orientado a objetos, cuando se definen las interacciones entre los objetos, tomamos decisiones sobre la asignación de responsabilidades a las clases. Si se hacen de forma adecuada los sistemas tienden a ser mas fáciles de entender, mantener y ampliar, y se nos presenta la oportunidad de reutilizar los componentes en futuras aplicaciones.

Explicación:

Experto es un patrón que se usa más que cualquier otro al asignar responsabilidades.

Es un principio básico que suele utilizarse en el diseño orientado a objetos, el cual expresa simplemente la intuición de que los objetos hacen cosas relacionadas con la información que poseen.

Nótese que el cumplimiento de una responsabilidad requiere a menudo información distribuida en varias clases de objetos. Ello significa que hay varios expertos parciales que colaboran en la tarea, cuando la información se encuentra esparcida en varios objetos, estos habrán de interactuar a través de mensajes para compartir el trabajo.

El patrón experto da origen a diseño donde el objeto de software, realiza las operaciones que normalmente se aplican a la cosa real que representa: a esto Meter Coad lo llama estrategia de “Hacerlo Yo Mismo” [13]. Por ejemplo, en el mundo real una venta no nos indica su total sin la ayuda de aparatos electromecánicos; se trata de un concepto inanimado. Alguien calcula el total de la venta. Pero en el terreno del software orientado a objetos, todos los objetos del software están vivos o animados, y pueden asumir responsabilidades y hacer cosas. Básicamente, hacen cosas relacionadas con la información de que disponen. A esto se le puede llamar principio de animación en el diseño orientado a objetos.

El patrón experto ofrece una analogía con el mundo real. Acostumbramos a asignar responsabilidad a individuos que disponen de la información necesaria para llevar a cabo una tarea. Por ejemplo, en una empresa, ¿Quién será el encargado de preparar un estado de pérdidas y ganancias? El empleado que tiene acceso a toda la información necesaria para elaborarlo: quizá el director financiero. Y del mismo modo que los objetos del software colaboran por que la información esta esparcida, lo mismo sucede con el personal de una empresa. El director financiero podrá pedir a los encargados de cuentas por cobrar y de cuentas por pagar que generen reportes individuales sobre créditos y deudas.

Beneficios:

Se conserva el encapsulamiento, ya que los objetos se valen de su propia información para hacer lo que se les pide. Esto soporta un bajo acoplamiento, lo que favorece al hecho de tener sistemas más robustos y de fácil mantenimiento.

El comportamiento se distribuye entre las clases que cuentan con la información requerida, alentando con ello definiciones de clase sencillas y más cohesivas que son mas fáciles de comprender y mantener. Así se brinda el soporte a una alta cohesión.

2.2.2 Patrón Creador

Solución:

Asignar a la clase B la responsabilidad de crear una instancia de clase A en uno de los siguientes casos:

- B agrega los objetos A.
- B contiene los objetos A.
- B registra las instancias de los objetos A.
- B utiliza específicamente los objetos A
- B tiene los datos de inicialización que serán transmitidos a A cuando este objeto sea creado (así que B es un experto respecto a la creación de A).

B es un Creador de los objetos A.
Si existe más de una opción, prefiera a la clase B que contenga la clase A.
Problema:

¿Quién debería ser responsable de crear una nueva instancia de alguna clase?
La creación de objetos es una de las actividades mas frecuentes en un sistema orientado a objetos. En consecuencia, es conveniente contar con un principio general para asignar las responsabilidades concernientes a ella. El diseño bien asignado, puede soportar un bajo acoplamiento, una mayor claridad, el encapsulamiento y la reutilizabilidad.

Explicación:

El patrón creador guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. El propósito fundamental de este patrón en encontrar un creador que debemos conectar con el objeto producido en cualquier evento. Al escogerlo como creador se debe dar soporte al bajo acoplamiento.

El Agregado Agrega la parte, el Contenedor Contiene el Contenido, el Registro Registra. En un diagrama de clases se registran las relaciones muy frecuentes entre las clases. El patrón creador indica que la clase incluyente del contenedor o registro es idónea para asumir la responsabilidad de crear la cosa contenida o registrada. Desde luego, se trata tan solo de una directriz.

El concepto de agregación se uso en la al examinar el patrón Creador, Y su definición podría quedar así; La agregación incluye cosas que están en una sólida relación de parte-todo o de parte-estructura; por ejemplo, Cuerpo agrega pierna, o párrafo agrega oración.

En ocasiones se puede encontrar un patrón creador buscando la clase con los datos de inicialización que serán transferidos durante la creación. Este es en realidad un ejemplo del patrón experto. Los datos de inicialización se transmiten durante la creación a través de algún método de inicialización, como un constructor de un método o una clase que cuenta con parámetros.

Por ejemplo, supongamos que una instancia de una clase “Pago” al momento de ser creada necesita inicializarse con el “total_Venta”. Puesto que una clase “venta” conocería el total, seria un buen candidato para ser el parámetro creador de “Pago”.

Beneficios:

Se brinda soporte a un bajo acoplamiento, lo cual supone menos dependencias respecto al mantenimiento y mejores oportunidades de reutilización. Es probable que el acoplamiento no aumente, pues la clase creada tiende a ser visible a la clase creador, debido a las asociaciones actuales que nos llevaron a elegirla como el parámetro adecuado.

2.2.3 Patrón Bajo Acoplamiento

Solución:

Asignar una responsabilidad para mantener bajo acoplamiento.

Problema:
¿Cómo dar a una clase dependencia escasa y a un aumento de la reutilización?

El acoplamiento es una medida de la fuerza con que la clase esta conectada a otras clases, con que las conoce y con que recurre a ellas. Una clase con bajo acoplamiento no depende de muchas otras, depende del contexto.

Una clase con Alto acoplamiento recurre a muchas otras, este tipo de clases no es conveniente, pues presentan los siguientes problemas:

- Los cambios de las clases afines ocasionan cambios locales.
- Son mas difíciles de entender cuando están aisladas
- Son mas difíciles de reutilizar por que se requiere la presencia de otras clases de las que dependen.

El bajo acoplamiento es un principio que se debe recordar durante las decisiones de diseño: es la meta principal que es preciso tener presente siempre. Es un patrón evaluativo que un diseñador debe aplicar al juzgar sus decisiones en el diseño.

El bajo acoplamiento estimula asignar una responsabilidad de modo que su colocación no incremente el acoplamiento tanto que produzca los resultados negativos propios de un alto acoplamiento.

El bajo acoplamiento soporta el diseño de clases más independientes, que reducen el impacto de los cambios, y también más reutilizables, que acrecientan la oportunidad de una mayor productividad. No puede considerarse en forma independiente de otros patrones como Experto o Alta Cohesión, sino que mas bien ha de incluirse como uno de los principios del diseño que influyen en la decisión de asignar responsabilidades.

El acoplamiento talvez no sea tan importante si no se busca la reutilización. Para apoyar una mejor reutilización de los componentes, al hacerlos mas independientes, el contexto de las metas de reuso ha de tenerse en cuenta antes de intentar reducir al mínimo el acoplamiento. Por ejemplo, a veces se dedica demasiado tiempo a la obtención de componentes reutilizables en futuros proyectos utópicos, a pesar de que no se sabe con certeza que se necesitarán. Con ello no se trata de decir que se pierde tiempo al tratar de reutilizarlos, solo que han de atenderse las consideraciones de costo-beneficio.

Una subclase esta acoplada firmemente a su superclase. La decisión de derivarla de una superclase ha de ser estudiada con mucho cuidado, pues es una forma muy sólida de acoplamiento. Supongamos por ejemplo, que es necesario guardar persistentemente los objetos en una base de datos relacional, en este caso esta bastante generalizado el diseño para crear una superclase abstracta denominada Objeto Persistente del cual se derivan otras clases. Esta subclase tiene la desventaja de acoplar estrechamente objetos del dominio con un servicio en particular y la ventaja de una herencia automática del comportamiento de persistencia, una especie de matrimonio por conveniencia. Y esto rara vez es una buena decisión en las relaciones.

No existe una medida absoluta de cuando el acoplamiento es excesivo. Lo importante es que el diseñador puede determinar el grado actual de acoplamiento y si surgirán problemas en caso de incrementarlo. En términos generales, han de tener escaso acoplamiento las clases muy genéricas y con grandes probabilidades de reutilización.

El caso extremo de Bajo acoplamiento ocurre cuando existe poco o nulo acoplamiento entre las clases. Ello no conviene porque una metáfora esencial en la tecnología de objetos es un sistema de objetos conectados que se comunican entre si a través de mensajes. Si el bajo acoplamiento se lleva a los extremos, dará origen a un diseño deficiente por producir objetos incoherentes, atiborrados y complejos que hacen todo el trabajo, con muchos otros objetos muy pasivos y de acoplamiento cero que funcionan como meros depósitos de datos. Un grado moderado de acoplamiento entre las clases es normal y

necesario si quiere crearse un sistema orientado a objetos, donde las tareas se realicen por colaboración entre objetos conectados.

Beneficios:

- No se afectan por cambios de otros componentes.
- Fáciles de Entender por separado.
- Fáciles de reutilizar.

2.2.4 Patrón Alta Cohesión

Solución:

Asignar una responsabilidad de modo que la cohesión siga siendo alta.

Problema:

¿Cómo mantener la complejidad dentro de límites manejables?

En la perspectiva del diseño orientado a objetos, la cohesión es una medida de cuan relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme.

Una clase con baja cohesión hace muchas cosas no afines o un trabajo excesivo. No conviene este tipo de clases pues presentan los siguientes problemas:

- Son difíciles de comprender.
- Son difíciles de reutilizar.
- Son difíciles de conservar.
- Son delicadas, las afectan constantemente los cambios.

Las clases con baja cohesión a menudo representan un alto grado de abstracción o han asumido responsabilidades que deberían haber delegado a otros objetos.

Explicación:

Como el patrón Bajo Acoplamiento, también Alta Cohesión es un principio que se debería tener presente en todas las decisiones de diseño. Es la meta principal que ha de buscarse en todo momento. Es un patrón evaluativo que un desarrollador aplica al valorar sus decisiones de diseño.

Grady Booch señala que se da una alta cohesión funcional cuando los elementos de un componente (clase por ejemplo) “Colaboran para producir algún comportamiento bien definido” [8].

Enseguida se mencionan algunos escenarios que ejemplifican los diversos grados de la cohesión funcional:

1. Muy baja cohesión: Una clase es la única responsable de muchas cosas en áreas funcionales muy heterogéneas.
    - Supongamos que existe una clase llamada RDB-RPC-Interfaz, y que es la una encargada de interactuar con las bases relacionales de datos y de atender las llamadas de procedimientos remotas. Estas son dos áreas radicalmente distintas, y cada una requiere mucho código de soporte. Las responsabilidades deberían repartirse en una familia de clases relacionada con el acceso RBD y en otra familia relacionada con el soporte RPC.
2. Baja cohesión: Una clase tiene la responsabilidad exclusiva de una tarea compleja dentro de un área funcional.
    - Supongamos que existe una clase llamada RDBInterfaz, la única encargada de interactuar con bases relacionales de datos. Los métodos de una clase están todos

relacionados, pero hay muchos de ellos y una enorme cantidad de código de soporte, puede haber cientos o miles de métodos. La clase debería dividirse en una familia de clases ligeras que compartan el trabajo para ofrecer el acceso a RDB.

3. Alta cohesión: Una clase tiene responsabilidades moderadas en un área funcional y colabora con las otras para llevar a cabo las tareas.

    - Suponga que existe una clase denominada RDBInterfaz, que se encarga de una parte de la interacción con bases relacionales de datos. Interactúa con una docena de clases afines, mediante el acceso RDB para recuperar objetos y guardarlos.

4. Cohesión Moderada: Una clase tiene un peso ligero y responsabilidades exclusivas en unas cuantas áreas que están relacionadas lógicamente con el concepto de clase, pero no entre ellas.

    - Suponga que existe una clase denominada compañía, la única responsable de: a) conocer a sus empleados b)conocer la información financiera. Estas dos áreas no están estrechamente relacionadas entre ellas, aunque lógicamente los están en el concepto de compañía. Además el numero total de métodos públicos es pequeño, lo mismo que el código de soporte.

Una regla práctica es la siguiente: una clase de alta cohesión posee un numero relativamente pequeño de tareas, con una importante funcionalidad relacionada y poco trabajo por hacer, colabora con otros objetos para compartir el esfuerzo si la tarea es grande.

Una clase con mucha cohesión es útil porque es bastante fácil darle mantenimiento, entenderla y reutilizarla. Su alto grado de funcionalidad, combinada con una reducida cantidad de operaciones, también simplifica el mantenimiento y los mejoramientos. La ventaja que significa una gran funcionalidad, también soporta un aumento de la capacidad de reutilización.

El patrón alta cohesión, presenta semejanzas con el mundo real. Todos sabemos que, si alguien asume demasiadas responsabilidades, no será eficiente. Esto se observa en algunos gerentes que no han aprendido a delegar. Muestran baja cohesión, prácticamente ya están desligados.

Beneficios:

- Mejoran la claridad y la facilidad con que se entiende el diseño
- Se simplifican el mantenimiento y las mejoras en funcionalidad.
- A menudo se genera un bajo acoplamiento.
- La ventaja de una gran funcionalidad soporta una mayor capacidad de reutilización, por que una clase muy cohesiva puede destinarse a un propósito muy especifico.

2.2.5 Patrón Controlador

Solución:

Asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase que represente una de las siguientes opciones:

- El sistema global (controlador de fachada).
- La empresa u organización global (controlador de fachada).
- Algo en el mundo real que es activo (por ejemplo el papel de una persona) y que pueda participar en la tarea (controlador de tareas).
- Un manejador artificial de todos los eventos del sistema de un caso de uso, generalmente denominados “Manejador <nombre caso de uso>” (Controlador de casos de uso).

Se debe usar la misma clase de controlador con todos los eventos del sistema en el mismo caso de uso.

En la lista no figuran las clases ventana, aplicación, vista ni documento, estas clases no deben ejecutar las tareas asociadas a los eventos del sistema; generalmente las reciben y las delegan al controlador.

Problema:
¿Quién debería encargarse de atender un evento del sistema.

Un evento del sistema es un evento de alto nivel generado por un actor externo; es un evento de entrada externa. Se asocia a operaciones del sistema: las que emite en respuesta a los eventos del sistema. Por ejemplo cuando un cajero que usa un sistema de Terminal en el punto de venta oprime el botón "Terminar Venta", esta generando un evento sistémico que indica que la venta ha terminado. De modo similar, cuando un escritor que usa un procesador de palabras pulsa el botón "Revisar Ortografía", esta produciendo un evento que indica realizar una revisión ortográfica.

Un Controlador es un objeto de interfaz no destinada al usuario que se encarga de manejar un evento del sistema. Define además el método de su operación.

Durante el análisis del comportamiento del sistema, sus operaciones son asignadas al tipo sistema, para indicar que son operaciones del sistema. Pero ello no significa que una clase llamada sistema las ejecute durante en el diseño.

Mas bien durante el diseño a la clase controlador se la asigna la responsabilidad de las operaciones del sistema.

Explicación:
La mayor parte de los sistemas reciben eventos de entrada externa, los cuales generalmente incluyen una interfaz grafica para el usuario (GUI) operado por una persona. Otros medios de entrada son los mensajes externos o las señales procedentes de sensores, como sucede en los sistemas de control de procesos.

En todos los casos, si se recurre a un diseño orientado a objetos, hay que elegir los controladores que manejan esos eventos de entrada. Este patrón ofrece una guía para tomar decisiones apropiadas que generalmente se aceptan.

La misma clase controlador debería usarse con todos los eventos sistemáticos de un caso de uso, de modo que podamos conservar la información referente al estado del caso. Esta información será útil para identificar los eventos del sistema fuera de presencia (entre ellas por ejemplo una operación efectuar pago, antes de terminar venta).

Un defecto frecuente al diseñar controladores consiste en asignarles demasiada responsabilidad. Normalmente un controlador debería delegar a otros objetos el trabajo que ha de realizarse mientras coordina la actividad.

La primera categoría de controlador es un controlador de fachada que representa al sistema global. Es una clase que, para el diseñador, representa de alguna manera al sistema entero. podría tratarse de una unidad física, como una clase Caja, InterruptordeComunicaciones o Robot; una clase que representa todo el sistema de software, como SistemaInformacionalMenudeo, SistemadeManejodeMensajes o ControladordeRobot o cualquier otro concepto que, por decisión del diseñador, denote el sistema global.

Si se recurre a la cuarta categoría de controlador (Manejador artificial de casos de uso), habrá entonces un controlador para cada caso. Adviértase que este no es un objeto del dominio; es un concepto artificial cuyo fin es dar soporte al sistema (una fabricación pura en términos de los patrones de GRASP). Por ejemplo si en una aplicación de un punto de venta existen casos de uso como "comprar productos" y "devolver productos" habrá una clase ManejadordeComprarProductos y una clase ManejadordeDevolverProductos.

¿Cuándo deberíamos escoger un controlador de casos de uso?

Es una alternativa que debe de tenerse en cuenta si el hecho de asignar las responsabilices en cualquiera de las otras opciones de controlador genera diseños de baja cohesión o alto acoplamiento. Esto ocurre generalmente cuando un controlador comienza a saturarse con demasiadas responsabilidades. Un controlador de casos de uso constituye una buena alternativa cuando hay muchos eventos de sistema entre varios procesos: asigna un manejo a clases individuales controlables, además de que ofrece una base para reflexionar sobre el estado del proceso actual.

Un corolario importante del patrón controlador es que los objetos externos de conexión (por ejemplo los objetos ventana, los applets o pequeñas aplicaciones) y la capa de presentación no deberían tener la responsabilidad de llevar a cabo los eventos del sistema.

Dicho en otras palabras, las operaciones del sistema (las cuales reflejan los procesos de la empresa o el dominio) deberían manejarse en la capa de dominio de los objetos y no en las de interfaz, presentación o aplicación.

Beneficios:

Mayor potencial de los componentes reutilizables. Garantiza que la empresa o los procesos del dominio sean manejados por la capa de los objetos del dominio y no por la de la interfaz. Desde el punto de vista técnico las responsabilidades del controlador podrían cumplirse en un objeto de interfaz, pero esto supone que el código del programa y la lógica relacionada con la realización de los procesos del dominio puro quedarían incrustados en los objetos interfaz o ventana.

Un diseño de interfaz como controlador reduce la posibilidad de reutilizar la lógica de los procesos del dominio en aplicaciones futuras, por estar ligada a una interfaz determinada (por ejemplo un objeto similar a una ventana) que rara vez puede utilizarse en otras aplicaciones. En cambio, el hecho de delegar a un controlador la responsabilidad de la operación de un sistema entre las clases del dominio soporta la reutilización de la lógica para manejar los procesos afines del negocio en aplicaciones futuras.

Reflexionar sobre el estado del caso de uso. A veces es necesario asegurarse de que las operaciones del sistema sigan una secuencia legal o poder razonar sobre el estado actual de la actividad y las operaciones en el caso de uso subyacente. Por ejemplo, tal vez deba garantizarse que la operación "EfectuarPago" no ocurra mientras no se concluya la operación "TerminarVenta". De ser así, esta información sobre el estado ha de capturarse en alguna parte; en el controlador en una buena opción, sobre todo si se emplea a lo largo de todo el caso (cosa que se recomienda).

Problemas y soluciones:

Controladores saturados. Una clase controlador mal diseñada presentara baja cohesión: esta dispersa y tiene demasiadas áreas de responsabilidad; recibe el nombre de controlador saturado.

Entre los signos de saturación se pueden mencionar los siguientes:

- Hay una sola clase de controlador que recibe todos los eventos del sistema y estos son excesivos. Tal situación suele ocurrir si se escoge un controlador de papeles o de fachada.
- El controlador realiza el mismo muchas tareas necesarias para cumplir el evento del sistema, sin que delegue trabajo. Estro suele constituir una violación de los patrones Experto y Alta Cohesión.
- Un controlador posee muchos atributos y conserva información importante sobre el sistema o dominio (información que debería haber sido redistribuida entre otros objetos) y también duplica la información en otra parte.

Hay varias formas de resolver el problema de un controlador saturado, entre las que se encuentran las siguientes:

1. Agregar más controladores: un sistema no necesariamente debe tener uno solamente. Además de los controladores de fachada, recomendamos emplear los controladores de papeles o los de casos de uso. Considere, por ejemplo, una aplicación con muchos eventos sistémicos como podría serlo el sistema de reservaciones de una línea aérea. podría incluir los siguientes controladores:

Controladores de Papeles	Controladores de Casos de Uso
Agente de Reservaciones	Manejador de Hacer Reservaciones
Programador de Vuelos	Manejador Administración de Programación
Analista de Tarifas	Manejador de administración de Tarifas

2. Diseñe el controlador de modo que delegue fundamentalmente a otros objetos el desempeño de las responsabilidades de la operación del sistema.

Como advertencia se puede mencionar que los controladores de papeles pueden conducir a la obtención de diseños deficientes.

Asignar una responsabilidad a un objeto de papeles humanos en una forma que imite lo que ese papel realiza en el mundo real (por ejemplo, el objeto de software cajero que maneja EfectuarPago) es aceptable a condición de que el diseñador conozca los posibles riesgos y los evite. En particular se corre el peligro de generar un controlador poco coherente de papeles que no deleguen. Un buen diseño orientado a objetos "les da vida", al asignarles responsabilidades, aunque presenten cosas inanimadas del mundo real. Si escoge un controlador de papeles no hay que caer en la trampa de diseñar objetos de tipo humano que realicen todo el trabajo; opte mas bien por delegar.

En términos generales es aconsejable usar poco los controladores de papel.
La capa de presentación no maneja los eventos del sistema

Un corolario importante (lo repetimos) del patrón controlador es que los objetos de interfaz (por objeto objetos de ventanas, applets) y la capa de presentación no deberían encargarse de manejar los eventos del sistema. En otras palabras las operaciones de este (que reflejan los procesos de la empresa o el dominio) han de realizarse en la capa del dominio de objetos y no en la de interfaz, la de presentación y la de aplicación.

2.2.5.1 Sistemas del manejo de mensajes y el Patrón Comando

Muchas aplicaciones no cuentan con una interfaz para el usuario y, en cambio, reciben mensajes provenientes de algún otro sistema externo; son sistemas de manejo de mensajes. Un conmutador de telecomunicaciones es en buen ejemplo. El mensaje puede estar codificado en registro, en una secuencia de bytes sin procesar o en un mensaje "real" dirigido a un objeto, si se usa un mecanismo de comunicación ínter procesos orientado a objetos como Corba.

En una aplicación de una interfaz para usuario (una ventana, por ejemplo), la ventana puede decidir cual será el objeto controlador. Varias ventanas pueden colaborar con los controladores, sobre todo si se recurre a un controlador de casos de uso.

En cambio el diseño de la interfaz y del controlador es diferente en una aplicación de manejo de mensajes,. Tratándose de un caso simple, se puede recomendar el siguiente diseño. Si se desea un diseño mas complejo y de muchas características, se puede consultar el patrón emisor-receptor en [7].

Para manejar los mensajes de eventos del sistema en un sistema de este tipo:

1) Definir un solo controlador para todos los mensajes de eventos; puede ser un controlador de fachada o un controlador individual de casos de uso, con un nombre como manejador de mensajes.

2) Usar el patrón comando [18] para contestar la consulta. El patrón comando especifica la definición de una clase en cada mensaje o comando, todas ellas son el método ejecutar. El controlador creara una distancia comando correspondiente al mensaje del evento del sistema y lo enviara a un mensaje ejecutar. Las clases comando cuentan con un método único ejecutar que especifica las acciones de el como se aprecia en la figura.

Patrones relacionados:

- Comando: En un sistema de manejo de mensajes; un objeto aislado comando puede representar y manejar los mensajes [18].
- Fachada: La selección de un objeto que representa un sistema u organización global para que sea controlador lo hace un tipo fachada [18].

- Capas: Este en un patrón Siemens [7]. Colocar la lógica del dominio en su capa y no en la capa de presentación es parte del patrón capas
- Artefacto Puro: Es otro patrón de GRASP. El artefacto puro es una clase artificial, no un concepto del dominio. Un controlador de casos de uso es un tipo de artefacto puro.

2.3 Patrones Factoría y Fachada

2.3.1 Patrón Factoría

Solución:

Crear un objeto fabricación pura denominado factoría que maneje la creación.

Problema:

¿Quien debe ser el responsable de la creación de los objetos cuando existen consideraciones especiales?, como una lógica de creación compleja, el deseo de separar las responsabilidades de la creación para mejorar la cohesión, etc.

Un principio de diseño fundamental (normalmente considerado un principio de diseño de arquitectura) es el de diseñar para mantener una separación de intereses. Es decir, dividir en módulos o separar intereses distintos en áreas diferentes, de manera que cada una tenga un propósito cohesivo. Por ejemplo la capa de los objetos software destaca las responsabilidades centradas relativamente en la lógica de la aplicación, mientras que un grupo diferente de objetos es responsable de las cuestiones de conectividad con sistemas externos.

Por tanto la elección de un objeto del dominio para crear estos adaptadores no esta de acuerdo con el objetivo de separación de intereses y disminuye su cohesión.

La alternativa es aplicar el patrón factoría, en el que se define un objeto fabricación pura "factoría" para crear los objetos.

Los objetos factoría tienen varias ventajas:

- Separan la responsabilidad de la creación compleja en objetos de apoyo cohesivos.
- Ocultan la lógica de creación potencialmente compleja.
- Permiten introducir estrategias para mejorar el rendimiento de la gestión de la memoria, como objetos cache y de reciclaje.

En una factoría de servicios, la lógica que una clase se crea se resuelve leyendo el nombre de la clase desde una fuente externa (por ejemplo, por medio de una propiedad del sistema) y después se carga la clase dimanicamente. Este es un ejemplo de diseño dirigido por los datos parcial. El diseño consigue variaciones protegidas con respecto a los cambios en la clase de implementación. Sin cambiar el código fuente en la clase factoría, podemos crear instancias de nuevas clases cambiando el valor de sus propiedades.

La idea que se esconde detrás de este patrón es la de centralizar el sitio donde se crean los objetos, normalmente donde se crean objetos de una misma familia,

La clase factoría devuelve una instancia de un objeto según los datos que se le pasan como parámetros. Para que la creación centralizada de objetos sea lo más útil y eficaz posible, es de esperar que todos los objetos creados desciendan de la misma clase o implementen la misma interfaz (es decir, hagan una operación similar pero de distintas formas), así podemos usarlos todos de la misma manera, con los mismos métodos, sin importarnos que clase concreta estamos tratando en cada momento.

2.3.2 Patrón Fachada

Solución:

Definir un único punto de conexión con una interfaz o subsistema. Un objeto fachada que envuelve esta conexión al subsistema. Este evento fachada presenta una única interfaz unificada y es responsable de colaborar con los componentes del sistema.

Problema:

Se requiere una interfaz común unificada para un conjunto de implementaciones o interfaces dispares, como en un subsistema.

Podría no ser conveniente acoplarla con muchas cosas del subsistema, o la implementación del subsistema podría cambiar.

El patrón de diseño fachada sirve para proveer de una interfaz unificada sencilla que haga de intermediaria entre un cliente y una interfaz o grupo de interfaces más complejas.

En el análisis de requisitos se recomienda identificar los puntos concretos del escenario en todos los casos de uso. El patrón fachada envuelve y oculta un subsistema detrás de un objeto.

La fachada es un objeto Front-End que es el único punto de entrada para los servicios de un subsistema o interface, la implementación y otros componentes del subsistema son privados y no pueden verlos los componentes externos. La fachada proporciona variaciones protegidas frente a los cambios de las implementaciones de un subsistema.

Fachada puede:

- Hacer una biblioteca de software más fácil de usar y entender, ya que fachada implementa métodos convenientes para tareas comunes.
- Hacer el código que usa la librería más legible, por la misma razón.
- Reducir la dependencia de código externo en los trabajos internos de una librería, ya que la mayoría del código lo usa fachada, permitiendo así más flexibilidad en el desarrollo de sistemas.
- Envolver una colección mal diseñada de APIs con un solo API bien diseñado.

Entonces, si un cliente necesita acceder a parte de la funcionalidad de un sistema más complejo, una buena solución seria:

- Definir una interfaz que permita acceder solamente a esa funcionalidad.
- Crear un intermediario más legible.
- Crear una API intermedia, bien diseñada, que permita acceder a la funcionalidad de las demás.

El patrón fachada trata de simplificar la interface entre dos sistemas o componentes de software ocultando un sistema complejo detrás de una clase que hace las veces de pantalla o fachada.

La idea principal es la de ocultar todo lo posible la complejidad de un sistema, el conjunto de clases o componentes que lo forman, de forma que solo se ofrezca un (o unos pocos) punto de entrada al sistema tapado por la fachada.

Una ventaja más de usar una clase fachada para comunicar las dos partes o componentes, es la de aislar los posibles cambios que se puedan producir en alguna de las partes. Si se cambia, por poner un ejemplo, el medio de comunicación o de almacenamiento de una de las partes, la otra, que por ejemplo hace la presentación, no tiene porque enterarse, y viceversa.

2.4 Responsabilidades, Representación de Papeles y las Tarjetas CRC

Aunque no sean una parte formal del lenguaje UML, las tarjetas CRC [4] (Siglas en ingles de Class Responsability Colaborar). (Clase Responsabilidad y Colaborador) Son otra herramienta con la cual a veces se asignan las responsabilidades y se indica una colaboración con otros objetos. Fueron inventados por Kent Beck y Ward Cunninghan, principales promotores para que los diseñadores de objetos piensen en términos más abstractos sobre la asignación de responsabilidades y de las colaboraciones.

Las tarjetas CRC son tarjetas índices, una por cada clase sobre las cuales se abrevian responsabilidades de la clase y se anota una lista de objetos con los que colabora para desempeñarla. Suelen elaborarse en una sesión de grupos pequeños donde los participantes representan papeles de diversas clases. Cada grupo tiene las tarjetas CRC correspondientes a las clases cuyo papel esta desempeñado.

La representación de papeles hace divertido aprender a pensar en función de objetos y responsabilidad. Pero las tarjetas basadas en texto tienen la capacidad limitada para registrar las colaboraciones en forma exhaustiva. Los diagramas de colaboración y las de clase describen mejor las conexiones entre los objetos y el contexto general.

Se recomienda por su utilidad las actividades grupales de diseño dirigidas a asignar responsabilidades y a representar papeles; los resultados pueden apuntarse después de en las tarjetas CRC o en diagramas. Las tarjetas son un método de registrar los resultados de la asignación de responsabilidades y de las colaboraciones. El registro es mas satisfactorio si se utiliza este tipo de diagramas. El valor real no esta constituido por las tarjetas, sino por el análisis de asignación de responsabilidades y por la actividad relacionada con la representación de papeles.

CAPITULO III

Análisis de requerimientos del sistema

Objetivo del sistema:

Se planea desarrollar un sistema para el control de liquidaciones de una empresa a la que denominaremos “Gas de Puebla S. A. de C. V.”

Proyecto: Sistema de Caja de Gas de Puebla.

Descripción:

- Gas de Puebla S.A. de C.V. es una empresa que se dedica a la distribución de Gas L.P. la cual requiere un sistema que lleve a cabo el control de las liquidaciones que realizan todos los días los operadores de las rutas tanto de venta de cilindros, como de venta de gas estacionario, Este sistema debe poder registrar los datos de la venta de una unidad, ligada con el operador y el ayudante que realizó la venta durante el transcurso del día, así como los montos totales de Gas y de efectivo que ha de liquidar el operador.

- Las ventas se pueden realizar tanto en efectivo como a crédito, en el caso de que solo se realicen ventas en efectivo, solo se registraran los montos totales de la ruta en ese día, si la unidad realizo ventas a crédito, el sistema debe registrar los montos totales de las ventas a crédito y las ventas en efectivo, y registrar aparte los montos totales de cada una de las ventas que se realizaron a crédito, junto con los datos del cliente y los datos del vendedor, para así poder llevar un control de los créditos que se han otorgado.

- El sistema debe ser de acceso restringido, así que se debe validar un usuario que intente ingresar en el como un usuario que tenga los permisos necesarios, para esto los usuarios que tengan acceso al sistema estarán organizados en varios grupos como pueden ser cajeros, para usuarios que van a tener acceso a las opciones de caja, supervisores y administradores para usuarios que van a poder realizar operaciones mas delicadas dentro del sistema, como dar de alta operadores, unidades, clientes o usuarios en el sistema, generar y revisar los reportes que el sistema va a poder crear.

- El sistema requiere de un formato para los reportes que se va a poder generar, para que estos se puedan leer por aquellos que tengan que verificarlos para poder realizar sus tareas diarias, este reporte deberá contener los montos que ha liquidado cada operador, ordenados por el numero de ruta que ha usado en ese día el operador, así como el concentrado de los totales que se obtienen por el concentrado de todas las rutas y todas las pipas por separado, así como un reporte general que va acumulando el concentrado de los totales durante un periodo de una semana. El sistema deberá quedar abierto para que en etapas posteriores se le puedan agregar módulos para controlar correctamente la nómina de la empresa, los créditos que se han efectuado y las cobranzas que se tienen que llevar a cabo en fechas recientes.

Algunas de las Posibles operaciones podrían ser:

*Validación de usuarios:

El sistema es de acceso restringido y los usuarios que tienen acceso a el cuentan con distintos permisos para poder trabajar en el; así un gerente o administrador del sistema cuenta con permiso para modificar el precio del gas, lo cual se hace cada mes, agregar rutas y operadores nuevos al sistema, mientras que un cajero solo ha de poder registrar las liquidaciones de los operadores en ese día.

*Acceso a utilerías del sistema:

Cuando un usuario sea validado como administrador, se presentaran opciones para poder modificar los datos del sistema como por ejemplo:

-Dar de alta una ruta
-Dar de alta un operador
-Dar de alta un cliente para aplicar créditos.
--En general un Administrador podrá dar de alta o de baja a cualquier tipo de usuario que se encuentre registrado en el sistema, modificar sus datos y cambiar sus perfiles.

*Acceso a sistema de caja

En la sección de caja se pueden presentar las siguientes opciones:
-Liquidar Cilindros
-Liquidar Estacionario
-Corte de Caja
-Reportes

Aquí el sistema permite capturar y liquidar el total de las ventas que han de pagar las unidades que salieron a trabajar durante el día, se captura la cantidad de tanques vendidos de 20, 30, 45, y kgs. en anforitas. De la misma manera para las pipas se registra la cantidad de litros vendidos en el día, así como las servicios (notas) hechos con el numero de nota, la cantidad de litros vendidos y el monto total a liquidar, así como el tipo de venta, pues esta puede realizarse a crédito, en efectivo o por cheque.

El sistema de caja debe mandar un reporte de los kilos vendidos por unidad (Ruta, pipa), en un periodo de tiempo determinado (Semana, mes, días), este reporte solo debe mostrar el concentrado de los kilos y los montos totales que se han vendido por alguna ruta en el rango de tiempo definido.

Aquí también se registra la línea de crédito, el sistema debe reportar la cantidad y el monto del gas que se vendió a crédito, el cliente y posteriormente, calcular la fecha limite del vencimiento.

Los clientes son dados de alta previamente en el sistema, para poderle aplicar una venta a crédito.

El sistema debe conocer previamente los usuarios que ingresan al sistema, las unidades, los operadores de las unidades, y los clientes a los que se les realiza venta a crédito.

Algunas de las primeras funciones que se pueden ver a grandes rasgos y de una manera muy general son:

Funciones:

1. Permitir el acceso al sistema.
2. Dar de alta Usuarios del sistema.
3. Dar de alta Operador/ayudante de las unidades.
4. Dar de alta Unidades Rutas/Pipas que realizan las ventas.
5. Dar de alta clientes para ventas a crédito
6. Registrar liquidaciones de las unidades
7. Guardar las liquidaciones del día en la BD para generar los reportes.
8. Calcular totales de ventas de gas
9. Calcular totales de ventas en pesos
10. Generar reporte con totales del día
11. Generar reporte de los ingresos del día.
12. Calcular totales para reportes en periodos de tiempo. (Semanal, mensual, anual)
13. Calcular tiempo de vencimiento de ventas a crédito. (Créditos Vencidos).

Existen varios patrones de diseño, de los cuales nos vamos a apoyar para poder ir generando la delegación de responsabilidades y la delimitación de estas, una vez que tengamos mas o menos definidos los conceptos que se van a ir registrando conforme se vaya avanzando en el diseño de la solución al problema presentado, como por ejemplo el patrón experto, el patrón Creador, El patrón Alta Cohesión, El patrón Bajo Acoplamiento, y el patrón Controlador, o los patrones Factoría y Fachada. Estos patrones nos ayudaran a asignar las responsabilidades de una manera clara y concisa, al momento de realizar el diseño.

3.1 Análisis preliminar de casos de uso (casos de uso clásicos)

Se Presenta a continuación el desarrollo de unos casos de uso de manera informal, para ir desvelando las operaciones que van a ser necesarias en el sistema.

Casos de Uso

1-Caso de uso: Validar Usuario (Ingresar al Sistema).

Actores: Usuario (General)
Tipo: Primario
Función: Registrar y brindar permisos a un usuario.
Descripción:

Un usuario intenta ingresar al sistema, el sistema le pide que escriba su nombre y contraseña, el usuario teclea estos datos, los datos son buscados en la base de datos de los usuarios, se comparan y si existen, el usuario ingresa al sistema, de lo contrario dice que no existe el usuario o que la contraseña no es valida y se permite reingresar los datos.

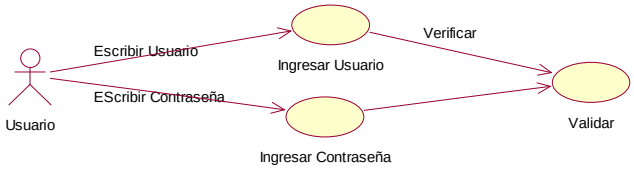


Figura 3.1 Caso de Uso: Validar Usuario

2-Caso de uso: Dar de Alta Usuario.

Actores: Usuario (Administrador)
Tipo: Primario
Descripción:

Un usuario validado como administrador ingresa al sistema para dar de alta un nuevo usuario del sistema, el cual va a pedir los datos personales y el grupo a el que va a pertenecer.

Los grupos serian; cajero o administrador, cuentasXcobrar, contabilidad, auxiliar_de_contabilidad. Cada uno de estos grupos contara con los permisos necesarios para realizar sus labores diarias.

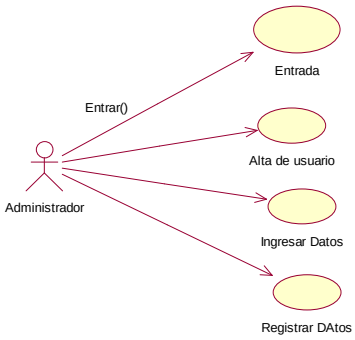


Figura 3.2 Caso de Uso: Alta de Usuario

3-Caso de uso: Dar de alta Operadores o/y Ayudantes.

Actores: Usuario (Administrador)
Tipo: Primario
Descripción:

Un administrador ingresa al sistema para dar de alta un Operador o un Ayudante, el cual quedara registrado en el sistema y con la autorización para poder liquidar una unidad, aquí solo habrá dos grupos que serán Operador y Ayudante, y ambos deben poder aparecer con opciones para liquidar una unidad.

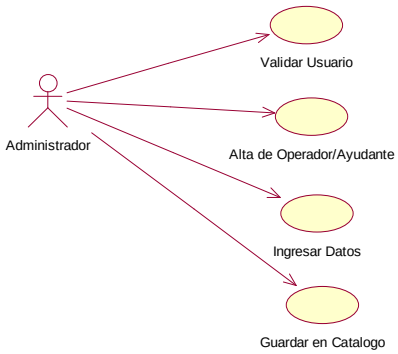


Figura 3.3 Caso de Uso: Dar de Alta Operador/Ayudante.

4-Caso de uso: Dar de alta Unidad: Ruta/Pipa.

Actores: Usuario (Administrador)
Tipo: Primario
Descripción:

Un administrador ingresa al sistema para dar de alta los datos de una nueva unidad repartidora de gas, la cual ha de quedar registrada en el catalogo de unidades, guardada en uno de los dos grupos, ya sea ruta o pipa, para que esta pueda ser tomada en cuenta en caja para liquidar una venta.

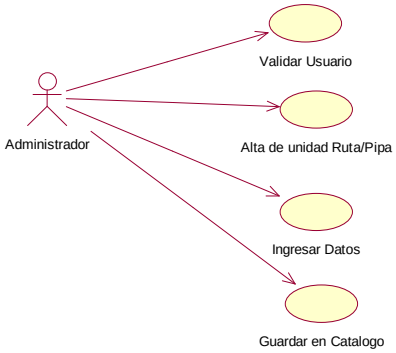


Figura 3.4 Caso de uso: Alta de Unidad Ruta/Pipa

5-Caso de uso: Dar de alta clientes (Para venta a crédito).

Actores: Usuario (Administrador)
Tipo: Primario
Descripción:

Un administrador ingresa al sistema para dar de alta un cliente, el cual desde ese momento cuenta con autorización para venta a crédito, al menos que aparezca con deudas fuera del periodo de pago (atrasadas).

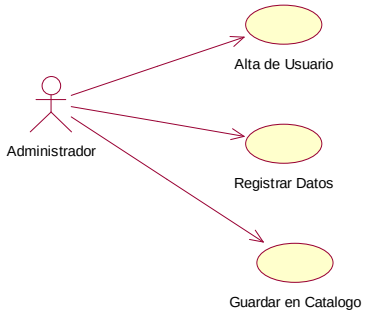


Figura 3.5 Caso de uso: Alta de Cliente

6-Caso de uso: Liquidar Ruta.

Actores: Cajero
Tipo: Primario (Esencial)
Descripción:

Un operador de ruta llega a caja a liquidar su venta del día, entrega su papeleta del día, el cajero ingresa los datos de la venta que marca la papeleta, que son datos del operador y el ayudante, los totales de cilindros que vendió, y en algunos casos si esta venta fue hecha a crédito, el sistema le regresa en pantalla el importe a cobrar, el cajero recibe el importe y entrega el cambio en caso de que lo haya, la información se guarda automáticamente en un registro de la base de datos, se acumulan los datos de los totales de cilindros, de kilos y de efectivo y se guardan en la base de datos para poder obtener el reporte posteriormente, cuando se haga el corte de caja.

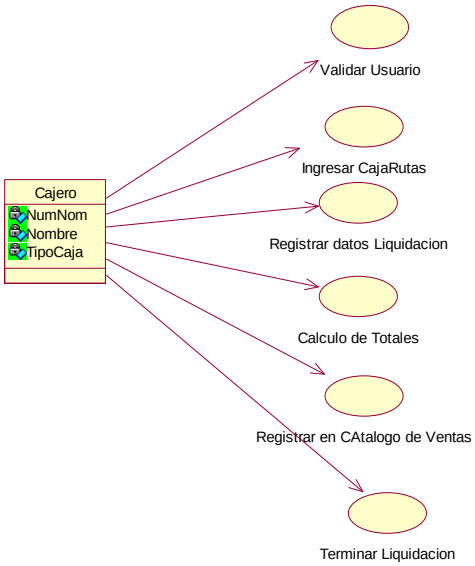


Figura 3.6 Caso de uso: Liquidar Ruta

7-Caso de uso: Liquidar Pipa.

Actores: Cajero
Tipo: Primario (Esencial)
Descripción:

Un operador de pipa llega a la caja y entrega su papeleta, el cajero ingresa los datos de el totalizador y después ve ingresando los datos de cada uno de los servicios que hizo el operador, nota por nota, especificando si fue en efectivo o a crédito, en caso de que sea a crédito, también se registra el cliente, y si es que este cliente tiene precio especial, el sistema regresa el total de efectivo que debe liquidar el operador, la información se guarda automáticamente en un registro de la basa de datos correspondiente. Y los totales se ven acumulando y guardando en un registro especial para el reporte final al hacer el corte de caja.

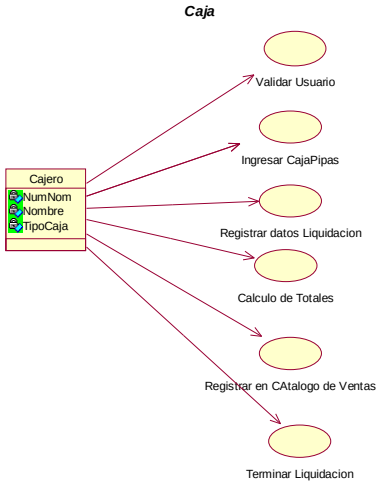


Figura 3.7 Caso de Uso: Liquidar Pipa

8-Caso de uso: Corte de Caja.

Actores: Cajero
Tipo: Primario
Descripción:

El cajero ha terminado de registrar las liquidaciones del día, hace el recuento de todo el efectivo, cheques y otros datos que tiene como ingresos y lo cuadra con el total que tiene registrado el sistema. El cajero genera e imprime el reporte que le corresponde. Aunque este debe quedar guardado en algún registro especial para que este pueda ser revisado posteriormente por los gerentes, pero no debe poder ser modificado por nadie después de ser generado.

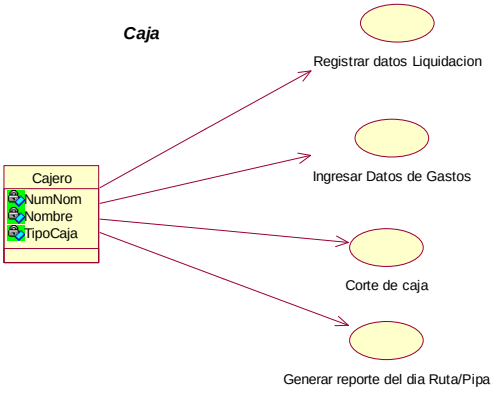


Figura 3.8 Caso de Uso Corte de Caja

9-Caso de uso: Generar Reporte.

Actores: Cajero o Administrador
Tipo: Primario
Descripción:

Una vez que han terminado de liquidar todas las unidades ya sea rutas o pipas, y se ha hecho el corte de caja se puede generar el reporte de caja, ya sea el que se manda a imprimir diario, o un reporte que se solicita de una fecha diferente o en un rengo de tiempo determinado o de fechas anteriores.

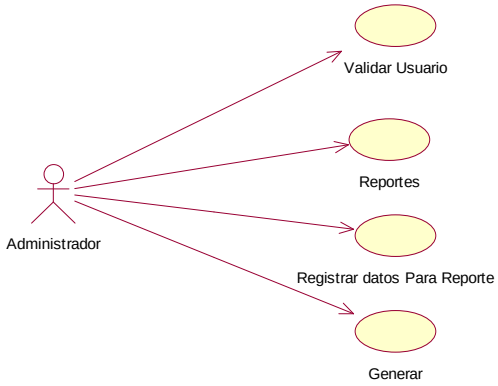


Figura 3.9 Caso de uso: Generar Reporte

10-Caso de uso: Dar salida a unidades (Opcional).

Actores: Otro
Tipo: Opcional
Descripción:

Un usuario asignado a esta tarea ingresa al sistema para asignar la carga con la que salio cada unidad, tanto de cilindros como de litros en pipa. Los datos son registrados para agilizar las liquidaciones de las rutas.

11-Caso de uso: Reporte de créditos.

Actores: Administrador, Auxiliar_de_Contabilidad, Cajero.
Tipo: Secundario.
Descripción:

El usuario solicita este reporte para verificar que créditos existen y la antigüedad de los mismos, el sistema despliega todos los clientes que tienen saldos por pagar y las fechas en que fueron hechas esas ventas a crédito.

3.2 Análisis del Modelo Conceptual

Antes de definir un modelo estático o de clases es necesario definir un modelo conceptual, el cual nos va a mostrar todos los conceptos presentes en el dominio del problema. Un concepto para este caso, en términos de la Programación Orientada a Objetos, es un objeto del mundo real; es decir, es la representación de cosas del mundo real y no de componentes de *software*. En este modelo no se definen operaciones (o métodos); en este modelo se pueden mostrar los conceptos, los atributos de los conceptos (opcionalmente) y la relación o asociación entre ellos. Informalmente se podría decir que un concepto es una idea, cosa u objeto. Para descubrirlos debemos analizar los sustantivos en las descripciones textuales del dominio del problema, es decir, de la descripción del sistema, de los requerimientos y de los Casos de Uso. También se puede tratar de llevar un patrón mediante el cual descubrir conceptos que van a formar parte del sistema y que tal vez no estarían contemplados sino hasta etapas posteriores del análisis.

Como por ejemplo el cuadro que se presenta abajo:

Conceptos.

Objetos Físicos Tangibles	Cilindro, Caja, Gas
Especificaciones, diseño o descripciones de casos	Liq. Cilindro, Liq estacionario
Lugares	Caja
Transacciones	Pago efectivo, Pago cheque, Credito
Papel de las Personas	Cajero, Aux_Cont,Administrador,Operador, Ayudante
Contenedores de Otras Cosas	Ruta_Cil, Pipa
Cosas dentro de un Contenedor	Cilindros, Litros_Gas, Operador, Ayudante
Otros sistemas externos al sistema	Excel
Organizaciones	Caja, Cuentas Por Cobrar
Eventos	Venta de Gas, Gastos de la Empresa
Procesos	TotalLitrosGas,TotalKilosGas,ReporteCaja
Reglas y Políticas	Políticas de Crédito, Políticas de Vales.
Catálogos	Usuarios, OperadorAyudante,RutasPipas, Clientes
Registro de Finanzas de trabajo, Contratos de asuntos legales	Reporte de Caja, Reporte de Clientes con Crédito
Instrumentos y servicios Financieros	Linea de Crédito
Manuales y Libros	Manual de Usuario

3.3 Diseño del Modelo Conceptual

Aquí se muestran conceptos de objetos de el mundo real, los cuales se van a integrar en objetos de conceptos de software, como las clases, y como estos se relacionan en las operaciones que se van llevando a cabo día a día, para poder ir viendo sus funciones y delegando responsabilidades a los objetos de software que van a ser creados.

Desde el inicio del sistema y el ingreso al mismo, se va checando la interacción entre objetos que se va a ir llevando a cabo para poder ir realizando la operación final, que en este caso es generar un reporte de las ventas diarias de las rutas y un concentrado de la semana, con el total de las ventas por vendedor, los totales que se van acumulando durante la semana, para en fases posteriores de el sistema poder realizar otras operaciones que actualmente se realizan de forma manual, como el calculo de la nomina por ejemplo.

3.4 Modelo Conceptual

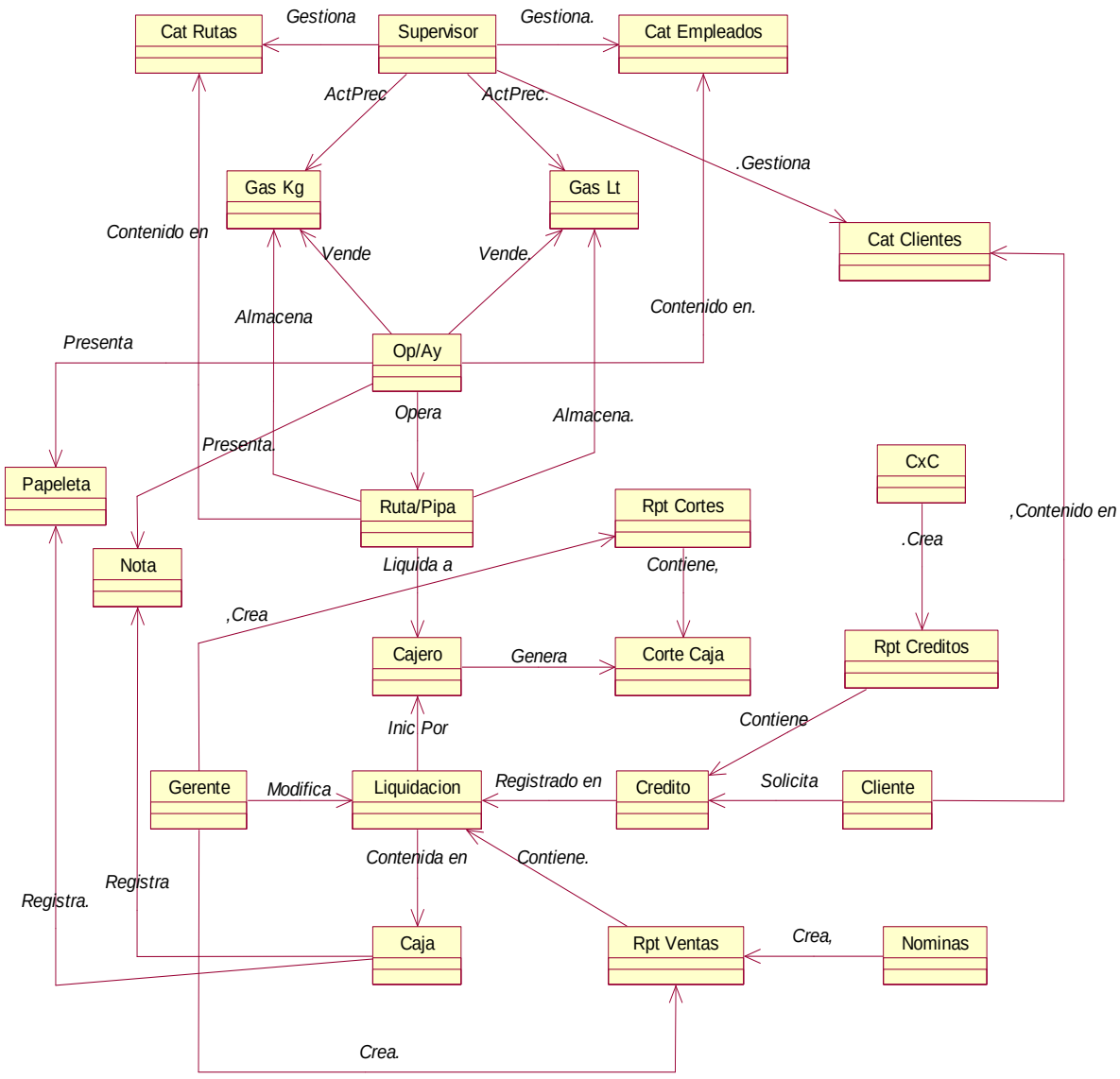


Figura 3.10 Diagrama Conceptual

Capitulo IV

Modelado de los casos de uso y análisis de una posible solución de manera intuitiva

4.1 Introducción

Los casos de uso son un mecanismo para ayudar a mantener los objetivos y requisitos del sistema simple y entendible para todo el personal involucrado dentro del desarrollo del mismo.

A través de la definición de las responsabilidades del sistema como casos de uso es posible identificar que debe hacer el sistema (Requisitos Funcionales) sin decir como lo hará. De hecho la definición de análisis frente al diseño se resume algunas veces como el Qué frente al Cómo.

Los Casos de Uso Completos muestran mas detalles y están estructurados, son útiles para entender en profundidad los objetivos, tareas y requisitos.

4.2 Casos de uso en formato completo con un acercamiento clásico al diseño de diagramas de secuencia y colaboraciones

En este capítulo van a ser desarrollados los casos de uso de manera detallada o competa, una vez que se tiene el diseño conceptual y que han sido identificados dichos casos de uso, posteriormente se muestra el diseño del diagrama de secuencias para cada caso de uso y un primer diagrama de colaboración general para dicho caso de uso, posteriormente en un capítulo mas adelante se establecerá una metodología del diseño de estos diagramas un poco mas formal con la ayuda de los patrones de diseño anteriormente descritos.

UC1: Registrar Ventas de Pipas
UC2: Registrar Ventas de Cilindros
UC3: Registrar Corte de Caja
UC4: Alta de Trabajadores en el Sistema
UC5: Baja de Trabajadores en el Sistema
UC6: Modificación de Trabajadores en el Sistema
UC7: Alta de Unidades en el Sistema
UC8: Baja de Unidades en el Sistema
UC9: Modificación de Unidades en el Sistema
UC10: Alta de Clientes en el Sistema
UC11: Baja de Clientes en el Sistema
UC12: Modificación de Clientes en el Sistema
UC13: Obtención de Reporte para Nómina.
UC14: Obtención de Reporte de Cuentas x Cobrar
UC15: Obtención de Reporte de Ventas
UC16: Obtención de Reporte de Cortes de Caja
UC17: Modificación de Precios de Kilo / Litro de Gas
UC18: Modificación de Ventas Registradas

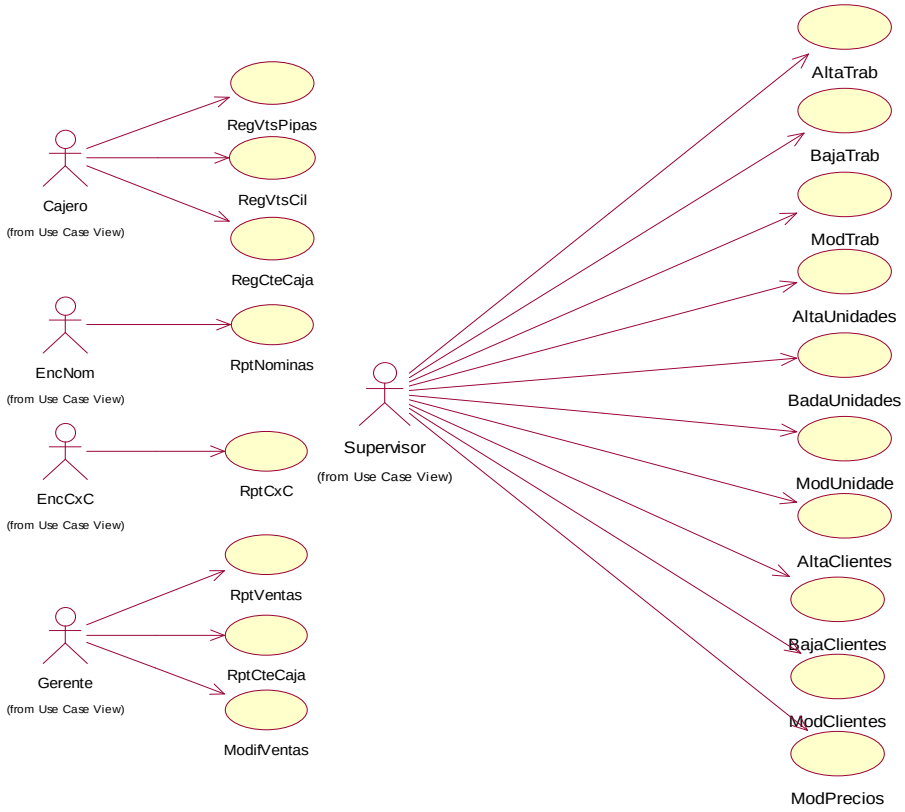


Figura 4.1 Diagrama los Casos de Uso por usuario

Van a ser desarrollados cada uno de estos casos de uso que han sido identificados, ubicando las operaciones principales para dicho caso de uso y agregando diagramas que salen tras un análisis hecho de manera intuitiva.

4.2.1 Caso de uso UC1: Registrar ventas de pipas

- Actor principal: CAJERO

  - ❖ Personal involucrado e Intereses

    - Cajero: Quiere entradas precisas, rápidas y sin errores, ya que si al hacer el corte éste no cuadra las perdidas se deducen de su salario.
    - Vendedor de Pipa: Quiere que las comisiones de sus ventas, así como el precio al que vendió, estén actualizados y sea correcto.
    - Planta: Quiere registrar las ventas con precisión y fluidez, sin perdida de datos a la hora de generar los reportes que van a ser consultados por los gerentes y los supervisores.

- Precondiciones

  - ❖ El cajero se identifica y autentifica.
  - ❖ El vendedor esta en la ventanilla con todas sus notas listas para ser registradas.
  - ❖ En caso de haber registrado notas a crédito, también se actualiza la BD de créditos.

- Post condiciones (Condiciones de Éxito)

  - ❖ Se registra la venta.
  - ❖ Se actualiza la base de datos de las ventas del día.
  - ❖ El sistema queda listo para recibir una nueva liquidación.

- Escenario Principal de éxito (Flujo Principal)

1. El Caso de Uso inicia cuando un operador se encuentra en la caja con sus notas y papeleta de ventas.
2. El cajero inicia el registro de una nueva venta
3. El cajero selecciona el tipo de venta que va a registrar (pipas)
4. El cajero introduce los datos del operador y la unidad (pipa) que le indica la papeleta.
5. El sistema valida los datos del operador y la unidad.
6. El cajero introduce los datos de la nota que va a ser liquidada (monto, cliente, precio, efectivo/crédito).
7. El sistema va actualizando montos de gas y de efectivo a liquidar. El cajero repite los pasos 6 y 7 hasta introducir el total de notas que le entrego el operador.
8. El sistema registra las notas y muestra el total del monto que debe ser liquidado, desglosando los montos en efectivo, crédito y cheque.
9. El cajero indica al vendedor el total a liquidar y solicita el pago.
10. El operador entrega el monto a liquidar, ya sea en efectivo o en documento, como cheques.
11. El cajero introduce en el sistema el monto que le entrego el vendedor, y el tipo de pago que se hizo, efectivo o cheque.
12. El sistema muestra el cambio que debe ser devuelto por el cajero, registra la venta completa, guarda toda la información de la venta.
13. Se actualiza la Base de Datos de Ventas de Pipas y la Base de Datos de Créditos.
14. El sistema indica que los cálculos se han hecho de forma correcta y la información se guardo adecuadamente, finaliza la operación y queda en espera para recibir una nueva venta.
15. El vendedor se va con la convicción de que su venta fue registrada.

- Extensiones (o flujos alternativos)

1. En cualquier momento el sistema falla

1.1 El sistema se reinicia sin llevar a cabo ningún cambio en el registro de la última venta que se estaba capturando.

4 El cajero se equivoca al capturar algún dato de la papeleta o de las notas

4.1 El sistema le indica al cajero que el dato no es correcto, limpia el dato y lo deja listo para que el cajero introduzca nuevamente el dato que corresponde

1-7 El cajero cancela la liquidación

1-7.1 Se finaliza el caso de uso, se limpian los campos y el sistema queda preparado para realizar una nueva liquidación.

Diagrama de Casos de Uso

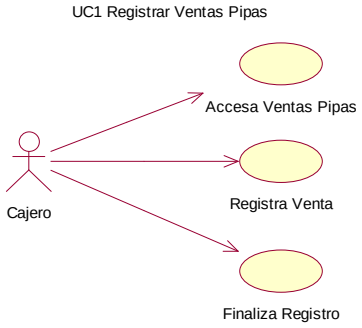


Figura 4.2 Caso de Uso: registrar ventas de pipas

Análisis de una Solución de Manera Intuitiva

Según el diagrama conceptual el cajero realiza la operación de liquidación, la cual esta contenida en caja, entonces en un diseño preeliminar se pueden discernir las principales operaciones, las cuales satisfacen las principales necesidades para llevar a cabo las metas del caso de uso "Registrar Ventas de Pipas", éstas operaciones principales que complementan el diagrama de arriba son:

1.-Iniciar la liquidación.
2.-Elegir el tipo de venta que se va a realizar para llevar a cabo las operaciones correspondientes.
3.-Seleccionar la unidad, el operador y el ayudante que realizaron la venta.
4.-Registrar las notas de las ventas hechas.
5.-Registrar el pago.
6.-Finalizar la operación.

En este caso el cajero es el encargado de realizar la liquidación, siendo él el actor principal del caso de uso, y de una manera intuitiva podríamos inferir que seria él el encargado de ejecutar los eventos de dicho caso de uso, llevando a cabo las principales operaciones, como iniciar las Liquidación, seleccionar el tipo de venta que se esta llevando a cabo, seleccionar la unidad, seleccionar el operador y ayudante, registrar las notas, registrar el pago y finalizar la venta, con lo que se podría asignar a una clase denominada Caja como la que contendría dichas tareas de acuerdo al estudio de la operación que se lleva a cabo en la planta de gas.

Diagrama de Secuencias

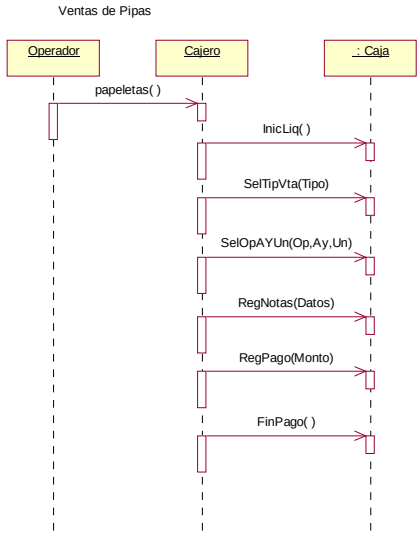


Figura 4.3 Diagrama de Secuencias por intuición del caso de uso No. 1

Por lo que el diagrama de secuencia queda como lo muestra la imagen de arriba. En donde se ve claramente que la clase caja seria la encargada de realizar todas las operaciones, generando así una clase que tiene Baja Cohesión al tener una alta complejidad ya que la clase caja realiza por si sola todas las operaciones para llevar a cabo la tarea.

Un diagrama de colaboración diseñado según el diagrama conceptual y por la intuición que nos da el caso de uso en la vida real que nos indica que el cajero ingresa a caja y la caja genera un nuevo registro de liquidación, la cual contiene su tipo de liquidación, el numero de unidad, un operador y un ayudante, y todos estos datos se ven reflejados en la venta, por lo que siguiendo esta manera de ver las cosas queda un diagrama de colaboración como sigue.

Diagrama de Colaboración

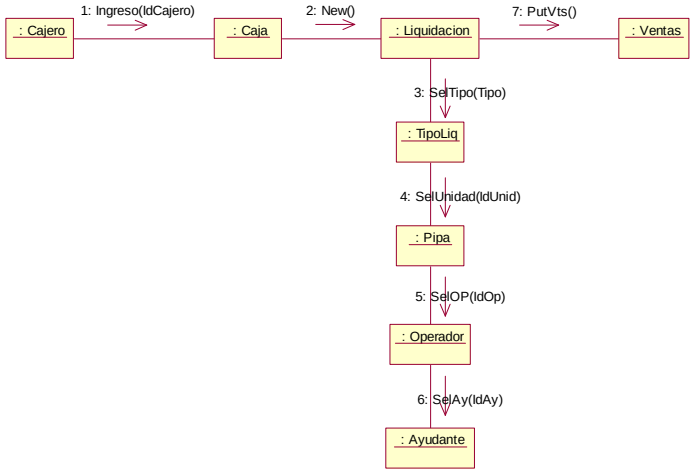


Figura 4.4 Diagrama de colaboración general para el caso de uso 1

Aquí, siguiendo con un diseño de manera intuitiva, se ha logrado visualizar que debe ser otra clase la que lleve a cabo la mayoría de las operaciones, Liquidación, y no la propia caja sea la que las lleve a cabo, para alguien que no tiene mucha experiencia, tal vez no es muy claro.
Mas adelante se va a realizar el diseño utilizando patrones, para observar como se desarrolla este diseño, la delegación de responsabilidades y el por que de estas.

4.2.2 Caso de uso UC2: Registrar ventas de cilindros

- Actor principal: CAJERO

  - ❖ Personal involucrado e Intereses

    - Cajero: Quiere entradas precisas, rápidas y sin errores, ya que si al hacer el corte ente no cuadra las perdidas se deducen de su salario.
    - Vendedor de Cilindros: Quiere que las comisiones de sus ventas, así como el precio al que vendió, estén actualizados y sea correcto.
    - Planta: Quiere registrar las ventas con precisión y fluidez, sin perdida de datos a la hora de generar los reportes que van a ser consultados por los gerentes y los supervisores.

- Precondiciones

  - ❖ El cajero se identifica y autentifica.
  - ❖ El vendedor esta en la ventanilla con su papeleta autorizada y lista para ser registrada.

- Post condiciones (Condiciones de Éxito)

  - ❖ Se registra la Venta.
  - ❖ Se actualiza la base de datos de las ventas del día de cilindros.
  - ❖ El sistema queda listo para recibir una nueva liquidación.

- Escenario Principal de éxito (Flujo Principal)

1. El Caso de Uso inicia cuando un operador se encuentra en la caja con su papeleta de ventas.
2. El cajero inicia el registro de una nueva venta
3. El cajero selecciona el tipo de venta que va a registrar (Cilindros).

4. El cajero introduce los datos del operador y la unidad (Ruta) que le indica la papeleta del repartidor.
5. El sistema valida los datos de la Unidad y el Operador.
6. El cajero introduce los datos de venta que le indica la papeleta (Cantidad Cilindros)
7. El sistema registra la venta y muestra el monto total que debe ser liquidado.
8. El cajero indica al vendedor el total a liquidar y solicita el pago.
9. El operador entrega el monto a liquidar tanto en efectivo como en documentos, tales como cheques.
10. El cajero introduce en el sistema el monto que le entrego el vendedor, y el tipo de pago que se hizo, efectivo o cheque.
11. El sistema muestra el cambio que debe ser devuelto por el cajero, registra la venta completa, guarda toda la información de la venta.
12. Se actualiza la Base de Datos de Ventas de Cilindros y la Base de Datos de Créditos en caso de haber registrado ventas a crédito.
13. El sistema indica que los cálculos se han hecho de forma correcta y la información se guardo adecuadamente, finaliza la operación y queda en espera para recibir una nueva venta.
14. El vendedor se va con la convicción de que su venta fue registrada.

- Extensiones (o Flujos Alternativos)

2. En cualquier momento el sistema falla
2.1 El sistema se reinicia sin llevar a cabo ningún cambio en el registro de la última venta que se estaba capturando.

4 El cajero se equivoca al capturar algún dato de la papeleta o de las notas
4.1 El sistema le indica al cajero que el dato no es correcto, limpia el dato y lo deja listo para que el cajero introduzca nuevamente el dato que corresponde

1-7 El cajero cancela la liquidación
1-7.1 Se finaliza el caso de uso, se limpian los campos y el sistema queda preparado para realizar una nueva liquidación.

Diagrama de Casos de Uso

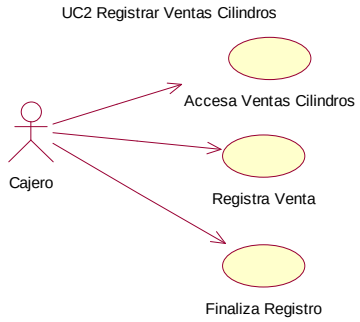


Figura 4.5 Caso de Uso: Registrar Ventas de Cilindros

Continuando con el análisis y completando este diagrama se identifican las operaciones principales:

1.-Iniciar la operación de liquidación.
2.-Elegir el tipo de venta que se va a realizar para llevar a cabo las operaciones correspondientes.
3.-Seleccionar la unidad, el operador y el ayudante que realizaron la venta.
4.-Registrar la papeleta de venta.
5.-Registrar el pago.
6.-Finalizar la operación.

Producto de un análisis intuitivo, se generan los siguientes diagramas con una posible solución.

Diagrama de Secuencias

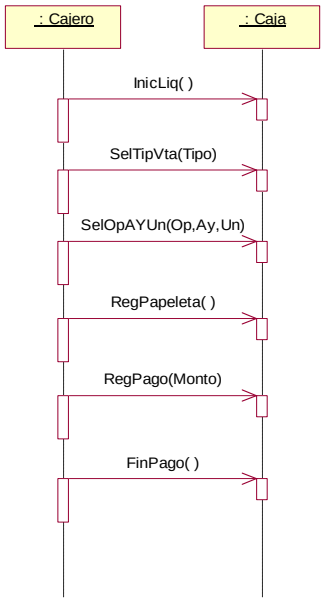


Figura 4.6. Diagrama de Secuencias por intuición del Caso de Uso No. 2

Diagrama de Colaboración

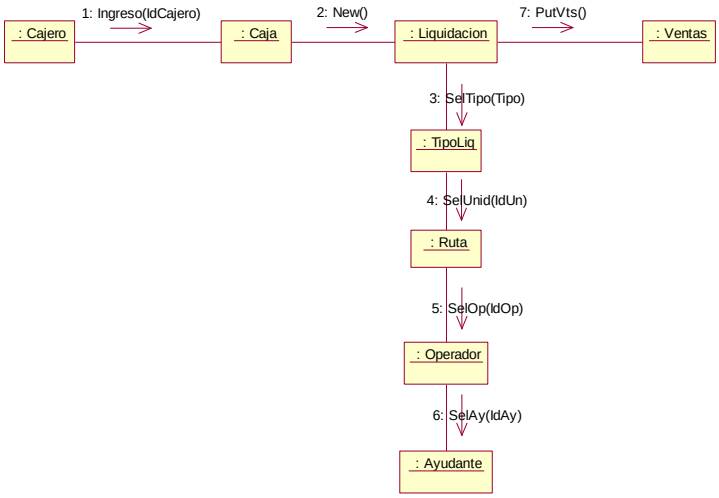


Figura 4.7 Diagrama de Colaboración General para el Caso de Uso 2

De manera análoga con el anterior caso de uso una clase es la que recibe la mayoría de responsabilidades, mas adelante se va a realizar el diseño de una solución utilizando patrones y se hará notar la diferencia.

4.2.3 Caso de uso UC3: Registrar corte de caja

- Actor principal: CAJERO

  - ❖ Personal involucrado e Intereses

    - Cajero: Quiere entradas precisas, rápidas y sin errores, ya que si al hacer el corte ente no cuadra las perdidas se deducen de su salario.
    - Planta: Quiere que los registros de las ventas del día en sus montos totales sean efectivamente los montos que se tienen físicamente al final del día.

- Precondiciones

  - ❖ Que se haya registrado en el sistema el total de las ventas del día, según sea el caso, en cilindros o en pipas

- Post condiciones (Condiciones de Éxito)

  - ❖ Se actualiza la base de datos de las liquidaciones del día.
  - ❖ Se cierra el ciclo de ventas del día actual.
  - ❖ Se abre un nuevo ciclo de ventas para el registro de las liquidaciones del siguiente día.
  - ❖ Se genera la autorización de modificación de registros capturados solo para el día actual de registro y para el día anterior a este.

- Escenario Principal de éxito (Flujo Principal)

1. El Caso de Uso inicia cuando el cajero ingresa a la sección del corte de caja y selecciona de que corte de caja se trata (Pipa o Cilindro).
2. El sistema le muestra al cajero en automático el total de los montos de los distintos tipos de pago aceptados (efectivo, crédito y cheque).
3. El cajero introduce los montos que cuenta físicamente de los diferentes tipos de pago recibidos en el día.
4. El sistema calcula y muestra automáticamente el diferencial de los montos totales tanto por tipo de pago aceptado como la suma total de estos.
5. El cajero guarda los registros o los modifica en caso de que así lo requiera el cajero y finaliza la operación

- Extensiones (o Flujos Alternativos)

1 En cualquier momento el sistema falla
1.1 El sistema se reinicia sin llevar a cabo ningún cambio en el registro del corte del día.

3 El cajero se equivoca al capturar algún dato de los montos de su corte
3.1 El sistema le indica al cajero que el dato no es correcto, limpia el dato y lo deja listo para que el cajero introduzca nuevamente el dato que corresponde

1-5 El cajero cancela la captura de su corte
1-5.1 Se finaliza el caso de uso, se limpian los campos y el sistema queda preparado para ya sea volver a capturar o salir de ahí.

Diagrama de Casos de Uso

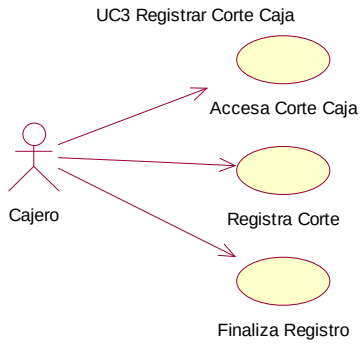


Figura 4.8 Caso de Uso: Registrar Corte de Caja

Se visualizan las operaciones principales para llevar a cabo la tarea del caso de uso:

1.-Iniciar la operación del Corte de Caja
2.-Registrar los montos de los tipos de pago recibidos.
3.-Finalizar la operación.

Diagrama de Secuencias

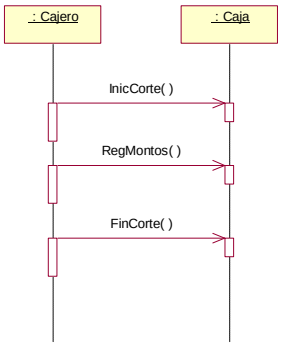


Figura 4.9 Diagrama de Secuencias por intuición del Caso de Uso No. 3

En este diagrama se puede visualizar como caja sigue recibiendo más responsabilidades, con lo que la cohesión va a la baja al incrementar sus tareas y su acoplamiento va creciendo al darle mas relaciones con otras clases.

Diagrama de Colaboración

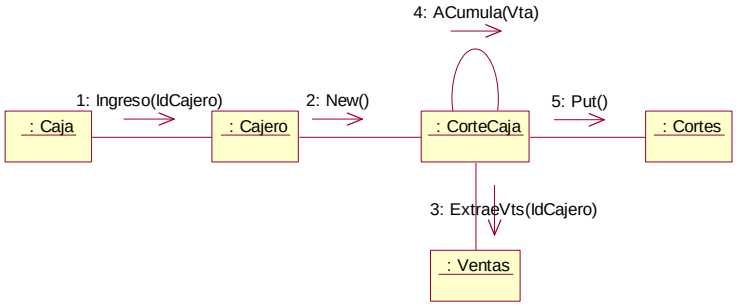


Figura 4.10 Diagrama de Colaboración General para el Caso de Uso 3

En este diagrama de colaboración se observa ya la delegación a otra clase para realizar el corte, pero la delegación de responsabilidades no esta muy clara.

4.2.4 Caso de uso UC4: Alta de trabajadores en el sistema

- Actor principal: SUPERVISOR DE OPERACION

  - ❖ Personal involucrado e Intereses

  - Cajero: Requiere que los registros de los trabajadores estén actualizados en tiempo para que les pueda hacer su carga de gas en el sistema y poderle registrar sus ventas cada día.
  - Trabajador: Quiere que sus datos estén dados de alta en el sistema de forma correcta para que se les pueda asignar una carga y se puedan registrar sus ventas ya que en base a ello se realizara su pago cada semana.

- Supervisor de operaciones: Debe registrar de forma precisa los datos de los trabajadores, ya que el es el responsable de la información que se carga y que de ahí se obtiene de las ventas y de los pagos de los trabajadores.

- Precondiciones

  - El supervisor de operaciones se identifica y autentifica.
  - El supervisor debe tener los datos completos del trabajador listos para ser capturados

- Post condiciones (Condiciones de Éxito)

  - Se actualiza la base de datos de los trabajadores.
  - El registro del trabajador esta listo para registrar las ventas de este.

- Escenario Principal de éxito (Flujo Principal)

1. El caso de uso inicia cuando un nuevo empleado ingresa en la empresa como chofer o ayudante de ventas y el supervisor ingresa a la sección del catalogo de trabajadores.
2. El sistema muestra el catalogo de empleados actual.
3. El supervisor elige la opción de "Nuevo Registro"
4. El sistema limpia los campos y los deja listos para que se introduzcan los datos del nuevo registro que se va a dar de alta
5. El supervisor introduce los datos del nuevo registro y los guarda.
6. El sistema valida los datos, actualiza la base de datos de trabajadores y se posiciona en el primer registro para que sea visualizado, finaliza la operación y manda un mensaje de que el registro se ha guardado correctamente.
7. El nuevo registro puede ser utilizado para realizar una liquidación desde el momento en que ha sido guardado su registro.

- Extensiones (o Flujos Alternativos)

1 En cualquier momento el sistema falla
1.1 El sistema se reinicia sin llevar a cabo ningún cambio en el último registro que se estaba capturando.

5 El supervisor se equivoca al capturar algún tipo de dato del trabajador.
5.1 El sistema le indica al supervisor que el dato no es correcto, limpia el dato y lo deja listo para que el supervisor introduzca nuevamente el dato que corresponde

1-5 El supervisor cancela la alta
1-5.1 Se finaliza el caso de uso, se limpian los campos y el sistema queda mostrando el catalogo de trabajadores.

Diagrama de Casos de Uso

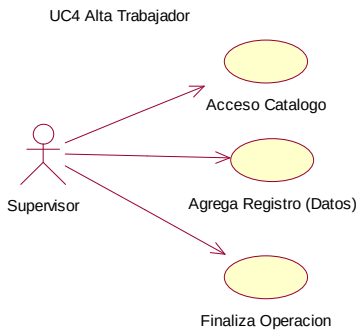


Figura 4.11 Caso de Uso: Dar de Alta un Trabajador en el Sistema

El siguiente paso es identificar las responsabilidades que llevaran a cabo las operaciones correspondientes para realizar el alta de trabajadores.

1.-Iniciar la operación de Alta de Trabajador
2.-Llevar a cabo el registro de los datos
3.-Finalizar la operación de registro.

Estas operaciones son suficientes para realizar el alta de un nuevo registro, y de acuerdo a esto se realiza el diagrama de secuencias

Diagrama de Secuencias

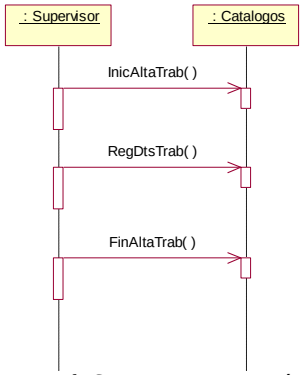


Figura 4.12 Diagrama de Secuencias por intuición del Caso de Uso No. 4

En este diagrama se observa como las operaciones van a ser realizadas por la clase "Catálogos"
De aquí se desprende un diagrama de colaboración el cual muestra la necesidad de asignar la responsabilidad de dar de alta a otra clase.

Diagrama de Colaboración

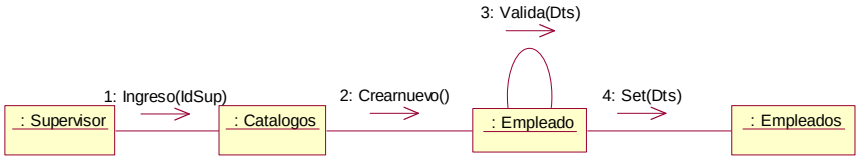


Figura 4.13 Diagrama de Colaboración General para el Caso de Uso 4

4.2.5 Caso de uso UC5: Baja de trabajadores en el sistema

- Actor principal: SUPERVISOR DE OPERACION

  - ❖ Personal involucrado e Intereses

    - Cajero: Requiere que los registros de los trabajadores estén actualizados en tiempo para que les pueda hacer su carga de gas en el sistema y poderle registrar sus ventas cada día.
    - Trabajador: Quiere que sus datos estén dados de alta en el sistema de forma correcta para que se les pueda asignar una carga y se puedan registrar sus ventas ya que en base a ello se realizara su pago cada semana.
    - Supervisor de Operación: Debe registrar de forma precisa los datos de los trabajadores, ya que el es el responsable de la información que se carga y que de ahí se obtiene de las ventas y de los pagos de los trabajadores.

- Precondiciones

- El supervisor de operaciones se identifica y autentifica.
- El supervisor debe tener la certeza de que el trabajador a dar de baja ya no va a ser utilizado en el sistema.

- Post condiciones (Condiciones de Éxito)

  - Se actualiza la base de datos de los trabajadores.
  - Al registro del trabajador que se da de baja ya no se le puede registrar venta alguna.

- Escenario Principal de éxito (Flujo Principal)

1. El Caso de Uso inicia cuando un empleado ha salido de la planta laboral y el supervisor ingresa a la sección del catalogo de trabajadores.
2. El sistema le muestra el catalogo actual de los trabajadores que se tienen dados de alta hasta el momento.
3. El supervisor busca el registro a eliminar y selecciona la opción de “Baja de Registro”
4. El sistema indica que el trabajador va a ser dado de baja y pregunta si es correcta la acción que va a realizar.
5. El supervisor confirma la operación.
6. El sistema modifica el estado del trabajador en la base de datos finaliza la operación e indica que la acción se realizo correctamente con un mensaje.
7. El registro ya no puede ser utilizado para realizar una liquidación desde el momento en que ha sido dado de baja su registro.

- Extensiones (o Flujos Alternativos)

1 En cualquier momento el sistema falla
1.1 El sistema se reinicia sin llevar a cabo ningún cambio en el último registro que se estaba eliminando.

1-4 El supervisor cancela la baja
1-4.1 Se finaliza el caso de uso, se limpian los campos y el sistema queda mostrando el catalogo de trabajadores.

Diagrama de Casos de Uso

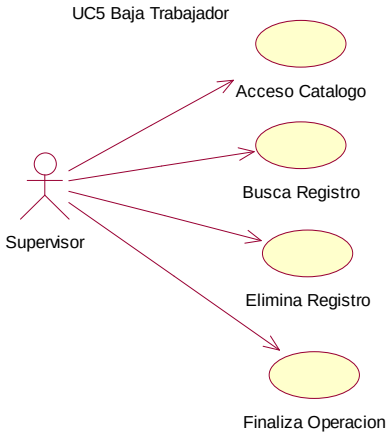


Figura 4.14 Caso de Uso: Dar de Baja un Trabajador en el Sistema

A continuación se listan las operaciones identificadas para llevar a cabo la tares del caso de uso:

1.-Inicial La operación de Baja
2.-Buscar y seleccionar el trabajador que va a ser dado de baja
3.-Confirmar la baja.
4.-Finalizar la operación.

Una vez que tenemos las operaciones procedemos a realizar el diagrama de secuencias y el de colaboración.

Diagrama de Secuencias

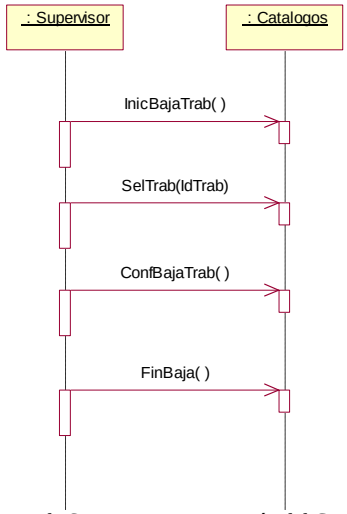


Figura 4.15 Diagrama de Secuencias por intuición del Caso de Uso No. 5

Diagrama de Colaboración

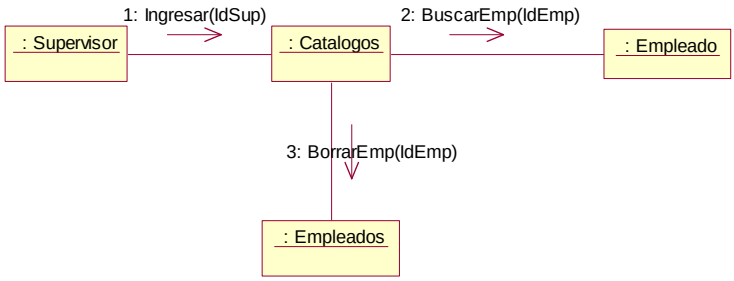


Figura 4.16 Diagrama de Colaboración General para el Caso de Uso 5

Aquí se observa como las clases siguen recibiendo responsabilidades y acoplamiento con otras clases.

4.2.6 Caso de uso UC6: Modificación de trabajadores en el sistema

- Actor principal: SUPERVISOR DE OPERACION

  - ❖ Personal involucrado e Intereses

  - Cajero: Requiere que los registros de los trabajadores estén actualizados en tiempo para que les pueda hacer su registro de ventas del día.
  - Trabajador: Quiere que sus datos estén dados de alta en el sistema de forma correcta para que se les pueda asignar una carga y se puedan registrar sus ventas ya que en base a ello se realizara su pago cada semana.
  - Supervisor de Operaciones: Debe registrar de forma precisa los datos de los trabajadores, ya que el es el responsable de la información que se carga y que de ahí se obtiene de las ventas y para los pagos de los trabajadores.

- Precondiciones

  - ❖ El supervisor de operaciones se identifica y autentifica.
  - ❖ El supervisor debe tener los datos del trabajador que van a ser modificados.

- Post condiciones (Condiciones de Éxito)

  - ❖ Se actualiza la base de datos de los trabajadores con los nuevos datos que se modificaron.

- Escenario Principal de éxito (Flujo Principal)

1. El Caso de Uso inicia cuando surge la necesidad de modificar algún dato de un trabajador y el supervisor ingresa a la sección del catalogo de trabajadores.
2. El sistema le muestra el catalogo actual de los trabajadores que se tienen dados de alta hasta el momento.
3. El supervisor busca el registro que se va a modificar y selecciona la opción de “Modificar Registro”
4. El sistema deja los campos en estado de edición, listos para ser modificados
5. El supervisor cambia los datos que requieran ser modificados y los guarda.
6. El sistema valida y actualiza la base de datos con los nuevos datos, finaliza la operación e indica que los cambios se guardaron de forma correcta.
7. El registro puede ser utilizado para realizar una liquidación en cualquier momento con los nuevos datos.

- Extensiones (o Flujos Alternativos)

1 En cualquier momento el sistema falla
1.1 El sistema se reinicia sin llevar a cabo ningún cambio en el último registro que se estaba modificando.

5 El supervisor se equivoca al capturar algún tipo de dato del trabajador.
5.1 El sistema le indica al supervisor que el dato no es correcto, limpia el dato y lo deja listo para que el supervisor introduzca nuevamente el dato que corresponde

1-5 El supervisor cancela la modificación.
1-5.1 Se finaliza el caso de uso, se limpian los campos y el sistema queda mostrando el catalogo de trabajadores.

Diagrama de Casos de Uso

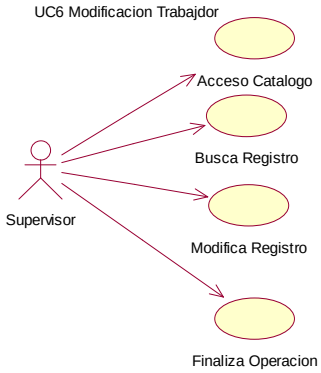


Figura 4.17 Caso de Uso: Modificar Datos de Trabajador

Identificando las operaciones tenemos las siguientes:

1.-Inicial La operación
2.-Buscar y seleccionar el trabajador al que se le van a modificar los datos
3.-Registrar los Datos del Trabajador
4.-Finalizar la operación.

De las operaciones podemos inferir los diagramas de secuencias y de colaboración.

Diagrama de Secuencias

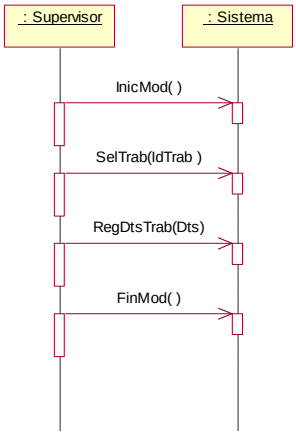


Figura 4.18 Diagrama de Secuencias por intuición del Caso de Uso No. 6

Diagrama de Colaboración

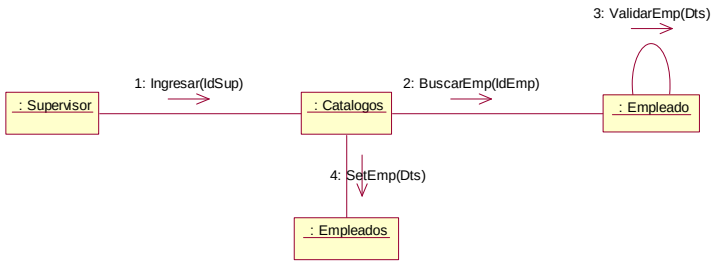


Figura 4.19 Diagrama de Colaboración General para el Caso de Uso 6

Podemos observar como las clases siguen recibiendo responsabilidades, incrementando la complejidad de las clases, su reusabilidad, y mantenimiento.

4.2.7 Caso de uso UC7: Alta de unidades en el sistema

- Actor principal: SUPERVISOR DE OPERACION

  - ❖ Personal involucrado e Intereses

  - Cajero: Requiere que los registros de las unidades estén actualizados en tiempo para que les pueda hacer su carga de gas en el sistema y poder registrar las ventas del operador que se la llevo cada día.
  - Supervisor de operaciones: Debe registrar de forma precisa los datos de los unidades, ya que el es el responsable de la información que se carga y que de ahí se obtiene de las ventas y del rendimiento de dichas unidades.

- Precondiciones

  - ❖ El supervisor de operaciones se identifica y autentifica.
  - ❖ El supervisor debe tener los datos completos de la unidad listos para ser capturados

- Post condiciones (Condiciones de Éxito)

  - ❖ Se actualiza la base de datos de las unidades.
  - ❖ El registro de la unidad esta listo para registrar las ventas de esta.

- Escenario Principal de éxito (Flujo Principal)

1. El Caso de Uso inicia cuando se requiere agregar una nueva unidad para que salga a vender y el supervisor ingresa a la sección del catalogo de unidades a realizar esta operación.
2. El sistema le muestra el catalogo actual de las unidades que se tienen dados de alta hasta el momento.
3. El supervisor selecciona la opción de “Agregar Registro”
4. El sistema limpia los campos y los deja listos para que se introduzcan los datos del nuevo registro que se va a dar de alta
5. El supervisor introduce los campos del nuevo registro y los guarda.
6. El sistema valida los datos, actualiza la base de datos de unidades, se posiciona en el primer registro para que sea visualizado, finaliza la operación y manda un mensaje señalando que el registro se ha guardado correctamente.
7. El nuevo registro puede ser utilizado para realizar una liquidación desde el momento en que ha sido guardado su registro.

- Extensiones (o Flujos Alternativos)

1 En cualquier momento el sistema falla
1.1 El sistema se reinicia sin llevar a cabo ningún cambio en el último registro que se estaba capturando.

5 El supervisor se equivoca al capturar algún tipo de dato de la unidad.
5.1 El sistema le indica al supervisor que el dato no es correcto, limpia el dato y lo deja listo para que el supervisor introduzca nuevamente el dato que corresponde

1-5 El supervisor cancela la alta
1-5.1 Se finaliza el caso de uso, se limpian los campos y el sistema queda mostrando el catalogo de unidades.

Diagrama de Casos de Uso

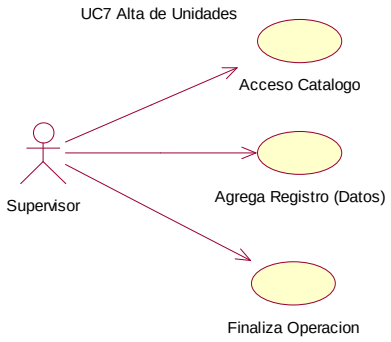


Figura 4.20 Caso de Uso: Alta de Unidades en el Sistema

Listando las operaciones que se identifican tenemos:

1.-iniciar la operación de alta de unidades
2.-Registrar los datos de la nueva unidad.
3.-Finalizar la operación de alta.

De estas operaciones obtenemos los diagramas de secuencia y colaboración.

Diagrama de Secuencias

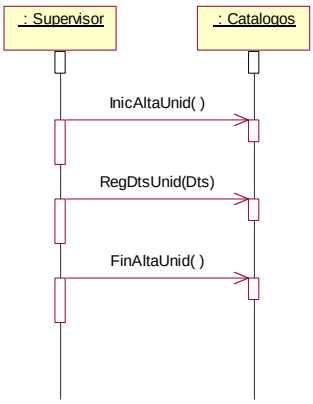


Figura 4.21 Diagrama de Secuencias por intuición del Caso de Uso No. 7

Diagrama de Colaboración

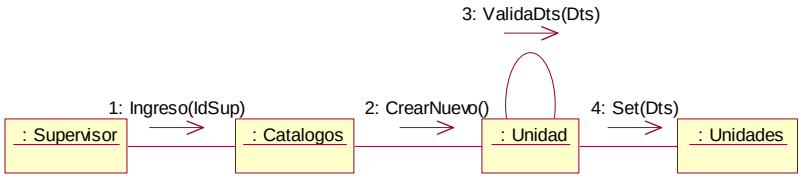


Figura 4.22 Diagrama de Colaboración General para el Caso de Uso 7

4.2.8 Caso de uso UC8: Baja de unidades en el sistema

- Actor principal: SUPERVISOR DE OPERACION

  - ❖ Personal involucrado e Intereses

  - ▪ Cajero: Requiere que los registros de las unidades estén actualizados en tiempo para que les pueda hacer su carga de gas en el sistema y poderle registrar sus ventas cada día.
  - ▪ Supervisor de Operación: Debe registrar de forma precisa los datos de las unidades, ya que el es el responsable de la información que se carga y que de ahí se obtiene de las ventas y de su rendimiento.

- Precondiciones

  - ❖ El supervisor de operaciones se identifica y autentifica.
  - ❖ El supervisor debe tener la certeza de que la unidad a dar de baja ya no va a ser utilizada en el sistema.

- Post condiciones (Condiciones de Éxito)

  - ❖ Se actualiza la base de datos de unidades.
  - ❖ Al registro de unidad que se da de baja ya no se le puede registrar venta alguna.

- Escenario Principal de éxito (Flujo Principal)

1. El Caso de Uso inicia cuando se va a remover una unidad del sistema y el supervisor ingresa a la sección del catalogo de unidades.
2. El sistema le muestra el catalogo actual de las unidades que se tienen dados de alta hasta el momento.
3. El supervisor busca el registro a eliminar selecciona la opción de “Baja de Registro”
4. El sistema indica que la unidad va a ser dada de baja y pregunta si es correcta la acción que va a realizar.
5. El supervisor asiente.
6. El sistema modifica el estado de la unidad en la base de datos, finaliza la operación e indica que la acción se realizo correctamente con un mensaje.
7. El registro ya no puede ser utilizado para realizar una liquidación desde el momento en que ha sido dado de baja su registro.

- Extensiones (o Flujos Alternativos)

1 En cualquier momento el sistema falla
1.1 El sistema se reinicia sin llevar a cabo ningún cambio en el último registro que se estaba eliminando.

1-4 El supervisor cancela la baja
1-4.1 Se finaliza el caso de uso, se limpian los campos y el sistema queda mostrando el catalogo de unidades.

Diagrama de Casos de Uso

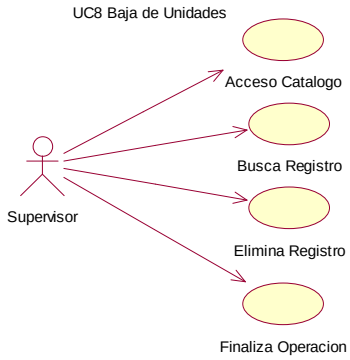


Figura 4.23 Caso de Uso: Dar de Baja Unidades del Sistema

Deducimos las operaciones para llevar a cabo las tareas del caso de uso.

1.-iniciar la operación de baja.
2.-Seleccionar la unidad a ser dada de baja del catalogo.
3.-Confirmar la baja y realizar la operación.
4.-finalizar la operación de baja.

De estas operaciones llevamos a cabo el diagrama de secuencias y colaboración.

Diagrama de Secuencias

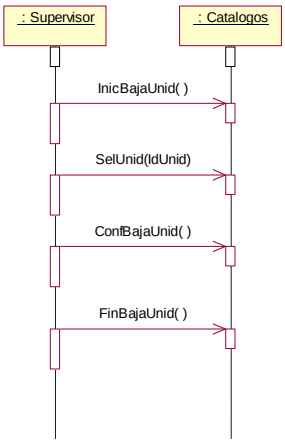


Figura 4.24 Diagrama de Secuencias por intuición del Caso de Uso No. 8

Diagrama de Colaboración

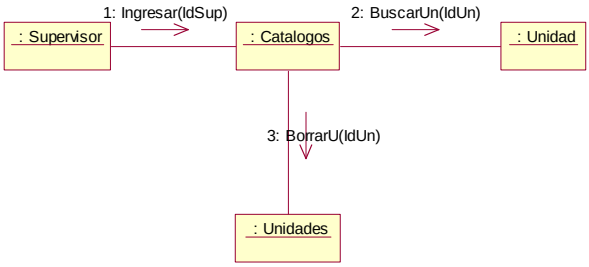


Figura 4.25 Diagrama de Colaboración General para el Caso de Uso 8

4.2.9 Caso de uso UC9: Modificación de unidades en el sistema

- Actor principal: SUPERVISOR DE OPERACION

  - ❖ Personal involucrado e Intereses

  - Cajero: Requiere que los registros de las unidades estén actualizados en tiempo para que les pueda hacer su registro de ventas del día.
  - Supervisor de Operaciones: Debe registrar de forma precisa los datos de las unidades, ya que el es el responsable de la información que se carga y que de ahí se obtiene.

- Precondiciones

  - ❖ El supervisor de operaciones se identifica y autentifica.
  - ❖ El supervisor debe tener los datos de la unidad que van a ser modificados.

- Post condiciones (Condiciones de Éxito)

  - ❖ Se actualiza la base de datos de unidades con los nuevos datos que se modificaron.

- Escenario Principal de éxito (Flujo Principal)

1. El Caso de Uso inicia cuando el supervisor ingresa a la sección del catalogo de unidades para modificar datos de alguna unidad que ya existe en el sistema.
2. El sistema le muestra el catalogo actual de unidades que se tienen dados de alta hasta el momento.
3. El supervisor busca el registro que se va a modificar y selecciona la opción de "Modificación de Registro".
4. El sistema deja los campos en estado de edición, listos para ser modificados
5. El supervisor cambia los datos que requieran ser modificados y los guarda.
6. El sistema valida y actualiza los datos con los nuevos, finaliza la operación e indica que los cambios se guardaron de forma correcta.
7. El registro puede ser utilizado para realizar una liquidación en cualquier momento con los nuevos datos.

- Extensiones (o Flujos Alternativos)

1 En cualquier momento el sistema falla
1.1 El sistema se reinicia sin llevar a cabo ningún cambio en el último registro que se estaba modificando.

5 El supervisor se equivoca al capturar algún tipo de dato de la unidad.
5.1 El sistema le indica al supervisor que el dato no es correcto, limpia el dato y lo deja listo para que el supervisor introduzca nuevamente el dato que corresponde

1-5 El supervisor cancela la modificación.
1-5.1 Se finaliza el caso de uso, se limpian los campos y el sistema queda mostrando el catalogo de unidades.

Diagrama de Casos de Uso

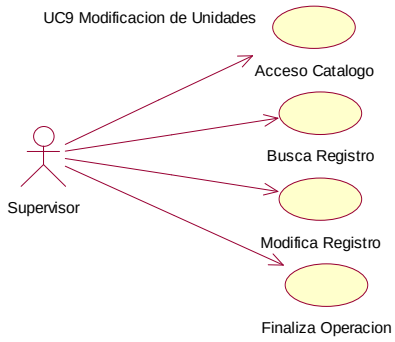


Figura 4.26 Caso de Uso: Modificación de Unidades

Identificamos y listamos las operaciones con las que llevaremos a cabo las tareas del caso de uso:

1.-Iniciar operación de modificación.
2.-Buscar y seleccionar la unidad.
3.-Registrar los nuevos datos de la unidad.
4.-Finalizar esta operación.

Una vez que tenemos los pasos a seguir para llevar a cabo la operación realizamos los diagramas de secuencia y de colaboración.

Diagrama de Secuencias

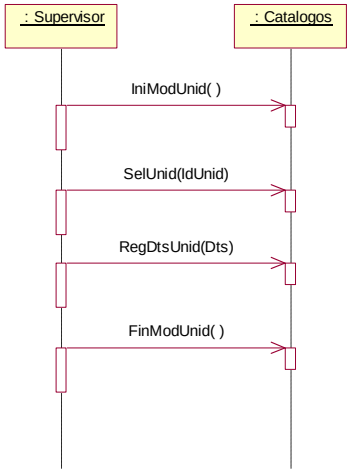


Figura 4.27 Diagrama de Secuencias por intuición del Caso de Uso No. 9

Diagrama de Colaboración

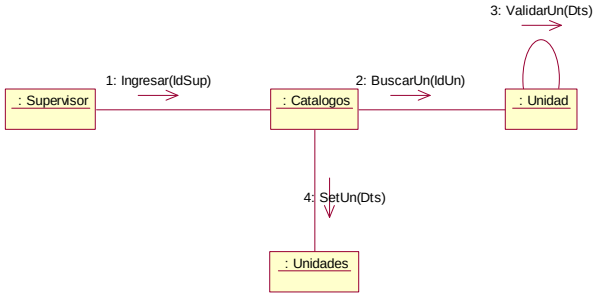


Figura 4.28 Diagrama de Colaboración General para el Caso de Uso 9

4.2.10 Caso de uso UC10: Alta de clientes en el sistema

- Actor principal: SUPERVISOR DE OPERACION

  - ❖ Personal involucrado e Intereses

  - ▪ Cajero: Requiere que los registros de los clientes estén actualizados en tiempo para que en caso de haber solicitado algún crédito este pueda ser registrado de manera rápida y eficiente.
  - ▪ Supervisor de operaciones: Debe registrar de forma precisa los datos de los clientes que han sido autorizado a recibir créditos.

- Precondiciones

  - ❖ El supervisor de operaciones se identifica y autentifica.
  - ❖ El supervisor debe tener los datos completos del cliente y listos para ser capturados

- Post condiciones (Condiciones de Éxito)

  - ❖ Se actualiza la base de datos de los clientes.
  - ❖ El registro del cliente esta listo para recibir los créditos que se le han dado.

- Escenario Principal de éxito (Flujo Principal)

1. El Caso de Uso inicia cuando se requiere agregar un nuevo cliente que ha sido autorizado para recibir ventas a crédito y el supervisor ingresa al catalogo de clientes a realizar esta operación.
2. El sistema muestra el catalogo actual de clientes que se tienen dados de alta hasta el momento.
3. El supervisor selecciona la opción de “Agregar Registro”
4. El sistema limpia los campos y los deja listos para que se introduzcan los datos del nuevo registro que se va a dar de alta
5. El supervisor introduce los campos del nuevo registro y los guarda.
6. El sistema valida los datos, actualiza la base de datos de clientes, se posiciona en el primer registro para que sea visualizado, finaliza la operación y manda un mensaje señalando que el registro se ha guardado correctamente.
7. El nuevo registro puede ser utilizado para realizar un cargo de crédito desde el momento en que ha sido guardado su registro.

- Extensiones (o Flujos Alternativos)

1 En cualquier momento el sistema falla
1.1 El sistema se reinicia sin llevar a cabo ningún cambio en el último registro que se estaba capturando.

5 El supervisor se equivoca al capturar algún tipo de dato de la unidad.
5.1 El sistema le indica al supervisor que el dato no es correcto, limpia el dato y lo deja listo para que el supervisor introduzca nuevamente el dato que corresponde

1-5 El supervisor cancela la alta
1-5.1 Se finaliza el caso de uso, se limpian los campos y el sistema queda mostrando el catalogo de unidades.

Diagrama de Casos de Uso

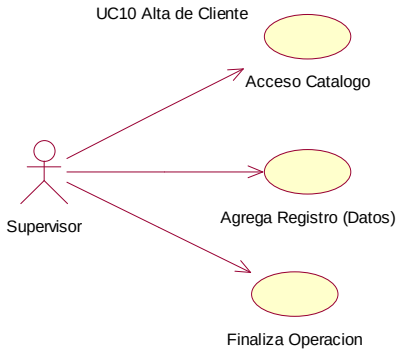


Figura 4.29 Caso de Uso: Dar de Alta un Cliente en el Sistema

Identificamos y listamos los pasos para llevar a cabo la tarea del caso de uso.

1.-Iniciar el alta de un nuevo cliente.
2.-Registrar los datos del nuevo cliente.
3.-Finalizar la operación.

De aquí partimos para obtener los diagramas de secuencia y de colaboración.

Diagrama de Secuencias

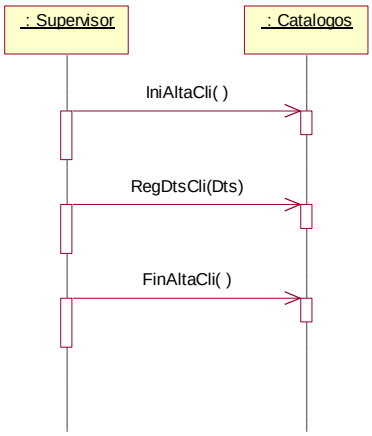


Figura 4.30 Diagrama de Secuencias por intuición del Caso de Uso No. 10

Diagrama de Colaboración

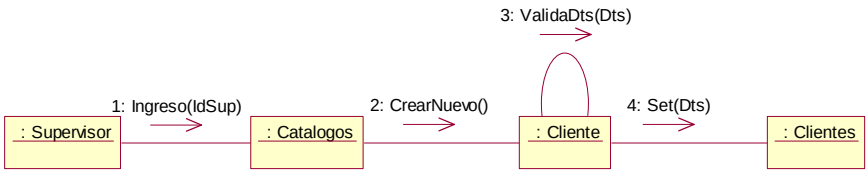


Figura 4.31 Diagrama de Colaboración General para el Caso de Uso 10

4.2.11 Caso de uso UC11: Baja de clientes en el sistema

- Actor principal: SUPERVISOR DE OPERACION

  - ❖ Personal involucrado e Intereses

    - Cajero: Requiere que los registros de los clientes estén actualizados en tiempo para que les pueda hacer el cargo del monto que le fue vendido a crédito.
    - Supervisor de Operación: Debe registrar de forma precisa los datos de los clientes.

- Precondiciones

  - ❖ El supervisor de operaciones se identifica y autentifica.
  - ❖ El supervisor debe tener la certeza de que el cliente a dar de baja ya no va a ser requerido en el sistema.

- Post condiciones (Condiciones de Éxito)

  - ❖ Se actualiza la base de datos de clientes.
  - ❖ Al registro del cliente que se da de baja ya no se le puede registrar venta alguna.

- Escenario Principal de éxito (Flujo Principal)

1. El Caso de Uso inicia cuando se va a remover un cliente del sistema y el supervisor ingresa a la sección del catalogo de clientes.
2. El sistema le muestra el catalogo actual de los clientes que se tienen dados de alta hasta el momento.
3. El supervisor busca el registro a eliminar selecciona la opción de “Baja de Registro”
4. El sistema indica que el cliente va a ser dado de baja y pregunta si es correcta la acción que va a realizar.
5. El supervisor asiente.
6. El sistema modifica el estado del cliente en la base de datos, finaliza la operación e indica que la acción se realizo correctamente con un mensaje.
7. El registro ya no puede ser utilizado para realizar una cargo desde el momento en que ha sido dado de baja su registro.

- Extensiones (o Flujos Alternativos)

1 En cualquier momento el sistema falla
1.1 El sistema se reinicia sin llevar a cabo ningún cambio en el último registro que se estaba eliminando.

1-4 El supervisor cancela la baja
1-4.1 Se finaliza el caso de uso, se limpian los campos y el sistema queda mostrando el catalogo de clientes.

Diagrama de Casos de Uso

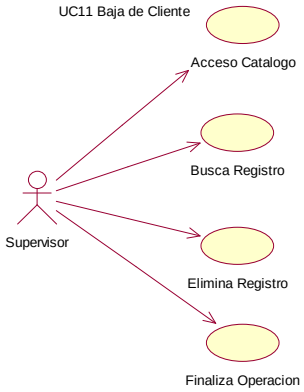


Figura 4.32 Caso de Uso: Dar de Baja un Cliente del Sistema

Listando las tareas a realizar para realizar la baja tenemos:

1.-Iniciar la operación de baja.
2.-Buscar el registro del cliente a ser dado de baja del catalogo.
3.-Confirmar la baja del registro y realizar la operación
4.-Finalizar la tarea de baja.

De aquí obtenemos los diagramas de secuencia y colaboración.

Diagrama de Secuencias

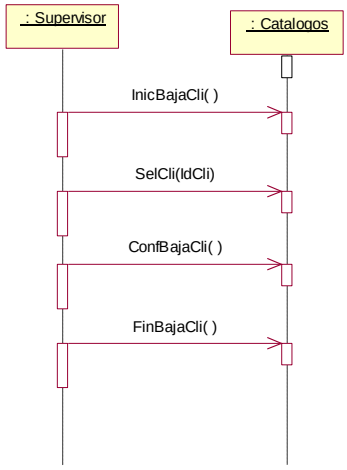


Figura 4.33 Diagrama de Secuencias por intuición del Caso de Uso No. 11

Diagrama de Colaboración

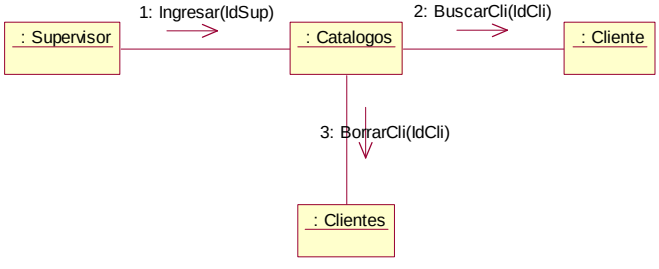


Figura 4.34 Diagrama de Colaboración General para el Caso de Uso 11

4.2.12 Caso de uso UC12: Modificación de clientes en el sistema

- Actor principal: SUPERVISOR DE OPERACION

  - ❖ Personal involucrado e Intereses

    - Cajero: Requiere que los registros de los clientes estén actualizados en tiempo para que les pueda hacer el cargo de sus créditos.
    - Supervisor de Operaciones: Debe registrar de forma precisa los datos de los clientes que han sido autorizados a recibir créditos.

- Precondiciones

  - ❖ El supervisor de operaciones se identifica y autentifica.
  - ❖ El supervisor debe tener los datos del cliente que van a ser modificados.

- Post condiciones (Condiciones de Éxito)

  - ❖ Se actualiza la base de datos de clientes con los nuevos datos que se modificaron.

- Escenario Principal de éxito (Flujo Principal)

1. El Caso de Uso inicia cuando el supervisor ingresa a la sección del catalogo de clientes para modificar datos de algún cliente que ya existe en el sistema.
2. El sistema le muestra el catalogo actual de clientes que se tienen dados de alta hasta el momento.
3. El supervisor busca el registro que se va a modificar y selecciona la opción de "Modificación de Registro".
4. El sistema deja los campos en estado de edición, listos para ser modificados
5. El supervisor cambia los datos que requieran ser modificados y los guarda.
6. El sistema valida y actualiza los datos con los nuevos, finaliza la operación e indica que los cambios se guardaron de forma correcta.
7. El registro puede ser utilizado para realizar un cargo de crédito en cualquier momento con los nuevos datos.

- Extensiones (o Flujos Alternativos)

1 En cualquier momento el sistema falla
1.1 El sistema se reinicia sin llevar a cabo ningún cambio en el último registro que se estaba modificando.

5 El supervisor se equivoca al capturar algún tipo de dato del cliente.
5.1 El sistema le indica al supervisor que el dato no es correcto, limpia el dato y lo deja listo para que el supervisor introduzca nuevamente el dato que corresponde

1-5 El supervisor cancela la modificación.
1-5.1 Se finaliza el caso de uso, se limpian los campos y el sistema queda mostrando el catalogo de unidades.

Diagrama de Casos de Uso

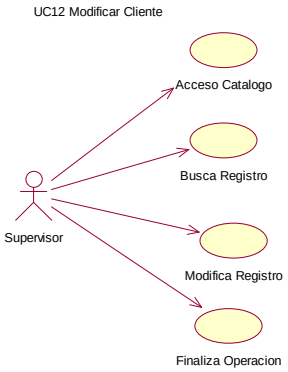


Figura 4.35 Caso de Uso: Modificación de Clientes

Ahora listamos las tareas a llevar a cabo para realizar la operación de modificar el registro de cliente.

1.-Iniciar la operación de modificación.
2.-Buscar y seleccionar el registro del cliente a ser modificado.
3.-Registrar los nuevos datos del cliente.
4.-Finalizar la operación de modificación de datos.

De aquí llevamos a cabo los diagramas de secuencia y colaboración.

Diagrama de Secuencias

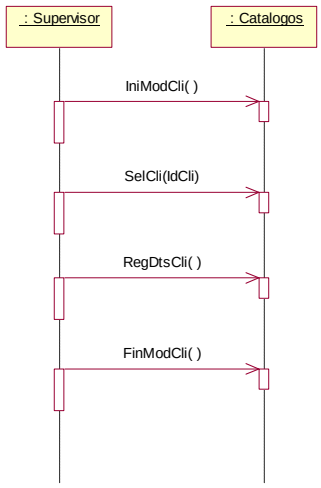


Figura 4.36 Diagrama de Secuencias por intuición del Caso de Uso No. 12

Diagrama de Colaboración

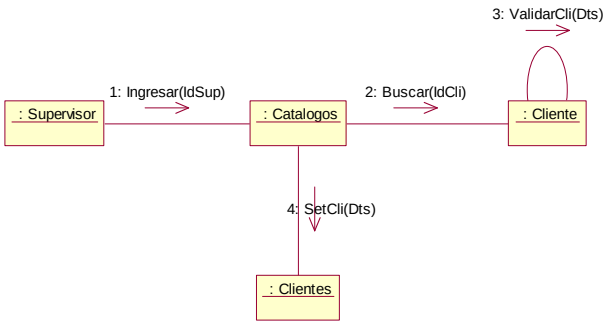


Figura 4.37 Diagrama de Colaboración General para el Caso de Uso 12

4.2.13 Caso de uso UC13: Obtención de reporte para nómina

- Actor principal: ENCARGADO DE NÓMINA

  - Personal involucrado e Intereses

    - Encargado de Nómina: Requiere que los datos obtenidos en el reporte sean correctos, para que en base a ellos se pueda realizar el calculo correcto de la nomina de los trabajadores.
    - Trabajador: Quiere que los datos de sus ventas generados en el reporte sean correctos, pues en base a estos datos se realizara el calculo de su pago en la nomina.

- Precondiciones

  - El Encargado de Nomina se identifica y autentifica.
  - Los datos de la semana deben estar capturados en el sistema

- Post condiciones (Condiciones de Éxito)

  - Se muestran en pantalla los datos de las ventas en el periodo de fechas elegido.
  - Se permite la generación del reporte mostrado en pantalla hacia un archivo en Excel, hacia la impresora o hacia un archivo de texto.

- Escenario Principal de éxito (Flujo Principal)

1. El Caso de Uso inicia cuando el encargado de la nomina requiere obtener este reporte e ingresa a la sección de reporte para nomina.
2. El sistema permite seleccionar el filtro del rango de fechas para que sea mostrada la información de las ventas en ese periodo de fechas dado.
3. El usuario introduce una fecha inicial y una fecha final, las cuales van a ser el periodo en el cual va a ser mostrado el reporte y acepta.
4. El sistema aplica el filtro y muestra en pantalla los datos de los trabajadores con sus ventas por día y el concentrado del total de las ventas en el periodo elegido.
5. El usuario puede imprimir el reporte o guardarlo en un archivo ya sea Excel o de texto para que lo use de la manera que lo requiera.
6. El sistema manda a imprimir los datos del reporte y finaliza la operación

- Extensiones (o Flujos Alternativos)

3. El encargado de nominas introduce una fecha final menor a la fecha inicial.
3.1 El sistema indica que el error y limpia los campos de fechas para que sean introducidos nuevamente.

1-5 El encargado de nominas cancela la ejecución del reporte.
1-5.1 Se finaliza el caso de uso y el sistema queda en la pantalla principal.

Diagrama de Casos de Uso

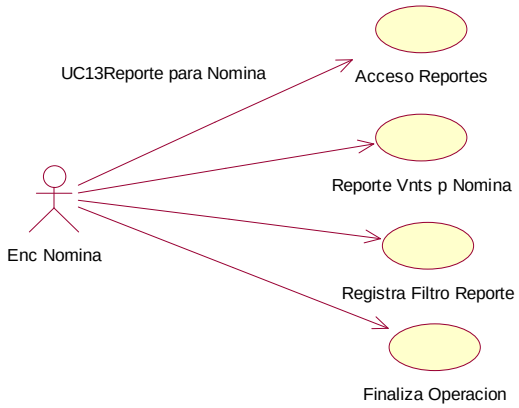


Figura 4.38 Caso de Uso: Reporte de Ventas para Nómina

Listamos las tareas a ser realizadas para llevar a cabo el caso de uso

1.-Iniciar la operación de llevar a cabo el reporte.
2.-Seleccionar el filtro a ser aplicado para el reporte
3.-Imprimir el reporte en pantalla.

De aquí se llevan a cabo los diagramas de secuencia y colaboración para estas tareas.

Diagrama de Secuencias

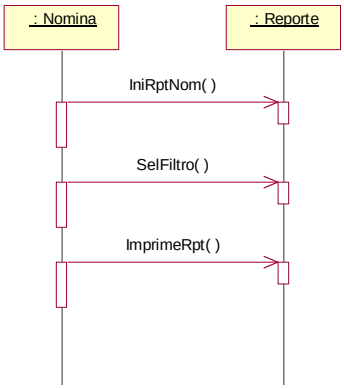


Figura 4.39 Diagrama de Secuencias por intuición del Caso de Uso No. 13

Diagrama de Colaboración

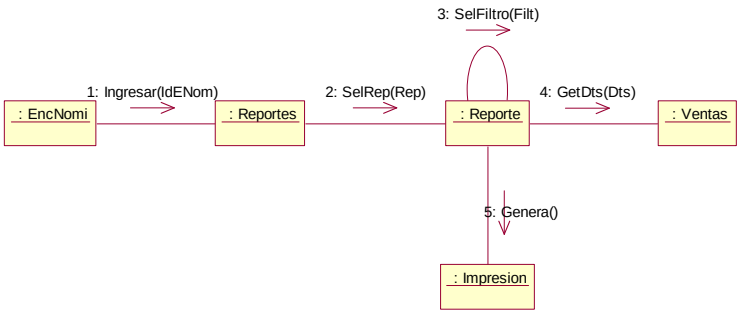


Figura 4.40 Diagrama de Colaboración General para el Caso de Uso 13

4.2.14 Caso de uso UC14: Obtención de reporte de cuentas x cobrar

- Actor principal: ENCARGADO DE CUENTAS X COBRAR

  - Personal involucrado e Intereses

    - Encargado de Cuentas X Cobrar: Requiere que los datos obtenidos en el reporte sean correctos, para que en base a ellos se puedan tomar las decisiones de los clientes a los que se les va a cobrar y a los que ya no se les puede surtir.

- Precondiciones

  - El Encargado de C x C se identifica y autentifica.
  - Los datos de las ventas a crédito deben estar capturados en el sistema.

- Post condiciones (Condiciones de Éxito)

  - Se muestran en pantalla los datos de las ventas a crédito en el periodo de fechas elegido.
  - Se permite la generación del reporte mostrado en pantalla hacia un archivo en Excel, hacia la impresora o hacia un archivo de texto.

- Escenario Principal de éxito (Flujo Principal)

1. El Caso de Uso inicia cuando el encargado de C x C ingresa a la sección de reporte de C x C a obtener dicho reporte.
2. El sistema permite que se seleccione el filtro del rango de fechas para que le sea mostrada la información de las ventas a crédito en ese periodo de fechas dado.
3. El usuario introduce una fecha inicial y una fecha final, las cuales van a ser el periodo en el cual va a ser mostrado el reporte y acepta.
4. El sistema aplica el filtro y muestra en pantalla los datos de los clientes con sus compras por día y el concentrado del total de las compras a crédito en el periodo elegido.
5. El usuario puede imprimir el reporte o guardarlo en un archivo ya sea Excel o de texto para que lo use de la manera que lo requiera.
6. El sistema manda a imprimir los datos del reporte y finaliza la operación

- Extensiones (o Flujos Alternativos)

3. El encargado de C x C introduce una fecha final menor a la fecha inicial.
3.1 El sistema indica que el error y limpia los campos de fechas para que sean introducidos nuevamente.

1-5 El encargado de C x C cancela la ejecución del reporte.
1-5.1 Se finaliza el caso de uso y el sistema queda en la pantalla principal.

Diagrama de Casos de Uso

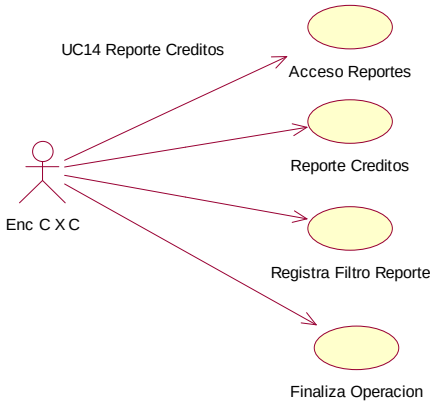


Figura 4.41 Caso de Uso: Reporte de Ventas a Crédito

Listamos las tareas a llevarse a cabo para realizar el reporte que se pide en este caso de uso.

1.-Iniciar la operación de reporte de créditos.
2.-Seleccionar el filtro a ser aplicado al reporte.
3.-Imprimir el reporte con los filtros solicitados.

De aquí se realizan los diagramas de secuencia y colaboración.

Diagrama de Secuencias

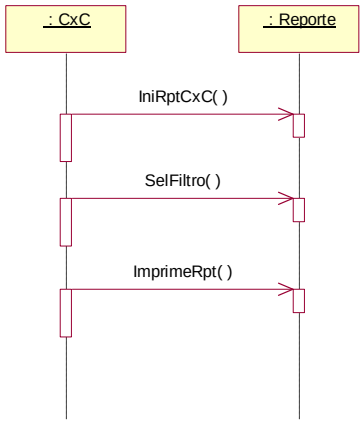


Figura 4.42 Diagrama de Secuencias por intuición del Caso de Uso No. 14

Diagrama de Colaboración

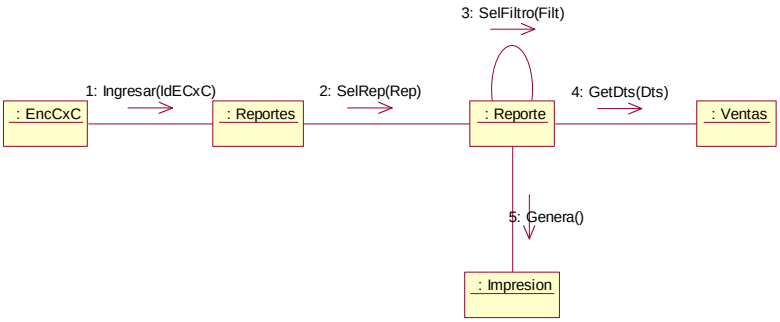


Figura 4.43 Diagrama de Colaboración General para el Caso de Uso 14

4.2.15 Caso de uso UC15: Obtención de reporte de ventas

- Actor principal: Gerente Operativo

  - ❖ Personal involucrado e Intereses

  - ▪ Gerente Operativo: Requiere que los datos obtenidos en el reporte sean correctos, para que en base a ellos se puedan tomar las decisiones con respecto al rendimiento de los operadores que manejan una unidad.

- Precondiciones

  - ❖ El Gerente se identifica y autentifica.
  - ❖ Los datos de las ventas deben estar capturadas en el sistema.

- Post condiciones (Condiciones de Éxito)

  - ❖ Se muestran en pantalla los datos de las ventas por empleado en el periodo de fechas elegido.
  - ❖ Se permite la generación del reporte mostrado en pantalla hacia un archivo en Excel, hacia la impresora o hacia un archivo de texto.

- Escenario Principal de éxito (Flujo Principal)

1. El Caso de uso inicia cuando el gerente ingresa a la sección de reporte de Ventas.
2. El sistema permite que se seleccione el filtro del rango de fechas y el rango de empleados para que le sea mostrada la información de las ventas en ese periodo de fechas dado.
3. El gerente introduce una fecha inicial, una fecha final y un rango de trabajadores como filtros para mostrar la información.
4. El sistema aplica el filtro y muestra en pantalla los datos de los operadores con sus ventas por día y el concentrado del total de las ventas en el periodo elegido.
5. El usuario puede imprimir el reporte o guardarlo en un archivo ya sea Excel o de texto para que lo use de la manera que lo requiera.
6. El sistema manda a imprimir los datos del reporte y finaliza la operación

- Extensiones (o Flujos Alternativos)

3. El gerente introduce una fecha final menor a la fecha inicial.
3.1 El sistema indica que el error y limpia los campos de fechas para que sean introducidos nuevamente.

1-5 El gerente cancela la ejecución del reporte.
1-5.1 Se finaliza el caso de uso y el sistema queda en la pantalla principal.

Diagrama de Casos de Uso

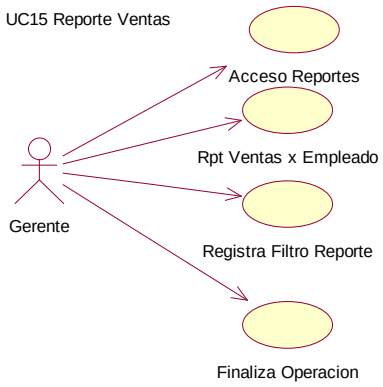


Figura 4.44 Caso de Uso: Reporte de Ventas en General

De aquí se listan las operaciones para llevar a cabo las operaciones del caso de uso

1.-Iniciar la operación de reporte de ventas.
2.-Seleccionar el filtro a ser aplicado para el reporte.
3.-Imprimir el reporte con los filtros seleccionados.

De estas operaciones pasamos a los diagramas de secuencia y de colaboración.

Diagrama de Secuencias

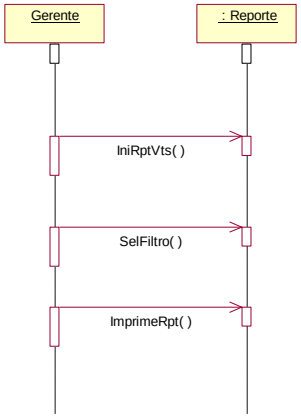


Figura 4.45 Diagrama de Secuencias por intuición del Caso de Uso No. 15

Diagrama de Colaboración

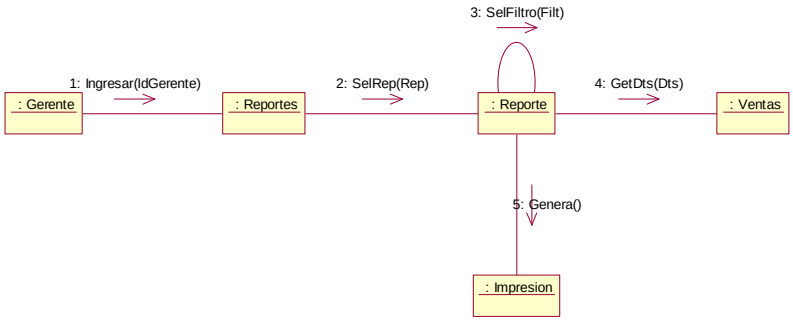


Figura 4.46 Diagrama de Colaboración General para el Caso de Uso 15

4.2.16 Caso de uso UC16: Obtención de reporte de cortes de caja

- Actor principal: Gerente

  - Personal involucrado e Intereses

    - Gerente Operativo: Requiere que los datos obtenidos en el reporte sean correctos, para que en base a ellos se puedan tomar las decisiones con respecto al rendimiento de los operadores que manejan una unidad.

- Precondiciones

  - El Gerente se identifica y autentifica.
  - Los datos de los cortes deben estar capturadas en el sistema.

- Post condiciones (Condiciones de Éxito)

  - Se muestran en pantalla los datos de los cortes de caja en el periodo de fechas elegido.
  - Se permite la generación del reporte mostrado en pantalla hacia un archivo en Excel, hacia la impresora o hacia un archivo de texto.

- Escenario Principal de éxito (Flujo Principal)

1. El Caso de Uso inicia cuando algún gerente ingresa a la sección de reporte de Cortes de Caja para generar el mismo.
2. El sistema permite la selección del filtro del rango de fechas y el rango de empleados para que le sea mostrada la información de los cortes en el periodo de fechas dado.
3. El gerente introduce una fecha inicial, una fecha final como filtros para mostrar la información.
4. El sistema aplica el filtro y muestra en pantalla los datos de los cortes y el concentrado de estos en el periodo elegido.
5. El usuario puede imprimir el reporte o guardarlo en un archivo ya sea Excel o de texto para que lo use de la manera que lo requiera.
6. El sistema manda a imprimir los datos del reporte y finaliza la operación

- Extensiones (o Flujos Alternativos)

3. El gerente introduce una fecha final menor a la fecha inicial.
3.1 El sistema indica que el error y limpia los campos de fechas para que sean introducidos nuevamente.

1-5 El gerente cancela la ejecución del reporte.
1-5.1 Se finaliza el caso de uso y el sistema queda en la pantalla principal.

Diagrama de Casos de Uso

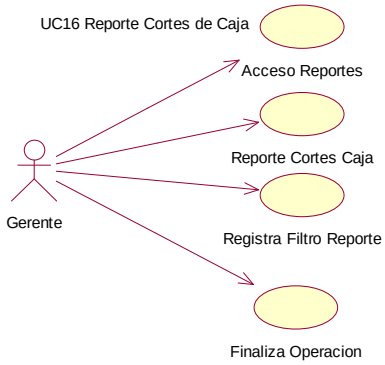


Figura 4.47 Caso de Uso: Reporte de Cortes de Caja

A continuación se listan las operaciones a seguir para realizar la tarea del caso de uso.

1.- Iniciar la operación del reporte de los cortes de caja.
2.-Seleccionar el filtro a ser aplicado para este reporte.
3.-Imprimir el reporte con los filtros aplicados.

De aquí se infiere la realización de los diagramas de secuencia y de colaboración.

Diagrama de Secuencias

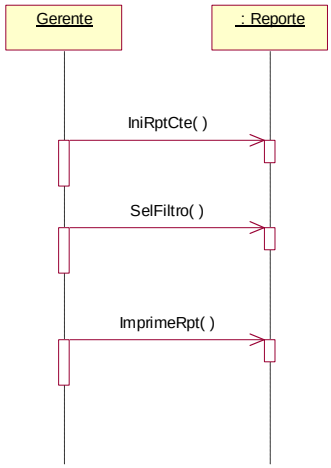


Figura 4.48 Diagrama de Secuencias por intuición del Caso de Uso No. 16

Diagrama de Colaboración

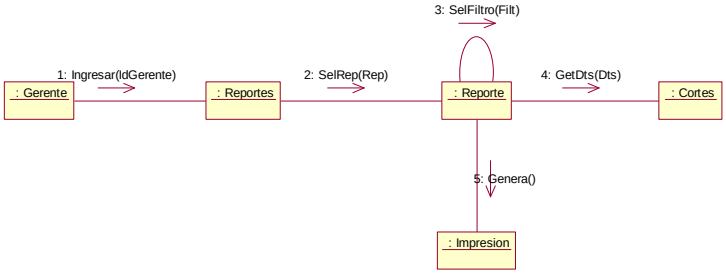


Figura 4.49 Diagrama de Colaboración General para el Caso de Uso 16

4.2.17 Caso de uso UC17: Modificación de precios de Kg. / Lt. de Gas

- Actor principal: SUPERVISOR DE OPERACION

  - ❖ Personal involucrado e Intereses

    - Cajero: Requiere que los precios estén actualizados en tiempo para que los datos de las ventas sean correctos, de acuerdo a lo que vendieron los operadores.
    - Supervisor de Operaciones: Debe registrar de forma precisa los nuevos precios cada mes, de acuerdo a los lineamientos que marca la Secretaría de Energía.
    - Gerente: Requiere que los precios sean correctos, para que la venta se realice de manera correcta y no registren perdidas por precios menores a los que deben ser.

- Precondiciones

  - ❖ El supervisor de operaciones se identifica y autentifica.
  - ❖ El supervisor debe tener los nuevos precios del Kilo y del litro de Gas.
  - ❖

- Post condiciones (Condiciones de Éxito)

  - ❖ Se actualiza el registro de los precios del Kilo y del Litro.

- Escenario Principal de éxito (Flujo Principal)

1. El Caso de Uso inicia cuando el supervisor ingresa a la sección de precios del mes para actualizarlos.
2. El sistema muestra los precios que se tienen registrados.
3. El supervisor selecciona la opción de “Modificar Registro”.
4. El sistema deja los campos en estado de edición, listos para ser modificados
5. El supervisor cambia los datos que requieran ser modificados y los guarda.
6. El sistema actualiza la base de datos con los nuevos precios finaliza la operación e indica que los cambios se guardaron de forma correcta.
7. El registro puede ser utilizado para realizar una liquidación en cualquier momento con los nuevos datos.

- Extensiones (o Flujos Alternativos)

1 En cualquier momento el sistema falla
1.1 El sistema se reinicia sin llevar a cabo ningún cambio en el registro de precios.

5 El supervisor se equivoca al capturar el tipo del dato del precio.
5.1 El sistema le indica al supervisor que el dato no es correcto, limpia el dato y lo deja listo para que el supervisor introduzca nuevamente el dato que corresponde

1-5 El supervisor cancela la modificación.
1-5.1 Se finaliza el caso de uso, se limpian los campos y el sistema queda mostrando los precios dados hasta el momento.

Diagrama de Casos de Uso

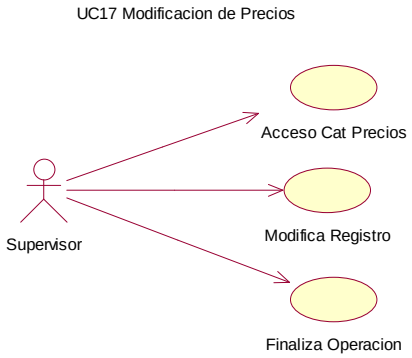


Figura 4.50 Caso de Uso: Modificar Precios

De aquí obtenemos las operaciones a seguir para realizar la tarea que encierra este caso de uso.

1.-Iniciar lo operación de modificación de precios.
2.-Registrar los nuevos precios de los productos gas kilo y gas litro.
3.-finalizar la operación de modificación de precios.

A continuación se muestra los diagramas de secuencia y colaboración inferidos de estas tareas.

Diagrama de Secuencias

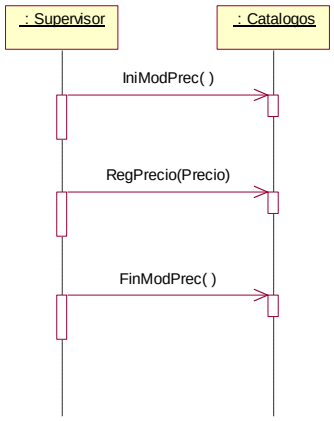


Figura 4.51 Diagrama de Secuencias por intuición del Caso de Uso No. 17

Diagrama de Colaboración

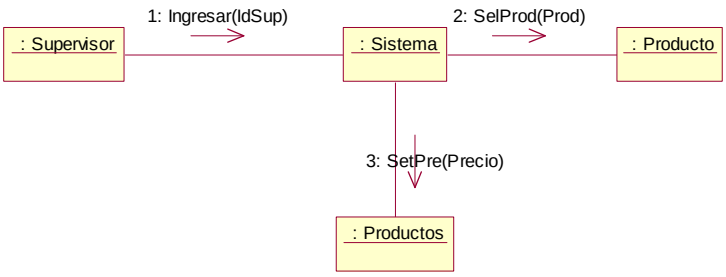


Figura 4.52 Diagrama de Colaboración General para el Caso de Uso 17

4.2.18 Caso de uso UC18: Modificación de ventas registradas

- Actor principal: Gerente

  - ❖ Personal involucrado e Intereses

  - Cajero: Requiere que los registros de las ventas estén capturadas de manera correcta dentro del sistema, aunque puede cometer algún error y este debe poder modificarse.
  - Gerente: Debe supervisar que los datos sean los correctos dentro de las ventas, para que estos no vayan a arrojar datos erróneos a la hora de imprimir y analizar los reportes.

- Precondiciones

  - ❖ El gerente de operaciones se identifica y autentifica.
  - ❖ El gerente debe tener los datos correctos de la venta que van a ser modificados.

- Post condiciones (Condiciones de Éxito)

  - ❖ Se actualiza la base de datos de las ventas con los nuevos datos que se modificaron.

- Escenario Principal de éxito (Flujo Principal)

1. El Caso de Uso inicia cuando se requiere modificar una venta y el gerente ingresa a la sección de las ventas que ya han sido capturadas.
2. El sistema muestra el primer registro de las ventas que se encuentran en la lista de ventas efectuadas.
3. El gerente selecciona el registro que se va a modificar y selecciona la opción de “Modificar Registro”.
4. El sistema deja los campos en estado de edición, listos para ser modificados
5. El gerente modifica los datos que requieran ser modificados y los guarda.
6. El sistema actualiza la base de datos con los nuevos datos, finaliza la operación e indica que los cambios se guardaron de forma correcta.
7. El registro ha sido modificado y puede desplegarse en los reportes de ventas que el sistema despliega.

- Extensiones (o Flujos Alternativos)

1 En cualquier momento el sistema falla
1.1 El sistema se reinicia sin llevar a cabo ningún cambio en el último registro que se estaba modificando.

5 El gerente se equivoca al capturar algún tipo de dato de la venta.
5.1 El sistema le indica al gerente que el dato no es correcto, limpia el dato y lo deja listo para que el gerente introduzca nuevamente el dato que corresponde

1-5 El gerente cancela la modificación.
1-5.1 Se finaliza el caso de uso, se limpian los campos y el sistema queda mostrando el catalogo de unidades.

Diagrama de Casos de Uso

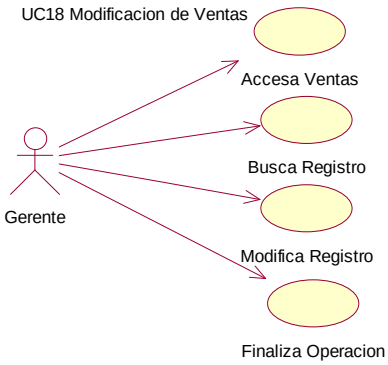


Figura 4.53 Caso de Uso: Modificar Ventas

Listamos las tareas a realizar para llevar a cabo la operación del caso de uso.

1.-iniciar la operación de modificación de ventas registradas.
2.-Buscar y seleccionar el registro de la venta a ser modificada.
3.-Registrar los nuevos datos de dicha venta.
4.-Finalizar la operación de modificación de ventas.

A continuación los diagramas de secuencia y de colaboración que se infieren de estas operaciones.

Diagrama de Secuencias

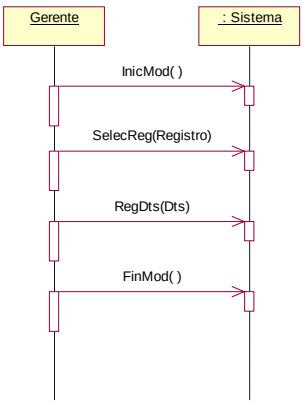


Figura 4.54 Diagrama de Secuencias por intuición del Caso de Uso No. 18

Diagrama de Colaboración

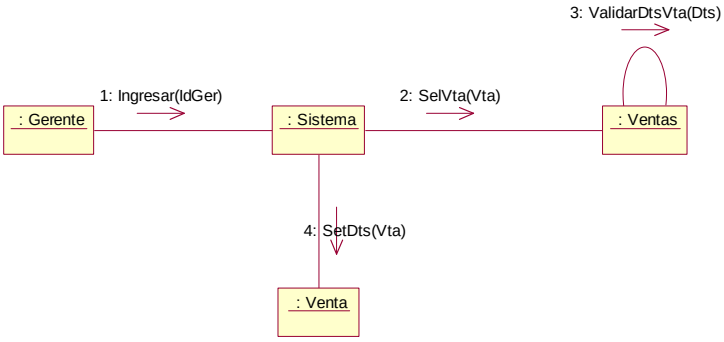


Figura 4.55 Diagrama de Colaboración General para el Caso de Uso 8

Capítulo V

Análisis y diseño utilizando patrones

Siguiendo con el diseño de la solución, en este capitulo vamos a discernir acerca de la solución utilizando patrones, vamos a retomar las operaciones localizadas en los casos de uso del capitulo anterior para desarrollar los contratos de colaboración, a modelar de manera diferente las diagramas de secuencia y de colaboración para observar las diferencias de estos con los mostrados en el capitulo anterior.

5.1 Caso de uso UC1: Registrar ventas de pipas

Siguiendo un diseño de la solución del caso de uso utilizando patrones, lo primero a tener en cuenta como generalmente debe ser, es identificar la finalidad principal del caso de uso, que es registrar una liquidación de pipas, y el fin principal de una liquidación de pipa es calcular el monto total de esa liquidación. Es decir, lo primero es asignar las responsabilidades estableciendo claramente las tareas a desarrollar.
Vamos a aplicar los Patrones GRASP para diseñar una solución del caso de uso.
Primero vamos a aplicar el patrón experto, con el cual vamos a buscar al experto en información del caso para asignarle una responsabilidad, para lo cual nos vamos a hacer las siguientes preguntas

1.- ¿Quién es el responsable de conocer el total de una liquidación?
Para esto vamos a hacer la analogía con el mundo real, con lo cual quizá el registro de la misma liquidación es un objeto que debe conocer el total de la liquidación, o sea de ella misma.
2.- ¿Qué información se necesita para determinar este total?
Conocer todas las instancias de línea de venta (en este caso las notas) y la suma de sus subtotales.
3.- ¿Qué información se necesita para determinar el subtotal de cada nota?
En este caso para saber el total de una nota se requiere de la cantidad de litros y el precio por litro que se le esta dando al cliente al que se le expidió esa nota.
En este caso la Nota es el experto de cada uno de sus totales, pues en la nota se especifican el total de litros, el precio por litro de la venta, y el total del monto a liquidar.

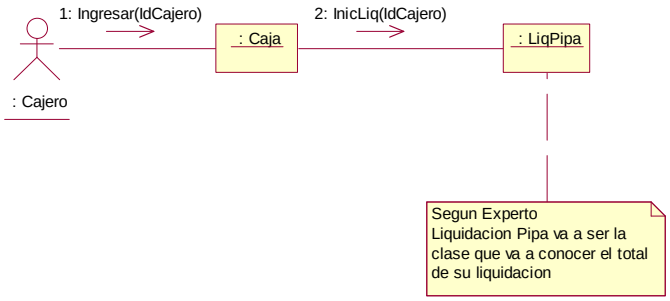


Figura 5.1 Asignación de responsabilidad según el patrón experto

Siguiendo con el diseño con patrones, vamos a aplicar el patrón Creador. Este patrón hace referencia a la asignación de la responsabilidad de la creación de una nueva instancia de la liquidación de pipas. En este caso un poco de manera intuitiva caja esta llevando la responsabilidad de iniciar la nueva liquidación de pipas. Haciendo un pequeño paréntesis, vamos a echar un vistazo al patrón factoría, el cual responde a la pregunta ¿Quién debe ser el responsable de la creación de los objetos cuando existan consideraciones especiales, como una lógica de creación compleja, con el deseo de separar las responsabilidades de la creación para mejorar la cohesión?

Aquí se podría considerar una nueva clase venta la cual se encargue de la creación de la liquidación de pipa, haciendo referencia al patrón creador, pero haciendo referencia al patrón factoría, a esa misma clase podríamos asignarle la responsabilidad de crear la liquidación de rutas posteriormente, lo cual nos hace tomar algunas consideraciones especiales para llevar a cabo las instancias tanto de una como de otra liquidaciones.

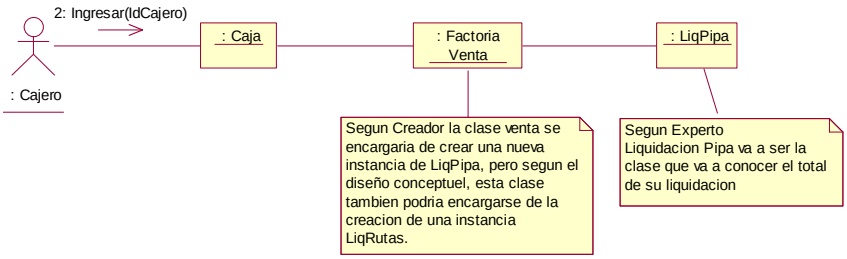


Figura 5.2 Asignación de responsabilidad según el patrón creador

Continuando con el diseño con patrones, vamos a aplicar el patrón controlador. Con este patrón vamos a asignar a alguna clase la responsabilidad de controlar los eventos del sistema a la clase que representa el evento de la liquidación de pipas que estamos realizando.

En este caso Caja seria un buen controlador, ya que seria la puerta de entrada a la liquidación de pipas.

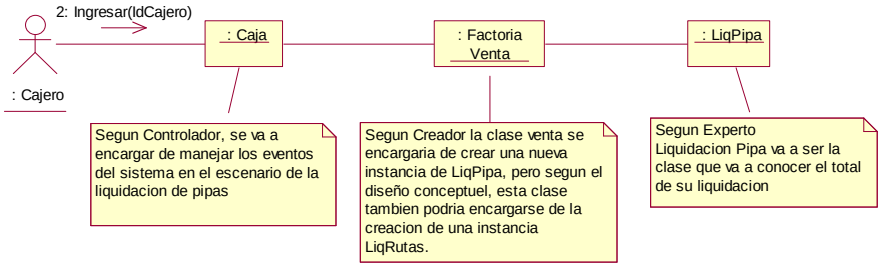


Figura 5.3 Asignación de responsabilidad según el patrón controlador

Una vez que se han establecido las clases y se han vislumbrado las responsabilidades que van a manejar estas clases podemos diseñar el diagrama de secuencia.

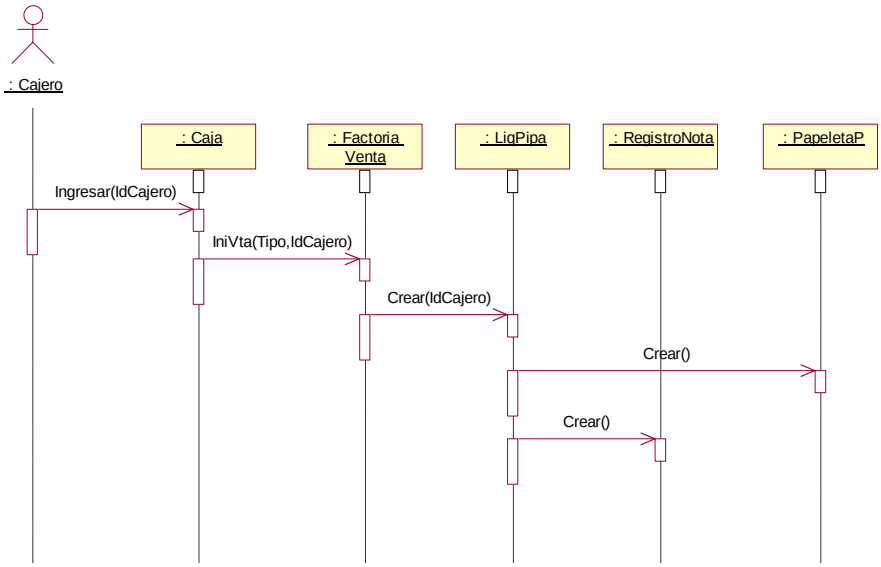


Figura 5.4 Desarrollo de Diagrama de Secuencias Utilizando Patrones

Aquí se ve claramente la diferencia de este diagrama de secuencia, con el mostrado en el capítulo anterior para este mismo caso de uso.

Se puede identificar claramente como en un diseño realizado sin patrones se podría caer en el error de asignarle a una clase una gran cantidad de responsabilidades provocando así una baja cohesión, y el manejar muchas responsabilidades, es común que el acoplamiento de dicha clase sea alto, al tener una gran cantidad de tareas por realizar.

Por otro lado utilizando patrones, vamos a ir asignando responsabilidades de una manera ordenada, con lo que vamos a favorecer una alta o moderada cohesión, lo cual va a favorecer para mantener un bajo acoplamiento, al establecer tareas bien definidas a las clases y sus relaciones entre ellas. Vamos a realizar los contratos y las colaboraciones de acuerdo a lo que hemos realizado.

Contratos de Operaciones y Colaboraciones

Operación: InicLiq(idCajero)
Responsabilidades: Crear una nueva liquidación, Preparar la base de datos para la liquidación, asociar el cajero al movimiento.
Caso de Uso: Liquidación Pipas
Controlador: Sistema
Precondiciones: El sistema Conoce el ID del Cajero
Postcondiciones: -Se abrió la base de datos de las Liquidaciones - Se genero la nueva instancia de venta “vp” - Vp se asocio con el cajero que lo genero

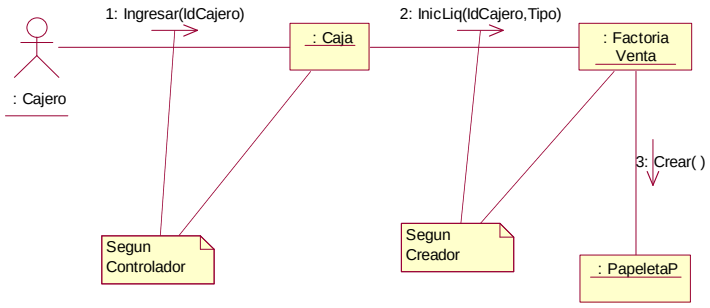


Figura 5.5 Diagrama de Colaboración para lo operación InicLiq(IdCajero)

Operación: SelTipVta(tipo)
Responsabilidades: Seleccionar el tipo de venta para pipas
Caso de Uso: Liquidación Pipas
Controlador: Sistema
Precondiciones: Se tiene en curso el cajero que genero la liquidación, se esta procesando la liquidación
Postcondiciones: - Vp se asocio al tipo de venta “pipas”.

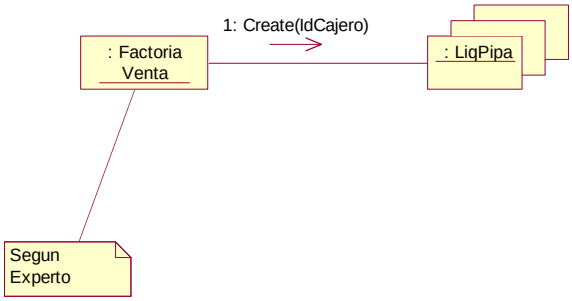


Figura 5.6 Diagrama de Colaboración para lo operación SelTipVta(tipo)

Operación: SelOpAyUn(op,ay,un)
Responsabilidades: Buscar los datos de la unidad que va a liquidar, así como su operador y su ayudante, asociar los ID de estos a la venta vp
Caso de Uso: Liquidación Pipas
Controlador: Sistema
Precondiciones: El sistema conoce la unidad, el operador y el ayudante que se van a registrar con la venta
Postcondiciones: - se asociaron la unidad, operador y ayudante a |la venta vp

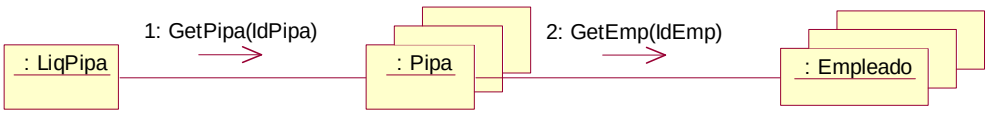


Figura 5.7 Diagrama de Colaboración para lo operación SelOpAyUn(op,ay,un)

Operación: RegNotas(nota)
Responsabilidades: Generar los registros de las notas de venta, Asociar los registros de las notas a la venta vp
Caso de Uso: Liquidación Pipas
Controlador: Sistema y Cajero
Precondiciones: Se esta procesando la liquidación.
Postcondiciones: - Se ha generado un registro por cada nota de venta captada - Se ha acumulado el monto al total de venta - se asocio cada nota con la venta vp.

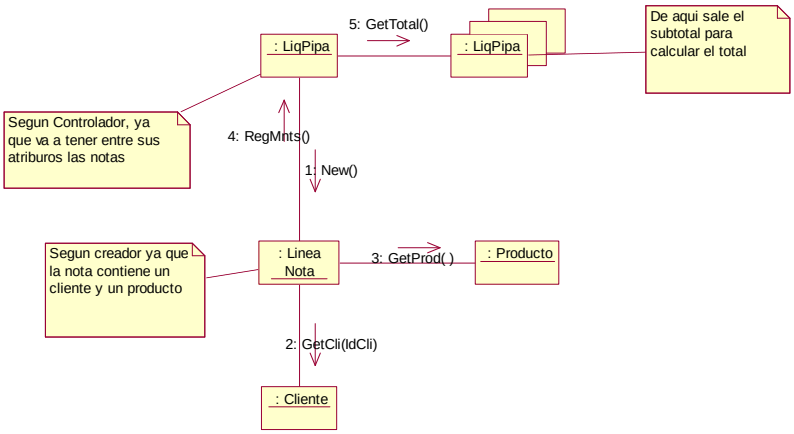


Figura 5.8 Diagrama de Colaboración para lo operación RegNotas(nota)

Operación: RegPago(Monto)
Responsabilidades: Indicar el cambio que debe ser devuelto por el cajero, asociar el monto total de venta a la liquidación, asociar los montos de los diferentes tipos de pago efectuados en la liquidación.
Caso de Uso: Liquidación Pipas
Controlador: Sistema
Precondiciones: Se han registrado el total de las notas de venta. Se entrego el monto equivalente al total de venta al cajero.
Postcondiciones: - Se registro cada monto total con el tipo de pago que se realizo. - Se cerró el registro de notas de venta.

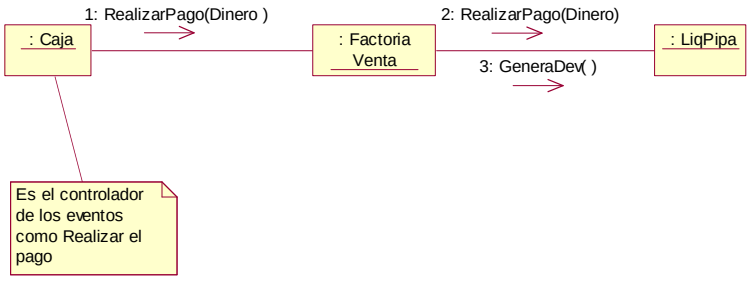


Figura 5.9 Diagrama de Colaboración para lo operación RegPago(Monto)

Operación: FinLiq( )
Responsabilidades: Cerrar la base de datos, Finalizar la Operación de Venta, Preparar el Sistema para registrar una nueva Liquidación.
Caso de Uso: Liquidación Pipas
Controlador: Sistema
Precondiciones: Se ha terminado de registrar los datos de la venta.
Postcondiciones: - Se cerró la operación de liquidación - Se dejo listo el sistema para iniciar una nueva liquidación.

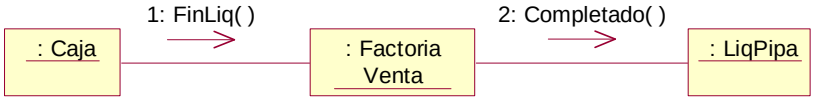


Figura 5.10 Diagrama de Colaboración para lo operación FinLiq()

A continuación el Diagrama de Clases de Diseño

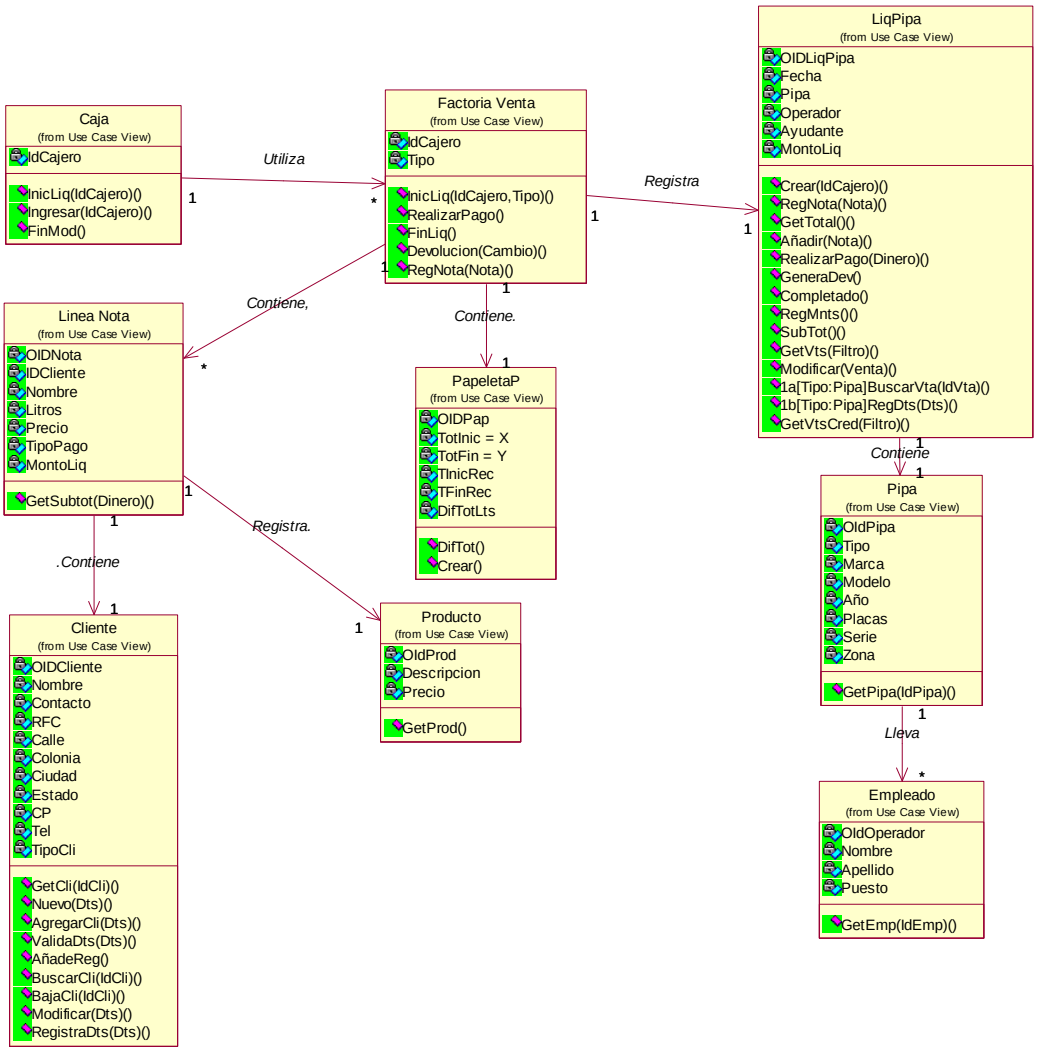


Figura 5.11 Diagrama de Clases de Diseño para el caso de uso “Registrar Ventas de Pipas”

Por otro lado, una Clase fachada va a ser la encargada de recibir los eventos de la GUI y de comunicarse con la clase Caja, ya que la clase Caja es la clase controladora y la que va a hacer la solicitud de los eventos que van a pasar de la clase fachada, los que van a ser iniciados por la GUI.

En este caso de acuerdo a la interface y el funcionamiento se puede localizar los eventos que van a detonar los eventos del sistema.

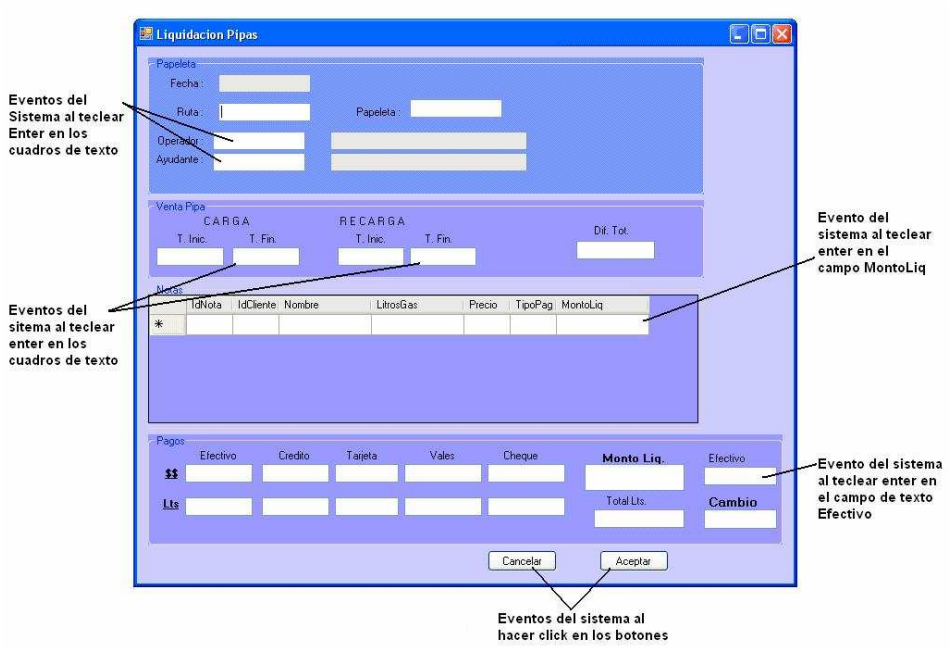


Figura 5.12 Localización de eventos de la interfaz de usuario para la liquidación de pipas

Así, por ejemplo al hacer click en el botón aceptar, se pasaría el evento click de dicho botón al método BtnAceptarClick, el cual iniciaría los eventos del contrato de operación FinLiq(), para indicar que se ha terminado el escenario.

La Clase Fachada de la Venta va a ser la encargada de relacionarse con el controlador del caso de uso

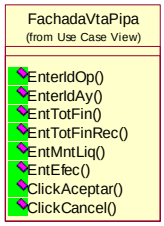


Figura 5.13 Clase FachadaVtaPipa con sus eventos.

Los eventos que se generan desde el Interface de la Liquidación no controlaran ninguna operación del funcionamiento del sistema, pues esta tarea la realizara la clase fachada, y en caso de cambiar la interface, o agregarle algún funcionamiento mas u operación a esta interface, esta no va a interferir con el diseño de la aplicación, ya que esta es independiente.

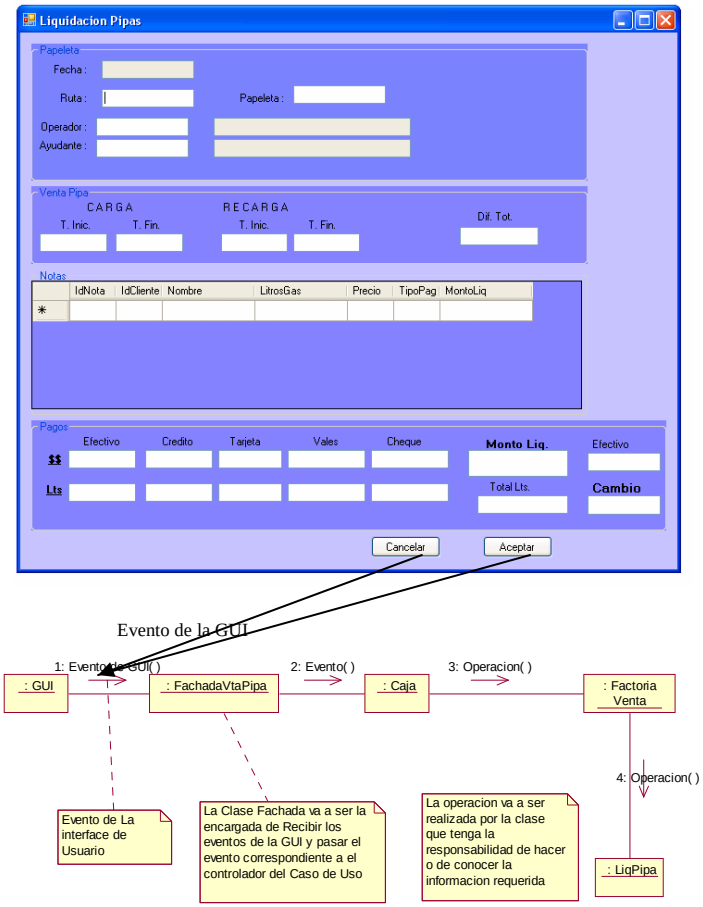


Figura 5.14 Asignación de Eventos de la GUI a la Fachada del Sistema

Así tenemos el nuevo Diagrama de Clases de Diseño:

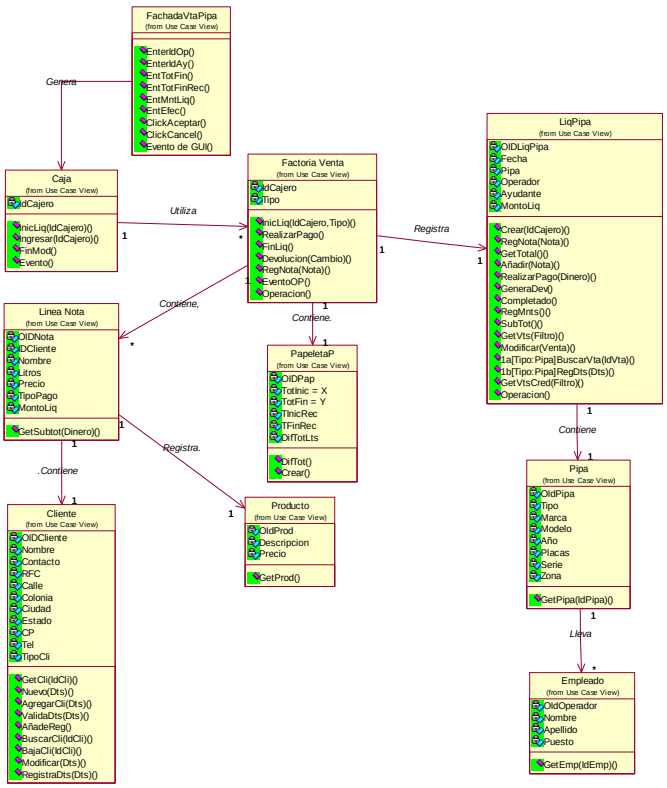


Figura 5.15. Diagrama de clases de diseño para el caso de uso uno implementando la clase fachada

5.2 Caso de uso UC2: Registrar ventas de cilindros

Aplicando patrones GRASP vamos a asignar las responsabilidades a las clases que vayamos identificando que deban ser las encargadas de llevar a cabo alguna tarea.

Aplicando el patrón experto se va a buscar al experto en información del caso para asignarle una responsabilidad, para lo cual nos vamos a hacer las preguntas:

1.- ¿Quién es el responsable de conocer el total de una liquidación?
Para esto vamos a hacer la analogía con el mundo real, con lo cual la misma liquidación es un objeto que debe conocer el total de la liquidación, al igual que en el caso de uso anterior.
2.- ¿Qué información se necesita para determinar este total?
Conocer su instancia correspondiente de la papeleta, la cual contiene la información de la venta realizada, como son el total de cilindros en sus diferentes denominaciones.

En este caso la Papeleta es el experto de su total, pues en ella se especifican el total de cilindros, tanto de 20, 30, 45, kilos y las ánforas, así como el total del monto a liquidar.
Siguiendo con este diseño, vamos a aplicar el patrón Creador. Este patrón nos va a servir para asignarle la responsabilidad de la creación de una nueva instancia de la liquidación de cilindros.

Aquí se podría considerar la clase venta para que se encargue de la creación de la liquidación de cilindros, haciendo referencia al patrón creador, pero haciendo referencia al patrón factoría, pues esta misma clase tiene la responsabilidad de la creación de la instancia de la liquidación de pipas.

A continuación vamos a aplicar el patrón controlador. Con este patrón vamos a asignar a alguna clase la responsabilidad de controlar los eventos del sistema a la clase que representa el evento de la liquidación de cilindros que estamos realizando.

En este caso Caja seria un buen controlador, ya que seria la puerta de entrada a la liquidación de cilindros.

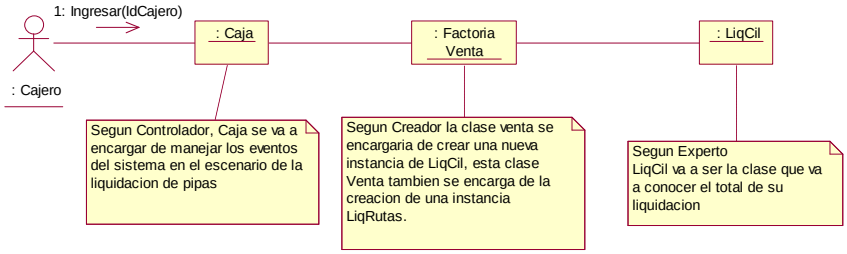


Figura 5.16 Asignación de responsabilidades para el UC2 utilizando patrones

Una vez que se han establecido las clases y se han vislumbrado las responsabilidades que van a manejar estas clases podemos diseñar el diagrama de secuencia.

Diagrama de Secuencias

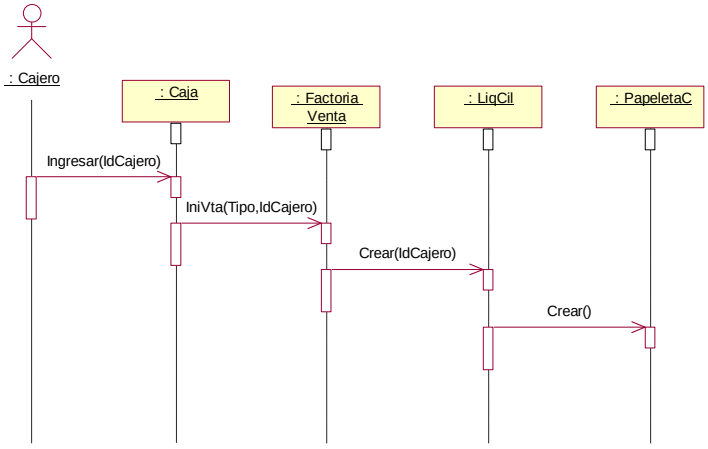


Figura 5.17 Diseño del Diagrama de Secuencias según las responsabilidades asignadas con patrones

No se puede dejar de hacer referencia al diagrama de secuencia de este caso de uso que se presentó en el capitulo anterior para hacer la comparación y hacer notar que se vuelven mas simples las clases al delegar responsabilidades a otras clases, para mejorar la cohesión al quitarle responsabilidades a estas, lo que las vuelve mas simples, mantenibles y entendibles.

A continuación se desarrollan los contratos de operaciones y los diagramas de colaboración de acuerdo a las operaciones, para desarrollar el caso de uso, que se encontraron en el capitulo anterior.

Contratos de Operaciones y Colaboraciones

Operación: InicLiq(idCajero)
Responsabilidades: Crear una nueva liquidación, preparar la base de datos para la liquidación de cilindros, asociar el cajero a la liquidación.
Caso de Uso: Liquidación Cilindros
Controlador: Sistema
Precondiciones: El sistema Conoce el ID del Cajero.
Postcondiciones: -Se abrió la base de datos de las Liquidaciones - Se genero la nueva instancia de venta “vc” - vc se asocio con el cajero que lo genero

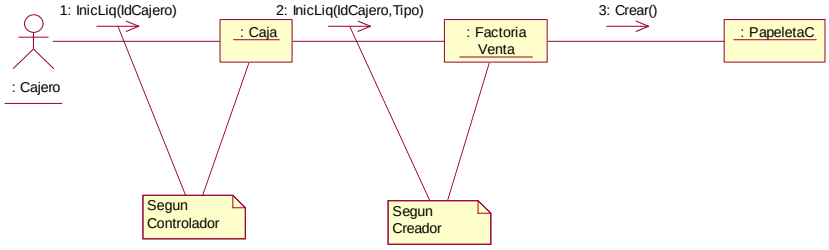


Figura 5.18 Diagrama de Colaboración para lo operación InicLiq(IdCajero)

Operación: SelTipVta(tipo)
Responsabilidades: Seleccionar el tipo de venta para pipas
Caso de Uso: Liquidación Cilindros
Controlador: Sistema
Precondiciones: Se tiene en curso el cajero que genero la liquidación, se esta procesando la liquidación
Postcondiciones: - vc se asocio al tipo de venta “cilindros”.

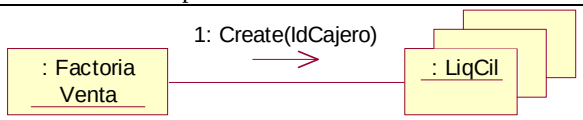


Figura 5.19 Diagrama de Colaboración para lo operación SelTipVta(tipo)

Operación: SelOpAyUn(op,ay,un)
Responsabilidades: Buscar los datos de la unidad que va a liquidar, buscar el operador que realizo las ventas y su ayudante, asociar los ID de estos a la venta vc.
Caso de Uso: Liquidación Cilindros
Controlador: Sistema
Precondiciones: El sistema conoce la unidad, el operador y el ayudante que se van a registrar con la venta
Postcondiciones: - se asociaron la unidad, operador y ayudante a la venta vc.

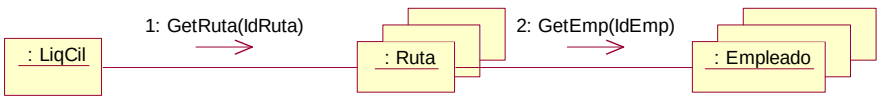


Figura 5.20 Diagrama de Colaboración para lo operación SelOpAyUn(op,ay,un)

Operación: RegPapeleta(Datos)
Responsabilidades: Generar el registro de las papeleta de la venta, Asociar el registro de la papeleta a la venta vc
Caso de Uso: Liquidación Cilindros
Controlador: Sistema y Cajero
Precondiciones: Se esta procesando la liquidación.
Postcondiciones: - Se ha generado el registro para los datos que lleva la papeleta - Se asocio la papeleta con la venta vc. - Se realizo el cálculo del monto a liquidar según lo vendido.

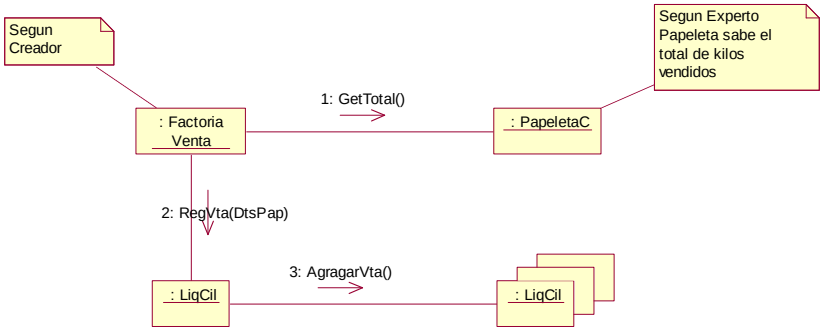


Figura 5.21 Diagrama de Colaboración para lo operación RegPapeleta(Datos)

Operación: RegPago(Monto)
Responsabilidades: Indicar el cambio que debe ser devuelto por el cajero, asociar el monto total de venta a la liquidación, asociar los montos de los diferentes tipos de pago efectuados en la liquidación.
Caso de Uso: Liquidación Cilindros
Controlador: Sistema
Precondiciones: Se han registrado los datos de la papeleta. Se entrego el monto equivalente al total de venta al cajero.
Postcondiciones: - Se registro cada monto total con el tipo de pago que se realizo. - Se cerró el registro de la papeleta.

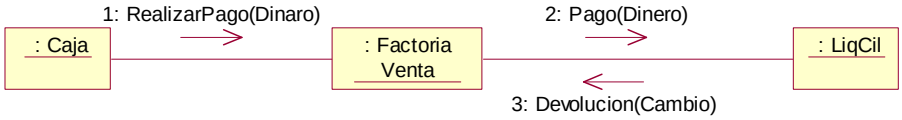


Figura 5.22 Diagrama de Colaboración para lo operación RegPago(Monto)

Operación: FinLiq( )
Responsabilidades: Cerrar la base de datos, Finalizar la Operación de Venta, Preparar el Sistema para registrar una nueva Liquidación.
Caso de Uso: Liquidación Cilindros
Controlador: Sistema
Precondiciones: Se ha terminado de registrar los datos de la venta.
Postcondiciones: - Se cerró la operación de liquidación - Se dejo listo el sistema para iniciar una nueva liquidación.

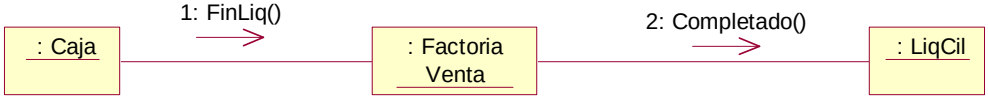


Figura 5.23 Diagrama de Colaboración para lo operación FinLiq()

A continuación se hace referencia al patrón fechada para generar una clase fachada para la liquidación de cilindros, la cual va a ser la encargada de recibir los eventos de la GUI, los cuales se identifican en la figura, y comunicarse con la clase Caja, que es la controladora de la operación del caso de uso.

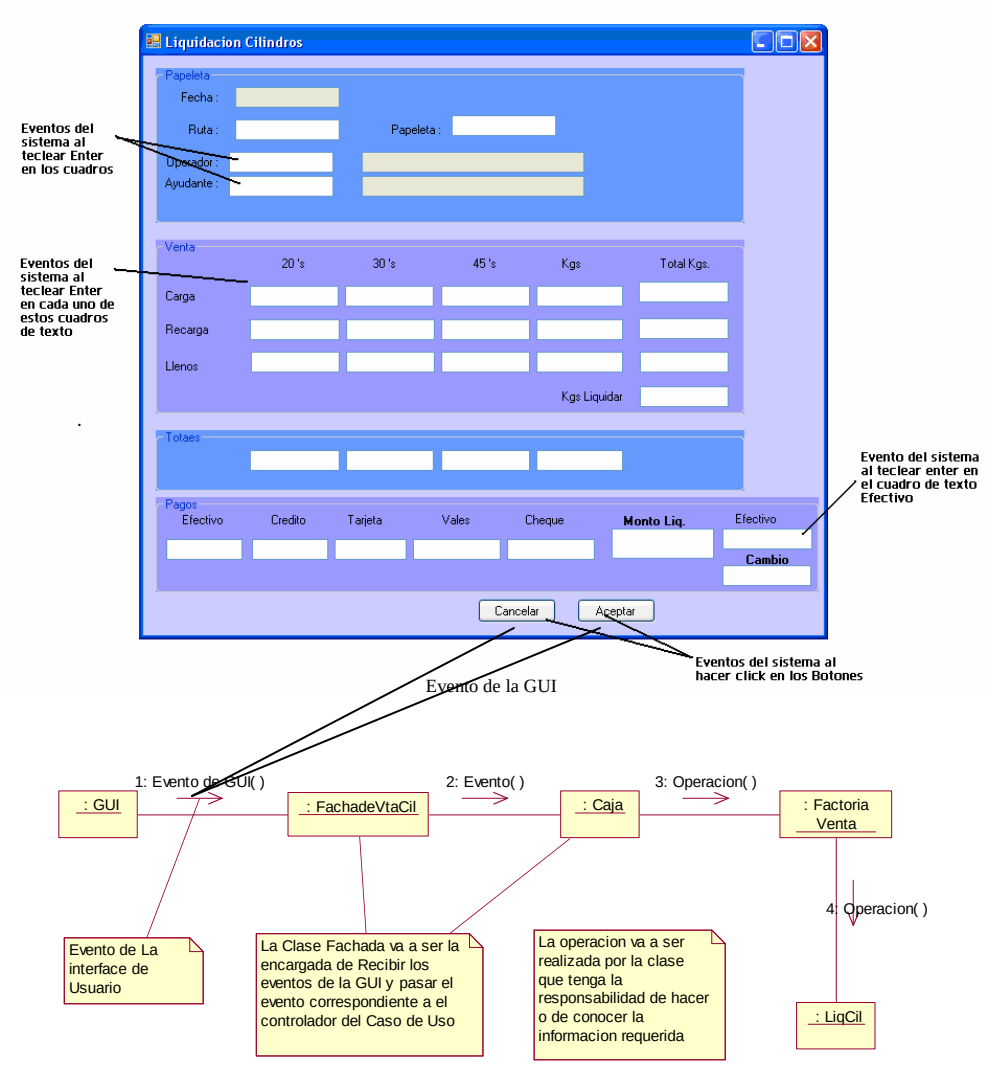


Figura 5.24 Asignación de Eventos de la GUI a la fachada del sistema de venta de cilindros

Los eventos que se generan desde la interface de la liquidación no controlarán ninguna operación del funcionamiento del sistema, pues esta tarea la realizara la clase fachada, y en caso de cambiar la interface, o agregarle algún funcionamiento más u operación a esta interface, esta no va a interferir con el diseño de la aplicación, ya que esta es independiente.

De esta manera tenemos el Diagrama de Clases de Diseño:

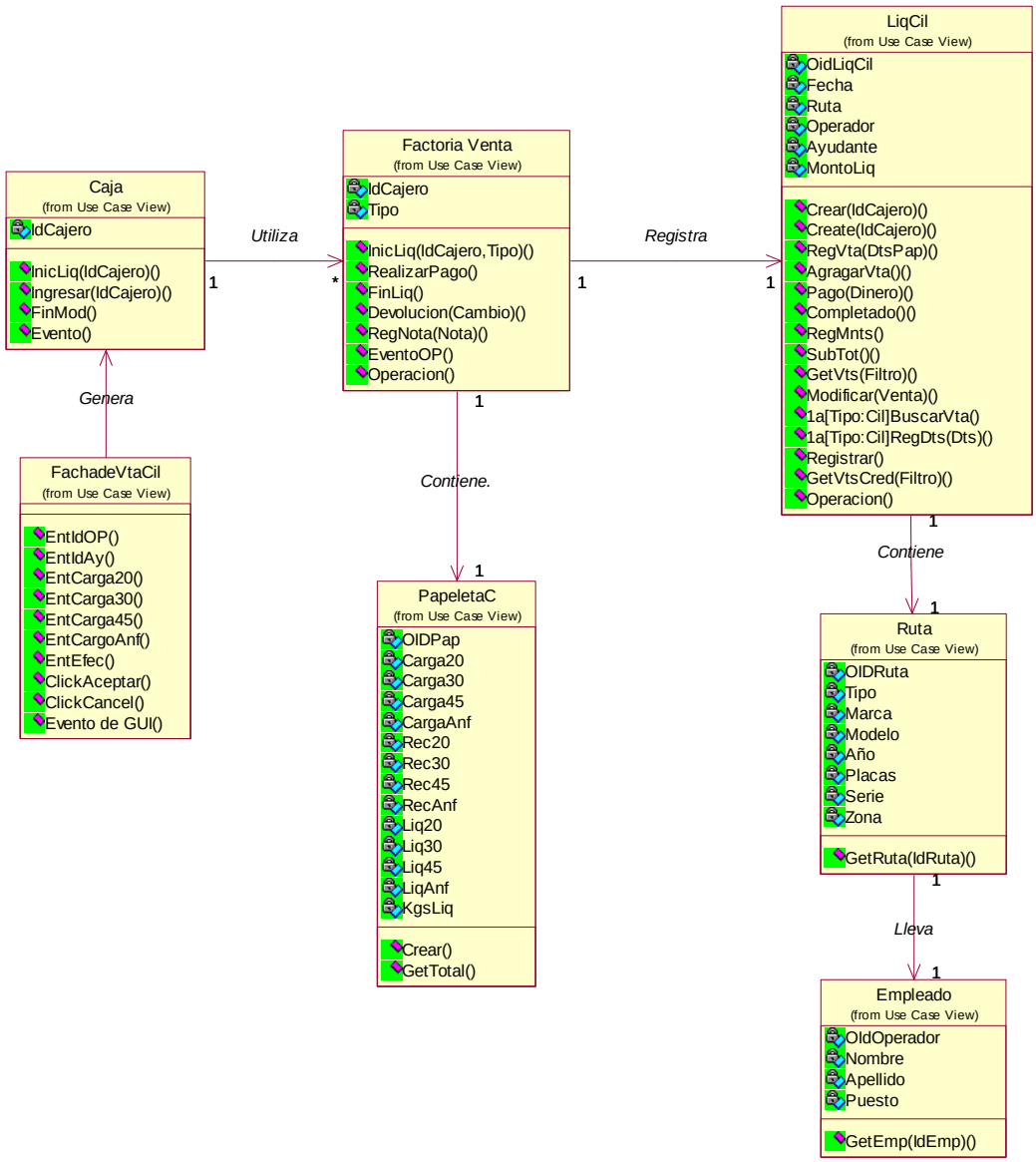


Figura 5.25. Diagrama de Clases de Diseño para el Caso de Uso 2 implementando la clase fachada

5.3 Caso de uso UC3: Registrar corte de caja

Siguiendo con el diseño utilizando patrones GRASP, en este caso de uso aplicando el patrón Experto, nos hacemos la pregunta ¿Quien es el responsable de saber los montos totales de un corte de caja? Y haciendo la analogía con el mundo real, estos montos quedan plasmados en un reporte que se llama corte de caja, con lo que se infiere que se puede crear una clase Corte de caja para que ella guarde la información de los montos totales de esta operación.
¿Qué información se necesita para determinar este total? Es necesario saber el total de los montos de todas las liquidaciones que hayan sido registradas por el cajero que va a realizar el corte.

Aplicando el patrón creador, el cual nos guía en la asignación de la responsabilidad de crear una tarea u objeto, y en este caso lo que vamos a realizar es registrar el corte de caja, con lo que una buena clase candidata seria Registro de Venta, el cual va a controlar la creación de corte de caja ya que el patrón creador sugiere una clase contenedor o registro, haciendo referencia al concepto agregación el que involucra que se encuentran en una relación Todo-Parte o Ensamblaje-Parte, y en este caso Registro de Venta hará referencia a todas las liquidaciones tanto de cilindros como de pipas.

Nuevamente caja suena como el controlador lógico de los eventos de este caso de uso, al ser la puesta de entrada al mismo, de esta manera tenemos:

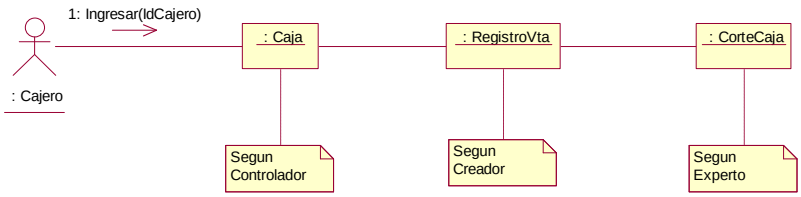


Figura 5.26 Asignación de responsabilidades para el UC3 utilizando patrones

Una vez establecido lo anterior podemos llevar a cabo el diagrama de Secuencias.

Diagrama de Secuencias

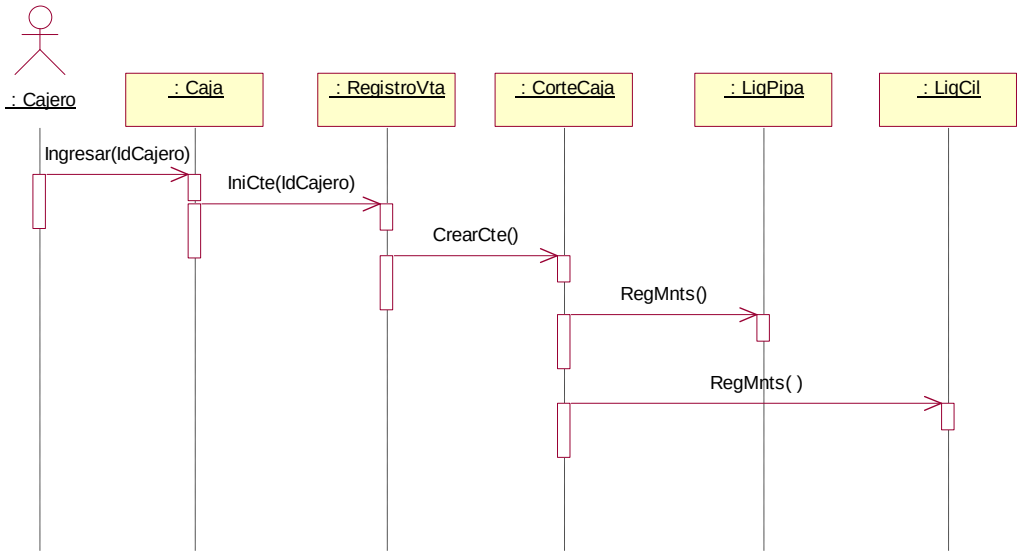


Figura 5.27 Diseño del Diagrama de Secuencias según las responsabilidades asignadas con los patrones

Si revisamos este diagrama y lo comparamos con el mismo pero del capitulo anterior, observamos como las responsabilidades de las clases desminuyen al delegar a otra clase las tareas a realizarse para cumplir con el fin del caso de uso.
Ahora vamos a realizar los contratos de operaciones y colaboraciones.

Contratos de Operaciones y Colaboraciones

Operación: InicCorte(idCajero)
Responsabilidades: Iniciar el registro de un corte de caja, Calcular los montos totales según el tipo de pago que se ha realizado en cada liquidación.
Caso de Uso: Corte de caja
Controlador: Sistema
Precondiciones: El sistema conoce el Id del cajero, las liquidaciones han terminado de realizarse en su totalidad.
Postcondiciones: - Se abrió la base de datos CorteCaja - Se genero una nueva instancia de corte de caja "CtCj" - Se asocio el id Del cajero con la instancia generada CtCj - Se han sumado los montos de las liquidaciones que registró el cajero, por tipo de pago efectuado, (efectivo, cheque o crédito)

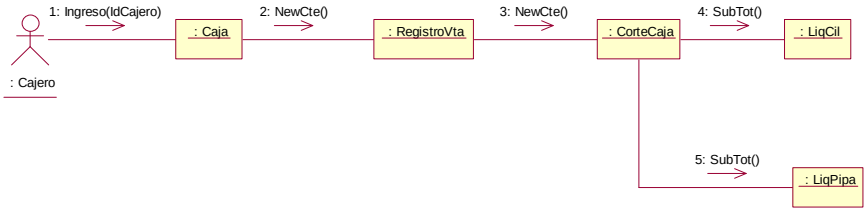


Figura 5.28 Diagrama de Colaboración para lo operación InicCorte(IdCajero)

Operación: RegMnts(monto)
Responsabilidades: Registrar los montos que tiene el cajero en su caja, mostrar la diferencia que hay entre estos montos y los cálculos de los totales que registró el sistema
Caso de Uso: Corte de Caja
Controlador: Sistema y Cajero
Precondiciones: El sistema conoce los totales de los montos por tipo de pago registrados por el cajero.
Postcondiciones: - Se realizo la comparación de los montos calculados con los montos registrados - Se asociaron los montos calculados y los montos registrados con el corte CtCj

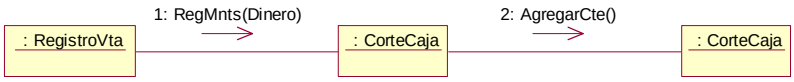


Figura 5.29 Diagrama de Colaboración para lo operación RegMnts(monto)

Operación: FinCorte( )
Responsabilidades: Cerrar la base de Datos, Finalizar la operación,
Caso de Uso: Corte de Caja
Controlador: Sistema
Precondiciones: Se terminaron de registrar los montos, La diferencia de la comparación debe ser cero
Postcondiciones: -Se Cerró la operación de Corte de Caja

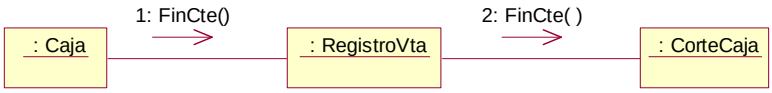


Figura 5.30 Diagrama de Colaboración para lo operación FinCorte()

Volviendo a hacer referencia al patrón Fachada, vamos a ubicar los eventos que genere la interface para el caso de uso.

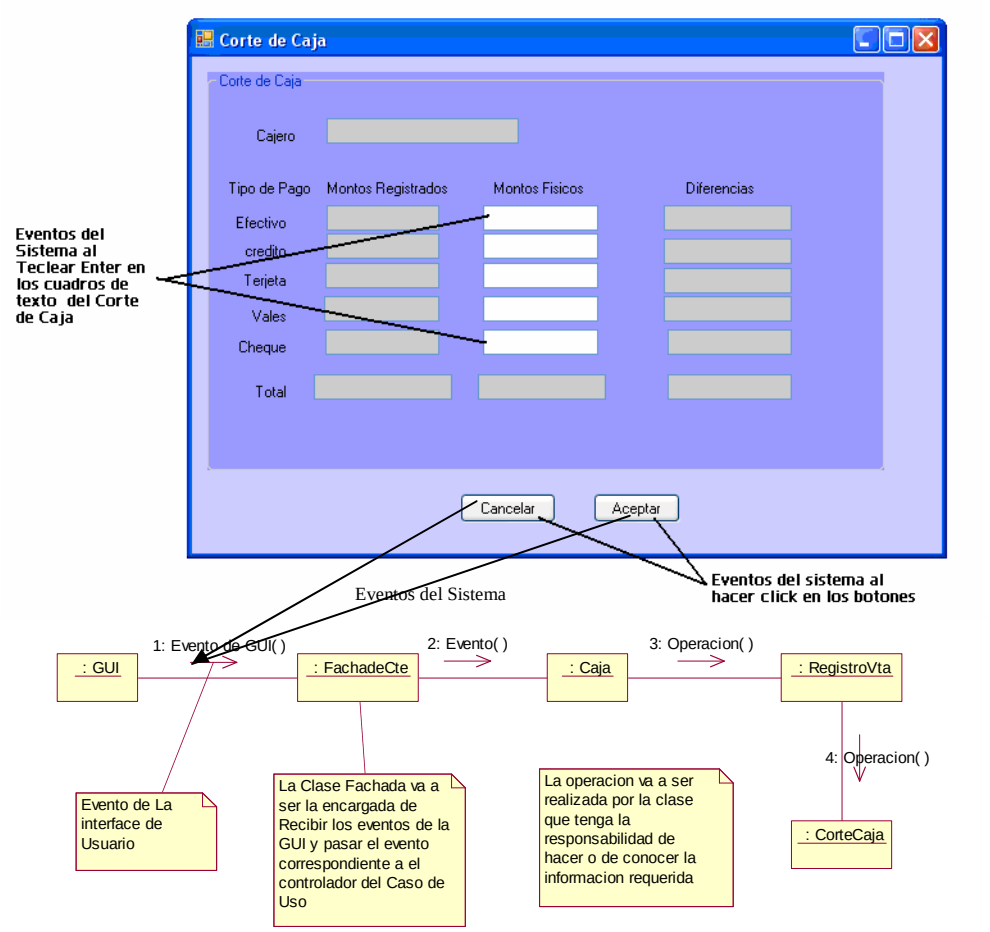


Figura 5.31 Asignación de Eventos de la GUI a la fachada del sistema de RegCteCaja

De esta manera, los eventos que se generan de la Interface en el corte de caja no controlaran ninguna operación del funcionamiento del sistema, pues esta tarea la lleva a cabo la clase fachada, y algún cambio en la interface no va a afectar el funcionamiento del sistema.

Y así, tenemos el Diagrama de Clases de Diseño

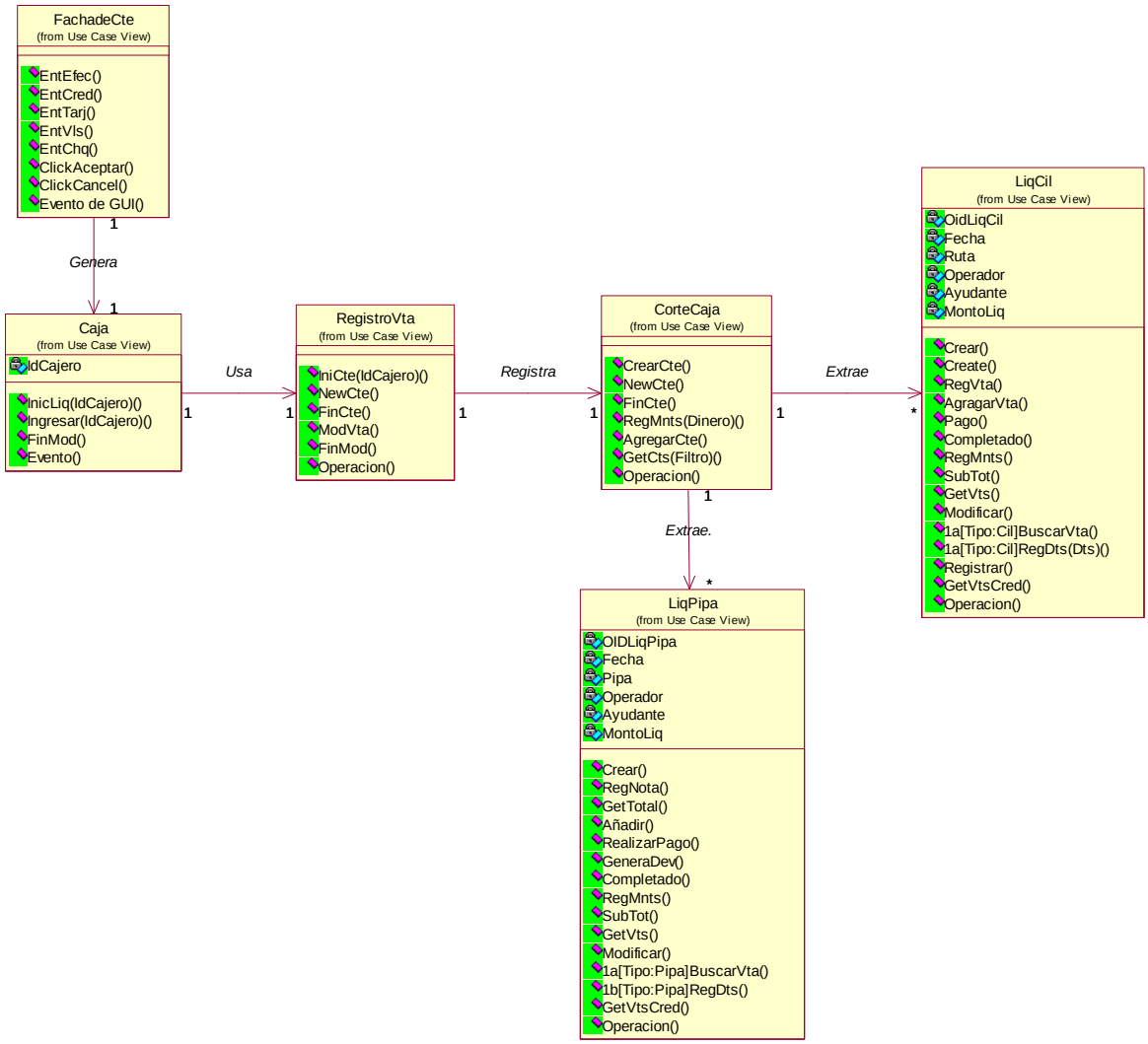


Figura 5.32 Diagrama de Clases de Diseño para el Caso de Uso Registrar Corte de Caja

5.4 Caso de uso UC4: Alta de trabajadores en el sistema

Aplicando nuevamente patrones GRASP vamos a asignar responsabilidades para llevar a cabo el alta de un nuevo registro de un trabajador.

Primero aplicando el patrón experto vamos a asignar la responsabilidad de saber los datos a ser registrados a una clase "Empleado", que va a ser la que va a contener los datos del nuevo trabajador que va a ser dado de alta, como son su nombre, apellidos, y puesto, etc.

Luego, aplicando el patrón Creador, vamos a utilizar una clase registro, ya que la operación va a ser la de registrar un nuevo empleado, por lo que la responsabilidad de crear la instancia de un nuevo trabajador va a ser una clase "Registro de Trabajador".

Finalmente, para el manejo y control de los eventos del caso de uso vamos a utilizar una clase "Catalogo de Trabajadores" ya que haciendo una analogía con el mundo real, seria alguien encargado del catálogo de empleados (Recursos Humanos), el encargado de llevar el registro de estos, así como saber los movimientos de dichos trabajadores dentro de la empresa.

De lo anterior podemos ver que las clases encargadas de dichas tareas quedan así:

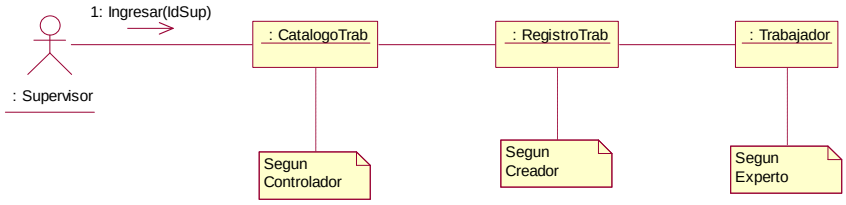


Figura 5.33 Asignación de Responsabilidades para el alta de trabajadores en el sistema

Una vez que tenemos identificadas las clases con sus responsabilidades o tareas, podemos llevar a cabo el diagrama de secuencias para identificar la secuencia de dichas operaciones.

Diagrama de Secuencias

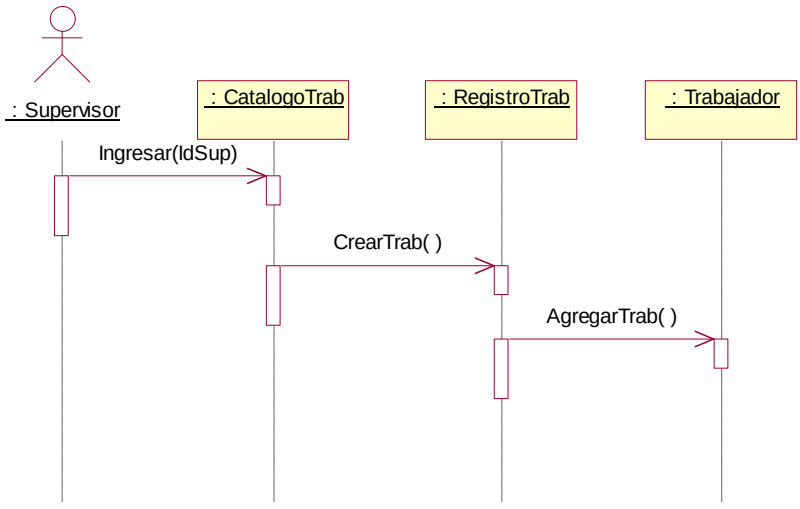


Figura 5.34 Diseño del Diagrama de Secuencias según las responsabilidades asignadas con patrones

Si observamos este diagrama y lo comparamos con el que habíamos realizado en el capitulo anterior se puede observar como las tareas están repartidas y no es solo una la encargada de hacer todo, con lo que se favorece la alta cohesión y el bajo acoplamiento.

Con las clases que hemos identificado y con las operaciones que obtuvimos en el capitulo anterior para este caso de uso vamos a desarrollar los contratos de operaciones y los diagramas de colaboración.

Contratos de Operaciones y Colaboraciones

Operación: InicAltaTrab( )
Responsabilidades: Iniciar el registro de un nuevo trabajador, Preparar el catalogo de Trabajadores para el nuevo registro.
Caso de Uso: Alta de Trabajador
Controlador: Sistema
Precondiciones: El sistema conoce el ID del Supervisor
Postcondiciones: - Se genero una nueva instancia de AltaTrab "AlTr" - Se abrió el Catalogo de Trabajadores

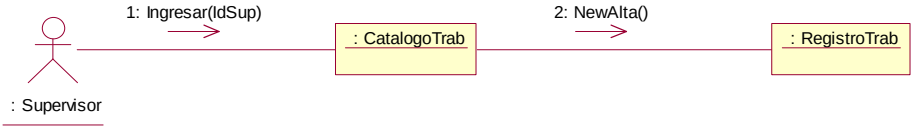


Figura 5.35 Diagrama de Colaboración para lo operación InicAltaTrab()

Operación: RegDtsTrab(Dts)
Responsabilidades: Generar el registro del nuevo trabajador en el catalogo.
Caso de Uso: Alta de Trabajador.
Controlador: Sistema, Supervisor.
Precondiciones: Esta en Transición el alta del trabajador.
Postcondiciones: - Se registraron los datos del nuevo trabajador en el catalogo.

Figura 5.36 Diagrama de Colaboración para lo operación RegDtsTrab(Dts)

Operación: FinAlta( )
Responsabilidades: Cerrar el catalogo de trabajadores, Finalizar la Operación.
Caso de Uso: Alta Trabajador
Controlador: Sistema
Precondiciones: Se validaron los datos del nuevo trabajador.
Postcondiciones: Se cerró la operación del Alta de Trabajador.

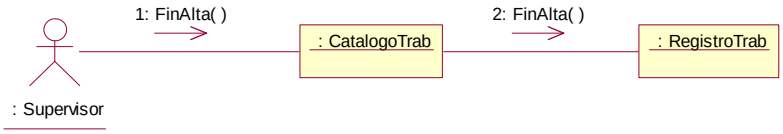


Figura 5.37 Diagrama de Colaboración para lo operación FinAlta()

Con el patrón fachada vamos a generar la clase que va a ser la encargada de recibir los eventos de la interface, y sus operaciones para llevar a cabo las tareas para realizar el caso de uso como se requiere.

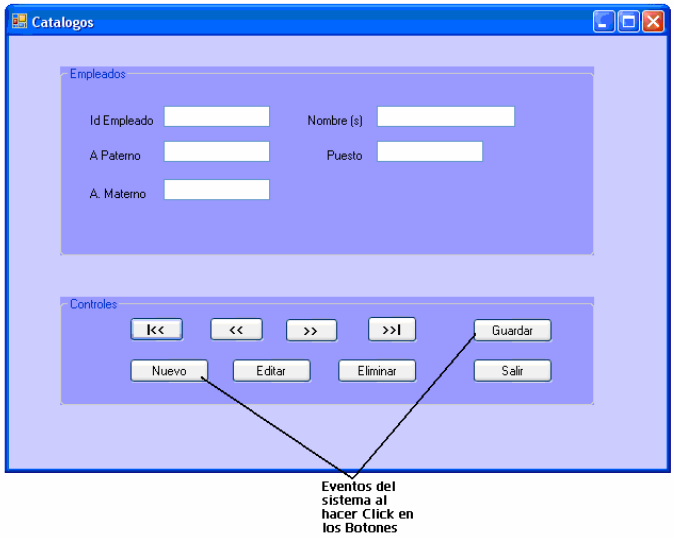


Figura 5.24 Localización de los eventos de la GUI para el alta de empleados

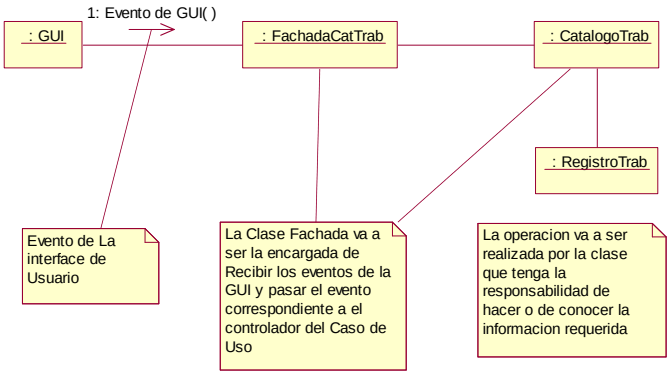


Figura 5.38 Clase fachada del sistema para alta de empleados

Identificadas las clases, sus colaboraciones y las tareas que estas llevan a cabo tenemos el Diagrama de Clases de Diseño.

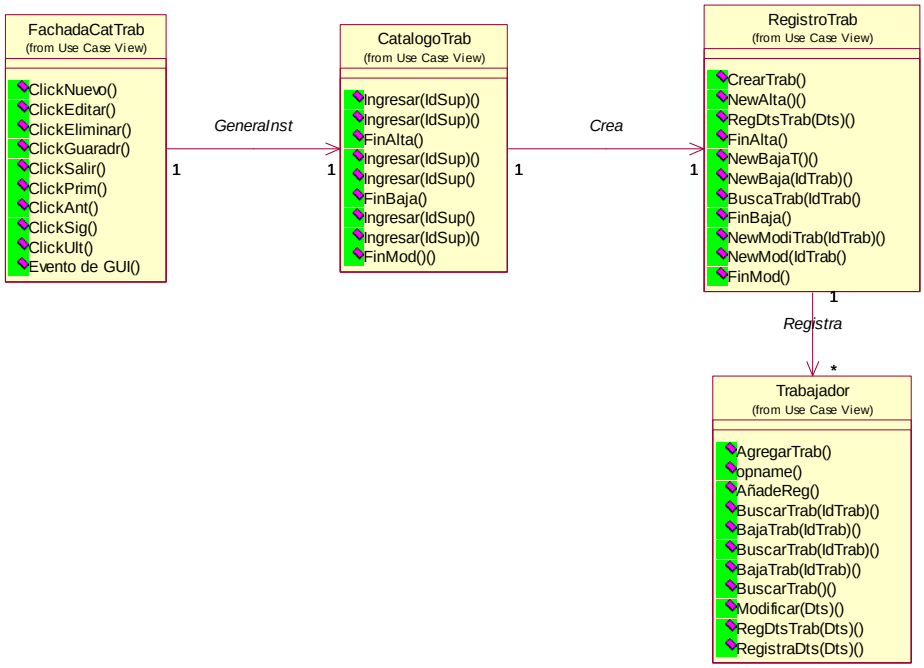


Figura 5.39 Diagrama de Clases de Diseño para el caso de uso alta de empleados

5.5 Caso de uso UC5: Baja de trabajadores en el sistema

Aplicando el patrón experto para asignar la responsabilidad de saber los datos del registro del empleado que va a ser dado de baja, se observa que al igual que en el caso de uso anterior es el mismo empleado el que debe saber sus datos.

Después, al aplicar el patrón Creador, observamos que también el creador del caso de uso anterior debe ser el encargado de crear la instancia de la clase del trabajador que va a ser dado de baja, por lo que la responsabilidad de crear dicha instancia del trabajador va a ser una clase "Registro de Trabajador".

De manera similar se puede inferir que la clase "Catalogo de Trabajadores" va a ser la encargada del manejo y control de los eventos del caso de uso.

De lo anterior podemos ver que las clases encargadas de dichas tareas van a ser las mismas que en el caso de uso anterior:

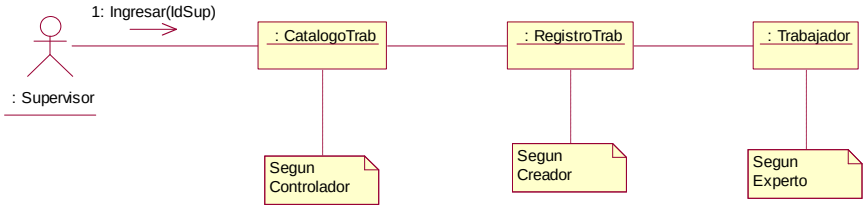


Figura 5.40 Asignación de responsabilidades para el case de uso utilizando patrones

Una vez que tenemos identificadas las clases con sus responsabilidades o tareas, podemos llevar a cabo el diagrama de secuencias para identificar la secuencia de dichas operaciones.

Diagrama de Secuencias

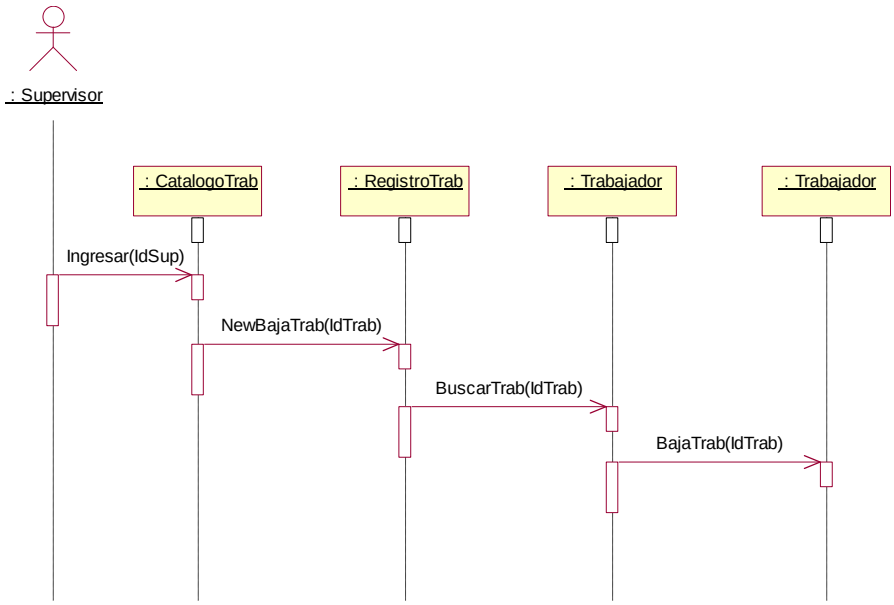


Figura 5.41 Diseño del diagrama de secuencias según las responsabilidades asignadas con patrones

Identificadas las clases y las operaciones para realizar las tareas del caso de uso procedemos con los contratos y colaboraciones.

Contratos de Operaciones y Colaboraciones

Operación: InicBaja( )
Responsabilidades: Preparar el catalogo para la eliminación del registro, Se genera una nueva instancia BajaTrab "BajTrab·
Caso de Uso: Baja de Trabajador
Controlador: Sistema
Precondiciones: El sistema conoce el Id del Supervisor
Postcondiciones: - Se creo la nueva instancia BajaTrab( ) - Se abrió la base de datos para eliminar el registro.

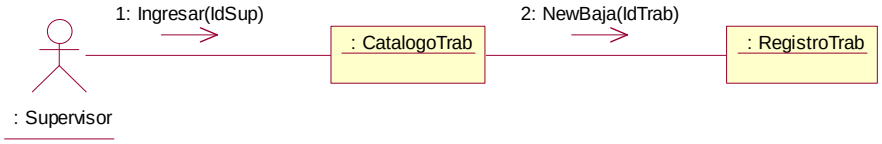


Figura 5.42 Diagrama de Colaboración para lo operación InicBaja()

Operación: SelecTrab(IdTrab)
Responsabilidades: Buscar el registro del trabajador que va a ser eliminado.
Caso de Uso: Baja Trabajador.
Controlador: Sistema
Precondiciones: El sistema conoce el Id del Trabajador.
Postcondiciones: - Se Asocio el registro del trabajador a la instancia BajTrab( )

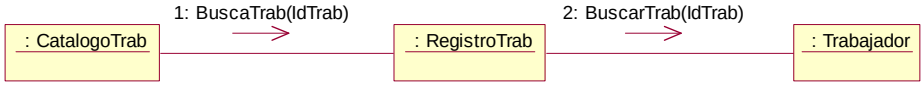


Figura 5.43 Diagrama de colaboración para lo operación SelTrab(IdTrab)

Operación: ConfirmBaja(IdTrab)
Responsabilidades: Llevar a cabo la baja del trabajador.
Caso de Uso: Baja Trabajador.
Controlador: Sistema
Precondiciones: El sistema conoce el Id del Trabajador.
Postcondiciones: - Se Elimino el registro del trabajador del catalogo de trabajadores

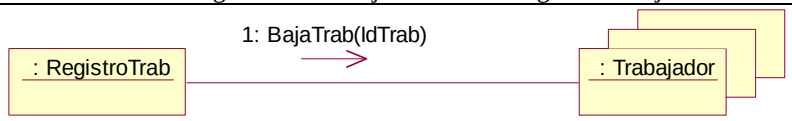


Figura 5.44 Diagrama de Colaboración para la operación ConfirmTrab(IdTrab)

Operación: FinOp( )
Responsabilidades: Cerrar el Catalogo de Trabajadores, Finalizar la Operación.
Caso de Uso: Baja de Trabajador
Controlador: Sistema
Precondiciones: Se realizo la eliminación del registro
Postcondiciones: -Finalizo la Operación de Baja.

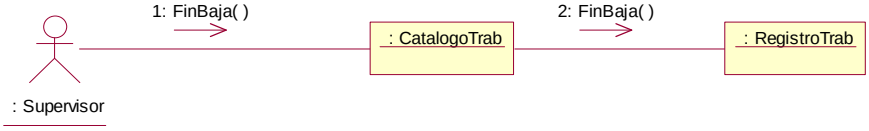


Figura 5.45 Diagrama de Colaboración para lo operación FinOp()

En este caso de uso la interface es la misma que la del caso de uso anterior, los eventos que se activan son diferentes pero las clases involucradas en dichos casos de uso son las mismas por lo que el patrón Fachada nos da como resultado la misma clase fachada.

De igual manera al ser las mismas clases las involucradas el Diagrama de Clases de Diseño es el mismo que el del Caso de Uso Anterior.

5.6 Caso de uso UC6: Modificación de trabajadores en el sistema

De manera similar que en el caso de uso anterior al aplicar el patrón experto para asignar la responsabilidad de saber los datos del registro del empleado que vamos a modificar, se observa que el mismo empleado debe saber sus datos.

También, al aplicar el patrón Creador, observamos que el creador del caso de uso anterior debe ser el encargado de crear la instancia de la clase del trabajador que vamos a modificar, por lo que la responsabilidad de crear dicha instancia del trabajador va a ser la clase "Registro de Trabajador".

De la misma manera se puede inferir que la clase Catalogo de Trabajadores va a ser la encargada del manejo y control de los eventos del caso de uso.

De lo anterior podemos ver que las clases encargadas de dichas tareas van a ser las mismas que en el caso de uso anterior:

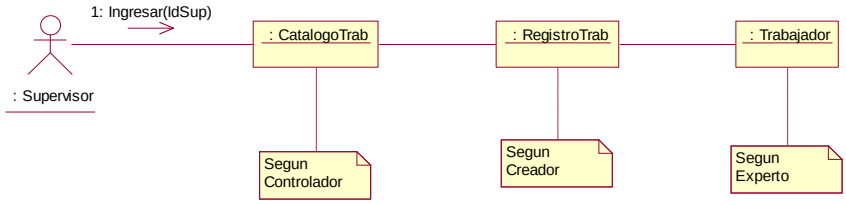


Figura 5.46 Asignación de responsabilidades para modificar datos de trabajadores utilizando patrones

Una vez que tenemos identificadas las clases con sus responsabilidades o tareas, podemos llevar a cabo el diagrama de secuencias para identificar la secuencia de dichas operaciones.

Tras identificadas las clases con sus responsabilidades llevamos a cabo el diagrama de secuencias para identificar la secuencia de dichas operaciones.

Diagrama de Secuencias

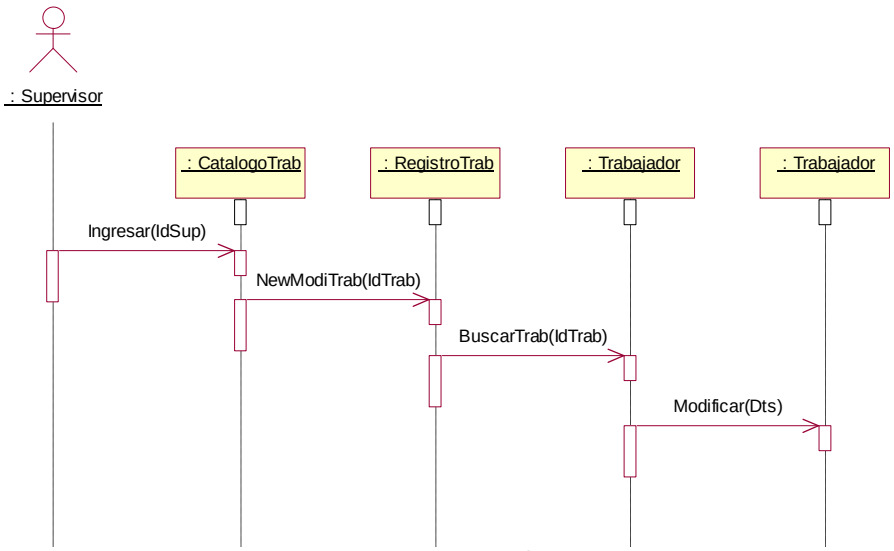


Figura 5.47 Diseño del Diagrama de Secuencias según las responsabilidades asignadas con patrones

Se llevan a cabo los contratos y los diagramas de colaboración.

Contratos de Operaciones y Colaboraciones

Operación: InicMod( )
Responsabilidades: Preparar el catalogo para la Edición del registro, Se genera una nueva instancia ModTrab "MTrab".
Caso de Uso: Modificar Trabajador
Controlador: Sistema
Precondiciones: El sistema conoce el Id del Supervisor
Postcondiciones: - Se genero una nueva instancia ModTrab "MTrab" - Se abrió la Base de Datos para editar el registro.

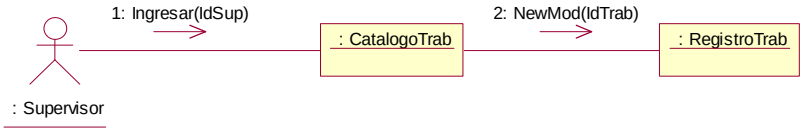


Figura 5.48 Diagrama de Colaboración para la operación InicMod()

Operación: SelTrab(IdTrab)
Responsabilidades: Buscar el registro del trabajador que va a ser modificado.
Caso de Uso: Modificar Trabajador
Controlador: Sistema
Precondiciones: El sistema conoce el Id del trabajador
Postcondiciones: - Se asocio el Id del trabajador con la instancia MTrab.

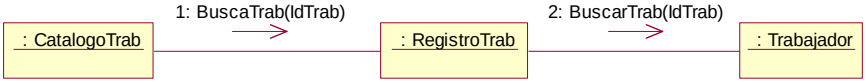


Figura 5.49 Diagrama de Colaboración para la operación SelTrab(IdTrab)

Operación: RegDtsTrab(Dts)
Responsabilidades: Editar el registro del Trabajador
Caso de Uso: Modificar Trabajador
Controlador: Sistema, supervisor
Precondiciones: Esta en Transacción la edicion del registro del trabajador
Postcondiciones: - Se registraron los nuevos datos del Trabajador en el catalogo

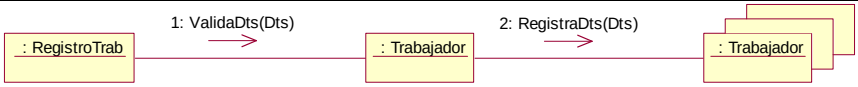


Figura 5.50 Diagrama de Colaboración para lo operación RegDtsTrab(Dts)

Operación: FinMod( )
Responsabilidades: Cerrar el Catalogo de Trabajadores, Finalizar la Operación.
Caso de Uso: Modificar Trabajador.
Controlador: Sistema
Precondiciones: Se validaron los nuevos datos del trabajador
Postcondiciones: - Se cerro la operación ModifTrab( )

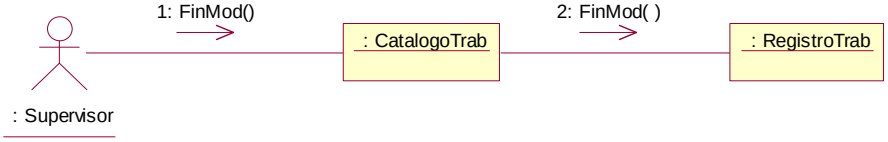


Figura 5.51 Diagrama de Colaboración para lo operación FinMod()

De la misma manera que en los dos casos de uso anteriores la interfase es la misma, lo que cambia sen los eventos que llevan a cabo las operaciones de este caso de uso, y las relaciones entre las clases son las mismas, por lo que la fachada de eventos y el Diagrama de Clases de Diseño son los mismos, con sus respectivas operaciones para este caso de uso.

5.7 Caso de uso UC7: Alta de unidades en el sistema

Aplicando patrones GRASP vamos a asignar responsabilidades para llevar a cabo el Alta de un nuevo registro de una unidad.

Primero aplicando el patrón experto vamos a asignar la responsabilidad de saber los datos a ser registrados a una clase “Unidad”, ya que la unidad seria la que sabe sus propios datos, como modelo, año, número de serie, etc.

Luego, aplicando el patrón Creador, vamos a utilizar una clase registro, ya que la operación va a ser la registrar una nueva Unidad, por lo que la responsabilidad de crear la instancia de un nuevo trabajador va a ser una clase “Registro de Unidad”.

Finalmente, para el manejo y control de los eventos del caso de uso vamos a utilizar una clase “Catalogo de Unidades” ya que haciendo una analogía con el mundo real, seria alguien encargado del catalogo de unidades, el encargado de llevar el registro de estos, así como saber los movimientos de las unidades dentro de la empresa.

De lo anterior podemos ver que las clases encargadas de dichas tareas quedan así:

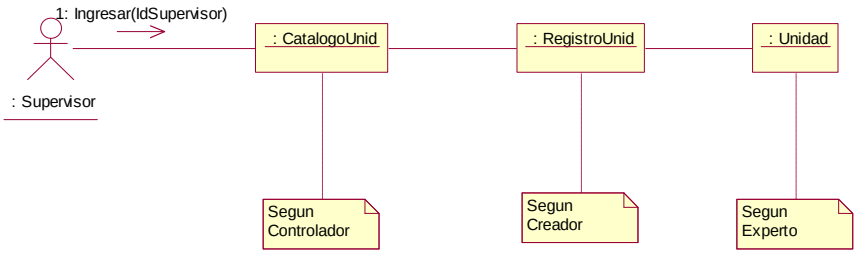


Figura 5.52 Asignación de responsabilidades para el alta de unidades en el sistema

Una vez identificadas las clases con sus responsabilidades o tareas, podemos llevar a cabo el diagrama de secuencias para identificar la secuencia de dichas operaciones.

Diagrama de Secuencias

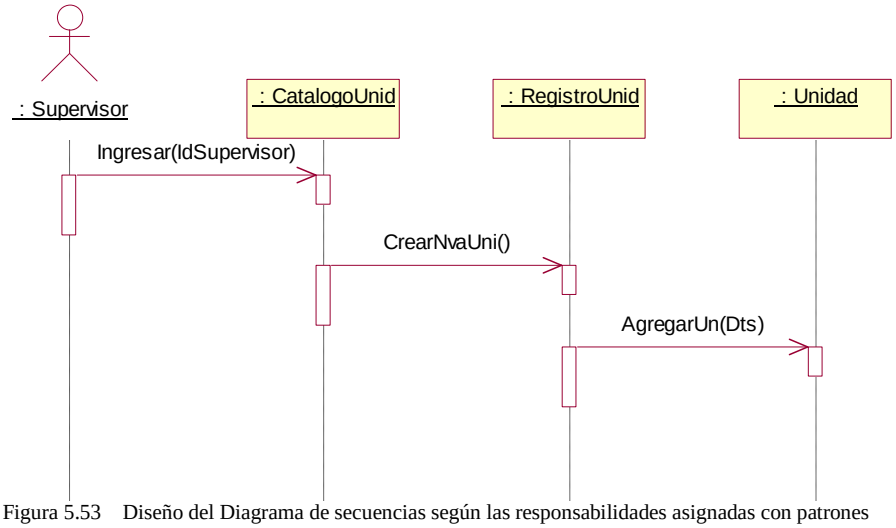


Figura 5.53 Diseño del Diagrama de secuencias según las responsabilidades asignadas con patrones

Si observamos este diagrama y lo comparamos con el que habíamos realizado para este caso de uso en el capitulo anterior, se puede observar como las tareas están repartidas y no es solo una la encargada de hacer todo, con lo que se favorece la alta cohesión y el bajo acoplamiento.

Con las clases que hemos identificado y con las operaciones que obtuvimos en el capitulo anterior para este caso de uso vamos a desarrollar los contratos de operaciones y los diagramas de colaboración.

Contratos de Operaciones y Colaboraciones

Operación: InicAltaUn( )
Responsabilidades: Iniciar el registro de una nueva unidad, Preparar el catalogo de Unidades para el nuevo registro.
Caso de Uso: Alta de Unidad
Controlador: Sistema
Precondiciones: El sistema conoce el ID del Supervisor
Postcondiciones: - Se genero una nueva instancia de AltaUnid "AlUni" - Se abrió el Catalogo de Unidades

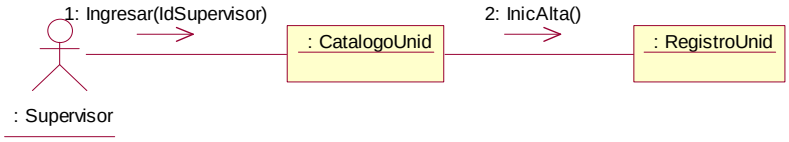


Figura 5.54 Diagrama de Colaboración para lo operación InicAltaUn()

Operación: RegDtsUnid(Dts)
Responsabilidades: Generar el registro de la nueva unidad en el catalogo.
Caso de Uso: Alta de .unidad
Controlador: Sistema, Supervisor.
Precondiciones: Esta en Transición el alta de la unidad.
Postcondiciones: - Se registraron los datos de la nueva unidad en el catalogo.

Figura 5.55 Diagrama de Colaboración para la operación RegDtsUnid(Dts)

Operación: FinAlta( )
Responsabilidades: Cerrar el catalogo de unidades, Finalizar la Operación.
Caso de Uso: Alta Unidad
Controlador: Sistema
Precondiciones: Se validaron los datos de la nueva unidad.
Postcondiciones: Se cerró la operación del Alta de unidad.

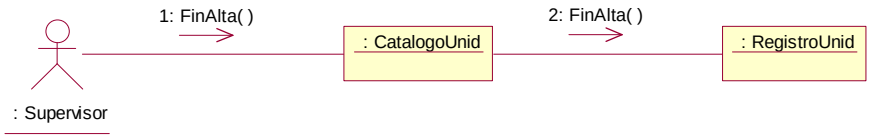


Figura 5.56 Diagrama de Colaboración para la operación FinAlta()

Con el patrón fachada vamos a generar la clase que va a ser la encargada de recibir los eventos de la interface, y las operaciones para llevar a cabo las tareas para realizar el caso de uso como se requiere.

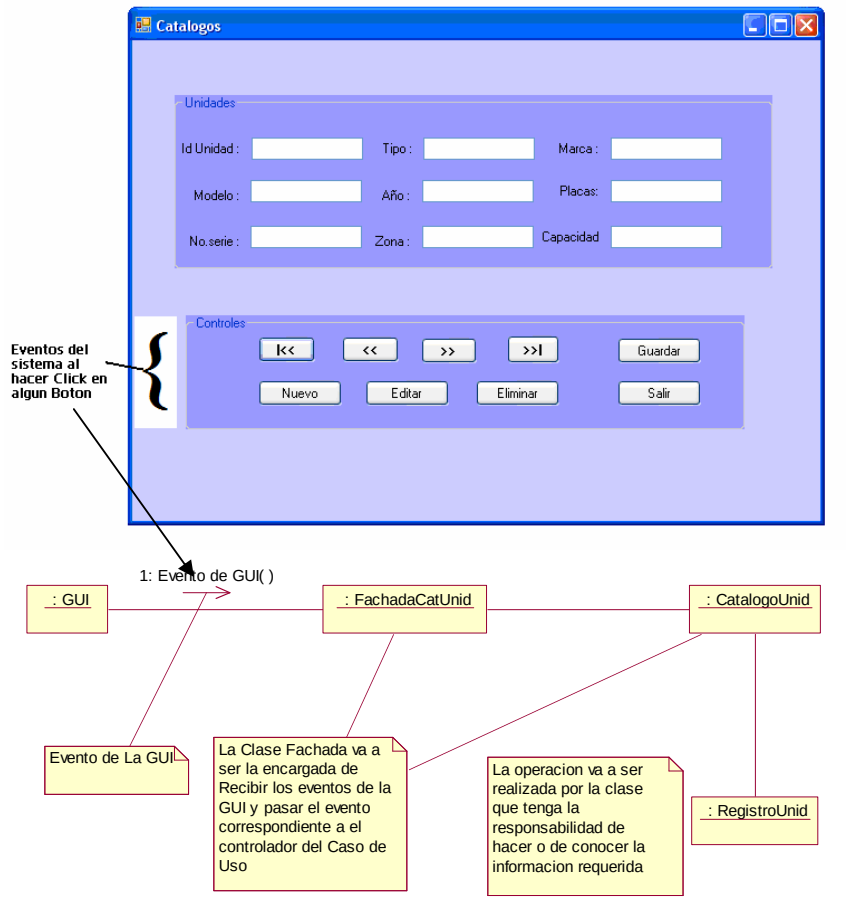


Figura 5.57 Identificación de los eventos de la GUI asignados a la clase fachada del sistema para alta de unidades

Identificadas las clases, sus colaboraciones y las tareas que estas llevan a cabo tenemos el Diagrama de Clases de Diseño.

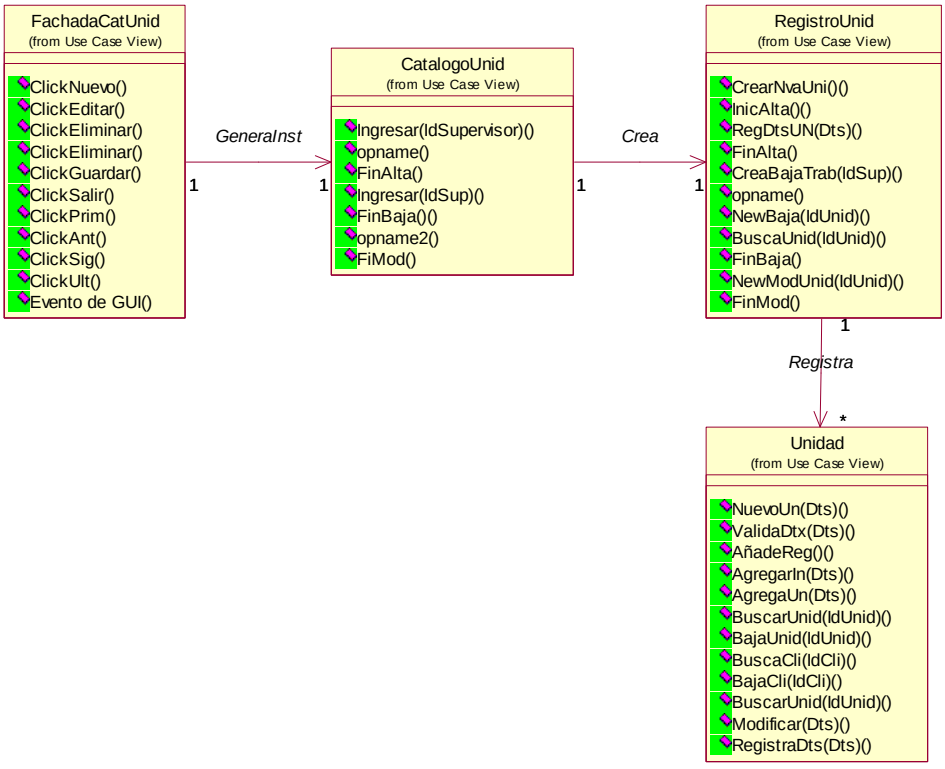


Figura 5.58 Diagrama de Clases de Diseño para el caso de uso alta de unidades

5.8 Caso de uso UC8: Baja de unidades en el sistema

Nuevamente se aplica el patrón experto para asignar la responsabilidad de saber los datos del registro de la unidad que va a ser dada de baja, se observa que al igual que en el caso de uso anterior es la misma Unidad el que debe saber sus datos.

Después, al aplicar el patrón creador, observamos que también el creador del caso de uso anterior debe ser el encargado de crear la instancia de la clase de la unidad que va a ser dada de baja, por lo que la responsabilidad de crear dicha instancia de la unidad va a ser una clase "Registro de Unidad".

De igual forma se puede inferir que una clase de Catalogo de Unidades va a ser la encargada del manejo y control de los eventos del caso de uso.

De lo anterior podemos ver que las clases encargadas de dichas tareas van a ser las mismas que en el caso de uso anterior:

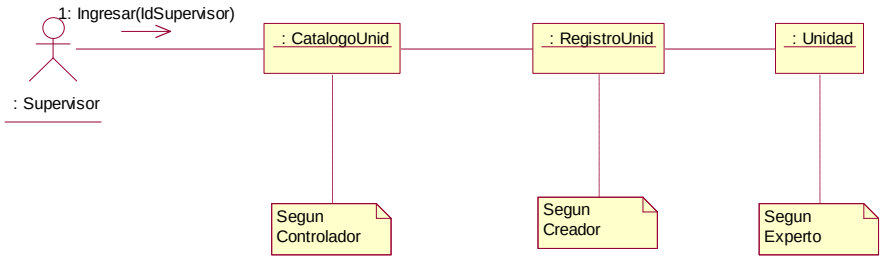


Figura 5.59 Asignación de Responsabilidades para el caso de uso utilizando patrones

Así pues, podemos llevar a cabo el diagrama de secuencias para identificar como se van a llevar a cabo dichas operaciones.

Diagrama de Secuencias

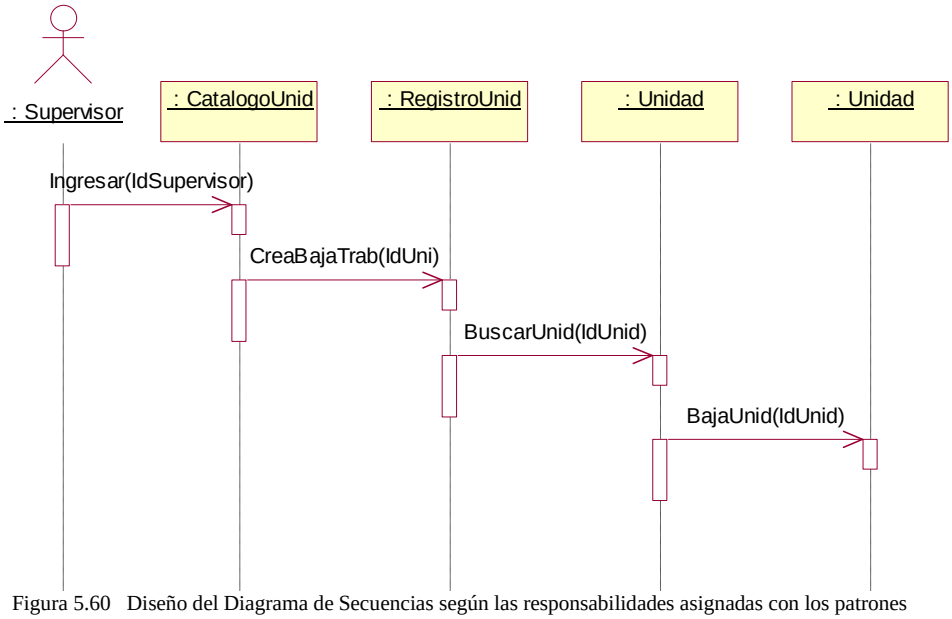


Figura 5.60 Diseño del Diagrama de Secuencias según las responsabilidades asignadas con los patrones

Identificadas las clases y las operaciones para realizar las tareas del caso de uso procedemos con los contratos y colaboraciones

Contratos de Operaciones y Colaboraciones

Operación: InicBaja( )
Responsabilidades: Preparar el catalogo para la eliminación del registro, Se genera una nueva instancia BajaUni "bajUnid"
Caso de Uso: Baja de Unidad
Controlador: Sistema
Precondiciones: El sistema conoce el Id del Supervisor
Postcondiciones: - Se creo la nueva instancia bajUnid( ) - Se abrió la base de datos para eliminar el registro.

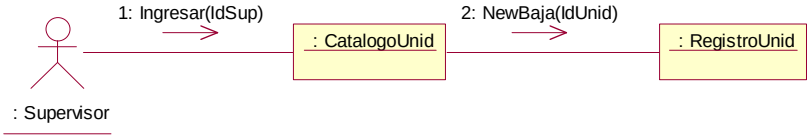


Figura 5.61 Diagrama de Colaboración para la operación InicBaja()

Operación: SelecUni(IdUni)
Responsabilidades: Buscar el registro de la unidad que va a ser eliminado.
Caso de Uso: Baja Unidad.
Controlador: Sistema
Precondiciones: El sistema conoce el Id de la Unidad.
Postcondiciones: - Se Asocio el registro de la Unidad a la instancia bajUnid( )

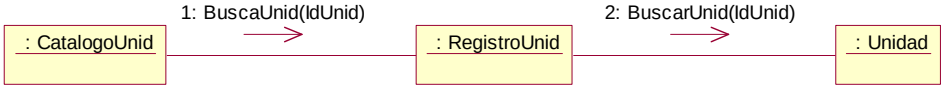


Figura 5.62 Diagrama de Colaboración para la operación SelecUni(IdUnid)

Operación: ConfirmBaja(IdUnid)
Responsabilidades: Llevar a cabo la baja de la unidad.
Caso de Uso: Baja Unidad.
Controlador: Sistema
Precondiciones: El sistema conoce el Id de la Unidad.
Postcondiciones: - Se Elimino el registro de la Unidad del catalogo de Unidades.

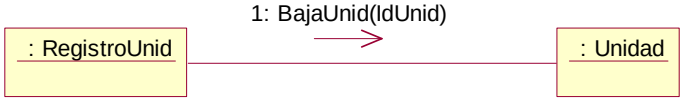


Figura 5.63 Diagrama de Colaboración para la operación ConfirmBaja(IdUnid)

Operación: FinOp( )
Responsabilidades: Cerrar el Catalogo de Unidades, Finalizar la Operación.
Caso de Uso: Baja de Unidad
Controlador: Sistema
Precondiciones: Se realizo la eliminación del registro
Postcondiciones: -Finalizo la Operación de Baja.

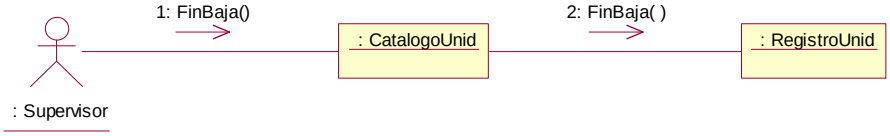


Figura 5.64 Diagrama de Colaboración para la operación FinOp()

En este caso de uso la interface es la misma que la del caso de uso anterior, los eventos que se activan son diferentes pero las clases involucradas en dichos casos de uso son las mismas por lo que el patrón Fachada nos da como resultado la misma clase fachada.

De la misma manera al ser las mismas clases las involucradas el Diagrama de Clases de Diseño es el mismo también.

5.9 Caso de uso UC9: Modificación de unidades en el sistema

De manera similar que en el caso de uso anterior al aplicar el patrón experto para asignar la responsabilidad de saber los datos del registro del empleado que vamos a modificar, se observa que la unidad debe saber sus datos.

También, al aplicar el patrón Creador, observamos que el creador del caso de uso anterior debe ser el encargado de crear la instancia de la clase del trabajador que vamos a modificar, por lo que la responsabilidad de crear dicha instancia del trabajador va a ser una clase Registro de Trabajador.

De la misma manera se puede inferir que una clase Catalogo de Unidades va a ser la encargada del manejo y control de los eventos del caso de uso.

De lo anterior podemos ver que las clases encargadas de dichas tareas van a ser las mismas que en el caso de uso anterior:

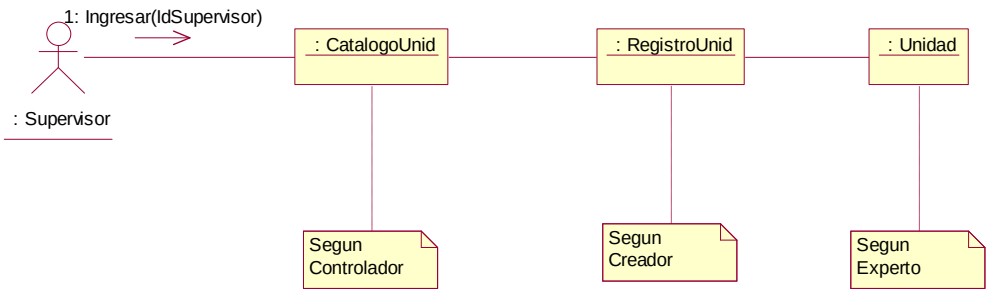


Figura 5.65 Asignación de Responsabilidades para modificar datos de unidades utilizando patrones

Una vez localizadas las clases con sus responsabilidades, podemos llevar a cabo el diagrama de secuencias para identificar la secuencia de dichas operaciones.

Diagrama de Secuencias

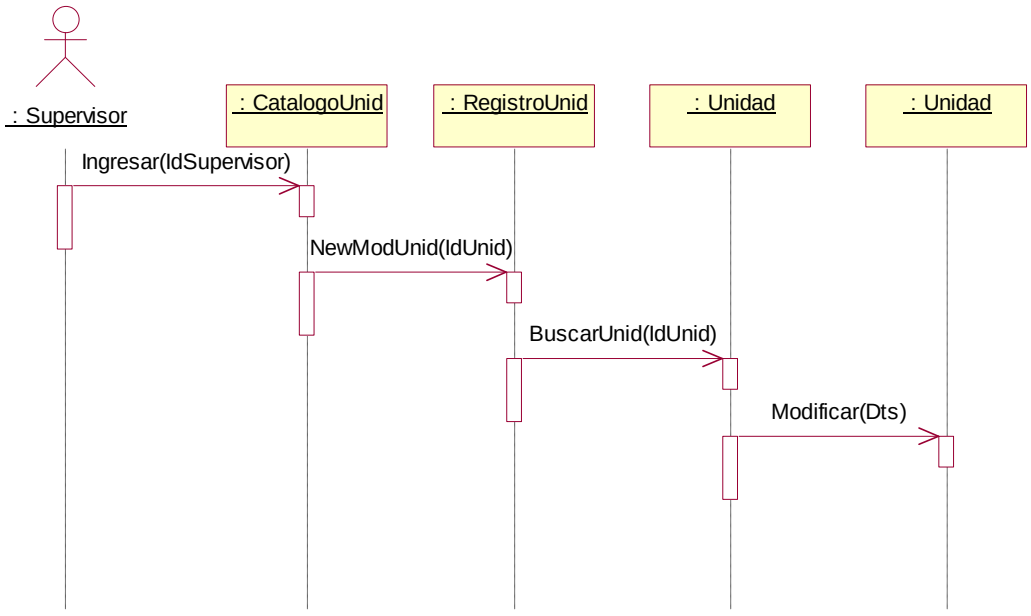


Figura 5.66 Diseño del Diagrama de Secuencias según las responsabilidades asignadas con patrones

A continuación llevamos a cabo los contratos y colaboraciones del caso de uso.

Contratos de Operaciones y Colaboraciones

Operación: InicMod( )
Responsabilidades: Preparar el catalogo para la Edición del registro, Se genera una nueva instancia ModUnid "MUnid".
Caso de Uso: Modificar Unidad
Controlador: Sistema
Precondiciones: El sistema conoce el Id del Supervisor
Postcondiciones: - Se generó una nueva instancia ModUnid "MUnid" - Se abrió la Base de Datos para editar el registro.

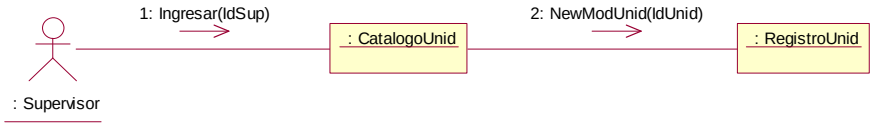


Figura 5.67 Diagrama de Colaboración para la operación InicMod()

Operación: SelUni(IdUni)
Responsabilidades: Buscar el registro de la unidad que va a ser modificado.
Caso de Uso: Modificar Unidad
Controlador: Sistema
Precondiciones: El sistema conoce el Id de la Unidad
Postcondiciones: - Se asocio el Id de la unidad con la instancia MUnid.

Figura 5.68 Diagrama de Colaboración para la operación SelUni(IdUni)

Operación: RegDtsUnid(Dts)
Responsabilidades: Editar el registro de la Unidad
Caso de Uso: Modificar Unidad
Controlador: Sistema, supervisor
Precondiciones: Esta en Transacción la edicion del registro de la unidad
Postcondiciones: - Se registraron los nuevos datos de la unidad en el catalogo

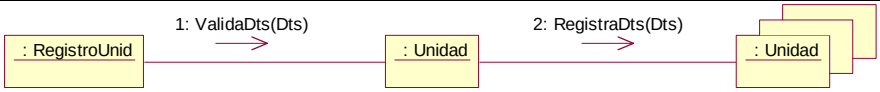


Figura 5.69 Diagrama de Colaboración para la operación RegDtsUnid(Dts)

Operación: FinMod( )
Responsabilidades: Cerrar el Catalogo de unidades, Finalizar la Operación.
Caso de Uso: Modificar Unidad.
Controlador: Sistema
Precondiciones: Se validaron los nuevos datos de la unidad
Postcondiciones: - Se cerro la operación ModifUnid( )

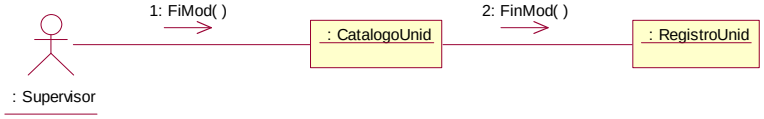


Figura 5.70 Diagrama de Colaboración para la operación FinMod()

De la misma manera que en los dos casos de uso anteriores la interface es la misma, lo que cambia son los eventos que llevan a cabo las operaciones de este caso de uso, las relaciones entre las clases son las mismas, por lo que la fachada de eventos y el Diagrama de Clases de Diseño son los mismos, con sus respectivas operaciones para este caso de uso.

5.10 Caso de uso UC10: Alta de clientes en el sistema

Aplicando patrones GRASP vamos a asignar responsabilidades para llevar a cabo el Alta de un nuevo registro de un Cliente para poder llevar a cabo una venta a crédito para este cliente.
Primero aplicando el patrón experto vamos a asignar la responsabilidad de saber los datos a ser registrados, con lo que una instancia del Cliente a utilizar seria el que tendría la información del mismo cliente, así que una clase Cliente seria el experto de saber los datos del nuevo trabajador que va a ser dado de alta, como son su nombre, apellidos, y RFC, etc.
Luego, aplicando el patrón Creador, vamos a utilizar una clase registro, ya que la operación va a ser la de registrar un nuevo Cliente, por lo que la responsabilidad de crear la instancia de un nuevo Cliente va a ser una clase Registro de Cliente.

Finalmente, para el manejo y control de los eventos del caso de uso vamos a utilizar una clase "Catalogo de Clientes" ya que haciendo una analogía con el mundo real, seria alguien encargado del catalogo de Clientes, el encargado de llevar el registro de estos, así como saber las ventas que se les registran a estos clientes.

De lo anterior podemos ver que las clases encargadas de las responsabilidades quedan así:

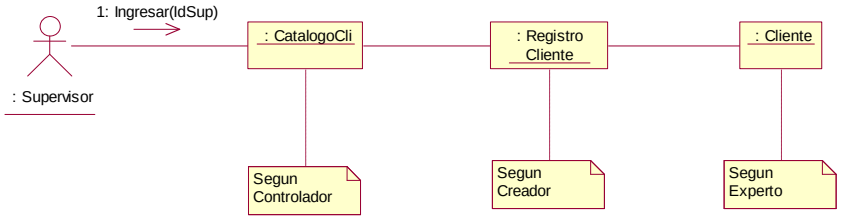


Figura 5.71 Asignación de Responsabilidades para el alta de clientes en el sistema

Una vez que tenemos identificadas las clases con sus responsabilidades o tareas, podemos llevar a cabo el diagrama de secuencias para observar como se llevan a cabo las operaciones.

Diagrama de Secuencias

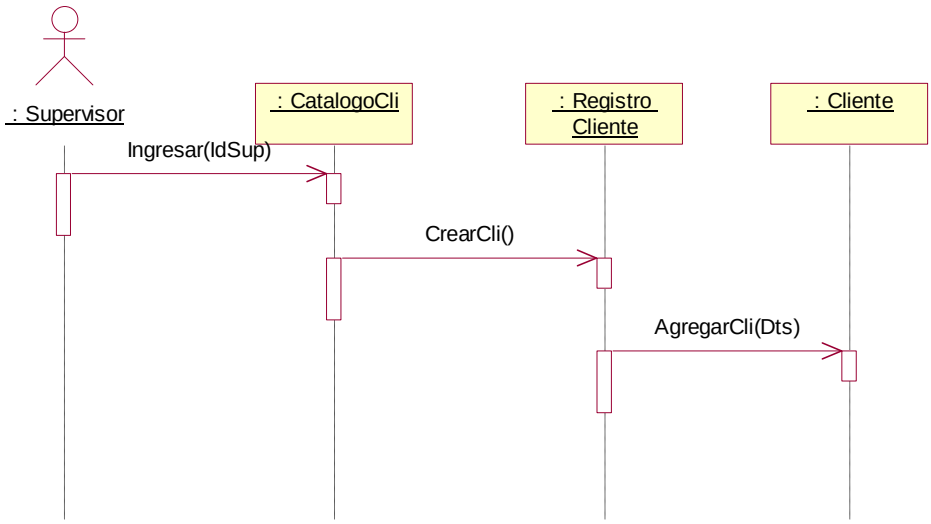


Figura 5.71 Diseño del Diagrama de Secuencias según las responsabilidades asignadas con patrones

Si observamos este diagrama y lo comparamos con el que habíamos realizado en el capitulo anterior se puede observar como las tareas están repartidas y no es solo una la encargada de hacer todo, con lo que se favorece la alta cohesión y el bajo acoplamiento.

Con las clases que hemos visto que vamos a requerir y con las operaciones que obtuvimos en el capitulo anterior para este caso de uso vamos a desarrollar los contratos de operaciones y los diagramas de colaboración.

Contratos de Operaciones y Colaboraciones

Operación: InicAltaCli( )
Responsabilidades: Iniciar el registro de un nuevo cliente, Preparar el catalogo de Clientes para el nuevo registro.
Caso de Uso: Alta de Cliente
Controlador: Sistema
Precondiciones: El sistema conoce el ID del Supervisor
Postcondiciones: - Se genero una nueva instancia de AltaCli "AlCli" - Se abrió el Catalogo de Clientes

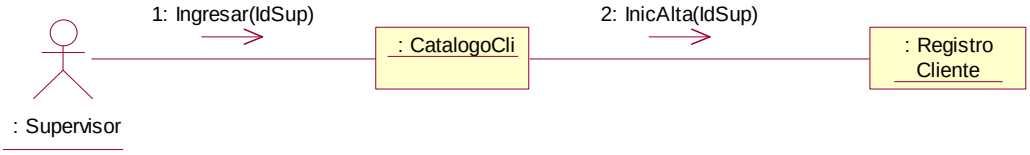


Figura 5.72 Diagrama de Colaboración para la operación InicAltaCli()

Operación: RegDtsCli(Dts)
Responsabilidades: Generar el registro del nuevo cliente en el catalogo.
Caso de Uso: Alta de Cliente.
Controlador: Sistema, Supervisor.
Precondiciones: Esta en Transición el alta del cliente.
Postcondiciones: - Se registraron los datos del nuevo cliente en el catálogo.

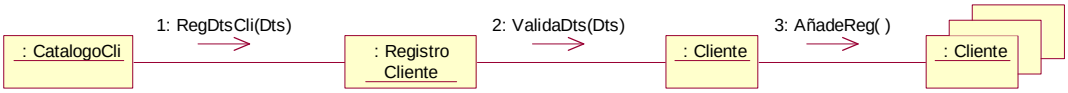


Figura 5.73 Diagrama de Colaboración para la operación RegDtsCli(Dts)

Operación: FinAlta( )
Responsabilidades: Cerrar el catalogo de Clientes, Finalizar la Operación.
Caso de Uso: Alta de Cliente
Controlador: Sistema
Precondiciones: Se validaron los datos del nuevo cliente.
Postcondiciones: Se cerró la operación del Alta de cliente.

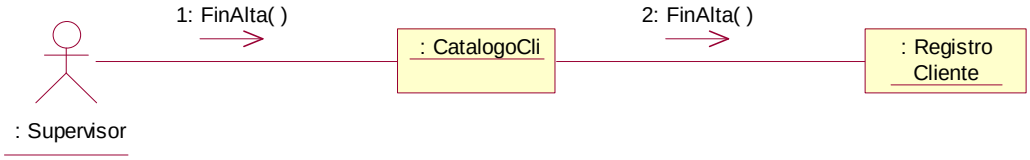


Figura 5.74 Diagrama de Colaboración para la operación FinAlta()

Con ayuda del patrón fachada vamos a generar la clase que va a ser la encargada de recibir los eventos de la interface, y sus operaciones para llevar a cabo las tareas y así cumplir con el caso de uso.

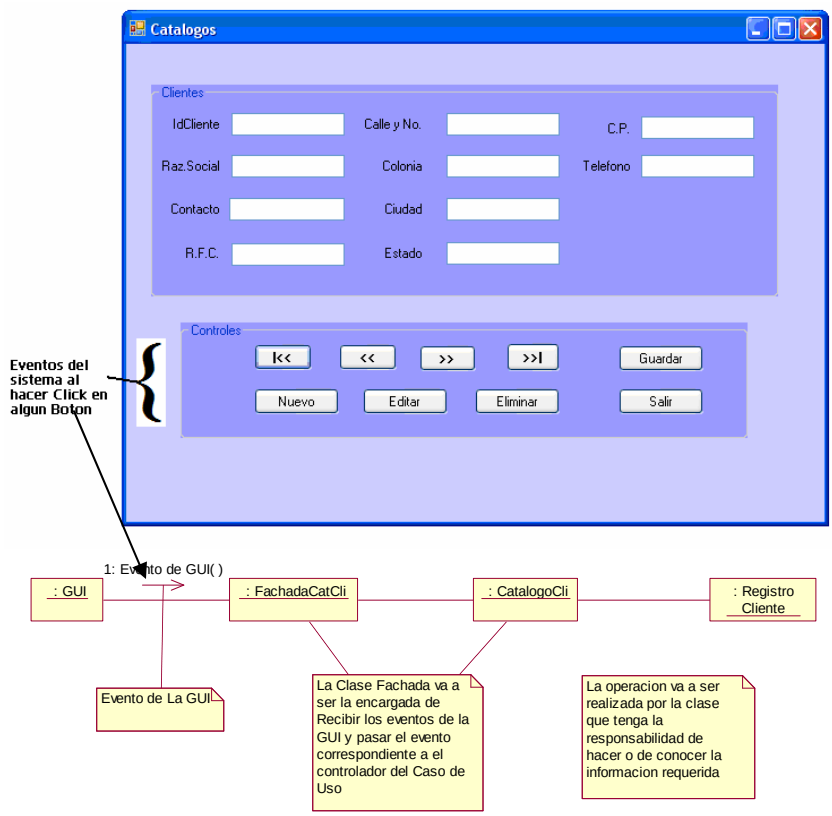


Figura 5.75 Localización de los Eventos de la GUI para el alta de empleados y asignación a la clase fachada

Identificadas las clases, sus colaboraciones y las tareas que estas llevan a cabo tenemos el Diagrama de Clases de Diseño.

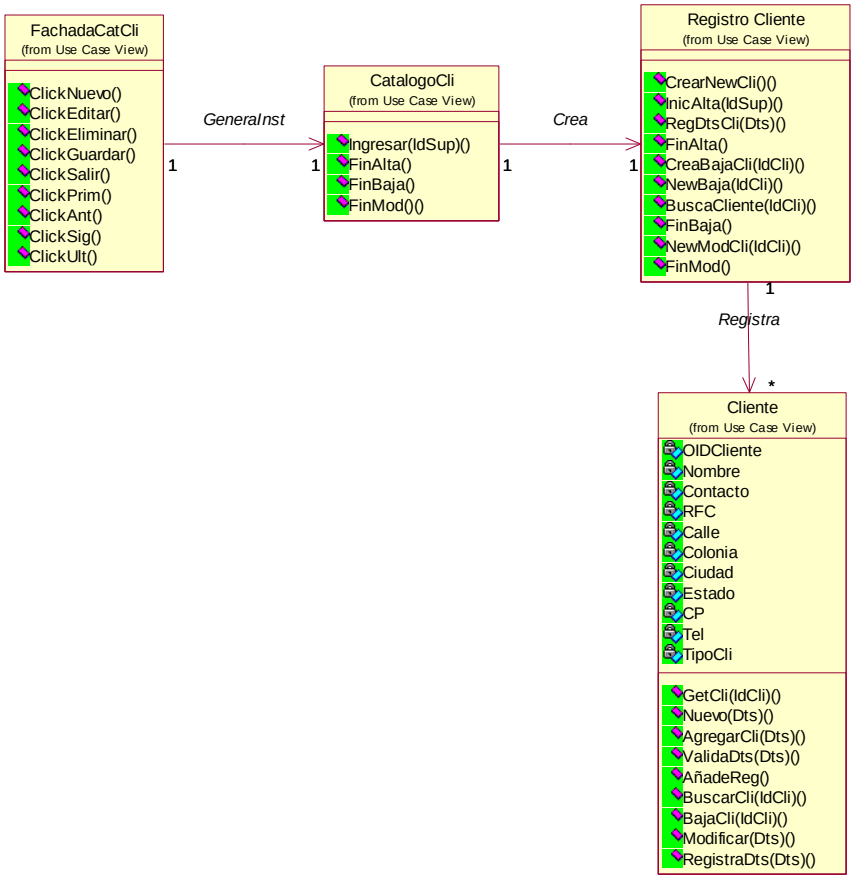


Figura 5.76 Diagrama de Clases de Diseño para el caso de uso alta de clientes

5.11 Caso de uso UC11: Baja de clientes en el sistema

Nuevamente tratando de aplicar el patrón experto para asignar la responsabilidad de saber los datos del registro del Cliente que va a ser dado de baja, se observa que al igual que en el caso de uso anterior es el mismo Cliente el que debe saber sus datos.

Después, al aplicar el patrón Creador, observamos que también el creador del caso de uso anterior debe ser el encargado de crear la instancia de la clase del cliente que va a ser dado de baja, por lo que la responsabilidad de crear dicha instancia del cliente va a ser una clase Registro de Cliente.

De manera similar se puede inferir que la clase Catalogo de Clientes va a ser la encargada del manejo y control de los eventos del caso de uso.

De lo anterior podemos ver que las clases encargadas de dichas tareas van a ser las mismas que en el caso de uso anterior:

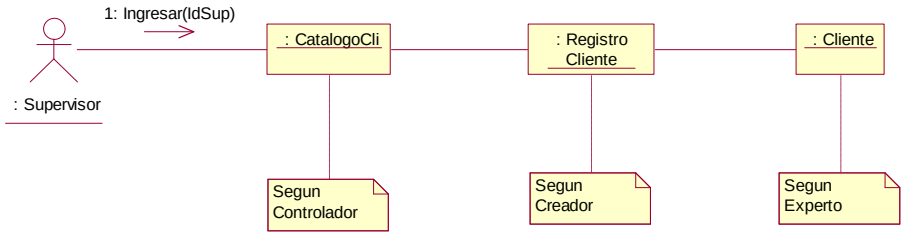


Figura 5.77 Asignación de Responsabilidades para el caso de uso utilizando patrones

Una vez que tenemos identificadas las clases con sus responsabilidades o tareas, podemos llevar a cabo el diagrama de secuencias para identificar la secuencia de dichas operaciones.

Diagrama de Secuencias

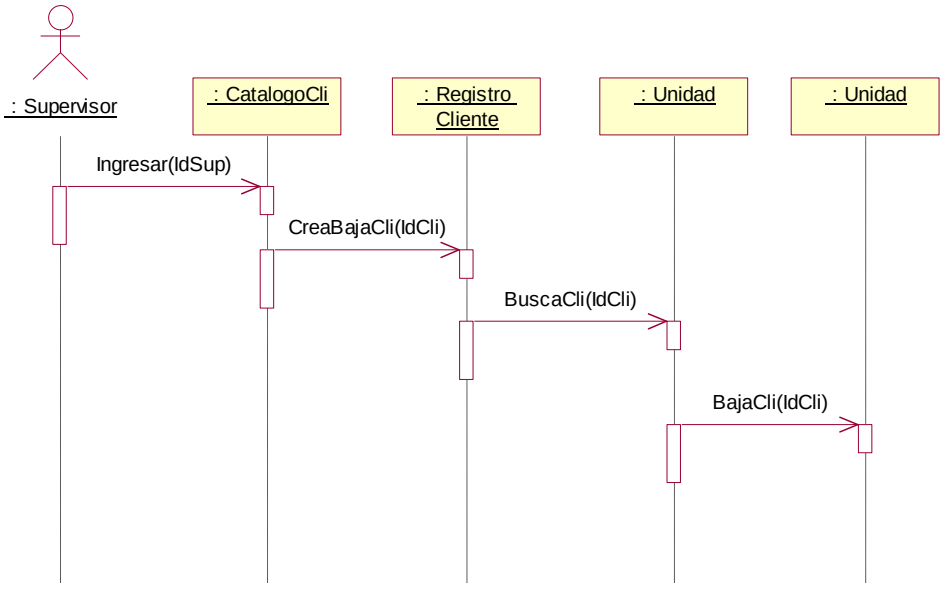


Figura 5.78 Diseño del Diagrama de Secuencias según las responsabilidades asignadas con patrones

Identificadas las clases y las operaciones para realizar las tareas del caso de uso procedemos con los contratos y colaboraciones.

Contratos de Operaciones y Colaboraciones

Operación: InicBaja( )
Responsabilidades: Preparar el catalogo para la eliminación del registro, Se genera una nueva instancia BajaCli "bajCli"
Caso de Uso: Baja de Unidad
Controlador: Sistema
Precondiciones: El sistema conoce el Id del Supervisor
Postcondiciones: - Se creo la nueva instancia bajCli( ) - Se abrió la base de datos para eliminar el registro.

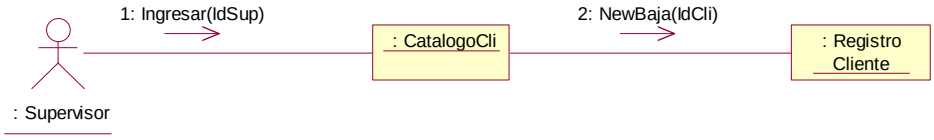


Figura 5.79 Diagrama de Colaboración para la operación InicBaja()

Operación: SelecCli(IdCli)
Responsabilidades: Buscar el registro del cliente que va a ser eliminado.
Caso de Uso: Baja de Clientes.
Controlador: Sistema
Precondiciones: El sistema conoce el Id del Cliente.
Postcondiciones: - Se Asocio el registro del Cliente a la instancia bajCli( )

Figura 5.80 Diagrama de Colaboración para la operación SelecCli(IdCli)

Operación: ConfirmBaja(IdCli)
Responsabilidades: Llevar a cabo la baja del Cliente.
Caso de Uso: Baja Trabajador.
Controlador: Sistema
Precondiciones: El sistema conoce el Id del Cliente.
Postcondiciones: - Se Eliminó el registro del Cliente del catalogo de Clientes

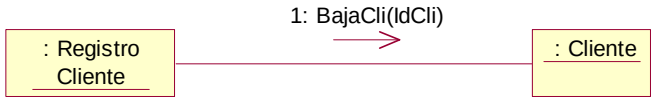


Figura 5.81 Diagrama de Colaboración para la operación ConfirmBaja(IdCli)

Operación: FinOp( )
Responsabilidades: Cerrar el Catalogo de Clientes, Finalizar la Operación.
Caso de Uso: Baja de Clientes
Controlador: Sistema
Precondiciones: Se realizo la eliminación del registro
Postcondiciones: -Finalizo la Operación de Baja.

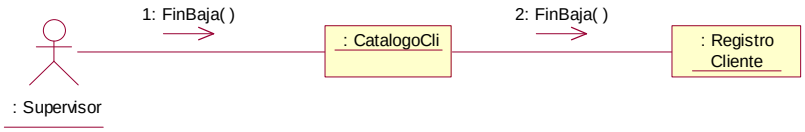


Figura 5.82 Diagrama de Colaboración para la operación FinOp()

En este caso de uso la interface es la misma que la del caso de uso anterior, los eventos que se activan son diferentes pero las clases involucradas en dichos casos de uso son las mismas por lo que el patrón Fachada nos da como resultado la misma clase fachada que en el caso de uso anterior.

De la misma manera al ser las mismas clases las involucradas el Diagrama de Clases de Diseño es el mismo.

5.12 Caso de uso UC12: Modificación de clientes en el sistema

De manera similar que en el caso de uso anterior al aplicar el patrón experto para asignar la responsabilidad de saber los datos del registro del Cliente que vamos a modificar, se observa que el mismo empleado debe saber sus datos.

También, al aplicar el patrón Creador, observamos que el creador del caso de uso anterior debe ser el encargado de crear la instancia de la clase del Cliente que vamos a modificar, por lo que la responsabilidad de crear dicha instancia del Cliente va a ser la clase Registro de Cliente.

De la misma manera se puede inferir que una clase Catalogo de Clientes va a ser la encargada del manejo y control de los eventos del caso de uso.

De lo anterior podemos ver que las clases encargadas de las tareas en este caso de uso van a ser las mismas que en el caso de uso anterior:

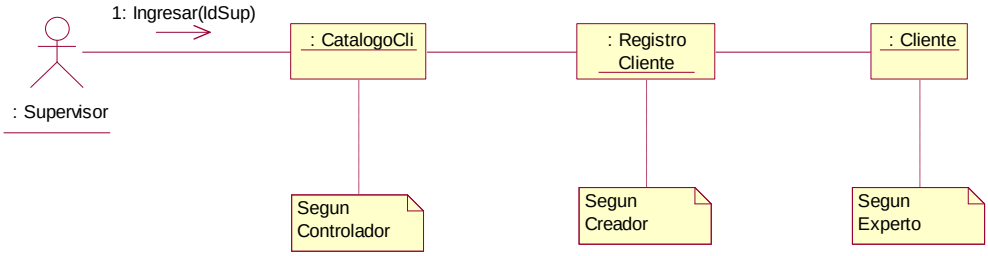


Figura 5.83 Asignación de Responsabilidades para modificar datos de los clientes utilizando patrones

Una vez que tenemos identificadas las clases con sus responsabilidades o tareas, podemos llevar a cabo el diagrama de secuencias para identificar el funcionamiento de las operaciones.

Diagrama de Secuencias

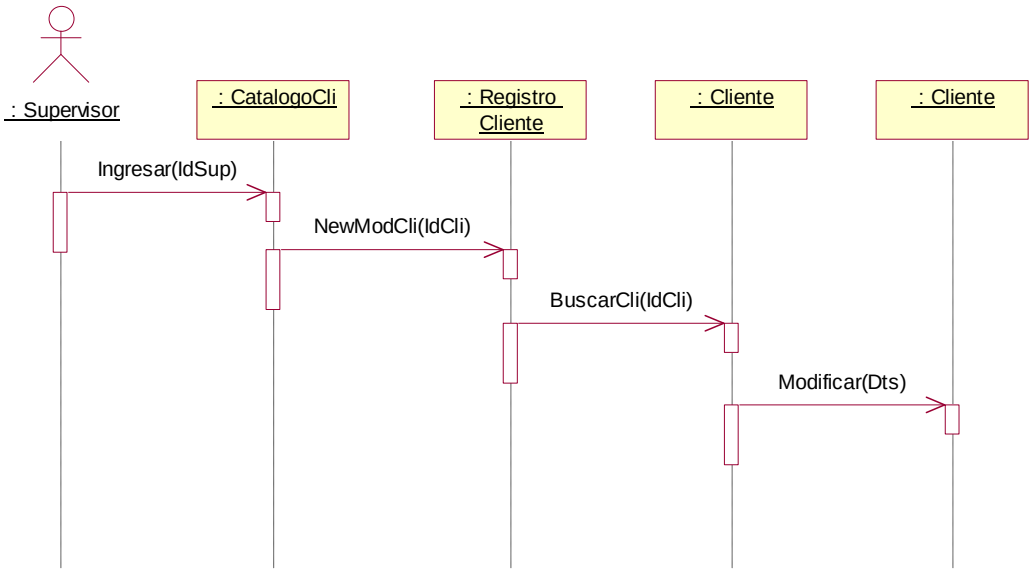


Figura 5.84 Diseño del Diagrama de Secuencias según las responsabilidades asignadas con patrones

Continuamos con los contratos y los diagramas de colaboración.

Contratos de Operaciones y Colaboraciones

Operación: InicMod( )
Responsabilidades: Preparar el catalogo para la Edición del registro, Se genera una nueva instancia ModCli "MCli".
Caso de Uso: Modificar Cliente
Controlador: Sistema
Precondiciones: El sistema conoce el Id del Supervisor
Postcondiciones: - Se genero una nueva instancia ModCli "MCli" - Se abrió la Base de Datos para editar el registro.

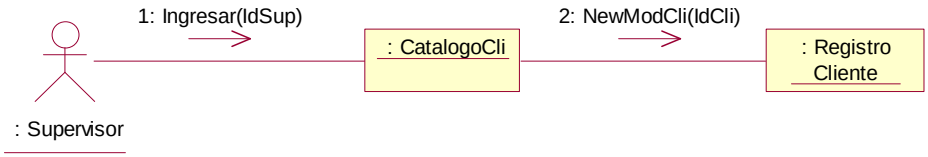


Figura 5.85 Diagrama de Colaboración para la operación InicMod()

Operación: SelCli(IdCli)
Responsabilidades: Buscar el registro del Cliente que va a ser modificado.
Caso de Uso: Modificar Cliente
Controlador: Sistema
Precondiciones: El sistema conoce el Id del Cliente
Postcondiciones: - Se asocio el Id del cliente con la instancia MCli.

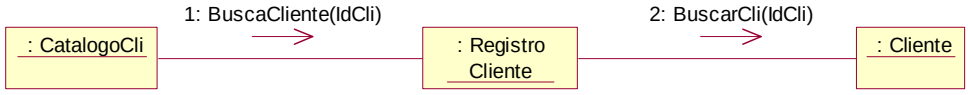


Figura 5.86 Diagrama de Colaboración para la operación SelCli(IdCli)

Operación: RegDtsCli(Dts)
Responsabilidades: Editar el registro del Cliente
Caso de Uso: Modificar Cliente
Controlador: Sistema, supervisor
Precondiciones: Esta en Transacción la edicion del registro del cliente
Postcondiciones: - Se registraron los nuevos datos del cliente en el catalogo

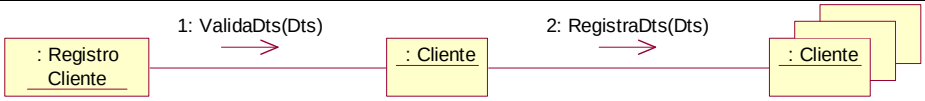


Figura 5.87 Diagrama de Colaboración para la operación RegDtsCli(Dts)

Operación: FinMod( )
Responsabilidades: Cerrar el Catalogo de clientes, Finalizar la Operación.
Caso de Uso: Modificar Unidad.
Controlador: Sistema
Precondiciones: Se validaron los nuevos datos del cliente
Postcondiciones: - Se cerro la operación ModifCli( )

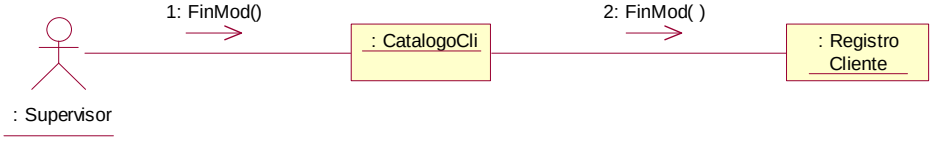


Figura 5.88 Diagrama de Colaboración para la operación FinMod()

De la misma manera que en los dos casos de uso anteriores la interfase es la misma, lo que cambia son los eventos que llevan a cabo las operaciones de este caso de uso, y las relaciones entre las clases son las mismas, por lo que la fachada de eventos y el Diagrama de Clases de Diseño son los mismos, con sus respectivas operaciones para este caso de uso.

5.13 Caso de uso UC13: Obtención de reporte para nómina

Vamos a aplicar patrones GRASP para asignar responsabilidades para llevar a cabo el Reporte de Nómina.

Primero aplicando el patrón experto vamos a asignar la responsabilidad de saber los datos de las ventas que requiere el encargado de nóminas. En este caso seria un reporte de ventas en que contendría la información de las ventas. Con lo que una clase candidata seria Reporte de Ventas

Después, haciendo referencia al patrón creador, vamos a identificar quien seria el encargado de crear la instancia de este reporte de ventas, haciendo referencia a una parte de la definición del Patrón Creador, tenemos que "B contiene objetos de A", en este caso podemos ver que las ventas deben estar contenidas en algún lado y de ahí mostrarlas en el reporte de ventas, por tanto, aplicando el patrón Creador, un contenedor de ventas seria el responsable de crear la instancia del reporte de ventas al ser este el que contendría la información de las ventas a ser mostradas en el reporte, por consiguiente Contenedor de Ventas es una clase candidata a creador de la instancia de Reporte de Ventas.

Finalmente, para el manejo y control de los eventos del caso de uso vamos a tomar una clase Reporteador, el cual va a ser el encargado de tomar los eventos del caso de uso, pues a esta clase le vamos a dar la responsabilidad de representar el escenario de los eventos del reporte.
De lo anterior podemos ver que las clases encargadas de dichas tareas quedan así:

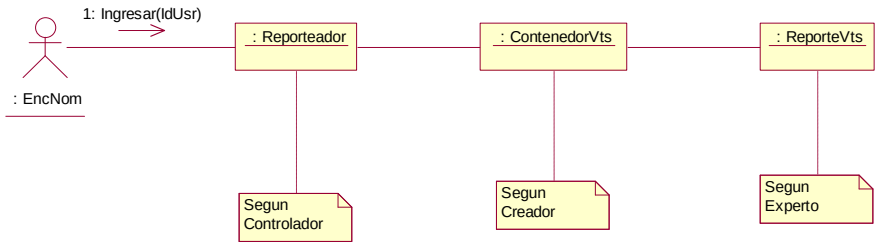


Figura 5.89. Asignación de Responsabilidades para el Reporte de Ventas para Nómina

Una vez que hemos identificado las clases con sus responsabilidades o tareas, podemos llevar a cabo el diagrama de secuencias para identificar como se desarrollan dichas operaciones.

Diagrama de Secuencias

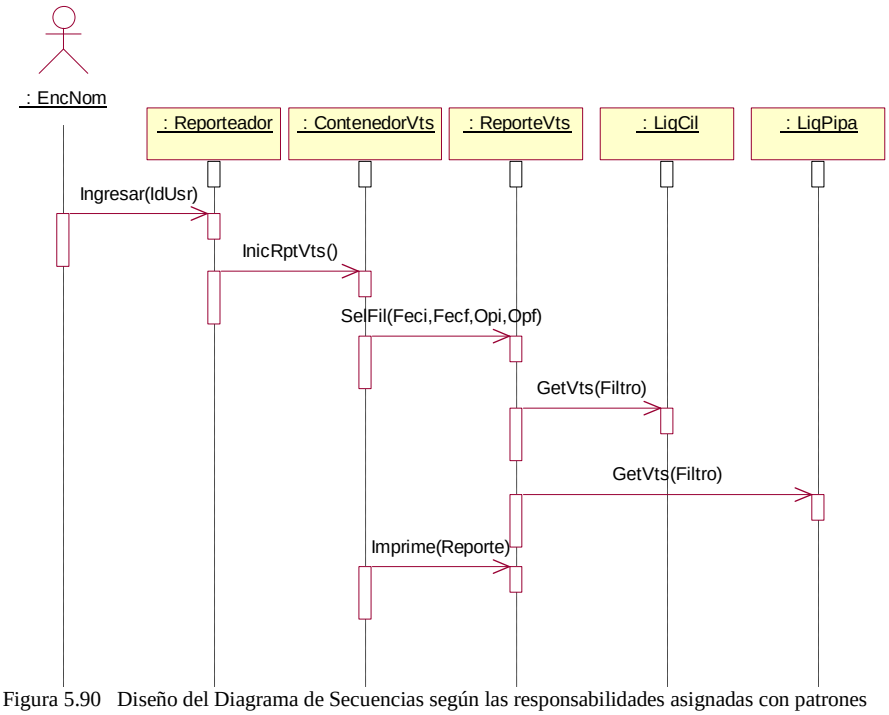


Figura 5.90 Diseño del Diagrama de Secuencias según las responsabilidades asignadas con patrones

Nuevamente haciendo referencia al diagrama de secuencias de este caso de uso en el capitulo anterior podemos observar como el diagrama que aquí hemos desarrollado, las tareas están repartidas y no solo una es la encargada de hacer todo, con lo que se favorece la alta cohesión y el bajo acoplamiento.

Con las clases que hemos identificado y con las operaciones que obtuvimos en el capitulo anterior para este caso de uso vamos a desarrollar los contratos de operaciones y los diagramas de colaboración.

Contratos de Operaciones y Colaboraciones

Operación: InicRepNom( )
Responsabilidades: Abrir la tabla de ventas para extraer la información.
Caso de Uso: Reporte de Ventas Para Nomina
Controlador: Sistema
Precondiciones: El sistema conoce el Id del encargado de nominas
Postcondiciones: - Se Genero una nueva instancia de RptNomi "repNom"

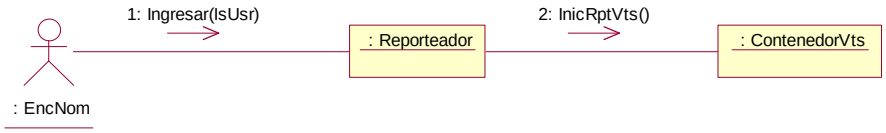


Figura 5.91 Diagrama de Colaboración para la operación InicRepNom()

Operación: SelFiltro(fechini, fechfin, opini, opfin)
Responsabilidades: Registrar los Filtros a ser aplicados para generar el reporte.
Caso de Uso: Reporte de Ventas para Nómina.
Controlador: Sistema, Enc. Nómina
Precondiciones: La generación del reporte esta en Proceso.
Postcondiciones: - Se registraron los filtros a ser aplicados para la generación del reporte.

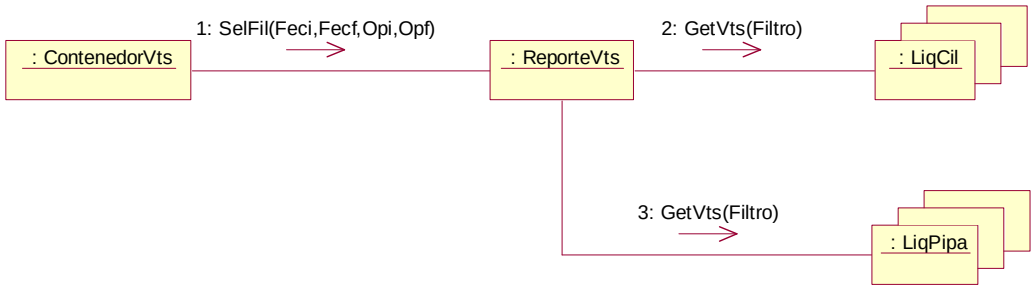


Figura 5.92 Diagrama de Colaboración para la operación SelFiltrb(fechini, fechfin, opini, opfin)

Operación: ImprimeRpt( )
Responsabilidades: Mandar a imprimir el reporte generado
Caso de Uso: Reporte de ventas para nomina
Controlador: Sistema
Precondiciones: El reporte se ha generado
Postcondiciones: - Se mando a imprimir el reporte. - Se cerro la operación repNom( )

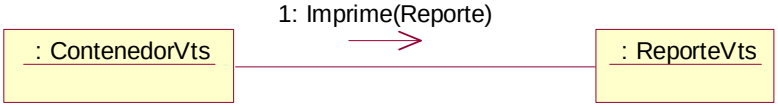


Figura 5.93 Diagrama de Colaboración para la operación ImprimeRpt()

Con el patrón fachada vamos a generar la clase que va a ser la encargada de recibir los eventos de la interface y sus operaciones, para llevar a cabo las tareas para realizar el caso de uso como se requiere. Aquí identificamos los eventos de la interface.

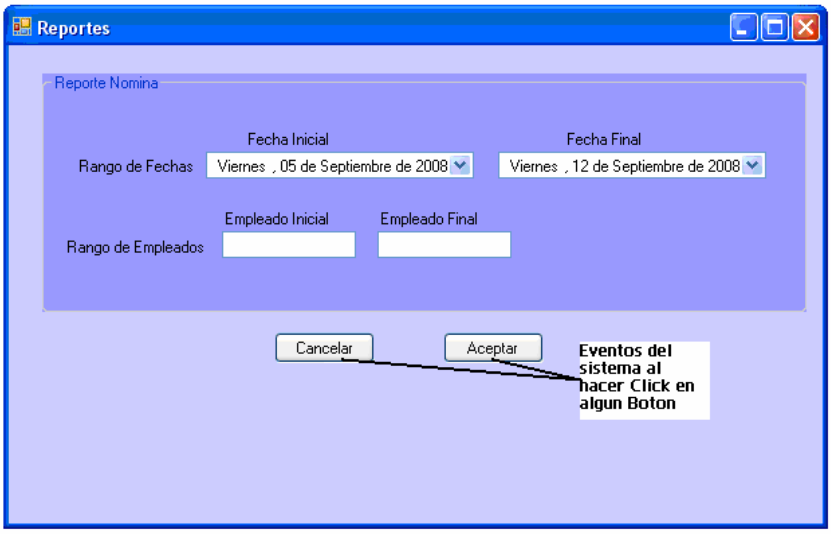


Figura 5.94 Localización de los Eventos de la GUI para el reporte para nóminas

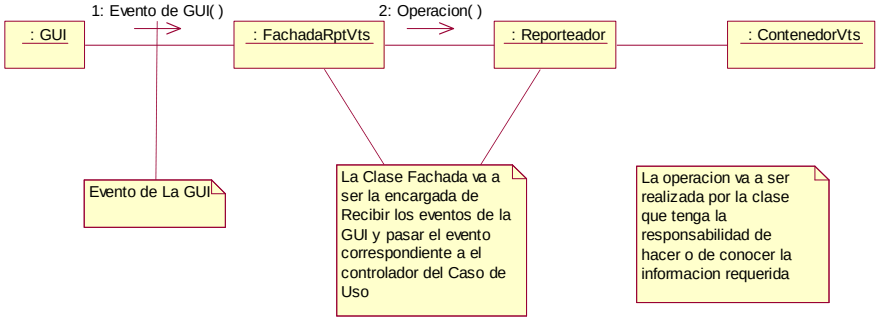


Figura 5.95 Clase fachada del sistema para el reporte

Identificadas las clases, sus colaboraciones y las tareas que estas llevan a cabo tenemos el Diagrama de Clases de Diseño.

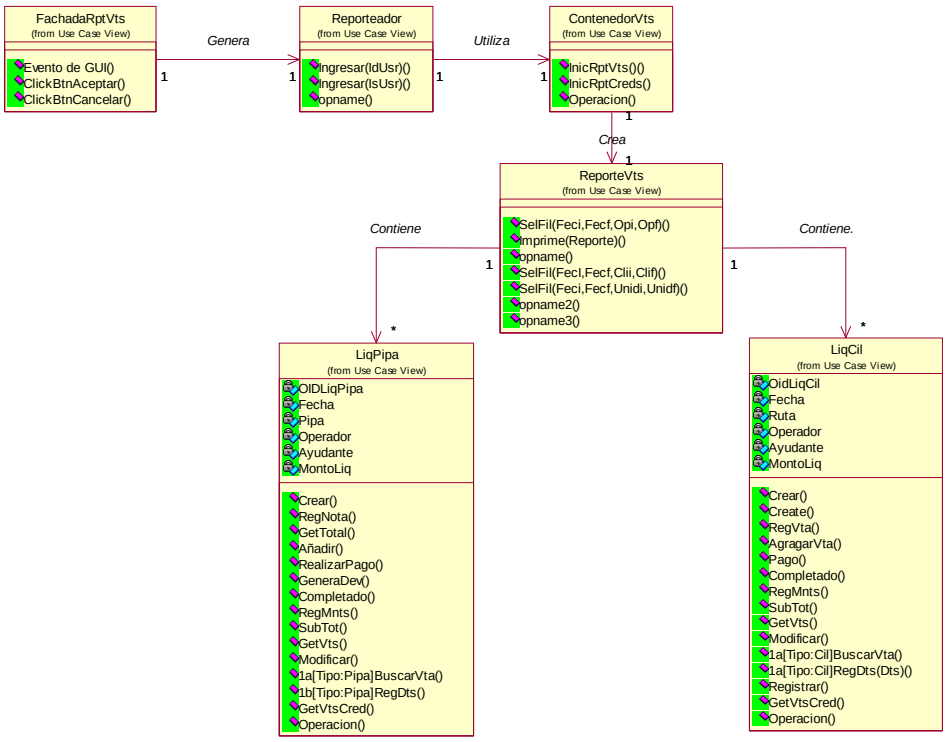


Figura 5.96 Diagrama de Clases de Diseño para el caso de uso reporte de nóminas

5.14 Caso de uso UC14: Obtención de reporte de cuentas x cobrar

Aplicando patrones GRASP para asignar responsabilidades para llevar a cabo el Reporte de Créditos tenemos que, primero aplicando el patrón experto vamos a asignar la responsabilidad de saber los datos de las ventas a crédito que requiere el encargado de Cuentas por Cobrar. En este caso seria un reporte de ventas en que contendría la información de las ventas hechas a crédito. Con lo que una clase candidata seria Reporte de Ventas a Crédito.

Después, haciendo referencia al patrón creador, vamos a identificar quien seria el encargado de crear la instancia del reporte de ventas a crédito, observamos que los datos requeridos para este reporte son ventas al igual que en el caso de uso anterior, por lo tanto puede ser que el contenedor de ventas sea el responsable de crear la instancia del reporte de ventas al ser este el que contendría la información de las ventas a ser mostradas en el reporte, por consiguiente el Contenedor de Ventas es la clase candidata directa para crear la instancia de Reporte de Ventas.

Finalmente, para el manejo y control de los eventos del caso de uso vamos a tomar la clase Reporteador, el cual seria el encargado de tomar los eventos del caso de uso, ya que los eventos son similares a los del caso de uso anterior. De lo anterior podemos ver que las clases encargadas de dichas tareas quedan así:

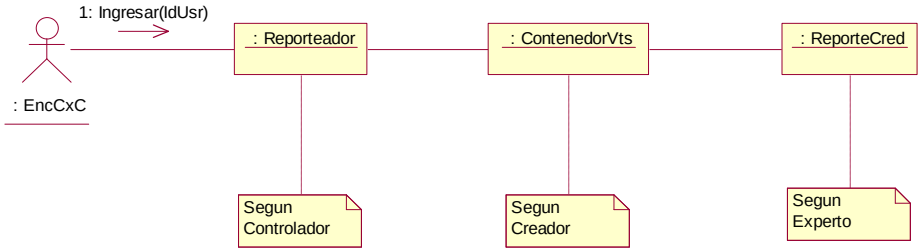


Figura 5.97. Asignación de Responsabilidades para el Reporte de C x C

Una vez que hemos identificado las clases con sus responsabilidades o tareas, podemos llevar a cabo el diagrama de secuencias para identificar como se desarrollan estas operaciones.

Diagrama de Secuencias

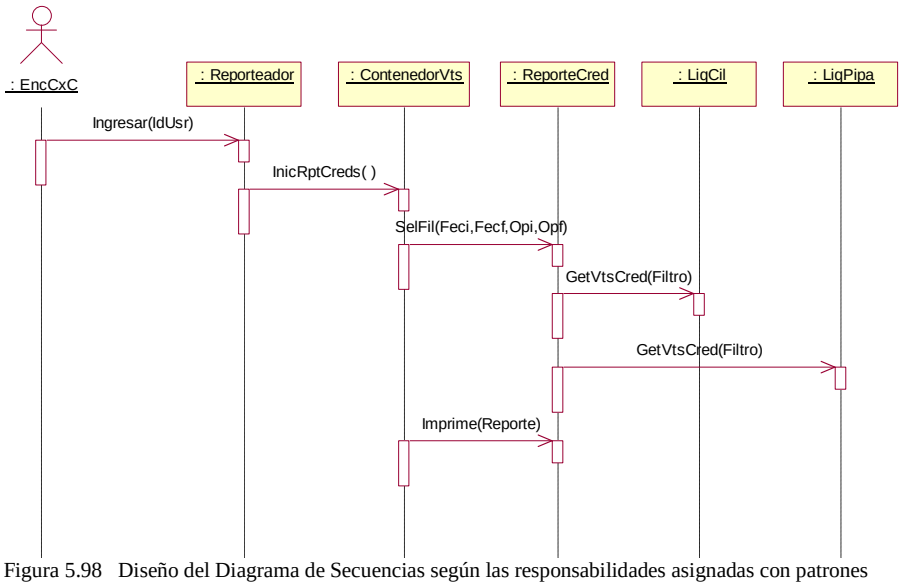


Figura 5.98 Diseño del Diagrama de Secuencias según las responsabilidades asignadas con patrones

Una vez que hemos identificado las clases y con las operaciones que obtuvimos en el capitulo anterior para este caso de uso vamos a desarrollar los contratos de operaciones y los diagramas de colaboración.

Contratos de Operaciones y Colaboraciones

Operación: InicRepCxC( )
Responsabilidades: Abrir la tabla de ventas para extraer la información.
Caso de Uso: Reporte de CxC
Controlador: Sistema
Precondiciones: El sistema conoce el Id del encargado de CxC
Postcondiciones: - Se Genero una nueva instancia de RptCxC "repCxC"

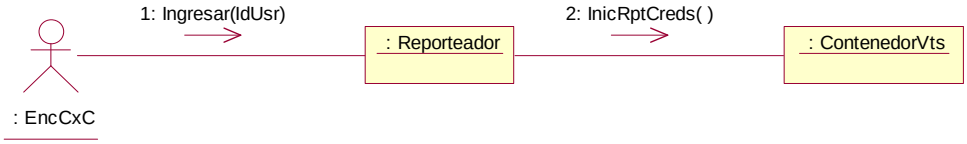


Figura 5.99 Diagrama de Colaboración para la operación InicRepCxC()

Operación: SelFiltro(fechini, fechfin)
Responsabilidades: Registar los Filtros a ser aplicados para generar el reporte.
Caso de Uso: Reporte de CxC.
Controlador: Sistema, Enc. CxC
Precondiciones: La generación del reporte esta en Proceso.
Postcondiciones: - Se registraron los filtros a ser aplicados para la generación del reporte.

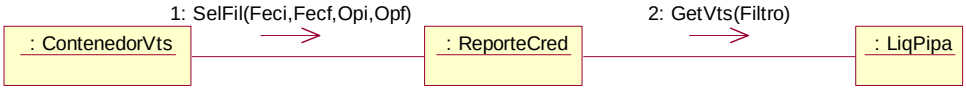


Figura 5.100 Diagrama de Colaboración para la operación SelFiltro(fechini, fechfin)

Operación: ImprimeRpt( )
Responsabilidades: Mandar a imprimir el reporte generado
Caso de Uso: Reporte de CxC
Controlador: Sistema
Precondiciones: El repote se tiene generado
Postcondiciones: - Se mando a imprimir el reporte. - Se cerro la operación repCxC( )

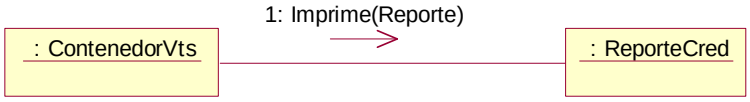


Figura 5.101 Diagrama de Colaboración para la operación ImprimeRpt()

Con el patrón fachada vamos a generar la clase que va a ser la encargada de recibir los eventos de la interface, y sus operaciones para llevar a cabo las tareas para realizar el caso de uso como se necesita. Aquí identificamos los eventos de la interface.

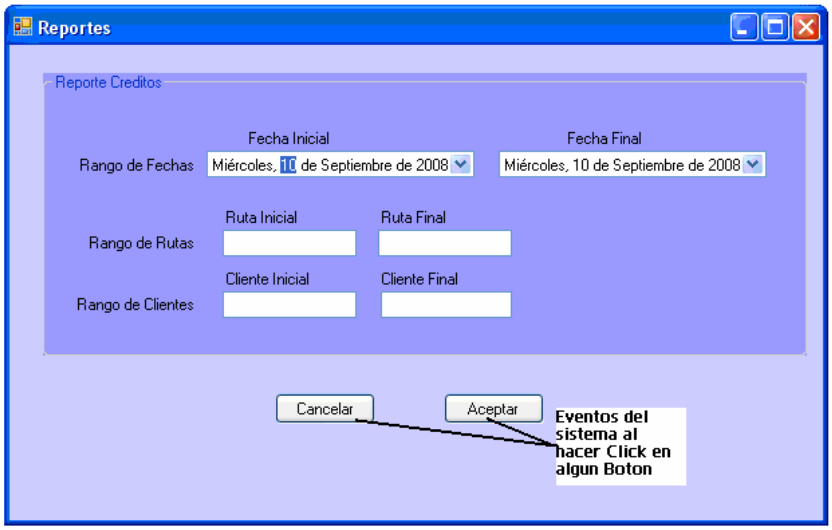


Figura 5.102 Localización de los eventos de la GUI para el reporte de CxC

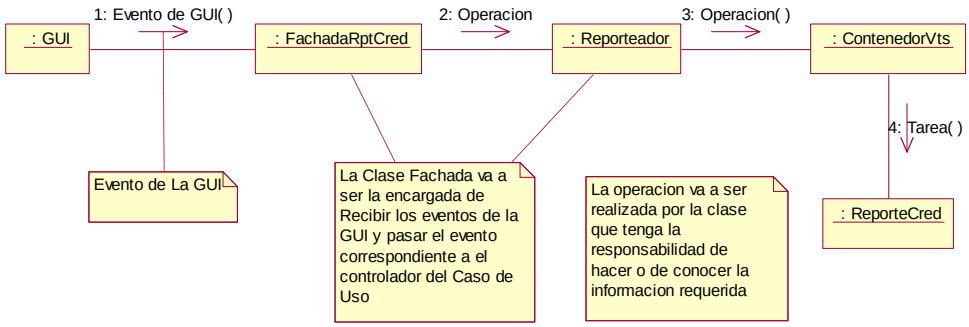


Figura 5.103 Clase fachada del sistema para el reporte de CxC

Identificadas las clases, sus colaboraciones y las tareas que estas llevan a cabo tenemos el Diagrama de Clases de Diseño.

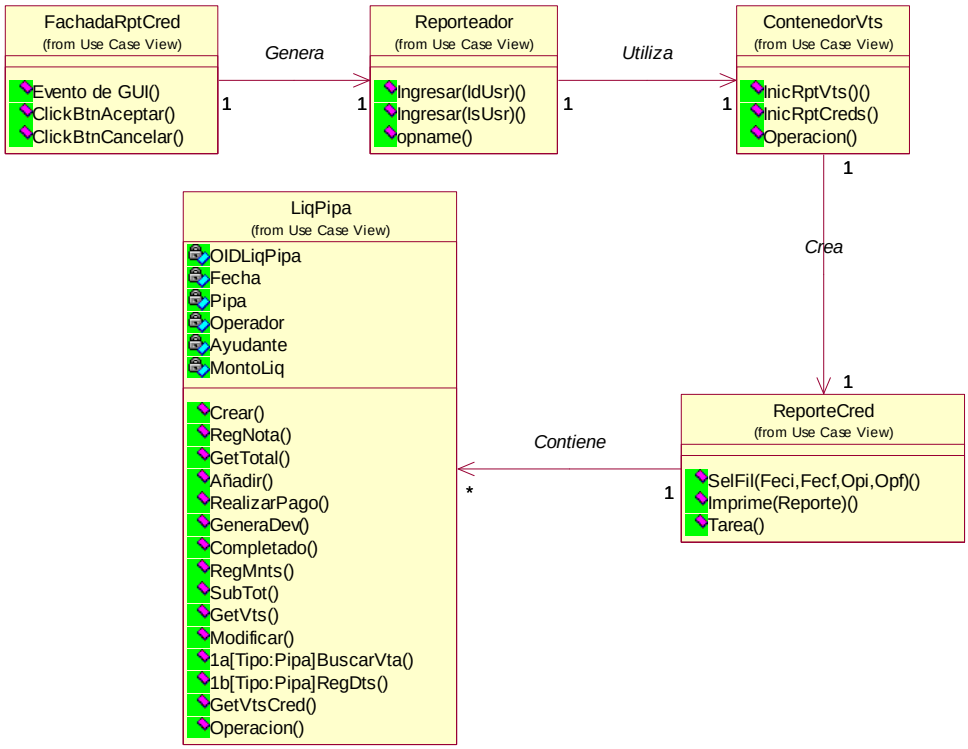


Figura 5.104 Diagrama de Clases de Diseño para el caso de uso reporte de CxC

5.15 Caso de uso UC15: Obtención de reporte de ventas

Al observar las tareas del caso de uso, observamos que son similares a las del Reporte para Nominas, por lo que las clases Experto, Creador, y Controlador van a ser las mismas, al ver que los eventos y las operaciones son también las mismas.

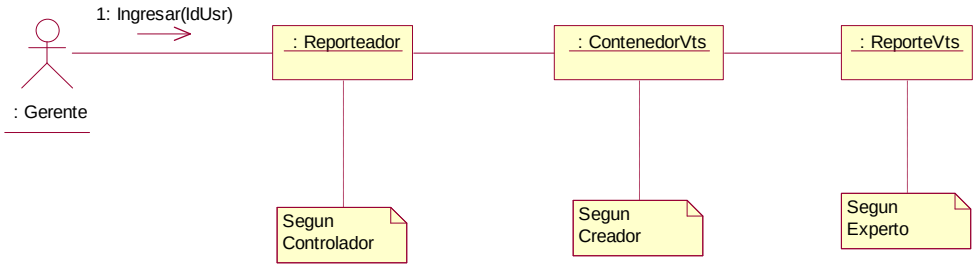


Figura 5.105 Asignación de responsabilidades para el reporte de ventas

Al igual que las clases que llevan las responsabilidades, el diagrama de secuencias también es similar. La diferencia en este caso de uso es el actor que lleva a cabo el caso de uso.

Diagrama de Secuencias

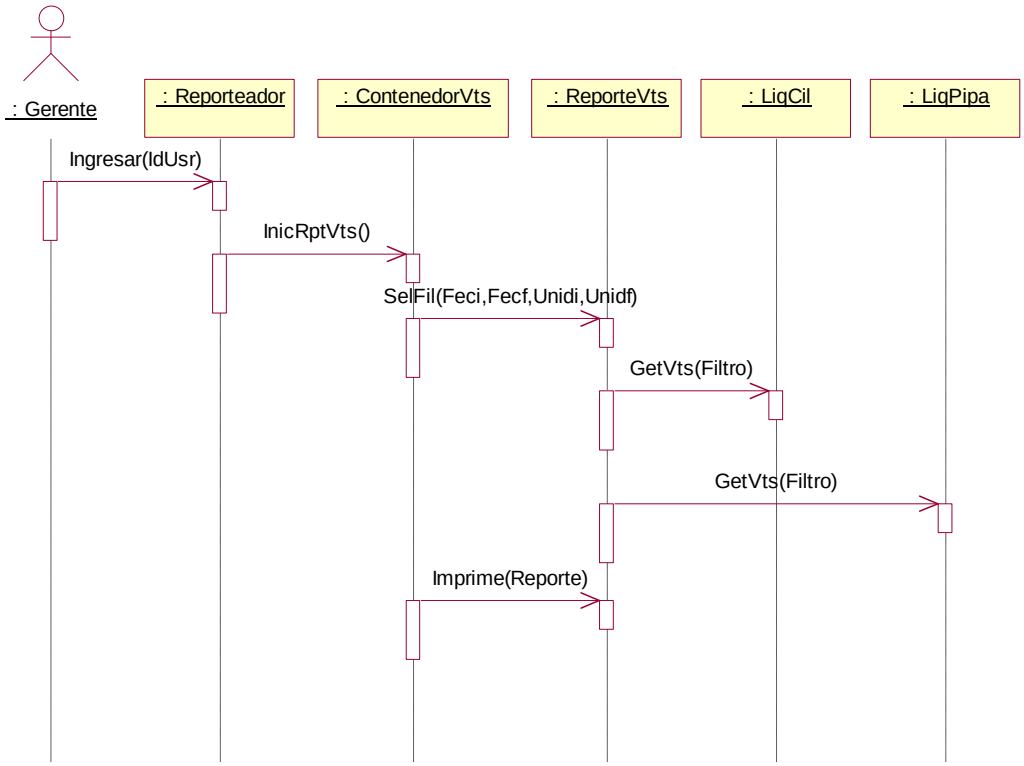


Figura 5.106 Diseño del Diagrama de Secuencias según las responsabilidades asignadas con patrones

También los Contratos y Colaboraciones son similares a los del caso de uso Reporte de Ventas para Nómina.

Contratos de Operaciones y Colaboraciones

Operación: InicRepVentas( )
Responsabilidades: Abrir la tabla de ventas para extraer la información.
Caso de Uso: Reporte de Ventas
Controlador: Sistema
Precondiciones: El sistema conoce el Id del gerente
Postcondiciones: - Se Genero una nueva instancia de RptVts "repVent"

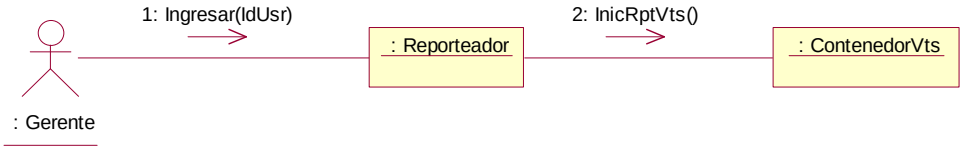


Figura 5.107 Diagrama de Colaboración para la operación InicRepVentas()

Operación: SelFiltro(fechini, fechfin, uniIni,unifin)
Responsabilidades: Registar los Filtros a ser aplicados para generar el reporte.
Caso de Uso: Reporte de Ventas
Controlador: Sistema, Gerente
Precondiciones: La generación del reporte esta en Proceso.
Postcondiciones: - Se registraron los filtros a ser aplicados para la generación del reporte.

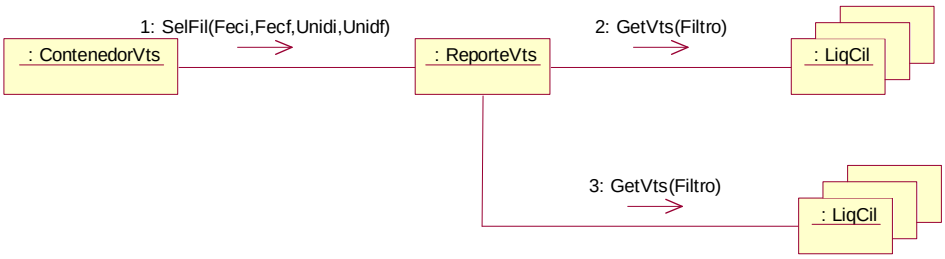


Figura 5.108 Diagrama de Colaboración para la operación SelFiltrb(fechini fechini, fechfin, uniIni,unifin)

Operación: ImprimeRpt( )
Responsabilidades: Mandar a imprimir el reporte generado
Caso de Uso: Reporte de Ventas
Controlador: Sistema
Precondiciones: El repote se tiene generado
Postcondiciones: - Se mando a imprimir el reporte. - Se cerro la operación repVent( )

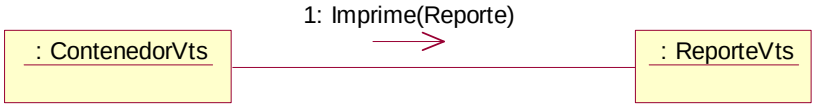


Figura 5.109 Diagrama de Colaboración para la operación ImprimeRpt()

El patrón fachada nos muestra que los eventos son los mismos que en el caso de uso del reporte para nómina, pero los parámetros del filtro varían, aunque la fachada puede mantenerse.

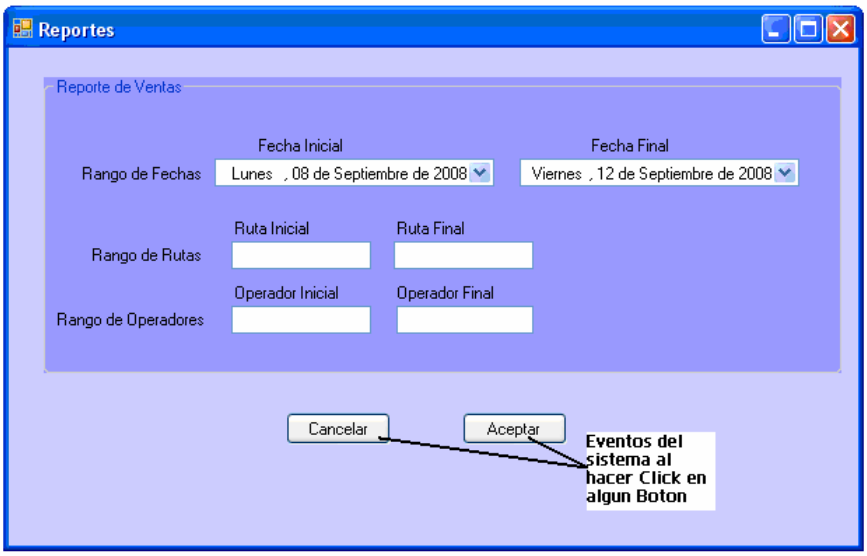


Figura 5.110 Localización de los eventos de la GUI para el reporte de ventas

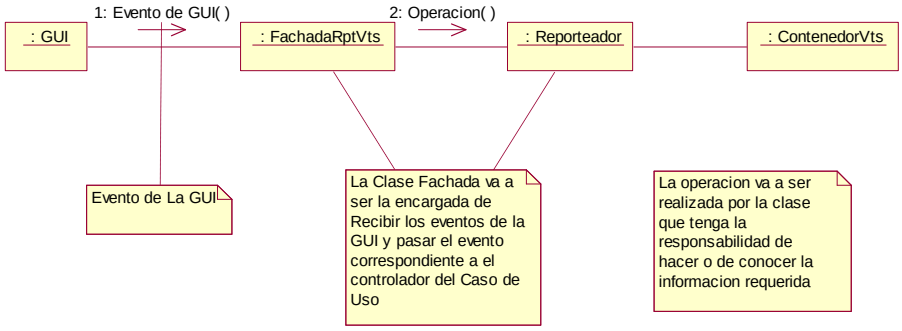


Figura 5.111. Clase Fachada del Sistema Para el Reporte de Ventas

Y el Diagrama de Clases de Diseño también se mantiene.

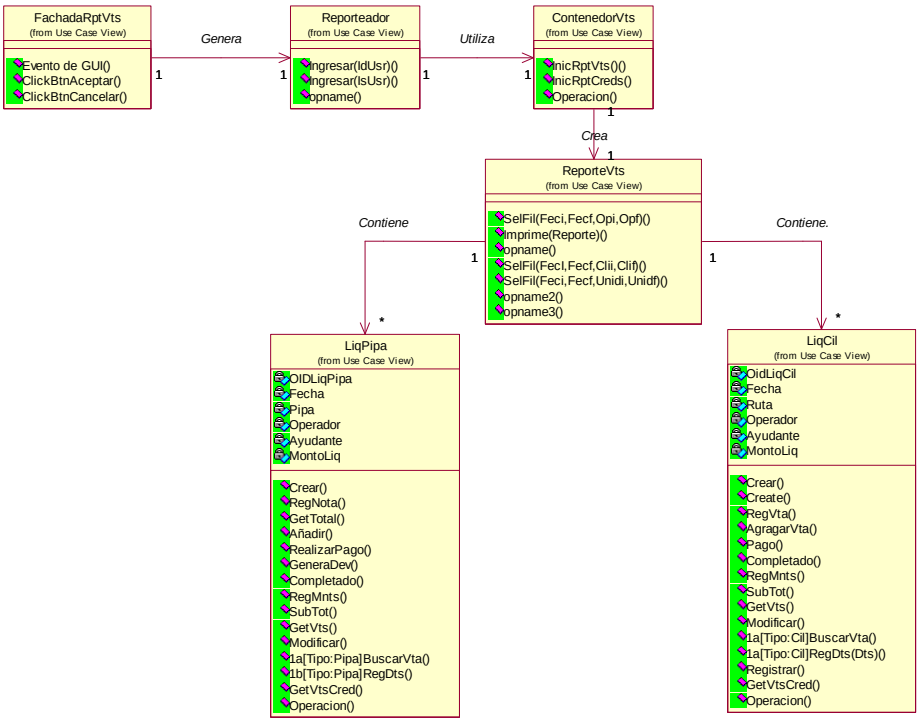


Figura 5.112 Diagrama de Clases de Diseño para el caso de uso reporte de ventas

5.16 Caso de uso UC16: Obtención de reporte de cortes de caja

Aplicando el patrón experto vamos a asignar la responsabilidad de saber los datos de los cortes de caja que requiere el gerente. En este caso seria un reporte de los cortes de caja el que contendría la información de estos cortes. Con lo que una clase candidata seria Reporte de Cortes.

Después, haciendo referencia al patrón creador, vamos a identificar quien seria el encargado de crear la instancia del reporte de ventas, haciendo referencia nuevamente a la definición del patrón creador, tenemos que "B contiene objetos de A", en este caso podemos ver que los cortes deben estar contenidas en algún lado y de ahí mostrarlas en el reporte de ventas, por tanto, aplicando el patrón creador, un contenedor de los cortes seria el responsable de crear la instancia del reporte de cortes al ser este el que contendría la información de los cortes a ser mostradas en el reporte, por consiguiente Contenedor de Cortes es una clase candidata a creador de la instancia de Reporte de Cortes.

Finalmente, para el manejo y control de los eventos del caso de uso vamos a volver tomar la clase "Reporteador", ya que los eventos del caso de uso son similares en le generación del reporte, lo que cambia son los parámetros que van a ser aplicados. De lo anterior podemos ver que las clases encargadas de dichas tareas quedan así:

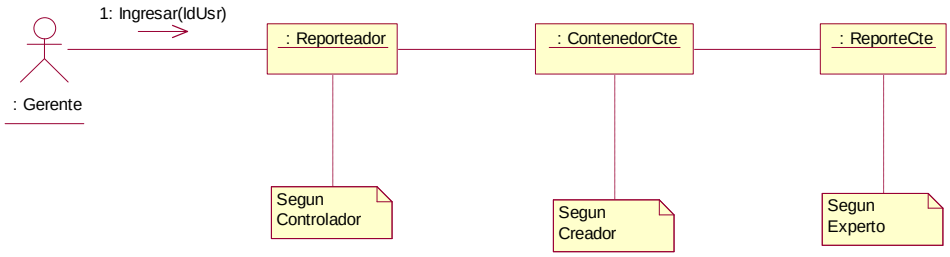


Figura 5.113 Asignación de Responsabilidades para el reporte de cortes de caja

Una vez que hemos identificado las clases con sus responsabilidades o tareas, podemos llevar a cabo el diagrama de secuencias para identificar como se desarrollan dichas operaciones.

Diagrama de Secuencias

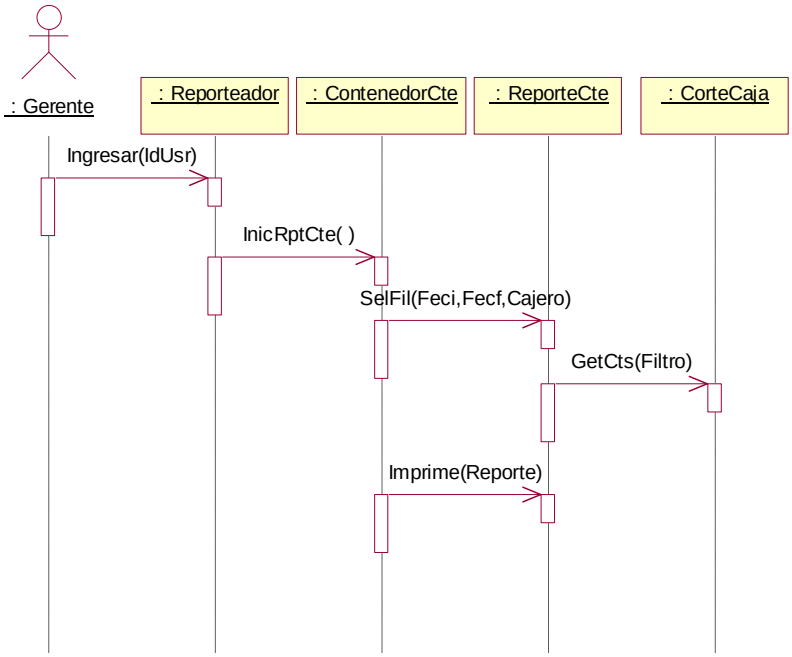


Figura 5.114 Diseño del Diagrama de Secuencias según las responsabilidades asignadas con patrones

Con las clases que hemos identificado y con las operaciones que obtuvimos en el capitulo anterior para este caso de uso vamos a desarrollar los contratos de operaciones y los diagramas de colaboración.

Contratos de Operaciones y Colaboraciones

Operación: InicRepCts( )
Responsabilidades: Abrir la tabla de Cortes para extraer la información.
Caso de Uso: Reporte de Cortes de Caja
Controlador: Sistema
Precondiciones: El sistema conoce el Id del gerente
Postcondiciones: - Se Genero una nueva instancia de RptCotVent "repCV"

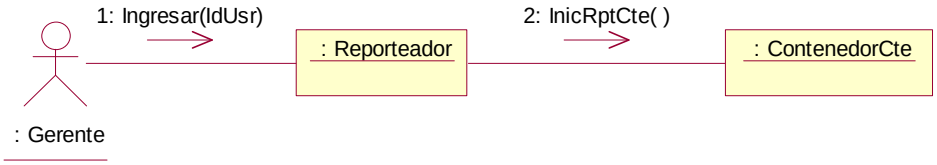


Figura 5.115 Diagrama de Colaboración para la operación InicRepNom()

Operación: SelFiltro(fechini, fechfin, Cajini, cajfin)
Responsabilidades: Registar los Filtros a ser aplicados para generar el reporte.
Caso de Uso: Reporte de Cortes de Caja.
Controlador: Sistema, gerente
Precondiciones: La generación del reporte esta en Proceso.
Postcondiciones: - Se registraron los filtros a ser aplicados para la generación del reporte.

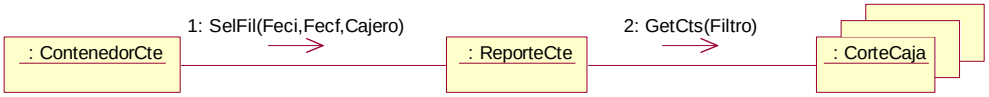


Figura 5.116 Diagrama de Colaboración para la operación SelFiltrb(fechini, fechfin, Cajini, cajfin)

Operación: ImprimeRpt( )
Responsabilidades: Mandar a imprimir el reporte generado
Caso de Uso: Reporte de Cortes de Caja
Controlador: Sistema
Precondiciones: El reporte se ha generado
Postcondiciones: - Se mando a imprimir el reporte. - Se cerro la operación repCV ( )

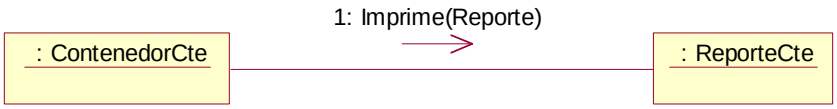


Figura 5.117 Diagrama de Colaboración para la operación ImprimeRpt()

Con el patrón fachada vamos a generar la clase que va a ser la encargada de recibir los eventos de la interface, y sus operaciones para llevar a cabo las tareas para realizar el caso de uso como se necesita. Aquí identificamos los eventos de la interface.

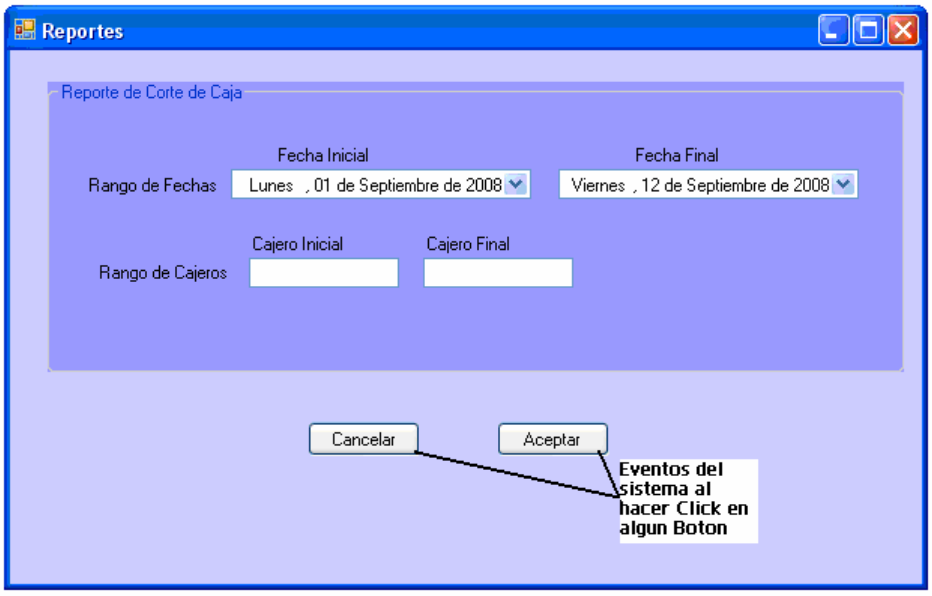


Figura 5.118 Localización de los eventos de la GUI para el reporte de cortes

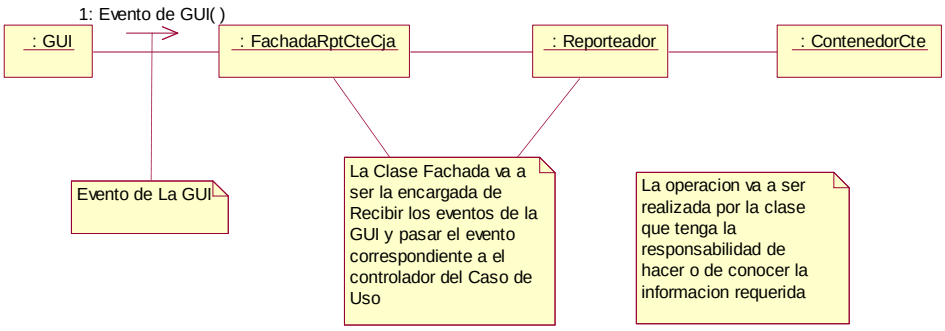


Figura 5.119 Clase fachada del sistema para el reporte

Identificadas las clases, sus colaboraciones y las tareas que estas llevan a cabo tenemos el Diagrama de Clases de Diseño.

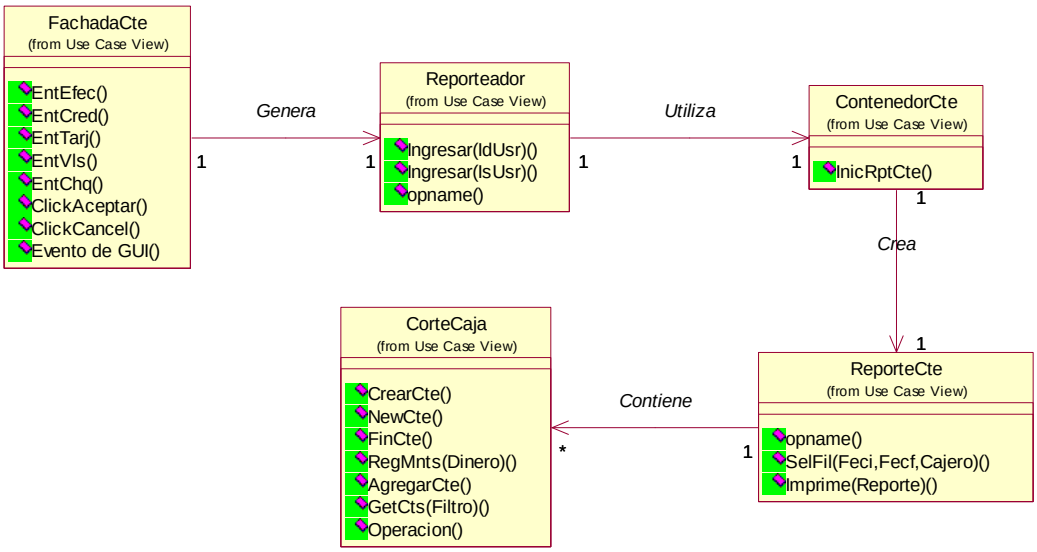


Figura 5.120 Diagrama de Clases de Diseño para el caso de uso reporte de cortes de caja

5.17 Caso de uso UC17: Modificación de precios de kilo / litro de gas

Aplicando nuevamente patrones GRASP vamos a asignar responsabilidades para llevar a cabo la modificación de los precios del litro y del kilo de gas.
Primero aplicando el patrón experto vamos a asignar la responsabilidad de saber los precios actuales del gas, en este caso el litro y el kilo actualmente tienen un precio, los cuales se encuentran registrados cada uno en sus propios registros, por lo que el mismo gas sabe su precio actual, por lo tanto gas es la clase candidata.

Luego, aplicando el patrón Creador, vamos a utilizar una clase registro, ya que la operación va a ser la de registrar un nuevo precio, por lo que la responsabilidad de crear la instancia de la clase gas para modificar sus precios va a ser una clase Registro de Gas.

Finalmente, para el manejo y control de los eventos del caso de uso vamos a utilizar una clase Catalogo de Productos, o Catalogo de Gas ya que nuevamente, haciendo una analogía con el mundo real, seria un catalogo de productos donde se tendrían los precios y ahí seria donde se modificarían, es decir en el catalogo de productos se lleva el control de los eventos de los cambios que puedan sufrir estos productos.
De lo anterior podemos ver que las clases encargadas de dichas tareas quedan así:

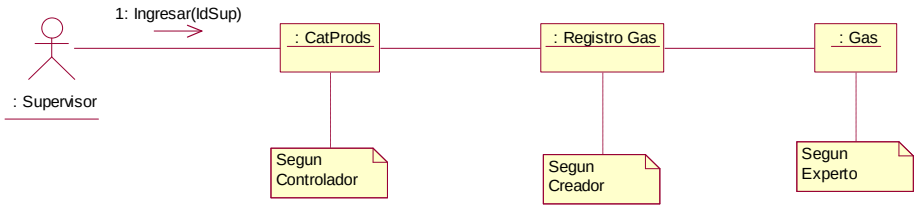


Figura 5.121 Asignación de responsabilidades para modificar los precios

Ahora que tenemos identificadas las clases con sus responsabilidades o tareas, vamos a llevar a cabo el diagrama de secuencias para observar como se llevan a cabo las operaciones.

Diagrama de Secuencias

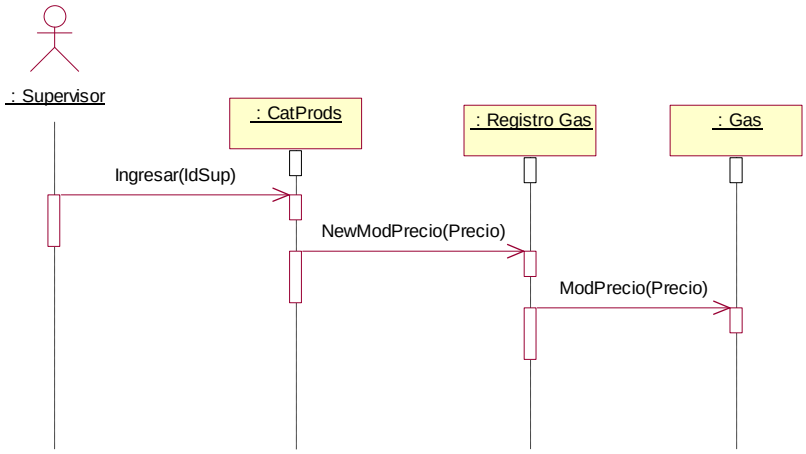


Figura 5.122 Diseño del Diagrama de Secuencias según las responsabilidades asignadas con patrones

Continuamos con los contratos y los diagramas de Colaboración.

Contratos de Operaciones y Colaboraciones

Operación: InicMod( )
Responsabilidades: Preparar el catalogo de productos para modificar los precios, Generar una nueva instancia modPre
Caso de Uso: Modificar Precios
Controlador: Supervisor
Precondiciónes: El sistema conoce el Id del Supervisor
Postcondiciones: - Se genero una nueva instancia modPre( ) de ModifPrec( )

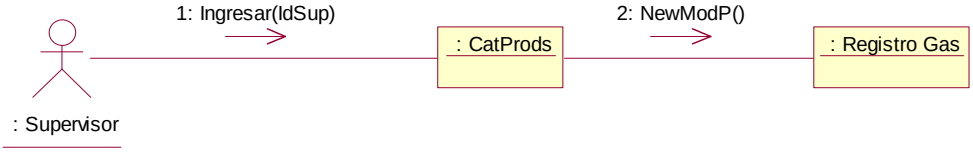


Figura 5.123 Diagrama de Colaboración para la operación InicMod()

Operación: RegPrecio(Precio)
Responsabilidades: Editar los Precios de los registros.
Caso de Uso: Modificar Precios
Controlador: Sistema, Supervisor.
Precondiciones: Esta en Transacción la modificación de los precios
Postcondiciones: - Se registraron los nuevos precios del kilo y del litro de gas.

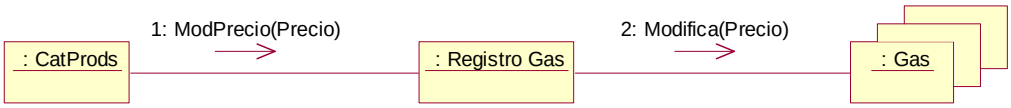


Figura 5.124 Diagrama de Colaboración para la operación RegPrecio(Precio)

Operación: FinRegPre( )
Responsabilidades: Cerrar el catalogo de Productos, Finalizar la Operación.
Caso de Uso: Madificar Precios.
Controlador: Sistema
Precondiciones: Se validaron los nuevos precios del Gas
Postcondiciones: - Se cerró la operación modPre( )

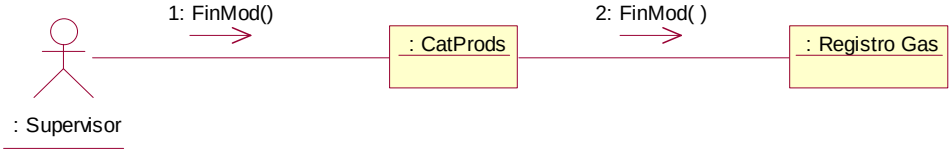


Figura 5.125 Diagrama de Colaboración para la operación RegPre(Pre)

Nuevamente identificamos los eventos que se desprenden de la interface y una clase que sirva para interactuar con las clases que llevan a cabo las operaciones.

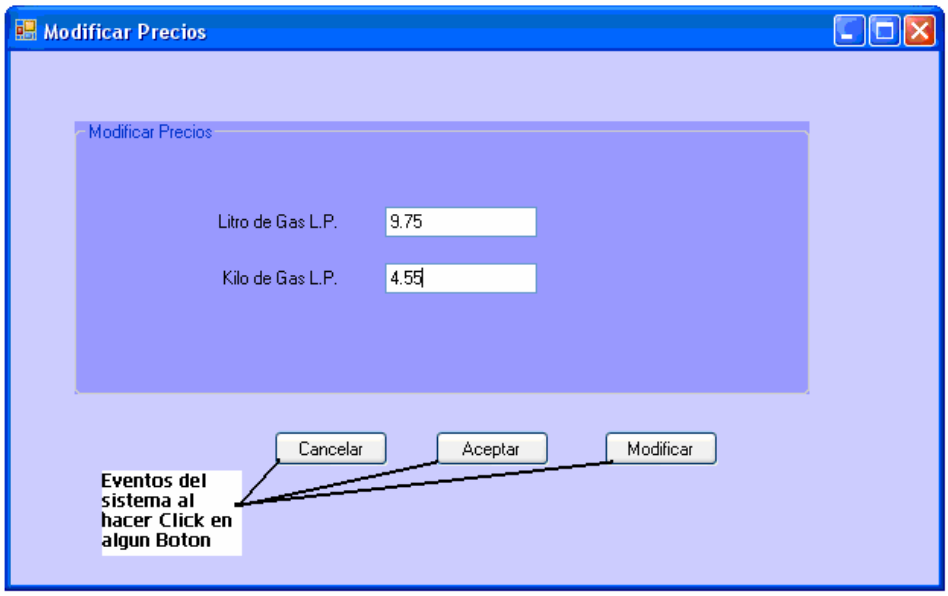


Figura 5.126 Localización de los eventos de la GUI para la modificación de precios

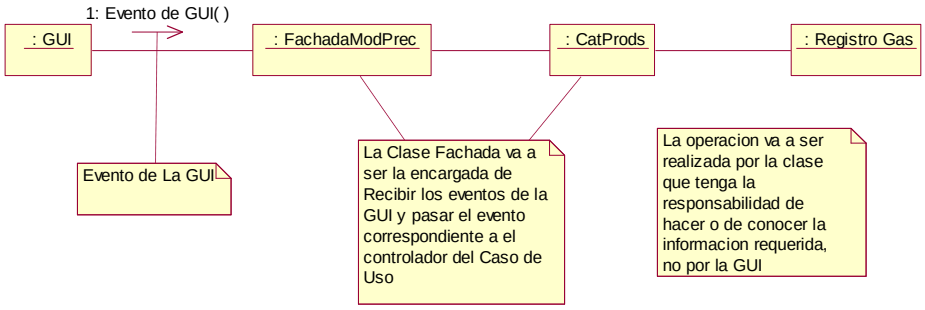


Figura 5.127 Clase fachada del sistema para la GUI de la modificación de precios

Procedemos a Desarrollar el Diagrama de Clases de Diseño.

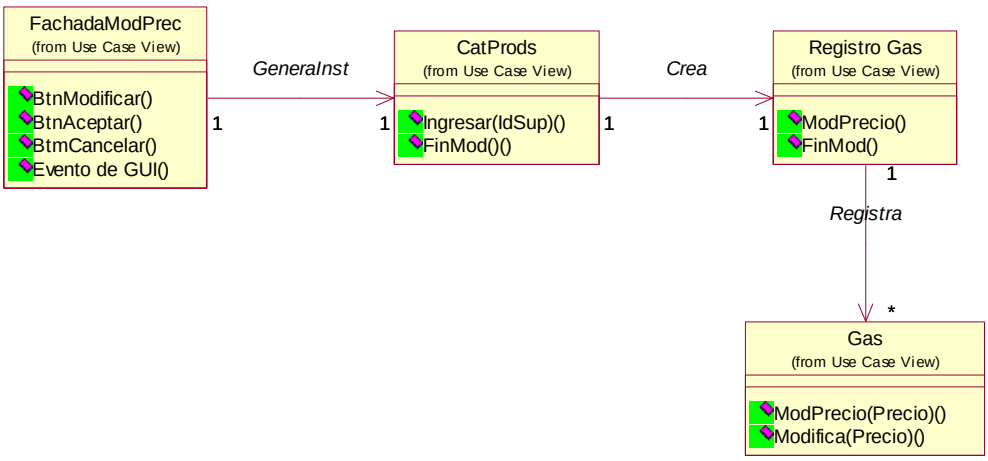


Figura 5.128 Diagrama de Clases de Diseño para el caso de uso de modificación de precios

5.18 Caso de uso UC18: Modificación de ventas registradas

Aplicando patrones GRASP para asignar responsabilidades para llevar a cabo la modificación de alguna venta registrada, Primero aplicando patrón experto vamos a asignar la responsabilidad de saber los datos de una liquidación. En este caso existen dos tipos de liquidación, tanto de cilindros como de pipas, por lo que vamos a asignar la responsabilidad a una clase genérica de liquidación que también va a fungir como controladora de los eventos de la creación de la instancia de liquidación a ser modificadas, ya sea de cilindros o de pipas.

Entonces, aplicando el patrón Creador, vamos a utilizar una clase registro, ya que la operación va a ser la de modificar una liquidación que ya fue registrada, por lo que la responsabilidad de crear la instancia de la liquidación a modificar va a ser la clase "Registro de Ventas".

Finalmente, para el manejo y control de los eventos del caso de uso vamos a utilizar una clase Caja, ya que ha sido la clase controlador para el registro de las liquidaciones.

De lo anterior podemos ver que las clases encargadas de dichas tareas quedan así:

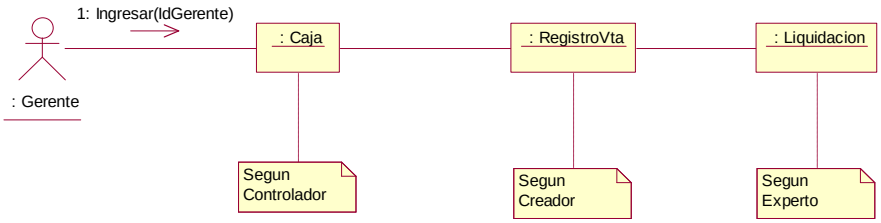


Figura 5.128 Asignación de responsabilidades para modificar ventas

Una vez que tenemos identificadas las clases con sus responsabilidades o tareas, podemos llevar a cabo el diagrama de secuencias para identificar como se llevaran a cabo dichas operaciones.

Diagrama de Secuencias

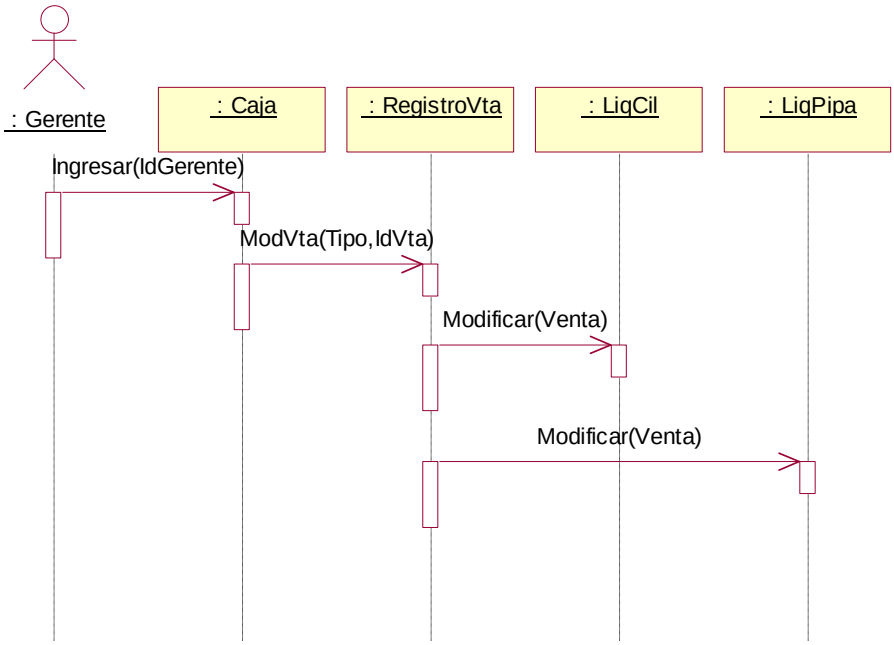


Figura 5.129 Diseño del Diagrama de Secuencias según las responsabilidades asignadas con patrones

Con las clases que hemos identificado y con las operaciones que obtuvimos en el capitulo anterior para este caso de uso vamos a desarrollar los contratos de operaciones y los diagramas de colaboración.

Contratos de Operaciones y Colaboraciones

Operación: InicMod( )
Responsabilidades: Preparar el catalogo para la Edición del registro, Se genera una nueva instancia ModVts "MVen".
Caso de Uso: Modificar Ventas
Controlador: Sistema
Precondiciones: El sistema conoce el Id del Gerente
Postcondiciones: - Se genero una nueva instancia ModVts "" - Se abrió la Base de Datos para editar el registro.

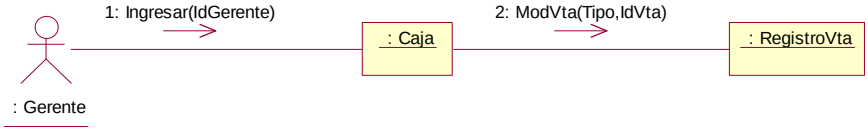


Figura 5.130 Diagrama de Colaboración para la operación InicMod()

Operación: SelVta(IdUni)
Responsabilidades: Buscar el registro de la unidad que contiene la venta a ser modificada.
Caso de Uso: Modificar Ventas
Controlador: Sistema
Precondiciones: El sistema conoce el Id de la Unidad
Postcondiciones: - Se asocio el Id de la unidad con la instancia MVen.

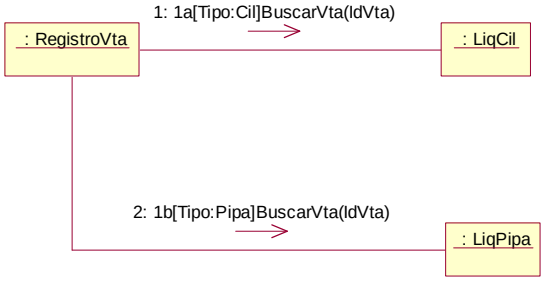


Figura 5.131 Diagrama de Colaboración para la operación SelVta(IdUni)

Operación: RegDtsVent(Dts)
Responsabilidades: Editar el registro de la Venta
Caso de Uso: Modificar Ventas
Controlador: Sistema, Gerente
Precondiciones: Esta en Transacción la edicion del registro de la venta
Postcondiciones: - Se registraron los nuevos datos de la venta en el catalogo

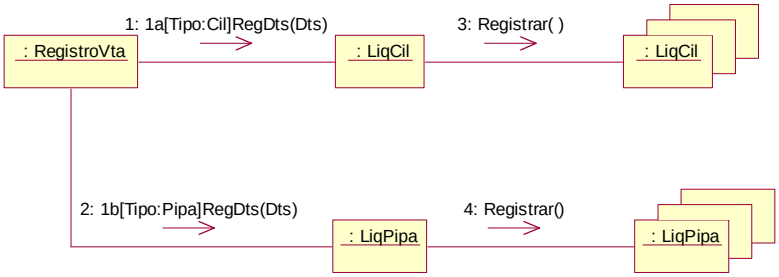


Figura 5.132 Diagrama de Colaboración para RegDtsVent(Dts)

Operación: FinMod( )
Responsabilidades: Cerrar la tabla de ventas, Finalizar la Operación.
Caso de Uso: Modificar Ventas.
Controlador: Sistema
Precondiciones: Se validaron los nuevos datos de la venta
Postcondiciones: - Se cerro la operación MVen ( )

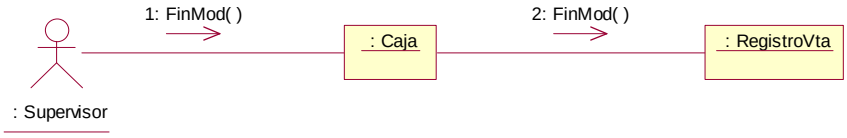


Figura 5.133 Diagrama de Colaboración para FinMod()

Con el patrón fachada vamos a generar la clase que va a ser la encargada de recibir los eventos de la interface, y las operaciones que se identifiquen para llevar a cabo las tareas para realizar el caso de uso como se requiere.

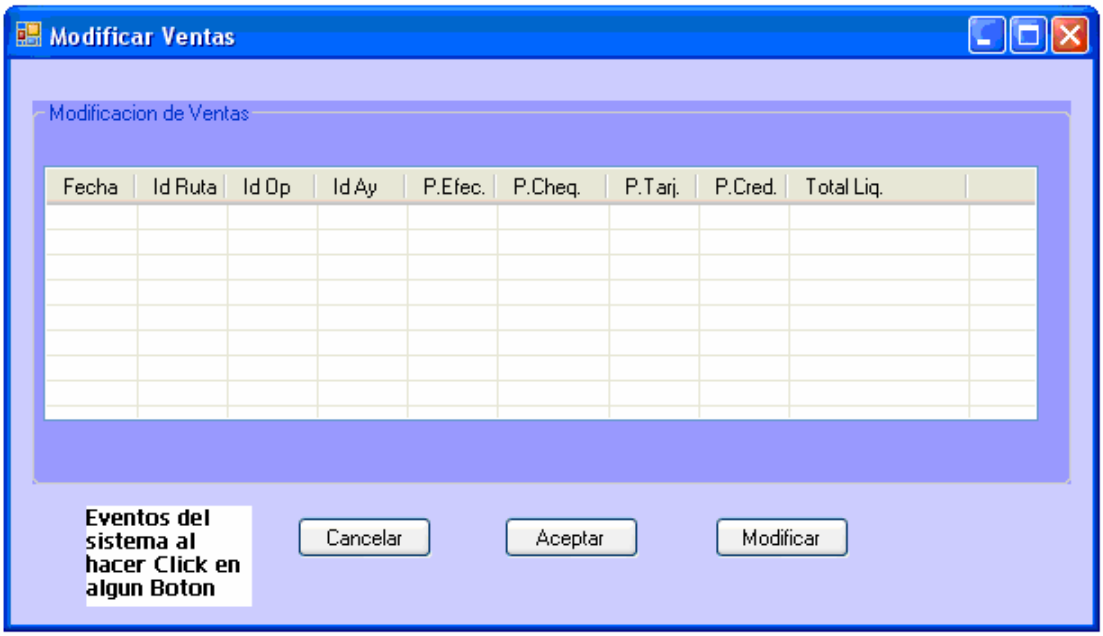


Figura 5.134 Localización de los eventos de la GUI para modificar ventas

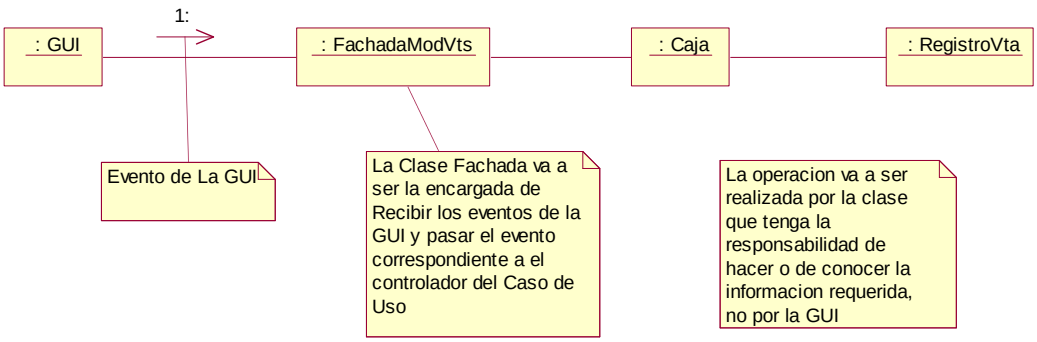


Figura 5.135 Clase fachada del sistema para modificación de ventas

Identificadas las clases, sus colaboraciones y las tareas que estas llevan a cabo tenemos el Diagrama de Clases de Diseño.

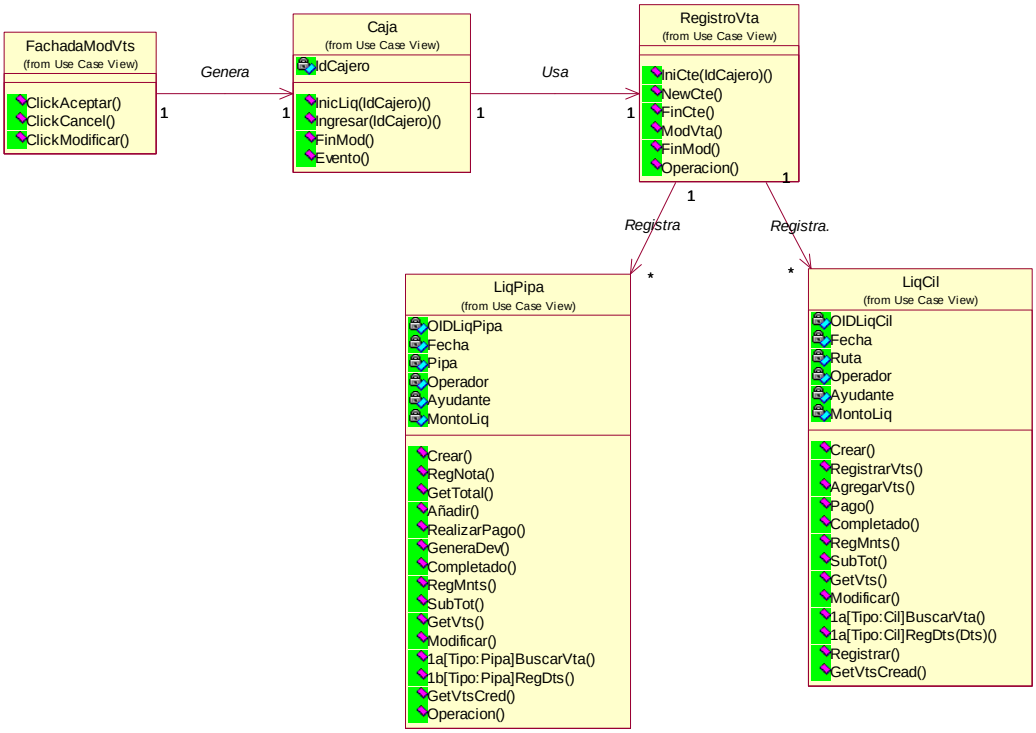


Figura 5.136 Diagrama de Clases de Diseño para el caso de uso modificar ventas

5.19 Otros patrones de diseño

Aplicando mas patrones para la relación de los Objetos con una Base de Datos Relacional

5.19.1 Patrón: Representación de Objetos como Tabla

Este patrón propone la definición de una tabla en una BDR por cada clase de objetos persistente.
Si un objeto solo tiene atributos de tipos de datos primitivos, la correspondencia es directa.
Así por ejemplo tendríamos para un objeto Fabricante, la correspondencia con su tabla.

Fabricante
Nombre Ciudad …
…

:Fabricante
Nombre=Fruta Fresca
Ciudad= El Valle

Nombre	Ciudad
Fruta Fresca	El Valle
Expresión Celestial	San Jacobo

5.19.2 Patrón: Identificador de Objeto

Es conveniente con una forma consistente de relacionar los objetos con los registros, y ser capaces de asegurar que la materialización repetida de un registro no da como resultados objetos duplicados.

El patrón Identificador de Objeto propone asignar in Identificador de Objeto (OID) a cada registro y objeto.
Un OID normalmente es un valor alfanumérico, cada uno es único para cada objeto específico. Existen varios enfoques para generar identificadores únicos para los OIDs, variando desde únicos para una base de datos, a únicos globalmente: generadores de secuencia de Base de Datos, la estrategia de generación de claves.

En el campo de los Objetos, un OID se representa mediante una interfaz o clase OID que encapsula el valor real y su representación. En un BDR, normalmente se almacena como un valor de tipo carácter de longitud fija.

Cada tabla tendrá un OID como Clave primaria, y cada objeto también tendrá (Directa o indirectamente) un OID. Si se asocia cada objeto con un OID, y cada Tabla tiene un OID como clave primaria, cada objeto se corresponde de manera única con una fila de alguna tabla.

Fabricante
Nombre Ciudad Oid:OID …
…

:Fabricante
Nombre=Fruta Fresca
Ciudad= El Valle
Oid= xyz123

OID	Nombre	Ciudad
xyz123	Fruta Fresca	El Valle
Abc345	Expresión Celestial	San Jacobo

De acuerdo con el patrón: Representación de Objetos como Tablas identificamos las Clases ue se pueden corresponder directamente con una tabla de la BDR, a las cuales también les aplicamos el patrón Identificador de Objeto, con lo cual las clases y las tablas quedan así:

Tabla Cliente

IdUnidad	Nombre	Contacto	RFC	Calle	Colonia	Ciudad	Estado	CP	Tel	TipoCLi

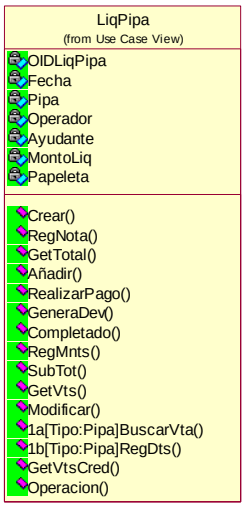


Tabla Liquidación de Pipa

IdLiq	Fecha	IdPipa	IdOp	IdAyud	MontoLiq	Papeleta

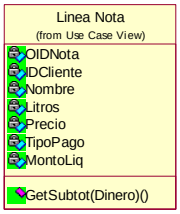


Tabla Nota

IdNota	IdCli	Nombre	Litros	Precio	TipoPag	MontoNota	IdLiqP

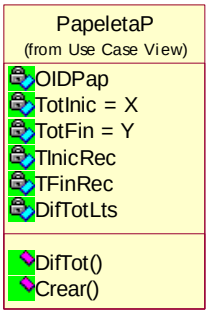
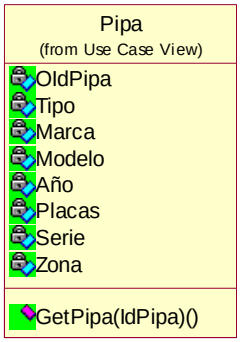


Tabla PapeletaPipa

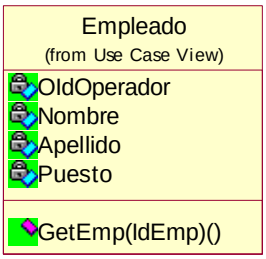
IdPap	TotIn	TotFin	RecTotIn	RecTotFin	DifLts	IdLiqP

Tabla Pipa



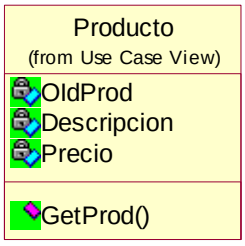
IdPip	Tipo	Marca	Modelo	Año	Placas	Serie	Zona

Tabla Empleado



IdEmp	Nombre	Apellido	Puesto

Tabla Producto



IdProd	Descripción	Precio

IdLiqRuta	Fecha	IdRuta	IdOp	IdAy	Monto	IdPap

Así de esta manera realizamos la uní valencia de cada tabla de la base de datos de relacional con cada clase que utilizamos para llevar a cabo la solución de las operaciones de los casos de uso planteados, y cada registro de dicha tabla se corresponde unívocamente con un objeto de la clase a la que representa dicha tabla.

Capítulo 6

Conclusiones y trabajos futuros

En el presente trabajo se ha establecido el análisis y diseño de una solución a un problema planteado utilizando los patrones de diseño GRASP (General Responsability Assignment Software Patterns, Patrones Generales de Software para Asignar Responsabilidades) los cuales como su nombre lo dice, resultan una herramienta muy útil para establecer las responsabilidades en el diseño orientado a objetos. Pero yendo un poco más allá, nos pueden permitir llegar a establecer una estructura bien definida en el diseño de software. Al identificar en cada problema la estructura que se ha utilizado de acuerdo al contexto que envuelve a esta solución.

El tener una estructura de software definida para un tipo de Problema especifico es útil, ya que aunque no hay dos aplicaciones iguales, existen paralelismos entre aplicaciones construidas para resolver problemas similares, entonces al tratar de establecer relación entre una solución a un problema dado y la que se esta desarrollando nos puede servir para entender mejor las necesidades del cliente y tratar de aplicar una posible solución que fue estructurada en problemas anteriores.

Loa patrones nos ayudan a encontrar y entender mejor dicha estructura y a ubicar esta estructura en una aplicación posterior, permitiéndonos concentrarnos en parte del problema, proporcionándonos una visión mas estrecha del problema, pues a veces el cliente no expone adecuadamente los requisitos por que no los conoce. Pero por medio de la experiencia que nos brinda el uso y reuso a través de patrones, nosotros podemos ayudarle a descubrir esos requisitos.

Con la ayuda de los patrones podemos desarrollar los diagramas uml con los que vamos modelando la solución, pues los patrones nos proponen esquemas que ya han sido usados de manera satisfactoria y que por supuesto nos van a brindar una respuesta favorable el momento de aplicarlos, pues como lo vimos durante el desarrollo de los diagramas de interacción del diseño propuesto en los capítulos anteriores, por medio de los patrones se pudo especificar una serie de elementos y sus relaciones entre estos de manera que nos permiten describir mejores soluciones a los diferentes problemas que van apareciendo en un contexto especifico.

Precisamente, esa es la esencia de los patrones de diseño, adquirir buenas practicas que nos permitan mejorar la calidad del diseño de un sistema, determinando elementos que soporten roles útiles en un contexto dado, haciendo mas legible la solución y haciéndola mas flexible a cambios. Esta flexibilidad es la que nos va a permitir la reusabilidad de una solución estructurada con anterioridad a un problema que se presenta en un contexto similar al que tratamos de reusar.

Así pues, se dice que la meta de una función es la que define la forma, es decir que la estructura de una solución esta en función con lo que tiene que hacer. Al diseñar una solución con patrones, podemos definir una estructura mas clara para alcanzar la meta de cada función, una vez diseñada y ubicada esta estructura, va a ser mas sencillo tratar de relacionarla con sistemas que tengan objetivos similares, si los procesos que llevamos a cabo están bien definidos vamos a poder ubicarlos en un contexto similar de un problema diferente y vamos a poder aplicarlos con la experiencia que se ha ganado a través de problemas anteriores en los que se aplicó esta estructura.

El desarrollo de una solución con patrones constituye un marco conceptual en el diseño de la arquitectura de nuestros sistemas, ya que como la función define la forma, sintetizan por lo general soluciones arquitectónicas y estructurales bien probadas y muy útiles dentro del tipo de problemas que modelan.

Los patrones nos van a permitir identificar y completar los casos de uso básicos expuestos por el cliente, comprender la arquitectura del sistema a construir así como su problemática, y buscar componentes que ya tengamos desarrollados que cumplan con los requisitos del tipo de sistema a construir, o sea que nos permiten obtener de una manera mas sencilla la arquitectura base que buscamos durante la fase de diseño y modelado de la solución en un contexto dado.

En la programación orientada a objetos la reutilización de software es una de las principales metas que se busca conseguir, pero en un campo tan cambiante como el del software, todo necesita evolucionar y nada es definitivo. La necesidad de innovar y reflexionar seguirá siendo un elemento muy importante en el desarrollo de las aplicaciones.

Por otro lado, la estructura aquí dada queda abierta para que en un futuro se le puedan agregar los módulos que sean necesarios. Por ejemplo el cálculo de las cuentas por cobrar a los clientes con saldos atrasados, para cancelarles el crédito hasta que no liquiden sus cuentas pendientes. Esto sin hacer modificaciones al modulo actual o implicando cambios mínimos en este, como por ejemplo un cambio necesario seria agregar un campo en el catalogo de clientes tiempodeCredito, para indicar el tiempo que va a permitírsele a un cliente tener saldos sin pagar y de ahí realizar las operaciones necesarias para calcular si algún crédito ha llegado al limite de ese tiempo para ejercerle la cobranza o incluso si lo ha rebasado para cancelarle ventas a crédito hasta que salde sus deudas. Este cambio no implica una modificación mayor que altere el funcionamiento actual.

En general al realizar una asignación de responsabilidades a las clases bien definida, va a ser más sencillo el acoplamiento de los nuevos módulos, manteniendo una granularidad moderada, una cohesión alta y un acoplamiento bajo, aplicando patrones descubriendo nuestras estructuras en estos nuevos módulos.

Al final de cuentas de lo que se trata es de desarrollar una solución a un problema dado de la forma más rápida posible, clara y concisa.

Sin embargo como dijo Isaac Newton, “Si he llegado a ver mas lejos que otros, es por que me subí a hombros de gigantes”

Aprovechemos este consejo y construyamos e innovemos usando la experiencia que otros nos han legado en los patrones y estructuremos nuestras propias arquitecturas para seguir creciendo y llegar a ser nosotros mismos unos gigantes.

Bibliografía

[1] Abbott, R. 1983. “Program Design by Informal English Descriptions.” Communications of the ACM vol. 26(11).

[2] Alexander, C., Ishikawa, S. y Silverstein, M. 1977. “A Pattern Language-Towns-Building-Construction.” Oxford University Press.

[3] Barros Justo, José Luis “A repository to support requirement specifications reuse” Proc of ICNZ ’96 Int. Cont on Information Systems of New Zealand, Palmerston North New Zealand, October 1996

[4] Beck, K. y Cunningham, W. 1989. “A Laboratory for Object-oriented Thinking.” Proceedings of OOPSLA 89. SIGPLAN Notices, vol. 24, núm. 10.

[5] Beck, K. 1994. “Patterns and Software Development.” Dr Dobbs Journal. Feb., 1994.

[6] Booch, G., Jacobson, I. y Rumbaugh, J. 1997. “The UML specification documents.” Santa Clara, CA.: Rational Software Corp. Véanse los documentos en www.rational.com.

[7] Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P. y Stal, M. 1996. “A System of Patterns.” West Sussex, Inglaterra: Wiley.

[8] Booch, G., 1994. “Object-Oriented analysis and Design” Redwood City CA.: Benjamin/ Cummings.

[9] Booch, G., 1996. “Object Solutions: Managing the Object-Oriented Project.” Menlo Park, CA.: Addison-Wesley.

[10] Brooks, F., 1975. “The Mythical Man-Month.” Reading, MA.: Addison-Wesley.

[11] Brown, K. y Whitenack, B. 1996. Crossing Chasms. “Pattern Languages of Program Design vol. 2.” Reading, MA.: Addison-Wesley.

[12] Caldiera, Gianluigi; Basili, Victor R. “Identifying and Qualifying Reusable Software Componentes” Computer, IEEE Computer Society, Vol. 24, No. 2, pp. 61-69 February 1991

[13] Coad, P. 1995. “Object Models: Strategies Patterns and Applications.” Englewood Cliffs, NJ.: Prentice-Hall.

[14] Coleman, D., et al. 1994. “Object-Oriented Development: The Fusion Method.” Englewood Cliffs, NJ.: Prentice-Hall.

[15] Constantine, L. 1997. “The Case for Essential Use Cases.” Object Magazine. Mayo, 1997. NY, NY: SIGS Publications.

[16] Coplien, J. 1995. “The History of Patterns.” Véase http://c2.com/cgi/wiki?HistoryOf-Patterns.

[17] Fowler, M. 1996. “Analysis Patterns: Reusable Object Models.” Reading, MA: Addison-Wesley.

[18] Gamma, E., Helm, R., Johnson, R. y Vlissides, J. 1995. “Design Patterns.” Reading, MA.: Addison-Wesley.

[19] Gause, D. y Weinberg, G. 1989. “Exploring Requirements.” NY, NY.: Dorset House.

[20] Jacobson, I., et al. 1992. “Object-Oriented Software Engineering: A Use Case Driven Approach.” Reading, MA.: Addison-Wesley.

[21] Lieberherr, K., Holland I. y Riel. A. 1988. “Object-Oriented Programming: An Objective Sense of Style.” OOPSLA 88 Conference Proceedings. NY, NY: ACM SIGPLAN.

[22] Martin, J. y Odell, J. 1995. “Object-Oriented Methods: A Foundation.” Englewood Cliffs, NJ.: Prentice-Hall.

[23] Rumbaugh, J., et al. 1991. “Object-Oriented Modelling and Design.” Englewood Cliffs, NJ.: Prentice-Hall.

[24] Rumbaugh, J. 1997. “Models Through Development Process.” Journal of Object-Oriented Programming, Mayo. 1997. NY, NY: SIGS Publications.

[25] Schmidt, Douglas C.; Fayad, Mohamed; Johnson, Ralph E. “Software Patterns.” Comunications of the ACM, 1996.

[26] Schulte, R, 1995. “Three-Tier Computing Architectures and Beyond.” Informe Publicado. GartnerGroup.

[27] Wirfs-Brock, R. 1993. “Designing Scenarios: Making the Case for a Use Case Frame-work.” Smalltalk Report, nov-dic 1993, NY,NY: SIGS Publications.