



Universidad Nacional de Río Cuarto  
Facultad de Ciencias Exactas Físico, Química y Naturales  
Departamento de Computación

**Trabajo final para el título de Analista en Sistemas**

**“Sistema Web Inmobiliario  
Martínez Latorre”**

Integrantes:

- ✓ Ocampo, Esteban. DNI: 33046044.
- ✓ Pérez García, Agustín. DNI: 34771232.

*Noviembre 2014. Río Cuarto, Córdoba.*

# Índice

1 Introducción .....	6
2 Narrativa del problema .....	7
3 Captura de requerimientos .....	9
3.1 Modelo de dominio .....	9
3.2 Modelo de negocio .....	10
3.2.1 Reglas de negocio .....	10
3.2.2 Glosario de términos .....	10
3.2.3 Actores .....	11
3.2.4 Casos de uso del negocio .....	11
3.3 Modelo de casos de uso del sistema .....	19
3.3.1 Casos de uso del sistema .....	20
3.3.4 Planificación de desarrollo del proyecto .....	26
4 Primera Iteración .....	29
4.1 Etapa de Análisis .....	29
4.1.1 Introducción .....	29
4.1.2 Modelo de análisis .....	30
4.1.3 Clases de Análisis .....	30
4.1.4 Realización de caso de uso-análisis .....	31
4.1.5 Paquete de análisis .....	31
4.1.6 Descripción de la arquitectura .....	32
4.1.7 Caso de uso: Gestionar Alquiler .....	32
4.1.8 Gestionar venta .....	36
4.2 Etapa de Diseño .....	40
4.2.1 Introducción .....	40
4.2.2 Modelo de diseño .....	40
4.2.3 Clase de diseño .....	41
4.2.4 Realización de caso de uso-diseño .....	41
4.2.5 Subsistema de diseño .....	41
4.2.6 Descripción de la arquitectura .....	42
4.2.7 Caso de uso: Gestionar Alquiler .....	44
4.2.8 Caso de uso: Gestionar venta .....	48
4.2.9 Modelo de persistencia .....	52

4.3 Implementación .....	54
4.3.1 Introducción .....	54
4.3.2 Modelo de Implementación.....	54
4.3.3 Modelo de Despliegue .....	55
4.3.4 Modelo de Componentes.....	56
4.4 Prueba .....	57
4.4.1 Introducción .....	57
4.4.2 Caja Blanca .....	57
4.4.3 Caja Negra .....	59
5 Segunda Iteración .....	60
5.1 Análisis.....	60
5.1.1 Introducción .....	60
5.1.2 Caso de uso: Gestionar Cliente .....	60
5.1.3 Caso de uso: Listados .....	63
5.2 Diseño.....	66
5.2.1 Introducción .....	66
5.2.2 Caso de uso: Gestionar Cliente .....	66
5.2.3 Caso de uso: Listados .....	69
5.3 Implementación .....	74
5.4 Prueba .....	74
5.4.1 Introducción .....	74
5.4.2 Caja Blanca .....	74
5.4.3 Caja Negra .....	76
6 Conclusiones .....	77
7 Trabajos futuros .....	78
8 Bibliografía.....	79
9 Anexos.....	80
9.1 Anexo I Proceso Unificado .....	80
9.1 Anexo II Patrones de Diseño .....	86
9.2 Anexo III Notación BPMN .....	95
9.3 Anexo IV Plantilla Genérica para descripción de caso de uso.....	101

## Índice de figuras

Figura 1: Modelo de Dominio .....	9
Figura 2: Diagrama de casos de uso del negocio.....	12
Figura 3: Diagrama BPD para caso de uso: Alquilar inmueble.....	15
Figura 4: Diagrama BPD para caso de uso: Vender inmueble .....	16
Figura 5: Diagrama BPD para caso de uso: Recepcionar Inmueble .....	17
Figura 6: Diagrama BPD para caso de uso: Administrar inmueble .....	18
Figura 7: Diagrama BPD para caso de uso: Registrar subasta .....	19
Figura 8: Trazabilidad de los casos de uso del negocio a los casos de uso del sistema.....	20
Figura 9: Diagrama de casos de uso del sistema.....	22
Figura 10: Descripción de caso de uso adaptable a plantilla genérica: Gestionar Venta.....	23
Figura 11: Descripción de caso de uso adaptable a plantilla genérica: Gestionar Alquiler .....	24
Figura 12: Descripción de caso de uso no adaptable a plantilla genérica: Listar inmueble por ubicación .....	25
Figura 13: Descripción de caso de uso no adaptable a plantilla genérica: Verificar Administrador .....	25
Figura 14: Clasificación de casos de uso en principales y secundarios .....	26
Figura 15: Planificación dividida en 2 iteraciones (diagrama de Gantt) .....	28
Figura 16: Trazabilidad de casos de uso del sistema a clases de análisis para el caso de uso: Gestionar Alquiler .....	33
Figura 17: Diagrama de clases de análisis para caso de uso: Gestionar Alquiler.....	34
Figura 18: Diagrama de colaboración del caso de uso: Gestionar Alquiler (escenario 1) .....	35
Figura 19: Diagrama de colaboración del caso de uso: Gestionar Alquiler (escenario 2) .....	36
Figura. 20: Trazabilidad de casos de uso del sistema a clases de análisis para caso de uso: Gestionar Venta.....	37
Figura 21: Diagrama de clases de análisis para caso de uso: Gestionar Venta .....	38
Figura 22: Diagrama de colaboración del caso de uso: Gestionar Venta (escenario 1) .....	38
Figura 23: Diagrama de colaboración del caso de uso: Gestionar Venta (escenario 2) .....	39
Figura 24: Trazabilidad de clases de análisis a clases de diseño para caso de uso: Gestionar Alquiler.....	45
Figura 25: Diagrama de clases de diseño para caso de uso: Gestionar Alquiler .....	46
Figura 26: Diagrama de secuencia para caso de uso: Gestionar Alquiler (escenario 1).....	47
Figura 27: Diagrama de secuencia para caso de uso: Gestionar Alquiler (escenario 2).....	48
Figura 28: Trazabilidad de clases de análisis a clases de diseño para caso de uso: Gestionar Venta .....	49
Figura 29: Diagrama de clases de diseño para caso de uso: Gestionar Venta.....	50
Figura 30: Diagrama de secuencia para caso de uso: Gestionar Venta (escenario 1) .....	51
Figura 31: Diagrama de secuencia para caso de uso: Gestionar Venta (escenario 2) .....	52
Figura 32: Modelo de clases persistentes.....	53

Figura 33: Diagrama de Despliegue.....	55
Figura 34: Diagrama de Componentes .....	56
Figura 35: Grafo de flujo para criterio de cobertura de arco de caso de uso: Alta Propietario...	58
Figura 36: Prueba de clases de equivalencia para caso de uso: Alta Propietario.....	59
Figura 37: Trazabilidad de casos de uso del sistema a clases de análisis para caso de uso: Gestionar Cliente.....	61
Figura 38: Diagrama de clases de análisis para caso de uso: Gestionar Cliente .....	61
Figura 39: Diagrama de Colaboración del caso de uso: Gestionar Cliente (escenario 1) .....	62
Figura 40: Diagrama de Colaboración del caso de uso: Gestionar Cliente (escenario 2) .....	63
Figura 41: Trazabilidad de casos de uso del sistema a clases de análisis para caso de uso: Listados .....	64
Figura 42: Diagrama de clases de análisis para caso de uso: Listados .....	64
Figura 43: Diagrama de Colaboración del caso de uso: Listados (escenario 1) .....	65
Figura 44: Diagrama de Colaboración del caso de uso: Listados (escenario 2) .....	65
Figura 45: Trazabilidad de clases de análisis a clases de diseño para caso de uso: Gestionar Cliente .....	66
Figura 46: Diagrama de clases de diseño para caso de uso: Gestionar Cliente .....	67
Figura 47: Diagrama de secuencia para caso de uso: Gestionar Cliente (escenario 1) .....	68
Figura 48: Diagrama de secuencia para caso de uso: Gestionar Cliente (escenario 2) .....	69
Figura 49: Trazabilidad de clases de análisis a clases de diseño para caso de uso: Listados ..	70
Figura 50: Diagrama de clases de diseño para caso de uso: Listados .....	71
Figura 51: Diagrama de secuencia para caso de uso: Listados (escenario 1) .....	72
Figura 52: Diagrama de secuencia para caso de uso: Gestionar Cliente (escenario 2) .....	73
Figura 53: Grafo de flujo para criterio de cobertura de arco de caso de uso: Modificar Cliente	75
Figura 54: Prueba de clases de equivalencia para caso de uso: Modificar Cliente .....	76

## **1 Introducción**

En el marco de proyecto final de la carrera de Analista en Sistemas, se desarrolló un sistema web para la Inmobiliaria Martinez Latorre.

En la ingeniería de software se denomina Sistema web a aquellas herramientas que los usuarios pueden utilizar accediendo a un servidor web a través de Internet mediante un navegador. En otras palabras, es una aplicación software que se codifica en un lenguaje soportado por los navegadores web en la que se confía la ejecución al navegador.

La metodología de desarrollo que se utilizo fue Proceso Unificado (ver Anexo I), el cual es un marco de desarrollo de software que se caracteriza por estar dirigido por casos de uso, centrado en la arquitectura y por ser iterativo e incremental.

La inmobiliaria Martinez Latorre es una inmobiliaria fundada en el año 2007, con sede en Rio Cuarto, la cual se dedica a la venta y alquiler de inmuebles, ofreciendo asesoramiento para cualquier inversión inmobiliaria, garantizando profesionalismo y atención personalizada en cada operación, cuenta con un equipo de profesionales especialistas en legislación y contrato, asesoría contable, arquitectura, tasaciones, construcción, mantenimiento de inmuebles y decoración.

## **2 Narrativa del problema**

La inmobiliaria Martínez Latorre se dedica a la venta, alquiler de inmuebles. Estos inmuebles pueden ser casas, oficinas, departamentos, campos, quintas o terrenos.

Para vender y alquilar inmuebles, la empresa lleva un registro manual de datos de los mismos que son dados a conocer por el propietario. Este firma una planilla que lo compromete por un tiempo determinado a no operar en forma particular. El propietario se queda con un duplicado de la planilla y la empresa con el original. Una vez firmada la planilla que le otorga a la empresa cierto poder sobre el inmueble, el propietario procede a la entrega de una copia de las llaves.

Cuando un cliente se acerca a la inmobiliaria con la intención de comprar, vender o alquilar un inmueble, el mismo es atendido por un operador que lo asesorará convenientemente mediante explicación verbal, fotografías y videos de las propiedades, como también la posibilidad de visitar el inmueble en cuestión acompañado por personal de la empresa.

En el caso de que el cliente se decida por alquilar la propiedad, se le entrega una planilla de solicitud de alquiler para que llene con sus datos personales y otros de interés de la empresa. Estos datos quedan registrados a disposición de la empresa. Al garante se le facilita una planilla para ser completada con diversos datos y se le solicita la documentación respaldatoria, de esta forma se verifica a través de internet que los datos, tanto los del cliente como los del garante son verídicos y confiables. En caso de concretarse el alquiler se procede a firmar un contrato entre las partes involucradas.

La operatoria para la venta es similar a la de alquiler, sólo que en el caso de concretarse una venta, se deriva la misma a un escribano para que certifique el contrato de compra-venta. El escribano actúa como fedatario de la operación. La venta puede realizarse de contado o financiada. Si se realiza financiada, se entrega la escritura una vez concretado el pago.

Sea una venta o un alquiler la que se efectué, además de los datos del inmueble, se registran manualmente también los del propietario y los del comprador o inquilino, con sus observaciones pertinentes.

Una vez realizada la venta o alquiler, el pago de los impuestos y servicios los pueden realizar el propietario, el inquilino, la inmobiliaria o pueden repartirse dichos gastos. La inmobiliaria posee un registro de los mismos.

En el caso de un inquilino, se le permite trabajar con un saldo deudor de hasta \$200, esto significa que el cliente puede hacer entregas de dinero en todo momento en el local de atención, y su deuda no debe superar nunca el saldo establecido. Si esto sucede, se informa al inquilino y se le da un plazo de dos días para normalizar su cuenta. Superados estos dos días, y no habiendo saldado la deuda, se envía una carta documento al garante informando de tal hecho. Nuevamente, si a las 48 horas no se efectuó el pago, se procede a la anulación del contrato, desalojo y por lo tanto, quita de llaves. La inmobiliaria para realizar estas actividades, consulta diariamente el saldo de los titulares de las cuentas corrientes. Los clientes que no pagaron a término se agandan como clientes morosos.

Si el titular de la cuenta corriente es el propietario, la inmobiliaria deposita en su cuenta el pago de los alquileres. En todo momento, este puede consultar o bien retirar todo o una parte del dinero acumulado.

El pago de los impuestos y servicios los pueden realizar el propietario, el inquilino, la inmobiliaria o bien repartirse. Para la liquidación de impuestos y servicios que realiza la inmobiliaria existe un registro de los mismos.

Mensualmente a los propietarios que ofrecen en alquiler su inmueble, se les cobra honorarios de administración, es decir, la comisión que le corresponde a la inmobiliaria. Al inquilino junto con el pago del primer mes de alquiler se le cobra un mes de depósito, que también va como comisión para la inmobiliaria. Para el caso de una venta, la comisión en un porcentaje del monto total del inmueble.

También, cada propiedad tiene asignada una llave que puede estar en manos de la inmobiliaria, el propietario o de un tercero, por ej. otra inmobiliaria con la que se comparte la información.

La empresa no realiza las subastas, sino que solo lleva un registro manual de las mismas y de aquellas a realizar.

Esta inmobiliaria comparte ciertos datos de propiedades a vender o alquilar, con otras inmobiliarias, para que tanto una u otra tengo la opción de concretar la operación. Los datos que se comparten son restringidos y se tratan de la propiedad en particular y nunca del propietario. La compartimentación de datos se basa en tener la posibilidad de poder concretar alguna operación de venta o alquiler por parte de las inmobiliarias que comparten los datos, teniendo prioridad aquella que posea la propiedad. En el caso de que alguna inmobiliaria con la que Martínez Latorre comparte datos, concrete una venta o alquiler, la comisión es del 50% de las ganancias para cada una.

La empresa atiende consultas sobre disponibilidad de propiedades, personalmente, por teléfono o e-mail. Si existe tal disponibilidad, se informa de las características de la misma; si no se haya y el cliente está de acuerdo, se agenda la consulta para posteriormente comunicarle al interesado que ya se dispone del inmueble consultado (en caso de que así sea). Esta última actividad se realiza periódicamente.

No es necesario que un interesado consulte por una propiedad de las formas mencionadas anteriormente, sino que la inmobiliaria tiene pegados en una pizarra pequeños carteles que informan sobre los datos más importantes de los inmuebles a alquilar y vender, los mismos están situados dentro del local de atención.

La inmobiliaria mantiene también, información acerca de los propietarios, clientes (inquilinos, compradores) con sus garantes y martilleros. Estos datos son registrados manualmente y obtenidos de las planillas que firman estos cuando se concreta alguna operación o cuando un propietario deja su propiedad para vender o alquilar. Para el caso del martillero, solo se registran los datos cuando se subasta una propiedad, y los mismos se consiguen de los ficheros de las propiedades subastadas.

La inmobiliaria lleva un registro manual de las publicaciones realizadas y ha realizar. Así como también de los carteles que posee la empresa utilizados en las propiedades a vender o alquilar. En el caso de los carteles, estos son retirados una vez que se concreta una venta o alquiler y se los deja listos para una posterior utilización.

### 3 Captura de requerimientos

En esta etapa comienza a emplearse la metodología de Proceso Unificado, la cual es un marco de desarrollo de software que se caracteriza por estar dirigido por casos de uso, centrado en la arquitectura y por ser iterativo e incremental.

El objetivo del Proceso Unificado en esta etapa es, suponiendo que el usuario no es un especialista informático, ser capaces de hacer entender al cliente el resultado de los requisitos, utilizando el lenguaje del cliente e introduciendo (con mucho cuidado) formalidad y estructuras.

#### 3.1 Modelo de dominio

Un modelo de dominio captura los tipos más importantes de objetos en el contexto del sistema. Los objetos del dominio representan las cosas que existen o los eventos que suceden en el entorno en el que trabaja el sistema.

Muchas clases del dominio pueden obtenerse o bien de una especificación de requisitos o mediante entrevistas con expertos del dominio. Las clases del dominio aparecen generalmente en 3 formas típicas:

- Objetos del negocio, que representan cosas que se manipulan en el negocio.
- Objetos del mundo real y conceptos de los que sistema debe hacer un seguimiento.
- Sucesos que ocurrirán o han ocurrido.

El modelo del dominio se describe mediante diagramas de clases UML. Estos diagramas muestran a los clientes, usuarios, revisores y a otros desarrolladores las clases de dominio y como se relacionan unas con otras mediante asociaciones.

En la figura 1 se muestra el modelo de dominio del sistema.

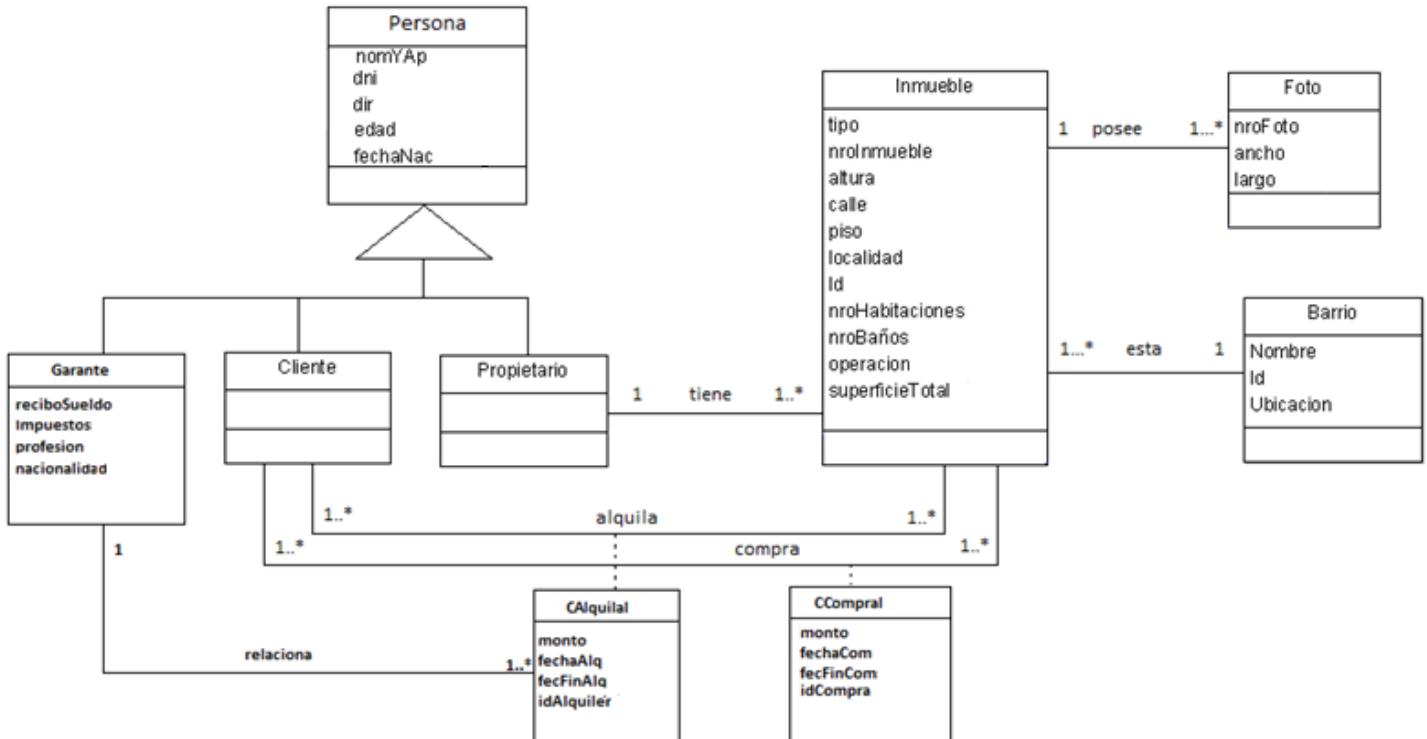


Figura 1: Modelo de Dominio

## **3.2 Modelo de negocio**

Estudia el negocio sobre el que se implementará el sistema. Este modelo permite:

- Entender la estructura y la dinámica de la organización.
- Entender los problemas corrientes e identificar potenciales o posibles mejoras.
- Asegurar a los clientes, usuarios, y desarrolladores que tienen una comprensión común de la organización.
- Derivar los requerimientos del sistema, necesarios para soportar la estructura y dinámica de la organización.

### **3.2.1 Reglas de negocio**

En una organización, tanto los procesos como los datos que estos manejan, están restringidos por las reglas de negocio. Estas reglas aseguran que la actividad de la empresa se lleva a cabo de acuerdo a las restricciones impuestas desde el entorno (leyes o normas) o desde adentro de la propia organización.

Para este sistema hemos encontrado las siguientes reglas de negocio:

- La empresa no comparte datos con otras inmobiliarias que no se encuentren matriculadas.
- En el caso de que se comparte información de una propiedad, no se deben informar datos del propietario de la misma.
- No se realizan alquileres y ventas de inmuebles a clientes menores de 21 años.
- El garante de un alquiler o venta no debe ser menor de 21 años y no puede tener deudas.
- A cada garante de una venta o alquiler se le solicita una documentación respaldatoria que consta de una fotocopia del DNI y de los bienes q posee.
- El contrato del alquiler se firma sólo entre el dueño de la inmobiliaria, el inquilino y el garante.
- En la venta financiada de un inmueble, sólo se entregan las llaves, una factura o un recibo. La escritura sólo se entrega una vez finalizado el pago del mismo.
- En la venta de contado de un inmueble, se entregan las llaves, una factura y la escritura.
- La comisión en el alquiler de una propiedad, corresponde a un mes de alquiler y honorarios al propietario.
- La comisión en la venta de inmueble consta de un porcentaje del monto total por la cual se hizo.
- El pago de impuestos y servicios los pueden realizar el propietario, el inquilino, la inmobiliaria o repartírselos.

### **3.2.2 Glosario de términos**

Define los términos más importantes usados por el negocio. La definición debe ser única y consistente durante todo el proyecto.

Cliente: Persona que habitualmente compra o alquila inmuebles en la inmobiliaria.

Inmobiliaria: Empresa o sociedad que se dedica a construir, arrendar, vender y administrar viviendas.

Garante: Persona que ofrece su patrimonio como garantía de las obligaciones asumidas por el inquilino.

Inquilino: Arrendatario de una vivienda.

Propietario: Persona que tiene derecho de propiedad sobre una cosa, en este caso de bienes inmuebles. La persona que ofrece su inmueble para alquilar o comprar.

Subasta: Venta pública de bienes que se hace al mejor postor.

Arrendar: Ceder o adquirir por un precio el aprovechamiento temporal de un inmueble.

### 3.2.3 Actores

Los actores representan a cualquier ente externo que colabora con el sistema. Cada usuario y cada sistema externo con los que interactúa el sistema son representados por uno o más actores. Un actor juega un papel por cada caso de uso con el que colabora. Cada vez que un usuario interactúa con el sistema, la instancia correspondiente del actor está desarrollando ese papel.

Los actores que se identificaron en el sistema son:

Operador de venta: cumple la función de ser el intermediario entre el cliente y el dueño del inmueble.

Cliente: Es el interesado en la compra o alquiler de un inmueble.

Propietario del inmueble: Dueño de el/los inmueble/s que ofrece/n la inmobiliaria.

Garante: persona que sirve de garantía para una operación de venta o alquiler.

BCRA: representa a la base del Banco Central de la República Argentina, la cual cumple el rol de verificar la morosidad de un cliente o un garante.

### 3.2.4 Casos de uso del negocio

Los casos de uso del negocio representan los procesos del negocio de una empresa, es decir, funcionalidades más significativas que esta misma ejerce.

Los casos de uso más importantes que se encontraron son:

1. Alquilar inmueble
2. Vender inmueble.
3. Recepcionar inmueble.
4. Administrar inmueble.
5. Registrar subasta.

La figura 2 es un diagrama que modela las principales funciones o procesos del negocio. Su principal objetivo es ayudar al logro de un mejor y claro entendimiento del contexto del negocio, al que posiblemente se le desarrollará e implementará un sistema de software. También facilita la identificación de roles y responsabilidades de los miembros del negocio.

El modelo de negocio se construyó a partir de la información obtenida en entrevistas mantenidas con el cliente. Este modelo nos permite crear una representación de la realidad organizacional actual de la Inmobiliaria e identificar los actores y las distintas actividades que se llevan a cabo allí. Se puede observar a continuación el Diagrama de casos de uso del Negocio.

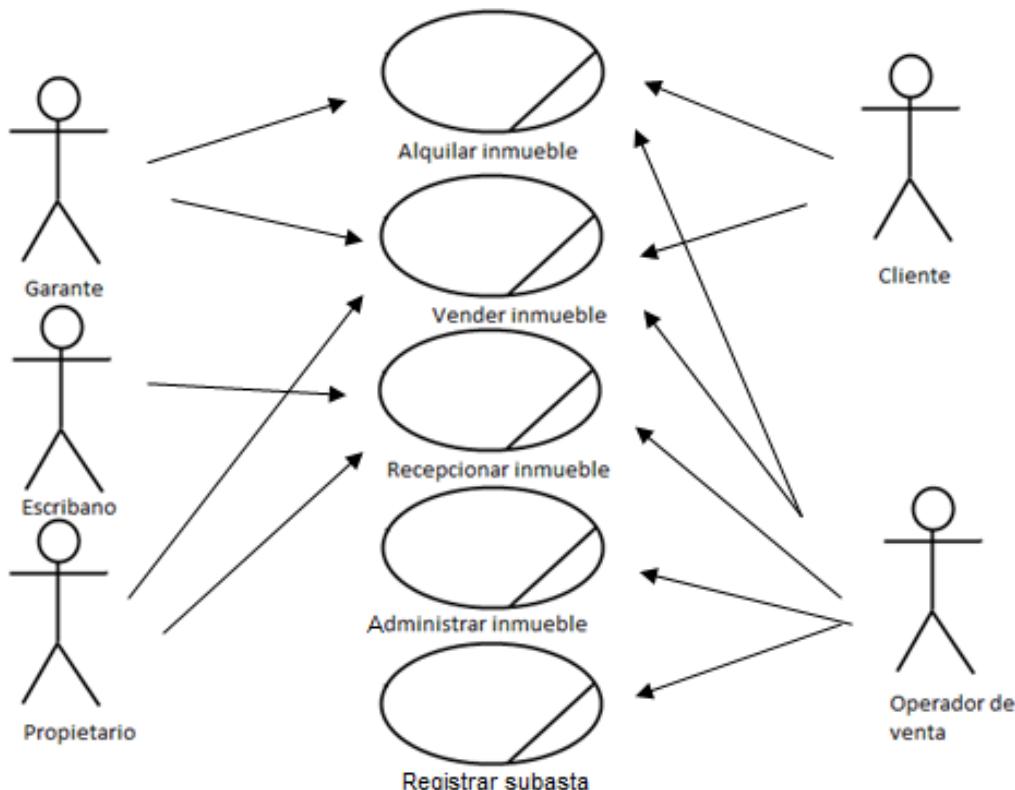


Figura 2: Diagrama de casos de uso del negocio

### 3.2.4.1 Descripción de casos de uso

Para la descripción de los casos de uso del negocio se utiliza un diagrama BPD (ver anexo III), este diagrama está diseñado para representar gráficamente la secuencia de todas las actividades que ocurren durante un proceso, incluye además toda la información que se considera necesaria para el análisis.

A continuación se detalla cada caso de uso seguido de sus diagramas BPD.

#### **Alquilar Inmueble**

El cliente consulta personalmente, por teléfono, o e-mail al empleado de la inmobiliaria la disponibilidad de una propiedad para alquilar, de acuerdo a sus preferencias. El empleado procede a buscar si dispone de dicho inmueble conforme a las características expresadas por

el cliente. En caso de no disponer del inmueble solicitado, si el cliente lo desea, se agenda la consulta, para que transcurridos los días acordados, comunicarle que se dispone o no de un inmueble con las características solicitadas. Caso contrario, se le informa acerca de las propiedades que se le pueden ofrecer.

Si el cliente está interesado en alguna propiedad en particular, se lo lleva a ver el inmueble personalmente, y si decidió alquilar, se entrega una planilla de solicitud de alquiler, para que se llene con sus datos personales y otros que son de interés para el negocio. También se entrega una planilla para que la complete el garante con sus datos particulares y se solicita una documentación respaldatoria. Ambas planillas sirven a la empresa como registro del cliente y del garante. Una vez finalizado este trámite, de acuerdo a lo informado en las planillas y a la documentación respaldatoria, se verifica vía internet en las bases de DECIDIR (empresa especializada en prestar servicios de procesamiento de datos a empresas e instituciones financieras que quieren desarrollar el canal de comercio electrónico) y BCRA si no se trata de morosos. En caso de que el cliente sea un moroso se rechaza la posibilidad de alquilarle la propiedad. En caso de que el moroso es el garante, se explica al cliente que debe buscar otro garante para poder realizar el contrato de alquiler entre las partes involucradas (inmobiliaria e inquilino). Cuando terminan de firmar el contrato, se entrega las llaves del inmueble al inquilino.

Por un alquiler, la inmobiliaria cobra como comisión un mes de depósito al inquilino y un valor fijo mensual al propietario. La inmobiliaria realiza liquidación de impuestos y servicios. El pago de estos los pueden realizar el propietario, el inquilino, la inmobiliaria o bien repartirse.

En la figura 3 se muestra el diagrama BPD para el caso de uso: Alquilar inmueble.

## Vender Inmueble

El cliente consulta personalmente, por teléfono, o vía web al personal de la inmobiliaria por la disponibilidad de un inmueble de acuerdo a sus preferencias. El personal de la inmobiliaria procede a buscar si dispone de dicho inmueble. Si el cliente está a gusto con lo informado se lo lleva a ver el inmueble personalmente. En caso de que el cliente está interesado en comprar, se entrega al mismo una panilla para que llene con sus datos personales y otros de interés para la empresa. En caso de no disponer del inmueble solicitado, si el cliente lo desea, se agenda la consulta, para que transcurridos los días acordados, comunicarles que se dispone o no de un inmueble con las características solicitadas.

Luego, mediante presencia de un escribano se firma un contrato de compra-venta entre el dueño de la inmobiliaria, el comprador, el escribano y el propietario.

Si la paga es de contado, y se ejecuta en forma total por el valor del inmueble, entonces se hace entrega al comprador de las llaves, la escritura y la factura correspondiente. Ahora, si el pago es en forma parcial lo único que se le habilita al cliente son las llaves y un recibo provisorio por la cantidad abonada. Cuando termina de pagar el monto adeudado se entrega la factura y la escritura del inmueble.

Si el pago es financiado, la inmobiliaria entrega al cliente únicamente la llave de la propiedad y la factura por el monto abonado; y si es parcial, un recibo por dicho monto. Cuando el cliente cancela la deuda en forma total, se hace entrega de la escritura. La inmobiliaria cobra una comisión de la venta, que es un porcentaje del monto total del inmueble.

En la figura 4 se muestra el diagrama BPD para caso de uso: Vender inmueble.

## **Repcionar Inmueble**

El propietario le informa a la inmobiliaria personalmente, por teléfono o mail que tiene una propiedad para vender y/o alquilar. La inmobiliaria le explica las condiciones y el manejo de la empresa sobre las propiedades en comisión. Si el cliente está de acuerdo, se toman los datos de la misma y se registran los datos del cliente.

Se entrega al propietario una planilla que compromete al mismo por un tiempo determinado a no operar en forma particular (ajena a la empresa). El propietario se quedará con un duplicado de la misma y la empresa con el original.

También se toma nota si se van a compartir datos o no con otras inmobiliarias, los mismos son compartidos con inmobiliarias matriculadas.

Se solicita una copia de las llaves para poder mostrar el inmueble a algún interesado. Se imprime una tarjeta para exponer en vidriera los datos de la propiedad, con el fin de captar clientes, y finalmente se le asigna un cartel que es colocado en la propiedad, guardando los datos del mismo manualmente.

En la figura 5 se muestra el diagrama BPD para caso de uso: Repcionar inmueble.

## **Administrar Inmueble**

Tanto a un inquilino, como a un propietario de un inmueble, se le otorga la posibilidad de tener en cuenta corriente un saldo deudor de hasta \$200. Si el inquilino es el que acepta, se pide la documentación que satisface los requisitos impuestos por el negocio. Este puede abonar cuando lo deseé la cantidad de dinero que quiera en el local de atención de la inmobiliaria. Si el saldo de la cuenta corriente es deudor y supera los \$200, se informa al mismo y se da un plazo de dos días para normalizar su cuenta. Superado los dos días, y no habiendo saldado la deuda, se envía una carta documento al garante informando de tal hecho. Nuevamente, si a las 48 horas no se efectuó el pago, se procede a la anulación del contrato, desalojo, quita de llaves y se agenda al cliente como moroso.

Si el titular de la cuenta corriente es el propietario, la inmobiliaria deposita en su cuenta el pago de los alquileres.

En la figura 6 se muestra el diagrama BPD para caso de uso: Administrar inmueble.

## **Registrar Subasta**

Al operador de venta le informan que se ha realizado o que se realizará una subasta. El operador de venta registra los datos.

En la figura 7 se muestra el diagrama BPD para caso de uso: Registrar subasta.

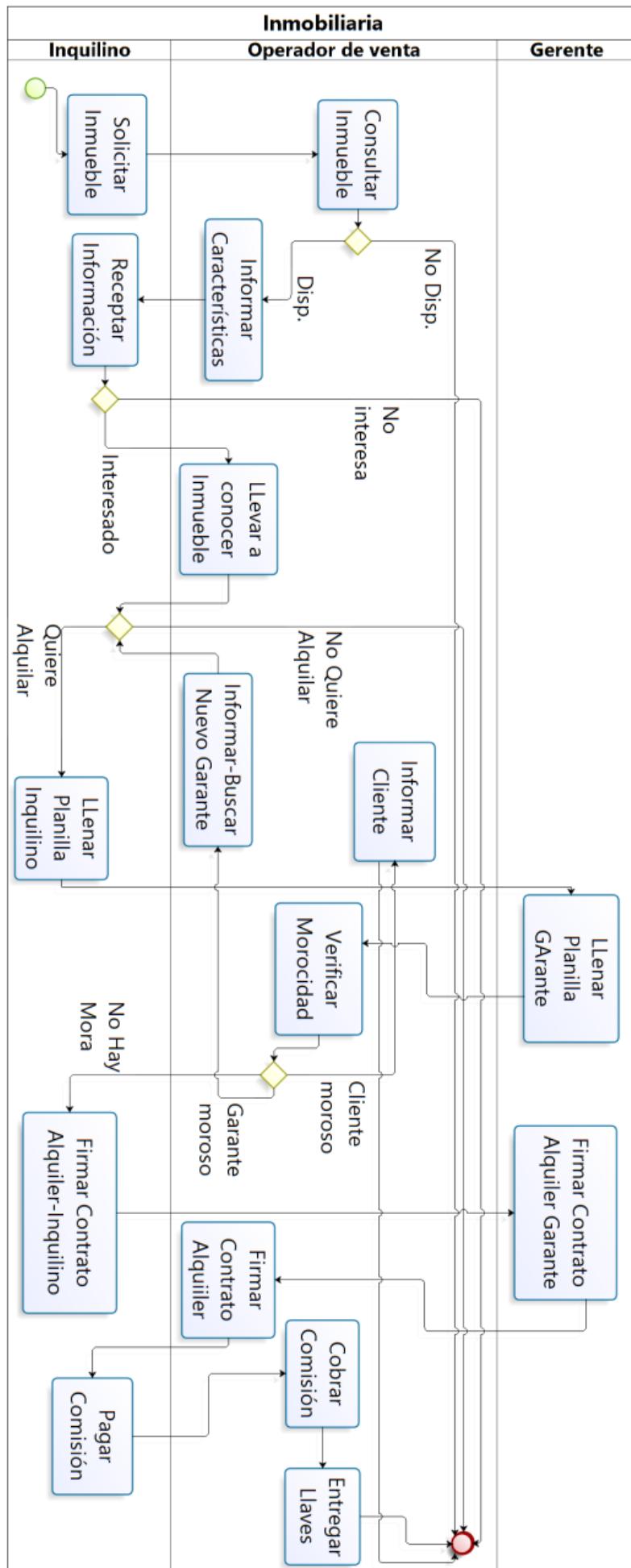


Figura 3: Diagrama BPD para caso de uso: Alquilar inmueble

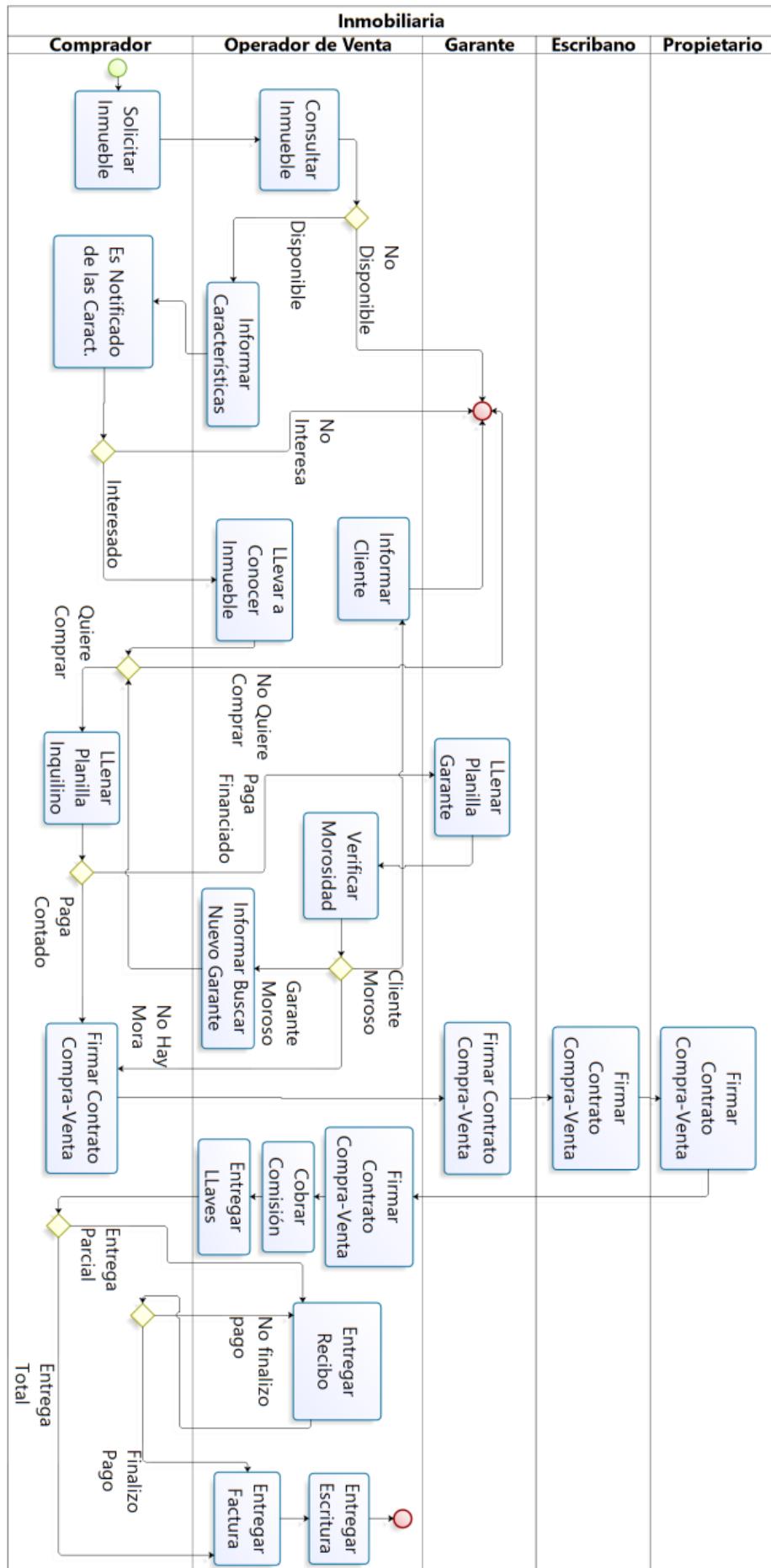


Figura 4: Diagrama BPD para caso de uso: Vender inmueble

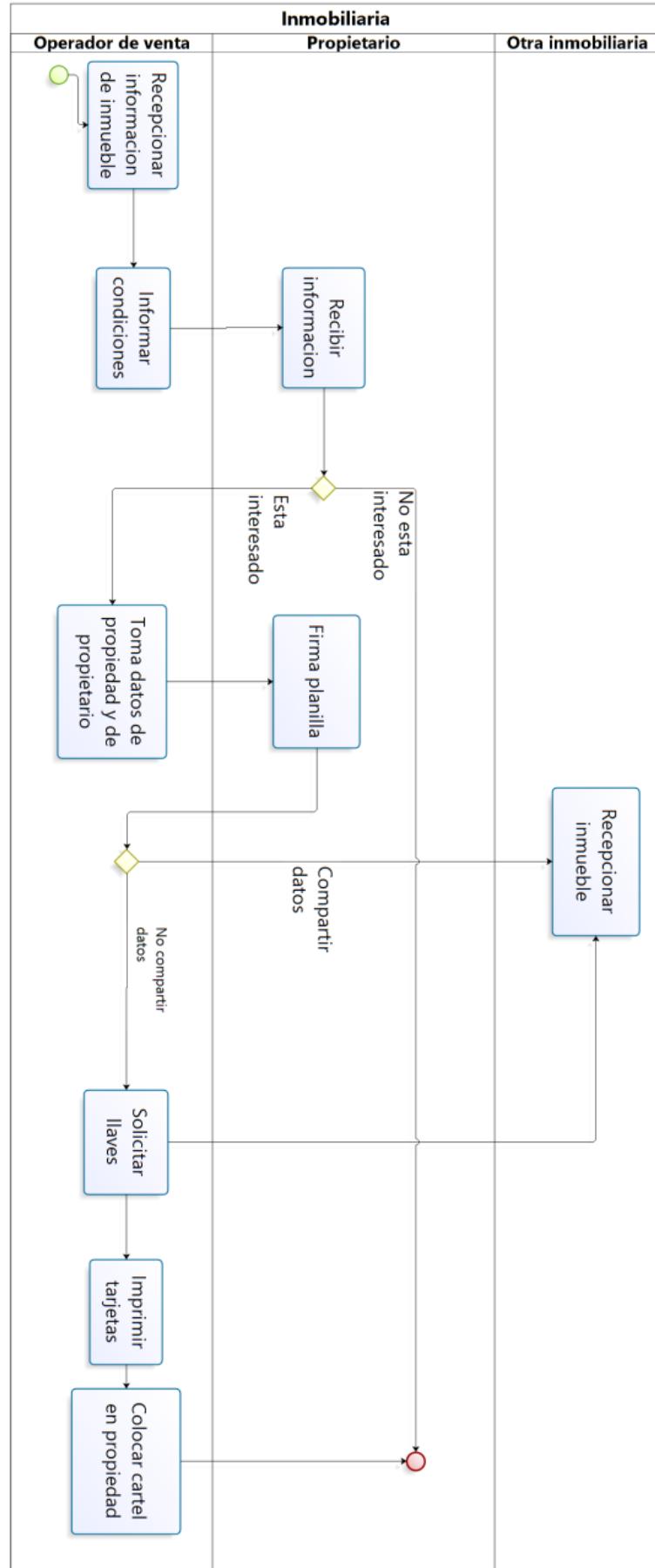


Figura 5: Diagrama BPD para caso de uso: Recepcionar Inmueble

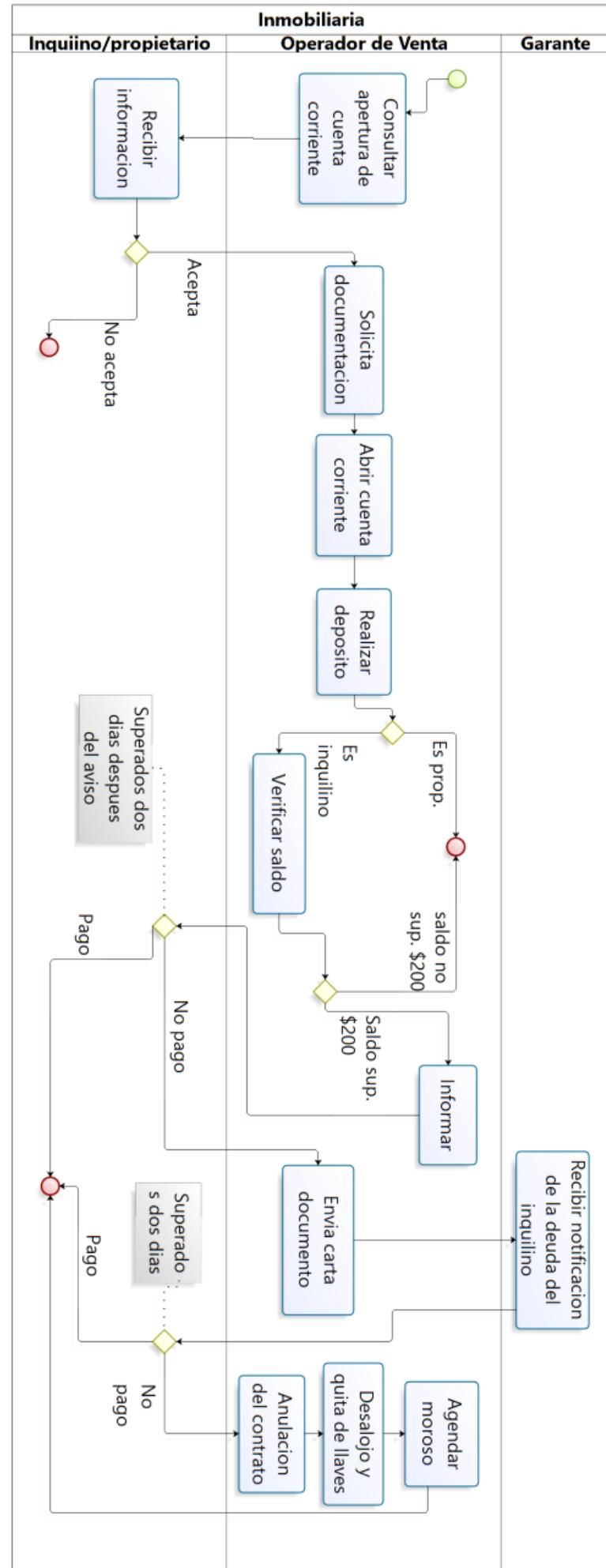


Figura 6: Diagrama BPD para caso de uso: Administrar inmueble

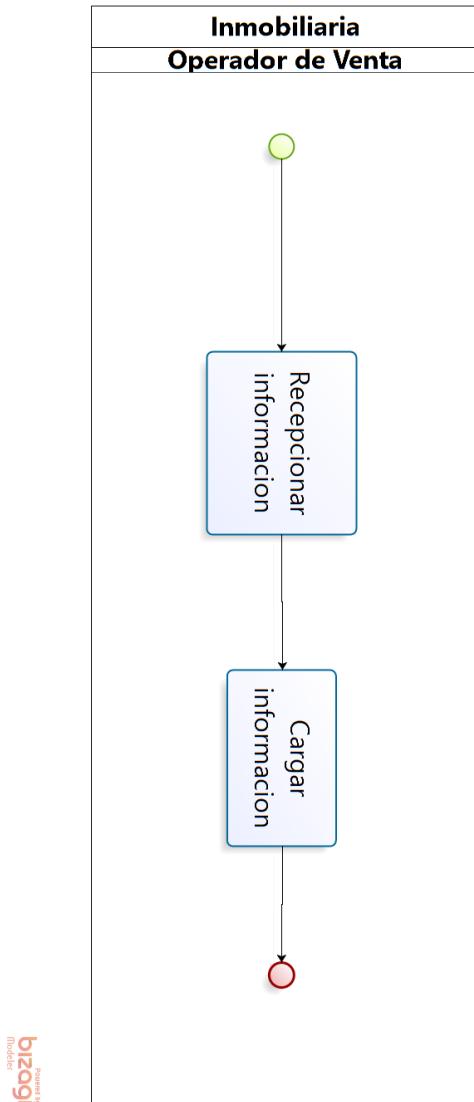


Figura 7: Diagrama BPD para caso de uso: Registrar subasta

### 3.3 Modelo de casos de uso del sistema

El modelo de casos de uso del sistema es un modelo que describe los requerimientos del sistema en términos de casos de uso. Su principal propósito es comunicar el comportamiento del sistema a los clientes y/o usuarios finales.

El modelo de casos de uso del sistema permite a los desarrolladores de software y a los clientes llegar a un acuerdo sobre las condiciones y posibilidades que debe cumplir el sistema. El modelo de casos de uso del sistema contiene actores, casos de uso y las relaciones que existen entre ambos, que se muestra gráficamente en el diagrama de caso de uso. En definitiva, el modelo de casos de uso es el resultado de los requerimientos del sistema y es usado como entrada a las etapas de análisis, diseño y prueba.

En la figura 8 se muestra la trazabilidad de los casos de uso del negocio a los casos de uso del sistema.

Caso de uso de negocio	Caso de uso de Sistema
Alquiler inmueble	Gestionar alquiler(ABM) Consultar alquiler Consultar inmueble Gestionar cliente Gestionar garante(ABM) Consultar garante Enviar solicitud
Vender inmueble	Consultar inmueble Gestionar cliente Consultar venta(ABM) Consultar cliente Gestionar venta Enviar solicitud
Repcionar inmueble	Verificar administrador Gestionar cliente Gestionar inmueble Control administrador Consultar cliente Gestionar solicitudes Gestionar posibles clientes Gestionar propietario
Administrar inmueble	Consulta administrador Consultar cliente Listados(10)

Figura 8: Trazabilidad de los casos de uso del negocio a los casos de uso del sistema

### 3.3.1 Casos de uso del sistema

A continuación se listarán los casos de uso del sistema:

- Consultar inmueble
- Consultar cliente
- Consultar alquiler
- Consultar venta
- Consultar administrador
- Consultar garante
- Consultar propietario
- Verificar administrador
- Enviar solicitud
- Listar operaciones:
  - Por disponibilidad
  - Por ubicación
  - Por precio
  - Por prop. Alquiladas
  - Por prop. Vendidas
  - Por prop. Para alquilar
  - Por prop. Para vender
  - Por ventas de administradores
  - Por Alquileres de administradores

- De clientes
- Gestionar alquiler
  - Ingresar alquiler
  - Modificar alquiler
  - Eliminar alquiler
- Gestionar venta
  - Ingresar venta
  - Modificar venta
  - Eliminar venta
- Gestionar administrador
  - Ingresar inmueble
  - Modificar inmueble
  - Eliminar inmueble
- Gestionar inmueble
  - Ingresar inmueble
  - Modificar inmueble
  - Eliminar inmueble
- Gestionar cliente
  - Ingresar cliente
  - Modificar cliente
  - Eliminar cliente
- Gestionar Garante
  - Ingresar garante
  - Modificar garante
  - Eliminar garante
- Gestionar solicitud
  - Alta solicitud
  - Baja solicitud
- Gestionar posibles clientes
  - Alta posible cliente
  - Baja posible cliente
- Gestionar propietario
  - Alta propietario
  - Baja propietario
  - Modificar propietario

El la figura 9 se representan los casos de uso y su interacción con los actores. Su ventaja principal es la facilidad para interpretarlos, lo que hace que sean especialmente útiles en la comunicación con el cliente.

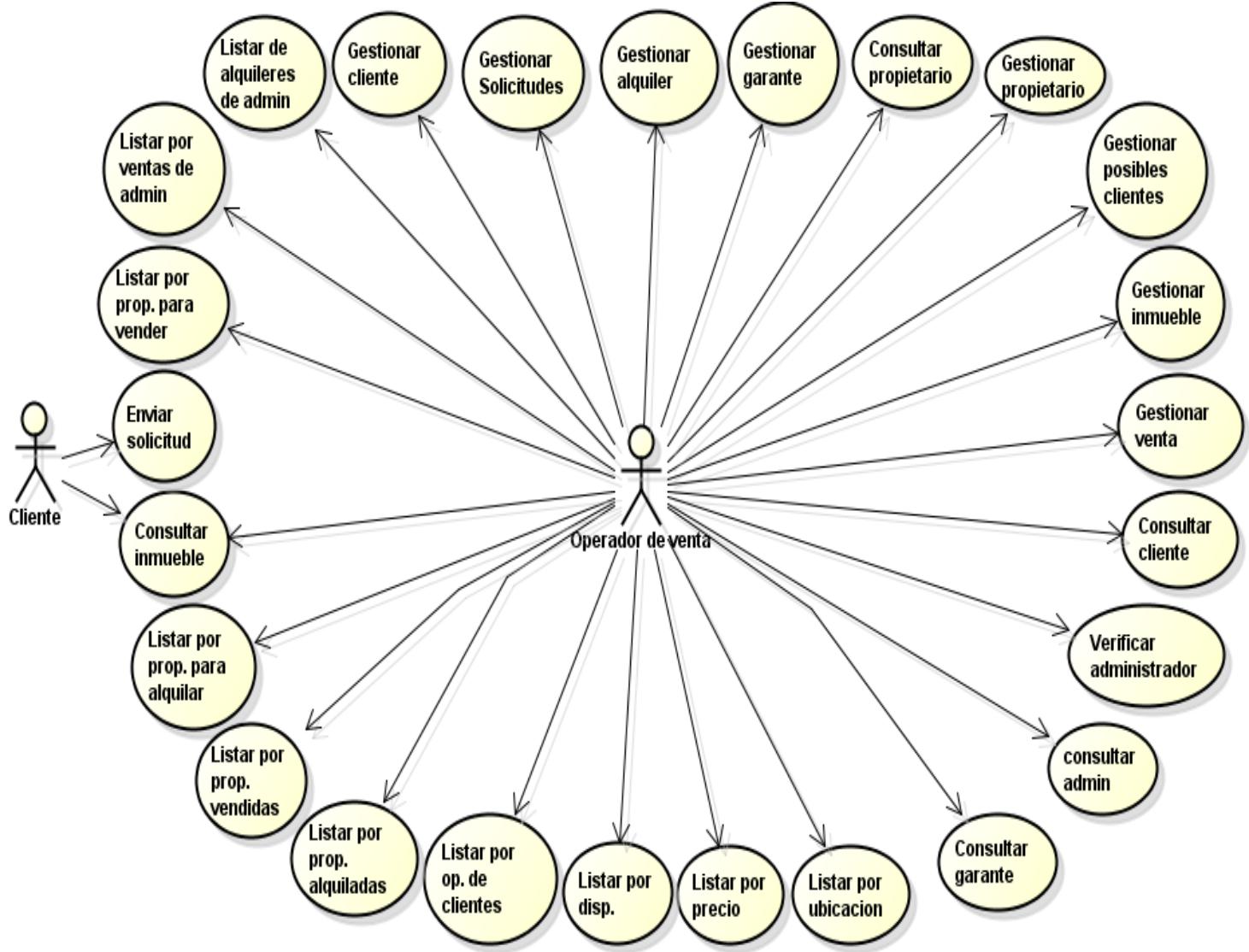


Figura 9: Diagrama de casos de uso del sistema

## Uso de Plantillas Genéricas para la descripción de Casos de Uso

Se presenta la descripción de casos de uso del sistema, algunos de los cuales se adjuntan a una descripción general mientras que existen otros casos de uso que no se ajustan a estas plantillas; estos casos de uso, serán tratados de manera diferente con una descripción más detallada de su comportamiento.

## Descripción de los Casos de Uso adaptables a Plantillas Genéricas

Con la utilización de las plantillas (ver anexo IV), se busca identificar Casos de uso similares en su comportamiento y explicar en forma precisa algunas características de estos, para guiar en el desarrollo del proceso de software.

Se presentan instancias de estas plantillas para describir esos comportamientos. Estas se componen de la asignación de valores a los actores que interactúa con el caso de uso, a cada uno de los atributos considerados en los mismos, describiendo si son claves,

verificaciones de formato o consideraciones especiales y la inclusión de otros casos de uso por medio de una acción a realizar.

A continuación se muestran dos ejemplos de instancias de las plantillas, correspondientes a los Casos de Uso Gestión de Venta y Gestión de Alquiler.

## Caso De Uso: Gestión de Venta

En la figura 10 se detalla la descripción del caso de uso: Gestión de Venta, adaptado a una plantilla genérica.

Descripción: Este caso de uso se utiliza para insertar, modificar, eliminar y consultar una venta.

Elemento: Venta.

Actor: Operador de venta.

	<u>Instancia: Plantilla 1 (Inserción de &lt;&lt;Venta&gt;&gt;)</u>			<u>Instancia: Plantilla 3 (Eliminación de &lt;&lt;Venta&gt;&gt;)</u>		<u>Instancia: Plantilla 2 (Modificación de &lt;&lt;Venta&gt;&gt;)</u>		<u>Instancia: Plantilla 4 (Consultar &lt;&lt;Venta&gt;&gt;)</u>	
ATRIBUTOS	CLAVE	VERIFICACION	ACCION	VERIFICACION	ACCION	VERIFICACION	ACCION	VERIFICACION	ACCION
IdVenta	SI	Debe ser no nulo y único	Include (Generar Id)	Validar existencia		Validar existencia		Es un criterio de búsqueda, no nulo	
monto		Debe ser no nulo							
fecha		Debe ser no nulo  El formato es de 10 dígitos separando el día del mes y el mes del año con barras(“/”)				No se modifica			

Figura 10: Descripción de caso de uso adaptable a plantilla genérica: Gestión de Venta

## Caso De Uso: Gestionar Alquiler

En la figura 11 se detalla la descripción del caso de uso: Gestionar Alquiler, adaptado a una plantilla genérica.

Descripción: Este caso de uso se utiliza para insertar, modificar, eliminar y buscar un alquiler.

Elemento: Alquiler.

Actor: Operador de venta

	Instancia: Plantilla 1 (Inserción de <<Alquiler>>)			Instancia: Plantilla 3 (Eliminación de <<Alquiler>>)		Instancia: Plantilla 2 (Modificación de <<Alquiler>>)		Instancia: Plantilla 4 (Consultar <<Alquiler>>)	
ATRIBUTOS	CLAVE	VERIFICACION	ACCION	VERIFICA CION	ACCION	VERIFICA CION	ACCION	VERIFICA CION	ACCIO N
idAlquiler	SI	Debe ser no nulo y único	Include (Generar Id)	Validar existencia		Validar existencia		Es un criterio de búsqueda, no nulo	
Monto		Debe ser no nulo.							
fecha		Debe ser no nulo. El formato es de 10 dígitos separando el día del mes y el mes del año con barras(“/”)				No se modifica			

Figura 11: Descripción de caso de uso adaptable a plantilla genérica: Gestionar Alquiler

## Descripción de Casos de Uso no adaptables a Plantillas Genéricas

A continuación se presenta la descripción de comportamiento de los Casos de Uso del Sistema Listar operaciones por ubicación y Verificar administrador.

Los Casos de Uso se muestran estableciendo una descripción textual de su comportamiento, detallando sus atributos, el actor, reglas de negocio, pre y pos condiciones, al igual que los genéricos, con la diferencia de que el flujo de eventos es independiente en cada Caso de Uso.

## Listar operaciones por ubicación

En la figura 12 se detalla la descripción del caso de uso: Listar operaciones por ubicación.

Descripción: El operador de venta solicita un listado de las operaciones realizadas en un periodo.

pre-condición: No tiene

Flujo de eventos

Usuario	Sistema
1- Ingres al sistema con un nombre de usuario y contraseña que sean válidos	
2- Ingres a la opción admin	
	3- Muestra el listado con las propiedades disponibles
4- Ingres en la opción “Listar por” ubicación	
	5- Muestra el listado ordenado alfabéticamente por la ubicación de los inmuebles

Figura 12: Descripción de caso de uso no adaptable a plantilla genérica: Listar inmueble por ubicación

Pos-condición: listado emitido por el sistema

## Verificar administrador

En la figura 13 se detalla la descripción del caso de uso: Verificar administrador.

Descripción: El caso de uso comienza cuando el operador de venta desea ingresar al sistema. El caso de uso verifica si el operador de venta tiene permiso para utilizar el sistema.

Precondición: No tiene

Flujo de eventos

Usuario	Sistema
1- Ingres la contraseña para verificar si tiene permiso en la utilización del sistema	
2- Selecciona la opción para confirmar la existencia	
	3- Busca el administrador ingresado
	4- Permite el ingreso al mismo

Figura 13: Descripción de caso de uso no adaptable a plantilla genérica: Verificar Administrador

Pos-condición: Permiso para utilizar el sistema

### 3.3.4 Planificación de desarrollo del proyecto

El primer paso en la planificación del proyecto es definir y documentar los objetivos y restricción del proyecto. Un proyecto necesita un estado claro de objetivos para guiar a los ingenieros en sus decisiones diarias. El objetivo de la planificación es identificar todos los requerimientos externos para restringir los proyectos. Otra decisión crítica es determinar los recursos requeridos por el proyecto. Los recursos incluyen el número y nivel de conocimiento del personal, la cantidad de recursos computacionales. El problema de predecir cuantos ingenieros y otros recursos son necesarios para un proyecto es conocer la estimación del costo del software. El costo de un proyecto de software es directamente proporcional al número de ingenieros que se necesitan para un proyecto.

La estimación del costo es parte de la etapa de planificación de cualquier ingeniería. La diferencia en la estimación del costo en la ingeniería en software y otras disciplinas es que la ingeniería en software, el principal costo está dado por el personal empleado en el proyecto. Además, da una visión del desarrollo del proyecto, se necesita evaluar el proyecto y evaluarla en caso de ser necesaria. En el monitoreo del proyecto, el administrador necesita determinar cuántos trabajadores tiene realmente y cuál es la tarea que van a realizar.

En la figura 14 se listan los casos de uso clasificados en principales y secundarios.

Principales	Secundarios
<ul style="list-style-type: none"><li>• Gestionar alquiler<ul style="list-style-type: none"><li>Ingresar alquiler</li><li>Modificar alquiler</li><li>Eliminar alquiler</li></ul></li><li>• Gestionar venta<ul style="list-style-type: none"><li>Ingresar venta</li><li>Modificar venta</li><li>Eliminar venta</li></ul></li><li>• Gestionar inmueble<ul style="list-style-type: none"><li>Ingresar inmueble</li><li>Modificar inmueble</li><li>Eliminar inmueble</li><li>Consultar inmueble</li></ul></li><li>• Gestionar propietario<ul style="list-style-type: none"><li>Ingresar cliente</li><li>Modificar cliente</li><li>Eliminar cliente</li><li>Consultar propietario</li></ul></li><li>• Gestionar administrador<ul style="list-style-type: none"><li>Ingresar administrador</li><li>Modificar administrador</li><li>Eliminar administrador</li><li>Consultar administrador</li></ul></li><li>• Gestionar solicitud<ul style="list-style-type: none"><li>Ingresar solicitud</li><li>Eliminar solicitud</li></ul></li></ul>	<ul style="list-style-type: none"><li>• Gestionar posible cliente<ul style="list-style-type: none"><li>Ingresar posible cliente</li><li>Eliminar posible cliente</li></ul></li><li>• Gestionar cliente<ul style="list-style-type: none"><li>Ingresar cliente</li><li>Modificar cliente</li><li>Eliminar cliente</li><li>Consultar cliente</li></ul></li><li>• Listar operaciones:<ul style="list-style-type: none"><li>• Por disponibilidad</li><li>• Por ubicación</li><li>• Por precio</li><li>• Por prop. Alquiladas</li><li>• Por prop. Vendidas</li><li>• Por prop. Para alquilar</li><li>• Por prop. Para vender</li><li>• Por ventas de administradores</li><li>• Por Alquileres de administradores</li><li>• De clientes</li></ul></li><li>• Consultar alquiler</li><li>• Consultar venta</li><li>• Verificar administrador</li></ul>

Figura 14: Clasificación de casos de uso en principales y secundarios

## Una planificación dividida en 2 iteraciones

Con el objetivo de incrementar la productividad del proyecto, dado que el equipo trabaja de forma más eficiente cuando tiene objetivos a corto plazo, se optó por dividir el proyecto en 2 iteraciones.

A continuación se listan los Casos de Uso tomados para la Primera Iteración seguido de los Casos de Uso tomados para la Segunda Iteración.

Casos de uso de la Primera Iteración:

- Gestionar alquiler
  - Ingresar alquiler
  - Modificar alquiler
  - Eliminar alquiler
  - Consultar alquiler
- Gestionar venta
  - Ingresar venta
  - Modificar venta
  - Eliminar venta
  - Consultar venta
- Gestionar inmueble
  - Ingresar inmueble
  - Modificar inmueble
  - Eliminar inmueble
  - Consultar inmueble
- Gestionar propietario
  - Ingresar cliente
  - Modificar cliente
  - Eliminar cliente
  - Consultar propietario
- Gestionar administrador
  - Ingresar administrador
  - Modificar administrador
  - Eliminar administrador
  - Consultar administrador

Casos de uso de la Segunda Iteración:

- Gestionar posible cliente
  - Ingresar posible cliente
  - Eliminar posible cliente
- Gestionar cliente
  - Ingresar cliente
  - Modificar cliente
  - Eliminar cliente
  - Consultar cliente
- Listar operaciones:
  - Por disponibilidad
  - Por ubicación
  - Por precio
  - Por prop. Alquiladas
  - Por prop. Vendidas

- Por prop. Para alquilar
- Por prop. Para vender
- Por ventas de administradores
- Por Alquileres de administradores
- De clientes
- Verificar administrador
- Gestionar solicitud
  - Ingresar solicitud
  - Eliminar solicitud

El diagrama de Gantt es una útil herramienta gráfica cuyo objetivo es exponer el tiempo de dedicación previsto para diferentes tareas o actividades a lo largo de un tiempo total determinado.

En la figura 15 se detalla el diagrama de Gantt de este proyecto.

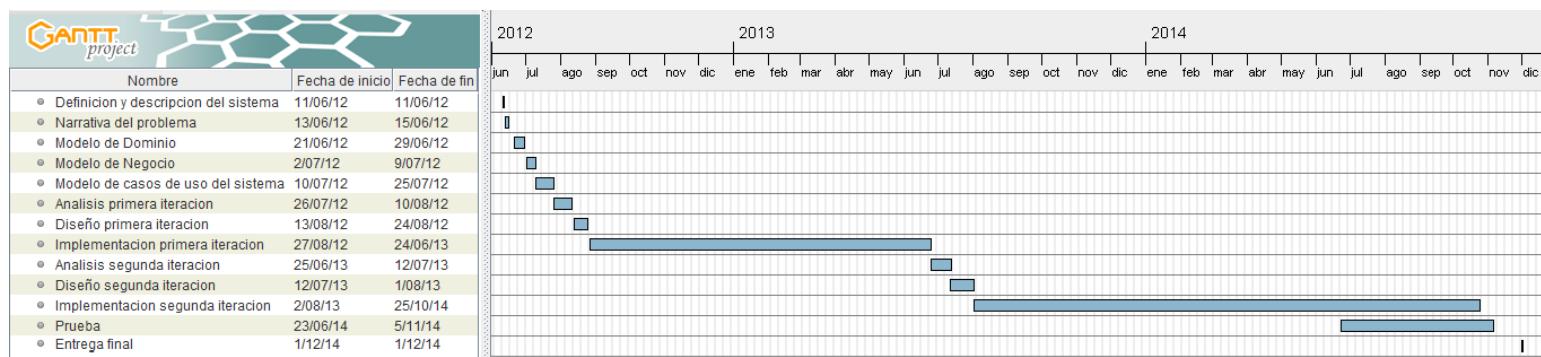


Figura 15: Planificación dividida en 2 iteraciones (diagrama de Gantt)

## 4 Primera Iteración

En esta etapa comienza a desarrollarse la primera iteración la cual consiste en el análisis, diseño, implementación y prueba de los siguientes casos de uso:

- Gestionar alquiler
  - Ingresar alquiler
  - Modificar alquiler
  - Eliminar alquiler
  - Consultar alquiler
- Gestionar venta
  - Ingresar venta
  - Modificar venta
  - Eliminar venta
  - Consultar venta
- Gestionar inmueble
  - Ingresar inmueble
  - Modificar inmueble
  - Eliminar inmueble
  - Consultar inmueble
- Gestionar propietario
  - Ingresar cliente
  - Modificar cliente
  - Eliminar cliente
  - Consultar propietario
- Gestionar administrador
  - Ingresar administrador
  - Modificar administrador
  - Eliminar administrador
  - Consultar administrador

### 4.1 Etapa de Análisis

#### 4.1.1 Introducción

Durante el análisis, se analizan los requisitos que se describieron en la captura de requisitos, refinándolos y estructurándolos.

En este análisis, se hará enfoque en los casos de uso de la primera iteración: gestionar alquiler y gestionar venta. El objetivo, es conseguir una comprensión más precisa de los requisitos y una descripción de los mismos que sea fácil de mantener y que ayude a estructurar el sistema entero. En el análisis se razona más sobre los aspectos internos del sistema para resolver aquellos relativos a la interferencia de casos de uso.

El análisis tiene principal participación en las primeras iteraciones de la fase de elaboración. Esto ayuda a obtener una arquitectura estable y sólida, y facilita una comprensión global de requisitos.

El lenguaje que se utiliza en la etapa de análisis se basa en un modelo de objetos conceptual, que llamamos modelo de análisis.

#### 4.1.2 Modelo de análisis

El modelo de análisis ayuda a refinar los requisitos más detalladamente y permite razonar sobre los aspectos internos del sistema, incluido los recursos internos compartidos. Un recurso interno puede representarse como un objeto en el modelo.

Dentro del modelo de análisis, los casos de uso se describen mediante clases de análisis y sus objetos. Esto se representa a través de colaboraciones dentro del modelo que llamamos realizaciones de caso de uso – análisis.

Los artefactos que componen el modelo de análisis son:

- Clase de análisis
- Realización de casos de uso – análisis
- Paquete de análisis
- Descripción de la arquitectura

Un modelo de análisis posee las siguientes características:

- Descripto con el lenguaje desarrollador
- Se obtiene una vista interna del sistema
- Estructurado por clases y paquetes estereotipados
- Es utilizado por los desarrolladores para comprender como debe darse forma al sistema
- No debe contener redundancias, inconsistencias entre requisitos
- Especifica cómo se debe llevar a cabo la funcionalidad dentro del sistema
- Define realizaciones de casos de uso, y cada una de ellas representa el análisis de un caso de uso del modelo de casos de uso.

#### 4.1.3 Clases de Análisis

Una clase de análisis representa una abstracción de una o varias clases y/o subsistemas del diseño del sistema- Esta abstracción posee las siguientes características.

- Una clase de análisis se centra en el tratamiento de los requisitos funcionales hasta llegar a las actividades de diseño e implementación subsiguiente.
- Su comportamiento se define mediante responsabilidades. Una responsabilidad es una descripción textual de un conjunto cohesivo del comportamiento de una clase.
- Una clase de análisis define atributos. Normalmente los tipos de esos atributos son conceptuales y reconocibles en el dominio del problema.
- Una clase de análisis participa en relaciones.

Las clases de análisis siempre encajan en uno o tres estereotipos básicos: de interfaz, de control o de entidad. Cada estereotipo implica una semántica específica, lo cual contribuye un método potente y consistente de identificar y describir las clases de análisis.

**Clases de interfaz:** Las clases de interfaz se utilizan para modelar la interacción entre el sistema y sus actores. Estas clases modelan las partes del sistema que dependen de sus actores y representan a menudo abstracciones de ventanas, formularios, paneles, interfaces de comunicaciones, impresoras y terminales.

**Clases de entidad:** Se utilizan para modelar información que posee una vida larga y que es persistente. Estas modelan la información y el comportamiento asociado a algún concepto, como una persona, un objeto del mundo real, o un suceso del mundo real.

**Clases de control:** Representan coordinación y control de otros objetos y se usan para encapsular el control de un caso de uso en concreto. Estas manejan y coordinan las acciones y los flujos de control principales, y transmiten trabajos a otros objetos.

#### 4.1.4 Realización de caso de uso-análisis

Una realización de caso de uso-análisis es una colaboración dentro del modelo de análisis que describe como se ejecuta un caso de uso en términos de clase del análisis y de sus objetos del análisis en interacción.

Una realización de caso de uso posee una descripción textual del flujo de eventos, diagramas de clases del análisis y diagramas de interacción para un escenario en particular. Un diagrama de clases de análisis muestra las clases participantes y las relaciones entre estas para un caso de uso concreto. Una clase puede estar presente en varios diagramas de clases.

Un diagrama de interacción del análisis muestra las interacciones entre los objetos de las clases participantes en el desarrollo del caso de uso, creando enlaces entre ellos y añadiendo mensajes a esos enlaces.

Un flujo de eventos-análisis es una descripción textual que complementa un diagrama de interacción. Un flujo de eventos-análisis no debe mencionar atributos, responsabilidades y asociaciones de objetos.

#### 4.1.5 Paquete de análisis

Los paquetes de análisis proporcionan un medio para organizar los artefactos del modelo de análisis en piezas manejables. Un paquete del análisis puede constar de clases de análisis, realizaciones de casos de uso, y de otros paquetes de análisis.

Los paquetes de análisis deben ser cohesivos, es decir, sus contenidos deben estar fuertemente relacionados, y también débilmente acoplados, es decir, que las dependencias entre unos y otros deben ser mínimos.

#### **4.1.6 Descripción de la arquitectura**

La descripción de la arquitectura contiene una vista de la arquitectura del modelo de análisis que muestra sus artefactos significativos para la arquitectura.

Los siguientes artefactos del modelo de análisis se consideran significativos para la arquitectura:

- La descomposición del modelo de análisis en paquetes de análisis y sus dependencias
- Las clases de análisis (de interfaz, de control, de entidad)
- Realizaciones de casos de uso

Arquitectura del análisis:

La arquitectura del análisis de la primera iteración consta de los casos de uso que se detallan a continuación.

#### **4.1.7 Caso de uso: Gestionar Alquiler**

En la figura 16 se muestra la trazabilidad de los casos de uso del sistema a clases de análisis del caso de uso Gestionar Alquiler.

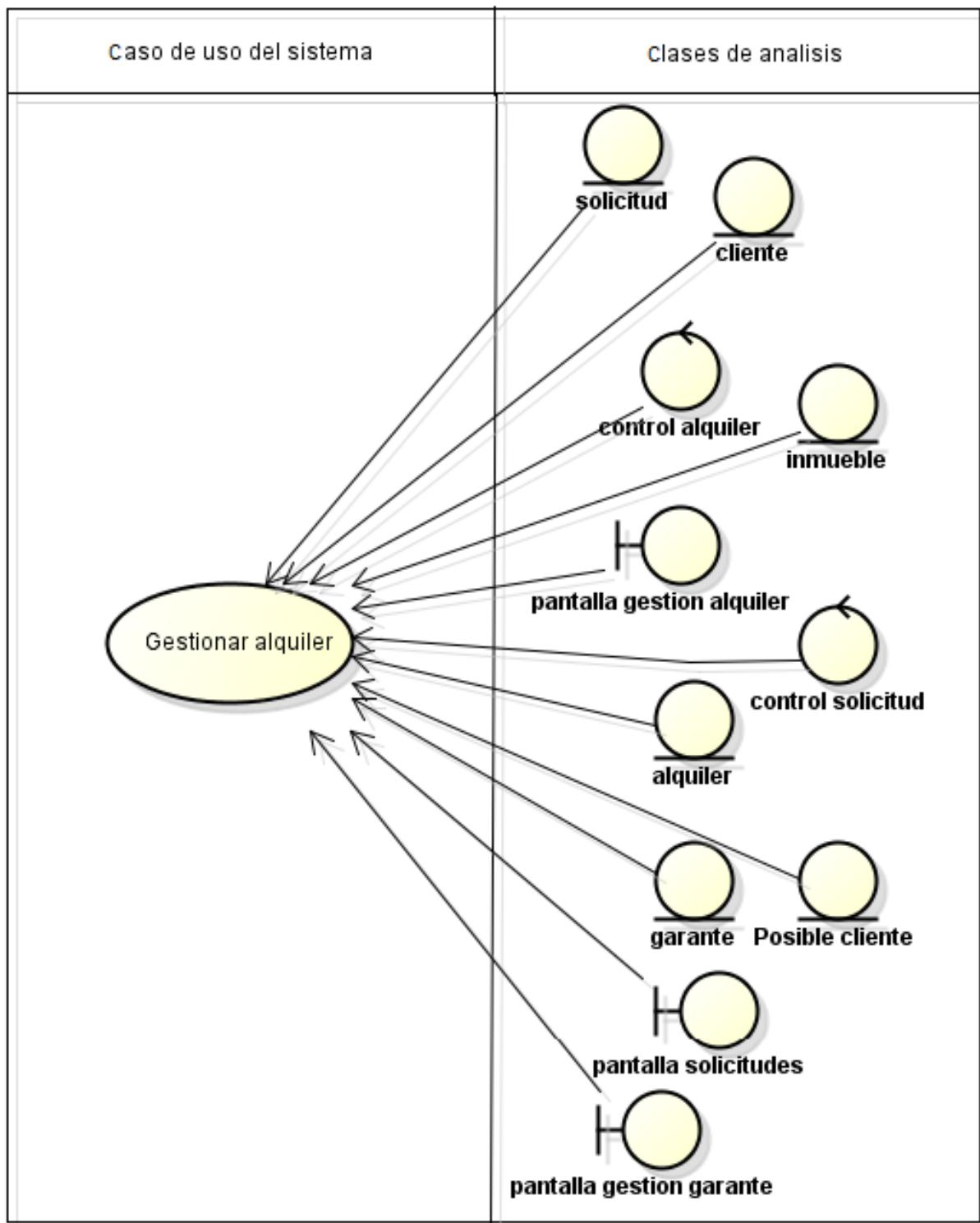


Figura 16: Trazabilidad de casos de uso del sistema a clases de análisis para el caso de uso: Gestionar Alquiler

En la figura 17 se representa el diagrama de clases de análisis para el caso de uso: Gestionar Alquiler.

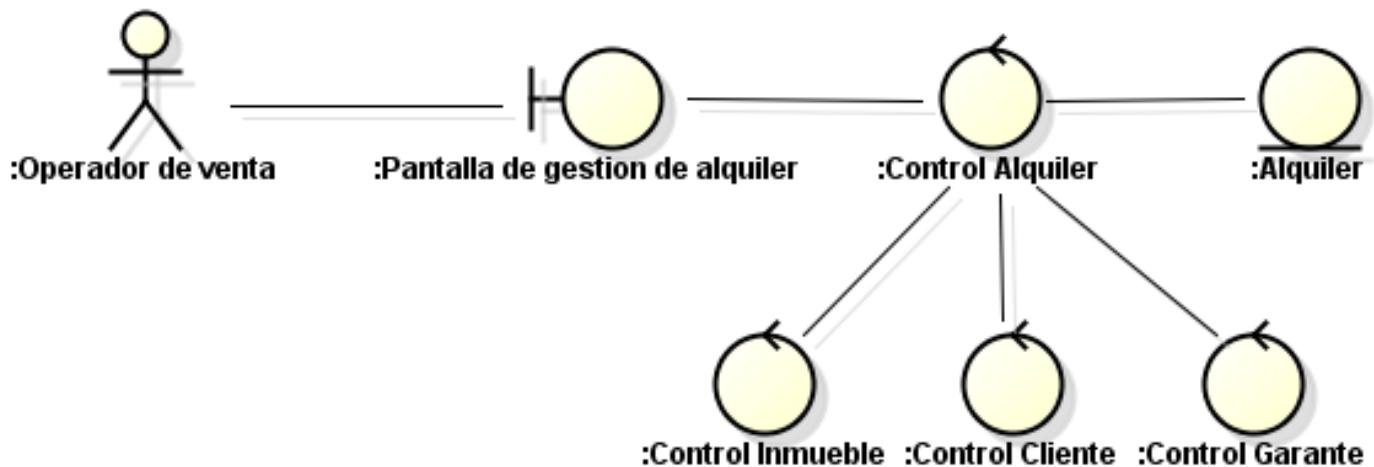


Figura 17: Diagrama de clases de análisis para caso de uso: Gestionar Alquiler

A continuación se detallan escenarios concretos a través de diagramas de colaboración del caso de uso Gestionar Alquiler.

### Escenario 1: Ingresar Alquiler

Descripción: El operador de venta ingresa datos referentes al alquiler de un inmueble en particular que fue solicitado por un cliente.

En la figura 18 se representa el escenario 1 del diagrama de colaboración para el caso de uso: Gestionar Alquiler.

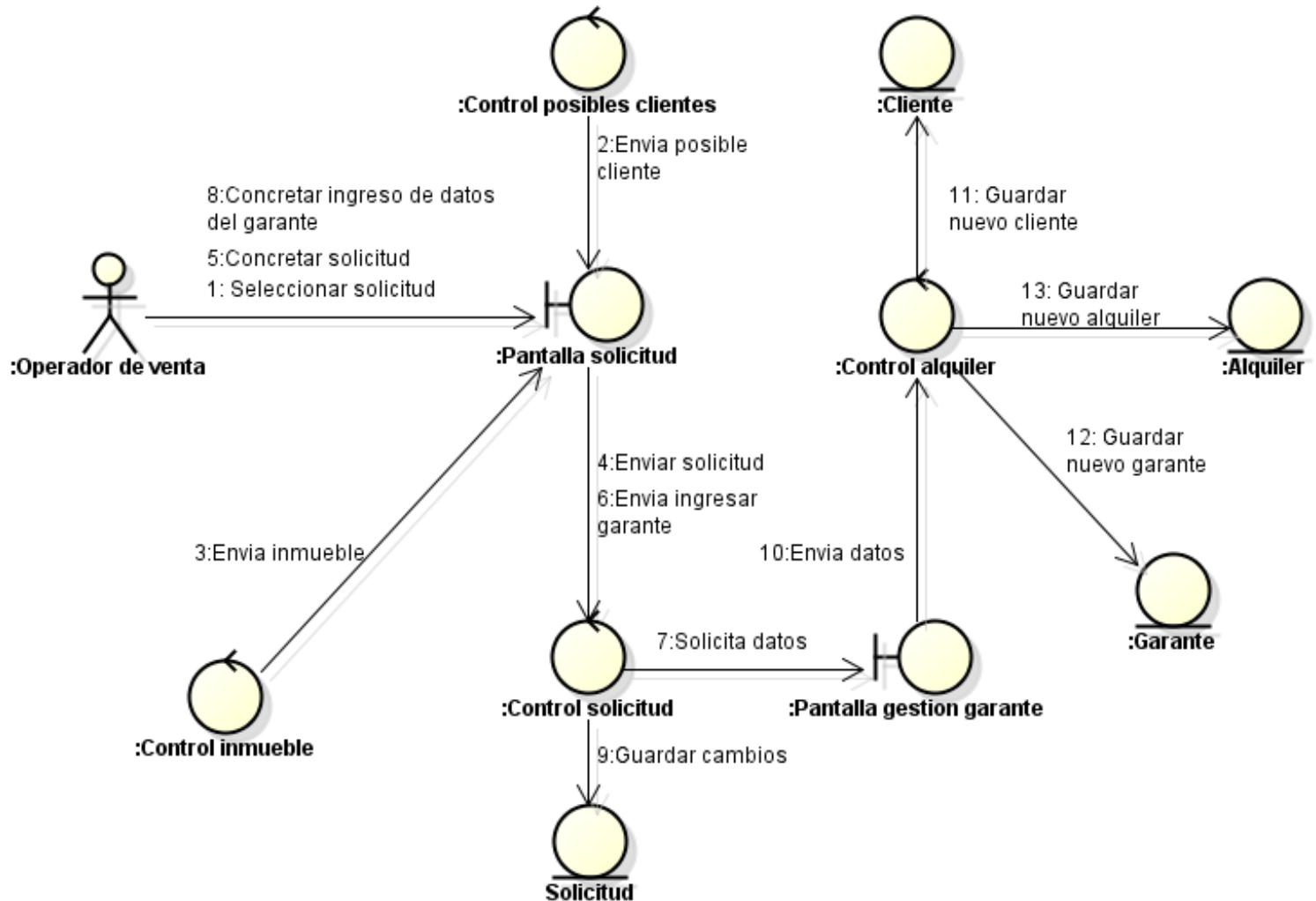


Figura 18: Diagrama de colaboración del caso de uso: Gestionar Alquiler (escenario 1)

Flujo de eventos: El operador de venta selecciona una solicitud (1). La solicitud solicita los datos del inmueble y del posible cliente (2, 3). Luego, se concreta la solicitud y se solicitan los datos del garante (4, 5, 6, 7). Se concretan los datos del garante y la solicitud pasa a ser antigua (8, 9). Finalmente se guardan los datos del nuevo cliente, garante y alquiler en el sistema.

## Escenario 2: Modificar Alquiler

Descripción: el operador de venta modifica datos correspondientes al alquiler de un inmueble.

En la figura 19 se representa el escenario 2 del diagrama de colaboración para el caso de uso: Gestionar Alquiler.

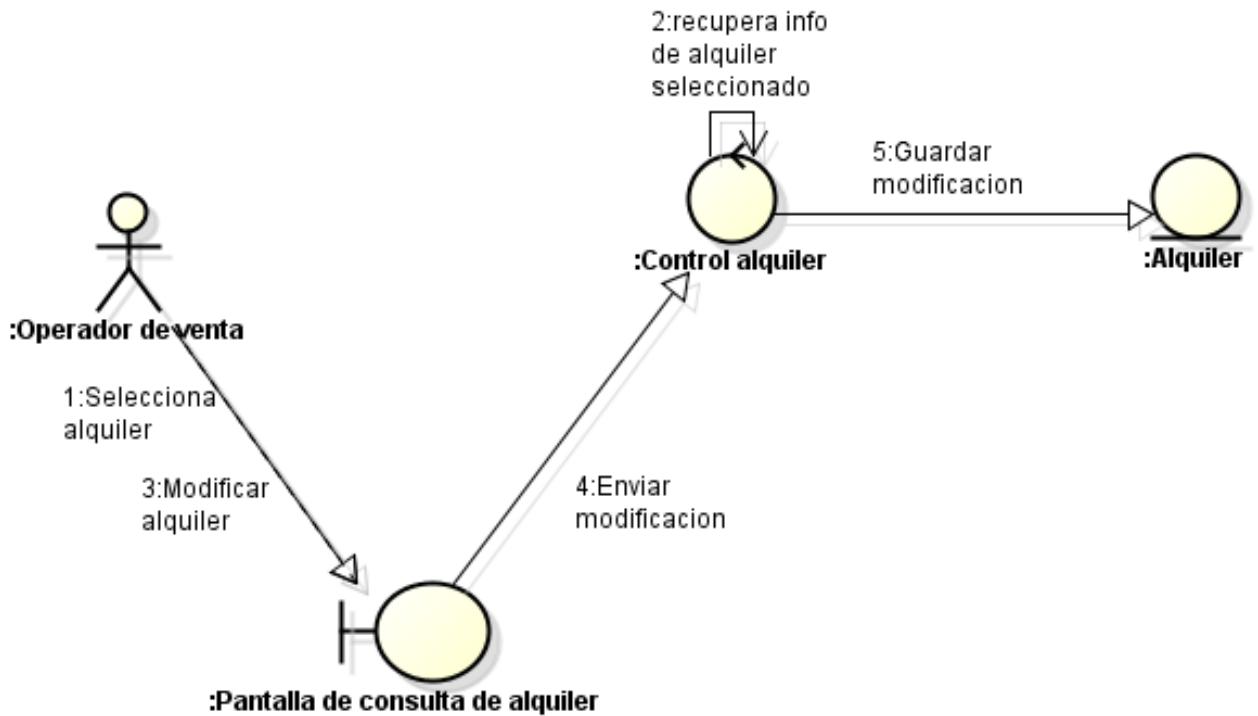


Figura 19: Diagrama de colaboración del caso de uso: Gestionar Alquiler (escenario 2)

Flujo de eventos: El operador de venta selecciona un alquiler (1). Luego de recuperar la información de ese alquiler, modifica los datos que desea y guarda los cambios (2, 3, 4 ,5).

#### 4.1.8 Gestionar venta

En la figura 20 se representa la Trazabilidad de casos de uso del sistema a clases de análisis para caso de uso: Gestionar Venta.

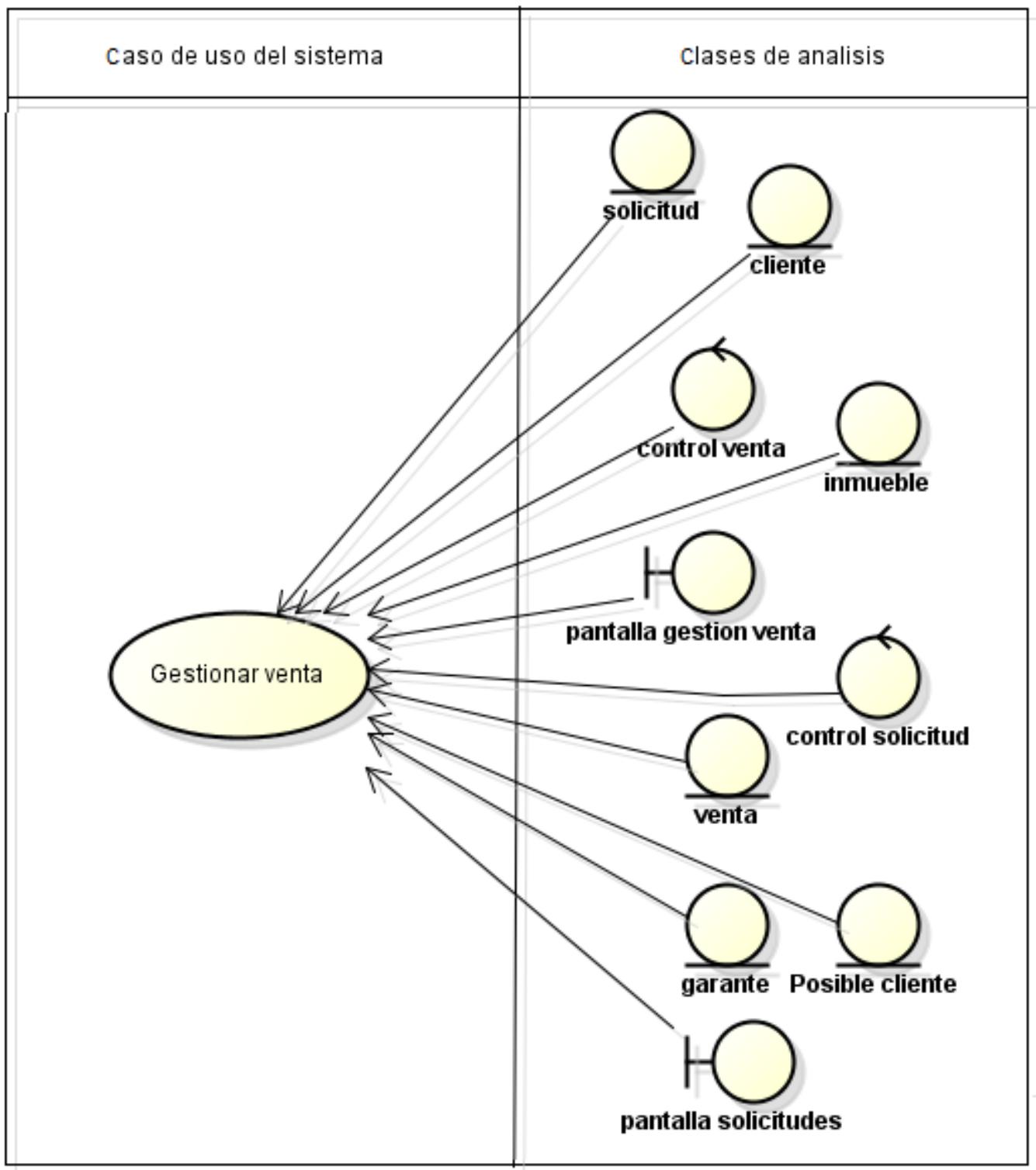


Figura. 20: Trazabilidad de casos de uso del sistema a clases de análisis para caso de uso: Gestionar Venta

En la figura 21 se representa el diagrama de clases de análisis para el caso de uso: Gestionar Venta.

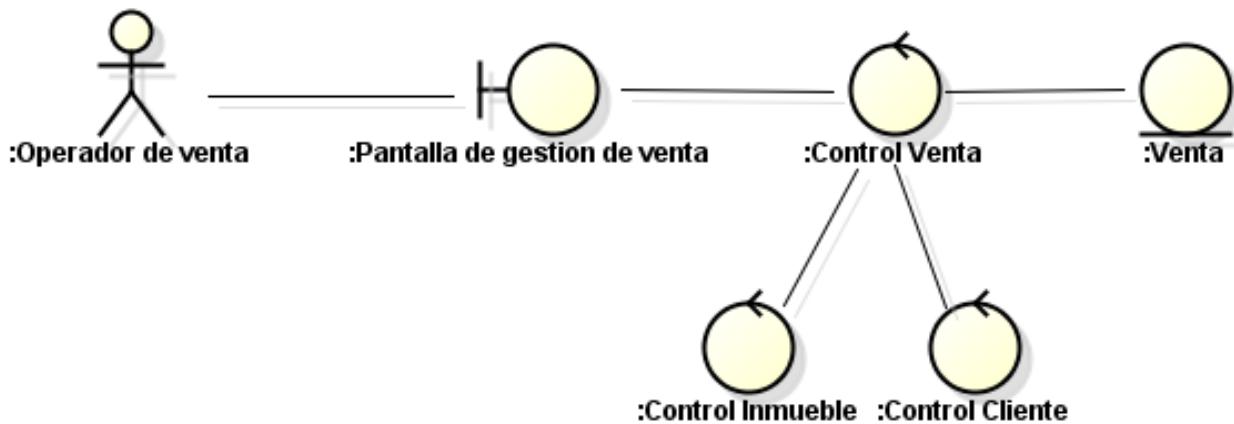


Figura 21: Diagrama de clases de análisis para caso de uso: Gestionar Venta

En la figura 22 se representa el diagrama de colaboración del caso de uso: Gestionar Venta.

Descripción: El operador de venta modifica datos correspondientes a la venta de un inmueble que existe en el sistema.

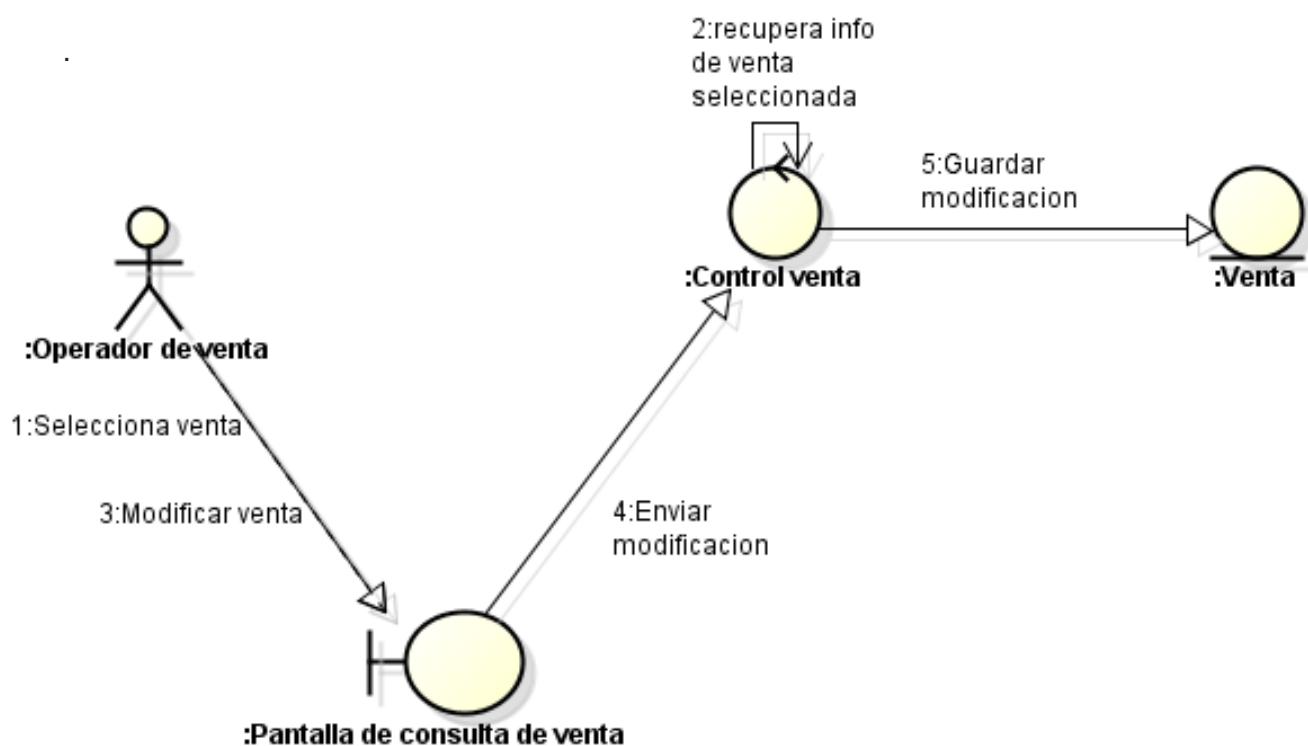


Figura 22: Diagrama de colaboración del caso de uso: Gestionar Venta (escenario 1)

Flujo de eventos: El operador de venta selecciona una venta (1). Luego de recuperar la información de esa venta, modifica los datos que desea y guarda los cambios (2, 3, 4 ,5).

## Escenario 2: Eliminar Venta

En la figura 23 se representa el diagrama de colaboración del caso de uso: Gestionar Venta.

Descripción: El operador de venta elimina una venta existente de un inmueble determinado.

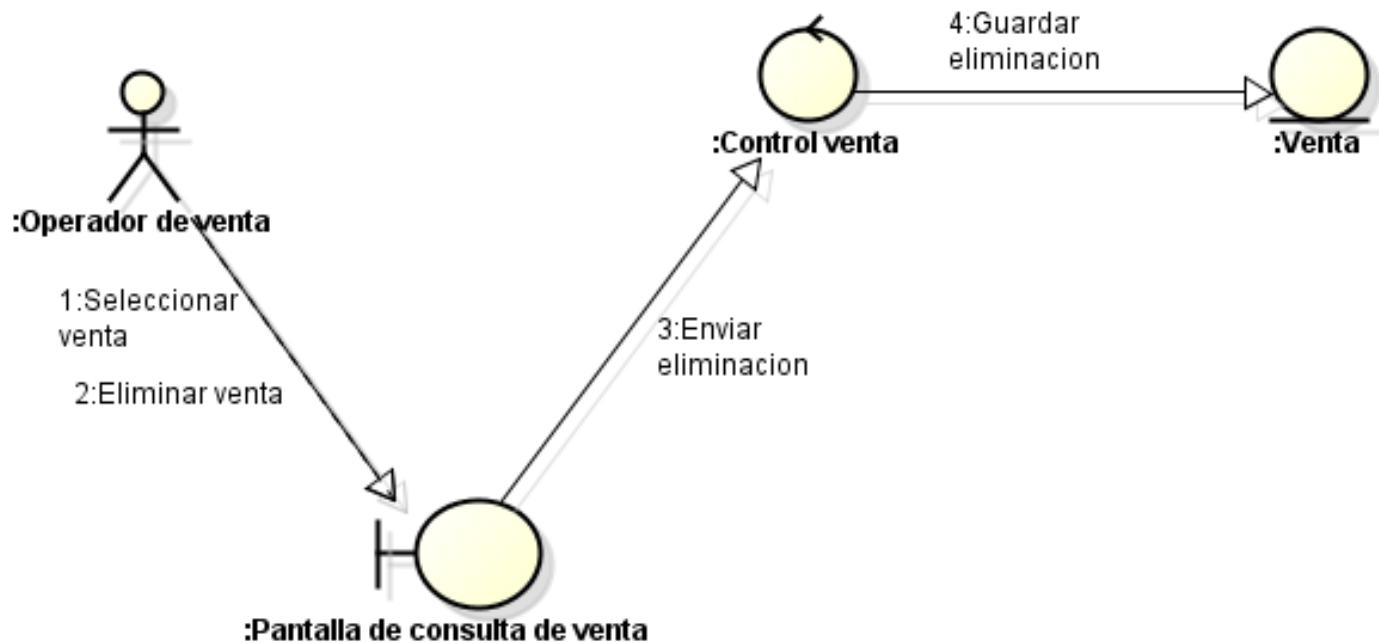


Figura 23: Diagrama de colaboración del caso de uso: Gestionar Venta (escenario 2)

Flujo de eventos: El operador de venta selecciona una venta (1). Luego elimina y guarda los cambios (2, 3, 4).

## 4.2 Etapa de Diseño

### 4.2.1 Introducción

En el diseño se modela el sistema y se encuentra su forma para que soporte todos los requisitos funcionales y no funcionales que le pertenecen. En lo que respecta a funcionalidad, como primer paso se desarrollará la identificación de Clases de Diseño a partir de Clases de Análisis, pudiendo identificar una correspondencia directa entre estas. Luego, se utilizarán Diagramas de Clases para reflejar las relaciones entre las clases de diseño participantes. Finalmente, la funcionalidad se verá reflejada en Diagramas de Secuencia, donde se podrán apreciar las secuencias de acciones de los casos de usos. Las secuencias de acciones de un caso de uso comienzan cuando un actor invoca al Caso de Uso mediante el envío de algún tipo de mensaje al sistema, es decir, un objeto de diseño llama a otro objeto mediante un mensaje, y de esta manera interactúan para llevar a cabo el Caso de Uso.

Los requisitos no funcionales que se deben tratar, suelen estar relacionados con aspectos como: Persistencia, distribución de objetos, gestión de transacciones, lenguaje de programación elegido y protocolos de transferencias de información. Esto es, definir la Arquitectura del Sistema.

Los propósitos del diseño son:

- Adquirir una comprensión global de los aspectos relacionados con los requisitos no funcionales relacionados con el lenguaje de programación, sistemas operativos, manejadores de base de datos, etc.
- Crear un punto de partida para actividades de implementación subsiguientes.
- Descomponer los trabajos de implementación en partes más manejables que pueden ser llevadas a cabo por diferentes equipos de desarrollo.
- Capturar antes las interfaces entre los subsistemas en el ciclo de vida del software.
- Crear una abstracción de la implementación del sistema.

El diseño es el centro de atención al final de la fase de elaboración y en el comienzo de las iteraciones de la fase de construcción. Esto contribuye a una arquitectura estable y sólida.

### 4.2.2 Modelo de diseño

El modelo de diseño es un modelo de objetos que describe la realización física de los casos de uso centrándose en como los requisitos funcionales y no funcionales tienen impacto en el sistema. También sirve como entrada fundamental a las actividades de implementación. En el modelo de diseño los casos de uso son realizados por las clases de diseño y sus objetos. Estos se representan por colaboraciones en el modelo de diseño y denota una realización de casos de uso-diseño.

Este modelo posee las siguientes características:

- Es un modelo físico porque es un plano de la implementación.
- Es muy formal y tiene varias copias

- Es dinámico, ya que es centrado en las secuencias.
- Debe ser mantenido durante todo el ciclo de vida del software
- Utiliza cualquier número de estereotipo físico sobre las clases

#### **4.2.3 Clase de diseño**

Una clase de diseño es una abstracción de una clase o construcción similar en la implementación del sistema, donde:

- El lenguaje utilizado para especificar una clase del diseño es lo mismo que el lenguaje de programación, y las operaciones, parámetros, atributos, tipos y otros son especificados utilizando la sintaxis del lenguaje de programación elegido
- Las relaciones entre clases de diseño, tienen un significado directo cuando la clase es implementada.
- Una clase de diseño puede posponer el manejo de algunos requisitos para las subsiguientes actividades de implementación, indicándolos como requisitos de implementación de la clase.
- Una clase de diseño generalmente aparece como un estereotipo que se corresponde con una construcción en el lenguaje de programación utilizado.
- Una clase de diseño puede proporcionar interfaces si tiene sentido hacerlo en el lenguaje de programación utilizado en la implementación.

#### **4.2.4 Realización de caso de uso-diseño**

Una realización de caso de uso-diseño es una elaboración en el modelo de diseño que describe como se ejecuta un caso de uso específico, en términos de clases de diseño y sus objetos.

Una realización de caso de uso-diseño posee una descripción del flujo de eventos textual, diagramas de clases de diseño y diagramas de interacción para un escenario concreto. Un diagrama de clases de diseño muestra las clases participantes, subsistemas y sus relaciones en la realización de un caso de uso concreto. Un diagrama de secuencia muestra las interacciones entre objetos de las clases de diseño mediante transferencia de mensajes entre estos, teniendo en cuenta el orden del tiempo en que se ejecutan.

Un flujo de eventos-diseño es una descripción textual que explica y contempla los diagramas de secuencias. Este flujo no debe hacer mención de ninguno de los atributos, operaciones y asociaciones de los objetos debido a que estos últimos cambian con bastante frecuencia.

#### **4.2.5 Subsistema de diseño**

Los subsistemas de diseño son una forma de organizar los artefactos del modelo de diseño en piezas más manejables. Un subsistema puede constar de clases de diseño, realizaciones de casos de uso-diseño, interfaces y otros subsistemas. Estos pueden proporcionar interfaces que

representan la funcionalidad que exportan en términos de operación. Un subsistema debe ser cohesivo y débilmente acoplado.

#### 4.2.6 Descripción de la arquitectura

La descripción de la arquitectura contiene una vista de la arquitectura del modelo de diseño que muestra sus artefactos relevantes para la arquitectura.

Los siguientes artefactos del modelo de diseño se consideran significativos para la arquitectura:

- La descomposición del modelo de diseño en subsistemas, sus interfaces, y las dependencias entre ellos.
- Clases de diseño.
- Realizaciones de casos de uso-diseño que describen alguna funcionalidad importante y crítica, que impliquen muchas clases de diseño.

En lo que respecta al lenguaje de programación elegido, se trabaja con el lenguaje Java, el cual sirve para crear todo tipo de aplicaciones (Locales, Internet, Cliente-Servidor). Es un lenguaje de objetos porque trabaja con objetos y utiliza el paradigma (o concepto) de Programación Orientada a Objetos, además es independiente de la plataforma porque puede ejecutarse sobre cualquier sistema operativo. Se lo reconoce principalmente por ser un lenguaje simple, distribuido, interpretado, sólido, seguro, de arquitectura neutral, portable, de alto desempeño, de multi-hilos y dinámico.

También se utiliza JSP (Java Server Pages) que es una tecnología que ayuda a crear páginas web dinámicas basadas en HTML utilizando el lenguaje java. La principal ventaja de JSP frente a otros lenguajes es que el lenguaje Java es un lenguaje de propósito general que excede el mundo web y que es apto para crear clases que manejen lógica de negocio y acceso a datos de una manera prolífica. Esto permite separar en niveles las aplicaciones web, dejando la parte encargada de generar el documento HTML en el archivo JSP. Otra ventaja es que JSP hereda la portabilidad de Java, y es posible ejecutar las aplicaciones en múltiples plataformas sin cambios.

Como es de esperarse algunos objetos del proceso deben ser persistentes; para conseguirlo se puede utilizar un sistema de gestión de bases de datos orientado a objetos, o bien uno relacional. Pero la mejor elección depende de cómo se deba acceder y actualizar los objetos y de la facilidad de implementación. Se ha optado por una base de datos relacional porque permite un procesamiento más eficiente de solicitudes y almacenamiento de información; además, almacenar información sin redundancia, pero que a la vez nos permitan recuperar información fácilmente.

Una base de datos de este tipo consiste en una colección de tablas, a cada una de las cuales se les asigna un nombre único. Hablamos de colección de tablas porque este tipo de Base de Datos almacena sus datos en diferentes tablas y logrando flexibilidad y velocidad.

Una fila de la tabla representa una relación entre un conjunto de valores, puesto que una tabla es una colección de dichas relaciones, hay una estrecha correspondencia entre el concepto

de tabla y el concepto matemático de relación, del cual toma su nombre el modelo de datos relacional.

Se entiende por persistencia como la acción de preservar la información de un objeto de forma permanente (guardar), pero a su vez también se refiere a poder recuperar la información del mismo (leer) para que pueda ser nuevamente utilizada. En este proyecto se utiliza como capa persistente, una persistencia en Java llamada JDO.

En cuanto al diseño, se utilizan dos patrones de diseño (Ver anexo II para más información): el patrón Singleton y el MVC (model, view, controller).

El patrón Singleton brinda la posibilidad de garantizar que solo hay una instancia de un objeto determinado y que se puede acceder a ella por cualquier otro objeto. Este patrón es muy útil en situaciones en donde se quieren compartir recursos únicos, particularmente, se utiliza este patrón para representar a la base de datos.

Por otro lado, el patrón MVC separa los datos del sistema, la interfaz de usuario, y la lógica de control en tres componentes distintos. La finalidad del patrón es mejorar la reusabilidad por medio del desacople entre la vista y el modelo. Los elementos del patrón son los siguientes:

- **Modelo:** Es el responsable de acceder a la capa de almacenamiento de datos. Lo ideal es que el modelo sea independiente del sistema de almacenamiento. También define la funcionalidad del sistema y lleva un registro de las vistas y controladores del sistema.
- **Controlador:** Se encarga de recibir los eventos de entrada (un clic, un cambio en algún campo de texto, etc.). También contiene reglas de gestión de eventos. Estas acciones pueden suponer peticiones al modelo o a las vistas.
- **Vista:** Se encarga de recibir datos del modelo y los muestra al usuario.

La utilización de este patrón posee las siguientes ventajas:

- Es posible tener diferentes vistas para un mismo modelo.
- Es posible construir nuevas vistas sin necesidad de modificar el modelo subyacente.
- Proporciona un mecanismo de configuración a componentes complejos mucho más tratable que el puramente basado en eventos (el modelo puede verse como una representación estructurada del estado de la interacción).

El Flujo que sigue el control de la implementación del MVC es de la siguiente manera:

- El usuario interactúa con la interfaz de usuario de alguna forma (por ejemplo, el usuario pulsa un botón).
- El controlador recibe (por parte de los objetos de la interfaz-vista) la notificación de la acción solicitada por el usuario. El controlador gestiona el evento que llega.
- El controlador accede al modelo, actualizándolo, modificándolo (si es necesario) de forma adecuada a la acción solicitada por el usuario (por ejemplo, el controlador actualiza el ingreso de un administrador al sistema).
- El controlador delega a los objetos de la vista la tarea de desplegar la interfaz de usuario. La vista obtiene sus datos del modelo para generar la interfaz apropiada para el usuario donde se refleja los cambios en el modelo (por ejemplo, muestra un cartel de usuario incorrecto en caso de que se hayan ingresado los datos incorrectamente).

- La interfaz de usuario espera nuevas interacciones del usuario, comenzando el ciclo nuevamente.

#### **4.2.7 Caso de uso: Gestionar Alquiler**

En la figura 24 se muestra la trazabilidad de clases de análisis a clases de diseño para caso de uso: Gestionar Alquiler

Una vez identificadas las clases de diseño, se deben recoger las mismas en un diagrama de clases asociado a la realización anterior. Se realizará este diagrama para mostrar las clases participantes y las relaciones que se utilizan en la realización del Caso de uso.

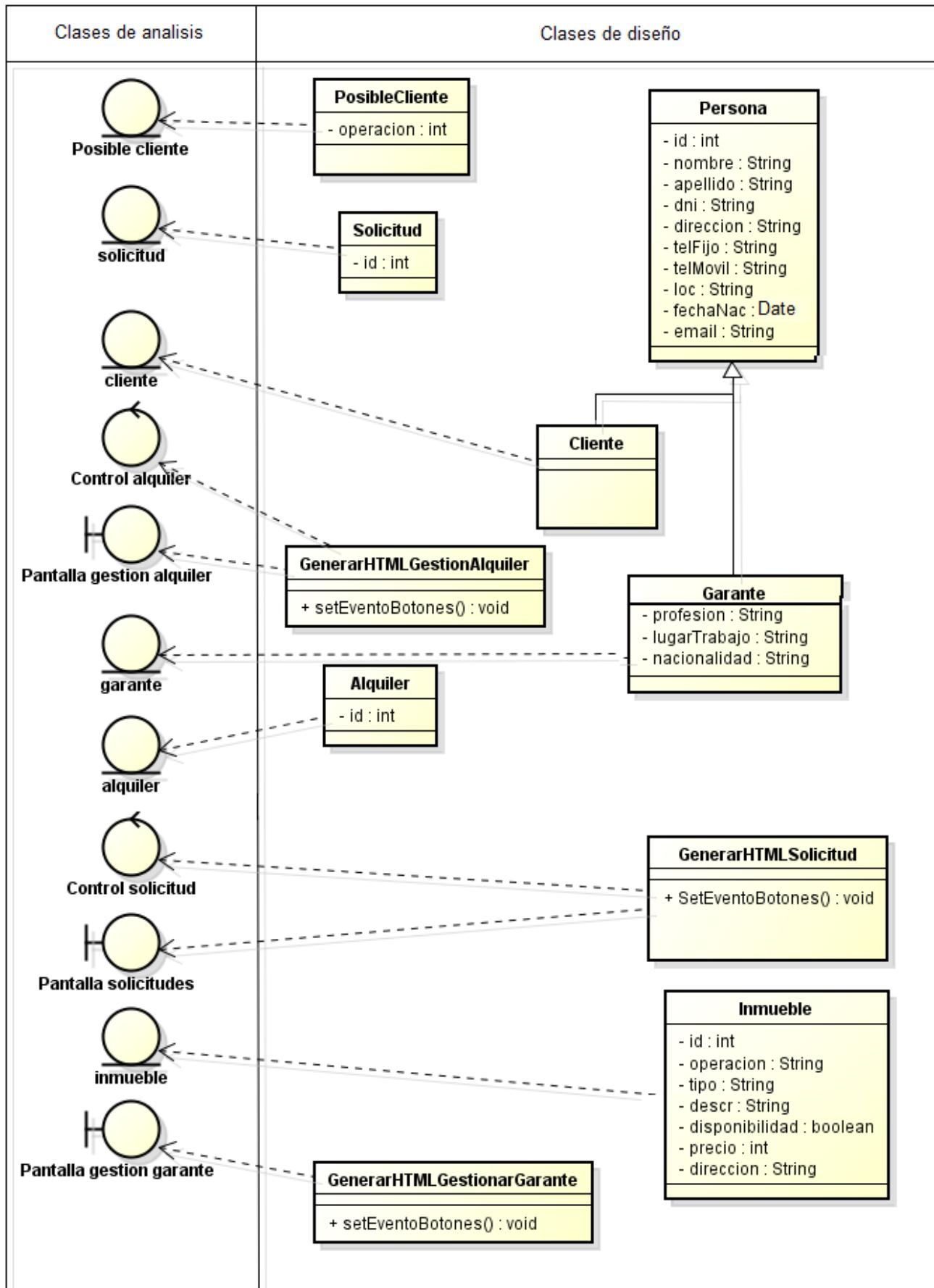


Figura 24: Trazabilidad de clases de análisis a clases de diseño para caso de uso: Gestionar Alquiler

En la figura 25 se muestra el diagrama de clases de diseño para caso de uso: Gestionar Alquiler:

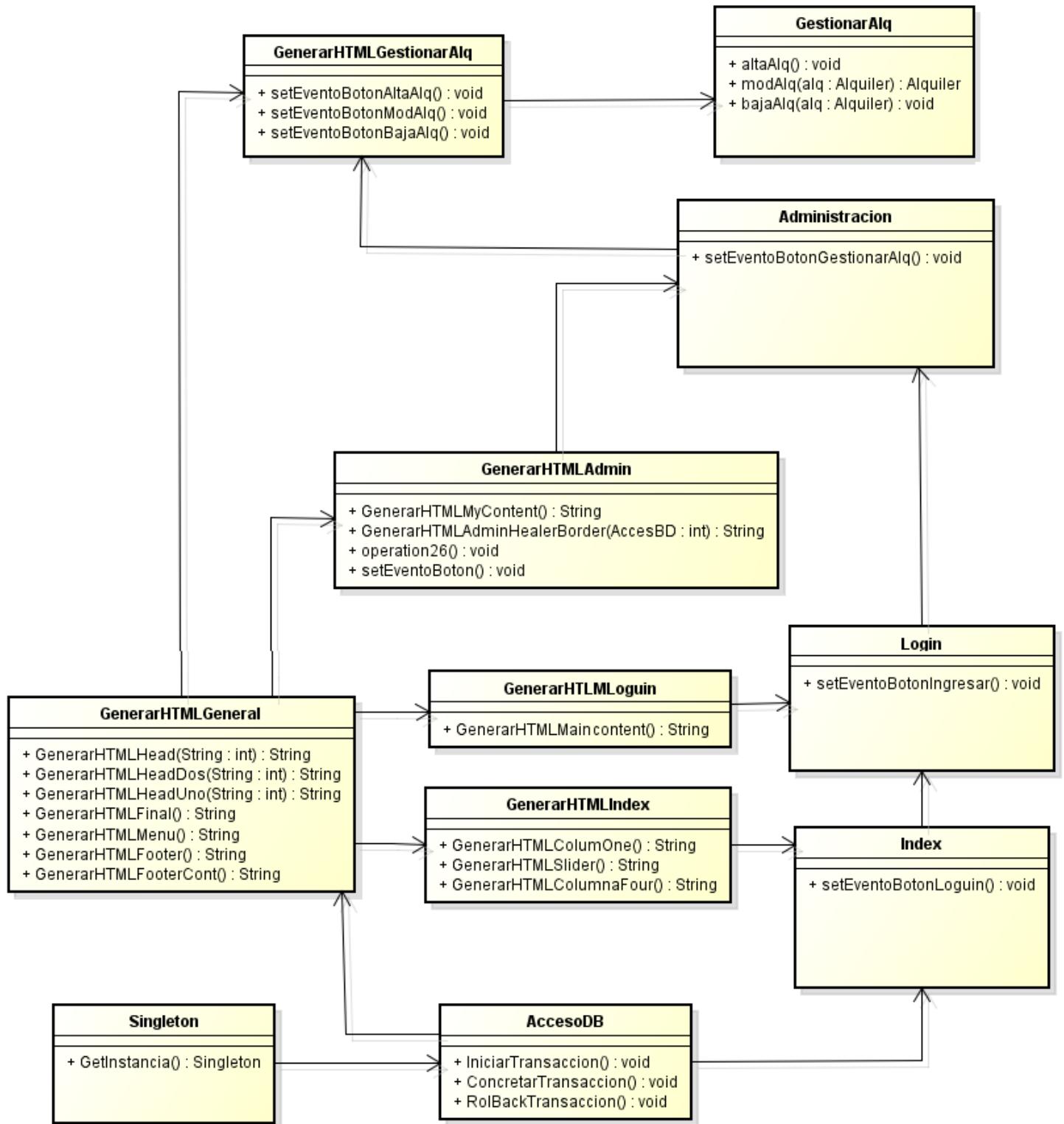


Figura 25: Diagrama de clases de diseño para caso de uso: Gestionar Alquiler

Su funcionamiento es el siguiente:

- GenerarHTMLGeneral: es la clase principal, la cual contiene la vista general del sistema (menú, logo, pie de página).
- GenerarHTMLAdmin: Clase que hereda de GenerarHTMLGeneral la cual contiene la vista de todas las funciones del administrador.
- GenerarHTMLLoguin: Clase que hereda de GenerarHTMLGeneral que contiene la vista de inicio de sesión.
- GenerarHTMLIndex: Clase que hereda de GenerarHTMLGeneral que contiene la vista de la página principal
- GenerarHTMLGestionarAlquiler: Clase que hereda de GenerarHTMLGeneral que contiene la vista de la gestión de los alquileres.
- Administracion: Clase que permite acceder a las funcionalidades del administrador.
- GestionarAlq: Clase que permite dar de alta, modificar o eliminar un alquiler.

A continuación se detallan escenarios concretos a través de diagramas de secuencia de los casos de uso Gestionar venta y luego de Gestionar alquiler.

### Escenario 1:

En la figura 26 se muestra el escenario 1 del diagrama de secuencia para el caso de uso: Gestionar Alquiler (escenario 1)

Descripción: El administrador con el nombre “Juan” (ya ingresado en el sistema) desea ingresar un alquiler con la fecha “12/04/2014”, el monto “2500”, el id del inmueble “2”, el id del cliente “10”, y el id del garante “15”.

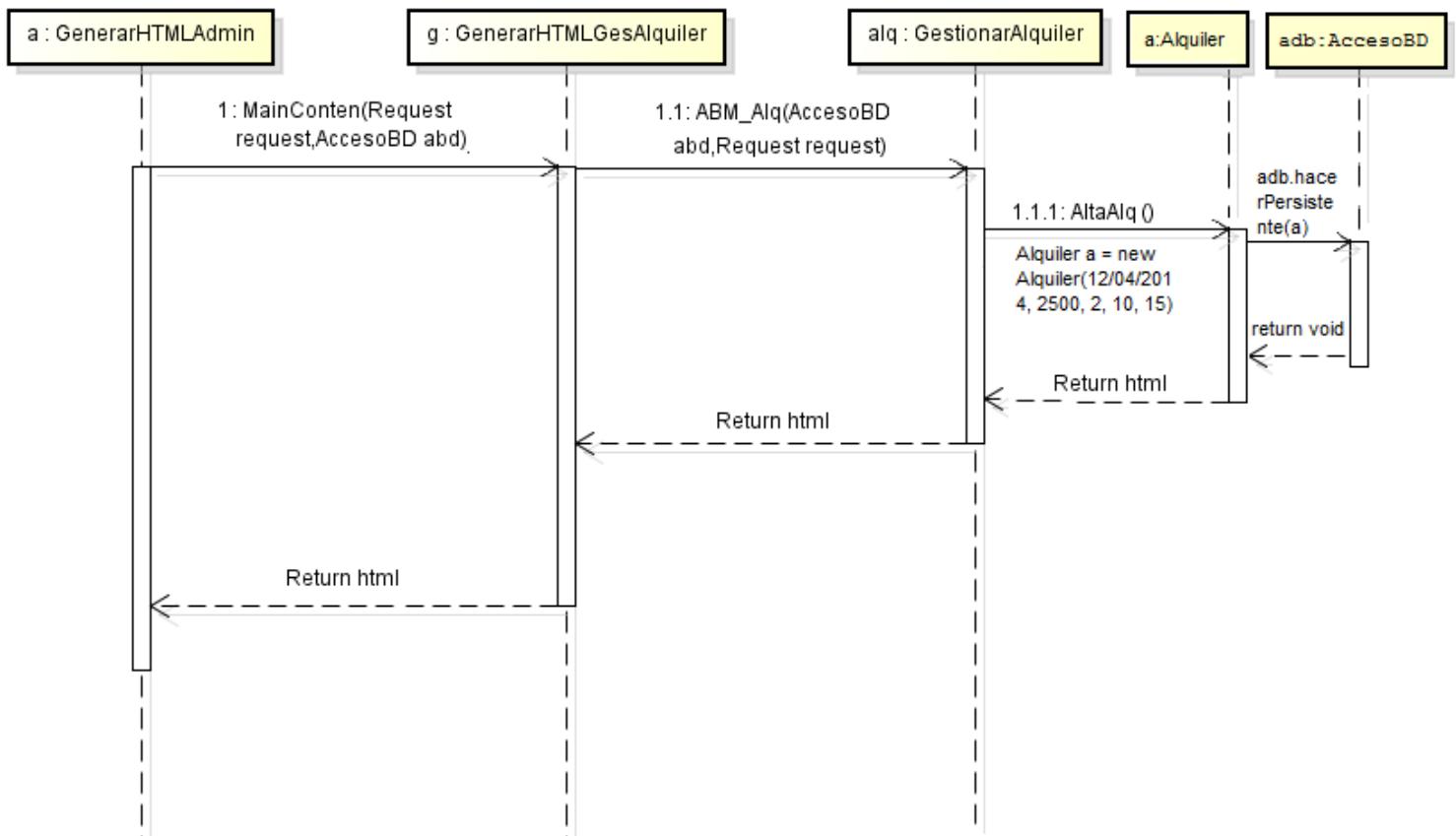


Figura 26: Diagrama de secuencia para caso de uso: Gestionar Alquiler (escenario 1)

## Escenario 2:

En la figura 27 se muestra el escenario 2 del diagrama de secuencia para el caso de uso: Gestionar Alquiler (escenario 2)

Descripción: El administrador con el nombre “Juan” desea eliminar el alquiler con el id = “22”

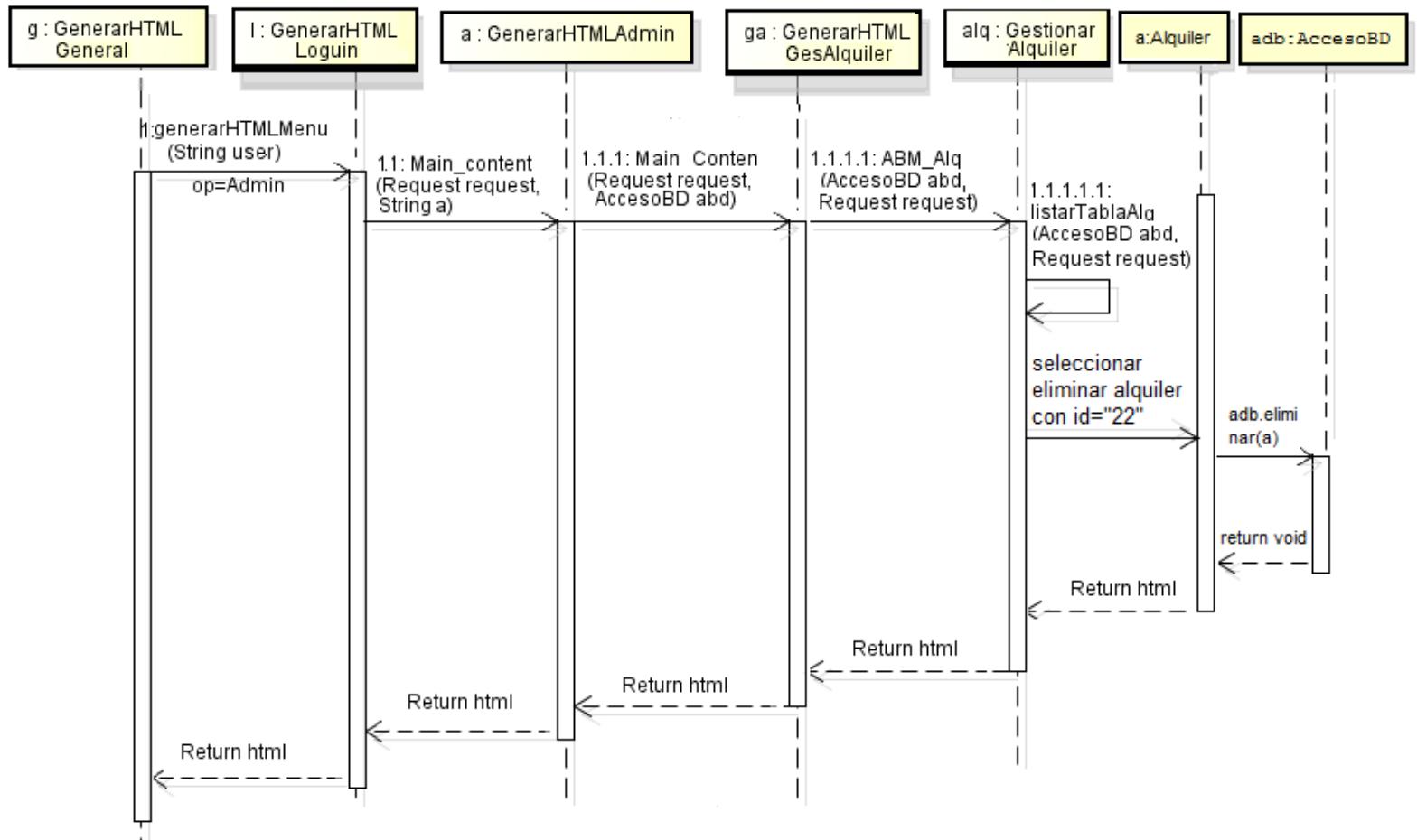


Figura 27: Diagrama de secuencia para caso de uso: Gestionar Alquiler (escenario 2)

### 4.2.8 Caso de uso: Gestionar venta

En la figura 28 se muestra la trazabilidad de clases de análisis a clases de diseño para caso de uso: Gestionar Venta.

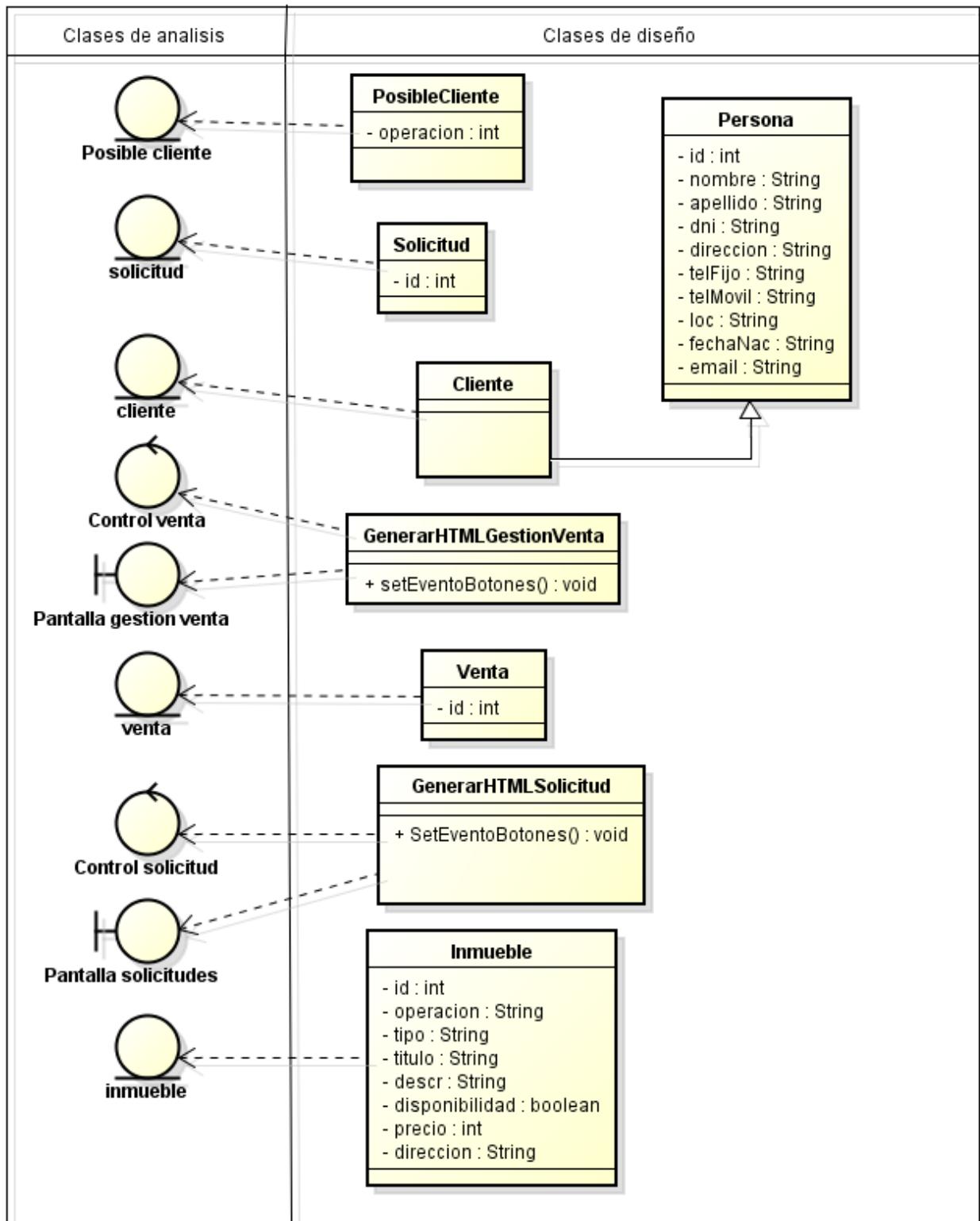


Figura 28: Trazabilidad de clases de análisis a clases de diseño para caso de uso: Gestionar Venta

En la figura 29 se muestra el diagrama de clases de diseño para caso de uso: Gestionar Venta.

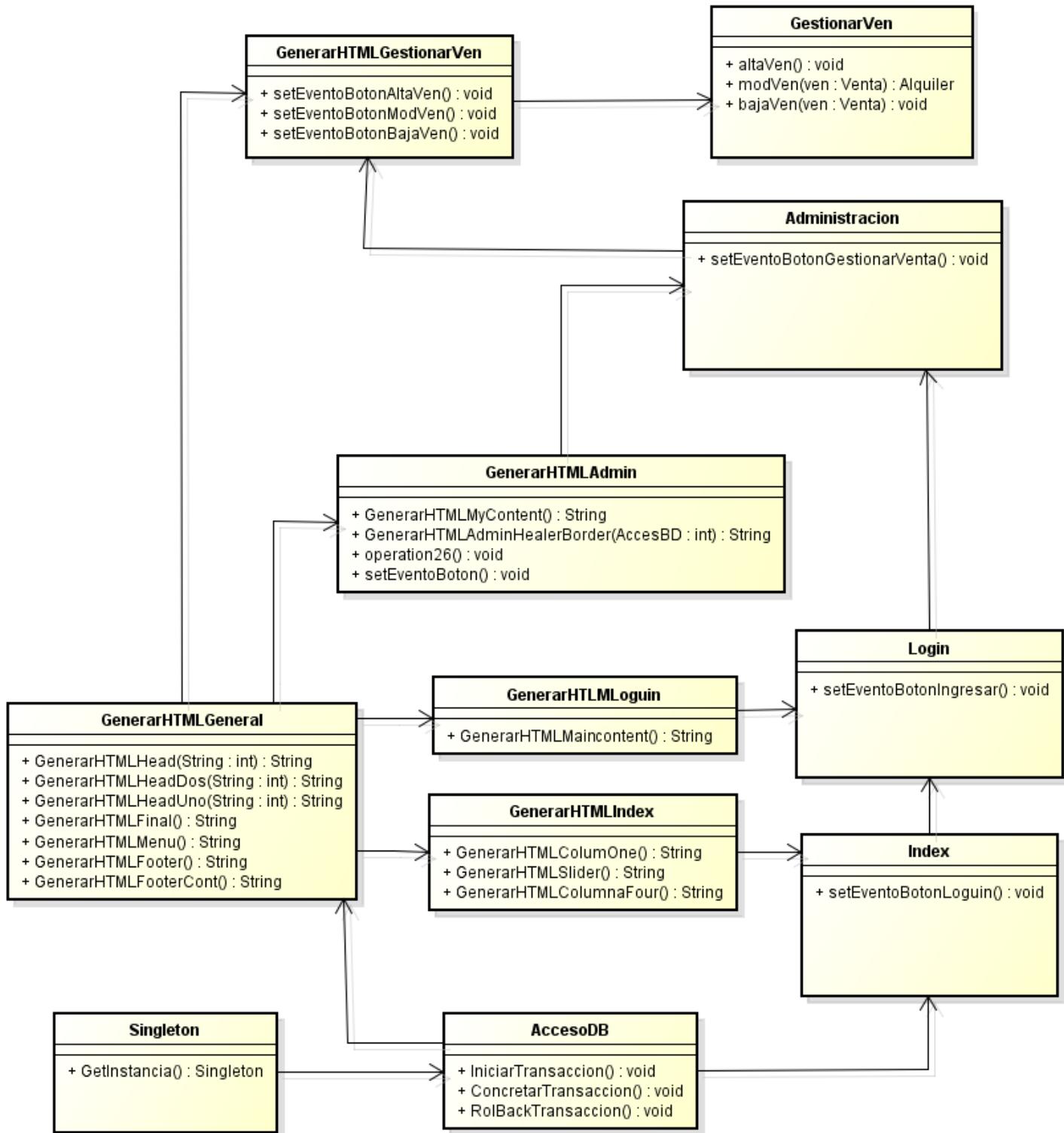


Figura 29: Diagrama de clases de diseño para caso de uso: Gestionar Venta

Su funcionamiento es el siguiente:

- GenerarHTMLGestionarVen: Clase que hereda de GenerarHTMLGeneral que contiene la vista de la gestión de las ventas.
- GestionarVen: Clase que permite dar de alta, modificar o eliminar un alquiler.

En cuanto a las clases GenerarHTMLGeneral, GenerarHTMLAdmin, GenerarHTMLLoguin, GenerarHTMLIndex y Administracion, su función es la misma que en el caso de uso: Gestionar Alquiler.

### Escenario 1:

En la figura 30 se muestra el escenario 1 del diagrama de secuencia para el caso de uso: Gestionar Venta.

Descripción: El administración “Juan” (ya ingresado en el sistema) desea listar todas las ventas.

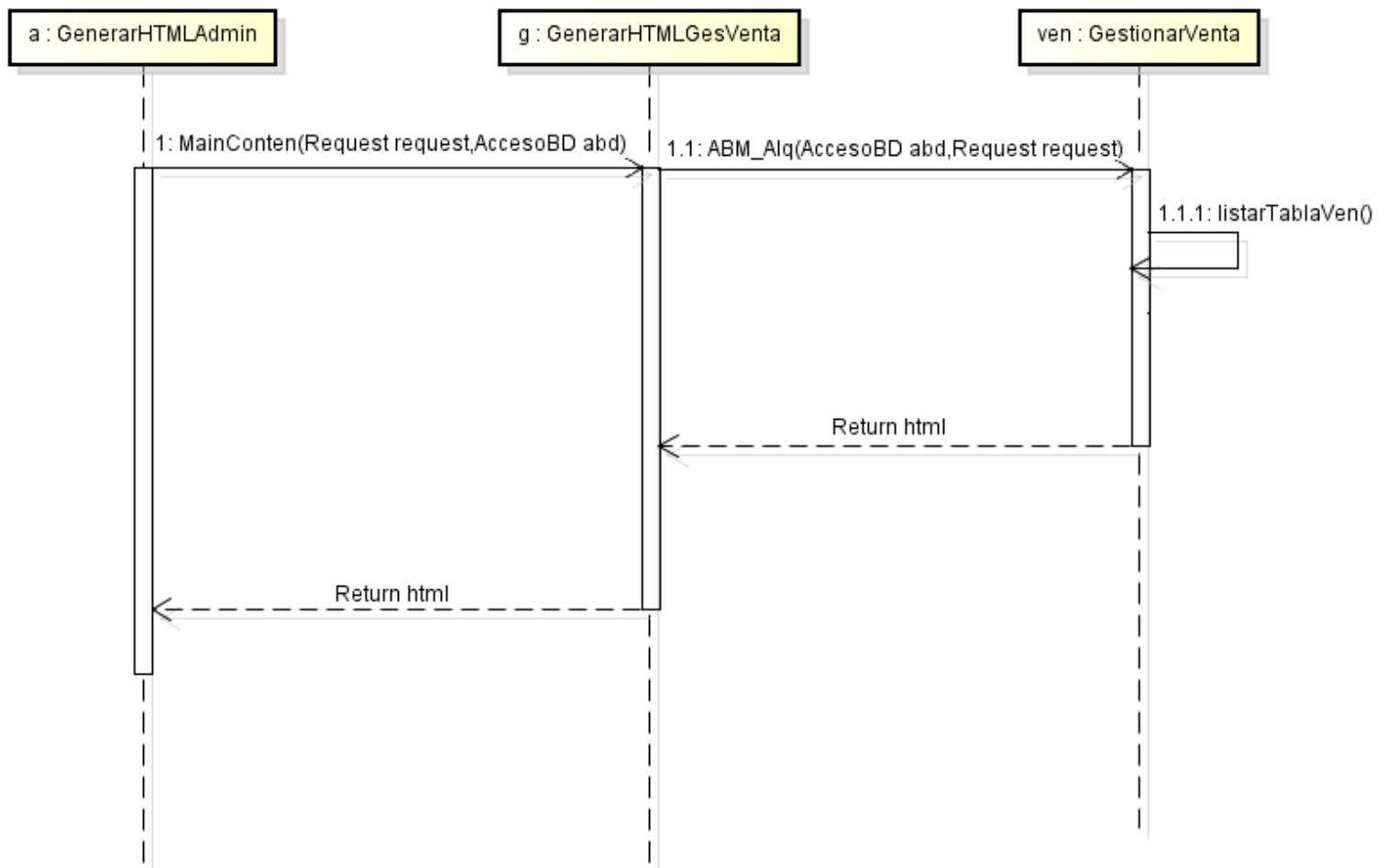


Figura 30: Diagrama de secuencia para caso de uso: Gestionar Venta (escenario 1)

## Escenario 2:

En la figura 31 se muestra el escenario 2 del diagrama de secuencia para el caso de uso: Gestionar Venta.

Descripción: El administrador “Juan” (ya ingresado en el sistema) desea eliminar la venta con el id número “10”.

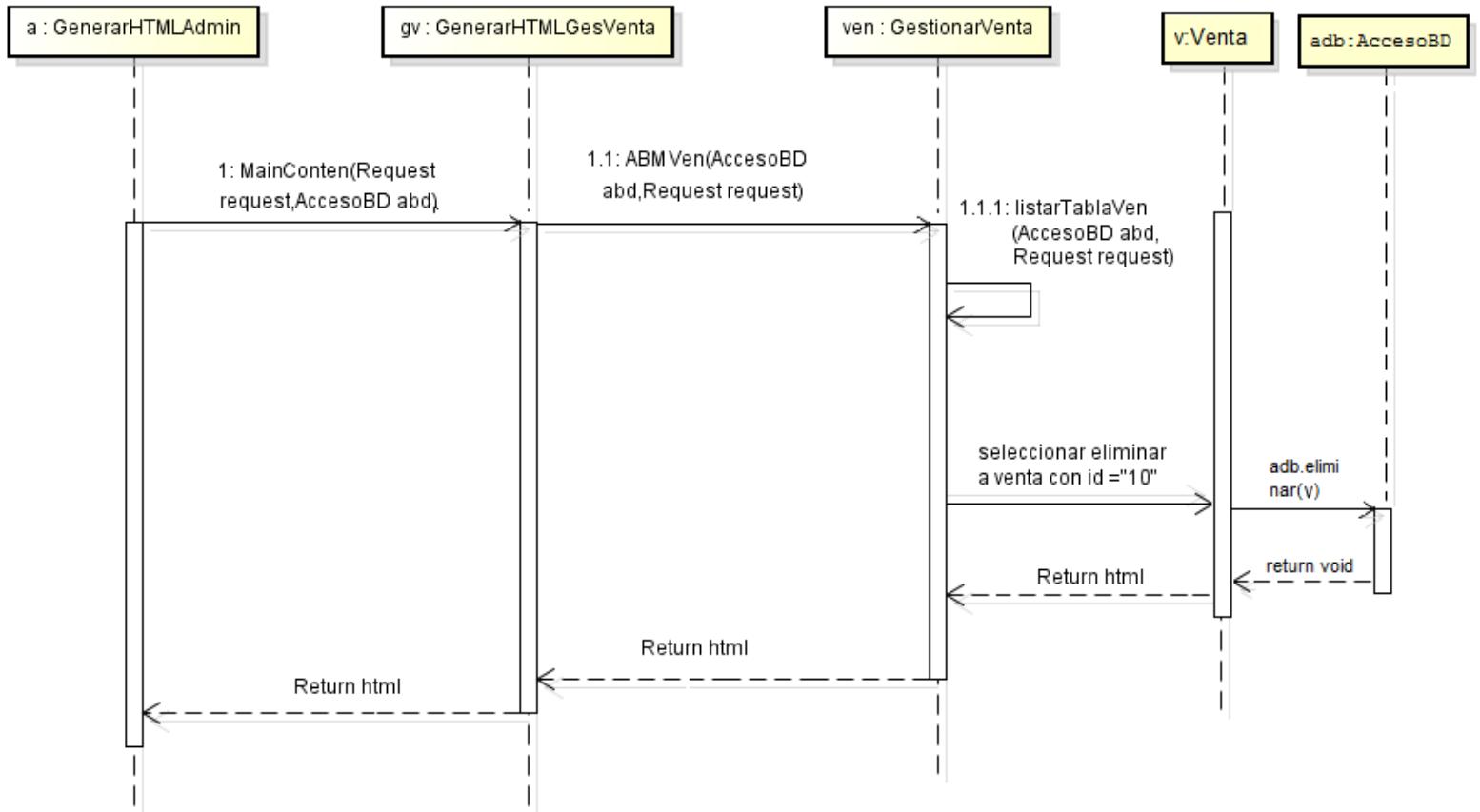


Figura 31: Diagrama de secuencia para caso de uso: Gestionar Venta (escenario 2)

### 4.2.9 Modelo de persistencia

El propósito del diseño de la base de datos es asegurar que los datos se almacenan efectiva y consistentemente. El proceso de diseño de una base de datos se guía por algunos principios. El primero de ellos es que se debe evitar la información duplicada o, lo que es lo mismo, los datos redundantes, porque malgastan el espacio y aumentan la probabilidad de que se produzcan errores e incoherencias. El segundo principio es que es importante que la información sea correcta y completa. Si la base de datos contiene información incorrecta, los informes que recogen información de la base de datos contendrán también información incorrecta.

Un buen diseño de base de datos es, por lo tanto, aquél que:

- Divide la información en tablas basadas en temas para reducir los datos redundantes.
- Proporcionar el acceso a la información necesaria para reunir la información de las tablas cuando así se precise.
- Ayuda a garantizar la exactitud e integridad de la información.
- Satisface las necesidades de procesamiento de los datos y de generación de informes.

En la figura 32 se muestra el diagrama de modelo de clases persistentes, se puede observar la información persistente que describe como se relacionan las clases participantes en la misma.

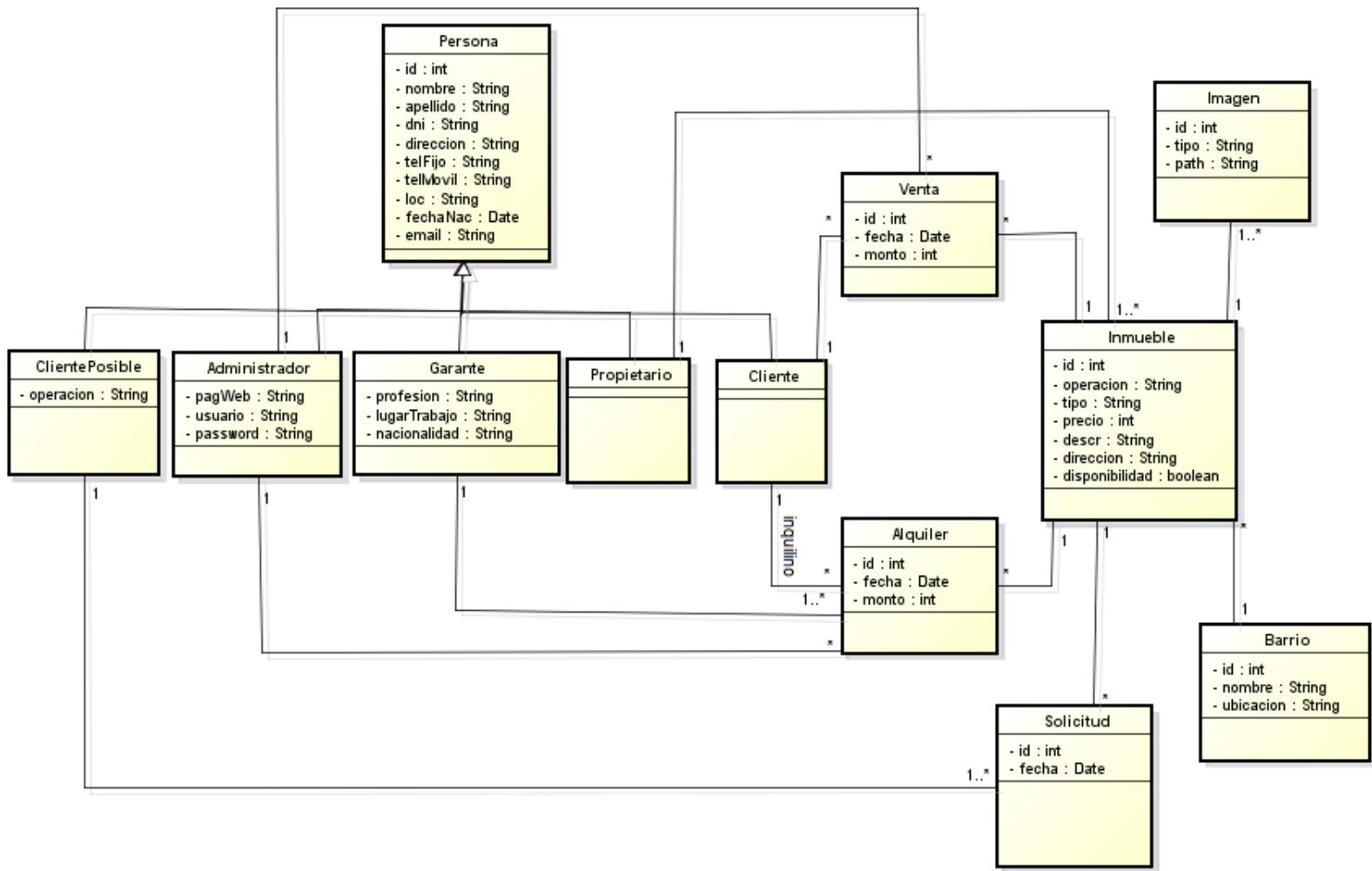


Figura 32: Modelo de clases persistentes

## **4.3 Implementación**

### **4.3.1 Introducción**

En esta etapa se procede a escribir el código que va a dar lugar al sistema. Esta etapa es la más costosa ya que requiere de tareas de organización, utilización de herramientas y recursos adecuadamente. También se debe contar con la capacidad de resolver problemas y errores inesperados que pueden llegar a ocurrir durante este proceso. Cabe destacar que es la etapa que lleva más tiempo ya que se recrea todo lo asentado anteriormente.

El propósito de la implementación es desarrollar la arquitectura y el sistema como un todo. Es decir:

- Planificar las integraciones de sistema necesarias en cada iteración, de forma incremental, o sea con una sucesión de pasos pequeños y manejables.
- Distribuir el sistema asignando componentes ejecutables a nodos en el diagrama de despliegue, basándonos en las clases activas encontradas durante el diseño.
- Implementar las clases y subsistemas encontrados durante el diseño.
- Probar los componentes individualmente, y a continuación integrarlos compilándolos y enlazándolos en uno o más ejecutables, antes de ser enviados para ser integrados y llevar a cabo las comprobaciones del sistema.

### **4.3.2 Modelo de Implementación**

El modelo de implementación describe como los elementos del modelo de diseño, como las clases, se implementan en términos de componentes, como se organizan los componentes y como dependen los componentes unos de los otros.

Un componente es un empaquetamiento físico de los elementos de un modelo. Tienen relaciones de traza con los elementos del modelo que implementan.

Los subsistemas de implementación proporcionan una forma de organizar los artefactos en cada nueva versión del sistema, simplificando así los problemas de integración.

Para la generación de código se utiliza Java 7 dentro del entorno de desarrollo NetBeans el cual es un entorno de desarrollo integrado libre, hecho principalmente para el lenguaje de programación Java.

Para la manipulación (introducir, acceder y procesar) de datos se utiliza un gestor de base de datos llamado Workbench, esta elección se debe a que Workbench es un gestor de base de datos relacionales completo. Además se utiliza como capa persistente, una persistencia en Java llamada JDO. Esta capa persistente está implementada en Java por la librería jpox-java5-1.2.3.jar, la cual se utiliza para el desarrollo del proyecto.

Cabe destacar además que el sistema está testeado bajo el navegador Google Chrome.

En cuanto a las bibliotecas que se utilizan durante la implementación se encuentran:

- MySQL JDBC Driver – mysql – connector-java-5.1.23-bin.jar: La cual se encarga de la conexión del sistema con la base de datos.
- Cos.jar: Incluye las clases MultipartRequest y MultipartParser las cuales son muy útiles para los desarrolladores de sistemas cliente/servidor ya que permiten manejar tanto la carga como la descarga de archivos, administrar conexiones de socket, parsear parámetros, entre otras utilidades.
- jdo2-api-2.1.jar: Es una librería estandar de interfaces basada en el modelo Java de abstracción estándar de persistencia.
- jpxo-jav5-1.2.3.jar: Esta librería es una implementación de los estándares JDO 1.0 y 2.0 de persistencia en Java.
- Log4j-1.2.8.jar: Es una biblioteca de registro para Java desarrollada por Apache. Con esta librería es posible habilitar el registro en tiempo de ejecución sin modificar el código del sistema. Este registro proporciona al programador detalles de errores del sistema.
- Uploadbean.jar: Permite cargar archivos desde el navegador. Puede almacenar los archivos subidos en una carpeta, un archivo ZIP, una base de datos o en la Memoria.

#### 4.3.3 Modelo de Despliegue

Un diagrama de despliegue es un diagrama que muestra la configuración de los nodos que participan en la ejecución y de los componentes que residen en ellos. Estos diagramas contienen generalmente nodos y relaciones de dependencia y asociación. Además pueden contener notas y restricciones, componentes residentes en algún nodo y paquetes o subsistemas.

Los diagramas de despliegue se utilizan para modelar la vista estática de despliegue de un sistema. La misma cubre la distribución, entrega e instanciación de las partes que configuran el sistema físico.

En la figura 33 se muestra el diagrama de despliegue del sistema.

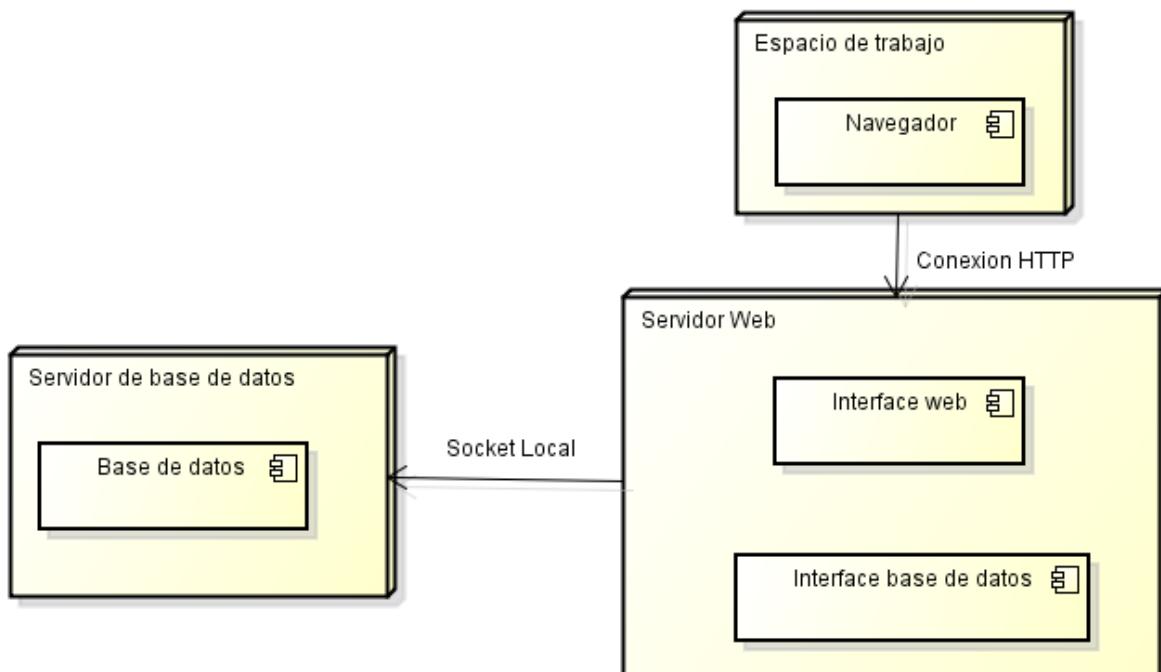


Figura 33: Diagrama de Despliegue

#### 4.3.4 Modelo de Componentes

Un diagrama de componentes muestra un conjunto de componentes y sus relaciones. Nos provee la organización y las dependencias entre un grupo de componentes. Estos diagramas se utilizan para modelar la vista de implementación estática de un sistema. Esto implica modelar los elementos físicos que residen en un nodo, tales como ejecutables, bibliotecas, archivos y documentos.

Los diagrama de componentes no solo son importantes para visualizar, especificar y documentar sistemas basados en componentes, sino también para construir sistemas ejecutables mediante ingeniería directa e inversa.

En la figura 34 se muestra el diagrama de componentes del sistema.

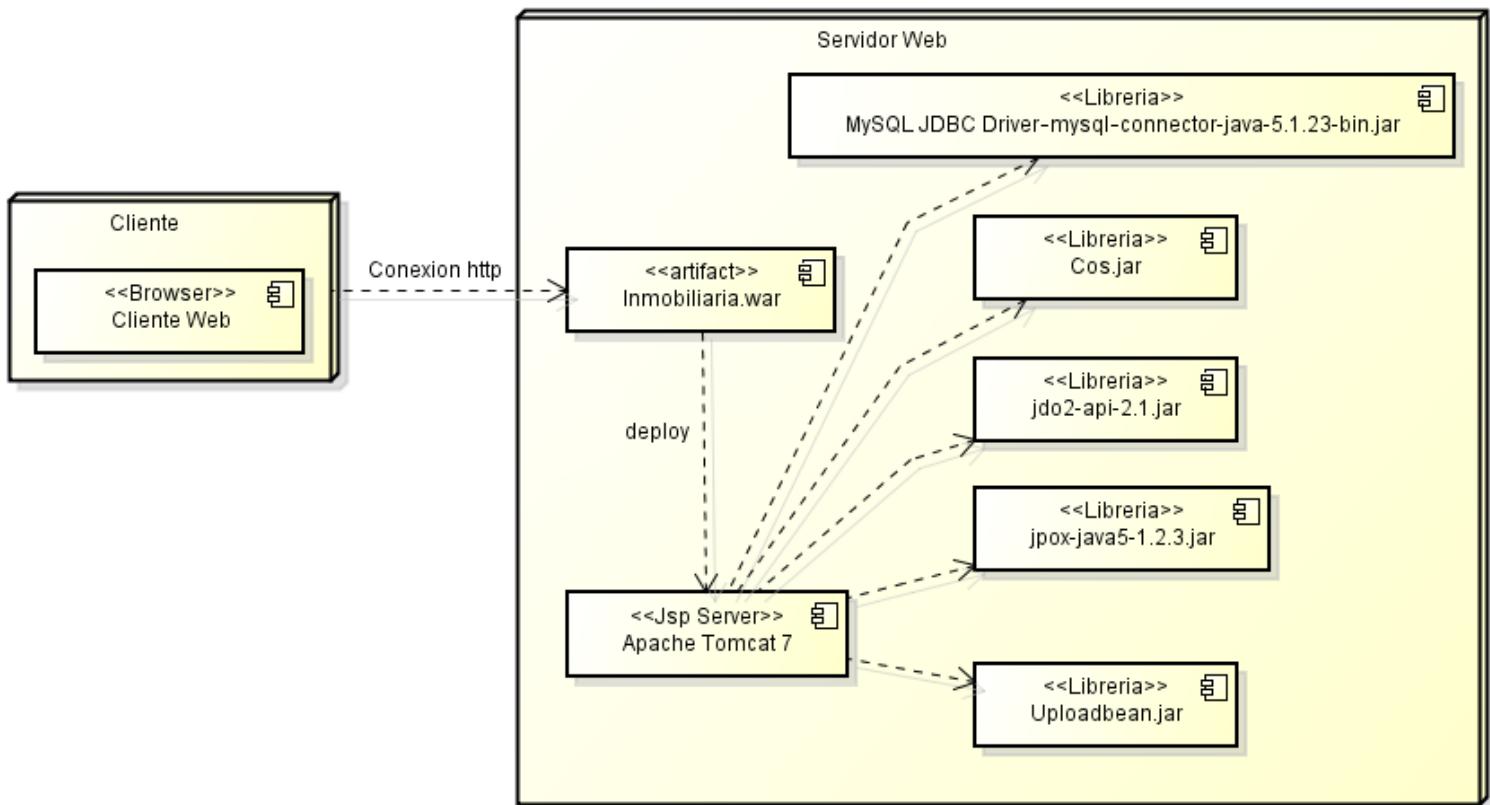


Figura 34: Diagrama de Componentes

## **4.4 Prueba**

### **4.4.1 Introducción**

En esta etapa se determina cómo operan los componentes de un sistema en situaciones representativas y verificar si su comportamiento es el esperado. En este proyecto, se procederán a efectuar pruebas de unidad.

La prueba de unidades consiste en probar módulos individuales, se hará enfoque a las pruebas de Caja Negra y Caja Blanca.

### **4.4.2 Caja Blanca**

Las pruebas de caja blanca se centran en los detalles procedimentales del software, por lo que su diseño está fuertemente ligado al código fuente. El testeador escoge distintos valores de entrada para examinar cada uno de los posibles flujos de ejecución del programa y cerciorarse de que se devuelven los valores de salida adecuados.

Pruebas de Caja Blanca o Estructural:

- Criterio de Cobertura de Sentencia: Ejecuta el programa y se asegura de que todas sus sentencias son ejecutadas al menos un vez.
- Criterio de Cobertura de Arco: Describe el programa con un grafo del flujo de control del programa, y asegura que se recorren todos los arcos en dicho grafo.
- Criterio de Cobertura de Condición: Divide las expresiones booleanas complejas en sus componentes e intenta cubrir todos los valores posibles de cada uno de ellos.
- Criterio de Cobertura de Camino: Asegura que todos los caminos del grafo de flujo de control del programa son atravesados al menos una vez.

En la figura 35 se detalla la prueba de caja blanca a través de un grafo de flujo para el caso de uso: Alta Propietario.

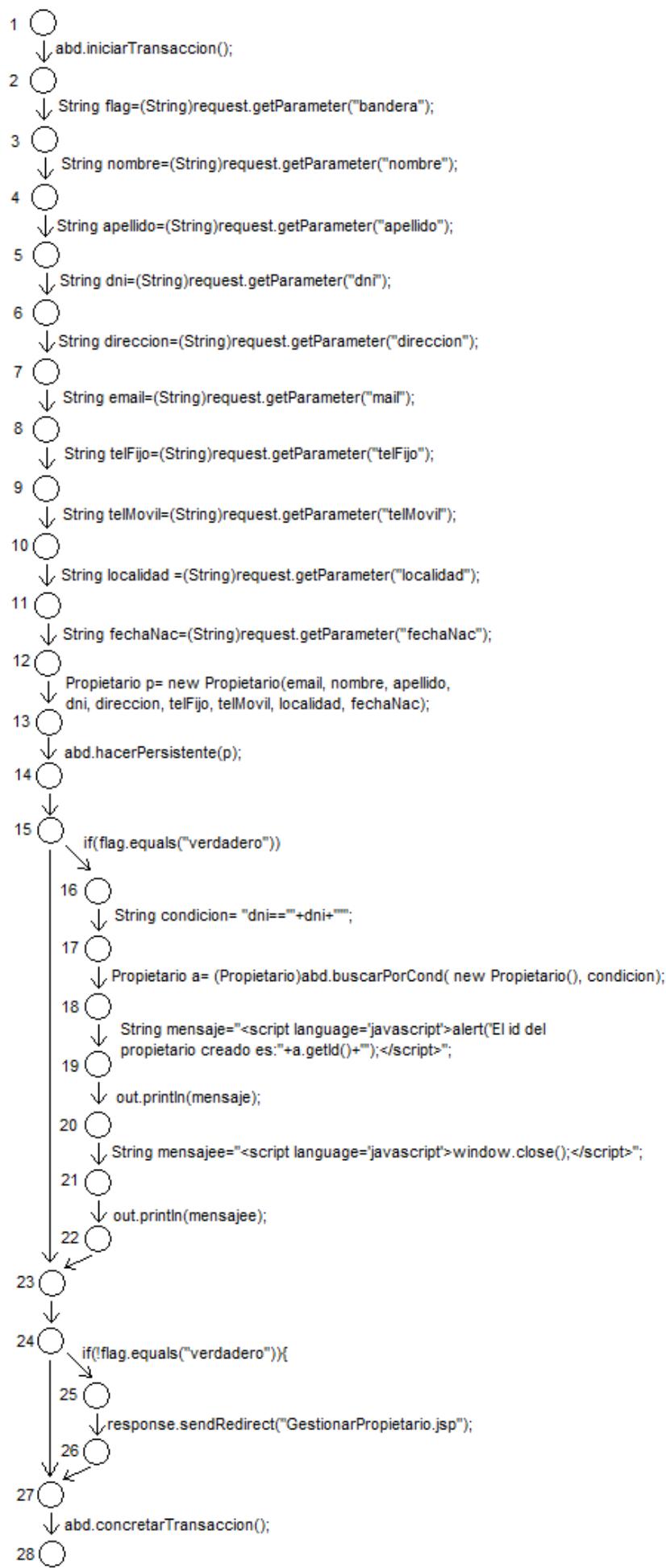


Figura 35: Grafo de flujo para criterio de cobertura de arco de caso de uso: Alta Propietario

Conjuntos de casos de prueba:

```
T1 = {<flag = "verdadero", nombre = "Juan", apellido="Gomez",dni ="34554221" dirección = "San Martin 240" email= "juanGomez@gmail.com" telFijo="46775332" telMovil = "154321544" localidad= "Rio Cuarto" fechaNac= "12/10/1984>,<flag = "null" , nombre = "Juan", apellido="Gomez",dni ="34554221" dirección = "San Martin 240" email= "juanGomez@gmail.com" telFijo="46775332" telMovil = "154321544" localidad= "Rio Cuarto" fechaNac= "12/10/1984>} no encuentra el error
```

```
T2 = {<flag = "verdadero", nombre = "Juan", apellido="Gomez",dni ="hola" dirección = "San Martin 240" email= "juanGomez@gmail.com" telFijo="46775332" telMovil = "154321544" localidad= "Rio Cuarto" fechaNac= "12/10/1984>,<flag = "null" , nombre = "Juan", apellido="Gomez",dni ="34554221" dirección = "San Martin 240" email= "juanGomezArroba@gmail.com" telFijo="46775332" telMovil = "154321544" localidad= "Rio Cuarto" fechaNac= "12/10/1984>} encuentra el error!
```

#### 4.4.3 Caja Negra

Se denomina caja negra a aquel elemento que es estudiado desde el punto de vista de las entradas que recibe y las salidas o respuestas que produce, sin tener en cuenta su funcionamiento interno. Por lo tanto, de una caja negra deben estar muy bien definidas sus entradas y salidas, es decir, su interfaz; en cambio, no se precisa definir ni conocer los detalles internos de su funcionamiento.

Prueba de Caja Negra o Funcional:

- Análisis del Valor Límite: Define un rango de valores posibles para una variable y realiza la prueba focalizando en los valores de los extremos del rango.
- Prueba de Clases de Equivalencia: Permite particionar un conjunto, donde dicha partición tiene una colección de subconjuntos mutuamente disjuntos y su unión forma el conjunto entero.
- Pruebas basadas en Tablas de Decisión: Se utilizan para representar y analizar la complejidad de las relaciones lógicas. Describen situaciones en las cuales un número de combinaciones de acciones son realizadas bajo la variación de un conjunto de condiciones

En la figura 36 se detalla la prueba de clase de equivalencia (caja negra) través del caso de uso: Alta Propietario.

Entrada de datos válida para caso de uso: Alta Propietario

num = {0,1,...,9}+

letra = {a,b,...z,A,B,...,Z}+

nombre	apellid o	dni	direccion	mail	telFijo	telMovil	localid ad	fechaNac	Salida esperada
letra	letra	letra	letra\num	letra	num	num	letra	num	Error
letra	letra	num	letra\num	letra	num	num	letra	num\nletra	Valida
letra	letra	num	letra\num	num	num	letra	letra	letra	Error
letra	letra	num	num	letra	letra	num	letra	letra	Error

Figura 36: Prueba de clases de equivalencia para caso de uso: Alta Propietario

## 5 Segunda Iteración

En esta etapa comienza a desarrollarse la segunda iteración la cual consiste en el análisis, diseño, implementación y prueba de los siguientes casos de uso:

- Gestionar posible cliente
  - Ingresar posible cliente
  - Eliminar posible cliente
- Gestionar cliente
  - Ingresar cliente
  - Modificar cliente
  - Eliminar cliente
  - Consultar cliente
- Listar operaciones:
  - Por disponibilidad
  - Por ubicación
  - Por precio
  - Por prop. Alquiladas
  - Por prop. Vendidas
  - Por prop. Para alquilar
  - Por prop. Para vender
  - Por ventas de administradores
  - Por Alquileres de administradores
  - De clientes
- Verificar administrador
- Gestionar solicitud
  - Ingresar solicitud
  - Eliminar solicitud

### 5.1 Análisis

#### 5.1.1 Introducción

En este análisis, se hará enfoque en los casos de uso de la segunda iteración: Gestionar cliente y Listados. En lo que respecta al Modelo de análisis, clases de análisis, realización de caso de uso-análisis, paquetes de análisis y descripción de la arquitectura, se emplearan de la misma manera que la propuesta en la iteración 1.

#### 5.1.2 Caso de uso: Gestionar Cliente

En la figura 37 se muestra la trazabilidad de casos de uso del sistema a clases de análisis para caso de uso: Gestionar Cliente

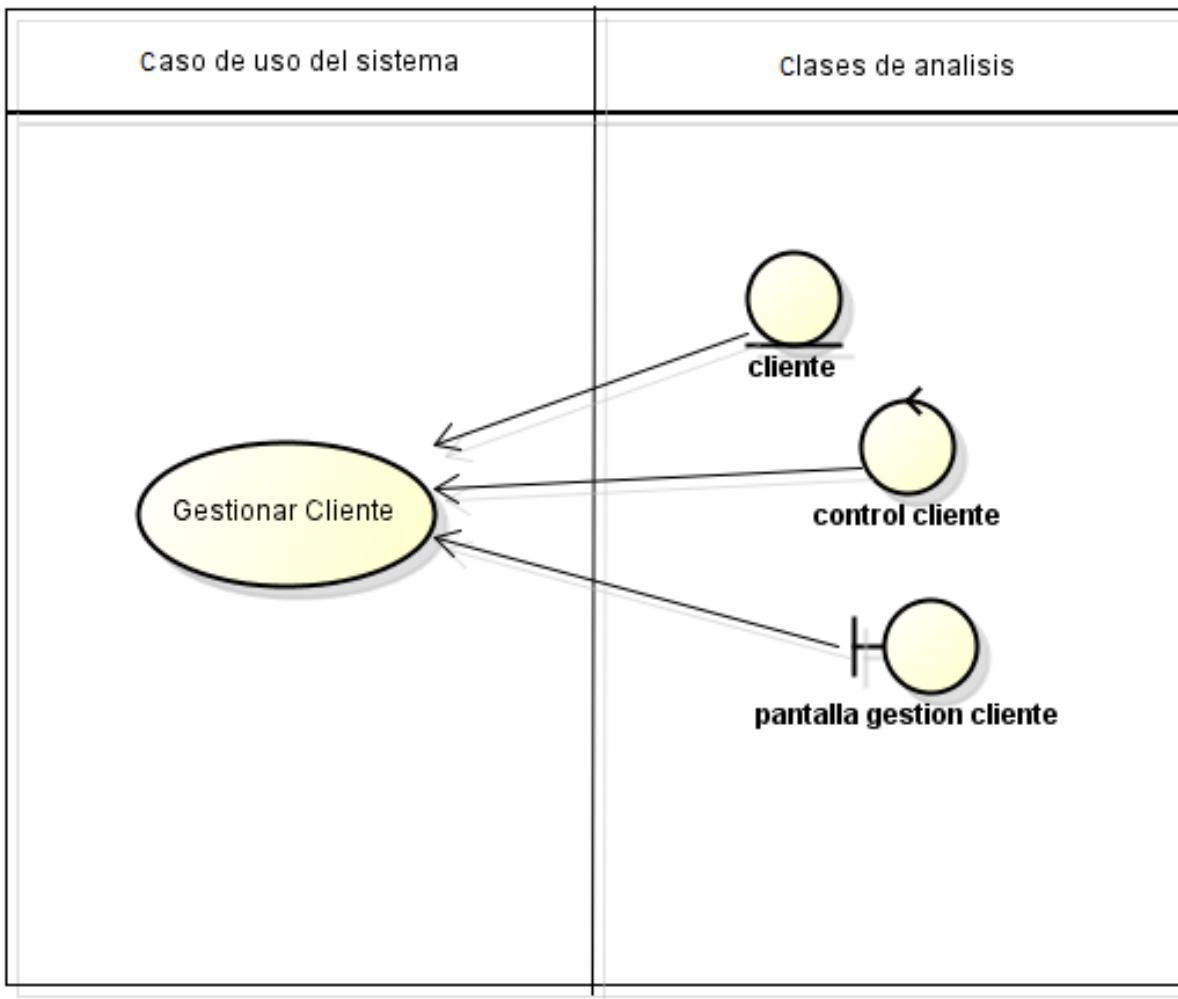


Figura 37: Trazabilidad de casos de uso del sistema a clases de análisis para caso de uso: Gestionar Cliente

En la figura 38 se muestra el diagrama de clases de análisis para el caso de uso: Gestionar Cliente.

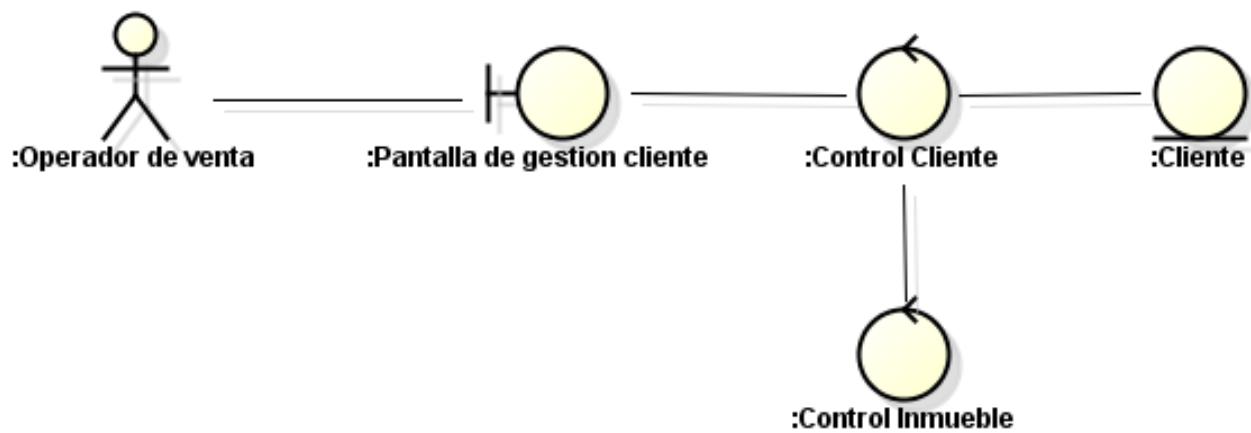


Figura 38: Diagrama de clases de análisis para caso de uso: Gestionar Cliente

A continuación se detallan escenarios concretos a través de diagramas de colaboración del caso de uso: Gestionar Cliente.

### Escenario 1: Ingresar Cliente

En la figura 39 se muestra el escenario 1 del diagrama de colaboración del caso de uso: Gestionar Cliente

Descripción: El operador de venta ingresa y guarda en el sistema datos referentes a un nuevo cliente.

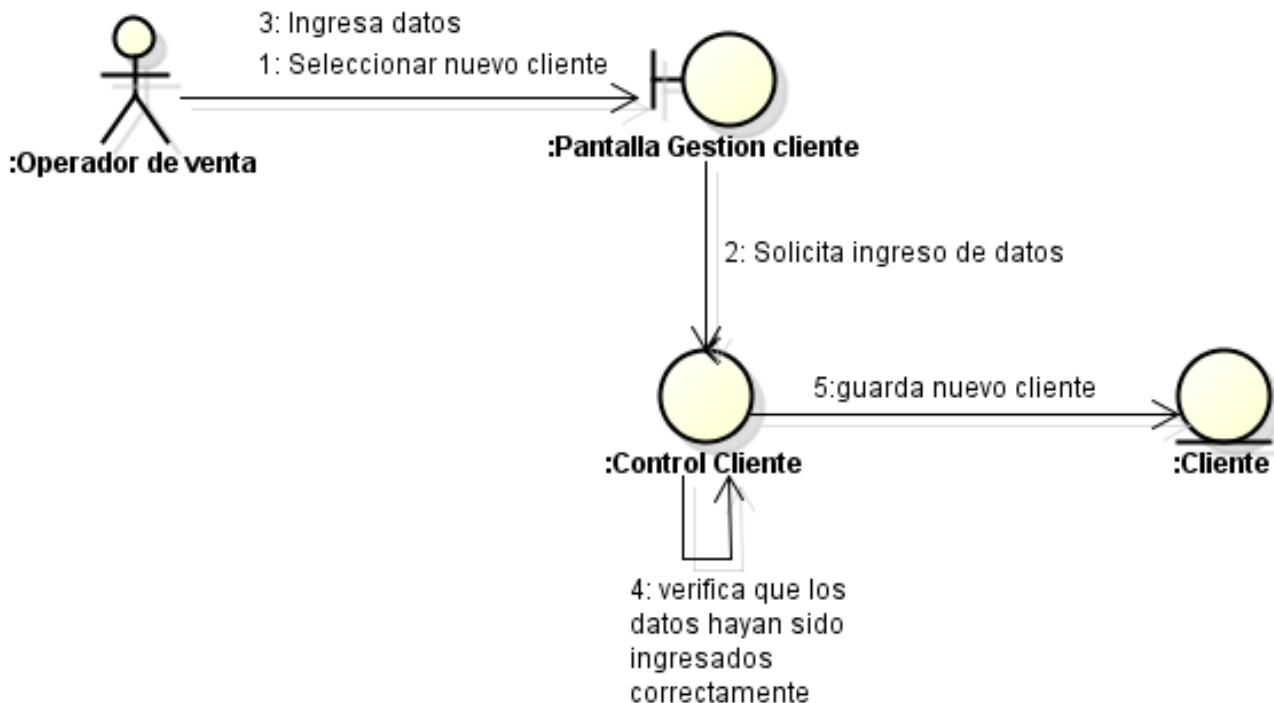


Figura 39: Diagrama de Colaboración del caso de uso: Gestionar Cliente (escenario 1)

Flujo de eventos: El operador de venta selecciona la opción de crear un nuevo cliente (1), luego se solicitan e ingresan los datos del nuevo cliente (2,3), a continuación se verifican los datos (4) y se guarda el nuevo cliente en el sistema (5).

### Escenario 2: Eliminar cliente

En la figura 40 se muestra el escenario 2 del diagrama de colaboración del caso de uso: Gestionar Cliente

Descripción: El operador de venta posicionado en la lista de los clientes selecciona y elimina un cliente existente.

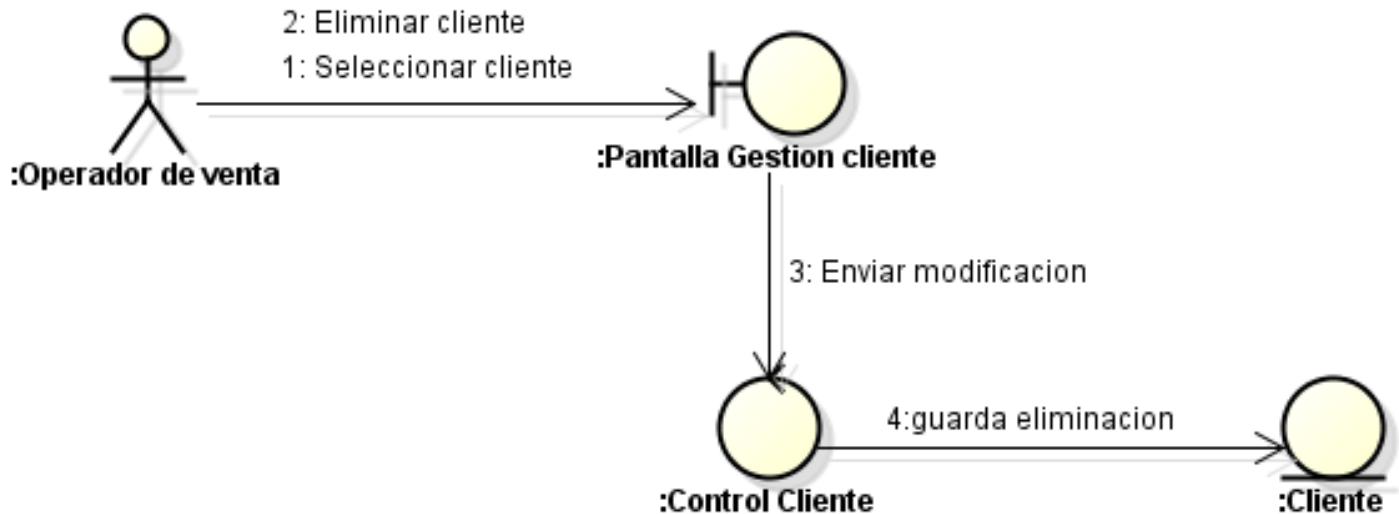


Figura 40: Diagrama de Colaboración del caso de uso: Gestionar Cliente (escenario 2)

Flujo de eventos: El operador de venta selecciona un cliente (1) y oprime el botón de eliminar (2), luego se elimina el cliente y se guarda en el sistema (3,4).

### 5.1.3 Caso de uso: Listados

En la figura 41 se muestra la trazabilidad de casos de uso del sistema a clases de análisis pasa el caso de uso: Listados.

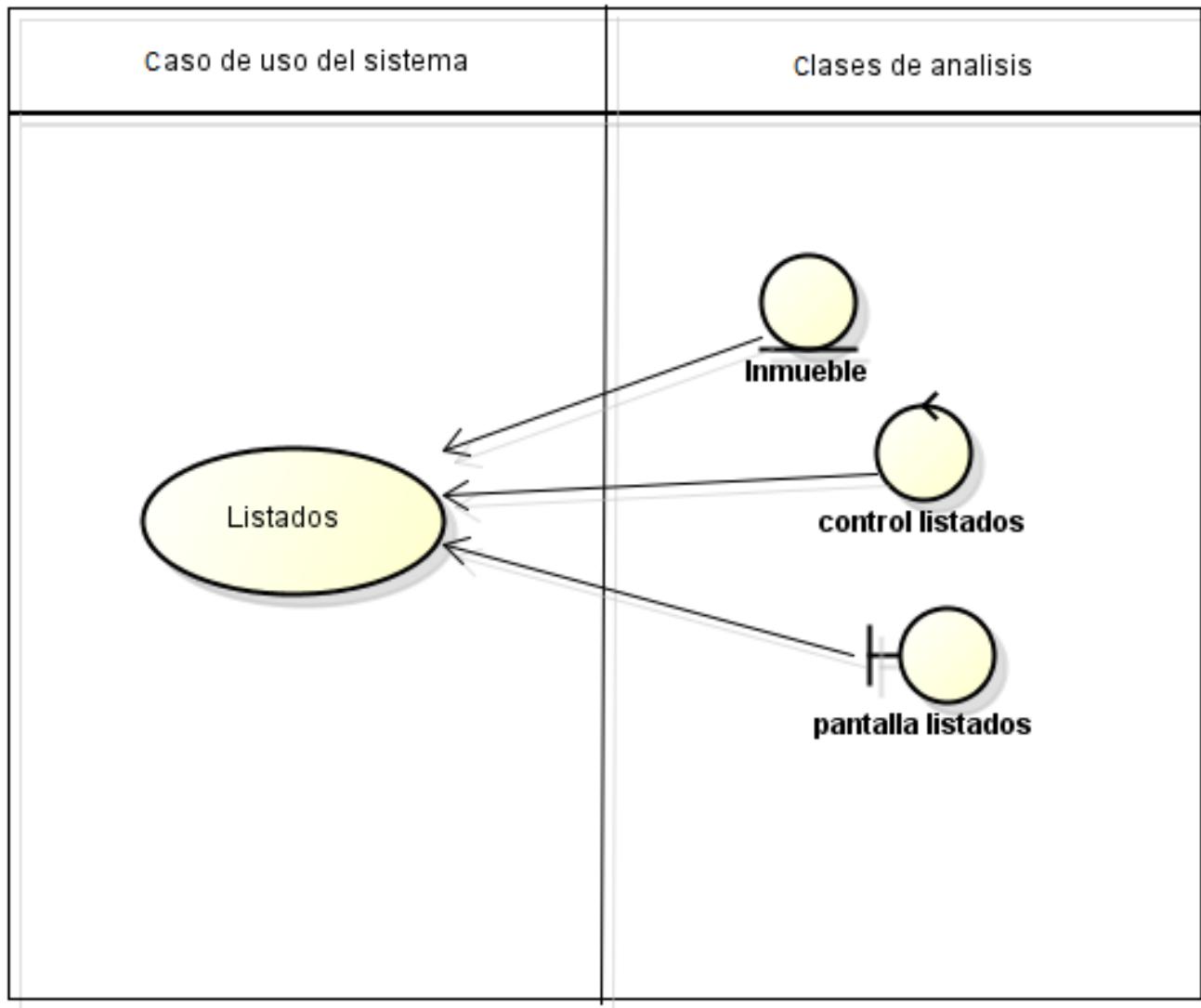


Figura 41: Trazabilidad de casos de uso del sistema a clases de análisis para caso de uso: Listados

En la figura 42 se muestra el diagrama de clases de análisis para caso de uso: Listados

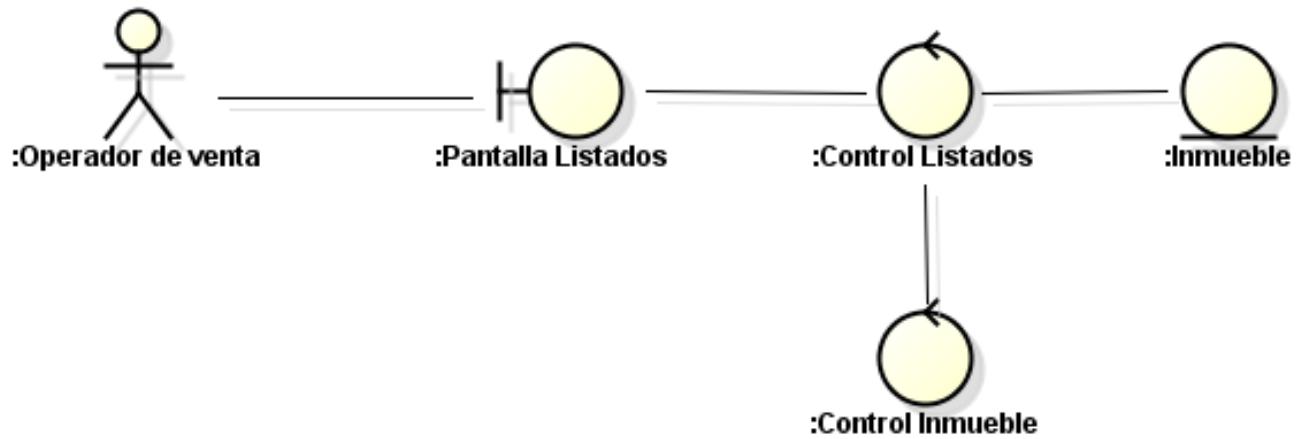


Figura 42: Diagrama de clases de análisis para caso de uso: Listados

## Escenario 1: Listar por precio

En la figura 43 se muestra el escenario 1 del diagrama de colaboración del caso de uso: Listados.

Descripción: El operador de venta desea ordenar los inmuebles por precio.

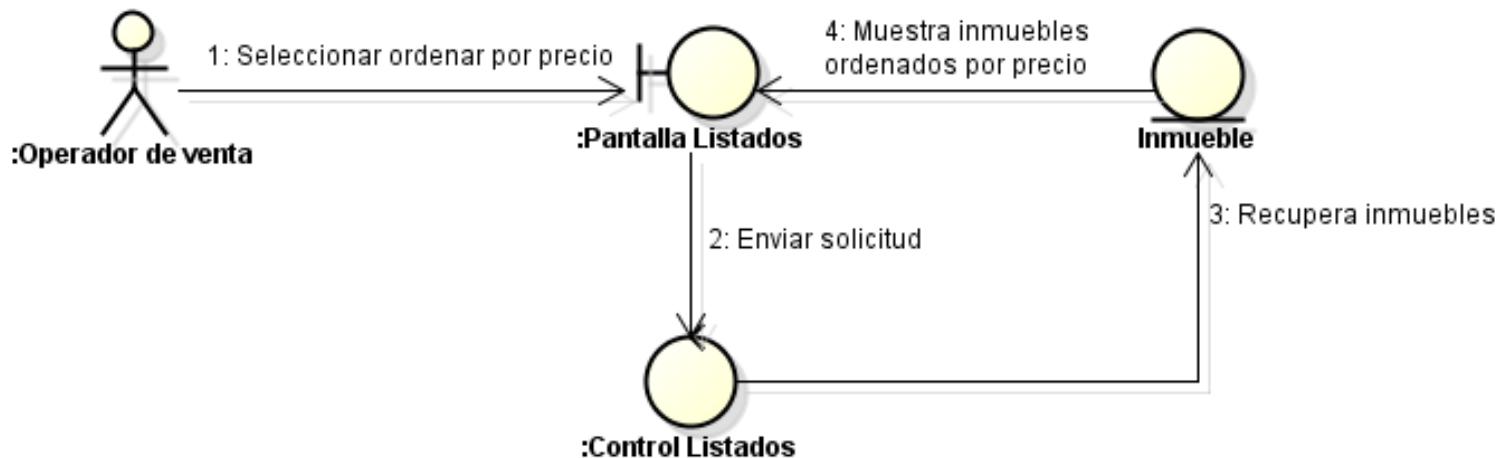


Figura 43: Diagrama de Colaboración del caso de uso: Listados (escenario 1)

Flujo de eventos: El operador de venta selecciona la opción de ordenar por precio (1), a lo que luego el sistema recupera todos los inmuebles y los muestra ordenados por precio (2,3,4).

## Escenario 2: Listar propiedades disponibles.

En la figura 44 se muestra el escenario 2 del diagrama de colaboración del caso de uso: Listados.

Descripción: El operador de venta desea listar los inmuebles disponibles.

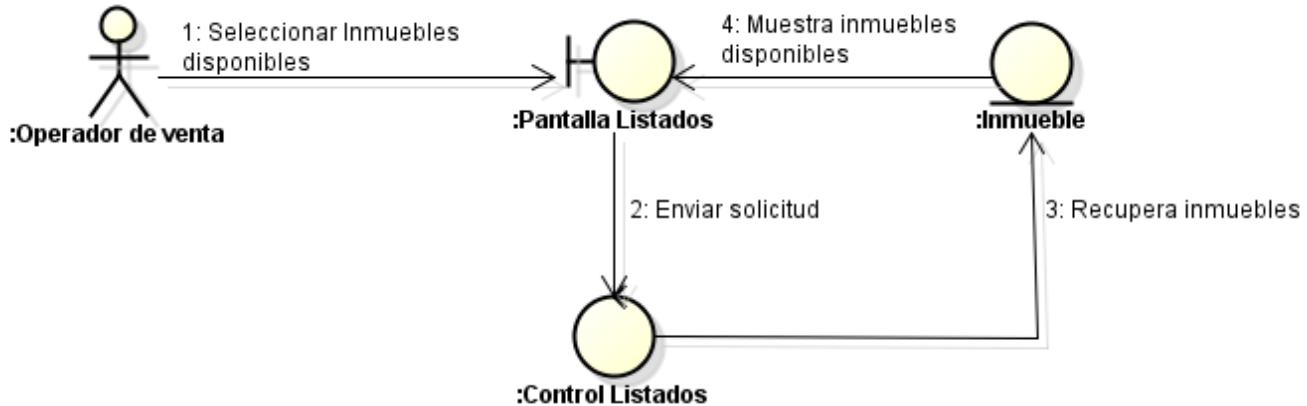


Figura 44: Diagrama de Colaboración del caso de uso: Listados (escenario 2)

Flujo de eventos: El operador de venta selecciona la opción de mostrar inmuebles disponibles (1), luego el sistema recupera todos los inmuebles y muestra los disponibles por pantalla (2,3,4).

## 5.2 Diseño

### 5.2.1 Introducción

Para el diseño de la segunda iteración, en lo que respecta a modelo de diseño, clase de diseño, realización de caso de uso-diseño, subsistema de diseño y descripción de la arquitectura, no habrá cambios con respecto a la iteración 1.

### 5.2.2 Caso de uso: Gestionar Cliente

En la figura 45 se muestra la trazabilidad de clases de análisis definidas hacia las clases de diseño a las cuales se van a implementar en el sistema.

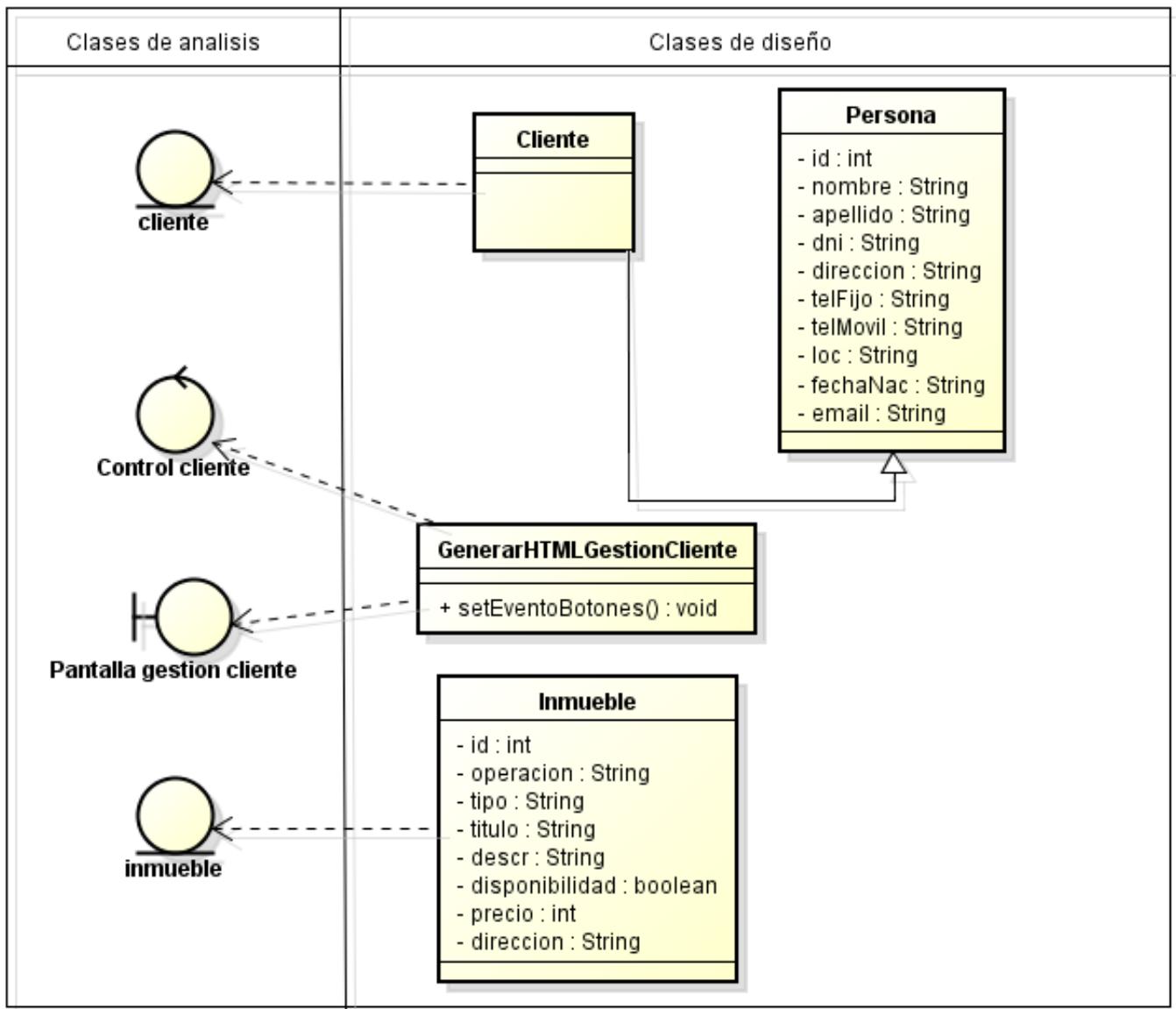


Figura 45: Trazabilidad de clases de análisis a clases de diseño para caso de uso: Gestionar Cliente

Una vez identificadas las clases de diseño, se debe recoger las mismas en un diagrama de clase asociado a la realización anterior. Se realiza este diagrama para mostrar las clases participantes y las relaciones que se utilizan en la realización del caso de uso.

En la figura 46 se muestra el diagrama de clases de diseño para caso de uso: Gestionar Cliente.

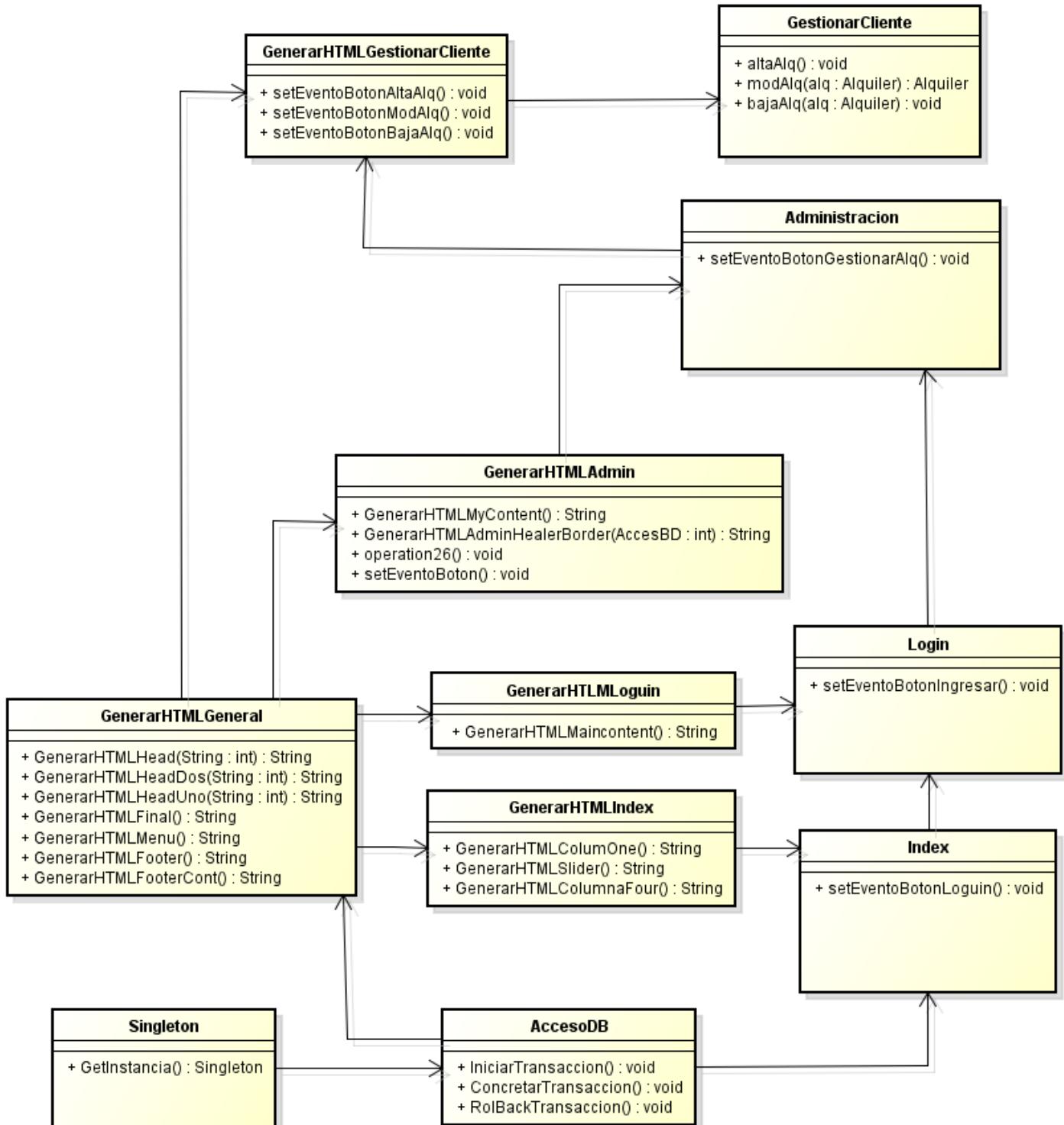


Figura 46: Diagrama de clases de diseño para caso de uso: Gestionar Cliente

Su funcionamiento es el siguiente:

- GenerarHTMLGestionarCliente: Clase que hereda de GenerarHTMLGeneral que contiene la vista de la gestión de los clientes.
- GestionarCliente: Clase que permite dar de alta, modificar o eliminar un alquiler.
- En cuanto a las clases GenerarHTMLGeneral, GenerarHTMLAdmin, GenerarHTMLLoguin, GenerarHTMLIndex y Administracion, su función es la misma que en el caso de uso: Gestionar Alquiler.

A continuación se detallan escenarios concretos a través de diagramas de secuencia del casos de uso Gestionar cliente.

### Escenario 1:

En la figura 47 se muestra el escenario 1 del diagrama de secuencia del caso de uso: Gestionar Cliente

Descripción: El administrador “Pablo” (ya ingresado en el sistema) desea ingresar un cliente en el sistema con los siguientes datos: nombre: “Pedro”, apellido: “Gómez”, DNI: “34778221”, dirección: “San Martin 231” tel. fijo: “4653223”, tel. móvil: “154526997”, localidad: “Rio Cuarto”, fecha de nacimiento: “10/11/1989”, e-mail: “pabloGomez21@hotmail.com”.

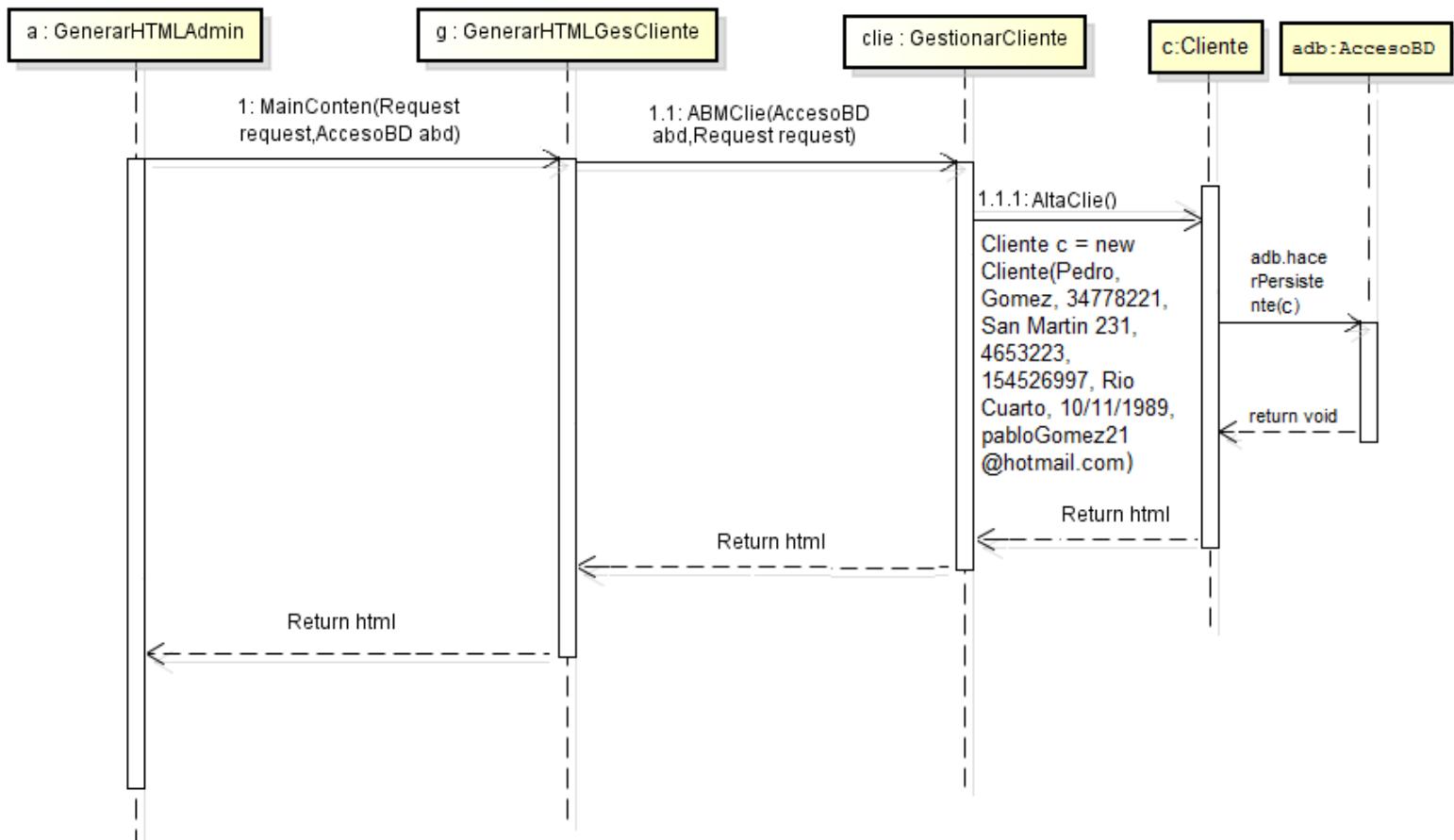


Figura 47: Diagrama de secuencia para caso de uso: Gestionar Cliente (escenario 1)

## Escenario 2:

En la figura 48 se muestra el escenario 1 del diagrama de secuencia del caso de uso: Gestionar Cliente.

Descripción: El administrador “Pablo” (ya ingresado en el sistema” desea eliminar el cliente con el ID “28”.

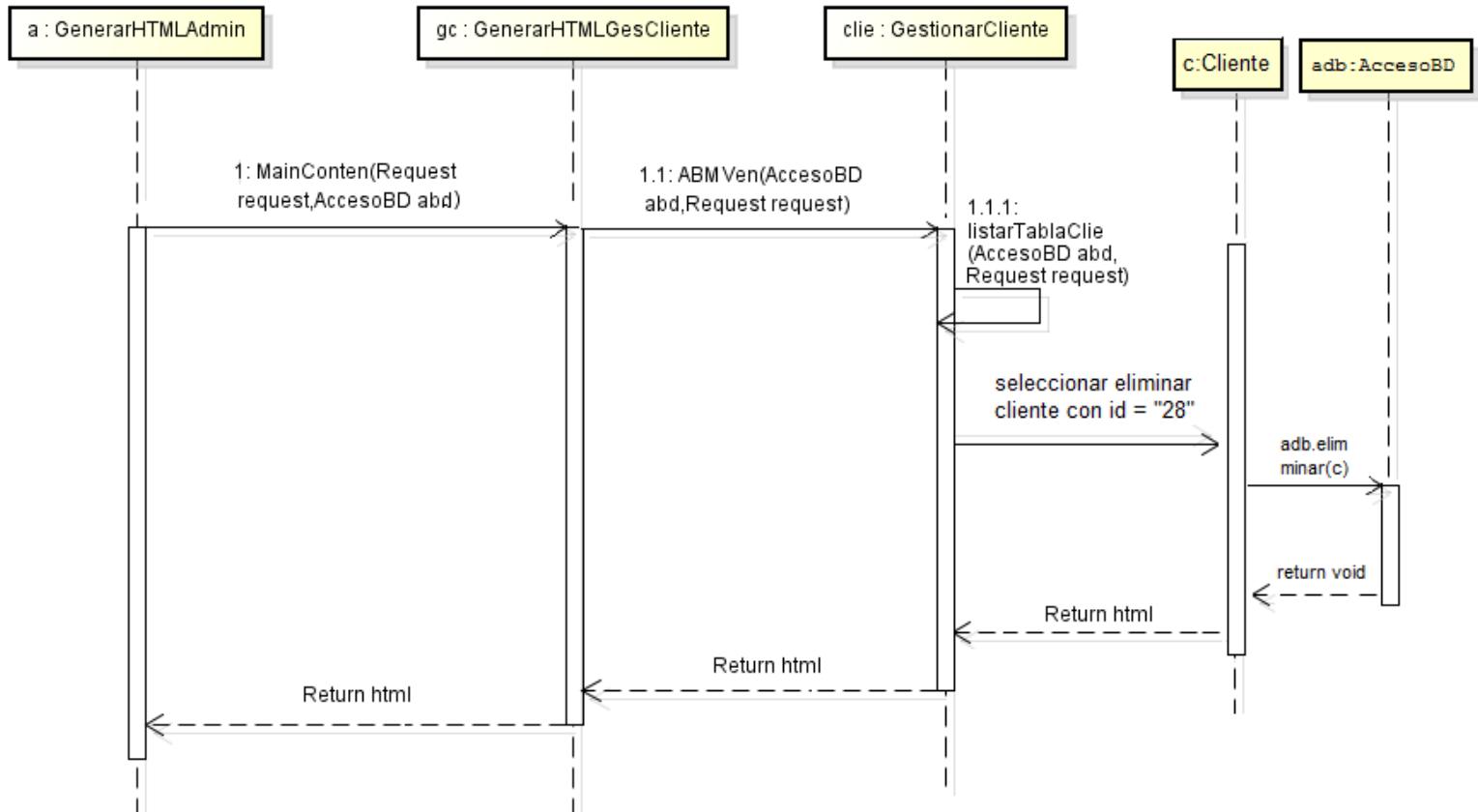


Figura 48: Diagrama de secuencia para caso de uso: Gestionar Cliente (escenario 2)

### 5.2.3 Caso de uso: Listados

En la figura 49 se muestra la trazabilidad de clases de análisis a clases de diseño para el caso de uso: Listados

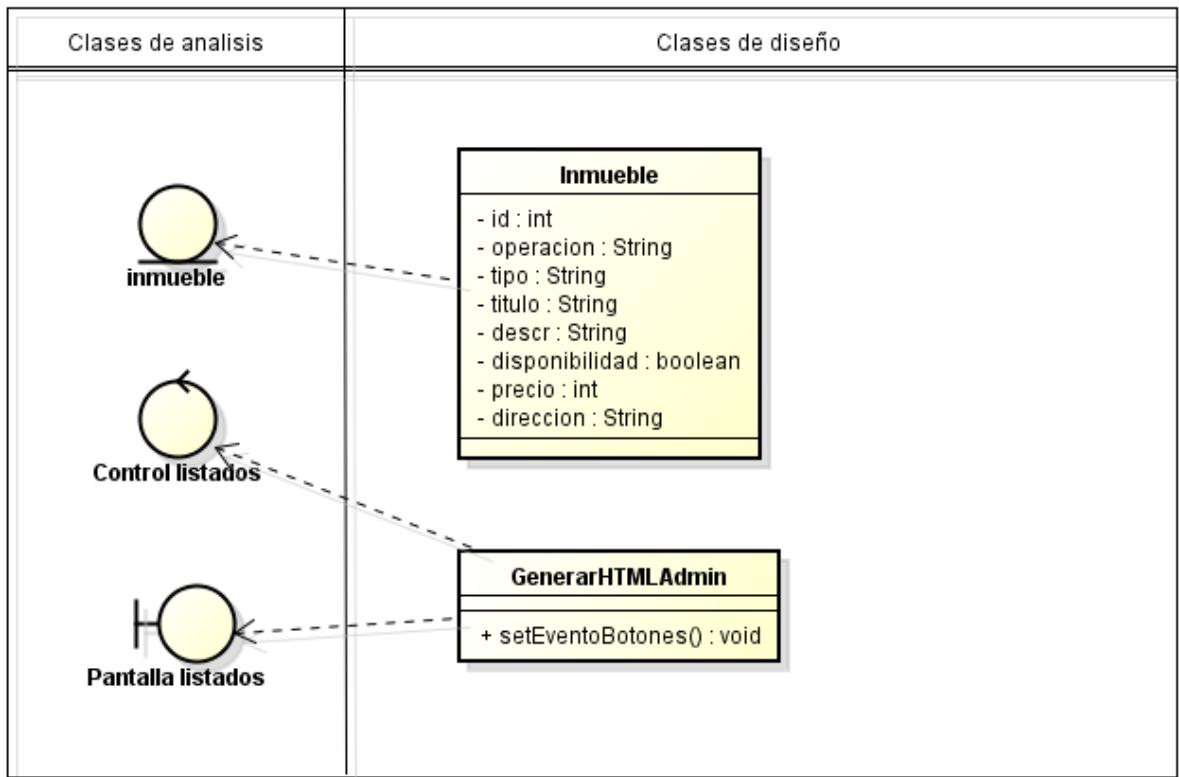


Figura 49: Trazabilidad de clases de análisis a clases de diseño para caso de uso: Listados

En la figura 50 se muestra el diagrama de clases de diseño para caso de uso: Listados

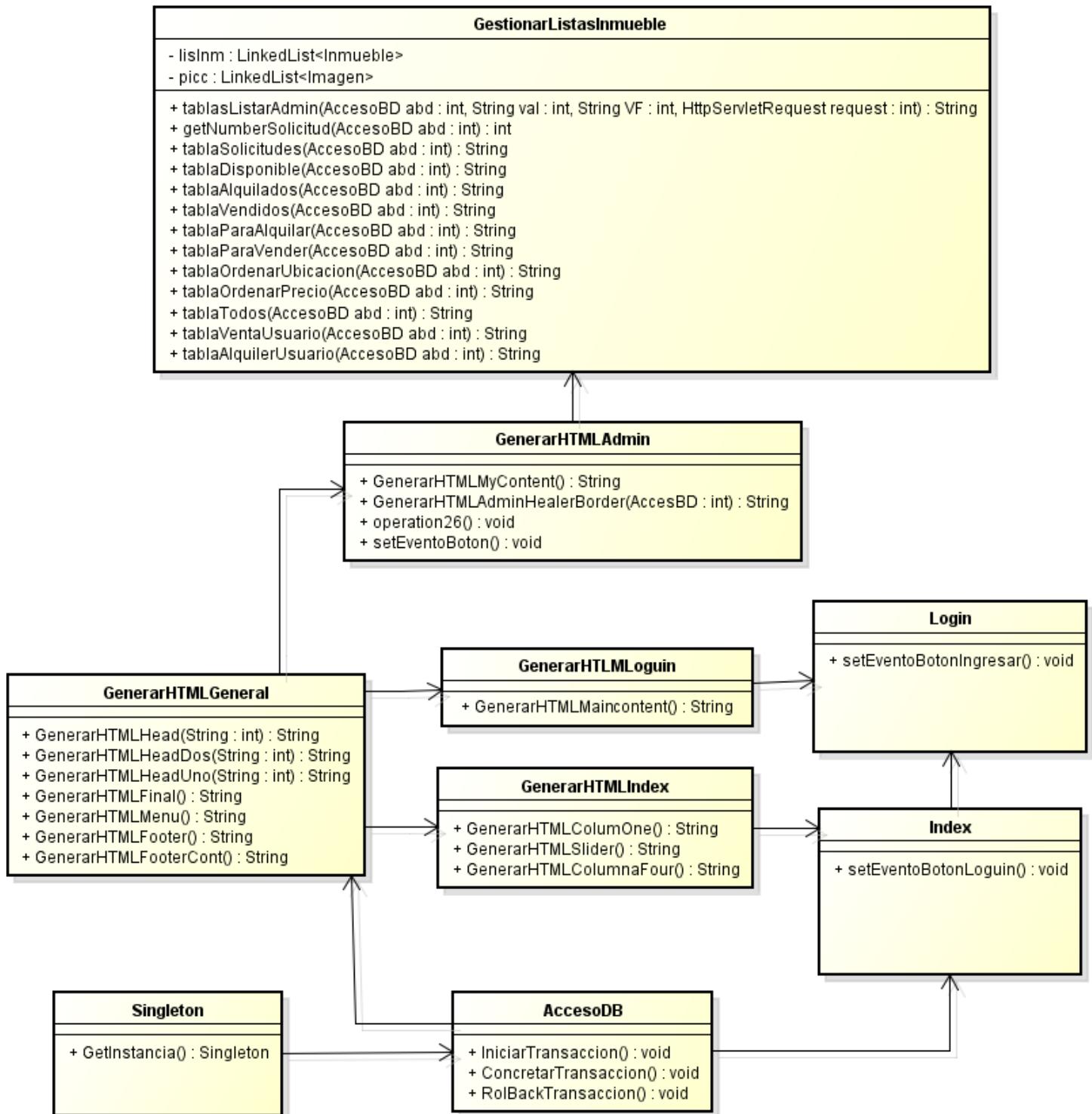


Figura 50: Diagrama de clases de diseño para caso de uso: Listados

Su funcionamiento es el siguiente:

- GestionarListasInmueble: Esta clase se encarga de procesar la lista de inmuebles solicitada y mostrarla en pantalla.

En cuanto a las clases GenerarHTMLGeneral, GenerarHTMLAdmin, GenerarHTMLLoguin, GenerarHTMLIndex y Administracion, su función es la misma que en el caso de uso: Gestionar Alquiler.

A continuación se detallan escenarios concretos a través de diagramas de secuencia del caso de uso: Listados.

### Escenario 1:

En la figura 51 se muestra el escenario 1 del diagrama de secuencia para caso de uso: Listados.

Descripción: El administrador “Juan” (ya ingresado en el sistema) desea listar los inmuebles por precio.

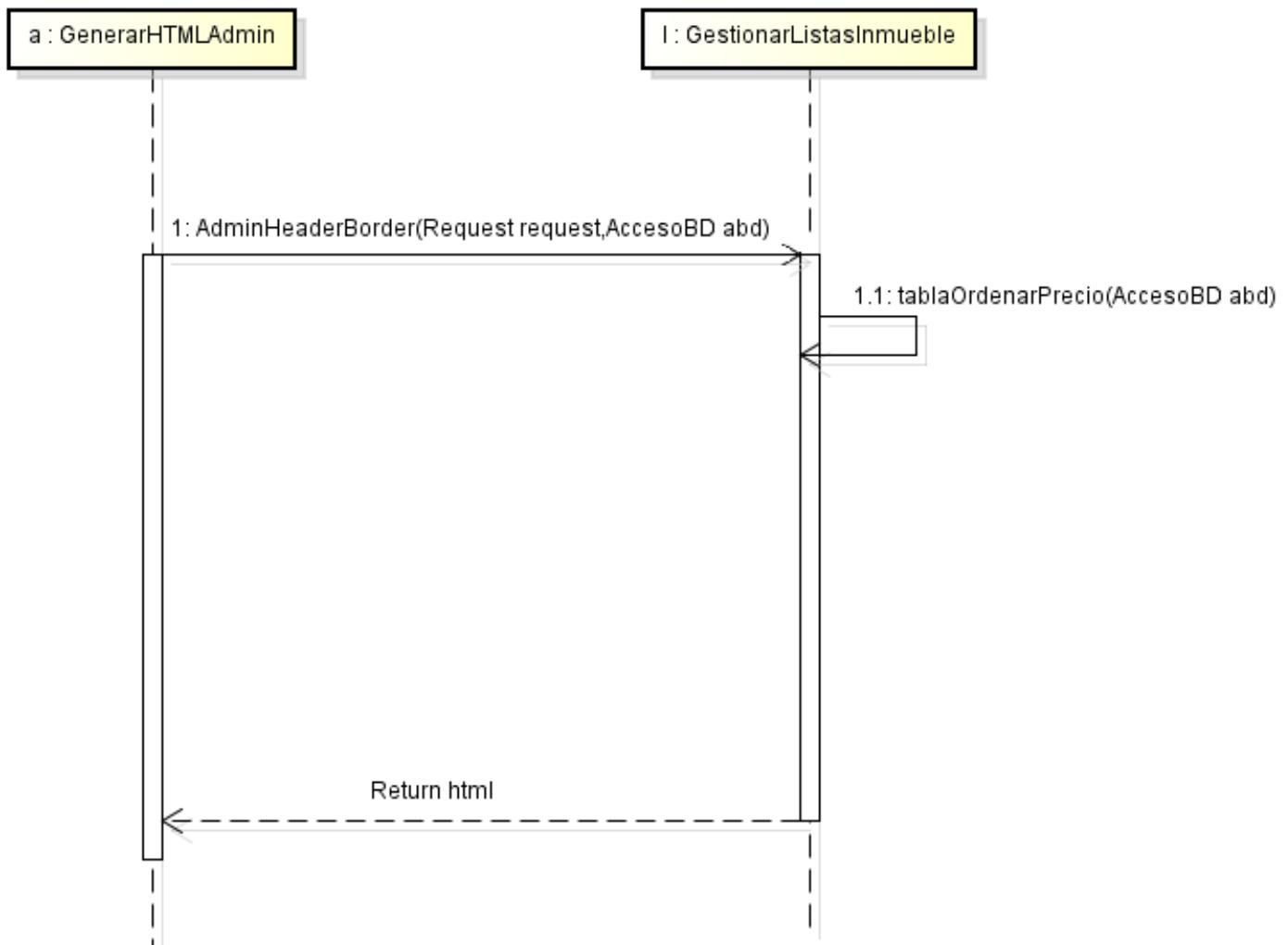


Figura 51: Diagrama de secuencia para caso de uso: Listados (escenario 1)

## Escenario 2:

En la figura 52 se muestra el escenario 2 del diagrama de secuencia para caso de uso: Listados.

Descripción: El administrador “Juan” (ya ingresado en el sistema) desea listar los alquileres de los administradores.

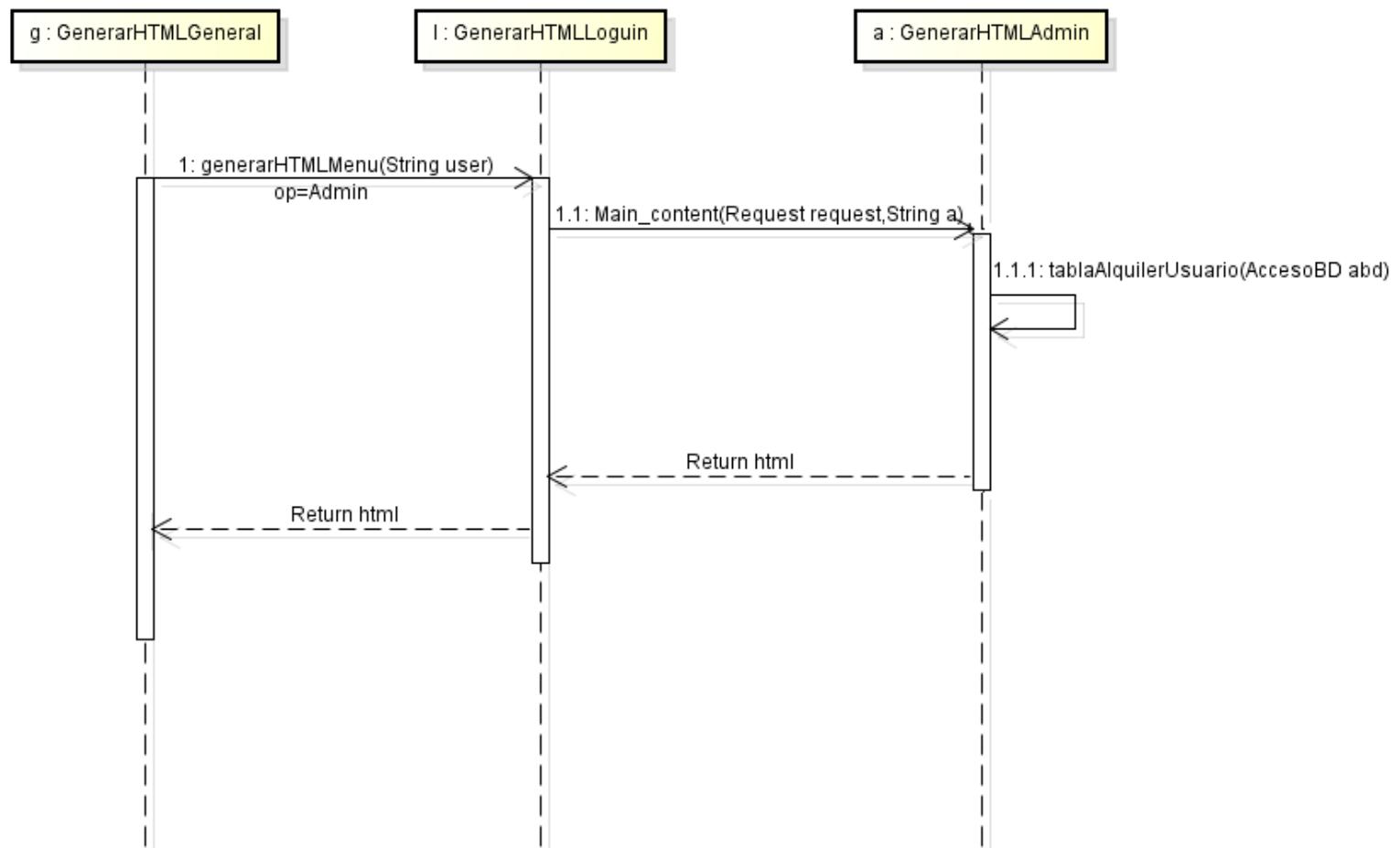


Figura 52: Diagrama de secuencia para caso de uso: Gestionar Cliente (escenario 2)

## **5.3 Implementación**

Esta etapa se lleva a cabo en las mismas condiciones que las definidas en la primera iteración pero se implementan los casos de uso definidos en la segunda iteración.

## **5.4 Prueba**

### **5.4.1 Introducción**

En esta etapa se determina cómo operan los componentes de un sistema en situaciones representativas y se verifica si su comportamiento es el esperado. En esta iteración, no habrá cambios con respecto a la primera iteración en la metodología en la que se desarrolla la prueba, por lo que se procederá a efectuar pruebas de unidad.

### **5.4.2 Caja Blanca**

En la figura 53 se detalla la prueba de caja blanca a través de un grafo de flujo para el caso de uso: Modificar Cliente.



Figura 53: Grafo de flujo para criterio de cobertura de arco de caso de uso: Modificar Cliente

Conjuntos de casos de prueba:

T1 = {<idClie = "10", nombre = "Pedro", apellido="Gonzales", dni ="34554221" dirección = "San Juan 240" email= "PedroGonzales@gmail.com" telFijo="46715332" telMovil = "154321444" localidad= "Rio Cuarto" fechaNac= "12/10/1984>} no encuentra el error

T2 = {<idClie = "12", nombre = "Pedro", apellido="Gonzales", dni ="34554221" dirección = "San Juan 240" email= "PedroGonzales@@" telFijo="46715332" telMovil = "154321444" localidad= "Rio Cuarto" fechaNac= "12/10/1984>} encuentra el error!

### 5.4.3 Caja Negra

En la figura 54 se detalla la prueba de clase de equivalencia (caja negra) del caso de uso: Alta Venta.

Entrada de datos válida para caso de uso: Modificar Cliente

num = {0,1,...,9}+

letra = {a,b,...z,A,B,...,Z}+

fecha = {num"/"num"/"num}

nombre	apellido	dni	direccion	mail	telFijo	telMovil	localidad	fechaNac	Salida esperada
letra	letra	letra	letra\num	letra	num	num	letra	num/num	Error
letra	letra	num	letra\num	letra@le tra.com	num	num	letra	num/num/ num	Valida
letra	letra	num	letra\num	num	num	letra	letra	letra	Error
letra	letra	num	num	letra	letra	num	letra	letra	Error

Figura 54: Prueba de clases de equivalencia para caso de uso: Modificar Cliente

## **6 Conclusiones**

El sistema realizado está destinado a la inmobiliaria Martínez Latorre, una empresa interesada en administrar y hacer crecer su negocio gracias a una solución de gestión sólida, confiable, fácil de controlar y adecuada a sus necesidades.

Con el adecuado uso del sistema, se garantiza evitar errores involuntarios que se producen debido al trabajo manual, la mayoría de las veces la importancia de estos errores es muy grande, con costos mucho más altos que el valor de una solución integrada y confiable como la que se propone.

El sistema web presentado es producto de un seguimiento constante y un profundo análisis del funcionamiento de la empresa, con lo cual se logra un producto confiable, que se adapta al trabajo actual de la misma, siendo flexible a futuros cambios. Cabe destacar que este sistema fue diseñado de manera tal que los usuarios puedan aprender a usarlo fácilmente amoldándose a su uso sin tener que realizar grandes cambios en su forma de trabajo.

Durante el desarrollo de este proyecto, fuimos aplicando conocimientos adquiridos durante el transcurso de nuestra carrera, como así también durante la realización del proyecto, igualmente todos fueron aplicados con el fin de desarrollar un software que cumple con los requerimientos planteados.

Cabe destacar además, que ha terminado una etapa muy importante, la cual nos ha servido en muchos aspectos a formarnos como futuros profesionales. Destacamos todos los contenidos aprendidos, la dedicación, y la práctica de hacer un sistema tan grande y complejo, abriéndonos camino a nuestra primera experiencia como Analistas en Computación.

## 7 Trabajos futuros

A continuación se detallarán distintas mejoras o futuras extensiones que notamos de gran importancia destacar:

- En el buscador de propiedades, agregar el campo de búsqueda “Localidad” para una búsqueda más eficiente, actualmente no está implementado ya que esta empresa está dentro del ámbito local (Rio Cuarto).
- La implementación de impresión de contratos formales ya sean tanto de alquileres como de ventas.
- La muestra de datos, este es un punto que se iría mejorando una vez que el personal de la Inmobiliaria Martínez Latorre vaya utilizando el sistema. Es decir, la idea es que los usuarios determinen sobre qué datos son o no de interés para una determinada ventana.
- Agregar una sección de Gestionar Subastas.
- Implementar tipos de administradores, los cuales van desde gerente hasta empleado con sus respectivos privilegios.
- Agregar paginación, la cual es muy útil cuando se obtienen consultas que arrojan muchos elementos, ya que el usuario tiene que esperar que termine su carga la mayoría de las veces innecesariamente, con paginación una consulta que arroja por ej. 100 inmuebles, estaría dividida en 10 páginas de 10 inmuebles cada uno.
- Agilizar la interacción de la interfaz para que resulte más cómoda para el usuario.

Cabe destacar que estamos abiertos a opiniones y/o sugerencias una vez que se empiece a utilizar el sistema en la empresa, ya sean tanto de diseño como de funcionalidades.

## 8 Bibliografía

- Roger S. Pressman. (2002) Ingeniería del Software. Un Enfoque Práctico. Quinta Edición.
- Booch Grady, Rumbaugh James, Jacobson Ivar (2006) El Lenguaje Unificado de Modelado. Addison Wesley. Pearson Education. Capítulos 26, 29 y 30.
- Jacobson Ivar, Booch Grady, Rumbaugh James (1999) The Unified Software Development Process. Addison Wesley.
- Paul Jorgensen (1995) Software Testing. A Craftsman's Approach. Capítulo 5: Boundary Value Testing Capítulo 6: Equivalence Class Testing Capítulo 7: Decision Table Based Testing
- C. Ghezzi, M. Jazayeri, D. Mandrioli (1991) Fundamentals of Software Engineering
- Sitio web MYSQL: <http://dev.mysql.com>
- Sitio web Jdo: <http://tjdo.sourceforge.net>
- Sitio web de NetBeans: <https://netbeans.org/>
- Sitio web de utilidades Java: [www.javahispano.org](http://www.javahispano.org)

## 9 Anexos

### 9.1 Anexo I Proceso Unificado

El Proceso Unificado no es simplemente un proceso, sino un marco de trabajo extensible que puede ser adaptado a organizaciones o proyectos específicos. De la misma forma, el Proceso Unificado de Rational, también es un marco de trabajo extensible, por lo que muchas veces resulta imposible decir si un refinamiento particular del proceso ha sido derivado del Proceso Unificado o del RUP. Por dicho motivo, los dos nombres suelen utilizarse para referirse a un mismo concepto.

El nombre **Proceso Unificado** se usa para describir el proceso genérico que incluye aquellos elementos que son comunes a la mayoría de los refinamientos existentes. También permite evitar problemas legales ya que Proceso Unificado de Rational o RUP son marcas registradas por IBM (desde su compra de Rational Software Corporation en 2003). El primer libro sobre el tema se denominó, en su versión española, *El Proceso Unificado de Desarrollo de Software* (ISBN 84-7829-036-2) y fue publicado en 1999 por Ivar Jacobson, Grady Booch y James Rumbaugh, conocidos también por ser los desarrolladores del UML, el Lenguaje Unificado de Modelado. Desde entonces los autores que publican libros sobre el tema y que no están afiliados a Rational utilizan el término Proceso Unificado, mientras que los autores que pertenecen a Rational favorecen el nombre de Proceso Unificado de Rational.

#### **Proceso unificado a través del lenguaje Unificado de Modelado (UML)**

**UML** (Unified Modeling Language), es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; está respaldado por el OMG (Object Management Group).

Se puede aplicar en el desarrollo de software gran variedad de formas para dar soporte a una metodología de desarrollo de software (tal como el Proceso Unificado), pero no específica en sí mismo qué metodología o proceso usar.

Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio, funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y compuestos reciclados.

Es importante remarcar que UML es un "lenguaje de modelado" para especificar o para describir métodos o procesos. Se utiliza para definir un sistema, para detallar los artefactos en el sistema y para documentar y construir. En otras palabras, es el lenguaje en el que está descrito el modelo.

UML no puede compararse con la programación estructurada, pues UML significa Lenguaje Unificado de Modelado, no es programación, solo se diagrama la realidad de una utilización en un requerimiento. Mientras que, programación estructurada, es una forma de programar como lo es la orientación a objetos, la programación orientada a objetos viene siendo un complemento perfecto de UML, pero no por eso se toma UML sólo para lenguajes orientados a objetos.

UML cuenta con varios tipos de diagramas, los cuales muestran diferentes aspectos de las entidades representadas, algunos de estos diagramas son los siguientes:

- Diagrama de Clases: es un tipo de diagrama estático que describe la estructura de un sistema mostrando sus clases, orientados a objetos.
- Diagrama de Componentes: representa cómo un sistema de software es dividido en componentes y muestra las dependencias entre estos componentes. Los componentes físicos incluyen archivos, cabeceras, bibliotecas compartidas, módulos, ejecutables, o paquetes. Los diagramas de Componentes prevalecen en el campo de la arquitectura de software pero pueden ser usados para modelar y documentar cualquier arquitectura de sistema.
- Diagrama de Casos de Uso: es una forma de diagrama de comportamiento UML mejorado. El Lenguaje de Modelado Unificado (UML), define una notación gráfica para representar casos de uso llamada modelo de casos de uso.
- Diagrama de secuencia: es un tipo de diagrama usado para modelar interacción entre objetos en un sistema.
- Un diagrama de colaboración: en las versiones de UML 1.x es esencialmente un diagrama que muestra interacciones organizadas alrededor de los roles.
- El Diagrama de Despliegue es un tipo de diagrama que se utiliza para modelar la disposición física de los artefactos software en nodos (usualmente plataforma de hardware).

## **El proceso Unificado de Desarrollo de Software o simplemente Proceso Unificado**

Es un marco de desarrollo de software que se caracteriza por estar dirigido por casos de uso, centrado en la arquitectura y por ser iterativo e incremental. El refinamiento más conocido y documentado del Proceso Unificado es el Proceso Unificado de Rational o simplemente RUP.

## Características

- **Iterativo e Incremental:** El Proceso Unificado es un marco de desarrollo iterativo e incremental compuesto de cuatro fases denominadas Inicio, Elaboración, Construcción y Transición. Cada una de estas fases es a su vez dividida en una serie de iteraciones (la de inicio puede incluir varias iteraciones en proyectos grandes). Estas iteraciones ofrecen como resultado un incremento del producto desarrollado que añade o mejora las funcionalidades del sistema en desarrollo.  
Cada una de estas iteraciones se divide a su vez en una serie de disciplinas que recuerdan a las definidas en el ciclo de vida clásico o en cascada: Análisis de requisitos, Diseño, Implementación y Prueba. Aunque todas las iteraciones suelen incluir trabajo en casi todas las disciplinas, el grado de esfuerzo dentro de cada una de ellas varía a lo largo del proyecto.
- **Dirigido por los casos de uso:** En el Proceso Unificado los casos de uso se utilizan para capturar los requisitos funcionales y para definir los contenidos de las iteraciones. La idea es que cada iteración tome un conjunto de casos de uso o escenarios y desarrolle todo el camino a través de las distintas disciplinas: diseño, implementación, prueba, etc. El proceso dirigido por casos de uso es el rup. Nota: en UP se está Dirigido por requisitos y riesgos: de acuerdo con el Libro UML 2 de ARLOW, Jim que menciona el tema.
- **Centrado en la arquitectura:** El Proceso Unificado asume que no existe un modelo único que cubra todos los aspectos del sistema. Por dicho motivo existen múltiples modelos y vistas que definen la arquitectura de software de un sistema. La analogía con la construcción es clara, cuando construyes un edificio existen diversos planos que incluyen los distintos servicios del mismo: electricidad, fontanería, etc.
- **Enfocado en los riesgos:** El Proceso Unificado requiere que el equipo del proyecto se centre en identificar los riesgos críticos en una etapa temprana del ciclo de vida. Los resultados de cada iteración, en especial los de la fase de Elaboración deben ser seleccionados en un orden que asegure que los riesgos principales son considerados primero.

## **¿Por qué analizar y diseñar?**

En resumidas cuentas, la cuestión fundamental del desarrollo del software es la escritura del código. Después de todo, los diagramas son solo imágenes bonitas. Ningún usuario va a agradecer la belleza de los dibujos; lo que el usuario quiere es software que funcione (UML Gota a Gota, Addison Wesley, Pag 7). Por lo tanto, cuando considere usar el UML es importante preguntarse por qué lo hará y como le ayudara a usted cuando llegue el momento de escribir el código. No existe una evidencia empírica adecuada que demuestre si estas técnicas son buenas o malas; Pero lo que si es cierto es que es de considerable ayuda para las etapas de mantenimiento en proyectos de mediana/avanzada envergadura.

La metodología de UP, es un método iterativo de diseño de software que describe cómo desarrollar software de forma eficaz, utilizando técnicas probadas en la industria.

El Proceso Unificado de Desarrollo de Software o simplemente Proceso Unificado es un marco de desarrollo de software que se caracteriza por estar dirigido por casos de uso, centrado en la arquitectura, enfocado en el riesgo, y por ser iterativo e incremental.

El Proceso Unificado no es simplemente un proceso, sino un marco de trabajo extensible que puede ser adaptado a organizaciones o proyectos específicos.

El nombre Proceso Unificado se usa para describir el proceso genérico que incluye aquellos elementos que son comunes a la mayoría de los refinamientos existentes. Es una metodología orientada a conducir el proceso de desarrollo de software en sus aspectos técnicos; los flujos y productos de trabajo de UP no incluyen la administración del proyecto.

## **Fases de Desarrollo**

### **Fase de Inicio.**

- Es la fase más pequeña del proyecto e, idealmente, debe realizarse también en un periodo de tiempo pequeño (una única iteración).
- El hecho de llevar a cabo una fase de inicio muy larga indica que se está realizando una especificación previa excesiva, lo que responde más a un modelo en cascada.
- Objetivos:
  1. Establecer una justificación para el proyecto.
  2. Establecer el ámbito del proyecto.
  3. Esbozar los casos de uso y los requisitos clave que dirigirán las decisiones de diseño.
  4. Esbozar las arquitecturas candidatas.

- 5. Identificar riesgos.
- 6. Preparar el plan del proyecto y la estimación de costes.
  
- El hito de final de fase se conoce como Hito Objetivo del Ciclo de Vida.

### **Fase de Elaboración.**

- Durante esta fase se capturan la mayoría de los requisitos del sistema.
- Los objetivos principales de esta fase serán la identificación de riesgos y establecer y validar la arquitectura del sistema.
- Base de Arquitectura Ejecutable:
  - La arquitectura se valida a través de la implementación de una Base de Arquitectura Ejecutable: se trata de una implementación parcial del sistema que incluye los componentes principales del mismo.
  - Al final de la fase de elaboración la base de arquitectura ejecutable debe demostrar que soporta los aspectos clave de la funcionalidad del sistema y que muestra la conducta adecuada en términos de rendimiento, escalabilidad y coste.
- Al final de la fase se elabora un plan para la fase de construcción.
- El hito arquitectura del ciclo de vida marca el final de la fase.

### **Fase de construcción.**

- Es la fase más larga de proyecto.
- El sistema es construido en base a lo especificado en la fase de elaboración.
- Las características del sistema se implementan en una serie de iteraciones cortas y limitadas en el tiempo.
- El resultado de cada iteración es una versión ejecutable de software.
- El hito de capacidad operativa inicial marca el final de la fase.

### **Fase de transición.**

- En esta fase el sistema es desplegado para los usuarios finales.

- La retroalimentación recibida permite incorporar refinamientos al sistema en las sucesivas iteraciones.
- Esta iteración también cubre el entrenamiento de los usuarios para la utilización del sistema.
- El hito de lanzamiento del producto marca el final de la fase.

## **Disciplinas**

### **Modelado del negocio.**

- El objetivo es establecer un canal de comunicación entre los ingenieros del negocio y los ingenieros del software.
- Los ingenieros del software deben conocer la estructura y dinámica de la organización objetivo (el cliente), los problemas actuales y sus posibles mejoras.
- Se plasma en la identificación del modelo del dominio en el que se visualizan los aspectos básicos del dominio de aplicación.

### **Requisitos.**

- El objetivo es describir que es lo que tiene que hacer el sistema y poner a los desarrolladores y al cliente de acuerdo en esta descripción.

### **Análisis y diseño.**

- Describe como el software será realizado en la fase de implementación.
- Se plasma en un modelo de diseño que consiste en una serie de clases (agrupadas en paquetes y subsistemas) con interfaces bien definidos.
- También contiene descripciones de cómo los objetos colaboran para realizar las acciones incluidas en los casos de uso.

### **Implementación.**

- Se implementan las clases y objetos en términos de componentes (ficheros fuentes, binarios, ejecutables, entre otros).

### **Prueba.**

- Se comprueba que el funcionamiento es correcto analizando diversos aspectos: los objetos como unidades, la integración entre objetos, la implementación de todos los requisitos, entre otros.

### **Despliegue.**

- Se crea la versión externa del producto, se empaqueta, se distribuye y se instala en el lugar de trabajo. También se da asistencia y ayuda a los usuarios.

### Gestión de configuraciones y cambios.

- Gestiona aspectos como los sistemas de control de versiones.
- Controla las peticiones de cambios clasificándolas según su estado (nueva, registrada, aprobada, asignada, completa, entre otros).
- Los datos se almacenan en una base de datos y se pueden obtener informes periódicos.
- Herramientas como Rational ClearQuest o Bugzilla.

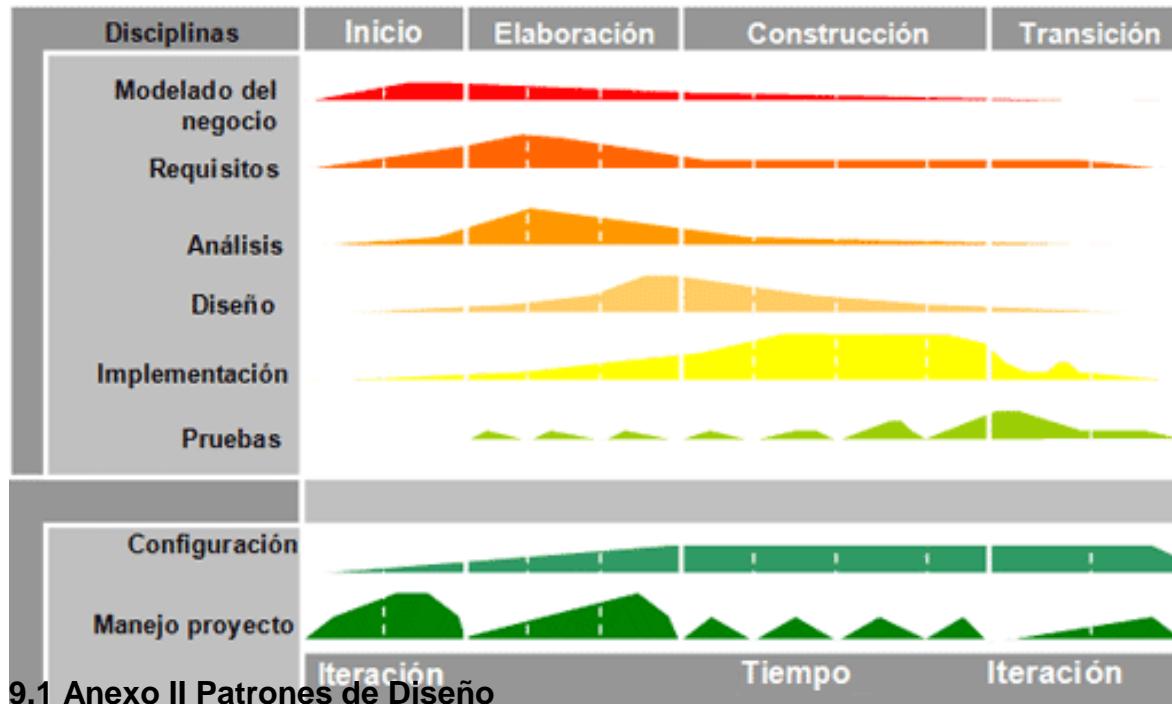
### Gestión del proyecto.

- Encargada de definir los planes del proyecto global, los planes de fase y los planes de iteración.

### Entorno.

- Se centra en las actividades necesarias para configurar el proceso de un proyecto.
- El objetivo es proveer a la organización de desarrollo software de un entorno de trabajo (que incluye procedimientos y herramientas) que soporten al equipo de desarrollo.

El ciclo de vida del Proceso Unificado está sintetizado en la siguiente ilustración.



## Patrones de Diseño

Los patrones de software son soluciones de diseño reusables a problemas recurrentes.

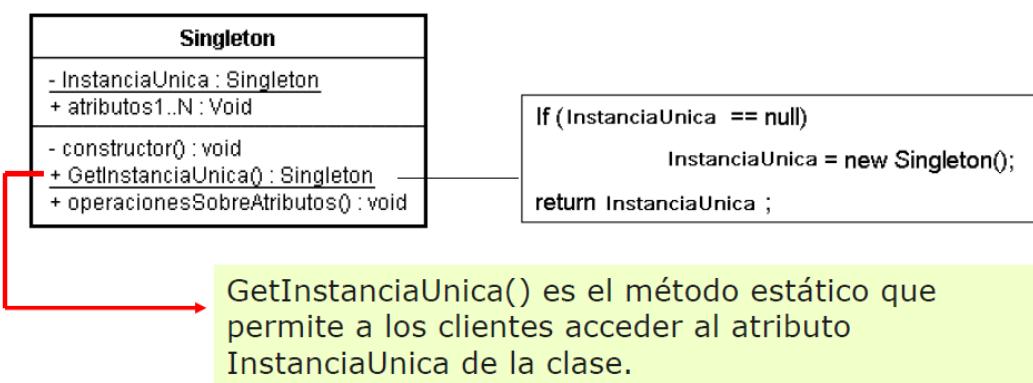
### Clasificación de Patrones

- **Creacionales**  
Resuelven problemas relativos a la creación de objetos.
- **Estructurales**  
Resuelven problemas relativos a la composición de clases y objetos.
- **Comportamiento**  
Resuelven problemas relativos a la interacción entre los objetos y la distribución de responsabilidades.

Creacionales	Estructurales	Comportamiento
Singleton	Composite	Command
Abstract Factory	Proxy	Iterator
Factory Method	Facade	Mediator Template Method Visitor Observer

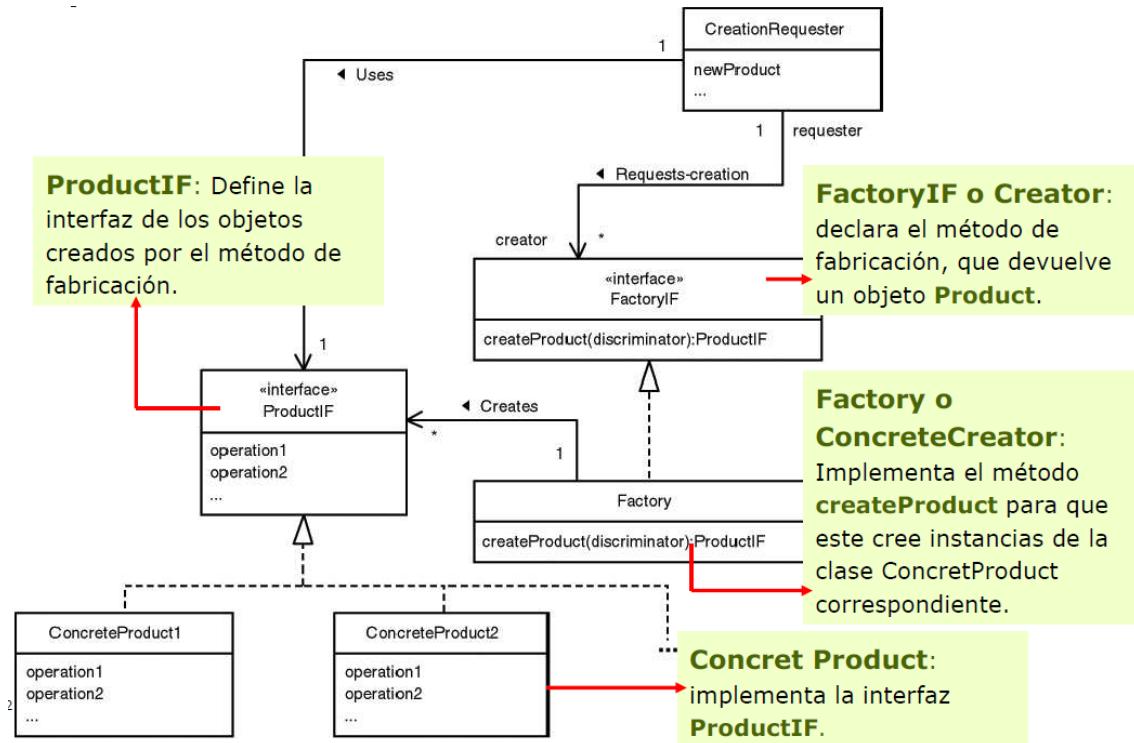
### Patrón Singleton

Asegura que solo existe una instancia de cada clase, y que todos los objetos que la usan accederán a la misma instancia.



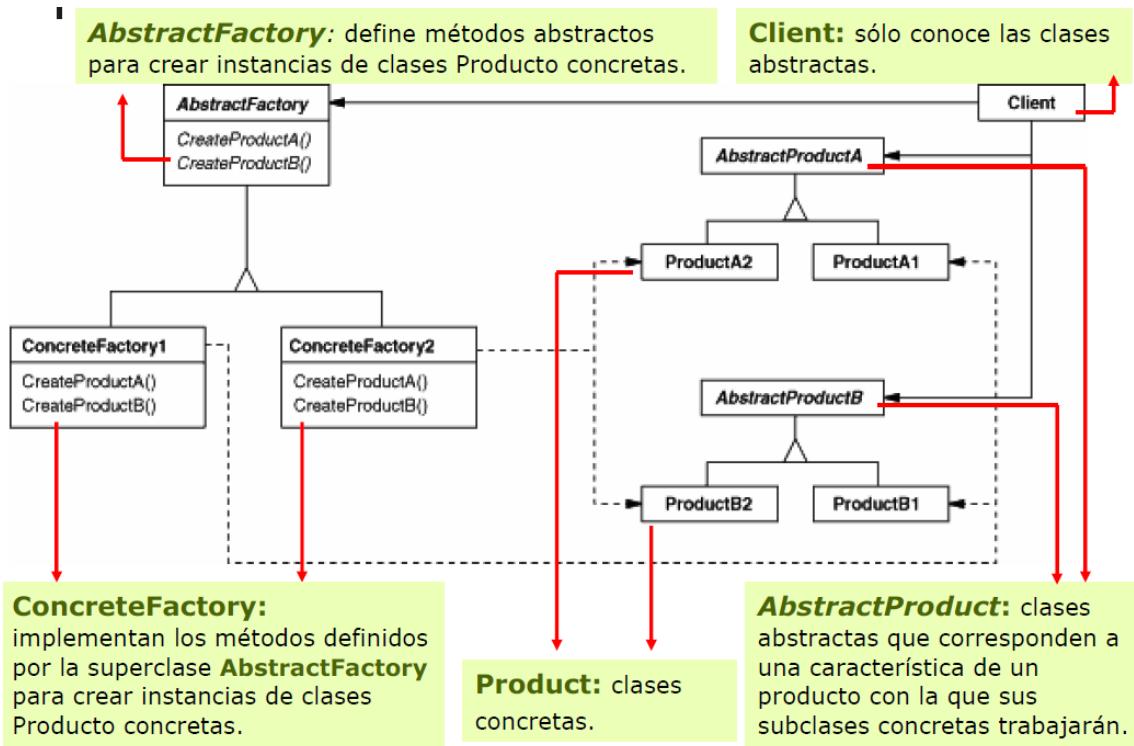
## Patrón Factory Method

Define una interfaz para la creación de objetos, pero delega en las subclases la decisión de qué o cual clase instanciar.



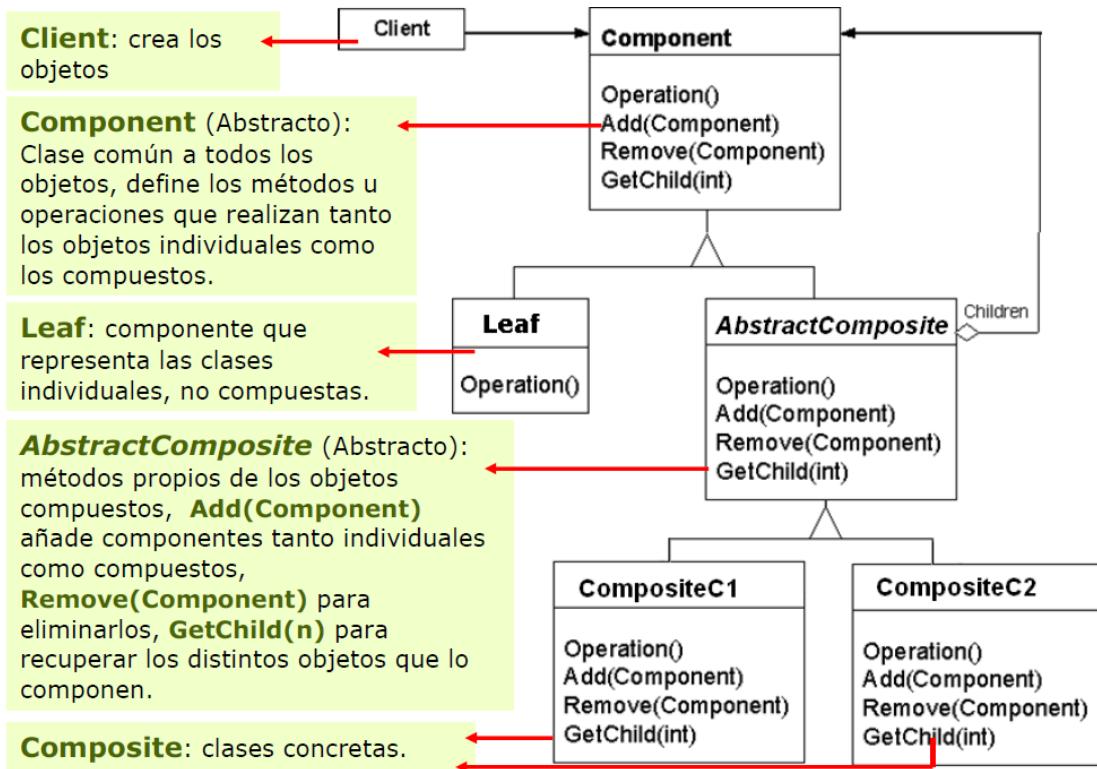
## Patrón Abstract Factory

Provee una interfaz que permite crear familias de objetos relacionados o dependientes, sin especificar sus clases concretas.



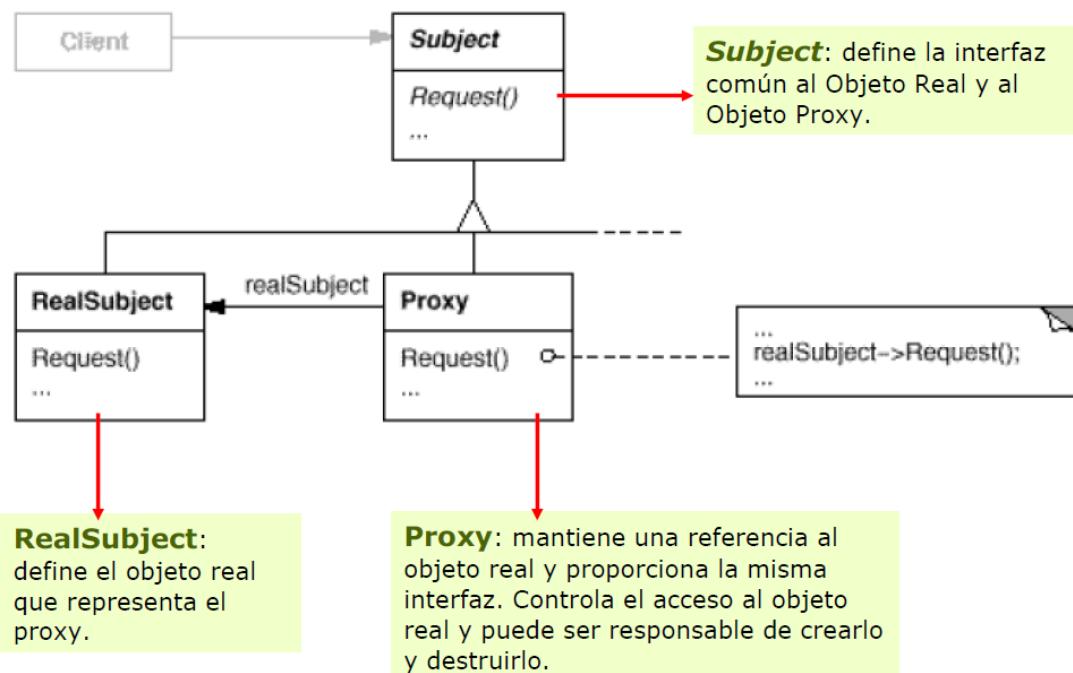
## Patrón Composite

Permite que el cliente maneje indistintamente objetos individuales o composiciones de objetos y los trate como una estructura compuesta uniforme.



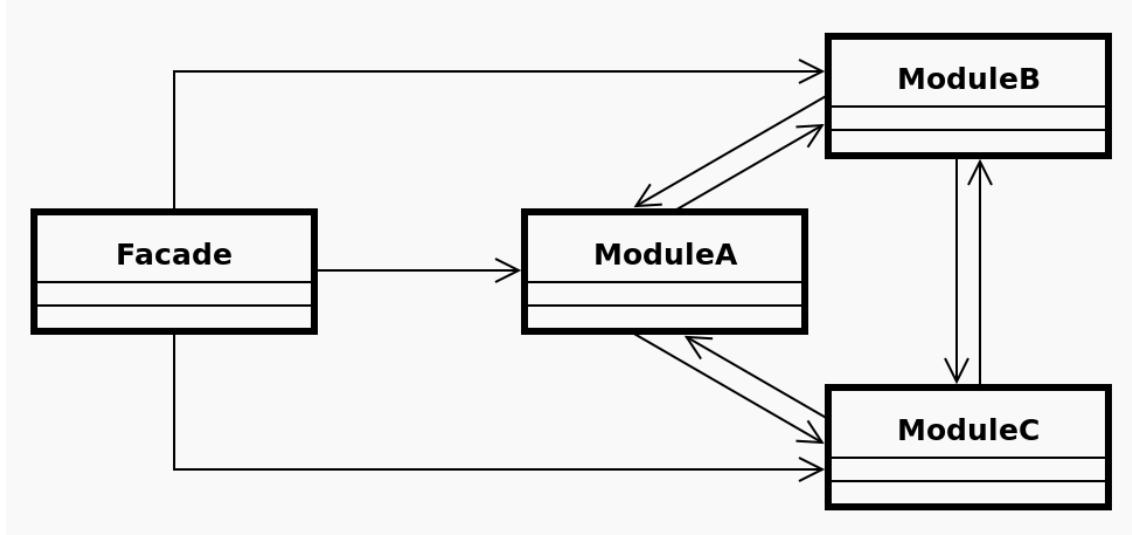
## Patrón Proxy

Un objeto sustituye a otro para controlar el acceso al mismo, generalmente por motivos de eficiencia.



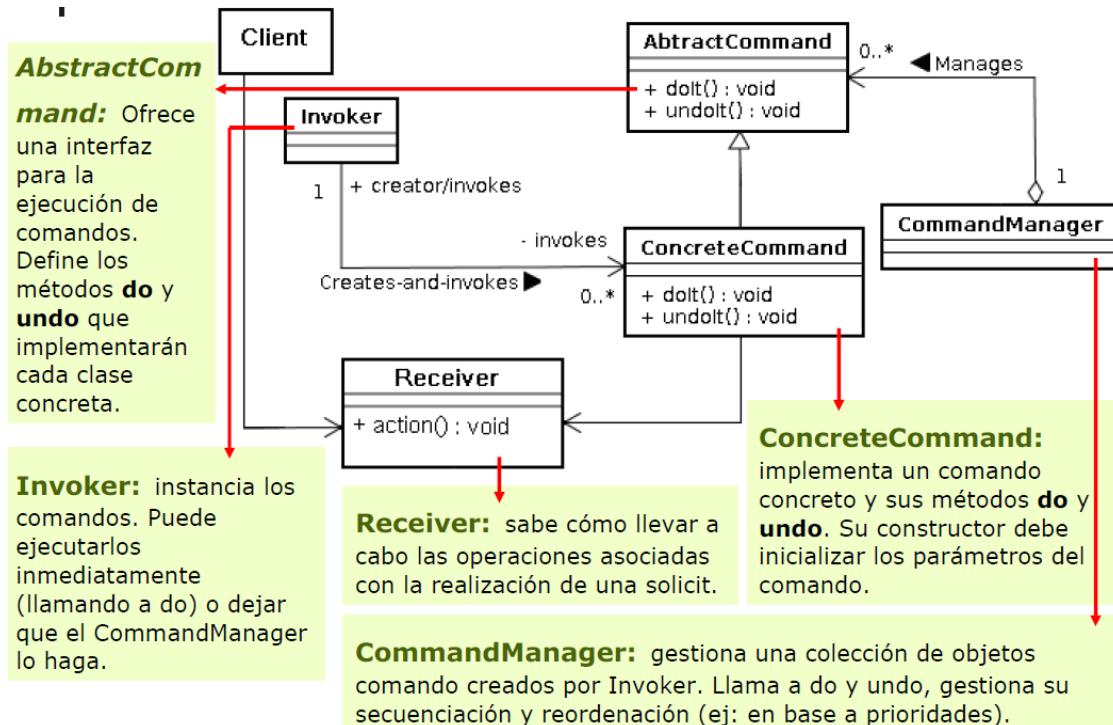
## Patrón Facade

El patrón fachada viene motivado por la necesidad de estructurar un entorno de programación y reducir su complejidad con la división en subsistemas, minimizando las comunicaciones y dependencias entre éstos.



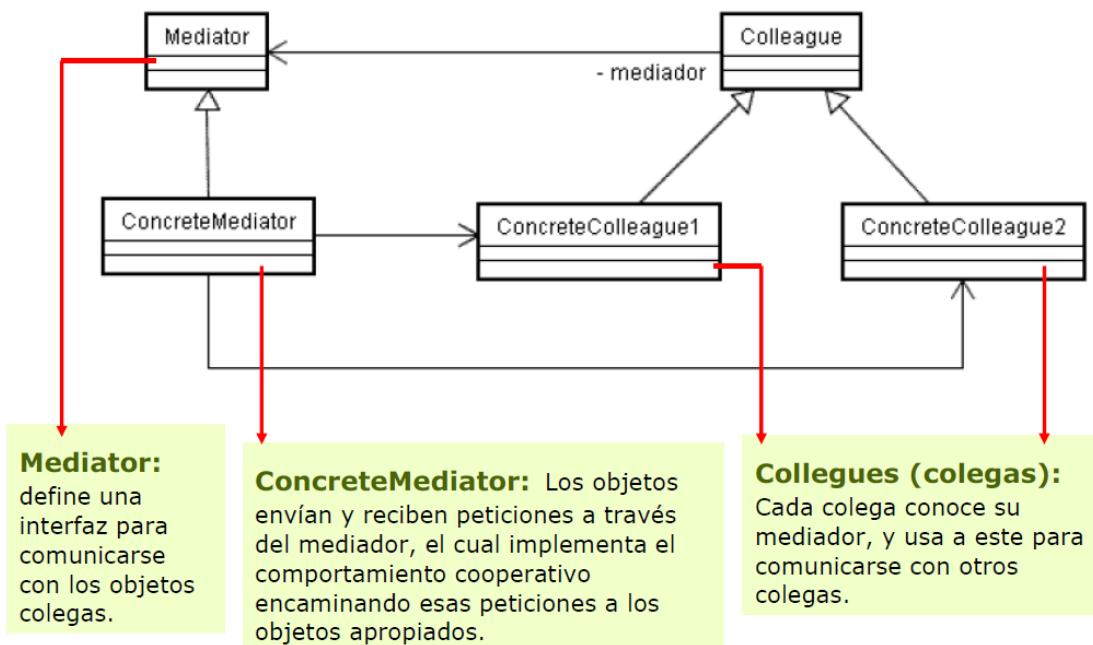
## Patrón Command

Su propósito es encapsular una solicitud como un objeto.



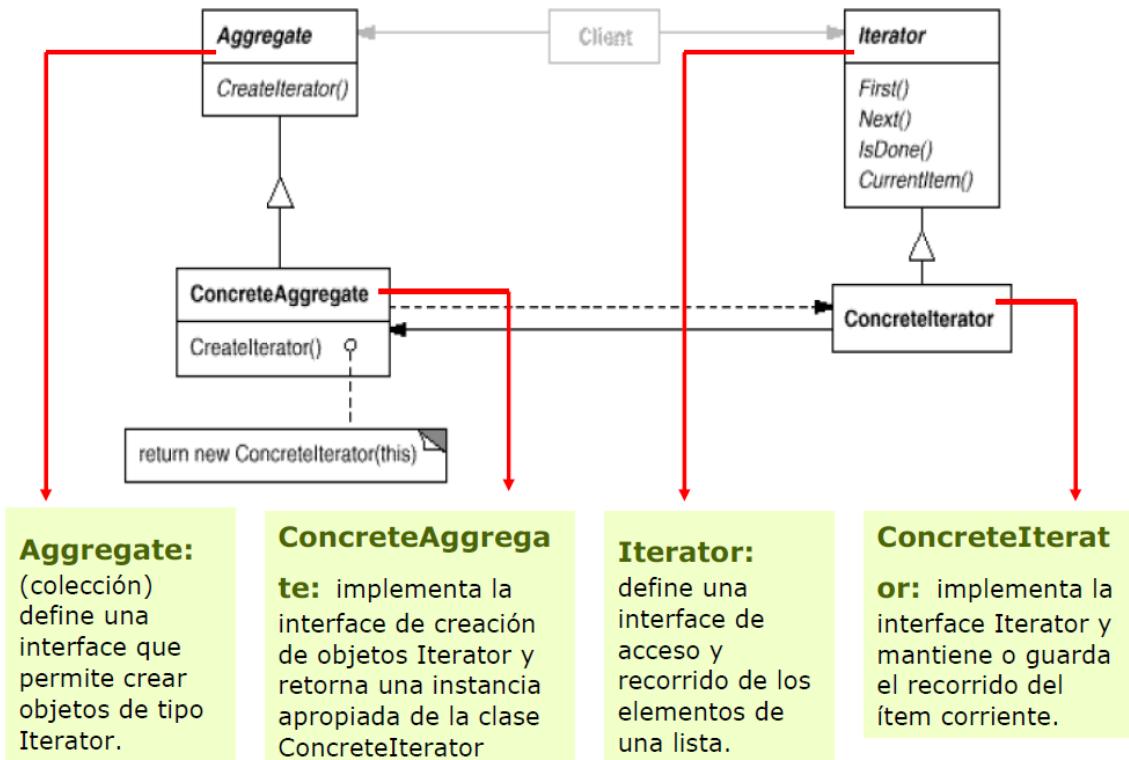
## Patrón Mediator

Define un objeto que encapsula cómo interactúan un conjunto de objetos. Promueve un bajo acoplamiento al evitar que los objetos se refieran unos a otros explícitamente, y permite variar la interacción entre ellos de forma independiente.



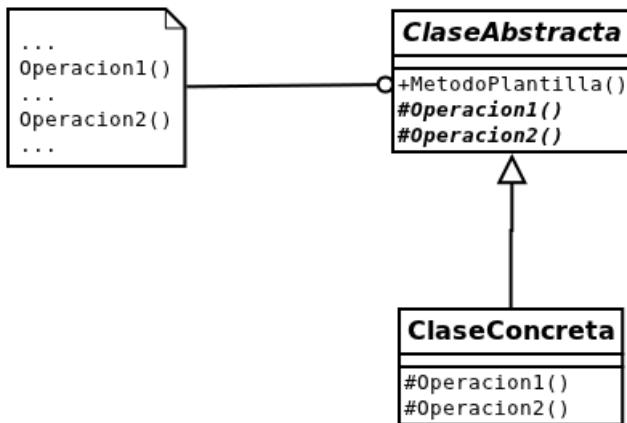
## Patrón Iterator

Provee un camino para acceder a los elementos de una colección secuencial de objetos sin exponer la representación interna de dicho recorrido.



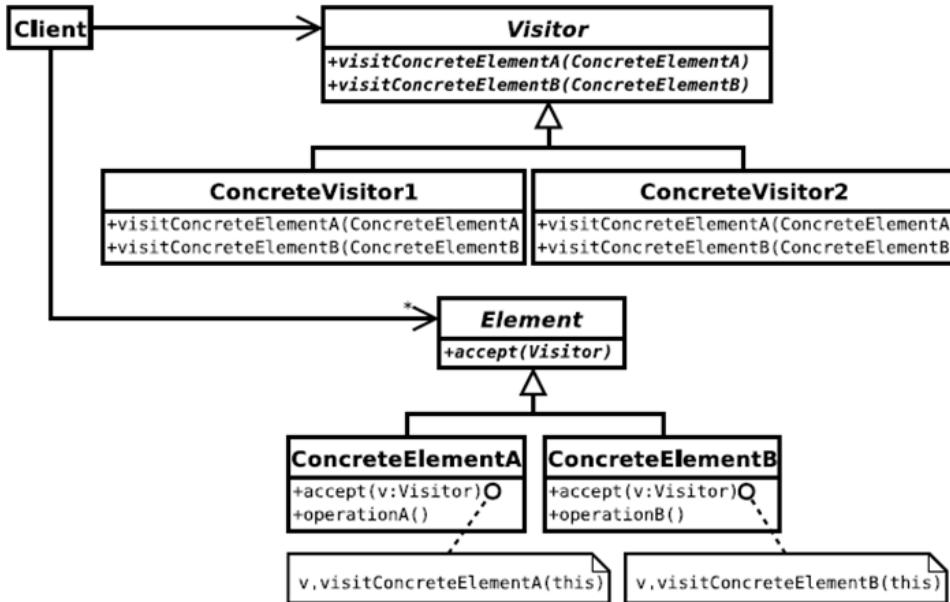
## Patrón Template Method

Permitir que ciertos pasos de un algoritmo definido en una operación de una superclase, sean redefinidos en las subclases sin necesidad de tener que sobrescribir la operación entera.



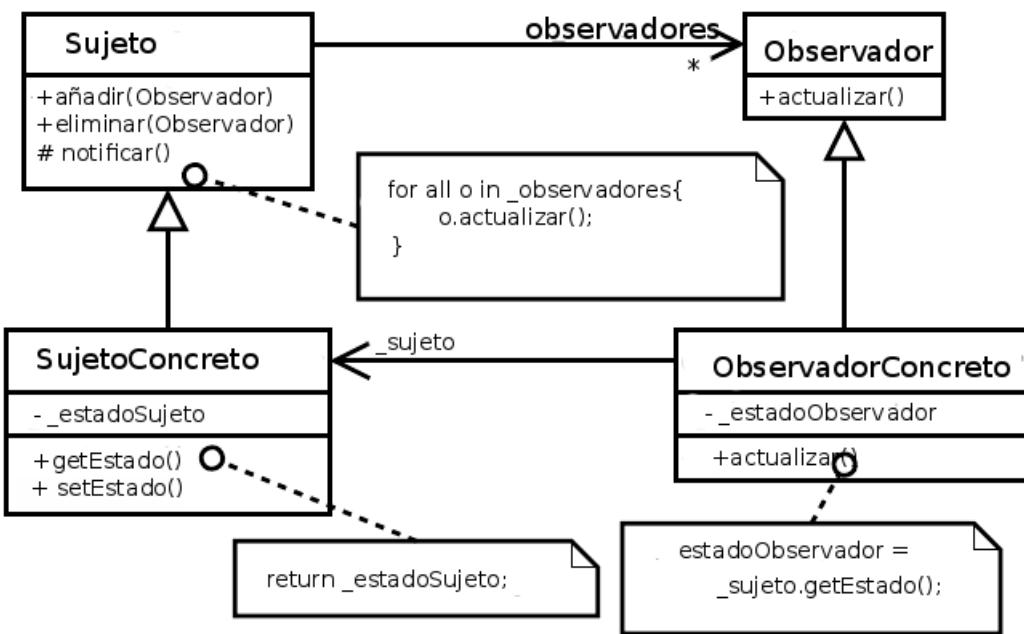
## Patrón Visitor

Es un patrón de comportamiento, que permite definir una operación sobre objetos de una jerarquía de clases sin modificar las clases sobre las que opera.



## Patrón Observer

Define una dependencia del tipo uno-a-muchos entre objetos, de manera que cuando uno de los objetos cambia su estado, notifica este cambio a todos los dependientes.



## 9.2 Anexo III Notación BPMN

### Notación para el Modelado de Procesos de Negocio

Notación para el Modelado de Procesos de Negocio o BPMN (en inglés Business Process Modeling Notation) es una notación gráfica estandarizada que permite el modelado de procesos de negocio, en un formato de flujo de trabajo (workflow). BPMN fue inicialmente desarrollada por la organización Business Process Management Initiative (BPMI), y es actualmente mantenida por el OMG (Object Management Group), después de la fusión de las dos organizaciones en el año 2005. Su versión actual, a abril de 2011, es la 2.0.

El principal objetivo de BPMN es proveer una notación estándar que sea fácilmente legible y entendible por parte de todos los involucrados e interesados del negocio (stakeholders). Entre estos interesados están los analistas de negocio (quienes definen y redefinen los procesos), los desarrolladores técnicos (responsables de implementar los procesos) y los gerentes y administradores del negocio (quienes monitorean y gestionan los procesos). En síntesis BPMN tiene la finalidad de servir como lenguaje común para cerrar la brecha de comunicación que frecuentemente se presenta entre el diseño de los procesos de negocio y su implementación.

Actualmente hay una amplia variedad de lenguajes, herramientas y metodologías para el modelamiento de procesos de negocio. La adopción cada vez mayor de la notación BPMN como estándar ayudará a unificar la expresión de conceptos básicos de procesos de negocio (por ejemplo procesos públicos y privados, orquestación, coreografía, etc.) así como conceptos avanzados de modelamiento (por ejemplo manejo de excepciones, compensación de transacciones, entre otros).

#### Ámbito de la BPMN.

BPMN está planeada para dar soporte únicamente a aquellos procesos que sean aplicables a procesos de negocios. Esto significa que cualquier otro tipo de modelado realizado por una organización con fines distintos a los del negocio no estará en el ámbito de BPMN. Por ejemplo, los siguientes tipos de modelado no estarían en el ámbito de BPMN:

- Estructuras organizacionales
- Descomposición funcional
- Modelos de datos

Adicionalmente, a pesar de que BPMN muestra el flujo de datos (mensajes) y la asociación de artefactos de datos con las actividades, no es de ningún modo un diagrama de flujo de datos.

#### Elementos.

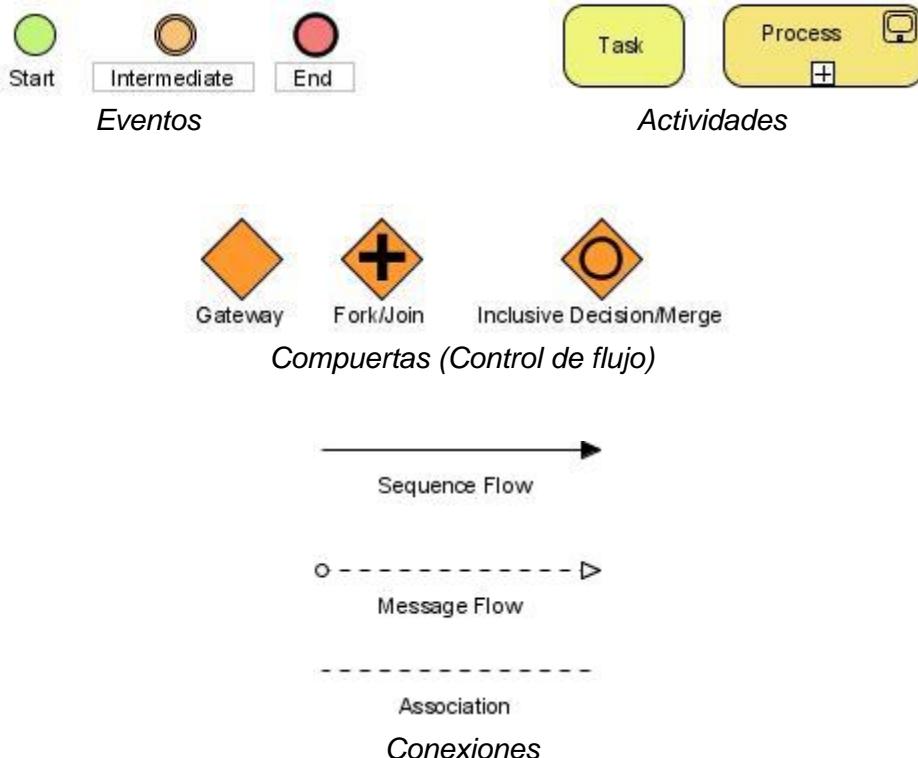
El modelamiento en BPMN se realiza mediante diagramas muy simples con un conjunto muy pequeño de elementos gráficos. Con esto se busca que para los usuarios del negocio y los desarrolladores técnicos sea fácil entender el flujo y el proceso. Las cuatro categorías básicas de elementos son:

- **Objetos de flujo:** Eventos, Actividades, Rombos de control de flujo (Gateways)
- **Objetos de conexión:** Flujo de Secuencia, Flujo de Mensaje, Asociación
- **Swimlanes (Carriles de piscina):** Pool, Lane
- **Artefactos:** Objetos de Datos, Grupo, Anotación

Estas cuatro categorías de elementos nos dan la oportunidad de realizar un diagrama simple de procesos de negocio (en inglés Business Process Diagram o BPD). En un BPD se permite definir un tipo personalizado de Objeto de Flujo o un Artefacto, si con ello se hace el diagrama más comprensible.

## Objetos de Flujo y Objetos de Conexión.

Los Objetos de Flujo son los elementos principales descritos dentro de BPMN y consta de tres elementos principales; Eventos, Actividades y Compuertas (Control de Flujo).



## Eventos

Están representados gráficamente por un círculo y describe algo que sucede (lo contrario de las Actividades que es algo que está hecho). Los eventos también pueden ser clasificados como Capturado o Lanzado.

- **Evento Inicial:** Actúa como un disparador de un proceso. Se representa gráficamente por un círculo de línea delgada y dentro del círculo está relleno de color verde. Este evento permite Capturar.
- **Evento Final:** Indica el final de un proceso. Está representado gráficamente por un círculo de línea gruesa y dentro del círculo está relleno del color rojo. Este evento permite Lanzar.
- **Evento intermedio:** Indica que algo sucede entre el evento inicial y el evento final. Está representado gráficamente por un círculo de doble línea simple y dentro del círculo relleno de color naranja. Este evento puede Capturar o Lanzar.

## Actividades

Están representadas por un rectángulo con sus vértices redondeados y describe el tipo de trabajo que será realizado.

- Tarea: Una tarea representa una sola unidad de trabajo que no es o no se puede dividir a un mayor nivel de detalle de procesos de negocio sin diagramación de los pasos de un procedimiento.
- Sub-proceso: Se utiliza para ocultar o mostrar otros niveles de detalle de procesos de negocio - cuando se minimiza un sub-proceso se indica con un signo más en contra de la línea inferior del rectángulo, cuando se expande el rectángulo redondeado permite mostrar todos los objetos de flujo, los objetos de conexión, y artefactos. Tiene, de forma auto-contenida, sus propios eventos de inicio y fin; y los flujos de proceso del proceso padre no deben cruzar la frontera.
- Transacción: Es una forma de sub-proceso en la cual, todas las actividades contenidas deben ser tratadas como un todo. Las transacciones se diferencian de los sub-procesos expandidos por estar rodeando por un borde de doble línea.

## **Compuertas (Control de Flujo)**

Están representadas por una figura de diamante y determina si se bifurca o se combina las rutas dependiendo de las condiciones expresadas.

Los objetos de flujo permitirán conectar cada uno de los objetos de conexión; en la cual, consta de tres tipos: Secuencias, Mensajes y Asociaciones.

## **Flujo de Secuencia**

Está representado por línea simple continua y flechada; y muestra el orden en que las actividades se llevarán a cabo. El flujo de secuencia puede tener un símbolo al inicio, un pequeño diamante indica uno de su número de flujos condicionales desde una actividad, mientras que una barra diagonal (slash) indica el flujo por defecto desde una decisión o actividad con flujos condicionales.

## **Flujo de mensaje**

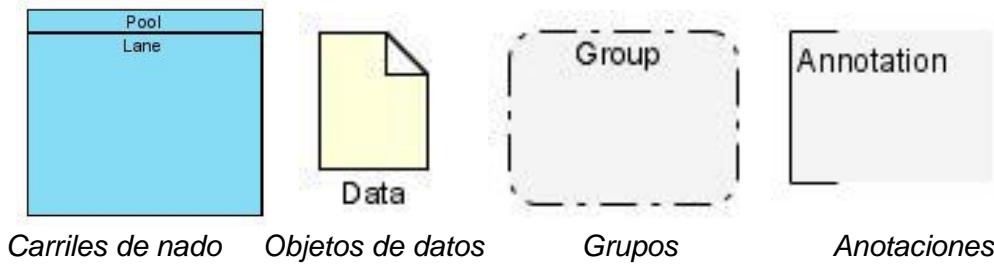
Está representado por una línea discontinua con un círculo no relleno al inicio y una punta de flecha no rellena al final. Esto nos dice, que el flujo de mensaje atraviesa la frontera organizacional (por ejemplo, entre piscinas). Un flujo de mensaje no puede ser utilizado para conectar actividades o eventos dentro de la misma piscina.

## **Asociaciones**

Están representadas por una línea punteada. Se suele usar para conectar artefactos o un texto a un objeto de flujo y puede indicar muchas direccionalidades usando una punta de flecha no rellena (hacia el artefacto para representar a un resultado, desde el artefacto para representar una entrada, y los dos para indicar que se lee y se actualiza). La No Direccionalidad podría usarse con el artefacto o un texto está asociado con una secuencia o flujo de mensaje (como el flujo muestra la dirección).

## **Carriles de Nado y Artefactos.**

Los Carriles de Nado son un mecanismo visual de actividades organizadas y categorizadas, basados en organigramas funcionales cruzados y en BPMN consta de dos tipos:



## Piscina

Representa los participantes principales de un proceso, por lo general, separados por las diferentes organizaciones. Una piscina contiene uno o más carriles (en la vida real, como una piscina olímpica). Una piscina puede ser abierta (por ejemplo, mostrar el detalle interno) cuando ella se presenta como un gran rectángulo que muestra uno o más carriles, o cerrada (por ejemplo, esconder el detalle interno) cuando ella se presenta como un rectángulo vacío se extiende el ancho o el alto del diagrama.

## Carril

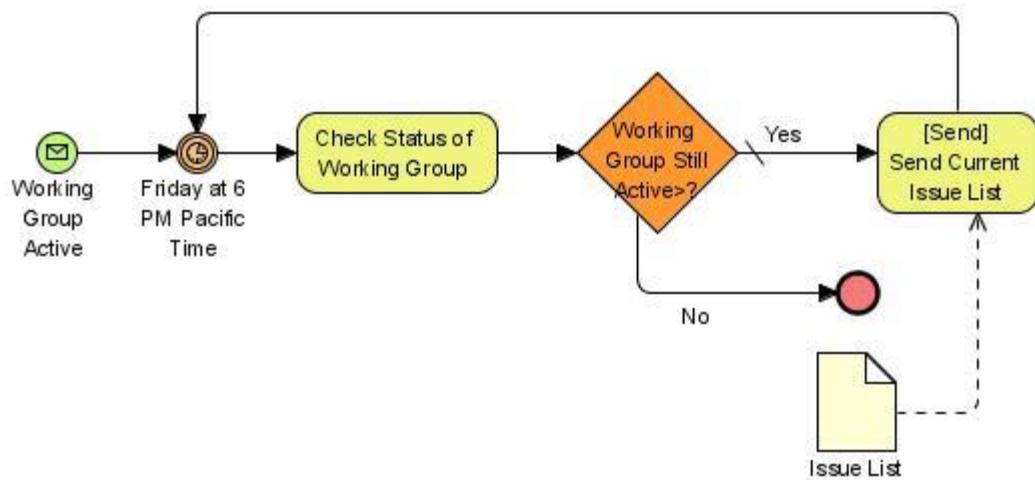
Usado para organizar y categorizar las actividades dentro de una piscina de acuerdo a su función o rol; y se presenta como un rectángulo estrecho de ancho o de alto de la piscina. Un carril contiene objetos de flujo, objetos de conexión y artefactos.

Los Artefactos permiten a los desarrolladores llevar algo más de información en el modelo o diagrama. De esta manera, el modelo o diagrama se hace más legible. Son tres artefactos pre-definidos y son:

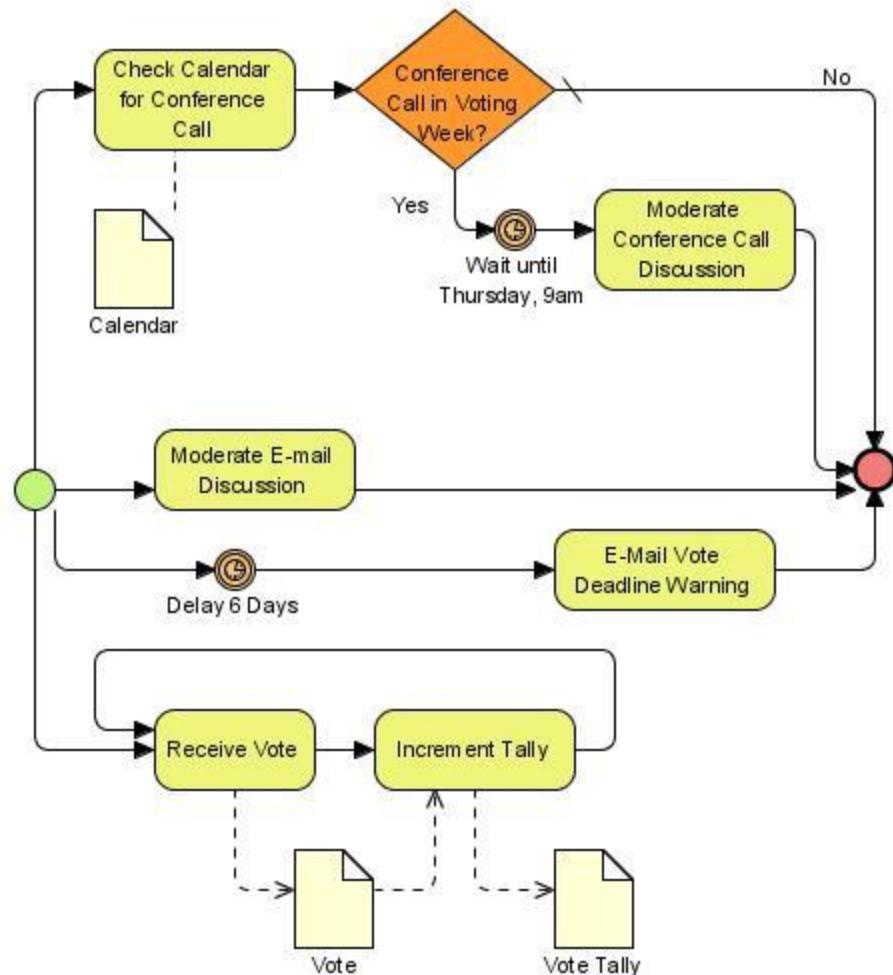
- **Objetos de Datos:** Muestra al lector cual es la data que deberá ser requerida o producida en una actividad.
- **Grupos:** Está representado por un rectángulo de líneas discontinuas y vértices redondeados. El Grupo es utilizado para agrupar diferentes actividades pero no afecta al flujo dentro de un diagrama.
- **Anotación:** Es utilizado para darle al lector una descripción entendible del modelo o diagrama

## Ejemplos de Diagramas de Procesos de Negocios.

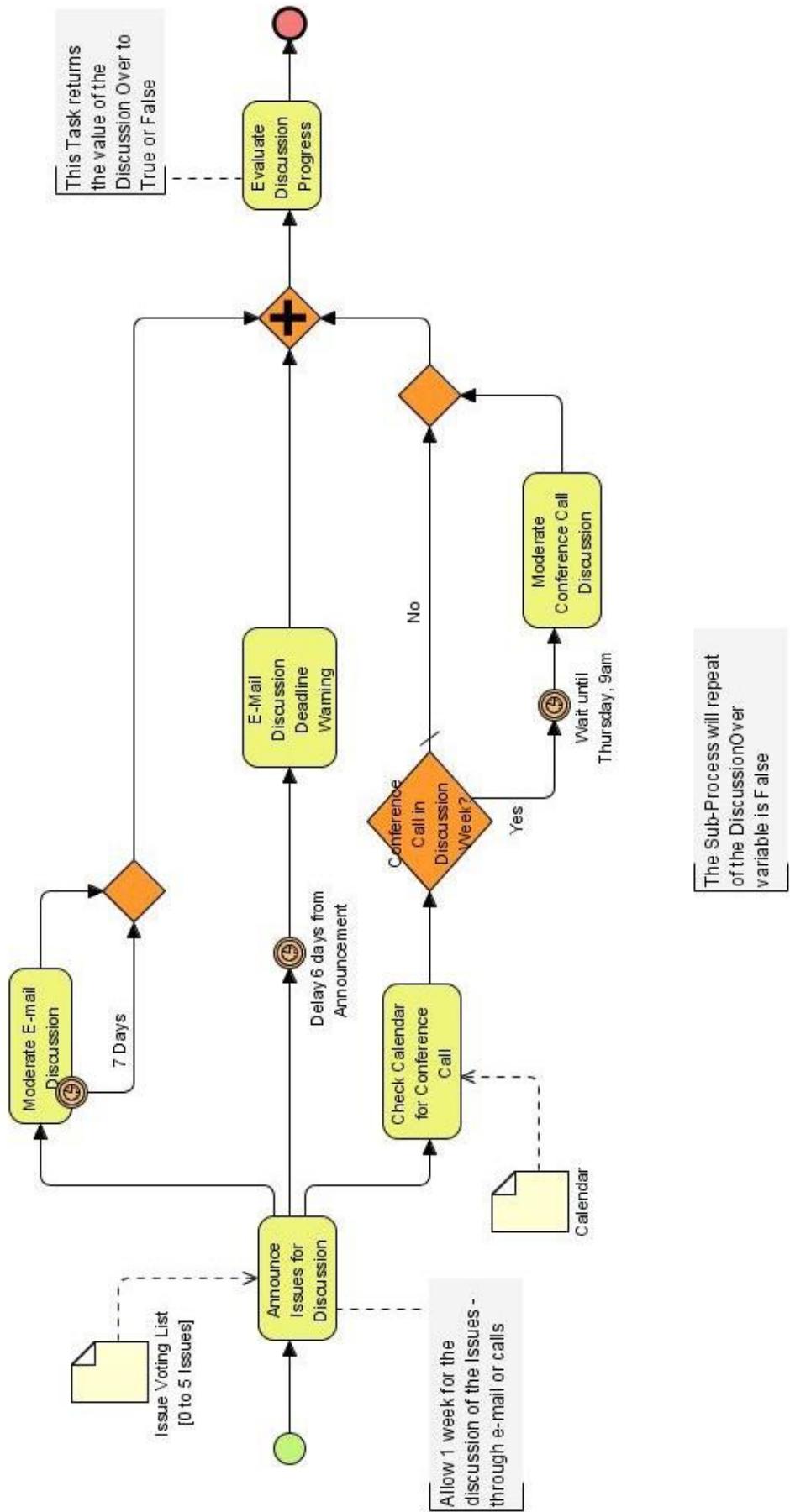
### Proceso con un flujo normal.



### Recolección de Votos:



### Ciclo de Discusión:



### 9.3 Anexo IV Plantilla Genérica para descripción de caso de uso

#### Plantilla Genérica

Esta plantilla sirve para establecer un patrón el cual se repite para ciertos casos de uso, en este caso, se utiliza para describir aquellos casos en los que podemos gestionar ciertos casos de uso. Por ejemplo: El caso de uso Gestión Cliente permite describir los casos de uso: Insertar Cliente, Modificar Cliente, Baja Cliente y Consulta Cliente con la siguiente plantilla.

	<u>Instancia: Plantilla 1 (Inserción de &lt;&gt;Elemento&gt;&gt;)</u>			<u>Instancia: Plantilla 3 (Eliminación de &lt;&gt;Elemento&gt;&gt;)</u>		<u>Instancia: Plantilla 2 (Modificación de &lt;&gt;Elemento&gt;&gt;)</u>		<u>Instancia: Plantilla 4 (Consultar &lt;&gt;Elemento&gt;&gt;)</u>	
ATRIBUTOS	CLAVE	VERIFICACION	ACCION	VERIFICACION	ACCION	VERIFICACION	ACCION	VERIFICACION	ACCION
Atr 1(id)	SI	Debe ser no nulo y único	Include (Generar Id)	Validar existencia		Validar existencia		Es un criterio de búsqueda, no nulo	
Atr 2									
Atr 3									
...									
Atr n									

En la primera columna vamos a tener listados los atributos, seguido de la columna Clave, esta se marcará cuando estemos en presencia del atributo clave, luego en la columna Verificación establecerá las verificaciones que se deben hacer para insertar, eliminar, modificar o consulta cierto elemento. Por último la comuna Acción, determina que acciones se llevan a cabo cuando se lleva a cabo el caso de uso que se esté tratando.