UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE CIENCIAS

SISTEMA DE OPORTUNIDADES INTERNAS PARA UNA ORGANIZACIÓN EMPRESARIAL

T E S I S

QUE PARA OBTENER EL TÍTULO DE :

LICENCIADA EN CIENCIAS DE LA COMPUTACIÓN

P R E S E N T A :

YASMINE MACEDO REZA

DIRECTOR DE TESIS: M. EN C. MARIA GUADALUPE ELENA IBARGÜENGOITIA GONZALEZ

MÉXICO, D.F. AGOSTO DEL 2006

A mi familia y a mi novio

Agradecimientos:

Para mis papás: Damián Macedo Aguilar y Bernardina Reza Melesio, que han sido el pilar en mi vida.

Para mi novio Efrén Arturo que me ha dado todo el apoyo, la compresión y el amor para la elaboración de este proyecto.

Para mis hermanas Olivia, Belinda, Deyanira, Anita y Betzayda que le han dado colorido a mi vida y por todo el cariño.

Para la maestra Lupita que sin su apoyo y disponibilidad no hubiera sido posible este trabajo.

Para todos aquellos buenos amigos que han estado presente en mi vida. Por todos los buenos momentos y el apoyo recibido.

ÍNDICE

Introducción

El software es un arma estratégica con el cual las empresas o los gobiernos pueden conseguir enormes reducciones de costos y tiempos de producción tanto para bienes como para servicios. Es imposible reaccionar con rapidez frente al dinamismo del mercado sin unos buenos procesos de organización establecidos. En el entorno de una economía global que opera las 24 horas del día, siete días de la semana, muchos de esos procesos no pueden funcionar sin el software. Un buen proceso de software es, por tanto, un elemento crítico para el éxito de cualquier organización.

El objetivo del presente trabajo es aplicar el Proceso Unificado a un problema, en particular se ha elegido el "Sistema de Oportunidades Internas para una Organización Empresarial" por tanto se desarrollaran las fases de captura de requisitos, análisis y diseño, las fases de implementación y prueba se dejan fuera del alcance de este proyecto.

En el presente trabajo se desarrollará un sistema siguiendo el Proceso Unificado de software:

Capítulo 1: En este capítulo se explicará el proceso unificado. El objetivo del Proceso Unificado es guiar a los desarrollares en la construcción eficiente de sistemas que se ajusten a las necesidades.

Capítulo 2: Se dará una introducción a la tecnología J2EE, describiendo sus principales componentes, y la utilización del Modelo Vista Controlador.

Capítulo 3: Descripción del problema. Se plantearán los requisitos de usuario para el sistema de Oportunidades Internas. El objetivo de este capítulo es guiar el desarrollo del sistema con la descripción de las funcionalidades detalladas del sistema.

Capítulo 4: Captura de Requisitos como casos de uso. Se identificarán los diferentes actores que interactúan con el sistema, y se describen los principales casos de uso. Los casos de uso proporcionan un medio intuitivo y sistemático para capturar los requisitos funcionales con un énfasis especial en el valor añadido para cada usuario individual o para cada sistema externo.

Capítulo 5: Análisis. Se analizarán los requisitos que se describieron en la captura de requisitos, refinándolos y estructurándolos. El objetivo de hacerlo es conseguir una compresión más precisa de los requisitos y una descripción de los mismos que sea fácil de mantener y que nos ayude a estructurar el sistema entero.

Capítulo 6: Diseño. Se modelarán las partes principales del sistema, y se encuentra la forma para que soporte los requisitos.

Finalmente se dan las conclusiones.

Capítulo 1

Proceso Unificado de Software

El problema del software está en a la dificultad que afrontan los desarrolladores para coordinar las múltiples cadenas de trabajo de todo proyecto de software. La comunidad de desarrolladores necesita un proceso que integre las múltiples facetas del desarrollo, un proceso que:

- Proporcione una guía para ordenar las actividades de un equipo.
- Dirija las tareas de cada desarrollador por separado y del equipo como un todo.
- Especifique los artefactos que deben desarrollarse.
- Ofrezcan criterios para el control y la medición de los productos y actividades del proyecto.

1.1 Proceso Unificado

El Proceso Unificado es un conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema de software.

El Proceso Unificado utiliza al Lenguaje de Modelado Unificado (Unified Modeling Language, UML) para preparar todos los modelos de un sistema de software. Un sistema de software ve la luz para dar servicios a sus usuarios. El término usuario no hace sólo referencia a usuarios humanos sino también a otros sistemas.

Una secuencia de acciones es una interacción con el usuario que se de denomina **caso de uso**. Un caso de uso es un fragmento de funcionalidad del sistema que proporciona al usuario un resultado importante. Los casos de usos representan los requisitos funcionales. Todos los casos de usos juntos constituyen el modelo de casos de uso, el cual describe la funcionalidad total del sistema, sin embargo los casos de usos no son sólo una herramienta para especificar los requisitos de un sistema, también guían su diseño, implementación y prueba; esto es, guían el proceso de desarrollo.

Por otro lado, se encuentra la arquitectura del sistema de software, que se describe mediante diferentes vistas del sistema en construcción.

El concepto de **arquitectura del software** incluye los aspectos estáticos y dinámicos más significativos de los sistemas. La arquitectura surge de las necesidades de la empresa, como las perciben los usuarios y los inversores, y se refleja en los casos de usos. La arquitectura es una vista del diseño completo con las características más importantes resaltadas dejando los detalles de lado.

El Proceso Unificado es **iterativo e incremental**. Las iteraciones hacen referencia a pasos en el flujo de trabajo, y los incrementos, al crecimiento del producto. Para una efectividad máxima las iteraciones deben estar controladas; esto es, deben seleccionarse y ejecutarse de una forma planificada.

En cada iteración, los desarrolladores identifican y especifican los casos de uso relevantes, crean un diseño utilizando la arquitectura seleccionada como guía, implementan el diseño mediante componentes, y verifican que los componentes satisfagan a los casos de uso.

La arquitectura proporciona la estructura sobre la cual se guian las iteraciones, mientras que los casos de usos definen los objetivos y dirigen el trabajo de cada iteración.

El Proceso Unificado se repite a lo largo de una serie de ciclos que constituyen la vida de un sistema. Cada ciclo concluye con una versión funcional del producto para los clientes.

Cada ciclo consta de cuatro fases: inicio, elaboración, construcción, transición. Cada fase se subdivide a su vez en iteraciones.

Durante la fase de **inicio**, se desarrolla una descripción del producto final a partir de una buena idea y se presenta el análisis de negocio para el producto.

Durante la fase de **elaboración**, se especifican en detalle la mayoría de los casos de uso del producto y se diseña la arquitectura del sistema.

Durante la fase de **construcción** se crea el producto, la línea de la base de la arquitectura crece hasta convertirse en el sistema completo. La descripción evoluciona hasta convertirse en un producto preparado para entregarse a la comunidad de usuarios.

La fase de **transición** cubre el periodo durante el cual el producto se convierte en versión beta. En la versión beta un número reducido de usuarios con experiencia prueba el producto e informa de defectos y deficiencias.

En la figura 1.1 se muestran los cinco flujos de trabajo que son: requisitos, análisis, diseño, implementación y prueba, y tienen lugar sobre las cuatro fases: inicio, elaboración, construcción y transición.

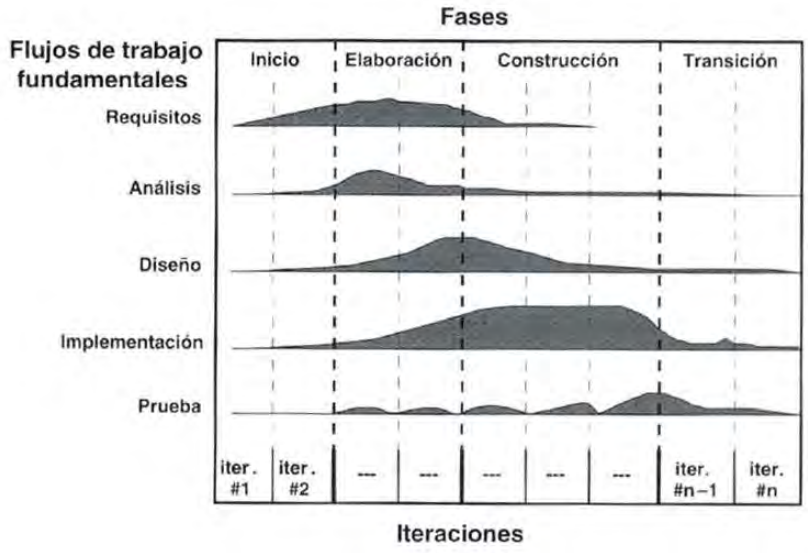


Figura 1.1 Ciclo de vida del software [Jacobson]

Cada una de las fases termina con un hito principal, el objetivo de cada hito principal es garantizar que los diferentes modelos de flujo de trabajo evolucionan de manera equilibrada durante el ciclo de vida del producto.

Aunque cada iteración discurre a lo largo de los flujos de trabajo de requisitos, análisis, diseño, implementación y pruebas, las iteraciones tienen énfasis diferentes en las distintas fases, como se muestra en la Figura 1.1 . Durante las fases de inicio y elaboración, la mayoría de esfuerzos se dedica a la captura de requisitos y a un

análisis y diseños preliminares. Durante la construcción el énfasis pasa al diseño detallado, la implementación y la prueba.

Al final de cada iteración los desarrolladores deben de iniciar un nuevo ciclo. Para llevar a cabo el siguiente ciclo de manera eficiente, se necesitan todas las representaciones del producto de software:

- Un modelo de casos de uso y su relación con los usuarios.
- Un modelo del análisis, con dos propósitos: refinar los casos de uso con más detalle y establecer la asignación inicial de funcionalidad del sistema a un conjunto de objetos que proporcionan el comportamiento.
- Un modelo de diseño que define (a) la estructura estática del sistema en la forma de subsistemas, clases e interfaces y (b) los casos de uso reflejados como colaboraciones entre los subsistemas, clases e interfaces.

1.2 Captura de Requisitos

Se llama **captura de requisitos** al acto de descubrir, lo que se necesita que haga el sistema. Es el proceso de averiguar, normalmente en circunstancias difíciles, lo que se debe de construir.

Los usuarios generalmente no saben cómo pueden hacer más eficiente la operación en su conjunto. La mayoría de los usuarios no sabe qué partes de su trabajo pueden transformarse en software. Francamente con frecuencia los usuarios no saben cuáles son los requisitos, ni tampoco cómo especificarlos en forma precisa.

El modelo de casos de uso se desarrolla a lo largo de varios incrementos del desarrollo, donde las iteraciones añadirán nuevos casos de uso y/o añadirán detalle de las descripciones de los casos de uso existentes.

- Durante la fase de inicio, los analistas identifican la mayoría de los casos de uso para delimitar el sistema y el alcance del proyecto y para detallar los más importantes.
- Durante la fase de elaboración, los analistas capturan la mayoría de los requisitos restantes para que los desarrolladores puedan estimar el tamaño del esfuerzo de desarrollo que se requerirá.
- Los requisitos restantes se capturan (e implementan) durante la fase de construcción.
- Casi no hay captura en la fase de transición, a menos que haya requisitos que cambien.

El propósito fundamental del flujo de trabajo de captura de requisitos es guiar el desarrollo hacia el sistema correcto. Esto se consigue mediante una descripción de los requisitos del sistema, suficientemente buena como para que pueda llegarse a un acuerdo entre el cliente y los desarrolladores sobre qué debe y qué no debe hacer el sistema.

Un reto fundamental para conseguirlo es que el cliente, que la mayoría de las veces se asume que será un especialista no informático, debe ser capaz de leer y comprender el resultado de la captura de requisitos. Para alcanzar este objetivo se debe utilizar el lenguaje del cliente para describir sus necesidades. Consecuentemente, se debería tener mucho cuidado en ser claros y consistentes escribiendo con formalidad y estructura, y cuando se incluyan detalles sobre el funcionamiento interno del sistema.

La posibilidad de tener puntos de partida tan dispares como una vaga noción y una especificación de requisitos detallada sugiere que los analistas necesitan ser capaces de adaptar sus técnicas a la captura de requisitos en cada situación.

Aparte de las diferencias en los puntos de partida, ciertos pasos son factibles en la mayoría de los casos:

- Enumerar los requisitos candidatos.
- Comprender el contexto del sistema.
- Capturar requisitos funcionales.
- Capturar requisitos no funcionales.

A continuación se describen brevemente cada uno se estos pasos:

- **Enumerar los requisitos candidatos**

Durante la vida del sistema, los clientes, usuarios, analistas y desarrolladores aparecen con muchas buenas ideas que podrían convertirse en verdaderos requisitos. Esta lista de características crece a medida que se añaden nuevos elementos y mengua cuando algunas características se convierten en requisitos y se transforman en artefactos como casos de uso.

Cada característica debe tener un nombre corto y una breve explicación o definición, información suficiente para poder hablar de ella durante la planificación del producto, también tiene un conjunto de valores de planificación:

- Estado (propuesto, aprobado, incluido o validado).
- Coste estimado de implementación (en términos de tipos de recursos y horas-persona).
- Prioridad (critico, importante o secundario).
- Nivel de riesgo asociado a la implementación de la característica (critico, significativo u ordinario).

- **Comprender el contexto del sistema**

Hay por lo menos dos aproximaciones para expresar el contexto de un sistema en una forma utilizable para desarrolladores de software: modelado del dominio y modelado del negocio. El modelado del dominio describe los conceptos importantes del contexto como objetos del dominio, y enlaza estos objetos unos con otros. El modelo del negocio puede describirse como un supraconjunto del modelo del dominio, e incluye las reglas del negocio.

El arquitecto y el jefe del proyecto juntos deciden si se prepara un modelo del dominio, si se recorre todo el camino y se prepara un modelo del negocio entero, o si no se hace ninguno de esos modelos.

- **Capturar requisitos funcionales**

La técnica inmediata para identificar los requisitos del sistema se basa en los casos de uso. Estos casos de usos capturan tantos los requisitos funcionales como los no funcionales que son específicos en cada caso de uso.

- **Captura de requisitos no funcionales**

Los requisitos no funcionales especifican propiedades del sistema, como restricciones del entorno o de la implementación, rendimientos, dependencias de la plataforma, facilidad de mantenimiento, extensibilidad y fiabilidad.

Debido a que los requisitos cambian constantemente, se necesita alguna forma de actualizarlos de manera controlada. Se hace en las iteraciones, donde cada iteración reflejará algún cambio en el conjunto de requisitos, pero el número de cambios normalmente disminuirá a medida que adentra en la fase de construcción y a medida que se estabilicen los requisitos.

Mediante la utilización del modelo del negocio como entrada, un analista emplea una técnica sistemática para crear un modelo de casos de uso tentativo.

El analista, identifica un actor por cada trabajador y por cada actor del negocio que se convertirá en el usuario del sistema de información.

La forma mas directa de identificar un conjunto tentativo de casos de uso es crear un caso de uso para el actor correspondiente a cada rol de cada trabajador y de cada actor del negocio.

Los requisitos adicionales son fundamentalmente requisitos no funcionales que no pueden asociarse a ningún caso de uso en concreto –en cambio cada uno de estos requisitos tiene impacto en varios casos de uso o en ninguno--. Los requisitos adicionales se capturan de forma muy parecida a como se hacía en la especificación de requisitos tradicional. Después se utilizan durante el análisis y el diseño junto al modelo de casos de uso.

- Requisito de interfaz. Especifica la interfaz como el elemento externo con el cual debe interactuar el sistema.
- Requisito físico especifica una característica física que debe poseer el sistema, como su material, forma, tamaño o peso.
- Restricción de diseño. Limita el diseño de un sistema, como lo hacen las restricciones de extensibilidad y mantenibilidad, o las restricciones relativas a la reutilización de sistemas heredados o partes esenciales de los mismos.
- Restricción de implementación especifica o limita la codificación o construcción de un sistema.

¿Qué es un modelo del dominio?

Un modelo del 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.

Muchos de los objetos del dominio o clases pueden obtenerse de una especificación de requisitos o mediante la entrevista con expertos del dominio. Las clases del dominio aparecen en tres formas típicas:

- Objetos del negocio que representan cosas que se manipulan en el negocio, como pedidos, cuenta y contratos.
- Objetos del mundo real y conceptos de los que el sistema debe hacer un seguimiento, como la aviación enemiga, misiles y trayectorias.
- Sucesos que ocurrirán o han ocurrido.

El modelado del dominio se describe mediante diagramas UML (especialmente mediante diagramas de clases)

¿Qué es un modelo del negocio?

Un modelo de casos de uso del negocio describe los procesos de negocio de una empresa en términos de casos de uso del negocio y actores del negocio que se corresponden con los procesos del negocio y los clientes, respectivamente. Al igual que el modelo de casos de uso para un sistema de software, el modelo de casos de uso del negocio presenta un sistema desde la perspectiva de su uso, y esquematiza como proporciona valor a sus usuarios.

Un modelo de objetos del negocio es un modelo interno a un negocio. Describe como cada caso de uso de negocio es llevado a cabo por parte de un conjunto de trabajadores que utilizan un conjunto entidades del negocio y de unidades de trabajo.

El modelo del negocio se desarrolla, por tanto, en dos pasos:

1. Los modeladores del negocio deben confeccionar un modelo de casos de uso del negocio que identifique los actores del negocio y los casos de uso del negocio que realizan los actores.
2. Los modeladores deben desarrollar un modelo de objetos del negocio compuesto por trabajadores, entidades del negocio, unidades de trabajo que juntos realizan los casos de uso del negocio.

Mediante la utilización del modelo del negocio como entrada, un analista emplea una técnica sistemática para crear un modelo de casos de uso tentativo.

1.3 Captura de requisitos como casos de uso

La fase de requisitos es para desarrollar un modelo del sistema que se va a construir, y la utilización de los casos de uso es una forma adecuada de crear ese modelo.

Los casos de uso proporcionan un medio intuitivo y sistemático para capturar los requisitos funcionales con énfasis especial en el valor añadido para cada usuario individual o para cada sistema externo.

1.3.1 Artefactos

Los artefactos fundamentales que se utilizan en la captura de requisitos son el modelo de casos de uso, que incluye los casos de uso y los actores. También puede haber otros tipos de artefactos, como prototipos de interfaz de usuario.

El modelo de casos de uso sirve como acuerdo entre clientes y desarrolladores, y proporciona la entrada fundamental para el análisis, el diseño y las pruebas.

El modelo de casos de uso describe lo que hace el sistema para cada tipo de usuario.

Cada uno de éstos se representa mediante uno o más actores. Una vez que hemos identificado todos los actores del sistema, tenemos identificado el entorno externo del sistema.

Un actor juega un papel por cada caso de uso con el que interacciona. Una instancia de un actor es un usuario concreto que interactúa con el sistema.

Cada forma en que los actores usan el sistema se representa con un caso de uso. Los casos de uso son "fragmentos" de funcionalidad que el sistema ofrece para aportar un resultado de valor para sus actores.

Por tanto, un caso de uso es una especificación del comportamiento de "cosas" dinámicas, en este caso, de instancias de casos de uso.

El flujo de sucesos para cada caso de uso puede plasmarse como una descripción textual de la secuencia de acciones del caso de uso.

Se llaman requisitos especiales a la descripción textual que agrupa todos los requisitos del tipo de los requisitos no funcionales, sobre el caso de uso y que deben tratarse en flujos de trabajo posteriores como análisis, diseño o implementación.

La vista de arquitectura del modelo de casos de usos debería incluir los casos de uso que describan una funcionalidad importante y crítica, o que se apliquen a algún requisito importante que deba desarrollarse pronto dentro del ciclo de vida de software.

Podemos utilizar un *glosario* para definir términos comunes importantes que los analistas utilizan al describir el sistema. Un glosario es muy útil para alcanzar un consenso entre los desarrolladores relativo a la definición de los diversos conceptos y nociones, y para reducir en general el riesgo de confusiones.

Los prototipos de interfaz de usuario nos ayudan a comprender y especificar las iteraciones entres actores humanos y el sistema durante la captura de requisitos.

1.3.2 Trabajadores

El paso siguiente es examinar los trabajadores responsables de esos artefactos, un trabajador es un puesto al cual se puede asignar una persona "real". Con cada trabajador tenemos una descripción de las responsabilidades y el comportamiento esperado del mismo.

El *analista del sistema* es el responsable del conjunto de requisitos que están modelados en los casos de uso, lo que incluye todos los requisitos funcionales y no funcionales que son casos de uso específicos. El analista de sistemas es responsable de delimitar el sistema, encontrando los actores y los casos de uso y asegurado que el modelo de casos de uso es completo y consistente.

El analista de sistemas esta asistido por otros trabajadores que asumen las responsabilidades de las descripciones detalladas de uno o más casos de uso. Esos trabajadores se denominan *especificadores de casos de uso.*

Los *diseñadores de interfaces* de usuario dan forma visual a las interfaces de usuario. Esto puede implicar el desarrollo de prototipos de interfaces de usuario para algunos casos de uso, habitualmente un prototipo para cada actor.

El *arquitecto* participa en el flujo de trabajo de los requisitos para describir la vista de la arquitectura del modelo de casos de uso.

1.3.3 Flujo de trabajo

Se describirá los flujos de trabajo como una secuencia de actividades que están ordenadas, así que una actividad produce una salida que sirve de entrada a la siguiente actividad.

Se describen a continuación las actividades que típicamente aparecen en la iteración de elaboración:

1. Actividad: encontrar actores y casos de uso

Esta actividad costa de cuatro pasos:

- **Encontrar los actores**. La tarea de encontrar a los actores depende del punto de partida. Si se tiene un modelo del negocio del cual partir, encontrar los actores resulta sencillo. En otro caso, el analista del sistema, junto con el cliente, identifica los usuarios e intenta organizarlos en categorías representadas por actores. Posteriormente da nombre a los actores y describe brevemente los papeles de cada actor y para qué utilizará el sistema el actor. Estos actores pueden utilizarse ahora como punto de partida para encontrar los casos de uso.

- **Encontrar los casos de uso**. El actor necesitará normalmente casos de uso para soportar su trabajo de creación, cambio, rastreo, eliminación o estudio de los objetos del negocio. Se elige un nombre para cada caso de uso de forma que nos haga pensar en la secuencia de acciones concreta que añade valor a un actor. El nombre de un caso de uso a menudo comienza con un verbo, y debe reflejar cuál es el objetivo de la interacción entre el actor y el sistema. Un caso de uso entrega un resultado que se puede observar y que añade valor a un actor en concreto. Obsérvese que hay dos frases claves en estas directrices que constituyen criterio útiles para la identificación de casos de uso, *resultado de valo*r y un *actor en concreto.*

- **Describir brevemente cada caso de uso**. Se describe brevemente cada caso de uso, en primer lugar con algunas frases que resumen las acciones y más tarde, con una descripción paso a paso de lo que el sistema necesita hacer cuando interactúa con sus actores.

- **Describir el modelo de los casos de uso completo**. Se preparan diagramas y descripciones para explicar el modelo de casos de uso en su totalidad.

Cuando la descripción del modelo de casos de uso esté preparada, se da el visto bueno del modelo de casos de uso la gente que no forma parte del equipo de desarrollo, determinan si:

- Se han capturado como casos de uso todos los requisitos funcionales necesarios.

- La secuencia de acciones es correcta, completa y comprensible para cada caso e uso.
- Se identifica algún caso de uso que no proporcione valor. Si es así, ese caso de uso debería reconsiderarse.

2. Actividad: priorizar casos de uso

El propósito de esta actividad es priorizar los casos de uso para determinar cuales son necesarios para el desarrollo, en las primeras iteraciones, y cuáles pueden dejarse para más tarde.

3. Actividad: detallar los casos de uso

Se detalla paso a paso cada caso de uso en una especificación precisa de la secuencia de acciones. El resultado de esta actividad es la descripción detallada del caso de uso en forma de textos y diagramas.

- Estructuración de la descripción de casos de uso.

  ¿Qué incluir en una descripción de caso de uso?
  - La descripción de un caso de uso debe definir el estado inicial, como precondición.
  - Cómo y cuando comienza el caso de uso.
  - El orden requerido en el que las acciones se deben ejecutar, en caso de haberlo. Se define como una secuencia numerada.
  - Cómo y cuando termina el caso de uso.
  - Una descripción de caso de uso debe definir los posibles estados finales como postcondiciones.
  - Los caminos de ejecución que no están permitidos.
  - Las descripciones de caminos alternativos que están incluidos junto con la descripción del camino básico.
  - Las descripciones de los caminos alternativos que han sido extraídas de la descripción del camino básico.
  - La interacción del sistema con los actores y qué cambios producen.
  - La utilización de objetos, valores y recursos de sistema. Dicho de otra forma, hemos descrito la secuencia de acciones en la utilización de un caso de uso, y hemos asignado valores a los atributos de los casos de uso.
  - Se debe describir explícitamente que qué hace el sistema. Debemos ser capaces de separar las responsabilidades del sistema y la de los actores, si no la descripción del caso de uso no será suficientemente precisa para poder utilizarla como especificación del sistema.

  La forma de plantear cada uno de los casos de uso será la siguiente:

  - *Nombre del Caso de Uso:* El nombre que se le da a la secuencias de pasos a ejecutar.

  - *Actor:* El actor que utiliza este caso de uso.

  - *Descripción:* Se detalla lo que se desea obtener por parte del sistema después de interactuar con el actor.

- *Precondiciones:* El estado actual en el que se encuentra el sistema, a partir del cual se desarrollará el caso de uso.

- *Flujo:* Se enumera en una tabla paso a paso las acciones del actor y las del sistema para completar el caso de uso.

- *Excepciones:* Se enumeran las posibles excepciones para el caso de uso dado.

- *Postcondiciones:* Se describe el estado en el que sistema se encuentra después de haber realizado el caso de uso.

4. Actividad: prototipar la interfaz de usuario.

Se necesita diseñar la interfaz de usuario que permita al usuario llevar a cabo los casos de uso de manera eficiente. Se comienza con los casos de uso y se intenta discernir qué se necesita de las interfaces de usuarios para habilitar los casos de uso para cada actor. Esto es, se hace un diseño lógico de la interfaz de usuario. Después se crea el diseño físico de la interfaz del usuario y desarrollamos prototipos que ilustren cómo pueden utilizar el sistema los usuarios para ejecutar los casos de uso.

- Crear el diseño lógico de una interfaz de usuario

El diseñador de interfaces de usuario identifica y especifica los elementos de la interfaz actor por actor, recorriendo todos los casos de uso a los que el actor puede acceder, e identificando los elementos apropiados de la interfaz de usuario para cada caso de uso.

Se debería responder a las siguientes preguntas para cada actor:

- ¿Qué elementos de la interfaz de usuario se necesitan para posibilitar el caso de uso?
- ¿Cómo deberían relacionarse unos con otros?
- ¿Cómo se utilizarán en los diferentes casos de uso?
- ¿cuál debería ser su apariencia?
- ¿Cómo deberían manipularse?

Para determinar qué elementos de interfaz de usuario necesitan ser accesibles al actor en cada caso de uso, podemos contestar las siguientes preguntas:

- ¿Qué clases del dominio, entidades del negocio o unidades de trabajo son más adecuadas como elementos de la interfaz de usuario para cada caso de uso?
- ¿Con qué elementos de la interfaz de usuario va a trabajar el actor?
- ¿Qué acciones puede invocar el actor, y qué decisiones puede tomar?
- ¿Qué guía o información va necesitar el actor antes de invocar cada acción de los casos de uso?
- ¿Qué información debe proporcionar el actor al sistema?
- ¿Qué información debe proporcionar el sistema al actor?
- ¿Cuáles son los valores medios de todos los parámetros de entrada o salida?

- Creación del diseño y prototipos físicos de interfaz de usuario

Puede haber varios prototipos, quizá uno para cada actor, para verificar que cada actor puede ejecutar el caso de uso que necesita.

La validación de las interfaces de usuarios a través de revisiones de prototipo y esquemas en los primeros momentos pueden prevenir mucho errores que serán más caros de corregir después. Los prototipos también pueden revisarse superficialmente en la descripción de casos de uso, y permitir que se corrijan después de que los casos de uso pasen a su diseño. Los revisores deben verificar que cada interfaz de usuario:

- Permite que el actor navegue de forma adecuada.
- Proporciona una apariencia agradable y una forma consistente de trabajo con la interfaz de usuario.
- Cumple con estándares relevantes como color, tamaño de los botones y situación de las barras de herramientas.

1.4 Análisis

Durante el análisis, se revisan los requisitos que se describieron en la captura de requisitos, refinándolos y estructurándolos. El objetivo de hacerlo es conseguir una compresión más precisa de los requisitos y una descripción de los mismos que sea fácil de mantener y que ayude a estructurar el sistema entero.

El propósito fundamental del análisis es revisar los requisitos con mayor profundidad, pero con la diferencia de que puede utilizarse el lenguaje de los desarrolladores para describir los resultados.

En consecuencia, en el análisis se pueden revisar más los aspectos internos del sistema, y por tanto resolver aspectos relativos a la interferencia entre casos de uso. También se puede utilizar un lenguaje más formal para apuntar detalles relativos a los requisitos del sistema.

El lenguaje que se utiliza en el análisis se basa en un modelo de objetos conceptual, que se llama modelo de análisis, que ayuda a refinar los requisitos y permite razonar sobre los aspectos internos del sistema, incluidos sus recursos compartidos internos.

El modelo de análisis puede considerarse una primera aproximación al modelo de diseño. Mediante la conservación de estructura del modelo de análisis en el diseño, se obtiene un sistema que debería ser mantenible como un todo, será flexible a los cambios en los requisitos, e incluirá elementos que podrán ser reutilizados cuando se construyan sistemas parecidos.

Analizar los requisitos en la forma de un modelo de análisis es importante por varios motivos:

- El modelo de análisis ofrece una especificación más precisa de los requisitos que la que se tiene como resultado de la captura de requisitos, incluyendo a modelo de casos de uso.

- El modelo de análisis se describe, utilizando el lenguaje de los desarrolladores, y puede por tanto introducir un mayor formalismo y ser utilizado para razonar sobre los funcionamientos internos del sistema.
- El modelo de análisis estructura los requisitos de un modo que facilita su compresión, su preparación, su modificación, y en general, su mantenimiento.
- El modelo de análisis puede considerarse como una primera aproximación al modelo del diseño, y es por tanto una entrada fundamental cuando se da forma al sistema en el diseño y en la implementación.

Las iteraciones iniciales de la elaboración se centran en el análisis. Eso contribuyen a obtener una arquitectura estable y sólida y facilita la comprensión en profundidad de los requisitos.

1.4.1 Artefactos

- Modelo de análisis

La estructura impuesta por el modelo de análisis se define mediante una jerarquía. El modelo de análisis se representa mediante un sistema de análisis que denota el paquete de más alto nivel del modelo. La utilización de otros paquetes de análisis es por tanto una forma de organizar el modelo de análisis en partes más manejables que representan abstracciones de subsistemas y posiblemente capas completas del diseño del sistema. Dentro del modelo de análisis, los casos de uso se describen mediante clases de análisis y sus objetos.

- Clase del 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:

- Las clases de análisis siempre encajan en uno de tres estereotipos básicos: de interfaz, de control o de entidad.
- Se centra en el tratamiento de los requisitos funcionales y pospone los no funcionales.
- Define atributos.
- Participa en relaciones, aunque esas relaciones son más conceptuales que sus contrapartidas de diseño e implementación.

Las clases de interfaz se utilizan para modelar la interacción entre el sistema y sus actores. Esta interacción a menudo implica recibir información y peticiones de los usuarios y sistemas externos. Modelan las partes del sistema que depende de sus actores, lo cual implica que clarifican y reúnen los requisitos en los límites del sistema. Representan a menudo abstracciones de ventanas, formularios, paneles, interfaces de comunicaciones, interfaces de impresoras, sensores, terminales, y API. Cada clase de interfaz debería asociarse con al menos un actor, y viceversa.

Las clases de entidad se utilizan para modelar información que posee una vida larga y que es a menudo persistente. Las clases de entidad modelan la información y el comportamiento asociado de algún fenómeno o concepto, como una persona, un objeto del mundo real, o un suceso del mundo real. Suelen mostrar una estructura de datos lógica y contribuyen a comprender de qué información depende el sistema.

Las clases de control representan coordinación, secuencia, transacciones, y control de otros objetos y se usan con frecuencia para encapsular el control de un caso de uso en concreto. También se utilizan para representar derivaciones y cálculos complejos, como la lógica del negocio, que no pueden asociarse con ninguna información concreta, de larga duración, almacenada por el sistema.

- Realización de caso de uso del análisis

Una realización de caso de uso del análisis es una colaboración dentro del modelo de análisis que describe cómo se lleva a cabo y se ejecuta un caso de uso determinado en términos de las clases del análisis y de sus objetos.

Una realización de casos de uso posee una descripción textual del flujo de sucesos, diagramas de clases que muestran sus clases del análisis participantes, y diagramas de interacción que muestran la realización de un flujo o escenario particular del caso de uso en términos de interacciones de objetos del análisis.

Diagramas de clases. Una clase de análisis y sus objetos normalmente participan en varias realizaciones de casos de uso. Durante el análisis se coordinan todos los requisitos sobre una clase y sus objetos que pueden tener diferentes casos de uso. Para hacerlo, se juntan en diagramas de clases a las realizaciones de casos de uso, mostrando las clases participantes y sus relaciones.

Diagramas de interacción. La secuencia de acciones en un caso de uso comienza cuando un actor invoca el caso de uso mediante el envío de algún tipo de mensaje al sistema. Si consideramos el "interior" del sistema, un objeto de interfaz recibirá este mensaje del actor. El objeto de interfaz enviará a su vez, un mensaje a algún otro objeto, y de esta forma los objetos implicados interactuarán para llevar a cabo el caso de uso. En el análisis se prefiere modelar con diagramas de colaboración ya que el objetivo fundamental es identificar requisitos y responsabilidades sobre los objetos, y no identificar secuencias de interacción detalladas y ordenadas cronológicamente.

En los diagramas de colaboración, se muestran las interacciones entre objetos creando enlaces entre ellos y añadiendo mensajes a estos enlaces. El nombre del mensaje debería denotar el propósito del objeto invocante en la interacción con el objeto invocado.

- Paquete del análisis

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

- Paquetes de servicio

Aparte de proporcionar casos de uso a sus actores todo sistema también proporciona un conjunto de servicios para ofrecer a sus usuarios los casos de uso necesarios para llevar a cabo su negocio.

Los paquetes de servicio:

- Contiene un conjunto de clases relacionadas funcionalmente.
- Es indivisible.

- Cuando se lleva a cabo un caso de uso, puede que sean participantes uno o más paquetes de servicio.
- Depende a menudo de otro paquete de servicio.
- Normalmente sólo es relevante para uno o unos pocos actores.
- La funcionalidad definida, cuando se diseña e implementa puede gestionarse como unidad de distribución separada.
- Pueden ser mutuamente excluyentes, o pueden representar diferentes aspectos o variantes del mismo servicio.
- Constituyen una entrada fundamental para las actividades de diseño e implementación subsiguientes, dado que ayudarán a estructurar los modelos de diseño e implementación en términos de subsistemas de servicio.

- Descripción de la arquitectura del análisis

La descripción de la arquitectura contiene una vista, que muestra sus artefactos significativos para la arquitectura. Los siguientes artefactos del modelo de análisis normalmente se consideran significativos para la arquitectura:

- La descomposición del modelo de análisis en paquetes de análisis y sus dependencias.
- Las clase fundamentales del análisis como las clases de entidad que encapsulan un fenómeno importante del dominio del problema; las clases de interfaz que encapsulan interfaces de comunicación importantes y mecanismos de interfaz de usuario; las clases de control que encapsulan importantes secuencias con un amplia cobertura; y clases del análisis que son generales, centrales, y que tienen muchas relaciones con otras clases del análisis.
- Realizaciones de casos de uso que describen cierta funcionalidad importante y crítica; que implican muchas clases del análisis.

1.4.2 Flujo de Trabajo

Los arquitectos comienzan la creación del modelo del análisis, identificando los paquetes de análisis principales, las clases de entidad evidentes, y los requisitos comunes. Después los ingenieros de casos de uso realizan cada caso de uso en términos de las clases del análisis participantes exponiendo los requisitos de comportamiento de cada clase. Los ingenieros de componentes especifican posteriormente estos requisitos y los integran dentro de cada clase creando responsabilidades, atributos y relaciones consistentes para cada clase. Durante el análisis, el arquitecto identifica de manera continua nuevos paquetes del análisis, clases, y requisitos comunes a medida que el modelo de análisis evoluciona, y los ingenieros de componentes responsables de los paquetes del análisis concretos continuamente los refinan y los mantienen.

1. Actividad: definir la arquitectura del análisis

El propósito de la **arquitectura del análisis** es esbozar el modelo de análisis y la arquitectura mediante la identificación de paquetes del análisis, clases del análisis evidentes, y requisitos especiales comunes.

- Identificación de paquetes del análisis.

Los paquetes del análisis proporcionan un medio para organizar el modelo de análisis en piezas más pequeñas y más manejables. Pueden, bien identificarse inicialmente como forma de dividir el trabajo de análisis, o bien encontrarse a medida que el

modelo de análisis evoluciona y “crece” convirtiéndose en una gran estructura que debe descomponerse.

Una forma directa de identificar paquetes del análisis es asignar la mayor parte de un cierto número de casos de uso a un paquete concreto y después realizar la funcionalidad correspondiente dentro de ese paquete.

Los paquetes de estos tipos localizan los cambios respectivamente en un proceso del negocio, en el comportamiento de un actor, y en un conjunto de casos de uso estrechamente relacionados. Esta técnica simplemente nos ayuda inicialmente a asignar casos de uso a paquetes.

La identificación adecuada de los paquetes de servicio se suele hacer cuando el trabajo de análisis está avanzado, momento en el que los requisitos funcionales se comprenden bien y existe la mayoría de las clases del análisis. Todas las clases del análisis dentro del mismo paquete de servicio contribuyen al mismo servicio.

Deberían definirse dependencias entre los paquetes del análisis si sus contenidos están relacionados.

- Identificación de clases obvias.

Suele ser adecuado preparar una propuesta preliminar de las clases de entidad más importantes basado en las clases del dominio o las entidades del negocio que se identificarán al crea las realizaciones de los casos de uso.

- Identificación de requisitos especiales comunes.

Un requisito especial es un requisito que aparece durante el análisis y que es importante anotar de forma que pueda ser tratado adecuadamente en las siguientes actividades de diseño e implementación. Como ejemplo citamos las limitaciones o restricciones sobre:

- Persistencia.
- Distribución y concurrencia.
- Características de seguridad.
- Tolerancia a fallos.
- Gestión de transacciones.

2. Actividad: Analizar los casos de uso

Analizamos un caso de uso para:

- Identificar las clases del análisis cuyos objetos son necesarios para llevar a cabo el flujo de sucesos de caso de uso.
- Distribuir el comportamiento del caso de uso entre los objetos del análisis que interactúan.
- Capturar requisitos especiales sobre la realización del caso de uso.

- Identificación de clases del análisis

En este paso, se identifican las clases de control, entidad, e interfaz necesarias para realizar los casos de uso y se esbozan sus nombres, responsabilidades, atributos y relaciones.

Se pueden utilizar las siguientes reglas en general para identificar las clases del análisis:

- Identificar clases de entidad mediante el estudio en detalle de la descripción del caso de uso y de cualquier modelo del dominio que se tenga, y después considerar qué información debe utilizarse y manipularse en la realización del caso de uso.
- Identificar una clase de interfaz central para cada actor humano, y dejar que esta clase represente la ventana principal del interfaz del usuario con el cual interactúa el actor.
- Identificar una clase de interfaz primitiva para cada clase de entidad que hayamos encontrado anteriormente. Estas clases representan objetos lógicos con los cual interactúa el actor.
- Identificar una clase de interfaz central para cada actor que sea un sistema externo, y dejar que esta clase represente la interfaz de comunicación.
- Identificar una clase de control responsable del tratamiento del control y de la coordinación de la realización del caso de uso, y después refinar esta clase de control de acuerdo a los requisitos del caso de uso.

- Descripción de interacciones entre objetos del análisis

Cuando se tiene un esbozo de las clases necesarias para realizar el caso de uso, debemos describir cómo interactúan sus correspondientes objetos del análisis. Esto se hace mediante diagramas de colaboración que contienen las instancias de actores participantes, los objetos del análisis, y sus enlaces.

Un diagrama de colaboración se crea comenzando por el principio del flujo del caso de uso, y continuando el flujo paso a paso decidiendo qué interacciones de objetos del análisis y de instancias de actor son necesarias para realizarlo. Se puede observar lo siguiente sobre estos diagramas de colaboración:

- El caso de uso se invoca mediante un mensaje proveniente de una instancia de un actor sobre un objeto de interfaz.
- Cada clase del análisis identificada en el caso anterior debería tener al menos un objeto que participase en un diagrama de colaboración.
- Los mensajes no se asocian a operaciones debido a que no especificamos operaciones en las clases del análisis.
- Los enlaces en el diagrama normalmente deben ser instancias de asociaciones entre clases del análisis.
- La secuencia en el diagrama o debería ser nuestro objetivo principal y puede eliminarse si es difícil de mantener o crea confusión en el diagrama.
- El diagrama de colaboración debería tratar todas las relaciones del caso de uso que se está realizando.

- Captura de requisitos especiales

En este paso se reúnen todos los requisitos sobre la realización de caso de uso que se identifican en el análisis pero deberían tratarse en el diseño y en la implementación, tales como los requisitos no funcionales.

3. Actividad: analizar una clase

Los objetivos de analizar una clase son:

- Identificar y mantener las responsabilidades de una clase del análisis, basadas en su papel en las realizaciones de casos de uso.
- Identificar y mantener los atributos y relaciones de la clase del análisis.
- Capturar requisitos especiales sobre la realización de la clase del análisis.

- Identificar responsabilidades

Las responsabilidades de una clase pueden recopilarse combinando todos los roles que cumple en diferentes realizaciones de caso de uso. Se pueden identificar todas las realizaciones de caso de uso en las cuales participa la clase mediante el estudio de sus diagramas de clases y de interacción.

- Identificación de atributos

Un atributo especifica una propiedad de una clase del análisis, y normalmente es necesaria para las responsabilidades de su clase. Se debería tener en mente las siguientes normas generales cuando para identificación de atributos:

- El nombre del atributo debería ser un nombre
- Recuérdese que el tipo de los atributos debería ser conceptual en el análisis, y, si es posible, no debería verse restringido por el entorno de la implementación.
- Al decidir el tipo de un atributo, debemos intentar reutilizar tipos ya existentes.
- Una determinada instancia de un atributo no puede compartirse por varios objetos del análisis.
- Si una clase del análisis se hace demasiado difícil de entender por culpa de sus atributos, algunos de esos atributos podrían separarse en clases independientes.
- Los atributos de las clases de entidad suelen ser bastantes evidentes.
- Los atributos de las clases de interfaz que interactúan con los actores humanos suelen representar elementos de información manipulados por los actores.
- Los atributos de las clases de interfaz que interactúan con los actores que representan sistemas suelen representar propiedades de una interfaz de comunicación
- Los atributos de las clases de control son pocos frecuentes debido a su corto tiempo de vida.
- A veces no son necesarios los atributos formales.
- Si una clase tiene muchos atributos o atributos complejos, podemos mostrarlo en un diagrama de clases aparte, que sólo muestre la sección de atributos.

- Identificación de asociaciones y agregaciones

Los objetos del análisis interactúan unos con otros mediante enlaces en los diagramas de colaboración. Estos enlaces suelen ser instancias de asociaciones entre sus correspondientes clases.

Se debería minimizar el número de relaciones entre clases. No son las relaciones del mundo real lo que se debe modelar como agregaciones o asociaciones, si no las relaciones que deben existir en respuesta a las demás de las diferentes realizaciones de caso de uso.

Las agregaciones deberían utilizarse cuando lo objetos representan:

- Conceptos que se contienen físicamente uno al otro.
- Conceptos que están compuestos uno de otro.
- Conceptos que forman una colección conceptual de objetos.

- Identificación de generalizaciones

Las generalizaciones deberían utilizarse durante el análisis para extraer comportamiento compartido y común entre varias clases del análisis diferentes.

1.5 Diseño

El diseño es el centro de atención al final de la fase de elaboración y el comienzo de las iteraciones de construcción. Esto contribuye a una arquitectura estable y sólida y a crear un plano del modelo de implementación.

1.5.1 Artefactos

- Modelo de diseño

El modelo de diseño es un modelo de objetos que describe la realización física de los casos de uso centrándose en cómo cubrir los requisitos funcionales y los no funcionales.

El modelo del diseño se representa por una forma de organizar el modelo en porciones más manejables.

En el modelo de diseño, los casos de uso son realizados por las clases de diseño y sus objetos. Esto se representa por colaboraciones en el modelo de diseño y denota realización de caso de uso-diseño.

- Clase del diseño

Una clase de diseño es una abstracción de una clase similar en la implementación del sistema:

- El lenguaje utilizado para especificar una clase del diseño es lo mismo que en el lenguaje de programación.
- La visibilidad de los atributos y las operaciones de una clase de diseño se especifica.
- Las relaciones de aquellas clases del diseño implicadas con otras clases, a menudo tienen un significado directo cuando la clase es implementada.
- Los métodos de una clase del diseño tienen correspondencia directa con el correspondiente método en la implementación de las clases.
- La clase de diseño puede posponer el manejo de algunos requisitos para las subsiguientes actividades de implementación, indicándolos como requisitos de implementación de la clase.

- La clase de diseño puede realizar interfaces si tiene sentido hacerlo en el lenguaje de programación.
- La clase de diseño puede activarse, implicando que objetos de la clase mantengan su propio hilo de control y se ejecuten concurrentemente con otros objetos activos.

- Realización de caso de uso del diseño

La realización de caso de uso del diseño es el modelo de diseño que describe cómo se realiza un caso de uso específico, y cómo se ejecuta, en términos de clases de diseño y sus objetos. Una realización de caso de uso-diseño representa la traza directa a una realización de caso de uso-análisis en el modelo del análisis.

La realización de caso de uso del diseño tiene una descripción del flujo de eventos textual, diagramas de clases que muestra las clases de diseño participantes, y diagramas de interacción que muestran la realización de un flujo o escenario concreto de un caso de uso en términos de interacción entre objetos del diseño.

Diagramas de clases
Una clase de diseño y sus objetos, y de ese modo también los subsistemas que contienen las clases del diseño, a menudo participan en varias realizaciones de casos de uso. También puede darse el caso de algunas operaciones, atributos y asociaciones sobre una clase específica que son relevantes para sólo una realización de caso de uso. En la Figura 1.2 se muestra un ejemplo de un diagrama de clases.

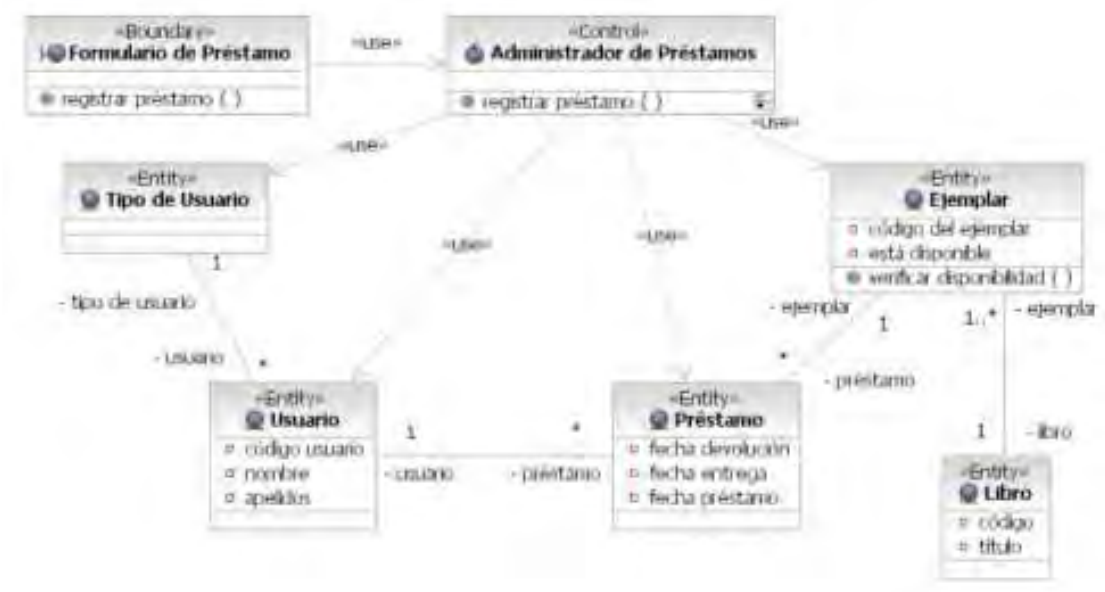


Figura. 1.2 Ejemplo de diagrama de clases [OMG]

Diagramas de interacción
La secuencia de acciones en un caso de uso comienza cuando un actor invoca el caso de uso mediante el envío de algún tipo de mensaje al sistema. Si consideramos el "interior" del sistema, tendremos algún objeto de diseño que recibe el mensaje del actor. Después el objeto del diseño llama a algún otro objeto, y de esta manera los objetos implicados interactúan para realizar y llevar a cabo el caso de uso. La Figura 1.3 muestra un ejemplo de diagrama de iteración, y la secuencia de acciones de los objetos dados.

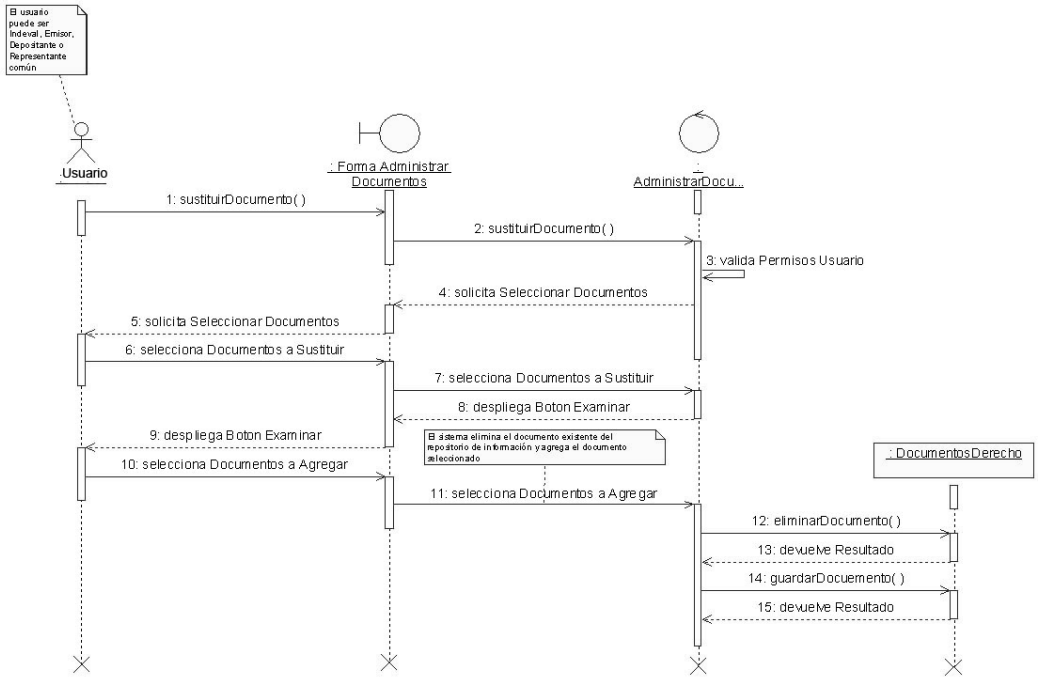


Figura 1.3 Ejemplo de diagrama de interacción.

- Interfaz

Las interfaces se utilizan para especificar las operaciones que proporcionan las clases y subsistemas del diseño.

Una clase de diseño que proporcione una interfaz debe proporcionar también métodos que realicen las operaciones de la interfaz. Un subsistema que proporcione una interfaz debe contener también las clases del diseño.

Las interfaces constituyen una forma de separar la especificación de la funcionalidad en términos de operaciones, de sus implementaciones en términos de métodos. Esta distinción hace independiente de la implementación de la interfaz a cualquier cliente que dependa de ella. La mayoría de las interfaces entre subsistema se consideran relevantes para la arquitectura debido a que definen las interacciones permitidas entre los subsistemas.

- Descripción de la arquitectura (vista del modelo de diseño)

La descripción de la arquitectura contiene una vista, que muestra los artefactos relevantes para la arquitectura.

Suelen considerarse significativos para la arquitectura los siguientes artefactos del modelo del diseño:

- La descomposición del modelo de diseño de subsistemas, su interfaces, y las dependencias entre ellos.
- Clases del diseño fundamentales como clases que poseen una traza con clases del análisis, clases activas, y clases del diseño que sean generales y

centrales, que representen mecanismos de diseño genéricos, y que tengan muchas relaciones con otra clases del diseño.
- Realizaciones de caso de uso-diseño que describan alguna funcionalidad importante y crítica que debe desarrollarse pronto dentro del ciclo de vida del software, que impliquen muchas clases del diseño y por tanto tengan una cobertura amplia.

- Modelo de distribución

El modelo de distribución es un modelo de objetos que describe la distribución física del sistema en términos de cómo se distribuye la funcionalidad entre lo nodos de cómputo.

Se puede observar lo siguiente sobre el modelo de distribución:

- Cada nodo representa un recurso de cómputo, normalmente un procesador o un dispositivo hardware similar.
- Los nodos poseen relaciones que representan medios de comunicación entre ellos, tales como internet, intranet, bus, y similares.
- El modelo de distribución puede describir diferentes configuraciones de red.
- La funcionalidad de un nodo se define por los componentes que se distribuyen sobre ese nodo.

1.5.2 Flujo de trabajo

Los arquitectos inician la creación de los modelos de diseño y de distribución. Ellos esbozan los nodos del modelo de distribución, los subsistemas principales y sus interfaces, las clases del diseño importantes, y los mecanismos genéricos de diseño el modelo de diseño. Después, los ingenieros de casos de uso realizan cada caso de uso en términos de clases y/o subsistemas del diseño participantes y sus interfaces. Las realizaciones de caso de uso resultantes establecen los requisitos de comportamiento para cada clase o subsistema que participe en alguna realización de caso de uso. Los ingenieros de componentes especifican a continuación los requisitos, y los integran dentro de cada clase, o bien mediante la creación de operaciones, atributos y relaciones consistentes sobre cada clase, o bien mediante la creación de operaciones consistentes en cada interfaz que proporcione el sistema.

1. Actividad: diseño de la arquitectura

El objetivo del diseño de la arquitectura es esbozar los modelos de diseño y distribución y su arquitectura mediante la identificación de los siguientes elementos:

- Nodos y sus configuraciones de red.
- Subsistemas y sus interfaces.
- Clases del diseño significativas para la arquitectura, como las clases activas.
- Mecanismos de diseño genéricos que tratan requisitos comunes.

- Identificación de nodos y configuraciones de red

Las configuraciones de red habituales utilizan un patrón de tres capas en el cual la interfaz de usuario se deja en una capa, la funcionalidad de base de datos en otra, y la lógica del negocio o de de la aplicación en una tercera.

- Identificación de subsistemas y de sus interfaces

Los subsistemas constituyen un medio para organizar el modelo de diseño en piezas manejables. Pueden bien identificarse inicialmente como forma de dividir el trabajo de diseño, o bien pueden irse encontrando a medida que el modelo de diseño evoluciona y va "creciendo" hasta convertirse en una gran estructura que debe ser descompuesta.

En este paso identificamos los subsistemas de las capas específica de la aplicación y general de la aplicación.

El software del sistema y la capa intermedia constituyen los cimientos de un sistema, ya que toda la funcionalidad descansa sobre software como sistemas operativos, sistema de gestión de base de datos, software de comunicaciones, tecnologías de distribución de objetos, kits de diseño de GUI, y tecnología de gestión de transacciones.

Deberían definirse dependencias entre subsistemas si sus contenidos tiene relaciones unos con otros. La dirección de la dependencia debería ser la misma que la dirección de la relación.

Las interfaces proporcionadas por un subsistema definen operaciones que son accesibles "desde fuera" del subsistema. Estas interfaces las proporcionan clases o bien otros subsistemas dentro del subsistema.

- Identificación de clases del diseño relevantes para la arquitectura

Se puede esbozar inicialmente algunas clases del diseño a partir de las clases del análisis. Además, de utilizar las relaciones entre esas clases en el análisis para identificar un conjunto tentativo de relaciones entre las correspondientes clases de diseño.

El arquitecto también identifica otros requisitos. Por ejemplo:

- Los requisitos de rendimiento, tiempo de respuesta y disponibilidad que tienen lo diferentes actores en su interacción con el sistema.
- La distribución del sistema sobre los nodos.
- Otros requisitos como requisitos sobre el arranque y terminación del sistema, progresión, evitar el interbloqueo, evitar la inanición, reconfiguración de nodos y la capacidad de los nodos.

- Identificación de mecanismos genéricos de diseño

En este paso se identifican requisitos comunes, se decide como tratarlos, teniendo en cuenta las tecnologías de diseño e implementación disponibles. El resultado es un conjunto de mecanismo genéricos de diseño que pueden manifestarse como clases, colaboraciones o incluso subsistemas.

Los requisitos que deben tratarse suelen estar relacionados con aspectos como:

- Persistencia.
- Distribución transparente de objetos.
- Características de seguridad.
- Detección y recuperación de errores.
- Gestión e transacciones.

2. Actividad: diseño de un caso de uso

Los objetivos del diseño de un caso de uso son:

- Identificar las clases del diseño y/o los subsistemas cuyas instancias son necesarias para llevar a cabo el flujo del caso de uso.
- Distribuir el comportamiento del caso de uso entre los objetos del diseño que interactúan y/o entre los subsistemas participantes.
- Definir los requisitos sobre las operaciones de las clases del diseño y/o sobre los subsistemas y sus interfaces.
- Capturar los requisitos e implementación del caso de uso.

- Identificación de clases del diseño participantes

En este paso se identificará las clases del diseño que se necesitan para realizar el caso de uso. Se debe hacer principalmente lo siguiente:

- Estudiar las clases del análisis que participan en la correspondiente realización de caso de uso análisis.
- Estudiar los requisitos especiales. Identificar las clases del diseño que realizan esos requisitos especiales.

- Descripción de las interacciones entre los objetos del diseño

Cuando se tiene un esquema de las clases del diseño necesarias para realizar el caso de uso, se debe describir cómo interactúan sus correspondientes objetos del diseño. Esto se hace mediante diagramas de secuencias que contienen las instancias de los actores, los objetos del diseño, y las transmisiones de mensaje entre éstos, que participan en el caso de uso.

Para crear un diagrama de secuencia, se debe comenzar por el principio del flujo del caso de uso, y después seguir el flujo paso a paso. En la mayoría de los casos, los objetos se ajustan de manera natural a la secuencia de interacciones de la realización de caso de uso. Se debe observar lo siguiente sobre estos diagramas de secuencia:

- El causante de la invocación del caso de uso es un mensaje de una instancia de un actor hacia un objeto del diseño.
- Cada clase del diseño identificada en el paso anterior debería tener al menos un objeto del diseño participante en el diagrama de secuencia.
- Los mensajes que realizan el caso de uso se envían entre líneas de vida de los objetos.
- La secuencia en el diagrama debería ser la principal preocupación, ya que la realización del caso de uso del diseño es la entrada principal para la implementación del caso de uso.
- Se utilizan etiquetas y el flujo de sucesos del diseño para complementar los diagramas de secuencia.
- El diagrama de secuencia debería tratar todas las relaciones del caso de uso que realiza.

3. Actividad: diseño de una clase

El propósito de diseñar una clase es crear una clase del diseño que cumpla su papel en las realizaciones de los casos de uso y los requisitos no funcionales que se aplican a éstos. Esto incluye el mantenimiento del diseño de clases en sí mismo y los siguientes aspectos de éste:

- Sus operaciones.
- Sus atributos.
- Las relaciones en las que participa.
- Sus dependencias con cualquier mecanismo de diseño genérico.
- Los requisitos relevantes a su implementación.
- La correcta realización de cualquier interfaz requerida.

- Esbozar la clase del diseño

Como un primer paso, se necesita esbozar una o varias clases del diseño, dada la entrada en términos de clases de análisis y/o interfaces:

- Diseñar las clases de interfaz es dependiente de la tecnología de interfaz específica que se utilice.
- Diseñar las clases de entidad que representen información persistente a menudo implica el uso de tecnología de bases de datos específica.
- Diseñar las clases de control es una tarea delicada. Debido a que se encapsulan secuencias, coordinación de otros objetos o algunas veces pura lógica del negocio, es necesario considerar los aspectos de distribución, rendimiento y transacción.

- Identificar operaciones

En esta etapa se identifican las operaciones que la clases de diseño van a necesitar y se describen esas operaciones utilizando la sintaxis de los lenguajes de programación. Algunas entradas importantes en esta etapa son:

- Las responsabilidades de cualquier clase del análisis que tenga una traza con la clase del diseño.
- Los requisitos especiales de cualquier clase de análisis que tenga una traza con la clase del diseño.
- La interfaces que la clase de diseño necesita proporcionar.
- Las realizaciones de caso de uso-diseño en las que la clase participa.

- Identificar atributos

En este paso, se identifican los atributos requeridos por la clase de diseño y se describen utilizando la sintaxis del lenguaje de programación. Un atributo especifica una propiedad de una clase de diseño y está a menudo implicado y es requerido por las operaciones de la clase.

- Identificar asociaciones y agregaciones

Los objetos de diseño interactúan unos con otros en los diagramas de secuencia. Las instancias de las asociaciones deben ser utilizadas para abarcar las referencias con otros objetos, y para agrupar objetos en agregaciones con el propósito de enviarles mensajes.

- Describir los métodos

Los métodos pueden se utilizados durante el diseño para especificar cómo se deben realizar las operaciones.

- Describir estados

Algunos objetos del diseño son estados controlados, lo que significa que sus estados determinan su comportamiento cuando reciben un mensaje. En estos casos, es significativa la utilización de diagramas de estado para describir las diferentes transiciones del estado de un objeto del diseño. Cada diagrama de estados es útil para la implementación de la correspondiente clase del diseño.

.

Capítulo 2

Introducción a J2EE

El mundo de la tecnología crece rápidamente. Las aplicaciones empresariales deben ofrecer servicios que necesitan del entorno empresarial global, asegurando que los datos de usuario permanezcan privados, protección en la integridad de los datos y asegurar que las transacciones del negocio sean certeras y procesadas rápidamente. Las empresas necesitan reducir sus costos, extender sus metas, bajar tiempo de respuestas a clientes y empleados. En este capitulo veremos una introducción a J2EE que reduce costos y complejidad de desarrollo de servicios multicapa. Las aplicaciones J2ee pueden ser rápidamente desarrolladas y evolucionar fácilmente a lo el negocio o la empresa soliciten.

1.1 J2EE

Java 2 Enterprise Edition o J2EE, es un paquete de especificaciones para permitir el desarrollo de aplicaciones empresariales. Las especificaciones resumen varios de los componentes necesarios en un sistema empresarial.

En 1995 nace el lenguaje de programación orientado a objetos Java, en 1998 J2SE convierte a Java en no sólo un lenguaje sino una plataforma, esta plataforma era suficiente para crear aplicaciones *stand-alone*, pero faltaba un estándar para desarrollar aplicaciones empresariales. J2EE fue introducido en 1998.

Java 2 Enterprise Edition (J2EE) es una arquitectura multicapa para implementar aplicaciones de tipo empresarial y aplicaciones basadas en la Web. Esta tecnología soporta una gran variedad de tipos de aplicaciones desde sistemas Web de gran escala a pequeñas aplicaciones cliente-servidor. El objetivo principal de la tecnología J2EE es crear un modelo simple de desarrollo para sistemas empresariales utilizando componentes basados en el modelo de aplicación. En este modelo dichos componentes utilizan servicios proporcionados por el contenedor de aplicaciones, que de otro modo tendrían que estar incorporados en el código del sistema.

Las aplicaciones J2EE están compuestas de diferentes componentes. Un componente J2EE es una unidad de software funcional auto-contenido que se ensambla dentro de un sistema J2EE con sus clases de ayuda y archivos que se comunican con otros componentes de la aplicación. La especificación J2EE define los siguientes componentes:

- Las **Aplicaciones Clientes** y los **Applets** son componentes que se ejecutan en el lado del cliente.
- Los componentes **Java Servlet,** la tecnología **JavaServer Pages** son componentes Web que se ejecutan en el lado del servidor.
- Los **Enterprise JavaBeans** (beans enterprise) son componentes de negocio que se ejecutan en el servidor de aplicación.
- Todos esos componentes se ensamblan en una aplicación J2EE, se verifica que están bien formados y que cumplen la especificación J2EE, y se despliegan en el entorno de producción, donde se ejecutan y son controlados por el servidor de aplicaciones J2EE.

Además de estos componentes principales, J2EE incluye servicios estándar y tecnologías de soporte como:

- **Java Database Connectivity** (JDBC) tecnología que proporciona acceso a sistemas de bases de datos relacionales.

- **Java Transaction API (JTA)** o **Java Transaction Service (JTS)** proporciona soporte para transacciones a los componentes J2EE.
- **Java Messaging Service (JMS)** para comunicación asíncrona entre componentes J2EE.
- **Java Naming** y **Directory Interface (JNDI)** proporcionan accesos a nombres y directorios

Un cliente Web consta de dos partes: páginas Web dinámicas que contienen distintos tipos de lenguajes de marcado (HTML, XML, y otros), que son generados por los componentes Web que se ejecutan en la capa Web, y un navegador Web, que dibuja las páginas recibidas desde el servidor. Otra categoría de clientes Web son los conocidos como clientes *thin* (pequeños). Este tipo de clientes normalmente no hacen cosas como consultas a bases de datos o ejecutar reglas complejas de negocio. Cuando se utilizan clientes pequeños, las operaciones pesadas como éstas las manejan los beans enterprise que se ejecutan en el servidor J2EE donde pueden tratar con la seguridad, los servicios y el rendimiento de las tecnologías del lado del servidor J2EE.

Una aplicación cliente se ejecuta sobre una máquina cliente y proporciona una forma para que los usuarios puedan manejar tareas que requieren una interfaz de usuario más rico que el que puede proporcionar un lenguaje de marcado. Normalmente tienen una interfaz gráfica de usuario (GUI) creado con los APIs **Swing** o **Abstract Window Toolkit (AWT)**. Las aplicaciones cliente acceden directamente a los beans enterprise que se ejecutan en la capa de negocio. Pero si se necesita un cliente Web pueden abrir una conexión HTTP para establecer comunicación con un servlet que se ejecute en la capa Web.

Los componentes Web de J2EE pueden ser servlets o páginas JSP. Los servlets son clases Java que procesan dinámicamente las peticiones y construyen las respuestas. Las páginas JSP son documentos basados-en-texto que se ejecutan como servlets pero permiten una aproximación más natural para crear contenido estático. Las páginas HTML y los applets se juntan con los componentes Web durante el ensamble de la aplicación, pero la especificación J2EE no los considera como componentes J2EE. De forma similar, las clases de utilidades del lado del servidor también se unen a los componentes Web como páginas HTML, pero tampoco se consideran como componentes J2EE.

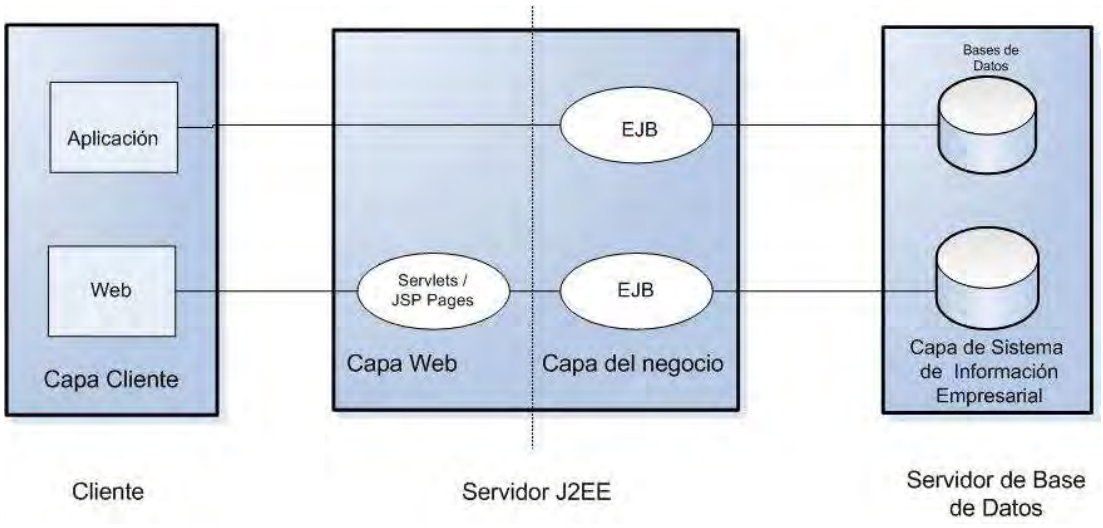


Figura 2.1 Componentes de la Arquitectura de 3 capas

La capa Web podría incluir componentes JavaBeans para manejar la entrada del usuario y enviar esta entrada a los beans enterprise que se ejecutan en la capa de negocio para su procesamiento según la figura anterior.

El código de negocio, que es la lógica que resuelve o cumple las necesidades de un negocio particular, como la banca, la venta, o la financiación, se maneja mediante beans enterprise que se ejecutan en la capa de negocio.

2.2 El Modelo Vista-Controlador (MVC)

El MVC provee de un patrón para separar la lógica de presentación (vista), lógica del negocio (control), y los datos (modelo). Típicamente, los beans de entidad son utilizados para proveer la lógica del modelo, mientras una mezcla de beans de entidad y sesión se utilizan para implementar la lógica de control, y los componentes Web son usados para implementar lógica de control y de presentación.

El principal objetivo de la arquitectura MVC es aislar tanto los datos de la aplicación como el estado (modelo) de la misma, del mecanismo utilizado para representar (vista) dicho estado, así como para modularizar esta vista y modelar la transición entre estados del modelo (controlador). Las aplicaciones MVC se dividen en tres grandes áreas funcionales:

- **Vista**: la presentación de los datos
- **Controlador**: el que atenderá las peticiones y componentes para toma de decisiones de la aplicación
- **Modelo**: la lógica del negocio o servicio y los datos asociados con la aplicación

El propósito del MVC es aislar los cambios. Es una arquitectura preparada para los cambios, que desacopla datos y lógica de negocio de la lógica de presentación, permitiendo la actualización y desarrollo independiente de cada uno de los citados componentes. En la figura 2.2 se muestra un diagrama de este modelo.

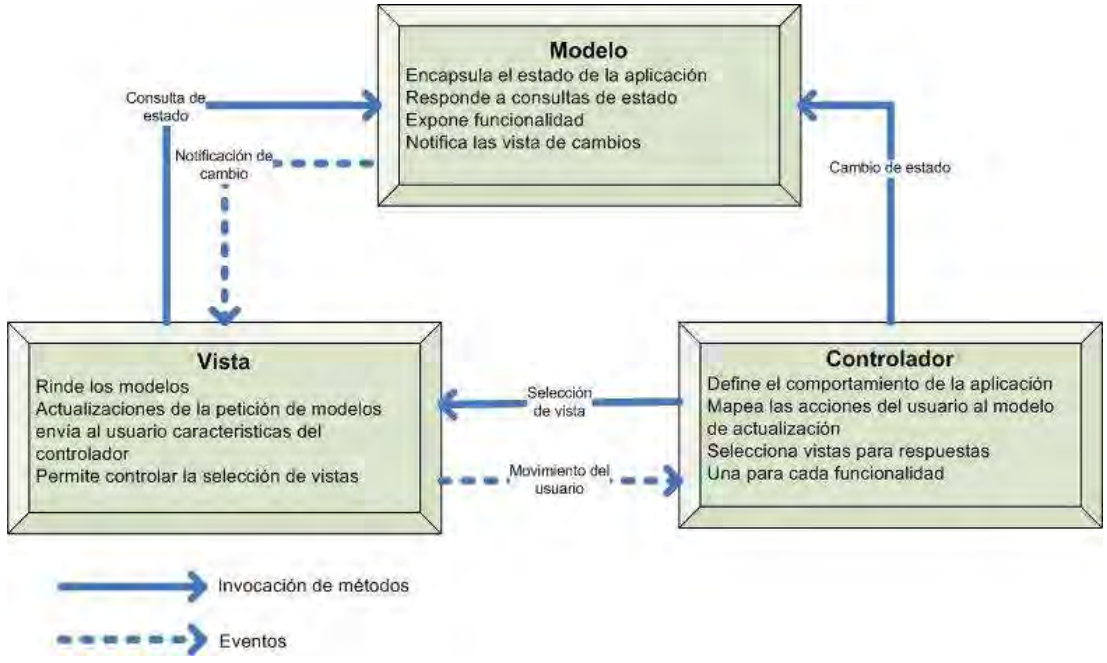


Figura 2.2 Modelo Vista Controlador

Este modelo realza el mantenimiento, escalabilidad y modificación a sistemas empresariales, permitiendo entretejer presentación, lógico de negocio y acceso a datos es sencillo para un simple cliente, pero debido a la forma en que se encuentra involucrado el Internet, la siguiente figura muestra algunas aplicaciones.

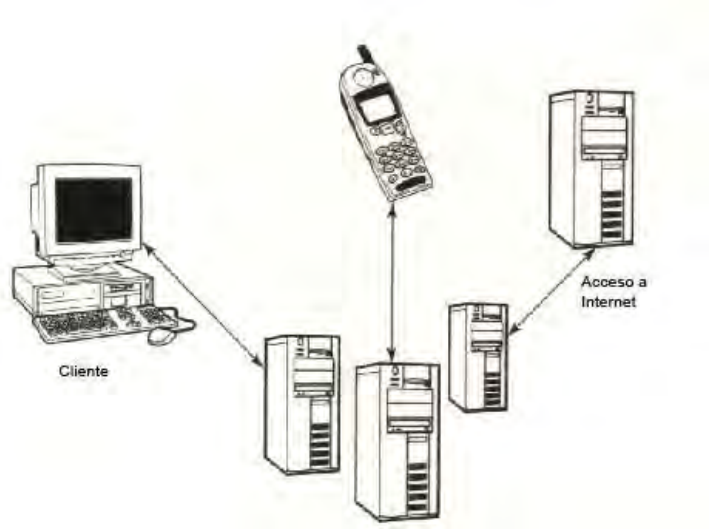


Figura 2.3 Ejemplo de un uso del patrón modelo vista controlador

Actualmente los desarrollos tienen diferentes líneas de usuarios, como pueden ser navegadores web, sistemas wireless y web service. Un sistema empresarial puede ser dividido en tres partes lógicas, el modelo la vista y el controlador para mantenimiento y escalabilidad. El equipo de desarrollo de software realiza el diseño, la implementación y el mantenimiento será comprendido por individuos con diferentes habilidades.

El modelo MVC tiene las siguientes fortalezas:

- Componentes y subsistemas que pueden ser vistos de diferentes formas, por ejemplo un html o un wml como xml.
- Múltiples orígenes pueden invocar diferentes comportamientos en un mismo componente.
- El comportamiento de un componente cambia con su uso.
- La representación del componente cambia con su uso.
- El componente puede ser reutilizado con un mínimo de remodificación.
- El soporte a múltiples vistas e interacciones no debería afectar componentes que procesen y proporcionen funciones al centro de la aplicación empresarial.

Implementación.

El modelo MVC permite separar los componentes de negocio de la capa de presentación o de vista de la del controlador que usa los componentes de negocio. Diferentes modelos de presentación pueden usar el mismo modelo de datos, así múltiples clientes como hypertext markup language (HTML), wireless markup language (WML) y el extensible markup language (XML), pueden ser todos fácilmente implementados.

Vista

La parte del despliegue en este patrón es la vista, pueden ser utilizados diferentes tecnologías para ver y mostrar el modelo como puede ser el caso de AWT, la vista notifica al modelo y el modelo notificara a la vista cuando exista alguna actualización, el siguiente diagrama muestra la interacción de la vista con el controlador y el modelo para una actualización de la vista.

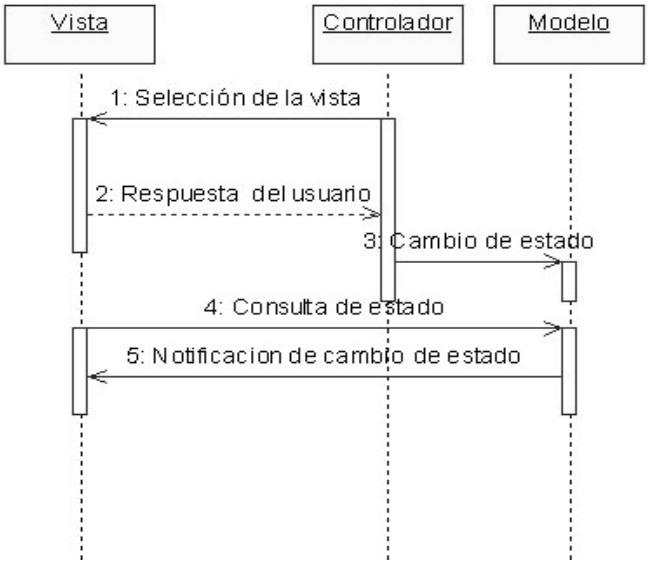


Figura 2.4 Diagrama de secuencia del patrón MVC

Controlador

El controlador administra el flujo entre la vista y el modelo, basado en la vista el controlador lo traduce en una acción que será manejado por el modelo, por ejemplo una forma en una aplicación web en donde el usuario esta actualizando la información de su personal presionando un botón de actualización puede causar que el controlador notifique al modelo la actualización a la base de datos.

Modelo

El modelo representa ambos la lógica de negocios y los datos usados para representar la vista, la lógica de negocios manipula y consulta datos para desplegarlos en la vista.

Estrategias.

La vista puede ser implementada usando html, java server pages (jsp), java server faces (jsf) o servlets, el controlador podría ser un servlet que transforme los *request* de la vista al modelo, el modelo puede ser dividido en dos componentes, un componente de lógica de negocio, a menudo implementado con *enterprise java beans* y datos representados en una base de datos.

Separando varios componentes de la aplicación en capas permite la reutilización de componentes a través de aplicaciones, los equipos pueden trabajar independientemente en una capa e integrarlo posteriormente, este modelo trabaja muy bien con *extreme programming*, esta es una metodología de software que puede ser usada en proyectos con equipos pequeños uno de los principios de esta metodología es dividir el equipo en actividades basadas en roles, sub-equipos que interactúan entre si, de forma q todos los miembros tienen experiencia en lo que hacen los demás dentro de la aplicación. Este desarrollo proporciona colaboración dentro del equipo.

Capítulo 3

Descripción del Problema

En este capítulo se describe detalladamente los requisitos del sistema dados por el cliente.

3.1 Requisitos del sistema.

En una institución, se maneja una intranet dirigida a todo empleado que trabaje para ella. La intranet cuenta con una sección muy amplia en Recursos Humanos donde se encuentran varias aplicaciones para el desarrollo, información, diversión e incluso evaluación de los empleados.

La aplicación Oportunidades Internas consiste en postular a los empleados a las plazas que constantemente son abiertas en la institución, éste es el problema al cuál se propone una solución en el presente trabajo.

Los objetivos que debe de cubrir la aplicación como en toda búsqueda de empleo es necesario presentar un currículum en el cual se muestre toda la trayectoria laboral, por lo que es deseable poder obtener o capturar todos los datos que lo conforman en la aplicación, en el caso de obtención tener acceso a la base de datos general de empleados de la institución. Se requiere un proceso de selección en el cual se elija al empleado que más se acerque al perfil buscado.

Las políticas para utilizar el sistema serán las siguientes:

Reglas Generales del sistema Oportunidades Internas

- El desarrollo de los empleados es responsabilidad de todos y cada uno de los que laboran en la institución y será fomentado a través del programa denominado Oportunidades Internas, que consiste en la publicación en la intranet de las vacantes existentes para que todos los empleados que lo deseen y tengan el perfil solicitado puedan participar como posibles candidatos para ocuparla.
- El programa Oportunidades Internas, se apegará a la política vigente sobre incrementos de sueldo y movimientos de personal.

Reglas para los jefes

- Ningún jefe podrá obstaculizar o negar la participación libre de sus empleados para postularse a una plaza vacante.
- Este programa está orientado para fomentar el desarrollo profesional del personal, podrá participar en él, de manera libre y voluntaria, en donde no se permite por parte de las áreas de origen, ninguna actitud comentario o acción tendiente a disuadir al empleado que desee participar, ni mucho menos que la participación en este programa sea causa para obstaculizar en la obtención de cualquier beneficio merecido por el empleado.
- No es válida ninguna promesa de cambio o mejora de condiciones de trabajo, puesto o salario, que no esté debidamente aprobada por Recursos Humanos mediante los canales institucionales.
- Si cualquiera de estos acuerdos del programa Oportunidades Internas no son cumplidos por algún jefe, el programa podría suspenderse.
- Al ser aceptado el candidato para la vacante publicada en Oportunidades Internas, el plazo para efectuar la transferencia al nuevo departamento será de 30 días como máximo.

Reglas para los empleados

- La postulación para ocupar una plaza vacante será mantenida en forma confidencial si así lo prefiere y lo manifiesta el propio empleado.
- El empleado podrá solicitar que sea Recursos Humanos (RRHH) quien le comunique a su jefe inmediato de su postulación en el momento apropiado, que será justamente antes de la primera entrevista.
- Para postularse a una plaza vacante se deberá tener una antigüedad mínima de **18 meses** en el puesto actual.

El empleado debe poder:

- Aceptar o no aceptar las políticas de la aplicación. De aceptarlas podrá tener acceso a la búsqueda de vacantes.

- Buscar las vacantes que se convoquen para la región a la que pertenece, las posibles regiones son las siguientes:

  TODAS
  CENTRO
  METRO NORTE
  METRO SUR
  METROPOLITANA
  NORESTE
  NOROESTE
  NORTE
  OCCIDENTE
  SUR
  SURESTE

- Visualizar la búsqueda desplegada, la cual refleja los siguientes campos:

  Puesto a cubrir
  Departamento
  Dirección ejecutiva
  Dirección general
  Posición

- Elegir una vacante dentro de la lista de consulta.

- Ver toda la información relacionada con la vacante:

  Región
  Dirección general
  Dirección ejecutiva
  Departamento o Sucursal
  Puesto a cubrir
  Posición
  Reportar a

  Domicilio Laboral
  Calle, Colonia, Ciudad, Estado, Código Postal

  Tareas a realizar
  Perfil
  Fecha

- Decidir si postularse o regresar a la lista de vacantes

Al decidir postularse, deberá autentificarse como empleado con su número de expediente e ingresar a su información general de la base de datos de personal, ésta debe ser:

Datos del Postulante
Apellido Paterno
Apellido Materno
Nombres
Fecha de Nacimiento
Región
Dirección general
Dirección ejecutiva
Departamento o Sucursal
Puesto actual
Sueldo Actual
Si es empleado interno o externo

Domicilio Laboral
Calle, Colonia, Ciudad, Estado, Código Postal
Correo electrónico
Teléfono y extensión

Formación Académica
Nivel Máximo de Estudios
Especialidad (carrera)
Titulado
Año de titulación

En caso de que su información no esté completa tendrá que actualizar en el Sistema de Actualización de personal.

- Deberá completar su información de otros estudios y los puestos en donde ha trabajado, dentro del grupo así como en otras instituciones.

- Finalmente podrá postularse.

El administrador necesita:

- Autenticarse como administrador.

- Dar de alta nuevas Oportunidades Internas, con los siguientes datos:

  Región
  Dirección general
  Dirección ejecutiva

Departamento o Sucursal
Puesto a cubrir
Posición
Reportar a

Domicilio Laboral
Calle, Colonia, Ciudad, Estado, Código Postal
Tareas a realizar
Perfil
Fecha (vigencia)
Marcar la vacante como activa o no.

- Consultar las vacantes por: región, fecha y activa o no.

- Ver en la búsqueda los siguientes datos:
  Puesto a cubrir
  Departamento
  Dirección ejecutiva
  Dirección general
  Posición

  Con la opción de habilitar o deshabilitar las vacantes que se muestran.

- Entrar a la vacante donde podrá consultar/modificar los siguientes datos de la misma:

  Región
  Dirección general
  Dirección ejecutiva
  Departamento o Sucursal
  Puesto a cubrir
  Posición
  Reportar a
  Domicilio Laboral
  Calle, Colonia, Ciudad, Estado, Código Postal
  Tareas a realizar
  Perfil
  Fecha
  Activa o no

- Eliminar una vacante.

- Consultar los postulantes de una vacante, buscando por el estado de selección del postulante:

  Postulado
  Seleccionado
  Rechazado

- Ver los siguientes datos generales del postulado:

  Eliminar (Tendrá opción de eliminar)
  No. Expediente
  Puesto actual
  Departamento
  Dirección general
  Sueldo actual
  Estado
  C.V. (Liga para generar currículo)

- Consultar al postulado mas detalladamente

  Datos del Postulante
  Apellido Paterno
  Apellido Materno
  Nombres
  Fecha de Nacimiento
  Región
  Dirección general
  Dirección ejecutiva
  Departamento o Sucursal
  Puesto actual
  Sueldo Actual
  Si es empleado interno o externo

  Domicilio Laboral
  Calle, Colonia, Ciudad, Estado, Código Postal
  Correo electrónico
  Teléfono y extensión

  Formación Académica
  Nivel Máximo de Estudios
  Especialidad (carrera)
  Titulado
  Año de titulación

- Poder seleccionar, rechazar, cambiar status de acuerdo a si el postulante se apega al perfil requerido. Los status posibles son:

  Validación de datos
  Entrevista en Recursos Humanos
  Currículum Jefe
  Entrevista Jefe
  Aceptado

  Los cuales solo tienen sentido si un postulante ha sido previamente seleccionado.

- Documentar el por qué el postulante fue rechazado.

- Consultar reportes estadísticos. Con los siguientes filtros:

  Total de Postulantes por Vacantes, con la siguiente información**:**
  Dirección general
  Dirección ejecutiva
  Departamento
  Puesto a cubrir
  Número de postulantes

  Total de Postulantes por Región
  Dirección general
  Dirección ejecutiva
  Departamento
  Puesto a cubrir
  Número de postulaciones recibidas

  Total de Empleados Promovidos a través del Programa

Dirección general
Dirección ejecutiva
Departamento
Puesto a cubrir
Número de candidatos
Expediente
Nombre

Reporte de Vacantes activas y vigentes por Región y Puesto
Dirección general
Dirección ejecutiva
Departamento
Puesto a cubrir
Número de vacantes activas y vigentes

Reporte de Vacantes por Status
Dirección general
Dirección ejecutiva
Departamento
Puesto a cubrir
Status de vacantes activas en proceso
(Número de postulantes en cada estado)
- Validación de datos
- Entrevista en Recursos Humanos
- Currículum Jefe
- Entrevista Jefe
- Aceptado

Total

Reporte por Motivos de Rechazo
Dirección general
Dirección ejecutiva
Departamento
Puesto a cubrir
Motivos de Rechazo
(Número de postulantes por cada motivo)
- Perfil
- Zona Geográfica
- Sueldo Solicitado
- Ranking
- Otras

Total

Empleados postulados en más de una vacante a la vez
Dirección general
Dirección ejecutiva
Departamento
Puesto a cubrir
Empleados Postulados más de una vez
(Número de postulantes en cada estado)
- Expediente
- Nombre
- Puesto Actual

Capítulo 4

Captura de Requisitos como casos de uso

En este capítulo se llevarán a cabo las actividades descritas en la parte del capítulo 1, captura de requisitos como casos de uso.

4.1 Modelado del negocio

Antes de comenzar con el flujo de trabajo de este capítulo se desarrollarán las actividades previas que son el modelado del dominio y casos de uso del modelo del negocio, mencionado en la sección 1.2

Casos de uso del negocio

En los requisitos plasmados en la descripción del problema se tiene el sistema con un empleado que desea entrar en la lista de selección para una vacante, por tanto el sistema interactúa con un postulante, en la parte administrativa se desea uno o más usuarios privilegiados para la manipulación de vacantes y selección de postulantes, por tanto el sistema interactúa con uno o mas usuarios con posibilidad de administrar.

Para el caso de uso de negocio: postulación del empleando, se tiene la siguiente secuencia:

1. El postulante ingresa al sistema
2. Revisa las vacantes disponibles
3. Da de alta su postulación

Para el caso de uso de negocio: administración de vacantes, se tiene la siguiente secuencia:

1. El administrador ingresa al sistema
2. Manipula las vacantes y/o la postulantes
3. Realiza consultas

Modelo del dominio

Se obtienen las clases del dominio Vacante, Postulante, Administrador y Currículum. En la Figura 4.1 se tiene el diagrama del dominio de estos conceptos, mostrando los atributos que se identifican rápidamente.

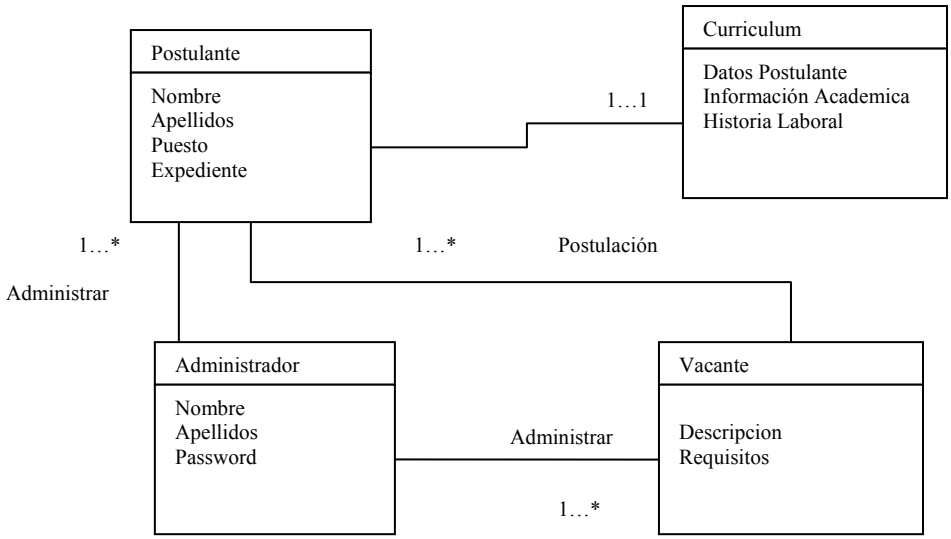


Figura 4.1 Diagrama de clases en el modelo del dominio, que captura los conceptos más importantes del contexto del sistema

4.2 Encontrar actores y casos de uso.

Se identifican los actores y casos de uso para delimitar el sistema de su entorno, se delimitan quien y qué funcionalidad se espera del sistema. En las figuras 4.2 y 4.3 se muestran los diagramas generales de casos de uso para administrador y postulante, respectivamente.

Se puede partir del modelo del negocio para encontrar los actores postulante y administrador.

Postulante

Un postulante representa un empleado que desea promoverse a un puesto mejor dentro de la institución. Utiliza el sistema para postularse a una vacante publicada.

Administrador

El administrador del sistema representa a la persona encargada del área de recursos humanos de reclutar personas dentro de la misma empresa para mejores puestos. Este usuario utiliza el sistema y cuenta con privilegios para realizar alta, baja y modificación de vacantes, consultar postulantes por vacante, seleccionar, rechazar postulantes, enviar currículo de postulantes. Y consultar reportes y estadísticas de vacantes y/o postulantes.

4.3 Priorizar casos de uso

En este caso solo se plantea los casos de uso necesarios para la primera iteración.

A continuación se definen los casos de uso para cada actor:

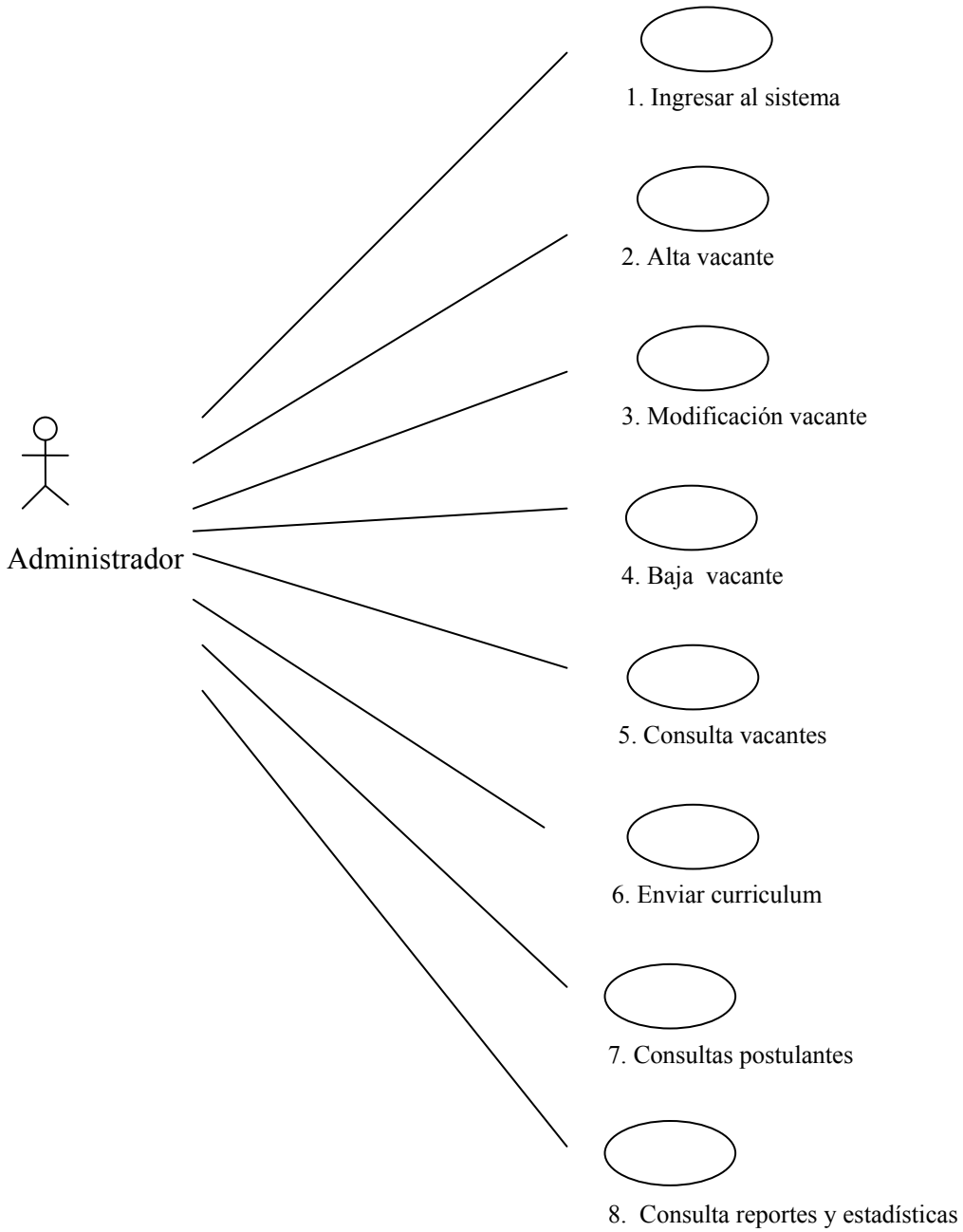


Figura 4.2 Diagrama general de casos de uso del administrador

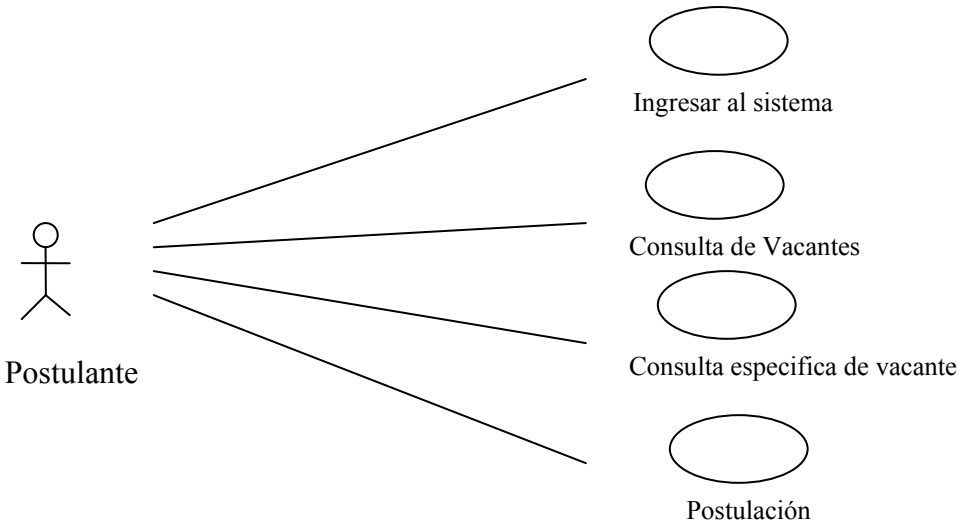


Figura 4.3 Diagrama general de casos de uso del postulante

4.4 Detallar casos de uso

Detallar cada caso de uso paso a paso en una especificación precisa de la secuencia de acciones. A continuación se detallan todos los casos de uso para la aplicación:

Caso de uso: Ingresar Sistema
Actor: Postulante.

Descripción: El postulante ingresa al sistema después de que se validó su número de expediente.

Precondiciones:

- El postulante está interesado en enterarse de las vacantes publicadas en la institución.
- El postulante cuenta con número de empleado.

Flujo:

ACTOR		**SISTEMA**		
Paso	**Acción**	**Paso**	**Acción**	**Excepción**
1	El postulante entra a la página	2	El sistema carga la página	

	de ingresar al sistema.		de ingresar al sistema.	
3	El postulante ingresa su número de expediente y realiza la petición de ingreso al sistema	4	El sistema valida si existe el número de expediente y si los datos del postulante están actualizados.	E1
		5	El sistema muestra los datos del postulante para que sean revisados.	E2
6	El postulante verifica sus datos y entra al sistema	7	El sistema despliega la pantalla con las políticas del mismo.	E3
8	El postulante acepta las políticas	9	El sistema muestra al postulante la pantalla de vacantes.	

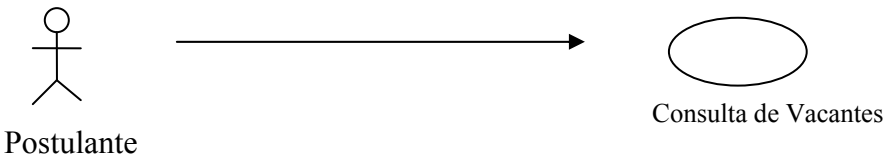
Excepciones:

Excepción	Nombre	Acción
E1	Número de expediente no válido	Indicarle al postulante que su número no esta registrado y pedirle que se registre en el sistema de Recursos Humanos.
E2	Datos no actualizados	Se le indica al usuario que entre al sistema de Actualización de Personal, para que sus datos sean actualizados.
E3	No se aceptan las políticas	Salir del sistema.

Poscondiciones:

- El postulante ya se encuentra dentro del sistema.
- El postulante se ha posicionado en la pantalla principal de búsqueda de vacantes.

Caso de uso: Consulta de Vacantes
Actor: Postulante



Descripción: El postulante Consulta las Vacantes

Precondiciones:

- El postulante ha ingresado al sistema.
- El sistema se encuentra en la página principal de búsqueda de vacantes.

Flujo:

ACTOR		SISTEMA		
Paso	**Acción**	**Paso**	**Acción**	**Excepción**
1	Elige los parámetros de búsqueda			
2	Lanza una petición al sistema para la consulta de vacantes	3	Despliega la lista de vacantes activas	E1

Excepciones:

Excepción	Nombre	Acción
E1	No existen vacantes activas	Indicar que por el momento no hay vacantes disponibles

Poscondiciones:

- El postulante puede revisar las vacantes disponibles.
- El sistema ha desplegado la lista de vacantes.

Caso de uso: Consulta especifica de vacantes
Actor: Postulante

Descripción: El postulante consulta un vacante de su interés.

Precondiciones:

- El postulante está interesado en enterarse en los detalles de una vacante en particular.
- El postulante revisó la lista general de vacantes.

Flujo:

ACTOR		SISTEMA		
Paso	**Acción**	**Paso**	**Acción**	**Excepción**
1	El postulante elige una vacante de su interés	2	El sistema carga la página con los datos de la vacante elegida	

Excepciones:

No hay

Poscondiciones:

- El postulante se encuentra en la vacante elegida.
- El sistema ha desplegado la información de la vacante de interés.

Caso de uso: Postulación
Actor: Postulante

Descripción: El actor se postula a una vacante de interés

Precondiciones:

- El postulante ha elegido una vacante de interés
- Ha decido postularse para dicha vacante.

Flujo:

ACTOR		**SISTEMA**		
Paso	**Acción**	**Paso**	**Acción**	**Excepción**
1	Manda petición para postularse para la vacante activa	2	Almacena los datos del postulante y envía mensaje de confirmación	E1

Excepciones:

Excepción	**Nombre**	**Acción**
E1	No se guardaron los datos	Enviar mensaje de error

Poscondiciones:

- El postulante queda en espera de noticias de su postulación

Caso de uso: Ingresar al sistema
Actor: Administrador

Descripción: El administrador ingresa al sistema después de que se valido como usuario privilegiado.

Precondiciones:

- Se cuenta con clave y password de usuario administrador
- El usuario esta posicionado en la página principal de acceso al sistema en la parte administrativa

Flujo:

ACTOR		**SISTEMA**		
Paso	**Acción**	**Paso**	**Acción**	**Excepción**
1	Ingresa login y password, posteriormente solicita ingresar al sistema	2	Muestra la pantalla principal del sistema administrativo	E1

Excepciones:

Excepción	**Nombre**	**Acción**
E1	El login y/o password es incorrecto	Enviar mensaje de error indicado que el acceso ha sido denegado

Poscondiciones:

- El administrador se encuentra en la pantalla principal
- El administrador ha entrado al sistema

Caso de uso: Alta Vacante
Actor: Administrador

Descripción: El administrador da de alta una vacante

Precondiciones:

- El administrador ha ingresado al sistema.

Flujo:

ACTOR		**SISTEMA**		
Paso	**Acción**	**Paso**	**Acción**	**Excepción**
1	Ingresa a alta de vacantes.	2	Despliega la forma para dar de alta una vacante	
3	Llena la forma y envía la petición de guardar	4	Guarda el registro de la nueva vacante en la base de datos	E1

Excepciones:

Excepción	Nombre	Acción
E1	No se pudo guardar la vacante	Enviar error de que no se pudo completar el guardado de la vacante

Poscondiciones:

- La vacante ha sido guardada en la base de datos.

Caso de uso: Modificación Vacante
Actor: Administrador

Descripción: El administrador modifica una vacante

Precondiciones:

- El administrador ha ingresado al sistema.

Flujo:

ACTOR		SISTEMA		
Paso	**Acción**	**Paso**	**Acción**	**Excepción**
1	Ingresa a consulta de vacantes y elige una vacante.	2	El sistema muestra los datos de la vacante que pueden ser editados	
3	Modifica los datos y envía la petición de actualizar	4	Guarda el registro de la vacante modificada en la base de datos	E1

Excepciones:

Excepción	Nombre	Acción
E1	No se pudo guardar la vacante	Enviar error de que no se pudo completar el guardado de la vacante

Poscondiciones:

- La vacante ha sido modificada y guardada en la base de datos.

Caso de uso: Baja Vacante
Actor: Administrador

Descripción: El administrador da de baja una vacante

Precondiciones:

- El administrador ha ingresado al sistema.

Flujo:

ACTOR		**SISTEMA**		
Paso	**Acción**	**Paso**	**Acción**	**Excepción**
1	Ingresa consulta de vacantes, elige una vacante	2	Despliega los datos de la vacante	
3	Elige dar de baja una vacante	4	Elimina el registro de la vacante en la base de datos	E1

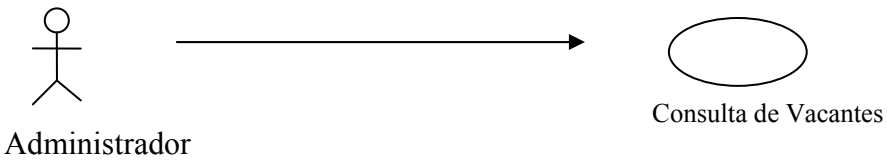
Excepciones:

Excepción	**Nombre**	**Acción**
E1	No se pudo eliminar la vacante	Enviar error de que no se pudo eliminar la vacante en la base de datos

Poscondiciones:

- La vacante ha sido eliminada en la base de datos.

Caso de uso: Consulta de Vacantes
Actor: Administrador



Descripción: El administrador consulta vacantes

Precondiciones:

- El administrador ha ingresado al sistema.
- El administrador se encuentra en la página principal de búsqueda de vacantes

Flujo:

ACTOR		SISTEMA		
Paso	**Acción**	**Paso**	**Acción**	**Excepción**
1	Elige los parámetros de búsqueda	3	Despliega la lista de vacantes activas	
2	Lanza una petición al sistema para la consulta de vacantes			

Excepciones:

No hay

Poscondiciones:

- El administrador puede revisar todas vacantes.
- El sistema ha desplegado la lista de vacantes

Caso de uso: Consulta especifica de vacante
Actor: Administrador

Descripción: El administrador consulta una vacante específica

Precondiciones:

- El administrador desea consultar una vacante.
- El administrador se encuentra en la lista general de vacantes

Flujo:

ACTOR		SISTEMA		
Paso	**Acción**	**Paso**	**Acción**	**Excepción**
1	El administrador elige una vacante	2	El sistema carga la página con los datos de la vacante elegida	

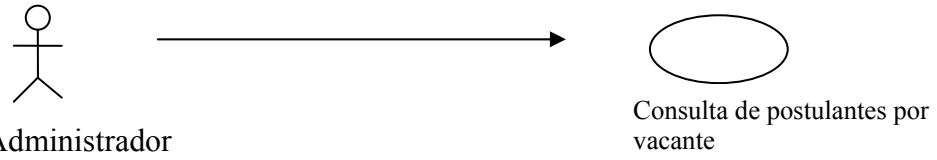
Excepciones:

No hay

Poscondiciones:

- El administrador se encuentra en la vacante elegida
- El sistema se ha desplegado la información de la vacante de interés.

Caso de uso: Consulta de postulantes por vacante
Actor: Administrador



Descripción: El administrador consulta los postulantes por vacante.

Precondiciones:

- El sistema se encuentra en la página de una vacante en particular

Flujo:

ACTOR		**SISTEMA**		
Paso	**Acción**	**Paso**	**Acción**	**Excepción**
1	Realiza la petición de consultar postulantes para la vacante actual	3	Despliega la lista de de postulantes	E1

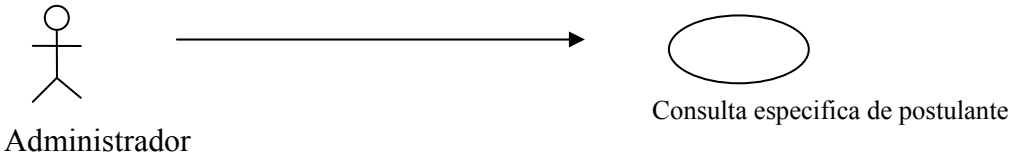
Excepciones:

Excepción	**Nombre**	**Acción**
E1	No existen postulantes	Indicar que por el momento no hay postulantes para la vacante

Poscondiciones:

- El administrador puede revisar los postulantes propuestos para la vacante
- El sistema ha desplegado la lista de postulantes para una vacante en especifico

Caso de uso: Consulta especifica de postulante
Actor: Administrador



Descripción: El administrador consulta cada uno de los postulantes para una vacante para ver si se adecua el perfil buscado.

Precondiciones:

- El administrador se encuentra en la lista general de postulantes para una vacante

Flujo:

ACTOR		**SISTEMA**		
Paso	**Acción**	**Paso**	**Acción**	**Excepción**
1	El administrador elige un postulante	2	El sistema carga la página con los datos del postulante elegido	

Excepciones:

No hay

Poscondiciones:

- El administrador se encuentra en la página del postulante elegido
- El sistema ha desplegado la información del postulante elegido.

Caso de uso: Cambiar status postulante
Actor: Administrador

Descripción: El administrador cambia status del postulante, si éste ha coincido o no con el perfil solicitado para la vacante.

Precondiciones:

- El administrador se encuentra en la página del los datos del postulante

Flujo:

ACTOR		SISTEMA		
Paso	**Acción**	**Paso**	**Acción**	**Excepción**
1	Envía al sistema la petición de cambiar status	2	Despliega la lista posibles status	
3	Elige el status que convenga al perfil del postulante	4	Despliega una pantalla donde pregunta el motivo del status	
5	Escribe o elige el motivo	6	Guarda el status y el motivo en la base de datos	

Excepciones:

Excepción	Nombre	Acción
E1	No se pudo actualizar el status	Indicar que por el momento no se puedo realizar el cambio de status

Poscondiciones:

- El administrador ha cambiado el status del postulante.
- El sistema ha hecho las modificaciones en la base de datos.

Caso de uso: Enviar curriculum
Actor: Administrador

Descripción: El administrador envía un currículo a una tercera persona

Precondiciones:

- El administrador se encuentra en la página del los datos del postulante

Flujo:

ACTOR		SISTEMA		
Paso	**Acción**	**Paso**	**Acción**	**Excepción**
1	Realiza la petición de enviar currículo	2	El sistema abre una ventana para pedir la dirección e-mail	

			donde desea enviarse	
3	Coloca la dirección y elige enviar	4	Envía el correo a la dirección elegida	

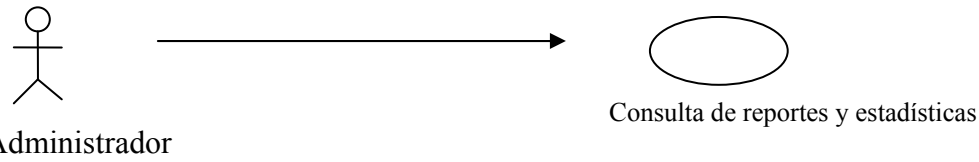
Excepciones:

No hay

Poscondiciones:

- El administrador ha enviado un correo a un tercero.
- El sistema ha procesado el envío del correo.

Caso de uso: Consulta de reportes y estadísticas
Actor: Administrador



Descripción: El administrador consulta reportes y estadísticas sobre vacantes, postulantes, status.

Precondiciones:

- El administrador ha ingresado al sistema.
- El sistema se encuentra en la página principal de reportes y estadísticas

Flujo:

ACTOR		**SISTEMA**		
Paso	**Acción**	**Paso**	**Acción**	**Excepción**
1	Elige los parámetros de búsqueda	2	Despliega el reporte elegido con los parámetros elegidos	

Excepciones:

No hay

Poscondiciones:

- El administrador revisa los reportes desplegados
- El sistema despliega los reportes que le han sido solicitados.

4.5 Prototipar la interfaz de usuario

- Creación del diseño lógico de una interfaz de usuario:

La manipulación de la información será mediante formas, para el despliegue de la información se utilizarán tablas con la información que se pide dentro de la descripción del problema.

La apariencia será señalando el nombre de sistema y del área de Recursos humanos, ya sea con logotipos o con el nombre del sistema.

- Creación del diseño y prototipos físicos de interfaz de usuario:

A continuación se muestra algunos prototipos físicos de la interfaz de usuario:

En la figura 4.4 se muestra la pantalla de bienvenida para el empleado (postulante) donde se pueden leer las políticas al ingresar al sistema y el postulante acepta las reglas.

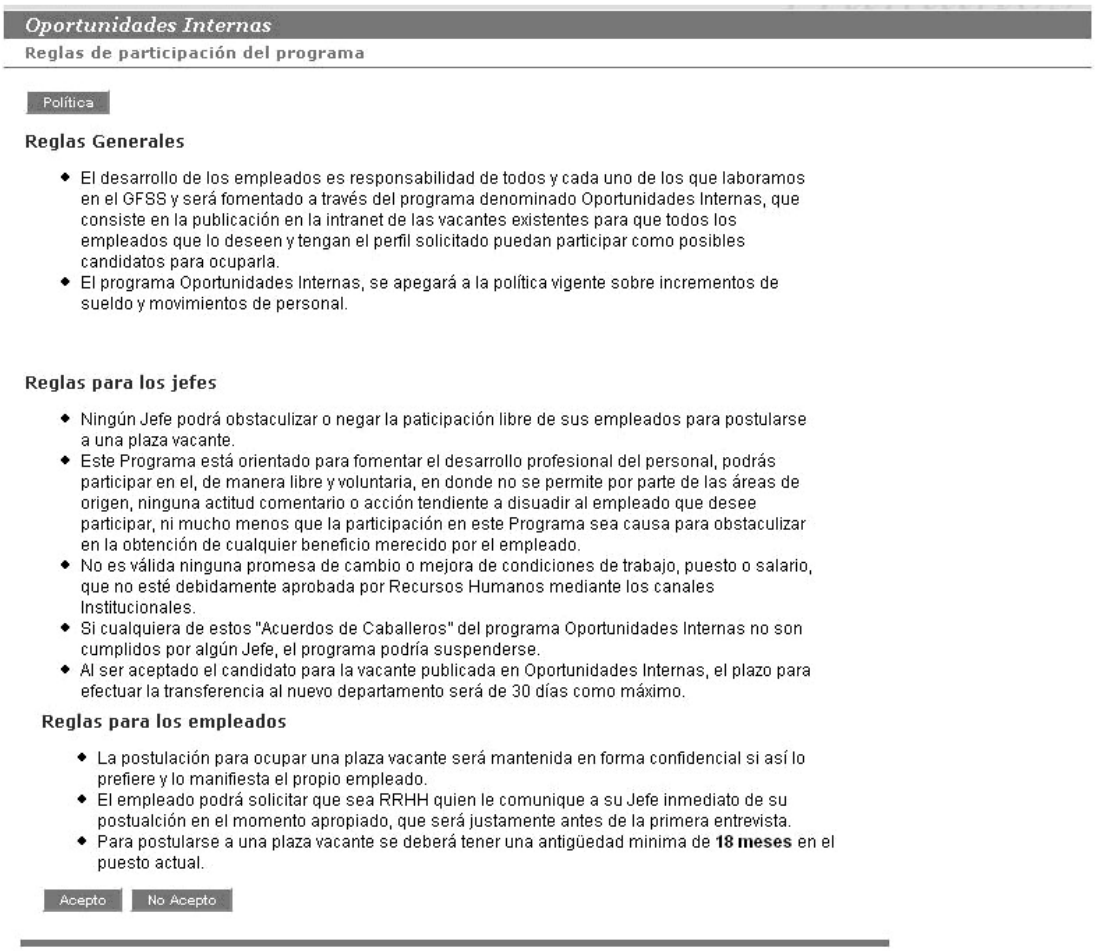


Figura 4.4 Politicas del programa Oportunidades Internas

Después de elegir la vacante de su interés el postulante deberá ingresar su número de expediente, y los datos de la forma serán actualizados por la base de datos de empleados existente, en caso de que no existan los datos se pedirá ingresar al Sistema de Actualización de Datos, como se muestra en la figura 4.5

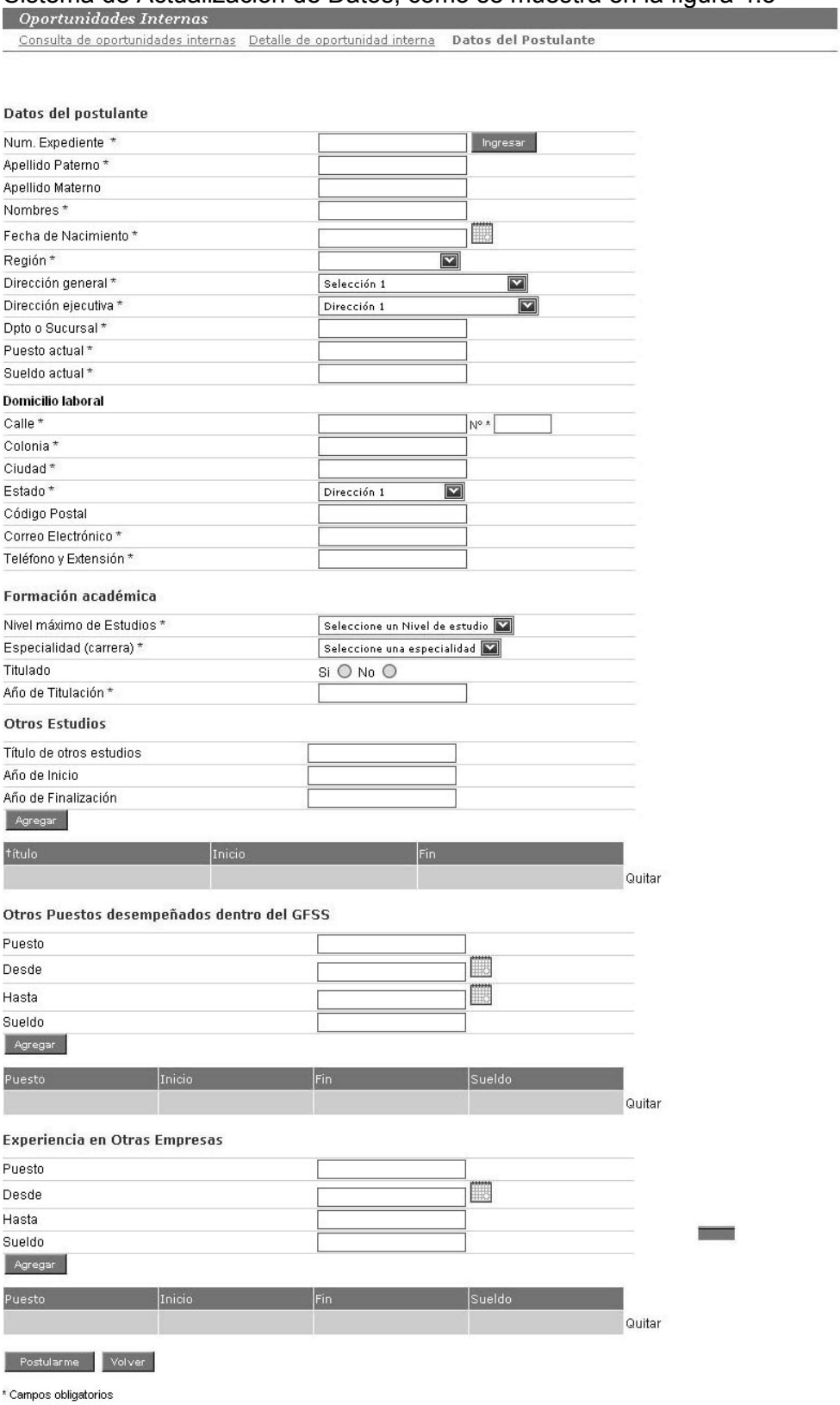


Figura 4.5 Forma de obtención y/o actualización de datos

En la figura 4.6 se muestra la pantalla de consulta por región para vacantes disponibles que el postulante podrá ver, una vez aceptada las políticas del programa de Oportunidades Internas.

Figura 4.6 Consulta de Vacantes disponibles

Por otro lado, la pantalla de acceso para el administrador al sistema se muestra en la figura 4.7.

Figura 4.7 Pantalla de acceso para el administrador.

El prototipo para la pantalla de alta de vacantes se plantea en la figura 4.8.

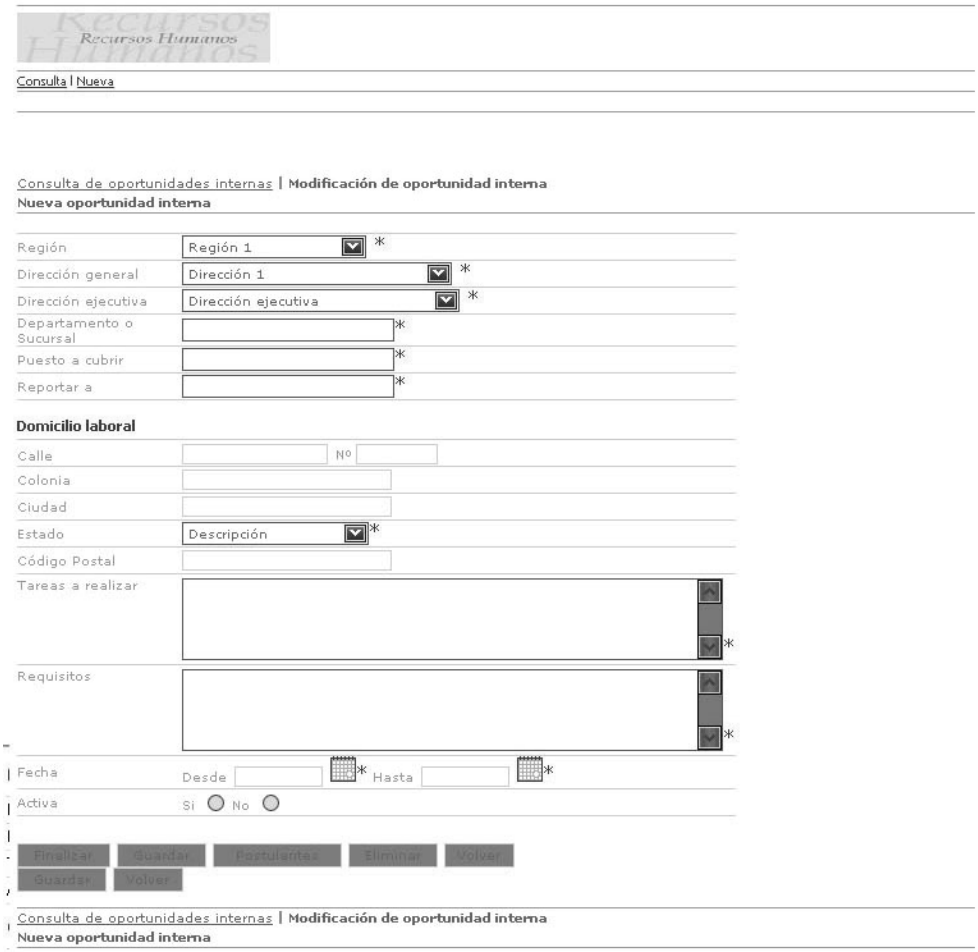


Figura 4.8 Pantalla alta vacante.

El prototipo para que el administrador pueda sonsultar los postulantes por cada vacante y realizar la selección se muestra en la figura 4.9.

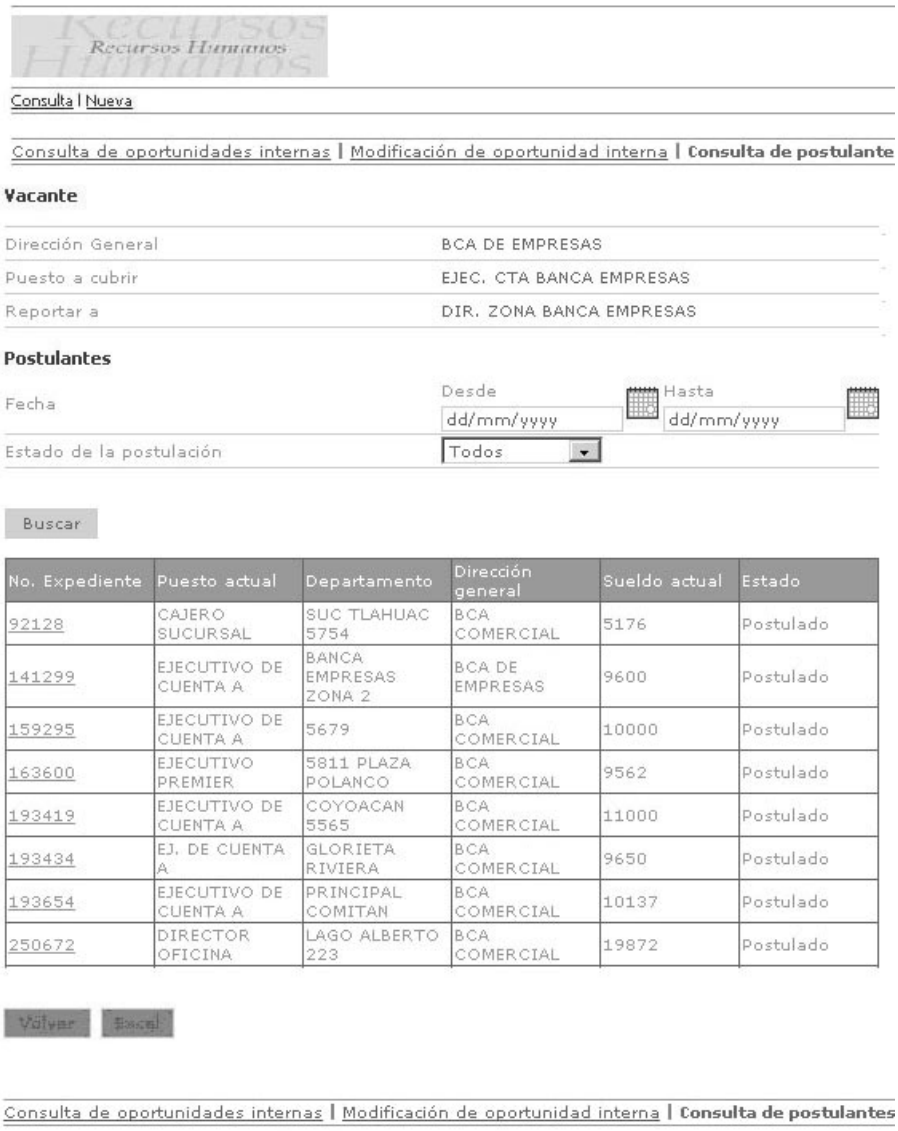


Fig 4.9 Pantalla consulta de postulantes por vacante.

Capítulo 5

Análisis

En este capítulo se realizarán las actividades de análisis, aplicadas al proyecto en base a los requerimientos dados en el capitulo anterior.

5.1 Análisis de la arquitectura

- Identificación de paquetes del análisis

Proporcionan un medio para organizar el modelo de análisis en piezas más pequeñas y manejables. A continuación se realiza la identificación de paquetes de servicio que encapsulan clases relacionadas funcionalmente. En la figura 5.1, 5.2 y 5.3 se muestran los paquetes identificados para cada servicio dentro del sistema.

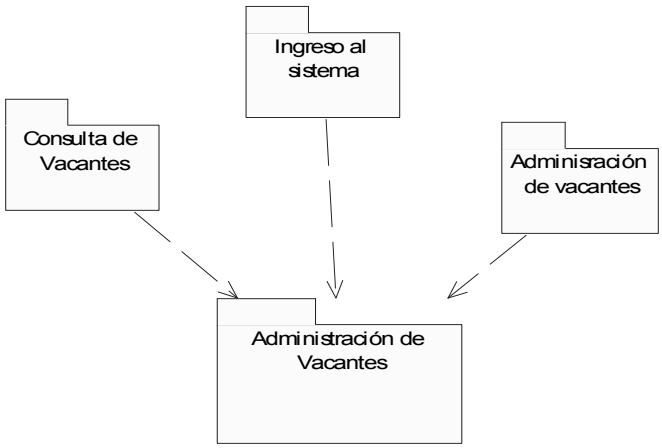


Figura 5.1 Paquetes del análisis que contribuyen al servicio para Vacantes

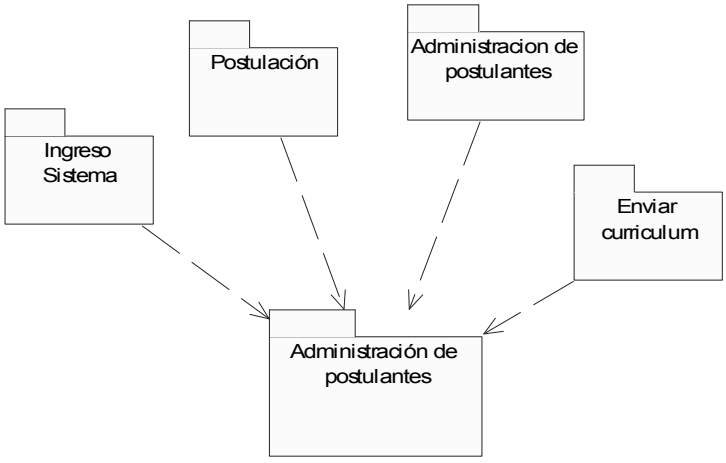


Figura 5.2 Paquetes del análisis para el servicio de Postulantes

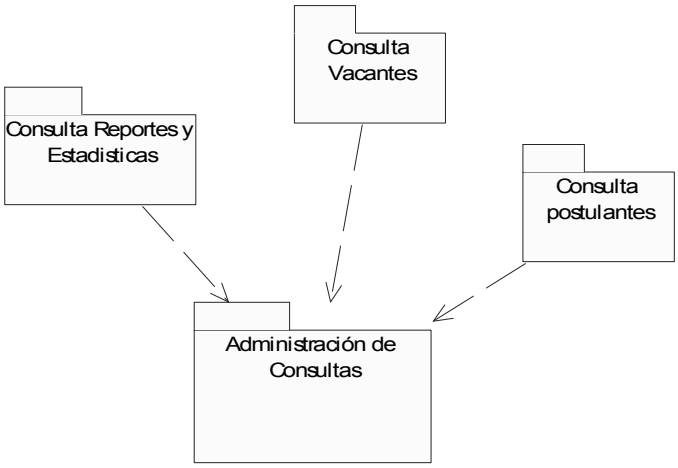


Figura 5.3 Paquetes de Análisis para el servicio de Consultas

- Identificación de clases obvias. Se realiza una propuesta preliminar en base a las clase de entidad basadas en las clases del modelo del dominio.

    - Postulante
    - Administrador
    - Vacante
    - Postulación
    - Curriculum

- Identificación de requisitos especiales comunes

  El sistema trabajará dentro de la intranet de la institución.

  La seguridad estará separada en dos módulos:
  Acceso para Administrador
  Acceso para Postulante (empleado)

5.2 Analizando casos de uso

Los casos de uso se analizaran a partir de la captura detallada de requisitos como casos de uso.

- Identificación de las clases del análisis.

  En este paso se identifican las clases de control, entidad e interfaz necesarias para realizar los casos de uso. En esta sección también se muestran los resultados de realizar las actividades de identificación de atributos, identificación de responsabilidades, identificación de agregaciones y asociaciones.

- Descripción de interacciones entre objetos del análisis

Cuando se tiene un esbozo de las clases necesarias para realizar los casos de uso, se debe describir cómo interactúan sus correspondientes objetos del análisis. A continuación las figuras 5.4, 5.5 muestran el diagrama de secuencia para un ingreso al

sistema del postulante para un flujo normal excepcional, en las figuras 5.6 y 5.7 se muestran los diagramas de secuencia para la postulación, para un flujo normal y excepcional, respectivamente.

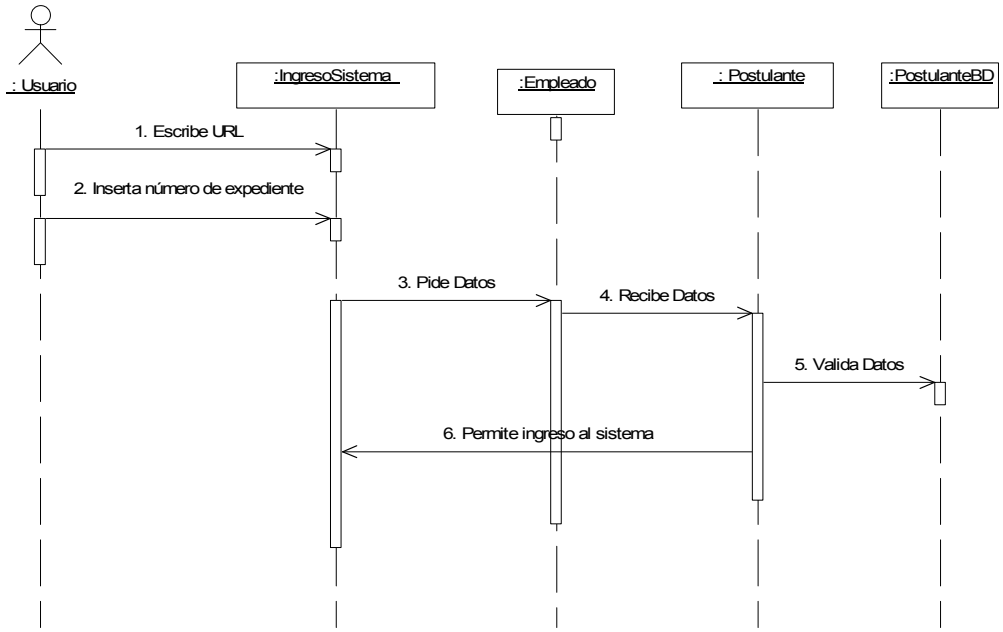


Figura 5.4 Ingreso al sistema del Postulante válido.

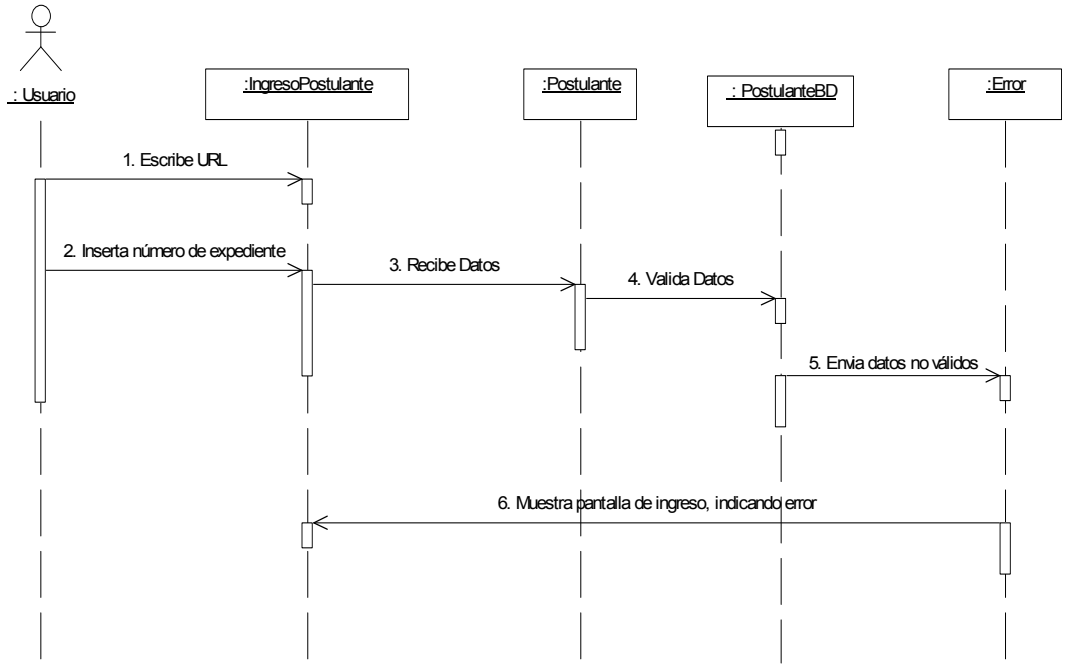


Figura 5.5 Ingresar al sistema Postulante no válido.

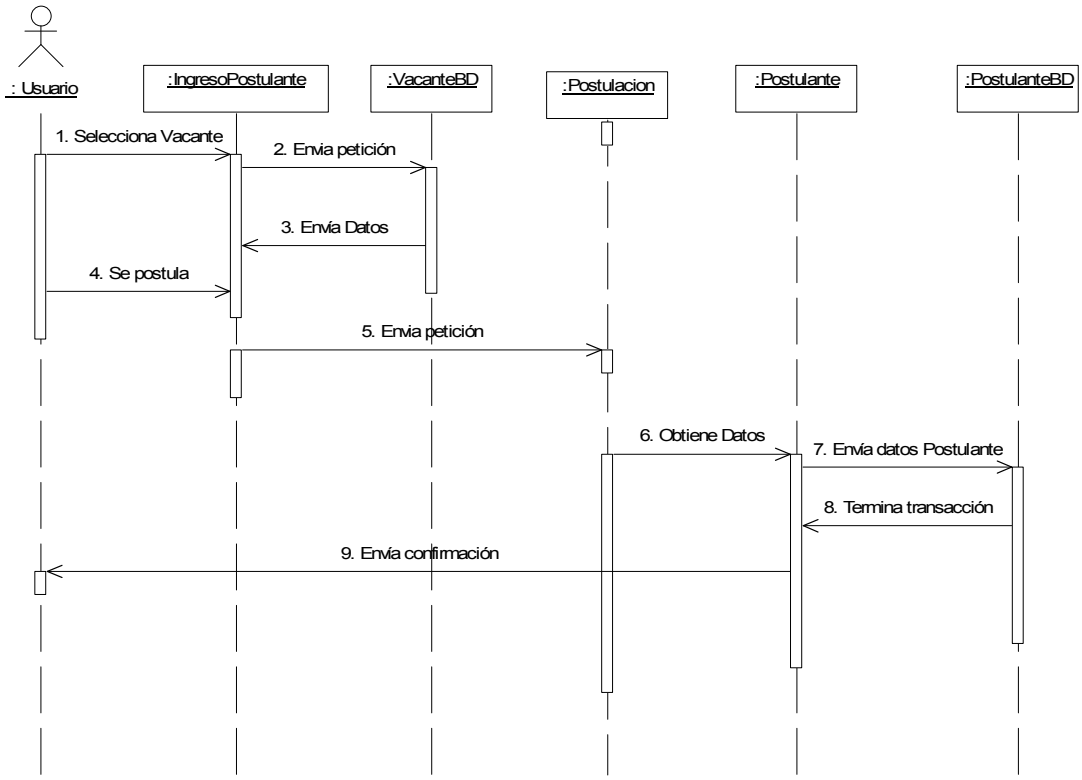


Figura 5.6 Postulación válida.

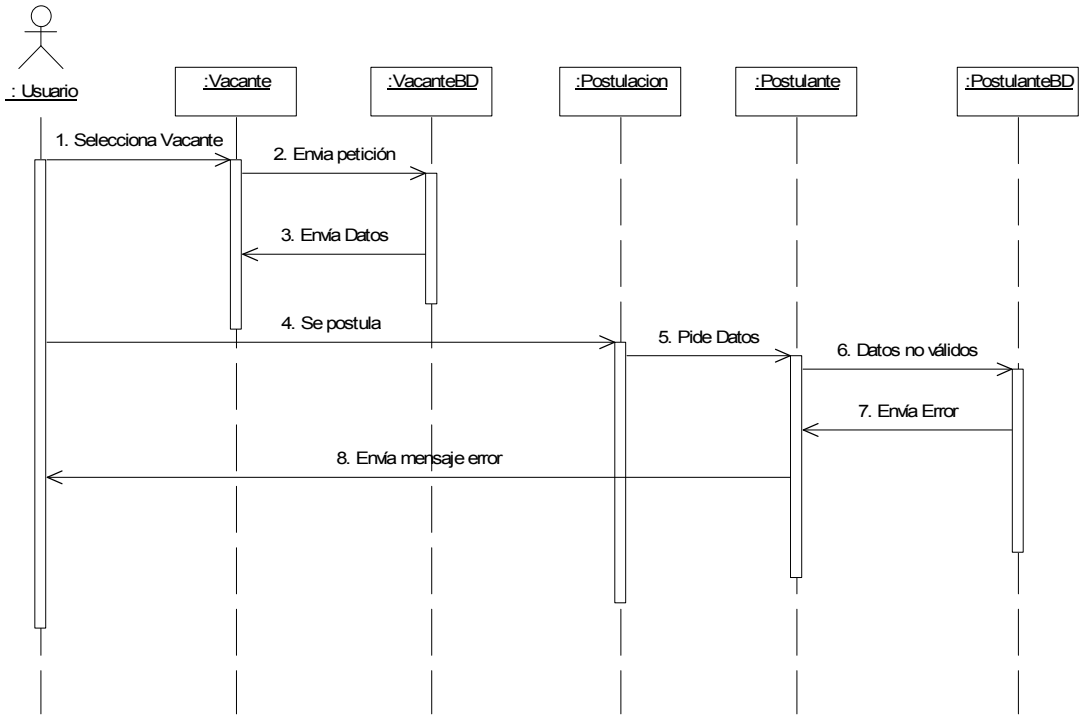


Figura 5.7 Postulación no válida.

5.3 Analizar una clase

Se identificaron los atributos y las responsabilidades para cada clase basadas en su papel en las realizaciones de los casos de uso.

Se identificaron las clases de entidad a partir del estudio del modelo del dominio, en la figura 5.8 se muestra el diagrama de clases de las clases de Entidad.

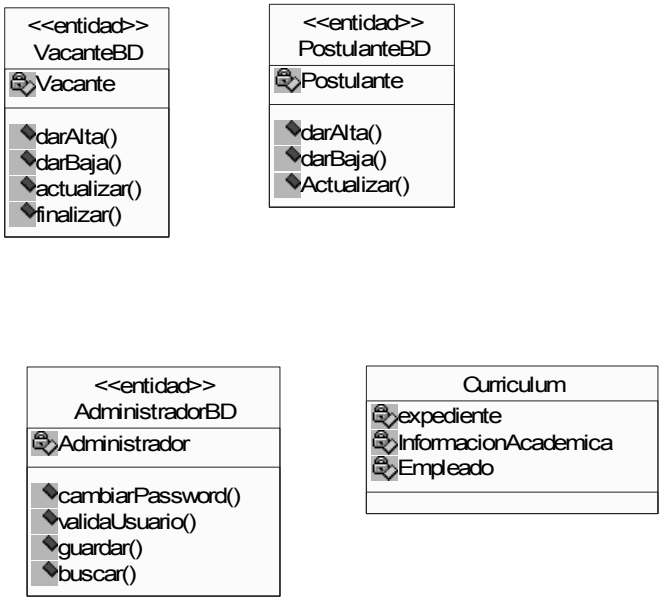


Figura 5.8 Clases de Entidad

Se identifican las clases de control responsables de la coordinación de las clases de entidad para la realización de los casos de uso. En la figura 5.9 se muestra el diagrama de clases de clases de control.

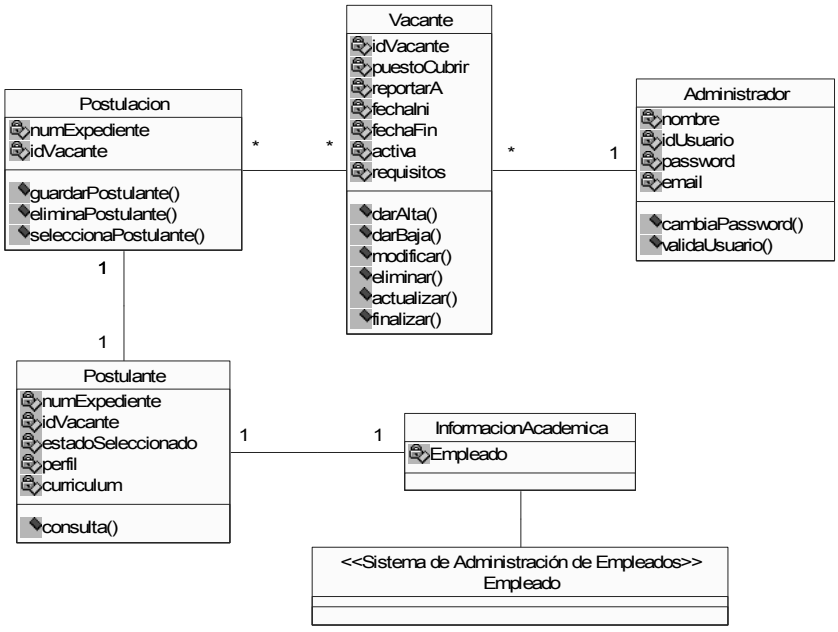


Figura 5.9 Clases de Control

Se identifica una clase interfaz primitiva por cada clase de entidad, como se muestra en el diagrama de clases de la figura 5.10

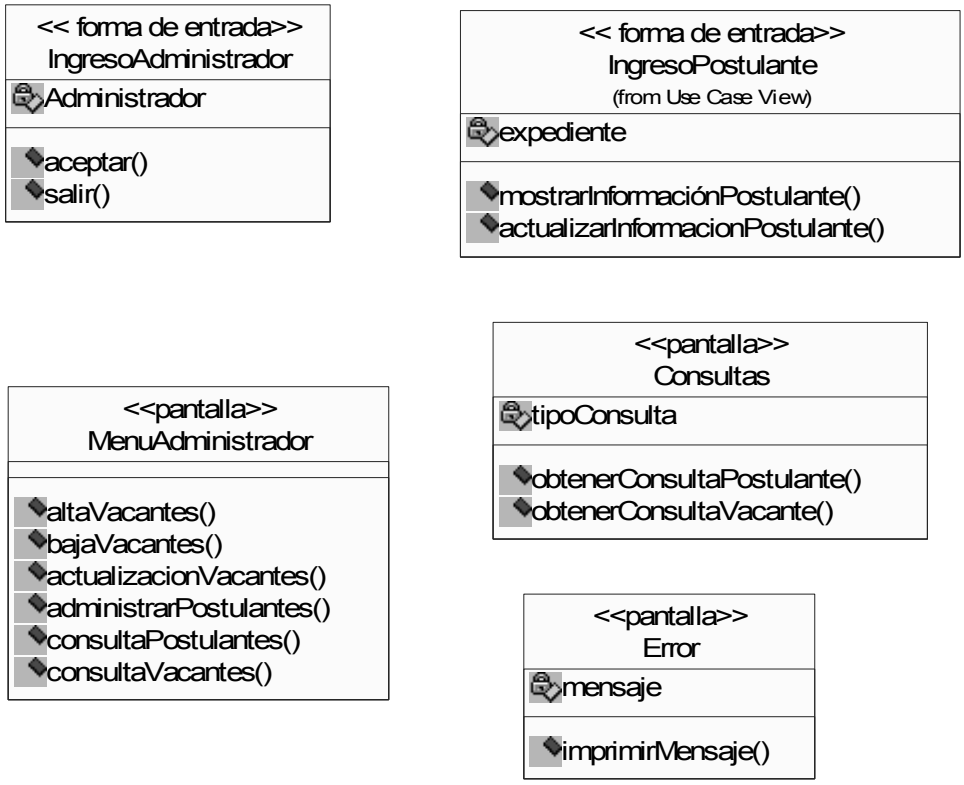


Fig 5.10 Clases de Interfaz

Capítulo 6

Diseño

En este capítulo se desarrollará el diseño del sistema, realizando las actividades descritas en la sección 1.5

6.1 Diseño de la arquitectura

- Identificación de nodos configuraciones de red

El sistema se ejecutará sobre dos servidores, uno que corresponde al servidor de la institución dedicado al área de recursos humanos que ejecuta los componentes web y que solo se encuentra disponible en la intranet y otro dedicado a base de datos con la información de los empleados de la institución, el esquema de configuración de red se muestra en la figura 6.1.

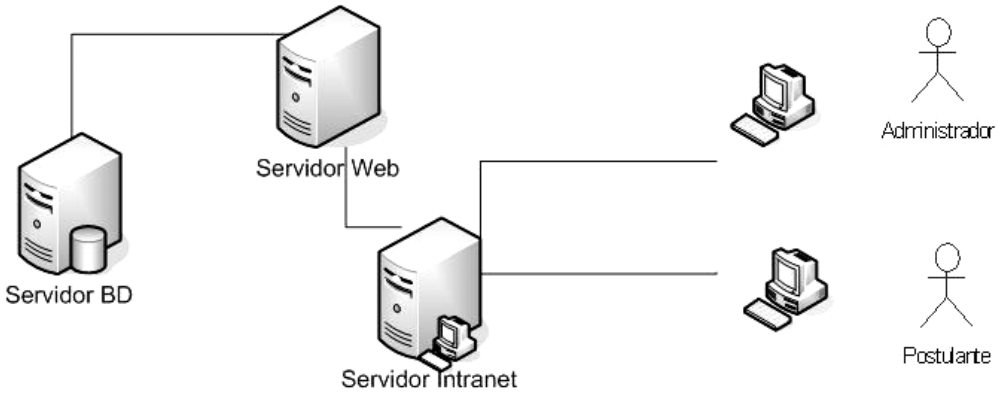


Figura 6.1 Diagrama de distribución para el sistema de Oportunidades Internas.

- Identificación de subsistemas y de sus interfaces

La siguiente figura muestra los subsistemas, la definición de subsistemas y capas del sistema Oportunidades Internas utilizando la tecnología J2EE.

A bajo nivel el software del sistema se construirá sobre TCP/IP para la comunicación dentro de la intranet, la capa intermedia esta compuesta por las bibliotecas de java que hacen posible la ejecución, y el despliegue dentro del navegador.

Las capas que se muestran en la figura 6.2 representan la parte administrativa a la que solo los usuarios privilegiados tienen acceso y se encuentran dentro de la misma interfaz de acceso, por ultimo la capa especifica de la aplicación a la que todos los usuarios tienen acceso.

- Identificación de clases del diseño relevantes para la arquitectura

Se identifican las clases a partir de las clases del análisis.

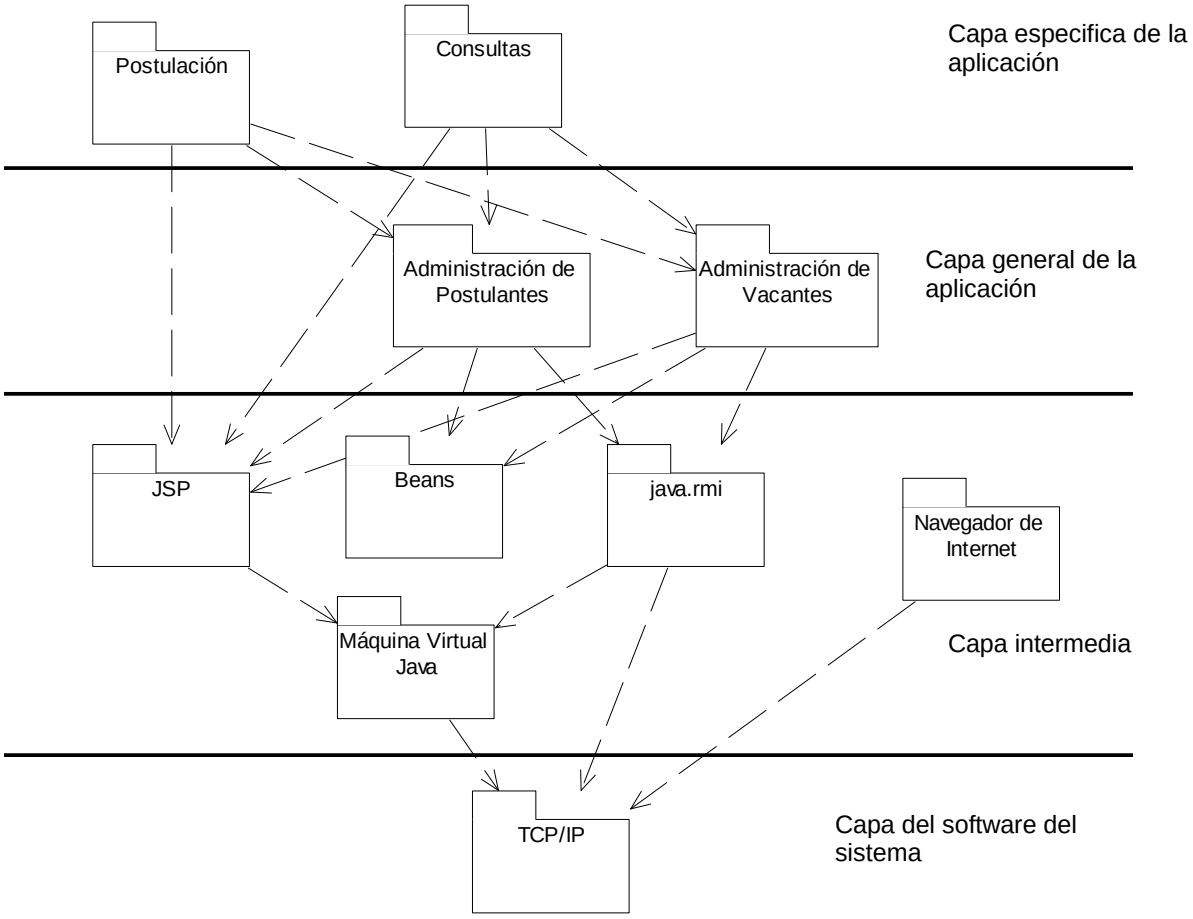


Figura 6.2 Diagrama de dependencias y capas de algunos de los subsistemas.

- Identificación de mecanismos genéricos de diseño

Se identifican los objetos que deben ser persistentes, como el objeto Postulación, para conseguirlo se utiliza un sistema de gestión de base de datos relacional.

Para que el sistema sea seguro, también se identifican los objetos que necesitaran validarse permanentemente dentro de la navegación del sistema Administrador y Postulante

6.2 Diseño de casos de uso

- Identificación de clases de diseño participantes

  En la figura 6.3 se identifican las clases del diseño que se necesitan para realizar el caso de uso de postulación.

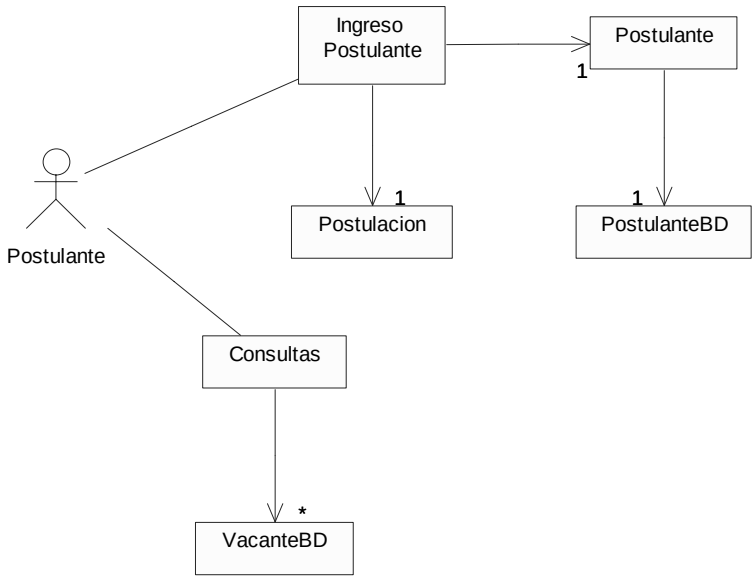


Figura 6.3 Se muestran las clases activas del caso de uso de postulación.

- Descripción de las iteraciones entre objetos del diseño

  Utilizando el esquema anterior se realiza un esquema de interacción de los objetos de las clases de diseño necesarias para realizar el caso de uso de postulación, este diagrama se despliega en la figura 6.4.

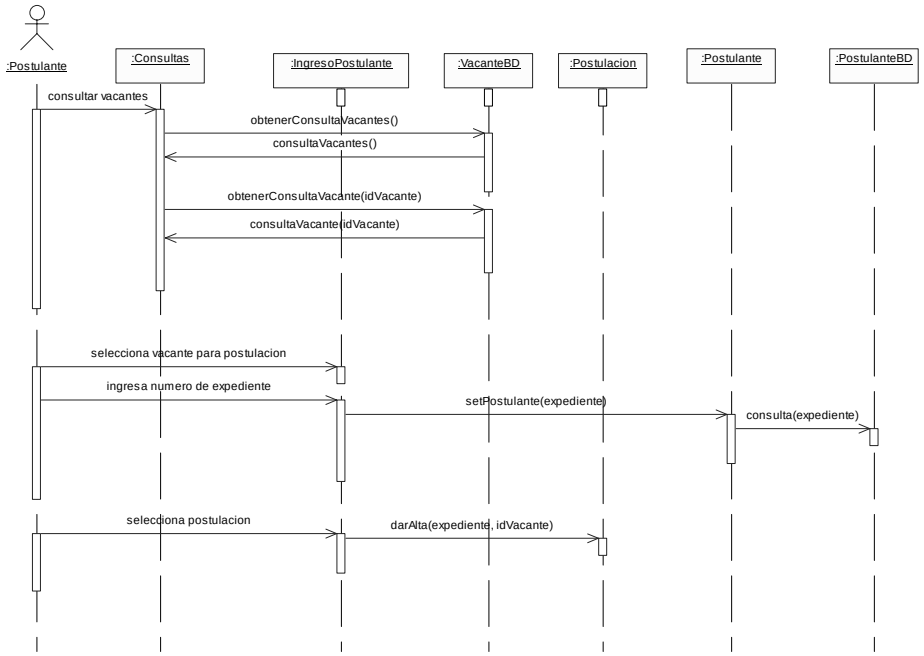


Figura 6.4 Diagrama de interacción de objetos del diseño.

6.3 Diseño de clases

- Identificación de las clases del diseño.

  En este paso se identifican las clases de control, entidad e interfaz necesarias para realizar los casos de uso. En esta sección también se muestran los resultados de realizar las actividades de identificación de atributos, identificación de responsabilidades, identificación de agregaciones y asociaciones. Basándose en las clases del análisis los métodos son descritos utilizando el lenguaje a utilizar Java. En la figura 6.5 se muestran las clases de entidad y sus relaciones, análogamente en las figuras 6.6 y 6.7 las clases de control de interfaz, respectivamente.

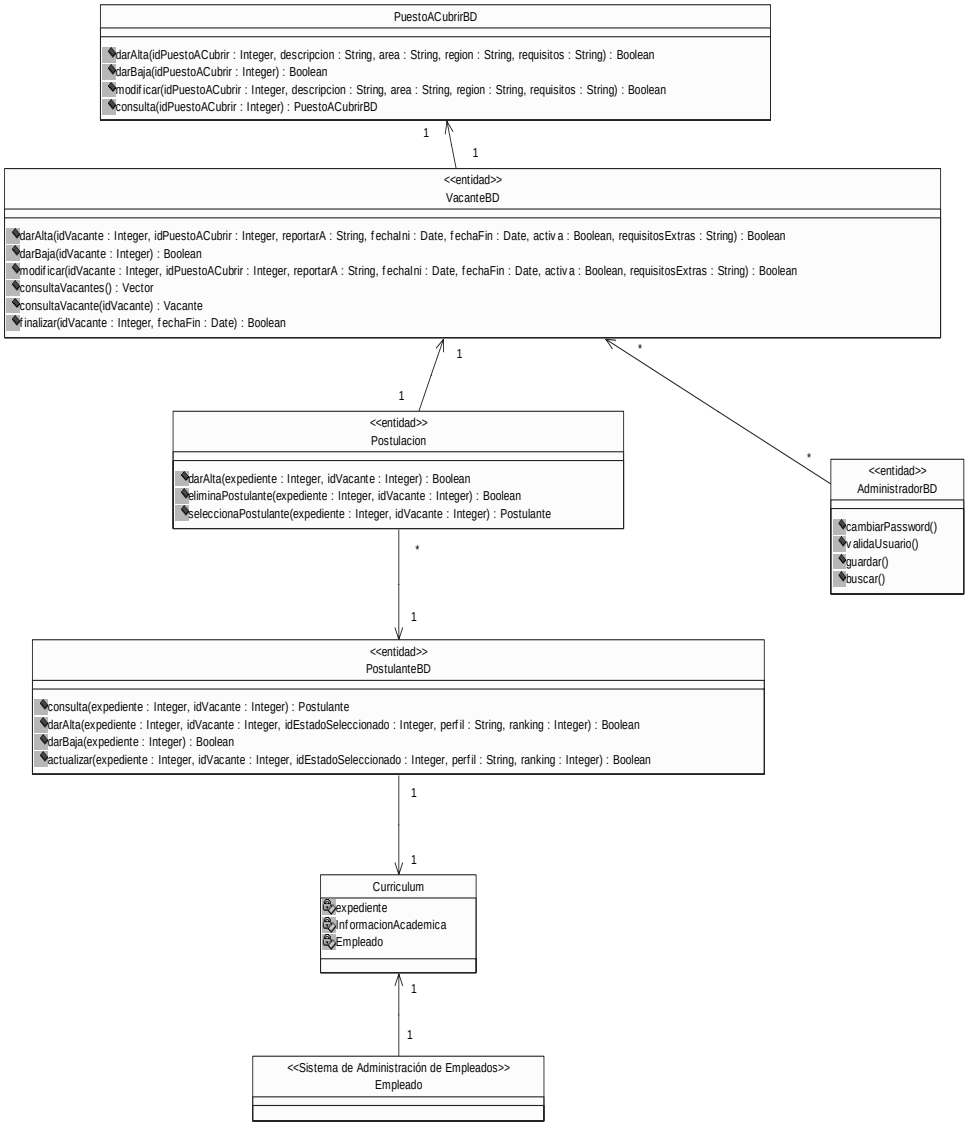


Figura 6.5 Diagrama de las clases de entidad

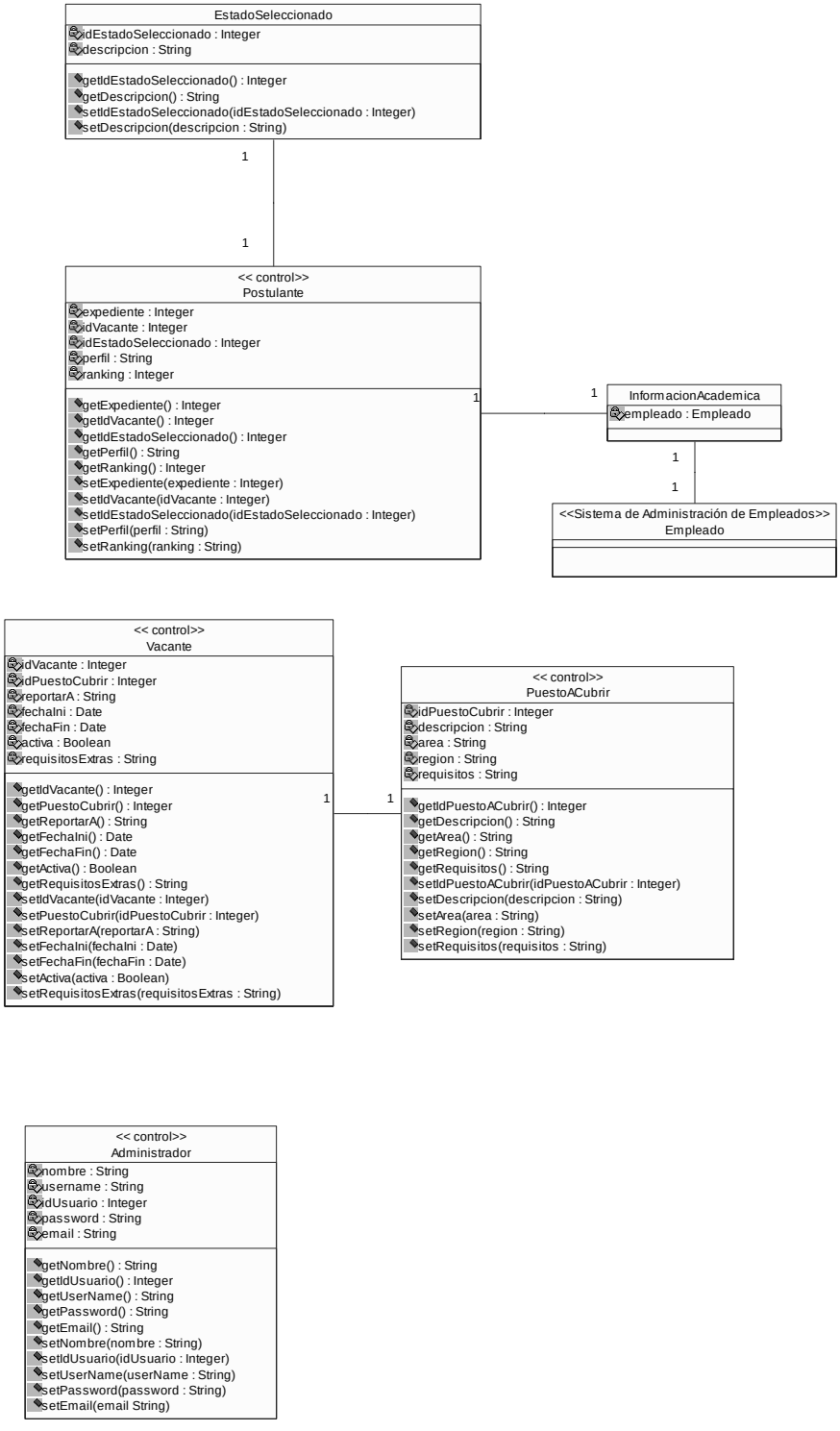


Figura 6.6 Diagrama de clases de control

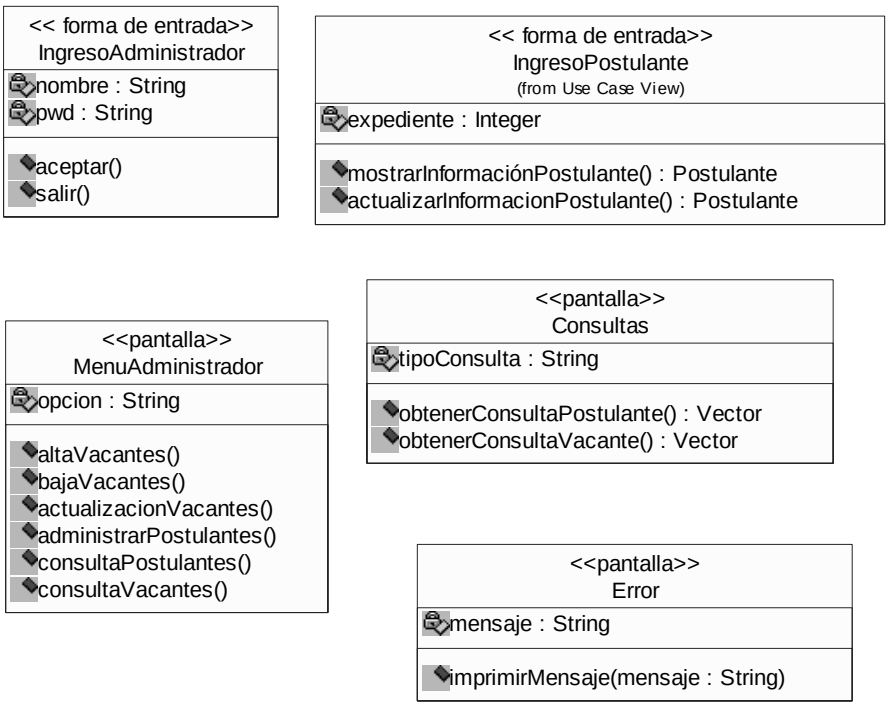


Figura 6.7 Diagrama de clases de interfaz

Conclusiones

.

Los casos de usos son un buen punto de partida para los siguientes flujos de trabajo: análisis y diseño. Los casos dirigieron el trabajo a lo largo del flujo de trabajo iteración por iteración.

Del análisis se obtuvo un modelo de objetos conceptual que analiza los requisitos mediante su refinamiento y estructuración.

El principal resultado del diseño es el modelo de diseño que se esfuerza para conservar la estructura del sistema impuesta por el modelo del análisis, y que sirve como esquema para la implementación.

La implementación y pruebas del sistema quedaron fuera del alcance del proyecto, debido a que el trabajo propuesto llego hasta fase de diseño.

Se ha diseñado un sistema, siguiendo el Proceso Unificado de Software, con lo que se ha obtenido un diseño con bases y listo para comenzar la fase de implementación y tomar en cuenta las fases desarrolladas para el siguiente ciclo de vida.

Anexo 1

En este anexo se revisará la evolución de UML, y se verán los principales cambios, como una manera de ver como podría enriquecerse el Proceso Unificado utilizando los nuevos diagramas. En el presente trabajo se utilizó la versión UML 1.5.

Se utilizó la versión 1.5 porque para aplicar el Proceso Unificado de Software en el sistema de Oportunidades Internas no era necesario optar por los nuevos elementos que ofrece UML 2.0 ya que la complejidad y la realización de la actividades del proceso no requerían la utilización de éstos, además de ser la versión que se encuentra aún en mayor circulación en el ambiente de desarrollo, ya que UML 2.0 aun no se termina de adoptar en todas las herramientas de modelado.

UML

La primera versión de UML permitió a las personas comunicar diseños sin ambigüedad, que mostraban la esencia de un diseño, e incluso que mapeaba los requisitos funcionales a las soluciones de software. Sin embargo, el mundo cambió con el conocimiento de que los sistemas podían ser modelados, es decir que el software que se diseñaba, podría también beneficiarse de un lenguaje unificado tal como UML.

Los factores que conducían del desarrollo de software orientado a componentes,, de UML ejecutable, y de la necesidad de compartir modelos entre diversas herramientas pusieron demandas en UML que no habían sido consideradas en el diseño original para resolver.

UML 1.x y todas sus revisiones anteriores fueron diseñados como lengua unificada para los seres humanos. Cuando empezó a importar compartir modelos en diferentes medios, se utilizaron herramientas para UML 1.x. Las reglas de la notación de UML 1.x's y su meta-modelo no definido formalmente fueron suficientes para permitir compartir modelos máquina a máquina.

Aunque UML 1.5 describió un sistema bastante completo, el modelo que describe el meta-modelo se fue parchando y volviéndose excesivamente complejo. UML había llegado a ser excesivamente complejo, frágil, y difícil de extender; era hora para una re-arquitectura.

El Nuevo Enfoque en UML 2.0

En las versiones previas del UML, se hacía un fuerte hincapié en que UML **no** era un lenguaje de programación. Un modelo creado mediante UML no podía ejecutarse. En el UML 2.0, esta idea cambió de manera drástica y se modificó el lenguaje, de manera tal que permitiera capturar mucho más comportamiento *(Behavior*). De esta forma, se permitió la creación de herramientas que soporten la automatización y generación de código ejecutable, a partir de modelos UML.

Estándares que conforman el UML

- Superestructura: Es la especificación que usamos todos los días. Aquí se encuentran todos los diagramas que la mayoría de los desarrolladores conocen.
- Infraestructura: Conceptos de bajo nivel. **Meta-Modelo** da soporte a la superestructura, entre otras.
- OCL: Lenguaje de restricción. De utilidad para especificar conceptos ambiguos sobre los distintos elementos del diagrama.
- XMI / Intercambio de diagramas: Permite compartir diagramas entre diferentes herramientas de modelado UML.

Reestructuración del Lenguaje

Para lograr los objetivos enunciados, varios aspectos del lenguaje fueron reestructurados y/o modificados. La especificación se separó en cuatro especificaciones (paquetes) bien definidas, tal como se muestra en la figura A.1 Es interesante destacar que el UML 2.0 puede definirse a sí mismo. Es decir, su estructura y organización es modelable utilizando el propio UML 2.0; de esta manera, se da un ejemplo de utilización del UML en un dominio distinto al del desarrollo de software. En este caso, cada paquete del diagrama representa cada una de las cuatro especificaciones que componen el lenguaje.

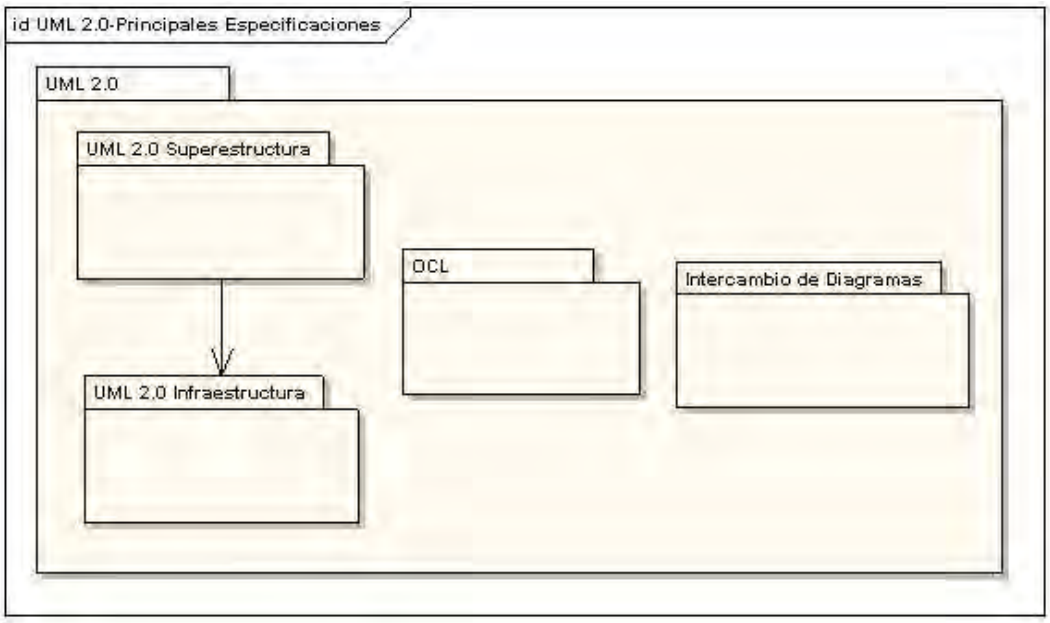


Figura A.1 Especificaciones principales del UML 2.0

En la figura A.2 se muestra como esta constituido UML 1.x por siete diagramas básicos y dos diagramas que constituyen variaciones de dos de los anteriores.

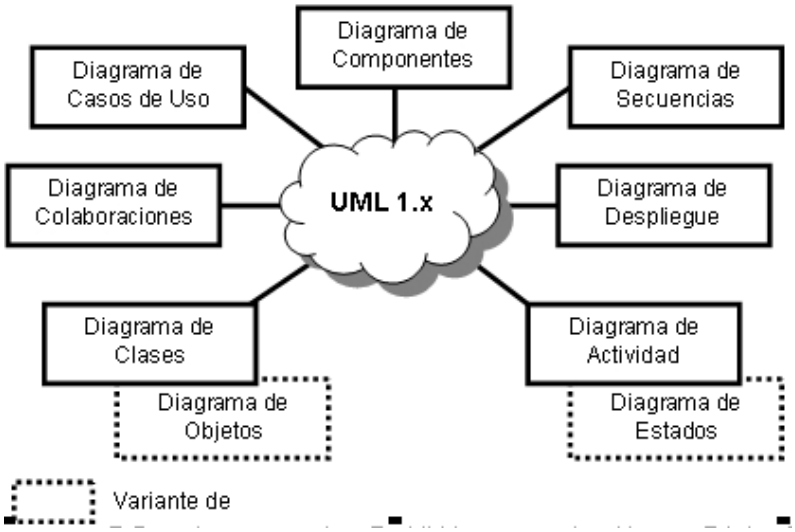


Figura A.2 Diagramas de UML 1.x [OMG]

En UML 2.0 se definen una serie de diagramas adicionales a los establecidos en UML 1.x. El conjunto de diagramas se encuentra organizado en torno a dos categorías: diagramas estructurales (ubicados en la parte superior e izquierda) y diagramas dinámicos o de comportamiento (ubicado en la parte inferior y derecha), en particular los diagramas de interacción se representan en el recuadro. Los diferentes diagramas son indicados en la figura A.3.

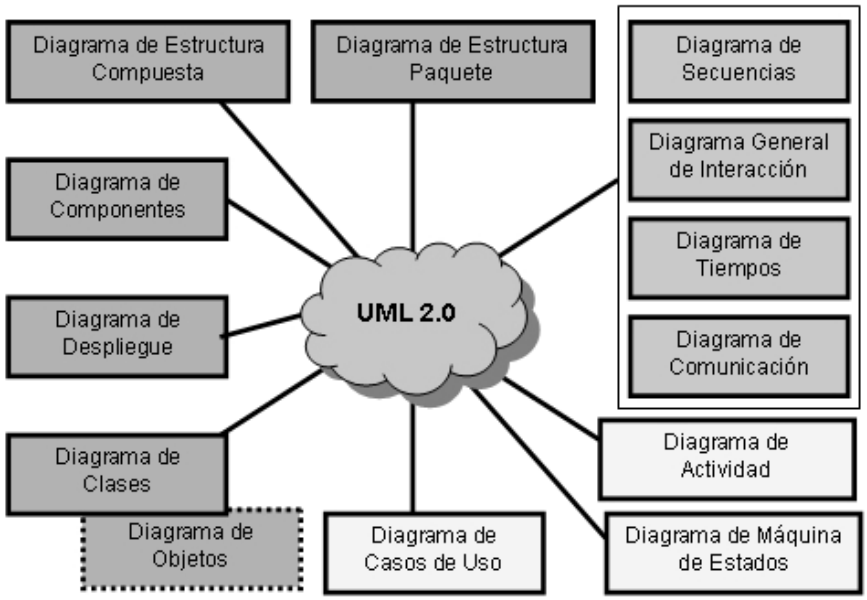


Figura A.3 Diagramas de UML 2.0 [OMG]

UML 2.0 es la mayor revisión que se le ha hecho a UML desde la versión 1.0. El modelo conceptual ha sido reestructurado completamente y nuevos diagramas han sido incorporados. Los diagramas tradicionales también han sido mejorados. La nueva versión permitirá a los fabricantes de herramientas CASE proporcionar a los analistas, arquitectos y desarrolladores, herramientas cada vez más potentes que les permitan aprovechar mejor los modelos y como consecuencia generar una mayor cantidad código reduciendo significativamente el ciclo de desarrollo de sus aplicaciones.

A continuación en la tabla A.1 se presentan los cambios de diagramas en las versión de UML.

Diagrama	Qué modela?	A partir de UML 1.x ó UML 2.0	Cambios significativos entre versiones
Casos de Uso	Relación entre los actores y el sistema, y los casos de uso. También ayuda a mapear los requerimientos del sistema.	UML 1.x	Se agregaron estructuras compuestas: un Caso de Uso puede ser parte de cualquier clasificador (no solamente packages). Por ejemplo, un caso de uso puede ser parte de una clase.
Actividad	Actividades paralelas y secuenciales dentro del sistema.	UML 1.x	UML 1 manejaba los diagramas de estados como un caso especial de diagrama de actividad. UML 2 rompe esta dependencia y como resultado remueve las reglas de patrones de procesos. El resultado es un mayor entendimiento del flujo y de la transición de estados. Aparecen nuevos elementos: tiempo, señales aceptadas, flujo de transformación, subdiagramas, entre otros.
Clases	Clases, tipos e interfaces y las relaciones entre ellas	UML 1.x	Las multiplicidades discontinuas fueron eliminadas. Se han agregado diferentes palabras claves de dependencia común, las palabras <<parameter>> y <<local>>.
Objetos	Instancias de Objetos entre las clases definidas en los diagramas de clases en configuraciones que son importantes en el sistema	UML 1.x	ninguno.
Secuencia	Interacciones entre los objetos donde el orden de las interacciones es importante	UML 1.x	Las modificaciones de los diagramas de secuencias básicamente fueron permitir la reutilización de los diagramas, agregando los elementos de tipos Fragmento Combinado y Operadores de Interacción que contienen un cierto número de operandos y un identificador en el pentágono superior izquierdo del operador.
Comunicación	Los diferentes caminos en los cuales los objetos interactúan y las conexiones que son necesarias para soporta esta interacción	UML 1.x	Se renombró a partir de los diagramas de colaboración. Por ser las colaboraciones un diagrama de interacción, al igual que los diagramas de secuencias, heredan la misma

			capacidad de soportar fragmentos combinados.
Tiempo	Cambios en el estado, o la condición, de una línea de vida de una instancia (de un Clasificador o un Rol de un clasificador), a lo largo del tiempo y de manera lineal. El uso más común es mostrar el cambio de estado de un objeto a lo largo del tiempo, en respuesta a los eventos o estímulos aceptados	UML 2.0	nuevo
Revisión de Interacción	Colecta diagramas de secuencia, comunicación y tiempo para capturar las interacciones más importantes que ocurren dentro del sistema.	UML 2.0	nuevo
Estructura compuesta	Las partes internas de una clase y sus componentes, pueden describir relaciones entre clases en un contexto dado.	UML 2.0	nuevo
Componentes	Los componentes importantes dentro del sistema y las interfaces que usan para interactuar con cada otro.	UML 1.x	Toma un nuevo significado en UML 2.0. El cambio en la notación de los componentes. Ahora se notan como un estereotipo, en vez de los dos rectángulos pequeños cruzados. Otro cambio importante, son los conectores de ensamblado (assembly connectors) que se utilizan para representar las interfaces provistas y requeridas por un componente.
Paquetes	La organización jerárquica de grupos de clases y sus componentes	UML 2.0	nuevo
Maquina de estados	El estado de los objetos durante su tiempo de vida y los eventos que pueden alterarlo.	UML 1.x	UML 1.x separaba las acciones de corta vida de las actividades de larga vida. UML 2.0 utiliza ambas acciones a la vez y usa el termino *do-activity* para las actividades de larga duración.
Despliegue	Representa la arquitectura de ejecución de un sistema. Éste, representa los artefactos del sistema como nodos, los cuales son conectados a través de caminos de comunicación para crear redes de sistemas de complejidad arbitraria	UML 1.x	A los antiguos diagramas de despliegue se le agregaron los siguientes elementos: -Artefactos UML 2.0 permite representar cualquier elemento empaquetable. Los artefactos pueden tener atributos. - Especificaciones de despliegue son una colección de propiedades (properties) que especifican cómo un artefacto será desplegado.

Tabla A.1 Diferencias entre versiones UML 1.x y UML 2.0

En las figuras de A.4 a la figura A.12 se muestran algunos ejemplos de los diagramas nuevos o actualizados en UML 2.0 obtenidos del tutorial: “El Poder Semántico del UML 2.0 en la Práctica” realizado por Lic. Valerio Adrián (http://www.epidataconsulting.com)

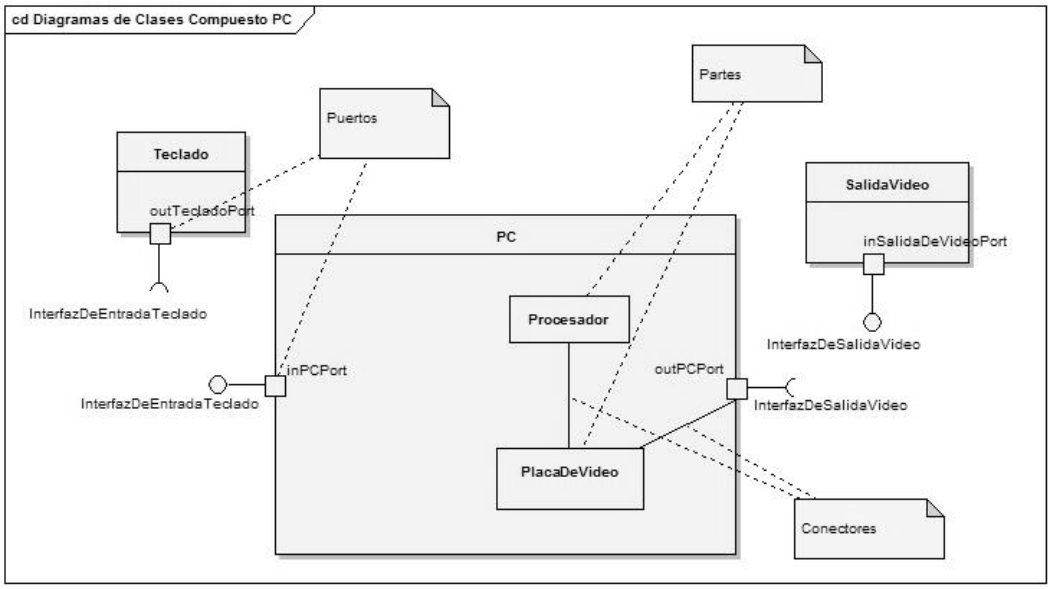


Figura A.4 Diagrama de clases

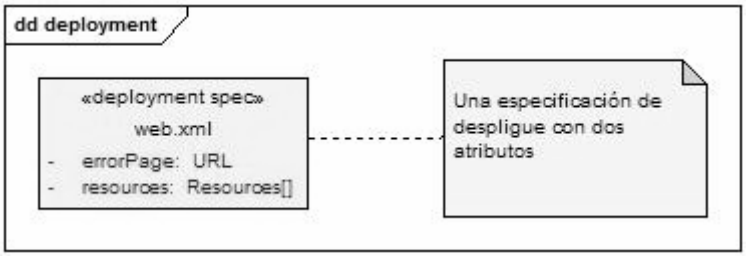


Figura A.5 Diagrama de despliegue

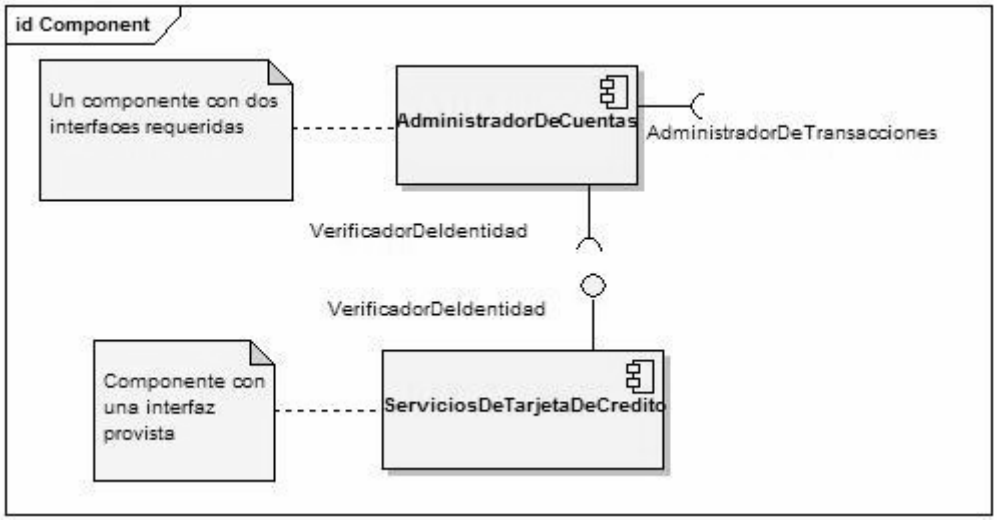


Figura A.6 Diagrama de componente

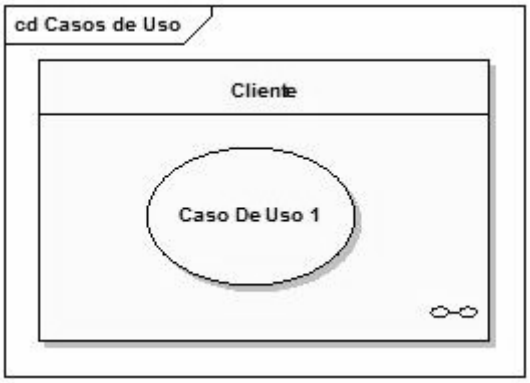


Figura A.7 Diagrama de casos de uso con estructuras compuestas

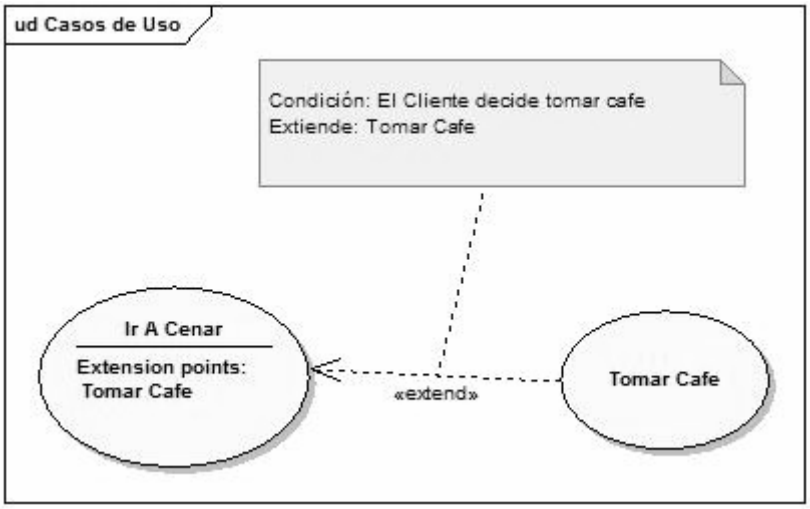


Figura A.8 Diagrama de casos de uso con condiciones de extensión

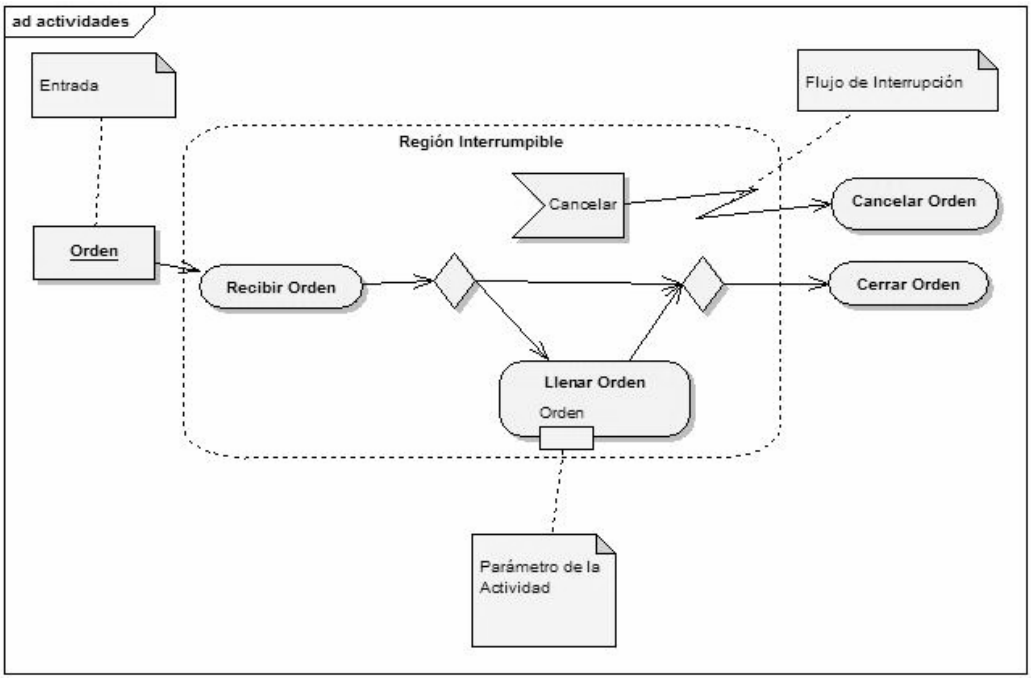


Figura A.9 Diagrama de actividades

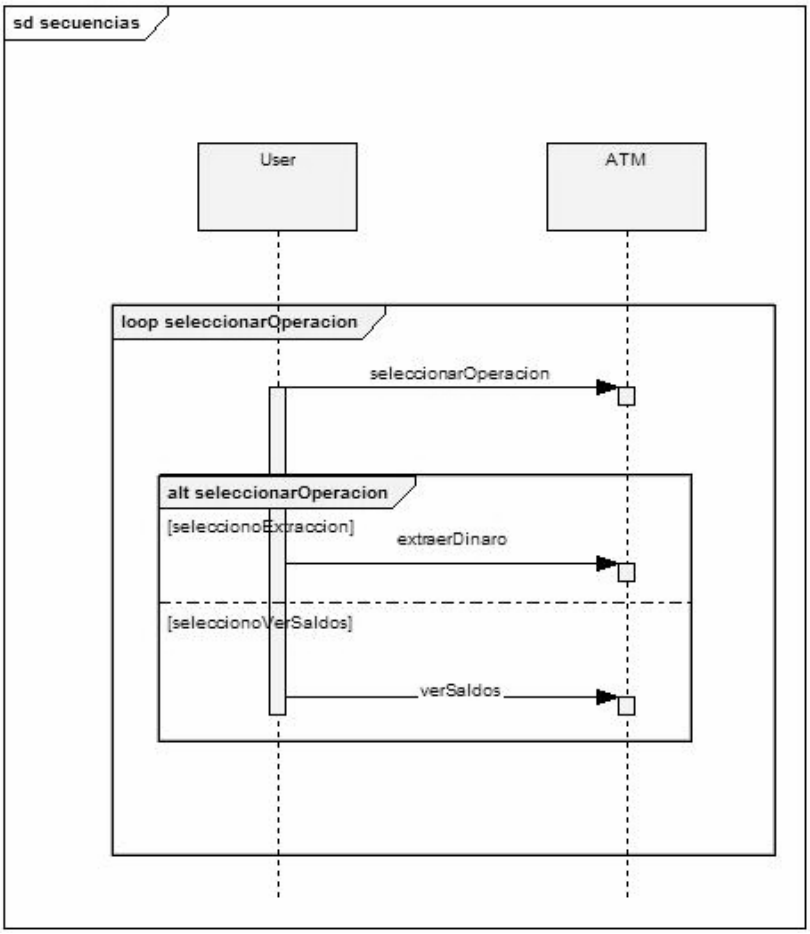


Figura A.10 Diagrama de interacción con fragmentos combinados

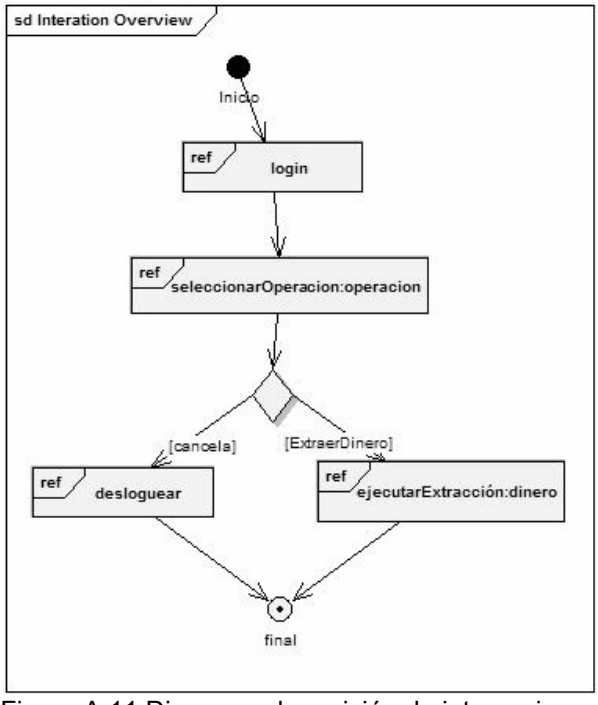


Figura A.11 Diagrama de revisión de interacciones

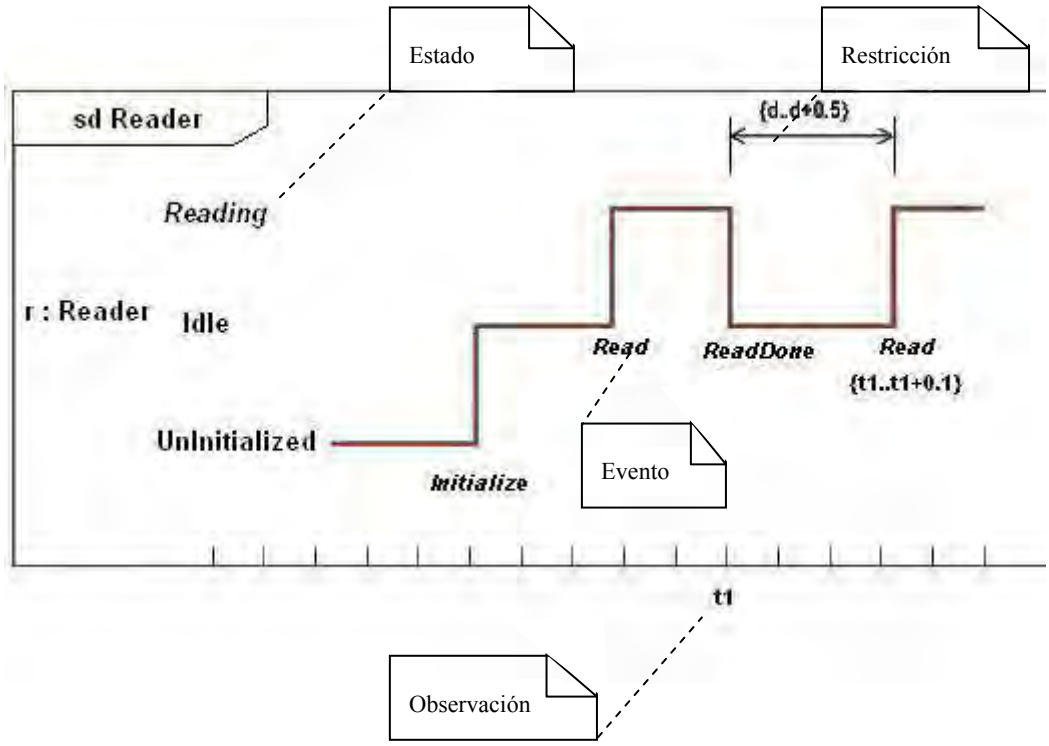


Figura A.12 Diagrama de tiempos

Bibliografía

Jacobson Ivar, Booch Grady, Rumbaugh James
El Proceso Unificado de Desarrollo de Software
Ed. Addison Wesley 2004

McGovern, Adatia Rahim, Fain Yakovx
Java 2 Enterprise Edition 1.4 (J2EE 1.4) Bible
Ed. Wiley Publishing, Inc. 2003

Booch Grady, Rumbaugh James, Jacobson Ivar
The Unified Modeling Language User Guide
Ed. Addison Wesley 1998

Halmiton Kim, Miles Rusell
Learning UML 2.0
Ed. O'Reilly, 2006

Fowler Martin
UML Distilled: A Brief Guide to the Standard Object Modeling Language
Ed. Addison Wesley, 2003

Pilone Dan, Pitman Neil
UML 2.0 in a Nutshell
Ed. O'Reilly, 2005

URL's consultados

http://www.geocities.com/SiliconValley/Network/1582/uml-example.htm
http://atlas.kennesaw.edu/~dbraun/csis4650/A&D/UML_tutorial/
http://www.uml.org/
http://aolive.blogspot.com/2005/02/es240-interfcies-versus-classes.html
http://www.americasistemas.com.pe/noticiero_digital/omg.htm
http://www.epidataconsulting.com/tikiwiki/tiki-read_article.php?articleId=31
http://www.epidataconsulting.com/tikiwiki/tiki-read_article.php?articleId=15