LENGUAJE UNIFICADO MODELADO “UML”

TRABAJO ELABORADO POR:

KARLA TATIANA OLAYA TORRES (406706)

MILLER ANDRES GALINDO DUCUARA (412088)

ROLDAN ESTIVEN POLANCO (408129)

UNIVERSIDAD COOPERATIVA DE COLOMBIA

INGENIERIA DE SISTEMAS

NEIVA

2015

Universidad Cooperativa de Colombia

LENGUAJE UNIFICADO MODELADO “UML”

TRABAJO ELABORADO POR:

KARLA TATIANA OLAYA TORRES (406706)

MILLER ANDRES GALINDO DUCUARA (412088)

ROLDAN ESTIVEN POLANCO (408129)

CURSO: INGENIERIA DE SOFTWARE I

DOCENTE: IRLESA INDIRA SANCHEZ MEDINA

UNIVERSIDAD COOPERATIVA DE COLOMBIA

INGENIERIA DE SISTEMAS

NEIVA

2015

Universidad Cooperativa de Colombia

TABLA DE CONTENIDO

LISTA DE ILUSTRACIONES

1. LENGUAJE UNIFICADO DE MODELADO (UML)

1.1 Concepto

El lenguaje unificado de diagrama o notación (UML) sirve para especificar, visualizar y documentar esquemas de sistemas de software orientado a objetos. UML facilita la creación de un producto de alta calidad, especialmente durante fases de análisis y diseño del proyecto. UML también puede usarse para documentar sus diseños de software para ayudarle a usted y al resto de desarrolladores.

Tener una buena maqueta del software es la mejor forma de comunicarse con otros desarrolladores que participen en el proyecto, así como con sus clientes. Una buena maqueta de extremadamente importante para los proyectos de mediano o gran tamaño, pero también resulta útil para los más pequeños. Aunque trabaje en un pequeño proyecto personal, podrá beneficiarse de una buena maqueta porque esta le proporcionará una visión global que le ayudará en la creación de un mejor código.

1.2 Funcionalidad de UML

El lenguaje unificado de diagrama o notación (UML) sirve para especificar, visualizar y documentar esquemas de sistemas de software orientado a objetos. UML no es un método de desarrollo, lo que significa que no sirve para determinar qué hacer en primer lugar o cómo diseñar el sistema, sino que simplemente le ayuda a visualizar el diseño y a hacerlo más accesible para otros. UML está controlado por el grupo de administración de objetos (OMG) y es el estándar de descripción de esquemas de software.

UML está diseñado para su uso con software orientado a objetos, y tiene un uso limitado en otro tipo de cuestiones de programación.

UML se compone de muchos elementos de esquematización que representan las diferentes partes de un sistema de software. Los elementos UML se utilizan para crear diagramas, que representa alguna parte o punto de vista del sistema.

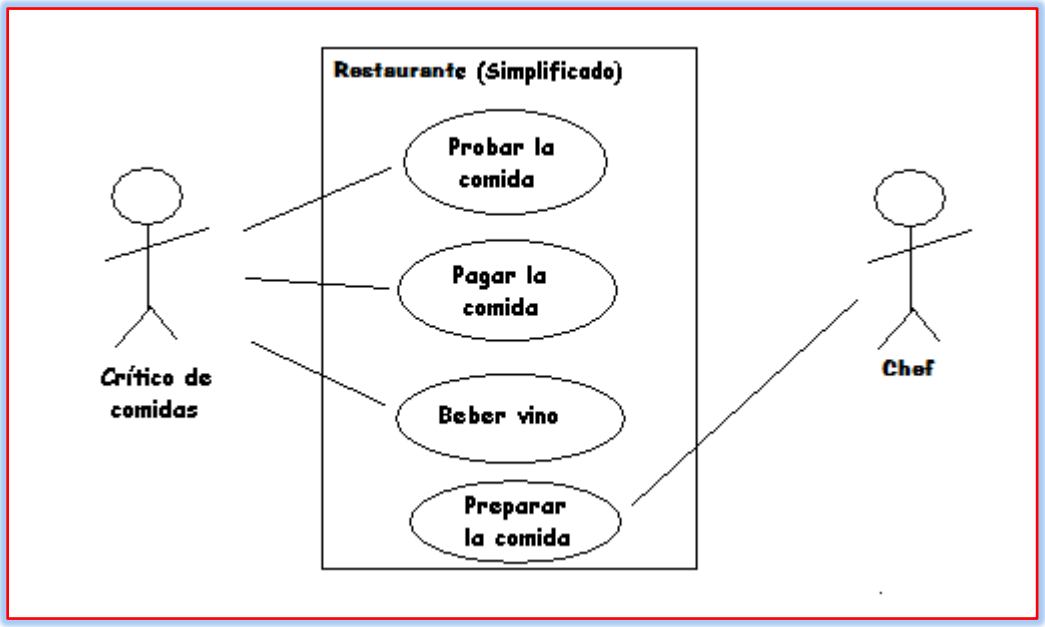
1.3 Tipos de Diagramas

1.3.1 Diagrama de Casos de Uso

Los diagramas de casos de uso describen las relaciones y las dependencias entre un grupo de *casos de uso* y los actores participantes en el proceso.

Es importante resaltar que los diagramas de casos de uso no están pensados para representar el diseño y no puede describir los elementos internos de un sistema. Los diagramas de casos de uso sirven para facilitar la comunicación con los futuros usuarios del sistema, y con el cliente, y resultan especialmente útiles para determinar las características necesarias que tendrá el sistema. En otras palabras, los diagramas de casos de uso describen *qué* es lo que debe hacer el sistema, pero no cómo.

Ilustración 1. Diagrama de Caso de Uso



1.3.2 Caso de uso

Un caso de uso describe, desde el punto de vista de los actores, un grupo de actividades de un sistema que produce un resultado concreto y tangible.

Los casos de uso son descriptores de las interacciones típicas entre los usuarios de un sistema y ese mismo sistema. Representan el interfaz externo del sistema y

especifican qué requisitos de funcionamiento debe tener este (recuerde, únicamente el qué, nunca el cómo).

Cuando se trabaja con casos de uso, es importante tener presentes algunas sencillas reglas:

- Cada caso de uso está relacionado como mínimo con un actor
- Cada caso de uso es un iniciador (es decir, un actor)
- Cada caso de uso lleva a un resultado relevante (un resultado con «valor intrínseco»)

Los casos de uso pueden tener relaciones con otros casos de uso. Los tres tipos de relaciones más comunes entre casos de uso son:

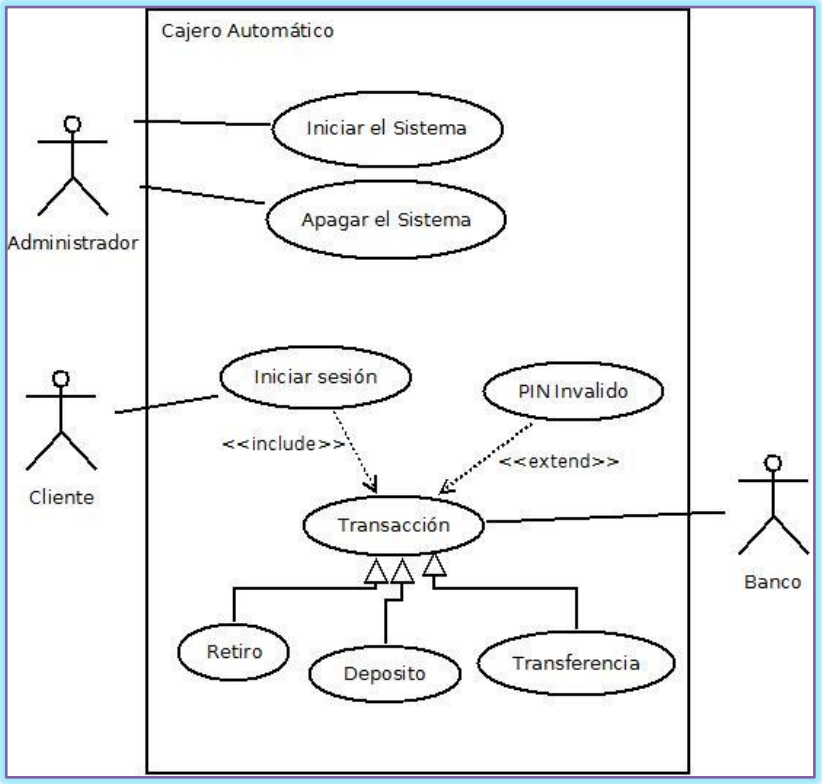
- **<<include>>** Especifica una situación en la que un caso de uso tiene lugar *dentro* de otro caso de uso.
- **<<extends>>** Especifica que en ciertas situaciones, o en algún punto (llamado punto de extensión) un caso de uso será extendido por otro.
- **Generalización** Especifica que un caso de uso hereda las características del «super» caso de uso, y puede volver a especificar algunas o todas ellas de una forma muy similar a las herencias entre clases.

1.3.3 Actor

Un actor es una entidad externa (de fuera del sistema) que interacciona con el sistema participando (y normalmente iniciando) en un caso de uso. Los actores pueden ser gente real (por ejemplo, usuarios del sistema), otros ordenadores o eventos externos.

Los actores no representan a personas *físicas* o a sistemas, sino su *rol*. Esto significa que cuando una persona interactúa con el sistema de diferentes maneras (asumiendo diferentes papeles), estará representado por varios actores.

Ilustración 2. Ejemplo de Actor



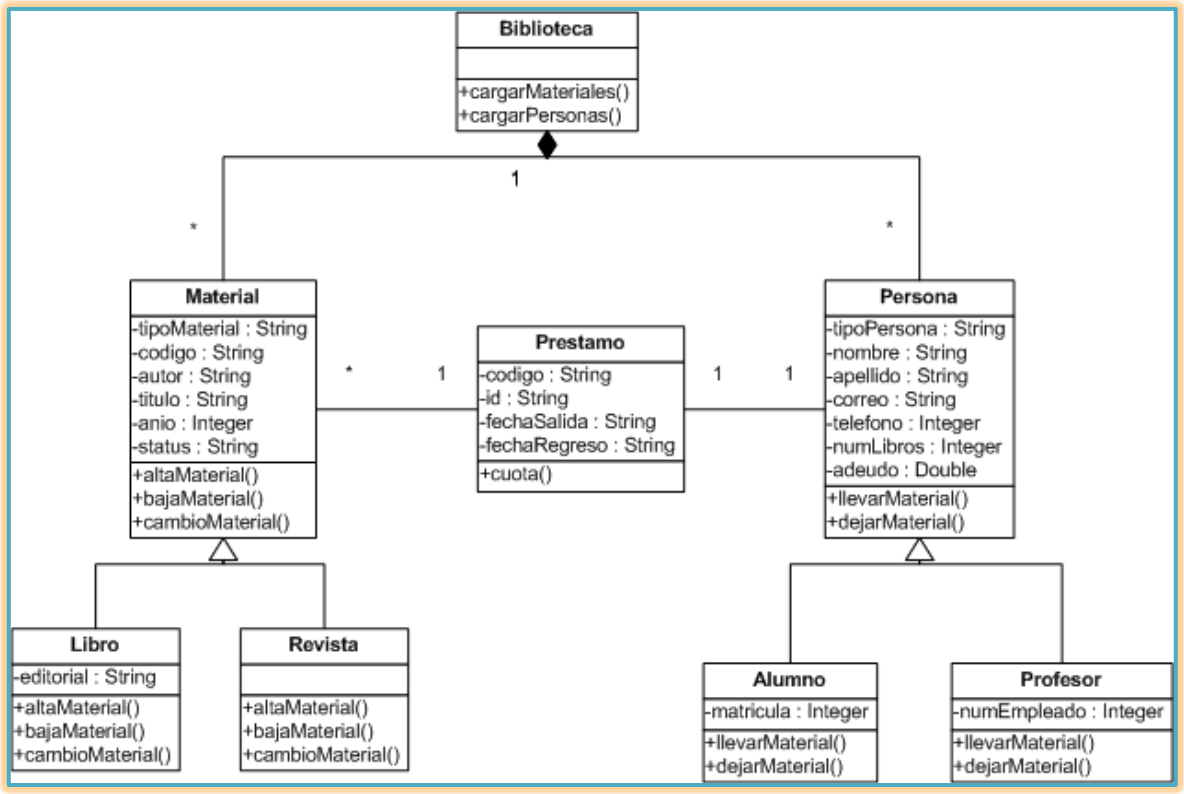
1.3.4 Descripción de Casos de Uso

Las descripciones de casos de uso son reseñas textuales del caso de uso. Normalmente tienen el formato de una nota o un documento relacionado de alguna manera con el caso de uso, y explica los procesos o actividades que tienen lugar en el caso de uso.

1.4 Diagrama de Clases

Los diagramas de clases muestran las diferentes clases que componen un sistema y cómo se relacionan unas con otras.

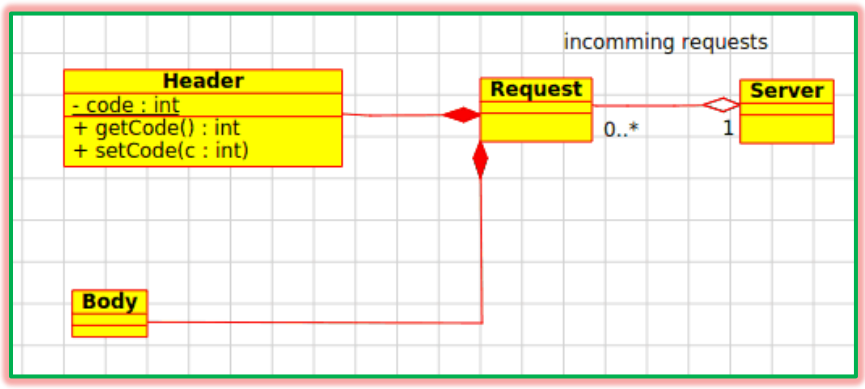
Ilustración 3. Ejemplo de Diagrama de Clases



Se dice que los diagramas de clases son diagramas «estáticos» porque muestran las clases, junto con sus métodos y atributos, así como las relaciones estáticas entre ellas: qué clases «conocen» a qué otras clases o qué clases «son parte» de otras clases, pero no muestran los métodos mediante los que se invocan entre ellas.

Fuente: (PRESSMAN)

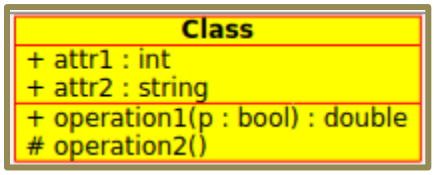
Ilustración 4. Diagrama de Clases



1.4.1 Clase

Una clase define los atributos y los métodos de una serie de objetos. Todos los objetos de esta clase (instancias de esa clase) tienen el mismo comportamiento y el mismo conjunto de atributos (cada objetos tiene el suyo propio). En ocasiones se utiliza el término «tipo» en lugar de clase, pero recuerde que no son lo mismo, y que el término tipo tiene un significado más general.

Ilustración 5. Representación visual de una clase



En ¨, las clases están representadas por rectángulos, con el nombre de la clase, y también pueden mostrar atributos y operaciones de la clase en otros dos «compartimentos» dentro del rectángulo.

1.4.2 Atributos

En UML, los atributos se muestran al menos con su nombre, y también pueden mostrar su tipo, valor inicial y otras propiedades. Los atributos también pueden ser mostrados visualmente:

- + Indica atributos *públicos*
- # Indica atributos *protegidos*
- - Indica atributos *privados*

1.4.3 Operaciones

Las operaciones (métodos) también se muestran al menos con su nombre, y pueden mostrar sus parámetros y valores de retorno. Las operaciones, al igual que los atributos, se pueden mostrar visualmente:

- + Indica operaciones *públicas*
- # Indica operaciones *protegidas*
- - Indica operaciones *privadas*

1.4.4 Plantillas

Las clases pueden tener plantillas, un valor usado para una clase no especificada o un tipo. El tipo de plantilla se especifica cuando se inicia una clase (es decir cuando se crea un objeto). Las plantillas existen en C++ y se introducirán en Java 1.5 con el nombre de Genéricos.

1.5 Asociaciones de Clases

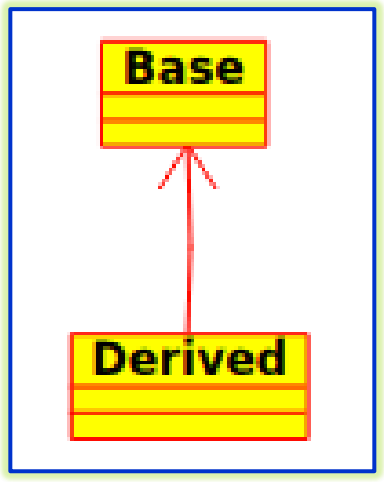
Las clases se puede relacionar (estar asociadas) con otras de diferentes maneras:

1.5.1 Generalización

La herencia es uno de los conceptos fundamentales de la programación orientada a objetos, en la que una clase «recoge» todos los atributos y operaciones de la clase de la que es heredera, y puede alterar/modificar algunos de ellos, así como añadir más atributos y operaciones propias.

En UML, una asociación de *generalización* entre dos clases, coloca a estas en una jerarquía que representa el concepto de herencia de una clase derivada de la clase base. En UML, las generalizaciones se representan por medio de una línea que conecta las dos clases, con una flecha en el lado de la clase base.

Ilustración 6. Representación visual de una generalización en UML



1.5.2 Asociaciones

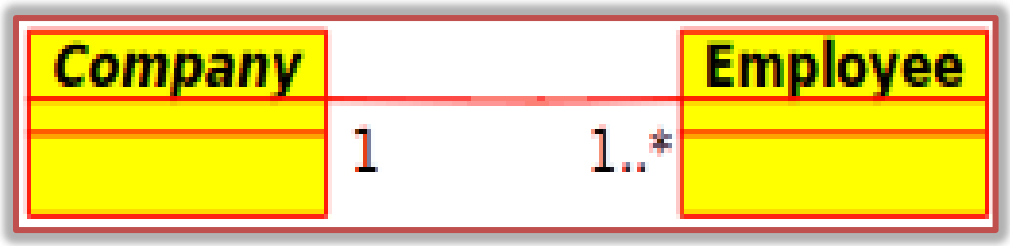
Una asociación representa una relación entre clases, y aporta la semántica común y la estructura de muchos tipos de «conexiones» entre objetos.

Las asociaciones son los mecanismos que permite a los objetos comunicarse entre sí. Describe la conexión entre diferentes clases (la conexión entre los objetos reales se denomina conexión de objetos o *enlace*).

Las asociaciones pueden tener un papel que especifica el propósito de la asociación y pueden ser unidireccionales o bidireccionales (indicando si los dos objetos participantes en la relación pueden intercambiar mensajes entre sí, o es únicamente uno de ellos el que recibe información del otro). Cada extremo de la asociación también tiene un valor de multiplicidad, que indica cuántos objetos de ese lado de la asociación están relacionados con un objeto del extremo contrario.

En UML, las asociaciones se representan por medio de líneas que conectan las clases participantes en la relación, y también pueden mostrar el papel y la multiplicidad de cada uno de los participantes. La multiplicidad se muestra como un rango [mín...máx] de valores no negativos, con un asterisco (*) representando el infinito en el lado máximo.

Ilustración 7. Representación visual de una asociación en UML



1.5.3 Acumulación

Las acumulaciones son tipos especiales de asociaciones en las que las dos clases participantes no tienen un estado igual, pero constituyen una relación «completa». Una acumulación describe cómo se compone la clase que asume el rol completo de otras clases que se encargan de las partes. En las acumulaciones, la clase que actúa como completa, tiene una multiplicidad de uno.

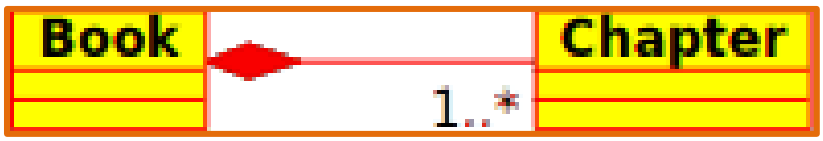
Ilustración 8. Representación visual de una relación de acumulación en UML

En UML, las acumulaciones están representadas por una asociación que muestra un rombo en uno de los lados de la clase completa.

1.5.4 Composición

Las composiciones son asociaciones que representan acumulaciones *muy fuertes*. Esto significa que las composiciones también forman relaciones completas, pero dichas relaciones son tan fuertes que las partes no pueden existir por sí mismas. Únicamente existen como parte del conjunto, y si este es destruido las partes también lo son.

Ilustración 9. Representación visual de una relación de composición en UML



En UML, las composiciones están representadas por un rombo sólido al lado del conjunto.

1.6 Otros componentes de los diagramas de clases

Los diagramas de clases pueden contener más componentes aparte de clases.

1.6.1 Interfaces

Las interfaces son clases abstractas, lo que significa que no es posible crear instancias directamente a partir de ellas. Pueden contener operaciones, pero no atributos. Las clases pueden heredar de las interfaces (a través de una asociación de realización) y de estos diagramas sí es posible crear instancias.

1.6.2 Tipo de datos

Los tipos de datos son primitivas construidas normalmente en algunos lenguajes de programación. Algunos ejemplos comunes son los enteros y los booleanos. No pueden tener relación con clases, pero las clases sí pueden relacionarse con ellos.

1.6.3 Enumeraciones

Las enumeraciones son simples listas de valores. Un ejemplo típico de esto sería una enumeración de los días de la semana. Las opciones de una enumeración se llaman «literales de enumeración». Al igual que los tipos de datos, no pueden relacionarse con las clases, pero las clases sí pueden hacerlo con ellos.

1.6.4 Paquetes

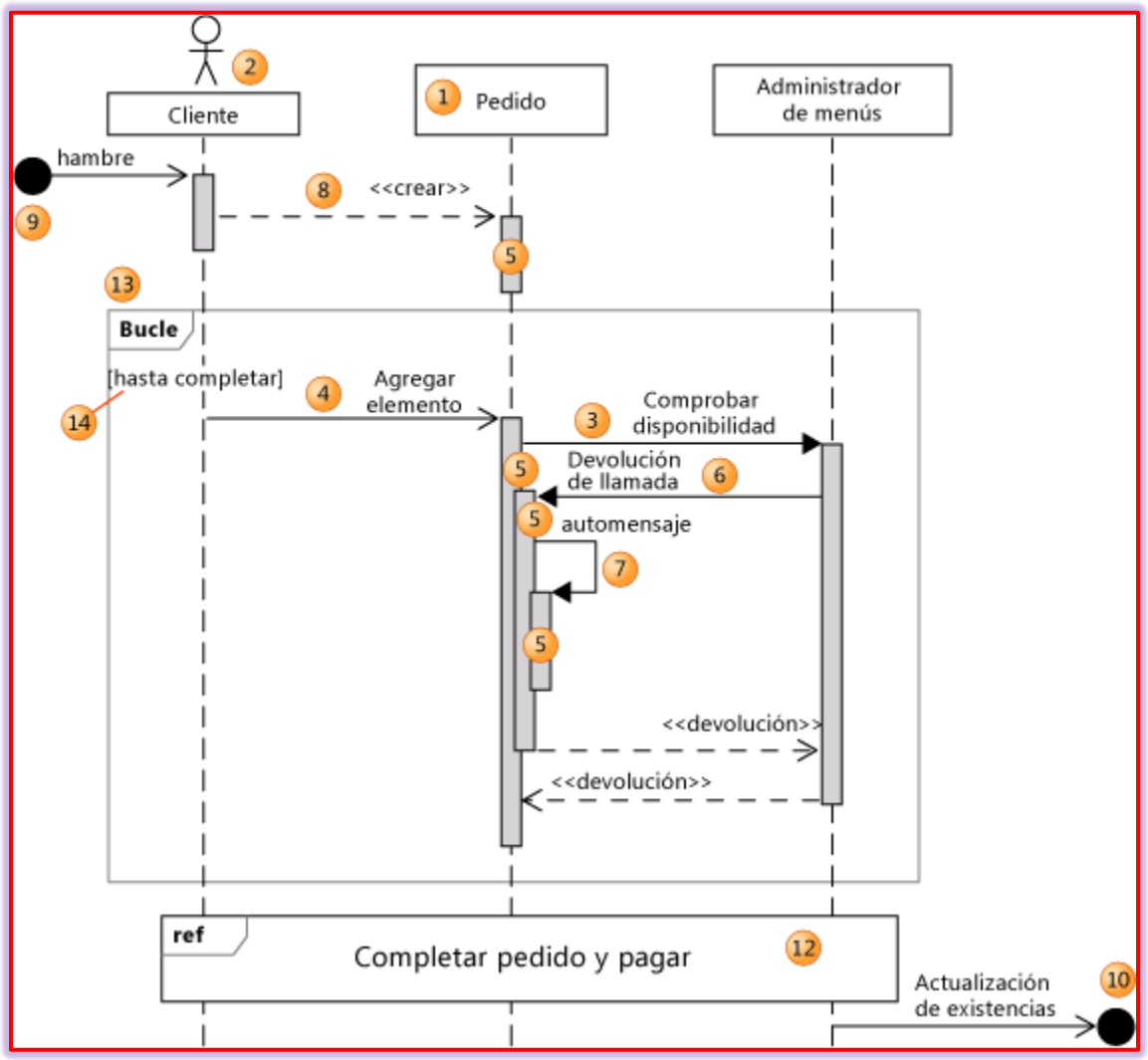
Los paquetes, en lenguajes de programación, representan un espacio de nombres en un diagrama se emplean para representar partes del sistema que contienen más de una clase, incluso cientos de ellas.

1.7 Diagramas de Secuencia

Los diagramas de secuencia muestran el intercambio de mensajes (es decir la forma en que se invocan) en un momento dado. Los diagramas de secuencia ponen especial énfasis en el orden y el momento en que se envían los mensajes a los objetos.

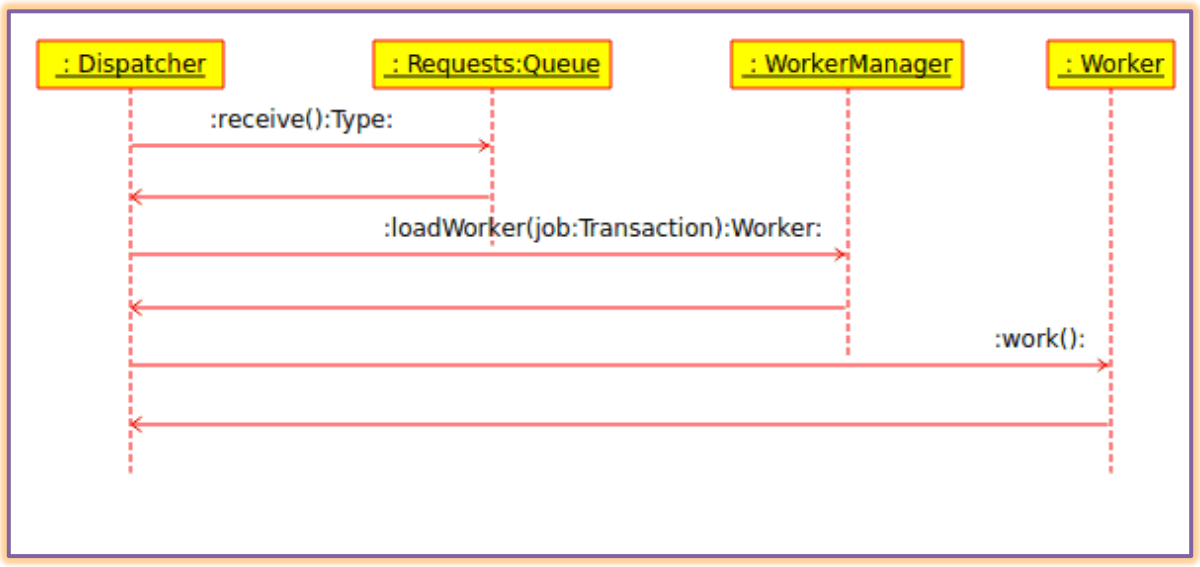
En los diagramas de secuencia, los objetos están representados por líneas intermitentes verticales, con el nombre del objeto en la parte más alta. El eje de tiempo también es vertical, incrementándose hacia abajo, de forma que los mensajes son enviados de un objeto a otro en forma de flechas con los nombres de la operación y los parámetros.

Ilustración 10. Diagrama de secuencia de Pedido



Los mensajes pueden ser o bien síncronos, el tipo normal de llamada del mensaje donde se pasa el control a objeto llamado hasta que el método finalice, o asíncronos donde se devuelve el control directamente al objeto que realiza la llamada. Los mensajes síncronos tienen una caja vertical en un lateral del objeto invocante que muestra el flujo del control del programa.

Ilustración 11. Umbrello UML Modeller mostrando un diagrama de secuencia.

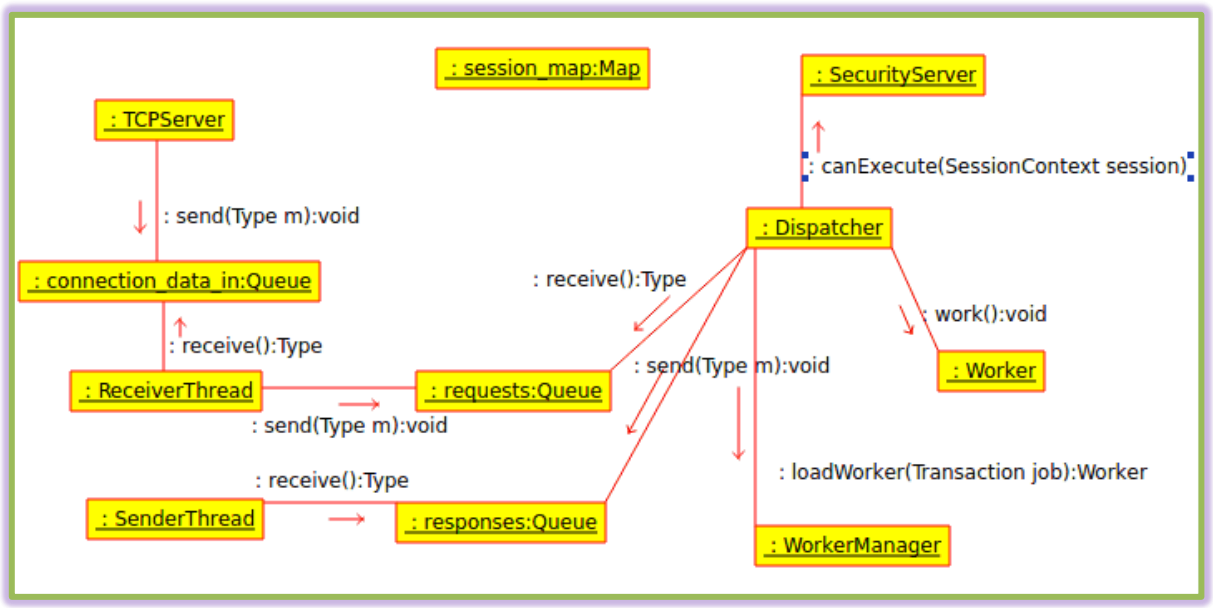


1.8 Diagramas de Colaboración

Los diagramas de colaboración muestran las interacciones que ocurren entre los objetos que participan en una situación determinada. Esta es más o menos la misma información que la mostrada por los diagramas de secuencia, pero destacando la forma en que las operaciones se producen en el tiempo, mientras que los diagramas de colaboración fijan el interés en las relaciones entre los objetos y su topología.

En los diagramas de colaboración los mensajes enviados de un objeto a otro se representan mediante flechas, mostrando el nombre del mensaje, los parámetros y la secuencia del mensaje. Los diagramas de colaboración están indicados para mostrar una situación o flujo programa específicos y son unos de los mejores tipos de diagramas para demostrar o explicar rápidamente un proceso dentro de la lógica del programa.

Ilustración 12. Umbrello UML Modeller mostrando un diagrama de colaboración



1.9 Diagrama de Estado

Los diagramas de estado muestran los diferentes estados de un objeto durante su vida, y los estímulos que provocan los cambios de estado en un objeto.

Los diagramas de estado ven a los objetos como máquinas de estado o autómatas finitos que pueden estar en un conjunto de estados finitos y que pueden cambiar su estado a través de un estímulo perteneciente a un conjunto finito. Por ejemplo, un objeto de tipo *NetServer* puede tener durante su vida uno de los siguientes estados:

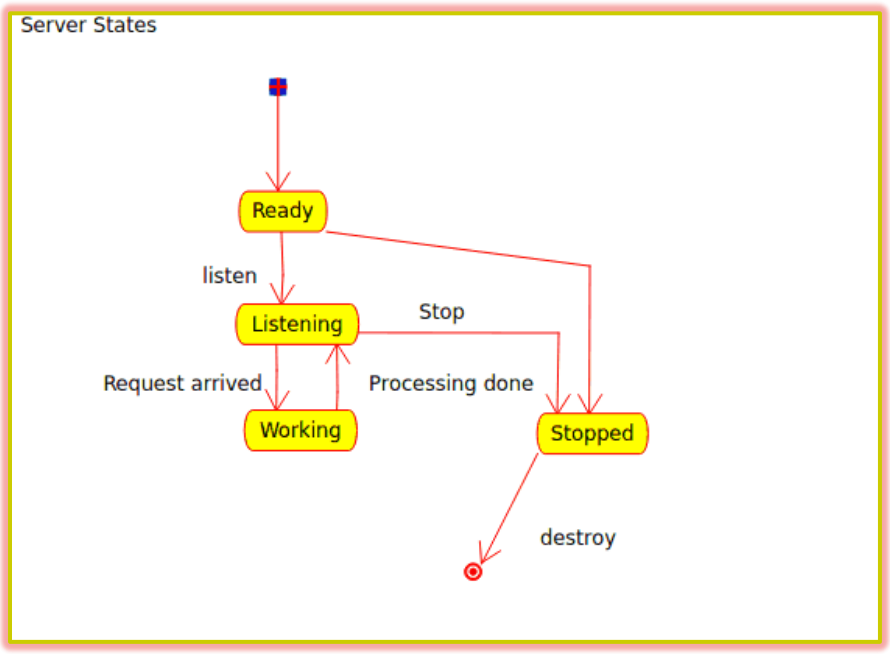
- Listo
- Escuchando
- Trabajando
- Detenido

Los eventos que pueden producir que el objeto cambie de estado son

- Se crea el objeto
- El objeto recibe un mensaje de escucha
- Un cliente solicita una conexión a través de la red

- Un cliente finaliza una solicitud
- La solicitud se ejecuta y ser termina
- El objeto recibe un mensaje de detención

Ilustración 13. Umbrello UML Modeller mostrando un diagrama de estado



1.9.1 Estado

Los estados son los ladrillos de los diagramas de estado. Un estado pertenece a exactamente una clase y representa un resumen de los valores y atributos que puede tener la clase. Un estado UML describe el estado interno de un objeto de una clase particular

Tenga en cuenta que no todos los cambios en los atributos de un objeto deben estar representados por estados, sino únicamente aquellos cambios que pueden afectar significativamente a la forma de funcionamiento del objeto

Hay dos tipos especiales de estados: inicio y fin. Son especiales en el sentido de que no hay ningún evento que pueda devolver a un objeto a su estado de inicio, y de la misma forma no hay ningún evento que pueda sacar a un objeto de su estado de fin.

1.10 Diagrama de Actividad

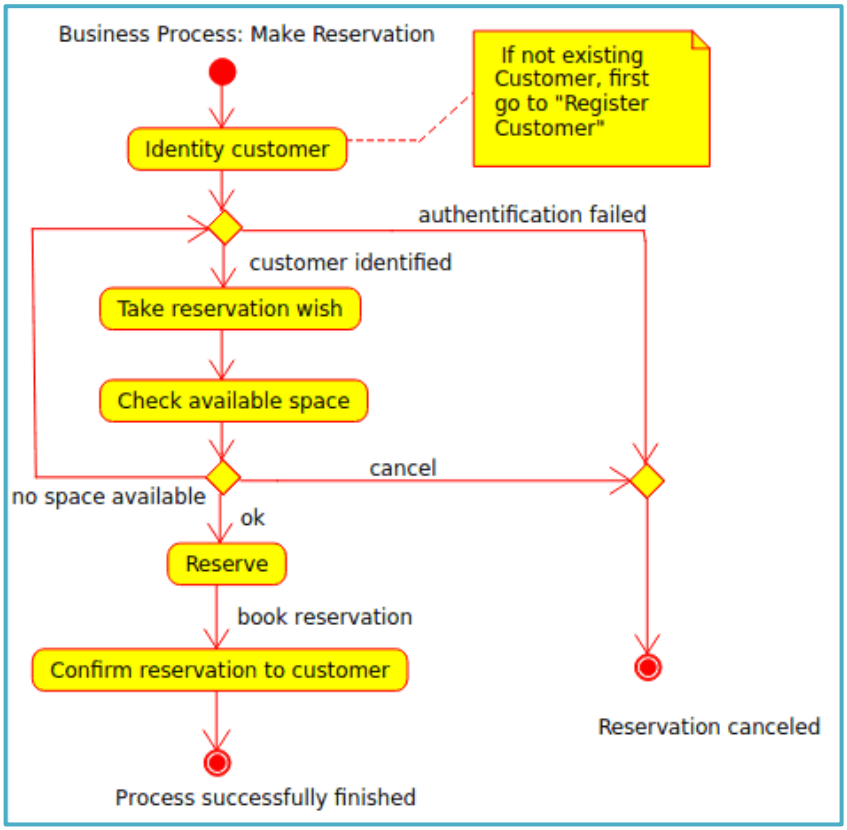
Los diagramas de actividad describen la secuencia de las actividades en un sistema. Los diagramas de actividad son una forma especial de los diagramas de estado, que únicamente (o mayormente) contienen actividades.

Los diagramas de actividad son similares a los diagramas de flujo procesales, con la diferencia de que todas las actividades están claramente unidas a objetos.

Los diagramas de actividad siempre están asociados a una *clase*, a una *operación* o a un *caso de uso*.

Los diagramas de actividad soportan actividades tanto secuenciales como paralelas. La ejecución paralela se representa por medio de iconos de fork/espera, y en el caso de las actividades paralelas, no importa en qué orden sean invocadas (pueden ser ejecutadas simultáneamente o una detrás de otra).

Ilustración 14. Umbrello UML Modeller mostrando un diagrama de actividad



1.10.1 Actividad

Una actividad es un único paso de un proceso. Una activa es un estado del sistema que actividad interna y, al menos, una transición saliente. Las actividades también pueden tener más de una transición saliente, si tienen diferentes condiciones.

Las actividades pueden formar jerarquías, lo que significa que una actividad puede estar formada de varias actividades «de detalle», en cuyo caso las transiciones entrantes y salientes deberían coincidir con las del diagrama de detalle.

1.11 Elementos de Ayuda

Existen unos pocos elementos en UML que no tiene un valor semántico real en la maqueta, pero que ayudan a clarificar partes del programa. Estos elementos son

- Línea de texto
- Notas de texto y enlaces
- Cajas

Las líneas de texto son útiles para añadir información textual a un diagrama. Es texto es libre y no tiene ningún significado para la maqueta.

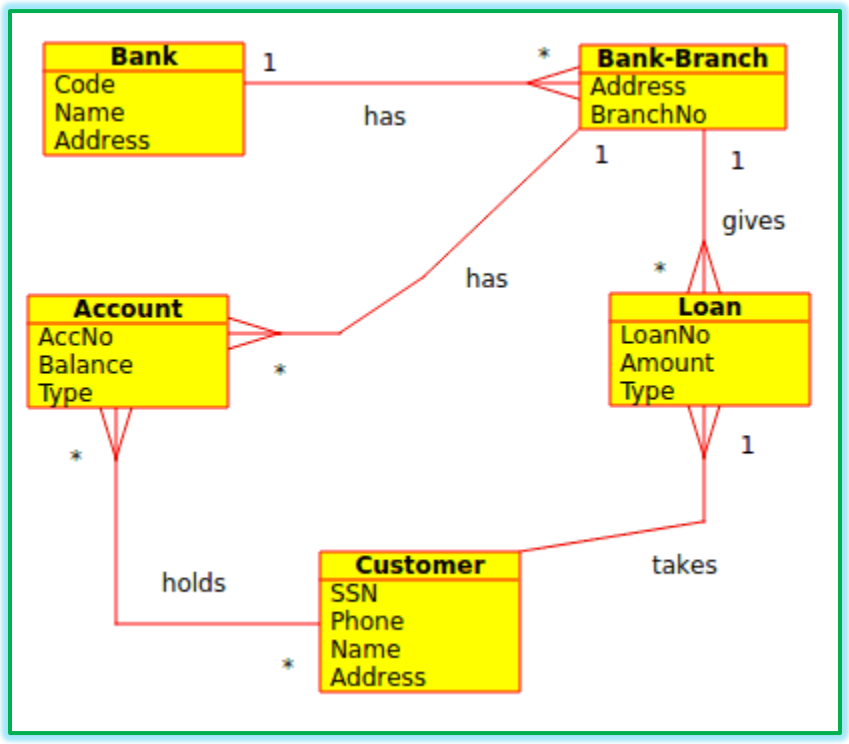
Las notas son útiles para añadir información más detallada de un objeto o una situación específica. Tienen la gran ventaja de que se pueden anclar a los elementos UML para mostrar que una nota «pertenece» a un objeto o situación específicos.

Las cajas son rectángulos repartidos libremente que pueden usarse para juntar objetos haciendo los diagramas más legibles. No tienen significado lógico en la maqueta.

1.12 Diagramas de relación de entidad

Los diagramas de relaciones de entidad (diagramas ER) muestran el diseño conceptual de las aplicaciones de bases de datos. Representan varias entidades (conceptos) en el sistema de información y las relaciones y restricciones existentes entre ellas. Una extensión de los diagramas de relaciones de entidad llamado «diagramas de relaciones de entidad extendida» o «diagramas de relaciones de entidad mejoradas» (EER), se utiliza para incorporar las técnicas de diseño orientadas a objetos en los diagramas ER.

Ilustración 15. Umbrello mostrando un diagrama de relaciones de entidad



1.12.1 Entidad

Una *Entidad* es cualquier concepto del mundo real con una existencia independiente. Puede ser un objeto con una existencia física (ejemplo, máquina, robot) o puede ser un objeto con una existencia conceptual (p. ej.: Curso de universidad). Cada entidad tiene un conjunto de atributos que describen las propiedades de la entidad.

Nota: No existen notaciones estándar para representar los diagramas ER. Los diferentes textos sobre este asunto utilizan diferentes notaciones. Los conceptos y notaciones para los diagramas EER utilizados en Umbrello provienen del siguiente libro: *Elmasri R. y Navathe S. (2004). Fundamentals of Database Systems 4ª ed. Addison Wesley* (Fundamentos de los sistemas de bases de datos)

En un diagrama ER, las entidades se representan como rectángulos, con el nombre de la clase, y también pueden mostrar atributos y operaciones de la clase en otros dos «compartimentos» dentro del rectángulo.

1.12.2 Atributos de la Entidad

En los diagramas ER, los atributos de la entidad se muestra con su nombre en un compartimento diferente de la entidad a la que pertenecen.

1.12.3 Restricciones

Las restricciones en los diagramas ER especifican las restricciones de los datos en el esquema de información.

Existen cuatro tipos de restricciones soportadas por Umbrello:

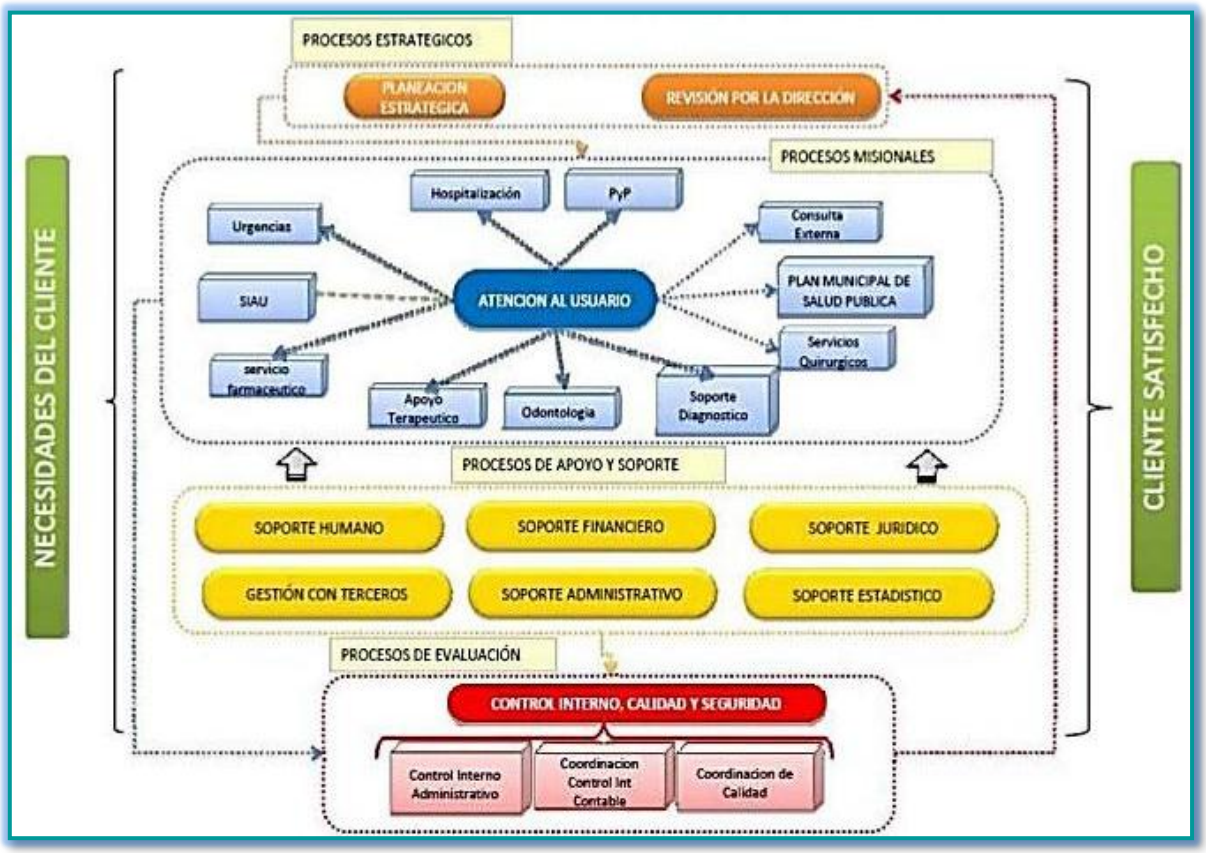
- *Clave primaria:* El conjunto de atributos declarados como *clave primaria* es única para la entidad. Solo puede haber una clave primaria en una entidad y ninguno de los atributos que la componen puede ser NULL.
- *Clave única:* El conjunto de atributos declarados como *única* son únicos para la entidad. Pueden haber muchas restricciones únicas en una entidad. Los atributos que lo componen pueden tener el valor NULL. Las claves únicas y primarias identifican de forma única una fila de una tabla (entidad)
- *Clave externa:* Una clave externa es una restricción referencia entre dos tablas. La clave externa identifica una columna o un conjunto de columnas en una tabla (referenciada) que referencia una columna o conjunto de columnas en otra tabla (referenciada). Las columnas en la tabla referenciada deben formar una clave primaria o una clave única.
- *Restricción de comprobación:* Una restricción de comprobación (también conocida como restricción de comprobación de tabla) es una condición que define los datos válidos cuando se añaden o actualizan datos en una tabla de la base de datos relacional. Se aplicará una restricción a cada fila de la tabla. La restricción debe ser un predicado. Puede referirse a una o varias columnas de la tabla.

Ejemplo: precio >= 0

Fuente: (Lenguaje Unificado Modelado UML)

2. EMPRESA DE SALUD

Ilustración 11. Mapa de Proceso (Medicina EPS



2.1 Problema, Oportunidades y Objetivos

Problema

La empresa “Medicina EPS” carece de un sistema de gestión integral en cuanto a la organización y funcionamiento de procesos en la prestación de servicios a los usuarios, afectando así el buen desempeño de la gestión en cuanto a la información manejada.

Oportunidades

Para toda entidad, la salud del capital humano debe convertirse en un tema estratégico para mejorar su competitividad. En el cual las oportunidades que se obtienen implementando un sistema de información son las siguientes:

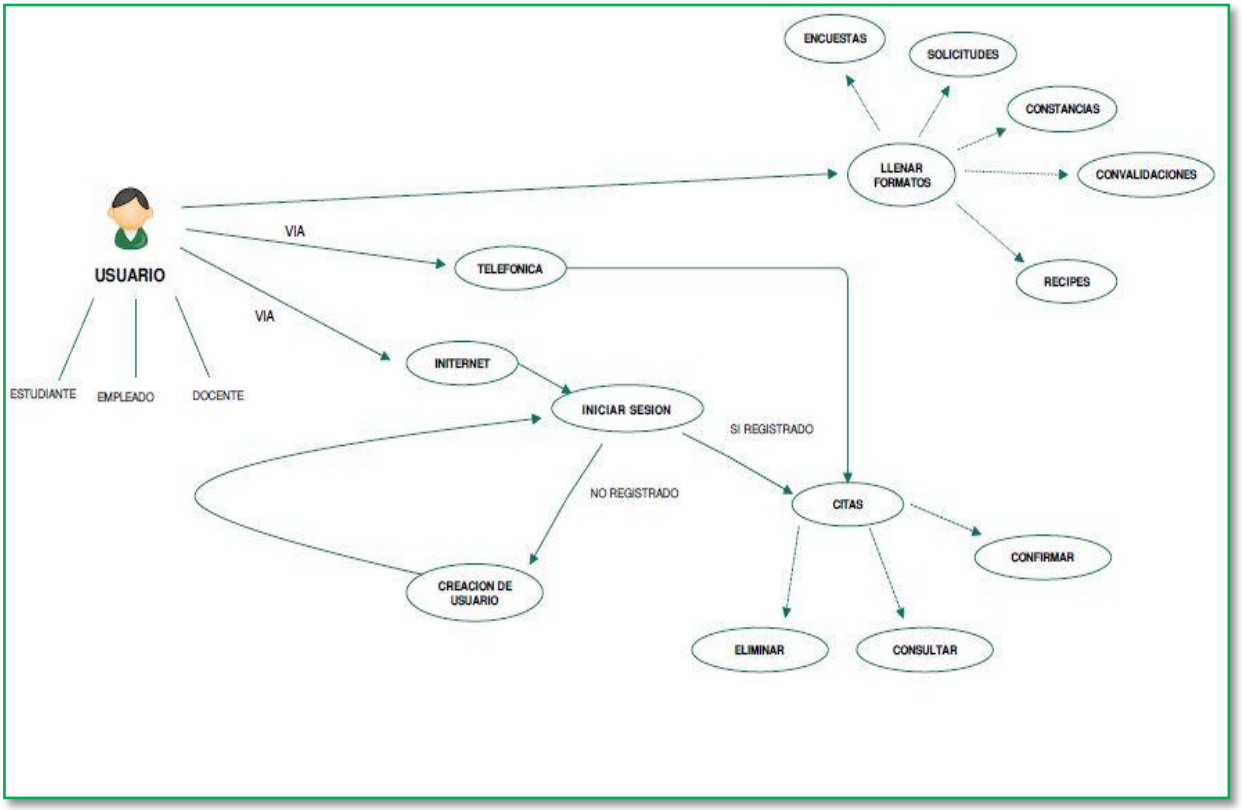
- Acceso rápido a la información y por ende mejora en la atención a los usuarios.
- Generación de informes e indicadores, que permiten corregir fallas difíciles de detectar y controlar con un sistema manual.
- Evitar pérdida de tiempo recopilando información que ya está almacenada en bases de datos que se pueden compartir.
- Impulso a la creación de grupos de trabajo e investigación debido a la facilidad para encontrar y manipular la información.
- Organización en el manejo de archivos e información clasificada por temas de interés general y particular, ya sean (citas médicas, historias clínicas, etc.)
- Aumento de la productividad gracias a la liberación de tiempos en búsqueda y generación de información repetida.

Todos estos se encuentran entre los programas integrales de salud empresarial; desde la campaña de comunicación interna, hasta cuestionarios de salud, valoraciones médicas y actividades de seguimiento. De esta manera, mejoran la calidad de vida de la las personas e incrementan su compromiso, optimizando y automatizando los procesos operativos internos, de los flujos de información y de los reportes.

Objetivos

- Diseñar y desarrollar un sistema de Información capaz de tramitar, asignar y programar el servicio de citas médicas, de Medicina General, Odontología y con especialistas a los usuarios de la Entidad Privada Prestadora de servicios de salud “Medicina EPS”
- Realizar el análisis de requerimientos para establecer los alcances y limitaciones del Sistema de Información.
- Seleccionar y generar un modelo de Ingeniería de Software que sea acorde a los estándares de la arquitectura de un Sistema de Información.
- Proyectar el diseño de la arquitectura de un Sistema de Información, acorde con las necesidades y exigencias requeridas para tal efecto.
- Realizar una fase de implementación y pruebas con su respectiva documentación, para validar y verificar el correcto funcionamiento de aquel Sistema de Información.

Ilustración 12. Diagrama de Caso de Uso (Medicina EPS)



BIBLIOGRAFIA

Lenguaje Unificado Modelado UML. (s.f.). Obtenido de https://docs.kde.org/trunk4/es/kdesdk/umbrello/uml-basics.html

PRESSMAN, R. S. (s.f.). *Ingeniería de software un enfoque práctico.*