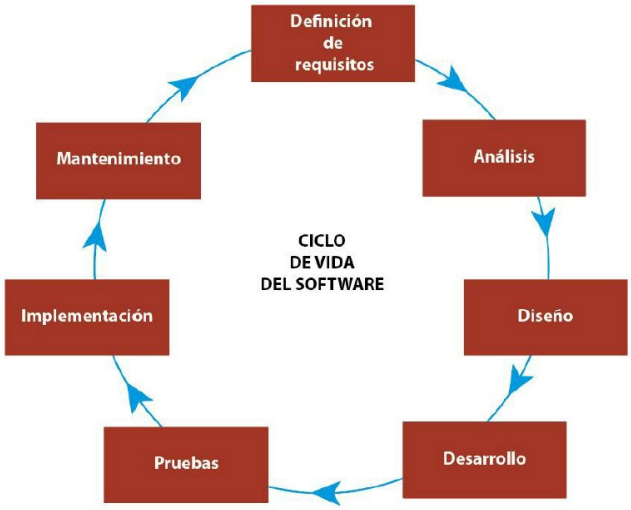
U7. Introducción del lenguaje unificado de modelado (UML)

1. INTRODUCCIÓN.

Atendiendo al ciclo de vida de un software habíamos visto que la **tarea de diseño** es una formalización adicional de la fase de análisis en un proyecto de software. Donde se tomarán cuenta todas las consecuencias del ámbito de la implementación de nuestro software y se definirán las especificaciones detalladas de cada uno de los elementos que forman la aplicación, incluyendo sus operaciones, atributos, el comportamiento que tienen con otros módulos, etc.



Definición de requisitos	Necesidades. Objetivos. Estudio de la viabilidad. Alcance del sistema.
Análisis	Definición de los requisitos. Análisis de los requisitos. Recursos. Costes.
Diseño	Clasificación de los requisitos. Organización de la arquitectura del software. Representación de las partes del sistema y su interrelación.
Desarrollo	Codificado y depuración del código. Generación de documentación.
Pruebas	Verificación del sistema.
Implementación	Instalación del sistema en producción.
Mantenimiento	Mejoras. Corrección de errores. Ampliación de funcionalidades.

Durante el **diseño** podemos analizar los resultados de la futura implementación y comprobar si los resultados del análisis son apropiados. Esta es **una parte fundamental del ciclo de vida del software** pues es la fase más temprana de creación y donde se pueden encontrar errores estructurales, que si no se corrigen pueden provocar que la implementación no sea viable en las fases posteriores.

El diseño debe cumplir una serie de principios generales:

- Modularidad.
- Reusabilidad.
- Portabilidad.
- Flexibilidad.
- Extensibilidad.

Se debe tener en cuenta:

- **Diseño de datos.** Define la estructura de los datos y sus relaciones (Ej. Base de datos y la relación entre los datos).

- **Diseño arquitectónico.** Relación entre elementos estructurales del programa, clases, estructuras de control, funciones. (Ej. Definir patrón MVC)
- **Diseño de la interfaz.** Comunicación (elementos y acciones de los componentes de la interfaz) entre el usuario y la aplicación. (Ej. Wireframe GUI).
- **Diseño a nivel de componentes.** Se obtiene una descripción procedimental de los componentes de nuestro diseño.

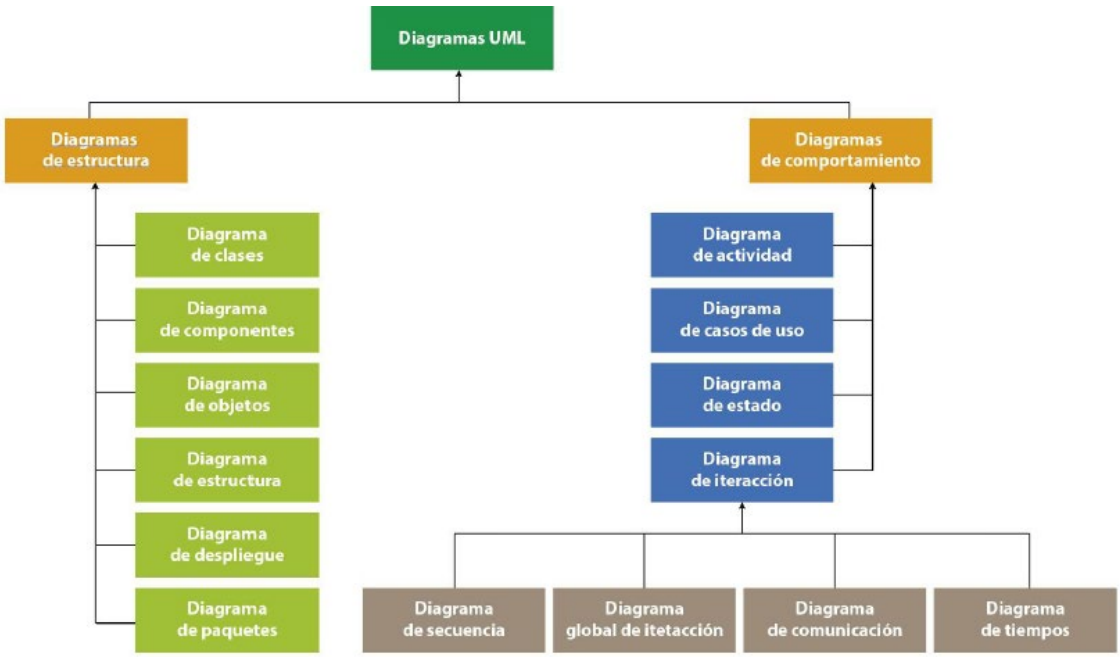
2. UML. ¿Qué es? Tipos.

NOTA: UML no es un lenguaje de programación ni es un proceso de desarrollo de software. UML nos proporciona una semántica para que el desarrollador pueda describir un modelo de software y así otro desarrollador pueda interpretar ese modelo sin ambigüedad.

"UML sigue la idea de que, mediante gráficos y dibujos, las cosas se entienden mejor, de tal modo que dispone de todos los elementos necesarios para realizar una descripción de un sistema."

Con UML podemos hacer una representación de todo nuestro desarrollo de software. El comportamiento de los métodos, los patrones utilizados, el uso que se dará de cada módulo o incluso los atributos y métodos de una clase además de sus relaciones.

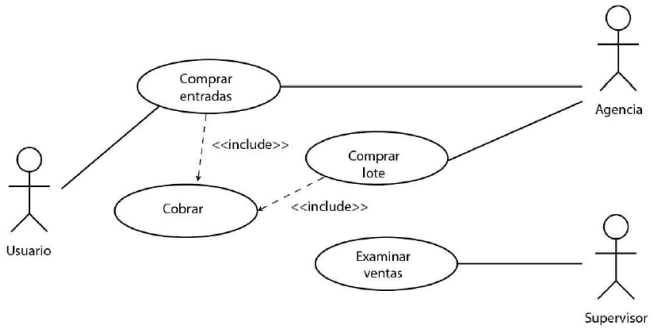
Esta representación UML proporciona una serie de diagramas estandarizados:



3. DIAGRAMAS UML

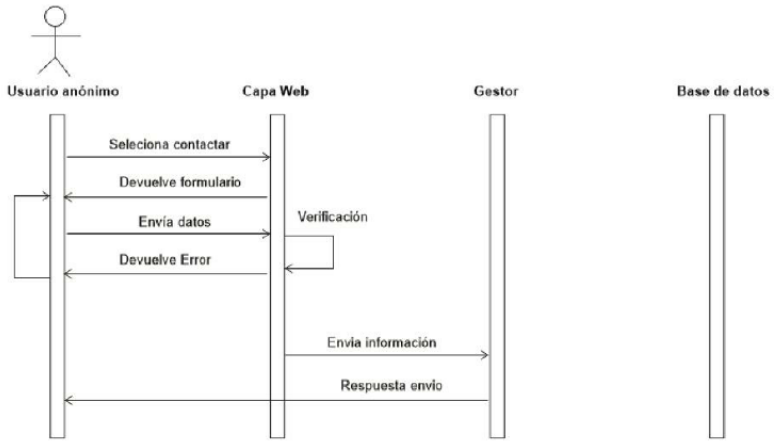
3.1. DIAGRAMA DE CASOS DE USO.

Con este diagrama representamos la funcionabilidad que tendrá el sistema para cada uno de los actores que controlen alguna de las funciones. Estos actores pueden ser personas u otros sistemas que interactúen con él.



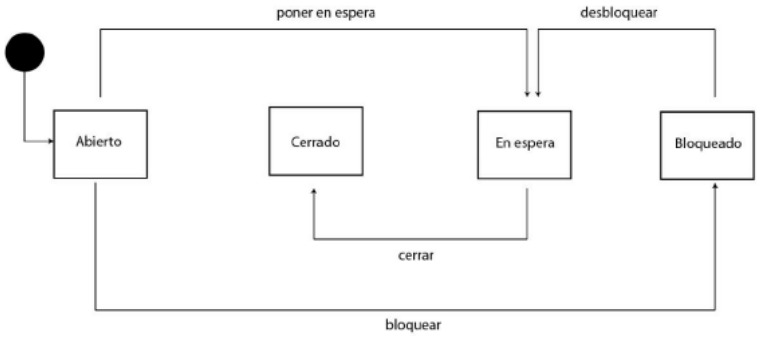
3.2. DIAGRAMA DE SECUENCIAS.

Los diagramas de secuencia modelan el paso de mensajes entre objetos en un intervalo de tiempo. De tal modo que con este tipo de diagramas podemos describir los procesos que llevará a cabo una acción determinada. Permite mostrar de una forma visual el intercambio de mensajes/procesos dentro de un protocolo.



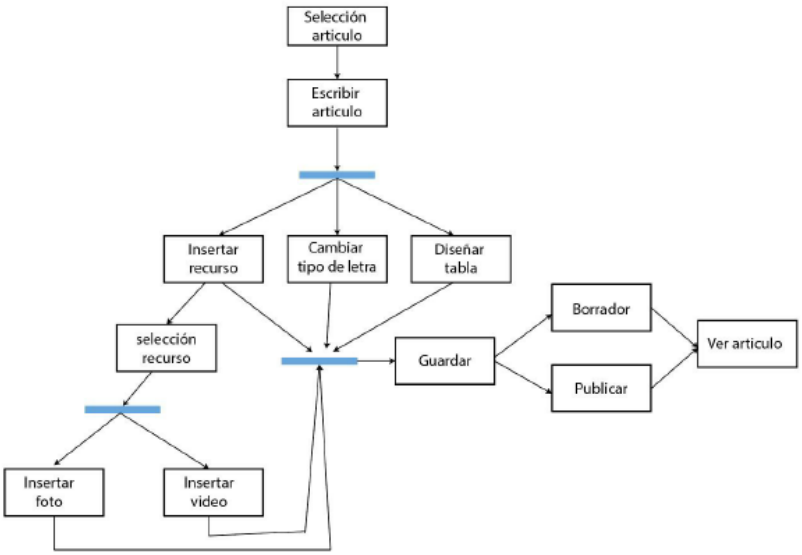
3.3. DIAGRAMA DE ESTADOS.

Modelar los diferentes estados dentro de un algoritmo que tendrá un objeto dentro del programa, en su tiempo de vida.



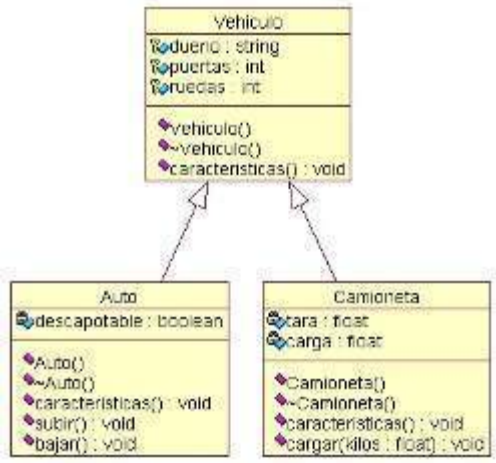
3.4. DIAGRAMA DE ACTIVIDAD.

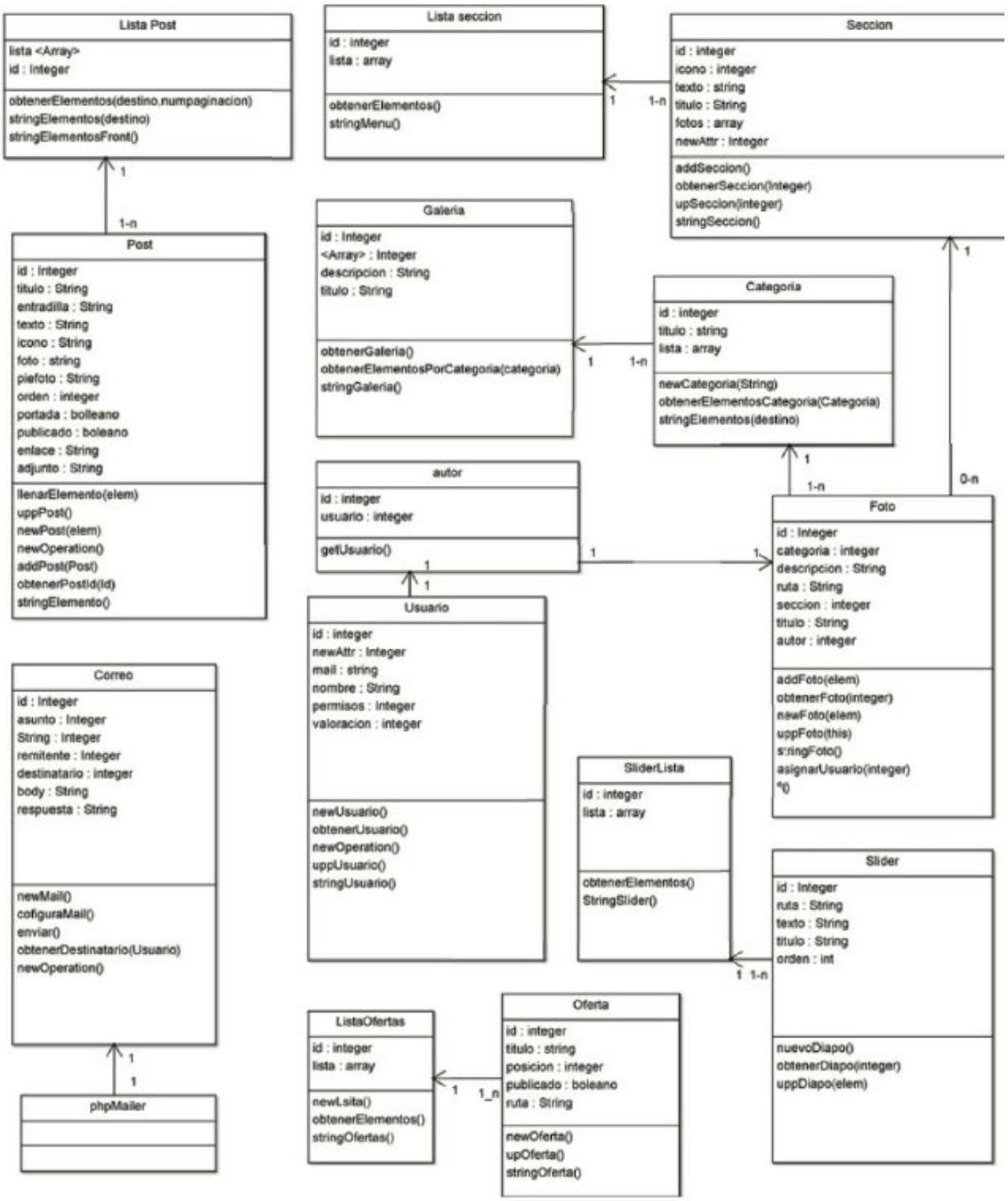
Con los diagramas de actividad determinamos el flujo de trabajo de los componentes del sistema. Se pueden usar como alternativa a los diagramas de estados o para especificar más en detalle un caso de uso.



3.5. DIAGRAMA DE CLASES.

El diagrama de clases es uno de los más importantes del modelado, ya que nos describe la estructura de objetos de nuestra aplicación. Cuando definimos cada clase indicamos los atributos y métodos principales, además de sus relaciones y visibilidad.

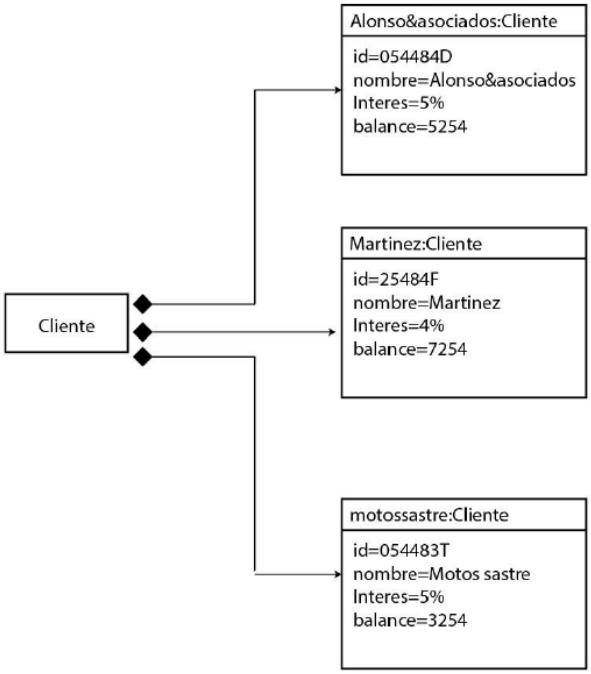




3.6. DIAGRAMA DE OBJETOS

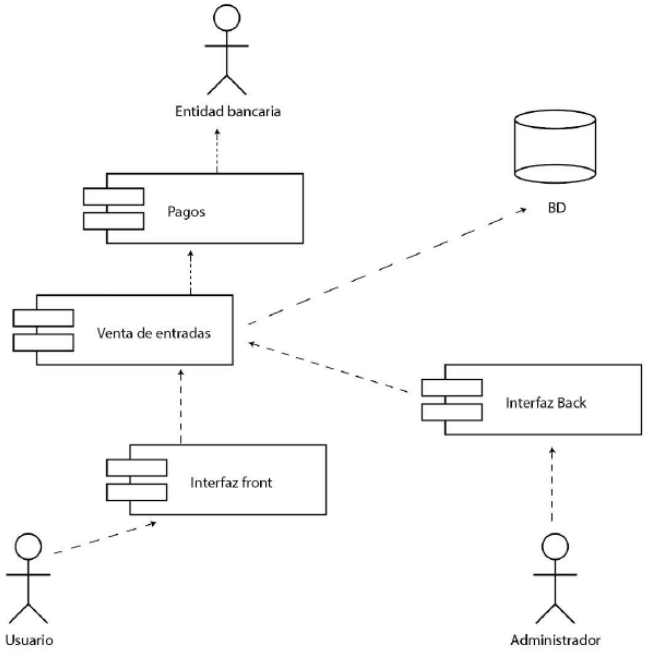
Modelan la estructura de los objetos en un momento dado. Ojo: No confundirlos con los diagramas de clases.

Un objeto cambia continuamente según el proceso que se esté realizando, por lo que los diagramas de objetos solo representan los valores de un instante a lo largo de su vida.



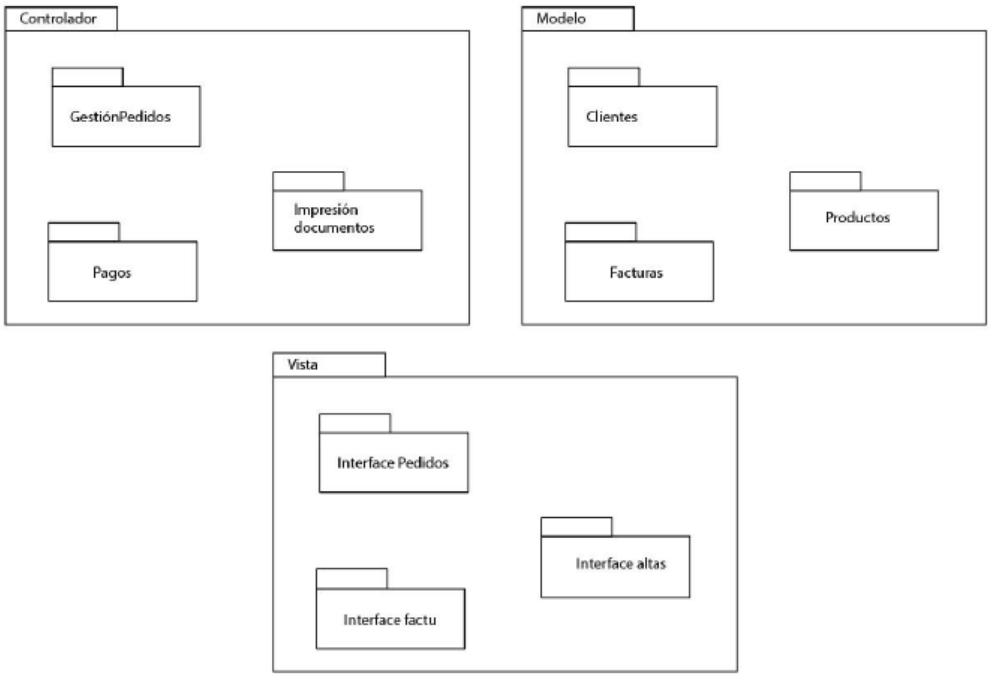
3.7. DIAGRAMA DE COMPONENTES.

El diagrama de componentes *muestra las piezas del software, los controladores utilizados, etc.*



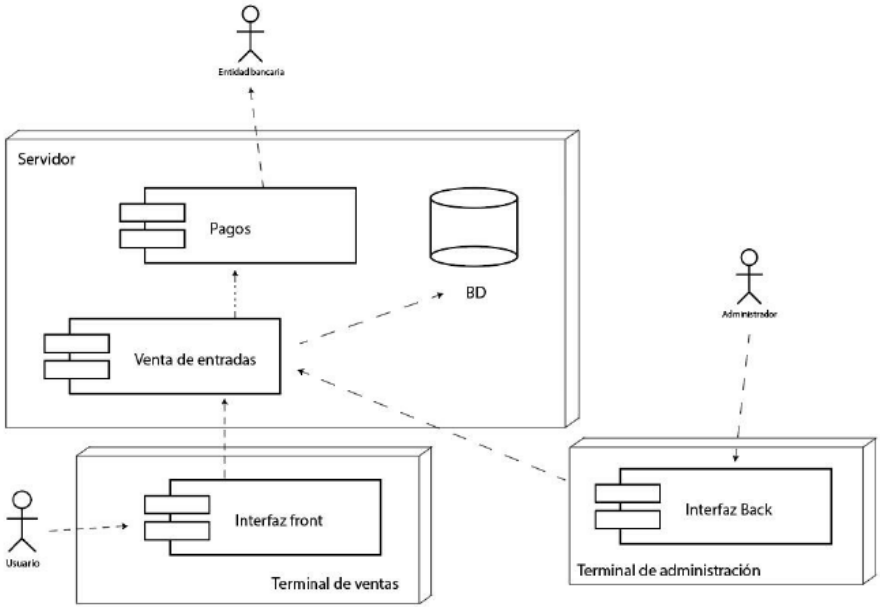
3.9 DIAGRAMA DE PAQUETES.

Los elementos de los paquetes pueden relacionarse con otros elementos de otros paquetes generando una dependencia.



3.10. DIAGRAMA DE IMPLEMENTACIÓN.

El diagrama de implementación indica la disposición y arquitectura del sistema.
Mejora funcional diagrama de



4. ESTRUCTURAS DE LOS DIAGRAMAS UML.

En el punto anterior hemos descrito los principales diagramas de representación de un sistema con UML.
Todos ellos contienen una serie de símbolos y también las relaciones que tienen entre ellos.

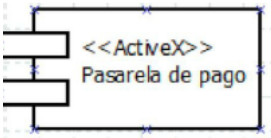
<< algunos símbolos hacen referencia a elementos
estructurales, mientras otros hacen referencia al comportamiento >>

4.1. ELEMENTOS ESTRUCTURALES.

Clase

Las clases son la unidad básica que encapsula la información de un objeto. Está formada por atributos y métodos. En UML se representa por un rectángulo que posee tres divisiones:

- Superior: nombre de la clase.
- Medio: atributos de la clase.
- Inferior: métodos de la clase.



Componente: Los componentes representan elementos físicos del sistema.

Nodo. Es un elemento físico que existe en tiempo de ejecución. Representa un recurso computacional que consume recursos del sistema.

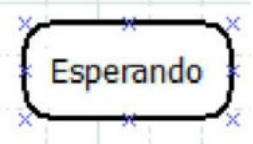
4.2. ELEMENTOS DE COMPORTAMIENTOS.

Interacción

Comportamiento que comprende un conjunto de intercambio de mensajes entre un conjunto de objetos con un contexto particular para lograr un propósito específico. Una interacción involucra otros elementos, incluyendo mensajes, secuencias, enlaces, etc.

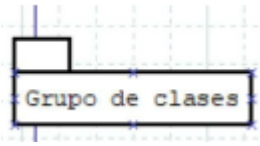
Estado

Comportamiento que especifica la secuencia de estados de un objeto o una interacción durante su vida en respuesta a eventos, junto con las respuestas de esos eventos.



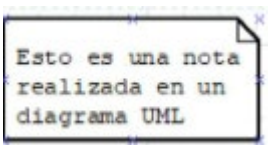
Paquetes

Los paquetes son las partes de organización de los modelos UML. Un paquete es un mecanismo de propósito general para la organización de elementos en grupos



Anotación.

Partes explicativas de los modelos de UML. Una nota es simplemente un símbolo para representarlas limitaciones y comentarios asociados a un elemento o una colección de elementos.

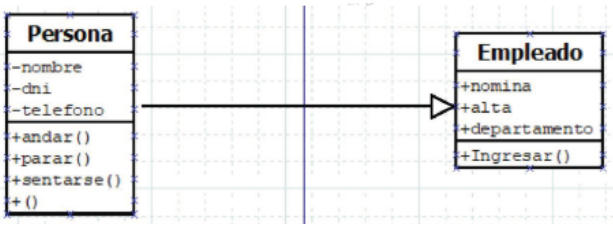


4.5. RELACIONES

Dependencia

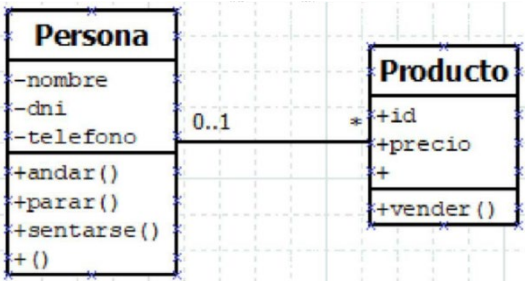
Relación semántica entre dos elementos en la que el cambio en un elemento (el independiente) puede afectar al otro (el dependiente).

Generalización. Relación de especialización y/o generalización, en la cual los objetos del elemento especializado (el hijo) son sustituidos por características del elemento generalizado (el padre).



Asociación

Relación estructural que describe un conjunto de enlaces (conexiones entre objetos). La agregación es un tipo especial que representa una relación estructural entre un todo y sus partes.



HERRAMIENTAS MUY SENCILLAS PARA REALIZAR DIAGRAMAS DE FLUJO:

- https://www.draw.io/
- https://www.lucidchart.com/

- https://www.genmymodel.com/