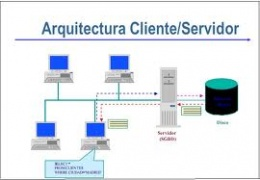
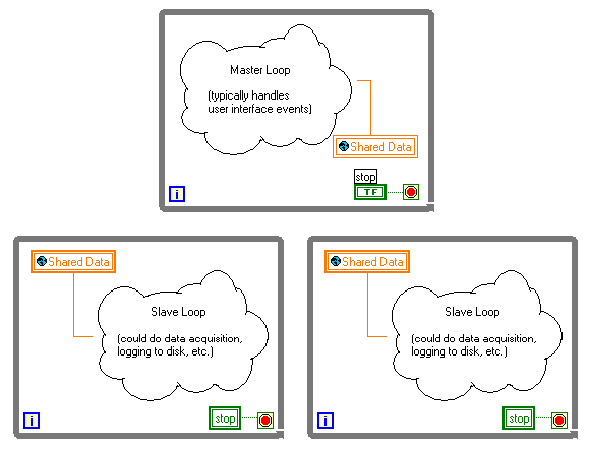
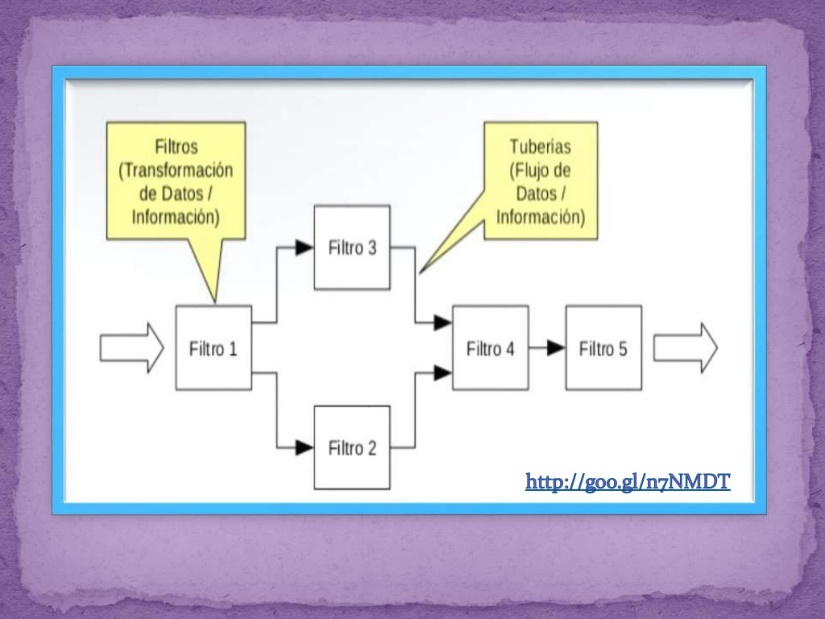
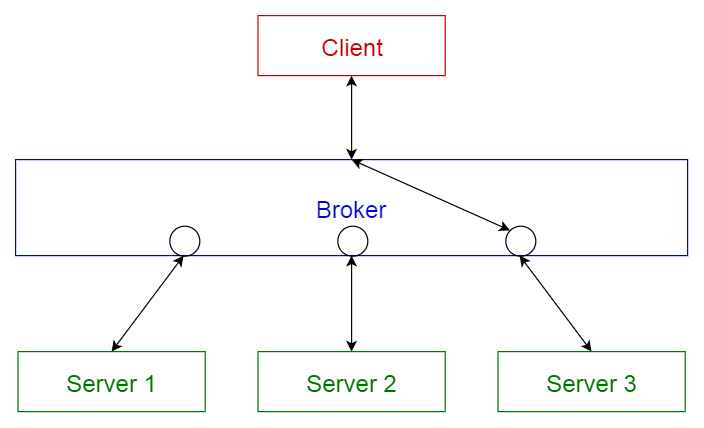
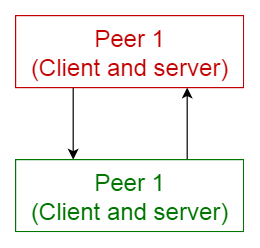
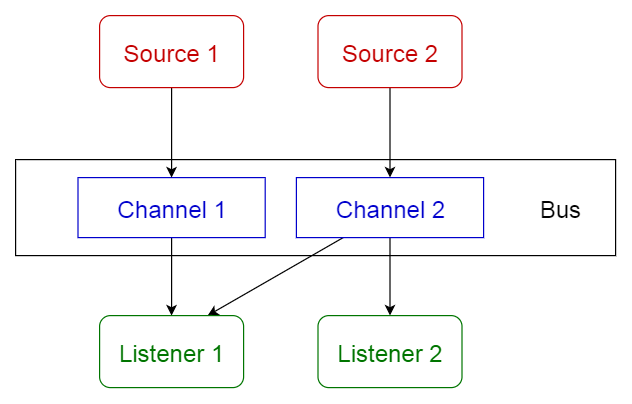
Arquitecturas de desarrollo

1. Patrón de capas: es una de las técnicas mas comunes que se utiliza para dividir sistemas de software complicados, en este esquema la capa más alta utiliza varios servicios definidos por la inferior.
2. Patrón Cliente-Servidor: consiste en un cliente que realiza peticiones a otro programa (el servidor) que le da respuesta, la interacción cliente-servidor es el soporte de la mayor parte de la comunicación por redes.
3. Patrón Maestro-esclavo: es utilizada cuando se tienen dos o mas procesos que necesitan ejecutarse simultáneamente y continuamente a dos velocidades.
4. Patrón flujo de datos: es un conjunto de componentes denominados “filtros”, conectados entre si por “tuberías”, que transmiten los datos desde un componente al siguiente, cada filtro trabaja de manera independiente situado antes o después de él.
5. Patrón intermediario: este patrón se una para estructurar sistemas distribuidos con componentes desacoplados. Estos componentes pueden interactuar entre sí mediante invocaciones de servicios remotos.



1. Patrón de igual a igual: en este patrón, los componentes individuales se conocen como pares. Los pares pueden funcionar tanto como un cliente, solicitando servicio de otros pares, y como un servidor, proporcionando servicios a otros pares.
2. Patrón de bus de evento: este patrón principalmente con eventos y tiene 4 componentes principales; fuentes de evento, escucha de evento, canal y bus de eventos. Las fuentes publican los mensajes en canales particulares en un bus de eventos.

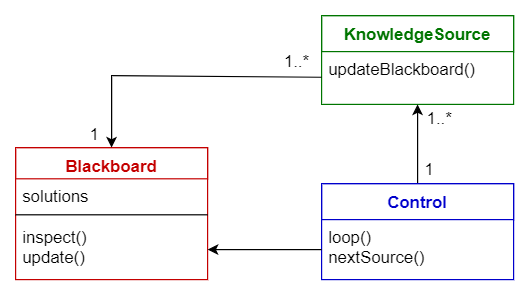


1. Patrón de modelo-vista-controlador: divide una aplicación interactiva en 3 pares, como
2. Modelo: contiene la funcionalidad y los datos básicos.
3. Vista: muestra la información al usuario (se puede definir más de una vista).
4. Controlador: maneja la entrada del usuario.

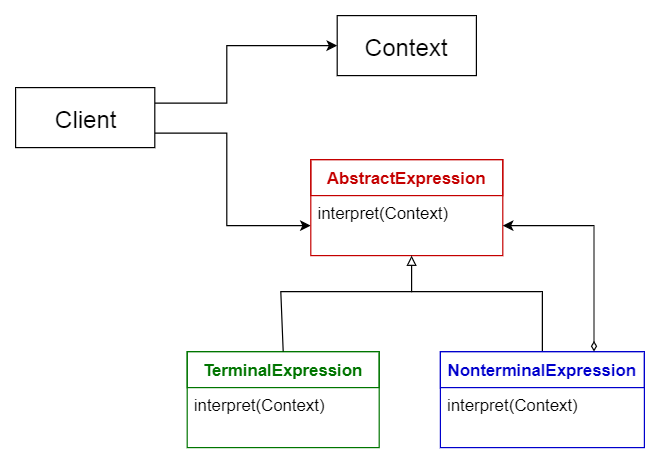


Esto se hace para separar las representaciones internas de información de las formas en que se presenta y acepta la información de usuario. Desacopla los componentes y permite la reutilización eficiente del código.

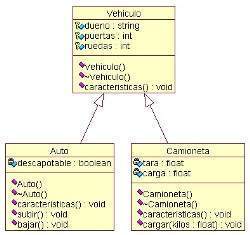
1. Patrón de pizarra: este patrón es útil para problemas para los que no conocen estrategas de solución deterministas. El patrón de pizarra conta de 3 componentes principales.
2. Pizarra: una memoria global estructurada que contiene objetos del espacio de solución.
3. Fuente de conocimiento: módulos especializados con su propia representación.
4. Componente de control: selecciona, configura y ejecuta módulos.



1. Patrón de interprete: este patrón se usa para diseñar un componente que interpreta programas escritos en un lenguaje dedicado. Especifica principalmente como evaluar las líneas de programas, conocidas como oraciones o expresiones escritas en un idioma particular.



1. Diagrama de clases: es un tipo de diagrama de estructura estática que describe la estructura de un sistema, sus atributos, operaciones (o métodos), y las relaciones entre los objetos, sirve para visualizar las relaciones entre clases que involucran el sistema, las cuales pueden ser asociativas, de herencia, de uso y de agregación.



**Ventajas**

* Genera un código automáticamente.
* Propone soluciones a algunos errores.
* Representa las relaciones entre las clases de sistema.
* Se diseña los componentes de los sistemas.
* Se protegen los datos.
* Se posibilita una reducción de acoplamiento.
* Mas fácil la comunicación entre los programadores, descubrimiento de fallas del sistema en el diseño Mejor diseño del sistema ofrece más documentación.

**Desventajas**

* Los diagramas de clases especifican qué clases hay y cómo están relacionadas, pero no cómo interactúan para alcanzar comportamientos particulares.
* El método tiende hacer muy lento.
* La instalación es muy costosa