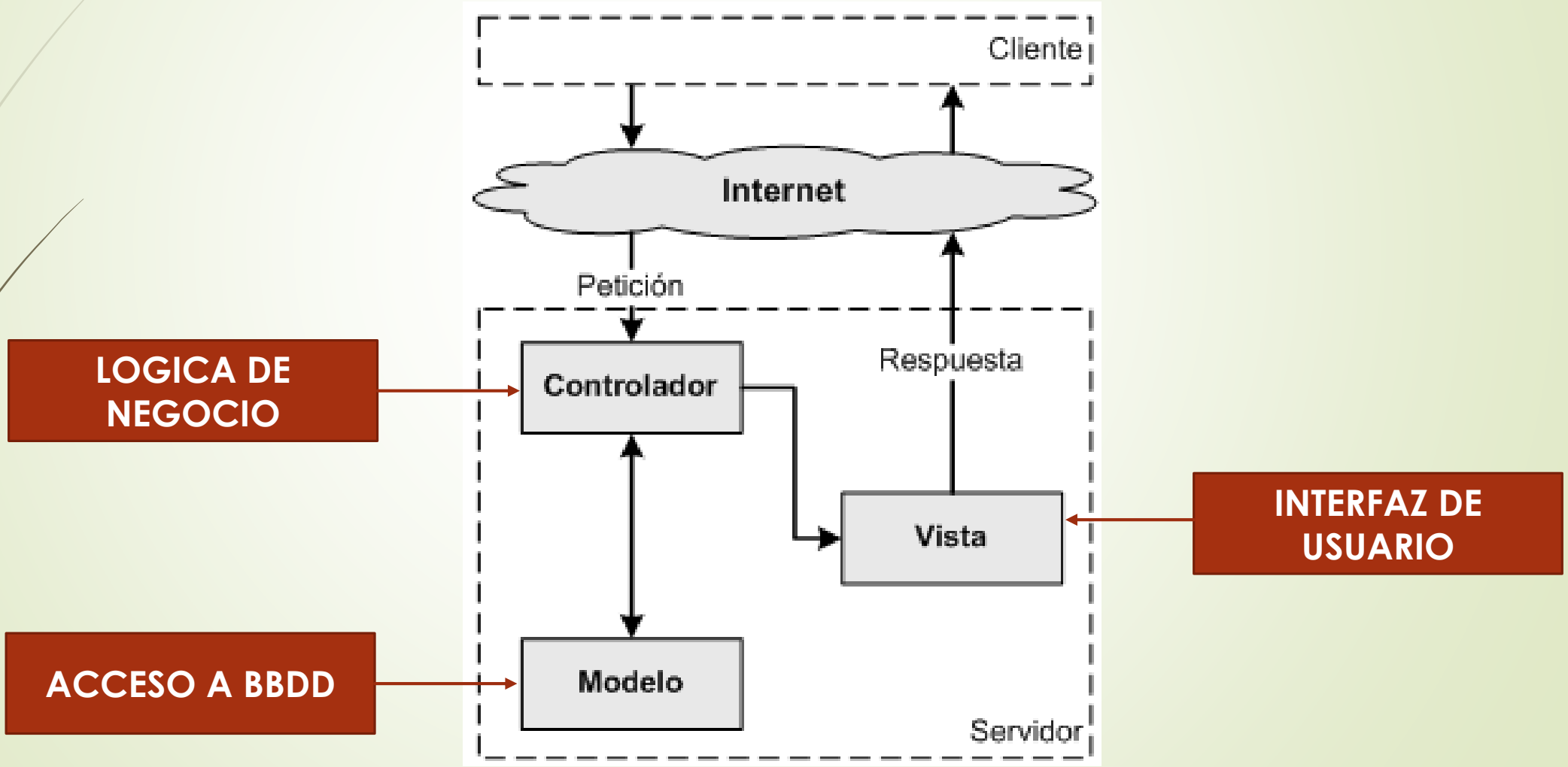
MODELO VISTA CONTROLADOR

DESARROLLO WEB EN ENTORNO SERVIDOR

INTRODUCCION

- En el **paradigma MVC**, las entradas del usuario, los modelos del mundo exterior y la retroalimentación visual son **explícitamente separados**.
- El **modelo–vista–controlador (MVC)** es un patrón de arquitectura de software, que separa los datos y la lógica de negocio de una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones. Para ello MVC propone la construcción de tres componentes distintos que son el **modelo**, la **vista** y el **controlador**, es decir, por un lado define componentes para la representación de la información, y por otro lado para la interacción del usuario. Este patrón de arquitectura de software se basa en las ideas de reutilización de código y la separación de conceptos, características que buscan facilitar la tarea de desarrollo de aplicaciones y su posterior mantenimiento.

ESQUEMA DEL PATRON MVC



ELEMENTOS DEL PATRON MVC

- El **Modelo**: Es la representación de la información con la cual el sistema opera, por lo tanto **gestiona todos los accesos a dicha información**, tanto consultas como actualizaciones, implementando también los privilegios de acceso que se hayan descrito en las especificaciones de la aplicación (lógica de negocio). Envía a la 'vista' aquella parte de la información que en cada momento se le solicita para que sea mostrada (típicamente a un usuario). Las peticiones de acceso o manipulación de información llegan al 'modelo' a través del 'controlador'.
- El **Controlador**: **Responde a eventos** (usualmente acciones del usuario) **e invoca peticiones al 'modelo'** cuando se hace alguna solicitud sobre la información (por ejemplo, editar un documento o un registro en una base de datos). También puede **enviar comandos a su 'vista' asociada** si se solicita un cambio en la forma en que se presenta el 'modelo' (por ejemplo, desplazamiento o scroll por un documento o por los diferentes registros de una base de datos), por tanto se podría decir que el 'controlador' hace de **intermediario entre la 'vista' y el 'modelo'**.
- La **Vista**: **Presenta el 'modelo'** (información y *lógica de negocio*) **en un formato adecuado** para interactuar (usualmente la interfaz de usuario) por tanto requiere de dicho 'modelo' la información que debe representar como salida.

VENTAJAS

- **La implementación se realiza de forma modular**.
- **Sus vistas muestran información actualizada siempre**. El programador no debe preocuparse de solicitar que las vistas se actualicen, ya que este proceso es realizado automáticamente por el modelo de la aplicación.
- **Cualquier modificación de tipo aumentar/modificar métodos o cambiar los datos contenidos**, implica una **modificación sólo en el modelo y las interfaces del mismo con las vistas**, no todo el mecanismo de comunicación y de actualización entre modelos.
- **Las modificaciones a las vistas no afectan al modelo ni al controlador**, simplemente se modifica la representación de la información, no su tratamiento.
- MVC es **patrón de diseño que** ofrece una posibilidad **extensibilidad y de mantenibilidad muy amplias**.

DESVENTAJAS

- Para desarrollar una aplicación bajo el patrón de diseño MVC es necesario una **mayor dedicación en los tiempos iniciales del desarrollo**. Normalmente el patrón exige al programador desarrollar un mayor número de clases que, en otros entornos de desarrollo, no son necesarias. Sin embargo, esta desventaja es muy relativa ya que posteriormente, en la etapa de mantenimiento de la aplicación, una aplicación MVC es mucho más mantenible, extensible y modificable que una aplicación que no lo implementa.
- **MVC requiere la existencia de una arquitectura inicial** sobre la que se deben construir clases e interfaces para modificar y comunicar los módulos de una aplicación. Esta arquitectura inicial debe incluir, por lo menos, un mecanismo de eventos para poder proporcionar las notificaciones que genera el modelo de aplicación; una clase Modelo, otra clase Vista y una clase Controlador genéricas que realicen todas las tareas de comunicación, notificación y actualización que serán luego transparentes para el desarrollo de la aplicación.
- **MVC es un patrón de diseño preferiblemente orientado a objetos** por lo que su implementación es sumamente costosa y difícil en lenguajes que no siguen este paradigma.