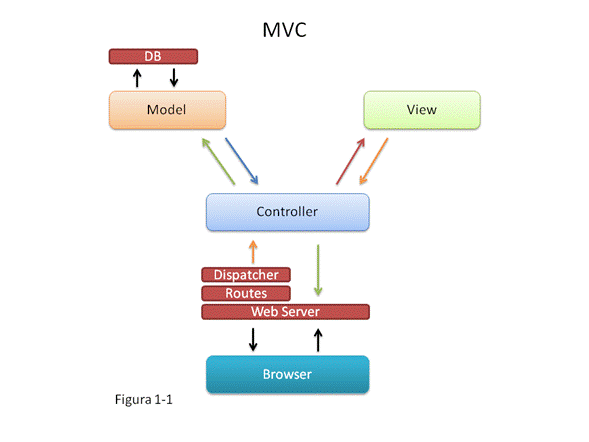
***Modelos Vista Controlado***

*Es un enfoque de software que separa la lógica de una aplicación de la presentación o interfaz de usuario.*

*Esto surge de la necesidad de crear software más robustos y ordenado con un ciclo de vida más adecuado, donde se potencie la facilidad de mantenimiento, la reutilización del código y la separación de conceptos.*



*De manera genérica, los componentes de MVC se podrían definir como sigue:*

* ***Modelo****: Representa las estructuras de datos. Típicamente las clases del modelo contendrán funciones que nos ayudaran a mostrar, insertar, actualizar y eliminar información de la base de datos.*

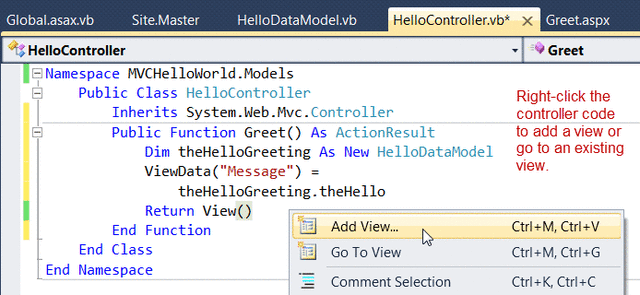
*El modelo es el responsable de:*

* *Acceder a la capa de almacenamiento de datos. Lo ideal es que el modelo sea independiente del sistema de almacenamiento.*
* *Define las reglas de negocio (la funcionalidad del sistema). Un ejemplo de regla puede ser: "Si la mercancía pedida no está en el almacén, consultar el tiempo de entrega estándar del proveedor".*
* *Lleva un registro de las vistas y controladores del sistema.*
* *Si estamos ante un modelo activo, notificará a las vistas los cambios que en los datos pueda producir un agente externo (por ejemplo, un fichero por lotes  que actualiza los datos, un temporizador que desencadena una inserción, etc.)*
* ***Controlador****: Sirve como un intermediario entre el Modelo, la Visa y cualquier otro recurso necesario para procesar la solicitud HTTP y echar a andar una página web.*

*Aquí veremos todo el código relacionado con la lógica y funcionamiento de las aplicaciones web.*

*El controlador es responsable de:*

* *Recibe los eventos de entrada (un clic, un cambio en un campo de texto, etc.).*
* *Contiene reglas de gestión de eventos, del tipo "SI Evento Z, entonces Acción W". Estas acciones pueden suponer peticiones al modelo o a las vistas. Una de estas peticiones a las vistas puede ser una llamada al método "Actualizar()". Una petición al modelo puede ser "Obtener\_tiempo\_de\_entrega ( nueva\_orden\_de\_venta )".*

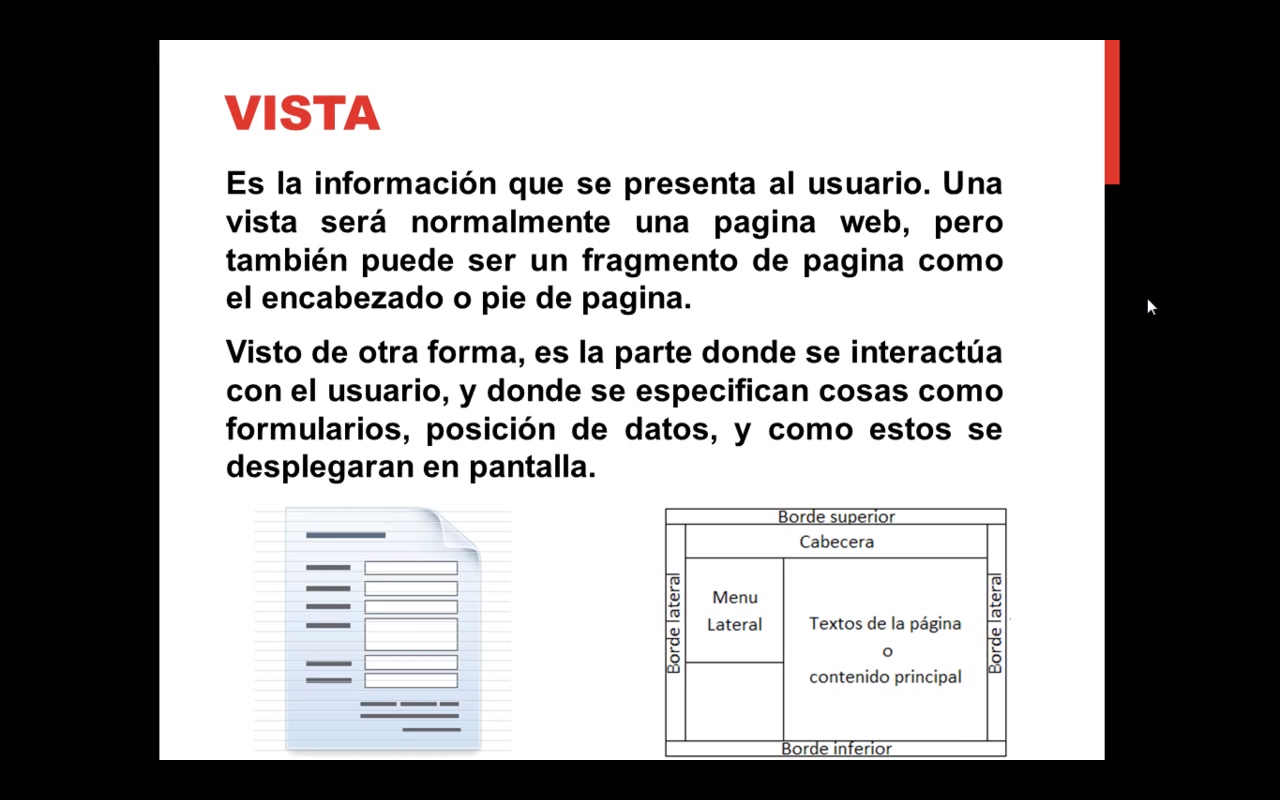


* ***Vista****: Es la información que se presenta al usuario. Una vista será normalmente una pagina web, pero también puede ser un fragmento de pagina como el encabezado o pie de pagina.*

*Visto de otra forma, es la parte donde se interactúa con el usuario, y donde se especifican cosas como formularios, posición de datos y como estos se desplegaran en pantalla*

*Las vistas son responsables de****:***

* *Recibir datos del modelo y los muestra al usuario.*
* *Tienen un registro de su controlador asociado (normalmente porque además lo instancia).*
* *Pueden dar el servicio de "Actualización ()", para que sea invocado por el controlador o por el modelo (cuando es un modelo activo que informa de los cambios en los datos producidos por otros agentes).*

**

***Pasos en el modelo vista controlador***

*Cada desarrollador tiene la libertad de organizarse como mejor le parezca. Lo importantes es visualizar los resultados de la aplicación.  Sin embargo, esta es de las formas posibles que para utilizar el MVC en tu aplicación. Puedes empezar creando las clases, luego creando la Interfaz de Usuario y finalmente trabajar la lógica del programa.*

* *Paso 1 – Diseño de las clases*

*Crear el diagrama UML para la aplicación.*

*Cada “cosa” en su programa llegará a ser una clase. Por ejemplo una institución, materia, persona, vehículo, etc.*

*Cada clase tendrá atributos como el color, cantidad. También tendrá  métodos o acciones como caminar, frenar, levantar, etc.*

* *Paso 2 – Interface de usuario.*

*Ahora es importante saber que información debe aparecer en  las pantallas. En muchos casos esto podría llegar  a verse como un formulario.*

* *Paso 3 – Codificación.*

*Ahora usted puede enlazar las clases y la interfaz de usuario.*

*Esta es la parte de programación y viene a ser la última, no la primera.*

*Algunos lenguajes usados para cada capa del MVC son:*

***Modelo:*** *MySQL, SQLite, ORACLE y SQL Server.*

***Vista:*** *CSS3, HTML5, jQuerv AJAX y JavaScript.*

***Controlador:*** *PHP, .NET y Java.*

***Interacción de los componentes***

*Aunque se pueden encontrar diferentes implementaciones de****MVC****, el flujo de control que se sigue generalmente es el siguiente:*

1. *El usuario interactúa con la interfaz de usuario de alguna forma (por ejemplo, el usuario pulsa un botón, enlace, etc.)*
2. *El controlador recibe (por parte de los objetos de la*[*interfaz*](https://es.wikipedia.org/wiki/Interfaz_de_usuario)*-vista) la notificación de la acción solicitada por el usuario. El controlador gestiona el evento que llega, frecuentemente a través de un gestor de eventos (handler) o callback.*
3. *El controlador accede al modelo, actualizándolo, posiblemente modificándolo de forma adecuada a la acción solicitada por el usuario (por ejemplo, el controlador actualiza el carro de la compra del usuario). Los controladores complejos están a menudo estructurados usando un*[*patrón de comando*](https://es.wikipedia.org/wiki/Command_(patr%C3%B3n_de_dise%C3%B1o))*que encapsula las acciones y simplifica su extensión.*
4. *El controlador delega a los objetos de la vista la tarea de desplegar la interfaz de usuario. La vista obtiene sus datos del modelo para generar la interfaz apropiada para el usuario donde se reflejan los cambios en el modelo (por ejemplo, produce un listado del contenido del carro de la compra). El modelo no debe tener conocimiento directo sobre la vista. Sin embargo, se podría utilizar el patrón*[*Observador*](https://es.wikipedia.org/wiki/Observer_(patr%C3%B3n_de_dise%C3%B1o))*para proveer cierta indirección entre el modelo y la vista, permitiendo al modelo notificar a los interesados de cualquier cambio. Un objeto vista puede registrarse con el modelo y esperar a los cambios, pero aun así el modelo en sí mismo sigue sin saber nada de la vista. Este uso del patrón*[*Observador*](https://es.wikipedia.org/wiki/Observer_(patr%C3%B3n_de_dise%C3%B1o))*no es posible en las aplicaciones Web puesto que las clases de la vista están desconectadas del modelo y del controlador. En general el controlador no pasa objetos de dominio (el modelo) a la vista aunque puede dar la orden a la vista para que se actualice. Nota: En algunas implementaciones la vista no tiene acceso directo al modelo, dejando que el controlador envíe los datos del modelo a la vista. Por ejemplo en el MVC usado por Apple en su framework Cocoa. Suele citarse como Modelo-Interface-Control, una variación del MVC más puro*
5. *La interfaz de usuario espera nuevas interacciones del usuario, comenzando el ciclo nuevamente....*

***MVC y bases de datos***

*Muchos sistemas*[*informáticos*](https://es.wikipedia.org/wiki/Inform%C3%A1ticos)*utilizan un*[*Sistema de Gestión de Base de Datos*](https://es.wikipedia.org/wiki/Sistema_de_gesti%C3%B3n_de_base_de_datos)*para gestionar los datos que debe utilizar la aplicación; en líneas generales del* ***MVC****dicha gestión corresponde al modelo. La unión entre capa de presentación y capa de negocio conocido en el paradigma de la capas representaría la integración entre la****Vista****y su correspondiente****Controlador****de eventos y acceso a datos, MVC no pretende discriminar entre capa de negocio y capa de presentación pero si pretende separar la capa visual gráfica de su correspondiente programación y acceso a datos, algo que mejora el desarrollo y mantenimiento de la Vista y el Controlador en paralelo, ya que ambos cumplen ciclos de vida muy distintos entre sí.*

***Uso en aplicaciones Web***

*Aunque originalmente MVC fue desarrollado para aplicaciones de escritorio, ha sido ampliamente adaptado como arquitectura para diseñar e implementar aplicaciones en los principales lenguajes de programación. Se han desarrollado multitud de frameworks, comerciales y no comerciales, que implementan este patrón (ver apartado siguiente "Frameworks MVC"); estos frameworks se diferencian básicamente en la interpretación de como las funciones MVC se dividen entre*[*cliente y servidor*](https://es.wikipedia.org/wiki/Cliente-servidor)*.*

*Los primeros frameworks MVC para desarrollo web planteaban un enfoque de*[*cliente ligero*](https://es.wikipedia.org/wiki/Cliente_liviano)*en el que casi todas las funciones, tanto de la vista, el modelo y el controlador recaían en el servidor. En este enfoque, el cliente manda la petición de cualquier*[*hiperenlace*](https://es.wikipedia.org/wiki/Hiperenlace)*o*[*formulario*](https://es.wikipedia.org/wiki/Formulario_web)*al controlador y después recibe de la vista una página completa y actualizada (u otro documento); tanto el modelo como el controlador (y buena parte de la vista) están completamente alojados en el servidor. Como las tecnologías web han madurado, ahora existen frameworks como*[*JavaScriptMVC*](https://en.wikipedia.org/wiki/JavaScriptMVC)*,*[*Backbone*](https://en.wikipedia.org/wiki/Backbone.js)*o [jQuery](https://es.wikipedia.org/wiki/JQuery" \o "JQuery)que permiten que ciertos componentes MVC se ejecuten parcial o totalmente en el cliente (véase*[*AJAX*](https://es.wikipedia.org/wiki/AJAX)*).*

*¿Qué es un Framework?*

*Es un marco de trabajo o conjunto de bibliotecas orientadas a la reutilización de componentes de software para el desarrollo rápido de aplicaciones.*