

UNIVERSIDAD NACIONAL PROGRAMACIONIV

PROF:LIC.CRISTHIAN GARITA
EMAIL:STEBANMAC@GMAIL.COM
TEL:2000-7786



Sitio del Curso: <http://www.moodle.realdesigncr.com/>

ARQUITECTURA DE APLICACIONES

- Definición:
 - La Arquitectura del Software es la organización fundamental de un sistema formada por sus componentes, las relaciones entre ellos y el contexto en el que se implantarán, y los principios que orientan su diseño y evolución?.

ARQUITECTURA DE APLICACIONES

- Tiene la responsabilidad de:
 - Definir los módulos principales
 - Definir las responsabilidades que tendrá cada uno de estos módulos
 - Definir la interacción que existirá entre dichos módulos:
 - Control y flujo de datos
 - Secuenciación de la información
 - Protocolos de interacción y comunicación
 - Ubicación en el hardware

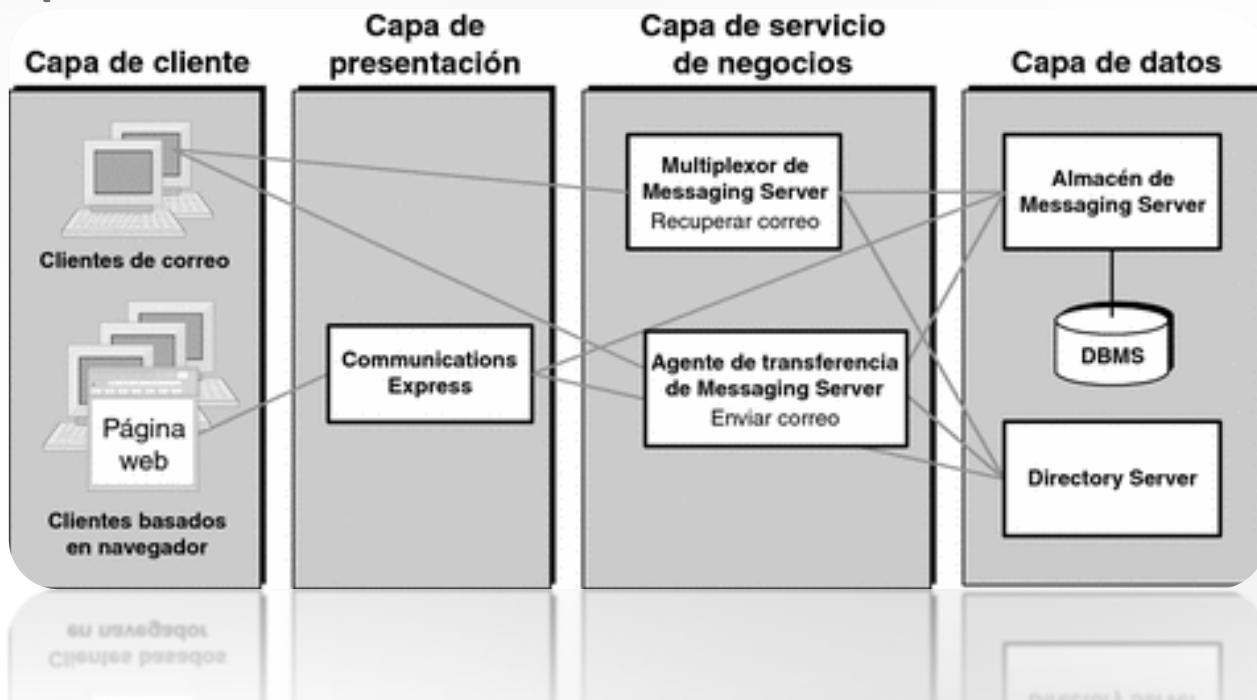
ARQUITECTURA DE APLICACIONES

- La Arquitectura del Software aporta una visión abstracta de alto nivel, posponiendo el detalle de cada uno de los módulos definidos a pasos posteriores del diseño.

ARQUITECTURA APLICA.

EJEMPLO

- Ejemplo de arquitectura separada por componentes



PATRONES DE DISE.

INTRODUCCION

- El diseño OO es difícil y el diseño de software orientado a objetos reutilizable lo es aún más.
- Los diseñadores expertos no resuelven los problemas desde sus principios; reutilizan soluciones que han funcionado en el pasado.
 - Se encuentran patrones de clases y objetos de comunicación recurrentes en muchos sistemas orientados a objetos.
 - Estos patrones resuelven problemas de diseño específicos y hacen el diseño flexible y reusable.

PATRONES DE DISE.

DEFINICION DE UN PATRON

- “Cada patrón describe un problema que ocurre una y otra vez en nuestro entorno y describe también el núcleo de la solución al problema, de forma que puede utilizarse un millón de veces sin tener que hacer dos veces lo mismo.”

PATRONES DE DISE.

DEFINICION DE UN PATRON DE DISENO

- “Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí adaptada para resolver un problema de diseño general en un contexto particular.”

PATRONES DE DISE.

- El uso de patrones ayuda a obtener un software de calidad (reutilización y extensibilidad)
- Elementos de un patrón:
 - Nombre: describe el problema de diseño.
 - El problema: describe cuándo aplicar el patrón.
 - La solución: describe los elementos que componen el diseño, sus relaciones, responsabilidades y colaboración.

PATRONES DE DISE.

CLASIFICACION DE PATRONES DE DISENO

- Según su propósito:
 - De creación: conciernen al proceso de creación de objetos.
 - De estructura: tratan la composición de clases y/o objetos.
 - De comportamiento: caracterizan las formas en las que interactúan y reparten responsabilidades las distintas clases u objetos.

MVC

MODELO-VISTA-CONTROLADOR

- Explicación del patrón de diseño o modelo de abstracción MVC (Modelo-Vista-Controlador), que separa y relaciona los tres componentes principales de una aplicación: la definición de los datos (el modelo), su presentación (la vista) y la definición funcional o lógica de aplicación (el controlador).
- Autor: Georges E. Alfaro S.

MVC

MODELO-VISTA-CONTROLADOR

El modelo encapsula el estado del sistema

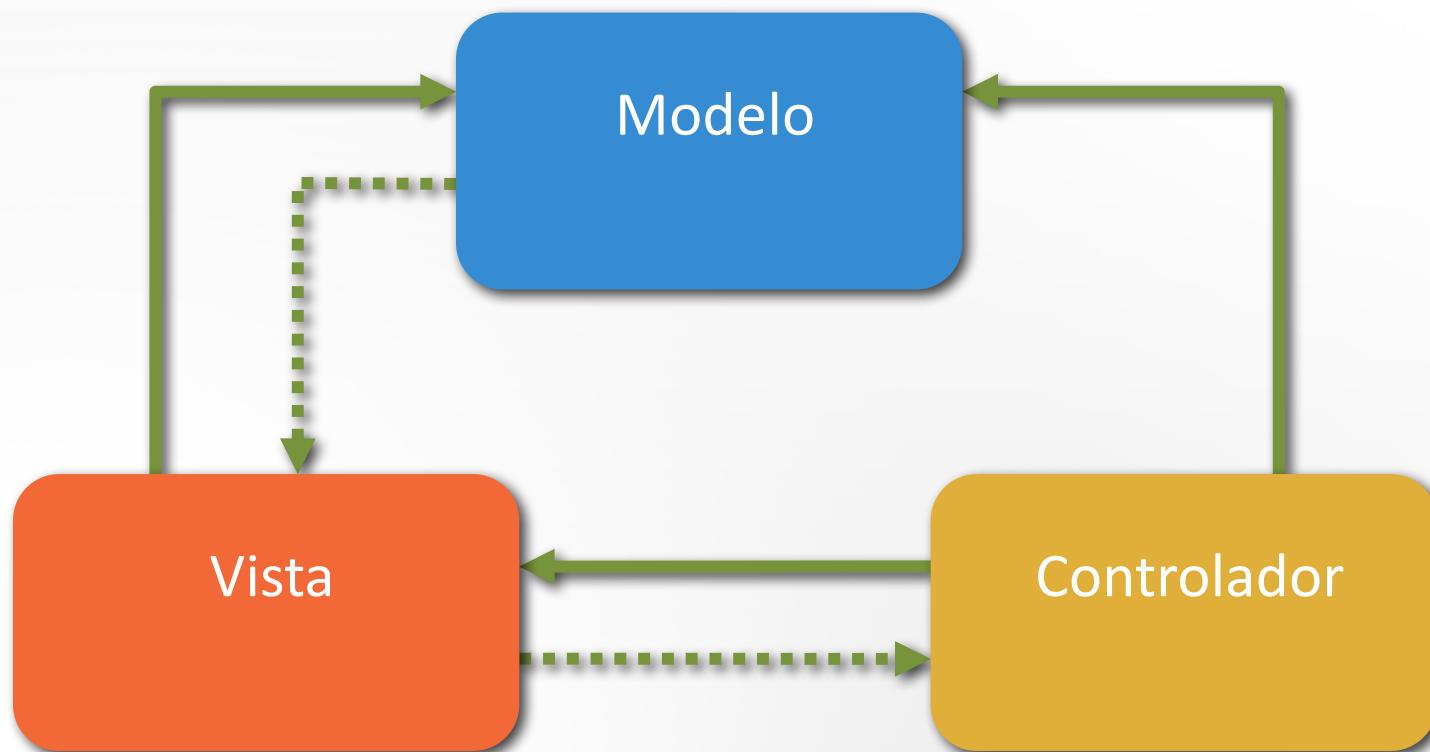
La vista muestra el estado del sistema

El controlador define el comportamiento del sistema

13

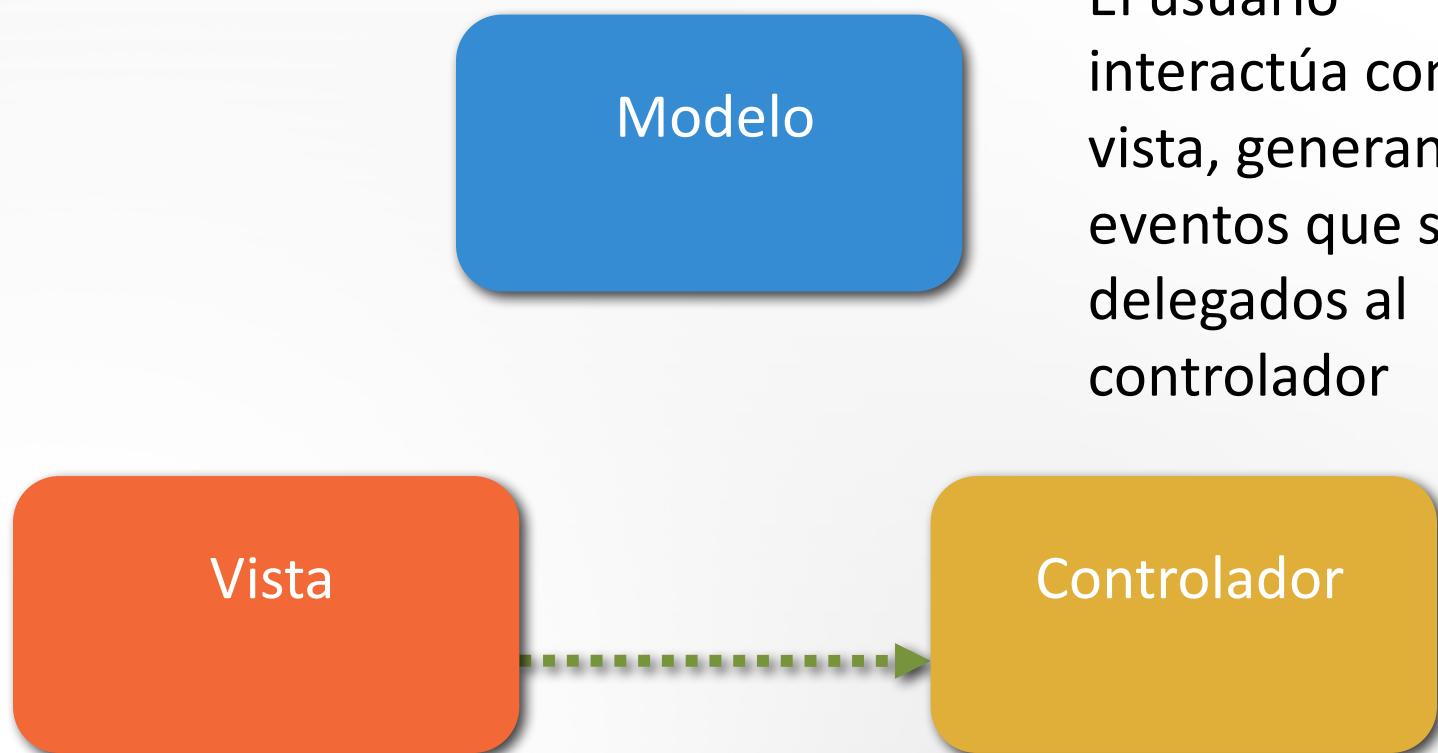
MVC

MODELO-VISTA-CONTROLADOR



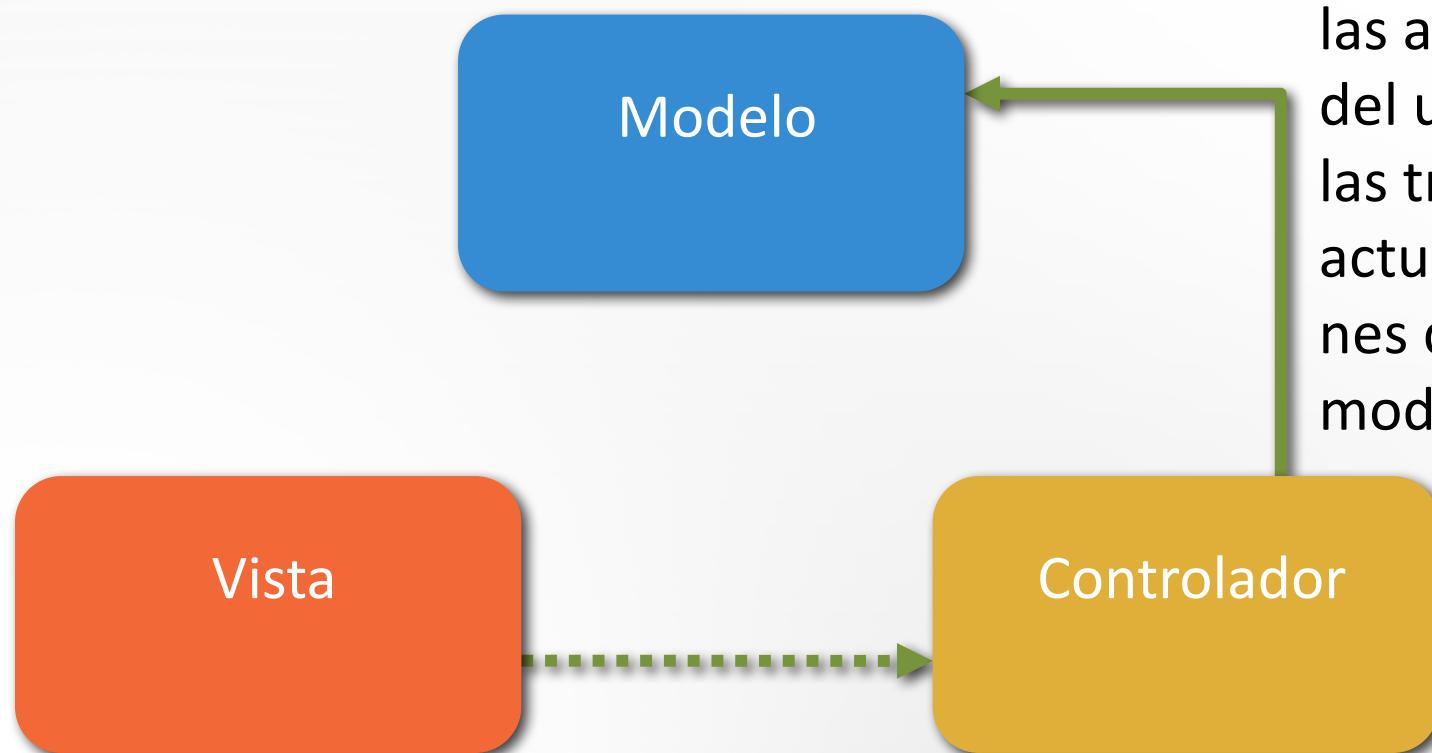
MVC

MODELO-VISTA-CONTROLADOR



MVC

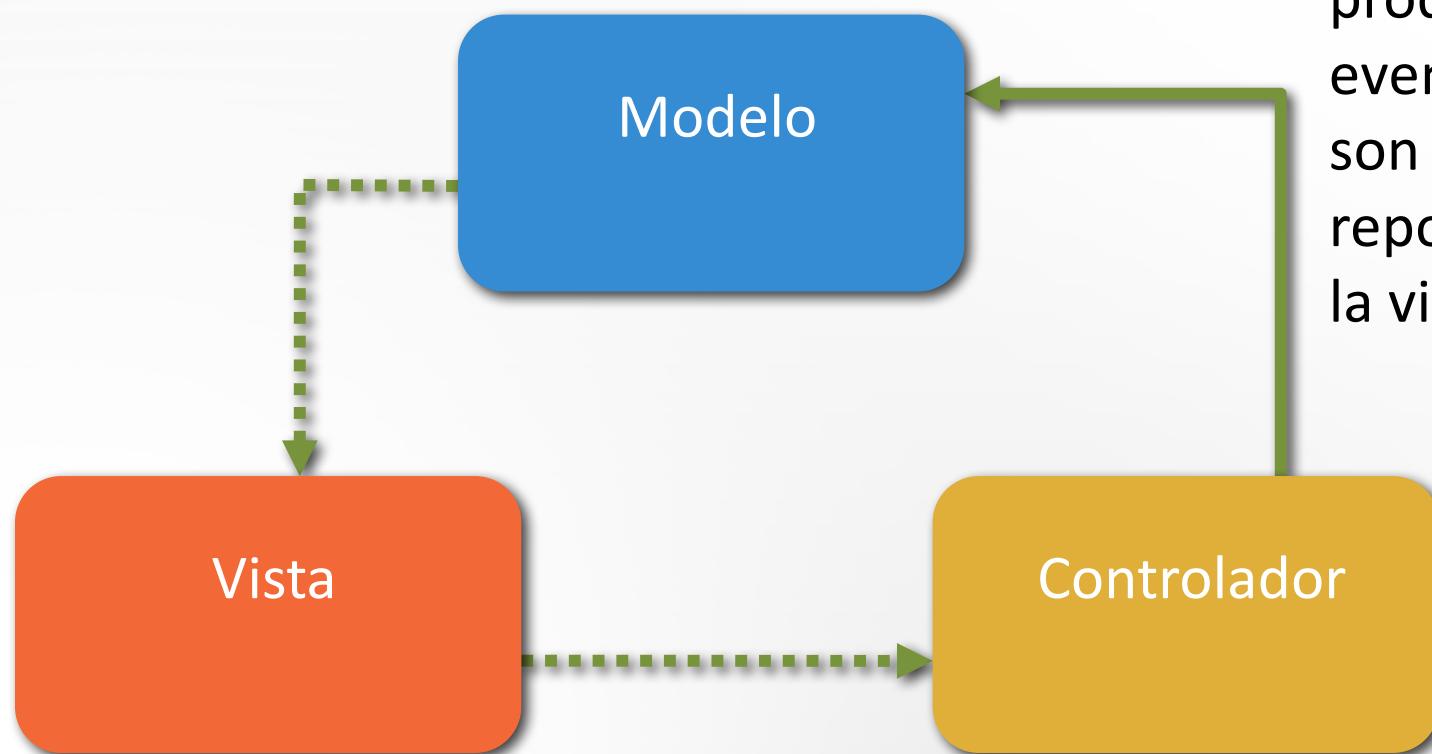
MODELO-VISTA-CONTROLADOR



En el controlador interpreta las acciones del usuario y las traslada a actualizaciones del modelo

MVC

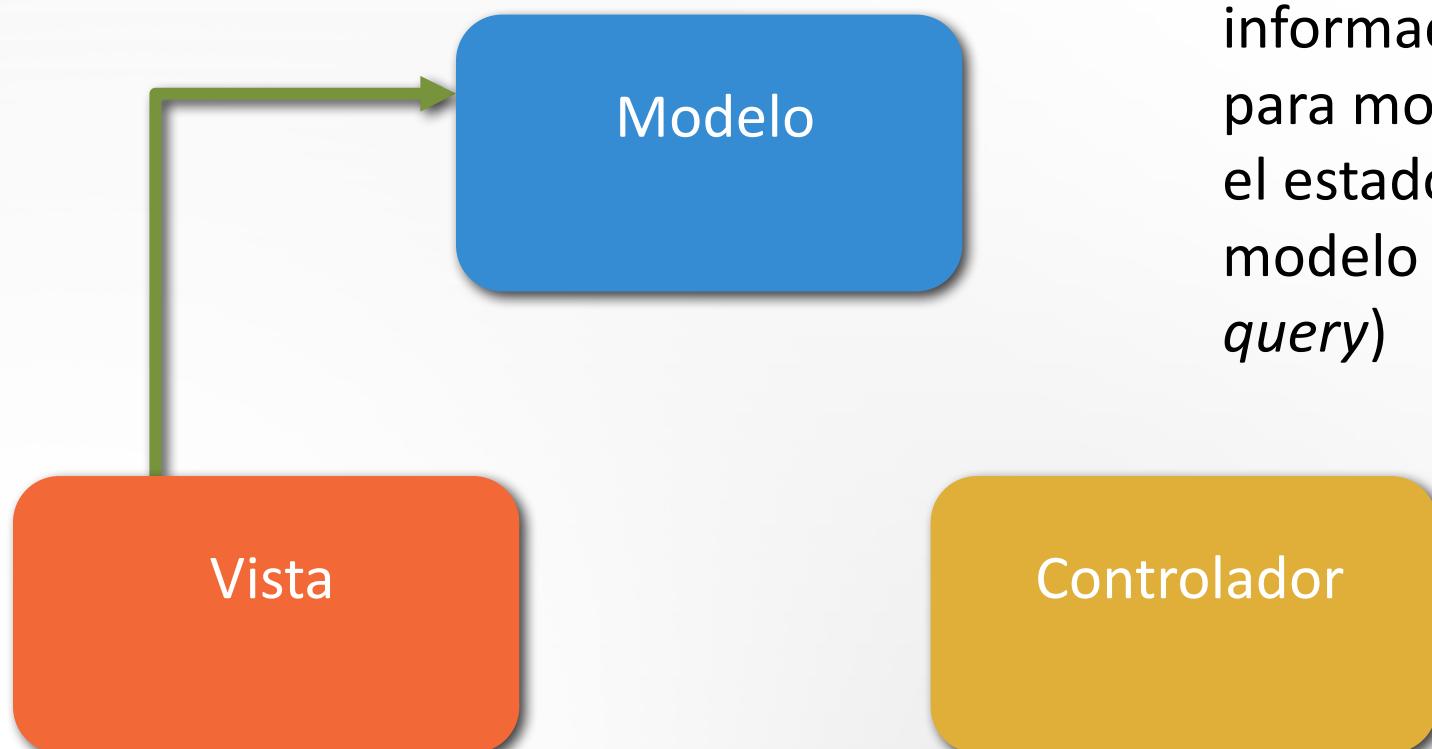
MODELO-VISTA-CONTROLADOR



Los cambios en el modelo producen eventos que son reportados a la vista

MVC

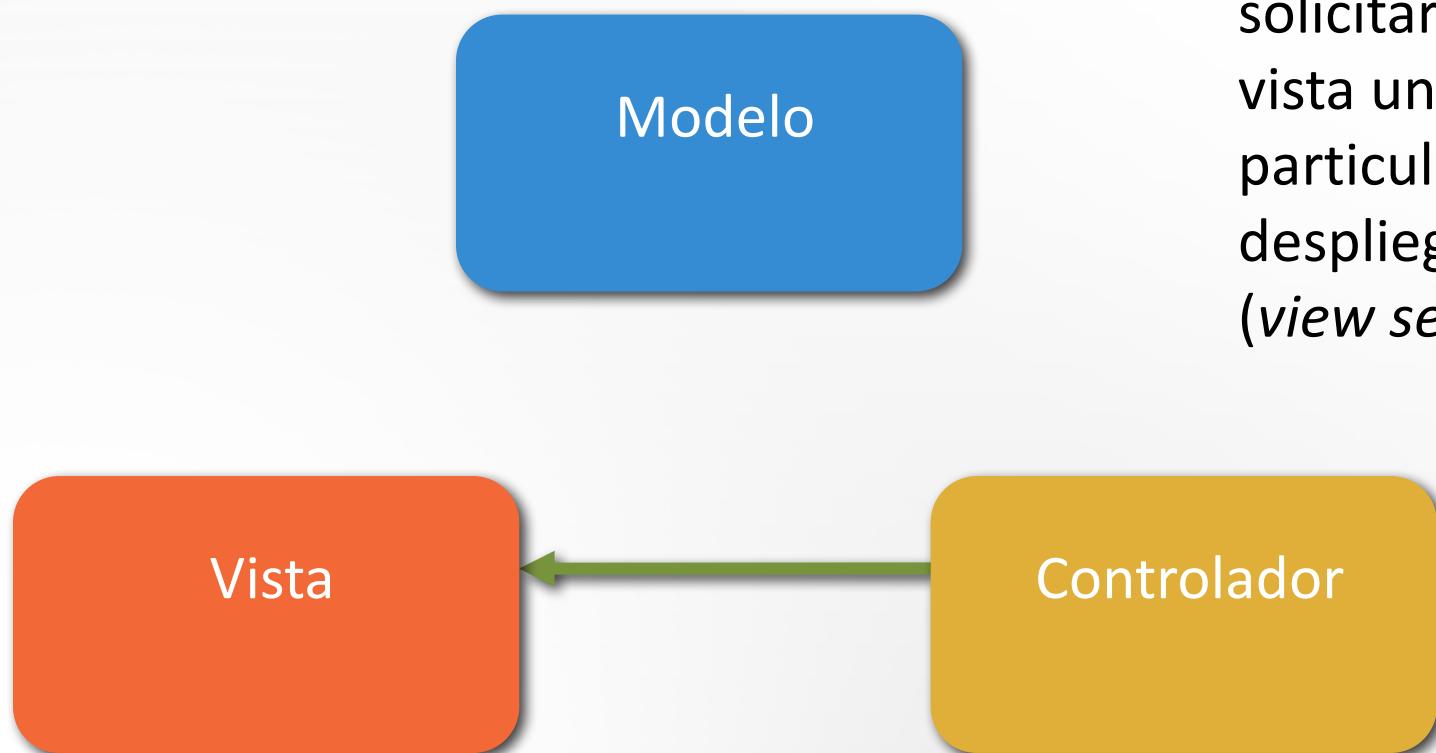
MODELO-VISTA-CONTROLADOR



La vista puede solicitar información para mostrar el estado del modelo (*state query*)

MVC

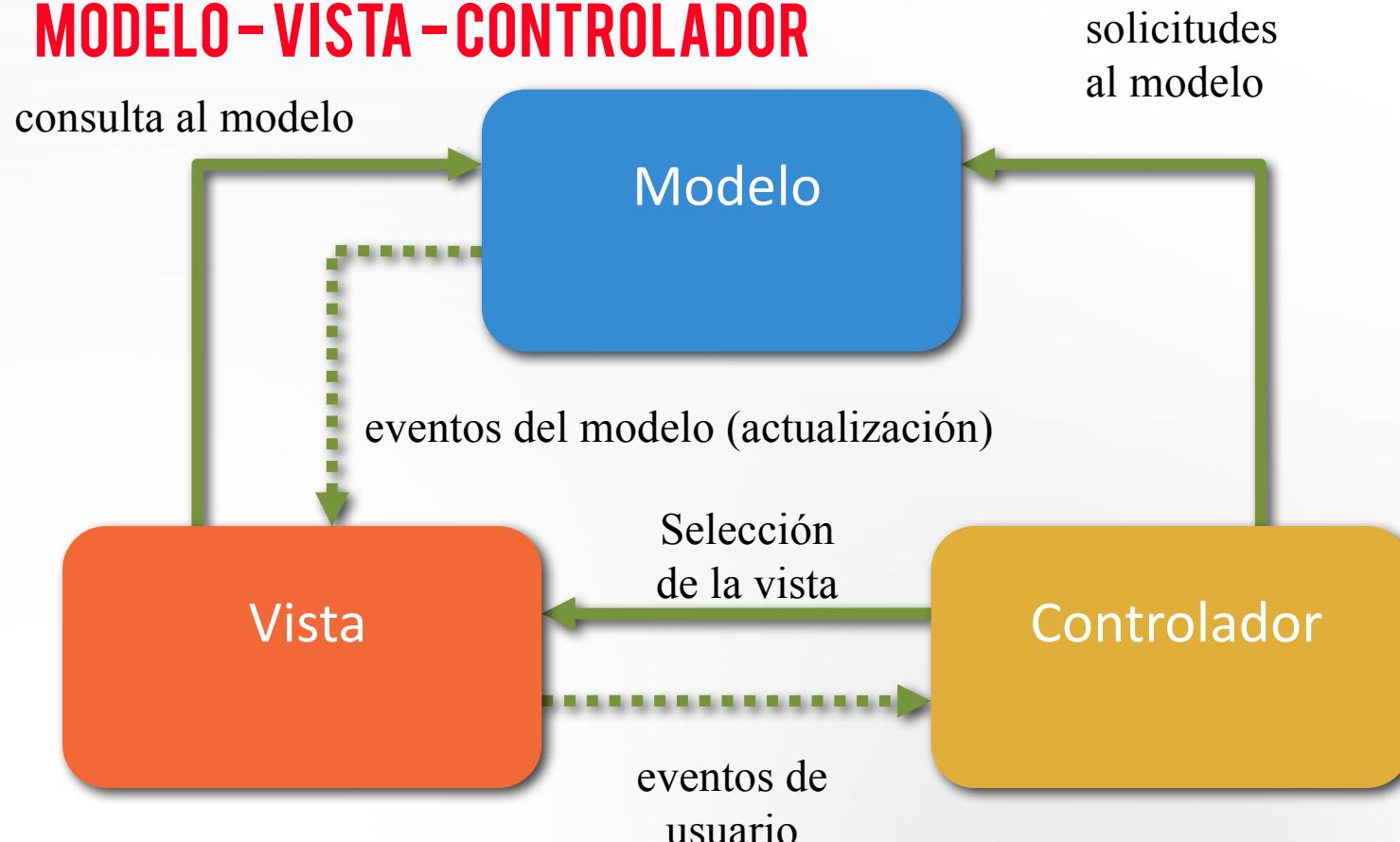
MODELO-VISTA-CONTROLADOR

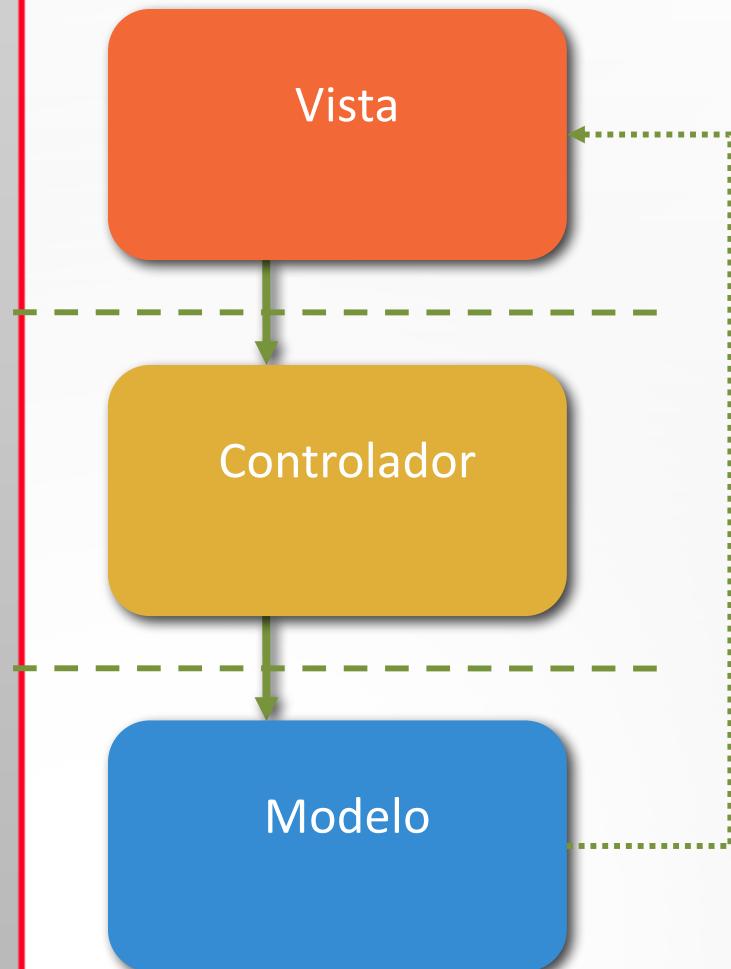


El controlador
puede
solicitarle a la
vista un modo
particular de
despliegue
(*view select*)

MVC

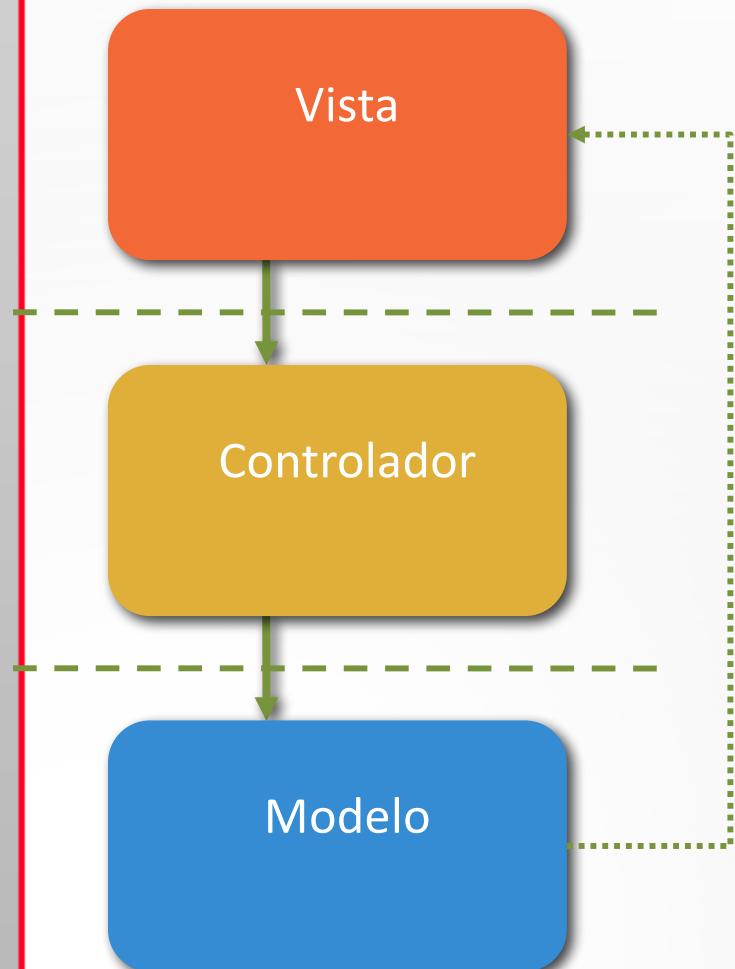
MODELO-VISTA-CONTROLADOR



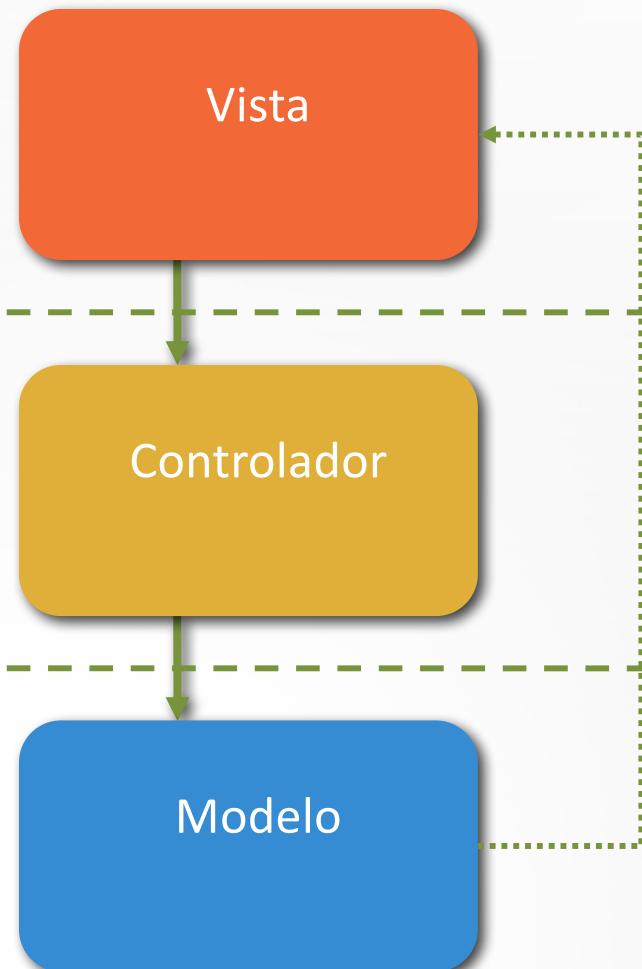


En el patrón MVC, la comunicación entre el modelo y la vista es indirecta.

Se implementa a menudo usando un patrón *Observador*.



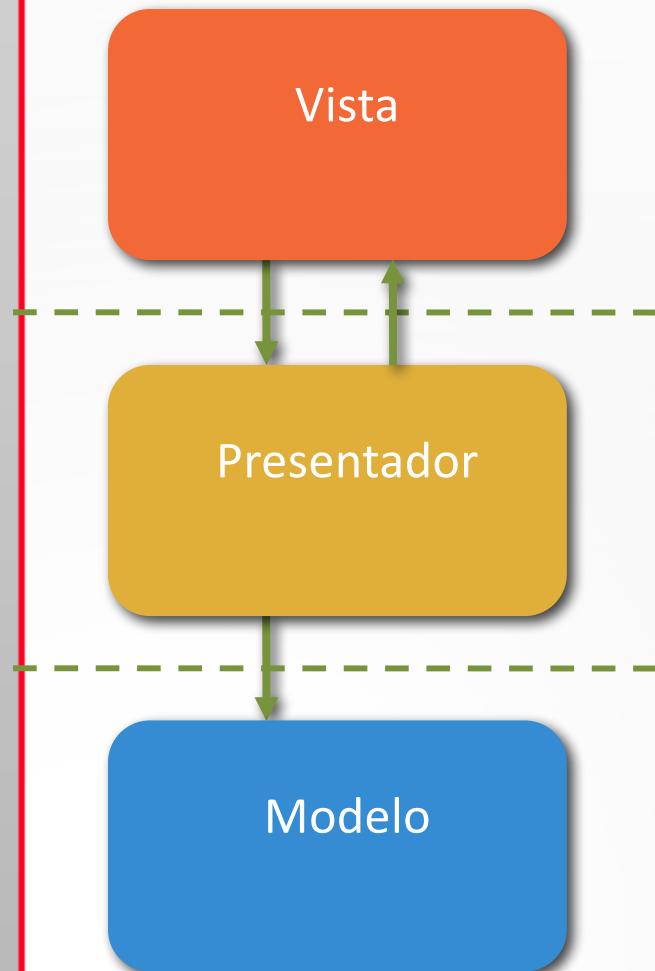
Cada componente de la aplicación se encuentra separado físicamente de los otros, en una biblioteca de clases independiente.



Capa de presentación
(*presentation layer*)

Capa de lógica de aplicación
(*application logic layer*)

Capa de acceso a datos
(*data access layer*)



En la arquitectura MVP (Model-View-Presenter), la interacción entre la vista (presentación) y el modelo es manejada por la misma clase de control (presentador).