|  |
| --- |
| INfortec |
| Arquitectura de Software |
| Sistema de Transporte y Planificación |
|  |
|  |
| **[Seleccione la fecha]** |

|  |
| --- |
| En este documento indicaremos las características de las tecnologías que se utilizan en el desarrollo de este software. |

# Introducción

El uso de un marco en una aplicación presenta muchas ventajas durante la ejecución de una aplicación de software.

Un marco de trabajo proporciona una infraestructura reutilizable que aborda muchos detalles de implementación comunes a través de las aplicaciones y ayuda a evitar volver a descubrir y re-implementación de soluciones a los problemas comunes de implementación. Además, un marco a menudo se ajusta a un modelo de programación que representa una solución eficaz a los problemas de la construcción de un tipo específico de aplicación.

Para Java 2 Enterprise Edition (J2EE) aplicaciones web, los marcos basados ​​en el patrón de diseño Model-View-Controller (MVC) son considerados los mejores de su clase, y entre ellos, Struts ha sido el más popular.

Más recientemente, el comité de estándares J2EE lanzó el JavaServer Faces (JSF) especificación ofrece un framework MVC con beneficios adicionales.  
En este documento se describen las ventajas del patrón de diseño MVC y proporciona una visión general de la JSF.

Se discute su propósito, se enumeran sus ventajas y desventajas, y proporciona directrices sobre cómo elegir entre ellos.

Model-View-Controller

Model-View-Controller (MVC) es un patrón de diseño que separa interfaz de usuario de una aplicación a partir de su lógica de negocio.

Lo hace por capas de la arquitectura de la aplicación en tres partes: Modelo, Vista y Controlador. La figura 1 muestra la arquitectura MVC como se aplica a aplicaciones Web.

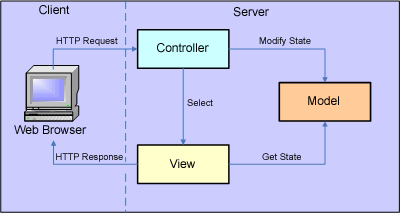


Figura 1: Arquitectura MVC para aplicaciones Web

Modelo  
El modelo representa el estado de la aplicación y define las acciones comerciales que lo modifican (datos persistentes y la lógica de negocios). Puede ser consultado acerca de su estado (por lo general por la vista) y se le pedirá que lo cambie (por lo general por el controlador). No sabe nada acerca de la vista o al controlador.

Vista  
La vista proporciona la presentación del Modelo. Se representa el aspecto de la aplicación, es decir, su interfaz de usuario. Es responsable de la entrada y salida de datos hacia y desde el usuario. La vista puede obtener el modelo de estado, pero no puede modificarlo.

Controlador  
El controlador reacciona a la entrada del usuario e informa al Modelo para cambiar su estado en consecuencia. Específicamente, procesa las solicitudes de entrada de usuario mediante el envío de ellos a las funciones de lógica de negocio apropiados (en el modelo) y la selección de la respuesta al usuario (la vista) basándose en el resultado.

Beneficios  
El patrón de diseño MVC separación de la lógica de negocio a partir de los resultados de la presentación de los siguientes beneficios:

* **Mantenibilidad mejorada**  
  Debido a la Vista y el Modelo de capas están desconectados, puede cambiar la interfaz de usuario sin afectar a las reglas de negocio y vice-versa. El impacto de los cambios es, por lo tanto, reduce al mínimo.
* **Modelo de Reutilización**  
  Se pueden crear múltiples vistas del mismo modelo. Por ejemplo, si su aplicación debe ser compatible con diferentes tipos de dispositivos de cliente (por ejemplo, teléfonos celulares y PDAs), puede crear nuevos puntos de vista específicos para cada tecnología y volver a utilizar el mismo modelo.
* **Separación de Responsabilidades**  
  Funciones de desarrollo pueden ser separados permitiendo que los diferentes miembros del equipo de desarrollo para centrarse en su área de especialización. Diseñadores de páginas web, por ejemplo, puede ser responsable de la capa de la Vista y trabajar de forma independiente de los desarrolladores de Java que se pueden concentrar en la aplicación del controlador y las capas del modelo.

JAVA ECLIPSE EE

# SQL Server 2008