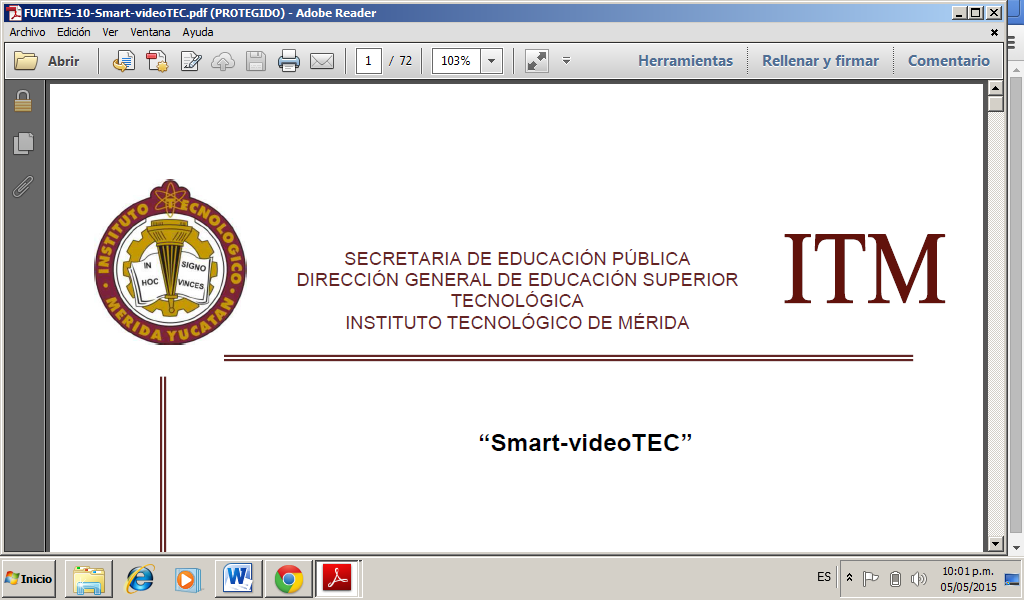
**TECNOLÓGICO NACIONAL DE MEXICO**

**INSTITUTO TECNOLÓGICO DE MÉRIDA**

**INGENIERIA EN SISTEMAS COMPUTACIONALES**

**Desarrollo FrontEnd**

[***MVC***](http://tecvi.itmerida.mx/mod/resource/view.php?id=76287)

**ALUMNA:**

**Aldo Gaspar Alcala Morales**

**PROFESOR:**

**Héctor Cetina**

**9SA**

**Fecha de Entrega: 8/octubre/2019**

Los frameworks web son un conjunto de herramientas, estilos y librerías dispuestas a través de una estructura o esqueleto base, para el desarrollo de aplicaciones web más escalables y sencillas de mantener.

Ventajas:

1. Documentación y comunidad.

La cantidad de documentación que podremos encontrar sobre un framework web, suele ser enorme y además con una gran comunidad detrás, respondiendo preguntas y desarrollando nuevas funcionalidades.

1. Reutilización del código.

Uno de los puntos fuertes de los frameworks web es el modularidad de su código y la capacidad para poder hacer múltiples proyectos con el mismo código, cambiando simplemente los textos.

1. Arquitectura y metodología.

La mayoría de frameworks web del mercado usan arquitecturas y metodologías actuales, como el [Modelo-Vista-Controlador](https://es.wikipedia.org/wiki/Modelo%E2%80%93vista%E2%80%93controlador).

1. Plantillas web.

Las plantillas facilitan mucho el trabajo de los desarrolladores web y los frameworks no se quedan atrás en esto. Algunos frameworks Frontend como [Bootstrap](https://getbootstrap.com/) cuentan con grandes cantidades de plantillas y componentes desarrollados por su extensa comunidad.

1. Seguridad web.

Los frameworks web suelen contar con medidas de seguridad para proteger nuestros datos y los de nuestros clientes, ayudando en gran medida en uno de los temas que lleva de cabeza a grandes empresas de servicios web desde el 2017.

1. Posicionamiento en motores de búsqueda.

El [posicionamiento web SEO](https://www.illusionstudio.es/servicios/posicionamiento-natural-en-buscadores-seo) on page es muy importante si queremos lograr aparecer en las primeras posiciones de buscadores como Google. Por eso muchos frameworks web ya implementan en su estructura código para poder lograrlo más fácilmente.

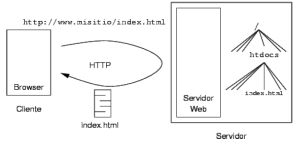
1. Ofertas de empleo.

Si una persona llega a aprender a usar un framework web de manera solvente, esa persona tendrá muchas posibilidades de encontrar un trabajo más fácilmente y con un sueldo mayor.

La arquitectura de un Sitio Web tiene tres componentes principales:

* Un servidor Web
* Una conexión de red
* Uno o más clientes

El servidor Web distribuye páginas de información formateada a los clientes que las solicitan. Los requerimientos son hechos a través de una conexión de red, y para ello se usa el protocolo HTTP. Una vez que se solicita esta petición mediante el protocolo HTTP y la recibe el servidor Web, éste localiza la página Web en su sistema de archivos y la envía de vuelta al navegador que la solicitó.

[](https://programacionwebisc.files.wordpress.com/2013/11/unidad2.png)

* CGI

Common Gateway Interface fue la primera técnica utilizada para que el contenido de las páginas web se generará de manera dinámica, es común encontrar en los diferentes servidores web el módulo que soporta la ejecución de  
CGIs.

De manera resumida se puede decir que el CGI es un mecanismo de comunicación entre el servidor web y una aplicación externa, esta aplicación puede estar desarrollada en casi cualquier lenguaje, este solo debe cumplir la condición de ser soportado por el servidor http, es común encontrar que la mayoría de las aplicaciones CGIs se encuentren desarrolladas con el lenguaje PERL.

Este mecanismo tiene deficiencias que evita su uso a gran escala, la más conocida es en cuanto a rendimiento, ya que por cada petición que se realice en el servidor se crea un nuevo proceso, lo cual tiene un costo muy alto en lo que a recursos del sistema se refiere.

* JSF

JSF es un framework MVC (Modelo-Vista-Controlador) basado en el API de Servlets que proporciona un conjunto de componentes en forma de etiquetas definidas en páginas XHTML mediante el framework Facelets. Facelets se define en la especificación 2 de JSF como un elemento fundamental de JSF que proporciona características de plantillas y de creación de componentes compuestos. Antes de la especificación actual se utilizaba JSP para componer las páginas JSF.

JSF utiliza las páginas Facelets como vista, objetos Javabean como modelos y métodos de esos objetos como controladores. El servlet FacesServlet realiza toda la tediosa tarea de procesar las peticiones HTTP, obtener los datos de entrada, validarlos y convertirlos, colocarlos en los objetos del modelo, invocar las acciones del controlador y renderizar la respuesta utilizando el árbol de componentes.

Entrando un poco más en detalle, JSF proporciona las siguientes características destacables:

* Definición de las interfaces de usuario mediante vistas que agrupan componentes gráficos.
* Conexión de los componentes gráficos con los datos de la aplicación mediante los denominados beans gestionados.
* Conversión de datos y validación automática de la entrada del usuario.
* Navegación entre vistas.
* Internacionalización
* A partir de la especificación 2.0 un modelo estándar de comunicación Ajax entre la vista y el servidor.
* HTML

El archivo HTML estático puede contener referencias a portlets, contenedores y navegación. Define los lugares de la página del portal que aloja portlets y contenedores de portlets. Cuando se representa la página, el servidor llena estos lugares con el contenido (posiblemente dinámico) del portlet y con un microformato que define los metadatos de estos portlets. Por ejemplo, las acciones de portlet y el título de portlet. Con este objetivo, el portal proporciona los siguientes microformatos:

* El portlet microformat (microformato del portlet) define las ventanas y acciones del portlet, como Editar valores predeterminados, Configurar, Maximizar, Minimizar, Personalizar y Suprimir.
* El container microformat (microformato de contenedor) define los contenedores de portlet como marcadores de los portlets.
* Para las acciones de arrastrar y soltar, el microformato del portlet puede proporcionar el origen de arrastrar y el microformato del contenedor proporciona el destino de soltar.
* El navigation microformat (microformato de navegación) define la navegación si la página estática se representa como página web.

Los microformatos de la ventana del Portet y el contenedor del portlet pueden contener ID de objeto. El servidor puede gestionar estos ID de objeto de manera dinámica.

Para escribir la página estática, puede utilizar técnicas de JavaScript o CSS que utilizan microformatos para producir y representar una interfaz de usuario fácil de utilizar

MVC es una propuesta de diseño de software utilizada para implementar sistemas donde se requiere el uso de interfaces de usuario. Surge de la necesidad de crear software más robusto con un ciclo de vida más adecuado, donde se potencie la facilidad de mantenimiento, reutilización del código y la separación de conceptos.

Su fundamento es la separación del código en tres capas diferentes, acotadas por su responsabilidad, en lo que se llaman Modelos, Vistas y Controladores, o lo que es lo mismo, *Model, Views & Controllers*, si lo prefieres en inglés

* Ventajas:
* Separación clara de dónde tiene que ir cada tipo de lógica, facilitando el mantenimiento y la escalabilidad de nuestra aplicación.
* Sencillez para crear distintas representaciones de los mismos datos.
* Facilidad para la realización de pruebas unitarias de los componentes, así como de aplicar desarrollo guiado por pruebas (Test Driven Development o TDD).
* Reutilización de los componentes.
* No hay ViewState ni ciclo de vida de las páginas. Con menos peso, menos complejidad.
* Motor de Routing asociando una URL concreta con su correspondiente controlador, permitiendo URL semánticas (es decir que tengan la forma *«http://servidor/catalogos/sandalias»* en lugar de *«http://servidor/catalogos/articulos.aspx?familia=sandalias»*). Las URL semánticas se indexan mejor en los buscadores, siendo más adecuadas para el posicionamiento web.
* Modelo: Representa las reglas de negocio de nuestra aplicación. Se encarga de mantener la persistencia de los datos, guardando o recuperando la información independiente de medio utilizado (ficheros XML, bases de datos, etc…).
* Vista: Representa los componentes que muestran la interfaz de la aplicación, mostrando la información obtenida a partir del modelo, de manera que el usuario pueda visualizarla. Básicamente las vistas contienen el código de presentación que se va a enviar al navegador.
* Controlador: Representa los componentes que se encargan de la interacción del usuario, actuando de intermediario entre el usuario, el modelo y la vista. El controlador recoge las peticiones del usuario, interacciona con el modelo y finalmente selecciona que vista es la adecuada para mostrar los datos en cuestión.



1. **El usuario** realiza una solicitud a nuestro sitio web. Generalmente estará desencadenada por acceder a una página de nuestro sitio. Esa solicitud le llega al controlador.
2. **El controlador** comunica tanto con modelos como con vistas. A los modelos les solicita datos o les manda realizar actualizaciones de los datos. A las vistas les solicita la salida correspondiente, una vez se hayan realizado las operaciones pertinentes según la lógica del negocio.
3. **Para producir la salida**, en ocasiones las vistas pueden solicitar más información a los modelos. En ocasiones, el controlador será el responsable de solicitar todos los datos a los modelos y de enviarlos a las vistas, haciendo de puente entre unos y otros. Sería corriente tanto una cosa como la otra, todo depende de nuestra implementación; por eso esa flecha la hemos coloreado de otro color.
4. **Las vistas** envían al usuario la salida. Aunque en ocasiones esa salida puede ir de vuelta al controlador y sería éste el que hace el envío al cliente, por eso he puesto la flecha en otro color.

**Menciona cuales son las tecnologías (3 frameworks) que usan este tipo de arquitectura para el desarrollo de aplicaciones web y realiza una tabla comparativa con sus características.**

|  |  |
| --- | --- |
| **Framework** | **Caracteristicas** |
| **CodeIgniter** | * Utiliza un framework ligero, hecho pensando en el rendimiento. * Comienza rápidamente, gracias a la simplicidad del framework y la excelente documentación. * Crea aplicaciones escalables utilizando la arquitectura basada en MVC. |
| **Zend** | * Utiliza un framework PHP orientado a objetos con una arquitectura MVC. * Reutiliza tu código gracias al diseño de la plataforma. * Integra Zend con bibliotecas externas fácilmente. * Usa solo los componentes que desees e ignora todo lo demás. |
| **Phalcon** | * Usa un framework PHP basado en C. * Aprovecha el fantástico rendimiento de Phalcon y la reduce sobrecarga de recursos. * Utiliza solo los módulos y bibliotecas que necesitas. |

**REFERENCIAS BIBLIOGRAFICAS**

<https://www.illusionstudio.es/que-es-un-framework-web>

<https://programacionwebisc.wordpress.com/2-1-arquitectura-de-las-aplicaciones-web/>

<https://edgarbc.wordpress.com/tecnologias-para-el-desarrollo-de-aplicaciones-web/>

<http://www.jtech.ua.es/j2ee/publico/jsf-2012-13/sesion01-apuntes.html>

<https://www.ibm.com/support/knowledgecenter/es/SSYJ99_8.5.0/dev/spa_define_page.html>

<https://desarrolloweb.com/articulos/que-es-mvc.html>

<https://www.clavei.es/blog/que-aporta-mvc-al-desarrollo-de-aplicaciones-web/>