|  |  |  |  |
| --- | --- | --- | --- |
| **CARRERA** | **SEMESTRE** | **CÓDIGO MATERIA** | **NOMBRE DE LA MATERIA** |
| Ingeniería Informática | 8º | 805 | Programación Web |

|  |  |  |
| --- | --- | --- |
| **PRÁCTICA N.ª:** | **TEMA:** | JSP, Servlet y Operaciones SQL |
| 1  2  3  4 | **NOMBRES:** | Bryan Acosta |
| **FECHA ENTREGA:** | 08 de noviembre del 2018 |

|  |  |
| --- | --- |
| **1.** | **INTRODUCCIÓN** |
| La plataforma de Java EE es un conjunto de especificaciones en el API que te permiten construir aplicaciones web. Un contenedor web se encarga de implementar todas las especificaciones del API, además de proveer servicios, para el manejo y ejecución de componente web, tales como servlets, JSPs, etc.  Java EE provee un perfil web y modelo de aplicación estandarizado que permite definir una arquitectura para la implementación de aplicaciones multinivel. Un perfil web permite a los desarrolladores crear aplicaciones web con un conjunto de tecnologías bien definidas. En las aplicaciones multinivel, la funcionalidad de la aplicación es separada dentro de diversas áreas conocidas como niveles o capas.  Uno de estos esquemas, dividido en capas es la arquitectura MVC (Modelo Vista Controlador), la cual proporciona una clara separación entre las distintas responsabilidades de la aplicación.  **EL CONTROLADOR**  Se puede decir que el controlador es el “cerebro” de la aplicación. Todas las peticiones a la capa intermedia que se realicen desde el cliente son dirigidas al controlador, cuya misión es determinar las acciones a realizar para cada una de estas peticiones e invocar al resto de los componentes de la aplicación (modelo y la vista) para que realicen las acciones requeridas en cada caso, encargándose también de la coordinación de todo el proceso. Suponiendo, que una petición requiera enviar como respuesta al cliente determinada información existente en una BD, el Controlador solicitara los datos necesarios al modelo y, una vez recibidos, se los proporcionara a la vista para que esta les aplique el formato de presentación correspondiente y envié la respuesta al cliente.  En aplicaciones JavaEE el controlador es implementado mediante un **servlet** central que, dependiendo de la cantidad de tipos de peticiones que debe gestionar, puede apoyarse en otro servlets auxiliares para procesar cada petición.  **LA VISTA**  Es la encargada de generar las respuestas (habitualmente XHTML) que deber ser enviadas al cliente. Cuando esta respuesta tiene que incluir datos proporcionados por el Controlador, el código XHTML de la página no será fijo si no que deberá ser generados de forma dinámica, por lo que su implementación correrá a cargo de una página JSP.  Las páginas JSP resultan mucho mas adecuadas para la generación de las vistas que los servlets pues, al ser documentos de texto, resulta sencilla la inclusión de bloques estáticos XHTML.  Cuando la información que se va a enviar es estática, es decir, no depende de datos extraídos de un almacenamiento externo, podrá ser implementada por una página o documento XHTML.  **EL MODELO**  En la arquitectura MVC la lógica de negocio de la aplicación, incluyendo el acceso a los datos y su manipulación, esta encapsulada dentro del modelo. El Modelo lo forman una serie de componentes de negocio independientes del Controlador y la Vista, permitiendo así su reutilización y el desacoplamiento entre las capas.  En una aplicación JavaEE el modelo puede ser implementado mediante clases estándar Java o a través de Enterprise JavaBeans. (Armando, 2015) | |

|  |  |
| --- | --- |
| **2.** | **LOGRO DE APRENDIZAJE** |
| * Comprender el funcionamiento de una aplicación implementada mediante un modelo MVC. * Conocer que tipo de archivos componen cada capa del modelo , de la vista y del controlado * Creación de aplicaciones Web multiplataforma que brinden servicios a clientes, estas aplicaciones no necesitaran ser instaladas, únicamente se necesita un navegador para acceder a ellas. | |

|  |  |
| --- | --- |
| **3.** | **MARCO TEÓRICO** |
| **FUNCIONAMIENTO DE UNA APLICACIÓN MVC**  Una vez analizados los distintos bloques MVC resulta sencillo comprender el funcionamiento de este tipo de aplicaciones. Para ello, analizaremos los procesos que tienen lugar en la capa intermedia desde que llega la petición procedente de la capa cliente hasta que se genera la respuesta:   * **Captura de la petición en el controlador**. Como hemos dicho, todas las peticiones que se reciben en la aplicación son centralizados en el Controlador, el cual a partir de la URL de la solicitud determina el tipo de la operación que quiere llevar a cabo el cliente. Normalmente, esto se hace analizando el valor de algún parámetro que se envía anexado a la URL de la petición y que se utiliza con esta finalidad:   *url?operacion=validar*   * Procesamiento de la petición. Una vez que el Controlador determina la operación a realizar, procede a ejecutar las acciones pertinentes, invocando para ellos a los diferentes métodos expuestos por el Modelo.   Dependiendo de las acciones a realizar, el Modelo necesitara manejar los datos enviados por el cliente en la petición, datos que le serán proporcionados por el Controlador. De la misma manera, los resultados generados por el Modelo serán entregados directamente al Controlador.   * Generación de respuestas. Los resultados devueltos por el Modelo al Controlador son depositados por éste en una variable de petición, sesión o aplicación, según el alcance que deban tener. A continuación, el Controlador invoca a la pagina JSP que debe encargarse de generar la vista correspondiente, esta página accederá a la variable de ámbito donde estén depositados los resultados y los utilizará para generar dinámicamente la respuesta XHTML que serán enviada al cliente. (Martin, 2007) | |

|  |  |  |
| --- | --- | --- |
| **4.** | **PROCEDIMIENTO (DESCRIPCIÓN)** | |
| **A** | **HARDWARE REQUERIDO** | **SOFTWARE REQUERIDO** |
|  | * Procesador x86-x64 o AMD 2.5GHz + * RAM 4GB+(mínimo) * GPU 128MB+ | * Windows 7 o superior * MySQL * Web Browser: Google Chrome 30+, Firefox 24+, Opera 20+, Edge, IE12+ * NetBeans IDE 8.1 * Glassfish |
| **B** | **DESARROLLO DE LA PRÁCTICA** | |
|  | **Práctica 1:** **JSP utilizando una clase Java para realizar la suma de dos números.**   1. Creación de un nuevo proyecto siendo el tipo de proyecto Web Application. En el caso de tener correctamente instalado el GlassFish, automáticamente escogeremos el que se encuentra por default. 2. Se creará la siguiente estructura de archivos.        1. Y como parte del “Modelo” tendremos una clase Java *BeanOperacion.java.* donde tendremos los métodos get-set de los dos números.      1. Como parte de la Vista se creara un archivo .jsp de nombre *jspSuma2,*  en el cual se hace uso de etiquetas html para darle diseño a la página.     Como parte importante es en este archivo donde llamaremos a nuestra clase *BeanOperacion.java* para hacer uso de los métodos creados y de sus atributos    **Práctica 2: JSP, Servlets y Operaciones SQL.**   1. Creación de una base de datos en Mysql llamada ORCL con la siguiente tabla con sus respectivos atributos. (Anexo 1 ) 2. Creación de un nuevo proyecto Web Application con la siguiente estructura.     Al realizar una interacción con la base de datos es necesario añadir el Driver que nos permite la conexión entre nuestra aplicación y la BD.   1. Dentro del paquete clases, se creará una clase *clase\_persona2.java*  la cual contendrá un método para establecer la conexión con la BD *procedimiento\_conexion ()*, en el cual una variable *var\_conexion* es la que identifica toda la información de la BD a la que se conectará.     Se creará otro método para realizar una consulta *procedimiento\_consultar()* el cual incluirá la consulta que permite extraer los datos de la BD.      y otro que permita mostrar la información solicitada en cuadros de texto *procedimiento\_mostrar\_consulta().*   1. Dentro de páginas Web, se creará un archivo JSP llamado *consultaDB.jsp* el cual principalmente contiene las líneas de interacción con el modelo y un cuadro de texto que mostrará la cedula de la cual será ira hacer la búsqueda.     **Práctica 3: JSP, Servlets y Operaciones SQL.**  Desarrollo de la práctica:   1. Desarrollamos una base de datos a la cual llamaremos orcl que tendrá una tabla llamada persona en la cual tendrá los atributos cedula, nombre, apellido, sexo. 2. Dentro del proyecto importar la librería del motor de base de datos MySQL. 3. Creamos un nuevo proyecto en NetBeans y escogemos un proyecto de tipo Web Application. 4. Creamos un paquete a cuál llamaremos paquete\_clases y dentro de este paquete creamos una clase java a cual llamaremos Clase\_Persona1. 5. Dentro de la clase java Clase\_Persona1    1. Realizamos herencia de la clase Object    2. Declaramos un arreglo que nos permita identificar los campos de la tabla persona, de igual forma creamos los metodos get y set de este arreglo.    3. Declaramos variables de tipo SQL que nos permita establecer la conexión a través de los métodos respectivos a la base de datos orcl.    4. Declaramos un método que nos permita realizar una sentencia SQL de inserción a la tabla persona de la base de datos orcl. 6. Dentro del paquete Web Pages agregamos un nuevo JSP al cual llamaremos ingresoDB. 7. Dentro de este JSP desarrollamos la vista con los respectivos componentes para el ingreso de los datos de la persona, de igual forma llamamos a los métodos de nuestra clase Clase\_Persona2 para la ejecución de la sentencia SQL de inserción. 8. Ejecutamos nuestra aplicación y verificamos su buen funcionamiento.   **Práctica 4: JSP, y sus componentes básicos**   1. Se hace uso de la mismo BD ocupada para la práctica anterior. 2. Creación de un nuevo proyecto Web Application, con la siguiente estructura.     La interacción con la BD nos obliga a incluir el driver que nos permita dicha conexión.   1. En la creación de la clase *Clase\_Persona1.java,* que se encuentra dentro del paquete *paquete\_clases* se creará un arreglo , el cual permitirá la captura de los datos a ingresar.   Tendremos una método *procedimiento\_conexion()* para la establecer la conexión de la BD, mostrando todos los parámetros para poder realizar dicho procedimiento.  Finalmente se tendrá un método *procedimiento\_ingreso()* el cual permite el ingreso de los datos a la BD.     1. En la vista, la cual representa el archivo *ingresoDB.jsp*, se crean los cuadros de texto para cada requerimiento y adicionalmente se hace una validación en el caso de que se ingresen valores nulos. | |
|  | **RESULTADOS DE LA PRÁCTICA** | |
|  | **Practica 1:**    Ilustración 1. Suma de datos  **Practica 2:**    Ilustración 1. Consulta de datos  **Practica 3:**    Ilustración 1. Ingreso de datos  **Practica 4:**    Ilustración 1. Ingreso de datos | |

|  |  |
| --- | --- |
| **5.** | **CONCLUSIONES Y RECOMENDACIONES** |
| ***Conclusiones:***   * Sin tomar en cuenta ninguna arquitectura para el diseño de una aplicación web se la puede exponer sin error alguno. * Las aplicaciones Web, aportan interoperabilidad entre aplicaciones de software independientemente de sus propiedades o de las plataformas sobre las que se ejecuten. * Este tipo de servicios no requiere software adicional, únicamente necesita un navegador para ser usada.   ***Recomendaciones:***   * Tomar en cuenta los nombres que se le establecen a todos los términos de la BD para no tener inconvenientes al momento de su conexión. * Encapsular los atributos que van a ser almacenados en los Beans, para que no sean modificados y cumplir los estándares de programación. | |

|  |  |
| --- | --- |
| **6.** | **ANEXOS** |
| Se adjuntará un CD con el proyecto generado para la correspondiente revisión por parte del Ingeniero. | |

|  |  |
| --- | --- |
| **7.** | **REFERENCIAS** |

# Bibliografía

Armando, P. (20 de 01 de 2015). *Tópicos en Java Mx*. Obtenido de Introducción a la Arquitectura de JavaEE: http://javaconfigmx.blogspot.com/2015/01/introduccion-la-arquitectura-de-java-ee.html

Martin, A. J. (2007). *Struts.* Madrid: RA-MA.

|  |  |
| --- | --- |
| **DICTADO POR:** | Julio César Mendoza Tello |