**HOJA INTERACTIVA DE APRENDIZAJE Nº 09**

**Apellidos y Nombres: Villegas Siguas iori**

**Docente:** Rony Hancco Carpio **Fecha de entrega:** 09/06/2021

**MVC**

1. **Indicadores:**
   1. Identifica uso de capas en una aplicación backend.
   2. Identifica el uso de inyección de dependencias.
   3. Crea aplicaciones usando las capas Controlador, servicio y DAO.
   4. Prueba los servicios Rest creados.
2. **Actividades para desarrollar en forma individual:** 
   1. Complete la tabla sobre la sintaxis Java:

|  |  |
| --- | --- |
| **Código fuente** | **Errores** |
| @RestController public class Controlador {  @Autowired  private Servicio servicio;  @ResponseMapping(value = "/obtener-productos",  method = RequestMethod.*POST*,  produces = "application/json;charset=utf-8"  )  public @RequestBody RespuestaProducto obtenerProductos(){  RespuestaProducto respuestaProducto = new RespuestaProducto();  respuestaProducto.setLista(servicio.obtenerProductos());  return respuestaProducto;  } } | La parte de private en el servicio es el primer error que sale a las vista del código y poner serlista porque ya respuestapodructo es una lista que contiene a los productos. |
| public class DaoImpl implements Dao{    private JdbcTemplate jdbcTemplate;  private Connection conexion;  } | Lo que le falta conexion.jdbcTemplate porque qui solo a creado los objetos de JdbcTemplate y connection. |
| public class ServicioImpl implements Servicio{  @Autowired  private Dao dao;  } | Lo que no está bien en la sintaxis es que solo debería decir Dao dao el private está de más. |

* 1. Complete la tabla escribiendo los conceptos aprendidos en clase

|  |
| --- |
| **Conceptos** |
| **Patrones de diseño estructural**  Los patronesestructurales explican cómo ensamblar objetos y clases en estructuras más grandes. |
| **Decorator**  Es un patrón de diseño estructural que usa la agregación como sustituto de herencia pura. |
| **Facade**  Es un tipo de modelo de diseño estructural. Inspirado para reducir la completitividad y proporciona una interfaz de una biblioteca o más bibliotecas de forma simplificada. |
| **Adapter**  Permite la colaboración entre objetos con interfaces incompatibles. |
| **Bridge**  Permite separar las clases extensas mediante la abstracción e implementación permitiendo desarrollar de forma independiente. |
| **MVC** (Modelo-Vista-Controlador) es un patrón en el diseño de software para implementar interfaces ya sea de usuario, datos y lógica de control. |
| **Controlador**  El controlador permite recibir las peticiones HTTP y responderlas. Para el proyecto usaremos peticiones y respuestas en formato JSON. |
| **DAO:**  Es un componente que abastece un interfaz común entre la aplicación y uno o más dispositivos de almacenamiento de datos, en pocas palabras se encarga de realizar las operaciones con base de datos. |
| **Inyección de dependencias:**  Es el proceso de llamar a elementos o funcionalidades de librerías para que os facilite o no apoye en el procesamiento de los códigos. |

* 1. Complete la tabla.

|  |
| --- |
| **Código fuente: Cree una consulta SQL que contemple actualice datos de un registro de tabla en una Clase DaoImpl.** |
| UPDATE nombre = ”Joaquin” WHERE id = “1”; |

* 1. Suba el documento resuelto y la solución del caso de estudio en un directorio llamado lab-9.

1. **Referencia de material** 
   1. Alvarez, Cecilio (2016). ¿Qué es Spring Boot?. Recuperado de <https://www.arquitecturajava.com/que-es-spring-boot/>
   2. Spring (2018). Building a RESTful Web Service. Recuperado de <https://spring.io/guides/gs/rest-service/>

1. **Conclusiones y recomendaciones**

* Se recomienda que se entienda el proceso de las capas y funcionalidades del proceso adecuado para llena y que todo este funcional.
* Es indispensable retroalimenta la información bridada y saber las definiciones so funcionalidades de las dependencias llamadas.
* Además, debes saber de qué librerías se van a utilizar para poder importarlo y llamarlos**.**