**OBSERVACIONES IMPORTANTE ANTES DE INICIAR:**

Cada trabajo o actividad académica que sea desarrollado por usted como evidencia de su proceso de aprendizaje, debe estar 100% libre de frases copiadas desde cualquier fuente original (documento, libro, página web, proyecto de código fuente, video tutorial, etc) de información.

En ningún momento se le está pidiendo que haga las cosas de 0 y olvide por completo los conocimientos sobre los cuales se basa algún tema de esta asignatura, por el contrario, lo estamos animando a que lea, investigue, comprenda, deduzca, analice, sintetice, concluya, interprete, argumente, domine, practique, critique, niegue o afirme con bases sólidas basadas en la lectura y el estudio autónomo y autodidacta.

Tenga muy presente que mientras estudias esta profesión y durante toda tu vida, siempre seras evaluado y medido por un docente, por un instructor, por tu equipo de trabajo, por tus jefes, por la sociedad o por tus clientes.

Antes de realizar cualquier acción indebida en el desarrollo de esta y cualquier otra actividad académica, recuerda que actualmente existen muchas herramientas modernas y avanzadas que existen para analizar, buscar, identificar, rastrear y comparar la idoneidad y originalidad de cada contenidos digital (texto, imágenes, archivos, etc), con especial profundidad en archivos de documentos académico o que contienen fuentes de programas informáticos, las cuales permiten detectar plagio de una manera tremendamente simple, rápida y automatizada.

Evita que el pensamiento del mínimo esfuerzo y el facilismo invada tu mente y tus acciones, ya que este destruirá tus sueños de crecimiento y éxito profesional.

**OBSERVACIONES**:

##### **1.** **ACTIVIDAD INDIVIDUAL**

1. CADA ALUMNO (SIN EXCEPCIÓN ALGUNA) DEBE SUBIR LA ACTIVIDAD DESDE SU RESPECTIVA CUENTA EN SIMA-PESAD
2. SOLO SE EVALUARÁ Y CALIFICARÁ A LOS ESTUDIANTES QUE HAYAN SUBIDO LA ACTIVIDAD, LOS ALUMNOS QUE NO ENTREGUEN LA ACTIVIDAD SERÁN EVALUADOS CON NOTA 1.0

**FORMA Y FORMATO DE ENTREGA DE LA ACTIVIDAD:**

Archivo comprimido en **.ZIP** (No se calificarán archivos en otro formato)

Nombre del archivo: **actividad\_2\_nombre\_apellido\_devweb\_2023-1.ZIP**

Contenido del archivo comprimido:

* Un archivo comprimido con la carpeta que contiene el código fuente del ejercicio y el código de la base de datos base\_de\_datos.sql
* Documentos de texto en formato DOCX u ODT (no se aplicarán en otro formato) :
* Nombre del archivo: **actividad\_2\_nombre\_apellido\_devweb\_2023-1.DOCX**
* El desarrollo y entrega del documento debe contener todos los ítem típicos y característicos de un trabajo académico.

1. Presentación
2. Enlace del repositorio de GitHub donde se puede ver y clonar el proyecto.
3. Tabla de contenido
4. Introducción
5. Objetivos
6. Justificación
7. Desarrollo
   1. Colocar fragmentos de pantallazos importantes del código, donde se debe explicar el objetivo de dicho código, de acuerdo al nivel de detalle y calidad del desarrollo de esta sección, así mismo será la calificación.
8. Síntesis y argumentación
9. Problemas encontrado y sus respectivas soliciones
10. Bibliografía

**Normas APA para la elaboración de esta actividad**

<https://www.colconectada.com/normas-apa/>

**DESARROLLO DE LA ACTIVIDAD :**

ACTIVIDAD UNIDAD 2:

De acuerdo a lo estipulado y acordado en tutorías, esta actividad tiene como fin realizar un ejemplo simple como desarrollar una aplicación Web usando el Patrón de diseño MCV, para lo cual el docente ha suministrado abundantes ejemplo y material complementario que profundizan mucho más lo poco que se puede explicar en tutorías.

**INSTRUCCIONES**:

1. **Realizar la BD** del ejercicio asignado
2. Construir el paquete **MODELO** (Carpeta con Clases entidades que mapean a las tablas). co.edu.udec.devweb.activdad2.tu\_nombre\_apellido.**modelo**
3. Crear el paquete entidades co.edu.udec.devweb.activdad2.tu\_nombre\_apellido.**modelo.entidades**
4. Dentro del paquete entidades crear una Clase Entidad por cada Tabla de la BD
5. Crear el paquete dato co.edu.udec.devweb.activdad2.tu\_nombre\_apellido.**modelo.dao**
6. Crear un clase Dato por cada Clase Entidad (las que realizan las operaciones de CRUD de los objetos Entidades sobre la BD), por ejemplo EntidadXyzCrud
   1. método void **agregar**(EntidadXyz e) throws Exception
   2. método EntidadXyx **buscarPorId**( Integer Id) throws Exception
   3. método void **editar**(EntidadXyz e) throws Exception
   4. método void **eliminarPorId**(Integer id) throws Exception
   5. método List<EntidadXyz> **listarTodo**()
   6. método List<EntidadXyz> **buscarPorPropiedadX**( Object propiedadX)
   7. método int **contar**()
7. Construir la capa de **VISTA,** una carpeta que contiene una subcarpeta por cada clase entidad del paquete modelo. Es recomendable que cada subcarpeta se llame igual o parecida a su respectiva clase de la capa entidades. Dentro de cada subcarpeta debe existir los archivos JSP con la logica necesaria para permitir al usuario interactuar con las operaciones CRUD correspondientes a dicha entidad, por ejemplo::
   1. agregar.jsp
   2. buscar.jsp
   3. editar.jsp
   4. eliminar.jsp
   5. listar\_todo.jsp
   6. listar\_con\_filtro.jsp
8. Para la clase **entidad Usuario** (la que mapea la tabla Usuarios) se debe agregar además:
   1. Un archivo llamado **login**.jsp, el cual será la Vista para hacer login
   2. Un archivo llamado **recoverypass**.jsp el cual debe permitir al usuario ver y cambiar la contraseña (ideas es que la puedan enviar al correo electrónico del usuario) en caso de olvido.
9. En la capa de Vista debe existir un archivo **index**.html en cada subcarpeta
10. Debe existir también el siguiente conjunto de subcarpetas:
    1. Una carpeta **css** para guardar los archivos con código CS
    2. Una carpeta **js** para guardar los archivos con código JavaScripr
    3. Una carpeta **img** para guardar los archivos con imágenes
    4. Una carpeta **medios** para guardar los archivos multimedia (videos y audio)
11. Crear carpeta **controladores,** en la cual debemos colocar un archivo por cada Clase Entidad en el paquete **modelo.entidad**. Cada uno de estos archivos debe contener una clase Java de tipo Servlet, la cual de acuerdo a la logica de cada ejercicio, esta clase debe tener métodos con las siguiente lógica:
    1. Capturar o recuperar los parámetros enviados por el usuario mediante (petición POST generalmente) o por medio de un enlace (petición GET generalmente) el uso de los formularios que contiene las páginas web de la capa Vista.
    2. Validar el formato de los datos enviados por la Vista
    3. Invocar los métodos de la clase EntidadXyxCrud
       1. **agregarEntidadXyx()**: Recupera los parámetros enviados por la petición POST, crea un objeto EntidadXyz, lo guarda en la BD (usando el método **)**de la clase EntidadXyzCrud) y luego redirecciona a la vista **agregar**.jsp informando que ha sido guardado y cuantos objetos EntidadXyz existen actualmente en la bd.
       2. **buscarEntidadXyx()**: Usa el parámetro Id enviado por la petición GET y realiza una búsqueda del objeto EntidadXyz en la Bd (usando el método **buscarPorId** de la clase EntidadXyzCrud), si lo encuentra, entonces lo guarda como un objeto de la sesión y redirecciona a la vista **buscar**.jsp, esta última, recupera el objeto de la sesión muestra los datos en el formulario.
       3. **editarEntidadXyx()**: Toma el objeto entidad encontrado y guardado en sesión (metodo **buscarEntidadXyx**) para cambiar o modificar sus datos, tomando como entrada los nuevos datos enviados por la petición POST, luego guarda los cambios en la BD haciendo uso del método **editar** de la clase EntidadXyzCrud. Por último, redirecciona a la página buscar.jsp informando sobre el resultado del proceso.
       4. **eliminarEntidadXyx()**: Toma el objeto entidadesXyz encontrado y guardado en sesión, compara su ID con el ID enviado por la petición mediante la petición GET, si son iguales entonces, lo elimina de la BD (usando el método **eliminar** de la clase EntidadXyzCrud), borra el objeto entidad guardado en la sesión y por último redirecciona a la vista **eliminar**.jsp informando cuanto objetos de tipo EntidadesXyz existem actualmente ne la BD
       5. **listarEntidadXyx()**: Atiende la petición GET realizada desde el navegador, realiza una consulta de objetos EntidadesXyz en la BD (usando el método **listar\_todo**() de la clase EntidadXyzCrud, guardando el resultado en una lista y esta asu vez, es guardada como una variable de la sesión HTTP. Por último redirecciona al archivo **listar\_todo**.jsp, la cual recupera la lista de objetos desde la Sesión HTTP y muestra los datos en una tabla HTML de forma dinámica.
       6. **buscarEntidadXyxPorFiltro():** Actua similar al método **listarEntidadXyx,** con la diferencia que este método recupera parámetros enviados por peticion GET desde el navegador y los toma como insumo para realizar una consultas de objetos EntidadesXyz sobre la BD
       7. Para el caso de la clase **ControladorUsuario**.jsp, esta además de los métodos anteriores también debe tener los siguientes métodos.
          1. **iniciar\_sesion():** Recupera los parámetros (email y password) enviados por una petición POST, luego consulta si existe un usuario en la BD con ese email o id (usando el método buscarPorId de la clase EntidadXyzCrud), si existe entonces, compara el password con el parámetro recuperado desde la peticion, si son iguales, guarda el objeto Usuario en la Sesión HTTP y redirecciona a la página de inicio (menú principal de la aplicación), si no existe entonces, borra el usuario de la sesión y redirecciona a la página iniciar\_sesion.jsp informando sobre el error.
          2. **cerrar\_sesion()**: Atiende la petición GET hecha desde el navegador web, luego elimina de la sesión al objeto Usuario, redirecciona a la vista **iniciar\_sesion**.jsp
          3. **recordar\_pass()**: Recupera los datos de la petición POST (cédula y respuesta, siendo respuesta el valor de una pregunta que solo el usuario conoce), busca el usuario en la BD (usando el método buscarPorId de la clase EntidadXyzCrud) y compara la respuesta enviada con la respuesta que contiene el objeto Usuario encontrado en la BD, si la respuesta es correcta, entonces, le muestra la clave correcta.
          4. **cambiar\_pass()**: Primero debe iniciar sesión, si no conoce la clave, debe recordarla con el paso anterior. Recupera la contraseña enviada como parámetro de la petición POST, recupera el objeto de la sesión, cambia la contraseña actual por la nueva contraseña, actualiza los datos en la BD (usando el método editar de la clase EntidadXyzCrud), vuelve a actualizar el objeto Usuario guardado en la sesión, envía un correo electrónico con la nueva contraseña y por último redirecciona a index.jsp (el menú)