EVALUACION MÓDULO 2

# INSTRUCCIONES GENERALES

* Lea detenidamente las indicaciones antes de empezar a tirar código como loco.
* Utilice como base principal de consulta, los ejercicios anteriores de su repositorio.
* No se olvide de respirar.
* No ha llegado a este punto por suerte, ni por casualidad, ni por coincidencia, sino por su esfuerzo, demuestre todo lo que sabe. Que le vaya super bien!!!

# PARTE 1 – SQL (40 puntos)

**INSTRUCCIONES GENERALES**

Debe guardar un script con todo el código que se le solicita. La evaluación la realizaremos en su propio puesto de trabajo, ejecutando su script.

***CONCEPTOS NUEVOS***

***Tipo de dato AUTOINCREMENTAL***

Es un tipo de dato entero, que se incrementa de forma automática cada vez que se ingresa un registro en la base de datos. La forma de generar un AUTOINCREMENTAL no es estándar en SQL, depende del motor de base de datos que utilice. En PostgreSQL, el tipo de dato para AUTOINCREMENTAL es serial. Así, por ejemplo, en este código:

create table prueba (

codigo serial not null, nombre varchar(100),

)

Se crea una columna llamada codigo, de tipo autoincremental

Cuando se inserta un registro en la tabla, NO SE INGRESA el valor autoincremental, a continuación, un ejemplo del insert en la tabla prueba

insert into prueba (nombre) values (‘justo en la prueba enseñan el autoincremental’);

***Tipo de dato TIMESTAMP***

Para crear una columna que registre la fecha y la hora al mismo tiempo, se utiliza el tipo de dato TIMESTAMP

Puede ingresar los registros como un String, con el siguiente formato: yyyy/MM/dd hh:mm, por ejemplo ‘2020/10/28 09:45’

# EJERCICIO 1

Crear un Database con el nombre evaluacion2.

Escribir los scripts SQL que permitan crear las siguientes tablas:

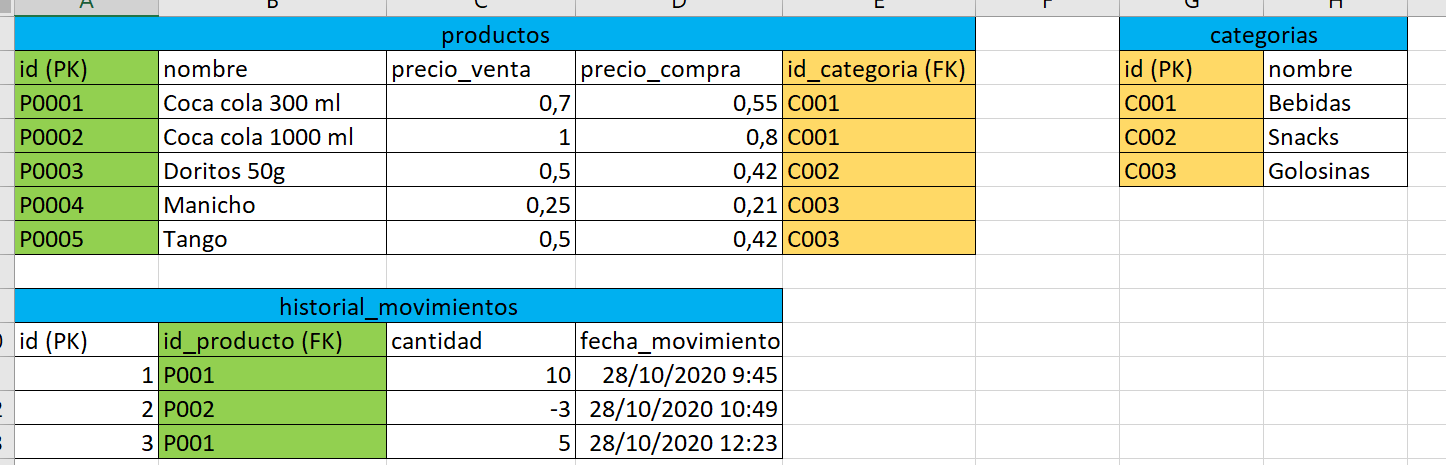


Figura 1. Tablas con datos

En cada tabla se indica la columna que es primary key (PK) Las tablas tienen las siguientes relaciones:



Figura 2. Relaciones entre tablas

En la figura 1, se puede apreciar las columnas que sirven para establecer la relación de clave foránea (FK). Por ejemplo, en productos tiene la columna id\_categoria, que se indica que es la columna de clave foránea (FK) y se encuentra pintada del mismo color que la columna id de categorías que es con la que se establece la relación.

Restricciones de null: Todas las columnas de todas las tablas son obligatorias

Tipos de datos: para enteros usaremos int, para valores monetarios money, los códigos de productos y categorías serán de tipo char, de tamaño 5 y 4 respectivamente. Para la fecha\_movimiento, usar timestamp

# EJERCICIO 2

Crear los inserts para cada tabla, que permitan poblar las tablas con los valores que se presentan en la figura 1.

# EVALUACIÓN

Todos los scripts deben ejecutarse de corrido, para esto debe colocar; al final de cada script. Antes de los scripts de creación agregar las sentencias:

drop table if exists historial\_movimientos; drop table if exists productos;

drop table if exists categorias;

Estas sentencias permitirán borrar las tablas si están creadas

Luego deben estar sus scripts de creación de las tablas, seguidos de los inserts Al final colocar las sentencias:

select \* from productos; select \* from categorias;

select \* from historial\_movimientos;

***Una vez que tenga el script completo, informar al personal de apoyo para que proceda con la calificación.***

# PARTE 2 - JAVA (60 puntos)

(Cambiar arquetipo para poder usar rest web services)

Crear un proyecto Maven, usando el archetype group id: io.github.juneau001, artifact id: webapp-javaee8, version: 1.3:

GroupId: com.krakedev ArtifactId: evaluacion Version: 1.0.0

# EJERCICIO 1 - ENTIDADES (5 puntos)

Agregar un paquete com.krakedev.evaluacion.entidades y agregar las entidades que permitan representar el modelo de base de datos:

Categoria Producto

HistorialMovimiento

Cada entidad debe tener los atributos correspondientes a las columnas de la tabla que representa, con los tipos de datos adecuados.

Para el tipo serial de SQL, utilizar int en java

Para el tipo timesTamp de SQL, usar java.util.Date en java Para el tipo money de SQL, usar BigDecimal en java

Cada entidad debe tener atributos, getters y setters, un constructor vacío, un constructor con parámetros y el método toString sobreescrito.

# EJERCICIO 2 - EXCEPCIONES (5 PUNTOS)

# (ANTES DE EMPEZAR CON EL POSTMAN TOCA INSTALAR APACHE TOMCAT VERSION 9 EN LAS MAQUINAS)

***Concepto NUEVO***

Al conjunto de excepciones de Java, le podemos agregar nuestras propias excepciones, partiendo del concepto que Exception es la mamá de todas las excepciones. Para crear nuestras propias excepciones, simplemente debemos crear una clase que herede de Exception.

EJERCICIO

Crear un nuevo paquete com.krakedev.evaluacion.excepciones, dentro del mismo crear una nueva clase llamada KrakeException, que herede de Exception. Si no recuerda como heredar, revise sus ejercicios de Herencia.

Dentro de KrakeException, definir un constructor que reciba un String llamado mensaje. En el código del constructor, vamos a llamar al constructor del padre y le pasamos mensaje, de esta forma cuando invoquemos al constructor de KrakeException, este va a invocar al constructor de Exception y le va a pasar el mensaje. Este mensaje es el que luego se usa cuando hacemos

getMessage para mostrar la excepción al usuario. Para invocar al constructor de papá se usa super, en este caso super(mensaje).

# EJERCICIO 3 – CONFIGURACION (10 PUNTOS)

Crear un nuevo paquete com.krakedev.evaluacion.utils y copiar la clase ConexionBDD que les será entregada.

En el archivo web.xml agregar los códigos presentados en el archivo txt.

Agregar en el pom.xml las dependencias adjuntadas en el txt.

Configurar el Datasource en apache tomcat, ingresar a la carpeta del mismo, después a la carpeta conf y modificar lo siguiente:

* Copiar el tag resource del archivo txt que se les será entregado, colocarlo dentro del archivo server.xml dentro del tag GlobalNamingResource, configurar el url para que se conecte con el Database que crearon.

Modificar ConexionBDD para que trabaje con los datos de la base de datos a la que se va a conectar

Modificar los métodos de ConexionBDD y Convertidor para que en los bloques catch, en lugar de lanzar una excepción del tipo Exception con mensaje de usuario, lancen una excepción del tipo KrakeDevException. Modificar las firmas de los métodos para que en lugar de hacer un throws Exception, hagan un throws KrakeDev Exception.

En el método conectar de ConexionBDD, agregar un mensaje de debug usando log4j, que diga “obteniendo conexion”.

En el disco local C en la carpeta config de apache, configurar el archivo tomcat users

El export para las prubas se hacen hacia la carpeta del apache tomcat como en los repasos anteriores.

Crear un nuevo paquete com.krakedev.evaluacion.test, crear una clase TestConexionBDD con un main. Invocar a conectar, manejar la excepción mostrando en consola el mensaje de error que trae la excepción.

# EVALUACION

Se evaluará el escenario en el cual la conexión es exitosa y cuando sea fallida.

Para provocar una conexión fallida, puede poner el password incorrecto, en este caso, se evaluará que el mensaje de usuario se muestre en consola y el contenido del archivo de log

***Una vez que esté en este punto, llamar al personal de apoyo para que le califiquen***

# EJERCICIO 4 – SERVICIOS (10 PUNTOS)

Crear un nuevo paquete com.krakedev.evaluacion.servicios.

Dentro del web.xml cambiar el param-value para que se encuentre el paquete de servicios.

Crear una clase llamada ServiciosCategoria

Crear el método estático insertar, recibe un parámetro de tipo Categoria y no retorna nada.

El método inserta en la base de datos la Categoria que recibe como parámetro. Cuando capture la excepción y lance una nueva con mensaje de usuario, usar KrakeDevException en lugar de Exception. La firma del método debe avisar que el método podría lanzar KrakeDevException.

Cuando capture las excepciones en este código: debe tener dos catch, un catch de

KrakeDevException, en cuyo caso solo relanza la exception, por ejemplo throw e; porque esa ya viene con mensaje de usuario y otro catch de Exception, donde lanza una nueva excepción, por ejemplo throw new KrakeDevException(“mensaje de usuario”); con el mensaje de usuario

correspondiente.

Crear una clase TestInsertarCategoria con un método main, desde el main insertar una nueva categoría en la base de datos. Manejar correctamente las excepciones.

# EJERCICIO 5 – ACTUALIZACION (10 PUNTOS)

En ServiciosCategoria, crear el método estático actualizar, recibe un parámetro de tipo Categoria, actualizar el nombre en la base de datos, en función del id.

El manejo de excepciones debe ser igual que el ejercicio 4.

Crear una clase TestActualizarCategoria con un método main, desde el main actualizar una categoría en la base de datos. Manejar correctamente las excepciones.

# EJERCICIO 6 – CONSULTA (10 PUNTOS)

En ServiciosCategoria, crear el método estático buscarPorId, recibe un String y retorna un objeto Categoria con la información buscada. En caso de que no exista, retorna null.

Crear una clase TestBuscarCategoria con un método main, desde el main invocar a buscar. Si encontró la categoría mostrar la información, caso contrario mostrar un mensaje: “No existe la categoría buscada”

# EJERCICIO 7 – RECUPERAR TODOS (10 PUNTOS)

En ServiciosCategoria, crear el método estático recuperarTodos, no recibe parámetros y

retorna un ArrayList de Categoria con todas las categorías que recupera desde la base de datos.

Crear una clase TestRecuperarCategorias con un método main, desde el main invocar a recuperarTodos y mostrar en pantalla el contenido del ArrayList de respuesta.

***Ha terminado su evaluación, llamar al personal de apoyo para que le califiquen***