



ESCUELA DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

INTERFACES – 710044M

PRÁCTICA DE LABORATORIO No. 3

DESARROLLO DE UNA API PARA LA GESTIÓN DE LA DB DE UN LABORATORIO VIRTUAL

TABLA DE CONTENIDOS

1. OBJETIVOS	1
2. INTRODUCCIÓN	1
3. PROCEDIMIENTO	4
4. REFERENCIAS	6

1. OBJETIVOS

- Introducir al estudiante al uso de JavaDoc como herramienta para documentar el código generado al desarrollar una API en Java.
- Introducir al estudiante al proceso de desarrollo de APIs para la conexión y operación de una base de datos usando DAO (Data Access Objects).
- Implementar una API para la gestión de la conexión y la operación a una base de datos.

2. INTRODUCCIÓN

JavaDoc es una herramienta que hace parte de la distribución de desarrollo de Java. Es muy útil para la documentación de los aplicativos Java y sobre todo en el desarrollo de APIs. JavaDoc es un comando que normalmente se invoca usando una interfaz de comandos. Este comando reside en el directorio **bin** de la distribución de Java, lo cual implica que, si se realizaron las actualizaciones de las variables de entorno adecuadamente, este comando ya se encuentra disponible para su uso.

Con el fin de emplearlo, éste asume que ciertas tareas han sido juiciosamente realizadas por el desarrollador, entre ellas se puede citar:

1. Cada clase o método debe poseer un comentario.
2. Cada método se especifica con parámetros y salidas, los cuales debe estar descritos apropiadamente.

Por ejemplo, para introducir la descripción de una clase se usan los comentarios en Java de la siguiente manera:

```
/**  
 * <b>Descripción</b> <p align=justify>  
 * Interfaz para la obtener y modificar información de la base de datos
```

```

* del experimento, teniendo en cuenta las variables atmosféricas, datos
* sobre el lugar de experimentación y el protocolo seguido.
* <p>
* @author Eval Bladimir Bacca Cortes (evbacca@univalle.edu.co)
* @version $Revision: 1.0 $
*/

```

Nótese que se soportan TAGS html para resaltar la información, esto es debido a que una vez se procesan los archivos de una API con JavaDoc se genera una página WEB automáticamente.

JavaDoc soporta una serie de etiquetas las cuales inician con el carácter @, la siguiente tabla resume los caracteres de JavaDoc que se soportan:

Etiqueta	Descripción	Aplica a
@see	A continuación de esta etiqueta se escribe la clase con la cual se asocia la que actualmente se está documentando.	Clase, Método o Variable
@author	Se escribe a continuación el autor de la clase	Clase
@version	Se escribe a continuación la versión y fecha de actualización	Clase
@param	A continuación se escribe el parámetro del método y luego su descripción.	Método
@return	Se escribe a continuación una descripción del valor retornado	Método
@exception	Se escribe el nombre de la excepción y una descripción.	Método
@deprecated	Declara que algún ítem es obsoleto.	Clase, Método o Variable

Con el fin de describir la funcionalidad de un método en particular y especificarlo, se hace uso de los comentarios y de las etiquetas de la siguiente manera:

```

/**
 * Obtiene las propiedades de una prueba con el Boroscopio determinada. Estas propiedades se
almacen en un arreglo
 * asociativo cuyas llaves son: VEL_DESPLAZAMIENTO, PROFUNDIDAD, FPS, ESPESOR ASFALTO,
ESPESOR_BASE, ESPESOR_SUB_BASE,
 * IMAGEN_PROCESADA
 *
 * @param idBoroscopio ID de la prueba con el Boroscopio
 * @return Arreglo asociativo con la estructura definida anteriormente.
 */
public Hashtable getPropiedadesBoroscopio(int idBoroscopio) throws SQLException
{
    Statement stm = robopavConn.createStatement();
    ResultSet res;
    Hashtable datosDB = new Hashtable();

```

```

        res = stm.executeQuery("select vel_desplazamiento, profundidad, fps, espesor_asfalto,
    espesor_base, espesor_sub_base, imagen_procesada from robopav_Boroscopio where
    id='"+String.valueOf(idBoroscopio)+"'");

        res.next();
        datosDB.put("VEL_DESPLAZAMIENTO", new Integer(res.getInt("vel_desplazamiento")));
        datosDB.put("PROFUNDIDAD", new Float(res.getFloat("profundidad")));
        datosDB.put("FPS", new Integer(res.getInt("fps")));
        datosDB.put("ESPESOR_ASFALTO", new Float(res.getFloat("espesor_asfalto")));
        datosDB.put("ESPESOR_BASE", new Float(res.getFloat("espesor_base")));
        datosDB.put("ESPESOR_SUB_BASE", new Float(res.getFloat("espesor_sub_base")));
        datosDB.put("IMAGEN_PROCESADA", res.getString("imagen_procesada"));

        return datosDB;
    }
}

```

JavaDoc puede ser invocado usando un comando en la consola de comandos con todos sus parámetros, o más cómodamente, se puede especificar un archivo con todas sus opciones. A continuación, se mostrará la segunda opción, para la cual se necesitan dos archivos: uno llamado **options** y otro **packages**. Un ejemplo del contenido del archivo **options** es el siguiente:

Opción	Observación
-d co\edu\univalle\ir\doc	Carpeta destino de la documentación
-sourcepath .	Folder donde está el API
-use	Crear páginas que describen el uso de métodos y clases.
-splitIndex	Crear índice
-windowtitle 'JAVA Infrared Image API - DB Access'	Título mostrado en navegador para la documentación.
-doctitle 'JAVA Infrared Image API - DB Access'	Título para la página de inicio.
-header ' JAVA Infrared Image API <p> DB Access'	Texto de cabecera para cada página HTML.
-bottom 'Universidad del Valle <p> Escuela de Ingeniería Eléctrica y Electrónica <p> P.S.I. - Percepción y Sistemas Inteligentes'	Texto inferior de cada página HTML.
-group "Paquetes Principales" "java.sql.*:java.util.*:java.io.*"	Agrupación de paquetes usados en el API.
-group "Paquetes Adicionales" "co.edu.univalle.ir.*"	

El archivo **packages** es muy simple y tiene el siguiente contenido: **co.edu.univalle.ir**. El cual especifica el paquete o API que se está documentando. Una vez definidos estos archivos, el comando a ejecutar es el siguiente:

3. PROCEDIMIENTO

A lo largo de las prácticas de laboratorio del curso de Interfaces se desarrollará una serie de aplicativos que permitan la implementación de un pequeño laboratorio virtual. Uno de ellos es una aplicación en Java con el fin de supervisar y gestionar un hardware que se encuentra conectado a través de un puerto serie.

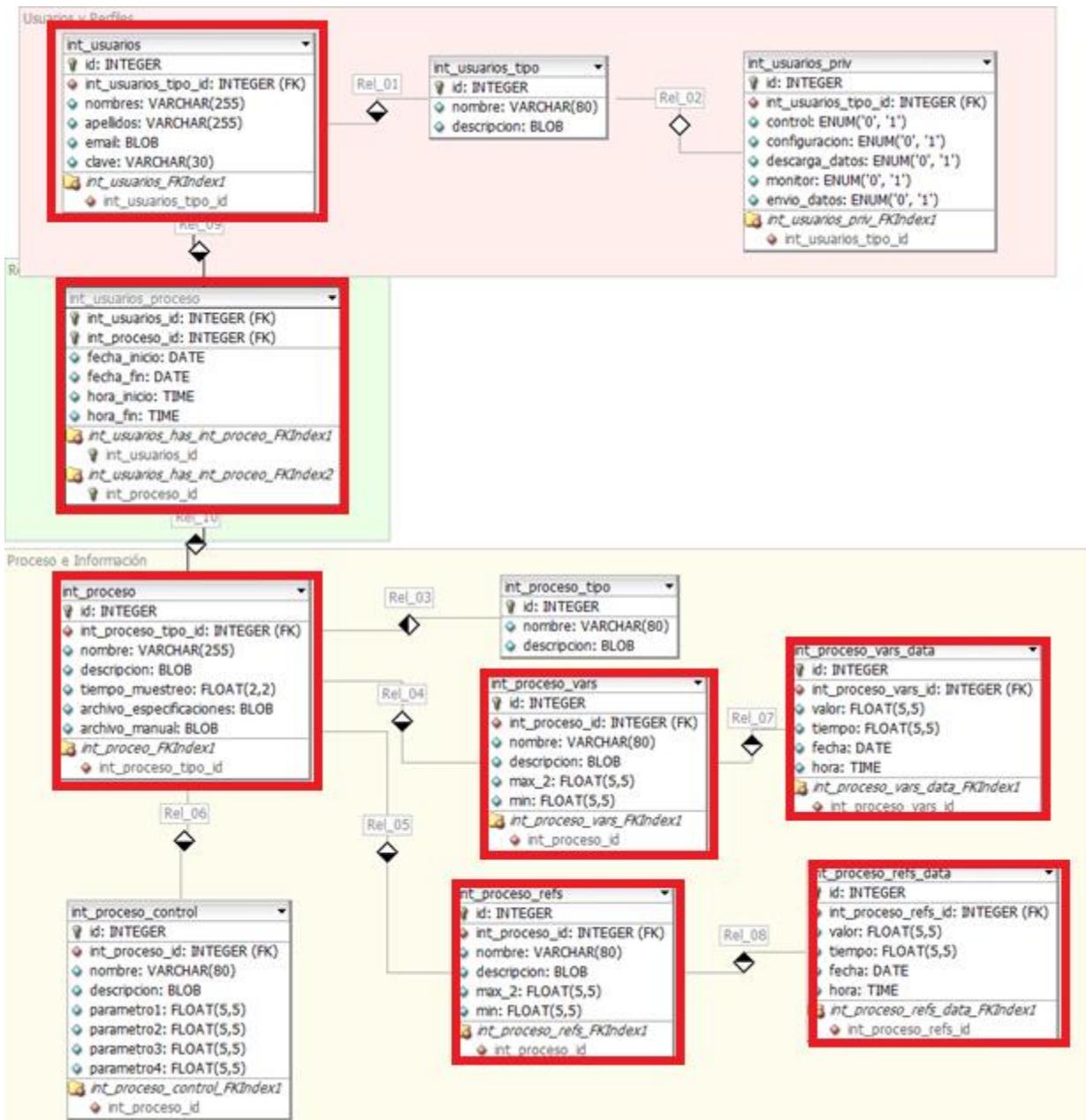


Figura No. 1 Base de datos del laboratorio virtual.

En esta práctica de laboratorio se especificará e implementará el API para la conexión y gestión de la base de datos usando DAO y JDBC. Considerando la base de datos mostrada en la Figura No. 1, se requiere la implementación de un paquete (API) basado en DAO con los siguientes requerimientos:

- 1. Conexión** – Un objeto para el establecimiento de la conexión con la DB, el cual debe implementarse para llevarse a cabo usando los parámetros del constructor o un archivo de texto.

2. **Manejo de Usuarios** – Una interfaz DAO para la gestión de registros de la tabla *int_usuarios*. **NOTA:** la selección de registros puede ser filtrada por diversos campos de esta tabla. Y para la inserción, selección, edición y borrado de la programación para el uso de un proceso particular por un usuario específico en la tabla *int_usuarios_procesos*.
3. **Manejo de Procesos** – Una interfaz DAO para la inserción, selección, edición y borrado de registros en la tabla *int_proceso*.
4. **Manejo de Datos y Variables del Proceso** – Una interfaz DAO para la inserción, selección, y borrado de registros en las tablas *int_proceso.refs_data*, e *int_proceso.vars_data*. Y una interfaz DAO para la selección de registros en las tablas *int_proceso.refs*, *int_proceso.control* e *int_proceso.vars*.



Figura No. 2 Diagrama de concepto del laboratorio.

Finalmente, con el fin de probar el buen funcionamiento del API, los siguientes serán los entregables a calificar:

1. API empaquetado en un JAR.
2. JavaDoc del API.
3. Modificar la aplicación del laboratorio virtual de tal manera que:
 - a. Se use el DAO de acceso a la tabla de *int_usuarios* para hacer una interfaz de login al aplicativo ya previamente desarrollado.
 - b. Usando *phpMyAdmin* ingresar un registro para un proceso en la tabla *int_proceso* y que contenga la información necesaria para describir el hardware que se ha venido usando.
 - c. Usando *phpMyAdmin* ingresar los registros correspondientes en la tabla *int_proceso_vars* e *int_proceso_refs* para las variables de entrada (8 analógicas y 4 digitales) y las variables de salida (4 digitales) respectivamente.
 - d. Usar los DAO de *int_proceso_vars_data* para almacenar las 8 entradas analógicas y las 4 digitales que vienen del sistema embebido; así como para que la GUI tome los datos desde esta tabla para graficar las variables.
 - e. Usar los DAO de *int_proceso_refs_data* para que la GUI almacene el estado de la salida digital correspondiente, luego se lea esta tabla para enviar la información al sistema embebido.
 - f. Usar los DAO de *int_proceso* para modificar el tiempo de muestro en la GUI, y que luego sea enviado al sistema embebido.
 - g. En la Figura No. 2, se debe observar que hay un programa en Java adicional que usa el puerto serial RS232 y la conexión con la DB para almacenar las 8 variables analógicas, las 4 variables digitales de entrada provenientes del sistema embebido; obtiene de la DB el tiempo de muestreo y las 4 variables de salida digitales para ser enviadas al sistema embebido.

- h. Tanto la GUI como el programa del punto g) usan la API de los DAO para acceder a la DB.

4. REFERENCIAS

- [1] George Reese, "Database Programming with JDBC & JAVA", O'Reilly, Segunda edición.
- [2] Larman, C., Applying UML and Patterns. **An Introduction to Object-Oriented Analysis and Design and the Unified Process**. 2 ed: Prentice Hall.
- [3] Violet UML Editor, <http://alexdp.free.fr/violetumleditor/page.php>