MANUAL PARA

ADMINISTRADORES

Documentación orientada a gestionar la Base de Datos, solventar los errores en la ejecución del programa e interpretar la codificación en Java.

**Versión 1.0**



Por Miguel Olmo Hernando

# **PRESENTACIÓN**

Este manual presenta una propuesta de la estructura desarrollada en el programa elaborado por el Equipo 6, cuyo integrante en solitario Miguel Olmo Hernando es un estudiante de Primero de Desarrollo de Aplicaciones Web del Instituto Egibide Arriaga de Vitoria-Gasteiz (Araba).

* **Entre los objetivos que persigue esta propuesta se tienen tres puntos clave:**

• Identificar El Problema - Enunciado:

• Definir Y Representar El Problema Del Enunciado - Fases Del Proyecto:

• Temporización - Rúbricas - Criterios De Evaluación Del Proyecto:

* **Para completar el manual tienen a su disposición una galería de imágenes de mayor calidad y resolución en el repositorio de GitHub, carpeta Imágenes. De igual forma pueden acceder al Blog en el que se ha generado semanalmente entradas, en las cuales puede ir viendo la evolución del proyecto.**

• poner link mig

* **A destacar en este manual:**
* Se desarrolla mediante la siguiente estructura:

<https://es.wikipedia.org/wiki/Partes_del_libro>

* Esta creado mediante las herramientas de Microsoft Office 2016 y NitroProPDF. Si tienen problemas con la plantilla de Word utilicen el PDF para este visionado. Recuerden que en este caso puede ser que no tengan disponible el índice multimedia generado.

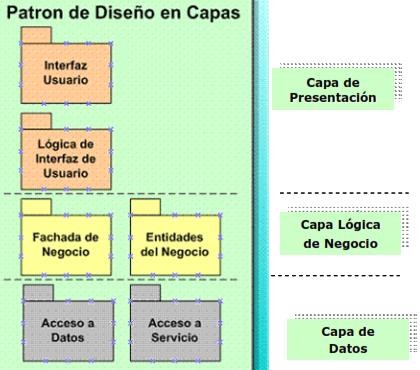
# **INTRODUCCIÓN**



## **ARQUITECTURA DE LA APLICACIÓN**

Uno de los patrones arquitectónicos más usados en la actualidad para el desarrollo de aplicaciones de porte empresarial es el denominado “arquitectura de 3 o más capas”.

Este estilo arquitectural separa en algunos casos físicamente, los aspectos de presentación de la aplicación (interfaz de usuario), la lógica de negocios (automatización del flujo trabajo) y la gestión de los datos (bases de datos), tal como se muestra en la siguiente imagen:



La capa de presentación es la encargada de manejar la interfaz del usuario, controlando la captura y presentación de los datos y recibiendo los eventos accionados por los usuarios a través de la interfaz.

Esta capa se comunica únicamente con la capa lógica de negocios.

La capa lógica de negocios tiene la responsabilidad de manejar la funcionalidad del sistema, implementando a través de objetos de negocio las reglas de negocio que deben cumplirse. Esta capa se comunica con la capa de presentación para recibir solicitudes y presentar resultados y con la capa de datos, para solicitar al gestor de base de datos que almacene o recupere datos.

La capa de datos (llamada en algunos casos capa de persistencia) es la responsable del almacenamiento y recuperación de los datos. Se comunica únicamente con la capa de lógica de negocios.

Un ejemplo de la descripción y estructura de la arquitectura para nuestra aplicación es la siguiente:



El sistema empleará el estilo arquitectónico de capas y será organizado en tres capas: la capa de interfaz, la capa de la aplicación y la capa de almacenamiento.

La capa de interfaz contendrá la interfaz gráfica del usuario que le permitirá a los usuarios interactuar con la aplicación. Esta capa será implementada usando principalmente Java y gracias al software NetBeans.

La capa de la aplicación contendrá la lógica y reglas para almacenar datos en la capa de la base de datos y también para recuperar éstos de acuerdo con las necesidades del usuario.

Finalmente, la capa de almacenamiento guardará los datos requeridos por el sistema. Para la gestión de la Base de Datos usamos Vagrant que es quien nos permite generar un entorno de desarrollo mediante una Máquina Virtual en la que a posteriori junto con SQL Developer insertaremos las tablas, procedimientos, paquetes y disparadores necesarios para generar, obtener o almacenar los datos.

### **MODELO UML**

Se presenta a continuación el modelado de la aplicación, basado en la metodología UML. UML (Lenguaje de Modelado Unificado) es una especificación de notación orientada a objetos, el cual se compone de diferentes diagramas, los cuales representan las diferentes etapas del desarrollo del proyecto.

La aplicación usada para generar este modelo es Data Modeler.

### **MODELO MER**

Turno del Diagrama de Entidad Relación, gracias a este diagrama representamos las entidades relevantes de nuestra aplicación y obtenemos así la información de sus interrelaciones y propiedades.

La aplicación usada para generar este modelo es Data Modeler.

### **DIAGRAMA DE CLASES**

Generamos mediante la aplicación Visual Paradigm este diagrama de clases que nos permitirá representar gráficamente y de manera estática la estructura general de nuestra aplicación, mostrando cada una de las clases y sus interacciones (como herencias, asociaciones, etc.), representadas en forma de bloques, los cuales son unidos mediante líneas y arcos.

Este diagrama es fundamental del modelado con UML, siendo ampliamente utilizados tanto para análisis como para diseño de sistemas y software en general.

# **ÍNDICE**

[1. PRESENTACIÓN 2](#_Toc514628799)

[2. INTRODUCCIÓN 3](#_Toc514628800)

[2.1. ARQUITECTURA DE LA APLICACIÓN 3](#_Toc514628805)

[2.1.1. MODELO UML 5](#_Toc514628806)

[2.1.2. MODELO MER 5](#_Toc514628807)

[2.1.3. DIAGRAMA DE CLASES 5](#_Toc514628808)

[3. ÍNDICE 6](#_Toc514628809)

[4. INSTALACIÓN DEL SOFTWARE 7](#_Toc514628810)

[4.1. VAGRANT 7](#_Toc514628813)

[4.2. SQL DEVELOPER 10](#_Toc514628814)

[4.3. NETBEANS 12](#_Toc514628815)

[5. GESTIÓN Y CARGA DE LA BASE DE DATOS 13](#_Toc514628816)

[5.1. EJECUTAR SCRIPT ORACLE 14](#_Toc514628818)

[5.2. CARGAR LOS PAQUETES Y PROCEDIMIENTOS 15](#_Toc514628819)

[5.3. CARGAR LOS TRIGGERS / DISPARADORES 17](#_Toc514628820)

[5.4. PAQUETE – TRIGGER - CONTROLAR EL PRESUPUESTO DEL EQUIPO - MUTANTES 18](#_Toc514628821)

[6. GESTIÓN PREVIA DEL PROGRAMA 20](#_Toc514628822)

[6.1. ABRIR EL PROYECTO CON NETBEANS 20](#_Toc514628824)

[6.2. SOLUCIONAR PROBLEMAS – LIBRERIAS 21](#_Toc514628825)

[6.3. EDITAR LOS PARAMETROS DE CONEXIÓN 23](#_Toc514628826)

[6.4. MOSTRANDO LA ESTRUCTURA INTERNA DEL PROYECTO 24](#_Toc514628827)

[7. PROGRAMA EN EJECUCIÓN 25](#_Toc514628828)

# **INSTALACIÓN DEL SOFTWARE**



## **VAGRANT**

Antes de ponernos con la Interfaz Gráfica de nuestra Aplicación y explicar cómo navegar a través de ellas, es imprescindible comentar este punto, sin tener las herramientas que se requieren, no podrán ejecutar correctamente la aplicación.

**Este proceso, se resumen en cinco puntos clave:**

1. Deben de acceder al siguiente enlace y seguir los pasos que se muestran para poder **instalar Vagrant** <https://github.com/Egibide/vagrant-oracle-12-ee-linux>

* Si en este punto tienen problemas al cargar el Vagrant fuera de Egibide, eliminen la siguiente línea del archivo Vagrantfile. Utilicen un editor de texto como Notepad++

**Buscar y quitar:**

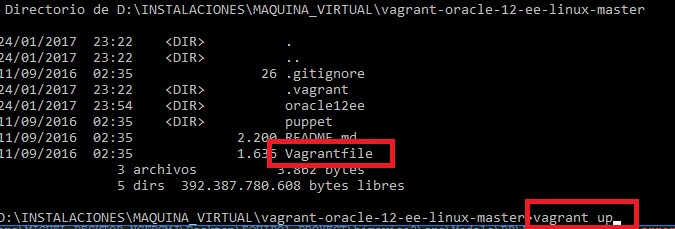
db12102.vm.box\_url = ["~~http://nunki.diocesanas.org/mv/vagrant/centos-6.5-x86\_64.box~~", "https://dl.dropboxusercontent.com/s/np39xdpw05wfmv4/centos-6.5-x86\_64.box"]

**Para dejar la siguiente:**

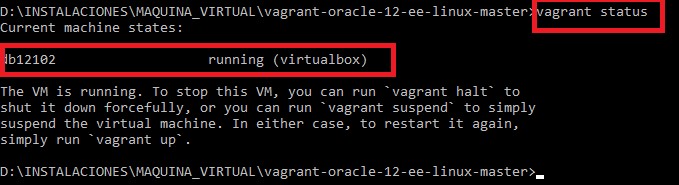
db12102.vm.box\_url = ["https://dl.dropboxusercontent.com/s/np39xdpw05wfmv4/centos-

6.5-x86\_64.box"]

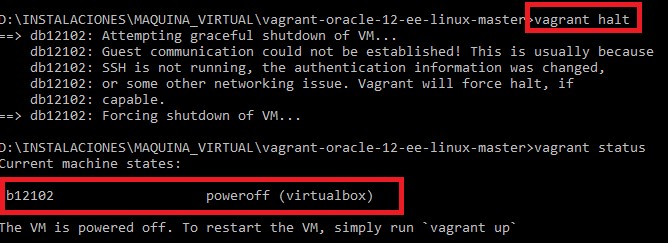
* Quitando la URL de nunki.diocesanas está comprobado que CentOs se descarga antes y no genera errores si se pretende instalar fuera de las instalaciones de Egibide.
* Una vez tengan realizados todos los pasos que se indican en el link del repositorio anteriormente mostrado: <https://github.com/Egibide/vagrant-oracle-12-ee-linux> ya podrán ejecutar Vagrant muy fácilmente:



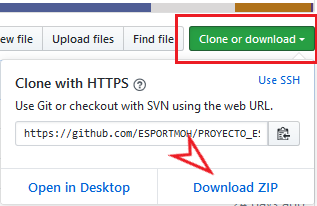
* Comprueben mediante el comando *vagrant status* que efectivamente la máquina está cargando:



* Si desean parar Vagrant, siguiendo los pasos indicados (vagrant halt) es igual de sencillo:



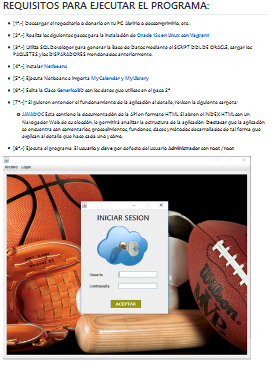
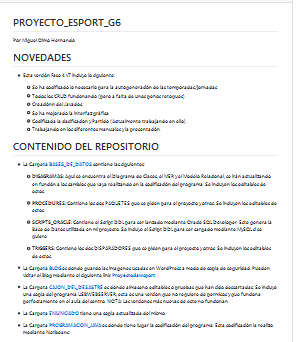
* Llegados a este punto, accedan al repositorio y comiencen su descarga. Esta descarga se recomienda gestionar en una carpeta previamente creada en el escritorio o el home de su Sistema Operativo, luego pueden descomprimirlo y tener acceso a todo el proyecto.



* + Link al Repositorio de la Organización E-SPORT:

[**https://github.com/ESPORTMOH/PROYECTO\_ESPORT\_G6**](https://github.com/ESPORTMOH/PROYECTO_ESPORT_G6)

* Al pie del repositorio encontrarán el README, este os guiara por el contenido del mismo, como por los mismos pasos que aquí se indican.



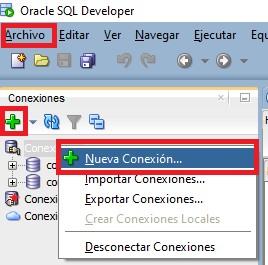
## **SQL DEVELOPER**

1. Ahora requieren de la herramienta SQLDeveloper de Oracle, pueden seguir los pasos del punto 1) o bien pueden descargarla mediante otros medios.

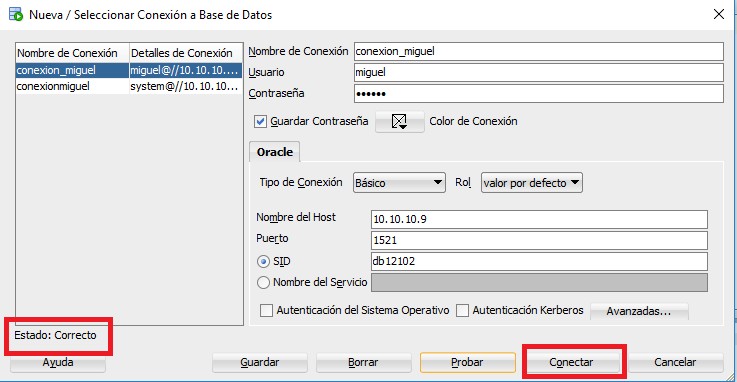
<http://www.oracle.com/technetwork/developer-tools/sql-developer/downloads/index.html>

Con los pasos 1), 2) terminados y estando Vagrant ejecutado, ahora es necesario mediante SQL Developer crear una conexión, pueden al igual que en el punto 2) usar los datos de conexión que se muestran o bien generar otros, lo que sí tienen que tener muy en cuenta es recordar dichos datos, pues luego nuestra Aplicación requerirá de ellos para conectarnos a la Base de Datos. Si ya tienen una conexión establecida con anterioridad o ya conoce el funcionamiento de este punto, diríjase al siguiente apartado **– Gestión y Carga de la Base de Datos -**, si no continúe.

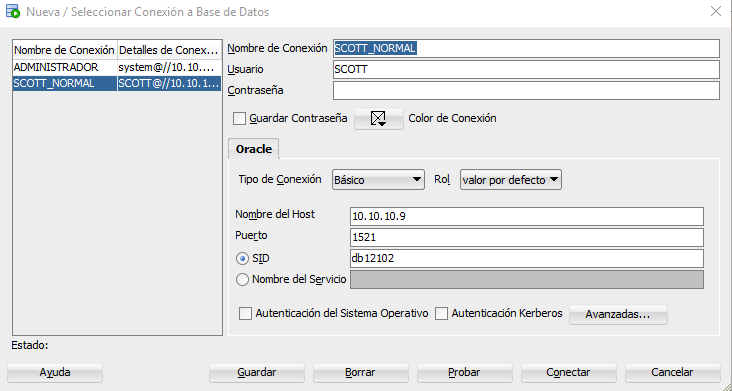
* Abrimos SQL Developer y tenemos varias formas de crear una conexión, desde archivo nueva conexión, desde el + verde o bien haciendo clic en el botón derecho del ratón y Nueva Conexión.



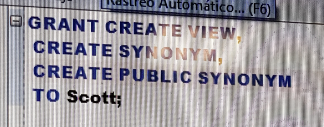
* Introduzca los datos de su nueva conexión, recuerde que inicialmente tiene que entrar como usuario administrador así que ojo con lo que hacen. Rellenados los campos, pulsar el botón Probar y así podréis ver si os funciona, si funciona conectaros. Una vez acceden como usuario Administrador pueden crear un nuevo usuario con menos permisos y evitar riesgos.



* Nota: La contraseña de sys y system es oracle.



1. Ultimo paso, cargar nuestra Base de Datos, paquetes, procedimientos y disparadores en SQL Developer. Para este paso se recomienda generar un nuevo usuario y no utilizar el System, en este caso se ha generado el de SCOTT.

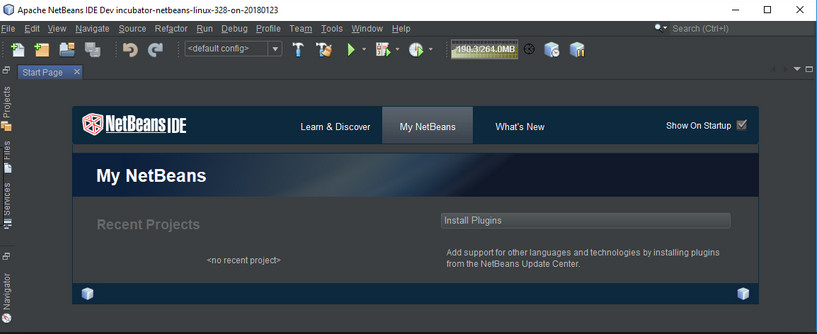


* Ejemplo de cómo dar permisos a un usuario de SQL Developer
* Como crear un nuevo usuario: [http://cursoanpefernandoortiz.blogspot.com.es/2014/12/crear-un-nuevo-usuario-oracle-con-](http://cursoanpefernandoortiz.blogspot.com.es/2014/12/crear-un-nuevo-usuario-oracle-con-sql.html) [sql.html](http://cursoanpefernandoortiz.blogspot.com.es/2014/12/crear-un-nuevo-usuario-oracle-con-sql.html)

## **NETBEANS**

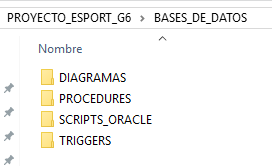
1. Es el momento de instalar NetBeans, para ello accedan a la Web para su descarga y sigan los pasos que se indican:

* Link descarga: <https://netbeans.org/>
* Manual de instalación: <https://netbeans.org/community/releases/82/install.html>



# GESTIÓN Y CARGA DE LA BASE DE DATOS

* Si ya disponen de una conexión con anterioridad y han pasado a este punto sin necesidad de hacer los anteriores. - o bien quiere cargar de nuevo nuestra Base de Datos le recomendamos borre cualquier tabla que tenga creada antes. En caso de tener la misma, el Script se encargará de ello por sí solo, pero os muestro como a cómo hacerlo de forma manual.
* Diríjanse a la carpeta en donde descargaron y descomprimieron el repositorio, localicen la siguiente carpeta *BASES\_DE\_DATOS* en ella se ubican las siguientes:



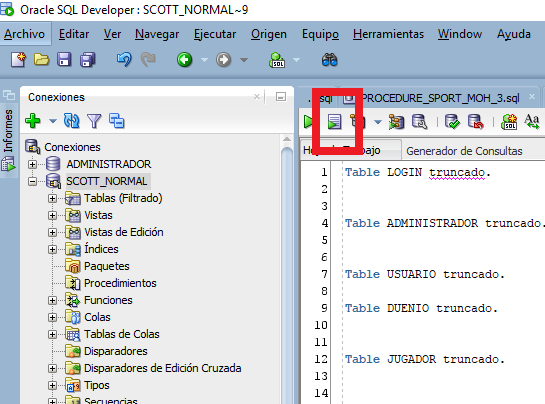
* DIAGRAMAS, contiene los diferentes diagramas mencionados anteriormente y sus editables.
* PROCEDURES, contiene los Paquetes y Procedimientos que se cargaran en la Base de Datos, también se incluyen pruebas y editables.
* SCRIPTS\_ORACLE, contiene el fichero DDL de ORACLE que nos permitirá crear la Base de Datos, se incluye otro para MySQL que no usaremos en este caso y editables y pruebas y comprobaciones varias.
* TRIGGERS, contiene los disparadores que se requieren para el correcto funcionamiento de la Base de Datos, de igual forma se incluyen los editables y pruebas varias.



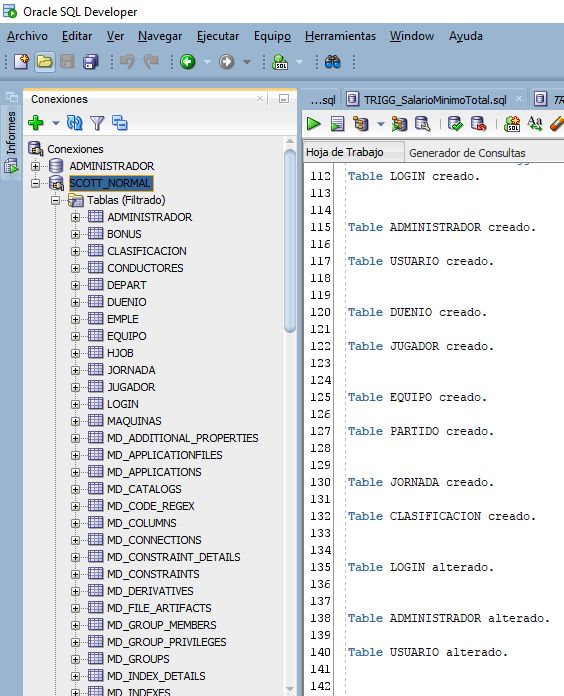
## EJECUTAR SCRIPT ORACLE

Entramos en la carpeta *SCRIPTS\_ORACLE*, seleccionamos el *ScriptDDL\_ESPORT\_ORACLE\_OK\_MOH* y lo arrastramos para soltarlo en el SQL Developer (recordamos que la conexión a de establecerse como mencionamos en pasos anteriores para que este paso nos funcione).

Una vez lo tengamos, simplemente ejecutamos el Script (ver icono remarcado en rojo). Inicialmente nos borra cualquier tabla que pueda llamarse igual a las que ya pueda tener en su BD, por lo que deberá de revisar este si quiere mantener su información. De igual forma pasa con los paquetes, procedimientos y disparadores.



Este Script genera todas las tablas, restricciones e inserciones de datos necesarias para cargar a posteriori el programa con un contenido base con el que poder empezar a trabajar.

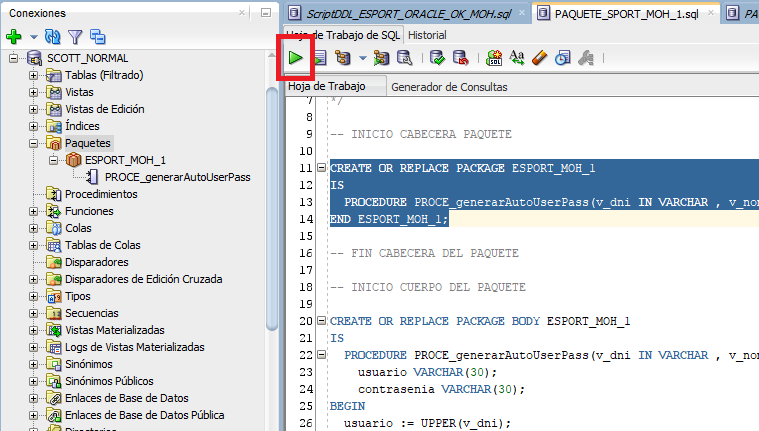


## CARGAR LOS PAQUETES Y PROCEDIMIENTOS

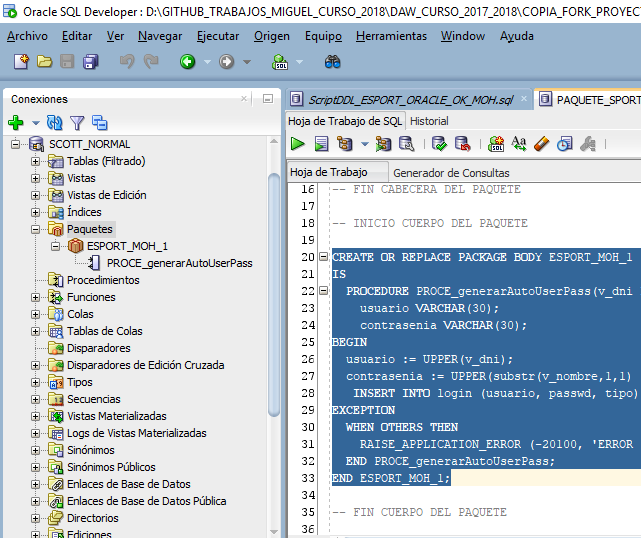
No vamos a complicarnos y como anteriormente, nos vamos a la carpeta correspondiente *PROCEDURES*, seleccionamos el *PAQUETE\_SPORT\_MOH\_1*, *PAQUETE\_SPORT\_MOH\_2* y el *PROCEDURE\_SPORT\_MOH\_3,* los arrastramos y soltamos en el SQL Developer.

A continuación, vamos a cargar estos en nuestra BD, pueden empezar por el que quieran, os mostramos como cargar un paquete y un procedimiento.

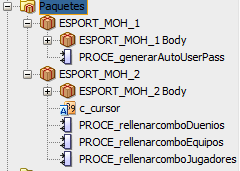
En el caso de los Paquetes, cargamos primero la **cabecera**, la seleccionamos y le damos al icono remarcado en rojo.



Después hacemos lo mismo, pero con el **cuerpo** del paquete.

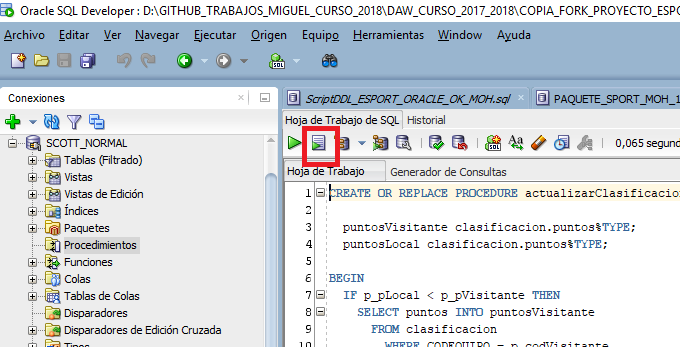


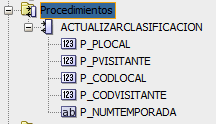
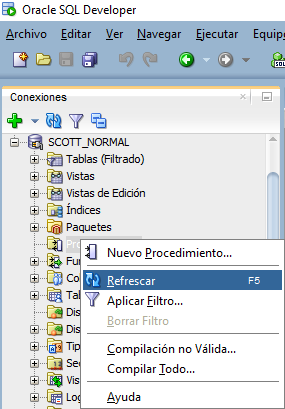
Por último, comprobamos que se han cargado correctamente en nuestra Base de Datos.



* Este paso lo vamos a realizar con los dos Paquetes que se muestran.

Para cargar el Procedimiento, basta con ejecutarlo, no se requiere de una cabecera o cuerpo.



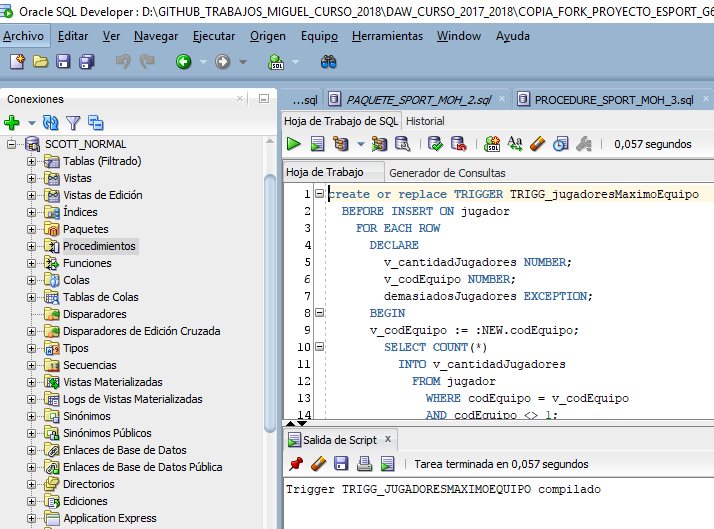


* Si no ven a la primera que los paquetes o procedimientos están cargados, refresque.

## CARGAR LOS TRIGGERS / DISPARADORES

Vamos a la carpeta *TRIGGERS* y como con los paquetes, seleccionamos su contenido *TRIGG\_JugadoresMaximoEquipo* y *TRIGG\_SalarioMinimoTotal* y los arrastramos y soltamos en el SQL Developer.

Ejecutamos *TRIGG\_JugadoresMaximoEquipo* y podemos ver que este se compila perfectamente.



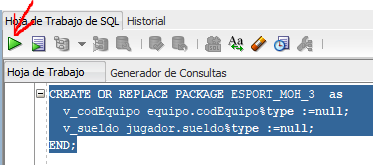
## PAQUETE – TRIGGER - CONTROLAR EL PRESUPUESTO DEL EQUIPO - MUTANTES

Este punto se ha decidido dejar separado de los demás, puesto que el fichero *TRIGG\_SalarioMinimoTotal* nos permite controlar que la Tabla Equipo no Mute. ¿Qué quiere decir, Mutante? Una tabla mutante es una tabla que está siendo modificada. Si un ‘trigger’ hace referencia a la propia tabla sobre la que se dispara se producirá una excepción con error ORA-04091.

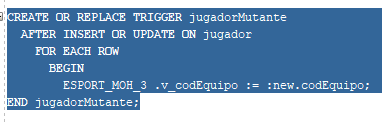
Según esto, cualquier ‘Trigger’ que haga una simple consulta sobre la tabla que lo dispara generará dicho ORA-04091.

¿Qué hacemos para controlarlo? Para controlarlo es necesario utilizar una colección definida en un paquete. Luego demuestro cómo usar el disparador compuesto y agregarlo en nuestra Base de Datos, para resolver el problema de forma mucho más simple:

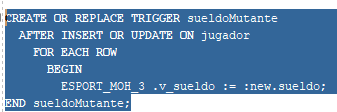
Con el *TRIGG\_SalarioMinimoTotal* en SQL Developer vamos a seleccionar y ejecutar el Paquete *ESPORT\_MOH3* como se muestra en la siguiente imagen:



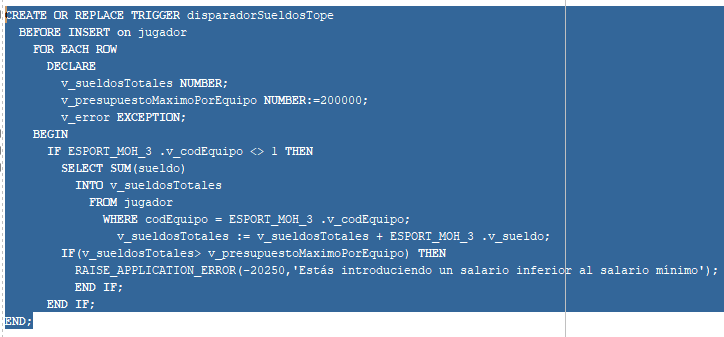
Luego hacemos lo mismo con el Trigger llamado *jugadorMutante:*



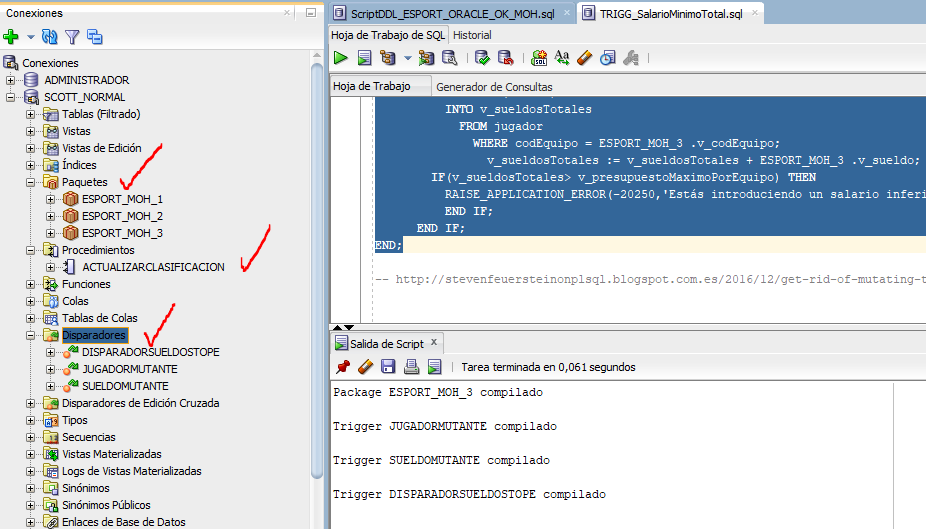
Y con el Trigger *sueldoMutante:*



Por último, cargamos el Trigger *disparadorSueldosTope* que nos controlará la suma total de los salarios de los jugadores con el presupuesto máximo de 200000€ del equipo, llegados a este máximo el Dueño no podrá fichar un jugador.



Finalmente podemos ver que se han generado e insertado correctamente en nuestra Base de Datos:



Los Triggers saltaran automáticamente en nuestra Base de Datos según la tabla en la que intervengan, dicho mensaje de error se verá reflejado en la aplicación. Los procedimientos y paquetes son los que llamaremos directamente al cargar una función de la aplicación usando la **Interfaz CallableStatement.**



* Ejemplo de llamada al procedimiento ESPORT\_MOH\_2

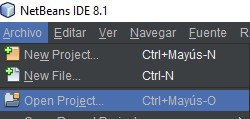
# GESTIÓN PREVIA DEL PROGRAMA

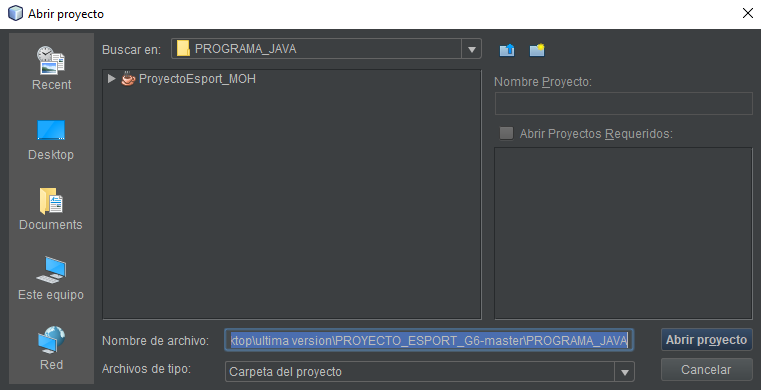


## ABRIR EL PROYECTO CON NETBEANS

Ya con nuestra Base de Datos preparada, solo nos queda ejecutar el programa creado en Java, en este caso y como se menciona al principio de este manual, usaremos Netbeans como plataforma de carga.

Ejecutamos Netbeans y vamos a abrir el Proyecto.



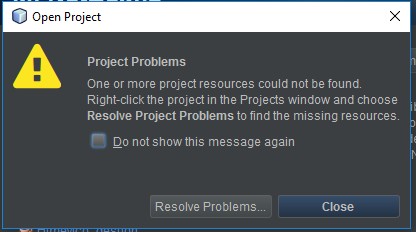


Como se ve en la imagen, el Proyecto (*ProyectoESport\_MOH*) se encuentra en la carpeta PROGRAMA\_JAVA del repositorio extraído.

## SOLUCIONAR PROBLEMAS – LIBRERIAS

La primera vez que abramos el proyecto, Netbeans nos mostrara un error, este indica que algunas de las librerías que se han usado para codificar el programa no las tiene referenciadas/establecidas Netbeans por lo que tendrán que seguir los siguientes pasos.

* Este es el mensaje de error, clic en Resolve Problems o Resolver Problemas



* La segunda ventana nos indicara los errores concretos que tenemos, bien uno de ellos tiene que ver con el tipo de librería usada para generar los calendarios y el otro con la conexión.

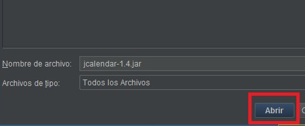




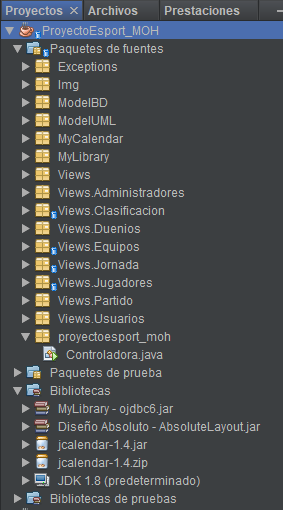
* Vamos en orden, seleccionamos el primero y clicamos en RESOLVE, nos abrirá una ventana en la que tendremos que seleccionar la ruta en la que se encuentran las librerías que se requieren (estas están dentro del mismo proyecto, no hay problema), concretamente se encuentran en estas dos carpetas de la imagen:



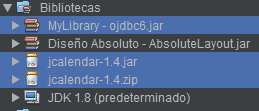
* Una vez ubicada la carpeta seleccionan la librería y hagan clic en Abrir. Este paso obviamente lo tendremos que hacer dos veces, una por cada error de librería.



Estos pasos nos han permitido poder abrir el proyecto, el cual tendrá la siguiente estructura:



* Y en la parte final, las librerías que hemos agregado.



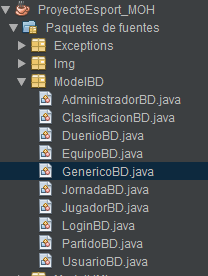
* En el caso de la librería del Calendario, si necesitan hacer un cambio en los calendarios, deberán de añadir a la paleta dicha selección de calendarios. Sigan estos pasos si lo desean:

[Agregar nuevos componentes a la paleta de NetBeans](https://javiergarciaescobedo.es/programacion-en-java/2-clases-y-objetos/23-agregar-nuevos-qbeansq-java-a-la-paleta-de-netbeans)

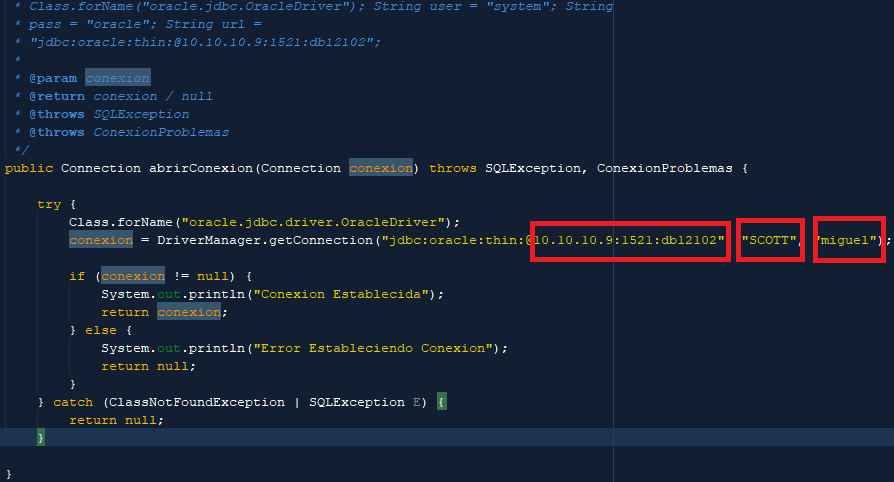
## EDITAR LOS PARAMETROS DE CONEXIÓN

Para que las operaciones realizadas al ejecutar el programa tengan efecto en nuestra Base de Datos o viceversa se requiere una clase que nos permita establecer o cerrar la conexión, para ello vamos a editar dicha clase con el fin de poder trabajar en la Base de Datos generada anteriormente.

* Nos vamos a ModelBD y hacemos doble clic en GenericoBD.



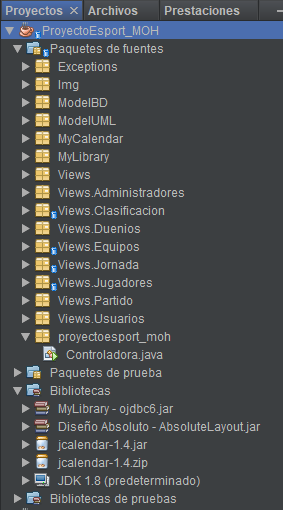
* Es esta clase en donde vamos a poder poner los parámetros de nuestro servidor BD o los datos generados en los puntos 4.1 y 4.2 de la Instalación del Software previa. Ver recuadros marcados en rojo.



* Recuerdo, tienen que usar sus datos, no los que se muestran, estos son orientativos.

## MOSTRANDO LA ESTRUCTURA INTERNA DEL PROYECTO

La estructura del proyecto en NetBeans no debe mostrarnos ningún elemento de fallo y ha de ser exacta a la de la imagen anteriormente mostrada:



En la carpeta **Exceptions** se recogen todos los posibles errores que pueda generar una función o un usuario, por ejemplo, que falten datos para generar un equipo o si en el caso anterior no establecemos una correcta conexión esta se verá reflejada en pantalla.

Las carpetas **Img, MyCalendar, MyLibrary** contienen por orden las imágenes usadas en la aplicación, la librería para el calendario y el driver de Oracle que ya mencionados.

En la carpeta **ModelUML** es donde tienen lugar las Clases que se han presentado en el Diagrama de Clases, sus atributos, constructores, métodos getter y setter con los que poder traer o acceder a los atributos.

En la carpeta **ModelBD** se ecuentras las mismas clases que en ModelUML pero que nos permitirán realizar todas las llamadas programadas a nuestra Base de Datos para realizar las operaciones CRUD, rellenar combos, etc…

La carpeta **Views** contiene las vistas (ventanas) de Home e Inicio de Sesión, esta carpeta a su vez contiene otras ocho carpetas. Estas carpetas también contienen las ventanas que nosotros como administradores vamos a poder controlar y ver al igual que las propias para los usuarios registrados, así como para los dueños de los equipos que realicen los fichajes o consultas oportunas.

La carpeta **proyectoesport\_moh** contiene la clase controladora (el musculo motor) que nos permitirá recoger todas las solicitudes generadas en las ventanas por los usuarios y pasarlas según corresponda a una clase BD para realizar x consulta o que nos abra otra ventana. Es el núcleo inicial de un camino por el cual nos permite acceder a otros. Mientras que lo generado en la Base de Datos es para nosotros (el cerebro) quien guarda la información.

Por ultimo podrá ver las **bibliotecas** con las librerías importadas con antelación.

# PROGRAMA EN EJECUCIÓN

El programa en ejecución muestra una Interfaz sencilla, amigable y de fácil comprensión.