

SEP

SES

TNM

INSTITUTO TECNOLÓGICO DE CHIHUAHUA II



“OBEX: EXPLORADOR DE OBJETOS EN AMBIENTES ERP”

TESIS

PARA OBTENER EL GRADO DE

MAESTRO EN SISTEMAS COMPUTACIONALES

PRESENTA

EDUARDO TORRES ÁVILA

DIRECTOR DE TESIS

M.I.S.C. JESÚS ARTURO ALVARADO
GRANADINO

CO-DIRECTOR DE TESIS

DR. HERNÁN DE LA GARZA
GUTIÉRREZ

CHIHUAHUA, CHIH. A DICIEMBRE DE 2018

DEDICATORIA

A mi adorada esposa, sin cuyo amor, paciencia y apoyo incondicional no hubiera podido llegar a este momento.

A mis hijas, mi inagotable razón de seguir siempre adelante.

A mi madre, que estoy seguro me ha acompañado también en este recorrido, desde su cielo.

AGRADECIMIENTOS

Agradezco a Dios con todo mi corazón por permitirme ver la conclusión de este proyecto.

A mis compañeros de trabajo que de alguna manera estuvieron involucrados a lo largo de este proceso, y que nunca escatimaron su tiempo y apoyo.

Especialmente, a todos mis profesores en esta Maestría, porque en este par de años y meses me brindaron la oportunidad de renovarme, y redescubrir el gusto por el reto constante que significan las tecnologías de la información, de inicio mi principal motivo para elegir esta profesión hace ya casi 30 años.

RESUMEN

Cuando se trabaja en el desarrollo de software que debe funcionar en sistemas ERP comerciales es común el uso de herramientas CASE de código cerrado. Mientras que estas herramientas facilitan el desarrollo, pueden complicar el análisis del impacto de un cambio o reutilización de un objeto en un programa nuevo. Si el desarrollo se realiza utilizando “Oracle Developer®”, propiedad de Oracle Corporation, los archivos que contienen todos los objetos son creados en un formato que impide que puedan ser analizados si no es utilizando dicha herramienta CASE. Las técnicas de Procesamiento de Lenguaje Natural sin embargo pueden ayudar a analizar el contenido de estos archivos, sin importar su formato, haciendo más rápido el proceso de planeación de pruebas de calidad y estimación de impacto de actualizaciones y nuevos programas en un sistema ERP.

ABSTRACT

When developing software designed to work in commercial ERP systems ‘environments, it is common the utilization of CASE tools, which most of the time are proprietary software. While these tools make the development process easier and faster, they may complicate the analysis of impact caused by updates or new programs added to the system. If this development is done using “Oracle Developer®”, owned by Oracle Corp., the files containing all the objects are created in a format that allows this analysis only if they are open within this CASE tool. Natural Language Processing techniques however, can be used to analyze the content of these files, regardless of their format, speeding up the process of planning quality tests and estimating the impact of updates and new programs in ERP systems.

CONTENIDO

I.	Introducción	1
1.1	Introducción.....	1
1.2	Planteamiento del problema	2
1.3	Hipótesis	4
1.4	Alcances y limitaciones	5
1.5	Justificación.....	5
1.6	Objetivo	6
II.	Estado del Arte	7
2.1	Plataformas de entrega ágil empresarial.....	7
III.	Marco Teórico	9
3.1	Herramientas CASE	9
3.2	Inteligencia Artificial	9
3.3	Procesamiento de Lenguaje Natural	10
3.4	Procesamiento de Lenguaje Natural - Tokenización	11
3.5	Sistema ERP	11
3.6	Codificación UTF-8	11
IV.	Desarrollo	12
4.1	OBEX: Proceso General.....	12
4.2	Contenedores de Código	13
4.3	Procesamiento de Lenguaje Natural y Tokenización.....	18
4.4	Detección de Objetos	19
4.5	Ejecución de pruebas.....	21
V.	Resultados	25
VI.	Conclusiones.....	28
VII.	Bibliografía	29
VIII.	Anexos	30
8.1	Código fuente OBEX.....	30

ÍNDICE DE FIGURAS

1.1 Reporte creado en Oracle Developer©.....	3
1.2 Reporte creado en Oracle Developer, editado desde Notepad	3
1.3 Plantilla html creada en Oracle JDeveloper©	4
2.1 Página web de Panaya©	7
3.1 Campos de la Inteligencia Artificial.....	9
3.2 Procesamiento de Lenguaje Natural aplicado en búsquedas web.....	10
4.1 OBEX: Proceso general.....	12
4.2 Ejemplo de programación Java utilizando Oracle JDeveloper©	13
4.3 Ejemplo de programa SQL, visualizado con la herramienta Toad for Oracle 12.6©	14
4.4 Ejemplo de plantilla programada en Oracle Forms Builder©	15
4.5 Ejemplo de reporte programado en Oracle Reports Builder©	16
4.6 Ejemplo de “Workflow” programado en Oracle Workflow Builder©	17
4.7 Reporte creado en Oracle Developer©, editado desde Sublime Text© usando Codificación UTF-8	17
4.8 Código Java para abrir un archivo usando codificación UTF-8 y extraer los tokens.....	19
4.9 Código Java para detección y eliminación de caracteres de control y otros ilegibles.....	20
4.10 Código Java para detección de objetos dentro del texto analizado.....	20
4.11 Activación de Apache Derby Network Server desde NetBeans IDE 8.2	21
4.12 Ventana de ejecución de OBEX.....	22
4.13 Código Java para la conexión a la base de datos Oracle	22
4.14 Información mostrada si la conexión a la base de datos resulta exitosa	23
4.15 Archivo de texto generado por OBEX	23
4.16 Archivo generado por OBEX, abierto desde MS Excel©	24
5.1 Ejecución de OBEX sobre archivos de plantillas y reportes.....	25
5.2 Resultado en MS Excel© de la ejecución de OBEX sobre archivos de plantillas y reportes ...	26
5.3 Ejecución final de OBEX sobre archivos de plantillas y reportes	27

I. INTRODUCCIÓN

1.1. Introducción.

Interceramic es una empresa orgullosamente mexicana, líder en la fabricación y distribución de pisos y azulejos cerámicos, muebles de baño, materiales de instalación y piedra natural, con presencia en México, Estados Unidos, Canadá, Guatemala y Panamá.

A fin de poder mantener sus operaciones en estos países, la empresa requiere de un conjunto de avanzados sistemas y tecnologías de cómputo, con un equipo humano dedicado a mantener y soportar dichos sistemas.

Para este propósito se vale de Oracle E-Business Suite® (sistema ERP) como el software principal para el control de prácticamente todas sus operaciones corporativas, tales como: manufactura, control de almacenes, ventas, compras, finanzas y administración.

Actualmente se utiliza un sistema desarrollado “hecho en casa” llamado CAT para el control de todos los programas realizados por el equipo de desarrolladores de software de la empresa. Estos programas son desarrollados, ya sea para complementar funcionalidad existente en las aplicaciones Oracle® utilizadas por la empresa, o para ayudar en la ejecución de tareas para las cuales no existe una función en las aplicaciones estándar de Oracle®. Casi todo el desarrollo de software se realiza utilizando Oracle Developer®, que es una herramienta CASE propiedad de Oracle Corporation.

Cuando surge la necesidad de desarrollar un nuevo programa, o dar mantenimiento a programas existentes, se utiliza el sistema CAT para capturar los requerimientos del nuevo programa, y asignar prioridades, programador, tiempos de entrega, etc. Así mismo, dentro de este sistema se identifican los objetos que serán necesarios modificar, o crear, para cubrir los requerimientos del usuario.

1.2. Planteamiento del problema.

CAT contiene el registro de todos los objetos que han sido utilizados en cada desarrollo desde que se inició con su utilización hace aproximadamente 6 años. Sin embargo, no se tiene en este momento una base de datos que contenga exactamente los objetos que forman parte de cada programa actualmente en operación. Esto significa que cada vez que un programa va a ser modificado se necesita revisar su código, plantillas, relaciones entre tablas de datos, etc., para poder identificar cuáles otros programas podrían ser afectados por dicha modificación. Sobre decir que una falla en la identificación de todos los programas afectados por el cambio en un objeto resulta en pruebas incompletas del nuevo programa, lo que a su vez lleva (en casos extremos), a enfrentar problemas importantes en la operación del sistema cuando se tiene ya trabajando en un ambiente de producción.

Lo anterior aplica para cualquier objeto desarrollado en casa, utilizando tecnologías de desarrollo Oracle. Pero también se presenta un problema similar cuando se tiene que efectuar una actualización de versión de las aplicaciones Oracle. Cuando la nueva versión de las aplicaciones implica cambios en tablas, vistas, programas de interface abierta (usados para interactuar entre las aplicaciones Oracle y los programas desarrollados en casa), etc., es necesario que el equipo de programadores identifique en cuáles programas “propios” se utilizan esos objetos, para poder adaptarlos a los objetos de la nueva versión. Tan sólo este ajuste a los programas desarrollados en casa puede representar hasta un año completo de trabajo del equipo de programadores, principalmente por el trabajo casi manual de identificar qué partes de qué programas deben ser actualizados.

Dado que todo el desarrollo se realiza utilizando Oracle Developer®, los archivos que contienen todos los objetos están en un formato especial, lo que ocasiona que tengan que ser analizados forzosamente mediante dicha herramienta CASE (figura 1.1).

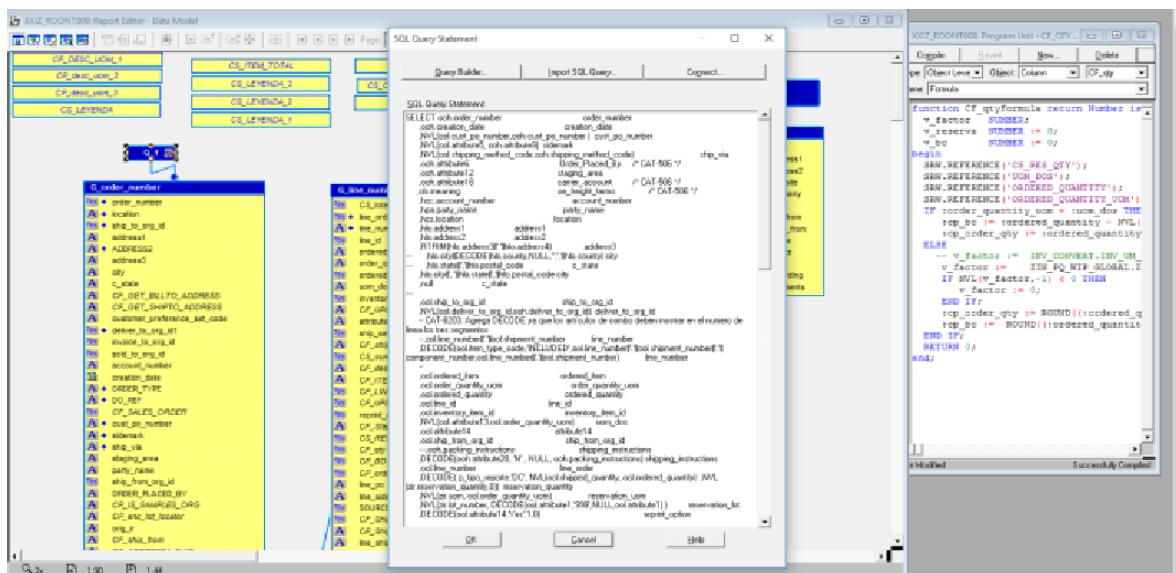


Figura 1.1. Reporte creado en Oracle Developer©.

Sin embargo, si se abre este mismo archivo desde un procesador de texto estándar, la mayor parte del contenido serán caracteres de control y textos que parecieran no tener sentido (figura 1.2):

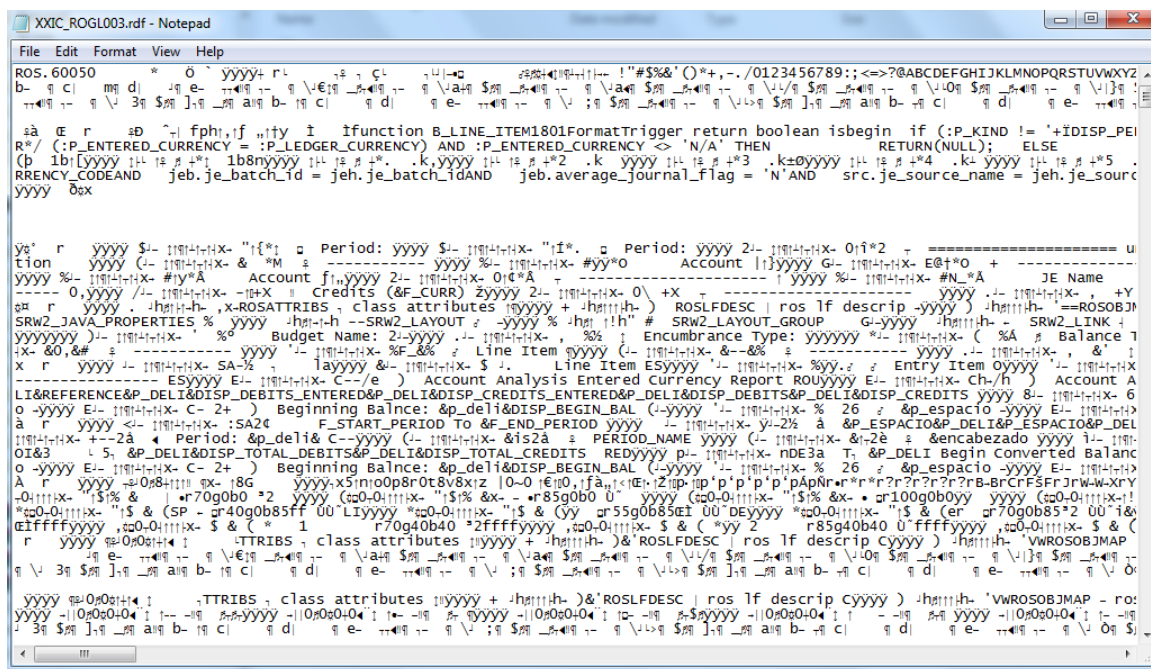


Figura 1.2. Reporte creado en Oracle Developer, editado desde Notepad.

La figura 1.3 muestra el entorno de desarrollo JDeveloper para crear una plantilla HTML.

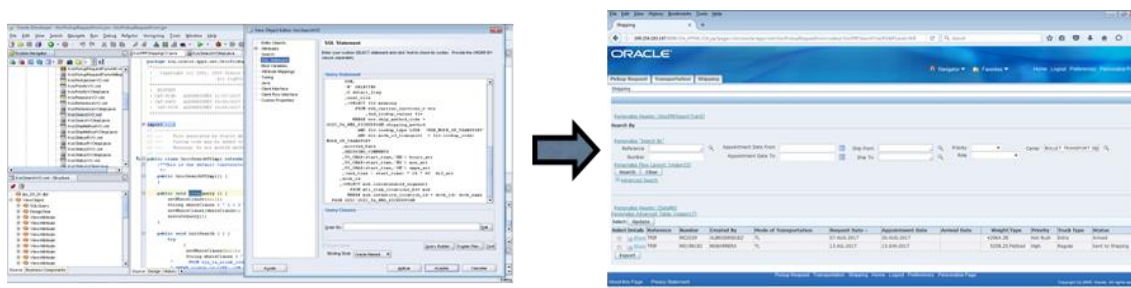


Figura 1.3. Plantilla html creada en Oracle JDeveloper.

Se necesita por lo tanto de alguna herramienta que permita analizar y determinar con certeza todos los objetos involucrados en:

- Un programa que será modificado, o en
- Un programa nuevo que requiera de objetos ya conocidos y que pueden ser reutilizados.

1.3. Hipótesis.

La correcta detección de los objetos involucrados en cualquier programa a ser creado o actualizado, permitirá al equipo de sistemas, analistas y programadores, completar correctamente la etapa de pruebas de calidad del programa, evitando, o disminuyendo de manera significativa, la posibilidad de encontrar errores en las diferentes etapas de desarrollo, pero sobre todo evitar que se presenten cuando el programa se encuentra corriendo en producción.

Para llegar a esta detección es necesario contar con una herramienta que permita encontrar dentro de cualquier objeto de desarrollo Oracle todos los objetos involucrados en un programa. Dado que estos objetos pueden ser librerías escritas en lenguaje java, programas PLSQL, plantillas, reportes o flujos de trabajo, será necesario que dicha herramienta sea

capaz de analizar archivos de cualquier tipo, ya que todos los objetos mencionados han sido creados utilizando Oracle Developer®, que es una herramienta CASE totalmente gráfica y por lo tanto los archivos que genera sólo son editables desde esta misma herramienta.

Se plantea por lo tanto que: Aplicando técnicas de Procesamiento de Lenguaje Natural (PLN) y Reconocimiento de Patrones se debe lograr la correcta detección de cualquier objeto de desarrollo Oracle utilizado dentro de cualquier programa y para ello se desarrollará el software denominado OBEX.

1.4. Alcances y Limitaciones.

Se busca el desarrollo de un programa que ejecute el análisis y búsqueda del objeto que va a ser modificado dentro de todo el universo de objetos de programación instalados en una instancia de aplicaciones Oracle. Dicho objeto de programación podrá ser un desarrollo propio o un desarrollo de Oracle, partiendo del entendido de que se desea encontrar en cuales desarrollos propios se utiliza determinado objeto de Oracle Applications.

1.5. Justificación.

Existe en este momento un software llamado “Panaya” (Panaya Corporate) que puede ser utilizado en proyectos de migración de versión de aplicaciones Oracle, ya que permite hasta cierto punto la identificación de los objetos que hayan sufrido algún cambio en la nueva versión, y que estén siendo utilizados en desarrollos personalizados de un cliente de Oracle; sin embargo, la información que puede obtenerse con este software aún necesita de un análisis detallado por parte del programador, su costo es muy elevado, y está restringido a únicamente la búsqueda de objetos que están siendo afectados por el cambio de una versión específica.

De hecho, este software ya se utilizó como guía para determinar los programas que serían afectados durante la última actualización de Oracle EBS que se llevó a cabo en la empresa

a finales del año 2015. Fue precisamente debido a este análisis extra que el equipo de programadores tuvo que realizar, que el proceso de actualización de códigos fuente para adaptarlos a la nueva versión de Oracle EBS®, consumió aproximadamente 3 a 4 meses más de lo estimado al inicio de ese proyecto.

Dado lo anterior, se puede decir que no existe en este momento en el mercado una herramienta que realice las funciones que se pretenden cubrir con el presente trabajo.

1.6. Objetivo.

- Desarrollar una aplicación de software que permita identificar si un objeto utilizado en un programa es utilizado a su vez en otros objetos de programación.

II. ESTADO DEL ARTE

2.1 Plataformas de entrega ágil empresarial.

En la actualidad se pueden encontrar herramientas que ofrecen ayuda para analizar el impacto de cambios y actualizaciones de versiones de los sistemas ERP comerciales que dominan el mercado, como por ejemplo Oracle© y SAP©.

Estas herramientas se identifican como “Plataformas de Entrega Ágil Empresarial” o “Enterprise Agile Delivery Platform” en inglés.

Una de estas herramientas, quizá de las más conocidas, es Panaya© (www.panaya.com). (Ver figura 2.1).

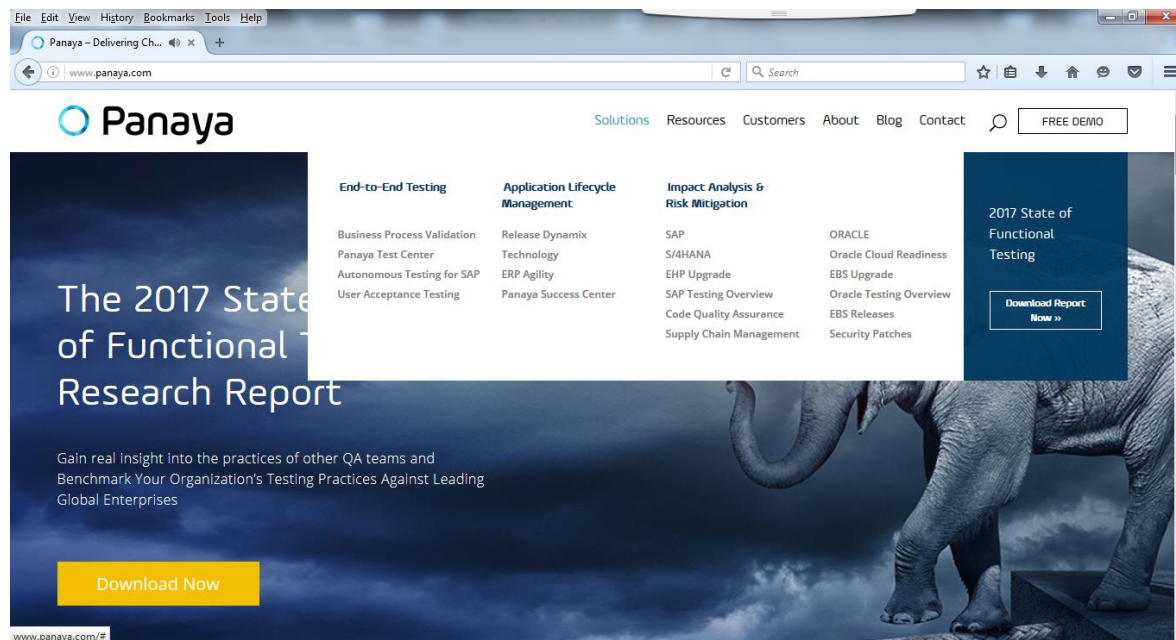


Figura 2.1. Página web de Panaya©.

Panaya© provee un enfoque centrado en procesos para el manejo del ciclo de vida de aplicaciones, con su producto *Panaya Release Dynamix* (RDx), una plataforma de entrega ágil empresarial basada en “la nube”, además de ofrecer valor de entregas continuas y ágiles

al mundo de aplicaciones empresariales desde el requerimiento inicial para hacer un cambio, pasando por las pruebas y hasta la puesta en producción,

Panaya Test Center habilita un enfoque centrado en procesos para el manejo de pruebas, permitiendo visibilidad en tiempo real del estado de requerimientos conforme son entregados. Impulsado por tecnologías de Aprendizaje de Máquina (Machine Learning), *Panaya Autonomous Testing* provee la automatización y aceleración de pruebas de regresión y pruebas de aceptación de usuario.

Panaya© utiliza análisis de código en tiempo real para aplicaciones SAP©, Oracle© y SFDC© (salesforce.com), y provee información útil de todas las transacciones impactadas para asegurar que aquellas áreas impactadas sean probadas y predecir el impacto completo de un cambio, antes de que el cambio se lleve a cabo.

La base de clientes de Panaya© se extiende a más de 1,600 empresas en todo el mundo, incluyendo alrededor de 220 de las 500 compañías listadas en Fortune 500©, con mercados que van desde servicios básicos, gas y petróleo, automotriz, farmacéutico, infraestructura y otros servicios.

Sin embargo, al menos hasta la versión de Panaya© utilizada por la empresa durante el último proceso de migración de Oracle EBS© a la versión 12.1.3, esta herramienta no generaba información con el nombre específico de la función, sino que únicamente reportaba el nombre del paquete afectado por la migración. Por lo tanto, los programadores tenían que llevar a cabo una revisión exhaustiva del código dentro del paquete reportado por Panaya© para determinar exactamente la función afectada por el cambio de versión.

III. Marco Teórico

3.1 Herramientas CASE

INGENIERÍA DE SOFTWARE ASISTIDA POR COMPUTADORAS (COMPUTER AIDED SOFTWARE ENGINEERING):

Son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero. Ver [Pressman, Roger S.].

3.2 Inteligencia Artificial (IA)

La IA es una disciplina académica relacionada con la computación cuyo objetivo es emular algunas de las facultades intelectuales humanas en sistemas artificiales. En la figura 3.1., se pueden observar algunos de los campos que abarca la IA. Ver a [Benítez, R., Escudero, G., Kanaan, S., & Masip Rodó, D.].

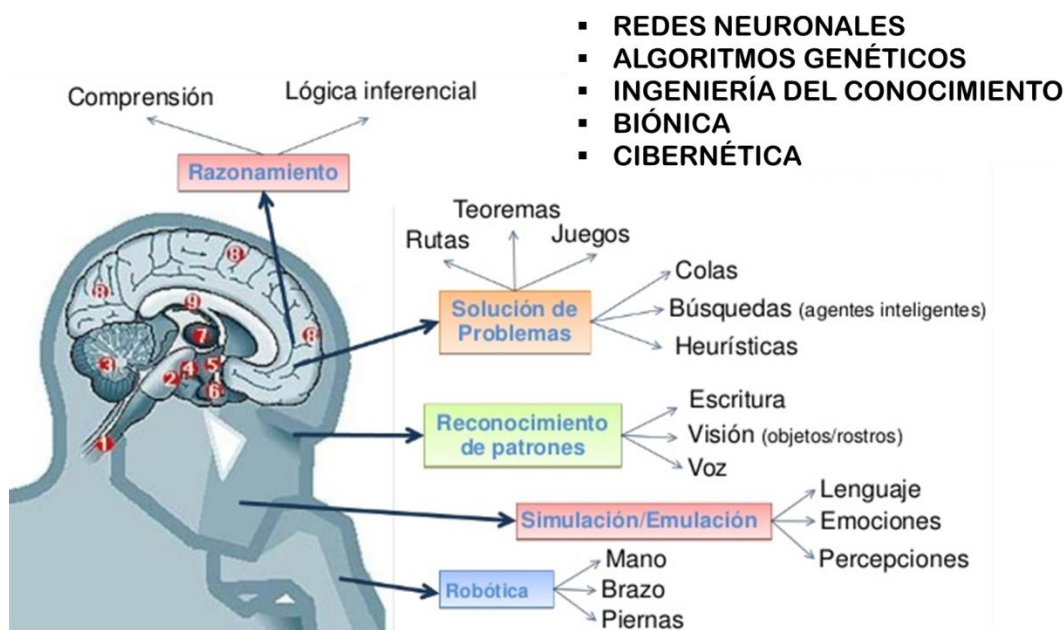


Figura 3.1. Campos de la Inteligencia Artificial.

3.3 Procesamiento de Lenguaje Natural

Nace como una sub-área de la IA y la Lingüística, con el objeto de estudiar los problemas derivados de la generación y comprensión automática del lenguaje natural. Ver [Mari Vallez y Rafael Pedraza-Jimenez.].

Permite el procesamiento de enormes cantidades de información en formato texto. Por ejemplo, en los motores de búsqueda web (ver figura 3.2), en las herramientas de traducción automática o en la generación automática de resúmenes.

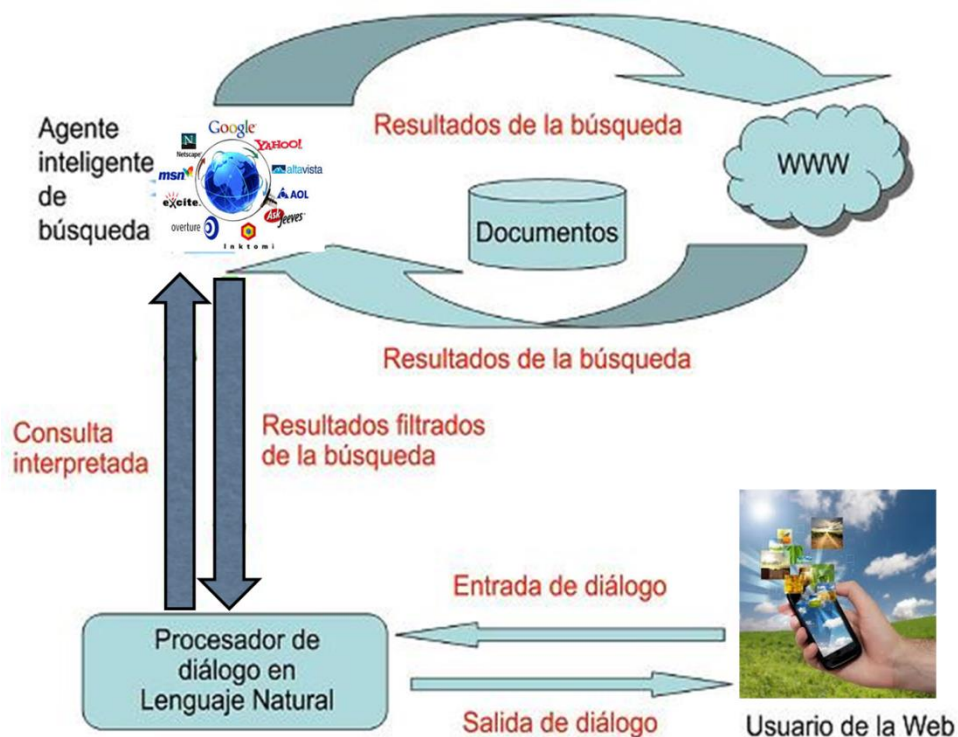


Figura 3.2. Procesamiento de Lenguaje Natural aplicado en búsquedas web.

3.4 Procesamiento de Lenguaje Natural

TOKENIZACIÓN

Regularmente las aplicaciones de PLN necesitan trabajar con texto normalizado, lo que significa realizar los siguientes pasos:

- 1.-Tokenización de las palabras en un texto.*
- 2.-Normalización del formato de las palabras en un texto.*
- 3.-Segmentación de las oraciones en un texto.*

A continuación, se definen dos conceptos estrechamente relacionados con la Tokenización:

- Tipo (Type): Un elemento del vocabulario, entendido éste como un conjunto de palabras.
- Token: una instancia de un Tipo en un texto dado.

Para más acerca del proceso de Tokenización ver [Roger Bilisoly] y [Hércules Antonio do Prado & Edilson Ferneda].

3.5 Sistema ERP

Un sistema de tipo “Enterprise Resource Planning” o ERP, por sus siglas en inglés, son sistemas de planeación de recursos empresariales, es decir, que son programas que se encargan de controlar todas las operaciones de una empresa, desde la producción hasta la distribución, incluyendo el manejo de recursos humanos. Al estar todas las áreas integradas en un solo sistema y plataforma, toda la información generada para la operación, análisis y planeación se apegan a los mismos estándares de calidad y certeza, facilitando el control y planeación de todos los recursos de una compañía.

3.6 Codificación UTF-8

UTF-8 (8-bit Unicode Transformation Format) es un formato de codificación de caracteres “Unicode” e “ISO 10646” utilizando símbolos de longitud variable (de 1 a 4 bytes por carácter Unicode). Este formato fue creado por Robert C. Pike y Kenneth L. Thompson, en 1992.

IV. Desarrollo

En este capítulo se ofrece una explicación a más detalle del método utilizado para resolver este problema, así como las pruebas realizadas al software resultante de este proceso.

4.1 OBEX: Proceso General

La figura 4.1 muestra el proceso general de OBEX. Las siguientes secciones explican cada paso de este proceso.

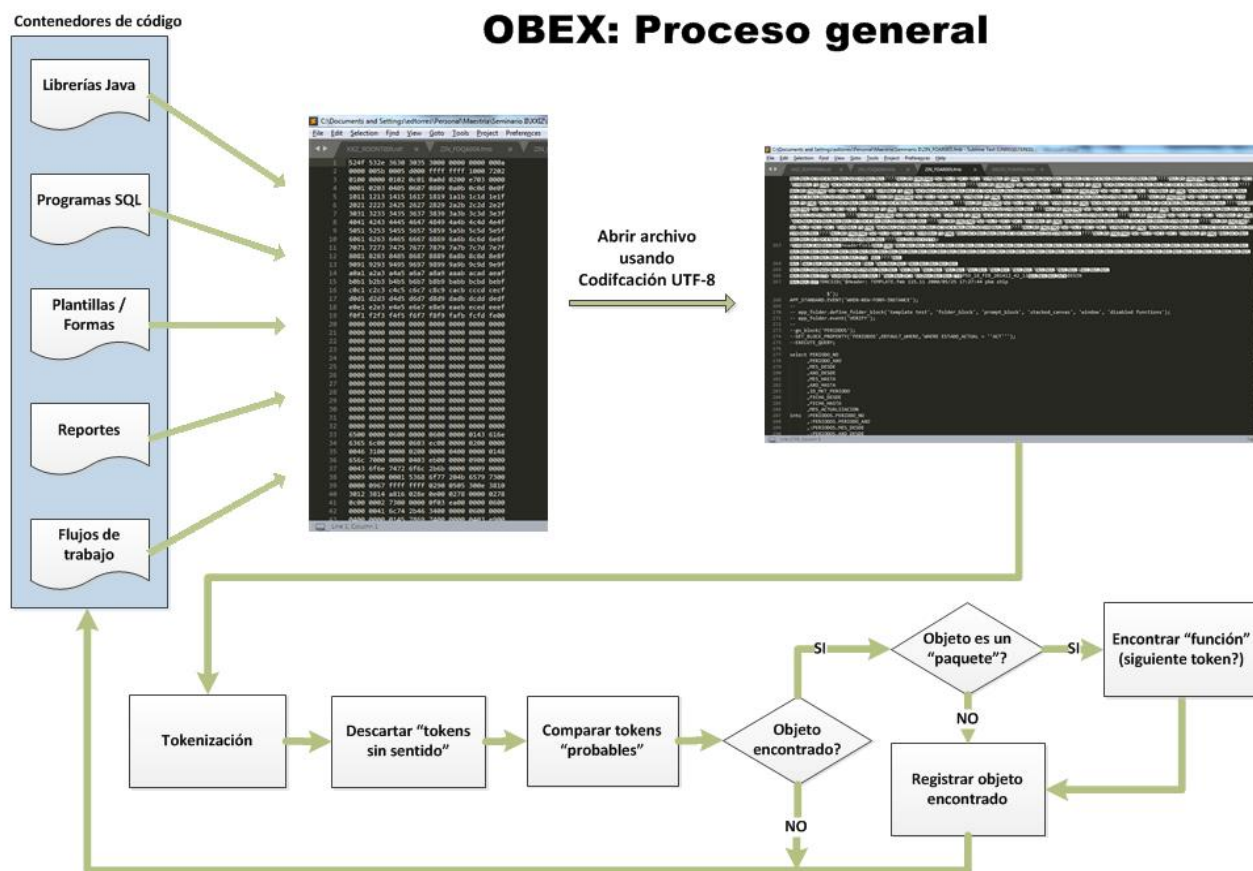


Figura 4.1. OBEX: Proceso general.

4.2 Contenedores de código

Como primer paso se identificaron los siguientes tipos de archivos contenedores de código de programación:

- **Librerías Java.** El ecosistema Java contiene numerosos elementos que lo conforman como una gran plataforma de desarrollo de software en todo el mundo. Se habla de Java como un lenguaje de programación, porque principalmente eso es para lo que está destinado, pero en su configuración contiene múltiples librerías, funciones, y herramientas que permiten desarrollar infinidad de programas de todo tipo.

Las librerías de Java consisten en diferentes conjuntos de clases que poseen una serie de atributos que facilitan muchas operaciones para los programadores, pues permiten reutilizar los códigos existentes. (Ver figura 4.2).

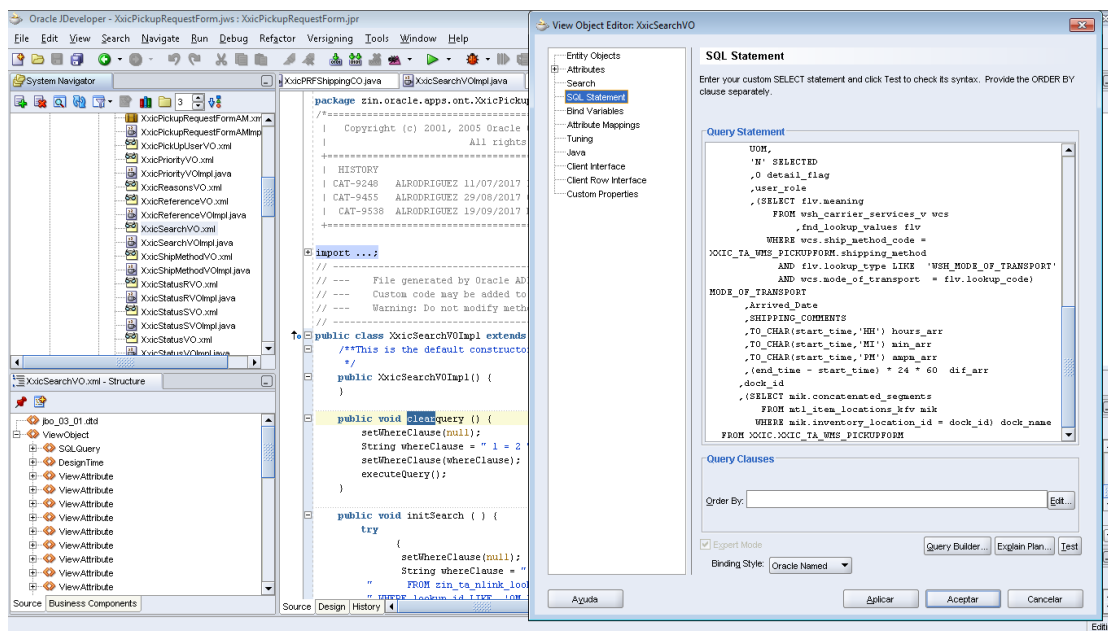


Figura 4.2. Ejemplo de programación Java utilizando Oracle JDeveloper®.

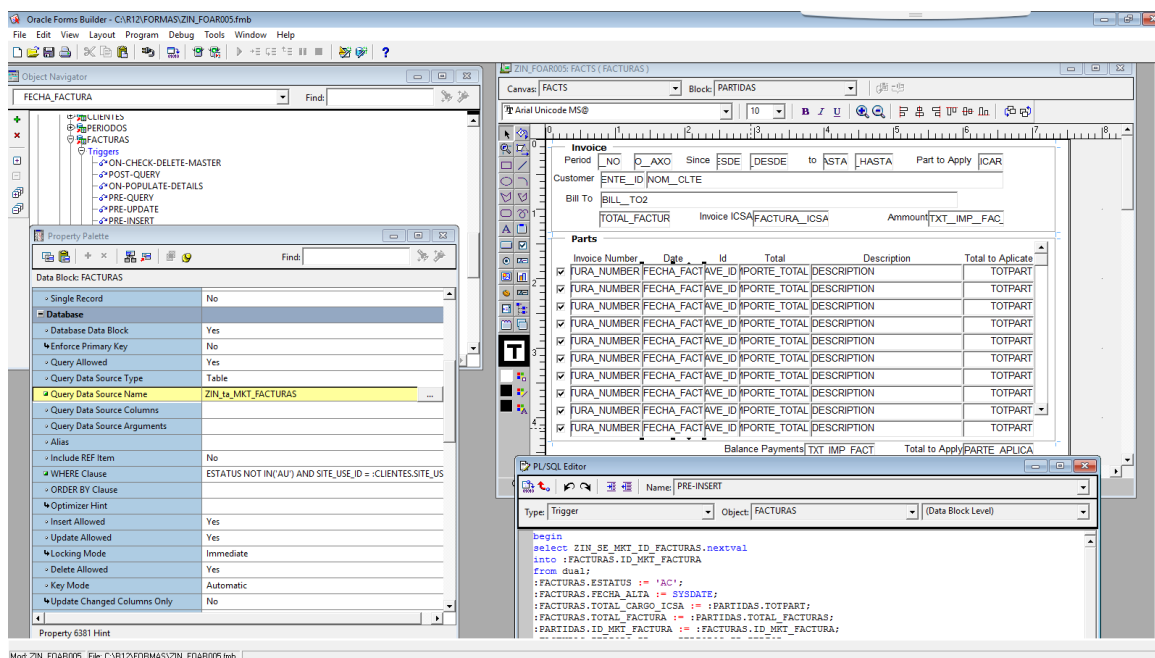


Figura 4.4. Ejemplo de plantilla programada en Oracle Forms Builder®.

- **Reportes (Reports).** Típicamente un reporte representa la manera más normal de presentar información guardada en la base de datos, de manera que pueda ser visualizada y analizada por el usuario. En el entorno de Oracle EBS® un reporte puede ser generado para ser analizado desde una hoja de cálculo en MS Excel®, como un archivo de texto, una impresión en papel o un documento con formato XML; es la manera más interactiva de comunicación entre el usuario y Oracle EBS. La figura 4.5 muestra un ejemplo del cómo se programa un reporte usando Oracle Reports Builder®.

Company: scf entidad legal
Document: Picking/Packing list **** RE-PRINT **** Picking/Packing list
Customer #: F account number1
Customer name: F party_name4
Bill-To: F party_name3
F_ADDRESS1
Ship-To: F party_name2
F_ADDRESS1
Deliver-To: F party_name1
F_address_del
Special instructions:
F_staging_area1
P_CREDIT_MSG
Line # Product sku Product description Quantity Ordered Quantity B/Q
order F ordered item lot_locator F ordered_quantity4 ordered_quantity3 ordered qu
F pick

Figura 4.5. Ejemplo de reporte programado en Oracle Reports Builder©.

- *Flujos de trabajo (Workflows)*. Son programas que controlan la secuencia de un proceso dentro del sistema, y que permiten al usuario “responder” a solicitudes originadas por otro usuario, por ejemplo para la autorización de una orden de compra, una orden para devolución de material a una tienda, etc. La figura 4.6 muestra un ejemplo de cómo es la programación de un “workflow” en Oracle© mediante la herramienta Oracle Workflow Builder©.

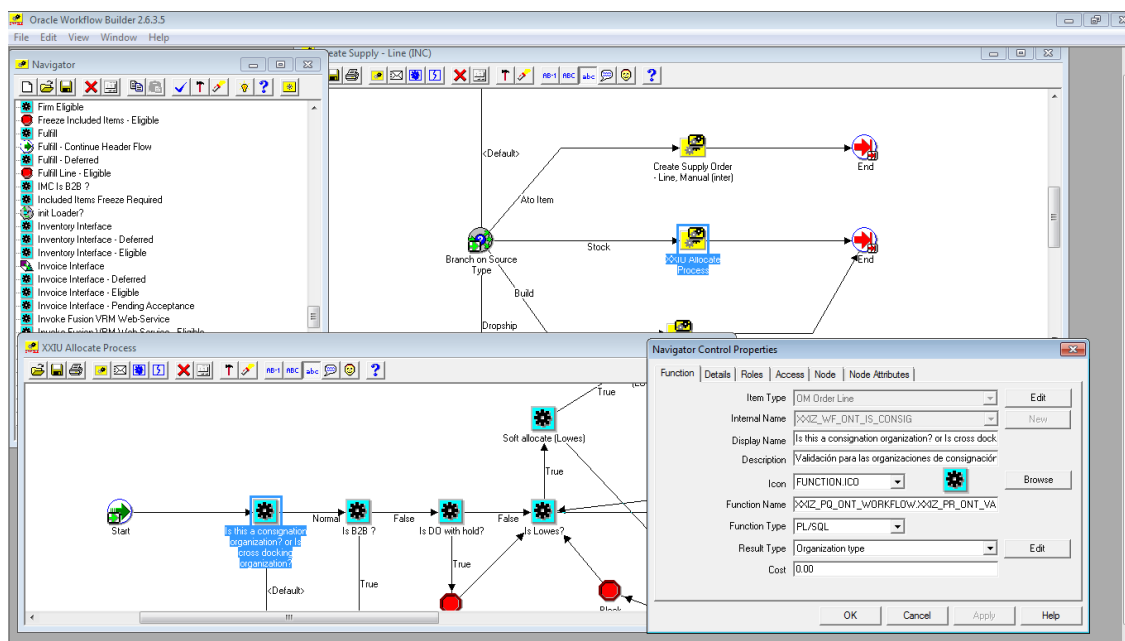


Figura 4.6. Ejemplo de “Workflow” programado en Oracle Workflow Builder©.

Cada uno de estos contenedores son archivos que resultan ilegibles a simple vista utilizando cualquier editor de texto. Por lo tanto, el siguiente paso es abrir estos con una codificación UTF-8. Esta codificación permite obtener una versión más “accesible” del mismo archivo (ver figura 4.7).

```

597
598 srw.user_exit('FND SRWEXIT');
599 return (TRUE);
600 end;
601 T????<0x00><0x05><0x05><0x0e>8<0x10>0<0x12>0<0x13>?<0x16><0x00><0x1e><0x08><0x00><0x1
<0x00><0x01>0<0x00>????<0x01>?<0x05><0x05><0x0e>8<0x10>0<0x12>8<0x14>?<0x16><0x01>?<0
return boolean is          v_email VARCHAR2(500);
602
603
604 begin
605   SRW.USER_EXIT( 'FND SRWINIT' );
606
607   IF :p_email IS NOT NULL THEN
608     srw.message(201,':p_email '||:p_email );
609     v_email := :p_email;
610
611     ZIN_PQ_INT_REQUEST_MAIL.ZIN_PR_INT_LANZA_REQUEST (FND_GLOBAL.CONC_REQUEST_ID, v_em
612   END IF;
613
614   SRW.USER_EXIT( 'FND SRWEXIT' );
615   return (TRUE);
616 end;????<0x00><0x1c><0x05><0x05><0x0e>8<0x10>0<0x12>0<0x13>?<0x16><0x00><0x1a>
617 <0x00><0x0f>?<0x01><0x04><0x00><0x00><0x00><0x01>?<0x00>????<0x00><0x1c><0x05><0x05>0<0x0
<0x00><0x0f>?<0x01><0x04><0x00><0x00><0x00><0x01>p<0x00>????<0x00>?<0x07><0x1b>?:?N?V?^<0
?<0x00>?<0x00><0x10>SQL*ReportWriter<0x00><0x00><0x00><0x06>REPORT<0x00><0x04>APPS<0x00><
618 XXIC_ROONT003<0x00><0x01><0x00><0x00><0x00><0x01>?<0x18>?<0x00><0x04>APPSxu<0x07><0x14><0
PPSxu<0x08><0x0b><0x1c>?<0x00>?<0x00><0x02>
619 <0x10><0x00>
620 <0x10><0x00>

```

Figura 4.7. Reporte creado en Oracle Developer©, editado desde Sublime Text© usando Codificación UTF-8.

4.3 Procesamiento de Lenguaje Natural y Tokenización

De acuerdo con [Mary Vázquez y Rafael Pedraza-Jiménez], en “El Procesamiento del Lenguaje Natural en la Recuperación de Información Textual y áreas afines”, el Procesamiento de Lenguaje Natural (PLN) “Nace como una sub-área de la Inteligencia Artificial y la Lingüística, con el objeto de estudiar los problemas derivados de la generación y comprensión automática del lenguaje natural”. Permite el procesamiento de enormes cantidades de información en -- formato texto, por ejemplo, en los motores de búsqueda web, en las herramientas de traducción automática o en la generación automática de resúmenes.

Una de las técnicas más comunes del PLN es la tokenización o “tokenization” en idioma inglés. La página [del Procesamiento del Lenguaje Natural de la Universidad de Stanford en Estados Unidos] habla acerca de la Tokenización, lo siguiente: «Dada una sentencia de texto, la tokenización es el proceso de romper dicha sentencia en trozos llamados “tokens”, pero al mismo tiempo descartar ciertos caracteres, como por ejemplo los signos de puntuación. Estos tokens a menudo se denominan “términos” o “palabras”, pero algunas veces es necesario hacer la distinción entre Token, Tipo y Término: Un Token es una instancia de una secuencia de caracteres, en algún documento en particular, y que son agrupados juntos como una unidad semántica útil para su procesamiento. Un Tipo es la clase de todos los tokens que contienen la misma secuencia de caracteres y un Término es un tipo (quizás normalizado) que se incluye en el diccionario del Sistema de Recuperación de Información (SRI)». Para obtener una clasificación de los SRI ver a [Mohamed Zakaria Kurdi].

OBEX utiliza la librería de código abierto “*java.util.StringTokenizer*” para la manipulación y extracción de tokens del texto resultando una vez que el archivo es abierto en codificación UTF-8. La figura 4.8 muestra la porción de código Java utilizada para este propósito:


```
try {
    BufferedReader entradaArchivo=new BufferedReader(new InputStreamReader(new FileInputStream(fileName),"UTF-8"));
    String linea;
    String w="";
    String tokenant="";
    String tokensig="";
    Boolean encontrado;
    while ((linea = entradaArchivo.readLine()) != null) {
        //linea = entradaArchivo.readLine();
        StringTokenizer tokens = new StringTokenizer( linea,"\n,;. " );
        while (tokens.hasMoreTokens()) {
            tokenant = w;
            w = tokens.nextToken();
        }
    }
}
```

Figura 4.8. Código Java para abrir un archivo usando codificación UTF-8 y extraer los tokens.

4.4 Detección de objetos

La solución propuesta para este problema se basa en la aplicación del concepto de tokenización del texto resultante al abrir un archivo con codificación UTF-8.

Este proceso de tokenización se realiza mediante un programa desarrollado en lenguaje Java, que permite la utilización de librerías de código abierto para el tratamiento de textos.

Cada una de las “palabras” o de los “tokens” extraídos del texto son analizados y comparados contra una base de datos, en la cual se tiene almacenados todos los objetos utilizados en el sistema ERP de la empresa. Sin embargo, no todos los tokens extraídos son sujetos a análisis; únicamente aquellos que pudieran tener algún sentido son comparados en la base de datos. Esta selección se basa inicialmente en el descarte de todos los tokens que contienen caracteres de control, y otros que por ser ilegibles no podrían ser parte del nombre de un objeto que pudiera ser utilizado en un entorno de desarrollo Oracle Developer©. La figura 4.9 muestra una parte del código Java utilizado para la detección y eliminación de estos caracteres, haciendo uso de la librería de código abierto “*java.util.StringTokenizer*”:

```
// strips off all non-ASCII characters|
w = w.replaceAll("[^\\x00-\\x7F]", "");

// erases all the ASCII control characters
w = w.replaceAll("[\\p{Cntrl}&&[^\r\n\t]]", "");

// removes non-printable characters from Unicode
w = w.replaceAll("\\p{C}", "");
```

Figura 4.9. Código Java para detección y eliminación de caracteres de control y otros ilegibles.

Para más acerca del proceso de Tokenización ver [Roger Bilisoly] y [Hércules Antonio do Prado & Edilson Ferneda].

Finalmente, si un token se corresponde con el nombre de un objeto (tabla, vista, paquete, función, “trigger”, etc.) es reportado como parte del archivo que está siendo analizado, pero si un objeto encontrado, es de tipo “Paquete”, se deberá analizar el siguiente token para determinar si el texto cumple con las características para considerarse como el nombre de la función específica que está siendo utilizada dentro de este paquete. La figura 4.10 muestra el código Java utilizado para este fin:

```
w = w.toLowerCase();
tokensig = w;
if("PACKAGE".equals(objtipo.trim())){
    objtablealias = tokensig;
    objtablealias = objtablealias.toLowerCase();
    System.out.println(objowner + "|" + objbuscado + "." + objtablealias + "|" + "PACKAGE&FUNCTION|" + fileName.getName());
    objtipo = "";
}
encontrado = false;
if (w.length()>1){
    for (int j=0; j<objetos.length;j++){
        objbuscado = objetos[j][0];
        objtipo = objetos[j][1];
        objowner = objetos[j][2];
        if (objbuscado.length()>1){
            if (w.equals(objbuscado.toLowerCase())){
                //printout object name, object type and file name where object was found:
                System.out.println(objowner + "|" + objbuscado + "|" + objtipo.trim() + "|" + fileName.getName() );
                cont++;
                encontrado = true;
                jTextField1.setText(objbuscado);
                jTextField1.update(jTextField1.getGraphics());
            }
            if (encontrado) {
                break;
            }
        }
    }
}
if (!encontrado) {
    objtipo = "";
}
}
```

Figura 4.10. Código Java para detección de objetos dentro del texto analizado.

Este proceso se repite por cada token extraído de cada archivo que forme parte del universo de objetos de todo el sistema ERP (ver nuevamente la figura 4.1).

4.5 Ejecución de pruebas

Todas las ejecuciones del OBEX, tanto para pruebas como para la obtención de la información que finalmente se tomó como resultado válido para ser utilizado por los programadores, se hicieron desde NetBeans IDE 8.2, el cual es un el ambiente de desarrollo para lenguaje Java de código abierto muy utilizado a nivel mundial.

Durante la primera etapa de pruebas no se tenía acceso a la base de datos de Oracle, por lo que las tablas con la información de los objetos contenidos en el ERP eran cargadas en una base de datos en MySQL a partir de hojas de cálculo de MS Excel©. Debido a esto, el primer paso antes de ejecutar el programa era activar el servicio Apache Derby Network Server, utilizado como el servicio para acceder a la base de datos de MySQL. La figura 4.11 muestra este paso:

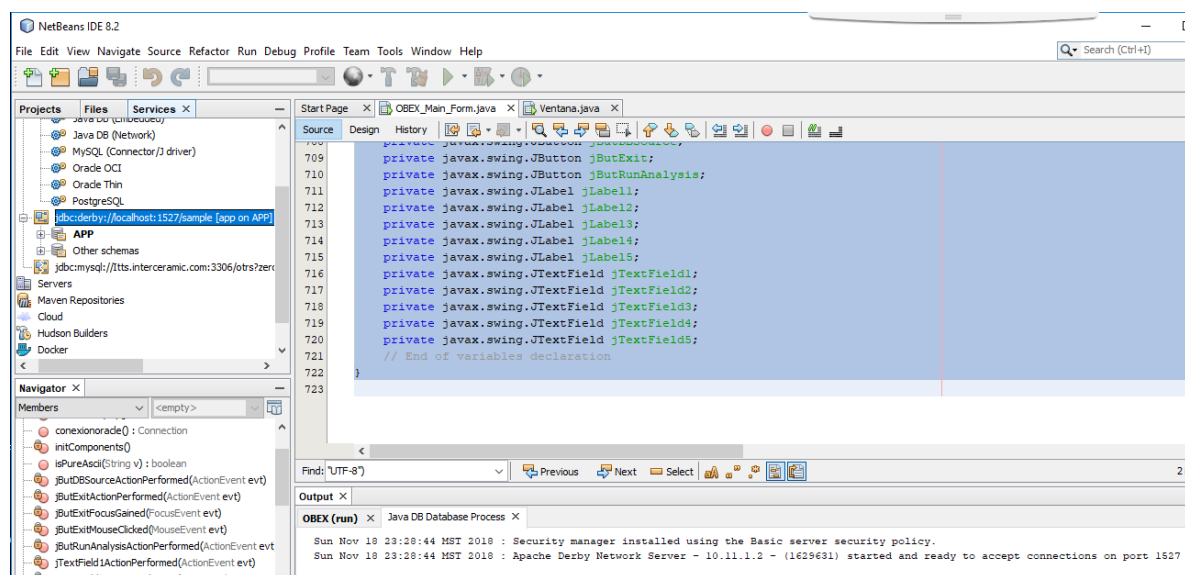


Figura 4.11. Activación de Apache Derby Network Server desde NetBeans IDE 8.2.

Una vez habilitado el servicio para la base de datos, se procedía con la ejecución del programa. Desde ese momento la interfaz de usuario se mantuvo muy simple: ofrece la opción de lanzar la

búsqueda de objetos dentro de reportes, plantillas y archivos XML, o dentro de objetos de programación contenidos en la misma base de datos Oracle®. El resto de los campos en la ventana de ejecución muestran el estatus de la búsqueda. La figura 4.12 muestra esta ventana:

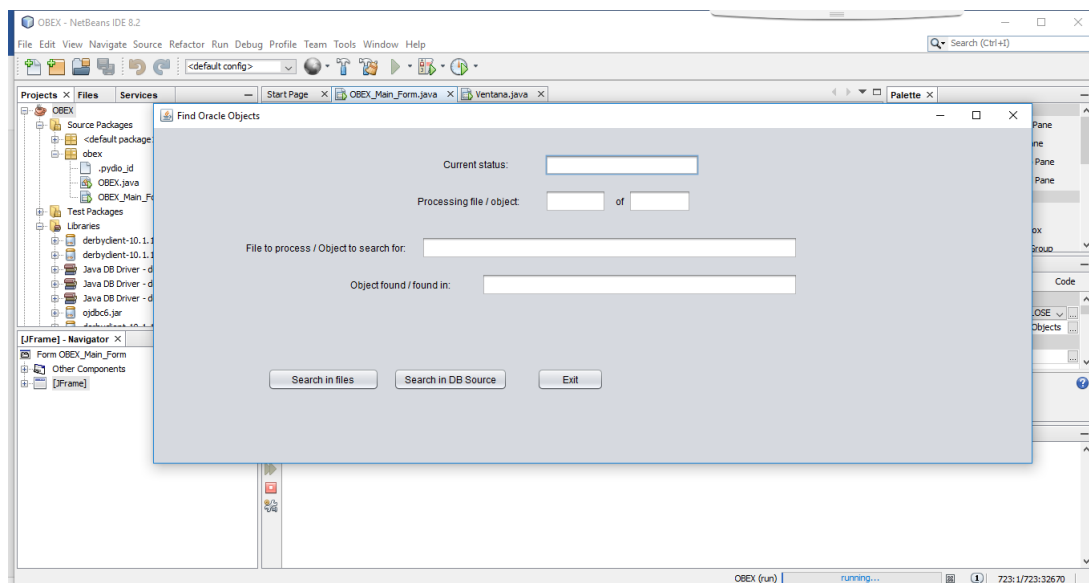


Figura 4.12. Ventana de ejecución de OBEX.

Eventualmente se obtuvieron los permisos para acceder directamente a las tablas en la base de datos Oracle® donde se guarda la información de los objetos utilizados por el sistema ERP. Para este propósito se utilizó a partir de ese momento el cliente “JDBC Thin driver” (ojdbc6.jar and ojdbc5.jar). La figura 4.13 muestra el código Java para la conexión a la base de datos Oracle.

```
Connection connoracle = DriverManager.getConnection
("jdbc:oracle:thin:APPS@lddebs04.interceramic.com:2530/CRP6", "*****", "*****");

// @TNSNames Entry, userid, password

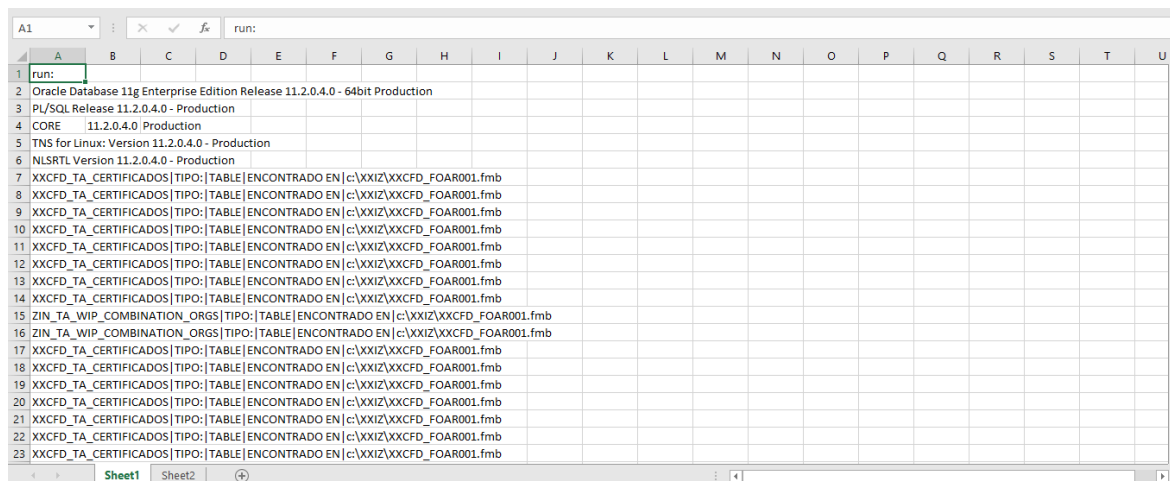
// el código original contiene el usuario y password para conectar a la base de datos, sin embargo

// en este documento se oculta por razones de seguridad.
```

Figura 4.13. Código Java para la conexión a la base de datos Oracle.

Lo primero que el programa hace es establecer la conexión con la base de datos de Oracle EBS®, ya que ahí se guarda toda la información acerca de los objetos utilizados por el ERP, tanto

Este archivo de texto puede abrirse desde una hoja de cálculo, como MS Excel© para facilitar su revisión. (Ver figura 4.16).



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	run:																				
2	Oracle Database 11g Enterprise Edition Release 11.2.0.4.0 - 64bit Production																				
3	PL/SQL Release 11.2.0.4.0 - Production																				
4	CORE 11.2.0.4.0 Production																				
5	TNS for Linux: Version 11.2.0.4.0 - Production																				
6	NLSRTL Version 11.2.0.4.0 - Production																				
7	XXCFD_TA_CERTIFICADOS TIPO: TABLE ENCONTRADO EN c:\XXIZ\XXCFD_FOAR001.fmb																				
8	XXCFD_TA_CERTIFICADOS TIPO: TABLE ENCONTRADO EN c:\XXIZ\XXCFD_FOAR001.fmb																				
9	XXCFD_TA_CERTIFICADOS TIPO: TABLE ENCONTRADO EN c:\XXIZ\XXCFD_FOAR001.fmb																				
10	XXCFD_TA_CERTIFICADOS TIPO: TABLE ENCONTRADO EN c:\XXIZ\XXCFD_FOAR001.fmb																				
11	XXCFD_TA_CERTIFICADOS TIPO: TABLE ENCONTRADO EN c:\XXIZ\XXCFD_FOAR001.fmb																				
12	XXCFD_TA_CERTIFICADOS TIPO: TABLE ENCONTRADO EN c:\XXIZ\XXCFD_FOAR001.fmb																				
13	XXCFD_TA_CERTIFICADOS TIPO: TABLE ENCONTRADO EN c:\XXIZ\XXCFD_FOAR001.fmb																				
14	XXCFD_TA_CERTIFICADOS TIPO: TABLE ENCONTRADO EN c:\XXIZ\XXCFD_FOAR001.fmb																				
15	ZIN_TA_WIP_COMBINATION_ORGS TIPO: TABLE ENCONTRADO EN c:\XXIZ\XXCFD_FOAR001.fmb																				
16	ZIN_TA_WIP_COMBINATION_ORGS TIPO: TABLE ENCONTRADO EN c:\XXIZ\XXCFD_FOAR001.fmb																				
17	XXCFD_TA_CERTIFICADOS TIPO: TABLE ENCONTRADO EN c:\XXIZ\XXCFD_FOAR001.fmb																				
18	XXCFD_TA_CERTIFICADOS TIPO: TABLE ENCONTRADO EN c:\XXIZ\XXCFD_FOAR001.fmb																				
19	XXCFD_TA_CERTIFICADOS TIPO: TABLE ENCONTRADO EN c:\XXIZ\XXCFD_FOAR001.fmb																				
20	XXCFD_TA_CERTIFICADOS TIPO: TABLE ENCONTRADO EN c:\XXIZ\XXCFD_FOAR001.fmb																				
21	XXCFD_TA_CERTIFICADOS TIPO: TABLE ENCONTRADO EN c:\XXIZ\XXCFD_FOAR001.fmb																				
22	XXCFD_TA_CERTIFICADOS TIPO: TABLE ENCONTRADO EN c:\XXIZ\XXCFD_FOAR001.fmb																				
23	XXCFD_TA_CERTIFICADOS TIPO: TABLE ENCONTRADO EN c:\XXIZ\XXCFD_FOAR001.fmb																				

Figura 4.16. Archivo generado por OBEX, abierto desde MS Excel©.

V. Resultados.

Hasta el momento de la presentación de este trabajo, se han llevado a cabo pruebas con el total de objetos y archivos contenidos en la base de datos del ERP utilizado en la empresa.

El resultado de dichas pruebas ha sido revisado y validado por el equipo de desarrollo de software, tomando de manera aleatoria archivos de reportes y plantillas, y realizando una inspección visual dentro de Oracle Developer© del código contenido en los archivos y comparando lo que ellos han encontrado contra los resultados generados por OBEX.

La figura 5.1 muestra los parámetros de ejecución utilizados durante la ejecución final de OBEX, mientras que en la figura 5.2 podemos apreciar (abiertos desde MS Excel©) algunos de los datos obtenidos a partir de esta ejecución.

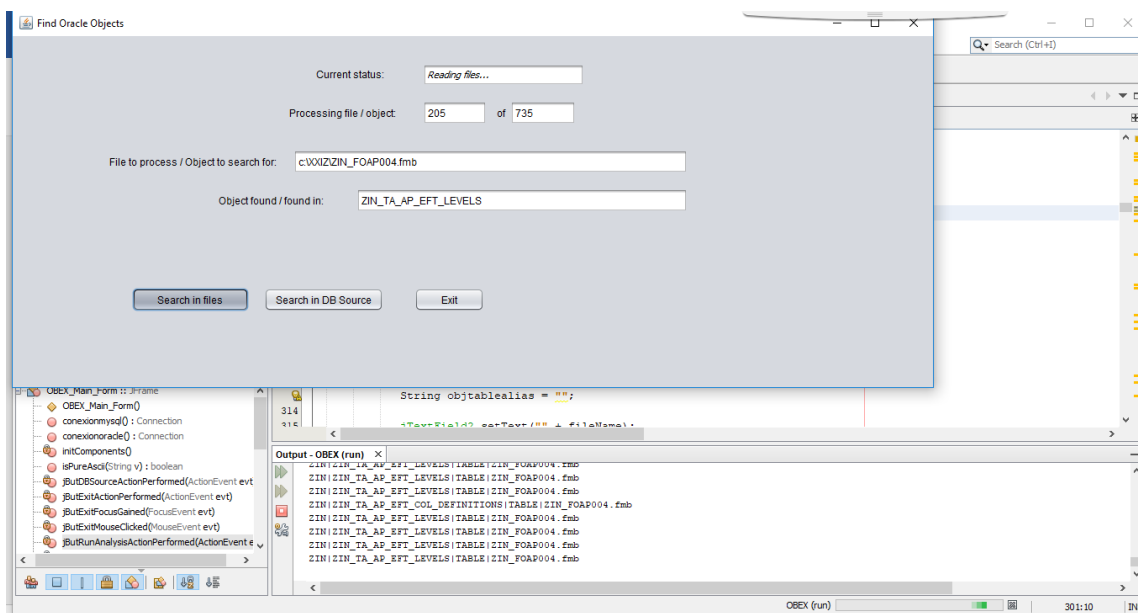


Figura 5.1. Ejecución de OBEX sobre archivos de plantillas y reportes.

Objetos Reports y Forms al 31Aug18 - Excel

	A	B	C	D
1	Owner	Object name	Type	Found in:
2	XXCFD	XXCFD_TA_CERTIFICADOS	TABLE	XXCFD_FOAR001.fmb
3	ZIN	ZIN_TA_WIP_COMBINATION_ORGS	TABLE	XXCFD_FOAR001.fmb
4	XXCFD	XXCFD_TA_AR_CFD	TABLE	XXCFD_FOAR002.fmb
5	XXCFD	XXCFD_TA_PARAMETROS	TABLE	XXCFD_FOAR002.fmb
6	APPS	XXCFD_PQ_AR_REPORTES	PACKAGE	XXCFD_FOAR002.fmb
7	APPS	XXCFD_PQ_AR_REPORTES.xxcf_ar_file_exists	PACKAGE&FUNCTION	XXCFD_FOAR002.fmb
8	APPS	XXCFD_PQ_AR_REPORTES.xxcf_ar_gen_file_temp	PACKAGE&FUNCTION	XXCFD_FOAR002.fmb
9	APPS	ZIN_PQ_AR_SUBMIT_INV_PRNT_PKG	PACKAGE	XXCFD_FOAR002.fmb
10	APPS	ZIN_PQ_AR_SUBMIT_INV_PRNT_PKG.zin_pr_ar_sub_prtg_prgm	PACKAGE&FUNCTION	XXCFD_FOAR002.fmb
11	APPS	ZIN_PQ_AR_CFD_PKG	PACKAGE	XXCFD_FOAR002.fmb
12	APPS	ZIN_PQ_AR_CFD_PKG.zin_pr_ar_delete_row	PACKAGE&FUNCTION	XXCFD_FOAR002.fmb
13	ZIN	ZIN_TA_NLINK_LOOKUP	TABLE	XXCFD_FOAR002.fmb
14	XXCFD	XXCFD_TA_AR_CFD	TABLE	XXCFD_FOAR003.fmb
15	XXCFD	XXCFD_TA_PARAMETROS	TABLE	XXCFD_FOAR003.fmb
16	APPS	XXCFD_PQ_AR_REPORTES	PACKAGE	XXCFD_FOAR003.fmb
17	APPS	XXCFD_PQ_AR_REPORTES.xxcf_ar_file_exists	PACKAGE&FUNCTION	XXCFD_FOAR003.fmb
18	APPS	XXCFD_PQ_AR_REPORTES.xxcf_ar_gen_file_temp	PACKAGE&FUNCTION	XXCFD_FOAR003.fmb
19	APPS	XXCFD_PQ_AR_REPORTES.xs97	PACKAGE&FUNCTION	XXCFD_FOAR003.fmb
20	APPS	ZIN_PQ_AR_RCPT_PRNT_PKG	PACKAGE	XXCFD_FOAR003.fmb
21	APPS	ZIN_PQ_AR_RCPT_PRNT_PKG.zin_pr_ar_sub_prtg_prgm	PACKAGE&FUNCTION	XXCFD_FOAR003.fmb
22	APPS	ZIN_PQ_AR_RCPT_PRNT_PKG.xs5	PACKAGE&FUNCTION	XXCFD_FOAR003.fmb
23	APPS	ZIN_PQ_AR_RCPT_PRNT_PKG.fnd	PACKAGE&FUNCTION	XXCFD_FOAR003.fmb

Objects fnd_concurrent_programs fnd_form Sheet5

Figura 5.2. Resultado en MS Excel© de la ejecución de OBEX sobre archivos de plantillas y reportes.

Como comentario adicional, la última ejecución del programa realizada para obtener los datos que finalmente serán utilizados por el equipo de programadores, concluyó en 88 minutos, como se muestra en la figura 5.3.

VI. Conclusiones.

Debido a que la herramienta comercial que se utilizaba antes de desarrollar OBEX no proporcionaba toda la información necesaria para dar soporte al equipo de desarrolladores de la empresa, aunado al hecho de que todas las herramientas usadas en la generación OBEX son de código abierto, se logró una solución eficiente a un costo significativamente menor al de la renta anual de Panaya©.

Debido a cláusulas de confidencialidad con el proveedor del software comercial utilizado durante el proceso de migración del ERP a la versión de Oracle EBS© que actualmente se utiliza en la empresa, no es posible ofrecer un monto exacto o aproximando del ahorro monetario que significa no tener que utilizarlo nuevamente para identificar los objetos de programación personalizados usados en el sistema. Sin embargo, se puede establecer que el monto de este ahorro se sitúa en el orden de las decenas de miles de dólares, aún sin considerar el tiempo y recursos de personal (programadores) que ya no tendrán que dedicarse a la identificación exacta de las funciones dentro de los paquetes de código, porque esta información es proporcionada ya por OBEX.

Finalmente, el presente trabajo muestra que la aplicación de una de las técnicas más comunes y sencillas del Procesamiento de Lenguaje Natural, como lo es la Tokenización, permitió encontrar una manera de analizar archivos generados por una herramienta CASE comercial sin tener que revisarlos uno a uno desde esa misma herramienta.

VII. Bibliografía.

Benítez, R., Escudero, G., Kanaan, S., & Masip Rodó, D. (2013). Inteligencia Artificial Avanzada. Barcelona: Editorial UOC.

Pressman, Roger S. Ingeniería de Software, un enfoque práctico. Quinta edición. S.I. : McGraw-Hill Companies, 2002. ISBN: 8448132149

Mari Váñez y Rafael Pedraza-Jiménez. El Procesamiento del Lenguaje Natural en la Recuperación de Información Textual y áreas afines [en línea]. "Hipertext.net", núm. 5, 2007. <https://www.upf.edu/hipertextnet/numero-5/pln.html>

Roger Bilisoly. Practical Text Mining With Perl. United States of America, (2008). John Wiley & Sons, Inc.

Hércules Antonio do Prado & Edilson Ferneda. Emerging Technologies of Text Mining: Techniques and Applications. United States of America, (2008). Information Science Reference (an imprint of IGI Global)

Mohamed Zakaria Kurdi. Natural Language Processing and Computational Linguistics 2. United States of America, (2017). John Wiley & Sons, Inc.

Stanford University, Natural Language Processing, Tokenization. <https://nlp.stanford.edu/IR-book/html/htmledition/tokenization-1.html>

ANEXOS

Código fuente OBEX.

/*

/*

* To change this license header, choose License Headers in Project Properties.

* To change this template file, choose Tools | Templates

* and open the template in the editor.

*/

/**

*

* @author Eduardo Torres Avila

*/

```
package obex;
```

```
import java.util.Arrays;
```

```
import java.util.StringTokenizer;
```

```
import java.io.*;
```

```
import java.net.*;
```

```
import java.sql.Connection;
```

```
import java.sql.DriverManager;
```

```
import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

import java.text.SimpleDateFormat;

import java.util.Calendar;

import java.util.Date;

import java.util.Scanner;

import java.util.logging.Level;

import java.util.logging.Logger;

import java.nio.ByteBuffer;

import java.nio.CharBuffer;

import java.nio.charset.Charset;

import java.nio.charset.CharsetDecoder;

import java.nio.charset.CharacterCodingException;


public class OBEX_Main_Form extends javax.swing.JFrame {


    /**
     * Creates new form OBEX_Main_Form
     */

    public OBEX_Main_Form() {
```

```
initComponents();  
  
}  
  
/**  
  
 * This method is called from within the constructor to initialize the form.  
 * WARNING: Do NOT modify this code. The content of this method is always  
 * regenerated by the Form Editor.  
 */  
  
@SuppressWarnings("unchecked")  
// <editor-fold defaultstate="collapsed" desc="Generated Code">  
private void initComponents() {  
  
    jButtonRunAnalysis = new javax.swing.JButton();  
  
    jTextField1 = new javax.swing.JTextField();  
  
    jLabel1 = new javax.swing.JLabel();  
  
    jTextField2 = new javax.swing.JTextField();  
  
    jTextField3 = new javax.swing.JTextField();  
  
    jTextField4 = new javax.swing.JTextField();  
  
    jLabel2 = new javax.swing.JLabel();  
  
    jLabel3 = new javax.swing.JLabel();  
  
    jLabel4 = new javax.swing.JLabel();  

```

```
jButDBSource = new javax.swing.JButton();

jButExit = new javax.swing.JButton();

jLabel5 = new javax.swing.JLabel();

jTextField5 = new javax.swing.JTextField();


setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

setTitle("Find Oracle Objects");


jButRunAnalysis.setText("Search in files");

jButRunAnalysis.setActionCommand("");

jButRunAnalysis.addActionListener(new java.awt.event.ActionListener() {

    public void actionPerformed(java.awt.event.ActionEvent evt) {

        jButRunAnalysisActionPerformed(evt);

    }

});


jTextField1.setEditable(false);

jTextField1.addActionListener(new java.awt.event.ActionListener() {

    public void actionPerformed(java.awt.event.ActionEvent evt) {

        jTextField1ActionPerformed(evt);

    }

})
```

```
});

jTextField1.addPropertyChangeListener(new java.beans.PropertyChangeListener() {

    public void propertyChange(java.beans.PropertyChangeEvent evt) {

        jTextField1PropertyChange(evt);

    }

});

jLabel1.setText("Object found / found in:");

jTextField4.addActionListener(new java.awt.event.ActionListener() {

    public void actionPerformed(java.awt.event.ActionEvent evt) {

        jTextField4ActionPerformed(evt);

    }

});

jTextField4.addPropertyChangeListener(new java.beans.PropertyChangeListener() {

    public void propertyChange(java.beans.PropertyChangeEvent evt) {

        jTextField4PropertyChange(evt);

    }

});

jLabel2.setText("of");
```



```
jLabel3.setText("Processing file / object:");
```

```
jLabel4.setText("File to process / Object to search for:");
```

```
jButDBSource.setText("Search in DB Source");
```

```
jButDBSource.setToolTipText("");
```

```
jButDBSource.setActionCommand("");
```

```
jButDBSource.setName(""); // NOI18N
```

```
jButDBSource.addActionListener(new java.awt.event.ActionListener() {
```

```
    public void actionPerformed(java.awt.event.ActionEvent evt) {
```

```
        jButDBSourceActionPerformed(evt);
```

```
    }
```

```
});
```

```
jButExit.setText("Exit");
```

```
jButExit.addFocusListener(new java.awt.event.FocusAdapter() {
```

```
    public void focusGained(java.awt.event.FocusEvent evt) {
```

```
        jButExitFocusGained(evt);
```

```
    }
```

```
});
```

```
jButExit.addMouseListener(new java.awt.event.MouseAdapter() {  
    public void mouseClicked(java.awt.event.MouseEvent evt) {  
        jButExitMouseClicked(evt);  
    }  
});  
  
jButExit.addActionListener(new java.awt.event.ActionListener() {  
    public void actionPerformed(java.awt.event.ActionEvent evt) {  
        jButExitActionPerformed(evt);  
    }  
});  
  
jLabel5.setText("Current status:");  
  
jTextField5.setEditable(false);  
  
jTextField5.setFont(new java.awt.Font("Tahoma", 2, 11)); // NOI18N  
  
jTextField5.addActionListener(new java.awt.event.ActionListener() {  
    public void actionPerformed(java.awt.event.ActionEvent evt) {  
        jTextField5ActionPerformed(evt);  
    }  
});
```



```
.addGap(18, 18, 18)

.addComponent(jButDBSource)

.addGap(37, 37, 37)

.addComponent(jButExit, javax.swing.GroupLayout.PREFERRED_SIZE,
83, javax.swing.GroupLayout.PREFERRED_SIZE))

.addGroup(layout.createSequentialGroup())

.addGap(115, 115, 115)

.addComponent(jLabel4, javax.swing.GroupLayout.PREFERRED_SIZE,
207, javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addComponent(jTextField2,
javax.swing.GroupLayout.PREFERRED_SIZE, 469,
javax.swing.GroupLayout.PREFERRED_SIZE)))

.addGroup(layout.createSequentialGroup())

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addGap(329, 329, 329)

.addComponent(jLabel3, javax.swing.GroupLayout.PREFERRED_SIZE,
147, javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED))
```

```
.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,  
layout.createSequentialGroup()  
  
    .addContainerGap()  
  
    .addComponent(jLabel5)  
  
    .addGap(46, 46, 46)))  
  
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  
  
    .addComponent(jTextField5,  
javax.swing.GroupLayout.PREFERRED_SIZE, 192,  
javax.swing.GroupLayout.PREFERRED_SIZE)  
  
    .addGroup(layout.createSequentialGroup()  
  
        .addComponent(jTextField3,  
javax.swing.GroupLayout.PREFERRED_SIZE, 76,  
javax.swing.GroupLayout.PREFERRED_SIZE)  
  
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)  
  
        .addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED_SIZE,  
16, javax.swing.GroupLayout.PREFERRED_SIZE)  
  
        .addComponent(jTextField4,  
javax.swing.GroupLayout.PREFERRED_SIZE, 78,  
javax.swing.GroupLayout.PREFERRED_SIZE))))))  
  
    .addContainerGap(292, Short.MAX_VALUE))  
  
);  
  
layout.setVerticalGroup(
```

```
layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

    .addGroup(layout.createSequentialGroup()

        .addGap(33, 33, 33)

        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

            .addComponent(jLabel5)

            .addComponent(jTextField5, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))

        .addGap(18, 18, 18)

        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

            .addComponent(jTextField3, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)

            .addComponent(jLabel2)

            .addComponent(jTextField4, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)

            .addComponent(jLabel3, javax.swing.GroupLayout.PREFERRED_SIZE, 20,
javax.swing.GroupLayout.PREFERRED_SIZE))

        .addGap(30, 30, 30)

        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

            .addComponent(jTextField2, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
.addComponent(jLabel4))

.addGap(18, 18, 18)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

    .addComponent(jLabel1)

    .addComponent(jTextField1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 90,
Short.MAX_VALUE)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

    .addComponent(jButRunAnalysis)

    .addComponent(jButDBSource)

    .addComponent(jButExit))

.addGap(90, 90, 90))

);

pack();

} // </editor-fold>
```

```
public boolean isPureAscii(String v) {
```

```
byte bytearray [] = v.getBytes();

CharsetDecoder d = Charset.forName("US-ASCII").newDecoder();

try {

    CharBuffer r = d.decode(ByteBuffer.wrap(bytearray));

    r.toString();

}

catch(CharacterCodingException e) {

    return false;

}

return true;

}

private void jButRunAnalysisActionPerformed(java.awt.event.ActionEvent evt) {

    Connection conn=null;

    Statement stmt = null;

    String query = "";

    String[][] objetos = null;
```



```
Connection connora=null;

int rows = 1;

jTextField5.setText("Connecting to Oracle DB..." );

jTextField5.update(jTextField5.getGraphics());

try {

    connora = conexionoracle();

    jTextField5.setText("Connected to Oracle DB..." );

    jTextField5.update(jTextField5.getGraphics());

    query = "SELECT * FROM dba_objects WHERE (object_type like '%PACKAGE' OR
object_type like '%TRIGGER%' OR object_type like '%VIEW%' OR object_type like
'%TABLE%' OR object_type like '%FUNCTION%' OR object_type like '%SEQUENCE%'
OR object_type like '%PROCEDURE%') AND (object_name like 'XX%' OR object_name
like 'ZIN%' OR object_name like 'HSEOE%')";

    //System.out.println(sfecha);

    stmt = connora.createStatement(
ResultSet.TYPE_SCROLL_INSENSITIVE,ResultSet.CONCUR_READ_ONLY);

    ResultSet rs = stmt.executeQuery(query);

    jTextField5.setText("Getting database objects..." );

    jTextField5.update(jTextField5.getGraphics());

    rs.last();
```

```
rows = rs.getRow();

// Move to beginning

rs.beforeFirst();

objetos = new String[rows][3];


int contobj = 0;


while (rs.next()) {

    objetos[contobj][0]= rs.getString("OBJECT_NAME");

    objetos[contobj][1]= rs.getString("OBJECT_TYPE");

    objetos[contobj][2]= rs.getString("OWNER");

    contobj++;

}


} catch (SQLException ex) {

    Logger.getLogger(OBEX_Main_Form.class.getName()).log(Level.SEVERE, null, ex);

}


FilenameFilter filter = new FilenameFilter() {
```

```
public boolean accept(File dir, String name) {  
    return name.endsWith("ZIN_FOWIP021.fmb");  
}  
};  
  
File folder = new File("c:/XXIZ");  
  
/*  
  
PrintStream fileStream;  
  
try {  
    fileStream = new PrintStream("output.txt");  
    System.setOut(fileStream);  
} catch (FileNotFoundException ex) {  
    Logger.getLogger(OBEX_Main_Form.class.getName()).log(Level.SEVERE, null, ex);  
}  
  
*/  
  
jTextField5.setText("Reading files..." );  
  
jTextField5.update(jTextField5.getGraphics());  
  
File[] listOfFiles = folder.listFiles();  
  
//File[] listOfFiles = folder.listFiles(filter);  
  
int totalFiles = listOfFiles.length;
```

```
jTextField4.setText("" + totalFiles);

jTextField4.update(jTextField4.getGraphics());


for (int i = 0; i < listOfFiles.length; i++) {

    File fileName = listOfFiles[i];

    int cont = 0;

    //String objbuscado = "zin_ta_nlink_lookup";

    String objbuscado = "";

    String objtipo = "";

    String objowner = "";

    String objtablealias = "";


    jTextField2.setText("" + fileName);

    jTextField2.update(jTextField2.getGraphics());


    jTextField3.setText("" + (i+1));

    jTextField3.update(jTextField3.getGraphics());


    try {

        //BufferedReader entradaArchivo=new BufferedReader(new InputStreamReader(new
        FileInputStream(fileDir),"UTF-8"));
```

```
BufferedReader entradaArchivo=new BufferedReader(new InputStreamReader(new
FileInputStream(fileName),"UTF-8"));
```

```
String linea;
```

```
String w="";
```

```
String tokenant="";
```

```
String tokensig="";
```

```
Boolean encontrado;
```

```
Boolean iniciabloque;
```

```
iniciabloque = false;
```

```
//linea = entradaArchivo.readLine();
```

```
while ( ((linea = entradaArchivo.readLine()) != null)) { //&& (!(linea.contains("/") ))
&& (!(linea.contains("*/") )) ) {
```

```
    //linea = entradaArchivo.readLine();
```

```
    StringTokenizer tokens = new StringTokenizer( linea,"n,;. " );
```

```
    //StringTokenizer tokens = new StringTokenizer( linea," ;" );
```

```
    while (tokens.hasMoreTokens()) {
```

```
        tokenant = w;
```

```
        w = tokens.nextToken();
```

```
    // strips off all non-ASCII characters
```

```
w = w.replaceAll("[^\\x00-\\x7F]", "");

// erases all the ASCII control characters

w = w.replaceAll("[\\p{Cntrl}&&[^\r\n\t]]", "");

// removes non-printable characters from Unicode

w = w.replaceAll("\\p{C}", "");

if (w.endsWith("*/")){

    iniciabloque = false;

}

if (w.startsWith("/*") || iniciabloque){

    //System.out.println("***COMIENZA BLOQUE COMENTADO***");

    iniciabloque = true;

    while (tokens.hasMoreTokens()) {

        tokenant = w;

        w = tokens.nextToken();

        w = w.replaceAll("[^\\x00-\\x7F]", "");
```

```
w = w.replaceAll("[\\p{Cntrl}&&[^\r\n\t]]", "");

w = w.replaceAll("\\p{C}", "");

if (w.endsWith("*/")){

    iniciabloque = false;

}

}

//break;

}

else{

    if (w.startsWith("/") || w.startsWith("--")){

        //System.out.println("***COMIENZA LINEA COMENTADA***");

        while (tokens.hasMoreTokens()) {

            tokenant = w;

            w = tokens.nextToken();

        }

        //break;

    }

}

w = w.toLowerCase();
```

```
tokensig = w ;

//jTextField6.setText(w);

//jTextField6.update(jTextField6.getGraphics());

//System.out.println("TOKEN: " + w);

/*if("TABLE".equals(objtipo.trim())){

    objtablealias = tokensig;

    objtablealias = objtablealias.toLowerCase();

    System.out.println(objbuscado + " *ALIAS* " + objtablealias +
"|ENCONTRADO EN|" + fileName );

    objtipo = "";

} */

if("PACKAGE".equals(objtipo.trim())){

    objtablealias = tokensig;

    objtablealias = objtablealias.toLowerCase();

    System.out.println(objowner + "|" + objbuscado + "." + objtablealias + "|" +
"PACKAGE&FUNCTION|" + fileName.getName());

    objtipo = "";

}

encontrado = false;

if (w.length()>1){
```



```
for (int j=0; j<objetos.length;j++){  
  
    objbuscado = objetos[j][0];  
  
    objtipo = objetos[j][1];  
  
    objowner = objetos[j][2];  
  
    if (objbuscado.length()>1){  
  
        //jTextField5.setText(w);  
  
        //jTextField5.update(jTextField5.getGraphics());  
  
        if (w.equals(objbuscado.toLowerCase())){  
  
            //printout object name, object type and file name where object was  
found:  
  
            System.out.println(objowner + "|" + objbuscado + "|" + objtipo.trim() +  
            "|" + fileName.getName() );  
  
            cont++;  
  
            encontrado = true;  
  
            jTextField1.setText(objbuscado);  
  
            jTextField1.update(jTextField1.getGraphics());  
  
        }  
  
        if (encontrado) {  
  
            break;  
  
        }  
  
    }  
}
```

```
    }

    if (!encontrado) {

        objtipo = "";

    }

}

}

//linea = entradaArchivo.readLine();

}

} catch (FileNotFoundException ex) {

    Logger.getLogger(OBEX_Main_Form.class.getName()).log(Level.SEVERE, null, ex);

} catch (IOException ex) {

    Logger.getLogger(OBEX_Main_Form.class.getName()).log(Level.SEVERE, null, ex);

}

}

System.exit(0);
```

```
}
```

```
private void jTextField1ActionPerformed(java.awt.event.ActionEvent evt) {
```

```
    // TODO add your handling code here:
```

```
}
```

```
private void jTextField1PropertyChange(java.beans.PropertyChangeEvent evt) {
```

```
    // TODO add your handling code here:
```

```
}
```

```
private void jTextField4ActionPerformed(java.awt.event.ActionEvent evt) {
```

```
    // TODO add your handling code here:
```

```
}
```

```
private void jTextField4PropertyChange(java.beans.PropertyChangeEvent evt) {
```

```
    // TODO add your handling code here:
```

```
}
```

```
private void jButtonSourceActionPerformed(java.awt.event.ActionEvent evt) {
```

```
    // TODO add your handling code here:
```

```
Connection conn=null;

Statement stmt = null;

Statement stmt2 = null;

String query = "";

String query2 = "";

String[][] objetos = null;

String[][] objetos2 = null;
```

```
Connection connora=null;

int rows = 1;

int rows2 = 1;

int contobj = 0;

int contobj2 = 0;
```

```
jTextField5.setText("Connecting to Oracle DB..." );

jTextField5.update(jTextField5.getGraphics());

try {
```

```
    connora = conexioracle();

    jTextField5.setText("Connected to Oracle DB..." );

    jTextField5.update(jTextField5.getGraphics());
```

```
//query = "SELECT * FROM dba_objects WHERE (object_name like  
'XXCFD_TA_CERTIFICADOS%')";
```

```
query = "SELECT * FROM dba_objects WHERE (object_type like '%PACKAGE' OR  
object_type like '%TRIGGER%' OR object_type like '%VIEW%' OR object_type like  
'%TABLE%') AND (object_name like 'XX%' OR object_name like 'ZIN%' OR object_name  
like 'HSEO%')";
```

```
//System.out.println(sfecha);
```

```
stmt = connora.createStatement(  
ResultSet.TYPE_SCROLL_INSENSITIVE,ResultSet.CONCUR_READ_ONLY);
```

```
ResultSet rs = stmt.executeQuery(query);
```

```
jTextField5.setText("Getting database objects..." );
```

```
jTextField5.update(jTextField5.getGraphics());
```

```
rs.last();
```

```
rows = rs.getRow();
```

```
// Move to beginning
```

```
rs.beforeFirst();
```

```
objetos = new String[rows][3];
```

```
contobj = 0;
```

```
while (rs.next()) {
```

```
        objetos[contobj][0]= rs.getString("OBJECT_NAME");

        objetos[contobj][1]= rs.getString("OBJECT_TYPE");

        objetos[contobj][2]= rs.getString("OWNER");

        contobj++;

    }

} catch (SQLException ex) {

    Logger.getLogger(OBEX_Main_Form.class.getName()).log(Level.SEVERE, null, ex);

}

/*

PrintStream fileStream;

try {

    fileStream = new PrintStream("output.txt");

    System.setOut(fileStream);

} catch (FileNotFoundException ex) {

    Logger.getLogger(OBEX_Main_Form.class.getName()).log(Level.SEVERE, null, ex);

}

*/
```

```
jTextField5.setText("Reading files..." );  
  
jTextField5.update(jTextField5.getGraphics());
```

```
jTextField4.setText("" + contobj);  
  
jTextField4.update(jTextField4.getGraphics());
```

```
String objbuscado = "";  
  
String objtipo = "";  
  
String objowner = "";  
  
String objtablealias = "";  
  
String StrQuery = "";  
  
for (int i = 0; i < contobj; i++) {  
  
    //File fileName = listOfFile[i];  
  
  
    //String objbuscado = "zin_ta_nlink_lookup";  
  
    objbuscado = objetos[i][0];  
  
    objtipo = objetos[i][1];  
  
    objowner = objetos[i][2];  
  
    objbuscado = objbuscado.toLowerCase();
```

```
StrQuery = "SELECT DISTINCT * FROM dba_source WHERE (name like 'XX%' OR  
name like 'ZIN%' OR name like 'HSOE%') and (lower(text) like '%" + objbuscado + "%)";
```

```
//System.out.println("QUERY:|" + StrQuery );
```

```
query2 = StrQuery;
```

```
//query2 = "SELECT DISTINCT * FROM dba_source WHERE (name like 'XX%' OR  
name like 'ZIN%' OR name like 'HSOE%') and (lower(text) like '%xxcfd_ta_certificados%')";
```

```
try {
```

```
    //System.out.println(sfecha);
```

```
    stmt2 = connora.createStatement(  
ResultSet.TYPE_SCROLL_INSENSITIVE,ResultSet.CONCUR_READ_ONLY);
```

```
    ResultSet rs2 = stmt2.executeQuery(query2);
```

```
    //rs2.last();
```

```
    //rows2 = rs2.getRow();
```

```
    // Move to beginning
```

```
    //rs2.beforeFirst();
```

```
    jTextField2.setText("" + objbuscado);
```

```
    jTextField2.update(jTextField2.getGraphics());
```

```
    jTextField3.setText("" + (i+1));
```

```
    jTextField3.update(jTextField3.getGraphics());
```



```
String linea2;
```

```
String w="";
```

```
String tokenant="";
```

```
String tokensig="";
```

```
Boolean encontrado;
```

```
while (rs2.next()) {
```

```
    linea2 = rs2.getString("TEXT");
```

```
    //System.out.println("TEXT:" + linea2 );
```

```
    StringTokenizer tokens = new StringTokenizer( linea2, " ,");
```

```
    jTextField1.setText("");
```

```
    jTextField1.update(jTextField1.getGraphics());
```

```
    //StringTokenizer tokens = new StringTokenizer( linea, " ,;" );
```

```
    while (tokens.hasMoreTokens()) {
```

```
        tokenant = w;
```

```
        w = tokens.nextToken();
```

```
        w = w.toLowerCase();
```

```
        tokensig = w ;
```

```
        //jTextField6.setText(w);
```

```
        //jTextField6.update(jTextField6.getGraphics());
```

```
        //System.out.println("TOKEN: " + w);
```

```

/*if("TABLE".equals(objtipo.trim())){

    objtablealias = tokensig;

    objtablealias = objtablealias.toLowerCase();

    System.out.println(objbuscado + " *ALIAS* " + objtablealias +
"|ENCONTRADO EN|" + fileName );

    objtipo = "";

} */


//jTextField5.setText(w);

//jTextField5.update(jTextField5.getGraphics());

//System.out.println("TOKEN ANALIZADO: " + w );

if ("PROCEDURE".equals(tokenant)){

    System.out.println("PROCEDURE " + tokensig + "|ENCONTRADO EN|" +
rs2.getString("NAME") );

}

if (w.equals(objbuscado.toLowerCase())){

    //printout object name, object type and file name where object was
found:

    System.out.println(objowner + "|" + objbuscado + "|" + objtipo.trim() +
"|ENCONTRADO EN|" + rs2.getString("NAME") + "|LINE|" + rs2.getString("LINE") );

```

```
        encontrado = true;

        jTextField1.setText(rs2.getString("NAME"));

        jTextField1.update(jTextField1.getGraphics());

    }

}

}

} catch (SQLException ex) {

    Logger.getLogger(OBEX_Main_Form.class.getName()).log(Level.SEVERE, null,
ex);

}

}

System.exit(0);
```

```
}
```

```
private void jButExitActionPerformed(java.awt.event.ActionEvent evt) {
```

```
    // TODO add your handling code here:
```

```
    System.exit(0);
```

```
}
```

```
private void jTextField5ActionPerformed(java.awt.event.ActionEvent evt) {
```

```
    // TODO add your handling code here:
```

```
}
```

```
private void jButExitFocusGained(java.awt.event.FocusEvent evt) {
```

```
    // TODO add your handling code here:
```

```
    System.exit(0);
```

```
}
```

```
private void jButExitMouseClicked(java.awt.event.MouseEvent evt) {
```

```
    // TODO add your handling code here:
```

```
System.exit(0);

}

public Connection conexionmysql() throws ClassNotFoundException{

    Connection conect=null;

    String bd="sample";

    String url="jdbc:derby://localhost:1527/"+bd;

    String user="app";

    String pwd="app";

    try{

        //Class.forName("com.mysql.jdbc.Driver");

        Class.forName("org.apache.derby.jdbc.ClientDriver").newInstance();

        conect=DriverManager.getConnection(url,user,pwd);

        System.out.println("Conexión exitosa!!");

        //jTextArea1.setText(jTextArea1.getText() + "\n\n" + "Conexión exitosa!!" + "\n");

        //JOptionPane1.showMessageDialog(null,"Conexion correcta a "+this.bd);

    }

    catch (SQLException err){
```

```
//JOptionPane.showMessageDialog(null,"Conexion Incorrecta");

System.out.println(err.getMessage());

} catch (InstantiationException ex) {

    Logger.getLogger(OBEX_Main_Form.class.getName()).log(Level.SEVERE, null, ex);

} catch (IllegalAccessException ex) {

    Logger.getLogger(OBEX_Main_Form.class.getName()).log(Level.SEVERE, null, ex);

}

return conect;

}
```

```
public Connection conexionoracle() throws SQLException{
```

```
    try

    {

        Class.forName ("oracle.jdbc.driver.OracleDriver");

    }

    catch (ClassNotFoundException e)

    {

        e.printStackTrace();

    }

}
```

```
Connection connoracle = DriverManager.getConnection

("jdbc:oracle:thin:APPS@lddebs04.interceramic.com:2530/CRP6", "*****",
"*****");

//      @TNSNames_Entry, userid, password

/*

Statement stmtoracle = connoracle.createStatement();

ResultSet rset =

    stmtoracle.executeQuery("select BANNER from SYS.V_$VERSION");

while (rset.next())

    System.out.println (rset.getString(1)); // Print col 1

stmtoracle.close();

*/

return connoracle;

}
```

```
/**  
  
 * @param args the command line arguments  
  
 */  
  
public static void main(String args[]) {  
  
    /* Set the Nimbus look and feel */  
  
    //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">  
  
    /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and  
feel.  
  
    * For details see  
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html  
  
    */  
  
    try {  
  
        for (javax.swing.UIManager.LookAndFeelInfo info :  
javax.swing.UIManager.getInstalledLookAndFeels()) {  
  
            if ("Nimbus".equals(info.getName())) {  
  
                javax.swing.UIManager.setLookAndFeel(info.getClassName());  
  
                break;  
  
            }  
  
        }  
  
    } catch (ClassNotFoundException ex) {  
  
  
        java.util.logging.Logger.getLogger(OBEX_Main_Form.class.getName()).log(java.util.logging  
.Level.SEVERE, null, ex);  
  
    }  
  
}
```



```
    } catch (InstantiationException ex) {
```

```
java.util.logging.Logger.getLogger(OBEX_Main_Form.class.getName()).log(java.util.logging
.Level.SEVERE, null, ex);
```

```
    } catch (IllegalAccessException ex) {
```

```
java.util.logging.Logger.getLogger(OBEX_Main_Form.class.getName()).log(java.util.logging
.Level.SEVERE, null, ex);
```

```
    } catch (javax.swing.UnsupportedLookAndFeelException ex) {
```

```
java.util.logging.Logger.getLogger(OBEX_Main_Form.class.getName()).log(java.util.logging
.Level.SEVERE, null, ex);
```

```
    }
```

```
//</editor-fold>
```

```
/* Create and display the form */
```

```
java.awt.EventQueue.invokeLater(new Runnable() {
```

```
    public void run() {
```

```
        new OBEX_Main_Form().setVisible(true);
```

```
    }
```

```
});
```

```
}
```

```
// Variables declaration - do not modify
```

```
private javax.swing.JButton jButDBSource;
```

```
private javax.swing.JButton jButExit;  
  
private javax.swing.JButton jButRunAnalysis;  
  
private javax.swing.JLabel jLabel1;  
  
private javax.swing.JLabel jLabel2;  
  
private javax.swing.JLabel jLabel3;  
  
private javax.swing.JLabel jLabel4;  
  
private javax.swing.JLabel jLabel5;  
  
private javax.swing.JTextField jTextField1;  
  
private javax.swing.JTextField jTextField2;  
  
private javax.swing.JTextField jTextField3;  
  
private javax.swing.JTextField jTextField4;  
  
private javax.swing.JTextField jTextField5;  
  
// End of variables declaration  
  
}
```