

UNIVERSIDAD TÉCNICA DEL NORTE



FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CARRERA DE INGENIERÍA EN SISTEMAS COMPUTACIONALES

Documento Técnico del Proyecto de Tesis

Tema: “Implantación de una herramienta ERP con licencia GPL y desarrollo del anexo transaccional para la Empresa DIPROMAC”

Autora: Angela Natalia Rojas Tobar

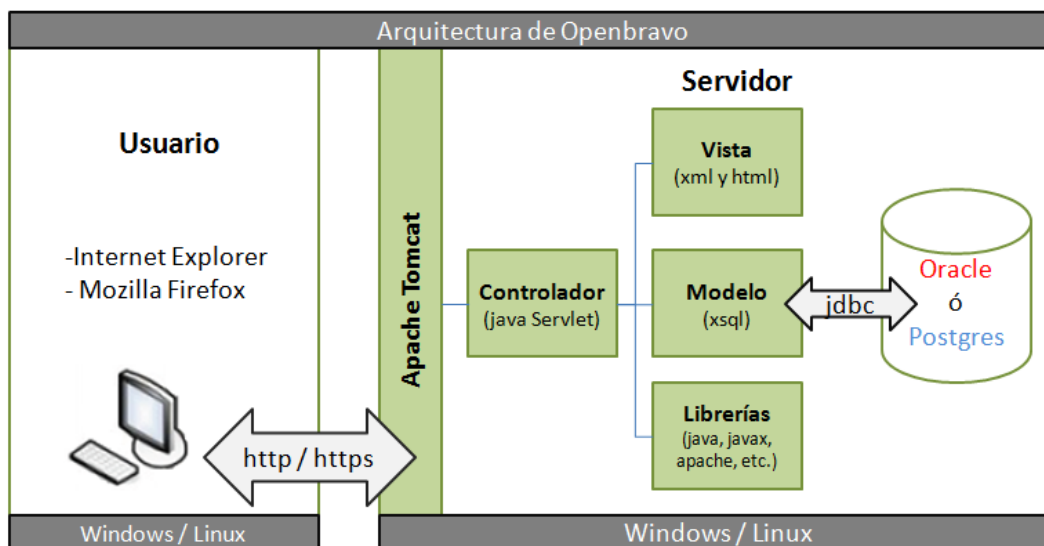
Director: Ing. Irving Reascos

Ibarra – Ecuador, 2011

Arquitectura del sistema Openbravo.

Openbravo es una aplicación con arquitectura cliente/servidor web escrita en código Java. Se ejecuta sobre Apache Tomcat y tiene soporte para bases de datos PostgreSQL y Oracle. Es una aplicación completamente web que ha sido desarrollada siguiendo el modelo MVC (Modelo, Vista, Controlador), lo que facilita el separación de las áreas de desarrollo, permitiendo el crecimiento sostenible de la aplicación y una mayor facilidad en el mantenimiento del código. La mayor parte del código se genera automáticamente por el motor denominado WAD (Wizard for Application Development), basándose en la información contenida en el diccionario del modelo de datos (Data Model Dictionary). El motor WAD ejecuta y recompila la aplicación cada vez que el administrador modifica la configuración para adaptarla a un nuevo requerimiento.

- **Modelo:** Archivos XSQL en los que se encuentran sentencias SQL las cuales serán ejecutadas.
- **Vista:** Archivos HTML y XML en los que se definen los formularios y las plantillas donde serán insertados los datos.
- **Controlador:** Java Servlets en los que se definen las acciones dentro de los formularios.



MVC-FF.

MVC Foundation Framework (MVC-FF) está compuesto por el set de utilidades desarrolladas por Openbravo como XmlEngine, SQLC y HttpBaseSecureServlet. MVC-FF es necesario para permitir el desarrollo y generación de archivos para el modelo, la vista y el controlador.

XmlEngine.

Es un servicio usado para crear documentos XML/HTML desde una plantilla en formato XML/HTML y un archivo XML de configuración con los datos dinámicos que serán insertados en la plantilla. El archivo de configuración mapea la fuente de datos con los lugares identificados en la plantilla. Las plantillas son leídas y almacenadas en memoria, luego cuando una página es requerida. La plantilla crea un documento el cual se llena con los datos obtenidos por la aplicación. Con esta herramienta Openbravo genera los formularios, crea reportes o los imprime.

SQLC.

SQL Compiler es un servicio que permite interactuar con la base de datos. La interacción se produce mediante un archivo XML el cual contiene sentencias estándar SQL y los parámetros que serán usados en las sentencias. SQLC lee el archivo y genera una clase java la que contendrá todo el código necesario para conectarse a la base de datos, ejecutar sentencias.

HttpBaseServlet.

HttpBaseServlet y HttpBaseSecureServlet son servlets desde los cuales todos los servlets del sistema heredan. Estos servlets implementan funcionalidades como autenticación, autorización, conectividad a la base de datos y errores.

Data Model Dictionary y WAD.

La generación de código es posible por el uso de Data Model Dictionary y por el WAD (Wizard for Application Development). El WAD automáticamente genera todos los archivos de la aplicación con el modelo MVC. Los archivos son generados mediante el uso de XmlEngine, SQLC y HttpBaseSecureServlet.

Herramientas para el desarrollo.

Los requerimientos para el entorno de desarrollo sugerido en el sitio de la comunidad de Openbravo ERP, y los cuales fueron empleados:

- PostgreSQL versión 8.4
- Sun Java Development Kit (JDK) versión 1.6
- Apache Tomcat versión 6.0.20
- Apache Ant versión 1.8.0
- Eclipse Ganymede versión 3.4.2
- Mozilla FireFox 3.6.10

Previo al desarrollo se deben establecer las variables de entorno de java, apache tomcat y de apache ant (ANT_OPTS, CATALINA_OPTS, JAVA_HOME). El código fuente se lo puede descargar de sitio oficial de Openbravo o también lo pueden descargar desde www.sourceforge.net, he trabajado con la versión 2.50.

Implementación del Módulo Anexo-T.

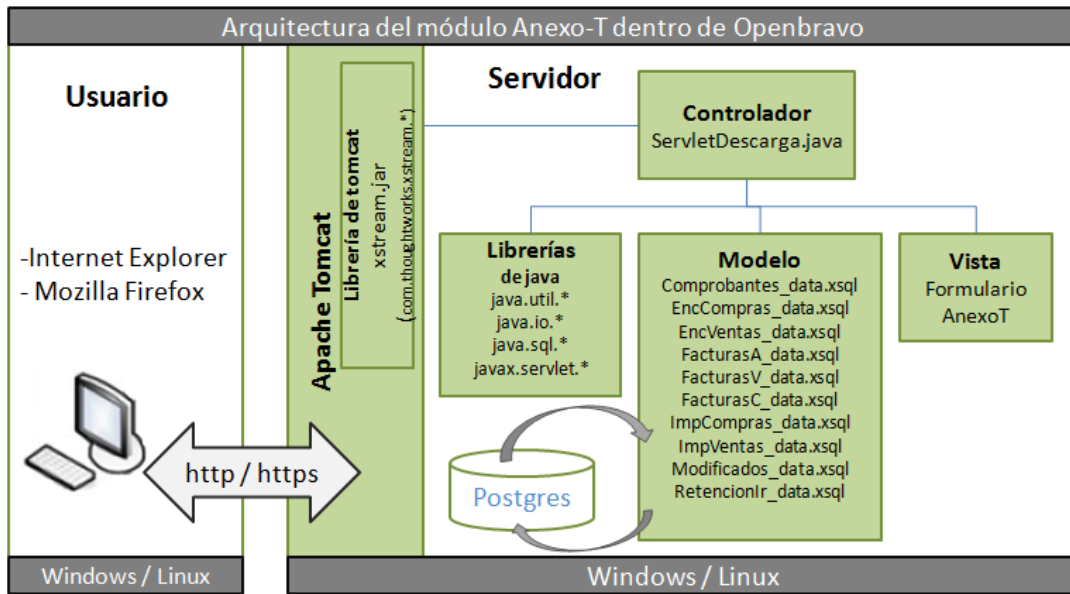
Librería adicional.

El funcionamiento del módulo Anexo-T depende de la existencia del JAR *xstream* dentro de la carpeta */lib* del *apache tomcat*.

Modificaciones adicionales al proyecto Openbravo.

Tomando en cuenta las características de la empresa distribución de leche DIPROMAC y los requerimientos para el anexo transaccional, el sistema Openbravo no contaba con toda la información para la generación del anexo transaccional por lo tanto fue muy necesario agregar algunos cambios a los módulos del sistema. Los cambios los realicé mediante el *framework* de desarrollo de *Openbravo*, modificaciones en la base de datos y ejecución de tareas *Ant*.

Arquitectura del módulo Anexo-T.



Funcionamiento.

Clases adicionales

Realicé con la ayuda del JAR *xstream* las siguientes clases java que contienen las respectivas anotaciones *@XStreamAlias* para la estructura del archivo XML que será generado:

- Iva.java
- DetalleAir.java
- DetalleAnulado.java
- DetalleCompra.java
- DetalleVenta.java

Realicé la clase *Generar.java* que permite generar el XML mediante el *xstream* y luego retornar un vector de tipo *byte* con la información del anexo.

```
public class Generar {  
  
    public static final String CABECERA_XML="<?xml version=\"1.0\" encoding=\"ISO-8859-1\"  
    standalone=\"yes\"?>\n";  
  
    public Generar (){}  
  
    public byte[] generar(Iva iva){  
        XStream stream = new XStream(new DomDriver());  
        stream.processAnnotations(Iva.class);  
        try {  
            String xml=CABECERA_XML.concat(stream.toXML(iva));  
            return xml.getBytes();  
        }  
        catch(Exception ex){  
            ex.printStackTrace();  
            return null;  
        }  
    }  
}
```

```
}  
}
```

Modelo

Luego realicé los archivos xsql que nos permiten trabajar con la base de datos. Una vez creados estos archivos al realizar la tarea *ant smartbuild* la herramienta SQLC (SQL Compiler que es parte del framework de Openbravo) genera las clases java a partir de los archivos xsql para facilitar el trabajo con la base de datos. Para la creación de los archivos xsql se debe respetar el estándar con el que trabaja Openbravo, es decir que el nombre del archivo debe terminar con *_data*. Por lo tanto cada archivo xsql tiene su respectiva clase java, el nombre de la clase java se lo establece dentro del archivo xsql:

- Comprobantes_data.xsql (ComprobantesData.java)
- EncCompras_data.xsql (EncComprasData.java)
- EncVentas_data.xsql (EncVentasData.java)
- FacturasA_data.xsql (FacturasAData.java)
- FacturasC_data.xsql (FacturasCData.java)
- FacturasV_data.xsql (FacturasVData.java)
- ImpCompras_data.xsql (ImpComprasData.java)
- ImpVentas_data.xsql (ImpVentasData.java)
- Modificados_data.xsql (ModificadosData.java)
- RetencionIr_data.xsql (RetencionIrData.java)

A continuación muestro la estructura de un archivo xsql en este caso del archivo *Enc_Compras_data.xsql* el cual contiene la consulta a la base de datos:

```
<?xml version="1.0" encoding="UTF-8"?>  
<SqlClass name="EncComprasData" package="org.xim.anexos.consulta">  
  <SqlClassComment>Class EncComprasData</SqlClassComment>  
  <SqlMethod name="select" type="preparedStatement" return="multiple">  
    <SqlMethodComment>Select encabezado factura</SqlMethodComment>  
    <Sql><![CDATA[  
      select c.em_ax_sustento as em_ax_sustento,  
        b.em_ax_tipoid as em_ax_tipoid,  
        b.em_ax_ciruc as em_ax_ciruc,  
        c.c_doctype_id as c_doctype_id,  
        to_date(c.dateinvoiced) as dateinvoiced,  
        c.em_ax_nestablecimiento as em_ax_nestablecimiento,  
        c.em_ax_npuntoemision as em_ax_npuntoemision,  
        c.documentno as documentno,  
        c.em_ax_nautorizacion as em_ax_nautorizacion  
        from c_invoice c join c_bpartner b on c.c_bpartner_id=b.c_bpartner_id  
        where c.c_invoice_id=?  
    ]]></Sql>  
    <Parameter name="paraminvoiceid"></Parameter>  
  </SqlMethod>  
</SqlClass>
```

En la etiqueta *<SqlClass>* se define ciertos atributos como *name* que es para el nombre de la clase java que generara al compilar y *package* que es para el nombre del paquete en donde se generará la clase java.

En la etiqueta *<SqlClassComent/>* donde definimos algún comentario para la clase java.

En la etiqueta <SqlCommand> se define los atributos *name* que es el nombre del método dentro de la clase java, *type* se define el tipo de ejecución de la consulta (constant, preparedStatement, statement, callableStatement) y return se define el retorno de una o múltiples filas (multiple, single).

En la etiqueta <SqlCommandComment/> se define un comentario para el método.

En la etiqueta <Sql><![CDATA[...]]></Sql> se define la sentencia SQL para la consulta a base de datos.

En la etiqueta <Parameter/> se define en el atributo *name* el nombre del parámetro para la consulta.

Controlador

El controlador es el *ServletDescarga.java* que se extiende de la clase *HttpSecureAppServlet* (*HttpSecureAppServlet* es una clase de Openbravo). En el método *doPost()* se almacenan en variables tipo String los parámetros básicos para las consultas a la base de datos. Los parámetros son *clienteid* (Client), *orgid* (Organization) y *fecha*. Estos parámetros se envían al método *anexoAT()*.

```
public void doPost(HttpServletRequest request, HttpServletResponse response) throws
IOException, ServletException {
    VariablesSecureApp vars = new VariablesSecureApp(request);

    if (vars.commandIn("DEFAULT")) {
        String clienteid=vars.getStringParameter("inpadClientId");
        String orgid=vars.getStringParameter("inpadOrgId");
        String fecha=vars.getStringParameter("inpaxFecha");

        if (clienteid !=null && orgid!=null && fecha!=null){
            anexoAT(response, vars, orgid, clienteid, fecha);
        }
        else
            pageError(response);
    }
}
```

En el servlet se importó el paquete *org.xim.anexos.clasexml*, este paquete contiene las clases java con las anotaciones respectivas para la estructura del archivo XML del anexo (Iva, DetalleAir, DetalleCompra, DetalleVenta, DetalleAnulado). El método *anexoAT(HttpServletResponse response, VariablesSecureApp vars, String orgid, String clienteid, String fecha)* trabaja con los objetos que indico a continuación:

```
Iva iva=new Iva();
List <DetalleCompra> detalleC=null;
List <DetalleVenta> detalleV=null;
List <DetalleAir> detalleAir=null;
List <DetalleAnulado> detalleA=null;
DetalleCompra dc=null;
DetalleVenta dv=null;
DetalleAir dair=null;
DetalleAnulado da=null;
```

Trabaja también con un objeto para la conexión a la base de datos:

```
Connection conn=null
```

Tiene también las siguientes variables las que me ayudarán a almacenar el número de ruc, la razón social, obtener el mes y año para filtrar la información de compras y ventas, crear un nombre para el archivo XML, identificar compras, identificar ventas y el estado, y una variable más para identificar si es o no cliente.

```
String numeroRuc="";
String razonSocial="";
String []fechaAT=fecha.split("-");
String mesAT=fechaAT[1];
String anioAT=fechaAT[2];
String nombreArchivo="AT"+mesAT+anioAT;
String compra="N";
String venta="Y";
String estadoFac="CO";
String anulado="VO";
String isCli="Y";
```

Dentro de un *try* hace la conexión a la base de datos mediante el método que se detalla a continuación el cual es de la clase *HttpSecureAppServlet*:

```
conn=getTransactionConnection();
```

Esta variable *con* nos permitirá trabajar con las clases java que fueron generadas por el SQLC y así podremos realizar las consultas a la base de datos. Luego de obtener la conexión se trabaja con una serie de comprobaciones y ciclos para obtener la información para el anexo. Por último se trabaja con la clase Generar y se descarga el archivo xml.

```
try{
    conn=getTransactionConnection();
    ///*Información informante*///
    ...

    ///*Parte de Compras*///
    ...

    ///*Parte de ventas*///
    ...

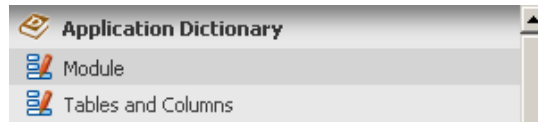
    ///*Parte de anulados*///
    ...

    ///*Para la descarga*///
    Generar generar=new Generar();
    byte[] bytes=generar.generar(iva);
    BufferedOutputStream output = null;
    if(bytes!=null){
        ByteArrayInputStream input = new ByteArrayInputStream(bytes);
        int contentLength = input.available();
        response.reset();
        response.setContentLength(contentLength);
        response.setContentType("text/xml");
        response.setHeader("Content-disposition", "attachment; filename=\""+
            nombreArchivo+ ".xml");
        output =newBufferedOutputStream(response.getOutputStream());
        while (contentLength-- > 0) {
            output.write(input.read());
        }
        output.flush();
        input.close();
        output.close();
    }
}
catch(Exception e){
    log("Error al conectar a la base de datos");
}
}
```

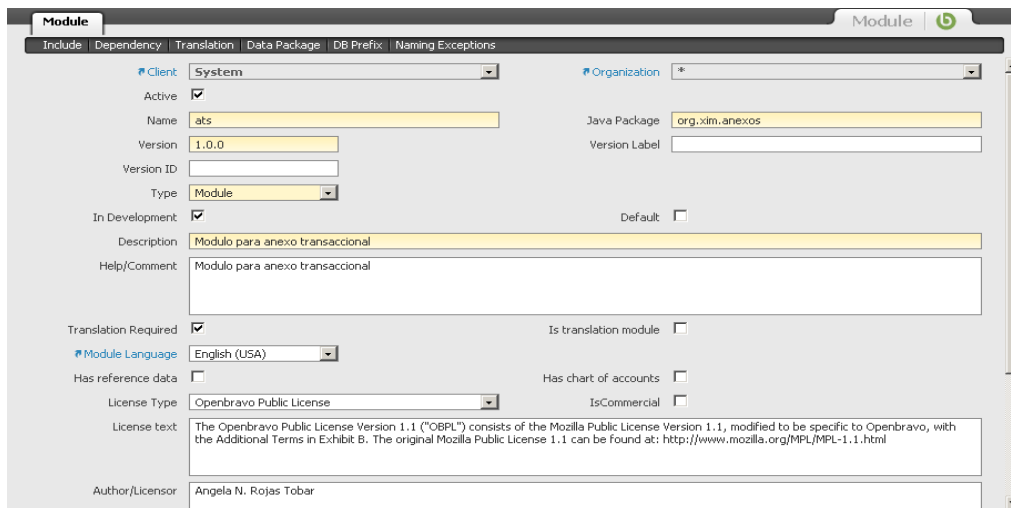
Vista

La vista que es el formulario AnexoT, lo creé siguiendo el estándar de desarrollo de Openbravo.

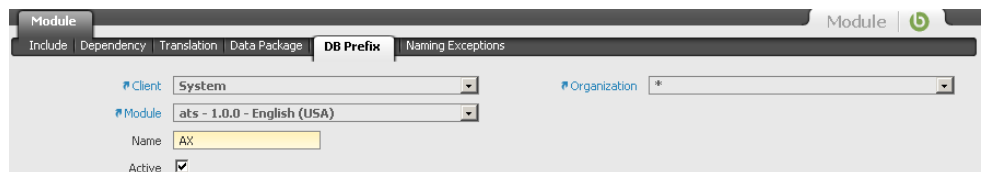
Primeramente se debe crear un módulo, esto lo hice dentro del Application Dictionary opción *Module*.



Dentro del formulario que se visualiza se hace clic en el ícono de nuevo registro y tenemos un formulario. Dentro de este se llenan los respectivos campos, el nombre del módulo es *ats* y el paquete de java que se usará para el módulo es *org.xim.anexos*, la versión es la *1.0.0*, el tipo es *Module*, una breve descripción en la que indico que es para el anexo transaccional, la licencia que se empleará en el módulo es la licencia pública de Openbravo y por último el nombre del autor.

A screenshot of the 'Module' form in the Application Dictionary. The form is titled 'Module' and has several tabs: 'Include', 'Dependency', 'Translation', 'Data Package', 'DB Prefix', and 'Naming Exceptions'. The 'Include' tab is active. The form contains the following fields and values: Client: System; Organization: *; Active: checked; Name: ats; Version: 1.0.0; Version ID: (empty); Type: Module; In Development: checked; Description: Modulo para anexo transaccional; Help/Comment: Modulo para anexo transaccional; Translation Required: checked; Module Language: English (USA); Is translation module: unchecked; Has reference data: unchecked; License Type: Openbravo Public License; IsCommercial: unchecked; License text: The Openbravo Public License Version 1.1 ("OBPL") consists of the Mozilla Public License Version 1.1, modified to be specific to Openbravo, with the Additional Terms in Exhibit B. The original Mozilla Public License 1.1 can be found at: http://www.mozilla.org/MPL/MPL-1.1.html; Author/Licensor: Angela N. Rojas Tobar.

Se debe crear un prefijo al módulo en el *tab DBPrefix*, este prefijo lo que permitirá es que al crear las tablas y sus campos o agregar campos a tablas existentes y antepone el prefijo al nombre, al compilar, el *framework* reconocerá que esa tabla o campo es parte de cierto módulo. Para mi módulo el prefijo que he nombrado es AX.

A screenshot of the 'DB Prefix' tab in the 'Module' form. The form is titled 'Module' and has several tabs: 'Include', 'Dependency', 'Translation', 'Data Package', 'DB Prefix', and 'Naming Exceptions'. The 'DB Prefix' tab is active. The form contains the following fields and values: Client: System; Organization: *; Module: ats - 1.0.0 - English (USA); Name: AX; Active: checked.

Una vez creado el módulo, todos los cambios que haga serán dentro del módulo *ats*, en los tutoriales de la comunidad de Openbravo sugieren que no se hagan cambios dentro del *core* (núcleo del sistema) de Openbravo. Guardé los cambios del módulo que he creado y ejecuté la tarea *ant export.database*.

Luego debí crear la tabla *ax_anexot* (con los campos para el formulario) en la base de datos, esta tabla se relaciona con la ventana. Luego subí la tabla al Application Dictionary del Openbravo y ejecuté la tarea *ant export.database*.

The screenshot shows the 'Table' configuration interface in the Openbravo Application Dictionary. The 'Tables and Columns' tab is active. The configuration includes:

- Client:** System
- Organization:** *
- Data Package:** Xim Anexos Data Package
- Name:** AX_AnexoT
- Description:** Anexo transaccional
- Help/Comment:** Tabla para el anexo transaccional
- Active:**
- DB Table Name:** AX_AnexoT
- View:**
- Java Class Name:** AXAnexoT
- Data Access Level:** All
- Type Area:** (empty)
- Window:** AnexoT
- PO Window:** (empty)
- Deletable Records:**
- Fully Audited:**
- High Volume:**
- Development Status:** Ready
- Acct date column:** (empty)
- Acct class name:** (empty)

A green button labeled 'Create Columns from DB' is visible at the bottom of the form.

Luego se relaciona la tabla a una nueva ventana, la cual creé también dentro del Application Dictionary del Openbravo. Esta es la modularidad que ofrece Openbravo a todas las personas que deseen contribuir con el proyecto.

The screenshot shows the 'Window' configuration interface in the Openbravo Application Dictionary. The 'Windows, Tabs, and Fields' tab is active. The configuration includes:

- Client:** System
- Organization:** *
- Module:** ats - 1.0.0 - English (USA)
- Name:** AnexoT
- Description:** Anexo transaccional
- Help/Comment:** Ventana usada para el anexo transaccional
- Active:**
- Window Type:** Maintain
- Sales Transaction:**
- Image:** (empty)
- Default:**

A green button labeled 'Copy Window Tabs' is visible at the bottom of the form.

Para el botón dentro del formulario, se lo hace también dentro del Application Dictionary. Dentro del menú seleccionamos *Tables and Columns*.

Dentro de este seleccioné la tabla que creé anteriormente *AX_AnexoT* y voy al *tab* de las columnas. Dentro del *tab* de columnas selecciono el campo *AX_Genera* que será el botón dentro del formulario.

Dentro de este formulario en el campo *Reference* seleccioné el ítem *Button* (esta opción permite visualizar el botón en el formulario) y dentro del campo *Process* seleccioné *AX_GenerarAnexo* el cual es un proceso que he creado dentro del *Application Dictionary* de Openbravo.

Para crear el proceso *AX_GenerarAnexo*, dentro del *Application Dictionary* seleccioné la opción *Report and Process*. Hacemos clic en el ícono de nuevo registro y damos un nombre al nuevo proceso. El estándar de Openbravo obliga a que los desarrolladores siempre trabajen con el prefijo que se nombró al módulo, en mi caso AX seguido del nombre del proceso.

Luego de llenar los campos, guardamos haciendo clic en el ícono del disquete y vamos a la *tab Process Class*.

Dentro del campo *Java Class Name* ponemos la ubicación de la clase que se debe ejecutar al hacer clic en el botón. En este caso se ejecutará el *ServletDescarga*.

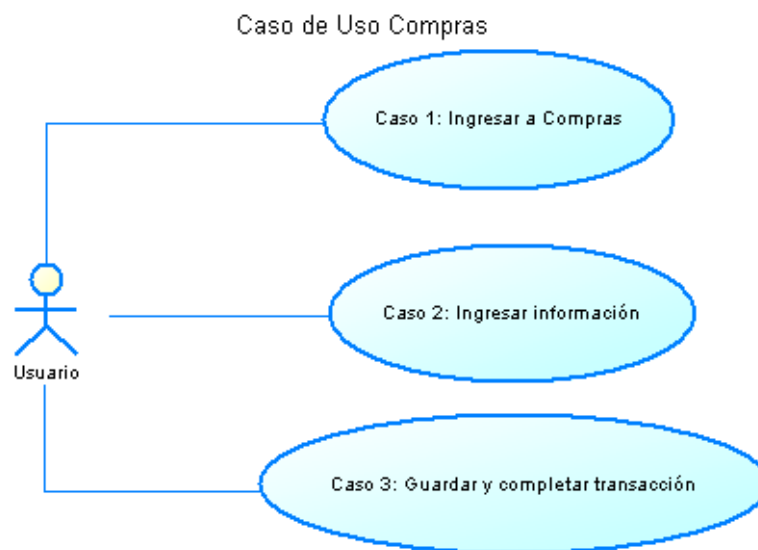
Luego de haber realizado todos los cambios se debe compilar la ventana ejecutando la tarea *ant compile.development -Dtab=AnexoT* al finalizar la ejecución se debe visualizar el mensaje *Build Successfull*.

Casos de Uso.

Caso de uso: COMPRAS

Descripción.

El usuario ingresa al módulo Gestión de Compras para registrar las compras de la empresa. Dentro de este escogemos la opción Transacciones >> Factura Proveedor y crea un nuevo registro para ingresar una compra. En este formulario se ingresa el proveedor, el tipo de comprobante (en este caso la factura), detalle de la factura y la retención.



Flujo de Eventos.

Flujo Básico.

Ingresar a Compras.

En el menú principal de la aplicación seleccionamos *Gestión de Compras* y luego damos clic en *Transacciones >> Factura Proveedor*.

Ingresar información.

El módulo Gestión de Compras nos presenta la siguiente ventana que es la cabecera de la factura. Dentro del tab **Cabecera** debemos llenar los campos:

- **Entidad y Organización** (que por default tiene el nombre de la empresa).
- **No. documento:** es el número secuencial de la factura.
- **No. Serie comprobante – establecimiento:** es el número de serie del establecimiento que tiene el comprobante de compra. Consta de tres dígitos.
- **No. Serie comprobante – punto de emisión:** es el número de serie del punto de emisión que tiene el comprobante de compra.

- **No. autorización del comprobante:** es el número de autorización que otorga SRI para la utilización del comprobante.
- **Sustento tributario:** se enlista los conceptos para el comprobante de compra, se debe seleccionar uno.
- **Documento de transacción:** se elige un documento para la transacción de compra en este caso AP Invoice que es el documento Factura.
- **Fecha Factura:** la fecha de facturación.
- **Fecha Impuesto:** la fecha de la vigencia del impuesto.
- **Fecha Contable:** la fecha de contabilización del documento.
- **Tercero:** el proveedor. Al seleccionar el proveedor automáticamente se llena el campo con la dirección del mismo.
- **Lista de Precio:** la lista de precios con la que realizará la transacción.
- **Forma de pago:** la forma de pago de la transacción.
- **Términos de pago:** los términos de pago de la transacción.
- **Retención:** este campo se lo utiliza cuando se va a realizar una retención en el impuesto a la renta. Cuando se ha seleccionado la retención se debe llenar los campos:
 - No. secuencial de comprobante de retención
 - No. serie comprobante de retención – establecimiento
 - No. serie comprobante de retención – punto de emisión
 - No. de autorización del comprobante de retención.
 - Fecha de la retención.

Los siguientes campos sólo se usarán cuando se esté trabajando con un documento de transacción AP Credit Invoice (Nota de credito): **No. secuencial del comprobante modificado, No. establecimiento del comprobante modificado, No. punto emisión del comprobante modificado y No. de autorización del comprobante modificado.**

The screenshot shows a 'Purchase Invoice' form with the following fields and values:

- Client:** Leche
- Organization:** Distribucion de Leche Andina
- Purchase Order:** (empty)
- Document No.:** <1000002>
- Invoice Date:** 30-11-2010
- Accounting Date:** 30-11-2010
- Tax Date:** 30-11-2010
- Price List:** lista compra
- Currency:** USD
- No. serie comprobante - establecimiento:** 000
- No. serie comprobante - punto de emision:** 000
- No. autorización comprobante:** (empty)
- Sustento tributario:** (empty)
- Order Date:** (empty)
- Order Reference:** (empty)
- Partner Address:** (empty)
- Active:**

Dentro del tab **Líneas** ingresamos el detalle de la factura, lo que compraremos. Debemos llenar los siguientes campos:

- **Producto:** el producto que compraremos
- **Cantidad:** la cantidad de producto.
- **Impuesto:** seleccionamos el impuesto del producto. Hay que tomar en cuenta la relación del producto con el impuesto que se ha configurado para no tener ningún error al guardar.

Guardar y completar transacción.

Para guardar la transacción de compra hacemos clic en el ícono del disquete y luego debemos visualizar el mensaje “1 fila/s actualizada” en un cuadro verde. En caso que lo visualicemos en un cuadro rojo nos indica un posible error por el mal ingreso de información.

Por último debemos completar la transacción de compra, para esto hacemos clic en el botón *Complete* de la parte inferior del formulario. Luego deberemos visualizar el mensaje “Proceso completado exitosamente” en un cuadro verde.

Pre condiciones

1. Debe estar creada la empresa en el sistema con la respectiva información y su plan de cuentas.
2. Se debe crear los tipos de documentos y llenar obligatoriamente el campo *descripción* con el número correspondiente.
3. Se debe crear a los proveedores dentro del módulo con su respectiva información como nombre del proveedor, número de RUC, nombre del contacto, dirección, retención, etc.
4. Se debe crear los tipos de impuesto dentro del módulo “módulo” detallando la información del impuesto como el nombre, categoría, porcentaje.
5. Se debe crear la o las listas de precios dentro del módulo con las que trabaja la empresa.
6. Se debe crear los productos de la empresa dentro del módulo “modulo” con la información requerida como categoría, nombre, precio, impuestos, proveedor, etc.
7. Si se trabaja con un documento AP Credit Memo (Nota de crédito) se debe ingresar la información del comprobante modificado en los campos antes mencionados.

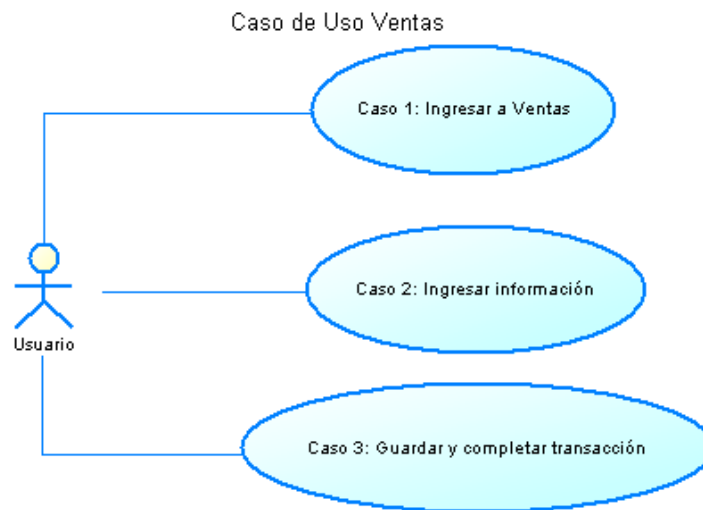
Pos condiciones.

1. Cuando se selecciona un producto dentro del detalle de la factura, este debe tener el mismo impuesto en el cual se especificó al crear el producto.
2. Una factura que no ha sido completada puede ser eliminada.
3. Una factura que ha sido anulada no puede ser reactivada.

Caso de uso: VENTAS

Descripción

El usuario ingresa al módulo Gestión de Ventas para registrar las ventas de la empresa. Dentro de este la escogemos la opción Transacción >> Factura cliente y crea un nuevo registro para ingresar una venta. En este formulario se ingresa el cliente, tipo de comprobante (en este caso la factura), detalle de la factura y la retención.



Flujo de Eventos.

Flujo Básico.

Ingresar a Ventas.

En el menú principal de la aplicación seleccionamos el módulo Gestión de Ventas y damos clic en la opción Transacciones >> Factura cliente.

Ingresar información.

Como podemos ver el formulario para realizar las ventas es muy similar al de las compras, dentro del tab **Cabecera** debemos llenar los siguientes campos:

- **Entidad y Organización** (que por default tiene el nombre de la empresa).
- **No. documento:** es el número secuencial de la factura.
- **No. Serie comprobante – establecimiento:** es el número de serie del establecimiento que tiene el comprobante de compra. Consta de tres dígitos.
- **No. Serie comprobante – punto de emisión:** es el número de serie del punto de emisión que tiene el comprobante de compra.

- **No. autorización del comprobante:** es el número de autorización que otorga SRI para la utilización del comprobante.
- **Documento transacción:** se elige un documento para la transacción de compra en este caso AR Invoice que es el documento Factura.
- **Fecha factura:** la fecha de facturación.
- **Fecha impuesto:** la fecha de la vigencia del impuesto.
- **Fecha contabilidad:** la fecha de contabilización del documento.
- **Tercero:** el proveedor. Al seleccionar el proveedor automáticamente se llena el campo con la dirección del mismo.
- **Lista de precio:** la lista de precios con la que realizará la transacción.
- **Forma de pago:** la forma de pago de la transacción.
- **Términos de pago:** los términos de pago de la transacción.
- **Retención:** este campo se lo utiliza cuando se va a realizar una retención en el impuesto a la renta. Cuando se ha seleccionado la retención se debe llenar los campos:
 - No. secuencial de comprobante de retención
 - No. serie comprobante de retención – establecimiento
 - No. serie comprobante de retención – punto de emisión
 - No. de autorización del comprobante de retención.
 - Fecha de la retención.

The screenshot shows the 'Sales Invoice' header form. Key fields include:

- Client:** Leche
- Organization:** Distribucion de Leche Andina
- Transaction Document:** AR Invoice
- Invoice Date:** 30-11-2010
- Tax Date:** 30-11-2010
- Price List:** lista venta
- Form of Payment:** Cash
- Payment Terms:** Inmediato
- Currency:** USD
- Accounting Date:** 30-11-2010
- Document No.:** <2000002>
- No. serie comprobante - establecimiento:** 000
- No. serie comprobante - punto de emision:** 000
- No. autorización comprobante:** (empty)
- No. secuencial comprobante retencion:** 000000000
- No. serie comprobante retencion - establecimiento:** 000
- No. serie comprobante retencion - punto de emision:** 000
- No. autorización comprobante de retencion:** 000000000

Dentro del tab **Lineas** ingresamos el detalle de la factura, lo que se va a vender. Es muy similar al tab para el proceso de compra. Debemos llenar los siguientes campos:

- **Producto:** el producto que compraremos
- **Cantidad:** la cantidad de producto.
- **Impuesto:** seleccionamos el impuesto del producto. Hay que tomar en cuenta la relación del producto con el impuesto que se ha configurado para no tener ningún error al guardar.

Guardar y completar transacción.

Para guardar la transacción de venta hacemos clic en el ícono del disquete y luego debemos visualizar el mensaje “1 fila/s actualizada” en un cuadro verde. En caso que lo visualicemos en un cuadro rojo nos indica un posible error por el mal ingreso de información.

Por último debemos completar la transacción de venta, para esto hacemos clic en el botón *Complete* de la parte inferior del formulario. Luego deberemos visualizar el mensaje “Proceso completado exitosamente” en un cuadro verde.

Pre condiciones

1. Debe estar creada la empresa en el sistema con la respectiva información y su plan de cuentas.
2. Se debe crear los tipos de documentos y llenar obligatoriamente el campo *description* (*descripción*) con el número correspondiente.
3. Se debe crear a los clientes dentro del módulo con su respectiva información como nombre del cliente, número de RUC o cédula de identidad, nombre del contacto, dirección, retención, etc.
4. Se debe crear los productos de la empresa dentro del módulo con la información requerida como categoría, nombre, precio, impuestos, proveedor, etc.
5. Deben existir productos vendibles dentro del stock de la empresa.
6. Se debe crear la o las listas de precios dentro del módulo con las que trabaja la empresa para realizar las ventas.

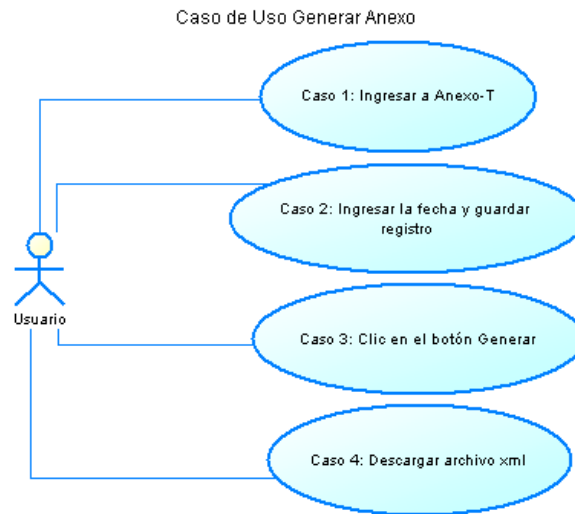
Pos condiciones.

1. Cuando se selecciona un producto dentro del detalle de la factura, este debe tener el mismo impuesto en el cual se especificó al crear el producto.
2. Una factura que no ha sido completada puede ser eliminada.
3. Una factura que ha sido anulada no puede ser reactivada.

Caso de uso: GENERAR ANEXO

Descripción.

El usuario ingresa al módulo Anexo-T para generar el anexo transaccional. Previo a la generación del anexo, la empresa debe tener ingresado las compras y ventas. La información del anexo será filtrada por la fecha que se especifique en el formulario.



Flujo de Eventos.

Flujo Básico.

Ingresar a Anexo-T.

En el menú principal de la aplicación seleccionamos hacemos clic en el módulo Anexo T.

Ingresar la fecha y guardar.

En la nueva ventana que presenta el módulo Anexo T debemos crear un nuevo registro para la generación del anexo transaccional. Para hacerlo damos clic en el ícono de la hoja en blanco (Crear nuevo registro), esta nos visualizará otra venta para ingresar el nuevo registro.

En la siguiente ventana ingresamos los datos para el registro como son cliente, organización y fecha del anexo. Los campos que se pintan en color tomate son obligatorios.

Luego hacemos clic en el ícono del disquete (Guardar).

Clic en el botón Generar.

Una vez creado el registro podemos generar el anexo transaccional. Hacemos clic en el botón AX_GenerarAnexo. Al hacer clic en este botón ejecuta el método doPost() del ServletDescarga.java.



Descargar archivo xml.

Una vez generado el archivo se visualiza la ventana para guardar y descargar el archivo

Pre condiciones.

1. Debe estar ingresada la información necesaria de la empresa como RUC, razón social, identificación del representante legal y su nombre, RUC del contador.
2. Se debe configurar de acuerdo al formato establecido los tipos de documentos, impuestos y retenciones.
3. Antes de generar el anexo se debe guardar en el registro con la fecha de generación.
4. Debe existir compras registradas.
5. Debe existir ventas registradas.

Pos condiciones.

1. Si no existe los datos necesarios de la empresa como RUC, razón social, identificación del representante legal y su nombre, RUC del contador, en anexo se generará en blanco.
2. Si no existe compras o ventas registradas en el sistema, el módulo Anexo-T simplemente no generará esas etiquetas.

Requerimientos de Hardware

El sistema Openbravo más el módulo Anexo-T necesitan de un equipo con las siguientes características:

- Procesador: como mínimo Pentium 4
- Sistema Operativo: Windows o Linux
- Memoria RAM: como mínimo 1 GB.

Requerimientos de Software

Los requerimientos de software para el funcionamiento de Openbravo más el módulo Anexo-T son:

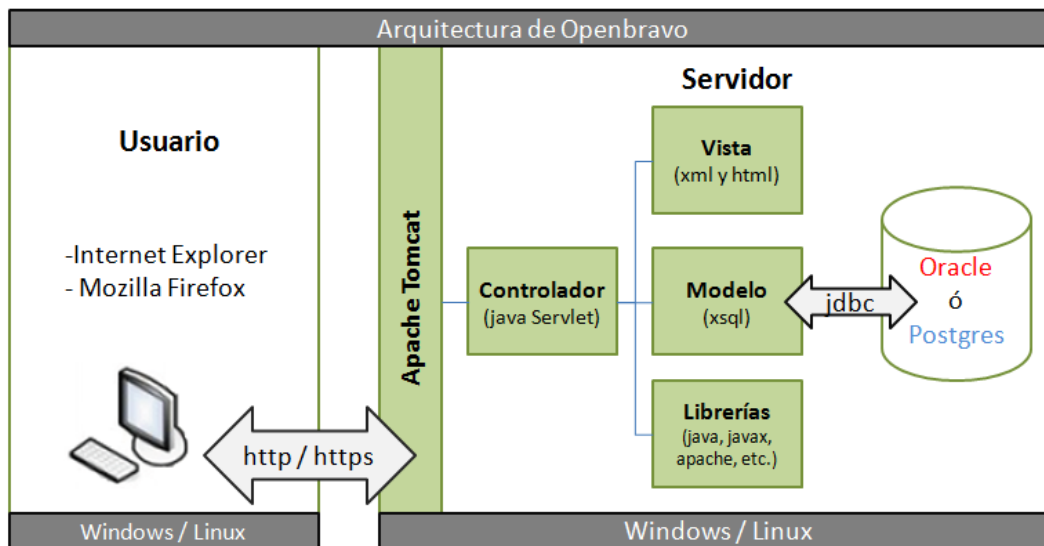
- PostgreSQL versión 8.4
- Sun Java Development Kit (JDK) versión 1.6
- Apache Tomcat versión 6.0.20
- Mozilla FireFox desde la versión 3.6.10 o Internet Explorer

Se deben establecer las variables de entorno de java y apache tomcat (ANT_OPTS, CATALINA_OPTS, JAVA_HOME). Adicionalmente dentro de la carpeta */lib* de tomcat debe agregarse el archivo *xstream.jar*

System Architecture Openbravo.

Openbravo is an application client / web server written in Java code. It runs on Apache Tomcat and has support for PostgreSQL databases and Oracle. It is a complete web application that has been developed following the MVC (Model, View, Controller), which facilitates the separation of the areas of development, enabling the implementation of sustainable growth and greater ease of code maintenance. Most of the code is automatically generated by the engine called WAD (Wizard for Application Development), based on the information contained in the dictionary of the data model (Model Data Dictionary). The engine runs WAD and recompiles the application every time the administrator changes the settings to suit a new requirement.

- Model: XSQL files found in the SQL statements which will be executed.
- View: HTML and XML files that define the forms and templates where the data will be inserted.
- Controller: Java Servlets in defining the actions within the forms.



MVC-FF.

MVC Foundation Framework (MVC-FF) comprises the set of utilities developed by Openbravo as XmlEngine, and HttpBaseSecureServlet SQLC. MVC-FF is necessary to enable the development and generation of files for the model, view and controller.

XmlEngine.

It is a service used to create documents XML / HTML from a template in XML / HTML and an XML configuration file with the dynamic data to be inserted in the template. The configuration file maps the data source with the sites identified in the template. The templates are read and stored in memory, then when a page is requested. The template creates a document which is filled with data obtained by the application. This tool generates Openbravo forms, create reports or print them.

SQLC.

SQL Compiler is a service that lets you interact with the database. The interaction occurs through an XML file that contains standard SQL statements and parameters to be used in sentences. SQLC

reads the file and generates a Java class that contains all the necessary code to connect to the database, execute judgments.

HttpBaseServlet.

HttpBaseSecureServlet HttpBaseServlet are servlets and from which all the servlets in the legacy system. These servlets implement features such as authentication, authorization, connectivity to the database and errors.

Data Model Dictionary and WAD.

Code generation is possible by using Data Model Dictionary and the WAD (Wizard for Application Development). The WAD automatically generates all the files the application with the MVC model. The files are generated using XmlEngine, SQLC and HttpBaseSecureServlet.

Development tools.

The requirements for the development environment at the site suggested the community of Openbravo ERP, and which were employed:

- PostgreSQL version 8.4
- Sun Java Development Kit (JDK) version 1.6
- Apache Tomcat version 6.0.20
- Apache Ant version 1.8.0
- Eclipse Ganymede version 3.4.2
- Mozilla Firefox 3.6.10

Prior to the development must be set environment variables for java, apache tomcat and apache ant (ANT_OPTS, CATALINA_OPTS, JAVA_HOME). The source code can be downloaded from Openbravo official site or also can be downloaded from www.sourceforge.net I have worked with version 2.50.

Implementation of Anexo-T module.

Additional libraries.

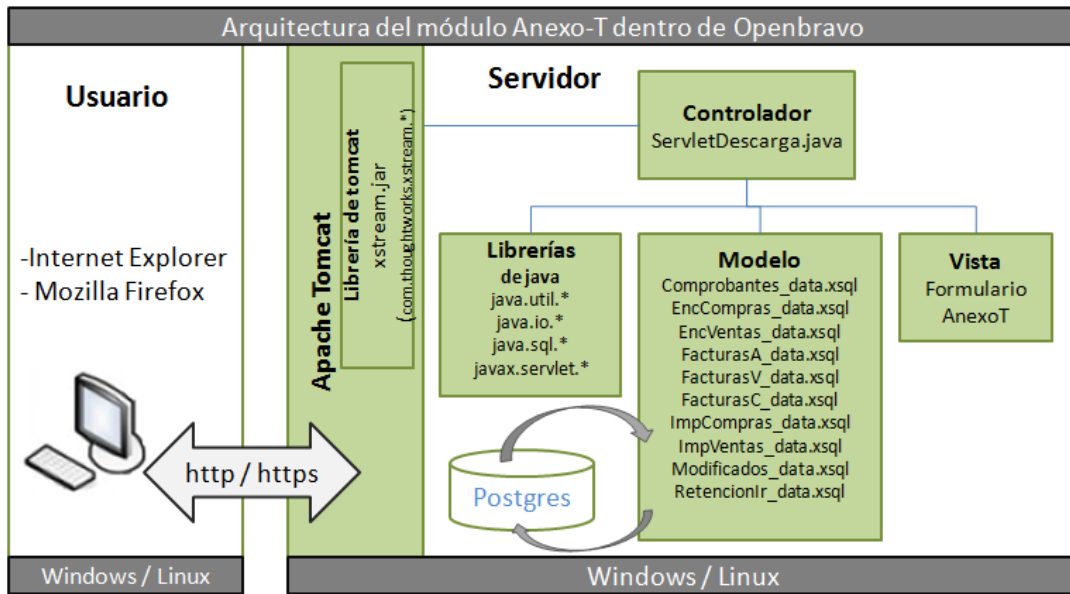
The operation of the Annex-G module depends on the existence of the XStream JAR in the / lib of tomcat apache.

Additional modifications to the project Openbravo.

Taking into account the characteristics of the milk distribution company DIPROMAC and requirements for Annex transactional Openbravo system did not have all the information for the generation of Annex transaction was therefore very necessary to add some changes to the system modules.

The changes are realized by Openbravo development framework, changes to the database and executing Ant tasks.

Architecture Anexo-T module.



Operation.

Additional Classes

Realized with the help of the following classes XStream JAR containing the respective java annotations @ XStreamAlias for the structure of the XML file will be generated:

- Iva.java
- DetalleAir.java
- DetalleAnulado.java
- DetalleCompra.java
- DetalleVenta.java

Generar.java class realized that generates the XML by XStream and then return a byte vector with the information in annex.

```
public class Generar {

public static final String CABECERA_XML="<?xml version=\"1.0\" encoding=\"ISO-8859-1\"
standalone=\"yes\"?>\n";

public Generar (){}

public byte[] generar(Iva iva){
XStream stream = new XStream(new DomDriver());
stream.processAnnotations(Iva.class);
try {
String xml=CABECERA_XML.concat(stream.toXML(iva));
return xml.getBytes();
}
catch(Exception ex){
ex.printStackTrace();
return null;
}
}
}
```

Model

Then I did xSQL files that allow us to work with the database. Once created these files to make the ant smartbuild SQLC tool (SQL Compiler is part of the framework of Openbravo) generates Java classes from xSQL files to facilitate the work with the database. To create the files must be respected xSQL standard with the working Openbravo, the file name must end with `_data`. Therefore, each file has its respective class xSQL java, the java class name is set forth in the file xSQL:

- `Comprobantes_data.xsql` (`ComprobantesData.java`)
- `EncCompras_data.xsql` (`EncComprasData.java`)
- `EncVentas_data.xsql` (`EncVentasData.java`)
- `FacturasA_data.xsql` (`FacturasAData.java`)
- `FacturasC_data.xsql` (`FacturasCData.java`)
- `FacturasV_data.xsql` (`FacturasVData.java`)
- `ImpCompras_data.xsql` (`ImpComprasData.java`)
- `ImpVentas_data.xsql` (`ImpVentasData.java`)
- `Modificados_data.xsql` (`ModificadosData.java`)
- `RetencionIr_data.xsql` (`RetencionIrData.java`)

Below shows the structure of a file in this case xSQL `Enc_Compras_data.xsql` file which contains the query to the database:

```
<?xml version="1.0" encoding="UTF-8"?>
<SqlClass name="EncComprasData" package="org.xim.anexos.consulta">
  <SqlClassComment>Class EncComprasData</SqlClassComment>
  <SqlMethod name="select" type="preparedStatement" return="multiple">
    <SqlMethodComment>Select encabezado factura</SqlMethodComment>
    <Sql><![CDATA[
      select c.em_ax_sustento as em_ax_sustento,
      b.em_ax_tipoid as em_ax_tipoid,
      b.em_ax_ciruc as em_ax_ciruc,
      c.c_doctype_id as c_doctype_id,
      to_date(c.dateinvoiced) as dateinvoiced,
      c.em_ax_nestablecimiento as em_ax_nestablecimiento,
      c.em_ax_npuntoemision as em_ax_npuntoemision,
      c.documentno as documentno,
      c.em_ax_nautorizacion as em_ax_nautorizacion
      from c_invoice c join c_bpartner b on c.c_bpartner_id=b.c_bpartner_id
      where c.c_invoice_id=?
    ]]></Sql>
    <Parameter name="paraminvoiceid"></Parameter>
  </SqlMethod>
</SqlClass>
```

The label is defined `<SqlClass>` certain attributes like name that is for the java class name that generated when you compile and package it for the name of the package where the Java class is generated.

The label `<SqlClassComent/>` where we define a comment for class java.

The label attribute defines `<SqlMethod>` name is the name of the method within the Java class, type defines the type of query execution (constant, `PreparedStatement`, `Statement`, `CallableStatement`) and return defines the return of a or multiple rows (multiple, single).

The label is defined `<SqlMethodComment/>` a comment for the method.

The label `<sql> <![CDATA [...]]></Sql>` SLQ statement is defined for the query to the database.

`<Parameter/>` The label is defined in the attribute name the name of the parameter to the query.

Controller.

The controller is the `ServletDescarga.java` extending class `HttpSecureAppServlet` (`HttpSecureAppServlet` Openbravo is a kind of). In the `doPost ()` is stored in String variables the basic parameters for the queries to the database. The parameters are client (Client), orgid (Organization) and date. These parameters are sent to `anexoAT` method ().

```
public void doPost(HttpServletRequest request, HttpServletResponse response) throws
IOException, ServletException {
    VariablesSecureApp vars = new VariablesSecureApp(request);

    if (vars.commandIn("DEFAULT")) {
        String clienteid=vars.getStringParameter("inpadClientId");
        String orgid=vars.getStringParameter("inpadOrgId");
        String fecha=vars.getStringParameter("inpaxFecha");

        if (clienteid !=null && orgid!=null && fecha!=null){
            anexoAT(response, vars, orgid, clienteid, fecha);
        }
        else
            pageError(response);
    }
}
```

The package was imported `javax.servlet` `org.xim.anexos.clasexml`, this package contains the Java classes with annotations for the respective XML file structure of the Annex (Iva, DetalleAir, DetalleCompra, DetalleVenta, DetalleAnulado). The method `anexoAT` (`HttpServletResponse` response, vars `VariablesSecureApp`, orgid String, String client, String date) works with the objects below:

```
Iva iva=new Iva();
List <DetalleCompra> detalleC=null;
List <DetalleVenta> detalleV=null;
List <DetalleAir> detalleAir=null;
List <DetalleAnulado> detalleA=null;
DetalleCompra dc=null;
DetalleVenta dv=null;
DetalleAir dair=null;
DetalleAnulado da=null;
```

It also works with an object to connect to the database:

```
Connection conn=null
```

It also has the following variables that will help me store the number of RUC, the name, get the month and year to filter information from purchases and sales, create a name for the XML file, identify purchases, sales and state identification and one more variable to identify whether or not a customer.

```
String numeroRuc="";
String razonSocial="";
String []fechaAT=fecha.split("-");
String mesAT=fechaAT[1];
String anioAT=fechaAT[2];
```

```
String nombreArchivo="AT"+mesAT+anioAT;
String compra="N";
String venta="Y";
String estadoFac="CO";
String anulado="VO";
String isCli="Y";
```

Within a try making the connection to the database using the method described below which is the kind `HttpSecureAppServlet`:

```
conn=getTransactionConnection();
```

This variable "conn" allow us to work with Java classes that were generated by SQLC so we can make queries to the database. After getting the connection working with a series of checks and cycles to get the information for the annex. Finally, working with the class generate and download the xml file.

```
try{
    conn=getTransactionConnection();
    ///Información informante///
    ...

    ///Parte de Compras///
    ...

    ///Parte de ventas///
    ...

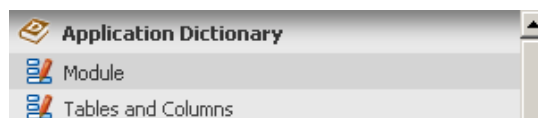
    ///Parte de anulados///
    ...

    ///Para la descarga///
    Generar generar=new Generar();
    byte[] bytes=generar.generar(iva);
    BufferedOutputStream output = null;
    if(bytes!=null){
        ByteArrayInputStream input = new ByteArrayInputStream(bytes);
        int contentLength = input.available();
        response.reset();
        response.setContentLength(contentLength);
        response.setContentType("text/xml");
        response.setHeader("Content-disposition", "attachment; filename=\""+
            nombreArchivo+ "\".xml");
        output =newBufferedOutputStream(response.getOutputStream());
        while (contentLength-- > 0) {
            output.write(input.read());
        }
        output.flush();
        input.close();
        output.close();
    }
}
catch(Exception e){
    log("Error al conectar a la base de datos");
}
```

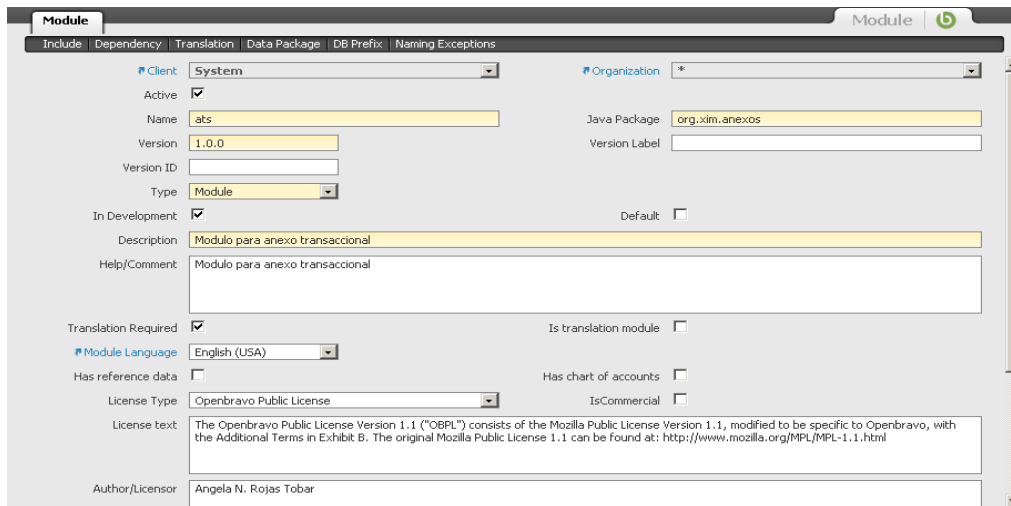
View.

The view is the form AnexoT, I created following the standard development of Openbravo.

First you must create a module, I do it in the Application Dictionary Module option.

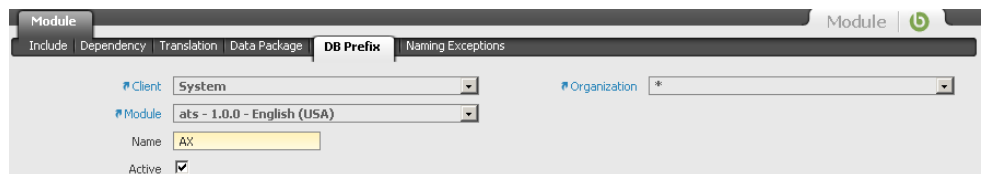


In the form that is displayed is clicked the icon again and have a registration form. Within this fill the respective fields, the module name is ats and java package that will be used for the module is org.xim.anexos, the version is 1.0.0, Module type is a brief description indicating that it is for Annex transaction, the license to be used in the module is the Openbravo public License, and finally the author's name.



The screenshot shows the 'Module' configuration form in Openbravo. The form is divided into several sections: Client (System), Organization (*), Active (checked), Name (ats), Version (1.0.0), Version ID, Type (Module), In Development (checked), Default (unchecked), Description (Modulo para anexo transaccional), Help/Comment (Modulo para anexo transaccional), Translation Required (checked), Is translation module (unchecked), Module Language (English (USA)), Has reference data (unchecked), Has chart of accounts (unchecked), License Type (Openbravo Public License), IsCommercial (unchecked), License text (The Openbravo Public License Version 1.1 ("OBPL") consists of the Mozilla Public License Version 1.1, modified to be specific to Openbravo, with the Additional Terms in Exhibit B. The original Mozilla Public License 1.1 can be found at: <http://www.mozilla.org/MPL/MPL-1.1.html>), and Author/Licensor (Angela N. Rojas Tobar).

You must create a prefix to the module in DBprefix tab, allowing this prefix is to create the tables and their fields or add fields to existing tables, and we put the prefix to the name, when compiling the framework recognizes that table or field is part of a module. For my module I have named the prefix is AX.



The screenshot shows the 'DB Prefix' configuration form in Openbravo. The form is divided into several sections: Client (System), Organization (*), Module (ats - 1.0.0 - English (USA)), Name (AX), and Active (checked).

After creating the module, any changes you make will be within the module ats in tutorials Openbravo community suggest that no changes be made within the core (kernel) of Openbravo. I kept changing the module that I created and ran the ant export.database.

Then I had to create the table ax_anexot (with fields for the form) in the database, this table is related to the window. Then I climbed the table to the Application Dictionary of Openbravo and ran the ant export.database.

After the table is related to a new window, which also created within the Application Dictionary of Openbravo. This is the modularity that Openbravo offers to all people interested in contributing to the project.

For the button inside the form, so does within the Application Dictionary. In the menu select Tables and Columns.

Within this select the table I created earlier and go to the tab AX_AnexoT columns. In the columns tab select the field that will AX_Genera button within the form.

Within this form in the Reference field Button selected item (this option to display the button on the form) and within the selected field AX_GenerarAnexo Process is a process which I created in the Application Dictionary of Openbravo. To create the AX_GenerarAnexo process within the Application Dictionary Report on the menu bar and Process. We click on the icon new record and give a name to the new process. The standard of Openbravo requires that developers always work with the prefix was appointed to the module, in my case AX followed by the name of the process.

After filling the fields, keep clicking the disk icon and go to the Process Class tab.

Inside the Java Class Name field put the location of the class to run when you click the button. In this case will run ServletDescarga.

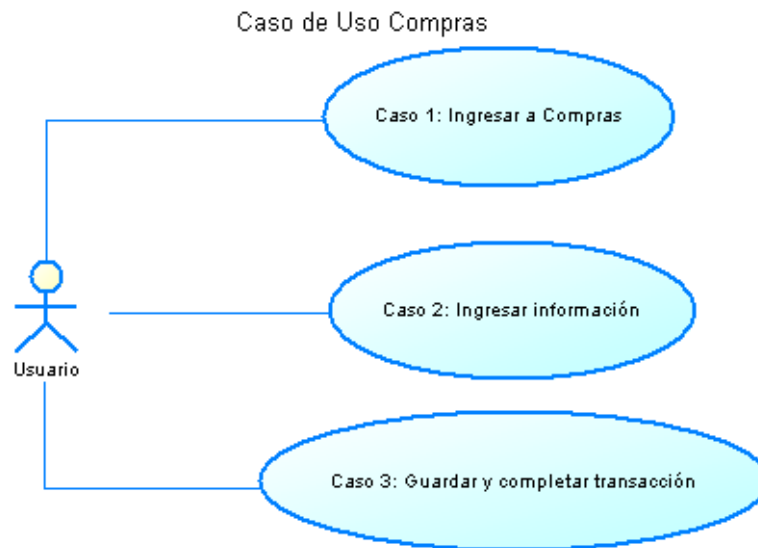
Having done all that you can compile the window running the ant-DTAB = AnexoT compile.development to finish running the message should be displayed Build Successful.

Use Case

Use Case: Purchase

Description.

The user enters the Procurement Manager module to record the purchases of the company. Within this option we choose the Transaction>> Purchase Invoice and make a new record to enter a purchase. This form enters the provider, type of proof (in this case the bill), detail of the bill and retention.



Flow of Events.

Basic Flow.

Sign Shopping.

In the main application menu, select Purchasing Management and then we click Transactions>> Billing Provider.

Login information.

The Purchasing Management module presents the following window is the head of the bill. In the Header tab should fill the fields:

- Entity and Organization (which by default has the name of the company).
- No. document: it is the sequential number of the invoice.
- Series No. Proof - establishment: the serial number of the establishment that has proof of purchase. It consists of three digits.
- Series No. Proof - point of issue: is the serial number of the emission point that has the receipt.
- Authorization No. Proof: The number of SRI authority granted for the use of the voucher.
- Support the tax: it lists the concepts for proof of purchase, you must select one.

- Transaction Document: Choosing a document to the purchase transaction in this case is the AP Invoice document.
- Invoice Date: The date of billing.
- Tax Date: The effective date of the tax.
- Accounting Date: The posting date of the document.
- Business Partner. Selecting the provider will automatically fill the address field of the same.
- List Price: the price list with which the transaction.
- Payment: the payment of the transaction.
- Terms of payment: the payment terms of the transaction.
- Retention: This field is used when you are making a withholding tax on income. When retention is selected fields must be completed:
 - No. sequential withholding
 - No. withholding series – introduction
 - No. withholding series - release point
 - No. authorization or withholding.
 - Date of retention.

The following fields are only used when working with a transaction document AP Credit Invoice (Credit Memo): No. sequential modified proof, no proof establishing the modified emission point No. modified and proof of authorization No. the modified proof.

Inside the Lines tab entered the details of the bill, we will buy. We must fill the following fields:

- Product: the product you buy
- Amount: The amount of product.
- Tax: select the product tax. We must take into account the relationship of the product with the tax is set to have no errors when saving.

Save and complete transaction.

To save the purchase transaction click on the icon of the floppy and then we display the message "1 row / s updated" in a green box. If you visualize a red box indicates a possible error for the wrong data entry.

Finally we complete the purchase transaction, for this click on Complete button at the bottom of the form. Then we display the message "Process completed successfully" in a green box.

Pre conditions

1. Must be created on a system with the respective information and chart of accounts.
2. You must create the kinds of documents and the description field must be filled with the corresponding number.
3. Must be created within the module suppliers with their respective information as vendor name, RUC number, contact name, address, retention, etc..
4. You must create the types of taxes within the module "module" detailing tax information such as name, category, percentage.
5. Should be created or price lists within the module works with the company.
6. You must create the company's products within the module "module" with the required information as a category, name, price, taxes, vendor, etc..
7. When working with AP Credit Memo document (credit memo) must enter the receipt information altered in areas mentioned above.

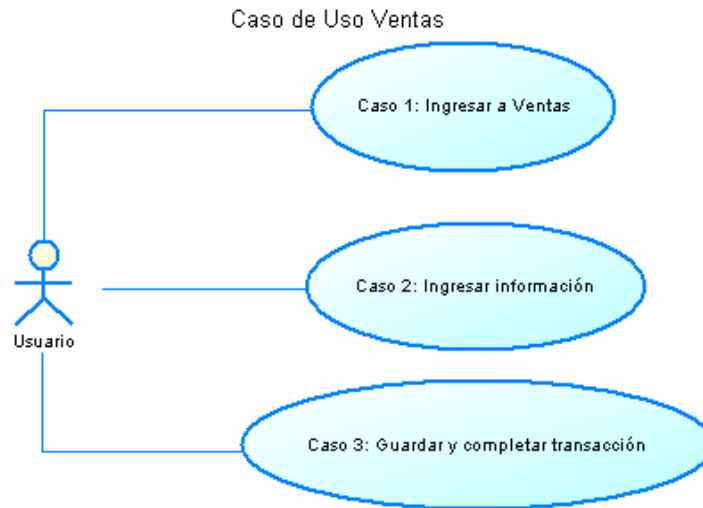
Pos conditions.

1. When a product is selected within the detail of the bill, it must have the same tax in which specified when creating the product.
2. A bill that has not been completed can be eliminated.
3. A bill has been canceled cannot be reactivated.

Use Case: SALES

Description

The user enters the Sales Management module to record sales of the company. In this transaction the option to choose the>> client bill creates a new record to enter a sale. In this form the customer is entered, type of proof (in this case the bill), detail of the bill and retention.



Flow of Events.

Basic Flow.

Login to Sales.

In the main application menu, select the Sales Management module and click on the option we Transactions>> Invoice customer.

Login information.

As we can see the form for sales is very similar to purchases within the Header tab should fill the following fields:

- Entity and Organization (which by default has the name of the company).
- No. document: it is the sequential number of the invoice.
- Series No. Proof - establishment: the serial number of the establishment that has proof of purchase. It consists of three digits.
- Series No. Proof - point of issue: is the serial number of the emission point that has the receipt.
- Authorization No. Proof: The number of SRI authority granted for the use of the voucher.
- Document Transaction: select a document for the purchase transaction in this case AR Invoice is the invoice document.
- Invoice date: date of billing.
- Tax Date: The effective date of the tax.
- Accounting Date: The posting date of the document.
- Third, the supplier. Selecting the provider will automatically fill the address field of the same.
- List price: the price list on which the transaction.
- Payment: the payment of the transaction.
- Terms of payment: the payment terms of the transaction.
- Retention: This field is used when you are making a withholding tax on income. When retention is selected fields must be completed:
 - No. sequential withholding
 - No. withholding series – introduction

- No. withholding series - release point
- No. authorization or withholding.
- Date of retention.

The screenshot shows a 'Sales Invoice' form with the following fields and values:

- Client:** Leche
- Organization:** Distribucion de Leche Andina
- Order Date:** (empty)
- Document No.:** <2000002>
- No. serie comprobante - establecimiento:** 000
- No. serie comprobante - punto de emision:** 000
- No. autorizacion comprobante:** (empty)
- Order Reference:** (empty)
- Description:** (empty text area)
- Transaction Document:** AR Invoice
- Invoice Date:** 30-11-2010 (with a green checkmark)
- Accounting Date:** 30-11-2010 (with a green checkmark)
- Tax Date:** 30-11-2010 (with a green checkmark)
- Business Partner:** (empty)
- Partner Address:** (empty)
- User/Contact:** (empty)
- Price List:** lista venta
- Sales Representative:** (empty)
- Currency:** USD
- Print Discount:**
- Payment Terms:** Inmediato
- Form of Payment:** Cash
- Withholding:** (empty)
- No. secuencial comprobante retencion:** 000000000
- No. serie comprobante retencion - establecimiento:** 000
- No. serie comprobante retencion - punto de emision:** 000
- No. autorizacion comprobante de retencion:** 000000000

Inside the Lines tab entered the details of the bill, which will sell. It is very similar to the tab for the shopping process. We must fill the following fields:

- Product: the product you buy
- Amount: The amount of product.
- Tax: select the product tax. We must take into account the relationship of the product with the tax is set to have no errors when saving.

Save and complete transaction.

To save the sales transaction we click the disk icon and then we display the message "1 row / s updated" in a green box. If you visualize a red box indicates a possible error for the wrong data entry.

Finally we complete the sales transaction, for this click on Complete button at the bottom of the form. Then we display the message "Process completed successfully" in a green box.

Pre conditions

1. Must be created on a system with the respective information and chart of accounts.
2. You must create the kinds of documents and Mandatory field description (description) with the corresponding number.
3. Should be created to customers within their respective module information such as customer name, number of RUC or ID, contact name, address, retention, etc.
4. You must create the company's products within the module with the required information as a category, name, price, taxes, vendor, etc.
5. There must be salable in the stock of the company.

6. Should be created or price lists within the module works with the company for sales.

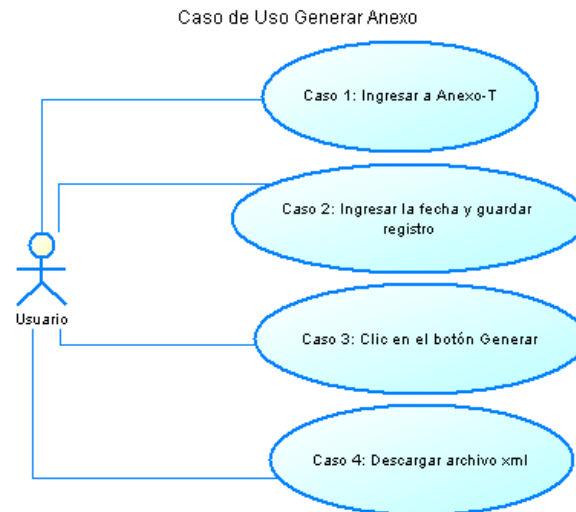
Pos conditions.

1. When a product is selected within the detail of the bill, it must have the same tax in which specified when creating the product.
2. A bill that has not been completed can be eliminated.
3. A bill has been canceled cannot be reactivated.

Use case: CREATE ANNEX

Description.

The user enters the module to generate the annex transactional. Prior to the generation of the annex, the company should have entered the purchases and sales. The information in annex will be filtered by the date specified on the form.



Flow of Events.

Basic Flow.

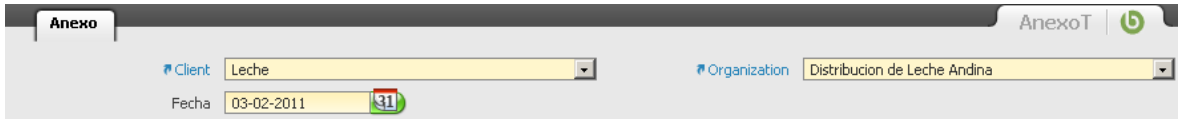
Sign in Annex-T.

In the main application menu select click on the module AnexoT.

Enter the date and save.

In the new window that displays the module AnexT we create a new record for the generation of annex transactional. To do this we click on the icon of the blank (Create new record), that we displayed another sale to enter the new record.

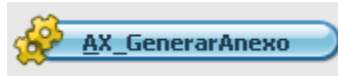
In the next window we enter the registration data such as customer, organization and date of the annex. Fields are painted tomato-red are mandatory.



Then click on the icon of the diskette (Save).

Click the Generate button.

After creating the record we can generate the transactional Annex. We AX_GenerarAnexo button. Clicking this button runs the doPost () method of ServletDescarga.java.



Download xml file.

Once the file a window is displayed to store and download the file

Pre conditions.

1. Must be entering the necessary information of the company as RUC name, legal representative's identification and his name, RUC counter.
2. Must be set according to the format established types of documents, taxes and deductions.
3. Before building the Annex should be kept on record with the date of generation.
4. You must be registered purchases.
5. You must be registered sales.

Post-conditions.

1. If there is the necessary data from the company as RUC name, legal representative's identification and his name, RUC counter in generic white annex.
2. If no purchases or sales recorded in the system, the Annex-T module simply will not generate such labels.

Hardware Requirements

The system module Openbravo most Anexo-T require a computer with the following characteristics:

- Processor: Pentium 4 at least
- Operating system: Windows or Linux
- RAM: at least 1 GB.

Software Requirements

The software requirements to run Openbravo plus Annex-G module are:

- PostgreSQL version 8.4
- Sun Java Development Kit (JDK) version 1.6
- Apache Tomcat version 6.0.20
- Mozilla Firefox from version 3.6.10 or Internet Explorer

Must set environment variables java and apache tomcat (ANT_OPTS, CATALINA_OPTS, JAVA_HOME). Also in the folder / lib must be added the file tomcat xstream.jar