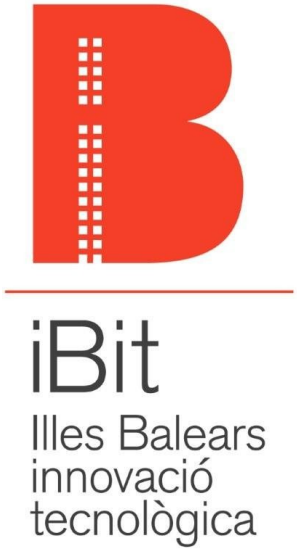


GestOli

Manual tècnic

Índice de contenido

1. Introducció

En aquest document s'exposa tota la informació tècnica per a entendre el desenvolupament de la aplicació Gest-OLI, amb la finalitat de que qualsevol persona amb coneixements tècnics i de programació en J2EE pugui realitzar qualsevol modificació/millora de la aplicació i implantar-la en producció.

Dividirem el document en quatre parts ben diferenciades:

- Disseny de base de dades
- Model de dades
- Desenvolupament
- Control de qualitat

2. Disseny de Base de Dades

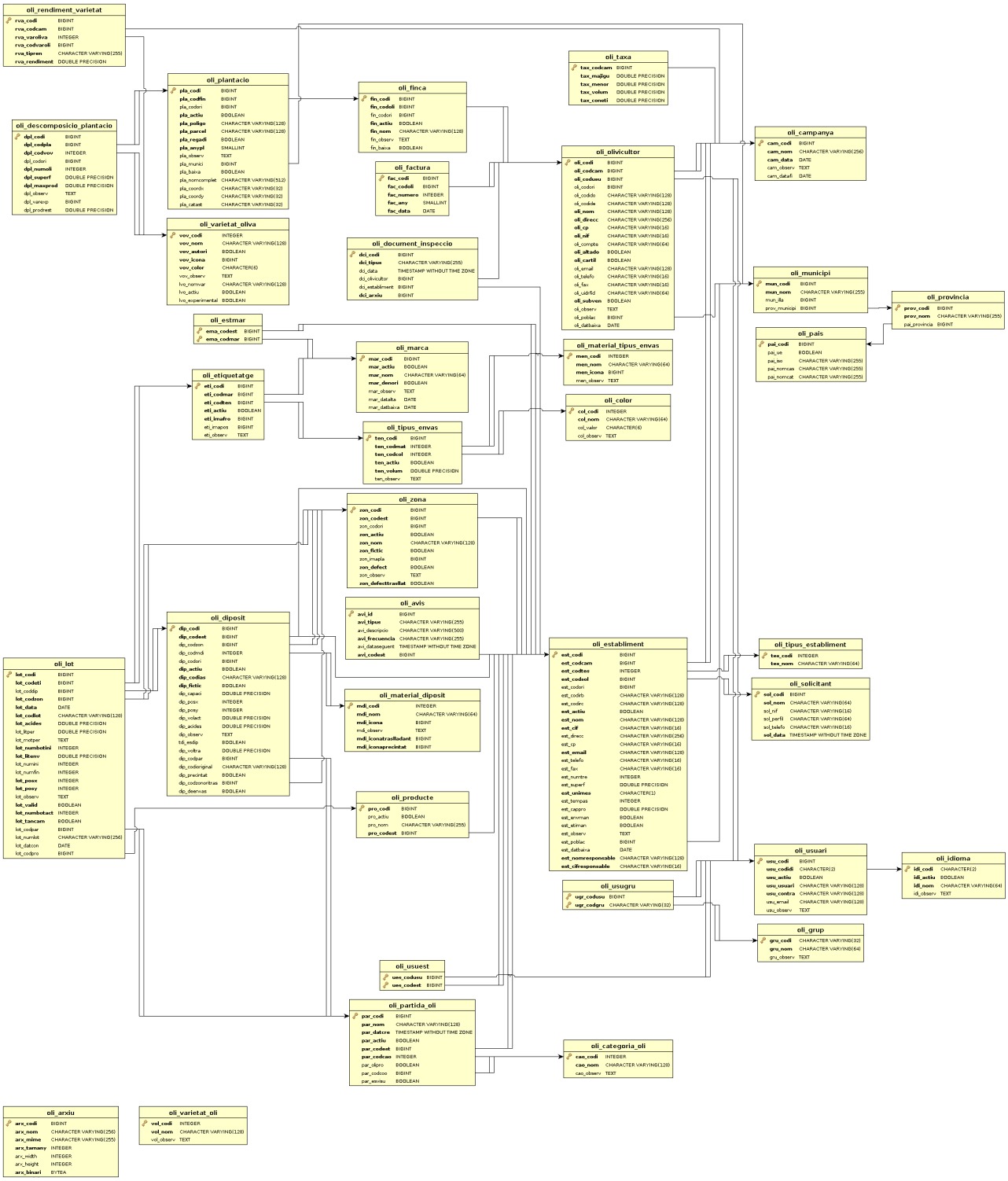
S'ha utilitzat **PostgreSQL 8.3.9** com a motor de base de dades, i **Hibernate3** com a eina de mapeig objecte-relacional per a la plataforma Java, que facilita el mapeig d'atributs entre una base de dades relacional tradicional i el model d'objectes d'una aplicació, mitjançant fitxers declaratius (XML) que permeten establir aquestes relacions.

Els esquemes ERD[1] corresponents amb el disseny, i agrupant per funcionalitat del programa, son els següents:

1 L'ERD en format PDF es poden trobar en ./docs/GEST-OLI – Document tècnic – ERD.pdf, on es pot ampliar per a veure-ho millor, i a més es pot veure un altre apartat, amb una vista general de tota la Base de Dades.
S'ha de tenir en compte que aquests apartats estan pensats per a ser visualitzats en pdf, i no en paper.

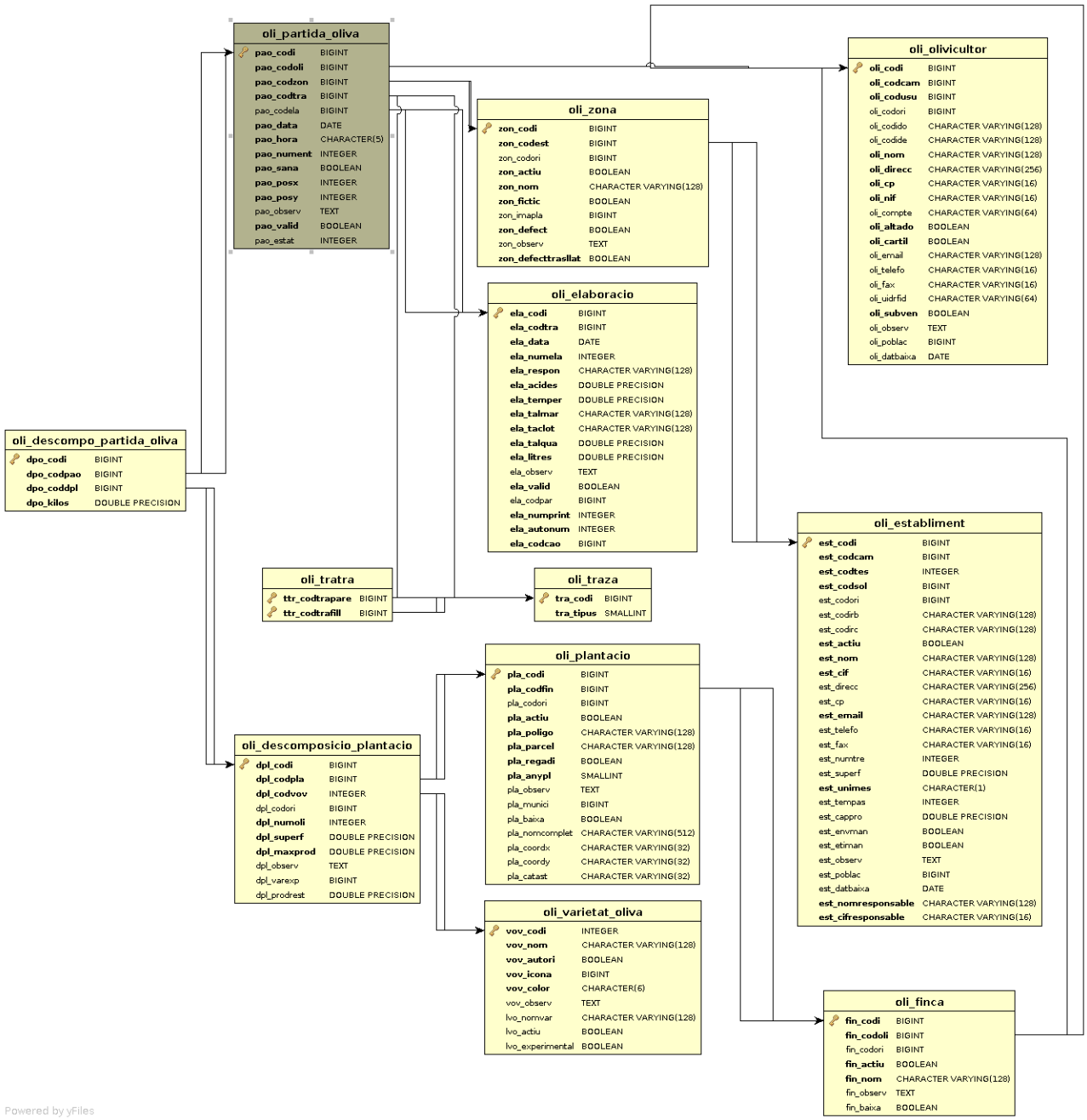
2.1. Configuració:

3.



4.

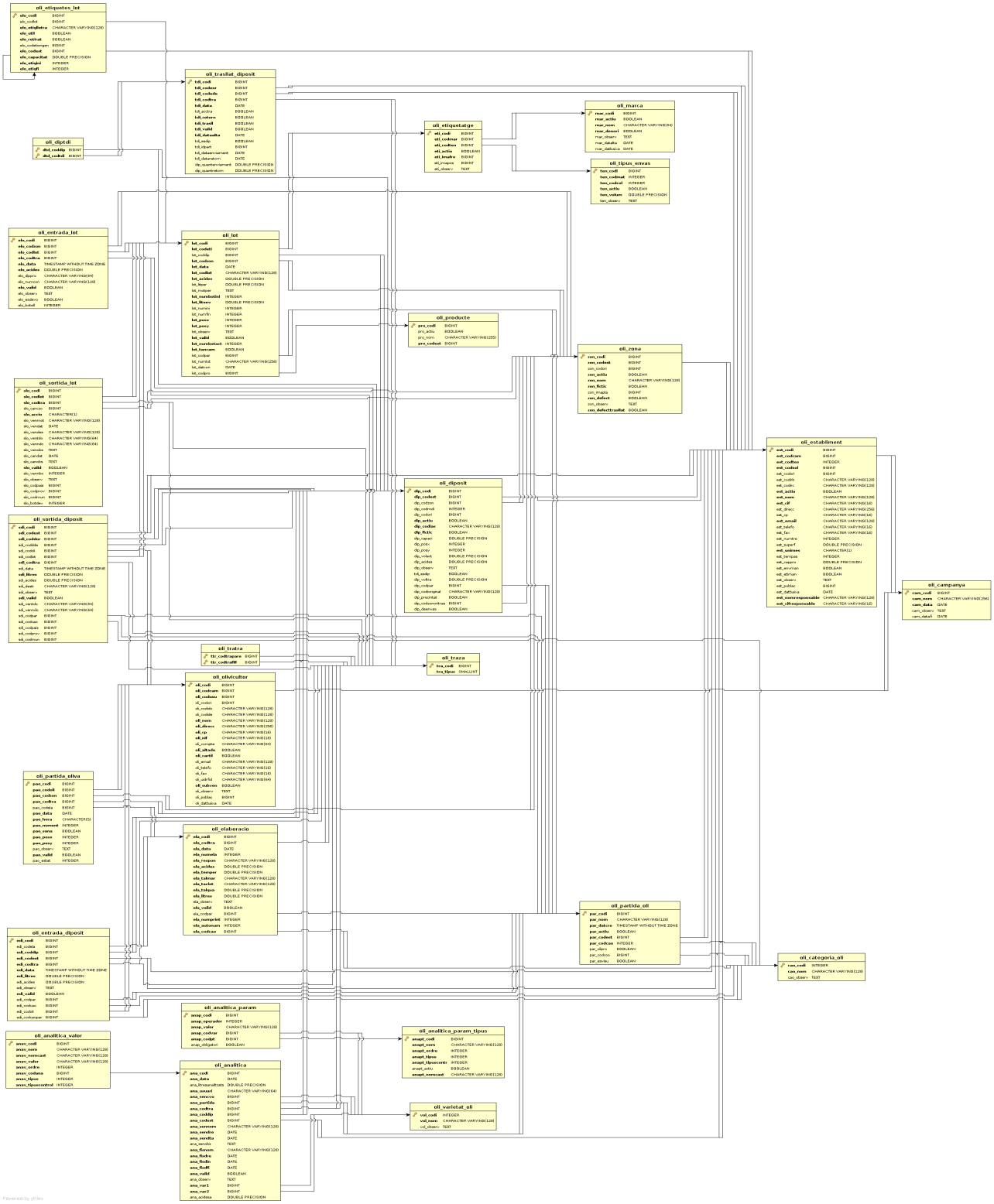
4.1. Entrada d'oliva:



5.

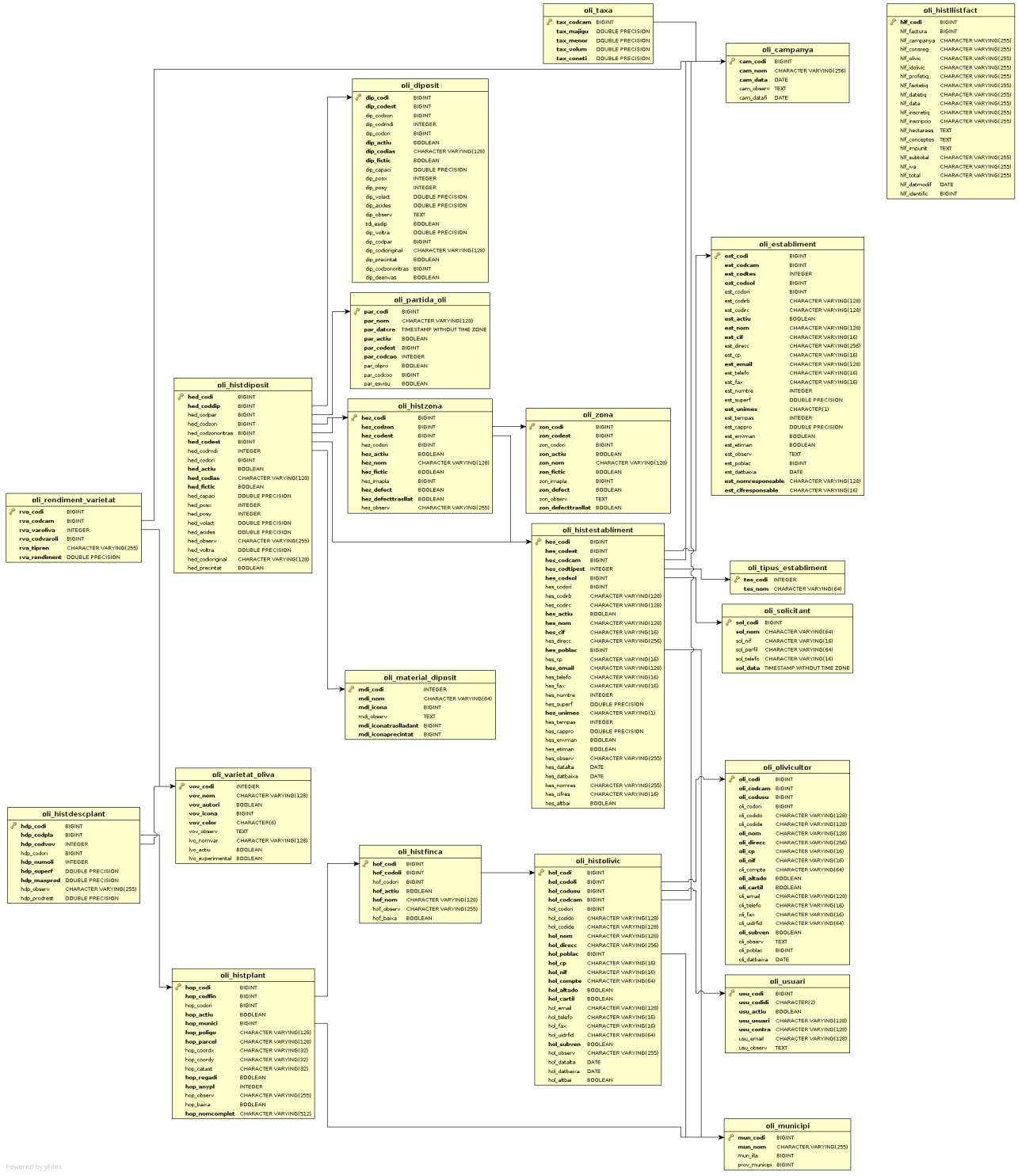
6.

6.1. Elaboració

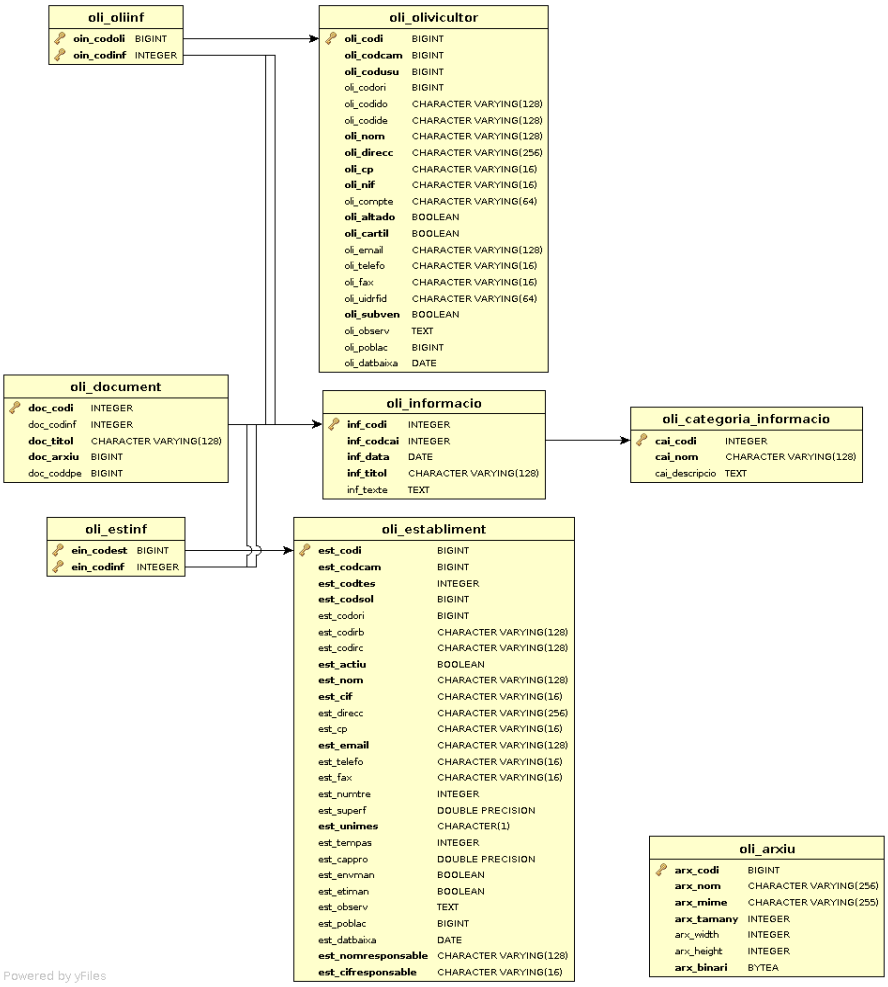


7.

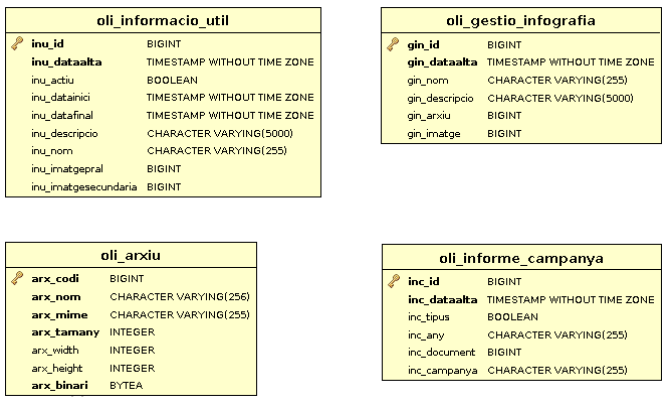
7.1. Canvi campanya



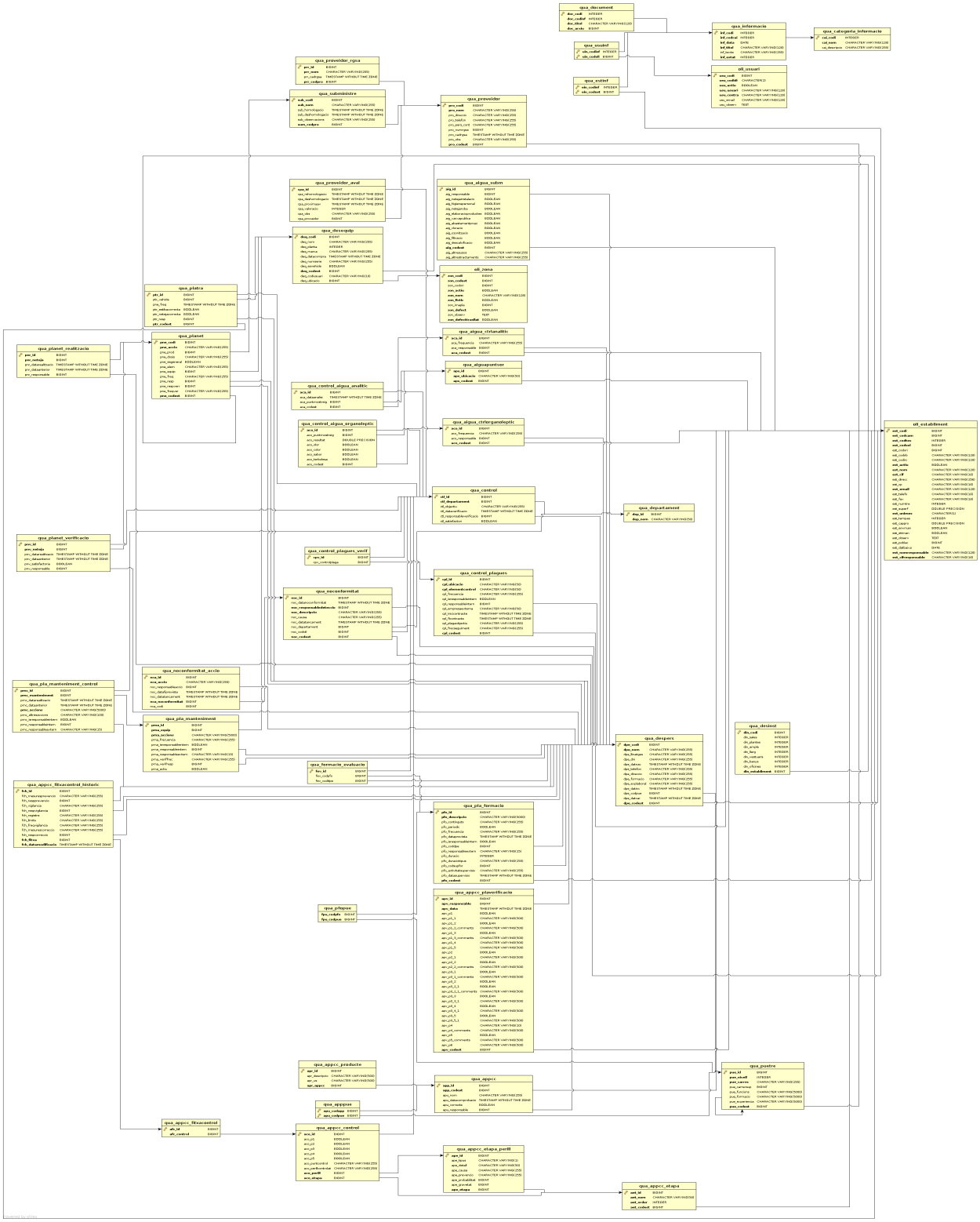
7.2. Informació



7.3. Front Office

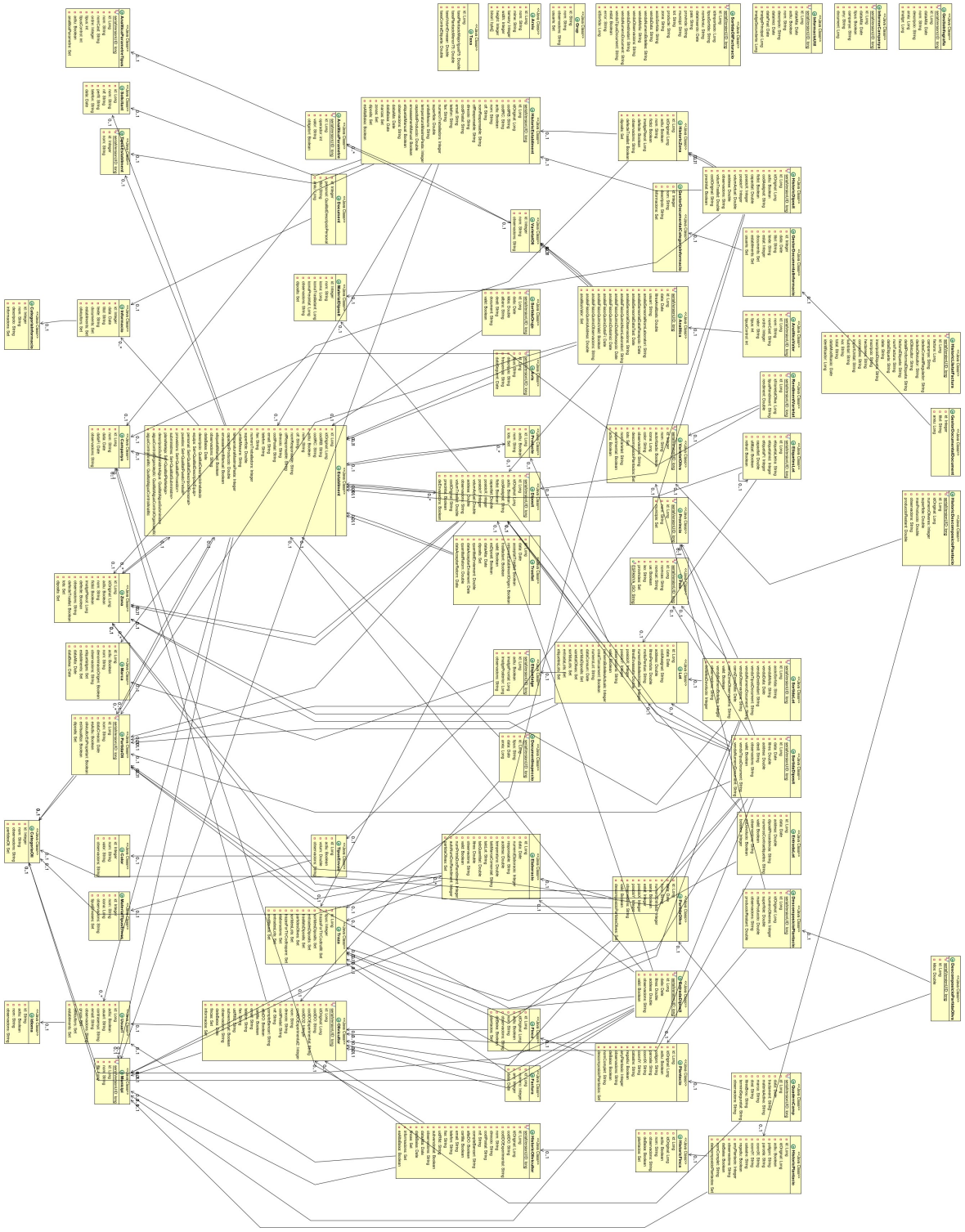


7.4. Qualitat

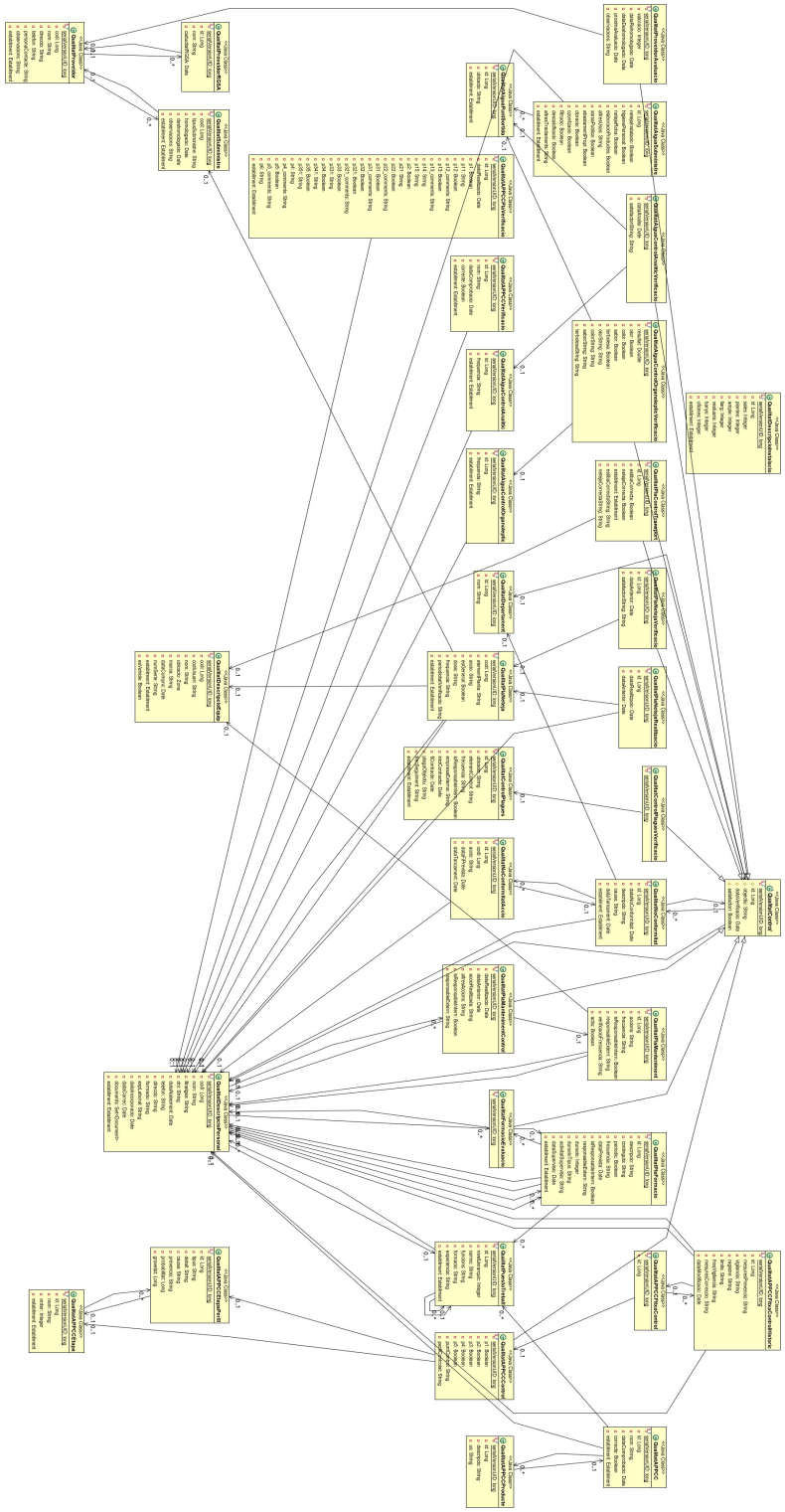


8. Model de dades

A continuació mostram els diagrames amb el model de dades:



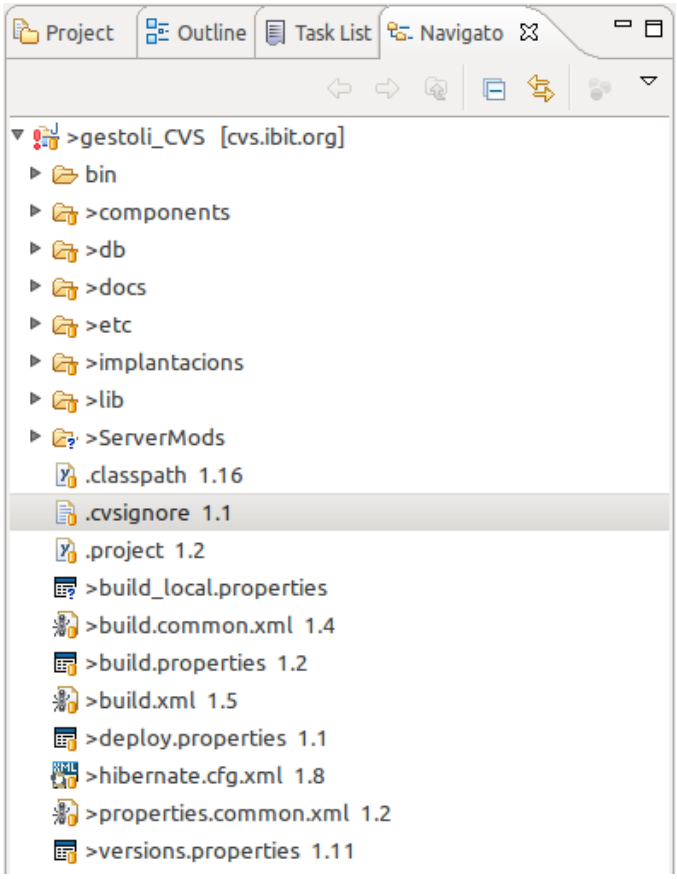
Hem separat la part de qualitat:



Com l'apartat anterior, aquest també el podem trobar en format PDF. El trobaren en *./docs/GEST-OLI - Document Tècnic - Model de Dades.pdf*.

9. DESENVOLUPAMENT

Per el desenvolupament de la aplicació s'ha utilitzat **Eclipse** com a entorn de desenvolupament integrat (IDE). Tot el codi font es troba en un repositori de IBIT, concretament en el servidor **cvs.ibit.org** i mòdul **gestoli**. Dins el codi font, es troben els arxius de configuració de l'Eclipse (.project y .classpath) amb els quals es muntaria automàticament el projecte de la següent manera:



Il·lustració : Projecte a Eclipse

Les classes Java es compilen amb JDK1.5 i la codificació dels arxius és UTF-8. S'ha utilitzat el framework de Java Spring[2] per a la construcció i validació de formularis.

A continuació s'explicarà cada un dels distints arxius, carpetes, packages, etc. que formen el projecte.

2 Spring: http://www.springsource.org/

9.1. root

- *.classpath*: arxiu de configuració del classpath en Eclipse
- *.cvsignore*: arxiu ignore de CVS per a aquesta carpeta arrel, que actualment només té tres entrades:

  output

  bin

  .settings

  que son els arxius i carpetes que es generen dinàmicament amb Eclipse i amb l'ant.
- *.project*: arxiu de configuració del projecte en Eclipse
- *build.local.properties*: arxiu de propietats per al build.xml
- *build.common.xml*: arxiu de propietats per al build.xml
- *build.properties*: arxiu de propietats per al build.xml
- *build.xml*: arxiu principal de l'execució amb distintes operacions a executar. La execució es pot realitzar des de una finestra del terminal, o configurar-se en l'eclipse com a 'external tool configuration', indicant les operacions a executar. Entre les operacions que es poden executar, destaquen:

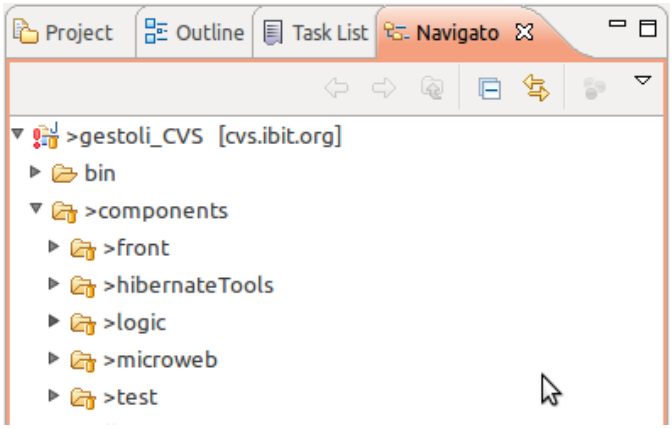
  ant clean → esborrar tots els arxius .class generats (esborrat de la carpeta "output")

  ant make → compila tots els arxius Java, crea totes les interfícies i implementacions dels EJBs i finalment genera l'arxiu gestoli.ear que contindrà tota la aplicació

  ant deploy.remote → copia l'arxiu gestoli.ear dins de la carpeta "deploy" del JBoss
- *deploy.properties*: arxiu complementari al build.xml
- *hibernate.cfg.xml*: arxiu de configuració del mapeig de Hibernate (Només per al plugin de hibernate).
- *properties.common.xml*: arxiu complementari al build.xml
- *versions.properties*: arxiu complementari al build.xml. Aquest arxiu serveix per a anotar les distintes versiones de arxius jar.

9.2. components

En aquest directori resideixen els 4 components distints que formen l'aplicación.

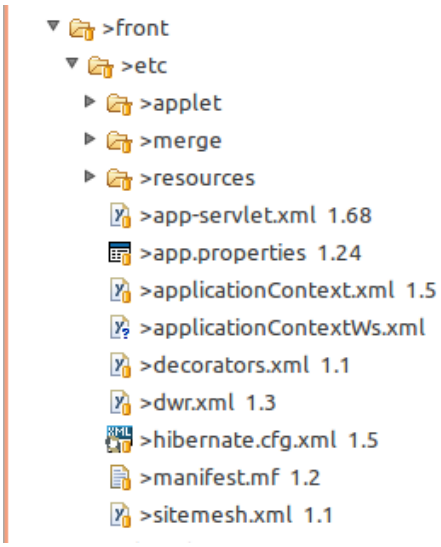


Il·lustració : Directori components

A continuació es detallen un a un per separat.

front

Conté tots els arxius necessaris per a la capa de negoci i presentació de l'aplicació. En l'arrel del directori, es troben els arxius *build.properties*, *build.xml* y *deploy.properties* necesaris per a la compilació del component *front*. A continuació es detallen els distints subdirectoris.



Il·lustració : Directori front

front/etc

Aquí es troben la majoria d'arxius de configuració, propietats, idioma, etc.

En la carpeta *applet* hi ha els arxius "drivers" par a la configuració del lector RFID[3].

En la carpeta *merge* es troben els arxius a partir dels quals es genera l'arxiu *web.xml* que defineix l'aplicació gestoli.war.

En la carpeta *resources* es troben els arxius de literals multi-idioma.

A continuació es detallen un poco més exhaustivament els arxius que es troben en l'arrel de *front/etc*:

- app.properties → propietats amb constants que s'injecten en les classes configurades per Spring
- applicationContext.xml → arxiu de configuració de Spring
- applicationContextWs.xml → arxiu complementari al applicationContext, amb les configuracions per al web service de Sistra.
- app-servlet.xml → arxiu de configuració de Spring. En aquest arxiu es configuren els *interceptors* i els *controladors* que son cridats en cada una de las distintes *URLs*
- decorators.xml → arxiu de configuració de sitemesh (indica la plantilla a utilitzar)
- dwr.xml → arxiu de configuració de *DWR*[4] on es configuren els distints serveis *AJAX*[5] que son cridats des de les pàgines *JSP*.
- hibernate.cfg.xml → arxiu de configuració del mapeig de Hibernate
- sitemesh.xml → arxiu de configuració de sitemesh

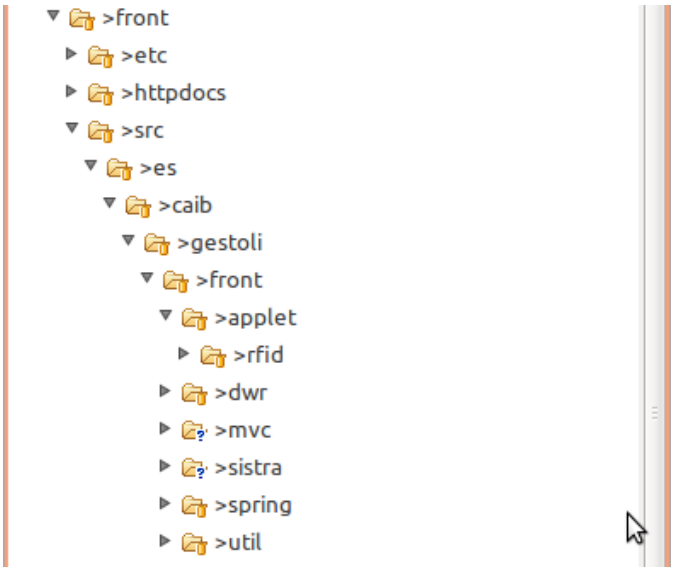
front/httpdocs

Aquí es troben tots els arxius pertanyents a la capa de presentació, tals com fulles d'estil, javascripts, imatges, JSPs, etc.

S'ha de tenir una especial atenció en la carpeta *httpdocs/applet*, ja que aquí es troba l'applet de interacció amb el lector RFID. Quan es fa l'*ant make* es depositen aquí les classes compilades de l'applet.

3 Lector RFID de Kimaldi: http://www.kimaldi.com/productos/sistemas_rfid
4 Direct Web Remoting: http://directwebremoting.org/dwr/index.html
5 Asynchronous JavaScript and XHTML: http://en.wikipedia.org/wiki/Ajax_(programming)

front/src



Il·lustració : Directori front/src

Aquí es troben totes les classes Java de la capa de negoci.

A continuació es detallen un poc més exhaustivament els diferents paquets que es troben en aquest directori:

- applet/rfid → classes de l'applet de interacció amb el lector RFID
- dwr → classes que defineixen els serveis AJAX
- spring → classes *Command*, *Controller*, *Delegate* i *Validator* de cada una de les diferents peticions al servidor. Aquestes classes implementen el framework Spring.
- spring/frontOffice → classes *Command*, *Controller*, *Delegate* i *Validator* de cada una de les diferents peticions al servidor des del frontOffice de l'aplicació.
- spring/interceptor → clase interceptores que s'executen sempre en primer lloc, en totes i cada una de les peticions al servidor
- spring/qualitat → classes *Command*, *Controller*, *Delegate* i *Validator* de cada una de les diferents peticions al servidor des del mòdul de qualitat.
- spring/views → classes que generen informes en formato PDF
- util → classes de utilitats varies

hibernateTools

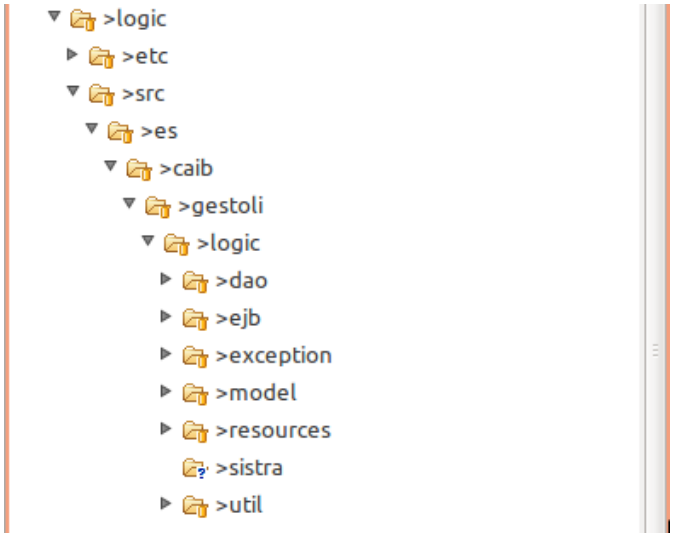
Aquest component s'utilitza únicament per a la creació de les classes de model a partir de las taules de base de dades. Es necessita tenir instal·lat el plugin HibernateTools[6] i amb els arxius *hibernate.reveng.xml, src/hibernate.cfg.xml* i *src/OliDoLibCustomStrategy.java* es generen automàticament totes les classes de model i el seu arxiu de mapeig corresponent *.hbm.xml*.

Logic

Il·lustració : Directori logic

Conté tots els arxius necessaris per a la capa de persistència de l'aplicación. En l'arrel del directori es troben els arxius *build.xml* i *deploy.properties* necessaris per a la compilació del component *logic*. A continuació es detallen els diferents subdirectoris.

logic/src



Il·lustració : Directori logic/src

6 Hibernate Tools for Eclipse and Ant: https://www.hibernate.org/255.html

Aquí es troben totes les classes Java de la capa de persistència.

A continuació es detallen un poc més exhaustivament els diferents paquets que es troben en aquest directori:

- dao → classes que a través de Hibernate realitzen totes les accions sobre la base de dades, tals com consultes, altes, modificacions i esborrats.
- ejb → classes EJB de sessió Stateless. Aqeustes fan de pont entre les classes de negoci i les de persistència, per, així, aïllar al màxim les diferents capes.
- exception → classe específica per a excepcions
- model → classes que modelen cada una de les distintes entitats de base de dades (i les relacions entre elles). Per a cada classe, hi ha també el seu arxiu de mapeig corresponent .hbm.xml de Hibernate.
- resources → aquí es troben les diferents icones de tipus d'oliva, dipòsits i envasos. Aquí també es troben les plantilles JRXML[7] dels diferents informes/documents que generen un arxiu PDF.
- util → classes de utilitats vàries

test

Aquest component conté les diferents classes java utilitzades per a una bateria de test específica.

Aquest apartat ja es comentarà més endavant en la secció de CONTROL DE QUALITAT.

7 Plantilla XML de JasperReports: http://jasperreports.sourceforge.net/

9.3. db

En aquest directori hi ha varis arxius relacionats amb la base de dades i la connexió de l'aplicació con aquesta.

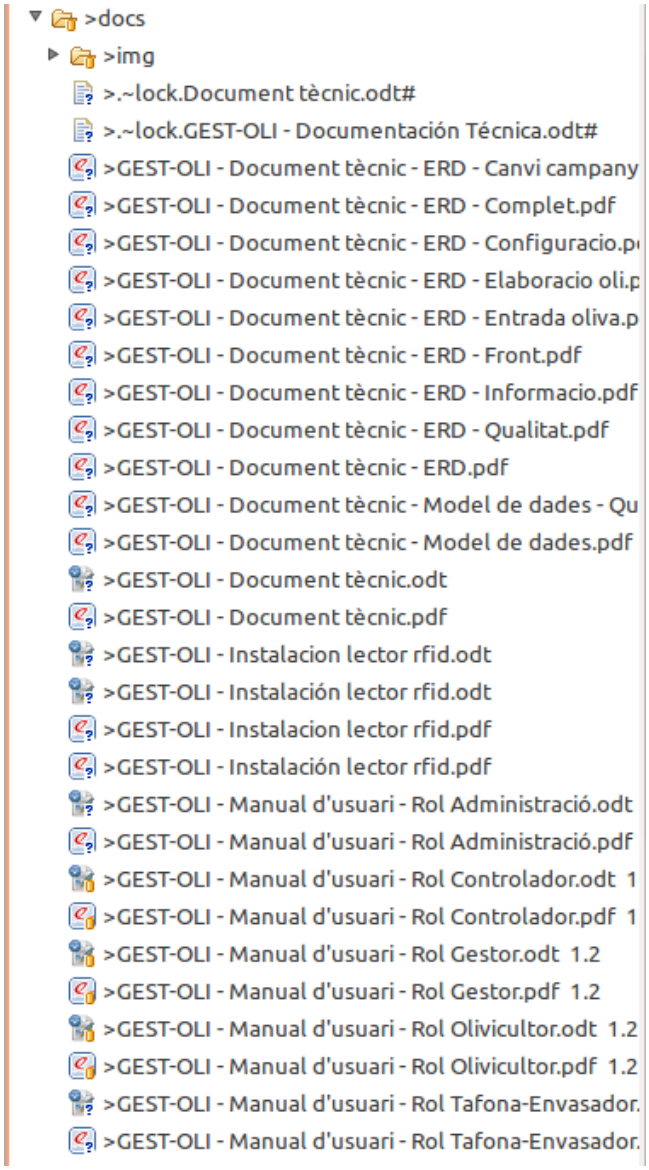
Il·lustració : Directori db

A continuació es detalla cada un dels arxius per separat:

- *gestoli_db.20100716.dump*: còpia de seguretat de la base de dades inicial
- *gestoli-ds.xml*: arxiu de configuració del *datasource*. Aquest arxiu s'ha de copiar en la carpeta *deploy* del Jboss
- gestoliNet.dmp.bz2: còpia de seguretat comprimida de la base de dades inicial de l'ampliació de getoli.
- *login-config.xml*: l'arxiu *login-config.xml* de la carpeta *conf* del JBoss, ha de tenir aquesta *application-policy*
- *mail-service.xml*: arxiu de configuració del servei de enviament de correus. Aquest arxiu conté l'mbean amb nom name="jboss:service=MailGestoli", el qual ha d'estar configurat en l'arxiu *mail-service.xml* de la carpeta *deploy* del Jboss
- qualitat.sql: script SQL amb la generació de l'apartat de qualitat de la base de dades inicial.
- scriptBD.sql: script SQL amb la generació de la base de dades inicial
- updateBD.sql: script SQL amb les modificacions sobre la base de dades inicial

9.4. docs

En aquest directori resideixen tots los documents i manuals de l'aplicació.



Il·lustració : directori docs

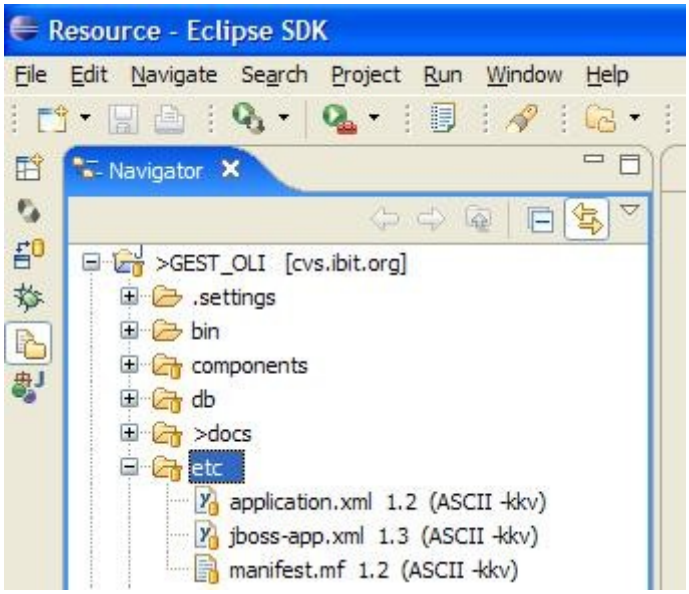
A continuació es detalla el contingut d'aquest directori:

- *img*: arxius amb les captures de pantalla dels diferents documents/manuals
- *javadoc*: javadoc en format HTML de totes les classes JAVA de l'aplicació
- *GEST-OLI - Document Tècnic.odt*: aquest document
- *GEST-OLI - Document Tècnica – ERD.pdf*: ERD de l'aplicación

- *GEST-OLI - Document Tècnic - Model de Dades.pdf*: model de dades de l'aplicació
- *GEST-OLI - Instalacion lector rfid.odt*: manual d'instal·lació del lector RFID
- *GEST-OLI - Manual d'usuari - Rol Administració.odt*: manual d'usuario amb rol de Administració
- GEST-OLI - Manual d'usuari - Rol Controlador.odt: manual d'usuario amb rol de Controlador
- GEST-OLI - Manual d'usuari - Rol Tafona - Envasador.odt: manual d'usuario amb rol de Tafona i/o Envasador
- GEST-OLI - Manual d'usuari - Rol Gestor.odt: manual d'usuario amb rol de Gestor
- GEST-OLI - Manual d'usuari - Rol Olivicultor.odt: manual d'usuario amb rol de Olivicultor

9.5. etc

En aquest directori es troben varis arxius de configuració de l'EAR.



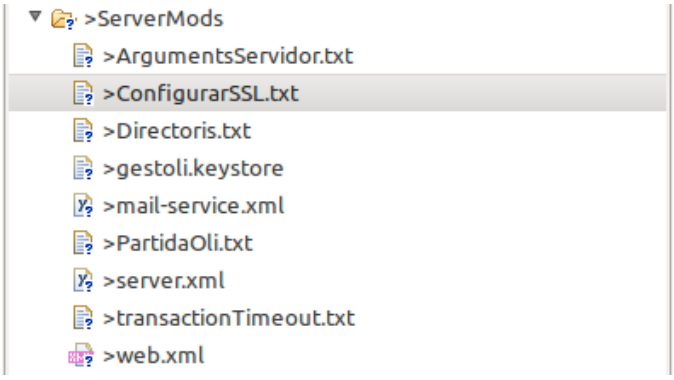
Il·lustració 5: directorio etc

9.6. lib

En aquest directori es troben totes les llibreries externes que utilitza l'aplicació.

9.7. serverMods

En aquest directori es troben els fitxers que expliquen les modificacions necessàries en la configuració bàsica del servisor jboss.



Il·lustració : directori ServerMods

A continuació es detalla el contingut d'aquest directori:

- *ArgumentsServidor.txt*: paràmetres a afegir al fitxer de configuració del jboss run.conf, a les opcions de la màquina virtual Java (JAVA_OPS)
- *ConfigurarSSL.txt*: configuracions necessàries del servidor per a configurar l'accés per SSL.
- *Directoris.txt*: *modificacions per a que millorar la seguretat, i no permetre llistar directoris del servidor.*
- *gestoli.keystore: magatzem de claus de gestoli (necessari per l'SSL)*
- *mail-service.xml: configuració de correu. Aquest fitxer s'ha de desplegar en el jboss, en el directori de deploy.*
- *PartidaOli.txt: script per a generar la seqüència de les partides d'oli*
- server.xml: fitxer server.xml de l'aplicació (no fa falta copiar-lo al servidor).
- *transactionTimeout.txt: modificació a efectuar al fitxer conf/jboss-service.xml* per a assegurar que no es tanqui la transacció del canvi de campanya abans que aquesta hagi acabat
- *web.xml: fitxer web.xml de l'aplicació (no fa falta copiar-lo al servidor).*

10. CONTROL DE QUALITAT

Par a testejar tota l'aplicació s'ha procedit a realitzar proves mitjançant diferents procediments, alguns automàtics i altres més tradicionals. A continuació es detallen els diferents mètodes.

10.1. Tests unitaris amb Selenium

Selenium IDE[8] és un plugin de Firefox que pertany al joc d'eines SeleniumHQ[9], i que permet realitzar jocs de proves sobre aplicacions web. Per a aconseguir-ho realitza la gravació de la acció seleccionada (navegació per una pàgina) en un "script", el qual es pot editar i parametritzar per a adaptar-se als diferents casos, i el que és més important, la seva execució es pot repetir tantes vegades com es vulgui (manual[10]).

Programes necesaris per a la creació/execució dels tests unitaris:

- Selenium IDE Extension per a Firefox: permet gravar i executar scripts d'accions sobre el navegador (registra tot el que estam fent). En particular interessa la opció de exportar l'script a codi jUnit[11].
- Selenium RC: té el servidor Selenium i el driver Selenium per a java. Es descomprimeix en qualsevol carpeta.
- Eclipse: els scripts creats amb el plugin per a Firefox poden executar-se directament amb jUnit des d'Eclipse.

El joc de tests creats per a aquesta ocasió es troben en *components/test*. A mode d'exemple es mostra la classe UsuariTest.java que ha creat Selenium IDE:

```
package es.caib.gestoli.test;

import junit.framework.TestCase;
import com.thoughtworks.selenium.DefaultSelenium;
import com.thoughtworks.selenium.Selenium;
import com.thoughtworks.selenium.SeleniumException;



public class UsuariTest extends TestCase {


        private Selenium selenium;


        public void setUp() throws Exception {
                selenium = new DefaultSelenium("https://localhost",8443,"*chrome");
                selenium.start();
        }

        public void testUsuariAlta() throws InterruptedException {
                selenium.open("/gestoli/Inici.html");
```

8 http://seleniumhq.org/download/
9 http://seleniumhq.org/
10 http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=seleniumIDE
11 http://www.junit.org/

```
		selenium.click("link=Configuració");
		selenium.click("link=Usuari");
		selenium.waitForPageToLoad("30000");
		selenium.click("enlaceCrearForm");
		selenium.waitForPageToLoad("30000");
		selenium.type("usuari", "GestorTest");
		selenium.type("contrasenya", "gestor");
		selenium.select("idiomaId", "label=Català");
		selenium.addSelection("id_grupsArray", "label=Consell Regulador - Gestor");
		selenium.click("moveRight");
		selenium.type("id_observacions", "Observacions gestor");
		selenium.click("enlaceGuardarForm");
		selenium.waitForPageToLoad("30000");
		assertTrue(selenium.isTextPresent("L'usuari s'ha creat amb èxit"));

	}


	public void testUsuariAltaYaExiste() throws InterruptedException {
		selenium.open("/gestoli/Inici.html");
		selenium.click("link=Configuració");
		selenium.click("link=Usuari");
		selenium.waitForPageToLoad("30000");
		selenium.click("enlaceCrearForm");
		selenium.waitForPageToLoad("30000");
		selenium.type("usuari", "GestorTest");
		selenium.type("contrasenya", "gestor");
		selenium.select("idiomaId", "label=Català");
		selenium.addSelection("id_grupsArray", "label=Consell Regulador - Gestor");
		selenium.click("moveRight");
		selenium.type("id_observacions", "Observacions gestor");
		selenium.click("enlaceGuardarForm");
		selenium.waitForPageToLoad("30000");
		assertTrue(selenium.isTextPresent("El nom d'usuari ja existeix"));

	}


	public void testUsuariBaja() throws InterruptedException {
		selenium.open("/gestoli/Inici.html");
		selenium.click("link=Configuració");
		selenium.click("link=Usuari");
		selenium.waitForPageToLoad("30000");

		// Busqueda
		String text=null;
		while (true){

			try{
				text=selenium.getText("link=GestorTest");
				break;
			}catch(SeleniumException e){

				try{
					if
(selenium.getAttribute("xpath=//div[@id='divSombra_0']/div[8]/span/a[last()]/@href").equals("#")){
						break;
					}
				}catch(SeleniumException e2){
					break;
				}
				selenium.click("//div[@id='divSombra_0']/div[8]/span/a[last()]");
				selenium.waitForPageToLoad("30000");
			}

		}
```

```
                assertTrue(text!=null);

                // Borrado
                selenium.click("link=GestorTest");
                selenium.waitForPageToLoad("30000");
                selenium.click("enlaceBorrarForm");
                assertTrue(selenium.getConfirmation().matches("^Esborrar el registre[\\s\\S] Aquesta acció no es pot
desfer\\.$"));
                selenium.waitForPageToLoad("30000");

                selenium.click("link=Configuració");
                selenium.click("link=Usuari");
                selenium.waitForPageToLoad("30000");

                // Busqueda
                text=null;
                while (true){

                        try{
                                text=selenium.getText("link=GestorTest");
                                break;
                        }catch(SeleniumException e){
                                try{
                                        if
(selenium.getAttribute("xpath=//div[@id='divSombra_0']/div[8]/span/a[last()]/@href").equals("#")){
                                                break;
                                        }
                                }catch(SeleniumException e2){
                                        break;
                                }
                                selenium.click("//div[@id='divSombra_0']/div[8]/span/a[last()]");
                                selenium.waitForPageToLoad("30000");
                        }

                }

                assertTrue(text==null);

        }


        public void tearDown() {
                selenium.stop();
        }


}
```

Els tests es poden associar a un macrocontenidor TestSuite de jUnit que té aquesta estructura:

```
package es.caib.gestoli.test;
import junit.framework.Test;
import junit.framework.TestSuite;
import junit.textui.TestRunner;


        public class VariosTestSuite extends TestSuite {

        public VariosTestSuite(String name) {
```

```
        super(name);
    }

    public static void main(String[] args) {
      TestRunner.run(suite());
    }

    public static Test suite() {
            TestSuite suite = new VariosTestSuite("Pruebas varias Gestoli");
            suite.addTestSuite(UsuariTest.class);
            suite.addTestSuite(TipusEnvasTest.class);
            suite.addTestSuite(EstablimentTest.class);
            suite.addTestSuite(OlivicultorTest.class);
            return suite;
    }

}
```

Els procediments setUp() i tearDown() s'executen abans i després de cada un dels altres mètodes. El codi dels altres procediments ve en gran part de la exportació a jUnit de Selenium IDE després d'haver registrat un script.

Per a l'execució dels tests:

- Amb Selenium Server: s'executa con *java -jar selenium-server.jar*
- Amb Eclipse: hi ha vàries formes d'executar els tests des d'Eclipse:
    - Fent clic dret en el nom del mètode dins de la Case i “run as... jUnit test” ens permet executar només la prova d'aquest mètode
    - Fent clic dret en el nom de l'arxiu TestCase i “run as... jUnit test” ens permet executar totes les proves que es troben en aquesta Case seqüencialment.
    - Fent clic dret en el nom de l'arxiu TestSuite i “run as... jUnit test” ens permet executar les proves de tota la Suite.

Selenium IDE a vegades no posa una espera després d'un assert de diàleg de confirmació que redirigeix a una altra pàgina. Si no es posa una espera hi pot haver resultats incorrectes, s'ha de posar manualment.

10.2. Tests “tradicionals”

Durant tota la fase de desenvolupament de cada una de les funcionalitats de l'aplicació, un membre de l'equip de desenvolupament (distint al que ha implementat la funcionalitat) ha desenvolupat els tests unitaris i funcionals (seguint les directrius del document de disseny proporcionat per l'IBIT) directament sobre el nostre entorn de desenvolupament.

Després de cada una de les diferents fases del projecte, s'ha realitzat un exhaustiu test realitzant tests de totes les funcionalitats.

També, després de cada nova implantació en l'entorn de producció, s'han realitzat tests sobre aquest entorno.

11. IMPLANTACIÓ

La aplicació desenvolupada en **JAVA JDK1.5** s'executa sobre un servidor d'aplicacions **JBoss 3.2.8** en un servidor **Linux**. La base de dades es **PostgreSQL 8.1**.

Per a la instalació del lector RFID s'ha de seguir el documento "**GEST-OLI - Instalación lector rfid**", també proporcionat juntament amb aquest document.

Dins del JBoss, en l'arxiu **conf/log4j.xml** s'ha d'incloure:

```
<logger name="es.caib.gestoli">
  <level value="INFO"/>
</logger>
```

per a fixar el nivell de log de les pròpies classes de l'aplicació.

També hem de configurar altres aspectes del JBoss tal com es comenta en els arxius del directori ServerMods (veure apartat 9.7):

- Modificar el fitxer run.conf:

```
JAVA_OPTS="-server -Xms256m -Xmx2048m -XX:MaxPermSize=384m -Djboss.bind.address=127.0.0.1
-Djavax.xml.stream.XMLInputFactory=com.bea.xml.stream.MXParserFactory
-Djavax.xml.stream.XMLEventFactory=com.bea.xml.stream.EventFactory
-Djavax.xml.stream.XMLOutputFactory=com.bea.xml.stream.XMLOutputFactoryBase"
```

- Configurar SSL:
  - *Copiar gestoli.keystore al directori <jboss_dir>\server\default\conf*
  - *afegir a <jboss_dir>\server\default\deploy\jbossweb-tomcat50.sar\server.xml:*

```
<!-- SSL/TLS Connector configuration using the admin devl guide keystore-->
<Connector port="8443" address="${jboss.bind.address}"
maxThreads="100" minSpareThreads="5" maxSpareThreads="15"
scheme="https" secure="true" clientAuth="false"
keystoreFile="${jboss.server.home.dir}/conf/gestoli.keystore"
keystorePass="gestoli" sslProtocol = "TLS" />
```

- Per a evitar que els directoris sense index s'indexin s'ha de modificar el fitxer <jboss_dir>\server\default\deploy\jbossweb-tomcat50.sar\conf\web.xml, on hem de modificar el paràmetre listings, i posar-lo a false:

```
<servlet>
   <servlet-name>default</servlet-name>
   <servlet-class>
       org.apache.catalina.servlets.DefaultServlet
   </servlet-class>
   <init-param>
      <param-name>debug</param-name>
      <param-value>0</param-value>
   </init-param>
   <init-param>
      <param-name>listings</param-name>
      <param-value>false</param-value>
   </init-param>
   <load-on-startup>1</load-on-startup>
</servlet>
```

- Modificar el TransactionTimeout al fitxer server/default/conf/jboss-service.xml:

```
<!-- The fast in-memory transaction manager. -->
  <mbean code="org.jboss.tm.TransactionManagerService"
    name="jboss:service=TransactionManager"
    xmbean-dd="resource:xmdesc/TransactionManagerService-xmbean.xml">
    <attribute name="TransactionTimeout">1200</attribute>
    <depends optional-attribute-name="XidFactory">jboss:service=XidFactory</depends>
  </mbean>
```

Per a crear la base de dades, accedirem a psql com a postgres, i executarem:

```
CREATE USER getoli WITH PASSWORD 'gestoli';
CREATE DATABASE gestoli OWNER gestoli;
GRANT ALL PRIVILEGES ON DATABASE gestoli TO gestoli;
```

on substituirem l'usuari i el password per el que es connectarà a l'aplicació (es troba en l'arxiu *gestoli-ds.xml*).

Un cop creada la base de dades, descomprimirem el fitxer gestoliNet.dmp.bz2, i restaurarem aquesta còpia inicial a la base de dades que acabam de crear:

```
pg_restore -U postgres -d gestoli gestoliNet.dmp
```

Posteriorment, utilitzant l'usuari de BBDD amb el que es connectarà l'aplicació executar l'script de l'arxiu **updateBD.sql**.

El format de les dates en la BBDD ha de ser DMY, per tant s'ha d'executar la següent comanda:

```
ALTER DATABASE gestoli SET DateStyle = 'ISO, DMY';
```

y posteriormente comprobar que esté correcto:

```
gestoli_db=#  SHOW datestyle;
 DateStyle
-----------
 ISO, DMY
(1 fila)
```

El locale del servidor ha de ser “es_ES”:

```
jboss@dgtic::~$ locale
LANG=es_ES.UTF-8
LC_CTYPE="es_ES.UTF-8"
LC_NUMERIC="es_ES.UTF-8"
LC_TIME="es_ES.UTF-8"
LC_COLLATE="es_ES.UTF-8"
LC_MONETARY="es_ES.UTF-8"
LC_MESSAGES="es_ES.UTF-8"
LC_PAPER="es_ES.UTF-8"
LC_NAME="es_ES.UTF-8"
LC_ADDRESS="es_ES.UTF-8"
LC_TELEPHONE="es_ES.UTF-8"
LC_MEASUREMENT="es_ES.UTF-8"
LC_IDENTIFICATION="es_ES.UTF-8"
LC_ALL=
```

L'EAR resultant de l'aplicació (gestoli.ear) es trobarà en la carpeta *output/product*, del projecte Eclipse, una vegada s'hagi executat la comanda *ant make*. Aquest EAR s'haurà de moure a la carpeta *deploy* del JBoss. En aquesta mateixa carpeta s'ha de copiar l'arxiu gestoli-ds.xml i el mail-service.xml (ja explicats en un apartat previ d'aquest document).

Arrancam el JBoss i ja podrem començar a utilitzar l'aplicació.

Podrem accedir amb l'usuari gestor / gestor.