Sistemes ERP-CRM. Explotació i adequació - II

Isidre Guixà Miranda

Sistemes de gestió empresarial

Índex

Introducció

L'explotació i adequació d'un sistema ERP-CRM implica conèixer les eines de desenvolupament que el propi ERP-CRM proporciona. En el cas d'OpenERP ens cal conèixer els mecanismes que proporciona el *framework* OpenObject en el qual es basa el desenvolupament d'OpenERP.

Una vegada es coneix l'arquitectura MVC del *framework* OpenObject i se sap interpretar el model, la vista i el controlador dels mòduls facilitats per OpenERP i s'és capaç de dissenyar-ne de nous, cal aprofundir en altres mecanismes importants que facilita OpenObject: herència –utilitzada per l'adaptació de mòduls existents–, seguretat, càrrega massiva de dades, desenvolupament d'assistents, generació de traduccions idiomàtiques, disseny d'informes, disseny de quadres de comandament i altres. Per aconseguir-ho hem estructurat la unitat en dos apartats.

L'apartat "OpenERP: desenvolupament avançat" té com a objectiu conèixer els mecanismes avançats de disseny que OpenObject proporciona. Així, veurem com aplicar l'herència en el disseny de model i vistes, com definir polítiques de seguretat, com preparar càrregues massives de dades, com desenvolupar assistents d'ajuda a l'usuari i com generar traduccions idiomàtiques.

L'apartat "Reporting & BI" està destinat al disseny d'informes i quadres de comandament i tindrà dos enfocaments: intern, generant informes i quadres de comandament que s'integrin en OpenERP, i extern, utilitzant una de les eines BI que avui en dia té més empenta i que permet generar informes i quadres de comandament connectant a qualsevol origen de dades.

El disseny d'informes que s'integrin en OpenERP es basa en utilitzar els mecanismes existents a OpenObject, ja siguin directament incorporats en el *framework* (informes dissenyats amb OpenOffice o LibreOffice) o facilitats per la comunitat (informes construïts amb el conegut dissenyador d'informes de codi lliure iReport). Pel que fa als quadres de comandament, OpenERP en possibilita la confecció a partir de vistes ja existents.

En referència a la utilització d'alguna de les eines BI per generar informes i quadres de comandament connectant a qualsevol origen de dades i, en particular, a la base de dades PostgreSQL d'una empresa gestionada per OpenERP, s'ha escollit la versió comunitària de JasperReportsServer. La causa d'aquesta elecció radica en el fet que l'eina de disseny dels informes és la mateixa eina iReport que utilitzarem per dissenyar una tipologia d'informes integrables en OpenERP i, d'aquesta manera, sumarem esforços. Per desgràcia, el disseny de quadres de comandament no existeix en la versió comunitària de JasperReports Server, però l'empresa JasperSoft ens permet instal·lar una versió de prova de la versió Enterprise, que sí permet el disseny de quadres de comandament, totalment operativa, per a 30 dies.

El seguiment d'aquesta unitat pressuposa que l'alumne és coneixedor de:

1. La implantació tècnica d'OpenERP. El seu coneixement s'adquireix a la unitat "Sistemes ERP-CRM. Implantació" del mòdul professional *Sistemes de gestió empresarial.*
2. La implementació del patró model-vista-controlador facilitat pel *framework* OpenObject, en el que es basa OpenERP. El seu coneixement s'adquireix a la unitat "Sistemes ERP-CRM. Explotació i adequació – I" del mòdul professional *Sistemes de gestió empresarial*.
3. El llenguatge XML. El seu coneixement s'adquireix al mòdul professional *Llenguatges de marques i sistemes de gestió de la informació*.
4. La programació orientada a objectes. El seu coneixement s'adquireix al mòdul professional *Programació*.
5. El llenguatge Python.

Per tal d'assolir un bon aprenentatge, cal estudiar els continguts en l'ordre indicat, sense saltar-se cap apartat. Quan es fa referència a algun annex del web, adreçar-s'hi i estudiar-lo. Una vegada estudiats els continguts del material paper i del material web, cal desenvolupar les activitats web.

Resultats d'aprenentatge

En finalitzar aquesta unitat l'alumne/a:

1. Realitza operacions de gestió i consulta de la informació seguint les especificacions de disseny i utilitzant les eines proporcionades pels sistemes ERP-CRM i solucions d'intel·ligència de negocis (BI).
    - Utilitza eines i llenguatges de consulta i manipulació de dades proporcionades pels sistemes ERP-CRM.
    - Genera formularis.
    - Genera informes, des del sistema ERP-CRM i des de solucions BI.
    - Genera quadres de comandament, des del sistema ERP-CRM i des de solucions BI.
    - Exporta informes.
    - Utilitza les funcionalitats d'accés centralitzat que proporcionen les solucions BI.
    - Documenta les operacions realitzades i les incidències observades.
2. Adapta sistemes ERP-CRM identificant els requeriments d'un supòsit empresarial i utilitzant les eines proporcionades pels mateixos.
    - Identifica les possibilitats d'adaptació de l'ERP-CRM.
    - Adapta definicions de camps, taules i vistes de la base de dades de l'ERP-CRM.
    - Adapta consultes.
    - Adapta interfícies d'entrada de dades i de processos.
    - Personalitza informes i quadres de comandament.
    - Adapta procediments emmagatzemats de servidor.
    - Realitza proves.
    - Documenta les operacions realitzades i les incidències observades.

1. OpenERP: desenvolupament avançat

L'OpenERP és desenvolupat a partir del *framework* OpenObject que es basa en el patró MVC. De manera que, per implementar un nou mòdul en OpenERP, cal definir-ne el model, la vista i el controlador. Però amb això no n'hi ha prou. El *framework* OpenObject és molt potent i facilita mecanismes per aconseguir altres fites, com ara:

- Adequar mòduls existents i així garantir que les actualitzacions d'aquests mòduls no destrossin les adequacions desenvolupades, fet que és el resultat de l'aplicació del mecanisme d'herència.
- Definir una política de seguretat que s'apliqui en el procés d'instal·lació del mòdul.
- Dissenyar assistents que facilitin l'execució de diversos processos.
- Incorporar dades a un mòdul, ja sigui per facilitar la demostració de les funcionalitats del mòdul per quan s'ha instal·lat l'empresa amb dades de demostració, o perquè el mòdul necessita un conjunt de dades bàsiques pel seu funcionament (per exemple, els diversos tipus d'IVA a aplicar en el mòdul comptable d'un país).
- Traduir un mòdul a diversos idiomes.

1.1 Herència

El *framework* OpenObject facilita el mecanisme de l'herència per tal que els programadors puguin adaptar mòduls existents i garantir a la vegada que les actualitzacions dels mòduls no destrossin les adequacions desenvolupades.

L'herència es pot aplicar en els tres components del patró MVC:

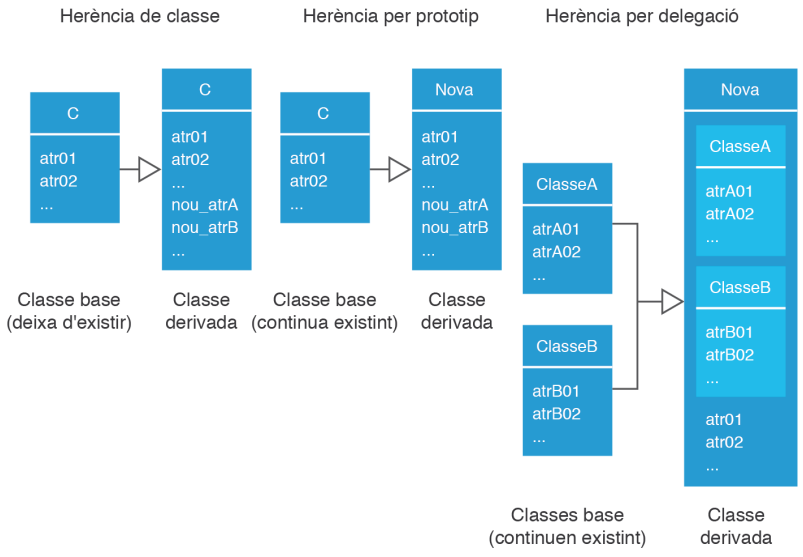
- En el model: possibilita ampliar les classes existents o dissenyar noves classes a partir de les existents.
- En la vista: possibilita modificar el comportament de vistes existents o dissenyar noves vistes.
- En el controlador: possibilita sobreescriure els mètodes existents o dissenyar-ne de nous.

OpenObject proporciona tres mecanismes d'herència: l'herència de classe, l'herència per prototip i l'herència per delegació. La taula 1.1 presenta els tres tipus i en presenta les principals diferències, i la figura 1.1 en mostra una representació.

Taula 1.1. Mecanismes d'herència facilitats pel framework OpenObject

Mecanisme	Característiques	Com es defineix
De classe	- Herència simple. - La classe original queda substituïda per la nova classe. - Afegeix noves funcionalitats (atributs i/o mètodes) a la classe original. - Les vistes definides sobre la classe original continuen funcionant. - Permet sobreescriure mètodes de la classe original. - En PostgreSQL, continua mapada en la mateixa taula que la classe original, ampliada amb els nous atributs que pugui incorporar.	- S'utilitza l'atribut `_inherit` en la definició de la nova classe Python: `_inherit = obj` - El nom de la nova classe ha de continuar sent el mateix que el de la classe original: `_name = obj`
Per prototip	- Herència simple. - Aprofita la definició de la classe original (com si fos un «prototipus»). - La classe original continua existint. - Afegeix noves funcionalitats (atributs i/o mètodes) a les aportades per la classe original. - Les vistes definides sobre la classe original no existeixen (cal dissenyar-les de nou). - Permet sobreescriure mètodes de la classe original. - En PostgreSQL, queda mapada en una nova taula.	- S'utilitza l'atribut `_inherit` en la definició de la nova classe Python: `_inherit = obj` - Cal indicar el nom de la nova classe: `_name = nou_nom`
Per delegació	- Herència simple o múltiple. - La nova classe «delega» certs funcionaments a altres classes que incorpora a l'interior. - Els recursos de la nova classe contenen un recurs de cada classe de la que deriven. - Les classes base continuen existint. - Afegeix les funcionalitats pròpies (atributs i/o mètodes) que correspongui. - Les vistes definides sobre les classes bases no existeixen a la nova classe. - En PostgreSQL, queda mapada en diferents taules: una taula per als atributs propis, mentre que els recursos de les classes derivades resideixen en les taules corresponents a les dites classes.	- S'utilitza l'atribut `_inherits` en la definició de la nova classe Python: `_inherits = ...` - Cal indicar el nom de la nova classe: `_name = nou_nom`

Figura 1.1. Mecanismes d'herència facilitats pel framework OpenObject



Per últim, comentar que el fitxer `__openerp__.py` d'un mòdul que contingui herència respecte altres mòduls n'haurà d'incloure els noms a l'apartat `depends`, per garantir que, en instal·lar el mòdul que hereta, els mòduls dels quals s'hereta ja estiguin instal·lats o s'instal·lin prèviament.

1.1.1 Herència en el model

El disseny d'un objecte (classe) d'OpenObject heretat és gairebé idèntic al disseny d'un objecte (classe) d'OpenObject no heretat; únicament hi ha dues diferències:

- Apareix l'atribut `_inherit` o `_inherits` per indicar l'objecte (herència simple) o els objectes (herència múltiple) dels quals deriva el nou objecte. La sintaxi a seguir és:

```
_inherit = 'nom.objecte.del.que.es.deriva'
_inherits = {'nom.objecte1':'nom_camp_FK1', ...}
```

- En cas d'herència simple, el nom (atribut `_name`) de l'objecte derivat pot coincidir o no amb el nom de l'objecte pare. També és possible no indicar l'atribut `_name`, fet que indica que el nou objecte manté el nom de l'objecte pare.

L'herència simple (`_inherit`) amb atribut `_name` idèntic al de l'objecte pare, s'anomena **herència de classe** i en ella el nou objecte substitueix l'objecte pare, tot i que les vistes sobre l'objecte pare continuen funcionant. Aquest tipus d'herència, la més habitual, s'utilitza quan es vol afegir dades (`_columns`) i/o modificar propietats de dades existents i/o modificar el funcionament d'alguns mètodes. En cas d'afegir dades, aquestes s'afegeixen a la taula de la base de dades en la qual estava mapat l'objecte pare.

Exemple d'herència de classe

L'herència de classe la trobem en molts mòduls que afegeixen dades i mètodes a objectes ja existents, com per exemple, el mòdul comptabilitat (`account`) que afegix dades i mètodes a l'objecte `res.partner`. Fixem-nos en el contingut del mòdul `account`:

```
class res_partner(osv.osv):
  _name = 'res.partner'
  _inherit = 'res.partner'
  _columns = {
    ...
    'debit_limit': fields.float('Payable limit'),
...
```

Podeu comprovar que la taula `res_partner` d'una empresa sense el mòdul `account` instal·lat no conté el camp `debit_limit`, que en canvi sí que hi apareix una vegada instal·lat el mòdul.

OpenERP té molts mòduls que deriven de l'objecte `res.partner` per afegir-hi característiques i funcionalitats.

L'herència simple (`_inherit`) amb atribut `_name` diferent al de l'objecte pare, s'anomena **herència per prototip** i en ella es crea un nou objecte que aglutina les dades (`_columns`) i mètodes que tenia l'objecte del qual deriva, juntament amb les noves dades i mètodes que pugui incorporar el nou objecte. En aquest cas, sempre es crea una nova taula a la base de dades per mapar el nou objecte.

Exemple d'herència per prototip

L'herència per prototip és difícil de trobar en els mòduls que incorpora OpenERP. Un exemple el tenim en el mòdul `base_calendar` en el qual podem observar el mòdul comptabilitat (`account`) que afegix dades i mètodes a l'objecte `res.partner`. Fixem-nos en el contingut del mòdul `account`:

```
class res_alarm(osv.osv):
  _name = 'res.alarm'
...
class calendar_alarm(osv.osv):
  _name = 'calendar.alarm'
  _inherit = 'res.alarm'
...
```

En una empresa que tingui el mòdul `base_calendar` instal·lat podeu comprovar l'existència de la taula `res_alarm` amb els camps definits a l'apartat `_atributs` de la classe `res_alarm` i la taula `calendar_alarm` amb camps idèntics als de la taula `res_alarm` més els camps definits a l'apartat `_atributs` de la classe `calendar_alarm`.

L'herència múltiple (`_inherits`) s'anomena **herència per delegació** i sempre provoca la creació d'una nova taula a la base de dades. L'objecte derivat ha d'incloure, per cada derivació, un camp `many2one` apuntant l'objecte del qual deriva, amb la propietat `ondelete='cascade'`. L'herència per delegació obliga que cada recurs de l'objecte derivat apunti a un recurs de cadascun dels objectes dels quals deriva i es pot donar el cas que hi hagi diversos recursos de l'objecte derivat que apuntin a un mateix recurs per algun dels objectes dels quals deriva.

Exemple d'herència per delegació

L'herència per delegació no és molt utilitzada en els mòduls que incorpora OpenERP. Un exemple el trobem en el mòdul `product` en el qual l'objecte `product.product` deriva per delegació de l'objecte `product.template`:

```
class product_template(osv.osv):
  _name = "product.template"
...
class product_product(osv.osv):
  _name = "product.product"
  _inherits = {'product.template': 'product_tmpl_id'}
  _columns = {
    ...
    'product_tmpl_id':
      fields.many2one ('product.template', 'Product Template',
                       required=True, ondelete="cascade"),
...
```

Tot i que `_inherits` està pensat per possibilitar l'herència múltiple, en aquest cas l'objecte `product.product` únicament deriva de l'objecte `product.template`. A nivell de la base de dades, la taula `product_template` conté els camps definits a l'atribut `_columns` de l'objecte `template.product` i la taula `product_product` conté els camps definits a l'atribut `_columns` de l'objecte `product.product`, entre els quals hi ha el camp `producte_tmpl_id`, de tipus `many2one`, utilitzat en la declaració de l'herència a `_inherits`. Cada recurs de `product.product` penja obligatòriament d'un recurs de `product.template` i per un recurs de `product.template` podem tenir diversos recursos de `product.product` que hi pengin.

Trobareu el mòdul `delegation_inheritance` dins l'apartat "Mòduls OpenERP i fitxers auxiliars" dels annexos del web.

Exemple d'herència múltiple

El mòdul `delegation_inheritance` defineix l'objecte `di.subclass` que deriva dels objectes `di.superclass1` i `di.superclass2` i facilita dues vistes: `tree` i `form`.

Feu una ullada a la definició de les classes i comproveu els següents funcionaments.

La vista `tree`, que és editable, permet afegir recursos a l'objecte `subclass`. Fixeu-vos que les dues primeres columnes corresponen a continguts dels objectes `superclass1` i `superclass2` i que per cada registre que afegim a través d'aquesta vista es genera un registre a cadascuna de les taules: `di_subclass`, `di_superclass1` i `di_superclass2`, de manera que el recurs de `di_subclass` referencia als altres recursos.

La vista `form`, en canvi, obliga a seleccionar recursos dels objectes `superclass1` i `superclass2`, i demostra que els recursos dels objectes pare poden ser referenciats per diversos objectes dels recursos derivats.

1.1.2 Herència en la vista

L'herència de classe possibilita continuar utilitzant les vistes definides sobre l'objecte pare, però en moltes ocasions interessa disposar d'una versió retocada. En aquest cas, és molt millor heretar de les vistes existents (per afegir, modificar o eliminar camps) que no pas reemplaçar-les completament.

El disseny d'una vista heretada és gairebé idèntic al disseny d'una vista no heretada; únicament cal afegir l'element:

```
1 <field name="inherit_id" ref="id_xml_vista_pare"/>
```

En cas que la vista `id_xml_vista_pare` resideixi en un mòdul diferent del que estem dissenyant, cal afegir el nom del mòdul al davant:

```
1 <field name="inherit_id" ref="modul.id_xml_vista_pare"/>
```

El motor d'herència d'OpenObject, en trobar una vista heretada, processa el contingut de l'element `arch`. Per cada fill d'aquest element que tingui algun atribut, OpenObject cerca a la vista pare una etiqueta amb atributs coincidents (excepte el de la posició) i, a continuació, combina els camps de la vista pare amb els de la vista heretada i estableix la posició de les noves etiquetes a partir dels següents valors:

- `inside` (per defecte): els valors s'afegeixen "dins" de l'etiqueta.
- `after`: afegeix el contingut després de l'etiqueta.
- `before`: afegeix el contingut abans de l'etiqueta.
- `replace`: reemplaça el contingut de l'etiqueta.

Per a reemplaçar el contingut d'un element `camp` s'utilitza la següent plantilla:

```
<field name="arch" type="xml">
  <field name="camp" position="replace">
    <field name="nou_camp" ... />
  </field>
</field>
```

Per esborrar un camp d'una vista s'utilitza un element buit amb l'atribut `position="replace"`:

```
<field name="arch" type="xml">
  <field name="camp" position="replace"/>
</field>
```

Per afegir camps dins de l'element especificat d'una vista s'utilitzen els valors `inside`, `after` o `before`. Les següents plantilles mostren com s'inseriria `nou_camp` abans i després de `camp`:

```
<field name="arch" type="xml">
  <field name="camp" position="before">
    <field name="nou_camp" .../>
  </field>
</field>
```

```
<field name="arch" type="xml">
  <field name="camp" position="after">
    <field name="nou_camp" .../>
  </field>
</field>
```

Per efectuar canvis en més d'un lloc, cal combinar les plantilles anteriors dins un element `data`:

```
<field name="arch"type="xml">
  <data>
    <field name="camp1" position="after">
      <field name="nou_camp1"/>
    </field>
    <field name="camp2" position="replace"/>
    <field name="camp3" position="before">
      <field name="nou_camp3"/>
    </field>
  </data>
</field>
```

En vistes complexes o heretades d'altres vistes pot ser complicat identificar el camp a partir del qual aplicar els valors `inside`, `after`, `before` o `replace`. Penseu, per exemple, que es podria donar el cas que aparegués un camp amb el mateix nom en dos llocs diferents de la mateixa vista. Quan això passa es pot utilitzar l'element `xpath` per descriure amb exactitud la ruta d'etiquetes XML fins arribar a l'etiqueta en la qual cal realitzar els canvis.

1.1.3 Herència en el controlador

L'herència en el controlador és un mecanisme conegut, ja que l'apliquem de forma inconscient quan ens veiem obligats a sobreescriure els mètodes de la capa ORM d'OpenObject en el disseny de molts mòduls.

Funció super()

El llenguatge Python recomana utilitzar la funció `super()` per invocar el mètode de la classe base quan s'està sobreescrivint en una classe derivada, en lloc d'utilitzar la sintaxi `nomClasseBase.metode(self...)`.

L'efecte de l'herència en el controlador es manifesta únicament quan cal sobreescriure algun dels mètodes de l'objecte del qual es deriva i per a fer-ho adequadament cal tenir en compte que el mètode sobreescrit en l'objecte derivat:

- A vegades vol substituir el mètode de l'objecte base sense aprofitar-ne cap funcionalitat: el mètode de l'objecte derivat no invoca el mètode sobreescrit.
- A vegades vol aprofitar la funcionalitat del mètode de l'objecte base: el mètode de l'objecte derivat invoca el mètode sobreescrit.

1.1.4 Exemple d'herència

L'organització *Empresa_IOC*, gestionada amb un OpenERP estàndard amb el mòdul `base_contact` activat, té la imperiosa necessitat de poder introduir el color d'ulls dels contactes dels tercers (*partners*).

Per aquest motiu, l'equip informàtic d'*Empresa_IOC*, molt assenyadament, decideix no modificar els mòduls on radica el disseny de l'objecte `res.partner` per no perdre les modificacions en qualsevol actualització de l'OpenERP i dissenya un mòdul particular que implementa el nou requeriment, anomenant-lo `ioc_contact_eye_color`. El prefix `ioc` permet identificar, de manera ràpida, els mòduls que ha desenvolupat *Empresa_IOC* per a la seva organització.

En el nou mòdul s'ha decidit incorporar:

- La definició de l'objecte `ioc.color` per possibilitar la introducció dels diversos colors d'ulls que poden tenir els contactes dels tercers.
- A la pestanya *Informació Extra* de la vista `form` dels contactes, el camp *Eye color* per permetre assignar el color d'ulls al contacte.
- A la pestanya *General* de la vista `form` dels tercers (clients i/o proveïdors) el camp *Eye color*, de només lectura, després del camp *Funció* (`function`).
- A més, aprofitarem per tal que el camp *Website* de la pestanya *General* de la vista `form` dels contactes, possibiliti navegar a l'adreça continguda, fet que la versió 6.1 d'OpenERP no permet.

Per últim, s'ha decidit que no és necessari incorporar un formulari específic per donar d'alta els colors ni una opció de menú a l'apartat de configuració, ja que en el

camp que hem afegit en el formulari de contactes, OpenERP facilita un formulari automàtic per gestionar els colors (altes, baixes i modificacions).

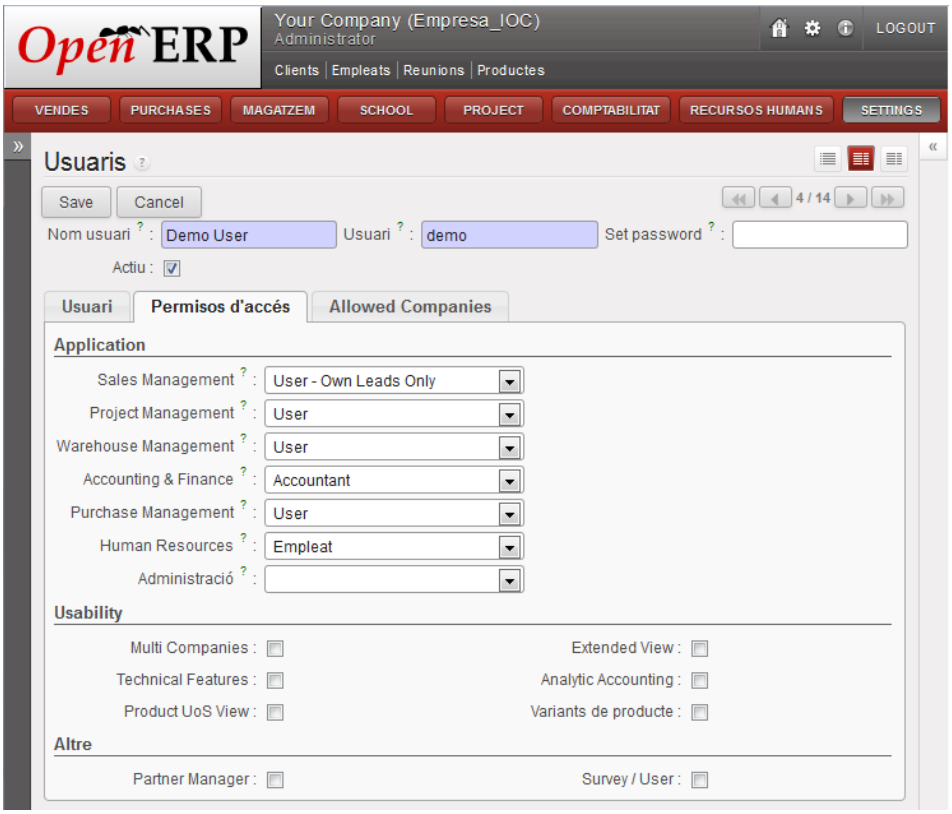
Trobareu el mòdul `ioc_contact_eye_color` dins l'apartat "Mòduls OpenERP i fitxers auxiliars" dels annexos del web.

1.2 Definició de la seguretat

L'apartat *Settings* de qualsevol empresa OpenERP conté dos apartats vinculats amb la seguretat en OpenERP: *Usuaris* (amb els subapartats *Usuaris* i *Grups*) i *Seguretat* (amb els subapartats *Regles de registres* i *Llista de controls d'accés*). Des d'aquests apartats, un usuari d'OpenERP, amb privilegis d'administració sobre l'empresa (usuari *admin* o similar) pot definir l'estratègia de seguretat que correspongui a l'empresa.

Els mòduls que s'instal·len en OpenERP poden incorporar un esquema de seguretat que defineix, com a mínim, grups de privilegis sobre els objectes dels mòduls, objectes que es poden veure a l'apartat *Settings* | *Usuaris* | *Grups*. A la seva vegada, cada grup de privilegis pot tenir associat un conjunt d'usuaris d'OpenERP. La figura 1.2 mostra els grups de privilegis que té assignats l'usuari *demo* d'OpenERP, que es crea en cas d'instal·lar l'empresa amb dades de demostració.

Figura 1.2. Grups de privilegis assignats a l'usuari demo d'OpenERP



La figura 1.2 correspon a una sessió de l'usuari *admin* en la qual podem observar que hi ha set mòduls instal·lats, corresponents a les pestanyes de la part superior, més l'opció *Settings*. Observem com la pestanya *Permisos d'accés* conté set apartats a la zona *Application*, un per cada mòdul instal·lat excepte pel mòdul

School. Les zones *Usability* i *Altre* tampoc tenen cap referència al mòdul *School*. Si obrim una sessió amb l'usuari *demo* comprovarem que no li apareix el mòdul *School*, tot i estar instal·lat a l'empresa.

En instal·lar un mòdul que no incorpori un sistema de seguretat, aquest mòdul únicament és accessible per l'usuari *admin*, encara que l'instal·li un altre usuari que tingui privilegis de configuració. Llavors, l'usuari *admin* haurà de crear manualment els grups de privilegis que cregui adequats sobre els objectes del mòdul i assignar-los als usuaris que corresponguin.

La majoria de mòduls incorporen un esquema de seguretat que pot ser més o menys sofisticat. La figura 1.2 ens mostra el grup de privilegis que té assignat l'usuari *demo* per a cada mòdul instal·lat (apartats *Application*). L'apartat *Settings | Usuaris | Grups* ens mostra la llista de tots els grups de privilegis definits per als mòduls instal·lats (figura 1.3). Fixem-nos en la jerarquia existent en els grups de privilegis definits (categoria/nomGrup). La llista de la figura 1.3 (incompleta) mostra: dos grups de privilegis (*Permisos d'accés* i *Configuració*) sota la categoria *Administració* (*Settings*), dos grups de privilegis (*User – Own Leads Only* i *Manager*) sota la categoria *Sales Mamangement* (mòdul *Vendes*), i així successivament. Cadascuna de les categories sota les quals s'aixopluguen diversos grups de privilegis es corresponen amb els apartats de la zona *Application* de la figura 1.2. A la figura 1.2, si despleguem la llista que acompanya la categoria *Sales Management*, hi veiem els dos grups de privilegis de la figura 1.3 sota *Sales Management*.

Figura 1.3. Grups de privilegis definits per als mòduls instal·lats en una empresa d'OpenERP

NOM GRUP
Administració / Permisos d'accés
Administració / Configuració
Human Resources / Empleat
Usability / Multi Companies
Usability / Extended View
Usability / Technical Features
Sales Management / User - Own Leads Only
Sales Management / Manager
Partner Manager
Usability / Analytic Accounting
Usability / Product UoS View
Usability / Variants de producte
Accounting & Finance / Invoicing & Payments
Accounting & Finance / Accountant
Accounting & Finance / Director
Warehouse Management / User
Warehouse Management / Manager
Sales Management / User - All Leads
Purchase Management / User
Purchase Management / Manager

El nostre interès se centra, en aquest moment, en saber com incorporar esquemes de seguretat en els mòduls que dissenyem, de manera que quan s'instal·lin ja hi hagi, com a mínim, un grup de privilegis definit, per tal que l'administrador només hi hagi d'assignar els usuaris.

Un esquema de seguretat bàsic d'un mòdul d'OpenERP es defineix en dos fitxers que s'acostumen a situar (no és obligatori) en una carpeta anomenada *security* i que han de ser referenciats des de l'apartat `update_xml` del fitxer __openerp__.py del mòdul. Aquests fitxers són:

- Fitxer XML que conté la definició dels grups (i de forma opcional: usuaris, menús i regles de negoci assignats a cada grup) que acostuma a anomenar-se nomModul_security.xml.
- Fitxer CSV que conté els privilegis de cada grup sobre els diferents objectes del mòdul i que s'anomena obligatòriament ir.model.access.csv. Aquest fitxer podria ser en format XML (llavors podria tenir qualsevol nom), però, a causa que la informació que conté es correspon amb el contingut d'una taula que sempre té el mateix format, és molt més senzill que sigui CSV ja que es pot emplenar ràpidament amb qualsevol full de càlcul.

El fitxer nomModul_security.xml ha de seguir una plantilla com la següent, que inclou un element `record` per cadascuna de les definicions que pot contenir el fitxer i que serà diferent segons el tipus de definició (grup o usuari, menú, regla assignada a un grup).

```
<?xml version="1.0" encoding="utf-8"?>
<openerp>
  <data noupdate="1">
    <record id="idGrup" model="res.groups">
      <field name="name">nomGrup</field>
      <field name="category_id" ref="..."/>
      <field name="implied_ids" eval="..."/>
      <field name="users" eval="..."/>
    </record>
    <record id="idUser" model="res.users">
      <!-- Crea/Modifica l'usuari "idUser"
        tot assignant-li grups -->
    </record>
    <record id="idMenu" model="ir.ui.menu">
      <!-- Crea/Modifica el menú "idMenu"
        tot assignant-li grups -->
    </record>
    <record id="idRegla" model="ir.rule">
      <!-- Definició de regles de negoci i assignació
        a grups i/o companyies -->
    </record>
  </data>
</openerp>
```

L'atribut `noupdate` de l'element `data` acostuma a tenir el valor 1 per indicar que en cas d'actualització de mòdul no s'instal·li l'esquema de seguretat que incorpora el mòdul, ja que podria sobreescriure l'esquema configurat per l'administrador de l'empresa, i el valor 0 per tal que s'instal·li sempre, sobreescrivint l'esquema existent. En cas que hi hagi parts de l'esquema de seguretat que no s'hagin de sobreescriure i altres que sí, es separen en dos elements `data` diferents, un amb `noupdate="1"` i l'altre amb `noupdate="0"`.

La plantilla anterior incorpora quatre tipus d'element `record` diferents. Posem especial atenció amb els que corresponen al model `res.groups`, que permeten definir els grups i al model `ir.ui.menu`, que permet assignar menús a un grup,

cosa que també es pot aconseguir des de la definició de menús. La resta de tipus d'elements record són complicats per una introducció al disseny d'esquemes de seguretat en OpenERP.

L'atribut `id` de l'element `record` del model `res.groups`, que ha de ser únic dins el mòdul, és un identificador XML per al grup, al qual es pot fer referència des del propi mòdul a través del seu nom o des de qualsevol altre mòdul a través de la sintaxi `nomMòdul.identificador`. Els atributs `id` dels elements `record` que no són del model `res.groups` poden ser nous (provocaran la creació d'un nou recurs) o existents (provocaran la seva modificació). Si ja existeixen i estan declarats en altres mòduls, cal introduir el nom del mòdul com a prefix: `nomMòdul.idRecurs`.

El valor de l'element `field` amb atribut `name="name"` és el nom del grup que es mostra a *Settings* | *Usuaris* | *Grups*.

El valor de l'element `field` amb atribut `name="category_id"` és per ubicar el grup per sota d'una categoria, que ha d'estar definida en el mateix mòdul o en un altre i que cal indicar a l'atribut `ref`. Per definir una nova categoria cal utilitzar l'element:

```
<record id="idCategoria" model="ir.module.category">
  <field name="name">nomCategoria</field>
  <field name="description">descripció</field>
  <field name="sequence">seq</field>
</record>
```

El valor de l'element `field` amb atribut `name="implied_ids"` és per indicar que la pertinença al grup suposa la pertinença automàtica als grups indicats a l'atribut `eval`, seguint la sintaxi:

```
eval="[(4,ref('idGrup1')),
      (4,ref('idGrup2')),
      ...
      ]"
```

L'escassa documentació sobre les combinacions que OpenObject facilita per poder assignar recursos existents en relacions `one2many` o `many2many` es troba, en OpenERP 6.1, a partir de la línia 3807 del fitxer `pathServer/openerp/osv/orm.py`.

Els grups de privilegis de cadascuna de les categories de l'apartat *Application* de la figura 1.2 tenen definida aquesta relació d'inclusió. Així, a la categoria *Sales Management* el grup *User - All Leads* inclou el grup *User - Own Leads Only* i el grup *Manager* inclou el grup *User - All Leads*. S'acostuma a anomenar *Manager* el grup de privilegis que conté privilegis totals sobre tots els objectes del mòdul.

Si entre els grups d'una categoria es dóna una relació d'inclusió com $A \supset B \supset C \supset \ldots$ no existirà mai la necessitat d'assignar a un usuari més d'un grup de privilegis. En tal situació, OpenERP ubica la categoria a la que pertanyin els grups de privilegis, a l'apartat *Application* (figura 1.2) en la posició que indiqui l'atribut `sequence` de la categoria. En canvi, si no tots els grups de la categoria estan relacionats sota una relació d'inclusió com $A \supset B \supset C \supset \ldots$ OpenERP ubica la categoria amb tots els seus grups de manera similar a la categoria *Usability* de la figura 1.2 que permet assignar diversos grups a un mateix usuari. Així, si els grups no s'assignen a cap categoria, OpenERP ubica els grups a la zona *Altre* de la figura 1.2.

El valor de l'element `field` amb atribut `name="users"` és per assignar al grup un o diversos usuaris dels que es coneix l'identificador XML, seguint la sintaxi:

```
eval="[(4,ref('idUsuari1')),
      (4,ref('idUsuari2')),
      ...
      ]"
```

Per exemple, per assignar l'usuari *admin* a un grup, caldria escriure:

```
eval="[(4, ref('base.user_root'))]"/>
```

Fixem-nos que per referir-nos a l'usuari *admin,* escrivim `base.user_root`, ja que l'usuari *admin* està definit en el mòdul `base` amb l'identificador `user_root`.

El fitxer ir.model.access.csv conté la informació dels privilegis assignats a cada grup sobre la totalitat dels objectes del mòdul. La primera línia conté, obligatòriament, el títol de les columnes:

```
"id","name","model_id:id","group_id:id","perm_read","perm_write","perm_create
    ","perm_unlink"
```

Les següents línies contenen les diverses assignacions de privilegis. Cal saber que:

- `id` és un identificador a decidir, que ha de ser únic i que no pot contenir cap punt.
- `name` és un nom que descriu el privilegi (pot contenir punts).
- `model_id:id` és el nom de l'objecte del model sobre el qual s'aplica el privilegi i ha d'anar precedit, forçosament, pel prefix `model_`. Si s'aplica sobre un objecte definit en un altre mòdul, cal utilitzar el prefix `nomMòdul.model_`.
- `group_id:id` és l'identificador del grup de privilegis (definit en el fitxerXML previ).
- `perm_read`, `perm_write`, `perm_create` i `perm_unlink` són valors 0/1 per indicar, respectivament, privilegis de lectura, escriptura, creació i eliminació sobre l'objecte.

La incorporació de línies en el fitxer ir.model.access.csv és molt fàcil amb la utilització de LibreOffice Calc.

És fonamental, en la definició dels grups i privilegis, que si es fa referència a una categoria o grup, aquests hagin estat prèviament definits. Per tant, la inclusió del fitxer ir.model.access.csv en el fitxer __openerp__.py ha de ser posterior a la inclusió del fitxer nomModul_security.xml.

A l'apartat "Mòduls OpenERP i fitxers auxiliars" dels annexos del web, trobareu la versió 12 del mòdul `school` a l'arxiu school_12.zip.

Exemple d'utilització d'esquema de seguretat en el mòdul school

Interessa incorporar un esquema de seguretat en el vigent mòdul `school` (versió *school_12*) que creï una categoria de nom *School* que contingui dos grups de privilegis:

Manager, amb permisos totals sobre tots els objectes del mòdul i *Subscriptions*, amb permisos adequats per poder assignar estudiants als cursos. Cal assignar l'usuari *admin* al grup *Manager*.

Si instal·leu la versió 13 del mòdul school podreu comprovar com a la llista de categories sota l'apartat *Application* de la figura 2 hi apareix la nova categoria *School* amb dos grups de privilegis (*Manager* i *Subscriptions*). Podreu comprovar que l'usuari *admin* té assignat el privilegi *Manager* i que cap altre usuari té assignat cap privilegi.

Si observeu el contingut de la versió 13 del mòdul `school` veureu que el fitxer `school_security.xml` incorpora la definició de la categoria *School* amb els dos grups *Manager* i *Subcriptions*. Podeu observar que el grup *Manager* inclou tots els privilegis del grup *Subscriptions* i que, a més, té assignat l'usuari *admin*.

Assigneu el grup de privilegis *Subscriptions* a l'usuari *demo* i procediu a obrir una sessió amb aquest usuari per efectuar subscripcions d'alumnes als cursos. Observeu que, d'entrada, només veu els menús que contenen opcions que gestionen objectes als quals té algun privilegi d'accés: *Configuration* | *Courses* amb permís de lectura als cursos i *Students* amb permisos per modificar el contingut d'estudiants i assignar-los cursos dels existents (vegeu les dues darreres línies de privilegis del fitxer ir.model.access.csv). Observeu, també, que per tal que el grup de privilegis *Subscriptions* permeti assignar cursos als estudiants, és necessari facilitar-li permís de lectura sobre els cursos.

Podeu trobar la versió millorada del mòdul `school` a l'arxiu school_13.zip dins l'apartat "Mòduls OpenERP i fitxers auxiliars" dels annexos del web.

1.3 Càrrega de dades

A vegades cal poder executar una càrrega massiva de dades en els objectes d'un mòdul. Algunes d'aquestes ocasions poden ser les migracions de dades en l'arrencada d'un OpenERP quan se substitueix un sistema informàtic; o bé la càrrega de dades de demostració quan s'instal·la qualsevol mòdul, si en el procés de creació de l'empresa es va indicar la càrrega de dades de demostració; i també la incorporació de dades inicials per al funcionament del mòdul.

En tots els casos és important efectuar la càrrega a través dels mecanismes facilitats per OpenERP i no intentar, de cap de les maneres, la introducció de dades a través d'instruccions SQL directes a la base de dades, ja que fent-ho a través d'OpenERP tindrem la seguretat que la lògica de negoci s'aplica en totes les operacions.

OpenERP permet preparar les dades a carregar en fitxers CSV i en fitxers XML. La principal diferència entre ambdós mètodes és que un fitxer CSV només permet la introducció de recursos (registres) d'un mateix objecte (taula), mentre que un fitxer XML permet la introducció de dades que afecten a diversos objectes (taules).

Pels fitxers CSV, el nom del fitxer ha de coincidir amb el nom de l'objecte (taula) on s'afegiran/actualitzaran els recursos (registres). La seva primera línia ha de contenir, obligatòriament, els noms dels camps de cada columna, i les següents línies han de contenir les dades dels recursos que s'han d'introduir. Un exemple és el fitxer ir.model.access.csv de la carpeta *security* dels mòduls que incorporen un esquema de seguretat.

Els fitxers XML segueixen una plantilla com la següent, en la qual cada element `record` conté les dades d'un recurs i el model al qual pertany, podent incorporar dades de recursos de diversos models.

```xml
<?xml version="1.0" encoding="utf-8"?>
<openerp>
  <data noupdate="1">
    <record id="idRecurs" model="nomObjecte">
      <field name="camp1">valor</field>
      <field name="camp2">valor</field>
      ...
    </record>
    <record id="idRecurs" model="nomObjecte">
      <field name="camp1">valor</field>
      <field name="camp2">valor</field>
      ...
    </record>
    ....
  </data>
</openerp>
```

L'atribut `noupdate` de l'element `data` acostuma a tenir el valor 1 per indicar que en cas d'actualització del mòdul no es carreguin les dades ja que podria sobreescriure les dades ja existents, i el valor 0 per tal que s'instal·li sempre, sobreescrivint les dades existents. En cas que hi hagi parts de la càrrega de dades que no s'hagin de sobreescriure i altres que sí, se separen amb dos elements `data` diferents, un amb `noupdate="1"` i l'altre amb `noupdate="0"`.

L'atribut `id` de cada element `record` que ha de ser únic dins el mòdul, és un identificador XML per al recurs, al qual es pot fer referència des del propi mòdul a través del seu nom o des de qualsevol altre mòdul a través de la sintaxi `nomMòdul.identificador`. Un exemple d'aquesta utilització el trobem quan necessitem fer referència, en algun lloc, a l'usuari *admin*, que escrivim `base.user_root`, ja que l'usuari *admin* està definit en el mòdul `base` amb identificador `user_root`.

Els fitxers amb dades de demostració a carregar (siguin CSV o XML) s'han d'indicar a l'apartat `demo_xml` del fitxer __openerp__.py del mòdul. Els fitxers amb dades a carregar únicament en la instal·lació del mòdul (siguin CSV o XML) s'han d'indicar a l'apartat `init_xml` del fitxer __openerp__.py del mòdul.

A l'apartat "Mòduls OpenERP i fitxers auxiliars" dels annexos del web trobareu el fitxer school_dades.xls.

Exemple de càrrega de dades de demostració en el mòdul school

Interessa incorporar un conjunt de dades de demostració en el vigent mòdul `school` (versió *school_13*) i el cap d'estudis d'un centre educatiu que ens vol ajudar, ens facilita informació en el fitxer school_dades.xls, que conté:

- Un full de càlcul anomenat `Organització`, amb tota la informació organitzativa sobre tres cicles formatius de la família professional d'Informàtica i comunicacions, amb els mòduls de cada cicle, les seves hores, la seva ubicació a primer o a segon curs, el nom del professor responsable de cada mòdul, amb el tipus de contracte (nomenat – interí) que assimilarem als conceptes que admet el camp `Contract` de l'objecte `Professor` del mòdul `school` (`named` – `trainee`). També conté dues columnes `Categoria` i `Subcategoria` corresponents al tipus de matèria dels mòduls, que no hem d'utilitzar.

- El full de càlcul `Alumnat` que conté les dades dels alumnes matriculats a cadascun dels cicles indicats en el full de càlcul anterior.

Per practicar els dos tipus de fitxers (CSV i XML) per introduir les dades de demostració, decidim utilitzar un fitxer XML per a les dades corresponents a cursos (cada mòdul de cada cicle el considerarem un curs) amb els seus professors i un fitxer CSV per introduir les

dades corresponents als alumnes i als cursos assignats, considerant que un alumne ha de tenir assignats els cursos corresponents als mòduls del cicle-curs en el qual està matriculat.

En la introducció de professors, decidim que tots ells dediquen 20 hores a la docència i que tots tenen per `Associated Partner` la companyia principal de l'empresa (`Empresa_IOC`).

La introducció de dades en el fitxer XML (professors i cursos) es pot fer manualment, amb un editor de textos pla, i no té gaire sentit muntar cap procés que, a partir del full de càlcul `Organització` generi el fitxer XML. Seria més costós muntar el procés que no pas introduir manualment les dades. Per aconseguir assignar la companyia principal de l'empresa a cada professor, anem a cercar al mòdul `base`, l'identificador XML de la companyia principal, que resulta ser `main_company`.

La introducció de dades en el fitxer CSV (estudiants i cursos assignats) és difícil de dur a terme manualment, a causa del volum d'informació que es genera i, en aquest cas, és més adequat cercar la manera de generar el fitxer CSV a partir dels fulls de càlcul `Organització` i `Alumnat` facilitats. Qualsevol programador té suficients recursos d'utilització dels fulls de càlcul per, a partir dels fulls facilitats, aconseguir un fitxer com school_auxiliar.xls, a partir del qual copiar els valors de les columnes H-L (només els valors, no les fórmules) en un nou full de càlcul per enregistrar-lo en format CSV.

La versió 14 del mòdul `school` es correspon a la versió millorada incorporant les dades de demostració aquí descrites. Per instal·lar-la, procediu prèviament a desinstal·lar la versió existent i instal·leu-la de nou, no actualitzant, ja que les dades de demostració només es carreguen en el procés d'instal·lació.

Observeu com, en el fitxer CSV, destinat a incloure els estudiants, s'indica, en una columna (`course_ids:id`) el conjunt de cursos que té assignat cada estudiant.

Per últim, cal comentar com seria un fitxer XML per a la inclusió dels estudiants, ja que per cada estudiant cal introduir el conjunt de cursos assignats. El contingut hauria de seguir la plantilla:

```
<?xml version="1.0" encoding="utf-8"?>
<openerp>
  <data noupdate="1">
    <record id="idStudent" model="school.student">
      <field name="idnum">valor</field>
      <field name="surname">valor</field>
      <field name="name">valor</field>
      <field name="course_ids" eval=
        "[(4,ref('idCurs1')),(4,ref('idCurs2')),...]"/>
    </record>
  </data>
</openerp>
```

Podeu trobar el fitxer school_auxiliar.xls i la versió 14 del mòdul `school` (arxiu school_14.zip) dins l'apartat "Mòduls OpenERP i fitxers auxiliars" dels annexos del web.

1.4 Programació d'assistents

Un assistent informàtic (*wizard* en anglès) és un programa pensat per abreujar o canalitzar els passos a seguir per dur a terme una tasca o, com a mínim, explicar els passos detalladament. En moltes ocasions s'utilitza per guiar als usuaris inexperts.

En OpenERP, un assistent és una successió de passos. Cada pas es composa de diverses accions:

- Mostra un formulari a l'usuari amb alguns botons.
- Recupera la informació introduïda per l'usuari i el botó seleccionat.

- Executa les accions que corresponguin.
- Envia una nova acció al client (formulari, informe...).

Per crear un assistent, cal definir un objecte OpenERP que no es fa persistent en el SGBD PostgreSQL, fet que s'aconsegueix fent que la classe Python que crea l'objecte OpenERP derivi de la classe `osv.osv_memory` en lloc de derivar de la classe `osv.osv`. La classe Python es defineix en un fitxer `.py` i la vista formulari es defineix en un fitxer XML, de forma similar a les classes que defineixen els objectes persistents d'OpenERP.

Els fitxers `.py` i XML corresponents als assistents d'un mòdul s'acostumen a ubicar en una carpeta anomenada *wizard* situada dins la carpeta del mòdul. El fitxer __init__.py del mòdul ha de contenir una sentència `import wizard` i la carpeta *wizard* ha de contenir un fitxer __init__.py amb la sentència `import` per al fitxer `.py` que conté les classes necessàries per a l'assistent. Els fitxers XML de l'assistent, ubicats normalment dins la carpeta *wizard*, han de ser referenciats des de l'apartat `update_xml` del fitxer __openerp__.py del mòdul.

Ja que els assistents defineixen una seqüència de passos, la classe que en defineix el model acostuma a tenir el camp especial `state`, de tipus `selection`, que permet definir els diversos estats pels quals passa l'assistent. El contingut d'aquest camp s'acostuma a utilitzar en la vista formulari per fer visibles i/o invisibles altres camps en funció del seu valor, de manera que l'usuari té la percepció que hi ha una successió de pantalles quan, en molts casos, és la mateixa però amb camps que canvien el seu estat de visibilitat.

Exemple de disseny d'un assistent en el mòdul school

Interessa incorporar uns assistents, en el vigent mòdul `school` (versió *school_14*), que permeti omplir el camp *First Date* d'alguns cursos existents (o de tots), amb una data introduïda per l'usuari.

En el menú *School* | *Configuration* es vol habilitar una nova opció de menú, de nom *Wizards*, que contingui dues opcions:

- Assignació d'una *First Date* comuna per tots els cursos, que ha de permetre que l'usuari introdueixi la data a assignar a tots els cursos existents a l'empresa, tinguin o no data introduïda.
- Assignació d'una *First Date* comuna per cursos filtrant per nom, que ha de permetre que l'usuari introdueixi la data a assignar a tots els cursos que el seu nom contingui un determinat text introduït per l'usuari.

En qualsevol cas, el procés ha de facilitar dos botons: *Cancel*, per cancel·lar el procés i *Assign*, per procedir a l'assignació. En cas d'executar l'assignació, l'assistent ha de mostrar una pantalla en la qual s'informi del nombre de cursos actualitzats amb un únic botó *Close* per tancar l'assistent.

La versió 15 del mòdul `school` es correspon a la versió millorada incorporant els assistents indicats. Observeu que conté una carpeta *wizard* amb els nous fitxers `.py` i XML.

El fitxer school_manage_course_date.py conté la definició de la classe no persistent `school_manage_course_date` que incorpora quatre camps: `new_date` per recollir la data que l'usuari vol assignar als cursos; `info_updates` per mostrar el nombre de cursos actualitzats; `course_name` per recollir el text que ha de contenir el nom en el segon assistent

demanat; i el camp `state` amb dos possibles valors (`Init` i `Done`) per distingir l'estat en el qual es troba el procés. Tot i que es demana dos assistents, com que són molt similars, ens serveix la mateixa classe `school_manage_course_date` per ambdós.

La classe `school_manage_course_date` incorpora dos mètodes: `assign_course_date_all`, que actualitza tots els cursos a partir de la data existent en el camp `new_date` del formulari, i `assign_course_date_name`, que actualitza els cursos el nom dels quals conté el contingut del camp `course_name` del formulari, amb la data existent en el camp `new_date` del formulari.

El fitxer school_manage_course_date_view.xml conté la definició dels menús indicats, amb les accions corresponents i les dues vistes (una per cada assistent). Les dues vistes es basen en el model `school.manage_course_date` i es diferencien en els camps que mostren (la corresponent al segon assistent, a diferència de la del primer assistent, mostra el camp `course_name`) i en el mètode que executa el botó *Assign*. Fixeu-vos que, per cadascuna de les vistes, s'aconsegueix l'aparença de dues pantalles diferents, a partir de la visibilitat i invisibilitat dels diferents camps i botons. La visibilitat es modifica a partir de l'estat en el qual es troba l'assistent.

El nou menú *wizards* s'ha assignat únicament al grup d'usuaris `group_school_manager`.

La versió 15 incorpora, en les vistes, a la part inferior esquerra, el camp `state` per tal que puguem veure com canvia el seu valor segons el moment en el qual es troba l'assistent. En la situació presentada, no té gaire sentit que el camp `state` sigui visible, però com que en altres casos sí que en pot tenir, l'hem incorporat per a veure'n la seva utilització, amb `widget="statusbar"`.

Per últim, una observació important sobre el contingut dels mètodes de la classe `school_manage_course_date` invocats pel botó *Assign* de les dues vistes. Fixeu-vos que aquests mètodes criden en dues ocasions el mètode `write`:

- Una, sobre l'objecte `course_obj` que gestiona els recursos del model `school.course`, indicant-li el conjunt de cursos a actualitzar (`course_ids`) i els camps a modificar (`values`):

```
1 values = {'date':new_date}
2 course_obj.write(cr, uid, course_ids, values)
```

- L'altra, sobre l'objecte `self`, és a dir, sobre l'objecte vinculat al formulari actiu, per actualitzar el contingut dels camps `info_udpates` i `state` del formulari i així aconseguir, en el formulari, la transició de la pantalla en què es demana la nova data (`state='init'`) a la pantalla en què s'informa del nombre de cursos actualitzats (`state='done'`):

```
1 values = {'state':'done','info_updates':len(course_ids),}
2 self.write(cr, uid, ids, values)
```

Podeu trobar la versió 15 del mòdul `school` (arxiu school_15.zip) dins l'apartat "Mòduls OpenERP i fitxers auxiliars" dels annexos del web.

Persistència del model dels assistents en versions 6.1 i 7.0 d'OpenERP

La documentació d'OpenERP afirma que el model que defineix un assistent, en derivar de la classe `osv.osv_memory`, no es fa persistent a la base de dades, com ha de ser, ja que no té cap sentit emmagatzemar les dades que l'usuari introdueix en l'assistent.

Les versions 6.1 i 7.0 d'OpenERP, però, fan persistents els models que deriven de la classe `osv.osv_memory` i les corresponents taules van enregistrant els valors introduïts pels usuaris en els assistents, fet que és totalment anòmal. Fins la versió 6.0 el funcionament era com ho documenta OpenERP. Sembla que és un error de les versions 6.1 i 7.0, ja que és una persistència mancada de sentit.

1.5 Traducció

OpenERP facilita mecanismes de traducció que permeten obtenir versions idiomàtiques de l'aplicació. En la creació d'una empresa en OpenERP, l'administrador d'OpenERP decideix l'idioma inicial en el qual s'instal·la l'empresa i l'empresa incorpora, d'entrada, l'idioma seleccionat i l'anglès. Una vegada l'empresa ha estat creada, l'administrador pot afegir nous idiomes a través de l'opció *Settings | Traduccions | Carrega una traducció oficial*. Els usuaris de l'empresa poden escollir el seu idioma (d'entre els idiomes instal·lats) des de l'opció *Preferences* que proporciona el client que utilitzen (web o GTK).

Les traduccions dels mòduls oficials i del client GTK es mantenen amb l'eina de traducció col·laborativa que proporciona Launchpad: translations.launchpad.net/openobject

L'idioma base d'OpenERP és l'anglès i, per això, es recomana que el desenvolupament s'efectuï en anglès per, posteriorment, traduir el mòdul als idiomes que interessi.

El procés d'instal·lació o actualització d'un mòdul en una empresa, instal·la el mòdul en els idiomes que l'empresa té activats, sempre que el mòdul incorpori les corresponents traduccions. Les traduccions que incorpora un mòdul estan ubicades a la subcarpeta *i18n* de la carpeta del mòdul i són un conjunt de fitxers amb extensió `.po` i nom identificador de l'idioma. Podeu comprovar-ho en qualsevol dels mòduls oficials que facilita OpenERP.

En informàtica s'acostuma a utilitzar el numerònim *i18n* per a la paraula *internationalization* a causa que entre la primera *i* i la darrera *n* hi ha 18 caràcters.

Les versions 6.x d'OpenERP utilitzen la codificació ISO 639-1 (ca.wikipedia.org/wiki/ISO_639-1) per als noms dels idiomes, de manera que el fitxer ca.po correspon a la traducció al català i el fitxer es.po correspon a la traducció al castellà. Per les variants d'un idioma en funció del país, s'utilitza la codificació xx_XX on xx correspon al codi de l'idioma i XX correspon al codi ISO 3166-1 alfa-2 (ca.wikipedia.org/wiki/ISO_3166-1). de manera que el fitxer es_AR.po correspon a la traducció a l'idioma espanyol d'Argentina i es_ME.po correspon a la traducció a l'idioma espanyol de Mèxic.

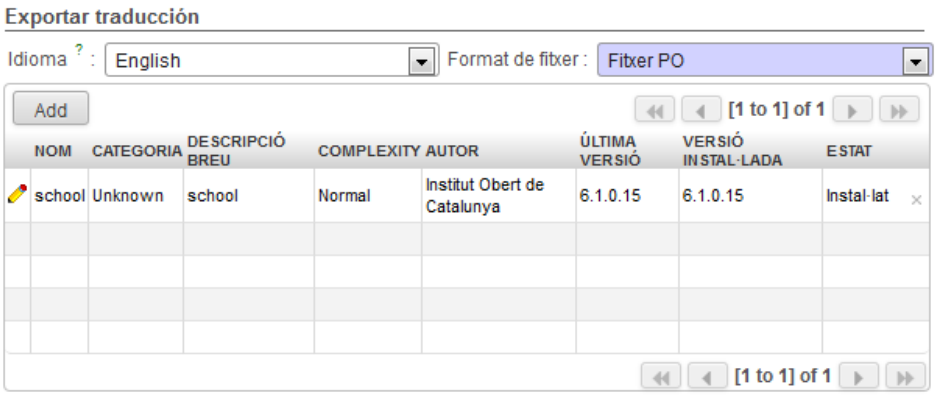
Per procedir a la traducció d'un mòdul que s'hagi elaborat en anglès, cal seguir els següents passos:

1. Tenir instal·lat, en una empresa, el mòdul a traduir.
2. Extreure, des d'OpenERP, per l'opció *Settings | Traduccions | Importa-Exporta | Exporta traducció*, tal com mostra la figura 1.4, tot indicant l'idioma *Anglès* (si el mòdul està desenvolupat en anglès, seguint les recomanacions), el format PO i el mòdul a traduir. L'eina d'extracció permet seleccionar altres formats, que no ens són interessants i permet exportar diversos mòduls, cosa que tampoc ens interessa perquè executa l'exportació en un únic fitxer PO i ens n'interessa un per a cada mòdul.
3. Enregistrar el fitxer resultant amb el nom del mòdul i extensió `.pot` a la subcarpeta *i18n* del mòdul. El fitxer `.pot` és el fitxer que conté els textos que OpenERP ha extret del mòdul i que traduirem als idiomes que ens interessi.
4. Per cada idioma al que vulguem traduir el mòdul, s'ha de copiar el fitxer nomModul.pot a un fitxer isoIdioma.po dins la mateixa carpeta *i18n*.

5. Traduir el contingut de cada fitxer isoIdioma.po a l'idioma que correspongui.

6. Executar el procés d'actualització del mòdul, que carregarà les traduccions existents.

Figura 1.4. Pantalla d'exportació de textos d'un mòdul per a la seva traducció



Els fitxers `.po` són fitxers de text pla que es poden editar amb qualsevol editor de textos. Un expert en fitxers `.po` podria efectuar la traducció amb qualsevol editor de textos, però si hem generat l'exportació en un fitxer `.po` en lloc d'un fitxer CSV, que és més fàcil d'editar, és perquè hi ha eines de traducció, com Poedit (www.poedit.net) que gestionen fitxers `.po` i que faciliten moltes funcionalitats.

Abans de procedir a la traducció de cap mòdul a una llengua, és molt important tenir una guia d'estil, i més en una aplicació com OpenERP, formada per una gran quantitat de mòduls i en la qual hi col·laboren nombroses persones. Així, us recomanem:

Podeu consultar la guia d'estil de Softcatalà a www.softcatala.org/wiki/Rebost:Guia_d'estil_de_Softcatalà, i la guia d'estil d'OpenERP a code.google.com/p/tinyerp-community/wiki/Traduccion_es_ES.

- La guia d'estil de Softcatalà, genèrica per a la traducció al català de productes informàtics.
- La guia d'estil d'OpenERP, per a la traducció de l'OpenERP al castellà.

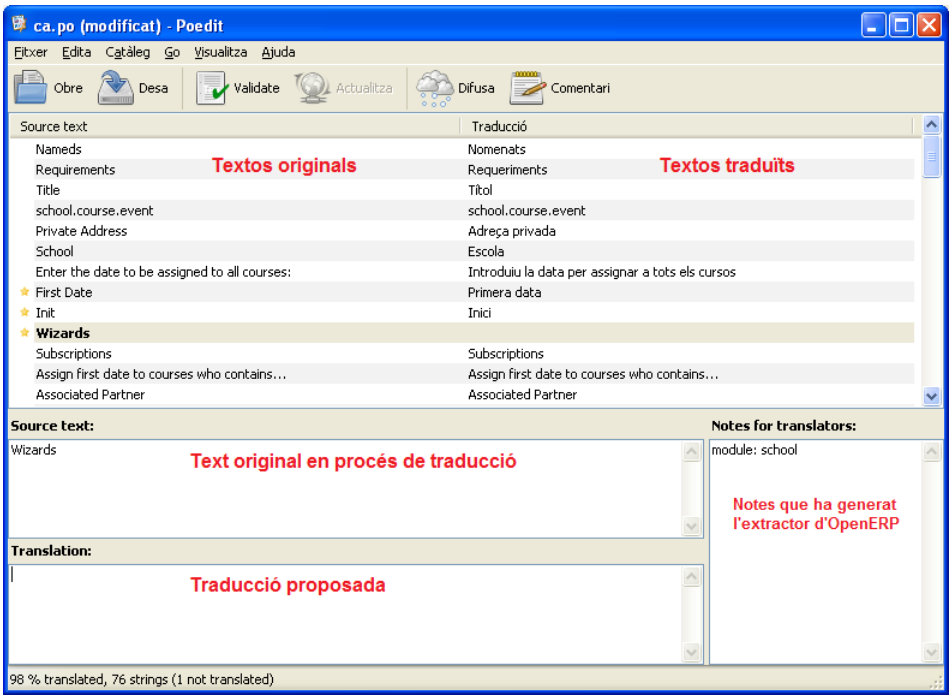
Exemple de com traduir les formes verbals

En català, la forma verbal que s'ha d'utilitzar en les situacions en les quals l'usuari s'adreça a l'ordinador per executar alguna acció (traducció de *Cancel, Exit, Close, Assign, Execute...*) és l'imperatiu en segona persona del singular (que correspon al tractament de "tu", com per exemple: *Cancel·la, Surt, Tanca, Assigna, Executa...*), mentre que en castellà és l'infinitiu (*Cancelar, Salir, Cerrar, Asignar, Ejecutar...*).

En català, la forma verbal que s'ha d'utilitzar en les situacions en les quals l'ordinador s'adreça a l'usuari per informar-lo o per fer-li alguna pregunta, és l'imperatiu en segona persona del plural (que correspon al tractament de "vós", com per exemple: *Seleccioneu, Voleu, Imprimiu, Assigneu...*), mentre que en castellà és l'imperatiu en segona persona del singular corresponent al tractament de "usted" (*Seleccione, Quiere, Imprima, Asigne...*).

Amb l'eina Poedit instal·lada, obrim el fitxer `.po` que interessi i procedim a la seva traducció. La utilització de l'eina Poedit és molt simple i intuïtiva, com mostra la figura 1.5.

Figura 1.5. Interfície del programa Poedit



Entre les funcionalitats que facilita Poedit, cal saber que:

- Permet posar comentaris a cada terme traduït.
- Permet marcar la traducció d'un terme com a *difusa* quan no n'estem convençuts, de manera que en un altre moment la podem identificar fàcilment.
- Permet saber el punt del mòdul del qual s'ha extret el text, fet que en ocasions ens pot ajudar a efectuar la traducció del terme en funció del seu context.
- Permet incorporar traduccions d'una nova versió de fitxer `.pot` al fitxer `.po` actual, fet que és molt d'agrair quan es retoca un mòdul pel qual ja hi havia una traducció efectuada.

Podeu trobar la versió 15 del mòdul `school`, traduïda al català, en l'arxiu school_15t.zip, dins l'apartat "Mòduls OpenERP i fitxers auxiliars" dels annexos del web.

L'eina d'extracció de textos d'OpenERP extrau, erròniament, els noms dels models que es creen en els fitxers `.py`, que no hem de traduir.

Exemple de textos extrets per OpenERP que no es poden traduir

Si editeu amb Poedit el fitxer ca.po de l'arxiu school_15t.zip, hi veureu els textos corresponents als models definits en el mòdul, que no s'ha traduït:

```
1  school.course
2  school.course.event
3  school.manage.course_date
4  school.professor
5  school.student
```

Finalment, cal recordar que en l'escriptura de mètodes i funcions en OpenObject, els textos de missatges inclosos en mètodes i funcions, per poder ser traduïbles han de ser introduïts amb la sintaxi `_('text')` i el fitxer `.py` ha de contenir `from tools.translate import _` a la capçalera.

2. Reporting & BI

Les solucions informàtiques per a la gestió comercial (ERP, CRM, HRM...) incorporen un conjunt predefinit d'informes i quadres de comandament que acostumen a ser adequats, i potser suficients, per a un nombre important d'organitzacions. Sempre hi ha, però, organitzacions que necessiten informes i quadres de comandament diferents dels facilitats per l'aplicació informàtica i això és també imprescindible quan es dissenyen mòduls que complementen el programari existent. Així doncs, ens convé disposar d'eines de *reporting* i *business intelligence* (BI) que ens permetin elaborar els informes i quadres de comandament que l'organització necessiti.

Els programaris de gestió empresarial actuals acostumen a incorporar eines que faciliten el disseny d'informes i quadres de comandament, integrats en l'aplicació i elaborats a partir de les dades gestionades pel propi sistema. Podem trobar-nos, però, en situacions en què calgui accedir a orígens de dades externs, possibilitat normalment no facilitada per les eines BI del sistema o en casos en els quals el programari no incorpori eines de *reporting* i *business intelligence*. En conseqüència, convé conèixer a fons les eines BI que facilita el sistema i les eines BI externes que complementin i/o substitueixin les del sistema.

2.1 Reporting extern a l'ERP: JasperReports

JasperReports (community.jaspersoft.com/project/jasperreports-library) és un motor de generació d'informes molt potent, sota llicència LGPL (programari lliure) proveït per Jaspersoft Corporation i escrit totalment en Java, que presenta les característiques següents:

- Facilita connectors per un gran nombre d'orígens de dades.
- Permet dirigir els informes cap a pantalla o cap a impressora o cap a fitxer en múltiples formats (PDF, RTF, XML, XLS, CSV, HTML, XHTML, text, DOCX o OpenOffice).
- Permet incloure gràfics (de barres, de sectors...) en els informes.
- Permet incloure informes (subinformes) dins altres informes.
- Permet incorporar filtres (paràmetres) en els informes, de manera que en el moment d'execució l'usuari pot introduir els valors amb què desitja que es confeccioni l'informe.
- Permet ser incrustat (hostatjat) en aplicacions desenvolupades en diferents llenguatges, és a dir, des del programa es pot invocar el motor JasperReports

per generar informes. Els llenguatges Java i PHP són un exemple d'aquesta possibilitat.

Pels programadors té molt d'interès conèixer un motor d'informes com JasperReports que pot ser invocat des d'aplicacions desenvolupades en diferents llenguatges.

El motor JasperReports genera l'informe a partir d'un document que conté el disseny de l'informe i d'un origen de dades al qual es connecta per anar a cercar les dades que bolca a l'informe. Ens convé, doncs, saber elaborar els documents que contenen el disseny dels informes en el format indicat per JasperReports.

Els documents que contenen el disseny d'un informe per a JasperReports són fitxers XML amb extensió `.jrxml` que podrien ser confeccionats amb qualsevol editor de textos, en cas de ser profunds coneixedors del "llenguatge JRXML". Per sort, en el mercat hi ha diferents eines de disseny d'informes per a JasperReports, que permeten dissenyar els informes des d'una interfície gràfica. Una d'aquestes eines és iReport Designer (http://community.jaspersoft.com/project/ireport-designer), proveït per Jaspersoft Corporation.

El motor JasperReports llegeix el fitxer `.jrxml` i en fa una compilació, generant un fitxer `.jasper` a partir del qual genera l'informe desitjat. L'execució és més ràpida si es facilita el fitxer JASPER.

L'eina iReport Designer, sota llicència AGPL, facilita una interfície gràfica que permet:

- Dissenyar sofisticats informes que contenen gràfics, imatges, subinformes, taules i molt més, en formats `.jrxml`, amb els quals es pot generar el compilat `.jasper`.
- Accedir als mateixos orígens de dades que facilita el motor JasperReports (a través de JDBC, TableModels, JavaBeans, XML, Hibernate, CSV i fonts personalitzades).
- Generar l'informe (incorpora el motor JasperReports) i publicar-ne el resultat per pantalla, impressora o en els mateixos formats que facilita el motor JasperReports (PDF, RTF, XML, XLS, CSV, HTML, XHTML, text, DOCX o OpenOffice)

Així doncs, disposem d'un motor d'informes (JasperReports) i d'una eina de disseny d'informes (iReport Designer) des de la qual es poden dissenyar informes i executar-los. JasperReports i iReport Designer, són útils per a una empresa amb un sistema de gestió empresarial sempre que l'empresa disposi d'un departament d'informàtica que desenvolupa programes en llenguatges en els quals es pot hostatjar el motor d'informes. En cas contrari, el grau d'utilitat baixa en picat, ja que per executar un informe caldria tenir instal·lat l'iReport Designer, disposar del disseny de l'informe i dels paràmetres de connexió amb el corresponent origen de dades i executar l'informe des de l'eina de disseny. És clar que manca una peça: una aplicació servidora d'informes.

JasperReports Server (http://community.jaspersoft.com/project/jasperreports-server), proveït per Jaspersoft Corporation, és un servidor d'informes i quadres de comandament que pot ser executat de forma independent, com un nucli d'informació per l'organització o incrustat en una aplicació web o mòbil. N'hi ha dues versions: la professional, de pagament, sota el nom de Jaspersoft 5 Business Intelligence Suite Enterprise i la comunitària, amb llicència AGPL, sota el nom de JasperReports Server Community. Ambdues versions faciliten una interfície

web per administrar el servidor i executar els informes i quadres de comandament allí ubicats. La taula 2.1 recull algunes funcionalitats que diferencien les dues versions.

Taula 2.1. Diferències entre les versions Enterprise i Community de JasperReports Server

Funcionalitat	Enterprise	Community
Ubicar i executar informes	Sí	Sí
Definir consultes i vistes	Sí	Sí
Definir camps de filtre i llistes de valors	Sí	Sí
Dissenyar informes	Sí	No
Dissenyar i executar quadres de comandament	Sí	No
Gestionar usuaris, rols i assignació de privilegis	Sí	Sí
Gestionar diverses organitzacions	Sí	No

La terna d'aplicacions formada pel motor d'informes **JasperReports**, l'eina de disseny **iReport Designer** i el servidor **JasperReports Server** (Community o Enterprise) pot ser molt útil per a qualsevol organització.

Una empresa que vulgui utilitzar JasperReports Server com a nucli d'informació en el qual ubicar un conjunt d'informes i quadres de comandament de l'empresa, amb accés diferenciat segons els usuaris, ha de seguir, com a mínim, els següents passos:

1. Instal·lar el servidor JasperReports Server (la versió adequada segons les prestacions requerides) en la màquina que es cregui convenient. No té perquè ser la mateixa màquina on resideix la base de dades que gestiona el sistema de gestió empresarial de l'organització, però sí que hauran de ser visibles per la xarxa, ja que JasperReports Server ha de tenir accés a la base de dades.

2. Instal·lar iReport Designer en les màquines dels usuaris que hagin de poder dissenyar informes amb aquesta eina.

3. En el servidor d'informes, crear l'esquema de seguretat (organitzacions, rols i usuaris) que correspongui i definir els accessos als orígens de dades del qual s'alimenten els informes.

4. Dissenyar els informes de l'organització amb iReport Designer i pujar-los al servidor, assignant els permisos d'accés que corresponguin.

5. En el cas de JasperReports Server Enterprise, dissenyar els quadres de comandament des de la interfície web i assignar-los els permisos d'accés que corresponguin.

2.1.1 Instal·lació i configuració de JasperReports Server Community 5.0

Ens disposem a instal·lar la versió Community de JasperReports Server en una màquina Windows per, bàsicament, instal·lar informes basats en la base de dades d'una empresa d'OpenERP. No hem d'oblidar, però, que podem utilitzar JasperReports Server per instal·lar informes sobre qualsevol base de dades. JasperReports Server Community també està disponible per a màquines Linux i Mac.

JasperReports Server necessita, pel seu funcionament, un servidor web Tomcat i un SGBD PostgreSQL. El paquet d'instal·lació incorpora els dos programaris (Tomcat i PostgreSQL) i el procés d'instal·lació demana, per cadascun dels programaris, si es vol que JasperReports n'instal·li la versió incorporada o si es vol utilitzar una versió ja existent. En cas de voler utilitzar un servidor PostgreSQL existent, cal que sigui de versió 9.x i que tingui l'usuari de nom *postgres* com a superusuari.

Ja que bàsicament utilitzarem JasperReports Server per informes sobre OpenERP i que les dades de les empreses gestionades per OpenERP resideixen en un servidor PostgreSQL, és una bona opció instal·lar JasperReports Server a la mateixa màquina en la qual resideix el servidor PostgreSQL que dóna servei a OpenERP, per tal d'utilitzar el mateix servidor PostgreSQL. Compte, però, amb la versió de PostgreSQL, que ha de ser 9.x o superior per a JasperReports Server 5.0. S'ha de tenir present que la versió de PostgreSQL que instal·la OpenERP 6.1 és la 8.3.4. També es pot indicar al procés d'instal·lació de JasperReports Server que instal·li l'SGBD PostgreSQL que incorpora en una màquina que ja disposi d'un PostgreSQL instal·lat, tot indicant-li que escolti per un port diferent del que està escoltant el PostgreSQL instal·lat.

El procés d'instal·lació segueix els passos següents:

1. Demana si es vol instal·lar el servidor Tomcat incorporat o si es vol utilitzar un servidor Tomcat existent. Indicarem que instal·li el servidor Tomcat incorporat.

2. Demana si es vol instal·lar el servidor PostgreSQL incorporat o si es vol utilitzar un servidor PostgreSQL existent. Decidirem segons les observacions anteriors. En cas que decidim utilitzar un PostgreSQL existent, ens informa que intentarà instal·lar les bases de dades que necessita i en sobreescriurà qualsevol duplicat. La base de dades que utilitza JasperReports Server s'anomena *jasperserver* i la sobreescriuria en cas de trobar-ne una amb el mateix nom.

3. Demana la configuració dels ports pels quals escoltarà el servidor Tomcat, proposant els valors 8080 per *Tomcat Server Port* i 8005 per *Tomcat Shutdown Port*. Cal vigilar que no tinguem altres servidors instal·lats que utilitzin aquests ports. La versió 6.1 d'OpenERP utilitza, si no s'ha indicat el contrari, els ports 8069 i 8070.

4. En cas que, en el punt 2, haguem decidit utilitzar un PostgreSQL existent, ens demana:

    - El directori en el qual resideixen els programes (subcarpeta *bin*) de PostgreSQL (*PostgreSQL Binary Directory*). En principi, el servidor PostgreSQL no té perquè residir a la mateixa màquina on estem instal·lant JasperReports Server, però ha de contenir una còpia de la carpeta *bin* del servidor PostgreSQL.
    - La IP o nom de la màquina en la qual resideix el servidor PostgreSQL (127.0.0.1 si és a la mateixa màquina).
    - El port pel qual escolta el servidor PostgreSQL (normalment 5432).
    - La contrasenya del superusuari *postgres* del servidor PostgreSQL. Obligatòriament ha d'existir un superusuari anomenat *postgres*.

5. En cas que, en el punt 2, haguem decidit instal·lar un nou servidor PostgreSQL, ens demana el port pel qual ha d'escoltar, proposant 5432. El servidor PostgreSQL que instal·la té l'usuari *postgres* com a superusuari amb contrasenya *postgres*.

6. Demana si es desitja la instal·lació de bases de dades i informes d'exemple. En cas afirmatiu, instal·la dues bases de dades anomenades *sugarcrm* i *foodmart*, que serien sobreescrites en cas d'existir (instal·lació en un PostgreSQL existent). Aconsellem, en una primera instal·lació, respondre afirmativament, ja que així podrem veure les possibilitats que permeten els informes dissenyats per JasperReports.

7. Demana si es desitja la instal·lació de l'eina de disseny iReport Designer. Si volem dissenyar informes des de la mateixa màquina on resideixi JasperReports Server, és adequat instal·lar-la, tot i que sempre la podrem instal·lar ja que Jaspersoft Corporation facilita el paquet instal·lador individual d'iReport Designer.

Una vegada executada la instal·lació, podem comprovar que:

- El servidor PostgreSQL (nou o existent) conté la base de dades *jasperserver* i, si hem demanat d'instal·lar els exemples, les bases de dades *sugarcrm* i *foodmart*.
- En el panell de control dels serveis de Windows, ha aparegut el servei *jasperreports Tomcat* amb tipus d'inici automàtic i, si hem decidit la instal·lació d'un servidor PostgreSQL, el servei *jasperreports PostgreSQL* amb tipus d'inici automàtic.
- En l'arbre de programes de Windows hi trobem les següents opcions:
    - *Start or Stop Services*: amb les opcions per engegar o aturar els serveis jasperreports instal·lats.
    - *JasperReports ServerDocumentation*: que ens obre una carpeta del sistema d'arxius en la qual ha instal·lat diversos documents.

- *JasperReports Server Login*: que porta a la interfície web facilitada per JasperReport per a tasques administratives i per a execució dels informes i quadres de comandament ubicats en el servidor
- *Jaspersoft Community Website*: per navegar al portal de Jaspersoft Community.
- *Jaspersoft Website*: per navegar al portal de Jaspersoft Corporation.
- *Start iReport Designer*: per posar en marxa l'eina iReportDesigner, en cas d'haver-la instal·lat.
- *Uninstall JasperReports Server*: per desinstal·lar el programari.

Engeguem els servidors (Tomcat i PostgreSQL) i executem l'opció *JasperReports Server Login* que ens obre la interfície web en la pantalla inicial de connexió, tal com mostra la figura 2.1.

La figura 2.1 ens mostra que la URL http://maquina:port/jasperserver/login.html porta a la pàgina inicial de JasperReports Server Community, en la qual `maquina` correspon a la identificació de la màquina (ip o nom) i `port` correspon al port pel qual està escoltant Tomcat.

JasperReports Server admet diversos idiomes i la interfície web es mostra en la primera llengua de presentació definida en el navegador. Si JasperReports Server no l'admet, es presenta en llengua anglesa. Cal saber també que els informes poden estar traduïts en diversos idiomes i JasperReports Server mostrarà l'informe en el mateix idioma en què treballi la interfície.

JasperReports Server 5.0 no admet el català. Podeu veure la interfície i els informes en castellà situant-lo com a primer idioma de presentació en el navegador.

FIGURA 2.1. Pantalla inicial de la interfície web de JasperReports Server

JasperReports Server Community instal·la tres usuaris: *anonymousUser* (sense contrasenya) que no té assignat cap tipus de permís, *jasperadmin* (contrasenya *jasperadmin*) amb permisos totals sobre el servidor i *joeuser* (contrasenya *joeuser*)

amb permisos d'usuari final. Obriu sessió amb els usuaris *jasperadmin* i *joeuser* per apreciar les diferents opcions disponibles en cada cas.

Llegiu la documentació que incorpora JasperReports Server Community per obtenir un coneixement acurat de totes les seves possibilitats.

En aquest moment, com a desenvolupadors d'informes, ens convé fer una ullada als exemples que conté JasperReports Server Community, per fer-nos una idea de la tipologia d'informes que podem desenvolupar. No entrarem en qüestions puramente administratives (gestió d'usuaris, rols i assignació de privilegis) destinades als administradors del sistema.

La relació d'informes que un usuari pot executar es troben a l'apartat *Library* (Biblioteca), on se'ns presenta una graella dels informes existents, amb les següents columnes:

- Nom de l'informe.
- Breu descripció de l'informe.
- Tipus, que en el cas de la versió Community sempre és *Report*, ja que aquesta versió no permet quadres de comandament.
- Dates de creació i de modificació.

La interfície web de JasperReports Server també facilita l'apartat *View | Repository* amb accés a tots els elements existents en el servidor, organitzats per carpetes amb diversos continguts. Els elements *Report* relacionats a l'apartat *Library* es poden localitzar en les diferents carpetes del *Repository*.

Si en el procés d'instal·lació de JasperReports Server hem permès la instal·lació de bases de dades i informes d'exemple, disposem d'un conjunt d'informes que ens permeten copsar les possibilitats d'aquest entorn de *reporting*. És altament aconsellable efectuar-hi una ullada. Els següents exemples corresponen a un recull dels que poden ser més interessants per al nostre aprenentatge.

Exemple d'informe: Customers Report

Si executeu aquest informe, que pel títol i la descripció sembla que ha de ser un llistat de clients, veureu que correspon a un llistat de les comandes de clients, agrupades per client, amb uns imports sumaritzats a peu de cada client.

En ser un informe de moltes pàgines, a la part superior esquerra de la interfície se'ns facilita un control per poder moure'ns per les diferents pàgines.

La part superior dreta de la interfície mostra diversos botons, alguns activats i altres desactivats. Entre els activats, disposem del botó *Back* que serveix per tirar enrere (abandonar l'informe) i del botó *Export* que permet obtenir el llistat en diversos formats (PDF, Excel, CSV, DOCx, RTF, Flash, ODT, ODS i XLSX).

Exemple d'informe: Employee Accounts

La descripció d'aquest informe ens diu que es tracta d'una llista dels comptes (clients) per empleat. Amb aquesta informació podem pensar que es tracta d'un llistat similar a l'anterior, en el qual apareixen els clients de l'empresa agrupats per empleat.

L'execució d'aquest informe ens mostra una pantalla amb el missatge *The report is empty*. Bé, podem pensar que la base de dades no té empleats introduïts. Però, a la part superior dreta de la interfície, observem el botó *Options* actiu. Si el premem, ens mostra una llista desplegable dels empleats, amb l'empleat *jim* seleccionat. Seleccioneu un altre empleat

i executeu de nou el llistat, prement el botó *OK* sota la llista desplegable. Proveu amb els empleats *jaime* i *matt*. Per cadascun d'aquests empleats es genera el corresponent informe.

Fixeu-vos en un greu error de disseny per aquest informe: no mostra, per enlloc, la informació de l'empleat pel qual s'està mostrant la llista de clients. Normalment, a la capçalera de l'informe s'introdueix informació explicativa sobre el contingut de l'informe, així com el moment temporal (data i hora) en la qual l'informe ha estat generat.

Exemple d'informe: Employee List

Aquest informe, pel seu nom, sembla que ha de ser un simple llistat dels empleats de l'empresa i, si l'executeu, veureu que així és.

Fixeu-vos, però, en la darrera columna *Accounts*, que fa referència als clients que té cada empleat, visualitzats en l'informe anterior. El valor d'aquesta columna per a cada empleat és un enllaç anomenat *view* que, si el premem, ens executa l'informe anterior per a l'empleat seleccionat. És a dir, **JasperReports Server proporciona navegabilitat entre informes**.

Exemple d'informe: Department

La descripció d'aquest informe ens diu que conté *input controls* (camps de filtre) i per això és del nostre interès.

En executar-lo, sense aparèixer cap camp de filtre, se'ns mostra una determinada informació. A la part superior dreta de la interfície observem el botó *Options* actiu. Si el premem, s'obre la finestra *Input Controls* amb tres possibilitats de filtre:

- Sexe: amb una llista multiselecció amb els valors possibles.
- Estat civil: amb dues caselles de selecció.
- Llista desplegable: per seleccionar el departament dels empleats.

Exemple d'informe: Cascading multi select example report

En executar aquest informe, a la part esquerra de la interfície apareix la zona *Options* que permet efectuar una selecció en cascada.

En primer lloc apareix una llista multiselecció de països.

En segon lloc, una llista multiselecció d'estats, en la qual només apareixen els estats dels països seleccionats en l'anterior camp de filtre.

En tercer lloc, una llista desplegable per seleccionar un client, en la qual només apareixen els clients dels estats seleccionats en l'anterior camp de filtre.

L'informe mostra, pels estats seleccionats, els primers quinze clients incloent-hi el client indicat en el tercer camp de filtre, malgrat que no estigui entre els primers quinze clients.

2.1.2 Utilització d'iReport Designer 5.0

L'eina iReport Designer és la que utilitzarem per dissenyar informes que posteriorment pujarem al servidor JasperReports Server. Aquesta eina pot estar instal·lada a qualsevol màquina. És una aplicació Java que necessita disposar d'un entorn JDK de Java per al seu funcionament.

La instal·lació de JasperReports Server facilita la possibilitat d'instal·lar l'iReport Designer i, a causa que JasperReports Server incorpora un entorn JDK de Java, l'iReport Designer instal·lat conjuntament amb JasperReports Server funciona sense necessitat d'instal·lacions complementàries de JDK.

La instal·lació d'iReport Designer independentment de JasperReports Server obliga a tenir un entorn JDK de Java a la màquina. La instal·lació d'iReport Designer en una màquina Windows és molt simple i únicament cal seguir les indicacions. En intentar l'execució de l'iReport Designer no es posa en marxa (i no apareix cap error) si no troba l'entorn JDK de Java instal·lat. Com que és possible que en una màquina convisquin diversos entorns JDK de Java, es pot indicar a iReport Designer la versió a utilitzar, en el paràmetre `jdkhome` del fitxer ireport.conf ubicat a la carpeta *etc* de la ubicació en la qual s'ha instal·lat iReport Designer.

Seguiu les indicacions de l'annex "Utilització de l'iReport Designer" per introduir-vos en la utilització d'aquesta eina.

Una vegada ens hem introduït en la utilització de l'iReport Designer, cal dur a la pràctica els diversos passos per assolir informes adequats a les necessitats de l'organització:

1. Definició de Report Datasource per connectar amb la base de dades.
2. Disseny d'informe simple (columnes).
3. Disseny d'informe mestre-detall.
4. Pujada d'informes a un JasperReport Server.
5. Incorporació de filtres a un informe.

Definició de Report Datasource

En qualsevol eina de *reporting* és necessari definir un origen de dades des d'on el motor de *reporting* obté les dades per poder elaborar l'informe. El motor JasperReport anomena *Report Datasource* a l'origen de dades i és necessari tenir-lo definit en l'iReport Designer, en temps de disseny, per poder disenyar l'informe i, també, en el JasperReport Server, en temps d'execució.

Definició d'un Report Datasource per connectar amb la base de dades Empresa_IOC del servidor OpenERP

Es desitja utilitzar iReport Designer per obtenir informes sobre la base de dades Empresa_IOC del servidor OpenERP i, per aquest motiu, ens cal definir, dins l'iReport Designer, el connector *Report Datasource* adequat.

Per aconseguir-ho, premem el botó *Report Datasources* i afegim un nou *Datasource* de tipus *Database JDBC Connection*, ja que l'SGBD d'OpenERP és PostgreSQL i JasperReports incorpora connector JDBC per connectar amb els SGBD PostgreSQL.

Emplenem els diferents apartats que se'ns demana, com segueix:

- *Name*: Empresa_IOC o qualsevol nom entenedor.
- *JDBC Driver*: escollim PostgreSQL.

- *JDBC URL*: emplenem l'URL de forma adequada segons la plantilla `jdbc:postgresql://maquina:port/nomBD`.
- *Username* - *Password*: usuari amb accés a la base de dades.

iReport Designer ens permet deixar enregistrada la contrasenya, de manera que no ens la demani en les successives ocasions en què la necessiti, però ens avisa del perill que hi ha ja que la contrasenya no queda encriptada. Tot i així, com que no estem en un entorn real de producció, és aconsellable deixar-la enregistrada ja que del contrari iReport Designer ens demanarà la contrasenya contínuament.

La pantalla que ens permet introduir aquestes dades, facilita un botó *Test* per comprovar la connectivitat del nou *Datasource*.

Disseny d'un informe simple

En l'aprenentatge del disseny d'informes el més adequat és iniciar-se tot dissenyant informes simples, consistents en un conjunt de columnes que contenen la informació.

Cal tenir present que els diversos informes d'una organització han de seguir la imatge corporativa (disseny similar, amb logo de l'empresa a la capçalera de l'informe) i no hem d'oblidar-nos:

- D'introduir a la capçalera les dades referents a la informació que conté l'informe.
- D'introduir a l'informe (capçalera o peu) la paginació i la data i hora d'elaboració de l'informe.

Trobareu una possible solució en el fitxer professors_PSQL.jrxml dins l'apartat "Mòduls OpenERP i informes" dels annexos del web. També hi trobareu el logo LogoIOC.png.

Disseny d'un informe de professors del mòdul school d'OpenERP

Volem dissenyar un informe dels professors del mòdul `school` d'OpenERP, amb les següents característiques:

- Columnes: nom, contracte, nombre d'hores que treballa setmanalment, telèfon, correu electrònic i adreça (carrer, codi postal i ciutat).
- Títol i logotip a la primera pàgina.
- Data i hora d'execució del llistat a la part superior esquerra de cada pàgina.
- Número de pàgina acompanyat del total de pàgines a la part superior dreta de cada pàgina.
- Títols de les columnes, a cada pàgina.
- Professors ordenats per nom.

Per comprovar l'execució de l'informe des d'iReport Designer, heu de tenir seleccionat un *Datasource* que apunti a la base de dades d'OpenERP en la qual teniu instal·lat el mòdul `school`. A més, el logo de l'informe (fitxer LogoIOC.png) ha de residir a la mateixa carpeta en la qual estigui ubicat l'informe.

Observeu que l'execució de l'informe provoca la generació d'un fitxer compilat anomenat nomInforme.jasper.

Pel que fa al disseny de l'informe, fixeu-vos que:

- La sentència SQL en la qual es basa, necessita relacionar les taules `school_professor` i `res_partner_address` amb un `outer join`, per garantir que apareguin els professors que no tinguin assignada cap adreça.
- Tot i que iReport Designer permet indicar ordenacions dels registres, és millor indicar la clàusula `ORDER BY` a la sentència SQL, ja que del contrari, el motor JasperReports ha d'esperar a tenir tots els registres que li subministra l'SGBD per efectuar-ne una ordenació en memòria i posteriorment confeccionar l'informe. En canvi, si l'SGBD facilita els registres ja ordenats, el motor d'informes pot anar confeccionant l'informe malgrat que no tingui tots els registres.

Disseny d'un informe mestre-detall

En un informe mestre-detall es mostra informació de dues entitats entre les quals hi ha establerta una relació 1:N.

En les empreses hi ha una gran quantitat d'informes mestre-detall: comanda amb les seves línies, client amb les seves comandes, producte amb les seves compres/vendes...

Les eines de *reporting* poden facilitar dos mecanismes de disseny per aconseguir informes mestre-detall:

- Elaborar un únic informe en el qual s'agrupa la informació segons el criteri que defineix la zona mestra.
- Elaborar informes diferenciats entre la part mestra i la part detall i incloure l'informe de la part detall com a un subinforme de la part mestra. En aquest cas, entre els informes cal poder definir una relació que fa de lligam entre la informació de l'informe mestre i la informació de l'informe detall.

Trobareu una possible solució a la primera forma d'obtenir l'informe, en el fitxer professor_with_courses_PSQL_Sencer.jrxml dins l'apartat "Mòduls OpenERP i informes" dels annexos del web. També hi trobareu el logo LogoIOC.png.

Disseny d'un informe mestre-detall de professors amb els cursos que imparteixen, pel mòdul school d'OpenERP

Volem dissenyar un informe mestre-detall de professors amb els cursos que imparteixen, pel mòdul `school` d'OpenERP, amb les següents característiques:

- Cada professor ha de començar en una nova pàgina i han d'aparèixer ordenats per nom.
- Per a cada professor interessa visualitzar la seva informació personal i la relació de cursos que imparteix.
- Informació personal d'un professor: nom, contracte, nombre d'hores que treballa setmanalment, telèfon, correu electrònic i adreça (carrer, codi postal i ciutat).
- Relació de cursos que imparteix: nom, descripció i nombre d'hores.
- Logotip a la capçalera de cada pàgina.
- Data i hora d'execució del llistat a la part inferior esquerra de cada pàgina.

L'informe demanat es pot plantejar de dues formes: una, amb un únic informe que contingui una sentència SQL que proporcioni les files resultants d'un `join` entre les taules `school_professor` i `school_course`, de manera que l'informe incorpori la definició d'un

grup per professor; l'altra, amb un informe encarregat de mostrar els professors, que incorpori un subinforme amb els cursos del professor.

En la solució basada en la sentència `join` entre les taules `school_professor` i `school_course`, observeu que el camp que defineix el grup ha de ser `id` de professor i no pot ser el nom (`name`), ja que hi podria haver diversos professors amb el mateix nom. A més, és imprescindible que la consulta contingui `order by name, id`, ja que iReport es basa en el canvi de valor pel camp d'agrupament (`id`) per iniciar un nou grup i, en conseqüència, cal garantir que el SGBD faciliti els registres a iReport agrupats per `id`.

Per incorporar un grup en un informe JasperReports, ens situem en el node principal del report (Report Inspector), premem el botó secundari del ratolí i en el menú contextual seleccionem l'opció *Add Report Group* i indiquem qualsevol dels camps que han d'anar en el grup. Això provoca que apareguin dues bandes (capçalera i peu) pel grup. Hi ubiquem els camps adequats. Si interessa, en les propietats del grup indiquem que comenci en una nova pàgina.

En la solució basada en un informe amb subinforme, fixeu-vos que l'informe detall ha d'incorporar, en la clàusula `WHERE` de la sentència SQL, el paràmetre `$P{Professor}` de tipus `integer`, creat a la zona *Parameters* de *Report Inspector*. L'informe mestre, en la propietat *Parameters* del full *Properties*, de l'objecte *Subreport*, ha d'emplenar el paràmetre *Professor* amb el contingut del camp `$F{id}`, aconseguint que en l'execució, per a cada professor, l'informe mestre invoqui l'informe detall amb el valor `id` del corresponent professor. Fixeu-vos, també, que el subinforme ha de tenir la propietat *Run to bottom* activada, per aconseguir que el subinforme ocupi tota la resta de la pàgina i el següent professor comenci en una nova pàgina.

Trobareu una possible solució a la segona forma d'obtenir l'informe, en els fitxers professor_with_courses_PSQL_Mestre.jrxml i professor_with_courses_PSQL_Detall.jrxml dins l'apartat "Mòduls OpenERP i informes" dels annexos del web. També hi trobareu el logo LogoIOC.png.

Pujada d'informes a un JasperReport Server Community

L'eina iReport Designer, a banda de permetre'ns el disseny de tot tipus d'informes, ens en permet l'execució. Però en una organització amb molts usuaris cal poder ubicar els diversos informes en un servidor d'informes i JasperReport Server Community és una bona opció.

Pujada d'informes generats amb iReport Designer a JasperReport Server Community

Ens interessa pujar els informes de professors de l'empresa Empresa_IOC d'OpenERP, generats amb iReport Designer, a un servidor JasperReport Community, per ser executats des d'un navegador per qualsevol usuari que hi tingui accés.

La pujada d'informes a un JasperReports Server es pot efectuar des de la interfície web que proporciona aquest servidor, però també des de l'eina iReport Designer. Ja que hi estem treballant, ho farem des d'aquesta eina.

En primer lloc establim connexió amb el servidor JasperReports a través de *Finestra* | *JasperReports Server Repository* creant, si encara no ho hem fet, una connexió amb el botó *Add new server*, que ens obre una pantalla per donar d'alta el servidor, introduint:

- *ID*: nom que ens permeti identificar el servidor, ja que podem tenir connexions establertes amb diversos servidors.
- *JasperReports Server URL*: URL adequada segons la plantilla `http://maquina:port/jasperserver/services/repository`, en la qual caldria substituir `jasperserver` per `jasperserver-pro` en cas d'un JasperReports Server Enterprise.
- *Username* - *Password*: usuari amb privilegis adequats en el servidor (`jasperadmin`, per exemple)

Si la connexió s'estableix sense problema, veurem que ens mostra la mateixa informació que visualitzem des de l'apartat *View*|*Repository* de la interfície web. Quan ens situem

damunt del nom del servidor i, premem el botó secundari del ratolí, ens apareix un menú contextual amb diverses possibilitats, entre les quals ens interessa l'opció *Add*.

L'opció *Add* apareix en els menús contextuals de gairebé tots els apartats que conté el repositori i l'executarem allà on ens interessi afegir elements. En la creació de qualsevol element, cal assignar-li un identificador (alfanumèric sense espais ni caràcters especials), un nom i, de forma opcional, una descripció. El nom i la descripció podran canviar en qualsevol moment, però l'identificador restarà immodificable per sempre més.

Per ser organitzats, el primer que farem és crear una nova carpeta a l'arrel del repositori, de nom *Empresa IOC* en el qual ubicarem tots els informes.

En segon lloc, crearem un *Datasource* que apunti a la base de dades PostgreSQL d'OpenERP en la qual ha d'anar el motor JasperReports a cercar les dades per confeccionar els informes. Sembla lògic ubicar aquest *Datasource* sota el node *Data Sources* del repository, tot i que també el podríem crear dins la carpeta *Empresa IOC* creada anteriorment. En crear el nou *Datasource* li hem de donar un nom entenedor (*Empresa IOC*) i cal emplenar adequadament el contingut de la pestanya *Data Source Details*, en la qual disposem d'un botó *Import from iReport* per capturar les dades del *Datasource* que segurament tenim definit en l'iReport Designer des del qual hem dissenyat els informes.

Per últim, situats a la carpeta *Empresa IOC*, procedim a afegir els tres informes, un a un, assignant-los un ID i un nom. El procés d'agregació d'un informe detecta altres recursos que utilitza l'informe (imatges i subinformes, per exemple) i en proposa la càrrega a partir del sistema de fitxers local, pujant-ne una còpia al servidor JasperReports.

Una vegada pujats els informes, ja són accessibles i executables des de la interfície web.

Incorporació de filtres a un informe

En moltes ocasions, els informes dissenyats han de mostrar únicament una part de la informació que poden mostrar i, per aconseguir-ho, cal afegir filtres en els quals l'usuari pugui introduir les dades de filtratge just abans de l'execució de l'informe. Les dades introduïdes per l'usuari en els filtres, s'afegeixen de manera adequada a la clàusula `WHERE` de la sentència SQL en la qual es basa l'informe.

La majoria d'entorns de *reporting* actuals faciliten filtres que permeten:

- Controlar el tipus de dada a introduir per l'usuari.
- Acompanyar el filtre d'un giny adequat al tipus de dada.
- Introduir els valors possibles a introduïr per l'usuari (atòmics o llistes, fixos o dinàmics, a partir del resultat d'una consulta SQL sobre la base de dades)

Exemple d'informe amb filtres

Volem retocar qualsevol dels dos informes de professors amb els seus cursos, de manera que en executar-los aparegui una llista multiselecció que mostri els diversos professors i permeti seleccionar els professors dels quals es vol generar l'informe.

Editem `professors_with_courses_PSQL_Sencer.jrxml`, hi creem un paràmetre (`pProfessors`) de tipus `Integer` i modifiquem la sentència SQL afegint la clàusula:

```
1 WHERE school_professor.id = $P{pProfessors}
```

En comprovar l'execució de l'informe des d'iReport Designer, apareix una pantalla per tal que introduïm un valor pel paràmetre `pProfessors`. Si volem veure l'informe d'algun professor, hem d'introduir el seu identificador dins la taula `school_professor` de la base de dades, que podem reconèixer amb qualsevol eina que ens permeti connectar amb un SGBD PostgreSQL (consola psql o eina pgAdmin, per exemple).

Una vegada hem comprovat la correcta execució des d'iReport Designer, volem pujar-lo a un servidor JasperReports en el qual, evidentment, no podem pretendre que els usuaris sàpiguen l'identificador del professor de qui vol obtenir l'informe i, en conseqüència, haurem de definir un *Input control* en el servidor, de tipus llista desplegable, que mostri els profesors de la base de dades.

Així doncs, pugem l'informe al servidor, de la manera habitual i, posteriorment, li afegim un *Input control* de tipus *Single Select Query*, que s'ha d'anomenar igual que el paràmetre (`pProfessors`) de l'informe.

El servidor JasperReport permet definir *Input controls* que puguin ser utilitzats en diversos informes. D'aquesta manera, pot ser adequat disposar d'una llista desplegable amb els professors de la base de dades. Per tal que l'usuari pugui escollir-ne un, podríem decidir crear un *Input control* genèric; del contrari, la lògica ens porta a crear-lo localment en l'informe que el necessiti.

La definició d'un *Input control* (genèric o local) obliga a indicar el tipus de control (hi ha diverses possibilitats) i les característiques d'obligatorietat, només lectura i visibilitat del control. A més, hi ha altres característiques per definir, en funció del tipus de control.

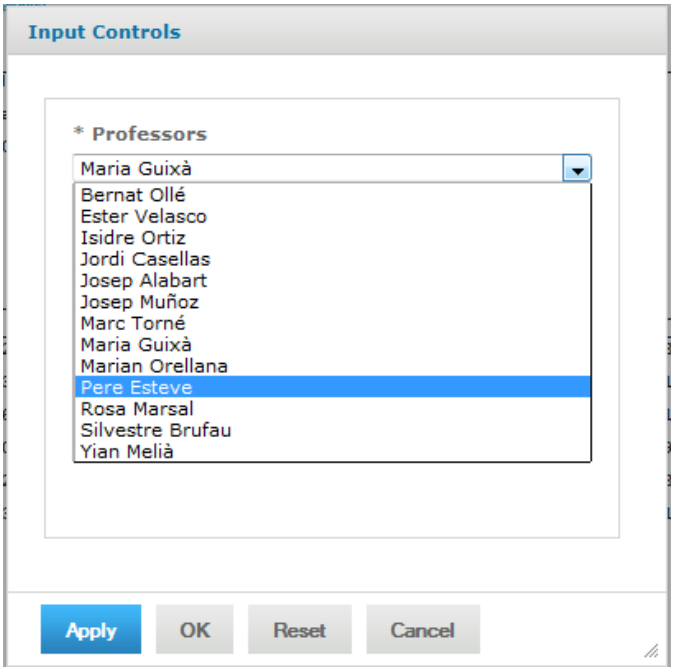
En un control *Single Select Query*, igual que en el nostre exemple, cal definir la *Query Resource* (que també pot ser genèrica o local al control), les columnes visibles per a l'usuari i la columna que cal tenir en compte per emplenar el paràmetre de l'informe. En el nostre exemple, la *Query Resource* és una consulta similar a:

```
1 select id, name from school_professor order by name''
```

on la columna a visualitzar és `name` i la columna de la qual agafar el valor és `id`.

Una vegada definit l'*Input control*, ja s'està en condicions d'executar l'informe des de la interfície web, tal com mostra la figura 2.2.

Figura 2.2. Filtre de tipus llista desplegable per a un informe



2.2 Reporting en OpenERP

En la versió 6.1 d'OpenERP disposem de dos mecanismes per integrar informes en els mòduls, cadascun d'ells basat en un motor de *reporting* específic:

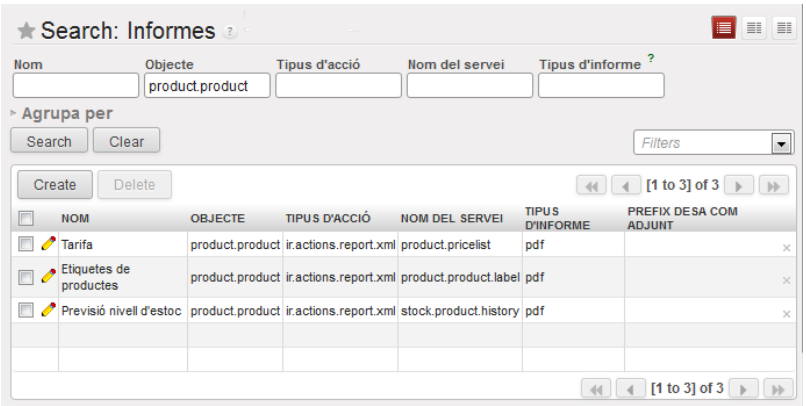
Podeu ampliar la informació sobre el motor d'informes RLTK de ReportLab a www.reportlab.com, sobre el motor d'informes JasperReport de Jaspersoft Corporation a community.jaspersoft.com/project/jasperreports-library) i sobre NaN·tic a www.nan-tic.com/ca/.

- Motor d'informes RLTK de ReportLab (sota llicència BSD –codi obert–) que permet generar informes en formats PDF. Forma part d'OpenERP com a motor oficial de reporting. RLTK (ReportLab Toolkit), basat en Python, genera informes a partir de plantilles codificades amb el llenguatge RML (*Report Markup Language*), que és un dialecte d'XML. Per evitar haver de conèixer el llenguatge RML, OpenERP facilita un connector per a OpenOffice o LibreOffice, que fa possible dissenyar l'informe des del processador de textos Writer i obtenir el fitxer RML corresponent.

- Motor d'informes JasperReport de Jaspersoft Corporation, que permet generar informes en múltiples formats (PDF, RTF, XML, XLS, CSV, HTML, XHTML, text, DOCX o OpenOffice). Aquesta opció de reporting no és oficial per OpenERP 6.1 i cal instal·lar el mòdul no oficial Jasper Reports desenvolupat inicialment per NaN·tic, i recentment constituït en el projecte Jasper Reports for OpenERP (https://launchpad.net/openobject-jasper-reports) dins Launchpad. El disseny d'informes JRXML (format pel motor JasperReports) per a OpenERP s'efectua amb l'eina iReport Designer.

La majoria de mòduls d'OpenERP van acompanyats de llistats. Així, si tenim el mòdul de *Magatzem* instal·lat i naveguem a *Magatzem* | *Productes* | *Productes*, podem observar que OpenERP ens facilita tres llistats: *Llista de preus*, *Etiquetes de productes* i *Previsió nivell d'estoc*. Els informes vinculats a un objecte d'OpenERP (`product.product`, per exemple) són accessibles, de forma automàtica, en els formularis vinculats a l'objecte (opció *Magatzem* | *Productes* | *Productes*, per exemple). En el client web apareixen en el panell lateral dret del navegador i en el client GTK apareixen en prémer el botó *Imprimir* de la botonera superior.

Figura 2.3. Pantalla que mostra els informes existents sobre un objecte d'OpenERP



L'opció *Setings* | *Personalització* | *Objectes de baix nivell* | *Accions* | *Informes* permet gestionar els informes instal·lats a l'empresa activa. Si fem una cerca dels informes existents sobre l'objecte `product.product`, veurem que n'hi ha tres (figura 2.3), com era d'esperar. Hi observem que els tres informes són de tipus PDF. La columna *Nom del servei* fa referència al nom de fitxer PDF que genera OpenERP. Aquesta pantalla permet afegir nous informes, eliminar informes existents i modificar paràmetres d'execució dels informes existents.

2.2.1 Disseny d'informes RML a través d'OpenOffice/LibreOffice

El disseny d'informes per OpenERP amb OpenOffice o LibreOffice és una manera ràpida de dissenyar informes que es poden integrar dins OpenERP i que pot dur a terme qualsevol usuari amb uns mínims coneixements de com està estructurada la informació a la base de dades d'OpenERP.

Per dissenyar informes per OpenERP amb OpenOffice o LibreOffice, necessiteu instal·lar a l'empresa d'OpenERP per la que vulgueu dissenyar els informes, el mòdul *OpenOffice Report Designer* que ja incorpora la instal·lació d'OpenERP. La instal·lació d'aquest mòdul és molt ràpida i de seguida mostra una pantalla informativa sobre els passos que s'han de seguir, en OpenOffice o LibreOffice, per instal·lar l'extensió que permet dissenyar els informes i se'n facilita la descàrrega des de la mateixa pantalla.

L'OpenOffice o LibreOffice no té perquè estar instal·lat en la màquina que conté el servidor OpenERP; només es necessita tenir-lo a les màquines des de les quals es vol dissenyar informes. Aquestes màquines han de tenir connectivitat amb el servidor OpenERP.

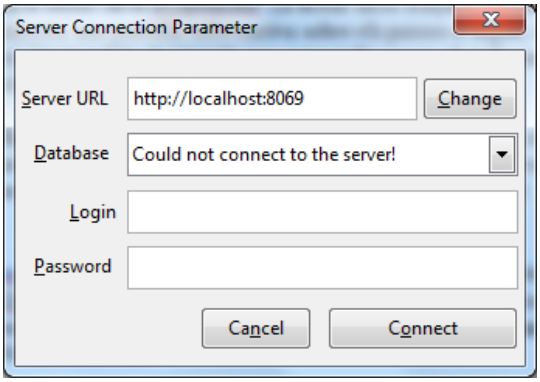
Instal·leu l'extensió openerp_report_designer.zip facilitada en el procés d'instal·lació del mòdul *OpenOffice Report Designer* en l'OpenOffice o LibreOffice que vulgueu utilitzar. L'extensió funciona en la majoria de versions d'aquests programaris. Després de reiniciar el Writer, observeu:

- La barra de menús conté la nova opció *Open ERP Report* amb un conjunt d'opcions que ens permeten dissenyar els informes.
- Hi ha la nova barra d'eines *Open ERP Report*. Si en algun moment tanqueu aquesta barra i la voleu tornar a obrir, en l'opció de barres d'eines de Writer no la trobareu amb el nom *Open ERP Report*, sinó amb el nom *Complement* seguit d'un número.

Per iniciar el disseny d'un informe, ja sigui nou o, fins i tot, d'un informe instal·lat en el servidor, el primer que cal fer és establir connexió amb el servidor, fet que podem assolir amb el primer botó de la barra d'eines o amb l'opció *Open ERP Report* | *Server parameters*. Apareix la pantalla de connexió (figura 2.4) que intenta connectar amb un servidor a la mateixa màquina. Si el troba, no apareix el

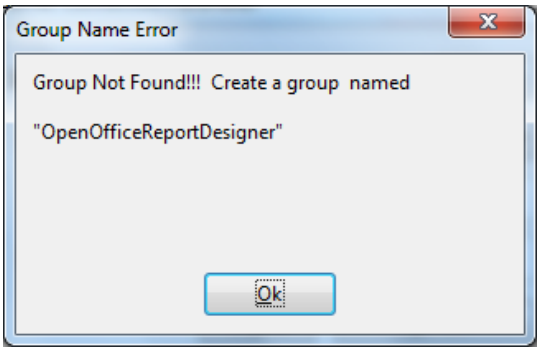
missatge *Could not connect to the server!*, sinó que la llista desplegable de l'apartat *Database* mostra les bases de dades existents en el servidor.

Figura 2.4. Pantalla de connexió des de Writer cap a un servidor OpenERP



Una vegada introduït el valor adequat a l'apartat *Server URL*, s'estableix connexió amb el servidor OpenERP i se selecciona la base de dades que correspongui, indicant un usuari i contrasenya adequats per l'empresa. Si l'empresa seleccionada no ha tingut mai una connexió des de Writer, apareixerà l'error de la figura 2.5, que ens indica que a l'empresa ha d'existir el grup d'usuaris `OpenOfficeReportDesigner`, al qual hauran de pertànyer els usuaris de l'empresa amb permís per dissenyar informes.

Figura 2.5. Error si l'empresa d'OpenERP encara no ha estat mai accedida des de Writer



Creem, doncs, el grup d'usuaris `OpenOfficeReportDesigner` i hi assignem els usuaris que utilitzarem per dissenyar informes (`admin`, per exemple). En cas de tenir el grup creat i intentar l'accés amb un usuari no pertanyent al grup, apareixeria l'error *You have not access these Report Designer*.

Una vegada tenim establerta una connexió, podem optar per crear un nou informe (*Open ERP Report* | *Open a new report*) o modificar un informe del servidor (*Open ERP Report* | *Modify Existing Report*). En escollir l'opció de modificació, al cap d'uns segons, ens apareix una pantalla amb tots els informes RML existents en el servidor i ens permet seleccionar-ne qualsevol, per modificar-lo i, posteriorment, tornar-lo a enviar al servidor. Abans, però, de modificar-ne cap, ens convé conèixer l'estructura dels informes RML generats amb Writer i, n'aprendrem tot creant nous informes.

Disseny d'un informe de professors pel mòdul school d'OpenERP

Volem dissenyar un informe dels professors pel mòdul `school` d'OpenERP, amb les següents característiques:

- Columnes: nom, contracte, nombre d'hores que treballa setmanalment, telèfon, correu electrònic i adreça (carrer, codi postal i ciutat).
- Imatge corporativa de l'empresa a totes les pàgines.
- Data i hora d'execució del llistat a la part superior dreta.

Per aconseguir-ho, creem un nou informe amb l'opció *Open ERP Report* | *Open a new report* i ens apareix una llista dels objectes existents a l'empresa i seleccionem l'objecte `school.professor` (al capdavall de la llista). Amb aquesta acció hem indicat que l'informe està vinculat a l'objecte `school.professor`, però no apareix res en pantalla. Aquest vincle queda a nivell intern i el podem veure a *Fitxer* | *Propietats* | *Propietats personalitzades* | *Informació 4*. Observeu, també, en les propietats personalitzades, el servidor OpenERP al que us connecteu (*Informació 1*) i l'usuari (*Informació 2*).

L'acció següent és definir un bucle (opció *Open ERP Report* | *Add a loop*) ja que tot informe es basa en un bucle sobre el recursos de l'objecte que l'usuari seleccionarà en el moment d'executar l'informe. En executar aquesta opció, ens apareix una pantalla en la qual hem de seleccionar l'objecte sobre el qual es basa el bucle (`school.professor`) i el camp sobre el qual iterarà el bucle, que en aquest cas, és `objects` (no se'ns facilita altra possibilitat), com mostra la part esquerra de la figura 2.6. Una vegada escollit `objects`, en el document en el qual estem dissenyant l'informe apareix el codi:

```
1 |-.professor.-|
```

El text anterior indica un bucle i es correspon a una instrucció de RML. Les instruccions RML s'escriuen entre dobles claudàtors i el connector instal·lat en Writer ens permet veure la visualització amb claudàtors tot seleccionant el corresponent botó de la barra d'eines o amb l'opció *Open ERP Report* | *Conversions Fields -> Brackets*. Si ho fem, veurem com el codi anterior canvia per:

```
1 [[repeatIn(objects,'professor')]]
```

La instrucció anterior indica que s'executarà un procés iteratiu sobre tots els recursos de l'objecte `school.professor` (camp *Informació 4* de les propietats personalitzades del fitxer ODT) i la paraula `professor` és un àlies per referir-nos a cada recurs. Podem, si voleu, posar un nom més curt, com `prof` o `pr`.

A continuació hem de seleccionar els camps que ens interessa obtenir a cada iteració, i això ho fem amb l'opció *Open ERP Report* | *Add a field* que ha de presentar-nos una pantalla com la part dreta de la figura 2.6, en la qual es facilita la llista dels camps accessibles des de l'objecte `school.professor` (triga a facilitar-se, cal tenir paciència) i com que aquest objecte conté camps `many2one`, OpenERP facilita els camps accessibles a través dels objectes referenciats pels camps `many2one`, fet que ens va molt bé per poder seleccionar correu electrònic i adreça. Seleccionem els camps:

```
1 [[ professor.name ]]
2 [[ professor.contract ]]
3 [[ professor.hours_available ]]
4 [[ professor.phone ]]
5 [[ professor.address_id.email ]]
6 [[ professor.address_id.street ]]
7 [[ professor.address_id.zip ]]
8 [[ professor.address_id.city ]]
```

La selecció dels camps no és obligatori fer-la a través d'*Open ERP Report* | *Add a field* i podem escriure'ls directament tenint en compte la sintaxi a utilitzar.

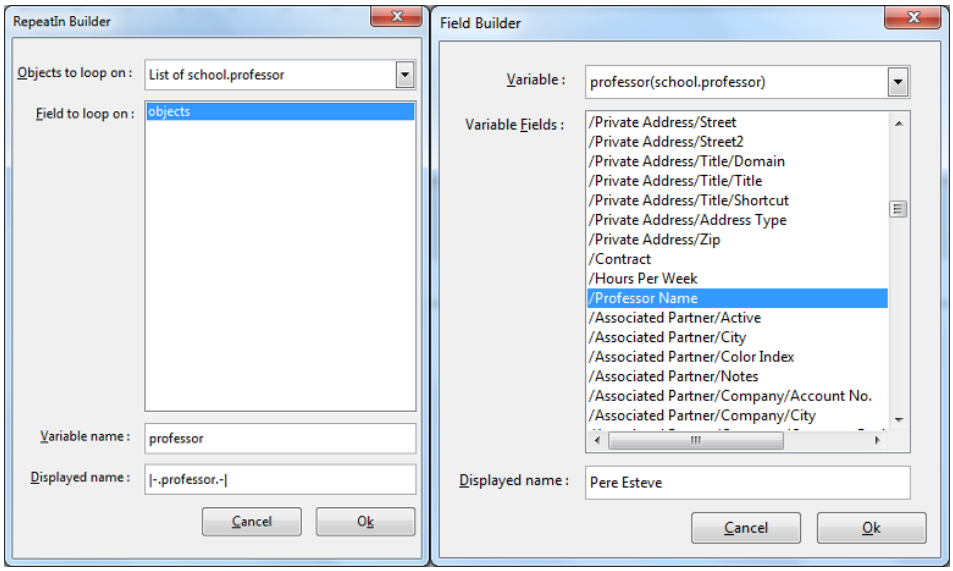
Per ubicar correctament els camps en columnes d'un llistat, ens cal crear una taula de 8 columnes i ubicar cada camp en una columna. El camp corresponent al bucle

(`|-.professor.-|`) també ha d'estar situar dins la taula; si el deixem fora, l'informe mostrarà cada recurs en una pàgina diferent.

En la solució facilitada, observeu el camp `[[time.ctime()]]` que mostra la data i hora del sistema en el moment d'execució de l'informe.

Trobareu una possible solució en el fitxer professors.odt dins l'apartat "Mòduls OpenERP i informes" dels annexos del web.

Figura 2.6. Pantalles de Writer per seleccionar l'objecte sobre el qual s'itera i els camps a visualitzar



Per les proves realitzades, sembla que la pantalla *Field Builder* (figura 2.6) només s'emplena si no es té activada la visualització amb claudàtors.

Una vegada l'informe està dissenyat, podem enviar-lo al servidor, amb l'opció *Open ERP Report* | *Send to the server* i comprovar-ne l'execució. En el moment d'enviar-lo, apareixerà una pantalla en la qual es demana:

- *Report name*: per introduir el nom amb el qual l'informe apareixerà en els clients web i GTK.
- *Technical name*: corresponent al nom que OpenERP proposarà pel fitxer PDF que genera la impressió. Writer proposa un nom format pel nom de l'objecte (`school.professor`) concatenat amb una tira alfanumèrica aleatòria.
- *Corporate Header*: per activar la incorporació automàtica de la imatge corporativa de l'empresa (capçalera i peu definits per a l'empresa).
- *Select Rpt. Type*: per seleccionar la sortida del fitxer, que pot ser PDF, ODT o HTML (normalment PDF).

Després d'enviar l'informe al servidor, podrem comprovar-ne l'aparició en els clients web i GTK al situar-nos en el formulari *Professors*. També podrem observar-ne l'existència a *Settings* | *Personalització* | *Objectes de baix nivell* | *Accions* | *Informes*.

L'informe, una vegada enviat al servidor, hi queda registrat amb un identificador (taula `ir_act_report_xml`), que també queda enregistrat a l'apartat *Informació 3* de les propietats personalitzades del fitxer ODT, de manera que cada vegada que s'intenta enviar el fitxer al servidor OpenERP, si el fitxer ja té un identificador introduït (d'anteriors enviaments), s'intenta substituir el formulari existent amb

igual identificador. Això suposa un problema si un informe (amb un identificador) es vol enviar a una altra empresa; caldrà eliminar prèviament el contingut del camp *Informació 3* de les propietats personalitzades. De la mateixa manera, en recuperar un informe d'una empresa d'un servidor OpenERP, el camp *Informació 3* queda documentat amb l'identificador de l'informe en l'empresa origen.

Trobareu una possible solució en el fitxer professor_with_courses.odt dins l'apartat "Mòduls OpenERP i informes" dels annexos del web.

Disseny d'un informe mestre-detall de professors amb els cursos que imparteixen, pel mòdul school d'OpenERP

Volem dissenyar un informe mestre-detall de professors amb els cursos que imparteixen, pel mòdul `school` d'OpenERP, amb les següents característiques:

- Cada professor ha de començar en una nova pàgina.
- Per a cada professor interessa visualitzar la seva informació personal i la relació de cursos que imparteix.
- Informació personal d'un professor: nom, contracte, nombre d'hores que treballa setmanalment, telèfon, correu electrònic i adreça (carrer, codi postal i ciutat).
- Relació de cursos que imparteix: nom, descripció i nombre d'hores.
- Imatge corporativa de l'empresa a cada pàgina.

L'informe demanat es basa en l'objecte `school.professor`.

Cada professor (bucle `[[repeatIn(objects,'professor')]]`) es mostra en una nova pàgina amb distribució capçalera-línies.

La capçalera incorpora les dades personals del professor:

```
[[ professor.name ]]
[[ professor.contract ]]
[[ professor.hours_available ]]
[[ professor.phone ]]
[[ professor.address_id.email ]]
[[ professor.address_id.street ]]
[[ professor.address_id.zip ]]
[[ professor.address_id.city ]]
```

La zona de línies conté la informació dels cursos que imparteix el professor, basat en un bucle sobre els recursos accessibles des del camp `professor.course_ids`. Podem introduir aquest bucle escrivint `[[repeatIn(professor.course_ids,'course_ids')]]` o executant *Open ERP Report* | *Add a loop*, fet que ens mostra tots els camps `many2one` i `many2many` de l'objecte `school.professor`, camps susceptibles de proporcionar nous bucles. La línia prèvia a la taula que conté la definició del bucle està formatada amb lletra de mida 2 (mínim possible en Writer), per tal que ocupi el mínim d'espai possible en l'informe.

La definició del bucle `course_ids` ha d'estar abans que la taula que conté els camps accedits a través de `course_ids`, i ambdós (bucle i taula) cal ubicar-los en una secció. En l'exemple que es proposa, s'ha creat una secció anomenada *Courses* (el nom pot ser qualsevol, però millor introduir un nom amb significat adequat).

El bucle a través del camp `professor.course_ids` ens permet incorporar les dades dels cursos en una taula amb tres columnes:

```
[[ course_ids and course_ids.name]]
[[ course_ids and course_ids.subject]]
[[ course_ids and course_ids.hours_total]]
```

OpenERP ens facilita, doncs, a través de Writer, la possibilitat de generar i instal·lar informes en una empresa. Els informes instal·lats, però, no formen

part de cap mòdul i és molt possible que els desenvolupadors de mòduls vulguin dissenyar informes que s'instal·lin a l'empresa quan s'instal·la el mòdul. En aquest cas, els informes cal dissenyar-los en llengua anglesa, com la resta del mòdul, ja que el mecanisme de traducció d'OpenERP també té en compte els textos dels informes RML.

Per incorporar un informe dissenyat amb Writer en un mòdul d'OpenERP, cal seguir els següents passos:

- Obtenir el fitxer RML corresponent, executant l'opció *Open ERP Report | Export to RML* des de Writer.
- Ubicar el fitxer RML en una carpeta `report` (recomanable, no obligatori). També hi ha el costum de deixar-hi el document ODT a partir del qual s'ha generat el fitxer RML.
- Dissenyar un fitxer XML que defineix un o diversos informes per al mòdul. Aquest fitxer es pot ubicar també dins la carpeta `report`. Ha de ser referenciat des de l'apartat `update_xml` del fitxer __openerp__.py del mòdul. El contingut d'aquest fitxer XML ha de ser com es mostra tot seguit, amb un element `report` per a cada informe:

```
<?xml version="1.0" encoding="utf-8"?>
<openerp>
  <data>
    <report
      id="identificadorDeInformeDinsElModul"
      model="objecteOpenErpEnElQueEsBasaElInforme"
      name="nomDelServei"
      rml="nomFitxerRMLambElCamiOnEsTroba"
        <!-- El camí ha d'incloure el nom de la
          carpeta del mòdul-->
      header="True/False"
      string="nomQueEsVisualitzaEnOpenERP"/>
  </data>
</openerp>
```

- Instal·lar el mòdul amb el(s) nou(s) informe(s) i actualitzar les traduccions als diversos idiomes amb els textos que incorporen els informes, per finalment disposar del mòdul amb els informes traduïts.

Podeu trobar la versió 17 del mòdul `school`, que incorpora els informes professors.rml i professor_with_courses.rml, amb la traducció al català, a l'arxiu school_17.zip, dins l'apartat "Mòduls OpenERP i informes" dels annexos del web.

En moltes ocasions, s'ha d'incloure en l'informe camps calculats que no estan definits en el model sobre el qual es basa. En aquests casos, la solució resideix en definir una classe Python que derivi de `report_sxw.rml_parse` en la qual s'incorporen els mètodes que efectuen els càlculs adequats. La nova classe, com és habitual, ha de residir en un fitxer `.py` i s'acostuma a ubicar en la carpeta `report`, fet que provoca la necessitat d'incloure un fitxer __init__.py dins la carpeta `report`, amb el contingut `import nomFitxer`, i d'incorporar la instrucció `import report` en el fitxer __init__.py del mòdul. La versió 17 del mòdul `school` no incorpora aquest muntatge ja que en els informes elaborats no ha estat necessari incorporar cap càlcul.

No hi ha molta documentació que faci referència al disseny d'informes RML a través d'OpenOffice/LibreOffice; en cas de necessitar-la, haureu de cercar exemples en les carpetes `report` dels mòduls que OpenERP incorpora.

2.2.2 Disseny d'informes JRXML a través d'iReport Designer

L'empresa NaN·tic (www.nan-tic.com/ca/) va desenvolupar un mòdul per a OpenERP que incorpora el motor de JasperReports i fa possible l'execució i la integració d'informes JRXML, generats normalment per l'eina de disseny iReport Designer. En el moment actual el mòdul es troba en el projecte Jasper Reports for OpenERP (https://launchpad.net/openobject-jasper-reports) dins Launchpad.

En primer lloc cal descarregar de Launchpad el mòdul `jasper_reports` per a la versió d'OpenERP que correspongui. En el nostre cas, segons les instruccions existents a Launchpad:

```
1 bzr branch lp:openobject-jasper-reports/6.1
```

El resultat de la descàrrega és la carpeta `jasper_reports` que correspon al mòdul Jasper Reports. S'instal·la de manera habitual.

En cas de no tenir un entorn JRE instal·lat, en el moment d'executar un informe Jasper, apareix el següent error:

La utilització del motor JasperReports en OpenERP necessita que la màquina en la qual resideixi el servidor OpenERP tingui instal·lat un entorn JRE de Java.

Els passos per dissenyar un informe JasperReport per a OpenERP són:

1. Executar l'opció *Settings* | *Personalització* | *Jasper Reports* | *Crea una plantilla de dades* per obtenir un arxiu XML amb la definició dels camps de l'objecte sobre el qual es dissenya l'informe i dels objectes per ell referenciats a través de camps relacionals. OpenERP mostra una pantalla en la qual hem d'emplenar el camp *Model* amb l'objecte sobre el qual es vol dissenyar l'informe i el camp *Depth* amb el nombre de nivells de camps relacionals que cal incloure en la plantilla XML. Amb els dos camps emplenats, premem el botó *Create* i OpenERP ens genera l'arxiu template.xml que enregistrem per utilitzar-lo en el següent pas.

2. En l'eina iReport Designer creem un *Datasource* de tipus *XML file datasource* i emplenem les dades tal com mostra la figura 2.7.

3. Creem un informe des d'iReport Designer i emplenem la *Report query* tal com mostra la figura 2.8, arrossegant els camps que corresponguin del panell dret cap al panell inferior, com indica la fletxa.

4. Dissenyem el contingut de l'informe de la manera habitual.

L'informe dissenyat no es pot executar des d'iReport Designer i caldrà integrar-lo en OpenERP per a comprovar-ne l'execució. Abans d'instal·lar-lo cal efectuar un retoc manual a l'arxiu JRXML generat per iReport Designer, ja que del contrari, en executar l'informe apareixeria l'error de la figura 2.9, provocat per una incompatibilitat entre el mòdul Jasper Reports i els formats JRXML generats per iReport Designer de versió posterior a 4.5. El retoc consisteix en editar el fitxer JRXML amb un editor de textos que sàpiga cercar expressions regulars

(per exemple, Notepad++) i eliminar totes les aparicions de l'expressió regular `uuid="\w*-\w*-\w*-\w*-\w*"`.

Figura 2.7. Datasource de tipus XML per crear informes JRXML per OpenERP

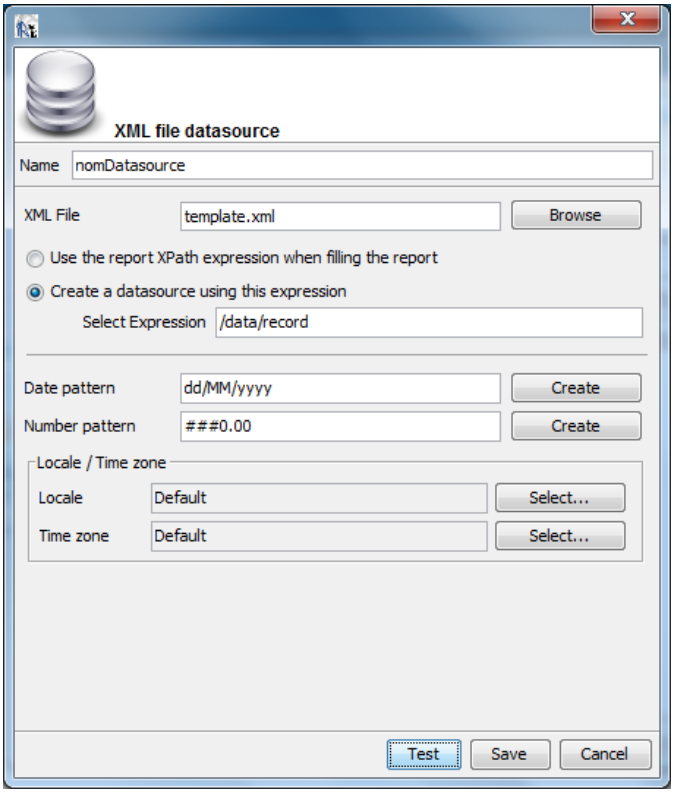


Figura 2.8. Generació de camps per un informe JRXML basat en dataxource XML

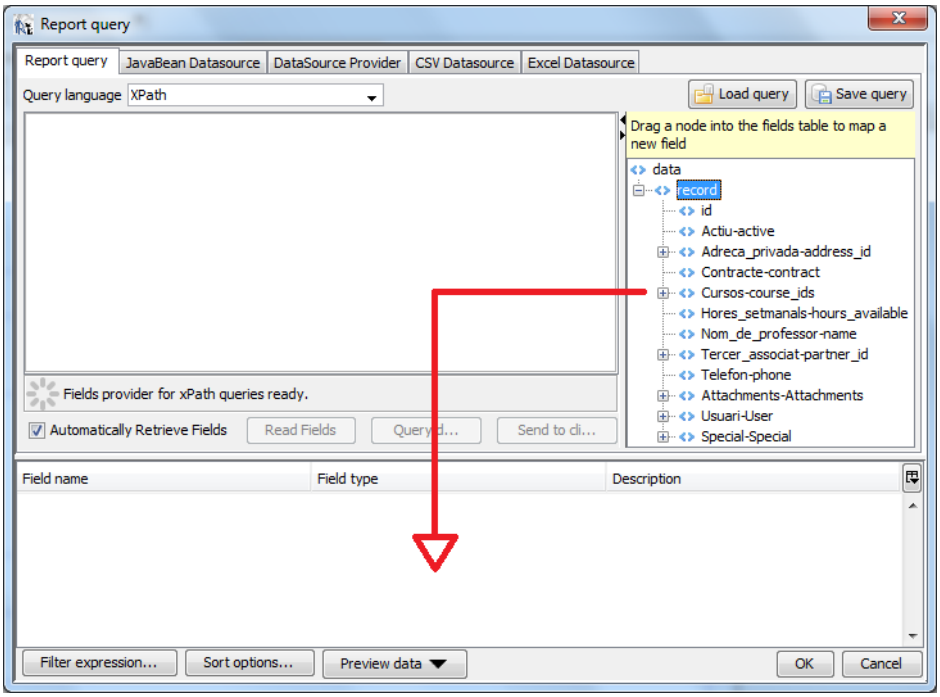
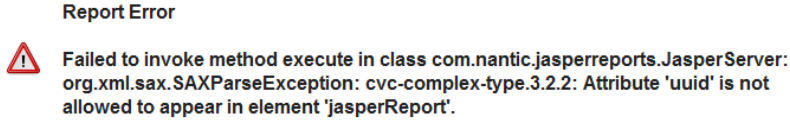


FIGURA 2.9. Error en executar informe JRXML generat amb iReport Designer en OpenERP



L'error de la figura 2.9 es pot evitar configurant iReport Designer per generar els fitxers JRXML en format 4.5 (opció *Eines* | *Opcions* | *General* | *Compatibilitat*).

Per integrar un informe JRXML en una empresa d'OpenERP, cal utilitzar l'opció *Settings* | *Personalització* | *Jasper Reports* | *Jasper Reports* i crear-hi un nou recurs, indicant:

- Nom: amb el qual l'informe apareixerà en els clients web i GTK.
- Model: amb el nom de l'objecte d'OpenERP en el qual es basa l'informe.
- Nom del servei: amb el nom que OpenERP ha de proposar pel fitxer que generi la impressió.
- Sortida Jasper: per seleccionar el format de l'informe (HTML, CSV, XLS, RTF, ODT, ODS, Text o PDF).

En el nou recurs cal afegir-hi els fitxers necessaris per executar l'informe: fitxer JRXML de l'informe, fitxers d'imatges inclosos a l'informe, altres fitxers JRXML incrustats dins l'informe principal... El fitxer principal ha de portar activada la casella *Per defecte*.

Una vegada enregistrat l'informe, podrem comprovar la seva aparició en els clients web i GTK al situar-nos en el formulari que correspon a l'objecte en el qual es basa l'informe. També podrem observar la seva existència a *Settings* | *Personalització* | *Objectes de baix nivell* | *Accions* | *Informes*.

Trobareu una possible solució en el fitxer professors.jrxml dins l'apartat "Mòduls OpenERP i informes" dels annexos del web. També hi trobareu la plantilla professors.xml utilitzada i el logo LogoIOC.png.

Disseny d'un informe de professors pel mòdul school d'OpenERP

Volem dissenyar un informe dels professors pel mòdul `school` d'OpenERP, amb les següents característiques:

- Columnes: nom, contracte, nombre d'hores que treballa setmanalment, telèfon, correu electrònic i adreça (carrer, codi postal i ciutat).
- Títol i logotip a la primera pàgina.
- Data i hora d'execució del llistat a la part superior esquerra de cada pàgina.
- Número de pàgina acompanyat del total de pàgines a la part superior dreta de cada pàgina.
- Títols de les columnes a cada pàgina.

La plantilla XML per obtenir l'informe demanat s'ha generat sobre l'objecte `school.professor` amb profunditat 2 ja que calia accedir a valors de l'objecte `res.partner.address` apuntat pel camp `address_id` de `school.professor`.

Trobareu una possible solució en el fitxer professor_with_courses .jrxml dins l'apartat "Mòduls OpenERP i informes" dels annexos del web. També hi trobareu la plantilla professors.xml utilitzada i el logo LogoIOC.png.

Per obtenir informes Jasper amb estructura mestre-detall, cal utilitzar la definició de grups, segons la forma habitual d'iReport Designer, i afegir la següent *property* al report (*node principal del report* | *Properties* | *Properties*):

```
Name: OPENERP_RELATIONS
Value: ['nomCamp_One2Many_o_Many2Many']
```

Disseny d'un informe mestre-detall de professors amb els cursos que imparteixen, pel mòdul school d'OpenERP

Volem dissenyar un informe mestre-detall de professors amb els cursos que imparteixen, pel mòdul `school` d'OpenERP, amb les característiques següents:

- Cada professor ha de començar en una nova pàgina i han d'aparèixer ordenats per nom.
- Per a cada professor interessa visualitzar la seva informació personal i la relació de cursos que imparteix.
- Informació personal d'un professor: nom, contracte, nombre d'hores que treballa setmanalment, telèfon, correu electrònic i adreça (carrer, codi postal i ciutat).
- Relació de cursos que imparteix: nom, descripció i nombre d'hores.
- Logotip a la capçalera de cada pàgina.
- Data i hora d'execució del llistat a la part inferior esquerra de cada pàgina.

La solució proposada es basa en la plantilla professors.xml generada sobre l'objecte `school.professor` amb nivell de profunditat 2, que conté la informació referent als cursos que imparteix cada professor. L'informe defineix un grup pel camp `id` de professor i conté la *property*:

```
Name: OPENERP_RELATIONS
Value: ['course_ids']
```

El mòdul Jasper Reports per OpenERP facilita la possibilitat d'instal·lar informes Jasper en una empresa. Els informes instal·lats, però, no formen part de cap mòdul i és molt possible que els desenvolupadors de mòduls vulguin dissenyar informes Jasper que s'instal·lin a l'empresa en instal·lar el mòdul. Per incorporar un informe Jasper en un mòdul d'OpenERP cal seguir el mateix procediment que pels informes RML. Els informes JasperReports incorporats en un mòdul no poden incloure un logo (ja que canvia per a cada organització) i hauríem de trobar la manera de poder utilitzar una imatge corporativa com en els informes RML.

El mòdul Jasper Reports per OpenERP també facilita la funcionalitat de traducció dels informes a diferents idiomes.

Podeu trobar la versió 18 del mòdul `school`, que incorpora els informes professors.jrxml i professor_with_courses .jrxml, sense logo, a l'arxiu school_18.zip, dins l'apartat "Mòduls OpenERP i informes" dels annexos del web.

Podeu trobar més informació sobre el mòdul Jasper Reports per OpenERP a: www.nan-tic.com/es/2009/jasperreports-ng/.

2.3 Solucions BI en OpenERP

OpenERP, a banda de les possibilitats de *reporting*, facilita dues funcionalitats que podem considerar en el camp del *Business Intelligence*:

- La definició d'objectes no persistents, basats en vistes de PostgreSQL.
- La definició de quadres de comandament.

2.3.1 Vistes basades en objectes no persistents

OpenObject permet dissenyar objectes no persistents, basats en vistes de PostgreSQL, que s'acostumen a utilitzar en el desenvolupament de gràfics, informes estadístics i quadres de comandament. Feu una ullada a la documentació oficial d'OpenObject.

Podeu consultar la documentació oficial d'OpenObject a doc.openerp.com/v6.0/developer/4_14_dashboard/index.html.

Podeu trobar la versió 19 del mòdul `school`, que incorpora la solució proposada, a l'arxiu school_19.zip, dins l'apartat "Mòduls OpenERP i informes" dels annexos del web.

Disseny de vista basada en un objecte no persistent

Volem aconseguir un gràfic que ens mostri el nombre de cursos que desenvolupa cada professor, en el mòdul `school`.

En principi seria lògic pensar que el gràfic s'hauria de poder dissenyar sobre el model `school.professor`, però això no és possible ja que no hi tenim el nombre de cursos que un professor desenvolupa.

Per altra banda, ens adonem que l'objecte `school.course` conté, per cada recurs, el professor assignat. Llavors, si el tipus de vista `graph` proporcionés un operador (per exemple, `count`) que permetés recomptar el nombre de registres d'un grup, podríem escriure quelcom similar a:

```
<record model="ir.ui.view"
    id="view_school_professor_graph">
  <field name="name">school.professor.graph</field>
  <field name="model">school.course</field>
  <field name="type">graph</field>
  <field name="arch" type="xml">
    <graph string="Number of courses by professor">
      <field name="prof_id"/>
      <field name="id" operator="count"/>
    </graph>
  </field>
  </record>
```

Però no és el cas. Potser en versions futures d'OpenObject hi haurà un operador `count`, de la mateixa manera que ara hi ha els operadors `min` i `max`.

Una possibilitat és retocar la classe `school.professor` per afegir-hi un nou camp `nre_cursos`, calculat a partir del camp `course_ids`. Pot ser un bon exercici, però no és la solució que donarem aquí.

Una altra possibilitat de solució és crear una classe no persistent, basada en una vista de PostgreSQL, que ens faciliti el càlcul desitjat, i desenvolupar la vista `graph` sobre la nova classe.

La solució incorpora la nova classe `school_professor_nbr` (dins el fitxer school.py) no persistent, basada en una vista PostgreSQL. Fixeu-vos que la sentència SQL està formada amb un `outerjoin` per aconseguir que els possibles professors sense cap curs apareguin a la vista. Fixeu-vos també, que incorpora un camp `id_name` que és la concatenació de l'`id` del professor amb el seu nom, per poder escollir aquest camp en la vista `graph`, ja que del contrari, si hi hagués dos professors amb el mateix nom, la vista `graph` els agruparia si prenguéssim el nom com a primer dels camps de la vista.

La solució també incorpora un parell de vistes (`tree` i `graph`) dins el fitxer school_view.xml i l'acció i opció de menú adequades per poder invocar les vistes. Si voleu, podeu incorporar-hi vistes `search` per facilitar les recerques.

Una vegada instal·lat el mòdul, si observeu la BD de PostgreSQL, podreu comprovar l'existència de la vista `school_professor_nbr`. Les classes no persistents es poden utilitzar en el disseny de vistes, informes (Writer o JasperReports) i quadres de comandament.

2.3.2 Quadres de comandament

OpenERP 6.1 facilita la possibilitat de dissenyar senzills quadres de comandament, ja sigui a nivell d'usuari, personalitzant el seu OpenERP, o a nivell d'incorporació en els mòduls per a la seva instal·lació.

Els quadres de comandament d'OpenERP 6.1 es confeccionen integrant qualsevol de les vistes existents a l'empresa. La versió 6.1 divideix la pantalla en dues zones (*left & right)* en les quals es pot anar ubicant diverses vistes:

- En el client web, queden situades, inicialment, una sota l'altra dins la zona a la qual han estat assignades. L'usuari les pot reubicar amb el mecanisme *drag & drop*, i disposa d'un botó *reset* per tornar a la ubicació original.
- En el client GTK, OpenERP les reubica una al costat de l'altra, dividint la pantalla en tantes columnes com vistes afegides.

De fet, no s'ubiquen vistes, sinó accions (que han d'existir) que invoquen una o diverses vistes. En cas que l'acció invoqui diverses vistes (apartat `view_mode` amb diverses opcions `tree`, `form`, `graph`...):

- El client GTK mostra, per cada acció, la primera de les vistes que indica l'apartat `view_mode` de l'acció i presenta un botó que permet anar alternant amb les altres vistes definides a l'apartat `view_mode`.
- El client web mostra, per cada acció, la primera de les vistes que indica l'apartat `view_mode` de l'acció i no permet navegar pels diferents tipus de vista definits a l'apartat `view_mode`.

Disseny d'un quadre de comandament per part de l'usuari

Suposem que l'usuari vol crear un quadre de comandament que incorpori la vista `tree` dels professors i la vista `graph` del nombre de cursos per professor. Fixem-nos que es tracta de dues vistes sobre diferents objectes d'OpenERP.

Executem l'opció *Settings* | *Personalització* | *Informe* | *Definició taulell* per crear un nou quadre de comandament i emplenem els camps:

- *Taulell*: amb un nom amb significat, com per exemple, *Professorat*.
- *Vista taulell*: en blanc, que crearà una vista amb el mateix nom que l'indicat a *Taulell* quan l'enregistrem.
- A la graella *Vista taulell*: afegim les accions que invoquen les vistes que volem visualitzar, i n'indiquem la ubicació a l'esquerra o a la dreta de la pantalla.

Per a la vista `tree` dels professors no tenim cap acció que la invoqui unívocament; disposem, però, de l'acció *Professors* (identificador `action_school_professor`) que invoca les vistes `tree` i `form`, que ens serveix per qualsevol dels dos clients, ja que en primer lloc està el mode `tree`. Seleccionem aquesta acció, la ubiquem a la zona esquerra de la pantalla i per títol introduïm el text que es desitja que aparegui al capdamunt de la vista (per exemple, *Professors*).

Per a la vista `graph` del nombre de cursos per professor tampoc tenim cap acció que invoqui unívocament aquesta vista, però disposem de l'acció *Nombre de cursos per professor* (identificador `action_school_professor_nbr`) que invoca les vistes `tree` i `graph`. Pel client GTK ens pot servir, perquè aquest client ens permet alternar entre els diferents modes, però el client web només permet veure la vista `tree` que està en primera posició i no és el que interessa. Fem-ho, però, per veure'n el funcionament i posteriorment ja ho retocarem.

Tenim el quadre de comandament definit, però no tenim cap opció de menú per invocar-lo. La definició del taulell incorpora un botó *Crea menú* que permet crear una opció de menú que invoqui el taulell. Només cal indicar el nom de la nova opció (*Quadres de comandament*) i seleccionar el menú pare (*Escola*, si volem que el menú principal tingui una nova opció de menú *Quadres de comandament* per incloure tots els quadres de comandament del mòdul *Cursos*).

Refresquem el menú i comprovem el funcionament des dels dos clients. Tal com hem avançat, el client web no permet alternar entre les diferents vistes associades a les accions invocades.

Per incorporar la vista `view_school_professor_nbr_graph` cal crear una nova acció, fet que es pot fer en el mateix punt en el qual es selecciona l'acció a assignar al taulell. Cal anar a modificar el taulell, eliminar de la graella l'acció *Nombre de cursos per professor* i afegir una vista creant una acció que invoqui la vista desitjada.

Podeu trobar la versió 20 del mòdul `school`, que incorpora una possible solució, a l'arxiu school_20.zip, dins l'apartat "Mòduls OpenERP i informes" dels annexos del web.

Disseny d'un quadre de comandament en un mòdul

Es vol incorporar, en el mòdul `school`, un quadre de comandament que incorpori la vista `tree` dels professors i la vista `graph` del nombre de cursos per professor. Fixem-nos que es tracta de dues vistes sobre diferents objectes d'OpenERP.

La definició d'un quadre de comandament consisteix en crear una vista `form` especial, una acció que invoqui la vista i una opció de menú que invoqui l'acció.

La vista `form` és especial perquè es defineix sobre el model `board.board`, específic per als quadres de comandament, amb arquitectura específica com podeu comprovar a la vista `board_school_professor_01` de la solució proposada. Aquesta vista `form` conté dos elements `column` (esquerra/dreta) en els quals cal ubicar les diverses accions que invocaran les vistes que ha de mostrar el quadre de comandament.

Per cada acció que cal incorporar en els elements `column`, cal tenir en compte que:

- En l'atribut `name` cal indicar l'identificador numèric de l'acció que s'invoca, que és desconegut (pot variar a cada instal·lació), però que en sabem el seu identificador XML i per això s'utilitza la sintaxi `"%(id)d"` en la qual `id` és l'identificador XML indicat dins el fitxer XML on resideix la declaració de l'acció.
- L'atribut `view_mode` permet canviar l'ordre de visualització de les vistes invocades per l'acció, cosa que no és possible quan el quadre de comandament es crea manualment des d'un client.
- L'atribut `creatable="true"` possibilita que des de la vista es pugui afegir registres, cosa que no és possible quan el quadre de comandament es crea manualment des d'un client.
- L'atribut `domain` permet indicar el domini de valors sobre els quals s'executa la vista.