Sistemes de gestió empresarial

CFGS.DAM.M10/0.13

Desenvolupament d'aplicacions multiplataforma

Generalitat de Catalunya
Departament d'Educació

ioc
institut obert
de catalunya

Aquesta col·lecció ha estat dissenyada i coordinada des de l'Institut Obert de Catalunya.

Coordinació de continguts
Joan Carles Pérez Vázquez

Redacció de continguts
Isidre Guixà Miranda

Primera edició: Setembre 2013

Dipòsit legal: B. 15218-2013

Introducció

Avui dia molta part de la competitivitat d'una empresa passa per tenir optimitzats i integrats els seus fluxos/processos interns i les seves relacions comercials externes, per aconseguir objectius bàsics com son les millores de la productivitat, la qualitat, el servei al client i la reducció de costos. Les organitzacions necessiten integració interna i col·laboració externa per augmentar les oportunitats de sobreviure en un mercat global, cada cop més competitiu.

Aquesta és la raó de la cerca d'un sistema que permeti assolir totes les necessitats del negoci, des de el control de les operacions financeres i la generació d'informes, les relacions i vendes amb els clients, la planificació i programació de la producció, fins al control d'inventari i costos. En l'actualitat, les organitzacions busquen implementar sistemes que els permeti controlar totes les àrea del negoci, de forma integral. Així, amb un sistema integrat, tota l'empresa, els seus sistemes i processos, es centralitzen i integren per al benefici de tota l'organització.

Com a conseqüència de tot aquest sistema integrat l'organització ha de veure com es generaren millors resultats, sempre que s'implanti el programari adequadament, oferint com a benefici el control i la monitorització de les operacions, l'eficiència administrativa, una productivitat més adequada, un servei al client més eficient, estalvi de costos i recolzament més fiable per a la pressa de decisions.

En aquest sentit, els sistemes ERP, de l'anglès *Enterprise Resource Planning*, coneguts àmpliament com a sistemes de planificació de recursos empresarials o sistemes de gestió empresarial, són sistemes que integren o pretenen integrar totes les dades i processos d'una organització en un sistema unificat.

Aquest mòdul té com a finalitat inicial l'aproximació conceptual al món dels sistemes de gestió empresarial i la seva implantació a l'empresa.

Al començament de la unitat "Sistemes ERP. Implantació", i des de un punt de vista teòric, s'examinen els diferents tipus de llicenciament de programaris ERP-CRM i els diferents tipus de desplegament de programari. També es veurà què són els sistemes ERP-CRM i les solucions BI i s'identifiquen els principals productes existents en l'actualitat. D'altra banda, s'identifiquen els punts a tenir en compte en la implantació tècnica d'un sistema ERP-CRM. A continuació es fa la instal·lació i configuració d'un ERP-CRM, en concret el programari OpenERP.

L'altra gran finalitat d'aquest mòdul és l'explotació i adequació dels sistemes de gestió empresarial desenvolupant components específics segons les necessitats i requeriments plantejats.

En la unitat "Sistemes ERP. Explotació i adequació", s'estudia el desenvolupament de nous mòduls que es puguin integrar al sistema, amb el programari i llenguatges específics. A continuació, aprendrem a ampliar i modificar mòduls existents, traduccions idiomàtiques, aspectes relatius a la seguretat, dissenyar i modificar informes

Els continguts d'aquest mòdul tenen una orientació bàsicament professionalitza-

dora, es proposa una formació pràctica i aplicable amb l'objectiu que l'alumnat aprengui a saber fer. Un cop llegits detingudament els continguts de cada apartat, cal realitzar les activitats i els exercicis d'autoavaluació per tal de posar en pràctica i comprovar l'assoliment dels coneixements adquirits. I, sobretot en les unitats referides a l'explotació i adequació dels sistemes ERP, és molt important la pràctica, en els propis ordinadors, dels exemples que es descriuen en el material docent.

Resultats d'aprenentatge

En finalitzar aquest mòdul l'alumne/a:

Sistemes ERP-CRM. Implantació

1. Identifica sistemes de planificació de recursos empresarials i de gestió de relacions amb clients (ERP-CRM) reconeixent les seves característiques i verificant la configuració del sistema informàtic.
2. Implanta sistemes ERP-CRM interpretant la documentació tècnica i identificant les diferents opcions i mòduls.

Sistemes ERP-CRM. Explotació i adequació

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).
2. Adapta sistemes ERP-CRM identificant els requeriments d'un supòsit empresarial i utilitzant les eines proporcionades pels mateixos.
3. Desenvolupa components per a un sistema ERP-CRM analitzant i utilitzant el llenguatge de programació incorporat.

Continguts

Sistemes ERP-CRM. Implantació

Unitat 1

Sistemes ERP-CRM. Implantació

1. Identificació de sistemes ERP-CRM i solucions BI
2. Implantació tècnica de sistemes ERP-CRM: OpenERP

Sistemes ERP-CRM. Explotació i adequació

Unitat 2

Sistemes ERP-CRM. Explotació i adequació - I

1. Explotació i adequació. OpenERP: el model
2. OpenERP: la vista i el controlador

Unitat 3

Sistemes ERP-CRM. Explotació i adequació - II

1. OpenERP: desenvolupament avançat
2. Reporting & BI

Sistemes ERP-CRM. Implantació

Isidre Guixà Miranda, Mikel López Villarroya

Sistemes de gestió empresarial

Índex

Introducció **5**

Resultats d'aprenentatge **7**

1 Identificació de sistemes ERP-CRM i solucions BI **9**
1.1 Llicències de programari . . . 9
1.2 Tipus de desplegament i requisits associats . . . 13
1.2.1 Des dels 'mainframes' fins al 'cloud computing' . . . 13
1.2.2 Requisits per a un desplegament . . . 17
1.3 Sistemes ERP . . . 19
1.3.1 Requisits per ser considerat ERP . . . 19
1.3.2 Funcionalitats dels sistemes ERP . . . 20
1.3.3 La llegenda de la implantació dels ERP . . . 31
1.4 Sistemes CRM i solucions BI, complements dels ERP? . . . 34
1.4.1 Funcionalitats dels sistemes CRM . . . 36
1.4.2 Funcionalitats de les solucions BI . . . 37

2 Implantació tècnica de sistemes ERP-CRM: Odoo **45**
2.1 Odoo ERP . . . 46
2.1.1 Característiques bàsiques . . . 46
2.1.2 Implantació tècnica d'Odoo . . . 48
2.2 Instal·lar Odoo . . . 49
2.2.1 Instal·lació d'Odoo 'All-In-One' en el SO Windows . . . 49
2.2.2 Instal·lació d'Odoo per a un servidor Linux . . . 54
2.2.3 Instal·lació d'Odoo sobre un contenidor Docker . . . 60
2.3 Usuaris al voltant d'un servidor Odoo . . . 64
2.4 Coneixements bàsics del servidor PostgreSQL . . . 66
2.4.1 Eina pgAdmin . . . 66
2.4.2 Configurar PostgreSQL per admetre connexions remotes a un host Windows . . . 73
2.5 Configuració inicial d'Odoo . . . 78
2.5.1 Configuració general d'Odoo . . . 80
2.5.2 Els mòduls no oficials. Instal·lació. . . . 85
2.5.3 Importació de dades a Odoo. Usuaris i treballadors . . . 90
2.6 Els mòduls d'Odoo . . . 93
2.6.1 Cicle de la comanda: vendes, compres i inventari . . . 93
2.6.2 Comunicació en línia: el mòdul web . . . 96

Introducció

Les empreses necessiten, per a una òptima gestió empresarial, un suport informàtic adequat a les seves necessitats. Per aquest motiu hi ha en el mercat diversos programes informàtics: gestió comercial, compravenda, facturació, comptabilitat, nòmines, producció, relació amb els clients... molts d'ells englobats en paquets que es distribueixen a unitats o de forma modular. El conjunt de programes que una empresa utilitza per acompanyar la seva gestió diària constitueixen el seu sistema de gestió empresarial.

Els primers sistemes de gestió empresarial van aparèixer amb el naixement de la informàtica, ja que un dels seus camps d'aplicació era el suport a la gestió de l'empresa. L'evolució per una banda de les tecnologies de la informació i, per l'altra, de la gestió de processos en les empreses, ens ha portat al moment actual en el qual es considera que els sistemes informàtics de gestió empresarial òptims per a una empresa són els anomenats sistemes ERP, de manera que podem trobar-nos amb la contradicció que el sistema informàtic de gestió d'una empresa es basi en un conjunt d'aplicacions que faciliten un funcionament complet i correcte i, en canvi, es consideri no òptim segons els cànons actuals, per no complir els requisits d'un sistema ERP.

La primera part del títol de la unitat formativa (*sistemes ERP-CRM*) introdueix el concepte de sistemes ERP i, també, el concepte de sistemes CRM. Els mots ERP i CRM corresponen a acrònims de l'anglès àmpliament utilitzats en la informàtica. Així, el mot ERP correspon a l'acrònim anglès d'*Enterprise Resource Planning* (planificació de recursos empresarials) i el mot CRM correspon a l'acrònim anglès de *Customer Relationship Management* (gestió de la relació amb els clients).

Per una banda tenim els sistemes ERP que engloben o pretenen englobar totes les dades i processos d'una organització en un sistema integrat. Per altra banda, tenim els sistemes CRM ideats per donar suport a la gestió de les relacions amb els clients, a la venda i al màrqueting. Si ens fixem en la definició dels sistemes ERP, la gestió que faciliten els sistemes CRM hauria d'estar inclosa en els sistemes ERP i la realitat és que la majoria dels actuals ERP incorporen una gestió completa CRM. Però també és veritat que encara hi ha moltes PIME que no tenen implantat un ERP o que tenen un ERP no actual i que, per elles, la implantació d'un sistema CRM pot portar a una millora substancial de la seva gestió. Per això és important conèixer també els sistemes CRM actuals.

La segona part del títol de la unitat formativa (*implantació*) porta a dues interpretacions. Una, d'abast molt ampli, que fa referència a tot el procés d'implantació d'un sistema ERP-CRM, dirigit per experts consultors. L'altre, d'abast més reduït, es refereix a la instal·lació, configuració i posada en marxa d'un sistema ERP-CRM, que podríem anomenar implantació tècnica del sistema ERP-CRM. El mot implantació en el títol d'aquesta unitat formativa fa referència a la segona interpretació.

Així doncs, la unitat formativa "Sistemes ERP-CRM. Implantació" contempla dues grans temàtiques: el coneixement dels sistemes ERP-CRM i la implantació tècnica dels sistemes ERP-CRM i, en conseqüència, s'ha dividit en dos apartats.

En l'apartat **"Identificació de sistemes ERP-CRM i solucions BI"** es presenten els sistemes ERP-CRM i les solucions BI. Pel que fa als sistemes ERP, veurem els requeriments per tal que un programari de gestió pugui ser considerat ERP, les funcionalitats facilitades pels sistemes ERP, la classificació actual dels sistemes ERP — tenint en compte els tipus de llicència i els tipus de desplegament—, la problemàtica de la implantació dels sistemes ERP i la situació dels ERP a les PIME, amb una visió dels principals productes actuals. Quant als sistemes CRM, farem una presentació de les seves funcionalitats i una ullada als principals productes actuals.

El mot BI correspon a l'acrònim anglès de *Business Intelligence* (intel·ligència de negoci). Les solucions BI són eines destinades a facilitar dades als dirigents empresarials, obtingudes a partir de les dades dels sistemes ERP-CRM, amb l'objectiu d'ajudar la presa de decisions. Els sistemes ERP-CRM incorporen cada vegada més solucions BI, però encara hi ha moltes PIME que no tenen implantat un ERP-CRM o que tenen un ERP-CRM no actual i que la utilització de solucions BI independents els pot ser molt convenient, és per això que s'inclouen en aquesta unitat formativa.

L'apartat **"Implantació tècnica de sistemes ERP-CRM: Odoo"** ens endinsa, com el seu nom indica, en què cal tenir en compte, de forma genèrica, per efectuar una implantació tècnica d'un sistema ERP-CRM. Com que no hi ha dos sistemes ERP-CRM en els quals el procés d'implantació tècnica sigui igual, posarem en pràctica el procés en el sistema ERP de codi obert Odoo.

Per tal d'assolir un bon aprenentatge cal estudiar els continguts en l'ordre indicat, sense saltar-se cap apartat i, quan es fa referència a algun annex del web, adreçar-s'hi i estudiar-lo. En els materials web trobareu molts vídeos referents a sistemes ERP-CRM i solucions BI; visualitzeu-los per fer-vos una idea dels productes que hi ha en el mercat. Una vegada estudiats els continguts del material paper i del material web, desenvolupeu les activitats web.

Què us sembla? Comencem!

Resultats d'aprenentatge

En finalitzar aquesta unitat l'alumne/a:

1. Identifica sistemes de planificació de recursos empresarials i de gestió de relacions amb clients (ERP-CRM) reconeixent les seves característiques i verificant la configuració del sistema informàtic.

- Reconeix els diferents sistemes ERP-CRM que hi ha al mercat.
- Compara sistemes ERP-CRM en funció de les seves característiques i requisits.
- Reconeix els magatzems de dades (*data warehouse*) acoblables i/o incorporats als diferents sistemes ERP-CRM.
- Identifica el sistema operatiu adequat a cada sistema ERP-CRM.
- Identifica el sistema gestor de dades adequat a cada sistema ERP-CRM.
- Verifica les configuracions del sistema operatiu i del gestor de dades per garantir la funcionalitat de l'ERP-CRM.
- Documenta les operacions realitzades.
- Documenta les incidències produïdes durant el procés.

2. Implanta sistemes ERP-CRM interpretant la documentació tècnica i identificant les diferents opcions i mòduls.

- Identifica els diferents tipus de llicència.
- Identifica els mòduls que componen el ERP-CRM.
- Realitza instal·lacions monoestació.
- Realitza instal·lacions client/servidor.
- Configura els mòduls instal·lats.
- Realitza instal·lacions adaptades a les necessitats plantejades en diferents supòsits.
- Instal·la i configura, si hi ha possibilitat, algun magatzem de dades adequat al sistema ERP-CRM.
- Comprova l'ERP-CRM.
- Documenta les operacions realitzades i les incidències.

1. Identificació de sistemes ERP-CRM i solucions BI

Les empreses necessiten, per a una òptima gestió empresarial, un suport informàtic adequat a les necessitats de l'empresa. Per aquest motiu hi ha en el mercat diversos programes informàtics: gestió comercial, compravenda, facturació, comptabilitat, nòmines, producció, relació amb els clients... molts d'ells englobats en paquets que es distribueixen com a unitats o de forma modular.

Els **sistemes ERP**, de l'anglès *Enterprise Resource Planning*, coneguts àmpliament com a sistemes de planificació de recursos empresarials, són sistemes que integren o pretenen integrar totes les dades i processos d'una organització en un sistema unificat. Aquesta definició pot portar a confondre els ERP amb els paquets comercials que engloben diversos programes. És important conèixer la frontera entre ambdós tipus de productes.

El centre de terminologia de la llengua catalana (TERMCAT) tradueix el terme ERP com a 'programari de gestió integrada'.

Els **sistemes CRM**, de l'anglès *Customer Relationship Management*, coneguts com a sistemes de gestió de la relació amb els clients, són sistemes que donen suport a la gestió de les relacions amb els clients, a la venda i al màrqueting.

Les **solucions BI**, de l'anglès *Business Intelligence*, conegudes com a solucions d'intel·ligència de negoci o solucions d'intel·ligència empresarial, són un conjunt d'eines destinades a facilitar dades als dirigents empresarials, obtingudes a partir de les dades dels sistemes ERP-CRM, amb l'objectiu d'ajudar a la presa de decisions. El ventall de solucions BI és ampli: des d'eines d'elaboració d'informes fins a sofisticades eines de gestió de cubs OLAP.

Abans de fer la instal·lació, configuració, explotació i adequació de sistemes ERP-CRM i solucions BI, ens convé conèixer:

1. Els tipus de llicenciament actuals.
2. Els tipus de desplegament (implantacions) actuals i requisits associats.
3. Les funcionalitats normalment proporcionades per les aplicacions ERP/CRM/BI.
4. Els principals productes existents en el mercat.

1.1 Llicències de programari

En el mercat actual trobem un gran nombre d'aplicacions que poden tenir utilitat a les empreses. Totes elles van acompanyades d'un determinat tipus de llicència. Per altra banda, ha proliferat un gran nombre de tipus de llicències de programari.

En conseqüència, ens cal poder reconèixer la llicència que acompanya cada programari i les seves implicacions.

Una **llicència de programari** és l'autorització o permís concedit pels autors del programari per poder-lo utilitzar, sota uns drets i deures.

A causa que els drets i deures que els autors poden assignar a les seves obres són de diversos tipus, han aparegut un gran nombre de tipus de llicències que, bàsicament, podem classificar en dos grans grups: programari privatiu i programari lliure.

Per **programari lliure** (*free software*) entenem aquell programari que respecta la llibertat total de l'usuari sobre el producte adquirit. Per **programari privatiu** entenem tot programari que no sigui lliure.

El nostre objectiu no és conèixer l'evolució que han tingut els conceptes *programari lliure* i *programari privatiu*, sinó conèixer els conceptes existents i utilitzats en el moment actual.

Programari privatiu

Hi ha força controvèrsia pel que fa a la nomenclatura de programari privatiu. Així, altres termes que s'utilitzen són programari propietari, programari esclau, programari tancat, programari privat i programari no lliure. El motiu de la controvèrsia radica en les connotacions dels diversos mots.

Pel que fa al programari lliure, ens cal saber que, segons la *Free Software Foundation*, un programari és lliure quan garanteix les **quatre llibertats** següents (enumerades a partir del valor zero); davant d'aquesta definició, qualsevol programari que violi alguna de les quatre llibertats passa a ser programari privatiu. Així, un programari és lliure quan es té:

1. Llibertat d'utilitzar el programa per a qualsevol propòsit.
2. Llibertat d'estudiar el funcionament del programa, modificant-lo i adaptant-lo a nous requisits.
3. Llibertat de distribuir còpies del programa.
4. Llibertat de millorar el programa i fer públiques les millores, de manera que tota la comunitat se'n beneficiï.

FREE SOFTWARE FOUNDATION

La 'Free Software Foundation' és una organització creada l'octubre de 1985 per Richard Stallman i altres promotors del programari lliure, per difondre aquest moviment.

Sovint, el concepte programari lliure es confon amb programari gratuït i/o amb codi obert i els tres conceptes són diferents, malgrat tenir punts en comú:

- La confusió entre programari lliure i **programari gratuït** és causada per l'ambigüitat del mot *free* en la llengua anglesa, on té doble significat: llibertat i gratuïtat. Certament, la majoria de programari lliure acostuma a ser gratuït, però això no és obligatori. Hi pot haver programari lliure no gratuït i programari gratuït no lliure. El concepte anglès a utilitzar per fer referència al programari gratuït (sigui o no lliure) és *freeware*.
- La confusió entre programari lliure i **codi obert** (*open source*) és simple d'explicar, ja que el programari lliure, per tal de garantir les llibertats 1 i 3, obliga a tenir accés al codi del programari, és a dir, el programari lliure té el codi obert. Però darrere dels mots programari lliure i codi obert hi ha dos moviments ben diferenciats des del punt de vista filosòfic.

La utilització del concepte de codi obert va aparèixer per primera vegada l'any 1998, quan alguns usuaris del moviment pel programari lliure el van utilitzar per substituir el nom *programari lliure* a causa de l'ambigüitat del terme *free* en la llengua anglesa. Però per alguns seguidors del moviment pel programari lliure la substitució no es va considerar adequada, ja que es perdia el sentit ètic i moral implícit en el mot llibertat utilitzat en la definició del programari lliure. Així es va produir una escissió del moviment pel programari lliure, apareixent la *Open Source Initiative*.

La 'Open Source Initiative' (OSI) és una organització creada el febrer de 1998 per Bruce Perens i Eric S. Raymond, dedicada a la promoció del codi obert.

La iniciativa pel codi obert exigeix que la distribució del **programari de codi obert** ha de verificar el següent decàleg:

1. Lliure redistribució: el programari ha de poder ser regalat o venut lliurement.
2. Codi font: el codi font ha d'estar inclòs o s'ha de poder obtenir lliurement.
3. Treballs derivats: la redistribució de modificacions ha d'estar permesa.
4. Integritat del codi font de l'autor: les llicències poden requerir que les modificacions siguin redistribuïdes només com a pegats.
5. Sense discriminació de persones o grups: no es pot deixar ningú a fora.
6. Sense discriminació d'àrees d'iniciativa: no es pot restringir a ningú que faci ús del programa en un camp específic d'activitat. Per exemple, no es pot impedir que el programa sigui utilitzat en un negoci o que s'utilitzi per a la investigació genètica.
7. Distribució de la llicència: s'ha d'aplicar els mateixos drets a tothom que rebi el programa.
8. La llicència no ha de ser específica d'un producte: el programa no es pot llicenciar només com a part d'una distribució major.
9. La llicència no ha de restringir cap altre programari: la llicència no pot obligar que algun altre programari que sigui distribuït amb el programari obert hagi de ser també de codi obert.
10. La llicència ha de ser tecnològicament neutral: l'acceptació de la llicència no es pot basar en una tecnologia o un estil d'interfície. Per exemple, no es pot requerir l'acceptació de la llicència a través d'un clic de ratolí o de cap forma específica del mitjà de suport del programari.

El decàleg del codi obert és compatible amb les quatre llibertats del programari lliure i, des d'un punt de vista pràctic, ambdós moviments són equivalents, però són totalment incompatibles des d'un punt de vista filosòfic.

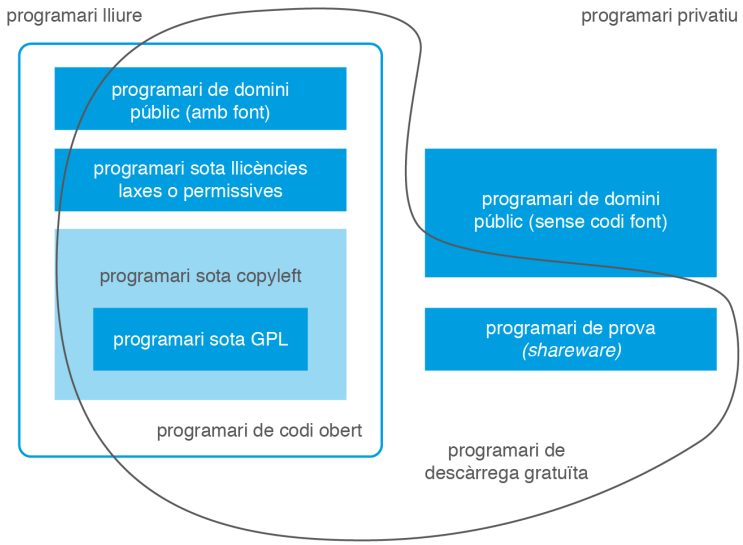
Pels defensors del codi obert, el fet de tenir accés total al codi font del programari és una qüestió pràctica que possibilita que el programari evolucioni, es desenvolupi i millori a una alta velocitat, més alta que la que es pot assolir en els processos convencionals de desenvolupament de programari. Pels defensors del codi obert

les llibertats esgrimides pel programari lliure no tenen importància; l'objectiu és, únicament, tenir accés al codi per tal d'assolir un codi millor. En conseqüència, pel moviment del codi obert, el codi tancat mai podrà ser millor que el codi obert.

Pels defensors del programari lliure allò que importa és la defensa de les llibertats; l'accés al codi és conseqüència de les llibertats 1 i 3 i la qualitat del codi tancat no té per què ser inferior a la del codi obert.

La distinció dels conceptes *programari lliure*, *programari privatiu* i *codi obert* és el primer pas per categoritzar un programari, però ens manca conèixer més conceptes utilitzats actualment. La figura 1.1, original de Chao-Kuei i posteriorment actualitzada per altres, situa les diferents categories del programari, que ens cal identificar:

FIGURA 1.1. Diferents categories del programari



1. Programari **de domini públic**: programari que no està protegit amb copyright. El copyright reflecteix la possessió del dret d'explotació i, per tant, només el pot fer constar el titular o cessionari d'aquest dret.

2. Programari **sota *copyleft*** (còpia permesa): les llicències *copyleft* són aquelles que exerceixen els autors del programari, emparats en la legislació de copyright, per permetre la lliure distribució de còpies i versions modificades d'una determinada obra. La majoria de les llicències *copyleft* exigeixen que els drets concedits es mantinguin en les versions modificades del producte.

3. Programari **sota GPL**: la llicència GPL (Llicència Pública General de GNU) és una llicència creada per la *Free Software Foundation*, orientada a protegir la lliure distribució, modificació i utilització del programari, de manera que el programari cobert per aquesta llicència és programari lliure i queda protegit de qualsevol intent d'apropiació que restringeixi les llibertats del programari lliure. La formulació de GPL és tan restrictiva que impedeix que el programari sota aquesta llicència pugui ser integrat en programari privatiu.

4. Programari **sota llicències laxes o permissives**: les llicències laxes o permissives són llicències de programari lliure flexibles respecte a la distribució, de manera que el programari pugui ser redistribuït com a programari lliure o privatiu. Són llicències sense *copyleft*, ja que consideren que el treball derivat no té per què mantenir el mateix règim de drets d'autor que l'original. Això dóna total llibertat a qui rep el programari per desenvolupar-ne qualsevol producte derivat, i li permet escollir entre l'ampli ventall de llicències existents. Des del punt de vista dels usuaris, però, aquestes llicències es poden considerar com una restricció a les llibertats que defensa el programari lliure. Exemples de llicències d'aquest tipus són les llicències BSD i MIT.

5. Programari **de prova** (*shareware*): les llicències *shareware* autoritzen la utilització d'un programa per tal que l'usuari l'avaluï i posteriorment l'adquireixi. Aquest programari acostuma a tenir unes limitacions, sigui en el temps d'utilització o en les funcionalitats permeses.

1.2 Tipus de desplegament i requisits associats

Tradicionalment, les aplicacions ERP/CRM/BI han estat allotjades a les instal·lacions de les organitzacions compradores de les llicències de l'aplicació; desplegament conegut majoritàriament com a *on-premise* i, en menor grau, com a *in-house*. Però això està canviant.

La història dels tipus de desplegament de les aplicacions de gestió empresarial ha anat lligada a l'evolució que ha tingut la tecnologia. En aquests moments podem dir que estem **entrant en una nova època**: l'època de la informàtica en núvol (*cloud computing*) i amb ella, diversos models de desplegament (IaaS, PaaS i SaaS) que s'imposaran o conviuran amb el model tradicional *on-premise*.

Per saber on som, ens convé, en un primer lloc, conèixer els tipus de desplegament que hi ha hagut al llarg de la història i, per poder dur a terme desplegaments en el moment actual, ens cal poder distingir els requisits associats.

1.2.1 Des dels 'mainframes' fins al 'cloud computing'

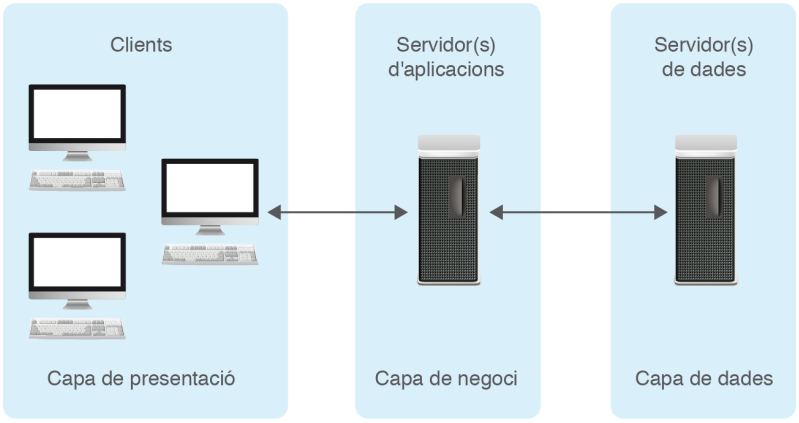
En la **primera època** (la dècada dels 60 i dels 70) les aplicacions residien en grans ordinadors (*mainframes*) ubicats en les dependències de l'organització i els usuaris disposaven de terminals (pantalles sense memòria ni capacitat de procés) connectades amb l'ordinador central.

La **segona època** arriba en la dècada dels 80, amb l'eclosió dels ordinadors personals. Les aplicacions empresarials van anar adoptant l'arquitectura de dues capes (client-servidor), en les quals continua existint l'ordinador central (servidor –un o diversos–) que conté les bases de dades i en la qual la terminal de l'anterior

època queda substituïda per l'ordinador personal que, en disposar de memòria i capacitat de procés, incorpora les aplicacions a executar. L'arquitectura client-servidor ensopega aviat amb el problema del manteniment de les aplicacions, ja que cada vegada que la lògica de negoci canvia o evoluciona cal actualitzar l'aplicació en tots els ordinadors personals clients.

Per aquest motiu, s'adopta ben aviat l'**arquitectura de tres capes** (presentació-negoci-dades) il·lustrada a la figura 1.2, en la qual els clients tenen aplicacions senzilles que únicament presenten les dades subministrades per un o diversos servidors d'aplicacions, contenidors de la capa de negoci, que confeccionen aquelles dades a partir de la informació subministrada pels servidors de la capa de dades.

FIGURA 1.2. Arquitectura de tres capes



La **tercera època** s'inicia a mitjans de la dècada dels 90, coincidint amb el *boom* d'Internet i va acompanyada de la contínua millora de l'amplada de banda. Les aplicacions empresarials cerquen mecanismes per facilitar la connexió dels òrgans de comandament de les empreses des d'ubicacions remotes. Això fa que proliferin programaris que, aprofitant Internet, faciliten la connectivitat remota i obren en els dispositius remots (portàtils i PDA) sessions client contra el servidor d'aplicacions. De ben segur que un dels programaris més coneguts és l'escriptori remot del sistema operatiu Microsoft Windows. Però aquests programaris presenten un problema: cal tenir instal·lat en el dispositiu remot el programari adequat per poder establir la connexió i això no sempre és factible. Ara bé, sense por a equivocar-nos, quin és el programari que tenen avui en dia tots els dispositius que es connecten a Internet, sigui quin sigui el sistema operatiu utilitzat (Windows, Linux, Mac, iOS, Android...)? Un navegador, oi? En conseqüència, es tracta d'aconseguir que a través del navegador puguem executar les aplicacions empresarials.

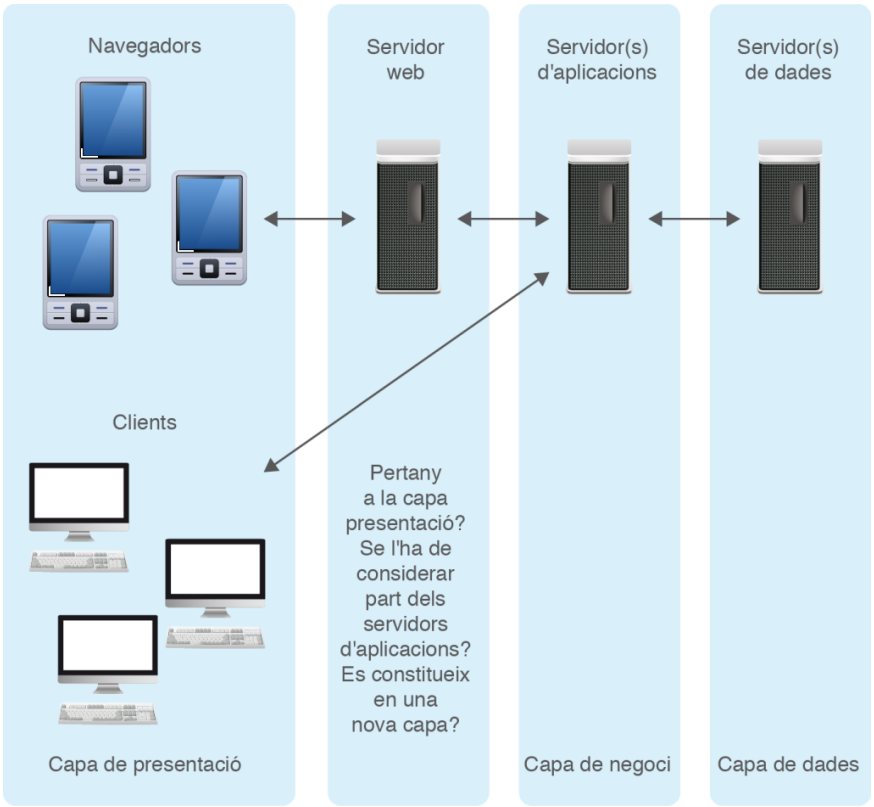
Durant la primera dècada del **segle XXI**, encara dins la tercera època, les aplicacions empresarials es van acomodant a la nova situació tecnològica i faciliten solucions accessibles des dels navegadors web. L'arquitectura de tres capes continua sent vàlida per a la nova situació. Simplement cal afegir un servidor web davant el(s) servidor(s) d'aplicacions per permetre la connexió des dels navegadors. Els clients tradicionals poden continuar existint i es comuniquen

directament amb el(s) servidor(s) d'aplicacions. La figura 1.3 n'il·lustra la situació.

En aquesta nova arquitectura hi ha desavinences sobre la capa on ubicar el servidor web. Hi ha autors que, a causa del fet que el servidor web simplement s'encarrega de confeccionar les pàgines que es visualitzen en el navegador, el consideren com a part de la capa de presentació. D'altres, com que és un servidor d'aplicacions, l'ajunten amb els servidors d'aplicacions on hi ha la capa de negoci. Per últim, hi ha autors que parlen d'arquitectura de quatre capes, destinant una capa específicament al servidor web.

L'arquitectura de quatre capes (aplicacions empresarials que permeten l'accés web) és d'extrema actualitat. Les aplicacions que no incorporen aquesta funcionalitat estan abocades a la desaparició. Poden sobreviure a causa del cost que suposa un canvi total de programari, però difícilment podran ampliar la seva quota de mercat.

FIGURA 1.3. Arquitectura de tres (o quatre) capes per a web



Finalment, ens trobem en el futur que ja és present: la **quarta època**. La informàtica en núvol (*cloud computing*) és un sistema d'emmagatzematge i ús de recursos informàtics basat en el servei en xarxa, que consisteix a oferir a l'usuari un espai virtual, generalment a Internet, en què pot disposar de les versions més actualitzades de maquinari i programari.

Hi ha tres models d'informàtica en núvol:

1. **Infraestructura com a servei** (IaaS, de *Infraestructure as a Service*), en el qual l'usuari contracta únicament les infraestructures tecnològiques (capa-

citat de procés, d'emmagatzematge i/o de comunicacions) sobre les quals instal·la les seves plataformes (sistemes operatius) i aplicacions. L'usuari té el control total sobre les plataformes i aplicacions, però no té cap control sobre les infraestructures.

2. **Plataforma com a servei** (PaaS, de *Platform as a Service*), en el qual l'usuari contracta un servei que li permet allotjar i desenvolupar les seves pròpies aplicacions (siguin desenvolupaments propis o llicències adquirides) en una plataforma que disposa d'eines de desenvolupament per tal que l'usuari pugui elaborar una solució; en aquest model, el proveïdor ofereix l'ús de la seva plataforma que a la vegada es troba allotjada en infraestructures, de la seva propietat o d'altri. L'usuari no té cap control sobre la plataforma ni sobre la infraestructura, però manté el control total sobre les seves aplicacions.

3. **Programari com a servei** (SaaS, de *Software as a Service*), en el qual l'usuari contracta la utilització d'unes determinades aplicacions sobre les quals únicament pot exercir accions de configuració i parametrització permeses pel proveïdor. L'usuari no té cap control sobre l'aplicació, la plataforma i la infraestructura.

Els models IaaS i PaaS ja fa temps que s'estan utilitzant (des que l'amplada de banda ho ha fet possible) i el model SaaS també en aplicacions vinculades a Internet, com per exemple el correu electrònic. En canvi, fins fa ben poc (cap a l'any 2010) no han començat a aparèixer aplicacions empresarials (ERP/CRM/BI) sota el model SaaS.

Distingir entre els models IaaS i PaaS

No hem de confondre tenir una aplicació empresarial en el núvol, de la qual nosaltres n'hem adquirit llicències, però hem optat per tenir-la instal·lada a Internet (model IaaS o PaaS) en lloc de tenir-la a casa nostra (*on-premise*), amb contractar la utilització d'una aplicació que algú té allotjada en el núvol (model SaaS) i per la qual no hem d'adquirir cap llicència sinó únicament prestacions (nombre d'usuaris i funcionalitats).

Entre els **beneficis del model SaaS**, cal considerar:

- Integració comprovada dels serveis en xarxa.
- Prestació de serveis en l'àmbit mundial.
- Cap necessitat d'inversió en maquinari.
- Implementació ràpida i sense riscos. La posada en marxa només precisa de la configuració i parametrització permesa pel proveïdor.
- Actualitzacions automàtiques ràpides i segures.
- Ús eficient de l'energia, davant l'energia requerida pel funcionament d'una infraestructura *on-premise*.

Entre els **inconvenients del model SaaS**, cal considerar:

- Dependència dels proveïdors de serveis.
- Disponibilitat de l'aplicació lligada a la disponibilitat d'Internet.
- Por a sostracció o robatori de les dades "sensibles" del negoci, ja que no resideixen a les instal·lacions de les empreses.
- Perill de monopolis referents als serveis facilitats pels proveïdors.
- Impossibilitat de personalitzar l'aplicació, fora de la configuració i parametrització permesa pel proveïdor.
- Actualitzacions periòdiques que poden incidir de manera negativa en l'aprenentatge dels usuaris d'orientació no tecnològica.
- Existència de focus d'inseguretat en els canals a recórrer per arribar a la informació, si no s'utilitzen protocols segurs (HTTPS) per no disminuir la velocitat d'accés.
- Possible degradació en els serveis subministrats pel proveïdor davant l'augment de clients.

El model Saas i les aplicacions empresarials (ERP/CRM/BI)

Sembla que el model SaaS és una tendència de futur, sobretot per petites i mitjanes empreses que no disposen de recursos informàtics adequats per poder respondre al repte d'adquirir llicències d'una aplicació empresarial (ERP/CRM/BI) i procedir a la seva instal·lació/configuració/personalització (sigui sota model *on-premise* o sota models IaaS/PaaS). Llavors, endinsar-nos en la implementació, explotació i adequació dels sistemes de gestió empresarial sembla una incongruència, oi? Prenguem-nos-ho per la banda positiva: ens convé introduir-nos en els sistemes de gestió empresarial per poder assessorar a les petites i mitjanes empreses que ens demanin consell i per dur a terme un correcte desplegament en aquelles organitzacions que optin pels models *on-premise* o IaaS/PaaS.

1.2.2 Requisits per a un desplegament

Els desplegaments d'aplicacions empresarials avui en dia poden tenir lloc sota dos models: ***on-premise*** (a casa del comprador de les llicències) o **IaaS/PaaS** (dues modalitats d'informàtica en núvol). En qualsevol cas, hem de pensar que l'aplicació empresarial està desenvolupada sota l'arquitectura web de tres capes i, per tant, cal disposar de:

- Servidor d'aplicacions.
- Servidor web, que possiblement compartirà maquinari amb el servidor d'aplicacions.
- Servidor de dades (SGBD) que molt possiblement serà un SGBD relacional o objecte-relacional.

Per atendre a aquestes necessitats, cal **avaluar què necessitem i què tenim**. Aquesta tasca, però, s'escapa de les capacitats d'un desenvolupador de programari i són tasques per encomanar a consultors i administradors de sistemes. Però, és possible que ens toqui fer-ho en una PIME que ens hagi demanat consell i no hi hagi consultors ni administradors de sistemes. En un cas així, caldrà:

- Identificar els requisits directes de maquinari (bàsicament RAM, CPU i capacitat de disc dur) especificats pel programari de gestió empresarial que s'ha d'instal·lar, tenint en compte la conveniència o no de virtualitzar els servidors.
- Identificar l'SGBD amb el que pot treballar el programari que s'ha d'instal·lar. Algunes vegades, un mateix programari de gestió empresarial permet utilitzar diferents SGBD, situació en la qual cal analitzar quin d'ells és millor en funció de les necessitats de l'empresa i del seu cost, tenint en compte que n'hi ha de molt potents amb versions gratuïtes. Així, per exemple, una botiga de bicicletes que adquireixi un ERP per dur la gestió informatitzada dels circuits compravenda, inventari i comptabilitat és possible que en tingui prou amb un SGBD ofimàtic, com ara Microsoft Access, mentre que un supermercat, amb el mateix ERP, és possible que no en tingui prou amb un SGBD ofimàtic i en precisi un de major potència.
- Identificar els requisits indirectes de maquinari a partir dels requisits de maquinària propis de l'SGBD escollit.
- Identificar mecanismes idonis per a efectuar còpies de seguretat de les dades que permetin la recuperació segons les necessitats de disponibilitat de l'organització. Tot programari de gestió empresarial ha d'anar acompanyat d'un mecanisme de recuperació adequat que cal testar periòdicament. En una organització amb disponibilitat 24x7 (és a dir, que no pot aturar en cap moment) caldrà preveure una estratègia de còpies de seguretat en calent i això repercutirà en l'elecció de l'SGBD. En canvi, en una organització que s'aturi unes hores al dia, podem preveure una estratègia de còpies de seguretat en fred.

  En qualsevol cas (còpies en calent o en fred), cal pensar en la necessitat o no de disposar d'un sistema de còpies que permeti, davant d'una catàstrofe, la recuperació de tots els moviments efectuats des de la darrera còpia de seguretat fins al moment de la catàstrofe.

  És a dir, ¿si la darrera còpia de seguretat (en calent o en fred) és de les 0.00 h de la nit anterior i a les 11.30 h es produeix una catàstrofe que ens obliga a tibar de la còpia de la nit anterior, podem assumir haver perdut tots els moviments efectuats des de la nit anterior fins al moment de la catàstrofe? Els grans SGBD permeten activar mecanismes de registre diari (*log* en anglès) que emmagatzemen cronològicament en un fitxer les operacions de processament de dades efectuades a la base de dades, de manera que davant una caiguda del sistema i de la darrera còpia de seguretat, permeten restablir tots els moviments efectuats en el sistema.
- Identificar mecanismes per recuperar el sistema informàtic davant un error de maquinari. Davant un mal funcionament de qualsevol peça de maquinari

(placa base, memòria, processador o disc dur), encara que tinguem contractat un servei de manteniment, podem assumir tenir el sistema aturat? Hi ha ocasions en què és possible (botiga d'informàtica, en la qual si fem alguna venda podem anotar-la a mà) i ocasions en què no és possible (us imagineu una botiga en línia d'Internet en la qual falla el sistema informàtic i ha d'estar aturada unes hores?). En el cas que no sigui possible una aturada d'hores, quina és la millor solució?

Avui en dia la utilització de servidors NAS per a l'emmagatzematge amb funcionalitats RAID activades conjuntament amb la virtualització dels servidors és, possiblement, la millor solució. Hi ha sistemes que permeten tenir els servidors virtualitzats en un servidor de virtualització que cap en un llapis òptic (és a dir, pot no ser en un disc dur) de manera que, davant una aturada de la màquina (problema de placa base, processador o memòria) podem utilitzar el llapis òptic per posar en marxa ràpidament el servidor de virtualització en qualsevol altra màquina (encara que tingui menys prestacions). A més, un problema en l'emmagatzematge en el servidor NAS queda cobert per les funcionalitats RAID activades, de manera que la recuperació pot ser molt ràpida.

1.3 Sistemes ERP

Els sistemes ERP, de l'anglès *Enterprise Resource Planning*, coneguts àmpliament com a sistemes de '**planificació de recursos empresarials**', són sistemes que integren o pretenen integrar totes les dades i processos d'una organització en un sistema unificat.

Així doncs, segons la definició anterior, un ERP ha de permetre la gestió de la producció (si l'organització incorpora processos productius), la gestió completa dels circuits de compravenda (logística, distribució, inventari i facturació) i la gestió financera. Poden incorporar també, en moltes ocasions, una gestió de recursos humans. En l'actualitat, molts d'ells incorporen una **gestió de CRM** (gestió de la relació amb els clients).

Com a tècnics, el nostre objectiu és instal·lar l'ERP i configurar-lo, parametritzar-lo o adequar-lo a les necessitats de l'organització. Per poder abordar amb garanties aquest objectiu ens cal conèixer prèviament com són els ERP, de la mateixa manera que un mecànic de cotxes, abans d'introduir-se en la mecànica, ha de conèixer com és un automòbil.

1.3.1 Requisits per ser considerat ERP

En el mercat hi ha moltes aplicacions de gestió empresarial i no totes elles poden ser considerades un ERP; són simplement aplicacions de gestió i hi ha

diferències fonamentals entre les aplicacions de gestió i els ERP, malgrat l'intent de moltes empreses, mitjançant estratègies de màrqueting, d'intentar vendre els seus productes amb la denominació ERP per tal d'obtenir un valor agregat als seus productes sense incrementar la seva funcionalitat.

Hi ha tres característiques fonamentals que defineixen un ERP:

- **És un sistema integral**: la mateixa definició d'ERP indica que és una aplicació que integra en un únic sistema tots els processos de negoci de l'empresa, així manté les dades d'una forma centralitzada. Això implica que la informació no pot estar duplicada i que només s'introdueix una única vegada. Aquesta definició descarta:
  - Programes basats en múltiples aplicacions (a vegades denominades *suite*) independents o modulars que dupliquen la informació (malgrat que l'enllacin automàticament).
  - Programes que no centralitzen la informació en una única base de dades.
  - Programes que no emmagatzemen les dades en un SGBD sinó que utilitzen sistemes gestors de fitxers, anteriors als SGBD.
- **És un sistema modular**: un ERP es compon de diversos mòduls, on cada mòdul se centra en una àrea de negocis de l'empresa. Normalment els ERP tenen uns mòduls troncals (bàsics) que s'adquireixen amb la compra de l'ERP (gestió de compravenda, control d'inventari, comptabilitat) i d'altres mòduls que s'adquireixen segons les necessitats de l'organització (gestió de projectes, gestió de campanyes, gestió de terminals punt de venda, comerç electrònic, producció per fases, traçabilitat, gestió de la qualitat, gestió de la cadena de subministrament...). És molt possible que una empresa no necessiti utilitzar, en un inici, tots els mòduls que facilita l'ERP, però és important saber que l'ERP els contempla, de cara a possibles necessitats de futur. En cas que sigui necessària la seva utilització, l'organització no es veurà abocada a un canvi de programari en les àrees on ja estava utilitzant l'ERP.
- **És un sistema adaptable**: no hi ha dues empreses iguals i, per això, els ERP han de permetre l'adaptació a necessitats diverses, objectiu que s'assoleix a través de la configuració i parametrització dels processos empresarials. Fins i tot alguns ERP disposen d'eines de desenvolupament integrades que permeten desenvolupar processos segons les necessitats de cada empresa.

1.3.2 Funcionalitats dels sistemes ERP

Un ERP integra en un únic sistema tots els processos de negoci de l'empresa: compravenda, producció, comptabilitat... Per una persona que mai hagi tingut contacte amb un ERP o amb una aplicació de gestió empresarial, què vol dir això? Ben segur que, si no s'ha interaccionat mai amb un ERP o aplicació de gestió,

s'ha de fer costa amunt entendre què vol dir "integra en un únic sistema tots els processos de negoci".

És important presentar, en un llenguatge entenedor per a persones no formades en la branca administrativa-comercial, les funcionalitats que acostumen a facilitar els programes de gestió empresarial. I ja que aquests materials estan adreçats a informàtics, utilitzarem mots de l'argot informàtic.

El programari de gestió empresarial acostuma a estar **presentat en apartats** (menús) que es corresponen bastant als mòduls instal·lats. A més, sempre hi ha uns apartats bàsics, existents independentment dels mòduls instal·lats.

Administració o configuració

L'apartat d'administració o configuració és bàsic i és una opció a la qual només tenen accés els usuaris administradors del producte i **des de la qual s'ha de poder**:

- Definir les dades de l'organització (nom, raó social, domicili fiscal, NIF...)
- Configurar els paràmetres de funcionament que permeti el programari d'acord amb els requisits de l'organització.
- Definir l'esquema de seguretat (usuaris, grups d'usuaris/rols i permisos d'accés de les diferents opcions del programari als usuaris/rols).

El procés d'instal·lació acostuma a crear un usuari administrador que és el que després podrà definir tot l'esquema de seguretat i també un conjunt de rols predefinits.

Fitxers mestres: tercers i productes

En les aplicacions informàtiques el concepte de fitxer mestre s'utilitza per fer referència a un conjunt de registres corresponents a un aspecte important dins l'aplicació. Així, per exemple, en una aplicació de gestió podríem parlar del fitxer mestre de clients, proveïdors, venedors, productes, pla de comptes i comandes, albarans o factures de compra o venda. Per altra banda, es continua utilitzant el mot fitxer, provinent de l'època en què les dades s'emmagatzemaven en sistemes gestors de fitxers, malgrat que avui en dia les dades s'emmagatzemin en SGBD.

Tradicionalment, en el programari de gestió empresarial quan es parla de **fitxers mestres** ens referim a les entitats client, proveïdors i productes, que existeixen per si mateixes i per les quals es facilita un formulari de manteniment, que normalment s'anomena fitxa del client, proveïdor o producte, des d'on gestionar els corresponents registres. La resta d'entitats, a escala informàtica, com per exemple comandes, albarans i factures, no s'acostumen a incorporar en el paquet dels fitxers mestres, perquè els seus registres no existeixen per si mateixos, sinó que necessiten l'existència d'altres entitats, com els clients, els proveïdors i els productes.

Darrerament, hi ha una tendència a englobar clients i proveïdors en una entitat anomenada **tercers o interlocutors comercials**. Això és a causa que un client de l'organització pot ser a la vegada proveïdor i, en conseqüència, les seves dades haurien d'estar duplicades en ambdós fitxers.

El concepte *tercer* és més genèric i engloba tots els ens amb els quals l'empresa pot mantenir una relació: clients, proveïdors, empleats, bancs i qualsevol altre tipus d'ens que pugui aparèixer. D'aquesta manera, si un empleat passa a ser, en un moment donat, client no tindrà informació duplicada en el sistema.

El manteniment de tercers acostuma a ser un programa que conté una pantalla principal que recull les dades principals del tercer (nom, NIF, domicili, telèfon, correu electrònic...) i unes caselles de verificació per marcar-lo com a client, proveïdor, empleat, banc, etc. Segons la manera que tingui activades les diferents caselles de verificació, s'activen diferents pantalles per informar de les dades necessàries, és a dir, si el tercer és marcat com a client, s'activa una pantalla amb les dades específiques del tercer quan actua com a client (tarifa assignada, domicilis d'enviament i de facturació, descomptes especials...) i de manera similar si el tercer és marcat com a proveïdor o com a empleat, etc.

El fitxer d'**articles o productes** és l'altre fitxer mestre fonamental en el programari de gestió empresarial. Què entenem per producte? Des del punt de vista del programari de gestió empresarial, dins el fitxer de productes hi entra:

- Tot allò que l'empresa ven (bé o servei) i que hagi estat adquirit o produït per l'empresa.
- Tot allò que l'empresa adquireix per poder satisfer les necessitats de producció (primeres matèries).

A vegades, algunes organitzacions també introdueixen en el fitxer de productes els conceptes de despesa (electricitat, aigua, lloguers...), ja que utilitzen el circuit de compra de l'ERP per introduir aquest tipus de despesa en l'aplicació comptable.

Observem que hi ha tipus de productes pels quals interessarà portar un inventari i d'altres pels quals l'inventari no té sentit (serveis, despeses...). Per tant, la fitxa d'un article o producte acostuma a incorporar una casella de verificació segons l'article és o no és inventariable.

Els articles s'acostumen a classificar, per poder obtenir estadístiques de compra, venda i/o producció de forma agrupada. Així, és molt normal veure com els ERP utilitzen conceptes com: categoria de producte, família de producte, grup de producte, etc.

Els articles també acostumen a tenir una casella de verificació per ser marcats com a article de compra, article de venda, article de consum en fabricació o bé article de producció. Segons tingui activades les diferents caselles de verificació, s'activen diferents pantalles per informar de les dades corresponents.

Una altra característica molt important i que no tots els ERP permeten és el poder **gestionar l'article sota diferents tipus d'unitats**. Així, per exemple, és possible

que comprem l'article en litres i el venguem en quilos o que el tipus d'unitat a utilitzar estigui en funció del client, en el cas de venda, o del proveïdor, en el cas de compra.

Les **existències mínimes i màximes** que es desitgen tenir d'un producte al magatzem és també una dada fonamental. Per una banda, en articles amb molta rotació pot interessar garantir una existència mínima, per tal de poder efectuar un servei ràpid en cas de venda o utilització en cas de producció (matèria de consum en processos de fabricació). I, per altra banda, pot interessar tenir assignada una existència màxima a cobrir en el cas que l'estoc de l'article sigui inferior a l'existència mínima. És molt interessant que l'ERP tingui mecanismes d'alerta per detectar els productes que, a causa d'un moviment de sortida (venda, consum de fabricació, regularització), passen a tenir un estoc inferior a l'existència mínima indicada, així s'avisa al responsable per tal que iniciï el procés de reposició que pertoqui (comprar-lo o fabricar-lo); el fet que l'article tingui assignada una existència màxima, pot servir per indicar la quantitat a reposar.

Un bon ERP hauria de permetre gestionar existències dels articles en diversos magatzems i indicar existències mínimes i màximes per a cada magatzem.

Molts ERP també contemplen, a títol informatiu a la fitxa del producte, les quantitats pendents de recepció (comandes de compra), les quantitats pendents de servir (comandes de venda), les quantitats pendents de consumir (en ordre de fabricació on el producte intervingui com a primera matèria) i les quantitats pendents de fabricar (quan es tracta d'un producte que fabriquen). Aquestes quantitats, que mai són modificables per l'usuari i que, en cas d'existir, són només de visualització, són redundants, ja que els seus valors són calculables a partir de comandes de compra, comandes de venda i ordres de fabricació, però el seu càlcul és costós (implicaria fer un recorregut per totes les comandes de compravenda i ordres de fabricació) i, per això, és possible que l'ERP les contempli a la fitxa de producte i les actualitzi de forma automàtica en els processos de gestió dels circuits de compravenda i fabricació.

Si la nostra organització gestiona productes peribles, cal que l'ERP faciliti un **control de lots**, amb dates de caducitat. Això implica que per cada producte perible que tenim en existència, cal saber els lots afectats, llur data de caducitat i el nombre d'unitats de cada lot. Així mateix, cal mantenir la traçabilitat, controlant els proveïdors i els clients implicats en la compravenda dels productes peribles.

Un tema similar a la gestió de lots, però sense data de caducitat ni necessitat de saber l'existència de cada lot, és la gestió de **números de sèrie**, necessària segons el tipus de producte que es comercialitzi. Els ERP han de facilitar també aquesta gestió.

Codis de producte dels clients o proveïdors

La relació comercial que es té amb els clients o proveïdors acostuma a ser la compravenda de productes del catàleg, però ben segur que la codificació i denominació dels productes no té res a veure amb la codificació i denominació dels mateixos productes pel client o proveïdor. En moltes ocasions, encara que no sempre, es fa necessari, en la documentació que s'intercanvia amb el client o proveïdor, incloure la codificació i denominació del producte per al client o proveïdor.

Això implica que l'ERP ha de facilitar la possibilitat d'introduir, per als clients o proveïdors que interessi, la codificació i denominació dels articles que ens compra o ven. Normalment els ERP faciliten dos programes, tal com mostra la figura 1.4.

FIGURA 1.4. Pantalles facilitades pels ERP per relacionar la nostra codificació d'articles amb la codificació dels proveïdors o clients

Codi de tercer: Nom:

Codi producte nostre	Denominació	Codi producte pel tercer	Denominació

Codi de producte: Denominació:

Codi de tercer	Nom	Codi producte pel tercer	Denominació

Taules bàsiques

Les taules bàsiques són fitxers de pocs registres i amb poca volatilitat (es modifiquen molt poc) que contenen definicions codificades de conceptes a utilitzar en molts dels programes de l'ERP.

Alguns exemples de taules bàsiques són països, províncies, tipus de clients, tipus de proveïdors, zones, idiomes, famílies de productes, grups de famílies, magatzems, unitats de mesura, formes de pagament, tipus d'enviament, tipus de comandes, sèries de facturació, formats d'impressió, fabricants, tipus de matèries, etc.

La taula de grups de famílies de productes permet tenir dos nivells de catalogació de productes.

Els continguts d'aquestes taules, a banda de ser utilitzats en els diversos processos informàtics de l'ERP (manteniments de fitxers mestres, circuits de compravenda, processos de fabricació...) poden ser bàsics a l'hora d'obtenir resultats en els processos d'intel·ligència de negoci, ja que són utilitzats per fer agrupacions.

Compres

L'apartat de compres comprèn els programes necessaris per cobrir el circuit de compres: tarifes de proveïdor, comandes a proveïdor, recepció de mercaderia i entrada de factura de proveïdor.

En referència a les tarifes de proveïdor, en moltes ocasions els proveïdors comuniquen les seves tarifes i per això interessa tenir-les introduïdes en el sistema informàtic. Això suposa, en principi, una gran feina d'introducció de dades i els ERP poden facilitar mecanismes per automatitzar la introducció de les tarifes de proveïdor. Així mateix, el **mòdul de tarifes de proveïdor** hauria de poder contemplar:

- Tarifes i/o descomptes especials en un interval de dates (ofertes).
- Tarifes i/o descomptes especials en funció de la quantitat de producte, definit segons un escalat (a més quantitat, menor preu net o major descompte).

La **gestió de comandes a proveïdors** és un programa que ha de permetre introduir en el sistema informàtic una comanda a proveïdor per tal de, una vegada introduïda, fer-la arribar al proveïdor. La tecnologia ens permet, avui en dia, una vegada la comanda ha estat enregistrada, enviar-la per correu electrònic o per fax al proveïdor, sense necessitat d'arribar-la a imprimir en paper. L'ERP, en el procés de generació de la comanda de compra, acostuma a proposar, per defecte, els valors que tenim pactats amb el proveïdor i que resideixen a la seva fitxa (manteniment de tercers – pestanya de proveïdors) i l'usuari simplement els haurà de validar.

S'ha de tenir en compte que el programa de gestió de comandes a proveïdors modifica el camp *quantitat pendent de rebre* de la fitxa dels productes que intervenen a la comanda, en el cas que la fitxa de producte contempli aquest camp (alguns ERP el contemplen).

La **recepció de la mercaderia** és un programa que ha de permetre introduir en el sistema informàtic la mercaderia que arriba a les nostres instal·lacions, fet que queda registrat en un document anomenat, normalment, albarà de compra i que ha de quedar associat al document que acompanya la mercaderia (albarà de venda del proveïdor). El programa ha de ser prou versàtil per permetre:

- Recepcionar només una part de la mercaderia que havia estat demanada en una comanda de compra (ja que pot ser que el proveïdor no enviï tot el que s'havia demanat) i efectuar el tancament de la comanda, malgrat no s'hagi servit tota la quantitat demanada o deixar la comanda parcialment servida.
- Recepcionar mercaderia que hagi estat demanada en diferents comandes de compra (al mateix proveïdor, és clar) i que el proveïdor la serveix en un mateix lliurament.
- Recepcionar mercaderia que no hagi estat demanada en una comanda de compra; aquesta situació no és gaire comuna i l'usuari que efectua l'entrada hauria de tenir un protocol d'actuació en aquest cas –algú l'hauria d'autoritzar.
- Localitzar qualsevol recepció de mercaderia efectuada a partir de l'identificador del document que l'acompanyava.

S'ha de tenir en compte que el programa de recepció de mercaderia modifica l'estoc dels productes afectats en el magatzem on s'està produint l'entrada i també

modifica el camp *quantitat pendent de rebre* de la fitxa dels productes recepcionats que provenien de la comanda, en cas que la fitxa del producte contempli aquest camp (alguns ERP el contemplen).

L'**entrada de factures de proveïdors** és un programa que ha de permetre, de forma molt ràpida, introduir una factura de proveïdor en el sistema informàtic. Ha de ser prou versàtil per permetre:

- Introduir una factura de despeses o d'immobilitzat sense necessitat d'haver introduït cap albarà previ.
- Introduir una factura de compra corresponent a un o diversos albarans de compra (del mateix proveïdor, és clar) ja introduïts en el sistema informàtic.
- Rebre factures electròniques.

Vendes

Aquest apartat comprèn els programes necessaris per cobrir el circuit de vendes: tarifes a clients, ofertes a clients, comandes de clients, lliurament de mercaderia i facturació.

En referència a les **tarifes de clients**, el programa hauria de poder contemplar:

- Tarifes i/o descomptes especials en un interval de dates (ofertes).
- Tarifes i/o descomptes especials en funció de la quantitat de producte, definit segons un escalat (a més quantitat, menor preu net o major descompte).

La **gestió d'ofertes a client** és un programa que ha de permetre introduir en el sistema informàtic una oferta a client per tal de, una vegada introduïda i enregistrada, fer-la arribar al client (via correu electrònic o per fax).

La **gestió de comandes de clients** és un programa que ha de permetre introduir en el sistema informàtic una comanda de client que pot haver arribat per diversos canals: telèfon, fax, correu electrònic, formulari d'una pàgina web... i que pot respondre a una oferta prèviament enviada al client.

A vegades, tant per les ofertes com per les comandes, els clients poden demanar la generació de l'anomenada **factura proforma**.

Factura proforma

Una factura proforma és un document basat en una oferta comercial amb la indicació exacta que tindrà la factura final. No té cap valor comptable ni com a justificant; s'utilitza principalment en comerç internacional per obtenir llicències d'importació, per obertura de crèdits documentaris o per enviaments de mostres comercials. Acostuma a incloure la data màxima de validesa.

A l'hora d'introduir la comanda de venda, l'ERP acostuma a proposar, per defecte, els valors que es tenen pactats amb el client, que es troben a la seva fitxa (*manteniment de tercers – pestanya de clients*). L'usuari simplement els haurà de validar.

S'ha de tenir en compte que el programa de gestió de comandes de client modifica el camp *quantitat pendent de servir* de la fitxa dels productes que intervenen a la comanda, en cas que la fitxa de producte contempli aquest camp (alguns ERP el contemplen).

Si la comanda del client s'ha rebut per telèfon i no n'ha quedat constància documental en la nostra organització és altament recomanable enviar-ne una còpia (per fax o correu electrònic) al client per sol·licitar-li la seva conformitat escrita. En canvi, si la comanda del client s'ha rebut amb un document del client, cal registrar a la nostra comanda l'identificador de comanda del client per facilitar-ne la localització davant de qualsevol incidència.

El **lliurament de la mercaderia** és un programa que ha de permetre generar les sortides de material cap a clients, per donar resposta als requisits de les comandes. Normalment el sistema, a partir de les dates de lliurament existents a les comandes de client i de les existències en el magatzem afectat, proposa una preparació de comandes (*picking*, en anglès) a servir, tot generant un informe que contempla tot el que es pot servir i també allò que no es pot servir. A partir d'aquí, algun responsable pren les decisions que calguin i el sistema ha de permetre, finalment, generar les sortides decidides. Cada sortida ha d'anar acompanyada del corresponent albarà de sortida, també anomenat albarà de venda i, si el port el realitza una agència de transport, és molt usual generar un albarà d'agència (s'ha de tenir present que l'albarà de venda conté una relació detallada dels productes –valorada o no– i l'agència de transport no té per què ser-ne coneixedora, sinó que només li cal saber el nombre de paquets, el pes i el volum).

El programa de lliurament de mercaderia modifica l'estoc dels productes afectats en el magatzem on s'està produint la sortida i també modifica el camp *quantitat pendent de servir* de la fitxa dels productes lliurats que provenien de la comanda, en cas que la fitxa del producte contempli aquest camp (alguns ERP el contemplen).

El **procés de facturació** és un programa que ha de permetre, de forma molt ràpida, la generació de les factures a client, sigui a partir de la comanda o de l'albarà de lliurament. Hi ha ERP que obliguen, per poder generar una factura, a disposar d'un albarà de lliurament de la mercaderia. Això suposa un mal de cap, ja que a vegades la factura cal generar-la una vegada la comanda de client ha estat acceptada, independentment de si la mercaderia ha estat o no lliurada. Així doncs, el procés de facturació ha de ser prou versàtil per permetre:

- Generar la factura a partir de la comanda amb l'obligatorietat o no d'haver servit la mercaderia (fet que ha de poder ser una característica de l'empresa per a tots els clients o configurable quant a client o tipologia de client o tipologia de comanda, etc.).
- Generar factura per comanda o poder agrupar diverses comandes en una factura o generar factura per una part d'una comanda (les parts servides, per exemple).
- Generar les factures que superin un determinat import, ja que a vegades no surt a compte, per les despeses de cobrament associades a una factura (girs bancaris, per exemple), generar factures d'import inferior a una determinada quantitat i és millor esperar que el client efectuï més despesa per agrupar en una sola factura diverses comandes del client.
- Generar factures electròniques.

- Contemplar diversos períodes de facturació (diari, setmanal, quinzenal, mensual...), ja que hi ha organitzacions que pacten, amb cada client, els períodes de facturació.

Fabricació

Un ERP, per definició, ha de permetre la gestió integrada de totes les àrees de l'empresa i en cas que l'empresa tingui processos de fabricació l'ERP n'ha de contemplar la seva gestió.

Els processos de fabricació són diferents en els diversos sectors productius i, en conseqüència, es fa difícil disposar d'un mòdul de fabricació que s'adapti a tots. Per aquest motiu, els fabricants d'ERP acostumen a facilitar **solucions específiques per a cada sector.** A tall d'exemple:

- Sector de la moda, sigui tèxtil o calçat, on és imperatiu poder gestionar paràmetres com temporades, talles o colors.
- Sector de l'alimentació, on és imprescindible la traçabilitat i el control de lots en totes les fases de producció.
- Sector de fabricació de maquinària.
- Sector d'arts gràfiques.

No és el nostre objectiu entrar en els processos de fabricació específics de cada sector. Podem, però, introduir els conceptes vinculats a un procés de fabricació bàsic consistent en l'obtenció d'un producte a partir d'un seguit de components, que poden ser adquirits a proveïdors com a primera matèria o bé ser fabricats prèviament a l'empresa. Els conceptes que cal conèixer són llista de materials, full de ruta i ordre de fabricació.

Una **llista de materials** (*bom* en anglès, de *bill of materials*), consisteix en una llista dels components necessaris per a l'obtenció del producte final.

En els **components** podem incorporar:

- Articles definits en el fitxer mestre de productes, que poden ser primeres matèries que adquirim a proveïdors o productes obtinguts en processos de fabricació interns.
- Mà d'obra dels operaris.

Els components que formen la llista van acompanyats de les quantitats necessàries per a la fabricació d'una determinada quantitat de producte final. En el cas que els components apareguin en quantitats molt petites, les quantitats es basen en la fabricació d'una quantitat superior a la unitat. La taula 1.1 mostra dos exemples de **llistes de materials**: una basada en 1 unitat de producte final i l'altra basada en 100 unitats de producte final.

Les llistes de materials de la taula 1.1 són molt simples; en realitat les llistes de materials acostumen a incorporar més dades, com per exemple:

- El codi de cada component, que hauria de ser obligatori, ja que la descripció pot no ser suficient per identificar el producte.
- La possibilitat d'indicar, per a cada component, si la quantitat necessària és fixa o és proporcional a la quantitat de producte final (cal pensar que a vegades, per una determinada fabricació, pot ser necessària una mà d'obra de preparació o uns materials de preparació, la quantitat dels quals no depèn de la quantitat de producte a fabricar).

Taula 1.1. Exemples de llistes de materials ('bom')

Producte: *Ordinador X* Quant.: *1u.*		**Producte: *Adob CKT* Quant.: *100 kg.***	
Component	**Quant.**	**Component**	**Quant.**
Font d'alimentació A	1 u.	TPF-II Granular G-900	69 kg.
Processador B	1 u.	Carbonat sòdic dens	30 kg.
Dissipador C	1 u.	Detergal G4 Blue	1 kg.
Placa base D	1 u.	Hores operari cat. 3	0,3 h.
Memòria RAM E	2 u.		
Disc dur SATA F	1 u.		
DVD G	1 u.		
Cable disc SATA	1 u.		
Caixa H	1 u.		
Cargols I	20 u.		
Mà d'obra muntador	0,75 h.		
Monitor J	1 u.		
Teclat K	1 u.		
Ratolí L	1 u.		
Cable alimentació	2 u.		

Un **full de ruta** (*rate routing*, en anglès) incorpora les diferents fases de fabricació d'un producte, amb les seccions o zones de la fàbrica que participen en cada fase i amb les operacions de producció a efectuar a cada fase.

Un **ordre de fabricació** és la concreció d'una fabricació d'un producte, amb la quantitat de producte a fabricar, la data de fabricació i la línia de producció a emprar.

Per **gestionar les ordres de fabricació** hi acostuma a haver tres processos:

1. Planificació de l'ordre, moment en què s'introdueix en el sistema la quantitat, la data i la línia de producció previstes. Aquest procés hauria de com-

provar, a partir de les dates de lliurament de les comandes de compravenda pendents de recepció o lliurament i de les dates de les ordres de fabricació planificades o en execució, la previsió d'existències dels components de l'ordre planificada, avisant de les possibles ruptures d'estoc.

2. Llançament de l'ordre, moment en el qual es reserven les quantitats necessàries per procedir a la fabricació de l'ordre. Si l'ordre havia estat planificada, canvia el seu estat de planificada a llançada.

3. Regularització de l'ordre, moment en el qual s'informa el sistema de la quantitat final de producte produït (que pot ser diferent de l'indicat en la planificació o llançament) així com les quantitats finals de productes consumits (que poden ser diferents dels previstos en la planificació o llançament) i hores d'operari emprades.

Serveis

Hi ha organitzacions en les quals el seu negoci està basat en els serveis; com per exemple, els serveis d'atenció tècnica (SAT), els serveis de consultoria, els serveis de gestoria... En aquestes situacions, les empreses necessiten disposar d'un mòdul de serveis que els permeti:

- Definir el servei amb les diferents fases, les hores d'operari de cada fase (amb l'assignació de l'operari concret o simplement de la categoria d'operari que haurà de dur a terme la fase) i, si s'escau, els materials necessaris.
- Efectuar un seguiment de les hores i materials emprats a cada fase.
- En els serveis de llarga durada, cal poder controlar el cost del servei a cada moment, per tal de detectar possibles desviacions respecte als costos previstos inicialment.

Comptabilitat i finances

A Internet es poden trobar molts materials introductoris referents a comptabilitat. A la secció "Annexos" del web, trobareu l'apartat "On adquirir coneixements de gestió empresarial?" amb alguna recomanació.

La presentació de les funcionalitats bàsiques dels ERP que fan referència a fitxers mestres, taules de suport, compres, vendes, producció i serveis es pot dur a terme utilitzant un llenguatge no gaire tècnic.

El mòdul de comptabilitat i finances ja no és tan fàcil d'introduir si no es tenen coneixements al respecte i no és l'objectiu d'aquest material introduir-lo. Els tècnics informàtics programadors que hagin d'adequar un ERP a les necessitats de l'empresa, desenvolupant-hi mòduls específics o utilitzant eines BI per obtenir informació per als responsables de l'empresa, han de tenir uns coneixements mínims de comptabilitat i finances per poder donar resposta a les necessitats que sorgeixin en aquest àmbit.

Una vegada coneguda la teoria de les principals funcionalitats que ens hauríem de trobar en un ERP, convé tenir un primer contacte amb els ERP actuals. A la secció "Annexos" del web trobareu l'apartat "Actualitat del programari de gestió empresarial", que presenta els principals ERP del mercat amb enllaços a vídeos en els quals se'n mostra el funcionament.

1.3.3 La llegenda de la implantació dels ERP

"If it's not broken, don't fix it" diuen els anglosaxons, en una frase que podria traduir-se a **'si funciona, no ho toquis'** i que en l'àmbit de la implantació d'ERP s'acostuma a sentir molt sovint.

Les empreses li tenen por, per no dir pànic, a un canvi en el seu programari de gestió empresarial, sigui o no ERP, i no els falta raó, ja que se sent parlar molt d'experiències negatives.

Les 10 raons que apareixen constantment com a **provocadores dels fracassos** de les implantacions d'ERP són:

1. Els processos de negoci de l'organització no han estat ben definits.
2. La implantació ha estat més llarga del que s'havia planificat.
3. Els costos de la implantació han estat més alts dels planificats.
4. Les activitats prèvies a la implantació van ser deficients.
5. El personal de l'organització no està capacitat.
6. La previsió d'utilització va ser massa ambiciosa.
7. No hi ha hagut una metodologia clara d'implantació.
8. La recepció d'informació o requisits per part dels usuaris no va ser completa.
9. No hi ha hagut el suport adequat per part dels responsables de l'organització.
10. No s'han gestionat adequadament les relacions interpersonals.

La implantació d'un ERP en una organització sobrepassa les responsabilitats dels tècnics que efectuen la implantació tècnica de l'ERP i dels programadors que l'adapten a les necessitats de l'organització, però tècnics i programadors es trobaran enmig d'implantacions i és convenient que tinguin coneixement de les bones pràctiques.

L'**anàlisi dels problemes** que provoquen el fracàs de la implantació d'un ERP ajuda a definir els punts a tenir en compte per aconseguir una bona implantació. Hi ha nombrosos estudis al respecte i, encara que tots volen el mateix (aconseguir una bona implantació), no tots defineixen el mateix nombre de punts a tenir en compte.

Com que el mot *decàleg*, a banda d'indicar 10 punts, connota un conjunt de punts bàsics per al desenvolupament d'una activitat, intentarem recollir en un decàleg adreçat als dirigents de l'organització on implantar un ERP els punts clau a tenir en compte:

1. **Començar a treballar amb temps**. En el moment en què es comença a intuir que el programari actual té deficiències que no es poden solucionar i que poden derivar en problemes greus, cal posar fil a l'agulla i començar la cerca d'un nou programari. Això implica analitzar les operacions de l'organització, la informació que es gestiona i els sistemes d'informació existents, amb els punts forts i els punts febles, documentant tot el procés. És altament recomanable que aquest procés l'efectuï algú extern a l'empresa, ja que l'experiència diu que el dia a dia no facilita que aquest estudi el desenvolupi gent interna.

2. **Escollir l'ERP adequat a l'organització.** Per fer-ho, cal cercar bé en el mercat i escoltar totes les opcions possibles, tant les de programari propietari com les de codi obert. L'organització que vol adquirir un ERP és especialista en el seu negoci i no pot pretendre ser-ho en ERP i, per tant, ha de confiar directament en els distribuïdors o en l'equip que hagi efectuat l'estudi del punt anterior. Convé avaluar, com a mínim, tres programaris alternatius, exigint una demostració específica per al nostre negoci i, per a cada programari i si és factible, convé avaluar dos distribuïdors diferents, valorant l'equip humà i el desplegament de medis que utilitzen en la implantació. És interessant considerar la possibilitat de mantenir els serveis de l'equip extern que hagi desenvolupat el punt anterior en tot el procés per tal que serveixi d'interlocutor entre l'organització i els distribuïdors.

3. **Esprémer al màxim la fase de tracte comercial**. En aquesta fase, l'empresa candidata a implantar l'ERP té total disponibilitat. Una vegada signat el contracte, malgrat que el tracte continuï sent correcte, s'ajusten a allò que s'ha signat i, en conseqüència, cal haver dedicat molt de temps a comprovar que les funcionalitats del programa s'ajustin als nostres requisits. En cas de detectar funcions essencials no suportades és altament recomanable cercar un altre programari que s'hi adeqüi millor. Cal tenir en compte que les adaptacions en un ERP són molt costoses i no sempre factibles i, per tant, és fonamental l'elecció de l'ERP adequat.

4. **Repassar molt bé el contracte, en especial l'abast del treball**. L'empresa implantadora acostuma a ser implacable a l'hora de facturar qualsevol cosa no prevista en el contracte i sempre tenen la paella pel mànec, ja que són els únics que saben la veritat del que hi ha al davant. Per això cal tornar a comentar que és molt interessant mantenir els serveis de l'equip extern que ha participat en el punt 1 com a interlocutor entre l'organització i l'empresa implantadora. El treball a desenvolupar ha d'incorporar, amb molt detall, els processos de formació de personal, punt molt important per aconseguir l'èxit de la implantació.

5. **Abans de signar, cal assegurar-se que la solució adquirida cobreix el 100% dels requisits**. De fet, això és conseqüència del que s'ha comentat en el punt 3, però a vegades hi ha funcionalitats cobertes per mòduls que es comercialitzen a banda i, és clar, ningú no ens ha enganyat perquè l'ERP ho cobreix a través d'un mòdul addicional; el problema apareix si no forma part del programari adquirit. En especial, cal tenir molt en compte

l'apartat relatiu a la intel·ligència de negoci (BI) per tal de poder accedir a la informació i generar informes i quadres de comandament.

6. **Disseny adequat del maquinari necessari**. La plataforma de maquinari sobre la qual s'ha de basar el funcionament informàtic de l'empresa és prou important per a dedicar-hi un estudi específic i valorar totes les solucions. Els departaments de sistemes de les empreses a vegades poden ser recelosos al canvi i sentir-se incòmodes amb noves plataformes que no dominen. Això no hauria de ser un problema si el canvi de plataforma ha de suposar un estalvi important i fiabilitat i rendiment iguals o millors.

7. **Solvència del procés d'implementació: equip i metodologia**. Cal conèixer la solvència de l'equip que durà a terme la implantació: qui formarà l'equip i quantes implantacions del programari han efectuat en empreses del mateix sector o amb funcionalitats similars. Així mateix és fonamental conèixer la planificació i metodologia que se seguirà i assumir-la per tal d'aconseguir l'èxit en el menor temps possible.

8. **Mínimes modificacions al programa**. Ja hem indicat abans que han de ser les mínimes indispensables i, a vegades, és preferible, si és possible, canviar l'operativa de l'empresa per adequar-la al funcionament del nou programari, abans que entossudir-se en unes modificacions que poden provocar problemes de rendiment i, fins i tot, problemes amb les actualitzacions del programari.

9. **Màxima atenció als usuaris**. Una implantació d'ERP pot suposar un xoc pels usuaris, que hauran de canviar de pantalles i, en molts casos, la forma de fer les coses. Per tant, cal aconseguir la màxima col·laboració dels usuaris, havent-los fet participar en els processos de preimplantació (anàlisi de les operacions que s'efectuen i informació que es gestiona i anàlisi dels productes candidats). Una vegada iniciada la implantació, han de rebre la formació i acompanyament adequats.

10. **Dedicació directiva a la implantació**. Durant el procés d'implantació, l'empresa ha de destinar al projecte recursos de primer nivell en termes de temps de l'alta direcció. És essencial un gerent de projecte de primera línia directiva, amb capacitat analítica, visió de negoci, resolutiu i amb interlocució en totes les àrees funcionals de l'empresa. És imprescindible la disponibilitat de la direcció general per l'adopció de decisions que li han d'arribar *mastegades* i, en conseqüència, és molt convenient el suport de recursos externs independents que aportin experiència i suport (els que havíem comentat en el punt 1 del decàleg i que haurien d'acompanyar-nos en tot el procés).

Aquest decàleg està adreçat als dirigents de l'organització en la qual s'ha d'implantar l'ERP. El tècnic informàtic que està llegint aquests materials no acostumarà a ser dirigent de l'organització, però convé que en sigui coneixedor, ja que:

- Pot formar part del departament TIC de l'organització on implantar l'ERP.
- Pot formar part d'un equip d'implantació de l'ERP.

- En petites empreses, pot haver esdevingut el cap del departament TIC i pot haver d'erigir-se en el responsable intern de la implantació.

Fixem-nos que els **punts del decàleg** es poden agrupar en tres fases: (1) Anàlisi, (2) Plantejament i disseny, i (3) Implantació. Un cop tenim l'ERP implantat i en funcionament amb total èxit cal passar a una quarta fase: (4) Postimplantació.

Les necessitats de les empreses evolucionen constantment i els ERP també ho fan. En conseqüència, a l'organització que ha implantat un ERP li convé **anar actualitzant-lo** a partir de les actualitzacions que facilita el fabricant. Això normalment s'articula a partir de contractes de suport o manteniment postimplantació amb l'empresa que ha efectuat la implantació.

El contracte de **suport o manteniment**, de pagament periòdic, pot incorporar:

- Conjunt d'hores de suport a preu zero o reduït.
- Per les hores que sobrepassin el conjunt anterior, descompte sobre el preu de venda.
- Accés als pegats i actualitzacions de l'ERP facilitats pel fabricant.
- Processos d'instal·lació de pegats i actualitzacions a preus especials.

Els ERP a les PIME

Fins fa uns anys, els grans fabricants d'ERP dirigien els seus productes a grans empreses i el mercat de les PIME quedava per a fabricants d'aplicacions de gestió (moltes vegades *suite*) que cobrien les necessitats de l'empresa sense que el seu producte pogués ser catalogat com un ERP. De fet, en parlar d'un ERP es tendeix a pensar en un sistema desenvolupat per a la gran empresa i amb un cost excessiu per a la PIME, tant en l'econòmic del producte com en el d'implantació.

Aquesta situació s'ha vist alterada en els darrers anys, en el quals els grans fabricants d'ERP han dirigit la seva mirada cap a les PIME i els ofereixen versions dels seus productes.

Podeu consultar el punt "Els ERP a les PIME", a la secció "Annexos" del web del mòdul.

1.4 Sistemes CRM i solucions BI, complements dels ERP?

Recordem les definicions de sistemes ERP, sistemes CRM i solucions BI:

- Els **sistemes ERP**, com a programari de gestió integrada, integren totes les dades i processos d'una organització en un sistema unificat.
- Els **sistemes CRM** donen suport a la gestió de les relacions amb els clients, a la venda i al màrqueting.

- Les **solucions BI** són eines destinades a facilitar dades als dirigents empresarials, obtingudes a partir de les dades dels sistemes ERP-CRM, amb l'objectiu de facilitar la presa de decisions.

Segons la definició d'ERP, aquests sistemes integren totes les dades i processos de l'organització i, en conseqüència, han d'incorporar la gestió de les relacions amb els clients (CRM) i podrien incorporar eines d'intel·ligència de negoci. Per tant, una organització amb ERP no s'hauria de plantejar la implantació de CRM i de solucions BI.

La majoria d'ERP actuals incorporen un mòdul de CRM que en alguns casos forma part de la base de l'ERP i en altres és un mòdul optatiu, un **CRM independent**. Però llavors, per què existeixen sistemes CRM que es comercialitzen independentment dels ERP? Qui els adquireix? Trobem la resposta en el fet que:

- Hi ha sistemes CRM que potser faciliten més funcionalitats que el mòdul CRM incorporat per l'ERP i l'organització precisa d'aquestes funcionalitats.
- Hi ha empreses que en lloc de tenir ERP disposen de diversos programes de gestió empresarial i els convé poder adquirir un CRM.

La implantació d'un CRM independent del programari de gestió comporta tenir dades duplicades en els dos sistemes (clients, ofertes, comandes, vendes, producte...) i, per minimitzar la duplicitat de l'entrada de dades i les incoherències, s'estableixen connexions amb la base de dades de l'ERP o del programari de gestió per tal d'alimentar la base de dades del sistema CRM.

Pel que fa a les solucions BI, els ERP actuals també incorporen eines que permeten obtenir informes per analitzar i que acostumen a formar part de la base de l'ERP. Però, per segons quin tipus d'informe o anàlisi a efectuar, és possible que el mòdul BI integrat a l'ERP encara no en faciliti l'adequada funcionalitat, tot i que molt probablement els ERP aniran evolucionant en la línia de la solució total. Així doncs, actualment és força usual adquirir una solució BI per obtenir resultats complementaris a la informació que facilita l'ERP.

Hi ha solucions BI que treballen directament amb la base de dades del programari de gestió comercial, però en moltes ocasions s'utilitza un **magatzem de dades** *(data warehouse)* on prèviament s'ha bolcat les dades a analitzar, en un format intel·ligent per facilitar les anàlisis previstes. Així, per exemple, per analitzar les vendes efectuades per tipus de producte i tipus de clients en els diferents mesos comparant els darrers tres anys, s'obtindrà uns resultats més ràpids si es disposa dels imports de venda agrupats per tipus de producte, tipus de client i mesos i anys, en lloc d'haver d'efectuar aquestes agrupacions cada vegada que es vol executar l'anàlisi comparativa. Certament, el fet de treballar amb magatzems de dades implica redundància de dades, ja que el seu contingut és calculable a partir de les dades existents en la base de dades del programari de gestió comercial, però l'estalvi de procés de dades és tan gran, que està àmpliament justificat.

Per tot això es pot respondre afirmativament a la pregunta que encapçala aquest apartat: els sistemes CRM i les solucions BI són companys de viatge dels ERP.

1.4.1 Funcionalitats dels sistemes CRM

L'acrònim CRM s'utilitza indistintament, per a dos conceptes:

1. CRM com a **estratègia de negoci** de l'organització focalitzada en el client, consistent en centrar els esforços en el coneixement dels clients, detectant les seves necessitats amb l'objectiu d'augmentar el seu grau de satisfacció, d'incrementar la fidelitat a l'organització i d'incrementar la rendibilitat o beneficis del client a l'organització.
2. CRM com a **sistema informàtic** ideat perquè l'organització pugui administrar tots els aspectes vinculats amb la gestió dels seus clients, de manera que un sistema CRM pot incloure de tot, des de tecnologia per recollir dades de les trucades telefòniques de l'àrea de vendes fins a llocs web on els clients tinguin accés als nostres productes (i quedi constància de les visites i del que hi han fet), incorporant-hi tota la informació provinent del circuit de venda del programari de gestió empresarial.

El nostre objectiu és conèixer el CRM com a aplicació informàtica, que ha de permetre assolir l'estratègia CRM adoptada per l'organització. Normalment, en un sistema CRM hi trobem els següents mòduls:

1. **Mòdul de clients**: permet introduir els clients de l'organització. Si el CRM forma part de l'ERP, el mòdul de clients coincideix amb el mòdul de l'ERP i, com a molt, incorpora més camps propis de la gestió del CRM, però no es produeix cap duplicitat de dades. En cas d'un sistema CRM independent, la situació més usual és que l'organització ja disposi d'un programari de gestió empresarial (sigui o no ERP) des d'on s'efectuen les vendes a clients i, en conseqüència, aquest mòdul suposa una duplicitat de dades, necessària per poder executar les funcionalitats que aporta el CRM. En aquestes situacions, per minimitzar la possibilitat d'errors i mantenir al dia els fitxers de clients d'ambdós programaris (gestió comercial i CRM), s'acorda gestionar els clients sempre a través d'un dels dos programaris i s'implementa un traspàs d'informació cap a la base de dades de l'altre programari, que s'hauria d'executar en temps real i, en el pitjor dels casos, automatitzar-ne l'execució a intervals regulars.
2. **Mòdul de clients potencials**: permet introduir les persones o organitzacions que representen alguna oportunitat de ser futurs clients.
3. **Mòdul de contactes**: permet gestionar les persones o organitzacions associades a un client (real o potencial) amb les quals l'organització es comunica amb la intenció de generar una oportunitat de negoci amb el client.

4. **Mòdul de productes**: permet gestionar els articles susceptibles de ser venuts. De la mateixa manera que amb el mòdul de clients, en el cas d'un sistema CRM independent es produeix una duplicitat amb els productes de l'aplicació de gestió empresarial de l'empresa.

5. **Mòdul de suport**: ha de permetre recollir tots els contactes entre l'organització i els clients (reals o potencials), sigui quin sigui el canal pel qual s'estableixin (telefònic, correu electrònic, fax, visita comercial, estand d'una fira, visita identificada al lloc web...), tot enregistrant els detalls del contacte i les possibles accions pendents d'executar arran del contacte, amb la data, el responsable i el contingut.

6. **Mòdul d'informes i gràfics**: per ajudar l'organització a obtenir informes personalitzats, per ajudar a prendre decisions oportunes de negoci. Aquest mòdul no deixa de ser una solució BI per al CRM.

Els CRM independents aporten, també, els mòduls que faciliten les accions pròpies del programari de gestió comercial i que són necessàries de controlar per poder tenir tota la informació al voltant dels clients. Per això, la llista de mòduls anteriors es pot veure ampliada amb:

1. **Mòdul d'ofertes**.

2. **Mòdul de gestió de comandes de venda**.

3. **Mòdul de gestió d'ordres de lliurament**.

4. **Mòdul de facturació**.

En cas de tenir implantat un sistema de gestió empresarial, de la mateixa manera que amb els clients i els articles, cal alimentar la base de dades del CRM amb la informació bàsica d'ofertes, comandes, enviaments i factures efectuades a través del sistema de gestió empresarial, per tal de disposar en el CRM de tota la informació i poder obtenir informes adequats.

Així doncs, per no veure'ns obligats a tenir duplicitat de dades a l'ERP i al CRM, s'imposa que els ERP incorporin el mòdul de CRM.

A la secció "Annexos" del web trobareu el punt "Presa de contacte amb sistemes CRM" que ens dóna a conèixer alguns dels productes CRM més utilitzats i hi ha presentacions que ens mostren les principals funcionalitats d'un CRM.

1.4.2 Funcionalitats de les solucions BI

Els sistemes ERP, CRM, HRM (*Human Resource Management*) són alguns dels innumerables tipus d'aplicacions implantades a les empreses, que es troben, en moltes ocasions, en plataformes diferents. A totes aquestes se sumen els documents impresos, arxius de diverses eines ofimàtiques, etc. cosa que converteix l'organització en un mar d'informació en el qual és difícil de trobar aquella que és determinant a l'hora de prendre decisions per al negoci. A vegades, pitjor que no tenir informació és tenir-ne molta.

La intel·ligència de negoci (BI) s'endinsa en la informació de l'organització amb l'objectiu de generar escenaris, pronòstics i informes que són subministrats als responsables de la presa de decisions.

Una aproximació de les àrees més comunes on s'apliquen les tècniques de la intel·ligència de negoci són:

- **Vendes:** anàlisi de vendes, detecció de clients importants, anàlisi de productes i tipus de productes, anàlisi de mercats, pronòstics i projeccions.
- **Màrqueting**: segmentació i anàlisi de clients, seguiment dels nous productes.
- **Finances**: anàlisi de despeses, rotació de cartera, raons financeres.
- **Fabricació**: productivitat de les línies de fabricació, anàlisi de residus, anàlisi de qualitat, rotació d'estoc, parts crítiques.

Per altra banda, en les organitzacions acostuma a existir una jerarquia que determina el tipus d'accions que es realitzen dins d'ella i, en conseqüència, el tipus de decisions que s'han de prendre. Tradicionalment s'han establert tres nivells jeràrquics:

1. **Estratègic**, en el qual la directiva decideix el camí que ha de seguir l'organització.
2. **Tàctic**, en el qual la gerència organitza i planifica les diverses àrees de l'empresa conjuntament amb els corresponents caps (màrqueting, vendes, finances, fabricació).
3. **Operatiu**, en els quals s'executen les operacions quotidianes de l'organització (diàries i rutinàries): operacions dels circuits de compravenda i fabricació i operacions comptables i financeres.

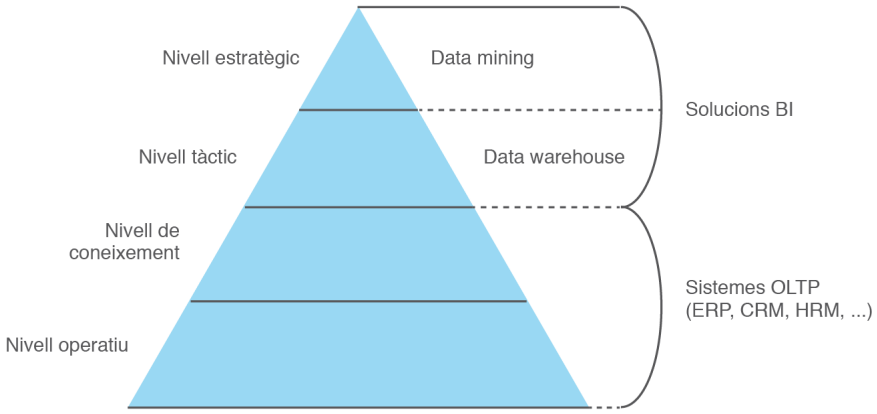
Aquest model tradicional de tres nivells s'ha vist ampliat darrerament per l'arribada de les TIC, amb un quart nivell que s'ubica entre el tàctic i l'operatiu, anomenat el nivell del **coneixement**, en el qual ubiquem tots els professionals que afegeixen valor a l'empresa per mitjà de les seves habilitats en les TIC.

Els diferents nivells, que també podríem anomenar rols, tenen diferents necessitats d'accés a les dades (el director general no té per què conèixer com s'introdueix en el sistema una oferta a client i en canvi sí que pot necessitar conèixer si s'està assolint els objectius de vendes per a l'exercici actual, mentre que la situació és totalment inversa per a un auxiliar administratiu del departament comercial). Els actors de tots els nivells necessiten informes, però la complexitat d'elaboració és molt diferent (l'auxiliar del departament comercial pot necessitar un simple llistat de les ofertes diàries, mentre que el director general necessita gràfiques que pugui visualitzar des de diferents dimensions). Per tant, es necessiten eines

informàtiques per elaborar informes adequats per a tots els nivells i la complexitat de les eines és molt diferent segons el nivell al qual han de servir.

La figura 1.5 mostra la correspondència entre els nivells jeràrquics d'organització d'una empresa i els tipus de sistemes de gestió de la informació normalment emprats, tenint en compte les necessitats d'informació de cada nivell. El contingut de la figura 1.5 no s'ha de prendre al peu de la lletra; és a dir, els actors dels nivells estratègic i tàctic poden utilitzar informes facilitats pels sistemes OLTP i els actors del nivell del coneixement poden també utilitzar algun informe proporcionat per les eines BI externes als sistemes OLTP.

Figura 1.5. Correspondència entre els nivells de l'empresa i els tipus de sistemes de gestió de la informació



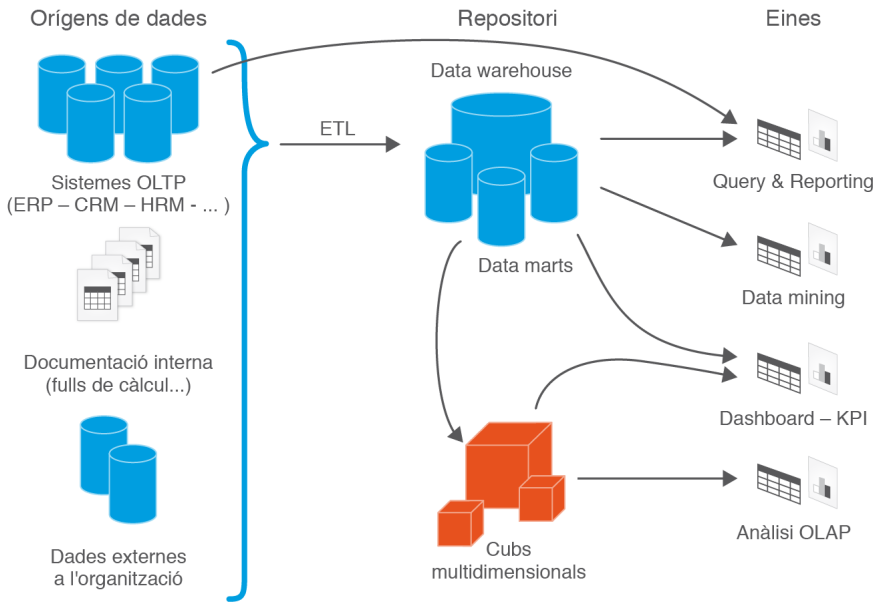
La figura 1.5 incorpora conceptes (OLTP, *data mining, data warehouse*) que hem de reconèixer juntament amb d'altres que estan vinculats al món BI: ETL, OLAP, KPI, *data mart, dashboard* i cubs multidimensionals.

Una eina BI ha de ser capaç de reunir informació dispersa per tota la companyia i, fins i tot, de diferents fonts, per tal de proporcionar als departaments l'accessibilitat, poder i flexibilitat necessaris per analitzar la informació. La figura 1.6 mostra tots els components que poden intervenir en una solució BI. La part esquerra de la figura mostra els diversos orígens de dades d'on pot provenir la informació que la solució BI reunirà en el repositori de la solució.

OLTP és l'acrònim anglès de **Procés de Transaccions En Línia** (*OnLine Transaction Processing*) per fer referència als sistemes que faciliten i administren aplicacions transaccionals, com és el cas dels ERP-CRM en els quals contínuament s'efectuen transaccions.

El **repositori** de la solució BI és el lloc centralitzat on la solució BI emmagatzema les dades recollides dels diversos orígens de dades, principalment de sistemes OLTP. En el repositori d'una solució BI hi podem distingir dos tipus de components: *data warehouse* (sempre present) i cubs multidimensionals (presents si la solució BI facilita l'anàlisi OLAP).

FIGURA 1.6. Components d'una solució BI completa



Un ***data warehouse*** (magatzem de dades) és una base de dades destinada a contenir una col·lecció de dades orientada a un determinat àmbit (empresa, organització, matèria...), integrada, no volàtil i variable en el temps, que ha de servir de base per a l'aplicació d'eines analítiques amb l'objectiu d'obtenir informació útil per a la presa de decisions. És a dir:

- **Orientada a un àmbit:** les dades contingudes estan organitzades de manera que tots els elements relatius a un mateix esdeveniment del món real queden relacionats.
- **Integrada:** conté les dades de tots els orígens de dades possibles, de forma consistent.
- **No volàtil:** la informació introduïda no es modifica ni elimina, és informació de només lectura que es manté per a futures consultes.
- **Variable en el temps:** els canvis produïts en les dades al llarg del temps hi queden registrats per tal que els informes puguin reflectir les variacions.

Un dels principals problemes a l'hora d'implementar un *data warehouse*, radica en el fet que les dades a integrar, en provenir d'orígens diversos, presenten inconsistències en format i codificació i això implica la necessitat de dissenyar un procés de filtratge, reestructuració de les dades i eliminació d'inconsistències abans de ser emmagatzemades en el *data warehouse*. Aquest procés és conegut com a **ETL**, acrònim de ***Extract, Transform and Load***.

La taula 1.2 mostra les principals diferències entre les bases de dades dels sistemes OLTP, dedicades a les operacions del dia a dia, i un *data warehouse*, dedicat a concentrar informació completament orientada a l'anàlisi.

Taula 1.2. Comparació entre les BD dels sistemes OLTP i els data warehouse

BD de sistemes OLTP	Data warehouse
Dades operacionals	Dades del negoci rellevants per informació
Orientada a les aplicacions	Orientat a l'analista
Dades actuals	Dades actuals + dades històriques
Dades al detall	Dades resumides amb cert detall
Canvia constantment	Estable

El *data warehouse* pot estar organitzat en *data marts*. Un ***data mart*** (aparador de dades) és un subconjunt de dades del *data warehouse*, corresponent a una unitat de negoci (àrea) de l'organització. Té l'objectiu de solucionar la problemàtica d'anàlisi de la corresponent àrea.

Un *data warehouse* es pot considerar com la col·lecció de *data marts* implementats en les diferents àrees de negoci de l'organització.

Les solucions BI aporten eines analítiques i la potència d'una solució BI es mesura a partir del nombre d'eines analítiques que facilita i de la potència de cadascuna d'elles.

Avui en dia les eines analítiques es tipifiquen en: *query&reporting, data mining,* KPI i *anàlisi OLAP*.

Les eines ***query&reporting*** (consultes i informes) són les tradicionals eines que permeten dissenyar i executar consultes sobre una base de dades i formatar el resultat en informes. La figura 1.6 mostra que aquestes eines s'apliquen sobre les bases de dades dels sistemes OLTP i sobre el *data warehouse* i *data marts*.

La majoria de sistemes OLTP (ERP, CRP...) faciliten eines *query&reporting* de fàcil aprenentatge que un usuari avantatjat pot utilitzar per dissenyar els informes que necessita i que no estan predefinits en el sistema.

Query&reporting ofimàtic

Les bases de dades ofimàtiques s'utilitzen en moltes ocasions com a eines query&reporting dels sistemes OLTP i data warehouse, a través de la connexió ODBC amb les BD del sistema OLTP o data warehouse.

Les eines ***data mining*** (mineria de dades) són eines d'alt nivell que sobrepassen l'objectiu d'aquest material. A títol informatiu cal saber que la mineria de dades consisteix en l'extracció no trivial d'informació que resideix de manera implícita en les dades, que era prèviament desconeguda i que pot resultar útil per algun procés. En altres paraules, la mineria de dades prepara, sondeja i explora les dades per obtenir informació oculta en elles.

OLAP és l'acrònim anglès de **Procés Analític en Línia** (*OnLine Analytical Processing*) per fer referència als sistemes que emmagatzemen grans quantitats de dades resumides obtingudes a partir de sistemes OLTP, amb l'objectiu d'efectuar-ne consultes.

El concepte OLAP va molt lligat al concepte *data warehouse* i a vegades es confonen. La diferència radica en el fet que *data warehouse* és un terme que s'utilitza per fer referència a les dades i OLAP és un concepte que s'utilitza per fer referència a les eines disponibles per avaluar i analitzar les dades dels *data warehouse*.

En parlar d'anàlisi OLAP apareixen els cubs multidimensionals o cubs OLAP o hipercubs. Un **cub multidimensional** és una representació matricial (N dimensions) de les dades planes representades via files i columnes en una taula relacional, utilitzat en l'anàlisi OLAP.

Exemple simplificat de construcció de 'data warehouse' i hipercub

La base de dades d'un ERP (suposem BD relacional) segurament té una taula on s'enregistren les vendes que s'efectuen. Suposem el disseny següent:

```
VENDA (#Client, #Producte, #Data, Quantitat, PreuUnitari)
ON {Client} REFERENCIA CLIENT
I {Producte} REFERENCIA PRODUCTE
```

Els dissenyadors del *data warehouse* han decidit que a nivell d'anàlisi no interessa mantenir el client, ni el producte ni la data, però sí que es necessita incorporar el tipus de client, la família de producte i el mes i any en què s'ha efectuat les vendes. Per tant, en el *data warehouse* s'ha dissenyat la taula següent, que agrupa les quantitats i la mitjana dels preus de venda:

```
VENDA_DW(#TipCli, #FamPro, #MesAny, SumQuantitat,
AVGPreu)
```

El procés ETL que emplena la taula `VENDA_DW` es preocupa de cercar totes les vendes del període que correspongui, agrupant-les per tipus de client, família de producte i mes o any, tot sumant les quantitats de producte venudes i calculant la mitjana dels preus aplicats.

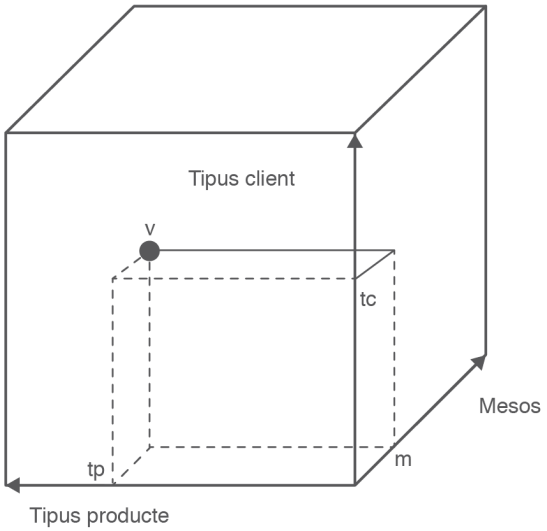
Amb aquest disseny, dins el *data warehouse* s'ha perdut la informació de detall de client, producte i data de venda. És a dir, s'ha disminuït la granularitat i, en conseqüència, l'anàlisi basada en el *data warehouse* podrà donar resultats pel que fa al tipus de producte, tipus de clients i intervals mensuals, però no pas a nivell de client, de producte i de data de venda. Si en el *data warehouse* s'hagués decidit emmagatzemar les dades en una estructura similar a la de la taula `VENDA` del nostre ERP, l'eina d'anàlisi tindria majors possibilitats analítiques, ja que podria analitzar les dades a nivell de detall i també pel que fa al resum que facilita `VENDA_DW`, però per fer això és necessari més espai en el *data warehouse*.

En terminologia de BI, la taula `VENDA` és una taula de fets (enregistra els fets que s'han produït) i la taula `VENDA_DW` és una taula agregada de fets. Les anàlisis a nivell resum s'executaran més ràpidament si disposem, en el *data warehouse*, de taules agregades de fets adequades al resum que cal analitzar. No és gens senzill decidir quines dades s'emmagatzemen en el *data warehouse* i amb quin nivell de granularitat.

Les dades de la taula `VENDA_DW` ens permeten construir diversos cubs multidimensionals, en els quals els atributs per analitzar es representen en els diversos eixos (dimensions) del cub.

Així, si volem analitzar la quantitat de vendes per tipus de producte, tipus de client i mesos, podem construir el cub tridimensional de la figura 1.7. Observem que en el punt `v` hi haurà el valor corresponent a la quantitat de venda del tipus de producte `tp` efectuada per clients del tipus `tc` en el mes `m`.

FIGURA 1.7. Exemple de cub OLAP tridimensional



És molt comú que la informació del *data warehouse* s'estructuri en cubs multidimensionals, ja que aquests preparen la informació per respondre a consultes dinàmiques amb un bon rendiment (temps de resposta). Els cubs multidimensionals no són, però, les úniques estructures de dades que utilitzen els *data warehouse.*

Per tal de facilitar el disseny de consultes OLAP, a causa que el llenguatge SQL obligava a escriure consultes complexes, es va crear el llenguatge MDX (*MultiDimensional Expressions*) que està pensat específicament per a efectuar consultes sobre cubs OLAP i, per tant, les consultes són molt més simples que les corresponents en el llenguatge SQL. El llenguatge MDX ha estat acollit per la majoria de proveïdors d'eines OLAP.

Per finalitzar amb la percepció dels conceptes més utilitzats al voltant de les solucions BI, apareguts tots ells en la figura 1.6, ens manca presentar els *dashboards* i els KPI.

KPI és l'acrònim anglès d'**Indicadors Claus d'Acompliment** (*Key Performance Indicators*) per fer referència a mètriques utilitzades per quantificar els objectius que reflecteixen el rendiment d'una organització i que generalment es recullen en el seu pla estratègic.

A la secció "Annexos" del web trobareu el punt "Presa de contacte amb solucions BI" que ens dóna a conèixer alguns dels productes BI més utilitzats i ens facilita presentacions que ens mostren les principals funcionalitats d'una solució BI.

Els responsables de l'organització tenen per dogma que "no es pot millorar allò que no es pot mesurar". En conseqüència, l'organització defineix el conjunt de KPI importants per a la seva evolució i per fer-ne un correcte seguiment es fa necessari disposar de quadres de comandament o *dashboards*.

Un ***dashboard*** és un tipus d'interfície interactiva d'usuari, dissenyada per proporcionar a l'usuari informació específica relativa a l'estat de l'empresa, representada normalment a través d'indicadors clau d'acompliment (KPI) i enllaços a informes rellevants. Existeixen senyals visuals, gràfics i controls de procés que centren l'atenció de l'usuari en les tendències, canvis i excepcions importants.

Hem d'imaginar un *dashboard* com un gran tauler de l'organització, on hi ha indicadors (com el tauler d'un vehicle) que mostren la realitat de les diferents

àrees de negoci. Imaginem que quan un valor d'un indicador baixa per sota d'un límit normal, s'encén una llum d'alerta que indica que cal posar-hi atenció i, si s'excedeix d'un valor tolerable, no només s'encén la llum sinó que a més ho indica mitjançant un senyal auditiu.

2. Implantació tècnica de sistemes ERP-CRM: Odoo

Quan es parla de sistemes ERP-CRM el mot *implantació* s'acostuma a utilitzar per fer referència al procés global que té com a objectiu final la posada en marxa d'un nou sistema ERP-CRM a l'organització. Aquest procés té diverses fases: anàlisi de requisits, estudi de possibles solucions, decisió per un producte, instal·lació i configuració, migració de dades -si s'escau-, formació dels usuaris i execució d'adaptacions -si s'escau-. Una implantació entesa així és un procés molt complex que ha d'estar dirigit per professionals especialistes (consultors).

Per **implantació tècnica** de sistemes ERP-CRM entenem el subprocés amb les següents operacions imprescindibles:

1. Instal·lació del programari, en un determinat maquinari i sistema operatiu, seguint les prescripcions del fabricant.
2. Instal·lació de mòduls addicionals que corresponguin, segons requeriments.
3. Configuració del programari, segons les possibilitats que es facilitin, per adequar-lo als requeriments de l'organització.
4. Verificació del correcte funcionament, segons requeriments.
5. Documentació de les operacions realitzades i incidències aparegudes, amb la seva resolució.

La llista d'operacions anteriors es pot veure ampliada amb **dues operacions més**:

- (**6**) Migració de dades del programari de gestió empresarial existent cap al nou programari, si així està contemplat en el projecte d'implantació del nou programari.
- (**7**) En cas que el programari que instal·lem incorpori un mecanisme de salvaguarda de les dades, posada en marxa i verificació de la correcta recuperació de les dades en el cas d'haver ocorregut un desastre.

Per executar aquestes operacions pot ser necessari l'ajut de dos **perfils professionals** diferents del nostre i que, en moltes ocasions, recauen en un mateix professional:

- L'administrador del sistema, en el cas que calgui efectuar alguna configuració pel que fa al sistema operatiu.
- L'administrador del sistema gestor de base de dades, en el cas que el nostre programari hagi de connectar amb un SGBD corporatiu, ja instal·lat o que calgui instal·lar.

Tot procés d'implantació tècnica d'un ERP-CRM consta de les operacions anteriors. En les diverses combinacions possibles de sistema ERP-CRM amb sistema operatiu i amb sistema gestor de bases de dades, hi intervenen tantes variables que cada combinació necessitaria el seu propi aprenentatge. És a dir, a diferència del procés d'aprendre a conduir, que ens serveix gairebé per a qualsevol cotxe, la implantació tècnica d'un sistema ERP-CRM no és estàndard i, per tant, un implantador necessita l'**aprenentatge concret** per a cada combinació possible. Per altra banda, l'experiència acumulada d'un implantador en diverses situacions ajuda que cada nova situació necessiti un aprenentatge més curt.

2.1 Odoo ERP

L'Odoo és un sistema ERP-CRM de codi obert.

Ja que no podem pretendre endinsar-nos en la implantació tècnica de qualsevol sistema ERP-CRM en qualsevol de les plataformes suportades (sistema operatiu + SGBD), ens centrarem en un producte i en la implantació hi posarem en pràctica totes les operacions. El producte escollit ha estat el sistema ERP-CRM de codi obert Odoo.

Odoo és un sistema ERP-CRM potent i actual, on conviuen una versió de codi obert (Odoo community) i una amb llicència (Odoo enterprise). Nosaltres, a partir d'ara, quan parlem d'Odoo ens referirem a Odoo community. Odoo era conegut fins a la versió 7 com a OpenERP, i prèviament com a TinyERP.

Llicència LGPL

La llicència LGPL és una llicència *copyleft* derivada de la GPL de GNU, però que permet que programes propietaris usin les llibreries i aplicacions sota aquesta llicència. L'LGPL permet l'ús de programes lliures amb programari propietari. El programa, per si mateix, es redistribueix com si estigués sota la llicència GPL, però es permet integrar-la a qualsevol altre programari gairebé sense cap limitació.

2.1.1 Característiques bàsiques

Les característiques bàsiques que hauríem de tenir en compte a l'hora d'avaluar un sistema ERP-CRM i els valors per al sistema Odoo són:

- **Llicència**:
  - LGPLv3.
- **Desplegaments possibles**:
  - On-premise sota els SO: Linux i Windows
  - SaaS.
  - odoo.sh és una plataforma per a desenvolupadors, que fa el *hosting* i permet de manera molt senzilla crear nous mòduls, provar-los, i portar-los a producció.
- **Arquitectura**. Client-servidor amb tres capes clarament diferenciades:
  - La base de dades en l'SGBD PostgreSQL, que només conté dades i no conté cap lògica de negoci (és a dir, no incorpora funcions, procediments o disparadors).

  - El servidor Odoo que conté tota la lògica de negoci amb un nucli base i una estructura que permet anar afegint mòduls segons les necessitats de l'organització. Odoo facilita una llarga llista de mòduls que es pot ampliar amb el disseny de mòduls propis per donar cobertura a les necessitats de l'organització.
  - La capa dels clients consisteix en un client web, accessible des de qualsevol navegador.

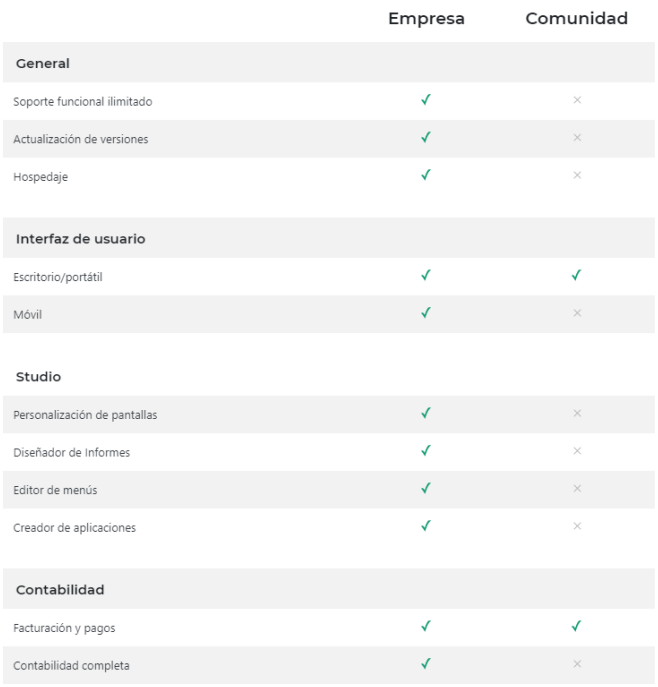
- **Sistema operatiu**:
  - Servidor sobre Windows o Linux.
- **SGBD**: PostgreSQL (el fet que la BD no contingui cap lògica de negoci fa pensar que hauria de ser fàcil la substitució de PostgreSQL per un altre SGBD).
- **Cost per a la versió Enterprise**:
  - Pot trobar-se una simulació a: bit.ly/31sXlWk. Com es pot comprovar els preus depenen del nombre d'usuaris i els mòduls que es trien (figura 2.1).

FIGURA 2.1. Preus d'Odoo

Respecte a les funcionalitats segons les versions, la **versió Enterprise** (de pagament) incorpora entre d'altres (figura 2.2):

- Llicència «Odoo Enterprise Edition License v1.0»
- Suport funcional il·limitat
- Actualització de versions
- Hosting
- Interfície d'usuari mòbil
- Odoo studio (dissenyar de manera gràfica mòduls, informes, pantalles...)
- Mòduls exclusius que amplien els estàndards.

FIGURA 2.2. Funcionalitats enterprise vs. community



	Empresa	Comunidad
General		
Soporte funcional ilimitado	✓	×
Actualización de versiones	✓	×
Hospedaje	✓	×
Interfaz de usuario		
Escritorio/portátil	✓	✓
Móvil	✓	×
Studio		
Personalización de pantallas	✓	×
Diseñador de Informes	✓	×
Editor de menús	✓	×
Creador de aplicaciones	✓	×
Contabilidad		
Facturación y pagos	✓	✓
Contabilidad completa	✓	×

Escollir el servidor

Ja que el servidor Odoo pot instal·lar-se sobre Windows i Linux, serà important decidir en quins tipus de servidor es porta a terme la instal·lació, segons les necessitats. També hi ha l'opció de fer servir el sistema de contenidors Docker, per a una instal·lació per a tasques de desenvolupament o per a revisió de noves versions.

2.1.2 Implantació tècnica d'Odoo

Les distribucions *All-In-One* (tot en un) incorporen totes les peces imprescindibles per poder instal·lar un servidor Odoo, incloent-hi, fins i tot, una versió de l'SGBD PostgreSQL. Com que se suposa que no som experts en instal·lació i configuració de l'SGBD PostgreSQL i, ja que se'ns facilita la distribució All-In-One que inclou la instal·lació d'una versió de PostgreSQL, la lògica ens diu que comencem per instal·lar aquesta distribució en un sistema operatiu conegut (Windows).

L'itinerari de **passos a seguir**, amb l'objectiu final de tenir els coneixements adequats per poder efectuar implantacions tècniques d'Odoo, és:

1. Instal·lar Odoo All-In-One en el SO Windows.
2. Adquirir coneixements bàsics del servidor PostgreSQL.
3. Instal·lar Odoo en SO Windows connectant amb un SGBD PostgreSQL existent.
4. Instal·lar Odoo en sistemes operatius Linux.
5. Instal·lar Odoo sobre un contenidor Docker.
6. Configuració inicial d'Odoo. Instal·lació de mòduls, configuració empresa.
7. Empleats vs. usuaris en odoo. Importació massiva.
8. Tercers en Odoo. Mòdul de compres i vendes. Inventari.
9. Web en Odoo. Creació de la web d'una empresa i botiga *online*.

2.2 Instal·lar Odoo

Hi ha diferents maneres d'instal·lar Odoo. El propòsit d'aquesta instal·lació és molt variat, des de la seva prova a una instal·lació en un entorn de producció, passant per la creació d'una màquina de desenvolupament de mòduls personalitzats. Igual que existeixen molts propòsits, existeixen moltes maneres d'instal·lar. Destaquem **tres tipus instal·lacions** diferents:

- Instal·lació del paquet *All-In-One* para Windows. La més senzilla, i amb propòsit de conèixer i provar el *software*.
- Instal·lació del codi font d'Odoo sobre un host Ubuntu Server. Aquesta seria una instal·lació amb l'objectiu de fer servir la màquina en un entorn de producció definitiu. Es podran conèixer a fons tots els elements que componen Odoo.
- Instal·lació d'Odoo sobre un contenidor Docker. Aquest tipus d'instal·lació està destinada a tasques de desenvolupament de codi o de prova de noves versions. Docker és una tecnologia actual que val la pena conèixer.

2.2.1 Instal·lació d'Odoo 'All-In-One' en el SO Windows

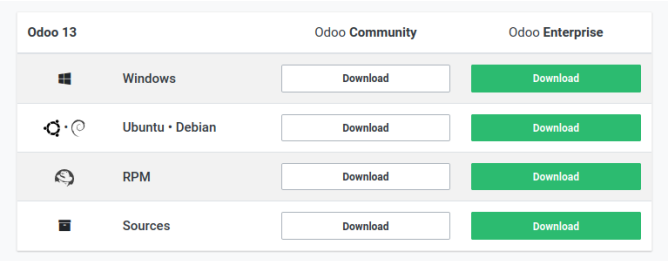
La instal·lació d'Odoo per a un SO Windows és la més senzilla, ja que incorpora en un senzill arxiu executable tots els components necessaris per a posar en marxa el servidor i el SGBD. És la més adequada per a fer un "tastet" del programari de manera ràpida. No és recomanable aquest tipus d'instal·lació, però, per a un servidor en producció.

Versions d'Odoo

Totes les versions d'Odoo community poden trobar-se a bit.ly/2IVxbF9.

A continuació es procedirà a instal·lar l'última versió d'Odoo publicada a la data de realització d'aquest document, encara que la instal·lació és totalment compatible amb altres versions d'Odoo (figura 2.3)

FIGURA 2.3. Descàrrega d'Odoo



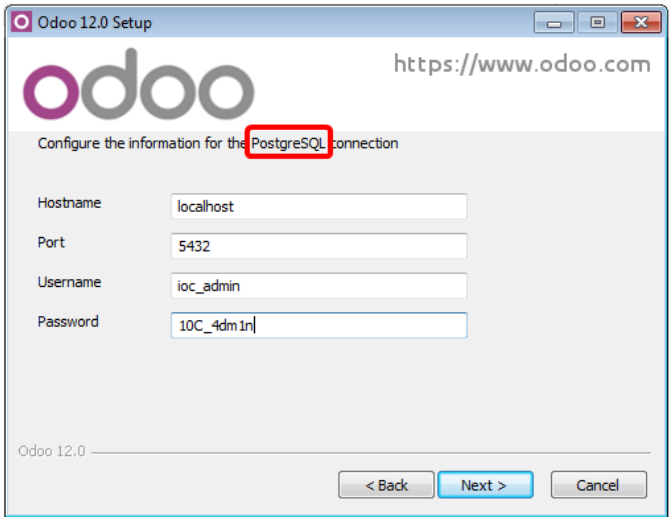
El procés proposa la instal·lació *All-In-One* amb Odoo Server (que incorpora el servidor Odoo i el client web) i PostgreSQL Database (necessari si no disposem ja d'un servidor PostgreSQL instal·lat). En aquesta primera instal·lació, mantindrem

els dos components (no podem tenir un SGBD PostgreSQL instal·lat a la màquina). Procedim, doncs, amb la instal·lació (figura 2.4)

FIGURA 2.4. Elecció dels components a instal·lar

Immediatament, se'ns facilita la pantalla de configuració per la connexió PostgreSQL, amb uns valors per defecte (figura 2.5).

FIGURA 2.5. Configuració del servidor PostgreSQL



En aquests moments aniria molt bé tenir uns mínims coneixements d'administració de SGBD i, si no pot ser, de connectivitat amb SGBD. El mínim que hem de saber és que la majoria de connexions a efectuar contra un SGBD utilitzen el **protocol TCP/IP** i, en conseqüència, necessitem conèixer:

- El nom o adreça IP de la màquina on hi ha instal·lat l'SGBD:
- El port TCP pel qual està escoltant l'SGBD.
- L'usuari i la contrasenya de l'usuari autoritzat a establir connexió.

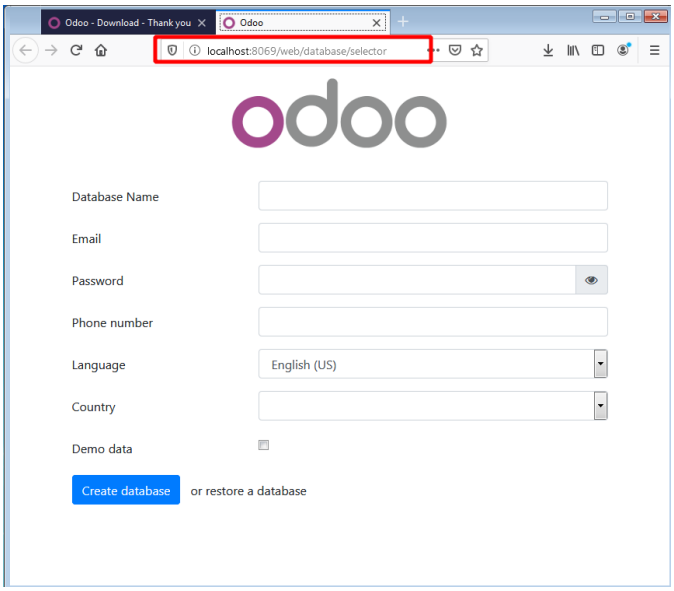
Com que el procés *All-In-One* instal·la el servidor PostgreSQL a la mateixa màquina on instal·lem el servidor Odoo, ja és correcte mantenir 'localhost' com a

valor per Hostname. Pel que fa al port (5432), cal saber que aquest és el port TCP pel qual normalment escolta un servidor PostgreSQL i, per tant, en mantindrem el valor. Pel que fa a l'usuari (openpg) que proposa el procés d'instal·lació, cal saber que aquest serà el superusuari de l'SGBD PostgreSQL que estem instal·lant. Podeu deixar l'usuari que proposa el procés, però també podeu canviar-lo segons les vostres preferències. Nosaltres el canviem i li assignem ioc_admin, amb contrasenya 10c_4dm1n. Aquesta informació és crítica i, per tant, cal tenir-la anotada en un lloc ben segur.

Després d'uns minuts, el procés finalitza. Veurem, a més a més, que instal·la Microsoft Visual C++. El mateix procés, en la darrera pantalla, ens facilita una casella de verificació anomenada Start Odoo per començar a treballar amb Odoo immediatament, a través d'una connexió HTTP.

Si manteniu l'opció activada i finalitzeu el procés, veureu com s'obre el navegador que tingueu per defecte i intenta connectar a l'URL `localhost:8069/web` com mostra la figura 2.6. Fixem-nos que el servidor web que instal·la Odoo escolta pel port 8069.

FIGURA 2.6. Servidor Odoo funcionant correctament



A la secció "Annexos" del web trobareu el punt "Recursos de programari" que inclou les versions dels programes als quals fem referència en aquests materials.

Una vegada finalitzada la instal·lació hi ha la possibilitat d'accedir-hi mitjançant un navegador i el port 8069. Altres característiques importants d'aquesta versió per a Windows fan referència a:

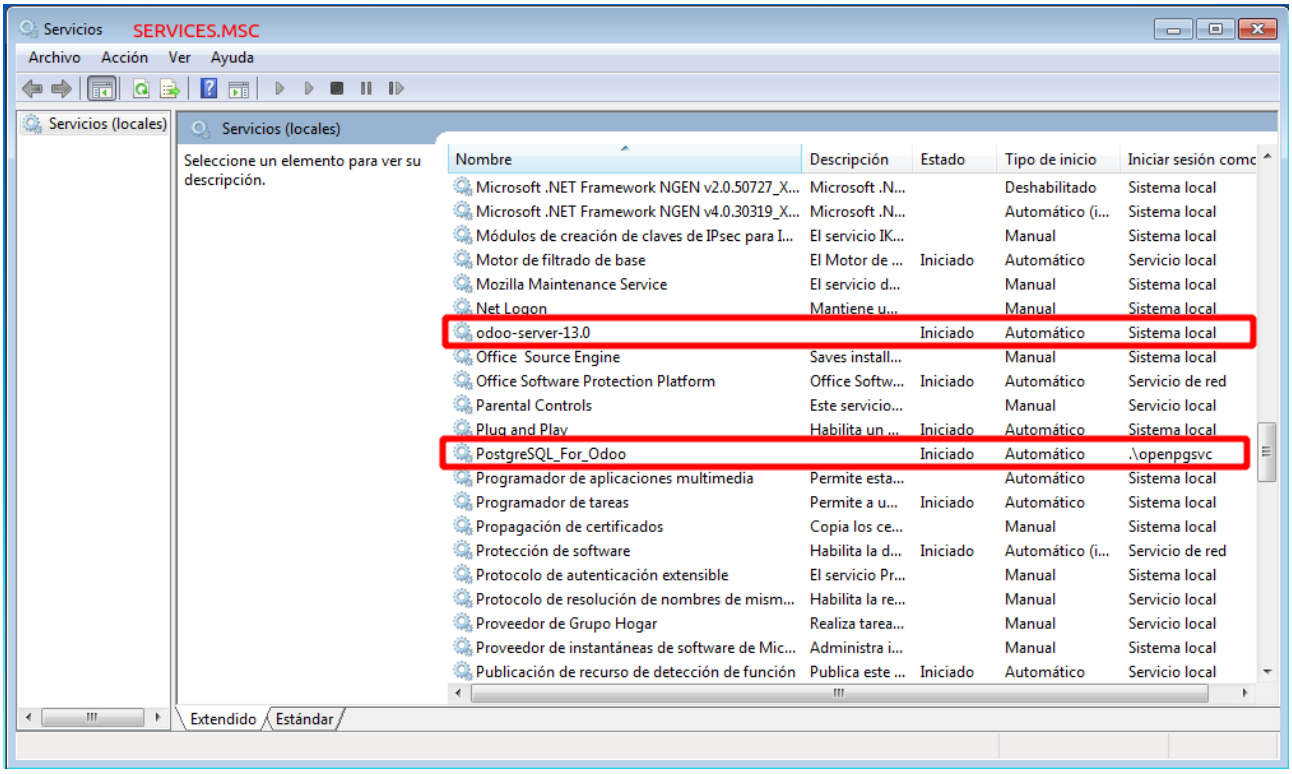
- Serveis
- Menú PostgreSQL
- Carpeta de la instal·lació

Serveis

Accedint al panell de control dels serveis del sistema operatiu (services.msc), poden observar-se dos serveis, engegats i amb inici automàtic (és a dir, es posen en marxa de forma automàtica quan s'engega la màquina; figura 2.7):

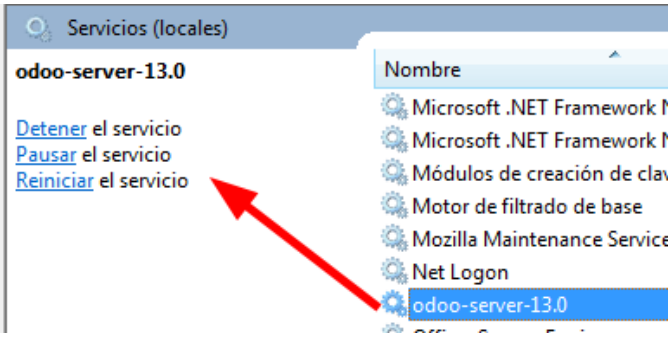
- PostgreSQL for Odoo
- odoo-server

Figura 2.7. Serveis creats al voltant d'Odoo



Des d'aquesta finestra podrem iniciar, aturar i reiniciar aquests dos serveis (figura 2.8).

Figura 2.8. Treballar amb els serveis

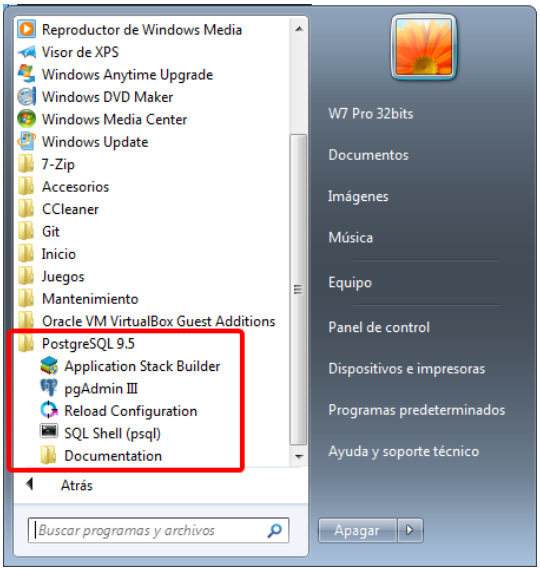


Menú PostgreSQL

A l'arbre de programes de Windows, hi observem el submenú PostgreSQL 9.5, amb diverses opcions per configurar i administrar l'SGBD PostgreSQL.

La instal·lació inclou el client pgAdminIII, que serà molt útil per l'accés a la base de dades i operacions bàsiques de consulta i administració (figura 2.9).

FIGURA 2.9. Menú PostgreSQL creat per Windows



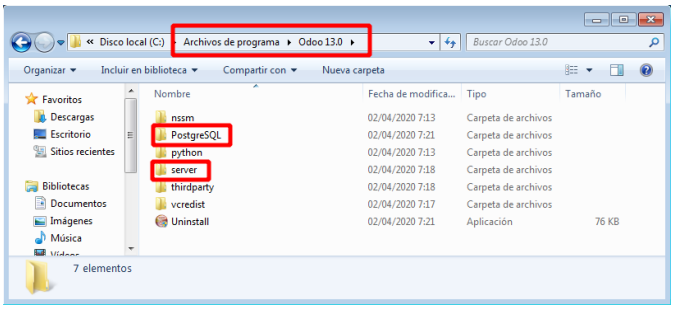
Tant PostgreSQL com el client pgAdminIII seran tractats en el punt "Coneixements bàsics del servidor PostgreSQL" d'aquest mateix contingut.

Carpeta de la instal·lació

A la carpeta del sistema d'arxius on s'ha instal·lat Odoo (possiblement C:\Archivos de programa\Odoo xx), poden trobar-se diverses carpetes. Destaquen (figura 2.10):

- **PostgreSQL**, que conté tot el que fa referència al servidor PostgreSQL.
- **server**, que conté tot el que fa referència al servidor Odoo (client web inclòs).

FIGURA 2.10. Carpeta d'Odoo a Windows



Dins la **carpeta server** destaquem:

- L'arxiu «odoo.log», que conté el log del servidor. Podrem trobar-hi tota la informació dels esdeveniments que hi ocorren. És molt important saber consultar aquest arxiu en cas d'errors.

- L'arxiu «odoo.conf», que conté la configuració del servidor. No hem de tocar res, ja que la instal·lació ho ha fet tot, però ens podrà servir per a futures instal·lacions en altres sistemes. Dins aquest arxiu pot consultar-se (figura 2.11):
  - `addons_path`: ruta a la carpeta on es troben els mòduls d'Odoo (i on es podrien afegir-ne mòduls de tercers).
  - `admin_passwd`: contrasenya necessària per a crear noves bases de dades.
  - `db_` variables que defineixen les dades de la base de dades (*user*, *password*, port...).

FIGURA 2.11. Contingut de l'arxiu odoo.conf

```
[options]
addons_path = C:\Program Files\Odoo 13.0\server\odoo\addons
admin_passwd = admin
bin_path = C:\Program Files\Odoo 13.0\thirdparty
csv_internal_sep = ,
data_dir = C:\Users\windows\AppData\Local\OpenERP S.A\Odoo
db_host = localhost
db_maxconn = 64
db_name = False
db_password = 10c_4dm1n
db_port = 5432
db_sslmode = prefer
db_template = template0
db_user = ioc_admin
```

2.2.2 Instal·lació d'Odoo per a un servidor Linux

Als "Annexos" del web trobareu el punt "Recursos de programari" que inclou les versions dels programes als quals fem referència en aquests materials.

La instal·lació d'Odoo *All-in-one* per a Windows és molt eficaç per a tenir ràpidament instal·lada l'aplicació, i així poder aprendre el seu funcionament. Si es desitja una versió de producció, però, haurà de crear-se un servidor dedicat, intentant maximitzar la seva eficiència.

En aquest cas, farem servir una màquina Ubuntu Server 18.04, i instal·larem tots els components necessaris per a treballar amb Odoo.

Aclariments sobre la instal·lació

Es portarà a terme la instal·lació de l'**última versió** al moment de desenvolupament d'aquest manual (v13), encara que és equivalent per a qualsevol altra versió moderna d'Odoo (a partir d'Odoo 10).

Ubuntu Server és un SO lliure que pot trobar-se de manera gratuïta al seu web oficial: bit.ly/2TgQXgg. Tot i que no és objecte d'aquests materials descriure la instal·lació del sistema operatiu; per fer-ho, pot fer-se servir aquest tutorial: bit.ly/34kSsAl.

Per portar a terme la instal·lació, seguirem la **documentació oficial d'Odoo**; concretament, el capítol "Installing Odoo / Source install / Linux". A la versió més actual a la redacció d'aquesta documentació l'enllaç és el següent: bit.ly/2TfGem9.

Preparació del sistema

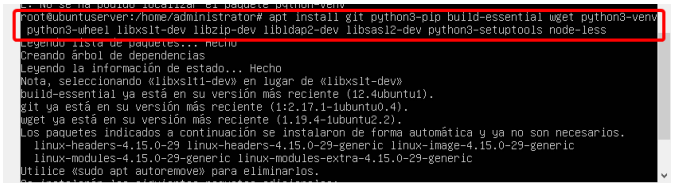
Primer de tot, haurem d'actualitzar la màquina i instal·lar diversos components que necessitarem (figura 2.12). Els més importants són:

- **git**. És un *software* de control de versions àmpliament utilitzat, que ens permetrà descarregar l'última versió d'Odoo des del seu repositori oficial.
- **python-pip**. És un mòdul python que ens permet instal·lar altres mòduls. Donem per fet que Python 3 s'ha instal·lat correctament amb el sistema operatiu.
- **Python-venv**. Permet crear entorns virtuals, amb els que podríem instal·lar, per exemple, diferents versions d'Odoo, amb diferents versions dels mòduls de Python.

Instal·lem, per tant, els components:

```
sudo apt update && sudo apt upgrade
sudo apt install git python3-pip python3-venv
```

Figura 2.12. Instal·lació dels components



També **crearem un usuari**, que serà el propietari dels arxius d'Odoo. Es dirà odoo13. La seva carpeta personal serà /opt/odoo13, que és on descarregarem tots els arxius.

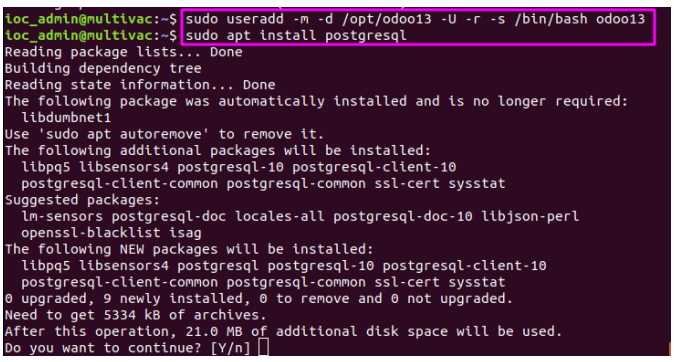
```
sudo useradd -m -d /opt/odoo13 -U -r -s /bin/bash odoo13
```

PostgreSQL

El SGBD que fa servir Odoo és PostgreSQL. És OpenSource, i la seva instal·lació és molt senzilla (figura 2.13).

```
sudo apt install postgresql postgresql-client
```

Figura 2.13. Instal·lació de PostgreSQL



Durant la instal·lació, es crea un usuari anomenat «postgres», amb el que es controla la base de dades. Fent-lo servir, es crea un superusuari intern de

postgreSQL, amb el nom «odoo13» i el *password* «odoo13». Serà l'usuari que farà servir odoo per a la creació y gestió de totes les seves bases de dades.

```
sudo su – postgres –c "createuser –s –P odoo13"
```

paràmetres "s" i "P"

A la sentència de creació de l'usuari "odoo13", destaquem el paràmetre "s", que fa que l'usuari sigui superuser, i el paràmetre "P", que fa que demani el seu *password* per *prompt*.

Més endavant, al punt "Configuració de PostgreSQL per escoltar connexions entrants" d'aquest mateix contingut, s'estudiarà la configuració a postgresql, per tal de permetre connexió a la base de dades des d'una màquina externa (modificació dels arxius postgresql.conf i pg_hba.conf).

Dependències

A continuació instal·lem els **paquets necessaris** per a poder treballar amb Odoo, marcats a la documentació oficial.

```
sudo apt install python3–dev libxml2–dev libxslt1–dev libldap2–dev libsasl2–dev
      \
    libtiff5–dev libjpeg8–dev libopenjp2–7–dev zlib1g–dev libfreetype6–dev \
    liblcms2–dev libwebp–dev libharfbuzz–dev libfribidi–dev libxcb1–dev libpq–
        dev
```

Descàrrega del codi font

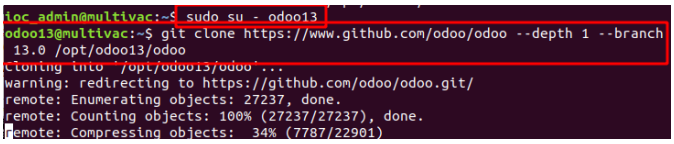
Canviant d'usuari a odoo13, es fa servir 'git' per a obtenir la versió d'Odoo desitjada (figura 2.14).

Paràmetre "–branch 13.0"

A la documentació oficial d'Odoo s'omet el paràmetre `-branch 13.0`, perquè se suposa que es pretén instal·lar l'última versió. És molt recomanable, però, fer-ho, per tal de controlar exactament quina versió s'està descarregant.

```
sudo su - odoo13
git clone https://www.github.com/odoo/odoo ––depth 1 ––branch 13.0 /opt/odoo13/
    odoo
```

FIGURA 2.14. Descàrrega del codi font



Instal·lació dels mòduls Python requerits

En aquest punt s'introduirà el concepte d'**entorn virtual** en Python. Els entorns virtuals permeten crear una instància de python virtual, on poden instal·lar-se mòduls per a un determinat projecte, de manera que no afectarà la resta de l'equip. També permet fer aquest entorn "portable", i poder tenir moure els requisits del programa amb el mateix programa, si es canvia la màquina on està instal·lat.

La sentència de creació de l'entorn virtual és molt senzilla. Recordeu que s'està fent servir l'usuari odoo13:

```
cd /opt/odoo13
python3 –m venv odoo–venv
```

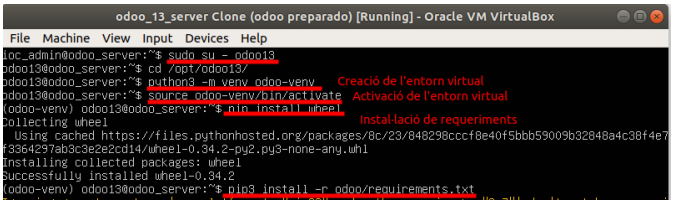
Per activar l'entorn virtual, i d'aquesta manera poder començar a instal·lar-hi tots els mòduls Python imprescindibles per a córrer Odoo, la comanda serà:

```
source venv/bin/activate
```

Al codi font d'Odoo hi ha un arxiu anomenat **requirements.txt**. Conté tots els mòduls Python que necessita Odoo per executar-se. Amb l'entorn virtual activat, fem servir python-pip per a instal·lar tots els mòduls amb una sola comanda. A més a més, instal·larem el mòdul *wheel* (figura 2.15).

```
pip3 install wheel
pip3 install -r odoo/requirements.txt
```

FIGURA 2.15. Instal·lació de dependències



Entorns virtuals de Python

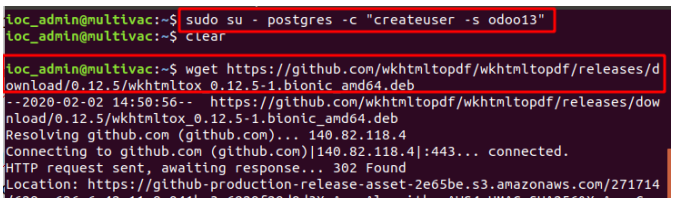
Per a més informació sobre els entorns virtuals de Python pot consultar-se bit.ly/3jkHor5.

wkhtmltopdf

Odoo genera els seus informes en html mitjançant Qweb, i fa servir l'eina wkhtmltopdf per a **convertir-la en PDF**, i així poder-la imprimir de manera senzilla. Des de la documentació oficial d'Odoo es recomana baixar la versió 0.12.5. Baixarem wkhtmltopdf fent servir `wget`, i després la instal·larem mitjançant `apt` (figura 2.16).

```
wget https://github.com/wkhtmltopdf/wkhtmltopdf/releases/download/0.12.5/
    wkhtmltox_0.12.5-1.bionic_amd64.deb
sudo apt install ./wkhtmltox_0.12.5-1.bionic_amd64.deb
```

FIGURA 2.16. Instal·lació de wkhtmltopdf



Creació de l'arxiu de configuració

Odoo fa servir un arxiu anomenat `odoo.conf` per a contenir la seva configuració. Es procedirà, doncs, a la seva creació. També podria incloure's la configuració com paràmetres quan es crida l'arxiu `odoo-bin` per a iniciar el servei, però amb l'arxiu `odoo.conf` assegurem la possibilitat de consultar i canviar la configuració de manera molt senzilla.

```
mkdir /opt/odoo13/odoo-custom-addons
sudo nano /etc/odoo13.conf
```

Introduirem la següent **informació**:

- `admin_password`. Serà el *password* que permetrà crear noves bases de dades (empreses a Odoo), o esborrar-les.
- Informació sobre el host, port, usuari i contrasenya per a la base de dades.
- Ruta a les carpetes de mòduls. Inclourem la carpeta amb els mòduls oficials i la que acabem de crear, per als mòduls de tercers.

A l'hora de desenvolupar **nous mòduls** o incorporar-ne mòduls de tercers, necessitarem un directori on desar-los. Com a mesura de seguretat, no farem servir el mateix on es troben tots els mòduls oficials, crearem una carpeta nova.

```
mkdir /opt/odoo13/odoo_custom_addons
```

Odoo proporcionar un exemple d'arxiu de configuració a la carpeta `debian`. Copiarem, per tant, aquest arxiu, i el ficarem a la carpeta arrel.

```
odoo$ cp debian/odoo.conf /etc/odoo13.conf
```

Ubicació de l'arxiu de configuració

En aquest cas s'ha decidit ficar l'arxiu de configuració a la carpeta `/etc`, per conviure amb scripts d'altres serveis, però no és obligatori, pot triar-se qualsevol altra carpeta, sempre que l'usuari que inicia Odoo disposi de permisos de lectura.

Ara només faltarà **obrir i editar l'arxiu**, de manera que quedi configurada la carpeta dels mòduls propis:

```
[options]
; This is the password that allows database operations:
admin_passwd = my_admin_passwd
db_host = False
db_port = False
db_user = odoo13
db_password = False
addons_path = /opt/odoo13/odoo/addons,/opt/odoo13/odoo_custom_addons
```

Aclariment sobre les variables

A la variable `addons_path` ficarem la ruta de la carpeta amb els mòduls oficials, i la carpeta amb els mòduls propis.

Les variables `db_host` i `db_port` amb el valor False indiquen que han de prendre els valors per defecte (localhost pel host i 5432 pel port). És recomanable, però, canviar el valor "False" pels valors corresponents per evitar errors.

Primera posada en marxa

Amb tots els passos anteriors el servidor ja està preparat per posar en marxa Odoo. Amb l'entorn virtual activat es cridarà a l'arxiu `odoo-bin`, ficant com a paràmetre la ruta de l'arxiu `odoo.conf`.

```
./odoo_bin –c odoo.conf
```

A continuació el servidor començarà a mostrar missatges, un dels quals dirà que Odoo està disponible al host 'localhost', i fent servir el port 8069.

Creació del servei

Abans d'executar-se Odoo, només falta crear un servei, que farà que el servidor s'iniciï amb el sistema operatiu com a servei, de manera que no és necessari la intervenció de ningú.

Generem un arxiu amb el servei, i fem que s'executi en iniciar el sistema.

```
sudo nano /etc/systemd/system/odoo13.service
```

El seu **contingut** serà, entre d'altres, el nom del servei, la ruta per iniciar el servidor i l'usuari i grup amb els que s'ha d'iniciar el servidor.

```
[Unit]
Description=Odoo13
Requires=postgresql.service
After=network.target postgresql.service

[Service]
Type=simple
SyslogIdentifier=odoo13
PermissionsStartOnly=true
User=odoo13
Group=odoo13
ExecStart=/opt/odoo13/odoo-venv/bin/python3 /opt/odoo13/odoo/odoo-bin -c /etc/
    odoo13.conf
StandardOutput=journal+console

[Install]
WantedBy=multi-user.target
```

Només queda provar que tot funciona, i activar el servei.

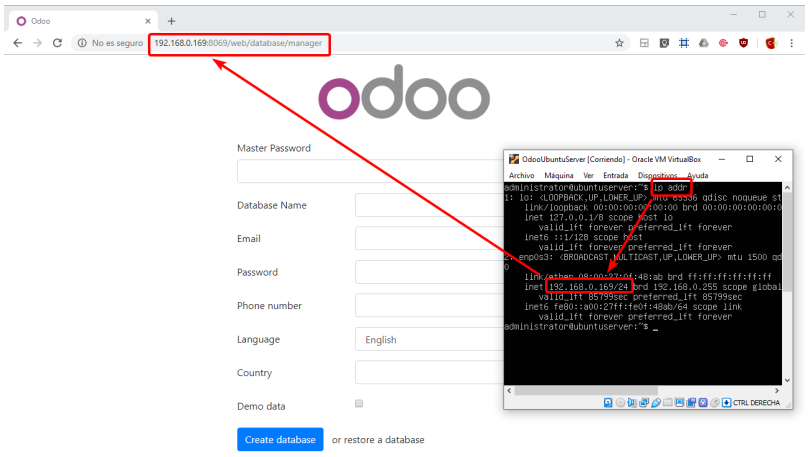
```
sudo systemctl daemon-reload
```

A partir d'aquest moment ja pot provar-se a **reiniciar la màquina**, i accedir a Odoo des de qualsevol altra màquina mitjançant un navegador, introduint l'adreça `<ip_server>:8069`.

Prova de funcionament

Com a últim pas de la instal·lació d'Odoo a Linux, només queda comprovar que tot el procés ha estat correcte. Revisem l'**adreça IP** de la màquina amb «ip addr», i la introduïm al navegador d'una altra màquina. Com no hem configurat el *firewall* del servidor Ubuntu, no hauria d'haver-hi cap problema per accedir-hi (figura 2.17).

FIGURA 2.17. Comprovació de la instal·lació



En el següent vídeo pot veure's el procés d'instal·lació complet, pas a pas:

https://player.vimeo.com/video/469120814

2.2.3 Instal·lació d'Odoo sobre un contenidor Docker

Els contenidors Docker ofereixen lleugeresa i portabilitat.

Segons la seva web oficial «la idea darrere Docker és crear contenidors lleugers i portables per a les aplicacions *software* que puguin executar-se en qualsevol màquina amb Docker instal·lat, independentment del sistema operatiu que la màquina tingui per sota, facilitant així també els desplegaments».

Per a accedir com a usuaris normals a una aplicació, aquesta aplicació *software* necessita estar executant-se en una màquina, en un ordinador. Però a més, depenent del tipus d'aplicació, aquest ordinador també necessita tenir instal·lades una sèrie de **dependències** perquè l'aplicació s'executi correctament: certa versió de Java instal·lada, un servidor d'aplicacions (per exemple tomcat, que és el programari que realment estarà executant la meva aplicació i fent que pugui interactuar amb ella).

Docker permet ficar en un contenidor ("una caixa", una entitat autocontinguda, tancada), totes aquelles coses que l'aplicació necessita per ser executada (java, Maven, tomcat ...) i la mateixa aplicació. Així l'usuari es pot portar aquest contenidor a qualsevol màquina que tingui instal·lat Docker i executar l'aplicació sense haver de fer res més, ni preocupar-se de quines versions de programari té instal·lades a aquesta màquina.

S'executarà l'aplicació de programari del contenidor de Docker, i dins d'ell estaran totes les llibreries i dependències que necessita aquesta aplicació per a funcionar.

Avui dia Docker és utilitzat pels desenvolupadors de manera molt habitual, i és molt important conèixer-lo.

En aquest material, instal·larem Docker en una màquina Ubuntu Linux 18.04, i sobre Docker farem córrer un contenidor per a postgreSQL i un altre per a Odoo.

Primer pas: Instal·lació de Docker

Per a fer la instal·lació de Docker, es tindrà com a referència el tutorial de la pàgina web de Docker.

Podeu trobar l'enllaç al tutorial d'instal·lació a la secció "Annexos" del web del mòdul.

El primer que farem serà **afegir el repositori i instal·lar Docker**. Actualitzarem la màquina i instal·larem les dependències, afegim la clau del repositori i afegim el repositori. Després s'ha de tornar a actualitzar els repositoris i procedir a la instal·lació.

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install \
    apt-transport-https \
    ca-certificates \
    curl \
    gnupg-agent \
    software-properties-common

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

sudo add-apt-repository \
   "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
   $(lsb_release -cs) \

sudo apt-get update

sudo apt-get install docker-ce docker-ce-cli containerd.io
```

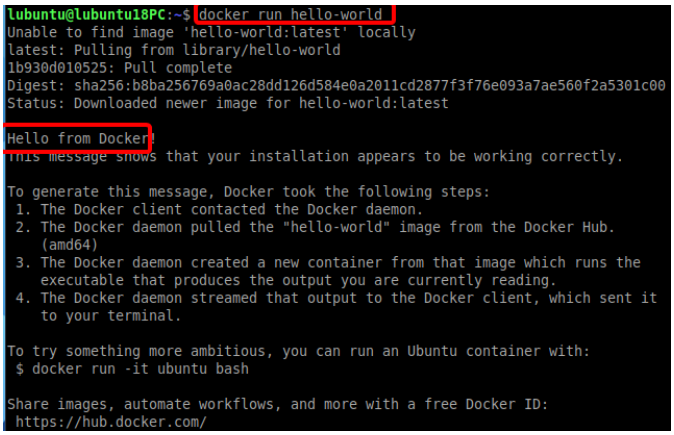
A l'hora d'executar Docker, tenim dues opcions: bé executar docker com a superusuari (sudo), o crear un grup amb el nom «docker», i afegir el nostre usuari.

```
sudo groupadd docker
sudo usermod -aG docker #USER
```

Després, portarem a terme la **verificació de la instal·lació i instal·lació de Compose**. Com a primera aplicació, instal·lem el contenidor «hello-world». És un contenidor estàndard que serveix per a comprovar que la instal·lació de Docker ha estat correcta (figura 2.18).

```
docker run hello-world
```

FIGURA 2.18. Funcionament del contenidor “Hello World”



Instal·larem també «Docker Compose». Ens permetrà crear guions en format YAML per a executar aplicacions multicontenidor. Avui dia és molt habitual l'ús de Docker Compose conjuntament amb Docker.

Trobareu la guia d'instal·lació de Docker Compose, a la secció “Annexos”.

En primer lloc descarreguem i instal·lem l'executable. A continuació se l'hi donaran permisos d'execució.

Format YAML

El format YAML és un llenguatge de serialització de dades molt habitual i cada dia més utilitzat. A més a més de Docker Composer, altres sistemes com ara Ansible o Netplan també ho fan servir. Pots trobar més informació als annexos.

```
1 sudo curl –L "https://github.com/docker/compose/releases/download/1.24.1/docker
      –compose–$(uname –s)–$(uname –m)" –o /usr/local/bin/docker–compose
2 sudo chmod +x /usr/local/bin/docker–compose
```

Per últim, es pot comprovar que la instal·lació ha estat un èxit demanant la versió de Composer.

```
1 docker–compose -version
```

Segon pas: Instal·lació d'Odoo sobre Docker

Podeu trobar la guia d'instal·lació a la secció “Annexos”.

Crearem un guió en format YAML. **Aixecarem un contenidor per al servidor postgreSQL, i un altre per Odoo**. A més a més, farem el contenidor d'Odoo depenent del de postgreSQL.

El nom per defecte de l'arxiu és `docker-compose.yml`.

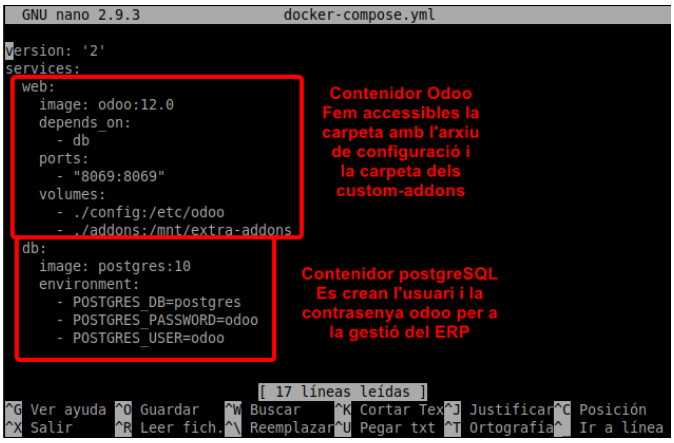
```
1 mkdir odoo13
2 cd odoo13/
3 nano docker–compose.yml
```

El contingut de l'arxiu `docker-compose.yml` (figura 2.19) per a l'última versió d'Odoo, en el moment de realització del present manual, és:

```
1 version: '2'
2 services:
3   web:
4     image: odoo:13.0
5     depends_on:
6       – db
7     ports:
8       – "8069:8069"
9     volumes:
```

```
10      - ./config:/etc/odoo
11      - ./addons:/mnt/extra-addons
12  db:
13    image: postgres:10
14    environment:
15      - POSTGRES_DB=postgres
16      - POSTGRES_PASSWORD=odoo
17      - POSTGRES_USER=odoo
```

FIGURA 2.19. Arxiu docker-composer.yml

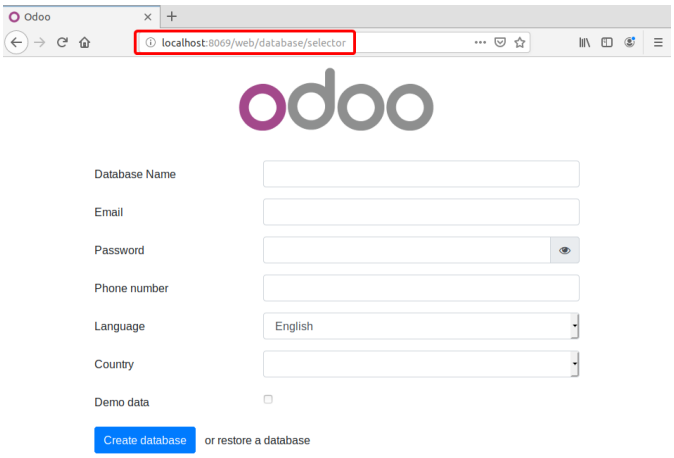


Es generen **dos contenidors**:

- Contenidor amb una instància de postgresql 10:
  - Genera una base de dades amb el nom postgres.
  - Genera un usuari amb el nom odoo i *password* Odoo.
- Contenidor amb una instància d'Odoo:
  - Es connecta al contenidor postgresql creat en el pas anterior.
  - Fa una redirecció de ports, de manera que el port 8069 del host es redirigeix al port 8069 del contenidor. Així podrà connectar-se a Odoo des de `localhost:8069`.
  - Fa una redirecció de carpetes, de manera que les carpetes del host `./config` i `./addons` donant accés a les carpetes `/etc/odoo` i `/mnt/extra-addons` del contenidor respectivament.

... i ja podem obrir el navegador i accedir a Odoo a l'adreça `localhost:8069` (figura 2.20).

FIGURA 2.20. Odoo sobre Docker en funcionament



Altres comandes amb Docker

- `docker ps`: mostra tots els contenidors que estan executant-se en un determinat moment. Si afegim -a, ens mostra tots els contenidors, tant els que s'estan executant com els que estan aturats (figura 2.21).
- `docker run`. Crea un nou contenidor, descarregant-lo d'internet si és necessari.
- `docker start`, *stop*, *pause*, *unpause*. Inicia, atura, pausa o reprèn un contenidor.
- `docker exec`. Executa una comanda en un contenidor que ja existeix.
- `docker rm`. Esborra un contenidor.

FIGURA 2.21. Llistat de contenidors

```
lubuntu@lubuntu18PC:~/odoo12$ docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS                      PORTS                              NAMES
81059a3c4c2c        odoo:12.0           "/entrypoint.sh odoo"    10 minutes ago      Up 10 minutes               0.0.0.0:8069->8069/tcp, 8071/tcp   odoo12_web_1
e0b5485461ef        postgres:10         "docker-entrypoint.s…"   10 minutes ago      Up 10 minutes               5432/tcp                           odoo12_db_1
6ced8bb83d2a        hello-world         "/hello"                 32 minutes ago      Exited (0) 32 minutes ago                                      bold_yonath
```

A continuació, pot veure's la mateixa instal·lació descrita fins ara en un vídeo pas a pas:

https://player.vimeo.com/video/469120851

2.3 Usuaris al voltant d'un servidor Odoo

Abans de procedir a gestionar empreses, és convenient entendre els tres tipus d'usuaris existents al voltant d'un servidor Odoo:

1. **Usuari del servidor PostgreSQL amb privilegi de creació de bases de dades**: és l'usuari de PostgreSQL que utilitza el servidor Odoo per crear i eliminar empreses (bases de dades) dins el servidor PostgreSQL. Una

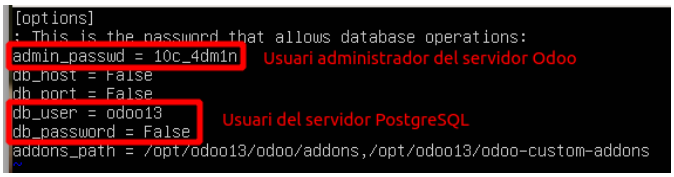
empresa d'Odoo es correspon amb una base de dades dins el servidor PostgreSQL.

- En cas que el servidor PostgreSQL hagi estat instal·lat pel procediment d'instal·lació d'Odoo All-In-One, el procés proposa l'usuari *openpg* (contrasenya *openpgpwd*) com a superusuari del servidor PostgreSQL, però es pot decidir qualsevol parella (usuari-contrasenya). Durant el tutorial d'instal·lació s'ha proposat ioc_admin/10c_4dm1n
- En cas que el servidor PostgresSQL hagi estat instal·lat de manera independent al procediment d'instal·lació d'Odoo, disposarà d'un superusuari i hauríem de poder utilitzar aquest (sempre que no s'anomeni *postgres*, ja que per qüestions de seguretat Odoo no permet aquest nom d'usuari) o qualsevol altre amb autorització per crear bases de dades per tal que l'utilitzi l'Odoo per crear i eliminar empreses (bases de dades) dins el servidor PostgreSQL. En un servidor Linux es pot crear aquest usuari amb la sentència `sudo su - postgres -c ‘‘createuser -s -P odoo13’’`.
- L'usuari del servidor PostgreSQL amb autorització per crear bases de dades i la seva contrasenya han de residir, respectivament, en les entrades `db_user` i `db_password` del fitxer `odoo.conf`. Si en comproveu el contingut, hi veureu l'usuari i la contrasenya que havíem indicat en el procés d'instal·lació.

2. **Usuari administrador del servidor Odoo:** és un usuari únic del servidor Odoo, que és el que pot crear i eliminar les empreses. No té nom i únicament té contrasenya. Aquesta contrasenya es configura a l'arxiu `odoo.conf`.

3. **Usuaris de cada empresa creada en el servidor Odoo:** el procés de creació d'una empresa ve donat amb un usuari administrador, que es genera mitjançant el formulari inicial de creació de l'empresa. El seu *username* serà correu electrònic del responsable. L'usuari *admin* de cada empresa té tots els privilegis i, una vegada connectat a l'empresa, pot crear usuaris, grups de privilegis sobre els objectes d'Odoo (tercers, productes, comandes, albarans, factures...) i assignar usuaris als diversos grups de privilegis. En cas de perdre la contrasenya de l'usuari *admin*, un usuari administrador del servidor PostgreSQL pot recuperar-la tot consultant-la directament la base de dades, a través de les eines d'accés que el servidor PostgreSQL facilita (figura 2.22).

Figura 2.22. Usuaris del servidor PostgreSQL i administrador del servidor Odoo



El procés de creació d'una empresa permet carregar unes dades de demostració i, en tal situació, també hi incorpora un usuari de nom *demo* i contrasenya *demo*. Aquest usuari no existeix en la creació d'una empresa buida.

2.4 Coneixements bàsics del servidor PostgreSQL

Aquest punt descriurà l'eina pgAdminIII instal·lada amb el paquet *All-In-One* per a Windows, encara que és 100% equivalent a la instal·lació de pgAdminIII en una màquina Linux.

PostgreSQL (www.postgresql.org) és un SGBD relacional distribuït sota llicència BSD, desenvolupat per PostgreSQL Global Development Group.

Versions de PostgreSQL

En 12-04-2020 hi ha vàries versions de PostgreSQL desenvolupant-se a la vegada, de manera que poden descarregar-se de la web les 12.2, 11.7, 10.12, 9.6.17, 9.5.21, i 9.4.26. La instal·lació d'Odoo per a Windows incorpora PostgreSQL en la seva versió 9.6, i a Linux hem fet servir la versió 10.12. A la documentació oficial d'Odoo no especifica quina ha de ser la versió del SGBD.

Nosaltres, com a implantadors tècnics d'ERP, hem de ser coneixedors de l'**estructura de la base de dades** per si hem de desenvolupar mòduls que complementin la funcionalitat que facilita l'ERP.

Com que la base de dades es troba implementada en un SGBD concret, ens convé conèixer les eines bàsiques de què disposem per **moure'ns amb facilitat dins de l'SGBD**. La majoria d'SGBD actuals faciliten dos tipus d'eines per accedir a les bases de dades i facilitar la gestió:

- Eines gràfiques i/o consoles textuals: basades en l'arquitectura client-servidor, obliguen a instal·lar l'eina a la màquina des de la qual es vol accedir a l'SGBD, que pot residir en una màquina remota.
- Eines gràfiques web: permeten l'accés des de navegadors i, per tant, eviten el fet d'haver d'instal·lar cap programari client.

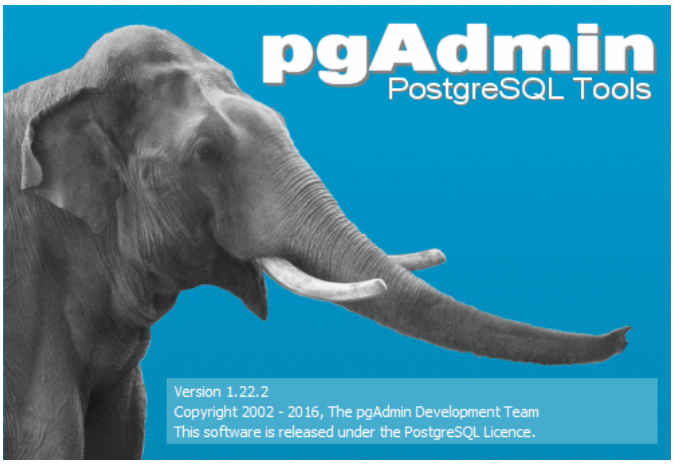
Per **accedir a PostgreSQL** disposem de moltes eines. Entre elles, cal conèixer l'existència de:

- Eina gràfica pgAdminIII, amb arquitectura client-servidor.
- A partir de PostgreSQL 10 pgadminIII per a Windows deixa de ser compatible 100% (encara que es mostra molt fiable per a fer les operacions més senzilles). S'ha creat pgAdminIV, que és una eina amb servidor web propi.
- Consola textual psql, amb arquitectura client-servidor.
- Eina gràfica phpPgAdmin, amb servidor web (necessita PHP).

2.4.1 Eina pgAdmin

L'eina pgAdmin és una de les eines més habituals d'utilitzar **per accedir a un servidor PostgreSQL** (local o remot) i es pot instal·lar a qualsevol màquina. La podem baixar de la pàgina oficial (www.pgadmin.org) i és tan habitual que l'Odoo la incorpora en la instal·lació per a Windows que facilita (figura 2.23).

Figura 2.23. Pantalla inicial de pgAdminIII

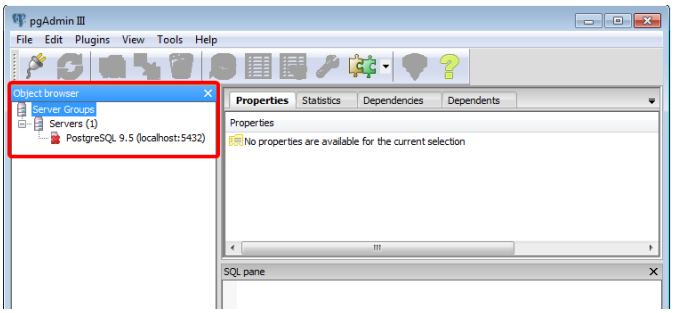


Caldrà **determinar** els servidors, els usuaris, les bases de dades i els seus esquemes.

Servidors

La part esquerra de la pantalla principal de pgAdmin està destinada a incorporar tots els servidors PostgreSQL als quals volem accedir amb aquesta eina, ja siguin a la mateixa màquina o en màquines remotes. Cal comentar que en una mateixa màquina hi poden coexistir **diversos servidors** PostgreSQL, amb la precaució que han d'escoltar per diferents ports (figura 2.24).

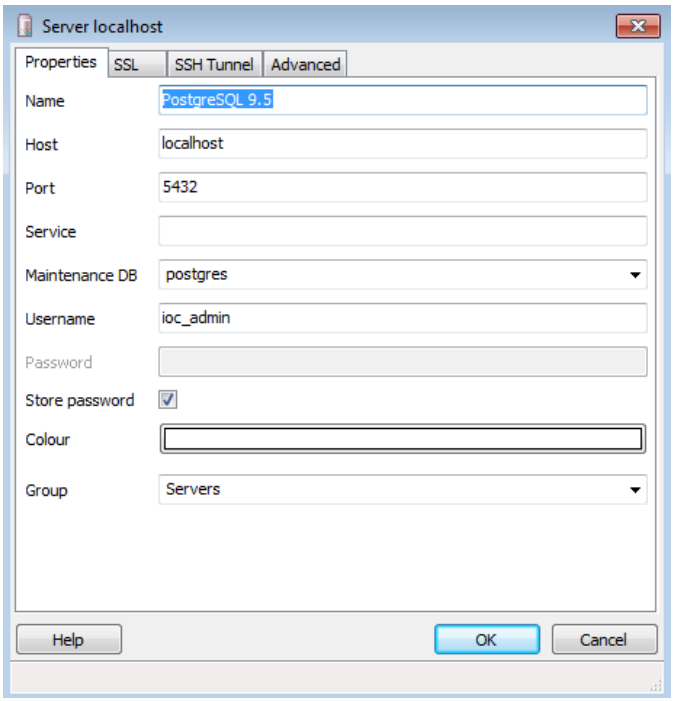
Figura 2.24. Servidors en pgAdminIII



Podem comprovar que l'eina pgAdmin que instal·la Odoo ja té registrat el servidor PostgreSQL instal·lat per Odoo a la màquina i el veiem acompanyat d'una creu vermella que indica que no hi estem connectats. Per establir connexió i poder gestionar el seu contingut cal situar-nos al damunt i prémer amb doble clic del ratolí.

Si ens situem damunt la connexió PostgreSQL for Odoo (localhost:5432) i premem el botó secundari del ratolí, podrem accedir a les propietats (figura 2.25).

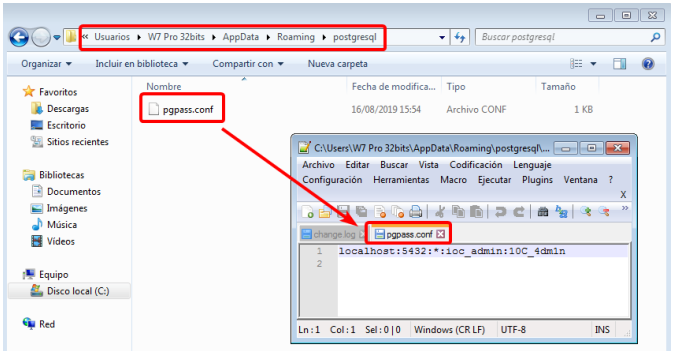
FIGURA 2.25. Propietats de la connexió



Observem que en aquesta pàgina de propietats podrem canviar l'usuari per defecte i també podrem enregistrar la contrasenya, una vegada connectats, per evitar haver d'introduir-la cada vegada que accedim al servidor PostgreSQL.

S'ha d'anar amb molt de compte a l'hora de deixar les **contrasenyes enregistrades**; això només s'hauria de fer en màquines de les quals tenim la seguretat que només hi tindran accés usuaris que, a la vegada, hagin de tenir accés als servidors PostgreSQL enregistrats en pgAdmin. Les contrasenyes enregistrades des de pgAdmin es troben en el fitxer pgpass.conf ubicat dins d'una carpeta anomenada PostgreSQL que es troba dins del perfil de l'usuari que ha creat les connexions i a la màquina client des de la qual s'executa pgAdmin (figura 2.26). Aquesta ubicació depèn de la versió del SO; en el cas de Win7: ...\users\nomUsuari\AppData\Roaming\postgresql

FIGURA 2.26. Arxiu per emmagatzemar els 'passwords'

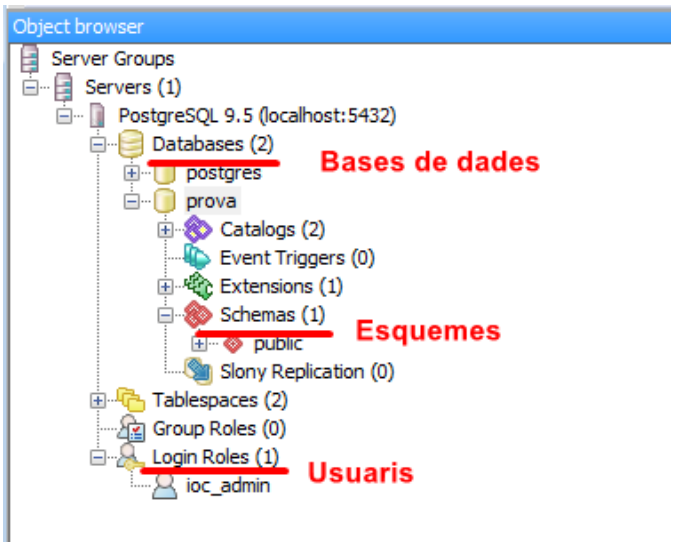


És important tenir-ho present, perquè la resta d'eines de PostgreSQL instal·lades en el sistema utilitzen el mateix arxiu per registrar o comprovar les contrasenyes. En conseqüència, si un usuari ha registrat una connexió des de pgAdmin amb un

usuari i la corresponent contrasenya, la resta d'eines de PostgreSQL no li exigiran introduir la contrasenya per establir connexió amb el servidor i utilitzaran l'usuari pel qual hi ha la contrasenya registrada.

Una vegada establerta la connexió contra un servidor PostgreSQL la part esquerra de la pantalla ens mostra el contingut del servidor PostgreSQL (figura 2.27).

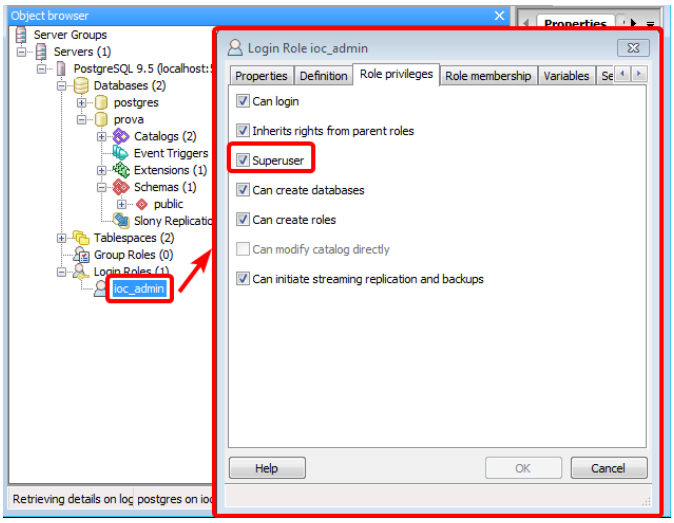
FIGURA 2.27. Bases de dades, esquemes, usuaris



Usuaris

Un servidor PostgreSQL pot tenir molts usuaris. La imatge ens mostra només un usuari, de nom ioc_admin, que és el **superusuari** del servidor PostgreSQL. Si ens hi situem al damunt i premem el botó secundari del ratolí per anar a les seves propietats, observarem que té privilegis totals (figura 2.28).

FIGURA 2.28. Propietats de l'usuario ioc_admin



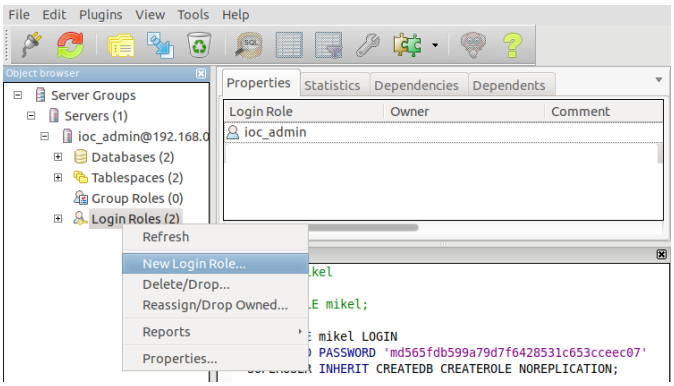
De moment no ens cal crear nous usuaris, però quan l'ERP es posa en execució en una organització, aquesta pot tenir usuaris finals que vulguin poder efectuar consultes contra les bases de dades. En aquest cas els haurem de facilitar un usuari

amb els **privilegis de lectura** adequats, però mai d'escriptura, a les bases de dades i taules o vistes que corresponguin.

No hem de confondre els usuaris de l'SGBD amb els usuaris de l'ERP. Els usuaris de l'ERP estan emmagatzemats en taules pròpies de l'ERP i l'ERP utilitza un usuari de PostgreSQL (ioc_admin en aquest cas) per accedir a la BD de l'empresa.

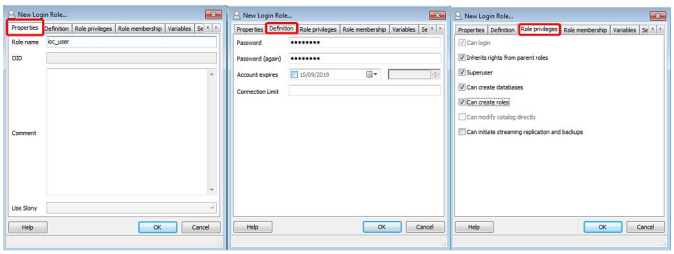
Crearem un nou usuari **superadministrador** per a no interferir amb l'usuari que ha creat Odoo. L'hi direm ioc_user, amb el *password* 10c_us3r. Farem click amb el botó dret i triarem «New login role» (figura 2.29).

FIGURA 2.29. Creació d'un nou usuari



A continuació triarem el nom, el *password*, i l'hi donarem privilegis de superusuari (figura 2.30).

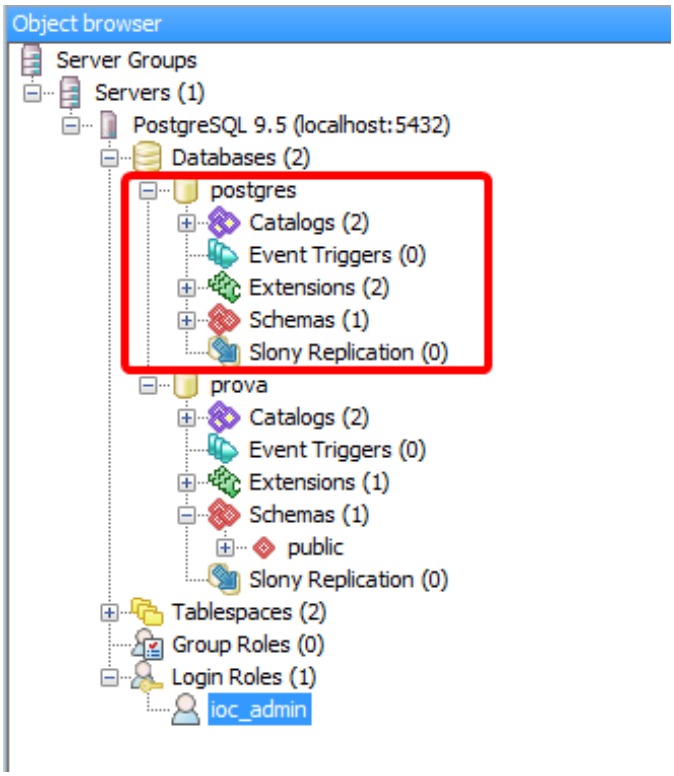
FIGURA 2.30. Creació d'un nou usuari



Bases de dades

Un servidor PostgreSQL pot tenir diverses bases de dades, però com a mínim n'ha de tenir una. En moltes ocasions, la primera base de dades que s'instal·la s'anomena "**postgres**", però podria tenir qualsevol altre nom (figura 2.31).

Figura 2.31. Base de dades postgres



El procés d'instal·lació d'Odoo ha seguit el costum d'anomenar "postgres" la primera base de dades del servidor. Aquesta primera base de dades és especial en el sentit que és la base de dades de manteniment del servidor i mai podrà ser eliminada. De fet, si intentem eliminar-la, ens apareixerà un missatge informant que és la base de dades de manteniment i no pot ser eliminada.

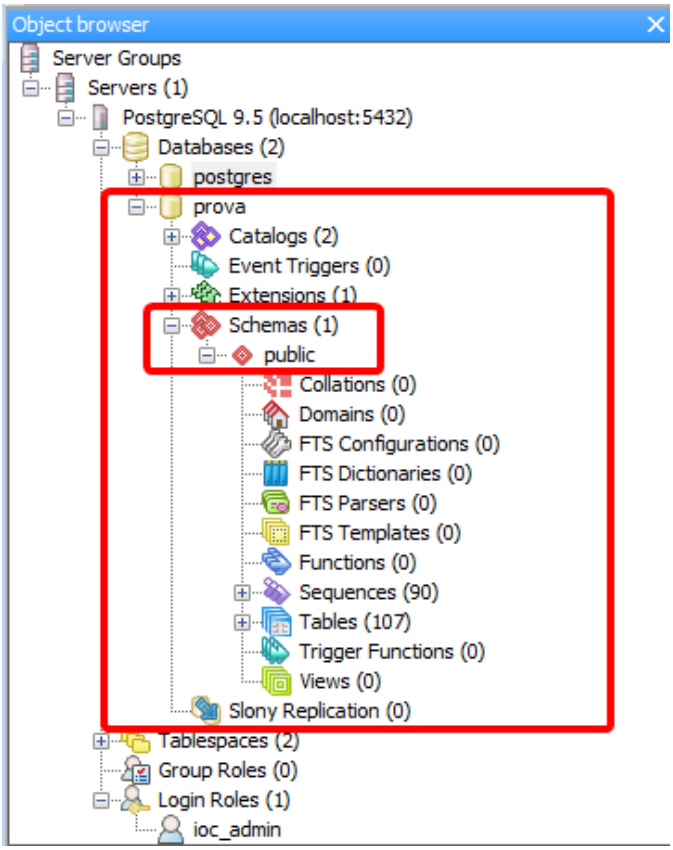
A mesura que anem creant empreses aniran apareixent les corresponents bases de dades en el servidor PostgreSQL. Cada base de dades tindrà els seus usuaris per gestionar l'empresa des d'Odoo.

Esquemes

Una base de dades de PostgreSQL està compartimentada en esquemes, i com a mínim hi ha un esquema de nom públic; els continguts d'aquest esquema són accessibles per a tots els usuaris que tinguin accés a la base de dades. Aquest és el cas de les bases de dades que crea Odoo. Cada empresa es correspon amb una base de dades que té tota la informació dins l'únic esquema de nom públic.

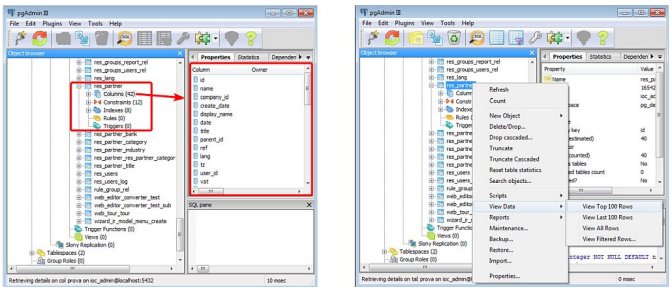
Un esquema conté tot allò que en altres SGBD és el **contingut d'una base de dades**: dominis, taules, vistes, funcions, seqüències i disparadors (figura 2.32).

FIGURA 2.32. Base de dades postgres



Durant la implantació d'un ERP pot ser necessari analitzar la base de dades, i conèixer les taules que la formen. Podem accedir a la llista de taules, i dins cada taula, als camps i la seva descripció (figura 2.33).

FIGURA 2.33. Visualització de les columnes d'una taula i els seus registres.

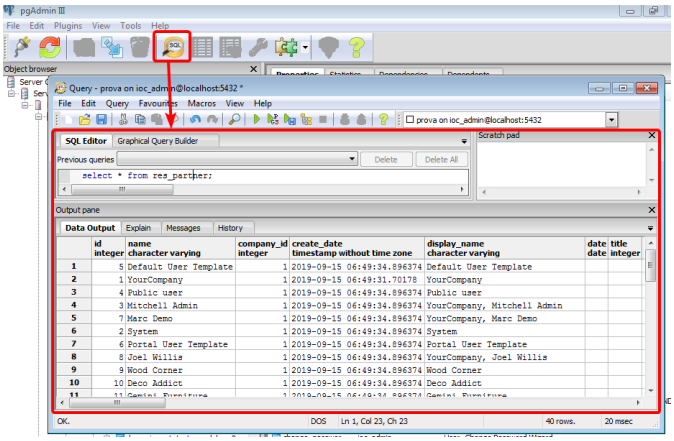


Treballar amb la base de dades

Si pensem utilitzar pgAdmin per executar sentències DML (insert, update, delete), cal saber que està configurada amb el comportament **autocommit on**. Això vol dir que qualsevol operació de modificació de dades sobre la base de dades és automàticament validada sense que l'usuari hagi d'efectuar *commit*, i, per tant, no és possible invocar un *rollback*.

PgAdmin incorpora una consola SQL on podem introduir les consultes i comprovar els resultats (figura 2.34).

FIGURA 2.34. Consola SQL



2.4.2 Configurar PostgreSQL per admetre connexions remotes a un host Windows

El servidor PostgreSQL que instal·la Odoo (i també la majoria d'instal·lacions de servidors PostgreSQL) estan configurades per admetre únicament connexions locals des de la màquina on s'allotja el servidor. En la majoria de les instal·lacions tindrem el servidor d'aplicació i de dades a la mateixa màquina, i per tant no necessitarem fer cap canvi.

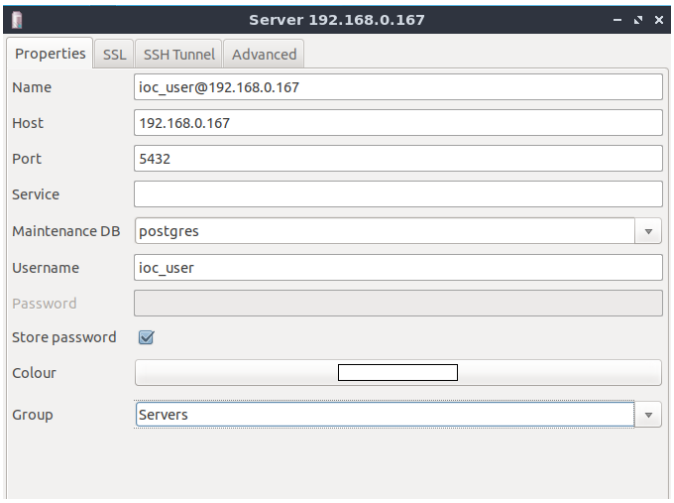
És molt possible, però, que vulguem accedir directament a les bases de dades del servidor PostgreSQL des d'altres eines clients, com per exemple una instal·lació pgAdmin ubicada en una màquina remota o una aplicació que es vol connectar a través d'ODBC.

Per aconseguir-ho ens cal retocar alguns paràmetres de configuració del servidor PostgreSQL i, per entendre la utilitat de cadascun, cal intentar la connexió remota, veure els errors que es produeixen i **anar aplicant les solucions que pertoquin** fins a aconseguir la connectivitat.

Farem servir, per tant, una altra màquina amb pgAdmin instal·lat. Intentem la gestió del servidor PostgreSQL des d'aquest pgAdmin. El primer que hem de fer és **enregistrar un servidor** a través de l'opció File | Add Server. Ens apareix la pantalla de propietats de la connexió (figura 2.35) en la qual ens cal introduir:

Als "Annexos" del web trobareu el punt "Recursos de programari", que inclou les versions dels programes als quals fem referència en aquests materials.

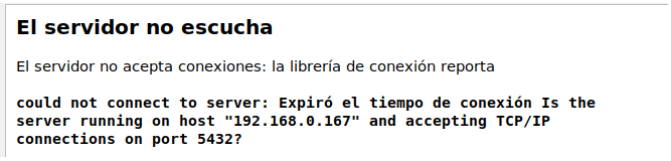
FIGURA 2.35. Nova connexió



1. *Name*: nom informatiu del servidor PostgreSQL amb el qual establim la connexió.
2. *Host*: IP de la màquina que conté el servidor PostgreSQL.
3. *Port*: per defecte 5432, encara que es pot canviar.
4. *MaintenanceDB*: base de dades de manteniment. Per defecte *postgres*.
5. *Username*: usuari de PostgreSQL amb drets de connexió.
6. *Password*: contrasenya de l'usuari indicat a *username*.

Una vegada introduïts aquests valors, premem "OK" per aconseguir la connexió i ens apareix un missatge d'error (figura 2.36). El missatge diu que a l'adreça IP ficada i pel port ficat no es troba cap servidor en execució que accepti connexions TCP/IP.

FIGURA 2.36. Error de connexió

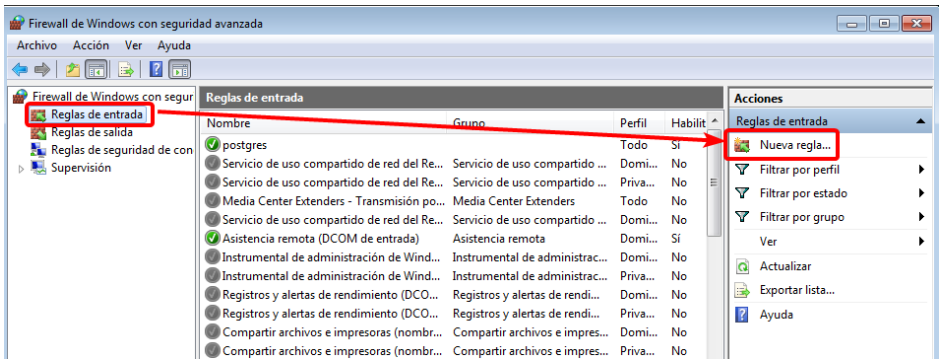


Toca realitzar diverses comprovacions.

'Firewalls'

Hi ha algun *firewall* que no permet la connexió al port 5432? En el cas d'un servidor Windows, haurem d'obrir el *firewall* de Windows amb seguretat avançada i comprovar si està funcionant, i si el port 5432 té alguna restricció. Si necessitem permetre el port 5432, crearem una **regla d'entrada** (figura 2.37).

FIGURA 2.37. Nova regla d'entrada



A continuació **omplirem un assistent** per a obrir el port 5432 de la nostra màquina. No és un procés complicat i es presenta a continuació (figura 2.38 i figura 2.39).

FIGURA 2.38. Nova regla d'entrada

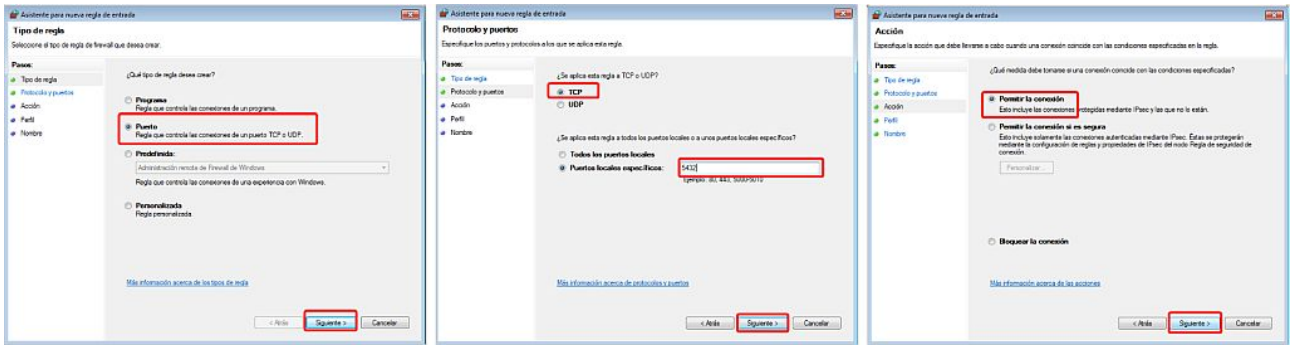
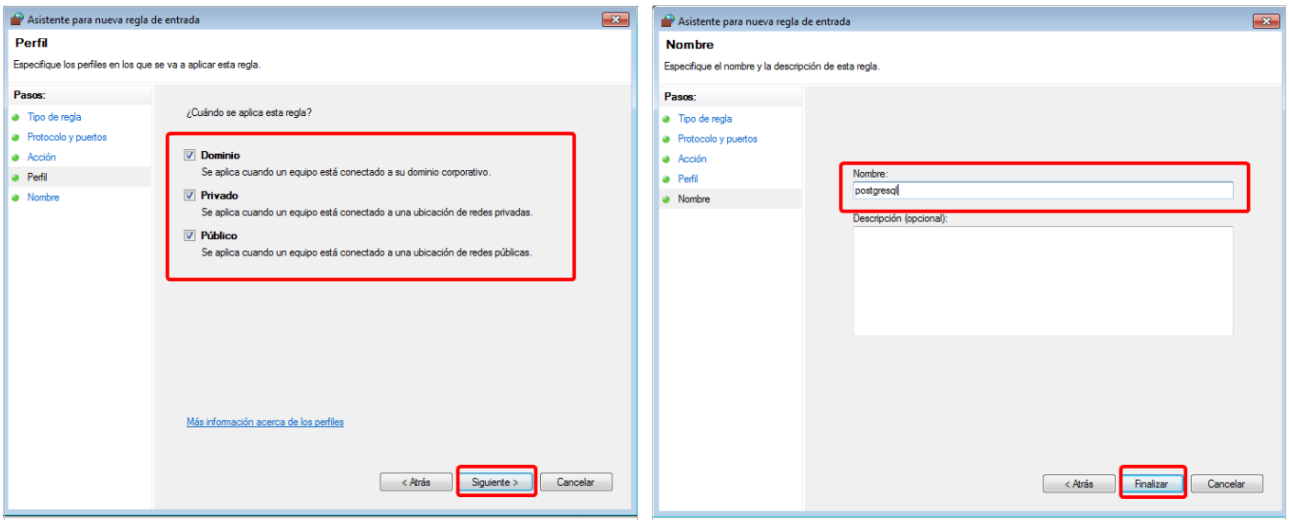


FIGURA 2.39. Nova regla d'entrada



A partir d'ara, podem estar segurs que Windows no plantejarà cap problema a la connexió. El següent pas serà que postgreSQL permeti l'accés.

En el cas de Linux, el servidor Ubuntu Server instal·lat amb anterioritat no disposa de *Firewall* instal·lat per defecte. En cas d'activar-lo s'hauria també d'obrir el port 5432.

Configuració de PostgreSQL per escoltar connexions entrants

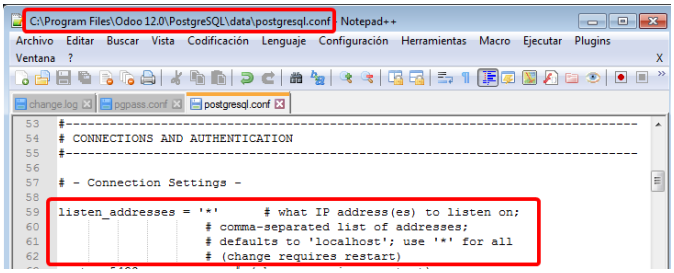
Per defecte, postgreSQL no escolta per totes les **adreces IP** de la màquina que conté el servidor, només ho fa per l'adreça 127.0.0.1. Si volem que escolti per

altres adreces IP pròpies, cal configurar-lo. L'arxiu de configuració que conté aquesta informació és *postgresql.conf*, el paràmetre *listen_addresses*.

```
#listen_addresses = 'localhost'
    # what IP address(es) to listen on;
    # comma-separated list of addresses;
    # (change requires restart)
```

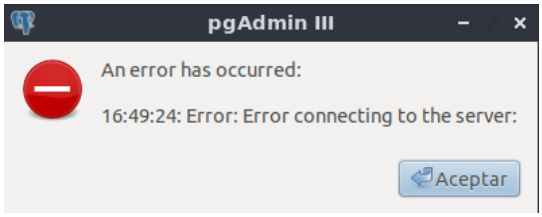
Eliminem el símbol # de l'inici que indica que el paràmetre està comentat i, a la paraula *localhost* hi afegim, separades per comes, les adreces IP de la màquina per les quals volem que el servidor PostgreSQL doni resposta. O, simplement, hi deixem un **asterisc** per indicar que escolti per totes les adreces IP que tingui definides (figura 2.40).

FIGURA 2.40. Paràmetre listen_addresses de l'arxiu postgresql.conf



Una vegada enregistrat el canvi en el fitxer postgresql.conf, ens cal reiniciar el servidor PostgreSQL (anant a services.msc) i tornar a intentar la connexió. Ens apareix un nou error (figura 2.41).

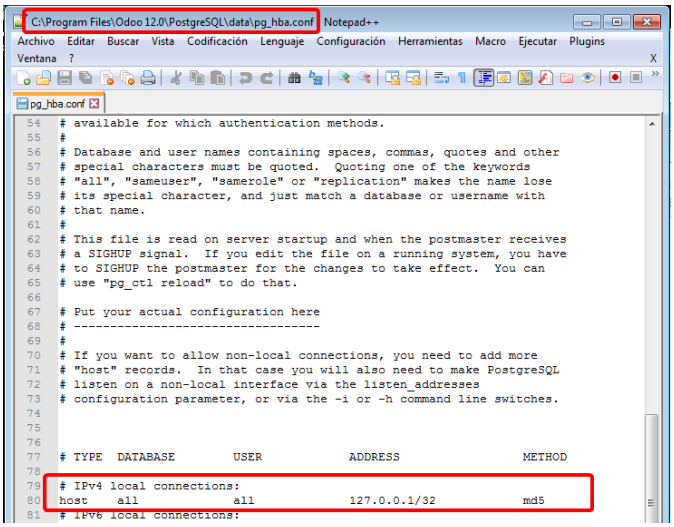
FIGURA 2.41. Error de connexió



L'usuari *ioc_admin* no té autorització per connectar amb la base de dades postgres des de la màquina actual. Canviarem aquesta configuració a l'arxiu **pg_hba.conf**.

L'arxiu mostra totes les màquines que poden connectar-se, a quina base de dades i amb quin usuari. Per defecte només es pot connectar des de 'localhost', a qualsevol base de dades i amb qualsevol usuari (figura 2.42).

FIGURA 2.42. pg_hba.conf



El fitxer pg_hba.conf és un fitxer que controla l'autenticació dels clients que es connecten al servidor PostgreSQL. Per una explicació detallada de totes les possibilitats que facilita aquest fitxer, cerqueu pg_hba.conf dins la documentació de PostgreSQL. De forma molt simplificada, ens cal saber que aquest fitxer conté línies cada una de les quals correspon a un permís d'autenticació. Inicialment ve configurat com:

```
# TYPE  DATABASE    USER    CIDR-ADDRESS          METHOD

# IPv4 local connections:
host    all         all     127.0.0.1/32          md5
# IPv6 local connections:
#host   all         all      ::1/128              md5
```

Fixem-nos amb l'única línia no comentada, que permet l'accés a totes les bases de dades de qualsevol usuari connectat a la mateixa màquina (127.0.0.1/32) i que utilitza una contrasenya encriptada amb el mètode *md5*.

La nomenclatura *CIDR-address* 127.0.0.1/32 es pot substituir per la nomenclatura que indica l'adreça i la màscara en dues columnes:

```
host  all   all    127.0.0.1    255.255.255.255     md5
```

Per permetre l'accés a qualsevol usuari des de la màquina amb IP 10.200.1.207, afegiríem una nova línia (sota l'existent inicialment):

```
host  all   all    10.200.1.207 255.255.255.255     md5
```

o equivalentment:

```
host  all   all    10.200.1.207/32                  md5
```

Per permetre l'accés a qualsevol usuari des de qualsevol màquina 10.200.x.x, afegiríem:

```
host  all   all    10.200.0.0    255.255.0.0        md5
```

o equivalentment:

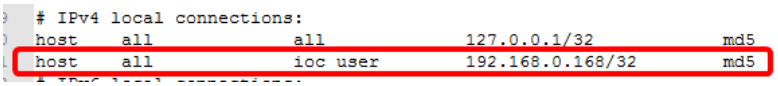
```
host  all   all    10.200.0.0/16                  md5
```

La columna *METHOD* permet múltiples possibilitats. Entre elles, la possibilitat de prohibir la connexió (valor *reject*). Cal **tenir en compte** que:

- El servidor PostrgreSQL carrega el contingut de l'arxiu pg_hba.conf a la memòria quan es posa en marxa i, per tant, caldrà reiniciar-lo davant de qualsevol modificació d'aquest arxiu.
- Quan s'intenta autenticar una connexió, l'avaluació segueix l'ordre de les diverses entrades existents a pg_hba.conf i s'aplica la primera entrada amb la qual s'aconsegueix una coincidència. Com a exemple, si l'entrada 10 restringeix la connexió per una IP XXX concreta, però en una entrada anterior a la 10 es concedeix accés per la IP XXX, la connexió serà autenticada sense cap problema.

Afegirem, per tant, una línia per a permetre que la nova màquina també pugui connectar-se (figura 2.43).

FIGURA 2.43. Permís de connexió a PostgreSQL



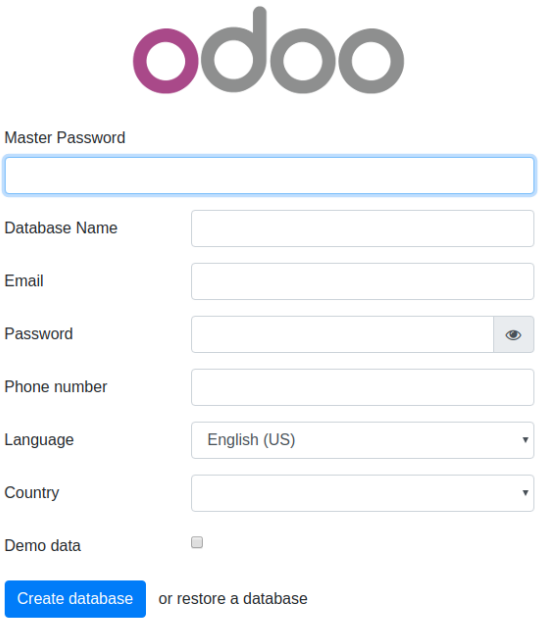
Fins aquí la configuració de PostgreSQL en una màquina Windows. Per a un host linux el procés és molt similar. Pot veure's el procés en aquest vídeo:

https://player.vimeo.com/video/469120862

2.5 Configuració inicial d'Odoo

El primer pas, una vegada s'ha implantat una instància d'Odoo, és la **creació de la base de dades de l'empresa**. Quan Odoo no troba cap base de dades al SGBD PostgreSQL mostra una pantalla per a tal efecte (figura 2.44).

Figura 2.44. Pantalla inicial de creació de bases de dades

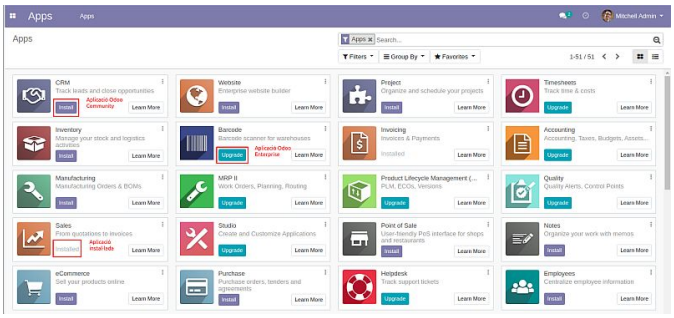


La informació que s'hi haurà d'introduir serà:

- ***Master Password***. Contrasenya mestra de la qual ja s'ha parlat anteriorment en aquest material. Està configurat a l'arxiu odoo.conf i és necessària tant per a crear com per a esborrar bases de dades.
- ***Database Name***. Nom de la base de dades. Es recomana fer servir el nom de l'empresa per a poder reconèixer-la fàcilment.
- **Dades de l'usuari administrador**. En aquest procés es crearan les credencials per a l'usuari gestor de l'ERP.
  - Email. Es farà servir també de nom d'usuari.
  - *Password*.
  - *Phone number*.
- ***Language***. Odoo és multilingüe, en aquesta opció es tria l'idioma per defecte de l'aplicació. Poden canviar-se o afegir-se més idiomes, perquè cada usuari pugui escollir.
- ***Country***. Aquesta informació és molt important per a temes fiscals, ja que Odoo aporta mòduls amb la fiscalitat de cada país.
- **Dades d'exemple**. Quan s'instal·la Odoo poden crear-se dades d'exemple per a cada mòdul, que ajuden a entendre millor el funcionament dels diferents mòduls. Òbviament aquesta funcionalitat només es necessitarà en una primera fase d'aprenentatge de l'aplicació.

Una vegada acceptat es procedeix a la instal·lació del nucli d'Odoo i les dades d'exemple (si escau). Quan tot el procés ha finalitzat, hi sortirà la **pantalla d'aplicacions d'Odoo**, on poden veure's tots els mòduls disponibles (figura 2.45).

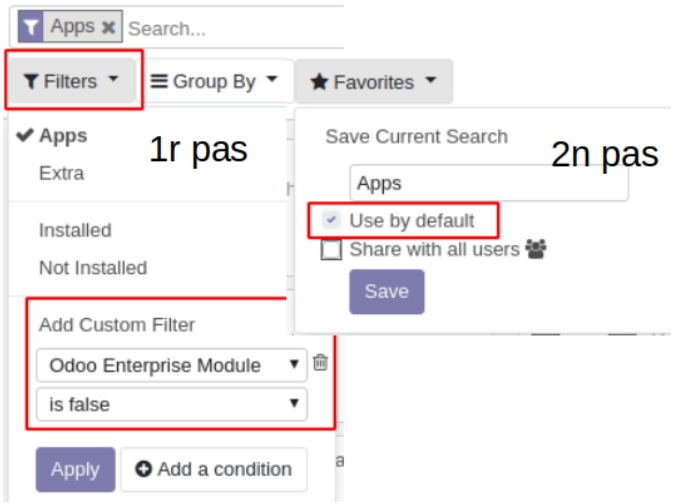
FIGURA 2.45. Pantalla d'aplicacions d'Odoo



En aquesta pantalla poden trobar-se aplicacions amb la llegenda "Install", i d'altres amb la llegenda "Upgrade". Les primeres pertanyen a Odoo Community, i poden instal·lar-se de manera gratuïta, però les segones són part d'Odoo Community, i en fer clic al botó ens portarà a una pantalla amb informació per a adquirir la versió de pagament.

Si tenir aquestes **aplicacions de pagament** a la vista resulta molest per l'administrador d'Odoo, pot configurar-se un filtre que les amaga, i fins i tot fer que sempre s'apliqui aquest filtre per defecte en accedir al mòdul d'aplicacions (figura 2.46).

FIGURA 2.46. Creació d'un filtre d'inici

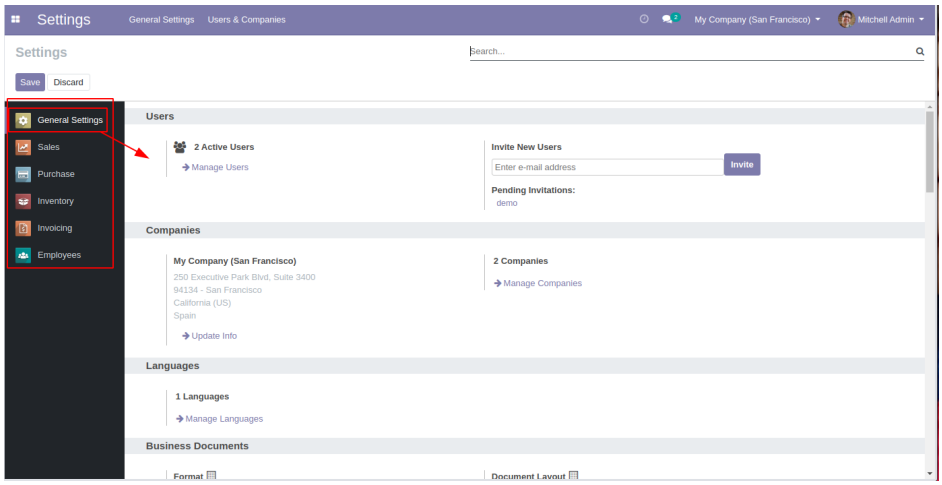


A continuació poden anar instal·lant-se els diferents mòduls que l'empresa necessita. És recomanable fer un bon estudi previ de les necessitats de l'empresa per tal de no instal·lar més mòduls dels necessaris.

2.5.1 Configuració general d'Odoo

Al menú principal pot accedir-se a l'apartat "settings", on es troben tant la configuració general com la de cadascun dels mòduls que s'han anat instal·lant (figura 2.47).

FIGURA 2.47. Menú de configuració d'Odoo



Dins les **opcions de configuració** generals d'Odoo poden destacar-se les següents opcions:

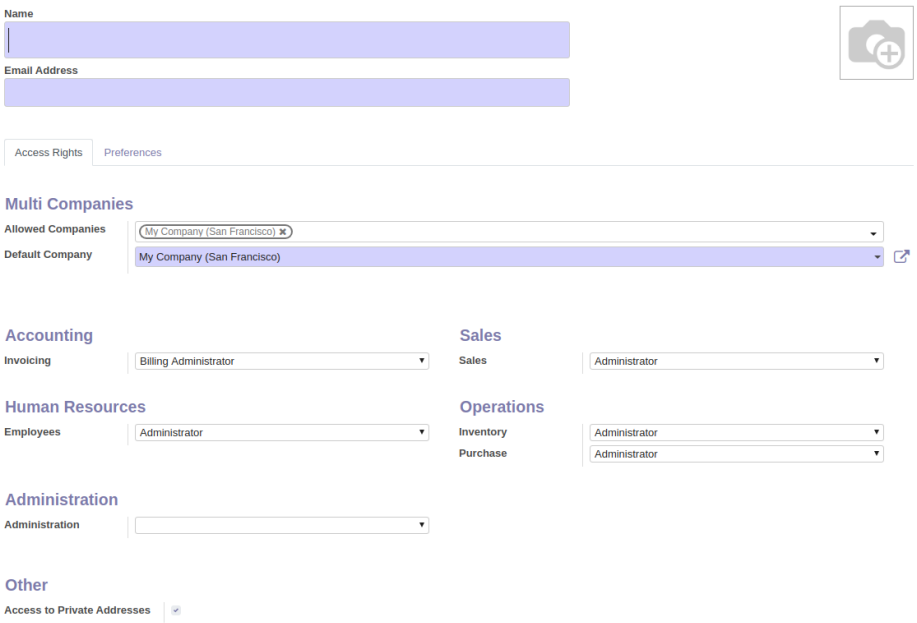
- Usuaris
- Configuració de l'empresa
- Idiomes
- Format dels informes
- Mode desenvolupador

Usuaris

Permet crear nous usuaris i configurar els seus privilegis de manera individual (figura 2.48).

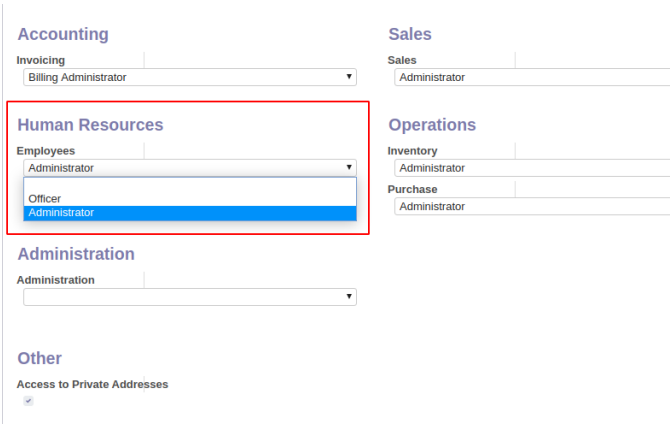
Més endavant, al punt "Importació de dades a Odoo. Usuaris i treballadors" d'aquest mateix contingut, es veurà la possibilitat d'importació massiva d'usuaris.

FIGURA 2.48. Creació d'un usuari en Odoo



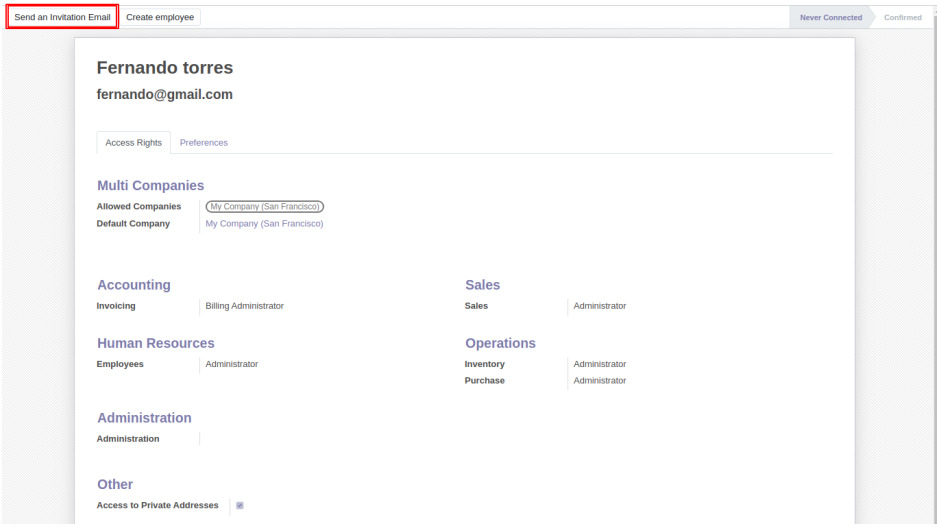
Destaca en aquest punt la gestió dels **permisos** de cada usuari. Cada mòdul instal·lat a Odoo ofereix en aquesta pantalla un o diversos grups. Assignar al nou usuari un grup d'un mòdul implica concedir-li privilegis en aquest mòdul (figura 2.49).

FIGURA 2.49. Diferents grups del mòdul de recursos humans



Una vegada creat el nou usuari, pot **enviar-se un correu electrònic** a la seva adreça amb les credencials pel seu primer accés a l'aplicació (figura 2.50).

FIGURA 2.50. Enviar correu electrònic amb les credencials

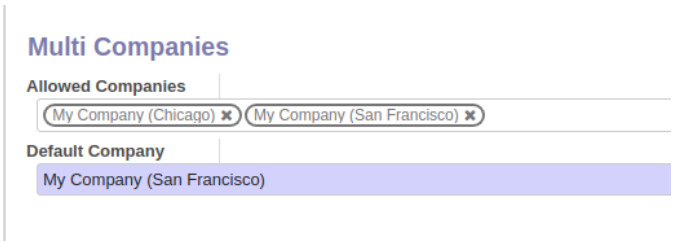


L'usuari "admin" es genera de manera automàtica amb la instal·lació, i tindrà tots els privilegis per defecte en tots els mòduls.

Configuració de l'empresa

Odoo permet la creació d'una o diverses empreses relacionades. Per exemple, poden generar-se dues delegacions diferents de la mateixa empresa. Cada usuari tindrà una empresa principal associada, i permís per a poder veure una o més empreses (figura 2.51).

Figura 2.51. Empreses associades a un usuari



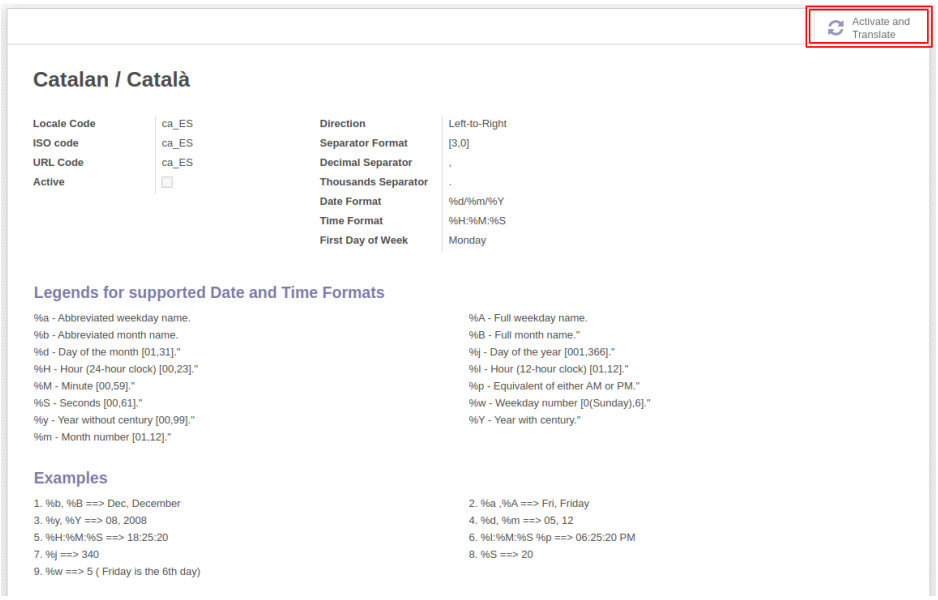
La versió Community no permet que aquestes multiempreses facin transaccions entre d'elles, però si la versió Enterprise.

Idiomes

En el procés d'instal·lació s'ha decidit l'idioma general de l'ERP. En l'apartat "idioma" de la configuració d'Odoo, però, **poden afegir-se més**.

Cada usuari podrà decidir quina llengua vol a la seva interfície, sempre dins els idiomes configurats en aquest apartat (figura 2.52).

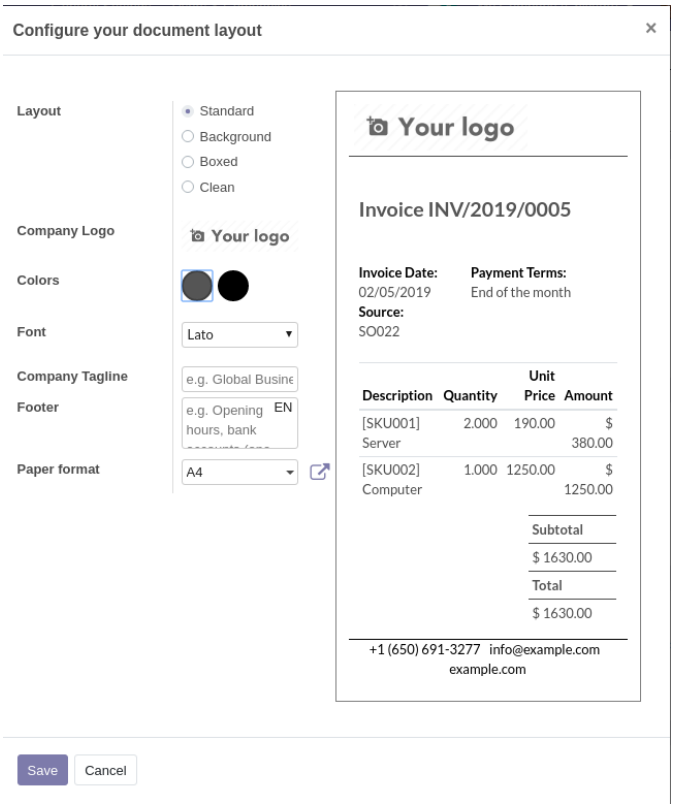
Figura 2.52. Instal·lació d'un nou idioma



Format dels informes

La majoria dels mòduls d'Odoo permeten generar informes en format pdf. Cada informe depèn del mòdul, però tots tenen en comú una mida fixa, i l'encapçalat i el peu de pàgina (figura 2.53). La informació del logo i els mitjans per a contactar amb l'empresa són extrets de la configuració de l'empresa.

Figura 2.53. Configuració dels informes



Mode desenvolupador

Per a poder accedir a **funcionalitats de perfil tècnic** dins Odoo ha d'activar-se el "developer mode". Permet, entre d'altres, accedir a les següents opcions:

- Creació de nous grups d'usuaris dins els mòduls
- Menú "technical" amb la configuració de paràmetres tècnics dins Odoo.
- Accés al mode "superuser"
- Accés a les característiques dels camps de cada vista de formulari (descripció del camp, classe a la qual pertany, detalls tècnics...)
- Instal·lació de mòduls no oficials

Com a resum de la configuració general pot veure's aquest vídeo:

https://player.vimeo.com/video/469120917

2.5.2 Els mòduls no oficials. Instal·lació.

Com tot Programari Lliure, Odoo permet a tercers l'elaboració de nous mòduls, que poden ser instal·lats amb normalitat. Aquests mòduls de tercers serviran per a **crear noves funcionalitats** no existents anteriorment, complementar mòduls existents o introduir particularitats regionals. Encara que qualsevol implantador de manera individual pot generar nou codi, és habitual que la comunitat treballi de manera col·legiada. Els mòduls creats per la comunitat estaran revisats per més gent, s'actualitzaran més ràpidament i seran molt més fiables.

Odoo OCA

Segons la seva web, l'Odoo Community Association (OCA) és una organització sense ànim de lucre que té com a missió promoure l'ús generalitzat d'Odoo i donar suport al desenvolupament col·laboratiu de les funcions d'Odoo (figura 2.54).

Figura 2.54. Odoo Community Association

Els **objectius de l'OCA** són:

- Ajudar i promoure el desenvolupament de programari col·laboratiu d'Odoo.
- Incentivar el desenvolupament d'Odoo i les seves característiques a l'hora de coordinar i organitzar el treball col·laboratiu del programari.
- Ajudar la comunitat defensant els seus interessos i la sostenibilitat dels seus desenvolupaments.
- Promoure l'ús de la solució Odoo.
- Facilitar sinergies, col·laboracions i esforços de recaptació de fons.
- Col·labora activament en la definició dels fulls de ruta de les noves versions de l'eina i la seva implementació.

En Odoo OCA poden trobar-se mòduls de diferent tipologia per a complementar una instal·lació d'Odoo. Aquests mòduls estan categoritzats per categoria, versió i maduresa, i són fàcilment descarregables i instal·lables des de la seva web oficial.

Web oficial i mòduls descarregables

Podeu visitar la web oficial d'Odoo OCA a: odoo-community.org; així com la seva tenda oficial: odoo-community.org/shop.

A continuació, podeu veure el vídeo de presentació de l'associació espanyola d'Odoo (2020):

https://www.youtube.com/embed/7TVAG8chWs0?controls=1

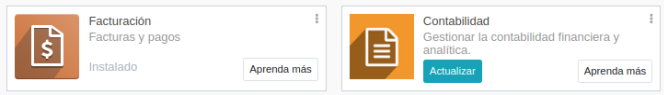
Recerca de nous mòduls (exemples interessants)

Es treballaran dos exemples reals per a mostrar la instal·lació de mòduls de tercers: la comptabilitat a Odoo i la "localització".

Exemple 1: comptabilitat

Quan un usuari comença a utilitzar Odoo community, una de les primeres sorpreses és l'absència d'algunes opcions bàsiques de comptabilitat. De fet, Odoo ofereix la instal·lació diferenciada dels mòduls de facturació i comptabilitat, sent aquest últim un mòdul per a la versió de pagament (figura 2.55).

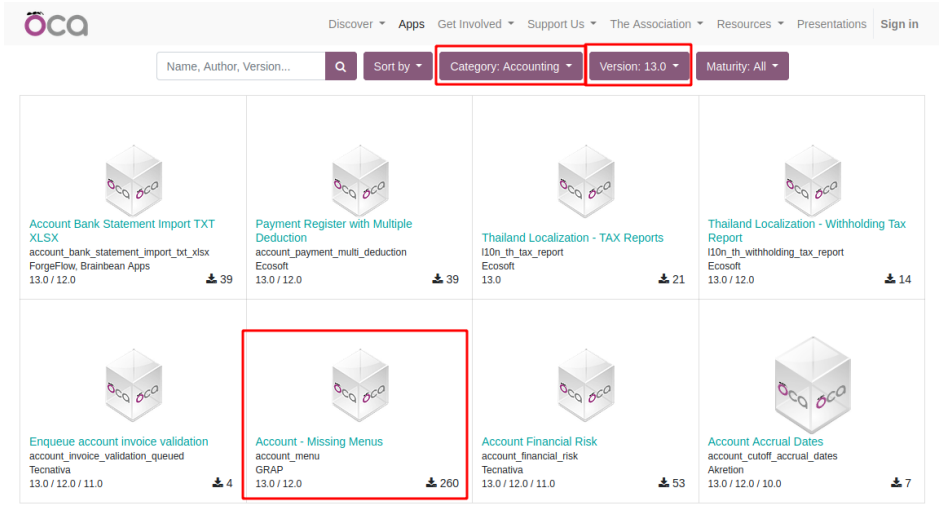
FIGURA 2.55. Facturació vs. Comptabilitat



Un bon implantador **buscarà alternatives** pels seus clients que volen fer servir la versió community. I el primer pas que donarà serà accedir a Odoo OCA, i buscar una aplicació gratuïta i sostinguda per la comunitat.

Una vegada introduïts els criteris de cerca, hi ha l'opció "Account - Missing Menus" (figura 2.56), que afegeix al mòdul de facturació opcions que hi existien fins a la versió 11, i que Odoo va decidir treure en endavant. Pot trobar-se a: bit.ly/37teHpA.

FIGURA 2.56. Account - Missing Menus



Exemple 2: localització

Cada país té regles molt concretes respecte a la seva fiscalitat. Quan un implantador vol configurar una aplicació de propòsit general i adaptar-la al país on treballarà, és molt important que tingui uns mínims coneixements sobre fiscalitat. A tots els mòduls encarregats de realitzar aquesta adaptació se'ls anomena de "localització".

Un bon implantador ha de tenir **coneixements sobre fiscalitat**, com es mostra al vídeo de presentació de novetats fiscals en Odoo 2020:

https://www.youtube.com/embed/r_z_hDwDXCg?controls=1

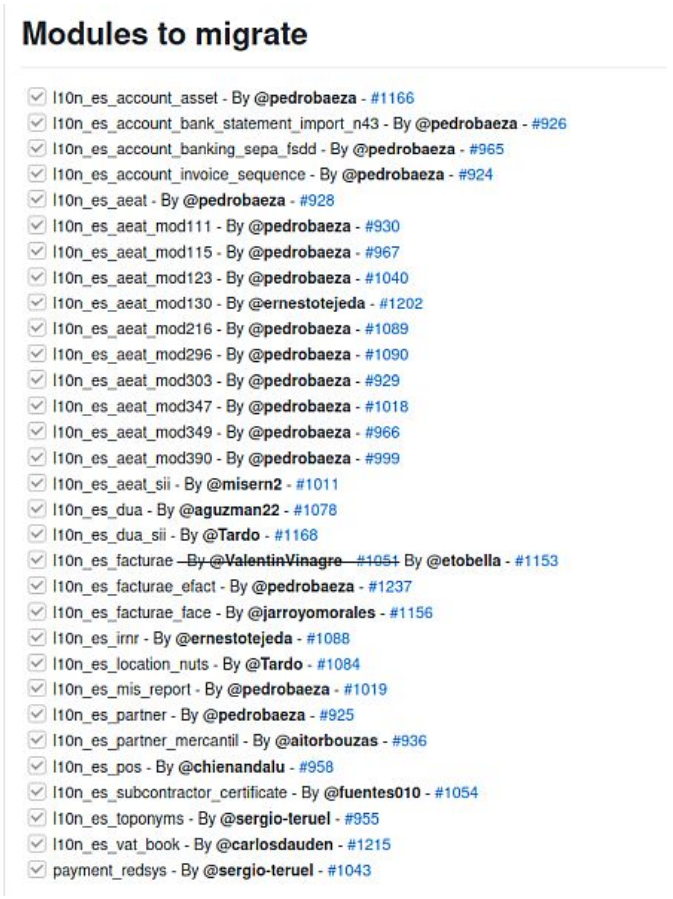
Molts dels processos que ha de portar a terme una empresa en la seva relació amb l'agència tributària poden realitzar-se de manera telemàtica, mitjançant els anomenats "models". Són uns arxius que segueixen un format determinat, i recullen informació sobre l'activitat econòmica de l'empresa. És lògic arribar a la conclusió de què si tota la informació està emmagatzemada a l'ERP de l'empresa, ha de ser possible crear de manera senzilla aquests models. L'associació espanyola d'Odoo (AEODOO), és un grup de treball dins Odoo OCA, i que estan especialitzats en la generació i actualització d'aquests mòduls que adeqüen Odoo a la realitat tributària d'Espanya. Accedint a la pàgina Github de l'AEODOO (bit.ly/2Ho7Brx) pot trobar-se tot el llistat de mòduls necessaris per a fer totes aquestes adaptacions.

l10n (localització)

Els mòduls d'Odoo relacionats amb la localització sempre tindran un nom que comença per l10n.

A la figura següent pot comprovar-se l'AEODOO manté un repositori amb tots els mòduls necessaris. És important destacar que, a partir d'un canvi de versió, tots els grups de treball de l'AEODOO han de migrar els mòduls existents de la versió anterior a l'actual, sent probable que trigui uns mesos arribar a tenir-los tots disponibles (figura 2.57 i figura 2.58).

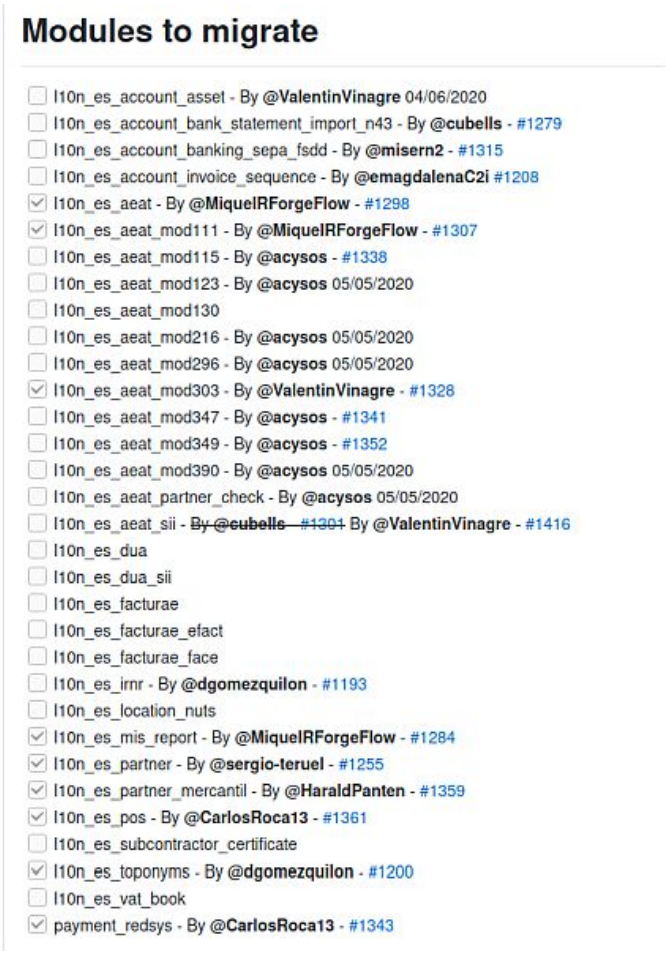
FIGURA 2.57. Llistat dels mòduls de localització espanyola per a Odoo 12 (tots disponibles)



Modules to migrate

- l10n_es_account_asset - By **@pedrobaeza** - #1166
- l10n_es_account_bank_statement_import_n43 - By **@pedrobaeza** - #926
- l10n_es_account_banking_sepa_fsdd - By **@pedrobaeza** - #965
- l10n_es_account_invoice_sequence - By **@pedrobaeza** - #924
- l10n_es_aeat - By **@pedrobaeza** - #928
- l10n_es_aeat_mod111 - By **@pedrobaeza** - #930
- l10n_es_aeat_mod115 - By **@pedrobaeza** - #967
- l10n_es_aeat_mod123 - By **@pedrobaeza** - #1040
- l10n_es_aeat_mod130 - By **@ernestotejeda** - #1202
- l10n_es_aeat_mod216 - By **@pedrobaeza** - #1089
- l10n_es_aeat_mod296 - By **@pedrobaeza** - #1090
- l10n_es_aeat_mod303 - By **@pedrobaeza** - #929
- l10n_es_aeat_mod347 - By **@pedrobaeza** - #1018
- l10n_es_aeat_mod349 - By **@pedrobaeza** - #966
- l10n_es_aeat_mod390 - By **@pedrobaeza** - #999
- l10n_es_aeat_sii - By **@misern2** - #1011
- l10n_es_dua - By **@aguzman22** - #1078
- l10n_es_dua_sii - By **@Tardo** - #1168
- l10n_es_facturae ~~- By @ValentinVinagre - #1051~~ By **@etobella** - #1153
- l10n_es_facturae_efact - By **@pedrobaeza** - #1237
- l10n_es_facturae_face - By **@jarroyomorales** - #1156
- l10n_es_irnr - By **@ernestotejeda** - #1088
- l10n_es_location_nuts - By **@Tardo** - #1084
- l10n_es_mis_report - By **@pedrobaeza** - #1019
- l10n_es_partner - By **@pedrobaeza** - #925
- l10n_es_partner_mercantil - By **@aitorbouzas** - #936
- l10n_es_pos - By **@chienandalu** - #958
- l10n_es_subcontractor_certificate - By **@fuentes010** - #1054
- l10n_es_toponyms - By **@sergio-teruel** - #955
- l10n_es_vat_book - By **@carlosdauden** - #1215
- payment_redsys - By **@sergio-teruel** - #1043

FIGURA 2.58. Estat de la migració dels mòduls de localització espanyola a Odoo 13



Instal·lació de mòduls de tercers

A continuació, es descriurà el procés d'instal·lació d'un mòdul de tercers, concretament el que correspon a l'eina de comptabilitat "account_menu". Els detalls, però, estan desenvolupats al vídeo que hi ha al final d'aquest punt.

Es parteix d'un servidor amb Odoo, concretament d'un servidor Ubuntu Server. Els **passos a portar a terme** seran (figura 2.59):

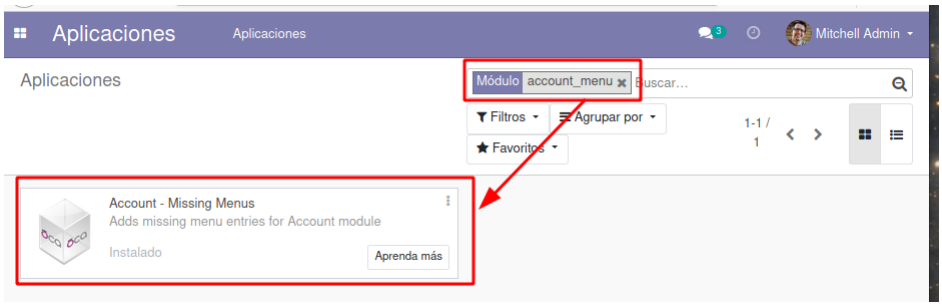
Podeu descarregar el servidor Ubuntu Server, des de la secció "Annexos".

1. Accedir a Odoo OCA, descarregar i descomprimir el mòdul.
2. Enviar la carpeta del mòdul al directori "odoo-custom-addons" del server on està Odoo. Pot fer-se mitjançant un client FTP, terminal (fent servir la comanda scp), o accedint directament al servidor (si fos possible).
3. Reiniciar el servei odoo13. Aquest pas no és obligatori, però evita errors, per exemple en mostrar les icones del mòdul.
4. Accedir al menú de configuració d'Odoo i activar el mode desenvolupador. També pot fer-se afegint a l'URL d'Odoo ?debug=1
5. Al menú d'aplicacions, actualitzar la llista d'aplicacions.
6. Ja pot trobar-se l'aplicació de tercers a Odoo amb normalitat.

És de vital importància que el propietari del servei Odoo (a la instal·lació d'exemple es diu odoo13) tingui **permís sobre els arxius del mòdul**. La recomanació és fer el següent per assegurar-se'n:

```
chown -R odoo13:odoo13 odoo-custom-addons
```

Figura 2.59. Mòdul account_menu disponible per a la seva instal·lació



Com a resum d'aquest punt, pot veure's el següent vídeo:

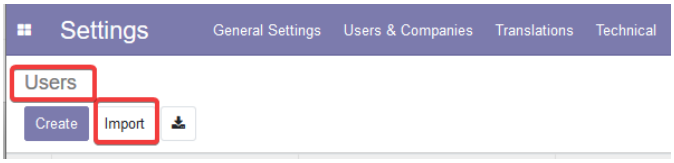
https://player.vimeo.com/video/469120944

2.5.3 Importació de dades a Odoo. Usuaris i treballadors

Una de les qüestions que més preocupa les empreses a l'hora d'implantar un ERP és la **migració de les seves dades**. Totes les empreses disposen d'una o diverses bases de dades (casolanes o professionals), i és responsabilitat de l'implantador portar a terme aquesta tasca. Com a primera opció per a la migració, es podria suggerir actuar directament sobre la base de dades. És una opció molt complicada i arriscada, però, ja que existeixen moltes relacions entre les diferents taules i exigeix un nivell de coneixement i de treball previ per a generar les consultes molt gran.

Per tal d'evitar aquest tipus de migració, Odoo proporciona la possibilitat d'importar dades **a totes les seves pantalles de tots els mòduls** (figura 2.60).

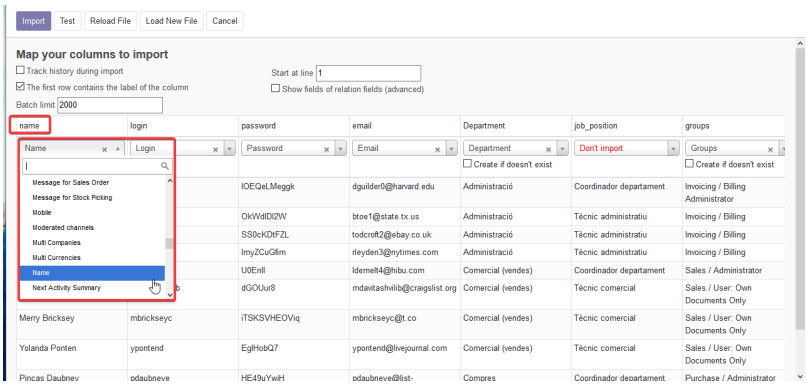
Figura 2.60. Importació de dades a Odoo



Odoo accepta dos formats d'arxiu diferents: **csv** (arxiu pla de text separat per comes) i **xslx** (full de càlcul tipus Microsoft Excel). A l'hora d'afegir columnes a aquest arxiu per a la migració, pot fer-se servir qualsevol camp existent a la base de dades.

Una vegada carregat un arxiu, pot seleccionar-se a quin camp de la base de dades pertany cadascuna de les columnes (figura 2.61).

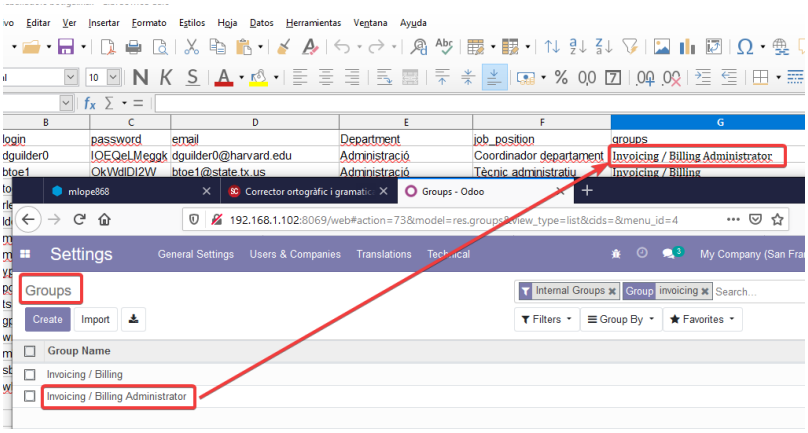
Figura 2.61. Elecció per a una columna de l'arxiu d'importació d'un camp de la base de dades



En aquest sentit, hi ha altres **recomanacions** a tenir en compte:

- Per a camps que es refereixen a categories (p. ex. el lloc de treball d'un treballador), existeix la possibilitat de crear el valor en cas de no existir-hi (*create if doesn't exist*).
- Abans de realitzar-se la importació pot fer-se una comprovació de què no hi haurà problemes (botó Test). En cas d'existir conflictes Odoo mostrarà un missatge donant-hi informació al respecte.
- A l'hora de fer servir categories ja existents, ha de ficar-se el seu nom exacte (figura 2.62). És preferible ficar Odoo en anglès, ja que assegura una compatibilitat total.

Figura 2.62. Per importar una categoria es fa servir el seu nom exacte, tal com surt a Odoo



Usuaris i treballadors

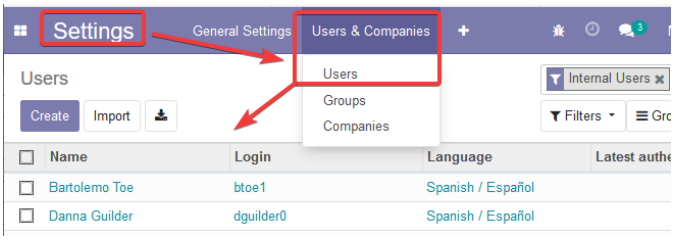
Una de les primeres migracions que ha de portar-se a terme en implantar un ERP és la d'usuaris i treballadors. Primerament, és important diferenciar aquests dos conceptes, ja que no són el mateix:

- Un **usuari** és tota aquesta persona que pot accedir a l'ERP i té diferents atribucions, com ara introduir vendes, compres, gestionar enviaments o contractacions.
- Un **treballador** o empleat és un membre de l'empresa, que com a tal ha de figurar al mòdul de recursos humans de l'ERP.

El model de negoci de molts ERP de pagament consisteix a cobrar a l'empresa per cadascun dels usuaris que pot connectar-s'hi. Serà, per tant, de vital importància fer una bona diferenciació entre usuaris i treballadors per a **no generar pagaments innecessaris**.

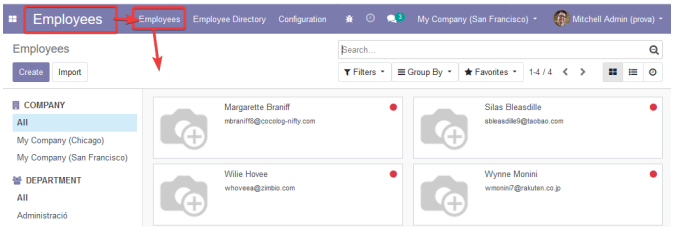
Tota la **gestió d'usuaris** (alta, importació, edició...) es porta a terme a la secció de configuració d'Odoo (figura 2.63).

Figura 2.63. Odoo. Usuaris



Tota la **gestió de treballadors** (alta, importació, edició...) es porta a terme al mòdul de recursos humans d'Odoo (figura 2.64).

Figura 2.64. Odoo. Empleats



A continuació pot veure's un vídeo on es realitza un exemple de migració d'usuaris i treballadors a Odoo:

https://player.vimeo.com/video/469120947

2.6 Els mòduls d'Odoo

Per a portar a terme totes les operacions de venda, consultoria, implantació i formació en un ERP és imprescindible conèixer aquesta eina. Aquest coneixement s'adquireix fonamentalment per l'experiència, però és molt important partir d'una bona base. En aquest material volem destacar dos processos en particular: el **cicle de la comanda** i la **comunicació en línia** (o *online*). Però cal saber que hi ha altres possibilitats que, encara que no s'esmentin, també són importants, com ara els processos de fabricació o la relació amb el client.

Manuals d'ús

El contingut a continuació no pretén ser un manual d'ús, només una petita mostra de les possibilitats que Odoo ofereix a les empreses. Els manuals d'usuari d'Odoo poden trobar-se a bit.ly/31vK2UY.

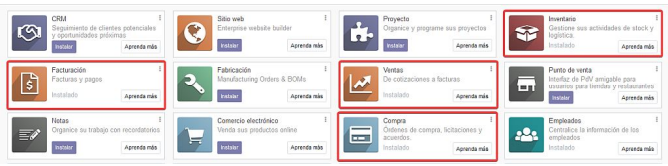
2.6.1 Cicle de la comanda: vendes, compres i inventari

El cicle de la comanda del client s'inicia quan aquest fa una sol·licitud de materials o productes a un proveïdor, és a dir, quan fa una comanda, i acaba quan rep el producte. També afegirem a aquest procés l'aprovisionament per part del proveïdor, ja que no podrà subministrar un bé si no el té al seu magatzem amb anterioritat.

Pel que fa a Odoo, els **mòduls que formen part d'aquest procés** són (figura 2.65):

- Mòdul d'inventari
- Mòdul de compres
- Mòdul de vendes
- Mòdul de facturació

FIGURA 2.65. Mòduls implicats en el cicle de la comanda

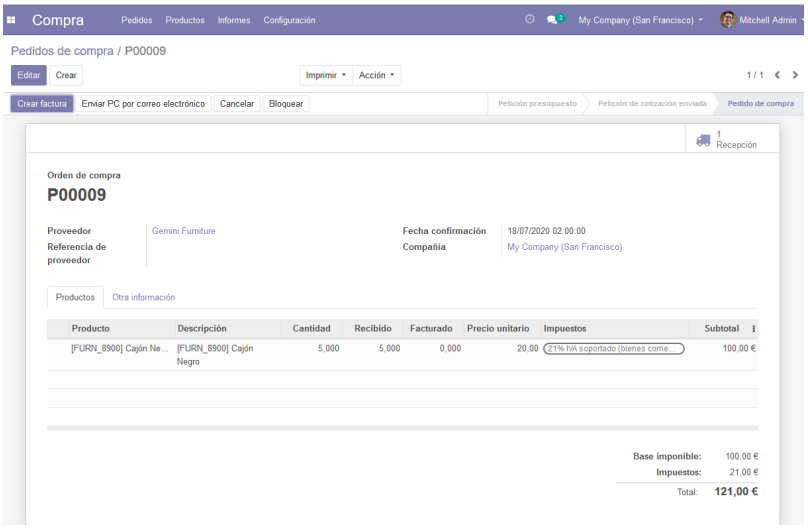


Convé recordar que Odoo és una aplicació modular, i per tant, la funcionalitat de cada mòdul serà **independent de la resta** (encara que puguin interactuar quan estan instal·lats). És per això que hi haurà objectes, com ara els productes o les empreses (proveïdors o clients), que estan al nucli d'Odoo, en comptes d'estar a un mòdul en concret.

Compres

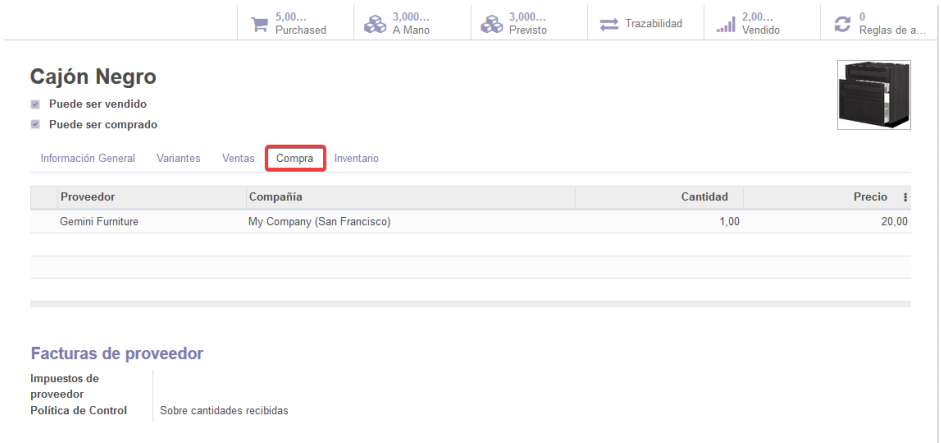
El mòdul de compres ajuda a una empresa a automatitzar el seu aprovisionament, sigui per a produir béns a partir de matèries primeres, sigui per a vendre'ls a continuació (figura 2.66).

FIGURA 2.66. Comanda de compra



Permetrà emmagatzemar, per a cada producte, diferents proveïdors, amb la seva quantitat mínima i preu per unitat. Això facilita generar comandes de compra per **aprovisionar el magatzem** de l'empresa (figura 2.67).

FIGURA 2.67. Preu de compra per a un producte



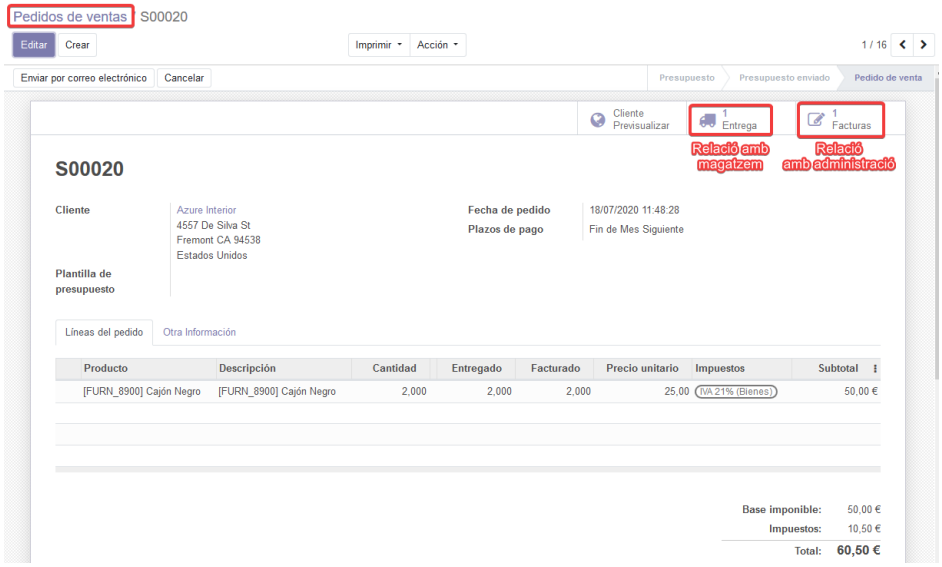
Vendes

Aquest mòdul proporciona eines per a la venda de productes o serveis (figura 2.68). Estèticament és molt similar al mòdul de compres, i gestiona la creació de pressupostos, acceptació de pressupostos i conversió en comanda, enviament de la comanda al client.

A més a més, el comercial que realitza la venda pot posar en marxa l'**enviament** (que després serà atès pels tècnics de magatzem), i la **factura** (que després serà

atesa pels tècnics de comptabilitat).

FIGURA 2.68. Mòdul de vendes

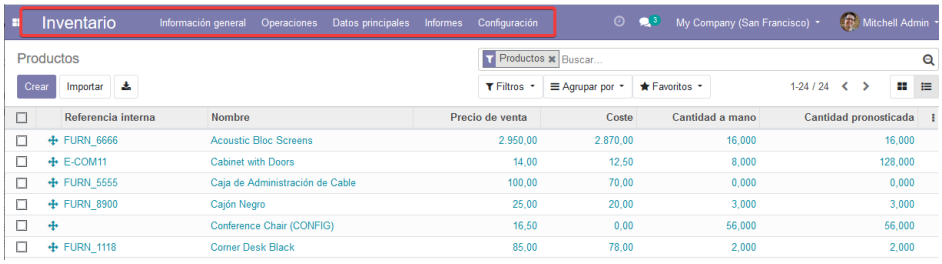


Inventari

El propòsit d'aquest mòdul és la **gestió de l'estoc** de productes de l'empresa (figura 2.69). Permet crear un o diversos magatzems, zones dins cada magatzem, moure productes d'un magatzem o zona a un altre, aprovisionar el magatzem de manera massiva...

Aquest mòdul té una importància vital en empreses grans, ja que ajuda a gestionar les existències, prevenint problemes com ara ruptures d'estoc o duplicació de compres.

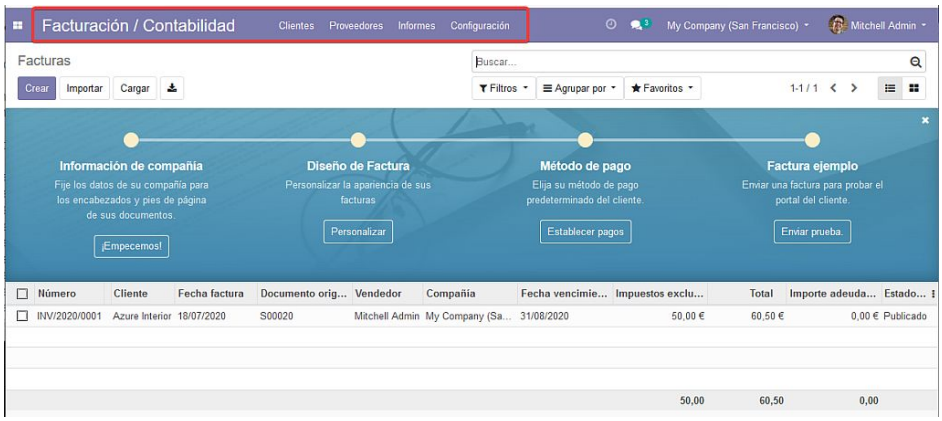
FIGURA 2.69. Mòdul d'inventari



Facturació

Gestiona els **pagaments i l'emissió** de factures de l'empresa (figura 2.70). Pot completar-se amb mòduls de comptabilitat i localització, per tal de generar la documentació necessària per a l'agència tributària.

Figura 2.70. Mòdul de facturació



La informació per a la gestió dels mòduls de tercers es troba al punt "Els mòduls no oficials. Instal·lació" d'aquest mateix contingut.

El següent vídeo descriu breument el funcionament dels mòduls de compres, vendes, inventari i facturació amb un exemple:

https://player.vimeo.com/video/469121000

2.6.2 Comunicació en línia: el mòdul web

Actualment, totes les empreses necessiten presència al web. La manera de comprar dels consumidors ha canviat, i és molt alt el percentatge d'aquests que consulten internet abans de decidir-se. Odoo ERP ofereix un senzill mòdul web que serà la **cara pública de l'empresa a la xarxa de xarxes**. Però la necessitat de presència web no acaba aquí, ja que són moltes les empreses que volen vendre els seus productes en línia (*online*). Una vegada es té la infraestructura web, la resta de l'empresa no necessita grans canvis per a afegir aquest **nou canal de venda**, i els beneficis són grans.

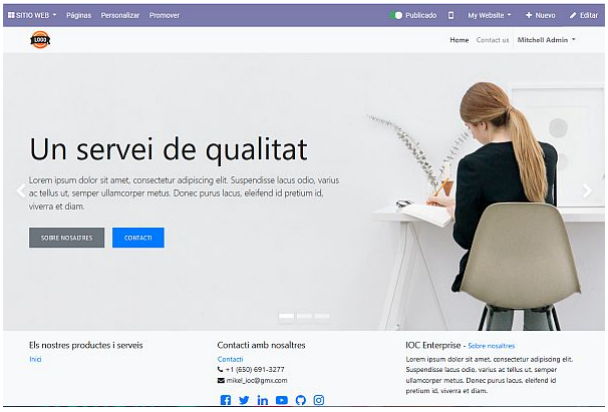
Crisis i oportunitats

Problemes actuals, com possibles confinaments provocats per pandèmies, obren grans oportunitats de negoci per a implantadors de solucions que incorporin botiga web a una empresa.

Web de l'empresa

Per a crear una senzilla web s'ha d'instal·lar a Odoo el mòdul **"Lloc web"**. Una vegada instal·lat haurà de triar-se una plantilla entre dues existents, i a continuació ja pot començar l'edició (figura 2.71).

Figura 2.71. Pàgina web creada amb Odoo

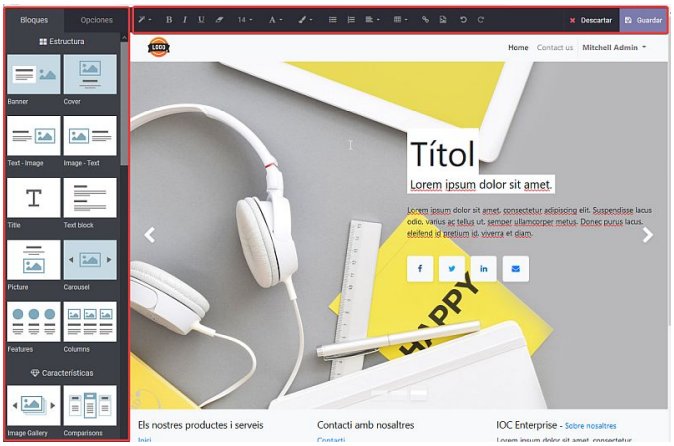


El mòdul web conté un editor per a generar elements estàtics i fins i tot dinàmics (com ara carrusels d'imatges). Encara que és important que la primera edició de la web la realitzi l'empresa amb el consultor, és molt probable que no necessitin ajuda per a canvis posteriors (figura 2.72).

Altres plantilles

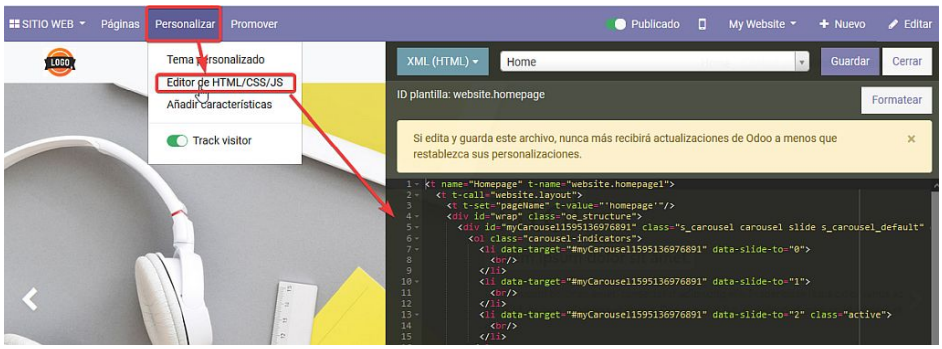
Poden trobar-se més plantilles a la pàgina oficial d'Odoo: bit.ly/3mc5lml.

Figura 2.72. Editor de contingut de la web



Per a **canvis estètics** més importants dins la plantilla, existeix la possibilitat d'editar el seu codi xml (figura 2.73). Òbviament, només s'aconsella accedir a aquesta secció a tècnics que dominin HTML, CSS i JS. Algunes de les plantilles fan servir el *framework* Bootstrap 4, per tant, també s'aconsella als implantadors que tinguin coneixements en aquesta tecnologia.

Figura 2.73. Editor del codi xml d'una plantilla



Boostrap

Es pot trobar informació sobre Boostrap a: bit.ly/37vAixC.

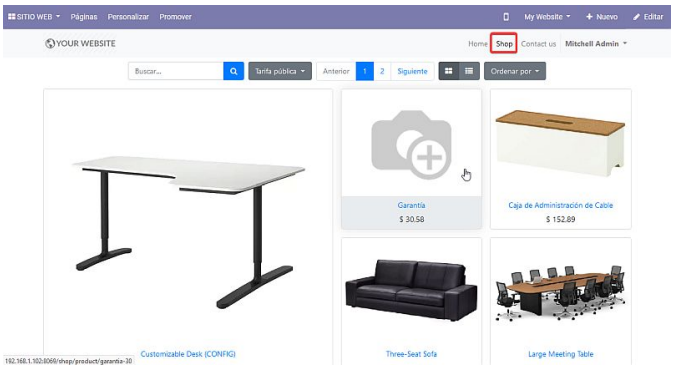
Odoo ofereix també la possibilitat de crear entrades de bloc, enquestes, formularis... totes aquestes funcionalitats estan deshabilitades per defecte, hauran d'instal·lar-se els mòduls corresponents per a poder utilitzar-les (figura 2.74).

FIGURA 2.74. Afegir contingut a la web

Botiga en línia o 'online'

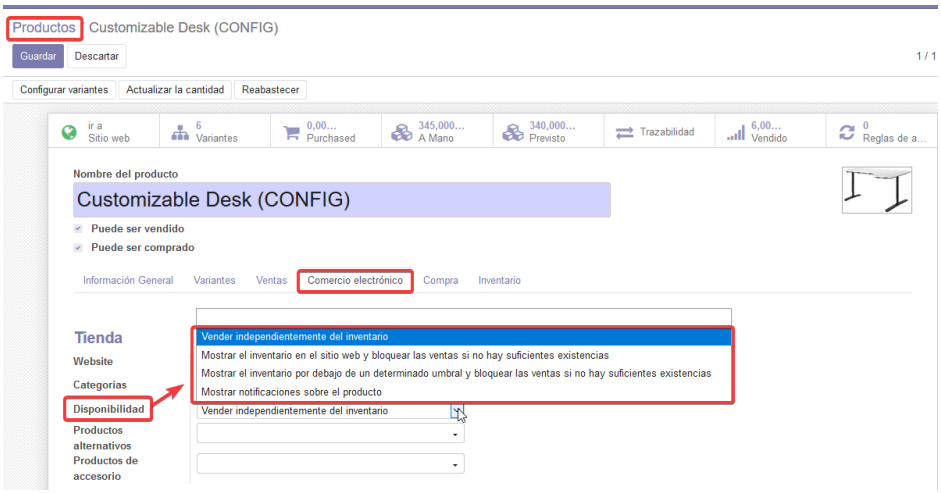
Una vegada Odoo disposa d'una estructura de productes, i els mòduls de compra, venda, inventari i facturació instal·lats, crear una botiga web és molt senzill. Realment, instal·lar el mòdul de comerç electrònic només posa a disposició de la web tot el **catàleg de productes existents**, la resta, tot el canal de venda, enviament i facturació, ja existeix (figura 2.75).

FIGURA 2.75. Botiga en línia a Odoo



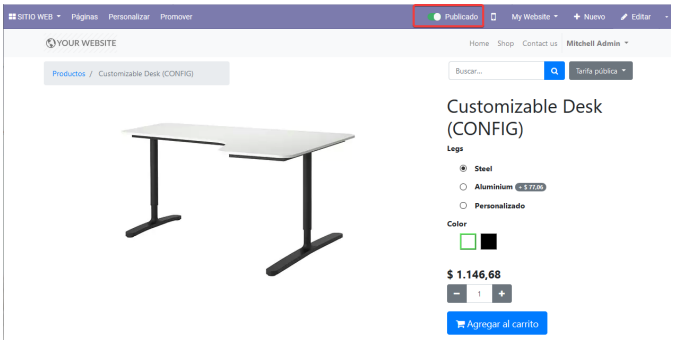
Una vegada instal·lat el mòdul, es genera una nova pestanya a les fitxes de producte, per tal de poder configurar el seu comportament. Entre d'altres, es configura si Odoo permetrà vendre un producte sense estoc, o si es publicarà el nombre d'existències (figura 2.76).

FIGURA 2.76. Nova pestanya de venda en línia a la fitxa del producte



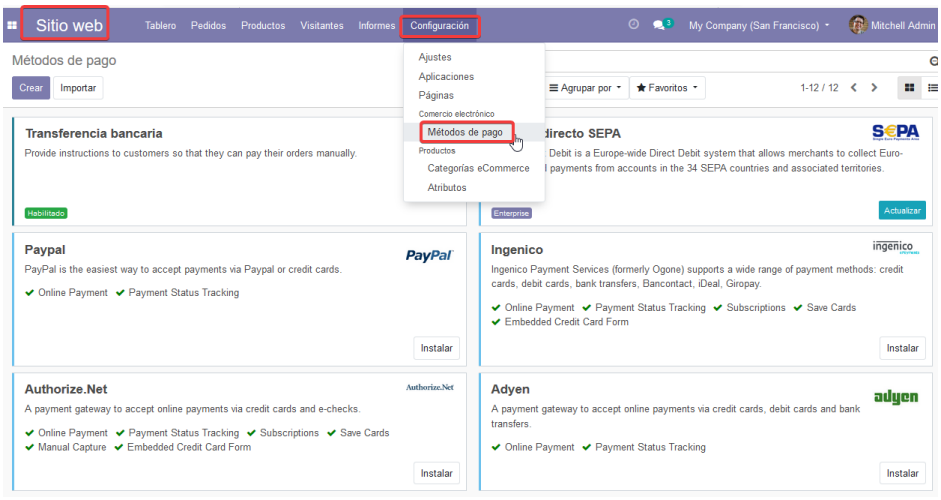
A la web es genera automàticament la secció **"shop"** per accedir a la botiga. Qualsevol producte o servei de la base de dades pot publicar-se, només ha d'activar-se aquesta opció a la seva fitxa (figura 2.77).

FIGURA 2.77. Fitxa del producte a la web



Per últim, destacar que a la secció de configuració de la web es poden modificar les **opcions de pagament**. Per defecte, només ve activada l'opció de transferència bancària, però poden fer-se servir diverses passarel·les de pagament professionals (figura 2.78).

FIGURA 2.78. Edició dels mètodes de pagament



El següent vídeo descriu breument el funcionament dels mòduls de web o comerç electrònic:

https://player.vimeo.com/video/469121009

Sistemes ERP-CRM. Explotació i adequació

Isidre Guixà Miranda i Mikel López Villarroya

Sistemes de gestió empresarial

Índex

Introducció **5**

Resultats d'aprenentatge **7**

1 Explotació i adequació. Odoo **9**
1.1 La base de dades d'Odoo . . . 12
1.1.1 Pas previ: connexió a la base de dades mitjançant pgadmin (vídeo) . . . 12
1.1.2 Identificar les taules . . . 13
1.1.3 Accés de només lectura a la base de dades . . . 17
1.1.4 Accés a PostgreSQL des d'aplicacions clients (vídeo) . . . 23

2 Desenvolupament de mòduls a Odoo **25**
2.1 Generació d'un entorn de desenvolupament amb l'IDE Pycharm (vídeo) . . . 25
2.2 Descripció del mòdul d'exemple: "manteni" . . . 26
2.3 Creació d'un mòdul a partir de zero . . . 27
2.3.1 Arxius de presentació d'un mòdul . . . 28
2.4 El model d'Odoo . . . 30
2.4.1 Disseny de classes amb Dia . . . 30
2.4.2 models.py . . . 32
2.4.3 Atributs de classe . . . 33
2.4.4 Camps possibles a Odoo . . . 34
2.5 La vista a Odoo . . . 42
2.5.1 Declaració de la vista . . . 42
2.5.2 Sintaxi genèrica i tipus de vistes . . . 43
2.5.3 Accions . . . 53
2.5.4 Generació de menús . . . 54
2.6 El controlador a Odoo . . . 56
2.6.1 Mètodes ORM més comuns . . . 57
2.6.2 Els decoradors . . . 57

Introducció

Una vegada efectuada la implantació tècnica d'un ERP, el programari ja hauria d'estar en condicions de ser utilitzat per l'organització per donar sortida a les necessitats. Però això no acostuma a passar, ja que les organitzacions, tot i que en moltes ocasions s'adapten al màxim a l'ERP, tenen necessitats que l'ERP no admet.

Per aquest motiu, per a tots els ERP es necessiten professionals programadors que siguin capaços d'adequar-los a les necessitats de l'organització. Com que hi ha un gran ventall d'ERP-CRM i és impossible abastar les tecnologies que cadascun d'ells utilitza, hem hagut de decidir-nos per un, per dur a la pràctica allò que cal fer en qualsevol posada en explotació d'un ERP-CRM. El producte escollit ha estat l'ERP-CRM de codi obert Odoo, en la seva versió 13.

Aquesta unitat està destinada a introduir-nos en l'arquitectura MVC (patró model-vista-controlador) facilitada pel *framework* OpenObject en el qual es basa Odoo, i s'ha estructurat en dos blocs.

L'apartat **"Explotació i adequació. Odoo"** introdueix els conceptes bàsics que cal tenir en compte en un procés d'explotació i adequació. És molt important conèixer la base de dades del nostre ERP, ja que ajuda a identificar on està la informació, i a poder extreure-la de manera lògica. S'aprendrà a generar un usuari de la base de dades amb privilegis de lectura, i fer-lo servir per a extreure dades de l'ERP. Amb aquesta informació es poden generar informes, taulers de comandament o similar.

El segon apartat, **"Desenvolupament de mòduls a Odoo"**, s'endinsa en el disseny del model de dades en OpenObject i la seva implementació en la base de dades PostgreSQL. S'estudiarà en detall la sintaxi i les normes per tal de generar un mòdul d'Odoo totalment funcional. Primer es generarà un diagrama UML del mòdul, per, a continuació, generar el model en llenguatge Python, les vistes en format XML i algunes nocions del controlador.

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

- La implantació tècnica d'Odoo. El seu coneixement s'adquireix a la unitat "Implantació tècnica de sistemes ERP-CRM: Odoo" del mòdul professional *Sistemes de gestió empresarial*.
- El llenguatge XML. El seu coneixement s'adquireix en el mòdul professional *Llenguatges de marques i sistemes de gestió de la informació*.
- La programació orientada a objectes. El seu coneixement s'adquireix en les unitats formatives relatives a programació orientada a objectes del mòdul professional *Programació*.
- El llenguatge Python. Realment no és necessari un coneixement molt pro-

fund, ja que totes les funcions aplicades estan convenientment explicades, però és important tenir la base en programació orientada a objectes per entendre les explicacions.

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, cal adreçar-s'hi i estudiar-lo. Una vegada estudiats els continguts del material en paper i del material web, s'han de desenvolupar les activitats web.

Resultats d'aprenentatge

En finalitzar aquesta unitat l'alumne/a:

1. Fa 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.
    - Exporta dades.
    - Automatitza les extraccions de dades mitjançant processos.
    - Documenta les operacions realitzades i les incidències observades.
2. Desenvolupa components per a un sistema ERP-CRM analitzant i utilitzant el llenguatge de programació incorporat.
    - Reconeix les sentències del llenguatge propi del sistema ERP-CRM.
    - Utilitza els elements de programació del llenguatge per crear components de manipulació de dades.
    - Modifica components de programari per afegir noves funcionalitats al sistema.
    - Integra els nous components de programari en el sistema ERP-CRM.
    - Verifica el funcionament correcte dels components creats.
    - Documenta tots els components creats o modificats.

1. Explotació i adequació. Odoo

A la secció "Annexos" del web del mòdul, dintre de "Recursos de programari", podeu trobar una còpia dels diversos programaris als quals es fa referència en aquests materials.

Els diversos enllaços proporcionats corresponen a les versions més actuals, a la redacció d'aquests materials.

L'explotació i adequació de l'ERP d'una organització és una tasca imprescindible, ja que garanteix que el programari es mantingui en condicions de ser utilitzat per l'organització per tal de donar sortida a les seves necessitats. Per poder-ho dur a terme, per una banda cal identificar les necessitats (tasca pròpia de consultors) i, per una altra, tenir un coneixement profund de l'ERP, tant en les funcionalitats que facilita (tasca de consultors i implantadors) com en les qüestions tècniques vinculades a l'ERP (tasca d'analistes i programadors).

La nostra tasca com a **programadors** consistirà a conèixer l'arquitectura de l'ERP i les eines de desenvolupament que s'han d'utilitzar per poder fer el vestit a mida que necessita l'organització. El gran ventall d'arquitectures i d'eines vinculades als ERP fa impossible efectuar un aprenentatge estàndard d'explotació i adequació d'ERP i, per tant, ens centrarem a clarificar els punts clau que s'han de tenir en compte i els posarem en pràctica sobre la **versió 13 d'Odoo**.

Odoo és un programari de gestió empresarial desenvolupat sobre el *framework* OpenObject de tipus **RAD** (*Rapid Application Development*). La facilitat dels entorns RAD rau en el fet que el desenvolupament d'aplicacions és molt simple per al programador, de manera que amb poc esforç es poden obtenir aplicacions d'altes prestacions.

L'OpenObject facilita diversos components que permeten construir l'aplicació:

- La **capa ORM** (*Object Relational Mapping*), entre els objectes Python i la base de dades PostgreSQL. El dissenyador-programador no efectua el disseny de la base de dades; únicament dissenya classes, per a les quals la capa ORM d'OpenObject n'efectuarà el mapatge sobre l'SGBD PostgreSQL.
- Una **arquitectura MVC** (model-vista-controlador) en la qual el model resideix en les dades de les classes dissenyades amb Python, la vista resideix en els formularis, llistes, calendaris, gràfics… definits en fitxers XML i el controlador resideix en els mètodes, definits en les classes, que proporcionen la lògica de negoci.
- Un **sistema de fluxos de treball** o *workflows*.
- **Dissenyadors d'informes**.
- Facilitats de **traducció** de l'aplicació a diversos idiomes.

El nostre objectiu és conèixer com es dissenya i s'implementa el model de dades en OpenObject. Abans, però, caldrà aprofundir en el coneixement de la **base de dades** d'una empresa d'Odoo, amb dues finalitats:

- Conèixer la relació existent entre els seus elements (taules i columnes) i els elements que observem en qualsevol dels formularis i informes d'Odoo.
- Saber com accedir a les dades de l'empresa atacant directament la base de dades.

Una vegada coneguda l'estructura d'una base de dades d'Odoo, ens podem endinsar en el **disseny del model** en Odoo, i ho fem en dues fases:

1. Utilitzem l'eina de diagramació Dia, que possibilita la generació d'un diagrama UML i a partir de l'eina ens iniciem en el disseny del model de dades d'Odoo.
2. Ens endinsem en el disseny del model de dades d'Odoo utilitzant el llenguatge Python.

La gran diversitat de funcionaments de les empreses fa que sigui altament improbable que un ERP estigui ideat per donar sortida a totes les **necessitats empresarials**. Quan es detecta una funcionalitat no coberta per l'ERP, hi ha tres camins per trobar-hi una solució. Per una banda, es pot adequar l'ERP per donar resposta a les funcionalitats requerides. Per l'altra, es pot adequar el funcionament de l'empresa a les funcionalitats facilitades per l'ERP. O bé un tercer camí, basat en la combinació dels dos camins anteriors.

La detecció de les **funcionalitats de l'empresa no admeses** per l'ERP que es vol implantar és una de les tasques principals en el procés de consultoria.

El fet que l'empresa canviï el seu funcionament per **adaptar-se a l'ERP**, tot i semblar molt brusc, és una solució que s'adopta en moltes ocasions, sobretot quan els consultors són capaços de demostrar als responsables de l'organització que el nou funcionament té clars avantatges respecte als mètodes emprats en aquell moment per l'empresa.

Cal tenir en compte que, a vegades, l'empresa pot haver establert els seus funcionaments –en temps passats– coaccionada per les possibilitats del sistema informàtic vigent en aquell moment, i aquests funcionaments han esdevingut, amb el pas del temps, maneres de fer fonamentals i intocables en l'organització. En aquestes ocasions, la mà esquerra dels consultors i implantadors és fonamental per dur a terme la implantació de l'ERP amb èxit.

Tot i això, no sempre es poden adequar els mètodes de l'empresa a les funcionalitats facilitades per l'ERP i, en conseqüència, cal adequar l'ERP, fet que pot comportar la necessitat de desenvolupar mòduls específics que es puguin acoblar a l'ERP i/o retocar mòduls (formularis i informes) de l'ERP.

L'adequació d'un ERP a través del **desenvolupament de mòduls específics** es pot dur a terme, en principi, amb qualsevol llenguatge de programació i accedint directament a la base de dades de l'ERP. Fer-ho així, però, comporta **inconvenients**:

- Accedir directament a la base de dades implica tenir un coneixement total de la base de dades i de les relacions existents entre els seus components. Aquestes relacions poden canviar en una actualització de l'ERP.
- Els mòduls desenvolupats amb un llenguatge de programació forà a l'ERP impliquen tenir peces fora de l'ERP, excepte si som capaços d'emular la interfície de l'ERP i d'introduir les noves opcions a l'arbre de menús de l'ERP.

El **retoc de mòduls** (formularis i informes) de l'ERP també té inconvenients:

- Només és possible si disposem del codi font de l'ERP (opció vàlida en programaris de codi obert).
- Els retocs efectuats han de garantir-ne la continuïtat en properes actualitzacions de l'ERP sense necessitat de tornar-los a desenvolupar.

Els diversos inconvenients presentats ens porten a la necessitat de conèixer i utilitzar els mètodes de desenvolupament i/o retoc de mòduls de l'ERP facilitats pel fabricant. A més, cal efectuar els processos d'**actualització de dades** (altes, baixes i modificacions) a través de les regles de negoci de l'ERP, que tenen en compte totes les implicacions, i mai amb accessos directes a la base de dades.

Cal tenir en compte, també, que a vegades pot ser convenient accedir a les dades en mode consulta per extreure'n informació. En aquests casos, tenint coneixement de l'estructura de la base de dades, l'accés directe a la base de dades pot ser adequat, amb la precaució que l'estructura pot canviar en futures actualitzacions de l'ERP.

Quan l'organització necessita informes que l'ERP no facilita...

Si la necessitat dels informes és **puntual**, és perfectament lícit desenvolupar-los amb qualsevol eina, tot accedint directament a la base de dades, i executar-los per aconseguir el resultat esperat. No tenim cap necessitat d'incorporar-los com a opcions de menú de l'ERP ni de guardar-los per a posteriors execucions. En cas de guardar-los, som conscients que després d'haver actualitzat l'ERP, pot ser necessari revisar el seu disseny davant de possibles canvis en la base de dades.

Si la necessitat dels informes és **periòdica**, és lògic desenvolupar-los amb les eines que faciliti l'ERP, utilitzant les seves regles de negoci, i incorporant-los com a noves opcions de menú dins l'ERP. En aquest cas, davant una actualització de l'ERP, és molt possible que les regles de negoci evocades continuïn existint (malgrat que hagi canviat l'estructura de la base de dades) i, en conseqüència, el nostre informe continuï funcionant.

A vegades, dins de les organitzacions hi ha persones que, possiblement per mancances de bones solucions BI vinculades a l'ERP, necessiten accedir a la base de dades per extreure'n informació (mode consulta) i, des de les eines ofimàtiques que dominen (bases de dades ofimàtiques i fulls de càlcul), desenvolupar informes i quadres de comandament (*dashboards*) adequats a les seves necessitats.

En aquest cas, l'accés directe (mode consulta) a la base de dades també és lògic, tot i que l'usuari ha de ser conscient que els seus muntatges es poden veure afectats davant d'una actualització de l'ERP.

Aquests raonaments ens porten a les **conclusions** següents:

- Cal conèixer l'estructura de la base de dades de l'ERP i possibilitar-hi accessos en mode consulta, per extreure'n informació que pugui ser gestionada des d'eines externes.
- Cal conèixer la gestió de regles de negoci de l'ERP.
- Cal conèixer les eines recomanades pel fabricant de l'ERP per al desenvolupament i/o retoc de mòduls de l'ERP, fet que pot anar acompanyat del coneixement de llenguatges de programació.

1.1 La base de dades d'Odoo

La gestió d'una empresa pot fer necessari, en un determinat moment, l'accés a la base de dades de l'ERP per tal d'obtenir informació en un format no facilitat per l'ERP. Per això, és important conèixer el disseny de la BD (base de dades) de l'ERP i, en el nostre cas, de l'Odoo.

A Odoo no hi ha un disseny explícit de la base de dades, sinó que la **base de dades d'una empresa d'Odoo** és el resultat del mapatge del disseny de classes de l'ERP cap a l'SGBD PostgreSQL, que és el que proporciona la persistència necessària per als objectes.

En conseqüència, l'Odoo no facilita cap disseny entitat-relació sobre la base de dades d'una empresa ni tampoc cap diagrama del model relacional. No obstant això, si ens interessa disposar d'un model relacional, es troben moltes eines que ens el poden construir a partir de la base de dades implementada en l'SGBD PostgreSQL.

1.1.1 Pas previ: connexió a la base de dades mitjançant pgadmin (vídeo)

Trobareu la informació prèvia necessària per a configurar el servidor PostgreSQL per tal que escolti connexions d'equips externs, a la unitat "Sistemes ERP-CRM. Implantació", apartat "Implantació tècnica de sistemes ERP-CRM: Odoo", secció "Configuració de PostgreSQL per escoltar connexions entrants".

Tant si estem treballant amb la instal·lació d'Odoo *all in one* per a Windows, com si ens estem connectant a un servidor extern, és important poder establir una connexió amb la base de dades per facilitar la seva comprensió.

A continuació podem veure un vídeo on s'explica aquesta connexió per als dos casos descrits anteriorment amb el client pgadmin.

https://player.vimeo.com/video/472569113

Versió 4 de pgadmin

Encara que la versió *all in one* d'Odoo continua instal·lant la versió 3 de pgadmin, aquesta ja no està sent mantinguda i, per tant, és recomanable anar-se passant a la versió 4. Totes les captures de pantalla que es troben en aquest material corresponen a pgadmin4.

1.1.2 Identificar les taules

Si sorgeix la necessitat de detectar la taula o les taules on resideix una informació determinada, és perquè es coneix l'existència d'aquesta informació gestionada des de l'ERP i, per tant, es coneix algun formulari de l'ERP a través del qual s'introdueix la informació. L'Odoo possibilita, mitjançant el client web, recuperar el nom de la classe Python que defineix la informació que s'introdueix a través d'un formulari i el nom de la dada membre de la classe corresponent a cada camp del formulari. Aquesta informació permet arribar a la taula i columna afectades, tenint en compte dues qüestions:

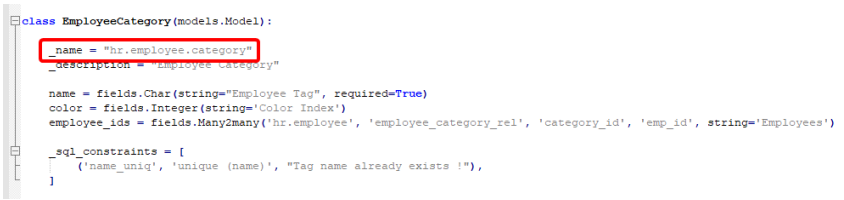
- Els **noms de les classes Python d'Odoo** sempre van en minúscula (s'utilitza el guió baix per fer llegibles els mots compostos) i segueixen la nomenclatura nom_del_modul.nom1.nom2.nom3..., en la qual s'utilitza el punt per indicar un cert nivell de jerarquia. Cada classe Python d'Odoo és mapada en una taula de PostgreSQL amb moltes possibilitats que el seu nom coincideixi amb el nom de la classe, tot substituint els punts per guions baixos.
- Els **noms dels atributs d'una classe Python** sempre van en minúscula (s'utilitza el guió baix per fer llegibles els mots compostos). Cada dada membre d'una classe Python d'Odoo que sigui persistent (una classe pot tenir dades membres calculades no persistents) és mapat com un atribut en la corresponent taula de PostgreSQL amb el mateix nom.

Noms de classes Odoo i taules PostgreSQL corresponents

Farem un exemple per veure la correspondència entre els noms de classes Odoo i corresponents taules PostgreSQL.

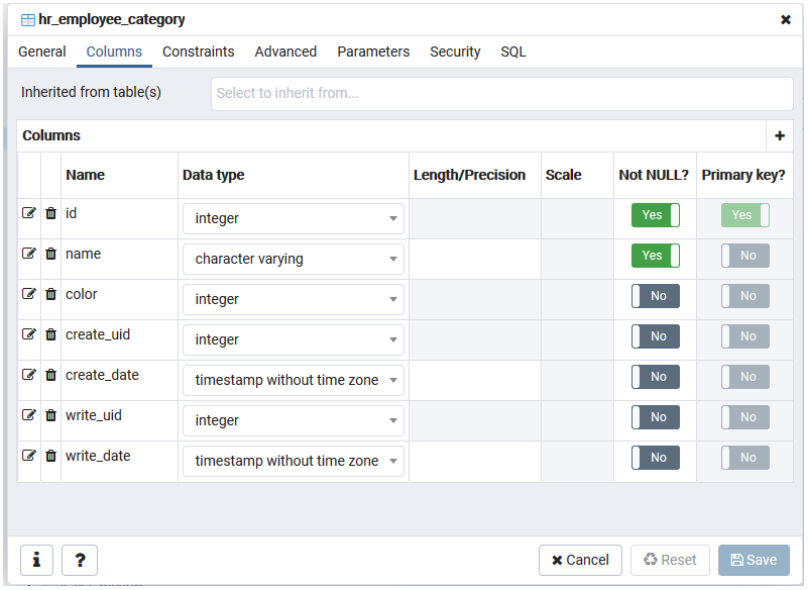
La classe Python `hr.employee.category` està pensada per emmagatzemar les diferents categories d'empleats que hi ha al departament de recursos humans. Si revisem el seu codi font, veiem que el nom segueix les regles esmentades anteriorment (figura 1.1).

FIGURA 1.1. Classe hr.employee.category



Ara anirem a la base de dades (fent servir pgadmin), esperant trobar una taula PostgreSQL amb el mateix nom, substituint els punts per guions baixos (figura 1.2).

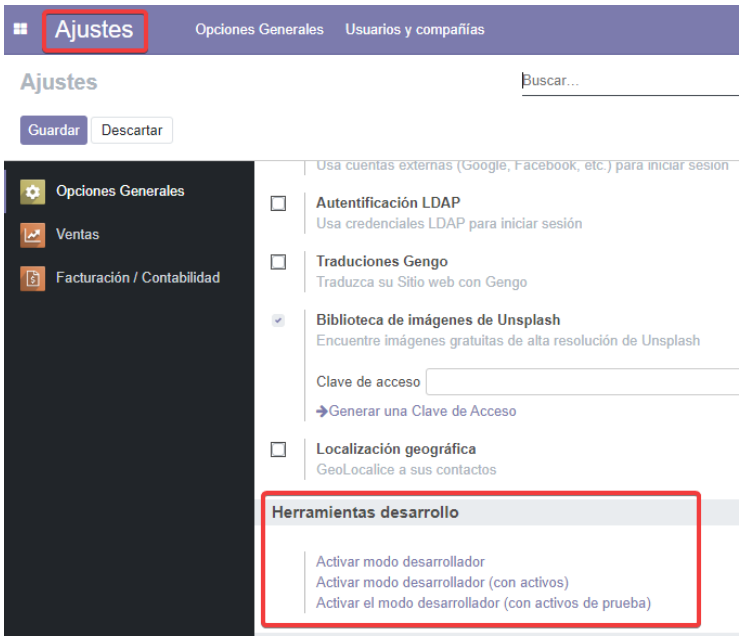
FIGURA 1.2. Taula hr_employee_category a la base de dades



D'aquesta manera, coneixent el nom de la classe i el nom de la dada membre, és molt possible conèixer el nom de la taula i de la columna corresponents. Es pot configurar el client web per tal que informi del nom de la classe i de la dada membre, en situar el ratolí damunt les etiquetes dels camps dels formularis. Aquesta opció es diu ***mode programador***. Tal com es pot veure a la figura 1.3, es troba al menú de configuració.

L'opció *mode programador* ja es va fer servir a la unitat "Sistemes ERP-CRM. Implantació" per a instal·lar mòduls propis.

FIGURA 1.3. Mode programador



Una vegada activat aquest mode, només apropant el ratolí al camp del formulari que ens interessa, veurem un *tooltip* amb informació.

Per al camp `adreça electrònica` obtenim que es tracta del camp `work_email` de l'objecte `hr.employee` i, per tant, si necessitem efectuar una consulta sobre la base de dades accedint a la data de la comanda, sabem que haurem d'anar al camp

`work_email` de la taula `hr_employee` de la base de dades de PostgreSQL (figura 1.4 i figura 1.5).

Figura 1.4. Camp adreça electrònica

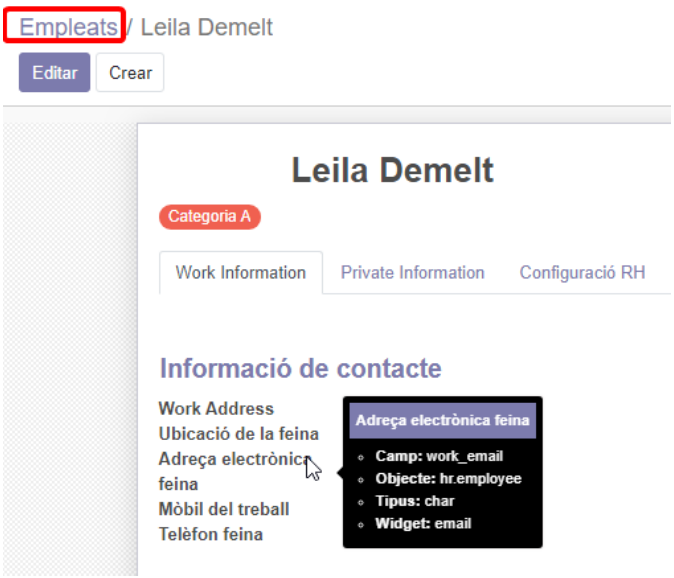
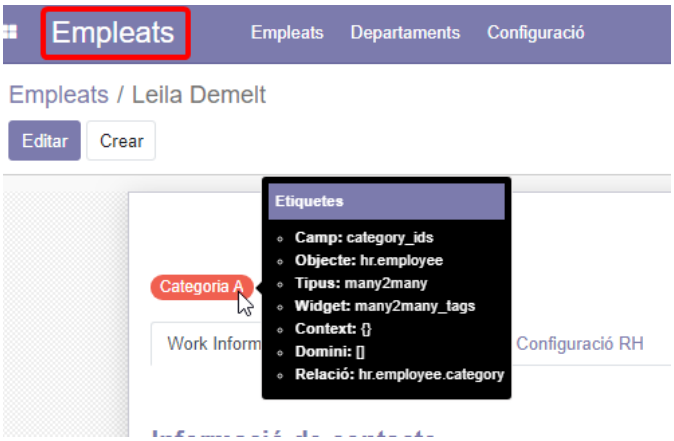


Figura 1.5. 'work_email' a la base de dades

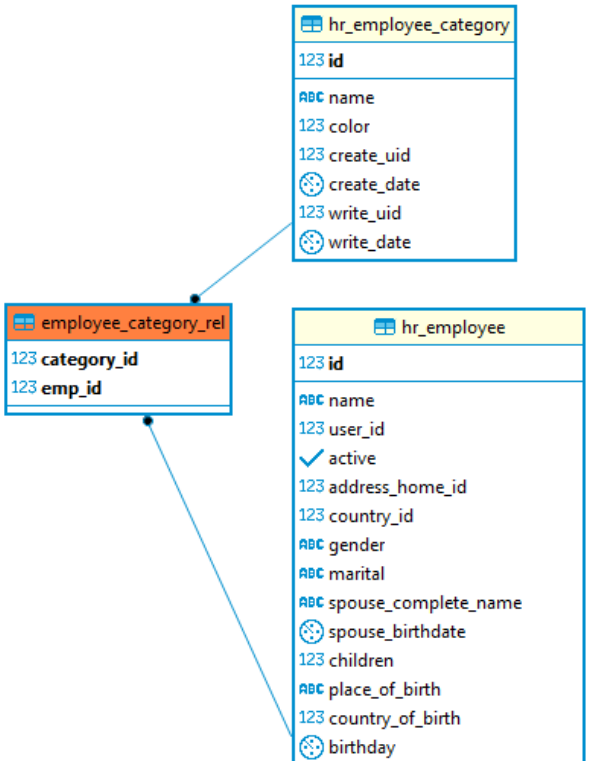
Per al camp `categoria` tenim molta més informació, ja que es tracta del camp relacional `category_ids` de l'objecte `hr.employee`, que fa referència a la relació `hr.employee.category`. Amb aquesta informació, si necessitem efectuar una consulta a la base de dades per accedir a la descripció de la botiga corresponent a la comanda sabem que haurem d'anar al camp `category_ids` de la taula `hr_employee` i a través d'aquest camp, establirem una relació (haurem de veure quina) amb la clau primària de la taula `hr_employee_category` (figura 1.6). Destaca la relació `Many2many`, que veurem més endavant.

FIGURA 1.6. 'Tooltip' amb informació



Si volem veure les relacions que s'estableixen entre les taules `hr_employee` i `hr_employee_category`, obrim una eina que permeti construir aquestes relacions, en aquest cas, ***dbeaver***. Aquest és el resultat (figura 1.7).

FIGURA 1.7. Relacions entre hr_employee i hr_employee_category



Podem veure que hi ha una taula intermediària, amb el nom `employee_category_rel`, que fa de pont entre totes dues taules.

1.1.3 Accés de només lectura a la base de dades

Les empreses acostumen a tenir, entre els seus responsables, usuaris finals que poden efectuar **consultes no previstes** a la base de dades i que, per aconseguir-ho, poden utilitzar eines gràfiques per elaborar consultes o, fins i tot, si són prou espavilats, executar consultes SQL des d'una consola d'accés.

En aquesta situació, cal facilitar als usuaris que calgui un usuari per accedir a l'SGBD PostgreSQL amb els privilegis d'accés, en mode consulta, als objectes de la base de dades que correspongui. S'aconsella seguir dos **passos**:

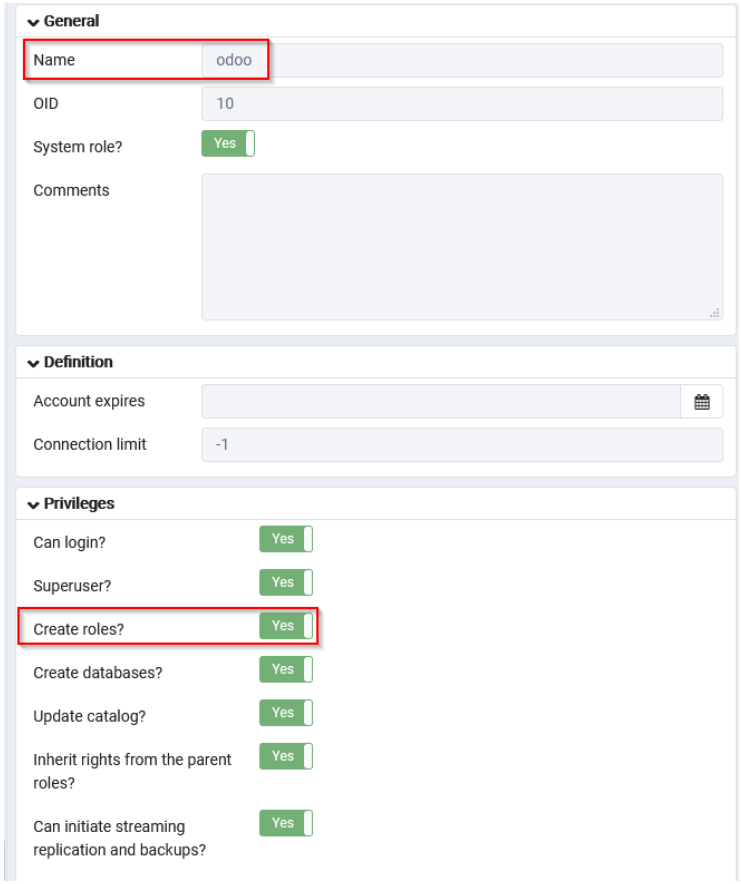
1. Crear els usuaris d'SGBD amb les contrasenyes que corresponguin.
2. Donar els privilegis d'accés adequats als usuaris que corresponguin.

L'accés a la base de dades per part d'usuaris finals ha de ser de **només lectura**.

Crear els usuaris d'SGBD

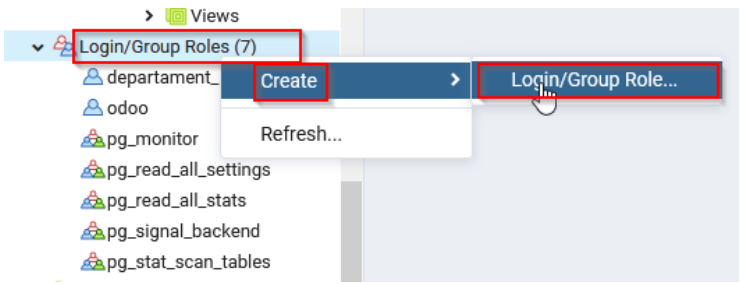
Per tal de crear els usuaris d'SGBD utilitzem l'eina d'administració de PostgreSQL pgAdmin. Hem de connectar-nos a l'SGBD PostgreSQL amb un usuari amb privilegi de creació d'usuaris (rol CREATEROLE) de l'SGBD. Amb aquestes dues coses, podem crear els usuaris d'SGBD amb les contrasenyes que corresponguin (figura 1.8).

Figura 1.8. Usuari amb CREATEROLE



Una vegada connectats amb aquest usuari, procedim a crear l'usuari o els usuaris que tindran **drets de consulta**. En aquest cas farem un usuari per al cap de recursos humans, i un altre per al cap d'administració. La figura 1.9 mostra la pantalla que facilita pgAdmin per crear un nou usuari (botó secundari del ratolí damunt del node *Login Roles* de l'*Object Browser* i seleccionar l'opció *New Login Role*).

Figura 1.9. 'New login role'



Distinció entre usuaris, grups d'usuaris i rols a PostgreSQL

L'eina pgAdmin mostra unificats els termes *Login Role* i *Group Role*. Si donem una ullada a la documentació del llenguatge SQL de PostgreSQL veiem l'existència de les instruccions `create user` (per a la creació d'usuaris) i `create group` (per a la creació de grups d'usuaris).

PostgreSQL utilitzava els conceptes usuaris i grups d'usuaris en versions anteriors a la 8.1, versió en la qual va substituir aquests conceptes pel concepte de *rol* amb dues variants (rol que pot iniciar sessió, *Login Role*, i rol que no pot iniciar sessió *Group Role*) i va introduir la nova instrucció SQL `create role`.

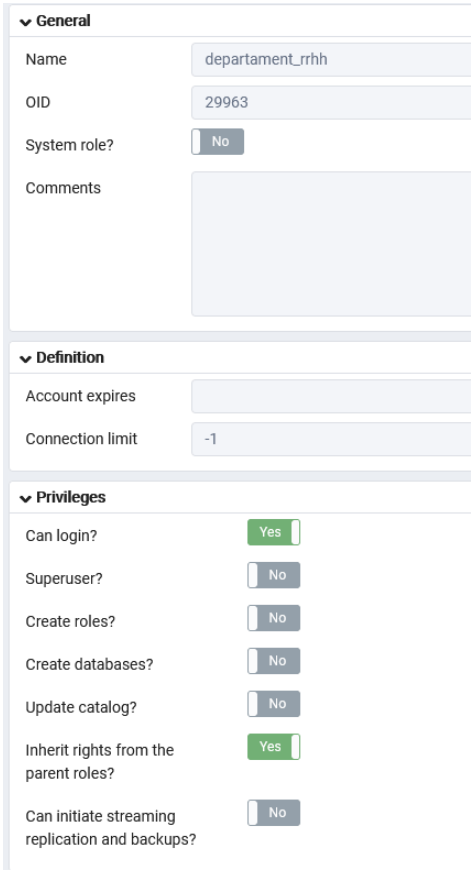
Per mantenir la compatibilitat amb el joc d'instruccions SQL de les versions anteriors a la 8.1 va mantenir l'existència de les instruccions `create user` i `create group`, de manera que `create user` equival a la creació d'un *Login Role* i `create group` equival a la creació d'un *Group Role*.

Donar privilegis d'accés

Suposem que volem crear un usuari de la base de dades cap del departament de Recursos Humans. Ens interessa que aquest usuari pugui connectar-se a la base de dades mitjançant una interfície web, com ara una aplicació BI, però és vital que no pugui fer canvis a la base de dades. Primer generarem aquest usuari, i a continuació triarem les taules que necessita consultar, i li donarem permisos de només lectura.

Crearem, per tant, un ***login role*** per a aquest usuari. Ens col·loquem als rols i fem clic en el botó dret (figura 1.10).

FIGURA 1.10. Creació de l'usuari per a Recursos Humans

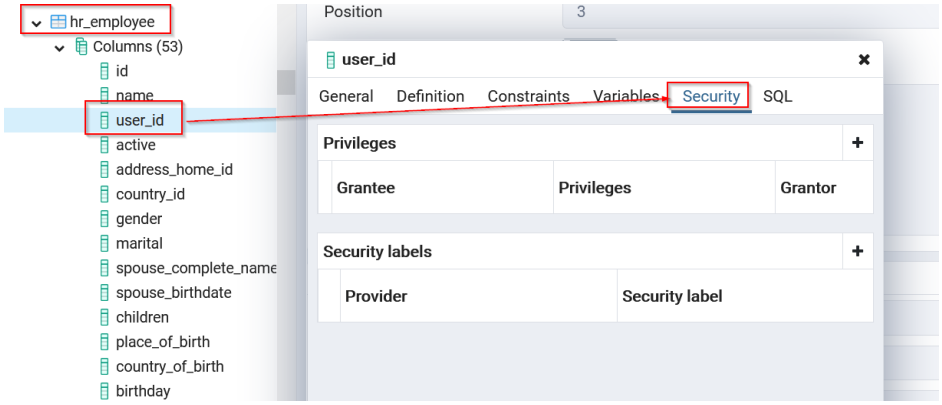


L'assignació de privilegis s'efectua sobre cada objecte i, a través de pgAdmin, per donar privilegis a un objecte determinat cal situar-se damunt del nom de l'objecte, dins l'*Object Browser*, editar-ne les propietats (amb el botó secundari del ratolí) i anar a la pestanya *Privileges*.

Els **privilegis** es poden concedir (figura 1.11):

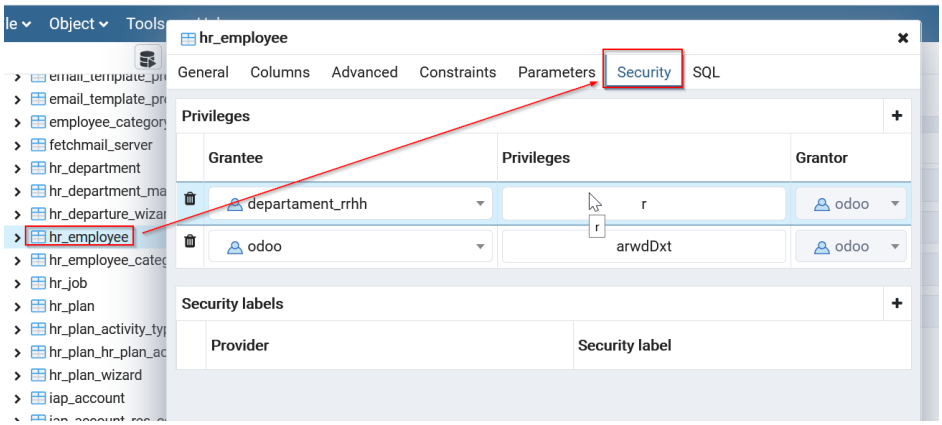
- Pel que fa a la base de dades (per permetre o no l'accés),
- Pel que fa a l'esquema (per utilitzar-lo i/o crear-hi objectes),
- Pel que fa als objectes de la base de dades (taules, vistes, funcions, disparadors...)
- Fins i tot, pel que fa a les columnes de les taules i vistes, per donar privilegis d'accés a columnes, cal situar-se damunt la columna i editar les seves propietats.

FIGURA 1.11. Edició de privilegis d'una columna a PostgreSQL



Per tant, donarem privilegis de només lectura a la taula `hr_employee` a l'usuari creat per a les consultes del departament de Recursos Humans. Seleccionarem la taula, anirem a les seves propietats, i a la pestanya *Seguretat* indicarem que té privilegis de lectura (figura 1.12).

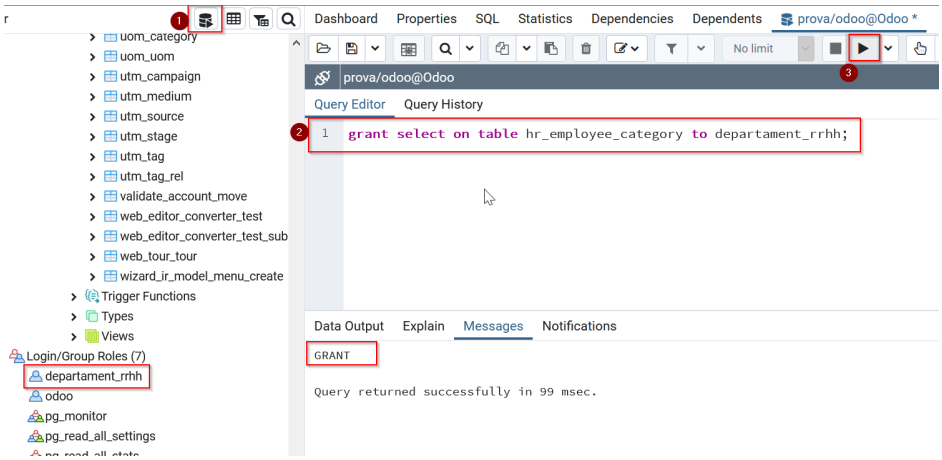
FIGURA 1.12. Privilegis de lectura assignats a l'usuari



Si per motius d'eficiència es prefereix fer servir sentències SQL en comptes de la finestra gràfica, a PgAdmin es pot fer de manera molt senzilla (figura 1.13).

És altament recomanable donar una ullada a la part de la documentació de PostgreSQL referent a l'assignació de privilegis.

FIGURA 1.13. Assignació de privilegis per sentència

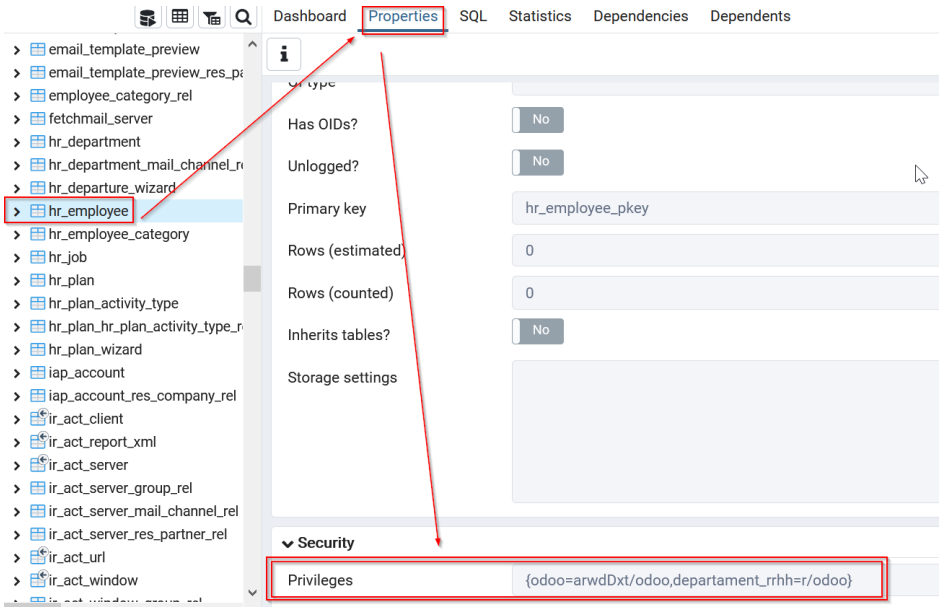


La **documentació de PostgreSQL** es troba a www.postgresql.org/docs i per nosaltres té interès donar una ullada a la part sobre els *Database Roles* i els *Privileges*, així com a les ordres SQL `grant`, `revoke` i `create role`.

Comprovació de privilegis

Una manera ràpida de veure, des de pgAdmin, els privilegis concedits a un objecte, és situar-nos damunt de l'objecte i veure, a la part dreta de la pantalla, el contingut de la **propietat ACL** (*Access Control List*); tal com es mostra a la figura 1.14.

FIGURA 1.14. Comprovació de privilegis per a la taula hr_employee

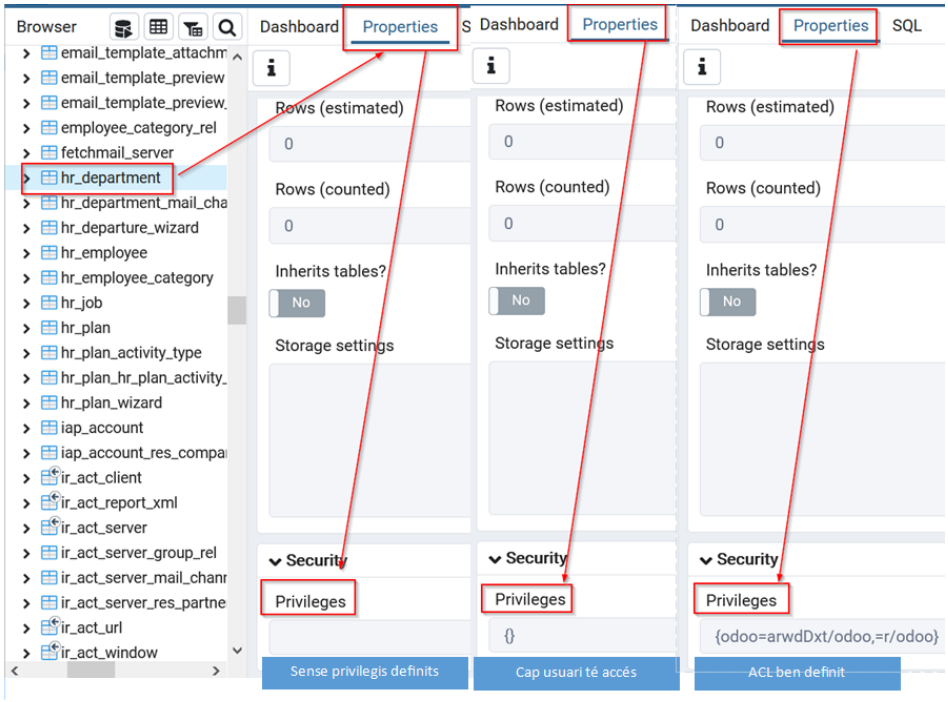


De la mateixa manera que hem vist la llista ACL sobre la taula `hr_employee`, podem situar-nos damunt de qualsevol objecte de la base de dades (esquema, taula, vista, columna...) i tindrem accés a la llista ACL dels privilegis concedits sobre aquell objecte.

Imaginem el cas en què una taula no té definida la llista ACL; això suposarà un **forat de seguretat**, ja que en principi permet l'accés a qualsevol usuari. Per tal

que no hi hagi cap privilegi d'accés, la llista ACL hauria de mostrar el valor {}. Podem aconseguir-ho afegint qualsevol usuari a l'esquema de seguretat, guardant i tornant a editar per esborrar l'usuari. Quan enregistrem el canvi, veurem que el valor d'ACL és {}. La situació ideal és tenir definits els privilegis de només un grup reduït d'usuaris. A la figura 1.15 podem revisar tots tres casos.

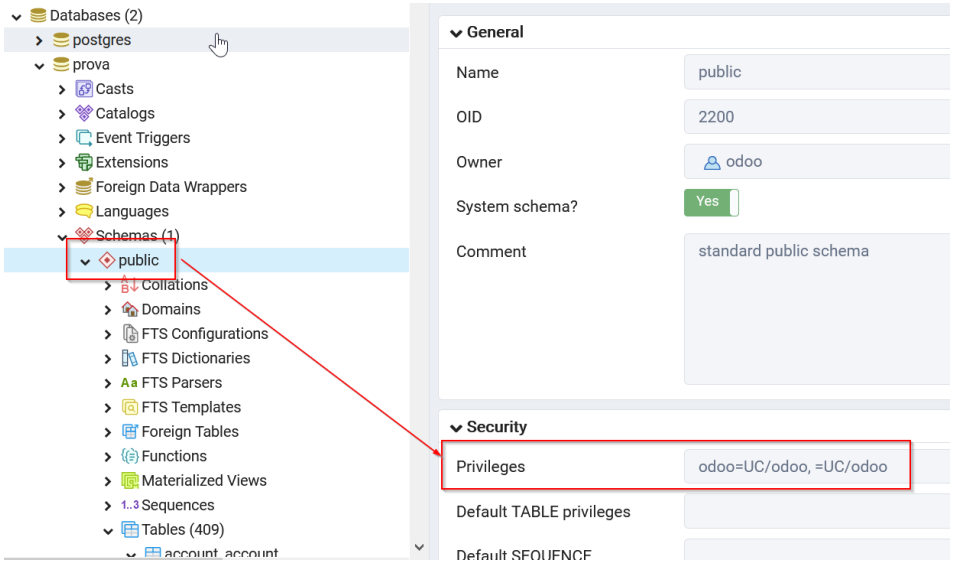
FIGURA 1.15. Diferents esquemes de seguretat per a la taula hr_department



Un punt molt important a tenir en compte en la gestió de privilegis de PostgreSQL és **conèixer els privilegis existents**, de forma automàtica, després de la creació d'una base de dades (vegeu la figura 1.16). Cal saber que:

- La base de dades es crea amb ACL no definida, fet que permet que qualsevol usuari del servidor PostgreSQL pugui obrir sessió en aquella base de dades.
- PostgreSQL facilita el rol `public`, que engloba tots els usuaris de forma automàtica.
- PostgreSQL facilita, a totes les bases de dades, l'esquema `public`, propietat de l'usuari que ha creat la base de dades, i amb privilegis d'utilització de l'esquema (`usage`) i creació d'objectes (`create`) al rol `public` (és a dir, a qualsevol usuari). Així, si observem la propietat ACL de l'esquema `public` d'una base de dades creada per l'usuari `odoo`, veiem el valor `{odoo=UC/odoo, =UC/odoo}` que hem de llegir com: l'usuari `odoo` té privilegis `UC` (`usage+create`) i el rol `public` (no apareix a l'esquerra del símbol =) té privilegis `UC` (`usage+create`) i que en ambdós casos han estat concedits per l'usuari `odoo` (valor que apareix després del símbol `/`).

FIGURA 1.16. ACL de l'esquema public



Un usuari qualsevol, pel fet de pertànyer al rol `public`, té accés `UC` sobre l'esquema `public` de qualsevol base de dades. Això implica que pot veure la relació (noms) de tots els objectes existents a l'esquema (taules, vistes...), veure la descripció de qualsevol objecte (taules, vistes...) i crear nous objectes dins l'esquema, però no pot accedir als continguts de les taules ni vistes, excepte si el propietari d'aquests objectes li concedeix accés.

En cas que hàgim de facilitar accés a la base de dades corresponent a una empresa de PostgreSQL a **nous usuaris** i no ens interessi mantenir aquesta situació, hem de fer el següent:

- Definirem el valor de la propietat ACL de la base de dades i indicarem els usuaris als quals es facilita el privilegi de connexió.
- Modificarem el valor de la propietat ACL de l'esquema `public`, eliminarem l'assignació de privilegis al rol `public` i assignarem la utilització (només `usage`) de l'esquema `public` als usuaris o rols corresponents. Aquesta acció executada mentre el servidor està engegat pot provocar que Odoo no pugui connectar amb la base de dades fins que es reinicïi el servidor.
- Assignarem els privilegis (normalment de lectura) als usuaris o rols corresponents sobre els objectes (taules, vistes, columnes...) que interessi.

1.1.4 Accés a PostgreSQL des d'aplicacions clients (vídeo)

En moltes ocasions serà necessari configurar estacions de treball per tal que algunes aplicacions instal·lades puguin connectar amb l'SGBD PostgreSQL per accedir a les dades emmagatzemades. Aquest és el cas de les eines ofimàtiques actuals, que acostumen a facilitar la connectivitat amb els SGBD a través de connectors ODBC o JDBC que cal tenir instal·lats a la màquina des d'on es vol utilitzar l'eina ofimàtica.

A continuació s'ofereixen dos vídeos amb exemples pràctics de connexió:

- **Connexió des de dBeaver** per conèixer les relacions entre les diferents taules. Creació d'un usuari amb accés de només lectura.

https://player.vimeo.com/video/472575228

- **Connexió des de LibreOffice Base** a la base de dades per a la creació d'una estació de consulta a la base de dades.

https://player.vimeo.com/video/472577762

2. Desenvolupament de mòduls a Odoo

El desenvolupament de mòduls a Odoo és indispensable per poder adequar l'ERP a les necessitats d'implantació en una organització. Per tal d'aprendre com desenvolupar mòduls a Odoo els **passos** seran els següents:

1. Crear un entorn de desenvolupament fent servir l'IDE Pycharm.
2. Generar el diagrama UML del mòdul que volem crear. Conèixer l'eina de diagramació Dia, que ens permetrà crear aquest diagrama de classes.
3. Conèixer l'estructura dels mòduls d'Odoo i començar a desenvolupar.

Coneixements bàsics del llenguatge UML i de Python

Per dissenyar diagrames amb l'eina Dia, és convenient tenir uns coneixements bàsics del llenguatge UML, de nivell similar als adquirits en el mòdul de Programació.

Alhora, per desenvolupar mòduls d'Odoo són necessaris uns coneixements bàsics del llenguatge Python. Fonamentalment, la nostra feina serà d'entendre el codi per a poder replicar-lo; treball adequat per a qualsevol programador, encara que estigui especialitzat en un altre llenguatge.

A la secció "Annexos", del web del mòdul, trobareu documents per a l'aprenentatge del llenguatge Python.

Durant tot aquest tema, treballarem la generació d'un mòdul d'Odoo per al manteniment industrial.

2.1 Generació d'un entorn de desenvolupament amb l'IDE Pycharm (vídeo)

Una de les necessitats de qualsevol implantador és disposar d'un bon entorn de desenvolupament, on poder fer un *debug* ràpid i còmode. Per a aquestes tasques existeixen els **IDE** (*Integrated Development Environment*). En aquests materials es farà servir l'**IDE Pycharm**, pels següents motius:

- És un IDE especialitzat en Python, i per tant ens ofereix un corrector sintàctic, molt apropiat per a quan no dominem Python.
- Té una versió *community*, i per tant gratuïta.
- Hi ha moltes referències a la instal·lació d'Odoo mitjançant Pycharm, molt important per a poder resoldre errors en la instal·lació.

Per a descriure amb detall la confecció de l'entorn de desenvolupament, podeu veure el següent vídeo:

https://player.vimeo.com/video/472580019

2.2 Descripció del mòdul d'exemple: "manteni"

Podeu trobar el codi font del mòdul manteni a la secció "Annexos" del web del mòdul, dintre de "Recursos de programari".

Per a portar a terme el manteniment de les màquines d'una indústria, és molt interessant comptar amb un programa que emmagatzemi tota la informació que es va generant. Partim de la següent situació: cada màquina que s'adquireix per part d'una empresa patirà un desgast al llarg de la seva vida útil. Per aconseguir allargar aquesta vida i evitar aturades es portarà a terme un manteniment. Distingim dos tipus diferents de manteniment:

- **Manteniment preventiu**: consisteix en certes mesures que ajudaran a evitar els problemes amb el maquinari (p. ex. renovar el greix de les màquines amb parts rotatives una vegada a la setmana). El fabricant aconsella quines són les actuacions, i la seva periodicitat, encara que moltes vegades és l'experiència dels operaris la que fa dissenyar el manteniment preventiu.
- **Manteniment correctiu**: consisteix a reparar els problemes que van sorgint. És inevitable, encara que s'ha d'intentar reduir tant com es pugui.

Cada vegada que es porta a terme el manteniment d'una màquina, sigui preventiu o sigui correctiu, s'omplirà una ordre de treball, que contindrà informació sobre quina màquina o màquines hi estan implicades, el treballador que ha portat a terme el manteniment, la data de començament i de finalització...

A més a més, podem definir més conceptes en l'àmbit de manteniment preventiu:

- **Gamma de manteniment**: és una llista de tasques, o instruccions, que s'ha de portar a terme en una màquina, amb una determinada periodicitat, i relatiu a un tema en concret. Una gamma pot ser comuna a moltes màquines, encara que els seus models o propòsits siguin diferents:
- **Instrucció de manteniment**: cadascuna de les tasques de les quals està feta una gamma de manteniment.

Posarem un exemple de gamma de manteniment: gamma de manteniment elèctric per a les màquines d'una empresa.

- **Periodicitat**: bimensual.
- **Instruccions**:
  - Mesura de rigidesa dielèctrica.

  - Verificar corbes de tret.
  - Revisió de línia de terra, connexions i mesurament de resistència.
  - Comprovació funcionament proteccions del transformador.

- **Màquines a les quals s'aplica** (dins la fàbrica):
  - Transformador TR-001
  - Transformador TR-002
  - Grup electrogen GE-001

2.3 Creació d'un mòdul a partir de zero

Una vegada exposat el problema inicial, ens dedicarem a crear un mòdul d'Odoo que pugui resoldre aquestes necessitats.

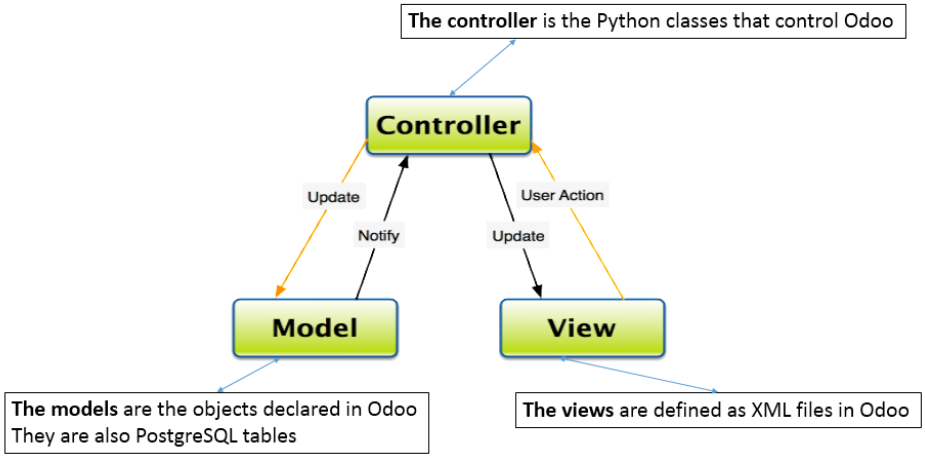
"Build an Odoo module"

Aquesta part de la teoria correspon al capítol "Build an Odoo module" de la documentació oficial d'Odoo: tinyurl.com/yavcdcdt.

Un mòdul inclou tots els elements del patró **model-vista-controlador**, que és la base de desenvolupament d'Odoo (figura 2.1):

- El model: es defineix en llenguatge Python, i són els objectes que declararem, i es traduiran en taules PostgreSQL.
- La vista: la definirem en llenguatge XML.
- El controlador: podrem dissenyar mètodes en llenguatge Python que controlen el funcionament d'Odoo.

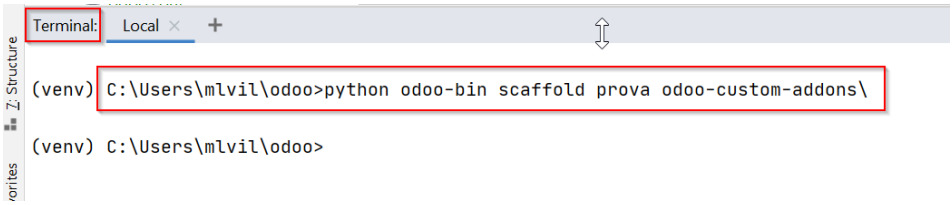
Figura 2.1. Model-vista-controlador



El primer pas serà **crear un mòdul buit**. L'arxiu odoo-bin, que serveix per posar en marxa el servidor, pot rebre el paràmetre `scaffold`, que crea un mòdul buit. Li haurem de dir com es diu el mòdul, i on col·locar-ho (figura 2.2).

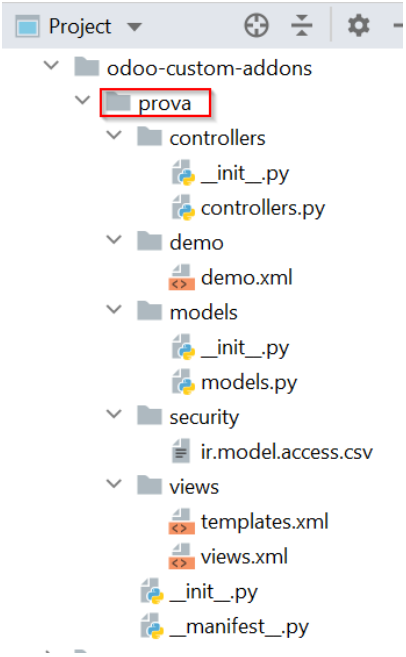
```
odoo-bin scaffold <module name> <where to put it>
```

FIGURA 2.2. Creació d'un mòdul buit a Odoo mitjançant odoo-bin scaffold



Aquesta instrucció ens crearà tota l'estructura d'arxius i carpetes d'un mòdul buit (figura 2.3).

FIGURA 2.3. Arxius i carpetes d'un mòdul nou



2.3.1 Arxius de presentació d'un mòdul

A partir d'aquest punt, totes les captures que es mostraran corresponen al mòdul d'exemple "manteni", visualitzats amb l'IDE Pycharm.

Començarem pels dos arxius que apareixen a la carpeta arrel del mòdul, __init__.py i __manifest__.py.

L'arxiu **__init__.py** (dos guions baixos abans i després de *init*) identifica el mòdul com un programa Python. S'hi col·loquen tots els fitxers Python que cal carregar per a l'execució del programa. Per defecte es carreguen els controladors i models. Vegeu-ho:

```
1 # -*- coding: utf-8 -*-
2
3 from . import controllers
4 from . import models
```

Per la seva banda, el fitxer **__manifest__.py** ha d'estar ubicat a l'arrel de la carpeta del mòdul i ha de tenir el format d'un diccionari de Python (figura **??**). És el responsable de determinar:

- Nom, descripció, autors, llicència, versió, categoria... del mòdul.
- Els arxius XML que s'analitzaran durant l'arrencada del servidor Odoo.
- Les dependències amb altres mòduls.

Els valors del diccionari Python que conté aquest fitxer són els següents:

- *name*: nom del mòdul en anglès.
- *summary*: descripció del propòsit del mòdul en una frase.
- *description*: text que descriu el mòdul.
- *author*: autor del mòdul.
- *website*: web de l'autor del mòdul.
- *category*: categoria a la qual pertany el mòdul. Text separat per barres / amb la ruta jeràrquica de les categories.
- *version*: versió del mòdul.
- *depends*: llista Python dels mòduls dels quals depèn aquest mòdul.
- *data*: llista Python dels noms de fitxers XML en instal·lar/actualitzar el mòdul. La ruta dels arxius ha de ser relativa a la carpeta on es troba el mòdul.
- *demo*: llista de Python de noms de fitxers XML que es carreguen si s'ha instal·lat Odoo amb les dades de demostració o exemple. Les rutes dels arxius han de ser relatives al directori on hi ha el mòdul.

Els diccionaris de Python es defineixen entre {} i les paraules clau separades per comes. Les llistes es defineixen entre [] i els seus valors també van separats per comes.

Exemple de l'arxiu __manifest__.py del mòdul manteni

```
# -*- coding: utf-8 -*-
{
    'name': "manteni",

    'summary': """
        Example module made to teach Odoo""",

    'description': """
        Example module made to teach Odoo
        Industrial maintenance
    """,

    'author': "Mikel López Villarroya",
    'website': "http://www.github.com/mlvillarroya",

    # Categories can be used to filter modules in modules listing
    # Check https://github.com/odoo/odoo/blob/master/odoo/addons/
        base/module/module_data.xml
    # for the full list
    'category': 'Industrial',
```

Diccionari de Python

Podeu trobar la definició de *dictionary* de Python en aquest enllaç: ja.cat/FaYXV.

Llista de categories

Podeu trobar la llista de categories possibles per als mòduls Odoo en aquest enllaç: ja.cat/hA8gD.

```
    'version': '0.1',

    # any module necessary for this one to work correctly
    'depends': ['base', 'hr'],

    # always loaded
    'data': [
        'security/security.xml',
        'security/ir.model.access.csv',
        'views/partner.xml',
        'views/views.xml',
        'views/templates.xml',
        'report/manteni_report.xml',
        'data/data.xml',
    ],
    # only loaded in demonstration mode
    'demo': [
        'demo/demo.xml',
    ],
    'installable': True,
    'application': True,
}
```

2.4 El model d'Odoo

Una vegada s'ha presentat el desenvolupament de mòduls a Odoo, es portarà a terme el primer pas important: la **creació del model**. Aquesta tasca és crítica, ja que proporciona els fonaments sobre els quals s'aixecarà tot el mòdul. És per això que el primer pas serà fer el disseny teòric del mòdul, mitjançant un **diagrama UML**; es farà servir l'eina **DIA**. A continuació, es determinaran els arxius bàsics a modificar per tal de crear aquest model.

2.4.1 Disseny de classes amb Dia

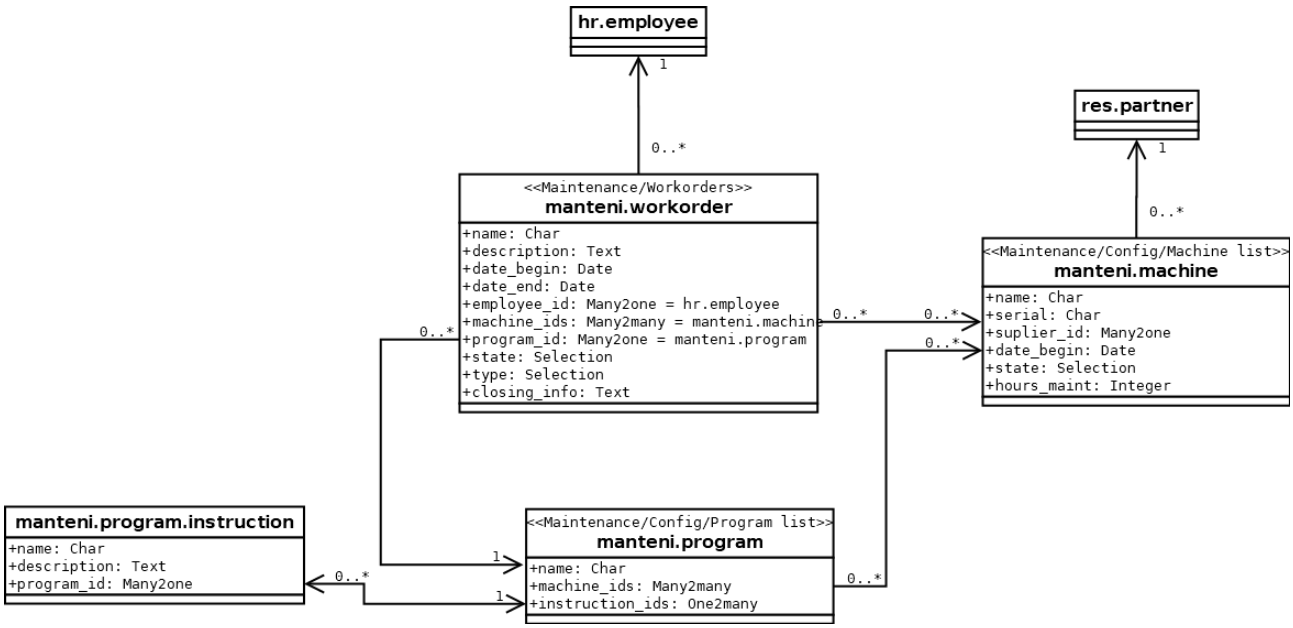
Dia és una aplicació gràfica de propòsit general per a la creació de diagrames, desenvolupada com a part del projecte GNOME. Està construïda de forma modular, amb diferents paquets de formes per a diferents necessitats. A nosaltres ens interessa aquesta eina per **dissenyar els diagrames UML** dels mòduls Odoo que volem desenvolupar.

Creació de diagrames

Hi ha moltes més opcions per a la creació de diagrames UML, tant instal·lables com *online*, totes perfectament vàlides. Una alternativa instal·lable i *open source* seria www.modelio.org, i una alternativa *online* www.diagrams.net o www.lucidchart.com.

Ja que hem començat a definir les necessitats per al mòdul de manteniment, crearem a continuació, ara amb Dia, el seu diagrama UML. Posem en marxa l'eina Dia i obrim el fitxer proporcionat (figura 2.4).

FIGURA 2.4. Diagrama UML per al mòdul manteni



L'eina Dia permet desenvolupar diversos tipus de diagrama i, com que ens interessen els diagrames UML, caldrà tenir seleccionada l'opció UML a la caixa d'eines de la part esquerra de la pantalla de Dia.

Eines per crear diagrames UML

En una situació com la presentada, a l'hora de desenvolupar un diagrama UML, és molt lògic pensar en un disseny que contingui classes per modelar els conceptes ordre de treball, màquina, gamma i instrucció.

- Classe `manteni.workorder`, per modelar les **ordres de treball**.
- Classe `manteni.machine`, per modelar les **màquines**.
- Classe `manteni.program`, per modelar les **gammes de manteniment**.
- Classe `manteni.program.instruction`, per modelar les **instruccions**.

Una primera ullada al diagrama permet introduir una sèrie de **conceptes** per a mòduls Odoo, amb les següents indicacions:

- Tots els elements d'un diagrama cal escriure'ls en anglès (fins i tot les etiquetes i els textos informatius). Si ens interessa tenir-los en català o castellà, utilitzarem posteriorment les eines de traducció que facilita Odoo.
- Totes les classes s'anomenen en minúscules i es recomana incorporar, com a prefix, el nom del mòdul al qual pertanyen.
- Els noms de mòduls, classes i membres de les classes han de ser escrits en minúscula amb guions baixos per fer llegibles els noms compostos.
- La nomenclatura indicada és la que marca Odoo i, com ja deveu saber, és diferent de les nomenclatures recomanades en altres entorns de POO, com és el cas del llenguatge Java.

Com que els noms de les classes de l'exemple tenen el prefix *manteni*, deduïm que es tracta del diagrama d'un mòdul anomenat *manteni*.

Un altre detall és la **nomenclatura de la classe** per a les instruccions, `manteni.program.instruction`. A l'hora de crear les classes estem donant informació sobre la jerarquia subjacent, ja que les instruccions són part de les gammes.

En el diagrama també observem que entre les classes hi ha uns **connectors** (associacions entre classes) de caràcter informatiu i ens serveixen únicament com a documentació gràfica:

- Una gamma pot contenir diverses instruccions.
- Cada ordre de treball conté només una gamma. Una gamma pot associar-se a moltes ordres de treball.
- Cada ordre de treball pot aplicar-se a diverses màquines. Una màquina pot veure's afectada per diverses ordres de treball.
- Cada ordre de treball serà efectuada per un treballador. Un treballador pot tenir associades diverses ordres de treball.
- Cada màquina ha estat adquirida a un proveïdor (*partner*). Cada proveïdor ens pot vendre diverses màquines.

Exemple de disseny d'un diagrama UML

Per a descriure amb detall la confecció d'un diagrama UML mitjançant un exemple diferent podeu veure el vídeo següent:

https://player.vimeo.com/video/472581624

2.4.2 models.py

Una vegada hem dissenyat el diagrama, començarem amb el desenvolupament del codi del model d'Odoo. Obrirem l'arxiu models.py. El model d'Odoo contindrà tots els objectes (classes) que s'han definit anteriorment amb DIA. Tota la informació s'introduirà a la carpeta *models*. Aquesta carpeta contindrà com a mínim dos arxius:

- **__init__.py**, on s'esmenten tots els fitxers que conté la carpeta, i seran importats per crear el mòdul.
- **models.py**, on es generaran totes les classes que componen el mòdul.

L'arxiu models.py conté, com hem dit, la definició del model del mòdul d'Odoo. Per definir un objecte (classe d'Odoo), cal definir una classe de Python i posteriorment instanciar-la.

Com a primer pas en la creació del model és molt interessant revisar l'arxiu models.py generat amb un mòdul buit.

Exemple de l'arxiu models.py per a un mòdul buit

```
# -*- coding: utf-8 -*-

# from odoo import models, fields, api


# class nom_modul(models.Model):
#     _name = 'nom_modul.nom_modul'
#     _description = 'nom_modul.nom_modul'

#     name = fields.Char()
#     value = fields.Integer()
#     value2 = fields.Float(compute="_value_pc", store=True)
#     description = fields.Text()
#
#     @api.depends('value')
#     def _value_pc(self):
#         for record in self:
#             record.value2 = float(record.value) / 100
```

Com podem veure, tenim diverses parts:

- **Inicialització**, especificant que la codificació és utf-8.
- **Importació dels mòduls Python que necessitarem**. En cas que se n'hagin d'importar més, s'haurà de fer en aquesta secció (per exemple, el mòdul "datetime").
- **Definició de les classes**. A continuació s'aniran definint una a una les classes que havíem proposat al diagrama UML.

El següent pas serà dissenyar les classes. Dividirem la creació de les classes en la definició d'atributs de classe i definició dels camps de la classe.

2.4.3 Atributs de classe

"ORM API / Models"

Aquesta part de la teoria correspon al capítol "ORM API / Models" de la documentació oficial d'Odoo: tinyurl.com/y7nkaj36.

Els atributs de classe ajuden a editar certes característiques de la classe. Només n'hi ha un d'**obligatori**, l'atribut `_name`. La resta són **optatius**. A part de l'atribut `_name` obligatori, podem trobar, entre d'altres, els següents:

- `_rec_name`: totes les classes per defecte tindran un camp anomenat `name` que emmagatzemarà el nom de cada registre, que s'utilitzarà per als camps relacionals. En cas de voler emmagatzemar com a nom un altre camp diferent, s'indicarà a `_rec_name`.

- `_inherit`: per treballar el mecanisme d'herència. Es tractarà més endavant en aquest manual.
- `_order`: permet declarar el camp que s'utilitzarà per ordenar els registres quan es faci una recerca sense especificar un ordre concret.
- `_auto`: defineix si s'ha de crear una taula de la base de dades amb la informació de la classe. És True per defecte.
- `_table`: en cas de no voler que es generi el nom de taula per defecte (nom de la classe substituint els punts per guions baixos), nom que tindrà.
- `_sql_constraints`: condicions SQL que volem que es revisin com a mesura de seguretat. Per exemple, ens permet verificar que en el registre afegit no estem introduint un valor ja existent.

2.4.4 Camps possibles a Odoo

Una vegada indicats els atributs de la classe, es procedeix a definir els diferents camps. Aquests camps s'hauran d'adaptar a la definició UML realitzada prèviament, i poden ser de diversos tipus; diferenciem entre:

- **Camps bàsics**
- **Camps avançats**:

Els **camps bàsics** són camps simples que contenen informació que s'emmagatzema a la base de dades. Poden ser de diversos tipus (numèric, caràcter, text, data ...). Alguns dels **paràmetres comuns** (tots són opcionals) a qualsevol camp bàsic són:

- `string`: text que veuran els usuaris relacionat amb el camp. Si no s'inclou, els usuaris veuran el nom del camp. Si es col·loca com a primer paràmetre només cal ficar-hi el nom, sense l'etiqueta.
- `help`: `tooltip` amb ajuda que veuran els usuaris.
- `readonly`: camp de només lectura (False per defecte).
- `required`: camp obligatori (False per defecte).
- `index`: generar un índex a la base de dades (False per defecte).
- `default`: valor per defecte que pren el camp. Es pot utilitzar un valor concret o trucar a una funció (que es descriurà després en el mateix arxiu), per al càlcul.
- `groups`: llista separada per comes dels grups amb accés a aquest camp.

Els **tipus de camps** que es poden utilitzar a Odoo són:

- `Char`: camp bàsic de text. Té dos paràmetres:
  - `size`: mida màxima del camp
  - `translate`: activa la traducció del camp si ho posem com a True.
- `Boolean`
- `Integer`
- `Float`: número en format coma flotant. Té un paràmetre:
  - `digits`: un parell de números (total, decimal), que representen la precisió del número flotant.

És obligatori i important que en totes les classes es defineixi un camp anomenat `name` de tipus `Char`, ja que s'utilitzarà per fer referència a la classe quan es creuen **camps relacionals**.

A la figura 2.5 podeu veure la classe `manteni.machine` del mòdul "manteni", que conté diversos camps bàsics.

Figura 2.5. Classe manteni.machine

```
class Machine(models.Model):
    # Machines
    _name = 'manteni.machine'

    name = fields.Char(size=32, string='Machine name', index=True)
    serial = fields.Char(size=32, string='Serial no', index=True)
    suplier_id = fields.Many2one('res.partner', string='Suplier', domain=[('parent_id', '=', False), ('machinery_seller', '=', True)])
    date_begin = fields.Date(string='Date begin', default=fields.Date.today)
    date_first_maintenance = fields.Date(compute='_maintenance_date')
    state = fields.Selection([('on_use', 'On use'), ('repairing', 'Repairing'), ('out_order', 'Out of order')],
                             string='State', default='on_use')
    city = fields.Char(related='suplier_id.city', store=False)
    hours_maint = fields.Integer(string='Hours till maintenances')
    active = fields.Boolean('Active', default=True)
```

Pel que fa als **camps avançats**, els més senzills serien els següents (figura 2.6):

- `Text`: és similar a `Char`, però utilitzat per a cadenes més llargues. No s'indica la mida, i es mostra en pantalla mitjançant una caixa multilínia. Té un paràmetre:
  - `translate`: activa la traducció del camp si ho posem com a True.
- `Selection`: crea un *combo* desplegable per triar un valor. Té un paràmetre obligatori:
  - `selection`: valors possibles del camp. Es presenten com una llista de parells (valor, text), separats per comes.

Figura 2.6. Camps text i selection a la classe manteni.workorder

```
class Workorder(models.Model):
    # Workorder to mantain/repair one of more machines
    _name = 'manteni.workorder'

    name = fields.Char('Title', size=64, required=True, translate=True)
    description = fields.Text('Description', translate=True)
    date_begin = fields.Date(string='Opening date', default=fields.Date.today)
    date_end = fields.Date(string='Closing date')
    employee_id = fields.Many2one('hr.employee', string='Employee', domain="[('department_id','=','Maintenance')]")
    employee_user_id = fields.Integer()
    machine_ids = fields.Many2many('manteni.machine', string='Machines')
    program_id = fields.Many2one('manteni.program', string='Program')
    state = fields.Selection([('not_started', 'Not started'), ('opened', 'Opened'), ('closed', 'Closed'), ('cancelled', 'Cancelled')],
        string='State', default='opened')
    type = fields.Selection([('corrective', 'Corrective'), ('preventive', 'Preventive')], string='Type of maintenance', default='corrective')
    closing_info = fields.Text('Closing information')
```

Hi ha **altres camps avançats**, com ara:

- Camps de data i hora (`Date` i `Datetime`)
- Camps relacionals (`relational`)
- Camps pseudorelacionals (`pseudo-relational`)
- Camps calculats (`computed`)
- Camps relacionats (`related`)

Camps de data i hora (date i datetime)

Aquest tipus de camp són molt importants a Odoo, ja que serveixen per **emmagatzemar dates i hores**. A més a més de poder definir i emmagatzemar dates i hores, podem utilitzar mètodes per operar (calcular edats, sumar o restar dates...).

Per **introduir un valor** en un cap del tipus `date` o `datetime`, ho podem fer mitjançant:

- Un `string` en el format correcte (YYYY-MM-DD o YYYY-MM-DD HH:MM:SS)
- Un objecte Python tipus `date` o `datetime`
- `False` o `none`

Si no estem segurs d'estar assignant un valor correcte a una data, podem fer servir el mètode `to_date()` o `to_datetime()`.

Un altre mètode molt útil és `today()`, ja que ens permet introduir la data actual, per exemple com a valor per defecte d'un camp de data (figura 2.7).

Figura 2.7. Camp de data a la classe manteni.machine

```
class Machine(models.Model):
    # Machines
    _name = 'manteni.machine'

    name = fields.Char(size=32, string='Machine name', index=True)
    serial = fields.Char(size=32, string='Serial no', index=True)
    suplier_id = fields.Many2one('res.partner', string='Suplier', domain=[('parent_id', '=', False), ('machinery_seller', '=', True)])
    date_begin = fields.Date(string='Date begin', default=fields.Date.today)
    date_first_maintenance = fields.Date(compute='_maintenance_date')
    state = fields.Selection([('on_use', 'On use'), ('repairing', 'Repairing'), ('out_order', 'Out of order')],
                             string='State', default='on_use')
    city = fields.Char(related='suplier_id.city', store=False)
    hours_maint = fields.Integer(string='Hours till maintenances')
    active = fields.Boolean('Active', default=True)
```

Camps relacionals (relational)

Entre les classes d'Odoo es poden establir **tres tipus de relacions**, similars a les del model entitat-relació per a les bases de dades. Aquestes relacions es defineixen mitjançant els atributs referencials següents:

- `Many2one`
- `One2many`

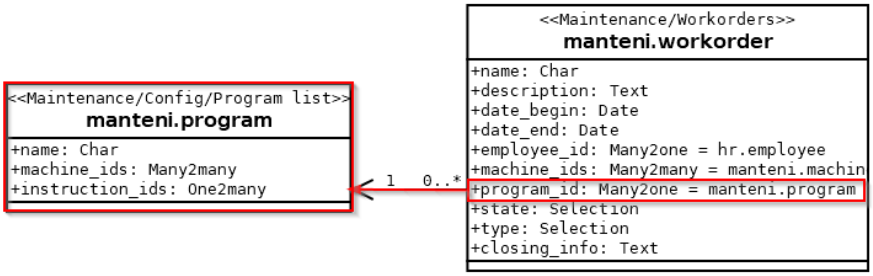
- `Many2many`

`Many2one`:

Representa una relació cap a una classe pare. Molts objectes de la classe que conté l'atribut poden estar relacionats amb el mateix objecte de la classe pare.

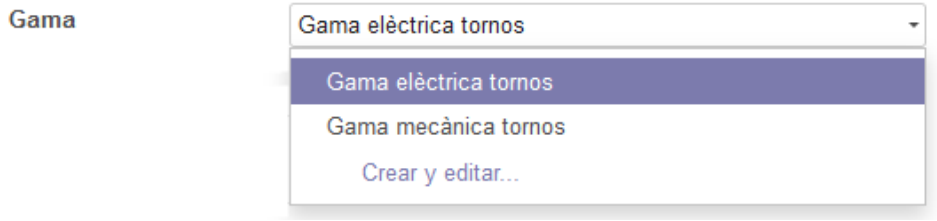
Com a exemple, la relació entre la classe `workorder` i la classe `program` és `Many2one`, ja que cada ordre de treball només té una gamma de manteniment associada (figura 2.8).

Figura 2.8. Relació "Many2one" entre la classe "workorder" i la classe "program"



Odoo mostra els camps `Many2one` acompanyats per una llista per seleccionar l'objecte de la classe pare. Això obliga que la classe referenciada pel camp `Many2one` contingui el camp `name` (figura 2.9).

Figura 2.9. Widget desplegable associat a un camp Many2one



Per definir un atribut `Many2one` a Odoo podem utilitzar (entre d'altres) els següents paràmetres (figura 2.10):

- `comodel_name` (obligatori): nom de la classe destí
- `domain` (opcional): permet filtrar per no veure tots els valors.
- `Ondelete` (opcional): què fer quan s'esborra el registre referit. Els valors possibles són `set_null`, `restrict` i `cascade`.

FIGURA 2.10. Definició del camp Many2one program_id

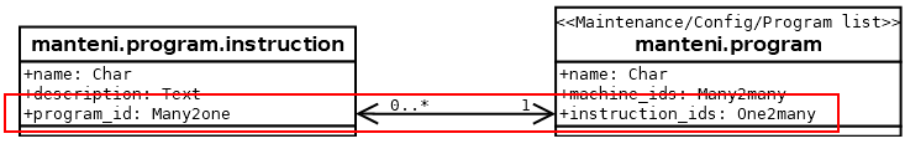
```
class Workorder(models.Model):
    # Workorder to mantain/repair one of more machines
    _name = 'manteni.workorder'

    name = fields.Char('Title', size=64, required=True, translate=True)
    description = fields.Text('Description', translate=True)
    date_begin = fields.Date(string='Opening date', default=fields.Date.today)
    date_end = fields.Date(string='Closing date')
    employee_id = fields.Many2one('hr.employee', string='Employee', domain="[('department_id','=','Maintenance')]")
    employee_user_id = fields.Integer()
    machine_ids = fields.Many2many('manteni.machine', string='Machines')
    program_id = fields.Many2one('manteni.program', string='Program')
    state = fields.Selection([('not_started', 'Not started'), ('opened', 'Opened'), ('closed', 'Closed'), ('cancelled', 'Cancelled')],
        string='State', default='opened')
    type = fields.Selection([('corrective', 'Corrective'), ('preventive', 'Preventive')], string='Type of maintenance', default='corrective')
    closing_info = fields.Text('Closing information')
```

`One2many`:

Representa una relació cap a una classe filla. Un objecte de la classe que conté l'atribut pot estar relacionat amb molts objectes de la classe filla. Cada atribut `One2many` ha de ser complementari d'un atribut `Many2one` que ha d'estar obligatòriament en la classe filla.

Tornem a la definició del mòdul. Cada instrucció pertany a una gamma de manteniment, i una gamma pot tenir més d'una instrucció. Per tant, tenint en compte les definicions, hi haurà un camp `One2many` a la classe de les gammes (`manteni.program`), i un camp `Many2one` corresponent i necessari a la classe de les instruccions (`manteni.instruction`); com podeu veure a la figura 2.11.

FIGURA 2.11. Relacions One2many i Many2one entre les classes program i instruction



Odoo mostra per als camps `One2many` una llista amb tots els objectes relacionats, i la possibilitat d'afegir-ne més (figura 2.12).

FIGURA 2.12. Widget de llista associat a un camp One2many

Per definir un atribut `One2many` a Odoo podem utilitzar (entre d'altres) els següents paràmetres (figura 2.13):

- `comodel_name` (obligatori): nom de la classe destí.
- `inverse_name` (obligatori): nom del camp invers `Many2one` que es troba a la classe destí.
- `domain` (opcional): permet filtrar per no veure tots els valors.

Figura 2.13. Definició del camp Many2one instruction_ids

```
class Program(models.Model):
    # Maintenance programs, conjunt of rules.
    _name = 'manteni.program'

    name = fields.Char(string='Name', size=64, required=True)
    instruction_ids = fields.One2many('manteni.program.instruction', 'program_id', string='Instructions')
```

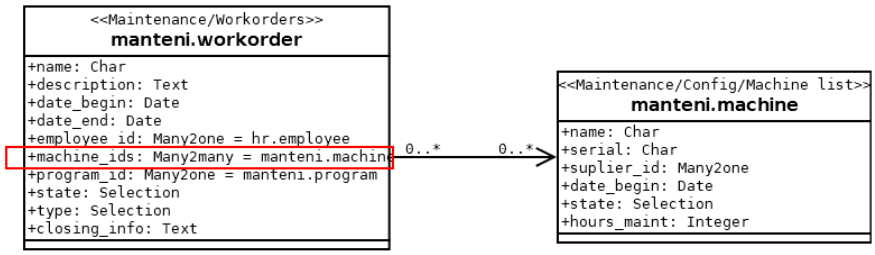
> L'existència d'un **atribut** `One2many` implica que hi ha d'haver un **atribut** `May2one` corresponent. No obstant això, l'existència d'un atribut `Many2one` no implica que hi hagi d'haver un atribut `One2may`.

`Many2many`:

Representa una relació de molts a molts entre dos objectes. Cada objecte d'una classe A pot estar relacionat amb molts objectes de la classe B i cada objecte de la classe B pot estar relacionat amb molts objectes de la classe A.

En el nostre cas, en una ordre de treball es pot demanar el manteniment de diverses màquines, i una màquina apareixerà en diverses ordres de treball, a mesura que va patint avaries (figura 2.14).

Figura 2.14. Relacions Many2many workorder i machine



Odoo mostra per als camps `Many2many` una llista amb tots els objectes relacionats, i la possibilitat d'afegir-ne més, igual que a `One2many` (figura 2.15).

Figura 2.15. Widget de llista associat a un camp Many2many

Per definir un atribut `Many2many` a Odoo podem utilitzar (entre d'altres) els següents paràmetres (figura 2.16):

- `comodel_name` (obligatori): nom de la classe destí.
- `relation` (opcional): nom de la taula de la base de dades que contindrà la relació entre les dues classes.

- `column1` (opcional): nom de la columna referida per al registre de la classe actual.
- `column2` (opcional): nom de la columna referida per al registre de la classe destí.
- `domain` (opcional): permet filtrar per no veure tots els valors.

FIGURA 2.16. Definició del camp Many2many machine_ids

```
class Workorder(models.Model):
    # Workorder to mantain/repair one of more machines
    _name = 'manteni.workorder'

    name = fields.Char('Title', size=64, required=True, translate=True)
    description = fields.Text('Description', translate=True)
    date_begin = fields.Date(string='Opening date', default=fields.Date.today)
    date_end = fields.Date(string='Closing date')
    employee_id = fields.Many2one('hr.employee', string='Employee', domain="[('department_id','=','Maintenance')]")
    employee_user_id = fields.Integer()
    machine_ids = fields.Many2many('manteni.machine', string='Machines')
    program_id = fields.Many2one('manteni.program', string='Program')
    state = fields.Selection([('not_started', 'Not started'), ('opened', 'Opened'), ('closed', 'Closed'), ('cancelled', 'Cancelled')],
        string='State', default='opened')
    type = fields.Selection([('corrective', 'Corrective'), ('preventive', 'Preventive')], string='Type of maintenance', default='corrective')
    closing_info = fields.Text('Closing information')
```

Els camps opcionals `relation`, `column1` i `column2` eren necessaris en versions anteriors, i per mantenir la compatibilitat amb mòduls antics no s'han retirat. Ara bé, no són necessaris.

Camps pseudorelacionals (pseudo-relational)

Aquest és un tipus especial (i molt poc utilitzat) de camps relacionals, on no es genera cap clau forana a la base de dades, només un `integer` que relaciona el camp amb el seu origen.

Hi ha dos tipus diferents de camps d'aquest tipus:

- `Reference`
- `Many2onereference`

L'ús d'aquest tipus de camp és tan reduït que realitzant una recerca al codi font d'Odoo només es troba tres vegades. És per això que no s'aprofundirà en el seu estudi.

Camps calculats (computed)

Potser ens interessa crear un camp que, en lloc d'utilitzar un valor emmagatzemat a la base de dades, es generi **mitjançant un càlcul**. Per exemple, podríem calcular de forma dinàmica l'edat d'un treballador de l'empresa mitjançant la seva data de naixement i el dia actual. D'aquesta manera, sense necessitat d'ocupar espai a la base de dades tindríem una dada actualitzada i utilitzable en cada moment.

Fan referència a un mètode que s'haurà de definir a continuació en llenguatge Python.

Com a exemple podem veure el càlcul de la data del primer manteniment, que es troba a la classe `machine` (figura 2.17).

FIGURA 2.17. Camp calculat date_first_maintenance

De la figura anterior, cal destacar que:

- En primer lloc, es troba un bucle *for record in self*, ja que l'objecte a calcular pot ser un conjunt de registres, no només un (per exemple, si volem mostrar les dates de totes les màquines a una taula).
- Es recull el camp `date_begin`, que és un camp de tipus `date`.
- Es fa servir la funció `timedelta` de la llibreria `datetime` de Python per tal de crear un objecte de tipus `date` amb les hores de manteniment, i poder-lo afegir a la data inicial.

Funció "timedelta" de Python

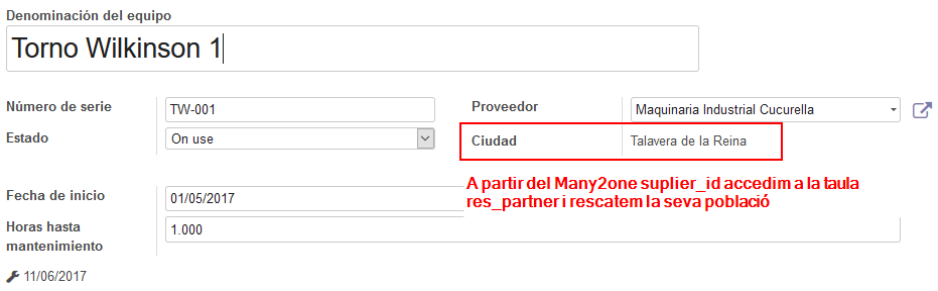
Podeu trobar informació sobre la funció `timedelta` de Python a: tinyurl.com/ya4fr95z.

Camps relacionats (related)

L'últim tipus de camp avançat és el relacionat. Els camps relacionats serveixen per **obtenir un valor d'un altre model**, seguint les relacions entre atributs. Això evita haver de duplicar informació a la base de dades.

Per exemple, a la classe `manteni.machine` hem creat el camp `suplier_id`, que fa referència a la taula `res_partner`. Si a més del nom de l'empresa voldríem mostrar la seva direcció, podríem utilitzar un camp `related`, que utilitzés `suplier_id` per consultar a la taula `res_partner` sense necessitat de duplicar la informació (figura 2.18).

FIGURA 2.18. Camp relacionat al mòdul manteni



Exemple de creació del model d'un mòdul

Per a descriure amb detall la confecció del model d'un mòdul mitjançant un exemple diferent, podeu veure el següent vídeo:

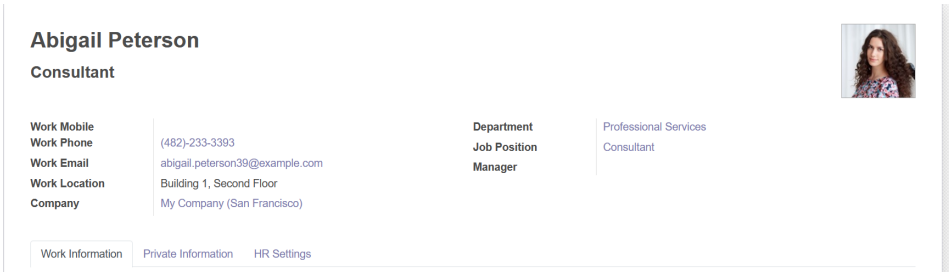
https://player.vimeo.com/video/472582195

2.5 La vista a Odoo

"Views"
Aquesta part de la teoria correspon al capítol "Views" de la documentació oficial d'Odoo: tinyurl.com/y76n6hsy.

Odoo és un programari de gestió empresarial desenvolupat sobre el *framework* OpenObject de tipus RAD (*Rapid Application Development*) que facilita una arquitectura MVC (model-vista-controlador) per als desenvolupaments. Una vegada es domina el disseny del model de dades d'una aplicació desenvolupada sobre OpenObject, com és el cas d'Odoo, cal entrar en el coneixement del disseny de la vista i del controlador. En les aplicacions desenvolupades sobre el *framework* OpenObject, el concepte vista engloba les **pantalles** que permeten exposar la informació a l'usuari (anomenades *vistes* o *views*) per a visualitzar-la i/o editar-la, i els menús, que faciliten un accés organitzat a les vistes. En conseqüència, ens cal aprendre a dissenyar les vistes (*views*) i els menús (figura 2.19).

FIGURA 2.19. Vista de la fitxa d'un treballador (mòdul de Recursos Humans)



Per tant, direm que les **vistes d'Odoo** són les pantalles que faciliten l'accés de l'usuari a la informació, tant per consultar-la com per modificar-la (altes, baixes i modificacions). Les vistes són dinàmiques i es construeixen en temps d'execució a partir de descripcions XML accessibles des del client, fet que en possibilita, fins i tot, la modificació en temps d'execució.

2.5.1 Declaració de la vista

La descripció de les vistes ha de residir en un **fitxer XML** que ha d'estar referenciat des de l'apartat data del fitxer descriptor del mòdul __manifest__.py. Així, en el mòdul d'exemple "manteni" hi observem la presència del fitxer anomenat views.xml i en el fitxer __manifest__.py hi trobem la línia (figura 2.20):

FIGURA 2.20. Línia de càrrega de l'arxiu views.xml a __manifest__.py

```
'data': [
    'security/security.xml',
    'security/ir.model.access.csv',
    'views/partner.xml',
    'views/views.xml',
    'views/templates.xml',
    'report/manteni_report.xml',
    'data/data.xml',
],
```

El nom del fitxer XML en el qual resideixen les vistes pot ser qualsevol, però és altament aconsellable utilitzar el nom genèric "views.xml" o una combinació en la qual aparegui el nom del mòdul i alguna cosa que n'indiqui el contingut (per exemple, "manteni.xml" o "manteni_views.xml").

2.5.2 Sintaxi genèrica i tipus de vistes

Més enllà dels diferents tipus de vistes que puguem trobar, hi ha una sintaxi genèrica **comuna a qualsevol vista**.

Els camps que admet aquesta **declaració genèrica** (entre d'altres), són els següents:

- `name` (obligatori): nom de la vista.
- `model`: classe a la qual fa referència aquesta vista.
- `priority`: prioritat per a carregar la vista. Si Odoo ha de carregar diverses vistes, triarà la que tingui un número de prioritat més baix.
- `arch`: descripció específica de la vista. **Aquesta és la part en què cada tipus de vista específic tindrà la seva sintaxi**.
- `groups_id`quins grups d'Odoo podran veure aquesta vista.
- `inherit_id`: veurem el concepte d'herència més endavant.

La sintaxi mínima de la declaració d'una vista serà similar a:

```
<record model="ir.ui.view" id="view_id">
  <field name="name">Nom_de_la_vista</field>
  <field name="model">Objecte_al_que_fa_referència</field>
  <field name="arch" type="xml">
    <!-- Aquí va el contingut específic de la vista->
    <VIEW_TYPE>
      <VIEW_SPECIFICATIONS/>
    </VIEW_TYPE>
    <!-- Fi del contingut específic de la vista->
  </field>
</record>
```

Aquesta **sintaxi genèrica** és comuna a tots els diferents tipus de vistes, i per tant cal entendre-la.

A banda, com hem dit, hi ha diferents **tipus de vistes**, cadascuna amb la seva sintaxi i especificacions; com ara:

- Vista de llista o arbre
- Vista de formulari
- Vista de gràfics
- Vista de calendari
- Vista de cerca

Vista de llista o arbre

"Views/List"

Aquesta part de la teoria correspon al capítol "Views/List" de la documentació oficial d'Odoo: tinyurl.com/y7rr7hx3.

Aquest tipus de vistes consisteix en una distribució de línies amb l'objectiu de **facilitar la visualització** i/o edició d'un conjunt de recursos d'un objecte. Igual que per a la resta de vistes, generarem el codi estàndard a partir de l'atribut `arch`, personalitzant la següent línia:

```
<field name="arch" type="xml">
    <tree atributs...>
    <diferents etiquetes>
  </tree>
</field>
```

Hi ha, entre d'altres, els següents **atributs**:

- `editable`: per defecte, per editar una llista fem clic sobre una de les línies, i s'obrirà el formulari definit per a l'edició. No obstant això, l'atribut `editable` ens permet fer-ho sobre la mateixa llista. Pot prendre els valors `top` o `bottom`.
- `default_order`: canvia l'ordre per defecte per un de personalitzat.
- `decoration` - {$ name}: permet canviar el color de les línies de la llista, en funció d'una variable determinada. S'introdueix una expressió Python per a ser avaluada. {$ Name} pot ser substituït pels valors següents:
  - Valors html (per exemple bf, font-weight: bold, it, font-style: italic)
  - Etiquetes predeterminades (`danger`, `info`, `muted`, `primary`, `success`, `warning`).
- `create, edit, delete`: permet desactivar aquestes funcions si l'atribut es col·loca a valor False.

Sota l'**etiqueta <tree>** podem trobar els següents elements:

- `button`: mostra un botó. Entre d'altres porta els atributs:
  - `icon`: icona del botó.
  - `string`: text del botó.
  - `type`: tipus del botó (`workflow`, `object`, `action`).
  - `states`: fa el botó invisible si no es compleix la condició.
  - `confirm`: missatge de confirmació per a la vostra acceptació.
- `Field`: l'element més habitual. Mostra un camp. Entre d'altres porta els atributs:
  - `name`: nom del camp.
  - `string`: text de la columna (per defecte l'etiqueta del camp)
  - `invisible`: es té en compte per a Odoo, però no apareix a la llista. Imprescindible si el camp és una de les condicions per canviar la decoració de la taula.
  - `groups`: grups que poden veure el camp.

Exemple de vista de llista o arbre

A la figura 2.22 i la figura 2.21 podeu veure la vista creada per a la classe `manteni.machine`.

Figura 2.21. Vista de llista de la classe manteni.machine

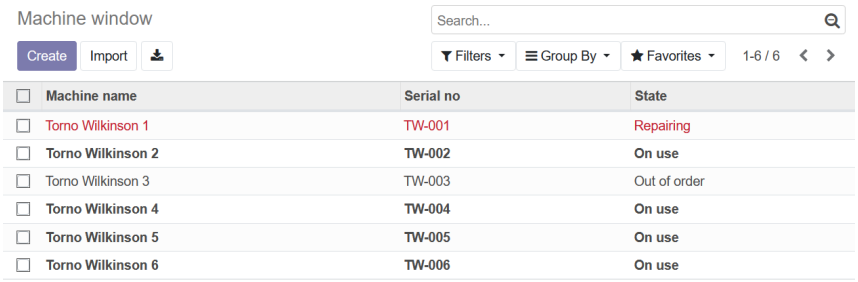


Figura 2.22. Codi per a la vista de llista de la classe manteni.machine

```
<record model="ir.ui.view" id="manteni.machine_list">
  <field name="name">Machine list</field>
  <field name="model">manteni.machine</field>
  <field name="arch" type="xml">
    <tree string="Machine list" decoration-bf="state=='on_use'" decoration-danger="state=='repairing'">
      <field name="name"/>
      <field name="serial"/>
      <field name="state"/>
    </tree>
  </field>
</record>
```

Vista de formulari

Views/Form

Aquesta part de la teoria correspon al capítol "Views/Form" de la documentació oficial d'Odoo: tinyurl.com/y752ltg8.

La vista de formulari està pensada per a **mostrar la informació d'un sol registre**. A diferència de la vista de llista, que és clarament funcional, la vista de formulari té un important component estètic: és clau ordenar la informació de manera clara i agradable. Igual que per a la resta de vistes, generarem el codi estàndard a partir de l'atribut `arch`, personalitzant la següent línia:

```
<field name="arch" type="xml">
    <form atributs...>
    <diferents etiquetes>
  </form>
</field>
```

En aquestes vistes diferenciarem els components estructurals (serveixen per a fixar l'estètica del formulari) i els components semàntics (indiquen quins camps sortiran al formulari).

Components estructurals:

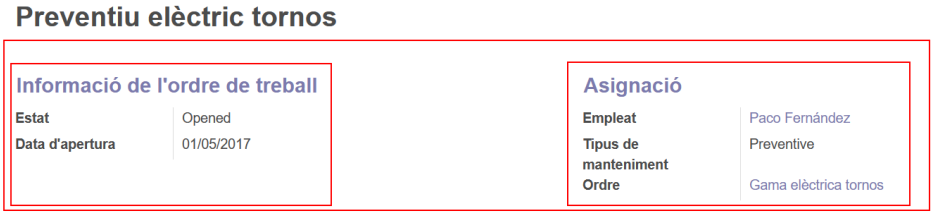
Aquests components estan pensats per a proveir ajuda visual al formulari. Són els següents:

- *sheet*: es fica a sota de l'etiqueta <form>, i serveix per crear una fulla, que fa el formulari més *responsive* (adaptable a pantalles de diferents mides).
- *header*: és la capçalera del formulari, que en edició avançada es fa servir per a col·locar botons o marcadors d'estat.
- *group*: és molt important per a la vista "form", ja que permet definir l'estructura de columnes del formulari. Dins un grup es creen dues columnes diferents, de manera que si fiquem una etiqueta <field>, mostrarà el nom del camp en una columna i el seu valor en una altra. El nombre de columnes que té un grup pot personalitzar-se mitjançant l'atribut `col`. Es pot ficar una etiqueta <group> dins una altra, de manera que subdividim el formulari en diferents columnes (vegeu la figura 2.23 i la figura 2.24).

Figura 2.23. Grups dins un altre grup a la vista form

```
<group>
    <group string="Workorder info">
        <field name="state" />
        <field name="date_begin" string="Opening date"/>
        <field name="date_end" string="Closing date" attrs="{'invisible':[('state','!=', 'closed')]}"/>
    </group>
    <group string="Assignation">
        <field name="employee_id"  />
        <field name="employee_user_id" invisible="1"/>
        <field name="type" />
        <field name="program_id" attrs="{'invisible':[('type', '=', 'corrective')]}"/>
    </group>
</group>
```

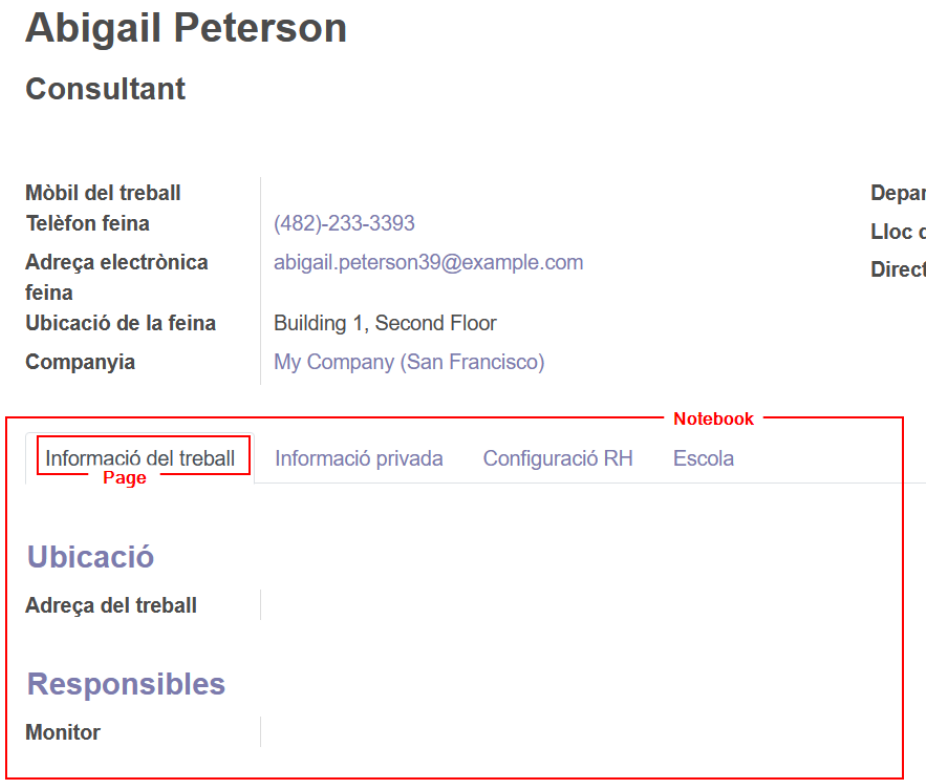
Figura 2.24. Resultat a Odoo del codi de la figura anterior



- *notebook*: defineix una secció amb pestanyes. Cada pestanya serà un element amb l'etiqueta <page>.

– *page*: el seu atribut més important és `string` (obligatori), que doni el títol de cada pestanya (figura 2.25).

FIGURA 2.25. Ús de 'notebook' i 'page' a la vista d'empleat



- *newline*: crea una nova fila a partir de l'etiqueta.
- *separator*: petita línia horitzontal (només té un efecte estètic).

Components semàntics:

Aquests components permeten la interacció amb el sistema Odoo. Són els botons, i els camps. Els botons es comporten de la mateixa manera que els de la vista de llista, i per tant no es tornaran a comentar. Ens centrem, doncs, en els camps <field>.

Hi ha diferents atributs que poden acompanyar els camps <field>. Els principals són:

- `name`. És obligatori i indica el nom del camp.
- `widget`. Els diferents tipus de camps es mostren de diferent manera per defecte. Amb aquest atribut es pot obligar a canviar l'aspecte. Presentem alguns exemples dels `widgets` per defecte per a diferents tipus de camps (figura 2.26, figura 2.27, figura 2.28 i figura 2.29):

Figura 2.26. Widget per defecte per als camps tipus Select

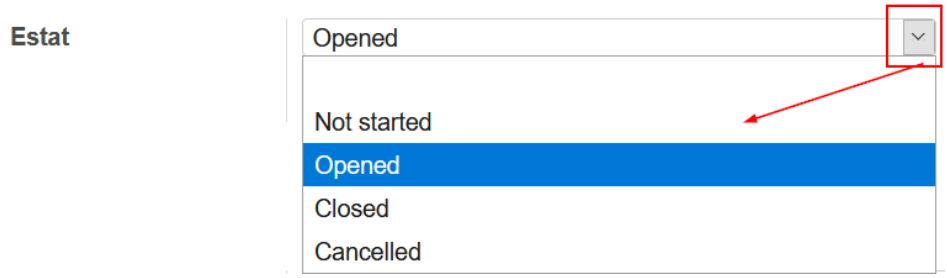


Figura 2.27. Widget per defecte per als camps tipus Date

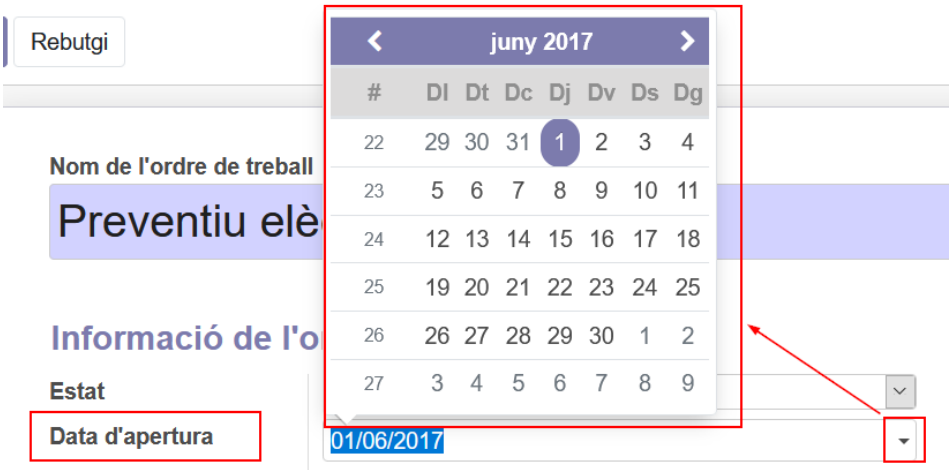


Figura 2.28. Widget per defecte per als camps tipus Many2one

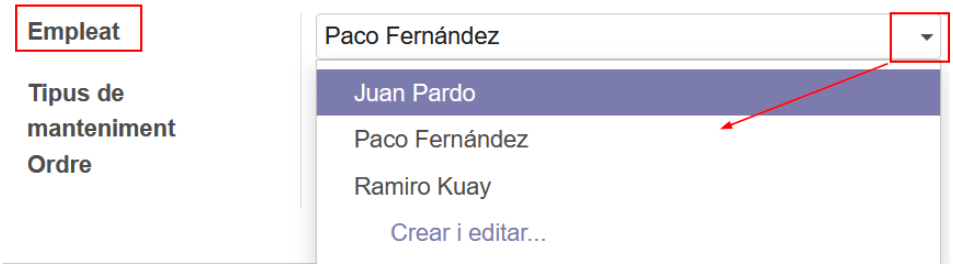


Figura 2.29. Widget per defecte per als camps tipus One2many i Many2many

- `class`. Tenim la possibilitat d'afegir classe `CSS`, de manera que podem canviar com es mostren els diferents camps. Alguns exemples:
  - `oe_inline`. Cancel·la un salt de línia entre dos camps.
  - `oe_left`, `oe_right`. Fa flotar el contingut en la direcció indicada.
  - `oe_read_only`, `oe_edit_only`. Fa que el camp només es mostri en el mode indicat (lectura o edició).

- `groups`. Restringeix la visualització del camp a certs grups (figura 2.30).

Figura 2.30. Ítem de menú del mòdul manteni restringit al grup manteni_manager

```
<menuitem name="Calendar" id="manteni.menu_1b" parent="manteni.menu_root" sequence="3"
          action="manteni.workorder_calendar_action_window"
          groups="group_manteni_manager"/>
```

- `domain`. Per a camps relacionals, determina els valors que hauran de mostrar-se, amagant la resta.
- `Context`. Serveix per crear filtres, enviant els camps implicats i la igualtat que han de complir (figura 2.31).

Figura 2.31. Ús de domain i context a una vista search del mòdul manteni

```
<filter string="Opened" name="opened" domain="[('state','=','opened')]"/>
<group string="Group By">
<filter name="group_by_program" string="Program" context="{'group_by': 'program_id'}"/>
```

- `readonly`. Fa el camp de només lectura.
- `required`. Fa el camp obligatori.
- `nolabel`. Evita mostrar l'etiqueta, quan el camp està dins un <group>.
- `help`. Mostra un `tooltip` amb un text d'ajuda.

Vista de gràfics

"Views/Graph"

Aquesta part de la teoria correspon al capítol "Views/Graph" de la documentació oficial d'Odoo: tinyurl.com/y9kc25qe.

Aquesta vista està pensada per a **visualitzar informació de manera gràfica**. Igual que per a la resta de vistes, generarem el codi estàndard a partir de l'atribut `arch`, personalitzant la següent línia:

```
<field name="arch" type="xml">
    <graph atributs...>
    <diferents etiquetes>
  </graph>
</field>
```

Es poden fer servir **dos atributs**, cap d'ells obligatori. Són els següents:

- `type`: a triar entre `bar` (per defecte), `pie` i `line` el tipus de gràfic que es farà servir.
- `stacked`: només per als gràfics de barres (`bar`). Agrupa les barres.

L'única etiqueta permesa dins els gràfics és <field>. Pot tenir els següents atributs:

- `name` (obligatori). El nom del camp que farem servir per a la vista.
- `title` (opcional). Text que es mostra a sobre del gràfic.

- `type`. Indica per a què es farà servir el camp. Pot tenir els valors:
  - `row`: el camp es farà servir per a agrupar (eix X).
  - `col`: només es fa servir per a taules.
  - `measure`: el camp servirà per a quantificar els camps agrupats (eix Y).
- `interval`. Per als camps tipus `date` o `datetime`, agrupa segons l'interval indicat (`day`, `week`, `month`, `quarter` o `year`).

Exemple de vista de gràfics

Com a exemple, es pot veure el gràfic generat al mòdul `hr_attendance`, que mostra les assistències dels treballadors de l'empresa. Com a camps agrupadors tenim `employee_id` i `check_in`, és a dir, presentarà a l'eix X els treballadors que han comunicat la seva entrada. Com a camp de mesura tenim `worked_hours`, és a dir, anirà sumant a l'eix Y les hores d'assistència de cada treballador (figura 2.32 i figura 2.33):

Figura 2.32. Gràfic generat automàticament per a la classe hr_attendance

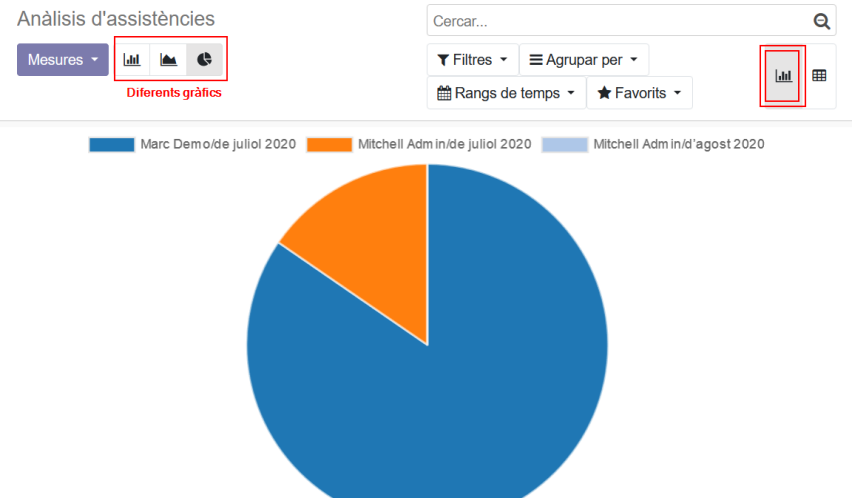


Figura 2.33. Codi font de la vista de gràfic per a la classe hr_attendance

```
<record id="hr_attendance_view_graph" model="ir.ui.view">
    <field name="name">hr.attendance.graph</field>
    <field name="model">hr.attendance</field>
    <field name="arch" type="xml">
        <graph string="Attendance">
            <field name="employee_id"/>
            <field name="check_in"/>
            <field name="worked_hours" type="measure"/>
        </graph>
    </field>
</record>
```

Vista de calendari

"Views/Calendar"

Aquesta part de la teoria correspon al capítol "Views/Calendar" de la documentació oficial d'Odoo: tinyurl.com/y8uv2pow.

Aquesta vista està pensada per a **visualitzar informació en forma de calendari**. Igual que per a la resta de vistes, generarem el codi estàndard a partir de l'atribut `arch`, personalitzant la següent línia:

```
<field name="arch" type="xml">
    <calendar atributs...>
    <diferents etiquetes>
  </calendar>
</field>
```

Els **atributs** que es poden fen servir (entre d'altres) són:

- `date_start` (obligatori). Camp que definirà la data de començament de cada esdeveniment.
- `date_stop`. Camp que definirà la data de finalització de cada esdeveniment.
- `date_delay`. Alternatiu a `date_stop`. Camp que definirà la durada d'un esdeveniment.
- `color`. Indica el camp per a fer la segmentació per colors. Tots els registres amb el mateix valor en aquest camp compartiran color en el calendari.
- `all_day`. `Boolean` que indica si els esdeveniments seran sempre de tot el dia.
- `mode`. Indiquem el mode per defecte del calendari: `day`, `week` o `month`.

Exemple de vista de calendari

A continuació, podeu veure la vista del calendari (figura 2.34) i el codi per a la vista (figura 2.35) per a la classe `manteni.workorder`.

Figura 2.34. Vista de calendari de la classe manteni.workorder

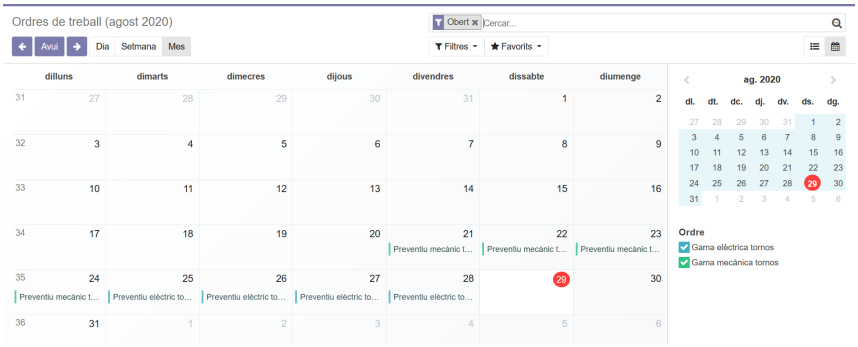


Figura 2.35. Codi per a la vista de llista de la classe manteni.workorder

```
<!-- calendar view -->
    <record model="ir.ui.view" id="manteni.workorder_calendar">
        <field name="name">workorder.calendar</field>
        <field name="model">manteni.workorder</field>
        <field name="arch" type="xml">
            <calendar string="Session Calendar" date_start="date_begin" color="program_id" mode="month">
                <field name="name"/>
            </calendar>
        </field>
    </record>
```

Vista de cerca

Aquesta part de la teoria correspon al capítol "Views/Search" de la documentació oficial d'Odoo: següent.

Aquesta vista està pensada per a crear criteris de cerca a les vistes de llista. Igual que per a la resta de vistes, generarem el codi estàndard a partir de l'atribut `arch`, personalitzant la següent línia:

```
<field name="arch" type="xml">
    <search atributs...>
    <diferents etiquetes>
  </search>
</field>
```

Podem crear quatre funcionalitats diferents:

- `field`: indica en quin camp haurà de buscar quan introduïm un text en el requadre de cerca. Odoo farà la cerca mitjançant el criteri escrit en el *textbox*, buscant en tots els `fields` que s'han introduït amb l'operació AND. Es poden afegir (entre d'altres), els següents atributs:
  - `name`: nom del camp.
  - `string`: etiqueta del camp.
  - `operator`: indica l'operador que farà servir Odoo per a buscar. Per exemple, quan busquem en un camp de text, Odoo generarà una cerca del tipus [(name, ilike, valor_introduït)]. Si afegim l'atribut `operator` podem canviar aquest `ilike` per defecte per un altre operador.
  - `filter_domain`: permet definir completament la cerca que volem fer. Accepta una llista amb una taula Python de tres elements (`name`, `operator`, `provided_value`).
  - `context`: permet incloure condicions (context).
  - `groups`: fa visible el camp només a grups d'usuaris concrets.
- `filter`: és una mena d'interruptor en la vista de cerca. Només es pot habilitar o deshabilitar. Aplicarà directament un criteri establert i mostrarà els resultats directament. Entre d'altres, pot tenir els següents atributs.
  - `string` (obligatori): el text que es mostrarà.
  - `domain`: crea el domini de cerca per a mostrar en pantalla.
  - `date`: nom d'un camp tipus `date` o `datetime`. Crea un submenú amb diverses opcions temporals.
  - `context`: diccionari de Python que es barreja amb el domini, per a crear el domini de cerca. Es fa servir molt quan volem fer una agrupació.
  - `help`: `tooltip` d'ajuda que es mostrarà.
  - `groups`: grups que poden fer servir el filtre.
- `separator`: serveix per a separar grups de filtres.
- `group`: serveix per a separar grups de filtres, més fàcil de llegir que el *separator*.

Exemple de vista de cerca

A continuació, podeu veure la vista de cerca (figura 2.36) i el codi per a la vista (figura 2.37) per a la classe `manteni.workorder`.

Figura 2.36. Vista de cerca de la classe manteni.workorder

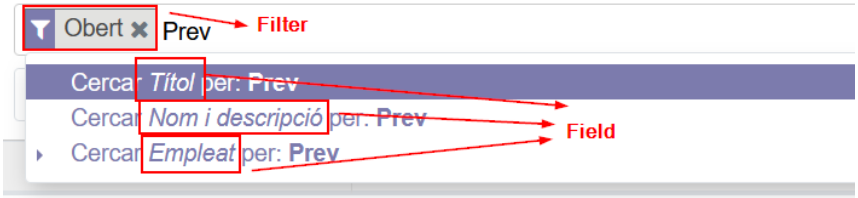


Figura 2.37. Codi per a la vista de cerca de la classe manteni.workorder

```
<!-- Search view -->
<record id="manteni.workorder_search_view"  model="ir.ui.view">
  <field name="model">manteni.workorder</field>
  <field name="arch" type="xml">
    <search>
        <field name="name"/>
        <field name="description" string="Name and description"
            filter_domain="['|', ('name', 'ilike', self), ('description', 'ilike', self)]"/>
        <field name="employee_id"/>
        <field name="program_id" widget="selection"/>
        <filter string="Opened" name="opened" domain="[('state','=','opened')]"/>
        <group string="Group By">
        <filter name="group_by_program" string="Program" context="{'group_by': 'program_id'}"/>
        </group>
    </search>
  </field>
</record>
```

Altres vistes

Hi ha més tipus de vistes a Odoo, les descripcions de les quals es poden trobar a la documentació oficial d'Odoo:

- Activity
- Cohort (només disponible a Odoo Enterprise)
- Dashboard (només disponible a Odoo Enterprise)
- Gantt (només disponible a Odoo Enterprise)
- Kanban
- Map (només disponible a Odoo Enterprise)
- Pivot

2.5.3 Accions

Les accions defineixen el comportament del sistema en resposta a una acció: `login`, prémer un botó, seleccionar un valor... Per a declarar una acció se n'haurà d'indicar el nom i el tipus. Hi ha diversos **tipus**:

"Actions"

Aquesta part de la teoria correspon al capítol "Actions" de la documentació oficial d'Odoo: tinyurl.com/yaqmxs5t.

- Obrir una finestra: `ir.actions.act_window`
- Obrir un web: `ir.actions.act_url`

- Executar un mètode al servidor: `ir.actions.server`
- Executar un informe: `ir.actions.report.xml`
- Executar una acció implementada enterament en el client: `ir.actions.client`
- Accions executades automàticament i periòdicament pel servidor: `ir.cron`

En aquest manual ens cenyirem exclusivament a les accions de finestra, ja que són les necessàries per al desenvolupament bàsic de mòduls. Podeu trobar informació detallada sobre la resta d'accions a la documentació oficial d'Odoo.

Accions "ir.actions.act_window"

La més comuna de les accions serveix per obrir una finestra de visualització d'un model mitjançant les seves vistes. Definirà el conjunt de vistes que es poden obrir (figura 2.38).

"Actions / Windows actions"

Aquesta part de la teoria correspon al capítol "Actions / Windows actions" de la documentació oficial d'Odoo: tinyurl.com/y8l26wwj.

Els seus **camps** (entre d'altres) són:

- `res_model`: classe de la qual prové l'acció.
- `view_mode`: llista separada per comes de les vistes disponibles, posant com a primera la que volem que aparegui per defecte.
- `Context` (opcional): informació addicional que passem a la vista.
- `Domain` (opcional): domini de filtratge que passem a les vistes.

Figura 2.38. Codi per a l'acció d'obrir la finestra de la classe manteni.workorder

```
    <!-- actions opening views on models -->
<record model="ir.actions.act_window" id="manteni.workorder_action_window">
    <field name="name">Work orders</field>
    <field name="res_model">manteni.workorder</field>
    <field name="view_mode">tree,form,calendar</field>
    <field name="context">{'search_default_opened': True}</field>
</record>
```

2.5.4 Generació de menús

"Data files / Shortcut / Menuitem"

Aquesta part de la teoria correspon al capítol "Data files / Shortcut / Menuitem" de la documentació oficial d'Odoo: tinyurl.com/yata9nlz.

Per a generar un menú en un mòdul d'Odoo, farem servir l'etiqueta **<menuitem>**, que cridarà a una acció que cridarà a les respectives vistes.

Els **valors específics** de cada menú són:

- `name`: és el títol del menú, és opcional; si no ho posem al menú, prendrà el nom de l'acció que executa.
- `id`: conté l'identificador XML del menú; ha de ser únic dins el mòdul.
- `action`: identificador de l'acció que executa el menú. Si no especifiquem l'acció, el sistema entén que es tracta d'un menú arrel i per tant que conté

altres menús i/o opcions. El disseny de la corresponent acció depèn del tipus d'execució que porta associada (obrir finestra, executar informe, arrencar un assistent...).

- `web_icon`: icona que mostra el client web a la pantalla d'instal·lació del mòdul.
- `parent`: identificador del menú pare al qual pertany. Si no ho especifiquem, el sistema entén que es tracta d'un menú arrel (menú principal).
- `groups`: grup d'usuaris que poden veure el menú. Si volem especificar més d'un grup, cal separar-los per comes (groups = "admin, user").

Exemple de la generació de menús pel mòdul **manteni**

```
    <!-- Top menu item -->
    <menuitem
            name="Maintenance"
            id="manteni.menu_root"
            web_icon="manteni,static/description/icon.png" groups
                ="group_manteni_user"/>

    <!-- menu categories -->
    <menuitem name="Config" id="manteni.menu_3" parent="manteni.
        menu_root" sequence="4"
                groups="group_manteni_manager"/>
    <!-- actions -->
    <menuitem name="Workorders" id="manteni.menu_1a" parent="
        manteni.menu_root" sequence="1"
              action="manteni.workorder_action_window" groups="
                  group_manteni_user"/>
    <menuitem name="Calendar" id="manteni.menu_1b" parent="
        manteni.menu_root" sequence="3"
              action="manteni.workorder_calendar_action_window"
                groups="group_manteni_manager"/>
    <menuitem name="Machine list" id="manteni.menu_2_1" parent="
        manteni.menu_3" sequence="5"
              action="manteni.machine_action_window"/>
    <menuitem name="Program list" id="manteni.menu_2_2" parent="
        manteni.menu_3" sequence="6"
              action="manteni.program_action_window"/>
    <menuitem id="res_partner_menu"
            parent="manteni.menu_3"
            action="action_machinery_seller_form"
            sequence="7" groups="group_manteni_manager"/>
```

Exemple de creació de les vistes d'un mòdul (vídeo)

Per a descriure amb detall la confecció de les vistes d'un mòdul mitjançant un exemple diferent, podeu veure el següent vídeo.

https://player.vimeo.com/video/473780744

2.6 El controlador a Odoo

Odoo és un *software* molt madur i que proporciona els mètodes per a, entre d'altres, crear, modificar i esborrar registres. Tot i que el llenguatge que hi ha a sota és Python, el desenvolupador no ha d'obsessionar-se amb aquest llenguatge, ja que es parteix d'una gran base. La capa ORM d'OpenObject facilita un seguit de mètodes que s'encarreguen del mapatge entre els objectes Python i les taules de PostgreSQL. Així, disposem de mètodes per crear, modificar, eliminar i cercar registres a la base de dades. Aquests mètodes són utilitzats de manera automàtica per OpenObject en l'execució dels diversos tipus de vista que OpenObject ens permet dissenyar.

En ocasions, però, pot ser necessari **alterar l'acció automàtica** de cerca–creació–modificació–eliminació facilitada per OpenObject i, llavors, haurem de sobreescriure els corresponents mètodes en les nostres classes.

Com a exemple d'aquesta necessitat, podem considerar el cas de la gestió de recursos humans (classe `hr.employee`) dins el fitxer hr.py del mòdul hr d'Odoo. Si hi donem una ullada, al final de la classe hi trobem el disseny, entre d'altres, dels mètodes *unlink* (eliminar) i *write* (modificar). Cadascun d'ells executa un seguit de comprovacions i/o accions i, si tot és correcte, invoca la crida dels corresponents mètodes *unlink*, *create* i *write* de la capa ORM. Així, el mètode *unlink* (eliminació de treballadors) elimina també el registre de la classe `resource` associat al treballador abans d'esborrar el registre de la seva taula.

Redefinició dels mètodes 'write' i 'unlink' per a la classe "hr.employee"

```
    def write(self, vals):
        if 'address_home_id' in vals:
            account_id = vals.get('bank_account_id') or self.
                bank_account_id.id
            if account_id:
                self.env['res.partner.bank'].browse(account_id).
                    partner_id = vals['address_home_id']
        if vals.get('user_id'):
            vals.update(self._sync_user(self.env['res.users'].
                browse(vals['user_id'])))
        res = super(HrEmployeePrivate, self).write(vals)
        if vals.get('department_id') or vals.get('user_id'):
            department_id = vals['department_id'] if vals.get('
                department_id') else self[:1].department_id.id
            # When added to a department or changing user,
                subscribe to the channels auto-subscribed by
                department
            self.env['mail.channel'].sudo().search([
                ('subscription_department_ids', 'in',
                    department_id)
            ])._subscribe_users()
        return res

    def unlink(self):
        resources = self.mapped('resource_id')
        super(HrEmployeePrivate, self).unlink()
        return resources.unlink()
```

2.6.1 Mètodes ORM més comuns

ORM API / Common ORM method

Aquesta part de la teoria correspon al capítol "ORM API / Common ORM methods" de la documentació oficial d'Odoo: tinyurl.com/yb7mks2u.

Mètode *create(vals_list)* → *records*. Crea un o diversos registres per a l'objecte on es defineix. Com a paràmetre d'entrada s'haurà de ficar un diccionari o una llista de diccionaris (si es vol afegir més d'un camp) amb els valors que es volen escriure.

El **format del diccionari** o llista de diccionaris ha de ser el següent: [{'field_name': field_value, ...}, ...]

Mètode *write(vals)*. Prenent un registre o conjunt de registres, modifica un camp o conjunt de camps amb els valors introduïts. Si es tracta de més d'un registre, tots seran modificats amb els mateixos valors pels camps. Com a paràmetre d'entrada s'haurà de ficar un diccionari amb els valors que es volen modificar. El format serà {'camp1': valor1, 'camp2': "valor2"}, etc.

Mètode *browse([ids])* → *records*. Fent servir com a entrada un identificador o una llista de números identificadors, ens torna els registres de la classe que coincideixen amb aquest identificador.

Mètode *search(args[, offset=0][, limit=None][, order=None][, count=False])*. Torna un subconjunt de registres coincidents amb els paràmetres de la cerca. Pot tenir els paràmetres:

- `args` – Un domini de cerca. Si està buit agafarà tots els registres.
- `offset (int)` – Nombre de resultats que ha d'ignorar.
- `limit (int)` – Nombre de resultats màxim que ha de donar (*default: all*).
- `order (str)` – Text per a ordenar els resultats.
- `count (bool)` – Si el seu valor és True, només torna el nombre de resultats coincidents (`default: False`).

Exemple del mètode search

Suposem que volem buscar tots els treballadors d'una empresa, continguts a l'objecte `company`. Podrem fer servir l'expressió: search([('company_id', '=', company.id)]).

Aquesta expressió pot trobar-se a l'arxiu hr_employee.py de la classe `hr_presence`.

Mètode *unlink()*. Esborra el registre o registres indicats.

2.6.2 Els decoradors

"ORM API / Method decoratos"

Aquesta part de la teoria correspon al capítol "ORM API / Method decoratos" de la documentació oficial d'Odoo: tinyurl.com/ya89leh9.

Tal com ocorre en el model d'Odoo amb els camps calculats, de vegades s'han de desenvolupar funcions per a modificar o ampliar les funcionalitats d'Odoo.

Si anem revisant el codi font, trobarem sentències que comencen amb una "@" a sobre d'aquestes funcions. Són els decoradors. Els decoradors permeten **modificar la manera de comportar-se d'una funció** (figura 2.39).

Els decoradors no són una funcionalitat exclusiva d'Odoo, són un recurs molt utilitzat a Python.

Els decoradors @api.multi i @api.one no estan permesos a partir d'Odoo 13, i per tant hauran d'eliminar-se si tenim mòduls preparats per a altres a versions.

FIGURA 2.39. Decorador a la classe "hr_employee"

```
@api.depends('address_home_id.parent_id')
def _compute_is_address_home_a_company(self):
    """Checks that chosen address (res.partner) is not linked to a company.
    """
    for employee in self:
        try:
            employee.is_address_home_a_company = employee.address_home_id.parent_id.id is not False
        except AccessError:
            employee.is_address_home_a_company = False
```

Els decoradors més interessants per al desenvolupament de mòduls a Odoo són:

- @api.depends
- @api.constrains
- @api.onchange

No s'ha de confondre "@api.constrains" amb "constraints"; podeu trobar informació a tinyurl.com/y44krt97.

@api.depends

Aquest decorador pot completar la funció d'un camp calculat, indicant quan s'ha de realitzar aquest càlcul, és a dir, de quins altres camps depèn.

Com a exemple tenim la funció `_amount_all` del mòdul de compres (codi font: tinyurl.com/yce2vu4r). Per a una compra, la funció d'actualització de les quantitats subtotal, total d'impostos i total només es "dispara" quan canvia el camp `price_total` de la classe `order_line`.

Destacarem també que @api.depends pot fer dependre una funció d'un camp que pertany a una classe diferent de la en la qual hi ha la funció.

Ús del decorador @api.depends

```
1     @api.depends('order_line.price_total')
2     def _amount_all(self):
3         for order in self:
4             amount_untaxed = amount_tax = 0.0
5             for line in order.order_line:
6                 amount_untaxed += line.price_subtotal
7                 amount_tax += line.price_tax
8             order.update({
9                 'amount_untaxed': order.currency_id.round(
                      amount_untaxed),
10                'amount_tax': order.currency_id.round(amount_tax)
                      ,
11                'amount_total': amount_untaxed + amount_tax,
12            })
```

@api.constrains

Aquest decorador serveix per a verificar certes condicions abans de guardar un registre a la base de dades.

A la classe `Contract` del mòdul de Recursos Humans (codi font tinyurl.com/yck3mnbt) hi ha una funció amb aquest tipus de decorador. Abans de guardar un contracte cal assegurar-se que la data d'inici no és més alta a la data de finalització. Per això es genera aquesta funció.

```
@api.constrains('date_start', 'date_end')
    def _check_dates(self):
        if self.filtered(lambda c: c.date_end and c.date_start > c.date_end):
            raise ValidationError(_('Contract start date must be earlier than
                contract end date.'))
```

@api.onchange

El decorador *onchange* i el decorador *depends* són similars, i moltes vegades poden fer-se servir indistintament. La principal diferència és que a @api.depends pot fer servir camps de classes externes a la classe on hi ha la funció.

A vegades pot resultar-nos útil canviar un valor abans de manera immediata, depenent del valor que ha pres un altre camp de la mateixa classe, per a això utilitzem les funcions `onchange`.

Els camps calculats, definits anteriorment, no necessiten una funció `onchange`, ja que s'actualitzen automàticament. Això és una novetat respecte a versions anteriors.

Per a la classe `manteni.workorder` hi ha dues funcions d'aquest tipus. La primera actualitza la data de finalització de l'ordre de treball si es tria l'estat *closed* a la data actual. La segona busca l'identificador d'usuari d'un treballador quan canviem l'empleat que durà a terme l'ordre de treball.

```
@api.onchange('state')
    def _onchange_state(self):
        if self.state == 'closed':
            self.date_end=date.today()

    @api.onchange('employee_id')
    def _onchange_employee_id(self):
        self.employee_user_id=self.employee_id.resource_id.user_id
```

Exemple de treball amb el controlador d'un mòdul (vídeo)

Per a descriure amb detall la creació de funcions i la modificació del controlador a Odoo mitjançant un exemple diferent podeu veure aquest vídeo.

https://player.vimeo.com/video/473779714

Sistemes ERP-CRM. Explotació i adequació - II

Mikel López Villarroya i Isidre Guixà Miranda

Sistemes de gestió empresarial

Índex

Introducció **5**

Resultats d'aprenentatge **7**

1 Odoo: desenvolupament avançat **9**
1.1 Seguretat a Odoo . . . 9
1.1.1 Usuaris, grups i permisos a Odoo . . . 9
1.1.2 Incorporar un esquema de seguretat al nostre mòdul . . . 12
1.2 Herència a Odoo . . . 15
1.2.1 Herència en el model . . . 16
1.2.2 Herència en la vista . . . 19
1.2.3 Herència en el controlador . . . 21
1.3 Dades d'exemple . . . 22
1.4 Traducció del mòdul . . . 25
1.4.1 Poedit . . . 25
1.4.2 Preparació dels arxius de la traducció . . . 25
1.4.3 Traducció del mòdul a Poedit . . . 27
1.4.4 Incorporació de la traducció a Odoo . . . 28

2 'Reporting' & BI **29**
2.1 'Reporting' extern: JasperReports i JasperStudio . . . 29
2.1.1 Instal·lació i configuració de JasperServer i JasperStudio (vídeo) . . . 31
2.1.2 Creació d'informes amb JasperStudio i generació des de JasperServer (vídeo) . . . 32
2.2 Generació d'informes natius d'Odoo . . . 32
2.2.1 Edició de la plantilla genèrica . . . 33
2.2.2 Creació d'un informe sobre una classe . . . 34
2.2.3 El llenguatge Qweb . . . 35
2.2.4 Llenguatge HTML als informes . . . 38
2.2.5 Exemple de la creació d'un informe a Odoo (vídeo) . . . 40

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'Odoo 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, 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 **"Odoo: 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, 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 Odoo, 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 Odoo es basa en utilitzar el llenguatge QWeb, molt senzill i potent, que permet fer servir unes etiquetes especials, amb el prefix t-, que enriqueixen la web amb camps d'Odoo. Mitjançant el programa wkhtmltopdf convertirem la web generada en un document pdf llest per a imprimir.

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 Odoo, 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 JasperStudio que utilitzarem per dissenyar una tipologia d'informes integrables en Odoo 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 que permet el disseny de quadres de comandament, totalment operativa, per a trenta dies.

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

1. La implantació tècnica d'Odoo. 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 Odoo. 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. Odoo: desenvolupament avançat

A la secció “Annexos”, del web del mòdul, trobareu diverses còpies dels recursos de programari que s’utilitzen en aquest contingut.

Una vegada es domina la generació de mòduls a Odoo, és important conèixer altres conceptes que, encara que són secundaris, donen al mòdul la funcionalitat necessària per a treballar a una empresa.

Els conceptes de desenvolupament avançat que cal conèixer són:

- **Esquema de seguretat**: ajuda a controlar l’accés al mòdul, creant grups i definint les atribucions de cadascun d’ells.
- **Herència**: permet modificar el nucli d’Odoo per adaptar-lo a les necessitats de l’empresa, però sense modificar el codi font de l’ERP.
- **Dades d’exemple**: ajuda a la comprensió del funcionament del mòdul a usuaris novells, mitjançant la incorporació de dades de mostra.
- **Traducció del mòdul**: davant la realitat de les empreses, on és habitual que tinguin sucursals a diferents països, s’inclou la possibilitat de traduir el mòdul, de manera que cada usuari pugui visualitzar-lo en l’idioma que tingui configurat al seu perfil.
- **Generació d’informes i BI**: un dels grans avantatges de fer servir un ERP és l’emmagatzematge massiu d’informació a una única base de dades. Agafar tota aquesta informació i presentar-la de manera que aporti valor afegit a la presa de decisions és vital a qualsevol empresa.

En aquest contingut es presentaran, un a un, tots aquests conceptes, i la seva aplicació pràctica a Odoo. Els diversos enllaços proporcionats corresponen a les versions més actuals, a la redacció d’aquests materials.

1.1 Seguretat a Odoo

La informació que s’emmagatzema dins un ERP és molt sensible, i és necessari que es controli totalment el seu accés. És per això que la seguretat suposa un punt molt important en el desenvolupament de qualsevol mòdul. En aquest apartat es generarà un esquema de seguretat basat en diferents grups d’usuaris, per a després atorgar a cadascun dels grups els privilegis que convingui.

"Security in Odoo"

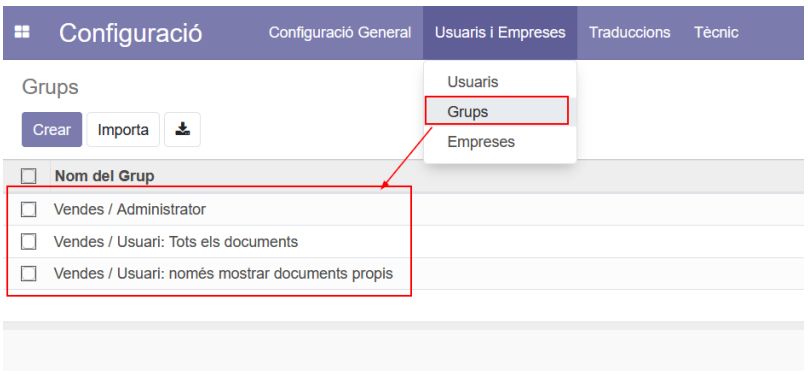
Aquesta part de la teoria correspon al capítol “Security in Odoo” de la documentació oficial d’Odoo: tinyurl.com/y7rrffbz.

1.1.1 Usuaris, grups i permisos a Odoo

Odoo disposa d’un sistema de privilegis basat en grups. Pel fet de pertànyer a un grup, un usuari tindrà dret a accedir a unes funcionalitats o unes altres.

Com a exemple es presentarà el mòdul de vendes (*sales*). Per conèixer els diferents accessos possibles s'ha d'anar en mode desenvolupador al mòdul de configuració, secció d'usuaris (figura 1.1).

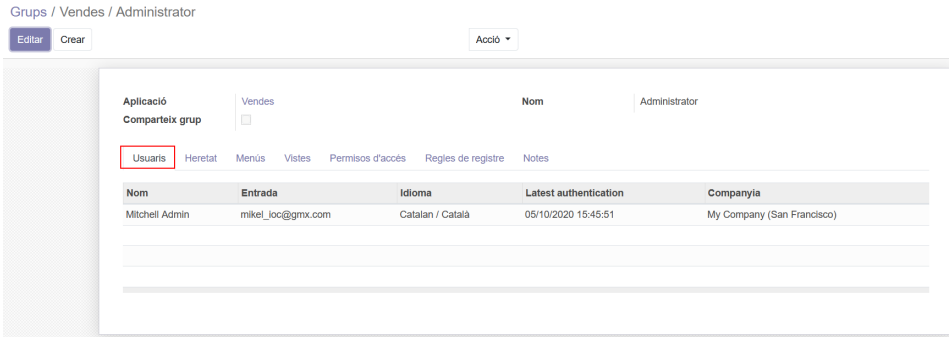
FIGURA 1.1. Grups associats al mòdul de vendes



Una vegada s'accedeix a la visualització del grup, poden trobar-se les següents opcions:

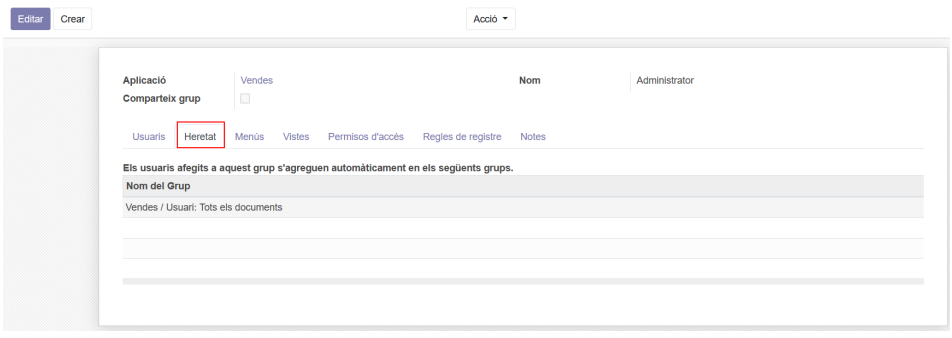
- **Usuaris membres del grup**: llista d'usuaris que formen part del grup. Possibilitat de donar-ne d'alta de nous (figura 1.2).

FIGURA 1.2. Usuaris membres del grup



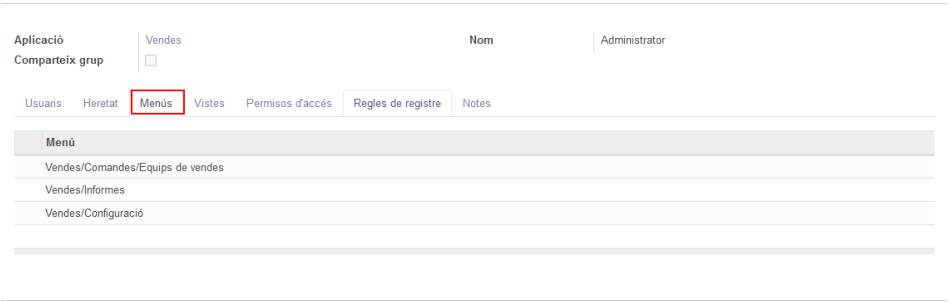
- **Grups dels quals s'hereten les característiques**: llista de grups dels quals aquest grup incorpora totes les seves característiques, sense necessitat d'haver-les d'especificar (figura 1.3).

FIGURA 1.3. Grups dels quals s'hereten les característiques



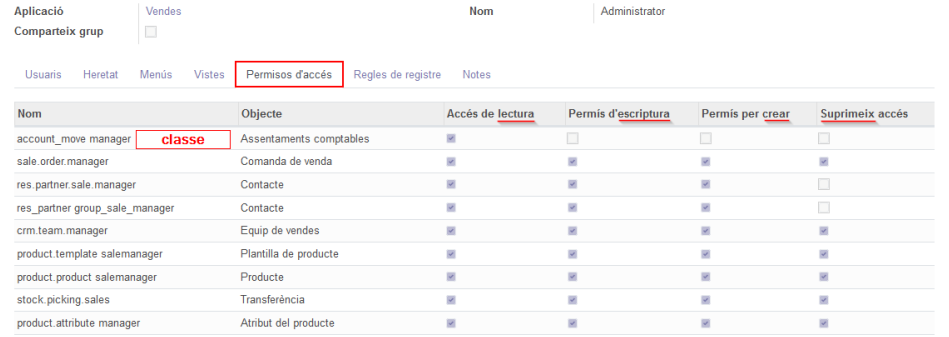
- **Menús específics** als quals aquest usuari té accés (figura 1.4).

FIGURA 1.4. Menús als quals accedeix l'usuari



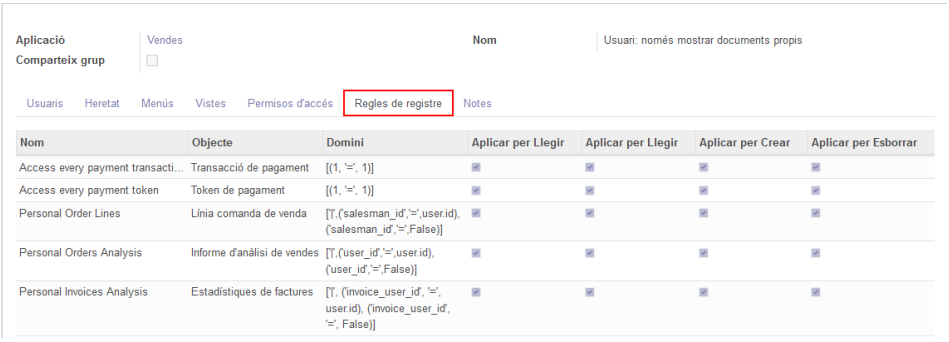
- **Permisos sobres les classes**: s'especifiquen totes les classes sobre les quals l'usuari té algun tipus de privilegi, i quins són, entre veure, editar, crear o suprimir (figura 1.5).

FIGURA 1.5. Permisos sobre les classes



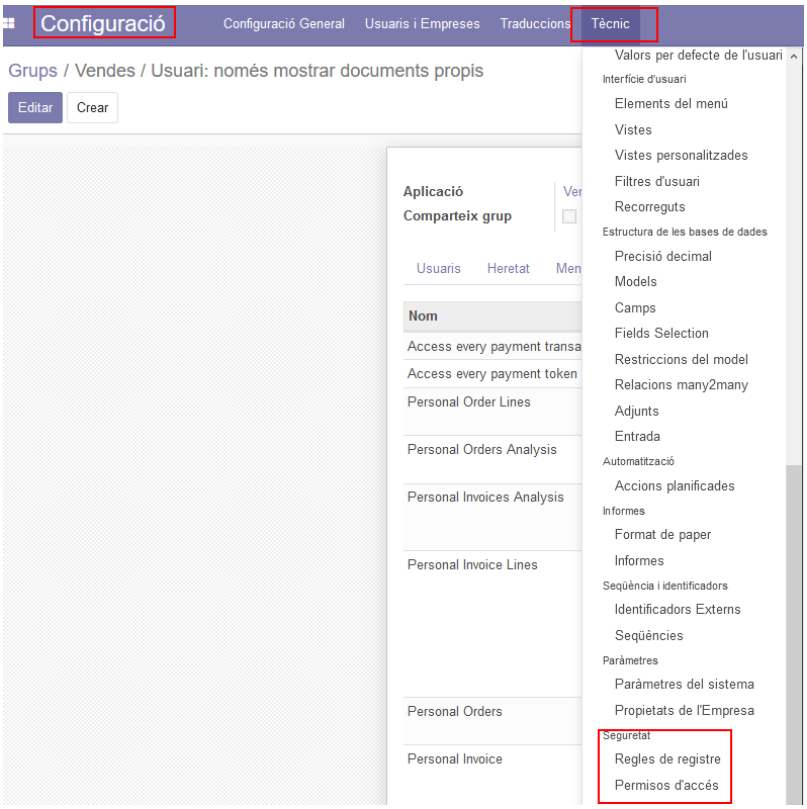
- També poden configurar-se **regles que limitin l'accés** als registres d'una classe. Per exemple, al grup de vendes "Usuari: només mostrar documents propis", les regles determinen que un usuari només pot accedir als documents que ell mateix ha generat (figura 1.6).

FIGURA 1.6. Regles per a l'accés als registres



A més a més, de cada grup en particular, poden trobar-se totes les regles i tots els permisos de l'ERP al menú *Configuració / Tècnic / Seguretat* (figura 1.7).

FIGURA 1.7. Menú de seguretat d'Odoo



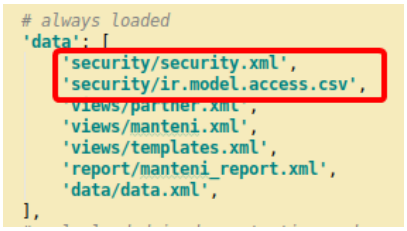
1.1.2 Incorporar un esquema de seguretat al nostre mòdul

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'Odoo es defineix en dos fitxers que s'acostumen a situar (no és obligatori) en una carpeta anomenada ***security*** i que han de ser referenciats al fitxer __manifest__.py del mòdul. Aquests fitxers són (figura 1.8):

- **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 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, com 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.

FIGURA 1.8. Arxius de seguretat al manifest



Security.xml

El fitxer 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"?>
<odoo>
  <data noupdate="1">
    <record id="idCategory" model="ir.module.category">
      <field name="name">nomGrup</field>
      <field name="description">Descripció...</field>
    </record>
    <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">
      <!-- Inclusió d'usuaris en el grup -->
    </record>
    <record id="idRegla" model="ir.rule">
      <!-- Definició de regles de negoci i assignació
        a grups i/o companyies -->
    </record>
  </data>
</odoo>
```

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" (figura 1.9):

FIGURA 1.9. Esquema bàsic de seguretat del mòdul manteni


```
<?xml version="1.0" encoding="utf-8"?>
<odoo>
    <data noupdate="1">

    <record model="ir.module.category" id="module_category_manteni">
        <field name="name">Maintenance</field>
        <field name="description">Helps you manage your factory's industrial maintenance</field>
        <field name="sequence">104</field>
    </record>

    <record id="group_manteni_user" model="res.groups">
        <field name="name">Maintenance Officer</field>
        <field name="category_id" ref="module_category_manteni"/>
    </record>

    <record id="group_manteni_manager" model="res.groups">
        <field name="name">Maintenance Manager</field>
        <field name="category_id" ref="module_category_manteni"/>
        <field name="implied_ids" eval="[(4, ref('group_manteni_user'))]"/>
        <field name="users" eval="[(4, ref('base.user_admin'))]"/>
    </record>

    </data>
</odoo>
```

- L'atribut **id** de l'element "record" del model "res.groups", ha de ser únic dins el mòdul; es tracta d'un identificador XML per al grup, al qual es pot fer referència des del mateix mòdul a través del seu nom o des de qualsevol altre mòdul a través de la sintaxi nomMòdul.identificador. (1)
  - 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*. (2)
- 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`. (3)
- 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`. (4)
- Si volem incloure de manera automàtica un usuari a un grup en la instal·lació, podem fer-ho de dues maneres:
  - Afegint un "record" del model "res.users", indicant-ne el grup al que s'afegeix.
  - Afegint al grup un camp amb el nom **"users"**. (5)

Inclusió d'un usuari

La sintaxi mostrada per a la inclusió d'un usuari en un grup (eval="[(4, ref('base.user_admin'))]") és pròpia d'Odoo, i serveix per afegir registres a un camp `One2many` o `Many2many`. Per a més informació pot consultar-se aquest tinyurl.com/yatfpcdg.

Definir prèviament categories i grups

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 manifest.py ha de ser posterior a la inclusió del fitxer security.xml.

ir.access.model.csv

Aquest fitxer 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 security.xml 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 (figura 1.10).

FIGURA 1.10. Fitxer ir.access.model.csv del mòdul "manteni" obert amb LibreOffice

	A	B	C	D	E	F	G	H
1	id	name	model_id:id	group_id:id	perm_read	perm_write	perm_create	perm_unlink
2	access_manteni_machine_manager	manteni.machine.manager	model_manteni_machine	group_manteni_manager	1	1	1	1
3	access_manteni_machine_user	manteni.machine.user	model_manteni_machine	group_manteni_user	1	0	0	0
4	access_manteni_program_manager	manteni.program.manager	model_manteni_program	group_manteni_manager	1	1	1	1
5	access_manteni_program_user	manteni.program.user	model_manteni_program	group_manteni_user	1	0	0	0
6	access_manteni_program_instruction_manger	manteni.program.instruction.manager	model_manteni_program_instruction	group_manteni_manager	1	1	1	1
7	access_manteni_program_instruction_user	manteni.program.instruction.user	model_manteni_program_instruction	group_manteni_user	1	0	0	0
8	access_manteni_workorder_manager	manteni.workorder.manager	model_manteni_workorder	group_manteni_manager	1	1	1	1
9	access_manteni_workorder_user	manteni.workorder.user	model_manteni_workorder	group_manteni_user	1	1	0	0
10	access_hr_employee_manager	hr.employee.user	hr.model_hr_employee	group_manteni_user	1	0	0	0
11	access_hr_employee_category_manager	hr.employee.category.user	hr.model_hr_employee_category	group_manteni_user	1	0	0	0

Exemple de la creació d'un esquema de seguretat (vídeo)

A continuació podeu veure un vídeo de tot el procés de creació de l'esquema de seguretat d'un mòdul.

https://player.vimeo.com/video/473782890

1.2 Herència a Odoo

El mecanisme d'herència permet als programadors adaptar mòduls existents garantint que les actualitzacions no provoquin funcionaments inesperats. Com a consultors d'Odoo, pot ser molt habitual que el client ens demani afegir nous

camps, fer-ne d'invisibles o modificar-ne d'altres, i fins i tot canviar mètodes. Per això, com que Odoo és de codi obert podríem **modificar el codi font** del mòdul en qüestió, però no és recomanable. Hi ha el mecanisme de l'herència, que permet el mateix però de manera més innòcua per al sistema.

L'herència es pot aplicar en els tres components del patró model-vista-controlador:

- 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.

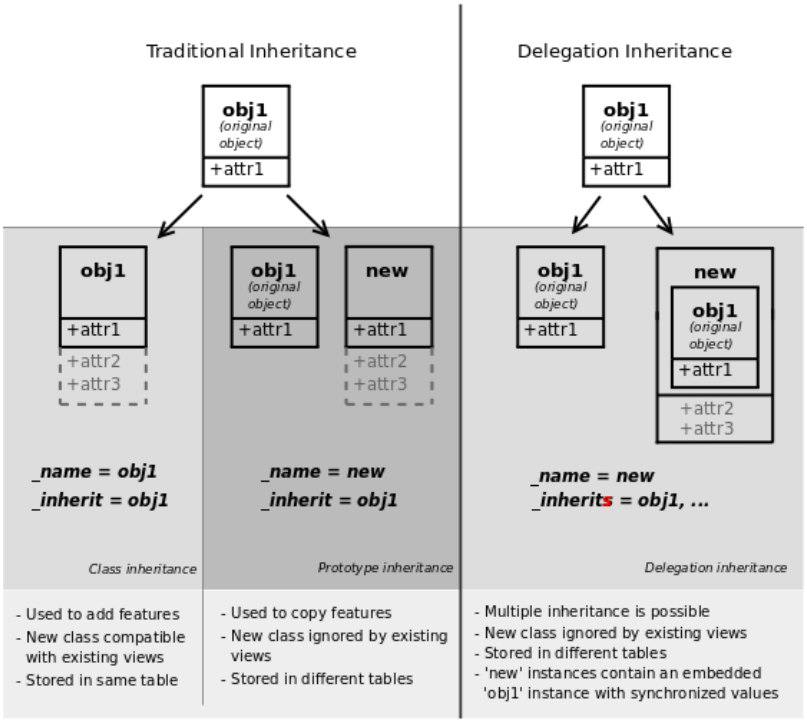
1.2.1 Herència en el model

"Inheritance and extension"

Aquesta part de la teoria correspon al capítol "Inheritance and extension" de la documentació oficial d'Odoo: tinyurl.com/yafnh2vm.

Tal com mostra la figura següent, hi ha tres tipus d'herència en el model a Odoo (figura 1.11):

FIGURA 1.11. Mecanismes d'herència en el model a Odoo



- Herència de **classe**: modificar una classe existent, afegint o traient característiques (mètodes, atributs...).
- Herència de **prototip**: crear una classe nova basada en una altra d'existent, afegint o traient característiques (mètodes, atributs...).

- Herència per **delegació**: vincula una nova classe amb una o més classes existents, de manera que quan es genera un nou objecte d'aquesta classe es genera un objecte de les classes vinculades.

El disseny d'un objecte (classe) heretat és gairebé idèntic al disseny d'un objecte (classe) 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.

Herència de classe

Aquest tipus d'herència, la més habitual, s'utilitza quan es volen afegir dades (`_columns`) i/o modificar propietats de dades existents i/o modificar el funcionament d'alguns mètodes. Les seves **característiques** són:

- 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.
- A 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 = obj1`
- El nom de la nova classe ha de continuar sent el mateix que el de la classe original: `_name = obj1`. Es pot decidir no posar-l'hi.

Com a exemple, podem veure l'arxiu partner.py del mòdul "manteni". Conté el següent codi:

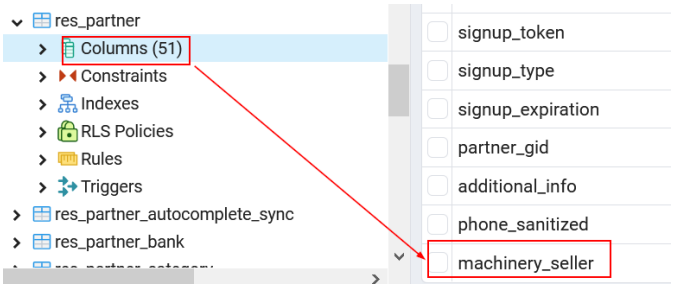
```
class Partner(models.Model):
    _inherit = 'res.partner'

    machinery_seller=fields.Boolean(string="Machinery seller", default=False)
```

Pel que fa a la **sintaxi**, cal destacar que la classe es diu igual que l'original i que s'ha afegit l'atribut `_inherit`, per declarar que la classe fa servir l'herència.

Aquesta classe serveix per incorporar a la classe original `res.partner` un nou camp a la base de dades, `machinery_seller`, del tipus booleà. Podem comprovar que, una vegada instal·lat el mòdul "manteni", aquest camp s'incorpora a la base de dades.

Com a **comprovació** de que l'herència ha estat un èxit, es recomana actualitzar el mòdul (havent reiniciat prèviament el servidor), i revisar a la base de dades que els canvis s'hagin aplicat. Per exemple, a la figura 1.11 pot comprovar-se com en instal·lar el mòdul "manteni" s'ha creat el camp `machinery_seller` a la classe `res_partner`.

FIGURA 1.12. Comprovació de la creació del nou camp al mòdul "manteni"



Herència per prototip

L'herència simple (`_inherit`) amb atribut `_name` diferent del 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.

Les seves **característiques** són:

- Aprofita la definició de la classe original (com si fos un "prototip").
- 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.
- A 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`

Com a exemple, hi ha diverses classes que hereten la classe `mail.thread`. Dota la classe de les funcions necessàries perquè la classe pugui interactuar amb les "discussions". Per exemple, la classe `hr.department` de l'arxiu hr.py conté aquest codi:

```
class Department(models.Model):
_name = "hr.department"
_description = "HR Department"
_inherit = ['mail.thread']
```

Herència per delegació

Aquest tipus d'herència **no s'utilitza gaire**, i a la documentació oficial d'Odoo es recomana no fer-lo servir (la frase concreta és *avoid it if you can*). Consisteix a crear una classe on certs camps es "deleguen" a altres classes (les anomenem classes filles).

Com a exemple podem veure l'estructura dels productes a Odoo. Existeix la classe `product.product`, que fa servir el següent codi a la seva creació.

```
class ProductProduct(models.Model):
    _name = "product.product"
    _description = "Product"
    _inherits = {'product.template': 'product_tmpl_id'}
```

Per tant, cada vegada que es generi una instància d'aquesta classe, automàticament es generarà una instància de la classe `product.template`. El camp que relaciona les dues classes serà `product_tmpl_id`. De fet, el camp amb el nom del producte es troba a la classe `product.template`, i no a `product.product`.

1.2.2 Herència en la vista

"Views: Inheritance"

Aquesta part de la teoria correspon al capítol "Views: Inheritance" de la documentació oficial d'Odoo: tinyurl.com/ybkd395r.

Quan fem herència en una classe, es poden fer servir les vistes de l'objecte pare. No obstant això, també ens pot interessar **disposar d'una versió retocada**; en aquest cas, el millor és heretar de les vistes existents i modificar, afegir o eliminar camps i no substituir totalment la vista original.

El disseny d'una vista heretada és idèntic al de la vista no heretada, només cal afegir el següent element:

```
<Field name = "inherit_id" ref = "id_xml_vista_padre" />
```

Si la vista id_xml_vista_pare resideix en un mòdul diferent del que estem dissenyant, hem d'afegir davant el nom del mòdul:

```
<Field name = "inherit_id" ref = "modulo.id_xml_vista_pare" />
```

El motor d'herència, en trobar una vista heretada, processa el contingut de l'element `arch`. Per cada fill d'aquest element que tingui algun atribut, 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:

- Per **establir la localització del nou camp**, es pot utilitzar:
  - Un element `xpath` que indiqui exactament en quina posició ficar el camp. L'estudi d'aquestes expressions no és objecte d'aquest manual. Poden trobar-se molts exemples als diferents mòduls d'Odoo.
  - Un element `field` on el seu atribut `name` coincideixi amb un camp amb aquest nom.
  - Qualsevol altre element, on el seu atribut `name` coincideixi amb un camp amb aquest nom.
- Per **establir la posició una vegada establerta la localització**, es pot utilitzar:
  - `inside` (per defecte): els valors s'afegeixen dins de l'etiqueta.
  - `after`: després de l'etiqueta.
  - `before`: abans de l'etiqueta.
  - `replace`: reemplaçant el contingut de l'etiqueta

Modificació d'un camp: exemples

Per substituir 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>
```

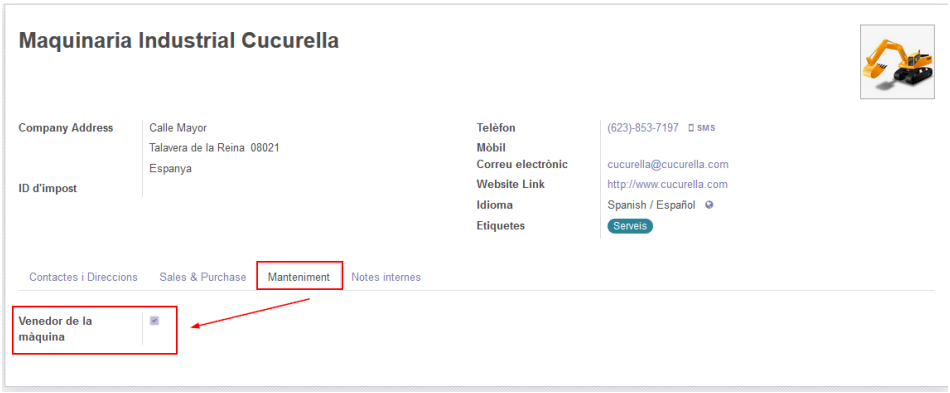
```
<field name="arch" type="xml">
  <field name="camp" position="after">
    <field name="nou_camp" .../>
  </field>
</field>
```

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>
```

Dins el mòdul d'exemple, "manteni", fem servir l'arxiu partner.xml, per a la vista de la classe heretada (es fa afegir un camp de tipus booleà, de nom `machinery_seller`) que hem creat en l'apartat anterior (figura 1.13).

Figura 1.13. Camp machinery_seller creat per herència i afegit a la vista



1.2.3 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** en el disseny de molts mòduls.

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...). Podem veure un exemple a l'arxiu «hr_employee.py» (figura 1.14).

Figura 1.14. Herència al controlador

```
@api.model
def create(self, vals):
    if vals.get('user_id'):
        user = self.env['res.users'].browse(vals['user_id'])
        vals.update(self._sync_user(user))
        vals['name'] = vals.get('name', user.name)
    employee = super(HrEmployeePrivate, self).create(vals)
    url = '/web#%s' % url_encode({
        'action': 'hr.plan_wizard_action',
        'active_id': employee.id,
        'active_model': 'hr.employee',
        'menu_id': self.env.ref('hr.menu_hr_root').id,
    })
    employee._message_log(body=_('<b>Congratulations!</b> May I recommend you to setup an <a href="%s">onboarding plan?</a>') % (url))
    if employee.department_id:
        self.env['mail.channel'].sudo().search([
            ('subscription_department_ids', 'in', employee.department_id.id)
        ])._subscribe_users()
    return employee
```

Com es pot veure, es modifica el mètode "create", afegint-hi comprovacions i funcions. Una vegada fetes, es crida el mètode original amb `super`.

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.

Exemple de l'aplicació de mecanismes d'herència (vídeo)

A continuació, podeu veure un vídeo amb l'aplicació de mecanismes d'herència:

https://player.vimeo.com/video/473782804

1.3 Dades d'exemple

"Data Files"

Aquesta part de la teoria correspon al capítol "Data Files" de la documentació oficial d'Odoo: tinyurl.com/yb3f7sxo.

De vegades, cal fer una **càrrega massiva de dades** en els objectes d'un mòdul. Això pot donar-se per diversos motius:

- Necessitat d'incloure informació imprescindible a un mòdul (per exemple, els tipus d'IVA d'un país quan instal·les el mòdul de comptabilitat regional).
- Càrrega de dades de demostració.

Aquesta càrrega no s'ha de fer mai actuant directament sobre la base de dades, ja que podria conduir al mal funcionament del sistema. Odoo proporciona mecanismes per a la càrrega de dades des de fitxers CSV (només permet la càrrega de recursos, registres, d'un mateix objecte, taula) o XML (permet la introducció de dades que afecten diversos objectes, taules):

- El nom del **fitxer CSV** ha de coincidir amb el nom de l'objecte (taula on s'afegiran o s'actualitzaran els recursos (registres). La primera línia ha de contenir, obligatòriament, el nom dels camps de cada columna, i les següents línies han de contenir les dades dels recursos que s'han d'introduir.
- Els **arxius XML** segueixen una plantilla com la següent, en la qual cada element rècord conté dades d'un recurs i el model al qual pertany, i pot incorporar dades de recursos de diferents models.

```
<? Xml version = "1.0" encoding = "UTF-8"?>
<Odoo>
    <Data noupdate = "1">
         <Rècord id = "idRecurso" model = "nombreObjeto">
            <Field name = "camp1"> valor </ field>
            <Field name = "camp2"> valor </ field>
             ...
        </ Record>
        <Rècord id = "idRecurso" model = "nombreObjeto">
            <Field name = "camp1"> valor </ field>
            <Field name = "camp2"> valor </ field>
             ...
        </ Record>
         ...
    </ Data>
</ Odoo>
```

- 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 del mòdul, és un identificador XML per al recurs. Pot fer-se referència a aquest `id` des del mateix mòdul via el seu nom o des de qualsevol altre mòdul via la sintaxi nombreMódulo.identificador.

- Els **arxius amb dades** (siguin CSV o XML) s'han d'indicar a l'arxiu __manifest__.py del mòdul. Diferenciem entre arxius amb dades imprescindibles del mòdul (es carreguen a la secció "data"), i arxius de demostració (es carreguen a la secció "demo"); com a la figura 1.15.

FIGURA 1.15. Càrrega de dades a l'arxiu __manifest__.py

```
# always loaded
'data': [
    'security/security.xml',
    'security/ir.model.access.csv',
    'views/partner.xml',
    'views/manteni.xml',
    'views/templates.xml',
    'report/manteni_report.xml',
    'data/data.xml',
],
# only loaded in demonstration mode
'demo': [
    'demo/demo.xml',
],
```

A continuació, es mostra la càrrega de dades del mòdul "manteni".

- La sintaxi mostrada per fer referència a un objecte ja creat (ref="dep_maintenance") és pròpia d'Odoo. Per a més informació podeu consultar: tinyurl.com/ydaq4sdq.

```
<odoo>
    <data>

        <!--Department-->

      <record id="dep_maintenance" model="hr.department">
          <field name="name">Maintenance</field>
      </record>

        <!--Jobs-->

      <record id="job_maintenance_manager" model="hr.job">
          <field name="name">Maintenance Manager</field>
          <field name="department_id" ref="dep_maintenance"/>
      </record>

      <record id="job_maintenance_officer" model="hr.job">
          <field name="name">Maintenance Officer</field>
          <field name="department_id" ref="dep_maintenance"/>
      </record>

    </data>
</odoo>
```

- La sintaxi mostrada per afegir objectes a un camp `many2one` (eval="[(6,0, ref('torno1'))]") és pròpia d'Odoo, i serveix per esborrar els registres existents a un camp `one2many` o `many2many` i posar-hi els indicats. Per a més informació podeu consultar: tinyurl.com/yatfpcdg.

```
<record id="instruction0" model="manteni.program.instruction">
            <field name="name">Comprobació de continuitat</field>
            <field name="description">Revisar la continuitat amb un tester</
                field>
            <field name="program_id" ref="program0"/>
        </record>

        <record id="workorder1" model="manteni.workorder">
            <field name="date_begin" eval="datetime.today()"/>
            <field name="description">Manteniment ordinari elèctric</field>
            <field name="employee_id" ref="employee_paco"/>
            <field name="employee_user_id" ref="user_paco" />
            <field name="machine_ids" eval="[(6, 0, [ref('torno1'), ref('torno2
                '), ref('torno3'), ref('torno4'), ref('torno5'), ref('torno6')
                ])]"/>
            <field name="program_id" ref="program0"/>
            <field name="state">opened</field>
            <field name="name">Preventiu elèctric tornos</field>
            <field name="type">preventive</field>
        </record>
```

Exemple de la creació de dades d'exemple (vídeo)

A continuació, podeu veure un vídeo amb la creació de dades d'exemple per a un mòdul.

https://player.vimeo.com/video/473782864

1.4 Traducció del mòdul

"Translating Modules"

Aquesta part de la teoria correspon al capítol "Translating Modules" de la documentació oficial d'Odoo: tinyurl.com/y87ebqpt.

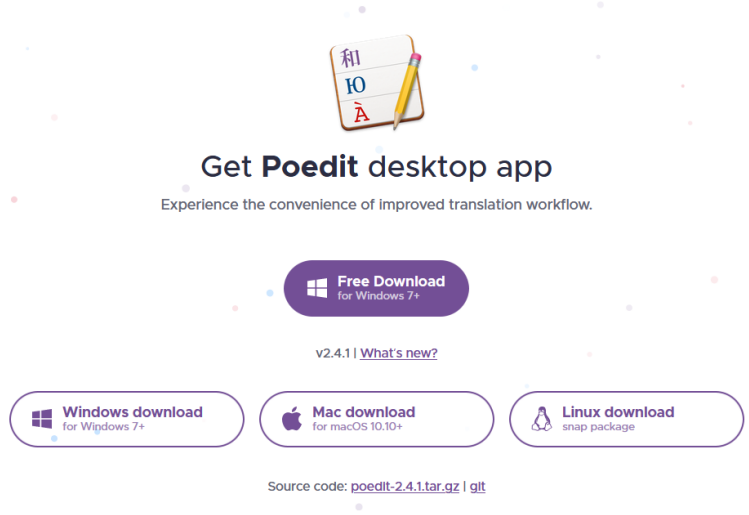
Les bones pràctiques d'Odoo indiquen que els mòduls sempre han de desenvolupar-se **en anglès**, encara que no estigui destinat a treballar-se en aquesta llengua. Una vegada s'ha finalitzat amb el desenvolupament, aquest mòdul es pot traduir de manera molt senzilla a múltiples idiomes. La traducció d'un mòdul pot portar-se a terme de diferents maneres, en aquests materials hem fet servir el format **.po**, i el *software* gratuït **Poedit**.

1.4.1 Poedit

Poedit (abans poEdit) és un editor lliure, obert i multiplataforma de catàlegs de Gettext utilitzats en el procés de localització. Està construït amb les wxWidgets i sota la llicència de la MIT. Té una versió gratuïta i una de pagament, que incorpora millores com ara la traducció automàtica del text.

Per **descarregar Poedit** només s'ha d'anar al seu web oficial. Hi ha versions per a Windows, Mac i Linux (figura 1.16).

FIGURA 1.16. Descàrrega de Poedit



1.4.2 Preparació dels arxius de la traducció

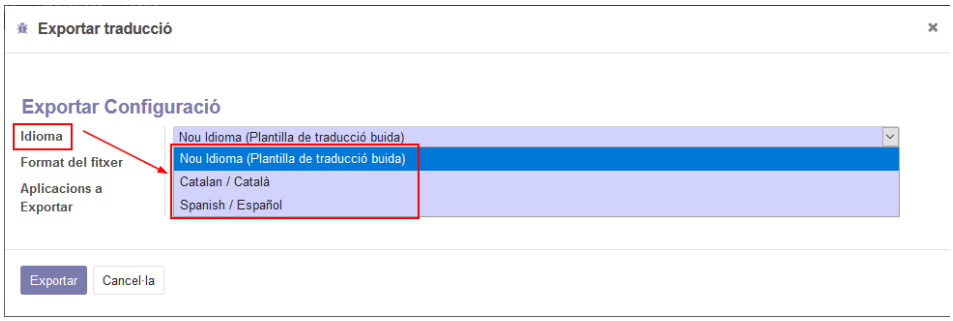
Per començar la traducció és necessari generar els arxius .po que s'editaran amb Poedit. Hi ha dues possibilitats: o bé crear una plantilla genèrica i partint d'aquesta anar creant els diferents idiomes, o bé descarregar directament un arxiu preparat per a cada idioma, i després editar-los. Tots dos processos són similars.

Per generar aquests arxius s'ha d'anar a la secció "Traduccions/Exportar traducció" del menú de configuració; amb el mode desenvolupador activat (figura 1.17).

FIGURA 1.17. Exportació de plantilles de traducció

El primer pas serà triar l'**idioma d'exportació**. Si se selecciona la primera opció (Nou idioma), es genera una plantilla buida amb extensió .pot. En cas de triar un dels idiomes existents a la instal·lació, es genera un arxiu de traducció .po. Tal com s'ha dit anteriorment, tots dos processos són equivalents (figura 1.18).

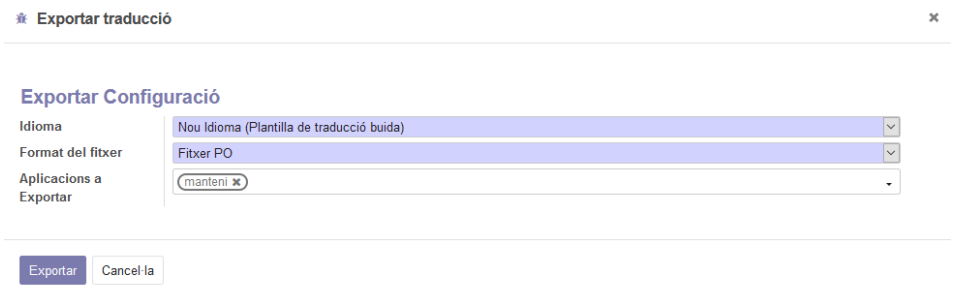
FIGURA 1.18. Elecció de l'idioma a exportar



A continuació, triarem el **format PO**, per a poder ficar-lo a Poedit.

Finalment, s'ha d'**afegir el mòdul** o mòduls dels quals es vol generar l'arxiu. Si es deixa aquest últim apartat en blanc, es generarà un arxiu per a traduir tots els mòduls d'Odoo instal·lats (figura 1.19).

FIGURA 1.19. Elecció del tipus d'arxiu a generar i el mòdul a traduir.

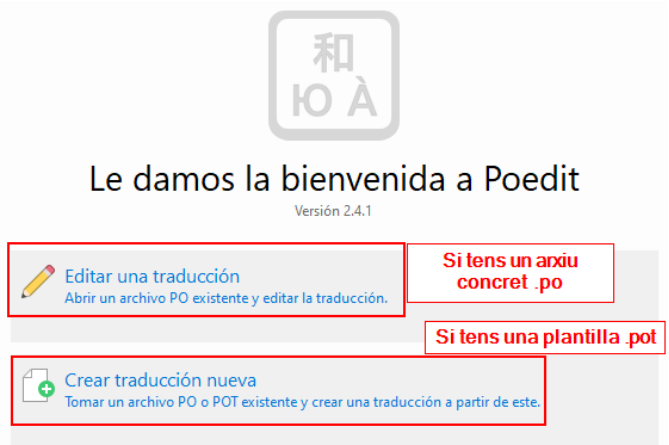


Es generarà l'arxiu, que descarregarem al disc dur, i incorporarem a Poedit.

1.4.3 Traducció del mòdul a Poedit

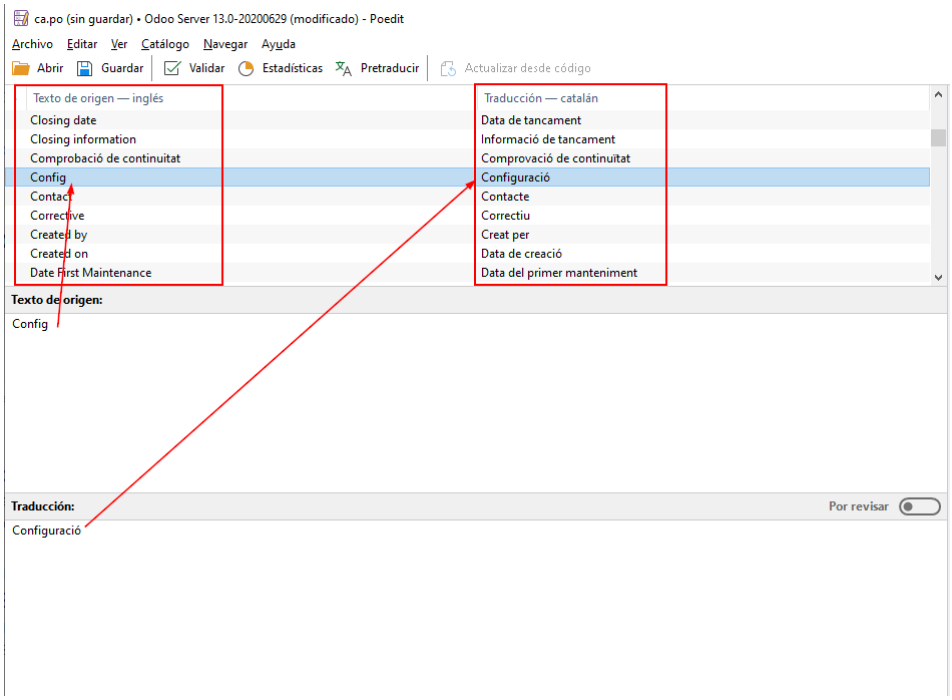
A la pantalla inicial de Poedit apareix un menú per a carregar un arxiu de traducció .po o bé una plantilla .pot. Una vegada es tria l'opció i l'arxiu amb el qual treballar (s'ha descarregat anteriorment), es pot començar la traducció (figura 1.20).

FIGURA 1.20. Pantalla inicial de Poedit



A partir d'aquest moment s'ha d'anar traduint cadascun dels termes proposats per Odoo i que estan dins l'arxiu a l'idioma triat (figura 1.21).

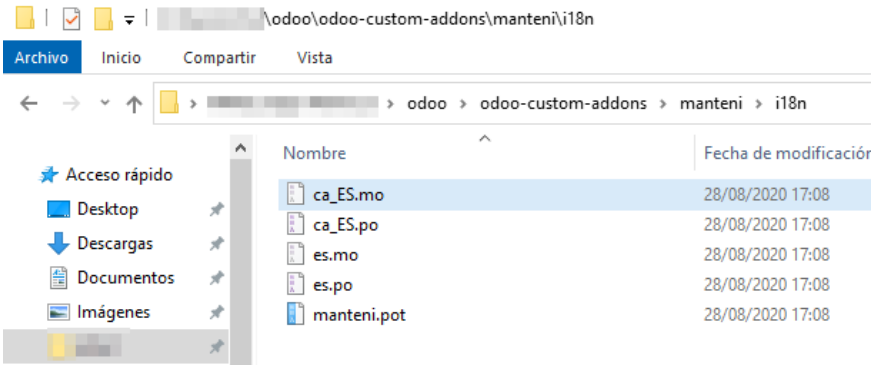
FIGURA 1.21. Traducció dels termes



1.4.4 Incorporació de la traducció a Odoo

Una vegada traduïts tots els termes i guardat l'arxiu .po, només queda incorporar-lo a Odoo. S'haurà de guardar aquest arxiu dins el mòdul, a una carpeta amb el nom **i18n**. Una vegada reiniciat el servidor i actualitzat el mòdul, podreu comprovar que la traducció s'aplica sense problemes (figura 1.22).

FIGURA 1.22. Carpeta i18n del mòdul manteni



Exemple de la traducció d'un mòdul

A continuació podeu veure un vídeo amb la traducció d'un mòdul.

https://player.vimeo.com/video/473782749

2. 'Reporting' & BI

Com ja haureu pogut comprovar, les solucions informàtiques per a gestió comercial incorporen un conjunt predefinit d'**informes i quadres de comandament**, i fins i tot la possibilitat de generar-ne de nous.

Hi ha, no obstant això, organitzacions que requereixen generar una gran quantitat d'informes i quadres de comandament **diferents dels oferts pel sistema**. Per això existeixen al mercat un gran nombre d'eines adequades per a generar aquest tipus d'informació segons les necessitats de l'empresa.

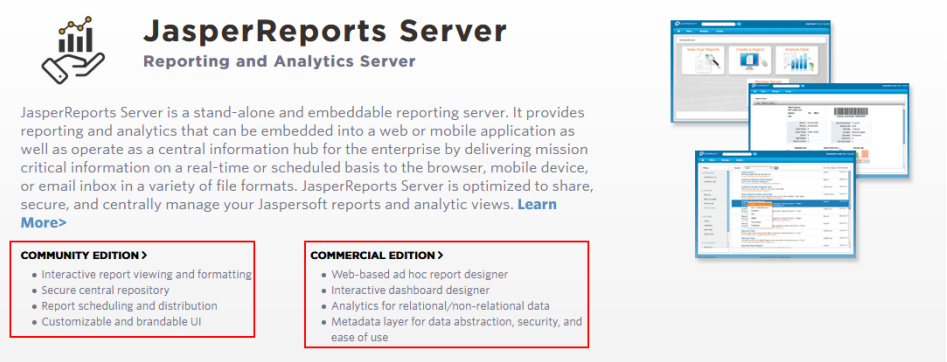
Així mateix, també se'ns pot plantejar la necessitat d'haver d'accedir a altres orígens de dades externes, per exemple en el cas de tenir un ERP i un CRM connectats, però en diferents bases de dades. Aquesta possibilitat no la sol oferir el propi sistema de gestió empresarial i, per tant, interessa conèixer les eines BI externes al sistema.

2.1 'Reporting' extern: JasperReports i JasperStudio

JasperReports és un motor de generació d'informes molt potent. La versió *community* té una llicència LGPL (programari lliure). Està escrit en java i té les característiques següents:

- Facilita connectors per a un gran nombre d'orígens de dades.
- Permet dirigir els informes cap a pantalla, impressora o arxiu en molts formats: (PDF, RTF, XML, XLS, CSV, HTML, XHTML, text, DOCX o OpenOffice).
- Permet introduir gràfics de múltiples formats en els informes.
- Permet incloure altres informes (subinformes) en els informes.
- Permet incorporar filtres (paràmetres en els informes), de manera que en el moment de l'execució l'usuari pot introduir els valors amb els quals desitja que es confeccioni l'informe.
- Permet ser incrustat en aplicacions desenvolupades en diferents llenguatges, és a dir, des del programa es pot invocar JasperReports per generar informes. Els llenguatges Java i PHP són exemples d'aquesta possibilitat (figura 2.1).

FIGURA 2.1. JasperReports



La versió de JasperServer amb la qual treballarem és la *community*, i per tant, gratuïta. La versió professional incorpora, entre d'altres, la capacitat de generar quadres de comandament (*dashboards*), molt útils per a la presa de decisions.

El motor JasperReports genera l'informe a partir d'un document que conté el disseny de l'informe (documents XML d'extensió .jrxml) i d'un origen de dades al qual es connecta per anar a buscar les dades que bolcarà l'informe. Al mercat hi ha eines que ens permeten dissenyar informes sense ser coneixedors del llenguatge JRXML, normalment usant interfícies gràfiques. Una d'aquestes eines és JasperStudio, de Jaspersoft Corporation.

JasperStudio, *front-end* gràfic de codi obert basat en Eclipse, facilita una interfície gràfica que permet:

- Dissenyar informes sofisticats que continguin gràfics, imatges, subinformes i molt més, en format .jrxml.
- Accedir als mateixos orígens de dades que facilita el motor JasperReports (via JDBC, TableModels, JavaBeans, XML, Hibernate, CSV i fonts personalitzables).
- Generar l'informe (incorpora el motor JasperReports) i publicar el resultat per pantalla, impressora o en els mateixos formats que facilita JasperReports (PDF, RTF, XML, XLS, CSV, HTML, XHTML, text, DOCX o OpenOffice).

Els passos per treballar amb JasperReports són els següents:

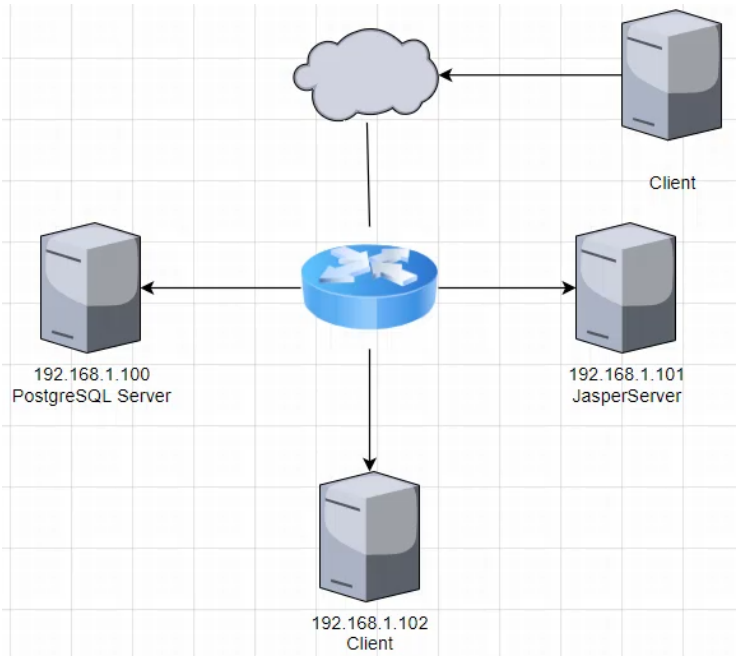
1. Instal·lar el servidor JasperReports Server a la màquina que es cregui convenient. Pot estar a la mateixa màquina on hi ha la base de dades, encara que no és necessari si totes dues màquines estan en xarxa. També pot fer-se servir un contenidor Docker.
2. Instal·lar JasperReports en les màquines dels usuaris que hagin de dissenyar informes.
3. En el servidor d'informes, crear un esquema de seguretat (organitzacions, rols i usuaris) que correspongui i definir els accessos als orígens de dades des d'on s'alimenten els informes.
4. Dissenyar els informes de l'organització amb JasperResports i pujar-los al servidor, assignant els permisos d'accés que corresponguin.
5. En cas de la versió 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 JasperServer i JasperStudio (vídeo)

El sistema amb el qual es proposa treballar és el següent (figura 2.2):

- Un servidor JasperReports ubicat a una màquina diferent a la màquina on hi ha Odoo.
- Un servidor de base de dades MariaDB o MySQL. En aquest cas s'ubicarà a la mateixa màquina que el servidor JasperReports, però no és obligatori.
- Un servidor PostgreSQL amb la base de dades Odoo.
- Un client que té instal·lat JasperStudio, i que s'encarrega de generar els servidors, i pujar-los al servidor JasperReports.
- Un cop funciona el sistema, qualsevol client (tant intern a la xarxa com extern), podria accedir-hi i generar informes.

Figura 2.2. Sistema proposat



Podeu veure la instal·lació i configuració de tot el sistema proposat en el següent vídeo.

https://player.vimeo.com/video/473785895

2.1.2 Creació d'informes amb JasperStudio i generació des de JasperServer (vídeo)

Una vegada tot el sistema està instal·lat, el procés de generació d'informes serà el següent:

- Des del client es genera la consulta a la base de dades que rescati tota la informació necessària.
- S'introdueix la consulta a JasperStudio, que valida contra el servidor de base de dades PostgreSQL que la consulta és correcta.
- Es genera l'informe.
- Es porta a terme una connexió amb la base de dades per construir l'informe amb dades reals.
- Es puja l'informe al servidor JasperServer.

A partir d'aquest moment qualsevol client que accedeixi al servidor JasperServer serà capaç de generar l'informe només fent clic a sobre. Fins i tot es poden crear paràmetres que facin de filtre a les dades de l'informe.

Es pot veure tot aquest procés en el següent vídeo.

https://player.vimeo.com/video/473786004

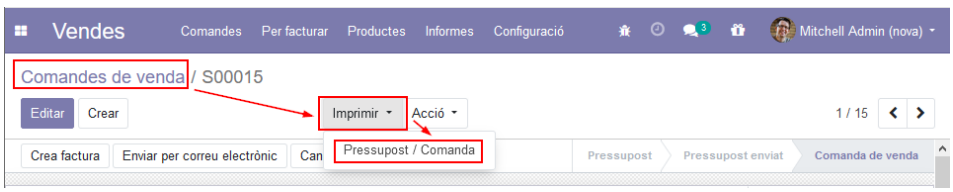
2.2 Generació d'informes natius d'Odoo

"QWeb reports"

Aquesta part de la teoria correspon al capítol "QWeb reports" de la documentació oficial d'Odoo. A la versió més actual a la redacció d'aquesta documentació l'enllaç és tinyurl.com/ydb9w6kt.

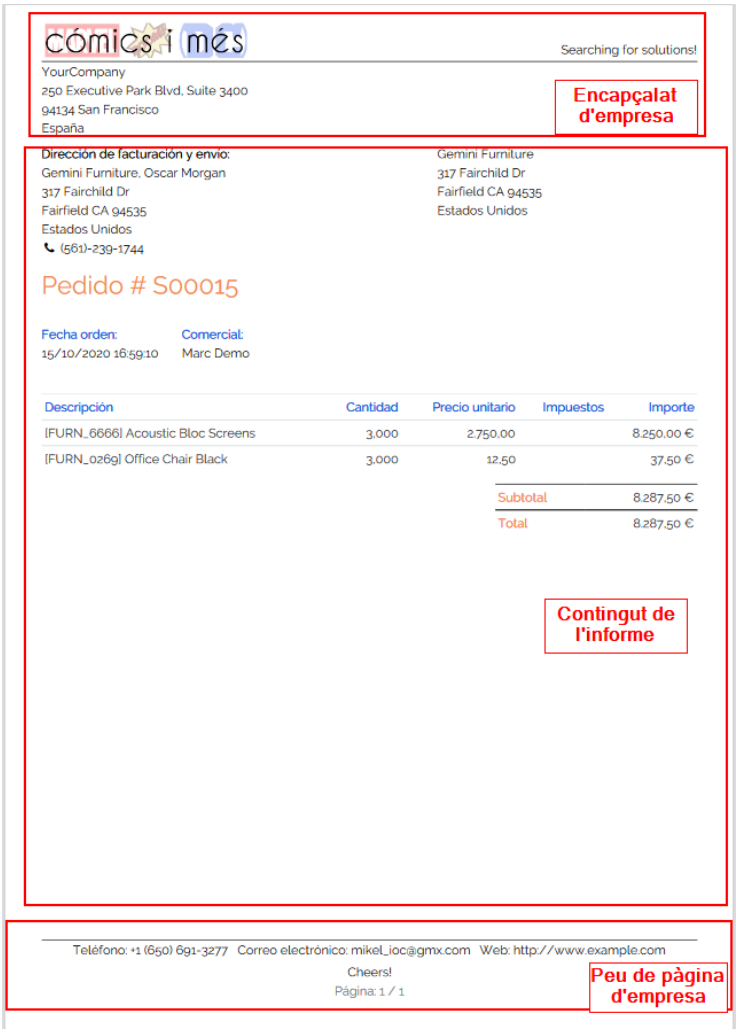
Aquesta opció, que genera els informes directament a Odoo, consisteix a generar un web en llenguatge HTML/QWeb, amb format similar a una vista. Una vegada creat, i amb el programa wkhtmltopdf, pot convertir-se l'HTML en PDF. Els informes creats d'aquesta manera poden imprimir-se a la vista del model al qual pertanyen. Sempre que hi hagi un informe disponible es pot veure un botó per generar-lo (figura 2.3).

FIGURA 2.3. Impressió de l'informe associat a l'objecte sale.order



Una vegada es fa clic a “Imprimir” apareix l’opció de guardar o visualitzar l’arxiu pdf generat. Aquest informe conté el contingut propi del document, a més a més de l’encapçalament i peu de pàgina de l’empresa (figura 2.4).

FIGURA 2.4. Contingut d’un informe QWeb d’Odoo.



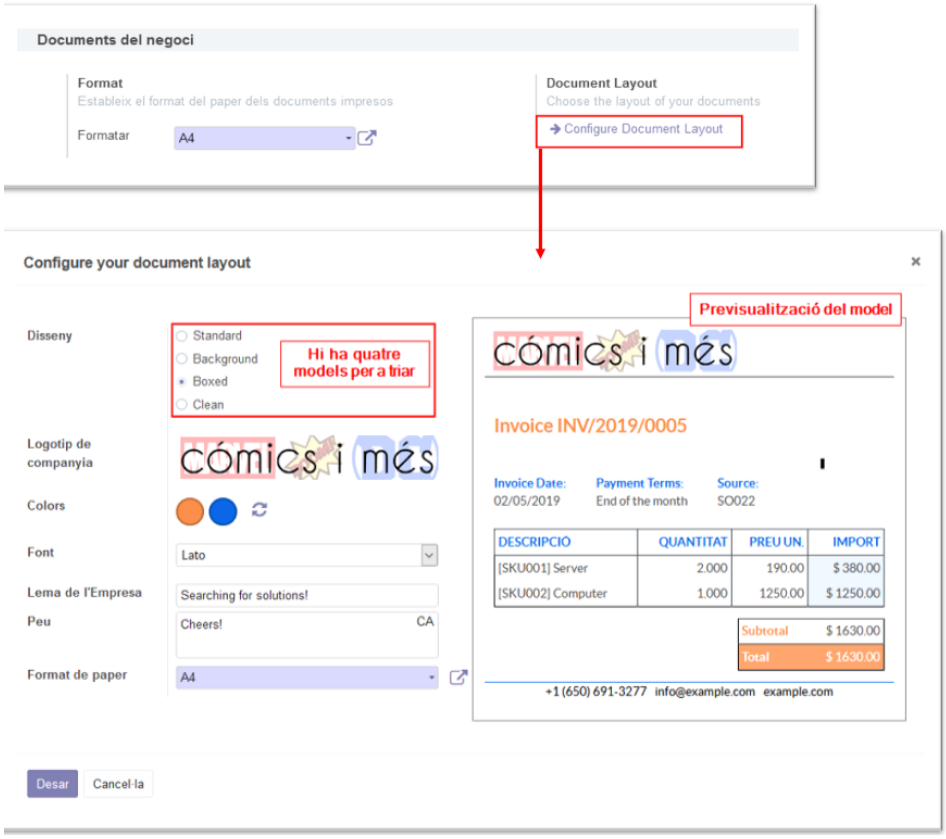
2.2.1 Edició de la plantilla genèrica

Tal com s’ha vist a la figura anterior, tots els informes generats per Odoo contenen un encapçalament i un peu de pàgina genèrics i comuns que incorporen informació sobre l’empresa i la data de generació de l’informe.

Hi ha dues configuracions possibles: bàsica i avançada.

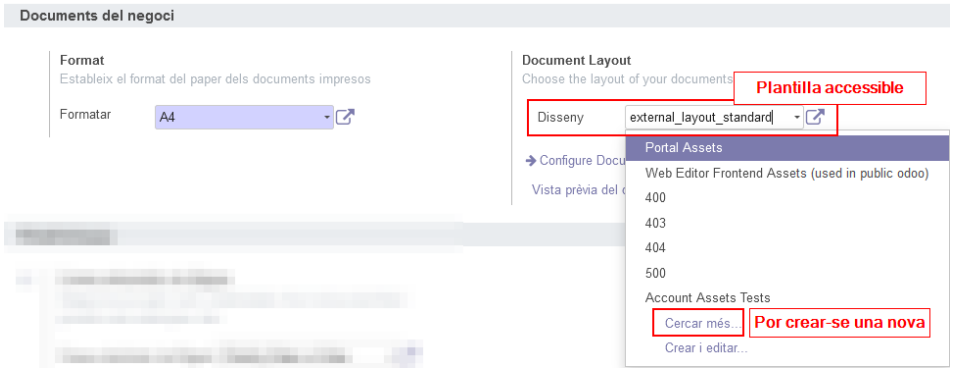
A la configuració bàsica pot triar-se entre quatre models diferents, editar-se el logo, la tipografia i els colors. També pot afegir-se un eslògan i un missatge per al peu de pàgina. Es pot accedir a aquesta configuració des de “Configuració” (figura 2.5).

FIGURA 2.5. Configuració bàsica de la plantilla.



Si es desitja una personalització total de la plantilla, i mitjançant el mode desenvolupador, es pot accedir al codi per a crear una còpia o fer-ne modificacions (figura 2.6).

FIGURA 2.6. Personalització total al mode desenvolupador



2.2.2 Creació d'un informe sobre una classe

Els informes es generen mitjançant un arxiu xml, que conté informació en format QWeb. A continuació es presenta la sintaxi genèrica per a la seva generació.

El format QWeb és objecte de la secció següent, on s'explicarà amb detall.

Crearem una carpeta dins del mòdul anomenada "report", i crearem un arxiu anomenat nombre_modulo_report.xml, on dissenyarem l'informe.

Format de l'arxiu xml

Els informes a Odoo comencen amb l'etiqueta <report>. A continuació hauran d'anar afegint-se els atributs:

- `id`: identificador extern de l'informe.
- `name`: nom de l'informe que apareixerà a Odoo.
- `model`: model del qual s'importarà la informació
- `report_type`: qweb-pdf per a PDF, qweb-html per a HTML.
- `report_name`: nom de l'arxiu PDF resultant.
- `groups`: grups permesos per utilitzar aquest informe

```
<report
    id="account_invoices"
    model="account.invoice"
    string="Invoices"
    name="account.report_invoice"
    report_type="qweb-pdf"
    print_report_name="object._get_report_filename()"
    attachment_use="True"
    attachment="(object.state in ('open','paid')) and
        ('INV'+(object.number or '').replace('/','')+'.pdf')"
/>
```

Seguidament es dissenyarà l'informe amb l'etiqueta , i utilitzant el llenguatge QWeb. A continuació es mostra un exemple d'informe mínim.

```
<template id="report_invoice">
    <t t-call="web.html_container">
        <t t-foreach="docs" t-as="o">
            <t t-call="web.external_layout">
                <div class="page">
                    <h2>Report title</h2>
                    <p>This object's name is <span t-field="o.name"/></p>
                </div>
            </t>
        </t>
    </t>
</template>
```

Com es pot comprovar, el llenguatge QWeb és xml amb etiquetes que fan referència a atributs d'un objecte. A continuació s'explica amb una mica de detall.

2.2.3 El llenguatge Qweb

QWeb és el motor de plantilles principal utilitzat per Odoo i s'utilitza principalment per generar pàgines i fragments HTML. Consisteix principalment en llenguatge XML, afegint-hi unes etiquetes especials, amb el prefix *t-*, que aporten interacció amb Odoo. Com a exemple, amb QWeb pot crear-se un informe on es

"QWeb"

Aquesta part de la teoria correspon al capítol "QWeb" de la documentació oficial d'Odoo. L'enllaç a la versió més actual de la redacció d'aquesta documentació és tinyurl.com/y8z3q3xu.

rescati un camp d'Odoo amb l'etiqueta `t-field`, o presenti informació depenent d'un camp amb l'etiqueta `t-if`. Per exemple, donat aquest codi:

```
<t t_if = "condition">
    <p> Test </ p>
</ t>
```

Si una vegada avaluada la condició "condition" el resultat és vertader (True), a l'informe es mostrarà la línia:

```
<p> Test </ p>
```

A continuació anirem presentant els tipus d'etiquetes qweb. Presentarem exemples de diferents mòduls.

t-field

La fem servir per a introduir camps d'un objecte dins l'informe. Com a exemple, a l'informe que imprimeix les ordres de treball al mòdul "manteni", rescatem, entre d'altres, el camp `name` (figura 2.7).

FIGURA 2.7. t-field al mòdul manteni

```
<odoo>
    <data>
        <report
            id="report_workorder"
            model="manteni.workorder"
            string="Workorder Report"
            name="manteni.report_workorder_view"
            file="manteni.report_workorder"
            report_type="qweb-pdf" />
        <template id="report_workorder_view">
            <t t-call="web.html_container">
            <t t-foreach="docs" t-as="doc">
            <t t-call="web.external_layout">
            <div class="page">
            <div class="container-fluid">
                <h2 class="text-center mt32">
                    Workorder:
                    <span t-field="doc.name"/>
                </h2>
```

t-esc

Amb aquesta etiqueta també es poden rescatar camps d'Odoo, però a més a més permet avaluar expressions Python a dins. Per exemple, al mòdul "sale" permet calcular el descompte aplicat a un producte (figura 2.8).

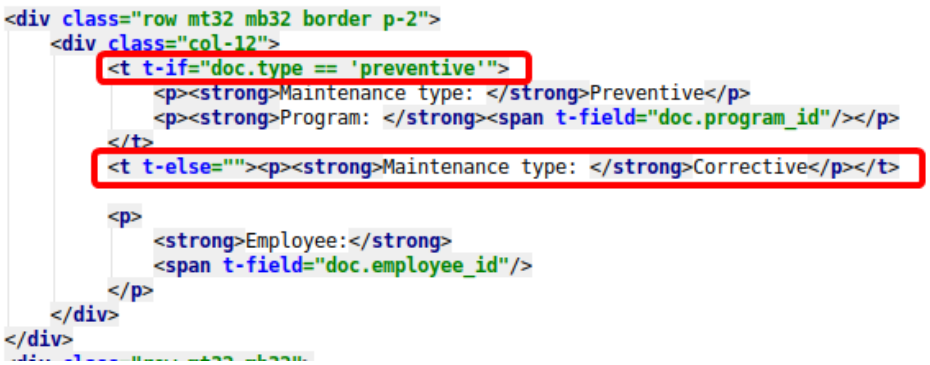
FIGURA 2.8. t-esc al mòdul sale

```
<div t-if="line.discount">
    <t t-esc="(1-line.discount / 100.0) * line.price_unit" t-options='{"widget": "float", "decimal_precision": "Product Price"}'/>
</div>
```

t-if

Serveix per afegir condicions. En cas de complir-se, mostrarà la informació a continuació (figura 2.9)

FIGURA 2.9. t-if al mòdul manteni



Com es pot veure, també existeix `t-else`, per a donar una alternativa. Per a més de dues alternatives utilitzem `t-elif`.

```
<div>
    <p t-if="user.birthday == today()">Happy birthday!</p>
    <p t-elif="user.login == 'root'">Welcome master!</p>
    <p t-else="">Welcome!</p>
</div>
```

"t-foreach"

Qweb té una directiva per a realitzar iteracions: `t-foreach`. Torna una col·lecció de valors per a iterar, que prendrà el nom que s'indica al paràmetre `t-as`. Si, per exemple, posem aquest codi:

```
<t t-foreach="[1, 2, 3]" t-as="i">
    <p><t t-esc="i"/></p>
</t>
```

El resultat seria:

```
<p>1</p>
<p>2</p>
<p>3</p>
```

Als informes del mòdul "manteni" es fa servir `t-foreach`. A continuació presentem un exemple:

```
<table class="table table-striped">
      <thead>
          <tr>
                  <th>Machines</th>
          </tr>
      </thead>
      <tbody>
          <tr t-foreach="doc.machine_ids" t-as="machine">
              <td><span t-field="machine.name"/></td>
          </tr>
      </tbody>
  </table>
```

2.2.4 Llenguatge HTML als informes

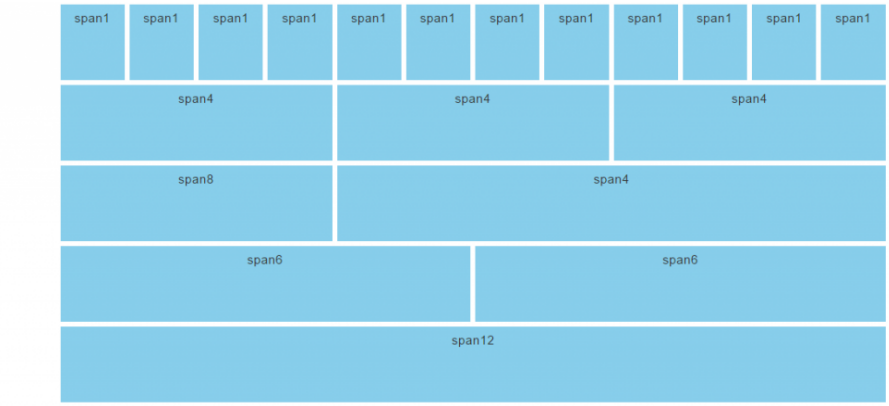
Bootstrap

L'aprenentatge de Bootstrap no és objecte d'aquest material, però pot trobar-se'n informació a: tinyurl.com/yxrvamn8.

Per a crear l'estructura dels informes es fa servir HTML. Odoo pot utilitzar les classes CSS del *framework* Bootstrap, uns dels més utilitzats del món. Odoo 13 utilitza Bootstrap 4.

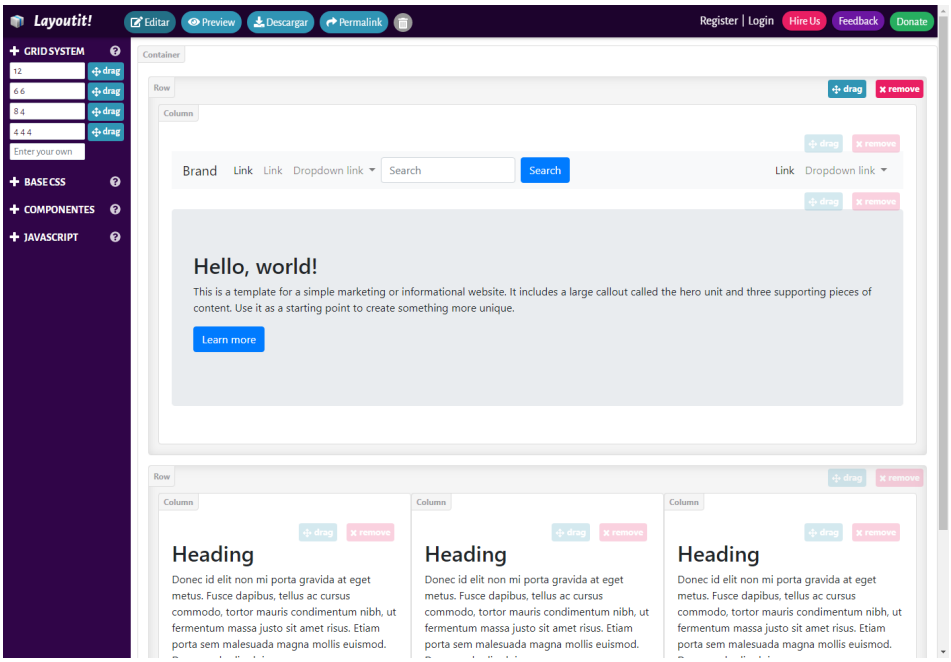
Les pàgines web basades en Bootstrap estan formades per contenidors, dins els quals es posen files i columnes, que poden prendre diferents mides (figura 2.10).

FIGURA 2.10. Grid de Bootstrap



Com que no és objecte d'aquest manual aprendre el *framework* Bootstrap, es recomana fer servir webs orientats al disseny web a Bootstrap de manera gràfica, que després tradueixen el disseny a llenguatge HTML. Un exemple d'aquest tipus de tecnologia seria la web www.layoutit.com (figura 2.11).

FIGURA 2.11. LayoutIt



Càrrega i prova de l'informe

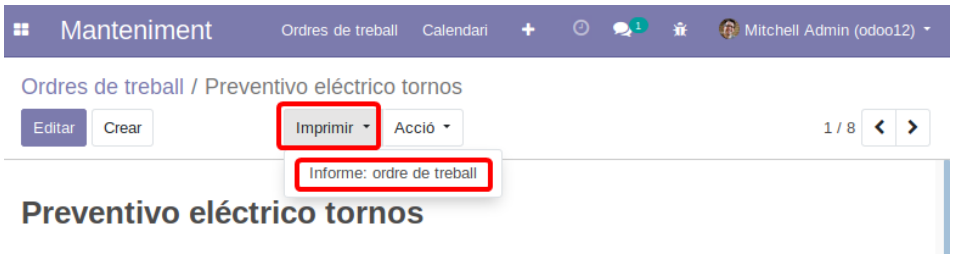
Amb l'informe ja dissenyat s'haurà d'incloure la ruta de l'arxiu a __manifest__.py, per a la seva càrrega quan s'instal·la el mòdul (figura 2.12).

Figura 2.12. Inclusió de la ruta de l'informe a __manifesty__.py

```
# always loaded
'data': [
    'security/security.xml',
    'security/ir.model.access.csv',
    'views/partner.xml',
    'views/manteni.xml',
    'views/templates.xml',
    'report/manteni_report.xml',
    'data/data.xml',
],
```

Una vegada fet, s'ha de reiniciar el servidor i actualitzar el mòdul. A continuació, a la vista de la classe corresponent podrà executar-se l'informe (figura 2.13).

Figura 2.13. Execució de l'informe

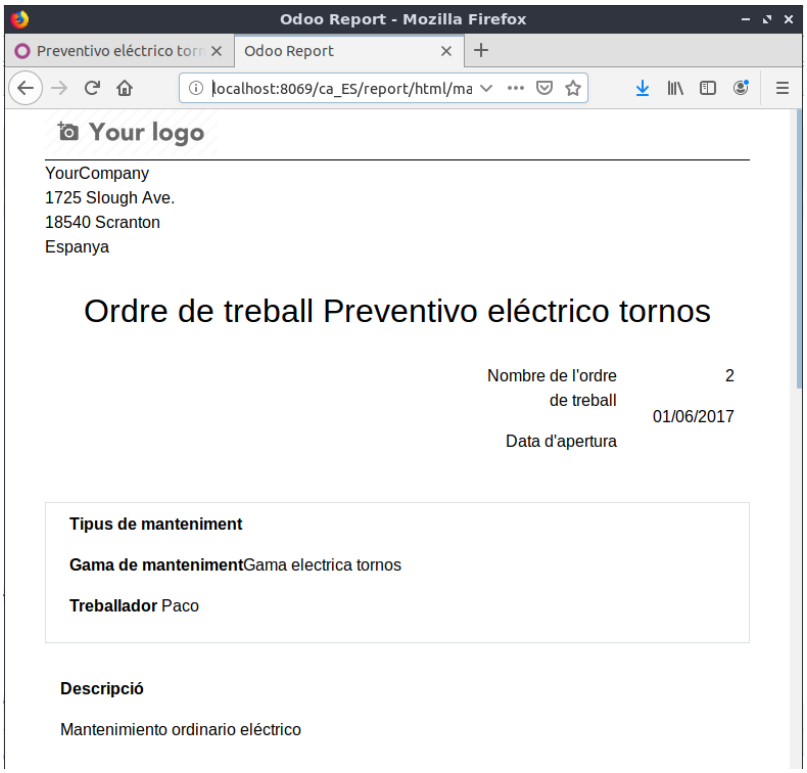


Una manera ràpida de comprovar l'execució de l'informe és fer servir la URL http://<server-address>/report/html/nom_informe/id_registre. Per exemple, si vull generar l'informe de l'ordre de treball amb `id 2` executaria la següent URL:

```
1 http://<server_address>/report/html/manteni.report_workorder_view/2
```

I aquest és el resultat (figura 2.14):

FIGURA 2.14. Prova de l'informe generada per pantalla



2.2.5 Exemple de la creació d'un informe a Odoo (vídeo)

A continuació podeu veure un vídeo amb la creació d'un informe a Odoo mitjançant el llenguatge QWeb.

https://player.vimeo.com/video/473785948