OpenERP

Overview, Introduction to Administration & Development

– EFREI –
– ESTIA –
Guillaume Rivière
Last update: March 2018

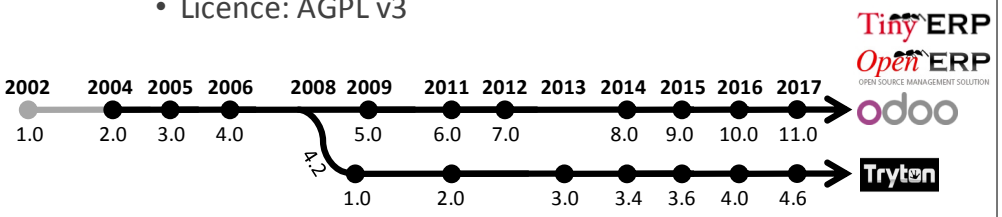
OUTLINE

WARNING !!!

- You are going to learn OpenERP
  - But keep in mind to select an ERP **only** because it is the best adapted to the case of your company

History of OpenERP

- Initiated by Fabien Pinckaers
  - in 2002, in Belgium
  - Written in Python
  - Database is PostgreSQL
  - Open Source *(but not free software, according to FSF)*
    - Licence: AGPL v3



OpenERP today

- Used for teaching since 2008
  - The CERTA network teach accounting to BTS
  - CERPEG
- Arround the world, we find dozens of **Customers**, **Vendors** and **Partners**
- Still supported Long Term Stable versions (LTS)
  - Version 7.0 released in December 2012
  - Version 8.0 released in September 2014
  - Version 9.0 released in October2015
    - Enterprise (20€/user/month) & Community (90%) Editions

Hardware needed

- 5 employees: 2 CPU 2GB RAM
- 20 employees: 4 CPU 8GB RAM
- 90 employees: 2x 8 CPU 32GB RAM
  - Split application and database server
- 250+ employees: 2x 12 CPU 128 GB RAM
  - Load balancing

Vendors / Consultants

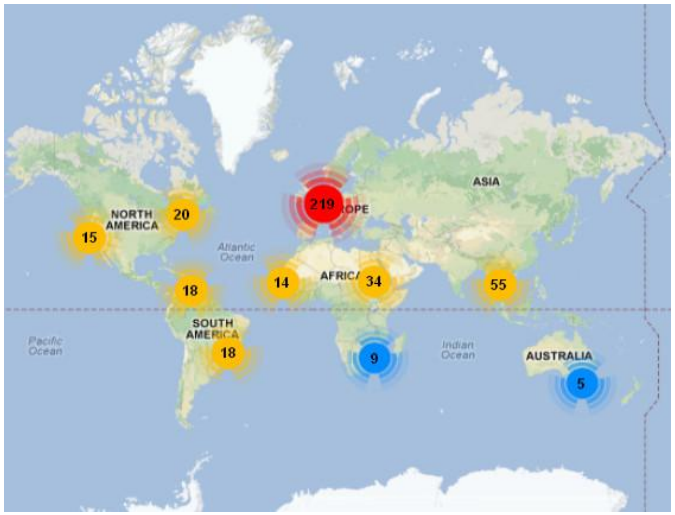
- Smile
- Julius
- Business & Decision
- Aunéor Conseil
- Anybox
- Camptocamp
- ZAD solutions
- Audaxis
- *and many many others…*

Partners

- Gold / Silver / Ready / Training Center
  - Pay annual fees

Partners' world map

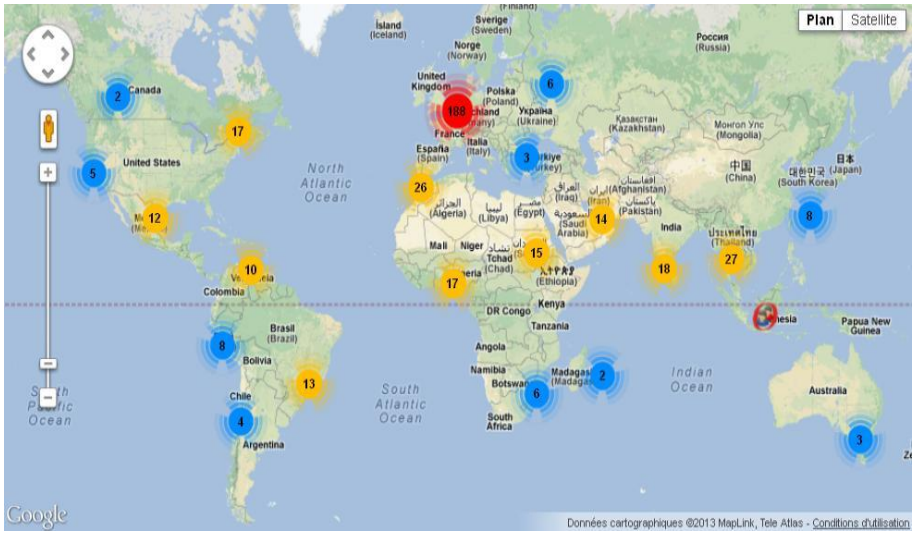
Western Europe	219
Africa	57
Asia	55
North America	35
South America	36
Australia	5



1 partner
2 → 9
10 → 99
≥ 100

(http://v6.openerp.com/partner-worldmap March 2013)

Partners worldwide



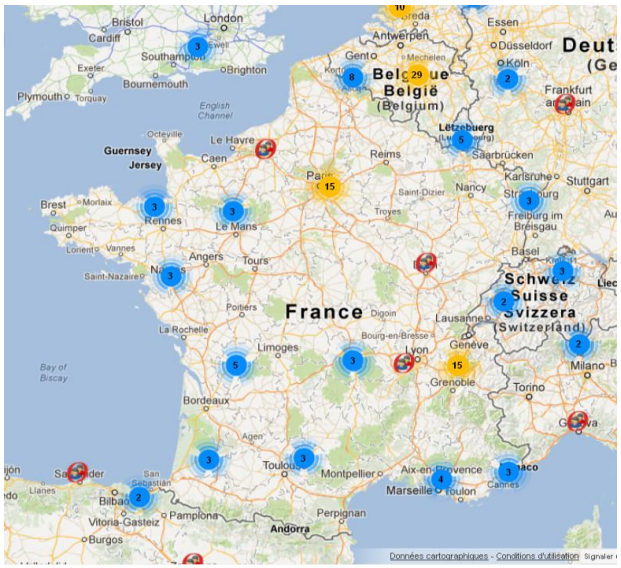
(http://v6.openerp.com/partner-worldmap March 2013)

Partners in Western Europe

(http://v6.openerp.com/partner-worldmap March 2013)

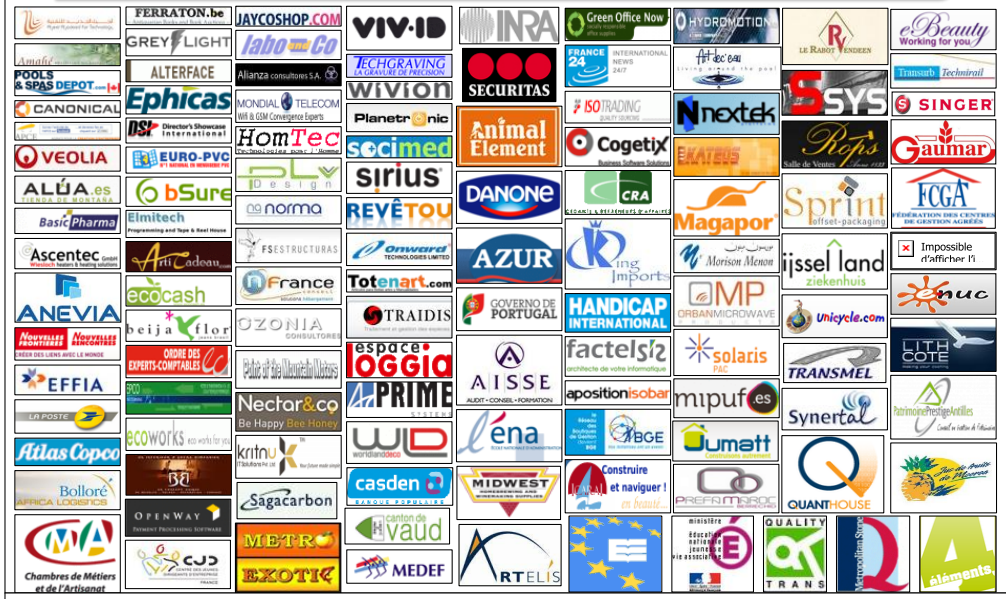
Partners in France

1 partner
2 → 9
10 → 99
≥ 100



(March 2013)

Customer references



OUTLINE

1. Overview
   - Birth and growth
   - Customers, Partners, Vendors
2. Administration
   - Versions, Architecture
   - Main modules
3. Development
   - Module programming
   - Webservice programming
4. Documentation

Timeline of versions

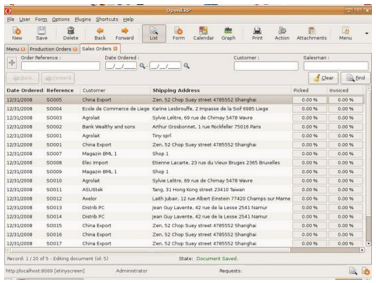
Tiny SPRL (May 2002)
OpenERP SA (2005)
Renamed (April 2008)
Tryton 1.0 (Nov. 2008)
Tryton 2.0 (April 2011)
Tryton 3.0 (Oct. 2013)
Tryton 4.0 (May 2016)

- **TinyERP 1.0** : 06/07/2004 (3 companies)
- **TinyERP 2.0** : 25/03/2005
- **TinyERP 3.0** : 02/09/2005
- **TinyERP 4.0** : 02/12/2006
- **TinyERP 4.1** : 13/06/2007 (web server)
- **TinyERP 4.2** : 01/10/2007
- **OpenERP 5.0** : 08/02/2009
- **OpenERP 6.0** : 20/01/2011
- **OpenERP 6.1** : 23/02/2012 (new web gui)
- **OpenERP 7.0** : 22/12/2012 (full web)
- **Odoo 8.0** : 18/09/2014 (front: e-com, CMS+builder, stocks)
- **Odoo 9.0** : 01/10/2015 (accounting)
- **Odoo 10.0** : 01/10/2016 (production)
- **Odoo 11.0** : 01/10/2017

LTS 9.0 will be supported until LTS 12.0 is released

Elements to run OpenERP

- Client
  - GTK client: until version 6.x
  - Web client: since version 4.1
  - « full web »: since version 7.0
- Server
  - Separated web server: from 4.1 to 6.x
  - Version 7.0: one-piece server
    - v7.0 : Python >= 2.6 (Python 2.7)
    - v6.1 : Python >= 2.6
    - v6.0 : Python >= 2.5



Compatibility

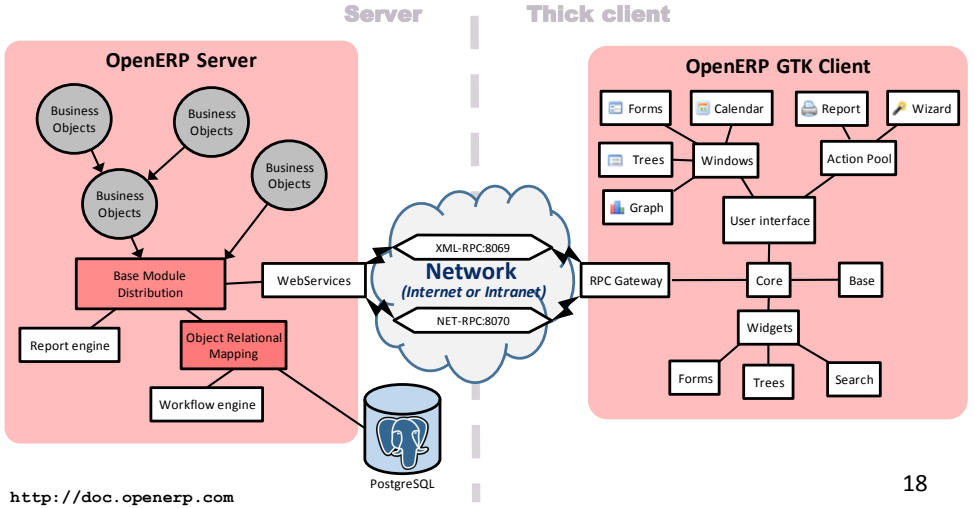
- Server operating system
  - Linux, Windows, Mac OS
- Web browser

	(OpenERP 7)	(Odoo 10)
– **Mozilla Firefox**	(≥ 13)	✓
– Microsoft IE	(≥ 9.0)	-
– Microsoft Edge	-	✓
– Safary	(N.C.)	✓
– Google Chrome	(≥ 22)	✓

- OpenERP Mobile Client (Smartphones, tablets)
  - Application Android, iOS (alternative : myodoo.com)

Technical Architecture

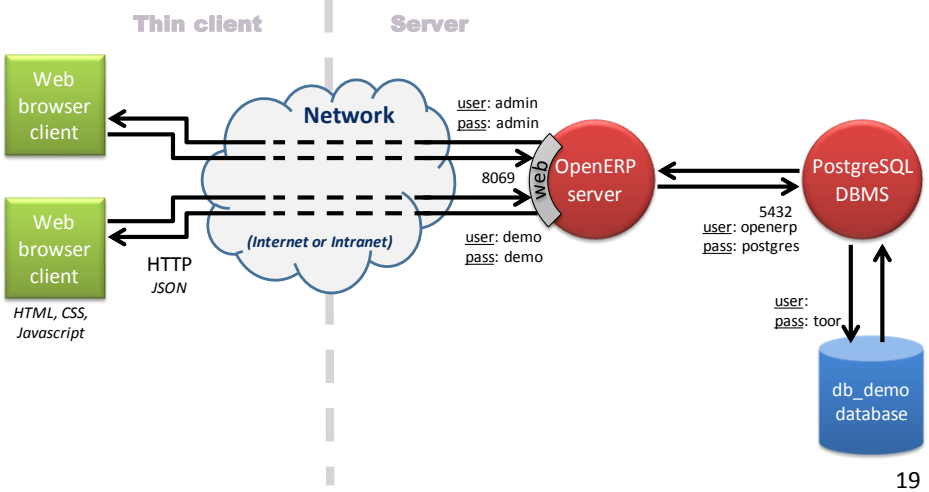
- OpenERP v6 (for GTK client)



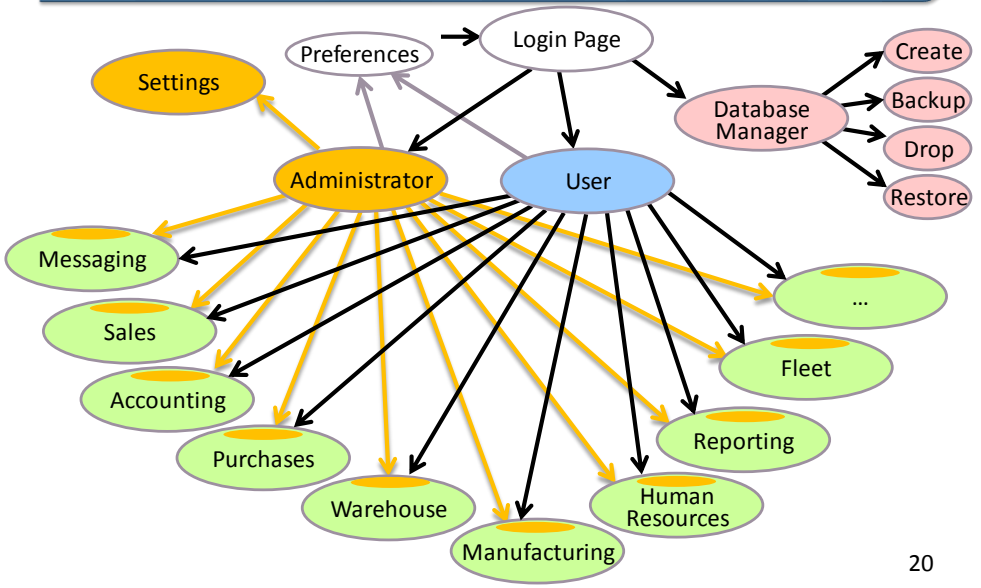
http://doc.openerp.com

Technical Architecture

- OpenERP v7 v8 v9 (web client)



Global navigation

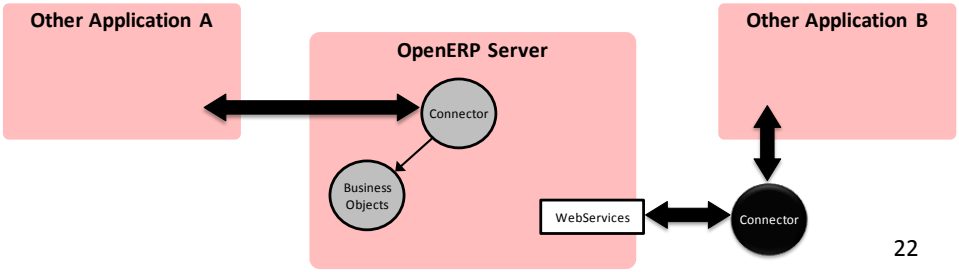


Administration

- Administrator(s) must define
  - The users
  - The access rights
  - The languages
  - The menus
  - The basis data of the company
    - Products, work centers, flows, etc.
  - The modules to install
  - The connections with other software
  - and many other things…

Connectors

- Enterprise Application Integration (EAI)
- Interoperate OpenERP and other programs
  - Webservice (program in Python, Java, PHP, C/C++, …)
  - Business objects (module in Python)
    - `http://openerp-connector.com/`

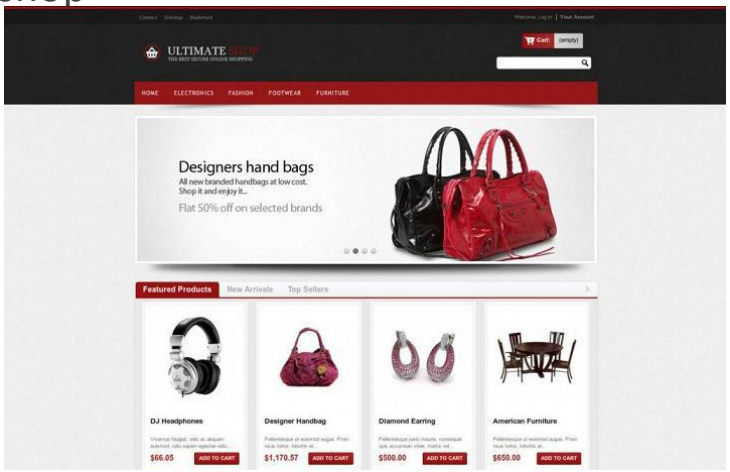


Connectors

- CMS
  - Joomla! (PHP/MySQL)
- E-commerce
  - OSCommerce (PHP/MySQL)
  - Magento (PHP)
  - PrestaShop (PHP/MySQL)
- CMS + eCommerce
  - eZ Publish (PHP/MySQL, PostgreSQL, SQLserver,Oracle)
- Reporting
  - JasperReports (Java)
  - ReportLab (Python)
- Business Intelligence
  - Pentaho (Java)
  - SpagoBI (Java)

E-commerce

- Prestashop

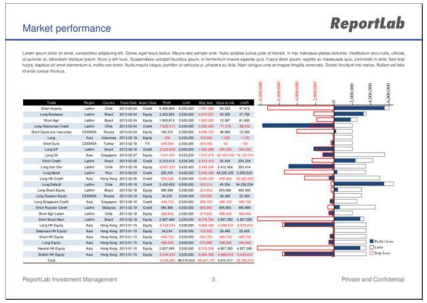


Reporting

- JasperReports

Reporting

- ReportLab
  - Financial reporting and dynamic graphics



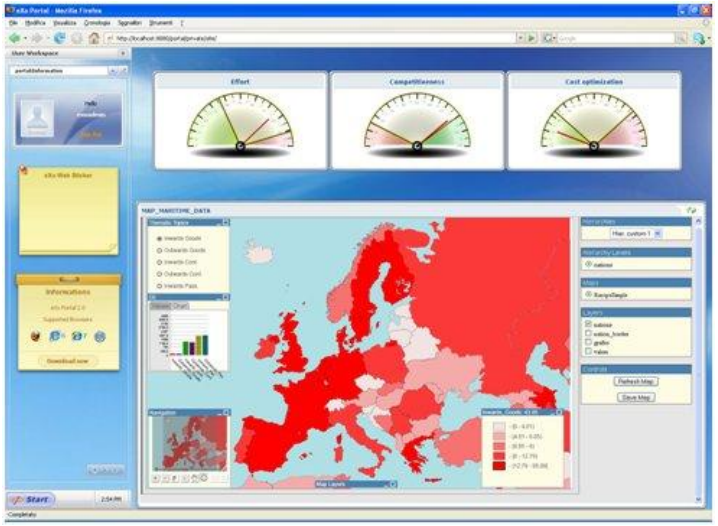
  - Natively used in OpenERP for reporting diagrams

Business Intelligence

- Pentaho

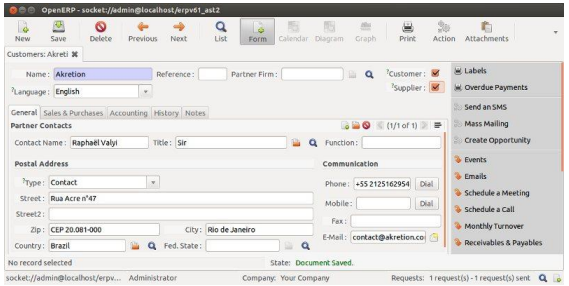
Business Intelligence

- SpagoBI



VOIP

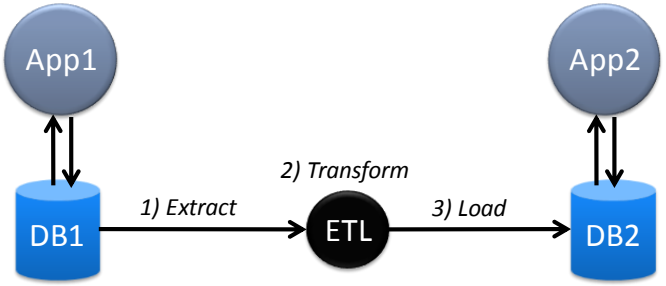
- CRM
  - Routing phone calls
    - Open the customer master record when they are calling
    - Call-center VoIP with Asterisk server
  - Automatically call a customer



Click2dial

ETL

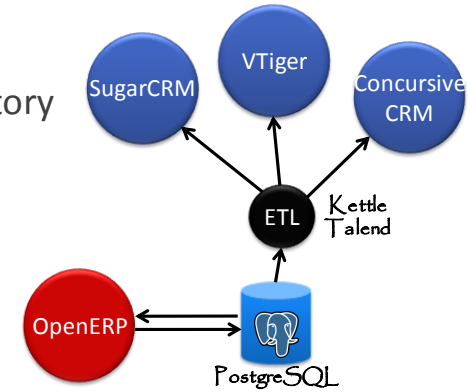
- **E**xtract-**T**ransform-**L**oad



  - Integrate data from multiple databases of different applications

ETL

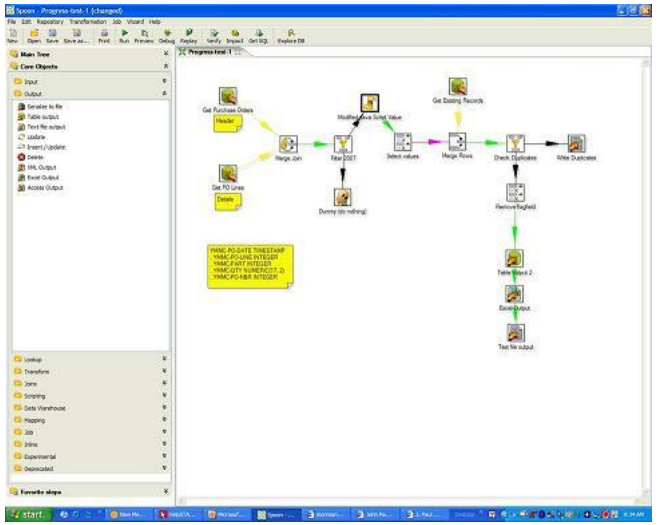
- CRM
  - Targeting the customers
  - Data mining of sales history
    - Weka (Java)
    - SugarCRM (PHP)
    - vTiger (PHP)
    - Concursive CRM (Java)



kettle talend WEKA SUGARCRM vtiger concursive

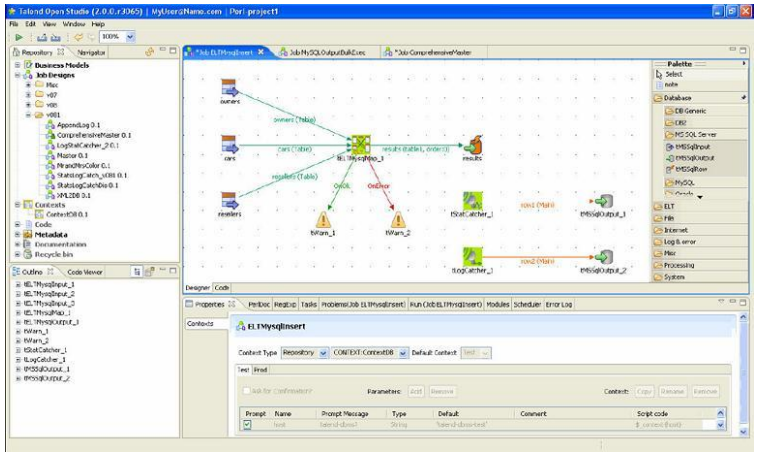
ETL

- Kettle



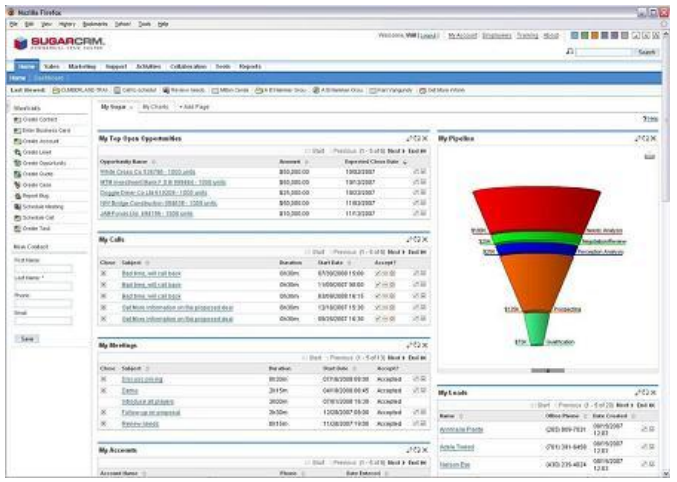
ETL

- Talend



CRM

- SugarCRM
  - B2B
  - B2C

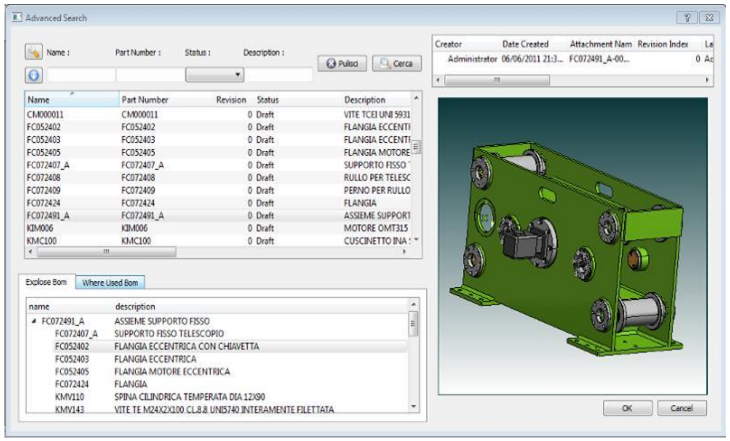


CRM

- vTiger
  - Fork of SugarCRM

PLM / PDM

- OpenERP PLM
  - **OmniaSolutions**



Manage management systems

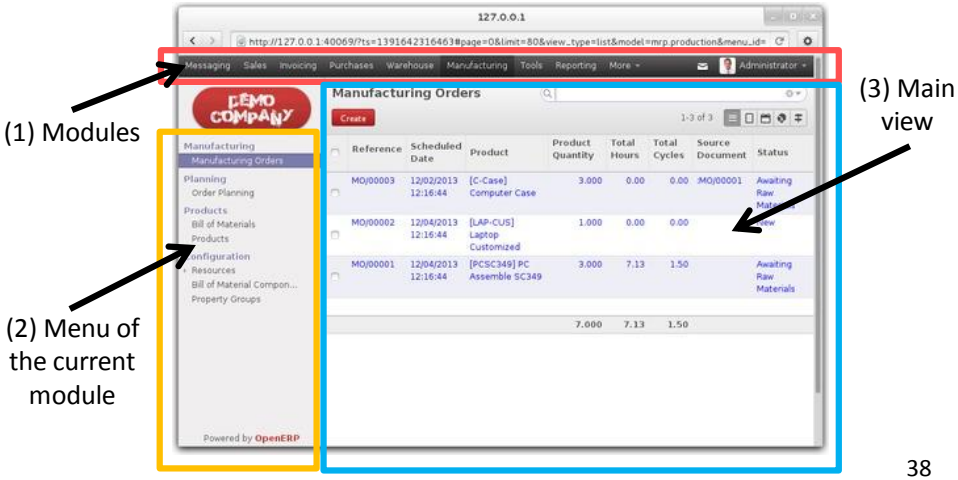
- Certifications
  - Quality (ISO 9001)
  - Environment (ISO 14001)
  - Security (ISO 27001, PCI-DSS)
  - IT services (ISO 20001, ITIL)
  - Health & Safety (OHSAS 18001)
- Extra menus and tabs
  - In several modules

www.pcsol.be

www.savoirfairelinux.com

User Interface

- User interfaces of ERPs are often quite simple



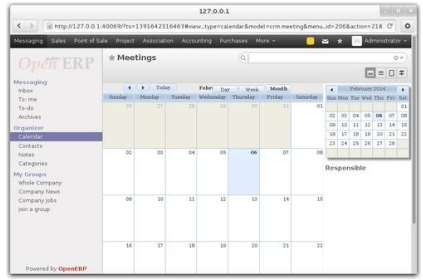
Screenshot of OpenERP v7.0

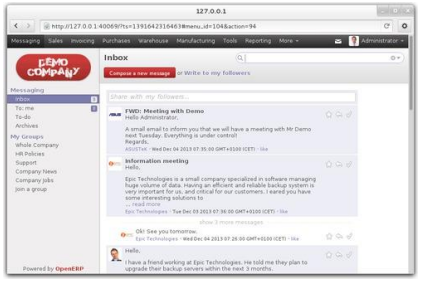
Main modules

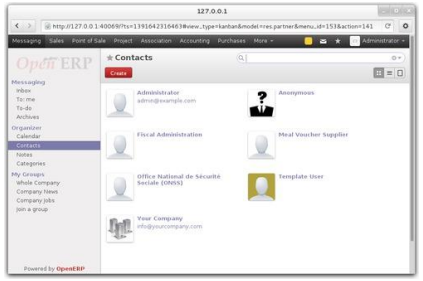
- Messaging
- Sales
- Point of Sales
- Project
- Association
- Accounting
- Purchases
- Warehouse
- Manufacturing
- Marketing
- Human Resources
- Fleet
- Events
- Knowledge
- Tools
- Reporting

Messaging

- Inbox
- Meetings calendar
- Contacts
- …







Screenshots of OpenERP v7.0

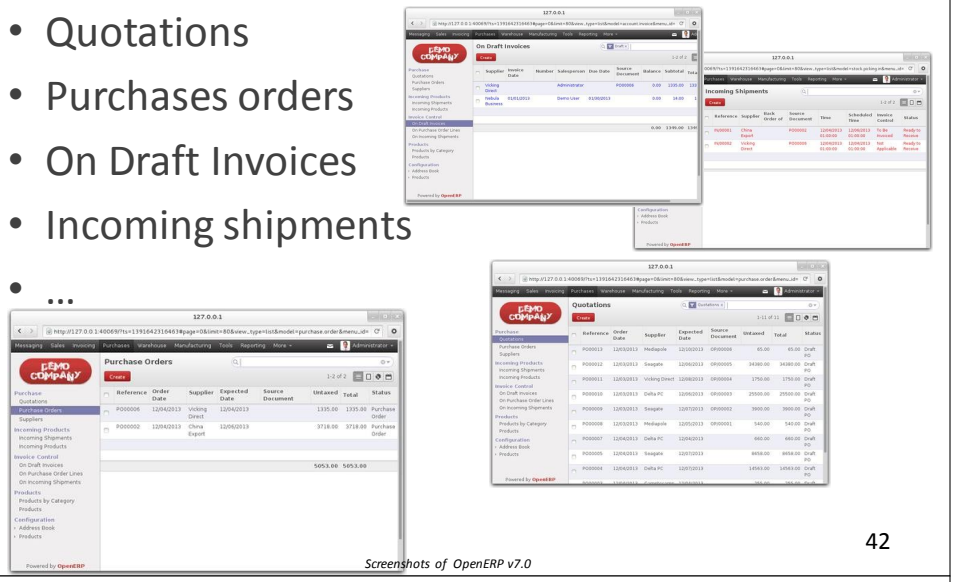
Sales

- Customers
- Products
- …

Screenshots of OpenERP v7.0

Purchases

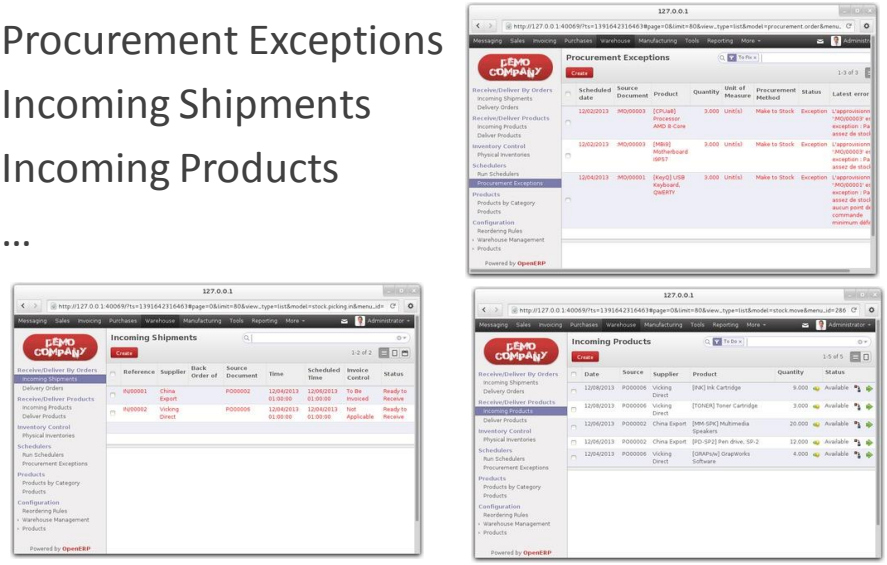
- Quotations
- Purchases orders
- On Draft Invoices
- Incoming shipments
- …



Screenshots of OpenERP v7.0

Warehouse

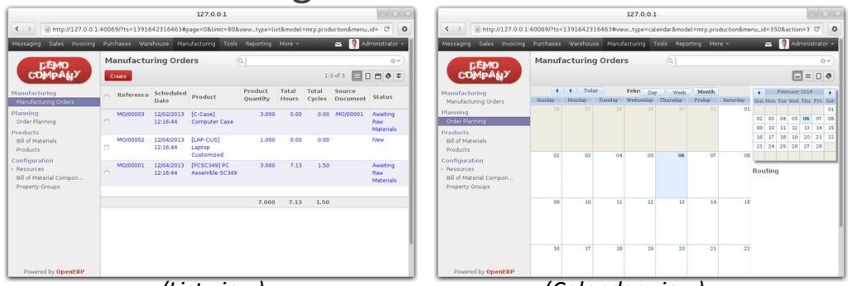
- Procurement Exceptions
- Incoming Shipments
- Incoming Products
- …



Screenshots of OpenERP v7.0

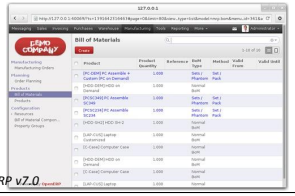
Manufacturing

- Manufacturing orders



(List view) *(Calendar view)*

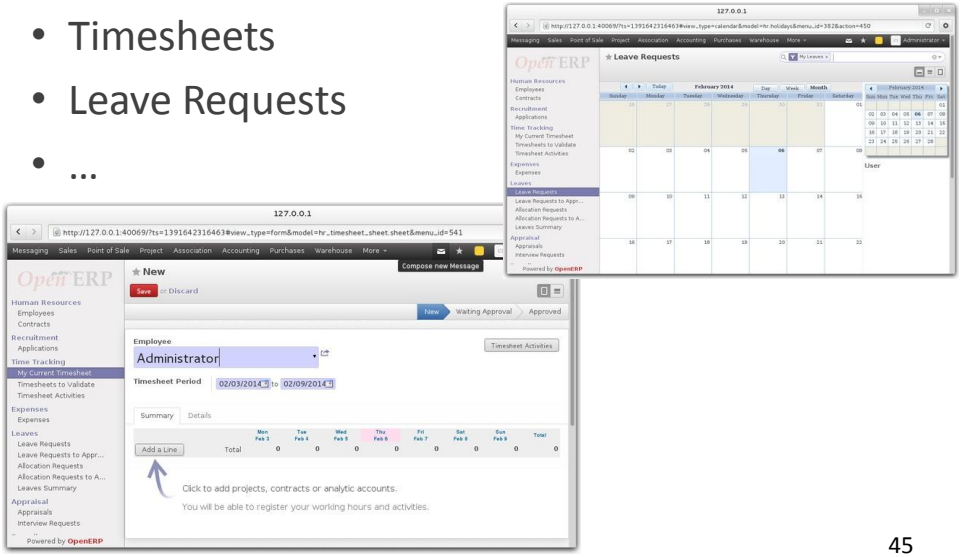
- Bill of materials



Screenshots of OpenERP v7.0

Human Resources

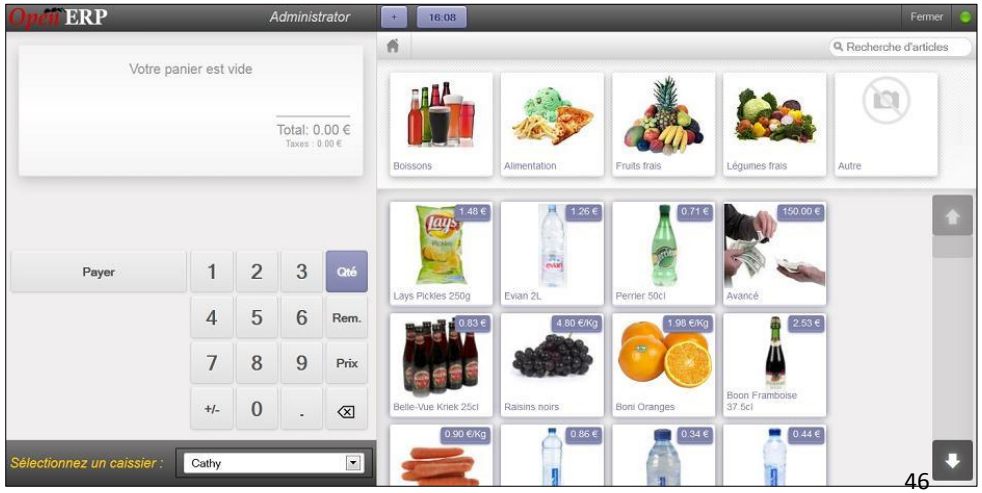
- Timesheets
- Leave Requests
- ...



Screenshots of OpenERP v7.0

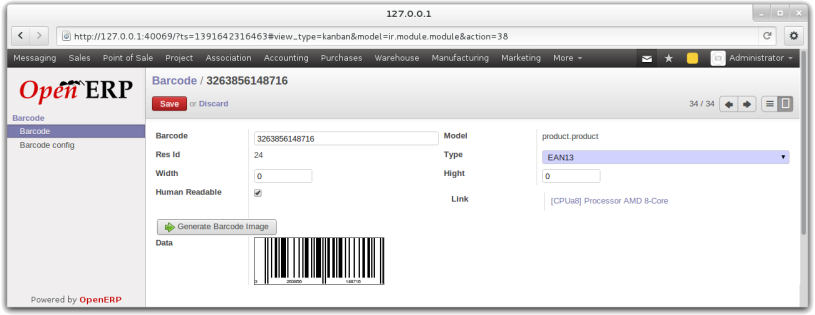
Point of sales

- Tactile interface



Point of sales

- Barcodes
- Receipts
- ...

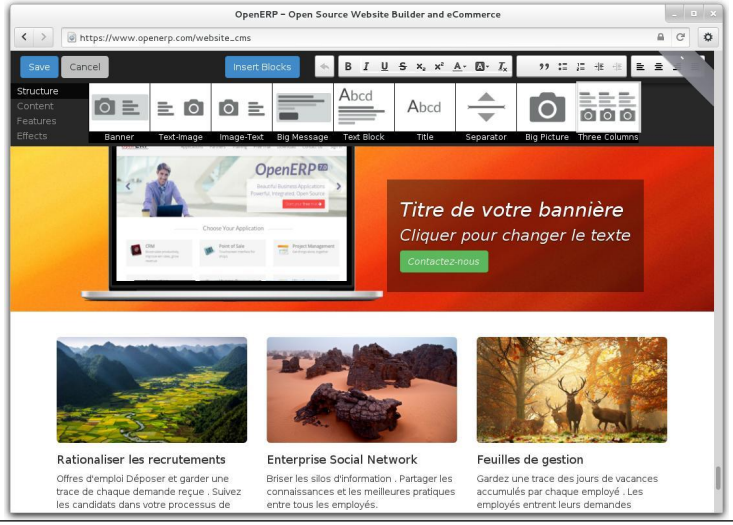


Point of sales

- PosBox `www.indiegogo.com/projects/opensource-your-shop`
  - Raspberry-Pi + Tablet + Printer + Scanner

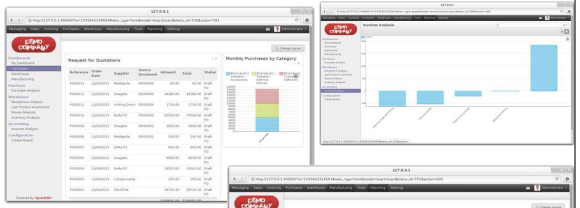
CMS & E-commerce

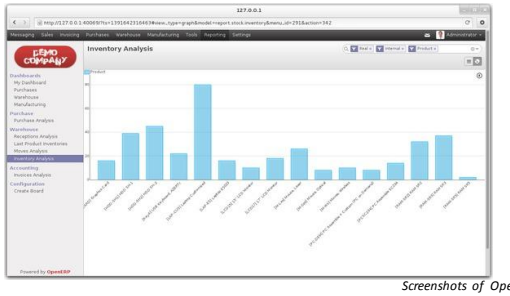
- Available since **Odoo 8.0** + website builder

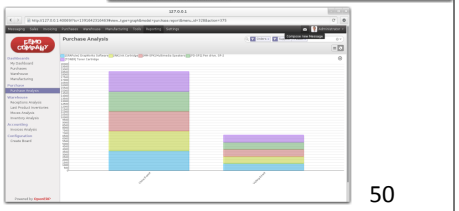


Reporting

- Invoice Analysis
- Purchase Analysis
- Inventory Analysis
- …



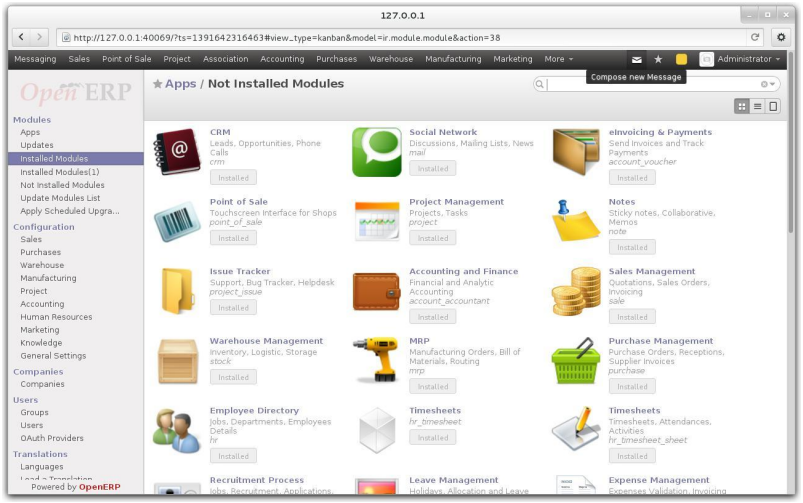




Screenshots of OpenERP v7.0

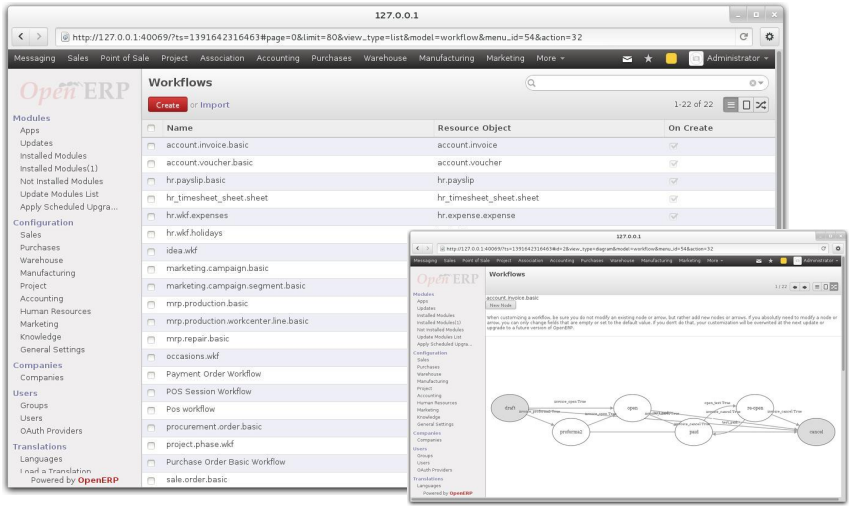
Settings
(Administrator)

- Install new module



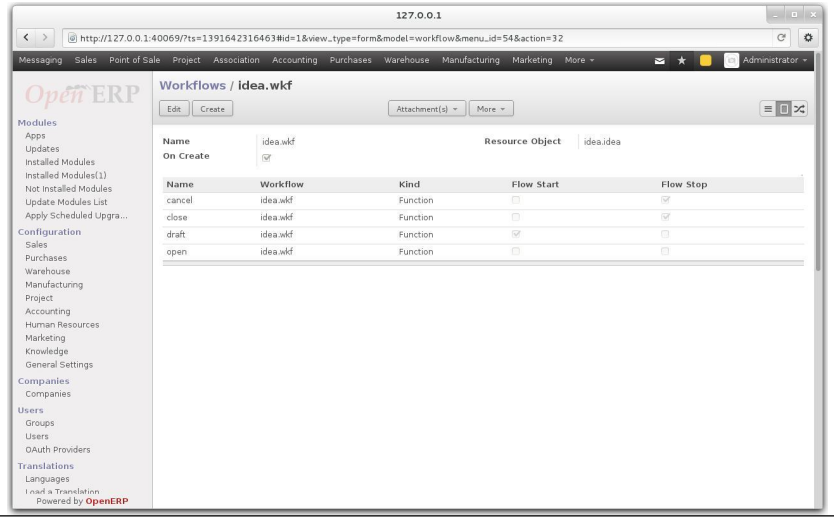
Settings
(Administrator)

- Parameter workflows



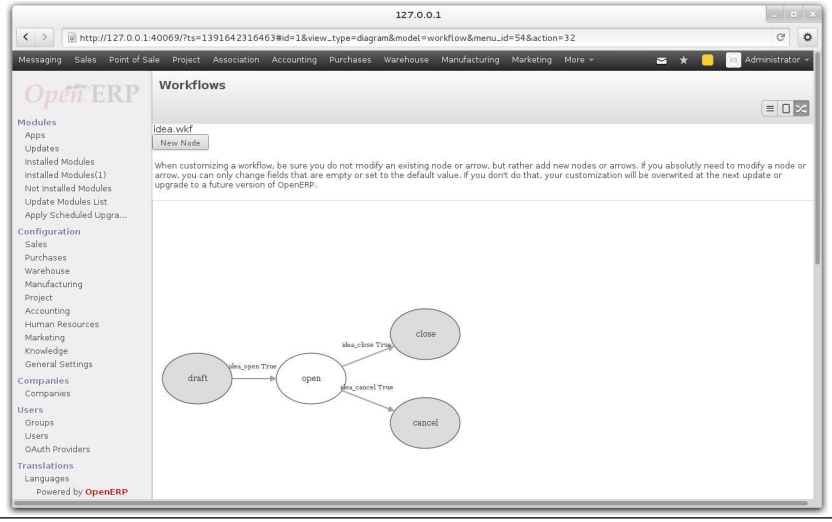
Settings
(Administrator)

- Example: workflow of an idea



Settings
(Administrator)

- Example: workflow of an idea



Getting sources

- Launchpad (Sourceforge, GNU Savannah, ...)
  - `https://launchpad.net/openobject`
- Version under developement (trunk)
  - `https://launchpad.net/openobject-server/trunk`
  - `https://launchpad.net/openobject-addons/trunk`
  - `https://launchpad.net/openerp-web/trunk`
- Older versions
  - `https://launchpad.net/openobject-server/7.0`
  - `https://launchpad.net/openobject-addons/7.0`
  - `https://launchpad.net/openerp-web/7.0`

Download sources

- Bazaar (cvs, svn, git, ...)
  - `bzr branch lp:openobject-server`
  - `bzr branch lp:openobject-addons`
  - `bzr branch lp:openerp-web`

Remove `.bzr` hidden directories!

- Older versions
  - `bzr branch lp:openobject-server/7.0`
  - `bzr branch lp:openobject-addons/7.0`
  - `bzr branch lp:openerp-web/7.0`

  - `bzr branch lp:openobject-server/6.0`
  - `bzr branch lp:openobject-addons/6.0`
  - `bzr branch lp:openerp-web/6.0`

Verticalizations

- Hostels
  - https://launchpad.net/~hotel-core-editors
- Electronic Medical Record and Hospital Information System
  - https://launchpad.net/medical
- Construction
  - https://launchpad.net/openerp-btp
  - https://launchpad.net/openerp-construction
- Pedagogy / Education Management System
  - https://launchpad.net/pedagogy
  - https://launchpad.net/openerp-ems

Verticalizations

- Textile & Spinning Manufacturing Industry
  - https://launchpad.net/openerp-textiles
- Humanitarian non-governmental organization
  - https://launchpad.net/openerp-humanitarian-ngo
- Library Management System
  - https://launchpad.net/openerp-library
- Veterinary and production farms
  - https://launchpad.net/openerp-openvet
- Agro farm management
  - https://launchpad.net/agrooffice

OUTLINE

1. Overview
   - Birth and growth
   - Customers, Partners, Vendors
2. Administration
   - Versions, Architecture
   - Main modules
3. Development
   - Module programming
   - Webservice programming
4. Documentation

Python

- Language used to program OpenERP modules
- Don't need to become an expert
- Short 12-slides-introduction
  - Indentation
  - Functions
  - Data structures
    - Lists, Tupples, Dictionnaries
  - Classes

3. DEVELOPMENT

Open ERP models

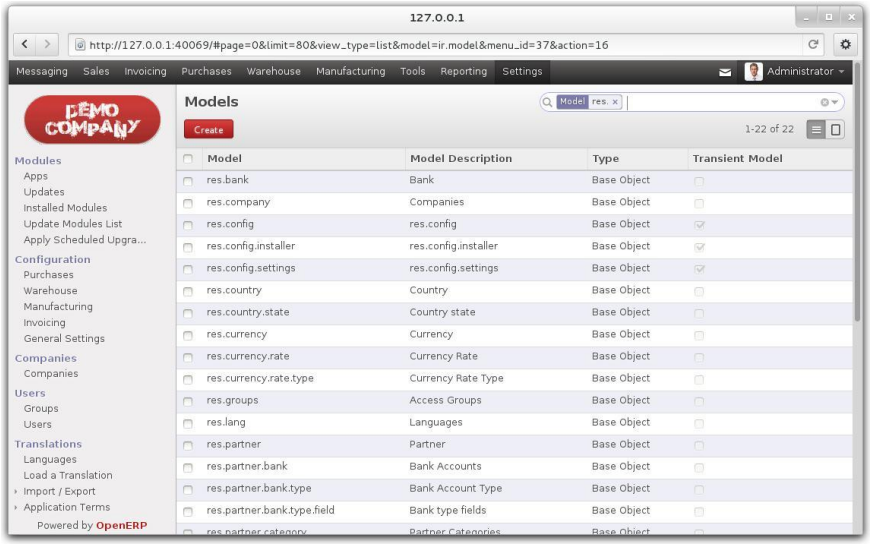
Module description

Fields of the model (columns)

Views (in XML)

Webservice communication

OpenERP database structure



OpenERP database structure

- Some of the numerous predefined models:
  - **base.module.configuration**
  - **base.module.import**
  - **base.module.update**
  - **res.users**
  - **res.partners**
  - **res.company**
  - **res.country**
  - **res.lang**
  - **res.currency**
  - **workflow**
  - **workflow.instance**
  - **workflow.triggers**
  - **workflow.activity**
  - **ir.ui.view**
  - **ir.ui.menu**
  - **ir.filters**
  - **ir.translation**
  - **ir.cron**

res.users

- Fields of the predefined model **res.users** :

Field name	*Field type*
id	integer
login	char
login_date	date
password	char
new_password	char
active	boolean
user_email	char
signature	text
company_id	many2one
company_ids	many2many
groups_id	many2many
partner_id	many2one
action_id	many2one
menu_id	many2one

Used in the modules:
auth_crypt,
auth_ldap,
auth_oauth,
auth_oauth_signup,
auth_openid,
auth_signup,
base,
hr,
mail,
note,
point_of_sale,
sale_crm,
share,
survey,
...

product.product

- Some fields of the model **product.product** :

Field name	*Field type*
code	char
color	integer
price	float
active	boolean
image	binary
track_incoming	boolean
track_outgoing	boolean
track_production	boolean
seller_id	many2one
seller_delay	integer
seller_qty	float
warehouse_id	many2one
qty_available	float
...	

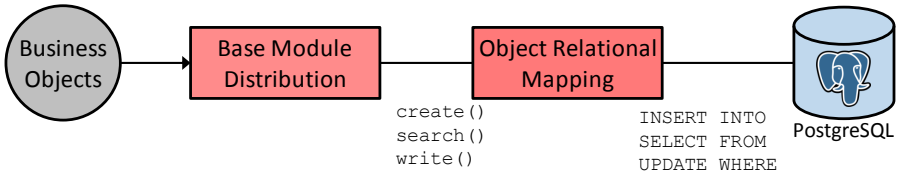
Used in the modules:
event_sale,
mrp,
point_of_sale,
procurement,
purchase_requisition,
stock,
stock_location,
warning,
...

A module

- A module defines a namespace
  - Several Python classes can be defined in the same namespace
- Each class defines a « model » / « business object », and fields are stored in PostgreSQL
  - Thanks to the ORM
  - All classes must inherit the class `osv.osv`
    - osv = Object SerVice
- Models can be shared by different modules

ORM

- **O**bject **R**elational **M**apping



  - Technique to program with a relational DB as if it was an object-oriented one
  - Makes the bridge between relational and OO worlds
  - Centralizes data integrity verifications and access rights management

Methods of the ORM

- **create**
- **search**
- **read**
- **browse**
- **write**
- **unlink**
- copy
- copy_data
- default_get
- fields_get
- name_get
- export_data
- import_data
- view_init
- check_access_rule
- get_xml_id
- name_search
- perm_read
- read_group
- fields_view_get

  - Can be called from business objects
  - Can also be called through WebServices

The **create()** method

`create(cr, user, vals, context=None)`

Create new record with specified value

Parameters:
- **cr** – database cursor
- **user** (integer) – current user id
- **vals** (dictionary) – field values for new record, e.g {'field_name': field_value, ...}
- **context** (dictionary) – (optional) context arguments, e.g. {'lang': 'en_us', 'tz': 'UTC', ...}

Returns: id of new record created

Raises:
- **AccessError**
  - if user has no create rights on the requested object
  - if user tries to bypass access rules for read on the requested object.
- **ValidateError** – if user tries to enter invalid value for a field that is not in selection
- **UserError** – if a loop would be created in a hierarchy of objects a result of the operation (such as setting an object as its own parent)

The **search()** method

`search(cr, user, args, offset=0, limit=None, order=None, context=None, count=False)`

Search for records based on a search domain

Parameters:
- **cr** – database cursor
- **user** (integer) – current user id
- **args** – list of tuples specifying the search domain [('field_name', 'operator', value), ...]. Pass an empty list to match all records.
- **offset** – (optional) number of results to skip in the returned values (default: 0)
- **limit** – (optional) max number of records to return (default: None)
- **order** – (optional) columns to sort by (default: self._order=id)
- **context** (dictionary) – (optional) context arguments, e.g. {'lang': 'en_us', 'tz': 'UTC', ...}
- **count** – (optional) if True, returns only the number of records matching the criteria, not their ids

Returns: id or list of ids of records matching the criteria

Return type: integer or list of integers

Raises:
- **AccessError**
  - if user tries to bypass access rules for read on the requested object.

The **browse()** method

`browse(cr, user, select, context=None, list_class=None, fields_process=None)`

Fetch records as objects allowing to use dot notation to browse fields and relations

Parameters:
- **cr** – database cursor
- **user** (integer) – current user id
- **select** – id or list of ids
- **context** (dictionary) – (optional) context arguments, e.g. {'lang': 'en_us', 'tz': 'UTC', ...}

Returns: object or list of objects requested

The **write()** method

`write(cr, user, ids, vals, context=None)`

Update records with given ids with the given field values

Parameters:
- **cr** – database cursor
- **user** (integer) – current user id
- **ids** – object id or list of object ids to update according to vals
- **vals** (dictionary) – field values to update, e.g {'field_name': new_field_value, ...}
- **context** (dictionary) – (optional) context arguments, e.g. {'lang': 'en_us', 'tz': 'UTC', ...}

Returns: `True`

Raises:
- **AccessError**
  - if user has no write rights on the requested object
  - if user tries to bypass access rules for write on the requested object
- **ValidateError** – if user tries to enter invalid value for a field that is not in selection
- **UserError** – if a loop would be created in a hierarchy of objects a result of the operation (such as setting an object as its own parent)

The **unlink()** method

`unlink(cr, user, ids, context=None)`

Delete records with given ids

Parameters:
- **cr** – database cursor
- **user** (integer) – current user id
- **ids** – object id or list of object ids to update according to vals
- **context** (dictionary) – (optional) context arguments, e.g. {'lang': 'en_us', 'tz': 'UTC', ...}

Returns: `True`

Raises:
- **AccessError**
  - if user has no write rights on the requested object
  - if user tries to bypass access rules for write on the requested object
- **UserError** – if a loop would be created in a hierarchy of objects a result of the operation (such as setting an object as its own parent)

3. DEVELOPMENT

Open ERP models

Module description

Fields of the model (columns)

Views (in XML)

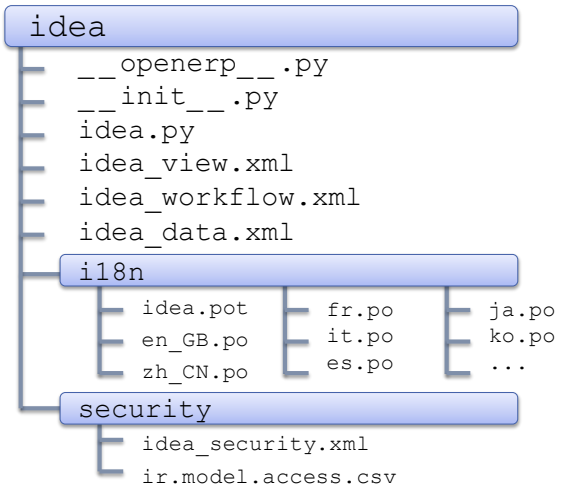
Webservice communication

Writting a module

- A module is composed of several files
  - Python files
    - Meta-informations about the module
    - Model for the ORM (Object-Relational Mapping)
  - XML files
    - Description of the views
    - Description of the workflows
    - Definition of demonstration data

Example of the module Idea

- Each module is contained in one directory



Python class of a model

- **Reserved Members** 
  - `_name` : business object name `'module_namespace.object_name'`
  - `_columns` : the fields of the model
  - `_defaults` : default values for fields
  - `_inherit` : `_name` of the parent
  - `_sql_constraints` : tuples defining SQL constraints
    - (name, sql_def, message)
  - `_constraints` : tuples defining Python constraints
    - (func_name, message, fields)
  - `_order` : the field to sort records (default field: `'id'`)
  - `_auto`, `_inherits`, `_log_access`, `_rec_name`, `_sql`, `_table`

Python class of a model

- Example of members declaration

```
class idea_idea(osv.osv):
  """ Idea """
  _name = 'idea.idea'
  _inherit = ['mail.thread']
  _columns = {
    'name': fields.char('Idea Summary', size=64, required=True,
    'description': fields.text('Description', help='Content of
    'state': fields.selection([('draft', 'New'),('open', 'Accep
  }
  _sql_constraints = [
    ('name', 'unique(name)', 'Idea name must be unique')
  ]
  _defaults = {
    'state': lambda *a: 'draft',
  }
  _order = 'name asc'
```


3. DEVELOPMENT

Open ERP models

Module description

Fields of the model (columns)

Views (in XML)

Webservice communication

Fields of the model

- Declared as a dictionnary in the `_columns` member of the class
- Example

```
class idea_idea(osv.osv):
  _columns = {
     'create_uid': fields.many2one('res.users', 'Creator', requ
     'name': fields.char('Idea Summary', size=64, required=True
     'description': fields.text('Description', help='Content of
     'category_ids': fields.many2many('idea.category', string='
     'state': fields.selection([('draft', 'New'),
         ('open', 'Accepted'),
         ('cancel', 'Refused'),
         ('close', 'Done')],
         'Status', readonly=True, track_visibility='onchange',
       )
    }
```

Columns (fields)

- **RESERVED COLUMNS NAMES (i.e. FIELDS)**
  - `id` : unique identifier for the object (created by ORM, don't add it!)
  - `name` : defines the label to display in lists, etc.
  - `state` : defines object life-cycle stages (used for workflow)
  - `active` : defines visibility
  - `sequence` : defines order
  - `create_date`, `create_uid`, `write_date`, `write_uid`: used to log meta information about the record (automatically defined and set by the ORM, so don't define this column!)
  - `parent_id`, `parent_left`, `parent_right` : defines tree structure on records

Type of fields

- Basic types
  - integer
  - float
  - boolean
  - char
  - text
  - selection
  - date
  - datetime
  - time
  - binary
- Relational types
  - many2one
  - one2many
  - many2many
  - ~~one2one~~ → many2one

Fields declaration

- Optional parameters
  - `string` : the field label **(required)**
  - `required` : set it to *`True`* if mandatory field
  - `readonly` : set it to *`True`* if not editable field
  - `help` : text for the help tooltip
  - `context` : dictionary with contextual parameters (for relational fields)
  - `states` : dynamically changes the field's attributes according to the changing state of the object

integer type

```
fields.integer(string='Field Name' [, Opt. Param.]),
```

`string` parameter, which defines the label

```
fields.integer('Field Name' [, Optional Parameters]),
```

- Example of declaration

```
_columns = {
 'level': fields.integer('Level of development'),
 'code': fields.integer('Reference code', readonly=True),
}
```

- Example of assignment

```
self.write(cr, uid, ids, {'level': 5})
```

float type

```
fields.float('Field Name' [, Optional Parameters]),
```

- Example of declaration

```
_columns = {
 'weight': fields.float('Product weight in kg'),
 'price': fields.float('Price in EUR', digits=(12,6)),
 'rate': fields.float('FRS-EUR exchange rate', digits=(12,6),
                      readonly=True),
}
```

- Example of assignment

```
self.write(cr, uid, ids, {'weight': 0.256})
```


boolean type

```
fields.boolean('Field Name' [, Optional Parameters]),
```

- Example of declaration

```
_columns = {
 'is_active': fields.boolean('Active product'),
 'msg_unread': fields.boolean('Unread message', readonly=True),
}
```

- Example of assignment

```
self.write(cr, uid, ids, {'msg_unread': False})
```


char type

- A string of limited length

```
fields.char('Field Name', size=n [, Optional Param.]),
```

- Example of declaration

```
_columns = {
 'name': fields.char('Category Name', size=64, required=True),
 'code': fields.char('Reference code', size=9, readonly=True),
}
```

- Example of assignment

```
self.write(cr, uid, ids, {'code': '000AR400M'})
```


text type

- A string with no limited length

```
fields.text('Field Name', [, Optional Parameters]),
```

- Example of declaration

```
_columns = {
 'description': fields.text('Description'),
 'purchase_msg': fields.text('Message for purchase order'),
}
```

- Example of assignment

```
self.write(cr, uid, ids, {'description': 'This article...'})
```

selection type

- Selection between predefined values

```
fields.selection(values, 'Field Name', [, Opt.Param.]),
```

  – where the parameter `values` is a list, or a tuple, of 2-tuples formed as `(key, value)`

  - Example with a list of 2-tuples:

```
values = [('n','Unconfirmed'),('c','Confirmed')]
```

  - Example with a tuple of 2-tuples:

```
values = (('n','Unconfirmed'),('c','Confirmed'))
```


selection type

- Examples of declaration

```
_columns = {
 'size': fields.selection(
     [('-1','Undefined'),('0','Small'),
      ('1', 'Medium'),('2', 'Large')],
     'Package size'),
}
```

```
countries = [('fr','France'),('es','Spain'),('it', 'Italy')]

_columns = {
 'dep': fields.selection(countries, 'Departure'),
 'arr': fields.selection(countries, 'Arrival'),
}
```

- Example of assignment

```
self.write(cr, uid, ids, {'size': '-1'})
```

date type

```
fields.date('Field Name' [, Optional Parameters]),
```

- Example of declaration

```
_columns = {
 'birthday': fields.date('Date of birth'),
 'last_mod': fields.date('Last modification', readonly=True),
}
```

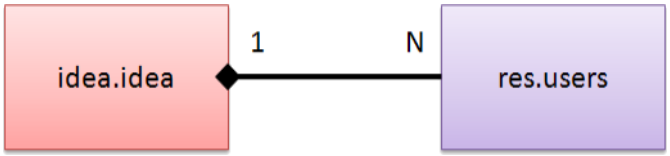
- Example of assignment

```
self.write(cr, uid, ids, {'last_mod': datetime.now()})
```


many2one relation

```
fields.many2one('model', 'Field Name' [, Opt.Param.]),
```

- Example



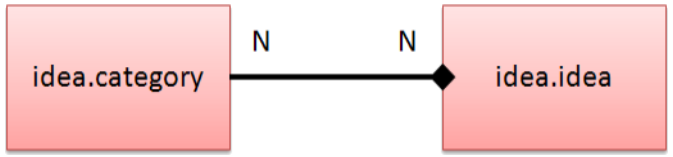

```
class idea_idea(osv.osv):
  _columns = {
    'create_uid': fields.many2one('res.users', 'Creator'),
  }
```

  – An idea is created by **1** (and only 1) user, but a user can create **N** ideas

many2many relation

```
fields.many2many('model', string='Field Name' [, Par]),
```

- Example



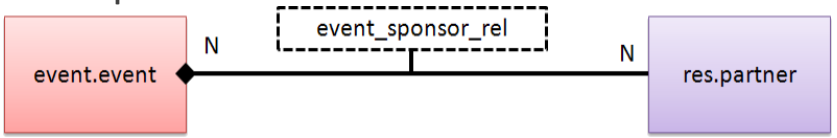

```
class idea_idea(osv.osv):
  _columns = {
    'category_ids': fields.many2many('idea.category',
                                     string='Tags'),
  }
```

– An idea can belong to **several** categories, and a category can describe **several** ideas

many2many relation

```
fields.many2many('modelB', 'rel', 'idA', 'idB' [, Par]),
```

- Example

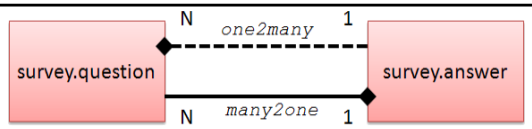



```
class event_event(osv.osv):
  _columns = {
    'sponsor_ids': fields.many2many('res.partner',
                                    'event_sponsor_rel',
                                    'event_id',
                                    'sponsor_id',
                                    'Sponsors'),
  }
```


one2many "virtual" relation

```
fields.one2many('model', 'id', 'Field Name', [, Par.]),
```

- Example



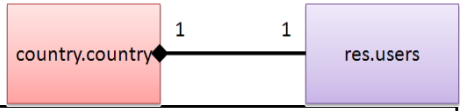

```
class survey_question(osv.osv):
  _columns = {
    'answer_choice_ids': fields.one2many('survey.answer',
                                         'question_id',
                                         'Answer'),
  }

class survey_answer(osv.osv):
  _columns = {
    'question_id': fields.many2one('survey.question',
                                   'Question'),
  }
```

Match corresponding foreign key

one2one relation (deprecated)

```
fields.one2one('model','Field Name' [, Opt.Param.]),
fields.many2one('model','Field Name' [, Opt.Param.]),
```

- Example




```
class country_country(osv.osv):
  _columns = {
    'president_id': fields.one2one('res.users', 'President'),
  }
```

```
class country_country(osv.osv):
  _columns = {
    'president_id': fields.many2one('res.users', 'President'),
  }
```

– Each country has **1** (and only 1) president, and a person can be president of only **1** country

Relation fields' names

- By convention,
  - many2one fields end with '_id'
  - one2many fields end with '_ids'
  - many2many fields end with '_ids'

```
class survey_answer(osv.osv):
  _columns = {
    'question_id': fields.many2one(
class survey_question(osv.osv):
  _columns = {
    'answer_choice_ids': fields.one2many(
class idea_idea(osv.osv):
  _columns = {
    'category_ids': fields.many2many(
```


3. DEVELOPMENT

Open ERP models

Module description

Fields of the model (columns)

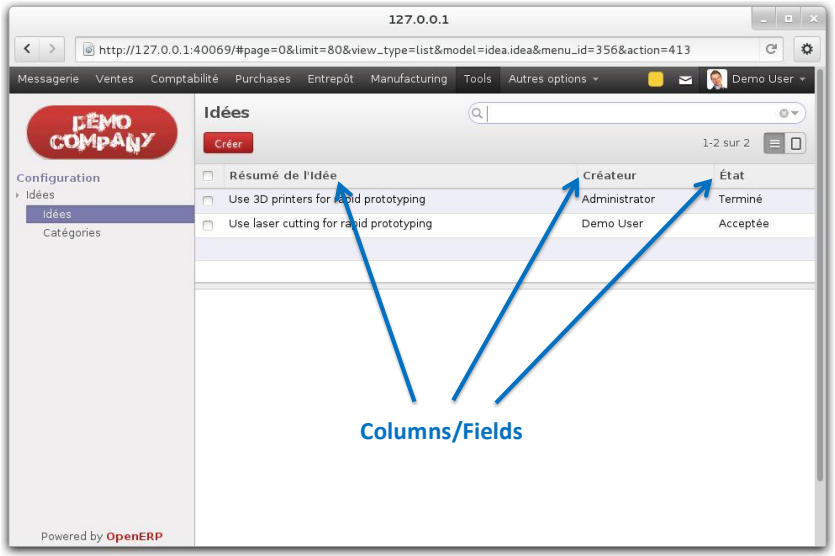
Views (in XML)

Webservice communication

XML files for views

- Views (ir.ui.view)
  - Tree view
  - Form view
  - Search view
- Actions (ir.actions.act_window)
- Menus (menuitem)

Tree view

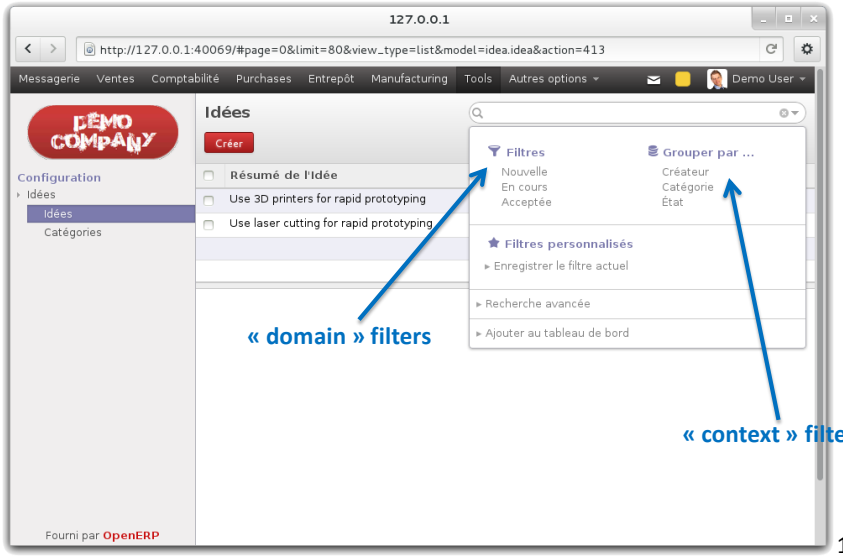


Tree view

- Extracted from `idea_view.xml`

```
<record model="ir.ui.view" id="view_idea_idea_tree">      ⟸ XML id
  <field name="name">idea.idea.tree</field>               ⟸ Name of the view
  <field name="model">idea.idea</field>                   ⟸ Name of the business object

  <field name="arch" type="xml">                          ⟸ Use XML tags to define the view
    <tree colors="blue:state == 'draft';black:state in ('open',
      <field name="name"/>                                ⟸ Names of the columns to display
      <field name="create_uid"/>                          ⟸
      <field name="state"/>                               ⟸
    </tree>
  </field>

</record>
```


Search view



Search view

- Extracted from `idea_view.xml`

```
<record model="ir.ui.view" id="view_idea_idea_search">      ⟸ XML id
 <field name="name">idea.idea.search</field>                 ⟸ Name of the view
 <field name="model">idea.idea</field>                      ⟸ Name of the business object
 <field name="arch" type="xml">                             ⟸ Use XML tags to define the view
  <search string="Ideas">
   <field name="name" string="Idea"/>
   <filter string="New" domain="[('state','=', 'draft')]" />
   <filter string="In Progress" domain="[('state','=', 'open')]" />
   <filter string="Accepted" domain="[('state','=','close')]" />
   <field name="category_ids"/>
   <group expand="0" string="Group By...">
    <filter string="Creator" context="{'group_by':'create_uid'}"/>
    <filter string="Category" context="{'group_by':'category_ids'}"/>
    <filter string="Status" context="{'group_by':'state'}"/>
   </group>
  </search>
 </field>
</record>
```


Filters

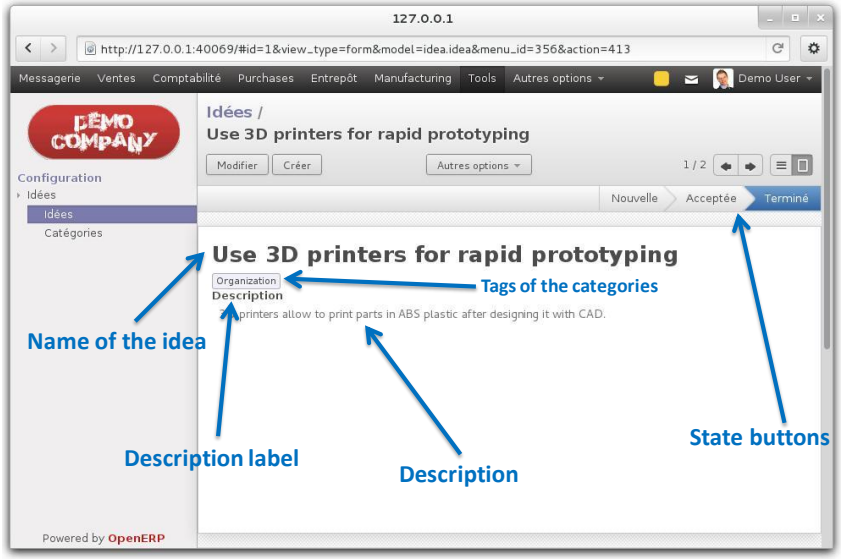
- Condition: (Field,Operator,Value)
  - Several conditions
    - Prefix notation (Polish notation)

```
['|',('country_code','=','be'),('country_ code','=','fr')]
```

  - XML characters

Operator	XML
=	=
≠	!=
<	<
≤	<=
>	>
≥	>=

Form view



Form view

- Extracted from `idea_view.xml`

```
<record model="ir.ui.view" id="view_idea_idea_form">        ⟸ XML id of the view
 <field name="name">idea.idea.form</field>                  ⟸ Name of the view
 <field name="model">idea.idea</field>                      ⟸ Name of the business object
 <field name="arch" type="xml">                             ⟸ Use XML tags to define the view
  <form string="Idea" version="7.0">
   <header>
    <button name="idea_open" string="Open" states="draft" class="oe_highlight"/>
    <button name="idea_close" string="Accept" states="open" class="oe_highlight"/
    <button name="idea_cancel" string="Refuse" states="open" class="oe_highlight"
    <field name="state" widget="statusbar" statusbar_visible="draft,open,close"/>
   </header>
   <sheet>
    <label for="name" class="oe_edit_only"/>
    <h1><field name="name"/></h1>
    <label for="category_ids" class="oe_edit_only"/>
    <field name="category_ids" widget="many2many_tags"/>
    <label for="description"/><newline/>
    <field name="description"/>
   </sheet>
   <div class="oe_chatter">
    <field name="message_follower_ids" widget="mail_followers"/>
```


Action and menuitem

- Extracted from `idea_view.xml`

```
<record model="ir.actions.act_window" id="action_idea_idea">
  <field name="name">Ideas</field>
  <field name="res_model">idea.idea</field>
  <field name="view_type">form</field>
  <field name="view_mode">tree,form</field>
  <field name="search_view_id" ref="view_idea_idea_search"/>
</record>

<menuitem name="Ideas" parent="menu_ideas" id="menu_idea_idea"
                       action="action_idea_idea" sequence="1"/>
```

- Must be defined at **after** the declarations of the views

3. DEVELOPMENT

Open ERP models

Module description

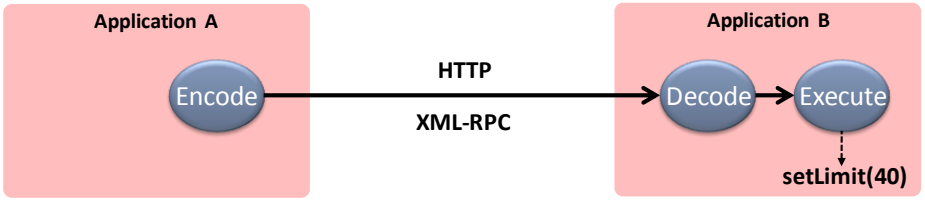
Fields of the model (columns)

Views (in XML)

Webservice communication

Webservices

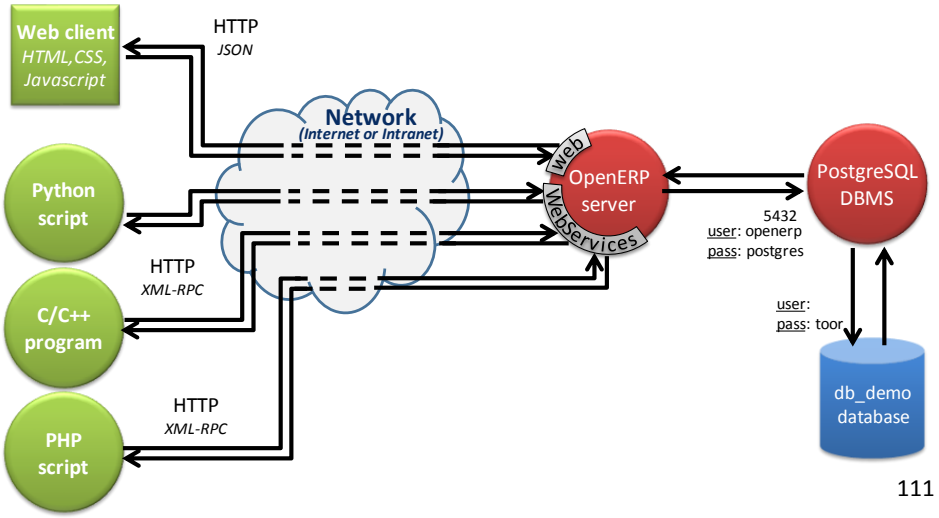
- SOA: **S**ervice **O**riented **A**rchitecture
- WOA: **W**eb **O**riented **A**rchitecture
- XML-RPC: XML **R**emote **P**rocedure **C**all ⇒ OpenERP




```
<?xml version="1.0" ?>
<methodCall>
 <methodName>setLimit</methodName>
 <params>
  <param><value><int>40</int></value></param>
 </params>
</methodCall>
```


Webservice with OpenERP

- OpenERP v7 v8 v9



OpenERP webservice

- Allows calling the methods of the ORM
  - **Create**: create a new record
  - **Write**: update a record
  - **Search**: look records matching a criteria
  - **Read**: get the values of a record
  - **Unlink**: delete a record
- Steps
  - 1 : connexion and identification
  - 2 : execution
  - 3 : disconnexion

Calling ORM methods

- Can be called from any program written with a language having a XML-RPC library
  - Python
  - C/C++
  - Objective-C
  - Java *(and also with android-xmlrpc)*
  - JavaScript
  - PHP *(with Pear library for PHP)*
  - Perl
  - ...

The **create()** ORM method

- `create({'field':'value'})`
  - Creates a new record with the specified value
  - Returns: id of the new record
- Example with Python

```
import xmlrpclib

# Get the uid
sock_common = xmlrpclib.ServerProxy ('http://127.0.0.1:8069/xmlrpc/common')
uid = sock_common.login('db_name', 'username', 'passwd')

# Ask for execution on OpenERP
sock = xmlrpclib.ServerProxy('http://127.0.0.1:8069/xmlrpc/object')
partner_id = sock.execute('db_name', uid, 'passwd',
                          'res.partner',
                          'create',
                          {'name': 'Fabien Pinckaers', 'lang': 'en_US', })
```


Example with PHP

- Example with PHP *(without error checking)*

```
require_once('XML/RPC.php'); // Include PEAR library for XML-RPC

// Login

$client = new XML_RPC_Client('/xmlrpc/common', "http://127.0.0.1:8069");
$client->setDebug(0);

$msg = new XML_RPC_Message('login');
$msg->addParam(new XML_RPC_Value('db_name', 'string'));
$msg->addParam(new XML_RPC_Value('username', 'string'));
$msg->addParam(new XML_RPC_Value('passwd', 'string'));

$resp = $client->send($msg); // Send execution request on OpenERP
$val = $resp->value();
$uid = $val->scalarval();
```


Example with PHP

```
// Request

$client = new XML_RPC_Client('/xmlrpc/object', "http://127.0.0.1:8069");
$client->setDebug(0);

$structVal = array(
   'name' => new XML_RPC_Value('Fabien Pinckaers', 'string'),
   'lang' => new XML_RPC_Value('en_US', 'string'),
);

$msg = new XML_RPC_Message('execute');
$msg->addParam(new XML_RPC_Value('db_name', 'string'));
$msg->addParam(new XML_RPC_Value($uid, 'int'));
$msg->addParam(new XML_RPC_Value('passwd', 'string'));
$msg->addParam(new XML_RPC_Value('res.partner', 'string'));
$msg->addParam(new XML_RPC_Value('create', 'string'));
$msg->addParam(new XML_RPC_Value($structVal, 'struct'));

$resp = $client->send($msg); // Send execution request on OpenERP
$val = $resp->value() ;
$id = $val->scalarval() ;
```


The **search()** ORM method

- `search([('arg1','=','value1')...], offset=0, limit=1000)`
  - arg1, arg2, .. ,argN: list of tuples specifying search criteria
  - offset: optional number of records to skip
  - limit: optional max number of records to return
  - Returns: list of IDS of records matching the given criteria

The **read()** ORM method

- `read([IDS], ['field1','field2',...])`
  - fields: optional list of field names to return (default: all fields)
  - Returns: the id of each record and the values of the requested field

The **write()** ORM method

- `write([IDS], {'field1':'value1','field2':3})`
  - Updates records with given ids with the given values
  - values: dictionary of field values to update
  - Returns: True

The **unlink()** ORM method

- `unlink([IDS])`
  - Deletes records with the given ids
  - Returns: True

OUTLINE

Documentation

- Documentation
  - `https://doc.odoo.com/`
  - `https://doc.odoo.com/7.0/fr/`
- Documentation technique
  - `https://doc.odoo.com/trunk/server/`
- Free E-Books
  - `https://www.odoo.com/ebooks`

Books

Authors	Title	Editor	Year	Pages
Fabien Pinckaers, Geoff Gardiner	Tiny ERP-Open ERP : Pour une gestion d'entreprise efficace et intégrée	Eyrolles	2008	276
F.Pinckaers, E.Van Vossel, B. Proust	Gestion commerciale et marketing avec Open ERP	Eyrolles	2012	183
Pinckaers,Gardiner, Van Vossel	Open ERP: a modern approach to integrated business management	Tiny SPRL	2011	456
Els Van Vossel, Fabien Pinckaers	Streamline your Manufacturing Processes with OpenERP	Tiny SPRL	2011	230
Fabien Pinckaers, Geoff Gardiner	Open ERP for Retail and Industrial Management	Tiny SPRL	2009	320
Els Van Vossel, Fabien Pinckaers	Integrate your Logistic Processes with OpenERP	Tiny SPRL	2011	230
Els Van Vossel, Fabien Pinckaers	Drive your Sales & Marketing Activities with OpenERP	Tiny SPRL	2011	202

Books

Authors	Title	Editor	Year	Pages
Els Van Vossel, Fabien Pinckaers	OpenERP for Accounting and Financial Management	Tiny SPRL	2012	188
Fabien Pinckaers, Geoff Gardiner	Analytic & Financial Accouting Book	Tiny SPRL	2009	157
Fabien Pinckaers, Els Van Vossel	Open Source Accounting with OpenERP	Tiny SPRL	2011	176
Y. Delsart, C. Van Nieuwenhuysen	OpenERP evaluation with SAP as reference	Feridis	2011	132
Simon André	Piloter Prestashop avec OpenERP	Anybox	2012	27
Florent Pigout	Piloter Magento avec OpenERP	Anybox	2011	91
	Open Object Business Intelligence	Tiny SPRL	2009	113
	Open Object Developer Book	Tiny SPRL	2009	213
	OpenERP Agile Implementation Memento	Open Object	2010	5
	OpenERP Technical Memento	Open Object	2010	18

Books

Authors	Title	Editor	Year	Pages
Greg Moss	Working with OpenERP	Packt	2013	334
M. Belaissaoui, A. Elkafil	Gestion Commerciale avec OpenERP 7 par la pratique	Éd. Univ. Européennes	2013	76
Karl Hellec	Open ERP : Découverte et mise en œuvre	GEP	2013	110
Daniel Reis	Odoo Development Essentials	Packt	2015	214
Daniel Reis	Odoo 10 Development Essentials	Packt	2016	289
Greg Moss	Working with Odoo 10	Packt	2017	415

Books for high school :

S. Fasciotti, F. Roche, *et al.*	2nd Bac Pro - Informatique - Office 2010, Access, Ciel, EBP PGI, OpenERP	Nathan	2012	245
D. Le Rouzic	Activités sur... OpenERP Réseau	B. Lacoste		160