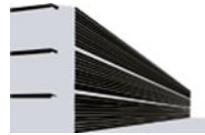




<b>PROYECTO</b>	MODULO DE HOSTELERIA EN OODO 8
<b>ALUMNO</b>	CARLOS HUELMO VAQUERO
<b>CICLO</b>	DESARROLLO DE APLIC. MULTIPLATAFORMA
<b>CURSO</b>	2014 – 2015
<b>ACADEMICO</b>	
<b>TUTOR</b>	CARLOS ALBERTO ESCLAPÉS

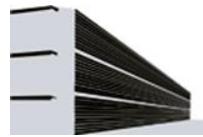


## AGRADECIMIENTOS

Aprovecho para dar las gracias al apoyo recibido por mis profesores de Sistemas de Gestión Empresarial, Carlos Alberto Esclapés y de Acceso a Datos, Sergio Galisteo Castro por su tiempo y paciencia para el desarrollo de este proyecto. A mi tutor Toni Santos por su trabajo y constancia durante los dos años del ciclo de DAM.

Tampoco olvidar al instructor de **DIAGRAM SOFTWARE, S.L.**, Rafael Priego así como a los distintos empleados del “**Departamento de Desarrollo**”: Cristian Moncho Ivorra, Mauro Cebriá, Francisco Casasempere “Sisco”, José Zambudio Bernabeu “Zambu”, a los universitarios Rubén Cerdà, Pedro Albujer, Héctor Azuar y a todas aquellas personas con las que he tenido una relación indirecta durante las prácticas en la empresa.

Alcoy, a 22 de mayo de 2015.



<b>ÍNDICE</b>	<b>PÁGINA</b>
<b>1.- Introducción.....</b>	<b>5</b>
1.1.- Breve resumen del proyecto.....	5
1.2.- Motivación.....	5
1.3.- Objetivos del proyecto.....	5
1.4.- Metodología de trabajo.....	6
1.5.- Otros comentarios.....	6
<b>2.- Introducción a Odoo.....</b>	<b>7</b>
2.1.- Los sistemas de gestión de recursos empresariales.....	7
2.1.1.- Las tecnologías en la empresa.....	7
2.1.2.- Definición de sistema de gestión.....	7
2.1.3.- Principales características de un sistema ERP.....	8
2.1.4.- Historial del ERP.....	9
2.1.5.- Objetivos y consideraciones al implantar un ERP.....	10
2.1.6.- Ventajas e inconvenientes de los ERP.....	10
<b>3.- Requerimientos.....</b>	<b>11</b>
3.1.- Requerimientos de hardware.....	11
3.2.- Requerimientos de software.....	11
3.3.- Fases del desarrollo de un software.....	11
3.4.- Ciclo de vida de la base de datos.....	12
<b>4.- Instalación y configuración de Odoo.....</b>	<b>12</b>
<b>5.- Diseño de la base de datos “L'Assaig”.....</b>	<b>12</b>
5.1.- Toma de datos.....	12
5.2.- Esquema conceptual (ER).....	13
5.3.- Esquema lógico (relacional).....	13
<b>6.- Creación del módulo.....</b>	<b>16</b>
6.1.- Estructura básica.....	16
6.2.- Modelos.....	18
6.3.- Vistas (views).....	24
6.3.1.- Vista search.....	24
6.3.2.- Vista form.....	25
6.3.3.- Vista tree.....	29
6.3.4.- Vista calendar.....	30
6.3.5.- Vista kanban.....	31
6.3.6.- Vista graph.....	32
6.4.- Actions.....	34



6.5.- Menús.....	36
6.6.- Temas y estilos.....	37
6.7.- Mejora de la descripción del módulo.....	37
<b>7.- Seguridad en Odoo.....</b>	<b>38</b>
7.1.- Fichero “ir.model.access.csv”.....	39
7.2.- Mejorar la seguridad del módulo.....	42
<b>8.- Internacionalización.....</b>	<b>48</b>
8.1.- Exportando nuestra traducción.....	49
8.2.- Fichero “ca_ES.po”.....	51
<b>9.- Reportes en Odoo.....</b>	<b>53</b>
9.1.- Motores de informes.....	53
9.2.- Webkit.....	54
9.3.- Reportalb.....	55
9.4.- Jasper Reports & iReports.....	59
9.5.- Máscara / formato a campos de un informe.....	66
9.6.- Realizar subinformes.....	68
9.7.- Seguridad en informes.....	69
<b>10.- Exportar e importar.....</b>	<b>70</b>
<b>11.- Vocabulario.....</b>	<b>73</b>
<b>12.- Problemas encontrados.....</b>	<b>74</b>
<b>13.- Fuentes / enlaces.....</b>	<b>75</b>
<b>ANEXO I .- Editores para traducciones.....</b>	<b>78</b>
<b>ANEXO II .- Creación de un módulo en webkit.....</b>	<b>80</b>
<b>ANEXO III .- Instalación de Odoo.....</b>	<b>105</b>
III.A.- MS Windows.....	105
III.B.- GNU / Linux Debian 7.8.....	110
III.C.- GNU / Linux Xubuntu 14.04.02 LTS.....	111
<b>Conclusiones finales.....</b>	<b>115</b>



## 1.- INTRODUCCIÓN.

### 1.1.- BREVE RESUMEN DEL PROYECTO.

Desde la dirección del centro nos han solicitado la creación de un módulo en OpenERP que gestione las mesas del restaurante L'Assaig y que además permita realizar reservas.

Para la gestión de las mesas habrá que tener en cuenta la superficie útil para ubicarlas así como la capacidad tomada como referencia a la hora de acomodar a los clientes, disponibilidad de las mesas, seguridad y permisos para los usuarios del módulo.

### 1.2.- MOTIVACIÓN.

Al utilizar por primera vez OpenERP durante el segundo curso de DAM (Desarrollo de Aplicaciones Multiplataforma) al alumno le gustó el funcionamiento y diseño de dicho programa aunque fuera en su versión más antigua, comprobando las características y funcionalidades de que dispone.

Aprovechando que la empresa donde el alumno realiza las FCT's (Formación en Centros de Trabajo) trabaja sobre OpenERP y los conocimientos adquiridos durante el curso en el módulo de Ssistemas de Gestión Empresarial es una oportunidad única de desarrollar, mejorar y profundizar sobre el funcionamiento completo (aunque sea en parte) sobre dicho software open source con licencia AGPL<sup>1</sup> de la mano de expertos programadores y desarrolladores.

### 1.3.- OBJETIVOS DEL PROYECTO.

Mediante este proyecto se pretende poner en práctica los conocimientos que el alumno ha adquirido durante el curso, más concretamente en el módulo de "Sistemas de Gestión Empresarial" y mostrarlos mediante esta memoria que detallaría el supuesto funcionamiento en una empresa dedicada al sector de la hostelería.

En el resumen del proyecto hemos indicado que la empresa utiliza un ERP<sup>2</sup>, esto es, un sistema donde se integra todo lo necesario para la gestión de una empresa y dentro de la misma aplicación.

El restaurante L'Assaig dispone actualmente de OpenERP por lo que el alumno desarrollará un nuevo módulo para mejorar la gestión de las mesas y reservas.

- 
- 1 **GNU Affero General Public License:** es una licencia copyleft (método para que un programa sea libre, exigiendo que todas las versiones modificadas y extendidas del mismo sean también libres, manteniendo los derechos de autor) derivada de la Licencia Pública General de GNU.
  - 2 **Enterprise Resource Planning,** llamado sistema de planificación de recursos empresariales, es un sistema de información empresarial que integran y manejan muchos de los negocios asociados con las operaciones de producción y de los aspectos de distribución de una compañía en la producción de bienes o servicios (fuente: Wikipedia).



#### 1.4.- METODOLOGIA DE TRABAJO.

En principio el alumno comenzó utilizando toda la documentación y ejercicios proporcionados durante el curso de Sistemas de Gestión Empresarial, después utilizando la última versión de OpenERP (Odoo) siguió con el desarrollo de un módulo básico que posteriormente fue aplicado como CMS o gestor web y que también se pretende añadir como una extensión al módulo de TPV para complementar dicho módulo proporcionado por Odoo pero de forma genérica.

El material consultado es en gran parte páginas web, foros, documentación oficial on-line de la propia empresa desarrolladora de Odoo, documentos PDF, etc., además de realizar consultas puntuales a los propios empleados de la empresa debido a la sobrecarga de trabajo a la que se encuentran expuestos semanalmente.

#### 1.5.- OTROS COMENTARIOS.

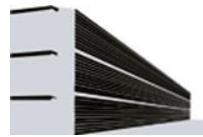
En mayo de 2014 “OpenERP” pasó a llamarse “Odoo” porque ya no es un ERP sino que la última versión (Odoo 8) ya incluye un gestor web (CMS<sup>3</sup>), un gestor de tienda online y un motor de inteligencia empresarial (Business Intelligence) por lo cuál la aplicación llega a ser más completa para gestionar toda una empresa: compras, ventas, almacén, fabricación, contabilidad, CRM, terminal de punto de venta (TPV), gestión de proyectos, recursos humanos, etc.

Gracias a la potencia, rendimiento y flexibilidad proporcionadas por Odoo podremos personalizarlo según nuestras necesidades presentes y/o futuras.

Durante la memoria del proyecto nos referiremos al sistema OpenERP como “Odoo” ya que el proyecto se ha desarrollado también en la última versión de Odoo “8”.

---

3 **Content Management System**, programa informático que permite crear una estructura de soporte para la creación y administración de contenidos (como páginas web), por parte de los administradores, editores, participantes y resto de usuarios. (fuente: Wikipedia).



## 2.- INTRODUCCIÓN A OODOO.

### 2.1.- LOS SISTEMAS DE GESTIÓN DE RECURSOS EMPRESARIALES (ERP).

#### 2.1.1.- Las tecnologías en la empresa.

Debido al volumen de información que debe manipular una empresa implica que la gestión sea clave para que la empresa sea competitiva en el mercado.

Hay una preocupación por la capacidad de gestionar de forma óptima todos los datos que se encuentran en la empresa.

Actualmente se ha visto como las Tecnologías de la Información y la Comunicación (TIC<sup>4</sup>) se han convertido en pieza clave de cualquier organización (sin importar su tamaño) e implican adoptar obligadamente un Sistema de Información (SI<sup>5</sup>) que proporcione la información veloz y eficazmente para el correcto funcionamiento de la empresa.

El problema que existe actualmente es que los Sistemas de Información utilizados por las empresas se encuentran repartidos en varias aplicaciones lo que provoca ineficacia de la gestión de los datos así como la duplicidad de los datos. Por tanto es necesario centralizar todos los datos y la información que utiliza la empresa, es decir, gestión óptima de los departamentos que la forman.

También lo que se busca de un sistema de gestión es que sea seguro, robusto y fiable para que el negocio pueda adaptarse a los cambios que se producen a nivel global.

Para lograrlo se puede elaborar un software personalizado adaptado para cubrir las necesidades y requisitos de la organización. O bien la adquisición de un software ya preparado y que abarque la gestión de todos los departamentos con la capacidad adaptativa a las características de la organización.

La alternativa a lo expuesto anteriormente, es mediante un ERP siendo una plataforma que posibilita la gestión y planificación de la empresa a todos los niveles permitiendo con ello la recopilación y centralización de la información siendo utilizado como herramienta fundamental para la toma de decisiones.

#### 2.1.2.- Definición de sistema de gestión (ERP).

Un sistema ERP (Enterprise Resource Planning) es un sistema que pretende unificar todos los datos y procesos de una organización en un sistema unificado. Su objetivo es satisfacer las necesidades de información de una empresa así como ser el apoyo o soporte a la dirección de un negocio más el control del cumplimiento de los objetivos.

4 **TIC:** son el conjunto de procesos, técnicas y dispositivos que integran funcionalidades de almacen, proceso y transmisión de datos mediante la utilización de hardware y software como un medio de sistema informático.

5 **SI:** es un sistema de recursos, datos y actividades que interactúen entre sí con la finalidad de dar soporte a las actividades de una empresa o negocio: optimizar procesos, proporcionar información relevante para la toma de decisiones y obtener ventajas competitivas.

Un ERP es la integración de una base de datos, una aplicación y una interfaz de usuario donde se recopila toda la información manipulada. Permite trabajar de forma rápida y eficaz gracias a que la información se encuentra disponible y actualizada en todo momento, logrando evitar la duplicidad de los datos al tener la información centralizada en una única base de datos compartida.

### 2.1.3.- Principales características de un sistema ERP.

Para que un sistema ERP sea considerado como tal, debe ser:

- Integral, unifica todos los procesos de negocio de la empresa y mantiene los datos de una manera centralizada. Facilita el control de todos los procesos y flujos de trabajo evitando información duplicada o incorrecta por malentendidos entre departamentos o grupos de trabajo.
- Modular, está compuesto por módulos y cada uno está centrado en una parte o en varias partes del negocio. Al dividir las funciones del negocio en módulos éstos se pueden incorporar o no según las necesidades que surjan en el tiempo.
- Adaptable, permite la personalización según las necesidades de la empresa, es decir, para que la configuración, parametrización de los procesos de la empresa adaptados a las características propias de cada entidad de acuerdo con los resultados que se quieran obtener.

A parte también tenemos otras características importantes:

- Que sean sistemas flexibles, capaces de hacer frente a las constantes transformaciones de la empresa.
- Permitan realizar simulaciones de la realidad para una mejor planificación.
- No debe existir ninguna limitación, por tanto todas las áreas de negocio de una empresa pueden ser cubiertas y gestionadas por el ERP.
- Transacciones producidas en los distintos departamentos serán procesados por el sistema de esta forma dispondremos de una base de datos centralizada para toda la organización.
- Están orientados a la interacción con los miembros de la empresa. Realización de simulaciones, etc.
- Ofrecen la posibilidad de realizar seguimientos, mediaciones, informes para evaluar la eventos o incidencias que ocurren en la empresa.
- Dan soporte a las funciones básicas de la actividad del negocio.

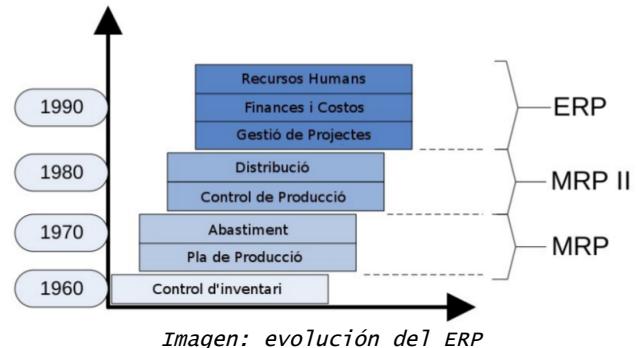


## 2.1.4.- Historia del ERP.

Podríamos decir que la aparición de los ERP tendría sus inicios en la década de los 40. Durante la Segunda Guerra Mundial, la gestión de la flota y la organización de los recursos de los que disponían para planificar las acciones bélicas forzó el desarrollo de los sistemas MRP.<sup>6</sup>

Con el tiempo, las empresas americanas adaptaron los MRP para la gestión y control de las áreas empresariales básicas, para luego evolucionar entre las décadas 60 y 70 para ayudar con la reducción de inventario de materiales y materias primas. Encontramos grandes ordenadores o mainframes dónde el usuario se conecta al servidor a través de consolas.

En los años 80, los MRP pasaron a llamarse MRP II<sup>7</sup>. Estos sistemas ya son capaces de considerar contratiempos y limitaciones que puede sufrir una empresa, acercándose cada vez más a una gestión más próxima a la realidad. Además aparecen los ordenadores personales (PC<sup>8</sup>) con lo que se implanta las arquitecturas de dos capas: cliente – servidor, ahora la aplicación a ejecutar reside en el cliente.



<sup>6</sup> Material Requirements Planning System: sistema que integra las actividades de producción y compras.

<sup>7</sup> Manufacturing Resource Planning: sistema para la planificación eficaz de todos los recursos de una empresa de fabricación.

<sup>8</sup> Personal Computer.



### 2.1.5.- Objetivos y consideraciones al implantar un ERP.

Una empresa interesada en instalar un sistema ERP busca tres objetivos estratégicos:

- Incrementar la productividad.
- Mejorar la calidad del producto o servicio.
- Reducir costes.

Su implementación puede tener otros objetivos:

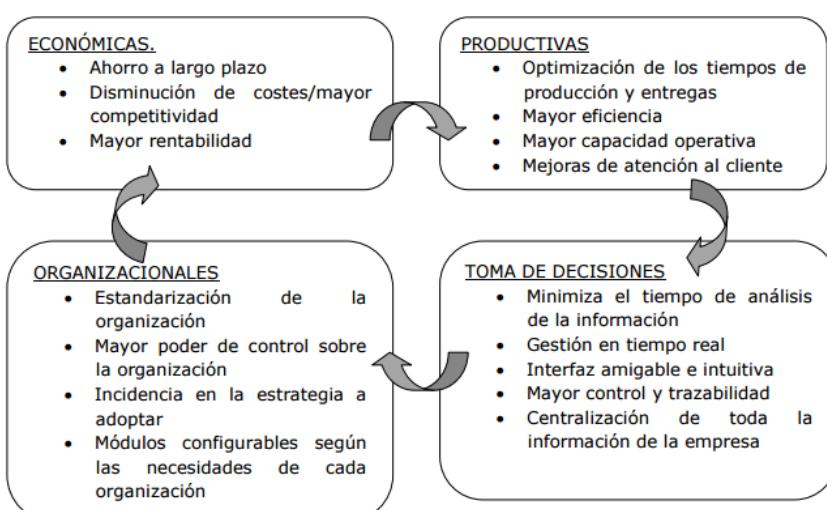
- Optimizar y automatizar procesos.
- Mejorar la toma de decisiones.
- Eliminar datos y operaciones redundantes.

También es importante analizar cuáles son las características de cada empresas y particularizar para cada caso por tanto, antes de implantar un ERP es necesario realizar las siguientes valoraciones:

1. ¿Qué es lo que se quiere conseguir con su implementación?
2. ¿Qué necesitan los usuarios que estarán en contacto con el sistema?
3. ¿Qué recursos tecnológicos necesitaremos?
4. ¿Qué requerimientos necesitará el sistema?
5. ¿Qué procesos harán falta?
6. ¿Qué acciones se tendrán que realizar para la integración del sistema con las aplicaciones existentes?

### 2.1.6.- Ventajas e inconvenientes de los ERP.

Las ventajas que podemos encontrar son las siguientes:



Fuente: <http://repositorio.bib.upct.es/pfc5478.pdf>, figura 6



Las desventajas que en principio tendría en el negocio son:

**ECONÓMICAS**

- Costosos a primera vista
- Beneficio a largo plazo
- Adquisición de software y de hardware.
- Costes indirectos

**ORGANIZACIONALES**

- Riesgo beneficios estratégicos
- Resistencia a compartir información interna entre departamentos.

**OTRAS**

- Tiempo elevado de implementación.
- No existen demasiados expertos en ERP.
- Algunos sistemas ERP pueden ser complicados de usar

Fuente: <http://repositorio.bib.upct.es/pfc5478.pdf>, figura 7

### 3.- REQUERIMIENTOS.

Antes de realizar la instalación tenemos que tener en cuenta los requerimientos de hardware y software así como el sistema operativo seleccionado para proceder a su instalación.

#### 3.1.- REQUERIMIENTOS DE HARDWARE (HW).

Necesitaremos un equipo con **procesador x86-64** o compatible, aunque durante las prácticas en la empresa, no es relevante. Para la memoria **RAM mínimo 512 MB** pero lo recomendado serían 2 GB aunque también dependería de la gestión que realiza el sistema operativo (en GNU/Linux es más eficiente, en principio). Además debemos disponer de **250 MB de espacio en disco duro** como mínimo.

#### 3.2.- REQUERIMIENTOS DE SOFTWARE (SW).

Odoo, está preparado para funcionar en los siguientes sistemas operativos: **MS Windows y GNU/Linux**. La arquitectura de 32 – 64 bits no afectará a la futura instalación del ERP en nuestro equipo.

Deberemos instalar previamente **PostgreSQL versión 9.1.14** aunque también es sabido que con versiones inferiores también funciona (en el caso de GNU/Linux), mientras que en MS Windows viene en el mismo ejecutable, dónde podremos indicar si queremos instalar PostgreSQL o no.

También necesitaremos instalar **Python 2.7.X y sus librerías** correspondientes, en Linux lo realizaremos por terminal mientras que en MS Windows se encuentran incluidos en el paquete de instalación.

#### 3.3.- FASES DEL DESARROLLO DE UN SOFTWARE (módulo en Odoo).

- **ANALISIS:** cuál es el problema a resolver.
- **DISEÑO:** cómo resolver el problema.
- **IMPLEMENTACIÓN:** construir con herramientas nuestro software.
- **PRUEBAS DE IMPLEMENTACIÓN:** que funciona o solventar errores de programación.
- **PUESTA EN FUNCIONAMIENTO:** revisiones, pruebas, información a los usuarios.



### 3.4.- BREVE EXPLICACIÓN DEL CICLO DE VIDA DE UNA BASE DE DATOS.

- **PLANIFICACIÓN ESTRATÉGICA:** esquema inicial del catálogo.
- **ANÁLISIS DEL SISTEMA:** modelo de datos (conceptual)
- **DISEÑO DEL SISTEMA:** esquemas y transacciones.
- **IMPLEMENTACIÓN DEL SISTEMA:** base de datos cargada.
- **CARGA/CONVERSIÓN, VALIDACIÓN Y TRANSFERENCIA:** base de datos y documentación.
- **OPERACION:** servicios.
- **MANTENIMIENTO.**

## 4.- INSTALACIÓN Y CONFIGURACION ODOO.

Sobre la instalación el detalle completo lo podemos encontrar en el ANEXO III del presente documento en dónde se indica la instalación en una distribución GNU/Linux Xubuntu 14.04.02.

## 5.- DISEÑO DE LA BASE DE DATOS “L'Assaig”.

En todo diseño de una base de datos necesitaremos seguir los siguientes pasos a la hora de diseñar y

### 5.1.- TOMA DE DATOS.

El restaurante de nuestro centro llamado “L'Assaig” quiere gestionar mediante un ERP (en nuestro caso Odoo) los clientes, mesas y las reservas que se realicen en el tiempo. Además nos pide:

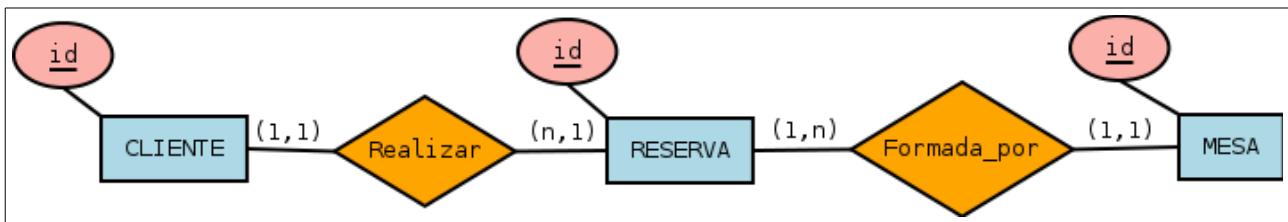
- La posibilidad de imprimir informes: clientes, mesas y reservas.
- Seguridad o reglas de negocio, es decir, creación de un usuario administrador del propio módulo y varios usuarios, junto con los permisos correspondientes.
- Plan de contingencia, en caso de la pérdida de datos.
- Incorporación de varias traducciones del módulo.

## 5.2.- ESQUEMA CONCEPTUAL (ER).

En principio y después de revisar la información recopilada, necesitaremos la creación de tres tablas:

- **Cliente y Mesa:** que servirán como mantenimiento, es decir, será posible dar de alta, modificar y/o borrar.
- **Reserva:** que “relacionará” el cliente con la mesa que ha solicitado.

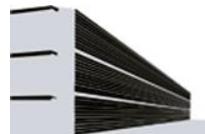
El esquema conceptual quedará de la siguiente forma:



## 5.3.- ESQUEMA LÓGICO (relacional).

Para el esquema lógico (o relacional) tendremos las siguientes tablas, relaciones así como las claves ajenas que dispondrá nuestra futura base de datos:

TABLAS Y RELACIONES
CLIENTE ( <u>id</u> , nombre, apellidos, teléfono, móvil, fecha_alta, observaciones)
RESERVA ( <u>id</u> , tipo_reserva, entrega, comensales, fecha_todo, observaciones, state, <b>cliente_id</b> , <b>mesa_id</b> )
MESA ( <u>id</u> , código, foto_mesa, comensales, estado, observaciones)



**CLAVES AJENAS (DIAGRAMA REFERENCIAL)**

REFERENCIA	¿ACEPTA NULO?	ACCIONES TRAS BORRADO	ACCIÓN TRAS MODIFICACIÓN
RESERVA.cliente_id → CLIENTE.id	No	Por defecto	Por defecto
RESERVA.mesa_id → MESA.id	No	Por defecto	Por defecto

**DOMINIOS**

Tabla	Atributo	Definición	Descripción (opcional)
CLIENTE	Id	Número - Entero	Clave principal
	Nombre	Texto tamaño 20	Nombre del cliente
	Apellidos	Texto tamaño 35	Apellidos del cliente
	Teléfono	Número - Entero Tamaño 9	
	Móvil	Número - Entero Tamaño 9	
	Fecha_alta	Fecha - Date	Fecha corta del alta
	Observaciones	Texto tamaño 100	Cuatro filas y cien columnas
	Situación	Texto - Selección	Distintos valores: activo, baja, moroso. Referido al cliente.
	Cliente_reserva_id	Número - Entero	Clave ajena que apunta a reservas

MESA	Id	Número - Entero	Clave principal
	Código	Texto tamaño 7	
	Ubicación	Texto - Selección	Distintos valores: comedor / terraza.
	Foto_mesa	Imagen - Binary	Guardar imagen de la mesa
	Comensales	Número - Entero	
	Estado	Texto Selección	Distintos valores: disponible, no disponible.
	Observaciones	Texto tamaño 50	Dos columnas
	Mesa_reserva_id	Número - Entero	Clave ajena que apunta a reservas



DOMINIOS (continuación)			
Tabla	Atributo	Definición	Descripción (opcional)
RESERVA	Id	Número – Entero	Clave principal
	Tipo_reserva	Texto - Selección	
	Entrega	Número – Decimal	
	Comensales	Número – Entero	
	Fecha_todo	Fecha – Date	Fecha, hora, minutos y segundos
	Observaciones	Texto tamaño 100	Cuatro columnas
	Cliente_id	Número – Entero	Clave ajena que apunta hacia la tabla cliente
	Mesa_id	Número – Entero	Clave ajena que apunta hacia la tabla mesa
	State	Texto – Selección	Distintos valores: pendiente, servida, anulada

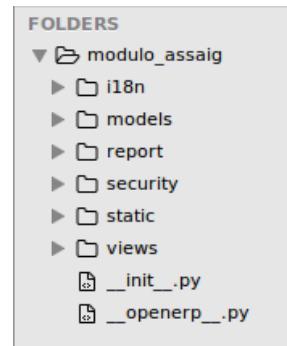


## 6.- CREACION DEL MÓDULO.

### 6.1.- ESTRUCTURA BÁSICA.

El siguiente paso es la creación del módulo, este será una carpeta con el nombre que queramos, en nuestro caso “modulo\_assaig” y dentro de éste tendremos la siguiente estructura:

- “\_\_openerp\_\_.py”, nos mostrará el detalle completo del módulo, autor, versión, detalle, qué archivos tiene que cargar, etc.
- “\_\_init\_\_.py”, que indicará la ruta a seguir a la hora de cargar el módulo, generalmente otra carpeta que se encuentra dentro “modulo\_assaig”, como por ejemplo <<import models>>. Este tipo de archivo lo podemos encontrar en la mayoría de subcarpetas.



El detalle del archivo “\_\_openerp\_\_.py” (versión por defecto) será el siguiente:

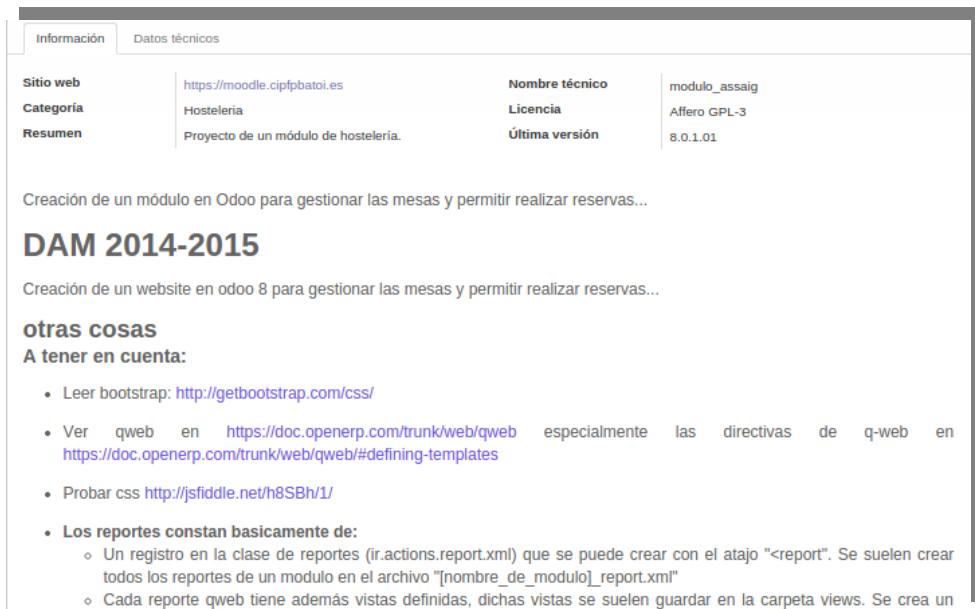
```

__openerp__.py

1 {
2     'name' : "L'ASSAIG",
3     'icon' : "/modulo_assaig/static/src/img/lassaig_logo_modulo.jpg",
4     'image-icon' : "/modulo_assaig/static/src/img/icon.jpg",
5     'version' : "1.01",
6     'author' : "Carlos Huelmo Vaquero",
7     'website' : "https://moodle.cipfpbatoi.es",
8     'category' : "Hosteleria",
9     'summary' : "Proyecto de un módulo de hostelería.",
10    'description' : """
11
12 Creación de un módulo en Odoo para gestionar las mesas
13 y permitir realizar reservas...
14
15 DAM 2014-2015
16 =====
17
18 Creación de un website en odoo 8 para gestionar las mesas
19 y permitir realizar reservas...
20
21 otras cosas
22 -----
23
24 A tener en cuenta:
25 -----
26 * Leer bootstrap: http://getbootstrap.com/css/
27 * Ver qweb en https://doc.openerp.com/trunk/web/qweb especialmente las directivas de q-web en https://doc.op
28 * Probar css http://jsfiddle.net/h8SBh/1/
29
30 * Los reportes constan basicamente de:
31     * Un registro en la clase de reportes (ir.actions.report.xml) que se puede crear con el atajo "<report>"
```



Como resultado, al cargar el módulo (instalado o no) se mostraría la información de la siguiente forma, al tener el módulo en la carpeta “/addons” en Odoo:



Sitio web	<a href="https://moodle.cipfpbatoi.es">https://moodle.cipfpbatoi.es</a>	Nombre técnico	modulo_assaig
Categoría	Hostelería	Licencia	Affero GPL-3
Resumen	Proyecto de un módulo de hostelería.	Última versión	8.0.1.01

Creación de un módulo en Odoo para gestionar las mesas y permitir realizar reservas...

### DAM 2014-2015

Creación de un website en odoo 8 para gestionar las mesas y permitir realizar reservas...

**otras cosas**

**A tener en cuenta:**

- Leer bootstrap: <http://getbootstrap.com/css/>
- Ver qweb en <https://doc.openerp.com/trunk/web/qweb> especialmente las directivas de q-web en <https://doc.openerp.com/trunk/web/qweb/#defining-templates>
- Probar css <http://jsfiddle.net/h8SBh/1/>
- Los reportes constan basicamente de:**
  - Un registro en la clase de reportes (ir.actions.report.xml) que se puede crear con el atajo "<report>". Se suelen crear todos los reportes de un modulo en el archivo "[nombre\_de\_modulo]\_report.xml"
  - Cada reporte qweb tiene además vistas definidas, dichas vistas se suelen guardar en la carpeta views. Se crea un

Tengamos en cuenta que el tamaño de la letra vendría dado por los símbolos “=” o “-” mientras que las viñetas y sus subapartados estarían identificados por el símbolo “\*”. Los links o enlaces a otras páginas automáticamente los reconoce y los marca en color azul (por defecto) recordemos que a la hora de mostrar los datos, información, etc., dependerá del navegador que tengamos instalado en nuestro equipo, por defecto usaremos “google chrome” al ser el más completo.

Después de la descripción, indicamos que la instalación del módulo dependería **“depends”** de que estuviera instalado el módulo base. Los datos a cargar o “data” por orden serían:

- Por lo que se refiere a la seguridad, “security/model\_data.xml”, indicamos la ruta relativa, dónde crearíamos una nueva categoría para indicar después los usuarios y permisos relacionados.
  - Luego daríamos de alta (al cargar el módulo) los grupos y usuarios que tendría dicho módulo en **“security/assaig\_security.xml”**.
  - Finalmente, el archivo CSV indicando **“las listas de controles de acceso”** para los grupos creados anteriormente, las cuales hacen referencia a los permisos de lectura, escritura y modificación (que afectan únicamente al modelo/módulo) dentro del archivo **“security/ir.model.access.csv”**.
  - Para las vistas, indicaremos a Odoo que se encuentran en **“views/assaig\_view.xml”**.
- ```

36      """
37      'depends' : ['base'],
38      'init_xml' : [],
39      'demo_xml' : [],
40      'update_xml' : [],
41      'data' : [
42          'security/model_data.xml',
43          'security/assaig_security.xml',
44          'security/ir.model.access.csv',
45          'views/assaig_view.xml',
46      ],
47      '#css': [
48          '#static/src/css/kanban_view.css',
49      ],
50      'installable' : True,
51      'application' : True,
52      'auto_install' : False,
53  }

```



Por último indicaremos con valores booleanos (true / false) que nuestro módulo **será instalable, de tipo aplicación** pero que no será auto-instalable.

## 6.2.- MODELS (modelos / tablas).

Los encontramos definidos en “**models/assaig.py**”, son los modelos que constará nuestro módulo, que se transformarán en tablas en la base de datos de PostgreSQL, con el mismo nombre, dónde se almacenará la información que posteriormente el/los usuario/-s introduzcan.

- Línea 1: lo primero es indicar el tipo de codificación, es decir, que no de error ante caracteres especiales: acentos, ”ç”, etc.
- Líneas 3 a 15: importar librerías y definir valores iniciales que posteriormente utilizaríamos posteriormente.

```

1 # -*- coding: utf-8 -*-
2
3 from openerp.osv import osv, fields, orm
4 from datetime import date
5 from dateutil import relativedelta
6 import time
7 from dateutil import parser
8 from types import *
9 # Para mensajes de log
10 import logging
11 _logger = logging.getLogger(__name__)
12 # Precisión para la entrega
13 #import openerp.addons.decimal_precision as dp
14 from openerp.exceptions import ValidationError
15 #from openerp import models, fields, api, exceptions
16

```

Como hemos indicado en anteriores puntos, necesitaremos tres clases: cliente, mesa y reserva. Con ello reflejaremos los clientes, las mesas y asociaremos estos dos a las reservas. Para ello la clase “cliente” estará compuesta por:

```

18 class cliente(osv.osv):
19
20     def _nom_apell_fnc(self, cr, uid, ids, fields, args, context):
21         res = {}
22         for r in self.browse(cr, uid, ids, context=context):
23             # res[r.id] = "%s, %s" % (r.apellidos, r.nombre)
24             res[r.id] = "%s %s" % (r.nombre, r.apellidos)
25         return res
26
27     _name = "assaig.cliente"
28     _rec_name = "name"
29     _description = "Tabla Mod. de clientes"
30
31     columns = {
32         'nombre' : fields.char('Nombre',size=20, help="Nombre del cliente.", required=True),
33         'apellidos' : fields.char('Apellidos', size=35, help="Apellidos del cliente.", required=True),
34         'telefono' : fields.integer('Teléfono', size=9,
35                                     help="Ejemplo >>> Teléfono 999999999", required=True),
36         'movil' : fields.integer('Móvil', size=9, help="Ejemplo >>> Móvil 999999999", required=True),
37         'fecha_alta' : fields.date('Alta'),
38         'observaciones' : fields.text('Observaciones', size_row=4, size_column=100, help="Anotaciones sobre el cliente"),
39         'cliente_reserva_id' : fields.one2many('assaig.reserva', 'cliente_id', 'Reserva cliente'),
40         'name' : fields.function(_nom_apell_fnc, type="char", size=140, string="Nombre y Apellidos", readonly=True),
41         'situacion' : fields.selection([('activo','Activo'),('baja','Baja')],
42   ('moroso','Moroso')), string="Situación del cliente"),
43     }
44
45     defaults = {
46         'fecha_alta' : lambda *a : time.strftime("%Y-%m-%d"),
47         'situacion' : 'activo',
48     }
49
50     _sql_constraints = [
51         ('nombre_unique', 'UNIQUE(nombre,apellidos)', 'El cliente ya existe'),
52     ]
53
54     _order = "apellidos, nombre"
55

```

Explicamos el código del modelo “cliente”:

- Línea 18, nombre de la clase y entre paréntesis hereda de la clase padre “osv.osv”.
- Líneas 20 a 25, “**\_nom\_apell\_fnc**” función que concatena los valores de los campos “nombre” y “apellidos”.
- Línea 27, **\_name = “assaig.cliente”**, es el nombre de la tabla que se creará y almacenará los valores de los registros en la base de datos Postgre. *Es el identificador interno para el modelo que estamos creando.*
- Línea 28, **\_rec\_name = ”name”**, este indica el campo a utilizar como descripción del “registro” cuando se referencia desde campos relacionados así como para una relación. Por defecto se utiliza el campo “name” el cuál es comúnmente encontrado en los modelos. Pero este atributo nos permite usar otro campo para dicho propósito.
- Línea 29, descripción de la tabla que podemos reconocer fácilmente.
- Líneas 32 a 43, son los campos que contendrá la tabla o modelo y posteriormente en la base de datos, los tipos de datos que podemos encontrar son: **char** (para texto números, letras, caracteres especiales), **integer** (números), **date** (fecha, horas, minutos y segundo), **text** (parecido a char), **float** (decimales), **selection** (desplegable con varios valores).

Los atributos que podemos indicar para cada campo son:

- “**size**”, tamaño que podrá almacenar dicho campo.
- “**help**”, texto de ayuda que aparecerá al situarnos dentro de dicho campo o cuando el cursor del ratón esté encima en la vista (por ejemplo para la vista tree o árbol).
- “**required**”, indica que si el valor de dicho campo no puede ser nulo por tanto deberá contener datos.
- “**size\_row**” y “**size\_column**”, utilizados para el campo de tipo “text” donde indicaremos las filas y las columnas respectivamente.
- “**string**”, nombre del campo (para el tipo “selection”) aunque si ponemos el texto en comillas simples sin indicar nada más, Odoo lo reconoce como el título/nombre del campo.
- “**readonly**”, sirve para que dicho campo no pueda ser modificado por el usuario y muestre un determinado valor, en este modelo, mostrará nombre y apellidos de la función que hemos descrito más arriba. **Si en el modelo indicamos campos de sólo lectura, afectará a las vistas, por lo que en principio sería recomendable hacerlo desde las vistas para no realizar modificaciones en el código python posteriormente.**

- Líneas 45 a 48, cada vez que demos de alta un registro con sus valores, los campos que están indicados aquí mostrarán por defecto los valores que nosotros como programadores queremos, por ejemplo, el campo “**fecha\_alta**” mostrará mediante un simple cálculo la fecha que tomará del sistema, mientras que el campo “**situación**” automáticamente mostrará el valor “activo”.
- Líneas 50 a 52, son las restricciones SQL que aplicaremos para controlar que el nombre y los apellidos no se dupliquen al insertarlos en la base de datos Postgres.



- Línea 54, indicamos que el orden en que se muestre los registros almacenados sea primero por apellidos y después por nombre, al no indicar nada, por defecto Odoo indica a Postgres que es de forma “ascendente”. **Como veremos después, en las vista “tree” es posible ordenar por el campo que queramos.**
- Línea 39, para establecer la relación con el modelo “reserva”, necesitaremos otro campo llamado “cliente\_reserva\_id” mediante un “one2many”, los atributos que tendrá la relación (por orden) son: el modelo reserva, un campo del modelo (cliente\_id) y el texto para éste.

Para el modelo “mesa” tendremos el siguiente código python:

```

57 class mesa(osv.osv):
58
59     def _codigo_plazas_fnc(self, cr, uid, ids, fields, args, context):
60         res = {}
61         for r in self.browse(cr, uid, ids, context=context):
62             res[r.id] = "%s %sp" % (r.codigo, r.comensales)
63         return res
64
65     _name = "assaig.mesa"
66     _rec_name = "name"
67     _description = "Tabla Mod. de mesas"
68
69     _columns = {
70         'codigo' : fields.char('Código', size=7, help="Ejemplo >>> MESA-XX", required=True),
71         'ubicacion' : fields.selection([
72             ('comedor','Comedor'),('terraza','Terraza')], 'Ubicación', required=True),
73         'foto_mesa' : fields.binary('Foto', help="Añadir imagen según comensales.", required=False),
74         'comensales' : fields.integer('Comensales',size=2, help="Número de comensales\nMínimo 1 persona.", required=True),
75         'estado': fields.selection([('disponible','Disponible'),('no disponible','No disponible')], 'Estado', required=True),
76         'observaciones' : fields.text('Observaciones', size_row=2, size_col=50),
77         'mesa_reserva_id' : fields.one2many('assaig.reserva','mesa_id','Reserva mesa'),
78         'name' : fields.function(_codigo_plazas_fnc,type="char", size=11, string="Referencia", readonly=True),
79     }
80
81     _defaults = {
82         'codigo' : 'MESA-0',
83         'comensales' : 4,
84     }
85
86     _sql_constraints = [
87         ('codigo_unique','UNIQUE(codigo)','La mesa ya existe.'),
88     }

```

Explicamos:

- Líneas 59 – 63, función que guardará en otro campo el código y el número de comensales.
- Línea 73, campo que almacenará la imagen de la mesa que demos de alta (tipo binary).
- Línea 74, campo de tipo integer (números) para indicar los comensales de la mesa.
- Línea 77, relación con el modelo reservas, a partir de una relación one2many, se indica por orden: el nombre del modelo / tabla, nombre de campo dentro del modelo “reserva” y el texto que tendrá dicho campo.
- Líneas 81 – 84, valores por defecto que tendrán los campos código y comensales.
- Líneas 86 – 88, restricción SQL para el campo código para evitar duplicidades.

Finalmente para el modelo reserva, tendremos el siguiente código:



```

89
90  class reserva(osv.osv):
91
92      def anula_reserva_f(self, cr, uid, ids, context=None):
93          cambiar_valor = 'anulada'
94          reserva_id = self.pool('assaig.reserva').search(cr,uid,[('state','=', 'pendiente')])
95          self.write(cr,uid,reserva_id[0],{'state': cambiar_valor}, context = context)
96          return True
97
98      def activa_reserva_f(self, cr, uid, ids, context=None):
99          cambiar_valor = 'pendiente'
100         reserva_id = self.pool('assaig.reserva').search(cr,uid,[('state','=', 'anulada')])
101         self.write(cr,uid,reserva_id[0],{'state': cambiar_valor}, context = context)
102         return True
103
104     def servir_reserva_f(self, cr, uid, ids, context=None):
105         cambiar_valor = 'servida'
106         reserva_id = self.pool('assaig.reserva').search(cr,uid,[('state','=', 'pendiente')])
107         self.write(cr,uid,reserva_id[0],{'state': cambiar_valor}, context = context)
108         return True
109
110     def cambia_reserva_f(self, cr, uid, ids, context=None):
111         cambiar_valor = 'pendiente'
112         reserva_id = self.pool('assaig.reserva').search(cr,uid,[('state','=', 'servida')])
113         self.write(cr,uid,reserva_id[0],{'state': cambiar_valor}, context = context)
114         return True
115

```

Explicamos:

- Líneas 92 – 114, serie de cuatro funciones que cambiarán el valor del campo “state” (estado) del modelo reserva según la condición de igualdad, es decir, si tiene el valor “pendiente” cambiará a “anulada”. Dichas funciones serán llamadas desde botones en la vista formulario de reservas mediante el atributo **name=”anula\_reserva\_f”** como veremos en el apartado de las vistas.

```

116     _name = "assaig.reserva"
117     _rec_name = "mesa_id"
118     _description = "Tabla Mod. de reservas"
119
120     _columns = {
121         'tipo_reserva' : fields.selection([
122             ('en persona','En persona'),
123             ('movil','Móvil'),
124             ('telefono','Teléfono'),
125             ('email','Email')]),
126         'Tipo de Reserva' ,required=True),
127         'entrega' : fields.float('Entrega',digits=(6,2), help="Euros entregados por el cliente."),
128         #'entrega' : fields.float('Entrega',digits_compute=dp.get_precision('Entrega'), help="Euros entregados por el cliente."),
129         'comensales' : fields.integer('Comensales',digits=(2), help="Número de comensales.\nMínimo 2.", required=True),
130         'fecha_todo' : fields.datetime('Fecha reserva', required=True),
131         'observaciones' : fields.text('Observaciones', size_row=5, size_column=100),
132         'cliente_id' : fields.many2one('assaig.cliente','Cliente', select=True, required=True,
133             domain="[('situacion','!=', 'moroso'),('situacion','=','activo')]]"),
134         'mesa_id' : fields.many2one('assaig.mesa','Mesa', select=True, required=True, domain="[( 'estado','=', 'disponible')]]"),
135         'state' : fields.selection([('pendiente', 'Pendiente'),
136             ('servida','Servida'),('anulada','Anulada')], 'Estado', required=True),
137     }
138
139     _defaults = {
140         'comensales' : 2,
141         'state' : 'pendiente',
142     }
143
144     _sql_constraints = [
145         ('cliente_id_unique','UNIQUE(cliente_id,mesa_id,fecha_todo)', 'La reserva ya existe.\nFecha idéntica'),
146     ]
147
148     _order = "fecha_todo desc"

```



Siguiendo con la explicación del código del modelo “reserva”, tenemos:

- Línea 117, el campo a utilizar como descripción del “registro” cuando se referencia desde campos relacionados así como para una relación será en este caso “**mesa\_id**” porque de lo contrario nos mostraría el OUID del registro correspondiente o el identificador correspondiente (un número). Por defecto se utiliza el campo “name” el cuál es comúnmente encontrado en los modelos. Pero este atributo nos permite usar otro campo para dicho propósito.
- Línea 127, campo de tipo float utilizado para almacenar la entrega que realiza el cliente, tendremos el atributo “**digits**” con dos valores el “6” indica el tamaño de la parte entera mientras que el “2” indica el tamaño para los decimales.
- Línea 128, podemos observar otra forma de “limitar” con mayor exactitud el atributo “**digits\_compute**” con lo que indicamos que el propio campo se aproxime más y lograr mejorar el formato del valor que será almacenado.
- Línea 130, el campo “**fecha\_todo**” de tipo “datetime”, almacenará la fecha (corta) incluyendo horas, minutos y segundo.
- Línea 132, “**cliente\_id**” campo que relaciona la clave ajena del modelo cliente con la de reservas mediante una relación “many2one”, es necesario que la relación sea en dos direcciones, es decir “one2many” desde cliente y “many2one” desde reserva a cliente para que luego se pueda mostrar los registros en las vistas. También tenemos un atributo “**domain**” el cual se encarga de “filtrar” desde el propio modelo que sólo se muestren aquellos clientes que cumplan las dos condiciones indicadas. **Al igual que pasaba con el atributo “readonly” si ya limitamos desde el modelo estaremos limitando la funcionalidad para mostrar los registros ya que desde las vistas también podemos hacerlo.**
- Línea 144 – 146, con la restricción SQL se pretende no duplicar el cliente, la mesa ni la fecha de reserva, pero sólo logramos que la reserva no se duplique únicamente por la fecha.
- Línea 148, indicamos que las reservas se ordenen de forma descendente, es decir, desde la fecha más reciente a la más antigua.

Por lo que se refiere al modelo “configuración”:



```

150 class configuracion(osv.osv):
151
152     def _total_comensales(self, cr, uid, ids, fields, args, context):
153         res = {}
154         for r in self.browse(cr, uid, ids, context=context):
155             res[r.id] = "%s" % (r.superficie*4)
156         return res
157
158     _name = "assaig.configuracion"
159     _rec_name = "name"
160     _description = "Tabla Mod. de la configuracion"
161
162     _columns = {
163         'name' : fields.char('Nombre', size=50, required=True),
164         'superficie' : fields.integer('Capacidad en mesas', help="Mesa-base = 4 comensales"),
165         'total_comensales' : fields.function(_total_comensales, string='Max. Comensales', type="integer", readonly=True),
166         'estado' : fields.boolean('Activo'),
167     }
168
169     _defaults = {
170         'name' : 'Default',
171         'superficie' : 06,
172         'estado' : True,
173     }
174
175     _sql_constraints = [
176         ('name_unique', 'UNIQUE(name)', 'Esa configuración ya existe.')
177     ]

```

Explicamos:

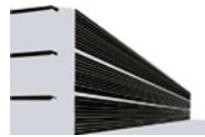
- Líneas 152 – 156, función que nos devuelve los comensales según el número de mesas que indiquemos.
- Línea 165, campo que “apunta” a la función “\_total\_comensales” que calcula automáticamente cuántos comensales tendremos en la superficie destinada a la restauración.
- Líneas 169 – 173, indicamos los valores por defecto, para el campo 'Nombre' (utilizado como clave principal) aparecerá la palabra “Default”, para 'superficie' el valor numérico “6” y para el estado, el valor booleano “True”.

Al final del archivo **assaig.py** indicamos que cargue las clases. Entonces al instalar el módulo Odoo creará las tablas a partir de los modelos indicados en dicho archivo. Igualmente podremos comprobar que se han creado dichas tablas desde terminal o bien utilizando “pgadminIII”.

```

179 cliente()
180 mesa()
181 reserva()
182 configuracion()
183
184

```



### 6.3.- VISTAS (views).

Como hemos indicado anteriormente, se encuentran dentro de un archivo XML en la subcarpeta “views/assaig\_view.xml”, lo primero que hemos hecho ha sido introducir primero las vistas search, tree y finalmente los form para cada modelo a fin de seguir un orden y que sea fácil de encontrar cada vista, al final encontraríamos los elementos del menú.

#### 6.3.1.- Vista “search”.

Explicamos el código para la búsqueda para cliente:

- Línea 14, indicamos su “id” y la referencia al modelo a utilizar “ir.ui.view”.
- Línea 16, hacemos referencia al modelo el cuál queremos que tenga dicha vista. En este caso “assaig.cliente” que es el nombre del modelo indicado en “models/assaig.py”.
- Línea 17, indicamos en “type” que es de tipo “search” (tree para árbol y form para formulario).
- Líneas 20 – 23, dentro de la etiqueta (o tag) search, añadiremos los nombres de los campos que queremos que filtre y mediante el atributo “select” con valor “1” para búsquedas normales (“2” para avanzadas).

```

14      <record id="cliente_filter" model="ir.ui.view">
15          <field name="name">view_cliente_filter</field>
16          <field name="model">assaig.cliente</field>
17          <field name="type">search</field>
18          <field name="arch" type="xml">
19              <search string="Buscar cliente">
20                  <field name="nombre" select="1" />
21                  <field name="apellidos" select="1" />
22                  <field name="fecha_alta" select="1" />
23                  <field name="situacion" select="1" />
24              </search>
25          </field>
26      </record>

```

Explicamos el código para la búsqueda para mesa:

```

29      <record id="mesa_filter_comedor" model="ir.ui.view">
30          <field name="name">view_mesa_filter</field>
31          <field name="model">assaig.mesa</field>
32          <field name="type">search</field>
33          <field name="arch" type="xml">
34              <search string="Buscar mesas">
35                  <field name="codigo" select="1" />
36                  <field name="ubicacion" select="1" />
37                  <field name="estado" select="1" />
38                  <!--
39                  <filter name="ubicacion" string="Comedor" domain=" [('ubicacion','=' , 'comedor')]" /> -->
40                  <filter name="ubicacion2" string="Terraza" domain=" [('ubicacion','=' , 'terraza')]" />
41
42          </search>
43      </field>
44  </record>

```

- Línea 40, a parte de indicar el nombre de los campos (ya definidos en el modelo) podemos hacer un “filtrado” desde la propia vista search, mediante el atributo “name” indicado dentro de la etiqueta “filter” (también podría ser record).



Esta vista podemos verla en Odoo, aparecerá la palabra “Terraza” y sólo nos mostrará las mesas con dicha ubicación indicada en el “domain”. Recordemos que la vista está asociada a un **menuitem** del menú general del módulo y cuyo atributo “action” apunta a un “ir.actions.act\_window”, es decir, que al clicar sobre dicho elemento, éste tendrá una acción que cargará la vista tree (u otras vistas indicadas) y en la cual, mediante el dominio indicado en dicha vista, filtrará. Ahora bien, si quitamos la palabra “terraza” nos mostrará todas las mesas independientemente de su ubicación.



The screenshot shows a search bar at the top with the text "Terraza" and a magnifying glass icon. Below the search results, a card displays information for a table named "MESA-09". The card includes the following details:

- MESA-09**
- Comensales: 4
- Ubicación: Terraza
- Estado: No disponible

At the bottom of the card are two buttons: "Borrar" (Delete) and "Editar" (Edit).

Para la búsqueda (search) de “reservas” tendremos el mismo código, ya comentado para la vista search de cliente.

### 6.3.2.- Vistas form.

El formulario para el modelo “assaig.cliente” tendremos el siguiente código:

```

62 <record model="ir.ui.view" id="cliente_form">
63     <field name="name">view_cliente_form</field>
64     <field name="model>assaig.cliente</field>
65     <field name="type">form</field>
66     <field name="arch" type="xml">
67         <form string="Cliente">
68             <sheet class="my_background_sheet">
69                 <newline/>
70                 <h1><field name="name"/></h1>
71                 <newline/>
72                 <group string="Datos del cliente:" col='4' colspan='2'>
73                     <field name="nombre" width="200" height="200"/>
74                     <field name="apellidos" />
75                     <field name="telefono" />
76                     <field name="movil" />
77                     <field name="fecha_alta" readonly="1" />
78                     <field name="situacion" />
79                     <newline/>
80                     <separator string="Información adicional:" colspan="4"/>
81                     <newline/>
82                     <notebook colspan="4">
83                         <page string="Reservas realizadas" >
84                             <field name="cliente_reserva_id" widget="one2many_list" readonly="1" edit="false">
85                                 <tree string="Reservas" colors="#006400:state=='pendiente';#A52A2A:state=='anulada';
86                                     #0000FF:state=='servida' fonts="bold:state=='pendiente'" class="my_background_sheet">
87                                     <field name="mesa_id"/>
88                                     <field name="tipo_reserva"/>
89                                     <field name="entrega"/>
90                                     <field name="comensales"/>
91                                     <field name="fecha_todo"/>
92                                     <field name="state"/>
93                             </tree>
94                         </field>
95                     </page>
96                 </group>
97             </sheet>
98         </form>
99     </field>
100 </record>
```

Explicamos:

- Línea 64, indicamos que la vista se realice a partir del modelo “assaig.cliente”.
- Línea 84, dentro del campo que establece la relación con el modelo “assaig.reservas” indicamos con el atributo **widget=”one2many\_list”** que es un campo relacionado, de sólo lectura (**readonly**) y no editable (**edit=”false”**).
- Línea 85 – 92, dentro del campo “cliente\_reserva\_id” introducimos el esquema de una



vista tree, el cuál mediante el atributo colors e indicando con colores en hexadecimal el valor del campo “**state**” y mediante el atributo **fonts** podemos darle el formato de fuente, en este caso en negrita (**bold**).

```

95 ▼
96
97
98
99 ▼
100
101
102
103
104
105
106
107
108
109
110
111
112
113
      <page string="Observaciones">
        <field name="observaciones" colspan="4" nolabel="1"
               placeholder="Escriba aquí sus observaciones..."/>
      </page>
      <page string="Histórico">
        <separator string="Creado por:"/>
        <field name="create_uid" readonly="1"/>
        <field name="create_date" readonly="1"/>
        <separator string="Última modificación:"/>
        <field name="write_uid" readonly="1"/>
        <field name="write_date" readonly="1"/>
      </page>
    </notebook>
  </group>
</sheet>
</form>
</field>
</record>

```

En la siguiente parte del código de la vista formulario tenemos:

- Mediante las etiquetas **<page>** indicamos las pestañas que aparecerán.
- Línea 97, utilizando el atributo “**placeholder**” mostrará un texto de ayuda dentro del campo de tipo texto para el registro observaciones.
- Líneas 100 – 106, aparte de los registros que se crearán al cargar el módulo a partir del archivo **assaig.py** que indica los modelos a crear, por defecto también hay una serie de campos que se crean automáticamente como son: **create\_uid** (nos indicará el usuario que ha creado los valores del registro), **create\_date** (fecha completa de la creación), **write\_uid** (qué usuario ha modificado los valores del registro) y **write\_date** (fecha completa de la modificación realizada).

Como ejemplo tenemos la siguiente captura de pantalla para mostrar lo explicado anteriormente.

**Jaime Casal Llopis**

**Datos del cliente:**

|          |            |                       |              |
|----------|------------|-----------------------|--------------|
| Nombre   | Jaime      | Apellidos             | Casal Llopis |
| Teléfono | 962213456  | Móvil                 | 643212134    |
| Alta     | 13/05/2015 | Situación del cliente | Activo       |

**Información adicional:**

| Reservas realizadas |                 | Observaciones | Histórico             |
|---------------------|-----------------|---------------|-----------------------|
| Mesa                | Tipo de Reserva | Entrega       | Comensales            |
| MESA-01 4p          | En persona      | 0,00          | 4 09/01/2015 12:17:33 |
| Estado              |                 |               |                       |
| Pendiente           |                 |               |                       |

El formulario para el modelo “assaig.mesa” tendremos el siguiente código:

```

115      <record model="ir.ui.view" id="mesa_form">
116          <field name="name">view_mesa_form</field>
117          <field name="model">assaig.mesa</field>
118          <field name="type">form</field>
119          <field name="arch" type="xml">
120              <form string="Mesas">
121                  <!--
122                      <header>
123                          <div style="color:blue;" align="center"><p>Esta es la cabecera...</p></div>
124                  </header>
125              </-->
126              <sheet class="my_background_sheet">
127                  <newline/>
128                  <h1>Datos de la Mesa</h1>
129                  <newline/>
130                  <separator string="Foto:" />
131                  <field name="foto_mesa" widget="image" class="oe_avatar oe_left"/>
132                  <group col="2" colspan="1">
133                      <field name="name" />
134                  </group>
135                  <group string="Detalle:" col='6' colspan='1'>
136                      <field name="codigo" />
137                      <field name="ubicacion" />
138                      <field name="comensales" />
139                      <field name="estado" />
140                  </group>
141                  <notebook colspan="4">
142                      <page string="Observaciones" col="1" colspan="1">
143                          <field name="observaciones" placeholder="Escriba aquí sus observaciones..."/>
144                  </page>

```

Explicamos:

- Líneas 122 – 124, mediante el “tag” **<header>**, podemos poner el encabezado a nuestra vista formulario, tenemos la posibilidad de añadir botones, imágenes, texto... Pero la cabecera se utiliza más que nada para el **“workflow”** es decir el ciclo que seguiría nuestro formulario como ejemplo, el modulo de ventas que trae Odoo.

```

153          </notebook>
154      </sheet>
155      <!-- <div class="oe_chatter" align="center">Esto es el pie...</div> -->
156  </form>
157 </field>
158 </record>
159

```

- Línea 155, fuera de la etiqueta **<sheet>** podemos indicar el pie del formulario, para ello utilizariamos un **<div>** con una clase CSS **“oe\_chatter”** que tiene Odoo, añadiendo las mismas cosas como en la cabecera. Pero utilizado para el **“workflow”**.



El formulario para el modelo “assaig.reserva” tendremos el siguiente código:

```

161      <record model="ir.ui.view" id="reserva_form">
162          <field name="name">view_reserva_form</field>
163          <field name="model">assaig.reserva</field>
164          <field name="type">form</field>
165          <field name="arch" type="xml">
166              <form string="Reservas">
167                  <sheet class="my_background_sheet">
168                      <newline/>
169                      <h1>Datos de la Reserva</h1>
170                      <newline/>
171                      <group col='4' colspan='2'>
172                          <field name="cliente_id" options="{'no_open':True,'no_create_edit':True}" placeholder="Seleccione un cliente"/>
173                          <field name="mesa_id" options="{'no_open':True,'no_create_edit':True}" placeholder="Seleccione una mesa"/>
174                          <field name="tipo_reserva" placeholder="-- Elija un tipo --"/>
175                          <field name="entrega" />
176                          <field name="comensales" />
177                          <field name="fecha_todo" />
178                      </group>
179                      <group string="Estado de la reserva" col='6' colspan='1'>
180                          <field name="state" readonly="1"/>
181                          <div class="oe_button_box oe_center oe_list_buttons">
182                              <button class="oe_button my_oe_highlight_green oe_read_only" string="Activar" name="activa_reserva_f" type="object" states="anulada"/>
183                              <button class="oe_button oe_highlight oe_read_only" string="Anular" name="anula_reserva_f" type="object" states="pendiente"/>
184                              <button class="oe_button my_oe_highlight_blue oe_read_only" string="Servir" name="servir_reserva_f" type="object" states="pendiente"/>
185                              <button class="oe_button my_oe_highlight_orange oe_read_only" string="Revertir" name="cambia_reserva_f" type="object" states="servida"/>
186                          </div>
187                      </group>

```

Explicamos el código más relevante:

- Línea 172, el campo “**cliente\_id**” es una clave ajena y que mantiene una relación **many2one** con el modelo cliente, mediante el atributo **options** podemos indicar que no se pueda acceder a la vista formulario del modelo cliente, así como restringir la creación y modificación.
- Línea 173, lo mismo pero para el campo “**mesa\_id**”.
- Líneas 181 -185, tendremos cuatro botones para cambiar el valor del campo “**states**”, con el atributo **name** indicamos que al pulsar el botón, Odoo vaya al modelo y busque la función “**activa\_reserva\_f**” (por ejemplo). Con el **atributo object** indicamos que es de tipo objeto para que acceda al modelo correspondiente. Por otra parte mediante CSS los cuatro botones tendrán colores diferentes y con la clase “**oe\_read\_only**” los botones no se mostrarán cuando creamos una reserva o la modifiquemos, siendo una forma de controlar las acciones del usuario. Con el atributo **states** indicamos que el botón no aparezca si el valor coincide con el campo.

**Datos de la Reserva**

|                             |                         |               |                     |
|-----------------------------|-------------------------|---------------|---------------------|
| Cliente                     | Jaime Casal Llopis      | Mesa          | MESA-01 4p          |
| Tipo de Reserva             | Jaime Casal Llopis      | Entrega       | 0,00                |
| Comensales                  | Paquita Fustel Pico     | Fecha reserva | 09/01/2015 12:17:33 |
| Estado de la reserva        | Francisco Garcia Garcia |               |                     |
| Estado                      | Alfonso Huelmo Vaquero  |               |                     |
| Observaciones               | Carlos Huelmo Vaquero   |               |                     |
| Hist.                       | Juan Jaime Llopis Casal |               |                     |
|                             | Justo Lopez Villa       |               |                     |
|                             | Buscar más...           |               |                     |
| <b>Creado por:</b>          |                         |               |                     |
| Administrator               |                         |               |                     |
| 09/06/2015 12:18:00         |                         |               |                     |
| <b>Última modificación:</b> |                         |               |                     |
| Administrator               |                         |               |                     |
| 09/06/2015 12:18:00         |                         |               |                     |



El formulario para el modelo “assaig.configuracion” tendremos el siguiente código:

No hay nada que resaltar, sólo que la idea de la vista era que mediante la configuración pudiéramos controlar las mesas que podríamos dar de alta en el establecimiento.

```
<record model="ir.ui.view" id="configuracion_form">
    <field name="name">view_configuracion_form</field>
    <field name="model">assaig.configuracion</field>
    <field name="type">form</field>
    <field name="arch" type="xml">
        <form string="Configuraciones">
            <sheet class="my_background_sheet">
                <newline/>
                <h1>Detalle de la configuración</h1>
                <newline/>
                <group col='2' colspan='2'>
                    <field name="name" />
                    <field name="superficie" />
                    <field name="total_comensales" />
                    <field name="estado" />
                </group>
            </sheet>
        </form>
    </field>
</record>
```

### 6.3.3.- Vistas tree.

La código de la vista tree para el modelo “assaig.cliente”:

```
229      <record model="ir.ui.view" id="cliente_tree">
230          <field name="name">view_cliente_tree</field>
231          <field name="model">assaig.cliente</field>
232          <field name="type">tree</field>
233          <field name="arch" type="xml">
234              <tree editable="top" delete="true" string="Clientes" colors="gray:situacion=='moroso';green:situacion=='activo';
red:situacion=='baja'" fonts="bold:situacion!='activo' or situacion=='activo'" class="my_background_row2">
235                  <field name="apellidos"/>
236                  <field name="nombre" />
237                  <field name="telefono" />
238                  <field name="movil" />
239                  <field name="observaciones" placeholder="Notas sobre el cliente"/>
240                  <field name="fecha_alta" readonly="1"/>
241                  <field name="situacion"/>
242              </tree>
243          </field>
244      </record>
```

Explicamos:

- Línea 234, con el atributo **editable="top"**, podemos añadir clientes desde la propia vista y siempre empezando por arriba. Es posible asignar colores con letra según el valor del campo situación mediante el **atributo colors**. Podemos además indicar que esté en negrita según las condiciones que indiquemos en el **atributo fonts**. En las otras vistas hemos hecho algo parecido pero probando otras “fuentes” y “colores”.

Clientes						
Guarda o Descartar						
	Apellido	Nombre	Teléfono	Observaciones	Alta	Situación del cliente
<input checked="" type="checkbox"/>	Juarez Fernandez	Sergio	965334556	701898976 Notas sobre el cliente	10/06/2015	Activo
<input checked="" type="checkbox"/>	Antón Bordiu	Luis	992323141	609001101	18/05/2015	Moroso
<input checked="" type="checkbox"/>	Casal Llopis	Jaime	962213456	643212134	13/05/2015	Activo
<input checked="" type="checkbox"/>	Casas Llopis	Fausto	933216590	621894532	13/05/2015	Baja



#### 6.3.4.- Vista calendar.

Dicha vista sólo la tendremos para el modelo “**assaig.reserva**”, lo declaramos mediante el siguiente código:

```

297      <record model="ir.ui.view" id="reserva_calendar">
298          <field name="name">view_reservas_calendar</field>
299          <field name="model">assaig.reserva</field>
300          <field name="type">calendar</field>
301          <field name="arch" type="xml">
302              <calendar string="Calendario" color="cliente_id" date_start="fecha_todo" mode="week">
303                  <field name="cliente_id" />
304                  <field name="fecha_todo" />
305              </calendar>
306          </field>
307      </record>
```

Explicamos:

- Línea 300, indicamos que el “**type**” es calendar.
- Línea 302 – 305, dentro de la etiqueta **<calendar>** indicaremos mediante el atributo color que nos dibuje el valor del campo “**cliente\_id**” (el cuál contiene el nombre y los apellidos del cliente) también *necesitaremos un campo de tipo date o datetime* (declarado previamente en el modelo) para indicarlo en el atributo date\_start. Con el atributo mode indicamos que al cargar la vista, nos muestre la semana (week) aunque puede tomar otros valores como: day y month. Tendríamos que poner con el atributo date\_end el campo “**fecha\_todo**” para indicar la finalización de la reserva, en caso de no ponerlo, Odoo añade una hora de “duración”. **Pero tenemos el problema de que mostraría las reservas independientemente de su estado (anulada, reservada, pendiente) por lo que habría que realizar un “domain” para mostrar sólo las pendientes.**

The screenshot shows the Odoo calendar interface for June 2015. The main calendar grid highlights a reservation on June 11, 2015, from 11:17 to 12:17, which is associated with the user Carlos Huelmo Vaquero. The reservation is shown as a yellow block on the calendar. The sidebar on the left shows the week starting on Monday, and the bottom right shows a list of users.

lun	mar	mié	jue	vie	sáb	dom
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30					



### 6.3.5.- Vistas kanban.

Las utilizaremos a la hora de mostrar la distribución de las mesas para el comedor y para la terraza indicando que el modelo a utilizar sea “assaig.reseva”.

```

310      <record id="mesa_kanban_1" model="ir.ui.view">
311          <field name="name">view_mesas_kanban1</field>
312          <field name="model">assaig.mesa</field>
313          <field name="type">kanban</field>
314          <field name="arch" type="xml">
315              <kanban>
316                  <field name="codigo" />
317                  <field name="comensales" />
318                  <field name="foto_mesa" />
319                  <field name="ubicacion"/>
320                  <templates>
321                      <t t-name="kanban-box">
322                          <div class="oe_product_vignette my_kanban">
323                              <a type="open">
324                                  
326                              </a>
327                          <div class="oe_product_desc">
328                              <h4>
329                                  <a type="edit" class="my_kanban_text_main">
330                                      <field name="codigo"/>
331                                  </a>
332                              </h4>
333                              <p></p>
334                              <ul>
335                                  <li class="my_kanban_text">Comensales: <field name="comensales"/></li>
336                                  <li class="my_kanban_text">Estado: <field name="estado"/></li>
337                                  <li class="my_kanban_text">Ubicación: <field name="ubicacion"/></li>
338                              </ul>
339                          </div>
340                      </t>
341                  </templates>
342              </kanban>
343          </field>
344      </record>
345

```

Explicamos:

- Líneas 315 – 319, indicaremos mediante la etiqueta <kanban> que la vista se compondrá de los siguientes campos (aunque no sería necesario ponerlos).
- Líneas 320 – 342, template o plantilla que mostraría la cuadrícula según los campos que hemos incluido dentro. Mediante el uso de etiquetas <t> se aplicarían el diseño. Con el atributo **type="open"** indicamos que al clicar sobre la foto que se encuentra en la ruta indicada en el atributo t-att-src podemos acceder a la vista formulario del modelo mesa, mientras que para el campo “**codigo**” tiene un atributo **type="edit"**, es decir, que al clicar en el valor de dicho campo accederíamos a la vista formulario para modificarlo directamente. Con el atributo **t-name="kanban-box"** aplicaremos por herencia la plantilla que tiene por defecto Odoo para mostrar los campos.



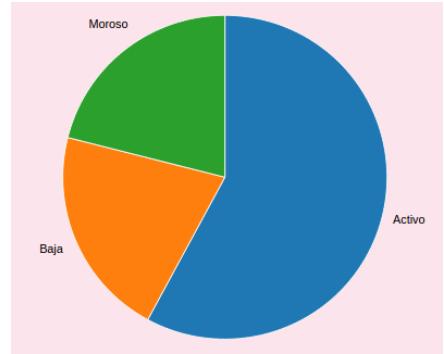
### 6.3.6.- Vistas graph.

Hemos incluido también cuatro vistas graph de cada tipo para que el jefe o “manager” del establecimiento disponga de información a partir de los datos almacenados en la base de datos:

**“Pie”** o tarta, indicado en type=“pie” y un campo de tipo selection el cuál contega varios valores, usará como modelo “assaig.cliente” para mostrar los datos:

```

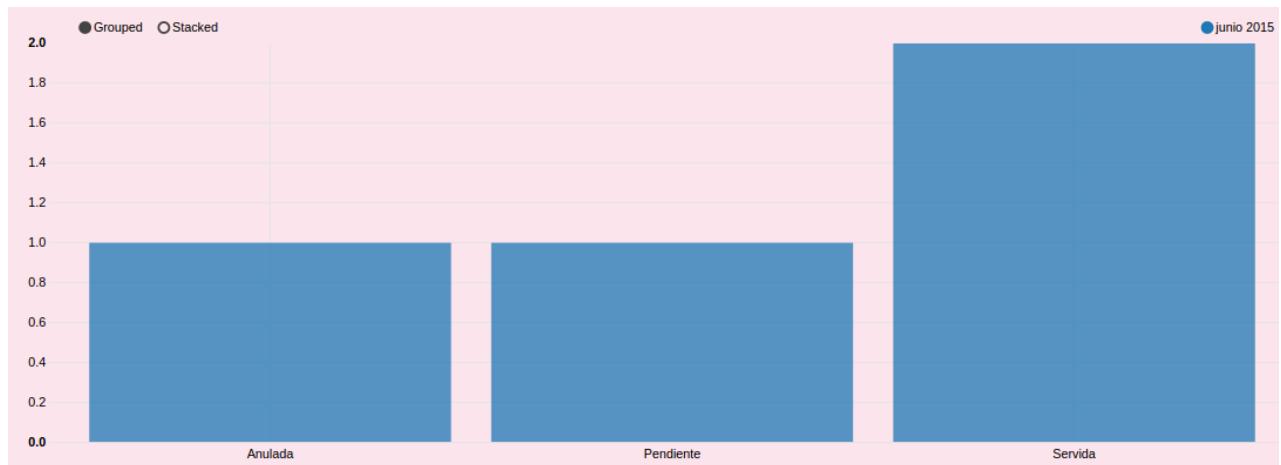
398      <record model="ir.ui.view" id="view_alta_cliente_graph">
399          <field name="name">view_reservas_fecha_graph</field>
400          <field name="model">assaig.cliente</field>
401          <field name="type">graph</field>
402          <field name="arch" type="xml">
403              <graph type="pie">
404                  <field name="situacion"/>
405              </graph>
406          </field>
407      </record>
```



**“Bar”** o barras, indicando varios campos para que nos muestre a partir de los datos del modelo “assaig.reserva” los tipos de reservas asociados a los clientes y la fecha completa de la reserva realizada. Podemos indicar que esté agrupado “grouped” o superpuesto “stacked”.

```

<record model="ir.ui.view" id="view_estado_reserva_graph">
    <field name="name">view_reservas_estado_graph</field>
    <field name="model">assaig.reserva</field>
    <field name="type">graph</field>
    <field name="arch" type="xml">
        <graph type="bar" Stacked="True">
            <field name="state"/>
            <field name="fecha_todo"/>
            <field name="cliente_id"/>
        </graph>
    </field>
</record>
```





**“Pivot”** o tabla / cuadrícula, en donde podemos indicar que un campo (por ejemplo “fecha\_todo”) se presente en filas o “row” indicando el intervalo en meses “month” (trimestres, semanas, etc.) e indicando como columna de resultado type=measure y que además sume las cantidades mediante el atributo operator=”+” (también podemos multiplicar “\*”, exponente “\*\*”, etc).

```
<record model="ir.ui.view" id="view_entrega_reserva_graph">
    <field name="name">view_reservas_fecha_graph</field>
    <field name="model">assaig.reserva</field>
    <field name="type">graph</field>
    <field name="arch" type="xml">
        <graph type="pivot">
            <field name="fecha_todo" type="row" interval="month"/>
            <field name="entrega" type="measure" operator="+"/>
        </graph>
    </field>
</record>
```

	✚ Total Entrega
✚ Total	180,00
✚ abril 2015	89,00
✚ Sergio Juarez Fernandez	89,00
✚ mayo 2015	35,00
✚ Paquita Fustel Pico	35,00
✚ junio 2015	56,00
✚ Jaime Casal Llopis	0,00
✚ Alfonso Huelmo Vaquero	56,00
✚ Carlos Huelmo Vaquero	0,00
✚ Juan Jaime Llopis Casal	0,00

Además es posible modificar los datos a mostrar al pulsar sobre el ícono “+” entonces Odoo nos presentará todos los campos que contiene el modelo “assaig.reserva”, pudiendo seleccionar los que queramos para que se muestren a la cuadrícula, es decir, **podemos obtener mayor detalle pero podemos incurrir en un exceso de información y podemos hacer que el usuario “se agobie”.**

	✚ Total Entrega
✚ Total	180,00
✚ abril 2015	89,00
✚ mesa	Cliente
✚ junio	Created by
	Last Updated by
	Mesa
	Estado
	Last Updated on
	Tipo de Reserva
	Fecha reserva
	Created on

## 6.4.- ACTIONS.

Hemos definido las siguientes acciones que estarán asociadas a cada menuitem de nuestro módulo. Cada action tiene como referencia un identificador o “**id**”, entonces dentro de la etiqueta **<menuitem...>** dispone de un atributo action que hace referencia a dicho “**id**” del action en cuestión.

Modelo “assaig.cliente” (mismo código y estructura para el resto de modelos):

```

447      <record model="ir.actions.act_window" id="action_cliente">
448          <field name="name">Clientes</field>
449          <field name="res_model">assaig.cliente</field>
450          <field name="view_type">form</field>
451          <field name="view_mode">tree,form</field>
452          <field name="help" type="html">
453              <p class="oe_view_nocontent_create">
454                  Pulse el botón "crear"
455              </p>
456              <p>Aún no tienen ningun cliente creado...</p>
457              <p class="texto_error_search">También es posible que la búsqueda no diera ningún resultado.</p>
458              <div align="center">
459                  
460              </div>
461              <div class="oe_chatter_my_footer" align="left">
462                  <p>PROYECTO 2º DAM CURSO 2014 - 2015</p>
463              </div>
464          </field>
465      </record>

```

Explicamos:

- Línea 451, las vistas que se cargarán serán “tree” y “form”, en este orden, podremos ver los iconos correspondientes en la esquina superior izquierda de la vista en Odoo.
- Línea 452, indicamos que el contenido es un “página html” de ayuda en caso de que no tengamos datos que mostrar o si realizamos una búsqueda y no devuelva resultados.
- Línea 453 – 455, a través de un CSS del propio Odoo incluimos una imagen que tiene por defecto Odoo.
- Línea 459, indicamos la ruta relativa dónde se encuentra la imagen a mostrar.
- Líneas 461 – 463, añadimos nuestro propio pie a la vista.





Dispondremos de otro action para ver la vista graph de los clientes que se encuentran en situación: morosos, activos y baja. Para ello utilizaremos el atributo

“ref” y el “id” de la vista graph correspondiente. **Los otros actions que sirven para cargar el resto de vistas graph (o calendar, por ejemplo) que hemos creado son exactamente igual cambiando el “view\_mode” y el valor del “ref” para el “view\_id”.**

```
<record model="ir.actions.act_window" id="action_alta_cliente_graph">
  <field name="name">Situación de los clientes</field>
  <field name="res_model">assaig.cliente</field>
  <field name="view_type">form</field>
  <field name="view_mode">graph</field>
  <field name="view_id" ref="view_alta_cliente_graph"/>
</record>
```

#### Actions que implican al modelo “assaig.mesa”:

Tendremos dos action que mostrarán dos vistas kanban diferentes, ambas se diferencian en “filtrar” la ubicación (líneas 502 y 503).

```
496      <record model="ir.actions.act_window" id="action_mesa_1">
497        <field name="name">Distribución mesas comedor</field>
498        <field name="res_model">assaig.mesa</field>
499        <field name="view_type">form</field>
500        <field name="view_mode">kanban,form</field>
501        <field name="view_id" ref="mesa_kanban_1"/>
502        <!-- <field name="context">{'search_default ubicacion': 1}</field> -->
503        <field name="domain">[('ubicacion','=', 'comedor')]</field>
504        <field name="help" type="html">
505          <p class="oe_view_nocontent_create">
506            Pulse el botón "crear"
507          </p>
508          <p>No hay mesas creadas para el comedor. Puede dar de alta una...</p>
509          <p class="texto_error_search">También es posible que la búsqueda no diera ningún resultado.</p>
510          <div align="center">
511            
512          </div>
513          <div class="oe_chatter_my_footer" align="left">
514            <p>PROYECTO 2º DAM CURSO 2014 - 2015</p>
515          </div>
516        </field>
517      </record>
```

Explicamos:

- Línea 500, al cargar el “action” la primera vista será la kanban y después la form.
- Línea 501, mediante el atributo ref indicaremos que cargue la vista con el “id” indicado.
- Línea 502, usando el atributo context indicamos que use la *vista search* definida anteriormente, dicho context hace referencia al atributo name del campo **<filter>** empleado en la vista search.
- Línea 503, con el uso del atributo domain, podemos filtrar directamente a partir del valor del campo “ubicación”, en este caso, mostrará sólo las mesas que estén en el comedor.



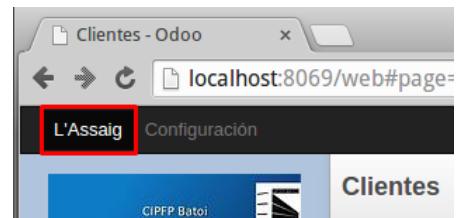
## 6.5.- MENÚS.

Recordemos que OpenERP ha sido desarrollado sobre el framework OpenObject, el cuál es de tipo RAD (Rapid Application Development), con arquitectura MVC (Model-View-Controller) y engloba las vistas y menús que facilitan el acceso a las vistas.

También los podemos encontrar en el mismo fichero de las vistas “views/assaig\_view.xml” aunque puede estar separado, como el resto de vistas, actions, etc. A partir de la línea 615 de dicho fichero encontramos el menú con sus submenús.

```
<menuitem id="menu_assaig_main" name="L'Assaig" action="action_cliente" />
```

- Línea 615, le damos un “id” propio, con el atributo name le damos el nombre al módulo y con el atributo action indicaremos qué action (vista/-s que tiene que cargar) debe realizar. Este código nos muestra en la barra superior el nombre del módulo, al pulsar sobre éste cargará la vista y su menú.



```
<menuitem id="menu_assaig_cliente" parent="menu_assaig_main" name="Gestionar Clientes" sequence="1"/>
<menuitem id="menu_assaig_cliente_alta" parent="menu_assaig_cliente" name="Clientes" action="action_cliente" />
```

- Crearemos después una sección, para ello tendremos que darle otro “id” pero indicando con el atributo parent que “cuelga” del principal, con el atributo name indicamos el texto que aparecerá al cargar el menú. El atributo sequence sirve para indicar el orden en que deberá mostrarse el menú (de arriba a abajo). ***El resto de menús y submenús es prácticamente igual por lo que no pondremos captura.***

Pero con la salvedad del submenú “Resumen según” donde mostraremos las vistas graph que ya hemos explicado en puntos anteriores.

```
<menuitem id="menu_assaig_resumen" parent="menu_assaig_main" name="Resumen según" sequence="4" groups="group_assaig_admin"/>
<menuitem id="menu_assaig_resumen_1" parent="menu_assaig_resumen" name="Tipo de reserva" action="action_tipo_reserva_graph"/>
<menuitem id="menu_assaig_resumen_2" parent="menu_assaig_resumen" name="Estado de la reserva" action="action_estado_reserva_graph"/>
<menuitem id="menu_assaig_resumen_3" parent="menu_assaig_resumen" name="Entregas a cuenta" action="action_entrega_reserva_graph"/>
<menuitem id="menu_assaig_resumen_4" parent="menu_assaig_resumen" name="Situación de los clientes" action="action_alta_cliente_graph"/>
```

- Destacamos el atributo groups el cual contiene como valor el nombre del grupo de administradores de nuestro módulo, esto quiere decir que sólo será visible para los usuarios miembros de dicho grupo, por ejemplo el usuario “manager”.



## 6.6.- TEMAS Y ESTILOS.

Odoo tiene sus propias clases CSS, la mayoría las podemos encontrar en “`addons/web/static/src/css/base.css`” o bien podemos ir “encontrando” dichas clases utilizando el “inspector” (*inspeccionar elemento*) en el caso de google chrome o con mozilla firefox.

Hemos intentado crear nuestros propios estilos pero heredando y modificando los existentes para ello hemos añadido las siguientes líneas de código al fichero “`views/assaig_view.xml`”:

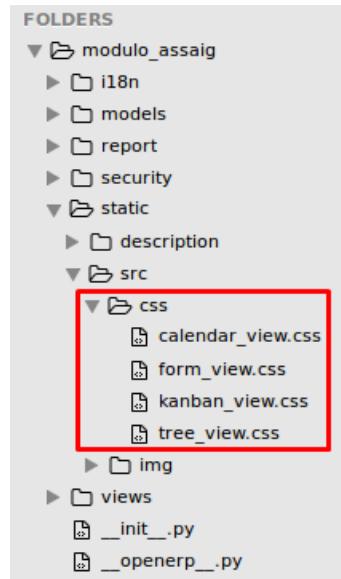
```

4      <template id="dam_assets_backend" name="Personalizar" inherit_id="web.assets_backend">
5          <xpath expr=". " position="inside">
6              <link rel="stylesheet" href="/modulo_assaig/static/src/css/kanban_view.css"/>
7              <link rel="stylesheet" href="/modulo_assaig/static/src/css/form_view.css"/>
8              <link rel="stylesheet" href="/modulo_assaig/static/src/css/tree_view.css"/>
9              <link rel="stylesheet" href="/modulo_assaig/static/src/css/calendar_view.css"/>
10         </xpath>
11     </template>

```

- Dentro de la etiqueta `<data>`, añadiremos otro tag `<template>`, indicaremos mediante el atributo `inherit_id` que herede las clases del módulo web, después mediante la expresión xpath indicaremos la ruta relativa de nuestros archivos CSS que afectarán a las diferentes vistas. Pero ya habíamos indicado que en el archivo manifest o detalle del módulo “`__openerp__.py`” el campo `'css': [ 'static/src/css/kanban_view.css' ]`, pero no afecta al cargar el módulo, solamente funcionaba con OpenERP 7 mientras y no en Odoo 8.

Debremos tener en cuenta el tipo de navegador utilizado para visualizar nuestras vistas, ya que hay estilos que no son soportados por la mayoría, por tanto recomendamos el uso de [google chrome](#) porque en firefox ha dado problemas.



## 6.7.- DESCRIPCIÓN DEL MÓDULO (MEJORA).

Ya habíamos visto que dentro del archivo “`__openerp__.py`” teníamos un campo llamado `'description'`, pues bien, ahora con Odoo 8, podemos mejorarlo, crearemos la carpeta “`/modulo_assaig/static/description`” y dentro de ésta tendremos un archivo `index.html` junto con imágenes que queramos mostrar. Al cargar el módulo o acceder a él desde el menú de configuración, podremos ver un cambio sustancial.



L'ASSAIG  
By Carlos Huelmo Vaquero  
Activar Desactivar

Información		Datos técnicos		Opciones instaladas	
Sitio web	<a href="https://noodle.cipbatox.es">https://noodle.cipbatox.es</a>	Nombre técnico	modulo_assaig	Licencia	Afero GPL-3
Categoría	Hostelería	Resumen	Proyecto de un módulo de hostelería.	Última versión	8.05.15b

**L'Assaig: una propuesta de hostelería en Odoo**  
Gestión de clientes, mesas y reservas

Breve historia del centro: en el curso 2005/06 el Centro dejó de ser un centro de formación profesional para transformarse en el curso 2009/10, en un Centro Integrado Público de Formación Profesional en el cual se imparte toda la formación profesional tanto reglada como ocupacional, es decir, formación inicial, formación de grado medio, formación de grado superior, formación ocupacional y formación continua.

El presente módulo ha sido desarrollado como proyecto final.



## 7.- SEGURIDAD EN ODOO.

Al instalar cualquier módulo que no contenga las reglas de registro ni listas de controles de acceso, sólo será accesible por el usuario “Administrator / Admin” además deberá crear manualmente los grupos de privilegios sobre los objetos / modelos de dicho módulo.

En el caso de que no se hubiese definido ni las reglas ni las listas de control de acceso, cada vez que carguemos y/o actualicemos nuestro módulo, el servidor lo indicará mediante mensajes de log, en los cuales se “nos aconseja” una posible configuración simple:

```
2015-04-23 07:18:42,602 18782 INFO prueba2 openerp.modules.loading: loading modulo_assaig/security/ir.model.access.csv
2015-04-23 07:18:42,701 18782 WARNING prueba2 openerp.addons.base.ir.ir_translation: module modulo_assaig: no translation for language es
2015-04-23 07:18:42,701 18782 WARNING prueba2 openerp.addons.base.ir.ir_translation: module modulo_assaig: no translation for language ca_ES
2015-04-23 07:18:42,702 18782 WARNING prueba2 openerp.addons.base.ir.ir_translation: module modulo_assaig: no translation for language en_GB
2015-04-23 07:18:42,702 18782 WARNING prueba2 openerp.addons.base.ir.ir_translation: module modulo_assaig: no translation for language fr
2015-04-23 07:18:42,734 18782 INFO prueba2 openerp.modules.loading: 51 modules loaded in 0.54s, 0 queries
2015-04-23 07:18:43,330 18782 WARNING prueba2 openerp.modules.loading: The model webassaig.cliente has no access rules, consider adding one.
E.g. access_webassaig_cliente,access_webassaig_cliente,model_webassaig_cliente,,1,0,0,0
2015-04-23 07:18:43,330 18782 WARNING prueba2 openerp.modules.loading: The model webassaig.mesa has no access rules, consider adding one.
E.g. access_webassaig_mesa,access_webassaig_mesa,model_webassaig_mesa,,1,0,0,0
2015-04-23 07:18:43,330 18782 WARNING prueba2 openerp.modules.loading: The model webassaig.reserva has no access rules, consider adding one.
E.g. access_webassaig_reserva,access_webassaig_reserva,model_webassaig_reserva,,1,0,0,0
```

Captura: openscript server-log syslog

La estructura de la seguridad de un módulo consiste en la definición de dos ficheros:

- Se colocan en la carpeta “**nombre\_modulo/security**” y deben estar referenciados dentro del archivo manifest o definición de nuestro módulo “**\_\_openerp\_\_.py**” en el apartado ‘**data**’.
- Un fichero **XML** el cual contendrá la definición de los grupos, usuarios, menús y reglas de negocio asignados a cada grupo.
- Un fichero **CSV** con los privilegios de cada grupo sobre los distintos objetos / modelos del módulo, llamado “**ir.model.access.csv**”, en el caso de utilizar cualquier otro nombre dará error al cargar y al actualizar la lista de módulos puesto que el servidor odoo no sabe qué es o qué hace referencia.

```
'data' : [
    'views/assaig_view.xml',
    'security/ir.model.access.csv',
],
```



## 7.1.- ESTRUCTURA DEL FICHERO “ir.model.access.csv”.

Siempre debe contener la primera línea con el siguiente contenido y formato:

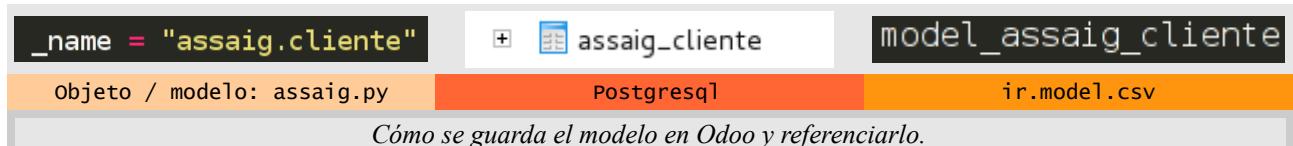
```
ir.model.access.csv x
1 id,name,model_id:id,group_id:id,perm_read,perm_write,perm_create,perm_unlink
```

En nuestro módulo, tendremos las siguientes líneas:

```
ir.model.access.csv x
1 id,name,model_id:id,group_id:id,perm_read,perm_write,perm_create,perm_unlink
2 access_cliente_public,assaig.cliente public,model_assaig_cliente,base.group_public,1,0,0,0
3 access_mesa_public,assaig.mesa public,model_assaig_mesa,base.group_public,1,0,0,0
4 access_reserva_public,assaig.reserva public,model_assaig_reserva,base.group_public,1,0,0,0
5 access_config_public,assaig.configuracion public,model_assaig_configuracion,base.group_public,0,0,0,0
```

Explicamos la estructura:

- “**id**”: identificador único y no deberá contener ningún . *Ejemplo: hemos indicado “access\_cliente\_public” para indicar que hace referencia al modelo cliente.*
- “**name**” : nombre que describe el privilegio. *Ejemplo: “assaig.cliente public”.*
- “**model\_id:id**” nombre del objeto / modelo sobre el que se aplicará el privilegio. Debe ir precedido por “**model\_**”. *Ejemplo: “model\_assaig\_cliente”.*

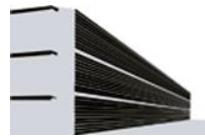


The screenshot shows the Odoo interface for creating a new model. The model name is set to `_name = "assaig.cliente"`. The interface includes tabs for `Objeto / modelo: assaig.py`, `Postgresql`, and `ir.model.csv`. A note at the bottom states: *Cómo se guarda el modelo en Odoo y referenciarlo.*

- “**group\_id:id**” identificador del grupo de privilegios, está definido en el fichero XML. *Ejemplo: “base.group\_public” indicamos un grupo ya creado en Odoo, donde cada usuario que forme parte tendrá los permisos que indiquemos, también es posible indicar otro grupo creado previamente por el Administrador y luego referenciarlo aquí.*
- Permisos sobre acceso a vistas, registros... Posibles valores “**1**” (permitir) y “**0**” (denegar):
  - “**perm\_read**” permiso de lectura.
  - “**perm\_write**” editar la información de los campos.
  - “**perm\_create**” crear registros.
  - “**perm\_unlink**” borrar registros.

Resumiendo, hemos creado cuatro listas de acceso para los cuatro modelos que forman nuestro módulo “modulo\_assaig” para acceder a los: clientes, mesas, reservas y configuración, además de dar permisos de sólo lectura para aquellos usuarios registrados dentro del grupo “public”.

También lo que podemos hacer es crear manualmente tendremos que ser usuario



“Administrador”, ir a “configuraciones” - “seguridad” y podremos acceder a las “Reglas de registros” y “Listas controles de acceso”:

Reglas de registros							
<a href="#">Crear</a> o Importar						1-3 de 3	
<input type="checkbox"/>	Objeto	Nombre	Global	Dominio		Aplicar para lectura	Aplicar para escritura
<input type="checkbox"/>	Empresa	Portal Personal Contacts	<input type="checkbox"/>	[('message_follower_ids','in',[user.commercial_partner_id.id])]	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
<input type="checkbox"/>	Empresa	res_partner: portal/public: read access on my commercial partner	<input type="checkbox"/>	[('id','child_of', user.commercial_partner_id.id)]	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

#### Reglas de registros.

\*\*\* Las reglas globales (no específicas de un grupo) son restricciones generales y no pueden ser sobrepasadas. Las reglas de grupo conceden permisos adicionales, pero están restringidas a los límites de las globales. Las reglas del primer grupo restringen más que las reglas globales, pero cualquier regla adicional de grupo añadirá más permisos. [Fuente Odoo](#).

Lista controles de acceso							
<a href="#">Crear</a> o Importar		1-33 de 33					
<input type="checkbox"/>	Nombre	Objeto	Grupo	Permiso para leer	Permiso para escribir	Permiso para crear	Permiso para eliminar
<input type="checkbox"/>	res_partner group_public	Empresa	Público	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	res_partner group_portal	Empresa	Portal	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

#### Lista de controles de acceso.

De momento nos vamos a centrar en crear y aplicar varias listas de controles de acceso para nuestro módulo directamente creadas a partir de nuestro archivo “**security/ir.model.access.csv**” por lo que cabe destacar que una vez cargado nuestro módulo, dichas listas aparecerían en el listado de controles de acceso:

Lista controles de acceso							
<a href="#">Crear</a> o Importar		1-4 de 4					
<input type="checkbox"/>	Nombre	Objeto	Grupo	Permiso para leer	Permiso para escribir	Permiso para crear	Permiso para eliminar
<input type="checkbox"/>	assaig.cliente public	Tabla de clientes	Público	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	assaig.mesa public	Tabla de mesas	Público	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	assaig.reserva public	Tabla de reservas	Público	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	assaig.configuracion public	Tabla de la configuracion	Público	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>



Observamos que nuestras listas de acceso aparecen, así como el objeto o modelo al cuál hacen referencia, el grupo y qué permisos están habilitados. Viéndolo con más detalle, los valores que presenta dicha tabla se obtienen de:

**“Nombre”** indicado en el fichero “**ir.model.access.csv**” >> **assaig.cliente public.**

```
ir.model.access.csv x
1 id,name,model_id:id,group_id:id,perm_read,pe
2 access_cliente_public,assaig.cliente public,
3 access_mesa_public,assaig.mesa public,model_
4 access_reserva_public,assaig.reserva public,
5 access_config_public,assaig.configuracion pu
```

**“Objeto”** indicado en “**models/assaig.py**” como descripción del modelo.

```
_name = "assaig.cliente"
_rec_name = "name"
_description = "Tabla de clientes"
```

**“Grupo”** indicado en el fichero “**ir.model.access.csv**”.

**group\_id:id**  
**base.group\_public**

Con estas listas de acceso **permitimos que “cualquier” usuario pueda, “ver”** (leer) todos los registros de las vistas que contenga el módulo **excepto** la referente a **la configuración**, con esto el usuario no podrá: crear, modificar, borrar ni escribir nuevos registros a los modelos asociados. Aunque lo más conveniente, en un primer momento es establecer todos los permisos a valor “0”.

Cuando creamos un nuevo usuario “**carlos**” desde “configuración” - “usuarios” - “usuarios”, en la pestaña “permisos de acceso” indicamos en el apartado “Otro” marcamos la casilla “Público” y automáticamente se habrá añadido en “usuarios” - “grupos”, al grupo “Público”:

Aplicación	Nombre	Público				
Grupo compartición	Portal	<input checked="" type="checkbox"/>				
Usuarios	Heredado	Menús	Vistas	Permisos de acceso	Reglas	Notas
Nombre	Usuario	Idioma	Última conexión			
Administrator	admin	Spanish / Español	23/04/2015			
carlos	carlos	Spanish / Español	23/04/2015			

Odoo 8: Detalle grupo público.



En la pestaña “permisos de acceso” del grupo podemos ver qué listas afectan a los usuarios que lo forman y sus correspondientes objetos.

Aplicación	Nombre						
Grupo compartición	Público						
	Usuarios	Heredado	Menús	Vistas	Permisos de acceso	Reglas	Notas
Objeto	Permiso para leer	Permiso para escribir	Permiso para crear	Permiso para eliminar	Nombre		
Empresa	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	res_partner	group_public	
Tabla de clientes	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	assaig.cliente	public	
Tabla de mesas	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	assaig.mesa	public	
Tabla de reservas	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	assaig.reserva	public	
Tabla de la configuracion	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	assaig.configuracion	public	

Odoo 8: permisos de acceso a objetos relacionados.

## 7.2.- MEJORAR LA SEGURIDAD DEL MÓDULO.

Desde la configuración de Odoo (“configuración” - “usuarios” - “grupos”), crearemos dos grupos:

- “assaig\_admin”: donde los usuarios tendrán todos los permisos para los objetos de nuestro módulo. Crearemos el usuario adminbatoi.
- “assaig\_cambrer”: en el cual los usuarios no podrán eliminar ni tampoco acceder a la vista del modelo configuración. Crearemos: cambrer1, cambrer2.

Al crear cada usuario, recordemos que en la pestaña “permisos de acceso” - “otro” ya nos aparecen los dos grupos que hemos creado:

### Otro

assaig_admin	<input checked="" type="checkbox"/>	assaig_cambrer	<input type="checkbox"/>
Creación de contactos	<input type="checkbox"/>	Portal	<input type="checkbox"/>
Público	<input checked="" type="checkbox"/>	Comentarios del sitio web	<input type="checkbox"/>

Seleccionamos un grupo u otro, según nuestro tipo de usuario.

Una vez que tengamos nuestros usuarios dados de alta y asignados a sus grupos correspondientes (assaig\_admin o assaig\_cambrer) también tendremos que indicar que pertenecen a los grupos público y portal para tener los permisos por defecto y poder acceder a nuestro servidor, por otra parte recordemos que los permisos se suman.

Por otro lado, también disponemos de la posibilidad de crear nuestras propias listas de control de



acceso dentro del grupo creado “**assaig\_admin**” mediante la propia interfaz proporcionada por Odoo:

Aplicación	<input type="text"/>	Nombre	assaig_admin			
Grupo compartición	<input type="checkbox"/>	Portal	<input type="checkbox"/>			
Usuarios	Heredado	Menús	Vistas	Permisos de acceso	Reglas	Notas
Objeto	Permiso para leer	Permiso para escribir	Permiso para crear	Permiso para eliminar	Nombre	
Tabla de clientes	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	admin	
Tabla de la configuracion	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	admin	
Tabla de reservas	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	admin	
Tabla de mesas	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	admin	
<a href="#">Añadir un elemento</a>						

El administrador tendrá control total en las vistas de nuestro módulo y de sus cuatro objetos / modelos para poder realizar modificaciones, etc.

Por lo que se refiere al grupo “**assaig\_cambrer**” realizaremos los mismos pasos pero los miembros del grupo no podrán borrar ningún registro ni tampoco tendrán ningún permiso para acceder a la configuración ya que esta reservado para el usuario “adminbatoi”:

Aplicación	<input type="text"/>	Nombre	assaig_cambrer			
Grupo compartición	<input type="checkbox"/>	Portal	<input type="checkbox"/>			
Usuarios	Heredado	Menús	Vistas	Permisos de acceso	Reglas	Notas
Objeto	Permiso para leer	Permiso para escribir	Permiso para crear	Permiso para eliminar	Nombre	
Tabla de clientes	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	camarero	
Tabla de mesas	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	camarero	
Tabla de reservas	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	camarero	
Tabla de la configuracion	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	camarero	
<a href="#">Añadir un elemento</a>						

Pero en vez de hacerlo de forma manual, podemos **dar de alta nuestros propios grupos durante la instalación de nuestro módulo** y posteriormente establecer las listas de control de acceso en el archivo “**ir\_model\_access.csv**” correspondiente.

Antes de crear el archivo XML donde especificaremos los dos grupos: administradores y usuarios, es necesario indicar la categoría (identificador) para que nuestro servidor los pueda registrar. Dentro de la carpeta “**/security**” creamos el archivo “**model\_data.xml**”, indicando su identificador y a qué tipo de modelo hace referencia:

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <openerp>
3      <data>
4          <record id="category_assaig" model="ir.module.category">
5              <field name="name">L'Assaig</field>
6              <field name="description">Grupos admin y users de L'Assaig</field>
7              <field name="sequence">15</field>
8          </record>
9      </data>
10 </openerp>

```

*Creación de categorías en “model\_data.xml”.*

Creamos por cada grupo una única categoría. Teniendo la categoría creada, el siguiente paso será crear un nuevo archivo “**assaig\_security.xml**” donde especificaremos cada grupo:

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <openerp>
3      <data noupdate="0">
4          <!-- GRUPO DE USUARIOS/CAMAREROS -->
5          <record id="group_assaig_users" model="res.groups">
6              <field name="description">Crear grupo usuarios</field>
7              <field name="name">Assaig Usuarios</field>
8              <field name="category_id" ref="modulo_assaig.category_assaig"/>
9          </record>
10         <!-- GRUPO DE ADMINISTRADORES -->
11         <record id="group_assaig_admin" model="res.groups">
12             <field name="description">Crear grupo administradores</field>
13             <field name="name">Assaig Admin</field>
14             <field name="category_id" ref="modulo_assaig.category_assaig"/>
15             <field name="users" eval="[(4, ref('base.user_root'))]"/>
16         </record>
17     </data>
18 </openerp>

```

*Grupos creados al instalar el módulo en “assaig\_security.xml”.*

Explicamos:

- Línea 3, “noupdate” con el valor “0”, el módulo instala el esquema de seguridad. En caso de tener valor “1” el módulo no instalaría el esquema de seguridad en caso de actualización.
- Línea 5, le damos un identificador y que toma como modelo “res.groups”.



- Línea 8, indicamos como referencia la categoría “modulo\_assaig.category\_assaig” creada anteriormente. Es necesario indicar el nombre del módulo y la categoría que contiene para que el sistema lo reconozca. **Si sólo ponemos name=“category”, el sistema indica que los grupos no pertenecen a ninguna categoría, por lo que aparecerán en “Otros” cuando queramos asociar un usuario a dichos grupos, pero si ponemos name=“category\_id” ya los muestra en su propia categoría.**
- Línea 15, mediante los atributos “users” y “eval” indicamos a Odoo que el grupo hereda los permisos de administrador, es decir, que los usuarios serán administradores.

Añadiremos más listas de acceso al archivo “**ir\_model\_access.csv**”, para los usuarios que estén en el grupo de “Assaig Users” y “Assaig Admin”:

```

6 access_cliente_admin,assaig.cliente.admin,model_assaig_cliente,group_assaig_admin,1,1,1,1
7 access_mesa_admin,assaig.mesa.admin,model_assaig_mesa,group_assaig_admin,1,1,1,1
8 access_reserva_admin,assaig.reserva.admin,model_assaig_reserva,group_assaig_admin,1,1,1,1
9 access_config_admin,assaig.configuracion.admin,model_assaig_configuracion,group_assaig_admin,1,1,1,1
10 access_cliente_users,assaig.cliente.users,model_assaig_cliente,group_assaig_users,1,1,1,0
11 access_mesa_users,assaig.mesa.users,model_assaig_mesa,group_assaig_users,1,1,1,0
12 access_reserva_users,assaig.reserva.users,model_assaig_reserva,group_assaig_users,1,1,1,0
13 access_config_users,assaig.configuracion.users,model_assaig_configuracion,group_assaig_users,0,0,0,0
14 access_config_admin,assaig.configuracion.admin,model_assaig_configuracion,group_assaig_users,0,0,0,0

```

*Detalle parcial archivo “ir\_model\_access.csv”.*

Como podemos observar los usuarios miembros de “Assaig Admin” dispondrán de todos los permisos, mientras que los miembros de “Assaig Users”. Ahora, tendremos que modificar el archivo manifest “**\_\_openerp\_\_.py**” e indicar dónde están los nuevos archivos que hemos creado **siendo también importante el orden, puesto que debe cargar antes las nuevas categorías para que a su vez pueda dar de alta los dos nuevos grupos y después aplicar las listas de acceso.**

```

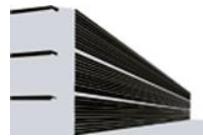
'data' : [
    'views/assaig_view.xml',
    'security/model_data.xml',
    'security/assaig_security.xml',
    'security/ir.model.access.csv',
],

```

Para comprobar todo lo que hemos hecho, en Odoo (después de la instalación / actualización) desde “configuración” - “grupos” podremos ver que se han creado. Dentro de cada grupo, en la pestaña “Permisos de acceso” podemos ver los permisos que habíamos indicado en “**ir\_model\_access.csv**”. Ahora ya podemos crear los usuarios y posteriormente asignarlos a estos dos grupos.

L'Assaig	
<input checked="" type="checkbox"/> Assaig Admin	<input type="checkbox"/> Assaig Usuarios
<b>Usabilidad</b>	
<input type="checkbox"/> Múltiples compañías	<input type="checkbox"/> Características técnicas
<b>Otro</b>	
<input type="checkbox"/> Creación de contactos	<input type="checkbox"/> None
<input type="checkbox"/> OpenOfficeReportDesigner	<input type="checkbox"/> Portal
<input type="checkbox"/> Público	<input checked="" type="checkbox"/> Comentarios del sitio web

*usuario miembro del grupo de administradores del módulo L'Assaig.*



En “**grupos**” tendremos una nueva categoría llamada “**L'Assaig**” que contendrá los dos grupos anteriormente citados. En los detalles de cada grupo podremos ver: los usuarios (por defecto el usuario “admin” queda como miembro). Además, en la pestaña “Permisos de acceso” podremos observar los permisos que tendrá dicho grupo (posteriormente podríamos modificar <> a mano>, pero recordemos que si actualizamos el módulo, nuestras modificaciones serán “machacadas”).

<input type="checkbox"/>	Nombre grupo
<input type="checkbox"/>	Administración / Permisos de acceso
<input type="checkbox"/>	Contabilidad y finanzas / Contable
<input type="checkbox"/>	Configuración técnica / Direcciones en los pedidos de venta
<input type="checkbox"/>	Configuración técnica / Contabilidad analítica
<input type="checkbox"/>	Configuración técnica / Contabilidad analítica para las ventas
<input type="checkbox"/>	L'Assaig / Assaig Admin
<input type="checkbox"/>	L'Assaig / Assaig Usuarios

Odoo también nos ofrece la posibilidad de **dar de alta usuarios automáticamente** e indicar a qué grupo pertencerán en el momento de instalar nuestro módulo. En nuestro caso, crearemos los siguientes usuarios:

- “**manager**”, aunque ya existe el usuario “Administrator” para controlar todas las aplicaciones. Además lo incluiremos como miembro del grupo “Assaig Admin” y a “Portal” para que pueda acceder a Odoo y a su vez al módulo “L'Assaig”.
- “**cambrex**”, el cuál incluiremos como miembro en los grupos “Assaig Users” y “Portal”.

Para lograr esto añadiremos el siguiente código al archivo “**assaig\_security.xml**”:

```

18  <record id="assaig_manager" model="res.users">
19    <field name="login">manager</field>
20    <field name="password">1234</field>
21    <field name="name">Manager</field>
22    <field name="signature">Manager</field>
23    <field name="groups_id" eval="[(6,0,[ref('group_assaig_admin'),ref('base.group_portal')])]" />
24 </record>
```

Explicamos:

- Líneas 19 a 21, indicamos el login, contraseña y el nombre del usuario.
- Línea 23, mediante el atributo **name="groups\_id"**, indicamos la pertenencia a diferentes grupos y mediante el atributo **eval="..."** especificamos éstos. Nótese que **el primer grupo** “está creado dentro del propio módulo” mientras que **el segundo** “hay que indicarlo” mediante la sintaxis “nombre\_módulo.grupo”. **La sintaxis a veces puede confundirnos con tanto paréntesis, corchete, comillas dobles o simples, etc.**



En el caso del usuario “cambrerX”, tendremos el código parecido:

```

26  <record id="assaig_user1" model="res.users">
27      <field name="login">cambrer8</field>
28      <field name="password">1234</field>
29      <field name="name">Cambrer8</field>
30      <field name="signature">Cambrer8</field>
31      <field name="groups_id" eval="[(6,0,[ref('group_assaig_users'),ref('base.group_portal')])]"/>
32  </record>
```

Explicamos:

- Línea 26, pondremos un identificador único para el nuevo usuario.
- Líneas 26 a 30, indicamos el login, contraseña y nombre del usuario.
- Línea 31, este usuario será miembro de los grupos: assaig\_users (propio del modulo) y portal.

Después en vez de actualizar el módulo, recomendamos borrar y reinstalar porque dará error al modificar los permisos y sus identificadores. Con esto si vamos a la “configuración” - “usuarios” - “usuarios” comprobamos que ha creado los usuarios que hemos definido y que cada uno es miembro de los grupos especificados. Recordemos que estos usuarios estarán marcados como activos y tan sólo necesitaremos iniciar sesión para poder empezar a trabajar, tendremos un aviso de Odoo indicando que se indique la “zona horaria por discrepancia”.

Administradores		Usuarios	
<input type="checkbox"/>	Usuarios	<input checked="" type="checkbox"/>	

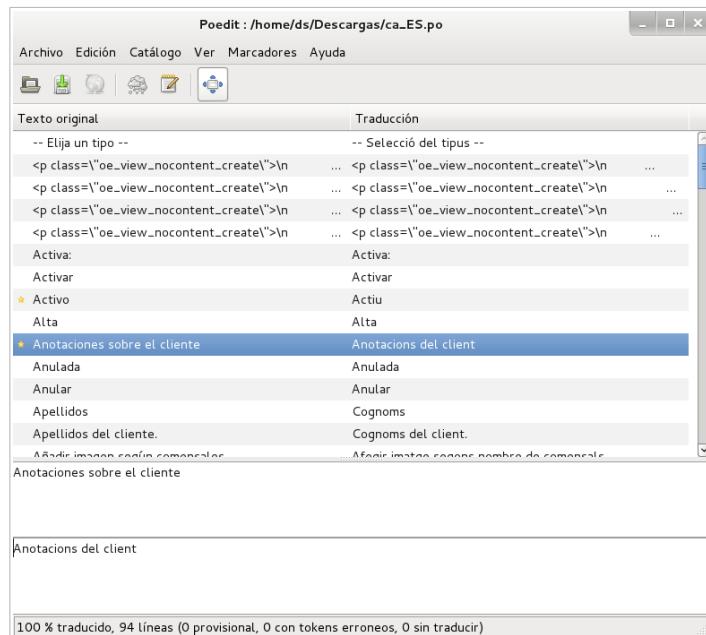
Usabilidad		Características técnicas	
<input type="checkbox"/>	Múltiples compañías	<input checked="" type="checkbox"/>	

Otro			
<input type="checkbox"/>	None	<input type="checkbox"/>	
<input type="checkbox"/>	Portal	<input checked="" type="checkbox"/>	
<input type="checkbox"/>	Público	<input type="checkbox"/>	Comentarios del sitio web



## 8.- INTERNACIONALIZACION.

Para continuar con el desarrollo de nuestro módulo, abordaremos ahora las “**traducciones**” en otros idiomas, por ejemplo: catalán, francés e inglés. En Odoo 8 (al igual que en OpenERP 7) tenemos la posibilidad **generar nuestra propia traducción** del módulo ya que de forma automática, Odoo traduce a nivel estándar o de forma incompleta y poco clara, por lo que podemos exportar el contenido el fichero según el idioma seleccionado.



Lo primero que tenemos que hacer es ir a la “**configuración**” - “**traducciones**” - “**idiomas**” para comprobar cuáles tenemos cargados en nuestro ERP. Siguiendo dentro de “**traducciones**” vamos a “cargar una traducción” del listado de idiomas que nos ofrece Odoo. Realizaremos estos sencillos pasos según los idiomas que queramos.

Después, al volver a “**traducciones**” - “**idiomas**” dispondremos de un listado con todos los idiomas que tenemos hasta el momento cargados:

### Cargar una traducción

<b>Idioma</b> <input checked="" type="checkbox"/> Catalán / Català	<b>Sobrescribir términos existentes</b> <input type="checkbox"/>
<b>Sitios web a traducir</b> <input type="checkbox"/> localhost	
<input style="background-color: #e03030; color: white; padding: 5px 10px; border-radius: 5px; font-weight: bold; margin-right: 10px;" type="button" value="Cargar"/> o <a href="#">Cancelar</a>	

	Nombre	Código local	Código ISO	Dirección	Traducible	Activo	
<input type="checkbox"/>	English	en_US		Izquierda-a-Derecha	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
<input type="checkbox"/>	Spanish / Español	es_ES	es	Izquierda-a-Derecha	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
<input type="checkbox"/>	Catalan / Català	ca_ES	ca_ES	Izquierda-a-Derecha	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
<input type="checkbox"/>	English (UK)	en_GB	en_GB	Izquierda-a-Derecha	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
<input type="checkbox"/>	French / Français	fr_FR	fr	Izquierda-a-Derecha	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	

*Idiomas “nuevos”: catalán, francés e inglés británico.*



## 8.1.- EXPORTANDO NUESTRA TRADUCCION.

Ahora lo que queremos es obtener el fichero de traducción PO a partir de nuestro módulo por lo que tendremos que ir a “traducciones” - “importar / exportar” - “exportar traducción”, aparecerá una ventana emergente:

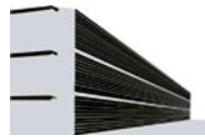


Explicamos:

- **Idioma:** nos mostrará los idiomas que tenemos “cargados” en nuestro servidor aunque también tenemos la posibilidad de indicar una plantilla de traducción vacía.
- **Formato del archivo:** tenemos las extensiones CSV, PO y TGZ. Según el formato del archivo podemos realizar las siguientes acciones:
  - *Formato CSV:* podemos editarla en una hoja de cálculo, siendo la columna más a la derecha la que contendrá las traducciones.
  - *Formato PO(T):* usando un editor PO, por ejemplo “POEdit” o un editor de texto (Gedit, por ejemplo). *Aunque también disponemos de otros programas de traducción como: Xemacs, OmegaT y Virtaal, entre otros.*
  - *Formato TGZ:* archivo comprimido que contiene la ruta completa así como la carpeta y el archivo PO, directamente preparado para cargar en la plataforma de traducción de Odoo.
- **Módulos a exportar:** indicaremos el nombre del módulo (recordemos que nos muestra el nombre que aparece en el archivo manifest “openerp\_\_.py”).

Al clicar sobre el botón “exportar” aparecerá otra ventana indicando que la exportación se ha realizado con éxito, nos mostrará el “link” del archivo generado, al pulsar sobre éste podremos guardarlo dónde deseemos para posteriormente realizar las modificaciones necesarias. Después pondremos el archivo de traducción dentro de la carpeta “../i18n/ca\_ES.po” (por ejemplo).

Una vez finalizados los cambios sobre el archivo PO o CSV, el siguiente paso sería actualizar el módulo, recordemos que el usuario que se identifique tenga seleccionado el idioma en el cuál hemos realizado la traducción, así como en la aplicación desde “configuración” - “idioma” y el “idioma predeterminado” también sea el mismo. **Hemos observado que es necesario reinstalar el módulo desde cero para que se carguen los cambios, a pesar de esto, la traducción se muestra de forma incompleta y parcial.**



Por tanto, la opción más correcta sería que dicha traducción la importáramos para que así de forma automática se modifiquen los campos aunque la efectividad se reduce al 90% del texto traducido. Desde “configuración” - “traducciones” - “importar/exportar” seleccionamos “importar” y aparecerá una nueva ventana en donde indicaremos el **“nombre del idioma”**, el código **“ca\_ES”** (catalán en nuestro caso) e **indicaremos la ruta** dónde tenemos la traducción del módulo.



Como resultado, ya tenemos la traducción al catalán de nuestro módulo.



**Clients / Carlos Huelmo Vaquero**

Nom	Carlos	Cognoms	Huelmo Vaquero
Telèfon	965334545	Mòvil	686203040
Alta	24/04/2015	Actiu	<input checked="" type="checkbox"/>
Morós	<input type="checkbox"/>		

**Información adicional:**

Observaciones  
És d'Alcoi.

*La traducción continúa sin ser precisa, es necesario realizar cambios manualmente.*



## 8.2.- ANALIZANDO EL CÓDIGO DEL ARCHIVO “ca\_ES.po”.

Abrimos el archivo con “Sublime Text”, para traducir el texto en la vista form observamos el siguiente código insertado por Odoo, explicamos:

```
18  #: module: modulo_assaig
19  #: view:assraig.reserva:modulo_assaig.reserva_form
20  msgid "-- Elija un tipo --"
21  msgstr "-- Selecció del tipus --"
```

- Línea 18: indicamos el nombre del módulo.
- Línea 19: se indica la vista (form) del modelo y el identificador del formulario.
- Líneas 20 y 21: como la vista formulario contiene un “placeholder” se cambiará el texto original “msgid” por el traducido “msgstr”.

```
<record model="ir.ui.view" id="reserva_form">
    ...
        placeholder="-- Elija un tipo --"/>
```

Para traducir un texto de la vista kanban, tenemos el siguiente código:

```
104  #: module: modulo_assaig
105  #: view:assraig.mesa:modulo_assaig.mesa_kanban
106  msgid "Activa:"
107  msgstr "Activa:"
```

- Línea 105: se indica que es una vista (kanban) y el identificador correspondiente.

Traducir texto en el código python **“assaig.py”**, en las líneas 115 y 116 hace referencia al campo “fields.boolean” que es el mismo en los modelos “assaig.cliente” y “assaig.configuracion”.

```
114  #: module: modulo_assaig
115  #: field:assraig.cliente,activo:0
116  #: field:assraig.configuracion,estado:0
117  msgid "Activo"
118  msgstr "Actiu"
```

```
_name = "assaig.cliente"
```

```
'activo' : fields.boolean('Activo'),
```

Así como el **“texto contenido en “help”** también se puede traducir:

```
145  #: module: modulo_assaig
146  #: help:assraig.cliente,apellidos:0
147  msgid "Apellidos del cliente."
148  msgstr "Cognoms del client."
```

```
help="Apellidos del cliente.",
```



Traducir texto de las **restricciones sql** establecidas en el modelo “archivo.py”.

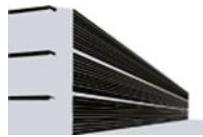
```
262 #. module: modulo_assaig
263 #: sql_constraint:assraig.cliente:0
264 msgid "El cliente ya existe"
265 msgstr "El client ja existeix!"
```

```
_sql_constraints = [
    ('nombre_unique','UNIQUE(nombre,apellidos)','El cliente ya existe'),
]
```

Traducir texto localizado en el **menú del módulo**:

```
246 #. module: modulo_assaig
247 #: model:ir.actions.act_window,name:modulo_assaig.action_mesa2
248 #: model:ir.ui.menu,name:modulo_assaig.menu_assaig_mesa_alta2
249 msgid "Distribución de las mesas."
250 msgstr "Distribució de les taules."
```

```
<menuitem id="menu_assaig_mesa_alta2" parent="menu_assaig_mesa" name="Distribución de las
mesas." action="action_mesa2" />
```



## 9.- REPORTS EN ODOO.

Son ficheros RML, HTML / MAKO o templates de OpenOffice con los que junto con la información de nuestro módulo, utilizaremos para generar informes PDF, HTML o ODT.

Existen varios motores de informes para OpenERP, los cuales se apoyan sobre librerías ya implementadas de python o terceros, los cuales nos requerirán instalar ciertos módulos en OpenERP para su utilización. En el caso de generar informes utilizando “HTML/MAKO” **requeriremos** del “bin” `wkhtmltopdf`<sup>9</sup> y del módulo “**report\_webkit**”.

Según el motor de informes que elijamos, podremos obtener distintos resultados, es decir:

- **Velocidad de desarrollo de los informes.** Por ejemplo, hay motores que nos facilitan una interfaz gráfica (Jasper Reports, OpenOffice).
- **Vistosidad del informe.**
- **Velocidad con los que OpenERP los imprime.** Por ejemplo “reportlab” (RML) es muy rápido en la generación de informes de pocas páginas, pero para informes más extensos “webkit” es la solución más apropiada.

### 9.1.- MOTORES DE INFORMES.

Disponemos de los siguientes:

- **REPORTLAB**, generación de informes a base de plantillas RML. Es “rígido” a la hora de programar pero potente para el diseño del informe. Permite utilizar OpenOffice / LibreOffice (archivos SWX<sup>10</sup>) como interfaz gráfica para diseñar los informes. Funcionalidades limitadas. No necesita ningún módulo ni librería de terceros. Como toma de contacto con los informes es la mejor opción.
- **WEBKIT**, por defecto implantado en Odoo 8. Utiliza plantillas MAKO<sup>11</sup> que nos permite utilizar: Javascript, CSS, HTML y Python. Es el más rápido a la hora de generar informes. Requiere el programa “wkhtml2pdf” instalado en el servidor.
- **JASPER REPORTS**, tenemos la posibilidad de utilizar la herramienta gráfica “iReport”, la cual, es compatible con: internacionalización, subinformes, tablas, códigos de barras. Utiliza una librería de terceros en JAVA ya implementada en el módulo “jasper\_reports” de OpenERP, por lo que necesitamos tener instalado “openjdk-6-jdk”. Se generan ficheros JRXML.

9 *Wkhtmltopdf* y *wkhtmltoimage*: son Open Source bajo licencia LGPLv3, son herramientas por línea de comandos para transformar o convertir código HTML en formato PDF así como los formatos de varias imágenes utilizando el motor renderizado QT Webkit.

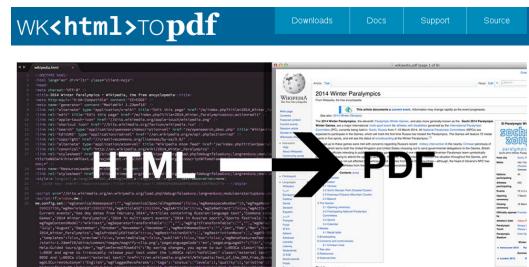
10 SWX: tipo de archivo utilizado para guardar versiones anteriores de OpenOffice.

11 *Mako*: es una plantilla que convierte una secuencia de texto que contiene cualquier tipo de contenido, XML, HTML, etc. Además puede contener directivas que representan la sustitución de variables, expresiones, estructuras de control (bucles, condiciones), comentarios, bloques de código python. Todo esto se compila con código python.



## 9.2.- INFORMES CON WEBKIT.

Se requiere previamente tener instalado “**wkhtmltopdf**” por lo que deberemos comprobarlo mediante el comando “**\$ wkhtmltopdf --version**”, en caso de no tenerlo instalado en nuestro servidor, desde nuestro navegador accederemos a la página web oficial “<http://wkhtmltopdf.org/downloads.html>” en donde seleccionaremos el sistema operativo así como la arquitectura (32/64 bits).



Después para instalar el paquete con extensión “DEB” ejecutaremos el siguiente comando:

```
# dpkg -i wkhtmltox-0.12.2.1_linux_wheezy-amd64.deb
```

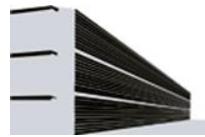
En el caso de que tuviéramos problemas con las dependencias, ejecutamos “# apt-get -f install”.

Para comprobar nuestra versión instalada: “\$ wkhtmltopdf --version”.

Si quisiéramos desinstalarlo: “#apt-get remove - -purge wkhtmltopdf”.

El siguiente paso es instalar el módulo “report\_webkit” desde “configuración” - “modulos locales”, pulsaremos instalar. Dentro de la carpeta de nuestro módulo crearemos otra carpeta “/report” con los siguientes ficheros:

**(FINALMENTE NO HEMOS CONTINUADO, PERO SEGUIREMOS INVESTIGANDO).**



### 9.3.- INFORMES CON REPORTLAB.

Desde “configuración” - “Módulos instalados” buscamos “**base report\_designer**” e instalamos el módulo. Al pulsar sobre el botón correspondiente nos aparecerá una ventana emergente indicando los pasos para poder instalar y configurar el “diseñador de informes de Odoo” en OpenOffice Writer así como el plugin necesario para añadirlo como extensión.



Cabe destacar que no es necesario instalar el plugin como la suite ofimática en el mismo equipo que contiene el servidor Odoo porque recordemos que desde cualquier equipo podemos conectarnos al servidor, sólo tendríamos que preocuparnos de la conectividad.

**Para LibreOffice 4.3 da problemas (en el 3.5 no) al cargar el plugin y no muestra las opciones mientras que en OpenOffice 4.1 funciona, en principio.**

#### Instalación del diseñador de informes de Odoo

Este complemento le permite crear/modificar los informes de Odoo en OpenOffice Writer.

#### Diseñador de informes de Odoo

Plug-in diseñador informe OpenObject | [http://xdev17.ds:8069/base\\_report\\_designer/static/base-report-designer-plugin/openerp\\_report\\_designer.zip](http://xdev17.ds:8069/base_report_designer/static/base-report-designer-plugin/openerp_report_designer.zip)

#### Pasos de instalación y configuración

- \* Save the OpenERP Report Designer plug-in.
- \* Follow these steps to install plug-in.

*Foto: instalación, pasos y configuración “Diseñador de informes”.*

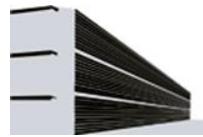
El primero paso será pulsar sobre el enlace y descargar el plugin. Éste será un archivo comprimido con extensión “ZIP” (openerp\_report\_designer.zip). Abrimos “OpenOffice Writer” y nos dirigimos a “herramientas” (tools) - “Extension manager” (gestor de extensiones) y añadiremos una nueva extensión, es decir, buscaremos el archivo que nos hemos bajado y lo instalaremos, después tendremos que reiniciar “writer” para que se apliquen los cambios. Al reiniciar podremos ver que se ha añadido una nueva opción al menú “**OpenERP Report Designer**”:



Después configuraremos los parámetros del servidor para conectarnos, dónde indicaremos:

- **Server:** localhost o la dirección IP de la máquina remota.
- **Port:** 8069, el puerto configurado en Odoo previamente.
- **Protocol Connection:** XML-RPC<sup>12</sup>. Aunque también podemos elegir NET-RPC, XML-RPC secure.

12 XML-RPC: protocolo de procedimiento de llamada remoto que utiliza XML para codificar los datos y HTTP como protocolo de transmisión de mensajes.



Pulsaremos el botón “next”, nos mostrará otra ventana emergente donde nos muestra la URL y las siguientes opciones a completar:

- **Database:** desplegable que nos mostrará las bases de datos disponibles y accesibles (en el caso de que los datos anteriores sean correctos).
- **Login:** usuario administrador (en nuestro caso “admin”) que se corresponde con la configuración de nuestro servidor Odoo.
- **Password:** contraseña, será la del usuario administrador.

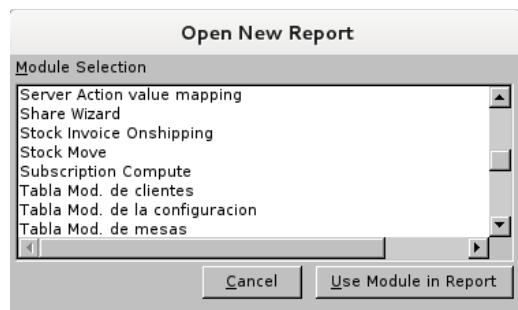
Server Connection Parameter		
Server URL <code>http://localhost:8069</code>		
Database	<input type="text" value="prueba2"/>	
Login	<input type="text" value="admin"/>	
Password	<input type="password" value="****"/>	
<input type="button" value="Cancel"/>	<input type="button" value="Previous"/>	<input type="button" value="Connect"/>

Finalmente pulsamos sobre “connect” para comprobar que ha sido satisfactorio, entonces aparecerá una nueva ventana indicando que la operación ha sido un éxito. Esto ocurrirá cada vez que creemos un nuevo documento de “writer” por lo que se recomienda dejar el archivo ODT dentro de la carpeta “/report”.

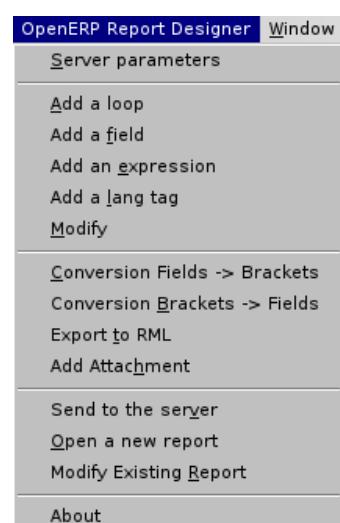
Otro problema que se nos presenta es que tenemos que indicar en las propiedades del nuevo documento (“Archivo” - “Propiedades” - pestaña “Personalizar Propiedades”) qué grupo y a qué modelo se apunta para obtener/modificar el informe. Como ya vimos en el apartado de “Seguridad” daremos de alta un nuevo grupo llamado “**OpenOfficeReportDesigner**” (o el que nos plazca), después lo añadiremos al usuario/-s que queramos los cuales podrán acceder y disponer de permiso para la realización de los informes y después indicar las reglas de acceso (control total) sobre el modelo “**ir.actions.report.xml**”.

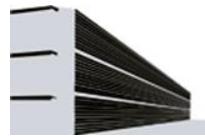
Properties				
Name	Type	Value		
Group	Text	OpenOfficeReportDesigner		
Info 1	Text	http://localhost:8069		
Info 2	Text	admin		
Info 4	Text	account.account		

Conectados a la base de datos del servidor dentro del menú “**OpenERP Report Designer**” crearemos un nuevo informe “**open a new report**” en donde podremos seleccionar en una nueva ventana la lista de modelos que se encuentran en nuestro servidor, con lo que relacionaremos nuestro informe con el modelo.



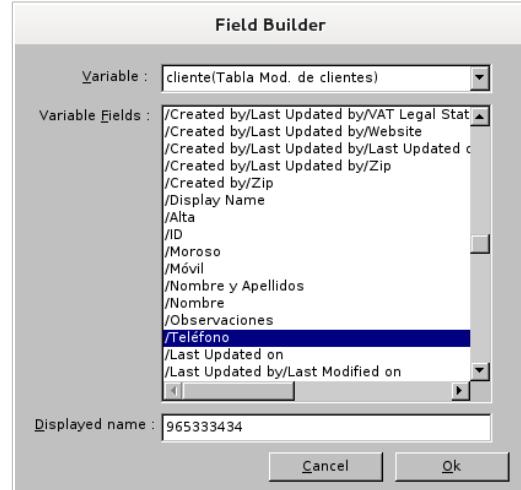
En nuestro caso tenemos los modelos de clientes, mesas y reservas, seleccionaremos “**Tabla Mod. De clientes**” para seleccionar el modelo en nuestro nuevo informe.





Volveremos al menú “OpenERP Report Designer” - “Add a loop” para elegir “los objetos” que contiene el modelo que habíamos seleccionado previamente. Despues seleccionaremos “Add a field” a partir del modelo, nos aparecerá un listado de “campos” que podremos seleccionar (uno o varios) para añadir a nuestro informe. Podemos ver directamente el valor o alternar en la conversión entre campos y “corchetes” (*conversion fields / brackets*) y ver los campos que forman parte del modelo.

Una vez insertados los “campos” necesarios, clicaremos en “Send Report to Server”, se mostrará una nueva ventana de diálogo en la cual daremos un nombre, por ejemplo “Reportlab\_listado\_mesas”, el “**nombre técnico**” se lo da el servidor automáticamente, también podemos indicar que tenga la cabecera de la empresa o no, y finalmente podemos elegir el tipo de informe: PDF, OpenOffice, HTML. Pulsamos “Send report” y nuestro nuevo documento cambiará a “tmpCJ6ICX.SXW”.



Iremos a Odoo, primero comprobaremos que el informe se ha creado y guardado con el resto, desde “configuración” - “informes” - “informes” filtraremos por el nombre del modelo “assaig.cliente” y ahí lo tenemos:

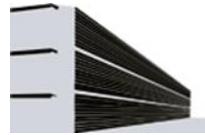
	Nombre	Modelo	Tipo de acción	Nombre de plantilla	Tipo de informe
<input type="checkbox"/>	Reportlab_listado_clientes	assaig.cliente	ir.actions.report.xml	assaig.cliente.rndO0jYg	RML PDF (obsoleto)

Desde el módulo “L'Assaig”, nos situamos en “Clientes”, seleccionamos todos los registros y pulsamos sobre el botón “Imprimir” y aparecerá nuestro informe.



Nuestro informe no lo podremos volver a subir, tendremos que modificarlo “**Modify existing report**”, primero abriremos un nuevo documento, nos conectaremos para después seleccionar el informe. **Pero nos da error “Details: Unsupported URL <file:///tmp/tmp3NWNwu.sxw> por lo que no podemos cargar el informe (debian – LibreOffice 3.5) y en MS Windows usando OpenOffice 4.1 también aunque la ruta cambia).**

También tenemos la posibilidad de borrar el informe seleccionado, que aún indicando que “no se ha podido borrar” después comprobamos que no aparece en informes ni tampoco a la hora de seleccionar el informe a imprimir desde “Clientes”.



El diseño de nuestro informe en writer:

OTRO LISTADO DE PRUEBA EN REPORTLAB

[[repeatIn(objects,'cliente')]]			
Nombre y apellidos	Teléfono	Móvil	Fecha registro
[[ cliente.name ]]	[[ cliente.telefono ]]	[[ cliente.movil ]]	[[ cliente.fecha_alta ]]

Finalmente podemos ver el resultado, aunque en este ejemplo de prueba, cada cliente aparece en cada hoja y no en forma de listado:

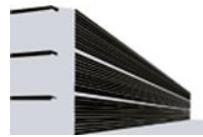


Phone:  
Mail: [info@yourcompany.com](mailto:info@yourcompany.com)

LISTADO DE LOS CLIENTES  
PRUEBA REPORT - REPORTLAB

NOMBRE	APELLIDOS	TELEFONO	MOVIL	ESTADO	MOROSO
Luis	Garcia Garcia	965333434	686909090		

Es demasiado lento a la hora de mostrar los campos y resta efectividad, además no tiene la misma funcionalidad que iReport, pero también es posible realizar informes poco complejos o muy simples, después faltará comprobar el consumo de recursos del sistema a la hora de generar informes como ya se ha comentado previamente.



## 9.4.- INFORMES CON JASPER REPORTS & IREPORT.

Utilizaremos la herramienta gráfica “iReports” versión 5.6.0 que nos descargaremos desde la web oficial [“http://community.jaspersoft.com/project/ireport-designer/releases”](http://community.jaspersoft.com/project/ireport-designer/releases) o bien en el siguiente enlace [“http://sourceforge.net/projects/ireport/”](http://sourceforge.net/projects/ireport/) en la que encontraremos la última versión 5.6.0 del 28 de mayo de 2014. Descomprimiremos el contenido del archivo, dentro de éste, iremos a la carpeta “/bin” y cambiaremos los permisos del archivo “ireport” en caso de ser necesario, así como indicar en las propiedades del archivo “permitir ejecutar el archivo como un programa”.



Al ejecutar el archivo “ireport” cargará y aparecerá una nueva ventana, con una pestaña de bienvenida.

El siguiente paso será descargar el módulo “Jasper Reports” desde los siguientes enlaces, que escribiremos en la ventana de terminal:

```
$ bzr branch lp:~german-ponce/+junk/jasper_reports
$ bzr branch lp:openobject-jasper-reports/7.0 jasper_reports
```

Ahora tendremos nuestro módulo descargado, lo copiaremos dentro de la carpeta “..../addons” o en otras carpetas que ya indicamos en la configuración del servidor en las cuales también se encuentran módulos. Iremos a Odoo y desde “configuración” - “módulos” - “actualizar lista de módulos” para instalarlo. **Pero nos da error de librerías y no es posible continuar.**



**Otra posible solución**, es descargar el módulo de “jasper\_reports” desde el siguiente enlace [“https://github.com/JayVora-SerpentCS/Jasperv8”](https://github.com/JayVora-SerpentCS/Jasperv8) y copiarlo junto con el resto de módulos, actualizamos e instalamos.

Entonces desde Odoo iremos a “configuración” - “técnico” - “Jasper Reports” pulsamos sobre “crea plantilla de datos” entonces nos aparecerá una ventana emergente donde tendremos que llenar los siguientes datos:

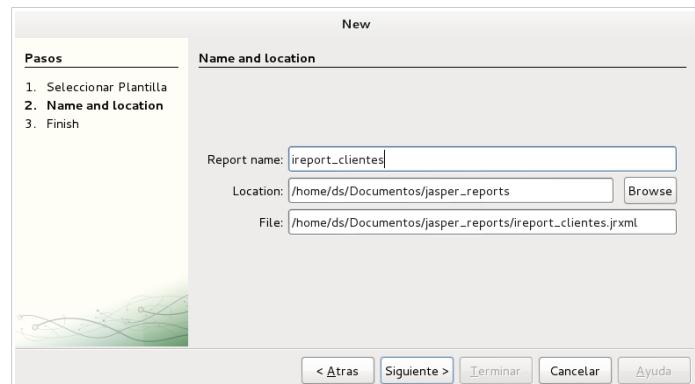
- **Model:** indicaremos el nombre del modelo / objeto, por ejemplo: “assaig.cliente”, también aparece un listado del nombre de los módulos, pero es más fiable ponerlo así, al final mostrará el nombre del modelo pero evitaremos errores.
- **Depth:** profundidad del archivo XML resultante, se recomienda “4” para que muestre todo el contenido, en caso de que esta cantidad fuera superior (5 ó más) dará error en la compilación.

Después al pulsar sobre el botón “crear” se generará una plantilla XML a partir de los datos del modelo antes indicado. Al pulsar sobre el enlace indicaremos dónde queremos guardarla, en nuestro caso en “/home/ds/Documentos/odoo\_templates\_jasper”.

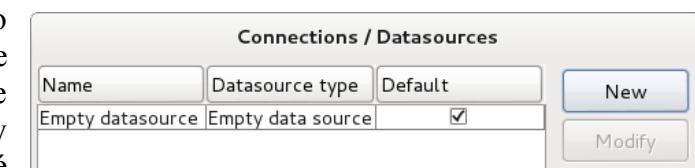


Desde iReport (teniendo la plantilla creada) crearemos un nuevo documento “jrxml” a partir de las plantillas de que dispone por defecto. Indicaremos el nombre del documento, la ruta dónde se guardará y el nombre que tendrá, nosotros lo guardaremos en “**/home/ds/Documentos/jasper\_reports**”.

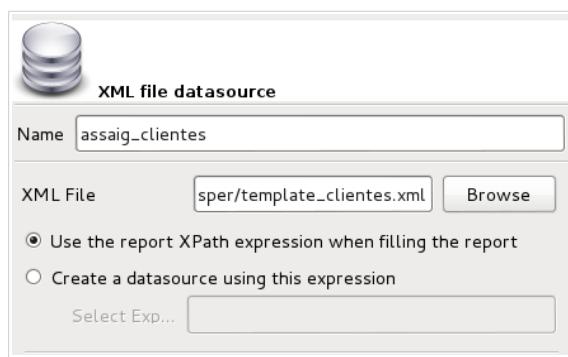
Desde la barra de menú buscaremos el ícono “report datasources” el cual nos abrirá un nuevo cuadro de diálogo (imagen de una base de datos con un enchufe – en Debian).



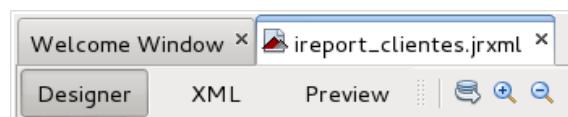
En este cuadro de diálogo llamado “Connections / Datasources” partimos de que no tenemos ninguna conexión, por lo que tendremos que crear una nueva “new” y aparecerá otra ventana para indicar con qué tipo de fuente de datos vamos a trabajar, en nuestro caso “**XML file datasource**” y pulsamos sobre el botón siguiente.



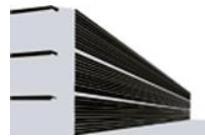
Otro cuadro de diálogo nos pedirá el archivo de datos XML “**xml file datasource**”, el cuál será la plantilla que nos ha creado el módulo de Jasper Reports desde Odoo a partir del modelo indicado. Le daremos un nombre, por ejemplo “**Listado\_clientes**” e indicaremos que utilice Xpath para llenar el informe. Pulsaremos “**save**”.



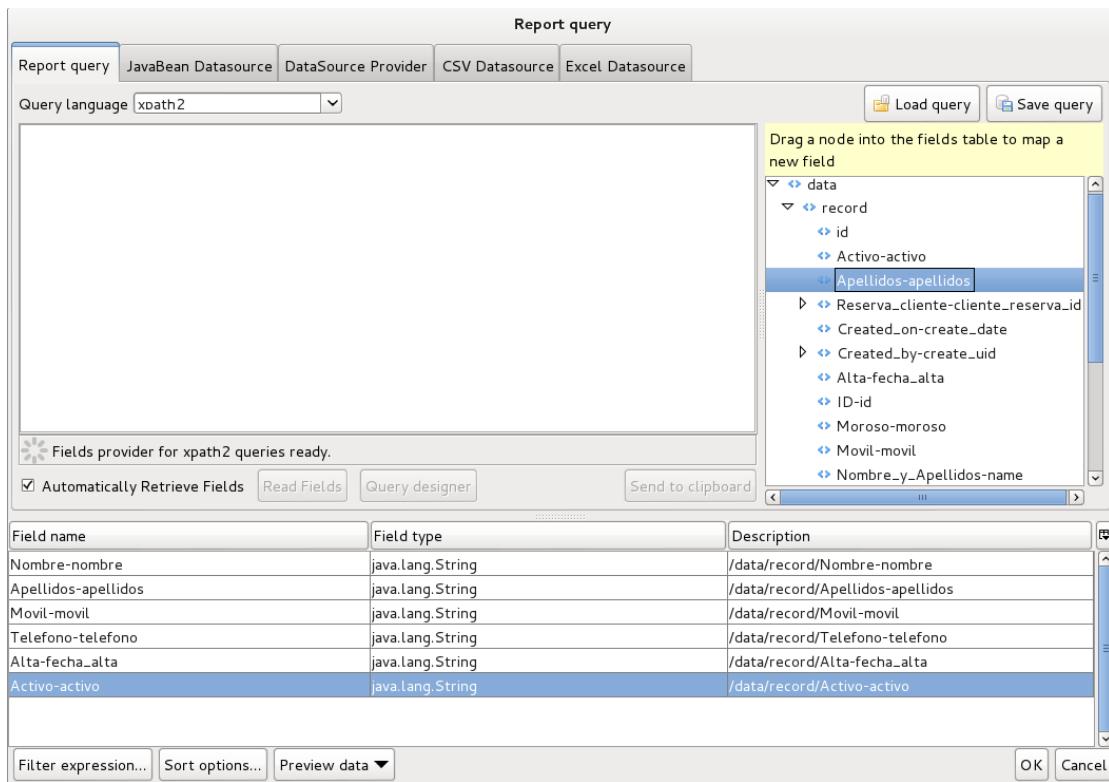
Ahora volveremos a estar en el cuadro de diálogo de conexiones y podremos ver que tenemos nuestra conexión establecida por defecto. Cerramos el cuadro de diálogo.



Nos encontramos en la vista diseño, debajo de la pestaña del nombre de nuestro archivo, tenemos el botón “**report query**” para extraer los datos del archivo XML. Pulsamos.



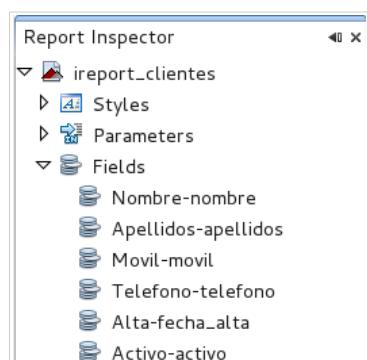
La ventana de “**report query**” aparecerá, nos interesará la primera pestaña, que es donde nos encontramos. Deberemos seleccionar el tipo de lenguaje “xpath2” para que reconozca la estructura del xml de fuente de datos. ***Si ponemos “xpath” también funcionará.***



The screenshot shows the "Report query" window in iReport. At the top, there are tabs: "Report query" (selected), "JavaBean Datasource", "DataSource Provider", "CSV Datasource", and "Excel Datasource". Below the tabs, a dropdown menu shows "Query language" set to "xpath2". To the right of the language dropdown are "Load query" and "Save query" buttons. A large central area contains a tree view of XML fields under the heading "Drag a node into the fields table to map a new field". The tree includes nodes like "data", "record", "id", "Activo-activo", and "Apellidos-apellidos". A message at the bottom of this area says "Fields provider for xpath2 queries ready.". Below the tree is a table titled "Fields provider for xpath2 queries ready." with columns "Field name", "Field type", and "Description". The table lists several fields: "Nombre-nombre" (java.lang.String, /data/record/Nombre-nombre), "Apellidos-apellidos" (java.lang.String, /data/record/Apellidos-apellidos), "Movil-movil" (java.lang.String, /data/record/Movil-movil), "Telefono-telefono" (java.lang.String, /data/record/Telefono-telefono), "Alta-fecha\_alta" (java.lang.String, /data/record/Alta-fecha\_alta), and "Activo-activo" (java.lang.String, /data/record/Activo-activo). At the bottom of the window are buttons for "Filter expression...", "Sort options...", "Preview data", "OK", and "Cancel".

Imagen: iReports – Report Query.

En el desplegable de la derecha podemos ver la estructura de los elementos que componen el archivo XML, seleccionaremos los que necesitemos simplemente con un “drag & drop” arrastraremos y lo soltaremos en el cuadro inferior tal y como se puede apreciar en la imagen. Cuando ya tengamos todos los campos que necesitemos, pulsaremos “ok”.



The screenshot shows the "Report Inspector" window. On the left, there is a tree view of report components. Under "ireport\_clients", there are "Styles", "Parameters", and a "Fields" section. The "Fields" section is expanded and shows six fields: "Nombre-nombre", "Apellidos-apellidos", "Movil-movil", "Telefono-telefono", "Alta-fecha\_alta", and "Activo-activo".

Regresamos a la vista diseño, en el “**inspector de informes**”, en el apartado “fields” (campos) aparecerán los que hemos seleccionado en el “report query”, los arrastraremos a nuestra hoja en blanco y procederemos a diseñar el informe: colores, encabezado, imágenes, etc. Una vez tengamos terminado el diseño guardamos los cambios. Volvemos a “jasper\_reports” en Odoo. **No olvidemos activar la compatibilidad para que los “uid” del xml generado por el módulo “jasper\_report” los quite sino petará, en “herramientas” - “opciones” - “iReport” - pestaña “general” - “compatibilidad” y seleccionar “JasperReports4.0.0”.**



Volvemos a Odoo y desde “**Jasper Reports**” - “**Jasper Reports**” creamos un nuevo informe y tendremos que llenar los siguientes campos del formulario:

Nombre	Listado_clientes	Modelo	Tabla Mod. de clientes
Report Name	Detalle_clientes	Salida de Jasper	PDF
Prefijo del adjunto al guardar como	<input type="text"/>	Recargar desde adjunto	<input type="checkbox"/>

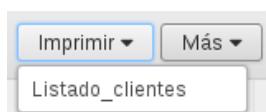
El “**nombre**” que pongamos será el nombre del archivo que generará “jasper\_report” con extensión PDF.

En el apartado “archivos” indicaremos el documento jsxml que hemos creado con “iReports” y debemos indicar que se use por defecto (predeterminado).

### Archivos

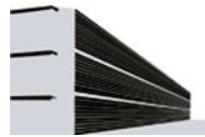
Nombre del archivo	Predeterminado
ireport_clientes.jrxml	<input checked="" type="checkbox"/>
<a href="#">Añadir un elemento</a>	

En “grupos” podemos indicar qué usuarios miembros podrán imprimir dicho informe. Guardamos los datos del nuevo informe. Después iremos a nuestro módulo “L'Assaig” e iremos “Gestionar clientes” - “Clientes” y en la vistas “tree” y “form” nos aparecerá un botón “imprimir” el cual mostrará nuestro informe, relacionado con el modelo “assaig.cliente”. Recordemos que deberemos seleccionar un cliente mínimo para poder imprimir el informe, de lo contrario nos aparecerá un mensaje indicando que seleccionemos uno.



Puede ocurrir que tengamos un error al imprimir el informe desde Odoo:





Resulta que es un error de sintaxis teniendo que **cambiar la palabra “xpath2” por “xPath”** (línea 6), por tanto habrá que modificar el “jrxml” que está cargado en el módulo “Jasper Reports” en la siguiente ruta “**/jasper\_reports/custom\_reports/ireport\_clientes.jrxml**”. Después de corregir el error, volvemos a intentar imprimir el informe, pero esta vez ya nos deja, nos pedirá la ruta dónde descargar el PDF generado, por ejemplo en “**/informes\_pdf\_odoo**”.

```

3   <property name="ireport.zoom" value="1.0"/>
4   <property name="ireport.x" value="0"/>
5   <property name="ireport.y" value="0"/>
6   <queryString language="xPath">
7       <! [CDATA[]]>
8   </queryString>
```

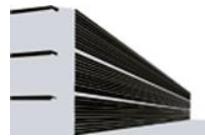
Diseño con iReports.

Documento PDF final.

En el caso de realizar modificaciones al informe “jrxml” que se encuentra en la carpeta “**/custom\_reports**” del módulo “**jasper\_report**” podemos editarlo con **iReport** pero cuando realicemos modificaciones, tendremos que borrar el “**.jasper**” generado para que actualice, de lo contrario veremos lo mismo.

Los informes que podríamos realizar serían los siguientes:

- Clientes:
  - Listado de todos los clientes.
  - Listado de clientes morosos.
- Mesas:
  - Listado de todas las mesas dadas de alta.
  - Listado de mesas del “comedor”.
  - Listado de mesas de la “terraza”.
  - Listado de mesas disponibles.
- Reservas:
  - Listado de reservas del día.
  - Listado según el tipo de reserva (en persona, móvil, teléfono).
  - Listado según el estado de la reserva (pendientes, anuladas, servidas).



También es posible que a partir de un informe que tengamos ya cargado en nuestro módulo, reutilizarlo para imprimir los datos dependiendo de la búsqueda realizada en la vista search:

Clientes								
Crear o Importar		Imprimir ▾		Más ▾				
	Apellidos	Nombre	Teléfono	Móvil	Observaciones	Alta	Moroso	Activo
<input checked="" type="checkbox"/>	Garcia Garcia	Luis	965333434	686909090		29/04/2015	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input checked="" type="checkbox"/>	García Sanchez	José	965332323	686774433		29/04/2015	<input type="checkbox"/>	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	Huelmo Vaquero	Carlos	965334545	690123212		29/04/2015	<input type="checkbox"/>	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	Huelmo Vaquero	Francisco	0	0		29/04/2015	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input checked="" type="checkbox"/>	Perez Sanchez	Antonio	0	0		30/04/2015	<input type="checkbox"/>	<input type="checkbox"/>
<input checked="" type="checkbox"/>	Sanchez Bravo	Luisa	0	0		29/04/2015	<input type="checkbox"/>	<input type="checkbox"/>

Módulo l'assaig en vista tree, realizando una búsqueda según el parámetro introducido.

Como ejemplo, queremos que nos muestre los clientes que se dieron de alta “30/04/2015”, entonces después del filtrado, marcamos todos los clientes y al pulsar sobre el botón “imprimir” seleccionaríamos el informe, en nuestro caso “listado\_clientes” e imprimiríamos el archivo PDF, después al visualizarlo, tenemos los registros que queríamos pero sin hacer un nuevo informe que filtrara por la fecha actual en la que nos encontráramos.



Nombre del cliente	Apellidos	Teléfono	Móvil	Fecha de registro
Luis	León Leones			2015-04-30 00:00:
Antonio	Perez Sanchez			2015-04-30 00:00:

También podemos realizar informes indicando que sólo queremos un listado de los clientes morosos. Entonces lo que tendremos que hacer es, después de seleccionar los campos que vamos a mostrar (o no) en el “report query” ir al botón “Filter expression” y mediante el editor indicaremos los campos y las condiciones. Con esto lograremos imprimir sólo aquellos clientes que son “morfos” y no están “activos”.



```
$F{Moroso-moroso} == "True" & $F{Activo-activo} == ""
```

En caso de que tengamos que **añadir imágenes o logotipos** a nuestro informe tendremos que añadirlas al informe que subamos al módulo de jasper reports, ya que insertábamos las imágenes desde una ruta absoluta, en caso de borrar los archivos, no se mostrarían en el informe y dará error. También podemos modificar el código XML desde iReport y poner el nombre del archivo sin indicar la ruta absoluta o relativa.

```
<image>
    <reportElement x="2" y="0" width="85" height="79"/>
    <imageExpression class="java.lang.String"><! [CDATA["batoi_logo.jpg"]]></imageExpression>
</image>
```



Antes que realizar un subreport, podemos **agrupar las reservas de cada cliente** para que nos las muestre en la línea de detalle. En la ventana del inspector, clicamos sobre el nombre de nuestro informe y con el botón derecho del ratón aparecerá el menú contextual y seleccionaremos la opción “**Add Report Group**”, daremos un nombre e indicaremos por cuál “objeto” realizará la agrupación, después podremos añadir o no el “nuevo grupo” a la cabecera y/o al pie del formulario. Posteriormente tendremos que activarlo, es decir, nos situamos sobre “**nombre\_Group Header 1**” - menú contextual - “**add band**” entonces añadiremos las etiquetas y campos que “se repetirán” mientras que en “**Detail 1**” pondremos los campos con datos.

New group wizard

<b>Pasos</b>	<b>Group criteria</b>
1. Group criteria	Group name
2. Details	reserva
<input checked="" type="radio"/> Group by the following report object: <b>Nombre-nombre</b> Field String	

▼ reservas Group Header 1
 

label Plazas
label Tipo de reserva
label Entrega
label Estado
label Nombre:
<b>T</b> \$F{Nombre-nombr...
label Apellidos:
<b>T</b> \$F{Apellidos-ap...
label Teléfono:
<b>T</b> \$F{Telefono-tel...
label Móvil:
<b>T</b> \$F{Movil-movil}
label Fecha de la reserva

▼ Detail 1
 

\$F{Plazas-plaza...
\$F{Tipo_de_Rese...
\$F{Entrega-entr...
\$F{Estado-state...
\$F{Fecha_reserv...

▼ reservas Group Footer

**Clients and reservations.**

Fecha and time of printing 04/05/2015 12:01:26

MODULO L'ASSAIG				
Nombre:	Luis	Teléfono:	96533434	Móvil:
Apellidos:	García García			686909090
Fecha de la reserva	Plazas	Tipo de reserva	Entrega	Estado
2015-04-30 08:30:17	2	en persona	100.000000000	pendiente
Nombre:	José	Teléfono:	965332323	
Apellidos:	García Sanchez	Móvil:	686774433	
Fecha de la reserva	Plazas	Tipo de reserva	Entrega	Estado
2015-04-30 08:31:47	2	movil	23.000000000	pendiente
Nombre:	Carlos	Teléfono:	965334545	
Apellidos:	Huelmo Vaquero	Móvil:	696123212	
Fecha de la reserva	Plazas	Tipo de reserva	Entrega	Estado
2015-04-30 08:31:47	2			
Nombre:	Francisco	Teléfono:		
Apellidos:	Huelmo Vaquero	Móvil:		
Fecha de la reserva	Plazas	Tipo de reserva	Entrega	Estado
2015-04-30 08:31:47	2			

Página: 1 de 2

Al agrupar por nombre del cliente, en “details” pondremos los campos referidos a la reserva, entonces tendremos los clientes sin reservas y los que no. Vemos que no es necesario en principio realizar subformularios.

No olvidemos indicar dentro de las propiedades del “report” la relación con la llave foránea key, es decir tendremos que indicar una nueva propiedad llamada “**OPENERP\_RELATIONS**” con valor “[‘cliente\_reserva\_id’]” que es la clave ajena que apunta a la tabla reservas, que es donde está la relación many2one, de este modo se llenarán los campos.

Además dentro de la ventana “**Report Query**”, podemos editar expresiones para realizar un filtrado de los datos, en estado del estado de la reserva, por ejemplo: **\$F{Estado-state}!="pendiente" & \$F{Estado-state}!="anulado"**. O bien: **\$F{Estado-state}!=""**, para que muestre sólo los clientes que tienen algún tipo de reserva y omitimos aquellos que nunca han tenido una reserva de cualquier tipo.

Marzo – Junio 2015

Página 65 de 114



## 9.5.- DAR FORMATO / MÁSCARA A LOS CAMPOS DEL INFORME.

### ■ Listado de clientes con reservas (agrupado).

Para el campo “Nombre”, al ser de tipo “String” podemos ir a las propiedades y en el apartado “Text Field Expression” escribir el siguiente código: \${Nombre-nombre}.toUpperCase(), con lo cual nos mostrará todos los valores en mayúsculas. Haremos lo mismo para el campo “Apellidos”.

Para el campo “Teléfono”, podemos cambiar el tipo de dato pero de forma especial para realizar la personalización. En postgres, es un campo de tipo “integer” pero en iReport pasa a ser “string” por lo que tendremos que cambiar (en la vista diseño) el “Expression Class” a “java.lang.Number” mientras que en la vista XML, el class será “java.lang.Integer”. Si el campo (u otros) tuviera valor “null” y no quisieramos que se mostrase en nuestro informe, desde diseño, marcaríamos la opción “blank when null”. Si se mantuviera como “string” y no hubiera datos, no mostraría nada en el informe.

### ■ Listado de Reservas.

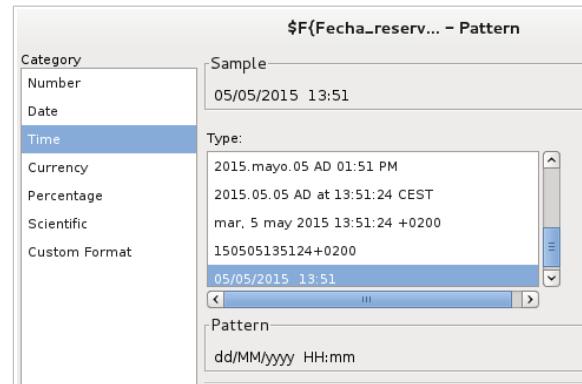
Para el campo “Entrega a cuenta”, en principio es un tipo de dato “String”, mejor dicho, iReport asigna como tipo de dato String a todos los campos que se utilicen para realizar el informe por defecto. Por lo que tendremos que ir a la vista diseño, a las propiedades del campo e indicar que sea de tipo “Double” y después en la vista XML cambiar la clase de ese “field” o campo por “Double” también para que así nos redondee a dos decimales.

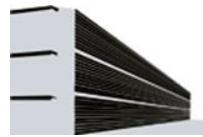
```
<field name="Entrega-entrega" class="java.lang.Double">
    <fieldDescription><![CDATA[ /data/record/Entrega-entrega ]]></fieldDescription>
</field>
```

Además, en las propiedades del campo, buscaremos la opción “Pattern” donde podremos indicar qué tipo de categoría es: número, fecha, hora, etc., o personalizarlo como nosotros queramos (custom format). Con todo esto conseguiremos redondear a dos decimales y que aparezca “euros” después de la cantidad separada por un espacio.

Para el campo “Fecha\_reserva-fecha\_todo” que también es de tipo “String”, tendremos que modificar en la vista diseño su “Expression Class” por “java.sql.Timestamp”, guardar los cambios e ir a la vista XML y en el class cambiarlo por “java.sql.Timestamp”. Y eso que en la base de datos postgresql está como campo “datetime”. Después, en las propiedades del campo, clicamos en “Pattern” e indicamos la categoría “date” pero realizando después nuestros propios cambios desde el “custom format”. Con lo cual mostramos correctamente la fecha completa así como las horas y minutos, por defecto nos mostraba el registro tal cual lo tiene almacenado en la base de datos de postgres. En caso de mantener el tipo de dato, no mostraría los cambios. Otra opción es utilizar el editor de expresiones y hacer “new SimpleDateFormat...” pero ha dado muchos problemas.

### ■ Listado de mesas.





Aquí abordaremos el **mostrar la imagen** que tenemos almacenada **en un campo de tipo binary** ("imagen") en la base de datos postgres (a través de python – ver creación del modelo). Entonces, creado el nuevo documento jrxml, cargado el origen de datos, etc., en la vista diseño, desde la "**Paleta**" arrastraremos y soltaremos el icono "image", después **en las propiedades** del campo "image" iremos a la opción "**image expression**" y añadiremos el siguiente código para que no nos muestre la ruta temporal de cada imagen: **new File(\${Foto-foto\_mesa})**. Como curiosidad, siguiendo en las propiedades de "image", el apartado "Expression Class" figura como java.lang.String.

En el campo "observaciones", hemos contemplado que si el texto excede el tamaño diseñado en el informe, éste no se mostrará completamente sino que "cortará" el texto y no se mostrará el resto para evitarlo tendremos que ir al campo "**\$F{Observaciones-observaciones}**" y en sus propiedades marcar la casilla "**Stretch with overflow**" para que añada líneas del texto hasta mostrarlo completamente.

### ■ Listado de clientes.

Queremos mostrar la fecha "dd/MM/yyyy" que postgres nos lo almacena en formato americano seguido de las horas, minutos y segundos, entonces en vez de cambiar la clase del tipo de dato en las vistas diseño y XML en iReport (String), nos lo haga directamente desde el editor de expresiones del propio campo. Sin cambiar el tipo de dato, escribiremos el siguiente código:

```
new SimpleDateFormat("dd/MM/yyyy").format(new SimpleDateFormat("yyyy-MM-dd").parse($F{Alta-fecha_alta}))
```

Como podemos comprobar la fecha ya no sale al revés, sino que se muestra conforme lo hemos indicado por código en vez de utilizar el "pattern".

**OJO: en el caso de borrar el módulo completamente, tendremos que volver a indicar para cada informe su modelo correspondiente AUNQUE seguirá mostrando los registros según el diseño de dicho informe. Por otra parte, recordemos que si realizamos modificaciones en el modelo, también deberemos hacerlo en cada informe.**



**Listado de clientes**

Fecha de impresión: martes 12 mayo 2015

Nombre	Apellidos	Teléfono	Fecha de registro
Macario	Cerda Jorda	965333434	12/05/2015
Carlos	Huelmo Vaquero	965512323	12/05/2015
Luis	Lopez Perez	965232323	12/05/2015
Martina	Perez Perez	965443434	12/05/2015
Francisco	Sanchez Sanchez	986753412	12/05/2015
Mario	Sevilla Granada		



## 9.6.- REALIZAR UN SUBINFORME.

A parte de agrupar, también tenemos la posibilidad de **realizar subinformes** con iReport. Imaginemos que nos han solicitado un formulario que muestre las reservas realizadas por cada cliente. Lo primero será crear y diseñar un nuevo documento jrxml e indicar que la fuente de datos es la plantilla XML del modelo “assaig.cliente” que ya teníamos generado por Odoo y descargado anteriormente. Seleccionamos el archivo, pulsamos botón derecho del ratón para que aparezca el **menú contextual**, elegir “**Propiedades**” e ir a “**Properties**” (la cuál contendrá 3 por defecto) al pulsar nos mostrará las propiedades: “zoom”, “x” e “y”, nosotros añadiremos “**OPENERP\_RELATIONS**” y el valor será la “clave foránea” - “[‘cliente\_reserva\_id’]” que se encuentra en el **modelo “assaig.cliente”** y que enlaza con el **modelo “assaig.reserva”** en la relación “*one2many*” (también podría ser una relación “*many2many*”) a través del campo “**cliente\_id**”. Es decir, estamos indicando a Odoo qué campo tiene que iterar, reservas del cliente X.

Clientes_con_reservas - Properties	
Name	Value
ireport.zoom	1.0
ireport.x	0
ireport.y	0
OPENERP_RELATIONS	[‘cliente_reserva_id’]

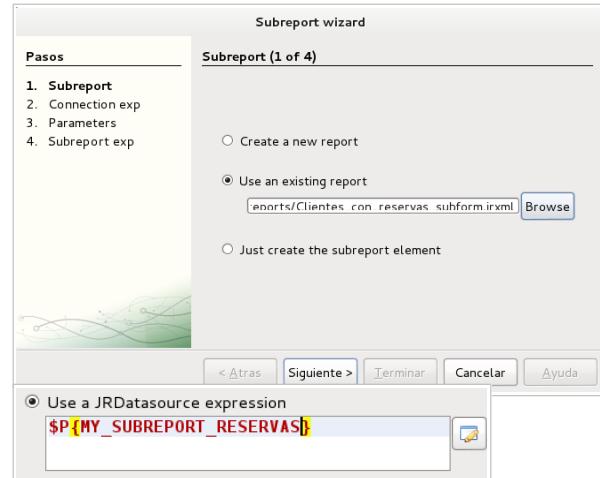
Creamos otro documento que será el “**subreport**” y que cargará los datos a partir de la plantilla XML del modelo “assaig.reserva”, terminado el diseño y el formato, procederemos a añadir en las propiedades del informe otro campo para que itere, el nombre será el mismo “**OPENERP\_RELATIONS**” pero el valor será “[‘cliente\_id’]” ya que hace referencia a los clientes.

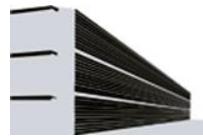
Cuando lo acabemos sólo nos faltará añadirlo al informe principal y enlazarlo. Pasos a seguir: En el report principal, crearemos un nuevo parámetro, por ejemplo “**MY\_SUBREPORT\_RESERVAS**”, en “Parameters” - “Aregar”, modificamos el nombre, en las propiedades del parámetro indicar que sea de la clase “object” (es importante pero aún no sé porqué).

MY_SUBREPORT_RESERVAS - Properties	
Propiedades	
Name	MY_SUBREPORT_RESERVAS
Parameter Class	Object
Use as a prompt	<input checked="" type="checkbox"/>
Default Value Expression	
Description	
Properties	No properties set

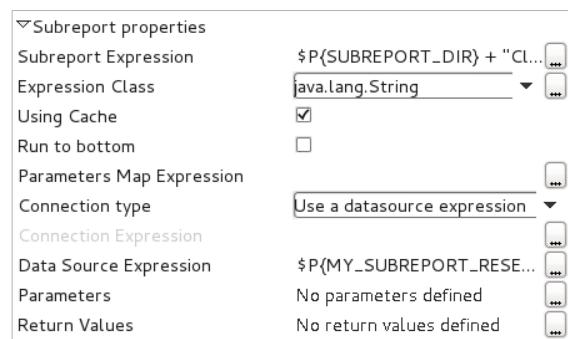
Después, añadiremos al report principal, un subreport desde la “Paleta” en la línea de detalle “**detail 1**”, nos aparecerá un asistente.

- Paso 1, indicaremos la ruta del subinforme que hemos creado.
- Paso 2, indicaremos en el parámetro para la conexión “connection exp” que use “a JRDataSource” e indicamos nuestro parámetro.
- Paso 3, no añadiremos ningún parámetro.





- Paso 4, indicaremos la ruta donde encontrará el subreport, seleccionamos “**Store the directory name in a parameter**”. Creará un nuevo parámetro “SUBREPORT\_DIR” que contiene la ruta del fichero jrxml. Marcaremos también la opción “Use as a prompt” para que se busque el Subreport en la misma ubicación donde está el principal.



The screenshot shows the "Subreport properties" section of the Jasper Reports properties dialog. It includes fields for "Subreport Expression" (\$P{SUBREPORT\_DIR} + "Cl..."), "Expression Class" (java.lang.String), "Using Cache" (checkbox checked), "Run to bottom" (checkbox unchecked), "Parameters Map Expression", "Connection type" (Connection Expression), "Data Source Expression", "Parameters" (No parameters defined), and "Return Values" (No return values defined).



The screenshot shows the Odoo client interface. A report titled "Clientes con reservas" is displayed. It includes fields for Name (Luis) and Phone (965333434). Below it, a sub-report titled "Reservas" is shown with columns for Delivery, Seats, Type of reservation, Reservation date, and Status of the reservation. The sub-report table is currently empty.

Desde la vista tree de “clientes” seleccionamos todos y después seleccionamos imprimir, se imprimirá el informe en todas las hojas pero el subinforme sólo en la primera. **De momento no se ha encontrado una solución al problema.**

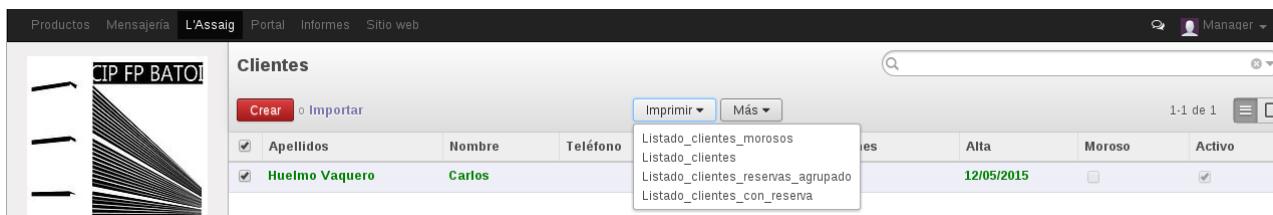
## 9.7.- SEGURIDAD EN INFORMES.

Por defecto, sólo el usuario Administrator (de Odoo) puede imprimir informes, pero queremos que los usuarios de nuestro módulo también puedan imprimir los informes. Entonces el primer paso es ir a cada informe creado y en “grupos” añadirlos desde el módulo “Jasper Reports” - “Jasper Reports” pero no nos aparece el botón de “Imprimir” al seleccionar un registro, **por lo que no sirve**

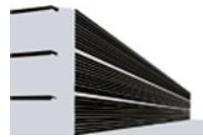
**indicar los grupos.** Entonces la solución que hemos encontrado es editar los permisos del usuario del cuál queremos que pueda imprimir informes, en la pestaña “permisos de acceso” en el apartado “Aplicación” tendremos que indicar que para “**Recursos Humanos**” el valor es “empleado” por tanto éste ya podrá acceder a la lista de informes disponibles dependiendo del modelo en el que se encuentre tanto en la vista “tree” como en “form”.



The screenshot shows the "Grupos" (Groups) settings screen in Odoo. It lists two groups: "Restaurante L'Assaig / Administradores" and "Restaurante L'Assaig / Usuarios".



The screenshot shows the Odoo client interface. A list of clients is displayed with a context menu open over one of the entries. The menu items include "Listado\_clientes\_morosos", "Listado\_clientes", "Listado\_clientes\_reservas\_agrupado", and "Listado\_clientes\_con\_reserva".



## 10.- EXPORTAR E IMPORTAR.

En Odoo también tenemos la posibilidad de **exportar los registros** de que disponemos y que están almacenados en la base de datos. Esto es bastante útil porque si desinstalamos nuestro módulo, perderemos todos los datos que contenga, entonces podemos exportarlos en un fichero CSV o XLS (en Excel) **como medida de seguridad**.

La opción de “exportar” se nos mostrará siempre en la vista “tree” al seleccionar uno o varios registros. Al clicar sobre “Exportar” nos aparecerá una nueva ventana emergente en la cuál indicaremos el tipo de exportación y el formato: CSV o XLS. Elegiremos los campos disponibles (del modelo), los añadiremos y después podemos “exportar fichero” y cerrar. En los “campos a exportar” podemos “guardar la lista de campos” con un nombre que queramos. También nos guardará los campos relacionales “many2one”, “one2many” y “many2many” y su contenido.

Clientes				
Crear o Importar				
	Apellidos	Nombre	Teléfono	Móvil
<input checked="" type="checkbox"/>	Casal Llopis	Jaime	962213456	
<input checked="" type="checkbox"/>	Garcia Garcia	Francisco	965334343	686908978
<input checked="" type="checkbox"/>	Huelmo Vaquero	Carlos	965332323	686784321 Es de Alcoy.
<input checked="" type="checkbox"/>	Lopez Lopez	Jose	986542312	701234567
<input checked="" type="checkbox"/>	Matos Lopez	Martina	913425678	700909090
<input checked="" type="checkbox"/>	Rama Tronco	Luis	933216590	621894532
<input checked="" type="checkbox"/>	Trujillo Vázquez	Mario	984321243	616234309

Tenga en cuenta que todos los registros que cumplen el filtro de búsqueda serán exportados, no sólo los seleccionados.

Tipo de Exportación: Importación-Exportación Compatible ▾ Formatos de Exportación: Excel ▾

**Campos disponibles**

Nombre
Activo
Alta
<b>Apellidos</b>
Created by
Created on
Last Updated by
Last Updated on
Moroso
<b>Móvil</b>
Nombre
Observaciones
▷ Reserva cliente
▷ Reserva cliente2
<b>Teléfono</b>

**Campos a exportar** Guardar lista de campos  
 Guardar como: Aceptar  
 Exportaciones guardadas:

**Suprimir**

- Añadir
- Eliminar
- Eliminar Todo

- Activo
- Alta
- Apellidos
- Created by
- Created on
- Last Updated by
- Last Updated on
- Moroso
- Móvil
- Nombre
- Observaciones

**Guardar archivo**

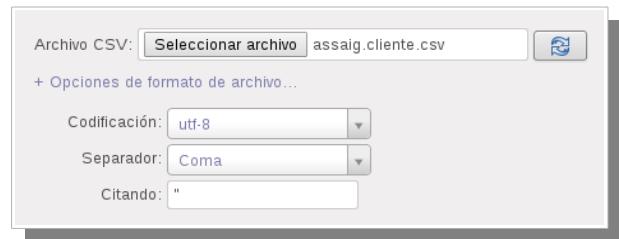
Nombre: assaig.cliente.xls  
 Guardar en la carpeta: < ds Documentos odoo\_exportar\_en\_

Lugares	Nombre
Buscar	assaig.cliente.xls
Usados reciente...	
ds	
Escritorio	

**Asistente de exportaciones en Odoo 8.**



**A la hora de importar**, la opción siempre está disponible pero sólo para archivos con extensión CSV, entonces buscamos el archivo CSV e indicaremos las opciones de formato de archivo. Una vez cargado el contenido del archivo podremos ver los registros y nombres de los campos si creemos que es posible que no todo esté correcto, con pulsar el botón “**validar**” el asistente nos indicará si se han encontrado errores o no. Después de validar, el siguiente paso será importar los datos, pulsamos “**importar**”.



### Mapee sus datos a Odoo

- Registrar historial durante la importación
- La primera fila del archivo contiene las etiquetas de la columnas

Todo parece correcto.

id	activo	fecha_alta	apellidos	create_uid/id	create_date
Id. externo	Activo	Alta	Apellidos	No importar	No importar
_export_.assaig_cliente_6	True	2015-05-13	Casal Llopis		2015-05-13 16:38:09
_export_.assaig_cliente_3	True	2015-05-13	Garcia Garcia		2015-05-13 16:38:09
_export_.assaig_cliente_1	True	2015-05-13	Huelmo Vaquero		2015-05-13 16:38:09
_export_.assaig_cliente_4	True	2015-05-13	Lopez Lopez		2015-05-13 16:38:09
_export_.assaig_cliente_5	True	2015-05-13	Matos Lopez		2015-05-13 16:38:09
_export_.assaig_cliente_7	True	2015-05-13	Rama Tronco		2015-05-13 16:38:09
_export_.assaig_cliente_19	True	2015-05-13	Trujillo Vázquez		2015-05-13 16:45:28

Cuando importamos y tenemos los mismo datos, no se duplican, se modifican. En el caso de que tengamos un campo que establezca una relación seguiremos las indicaciones que nos ofrece el asistente Odoo (durante la exportación/importación). Podemos realizar modificaciones en el archivo pero teniendo en cuenta las restricciones que hemos establecido en el modelo de lo contrario al validar nos puede dar mensajes como: que un registro tiene “**valor null para la columna X y viola la restricción not null...**” y en el modelo es “**required=True**”, también que la “**llave duplicada viola la restricción de unicidad**”, todo esto también dependerá si durante la importación hemos seleccionado “exportación-compatible” o “exportar todos los datos”

el valor null para la columna «ubicacion» viola la restricción not null en la fila 2 (9 más)

```
'ubicacion' : fields.selection([
    ('comedor','Comedor'),('terraza','Terraza')], 'Ubicación', required=True),
```

Llave duplicada viola restricción de unicidad «assaig\_mesa\_codigo\_unique» DETALLE: Ya existe la llave (codigo)=(MESA-01). en la fila 12

```
_sql_constraints = [
    ('codigo_unique','UNIQUE(codigo)','La mesa ya existe.'),
```

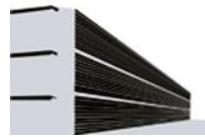


Otro problema que se nos puede presentar es que para poder importar y exportar, el usuario logeado (manager o cambrer8) tiene que pertenecer a “Recursos Humanos” como “empleado” es decir, tendríamos que ir a los permisos de ese usuario y activar la casilla de RRHH, de lo contrario nos dará un mensaje de error:

Es el mismo problema que teníamos a la hora de imprimir informes, también tendremos que hacer miembro a nuestro usuario “manager”, esto puede ser utilizado también como una “medida de seguridad” es decir, sólo podrá exportar e importar quién sea miembro de RRHH.

## 11.- VOCABULARIO.

- **CMS: Content Management System**, programa informático que permite crear una estructura de soporte para la creación y administración de contenidos (como páginas web), por parte de los administradores, editores, participantes y resto de usuarios.
- **ERP: Enterprise Resource Planning**, *llamado sistema de planificación de recursos empresariales, es un sistema de información empresarial que integran y manejan muchos de los negocios asociados con las operaciones de producción y de los aspectos de distribución de una compañía en la producción de bienes o servicios*
- **SI:** es un sistema de recursos, datos y actividades que interactúen entre sí con la finalidad de dar soporte a las actividades de una empresa o negocio: optimizar procesos, proporcionar información relevante para la toma de decisiones y obtener ventajas competitivas.
- **GNU Afferro General Public License:** es una licencia copyleft (método para que un programa sea libre, exigiendo que todas las versiones modificadas y extendidas del mismo sean también libres, manteniendo los derechos de autor) derivada de la Licencia Pública General de GNU.
- **MAKO:** es una plantilla que convierte una secuencia de texto que contiene cualquier tipo de contenido, XML, HTML, etc. Además puede contener directivas que representan la sustitución de variables, expresiones, estructuras de control (bucles, condiciones), comentarios, bloques de código python. Todo esto se compila con código python.
- **MRPS:** Material Requirements Planning System: sistema que integra las actividades de producción y compras.
- **Copyleft:** práctica que consiste en el ejercicio del derecho de autor con el objetivo de permitir la libre distribución de copias y versiones modificadas de una obra u otro trabajo, exigiendo que los mismos derechos sean preservados en las versiones modificadas.
- **XML-RPC:** protocolo de procedimiento de llamada remoto que utiliza XML para codificar los datos y HTTP como protocolo de transmisión de mensajes.
- **SEO:** Search Engine Optimization, posicionamiento en buscadores u optimización en motores de búsqueda, es el proceso de mejora de visibilidad de un sitio web en los resultados de los diferentes buscadores.
- **MRP:** sistema para la planificación eficaz de todos los recursos de una empresa de fabricación.
- **TIC:** son el conjunto de procesos, técnicas y dispositivos que integran funcionalidades de almacén, proceso y transmisión de datos mediante la utilización de hardware y software como un medio de sistema informático.
- **WKHTMLTOPDF:** son Open Source bajo licencia LGPLv3, son herramientas por línea de comandos para transformar o convertir código HTML en formato PDF así como los formatos de varias imágenes utilizando el motor renderizado QT Webkit.
- **SWX:** tipo de archivo utilizado para guardar versiones anteriores de OpenOffice.



## 12.- PROBLEMAS.

Al no haber profundizado lo suficiente durante el curso por problemas de tiempo y calendario disponible, el alumno se ha encontrado con problemas a la hora de desarrollar el módulo, por desgracia durante las 400 horas destinadas a las FCT's:

- Falta de disponibilidad del tutor asignado en la empresa.
- Comenzar con OperERP 6.1.
- Problemas con la plataforma a la hora de compilar, grabar...
- Mínimos conocimientos de OpenERP.
- Uso de las versiones de OpenERP 7 y Odoo 8 junto con las diferentes librerías asociadas.
- Compatibilidad del código de OpenErp 7 en Odoo 8.
- Instalación de Odoo en diferentes sistemas operativos aún siguiendo las instrucciones paso a paso.
- Conocimientos suficientes pero no completos de python 2.7.x.
- Al no recibir otras tareas por parte del instructor, el alumno no conoce el funcionamiento de la empresa ni qué funciones desarrollan los distintos departamentos.



## 13.- FUENTES / ENLACES.

Crear módulo en OpenERP v7:

<http://huber.salazarcarlos.com/?p=445>

(Fecha último acceso: 02/04/2015)

Agregar campos a formularios:

<http://huber.salazarcarlos.com/?p=396>

(Fecha último acceso: 02/04/2015)

Para poner icono a nuestro módulo:

<http://stackoverflow.com/questions/10722909/how-to-add-an-icon-to-an-openerp-custom-developed-module>

(Fecha último acceso: 02/04/2015)

Todo lo relacionado con OpenERP 8:

[http://poncesoft.blogspot.com.es/2014\\_01\\_01\\_archive.html](http://poncesoft.blogspot.com.es/2014_01_01_archive.html)

(Fecha último acceso: 02/04/2015)

Campos de introducción:

<http://poncesoft.blogspot.com.es/2014/04/introduccion-en-la-declaracion-de.html>

(Fecha último acceso: 07/04/2015)

Otros enlaces utilizados en Diagram Software, actualizar automáticamente Odoo 8 :

<http://misnotasdelinux.blogspot.com.es/2014106/instalar-odoo-openerp-80-en-debina.html>

<https://github.com/odoo/odoo>

(Fecha último acceso: 07/04/2015)

Uso modo desarrollador para crear menús, vistas, tablas:

<http://jdeveloper.wikispaces.com/10.7.-+OpenERP+I+BDs,+tablas,+menus,+vistas,+informes>

(Fecha último acceso: 07/04/2015)

Información sobre Odoo 8 y notas de la versión (Fecha primer acceso: 08/04/2015)

<https://www.fonse.net/blog/odoo-openerp-colombia-4/post/odoo-8-notas-de-la-version-3>

Obtención de datos generales y/o específicos (Fecha último acceso: 08/04/2015)

<http://es.wikipedia.org>

[www.infoacp.es/openerp](http://www.infoacp.es/openerp) (Fecha último acceso: 21/04/2015)

Herramientas para realizar traducciones

<http://poedit.net/download> (Fecha último acceso: 23/04/2015)

Realizar traducciones de un módulo en OpenERP 7 (Fecha último acceso 23/04/2015)

<http://poncesoft.blogspot.com.es/2014/01/idioma-personalizado-para-openerp.html>



Documentación de Odoo para realizar traducciones (Fecha último acceso 24/04/2015)  
[https://doc.odoo.com/v6.1/contribute/07\\_improving\\_translations.html](https://doc.odoo.com/v6.1/contribute/07_improving_translations.html)

**E-book: Odoo Development Essentials.**

Autor: Daniel Reis.

Editorial: Packt Publishing Ltd.

ISBN 978-1-78439-279-6.

Páginas: 214.

Abril 2015.

Instalación módulo de diseñador de reportes de OpenOffice/LibreOffice  
<http://huber.salazarcarlos.com/?p=375> (Fecha último acceso: 28/04/2015)

Sobre xPath2:

(Fecha último acceso: 29/04/2015).

[http://www.w3c.es/Prensa/2005/nota051103\\_xslt-xquery-xpath.html](http://www.w3c.es/Prensa/2005/nota051103_xslt-xquery-xpath.html)

Realizar informes con iReport (Fecha último acceso: 30/04/2015)

[www.jc-mouse.net/java/ireport/ireport-reportes-con-imagenes-de-la-base-de-datos](http://www.jc-mouse.net/java/ireport/ireport-reportes-con-imagenes-de-la-base-de-datos)

[www.academia.edu/9763005/Módulo DESARROLLO DE INTERFACES Código\\_0488](http://www.academia.edu/9763005/M%C3%B3dulo_DESARROLLO_DE_INTERFACES_C%C3%B3digo_0488)

(Fecha último acceso: 30/04/2015)

Gráficos en iReport (Fecha último acceso: 30/04/2015)

[www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=graficosiReport](http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=graficosiReport)

Subreports en Odoo (Fecha último acceso: 06/05/2015)

[www.bilbonet.net/es/Document/2014/OpenERP/OpenERP-Factura-Jasper-Report-impuestos-cobros.aspx](http://www.bilbonet.net/es/Document/2014/OpenERP/OpenERP-Factura-Jasper-Report-impuestos-cobros.aspx)

Reportlab-OpenOffice (Fecha último acceso: 07/05/2015)

<http://sahotaparamjitsingh.blogspot.com.es/2012/04/how-to-create-reports-in-openerp.html>

Sintaxis modelo relacional XML, identificación de los números empleados en “eval” para grupos y/o usuarios (Fecha último acceso: 11/05/2015)

<http://stackoverflow.com/questions/26011102/openerp-odoo-model-relationship-xml-syntax>

Crear permisos desde código, añadir varios grupos a un usuario (Fecha último acceso: 12/05/2015)

<http://www.todoopenerp.es/foro/thread-12-post-33.html>

Crear informes mediante diseño en Odoo (Fecha último acceso: 15/05/2015)

<http://odoo.guide/report-design-workshop/>

Añadir dos vistas para un mismo modelo (Fecha último acceso: 15/05/2015)

<http://stackoverflow.com/questions/29731167/create-two-graph-view-for-one-model-openerp>



Más información sobre Odoo. Interesante. (Fecha último acceso: 19/05/2015)  
<https://rm.aloxa.eu/projects/gestion-empresarial/wiki/Modelos>

Paquete todo en uno para instalar Odoo 8. (Fecha último acceso: 28/05/2015)  
<https://bitnami.com/stack/odoo>

Problemas de configuración con PostgreSQL – Bitnami. (Fecha último acceso: 28/05/2015)  
[https://wiki.bitnami.com/Applications/Bitnami\\_PostgreSQL#PostgreSQL\\_configuration\\_files](https://wiki.bitnami.com/Applications/Bitnami_PostgreSQL#PostgreSQL_configuration_files)

Instalación en debian. (Fecha último acceso: 28/05/2015)  
<https://www.odoo.com/documentation/8.0/setup/install.html>



## ANEXO I – Otros editores para realizar traducciones.



**Nombre:** Virtaal.

**Última versión:** 0.7.1

**Proveedor / Fabricante:** Zuza Software Fundation.

**Tipo de Software:** GNU GPL.

**Arquitectura:** 32 – 64 bits.

**Sistema operativo:** Multiplataforma.

**Última actualización:** 13/03/2013.

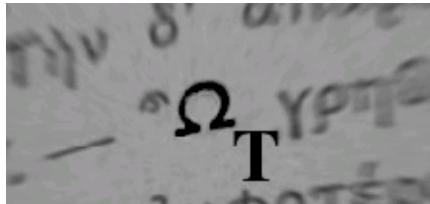
**Observaciones:** ¿Abandonado?

Página web "<http://sourceforge.net/projects/translate/files/Virtaal/>".

En Debian podemos instalarla con el siguiente comando: # apt-get install virtaal.

<p>Plataforma GNU/Linux</p>	<p>Plataforma MS Windows w2003s</p>
-----------------------------	-------------------------------------

The Virtaal application window shows two panes. The left pane displays a list of untranslated strings from a .po file, and the right pane shows the translated versions. The interface includes a toolbar at the top, a navigation bar, and a status bar at the bottom indicating translation progress.

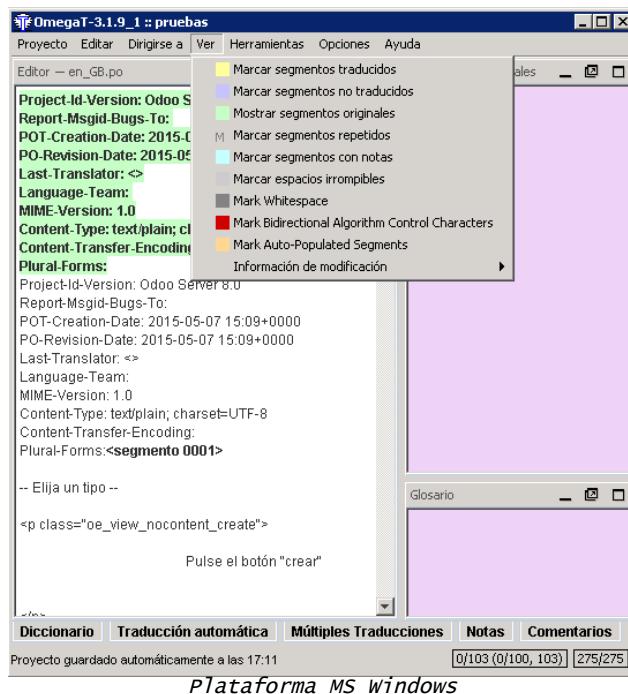


**Nombre:** Omega-T.  
**Última versión:** 3.1.9\_01  
**Proveedor / Fabricante:** Varios.  
**Tipo de Software:** GNU GPL 3  
**Arquitectura:** 32 – 64 bits.  
**Sistema operativo:** Multiplataforma.  
**Última actualización:** 20/04/2015.  
**Observaciones:** Muy completo

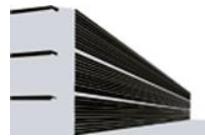
Página web "<http://www.omegat.org/es/omegat.html>".

Las características más relevantes:

- ✓ Coincidencias parciales.
- ✓ Propagación de coincidencias.
- ✓ Procesado simultáneo de proyectos con archivos múltiples.
- ✓ Uso simultáneo de múltiples memorias de traducción.
- ✓ Formatos de archivos: XHTML, HTML, MS Office 2007 XML, OpenOffice/StarOffice, XLIFF, Mediawiki, sólo texto.
- ✓ Soporte Unicode (UTF-8).
- ✓ Soporte para idiomas de izquierda a derecha.
- ✓ Compatible con otras aplicaciones de memoria de traducción (TMX).



Plataforma MS Windows



## ANEXO II - CREACION DE UN MODULO PARA WEBSITE EN OODOO 8.

Los módulos de Odoo 8 para website contienen el mismo esquema que los módulos de Openerp. Procedemos a crear el módulo “Web\_Assaig”, el cual consta de varios campos: dni, nombre, apellidos, teléfono, observaciones y fecha de alta.

Nuestro objetivo será conseguir una página para que desde la web podamos ver todos los clientes almacenados en la base de datos y desde otra habilitaremos un formulario para poder añadir más clientes.

### INSTALACION DEL MODULO “website” (Constructor de sitios web).

Como usuario administrador, iremos a “Configuración”, una vez cargados los módulos que trae por defecto Odoo, dentro de la búsqueda pondremos la palabra clave “website” para que nos muestre todos aquellos módulos (instalados o no) que contengan dicha palabra.



Prestaremos especial atención al módulo “Constructor de sitios web” que se encuentra en la carpeta “website” necesitaremos instalarlo previamente para poder después trabajar con nuestro propio módulo el cuál dependerá de él tanto por las herencias así como las clases heredadas y plantillas “templates”).

Dicho constructor es un **CMS** o **Content Management System**, *programa informático que permite crear una estructura de soporte para la creación y administración de contenidos (como páginas web), por parte de los administradores, editores, participantes y resto de usuarios.* (fuente: Wikipedia).

Este nuevo constructor de sitios web (editor web) y CMS (gestor de contenidos) cuyas principales características son:

- Edición total y en tiempo de ejecución sin interfaz de administración (contenidos, menús...).
- Uso de bloques de construcción para insertar y editar contenidos avanzados(banners, etc.).
- Diseño “fluid grid” donde se puede organizar de forma sencilla los componentes básicos.
- Aparecen los widgets y snippets dinámicos de CMS.
- Administrador de medios (fuentes, iconos, imágenes, videos).



- Soporte para múltiples temas (sólo instalado el de arranque por defecto).
- Templates (plantillas) basadas íntegramente en Bootstrap.
- Integrado con el back-end.

Mejoras en velocidad al resto de CMS (según Google Page Speed):

- Optimización del rendimiento del CMS.
- Menor tamaño y comprensión de JS, HTML y CSS.
- Un único archivo para cargar todos los archivos JavaScript y otro para cargar todos las hojas de estilo CSS, cualesquiera que sean los módulos instalados.
- Optimización de la memoria caché.

Mejoras para el comercio electrónico:

- Dispone de herramientas de optimización SEO<sup>13</sup>.
- Integración con Google Analytics<sup>14</sup>.
- Activación de las funciones de negocio listas para usar con un sólo click.
- Listas de correo, suscripciones, formularios de contacto, boletines, etc.
- “Clean sitemap” (mapa de índices para grandes sitios web).
- Red de integración social.

Adaptación móvil:

- Contenido adaptable de cada página al tamaño de la pantalla de un teléfono móvil, tableta u ordenador. Versión para móvil de forma instantánea.

Una vez finalizada la instalación de dicho módulo podremos ver que aparecerá un nuevo botón llamado “Sitio web” al lado de “Configuración”. Dicho módulo permite crear nuestro propio sitio web cómodamente y sin saber programas mediante el método “drag and drop” (arrastrar y soltar).



13 **Search Engine Optimization**, también conocido como posicionamiento web u optimización en motores de búsqueda, dicho proceso mejora la visibilidad de un sitio web en los resultados generados por los diferentes buscadores.

14 Herramienta analítica web de Google la cual ofrece información agrupada del tráfico que llega a los sitios según la audiencia, adquisición, comportamiento y las conversaciones que se llevan a cabo en el sitio web. Posibilidad de obtener informes de: usuarios, rendimientos del segmento de usuarios, sesiones por fuentes de tráfico, contenidos visitados, etc.



## **CONSTRUCCION DEL MODULO “Web\_Assaig”.**

Cada módulo está contenido en su propio directorio dentro del directorio “`../addons`” en nuestro caso podría ser “`/opt/odoo/addons`” o “`*****`”

Lo primero que haremos será crear una carpeta con el nombre de “`Web_Assaig`” en cuyo interior dispondremos la siguiente estructura:

**`__openerp__.py`** → fichero python que contiene un diccionario con toda la descripción del módulo. También conocido como “manifest”.

**`__init__.py`** → indica que módulos debe importar.

**`./views`** → carpeta que contiene los archivos con las vistas así como plantillas utilizadas en nuestro CMS.

**`./views/web_assaig_view.xml`** → archivo con las vistas: tree, form, search, kanban, etc.

**`./views/templates.xml`** → plantillas o vistas para mostrar los datos almacenados

**`./static/src`** → contiene a su vez otras carpetas para imágenes “`/img`” para javascript “`/js`” o plantillas CSS “`/css`”. Para mostrar las imágenes por ejemplo se indicarán mediante la ruta relativa, teniendo como raíz, la carpeta del propio módulo.

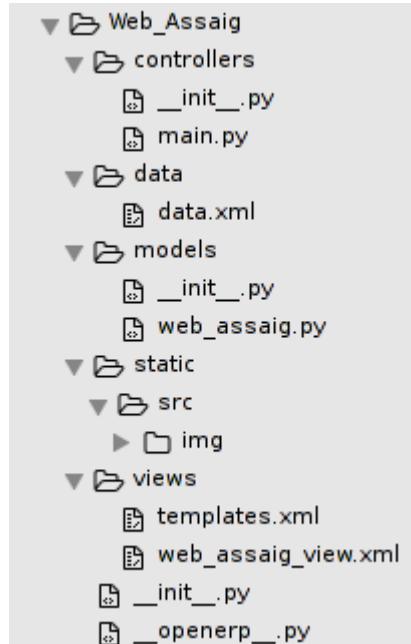
**`./models`** → contiene dos archivos python, uno para indicar qué archivo debe importar Odoo y el otro es donde se define el modelo que se mostrará posteriormente en las vistas definidas en el archivo XML.

**`./models/__init__.py`** → indica el archivo a importar.

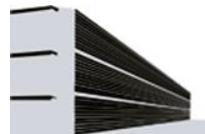
**`./models/web_assaig.py`** → contiene los campos usados para definir lo que el modelo puede almacenar y dónde. Contiene los objetos (que luego serán las tablas de la base de datos) que son declarados como clases de python extendiendo el modelo el cual integra en ellos de forma automatizada la persistencia del sistema.

**`./data/data.xml`** → sirve para crear un apartado en el menú del sitio web para que redirija la petición.

**`./controllers/main.py`** → para obtener funcionalidad de/los template/-s creados, especificaremos directorios y páginas mediante llamadas http.



*Imagen: estructura de un módulo en odoo 8.*



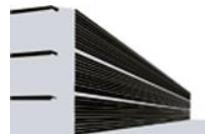
Ahora veremos el contenido de cada archivo:

Indicamos qué debe importar Odoo, es decir lo envía a las carpetas con ese nombre, donde en su interior contendrá otros archivos “`__init__.py`” que también indicarán archivos o carpetas a importar

```
__init__.py ×
1 import controllers
2 import models
```

```
--openerp--.py ×
1 {
2     'name' : "CHV WEBSERVICE ASSAIG",
3     'icon' : "/Web_Assaig/static/src/img/lassaig_logo_web.jpg",
4     'image-icon' : ['/Web_Assaig/static/src/img/lassaig_logo.jpg'],
5     'version' : "0.11",
6     'author' : "Carlos Huelmo Vaquero",
7     'website' : "https://moodle.cipfpbatoi.es",
8     'category' : "un website odoo 8",
9     'summary' : "Importando el modulo a tipo web service",
10    'description' : """
11
12 DAM 2014- 2015
13 =====
14
15 Creación de un website en odoo 8 para gestionar las mesas
16 y permitir realizar reservas...
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31     * Si se requiere de un wizard, tanto el .py como el .xml irían en la carpeta
32     """
33     'depends' : ['website'],
34     'init_xml' : [],
35     'demo_xml' : [],
36     'update_xml' : [],
37     'data' : [
38         'views/web_assaig_view.xml',
39         'views/templates.xml',
40         '#   'views/snippets.xml',
41         '#   'security/ir.model.access.csv',
42         'data/data.xml',
43     ],
44     'qweb' : ['static/src/xml/*.xml'],
45     'installable' : True,
46     'application' : True,
47     'auto_install' : False,
48
49 }
```

- El manifest contiene el nombre “`name`” del módulo, que posteriormente aparecerá en la lista de módulos disponibles al cargar la configuración.
- Icono a mostrar del módulo o “`icon`” que apunta a una ruta absoluta.
- Otros datos a mostrar como la versión, autor, página web, resumen, descripción.



- Dependencia “depends” de nuestro módulo, en nuestro caso, si el módulo “website” no está instalado, no se podrá instalar nuestro módulo.
- Los datos “data” es donde indicamos mediante rutas relativas dónde se encuentran las vistas, plantillas, permisos, redirigir peticiones, etc., en el momento de la instalación del módulo.
- Si es instalable (“installable”) nuestro módulo, en caso de no ponerlo, el valor por defecto es “true”.

### Los modelos.

`./models/__init__.py >>` indica que se importe el archivo “web\_assaig.py” que se encuentra en la misma carpeta.

```
__init__.py ✘
1 import web_assaig
```

El contenido del archivo comienza con la declaración para que podamos escribir caracteres especiales como acentos, etc. Después se importan las librerías necesarias utilizando “from” e “import” tal y como se muestra en la imagen.

```
web_assaig.py ✘
1 # -*- coding: utf-8 -*-
2
3 from openerp.osv import osv, fields, orm
4 from datetime import date
5 from dateutil import relativedelta
6 import time
7 from dateutil import parser
8 from types import *
9 # Para mensajes de log
10 import logging
11 _logger = logging.getLogger(__name__)
12 # Precisión para la entrega
13 import openerp.addons.decimal_precision as dp
14
```

Definimos el objeto “cliente”:



```

16  class cliente(osv.osv):
17
18      def _nom_apell_fnc(self, cr, uid, ids, fields, args, context):
19          res = {}
20          for r in self.browse(cr, uid, ids, context=context):
21              res[r.id] = "%s, %s" % (r.apellidos, r.nombre)
22          return res
23
24      _name = "webassaig.cliente"
25      _rec_name = "name"
26      _description = "Tabla de clientes"
27
28      _columns = {
29          'name' : fields.char('DNI', size=12,
30                                help="Introducir DNI >>> 99.999.999-X", required=True),
31          'nombre' : fields.char('Nombre',size=40,
32                                help="Nombre del cliente.", required=True),
33          'apellidos' : fields.char('Apellidos', size=100,
34                                help="Apellidos del cliente.", required=True),
35          'telefono' : fields.char('Teléfono / Móvil',size=21,
36                                help="Ejemplos >>> Teléfono (99)999 99 99\nMóvil 999 99 99 99", required=True),
37          'fecha_alta' : fields.date('Alta'),
38          'observaciones' : fields.text('Observaciones', size_row=5, size_column=100),
39          'cliente_reserva_id' : fields.one2many('webassaig.reserva','cliente_id', 'Reserva cliente'),
40          'nom_apell' : fields.function(_nom_apell_fnc,
41   type="char", size=140, string="Apellidos, Nombre", readonly=True),
42      }
43
44      _defaults = {
45          'fecha_alta' : lambda *a : time.strftime("%Y-%m-%d"),
46          'observaciones' : 'Escriba lo que proceda...',
47      }
48
49      _sql_constraints = [
50          ('name_unique','unique(name)', 'El DNI debe ser único'),
51      ]
52
53      _order = "apellidos, nombre"
54

```

**\_name:** será el nombre que tendrá la tabla de la base de datos dónde se almacenarán los campos que indiquemos como columnas y sus datos posteriormente.

**\_rec\_name:** sobreescrito del campo especial “\_name” como refuerzo del campo “\_name” para las vistas y búsquedas.

**\_description:** descripción de la tabla – objeto.

**\_columns:** son los campos que tendrá la tabla, indicando el tipo de dato (char, text, boolean, date, integer, float), un texto de ayuda mediante el argumento “help”, “size” tamaño del campo que recogerá el valor, si se obliga a escribir datos o no mediante “required”, entre comillas simples o bien mediante el argumento “string” aparecerá el texto a mostrar del campo en las diferentes vistas.

**\_defaults:** son los valores que mostrarán los campos/columnas indicados teniendo en cuenta el tipo de dato y su valor, en este ejemplo, el campo “fecha\_alta” mostrará la fecha del sistema. El campo “observaciones” mostrará un texto que si no modificamos en la edición, se guardará.

**\_sql\_constraints:** nos sirve para controlar qué campo y sus valores correspondientes deben ser únicos, es decir, que no se permiten duplicados, en caso afirmativo mostrará un mensaje.

**\_order:** indicamos que muestre los datos primero por apellidos y después por nombre de forma ascendente.

También tenemos **un método** para que concatene en otro campo los valores de los campos nombre y apellidos:

```

18     def _nom_apell_fnc(self, cr, uid, ids, fields, args, context):
19         res = {}
20         for r in self.browse(cr, uid, ids, context=context):
21             res[r.id] = "%s, %s" % (r.apellidos, r.nombre)
22         return res

```

Dicho método se llama desde el campo “**nom\_apell**” el cuál es de tipo “function” (función) donde se pondrá el nombre del método “**\_nom\_apell\_fnc**” como tipo de dato carácter ya que los otros dos campos también lo son, que muestre el título de la etiqueta y que sea de sólo lectura para que el usuario no lo pueda modificar.

```

'cliente_reserva_id' : fields.one2many('webassaig.reserva','cliente_id',
    'nom_apell' : fields.function(_nom_apell_fnc,
        type="char", size=140, string="Apellidos, Nombre", readonly=True),
}

```

Después tenemos un “campo relacional” en el campo “cliente\_reserva\_id” que apunta al objeto “webassaig.reserva” que contiene un campo “cliente\_id” y se le añade un texto. Es una relación de uno a muchos, en nuestro caso el cliente hace una o muchas reservas:

```

'fecha_alta' : fields.date('Alta'),
'observaciones' : fields.text('Observaciones', size_row=5, size_column=100),
'cliente_reserva_id' : fields.one2many('webassaig.reserva','cliente_id', 'Reserva cliente'),
'nom_apell' : fields.function(_nom_apell_fnc,
    type="char", size=140, string="Apellidos, Nombre", readonly=True),

```

Definimos el objeto llamado “mesa”:

```

56 class mesa(osv.osv):
57     _name = "webassaig.mesa"
58     _description = "Tabla de mesas"
59
60     _columns = {
61         'name' : fields.char('Codigo', size=7,
62             help="Por ejemplo: MESA-XX", required=True),
63         'plazas' : fields.integer('Plazas',size=2,
64             help="Número de comensales\nMínimo 2 personas.", required=True),
65         'disponible': fields.selection([('si','Si'),('no','No')],'Disponible', required=False),
66         'disponible2' : fields.boolean('Disponible'),
67     }
68
69

```

Tanto en el objeto “cliente” como en “mesa”, utilizamos el campo ‘name’ porque es una palabra reservada que muestra el valor de ese campo como identificador principal, de otra forma, aparecería el valor correspondiente al OUID o identificador general que da postgres a cada valor introducido en la base de datos.

El campo ‘disponible’ es de tipo selección, esto es que aparezca un desplegable mostrando los valores indicados.

El campo ‘disponible2’ es de tipo boolean lo que mostrará en las vistas que sea clicable o no.



Como valores por defecto, indicamos que 'name' cuya etiqueta a mostrar es “Código” mostrará un texto, mientras que “Plazas” siempre tendrá un valor entero por defecto.

```

70   _defaults = {
71     'name' : 'MESA-0',
72     'plazas' : 2,
73   }

```

También controlamos que el “Código” sea único evitando duplicados:

```

75   _sql_constraints = [
76     ('name_unique', 'unique(name)', 'El código de la mesa debe ser único.'),

```

Definimos el objeto llamado reserva:

```

79 class reserva(osv.osv):
80   _name="webassaig.reserva"
81
82   _description="Tabla de reservas"
83
84   _columns ={
85     'tipo_reserva' : fields.selection([
86       ('movil','Móvil'),
87       ('telefono','Teléfono'),
88       ('email','Email')]),
89     'Tipo de Reserva', required=True),
90     #'entrega' : fields.float('Entrega', digits=(6,2),
91     #  help="Euros entregados por el cliente."),
92     'entrega' : fields.float('Entrega', digits_compute=dp.get_precision('Entrega'),
93     help="Euros entregados por el cliente."),
94     'plazas' : fields.integer('Plazas', digits=(2),
95     help="Número de comensales.", required=True),
96     'fecha_alta' : fields.date('Alta', required=True),
97     'fecha_todo' : fields.datetime('Todo'),
98     'widget_hora' : fields.float('Widget de la hora'),
99     'observaciones' : fields.text('Observaciones', size_row=5, size_column=100),
100    'cliente_id' : fields.many2one('webassaig.cliente','Cliente Seleccionado',
101      select=True, required=True),
102  }
103
104   _defaults = {
105     'plazas' : 2,
106   }

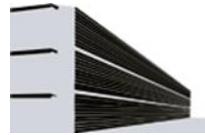
```

Los nuevos campos a explicar serían 'entrega' donde sería de tipo float, pudiendo limitarlo mediante dos decimales y de tamaño seis o bien importando la librería “openerp.addons.decimal\_precision”.

El campo 'fecha\_alta' es de tipo date que mostrará la fecha según la vista.

El campo 'fecha\_todo' es la ampliación del tipo date pero mostrando fecha y hora que también dependerá de la vista.

El campo 'widget\_hora' será de tipo float pero en la vista formulario indicaremos que el widget sea “float\_time” para que lo muestre como “horas:minutos”.

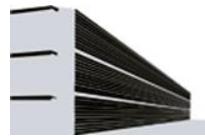


El campo relacional 'cliente\_id' será un campo “many2one” que apuntará al objeto “webassaig.cliente”, con el título “Cliente Seleccionado”, seleccionable y requerido. Para establecer la relación entre los dos objetos se han utilizado los atributos en los dos campos de dichas tablas para poder mostrar la relación porque de lo contrario no mostraría nada.

Al final del archivo y fuera de los objetos procedemos a llamarlos para inicializarlos.

```
111 cliente()
112 mesa()
113 reserva()
```

Con la definición de los tres objetos intentamos abstraer del mundo real la relación entre los clientes que realizan reservas y el mantenimiento de las mesas mediante el módulo correspondiente.

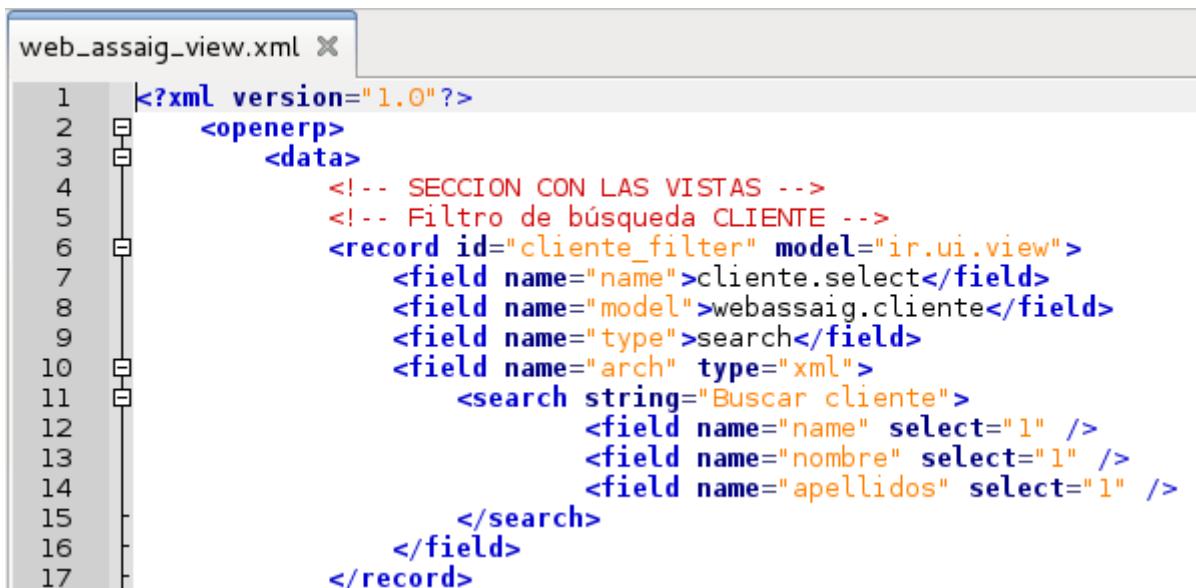


### Las vistas.

Dentro de la carpeta “..../views” tendremos el archivo “**web\_assaig\_view.xml**” de las vistas: tree (árbol), form (formulario), search (filtro de búsqueda), etc. El archivo xml ha sido ordenado primero con las vistas search, después vista formulario, vista árbol (de cada objeto) y el menú junto con sus submenus.

La estructura del archivo XML será el siguiente, mostrando un ejemplo de cada uno de los objetos utilizados como modelos de cada vista utilizada:

Vista “search” para “cliente”:



```

web_assaig_view.xml ✘
1  <?xml version="1.0"?>
2  <openerp>
3  <data>
4      <!-- SECCION CON LAS VISTAS -->
5      <!-- Filtro de búsqueda CLIENTE -->
6      <record id="cliente_filter" model="ir.ui.view">
7          <field name="name">cliente.select</field>
8          <field name="model">webassaig.cliente</field>
9          <field name="type">search</field>
10         <field name="arch" type="xml">
11             <search string="Buscar cliente">
12                 <field name="name" select="1" />
13                 <field name="nombre" select="1" />
14                 <field name="apellidos" select="1" />
15             </search>
16         </field>
17     </record>

```

- Línea 6: identificador de la vista search indicando el modelo a utilizar con valor obligatorio “ir.ui.view”.
- Línea 7: nombre de la vista.
- Línea 8: objeto al que hace referencia para mostrar los datos de la búsqueda.
- Línea 9: tipo de vista, en este caso “search” (filtro de búsqueda).
- Línea 10: arquitectura de la vista XML.
- Línea 11: cadena a mostrar (en Odoo 8 no aparece).
- Líneas 12-14: campos a mostrar para realizar el filtro de búsqueda, que se corresponden con las columnas del objeto cliente definido como modelo. El campo “name” hace referencia al DNI.

El filtro sólo aparecerá en la vista “tree”, podemos ver como resultado que al introducir un valor nos indica a qué debe asociarlo para mostrar el resultado.





Partiendo de que tenemos los siguientes clientes (vista tree):

CLIENTE							
<a href="#">Crear</a> o <a href="#">Importar</a>		1-3 de 3					
	DNI	Nombre	Apellidos	Apellidos, Nombre	Teléfono / Móvil	Observaciones	Alta
<input type="checkbox"/>	22543766M	Ulises	Figerola Sanchez	Figerola Sanchez, Ulises	912345673 / 617890990	Es de Albaida.	09/04/2015
<input type="checkbox"/>	123123123	carlos	huelmo	huelmo, carlos	965433456	Ninguna observación	08/04/2015
<input type="checkbox"/>	21672532T	Carlos	Huelmo Vaquero	Huelmo Vaquero, Carlos	965333838 / 686202020	Escriba lo que proceda...	09/04/2015

Por ejemplo indicaremos que “Buscar Nombre” y obtendremos como resultado a Ulises:

CLIENTE							
<a href="#">Crear</a> o <a href="#">Importar</a>		1-1 de 1					
	DNI	Nombre	Apellidos	Apellidos, Nombre	Teléfono / Móvil	Observaciones	Alta
<input type="checkbox"/>	22543766M	Ulises	Figerola Sanchez	Figerola Sanchez, Ulises	912345673 / 617890990	Es de Albaida.	09/04/2015

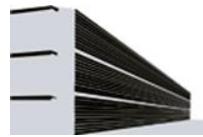
Vista “form” para “cliente”:

```

47 <record model="ir.ui.view" id="cliente_form">
48   <field name="name">CLIENTE</field>
49   <field name="model">webassaig.cliente</field>
50   <field name="type">form</field>
51   <field name="arch" type="xml">
52     <form string="Cliente">
53       <sheet>
54         <group col='4' colspan='2'>
55           <field name="name" />
56           <field name="nombre" />
57           <field name="apellidos" />
58           <field name="nom_apell" />
59           <field name="telefono" />
60           <field name="observaciones" />
61           <field name="fecha_alta" readonly="1" />
62         </group>
63       </sheet>
64     </form>
65   </field>
66 </record>
67

```

- Línea 49: la vista tendrá como modelo el objeto cliente para mostrar los datos.
- Línea 50: indicamos la vista “form” (formulario).
- Línea 52: texto que mostrará como cabecera al seleccionar la vista.
- Línea 53: se utiliza la etiqueta para que el contenido se muestre dentro de una hoja.
- Líneas 54 – 61: se agrupan los campos junto con sus etiquetas en 4 columnas, cada campo tendrá dos columnas (etiqueta y valor). El atributo “readonly” es otra forma de impedir que el usuario pueda modificar el valor de dicho campo (en el modelo también se emplea “readonly=True”).



Al crear un nuevo cliente, la vista formulario se presenta de la siguiente forma.

**CLIENTE / Nuevo**

Guardar o Descartar

Introducir DNI >>> 99.999.999-X

DNI	23456987P	Nombre	Jose Antonio
Apellidos	Aljaro Perez	Apellidos, Nombre	
Teléfono / Móvil	965431212 / 671234567	Observaciones	Escriba lo que proceda...
Alta	09/04/2015		

- En la esquina superior derecha, podemos ver que la vista seleccionada es formulario.
- En la etiqueta “DNI”, aparece un texto de ayuda, al situar el ratón encima, dicho texto se había definido previamente en “/models/web\_assaig.py” con el atributo “*help="texto"*”. Dicho texto de ayuda también se muestra en la vista tree.
- Los campos sombreados no pueden quedar vacíos, esto se consigue desde el código python en “/models” mediante el atributo “*required=True*”.
- La etiqueta “Alta” muestra la fecha del sistema y no es posible modificarlo ya que en la vista se ha indicado con el atributo “*readonly=1*”.
- Al guardar los datos del cliente, el campo “Apellidos, Nombre” se actualiza y muestra la concatenación de los apellidos y nombre. También podemos ver que el identificador es el DNI del cliente introducido, aparece en la parte superior izquierda (imagen inferior).

**CLIENTE / 23456987P**

Editar Crear Más ▾

4 / 4

DNI	23456987P	Nombre	Jose Antonio
Apellidos	Aljaro Perez	Apellidos, Nombre	Aljaro Perez, Jose Antonio
Teléfono / Móvil	965431212 / 671234567	Observaciones	Escriba lo que proceda...
Alta	09/04/2015		



Por lo que se refiere a la vista “tree” de mesas, la estructura XML es:

```

128      <!-- Declarar vista arbol MESAS -->
129      <record model="ir.ui.view" id="mesa_tree">
130          <field name="name">MESAS</field>
131          <field name="model">webassaig.mesa</field>
132          <field name="type">tree</field>
133          <field name="arch" type="xml">
134              <tree string="Mesas">
135                  <field name="name" />
136                  <field name="plazas" />
137                  <field name="disponible" />
138                  <field name="disponible2" />
139              </tree>
140          </field>
141      </record>
142

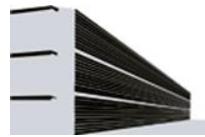
```

- Línea 132: tipo de vista, en este caso “tree”.
- Líneas 135 – 138: campos a mostrar en la vista.

Al cargar la vista:

MESA			
<a href="#">Crear</a>	<a href="#">Importar</a>		
Código	Plazas	Disponible	Disponible
<input type="checkbox"/> MESA-01	2 Si	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
<input type="checkbox"/> MESA-02	4 Si	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
<input type="checkbox"/> MESA-03	3 Si	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

- Podemos ver el texto de ayuda al situar el ratón en la columna “plazas”.



Podemos construir el menú del módulo donde queramos, lo primero será mostrar el nombre del módulo en el panel superior de Odoo, también podemos indicar que al pulsar sobre el “botón” del menú se produzca una acción, la cual tendremos que haber definido tambien en el archivo xml.

```
<!-- MENU -->
<menuitem id="menu_assaig_main" name="WEBSITE ASSAIG" action="action_cliente" />
```

Al no incluir el atributo “parent” por defecto se establece que es un menú principal.



Cuando se instale el módulo aparecerá el botón que mostrará el nombre del módulo.

Después crearemos la navegación por medio de menús.

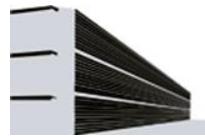
```
<menuitem id="menu_assaig_cliente" parent="menu_assaig_main" name="Gestionar Clientes" />
<menuitem id="menu_assaig_cliente_alta" parent="menu_assaig_cliente" name="Clientes" action="action_cliente" />

<menuitem id="menu_assaig_mesa" parent="menu_assaig_main" name="Gestionar Mesas" />
<menuitem id="menu_assaig_mesa_alta" parent="menu_assaig_mesa" name="Mesas" action="action_mesa" />

<menuitem id="menu_assaig_reserva" parent="menu_assaig_main" name="Gestionar Reservas" />
<menuitem id="menu_assaig_reserva_alta" parent="menu_assaig_reserva" name="Reservas" action="action_reserva" />
```



Cada menú / submenú tendrá su identificador, mediante el atributo “parent” estableceremos los hijos y subhijos del menú que se mostrará en la parte izquierda del módulo. Además de establecer el nombre del submenú también añadiremos una acción a realizar cuando se pulse sobre el nombre.



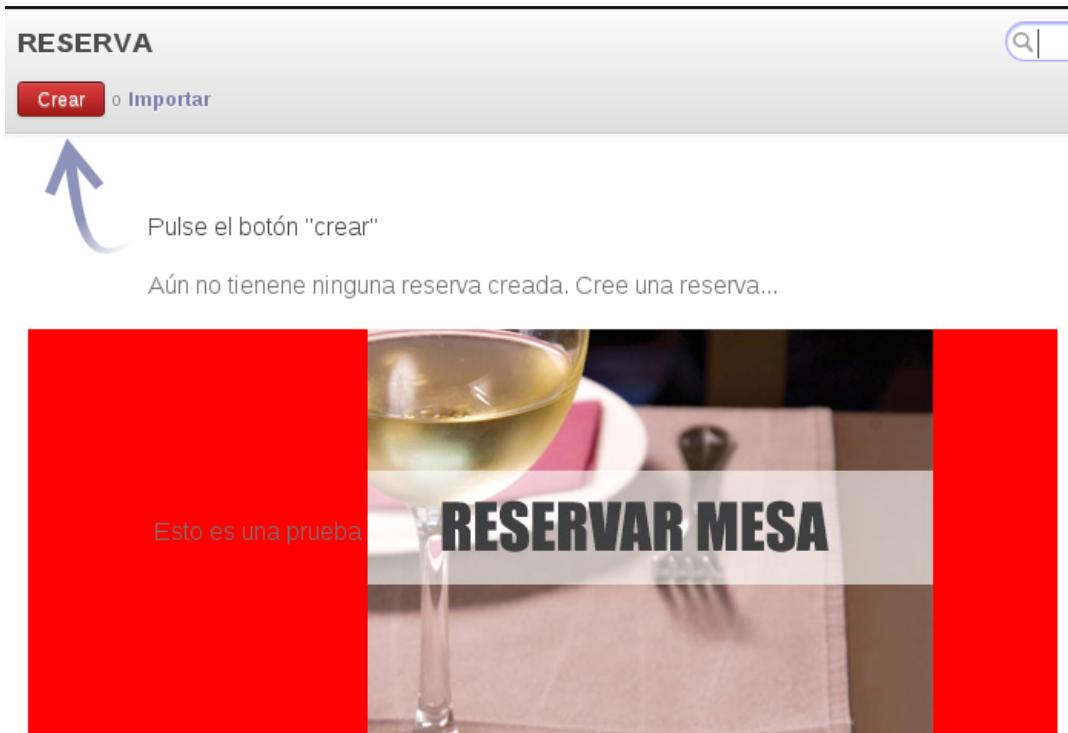
La siguiente parte será “definir las acciones”. En principio para el módulo, son prácticamente iguales:

```

210      <!-- ACCIONES A REALIZAR RESERVA-->
211      <record model="ir.actions.act_window" id="action_reserva">
212          <field name="name">RESERVA</field>
213          <field name="res_model">webassaig.reserva</field>
214          <field name="view_type">form</field>
215          <field name="view_mode">tree,form,calendar</field>
216          <field name="help" type="html">
217              <p class="oe_view_nocontent_create">
218                  Pulse el botón "crear"
219              </p>
220              <p>Aún no tiene ninguna reserva creada. Cree una reserva...</p>
221              <div style="background-color: red" align="center">Esto es una prueba
222                  
223              </div>
224          </field>
225      </record>

```

- Línea 211: indicamos el modelo de acción que trae Odoo por defecto.
- Línea 213: indicamos que la acción se realice sobre el objeto “webassaig.reserva”.
- Línea 214: tipo de vista seleccionada, formulario.
- Línea 215: lista ordenada de las vistas que se mostrarán.
- Línea 216 – 224: aparece la imagen de una flecha por defecto de Odoo cuando no hay datos que mostrar (definido con el atributo **class**) además como se indica que es **type="html"** podemos añadir etiquetas como si se tratase de una página web, podemos así incrustar javascript, fotografía, etc. Sea cual sea la vista se mostrará la imagen más texto, etc.





La estructura de la vista formulario para “reservas” queda como sigue:

```

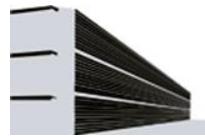
87      <!-- Declarar vista formulario RESERVAS -->
88      <record model="ir.ui.view" id="reserva_form">
89          <field name="name">RESERVAS</field>
90          <field name="model">webassaig.reserva</field>
91          <field name="type">form</field>
92          <field name="arch" type="xml">
93              <form string="Reservas">
94                  <sheet>
95                      <group col='2' colspan='2'>
96                          <field name="cliente_id" />
97                          <field name="tipo_reserva" />
98                          <field name="entrega" />
99                          <field name="plazas" />
100                         <field name="fecha_alta" />
101                         <field name="fecha_todo" />
102                         <field name="observaciones" />
103                         <field name="widget_hora" widget="float_time" />
104                     </group>
105                 </sheet>
106             </form>
107         </field>
108     </record>
```

- Línea 103: al campo “widget\_hora” se le añade el atributo referente al widget para que nos lo muestre en formato horas y minutos (hh:mm).

Cuando creamos una reserva, en vista formulario la podemos llenar con los siguientes datos:

Cliente Seleccionado	22543766M
Euros entregados por el cliente	Teléfono
Tipo de Reserva	
Entrega	0,00
Plazas	2
Alta	09/04/2015
Todo	09/04/2015 16:27:15
Observaciones	Preparar cuberteria de plata.
widget de la hora	16:27

- Los campos sombreados son obligados rellenar con datos.
- Las etiquetas “Alta” y “Todo” muestran los widgets por defecto de Odoo, son un selector de fecha y otro de fecha y hora.
- En “tipo de reserva” es un selector o desplegable dónde podremos seleccionar los distintos valores: Teléfono, móvil, email, etc.



Al tener establecida la relación con el objeto “clientes” desde “reserva” y viceversa, podemos ver todos los clientes dados de alta, aunque también tenemos la posibilidad de crear y editar.

Cliente Seleccionado	22543766M
Tipo de Reserva	23456987P
Entrega	22543766M
Plazas	123123123
Alta	21672532T
<a href="#">Crear y editar...</a>	

En la etiqueta “Alta” nos aparecerá el widget (proporcionado de forma automática) para seleccionar el mes, año y día. Recordemos que en el modelo especificamos que era 'fields.date'.

Alta	09/04/2015																																																
Todo	abr 2015																																																
Observaciones	widget de la hora																																																
<table border="1"> <tr> <td>Sem</td> <td>lu</td> <td>ma</td> <td>mi</td> <td>ju</td> <td>vi</td> <td>sá</td> <td>do</td> </tr> <tr> <td>14</td> <td></td> <td></td> <td>1</td> <td>2</td> <td>3</td> <td>4</td> <td>5</td> </tr> <tr> <td>15</td> <td>6</td> <td>7</td> <td>8</td> <td>9</td> <td>10</td> <td>11</td> <td>12</td> </tr> <tr> <td>16</td> <td>13</td> <td>14</td> <td>15</td> <td>16</td> <td>17</td> <td>18</td> <td>19</td> </tr> <tr> <td>17</td> <td>20</td> <td>21</td> <td>22</td> <td>23</td> <td>24</td> <td>25</td> <td>26</td> </tr> <tr> <td>18</td> <td>27</td> <td>28</td> <td>29</td> <td>30</td> <td></td> <td></td> <td></td> </tr> </table>		Sem	lu	ma	mi	ju	vi	sá	do	14			1	2	3	4	5	15	6	7	8	9	10	11	12	16	13	14	15	16	17	18	19	17	20	21	22	23	24	25	26	18	27	28	29	30			
Sem	lu	ma	mi	ju	vi	sá	do																																										
14			1	2	3	4	5																																										
15	6	7	8	9	10	11	12																																										
16	13	14	15	16	17	18	19																																										
17	20	21	22	23	24	25	26																																										
18	27	28	29	30																																													
<input type="button" value="Hoy"/> <input type="button" value="Hecho"/>																																																	

En el caso de la etiqueta “Todo” también se nos proporciona de forma automática el widget pero además podremos seleccionar las horas y los minutos.

Todo	09/04/2015 16:27:15																																																
Observaciones	widget de la hora																																																
<table border="1"> <tr> <td>Sem</td> <td>lu</td> <td>ma</td> <td>mi</td> <td>ju</td> <td>vi</td> <td>sá</td> <td>do</td> </tr> <tr> <td>14</td> <td></td> <td></td> <td>1</td> <td>2</td> <td>3</td> <td>4</td> <td>5</td> </tr> <tr> <td>15</td> <td>6</td> <td>7</td> <td>8</td> <td>9</td> <td>10</td> <td>11</td> <td>12</td> </tr> <tr> <td>16</td> <td>13</td> <td>14</td> <td>15</td> <td>16</td> <td>17</td> <td>18</td> <td>19</td> </tr> <tr> <td>17</td> <td>20</td> <td>21</td> <td>22</td> <td>23</td> <td>24</td> <td>25</td> <td>26</td> </tr> <tr> <td>18</td> <td>27</td> <td>28</td> <td>29</td> <td>30</td> <td></td> <td></td> <td></td> </tr> </table>		Sem	lu	ma	mi	ju	vi	sá	do	14			1	2	3	4	5	15	6	7	8	9	10	11	12	16	13	14	15	16	17	18	19	17	20	21	22	23	24	25	26	18	27	28	29	30			
Sem	lu	ma	mi	ju	vi	sá	do																																										
14			1	2	3	4	5																																										
15	6	7	8	9	10	11	12																																										
16	13	14	15	16	17	18	19																																										
17	20	21	22	23	24	25	26																																										
18	27	28	29	30																																													
<b>Tiempo</b> 16:27 <b>Hora</b> <input type="button" value="Ahora"/> <input type="button" value="Hecho"/> <b>Minuto</b> <input type="button" value="Ahora"/> <input type="button" value="Hecho"/>																																																	



Por otra parte hemos definido la vista “calendar” para las reservas:

```

161
162
163
164
165
166
167
168
169
170
171
172
173
174
    <!-- DECLARAR VISTA CALENDAR RESERVAS -->
    <record model="ir.ui.view" id="reserva_calendar">
        <field name="name">CALENDARIO RESERVAS</field>
        <field name="model">webassaig.reserva</field>
        <field name="type">calendar</field>
        <field name="arch" type="xml">
            <calendar string="Calendario" color="cliente_id"
                date_start="fecha_todo" mode="month">
                <field name="cliente_id" />
                <field name="fecha_alta" />
                <field name="fecha_todo" />
            </calendar>
        </field>
    </record>

```

- Línea 165: indicamos que el tipo de vista es “calendar”.
- Línea 167: asignamos el nombre al calendario, nos marcará con un color distinto cada vez que se registre una reserva el campo “cliente\_id” que es el que tiene la relación establecida, en este caso el DNI del cliente, como fecha de inicio indicaremos que tome los datos del campo “fecha\_todo” (fecha y hora), se establece la vista month (mes) de las tres que tiene (week, day) pero si no podemos nada, por defecto muestra el mes. Los campos que mostrará serán el DNI (cliente\_id), “fecha\_alta” y “fecha\_todo”.

En el caso de tener una reserva dada de alta, al seleccionar la vista calendar nos mostrará:

W	lun	mar	mié	jue	vie	sáb	dom	
14		30		1	2	3	4	5
15		6	7	8	9	10	11	12
16		13	14	15	16	17	18	19
17		20	21	22	23	24	25	26
18		27	28	29	30	1	2	3

lun mar mié jue vie sáb dom  

1	2	3	4	5		
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30			



Ahora ya tenemos nuestro módulo funcionando con normalidad: damos de alta clientes, mesas y reservas.

### Implementación en la web.

El siguiente paso es implementar el módulo en el sitio web, para ello lo primero que haremos será diseñar la parte para el sitio web **que muestre ordenadamente los datos almacenados en las vistas que hemos creado.**

Dichas vistas web también se realizan mediante archivos XML importados también de la misma forma para las vistas genericas de Odoo (form, tree, etc.) pero con la particularidad del uso del tag (etiqueta) **<template>**. A través de su “id” único podemos hacer referencia a ellas.

Crearemos un nuevo archivo en “`../views/templates.xml`” y también en el manifest “`__openerp__.py`” lo indicaremos como dato propio del módulo. Éste contendrá:

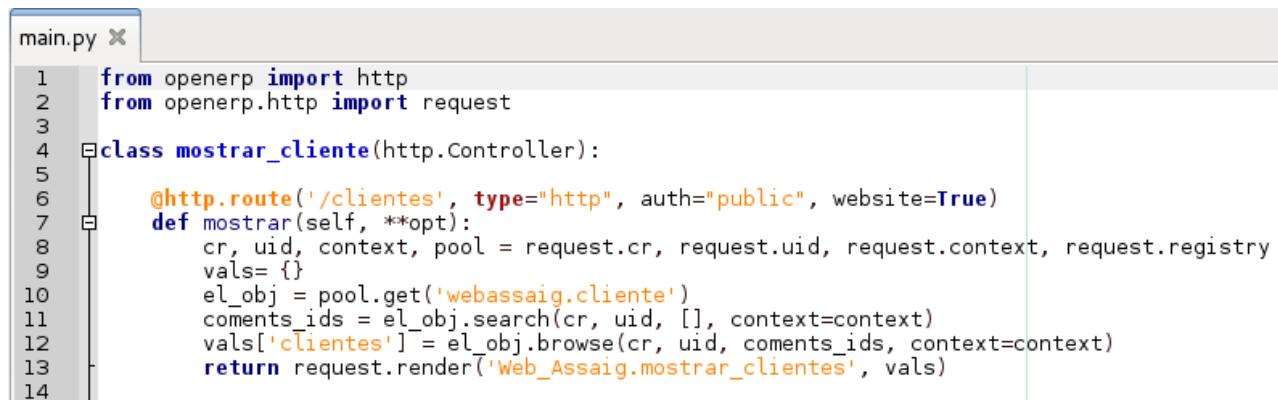
```
*templates.xml
1 <?xml version="1.0"?>
2 <openerp>
3   <data>
4     <!--MOSTRAR LOS CLIENTES-->>
5       <template id="mostrar_clientes" name="clientes">
6         <t t-call="website.layout">
7           <div id="wrap">
8             <h1 style="margin-left:15%;">Clientes</h1>
9             <div width="100%" align="center" style="margin-left:25%; margin-right:25%;">
10               <div width="100%" align="center">
11                 <h5><strong><button><a href="/page/Web_Assaig.nuevo_cliente">Click para añadir otro cliente</a></button></strong></h5>
12               </div>
13             <t t-foreach="clientes" t-as="cliente">
14               <h5 align="center"><strong><span width="100%" t-field="cliente.nombre"/></strong></h5>
15               <div id="apellidos" itemprop="apellidos" align="center" width="100%" t-field="cliente.apellidos"></div>
16               <div>Teléfono: <span width="100%" t-field="cliente.telefono"/></div>
17               <div>Observaciones: <span width="100%" t-field="cliente.observaciones"/></div>
18               <div>Fecha de alta: <span width="100%" t-field="cliente.fecha_alta"/></div>
19             </t>
20           </div>
21           <div width="100%" align="center">
22             <h5><strong><a href="/page/Web_Assaig.nuevo_cliente">Añadir un nuevo cliente.</a></strong></h5>
23           </div>
24         </div>
25       </template>
26     
```

*Podremos encontrar los datos de estos códigos xml en la base de datos, tabla “ir\_ui\_view”.*

- Línea 5: identificador del template para mostrar los clientes.
- Línea 6: `<t t-call="website.layout">`, es la “id” de la página web genérica creada en el módulo principal “website” el cual ya viene definido con cabecera, pie, menú, etc. Con ello al importarla no tendremos que replicar todo el código html de la página cada vez.
- Línea 13: `<t t-foreach="clientes" t-as="cliente">`, permite iterar sobre las variables en el objeto que pasaremos con los datos que queramos mostrar, clientes contendrá todos los campos y sus respectivos valores de la base de datos.

Por otra parte necesitaremos darle funcionalidad al template creado por lo que le pasaremos el objeto con las variables, además de especificar los directorios y páginas, para que sea posible haremos uso de las llamadas http mediante el uso de los 'controllers' del propio Odoo.

Crearemos en la carpeta “`../controllers/main.py`” el código correspondiente (recordemos también que ya lo hemos definido en el manifest del módulo e imports correspondientes):



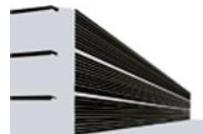
```

main.py ✘
1  from openerp import http
2  from openerp.http import request
3
4  class mostrar_cliente(http.Controller):
5
6      @http.route('/clientes', type="http", auth="public", website=True)
7      def mostrar(self, **opt):
8          cr, uid, context, pool = request.cr, request.uid, request.context, request.registry
9          vals= {}
10         el_obj = pool.get('webassaig.cliente')
11         coments_ids = el_obj.search(cr, uid, [], context=context)
12         vals['clientes'] = el_obj.browse(cr, uid, coments_ids, context=context)
13         return request.render('Web_Assaig.mostrar_clientes', vals)
14

```

- Líneas 1 – 2: importamos las librerías http (librería para las llamadas http) y “request” de la misma librería, la variable '**request**' contendrá la información de la petición http sobre el cliente.
- Línea 6: **@http.route**, indica que la siguiente función será la dirección de una llamada http.
- Línea 6: **type**, indica de qué tipo será (http o json).
- Línea 6: **auth**, es la autenticación para acceso a la petición. Valores: public, none, user...
- Línea 6: **website=True**, indicamos que la petición se realiza desde la website.
- Línea 7: **\*\*opt**, contiene las variables enviadas a la petición como por ejemplo, los datos de un formulario llenado en la web.

Con el siguiente código obtenemos los datos y redirigimos al usuario al template que hemos creado incluyendo estos. Pero aún no podremos acceder a la vista web que hemos creado ya que tendremos la petición escuchando en “dominio:puerto/clientes” por lo que necesitaremos crear un apartado en el menú que redirija a la petición. Para conseguirlo, a la hora de diseñar generaremos el dato en el momento de la instalación utilizando un archivo “**data.xml**”.



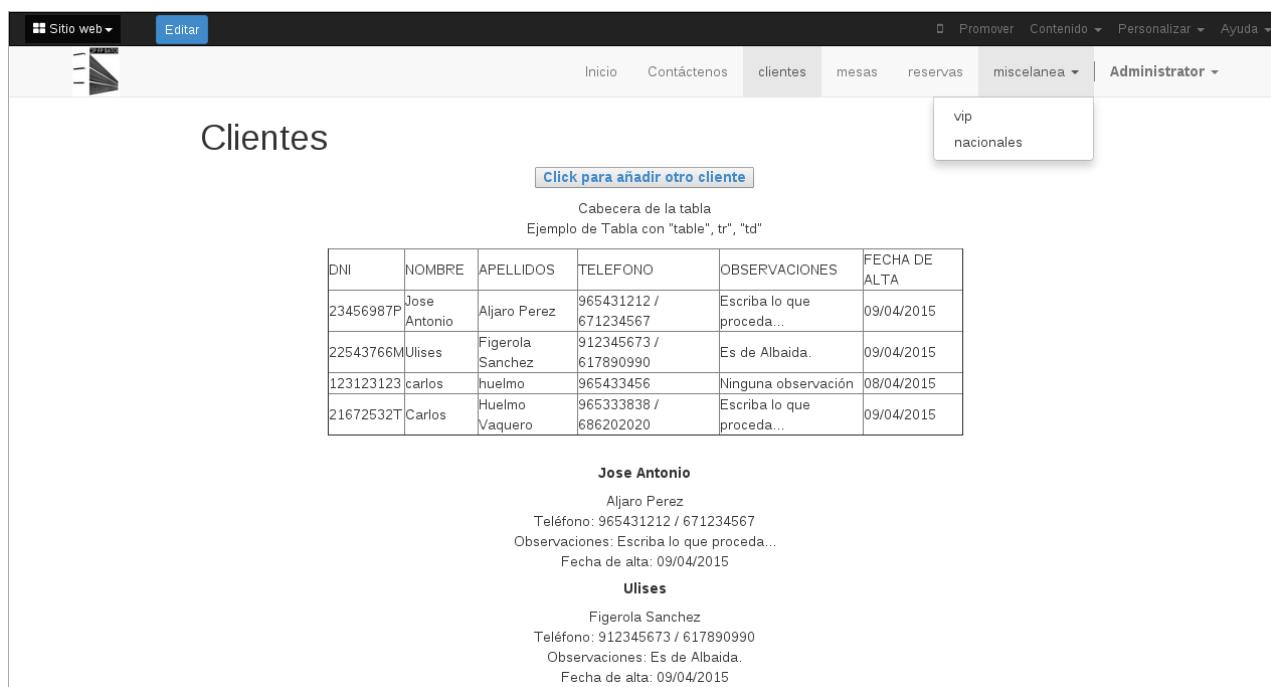
Crearemos en la carpeta “`../data/data.xml`” el código correspondiente (recordemos también que ya lo hemos definido en el manifest del módulo correspondiente):

```

data.xml
1  <?xml version="1.0"?>
2  <openerp>
3  <data noupdate="1">
4  <record id="menu_clientes" model="website.menu">
5   <field name="name">clientes</field>
6   <field name="url">/clientes</field>
7   <field name="parent_id" ref="website.main_menu"/>
8   <field name="sequence" type="int">80</field>
9  </record>
10 <record id="menu_mesas" model="website.menu">
11  <field name="name">mesas</field>
12  <field name="url">/mesas</field>
13  <field name="parent_id" ref="website.main_menu"/>
14  <field name="sequence" type="int">80</field>
15 </record>
16 <record id="menu_reservas" model="website.menu">
17  <field name="name">reservas</field>
18  <field name="url">/reservas</field>
19  <field name="parent_id" ref="website.main_menu"/>
20  <field name="sequence" type="int">80</field>
21 </record>
22 <record id="submenu_vip" model="website.menu">
23  <field name="name">vip</field>
24  <field name="url">/misclanea/vip</field>
25  <field name="parent_id" ref="menu_misclanea"/>
26  <field name="sequence" type="int">80</field>
27 </record>
28 <record id="submenu_nacionales" model="website.menu">
29  <field name="name">nacionales</field>
30  <field name="url">/misclanea/nacionales</field>
31  <field name="parent_id" ref="menu_misclanea"/>
32  <field name="sequence" type="int">80</field>
33 </record>
34 <record id="menu_misclanea" model="website.menu">
35  <field name="name">misclanea</field>
36  <field name="url">/misclanea</field>
37  <field name="parent_id" ref="website.main_menu"/>
38  <field name="sequence" type="int">80</field>
39 </record>
40 </data>
41 </openerp>

```

- Con lo cual obtendremos varias pestañas y una de ellas (misclanea) con dos submenús.
- No se permite más de un subnivel por defecto para el CMS.
- En el caso de que tenga submenús, la pestaña deja de comportarse como un botón.



The screenshot shows the Odoo website interface. At the top, there's a navigation bar with links for 'Sitio web', 'Editor', 'Promover', 'Contenido', 'Personalizar', 'Ayuda', and 'Administrator'. Below the navigation bar, there's a main menu with links for 'Inicio', 'Contáctenos', 'clientes', 'mesas', 'reservas', 'misclanea', and 'Administrator'. The 'misclanea' menu has two sub-items: 'vip' and 'nacionales'. In the center, there's a table titled 'Clientes' with columns: DNI, NOMBRE, APELLIDOS, TELEFONO, OBSERVACIONES, and FECHA DE ALTA. The table contains five rows of data. Below the table, there are two sections: 'Jose Antonio' and 'Ulises', each with their respective details: name, phone number, observations, and date of entry.

DNI	NOMBRE	APELLIDOS	TELEFONO	OBSERVACIONES	FECHA DE ALTA
23456987P	Jose Antonio	Aljaro Perez	965431212 / 671234567	Escriba lo que proceda...	09/04/2015
22543766M	Ulises Sanchez	Figerola Sanchez	912345673 / 617890990	Es de Albaida.	09/04/2015
123123123	carlos huelmo		965433456	Ninguna observación	08/04/2015
21672532T	Carlos Huelmo Vaquero		965333838 / 686202020	Escriba lo que proceda...	09/04/2015

**Jose Antonio**  
 Aljaro Perez  
 Teléfono: 965431212 / 671234567  
 Observaciones: Escriba lo que proceda...  
 Fecha de alta: 09/04/2015

**Ulises**  
 Figerola Sanchez  
 Teléfono: 912345673 / 617890990  
 Observaciones: Es de Albaida.  
 Fecha de alta: 09/04/2015

*Captura de pantalla del sitio web incluyendo nuestras pestañas añadidas y submenús.*

Crearemos una nueva vista para añadir nuevos clientes desde la página web. Dentro del archivo “..views/templates.xml”:

```

66 | <!--DAR DE ALTA MAS CLIENTES-->
67 □ <template id="nuevo_cliente" name="AddCliente" page="True">
68 |   <t t-call="website.layout">
69 □   <div id="wrap">
70 |     <h1 style="margin-left:15%">Añadir nuevo cliente</h1>
71 □     <form action="/clientes/nuevo_cliente" method="post" class="oe_login_form" style="margin-right:20%; margin-left:20%">
72 □       <div class="form-group">
73 |         <label for="name" class="control-label">DNI</label>
74 |         <input type="text" name="name" id="dni" class="form-control" required="required"/>
75 |
76 |         <label for="nombre" class="control-label">Nombre</label>
77 |         <input type="text" name="nombre" id="nombre" class="form-control" required="required"/>
78 |
79 |         <label for="apellidos" class="control-label">Apellidos</label>
80 |         <input type="text" name="apellidos" id="apellidos" class="form-control" required="required"/>
81 |
82 |         <label for="telefono" class="control-label">Teléfono</label>
83 |         <input type="text" name="telefono" id="telefono" class="form-control" required="required"/>
84 |
85 |         <label for="observaciones" class="control-label">Observaciones</label>
86 |         <textarea rows="5" name="observaciones" id="observaciones" class="form-control" placeholder="Escriba observaciones...." required="required"/>
87 |
88 </div>
89 □   <div class="clearfix oe_login_buttons">
90 |     <button type="submit" class="btn btn-primary">Guardar</button>
91 |     <button type="reset" class="btn btn-secondary">Limpiar</button>
92 |     <button type="button" class="btn btn-secondary"><a href="/clientes">Cancelar / Volver</a></button>
93 </div>
94 </form>
95 </div>
96 </t>
97 </template>
```

- Línea 67: **page="True"**, al añadirlo al template podremos hacer referencia a él mediante “/page/modulo\_nombre.id\_template” → “/page/Web\_Assaig.nuevo\_cliente”. Con lo cual podremos añadir un link fácilmente desde él a la página “clientes” que redirija a este nuevo template para añadir nuevos clientes.

Entonces añadiremos dos links al template de “mostrar\_clientes”:

Tendremos un botón al principio que redirige al template para añadir clientes:

```

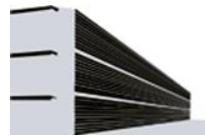
10 □   <div width="100%" align="center">
11 |     <strong><button><a href="/page/Web_Assaig.nuevo_cliente">Click para añadir otro cliente</a></button></strong></h5>
12 |   </div>
```

Después de mostrar los datos, hemos añadido un link que también hará lo mismo:

```

60 □   <div width="100%" align="center">
61 |     <h5><strong><a href="/page/Web_Assaig.nuevo_cliente">Añadir un nuevo cliente.</a></strong></h5>
62 |   </div>
```

- Líneas 90 – 92: añadimos tres botones con la siguiente funcionalidad, type=”submit” que guardará los datos introducidos, type=”reset” que borrará el contenido y el tercero que nos redirigirá a la página principal.



El siguiente paso será crear la petición http que recoja los datos y los guarde en la base de datos, por lo que añadiremos el siguiente código al archivo “`../controllers/main.py`” (las variables vienen en la variable `**post` en forma de http):

```

16 # PETICION RECOGE DATOS Y GUARDA EN LA BD
17 @http.route('/clientes/nuevo_cliente', type="http", auth="public", website=True)
18 def add_cli(self, **post):
19     cr, uid, context, pool = request.cr, request.uid, request.context, request.registry
20     pool.get('webassaig.cliente').create(cr, uid, post, context = context)
21     return request.redirect("/clientes")
22

```

En este momento podremos añadir nuevos clientes desde la página web y después nos los muestra:



Inicio Contáctenos clientes mesas reservas miscelanea ▾

## Clientes

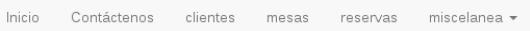
[Click para añadir otro cliente](#)

Cabecera de la tabla  
Ejemplo de Tabla con "table", "tr", "td"

DNI	NOMBRE	APELLIDOS	TELEFONO	OBSERVACIONES	FECHA DE ALTA
23456987P	Jose Antonio	Aljaro Perez	965431212 / 671234567	Escriba lo que proceda...	09/04/2015
22543766M	Ulises	Figerola Sanchez	912345673 / 617890990	Es de Albaida.	09/04/2015
123123123	carlos	huelmo	965433456	Ninguna observación	08/04/2015
21672532T	Carlos	Huelmo Vaquero	965333838 / 686202020	Escriba lo que proceda...	09/04/2015

Jose Antonio

Aljaro Perez  
Teléfono: 965431212 / 671234567  
Observaciones: Escriba lo que proceda...  
Fecha de alta: 09/04/2015



Inicio Contáctenos clientes mesas reservas miscelanea ▾

## Añadir nuevo cliente

DNI

Nombre

 Javier

! Completa este campo

Apellidos

 Alpiste Moreno

Teléfono

 965441212

Observaciones

 Escriba observaciones....

Guardar

Limpiar

Cancelar / Volver



**OJO:** Las restricciones en el modelo sólo tendrán efecto cuando se vayan a guardar los datos introducidos desde el website, mientras tendríamos que establecer también en el propio template “nuevo\_cliente” las mismas restricciones o similares para intentar controlar los errores cuando se produzcan. Por ejemplo si hay un DNI duplicado.

Al ser diseño web, tenemos que tener en cuenta que los navegadores no soportan ciertos atributos de las etiquetas utilizadas, por ejemplo, en firefox no está soportado que la etiqueta “<input> y su atributo type=”date” o ”datetime-local” mientras que en google chrome sí.

En el caso de que guardemos un cliente que contenga el mismo DNI mostrará el siguiente mensaje de error:



The server encountered an internal error and was unable to complete your request. Either the server is overloaded or there is an error in the application.

Desde terminal, el servidor mostrará los siguientes mensajes de error:

```
Archivo Editar Ver Buscar Terminal Ayuda
2015-04-10 07:14:17,667 31579 INFO prueba2 werkzeug: 127.0.0.1 - - [10/Apr/2015 07:14:17] "GET /page/Web_Assaig.nuevo_cliente HTTP/1.1" 200 -
2015-04-10 07:14:17,825 31579 INFO prueba2 werkzeug: 127.0.0.1 - - [10/Apr/2015 07:14:17] "POST /website/translations HTTP/1.1" 200 -
2015-04-10 07:14:17,860 31579 INFO prueba2 werkzeug: 127.0.0.1 - - [10/Apr/2015 07:14:17] "GET /favicon.ico HTTP/1.1" 404 -
2015-04-10 07:14:24,817 31579 ERROR prueba2 openerp.sql_db: bad query: INSERT INTO "webassaig_cliente" ("id", "name", "fecha_alta", "observaciones", "apellidos", "nombre", "telefono", "create_uid", "write_uid", "create_date", "write_date") VALUES(nextval('webassaig_cliente_id_seq'), '21672532T', '2015-04-10', 'rr', 'rr', 'rr', 'rr', 1, 1, (now() at time zone 'UTC'), (now() at time zone 'UTC')) RETURNING id
Traceback (most recent call last):
  File "/opt/odoo/openerp/sql_db.py", line 234, in execute
    res = self._obj.execute(query, params)
IntegrityError: llave duplicada viola restriccción de unicidad «webassaig_cliente_name_unique»
DETALLE: Ya existe la llave (name)=(21672532T).

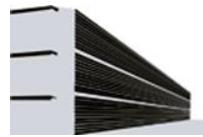
2015-04-10 07:14:24,821 31579 ERROR prueba2 openerp.sql_db: bad query: SELECT "ir_attachment".id FROM "ir_attachment" WHERE (( "ir_attachment"."type" = 'binary') AND (( "ir_attachment"."url" = '/clientes/nuevo_cliente')) ORDER BY "ir_attachment".id desc
Traceback (most recent call last):
  File "/opt/odoo/openerp/sql_db.py", line 234, in execute
    res = self._obj.execute(query, params)
InternalError: transacción abortada, las órdenes serán ignoradas hasta el fin de bloque de transacción

2015-04-10 07:14:24,822 31579 ERROR prueba2 openerp.addons.website.models.ir_http: 500 Internal Server Error:
```

Esto ocurre porque estamos añadiendo (INSERT) un nuevo cliente y hemos establecido que el DNI debe ser único en el campo “\_sql\_constraints” en el modelo / objeto “cliente”. ¿Cómo podemos controlar esto y para que no salte? Usando un “try...exception”.

Haríamos lo mismo para mostrar / añadir mesas y reservas.

(Seguiríamos por herencias en los templates, página 8 del documento de Mauro pero sería alargar el documento demasiado).



## ANEXO III.A – INSTALACION DE OODOO 8 EN MS WINDOWS 7.

Descargado el paquete de instalación desde la web oficial “**odoo\_8.0.latest.exe**”, al ejecutarlo “como administrador” nos aparecerá el asistente de turno. Lo primero que indicaremos será el idioma inglés puesto que sólo está disponible en inglés y francés. Acto seguido nos aparecerá la pantalla de bienvenida, pulsaremos el botón “next”. Aceptaremos los términos y condiciones.

Podremos seleccionar entre tres tipos de instalación:

- All In One (todo en uno).
- Server Only (solamente Odoo).
- Custom (personalizada)

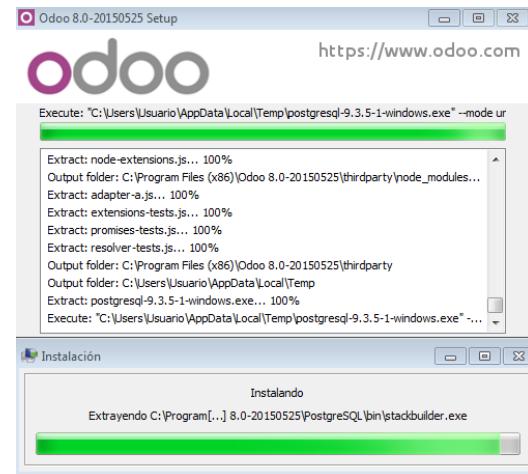
Necesitaremos 500MB de espacio en disco duro para la instalación puesto que no tenemos Postgres instalado en nuestro equipo.

En la siguiente ventana configuraremos la conexión con la base de datos de Postgres, para ello indicaremos:

- Hostname: **localhost**.
- Puerto: **5432** (por defecto).
- Usuario de la BD: **openpg**.
- Contraseña: **openpgpwd**.

Después indicaremos la ruta de la instalación de Odoo y pulsaremos el botón “install”.

Finalizada la instalación del servidor Odoo, el asistente procederá a la extracción y posterior instalación de PostgreSQL (en el caso de no tenerlo instalado, el asistente lo autodetecta).



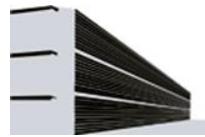


Al finalizar, el asistente nos indica que la instalación se ha completado, podremos iniciar Odoo o finalizar.

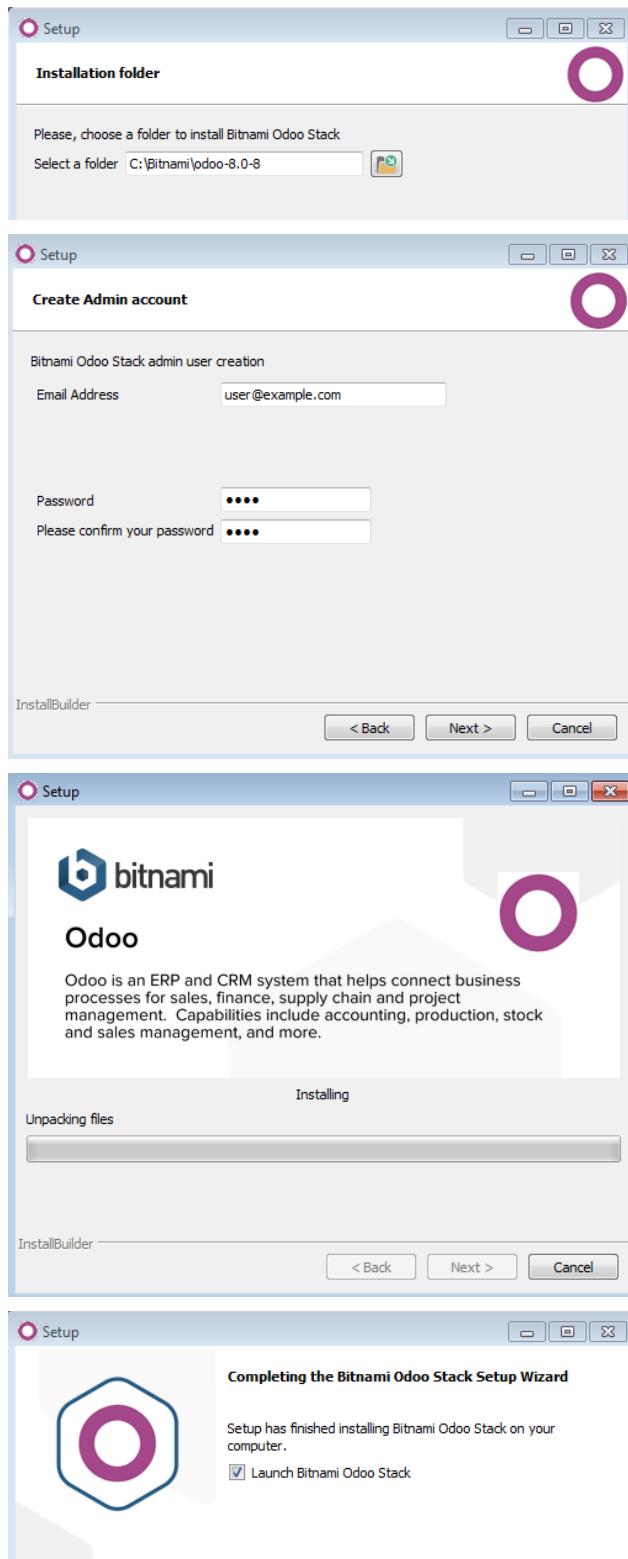
Se abrirá nuestro navegador con la ruta “localhost:8069”, pero dará error:



Possiblemente se deba a que no tenemos instalado “Python 2.7.x” en nuestro equipo por lo que hemos preferido desinstalar Odoo, instalar previamente python x64 bits. **Pero sigue persistiendo el problema con PostgreSQL, indica que el error es que no está escuchando pero revisando la configuración, está todo correcto...frustrante.**



Otra posible solución es utilizar el paquete de instalación des “**Bitnami**” desde su página web “<https://bitnami.com/stack/odoo/installer>” y seguir los pasos del asistente de turno:



Después de la ventana de bienvenida indicaremos la ruta donde se instalará Odoo.

Indicaremos la cuenta de correo y la contraseña “1234”.

Después dejaremos sin configurar:

- SMTP settings porque no queremos el soporte de configuración.
- No aprender más sobre el “Bitnami Cloud Hosting”.

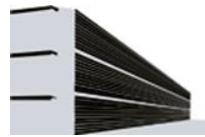
Después comenzará la instalación.

Aproximadamente dura unos 30 minutos.

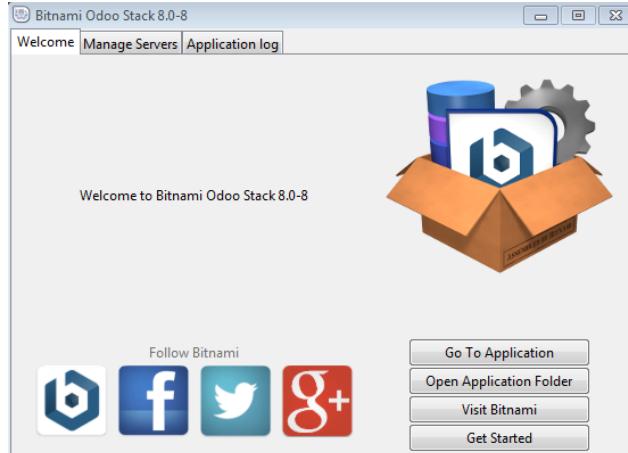
Se instalarán:

- Dependencias de OpenErp.
- Apache http server.
- Dependencias de Python.
- Servicio OpenErp.
- Creación schema de la BD.

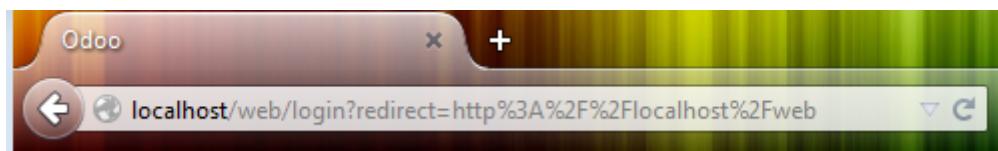
Finalizada la instalación podremos lanzar el “Bitnami Odoo Stack”



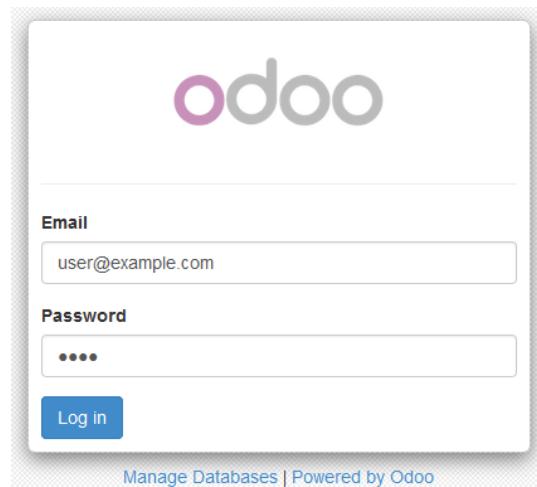
Nos aparecerá una nueva ventana en nuestro escritorio, en la pestaña de bienvenida podremos acceder a la aplicación (es del mismo tipo cuando instalamos WAMP)



Al pulsar sobre “Go To Application” abrirá nuestro navegador establecido por defecto y la ruta preestablecida.

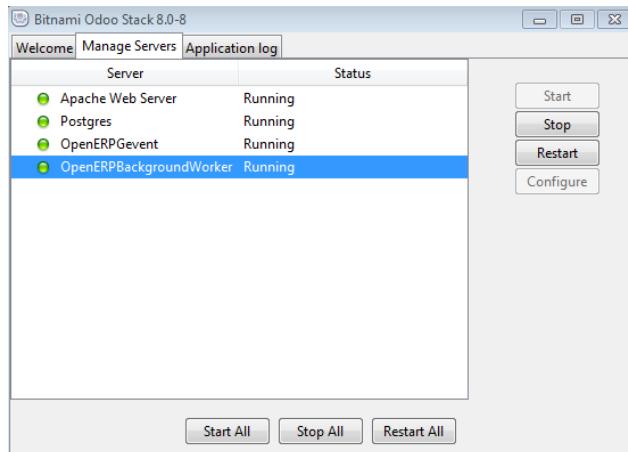


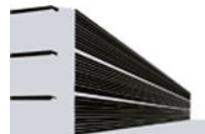
En donde pondremos nuestro “email” (usuario) y la contraseña (“1234”) para logearnos. También podremos administrar nuestras bases de datos como en anteriores versiones de Open.



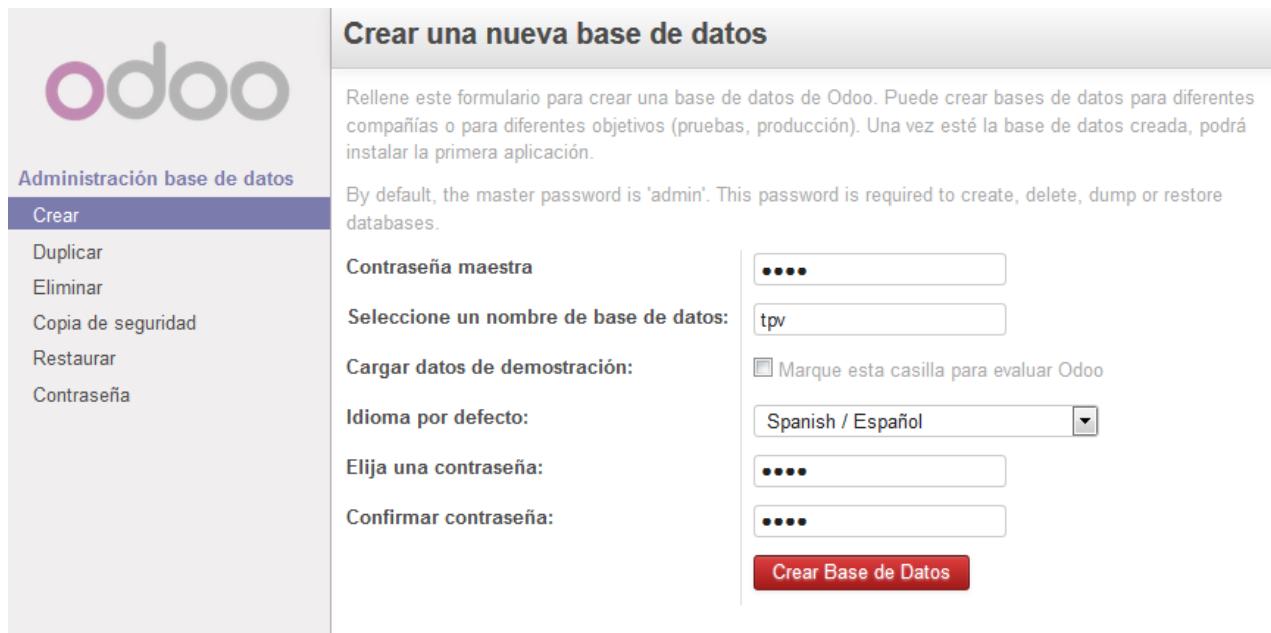
En la pestaña “Manage Servers” podremos ver los servidores que están funcionando:

- Apache.
- Postgres.
- OpenERP gevent.
- OpenERP background worker.





Desde la ventana de login del navegador indicaremos “Manage Databases” para crear una nueva base de datos, para ello indicaremos el nombre, idioma y la contraseña. Pulsaremos el botón “Crear Base de Datos”:

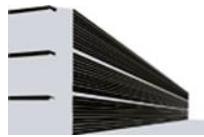


The screenshot shows the Odoo administration interface for creating a new database. On the left, there's a sidebar with options: 'Crear' (Create), 'Duplicar' (Duplicate), 'Eliminar' (Delete), 'Copia de seguridad' (Backup), 'Restaurar' (Restore), and 'Contraseña' (Password). The main area is titled 'Crear una nueva base de datos' (Create a new database). It contains the following fields:

- Contraseña maestra:** A password field containing '\*\*\*\*\*'.
- Seleccione un nombre de base de datos:** A text input field containing 'tpv'.
- Cargar datos de demostración:** A checkbox labeled 'Marque esta casilla para evaluar Odoo' (Check this box to evaluate Odoo).
- Idioma por defecto:** A dropdown menu set to 'Spanish / Español'.
- Elija una contraseña:** A password field containing '\*\*\*\*\*'.
- Confirmar contraseña:** A password field containing '\*\*\*\*\*'.
- Crear Base de Datos:** A red button at the bottom right.

El resto es igual como hacíamos en anteriores versiones de OpenERP, el problema radica en que **la instalación puede llevar 30 minutos como mínimo.**

**Pero tenemos otro problema, no podremos crear una nueva base de datos por problemas de permisos, la solución que proponen en los foros que se han consultado es “duplicar” la base de datos existente. Pero no tenemos permisos para hacer nada, así que trabajaríamos sobre la base de datos creada “bitnami\_openerp” para instalar los módulos, etc.**



## ANEXO III.B – INSTALACION DE OODOO 8 EN GNU/LINUX DEBIAN 7.

Mediante el uso de una distribución linux como es la DEBIAN 7.8 Wheezy (mediante VirtualBox) procederemos a realizar los pasos que nos indica la web oficial de odo que se pueden consultar en este link "<https://www.odoo.com/documentation/8.0/setup/install.html>" para la instalación del Odoo a partir de los repositorios DEB.

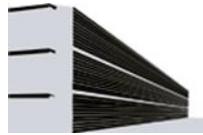
Ejecutaremos los siguientes comandos como root:

```
# wget -O - https://nightly.odoo.com/odoo.key | apt-key add -
# echo "deb http://nightly.odoo.com/8.0/nightly/deb/ ./" >> /etc/apt/sources.list
# apt-get update && apt-get install odoo
```

Con lo que automáticamente se instalarán todas las dependencias, se instalará Odoo y se ejecutará como daemon de forma automática. **Es posible que de “errores al procesar” con postgresql-9.1 pero el funcionamiento observado es normal. Hemos probado de instalar PostgreSQL antes desde los repositorios mediante el comando: “ # apt-get install postgres\* ” y ya no aparecen dichos errores de procesamiento.**

El archivo de configuración se encuentra en “**/etc/odoo/openerp-servconf.conf**” como queremos añadir la ruta de nuestros propios “addons” en addons\_path añadiríamos con una coma la ruta absoluta.

También se nos indica que para poder imprimir informes PDF es necesario instalar “wkhtmltopdf” pero no de los repositorios debian (que no se soporta por Odoo) sino la versión 0.12.1 disponible en la página web “<http://wkhtmltopdf.org/>”.



## ANEXO III.C – INSTALACION DE ODOO 8 EN GNU/LINUX XUBUNTU.

Desde la página oficial de odoo “” nos descargaremos la última versión para nuestro sistema operativo, una distribución Xubuntu 14.04.02-LTS.

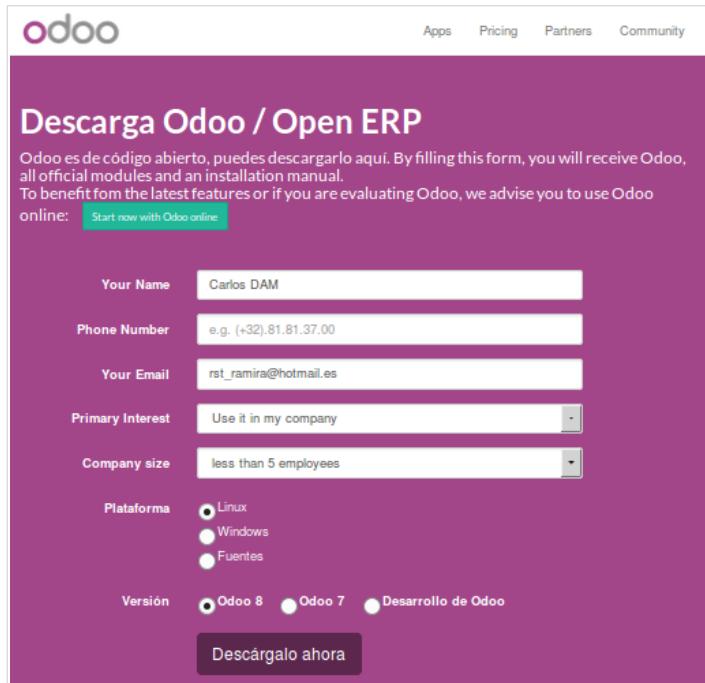
Nos podremos descargar los paquetes de instalación con extensión DEB y RPM, en nuestro caso utilizaremos el primer paquete.

**Primero,** instalaremos “PostgreSQL” mediante el siguiente comando:

```
> sudo apt-get install postgresql
```

Finalizada la instalación queda el servidor arrancado y funcionando (9.3). Además instalaremos el gestor gráfico para poder realizar la administración de la base de datos “pgadmin3”:

```
> sudo apt-get install pgadmin3
```



**Segundo,** crearemos el usuario openerp dentro del sistema:

```
chv@xubu:~$ sudo adduser --system --home=/home/openerp --group openerp
Añadiendo el usuario del sistema `openerp` (UID 117) ...
Añadiendo un nuevo grupo `openerp` (GID 126) ...
Añadiendo un nuevo usuario `openerp` (UID 117) con grupo `openerp` ...
Creando el directorio personal `/home/openerp' ...
```

Después, al usuario “postgres” le cambiaremos la contraseña por la “12345” mediante el siguiente comando: `> sudo passwd postgres`

**Tercero,** configuraremos un usuario en “PostgreSQL” para Odoo y así éste se podrá conectar a la base de datos. Nos loguearemos como usuario “postgres”: `> sudo su - postgres`

Crearemos y configuraremos el usuario “openerp” en PostgreSQL (contraseña “12345”):

```
$ createuser --createdb --username postgres --no-createrole --no-superuser
--pwprompt openerp
$ exit
logout
```



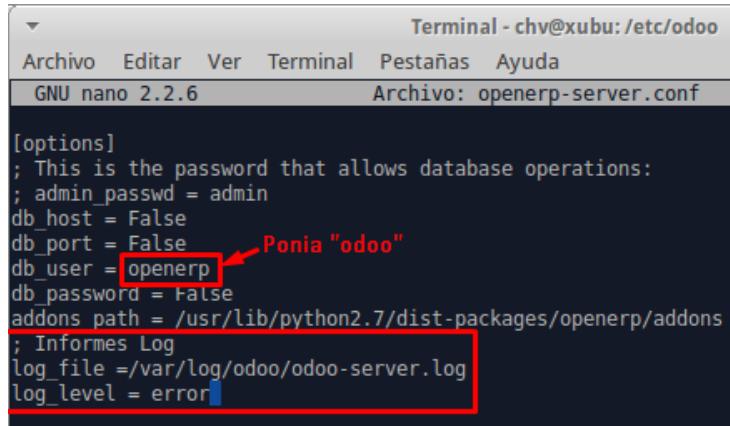
Además, nos descargaremos las librerías “Python” necesarias:

```
sudo apt-get install python-dateutil python-docutils python-feedparser python-gdata python-jinja2 python-ldap python-libxml2 python-mako python-mock python-openid python-psycopg2 python-psutil python-pybabel python-pychart python-pydot python-pyparsing python-reportlab python-simplejson python-tz python-unittest2 python-vatnumber python-vobject python-webdav python-werkzeug python-xlwt python-yaml python-zsi
```

**Cuarto**, finalizada la descarga e instalación de los paquetes desde los repositorios, procederemos a instalar “Odoo v8”. Instalaremos el paquete DEB que nos hemos descargado desde la página oficial anteriormente: > sudo dpkg -i odo\_8.0.latest\_all.deb

Es posible que tengamos problemas de dependencias, para corregirlo tendremos que ejecutar el siguiente comando: > sudo apt-get install -f

**Quinto**, accederemos al archivo de configuración que se encuentra en **“/etc/odoo/openerp-server.conf”** y realizaremos los siguientes cambios:

- **db\_user** = openerp.
  - **log\_file**, ruta y archivo dónde se guardarán todos los mensajes de error que se produzcan.
  - **log\_level**, indicamos de tipo error.
- 

Guardaremos los cambios.

Reiniciaremos Odoo con el comando > sudo service odoo [restart|start|stop].

Desde nuestro navegador escribimos la siguiente URL “localhost:8069” y ya podremos crear las bases de datos que queramos. Cuando finalice el proceso y cargue los datos, se nos mostrará la ventana con los módulos que trae por defecto Odoo.

**Problema: hemos reiniciado nuestra máquina virtual y al querer acceder a nuestra empresa desde el navegador, éste nos ha mostrado el error mostrado en la imagen.** En principio podríamos reiniciar el servidor Odoo, después comprobar si el propietario y el grupo del directorio “/var/run/postgresql” es “postgres”, en caso negativo ejecutaríamos: > sudo chown -R postgres:postgres /var/run/postgresql





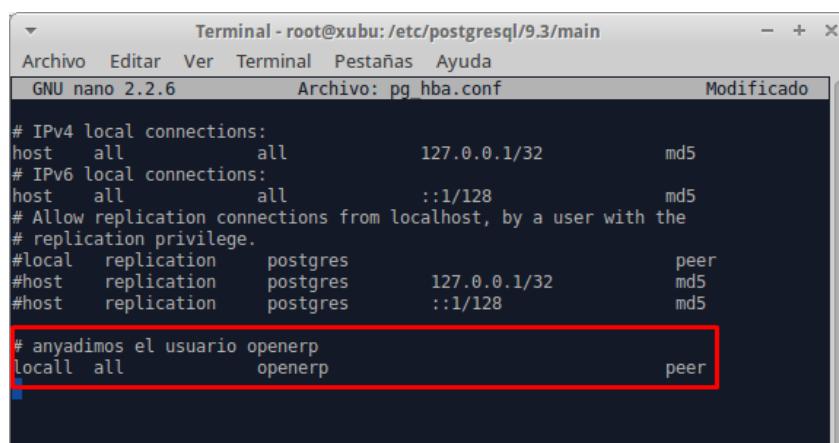
Además, si vamos al archivo de log en “`/var/log/odoo/odoo-server.log`” tendremos como pista que el usuario “openerp” no tiene permitido la autenticación en la base de datos:

```

return fun(self, *args, **kwargs)
File "/usr/lib/python2.7/dist-packages/openerp/sql_db.py", line 522, in borrow
    result = psycopg2.connect(dsn=dsn, connection_factory=PsycoConnection)
File "/usr/lib/python2.7/dist-packages psycopg2/_init_.py", line 179, in connect
    connection_factory=connection_factory, async=async)
OperationalError: FATAL: Peer authentication failed for user "openerp"
root@xubu:/var/log/odoo#

```

Tendremos que editar el archivo “`pg_hba.conf`” que se encuentra en “`/etc/postgresql/9.3/main`” y añadiremos al final del archivo:



```

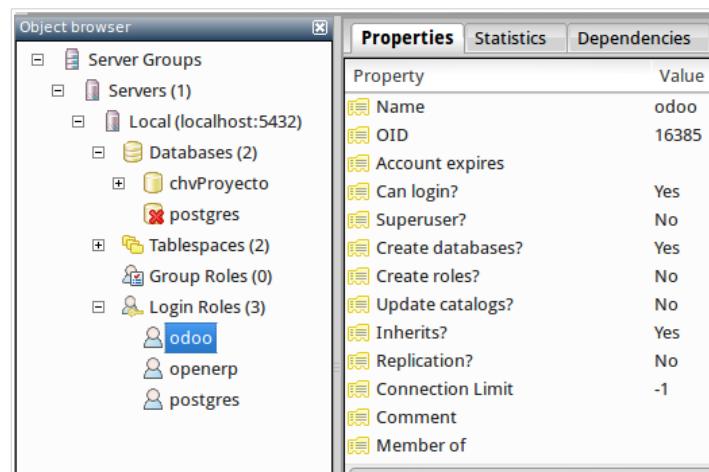
Terminal - root@xubu:/etc/postgresql/9.3/main
Archivo Editar Ver Terminal Pestañas Ayuda
GNU nano 2.2.6 Archivo: pg_hba.conf Modificado
# IPv4 local connections:
host all all 127.0.0.1/32 md5
# IPv6 local connections:
host all all ::1/128 md5
# Allow replication connections from localhost, by a user with the
# replication privilege.
#local replication postgres
#host replication postgres 127.0.0.1/32 md5
#host replication postgres ::1/128 md5
# anyadimos el usuario openerp
local all openerp peer

```

Detendremos y reiniciaremos el servidor “PostgreSQL”: `> sudo service postgresql stop`

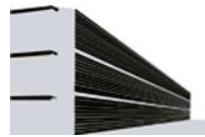
Pero esto tampoco resolverá el problema.

**SOLUCIÓN:** el problema es que con la instalación de Odoo, se ha creado el usuario “odoo” según nos aparece con “`pgadmin3`”. Por lo tanto, tendremos que modificar el archivo de configuración “`/etc/odoo/openerp-server.conf`” y en “`db_user`” poner “`odoo`”. Reiniciaremos los servidores “`postgresql`” y “`odoo`”.



Property	Value
Name	odoo
OID	16385
Account expires	
Can login?	Yes
Superuser?	No
Create databases?	Yes
Create roles?	No
Update catalogs?	No
Inherits?	Yes
Replication?	No
Connection Limit	-1
Comment	
Member of	

Por tanto, no necesitaremos ni crear el usuario “`openerp`” ni su “`home`” ni tampoco cambiar el “`db_user`” del archivo de configuración de “`openerp-server.conf`”.



**Finalmente**, ya configurada nuestra empresa, con los módulos que queramos, permisos de usuario, configuración de la zona horaria, etc., procederemos a descargar e instalar el resto de librerías necesarias:

```
> sudo apt-get install python-psycopg2 python-reportlab python-egenix-mxdatetime  
> sudo apt-get install python-tz python-pychart python-pydot python-lxml python-vobject python-yaml python-hippocanvas  
> sudo apt-get install python-openssl python-matplotlib python-dev build-essential python-setuptools python-mako python-django python-imaging python-beaker  
> sudo apt-get install python-openid python-pybabel
```



## CONCLUSIONES FINALES.

Las FCT's siempre ayudan a conocer el interior de una empresa, establecer lazos con otras personas desconocidas hasta ese momento, sentirse como uno más y cómo podría ser nuestro trabajo dentro de ésta pero no siempre es de color de rosa, es posible que no nos den nada que hacer o que nos dejen a nuestra suerte perdidos y frustrados sin poder hacer nada ni colaborar, pero a pesar de esto siempre debemos seguir hacia adelante hasta la finalización de las prácticas, tanto si hemos aprendido algo como si no.

Personalmente, he estado bien en la empresa, el trato ha sido inmejorable nos hemos integrado con el resto de trabajadores pero lo que hecho en falta es que no me dieran nada que hacer pero gracias a eso he llegado a conocer más profundamente cómo funciona Odoo 8. También es cierto que me han ayudado a resolver las dudas que han ido surgiendo a lo largo del desarrollo del presente módulo. Pero no he participado en ningún curso de formación en la misma empresa cuando ha habido otros alumnos en prácticas que sí han ido, en fin, esto es lo que hay, pues con resignación debo acatar.

Por último, es cierto que he recibido toda la atención posible por parte del instructor de la empresa (dos días si sumamos casi los tres meses de prácticas) puesto que tienen trabajo pero qué menos que mostrar mayor interés por el trabajo que está realizando el alumno e incluso intentar aportar algo más que el alumno no se percata respecto al proyecto.

Alcoy, a 10 de junio de 2015.