

## PROYECTO DEL SEGUNDO CUATRIMESTRE



**UNIVERSIDAD DE CASTILLA-LA MANCHA**  
**ESCUELA SUPERIOR DE INFORMÁTICA**

**INGENIERÍA**  
**EN INFORMÁTICA**

**PROYECTO FIN DE CARRERA**

Proyecto del Segundo Cuatrimestre

Buggin' & Debuggin':  
Cristian Del Cerro Gómez  
Javier García Simón  
Raúl Reguillo Carmona  
Laura Del Río Avilés

**Mayo, 2013**



**UNIVERSIDAD DE CASTILLA-LA MANCHA**  
**ESCUELA SUPERIOR DE INFORMÁTICA**  
Departamento de Tecnologías y Sistemas de Información

**PROYECTO FIN DE CARRERA**

Proyecto del Segundo Cuatrimestre

Autor: Buggin' & Debuggin':

Cristian Del Cerro Gómez

Javier García Simón

Raúl Reguillo Carmona

Laura Del Río Avilés

Director: Dr. Ismael Caballero

**Mayo, 2013**

**Buggin' & Debuggin':**  
**Cristian Del Cerro Gómez**  
**Javier García Simón**  
**Raúl Reguillo Carmona**  
**Laura Del Río Avilés**

Ciudad Real – Spain

*E-mail:* `buggin.debuggin@uclm.es`

*Teléfono:* 123 456 789

*Web site:* `http://esi.uclm.es`

© 2013 Buggin' & Debuggin':  
Cristian Del Cerro Gómez  
Javier García Simón  
Raúl Reguillo Carmona  
Laura Del Río Avilés

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Se permite la copia, distribución y/o modificación de este documento bajo los términos de la Licencia de Documentación Libre GNU, versión 1.3 o cualquier versión posterior publicada por la *Free Software Foundation*; sin secciones invariantes. Una copia de esta licencia esta incluida en el apéndice titulado «GNU Free Documentation License».

Muchos de los nombres usados por las compañías para diferenciar sus productos y servicios son reclamados como marcas registradas. Allí donde estos nombres aparezcan en este documento, y cuando el autor haya sido informado de esas marcas registradas, los nombres estarán escritos en mayúsculas o como nombres propios.

**TRIBUNAL:**

**Presidente:**

**Vocal 1:**

**Vocal 2:**

**Secretario:**

**FECHA DE DEFENSA:**

**CALIFICACIÓN:**

**PRESIDENTE**

**VOCAL 1**

**VOCAL 2**

**SECRETARIO**

Fdo.:

Fdo.:

Fdo.:

Fdo.:

# Resumen

El presente documento detalla una visión global del sector ERP así como de sus términos de uso, características y ventajas. Inicialmente se describirá qué es un ERP así como sus principales atributos, para después particularizar en un caso concreto de funcionamiento y desarrollo.

Para ilustrar el funcionamiento y filosofía de un ERP se ha escogido como tema central el desarrollo en **OpenERP**, al ser éste un ERP libre, de fácil acceso y bien documentado, evitando así licencias incómodas que no permitiesen una instalación ni el desarrollo de un componente funcional.

Se han elaborado a su vez una serie de manuales prácticos que, seguidos paso a paso, permitan llegar a realizar una aplicación funcional partiendo desde cero.

# Índice general

<b>Resumen</b>	<b>VI</b>
<b>Índice general</b>	<b>VII</b>
<b>Índice de cuadros</b>	<b>X</b>
<b>Índice de figuras</b>	<b>XI</b>
<b>Índice de listados</b>	<b>XII</b>
<b>Listado de acrónimos</b>	<b>XIII</b>
<b>Agradecimientos</b>	<b>XIV</b>
<b>1. Introducción</b>	<b>1</b>
1.1. Características . . . . .	1
1.2. Ventajas . . . . .	2
1.3. Desventajas . . . . .	3
1.4. Elección práctica . . . . .	4
1.5. Estructura del documento . . . . .	4
<b>2. Objetivos</b>	<b>5</b>
2.1. Objetivos generales . . . . .	5
2.1.1. Objetivos específicos . . . . .	5
<b>3. Estado del Arte</b>	<b>7</b>
3.1. SAP ERP . . . . .	7
3.1.1. Desarrollo. ABAP. . . . .	7
3.1.2. Sistemas compatibles . . . . .	7
3.1.3. Licencias SAP . . . . .	8
3.2. ERPs de código abierto actuales . . . . .	9

3.3. OpenERP . . . . .	10
3.3.1. Introducción . . . . .	10
3.3.2. Desarrollo . . . . .	10
3.3.3. OpenERP como servicio . . . . .	10
3.3.4. Licencias . . . . .	11
3.3.5. Compatibilidad . . . . .	11
<b>4. Metodologías</b>	<b>12</b>
4.1. Scrum . . . . .	12
4.2. Roles . . . . .	12
4.3. Product Backlog y Sprints . . . . .	13
4.4. Reuniones . . . . .	13
<b>5. Marco tecnológico</b>	<b>19</b>
5.1. Arquitectura . . . . .	19
5.1.1. Base de datos PostgreSQL . . . . .	20
5.1.2. Servidor OpenERP . . . . .	20
5.1.3. Servidor ORM . . . . .	21
5.1.4. Servidor web . . . . .	22
5.1.5. Módulos . . . . .	22
5.1.6. Cientes . . . . .	22
<b>6. Diseño de la herramienta</b>	<b>24</b>
6.1. Estructura de un módulo . . . . .	24
6.1.1. Python import file <code>__init__.py</code> . . . . .	25
6.1.2. Mafinfiesto <code>__openerp__.py</code> . . . . .	25
6.1.3. Objetos . . . . .	25
6.1.4. Ficheros XML . . . . .	26
6.1.5. Vistas . . . . .	27
6.1.6. Desarrollar nuevos objetos . . . . .	27
6.2. Una aplicación real: <i>Bugginv5</i> . . . . .	28
6.3. Una aplicación real: <i>ClientesBugginv2</i> . . . . .	30
6.4. Patrones de diseño . . . . .	30
<b>7. Pruebas</b>	<b>33</b>
7.1. Framework de pruebas . . . . .	33



<b>8. Conclusiones</b>	<b>34</b>
<b>A. Manual de instalación</b>	<b>38</b>
A.1. Instalación en Ubuntu Linux . . . . .	38
A.1.1. Preparando el servidor . . . . .	38
A.1.2. Crear el usuario OpenERP . . . . .	38
A.1.3. Instalar y configurar la base de datos PostgreSQL . . . . .	39
A.1.4. Instalar las bibliotecas necesarias de Python . . . . .	39
A.1.5. Instalar el servidor OpenERP . . . . .	40
A.1.6. Configurar la aplicación OpenERP . . . . .	40
A.1.7. Instalar el script de arranque . . . . .	41
A.1.8. Probar el servidor . . . . .	41
A.1.9. Automatizar el arranque y parada de OpenERP . . . . .	42
A.2. Instalación en Windows . . . . .	42
<b>B. Código fuente de la herramienta <i>Bugginv5</i></b>	<b>43</b>
<b>C. Código fuente de la herramienta <i>ClientesBugginv2</i></b>	<b>46</b>
<b>D. Definición de la empresa</b>	<b>50</b>
D.1. Componente organizacional . . . . .	50
D.1.1. Organigrama . . . . .	50
D.1.2. Acerca de la empresa Buggin' & Debuggin' . . . . .	51
D.1.3. Plan de inversión . . . . .	52
D.1.4. Estrategia de marketing y publicidad . . . . .	53
D.2. Proyectos OpenERP . . . . .	53
<b>Bibliografía</b>	<b>54</b>

## Índice de cuadros

3.1. Tecnologías ERP abiertas . . . . .	9
4.1. Reuniones llevadas a cabo . . . . .	16
6.1. Propiedades . . . . .	32

## Índice de figuras

4.1. Tabla de tareas . . . . .	14
4.2. Calendario de tareas . . . . .	15
4.3. Diagrama de Gantt (I) . . . . .	16
4.4. Diagrama de Gantt (y II) . . . . .	17
4.5. Informe de ejemplo . . . . .	18
5.1. Arquitectura global . . . . .	20
5.2. Model-View-Controller . . . . .	21
5.3. Relaciones MVC . . . . .	22
6.1. Estructura de un módulo . . . . .	25
6.2. Workflow . . . . .	28
6.3. Relación de ventas por mes . . . . .	29
6.4. Agrupación de ventas por mes en un gráfico . . . . .	29
6.5. Formulario del módulo . . . . .	29
6.6. Relación de clientes . . . . .	30
6.7. Relación de ventas . . . . .	31
6.8. Gráfico de ventas por cliente . . . . .	31
8.1. Costo presupuestado . . . . .	36
A.1. Login de usuario . . . . .	42
D.1. Organigrama de la empresa . . . . .	50

## Índice de listados

6.1. Ejemplo de vista en XML . . . . .	26
B.1. Archivo <code>__init__.py</code> . . . . .	43
B.2. Archivo <code>__openerp__.py</code> . . . . .	43
B.3. Archivo <code>bugginv5.py</code> . . . . .	44
B.4. Vista de <code>bugginv5</code> . . . . .	45
C.1. Archivo <code>__init__.py</code> . . . . .	46
C.2. Archivo <code>__openerp__.py</code> . . . . .	46
C.3. Archivo <code>clientesbugginv2.py</code> . . . . .	47
C.4. Vista de <code>clientesbuggin</code> . . . . .	48
C.5. Vista de <code>clientesbuggin</code> (continuación) . . . . .	49

## Listado de acrónimos

# Agradecimientos

A Ismael, por su dedicación, su devoción y su sempiterno buen humor hasta cuando le damos motivos para no hacer gala de él.

A Macario, por su imprescindible ayuda con el manejo de la herramienta en cuestión y recibirnos siempre tan amablemente.

Buggin' & Debuggin'

*Esta va por nosotros*

## Capítulo 1

# Introducción

LOS sistemas de planificación de recursos empresariales o «ERP» son sistemas de información que integran un conjunto de herramientas que facilitan a las empresas la gestión del negocio. Todas las herramientas están orientadas a operaciones de producción de bienes o servicios.

En un mundo donde la tecnología está cobrando cada vez más importancia y cambia continuamente, la filosofía y necesidad de las ERP se erige desde hace bastantes años, adaptándose a los tiempos pero manteniendo sus bases firmes:

- Optimización de los procesos empresariales
- Acceso a la información
- Posibilidad de compartir información entre todos los componentes de la organización
- Eliminación de datos y operaciones innecesarias de reingeniería

A la hora de usar una herramienta de ERP, encontramos que ofrece soluciones a los problemas clásicos que surgen cuando se lleva un negocio, independientemente de su magnitud. Se utiliza para resolución de problemas contables, mercantiles o fiscales permitiendo un mejor control sobre estos ámbitos.

Así pues, la meta de un ERP es apoyar a los clientes del negocio y manejar eficientemente la información relativa a una organización. Lo que en general se podría decir de cualquier software orientado al apoyo de una tarea.

## 1.1. Características

No obstante, existen ciertas características que distinguen a un ERP de otro software empresarial:

- **Modular:** La funcionalidad en un ERP se encuentra dividida en módulos, los cuales pueden ser instalados en función a las necesidades del cliente. Se adapta pues a la filosofía de la empresa, entendida como un conjunto de departamentos que se encuentran interrelacionados, comparten información y los procesos involucran a todas las partes.



- **Configurable:** Un ERP debe poder configurarse conforme a las necesidades del cliente mediante el desarrollo de código software. Es posible que las empresas necesiten particularizar ciertas tareas que ya ofrece por defecto el ERP en cuestión y para ello sea necesario adaptar cierta funcionalidad. Algunos ERP más avanzados pueden incorporar herramientas de programación de cuarta generación para un desarrollo rápido de nuevos módulos.
- **Base de datos centralizada:** Como elemento central en un negocio, se tiene la base de datos global. Es necesario pues un sistema robusto que soporte la cantidad de información que puede generar una organización (más grande o más pequeña) al cabo del tiempo y que ofrezca mecanismos para asegurar los datos, interacción consistente y por supuesto, sea eficiente.
- **Interacción entre componentes:** Al igual que ocurre en una empresa, en donde las distintas secciones deben interactuar e intercambiar información, en un ERP los módulos deben poder comunicarse para consolidar operaciones.
- **Captura de datos:** Los flujos de información que deban acabar en la base de datos deben ser compuestos por datos consistentes y completos, teniendo en cuenta que toda la organización hará uso de ellos en algún momento.
- **Reingeniería de procesos:** Generalmente una empresa que implante el ERP puede tener que modificar alguno de sus procesos para que esté alineado con el sistema, lo que se conoce como Reingeniería de Procesos.
- **ERP vs Suite:** El discriminante más notable entre un ERP y software de gestión se encuentra en la propia definición. Un ERP se entiende como la aplicación que integra en un único sistema todos los procesos de negocio de una empresa, con el fin de que los datos estén disponibles todo el tiempo, para todo el mundo de una manera centralizada. Mientras, el software de gestión, puede verse como un conjunto de programas basados en múltiples aplicaciones (o *suites*) independientes o modulares, cuya base de datos no es única.

## 1.2. Ventajas

A la hora de abordar el manejo de datos por parte de una empresa, se pueden encontrar muchas aplicaciones cerradas que no son personalizables y, por tanto, ofrecen menos garantías de ser óptimas para el negocio. Es bien sabido que para llevar a cabo un trabajo es mejor contar con las mejores herramientas y éstas, en definitiva, son las que mejor se adapten a nosotros. Por lo tanto un ERP, al ser configurable, ofrece al cliente la posibilidad de adaptarlo a sus necesidades, lo que de seguro repercute en la productividad.

La integración de los componentes también es una ventaja remarcable. La curva de aprendizaje por parte del usuario siempre será menor si todos los productos (módulos en este caso)

siguen una línea base de funcionamiento, que si por el contrario, son aplicaciones independientes.

Esto deriva a su vez en la comunicación entre aplicaciones. Evitamos la necesidad de algún tipo de software intermedio para facilitar el flujo de información desde una aplicación a otra.

Las versiones se controlan de manera colectiva. Cambiar la versión del producto es consistente con la de sus módulos. Pongamos que un conjunto de aplicaciones evolucionan a diferente ritmo. Todo se traduce después en problemas.

Los ERP incluyen mecanismos de seguridad que van desde lo más inocente hasta evitar riesgos externos importantes tales como el espionaje industrial o malversación.

### **1.3. Desventajas**

No obstante existe una serie de desventajas derivadas del uso e integración de un ERP. Generalmente se deben a la falta de personal experto en este sector o a la errónea concepción de usar una aplicación que haga todo el trabajo.

El éxito a la hora de utilizar un ERP depende del factor humano. La habilidad, la experiencia, la educación en el sistema y la usabilidad de éste son factores que repercuten negativamente a la hora de su utilización. Para obtener el mayor beneficio de su uso es necesario conocerlo en profundidad y explotarlo debidamente. Haciendo uso de una analogía automovilística, de nada sirve comprarse el mejor automóvil del mercado si no sabemos cambiar de marchas y además no ponemos interés en aprender.

El problema se multiplica cuando existe cambio de personal. Se requiere que la compañía ofrezca cursos periódicos del uso de su ERP así como reciclaje de conocimientos tras el paso de las versiones.

Otro factor a tener en cuenta es la instalación del sistema. Sus requerimientos pueden variar, pero son notables en todos los casos debido al volumen de información que se requiere manejar. No es trivial, por tanto, instalar un sistema y dejarlo funcionando, si éste es sensiblemente complejo.

El factor del coste también es importante. Más adelante se hablará de los particulares y las tendencias sobre el uso de un ERP. Pero cabe destacar aquí que la renovación de licencias suele ser un factor negativo a la hora de plantearse trabajar con estos sistemas.

La concepción que se tiene de los ERP es de un sistema rígido y difícil de adaptar al flujo de los trabajadores y procesos de negocio. Ya se ha mencionado la necesidad de instruir al personal en el uso de un ERP para paliar estos defectos. Del mismo modo, la usabilidad puede ser un lastre y es que se plantea que un sistema que deba integrar tanta capacidad de comunicación y gestión es complejo de adaptar a un paradigma de usabilidad intuitivo y de

fácil aprendizaje.

También se debe tener en cuenta que un ERP gestiona la información global y que no deben existir departamentos que actúen como cuello de botella de la organización, pues se verá reflejado a la hora de usar la herramienta. A esto se le une que en ocasiones existe un cierto recelo en compartir información entre departamentos.

## 1.4. Elección práctica

A la hora de realizar la inmersión en un sistema real, evitando las lacras del pago de excesivas licencias o la adquisición de código propietario con tiempos de evaluación, nos hemos decantado por la utilización y explotación de un ERP libre, de fácil acceso y bien documentado: **OpenERP**.

En lo sucesivo, trataremos con este ERP a la hora de hablar de metodologías, marcos tecnológicos, diseños, etc.

## 1.5. Estructura del documento

Para finalizar esta introducción, incluiremos una pequeña descripción del documento a partir de este punto, en relación con el proyecto llevado a cabo.

- **Capítulo 2: Objetivos del proyecto.** Se abordarán los objetivos que se persiguen con la realización del proyecto.
- **Capítulo 3: Estado del Arte.** Donde se indicará la panorámica actual de los ERP así como las tendencias en el sector.
- **Capítulo 4: Metodologías usadas.** Se pondrá de manifiesto la metodología escogida para el particular, así como una serie de datos de documentación al margen de este documento, en relación a gestión del proyecto y del personal.
- **Capítulo 5: Marco tecnológico.** Para el caso particular, se indicará cómo instalar el sistema base así como los entresijos organizativos del sistema escogido.
- **Capítulo 6: Diseño de la herramienta.** Un caso de desarrollo particularizado para la opción ERP escogida así como un manual acerca de cómo comenzar y desarrollar un módulo
- **Capítulo 7: Pruebas.** Sobre el módulo desarrollado, se indicarán las pruebas efectuadas.
- **Capítulo 8: Conclusiones.** Donde se pondrá de manifiesto una perspectiva global del sector particularizada a su vez para el caso escogido para este proyecto.

## Capítulo 2

# Objetivos

Al realizar este proyecto, se pretende abarcar de forma general ciertos aspectos del sector de la Ingeniería de Software actuales. El tema escogido como se ha nombrado anteriormente ha sido el desarrollo y manejo de la tecnología ERP, tratando temas desde su instalación hasta el desarrollo de módulos nuevos para su integración. A continuación, se va a detallar los objetivos tanto generales como específicos que se quieren cumplir al realizar por completo este proyecto.

### 2.1. Objetivos generales

El objetivo principal del proyecto será desarrollar un software relacionado con ERP incorporando los conceptos metodológicos y tecnológicos específicos de esta tecnología; consiguiendo así alcanzar el nivel de destreza y soltura suficiente y necesaria que nos ayude a transmitir de una manera rápida y sencilla los conocimientos adquiridos a través de este documento.

Por lo tanto, el documento, junto con todos sus anexos y adjuntos externos, servirá como apoyo y en vista a poder tener una cultura general abarcando varios temas de este campo. Es decir, este documento no solamente servirá para la superación positiva del curso, sino como base, tanto para los creadores como para el resto de personas que puedan acceder libremente a él, a contraer la información y experiencia que se declara de una forma sencilla y rápida si en un futuro necesitan utilizar esta tecnología.

#### 2.1.1. Objetivos específicos

En primer lugar para abordar correctamente este proyecto, el objetivo se centrará en la declaración y creación de una empresa con la que podamos tener contacto entre otras con la organización, estimaciones tanto de presupuestos como de esfuerzo y horas de trabajo y cálculos de beneficios. La empresa a gestionar será “Bugging and debugging” creada durante el año 2012 por los mismos miembros de grupo de este proyecto. Más detalladamente los campos que se abordará en cuestión de la empresa serán incluidos en el documento como anexos y serán tenidos en cuenta a la hora de la organización de trabajo que trataremos en el apartado de metodologías usadas. Los temas abordados serán los siguientes:

- Un listado de roles con funciones específicas.
- Una estructura organizativa de la empresa.
- Un listado de personas asociadas a esos roles, como los miembros del equipo y las tareas adjudicadas.
- Un parque de recursos de computación, tanto hardware como software.
- Una previsión de gastos de inversión.
- Costes del personal.
- Beneficios de la realización del proyecto para la empresa.
- Situación empresarial e impuestos.

Con todo ello, se pretende motivar y tener toma de contacto en la visualización de creación de una empresa y el intento de aproximación de costes y beneficios empresariales.

Tras establecer y declarar la empresa, el objetivo se centrará en la creación de un calendario a seguir para la realización el proyecto, intentando abordar en cada una de las tareas e hitos todos los subobjetivos que posee el proyecto. Más adelante se explicaran las situaciones por las que ha pasado estas entregas y la realización del trabajo.

Además, a lo largo de todas las tareas que forman el proyecto, se pretende junto con ERP desarrollar un módulo que se podrá incorporar a cualquier pyme junto con el resto de módulos que ofrece OpenERP. Así mismo, se prestará atención a la correcta utilización del servicio que nos ofrece ERP.

Por último, se pretendrá implantar y emplear en el desarrollo del software todo lo aprendido a lo largo de nuestra formación académica.

## Capítulo 3

# Estado del Arte

A día de hoy encontramos una gran cantidad de paquetes ERP consistentes y fiables para integrar en el negocio. No obstante, el más popular sigue siendo **SAP ERP**. En este capítulo vamos a dar una visión global de lo que es SAP ERP así como un conjunto de alternativas de código abierto.

### 3.1. SAP ERP

**SAP AG** es una multinacional del software de origen alemán, reconocida por su implicación en el software empresarial. Los productos SAP se centran en las ERP, siendo **SAP ERP** su producto principal, actualmente en la versión 6.0, el cual se divide en las siguientes soluciones:

- **SAP ERP Finanzas:** centrado en la administración del capital económico, informes financieros y gestión de riesgos.
- **SAP ERP Manejo de capital humano:** automatiza los procesos de recursos humanos tales como la administración de empleados o gestión de nóminas.
- **SAP ERP Operaciones:** orientado al control de la logística, desarrollo de productos o ventas.

#### 3.1.1. Desarrollo. ABAP.

El desarrollo en SAP se basa en los lenguajes C, C++ y ABAP/4 (Advanced Business Application Programming, en alemán, *Allgemeiner Berichts-Aufbereitungs-Prozessor*). Este último desarrollado por la misma compañía como lenguaje de alto nivel con una sintaxis similar a la de COBOL, específico, de cuarta generación con el que se pueden generar dos tipos de programas ejecutables: informes o módulos para cualquier herramienta SAP.

#### 3.1.2. Sistemas compatibles

SAP ERP es compatible con una considerable cantidad de plataformas. No obstante, está más orientado a plataformas Windows que al resto.

- Microsoft Windows
- OS/400
- Unix
- Solaris
- AIX
- HP UX
- Sinux

### 3.1.3. Licencias SAP

Cuando se adquiere un producto SAP ERP, se compra un kit con un conjunto de funcionalidades, especificando el número de usuarios que va a tener y su tipología. Lo normal es adquirir un estándar general:

1. Finanzas
2. Recursos humanos
3. Control y Planificación
4. Logística, compras y almacenamiento
5. Calidad
6. Mantenimiento
7. Producción
8. Ventas
9. etc.

También podemos adquirir una versión ERP más específica, denominada *vertical*, pudiendo encontrar módulos más allá de lo necesario, dependiendo del sector.

Todo ello requiere una licencia que se acumula en función de cuantos servicios queramos contratar. Finalmente, al importe de la compra se le debe añadir un porcentaje por uso de la base de datos correspondiente. Entre las disponibles para SAP tenemos:

- Informix
- Oracle
- Adabas
- Sysbase ASE
- IBM DB/2
- Microsoft SQL Server

Con la cifra definitiva se calcula el mantenimiento anual a pagar. Se distinguen dos versiones del mantenimiento: *Standard* con un 17 % del total y *Enterprise*, con un 22 %. Este último habilita actualizaciones, parches y soporte de alto nivel entre otras.

### 3.2. ERPs de código abierto actuales

Como alternativa más económica a SAP, encontramos una gran variedad de software código abierto que, si bien tiene sus limitaciones, no dejan de ser ERPs funcionales muy completos y competitivos. A continuación, una lista con los más extendidos.

ERP Package	Language base	License	Developer country
<i>AI.iO</i>	Java	Alliance Technologies	Worldwide
<i>Adaxa Suite</i>	Java	GPL	Australia/N.Zealand
<i>Adempiere</i>	Java	GPL	Spain
<i>Compiere</i>	Java	GPL/Commercial	US
<i>Dolibarr</i>	PHP, MySQL	GPL	
<i>EpesiBIM</i>	PHP, MySQL	MIT license	Poland, USA
<i>ERP5</i>	Python, Zope, MySQL	GPL	Brazil, France, Germany, Japan, Senegal
<i>ERPNext</i>	PHP, MySQL	GPL	India
<i>Fedena</i>	Ruby, MySQL	Apache License	India
<i>FrontAccounting</i>	PHP, MySQL	GPLv3	
<i>GNU Enterprise</i>	Python	GPLv3	
<i>HeliumV</i>	Java	AGPL	Austria, Germany
<i>JFire</i>	Java	LGPL	
<i>Kuali Foundation</i>	Java	ECL	
<i>LedgerSMB</i>	Perl, PostgreSQL	GPL	Worldwide
<i>OFBiz</i>	Apache, Java	Apache License 2.0	
<i>Openbravo</i>	Java	OBPL	Spain
<i>OpenERP</i>	Python, PostgreSQL	AGPLv3	Belgium, India, USA
<i>Phreedom</i>	PHP, JavaScript, MySQL	GPLv3	USA
<i>Postbooks</i>	C++, JavaScript, PostgreSQL	CPAL	
<i>SQL-Ledger</i>	Perl, PostgreSQL	GPL	
<i>Tryton</i>	Python	GPLv3	
<i>WebERP</i>	PHP, MySQL	GPLv2	

Cuadro 3.1: Tecnologías ERP abiertas



### 3.3. OpenERP

Fuente: [Wik13]

A continuación se describe OpenERP al ser el escogido para llevar a cabo el proyecto. Se darán unas pinceladas a su organización, delegando descripciones tecnológicas más precisas en capítulos posteriores.

#### 3.3.1. Introducción

OpenERP es una alternativa ERP de código abierto. Está publicado bajo licencia AGPL y actualmente se encuentra en su versión 7.0 (22 de diciembre de 2012).

Está compuesto por una serie de módulos estándar que abarcan los siguientes ámbitos:

- Administración de ventas
- CRM
- Gestión de proyectos
- Sistemas de gestión de almacenes
- Manufactura
- Contabilidad analítica y financiera
- Puntos de venta
- Gestión de activos
- Gestión de Recursos Humanos
- Gestión de inventario
- Ayuda técnica
- Campañas de marketing
- Flujos de trabajo

#### 3.3.2. Desarrollo

En la sección correspondiente se tratará sobre la arquitectura de OpenERP así como las aplicaciones. En concreto, OpenERP hace uso de Python como lenguaje de programación y PostgreSQL como sistema gestor de bases de datos.

#### 3.3.3. OpenERP como servicio

A partir de la versión 6.0, OpenERP facilita una versión como servicio. A través de su portal web se puede acceder a un cliente ligero que actúa contra una base de datos de demostración.

### 3.3.4. Licencias

Pese a que OpenERP y sus módulos son de código libre, se puede contratar una licencia que incluye:

- Acceso a módulos a través del web
- Actualizaciones automáticas
- Actualizaciones a futuras versiones
- Soporte funcional
- Hosting incluido
- Garantía de corrección de errores

### 3.3.5. Compatibilidad

OpenERP está orientado a trabajar principalmente bajo GNU/Linux, no obstante se pueden encontrar clientes y servidores para las siguientes plataformas:

- Windows
- Linux
- Unix
- MacOSX
- Android

## Capítulo 4

# Metodologías

### 4.1. Scrum

La metodología que se ha seguido a lo largo de todo el proyecto ha sido centrada a la utilización de metodologías ágiles, más concretamente, Scrum.

Scrum es un marco de trabajo para la gestión y desarrollo de software basada en un proceso iterativo e incremental utilizado comúnmente en entornos basados en el desarrollo ágil de software. No estando solamente enfocado a la gestión de procesos de desarrollo de software, sino a la gestión de proyectos en general. El objetivo de esta metodología es obtener resultados rápidos, adaptándose a los cambios de las necesidades de los clientes, siguiendo así un ciclo de vida iterativo e incremental, mejorando la gestión de los riesgos y aumentando la comunicación. El desarrollo de Scrum es basado en entregas parciales junto con celebraciones de reuniones de coordinación a lo largo de todo el proyecto. Por lo tanto, siguiendo esta metodología conseguimos que el equipo de desarrollo esté autoorganizado, multifuncional, no distribuidos, y de tamaño óptimo.

### 4.2. Roles

Los roles de Scrum y sus correspondencias que se han asignado a lo largo de todo el proyecto son:

- **Producto Owner:** En este proyecto este papel ha sido desempeñado por Ismael Caballero Muñoz, el cual, ha definido las características del producto, así como también las fechas de publicación y entrega. También ha priorizado y ajustado las características del producto según sea necesario en cada iteración. Y por último, aceptará o rechazará los resultados del trabajo.
- **Scrum Master:** Corresponde a Raúl Reguillo Carmona, representante de la gestión del proyecto, es el responsable de mantener y asegurar los valores del proyecto, asegurando que el equipo permanezca totalmente funcional y productivo, así como la comunicación y cooperación entre todos los componentes del proyecto y las funcionalidades.

- **Team:** se ajusta al equipo de desarrolladores los cuales trabajan a tiempo completo. Este equipo es formado por Cristian Del Cerro Gómez, Laura Del Río Avilés y Javier García Simón. Además, el Scrum Master de este proyecto, Raúl Reguillo, también trabajará como componente del equipo de desarrollo.

Todos y cada uno de los componentes del grupo de desarrollo de este proyecto podrán realizar tareas como desarrolladores, diseñadores, consultores, formadores e incluso de mantenimiento.

### 4.3. Product Backlog y Sprints

El proyecto ha ido avanzando a causa de una serie de entregables e hitos resultantes de las iteraciones incrementales que se han ido realizando en forma de sprints de 5 a 7 días. El Sprint es el período en el cual se lleva a cabo el trabajo en sí.

La duración de los sprints fue establecida por los miembros del equipo como periodos de una semana. Sin embargo, a causa de las circunstancias de la aproximación de la fecha de entrega del producto, ha visto conveniente ajustar y disminuir el tiempo de los sprints. Por lo tanto, se decidió que cada uno tuviese una duración de 4 a 6 días. Al final de cada sprint, el equipo ha realizado una reunión para presentar los avances logrados e ir recopilando el material necesario para abordar por completo el proyecto.

El conjunto de características que forma parte de cada sprint viene del Product Backlog, que es un conjunto de tareas que definen el trabajo a realizar.

Estas tareas se han representado a través de Microsoft Project, herramienta de administración de proyecto, que nos ha ayudado a la organización y asignación de tareas. También esta herramienta nos ha ayudado a través del diagrama de Gantt a determinar qué tareas son las que forman el camino crítico en el proyecto, así pudiéndolas realizar con un mayor esfuerzo para que no ocasionasen retrasos a las demás. A continuación se mostrarán la lista de tareas en forma de tabla que se han ido siguiendo junto con el diagrama y calendario correspondiente. Obsérvese que el diagrama de Gantt es el correspondiente a fecha del 3 de Mayo del 2013 ya que las tareas ya han sido realizadas al 100

### 4.4. Reuniones

A lo largo de la trayectoria que ha ido siguiendo el proyecto y acogiéndose a la estructura organizativa de Scrum, se han realizado una serie de reuniones que se pueden clasificar en las siguientes categorías:

- Reuniones de planificación del Sprint: identificación de tareas.
- Reuniones diarias (Daily Scrum): visión de la situación.









































Id	Modo de tarea	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras	Nombres de los recursos
1		Revisión de la Empresa "Bugging and Debugging"	1,56 días	jue 18/04/13	vie 19/04/13		
2		Roles y personas asociadas	0,06 días	jue 18/04/13	jue 18/04/13		
3		Esquema estructural organizativo	30 mins	jue 18/04/13	jue 18/04/13		Raul Reguillo
4		Parque de recursos (hardware y software)	30 mins	jue 18/04/13	jue 18/04/13		Javier García
5		Gastos de inversión	0,38 días	jue 18/04/13	jue 18/04/13	2	
6		Plan de inversión	2 horas	jue 18/04/13	jue 18/04/13		Cristian Del Cerro
7		Impuestos a pagar	30 mins	jue 18/04/13	jue 18/04/13		Raul Reguillo
8		Presupuestos	0,13 días	jue 18/04/13	jue 18/04/13	6;7	Laura Del Rio
9		Coste del personal	45 mins	jue 18/04/13	jue 18/04/13		
10		Mantenimiento físico	1 hora	jue 18/04/13	jue 18/04/13		
11		Beneficio del proyecto	8 horas	jue 18/04/13	vie 19/04/13	5	Cristian Del Cerro; Laura Del Rio
12		Creación de documentos acreditativos	2 horas	vie 19/04/13	vie 19/04/13	2;5FC-30 mins;11FC-1	Raul Reguillo
13		E1 - Hito Revisión	30 mins	lun 22/04/13	lun 22/04/13	1	Cristian Del Cerro; Raul Reguillo; Javier García; Laura Del Rio
14		Establecimientos de requisitos	2 días	lun 08/04/13	mié 10/04/13	1CC	Cristian Del Cerro; Laura Del Rio; Javier García; Raul Reguillo
15		Obtención de requisitos	4 horas	lun 08/04/13	lun 08/04/13		
16		Especificación de Casos de Uso	12 horas	lun 08/04/13	mié 10/04/13	15	
17		E2 - Hito Presupuesto y previsión	30 mins	jue 11/04/13	jue 11/04/13		Cristian Del Cerro; Laura Del Rio; Raul Reguillo; Javier García
18		Análisis de casos de uso	2,63 días	mié 10/04/13	vie 12/04/13	14	Cristian Del Cerro; Javier García; Laura Del Rio; Raul Reguillo
19		Identificación de Clases asociadas a un Caso de Uso	12 horas	mié 10/04/13	jue 11/04/13		Raul Reguillo; Javier García; Laura Del Rio; Cristian Del Cerro
20		Descripción de la Interacción de Objetos	12 horas	jue 11/04/13	vie 12/04/13	19FC-3 horas	Javier García; Cristian Del Cerro; Laura Del Rio; Raul Reguillo
21		Análisis de Clases	1,75 días	vie 12/04/13	mar 16/04/13	18	
22		Identificación de responsabilidades y atributos	10 horas	vie 12/04/13	mar 16/04/13		Raul Reguillo; Javier García
23		Identificación de Asociaciones y Agregaciones	9 horas	lun 15/04/13	mar 16/04/13	22FC-5 horas	Laura Del Rio; Cristian Del Cerro
24		Definición de interfaces de usuario	10,13 días	mar 16/04/13	mar 30/04/13	21	
25		Especificación de interfaz	7 horas	mar 16/04/13	mar 30/04/13		Laura Del Rio; Javier García
26		Diseño de Clases	0,88 días	jue 18/04/13	jue 18/04/13	21	
27		Diseño e Identificación para la realización de las clases	7 horas	jue 18/04/13	jue 18/04/13		Cristian Del Cerro; Raul Reguillo; Javier García; Laura Del Rio
28		Hito Reunión de documentación elaborada	30 mins	jue 02/05/13	jue 02/05/13	14;18;21;24;26	Cristian Del Cerro; Javier García; Laura Del Rio; Raul Reguillo
29		Instalación de sistema	2,38 días	mié 17/04/13	vie 19/04/13		Raul Reguillo; Javier García; Cristian Del Cerro; Laura Del Rio
30		Busqueda de información	14 horas	mié 17/04/13	jue 18/04/13		
31		Instalación del sistema base	3 horas	jue 18/04/13	vie 19/04/13		Postgre
32		Instalación del entorno de desarrollo	5 horas	jue 18/04/13	vie 19/04/13		OpenERP
33		Creación de manuales	3 horas	jue 18/04/13	jue 18/04/13		
34		Generación del código de los componentes y procedimientos	7,5 días	vie 19/04/13	mar 30/04/13	29FC+2 horas;26	Cristian Del Cerro; Javier García; Laura Del Rio; Raul Reguillo
35		Generación del código de componentes	60 horas	vie 19/04/13	mar 30/04/13		
36		Ejecución de las pruebas	2,5 días	mar 30/04/13	jue 02/05/13	34	
37		Ejecución de las pruebas unitarias	18 horas	mar 30/04/13	jue 02/05/13		Javier García; Cristian Del Cerro
38		Ejecución de las pruebas de integración	20 horas	mar 30/04/13	jue 02/05/13		Raul Reguillo; Laura Del Rio
39		E4 - Hito Entrega Proyecto/ prototipo	30 mins	vie 03/05/13	vie 03/05/13	36;34;29;26;21	Cristian Del Cerro; Javier García; Raul Reguillo; Laura Del Rio
40		E5 - Hito Exposición Proyecto	20 mins	lun 06/05/13	lun 06/05/13	39	Cristian Del Cerro; Javier García; Laura Del Rio; Raul Reguillo

Figura 4.1: Tabla de tareas

- Reunión del Revisión del Scrum (Sprint Review Meeting): presentación de lo desarrollado en un sprint.
- Retrospectiva del Sprint (sprint Retrospective): visión del proyecto tras un sprint.

Todas y cada una de las reuniones realizadas a lo largo del proyecto se han ido registrando siguiendo una plantilla mostrada a continuación. Todas salvo las reuniones diarias, las cuales se han llevado a cabo de una manera más informal.

Se han realizado un total de 23 reuniones que se pueden resumir en la siguiente tabla.

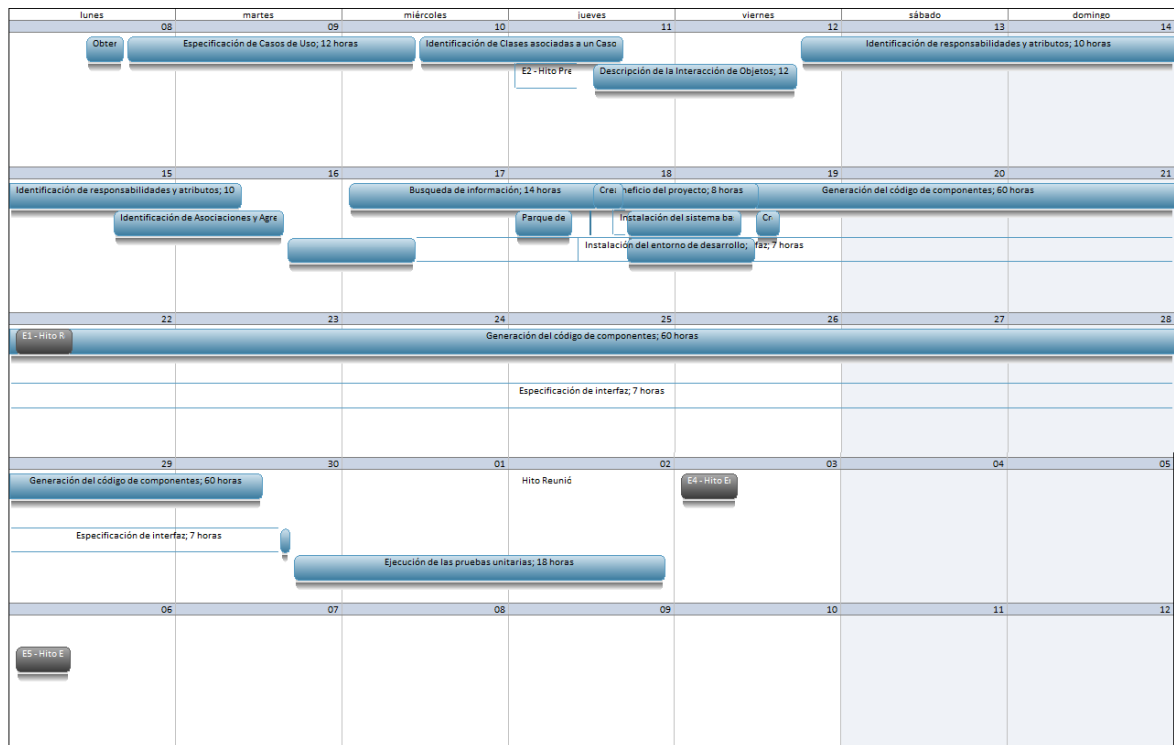


Figura 4.2: Calendario de tareas

A causa del gran número de reuniones de equipo, Como anexo se publicarán las actas de reunión de las tenidas a lo largo de todo el proyecto. Sin embargo, a causa del gran número de reuniones de equipo, de esta categoría solamente se publicarán algunas que serán tomadas como ejemplo del resto.

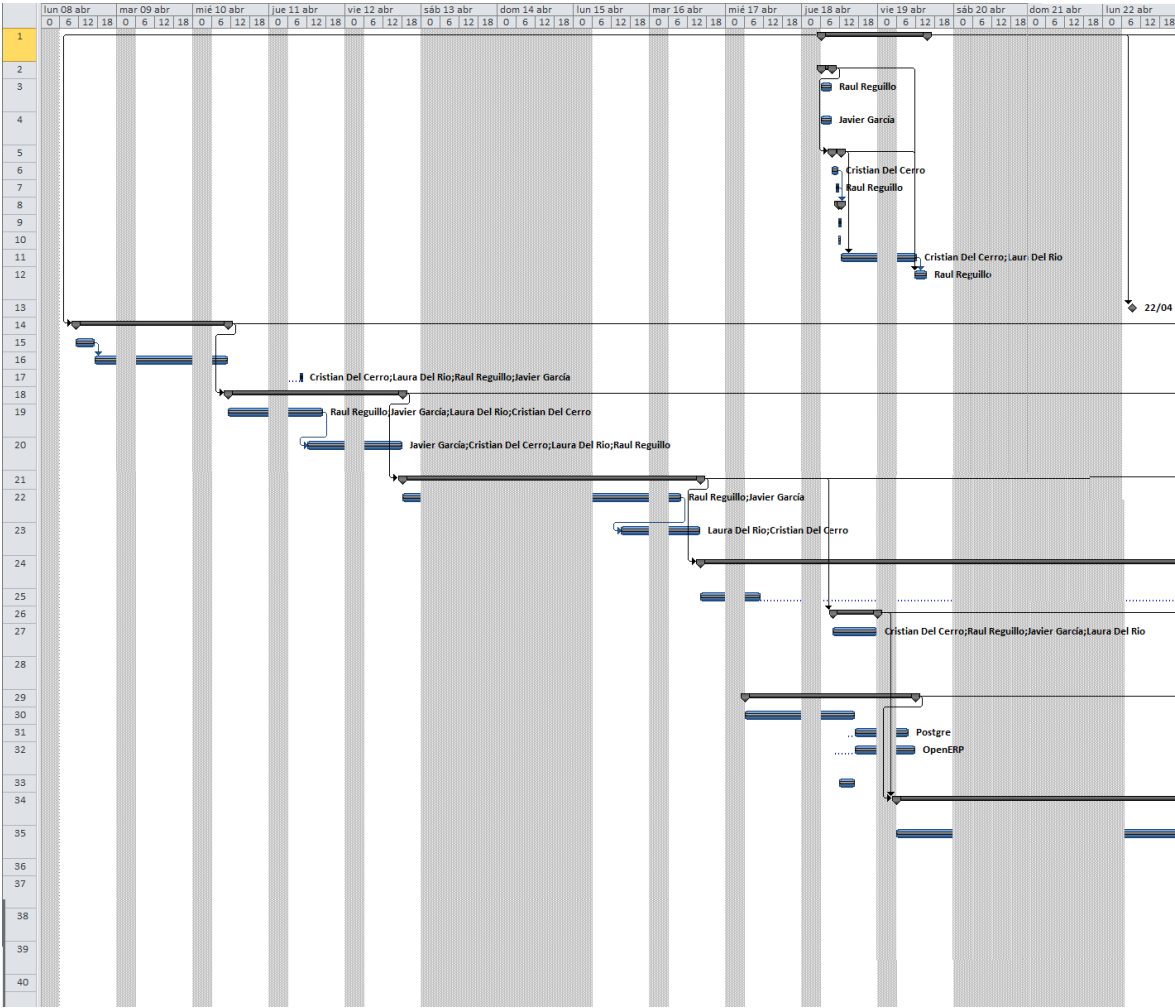


Figura 4.3: Diagrama de Gantt (I)

Reuniones con	Número	Motivo
Ismael Caballero Muñoz	3	Revisión de la trayectoria del proyecto
Macario Polo Usaola	2	Orientación en el uso de OpenERP
Equipo	17	Realización y avance del proyecto

Cuadro 4.1: Reuniones llevadas a cabo

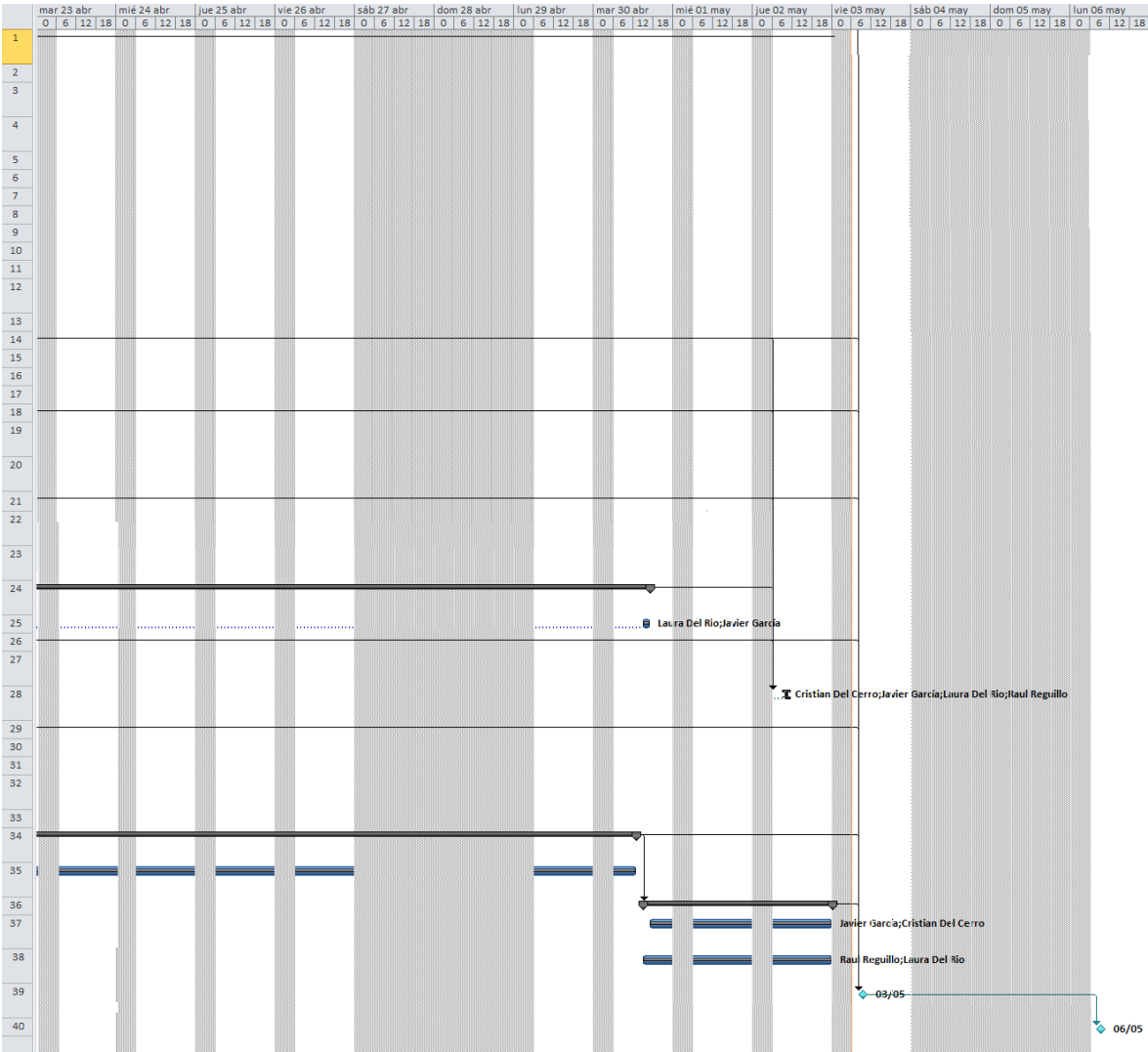


Figura 4.4: Diagrama de Gantt (y II)



Grupo de proyecto  
Ingeniería del Software  
Curso 2012-2013



Escuela  
Superior  
de Informática

## ACTA DE REUNIÓN

### Ingeniería del Software

Reunidos en \_\_\_\_\_ del edificio \_\_\_\_\_, el día \_\_\_\_\_ de \_\_\_\_\_ de \_\_\_\_\_, a las \_\_\_\_\_ horas, constituyendo ésta la \_\_\_\_\_ reunión, con la asistencia de todos los miembros del grupo, se propuso \_\_\_\_\_ siguiendo este esquema:

- 1.- \_\_\_\_\_.
- 2.- \_\_\_\_\_.
- 3.- \_\_\_\_\_.

### DESARROLLO DE LA SESIÓN

---

---

---

---

---

---

---

La duración aproximada de la reunión ha sido de \_\_\_\_\_.

Ciudad Real, \_\_\_\_\_

Cristian del Cerro Gómez  
Javier García Simón  
Laura del Río Avilés  
Raúl Reguillo Carmona

Figura 4.5: Informe de ejemplo

# Marco tecnológico

### 5.1. Arquitectura

OpenERP es un sistema multiusuario con una arquitectura en tres capas: una capa de datos para almacenar la información, una capa de aplicación para la funcionalidad y la lógica de negocio y una capa de presentación para la interfaz de usuario. Todas estas capas están separadas en OpenERP. La capa de aplicación está escrita como si fuera el núcleo y es donde se añaden los módulos específicos para adaptar o crear una versión de OpenERP concreta acorde con las necesidades concretas.

El sistema OpenERP tiene tres componentes fundamentales:

- Una base de datos PostgreSQL que contiene todas las bases de datos de usará OpenERP, contienen desde los datos de la aplicación hasta la mayoría de los elementos configuración del mismo OpenERP. También existe la posibilidad de utilizar bases de datos distribuidas.
- Un servidor OpenERP que contiene toda la lógica de negocio y se encarga de que OpenERP funcione correctamente. Una capa del servidor se encarga de comunicar con la base de datos PostgreSQL, el motor ORM. Otra capa permite la comunicación entre el servidor y el navegador web, la capa web. Es posible tener más de un servidor, por ejemplo para equilibrar la carga del sistema.
- Un cliente funcionando en un navegador web como una aplicación javascript.

A primera vista parece que la arquitectura en tres capas se corresponde con el patrón MVC (Modelo-Vista-Controlador) pero hay una diferencia fundamental en el funcionamiento de ambas. En la arquitectura en tres capas la capa cliente nunca se comunica directamente con la capa de datos, toda la comunicación ha de pasar necesariamente por la capa intermedia. Sin embargo en una arquitectura MVC la vista envía actualizaciones al controlador, el controlador actualiza el modelo y la vista recibe las actualizaciones directamente desde el modelo. Podríamos decir que una arquitectura en tres capas es una arquitectura lineal y la arquitectura MVC es triangular.

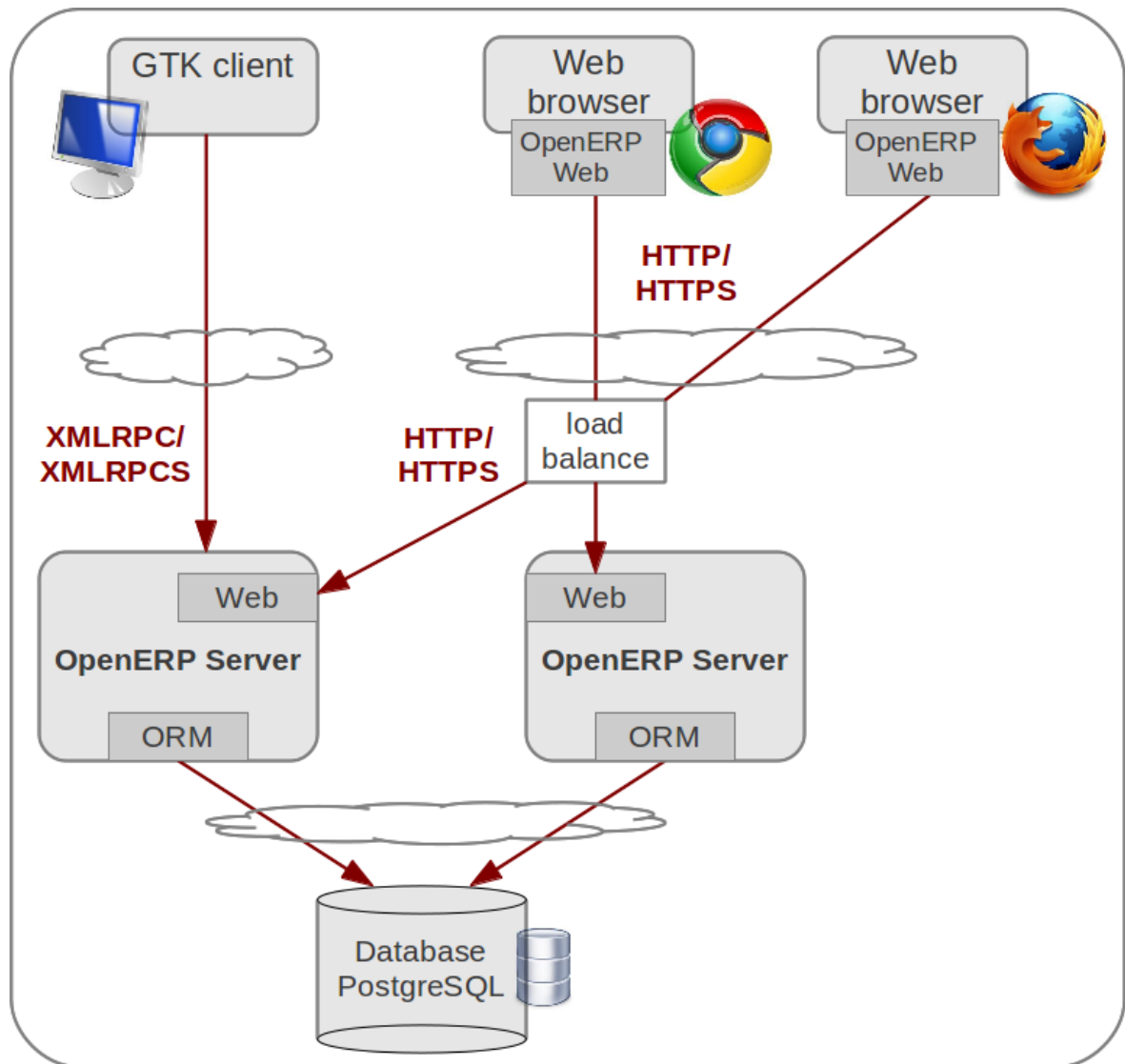


Figura 5.1: Arquitectura global

### 5.1.1. Base de datos PostgreSQL

La capa de datos de OpenERP la proporciona una base de datos relacional PostgreSQL. Aunque las consultas SQL pueden ejecutarse directamente desde los módulos OpenERP, la mayoría de los accesos a la base de datos relacional se hacen a través del servidor de la capa de mapeo objeto relacional.

### 5.1.2. Servidor OpenERP

OpenERP facilita una aplicación servidora sobre la que se pueden construir aplicaciones para una lógica de negocio concreta. Provee un framework de desarrollo completo desde el que se pueden contruir dichas aplicaciones.

Desde la perspectiva del desarrollador, el servidor actua como una biblioteca que oculta los

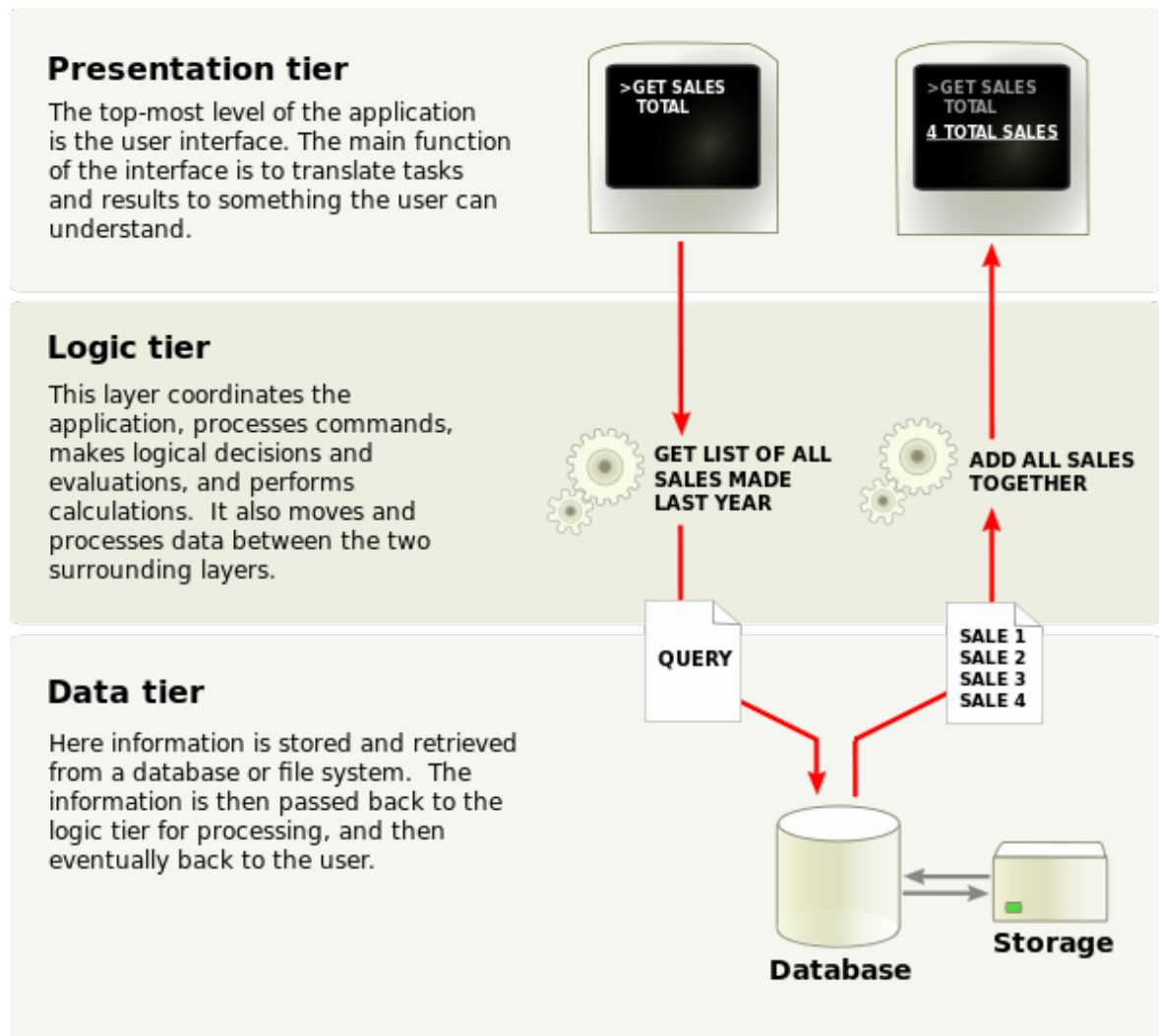


Figura 5.2: Model-View-Controller

detalles a bajo nivel y que ayuda a instalar y configurar las aplicaciones. Además el servidor facilita servicios como son las vistas, el motor de flujos de trabajo o el motor generador de informes.

### 5.1.3. Servidor ORM

La capa ORM es una de las características destacadas del servidor OpenERP. Los modelos de datos se describen en Python y OpenERP crea las tablas de las bases de datos usando tu ORM. Es importante destacar la importancia de entender la responsabilidad del ORM antes de intentar acceder a la base de datos a través de consultas directas a la base de datos. Cuando se usa una ORM, OpenERP se asegura de que los datos son coherentes y correctos. Por ejemplo, un módulo podría reaccionar de alguna manera al introducir datos nuevos en una tabla determinada, pero esa reacción sólo tendrá lugar si se usa una ORM y no si se ejecuta la inserción directamente sobre la base de datos.

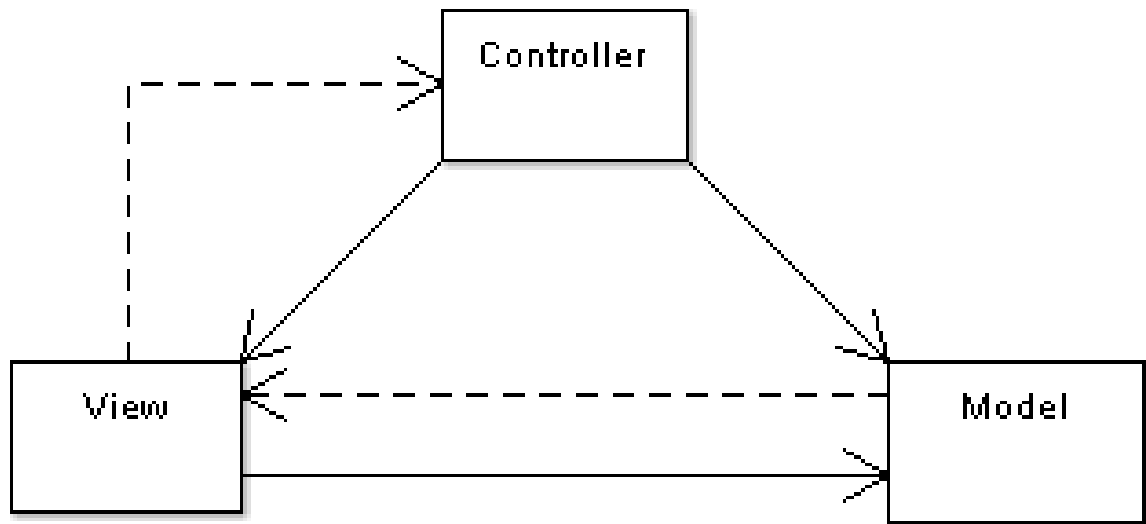


Figura 5.3: Relaciones MVC

Los servicios garantizados por ORM son, entre otros:

- Consistencia de los datos
- Proveer una interfaz que permita diseñar e implementar módulos eficientemente
- Seguridad a bajo nivel para usuarios y grupos
- Ejecutar acciones complejas sobre grupos de recursos

#### 5.1.4. Servidor web

La capa web permite a una interfaz comunicarse con un navegador. El servidor maneja consultas http para servir ficheros estáticos o contenidos dinámicos y consultas JSON-RPC(Remote Procedure Call codificado a través de JSON) de las RPC's(Remote Procedure Call) hechas desde el navegador.

#### 5.1.5. Módulos

Dependiendo de cada compañía el valor de OpenERP recaerá en diferentes módulos. El papel que juegan los módulos es el de implementar cualquier requerimiento de negocio. El servidor es el único componente necesario para añadir cualquier módulo de los que ya están incluidos en la distribución oficial de OpenERP o uno de los cientos que disponibles desarrollados por la comunidad.

#### 5.1.6. Clientes

Como la lógica de la aplicación está en el lado del servidor el cliente es muy sencillo. Solicita datos al servidor, los recupera y muestra los resultados en distintas maneras (formularios,

listas, calendarios, gráficos, etc). A través de la interacción del usuario se modifican datos que se envían al servidor. La aplicación por defecto de OpenERP es una aplicación JavaScript corriendo sobre un navegador que se comunica con el servidor usando JSON-RPC(Remote Procedure Call codificado a través de JSON).

## Capítulo 6

# Diseño de la herramienta

Enlazando con el punto anterior, se define el diseño de la herramienta como el de un módulo individual. Así pues, el diseño de la herramienta constará de las funcionalidades específicas aplicadas al patrón básico de representación en OpenERP. Estas funcionalidades se pueden extender más o menos pero básicamente son todas similares. Añadir una funcionalidad puede constar de añadir un directorio al del proyecto, en el que se implementen los archivos necesarios. También es posible definir nuevas vistas para cada parte, como se verá. No obstante, lo que debe quedar claro, es que un módulo en esencia comparte una misma estructura de diseño con otro cualquiera, independientemente de su complejidad.

### 6.1. Estructura de un módulo

Un módulo puede contener cualquiera de los elementos siguientes:

- Objetos de negocio: son clases Python que extienden de la clase `osv.Model`, la persistencia de estos recursos es manejada en su totalidad por el ORM de OpenERP.
- Datos: los archivos XML/CSV con metadatos, datos de configuración (los parámetros de los módulos) y los datos de prueba (opcionales pero recomendados para la realización de pruebas).
- Informes: plantillas para informes en formato RML (Report Markup Language), `html/mako` u OpenOffice pueden ser rellenas con cualquier tipo de datos del negocio y generar informes en `html`, `odt` o `pdf`.

Para crear un nuevo módulo hacen falta los siguientes pasos.

- Crear una carpeta con el nombre del módulo
- Crear un fichero que importe el módulo (`__init__.py`)
- Crear un manifiesto con los campos descriptivos más importantes (`__openerp__.py`)
- Crear un fichero Python que describe los objetos
- Crear un fichero `xml` que contenga los datos del módulo a través de vistas, menús o datos de prueba

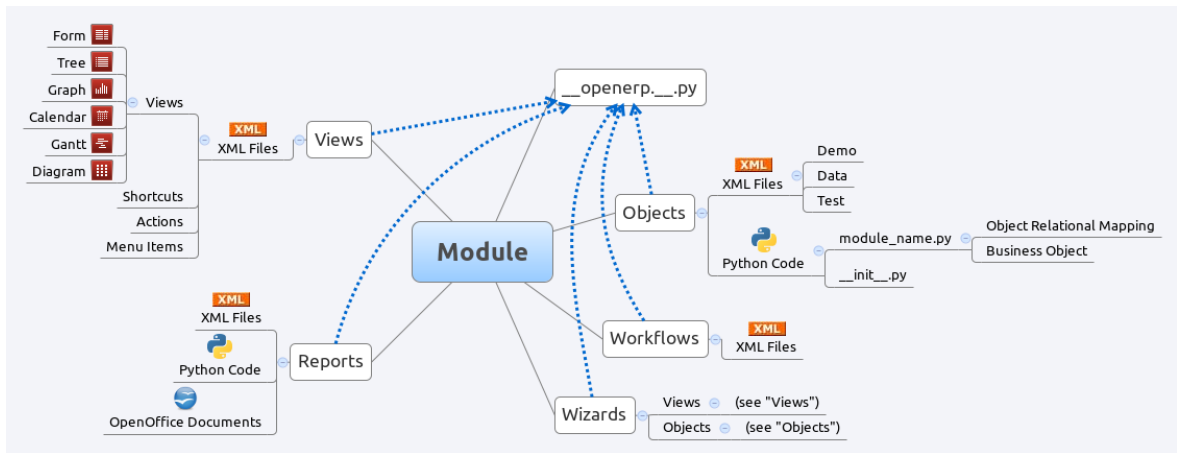


Figura 6.1: Estructura de un módulo

- Opcionalmente pueden crearse informes o flujos de trabajo.

### 6.1.1. Python import file `__init__.py`

Un módulo OpenERP es un módulo Python por lo que este fichero importará todos los ficheros Python u otros submódulos.

### 6.1.2. Mafinfiesto `__openerp__.py`

Este fichero no es más que un diccionario de términos literales de Python que se encargará de:

1. Qué ficheros xml deben ser parseados durante la inicialización del servidor.
2. Enumerar las dependencias del módulo creado con otros.
3. Declarar metadatos adicionales

Este fichero debe contener los siguientes valores, o al menos todos los necesarios:

### 6.1.3. Objetos

Todos los recursos en OpenERP son objetos, incluso los metadatos como los menús, acciones, informes, etc. son también objetos. El ORM de OpenERP está construido sobre PostgreSQL. Es posible consultar un objeto usado por OpenERP usando ORM o directamente usando sentencias SQL pero es realmente peligroso hacerlo así, puesto que utilizar ese 'atajo' se saltaría pasos importantes como comprobación de restricciones, modificaciones del flujo de trabajo o incluso podría suceder que diversos eventos asociados a algunas acciones no se disparasen.



### 6.1.4. Ficheros XML

Los ficheros XML se usan para inicializar o actualizar la base de datos cuando el módulo se instala o actualiza. Algunos de sus usos son:

- Declaración e inicialización de los datos de prueba
- Declaración de vistas
- Declaración de informes
- Declaración de flujos de trabajo

Una manera de insertar datos a través de un fichero XML sería como sigue:

```
<?xml version="1.0"?>
<openerp>
  <data>
    <record model="model.name_1" id="id_name_1">
      <field name="field1"> "field1 content" </field>
      <field name="field2"> "field2 content" </field>
      (...)
    </record>
    <record model="model.name_2" id="id_name_2">
      (...)
    </record>
    (...)
  </data>
</openerp>
```

Listado 6.1: Ejemplo de vista en XML

Con la etiqueta <record>insertamos nuevos datos en el objeto especificado con el atributo (obligatorio) <model>. Si queremos utilizar este recurso posteriormente podemos hacerlo referencia si definimos un valor al atributo id (opcional).

Un <record> puede contener campos del tipo <field> que indican los valores de los campos correspondientes, si no se especifica ninguno se usan los valores por defecto. Los atributos de un <field> son:

1. name (obligatorio): indica el nombre del campo
2. eval (opcional): una expresión en lenguaje Python para obtener el valor del campo
3. ref: una referencia a un id definido en el mismo fichero
4. model: el modelo que aparece en la búsqueda
5. search: permite encontrar los registros cuando no sabemos su id en el xml

### 6.1.5. Vistas

Las vistas son una manera de representar los objetos al cliente y se describen a través de XML. Indican al cliente como distribuir los datos de los objetos en la pantalla. Hay dos tipos de vistas: las vistas de formulario y las vistas de árbol (las listas son un tipo particular de árbol). Un mismo objeto puede tener varias vistas, el primer tipo que se defina será tomado como la vista por defecto. Así se puede tener una vista de árbol por defecto y otra más específica con más o menos información que se mostrará por ejemplo al hacer doble click sobre un ítem.

#### Ejemplo de uso

Imaginemos que queremos abrir un objeto.

- Una acción abre el objeto (se recuperan los datos del objeto, la vista y el dominio)
- El cliente solicita (con XML-RPC) al servidor qué vistas están definidas para ese objeto y qué datos deben mostrarse
- El cliente muestra los datos y los distribuye acorde a la vista.

### 6.1.6. Desarrollar nuevos objetos

Para desarrollar un nuevo tipo de objeto hay que hacer lo mínimo: crear el objeto y opcionalmente crear las vistas de ese objeto. No hay que escribir ninguna tabla a mano ni nada por el estilo en la base de datos PostgreSQL puesto que los propios objetos las crean automáticamente.

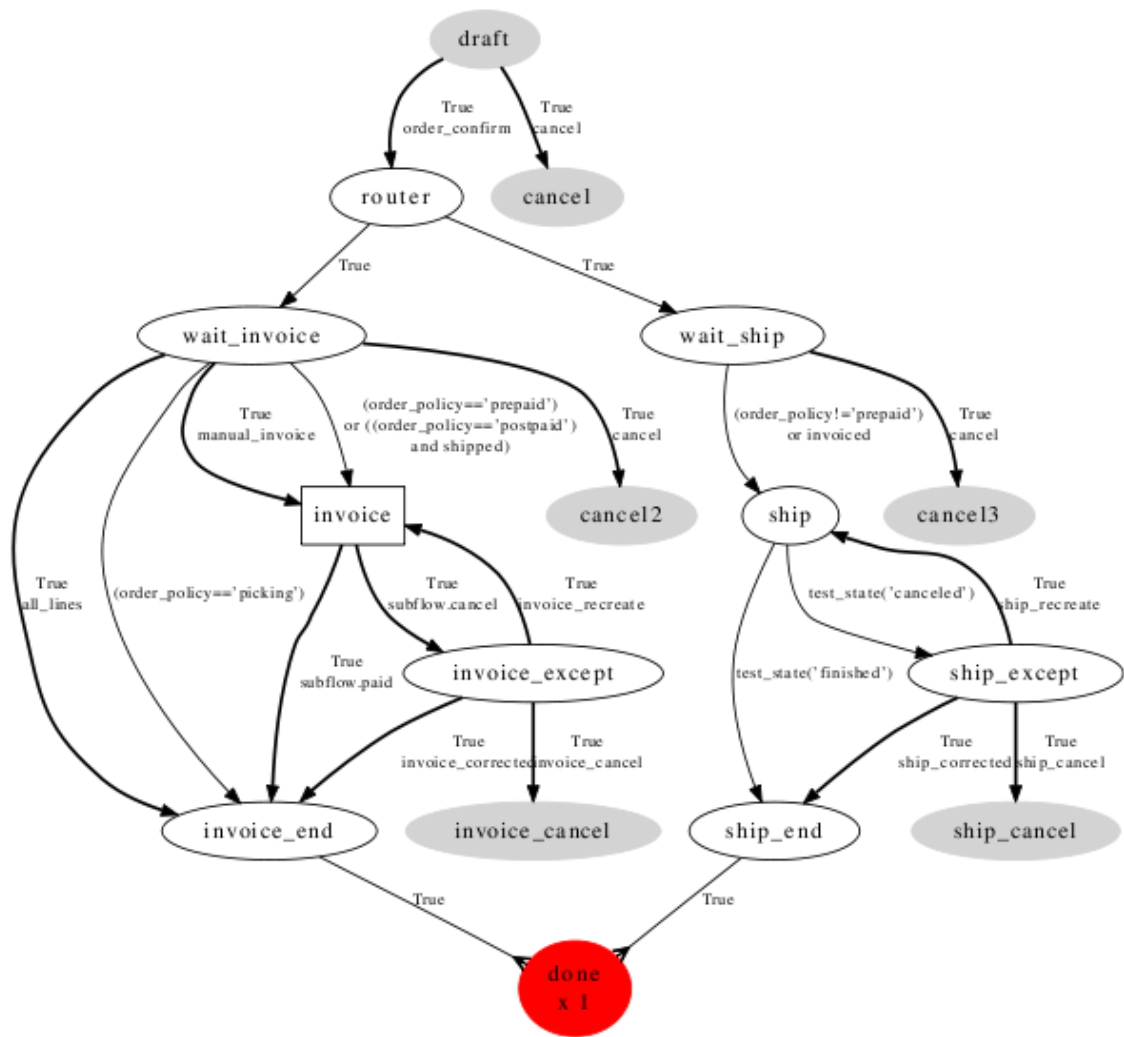
#### Flujos de trabajo

Los objetos y las vistas permiten definir formularios nuevos de forma muy simple, así como árboles/listas y las interacciones entre ellos. Aún así, no es suficiente en definirlos, hay que definir el comportamiento de esos objetos, por ejemplo:

- La confirmación de una venta debe generar una factura de acuerdo a ciertas condiciones
- Una factura pagada, bajo ciertas condiciones, hace que comience el proceso de envío

Los flujos de trabajo se describen estas interacciones con gráficos, los cuales se transcriben a ficheros xml y se asocian a los objetos.

En el gráfico cada nodo representa las acciones que se deben realizar (wait\_invoice, ship). Las flechas son las condiciones para llevar a cabo esas acciones (test\_state(finished), ship\_cancel). Los rectángulos representan otros flujos de datos (invoice).



Workflow: sale.order.basic

OSV: sale.order

Figura 6.2: Workflow

## 6.2. Una aplicación real: *Bugginv5*

Este módulo es propio de la empresa Buggin and Debuggin y se utiliza en su propio beneficio. El objetivo es el control de todos los ingresos y gastos que se producen en la empresa organizado por meses, para llevar un control estricto de la economía. Los datos que se requieren son los siguientes: -Mes -Gastos por salarios -Gastos por licencias -Gastos no planificados -Ingresos por cuotas -Ingresos por publicidad -Responsable de la introducción de los datos. (Por si surgen errores saber el responsable).

El módulo tiene la capacidad de realizar un gráfico comparativo de los ingresos por cuotas organizados por cada mes.

★ Search: Gastos e ingresos

Search Clear Filters

Gastos e ingresos New

MES	GASTO EN SALARIOS	GASTO EN LICENCIAS	GASTO NO ESPERADO	INGRESO EN CUOTAS	INGRESO EN PUBLICIDAD	RESPONSABLE DATOS
Enero	150.0000	240.0000	200.0000	1,200.0000	290.0000	Raúl
Febrero	1,400.0000	0.0000	0.0000	8,000.0000	200.0000	Cristian
Febrero	0.0000	200.0000	300.0000	400.0000	200.0000	Laura
Enero	500.0000	50.0000	90.0000	400.0000	200.0000	Cristian
Marzo	4,500.0000	0.0000	0.0000	8,000.0000	200.0000	Laura
Marzo	0.0000	0.0000	0.0000	300.0000	0.0000	Raúl
Abril	6,000.0000	0.0000	800.0000	12,000.0000	200.0000	Raúl

Figura 6.3: Relación de ventas por mes

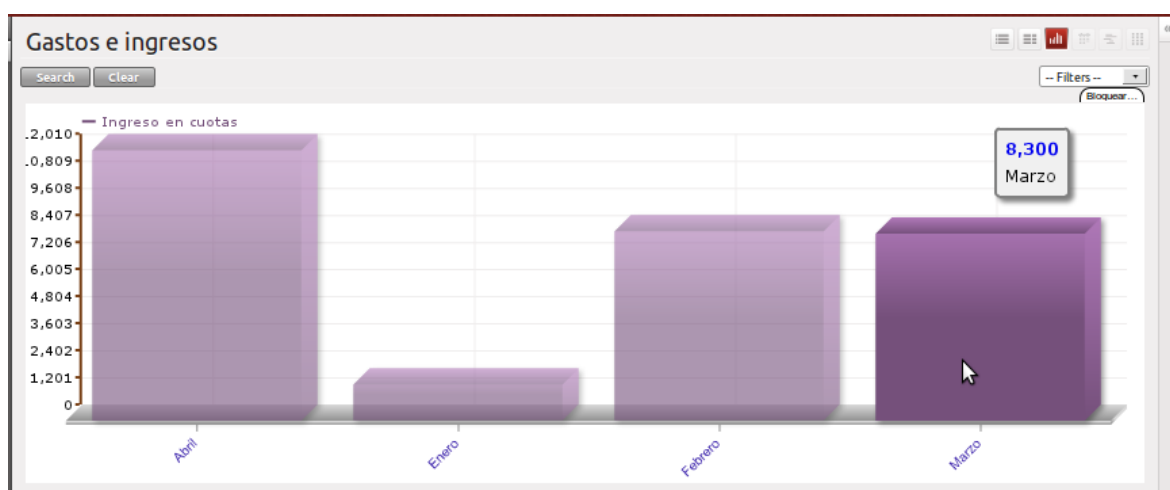


Figura 6.4: Agrupación de ventas por mes en un gráfico

Gastos e ingresos

Save Save & Edit Cancel

Mes:

Gasto en salarios:

Gasto en licencias:

Gasto no esperado:

Ingreso en cuotas:

Ingreso en publicidad:

Responsable datos:

Attachments Add

Customize

Manage Views

Customize Object

Other Options

Translate

Figura 6.5: Formulario del módulo

### 6.3. Una aplicación real: *ClientesBugginv2*

Este modulo contiene dos vistas, clientes y ventas. Clientes es la vista en la que se introducen los datos de los clientes. Los datos a introducir son los siguientes campos:

- Nombre
- Apellidos
- Email
- Telefono
- Domicilio
- Localidad
- Provincia
- Pais

Ventas es la vista donde se almacenan todas las ventas de la empresa. Con los siguientes campos.

- Fecha
- Cliente (este dato es un foreign key de la vista cliente)
- Precio

Para esta vista cabe la posibilidad de ver un gráfico de barras que muestra todos los clientes y la suma total de sus compras.

El código fuente de ambos módulos se puede encontrar en los anexos.



Figura 6.6: Relación de clientes

### 6.4. Patrones de diseño

Aplicar patrones de diseño como los indicados en [GHJV96] puede ser particularmente insidioso debido a la rigidez de la arquitectura sobre la cual trabajamos. Se pueden encontrar algunos ejemplos pero generalmente basta con definir operaciones básicas así como sus interacciones con la vista.

FECHA DE RECEPCION	CLIENTE	PRECIO
02/05/2013	Cliente1@cliente1.com	500.0000
01/05/2013	Cliente2@c2.com	400.0000
01/05/2013	Cliente1@cliente1.com	100.0000

Figura 6.7: Relación de ventas

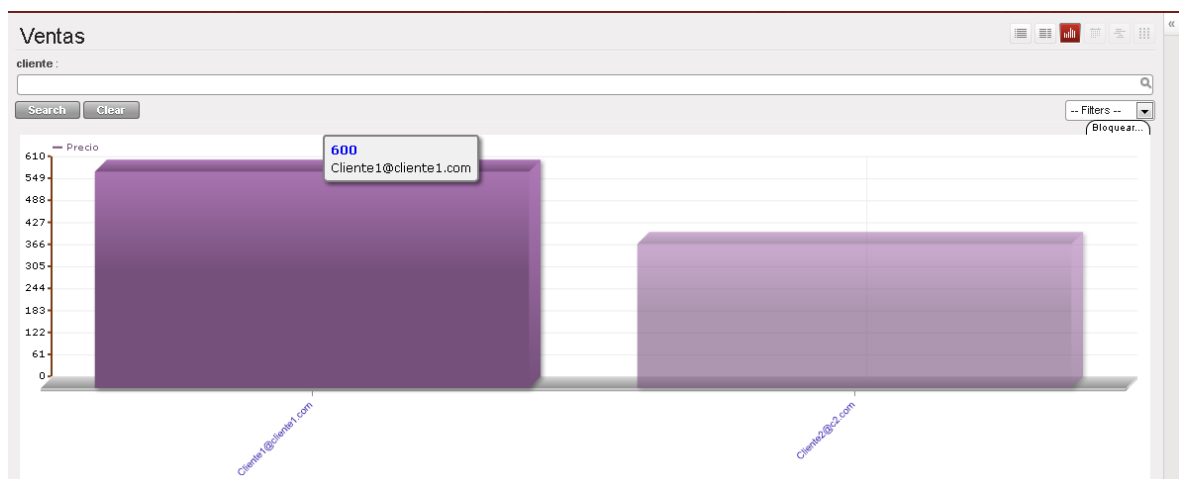


Figura 6.8: Gráfico de ventas por cliente

Nombre	Descripción
name	The name of the module in English.
version	The version of the module.
summary	Short description or keywords
description	The module description (text).
category	The category of the module
author	The author of the module.
website	URL of the website of the module.
license	The license of the module (default: AGPL-3).
depends	List of modules on which this module depends beside base.
data	List of .xml files to load when the module is installed or updated.
demo	List of additional .xml files to load when the module is installed or updated and demo flag is active.
installable	True or False. Determines whether the module is installable not.
auto_install	True or False (default: False). If set to “True“, the is a link module. It will be installed as soon as all its dependencies are installed.

Cuadro 6.1: Propiedades

# Pruebas

### 7.1. Framework de pruebas

Además de las pruebas basadas en YAML, OpenERP usa el framework de pruebas `unittest2` tanto para hacer pruebas a los paquetes del núcleo de OpenERP como a los paquetes externos. Las pruebas están divididas en tres grupos superpuestos:

- Un conjunto de pruebas que abarca todas las pruebas que puedan ejecutarse tras instalar un paquete. Este conjunto se conoce como *fast\_suite* y debe contener sólo las pruebas que se puedan ejecutar frecuentemente. En realidad la mayoría de las pruebas deberían ser consideradas lo suficientemente rápidas como para que se incluyeran en el conjunto *fast\_suite*. Un límite razonable de qué no es rápido serían las pruebas que durasen más de un minuto.
- Otro conjunto de pruebas llamado `checks` suministra pruebas de coherencia. Estas pruebas son condiciones invariantes que deben ser satisfechas en cualquier momento y se espera que se pasen en cualquier momento. Obviamente deben ejecutarse después de que un módulo se instale, pero también deben ejecutarse (pasar satisfactoriamente) después de una migración, actualización, etc.
- El tercer conjunto consta de todas las pruebas, las proporcionadas por los dos conjuntos anteriores y todas las que no se encuadran en los dos conjuntos anteriores.

Puesto que el conjunto de pruebas de coherencia ofrece una mayor garantía sobre el código y la estructura de la base de datos, las pruebas nuevas deben ser añadidas al conjunto `checks` siempre que sea posible. Dicho de otro modo, hay que evitar escribir pruebas que asuman que tanto los módulos como la base de datos están limpias y coherentes, hay que ser un poco enrevesado.

Como regla general al escribir un test hay que intentar añadirlo al conjunto `checks`. Si realmente se necesita que el módulo al que pertenece esté recién instalado, hay que añadirlo al *fast\_suite*. Finalmente, si no puede ser ejecutado en un tiempo aceptable, no hay que añadirlo explícitamente a ninguna lista.



## Capítulo 8

# Conclusiones

Tras unas intensas jornadas de trabajo tenemos una opinión bien formada en base a nuestra experiencia.

1. OpenERP es una herramienta muy versátil. La gran cantidad de módulos de casi cualquier ámbito hacen que se pueda personalizar la herramienta hasta el más mínimo detalle, no sólo con dichos módulos, si no pudiendo modificarlos a nivel de código para refinarlos a medida.
2. Los módulos son muy potentes y realizan todo el trabajo. No importa cuan complicados sean los procesos de negocio que soporte el módulo porque todo estará automatizado y se mantendrá coherente sin necesidad de que el usuario invierta tiempo o necesite conocer de forma exhausta todos esos procesos. Este punto es problemático en algunos casos puesto que todo puede ser excesivamente transparente al usuario dando la sensación de que el usuario hace demasiado poco.
3. La usabilidad de OpenERP aún no es comparable a su potencia. El limitado, pero rico, número de vistas de que dispone hace que en ocasiones no se encuentre la información que se necesita, por no hablar de la maraña de menús y filtros que suelen entorpecer más que ayudar a configurar el sistema.
4. La documentación es extensa pero no completa. Pese a la enorme cantidad de manuales, tutoriales y páginas web de información la mejor manera de aprender cómo se hacen los módulos es hacer ingeniería inversa a partir de uno ya terminado y completamente funcional puesto que hay muchos pequeños detalles que no se muestran explícitamente en ningún manual o tutorial y son imprescindibles a la hora de desarrollar cualquier mínima funcionalidad.
5. Versiones distintas necesitan módulos distintos. Las versiones de OpenERP no son compatibles entre sí y eso ha derivado en multitud de problemas en la instalación y desarrollo de los módulos. Hemos tenido que instalar todas las versiones en varios sistemas operativos para encontrar cual era la mejor combinación y poder desarrollar de manera eficiente. Ni siquiera los cambios de una versión a otra están claramente descritos y hay situaciones en la que la única metodología válida es la "prueba y error".

6. Necesidad de personal cualificado. Tanto para el desarrollo de módulos como para la explotación y uso de un sistema OpenERP es necesario personal cualificado y con experiencia que sea capaz de solventar todos los problemas que puedan surgir y cuya solución dista bastante de ser trivial, sólo habiéndose enfrentado a esos problemas con anterioridad y haber lidiado con ellos ofrece alguna garantía de solucionarlos en un tiempo prudencial.
7. Pese a la complejidad que supone definir cuál podría ser el coste total del desarrollo de un módulo en OpenERP los cálculos se han ajustado bastante a la realidad, en parte también por el plus que supone el no realizar ningún gasto en la explotación del sistema (en nuestro caso el presupuesto de explotar un sistema SAP aumentaría el presupuesto en más de 10.000 anuales).
8. Los problemas de compatibilidad, inexactitud de la documentación y nuestra propia falta de cualificación nos ha hecho perder un tiempo valiosísimo simplemente solucionando problemas que nada tienen que ver con el desarrollo de un módulo completamente funcional para OpenERP. Por estos motivos hemos ido reduciendo la funcionalidad de nuestro módulo a algo menos funcional pero que funciona y hemos intentado realizar un documento introductorio a OpenERP lo más completo y claro posible.
9. En el aspecto profesional nos hemos dado cuenta de la dificultad que puede suponer enfrentarse a un sistema completamente nuevo en el que no se tiene ni conocimientos ni experiencia previa y en que las ayudas (documentación, tutoriales, etc.) no son ni de lejos suficientes para lidiar con él. Esto puede ser un impedimento para continuar formándonos en esta tecnología o un incentivo para cubrir una necesidad de mercado que no todo el mundo puede cubrir por culpa de los problemas ya descritos.

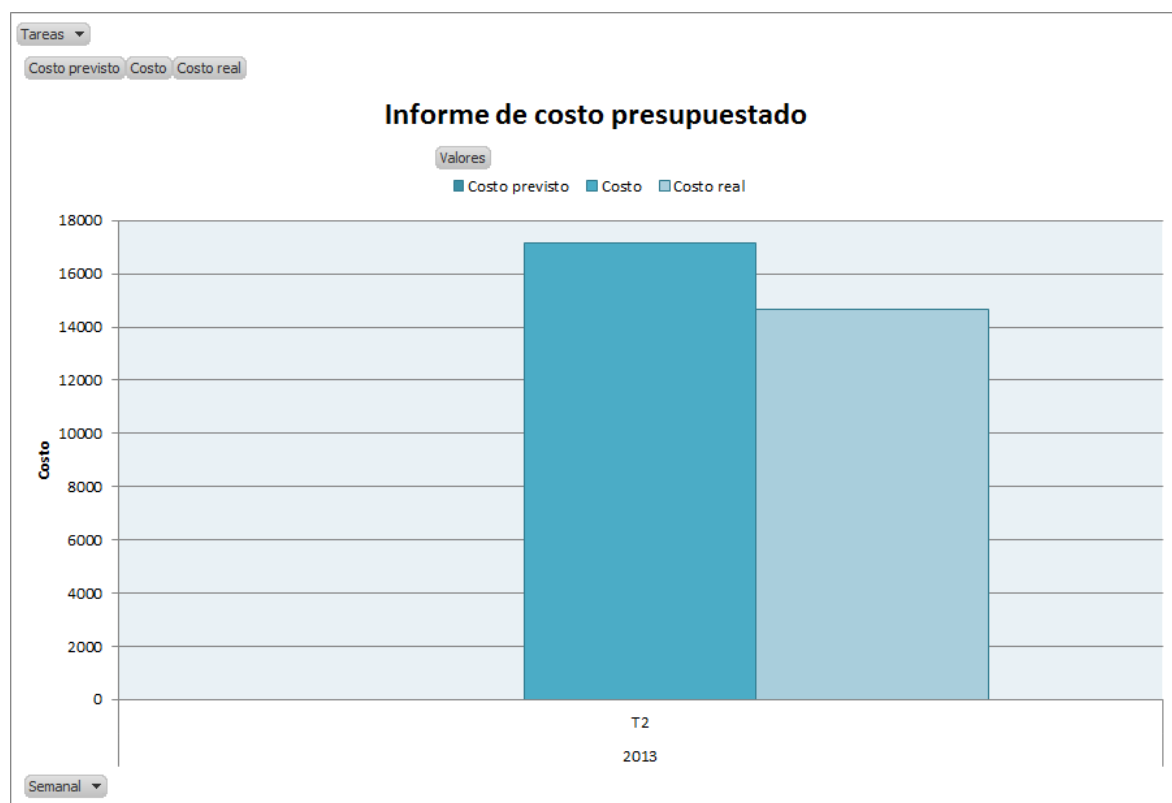


Figura 8.1: Costo presupuestado

ANEXOS

## Anexo A

# Manual de instalación

## A.1. Instalación en Ubuntu Linux

La primera opción sería descargar el paquete .deb de OpenERP e instalarlo, pero no nos permitiría tomar el control suficientemente de algunos aspectos como el lugar donde se instalará todo o hasta qué punto podremos personalizar la instalación por lo que optaremos por un modo más manual.

### A.1.1. Preparando el servidor

Primero vamos a instalar el paquete openssh-server para poder conectarnos remotamente y el paquete denyhosts para añadir un nivel de protección contra ataques de fuerza bruta. Podríamos añadir otros niveles para distintos ataques pero el objetivo es simplemente dejar claro que podemos añadir más, no hacerlo. Por ejemplo podríamos implementar un sistema de acceso por ssh basado en claves en vez de en contraseñas.

```
sudo apt-get install openssh-server denyhosts
```

Actualizamos para asegurarnos de que tenemos la última versión y todas las actualizaciones con:

```
sudo apt-get update sudo apt-get dist-upgrade
```

Tras reiniciar el servidor y comprobar que realmente podemos conectarnos vía ssh podemos pasar a instalar OpenERP.

### A.1.2. Crear el usuario OpenERP

Hay que crear un usuario para que sea el propietario de la aplicación y la ejecute puesto que no será una persona quien introduzca usuario y contraseña o arranque la aplicación, simplemente el usuario OpenERP dará un servicio. En Ubuntu los identificadores de usuario cuyo UID están por debajo de 1000 no tienen terminal y su login está deshabilitado.

```
sudo adduser -system -home=/opt/openerp -group openerp
```

Hemos dejado el home en /opt/openerp que será donde estará el código del servidor. Por supuesto que este directorio podría ser cualquier otro pero durante todo este manual todos los comandos están escritos como si el home fuera /opt/openerp, tenga en cuenta esto si va a cambiar el directorio.

Si queremos logearnos como el usuario openerp recién creado, pese a que no tiene login ni terminal debemos ejecutar el comando:

```
sudo su - openerp -s /bin/bash
```

### A.1.3. Instalar y configurar la base de datos PostgreSQL

Primero instalar el servidor de la base de datos con el comando:

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

Ahora hay que logearse como el usuario de postgres para tener los permisos necesarios.

```
sudo su - postgres
```

Hay que crear el usuario para la base de datos, así OpenERP tendrá permisos para conectarse a PostgreSQL y crear o eliminar bases de datos. Recuerde la contraseña puesto que será necesaria más adelante.

```
createuser -createdb -username postgres -no-createrole -no-superuser -  
pwprompt openerp
```

Tras escribir la contraseña salimos de la cuenta de usuario de postgres con

```
exit
```

### A.1.4. Instalar las bibliotecas necesarias de Python

Para eso sólo hay instalar todos los paquetes siguientes.

```
sudo apt-get install python-dateutil python-docutils python-feedparser  
python-gdata python-jinja2 python-ldap python-libxslt1 python-lxml python-  
mako python-mock python-openid python-psycpg2 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
```

Con esto todas las dependencias de OpenERP debes de estar ya satisfechas.

### A.1.5. Instalar el servidor OpenERP

En la dirección <http://nightly.openerp.com/7.0/nightly/> podemos encontrar todos los links a todas las descargas (paquetes .deb, .rpm y .exe incluidos). Nuestra opción será descargarnos el archivo comprimido donde están los fuentes para descomprimirlo en el home del usuario que hemos creado antes. Los comandos serían:

```
wget http://nightly.openerp.com/7.0/nightly/src/openerp-7.0-latest.tar.gz  
cd /opt/openerp  
sudo tar xvf /openerp-7.0-latest.tar.gz
```

Ahora hay que cambiar el propietario de todos los ficheros al usuario y grupo creados antes:

```
sudo chown -R openerp: *
```

Y para terminar:

```
sudo cp -a openerp-7.0 server
```

### A.1.6. Configurar la aplicación OpenERP

La configuración por defecto sólo necesita un par de cambios y cambiar el propietario de los archivos así como sus permisos:

```
sudo cp /opt/openerp/server/install/openerp-server.conf /etc/ sudo chown  
openerp: /etc/openerp-server.conf sudo chmod 640 /etc/openerp-server.conf
```

Para permitir que el servidor OpenERP arranque al inicio hay que cambiar del fichero `/etc/openerp-server.conf`:

```
sudo nano /etc/openerp-server.conf
```

la línea `db_password=False` por `db_password=pass`; donde `pass` es la contraseña que usamos en el paso 3.

Igualmente añadir la línea `logfile=/var/log/openerp/openerp-server.log` al final del fichero. Salir del editor y comprobar si el servidor está funcionando de la siguiente manera.

```
sudo su - openerp -s /bin/bash
/opt/openerp/server/openerp-server
```

Salimos del usuario openerp con `exit`

### A.1.7. Instalar el script de arranque

Para terminar instalaremos un script que permita al usuario arrancar y apagar el servidor automáticamente y arracar la aplicación con el usuario correcto. Copiamos y pegamos el archivo `openerp-server` en `/etc/init.d/` y le damos los permisos adecuados y cambiamos su propietario.

```
sudo chmod 755 /etc/init.d/openerp-server
sudo chown root: /etc/init.d/openerp-server
```

En el fichero de configuración hay una línea adicional para el archivo de logs. Hay que crear el directorio primero para que el servidor tenga algún sitio donde escribir.

```
sudo mkdir /var/log/openerp
sudo chown openerp:root /var/log/openerp
```

### A.1.8. Probar el servidor

Para arrancar el servidor sólo hay que ejecutar el comando:

```
sudo /etc/init.d/openerp-server start
```

Ahora podrían verse los archivos de log y si el servidor ha arrancado.

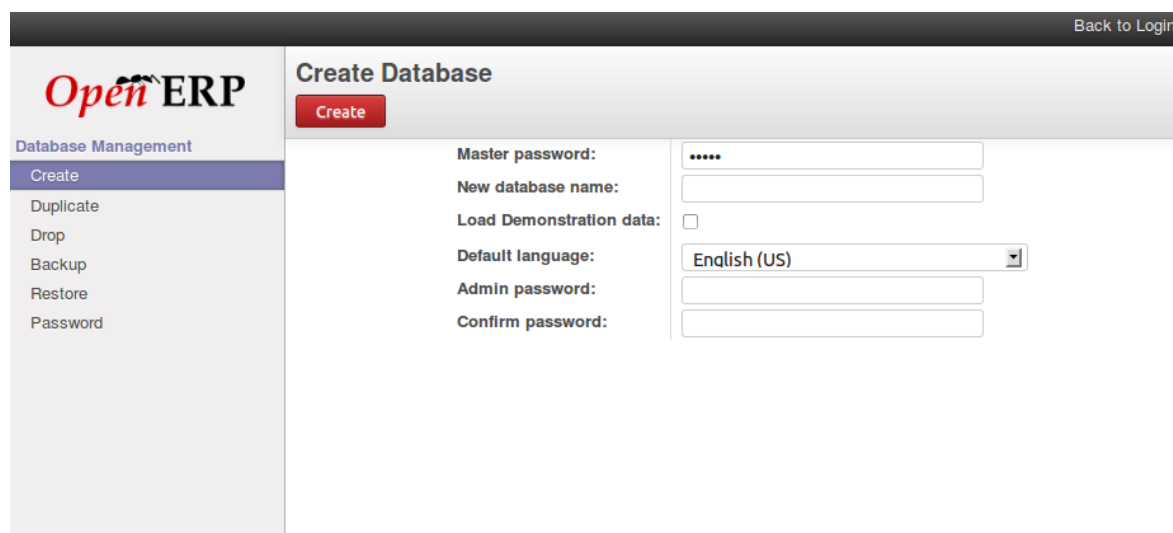
```
less /var/log/openerp/openerp-server.log
```

Ahora abrimos el navegador y vamos a la página

`http://IP_o_dominio.com:8069`

La contraseña por defecto del usuario admin es admin





The screenshot shows the 'Create Database' page of the OpenERP web interface. On the left is a sidebar with the 'OpenERP' logo and a 'Database Management' menu containing options like 'Create', 'Duplicate', 'Drop', 'Backup', 'Restore', and 'Password'. The main area is titled 'Create Database' and features a red 'Create' button. Below this, there are several form fields: 'Master password' (masked with dots), 'New database name' (empty), 'Load Demonstration data' (checkbox), 'Default language' (dropdown menu set to 'English (US)'), 'Admin password' (empty), and 'Confirm password' (empty). A 'Back to Login' link is located in the top right corner.

Figura A.1: Login de usuario

### A.1.9. Automatizar el arranque y parada de OpenERP

Para que el script arranque y pare automáticamente con el servidor de Ubuntu sólo hay que ejecutar:

```
sudo update-rc.d openerp-server defaults
```

## A.2. Instalación en Windows

La instalación en Windows es trivial, sólo hay que descargarse el archivo `openerp-allinone-setup-7.0-latest.exe` de la dirección <http://nightly.openerp.com/7.0/nightly/exe/> y seguir los pasos del asistente de instalación.

En las páginas oficiales de comunidades hispanohablantes o en la página principal de OpenERP se puede encontrar la documentación necesaria así como los binarios para instalar el ERP en cualquier sistema.

[Ope13]

## Anexo B

# Código fuente de la herramienta *Bugginv5*

```
import bugginv5
```

Listado B.1: Archivo `__init__.py`

```
{
    'name': 'Sistema Gestion Buggin and Debuggin',
    'version': '0.1',
    'category': 'Tools',
    'description': """Modulo para la gestion de nuestra empresa.""",
    'author': 'Buggin and Debuggin',
    'website': 'www.BugginAndDebuggin.com',
    'depends': ['base'],
    'init_xml': [],
    'update_xml': ['bugginv5_view.xml'],
    'demo_xml': [],
    'installable': True,
}
```

Listado B.2: Archivo `__openerp__.py`

```
from osv import osv, fields

class gastosingresos(osv.osv):
    _name = "bugginv5.gastosingresos"
    _description = "Gestion de gastos e ingresos de la mejor
        empresa del mundo"
    _columns = {
        'mes': fields.char('Mes',size=256,required=True),
        'gasto_salarios': fields.float('Gasto en salarios',
            required=True, digits=(14,4)),
        'gasto_licencias': fields.float('Gasto en licencias',
            required=True, digits=(14,4)),
        'gasto_extra': fields.float('Gasto no esperado',
            required=True, digits=(14,4)),
        'ingreso_cuota': fields.float('Ingreso en cuotas',
            required=True, digits=(14,4)),
        'ingreso_publicidad': fields.float('Ingreso en
            publicidad', required=True, digits=(14,4)),
        'responsable':fields.char('Responsable datos',
            required=True, size=256)
    }

gastosingresos()
```

Listado B.3: Archivo bugginv5.py

```

<?xml version="1.0"?>
<openerp>
  <data>
    <record model="ir.ui.view" id="gastosingresos_form">
      <field name="name">Gastos e ingresos</field>
      <field name="model">bugginv5.gastosingresos</field>
      <field name="type">form</field>
      <field name="arch" type="xml">
        <form string="Gastos e ingresos">
          <field name="mes"/>
          <field name="gasto_salarios"/>
          <field name="gasto_licencias"/>
          <field name="gasto_extra"/>
          <field name="ingreso_cuota"/>
          <field name="ingreso_publicidad"/>
          <field name="responsable"/>
        </form>
      </field>
    </record>
    <record model="ir.ui.view" id="gastosingresos_tree">
      <field name="name">Gastos e ingresos</field>
      <field name="model">bugginv5.gastosingresos</field>
      <field name="type">tree</field>
      <field name="arch" type="xml">
        <tree string="Gastos e ingresos">
          <field name="mes"/>
          <field name="gasto_salarios"/>
          <field name="gasto_licencias"/>
          <field name="gasto_extra"/>
          <field name="ingreso_cuota"/>
          <field name="ingreso_publicidad"/>
          <field name="responsable"/>
        </tree>
      </field>
    </record>
    <record model="ir.ui.view" id="gastosingresos_graph">
      <field name="name">Grafico mes cuota</field>
      <field name="model">bugginv5.gastosingresos</field>
      <field name="type">graph</field>
      <field name="arch" type="xml">
        <graph string="Ingresos" orientation="vertical" type="bar">
          <field name="mes"/>
          <field name="ingreso_cuota"/>
        </graph>
      </field>
    </record>
    <record model="ir.actions.act_window" id="action_gastosingresos">
      <field name="name">Gastos e ingresos</field>
      <field name="res_model">bugginv5.gastosingresos</field>
      <field name="view_type">form</field>
      <field name="view_mode">tree,graph,form</field>
    </record>
    <menuitem name="Bugginv5/Gastosingresos/GastosIngresosINFO" id="
      menu_bugginv5_gastosingresos" action="action_gastosingresos"/>
  </data>
</openerp>

```

Listado B.4: Vista de bugginv5

## Anexo C

# Código fuente de la herramienta *ClientesBugginv2*

```
import clientesbugginv2
```

Listado C.1: Archivo `__init__.py`

```
{
    'name': 'Sistema Gestion Buggin and Debuggin',
    'version': '0.1',
    'category': 'Tools',
    'description': """Modulo para la gestion de nuestra empresa.""",
    'author': 'Buggin and Debuggin',
    'website': 'www.BugginAndDebuggin.com',
    'depends': ['base'],
    'init_xml': [],
    'update_xml': ['clientesbugginv2_view.xml'],
    'demo_xml': [],
    'installable': True,
}
```

Listado C.2: Archivo `__openerp__.py`

```
from osv import osv, fields

class cliente(osv.osv):
    _name = "clientesbugginv2.cliente"
    _rec_name = 'email'
    _description = "Clientes"
    _columns = {
        'nombre':fields.char('Nombre',size=64),
        'apellidos':fields.char('Apellidos',size=64),
        'email':fields.char('Correo electronico',size=64),
        'telefono':fields.char('Telefono', size=20),
        'domicilio':fields.char('Domicilio', size=64),
        'localidad':fields.char('Localidad', size=64),
        'provincia':fields.many2one('res.country.state', 'Provincia', ondelete='set null'),
        'pais': fields.many2one('res.country', 'Pais', ondelete='set null', select=True)
    }

cliente()

class ventas(osv.osv):
    _name = "clientesbugginv2.ventas"
    _description = "Gestion de ventas"
    _columns = {
        'fecha':fields.date('Fecha de recepcion',size=64),
        'cliente':fields.many2one('clientesbugginv2.cliente', 'cliente', ondelete='cascade', select=True),
        'precio': fields.float('Precio', required=True, digits=(14,4))
    }

ventas()
```

Listado C.3: Archivo clientesbugginv2.py

```
<?xml version="1.0"?>
<openerp>
  <data>
    <record model="ir.ui.view" id="cliente_form">
      <field name="name">Clientes</field>
      <field name="model">clientesbugginv2.cliente</field>
      <field name="type">form</field>
      <field name="arch" type="xml">
        <form string="clientesbugginv2.cliente">
          <field name="nombre"/>
          <field name="apellidos"/>
          <field name="email"/>
          <field name="telefono"/>
          <field name="domicilio"/>
          <field name="localidad"/>
          <field name="provincia"/>
          <field name="pais"/>
        </form>
      </field>
    </record>
    <record model="ir.ui.view" id="cliente_tree">
      <field name="name">Clientes</field>
      <field name="model">clientesbugginv2.cliente</field>
      <field name="type">tree</field>
      <field name="arch" type="xml">
        <tree string="clientesbugginv2.cliente">
          <field name="nombre"/>
          <field name="apellidos"/>
        </tree>
      </field>
    </record>
  </data>
</openerp>
```

Listado C.4: Vista de clientesbuggin

```

<record model="ir.actions.act_window" id="action_cliente">
  <field name="name">Cliente</field>
  <field name="res_model">clientesbugginv2.cliente</field>
  <field name="view_type">form</field>
  <field name="view_mode">tree,form</field>
</record>
<record model="ir.ui.view" id="ventas_form">
  <field name="name">Ventas</field>
  <field name="model">clientesbugginv2.ventas</field>
  <field name="type">form</field>
  <field name="arch" type="xml">
    <form string="clientesbugginv2.ventas">
      <field name="fecha"/>
      <field name="cliente"/>
      <field name="precio"/>
    </form>
  </field>
</record>
<record model="ir.ui.view" id="ventas_tree">
  <field name="name">Ventas</field>
  <field name="model">clientesbugginv2.ventas</field>
  <field name="type">tree</field>
  <field name="arch" type="xml">
    <tree string="clientesbugginv2.ventas">
      <field name="fecha"/>
      <field name="cliente"/>
      <field name="precio"/>
    </tree>
  </field>
</record>

<record model="ir.ui.view" id="ventas_graph">
  <field name="name">Grafico ventas-cliente</field>
  <field name="model">clientesbugginv2.ventas</field>
  <field name="type">graph</field>
  <field name="arch" type="xml">
    <graph string="VentasCliente" orientation="vertical" type="bar">
      <field name="cliente"/>
      <field name="precio"/>
    </graph>
  </field>
</record>
<record model="ir.actions.act_window" id="action_ventas">
  <field name="name">Ventas</field>
  <field name="res_model">clientesbugginv2.ventas</field>
  <field name="view_type">form</field>
  <field name="view_mode">tree,form,graph</field>
</record>
<menuitem name="Clientes Ventas" id="menu_clientes_ventas"
  sequence="20"/>
<menuitem name="Cliente" id="menu_clientes_ventas_cliente" parent=
  "menu_clientes_ventas" sequence="10" action="action_cliente"/>
<menuitem name="Ventas" id="menu_clientes_ventas_ventas" parent="
  menu_clientes_ventas" sequence="20" action="action_ventas"/>
</data>
</openerp>

```

Listado C.5: Vista de clientesbuggin (continuación)



## Anexo D

# Definición de la empresa

## D.1. Componente organizacional

### D.1.1. Organigrama

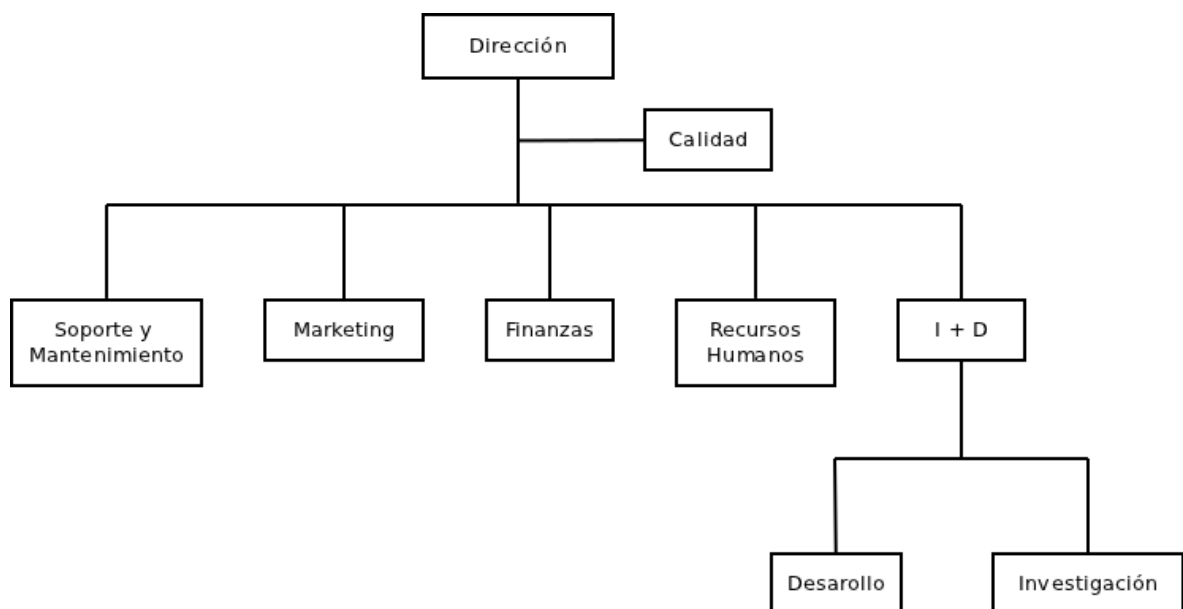


Figura D.1: Organigrama de la empresa

Distinguiéndose los siguientes cargos:

- **Director:** Persona encargada de la organización general y control de trabajo de los diferentes departamentos de la empresa. Raúl Reguillo
- **Recursos Humanos:** Se trata del departamento encargado de que en la empresa se encuentre en cada momento no sólo con la gente necesaria en cuantía sino también de forma cualitativa. Su trabajo también es conseguir que el trabajador se sienta gratificado, valorado y seguro en su puesto de trabajo. Raúl Reguillo
- **Marketing:** Es el área que hace que nuestra empresa tenga una imagen externa. Se encarga de las campañas publicitarias y el soporte a la publicidad de empresas externas como medio de financiación de la nuestra. Javier García

- **Finanzas:** Se ocupa de los estudios de adquisiciones de material, rentabilidad, inversiones, liquidez, reinversiones, de estar al día de las fluctuaciones en el mercado, de conocer las tendencias económicas y su implicaciones. Una función básica es la de planeación, pronóstico, cálculo para mantener el equilibrio económico de la empresa. Cristian del Cerro
- **I+D:** Es la sección empresarial que destina sus fuerzas al desarrollo de los servicios o productos que ofrece la empresa. El departamento de investigación es la cabeza pensante de nuestra empresa, y trata de mejorar nuestro producto estudiando nuevas tecnologías y analizando los requerimientos de los usuarios. Raúl Reguillo, Laura del Río, Javier García y Cristian del Cerro
- **Calidad:** Este departamento se encarga de que los desarrollos funcionen de forma correcta y continua. La comunicación con el usuario es continua, de tal forma que se atiende a las necesidades y sugerencias del usuario en todo momento. Javier García y Laura del Río

### D.1.2. Acerca de la empresa Buggin' & Debuggin'

#### Descripción

B&D es una pyme multidisciplinar dedicada a ofrecer soluciones software en cualquier ámbito relacionado con las TIs. El objetivo es desarrollar y mantener servicios y aplicaciones de carácter general, ofrecer asistencia personalizada, cobertura y mantenimiento.

#### Target

Nos dirigimos a:

- PyMEs que requieren de aplicaciones concretas y/o servicio técnico
- EMpresas que requieren aplicaciones para una cadena de productos (como pueda ser el caso de restaurantes, hoteleras, oficinas, etc.) y/o servicio técnico
- Particulares que requieran de servicio técnico
- Desarrollo de aplicaciones para nuevas tecnologías como smartphones, tablets, etc

#### Horarios de trabajo

- De lunes a jueves:
  - Entrada: desde las 7 a las 9 de la mañana
  - Núcleo central de trabajo: desde las 9 a las 14 y de las 16 a las 18 (7 horas)
  - Salida: desde las 18 a las 20 de la noche
- Viernes:
  - Salida de 14 a 16

### D.1.3. Plan de inversión

#### Gastos recurrentes

Todos los precios están en Euros salvo que se indique lo contrario:

Ver Anexos I, II y III

#### Costes sociales de los trabajadores

Salario bruto de 1,500 euros al mes con las pagas extras prorrateadas, y una base de cotización de 1,450 euros al mes. El coste por seguridad social para la empresa asciende a 483,33 euros mensuales, y el coste del trabajador sube a 1,983 euros mensuales.

#### Subtotal

- Costes sociales: 1983
- Alquiler de dominios: 12
- Apps for Business Google: 16
- Subcontratas:
  - Línea de Internet + teléfono: 20
  - Gastos variables de luz, agua y gas: 150 aproximadamente
  - Servicio de prensa: 30
- Alquiler de la oficina: 650
- Otros: 300
- **Total:** 3.161

#### Gastos no recurrentes

Todos los precios están en Euros salvo que se indique lo contrario:

Ver Anexos I, II y III

#### Subtotal

- Hardware:
  - 4 Equipos Dell latitude:  $4 * 871,00 = 3484,00$
  - Servidor Dell Poweredge T110 II: 1623,00
  - Monitor Panorámico Dell E2011H: 111,20
  - Pizarra interactiva: 846,00
  - Video Proyector: 317,23
  - Multifunción BROTHER: 109,90

- Software y licencias:
  - Licencia Visual Paradigm: 699,00 (U.S. Dollars)
- Material de oficina: 402,56
- Decoración y mobiliario: 5267,68
- Otros: 692,53
- **Total:** 13.392,33

#### **D.1.4. Estrategia de marketing y publicidad**

- Presencia en las principales redes sociales: twitter, facebook
- Página web de empresa

## **D.2. Proyectos OpenERP**

Adjunto a este documento se entrega un archivo de *MS Project* en el que se detalla cada parte de la planificación, costes y fechas seguidas a la hora de elaborar este proyecto software. Los costes de este proyecto también vienen reflejados en informes paralelos generados por esta herramienta. Con esto se quiere remarcar que los costes totales de la organización de la empresa son paralelos al coste de desarrollo de un proyecto particular en OpenERP, en el que ya intervienen factores temporales, costo de recursos, planificación, etc.

# Bibliografía

- [GHJV96] E. Gamma, R. Helm, R. Johnson, y J. Vlissides. *Design Patterns*. Addison-Wesley, 1996.
- [Ope13] OpenERP. *OpenERP Website*. <https://www.openerp.com>, 2013.
- [Wik13] Wikipedia. *Wikipedia*. [http://en.wikipedia.org/wiki/Enterprise\\_resource\\_planning](http://en.wikipedia.org/wiki/Enterprise_resource_planning), 2013.

Este documento fue editado y tipografiado con  $\text{\LaTeX}$   
empleando la clase **arco-pfc** que se puede encontrar en:  
[https://bitbucket.org/arco\\_group/arco-pfc](https://bitbucket.org/arco_group/arco-pfc)

[Respetar esta atribución al autor]